



Recovery Act: Geothermal Data Aggregation: Submission of Information into the National Geothermal Data System

Final Report: DOE Project DE-EE0002852

Date Submitted: June 24, 2014

Principal Investigator: David D. Blackwell

Sub Principal Investigators: Michael Golm, Bruce Cutright, Will Gosnold, Martin Kay, Seiichi Nagihara, Estela Smith, Jefferson Tester

SMU Geothermal Laboratory: Cathy Chickering and Maria Richards

Siemens Corporate Technology: Dmitriy Fradkin, Christoph Kuhmuench, Rajanikanth Tanikella

Notice: This report is distributed without limitation. When referenced in part or in whole, include the date of this report and Southern Methodist University Geothermal Laboratory

Project Abstract

The National Geothermal Data System (NGDS) is a Department of Energy funded effort to create a single cataloged source for a variety of geothermal information through a distributed network of databases made available via web services. The NGDS will help identify regions suitable for potential development and further scientific data collection and analysis of geothermal resources as a source for clean, renewable energy. A key NGDS repository or 'node' is located at Southern Methodist University developed by a consortium made up of:

- SMU Geothermal Laboratory
- Siemens Corporate Technology, a division of Siemens Corporation
- Bureau of Economic Geology at the University of Texas at Austin
- Cornell Energy Institute, Cornell University
- Geothermal Resources Council
- MLKay Technologies
- Texas Tech University
- University of North Dakota.

The focus of resources and research encompass the United States with particular emphasis on the Gulf Coast (on and off shore), the Great Plains, and the Eastern U.S. The data collection includes the thermal, geological and geophysical characteristics of these area resources. Types of data include, but are not limited to, temperature, heat flow, thermal conductivity, radiogenic heat production, porosity, permeability, geological structure, core geophysical logs, well tests, estimated reservoir volume, in situ stress, oil and gas well fluid chemistry, oil and gas well information, and conventional and enhanced geothermal system related resources. Libraries of publications and reports are combined into a unified, accessible, catalog with links for downloading non-copyrighted items. Field notes, individual temperature logs, site maps and related resources are included to increase data collection knowledge. Additional research based on legacy data to improve quality increases our understanding of the local and regional geology and geothermal characteristics. The software to enable the integration, analysis, and dissemination of this team's NGDS contributions was developed by Siemens Corporate Technology.

Final Report Table of Contents

Recovery Act: Geothermal Data Aggregation: Submission of Information into the National Geothermal Data System Final Report: DOE Project DE-EE0002852	0

Final Report Table of Contents.....	i - ix

Main Body of Report

Organization of Final Report.....	1
I. Executive Summary.....	2
II. Review of Statement of Project Objectives by Task.....	5
A. PROJECT OBJECTIVES.....	5
B. PROJECT SCOPE.....	6
C. TASKS TO BE PERFORMED	6
Phase I. Data Retrieval, Collection, Development and Quality Analysis.....	6
Phase II. Transfer and Validation of Information to Data System	17
Phase III. Execution of Data Maintenance Sustainability Plan	19
III. Summary of Accomplishments, Lessons Learned, Changes in Approach, & Opportunities for Further Study.....	23
A. Key Achievements	23
Data Collection, Analysis, and Aggregation.....	23
Data Quality Advances	26
New Data Interpretations.....	26
Expanding the knowledge base of geothermal energy research	26
Improved Access to Publications.....	26
B. Changes in Approach.....	27
Response to Ongoing Operation Requirement	27
Data Formats Offered	27
C. Lessons Learned.....	28
1. Evaluate strengths and weaknesses and plan accordingly	28
2. Plan for problems before they arise.....	28
3. Architecture should not change over the course of this type of project	29
4. Be wary of assumptions.....	29
5. Require data submitters to equate ‘their’ fields to standardized field names	29
6. Mapping to Content Models requires both knowledge of the raw data and familiarity with the content models.....	29
7. Scanning a library of publications was difficult for non-librarians	30
8. Be creative and flexible	31
9. A deadline for data submitter validation would have been helpful	31

10. Bi-weekly project updates with the DOE were valuable.....	31
11. Data Quality Assessment is never ‘done’	31
12. Metadata submission should be simplified.....	31
13. Overlapping data from State Geological Surveys needs further review.....	32
14. It is easy to work with people whom you like and respect.....	32
D. Opportunities for Additional Research or Activities.....	32

Appendices A - I

Note: Each Appendix is paginated separately from the main report body.

Refer to "Organization of Final Report" on page 1 for a brief description of each appendix.

If viewing the report as a PDF document, note that 'bookmarking' is enabled.

Appendix A: Links to Datasets on http://geothermal.smu.edu.....	Appendix A: 1
Copy of DownloadFilesButtonPage.htm.....	Appendix A: 1
GeoServer Links for each Dataset.....	Appendix A: 7
Aggregated Well Data.....	Appendix A: 8
Aggregated Borehole Temperature Observation Content Model Format	Appendix A: 9
Aggregated Fluid Flux Injection and Disposal Content Model Format.....	Appendix A: 10
Aggregated Geologic Reservoir in Content Model Format.....	Appendix A: 11
Aggregated Heat Flow Observation in Content Model Format.....	Appendix A: 12
Aggregated Radiogenic Heat Generation in Content Model Format.....	Appendix A: 13
Aggregated Thermal Conductivity Observation Content Model Format.....	Appendix A: 14
Aggregated Well Fluid Production Observation Content Model Format	Appendix A: 15
Aggregated Well Header Observation in Content Model Format.....	Appendix A: 16
Aggregated Well Log Observation in Content Model Format.....	Appendix A: 17
Aggregated Well Test Observation in Content Model Format.....	Appendix A: 18
SMU Heat Flow Data from Equilibrium Logs.....	Appendix A: 19
SMU Heat Flow Data from BHT.....	Appendix A: 20
BEG Master Wells Database.....	Appendix A: 21
BEG Geologic Reservoir Analysis.....	Appendix A: 22
Cornell NY and PA Heat Flow Database	Appendix A: 23
Texas Tech Off Shore Wells and BHT	Appendix A: 24
UND Radiogenic Heat Generation	Appendix A: 25
UND Thermal Conductivity.....	Appendix A: 26
UND Temperature at Depth.....	Appendix A: 27

Appendix B: List of Publications & Presentations from Consortium Team Members over Course of Project

Appendix B Table of Contents	Appendix B: 1

Graduate Theses and/or Dissertations, Dec. 2011 - May 2014.....	Appendix B: 2
Meetings, Papers and Conference Proceedings, 2010 - 2014	Appendix B: 3
Other Professional Publications	Appendix B: 14

Appendix C: User Manual for Use of GTDA

User Manual Table of Contents	Appendix C: ii
1 Preface	Appendix C: 3
1.1 Purpose and Audience	Appendix C: 3
1.2 Document Roadmap	Appendix C: 3
2 System Context	Appendix C: 4
3 Description of Individual User Screens	Appendix C: 5
3.1 Accessing Data	Appendix C: 5
3.2 Login Screen	Appendix C: 5
3.3 Map Search Page	Appendix C: 7
3.3.1 Filtering	Appendix C: 8
3.3.2 Accessing Data	Appendix C: 9
3.4 Publications Catalog	Appendix C: 13
3.5 Histograms	Appendix C: 14
3.6 User Management	Appendix C: 15
3.7 Submit Datasets	Appendix C: 17
3.7.1 Publish dialog (activated when publish button is clicked)	Appendix C: 21
3.8 Submit Related Resources	Appendix C: 21
3.8.1 Bibliography Upload	Appendix C: 21
3.8.2 Publications Upload Page	Appendix C: 23
3.8.3 Related Resources Submission	Appendix C: 25
3.8.3.1 Related Resources	Appendix C: 27
3.8.3.2 Field Notes	Appendix C: 27
3.8.3.3 TD Curves	Appendix C: 28
4 Terms.....	Appendix C: 29
4.1 Acronyms, and Abbreviations.....	Appendix C: 29
4.2 Glossary of Geological Terms	Appendix C: 29

Appendix D: Systems Administration Manual for GTDA

Systems Administration Manual Table of Contents	Appendix D: i
Preface.....	Appendix D: 3
1.1 Purpose and Audience.....	Appendix D: 3
1.2 Document Roadmap.....	Appendix D: 3
2 System Context.....	Appendix D: 4
3 Component Model.....	Appendix D: 5
3.1 Dynamic Views.....	Appendix D: 5

3.1.1 Use Case: Load Index Page	Appendix D: 5
3.1.2 Use Case: Filter Page	Appendix D: 6
3.1.3 Use Case: Select Area in Map & get details about 1 Well	Appendix D: 7
3.2 Component Descriptions: Client Side.....	Appendix D: 8
3.2.1 Ext JS v4.0.7	Appendix D: 8
3.2.2 OpenId client.	Appendix D: 9
3.2.3 OpenLayers v.2.12.	Appendix D: 9
3.2.4 GTDA Client.....	Appendix D: 10
3.3 Component Descriptions: Server Side	Appendix D: 14
3.3.1 Tomcat Application Server v6.0.24.	Appendix D: 14
3.3.2 OpenId Server.	Appendix D: 15
3.3.3 Shiro	Appendix D: 15
3.3.4 ESRI Geoportal Server v1.2.4.	Appendix D: 15
3.3.5 GeoServer v.2.4.....	Appendix D: 16
3.3.6 Jetty v9.1.	Appendix D: 17
Apache Solr v4.6.0.....	Appendix D: 18
3.3.7 GTDA Server.	Appendix D: 18
3.4 Database	Appendix D: 19
3.4.1 Entities	Appendix D: 20
3.4.2 GTDA database schemes	Appendix D: 22
3.5 Connectors.	Appendix D: 22
4 System Administration and Maintenance.....	Appendix D: 23
4.1 Passwords	Appendix D: 23
4.2 Starting and Stopping the System	Appendix D: 23
4.3 Backup and Restore	Appendix D: 24
4.3.1 What to Backup.	Appendix D: 24
4.3.1.1 Data.	Appendix D: 24
4.3.1.2 Application.....	Appendix D: 25
4.3.2 How to back up	Appendix D: 25
4.3.2.1 Backup options for files and folders	Appendix D: 26
4.3.2.2 Backing up the database	Appendix D: 26
4.3.3 How to restore data	Appendix D: 26
4.3.4 Commands to check the disk space	Appendix D: 27
4.4 Regeneration of CSW Entries..	Appendix D: 27
4.4.1 Trigger Indexing of Publication data..	Appendix D: 28
4.5 Location of Data and Components.....	Appendix D: 28
4.5.1 Tomcat	Appendix D: 28
4.5.2 Postgres	Appendix D: 29
4.5.3 SOLR.....	Appendix D: 32
4.6 GeoServer.	Appendix D: 33
4.7 Geoportal.	Appendix D: 33
4.8 GTDA Application.....	Appendix D: 33
4.9 Access to Gray Box about help documentation.....	Appendix D: 34
4.10 Documents..	Appendix D: 35
4.11 OS Maintenance, Backup and Restore.....	Appendix D: 35
4.11.1 OS Maintenance.	Appendix D: 35
4.11.1.1 OS Updates	Appendix D: 35
4.11.1.2 OS Backups.....	Appendix D: 36
4.11.1.3 OS Restores.....	Appendix D: 37
4.11.1.4 Alternatives	Appendix D: 38
4.11.1.4.1 Activate Rollback Feature of Yum.	Appendix D: 38

4.11.1.4.2 Install Backup tool	Appendix D: 38
5 Appendix 1: Terms and Glossary	Appendix D: 39
5.1 Acronyms and Abbreviations.....	Appendix D: 39
5.2 Glossary of Geological Relate Terms	Appendix D: 40
6 Appendix 2: Database Schema.	Appendix D: 42
 Appendix E: Sustainability Plan for the SMU Node of the NGDS	 Appendix E: 1
Appendix E Table of Contents	Appendix E: 2
Introduction.....	Appendix E: 3
Geothermal data submitted by the 7 research institutions	Appendix E: 4
Software Developed by Siemens for the Project	Appendix E: 7
Server Environment (system software and hardware)	Appendix E: 8
Security.....	Appendix E: 8
Physical Security.....	Appendix E: 8
System Security	Appendix E: 8
Backups.....	Appendix E: 9
Hardware and Software Routine Maintenance	Appendix E: 9
Disaster Recovery Plans	Appendix E: 9
Call Center and Customer Support Functions.....	Appendix E: 10
System Administration (Non-routine)	Appendix E: 10
Help-Desk.....	Appendix E: 10
Network Connectivity.....	Appendix E: 10
Support Agreement	Appendix E: 11
Data Residing on Non-SMU Servers	Appendix E: 12
Southern Methodist University Computing and Communications Policy	Appendix E: 13
 Appendix F: Description of Contents of the Hard Drive mailed to the DOE in February of 2014	
Appendix F Table of Contents.....	Appendix F: 2
0 Explanation of Hard Drive Materials	Appendix F: 3
1 Documentation	Appendix F: 3
A. End User Documentation.....	Appendix F: 3
1) User Manual	Appendix F: 3
2) GTDA Upload Forms.....	Appendix F: 3
3) Catalog Services for the Web (CSW) "Distributor.txt" Files	Appendix F: 3
4) Help Information	Appendix F: 3
B. Systems Operation and Administration Documentation	Appendix F: 4
1) Operations Manual	Appendix F: 4
2) Database Schema Documents	Appendix F: 4
3)Updating Static Portions of the Web Page	Appendix F: 4
C. Sustainability Plan.....	Appendix F: 5
D. Geoserver Links for 20 Datasets.....	Appendix F: 5

2 Datasets Downloaded from GTDA Feb 10, 2014	Appendix F: 5
A. Describe Columns	Appendix F: 5
3 PDF Publications Stored in Server	Appendix F: 5
A. Copies of Bibliographies	Appendix F: 6
B. Scanned Publications - Ok to Distribute	Appendix F: 6
C. Scanned Publications - NOT Ok to Distribute	Appendix F: 6
4 Related Resources and Logs Stored in Server	Appendix F: 6
A. BEG Links to Core Log Library	Appendix F: 6
B. Cornell Map Image	Appendix F: 6
C. SMU Field Notes	Appendix F: 6
B. SMU TD Curves	Appendix F: 6
C. Texas Tech Well Log Images	Appendix F: 6
5 Dump of GTDA Application Server sgtl1p1 Feb2014	Appendix F: 6
6 Dump of GTDA Database Server sgtl1d1 Feb2014.....	Appendix F: 6

Appendix G: Database Schema for GTDA

Appendix G Table of Contents.....	Appendix G: 1
Data Definition Language (DDL) for the GTDA System	Appendix G: 2
Tables and Fields in the Staging Database.....	Appendix G: 217
BEG Reservoir.....	Appendix G: 219
BEG Well Data	Appendix G: 224
Cornell Well Data	Appendix G: 230
Counties	Appendix G: 233
Field Reports ID	Appendix G: 234
Mat Views.....	Appendix G: 235
MLKay BHT Data	Appendix G: 235
Publications	Appendix G: 235
SMU Heat Flow (EQ Data – Part 1 of 3).....	Appendix G: 236
SMU Heat Flow (EQ Data – Part 2 of 3).....	Appendix G: 237
SMU Heat Flow (EQ Data – Part 3 of 3)	Appendix G: 238
SMU BHT Heat Flow	Appendix G: 238
SMU Heat Flow (EQ Data – All 3 of 3)	Appendix G: 241
States.....	Appendix G: 244
Publications	Appendix G: 244
TTU Well Data (Part 1 of 3)	Appendix G: 245
TTU Well Data (Part 2 of 3)	Appendix G: 245
TTU Well Data (Part 3 of 3)	Appendix G: 246
TTU Well Data (All 3 of 3).....	Appendix G: 247
UND Radiogenic Heat Data	Appendix G: 250
UND Thermal Conductivity Data.....	Appendix G: 253
UND Temperature Depth Data	Appendix G: 258
Well Log Headers	Appendix G: 261
Tables and Fields in the Core Database	Appendix G: 263

Aquifer.....	Appendix G: 269
BHT Measurements	Appendix G: 269
Casing.....	Appendix G: 270
County	Appendix G: 271
Dataset ID	Appendix G: 271
Dataset Job.....	Appendix G: 272
County	Appendix G: 272
Datasets.....	Appendix G: 273
Depth (& Mudweight Viscosity).....	Appendix G: 274
Field Report Mapping ID.....	Appendix G: 275
File Upload	Appendix G: 275
Filter Condition.....	Appendix G: 276
Fluid	Appendix G: 276
Formation.....	Appendix G: 279
Heat Flow	Appendix G: 279
Heat Flow Publication Link Ids.....	Appendix G: 280
Radiogenic Heat Generation IDs	Appendix G: 281
Heat Flow	Appendix G: 281
Ref_State	Appendix G: 281
Intervals	Appendix G: 281
Lithology	Appendix G: 282
Materialized Views	Appendix G: 282
Heat Flow Notes	Appendix G: 283
Precision TD Logs	Appendix G: 284
Publications	Appendix G: 285
Publication Metadata	Appendix G: 285
Quality Issue Log ID	Appendix G: 287
Radioactivity.....	Appendix G: 287
County	Appendix G: 287
Elevation.....	Appendix G: 288
County	Appendix G: 288
Data Source.....	Appendix G: 289
Quality Issue Type	Appendix G: 289
State	Appendix G: 290
Geologic Reservoir	Appendix G: 291
Shape	Appendix G: 292
Site Comments.....	Appendix G: 292
Publication Links & Related Resources	Appendix G: 293
Aggregated Well Data Tier 2 (Also called Surface Site County State)	Appendix G: 294
Temperature Series (TD Curves).....	Appendix G: 297
Borehole Temperature Observation Content Model	Appendix G: 297
Fluid Flux Injection Disposal Content Model	Appendix G: 301
Heat Flow Observation Content Model	Appendix G: 303
Radiogenic Heat Production Content Model	Appendix G: 308
Geologic Reservoir Content Model	Appendix G: 310
Thermal Conductivity Observation Content Model	Appendix G: 313

Well Fluid Production Observation Content Model	Appendix G: 316
Well Header Observation Content Model.....	Appendix G: 318
Well Log Observation Content Model	Appendix G: 322
Well Test Observation Content Model.....	Appendix G: 324
Temperature Series Mapping ID	Appendix G: 326
Thermal Conductivity	Appendix G: 326
Well Dates.....	Appendix G: 327
Well Info	Appendix G: 328
Well Info ID	Appendix G: 328
Well Test	Appendix G: 329
File System	Appendix G: 330
Table "filesystem.directories"	Appendix G: 330
Index "filesystem.file_system_dir_pkey"	Appendix G: 330
Index "filesystem.file_system_files_pkey"	Appendix G: 330
Table "filesystem.files"	Appendix G: 330
Staging Transform Operations	Appendix G: 332
Source Code for BEG Well Transform Operations.....	Appendix G: 332
Source Code for BEG Reservoir Transform Operations.....	Appendix G: 338
Source Code for Cornell Transform Operations	Appendix G: 340
Source Code for Create Materialized Views Transform Operations...	Appendix G: 345
Source Code for Drop Materialized Views Transform Operations.....	Appendix G: 346
Source Code for MLKay BHT Transform Operations	Appendix G: 346
Source Code for MLKay UIC Transform Operations	Appendix G: 348
Source Code for MLKay WB (Well Bore) Transform Operations	Appendix G: 350
Source Code for Re-Creation of All Materialized Views Transform Operations	Appendix G: 353
Source Code for Refresh Materialized Views Transform Operations .	Appendix G: 355
Source Code for SMU BHT Heat Flow Transform Operations.....	Appendix G: 355
Source Code for SMU EQ Heat Flow Transform Operations	Appendix G: 358
Source Code for Texas Tech University Transform Operations	Appendix G: 362
Source Code for UND Transform Operations	Appendix G: 367
Core Transform Operations.....	Appendix G: 374
Source Code for bounding_box Transform Operations.....	Appendix G: 375
Source Code for create_matview Transform Operations	Appendix G: 375
Source Code for drop_matview Transform Operations.....	Appendix G: 376
Source Code for generate_uuid Transform Operations	Appendix G: 376
Source Code for populateprecisionlog Transform Operations	Appendix G: 377
Source Code for recreate_all_matviews Transform Operations	Appendix G: 378
Source Code for refresh_matview Transform Operations	Appendix G: 380
Source Code for register_dataset Transform Operations	Appendix G: 380
Source Code for remove_dataset Transform Operations	Appendix G: 381
Source Code for update_all_matviews Transform Operations	Appendix G: 382
Source Code for update_surface_site_detailed Transform Operations	Appendix G: 384

Appendix H: Reports testing of the SMU Node for Compliance

Geoserver Report	Appendix H: 1
Method of Testing.....	Appendix H: 1
Test Results.....	Appendix H: 4
How to Test SMU Node Data in GeoServer	Appendix H: 5

Appendix I: Response to DOE Close Out Assessment.....Appendix I: 1

A. Response to the Assessment	Appendix I: 1
'Open' Discussion.....	Appendix I: 1
Final Report Location.....	Appendix I: 2
Data Quality.....	Appendix I: 2
B. Data Inventory and Quality Report	Appendix I: 3
Published Papers on Quality	Appendix I: 4
Screen Shot Images from October 2013 Presentation	Appendix I: 4
Tab 1: Process	Appendix I: 4
Tab 2: Content Model Format.....	Appendix I: 8
Tab 3: SMU BHT Submission	Appendix I: 9
Tab 4: SMU Side By Side Example.....	Appendix I: 12
Tab 5: Cornell Summary Example.....	Appendix I: 17
C. System Limitations	Appendix I: 30
Data Submissions	Appendix I: 30
Content Model Mapping.....	Appendix I: 30
Related Resources.....	Appendix I: 30
Publications - CSW Records.....	Appendix I: 31
Publications - Searching within GTDA Interface	Appendix I: 33
Corrupted Sites	Appendix I: 33
Distributor Text File Currency.....	Appendix I: 34
D. Conclusion	Appendix I: 34

Organization of Final Report

This report is a summary of how the project goals and objectives of Project DE-EE0002852 for the Aggregation of Geothermal Data for Submission to the National Geothermal Data System were accomplished. It includes the following sections and appendices:

- I. An **Executive Summary**, describing the purpose of the project and the key accomplishments. P. 2
- II. A listing of the **Statement of Project Objectives**, including the specific tasks necessary to complete the project objectives. Each specific task is followed by a description of how it was accomplished over the course of the 4 year project period (February 2010 through February 2014).
- III. A **Summary of Project Activities** from February 2010 through February 2014, including a discussion of lessons learned, changes in approach, remaining challenges, and opportunities for further study.

Appendix A, containing the **list of links** to access the various datasets collected for the project.

Appendix B, listing the **publications and presentations** of project team members over the course of the 4 year project term. While not all of these presentations are specifically about the Data Aggregation for the National Geothermal Data System, they do cover the research activities underway by the consortium team members which were incorporated into the data submissions to the NGDS.

Appendix C, containing the **user manual** for access to the interactive application available at <http://geothermal.smu.edu>

Appendix D, containing the **systems administration manual** for ongoing operation of the SMU Node of the NGDS residing at <http://geothermal.smu.edu>

Appendix E, containing the **Sustainability Plan** for the SMU Node of the NGDS.

Appendix F, outlining the contents of the **materials provided on an external hard drive** mailed to the Department of Energy in February of 2014. The hard drive includes a copy of all of the data collected for the project, the GTDA software, and documentation of the system.

Appendix G, describing the **database schema** used for the Geothermal Data Aggregation (GTDA) system. This was updated April 7, 2014.

Appendix H, agencies' **reports testing of the SMU Node** for compliance.

Appendix I, response to **Close-Out Assessment, Quality Presentation, and Known System Limitations**. A draft version of this final report was submitted February 12, 2014. On February 13, 2014 SMU conducted a web based meeting to present the project conclusions and final report. The DOE prepared an assessment of the project and requested inclusion of an October 2013 presentation on quality analysis efforts. Appendix I is primarily a response to the assessment and screen shots from the October 2013 presentation. In addition, Appendix I lists known system limitations.

I. Executive Summary

For over sixty years, geothermal energy has been recognized as one of the most promising renewable energy sources available. Resource studies continue to show enormous potential for electricity production capacity; therefore, it begs the question: why hasn't geothermal energy become a larger part of our energy 'mix'?

Properly understanding the available resource and project risks, are key to successful development and long-term operation. Depending upon the bias of the researcher/developer, "resource assessment" may mean evaluating any or all of the following:

- the thermal energy or heat in place for the location under study;
- the ease with which a particular location has access to necessary project components, such as the electrical grid;
- the ease with which suitable land can be purchased or leased and the permitting involved for drilling and/or injection obtained;
- the fluid type and volume, either naturally present or available to produce/inject;
- the pressure of the fluid circulating within the rock, which might serve as an additional source of power (geopressure geothermal);
- the access to capital, equipment, surveyors, attorneys, reservoir engineers, etc.

Historically, obtaining the information necessary to properly gauge a potential project's success or failure has been difficult, thus creating a barrier to entry for geothermal energy projects. The National Geothermal Data System (NGDS) seeks to further geothermal project development both by increasing the information available to project developers and by improving the access to, and quality of that information.

As a recipient of DOE American Recovery and Reinvestment Award DE0002852, the SMU Geothermal Laboratory consortium aggregated heat flow and related geological and geophysical information for submission to the NGDS. The organizations involved include:

- Bureau of Economic Geology at the University of Texas at Austin (BEG)
- Cornell Energy Institute at Cornell University
- Geothermal Resources Council (GRC)
- MLKay Technologies
- Texas Tech University
- University of North Dakota (UND)

Siemens Corporate Technology (SCT) provided systems integration services to amass the disparate data collections of each of these groups into a format that could become part of the new NGDS (Table 1). Siemens developed the SMU Geothermal Data Aggregation (GTDA) application to merge the different types of incoming data into a form that could be served to the NGDS utilizing the selected standards for data and data transfer. The data and application are available at <http://geothermal.smu.edu>.

Table 1. List of the project data summary

Data Provider	Data Item
SMU Geothermal Laboratory	• Nationwide heat flow and gradient data on >44,000 wells • >6,000 related resources tied to wells • >900 publication PDFs and related links
Bureau of Economic Geology at the University of Texas at Austin	• Texas reservoir analysis (55 different reservoirs;138 depth slices) • Well data on ~42,000 wells • Links to >269,000 samples in the BEG Core Log Library • ~400 publication PDFs
Cornell Energy Institute at Cornell University	• ~9,000 wells in NY & PA with heat flow calculations • Estimated temperatures @ depth to 10 km • ~600 publication PDFs • Economic Analysis report & maps
Geothermal Resources Council	• Bibliographic information for 33,000 publications • ~13,000 PDFs uploaded, 15,000 subject & geographic indexed • New online library website containing concise abstracts
MLKay Technologies	• > 1 million Texas oil and gas well site data • Including location, flow, test, depth, age, wellbore, etc.
Texas Tech University	• >40,000 wireline logs associated with over 8,100 wells • Largest off-shore temperature data collection from BHT in U.S., including location, corrected BHT, logs, etc.
University of North Dakota	• 245 Temperature @ depth data for 6 states • Significant contribution to SMU's nationwide heat flow dataset • 217 Thermal conductivity measurements • 350 Radiogenic heat production measurements • >125 publication PDFs
Siemens Corporate Technology	• Provided complete systems integration

Additionally, the project enabled the association of related data, such as a researcher's field notes, with well log measurements values, together with relevant scientific publications, all related to one or more specific wells with temperature logs. This information is now available to the public for the first time ever, in a variety of formats suitable for further study and analysis.

This report discusses the objectives of the project and addresses how each was met, the data collected, the structure of the GTDA database, the files available for sharing on the NGDS as a result of the project, and lessons learned. The improvement in advancement of data quality as a result of this project is of key importance. By working collaboratively with colleagues and by aggregating our data, we were afforded an opportunity to improve the overall quality of the data collection originally maintained individually by each group. We benefited from the information exchange with each other and in sharing new techniques and approaches to the study of heat flow and geothermal energy. The freedom to present complete data sets for submission, without the size constraints of journal publication, results in an immense increase in the usefulness of the data to the geothermal community.

The SMU Geothermal Laboratory would like to extend our thanks to all of the project participants, as well as to the Department of Energy and our financial supporters. It is our sincere hope that the

work effort leads to increased use of geothermal energy as a source of clean, reliable, baseload renewable energy.

II. Review of Statement of Project Objectives by Task

The project plan defined the objectives and scope, along with the specific tasks necessary to achieve each item. The project objectives and scope are included for reference below and appear in italics. Following the objectives and scope, 16 specific tasks necessary to accomplish the objectives are listed, also in italics. Following each assigned task, a narrative response (not in italics) summarizes the performance of that task over the course of the project. All project objectives were successfully achieved as a result of the completion of the 16 project tasks.

A. PROJECT OBJECTIVES

High quality information supporting geothermal research and development will be submitted to the National Geothermal Data System (NGDS):

- *An expanded and updated version of the SMU Heat Flow database that covers the whole onshore US and offshore regions on the Gulf of Mexico.*
- *The Geothermal Resources Council (GRC) library with over 36K documents and over 1.3M pages on geothermal research.*
- *Extensive information on legacy Enhanced Geothermal System research including data files related to projects and EGS in Europe.*
- *Current results of research in the mid-continent and eastern US.*
- *Detailed geopressure data, core logs, well logs and reservoir distribution and analysis from the Texas Bureau of Economic Geology (BEG) covering primarily eastern Texas with the approximate western boundary along the I35 corridor.*

In particular the efforts on documentation include: the BEG focus on the integration of geopressure data developed by DOE in the 1970-1980 and current analysis expertise which will include compilation of data on “net sand” and related information on the volume and quality of clastic reservoirs in the Gulf Coast; the recovery of the legacy EGS data; and meta-data associated with US thermal mapping crucial to be performed at this point because they are otherwise in danger of deterioration or complete loss.

In order to facilitate the computational and data management aspects of geothermal data collection, maintenance and submission, this project will use software, consulting services and a collaboration platform to its partners. This centralized approach to the IT aspect of geothermal data submission streamlines the overall process by avoiding duplicate efforts within disparate groups. The software will be developed according to the standards defined by the NGDS team to provide data and meta-data. This enables other people, such as co-awardees of DE-FOA-0000109 of all topic areas, to reuse and complement the tools.

Successful completion of this project will result in multiple positive outcomes. Primary benefits will include support for developers of geothermal power plants by decreasing the costs of the resource identification phase and the risks of the exploration phase of developers. The project will rescue important data from deterioration or complete loss and provide a set of tools to be used by other parties to submit data to the NGDS.

B. PROJECT SCOPE

Developing sustainable solutions for energy generation is one of the great challenges of mankind. It is therefore crucial that we continue to expand the use of renewable energy sources and develop better technologies to exploit them. A recent study has identified the need for an easy to use integrated source of information on geothermal resources, the National Geothermal Data System (NGDS). Access to such data will reduce the identification costs and investment risk for developers and help to identify regions where scientific data collection on geothermal resources should be conducted. A grant has been awarded to build the database infrastructure and specify data formats and protocols for submission of data by third parties. With great experience on the NGDS team, the geological side of the data system can be expected to be well represented leveraging on related efforts to build a Geoscience Information Network (GIN) with a national geologic map database. This project complements these development efforts by assembling a unique group of geothermal specialists and others with the ability to develop and provide large amounts of geothermally relevant information from existing databases, legacy and undigitized data, and new collection efforts, partially co-funded by other programs and sponsors, and at the same time quickly creating a format and submission infrastructure to compliment planned development of the NGDS..

*This project addresses the following challenges: The first challenge is to **complement geological data with additional information needed for geothermal resource assessment**. The second challenge is **data selection, integration, and quality assurance**. The third challenge lies in the **organizational aspect of geothermal data collection**.*

The geothermal knowledge base, information stored and accessible through this project, and technology expertise will enable the consistent application of quality standards and ensure a seamless initial and continued transfer of the data into the NGDS. Through the extensive network of past and current partners and experience with similar government programs the team will ‘hit the ground running’.

C. TASKS TO BE PERFORMED

Phase I. Data Retrieval, Collection, Development and Quality Analysis

Task 1. Data assessment

During this project’s data assessment task, the domain experts will assess available geospatial data according to integrity, usefulness, value to stakeholders, potential acquisition costs, and potential danger of deterioration. Particular emphasis will be put on EGS data. Finally many of the SMU participants will assess Oil & Gas associated geothermal data including geopressure data. Each SMU team will assess its own data and compile reports on the amount of existing and anticipated data, expected data quality and relevance to geothermal research. The SMU team’s reports will be circulated and discussed to make decisions on which data to include.

Each of the project participants evaluated their existing data, resources, and available skill sets.

Areas of overlap were discussed and a plan formulated to maximize the usefulness of the data collected, prioritizing the efforts of each project participant. The work effort extended well beyond gathering existing data, to include new data collection and new data synthesis as well.. The following is a description of the items submitted by each project team to the full SMU National Geothermal Data System (NGDS) Node, referred to as the SMU Geothermal Data Aggregation (GTDA). By the project conclusion, a significant quantity of data were collected, analyzed for quality, and incorporated into the SMU GTDA system.

SMU Geothermal Laboratory contributed two different nationwide heat flow databases to the project. One is based on equilibrium temperature measurements (over 14,000 sites) and the other is based on corrected bottom hole temperature (BHT) data from oil and gas industry wells (over 30,000 sites). In addition, scanned field notes and temperature-depth curves were associated with approximately 6000 specific sites in the heat flow database. Records were corrected and overlapping sites in the equilibrium heat flow database were linked between the original SMU National database and the UND Global Heat Flow database. New or related sites, which were not previously published because they lacked full heat flow content, are now included as gradient only information along with their detailed temperature data to fill-in data gaps. Finally, SMU submitted over 920 scanned publications, reports, and maps suitable for full text searching.

Bureau of Economic Geology at the University of Texas at Austin (BEG) contributed their Master Wells database, which includes approximately 42,000 oil and gas wells with data on bottom hole temperature, depth, corrected bottom hole temperatures (using the SMU-Harrison method), fluid type and content, and other information. Additionally, they contributed analysis of 138 depth slices of geologic reservoirs in Texas, making available the porosity and permeability values for the related formations not previously easily accessible to the public. This effort improves the ability for geothermal-geopressure resource analysis in Texas. The BEG also selected and scanned approximately 400 publications from their extensive library that were of most relevance to geothermal projects. BEG provided thousands of links to their library of rock core samples.

Cornell Energy Institute at Cornell University contributed a new Pennsylvania and New York oil and gas database of over 9,000 wells. Data were from the SMU Geothermal Laboratory's Eastern U.S. BHT database, the New York State Museum Office of Oil and Gas and the Pennsylvania Geological Survey. Cornell amalgamated the data for improved outlier detection, corrected bottom hole temperatures, determined estimated temperatures-at-depth for each well, estimated depths-to-temperatures for each well, and calculated heat flow for the sites. Their processing of the data utilized methods consistent with the SMU Project Team techniques for temperature corrections and heat flow calculation. The Cornell team and SMU team worked together to develop an approach to increase speed of temperature-depth calculations, initially based on the technique SMU used in the updated 2011 GOOGLE Geothermal Map of the Conterminous United States. Their maps and results were calculated at standard depth levels to aid in the resource evaluation of their region that is focused on enhanced geothermal systems (EGS) or heat pump type geothermal projects. They also contributed approximately 600 scanned publications suitable for full text searching.

Geothermal Resources Council (GRC) completely overhauled their online library, as well as inventorying their physical library collection for the project. They submitted all of the GRC

Transactions for full text searching, along with several thousand additional publications suitable for distribution. They provided URL links to their new online library of approximately 13,000 publications and to Stanford's workshop proceedings. They redesigned the cataloging of the library, which included updating the keywords and abstracts for thousands of publications, in addition to subject and geographic indexing nearly 15,000 catalog records.

With regard to library scanning contributions, in addition to GRC, four other members of the SMU Project Team (as detailed in this section) had collections of geothermal publications dating back over 40 years. All of the libraries were inventoried and the bibliographies shared with Siemens. Siemens combined the lists, de-duplicated references, removed items already scanned by outside groups such as OSTI.gov, and then distributed a scanning assignment prioritized by the capacity of each group. This document scanning list was then submitted to the Boise NGDS team and the State Geological Survey NGDS team in Arizona to avoid duplication. Rules for scanning were also agreed upon by all team members. It was determined when possible publications were scanned at 300 dpi (or greater) and had optical character recognition (OCR) completed on them for text searching within the documents.

MLKay Technologies contributed well header data, 'G10' (fluid flow and type) test information, fluid injection data, casing data, fluid production data, etc., for nearly 1,000,000 oil and gas wells in Texas across the entire state. While some wells have more information available than others, the volume of data is greater than the geoscience community has ever had access to before. The well sites include a current status of active, inactive and abandoned.

Texas Tech University contributed a database of over 8,100 off-shore oil and gas wells, including bottom hole temperatures, depth, corrected bottom hole temperatures, and thousands of associated scanned well logs. Unlike many on-shore oil and gas wells, the off-shore data typically includes multiple temperature measurements on different dates in the same well. This allows for use of the Horner correction technique, which combined with the well logs, provides the most complete temperature assessment of the Economic Zone of the Gulf of Mexico ever done.

University of North Dakota (UND) contributed their US portion of the Global Heat Flow database, which was incorporated into the SMU heat flow database submission. An extensive effort to add new data increased the number of sites with thermal conductivity measurements and radiogenic heat production on hundreds of rock samples. They logged wells using equilibrium temperature measurement equipment. They scanned over 125 publications for full text searching.

Task 2. Definition of data quality standards

Each team will formulate quality standards and domain knowledge for their data sources that can be translated to automated quality assurance mechanisms such as rules and statistical tests. In addition to mechanisms to identify quality problems, the experts should also identify potential remedies such as removal of values, interpolation from neighboring data items, etc. One milestone will be a report submitted to the NGDS that specifies what data will be submitted to them and how it will be quality controlled.

The quality of the data has been a focus from the initial stages of the project. Large amounts of

time were appropriated to update the data, rework it with new programs and against newly collected data to find input errors in the datasets (both new and old). The bi-weekly meetings and two separate 2-day workshops hosted by SMU furthered the discussions on methods of data correction and consistency analysis and how to develop codes for quality within the several databases. In some instances, it was decided that it was appropriate to include multiple entries for the same location if there was uncertainty as to which record was of higher confidence level.

Specific steps taken to determine data quality standards and improve the data collection, processing, and uploading into the SMU NGDS Node are listed below. Descriptions of the data collections appear in the text under Task 1 as well as in Appendix A.

1. Overlapping datasets were reviewed for error detection.
 - a. SMU and UND combined equilibrium heat flow databases identifying same sites with different values and more/less details that could be combined.
 - b. SMU and Cornell BHT heat flow data sets were shared; comparison of methodology and outputs were assessed.
 - c. BEG and MLKay Technologies shared data to find overlaps and to connect missing data between the two sets of Texas Railroad Commission oil and gas data and the BEG IGOR database. By not duplicating data submission efforts, the BEG effort for the project was then reallocated to focus on the suitability for geothermal-geopressure projects in Texas creating a new reservoir analysis product. Net sand thickness, porosity, permeability, and other key attributes were calculated using data from the BEG repository, as well as data from SMU and MLKay Technologies.
2. Lists of publications from each group were combined into one list that also included the database of OSTI publications. From this list, Siemens assigned scanning of the publications. This deduplication step avoided multiple groups scanning the same item.
3. Data quality was improved with use of a tool within the SMU NGDS Node for reviewing the data using histograms. The tool is capable of reviewing data in either normal or logarithmic scales to show outliers to submitting organizations prior to the decision to publish data. Additionally, the tool is available for general users to analyze the published aggregated data.
4. Submitting organizations also have the capability to download a 'null data' report listing the empty fields within their data submission prior to the decision to publish the dataset.
5. The BEG focus on reservoir heat estimates used the data from the other groups along with their own. Through the calculation process the data were reviewed and quality assessed with outliers either corrected or removed.
6. Cornell Master's Thesis (2014) by Andrea Aguirre studied how to use gridding methods to improve the data assessment of large BHT datasets.
7. Siemens created individual upload forms for each group's data submissions. These forms assist in finding errors in spreadsheets to make sure the formatting of the file matches the expected inputs. For example, a non-numeric character will not be accepted in a field expecting temperature.
8. The system allows the data submitters to download their last successfully published file. This aids ongoing sustainability by providing a record to future researchers of the format used by previous project team members.
9. The additional resources uploaded into the SMU NGDS Node give future end users the

ability to review the data and look at the original field notes, temperature logs, and publications, etc., of heat flow sites thereby empowering users with the ability to determine the quality of the data.

Task 3. Software requirements and development of data processing framework

This project's software for geothermal data integration will be designed and implemented as an open, extensible, and scalable package supporting efficient integration, quality assurance, and dissemination of geothermal data according to the standards to be defined by the lead NGDS team. It will incorporate tools and algorithms that integrate data from various sources in different formats, check the consistency and quality, and provide added value such as geospatial referencing of documents and interpolation of missing values.

Early in the project, interviews were conducted about the type of information expected to be collected and submitted by each of the participating groups. This information was used to develop a Software Requirement Specification document, which in turn was used to generate a Software Design Specification document. These two documents described the functionality of the system and were used as guides throughout the development process. Central to the process was the ability for the system to accept data from the different providers using their native formats and integrate it into a single aggregated format, which could then be queried to serve data to the NGDS using the formats prescribed by the lead team. As the project progressed, the SMU Project Team and Siemens provided significant input to the other NGDS project teams.

The development process used the 'Agile Development' process, which allows for rapid prototyping and a tight feedback cycle. Siemens maintained a development system at their facility in Princeton, NJ to test the new code before deploying to the production servers at SMU. The cycle of new feature addition, testing, deployment and feedback was approximately every two weeks, depending upon the stage of the project.

The architecture selected for the project did not change, even as software features were modified and added to over time. The architecture makes use of a variety of open source products which are highly scalable to accommodate a variety of circumstances. After exploring other options, the open source product PostgreSQL was selected as the underlying database for the system. Other key components include Geoserver, to provide the map services and delivery of the data to other nodes of the NGDS using standard exchange protocols, and Geoportal, to provide a repository for the catalog describing all of the information available for sharing with the other NGDS nodes.

During the first 18 months of the project, the team did experiment with the possibility of estimating missing values through the use of automated interpolation, using the Texas Railroad Commission Data provided by MLKay Technologies as a sample case. Results were not of sufficient quality to pursue further, given the available resources. However, the interactive web version of the software available did later incorporate the ability to readily identify wells in very close proximity or with very similar well names, as a value-added feature to researchers. This initial well matching capability enables the researcher to consider the surrounding area in their analysis and eases the ability to compare multiple group submissions of the same site or those within close proximity. Linking of wells with related resources (field notes, publication references, well log images and

temperature depth curves) are additional value-added capabilities incorporated into the system.

Task 4. Development of data repository

A data repository will be created to store the raw data, derived secondary information and summaries of the data quality. This repository will be accessible to the NGDS system via links from their programs.

The Geothermal Data Aggregation GTDA system developed by Siemens Corporate Technology for this project contains the data repository and is available from <http://geothermal.smu.edu>. The system resides on two servers located at SMU: “sgtl1p1” and “sgtl1d1”. The first, sgtl1p1, contains the application code that operates the system. Sgtl1d1 contains the PostgreSQL¹ (pronounced Postgres) database.

There are actually two data repositories within GTDA: a ‘staging’ repository and a ‘core’ database. Data are uploaded using upload routines created for each specific data submission. These upload routines deposit the raw data (typically one or more spreadsheets or database files) into a series of tables within the staging repository. After the data submitter has evaluated the uploaded data, and confirmed the data is of sufficient quality to proceed (using tools available in the system to do so), the data is ‘published’. During publishing the GTDA performs a number of transform operations on the data residing in the staging repository in order to move the data into the core database. In some cases, the transform operations are relatively simple, and in other instances quite complex. Examples of transform operations include:

- Extract data from a single field within the staging repository (township range and section are often combined by the submitters) and split it into three different fields in the core database.
- Combine multiple fields of data from the staging repository into a single field within the core database.
- Use a look-up function to transform the staged database value into a core database value, such as state or county code for state name, or ‘gw’ for Gas Well, etc.
- Unit conversions, such as transforming a gradient from degrees Fahrenheit per 1,000 feet into degrees Celsius per kilometer.
- If/then operations, such as determining the maximum value (temperature or depth) from multiple reported values.

Quality data are available to the public from both the staging repository and the core database through a series of web links.

1. From the staging repository: After the SMU Project Team approves their files for publication in the core database, this “original” file submission continues to stay in the staging repository of the GTDA. Although it was designed for future access as an example and backup for the submitting project team member, it was realized that these files had useful file structure and regional focus for many end-users. These submission files are now made available for the public to download via either WFS/WMS². WFS and WMS are two

¹ Refer to <http://www.postgresql.org/> for more information on the open source database, PostgreSQL.

² WFS/WMS refers to Web Feature Service and Web Map Service, two standards defined by the Open Geospatial Consortium (OGC) for data exchange, and selected by the NGDS as a means of transferring data.

standards developed by the Open Geospatial Consortium (OGC), and selected by the NGDS for data transfer. These nine datasets available for download included 2 datasets submitted by SMU, 2 datasets submitted by the Bureau of Economic Geology, 1 dataset submitted by Cornell University, 1 dataset submitted by Texas Tech University, and 3 datasets submitted by the University of North Dakota.

2. From the core database (after aggregation): From the core database are 11 datasets available for download from the NGDS via either WFS/WMS. One of these files is the dataset containing all of the project team sites included within GTDA along with location coordinates and items of interest (if available) such as temperature, depth, heat flow, etc. The other 10 files use an NGDS defined 'content model' layout. The NGDS has defined 'content models' for the exchange of similar data across the NGDS, such as Borehole Temperature Observations, or Well Header information.

A complete listing of the 20 datasets available for download and the links available for various formats of delivery are provided in Appendix A of this report and/or by accessing <http://geothermal.smu.edu/static/DownloadFilesButtonPage.htm>.

These datasets, as well as thousands of scanned publications, are accessible via the NGDS through the use of Catalog Services for the Web (CSW) entries. A CSW record contains metadata, or 'data about the data', for each of the 20 datasets mentioned above as well as each of the ~24,000 publications for which there are catalog records. These catalog records are visible to other nodes of the NGDS, allowing the data resources to be 'discoverable' and thus accessible. The catalog records of the items residing on (or available through) the SMU Node of the NGDS may be searched from <http://geothermal.smu.edu/geoportal/catalog/search/search.page>.

In addition to the datasets available for download via WFS/WMS, <http://geothermal.smu.edu> offers an interactive application interface, which provides additional capability to explore data residing within GTDA, including thousands of related resources, use of the histogram capability, full text searching of the scanned publications, and additional filtering options not readily available through the download files via WFS/WMS option.

Finally, while not available directly for general public access, the GTDA servers also maintain copies of the original files uploaded into the system, including the structured datasets as well as scanned publications and other related resources. Copies of all of the data residing in GTDA have been placed on an external hard drive for Department of Energy use. In addition to a 'data release' of all of the data and code on the SMU GTDA servers sgtl1p1 and sgtl1d1, the hard drive contains data extracted from the system, such as someone might obtain through use of the system download features. Appendix G of this report provides a copy of the detailed database schema used for the Geothermal Data Aggregation (GTDA) system; the schema was included on the DOE hard drive and instructions for retrieving updated versions from the system are contained within the Systems Operations Documentation.

Task 5. Import of data available into data repositories

Once the initial version of the data processing framework and the data repositories is available it will be used by data owners to test the import of any data available so far. This will be mainly

existing data listed in the project description but also additional data collected during this project as it becomes available.

As mentioned above, the GTDA system maintains a staging repository and a core database. Import routines were developed for each type of data submission. Additionally, SMU Project Team submitters may retrieve a copy of anything they uploaded into the system through these import routines, even if they elected not to publish them to the core database. Instructions for uploading data into the system are provided in the User Manual, available from geothermal.smu.edu and/or within the Documentation/End User Documentation section of the hard drive delivered to the Department of Energy. Copies of the 'data upload forms' customized for many types of data submission have also been provided.

Task 6. Quality control and improvement of data

The quality of the NGDS integrated database is of utmost importance for ensuring reliable estimates of geothermal favorability and financial risk. The data quality modules implemented will be executed on the data repositories generating reports and starting a feedback cycle with the data owners that may lead to changes in the quality standards, upload of corrected data, addition of error correcting modules, etc. A quality check for temperature data would scan the database for missing values, unusual temperatures (out of range or very different from other spatially close points) and generate a quality issues report and maps of the data.

Quality control and improvement of data were of great importance at every stage of the project. During data collection efforts, groups collaborated to assess overlapping data and identify changes that might be necessary. For example, UND and SMU had a number of points that were very similar in location, yet had different calculated heat flow values. By aggregating the data, these differences came to light, leading to detailed study of areas of inconsistency, including a review of original research documents, to improve the data submission to the NGDS.

During the data upload process, the GTDA system verifies that only numeric values are permitted for certain fields and disallows special characters, etc. The decision was made to permit submission of extreme values for temperature, depth, heat flow etc., since there were numerous instances of valid extreme values. Instead, the GTDA software allows the submitter to view the uploaded data in a series of histograms to identify outliers and then make decisions accordingly. The extremity of some of the legitimate outliers, such as the heat flow values for the area in and around Yellowstone National Park, led to a modification in the histogram capability to allow selection of a standard or logarithmic scale and/or the selection of a range of values to view in a histogram. Histogram viewing of data was deemed valuable enough to expand it to the core database of aggregated data, as well as to the staged un-aggregated repository.

Following submission and publication into the GTDA core database, additional data validation efforts were completed. The data aggregation proved to be a useful new tool in assessing the individual contributions. For example, study of the aggregated database illuminated instances where BEG and MLKay Technologies had information on overlapping wells with different coordinates, leading to a discovery of a data file that needed to be re-submitted. Similarly, Cornell and SMU had overlapping data in Pennsylvania and New York; when rows of data pulled from the aggregated core database were studied in detail, errors were found in both of the original data

submissions, as well as in the mapping from the two data sets to the various content models. Without the benefit of the overlap, this difference would have been much harder to have identified and corrected. Of importance in this latter effort, the ‘human eye’ type analysis, rather than automated tools, was best suited to identifying a potential problem and deciphering the source of the problems.

Task 7. Collection of data with ongoing projects

This project will collect data where possible according to the current projects the team members are associated with during this project. This data collection will use the same quality modules implemented to upload it into the software, reviewing it with previously existing data for overlap and accuracy.

Throughout the project, the entire SMU Project Team was fortunate to be involved in numerous research efforts overlapping with the data collection and analysis efforts. For example, two SMU Masters students obtained geophysics degrees over the course of the project and both submitted their research data and results to the NGDS. Ph.D. Candidate Joseph Batir worked with the Alaskan Energy Authority to update the Geothermal Map of Alaska and included the new data points collected within the SMU submissions. Similarly, UND faculty, graduates and undergraduates collected new data in 2012 and 2013 which were incorporated into their radiogenic heat, thermal conductivity, and temperature-at-depth submissions. Cornell graduate student theses were directly related to the NGDS submitted databases. Their intense focus instigated the cross-collaboration between SMU and Cornell graduate students, which has created a new norm for ongoing research efforts between the two schools. UND, SMU, and Cornell students all participated in the Department of Energy Student Competitions for geothermal research. The efforts by BEG to incorporate porosity, permeability, and fluids into their reservoir engineering have already increased the numbers of calls and requests for assistance in companies looking for coproduction project site placement. The BEG staff researchers on this project have now been hired by geothermal companies or been accepted into programs for advancing their degrees related to geothermal. Texas Tech researchers collected new data during the project term and are in the process of mapping the heat flow of the Gulf of Mexico with much higher detail than previously possible. The GRC’s new online library system has already greatly improved the ability to find their publications and will continue to be a useful tool after the full NGDS deployment, as demonstrated by a call this week from a western state land office evaluating the potential for geothermal power production in known geothermal areas that cross into state-owned lands.

A review of the papers, posters, and theses presented during the project term in Appendix B of this report provide insight into the breadth and depth of the incorporated research.

Additionally, the GTDA software allows for the ongoing submission of new or revised data in the future. SMU, BEG, Cornell, GRC, Texas Tech and UND each anticipate uploading more data and resources.

Task 8. Development of quality assurance reporting

The quality of the NGDS integrated database is of utmost importance for ensuring reliable estimates of geothermal favorability and financial risk. The team of geothermal experts and data owners will provide the domain knowledge for the formulation of data quality rules and will

include them as part of the reporting.

SMU hosted a two-day workshop with participation from across the SMU Project Team and included personnel from DOE Geothermal Technologies Office and the Arizona Geological Survey to discuss different aspects of heat flow data quality, including an assessment of the various methodologies for correcting bottom hole temperature (BHT) measurements of oil and gas wells to approximate equilibrium well bore temperature values. This workshop was very helpful in coordinating similar temperature and depth analysis efforts taking place at SMU, Cornell, BEG, Texas Tech, and UND. One result of this workshop was an effort lead by SMU to develop a method to merge the quality ratings for heat flow calculations derived from equilibrium temperature logs (measurements at multiple depths throughout a wellbore on a well that has returned to equilibrium after the disturbance created by drilling) versus those derived from oil and gas wells, which might consist of only one temperature measurement collected during the drilling process. SMU researcher Maria Richards created a MatLab program to assign a numerical quality rating that could account for the different factors that affect the confidence of heat flow determinations from a variety of circumstances. A GRC (2012) paper was written and presented to the heat flow community for feedback. While the program is available for others to use by accessing <http://geothermal.smu.edu/static/HeatFlowQualityCode> (and is provided on the hard drive of project data provided to the DOE), the fact is, there is no substitute for human due diligence of the specific reservoir analysis during a geothermal favorability and financial risk analysis.

A review of Cornell University graduate student Andrea Aguirre's Thesis (2014) on the use of gridding methods to improve the data assessment of large BHT datasets is an example of the synthesis of 3 ½ years of review of data quality in heat flow. This research will be presented again this year at the Stanford Geothermal Workshop to reach a larger audience.

Task 9. Requirements NGDS standards and protocols

Once the requirements are stated by the lead NGDS team, this project will incorporate them into the software and computer system to support the submission of data from the data repository along with meta-data records. This project will also assist the lead NGDS design team with NGDS data protocols by describing the type of data that will be provided.

The SMU Project Team participated in numerous discussions of NGDS standards and protocols. In addition to sharing a list of keywords and synonyms organized in a hierarchal structure, the team was quite involved in the development of the content models for data exchange across the NGDS. The SMU Project Team reviewed and suggested numerous modifications to standards for borehole temperature observation, well header observations, well log observation, and fluid production observation. Team members also created the foundation of several content models in collaboration with the Arizona Geological Survey in order to accommodate the data coming from the SMU consortium; these data sets included: heat flow observation, thermal conductivity observation, radiogenic heat production observation, well test observation, and fluid flux injection observation.

While an important part of the project, this activity proved significantly more time intensive than

anticipated. While most of the 10 content models referenced above are relatively accurate in the representation of the data they include, there is room for continued refinement, particularly in the area of improving consistency of terminology and formats between the different content models. For example, some content models accommodate multiple dates associated with a given well observation whereas others only accommodate a single date. When populating multiple content models from a single aggregated database such as GTDA, it becomes complex to handle such differences. In other instances the content model terms changed with multiple reviewers after data had been mapped to a particular field. The SMU heat flow data collection includes a field called ‘minimum temperature’, meaning the coldest temperature value; with the well intentioned goal of improving consistency across different content models, reviewers modified this field to ‘shallowest temperature’ or first depth measurement, which is not always the ‘coldest’.

Additionally, it would have been highly desirable to have known the specific content model formats prior to designing the database schema for GTDA. Given the timeline of the SMU Data Aggregation project, it was not possible to wait until the content models were finalized to design the database schema; having this information in advance would have undoubtedly informed the decision process for how to best organize the tables and fields for the aggregated core database.

Task 10. NGDS submission

A likely scenario for data submission to the NGDS is via links and portals as opposed to physical delivery of media or a manual upload. The software architecture will be extended by a web server component that retrieved data from the data repository and implements the transmission standards to be defined by the NGDS team.

The SMU Project Team data are available via links and portals, using web server components selected by the NGDS team, specifically WFS and WMS. Appendix A of this report provides a complete listing of the URLs. In all, 20 datasets are available in this method. For each of the datasets, download links are provided for:

- The full dataset delivered via WFS formatted as a CSV file (note: some are too large for this to be practical)
- The first 50 rows of the dataset delivered via WFS formatted as a CSV file
- The full dataset delivered via WFS formatted in XML (suitable for use in a web client program designed to understand XML)
- The description of the XML schema, for someone planning to import the data using XML
- The data set delivered via WMS version 1.1.1 for use with a map based program
- The data set delivered via WMS version 1.1.3 for use with a map based program
- A file describing the dataset fields for the individual submission file downloads or the relevant NGDS content model for the aggregated core datasets
- The command to query the number of records for the dataset

In addition to the information in Appendix A, URLs to these files are available in a table appearing on <http://geothermal.smu.edu/static/DownloadFilesButtonPage.htm>

As mentioned previously in the discussion of *Task 4. Development of the data repository*, the data are discoverable on the rest of the NGDS by virtue of the CSW records for each dataset and for each publication kept in Geoportal, running on the SMU Node of the NGDS. The other NGDS

nodes must ‘harvest’ these CSW records periodically in order to be able to direct users to the WFS/WMS links and reach the desired data.

In addition, the data were placed onto an external hard drive for delivery to the DOE. Because the data continue to go through additional corrections and updates, obtaining the data directly from <http://geothermal.smu.edu> in the future is recommended; however, this hard drive does serve as added backup.

Task 11. Maintenance of computers and software framework

The computers, servers, and software used will be updated and maintained throughout the SMU project. Multiple copies of the data will be kept in different locations as to not lose any data in a disaster situation such as fire, theft, or from natural causes.

The computers, servers, and software were all maintained with redundant backups in Dallas and Plano, Texas throughout the project. No fire, theft or natural disasters caused a loss of data. It proved very helpful to have a complete copy of the software and data at an alternate test environment at the Siemens facility in Princeton, particularly when the performance of the SMU Node became untenable due to unidentified software or configuration ‘bugs’. The debugging progress was slow and the most efficient resolution was to perform a ‘dump and restore’ using the Princeton version.

Phase II. Transfer and Validation of Information to Data System

Task 12. Testing of data submission to NGDS with validation

The transmission of the data to the NGDS requires testing to ensure that the high quality data is consistent. This may include the use of validation tools provided by the NGDS to perform simulation of data submission locally. Another possibility is for NGDS to provide a test server for submission that allows the user to check whether the data was uploaded successfully using checksums and manual validation.

It is of utmost importance for the other NGDS Nodes and related programs to be able to find and retrieve all the data and related files submitted from the SMU Node of the NGDS. In order for this to occur, three elements are required:

1. Catalog Services for the Web (CSW) – in order that the other nodes are aware of the data available from the SMU Node.
2. Web Feature Service (WFS) and Web Mapping Service (WMS) – compliance with the data transfer standards
3. Content Model delivery of the data – in order that applications harvesting like data from multiple nodes can merge the data and/or interpret it as if from a single source.

The Catalog Services for the Web (CSW) testing was completed by Siemens, SMU Geothermal Laboratory staff, and staff from Arizona Geological Survey’s NGDS team. As a catalog, data and related files will be detectable from any node through a basic file search on file title, keywords, bibliographic details, source, author, publication date, etc.

Compliance with the WFS/WMS delivery of data has been validated. On February 7, 2014, Paul Edwards, contractor to the National Energy Renewable Lab tested WFS and WMS access the datasets available for download from the SMU Node. Among other things, the test included successfully loading the datasets in as discrete 'layers' to the map view of Geothermal Prospector. A copy of his report indicating a 'pass' for all 20 datasets is included in Appendix H of this report. In addition to the report document, the accompanying email stated, *"It should be noted that this was not a load test, but rather a friendly kind of test where I was careful to only hit one layer at a time, and even at times would wait and retry the test if it seemed to be responding quite slowly. I think it would be quite possible to bog the server down by trying to view 2 or 3 of the big layers at once in WMS while zoomed out beyond viewing a single US state, for example. When testing I found it much more responsive to zoom in to an area like the Texas Panhandle before turning on a layer. At that level all of the layers rendered in a pretty reasonable amount of time."*

The reasonable response time was an improvement from NREL tests conducted the week prior. The previous testing prompted changes to the configuration to improve caching for rendering map images as well as intentionally limiting the largest dataset to 1 million records for WFS delivery to prevent timeouts for multiple users. Note that the largest of the datasets, Fluid Flux Injection Content Model delivery, contains nearly 10,000,000 records, which is larger than the capability of spreadsheet programs to consume. Thus, forgoing the delivery of the full dataset in a CSV format was deemed reasonable. The data may still be viewed as a map layer via WMS delivery into a program such as ArcGIS or Global Mapper, or retrieved in sections as XML via WFS, for users in need of the data.

SMU Geothermal Laboratory personnel tested the ability to pull the files via WMS into programs such as ArcGIS and Global Mapper as of January 27, 2014. Additionally, the files have been successfully tested for downloading via WFS as CSV files for import into a spreadsheet since September of 2013. SMU does not have the capacity to test the XML consumption of the data via WFS, but the NREL tests indicate that they operate as expected. The files were successfully downloaded and data configuration successful as of January 27, 2014.

For future testing, a convenient way to test the accessibility of data is through the Geoserver's Layer Preview Page: <http://geothermal.smu.edu:9000/geoserver/web/>. See additional details in Appendix H, section on GeoServer.

Task 13. Data submission to NGDS with validation

Once the transmission procedures are successfully tested the production submission process can start. All data to be stored or indexed in the NGDS will need to be published and the result will be validated to verify the successful donation of the data.

During the first phases of the project, Siemens received files directly from the project team members and loaded them into the staging repository. With the exception of the data submitted by MLKay Technologies, each of the data submissions were expected to be refreshed over time. It should be noted that during these initial stages, there were numerous changes to the file structures being submitted by each group. A 'freeze date' for changes to formats was established so that Siemens could know what format to use for writing upload routines for subsequent data

submissions. After completion of the upload routines in the spring of 2013, each group was instructed to upload their own files. By January 2014, every group has successfully uploaded data with the exception of the GRC catalog refresh. The GRC upload process is being reviewed to understand why their files are not accepted into the system from their offices in California. Note: their data refreshes have been successful from SMU workstations signed on using GRC user credentials and/or from Siemens facilities in Princeton, pointing towards the possibility of a potential firewall or network configuration setting.

The ability for each project team to upload their own files has been difficult to implement because of the submissions being required to have exact compliance with the GTDA system. Complicating this, data quality validation efforts in the fall and winter of 2013 precipitated the change to some of the data submission formats as late as January of 2014, over a year beyond the ‘freeze date’ of October 31st of 2012. Siemens worked as much as possible with users in creating additional formatting through-out the project, many of them beyond the October 31, 2012 date (e.g., a heat flow site publication reference criteria: AuthorYearCode was expanded to hold up to three entries, additional well identifiers were added to the SMU BHT dataset, etc.). To help future updates and additions, manuals and customized template forms have been created. They are included in Appendix C. The written instructions have been clarified several times in recent weeks and will continue to be updated to prevent common mistakes and upload difficulties. Use of GoToMeeting has been particularly helpful in resolving failed uploads. Recording a demonstration that may be accessed by users having difficulty might prove useful for supporting users in the future.

The NGDS has been designed as a decentralized collection of ‘Nodes’. As mentioned in the previous Task 12, Catalog Services for the Web (CSW) was selected as the means to share metadata between the nodes. CSW information for each dataset is accessible within GeoPortal of the GTDA system at: <http://geothermal.smu.edu/geoportal>. The SMU Node updates the CSW every time a file is uploaded with a timestamp. This updating is accomplished through submission of a specially formatted text file that each submitter must include with their data refreshes. The text file contains the dataset abstract, contact information, keywords, and other relevant metadata. This specially formatted text file is used by GTDA to generate the CSW records that are shared with the other NGDS nodes. While serviceable, the text file is difficult for the data submitters to read and understand. Additionally, there are numerous opportunities to edit program commands by accident when updating the file. A user friendly ‘form’ that would collect the metadata to populate the CSW information would have been preferable. As a plus, users will be able to change the metadata associated with their submissions at any time they choose, even without a data refresh.

Phase III. Execution of Data Maintenance Sustainability Plan

Task 14. Import of additional data into data repositories

As the team acquires more data of the same types as collected and published previously, the further quality controls, submission, and validation will be continuously performed and be supported by the software to minimize manual efforts. In case of new data types additional quality control modules and data submission modules may need to be designed, implemented and validated.

SMU, BEG, Cornell, GRC, Texas Tech, and UND each anticipate additional data, publications, or related resources submissions in the future. Upload routines were written within GTDA to accommodate continued uploads using the same format and layout as previously submitted, even after the project conclusion. In January of 2014, each of these groups tested the upload capability for new or revised data submissions. The Project Teams were successful with the exception of the GRC. Additional testing is underway, but the possibility of a firewall or network connectivity issue is suspected, given that SMU is able to sign on to the system using GRC credentials and upload the GRC catalog database without issue.

Going forward, the group may submit additional related resources, as well as datasets. Tier 1 type data (scans of maps, report, logs, etc.) may be tied to specific wells by name or API, governmental boundaries (state or county) or geographic coordinates (point, bounding box, or shape file).

The User Manual included in Appendix C describes how to upload data and SMU is in the process of augmenting the explanation through a series of screen shots in the event the person submitting data in the future is someone without prior experience with the project.

To minimize some of the manual efforts associated with data import, an ‘upload form’ was created for nearly every unique data submission type. (The submissions from Texas Tech and GRC come in as “.MDB” file types and thus do not lend themselves to an upload form.) The upload form not only provides some explanation, but it also documents the specific file layout expected and performs automated pre-checks of common mistakes, such as missing or invalid coordinates, non-numeric values in numeric fields, improper use of special characters, etc. before the user attempts to upload data. Once uploaded, the histogram capability described previously aids the data submitter in data validation by highlighting outliers which could potentially be erroneous data. The system can also generate a ‘null data’ report to identify empty fields prior to the decision to publish the data.

Task 15. Quality assurance and publication of additional data

The entire project is based on quality of data and its future usage by a variety of people with various backgrounds and needs. After the data are sent to the lead NGDS team there will be coordination with them for future data to enter the system and to follow similar quality checks that were used during the building process to remove duplication of data and confirm accuracy of the new data.

Since one of the goals of the project is to increase the use of data sets there will be journal publications and presentations made at professional meetings discussing the data that is available to increase interest in using it.

The discussion of *Task 2. Definition of data quality standards* and *Task 6. Quality control and improvement of data* within this section discuss the quality assurance process for the SMU Node in some detail. Additionally, SMU and Siemens Corporate Technology coordinated with personnel from the Arizona Geological Survey, the National Renewable Energy Laboratory and Boise State University to assure compliance with NGDS selected protocols and processes. In addition, the SMU Project Team shared the results of any work efforts deemed helpful to the other

complementary NGDS projects, including development of keyword lists for publication searching and elimination of duplicate publication scanning assignments.

The project team has prepared and presented dozens of papers to promote the understanding of the data collected and the NGDS in general. Please refer to Appendix B of this report containing a list of many of these, including several papers specifically about data quality and ongoing sustainability presented in 2012 and 2013 at GRC Annual Meetings and the Stanford Geothermal Workshops.

Task 16. System Environment Sustainability

During the development cycle, the systems environment will be housed at Southern Methodist University and maintained by the University's Information Technology Services (ITS) department. SMU's ITS will provide backup services of the software and the data records housed on storage devices at SMU. SMU makes use of off-site storage at an alternate location as part of the disaster recovery plan. SMU is capable of sustaining the systems environment (computer hardware, computer software, maintenance of the hardware and software, data backup services, Internet connectivity, system security, and disaster recovery services) beyond the project term, although there is not funding in place to do so at this time.

The SMU Geothermal Laboratory involved the Office of Information Technology early on in the project, with the realization that the task of adequately provisioning and maintaining the servers for public access was beyond the core strength area for the geothermal researchers. The servers used for this project are virtual machines (VMs) rather than physically discrete servers, which allows for more flexibility in sizing and adjustment of resources, more automated backup and disaster recovery processes, and more controlled secure access to the server and network connectivity. SMU's Office of Information Technology is prepared to maintain the existing system environment beyond the conclusion of the project, but there are no provisions in place for application software updates and/or modifications that may be warranted in the future. A copy of the Sustainability Plan developed by the SMU Geothermal Lab and SMU Information Technology Services is included in Appendix E.

The GTDA software documentation includes information directed at end users and at system support personnel. The information directed at end users provides detailed instructions for data submitters to contribute new data in the future. The system support documentation includes written copies of the database schema and SQL transform operations, should programming changes be necessary in the future. It also includes instructions for updating the portion of the interactive interface that is editable by members of the SMU Geothermal Laboratory. This documentation was also provided on the external hard drive provided to the DOE.

Finally, in the interest of ongoing sustainability, the decision was made to permit users to sign on to <http://geothermal.smu.edu> with 'guest' read-only privileges to reduce the need for ongoing administrative support of the application. User credentials will continue to be required for the data submitters, but the number of users is manageable enough to be within the scope of the SMU Geothermal Laboratory personnel capability.

Project Management and Reporting ***Project management and coordination:*** A project of this magnitude requires significant efforts for project management to ensure communication and efficient coordination among the project partners and technical teams. Responsibility for overall project management will be performed by Southern Methodist University (SMU) with delegation to subcontractor for development of the software framework and operations of data submission and to other subcontractors for compilation, collection and quality assurance of the their data. Communication will be done via regular teleconferences and a collaborative web-based authoring system. ***Reporting:*** All reports and other deliverables will be provided in accordance with the Federal Assistance Reporting Checklist, following the instructions included therein.

All reporting was completed thoroughly and in accordance with guidelines. Early in the project, the project consortium members codified guidelines for working together, including topics as how best to communicate with each other, how to credit authorship, how resolve differences should they arise, etc. The project received consistently high reviews at the annual DOE Peer Reviews, which is indicative of the overall project management. All organizations within the project consortium plan to continue the collaboration fostered through this project and look forward to working together in the future.

III. Summary of Accomplishments, Lessons Learned, Changes in Approach, and Opportunities for Further Study

A. KEY ACHIEVEMENTS

The primary purpose of this project: **Improve access to geothermal information and allow for new interpretation of data** by industry stakeholders, **thereby increasing its usefulness** for commercial geothermal energy development.

Before this project, information critical to project developers existed in multiple formats, layouts, units, in a combination of paper and electronic copy. Not only was it not possible to search the resources of multiple research institutions for related data, it was difficult to know who to contact for information on past and current projects for more information. For that matter, many project developers were unaware of the many different sources of geothermal information available.

The SMU Data Aggregation Project accomplished the primary purpose in numerous ways:

- ***Data Collection, Analysis, and Aggregation***
- ***Data Quality Advances***
- ***New Data Interpretations***
- ***Expanding the knowledge base of geothermal energy research***
- ***Improved Access to Publications***

Data Collection, Analysis, and Aggregation

This project collected millions of data records, both new and existing, from seven different institutions and transformed it into a format that the public can discover, search, download and analyze further (Table 2). The data are available by provider as well as aggregated and formatted into 10 content models, for a total of 20 different datasets. Appendix A of this report includes instructions for accessing the data that are now available to industry stakeholders, researchers, and the public. The table below summarizes the data available for download via WFS/WMS. Additional data can be obtained through login interactive use of the SMU Geothermal Node application at <http://geothermal.smu.edu>.

Table 2. List of data sets available and the estimated number of records per dataset. In spreadsheet terminology, this would equate to ‘rows’ rather than ‘cells’.

Description of Files Available for Download	Number of Records ³
Aggregated Well Data A listing of all sites with point location coordinates from SMU and all of the partner organizations. Bottom Hole Temperature (BHT), depth, heat flow and thermal conductivity values included when provided; see additional datasets below for more detailed heat flow and temperature data deliveries. Note: some sites are physical outcrops, rather than ‘wells’.	958,861
Borehole Temperature Observation in Content Model Format A nationwide aggregation of multiple data submissions containing temperature and depth data. Contains both equilibrium logged measurements and BHT values. Includes geothermal wells, oil and gas wells, water wells, test wells, etc.	171,759
Fluid Flux Injection and Disposal in Content Model Format Primarily data on Texas Oil and Gas wells provided by MLKay Technologies gathered from Texas Rail Road Commission information. The complete file contains nearly 10 million records, but is limited to 1,000,000 for WFS delivery.	9,815,860 (limited to 1 M via WFS)
Geologic Reservoir in Content Model Format Data on temperature, depth, net sand thickness, porosity, and other key geothermal reservoirs characteristics in Texas. Based on BEG analysis.	138
Heat Flow Observation in Content Model Format An aggregation of all heat flow values provided in the different submission files. Combines values based on equilibrium logged measurements from multiple depth intervals, values based on BHT data, and values from published data sources. Also includes sites without a heat flow value, but that have other important data useful in determining heat flow, such as ‘corrected’ temperature values, radiogenic heat production information, and thermal conductivity values.	125,011
Radiogenic Heat Production in Content Model Format An aggregation of radiogenic heat values provided in the different submission files. Includes data from gamma ray spectrometry measurements conducted by UND in their lab as well as in the field.	343
Thermal Conductivity Observation in Content Model Format An aggregation of all thermal conductivity values provided in the different submission files, including values based on lab measurements, estimates derived from COSUNA and lithology models, and those from published data sources.	52,245
Well Fluid Production Observation in Content Model Format Texas oil and gas industry fluid production data assembled by the BEG. Pulled from the BEG Consolidated Wells Contribution file, in the next section of files.	37,565

³ Most row counts were collected using GeoServer command to retrieve the number of features per dataset during the first week in February 2014. The database was known to have erroneously duplicated data submitted by Cornell; thus, the row counts for that dataset and several affected content models were adjusted *down* so as not to inflate them. The issue will be resolved soon, after which someone could use the command provided for each dataset in Appendix A to determine row counts in the database at later times.

Well Header Observation in Content Model Format An aggregation of all well locations, owner/operator, status, and date information represented in the various submission files. Note: Duplicate well entries were removed when possible. The same well may appear more than once with a different unique identifier if there was insufficient information to confirm an identical well from different submitting institutions.	922,448
Description of Files Available for Download	Number of Records
Well Log Observation in Content Model Format An aggregation of links to images of SMU Heat Flow database ‘field notes’, temperature depth curves, and other log types. Also contains links to images of thousands off-shore geophysical well logs assembled by Texas Tech University.	50,970
Well Test Observation in Content Model Format Texas oil and gas industry G10 injection well test data provided by MLKay Technologies gathered from Texas Rail Road Commission information.	949,089
SMU Heat Flow Database of Equilibrium Log Data and Geothermal Wells (includes sites from University of North Dakota Global Heat Flow Database) The dataset contains heat flow values from throughout the U.S. collected from over 12,600 locations. Site Heat Flow values are calculated based on temperatures at multiple depth intervals, site thermal conductivity and corrections (e.g., topographic effects). Some values are from literature sources whereas others are based on field logged equilibrium temperature measurements by the contributors. Links to thousands of temperature depth curves and associated field notes are available through either interactive use of the SMU NGDS Node or the Well Log Content Model.	14,768
SMU Heat Flow Database from BHT Data Used to update the Geothermal Map of North America for Google.org in 2011, the dataset of over 31,000 records contains ‘interval’ and ‘site’ heat flow calculations based on ‘corrected’ bottom hole temperature data from the oil and gas industry.	39,049
BEG at University of Texas Consolidated Well Database Contains raw and corrected BHT, depth, location, fluid production, pressure, etc. from over 42,000 wells throughout Texas. URLs to BEG’s extensive core log library are expected to be accessible as a related resource through the SMU Node after further integration.	42,961
BEG at University of Texas Reservoir Analysis From 55 different Texas geologic reservoirs, most with multiple depth ‘slices’. The analysis contains net sand volume and thickness calculations, which can be used by geothermal reservoir engineers. The submission file closely matches the content model layout for Geologic Reservoir.	138
Cornell University Heat Flow Database (NY and PA) The analysis includes over 8,900 BHT and depth points from throughout Pennsylvania and New York, including predicted temperatures at various depth and the estimated depths to various temperature levels for several thousand locations. Links to regional maps developed by Cornell will be accessible as related information through interactive use of the SMU Node of the NGDS.	8,919
Texas Tech University Off Shore Wells and BHT Database The collection includes raw and corrected BHT, depth, well status, location, ownership, etc. from over 8,100 different wells in the Gulf of Mexico. The collection contains a rich quantity of data from multiple depths and dates. In addition, thousands of scanned geophysical logs were provided by Texas Tech, which are available through interactive use of the SMU Node or from the Well Log Content Model.	25,317
University of North Dakota Radiogenic Heat Generation Database	

The collection contains data from over 350 specimens analyzed for heat generated by the Thorium, Potassium, or Uranium contents of the rocks. The submission file closely matches the content model layout for Radiogenic Heat Generation.	343
University of North Dakota Thermal Conductivity Database Samples were measured for thermal conductivity, which can be used to improve the accuracy of the heat flow calculations for the locations from which they were gathered. The submission file closely matches the content model layout for Thermal Conductivity Observations.	207
University of North Dakota Temperature at Depth Database Equilibrium Temperature-Depth log measurements of 245 wells in Iowa, Minnesota, Missouri, Nebraska, N. Dakota, and S. Dakota.	218

Data Quality Advances

The process of aggregating the data actually improved the quality of the separate collections. Because the data collections overlapped geographically, the SMU Project Team was able to detect previously unidentified discrepancies. Some of the discrepancies were the result of manual collection efforts, and others were the result of errors in data manipulation. By reviewing the data overlap, these errors came to light and were corrected. There is still opportunity for further work in this area, particularly as even more data are contributed to the National Geothermal Data System by other project teams.

New Data Interpretations

The data were interpreted in new ways as a result of the project collaboration efforts. For example, as part of the project, the data collected in Texas were incorporated into a reservoir analysis performed by the Bureau of Economic Geology (BEG) team. 55 high potential geological reservoirs were analyzed at various depth intervals (138 ‘slices’ in total) for geothermal favorability in Texas, an area ripe for coproduction of energy and enhanced geothermal energy projects. In addition to the extraction of hydrocarbons, opportunities to use the high geopressures and temperatures to power water desalination projects are of high interest. The oil and gas industry data contributed by MLKay Technologies, combined with the BEG porosity and permeability analysis, as well as the SMU heat flow analysis, is precisely the kind of information the project developers need to evaluate the commercial viability for these innovative projects.

Expanding the knowledge base of geothermal energy research

The graduate students and professors from the universities shared more than datasets – they exchanged methodologies and approaches to research. For example, the team studied the various approaches to correcting for the drilling effects on bottom hole temperature measurements; they learned how to interpret the corrected temperature data in combination with thermal conductivity of the rock layers and the radiogenic heat production of the basement rock to calculate heat flow; they evaluated methods for contouring maps of temperatures and heat flow as influenced by geological features. As a result of this project, a new generation of researchers is poised to carry the geothermal industry into new areas and opportunities.

Improved Access to Publications

Five different library collections were cataloged and thousands of publications were preserved through scanning. By applying optical character recognition, they are now full-text searchable.

The Geothermal Resources Council digital library was completely modernized and now includes abstracts and advanced searching. Instead of a researcher's prior work collecting dust on a shelf in a university lab, project developers will be able to learn that a publication on their topic of interest exists and be able to apply that information to their financial risk assessment. Future generations of researchers will be able to build on existing research they may not have ever discovered through review of published journals.

B. CHANGES IN APPROACH

Over the course of a project of this length and magnitude, there were certainly changes in approach, two of which are discussed in more detail below.

Response to Ongoing Operation Requirement

One of the more significant changes in approach occurred very shortly after the beginning of the project. At the time of the initial award, it was expected that SMU would be transferring the aggregated data to the NGDS, rather than maintaining it as a 'node' on the NGDS indefinitely. This change cascaded into other changes in approach. The overall impact to the project was not "good" or "bad", but a mix of both, with the positive outweighing the negative. The change precipitated modifications to the interactive application designed for the data submitters to accommodate public users. Some of the modifications to the interactive software were time intensive and likely delayed, but did not prevent the completion of other important tasks. Yet, we believe these enhancements will be useful to others beyond our project team and it serves to make the investment in the software benefit more users for a longer period of time. The shift to ongoing operation of the software also impacted the choice to migrate from commercial database products to free open source options. While the SMU Office of Information Technology has more experience with the commercial offerings, the open source platform is more fiscally responsible, given the lack of budget for maintaining the SMU node over time. The open source platform is not supported by professional maintenance services, but this lack is counterbalanced by the large overlap with other NGDS project teams using GeoServer and GeoPortal.

Data Formats Offered

Another example of a change in approach was the decision to make the aggregated data available using a content model layout ("tier 3"), AND the results as unaggregated "tier 2" data. While the content model format allows users and application developers to incorporate large amounts of data from multiple project teams, it presents limitations for certain users. To use an analogy, if you have a collection of 10 people from 10 countries all speaking their own language, a translation dictionary of common phrases is an excellent tool to allow them to communicate and exchange ideas. However, if two people speak the same language, telling them they may only use the dictionary of common phrases to communicate with each other would be frustrating and limiting. Despite many hours of involvement in helping to create and/or edit content models, some of the relevant data collected by the team still does not 'fit' into any of the content models available. In other instances, it can fit, but it's awkward. The heat flow content model, which this team helped create, was written with certain fields of information in mind. SMU has another 800 shallow gradient wells. They are too shallow for heat flow purposes, but are interesting for direct use or

geothermal heat pump applications. We can (and likely will) force fit the shallow gradient data into the heat flow content model by filling in numerous ‘required’ fields with -99999 or ‘nil:NA’. The data may also populate the borehole temperature observation, but that content model does not include geothermal gradient, which is a key geothermal parameter. In other instances, fields of data cannot be made to ‘fit’ into any existing content model. Cornell University researchers calculated temperatures-at-depth as well as depths-to-certain temperatures for all of New York and Pennsylvania. Estimated temperatures and depths are not a part of any content model. After wrestling with similar situations for every data submission, we took the approach of creating several new content models designed expressly for a particular type of data exchange (Radiogenic Heat, Thermal Conductivity, Geological Reservoir, Well Test, and Heat Flow) and populating those and 5 other content models using any and all aggregated data to the best of our ability, but recognizing that a user familiar with the data might find the format frustrating, we are also making the submission files available as discrete datasets. Because they are not in a standard layout, we have provided explanations of the data format and structure for all data provided that does not use a content model format.

C. LESSONS LEARNED

The project team accomplished an amazing amount of data collection and data analysis. We are proud of the accomplishments and hope that our efforts become the foundation for expanded use of geothermal energy in this country. Yet, as with any large, complex project, there are things that we would have done differently, given the benefit of hindsight. Likewise, there are some things we were repeatedly thankful for having considered early in the project. Lessons we learned, which we would hope others could benefit from if considering a similar project include:

1. *Evaluate strengths and weaknesses and plan accordingly*

The SMU Geothermal Laboratory did not have extensive server configuration or management experience. Nor do we have sophisticated system disaster recovery plans and procedures, redundant network connectivity, system security expertise, physically secured server rooms, Linux experience, etc. We recognized early on that the Office of Information Technology (OIT) was better suited to managing the server environment and have remarked numerous times since that adjusting the Statement of Project Objectives to reflect that realization was something we would do again.

2. *Plan for problems before they arise*

The team codified plans for working together as a team during the initial group calls. We discussed and agreed as a group upon how to communicate with each other, how to order the list of 8 participating organizations and/or author names, how to resolve difficulties should they arise, etc. This team was large and diversified in terms of geography and personalities. Yet, the group had consensus on the expectations from one another and worked well together throughout the project. We also codified the arrangement between the Geothermal Laboratory and SMU OIT early on in the project, which clarified to all involved performance of certain responsibilities. An instance where prior planning paid off was Siemens mirroring of the system in Princeton during the development process.

Generally, this is done to protect the production system from outages caused by untested new code, but the two environments also serve as redundancy in case of outages. In this instance, it proved helpful when poor system performance on the production server in Dallas could not be diagnosed, despite elimination of numerous possible causes and the involvement of outside expertise. This poor system performance caused us to fall behind schedule at a critical juncture in 2013. Being able to wipe the Dallas server and replace it from a restored dump of the development server proved a good resolution to the problem.

3. *Architecture should not change over the course of this type of project*

While changes to the system were required throughout the entire development, the basic system architecture remained stable once decided on by the end of 2010.

4. *Be wary of assumptions*

Most miscommunications arose from assumptions. The data submitters assumed that because Siemens asked for clarification about the meaning behind a number of fields and parameters, they must understand those about which no questions were asked. That was not necessarily the case. In hindsight, the data submitters should have provided a detailed explanation of the field descriptions with the initial data submission. We should have asked for these to have been included in the database schema documentation itself. Additionally, we should have asked for a more thorough explanation of the data schema in layman's terminology. While we understood that there was a staging and a core repository, we assumed that all information in the staging repository became part of the aggregated 'core' database, and this was not the case. There are in fact fields of submitted data in the staging repository that are not part of the core database.

5. *Require data submitters to equate 'their' fields to standardized field names*

Doing this again, we would ask each of our data providers to provide a spreadsheet or database table 'mapping' each of their submission field names to the list of possible content model field names. At the time the database schema was being determined, the content models were not far enough along to have done this for all of the datasets. However, the datasets that mapped most cleanly were those that were so different, they didn't populate any content model until one was written specifically for that dataset. In contrast, the datasets of consolidated well data, consisting of temperatures, corrected temperatures, various depth intervals, location identifiers, etc. populate 4 or more content models from a single submission file. Understanding which fields should go where becomes complex quickly and we relied heavily on Siemens to determine the mapping for us. In hindsight, the data submitters should have done this task. None of them would have liked it, as it is tedious and can be difficult, but it would have resulted in fewer errors and would have ensured that all submitted data 'fit' somewhere.

6. *Mapping to Content Models requires both knowledge of the raw data and familiarity with the content models*

SMU and MLKay Technologies actually did do the mapping to the content models for the MLKay Technologies submission, as was recommended in the previous point for all submissions. It took many attempts wrestling with it over the telephone and email before switching to a face-to-face meeting over 2 day period removed from normal distractions. While scheduling the face-to-face meeting was a bit challenging, the two day session turned out to be considerably less time intensive than detecting and correcting errors in mapping for the other contributions where Siemens did the mapping without explicit guidance. Again in hindsight, we have learned that to effectively map the data to content models, you must have someone extremely familiar with the raw data submission (abbreviations, units of measure used, field names, irregularities, collection techniques, etc.) and also someone extremely familiar with the fields names and descriptions of each pertinent content model. Based on our experiences, we find it unlikely that many data submitters will want to read the content model documentation thoroughly enough to be able to do this task independently. Simplification of the content model explanations to remove all information technology terminology might help. For example, the explanations could state if a field is required versus optional, rather than a cardinality of 1 versus .1. The latter expression may be more accurate from a programming perspective, but a cardinality of 1 doesn't mean 'required' to many people outside of information technology.

7. *Scanning a library of publications was difficult for non-librarians*

The labs with library publications to contribute all found the process required significantly more resources than anticipated. Each item was assessed to determine if it was of sufficient value to be worth scanning and/or if it was already available online. Bindings were removed. Copyright status was researched. The scanned item had to be validated for clarity, proper page sequence, and optical character recognition. Large format items were sent for outside scanning and the physical and electronic copies had to be rejoined with the portions scanned in-house. Bindings had to be repaired or replaced, again requiring coordination with outside service companies. File names had to be correlated to the unique publication ID in the bibliography and individuals had to keep track of whether an item was copyrighted and thus not for general distribution at the time of saving the file. If we had to do again, we would still catalog the library early on, but we would make note of the specific shelf location for each item and would label each item with a unique ID at the time of cataloging. The unique identifier would correlate to the physical organization of the items in the library, rather than to titles or authors, etc. unless that was the basis for the physical organization. The initial cataloging effort would also have been the best time to determine and record the copyright status. Instead, the initial cataloging done by most of the groups was limited to the information needed for a bibliography listing, causing difficulty locating specifically assigned publications later and/or determining the copyright status. Related to the first lesson learned about understanding strengths and weaknesses, the SMU group did shift to outsourcing the binding removal, scanning, OCR, and binding repair tasks, but not until late in the project. *For future DOE projects involving large amounts of document scanning specific guidance from experts (perhaps OSTI) with regard to best practices for cataloging, copyright research, OCR optional settings, etc. should be standard.* Recommendations for companies qualified to do outside scanning of mixed items (bound, unbound, stapled, multiple paper sizes and types, etc.) would have been helpful.

8. *Be creative and flexible*

When it became apparent late in the project that some data within the staging repository was not pulled into the core database and was thus unavailable to populate a content model field, we came up with the option to provide the submitted datasets as a Tier 2 data delivery. This option provided a number of benefits, such as aiding in the validation of what was actually submitted against the content model population. It also allowed users to see the full data submission, even if a field is not part of any content model. *If content models change dramatically over the next several years, these tier 2 data deliveries could be reformatted into the revised content model formats and submitted through the Geothermal Data Repository (GDR) server.*

9. *A deadline for data submitter validation would have been helpful*

The group did institute a ‘freeze date’ after which submission formats were not to change. However, there was not a hard deadline for the data submitter to validate that their data was populating the content model fields properly. SMU did much of this validation for the other groups, which led to the need for changes in the database schema, and in some instances, in the data submission formats, well after the freeze date. This requirement would have been challenging to have enforced, given that the system availability issues became untenable at the same time the content model population was being completed, creating a very narrow window for the data validation activity. The data submitters were far more interested in the data collection and data analysis than in the population of the content model fields, making it difficult to engage them in the data validation exercise. This is one ‘lesson learned’ for which there is no recommendation to prevent reoccurrence.

10. *Bi-weekly project updates with the DOE were valuable*

At times, bi-weekly project updates seemed a distraction from the project tasks, but without exception, every update call was valuable for sharing and/or generating ideas. Similarly, written quarterly reports were very time intensive to prepare, and no doubt to review; however, by forcing a review of all outstanding tasks, it helped ensure they were on track for completion.

11. *Data Quality Assessment is never ‘done’*

We found there was no substitute for the human eye, curiosity, and ‘playing with’ the data. We are still finding things to update and likely will indefinitely, given that new data is added to the collection continuously. Some corrections could require schema changes, if mapping is incorrect, and that will be difficult to accommodate in this system.

12. *Metadata submission should be simplified*

CSW information was collected late in process in the form of text files for each group to edit with their abstracts, keywords, responsible party information etc. This should have been done with a form or other technique for masking the commands needed from the data submitters. The fact that the data submitters have to opportunity to inadvertently delete a

required line within the text files transmitting the CSW information is of particular concern. Other NGDS project teams are likely using a form approach, which the SMU team should have replicated.

13. *Overlapping data from State Geological Surveys needs further review*

SMU reached out to numerous state geological surveys, with the knowledge that many would retrieve information from SMU's legacy heat flow database to submit as part of the state collection effort. Some augmented this information with additional data they had available within the state survey before submission to the NGDS, which is beneficial. *However, we suspect that many of those groups do not know all of the specifics for data they may have gathered from SMU and/or other sources. For instance, if they retrieved information from SMU's legacy system, the coordinates of wells would have been provided in NAD27 spatial referencing system. We have now updated our coordinates to WGS84, as required for the NGDS content models. If the state surveys did not know the spatial referencing system, they may have assumed it to be WGS84, leading the display of the data appearing to be in different locations and perhaps double the number of wells. Overlapping data collected from all of the NGDS nodes needs further review to detect and correct this and similar issues.*

14. *It is easy to work with people whom you like and respect*

This project team works well together and produced more than the sum of our parts. We would welcome the opportunity to continue working together in the future.

D. OPPORTUNITIES FOR ADDITIONAL RESEARCH OR ACTIVITIES

As mentioned in the lessons learned above, there are opportunities to continue refining data quality within our group as well as in combination with data coming in to the other NGDS nodes. We believe this to be an ongoing effort. We also would like to pursue Fairway Analysis using the amalgamation of data from the SMU Node as well as the state geological surveys to study specific reservoirs more closely. Additionally, we believe outreach efforts to inform the public of the resources would be worthwhile. Finally, Appendix I addresses known system limitations, including the the population of certain content model fields that warrant additional attention.

Appendix A: Links to Datasets on <http://geothermal.smu.edu>

When users access <http://geothermal.smu.edu>, they are presented with a welcome screen that offers them a choice of two avenues for accessing data: They may **'login'** to interactively view and query data sets and publications, or they may download complete data sets using the links from the **'download'** option.

The **'login'** option displays data through a map based interface and allows the user to filter data based on a number of parameters including location (state, county or a user defined 'bounding box'), temperature, depth, contributing organization, etc. Data meeting the filter criteria may then be exported for download from within the interactive application. The interactive application also contains links to related log file images, researcher field notes, referenced publications, temperature-depth curves, and provides the ability to view database fields in a histogram.

The **'download'** option offers complete unfiltered data sets made available in a number of formats suitable for download to a spreadsheet or consumable by applications that understand the Web Feature Service (WFS) and Web Map Service (WMS) protocols from the Open Geo Consortium.

Once users select the 'download' option, they are presented with overview information pertaining to the file types offered and caution statements regarding the large size of some of the datasets available. The page contains a table describing each of the datasets available for download as well as links to access the files. That page is replicated below. This appendix also contains a document listing the full WFS/WMS commands by dataset.

COPY OF DOWNLOADFILESBUTTONPAGE.HTM¹

Important Note about Large File Downloading

Please note that some datasets available are large and may be slow to retrieve, and/or are so voluminous that they might crash your web browser. This does not necessarily indicate a server failure. The "Full File as CSV" links below indicate if a file is particularly large. The CSV files are compressed, however.

Usually, large datasets are better retrieved using web clients (e.g. GNU Wget, cURL, etc.), or by using a tool to retrieve and display the data (e.g. Geothermal Prospector, or ArcGIS). If you have a tool which you wish to use, you may test its ability to consume the data by downloading the first 50 records within the dataset by using our "Sample 50" links in the table below before attempting to download the entire dataset.

About the Data available for Download

The project participants each submitted thousands of records of data into the SMU Node of the NGDS (SMU operates one of several [NGDS](#) 'nodes'). These data collections were aggregated to populate ten of the different data delivery '[content models](#)'. Content models are a standardized means of exchanging a particular type of data across the NGDS, such as information about heat flow or borehole temperature values. Information on the NGDS formats, including descriptions of the specific file layouts for all of the 30+ different possible content models, may be found at <http://schemas.usgin.org/models/>. Because the content models require certain fields to be populated, records without that type of information contain 'nil:missing' or '-99999', which account for some of the very large file sizes.

¹ <http://geothermal.smu.edu/static/DownloadFilesButtonPage.htm>, pulled on April 7, 2014

Additionally, the un-aggregated data contribution files are available for download through links below. Unlike the content model formatted data, most of the data submission files are not in a standard format, and are in the units of measure and columnar layouts as the original contributor submitted them. Because some of these datasets contain additional data/information that may be of interest to project developers or researchers that were not used in populating the content models, the decision was made to make them available as discrete datasets.

In all, 20 datasets are available for download:

- The “Aggregated Well Data” file lists all locations included within the other collections, along with some temperature, depth, heat flow and thermal conductivity information;
- The next 10 files are delivered in a ‘content model’ format specified for data exchange via the National Geothermal Data System (NGDS) containing data from across the project team;
- The next 9 are submitted data contribution files. Because most of these are not in a content model format, we have provided documentation of the column order and content for each file in a Column Heading Description PDF file.

NOTE: The full MLKay Technologies Texas Oil and Gas data submission of over 800,000 wells is not available for download as a discrete dataset. However, it was incorporated into the Well Header Observation content model, the Fluid Flux Injection and Disposal content model, the Borehole Temperature Observation content model, the Well Test content model, and the Aggregated Well Data file. Additional data from MLKay Technologies are available for download/export from within the interactive application reachable by selecting ‘login’ from the main welcome screen.

Selecting a Download File Format

Most Files are downloaded using a protocol called Web Feature Services (WFS) and/or Web Map Services (WMS). If you would like to view and manipulate the data in a spreadsheet program such as Microsoft® Excel®, select the option to download a ‘zipped’ version of the dataset as a CSV (Comma Separated Value) file. (These CSV files are refreshed periodically, and every attempt is made to keep them current, but they are not ‘instantly’ updated with each new data submission.) The column headings order and meaning are described in either the ‘about the dataset’ PDF document or through the content model explanation links. The XML format is not designed to be ‘human intelligible’; rather it assumes you have a web client program designed to understand XML format. The XML Schema describes the layout of the XML files to the web client. The Web Map Service (WMS) returns an image of the data points usable as a layer upon a map. The links provided return the capabilities of the service for use in programs such as Esri’s® ArcGIS and others.

Notices

There are no warranties expressed or implied. Decisions based on the information available should be made after a thorough due diligence by the user.

Microsoft and Excel are registered trademarks of Microsoft Corporation in the United States and/or other countries. Esri and ArcGIS are registered trademarks of Environmental Systems Research Institute, Inc. (Esri) in the United States and/or other countries.

Description of Files Available for Download	File Download Links	File Format Explanation	File Download Links using Web Map Service (WMS)
Aggregated Well Data A listing of all sites with point location coordinates from SMU and all of the partner organizations. Bottom Hole Temperature (BHT), depth, heat flow and thermal conductivity values provided; see additional datasets below for more detailed heat flow and temperature data	Full File as CSV (>200MB) Sample 50 Rows CSV Full File as XML	About the Dataset.PDF XML Schema (DescribeFeatureType)	WMS ver. 1.1.1 WMS ver. 1.3

deliveries. Note: some sites are outcroppings, rather than 'wells'.			
Content Model Files Available for Download			
Borehole Temperature Observation in Content Model Format A nationwide aggregation of multiple data submissions containing temperature and depth data. Contains both equilibrium logged measurements and BHT values. Includes geothermal wells, oil and gas wells, water wells, test wells, etc.	Full File as CSV (>73 MB) Sample 50 Rows CSV Full File as XML	Content Model Format XML Schema (DescribeFeatureType)	WMS ver. 1.1.1 WMS ver. 1.3
Fluid Flux Injection and Disposal in Content Model Format Primarily data on Texas Oil and Gas wells provided by MLKay Technologies gathered from Texas Rail Road Commission information. WARNING: The complete file contains nearly 10 M records.	First 1 Million Records as CSV (>VERY large) Sample 50 Rows CSV Full File as XML	Content Model Format XML Schema (DescribeFeatureType)	WMS ver. 1.1.1 WMS ver. 1.3
Geologic Reservoir in Content Model Format Data on temperature, depth, net sand thickness, porosity, and other key geothermal reservoirs characteristics in Texas. Based on BEG analysis.	Full File as CSV Sample 50 Rows CSV Full File as XML	Content Model Format XML Schema (DescribeFeatureType)	WMS ver. 1.1.1 WMS ver. 1.3
Heat Flow Observation in Content Model Format An aggregation of all heat flow values provided in the different submission files. Combines values based on equilibrium logged measurements from multiple depth intervals, values based on BHT data, and values from published data sources. Also includes sites without a heat flow value, but that have other important data useful in determining heat flow, such as 'corrected' temperature values, radiogenic heat production information, and thermal conductivity values.	Full File as CSV (>60 MB) Sample 50 Rows CSV Full File as XML	Content Model Format XML Schema (DescribeFeatureType)	WMS ver. 1.1.1 WMS ver. 1.3

Radiogenic Heat Production in Content Model Format An aggregation of radiogenic heat values provided in the different submission files. Includes data from gamma ray spectrometry measurements conducted by UND in their lab as well as in the field. Also includes any heat generation values from the heat flow determinations by SMU, Cornell, or UND.	Full File as CSV Sample 50 Rows CSV Full File as XML	Content Model Format XML Schema (DescribeFeatureType)	WMS ver. 1.1.1 WMS ver. 1.3
Thermal Conductivity Observation in Content Model Format An aggregation of all thermal conductivity values provided in the different submission files, including values based on lab measurements, estimates derived from COSUNA and lithology models, and those from published data sources.	Full File as CSV (>25 MB) Sample 50 Rows CSV Full File as XML	Content Model Format XML Schema (DescribeFeatureType)	WMS ver. 1.1.1 WMS ver. 1.3
Well Fluid Production Observation in Content Model Format Texas oil and gas industry fluid production data assembled by the BEG. Pulled from the BEG Consolidated Wells Contribution file, in the next section of files.	Full File as CSV (>21 MB) Sample 50 Rows CSV Full File as XML	Content Model Format XML Schema (DescribeFeatureType)	WMS ver. 1.1.1 WMS ver. 1.3
Well Header Observation in Content Model Format An aggregation of all well locations, owner/operator, status, and date information represented in the various submission files. Note: Duplicate well entries were removed when possible. The same well may appear more than once with a different unique identifier if there was insufficient information to confirm an identical well from different submitting institutions.	Full File as CSV (>350 MB) Sample 50 Rows CSV Full File as XML	Content Model Format XML Schema (DescribeFeatureType)	WMS ver. 1.1.1 WMS ver. 1.3
Well Log Observation in Content Model Format An aggregation of links to images of SMU Heat Flow database ‘field notes’, temperature depth curves, and other log types. Also contains links to images of thousands off-shore geophysical well logs assembled by Texas Tech University.	Full File as CSV (>33 MB) Sample 50 Rows CSV Full File as XML	Content Model Format XML Schema (DescribeFeatureType)	WMS ver. 1.1.1 WMS ver. 1.3
Well Test Observation in Content Model Format Texas oil and gas industry G10 injection well test data provided by MLKay Technologies gathered from Texas Rail Road Commission information.	Full File as CSV (>480 MB) Sample 50 Rows CSV Full File as XML	Content Model Format XML Schema (DescribeFeatureType)	WMS ver. 1.1.1 WMS ver. 1.3
Contribution/Submission Files Available for Download (‘Tier 2’ Structured Data; not in Content Model Format)			
SMU Heat Flow Database of Equilibrium Log Data and Geothermal Wells (note: includes sites from the Global Heat Flow Database from the University of North Dakota) Over 12,000 locations are included in this dataset. Site Heat Flow values are calculated based on temperatures at multiple depth intervals, site thermal conductivity determinations and corrections (such as topographic effects). Some values are from literature sources whereas others are based on field logged equilibrium temperature measurements by the contributors. Links to thousands of temperature depth curves and associated field notes are available through either interactive use of the SMU NGDS Node or the Well Log Content Model.	Full File as CSV Sample 50 Rows CSV Full File as XML	About the Dataset.PDF XML Schema (DescribeFeatureType)	WMS ver. 1.1.1 WMS ver. 1.3

SMU Heat Flow Database from BHT Data Used to update the Geothermal Map of North America for Google.org, the dataset of over 31,000 records contains 'interval' and 'site' heat flow calculations based on 'corrected' bottom hole temperature data from the oil and gas industry.	Full File as CSV Sample 50 Rows CSV Full File as XML	About the Dataset.PDF XML Schema (DescribeFeatureType)	WMS ver. 1.1.1 WMS ver. 1.3
BEG at University of Texas Consolidated Well Database Contains raw and corrected BHT, depth, location, fluid production, pressure, etc. from ~40,000 wells throughout Texas. URLs to BEG's extensive core log library are expected to be accessible as a related resource through the SMU Node after further integration.	Full File as CSV (>16 MB) Sample 50 Rows CSV Full File as XML	About the Dataset.PDF XML Schema (DescribeFeatureType)	WMS ver. 1.1.1 WMS ver. 1.3
BEG at University of Texas Reservoir Analysis From 55 different Texas geologic reservoirs, most with multiple depth 'slices'. The analysis contains net sand volume and thickness calculations, which can be used by geothermal reservoir engineers. The submission file closely matches the content model layout for Geologic Reservoir.	Full File as CSV Sample 50 Rows CSV Full File as XML	About the Dataset.PDF XML Schema (DescribeFeatureType)	WMS ver. 1.1.1 WMS ver. 1.3
Cornell University Heat Flow Database (NY and PA) The analysis includes over 8,900 BHT and depth points from throughout Pennsylvania and New York, including predicted temperatures at various depth and the estimated depths to various temperature levels for several thousand locations. Links to regional maps developed by Cornell will be accessible as related information through interactive use of the SMU Node of the NGDS.	Full File as CSV Sample 50 Rows CSV Full File as XML	About the Dataset.PDF XML Schema (DescribeFeatureType)	WMS ver. 1.1.1 WMS ver. 1.3
Texas Tech University Off Shore Wells and BHT Database The collection includes raw and corrected BHT, depth, well status, location, ownership, etc. from 8,200 different wells in the Gulf of Mexico. The collection contains a rich quantity of data from multiple depths and dates, with over 25,000 rows of data entries. In addition, thousands of scanned geophysical logs were provided by Texas Tech, which are available through interactive use of the SMU Node or from the Well Log Content Model.	Full File as CSV (>32 MB) Sample 50 Rows CSV Full File as XML	About the Dataset.PDF XML Schema (DescribeFeatureType)	WMS ver. 1.1.1 WMS ver. 1.3
University of North Dakota Radiogenic Heat Generation Database The collection contains data from over 350 specimens analyzed for heat generated by the Thorium, Potassium, or Uranium contents of the rocks. The submission file closely matches the content model layout for Radiogenic Heat Generation.	Full File as CSV Sample 50 Rows CSV Full File as XML	About the Dataset.PDF XML Schema (DescribeFeatureType)	WMS ver. 1.1.1 WMS ver. 1.3
University of North Dakota Thermal Conductivity Database 217 samples were measured for thermal conductivity, which can be used to improve the accuracy of the heat flow calculations for the locations from which they were gathered. The	Full File as CSV Sample 50 Rows CSV Full File as XML	About the Dataset.PDF XML Schema (DescribeFeatureType)	WMS ver. 1.1.1 WMS ver. 1.3

submission file closely matches the content model layout for Thermal Conductivity Observations.			
University of North Dakota Temperature at Depth Database Equilibrium Temperature-Depth log measurements of 245 wells in Iowa, Minnesota, Missouri, Nebraska, N. Dakota, and S. Dakota.	Full File as CSV Sample 50 Rows CSV Full File as XML	About the Dataset.PDF XML Schema (DescribeFeatureType)	WMS ver. 1.1.1 WMS ver. 1.3

NOTE REGARDING ‘FULL FILE’ AVAILABLE AS CSV: In the tables that follow, there is a GeoServer command to generate the ‘full file’ in WFS as a CSV file. The phrase ‘full file’ is somewhat misleading in that GeoServer is currently configured to deliver the full dataset using WFS *up to* 10,000 records. Because this GeoServer command generates the datasets ‘on the fly’, the limit helps maintain satisfactory performance for the end user and for the server. Several of the datasets in the collection are well over 100,000 records, with some in excess of 1,000,000. The GeoServer configuration may be modified by systems operations personnel, but not by the end user. As an alternative, the ‘Link to Static Zipped CSV File’ option appearing in the tables leads to a static version of the entire dataset available for CSV download. SMU personnel may periodically update the static full csv files available for download by running a script on the server.

APPENDIX A: GEOSERVER LINKS

GeoServer Links for Aggregated Well Data.....	8
GeoServer Links for Aggregated Borehole Temperature Observation in Content Model Format.....	9
GeoServer Links for Aggregated Fluid Flux Injection and Disposal in Content Model Format.....	10
GeoServer Links for Aggregated Geologic Reservoir in Content Model Format	11
GeoServer Links for Aggregated Heat Flow Observation in Content Model Format	12
GeoServer Links for Aggregated Radiogenic Heat Generation in Content Model Format	13
GeoServer Links for Aggregated Thermal Conductivity Observation in Content Model Format.....	14
GeoServer Links for Aggregated Well Fluid Production Observation in Content Model Format	15
GeoServer Links for Aggregated Well Header Observation in Content Model Format.....	16
GeoServer Links for Aggregated Well Log Observation in Content Model Format.....	17
GeoServer Links for Aggregated Well Test Observation in Content Model Format	18
GeoServer Links for SMU Heat Flow Data from Equilibrium Logs.....	19
GeoServer Links for SMU Heat Flow Data from BHT	20
GeoServer Links for BEG Master Wells Database.....	21
GeoServer Links for BEG Geologic Reservoir Analysis.....	22
GeoServer Links for Cornell NY and PA Heat Flow Database.....	23
GeoServer Links for Texas Tech Off Shore Wells and BHT	24
GeoServer Links for UND Radiogenic Heat Generation.....	25
GeoServer Links for UND Thermal Conductivity.....	26
GeoServer Links for UND Temperature at Depth	27

GEOSERVER LINKS FOR AGGREGATED WELL DATA

File Name Used in GTDA for WMS: gtda and/or aggregatedhf

File Name Used in GTDA for WFS: gtda:wells

Dataset delivered in Content Model Format?: No

WMS Ver. 1.1.1 Link	http://geothermal.smu.edu:9000/geoserver/wms?service=wms&version=1.1.1&request=GetCapabilities&namespace=gtda
WMS Ver. 1.3 Link	http://geothermal.smu.edu:9000/geoserver/wms?service=wms&version=1.3&request=GetCapabilities&namespace=gtda
WFS XML Link	http://geothermal.smu.edu:9000/geoserver/gtda/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=gtda:wells
WFS Full File As CSV Link (up to 10,000 records)	http://geothermal.smu.edu:9000/geoserver/gtda/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=gtda:wells&outputFormat=csv
Link to Static Zipped CSV File (entire dataset; ~ 75 MB)	http://geothermal.smu.edu/static/DatasetsZipped/core.surface_site_county_state_materialized_view.zip
WFS Sample 50 as CSV Link	http://geothermal.smu.edu:9000/geoserver/gtda/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=gtda:wells&outputFormat=csv&maxFeatures=50
XLM Schema Describe Feature Type Link	http://geothermal.smu.edu:9000/geoserver/gtda/ows?service=WFS&version=1.0.0&request=DescribeFeatureType&typeName=gtda:wells
'About the Dataset.PDF' or Link to Content Model Information	http://geothermal.smu.edu/static/DescribeColumns/AggregatedWellData.pdf
Number of Records 'hits' command	http://geothermal.smu.edu:9000/geoserver/gtda/ows?service=WFS&version=1.1.0&request=GetFeature&typeName=gtda:wells&resultType=hits

GEOSERVER LINKS FOR AGGREGATED BOREHOLE TEMPERATURE OBSERVATION IN CONTENT MODEL FORMAT

File Name Used in GTDA for WMS: aasg-borehole

File Name Used in GTDA for WFS: aasg-borehole:BoreholeTemperature

Dataset delivered in Content Model Format?: Yes

WMS Ver. 1.1.1 Link	http://geothermal.smu.edu:9000/geoserver/wms?service=wms&version=1.1.1&request=GetCapabilities&namespace=aasg-borehole
WMS Ver. 1.3 Link	http://geothermal.smu.edu:9000/geoserver/wms?service=wms&version=1.3&request=GetCapabilities&namespace=aasg-borehole
WFS XML Link	http://geothermal.smu.edu:9000/geoserver/aasg-borehole/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=aasg-borehole:BoreholeTemperature
WFS Full File As CSV Link (up to 10,000 records)	http://geothermal.smu.edu:9000/geoserver/aasg-borehole/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=aasg-borehole:BoreholeTemperature&outputFormat=csv
Link to Static Zipped CSV File (entire dataset; ~ 16.3 MB)	http://geothermal.smu.edu/static/DatasetsZipped/core.template_borehole_materialized.zip
WFS Sample 50 as CSV Link	http://geothermal.smu.edu:9000/geoserver/aasg-borehole/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=aasg-borehole:BoreholeTemperature&outputFormat=csv&maxFeatures=50
XLM Schema Describe Feature Type Link	http://geothermal.smu.edu:9000/geoserver/aasg-borehole/ows?service=WFS&version=1.0.0&request=DescribeFeatureType&typeName=aasg-borehole:BoreholeTemperature
‘About the Dataset.PDF’ or Link to Content Model Information	http://schemas.usgin.org/models/#boreholetemperature
Number of Records 'hits' command	http://geothermal.smu.edu:9000/geoserver/aasg-borehole/ows?service=WFS&version=1.1.0&request=GetFeature&typeName=aasg-borehole:BoreholeTemperature&resultType=hits

GEOSERVER LINKS FOR AGGREGATED FLUID FLUX INJECTION AND DISPOSAL IN CONTENT MODEL FORMAT

File Name Used in GTDA for WMS: aasg-fluidflux

File Name Used in GTDA for WFS: aasg-fluidflux:FluidFluxInjection

Dataset delivered in Content Model Format?: Yes

WMS Ver. 1.1.1 Link	http://geothermal.smu.edu:9000/geoserver/wms?service=wms&version=1.1.1&request=GetCapabilities&namespace=aasg-fluidflux
WMS Ver. 1.3 Link	http://geothermal.smu.edu:9000/geoserver/wms?service=wms&version=1.3&request=GetCapabilities&namespace=aasg-fluidflux
WFS XML Link	http://geothermal.smu.edu:9000/geoserver/aasg-fluidflux/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=aasg-fluidflux:FluidFluxInjection
WFS Full File As CSV Link (up to 10,000 records)	http://geothermal.smu.edu:9000/geoserver/aasg-fluidflux/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=aasg-fluidflux:FluidFluxInjection&outputFormat=csv&maxFeatures=1000000
Link to Static Zipped CSV File (entire dataset; ~ 102 MB)	http://geothermal.smu.edu/static/DatasetsZipped/core.template_fluidflux_materialized.zip
WFS Sample 50 as CSV Link	http://geothermal.smu.edu:9000/geoserver/aasg-fluidflux/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=aasg-fluidflux:FluidFluxInjection&outputFormat=csv&maxFeatures=50
XLM Schema Describe Feature Type Link	http://geothermal.smu.edu:9000/geoserver/aasg-fluidflux/ows?service=WFS&version=1.0.0&request=DescribeFeatureType&typeName=aasg-fluidflux:FluidFluxInjection
‘About the Dataset.PDF’ or Link to Content Model Information	http://schemas.usgin.org/models/#fluidfluxinjection
Number of Records 'hits' command	http://geothermal.smu.edu:9000/geoserver/aasg-fluidflux/ows?service=WFS&version=1.1.0&request=GetFeature&typeName=aasg-fluidflux:FluidFluxInjection&resultType=hits

GEOSERVER LINKS FOR AGGREGATED GEOLOGIC RESERVOIR IN CONTENT MODEL FORMAT

File Name Used in GTDA for WMS: aasg-reservoir

File Name Used in GTDA for WFS: aasg-reservoir:GeologicReservoir

Dataset delivered in Content Model Format?: Yes

WMS Ver. 1.1.1 Link	http://geothermal.smu.edu:9000/geoserver/wms?service=wms&version=1.1.1&request=GetCapabilities&namespace=aasg-reservoir
WMS Ver. 1.3 Link	http://geothermal.smu.edu:9000/geoserver/wms?service=wms&version=1.3&request=GetCapabilities&namespace=aasg-reservoir
WFS XML Link	http://geothermal.smu.edu:9000/geoserver/aasg-reservoir/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=aasg-reservoir:GeologicReservoir
WFS Full File As CSV Link (up to 10,000 records)	http://geothermal.smu.edu:9000/geoserver/aasg-reservoir/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=aasg-reservoir:GeologicReservoir&outputFormat=csv
Link to Static Zipped CSV File (entire dataset; ~ <1 MB)	http://geothermal.smu.edu/static/DatasetsZipped/core.template_reservoir_materialized.zip
WFS Sample 50 as CSV Link	http://geothermal.smu.edu:9000/geoserver/aasg-reservoir/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=aasg-reservoir:GeologicReservoir&outputFormat=csv&maxFeatures=50
XLM Schema Describe Feature Type Link	http://geothermal.smu.edu:9000/geoserver/aasg-reservoir/ows?service=WFS&version=1.0.0&request=DescribeFeatureType&typeName=aasg-reservoir:GeologicReservoir
‘About the Dataset.PDF’ or Link to Content Model Information	http://schemas.usgin.org/models/#geologicreservoir
Number of Records 'hits' command	http://geothermal.smu.edu:9000/geoserver/aasg-reservoir/ows?service=WFS&version=1.1.0&request=GetFeature&typeName=aasg-reservoir:GeologicReservoir&resultType=hits

GEOSERVER LINKS FOR AGGREGATED HEAT FLOW OBSERVATION IN CONTENT MODEL FORMAT

File Name Used in GTDA for WMS: aasg-heatflow

File Name Used in GTDA for WFS: aasg-heatflow:HeatFlow

Dataset delivered in Content Model Format?: Yes

WMS Ver. 1.1.1 Link	http://geothermal.smu.edu:9000/geoserver/wms?service=wms&version=1.1.1&request=GetCapabilities&namespace=aasg-heatflow
WMS Ver. 1.3 Link	http://geothermal.smu.edu:9000/geoserver/wms?service=wms&version=1.3&request=GetCapabilities&namespace=aasg-heatflow
WFS XML Link	http://geothermal.smu.edu:9000/geoserver/aasg-heatflow/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=aasg-heatflow:HeatFlow
WFS Full File As CSV Link (up to 10,000 records)	http://geothermal.smu.edu:9000/geoserver/aasg-heatflow/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=aasg-heatflow:HeatFlow&outputFormat=csv
Link to Static Zipped CSV File (entire dataset; ~ 16.6 MB)	http://geothermal.smu.edu/static/DatasetsZipped/core.template_heatflow_materialized.zip
WFS Sample 50 as CSV Link	http://geothermal.smu.edu:9000/geoserver/aasg-heatflow/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=aasg-heatflow:HeatFlow&outputFormat=csv&maxFeatures=50
XLM Schema Describe Feature Type Link	http://geothermal.smu.edu:9000/geoserver/aasg-heatflow/ows?service=WFS&version=1.0.0&request=DescribeFeatureType&typeName=aasg-heatflow:HeatFlow
‘About the Dataset.PDF’ or Link to Content Model Information	http://schemas.usgin.org/models/#heatflow
Number of Records 'hits' command	http://geothermal.smu.edu:9000/geoserver/aasg-heatflow/ows?service=WFS&version=1.1.0&request=GetFeature&typeName=aasg-heatflow:HeatFlow&resultType=hits

GEOSERVER LINKS FOR AGGREGATED RADIOGENIC HEAT GENERATION IN CONTENT MODEL FORMAT

File Name Used in GTDA for WMS: aasg-radiogenicheat

File Name Used in GTDA for WFS: aasg-radiogenicheat:RadiogenicHeatProduction

Dataset delivered in Content Model Format?: Yes

WMS Ver. 1.1.1 Link	http://geothermal.smu.edu:9000/geoserver/wms?service=wms&version=1.1.1&request=GetCapabilities&namespace=aasg-radiogenicheat
WMS Ver. 1.3 Link	http://geothermal.smu.edu:9000/geoserver/wms?service=wms&version=1.3&request=GetCapabilities&namespace=aasg-radiogenicheat
WFS XML Link	http://geothermal.smu.edu:9000/geoserver/aasg-radiogenicheat/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=aasg-radiogenicheat:RadiogenicHeatProduction
WFS Full File As CSV Link (up to 10,000 records)	http://geothermal.smu.edu:9000/geoserver/aasg-radiogenicheat/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=aasg-radiogenicheat:RadiogenicHeatProduction&outputFormat=csv
Link to Static Zipped CSV File (entire dataset; ~ <1 MB)	http://geothermal.smu.edu/static/DatasetsZipped/core.template_radiogenicheat_materialized.zip
WFS Sample 50 as CSV Link	http://geothermal.smu.edu:9000/geoserver/aasg-radiogenicheat/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=aasg-radiogenicheat:RadiogenicHeatProduction&outputFormat=csv&maxFeatures=50
XLM Schema Describe Feature Type Link	http://geothermal.smu.edu:9000/geoserver/aasg-radiogenicheat/ows?service=WFS&version=1.0.0&request=DescribeFeatureType&typeName=aasg-radiogenicheat:RadiogenicHeatProduction
‘About the Dataset.PDF’ or Link to Content Model Information	http://schemas.usgin.org/models/#radiogenicheatproduction
Number of Records 'hits' command	http://geothermal.smu.edu:9000/geoserver/aasg-radiogenicheat/ows?service=WFS&version=1.1.0&request=GetFeature&typeName=aasg-radiogenicheat:RadiogenicHeatProduction&resultType=hits

GEOSERVER LINKS FOR AGGREGATED THERMAL CONDUCTIVITY OBSERVATION IN CONTENT MODEL FORMAT

File Name Used in GTDA for WMS: aasg-thermalconductivity

File Name Used in GTDA for WFS: aasg-thermalconductivity:ThermalConductivity

Dataset delivered in Content Model Format?: Yes

WMS Ver. 1.1.1 Link	http://geothermal.smu.edu:9000/geoserver/wms?service=wms&version=1.1.1&request=GetCapabilities&namespace=aasg-thermalconductivity
WMS Ver. 1.3 Link	http://geothermal.smu.edu:9000/geoserver/wms?service=wms&version=1.3&request=GetCapabilities&namespace=aasg-thermalconductivity
WFS XML Link	http://geothermal.smu.edu:9000/geoserver/aasg-thermalconductivity/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=aasg-thermalconductivity:ThermalConductivity
WFS Full File As CSV Link (up to 10,000 records)	http://geothermal.smu.edu:9000/geoserver/aasg-thermalconductivity/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=aasg-thermalconductivity:ThermalConductivity&outputFormat=csv
Link to Static Zipped CSV File (entire dataset; ~ 5.1 MB)	http://geothermal.smu.edu/static/DatasetsZipped/core.template_thermalconductivity_materialized.zip
WFS Sample 50 as CSV Link	http://geothermal.smu.edu:9000/geoserver/aasg-thermalconductivity/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=aasg-thermalconductivity:ThermalConductivity&outputFormat=csv&maxFeatures=50
XLM Schema Describe Feature Type Link	http://geothermal.smu.edu:9000/geoserver/aasg-thermalconductivity/ows?service=WFS&version=1.0.0&request=DescribeFeatureType&typeName=aasg-thermalconductivity:ThermalConductivity
‘About the Dataset.PDF’ or Link to Content Model Information	http://schemas.usgin.org/models/#thermalconductivity
Number of Records 'hits' command	http://geothermal.smu.edu:9000/geoserver/aasg-thermalconductivity/ows?service=WFS&version=1.1.0&request=GetFeature&typeName=aasg-thermalconductivity:ThermalConductivity&resultType=hits

GEOSERVER LINKS FOR AGGREGATED WELL FLUID PRODUCTION OBSERVATION IN CONTENT MODEL FORMAT

File Name Used in GTDA for WMS: aasg-wellfluidproduction

File Name Used in GTDA for WFS: aasg-wellfluidproduction:FluidProduction

Dataset delivered in Content Model Format?: Yes

WMS Ver. 1.1.1 Link	http://geothermal.smu.edu:9000/geoserver/wms?service=wms&version=1.1.1&request=GetCapabilities&namespace=aasg-wellfluidproduction
WMS Ver. 1.3 Link	http://geothermal.smu.edu:9000/geoserver/wms?service=wms&version=1.3&request=GetCapabilities&namespace=aasg-wellfluidproduction
WFS XML Link	http://geothermal.smu.edu:9000/geoserver/aasg-wellfluidproduction/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=aasg-wellfluidproduction:FluidProduction
WFS Full File As CSV Link (up to 10,000 records)	http://geothermal.smu.edu:9000/geoserver/aasg-wellfluidproduction/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=aasg-wellfluidproduction:FluidProduction&outputFormat=csv
Link to Static Zipped CSV File (entire dataset; ~ 4.6 MB)	http://geothermal.smu.edu/static/DatasetsZipped/core.template_wellfluid_materialized.zip
WFS Sample 50 as CSV Link	http://geothermal.smu.edu:9000/geoserver/aasg-wellfluidproduction/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=aasg-wellfluidproduction:FluidProduction&outputFormat=csv&maxFeatures=50
XLM Schema Describe Feature Type Link	http://geothermal.smu.edu:9000/geoserver/aasg-wellfluidproduction/ows?service=WFS&version=1.0.0&request=DescribeFeatureType&typeName=aasg-wellfluidproduction:FluidProduction
‘About the Dataset.PDF’ or Link to Content Model Information	http://schemas.usgin.org/models/#fluidproduction
Number of Records 'hits' command	http://geothermal.smu.edu:9000/geoserver/aasg-wellfluidproduction/ows?service=WFS&version=1.1.0&request=GetFeature&typeName=aasg-wellfluidproduction:FluidProduction&resultType=hits

GEOSERVER LINKS FOR AGGREGATED WELL HEADER OBSERVATION IN CONTENT MODEL FORMAT

File Name Used in GTDA for WMS: aasg-wellheader

File Name Used in GTDA for WFS: aasg-wellheader:WellHeader

Dataset delivered in Content Model Format?: Yes

WMS Ver. 1.1.1 Link	http://geothermal.smu.edu:9000/geoserver/wms?service=wms&version=1.1.1&request=GetCapabilities&namespace=aasg-wellheader
WMS Ver. 1.3 Link	http://geothermal.smu.edu:9000/geoserver/wms?service=wms&version=1.3&request=GetCapabilities&namespace=aasg-wellheader
WFS XML Link	http://geothermal.smu.edu:9000/geoserver/aasg-wellheader/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=aasg-wellheader:WellHeader
WFS Full File As CSV Link (up to 10,000 records)	http://geothermal.smu.edu:9000/geoserver/aasg-wellheader/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=aasg-wellheader:WellHeader&outputFormat=csv
Link to Static Zipped CSV File (entire dataset; ~ 101 MB)	http://geothermal.smu.edu/static/DatasetsZipped/core.template_wellheader_materialized.zip
WFS Sample 50 as CSV Link	http://geothermal.smu.edu:9000/geoserver/aasg-wellheader/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=aasg-wellheader:WellHeader&outputFormat=csv&maxFeatures=50
XLM Schema Describe Feature Type Link	http://geothermal.smu.edu:9000/geoserver/aasg-wellheader/ows?service=WFS&version=1.0.0&request=DescribeFeatureType&typeName=aasg-wellheader:WellHeader
‘About the Dataset.PDF’ or Link to Content Model Information	http://schemas.usgin.org/models/#wellheader
Number of Records 'hits' command	http://geothermal.smu.edu:9000/geoserver/aasg-wellheader/ows?service=WFS&version=1.1.0&request=GetFeature&typeName=aasg-wellheader:WellHeader&resultType=hits

GEOSERVER LINKS FOR AGGREGATED WELL LOG OBSERVATION IN CONTENT MODEL FORMAT

File Name Used in GTDA for WMS: aasg-welllog

File Name Used in GTDA for WFS: aasg-welllog:WellLog

Dataset delivered in Content Model Format?: Yes

WMS Ver. 1.1.1 Link	http://geothermal.smu.edu:9000/geoserver/wms?service=wms&version=1.1.1&request=GetCapabilities&namespace=aasg-welllog
WMS Ver. 1.3 Link	http://geothermal.smu.edu:9000/geoserver/wms?service=wms&version=1.3&request=GetCapabilities&namespace=aasg-welllog
WFS XML Link	http://geothermal.smu.edu:9000/geoserver/aasg-welllog/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=aasg-welllog:WellLog
WFS Full File As CSV Link (up to 10,000 records)	http://geothermal.smu.edu:9000/geoserver/aasg-welllog/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=aasg-welllog:WellLog&outputFormat=csv
Link to Static Zipped CSV File (entire dataset; ~ 3.6 MB)	http://geothermal.smu.edu/static/DatasetsZipped/core.template_welllog_materialized.zip
WFS Sample 50 as CSV Link	http://geothermal.smu.edu:9000/geoserver/aasg-welllog/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=aasg-welllog:WellLog&outputFormat=csv&maxFeatures=50
XLM Schema Describe Feature Type Link	http://geothermal.smu.edu:9000/geoserver/aasg-welllog/ows?service=WFS&version=1.0.0&request=DescribeFeatureType&typeName=aasg-welllog:WellLog
'About the Dataset.PDF' or Link to Content Model Information	http://schemas.usgin.org/models/#welllog
Number of Records 'hits' command	http://geothermal.smu.edu:9000/geoserver/aasg-welllog/ows?service=WFS&version=1.1.0&request=GetFeature&typeName=aasg-welllog:WellLog&resultType=hits

GEOSERVER LINKS FOR AGGREGATED WELL TEST OBSERVATION IN CONTENT MODEL FORMAT

File Name Used in GTDA for WMS: aasg-welltests

File Name Used in GTDA for WFS: aasg-welltests:WellTestObservation

Dataset delivered in Content Model Format?: Yes

WMS Ver. 1.1.1 Link	http://geothermal.smu.edu:9000/geoserver/wms?service=wms&version=1.1.1&request=GetCapabilities&namespace=aasg-welltests
WMS Ver. 1.3 Link	http://geothermal.smu.edu:9000/geoserver/wms?service=wms&version=1.3&request=GetCapabilities&namespace=aasg-welltests
WFS XML Link	http://geothermal.smu.edu:9000/geoserver/aasg-welltests/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=aasg-welltests:WellTestObservation
WFS Full File As CSV Link (up to 10,000 records)	http://geothermal.smu.edu:9000/geoserver/aasg-welltests/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=aasg-welltests:WellTestObservation&outputFormat=csv
Link to Static Zipped CSV File (entire dataset; ~ 22.8 MB)	http://geothermal.smu.edu/static/DatasetsZipped/core.template_welltest_materialized.zip
WFS Sample 50 as CSV Link	http://geothermal.smu.edu:9000/geoserver/aasg-welltests/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=aasg-welltests:WellTestObservation&outputFormat=csv&maxFeatures=50
XLM Schema Describe Feature Type Link	http://geothermal.smu.edu:9000/geoserver/aasg-welltests/ows?service=WFS&version=1.0.0&request=DescribeFeatureType&typeName=aasg-welltests:WellTestObservation
'About the Dataset.PDF' or Link to Content Model Information	http://schemas.usgin.org/models/#WellTest
Number of Records 'hits' command	http://geothermal.smu.edu:9000/geoserver/aasg-welltests/ows?service=WFS&version=1.1.0&request=GetFeature&typeName=aasg-welltests:WellTestObservation&resultType=hits

GEOSERVER LINKS FOR SMU HEAT FLOW DATA FROM EQUILIBRIUM LOGS

File Name Used in GTDA for WMS: gt-da-contributions:SMUHeatFlow

File Name Used in GTDA for WFS: gt-da-contributions:SMUHeatFlow

Dataset delivered in Content Model Format?: No

WMS Ver. 1.1.1 Link	http://geothermal.smu.edu:9000/geoserver/wms?service=wms&version=1.1.1&request=GetCapabilities&namespace=gt-da-contributions:SMUHeatFlow
WMS Ver. 1.3 Link	http://geothermal.smu.edu:9000/geoserver/wms?service=wms&version=1.3&request=GetCapabilities&namespace=gt-da-contributions:SMUHeatFlow
WFS XML Link	http://geothermal.smu.edu:9000/geoserver/gt-da-contributions/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=gt-da-contributions:SMUHeatFlow
WFS Full File As CSV Link (up to 10,000 records)	http://geothermal.smu.edu:9000/geoserver/gt-da-contributions/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=gt-da-contributions:SMUHeatFlow&outputFormat=csv
Link to Static Zipped CSV File (entire dataset; ~ 5.9 MB)	http://geothermal.smu.edu/static/DatasetsZipped/staging.smu_hf_view_materialized.zip
WFS Sample 50 as CSV Link	http://geothermal.smu.edu:9000/geoserver/gt-da-contributions/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=gt-da-contributions:SMUHeatFlow&outputFormat=csv&maxFeatures=50
XLM Schema Describe Feature Type Link	http://geothermal.smu.edu:9000/geoserver/gt-da-contributions/ows?service=WFS&version=1.0.0&request=DescribeFeatureType&typeName=gt-da-contributions:SMUHeatFlow
'About the Dataset.PDF' or Link to Content Model Information	http://geothermal.smu.edu/static/DescribeColumns/SMUHeatFlow.pdf
Number of Records 'hits' command	http://geothermal.smu.edu:9000/geoserver/gt-da-contributions/ows?service=WFS&version=1.1.0&request=GetFeature&typeName=gt-da-contributions:SMUHeatFlow&resultType=hits

GEOSERVER LINKS FOR SMU HEAT FLOW DATA FROM BHT

File Name Used in GTDA for WMS: gt-da-contributions:SMUBHT

File Name Used in GTDA for WFS: gt-da-contributions:SMUBHT

Dataset delivered in Content Model Format?: No

WMS Ver. 1.1.1 Link	http://geothermal.smu.edu:9000/geoserver/wms?service=wms&version=1.1.1&request=GetCapabilities&namespace=gt-da-contributions:SMUBHT
WMS Ver. 1.3 Link	http://geothermal.smu.edu:9000/geoserver/wms?service=wms&version=1.3&request=GetCapabilities&namespace=gt-da-contributions:SMUBHT
WFS XML Link	http://geothermal.smu.edu:9000/geoserver/gt-da-contributions/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=gt-da-contributions:SMUBHT
WFS Full File As CSV Link (up to 10,000 records)	http://geothermal.smu.edu:9000/geoserver/gt-da-contributions/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=gt-da-contributions:SMUBHT&outputFormat=csv
Link to Static Zipped CSV File (entire dataset; ~ 2 MB)	http://geothermal.smu.edu/static/DatasetsZipped/staging.smu_bht_view_materialized.zip
WFS Sample 50 as CSV Link	http://geothermal.smu.edu:9000/geoserver/gt-da-contributions/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=gt-da-contributions:SMUBHT&outputFormat=csv&maxFeatures=50
XLM Schema Describe Feature Type Link	http://geothermal.smu.edu:9000/geoserver/gt-da-contributions/ows?service=WFS&version=1.0.0&request=DescribeFeatureType&typeName=gt-da-contributions:SMUBHT
'About the Dataset.PDF' or Link to Content Model Information	http://geothermal.smu.edu/static/DescribeColumns/SMUBHT.pdf
Number of Records 'hits' command	http://geothermal.smu.edu:9000/geoserver/gt-da-contributions/ows?service=WFS&version=1.1.0&request=GetFeature&typeName=gt-da-contributions:SMUBHT&resultType=hits

GEOSERVER LINKS FOR BEG MASTER WELLS DATABASE

File Name Used in GTDA for WMS: gt-da-contributions:BEGWells

File Name Used in GTDA for WFS: gt-da-contributions:BEGWells

Dataset delivered in Content Model Format?: No

WMS Ver. 1.1.1 Link	http://geothermal.smu.edu:9000/geoserver/wms?service=wms&version=1.1.1&request=GetCapabilities&namespace=gt-da-contributions:BEGWells
WMS Ver. 1.3 Link	http://geothermal.smu.edu:9000/geoserver/wms?service=wms&version=1.3&request=GetCapabilities&namespace=gt-da-contributions:BEGWells
WFS XML Link	http://geothermal.smu.edu:9000/geoserver/gt-da-contributions/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=gt-da-contributions:BEGWells
WFS Full File As CSV Link (up to 10,000 records)	http://geothermal.smu.edu:9000/geoserver/gt-da-contributions/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=gt-da-contributions:BEGWells&outputFormat=csv
Link to Static Zipped CSV File (entire dataset; ~ 4.5 MB)	http://geothermal.smu.edu/static/DatasetsZipped/staging.beg_well_view_materialized.zip
WFS Sample 50 as CSV Link	http://geothermal.smu.edu:9000/geoserver/gt-da-contributions/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=gt-da-contributions:BEGWells&outputFormat=csv&maxFeatures=50
XLM Schema Describe Feature Type Link	http://geothermal.smu.edu:9000/geoserver/gt-da-contributions/ows?service=WFS&version=1.0.0&request=DescribeFeatureType&typeName=gt-da-contributions:BEGWells
'About the Dataset.PDF' or Link to Content Model Information	http://geothermal.smu.edu/static/DescribeColumns/BEGMasterWells.pdf
Number of Records 'hits' command	http://geothermal.smu.edu:9000/geoserver/gt-da-contributions/ows?service=WFS&version=1.1.0&request=GetFeature&typeName=gt-da-contributions:BEGWells&resultType=hits

GEOSERVER LINKS FOR BEG GEOLOGIC RESERVOIR ANALYSIS

File Name Used in GTDA for WMS: gt-da-contributions:BEGReservoirs

File Name Used in GTDA for WFS: gt-da-contributions:BEGReservoirs

Dataset delivered in Content Model Format?: Yes

WMS Ver. 1.1.1 Link	http://geothermal.smu.edu:9000/geoserver/wms?service=wms&version=1.1.1&request=GetCapabilities&namespace=gt-da-contributions:BEGReservoirs
WMS Ver. 1.3 Link	http://geothermal.smu.edu:9000/geoserver/wms?service=wms&version=1.3&request=GetCapabilities&namespace=gt-da-contributions:BEGReservoirs
WFS XML Link	http://geothermal.smu.edu:9000/geoserver/gt-da-contributions/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=gt-da-contributions:BEGReservoirs
WFS Full File As CSV Link (up to 10,000 records)	http://geothermal.smu.edu:9000/geoserver/gt-da-contributions/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=gt-da-contributions:BEGReservoirs&outputFormat=csv
Link to Static Zipped CSV File (entire dataset; ~ <1 MB)	http://geothermal.smu.edu/static/DatasetsZipped/staging.beg_reservoir_view_materialized.zip
WFS Sample 50 as CSV Link	http://geothermal.smu.edu:9000/geoserver/gt-da-contributions/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=gt-da-contributions:BEGReservoirs&outputFormat=csv&maxFeatures=50
XLM Schema Describe Feature Type Link	http://geothermal.smu.edu:9000/geoserver/gt-da-contributions/ows?service=WFS&version=1.0.0&request=DescribeFeatureType&typeName=gt-da-contributions:BEGReservoirs
'About the Dataset.PDF' or Link to Content Model Information	http://geothermal.smu.edu/static/DescribeColumns/BEGReservoir.pdf
Number of Records 'hits' command	http://geothermal.smu.edu:9000/geoserver/gt-da-contributions/ows?service=WFS&version=1.1.0&request=GetFeature&typeName=gt-da-contributions:BEGReservoirs&resultType=hits

GEOSERVER LINKS FOR CORNELL NY AND PA HEAT FLOW DATABASE

File Name Used in GTDA for WMS: gt-da-contributions: CornellWells

File Name Used in GTDA for WFS: gt-da-contributions: CornellWells

Dataset delivered in Content Model Format?: No

WMS Ver. 1.1.1 Link	http://geothermal.smu.edu:9000/geoserver/wms?service=wms&version=1.1.1&request=GetCapabilities&namespace=gt-da-contributions: CornellWells
WMS Ver. 1.3 Link	http://geothermal.smu.edu:9000/geoserver/wms?service=wms&version=1.3&request=GetCapabilities&namespace=gt-da-contributions: CornellWells
WFS XML Link	http://geothermal.smu.edu:9000/geoserver/gt-da-contributions/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=gt-da-contributions: CornellWells
WFS Full File As CSV Link (up to 10,000 records)	http://geothermal.smu.edu:9000/geoserver/gt-da-contributions/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=gt-da-contributions: CornellWells&outputFormat=csv
Link to Static Zipped CSV File (entire dataset; ~ <1 MB)	http://geothermal.smu.edu/static/DatasetsZipped/staging.cornell_well_view_materialized.zip
WFS Sample 50 as CSV Link	http://geothermal.smu.edu:9000/geoserver/gt-da-contributions/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=gt-da-contributions: CornellWells&outputFormat=csv&maxFeatures=50
XLM Schema Describe Feature Type Link	http://geothermal.smu.edu:9000/geoserver/gt-da-contributions/ows?service=WFS&version=1.0.0&request=DescribeFeatureType&typeName=gt-da-contributions: CornellWells
'About the Dataset.PDF' or Link to Content Model Information	http://geothermal.smu.edu/static/DescribeColumns/CornellWells.pdf
Number of Records 'hits' command	http://geothermal.smu.edu:9000/geoserver/gt-da-contributions/ows?service=WFS&version=1.1.0&request=GetFeature&typeName=gt-da-contributions: CornellWells&resultType=hits

GEOSERVER LINKS FOR TEXAS TECH OFF SHORE WELLS AND BHT

File Name Used in GTDA for WMS: gt-da-contributions:TexasTechWells

File Name Used in GTDA for WFS: gt-da-contributions:TexasTechWells

Dataset delivered in Content Model Format?: No

WMS Ver. 1.1.1 Link	http://geothermal.smu.edu:9000/geoserver/wms?service=wms&version=1.1.1&request=GetCapabilities&namespace=gt-da-contributions:TexasTechWells
WMS Ver. 1.3 Link	http://geothermal.smu.edu:9000/geoserver/wms?service=wms&version=1.3&request=GetCapabilities&namespace=gt-da-contributions:TexasTechWells
WFS XML Link	http://geothermal.smu.edu:9000/geoserver/gt-da-contributions/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=gt-da-contributions:TexasTechWells
WFS Full File As CSV Link (up to 10,000 records)	http://geothermal.smu.edu:9000/geoserver/gt-da-contributions/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=gt-da-contributions:TexasTechWells&outputFormat=csv
Link to Static Zipped CSV File (entire dataset; ~ 1.1 MB)	http://geothermal.smu.edu/static/DatasetsZipped/staging.tt_well_view_materialized.zip
WFS Sample 50 as CSV Link	http://geothermal.smu.edu:9000/geoserver/gt-da-contributions/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=gt-da-contributions:TexasTechWells&outputFormat=csv&maxFeatures=50
XLM Schema Describe Feature Type Link	http://geothermal.smu.edu:9000/geoserver/gt-da-contributions/ows?service=WFS&version=1.0.0&request=DescribeFeatureType&typeName=gt-da-contributions:TexasTechWells
'About the Dataset.PDF' or Link to Content Model Information	http://geothermal.smu.edu/static/DescribeColumns/TexasTechWells.pdf
Number of Records 'hits' command	http://geothermal.smu.edu:9000/geoserver/gt-da-contributions/ows?service=WFS&version=1.1.0&request=GetFeature&typeName=gt-da-contributions:TexasTechWells&resultType=hits

GEOSERVER LINKS FOR UND RADIOGENIC HEAT GENERATION

File Name Used in GTDA for WMS: gt-da-contributions:UNDRadiogenicHeat

File Name Used in GTDA for WFS: gt-da-contributions:UNDRadiogenicHeat

Dataset delivered in Content Model Format?: Yes

WMS Ver. 1.1.1 Link	http://geothermal.smu.edu:9000/geoserver/wms?service=wms&version=1.1.1&request=GetCapabilities&namespace=gt-da-contributions:UNDRadiogenicHeat
WMS Ver. 1.3 Link	http://geothermal.smu.edu:9000/geoserver/wms?service=wms&version=1.3&request=GetCapabilities&namespace=gt-da-contributions:UNDRadiogenicHeat
WFS XML Link	http://geothermal.smu.edu:9000/geoserver/gt-da-contributions/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=gt-da-contributions:UNDRadiogenicHeat
WFS Full File As CSV Link (up to 10,000 records)	http://geothermal.smu.edu:9000/geoserver/gt-da-contributions/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=gt-da-contributions:UNDRadiogenicHeat&outputFormat=csv
Link to Static Zipped CSV File (entire dataset; ~ <1 MB)	http://geothermal.smu.edu/static/DatasetsZipped/staging.und_rad_view_materialized.zip
WFS Sample 50 as CSV Link	http://geothermal.smu.edu:9000/geoserver/gt-da-contributions/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=gt-da-contributions:UNDRadiogenicHeat&outputFormat=csv&maxFeatures=50
XLM Schema Describe Feature Type Link	http://geothermal.smu.edu:9000/geoserver/gt-da-contributions/ows?service=WFS&version=1.0.0&request=DescribeFeatureType&typeName=gt-da-contributions:UNDRadiogenicHeat
'About the Dataset.PDF' or Link to Content Model Information	http://geothermal.smu.edu/static/DescribeColumns/UNDRadiogenicHeat.pdf
Number of Records 'hits' command	http://geothermal.smu.edu:9000/geoserver/gt-da-contributions/ows?service=WFS&version=1.1.0&request=GetFeature&typeName=gt-da-contributions:UNDRadiogenicHeat&resultType=hits

GEOSERVER LINKS FOR UND THERMAL CONDUCTIVITY

File Name Used in GTDA for WMS: gt-da-contributions:UNDThermalConductivity

File Name Used in GTDA for WFS: gt-da-contributions:UNDThermalConductivity

Dataset delivered in Content Model Format?: Yes

WMS Ver. 1.1.1 Link	http://geothermal.smu.edu:9000/geoserver/wms?service=wms&version=1.1.1&request=GetCapabilities&namespace=gt-da-contributions:UNDThermalConductivity
WMS Ver. 1.3 Link	http://geothermal.smu.edu:9000/geoserver/wms?service=wms&version=1.3&request=GetCapabilities&namespace=gt-da-contributions:UNDThermalConductivity
WFS XML Link	http://geothermal.smu.edu:9000/geoserver/gt-da-contributions/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=gt-da-contributions:UNDThermalConductivity
WFS Full File As CSV Link (up to 10,000 records)	http://geothermal.smu.edu:9000/geoserver/gt-da-contributions/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=gt-da-contributions:UNDThermalConductivity&outputFormat=csv
Link to Static Zipped CSV File (entire dataset; ~ <1 MB)	http://geothermal.smu.edu/static/DatasetsZipped/staging.und_tc_view_materialized.zip
WFS Sample 50 as CSV Link	http://geothermal.smu.edu:9000/geoserver/gt-da-contributions/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=gt-da-contributions:UNDThermalConductivity&outputFormat=csv&maxFeatures=50
XLM Schema Describe Feature Type Link	http://geothermal.smu.edu:9000/geoserver/gt-da-contributions/ows?service=WFS&version=1.0.0&request=DescribeFeatureType&typeName=gt-da-contributions:UNDThermalConductivity
'About the Dataset.PDF' or Link to Content Model Information	http://geothermal.smu.edu/static/DescribeColumns/UNDThermalConductivity.pdf
Number of Records 'hits' command	http://geothermal.smu.edu:9000/geoserver/gt-da-contributions/ows?service=WFS&version=1.1.0&request=GetFeature&typeName=gt-da-contributions:UNDThermalConductivity&resultType=hits

GEOSERVER LINKS FOR UND TEMPERATURE AT DEPTH

File Name Used in GTDA for WMS: gt-da-contributions:UNDTempDepth

File Name Used in GTDA for WFS: gt-da-contributions:UNDTempDepth

Dataset delivered in Content Model Format?: No

WMS Ver. 1.1.1 Link	http://geothermal.smu.edu:9000/geoserver/wms?service=wms&version=1.1.1&request=GetCapabilities&namespace=gt-da-contributions:UNDTempDepth
WMS Ver. 1.3 Link	http://geothermal.smu.edu:9000/geoserver/wms?service=wms&version=1.3&request=GetCapabilities&namespace=gt-da-contributions:UNDTempDepth
WFS XML Link	http://geothermal.smu.edu:9000/geoserver/gt-da-contributions/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=gt-da-contributions:UNDTempDepth
WFS Full File As CSV Link (up to 10,000 records)	http://geothermal.smu.edu:9000/geoserver/gt-da-contributions/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=gt-da-contributions:UNDTempDepth&outputFormat=csv
Link to Static Zipped CSV File (entire dataset; ~ <1 MB)	http://geothermal.smu.edu/static/DatasetsZipped/staging.und_td_view_materialized.zip
WFS Sample 50 as CSV Link	http://geothermal.smu.edu:9000/geoserver/gt-da-contributions/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=gt-da-contributions:UNDTempDepth&outputFormat=csv&maxFeatures=50
XLM Schema Describe Feature Type Link	http://geothermal.smu.edu:9000/geoserver/gt-da-contributions/ows?service=WFS&version=1.0.0&request=DescribeFeatureType&typeName=gt-da-contributions:UNDTempDepth
'About the Dataset.PDF' or Link to Content Model Information	http://geothermal.smu.edu/static/DescribeColumns/UNDTempDepth.pdf
Number of Records 'hits' command	http://geothermal.smu.edu:9000/geoserver/gt-da-contributions/ows?service=WFS&version=1.1.0&request=GetFeature&typeName=gt-da-contributions:UNDTempDepth&resultType=hits

Appendix B: List of Publications and Presentations from Consortium Team Members over Course of Project

Project DE-EE0002852 supported research covered in numerous theses and conference presentations from 2010 through the end of 2013. The Masters of Science Theses and Doctoral Dissertations are listed in alphabetical order by author's last name. The meetings, papers, and conference proceedings are listed chronologically from 2010 through the end of calendar year 2013. Information regarding the materials listed below were made available to the Department of Energy's Office of Scientific and Technical Information (OSTI) in April of 2014. When protected by copyright, information regarding the source link was provided.

Table of Contents for Appendix B: List of Publications and Presentations from Consortium Team Members over Course of Project..... 1

Graduate Theses and/or Dissertations, December 2011 – May 2014	2
Meetings, Papers and Conference Proceedings, 2010 – 2014	3
GRC 2010 Annual Meeting, Sacramento, CA, October 24-27, 2010.....	3
Kansas Geophysics in the 21st Century, Wichita, KS, October 27-29, 2010.	5
GSA 2010 Annual Meeting, Denver, CO, October 31-November3, 2010.	5
AAPG Hedberg 2011, Napa, CA, March 14-18, 2011.	5
SMU Geothermal Energy Utilization Associated with Oil & Gas Development, Dallas, TX, June 13-15, 2011.....	5
GSA 2011 Annual Meeting, Minneapolis, MN, October 9-12, 2011.	6
GRC 2011 Annual Meeting, San Diego, CA, October 24-27, 2011.	6
NSF Workshop: Geothermal Potential of Sedimentary Basins, Salt Lake City, UT, November 7-9, 2011.....	7
2011 Fall Meeting AGU, San Francisco, CA, December 5-9, 2011.....	7
Dallas E&P Forum Raising the Dead: New Techniques and Technologies for Bringing Old Oil Fields Back to Life, Sponsored by SIPES and SPE, Dallas, TX, December 13, 2011.	8
37th Stanford Geothermal Workshop, Stanford, CA, February 11-13, 2012.	8
2012 AAPG Annual Convention and Exhibition, Long Beach, CA, April 22-25, 2012.	8
Arizona Geological NGDS Webinar, Webinar, April 26, 2012.....	9
GRC 2012 Annual Meeting, Reno, NV, September 30- October 3, 2012.	9
Presentation to Southwest Texas Electric Cooperative, El Dorado, TX, October 2, 2012.	10
62nd Annual Gulf Coast Association of Geological Societies and the Gulf Coast Section of SEPM Convention Solving for E3-Energy, Environment, Economy, Austin, TX, October 21-24, 2012.....	10
GSA 2012 Annual Meeting, Charlotte, NC, November 4-7, 2012.	10
2012 Fall Meeting AGU, San Francisco, CA, December 3-7, 2012.....	11

38th Stanford Geothermal Workshop, Stanford, CA, February 11-13, 2013.	11
SMU 2013 Geothermal Energy and Waste Heat to Power: Utilizing Oil and Gas Plays, Dallas, TX, March 12-14, 2013.....	12
CanGEA 2013 Geothermal Conference, Calgary, Alberta, Canada, March 20-22, 2013.....	12
GRC 2013 Annual Meeting, Las Vegas, NV, September 30-October 3, 2013.....	13
63rd Annual Gulf Coast Association of Geological Societies and the Gulf Coast Section of SEPM Convention, New Orleans, LA, October 6-8, 2013.....	13
2013 GSA Penrose Conference: Predicting and Detecting Natural and Induced Flow Paths for Geothermal Fluids in Deep Sedimentary Basins, Park City, UT, October 19-23, 2013.....	14
2013 PennWell Renewable Energy World Conference and Expo, Orlando, FL, November 12-14, 2013.....	14
2014 Harvesting Clean Energy, Helena, MT, February 4-6, 2014.	14
Other Professional Publications	14

GRADUATE THESES AND/OR DISSERTATIONS, DECEMBER 2011 – MAY 2014

(Listed alphabetically by author's last name):

Geothermal Resource Assessment: A Case Study of Spatial Variability and Uncertainty Analysis for the State of New York and Pennsylvania, Aguirre, G.A..

Master's Thesis: Cornell University, May 2014.

URL not yet available

Identifying Potential Geothermal Resources from Co-Produced Fluids Using Existing Data from Drilling Logs: Williston Basin, North Dakota, Crowell, Anna M..

Master's Thesis: University of North Dakota, December 2011.

<http://und.edu/libraries.cfm>

Thermal Modeling of Step-Out Targets at the Soda Lake Geothermal Field Churchill County, Nevada, Dingwall, Ryan.

Master's Thesis: Southern Methodist University, May 18, 2013. SMU Thesis 1540257

<http://search.proquest.com/docview/1416422186>

An Energy Return on Investment for a Geothermal Power Plant on the Texas Gulf Coast, Kampa, Kyle.

Master's Thesis: University of Texas at Austin, May 2013. BEG Thesis 21768 | KAMPA-THESIS-2013.pdf

<http://hdl.handle.net/2152/21768>

Terrestrial Heat Flow and Assessment of Enhanced Geothermal Systems Resources in Minnesota, Klenner, Robert C.L..

Master's Thesis: University of North Dakota, August 2012. UND Thesis 1520090

<http://search.proquest.com/docview/1113388223/DA193906573E4ACFPQ/2>

Geophysical Investigation and Assessment of the Rye Patch Known Geothermal Resource Area Rye Patch, Nevada, McDonald, Mark.

Doctoral Dissertation: University of North Dakota, November 2013.

<http://und.edu/libraries.cfm>

Correcting Heat Flow Data in the United States to Account for Climate Change, Njoku, Godswill.

Master's Thesis: University of North Dakota, December 2013.

<http://und.edu/libraries.cfm>

Evaluating Opportunities for Enhanced Geothermal system EGS-Based District Heating in New York and Pennsylvania, Reber, Tim.

Master's Thesis: Cornell University, May 26, 2013. Cornell Thesis 34090

<http://hdl.handle.net/1813/34090>

A Detailed Approach to Low-Grade Geothermal Resources in the Appalachian Basin of New York and Pennsylvania: Heterogeneities within the Geologic Model and their effect on Geothermal Resource Assessment, Shope, Elaina Nicole.

Master's Thesis: Cornell University, August 20, 2012. Cornell Thesis 31206

<http://hdl.handle.net/1813/31206>

Development, Analysis, and Application of a Well by Well Method for Estimating Surface Heat Flow for Regional Geothermal Resource Assessment, Stutz, George Raymond.

Master's Thesis: Cornell University, August 20, 2012. Cornell Thesis 31037 | grs76.pdf

<http://hdl.handle.net/1813/31037>

An Assessment of Value for Deep Sedimentary Geothermal Resources in Texas, Uddenberg, Matthew Emmanuel.

Master's Thesis: University of Texas at Austin, May 2012. BEG Thesis ETD-UT-2012-05-5641

<http://hdl.handle.net/2152/ETD-UT-2012-05-5641>

Thermal and Structural Assessment of the Pirouette Mtn Geothermal Prospect Southern Dixie Valley, Nevada, Williams, Mitchell.

Master's Thesis: Southern Methodist University, December 15, 2012. SMU Thesis 1532148

<http://search.proquest.com/docview/1284867047/3E4169B5447C4C3DPQ/9>

MEETINGS, PAPERS AND CONFERENCE PROCEEDINGS, 2010 – 2014

(Listed chronologically, then by first author's last name):

GRC1 2010 Annual Meeting, Sacramento, CA, October 24-27, 2010.

Evaluation of EGS Resources in the Eastern United States: Illinois Michigan, Indiana, Western Ohio, and Western Kentucky, Batir, Joseph & Blackwell, Dave.

GRC Transactions, Vol 34, pp. 295-299

<http://pubs.geothermal-library.org/lib/grc/1028661.pdf>

¹ Geothermal Resources Council

A Protocol for Estimating the Global Potential for EGS, Beardsmore, Graeme & Rybach, Ladislaus & Blackwell, David & Baron, Charles.

GRC Transactions, Vol 34, pp. 301-312

<http://pubs.geothermal-library.org/lib/grc/1028662.pdf>

Comparison and Discussion of the 6 KM Temperature Maps of the Western US Prepared by the SMU Geothermal Lab and the USGS, Blackwell, David & Stepp, Patrick & Richards, Maria.

GRC Transactions, Vol 34, pp. 515-519

<http://pubs.geothermal-library.org/lib/grc/1028696.pdf>

New Geothermal Resource Map of the Northeastern US and Technique for Mapping Temperature at Depth, Blackwell, David & Richards, Maria.

GRC Transactions, Vol 34, pp. 313-318

<http://pubs.geothermal-library.org/lib/grc/1028663.pdf>

Modeling and Simulation of the Interface between Geothermal Power Plant Based on Organic Rankine Cycle and the Electric Grid, Dahal , Samir & Salehfar, Hossein & Gosnold, Will & Mann, M..

GRC Transactions, Vol 34, pp. 1011-1015

<http://pubs.geothermal-library.org/lib/grc/1028779.pdf>

Geothermal Map of the Northeastern United States and the West Virginia Thermal Anomaly, Frone, Zachary & Blackwell, David.

GRC Transactions, Vol 34, pp. 339-343

<http://pubs.geothermal-library.org/lib/grc/1028668.pdf>

EGS Potential in the Northern Midcontinent of North America, Gosnold, Will & LeFever, Richard & Mann, Michael & Klenner, Robert & Salehfar, Hossein.

GRC Transactions, Vol 34, pp. 355-358

<http://pubs.geothermal-library.org/lib/grc/1028671.pdf>

Geothermal Power From Coproduced Fluids in the Williston Basin, Gosnold, Will & LeFever, Richard & Klenner, Robert & Mann, Michael.

GRC Transactions, Vol 34, pp. 557-560

<http://pubs.geothermal-library.org/lib/grc/1028702.pdf>

Development of Exploration Methods of Engineered Geothermal Systems Through Integrated Geothermal Systems Through Integrated Geophysical, Geophysical, Geologic and Geochemical Interpretation, Iovenitti , Joe L. & Tibuleac, Ileana & Hopkins, Deborah & Cladouhos, Trenton & Karlin, Robert & Wannermaker, Phillip & Kennedy, Mack & Blackwell, David & Clynne, Matthew.

GRC Transactions, Vol 34, pp. 363-368

<http://pubs.geothermal-library.org/lib/grc/1028673.pdf>

Geothermal Opportunities of Southeastern United States, Sica, Nicole & Blackwell, Dave & Frone, Zach & Park, Junghyun.

GRC Transactions, Vol 34, pp. 131-135

<http://pubs.geothermal-library.org/lib/grc/1028631.pdf>

Co-Generation Opportunities for Lower Grade Geothermal Resources in the Northeast - A Case Study of the Cornell Site in Ithaca, NY, Tester, Jeff & Joyce, W & Brown, L & Bland, B & Clark, A. & Jordan, T. & Andro-nicos, C. & Allmendinger, R. & Beyers, S. & Blackwell, D. & Richards, M. & Frone, Z. & Anderson, Brian.

GRC Transactions, Vol 34, pp. 475-483

<http://pubs.geothermal-library.org/lib/grc/1028690.pdf>

Kansas Geophysics in the 21st Century, Wichita, KS, October 27-29, 2010.

North Central US Heat Flow May be Undetermined, Gosnold, Will.

no url available

GSA² 2010 Annual Meeting, Denver, CO, October 31-November3, 2010.

Identifying Potential Geothermal Resources from Co-Produced Fluids Using Existing Data from Drilling Logs: Williston Basin, North Dakota, Crowell, Anna & Gosnold, William & Yarbrough, Lance D..

GSA Abstracts with Programs, Vol. 42 | No. 5, p. 489Paper No. 205-9

https://gsa.confex.com/gsa/2010AM/finalprogram/abstract_181947.htm

Enhanced Geothermal Systems in the Midcontinent, Gosnold, Will & Klenner, Rob.

GSA Abstracts with Programs, Vol. 42 | No. 5, p. 379Paper No. 153-4

https://gsa.confex.com/gsa/2010AM/finalprogram/abstract_181797.htm

Re-evaluating Terrestrial Heat Flow in Minnesota, Klenner, Rob & Gosnold, Will.

GSA Abstracts with Programs, Vol. 42 | No. 5, p. 379Paper No. 153-7

(abstract only) https://gsa.confex.com/gsa/2010AM/finalprogram/abstract_181549.htm

AAPG Hedberg 2011, Napa, CA, March 14-18, 2011.

High Performance Utilization of Lower Grade EGS Resources – Connecting Thermodynamic and Economic Criteria, Tester, Jeff & Lukawski, Maciej & Beckers, Koenraad & Fox, Don B. & Anderson, Brian.

(abstract only) http://www.searchanddiscovery.com/pdfz/abstracts/pdf/2011/hedberg-california/abstracts/ndx_tester.pdf.html

SMU Geothermal Energy Utilization Associated with Oil & Gas Development, Dallas, TX, June 13-15, 2011.

Using GIS to Estimate Geothermal Energy Reserves in Sedimentary Basins, Crowell, Anna.

<http://smu.edu/geothermal/> (navigate to 'conference')

² Geological Society of America

The Transformation of Tight Shale Gas Reservoirs to Geothermal Energy Production, Cutright, Bruce.
<http://smu.edu/geothermal/> (navigate to 'conference')

Google.org US Resource Project Update, Frone, Zachary.
<http://smu.edu/geothermal/> (navigate to 'conference')

North Dakota Geothermal Projects, Gosnold, Will.
<http://smu.edu/geothermal/> (navigate to 'conference')

Solving the Geothermal Data Puzzle, Kay, Martin.
<http://smu.edu/geothermal/> (navigate to 'conference')

A Borehole Geothermal Database for the US Exclusive Economic Zone of the Gulf of Mexico, Nagihara, Seiichi.
<http://smu.edu/geothermal/> (navigate to 'conference')

GSA 2011 Annual Meeting, Minneapolis, MN, October 9-12, 2011.

National Geothermal Data System, Anderson, Arlene F. & Cuyler, David & Snyder, Walter S. & Allison, M. Lee & Blackwell, David D. & Williams, Colin F..
GSA Abstracts with Programs, Vol. 43 | No. 5, p. 41 Paper No. 9-9
https://gsa.confex.com/gsa/2011AM/finalprogram/abstract_195159.htm

Determining the Volumes, Porosities, and Available Energy of Selected Reservoirs Utilizing GIS Methods: Williston Basin, North Dakota, Crowell, Anna M. & Gosnold, William.
GSA Abstracts with Programs, Vol. 43 | No. 5, p. 40 Paper No. 9-7
<https://gsa.confex.com/gsa/2011AM/webprogram/Paper196811.html>

Minnesota Geothermal Potential, Gosnold, Will D. & Klenner, R.C.L. & Hauck, Steven.
GSA Abstracts with Programs, Vol. 43 | No. 5, p. 41 Paper No. 9-12
https://gsa.confex.com/gsa/2011AM/finalprogram/abstract_197252.htm

Application of the Harrison Correction to Nebraska BHT Data, Ochsner, Aaron.
GSA Abstracts with Programs, Vol. 43 | No. 5, p. 104 Paper No. 34-6
<https://gsa.confex.com/gsa/2011AM/webprogram/Paper195224.html>

GRC 2011 Annual Meeting, San Diego, CA, October 24-27, 2011.

Data Integration into the National Geothermal Data System (NGDS), Blackwell, David & Moerchen, Fabian & Cutright, Bruce & Gosnold, Will & Duncan, Ian J. & Kay, Martin & Nagihara, Seiichi & Robinson, Curt & Tester, Jefferson.
GRC Transactions, Vol 35, pp. 1539-1543
<http://pubs.geothermal-library.org/lib/grc/1029451.pdf>

Temperature-At-Depth Maps For the Conterminous US and Geothermal Resource Estimate, Blackwell, David & Richards, Maria & Frone, Zachary & Ruza, Andrés & Dingwall, Ryan & Williams, Mitchell.

GRC Transactions, Vol 35, pp. 1545-1550
<http://pubs.geothermal-library.org/lib/grc/1029452.pdf>

GIS Analysis for the Volumes and Available Energy of Selected Reservoirs: Williston Basin, North Dakota, Crowell, Anna & Klenner, Rob C.L. & Gosnold, Will.
GRC Transactions, Vol 35, pp. 1557-1561
<http://pubs.geothermal-library.org/lib/grc/1029454.pdf>

Correcting Bottom Hole Temperatures: A Look at the Permian Basin (Texas), Williston Basin (North Dakota), Anadarko and Arkoma Basins (Oklahoma), Crowell, Anna & Gosnold, Will.
GRC Transactions, Vol 35, pp. 735-738
<http://pubs.geothermal-library.org/lib/grc/1029323.pdf>

Geothermal Map of Colorado, Wyoming, Montana, and Nebraska, Dingwall, Ryan & Blackwell, David.
GRC Transactions, Vol 35, pp. 739-742
<http://pubs.geothermal-library.org/lib/grc/1029324.pdf>

Implications of Post-glacial Warming for Northern Hemisphere Heat Flow, Gosnold, Will & Majorowicz, Jacek & Klenner, Rob & Hauck, Steve.
GRC Transactions, Vol 35, pp. 795-799
<http://pubs.geothermal-library.org/lib/grc/1029332.pdf>

2011 Geothermal Map of Arizona and New Mexico, (Poster Session), Williams, Mitchell & Blackwell, David.
GRC Transactions, Vol 35, pp. 1065-1068
<http://pubs.geothermal-library.org/lib/grc/1029377.pdf>

NSF³ Workshop: Geothermal Potential of Sedimentary Basins, Salt Lake City, UT, November 7-9, 2011.

Introduction and Overview of Geothermal Resources, Blackwell, David & Richards, Maria.
no url available

2011 Fall Meeting AGU⁴, San Francisco, CA, December 5-9, 2011.

National Geothermal Data System, Anderson, Arlene F. & Cuyler, David & Snyder, Walter S. & Allison, M. Lee & Blackwell, David D. & Williams, Colin F..
(abstract only)
<http://abstractsearch.agu.org/meetings/2011/FM/sections/IN/sessions/IN31D/abstracts/IN31D-03.html>

Re-Evaluating Geothermal Potential with GIS Methods and New Data: Williston Basin, North Dakota, Crowell, Anna M. & Gosnold, William D..

³ National Science Foundation

⁴ American Geophysical Union

(abstract only)

<http://abstractsearch.agu.org/meetings/2011/FM/sections/IN/sessions/IN33C/abstracts/IN33C-1480.html>

Minnesota Heat Flow and Geothermal Potential, Gosnold, William D. & Crowell, James & Bubach, Bailey & Wahl, Preston & Crowell, Anna M. & McDonald, Mark R..

(abstract only)

<http://abstractsearch.agu.org/meetings/2011/FM/sections/IN/sessions/IN33C/abstracts/IN33C-1482.html>

Gravity Survey of the Rye Patch KGRA, Rye Patch, Nevada, McDonald, Mark R. & Gosnold, William D..

(abstract only)

<http://abstractsearch.agu.org/meetings/2011/FM/sections/GP/sessions/GP41A/abstracts/GP41A-0982.html>

Dallas E&P Forum Raising the Dead: New Techniques and Technologies for Bringing Old Oil Fields Back to Life, Sponsored by SIPES and SPE, Dallas, TX, December 13, 2011.

There's no Stopping Geothermal Energy: Join the Movement to Re-Energize your Field, Richards, Maria & Blackwell, David.

no url available

37th Stanford Geothermal Workshop, Stanford, CA, February 11-13, 2012.

Geothermal Resources Assessment: A Detailed Approach to Low-Grade Resources in the States of New York and Pennsylvania, Shope, Elaina Nicole & Reber, Tim & Stutz, George & Aguirre, G.A. & Jordan, T. & Tester, Jeff.

PROCEEDINGS, Thirty-Seventh Workshop on Geothermal Reservoir Engineering, Vol 37, pp. 885-893 GRC 8008360

<http://www.geothermal-energy.org/pdf/IGAstandard/SGW/2012/Shope.pdf>

A Well by Well Method for Estimating Surface Heat Flow for Regional Geothermal Resource Assessment, Stutz, George & Williams, Mitchell & Frone, Zachary & Reber, Tim & Richards, Maria & Blackwell, David & Jordan, T. & Tester, Jeff.

PROCEEDINGS, Thirty-Seventh Workshop on Geothermal Reservoir Engineering, Vol 37, pp. 906-921 GRC 8008366

<http://www.geothermal-energy.org/pdf/IGAstandard/SGW/2012/Stutz.pdf>

2012 AAPG Annual Convention and Exhibition, Long Beach, CA, April 22-25, 2012.

The Transformation of Hydrofracked Reservoirs to Thermal Energy Production, Cutright, Bruce L..

http://www.searchanddiscovery.com/documents/2012/80223cutright/ndx_cutright.pdf

Geothermal Energy from Mature Gas Reservoirs; an Analysis of the Frio and Vicksburg Formations, South Texas, Hidalgo County. *Adapted from oral presentation., Smith, Colgan B. & Uddenberg, Matt E. & Setchko, Andrew K. & Cutright, Bruce L. & Stater, Adam & Bello, Shadiya & Kampa, Kyle.

http://www.searchanddiscovery.com/documents/2012/80235smith/ndx_smith.pdf

Arizona Geological NGDS Webinar, Webinar, April 26, 2012.

Where to Find Relevant Geothermal Data from Oil & Gas Sources, Kay, Martin.

<http://www.youtube.com/watch?v=CRPfSyjX6cs> AND/OR

<http://www.stategeothermaldata.org/system/files/FinalPresentation.pdf>

GRC 2012 Annual Meeting, Reno, NV, September 30- October 3, 2012.

Evaluation of the Geothermal Potential in the Rio Grande Rift: Truth or Consequences, New Mexico, Barse, Kirtipal & McDonald, Mark & Crowell, Anna M..

GRC Transactions, Vol 36, pp. 1315-1320

<http://pubs.geothermal-library.org/lib/grc/1030401.pdf>

Why Basin and Range Systems Are Hard to Find: The Moral of the Story is They Get Smaller With Depth!, Blackwell, David & Waibel, AlbertF. & Richards, Maria.

GRC Transactions, Vol 36, pp. 1321-1326

<http://pubs.geothermal-library.org/lib/grc/1030402.pdf>

Correcting Bottom-Hole Temperatures in the Denver Basin: Colorado and Nebraska,

Crowell, Anna & Ochsner, Aaron & Gosnold, Will.

GRC Transactions, Vol 36, pp. 201-206

<http://pubs.geothermal-library.org/lib/grc/1030229.pdf>

Evaluation of Geothermal Potential of Lightning Dock KGRA, New Mexico, Dahal, Samir & McDonald, Mark & Bubach, Bailey & Crowell, Anna M..

GRC Transactions, Vol 36, pp. 637-640

<http://pubs.geothermal-library.org/lib/grc/1030294.pdf>

Oak Ridge Institute for Science and Education (ORISE) U.S. DOE EERE National Geothermal Student Competition Finalist Poster Presentation by SMU Geothermal Laboratory: Heat Flow and Fluid Flow Modeling of the Western Snake River Plain, Frone, Zachary & Batir, Joseph & Dingwall, Ryan & Williams, Mitchell & Blackwell, David. (Announcement only) <http://orise.ornl.gov/science-education/capabilities/science-education-events/2012-national-geothermal-student-competition.aspx>

Thermostrostratigraphy of the Williston Basin, Gosnold, William & McDonald, MarkR. & Klenner, Robert & Merriam, Daniel.

GRC Transactions, Vol 36, pp. 663-670

<http://pubs.geothermal-library.org/lib/grc/1030298.pdf>

Implications of Post-Glacial Warming for Northern Alberta Heat Flow-Correcting for the Underestimate of the Geothermal Potential, Majorowicz, Jacek & Gosnold, Will & Gray,

Allan & Safanda, Jan & Klenner, Rob & Unsworth, Martyn.

GRC Transactions, Vol 36, pp. 693-698

<http://pubs.geothermal-library.org/lib/grc/1030303.pdf>

Proposed Reliability Code for Heat Flow Sites, Richards, Maria & Blackwell, David & Williams, Mitchell & Frone, Zachary & Dingwall, Ryan & Batir, Joseph & Chickering, Cathy.
GRC Transactions, Vol 36, pp. 211-217
<http://pubs.geothermal-library.org/lib/grc/1030231.pdf>

The Regional Thermal Regime in Dixie Valley, Nevada, USA, Thakur, Mahesh & Blackwell, David & Erkan, Kamil.
GRC Transactions, Vol 36, pp. 59-67
<http://pubs.geothermal-library.org/lib/grc/1030211.pdf>

Geothermal Exploration at Newberry Volcano, Central Oregon, Waibel, Albert & Frone, Zachary & Jaffe, Todd.
GRC Transactions, Vol 36, pp. 803-810
<http://pubs.geothermal-library.org/lib/grc/1030321.pdf>

Early Geothermal Exploration of Southern Dixie Valley: A Case Study, Williams, Mitchell & Blackwell, David.
GRC Transactions, Vol 36, pp. 819-824
<http://pubs.geothermal-library.org/lib/grc/1030323.pdf>

Presentation to Southwest Texas Electric Cooperative, El Dorado, TX, October 2, 2012.

The Impact of Transformational Technologies in Power Generation from Geothermal Energy, Cutright, Bruce L..
no url available

62nd Annual Gulf Coast Association of Geological Societies and the Gulf Coast Section of SEPM Convention Solving for E3-Energy, Environment, Economy, Austin, TX, October 21-24, 2012.

Gulf of Mexico Borehole Geothermal Data Integration into the National Geothermal Data Systems, Part 1: Texas Continental Shelf, Nagihara, Seiichi & Christie, Cory & Ogiyamien, Nosakhare & Oladokun, Motunrayo & Ajiboye, Olabisi & Ojeda, Sergio.
Transactions, Gulf Coast Association of Geological Societies, Vol. 62 (2012), pp. 589-591
(membership required for download)
<http://archives.datapages.com/data/gcags/data/062/062001/pdfs/589.pdf>

Developing Geothermal Energy in Texas: Mapping Temperatures and Resources, Richards, Maria C. & Blackwell, David D..
Transactions, Gulf Coast Association of Geological Societies, Vol. 62 (2012), pp. 351-363
(membership required for download)
<http://archives.datapages.com/data/gcags/data/062/062001/pdfs/351.pdf>

GSA 2012 Annual Meeting, Charlotte, NC, November 4-7, 2012.

An Assessment of Texas' Geothermal Resources and Economic Potential, Zafar, S.D. & Cutright, B.L..
GSA Abstracts with Programs, Vol. 44 | No. 7, p. 602 Paper No. 258-6
https://gsa.confex.com/gsa/2012AM/finalprogram/abstract_209399.htm

2012 Fall Meeting AGU, San Francisco, CA, December 3-7, 2012.

Thermomechanical Characteristics of the YNP/SRP Region, Blackwell, David D. & Kelley, Shari & Erkan, Kamil.

(abstract only) <http://fallmeeting.agu.org/2012/eposters/eposter/v13b-2837/>

Available Thermal Energy in the Denver Basin Dakota Group: Colorado and Nebraska, Crowell, Anna M. & Gosnold, William D..

<http://fallmeeting.agu.org/2012/files/2012/11/AGU-Poster-20121.pdf>

Using a Divided Bar Apparatus to Measure Thermal Conductivity of Samples of Odd Sizes and Shapes, Crowell, James & Gosnold, William D..

<http://fallmeeting.agu.org/2012/files/2012/11/Josh-AGU-Poster-2012-v21.pdf>

Migration of a Caldera Eruptive Center, Newberry Volcano, Oregon (Poster), Frone, Zachary & Waibel, Al & Blackwell, David D..

(abstract only) <http://fallmeeting.agu.org/2012/eposters/eposter/v31b-2785/>

Synthesis of Bottom Hole Temperatures and Heat Flow Data, Gosnold, William D. & Crowell, Anna M..

<http://fallmeeting.agu.org/2012/files/2012/11/H13G-1446-Gosnold-and-Crowell.pdf>

38th Stanford Geothermal Workshop, Stanford, CA, February 11-13, 2013.

Geothermal Resource Assessment: A Case Study of Spatial Variability and Uncertainty Analysis for the State of New York and Pennsylvania, Aguirre, Gloria Andrea & Stedinger, Jerry R. & Tester, Jefferson W..

PROCEEDINGS, Thirty-Eighth Workshop on Geothermal Reservoir Engineering, Vol 38, pp. 1419-1430 GRC 8018974

<http://www.geothermal-energy.org/pdf/IGAstandard/SGW/2013/Aguirre.pdf>

National Geothermal Data System (NGDS) Geothermal Data Domain: Assessment of Geothermal Community Data Needs, Anderson, Arlene & Blackwell, David & Chickering, Cathy & Boyd, Toni & Horne, Roland & MacKenzie, Matthew & Moore, Joseph & Nickull, Duane & Richard, Stephen & Shevenell, Lisa A..

PROCEEDINGS, Thirty-Eighth Workshop on Geothermal Reservoir Engineering, Vol 38, pp. 25-38 GRC 8018984

<http://www.geothermal-energy.org/pdf/IGAstandard/SGW/2013/Anderson.pdf>

Introducing GEOPHIRES v1.0: Software Package for Estimating Levelized Cost of Electricity and/or Heat from Enhanced Geothermal Systems, Beckers, Koenraad F. & Lukawski, Maciej Z. & Reber, Timothy J. & Anderson, Brian J. & Moore, Michal C. & Tester, Jefferson W..

PROCEEDINGS, Thirty-Eighth Workshop on Geothermal Reservoir Engineering, Vol 38, pp. 701-708 GRC 8018993

<http://www.geothermal-energy.org/pdf/IGAstandard/SGW/2013/Beckers.pdf>

A Proposed Hybrid Geothermal-Natural Gas-Biomass Energy System for Cornell University: Technical and Economic Assessment of Retrofitting a Low-Temperature Geothermal District Heating System and Heat Cascading Solutions, Lukawski, Maciej Z. & Vilaetis, Konstantinos & Gkogka, Lizeta & Beckers, Koenraad F. & Anderson, Brian J. & Tester, Jefferson W..

*PROCEEDINGS, Thirty-Eighth Workshop on Geothermal Reservoir Engineering, Vol 38, pp. 1393-1402*GRC 8019078

<http://www.geothermal-energy.org/pdf/IGAstandard/SGW/2013/Lukawski.pdf>

Three Heating Seasons Monitoring of Usage of Low Enthalpy Geothermal Resources: Exergetic Performance Analysis of an EAHE Assisted Agricultural Building, Ozgener, Onder & Ozgener, Leyla & Tester, Jefferson W..

*PROCEEDINGS, Thirty-Eighth Workshop on Geothermal Reservoir Engineering, Vol 38, pp. 1445-1456*GRC 8019103

<http://www.geothermal-energy.org/pdf/IGAstandard/SGW/2013/Ozgener.pdf>

SMU 2013 Geothermal Energy and Waste Heat to Power: Utilizing Oil and Gas Plays, Dallas, TX, March 12-14, 2013.

Assessment of Low Grade Geothermal Resources: A case study for NY and PA, Andrea, Aguirre & Camp, Erin.

<http://smu.edu/geothermal/> (navigate to 'conference')

Recoverable Thermal Energy for Geothermal Power Production in the Denver Basin, Crowell, Anna M. & Gosnold, William D..

<http://smu.edu/geothermal/> (navigate to 'conference')

Finding Geothermal Resources for Project Development, Cutright, Bruce L..

<http://smu.edu/geothermal/> (navigate to 'conference')

Status of the North Dakota Geothermal Oil Field Geothermal Projects, Gosnold, William D. & Barse, Kirtipal A..

<http://smu.edu/geothermal/> (navigate to 'conference')

The Impact of Well Drilling Costs on the Economics of Geothermal Energy Utilization, Lukawski, Maciej Z..

no url available

CanGEA⁵ 2013 Geothermal Conference, Calgary, Alberta, Canada, March 20-22, 2013.

Developing Geothermal Energy Resources, Mapping the Data, Richards, Maria.

no url available

⁵ Canadian Geothermal Energy Association

GRC 2013 Annual Meeting, Las Vegas, NV, September 30-October 3, 2013.

Sustaining the National Geothermal Data System: Considerations for a System Wide Approach and Node Maintenance, Allison, M. Lee & Chickering, Cathy & Patten, Kim & Anderson, Arlene & Richard, Stephen M..

GRC Transactions, Vol 37

<https://www.geothermal-library.org/index.php?mode=pubs&action=view&record=1030618>

National Geothermal Data System (NGDS) Geothermal Data: Community Requirements and Information Engineering, Anderson, Arlene & Blackwell, David & Chickering, Cathy & Boyd, Toni & Horne, Roland & MacKenzie, Matthew & Moore, Joseph & Nickull, Duane & Richard, Stephen & Shevenell, Lisa A..

GRC Transactions, Vol 37

<https://www.geothermal-library.org/index.php?mode=pubs&action=view&record=1030619>

Updated Surface Heat Flow Map of Alaska, Batir, Joseph F. & Blackwell, David D. & Richards, Maria C..

GRC Transactions, Vol 37

<https://www.geothermal-library.org/index.php?mode=pubs&action=view&record=1030682>

The Future of Geothermal Energy: The Shale Gas Analogy -- Significant Electrical EGS Resource Areas in the US, Blackwell, David & Frone, Zachary & Richards, Maria C..

GRC Transactions, Vol 37

<https://www.geothermal-library.org/index.php?mode=pubs&action=view&record=1030559>

GIS-Based Geothermal Resource Assessment of the Denver Basin: Colorado and Nebraska, Crowell, Anna & Gosnold, Will.

GRC Transactions, Vol 37

<https://www.geothermal-library.org/index.php?mode=pubs&action=view&record=1030684>

Detecting Spatial Trends in Thermal Conductivity in the Williston Basin, Crowell, James & Gosnold, William D..

GRC Transactions, Vol 37

<https://www.geothermal-library.org/index.php?mode=pubs&action=view&record=1030612>

Co-Produced Geothermal Resources and EGS in the Williston Basin, Gosnold, William & Barse, Kirtipal & Bubach, Bailey & Crowell, Anna & Crowell, James & Jabbari, Hadi & Sarnoski, Anthony & Wang, Dongmei.

GRC Transactions, Vol 37

<https://www.geothermal-library.org/index.php?mode=pubs&action=view&record=1030649>

63rd Annual Gulf Coast Association of Geological Societies and the Gulf Coast Section of SEPM Convention, New Orleans, LA, October 6-8, 2013.

Gulf of Mexico Borehole Geothermal Data Integration into the National Geothermal Data Systems, Part 2: North-central Continental Shelf, Nagihara, Seiichi & et al.,

Transactions, Gulf Coast Association of Geological Societies, Vol. 63 (2013), pp. 561-562

(membership required for download)

<http://archives.datapages.com/data/gcags/data/063/063001/pdfs/561.pdf>

2013 GSA Penrose Conference: Predicting and Detecting Natural and Induced Flow Paths for Geothermal Fluids in Deep Sedimentary Basins, Park City, UT, October 19-23, 2013.

Introduction and overview of Geothermal Resources, Blackwell, David.

no url available

Thermal Resources Beneath the Eastern United States Continental Margin, Blackwell, David.

no url available

Thermal Hydraulic Modeling of Discretely Fractured Geothermal Reservoirs, Fox, Don.

no url available

2013 PennWell Renewable Energy World Conference and Expo, Orlando, FL, November 12-14, 2013.

Developing Geothermal Energy in O/G Plays, using new NGDS Tools, Richards, Maria.

Speaker36422_Session6326_1.pdf

(download only available to conference delegates)

http://s36.a2zinc.net/clients/pennwell/rewna2013/Public/Cust_DownloadHandout.aspx?Task=Speaker36422_Session6326_1.pdf

2014 Harvesting Clean Energy, Helena, MT, February 4-6, 2014.

Waste heat and Geothermal Energy in O/G Plays, new available NGDS Tools, Richards, Maria.

no url available

OTHER PROFESSIONAL PUBLICATIONS

Texas' Geothermal Resource Base: A Raster-Integration Method for Estimating In-Place Geothermal-Energy Resources Using ARCGIS, Zafar, Syed Daniel & Cutright, Bruce L..

Geothermics, Vol. 50, pp. 148-154, April 1, 2014.

<http://dx.doi.org/10.1016/j.geothermics.2013.09.003>

GEOHERMAL DATA AGGREGATION

End User Manual

Version: V1.1

Table of Contents

1	Preface	3
1.1	Purpose and Audience	3
1.2	Document Roadmap	3
2	System Context.....	4
3	Description of Individual User Screens.....	5
3.1	Accessing Data.....	5
3.2	Login Screen	5
3.3	Map Search Page.....	7
	3.3.1Filtering.....	8
	3.3.2Accessing Data.....	9
3.4	Publications Catalog	13
3.5	Histograms	14
3.6	User Management.....	15
3.7	Submit Datasets	17
	3.7.1Publish dialog (activated when the publish button is clicked)	21
3.8	Submit Related Resources	21
	3.8.1Bibliography Upload	21
	3.8.2Publications Upload Page	23
	3.8.3Related Resources Submission	25
	3.8.3.1 Related Resources	27
	3.8.3.2 Field Notes.....	27
	3.8.3.3 TD Curves	28

4	Terms.....	29
4.1	Acronyms, and Abbreviations.....	29
4.2	Glossary of Geological Terms	29

List of Figures

Figure 1 Context diagram	4
Figure 2: Login Screen (http://geothermal.smu.edu).....	5
Figure 3:GTDA Login screen for visitors (right-hand side of http://geothermal.smu.edu/).....	6
Figure 4: Login panel for users with SMU Node credentials (right-hand side of http://geothermal.smu.edu/).....	7
Figure 5: Map Search Screen.	8
Figure 6: Screenshot of the system showing ballon info of one individual well. ...	10
Figure 7: Screenshot of the Well Details Page.	11
Figure 8: Screenshot of the data details view.	13
Figure 9: Screenshot of the publications catalog view.	14
Figure 10: Screenshot of the Histogram View.	15
Figure 11: Screenshot of the User Management Page.	16
Figure 12: Screenshot of the Upload Page for SMU Heatflow Data.	18
Figure 13: Screenshot of the job status tab.	19
Figure 14: Screenshot of the data review panel.....	20
Figure 15: Screenshot of the Upload Mechanism.....	22
Figure 16: Screenshot of the Job Status Page for Uploading the Bibliography.....	23
Figure 17: Publications Upload Page.	24
Figure 18: Job Status Page for the publications Upload Process.....	25
Figure 19: Upload Page for TD curves.	26
Figure 20: Job Status Page for uploading TD Curves.....	27

List of Tables

Table 1: Defined roles and the privileges associated with the roles.	17
Table 2: Abbreviations	29
Table 3: Terms	31

1 Preface

1.1 Purpose and Audience

This document is intended for a technical audience who need to understand how to operate the GTDA system. Targeted audience includes:

- **End User**
- System Administrator
- Developers

The purpose of this document is to describe to end users, from an operations point of view, the software implemented by Siemens to support the SMU Geothermal Laboratories efforts for the project

GEOHERMAL DATA AGGREGATION: SUBMISSION OF INFORMATION INTO THE NATIONAL GEOTHERMAL DATA SYSTEM (U.S. Department of Energy award number DE-EE0002852.)

1.2 Document Roadmap

This document outlines the architecture of NGDS and is structured in the following way:

- Section 4 provides a list with important terms
- Section 2 outlines the system context and high-level feature overview
- Section outlines the components of the system

2 System Context

As seen in Figure 1, the System facilitates the uploading, integration and submission of geothermal data.

The System also checks the integrity of the data. The System handles the submission of the meta-data records to the GTDA system, and transmits instances of the geothermal data, when requested. Also, the System provides a mechanism to handle maintenance issues relating to the operation of the system.

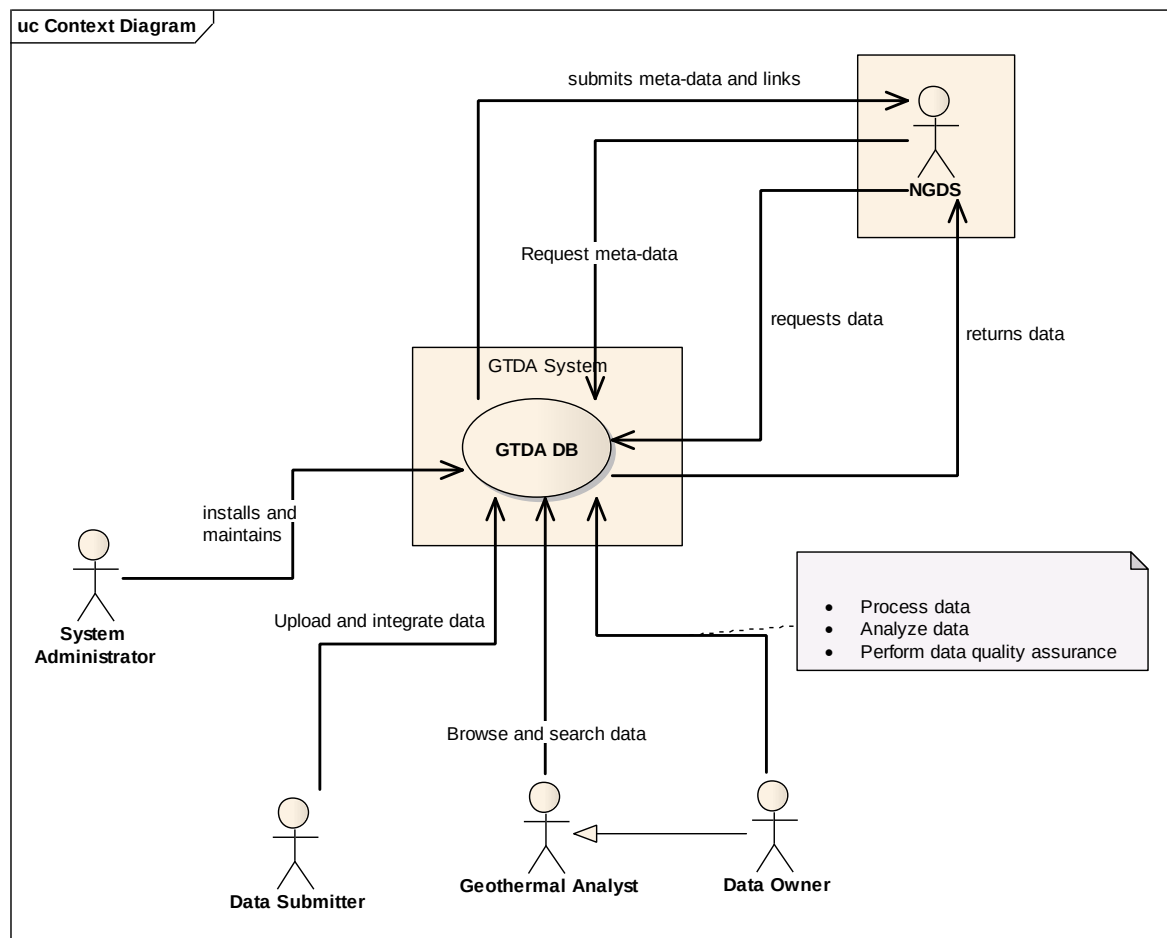


Figure 1 Context diagram

The System supports the following feature groups:

- Data input
- Data browsing
- Quality assurance
- Data submission
- System administration

3 Description of Individual User Screens

In the following sections we outline the functionality of the individual screens.

3.1 Accessing Data

The SMU Node provides two ways to access the data it contains: interactively via the map, or non-interactively by downloading the data in bulk. The login screen of the application provides a description of how to access the data via download. This avenue of access does not require the user to login. Interactive access, on the other hand does require the user to login, even if it is as a visitor. Logging in is described in the next section

3.2 Login Screen

The Login Screen allows users to request access to the system. In the following you find a Screenshot of the Login Screen.



Figure 2: Login Screen (<http://geothermal.smu.edu>).

Users wishing only to view the information and not to add or change it can log in anonymously. This can be done by opening the Visitor Panel and clicking on the “Login as Visitor” button.

Remark: it might be necessary to click on the “Login” button several times if at first click the system seems unresponsive.

The following screenshot shows the login screen with the guest login panel open:

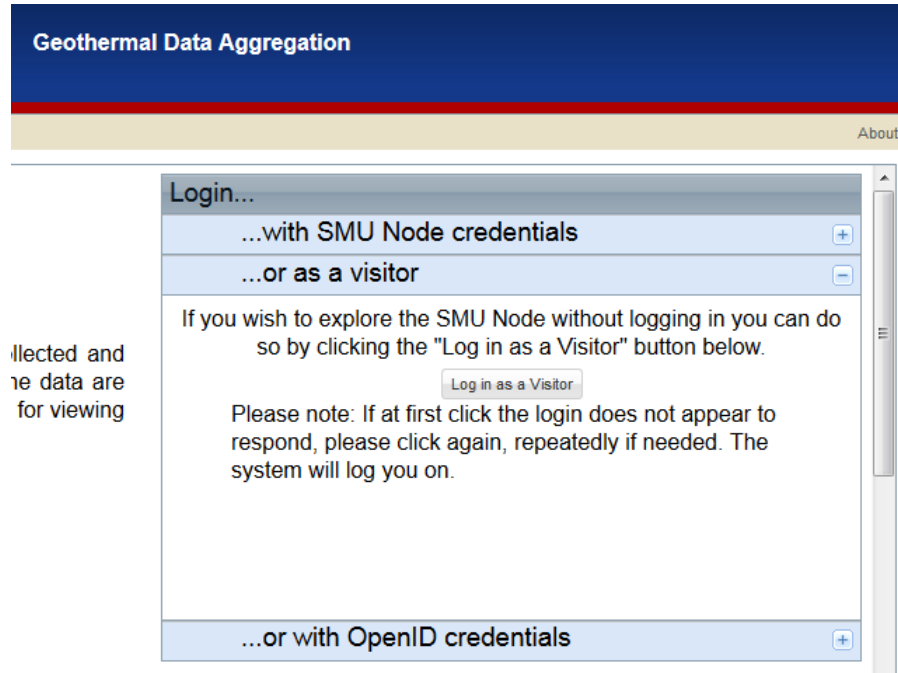


Figure 3:GTDA Login screen for visitors (right-hand side of <http://geothermal.smu.edu/>)

Users with contribution rights need to have SMU node credentials. They can use the 'SMU Node Login' Panel, shown below.

The screenshot shows a web interface with a dark blue header containing the text "I Data Aggregation". Below the header is a light yellow bar with the word "About" on the right. The main content area is a white box with a grey border. At the top of this box is a grey bar with the text "Login...". Below this is a light blue bar with the text "...with SMU Node credentials" and a minus sign icon. The main area of the box contains a "Username:" label followed by a text input field, a "Password:" label followed by a text input field, and a "Login" button. Below the input fields is a note: "Please note: If at first click the login does not appear to respond, please click again, repeatedly if needed. The system will log you on." At the bottom of the box are two more light blue bars: "...or as a visitor" with a plus sign icon, and "...or with OpenID credentials" with a plus sign icon.

Figure 4: Login panel for users with SMU Node credentials (right-hand side of <http://geothermal.smu.edu/>)

Users requiring login credentials and privileges to upload and publish data can request these from an administrator of the system by contacting ...

3.3 Map Search Page

Upon login, the user is presented with the map search screen:

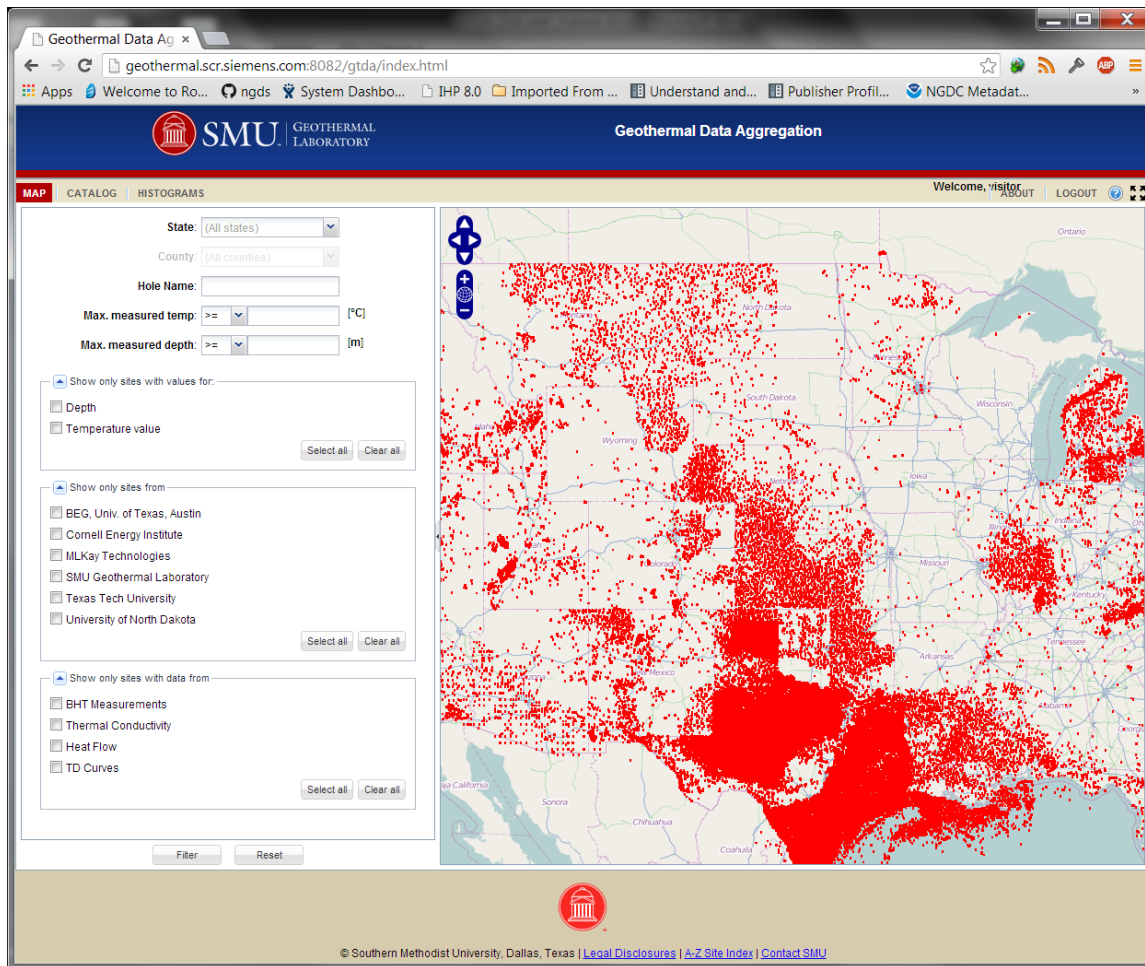


Figure 5: Map Search Screen.

The map shows the locations of wells about which the system has information. As can be seen in several areas (especially Texas), the map is densely populated with wells. The control buttons on the upper left corner of the map allow for panning and zooming into specific map regions.

Remark: Due to the large amount of data we recommend giving the system time to zoom into the region of interest.

3.3.1 Filtering

Using the filter panel on the left hand side the user can filter the found results. Enter and/or selecting search criteria and clicking the “Filter” button will cause the system to redraw the map. To clear the search results click the “Reset” button. Note that filtering does not automatically take place when the user has made a change to the search criteria. Rather, the user must click the “Filter” button for any changes in search criteria to take effect. Similarly, once the search criteria are reset the “Filter” button must be clicked to make the map reflect that no points are being filtered out.

The user can restrict the number of found results to **specific states and counties**. These widgets provide a drop-down selector. Multiple states and counties may be selected. Note that the county selector is disabled unless at least one state is selected. Furthermore, only the counties of the selected states are listed in the county selector.

It is also possible to select a specific well by providing its **hole name**. You need not provide the exact name of a hole. For the search text you enter, the system will place points on the map for which the search text occurs within any of the fields “hole name”, “API”, or “other location name”.

It is also possible to select wells that have a **Measured Maximum Temperature** that is at least or at most a certain value. Simply select either “>=” or “<=” from the drop down menu, enter the value to compare against, and click the “Filter” button. The same applies to the **Depth** of the Well as a filter criterion.

Since the system does not contain full data for every well (it depends on the uploaded data) one can also filter out wells for which no **depth** or **temperature value** was provided.

The filter panel also allows for filtering for wells for which the system has data from a specific data contributor. The node contains data from the following data providers:

- BEG, University of Texas, Austin
- Cornell Energy Institute
- MLKay Technologies
- SMU Geothermal Laboratory
- Texas Tech University
- University of North Dakota

Finally it is also possible to select those wells for which specific information types are available. The system allows for filtering for the following data types:

- BHT Measurements
- Thermal Conductivity
- Heat Flow
- TD Curves

3.3.2 Accessing Data

As described before, each red dot on the map represents a well about which the system provides information. After zooming into a specific region and filtering the wells of interest the user usually wants to access data about chosen wells. Data of a single well, as well as a list of resources related to that well can be obtained by clicking the point on the map. Alternately, a table of data for a given region can be obtained by selecting a region on the map. These two methods are described below.

When the user clicks on a point on the map, the system generates a small pop-up balloon at that point. The following Screenshot shows the balloon info of an individual well.

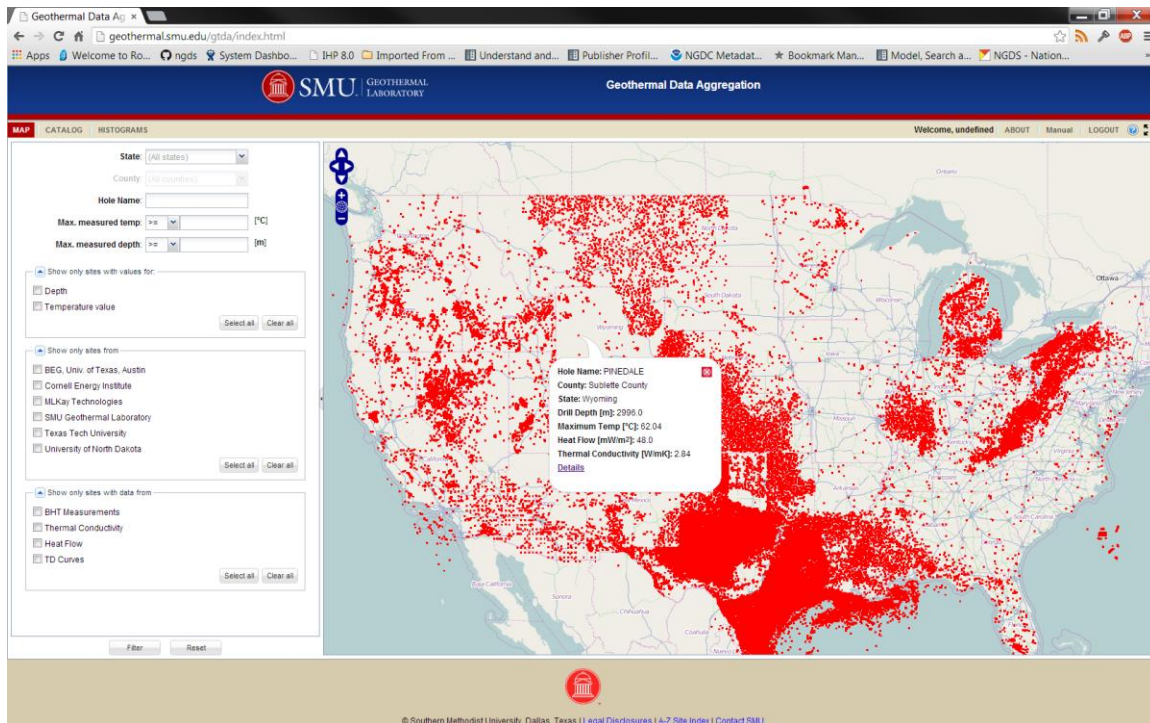


Figure 6: Screenshot of the system showing balloon info of one individual well.

The balloon provides summary information about the well at that location. The balloon also includes a link. By clicking the link the well's details page is opened in a new tab or window. The following Screenshot shows the Details Page for one well.

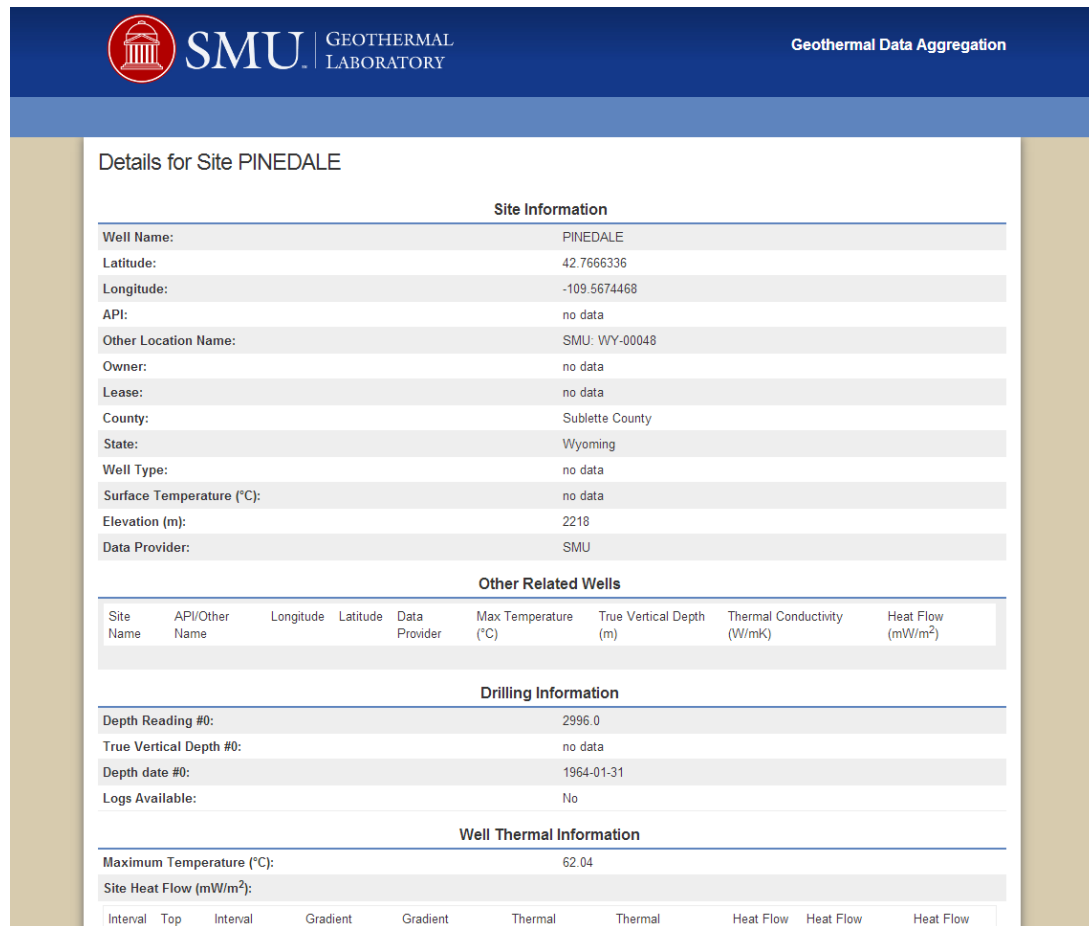


Figure 7: Screenshot of the Well Details Page.

Note that the link can be bookmarked and/or shared and is accessible directly without logging into the system.

The well details page also includes a list of “Other Related Wells.” This list includes wells that have the same geographic coordinates as the listed well, or coordinates that are very close. Since wells may overlap on the map, use this list to see what other wells might be at or near that location. The well you seek might be in this list. If it is not, then use the bounding-box method—described next—to get the wells of a specific area.

For sites with related temperature/depth curves (typically contributed by SMU and UND), these will be listed in the appropriate section along with a chart plotting the curves. By mousing over the curves you can see the values at various points on the curves. You can also toggle the visibility of curves by clicking their names in the chart's key. On the upper right is an icon from which a menu is available that allows you to print the chart or download it as a GIF, JPEG, or PNG image, or as a PDF.

In cases where there are multiple curves and/or curves with many points (total points plotted numbering over 1000) there is a likelihood that the script will be slow to draw, and react slowly to interaction, or become unresponsive. Your browser may warn you that a script on the page has become unresponsive and give you the option to allow it to continue or terminate the script. Testing has revealed that although it is time consuming to wait, the script rarely fails to eventually complete the task of you opt to let it continue. The Highcharts library may, however, have difficulty creating the chart for download.

Selecting an area using a bounding box: It is also possible to access the data about a certain area on the map in a tabular form. **Note:** Use this feature with caution! When choosing a large area on the map or an area with many wells it takes very long to create the results. We recommend to first zoom into a specific region of interest, and then mark a small area. Marking areas is done by pressing the Shift button and selecting the area of interest using the mouse (i.e. the user needs to press the shift button and the primary mouse button at the same time and then drag to draw a bounding box around a specific region on the map). Upon releasing the mouse button a window pops up that contains summary data available for the wells within the selected region. The data can be browsed in the window or downloaded as a CSV file. In order to download the data as a CSV file the user must first provide a file name and then press the 'Export to CSV' button. A new window pops up that allows for selecting which data one wants to download. When pressing the 'OK' button a set of CSV files is generated and compressed into a zip file. Depending on the size of the selected area this may take a long time. Once the zipped package is generated a download window pops up and the file is automatically downloaded into the download folder of the user's PC. The following Screenshot provides an overview:

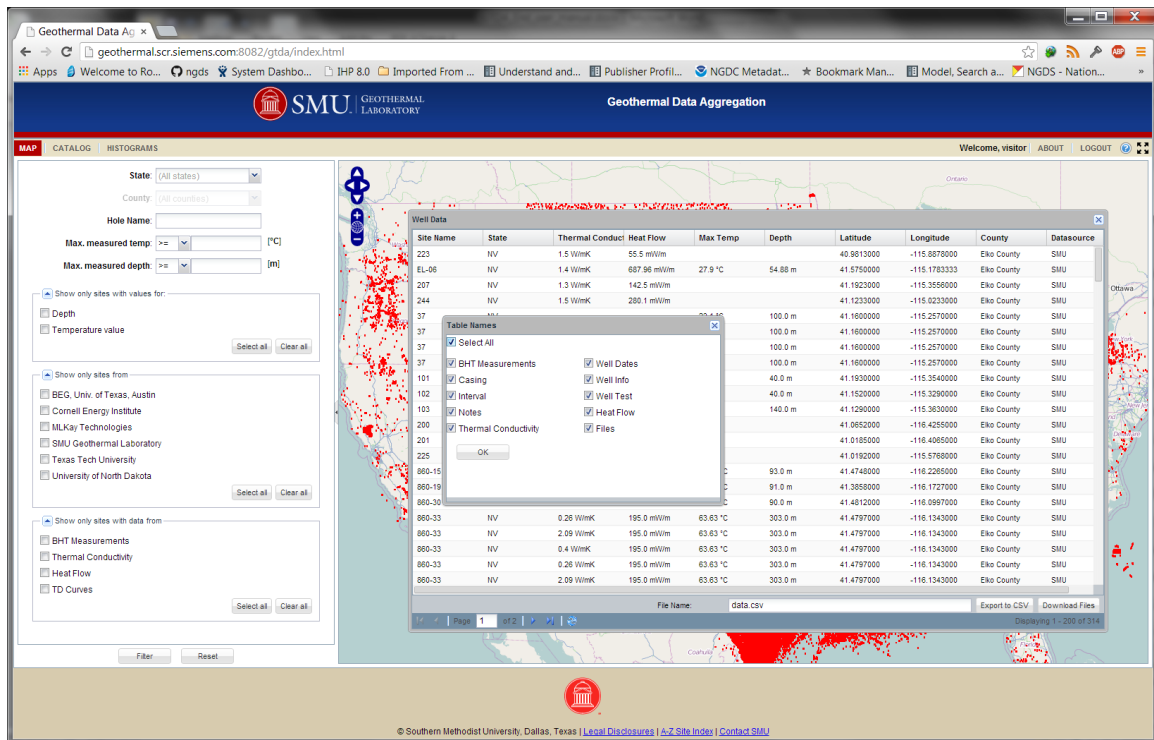


Figure 8: Screenshot of the data details view.

Remark: The popup windows must be closed using the ‘cross’ at the upper right corner of the popup window.

3.4 Publications Catalog

The publications Catalog view allows users to browse through the uploaded publications. In order to do so the user simply enters a search term in the search field and then clicks either the “Full Text” button or “Bibliography” button. The “Full Text” button retrieves documents in which the search terms are found within the document’s full text (including the title and author list and description.) The “Bibliography” button retrieves documents for which the search terms are found only within the title, author list, and description. In general, the search tries to be smart about the search terms by searching not only the exact terms, but subsets of them as well.

Within the search results, if a hyperlink for details of the bibliographic entry is available the user can click on the title of a found entry. This will open a new browser tab that is redirected to that hyperlink.

The following Screenshot shows the ‘Publications Catalog’ View:

SMU | GEOTHERMAL LABORATORY

Geothermal Data Aggregation

MAP CATALOG HISTOGRAMS ADMIN ▾ SUBMIT DATA SETS SUBMIT RELATED RESOURCES ▾ Welcome, smu ABOUT Manual LOGOUT

Seismicity Full Text Bibliography

Result	Author	Publication Date
Microseismic Monitoring of the World's Largest Potential HDR Reservoir microseismic monitoring, hot dry rock, European HDR project	R. Baria, S. Michelet, J. Baumgaertner, B. Dyer A...	2004
Workshops to rate and assign air and water issues for hydrothermal ener 15 GEOTHERMAL ENERGY; HYDROTHERMAL SYSTEMS; AIR POLLUTION CONTROL; LAND PO	Williams, J.M.; Wewerka, E.M.	1980
Environmental Development Plan (EDP). Enhanced gas recovery, FY 19' 03 NATURAL GAS; 29 ENERGY PLANNING, POLICY AND ECONOMY; COAL DEPOSITS; DEGAS		1978

Number of records found: 3

© Southern Methodist University, Dallas, Texas | [Legal Disclosures](#) | [A-Z Site Index](#) | [Contact SMU](#)

Figure 9: Screenshot of the publications catalog view.

For more sophisticated searches over the body of documents, you can use the Solr search interface, available at <http://geothermal.smu.edu/solr/#/collection1/query>. For details about how to use this querying interface, please see the Solr query documentation at <http://wiki.apache.org/solr/SolrQuerySyntax>.

3.5 Histograms

The histogram view allows for a rapid overview of the value distribution for specific data fields. The selector in the upper left corner enables the user to select the category of data to be reviewed in the histogram. For example the user may choose to review data of the category 'borehole temperature measurements' (abbreviated as bht-measurements). Within the chosen category the user next chooses the Field to be reviewed. For example the user may choose to review the field containing the corrected borehole temperature measurements called 'bht_corrected'.

Upon clicking on the "View Histogram" button the distribution of values is shown on the screen. The two fields 'Min Range' and 'Max Range' are automatically filled with the corresponding value from the chosen data field. The range values can be modified and upon pressing on the 'Apply Range' button the chosen ranges are applied. By pressing the 'Full Range' button the view resets to the full range of the data.

Finally, the vertical scrollbar on the right hand side allows for choosing the number of bins the histogram is constructed of.

The following Screenshot shows a histogram for corrected borehole temperatures:



Figure 10: Screenshot of the Histogram View.

Clicking on a bar once will show, in a grid below the histogram, the list of wells that make up the data represented by that bar. You can change the column upon which the list is sorted by clicking on the column header. You can also download the full list as CSV using the “Download as CSV” button to the upper right of the grid.

Double-clicking on a row in the grid will navigate you to the map screen with the selected well highlighted on the map.

3.6 User Management

This page is only available for users with administrator privileges. It consists of two sub-pages: Sub-page number one is the user management page and sub-page number two is the Account Request Page. The first page enables the administrator to manage the roles of the users in the system, reset their password and to deactivate and delete them.

The following Screenshot shows the user management page:

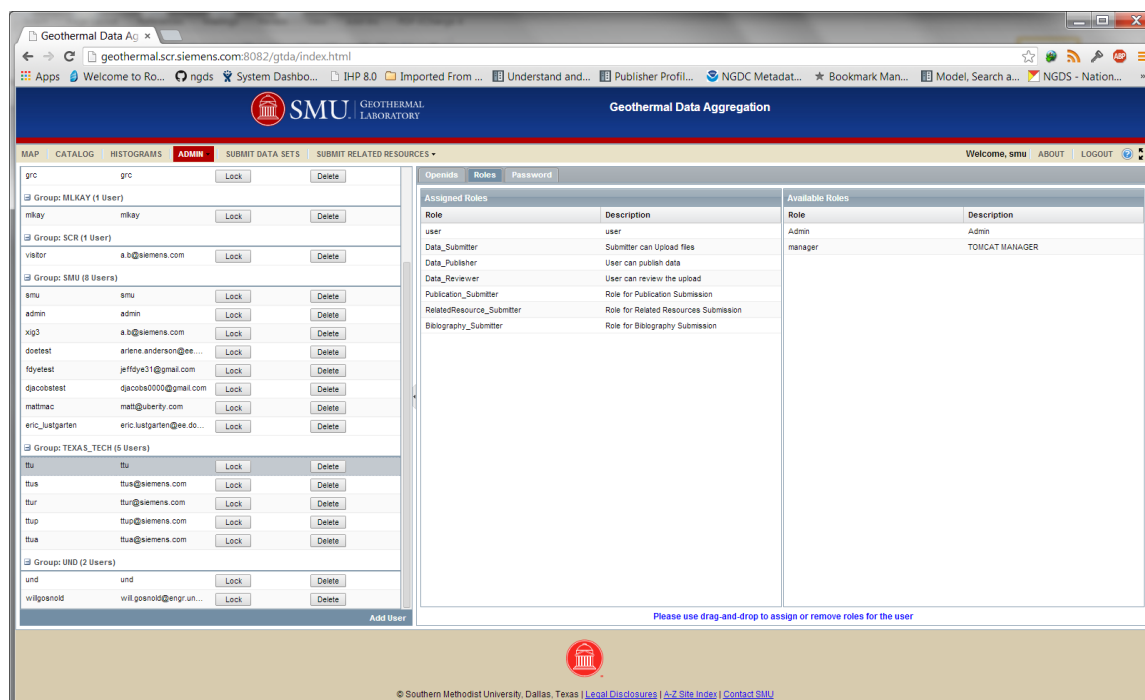


Figure 11: Screenshot of the User Management Page.

The screenshot shows the details about the roles of the user “ttu”. The table on the right half of the screen shows the roles that are assigned to the user and those once that are available but not assigned. By dragging them from one side to the other the administrator can change the role assignment.

The roles and the privileges they confer are listed in the table below

Role	Privilege	Screen
Admin	User account provisioning	
user	(not applicable)	
manager	(not applicable)	
Data_Submitter	Submission of data sets for review prior to publication	“File Upload” and “Job Status” tabs under “SUBMIT DATA SETS”
Data_Publisher	Publication of data sets that have been	“Publish” button on the “Review” tab under “SUBMIT DATA SETS”
Data_Reviewer	Review data sets for correctness prior to publication	“Review” tab under “SUBMIT DATA SETS”
Bibliography_Submitter	Submit and publish bibliographic data for publications	All tabs in “Bibliography Submission” under “SUBMIT RELATED RESOURCES”

RelatedResource_Submitter	Submit field notes, maps, temperature/depth curves, etc. as resources related to wells	All tabs in “Related Resource Submission” under “SUBMIT RELATED RESOURCES”
Publication_Submitter	Submit documents corresponding to imported bibliographic data	All tabs in “Publication Submission” under “SUBMIT RELATED RESOURCES”

Table 1: Defined roles and the privileges associated with the roles.

The additional tab “OpenId” allows to associate certain openId’s with the user. The tab “Password” allows the administrator to reset the user’s password.

3.7 Submit Datasets

Users with “Data_Submitter” privileges can submit data sets to the system.

Depending on the role users may upload the data of the type of the contributing institutes. Since each institute provides the data in its own format a different upload mechanism applies.

The following screenshot shows the heat flow file upload for the institute SMU:

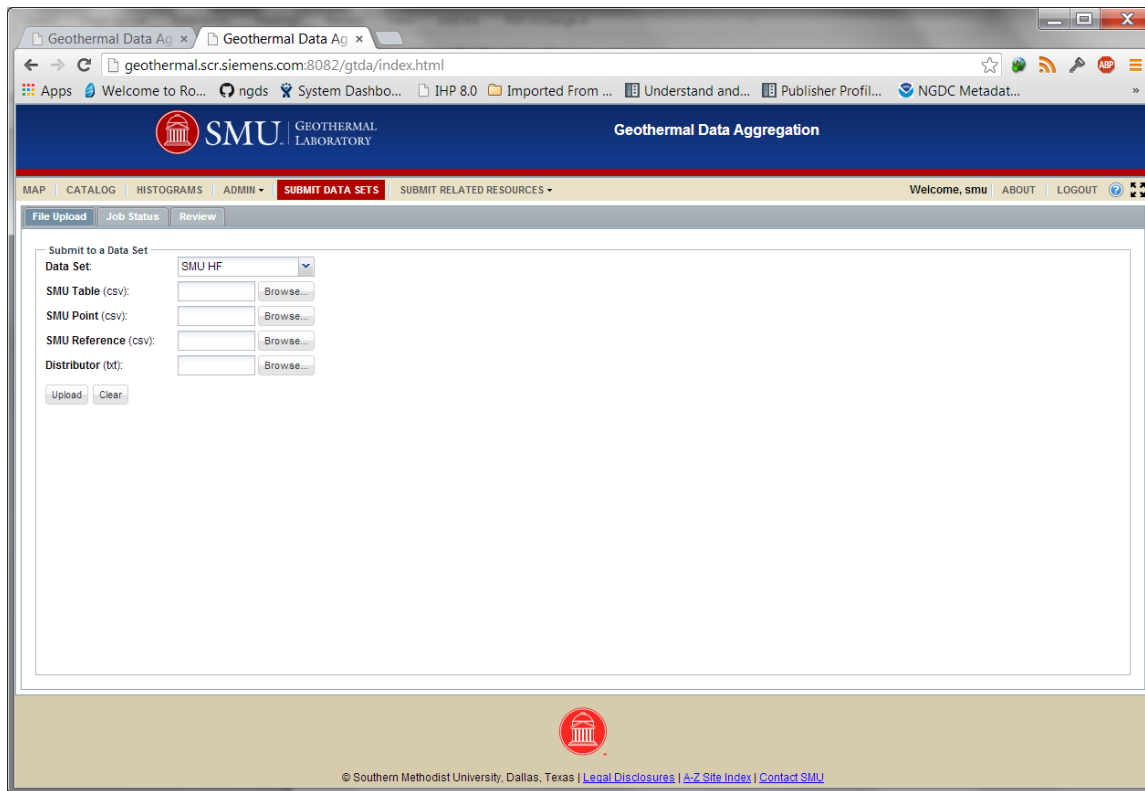


Figure 12: Screenshot of the Upload Page for SMU Heatflow Data.

As can be seen the system asks for 4 individual files that have to be uploaded.

Note that for each contributing organization, this page will look slightly different, and will accept input files of different formats.

Once the files have been uploaded the user can navigate to the “Job Status” tab. In this tab the user can monitor the progress of the upload process. When the job of uploading data is completed successfully the “Staged” message shows up. Failures are indicated as well. When clicking on the failure message the user gets some hints about what was wrong with the uploaded data, and where the offending issue can be found.

Also from the “Job Status” screen, the user can download the files that have been uploaded. Although the name of the file is different, the content is identical to what had been uploaded. The links appear in the rightmost column entitled “Download”.

The following screenshot shows the job status page for the SMU data upload.

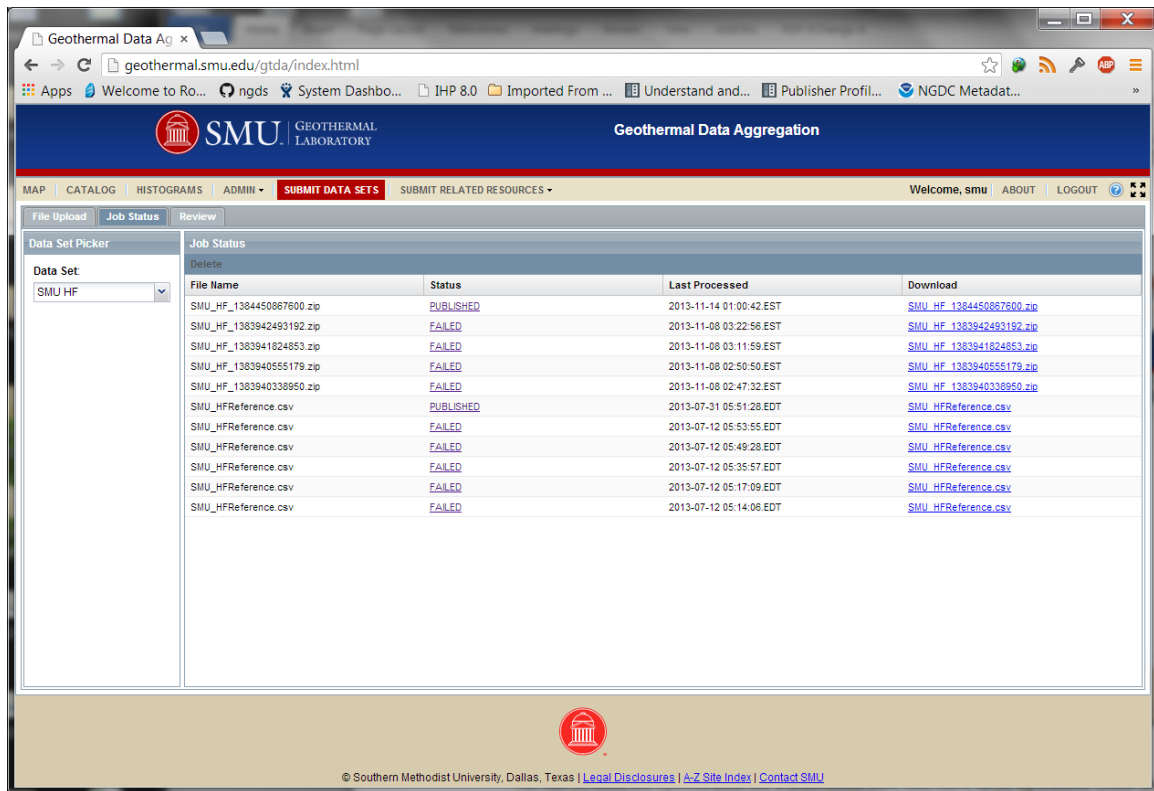


Figure 13: Screenshot of the job status tab.

Once data has been staged the user can choose the Review tab to check the uploaded data using histograms. The histogram functionality is similar to the histogram of the histogram tab. The following figure shows a screenshot for reviewing using histograms:



Figure 14: Screenshot of the data review panel.

Once satisfied with the data quality the user may publish the staged data. This is done by pressing the “Publish Dataset” button.

Note that in order to publish successfully, all three data sets for a given state must be staged.

Pressing the publishing button accomplishes the following:

- The data is aggregated along with all the already public data in the system. This includes incorporating this data into the appropriate AASG content model-compliant data sets.
- The catalog entry is updated based on any changes you made to the "Distributor" details when you uploaded the data set, making it easy for other services to find and use your data from this site.
- If the uploaded data is from Texas Tech the existing table containing only the Texas Tech University data set is updated.

The publishing process will take some time. Once the "Job Status" tab indicates it is complete, the data will appear on the map, and the web map service and web feature services will provide the newly updated data to anyone requesting it.

3.7.1 Publish dialog (activated when the publish button is clicked)

The publishing process has begun. The process will take some time to complete. See the "Job Status" tab to see whether it has completed successfully. If it has not, then the details of the failure will be presented there.

3.8 Submit Related Resources

Resources that are not data sets can be uploaded and (optionally) associated with wells via the menu items in the "Submit Related Resources" menu.

The "Bibliography Submission" menu item allows you to upload and update bibliography details for publications you are about to upload (To upload publications, see the "Publication Submission" menu item of the "Submit Related Resources" menu.).

Uploading publications and their bibliographic data are separated into two distinct steps: *Step 1* is to upload the bibliographic information and *step 2* is to upload the documents. They must be executed in this order.

It is assumed that there is not a document to be uploaded for every bibliographic entry but there is a bibliographic entry for every uploaded document.

The following two sections describe the two steps for uploading publications.

3.8.1 Bibliography Upload

To upload bibliographic data, use the "UNIVERSAL BIBLIOGRAPHY" item in the "Bibliography" pull-down menu, and provide an index.csv file.

Use the http://geothermal.smu.edu/static/manual/universal_bibliography.csv to help you arrange your data so that it will be import properly. The following aspects are important:

The combination of the fields "source" and "source_number" must be unique. The source number must be a number, not a string. We recommend to not do partial uploads but always a complete upload per source organization.

The following figure shows a screenshot of the upload page.

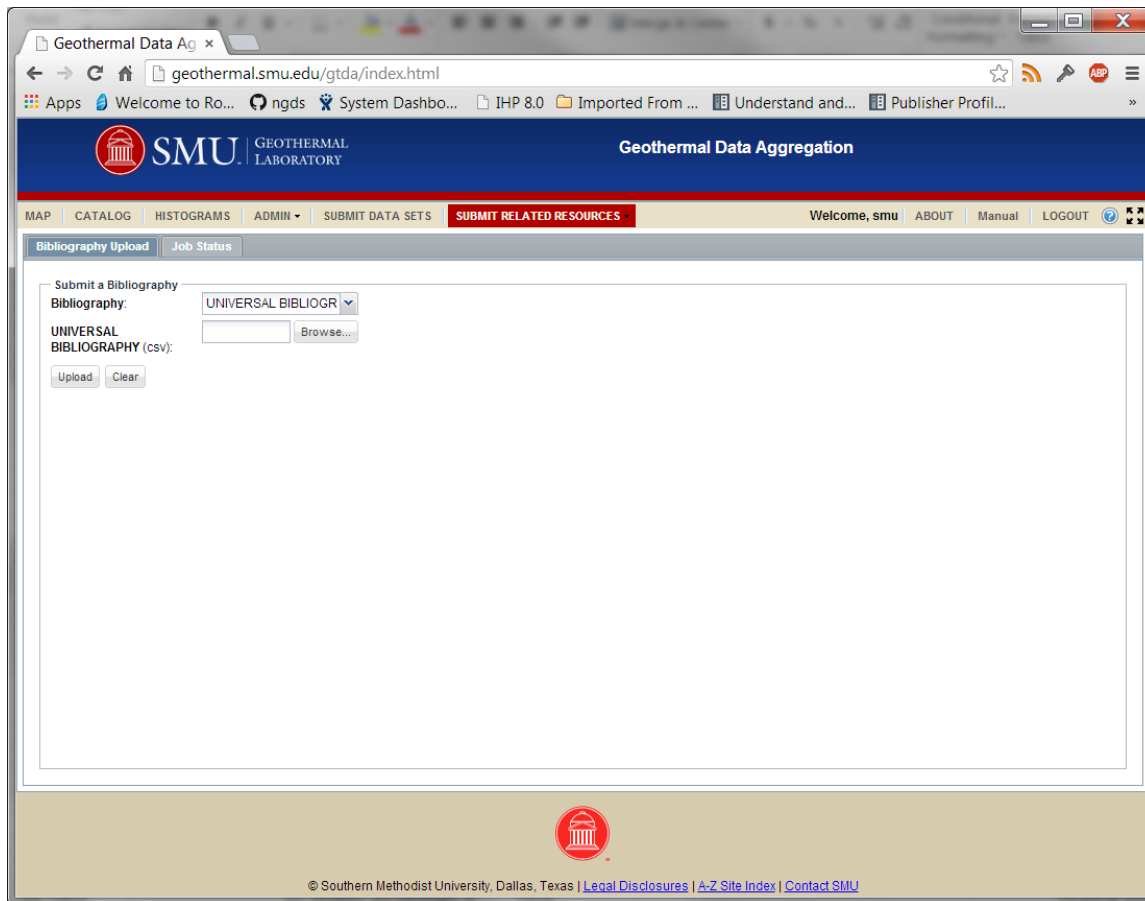


Figure 15: Screenshot of the Upload Mechanism.

For GRC, the bibliographic data should be delivered as a zipped Microsoft Access Database (MDB file).

The status of the upload process can be reviewed by clicking on the job status page:

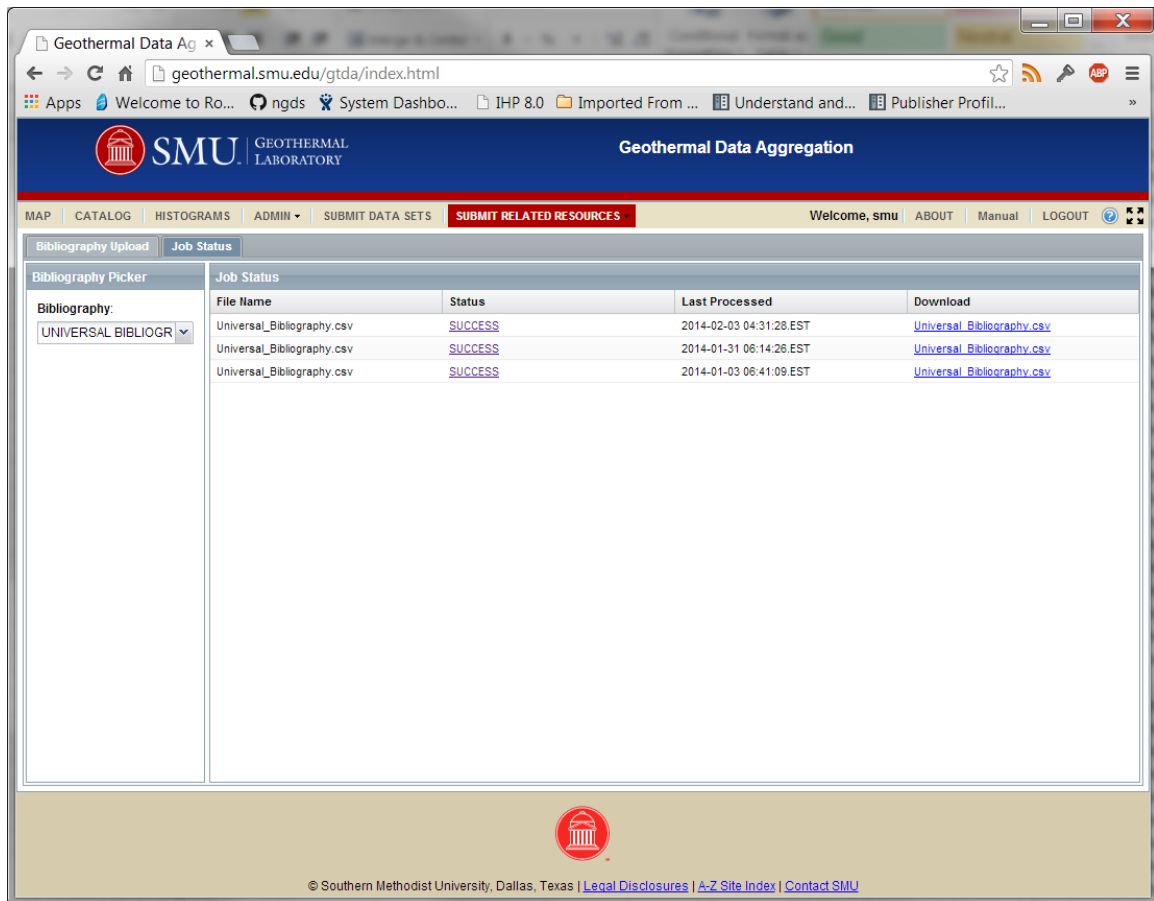


Figure 16: Screenshot of the Job Status Page for Uploading the Bibliography.

3.8.2 Publications Upload Page

Once the bibliography is successfully processed, proceed to the publication submission page. The following Screenshot shows the publications upload page.

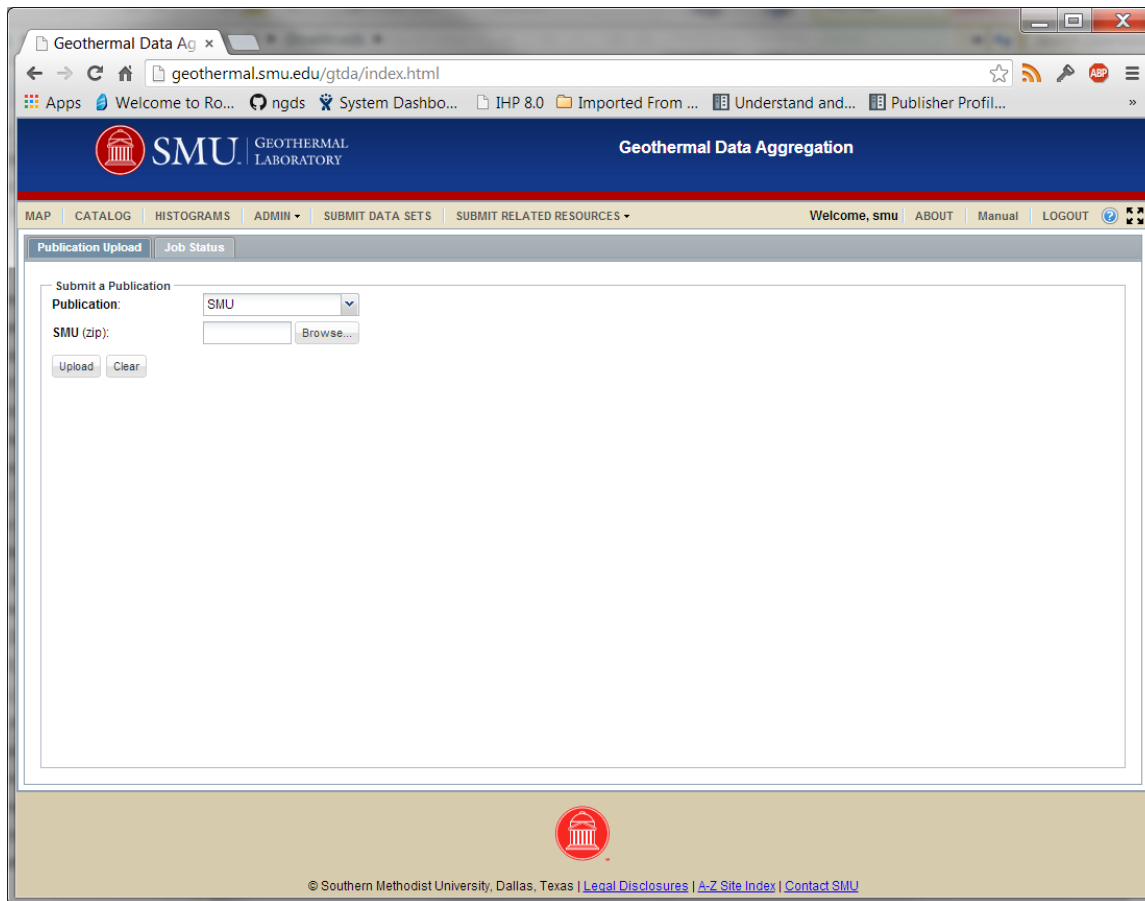


Figure 17: Publications Upload Page.

Publications are uploaded as zip files containing the publications as PDF. The ZIP file shall contain either or both of the following folders:

- SOURCE_Pubs_OK_to_distribute (example: SMU_Pubs_OK_to_distribute)
- SOURCE_Pubs_NOT_OK_to_distribute (example: SMU_Pubs_NOT_OK_to_distribute)

The first folder may contain publications that are ok to publish and the second may contain publications that must not be published.

Again, click on the job tab to review the status of the upload process:

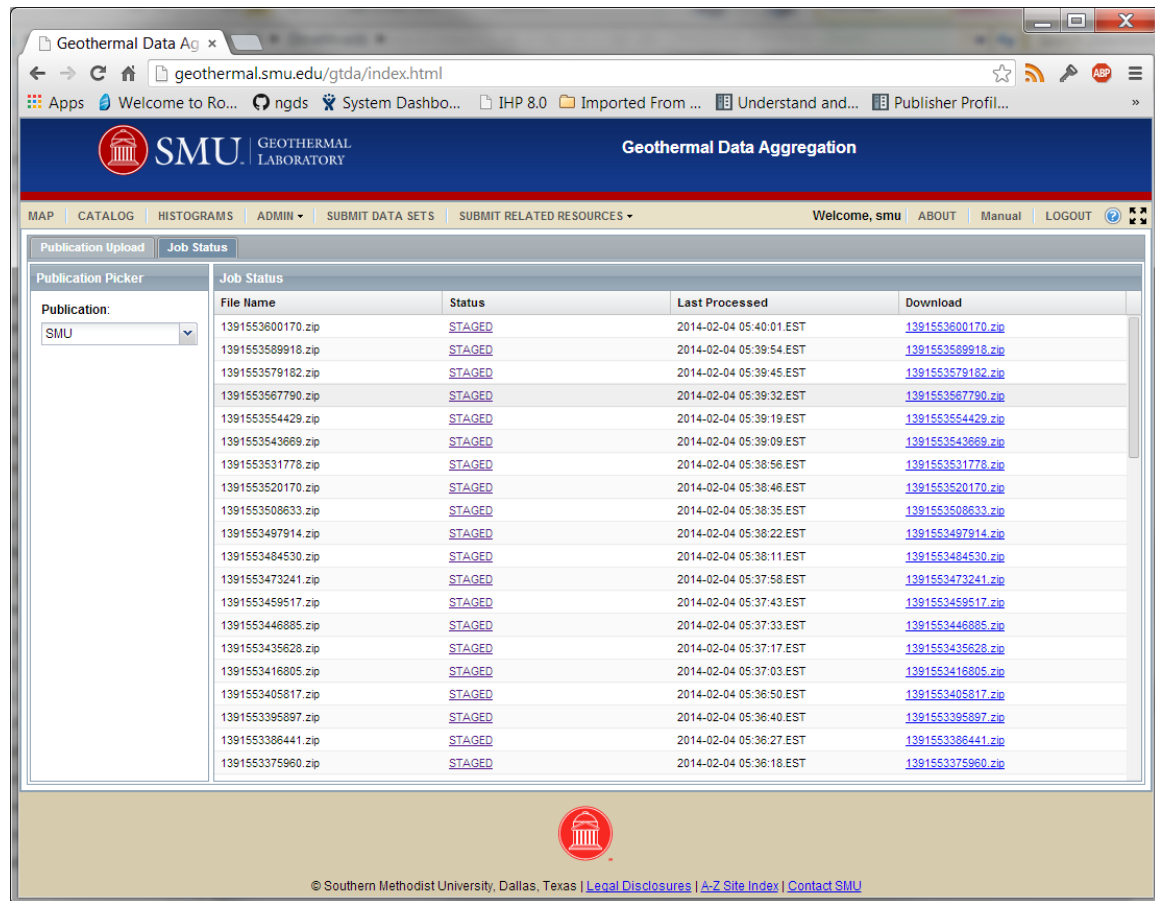


Figure 18: Job Status Page for the publications Upload Process.

3.8.3 Related Resources Submission

The "Related Resources Submission" screen allows you to upload items that you would like to have associated with wells. You can indicate the specific type of the resource (well log, well hog header, map, publication, comment, or field note) in the index.csv which must accompany the resources in the ZIP archive.

There are three "upload types" from which to choose. The upload type determines how the system processes the uploaded resource. The following three types are available:

- Related Resources
- Field Notes
- TD Curves

The following Screenshot shows the upload page for TD curves:

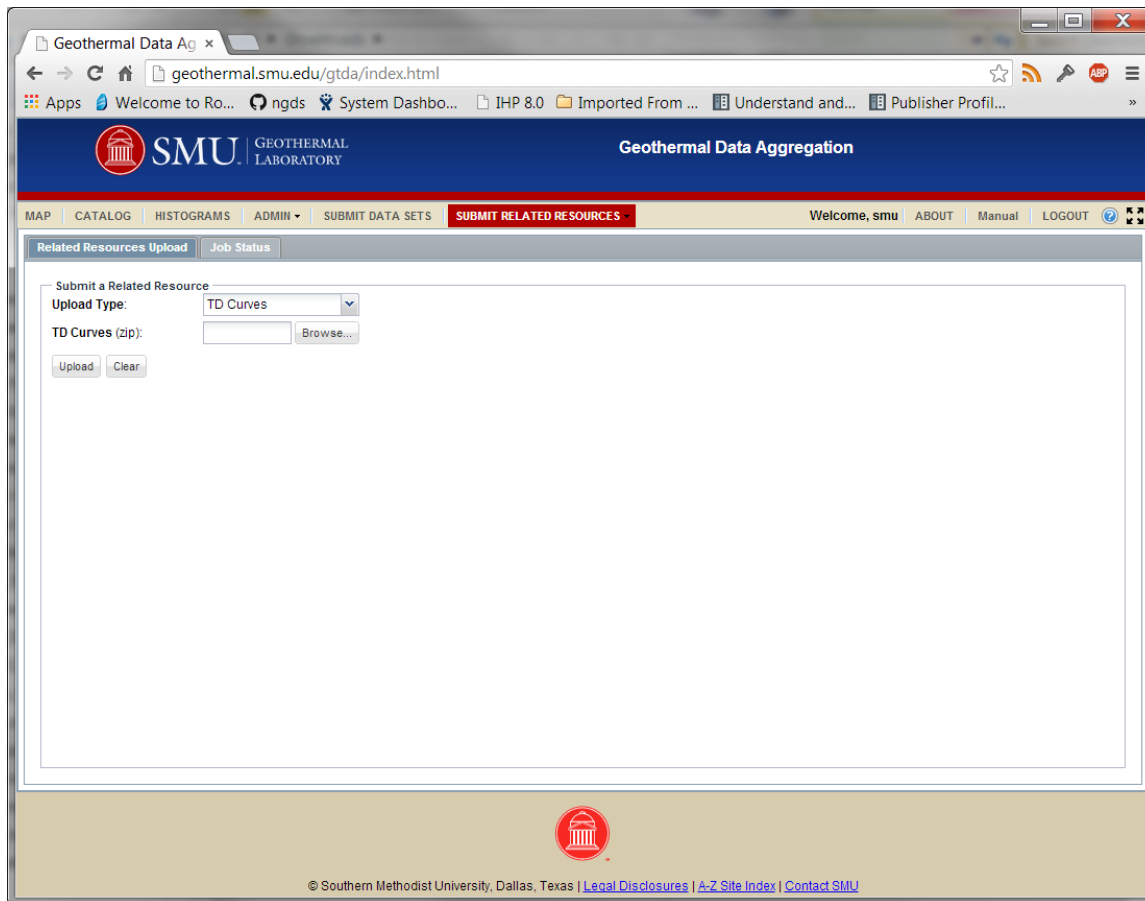


Figure 19: Upload Page for TD curves.

Again the Job Status Page informs the user about the success of the upload. In the case of related resources, a status of “STAGED” indicates that the upload was successful and no further action is required. There is no publication step as with data sets. Once the upload is complete and the resources are linked to the associated sites, you should be able to find them on the sites' well details pages under the appropriate section. The well details pages are accessible via the Map search, and via the histogram table.

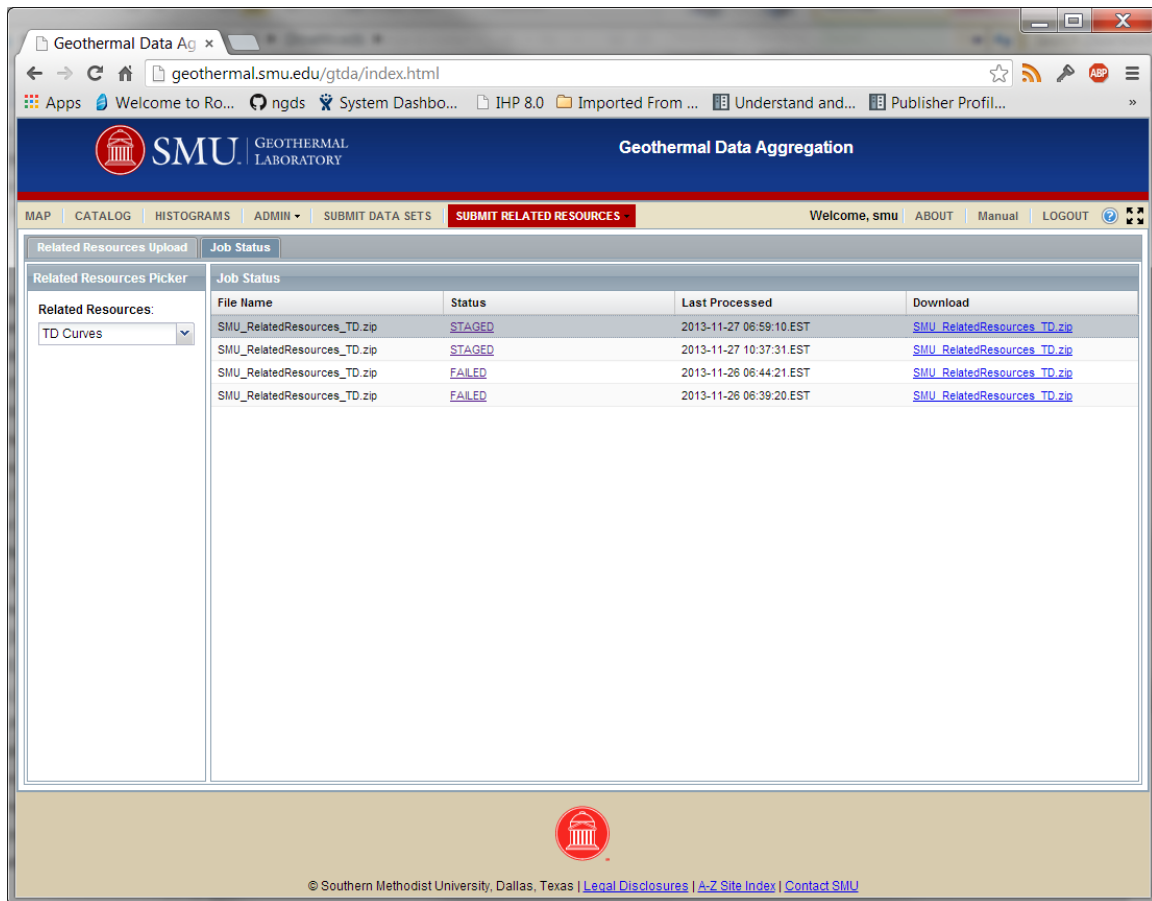


Figure 20: Job Status Page for uploading TD Curves..

In the following we give some brief details on the three different types of related Resources that can be uploaded.

3.8.3.1 Related Resources

Use the "Related Resources" upload type to upload any type of related resource. This includes well logs, well log headers, maps, publications, and comments. It also includes field notes for which the title of the document DOES NOT begin with the name of the site to which it is to be associated.

3.8.3.2 Field Notes

Use the "Field Notes" upload type only when each individual field note file in the ZIP archive is named with the first eight characters indicating the site to which it is to be associated.

3.8.3.3 TD Curves

Use the "TD Curves" upload type when uploading temperature depth curves. Up-loading via this type allows the system to process the temperature-at-depth data so that it can be rendered as a chart on the surface site's well details page.

4 Terms

In the following we list a set of important terms and definitions.

4.1 Acronyms, and Abbreviations

The following table lists the abbreviations used in this document, in order to promote their unique and unambiguous usage throughout the document and the Project.

Abbreviations	Definition
API	Application Programming Interface
CSV	Comma-Separated file format
CSW	Catalogue Service for the Web
DIS	Data Import Schema
DOE	Department of Energy
NetCDF	Network Common Data Form
NGDS	National Geothermal Data System
OAI-PMH	Open Archives Initiative – Protocol for Metadata Harvesting
OGC	Open Geospatial Consortium
ORM	Object-Relational Mapping
SDD	Software Design Description
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
UTM	Universal Transverse Mercator coordinate system
WCS	Web Coverage Service
WFS	Web Feature Service
WMS	Web Map Service
WSS	Web Services Specification

Table 2: Abbreviations

4.2 Glossary of Geological Terms

The following table lists the terms used in this document, in order to promote their unique and unambiguous usage throughout the document and the Project.

Terms	Definition
Geological feature	Any type of geological phenomena such as: • Faults • Fluid inclusions • Fluid rock interaction [Use for non-water fluids only, usually carbon dioxide] • Hydrothermal alteration [Same as rock-water interaction] • Fractures • Magmatism [Includes magma and magmatic processes; use more specific term “Volcanism” if applicable, although some articles address both; see also “Magma energy”] • Volcanism [Includes volcanoes and volcanic processes] • Permeability [Do not overuse; ability of fluid to flow through rock] • Pore pressure [Includes capillary pressure] • Porosity [Do not overuse; amount of pore space within rock] • Rock mechanics [Response of rocks to external forces, especially related to stress] • Seismic attenuation [Ability of rocks to absorb seismic waves] • Seismicity [Same as earthquakes; see also “Induced seismicity”] • Microseismicity [Same as microearthquakes] • Seismic velocity • Stress fields • Borehole breakouts [Bulging of wells, related to stress fields] • Tectonics [On a regional scale] • Any rock formations treated in detail ○ Breccia ○ Granite ○ Greywacke [also spelled graywacke] ○ Sedimentary basins
Data	Documents, Computer database, pictures, spreadsheets about geological features
Metadata	Summary of important characteristics of a piece of data used for the purpose of searching and discovery by users of a geological feature database.
Dataset	A group of geological data, typically related to one geological feature.
Geological survey	Gathering data using geological methods, or descriptions of regional geology; includes structure

Resources	A generic name to include data, metadata and other artifacts that are under the responsibility of a user.
Geophysical survey	Different types of descriptions of physical characteristics of regional characteristics including electromagnetic surveys, gravity surveys, magnetic, seismic, shallow temperature surveys, etc.
Heat flow	Energy coming to the surface in a given area, often measured in milliwatts/m ² and used in resource assessment; derived by multiplying temperature gradients and thermal conductivity; essentially equivalent to heat flux, which is an engineering term
Temperature gradients	Change in temperature with depth, e.g., °C/m; includes temperature depth profiles
Thermal conductivity	Ability to conduct heat; e.g., salt domes have high conductivity, while sedimentary basins have low conductivity
Well logging	Well measurements and descriptions, including borehole geology
Metadata template	A predetermined set of metadata attributes used to describe data about geological features. E.g. document meta-data include a document bibliographical citation, its geo-location, ownership, and URI (for the actual document)
Landmark	Locations and regions in the globe that are geo-referenced: city, district, state, county, known geothermal sites, etc. A land mark is implemented as a dictionary, a map from names into geographical coordinates.

Table 3: Terms

Siemens AG

GEOTHERMAL DATA AGGREGATION Operations Manual

Version: V1.2

Passwords REMOVED For Final Report

Version:	Date:	Author:	Details
1.0	01/15/2014	Siemens	Operations Manual for GTDA.
1.1	04/18/2014	Siemens	Reorganized sections as requested; added/edited content regarding backup/restore
1.2	6/19/2014	SMU	Adjusted header and footer to indicate Appendix D of Final Report. Version received May 8 th is identical except for this.

Table of Contents

1	Preface.....	3
1.1	Purpose and Audience	3
1.2	Document Roadmap	3
2	System Context	4
3	Component Model.....	5
3.1	Dynamic Views	5
3.1.1	Use Case: Load Index Page	5
3.1.2	Use Case: Filter Page	6
3.1.3	Use Case: Select Area in Map and get details about one Well	7
3.2	Component Descriptions: Client Side	8
3.2.1	Ext JS v4.0.7	8
3.2.2	OpenId client.....	9
3.2.3	OpenLayers v.2.12	9
3.2.4	GTDA Client	10
3.3	Component Descriptions: Server Side	14
3.3.1	Tomcat Application Server v6.0.24.....	14
3.3.2	OpenId Server	15
3.3.3	Shiro.....	15
3.3.4	ESRI Geoportal Server v1.2.4	15
3.3.5	GeoServer v.2.4	16
3.3.6	Jetty v9.1	17
	Apache Solr v4.6.0.....	18
3.3.7	GTDA Server.....	18
3.4	Database.....	19
3.4.1	Entities	20

3.4.2	GTDA database schemes	22
3.5	Connectors.....	22
4	System Administration and Maintenance.....	23
4.1	Passwords	23
4.2	Starting and Stopping the System.....	23
4.3	Backup and Restore.....	24
4.3.1	What to Backup	24
4.3.1.1	Data.....	24
4.3.1.2	Application	25
4.3.2	How to back up.....	25
4.3.2.1	Backup options for files and folders.....	26
4.3.2.2	Backing up the database	26
4.3.3	How to restore data	26
4.3.4	Commands to check the disk space	27
4.4	Regeneration of CSW Entries.....	27
4.4.1	Trigger Indexing of Publication data.....	28
4.5	Location of Data and Components.....	28
4.5.1	Tomcat.....	28
4.5.2	Postgres.....	29
4.5.3	SOLR.....	32
4.6	GeoServer	33
4.7	Geoportal.....	33
4.8	GTDA Application.....	33
4.9	Access to Gray Box about help documentation	34
4.10	Documents.....	35
4.11	OS Maintenance, Backup and Restore	35

4.11.1	OS Maintenance	35
4.11.1.1	OS Updates.....	35
4.11.1.2	OS Backups	36
4.11.1.3	OS Restores	37
4.11.1.4	Alternatives	38
4.11.1.4.1	Activate Rollback Feature of Yum.	38
4.11.1.4.2	Install Backup tool	38
5	Appendix 1: Terms and Glossary.....	39
5.1	Acronyms and Abbreviations.....	39
5.2	Glossary of Geological Relate Terms	40
6	Appendix 2: Database Schema	42

List of Figures

Figure 1 Context diagram	4
Figure 2 Main components of the GTDA system	5
Figure 3 Loading index page	6
Figure 4: Working with filters in the filter page	7
Figure 5 Select area in a map, and get details about a well.....	8
Figure 6 Log-in screen supporting OpenID authentication	10
Figure 7 GTDA main screen with filter tab and wells map	11
Figure 8 GTDA Catalog search	12
Figure 9 GTDA histograms.....	13
Figure 10 GTDA role-based user management	14
Figure 11 Geoportal main screen	16
Figure 12 GeoServer layers database	17
Figure 13 Solr repository browser	18
Figure 14 GTDA database	20
Figure 15 GTDA Entities.....	21
Figure 16: Screenshot of pgAdminIII	30
Figure 17: Screenshot of table 'publication'.....	31
Figure 18: Screenshot of the view 'publication_metadata.....	32
Figure 19: Snapshot of Oracle VirtualBox User Interface highlighting the Snapshot Feature.	36
Figure 20: Oracle VirtualBox User Interface showing the snapshot management interface. right-click on a Snapshot to restore it.....	37

List of Tables

Table 1: Abbreviations and Definitions (list in alphabetical order)	39
Table 2: Terms used in the GTDA System	40

1 Preface

1.1 Purpose and Audience

This document is intended for a technical audience who need to understand how to operate, upgrade software, and make minor changes to the software developed by Siemens Corporate Research for the SMU Geothermal Thermal Data System (GTDA) system staged at <http://geothermal.smu.edu>. Targeted audiences include:

- **System Operators**
- Software Architects
- Software Developers
- DoE Monitors

The purpose of the document is to clarify for the audience how to operate the GTDA software system and describe from an operations point of view, the software implemented by Siemens to support the SMU Geothermal Laboratory's efforts for the project.

GEOHERMAL DATA AGGREGATION: SUBMISSION OF INFORMATION INTO THE NATIONAL GEOHERMAL DATA SYSTEM (U.S. Department of Energy award number DE-EE0002852).

1.2 Document Roadmap

This document outlines the architecture of NGDS and is structured in the following way:

- Sections 0 and 3 outline the system context and provide a high-level overview of features
- Section 3 lists important system administration information such as how to create backups and how to maintain the system.
- Section 5 is an appendix of terms and acronyms
- Section 6 lists the database schema and describes how to generate it from the database.

2 System Context

The GTDA System facilitates the uploading, integration and submission of geothermal data (Figure 1). The System handles the submission of the meta-data records to the GTDA System from the SMU Project Participants, and transmits instances of the geothermal data, when requested. Also, the System provides a mechanism to handle maintenance issues relating to the operation of the system.

- Process data
- Analyze data
- Perform data quality assurance

Figure 1 Context diagram

The System supports the following feature groups:

- Data input
- Data browsing
- Quality assurance
- Data submission
- System administration

3 Component Model

Figure 2 represents the main components of which the system is comprised. Some components are built from scratch, such as the GTDA server, whereas other components are off-the-shelf components. These include: Geoportal, GeoServerGeoServer and Solr. All components, are delivered via the Tomcat and Jetty application servers (also off-the-shelf components).

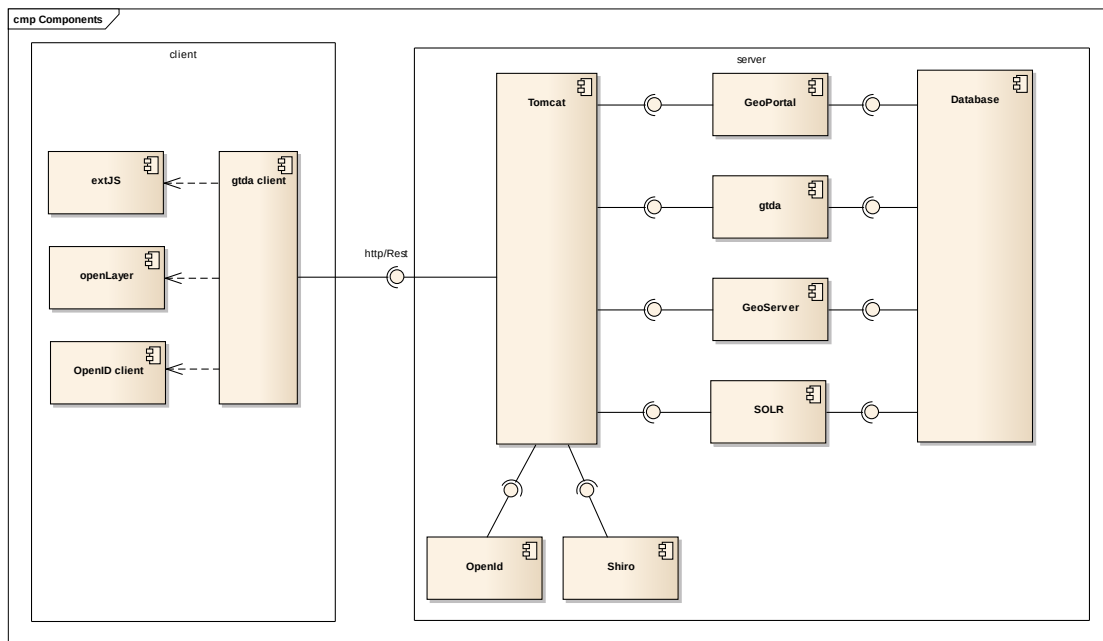


Figure 2 Main components of the GTDA system

3.1 Dynamic Views

In the following we discuss three use cases that visualize how data are being transported between the main system components. Our goal is to capture the main usage scenarios of the system, illustrating the role of the main system components in each scenario.

3.1.1 Use Case: Load Index Page

This is the initial use case in the system: The user browses to the home page of the system. We assume that the user is already logged in.

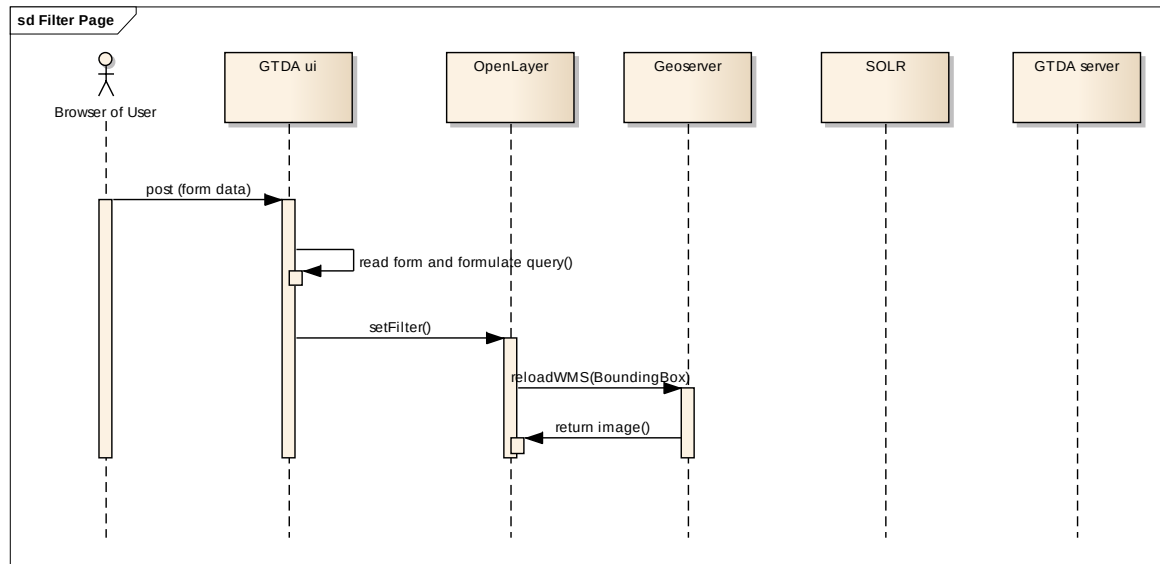


Figure 3 Loading index page

As can be seen the user receives the ExtJS-based Web page. The page creates an initial query (i.e. no filter at all, show all wells in the map area). The OpenLayer framework then requests the images from GeoServer. Both GTDA UI and OpenLayers are in the client side.

One aspect that can be seen here is how the UI communicates with all components of the server. Another interesting fact is that the GTDA System Server does not directly participate in this interaction, i.e., the results are retrieved from third party components without passing through the GTDA System API.

3.1.2 Use Case: Filter Page

In this use case the user modifies the filter for wells to be presented on the map. The user modifies the filter criterion (e.g. show only wells with a minimum temperature of at least 32°C) and submits the request.

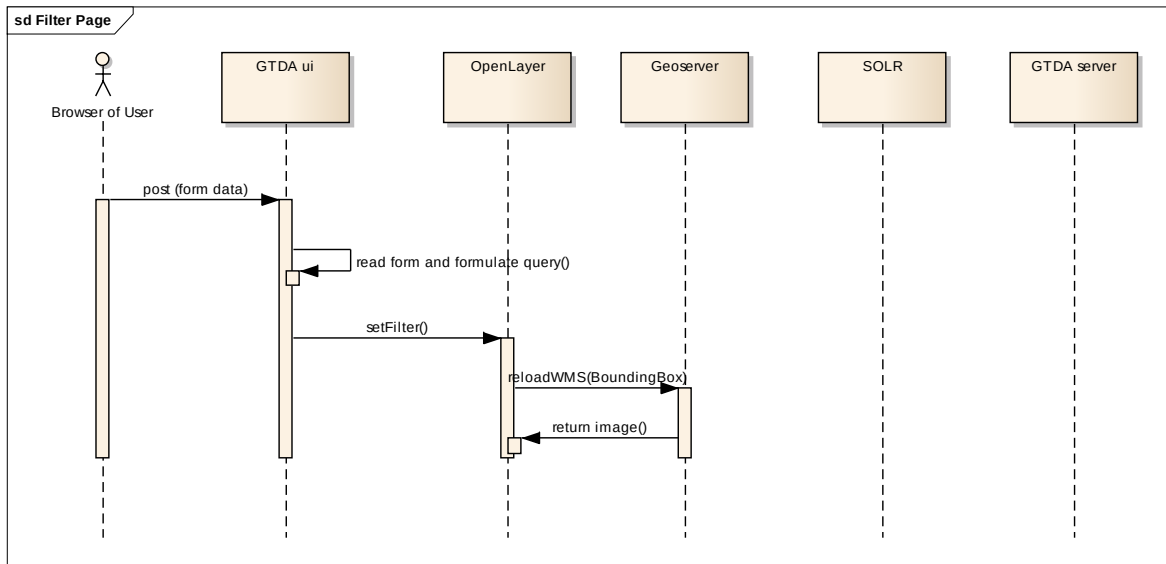


Figure 4: Working with filters in the filter page

The interesting aspect that can be seen on this request is that the UI is essentially creating a query which is then sent to OpenLayer from where it is sent to the GeoServer which renders the images.

3.1.3 Use Case: Select Area in Map and get details about one Well

The following image shows two use cases in a single diagram: First the user selects a certain region and afterwards he retrieves detailed information on one individual well.

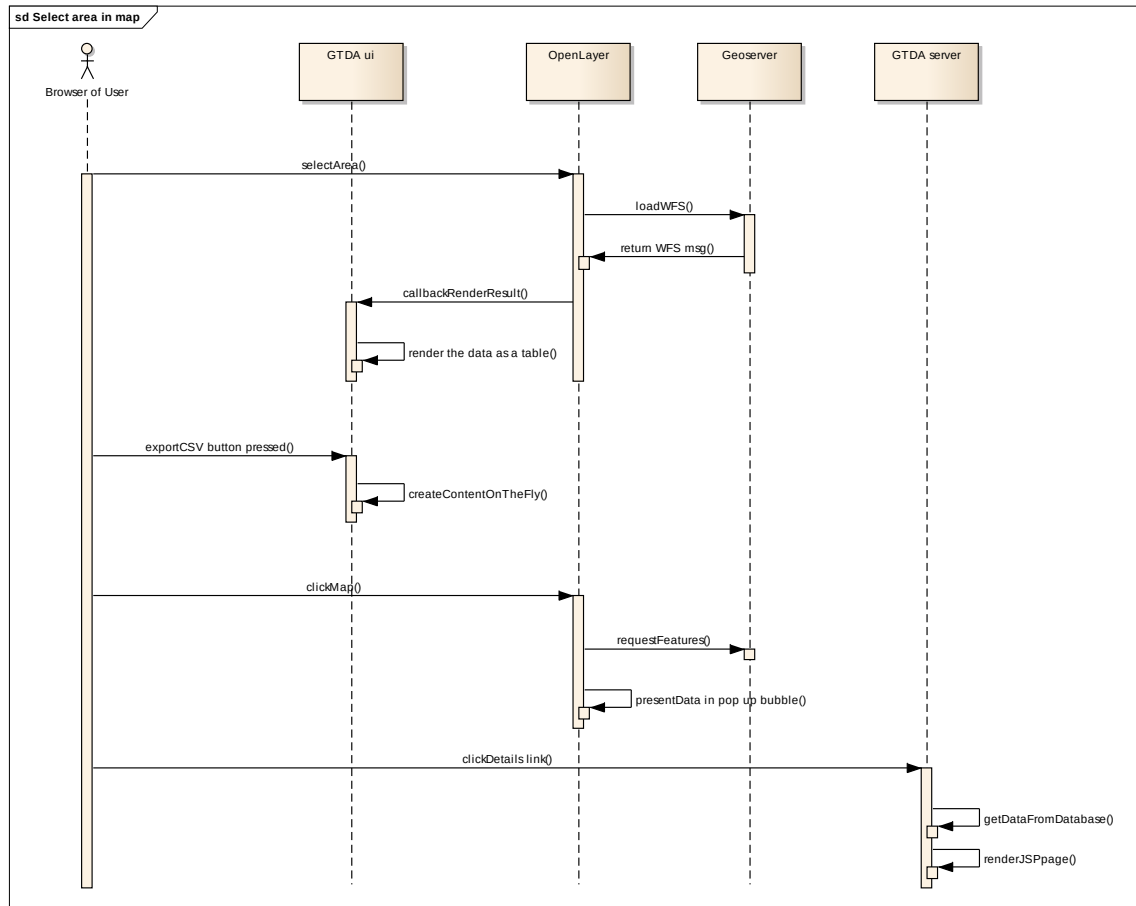


Figure 5 Select area in a map, and get details about a well

Note that in Figure 5, GTDA server participates only when more detailed structured data from the database is being displayed. All other interactions, including information about well location is gathered directly from the GeoServer.

3.2 Component Descriptions: Client Side

At the client-side, the system integrates three main components: the Ext JS toolkit library for JavaScript, implementing the user interface (or UI), the Open Layer library for dealing with maps and layers, and the OpenID client to support user authentication. We describe these components in more detail as follows.

3.2.1 Ext JS v4.0.7

Ext JS is a JavaScript UI framework provided by Sencha (<http://www.sencha.com/products/ExtJS/>). It comprises a set of widgets and a MVC programming model that allows the development of desktop-like UI within the web browser. Ext JS implements the model-view-controller pattern, and supports AJAX as the pattern for interaction with the server side Web services. AJAX allows scripts to update the content of a web site utilizing asynchronous XMLHttpRequest commands, described in XML or JSON (JavaScriptObject Notation). The results are used to modify the

web page content, via DOM model, without the need to reload the whole application web page. The use of MVC promotes separation of concerns, and improves maintainability of the client code; the use of Ajax (Asynchronous Java Script and XML) provides a fast and rich web client experience; whereas the use of RESTful web services implementation, mediating the communication between client and server through a well defined Web Services interface supports the independent evolution of client and server sides. Additional features of Ext JS include: customization, internationalization, and support for a set of widgets to display geo-located information such as GeoExt (<http://geoext.org/>). However, in the current implementation, we utilize OpenLayers library, further described below, which provides a more widely used and standards-compliant implementation of map widgets for JavaScript.

3.2.2 OpenId client

OpenId is a Web services standard for handling authentication defining standard protocols and message representations. It allows the authentication of an application to be delegated to third party identity providers. After authentication, users can sign in to a web site. Example authentication providers include Yahoo, Google, AOL, IBM, VeriSign and others. In particular, GTDA supports the authentication providers listed at Figure 6. Authentication providers can also store information such as name, gender, and other attributes. These can be shared with the application after authentication is performed. In addition, to use third party OpenID authentication services, GTDA implements OpenID standard in order to provide its own authentication. This will be further discussed, later on, in the OpenId Server section.

3.2.3 OpenLayers v.2.12

OpenLayers (<http://openlayers.org>) is a free JavaScript library for displaying and navigating through layered maps in Web applications. It is currently developed by the Open Source Geospatial Foundation (<http://www.osgeo.org/>) whose main goal is to promote the collaborative development of geospatial software.

OpenLayers provides a zoom-able canvas where map tiles from different providers (e.g. Google, Yahoo) can be mashed up with layers from third party map layer providers such as GeoServer instances. OpenLayers is compatible with the Web Map Service (WMS) standard protocol from the Open Geospatial Consortium. This protocol is used for representing and transmitting geo-referential images over the Web.

In GTDA, OpenLayers is used to render the Geo-located well data on a USA map as shown in Figure 7.

3.2.4 GTDA Client

The GTDA client implements the main look and feel of the application. It supports the search and visualization of geo-located information from the GTDA server and its auxiliary components as the GeoServer and Solr. This information includes data from the Aggregated Wells database, in addition to documents published about those wells.

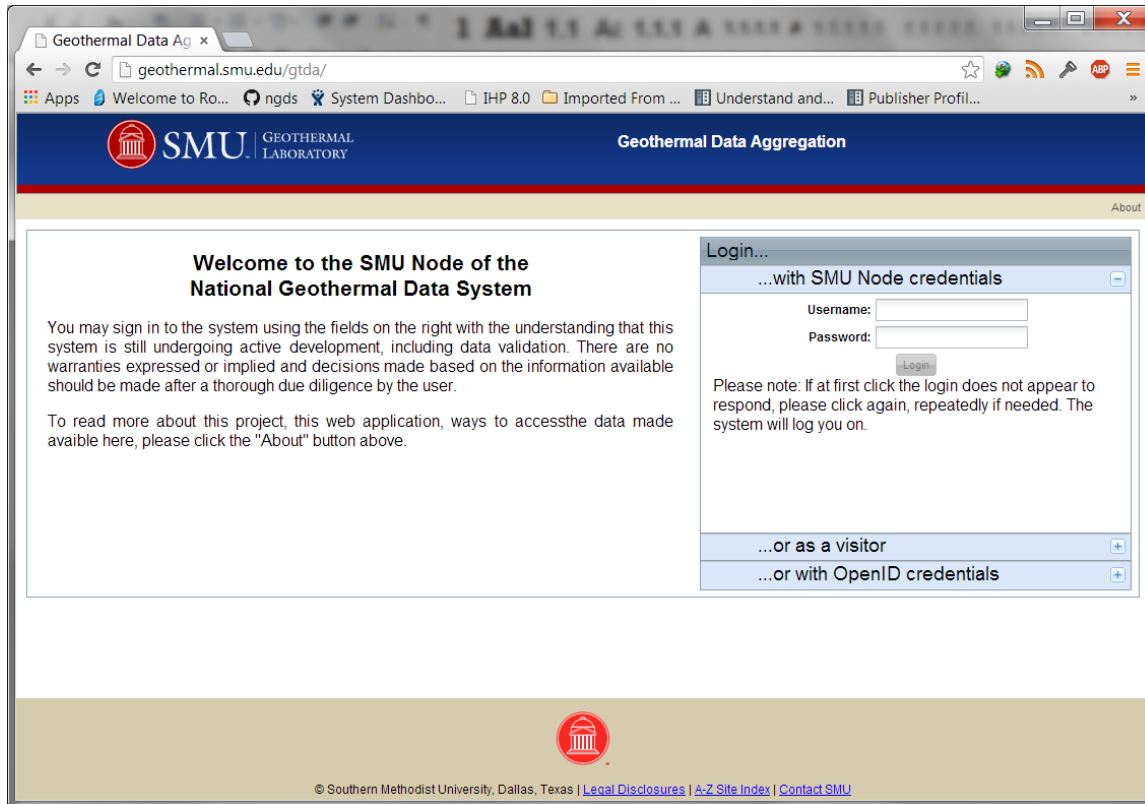


Figure 6 Log-in screen supporting OpenID authentication

Once authenticated via the screen of Figure 6, the user is granted access to the main GTDA application screen as shown in Figure 7. The application allows the visualization of well information on a World map. This information can be refined or filtered using the controls on the left hand side. Red squares represent a set of wells filtered by the user-provided criteria (in the left hand side). By clicking on a red dot in the map, a pop-up with well information details is displayed.

The map browsing feature is supported by OpenLayers UI component, in the client side, and by layers provided by GeoServer, in the server side. The UI component supports filtering of information with the help of GTDA server data. The well details are provided by the GTDA-Server component.

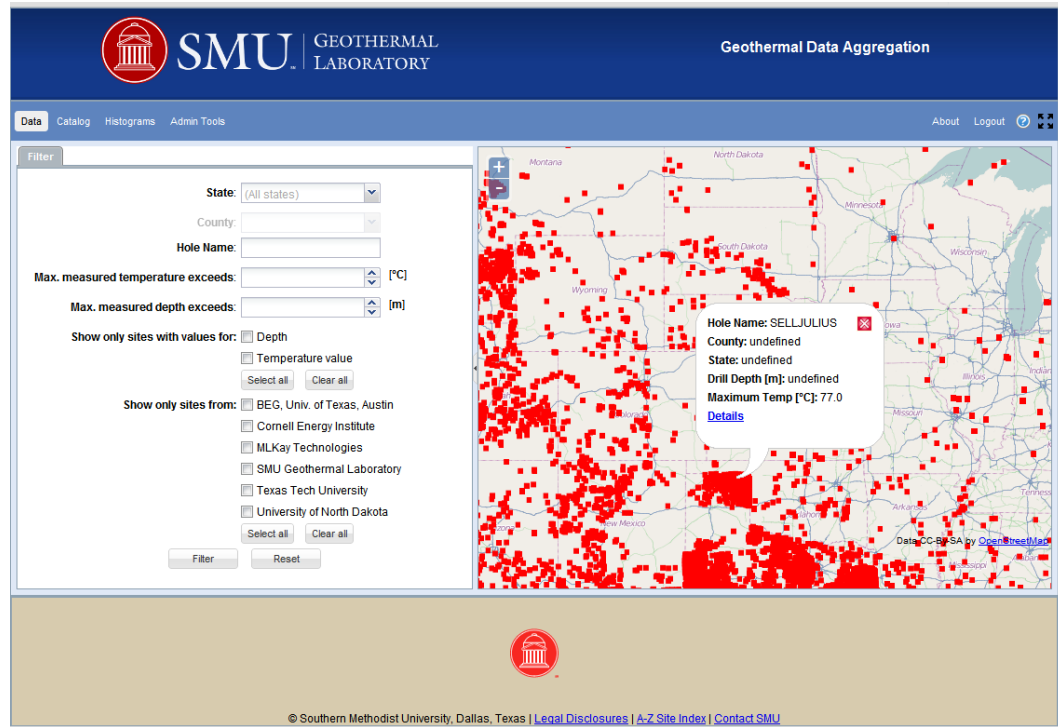
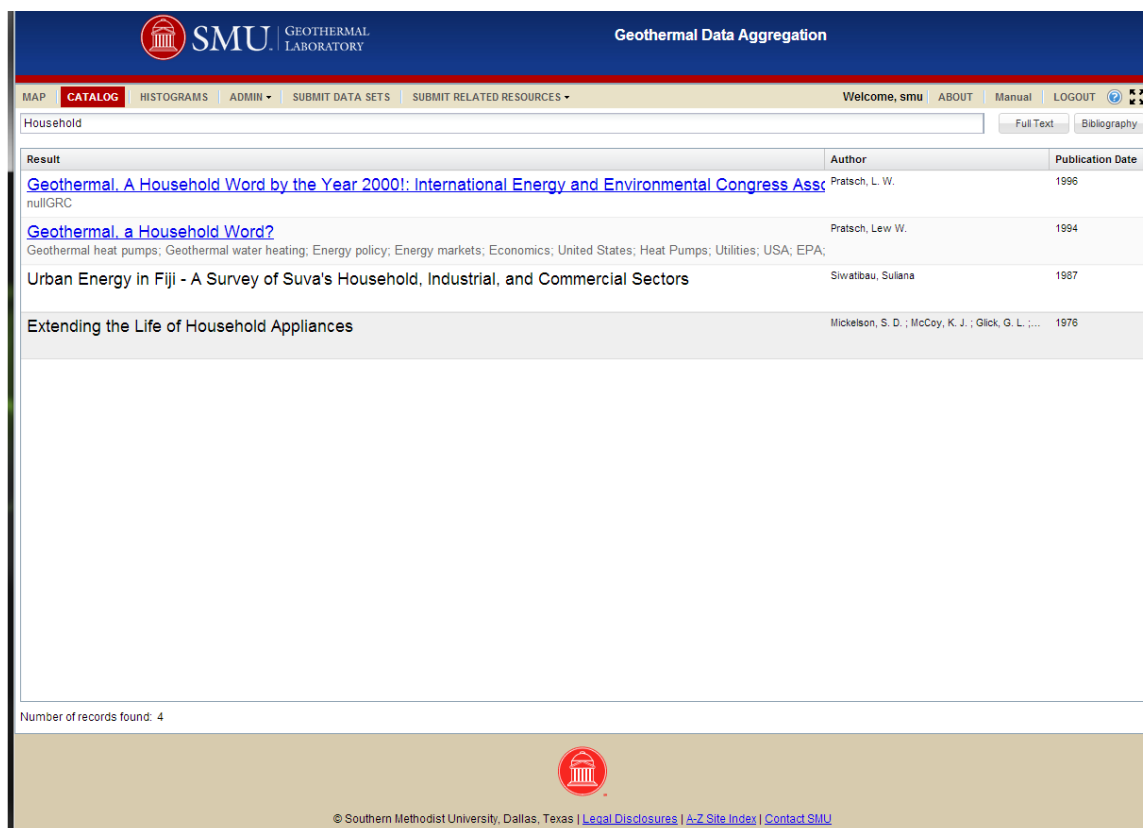


Figure 7 GTDA main screen with filter tab and wells map

Besides storing structured information about each well, e.g. drill depth, maximum temperature, and geo-location, GTDA stores additional documents associated to each well. In the catalogue view, as shown in Figure 8, the user can do a free text search for documents in the local wells catalogue. Documents are then shown. In the server side, this free text search capability is implemented by the Solr component.



The screenshot displays the Geothermal Data Aggregation (GTDA) Catalog search results page. The page features a blue header with the SMU Geothermal Laboratory logo and the title 'Geothermal Data Aggregation'. Below the header is a navigation bar with links: MAP, CATALOG, HISTOGRAMS, ADMIN, SUBMIT DATA SETS, and SUBMIT RELATED RESOURCES. A search bar contains the text 'Household'. To the right of the search bar are buttons for 'Full Text' and 'Bibliography'. The search results are displayed in a table with three columns: Result, Author, and Publication Date. The table contains four records. The first record is 'Geothermal. A Household Word by the Year 2000!: International Energy and Environmental Congress Ass...', authored by 'Pratsch, L. W.', with a publication date of '1996'. The second record is 'Geothermal. a Household Word?', authored by 'Pratsch, Lew W.', with a publication date of '1994'. The third record is 'Urban Energy in Fiji - A Survey of Suva's Household, Industrial, and Commercial Sectors', authored by 'Sivatabau, Sulana', with a publication date of '1987'. The fourth record is 'Extending the Life of Household Appliances', authored by 'Mickelson, S. D. ; McCoy, K. J. ; Glick, G. L. ; ...', with a publication date of '1976'. Below the table, it states 'Number of records found: 4'. The footer of the page includes the SMU logo and the text '© Southern Methodist University, Dallas, Texas | Legal Disclosures | A-Z Site Index | Contact SMU'.

Result	Author	Publication Date
Geothermal. A Household Word by the Year 2000!: International Energy and Environmental Congress Ass... nullIGRC	Pratsch, L. W.	1996
Geothermal. a Household Word? Geothermal heat pumps; Geothermal water heating; Energy policy; Energy markets; Economics; United States; Heat Pumps; Utilities; USA; EPA;	Pratsch, Lew W.	1994
Urban Energy in Fiji - A Survey of Suva's Household, Industrial, and Commercial Sectors	Sivatabau, Sulana	1987
Extending the Life of Household Appliances	Mickelson, S. D. ; McCoy, K. J. ; Glick, G. L. ; ...	1976

Figure 8 GTDA Catalog search

Some rudimentary analytics are also provided. Sets of well information can be gathered and visualized as graphics in the histograms screen, as illustrated in Figure 9, where the elevation of a group of wells is being compared.

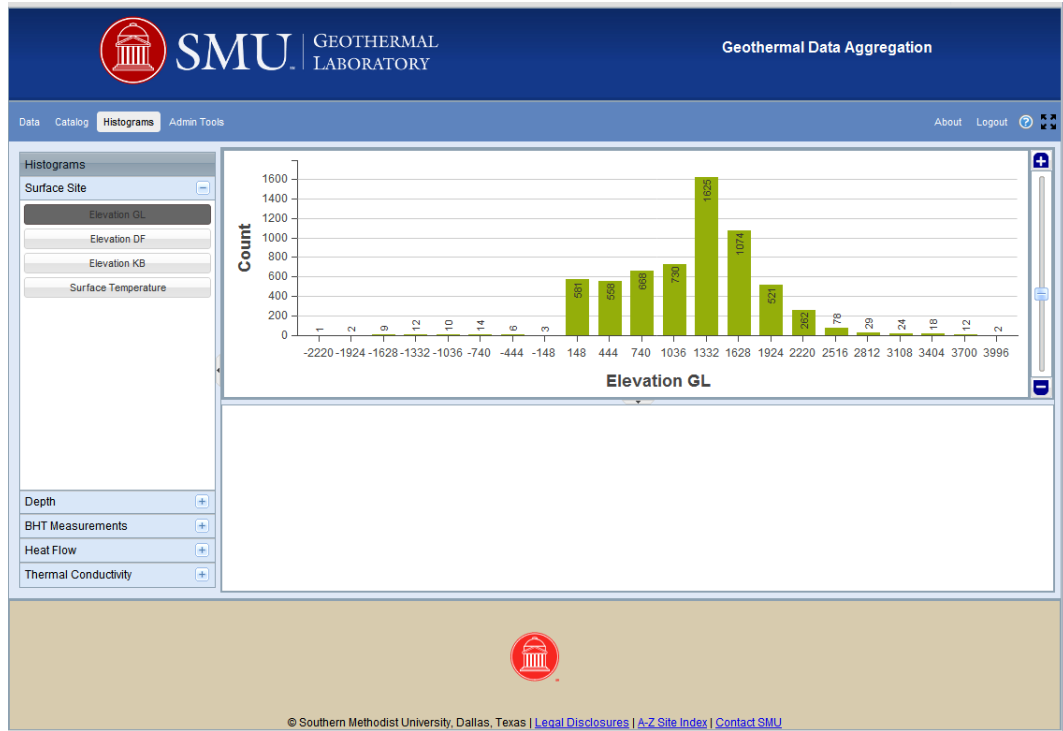


Figure 9 GTDA histograms

Finally, in the admin screen, administrators can create new users and groups, controlling their access to system information. The access control is role based.

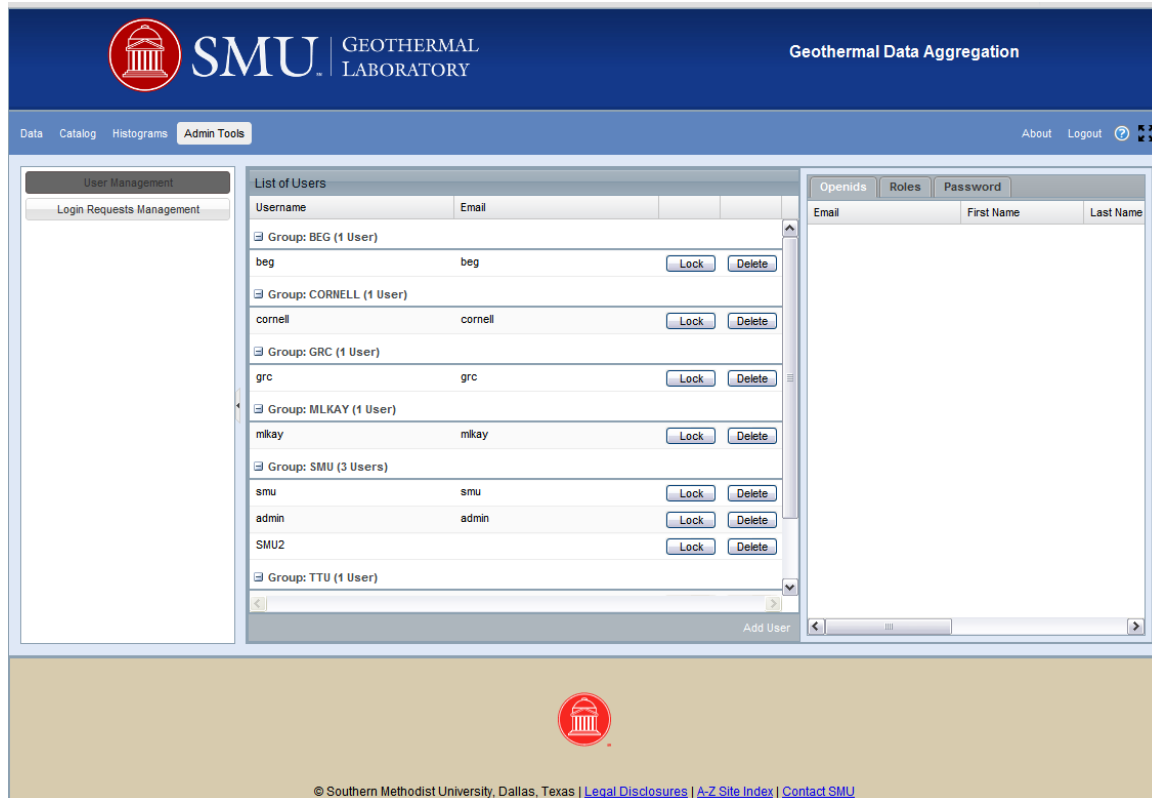


Figure 10 GTDA role-based user management

3.3 Component Descriptions: Server Side

The server side of GTDA is implemented as a combination of Tomcat web applications that are built on top of a Postgres database augmented with GIS extensions. The server-side functionality is provided by major Tomcat applications including Geoportal, GeoServer, and Solr. The GTDA provides meta-data about wells; GeoServer provides map layers based on the WCS requests; while Geoportal supports CSW request. Solr provides search capability for documents.

3.3.1 Tomcat Application Server v6.0.24

The Tomcat servlet container is a very popular application server utilized for development of Web applications. It provides a framework where Servlets can be installed. Servlets implement the basic HTTP request/response protocol provided by this framework, while the Tomcat container is responsible for the marshalling / unmarshalling of messages, multithreading, life cycle management, application location, HTTP and session management. The Tomcat version utilized in the GTDA is 6.0.24.

3.3.2 OpenId Server

This component allows an application to host its own OpenId server implementation. The GTDA implements its own OpenId server, thus providing a uniform mechanism for authentication. i.e. the client code performs the same authentication procedure whether it uses a third party id service or the GTDA service itself.

3.3.3 Shiro

Apache Shiro is a Java security framework for Web applications that provides authentication, authorization, cryptography, and session management. Shiro is used to support the implementation of the OpenId server API by the GTDA server, thus allowing GTDA to perform its own user authentication.

3.3.4 ESRI Geoportal Server v1.2.4

The ESRI Geoportal Server (<http://geoportal.sourceforge.net/>) is an open source implementation of the CSW (Catalog Service for the Web) standard. As such, it provides locations and descriptions of Web-based geospatial resources (typically GeoServer locations), and provides a Web UI (as shown in Figure 11) where these resources can be registered and searched for. In other words, it is an open source implementation of a directory service for geospatial data. Users are responsible for uploading META DATA to the portal so it can be found. Meta-data information include: the location, age, quality, and other characteristics of the resources.

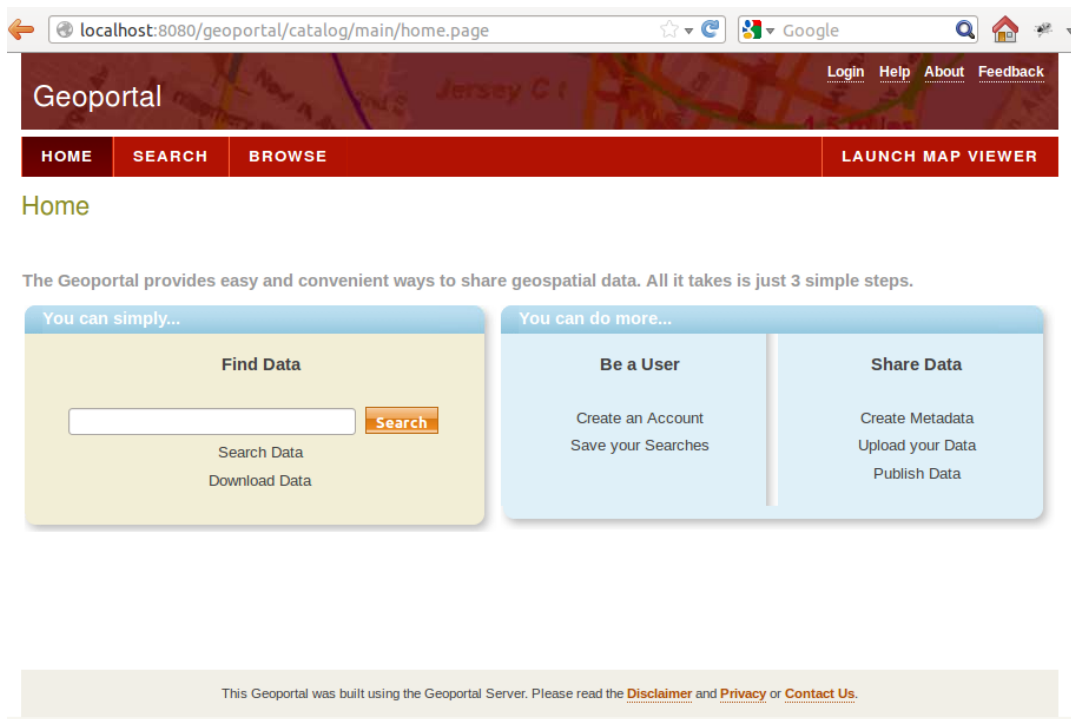


Figure 11 Geoportal main screen

In the context of the GTDA project, we reuse Geoportal's implementation of the OGC CSW 2.0.2 Catalog Web Service (http://www.geowebportal.org/dev_corner/serviceregistry/serviceregistry.jsp), in particular, its OGCCORE and ISO Application Profile components. These components provide XML schemas, style sheets, and request/response HTTP handlers that implement the CSW message format and protocols. By reusing these features GTDA becomes interoperable with CSW standard clients.

3.3.5 GeoServer v.2.4

GeoServer (<http://geoserver.org>) is a reference implementation of the Open Geospatial Consortium Web Feature Service (WFS). The WFS standard defines a Web Services API for the sharing of geographical features. In particular, GeoServer implements the OpenGIS Web Map Service (WMS - <http://www.opengeospatial.org/standards/wms/>) and Web Coverage Service (WCS - <http://www.opengeospatial.org/standards/wcs/>) specifications.

The WMS standard (WMS) specifies a simple HTTP-based protocol for requesting and transmitting geo-registered map images from one or more distributed geospatial databases. A WMS requests define the geographic layer(s) and area of interest to be processed. The response to the request is one or more geo-registered map images (returned as JPEG, PNG, etc) that can be displayed in a browser application. The interface also supports the ability to specify whether the returned images should be transparent so

that layers from multiple servers can be combined or not. In our GTDA, GeoServer is responsible for providing the layers utilized by the UI to display well information in its map.

The OGC Web Coverage Service (WCS) supports electronic retrieval of geospatial data as "coverages" – that is, digital geospatial information representing space/time-varying phenomena. GTDA does not utilize this feature at the moment.

In GTDA, GeoServer shares the same Postgres database with the GTDA main application. GeoServer is used to provide Layers: specific queries in a geographical database that provide shapes (typically dots or squares) in each row, so they can be displayed in GTDA map.

A screenshot of GeoServer, with a set of layers being published in GTDA is shown in Figure 12.

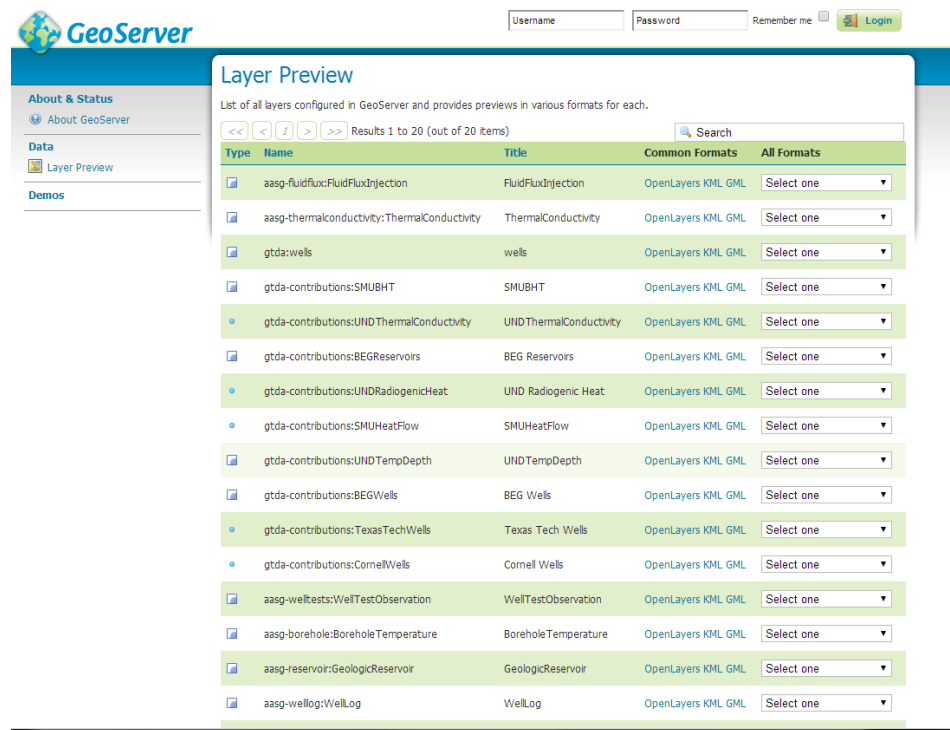


Figure 12 GeoServer layers database

3.3.6 Jetty v9.1

Jetty is the application server that serves the GeoServer web application. It serves the same purpose as Tomcat does for the GTDA application and Geoportal. It is hosted as a separate application in order to prevent the processor-intensive work that Geo-Server does from causing the other web applications from becoming unresponsive.

Apache Solr v4.6.0

Apache Solr is a platform for indexing and handling unstructured content-based search of documents. It provides a full RESTful XML/JSON interface for Apache Lucene indexing and searching APIs. An important feature of Solr 4.0 is its support for geospatial searches.

In GTDA, Solr is used to process queries about documents from wells in a certain geo-location (shape), as shown in Figure 8.

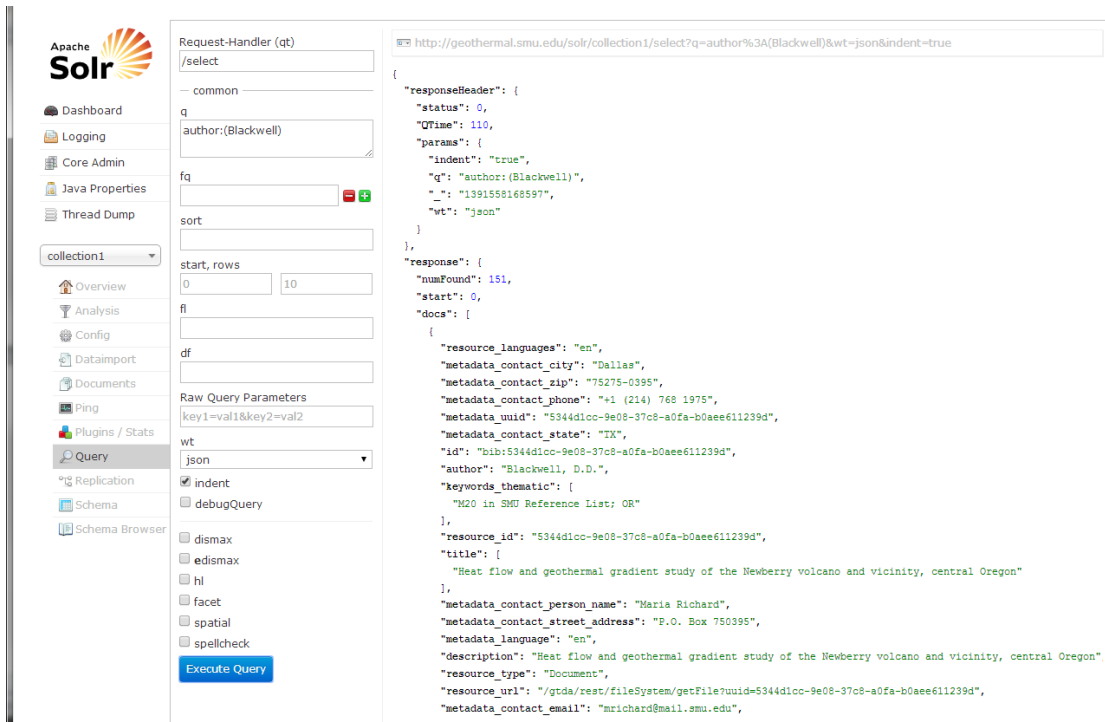


Figure 13 Solr repository browser

3.3.7 GTDA Server

The GTDA Server component provides a RESTful interface for the basic information utilized in the GTDA project.

The GTDA Server implementation is supported by the Spring Framework (3.1 and above). The following Spring framework modules are utilized: the inversion of control container, for loading and finding components; the model-view-controller for RESTful implementation of GTDA API; the batch processing for the importing and verification of different files into the system; the spring expression for parsing input files; the scripting JDBC for database communication; and the authentication module.

GTDA has 3 RESTful controllers, providing read-only methods: MapController, SiteDetailsController and ReportsController.

- **MapController** provides county information, i.e., the list of counties of a state in the USA. This information is utilized by the client to populate the states drop down menu in the UI.
- **SiteDetailsController** provides details about the wells as depth, precision, heat flow, site-related resources. It also provides access to precision logs about each well. That's the main interface to the system database.
- **ReportsController** provides reports about wells, publications, field notes, well logs, precision logs, and other queries, that are pre-determined types of reports that the system must provide. These are return as maps serialized to JSON or XML. It also provides data quality charts as a figure from different mime types.

Note that GTDA and GeoServer share the same PostgreSQL database, while GTDA provides individual well details including logs, the GeoServer provides basic well information, in particular, its position in the world map, and some meta-information.

3.4 Database

The GTDA database is supported by PostgreSQL v8.4 database (<http://www.postgresql.org/>). This open source database was chosen for its support for GIS extensions, supporting geo-located queries based on shapes. For example, it is possible to query for containment of geo-located points within geo-located regions (or polygons) in an efficient way.

In GTDA, the Postgres database is responsible for: Storing structured wells data and meta-data; providing database schemas to both GTDA and GeoServer. Postgres also provides temporary storage (staging area) for imported data from different clients. The GIS extensions of Postgres are used to convert (longitude, latitude) coordinates from customer's imported data—from the staging area—into well-formed data entries in the main database. These entries have shapes attached to each entry.

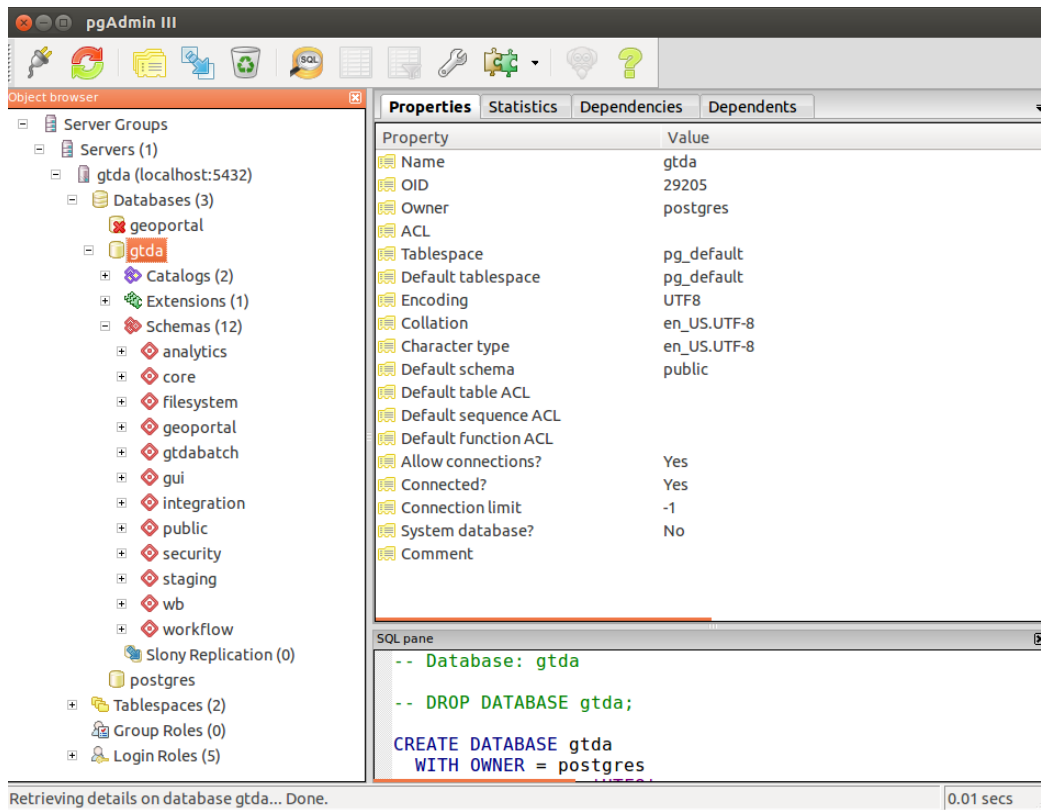


Figure 14 GTDA database

3.4.1 Entities

The following diagram depicts the entities in the GTDA project. In total there are 16 entities with only very limited relations between a few of the entities:

- CoreCounty
- CoreHeatFlow
- CoreReference
- CoreState
- County
- DataSource
- Depth
- OSTIXMLPublication
- PrecisionLog
- Publication
- QualityIssue
- SiteRelatedResourceId
- SiteRelatedResources
- SurfaceSite
- TemperatureSeries
- TempratureSeriesMapping

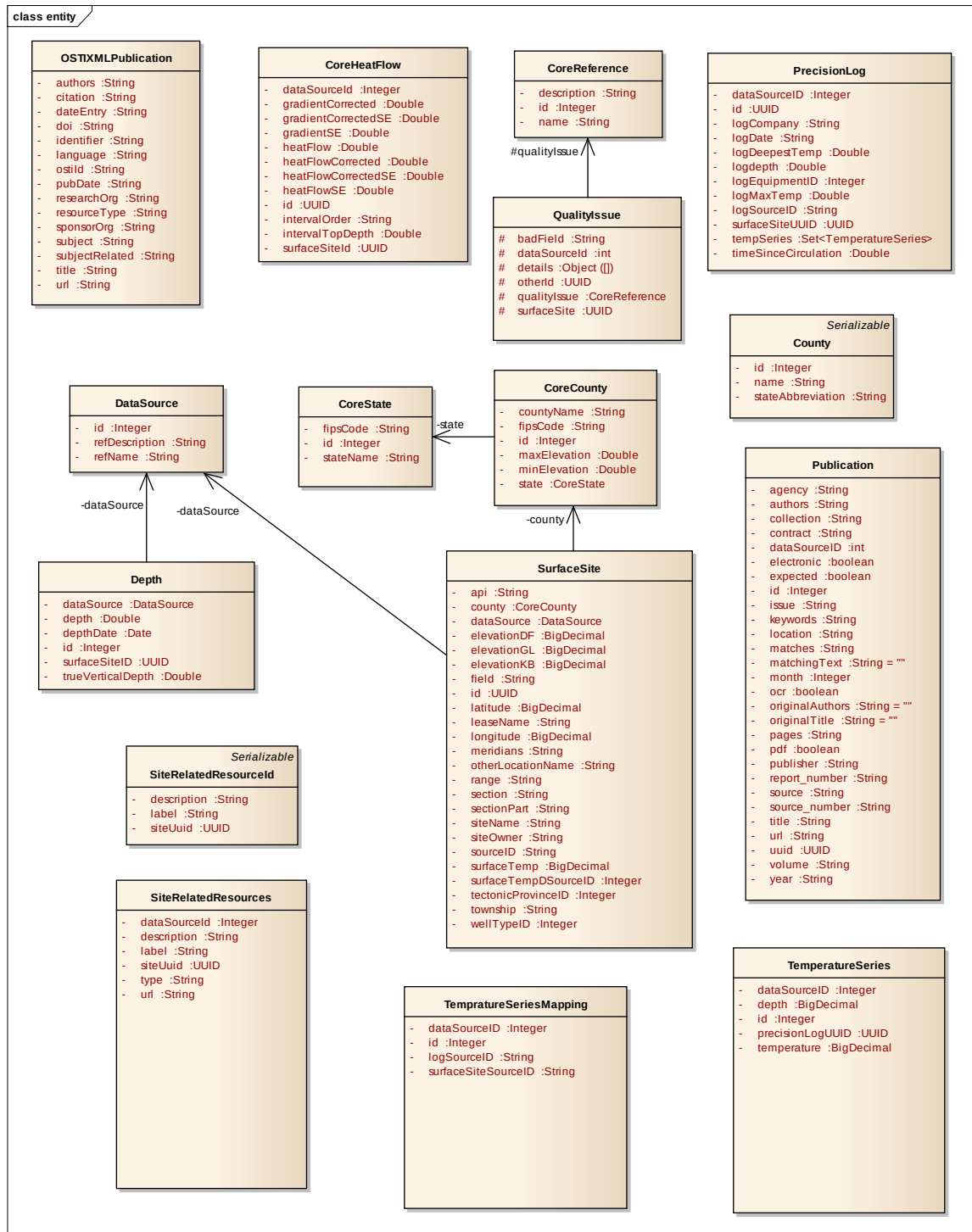


Figure 15 GTDA Entities

These DTO objects listed in Figure 15 mimic the entities and tables from the PostgreSQL database, and provide a convenient way to read and process data within the GTDA application.

3.4.2 GTDA database schemes

The database tables utilized in the GTDA application are serialized/de-serialized utilizing Spring Hibernate component, an extension of Spring Framework 3.1. A description of the DTO objects was discussed in the previous section.

3.5 Connectors

Different connectors are used to integrate the components of the system. The connection between the main application components and the database is provided by JDBC.

Spring is used internally to coordinate the communication between the RESTful controllers within Tomcat. The communication between clients and server is mediated by RESTful services over the HTTP protocol, and by standardized web-service protocols as WMS, WFS and CSW.

4 System Administration and Maintenance

4.1 Passwords

The system is protected by the following administrative passwords:

- GTDA
 - o Access: <http://geothermal.smu.edu/gtda/>
 - o Username: smu
 - o Password: *REMOVED For Final Report*
- GeoServer:
 - o Access: <http://geothermal.smu.edu:9000/geoserver/web/>
 - o Username: admin
 - o Password: *REMOVED For Final Report*
- Database (when being accessed via psql, pgAdmin, or another database exploring client):
 - o Username: postgres
 - o Password: *REMOVED For Final Report*
- GeoPortal
 - o Access: <http://geothermal.smu.edu/geoportal/>
 - o Username: admin
 - o Password: *REMOVED For Final Report*

4.2 Starting and Stopping the System

In order to perform certain administrative activities you may need to start or stop the system. To stop the GTDA, it is sufficient to shut down the Tomcat and Jetty servers. Log into sgtl1p1 and execute the following commands:

```
% sudo service tomcat6 stop
```

```
% sudo service jetty stop
```

Similarly to start it again, issue the following commands:

```
% sudo service tomcat6 start
```

```
% sudo service jetty start
```

The first command halts Tomcat, which stops the GTDA web application, as well as Geoportal and Solr. The second stops GeoServer.

Note that these actions do not start/stop the database. It is rare that you will need to stop the database server. To do so, log into sgtl1d1 and issue the following command:

```
% sudo service postgresql stop
```

Similarly, starting it again is accomplished with the following command:

```
% sudo service postgresql status
```

It is rare that either of the virtual machine need to be halted or restarted. This should only be done by, or with the supervision of OIT.

4.3 Backup and Restore

4.3.1 What to Backup

4.3.1.1 Data

In order to protect the system against data loss, we recommend backing up the data on a regular basis. More frequent backups are preferable, but note that depending on the type of backup you take, they may consume a great deal of storage space.

Data that should be backed up include

1. PostgreSQL database (instructions are given below)
2. Solr index at /data/solr
3. GeoServer data at /opt/geoserver-2.4.5/data_dir
4. Uploaded files folder at /data/files

Note that when backups are taken all parts should be backed up at the same time to maintain consistency with each other.

For example, the database contains the metadata for the documents that have already been indexed. This record should match the listing of documents in the /data/files and /data/gtda/ext directories. If they are inconsistent, although the system will start up and run normally, the differences between the file system details stored in the database and

the file system itself will cause odd runtime behavior when trying to retrieve files, and potentially cause the system to fail to rollback in cases where a data import fails to publish.

4.3.1.2 Application

In the rare event that the machine on which the SMU Node runs experiences a catastrophic failure which requires the system to be rebuilt, the following items should be backed up in order to restore the application:

- Tomcat configuration at /etc/gtda on sgtl1p1
- Scripts that are executed automatically on a regular basis
 - o /etc/cron.d/correct-privs-and-ownership.sh
 - o /etc/cron.d/geoserver_watchdog.sh
 - o /etc/cron.d/run-solr-indexer.sh
- Schedule for the aforementioned scripts at /etc/crontab
- Jetty configuration at /opt/jetty-distribution-9.1.3.v20140225 on sgtl1p1
- Static content of the web application at /usr/share/tomcat6/webapps/ROOT/static on sgtl1p1
- Database configuration at /var/lib/pgsql/data on sgtl1d1. In particular the following files should be backed up:
 - o pg_hba.conf
 - o postgresql.conf
 - o postgresql.conf.smu_node.dev
 - o postgresql.conf.smu_node.prodn

These items need not be backed up as frequently as the data, as they are not likely to change. Having at least one backup set to revert to is sufficient and important.

4.3.2 How to back up

To make sure no one is uploading documents or otherwise altering the data or configuration of the system's components while the system backup is being completed, a system shutdown is needed. To accomplish this task, it is sufficient to shut down the Tomcat and Jetty servers. See the section entitled "Starting and Stopping the System".

4.3.2.1 Backup options for files and folders

Files and folders can be backed up by any of the following means:

copying them to a separate disk (for information about how to copy files see the manual

page for the 'cp' command by logging into sgtl1p1 and typing the command 'man cp'.)

archiving them to a separate disk using the tar command (see the manual page for the

'tar' command by logging into sgtl1p1 and typing the command 'man tar'.)

using another dedicated archiving tool (most IT departments have such a tool and policies to ensure regular backups.) See the section entitled "Install Backup tool" for a recommendation.

4.3.2.2 Backing up the database

Backing up the database requires that the database be running. The following steps will produce a single file containing all the table, view, and function definitions as well as the data for all of the databases relevant to the GTDA's components. Log into sgtl1p1 (you can log into sgtl1d1, but it is not necessary.) Execute the command:

```
% sudo -u postgres pg_dump --format c --hostname sgtl1d1 \  
--username=postgres gtدا --file name_of_backup_file
```

where *name_of_backup_file* is a name you must provide. This command may take 20 minutes or more to complete, depending on the size of the database. The resulting file will exceed 1GB in size.

More information on pg_dump can be found at:

<http://www.postgresql.org/docs/8.4/static/backup-dump.html#BACKUP-DUMP-RESTORE>

4.3.3 How to restore data

Restoring data will require stopping the application servers while the data is being restored. See the section entitled "Starting and Stopping the System." In the case of files and folders, the backed-up items can be copied into their original locations from their backups using the corresponding commands for the way they were backed up (cp, tar, etc.)

For the database, log into sgtl1p1 and issue the command:

```
% sudo -u postgres pg_restore --hostname sgtl1d1 name_of_backup_file
```

where *name_of_backup_file* is the name of the backup file.

More information on pg_restore can be found under:

<http://www.postgresql.org/docs/8.4/static/backup-dump.html#BACKUP-DUMP-RESTORE>

Note that when you restore from a previous backup, any changes to the database and file system that were made between when the backup was taken and the current date will be lost. This includes (but is not limited to) changes to user passwords, the history of imports.

4.3.4 Commands to check the disk space

It is highly recommended to check the available disk space on the system. Linux provides a simple command to check for disk space:

```
$ df .
```

By executing this command Linux returns the available disk space as well as the used disk space to the user. It returns the information about the drive the user is currently in.

The following command provides information about the available disk space for all drives on the system:

```
$ df
```

By leaving out the point "." After the command df the system returns the information about available and used disk space for all drives mounted on this machine.

4.4 Regeneration of CSW Entries

The following HTTP POST commands can be used to generate the CSW entries. In order to execute these you will need a POSTing tool. You can use the Poster plugin for the Firefox web browser, or a similar plugin for the Chrome web browser. Alternately, you can POST from a command-line using a tool such as cUrl. See <http://curl.haxx.se> for details.

To generate CSW for all publications:

- <http://geothermal.smu.edu/gtda/rest/publication/exportCSW?timeStamp=1>

To generate CSW for data sets:

Data set	Export URL

SMU HF	http://geothermal.smu.edu/gtda/rest/file/exportCSW?uuid=4db6b79f-e92a-32c7-9622-89eb0fa421c8
SMU BHT	http://geothermal.smu.edu/gtda/rest/file/exportCSW?uuid=a3c17bda-5d65-35dd-b15c-4fb6d0a2a953
BEG ALL	http://geothermal.smu.edu/gtda/rest/file/exportCSW?uuid=cbafee9d-f4c0-3501-a46e-71e509d33f5c
BEG RES	http://geothermal.smu.edu/gtda/rest/file/exportCSW?uuid=0614324f-3726-32f4-bc3e-f800c6aa3d3d
CNL ALL	http://geothermal.smu.edu/gtda/rest/file/exportCSW?uuid=71c8563c-a98e-377f-a145-474353eec427
TXT ALL	http://geothermal.smu.edu/gtda/rest/file/exportCSW?uuid=4de8c501-889a-31ff-9347-e700127fdf49
UND	http://geothermal.smu.edu/gtda/rest/file/exportCSW?uuid=5d010959-597f-4f6e-a59f-3bf876b2396f

4.4.1 Trigger Indexing of Publication data

Re-Indexing of publications is triggered automatically at night (2am). However it is also possible to do this manually. In order to manually trigger the indexing of publication data execute the following commands:

- `/etc/cron.d/run-solr-indexer.sh`

This must be run with root privileges, i.e. by using `sudo`. It may take several hours to complete the indexing.

This script performs both the full-text indexing as well as the metadata indexing. This script writes its results to `/usr/share/tomcat6/logs/solr-indexer-TIMESTAMP.log` where `TIMESTAMP` is the date and time of when the execution began.

4.5 Location of Data and Components

In the following we outline important directories that contain key data and components of the system:

4.5.1 Tomcat

Tomcat is the Web Application Server we are using to serve the major components of our system (i.e. Geoportal, SOLR, GTDA software). It is a standard component for Linux operating systems. We are using Tomcat 6. The Tomcat home folder can be found here:

```
/usr/share/Tomcat6
```

The GTDA software component that is hosted by Tomcat6 can be found here:

```
/usr/share/Tomcat6/webapps/gtda
```

An important folder regarding Tomcat is the folder containing its log files. This folder can be found here:

```
/var/log/Tomcat6/
```

This folder contains logs for Tomcat itself (catalina.out) and the GTDA application (gtda-server.log) as well as for Geoportal (gpt-*<date>*.log.) In addition, the GeoServer log is available at:

```
/opt/geoserver-2.4.5/data_dir/logs/geoserver.log
```

Please refer to the [Tomcat6 documentation](#) for more details about Tomcat, and the [Jetty documentation](#) for more details about Jetty.

4.5.2 Postgres

PostgreSQL is software installed within the Linux Operating System. There is no installation and data folder that needs to be highlighted. However, the log files can be found on sgtl1d1, located at /var/lib/psql/data/pg_log.

PostgreSQL is configured to rotate log files weekly. This means that only seven log files are maintained, one for each day of the week. Each day that day's log file is overwritten.

In order to review the data in the database directly, the easiest access is available via the command-line tool psql, which can be run on sgtl1p1:

```
%psql -h sgtl1d1 -U postgres gtda
```

This will prompt you for the password, which is given below.

A more user-friendly way to browse the database is to use pgAdmin III. This is software that is available for Windows, Mac, and Linux: <http://www.pgadmin.org/index.php>.

After having pgAdmin installed on your local machine, connect to the database system. The parameters for connecting to the system are:

```
Server name: sgtl1d1
Port:       5432
Username:   postgres
Password:   r00tu$3r
Database:   gtda
```

Note that you may require some configuration changes to sgtl1p1 and sgtl1d1's iptables in order to allow users to connect to the database on sgtl1d1. Currently, only traffic from sgtl1p1 is allowed.

Please refer to the pgAdminIII documentation for details how to use it. PGAdmin is not a Web Application and therefore does not run in the browser. It is a desktop application available for Windows, Linux, and Mac OS.

Figure 16 is a Screenshot after having logged into the Admin tool (Windows Version):

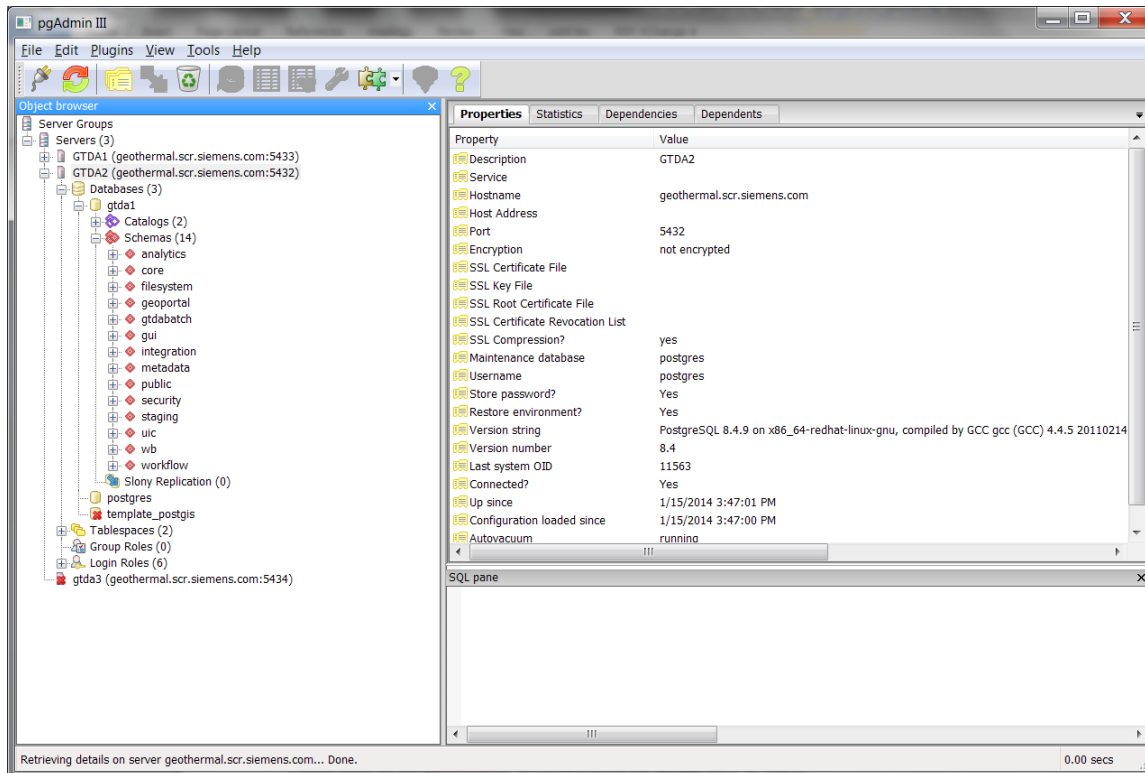


Figure 16: Screenshot of pgAdminIII

(Note that the screen shot here depicts the listing of databases on a test machine. On sgtl1d1 there is only one GTDA database, not three as depicted above.) In order to review the data have a look at the schemata defining the system. Most of the names of the schemata are self-explanatory. The essential data can be found under the Schema: 'core'. When opening the list of tables you will find a list of tables that contain the data imported using the data importing functionality of GTDA. As an example the following Screenshot shows the definition of the 'publication' that contains the core information about publications.

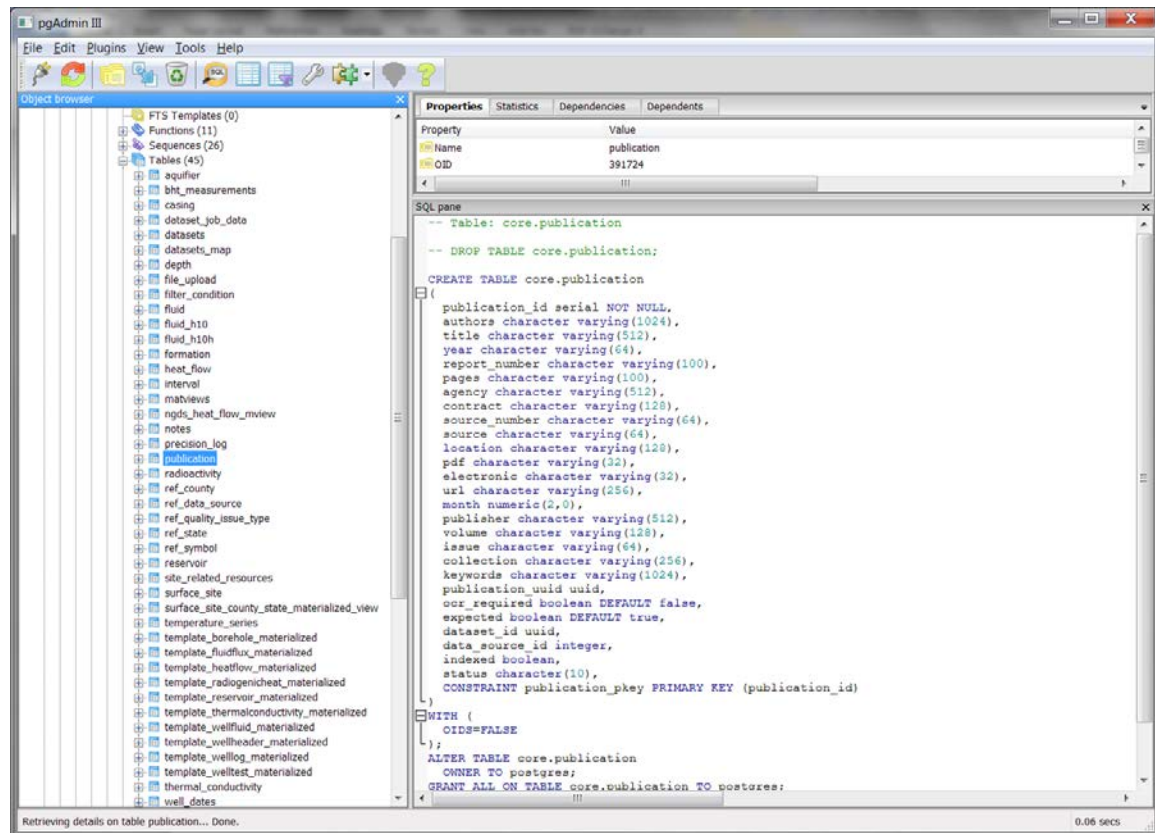


Figure 17: Screenshot of table 'publication'.

This table contains the allowable bibliographic information codes/size length for entries related to publications.

In order to view all entry in this table you can either use pgAdmin III or the psql command on the command line. To do this log into the server and execute the following command:

```
% \d+ core.publication
% select publication_id,title from core.publication limit 5;
```

Please also have a look at the views that have been created. As an example the view: publication_metadata contains all bibliographic information that is used for indexing. The following Screenshot shows the view:

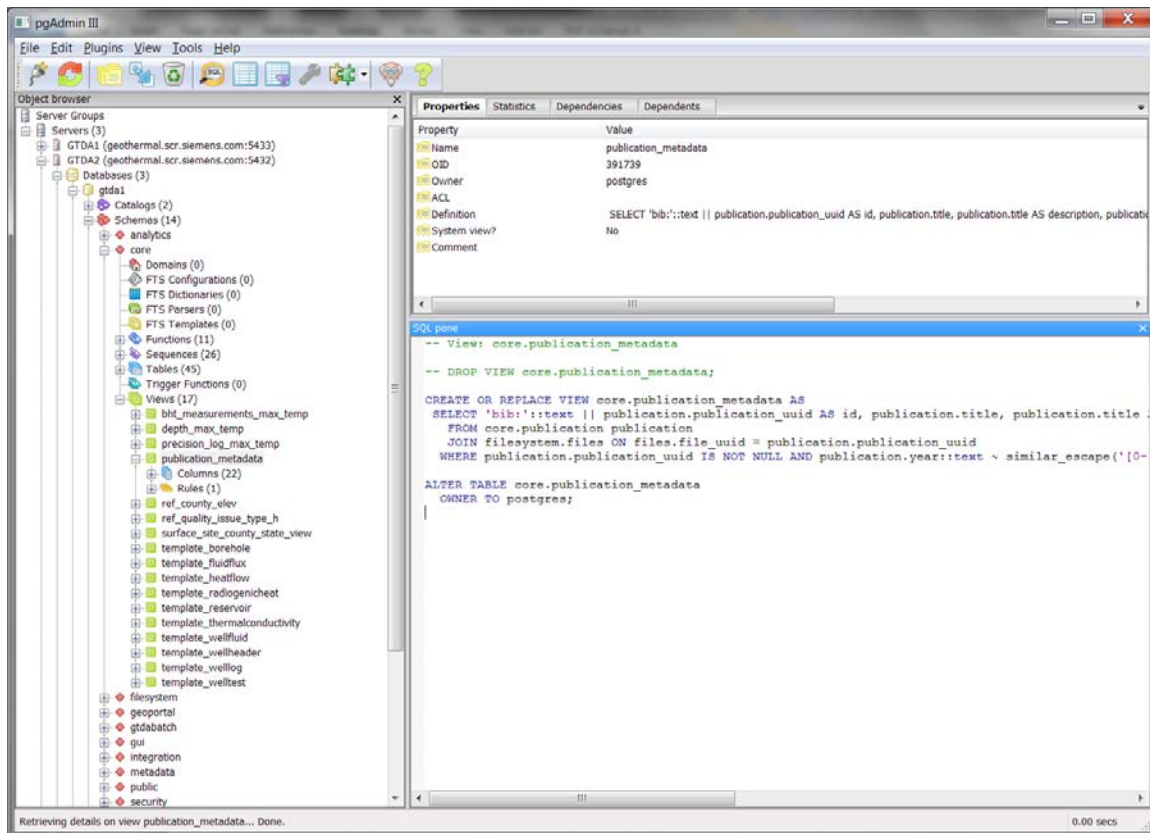


Figure 18: Screenshot of the view 'publication_metadata'.

4.5.3 SOLR

SOLR is the indexing machine that we are using to:

- Create a search index across the bibliographic data
- Full-text index of the uploaded documents

When the user executes a search request on the GTDA catalog page SOLR is the component that is answering that request. All SOLR information can be found in the folder:

/data/solr

The currently installed Solr version is 4.6.0. The configuration information for GTDA is defined in

/opt/solr/collection1

- Solr Index
 - o Location: sgt1p1:/data/solr/index
- Solr Configuration
 - o Location: sgt1p1:/opt/solr

4.6 GeoServer

GeoServer is a web application that is hosted by Tomcat. It effectively supports the open protocols WMS, WFS, and WCS that GTDA needs to provide. The GeoServer can be found in Tomcat's webapp directory:

```
/usr/share/tomcat6/webapps.geoserver
```

The log for geoserver can be found here:

```
/usr/share/tomcat6/webapps/geoserver/data/logs/geoserver.log
```

4.7 Geoportal

Geoportal is used to serve the table of contents of the GTDA node as CSW entries. Geoportal is also served by Tomcat. It can be found here:

```
/usr/share/tomcat6/webapps/geoportal
```

The CSW entries can be found in the following directory:

```
/usr/share/tomcat6/webapps/ROOT/csw-waf
```

The CSW entries are generated when documents and their bibliographies are uploaded into the system. Description how to create the CSW entries can be found in section 4.4.

4.8 GTDA Application

The GTDA software application is installed in Tomcat's webapp folder:

```
/usr/share/tomcat6/webapps/gtda
```

The details of the software components used have been outlined in section 4 of this document.

In the following we describe a methodology that can be used to find the input files for the UI that can be edited, e.g., in order to make small changes to the UI:

Portions of the documentation of GTDA have been externalized to be editable without modifying the code itself. These files reside entirely in `/usr/share/Tomcat6/webapps/ROOT/static` on `sgtl1p1`. Within this directory files can be uploaded using command-line tools such as SCP, or file transfer tools such as WinSCP, FileZilla, etc. Note that appropriate permissions to access this directory will be required, and are a matter for OIT to manage.

Files in `/usr/share/Tomcat6/webapps/ROOT/static` can be organized into subdirectories and managed as desired with the exception of those in

/usr/share/Tomcat6/webapps/ROOT/static/gtda. These files are static content used by the application.

Log files can be viewed on sgtl1p1, or downloaded to your personal computer for viewing using any text editor. Note that the file sizes tend to be large. To view files on sgtl1p1 use the `less` utility:

```
% sudo less log_file_name
```

You can also use the `monitor` script on sgtl1p1 at /home/philipp/bin/monitor. `monitor` calls the `less` command without requiring you to provide a path to the log file. Rather, you provide a nickname, e.g.

```
% sudo monitor gtda
```

is equivalent to

```
% less /usr/share/tomcat6/logs/gtda-server.log
```

An oft overlooked feature of `less` (also useful when using `monitor`) is that if you type `SHIFT-f` it will “`tail -f`” the log file, i.e. continuously refresh the file as more data is writing to it. With this feature you can observe log data being written in real time as GTDA, GeoServer, etc. are running and writing to the log. To stop following the file, type `CTRL-c` and you will be able once again to peruse the file normally. See the ‘`man less`’ for details.

GTDA log info can be found in the Tomcat logs:

- GTDA log file
 - Name: `gtda-server.log`
 - Location: `sgtl1p1:/var/log/tomcat6/gtda-server.log`
- Tomcat log file
 - Name: `catalina.log`
 - Location: `sgtl1p1:/var/log/tomcat6/logs/catalina.log`

4.9 Access to Gray Box about help documentation

The gray box about help documentation has been externalized and can be modified. The content for this box can be found on the following location:

/usr/share/tomcat6/webapps/ROOT/static/ManualGTDA.htm

4.10 Documents

All documents that have been uploaded (publications, data sets, related resources, etc.) are stored on sgtl1p1's file system. The following describes these locations and their contents:

- **/data/gtda:** "Legacy" documents (data sets and publications) that were loaded into the system prior to the existence of the GTDA's import functionality.
 - **/data/gtda/ext/Incoming:** Contains publications organized into folders by data provider. In the case of SMU, it contains field notes, temperature curves, and scanned library documents. Data sets appearing here are likely subsumed by those appearing in /data/files
 - **/data/gtda/ext/load:** Contains publications, field notes, well logs, and data sets organized by data provider.
- **/data/files:** Contains publications, data sets, and related resources organized by data provider. These have all been uploaded via the GTDA GUI. All newly imported data are written to folders within this directory. Any publications or data sets downloaded in their original form are pulled from within this directory.

4.11 OS Maintenance, Backup and Restore

4.11.1 OS Maintenance

The operating systems of both servers should be updated at a predetermined interval, preferable monthly, but quarterly would also be sufficient.

Note: at a certain point it can be expected that the system is not compatible with operating system updates,, e.g. because an update of a key component will be in-compatible with the implementation of GTDA.

This cannot be avoided.

Therefore it is important to first make a backup of the system before uploading it (see section below)

4.11.1.1 OS Updates

Before updating the operating system, make sure that you have a backup for example by using the Snapshot mechanism of Virtual Box outlined below.

CentOS 5 updates can be applied by root using the command:

```
% yum update
```

This is also documented in the Centos documentation:

<http://www.centos.org/docs/4/html/yum/>

4.11.1.2 OS Backups

In order to prevent that the system becomes unusable we highly recommend creating a complete system backup first.

Since the two machines run in virtual machines a straight-forward option is to either copy the complete machine or alternatively create a snapshot. This can be done with the UI of Oracle Virtual Box:

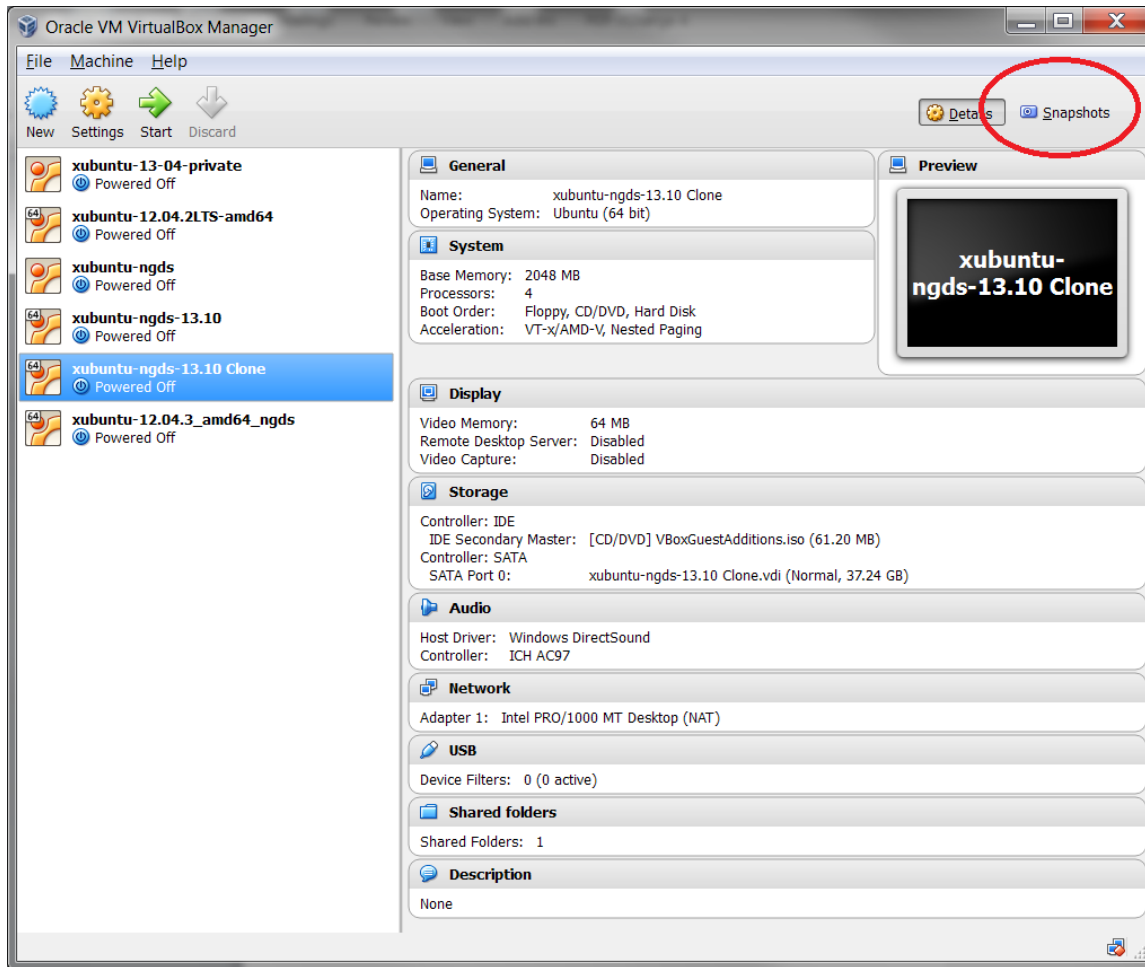


Figure 19: Snapshot of Oracle VirtualBox User Interface highlighting the Snapshot Feature.

Click on the Snapshot button in the upper right corner. This will open a new Window allowing you take a Snapshot of the system or alternatively clone the complete system. We recommend to at least once per year create a complete clone of the system and archive it on an external hard drive.

4.11.1.3 OS Restores

The UI can also be used to restore a snapshot of the system. After clicking on the Snapshot button described in the previous section the user can see previously taken snapshots. In order to restore a previous snapshot the user has to right-click on the chosen previous snapshot and then click on “Restore Snapshot”.

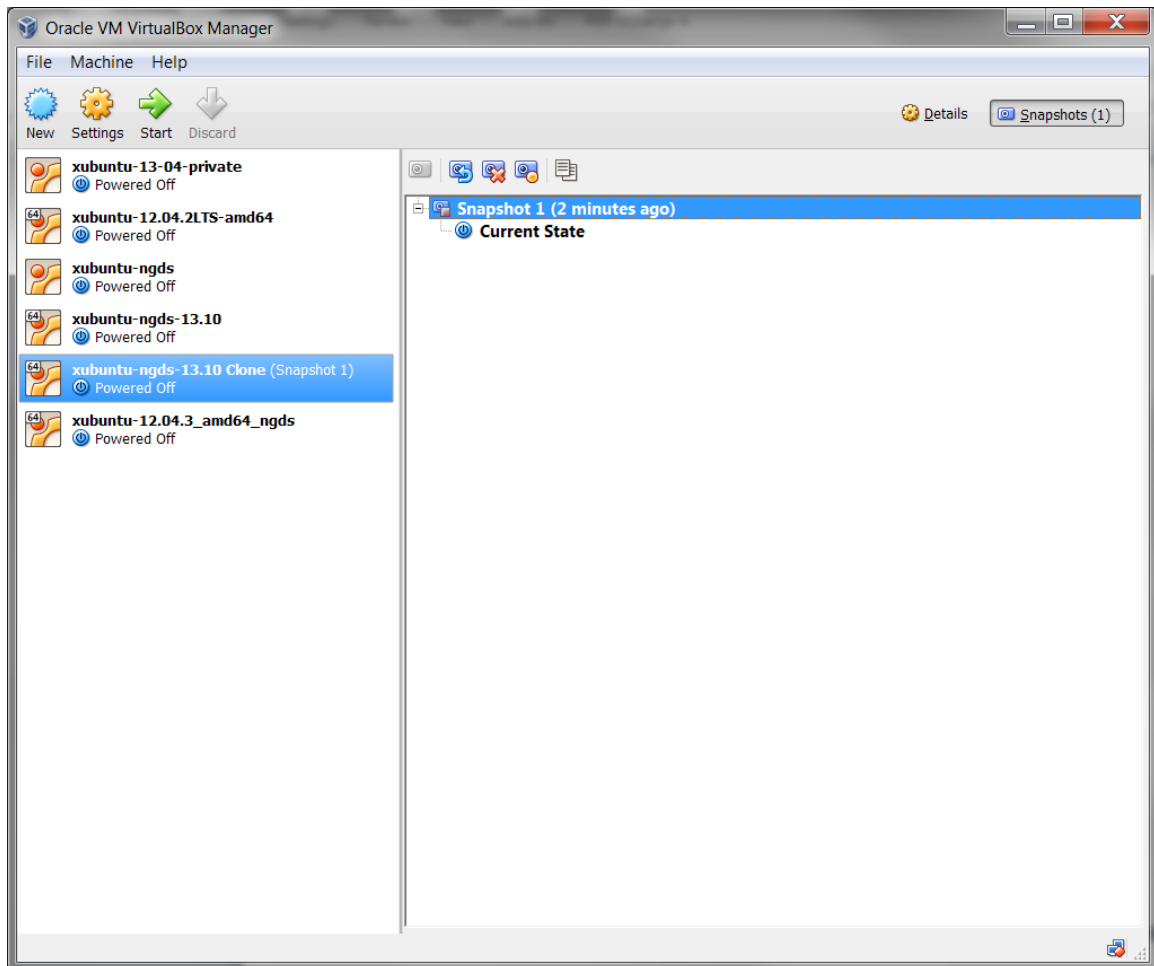


Figure 20: Oracle VirtualBox User Interface showing the snapshot management interface. right-click on a Snapshot to restore it.

More information on Snapshots and Cloning can be found on the Oracle Virtual-Box Home Page:

<https://www.virtualbox.org/manual/ch01.html>

4.11.1.4 Alternatives

4.11.1.4.1 Activate Rollback Feature of Yum.

In addition to Snapshots the system administrator can also enable Yum rollbacks. We do not recommend to rely on this feature to protect the system against incompatible back-ups.

To enable rollback, simply edit the Yum config file and add the following setting :

```
% vi /etc/yum.conf
```

And add :

```
tsflags=repackage
```

Now the features is enabled and will be effective for every new update you install. You'll notice a new action named "Repackage" before the usual "Updating" stage is shown.

Add the following line into the rpm macro file (create if non-existent) :

```
% vi /etc/rpm/macros
```

And add :

```
%_repackage_all_erasures 1
```

Now you can use the following command to go back in time:

```
rpm -Uvh --rollback '21:00'  
rpm -Uvh --rollback '3 hours ago'  
rpm -Uvh --rollback 'august 13'  
rpm -Uvh --rollback 'yesterday'
```

As mentioned above: We **do NOT recommend** relying on this mechanism and rather take Snapshots and Clones using VirtualBox.

4.11.1.4.2 Install Backup tool

Finally a solution is to use a backup tool to make complete backups of the system. One tool that is popular and considered very reliable is AMANDA. A detailed description of AMANDA can be found here:

<http://www.amanda.org/>

Key is to use an external hard drive that is either connected via USB or a network storage for the backups.

5 Appendix 1: Terms and Glossary

In the following we list a set of important terms and definitions.

5.1 Acronyms and Abbreviations

The following table lists the abbreviations used in this document, in order to promote their unique and unambiguous usage throughout the document and the Project.

Table 1: Abbreviations and Definitions (list in alphabetical order)

Abbreviations	Definition
API	Application Programming Interface (Not the American Petroleum Institute)
CSV	Comma-Separated Variable file format
CSW	Catalogue Service for the Web
DB	Database
DIS	Data Import Schema
DOE	Department of Energy
DOM	Document Object Model
GTDA	Geothermal Data Archive
JDBC	Java Database Connectivity
JSON	JavaScript Object Notation
MVC	Model View Controller
NGDS	National Geothermal Data System
NetCDF	Network Common Data Form
OAI-PMH	Open Archives Initiative – Protocol for Metadata Harvesting
OGC	Open Geospatial Consortium
ORM	Object-Relational Mapping
SDD	Software Design Description
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
UTM	Universal Transverse Mercator coordinate system
WCS	Web Coverage Service
WFS	Web Feature Service
WMS	Web Map Service
WSS	Web Services Specification

DTO	Data Transfer Object

5.2 Glossary of Geological Relate Terms

The following table lists the terms used in this document, in order to promote their unique and unambiguous usage throughout the document and the Project.

Table 2: Terms used in the GTDA System

Terms	Definition
Data	Documents, Computer database, pictures, spreadsheets about geological features
Dataset	A group of geological data, typically related to one geological feature.
Geological feature	Any type of geological phenomena such as: • Faults • Fluid inclusions • Fluid rock interaction [Use for non-water fluids only, usually carbon dioxide] • Hydrothermal alteration [Same as rock-water interaction] • Fractures • Magmatism [Includes magma and magmatic processes; use more specific term "Volcanism" if applicable, although some articles address both; see also "Magma energy"] • Volcanism [Includes volcanoes and volcanic processes] • Permeability [Do not overuse; ability of fluid to flow through rock] • Pore pressure [Includes capillary pressure] • Porosity [Do not overuse; amount of pore space within rock] • Rock mechanics [Response of rocks to external forces, especially related to stress] • Seismic attenuation [Ability of rocks to absorb seismic waves] • Seismicity [Same as earthquakes; see also "Induced seismicity"] • Microseismicity [Same as microearthquakes] • Seismic velocity • Stress fields • Borehole breakouts [Bulging of wells, related to stress fields] • Tectonics [On a regional scale] • Any rock formations treated in detail - o Breccia

	- o Granite - o Greywacke [also spelled graywacke] - o Sedimentary basins
Geological survey	Gathering data using geological methods, or descriptions of regional geology; includes structure
Geophysical survey	Different types of descriptions of physical characteristics of regional characteristics including electromagnetic surveys, gravity surveys, magnetic, seismic, shallow temperature surveys, etc.
Heat flow	Energy coming to the surface in a given area, often measured in milli-watts/m ² and used in resource assessment; derived by multiplying temperature gradients and thermal conductivity; essentially equivalent to heat flux, which is an engineering term
Landmark	Locations and regions in the globe geo-referenced to a specific value: city, district, state, county, known geothermal sites, etc. A land mark is implemented as a dictionary, a map from names into geographical coordinates.
Metadata	Summary of important characteristics of a piece of data used for the purpose of searching and discovery by users of a geological feature database.
Metadata template	A predetermined set of metadata attributes used to describe data about geological features, e.g., document meta-data include a document bibliographical citation, its geo-location, ownership, and URI (for the actual document)
Resources	A generic name to include data, metadata and other artifacts that are under the responsibility of a user.
Temperature gradients	Change in temperature with depth, e.g., °C/m, °C/km; includes temperature depth profiles
Thermal conductivity	Ability to conduct heat; e.g., salt domes have high conductivity, while sedimentary basins have low conductivity
Well logging	Well measurements and descriptions, including borehole geology

6 Appendix 2: Database Schema

The following tables are extracted from the PostgreSQL database on sgtl1d1 using the following commands:

1. Having already logged into sgtl1p1, connect to the database using the psql tool:

```
psql -h sgtl1d1 -U postgres gtda
```
2. At the psql prompt indicate the file into which you wish to write the table details:

```
\o 'outputfile-name-in-single-quotes'
```

Alternatively, one can also execute the following command at the command prompt of sgtl1p1:

```
% pg_dump -h sgtl1d1 -U Postgres gtda
```

The output lists tables and views for the core and staging schemas. For each table all of the columns that comprise the table, as well as their types and any modifiers that have been defined for them. For each view it additionally indicates the definition of the view, which is an SQL query upon other tables that populates the view.

Table "staging.beg_reservoir_data"

Column	Type	Modifiers
reservoiruri	character varying(255)	
reservoirname	character varying(255)	
label	character varying(255)	
otherid	character varying(255)	
othername	character varying(255)	
description	character varying(255)	
county	character varying(255)	
state	character varying(255)	
locationname	character varying(255)	
latdegree	double precision	
longdegree	double precision	

srs	character varying(255)	
locationuncertaintystatement	character varying(255)	
temperaturemax_f	double precision	
temperaturemin_f	double precision	
depthtotop_ft	double precision	
depthtobottom_ft	double precision	
dominantformationname	character varying(255)	
otherformationname	character varying(255)	
dominantlithologyname	character varying(255)	
otherlithologyname	character varying(255)	
area_acres	double precision	
width_ft	double precision	
length_ft	double precision	
sandvolume_cft	double precision	
blockvolume_cft	double precision	
percentsand	double precision	
averagenetsandthickness_ft	double precision	
wellcount	integer	
faultcount	integer	
faulted	character varying(255)	
faulttype	character varying(255)	
faulthydrauliccharacter	character varying(255)	
firstproductiondate	date	
lastproductiondate	date	
yearsinproduction	integer	
monthlyproduction_bbl	double precision	
productionreportdate	date	
averagemonthlywaterproduction_bbl	double precision	

minimummonthlywaterproduction_bbl	double precision	
maximummonthlywaterproduction_bbl	double precision	
waterinjection_bbl	double precision	
watersaturation_pct	double precision	
watertds_mgl	double precision	
averagefieldwidepressure_psi	double precision	
averagefieldwidegradient_degfft	double precision	
porositymax_pct	double precision	
porositymin_pct	double precision	
permeabilitymax_md	double precision	
permeabilitymin_md	double precision	
cementationmineralogy	character varying(255)	
trap	character varying(255)	
drive	character varying(255)	
gravity	double precision	
commodityofinterest	character varying(255)	
stimulation	character varying(255)	
averageperforatedthickness_ft	double precision	
relatedresource	character varying(255)	
source	character varying(1023)	
metadataauri	character varying(255)	
majorfields	character varying(511)	
cordinate_nw_long	double precision	
cordinate_nw_lat	double precision	
cordinate_ne_long	double precision	
cordinate_ne_lat	double precision	
cordinate_sw_long	double precision	
cordinate_sw_lat	double precision	

ordinate_se_long	double precision	
ordinate_se_lat	double precision	

View "staging.beg_reservoir_view"

Column	Type	Modifiers
reservoiruri	character varying(255)	
reservoirname	character varying(255)	
label	character varying(255)	
otherid	character varying(255)	
othername	character varying(255)	
description	character varying(255)	
county	character varying(255)	
state	character varying(255)	
locationname	character varying(255)	
latdegree	double precision	
longdegree	double precision	
srs	character varying(255)	
locationuncertaintystatement	character varying(255)	
temperaturemax_f	double precision	
temperaturemin_f	double precision	
depthtotop_ft	double precision	
depthtobottom_ft	double precision	
dominantformationname	character varying(255)	
otherformationname	character varying(255)	
dominantlithologyname	character varying(255)	
otherlithologyname	character varying(255)	
area_acres	double precision	

width_ft	double precision	
length_ft	double precision	
sandvolume_cft	double precision	
blockvolume_cft	double precision	
percentsand	double precision	
averagenetsandthickness_ft	double precision	
wellcount	integer	
faultcount	integer	
faulted	character varying(255)	
faulttype	character varying(255)	
faulthydrauliccharacter	character varying(255)	
firstproductiondate	date	
lastproductiondate	date	
yearsinproduction	integer	
monthlyproduction_bbl	double precision	
productionreportdate	date	
averagemonthlywaterproduction_bbl	double precision	
minimummonthlywaterproduction_bbl	double precision	
maximummonthlywaterproduction_bbl	double precision	
waterinjection_bbl	double precision	
watersaturation_pct	double precision	
watertds_mgl	double precision	
averagefieldwidepressure_psi	double precision	
averagefieldwidegradient_degfft	double precision	
porositymax_pct	double precision	
porositymin_pct	double precision	
permeabilitymax_md	double precision	
permeabilitymin_md	double precision	

cementationmineralogy	character varying(255)	
trap	character varying(255)	
drive	character varying(255)	
gravity	double precision	
commodityofinterest	character varying(255)	
stimulation	character varying(255)	
averageperforatedthickness_ft	double precision	
relatedresource	character varying(255)	
source	character varying(1023)	
metadataauri	character varying(255)	
majorfields	character varying(511)	
cordinate_nw_long	double precision	
cordinate_nw_lat	double precision	
cordinate_ne_long	double precision	
cordinate_ne_lat	double precision	
cordinate_sw_long	double precision	
cordinate_sw_lat	double precision	
cordinate_se_long	double precision	
cordinate_se_lat	double precision	

View definition:

```
SELECT beg_reservoir_data.reservoiruri, beg_reservoir_data.reservoirname, beg_reservoir_data.label, beg_reservoir_data.otherid, beg_reservoir_data.othername, beg_reservoir_data.description, beg_reservoir_data.county, beg_reservoir_data.state, beg_reservoir_data.locationname, beg_reservoir_data.latdegree, beg_reservoir_data.longdegree, beg_reservoir_data.srs, beg_reservoir_data.locationuncertaintystatement, beg_reservoir_data.temperaturemax_f, beg_reservoir_data.temperaturemin_f, beg_reservoir_data.depthtotop_ft, beg_reservoir_data.depthtobottom_ft, beg_reservoir_data.dominantformationname, beg_reservoir_data.otherformationname, beg_reservoir_data.dominantlithologyname, beg_reservoir_data.otherlithologyname, beg_reservoir_data.area_acres, beg_reservoir_data.width_ft, beg_reservoir_data.length_ft, beg_reservoir_data.sandvolume_cft, beg_reservoir_data.blockvolume_cft, beg_reser-
```

```
voir_data.percentsand, beg_reservoir_data.averagenetsandthickness_ft, beg_reser-  
voir_data.wellcount, beg_reservoir_data.faultcount, beg_reservoir_data.faulted, beg_re-  
servoir_data.faulttype, beg_reservoir_data.faultyhydrauliccharacter, beg_reser-  
voir_data.firstproductiondate, beg_reservoir_data.lastproductiondate,  
beg_reservoir_data.yearsinproduction, beg_reservoir_data.monthlyproduction_bbl, beg_re-  
servoir_data.productionreportdate, beg_reservoir_data.averagemonthlywaterproduction_bbl,  
beg_reservoir_data.minimummonthlywaterproduction_bbl, beg_reservoir_data.maximummonthly-  
waterproduction_bbl, beg_reservoir_data.waterinjection_bbl, beg_reservoir_data.watersatu-  
ration_pct, beg_reservoir_data.watertds_mgl, beg_reservoir_data.averagefieldwidepres-  
sure_psi, beg_reservoir_data.averagefieldwidegradient_degfft,  
beg_reservoir_data.porositymax_pct, beg_reservoir_data.porositymin_pct, beg_reser-  
voir_data.permeabilitymax_md, beg_reservoir_data.permeabilitymin_md, beg_reser-  
voir_data.cementationmineralogy, beg_reservoir_data.trap, beg_reservoir_data.drive,  
beg_reservoir_data.gravity, beg_reservoir_data.commodityofinterest, beg_reser-  
voir_data.stimulation, beg_reservoir_data.averageperforatedthickness_ft, beg_reser-  
voir_data.relatedresource, beg_reservoir_data.source, beg_reservoir_data.metadataauri,  
beg_reservoir_data.majorfields, beg_reservoir_data.cordinate_nw_long, beg_reser-  
voir_data.cordinate_nw_lat, beg_reservoir_data.cordinate_ne_long, beg_reservoir_data.cor-  
dinate_ne_lat, beg_reservoir_data.cordinate_sw_long, beg_reservoir_data.cordinate_sw_lat,  
beg_reservoir_data.cordinate_se_long, beg_reservoir_data.cordinate_se_lat
```

```
FROM staging.beg_reservoir_data;
```


Appendix E: Sustainability Plan



Sustainability Plan for the SMU Node of the National Geothermal Data System

Supported by U.S. Department of Energy Grant: DE-EE0002852

Project Principal Investigator:

David D. Blackwell, SMU Geothermal Laboratory

Document Authors:

Cathy Chickering, SMU Geothermal Laboratory

Allen Hughes, SMU Office of Information Technology

Don Kline, SMU Office of Information Technology

Rajanikanth Tanikella, Siemens Corporate Research

February 10, 2014

Version 2.1

Abstract: Southern Methodist University is one of several institutions sponsored by the Department of Energy to serve geothermal research and information to researchers, private industry, and the public. These interconnected facilities will form the National Geothermal Data System (NGDS). The 'SMU Node' of the NGDS consists of a database application written by Siemens Corporate Research (SCR), a division of Siemens Corporation, running on servers located at the SMU data center(s), connected to the internet. Upon completion of the project in 2014, the SMU Node of the NGDS needs to be self-sustaining, as no additional federal funding is anticipated for the annual maintenance and upkeep of the data, the application, the servers or the network connectivity. The purpose of this document is to codify the sustainability plans, recognizing that there will be limits to some aspects of support provided indefinitely.

Table of Contents

Introduction.....	3
Geothermal data submitted by the 7 research institutions	4
Software Developed by Siemens for the Project	7
Server Environment (system software and hardware)	8
Security	8
Physical Security.....	8
System Security	8
Backups.....	9
Hardware and Software Routine Maintenance	9
Disaster Recovery Plans	9
Call Center and Customer Support Functions.....	10
System Administration (Non-routine)	10
Help-Desk	10
Network Connectivity.....	10
Support Agreement	11
Data Residing on Non-SMU Servers	12
Southern Methodist University Computing and Communications Policy	13

Sustainability Plan for the SMU Node of the National Geothermal Data System

Introduction

Southern Methodist University is one of several institutions sponsored by the Department of Energy to serve geothermal research and information to researchers, private industry, and the public. These interconnected facilities will form the National Geothermal Data System (NGDS). The 'SMU Node' of the NGDS consists of a database application written by Siemens Corporate Technology (SCT), a division of Siemens Corporation, running on servers located at the SMU data center(s), connected to the internet. Upon completion of the project in 2014, the SMU Node of the NGDS needs to be self-sustaining, as no additional federal funding is anticipated for the annual maintenance and upkeep of the data, the application, the servers or the network connectivity. The purpose of this document is to codify the sustainability plans, recognizing that there will be limits to some aspects of support provided indefinitely.

The project team behind the SMU Node consists of personnel from:

- SMU Geothermal Laboratory
- Siemens Corporate Technology, a division of Siemens Corporation
- Bureau of Economic Geology, University of Texas at Austin
- Cornell Energy Institute, Cornell University
- Geothermal Resources Council
- MLKay Technologies
- Texas Tech University
- University of North Dakota

With the exception of Siemens, each of the organizations provided geothermal information covering decades of research. The information varies in type (library type publications, databases of heat flow or temperature data, images of well logs, etc.). In most instances, the information provided resides on servers located at SMU's data center, however there is the option for the data providers to retain the information on their own servers, and to provide access to that information with 'pointers' from the SMU Node. This sustainability plan focuses on data residing on the SMU Node servers.

Siemens provided data integration, software development, and data analytics services to the project team. The 'software' consists of:

- A database populated with the geothermal information provided by the 7 other project team members.

- An interactive interface for the 7 data providers to access the database¹, upload data, review the submitted data, and publish the submitted data.
- A Catalog Service for the Web (CSW) capability to share the information with other nodes on the NGDS.
- A Web Feature Service (WFS) and WMS (Web Map Service) capability to enable end user applications² to pull information from the database.
- Data analysis tools capable of detecting outliers and patterns, to highlight records that may need additional review or present opportunities for new study.

One aspect of the data integration services provided by Siemens as part of this project encompasses converting information from the original format into the format needed for the database and to support CSW and WFS interoperability with other NGDS nodes.

In addition to providing geothermal data, SMU is providing overall project management, the physical servers containing the data and the application, and the network access to the servers.

The remainder of this document discusses sustainability plans for:

- The geothermal data submitted by the 7 research institutions,
- The software developed by Siemens for the project,
- The servers, which consist of system software and hardware
- The network connectivity

Geothermal data submitted by the 7 research institutions

A primary motivation for the development of the NGDS was to preserve legacy geothermal data for continued and future research. Much of the information was previously in paper documents on a shelf in a lab, inaccessible to anyone outside of that physical facility. While the project has placed the information into a widely accessible electronic format, it needs to accommodate changes to the data, as new research progresses. The software design specifications include the capability for the 7 data providers to edit and/or add new data to the system, as they desire, over time.

¹ The SMU scope of work did not include developing an interface for use by the general public, industry, etc. That work effort was within the scope of a different sponsored institution. However, the interface developed for the 7 data providers on the SMU Project Team is available to other institutions, private industry, and the public for purposes of access the information on the database.

² Applications such as NREL's Geothermal Prospector, and others that may be developed in the future.

SMU, BEG, Cornell, GRC, Texas Tech, and UND each anticipate additional data, publication, or related resources submissions in the future. Individual upload routines for each institution and each type of data submission were written to accommodate continued uploads using the same format and layout as previously submitted, even after the project conclusion.

The system design of the SMU Node anticipates the inevitable turnover at submitting organizations. If the data submitter has had a change in the person responsible for updates, the new contact may not have ready access to a copy of the prior submission; the system now allows them to retrieve a copy of the last successful submission to refer to in formatting their update and/or reviewing values. Additionally, A number of ‘upload forms’ were created to assist the data submitters in providing their data in the prescribed format and screening for common formatting errors.

Data Submission Type	File Layout
SMU Heat Flow Database of Equilibrium Log Data and Geothermal Wells (note: includes sites from the Global Heat Flow Database from the University of North Dakota) The dataset contains nearly 12,000 heat flow values from throughout the U.S. collected from 12,604 locations and over 14,768 rows of data. Site Heat Flow values are based on temperatures at multiple depth intervals. Some values are from literature sources whereas others are based on field logged equilibrium temperature measurements. Links to thousands of temperature depth curves and associated field notes are available through either interactive use of the SMU NGDS Node or the Well Log Content Model.	About the Dataset.PDF XML Schema (DescribeFeatureType) This upload contains 3 files and thus has three upload forms which are delivered in a single zip file: Upload Form SMU HF Point Upload Form SMU HF Reference Upload Form SMU HF Heat Flow
SMU Heat Flow Database from BHT Data Used to update the Geothermal Map of North America for Google.org, the dataset of over 31,000 records contains ‘interval’ and ‘site’ heat flow calculations based on ‘corrected’ bottom hole temperature data from the oil and gas industry.	About the Dataset.PDF XML Schema (DescribeFeatureType) Upload Form SMU BHT
BEG at University of Texas Consolidated Well Database Contains raw and corrected BHT, depth, location, fluid production, pressure, etc. from ~40,000 wells throughout Texas. URLs to BEG’s extensive core log library are expected to be accessible as a related resource through the SMU Node after further integration.	About the Dataset.PDF XML Schema (DescribeFeatureType) Upload Form BEG Wells
BEG at University of Texas Reservoir Analysis From 55 different Texas geologic reservoirs, most with multiple depth ‘slices’. The analysis contains net sand volume and thickness calculations, which can be used by	About the Dataset.PDF XML Schema (DescribeFeatureType)



geothermal reservoir engineers. The submission file closely matches the content model layout for Geologic Reservoir.	Upload Form BEG Reservoir
Cornell University Heat Flow Database (NY and PA) The analysis includes over 8,900 BHT and depth points from throughout Pennsylvania and New York, including predicted temperatures at various depth and the estimated depths to various temperature levels for several thousand locations. Links to regional maps developed by Cornell will be accessible as related information through interactive use of the SMU Node of the NGDS.	About the Dataset.PDF XML Schema (DescribeFeatureType) Upload Form Cornell
Geothermal Resources Council Library Catalog Lists all publications in the GRC Library, including abstracts, keywords, bibliographic reference location, and URL for the online resources.	No Upload Form Available; MDB file submission
Texas Tech University Off Shore Wells and BHT Database The collection includes raw and corrected BHT, depth, well status, location, ownership, etc. from 8,200 different wells in the Gulf of Mexico. The collection contains a rich quantity of data from multiple depths and dates, with 76,886 rows of data entries. In addition, thousands of scanned geophysical logs were provided by Texas Tech, which are available through interactive use of the SMU Node or from the Well Log Content Model.	About the Dataset.PDF XML Schema (DescribeFeatureType) No Upload Form Available; MDB file submission
University of North Dakota Radiogenic Heat Generation Database. The collection contains data from over 350 specimens analyzed for heat generated by the Thorium, Potassium, or Uranium contents of the rocks. The submission file closely matches the content model layout for Radiogenic Heat Generation.	About the Dataset.PDF XML Schema (DescribeFeatureType) Upload Form UND HeatGeneration
University of North Dakota Thermal Conductivity Database 217 samples were measured for thermal conductivity, which can be used to improve the accuracy of the heat flow calculations for the locations from which they were gathered. The submission file closely matches the content model layout for Thermal Conductivity Observations.	About the Dataset.PDF XML Schema (DescribeFeatureType) Upload Form UND ThermalConductivity
University of North Dakota Temperature at Depth Database Equilibrium Temperature-Depth log measurements of 245 wells in Iowa, Minnesota, Missouri, Nebraska, N. Dakota, and S. Dakota.	About the Dataset.PDF XML Schema (DescribeFeatureType) No Upload Form. Workbook with multiple sheets.
Related Resources All submitters may link notes, publications, images, well logs, etc. to specific wells, either by well name or	Upload Form Related Resources

geographic identifiers. An index.csv file is uploaded with the items specifying how the item is to be related (county, state, polygon, well identifier, etc.)	
Bibliography and Publications SMU, BEG, Cornell, GRC, and UND may all continue to submit additional scanned publications. The bibliography listing the publication reference must first be uploaded, in order for the publications to be discoverable.	Upload Form Universal Bibliography

A User Manual available to the 7 contributing organizations provides detailed instructions, along with screen captures, of the upload process. Additionally, SMU Geothermal Laboratory personnel are developing tutorials to further aid the process. Access to the tutorials as well as the User Manual are available at: <http://geothermal.smu.edu/static/manual/index/GettingStartedSMUNodeofNGDS.htm>

Software Developed by Siemens for the Project

In the interest of ongoing sustainability, the decision was made to permit users to sign on to <http://geothermal.smu.edu> with 'visitor' read-only privileges to reduce the need for ongoing administrative support of the application. User credentials will continue to be required for the data submitters, but the number of users is manageable enough to be within the scope of the SMU Geothermal Laboratory personnel capability.

A full copy of the Schema developed by Siemens for the SMU Node of NGDS is archived at SMU, and is included on a hard drive submitted to the Department of Energy (DOE) as part of the final project documentation. The documentation includes items such as the commands³ to retrieve the database schema. In the future it would therefore be possible to make modifications to the system should the need arise and

³ Command to **retrieve the DDL**, which requires userid ssh access to the sgtl1p1 server and the password for user 'postgres' on sgtl1d1:

```
pg_dump -h sgtl1d1 -U postgres gtda --schema-only --schema=core --schema=staging
--schema=filesystem --format=plain --file filename.ddl
```

Command to generate the names of **the tables and fields for the core database**, which requires userid ssh access to the sgtl1p1 server and the password for user 'postgres' on sgtl1d1:

```
psql -h sgtl1d1 -U postgres gtda -c '\d+ core.*' > gtda-core-tables.txt 2> /dev/null
```

Command to generate the names of **the tables and fields for the staging database**, which requires userid ssh access to the sgtl1p1 server and the password for user 'postgres' on sgtl1d1:

```
psql -h sgtl1d1 -U postgres gtda -c '\d+ staging.*' > gtda-staging-tables.txt 2> /dev/null
```

Command to generate **the file system**, which requires userid and ssh access to server sgtl1p1 and the password for user 'postgres' on server sgtl1d1:

```
psql -h sgtl1d1 -U postgres gtda -c '\d+ filesystem.*' > gtda-filesystem-tables.txt 2> /dev/null
```

should additional funding and personnel be available. SMU currently has the ability and capacity to make minor modifications to static pages such as the <http://geothermal.smu.edu> introductory welcome screen text and the “help” documentation.

In addition to the Schema, Siemens provided full documentation of the software, including administrative functions. A User Manual is available, along with a Systems Operations Manual. Together they provide instruction to visitors of the SMU Node website, instruction for the 7 contributors for data upload, review, publishing etc., and instructions for administrative functions such as updating user ID's.

Server Environment (system software and hardware)

The servers provisioned for the SMU Node are managed by SMU's Office of Information Technology (OIT), not by the SMU Geothermal Laboratory. The servers are Virtual Machines (VM) running on VMware virtualization software and are located at the main SMU data center in Dallas, Texas, with capability to run at a secondary facility, approximately 20 miles away at SMU's Plano, Texas campus.

Sustainability was a key consideration in the selection of SMU's OIT for management of the servers. They have the expertise and infrastructure in place to provide appropriate security, data and system backups, hardware and software routine maintenance, disaster recovery, and call center support. Initially contracted for the three-year project, the SMU Geothermal Lab has an agreement for an additional 7 to 10 years of support from SMU OIT to continue the management of the SMU Node servers. The rate of change for information technology is such that maintenance of the existing platform for 10 years was deemed sufficient.

SMU's OIT follows the university's Computing and Communications Policy, which is appended to this document, as well a number of related policies available at <http://www.smu.edu/BusinessFinance/OIT/Infosec/Policy>.

Security

Security encompasses both physical and system security.

Physical Security

The SMU Data Center, as well as the backup facility, is secured behind a locked door controlled by a card reader with limited access.

System Security

The servers are behind a managed firewall that allows limited access to the internet. User account logins are limited to the servers and are password protected. As mentioned previously, we anticipate the public making use of federated logon based userids to limit the ongoing administrative burden of maintaining userids and passwords long term.



The security of the data housed by the application will be ensured by restricting access to only those users who have the appropriate privileges. Various levels of privilege allow only those users who are granted the privilege to interact with the data in a certain way. General public users making use of the federated logon userid will be granted read-only access to data that has been ‘published’ by the data owners. The act of publishing encompasses the determination that the information is ready to release to public view. Users from the 7 contributing organizations will be granted privileges to allow them to view, and in some cases manipulate, data that has not yet been ‘published’. The ability to contribute additional data, edit existing data, initiate imports of datasets and ‘publish’ data for public viewing each require additional privileges.

Backups

SMU Office of Information Technology (OIT) managed servers are routinely backed up to prevent loss of data. Data is backed up to tape and is housed at a site away from the main campus. The backup schedule for the SMU Node of the NGDS includes a full back up each week with incremental backups 4 days per week. The retention period is 30 days.

Hardware and Software Routine Maintenance

SMU OIT provides full hardware and software support of OIT managed computers. Routine maintenance of hardware includes monitoring, on-site problem determination, problem source identification and repair or replacement of defective items. Hardware upgrades or additional components are handled on an as needed basis as determined necessary by OIT to maintain reliability and efficiency.

Routine maintenance of software includes upgrades or addition of software required by the vendor. Routine operating system upgrades will be handled on a monthly basis. Any high priority software or operating systems updates will be evaluated and performed as needed. Because the system software is open source⁴, the maintenance will be inherently limited to what is made available by the open source community.

Disaster Recovery Plans

If the servers comprising the SMU Node of the NGDS become unavailable because of a disruption due to any reason whether a natural or human-induced disaster, the environment would be recovered at SMU’s campus in Plano, Texas. The SMU-in-Plano campus facility is located approximately 20 miles from the

⁴ Software environment includes Apache Redhat Enterprise Linux (RHEL), PostgreSQL, GeoPortal, GeoServer, and additional open source components.

main campus with close to 200,000 square feet of space with over 50 classrooms, seminar spaces, computer labs and a data center.

Because the servers are virtual servers running on VMware virtualization software, and SMU OIT supports a VMware virtualized environment at both the main campus as well as at the secondary site at SMU-in-Plano, the SMU Node could be recovered at the secondary site.

Call Center and Customer Support Functions

System Administration (Non-routine)

SMU OIT has assigned Systems Engineer Don Kline as the primary representative supporting this project. His area of expertise is Linux, so most of the upgrade or support requests involving the server bank would be performed by him. If Mr. Kline is not the responsible technician for completing the request (e.g., System Security and Network Connectivity are separate areas of expertise) he will communicate the request and be responsible for informing the requestor. If/when, Mr. Kline has moved into a new role, a qualified System Engineer with primary Linux responsibility will be assigned by OIT management.

Help-Desk

The Office of Information Technology at SMU provides support for the servers hosting the SMU Node. The SMU Help Desk can be contacted 24 hours a day, 7 day a week by phone at 214-768-4357 by email at help@smu.edu or online via <http://help.smu.edu> or <https://remotesupport.smu.edu>. The Help Desk staff uses remote tools as well as a shared knowledge base to help provide resolutions to reported issues as quickly as possible. All requests for assistance are documented in the Help Desk software. If the Help Desk is unable to resolve the issue, the ticket will be routed to the appropriate IT technician for resolution. The OIT UNIX Team provides direct support for the geothermal servers.

Network Connectivity

By utilizing the SMU OIT facilities, the SMU Node is able to take advantage of the internet connectivity in place for the university community. Use of this network connectivity, including support of the routers and other telecommunications facilities, is included in the server environment management agreement with SMU OIT.

The assumption is that the connectivity will be at least as good as what is in place in 2012 in terms of speed and capacity. SMU OIT may make changes as needed to maintain reliability, efficiency, and to protect the internal SMU network and server resources.

Support Agreement

The SMU Office of Information Technology has agreed to the following support of the environment for the next 10 years in exchange for a transfer of funds from the SMU Geothermal Laboratory:

<i>Item</i>	<i>Function Description</i>	<i>Comments</i>
1	3 Servers (VM) for Application, Web, and Database	Each with 16GB memory & Quad core processors. 2 servers with 500GB disk storage each and 1 server with 3 TB disk storage on shared Storage Array. Located in physically secured facility with temperature and humidity controls.
2	Routine Server Hardware and operating and system software maintenance	Software environment includes Apache Redhat Enterprise Linux (RHEL), PostgreSQL, GeoPortal, GeoServer, and additional open source components. Maintenance inherently limited to those made available by the open source community. Updates to VM hardware environment to be determined by SMU Office of Information Technology, as needed to maintain reliability and efficiency.
3	Storage Capacity increase allowance of up to 10% per year (simple, not compounded)	This is an allowance and will be charged up front to provide for the additional 10%/year capacity if and when needed. For a 10 year project life, up to 100% additional space on the shared array, or an additional 3 TB. Should storage capacity be required in excess of the growth allowance, OIT will allocate additional space at an incremental cost not to exceed \$500 per TB.
4	Support of Geothermal Data Aggregation Application developed by Siemens Corporate Research	In the event that updates or 'patches' to the application are provided by Siemens and/or the National Geothermal Data System administrators, SMU OIT will install them using the instructions provided by Siemens and/or the NGDS administrators. SMU OIT will assist with loading of large batch updates to records, provided they are in the same file layout/format/structure as currently supported by the applications, using the instructions provided by Siemens. Individual updates, including adding, deleting and modifying existing records will be handled by the users.
5	User Administration Support	SMU Geothermal Laboratory personnel and SMU OIT personnel will both have administration authority to set and modify permission levels for a minimum of 2 users per data provider (BEG at the University of Texas, Cornell University, Geothermal Resources Council, MLKay Technologies, SMU Geothermal Laboratory, Texas Tech University, and University of North Dakota). SMU Geothermal Laboratory and SMU OIT personnel will have the capability to block users, including general public users accessing the system via a federated logon when appropriate.

6	Internet Network Connectivity	Assumes use of the SMU Internet Connectivity in use at that time and accessibility by the general public via the internet. The assumption is that the speed and capacity of the connection will be at least as good

Data Residing on Non-SMU Servers

Sustainability plans for information residing on servers maintained by other members of the project team, such as the Geothermal Resources Council and the Bureau of Economic Geology at the University of Texas are beyond the scope of this plan and are the responsibility of their respective organizations.

Southern Methodist University Computing and Communications Policy

SOUTHERN METHODIST UNIVERSITY

POLICIES AND PROCEDURES

GENERALPOLICY NUMBER: 12.3
2002

REVISED AS OF: December 16,

COMPUTING AND COMMUNICATIONS POLICY

It is the policy of the University that the following rules shall govern the appropriate use of computing and network resources. These rules are intended to permit the greatest use of SMU computing resources consistent with the law and with general principles that govern an academic community. Individuals should be aware that the contents of all files maintained on computer equipment connected to the University network are subject to reasonable administrative inspection if there is reason to believe that violations of University policy and/or civil or criminal wrongdoing is occurring or has occurred. This policy does not preclude system maintenance activities by system administrators, including backing up all files to a server.

This Policy provides a minimum standard for computing and communication equipment and software usage at SMU and does not restrict creation and enforcement of policies for individual campus administrative or academic units. Policies existing elsewhere on campus must comply with this Policy.

A. Acquisition and Deployment of Equipment and Software

Any purchase of computing and communication equipment that may need to interact with other facilities on campus must take into account standards developed by the Campus Standards Committee. The Information Technology Services ("ITS") will make the Committee's recommendations available to the SMU community.

Installation and maintenance of equipment purchased with University funds shall be the responsibility of the purchaser. The purchasing unit shall ensure that the equipment conforms to the standards set by the unit's designated technology support group, whether internal or ITS. If equipment is installed and degrades or causes interruptions to University network services, the equipment shall be disconnected from the network until the purchaser identifies a correction.

ITS implements and maintains University site licenses for software to ensure that University users receive favorable pricing and support terms. Other purchasing units may implement and maintain separate site licenses that cover the unit's operations. The unit maintaining the site license is responsible for maintaining appropriate records of license agreements and for the deployment of licensed copies of the software. The unit may deploy the software itself or make arrangements with the ITS User Services Department to install software. In case of a software audit, each unit will produce the licenses it maintains and assist in the audit of its software. Designated technology support groups within units will perform audits for those units.

B. Usage Policy

SMU computing and networking facilities are intended for conducting and fostering the instructional, research, and administrative activities of the University; use for commercial purposes is expressly prohibited unless otherwise specified by SMU policies. This Policy is designed to provide access in the most open manner possible, while ensuring the proper use of computing facilities consistent with the general principles that govern our academic community. This policy also covers off-campus access to SMU computing resources.

As used in this policy, the term "Computer Resources" is defined to include computers, computer networks, connections to network services such as the Internet and web pages, subscriptions to external computer services, and all peripheral devices. The term "Computer Facility(ies)" includes laboratories, computing centers, public access areas, and other SMU-provided computing equipment. "Licensed Software" refers to copyrighted and proprietary programs, data, and documentation. "Software" refers to the programs, data, and documentation developed from the University's instructional and research activities by faculty, students and staff of SMU.

It is the responsibility of each individual to obtain authorization before his or her use of a Computer Facility. The failure of any designated support group to monitor or prevent unauthorized use does not relieve the individual of that responsibility.

C. Proscribed Activities

The following activities involving use of Computer Resources and Facilities are prohibited and may result, at a minimum, in the immediate denial of computer access and privileges:

- creating, modifying, executing, reverse engineering, or re-transmitting any computer program or instructions intended to gain unauthorized access to, or make unauthorized use of, a Computer Resource, Software, or Licensed Software;
- communicating any information concerning any password, personal identification number or other confidential information without the permission of its owner or the controlling authority of the Computer Facility that oversees the information;
- creating, modifying, executing, or re-transmitting any computer program or instructions intended to obscure the true identity of the sender of electronic mail or electronic messages, collectively referred to as

"Messages," including, but not limited to, forgery of Messages and/or alteration of system and/or user data used to identify the sender of messages;

- accessing or intentionally destroying files, Software, or Licensed Software on a Computer Resource without the permission of the owner of such files, Software, or Licensed Software or the controlling authority of the Computing Resource or Facility;
- making unauthorized copies of Licensed Software;
- communicating any credit card number or other financial account number without the permission of its owner;
- using Computer Resources in a manner inconsistent with the University's contractual obligations to suppliers of Computer Resources or with any published University policy;
- inhibiting or interfering with another's use of the network or computing resources, unless serving as an authorized Systems Administrator (see Policy 12.5);
- using any encryption device, system, or service that prevents compliance with University policy;
- transmitting unsolicited information that contains obscene, indecent, lewd or lascivious materials;
- violating any laws or participating in the commission or furtherance of any crime or other unlawful or improper purpose; and/or
- Conducting any commercial venture through smu.edu, including domain-name registration, file-hosting, or using any University address or telephone number as contact information for a commercial venture, unless otherwise allowed by SMU policies or authorized in writing by the President, the Provost, or a vice president, after consultation with the Controller.

D. Web Pages

Secondary Information Source Providers

The term "Secondary Information Source" refers to existing or future technology that permits text, graphics, photographs, sounds, video images, or other materials to be posted on an SMU-owned computer or listed as a link on an official SMU Internet home page for public access via the Internet. World Wide Web, list servers and gopher servers are examples of Secondary Information Sources.

Secondary Information Sources do not necessarily represent the official views of the University. While they may contain documents generated by University departments, organizations, students, or employees, they are not considered official University publications. Secondary Information Sources will be permitted on University-owned computer equipment within certain guidelines as a means of furthering education and research and fostering exchanges of ideas and opinions.

E. Sanctions

Employees found to be in violation of this Policy are subject to discipline in accordance with University Policy 9.34, "Dishonest, Fraudulent and Illegal Practices," University Policy 6.1, University Policy 6.16, and University Policies 9.12 and 9.13, as appropriate. Dishonest, fraudulent or illegal acts are regarded as egregious and generally warrant termination of employment without the need for progressive discipline.

Students found to be in violation of this Policy are subject to discipline in accordance with the SMU Policies for Community Life. Some cases may warrant investigation by law enforcement agencies.

Explanation of Hard Drive Materials

This document gives an overview of the materials included on the hard-drive as part of SMU Geothermal Laboratory Project DE-EE0002852 “Recovery Act: Geothermal Data Aggregation: Submission of Information into the National Geothermal Data System” for the U.S. Department of Energy. There are several references to this hard drive in the final project report dated February 13, 2014. This explanation is also provided in Appendix F of the final report.

In all, there are seven folders on the hard drive. The headings and subheadings of this document replicate the folder/file structure of the hard drive, as shown in the Table of Contents.

The explanatory document that was provided on the hard drive, and the draft version (February 13, 2014) of the Final Report Appendix F contained references to a URL that has since been changed. This updated version of Appendix F now reflects the change from <http://geothermal.smu.edu/geoserver/> to <http://geothermal.smu.edu:9000/geoserver/web/>.

Notices: This explanation document may be circulated without limitation; however, several of the materials contained on the hard drive are not for general distribution as indicated herein.

Appendix F Table of Contents

0 Explanation of Hard Drive Materials	3
1 Documentation	3
A. End User Documentation	3
1) User Manual	3
2) GTDA Upload Forms	3
3) Catalog Services for the Web (CSW) “Distributor.txt” Files	3
4) Help Information	3
B. Systems Operations and Administration Documentation	4
1) Operations Manual	4
2) Database Schema Documents	4
3) Updating the Static Portions of the Web Page	4
C. Sustainability Plan	5
D. GeoServer Links for 20 Datasets	5
2 Datasets Downloaded from GTDA Feb 10, 2014.....	5
A. DescribeColumns	5
3 PDF Publications Stored in Server.....	5
A. Copies of Bibliographies	5
B. Scanned Publications – Ok to Distribute	5
C. Scanned Publications – NOT Ok to Distribute.....	5
4 Related Resources and Logs Stored in Server.....	5
A. BEG Links to Core Log Library.....	6
B. Cornell Map Image	6
C. SMU Field Notes.....	6
D. SMU TD Curves	6
E. Texas Tech Well Log Images.....	6
5 Dump of GTDA Application Server sgtl1p1 Feb2014	6
6 Dump of GTDA Database Server sgtl1d1 Feb 2014	6

0 Explanation of Hard Drive Materials

This folder only contains this document, the explanation of the materials found on the hard drive. The document is provided as Appendix F in the final project report.

1 Documentation

This folder includes written documentation for the SMU Node of NGDS.

A. End User Documentation

1) User Manual (Also in Appendix C of Final Report)

The User Manual document was written by Siemens to document use of the system, including uploading data, reviewing data, publishing data, querying data, downloading data from the system, use of the interactive capabilities to visualize the data in a map, searching for publications, and more. This document is to be accessible when a user selects the ‘?’ and/or ‘help’ options. SMU Geothermal Laboratory personnel will have the ability to edit and/or supplement it as needed.

2) GTDA Upload Forms

A folder containing workbooks designed to aid the data submitters in uploading refreshed versions of their datasets, publications, and related resources. The Sustainability Plan and the User Manual include references to these forms.

3) Catalog Services for the Web (CSW) “Distributor.txt” Files

Most recently uploaded versions of the text files accompanying any new data submissions. These txt files for each organization contain the information used to create the CSW details exposed to the NGDS and/or visible by accessing <http://geothermal.smu.edu:9000/geoserver/web>. Any subsequent data submissions will need to contain the appropriate distributor.txt file; even if there have been no changes to the contact information, as this updates the metadata with the date of the most recent update.

4) Help Information

This refers to supplementary help information written by SMU Geothermal Laboratory personnel. Items include ‘quick tips’ and ‘frequently asked questions’ to augment the User Manual. When Siemens adds the user manual to the ‘static directory’ where these items will be stored, SMU will preface the user manual with an option to see ‘quick tips’ in lieu of the full manual.

a. Quick Tips

b. Frequently Asked Questions

B. Systems Operations and Administration Documentation

Unlike the End User Documentation, the items in this subfolder are designed for use by SMU personnel and/or others involved in support, backup, maintenance, disaster recovery, or application development activities.

1) Operations Manual (Also in Appendix D of Final Report)

The Operations Manual document was written by Siemens to document systems administration activities of the system, including an introduction to the system functionality, the database structure, instructions to ‘dump and restore’ the system, and more. SMU personnel will have the ability to edit and/or supplement it as needed. Because it contains sensitive password information, it is not in the static folder.

2) Database Schema Documents (Also in Appendix G of Final Report)

a. DDL

b. Database Transform Operations

c. Staging Repository Tables and Field Names

d. Core database Tables and Field Names

e. File System

3) Updating the Static Portions of the Web Page

a. Main Welcome Screen Page

Contains the .doc and .htm versions of the page accessed at <http://geothermal.smu.edu/gtda/>

b. Download Files Button Page

Contains the .doc and .htm versions of the page accessed at <http://geothermal.smu.edu/static/DownloadFilesButtonPage.htm>

c. DescribeColumns Folder

Contains the .xlsx and .pdf versions of the dataset descriptions for each of the Tier 2 datasets.

The Download Files Button Page contains hyperlinks to this folder.

d. ‘Read Me’ Text file Instructions for Updates to Pages

Instructional information necessary for editing or supplementing documents in the static portion of the website. Assumes the person has a registered userid on the servers with sudo and knows the password for the user ‘postgres’ (contained within the Systems Operations Manual).

C. Sustainability Plan (Also in Appendix E of Final Report)

An updated version of the Sustainability Plan. Also contains references to the GTDA Upload Forms.

D. GeoServer Links for 20 Datasets (Also in Appendix A of Final Report)

Hyperlinks to 20 datasets in numerous formats may be found at <http://geothermal.smu.edu/static/DownloadFilesButtonPage.htm> and/or by accessing <http://geothermal.smu.edu:9000/geoserver/web/> and selecting 'data layer'. This section documents the full GeoServer requests used to generate those files as well as links to query the number of features available, the XML schema of files delivered, etc.

2 Datasets Downloaded from GTDA Feb 10, 2014

This folder includes each of the 20 datasets available for delivery from the system formatted as CSV files. Alternatively, a user could access the most current version of these files using the hyperlinks found and/or by accessing <http://geothermal.smu.edu:9000/geoserver/web/> and selecting 'data layer'. The DescribeColumns subfolder contains dataset descriptions for the 10 'Tier 2' files. Note that the dataset descriptions are also located in the Systems Administration Documentation under the section regarding updating the static portions of the website.

A. DescribeColumns

3 PDF Publications Stored in Server

As with the datasets downloaded from GTDA, this folder contains information retrieved from the server. It contains scanned library type publications and reports. Optical Character Recognition was utilized for purposes of returning search results. Many publications are available as PDFs for download from the system ('OK to Distribute'). Publications that are 'NOT OK TO DISTRIBUTE' are copyrighted and/or otherwise restricted. They are included solely for returning bibliographic references to users performing publication searches. Possession of this hard drive does NOT indicate a license to distribute them. These items are explicitly excluded from the right to copy and distribute further.

A. Copies of Bibliographies

B. Scanned Publications – Ok to Distribute

C. Scanned Publications – NOT Ok to Distribute

4 Related Resources and Logs Stored in Server

This folder contains copies of the various related field notes, Temperature Depth Curves, well log images, and other 'related resources'. As with the publications and datasets, this information is also contained within the server, but is provided here for ease of access.

A. BEG Links to Core Log Library

B. Cornell Map Image

C. SMU Field Notes

D. SMU TD Curves

E. Texas Tech Well Log Images

5 Dump of GTDA Application Server sgtl1p1 Feb2014

This folder contains a ‘dump’ of the server containing the GTDA application, including the code itself. This item is not intended for general distribution. It is being provided to the Department of Energy to validate completion of the project objectives for project DE-EE0002852. Restore instructions may be found within the Systems Operations Manual.

6 Dump of GTDA Database Server sgtl1d1 Feb 2014

This folder contains a ‘dump’ of the server containing the database used by GTDA. This item is not intended for general distribution. It is being provided to the Department of Energy to validate completion of the project objectives for project DE-EE0002852. Restore instructions may be found within the Systems Operations Manual.

Appendix G: Database Schema for GTDA

The GTDA Systems Operations Manual (Appendix D) provided by Siemens Corporate Technology provides instructions for accessing the Database Schema and includes an example query result of one of the tables with fields. This Appendix G contains April 7, 2014 versions of the following:

- [the Data Definition Language \(DDL\),](#)
- [all tables and fields in the 'staging' repository \(with table of contents\),](#)
- [all tables and fields in the 'core' database \(with table of contents\),](#)
- [the file system,](#)
- [all 'staging' transform operations](#)
- [all 'core' transform operations](#)

For an introductory explanation of staging versus core areas of the system, as well as transform operations, please refer to the final report section "II. Review of Statement of Project Objectives, Task 4, Development of Data Repository" *on Page11 of the main final report body.*

Appendix G: Database Schema for GTDA	Appendix G: 1
Data Definition Language (DDL) for the GTDA System	Appendix G: 2
Tables and Fields in the Staging Database	Appendix G: 217
Tables and Fields in the Core Database	Appendix G: 263
File System	Appendix G: 330
Staging Transform Operations	Appendix G: 332
Core Transform Operations	Appendix G: 374

Data Definition Language (DDL) for the GTDA System

pulled April 7, 2014 using the command:

```
pg_dump -h sgtl1d1 -U postgres gtda --schema-only --schema=core --schema=staging --  
schema=filesystem --format=plain --file filename.ddl
```

*Requires signing onto sgtl1p1 using PuTTY. Also requires the password for the userid
'postgres'.*

--

-- PostgreSQL database dump

--

```
SET statement_timeout = 0;  
SET client_encoding = 'UTF8';  
SET standard_conforming_strings = off;  
SET check_function_bodies = false;  
SET client_min_messages = warning;  
SET escape_string_warning = off;
```

--

```
-- Name: core; Type: SCHEMA; Schema: -; Owner: gtda
```

--

```
CREATE SCHEMA core;
```

```
ALTER SCHEMA core OWNER TO gtda;
```

--

```
-- Name: filesystem; Type: SCHEMA; Schema: -; Owner: postgres
```

--

```
CREATE SCHEMA filesystem;
```

```
ALTER SCHEMA filesystem OWNER TO postgres;
```

--

```
-- Name: staging; Type: SCHEMA; Schema: -; Owner: postgres
```

--

```
CREATE SCHEMA staging;
```



```
ALTER SCHEMA staging OWNER TO postgres;
```

```
SET search_path = core, pg_catalog;
```

```
--  
-- Name: bounding_box(integer, integer, integer, integer); Type: FUNCTION; Schema: core;  
Owner: postgres  
--
```

```
CREATE FUNCTION bounding_box(lower_latitude integer, lower_longitude integer,  
upper_latitude integer, upper_longitude integer) RETURNS TABLE(name character varying,  
site_name character varying, latitude character varying, longitude character varying, state_code  
character varying, county character varying, max_temperature character varying, depth  
character varying, datasource character varying, thermal_conductivity character varying,  
heat_flow character varying)
```

```
    LANGUAGE plpgsql
```

```
    AS $_$
```

```
BEGIN
```

```
    RETURN QUERY SELECT DISTINCT COALESCE(s.site_name, s.api), s.site_name,  
s.latitude, s.longitude, st.state_abbr, c.county_name, bht.bht, GREATEST(d.depth,  
bht.bht_depth), ds.datasource, tc.conductivity, hf.heat_flow
```

```
    FROM core.surface_site s
```

```
    LEFT JOIN core.ref_county c ON c.id = s.county_id
```

```
    LEFT JOIN core.ref_state st ON st.id = c.state_id
```

```
    LEFT JOIN core.depth d ON d.surface_site_id = s.id
```

```
    LEFT JOIN core.bht_measurements bht ON bht.surface_site_id = s.id
```

```
    LEFT JOIN core.datasets ds ON ds.id = s.dataset_id
```

```
    LEFT JOIN core.thermal_conductivity tc ON tc.surface_site_id = s.id
```

```
    LEFT JOIN core.heat_flow hf ON hf.surface_site_id = s.id
```

```
    LEFT JOIN core.precision_log pl ON pl.surface_site_id = s.id
```

```
    where s.latitude > $1 and s.longitude > $2 and s.latitude < $3 and s.longitude < $4;
```

```
END;
```

```
$_$;
```

```
ALTER FUNCTION core.bounding_box(lower_latitude integer, lower_longitude integer,  
upper_latitude integer, upper_longitude integer) OWNER TO postgres;
```

```
--  
-- Name: create_matview(name, name); Type: FUNCTION; Schema: core; Owner: postgres  
--
```

```
CREATE FUNCTION create_matview(name, name) RETURNS void
```

```
    LANGUAGE plpgsql SECURITY DEFINER
```

```

AS $_$
DECLARE
    matview ALIAS FOR $1;
    view_name ALIAS FOR $2;
    entry core.matviews%ROWTYPE;
BEGIN
    SELECT * INTO entry FROM core.matviews WHERE mv_name = matview;

    IF FOUND THEN
        RAISE EXCEPTION 'Materialized view "%" already exists.',
            matview;
    END IF;

    EXECUTE 'REVOKE ALL ON ' || view_name || ' FROM postgres';
    EXECUTE 'GRANT SELECT ON ' || view_name || ' TO postgres';
    EXECUTE 'CREATE TABLE ' || matview || ' AS SELECT * FROM ' || view_name;
    EXECUTE 'REVOKE ALL ON ' || matview || ' FROM postgres';
    EXECUTE 'GRANT SELECT ON ' || matview || ' TO postgres';

    INSERT INTO core.matviews (mv_name, v_name, last_refresh)
        VALUES (matview, view_name, CURRENT_TIMESTAMP);

    RETURN;
END
$_$;

```

```

ALTER FUNCTION core.create_matview(name, name) OWNER TO postgres;

```

```

--
-- Name: drop_matview(name); Type: FUNCTION; Schema: core; Owner: postgres
--

```

```

CREATE FUNCTION drop_matview(name) RETURNS void
    LANGUAGE plpgsql SECURITY DEFINER
    AS $_$
DECLARE
    matview ALIAS FOR $1;
    entry core.matviews%ROWTYPE;
BEGIN
    SELECT * INTO entry FROM core.matviews WHERE mv_name = matview;

    IF NOT FOUND THEN
        RAISE EXCEPTION 'Materialized view % does not exist.', matview;
    END IF;

```

```

EXECUTE 'DROP TABLE ' || matview;
DELETE FROM core.matviews WHERE mv_name=matview;

RETURN;
END
$_$;

ALTER FUNCTION core.drop_matview(name) OWNER TO postgres;

--
-- Name: generate_uuid(character varying, character varying, character varying); Type:
FUNCTION; Schema: core; Owner: postgres
--

CREATE FUNCTION generate_uuid(datasource character varying, datatype character varying,
resource_id character varying) RETURNS uuid
    LANGUAGE plpgsql
    AS $$
DECLARE
    uri VARCHAR(128) := 'http://geothermal.smu.edu';
BEGIN
    IF (datasource IS NOT NULL)
        THEN uri := uri || '/' || datasource;
    END IF;
    IF (datatype IS NOT NULL)
        THEN uri := uri || '/' || datatype;
    END IF;
    IF (resource_id IS NOT NULL)
        THEN uri := uri || '/' || resource_id;
    END IF;
    RETURN uuid3(uri);
END;
$$;

ALTER FUNCTION core.generate_uuid(datasource character varying, datatype character
varying, resource_id character varying) OWNER TO postgres;

--
-- Name: populateprecisionlog(character varying, character varying); Type: FUNCTION;
Schema: core; Owner: postgres
--

CREATE FUNCTION populateprecisionlog(uuid character varying, logsourceid character
varying) RETURNS boolean

```

```

LANGUAGE plpgsql
AS $$
DECLARE
    precision_log_id uuid;
    sourceid character;
    temp_dataset_id uuid;
    temp_log_deepest_depth double precision;
    temp_log_deepest_temperature double precision;
    temp_log_max_temperature double precision;
BEGIN
    precision_log_id=$1;
    sourceid = $2;
    select td.precision_log_id , td.dataset_id,max(td.depth) OVER (PARTITION BY
td.precision_log_id),
        last_value(td.temperature) OVER (PARTITION BY td.precision_log_id ORDER by
td.depth),max(td.temperature) OVER (PARTITION BY td.precision_log_id)
        into precision_log_id, temp_dataset_id, temp_log_deepest_depth,
temp_log_deepest_temperature, temp_log_max_temperature from core.temperature_series td
        where td.precision_log_id = precision_log_id;
    IF NOT FOUND THEN
        RETURN FALSE;
    ELSE
        -- SET AGGREGATE VALUES FOR LOG
        UPDATE core.precision_log
        SET dataset_id = temp_dataset_id,
            precision_log_source_id = sourceid,
            log_depth = temp_log_deepest_depth,
            log_deepest_temperature = temp_log_deepest_temperature,
            log_max_temperature = temp_log_max_temperature
        WHERE id = precision_log_id;

        -- LINK LOG TO SURFACE_SITE (WELL) IF MAPPING IS KNOWN
        UPDATE core.precision_log l
        SET surface_site_id = s.id
        FROM core.ref_temperature_series_mapping m, core.surface_site s
        WHERE l.precision_log_source_id = m.precision_log_source_id
            AND l.dataset_id = m.data_source_id
            AND s.dataset_id = m.data_source_id
            AND s.source_id = m.surface_site_source_id
            AND l.id = precision_log_id;
        RETURN true;
    END IF;
END;
$$;

```

```
ALTER FUNCTION core.populateprecisionlog(uuid character varying, logsourceid character
varying) OWNER TO postgres;
```

```
--
-- Name: recreate_all_matviews(); Type: FUNCTION; Schema: core; Owner: postgres
--
```

```
CREATE FUNCTION recreate_all_matviews() RETURNS integer
  LANGUAGE plpgsql
  AS $$
BEGIN
```

```
-- recreate all materialized views in core:
```

```
RAISE INFO 'PostgreSQL is now dropping core.surface_site_county_state_materialized_view';
perform core.drop_matview('core.surface_site_county_state_materialized_view');
RAISE INFO 'core.surface_site_county_state_materialized_view has been dropped.';
```

```
RAISE INFO 'PostgreSQL is now dropping core.template_borehole_materialized';
perform core.drop_matview('core.template_borehole_materialized');
RAISE INFO 'core.template_borehole_materialized has been dropped.';
```

```
RAISE INFO 'PostgreSQL is now dropping core.template_fluidflux_materialized';
perform core.drop_matview('core.template_fluidflux_materialized');
RAISE INFO 'core.template_fluidflux_materialized has been dropped.';
```

```
RAISE INFO 'PostgreSQL is now dropping core.template_heatflow_materialized';
perform core.drop_matview('core.template_heatflow_materialized');
RAISE INFO 'core.template_heatflow_materialized has been dropped.';
```

```
RAISE INFO 'PostgreSQL is now dropping core.template_radiogenicheat_materialized';
perform core.drop_matview('core.template_radiogenicheat_materialized');
RAISE INFO 'core.template_radiogenicheat_materialized has been dropped.';
```

```
RAISE INFO 'PostgreSQL is now dropping core.template_reservoir_materialized';
perform core.drop_matview('core.template_reservoir_materialized');
RAISE INFO 'core.template_reservoir_materialized has been dropped.';
```

```
RAISE INFO 'PostgreSQL is now dropping core.template_thermalconductivity_materialized';
perform core.drop_matview('core.template_thermalconductivity_materialized');
RAISE INFO 'core.template_thermalconductivity_materialized has been dropped.';
```

```
RAISE INFO 'PostgreSQL is now dropping core.template_wellfluid_materialized';
perform core.drop_matview('core.template_wellfluid_materialized');
RAISE INFO 'core.template_wellfluid_materialized has been dropped.';
```

```
RAISE INFO 'PostgreSQL is now dropping core.template_wellheader_materialized';
```

```
perform core.drop_matview('core.template_wellheader_materialized');
RAISE INFO 'core.template_wellheader_materialized has been dropped.';
```

```
RAISE INFO 'PostgreSQL is now dropping core.template_welllog_materialized';
perform core.drop_matview('core.template_welllog_materialized');
RAISE INFO 'core.template_welllog_materialized has been dropped.';
```

```
RAISE INFO 'PostgreSQL is now dropping core.template_welltest_materialized';
perform core.drop_matview('core.template_welltest_materialized');
RAISE INFO 'core.template_welltest_materialized has been dropped.';
```

```
RAISE INFO 'PostgreSQL is now creating matview
core.surface_site_county_state_materialized_view';
perform
core.create_matview('core.surface_site_county_state_materialized_view','core.surface_site_cou
nty_state_view');
RAISE INFO 'core.surface_site_county_state_materialized_view has been created';
```

```
RAISE INFO 'PostgreSQL is now creating matview core.template_borehole_materialized';
perform core.create_matview('core.template_borehole_materialized','core.template_borehole');
RAISE INFO 'core.template_borehole_materialized has been created';
```

```
RAISE INFO 'PostgreSQL is now creating matview core.template_fluidflux_materialized';
perform core.create_matview('core.template_fluidflux_materialized','core.template_fluidflux');
RAISE INFO 'core.template_fluidflux_materialized has been created';
```

```
RAISE INFO 'PostgreSQL is now creating matview core.template_heatflow_materialized';
perform core.create_matview('core.template_heatflow_materialized','core.template_heatflow');
RAISE INFO 'core.template_heatflow_materialized has been created';
```

```
RAISE INFO 'PostgreSQL is now creating matview
core.template_radiogenicheat_materialized';
perform
core.create_matview('core.template_radiogenicheat_materialized','core.template_radiogenicheat
');
RAISE INFO 'core.template_radiogenicheat_materialized has been created';
```

```
RAISE INFO 'PostgreSQL is now creating matview core.template_reservoir_materialized';
perform core.create_matview('core.template_reservoir_materialized','core.template_reservoir');
RAISE INFO 'core.template_reservoir_materialized has been created';
```

```
RAISE INFO 'PostgreSQL is now creating matview
core.template_thermalconductivity_materialized';
perform
core.create_matview('core.template_thermalconductivity_materialized','core.template_thermalc
onductivity');
```

```

RAISE INFO 'core.template_thermalconductivity_materialized has been created';

RAISE INFO 'PostgreSQL is now creating matview core.template_wellfluid_materialized';
perform core.create_matview('core.template_wellfluid_materialized','core.template_wellfluid');
RAISE INFO 'core.template_wellfluid_materialized has been created';

RAISE INFO 'PostgreSQL is now creating matview core.template_wellheader_materialized';
perform
core.create_matview('core.template_wellheader_materialized','core.template_wellheader');
RAISE INFO 'core.template_wellheader_materialized has been created';

RAISE INFO 'PostgreSQL is now creating matview core.template_welllog_materialized';
perform core.create_matview('core.template_welllog_materialized','core.template_welllog');
RAISE INFO 'core.template_welllog_materialized has been created';

RAISE INFO 'PostgreSQL is now creating matview core.template_welltest_materialized';
perform core.create_matview('core.template_welltest_materialized','core.template_welltest');
RAISE INFO 'core.template_welltest_materialized has been created';

RETURN 0;
END;
$$;

ALTER FUNCTION core.recreate_all_matviews() OWNER TO postgres;

--
-- Name: refresh_matview(name); Type: FUNCTION; Schema: core; Owner: postgres
--

CREATE FUNCTION refresh_matview(name) RETURNS void
  LANGUAGE plpgsql SECURITY DEFINER
  AS $_$
DECLARE
  matview ALIAS FOR $1;
  entry core.matviews%ROWTYPE;
BEGIN

  SELECT * INTO entry FROM core.matviews WHERE mv_name = matview;

  IF NOT FOUND THEN
    RAISE EXCEPTION 'Materialized view % does not exist.', matview;
  END IF;

  EXECUTE 'DELETE FROM ' || matview;
  EXECUTE 'INSERT INTO ' || matview || ' SELECT * FROM ' || entry.v_name;

```

```

UPDATE core.matviews
SET last_refresh=CURRENT_TIMESTAMP
WHERE mv_name=matview;

RETURN;
END
$$;

ALTER FUNCTION core.refresh_matview(name) OWNER TO postgres;

--
-- Name: register_dataset(character varying, character varying, character varying); Type:
FUNCTION; Schema: core; Owner: postgres
--

CREATE FUNCTION register_dataset(datasource character varying, datatype character
varying, resource_id character varying) RETURNS uuid
LANGUAGE plpgsql
AS $$
DECLARE
    dset core.datasets%rowtype;
    id UUID;
BEGIN
    IF datasource IS NULL
        THEN RAISE EXCEPTION 'Missing information for parameter data source name';
    END IF;

    SELECT * FROM core.datasets d
        WHERE datasource = d.datasource
            AND coalesce(d.datatype, '') = coalesce(datatype, d.datatype, '')
            AND coalesce(d.resource_id, '') = coalesce(resource_id, d.resource_id, '')
    INTO dset;

    IF (dset.id IS null)
        THEN
            id := core.generate_uuid(datasource, datatype, resource_id);
            INSERT INTO core.datasets VALUES (id, datasource, datatype, resource_id);
        ELSE
            id := dset.id;
        END IF;

    RETURN id;
END;
$$;

```



```
ALTER FUNCTION core.register_dataset(datasource character varying, datatype character
varying, resource_id character varying) OWNER TO postgres;
```

```
--
-- Name: remove_dataset(character varying, character varying, character varying); Type:
FUNCTION; Schema: core; Owner: postgres
--
```

```
CREATE FUNCTION remove_dataset(datasource character varying, datatype character
varying, resource_id character varying) RETURNS integer
```

```
    LANGUAGE plpgsql
```

```
    AS $$
```

```
DECLARE
```

```
    datasets CURSOR FOR SELECT d.id FROM core.datasets d
```

```
    WHERE datasource = d.datasource
```

```
    AND coalesce(d.datatype, '') = coalesce(datatype, d.datatype, '')
```

```
    AND coalesce(d.resource_id, '') = coalesce(resource_id, d.resource_id, '');
```

```
BEGIN
```

```
    IF datasource IS NULL
```

```
        THEN RAISE EXCEPTION 'Missing information for parameter data source name';
```

```
    END IF;
```

```
    --OPEN datasets;
```

```
    FOR dsetid IN datasets LOOP
```

```
        BEGIN DELETE FROM core.heat_flow WHERE dataset_id=dsetid.id; END;
```

```
        BEGIN DELETE FROM core.radioactivity WHERE dataset_id=dsetid.id; END;
```

```
        BEGIN DELETE FROM core.depth WHERE dataset_id=dsetid.id; END;
```

```
        --BEGIN DELETE FROM core.surface_site_detailed WHERE dataset_id=dsetid.id;
```

```
    END;
```

```
        --BEGIN DELETE FROM core.site_comments WHERE dataset_id=dsetid.id; END;
```

```
        BEGIN DELETE FROM core.notes WHERE dataset_id=dsetid.id; END;
```

```
        BEGIN DELETE FROM core.thermal_conductivity WHERE dataset_id=dsetid.id;
```

```
    END;
```

```
        --BEGIN DELETE FROM core.filter_condition WHERE dataset_id=dsetid.id; END;
```

```
        --BEGIN DELETE FROM core.ngds_heat_flow_mview WHERE dataset_id=dsetid.id;
```

```
    END;
```

```
        --BEGIN DELETE FROM core.quality_issue_log WHERE dataset_id=dsetid.id; END;
```

```
        BEGIN DELETE FROM core.publication WHERE dataset_id=dsetid.id; END;
```

```
        --BEGIN DELETE FROM core.publication_reference WHERE dataset_id=dsetid.id;
```

```
    END;
```

```
        --BEGIN DELETE FROM core.site_publication_links WHERE dataset_id=dsetid.id;
```

```
    END;
```

```
        BEGIN DELETE FROM core.formation WHERE dataset_id=dsetid.id; END;
```

```

BEGIN DELETE FROM core.well_test WHERE dataset_id=dsetid.id; END;
BEGIN DELETE FROM core.well_dates WHERE dataset_id=dsetid.id; END;
BEGIN DELETE FROM core.well_info WHERE dataset_id=dsetid.id; END;

BEGIN DELETE FROM core.fluid_h10 WHERE dataset_id=dsetid.id; END;
BEGIN DELETE FROM core.fluid_h10h WHERE dataset_id=dsetid.id; END;
BEGIN DELETE FROM core.fluid WHERE dataset_id=dsetid.id; END;

BEGIN DELETE FROM core.temperature_series WHERE dataset_id=dsetid.id; END;
BEGIN DELETE FROM core.precision_log WHERE dataset_id=dsetid.id; END;

BEGIN DELETE FROM core.site_related_resources WHERE dataset_id=dsetid.id;
END;

--BEGIN DELETE FROM core.heat_flow_publication_links WHERE
dataset_id=dsetid.id; END;
BEGIN DELETE FROM core.bht_measurements WHERE dataset_id=dsetid.id; END;
BEGIN DELETE FROM core.casing WHERE dataset_id=dsetid.id; END;
BEGIN DELETE FROM core.aquifier WHERE dataset_id=dsetid.id; END;
BEGIN DELETE FROM core.interval WHERE dataset_id=dsetid.id; END;
BEGIN DELETE FROM core.surface_site WHERE dataset_id=dsetid.id; END;

BEGIN DELETE FROM core.reservoir WHERE dataset_id=dsetid.id; END;
END LOOP;

RETURN 0;
END;
$$;

ALTER FUNCTION core.remove_dataset(datasource character varying, datatype character
varying, resource_id character varying) OWNER TO postgres;

--
-- Name: update_all_matviews(); Type: FUNCTION; Schema: core; Owner: postgres
--

CREATE FUNCTION update_all_matviews() RETURNS integer
  LANGUAGE plpgsql
  AS $$
BEGIN

-- update materialized views in core:

```

```

perform core.refresh_matview('core.surface_site_county_state_materialized_view');
perform core.refresh_matview('core.template_borehole_materialized');
perform core.refresh_matview('core.template_fluidflux_materialized');
perform core.refresh_matview('core.template_heatflow_materialized');
perform core.refresh_matview('core.template_radiogenicheat_materialized');
perform core.refresh_matview('core.template_reservoir_materialized');
perform core.refresh_matview('core.template_thermalconductivity_materialized');
perform core.refresh_matview('core.template_wellfluid_materialized');
perform core.refresh_matview('core.template_wellheader_materialized');
perform core.refresh_matview('core.template_welllog_materialized');
perform core.refresh_matview('core.template_welltest_materialized');

```

```

RETURN 0;
END;
$$;

```

```

ALTER FUNCTION core.update_all_matviews() OWNER TO postgres;

```

```

--
-- Name: update_surface_site_detailed(); Type: FUNCTION; Schema: core; Owner: postgres
--

```

```

CREATE FUNCTION update_surface_site_detailed() RETURNS boolean
    LANGUAGE plpgsql
    AS $$
DECLARE
    row core.surface_site%ROWTYPE;
    temp_deepest_temperature double precision;
    temp_depth double precision;
BEGIN

    DELETE FROM core.surface_site_detailed;
    FOR row IN SELECT * FROM core.surface_site
    LOOP
        SELECT depth
            INTO temp_depth
            FROM core.depth where surface_site_id=row.id;
        SELECT max(log_deepest_temperature) OVER (PARTITION BY
surface_site_id)
            INTO temp_deepest_temperature
            FROM core.precision_log where surface_site_id=row.id;
        INSERT INTO core.surface_site_detailed(id, data_source_id,site_name
,source_id ,depth , deepest_temperature )
            VALUES(row.id ,row.data_source_id ,row.site_name,row.source_id
,temp_depth ,temp_deepest_temperature);
    END LOOP;
END;

```

```

        END LOOP;
        RETURN TRUE;
END;
$$;

ALTER FUNCTION core.update_surface_site_detailed() OWNER TO postgres;

SET search_path = staging, pg_catalog;

--
-- Name: beg_data_transformationx(); Type: FUNCTION; Schema: staging; Owner: postgres
--

CREATE FUNCTION beg_data_transformationx() RETURNS integer
    LANGUAGE plpgsql
    AS $_$

DECLARE
    dsetUUID uuid;
BEGIN

    select core.register_dataset('BEG','ALL',NULL) into dsetUUID;

    INSERT INTO core.surface_site (id, dataset_id ,api, latitude, longitude,
        lease_name, other_location_name, site_name,operator,contributor,comment)
    select distinct * from (SELECT
        first_value(core.generate_uuid('BEG','SURFACE_SITE',lat || ','|| long)) OVER
(PARTITION BY lat,long) AS id,
        dsetUUID as dataset_id ,
        first_value(api) OVER (PARTITION BY lat,long) as api,
        first_value(lat) OVER (PARTITION BY lat,long) as latitude,
        first_value(long) OVER (PARTITION BY lat,long) as longitude,
        first_value(lease) OVER (PARTITION BY lat,long) as lease_name,
        first_value("location") OVER (PARTITION BY lat,long) as
other_location_name,
        first_value(wellname) OVER (PARTITION BY lat,long) as site_name,
        first_value(operator_n) OVER (PARTITION BY lat,long) as operator,
        first_value('BEG: '||source) OVER (PARTITION BY lat,long) as contributor,
        first_value('BEG accession core database: '||accession_log_beg) OVER
(PARTITION BY lat,long) as comment
        FROM staging.beg_well_data) Z;

```

```
UPDATE core.surface_site SET shape = geometryfromtext('POINT(' || longitude || ' ' ||
latitude ||)'), 4326) WHERE dataset_id = dsetUUID;
```

```
UPDATE core.surface_site SET county_id=c.id FROM core.ref_county c WHERE
ST_Contains(c.geometry,shape) and dataset_id = dsetUUID;
```

```
INSERT INTO core.formation (surface_site_id, dataset_id, formation,formation_depth)
SELECT distinct
    core.generate_uuid('BEG','SURFACE_SITE',lat || ',' || long),
    dsetUUID as dataset_id,
    formation,
    depth_m
FROM staging.beg_well_data
where length(formation)>0;
```

```
INSERT INTO core.formation (surface_site_id, dataset_id, formation,formation_depth)
SELECT distinct
    core.generate_uuid('BEG','SURFACE_SITE',lat || ',' || long),
    dsetUUID as dataset_id,
    fm_at_td,
    tvd/3.280839895 -- converts to meters
FROM staging.beg_well_data
where length(fm_at_td)>0;
```

```
INSERT INTO core.well_info (surface_site_id, dataset_id, well_id)
SELECT distinct
    core.generate_uuid('BEG','SURFACE_SITE',lat || ',' || long) AS surface_site_id,
    dsetUUID as dataset_id,
    wellid
FROM staging.beg_well_data
where wellid is not null;
```

```
INSERT INTO core.depth (surface_site_id, dataset_id, true_vertical_depth)
SELECT distinct
    core.generate_uuid('BEG','SURFACE_SITE',lat || ',' || long),
    dsetUUID,
    tvd/3.280839895 -- converts to meters
FROM staging.beg_well_data WHERE tvd is not null;
```

```
INSERT INTO core.bht_measurements (id,surface_site_id, dataset_id, run_number,
bht_depth,bht,bht_corrected,drill_date)
SELECT distinct
    core.generate_uuid('BEG','BHT',lat || ',' || long || ';' || coalesce(depth_m,0) || ';' ||
row_number() OVER (partition by lat, long, depth_m)) AS id,
    core.generate_uuid('BEG','SURFACE_SITE',lat || ',' || long) AS surface_site_id,
```

```

        dsetUUID,
        number_run AS run_number,
        depth_m AS bht_depth,
        bht_c AS bht,
        bht_c_cor AS bht_corrected,
        log_date AS drill_date
    FROM staging.beg_well_data;

    INSERT INTO core.heat_flow (id,bht_measurements_id,surface_site_id, dataset_id,
    gradient_corrected)
    SELECT distinct
        core.generate_uuid('BEG','HEAT_FLOW',lat || ',' || long || ';' ||
        coalesce(depth_m,0) || ';' || row_number() OVER (partition by lat, long, depth_m)) AS id,
        core.generate_uuid('BEG','BHT',lat || ',' || long || ';' || coalesce(depth_m,0) || ';' ||
        row_number() OVER (partition by lat, long, depth_m)) AS bht_measurements_id,
        core.generate_uuid('BEG','SURFACE_SITE',lat || ',' || long) AS surface_site_id,
        dsetUUID,
        (gradient_per_1000ft*5/9)*3.280839895 AS gradient_corrected -- convert to
        delta(C)/km from delta(F)/1000ft
    FROM staging.beg_well_data WHERE gradient_per_1000ft is not null;

-- add well logs:

/*
-- query to get linking info:
select substring(L.log_image_addr from '[a-zA-Z0-9]+.tif+$') as filename, 'WELL_LOG' as
"type", L.log_types as description, core.generate_uuid('BEG','SURFACE_SITE',lat || ',' || long)
as site_identifier, W.api, W.lat as lat, W.long as long
    from staging.beg_logs_with_images L JOIN staging.beg_well_data W ON
    L.wellid=W.wellid
    WHERE W.wellid is not null

-- not used anymore
    INSERT INTO core.site_related_resources (surface_site_id, label, resource_id,url,
    dataset_id,"type",shape)
    SELECT distinct
        core.generate_uuid('BEG','SURFACE_SITE',lat || ',' || long),
        file_name,
        core.generate_uuid('BEG','WELL_LOG',file_name),
        'https://geothermal.smu.edu/gtda/rest/fileSystem/getFile?uuid=' ||
        core.generate_uuid('BEG','WELL_LOG',file_name),
        dsetUUID,
        'WELL_LOG',
        geometryfromtext('POINT(' || long || ' ' || lat ||)'), 4326)

```

```

FROM (
  select W.lat as lat, W.long as long, L.log_image_addr as file_name
  from staging.beg_logs_with_images L JOIN staging.beg_well_data W ON
L.wellid=W.wellid
  WHERE W.wellid is not null
) Z;
*/

```

--- add casing info:

```

INSERT INTO core.casing (surface_site_id, dataset_id, casing_size_inches,
casing_depth, liner_inches, liner_top, liner_base, choke_mm)
SELECT distinct
  core.generate_uuid('BEG','SURFACE_SITE',lat || ',' || long) AS
surface_site_id,
  dsetUUID,
  casing_size_inches,
  casing_depth,
  liner_inches,
  liner_top,
  liner_base,
  choke_mm
FROM staging.beg_well_data;

```

--- add interval info up to 8 times:

```

INSERT INTO core.interval (surface_site_id, dataset_id,top_depth, bottom_depth,
interval_length, water_volume, bhp, duration_hours, producing_reservoir)
SELECT distinct
  core.generate_uuid('BEG','SURFACE_SITE',lat || ',' || long) AS
surface_site_id,
  dsetUUID,
  top_ft_1/3.280839895,
  bottom_ft_1/3.280839895,
  interval_ft_1/3.280839895,
  watrvol_1_bbl,
  bhp_1,
  durtn_hrs_1,
  prod_reservoir_1
FROM staging.beg_well_data
WHERE top_ft_1 is not null and bottom_ft_1 is not null;

```

```

INSERT INTO core.interval (surface_site_id, dataset_id,top_depth, bottom_depth,
interval_length, water_volume, bhp, duration_hours, producing_reservoir)
SELECT distinct
  core.generate_uuid('BEG','SURFACE_SITE',lat || ',' || long) AS
surface_site_id,

```

```

        dsetUUID,
        top_ft_2/3.280839895,
        bottom_ft_2/3.280839895,
        interval_ft_2/3.280839895,
        watrvol_2_bbl,
        bhp_2,
        durtn_hrs_2,
        prod_reservoir_2
    FROM staging.beg_well_data
    WHERE top_ft_2 is not null and bottom_ft_2 is not null;

```

```

INSERT INTO core.interval (surface_site_id, dataset_id,top_depth, bottom_depth,
interval_length, water_volume, bhp, duration_hours, producing_reservoir)
    SELECT distinct
        core.generate_uuid('BEG','SURFACE_SITE',lat || ','|| long) AS
surface_site_id,

```

```

        dsetUUID,
        top_ft_3/3.280839895,
        bottom_ft_3/3.280839895,
        interval_ft_3/3.280839895,
        watrvol_3_bbl,
        bhp_3,
        durtn_hrs_3,
        prod_reservoir_3
    FROM staging.beg_well_data
    WHERE top_ft_3 is not null and bottom_ft_3 is not null;

```

```

INSERT INTO core.interval (surface_site_id, dataset_id,top_depth, bottom_depth,
interval_length, water_volume, bhp, duration_hours, producing_reservoir)
    SELECT distinct
        core.generate_uuid('BEG','SURFACE_SITE',lat || ','|| long) AS
surface_site_id,

```

```

        dsetUUID,
        top_ft_4/3.280839895,
        bottom_ft_4/3.280839895,
        interval_ft_4/3.280839895,
        watrvol_4_bbl,
        bhp_4,
        durtn_hrs_4,
        prod_reservoir_4
    FROM staging.beg_well_data
    WHERE top_ft_4 is not null and bottom_ft_4 is not null;

```

```

INSERT INTO core.interval (surface_site_id, dataset_id,top_depth, bottom_depth,
interval_length, water_volume, bhp, duration_hours, producing_reservoir)

```



```

SELECT distinct
    core.generate_uuid('BEG','SURFACE_SITE',lat || ',' || long) AS
surface_site_id,
    dsetUUID,
    top_ft_5/3.280839895,
    bottom_ft_5/3.280839895,
    interval_ft_5/3.280839895,
    watrvol_5_bbl,
    bhp_5,
    durtn_hrs_5,
    prod_reservoir_5
FROM staging.beg_well_data
WHERE top_ft_5 is not null and bottom_ft_5 is not null;

```

```

INSERT INTO core.interval (surface_site_id, dataset_id,top_depth, bottom_depth,
interval_length, water_volume, bhp, duration_hours, producing_reservoir)
SELECT distinct
    core.generate_uuid('BEG','SURFACE_SITE',lat || ',' || long) AS
surface_site_id,
    dsetUUID,
    top_ft_6/3.280839895,
    bottom_ft_6/3.280839895,
    interval_ft_6/3.280839895,
    watrvol_6_bbl,
    bhp_6,
    durtn_hrs_6,
    prod_reservoir_6
FROM staging.beg_well_data
WHERE top_ft_6 is not null and bottom_ft_6 is not null;

```

```

INSERT INTO core.interval (surface_site_id, dataset_id,top_depth, bottom_depth,
interval_length, water_volume, bhp, duration_hours, producing_reservoir)
SELECT distinct
    core.generate_uuid('BEG','SURFACE_SITE',lat || ',' || long) AS
surface_site_id,
    dsetUUID,
    top_ft_7/3.280839895,
    bottom_ft_7/3.280839895,
    interval_ft_7/3.280839895,
    watrvol_7_bbl,
    bhp_7,
    durtn_hrs_7,
    prod_reservoir_7
FROM staging.beg_well_data

```

```

WHERE top_ft_7 is not null and bottom_ft_7 is not null;

INSERT INTO core.interval (surface_site_id, dataset_id,top_depth, bottom_depth,
interval_length, water_volume, bhp, duration_hours, producing_reservoir)
SELECT distinct
    core.generate_uuid('BEG','SURFACE_SITE',lat || ','|| long) AS
surface_site_id,
    dsetUUID,
    top_ft_8/3.280839895,
    bottom_ft_8/3.280839895,
    interval_ft_8/3.280839895,
    watrvol_8_bbl,
    bhp_8,
    durtn_hrs_8,
    prod_reservoir_8
FROM staging.beg_well_data
WHERE top_ft_8 is not null and bottom_ft_8 is not null;

-- update materialized views in staging:
perform staging.refresh_matview('staging.beg_well_view_materialized');
perform core.update_all_matviews();

RETURN 0;
END;
$_$;

ALTER FUNCTION staging.beg_data_transformationx() OWNER TO postgres;

--
-- Name: beg_reservoir_transformationx(); Type: FUNCTION; Schema: staging; Owner:
postgres
--

CREATE FUNCTION beg_reservoir_transformationx() RETURNS integer
    LANGUAGE plpgsql
    AS $$

DECLARE
    dsetUUID uuid;
BEGIN
    RAISE INFO 'PostgreSQL is now starting staging.beg_reservoir_transformationx';
    select core.register_dataset('BEG','RESERVOIR',NULL) into dsetUUID;

    RAISE INFO 'PostgreSQL is now inserting into core.reservoir';

```

```

INSERT INTO core.reservoir
select distinct
core.generate_uuid('BEG','RESERVOIR',reservoirname),
dsetUUID,
reservoiruri,
reservoirname,
label,
otherid,
othername,
description,
county,
state,
locationname,
latdegree,
longdegree,
srs,
locationuncertaintystatement,
temperaturemax_f,
temperaturemin_f,
depthtotop_ft,
depthtobottom_ft,
dominantformationname,
otherformationname,
dominantlithologyname,
otherlithologyname,
area_acres,
width_ft,
length_ft,
sandvolume_cft,
blockvolume_cft,
percentsand,
averagenetsandthickness_ft,
wellcount,
faultcount,
faulted,
faulttype,
faulthydrauliccharacter,
firstproductiondate,
lastproductiondate,
yearsinproduction,
monthlyproduction_bbl,
productionreportdate,
averagemonthlywaterproduction_bbl,
minimummonthlywaterproduction_bbl,
maximummonthlywaterproduction_bbl,
waterinjection_bbl,

```

```

watersaturation_pct,
watertds_mgl,
averagefieldwidepressure_psi,
averagefieldwidegradient_degfft,
porositymax_pct,
porositymin_pct,
permeabilitymax_md,
permeabilitymin_md,
cementationmineralogy,
trap,
drive,
gravity,
commodityofinterest,
stimulation,
averageperforatedthickness_ft,
relatedresource,
source,
metadadatauri,
majorfields,
GeometryFromText('POLYGON((' ||
    cordinate_nw_lat || ' ' || cordinate_nw_long || ' ' ||
    cordinate_ne_lat || ' ' || cordinate_ne_long || ' ' ||
    cordinate_se_lat || ' ' || cordinate_se_long || ' ' ||
    cordinate_sw_lat || ' ' || cordinate_sw_long || ' ' ||
    cordinate_nw_lat || ' ' || cordinate_nw_long ||
    '))', 4326) as shape
FROM staging.beg_reservoir_data;
RAISE INFO 'PostgreSQL has completed inserting into core.reservoir';
--UPDATE core.surface_site SET shape = geometryfromtext('POINT(' || longitude || ' ' ||
latitude || ')', 4326) WHERE dataset_id = dsetUUID;
--UPDATE core.surface_site SET county_id=c.id FROM core.ref_county c WHERE
ST_Contains(c.geometry,shape) and dataset_id = dsetUUID;

-- update materialized views in staging
RAISE INFO 'PostgreSQL is now refreshing staging.beg_reservoir_view_materialized';
perform staging.refresh_matview('staging.beg_reservoir_view_materialized');
RAISE INFO 'PostgreSQL has completed refreshing staging.beg_reservoir_view_materialized';

--perform core.update_all_matviews();
RAISE INFO 'PostgreSQL is now refreshing core.template_reservoir_materialized';
perform core.refresh_matview('core.template_reservoir_materialized');
RAISE INFO 'PostgreSQL has completed refreshing core.template_reservoir_materialized';

RAISE INFO 'PostgreSQL has completed staging.beg_reservoir_transformationx';

RETURN 0;

```

```
END;
$$;
```

```
ALTER FUNCTION staging.beg_reservoir_transformationx() OWNER TO postgres;
```

```
--
-- Name: cornell_transformationx(); Type: FUNCTION; Schema: staging; Owner: postgres
--
```

```
CREATE FUNCTION cornell_transformationx() RETURNS integer
    LANGUAGE plpgsql
    AS $$
DECLARE
    dsetUUID uuid;
BEGIN
```

```
    select core.register_dataset('CORNELL','ALL',NULL) into dsetUUID;
```

```
    UPDATE staging.cornell_well_data SET WELL_NM=null where WELL_NM=";
```

```
    INSERT INTO core.surface_site (id, dataset_id ,api, site_name, operator, longitude,
latitude, other_location_name,field,section,basement_depth,plugged,
well_type,lithology,comment, average_heatflow)
select distinct * from (SELECT
```

```
    first_value(core.generate_uuid('CORNELL','SURFACE_SITE',API_WELLNO))
OVER (PARTITION BY API_WELLNO) AS id,
    dsetUUID AS dataset_id,
    first_value(API_WELLNO) OVER (PARTITION BY API_WELLNO)
AS api,
    first_value(WELL_NM) OVER (PARTITION BY API_WELLNO) AS
site_name,
    first_value("operator") OVER (PARTITION BY API_WELLNO) AS
operator,
    first_value(decimal_longitude) OVER (PARTITION BY
API_WELLNO) AS longitude,
    first_value(decimal_latitude) OVER (PARTITION BY API_WELLNO) AS
latitude,
    --first_value(permit_number) OVER (PARTITION BY API_WELLNO)
AS lease_name,
    first_value('town=' || town || '; municipality=' || municipality || '; farm=' ||
farm_name || '; pool=' || pool || '; quad=' || quad) OVER (PARTITION BY API_WELLNO) AS
other_location_name,
    first_value(field) OVER (PARTITION BY API_WELLNO) AS field,
```

```

        first_value(map_section) OVER (PARTITION BY API_WELLNO) AS
section,
        first_value(sed_thickness_m) OVER (PARTITION BY API_WELLNO)
AS basement_depth,
        first_value(well_status) OVER (PARTITION BY API_WELLNO) AS
plugged,
        first_value(well_type) OVER (PARTITION BY API_WELLNO) AS
well_type,
        first_value('cosuna=' || cosuna_col) OVER (PARTITION BY
API_WELLNO) AS lithology,
        first_value(data_source) OVER (PARTITION BY API_WELLNO) AS
comment,
        first_value(site_heat_flow) OVER (PARTITION BY API_WELLNO) as
average_heatflow
FROM (select * from staging.cornell_well_data order by API_WELLNO,
log_date) A) B;

```

```

UPDATE core.surface_site SET shape = geometryfromtext('POINT('|| longitude || ' ' ||
latitude ||)'), 4326) WHERE dataset_id = dsetUUID;
UPDATE core.surface_site SET county_id=c.id FROM core.ref_county c WHERE
ST_Contains(c.geometry,shape) and dataset_id = dsetUUID;

```

```

INSERT INTO core.formation (surface_site_id, dataset_id, formation,formation_depth)
SELECT distinct
core.generate_uuid('CORNELL','SURFACE_SITE',API_WELLNO), dsetUUID,
tdform, tvd_m
FROM staging.cornell_well_data where length(tdform)>0;

```

```

INSERT INTO core.well_dates ( surface_site_id, dataset_id, compl_date,
elevation)
SELECT distinct
core.generate_uuid('CORNELL','SURFACE_SITE',API_WELLNO), dsetUUID,
comp_date,
elevation/3.280839895 -- converts to meters
FROM staging.cornell_well_data where (comp_date is not null) or (elevation is
not null);

```

```

INSERT INTO core.well_info ( surface_site_id, dataset_id, permit)
SELECT distinct
core.generate_uuid('CORNELL','SURFACE_SITE',API_WELLNO),
dsetUUID,
permit_number

```

```
FROM staging.cornell_well_data where (permit_number is not null) AND
(length(permit_number)>0);
```

```
INSERT INTO core.depth (surface_site_id, dataset_id, depth_date, true_vertical_depth)
SELECT distinct
core.generate_uuid('CORNELL','SURFACE_SITE',API_WELLNO), dsetUUID,
td_date, tvd_m
FROM staging.cornell_well_data;
```

```
INSERT INTO core.bht_measurements (id,surface_site_id, dataset_id,
bht,bht_corrected,drill_date,bht_depth, bht_correction_type)
SELECT distinct * FROM
(SELECT
core.generate_uuid('CORNELL','BHT',API_WELLNO || ';' ||
coalesce(log_depth_m,0) || ';' || row_number() OVER (partition by API_WELLNO,
log_depth_m)) AS id,
core.generate_uuid('CORNELL','SURFACE_SITE',API_WELLNO) AS
surface_site_id,
dsetUUID,
bht_c AS bht,
harrison_bht_c AS bht_corrected,
log_date AS drill_date,
log_depth_m AS bht_depth,
cast('Harrison' as varchar(16)) as bht_correction_type
FROM staging.cornell_well_data order by API_WELLNO, log_depth_m) Z;
```

```
INSERT INTO core.thermal_conductivity (id,surface_site_id, dataset_id,conductivity)
SELECT distinct * FROM
(SELECT
core.generate_uuid('CORNELL','THERMAL_CONDUCTIVITY',API_WELLNO || ';' ||
coalesce(log_depth_m,0) || ';' || row_number() OVER (partition by API_WELLNO,
log_depth_m)) AS id,
core.generate_uuid('CORNELL','SURFACE_SITE',API_WELLNO) AS
surface_site_id,
dsetUUID,
average_conductivity
FROM staging.cornell_well_data
where (average_conductivity is not null)
order by API_WELLNO, log_depth_m) Z;
```

```
INSERT INTO core.heat_flow (id,
bht_measurements_id,thermal_conductivity_id,surface_site_id,dataset_id, gradient,
gradient_corrected, heat_flow,
interval_bottom_depth, interval_order, quality)
```

```

SELECT distinct * FROM
(SELECT
    core.generate_uuid('CORNELL','HEAT_FLOW',API_WELLNO || ';' ||
coalesce(log_depth_m,0) || ';' || row_number() OVER (partition by API_WELLNO,
log_depth_m)) AS id,
    core.generate_uuid('CORNELL','BHT',API_WELLNO || ';' ||
coalesce(log_depth_m,0) || ';' || row_number() OVER (partition by API_WELLNO,
log_depth_m)) AS bht_measurements_id,

    core.generate_uuid('CORNELL','THERMAL_CONDUCTIVITY',API_WELLNO || ';' ||
coalesce(log_depth_m,0) || ';' || row_number() OVER (partition by API_WELLNO,
log_depth_m)) AS thermal_conductivity_id,
    core.generate_uuid('CORNELL','SURFACE_SITE',API_WELLNO) AS
surface_site_id,

    dsetUUID,
    uncorr_grad AS gradient,
    harrison_corr_grad_c_km AS gradient_corrected,
    heat_flow_mw_m2 AS heat_flow,
    log_depth_m as interval_bottom_depth,
    interval_id as interval_order,
    interval_quality_code as quality
FROM staging.cornell_well_data order by API_WELLNO, log_depth_m) Z;

```

```

INSERT INTO core.bht_measurements (id,surface_site_id, dataset_id,
bht,bht_depth,estimated)
(SELECT
    core.generate_uuid('CORNELL','BHT',API_WELLNO || ';' || 1500' || ';' ||
random()) AS id,
    core.generate_uuid('CORNELL','SURFACE_SITE',API_WELLNO) AS
surface_site_id,

    dsetUUID,
    temp_est_1500m AS bht,
    1500 AS bht_depth,
    true as estimated
FROM staging.cornell_well_data);

```

```

INSERT INTO core.bht_measurements (id,surface_site_id, dataset_id,
bht,bht_depth,estimated)
(SELECT
    core.generate_uuid('CORNELL','BHT',API_WELLNO || ';' || 3000' || ';' ||
random()) AS id,
    core.generate_uuid('CORNELL','SURFACE_SITE',API_WELLNO) AS
surface_site_id,

    dsetUUID,
    temp_est_3000m AS bht,

```



```

        3000 AS bht_depth,
        true as estimated
    FROM staging.cornell_well_data);

    INSERT INTO core.bht_measurements (id,surface_site_id, dataset_id,
    bht,bht_depth,estimated)
        (SELECT
            core.generate_uuid('CORNELL','BHT',API_WELLNO || ';' || 4500 || ';' ||
random()) AS id,
            core.generate_uuid('CORNELL','SURFACE_SITE',API_WELLNO) AS
surface_site_id,
            dsetUUID,
            temp_est_4500m AS bht,
            4500 AS bht_depth,
            true as estimated
        FROM staging.cornell_well_data);

    INSERT INTO core.bht_measurements (id,surface_site_id, dataset_id,
    bht,bht_depth,estimated)
        (SELECT
            core.generate_uuid('CORNELL','BHT',API_WELLNO || ';' || 6000 || ';' ||
random()) AS id,
            core.generate_uuid('CORNELL','SURFACE_SITE',API_WELLNO) AS
surface_site_id,
            dsetUUID,
            temp_est_6000m AS bht,
            6000 AS bht_depth,
            true as estimated
        FROM staging.cornell_well_data);

    INSERT INTO core.bht_measurements (id,surface_site_id, dataset_id,
    bht,bht_depth,estimated)
        (SELECT
            core.generate_uuid('CORNELL','BHT',API_WELLNO || ';' || 7500 || ';' ||
random()) AS id,
            core.generate_uuid('CORNELL','SURFACE_SITE',API_WELLNO) AS
surface_site_id,
            dsetUUID,
            temp_est_7500m AS bht,
            7500 AS bht_depth,
            true as estimated
        FROM staging.cornell_well_data);

    INSERT INTO core.bht_measurements (id,surface_site_id, dataset_id,
    bht,bht_depth,estimated)
        (SELECT

```

```

        core.generate_uuid('CORNELL','BHT',API_WELLNO || ';' ||
random()) AS id,
        core.generate_uuid('CORNELL','SURFACE_SITE',API_WELLNO) AS
surface_site_id,
        dsetUUID,
        temp_est_10000m AS bht,
        10000 AS bht_depth,
        true as estimated
FROM staging.cornell_well_data);

INSERT INTO core.bht_measurements (id,surface_site_id, dataset_id,
bht,bht_depth,estimated)
(SELECT
        core.generate_uuid('CORNELL','BHT',API_WELLNO || ';to150' || ';' ||
random()) AS id,
        core.generate_uuid('CORNELL','SURFACE_SITE',API_WELLNO) AS
surface_site_id,
        dsetUUID,
        150 AS bht,
        est_depth_150c AS bht_depth,
        true as estimated
FROM staging.cornell_well_data);

INSERT INTO core.bht_measurements (id,surface_site_id, dataset_id,
bht,bht_depth,estimated)
(SELECT
        core.generate_uuid('CORNELL','BHT',API_WELLNO || ';to80' || ';' ||
random()) AS id,
        core.generate_uuid('CORNELL','SURFACE_SITE',API_WELLNO) AS
surface_site_id,
        dsetUUID,
        80 AS bht,
        est_depth_80c AS bht_depth,
        true as estimated
FROM staging.cornell_well_data);

-- update materialized views in staging
perform staging.refresh_matview('staging.cornell_well_view_materialized');
perform core.update_all_matviews();

RETURN 0;
END;

$$;

```

```

ALTER FUNCTION staging.cornell_transformationx() OWNER TO postgres;

--
-- Name: create_matview(name, name); Type: FUNCTION; Schema: staging; Owner: postgres
--

CREATE FUNCTION create_matview(name, name) RETURNS void
  LANGUAGE plpgsql SECURITY DEFINER
  AS $_$
DECLARE
  matview ALIAS FOR $1;
  view_name ALIAS FOR $2;
  entry staging.matviews%ROWTYPE;
BEGIN
  SELECT * INTO entry FROM staging.matviews WHERE mv_name = matview;

  IF FOUND THEN
    RAISE EXCEPTION 'Materialized view "%" already exists.',
      matview;
  END IF;

  EXECUTE 'REVOKE ALL ON ' || view_name || ' FROM postgres';
  EXECUTE 'GRANT SELECT ON ' || view_name || ' TO postgres';
  EXECUTE 'CREATE TABLE ' || matview || ' AS SELECT * FROM ' || view_name;
  EXECUTE 'REVOKE ALL ON ' || matview || ' FROM postgres';
  EXECUTE 'GRANT SELECT ON ' || matview || ' TO postgres';

  INSERT INTO staging.matviews (mv_name, v_name, last_refresh)
    VALUES (matview, view_name, CURRENT_TIMESTAMP);

  RETURN;
END
$_$;

ALTER FUNCTION staging.create_matview(name, name) OWNER TO postgres;

--
-- Name: drop_matview(name); Type: FUNCTION; Schema: staging; Owner: postgres
--

CREATE FUNCTION drop_matview(name) RETURNS void
  LANGUAGE plpgsql SECURITY DEFINER
  AS $_$
DECLARE
  matview ALIAS FOR $1;

```

```

    entry staging.matviews%ROWTYPE;
BEGIN
    SELECT * INTO entry FROM staging.matviews WHERE mv_name = matview;

    IF NOT FOUND THEN
        RAISE EXCEPTION 'Materialized view % does not exist.', matview;
    END IF;

    EXECUTE 'DROP TABLE ' || matview;
    DELETE FROM staging.matviews WHERE mv_name=matview;

    RETURN;
END
$_$;

ALTER FUNCTION staging.drop_matview(name) OWNER TO postgres;

--
-- Name: mlkay_bht_transformationx(); Type: FUNCTION; Schema: staging; Owner: postgres
--

CREATE FUNCTION mlkay_bht_transformationx() RETURNS integer
    LANGUAGE plpgsql
    AS $$

DECLARE
    dsetUUID uuid;
BEGIN

    select core.register_dataset('MLKay','WB',NULL) into dsetUUID;

    INSERT INTO core.bht_measurements (id, surface_site_id, dataset_id, bht,bht_depth,
drill_date, time_since_circulation)
        SELECT distinct
            core.generate_uuid('MLKay','WB',s.api || ';' ||
(drillertotaldepth/3.280839895) || ';' || coalesce(to_char(measurementdatetime,'YYYY-MM-DD'),"")) AS id,
            s.id AS surface_site_id,
            dsetUUID as dataset_id,
            (measuredtemperaturef-32)*5/9 AS bht, -- converts to C
            drillertotaldepth/3.280839895 AS bht_depth, -- coverts to meters
            measurementdatetime AS drill_date,
            circulationduration AS time_since_circulation
        FROM core.surface_site s , (select distinct * from staging.mlk_bht_data) b

```

```
where s.api=('42' || lpad('' || b.api42, 8,'0') || '0000')
and s.dataset_id=dsetUUID;
```

```
INSERT INTO core.casing (surface_site_id, dataset_id, casing_size_inches)
SELECT distinct
    core.generate_uuid('MLKay','WB',s.api) AS surface_site_id,
    dsetUUID,
    casingpipediameter
FROM core.surface_site s , (select distinct * from staging.mlk_bht_data) b
where s.api=('42' || lpad('' || b.api42, 8,'0') || '0000')
and s.dataset_id=dsetUUID;
```

```
INSERT INTO core.depth (surface_site_id, dataset_id, depth, depth_date, bit_diameter,
mud_weight, viscosity )
SELECT distinct
    core.generate_uuid('MLKay','WB',s.api) AS surface_site_id,
    dsetUUID,
    drillertotaldepth/3.280839895 AS bht_depth, -- coverts to meters
    measurementdatetime AS drilldate,
    bitdiametercollar as bit_diameter,
    mud_weight as mud_weight,
    viscosity as viscosity
FROM core.surface_site s , (select distinct * from staging.mlk_bht_data) b
where s.api=('42' || lpad('' || b.api42, 8,'0') || '0000')
and s.dataset_id=dsetUUID;
```

```
RETURN 0;
END;
$$;
```

```
ALTER FUNCTION staging.mlkay_bht_transformationx() OWNER TO postgres;
```

```
--
-- Name: mlkay_uic_transformationx(); Type: FUNCTION; Schema: staging; Owner: postgres
--
```

```
CREATE FUNCTION mlkay_uic_transformationx() RETURNS integer
    LANGUAGE plpgsql
    AS $$
```

```
DECLARE
    dsetUUID uuid;
BEGIN
```

```

select core.register_dataset('MLKay','WB',NULL) into dsetUUID;

INSERT INTO core.fluid (id, surface_site_id, dataset_id, cntl_no, lease_id, well_no,
field_no, oper, uic_class, bbl_vol_inj, max_inj_pressure,
                    top_inj_zone, bot_inj_zone, w_status, sw, fw, bw, air, gas, lpg, co2,
polymer, cycle_yr, fw_avg_or_max, fw_bbls_day)
SELECT distinct
core.generate_uuid('MLKay','WB','42' || lpad('' || r.api_number, 8,'0') || '0000;' ||
r.cntl_no) AS id,
core.generate_uuid('MLKay','WB','42' || lpad('' || r.api_number, 8,'0') || '0000') AS
surface_site_id,
dsetUUID as dataset_id,
r.cntl_no,
r.lease_id,
r.well_no,
r.field_no,
r.oper,
r.uic_class,
r.bbl_vol_inj,
r.max_inj_pressure,
r.top_inj_zone,
r.bot_inj_zone,
m.w_status,
m.sw,
m.fw,
m.bw,
m.air,
m.gas,
m.lpg,
m.co2,
m.polymer,
m.cycle_yr,
m.fw_avg_or_max,
m.fw_bbls_day
FROM uic.uic_root r, core.surface_site s, uic.uic_monitor m
where s.dataset_id=dsetUUID and s.api=('42' || lpad('' || r.api_number, 8,'0') ||
'0000') and r.cntl_no=m.cntl_no;

INSERT INTO core.fluid_h10 (injection_id,surface_site_id, dataset_id, h10_date,
h10_avg_inj_pressure, h10_max_inj_pressure, h10_total_vol_bbl, h10_total_vol_mcf)
SELECT distinct
core.generate_uuid('MLKay','WB','42' || lpad('' || r.api_number, 8,'0') || '0000;' ||
r.cntl_no) AS injection_id,
core.generate_uuid('MLKay','WB','42' || lpad('' || r.api_number, 8,'0') || '0000') AS
surface_site_id,
dsetUUID as dataset_id,

```

```

        h10_date,
        h10_avg_inj_pressure,
        h10_max_inj_pressure,
        h10_total_vol_bbl,
        h10_total_vol_mcf
    FROM uic.uic_root r, core.surface_site s, uic.uic_h10 h10
    where s.dataset_id=dsetUUID and s.api=('42' || lpad('' || r.api_number, 8,'0') ||
'0000') and r.cntl_no=h10.cntl_no;

```

```

        INSERT INTO core.fluid_h10h (injection_id,surface_site_id, dataset_id, h10h_date,
h10h_max_hydrocarb_psig, h10h_max_brine_psig)
        SELECT distinct
        core.generate_uuid('MLKay','WB','42' || lpad('' || r.api_number, 8,'0') || '0000;' ||
r.cntl_no) AS injection_id,
        core.generate_uuid('MLKay','WB','42' || lpad('' || r.api_number, 8,'0') || '0000') AS
surface_site_id,
        dsetUUID as dataset_id,
        h10h_date,
        h10h_max_hydrocarb_psig,
        h10h_max_brine_psig
    FROM uic.uic_root r, core.surface_site s, uic.uic_h10h h10h
    where s.dataset_id=dsetUUID and s.api=('42' || lpad('' || r.api_number, 8,'0') ||
'0000') and r.cntl_no=h10h.cntl_no;

```

```

-- --we don't need these, since no new surface sites are added --
--UPDATE core.surface_site SET shape = geometryfromtext('POINT(' || longitude || ' ' || latitude
||)'), 4326) WHERE dataset_id = dsetUUID;
--UPDATE core.surface_site SET county_id=c.id FROM core.ref_county c WHERE
ST_Contains(c.geometry,shape) and dataset_id = dsetUUID;

```

```

RETURN 0;
END;
$$;

```

```

ALTER FUNCTION staging.mlkay_uic_transformationx() OWNER TO postgres;

```

```

--
-- Name: mlkay_wb_transformationx(); Type: FUNCTION; Schema: staging; Owner: postgres
--

```

```

CREATE FUNCTION mlkay_wb_transformationx() RETURNS integer
    LANGUAGE plpgsql

```

```

AS $$

DECLARE
    dsetUUID uuid;
BEGIN

    select core.register_dataset('MLKay','WB',NULL) into dsetUUID;

    INSERT INTO core.surface_site (id, dataset_id ,api, district,
    plugged,fresh_water_converted,shut_in_date, block,section,
                                feet_from_survey,
    latitude,longitude,lease_name,section_block_survey,hole_open, contributor)
    SELECT distinct
    core.generate_uuid('MLKay','WB','42' || lpad('' || nl.api_number, 8,'0') || '0000')
AS id,
    dsetUUID as dataset_id,
    '42' || lpad('' || nl.api_number, 8,'0') || '0000' as api,
    r.field_district as district,
    r.plug_flag as plugged,
    case when r.fresh_water_flag='Y' then 1 else 0 end as fresh_water_converted,
    r.shut_in_date as shut_in_date,
    nl.block_number as block,
    nl.section as section,
    nl.feet_from_sur_sect_1 as feet_from_survey,
    nl.wgs84_latitude as latitude,
    -1*nl.wgs84_longitude as longitude,
    ol.lease_name as lease_name,
    ol.sec_blk_survey_loc as sec_block_survey,
    case when z.open_hole_code='OH' then 1 else 0 end as hole_open,
    case when (-1*nl.wgs84_longitude > 97.15) then 'Bureau of Economic Geology'
else 'MLKay Technologies' end as contributor
    FROM wb.wb_new_location_seg nl
    LEFT JOIN wb.wb_root r ON r.api_number=nl.api_number
    LEFT JOIN wb.wb_old_location_seg ol ON r.api_number=ol.api_number
    LEFT JOIN (select distinct api_number, open_hole_code from wb.wb_perf_seg
where open_hole_code='OH') Z ON r.api_number=z.api_number;

    INSERT INTO core.depth (surface_site_id, dataset_id, depth_date,depth)
    SELECT distinct
    core.generate_uuid('MLKay','WB','42' || lpad('' || r.api_number, 8,'0') || '0000') AS
surface_site_id,
    dsetUUID as dataset_id,
    r.orig_compl_date as completion_date,
    r.total_depth/3.280839895 as depth -- converts to meters
    FROM wb.wb_new_location_seg nl

```



```
LEFT JOIN wb.wb_root r ON r.api_number=nl.api_number
where r.total_depth>0;
```

```
INSERT INTO core.formation (surface_site_id, dataset_id, formation,formation_depth)
SELECT distinct
    core.generate_uuid('MLKay','WB','42' || lpad('' || nl.api_number, 8,'0') || '0000')
AS id,
    dsetUUID as dataset_id,
    fs.formation_name,
    fs.formation_depth/3.280839895 -- converts to meters
FROM wb.wb_new_location_seg nl
INNER JOIN wb.wb_formation_seg fs ON nl.api_number=fs.api_number;
```

```
INSERT INTO core.casing (surface_site_id, dataset_id, casing_size_inches,
casing_depth, casing_weight, casing_top,casing_left,file_date)
SELECT distinct
    core.generate_uuid('MLKay','WB','42' || lpad('' || nl.api_number, 8,'0') ||
'0000') AS surface_site_id,
    dsetUUID,
    cs.casing_size_data,
    cs.casing_depth_set,
    cs.casing_weight_lbs_ft[1],
    cs.top_of_cement_casing,
    cs.amount_casing_left,
    file_date
FROM (wb.wb_new_location_seg nl INNER JOIN wb.wb_casing_seg cs ON
nl.api_number=cs.api_number);
```

```
INSERT INTO core.well_info (surface_site_id, dataset_id, commodity, well_id,
permit)
SELECT distinct
    core.generate_uuid('MLKay','WB','42' || lpad('' || seg.api_number, 8,'0') || '0000')
AS surface_site_id,
    dsetUUID as dataset_id,
    oil as commodity,
    case when oil='O' then (seg.oil_district || '-' || seg.oil_lse_number || ' ' ||
seg.oil_well_number) else cast(gas_rrcid as text) end as well_id,
    permit_number as permit
FROM wb.wb_new_location_seg nl INNER JOIN wb.wb_well_id_seg seg ON
nl.api_number=seg.api_number;
```

```
INSERT INTO core.well_dates ( surface_site_id, dataset_id, compl_date,
drl_compl_date, water_injection_nbr, elevation, elevation_code)
```

```

SELECT distinct
  core.generate_uuid('MLKay','WB','42' || lpad('' || seg.api_number, 8,'0') || '0000')
AS surface_site_id,
  dsetUUID as dataset_id,
  seg.compl_date,
  seg.drl_compl_date,
  seg.water_injection_nbr,
  seg.elevation/3.280839895, -- converts to meters
  seg.elevation_code
from wb.wb_file_date_seg seg, wb.wb_new_location_seg nl
where nl.api_number=seg.api_number
and (elevation >0 OR water_injection_nbr!=null OR compl_date is not null OR
drl_compl_date is not null);

```

```

INSERT INTO core.well_test (surface_site_id, dataset_id, well_info, g10_id,
g10_rrcid, g10_tst_date, g10_dly_gas, g10_dly_cond, g10_dly_wtr,
                                g10_siwh_press, g10_shut_cd, g10_flow_press, g10_bhp,
g10_shut_press, g10_opr, g10_wh_pres_test_date)
SELECT distinct
  core.generate_uuid('MLKay','WB','42' || lpad('' || W.api_number, 8,'0') ||
'0000') AS surface_site_id,
  dsetUUID,
  gas_rrcid,
  G.g10_id,
  G.g10_rrcid,
  G.g10_tst_date,
  G.g10_dly_gas,
  G.g10_dly_cond,
  G.g10_dly_wtr,
  G.g10_siwh_press,
  G.g10_shut_cd,
  G.g10_flow_press,
  G.g10_bhp,
  G.g10_shut_press,
  g.g10_opr,
  G.g10_wh_pres_test_date
  from wb.gse10m G, (select distinct seg.gas_rrcid, seg.api_number from
wb.wb_well_id_seg seg, wb.wb_new_location_seg nl where gas='G' and
nl.api_number=seg.api_number) as W
  where G.g10_rrcid=W.gas_rrcid;

```

```

UPDATE core.surface_site SET shape = geometryfromtext('POINT(' || longitude || ' ' || latitude
||)'), 4326) WHERE dataset_id = dsetUUID;

```

```
UPDATE core.surface_site SET county_id=c.id FROM core.ref_county c WHERE
ST_Contains(c.geometry,shape) and dataset_id = dsetUUID;
```

```
RETURN 0;
END;
$$;
```

```
ALTER FUNCTION staging.mlkay_wb_transformationx() OWNER TO postgres;
```

```
--
-- Name: recreate_all_matviews(); Type: FUNCTION; Schema: staging; Owner: postgres
--
```

```
CREATE FUNCTION recreate_all_matviews() RETURNS integer
    LANGUAGE plpgsql
    AS $$
BEGIN
```

```
-- recreate all materialized views in core:
```

```
RAISE INFO 'PostgreSQL is now dropping staging.beg_reservoir_view_materialized';
perform staging.drop_matview('staging.beg_reservoir_view_materialized');
RAISE INFO 'staging.beg_reservoir_view_materialized has been dropped.';
```

```
RAISE INFO 'PostgreSQL is now dropping staging.beg_well_view_materialized';
perform staging.drop_matview('staging.beg_well_view_materialized');
RAISE INFO 'staging.beg_well_view_materialized has been dropped.';
```

```
RAISE INFO 'PostgreSQL is now dropping staging.cornell_well_view_materialized';
perform staging.drop_matview('staging.cornell_well_view_materialized');
RAISE INFO 'staging.cornell_well_view_materialized has been dropped.';
```

```
RAISE INFO 'PostgreSQL is now dropping staging.smu_bht_view_materialized';
perform staging.drop_matview('staging.smu_bht_view_materialized');
RAISE INFO 'staging.smu_bht_view_materialized has been dropped.';
```

```
RAISE INFO 'PostgreSQL is now dropping staging.smu_hf_view_materialized';
perform staging.drop_matview('staging.smu_hf_view_materialized');
RAISE INFO 'staging.smu_hf_view_materialized has been dropped.';
```

```
RAISE INFO 'PostgreSQL is now dropping staging.tt_well_view_materialized';
perform staging.drop_matview('staging.tt_well_view_materialized');
RAISE INFO 'staging.tt_well_view_materialized has been dropped.';
```

```
RAISE INFO 'PostgreSQL is now dropping staging.und_rad_view_materialized';
perform staging.drop_matview('staging.und_rad_view_materialized');
```

RAISE INFO 'staging.und_rad_view_materialized has been dropped.';

RAISE INFO 'PostgreSQL is now dropping staging.und_tc_view_materialized';
perform staging.drop_matview('staging.und_tc_view_materialized');
RAISE INFO 'staging.und_tc_view_materialized has been dropped.';

RAISE INFO 'PostgreSQL is now dropping staging.und_td_view_materialized';
perform staging.drop_matview('staging.und_td_view_materialized');
RAISE INFO 'staging.und_td_view_materialized has been dropped.';

RAISE INFO 'PostgreSQL is now creating matview staging.beg_reservoir_view_materialized';
perform
staging.create_matview('staging.beg_reservoir_view_materialized','staging.beg_reservoir_view'
);
RAISE INFO 'staging.beg_reservoir_view_materialized has been created';

RAISE INFO 'PostgreSQL is now creating matview staging.beg_well_view_materialized';
perform
staging.create_matview('staging.beg_well_view_materialized','staging.beg_well_view');
RAISE INFO 'staging.beg_well_view_materialized has been created';

RAISE INFO 'PostgreSQL is now creating matview staging.cornell_well_view_materialized';
perform
staging.create_matview('staging.cornell_well_view_materialized','staging.cornell_well_view');
RAISE INFO 'staging.cornell_well_view_materialized has been created';

RAISE INFO 'PostgreSQL is now creating matview staging.smu_bht_view_materialized';
perform staging.create_matview('staging.smu_bht_view_materialized','staging.smu_bht_view');
RAISE INFO 'staging.smu_bht_view_materialized has been created';

RAISE INFO 'PostgreSQL is now creating matview staging.smu_hf_view_materialized';
perform staging.create_matview('staging.smu_hf_view_materialized','staging.smu_hf_view');
RAISE INFO 'staging.smu_hf_view_materialized has been created';

RAISE INFO 'PostgreSQL is now creating matview staging.tt_well_view_materialized';
perform staging.create_matview('staging.tt_well_view_materialized','staging.tt_well_view');
RAISE INFO 'staging.tt_well_view_materialized has been created';

RAISE INFO 'PostgreSQL is now creating matview staging.und_rad_view_materialized';
perform staging.create_matview('staging.und_rad_view_materialized','staging.und_rad_view');
RAISE INFO 'staging.und_rad_view_materialized has been created';

RAISE INFO 'PostgreSQL is now creating matview staging.und_tc_view_materialized';
perform staging.create_matview('staging.und_tc_view_materialized','staging.und_tc_view');

```

RAISE INFO 'staging.und_tc_view_materialized has been created';

RAISE INFO 'PostgreSQL is now creating matview staging.und_td_view_materialized';
perform staging.create_matview('staging.und_td_view_materialized','staging.und_td_view');
RAISE INFO 'staging.und_td_view_materialized has been created';


RETURN 0;
END;
$$;


ALTER FUNCTION staging.recreate_all_matviews() OWNER TO postgres;

--
-- Name: refresh_matview(name); Type: FUNCTION; Schema: staging; Owner: postgres
--

CREATE FUNCTION refresh_matview(name) RETURNS void
  LANGUAGE plpgsql SECURITY DEFINER
  AS $_$
DECLARE
  matview ALIAS FOR $1;
  entry staging.matviews%ROWTYPE;
BEGIN

  SELECT * INTO entry FROM staging.matviews WHERE mv_name = matview;

  IF NOT FOUND THEN
    RAISE EXCEPTION 'Materialized view % does not exist.', matview;
  END IF;

  EXECUTE 'DELETE FROM ' || matview;
  EXECUTE 'INSERT INTO ' || matview || ' SELECT * FROM ' || entry.v_name;

  UPDATE staging.matviews
    SET last_refresh=CURRENT_TIMESTAMP
    WHERE mv_name=matview;

  RETURN;
END
$_$;


ALTER FUNCTION staging.refresh_matview(name) OWNER TO postgres;

```

```

--
-- Name: smu_bht_transformationx(); Type: FUNCTION; Schema: staging; Owner: postgres
--

CREATE FUNCTION smu_bht_transformationx() RETURNS integer
    LANGUAGE plpgsql
    AS $$
DECLARE
    dsetUUID uuid;
BEGIN

    select core.register_dataset('SMU','BHT',NULL) into dsetUUID;

    INSERT INTO core.surface_site (id, dataset_id ,api, field, latitude, longitude, operator,
surface_temperature,
        elevation_gl,elevation_df,elevation_kb,basement_depth, average_heatflow,
plugged, well_type,other_ids,site_name,"comment")
        select distinct * FROM (SELECT
            first_value(core.generate_uuid('SMU','SURFACE_SITE',
to_char(longitude,'999.9999') || to_char(latitude,'999.9999')) OVER (PARTITION BY
latitude,longitude) AS id,
            dsetUUID as dataset_id,
            first_value(api) OVER (PARTITION BY latitude,longitude) AS
api,
            first_value(field_name) OVER (PARTITION BY
latitude,longitude) as field_name,
            first_value(latitude) OVER (PARTITION BY latitude,longitude)
as latitude,
            first_value(longitude) OVER (PARTITION BY
latitude,longitude) as longitude,
            first_value(company_name) OVER (PARTITION BY
latitude,longitude) as operator,
            first_value(surface_temp ) OVER (PARTITION BY
latitude,longitude) as surface_temp,
            first_value(case when elevation_measured='GL' then elevation_m
else null end) OVER (PARTITION BY latitude,longitude) as elevation_gl,
            first_value(case when elevation_measured='DF' then elevation_m
else null end) OVER (PARTITION BY latitude,longitude) as elevation_df,
            first_value(case when elevation_measured='KB' then elevation_m
else null end) OVER (PARTITION BY latitude,longitude) as elevation_kb,
            first_value(basement_depth ) OVER (PARTITION BY
latitude,longitude) as basement_depth,
            first_value(well_ave_hf ) OVER (PARTITION BY
latitude,longitude) as average_heatflow,
            first_value(well_status) OVER (PARTITION BY
latitude,longitude) as plugged,

```

```

        first_value(well_type ) OVER (PARTITION BY
latitude,longitude) as well_type,
        first_value('permit ' || permit || '; FileNumber: ' || file_number || ';
OperationName: ' || operation_name) OVER (PARTITION BY latitude,longitude) as other_ids,
        first_value(split_part(surface_interval_id,'-',1) || '-BHT-' ||
split_part(surface_interval_id,'-',3)) OVER (PARTITION BY latitude,longitude) as site_name,
        first_value(reference_data_set) OVER (PARTITION BY
split_part(surface_interval_id,'-',1),split_part(surface_interval_id,'-',3)) as original_source
FROM staging.smu_bht_data) Z;

```

```

UPDATE core.surface_site SET shape = geometryfromtext('POINT('|| longitude || ' ' ||
latitude ||)'), 4326) WHERE dataset_id = dsetUUID;
UPDATE core.surface_site SET county_id=c.id FROM core.ref_county c WHERE
ST_Contains(c.geometry,shape) and dataset_id = dsetUUID;

```

```

INSERT INTO core.formation (surface_site_id, dataset_id, formation,formation_depth)
SELECT distinct
        core.generate_uuid('SMU','SURFACE_SITE', to_char(longitude,'999.9999') ||
to_char(latitude,'999.9999')),
        dsetUUID as dataset_id,
        formation,
        depth
FROM staging.smu_bht_data
where length(formation)>0;

```

```

INSERT INTO core.thermal_conductivity (id,surface_site_id, dataset_id,conductivity,
resistance, lithology_info)
SELECT
        core.generate_uuid('SMU','THERMAL_CONDUCTIVITY',
to_char(longitude,'999.9999') || to_char(latitude,'999.9999') || ';' || coalesce(depth,0) || ';' ||
        row_number() OVER (partition by latitude, longitude,
depth)) AS id,
        core.generate_uuid('SMU','SURFACE_SITE',
to_char(longitude,'999.9999') || to_char(latitude,'999.9999')) AS surface_site_id,
        dsetUUID,
        K ,
        resistance,
        k_source
FROM staging.smu_bht_data
WHERE (K is not null) or (resistance is not null);

```

```

INSERT INTO core.bht_measurements (id,surface_site_id, dataset_id,
bht,bht_corrected,bht_depth, drill_date, date_type, time_since_circulation, bht_correction_type)

```

```

SELECT
    core.generate_uuid('SMU','BHT', to_char(longitude,'999.9999') ||
to_char(latitude,'999.9999') || ';' || coalesce(depth,0) || ';' ||
    row_number() OVER (partition by latitude, longitude, depth)) AS
id,
    core.generate_uuid('SMU','SURFACE_SITE',
to_char(longitude,'999.9999') || to_char(latitude,'999.9999')) AS surface_site_id,
    dsetUUID,
    uncor_bht AS bht,
    bhtcorrected_temp AS bht_corrected,
    depth AS bht_depth,
    drilling_complete AS drilldate,
    'Completion' as date_type,
    tsc AS time_since_circulation,
    correction AS bht_correction_type
FROM staging.smu_bht_data
WHERE (uncor_bht is not null) or (bhtcorrected_temp is not null);

INSERT INTO core.depth (surface_site_id, dataset_id,depth, depth_date)
SELECT distinct
    core.generate_uuid('SMU','SURFACE_SITE',
to_char(longitude,'999.9999') || to_char(latitude,'999.9999')) AS surface_site_id,
    dsetUUID,
    depth as depth,
    drilling_complete AS depth_date
FROM staging.smu_bht_data
WHERE (depth is not null);

INSERT INTO core.well_dates (surface_site_id,
dataset_id,drl_compl_date,drl_start_date)
SELECT distinct
    core.generate_uuid('SMU','SURFACE_SITE',
to_char(longitude,'999.9999') || to_char(latitude,'999.9999')) AS surface_site_id,
    dsetUUID,
    drilling_complete as drl_compl_date,
    drilling_start as drl_start_date
FROM staging.smu_bht_data
WHERE (depth is not null);

-- need to have bht and tcu data
INSERT INTO core.heat_flow (id,bht_measurements_id, thermal_conductivity_id,
surface_site_id, dataset_id, gradient_corrected, heat_flow, interval_order, quality,
interval_top_depth, interval_bottom_depth)
SELECT
    core.generate_uuid('SMU','HEAT_FLOW', to_char(longitude,'999.9999')
|| to_char(latitude,'999.9999') || ';' || coalesce(depth,0) || ';' ||

```



```

        row_number() OVER (partition by latitude, longitude, depth)) AS
id,
        (case when ((uncor_bht is not null) or (bhtcorrected_temp is not null))
then
        core.generate_uuid('SMU','BHT', to_char(longitude,'999.9999') ||
to_char(latitude,'999.9999') || ';' || coalesce(depth,0) || ';' ||
        row_number() OVER (partition by latitude, longitude, depth))
        else null end) AS bht_measurements_id,
        (case when ((K is not null) or (resistance is not null)) then
        core.generate_uuid('SMU','THERMAL_CONDUCTIVITY',
to_char(longitude,'999.9999') || to_char(latitude,'999.9999') || ';' || coalesce(depth,0) || ';' ||
        row_number() OVER (partition by latitude, longitude, depth))
        else NULL end) AS thermal_conductivity_id,

        core.generate_uuid('SMU','SURFACE_SITE',
to_char(longitude,'999.9999') || to_char(latitude,'999.9999')) AS surface_site_id,
        dsetUUID,
        harrison_gradient AS gradient_corrected,
        q_calculated AS heat_flow_corrected,
        split_part(surface_interval_id,'-',4) AS interval_order,
        quality as quality,
        0 as interval_top_depth,
        depth as interval_bottom_depth
FROM staging.smu_bht_data
WHERE ((harrison_gradient is not null) or (q_calculated is not null)) and ((K is
not null) or (resistance is not null))
        and ((uncor_bht is not null) or (bhtcorrected_temp is not null));

-- update materialized views in staging
perform staging.refresh_matview('staging.smu_bht_view_materialized');
perform core.update_all_matviews();

RETURN 0;
END;

$$;

ALTER FUNCTION staging.smu_bht_transformationx() OWNER TO postgres;

--
-- Name: smu_hf_transformationx(); Type: FUNCTION; Schema: staging; Owner: postgres
--

CREATE FUNCTION smu_hf_transformationx() RETURNS integer
LANGUAGE plpgsql

```

```

AS $$
DECLARE
    dsetUUID uuid;
BEGIN

    select core.register_dataset('SMU','HF',NULL) into dsetUUID;

    INSERT INTO core.surface_site (id, dataset_id ,latitude, longitude, site_name,
township, range, section, other_ids, surface_temperature, elevation_gl,
    tectonic_province,lithology,comment, ams, mintemp, maxtemp,location_notes,
other_location_name)
        SELECT
            core.generate_uuid('SMU','SURFACE_SITE', gid) id,
            dsetUUID as dataset_id,
            n_lat as latitude,
            w_long as longitude,
            gid as site_name,
            (regexp_split_to_array(twn_rng_sec,E'[-/*\s]'))[1] as township,
            (regexp_split_to_array(twn_rng_sec,E'[-/*\s]'))[2] as range,

            (regexp_split_to_array(twn_rng_sec,E'[-/*\s]'))[3] as section,
            ('SMU: ' || gid) as other_ids,
            surf_temp as surface_temperature,
            collar_elevation as elevation_gl,
            tect_prov as tectonic_province,
            lit_info as lithology,
            coalesce(comment_info,' ') || ' ; ' || coalesce('water_table_depth:
'||water_table,' ') as comment,
            area_or_ams_map as ams,
            min_temp as mintemp,
            max_temp as maxtemp,
            location_notes as location_notes,
            ('SMU: ' || gid || ' ' || hole_name) as other_location_name
        FROM staging.smu2_hf_point;

    UPDATE core.surface_site SET shape = geometryfromtext('POINT('|| longitude || ' ' ||
latitude ||)'), 4326) WHERE dataset_id = dsetUUID;
    UPDATE core.surface_site SET county_id=c.id FROM core.ref_county c WHERE
ST_Contains(c.geometry,shape) and dataset_id = dsetUUID;

    INSERT INTO core.bht_measurements (id,surface_site_id, dataset_id, bht,bht_depth,
drill_date)
        SELECT distinct
            core.generate_uuid('SMU','BHT', gid || ';' || coalesce(drill_depth,0)) AS
id,
            core.generate_uuid('SMU','SURFACE_SITE', gid) AS surface_site_id,

```

```

        dsetUUID,
        bot_temp AS bht,
        drill_depth AS bht_depth,
        meas_date AS drilldate
FROM staging.smu2_hf_point
WHERE (drill_depth is not null);

```

```

INSERT INTO core.depth (surface_site_id, dataset_id,depth_date,depth)
SELECT distinct
        core.generate_uuid('SMU','SURFACE_SITE', gid) AS surface_site_id,
        dsetUUID,
        drill_date AS depth_date,
        drill_depth as depth
FROM staging.smu2_hf_point
WHERE (drill_depth is not null);

```

```

INSERT INTO core.thermal_conductivity (id,surface_site_id, dataset_id,conductivity,
conductivity_se, conductivity_nb_samples, porosity)
SELECT distinct
        core.generate_uuid('SMU','THERMAL_CONDUCTIVITY', gid || ';' ||
interval_id) AS id,
        core.generate_uuid('SMU','SURFACE_SITE', gid) AS surface_site_id,
        dsetUUID,
        avg_tcu,
        avg_tcu_std_err,
        bulk_tcu_repeats,
        porosity
FROM staging.smu2_hf_heatflow
where (avg_tcu is not null);

```

```

INSERT INTO core.aquifier (surface_site_id, dataset_id, water_table_depth,
aquifier_temperature, aquifier_temperature_se)
SELECT distinct
        core.generate_uuid('SMU','SURFACE_SITE', P.gid) AS surface_site_id,
        dsetUUID,
        P.water_table,
        H.aquifier_temp ,
        H.aquifier_temp_std_err
FROM staging.smu2_hf_heatflow H JOIN staging.smu2_hf_point P ON
h.gid=p.gid
where (P.water_table is not null);

```

```

INSERT INTO core.heat_flow (id,surface_site_id, thermal_conductivity_id,
bht_measurements_id,dataset_id, interval_order,interval_top_depth, interval_bottom_depth,
        gradient, gradient_se, gradient_corrected, gradient_corrected_se, heat_flow,
heat_flow_se, heat_flow_corrected, heat_flow_corrected_se,
        heat_generation, heat_generation_se,heat_generation_nb_samples, quality)
SELECT distinct
        core.generate_uuid('SMU','HEAT_FLOW', H.gid || ';' || H.interval_id || ';'
|| coalesce(H.depth,0) || ';' || coalesce(H.bottom,0)) AS id,
        core.generate_uuid('SMU','SURFACE_SITE', H.gid) AS
surface_site_id,
        case when (H.avg_tcu is not null) then
                core.generate_uuid('SMU','THERMAL_CONDUCTIVITY',
H.gid || ';' || H.interval_id)
        else null end,
        case when (P.drill_depth is not null) then
                core.generate_uuid('SMU','BHT', H.gid || ';' ||
coalesce(P.drill_depth,0))
        else null end,
        dsetUUID,
        H.interval_id,
        H.depth,
        H.bottom,
        H.geoth_grad ,
        H.geoth_grad_std_err ,
        H.cgeoth_grad ,
        H.cgeoth_grad_std_err ,
        H.heat_flow ,
        H.heat_flow_std_err ,
        H.cheat_flow ,
        H.cheat_flow_std_err,
        H.heat_production ,
        H.heat_production_std_err,
        H.num_hg,
        H.quality_code as quality
FROM staging.smu2_hf_heatflow H JOIN staging.smu2_hf_point P on
H.gid=P.gid;

-- add references to site_related_resources
INSERT INTO core.site_related_resources
(surface_site_id,dataset_id,shape,resource_id,"type",description,url, source)
SELECT distinct
        core.generate_uuid('SMU','SURFACE_SITE', Z.gid) as surface_site_id,
        dsetUUID,
        geometryfromtext('POINT('|| Z.w_long || ' ' || Z.n_lat ||)'), 4326) as shape,
        PB.publication_uuid as resource_id,

```

```

        'Publication' as "type",
        substring(Z.info from 1 for 128) as description,
        substring(Z.link from 1 for 128) as url,
        'SMU' as source
    FROM (select P.gid, P.w_long, P.n_lat, split_part(R.smu_id,' ',1) as source, trim(
leading '0' from split_part(R.smu_id,' ',2)) as source_number, R.*
        from (select
            trim( leading ' ' from split_part(new_pub_ref,' ',1)) as ref1,
            trim( leading ' ' from split_part(new_pub_ref,' ',2)) as ref2,
            trim( leading ' ' from split_part(new_pub_ref,' ',3)) as ref3,
            * from staging.smu2_hf_point where new_pub_ref!='#N/A') as P,
    staging.smu2_hf_ref R
        where (P.ref1=R.new_pub_ref or P.ref2=R.new_pub_ref or
P.ref3=R.new_pub_ref) and length(R.new_pub_ref)>0) Z,
        core.publication PB
    WHERE Z.source=PB.source and Z.source_number=PB.source_number;

-- update materialized views in staging
perform staging.refresh_matview('staging.smu_hf_view_materialized');
perform core.recreate_all_matviews();

RETURN 0;
END;

$$;

ALTER FUNCTION staging.smu_hf_transformationx() OWNER TO postgres;

--
-- Name: texas_tech_transformationx(); Type: FUNCTION; Schema: staging; Owner: postgres
--

CREATE FUNCTION texas_tech_transformationx() RETURNS integer
    LANGUAGE plpgsql
    AS $$

DECLARE
    dsetUUID uuid;
BEGIN

    select core.register_dataset('TEXAS_TECH','ALL',NULL) into dsetUUID;

    UPDATE staging.tt_well_data SET well_name=null where well_name="";

```

```

INSERT INTO core.surface_site (id, dataset_id ,api, site_name, field, lease_name,
elevation_kb, latitude, longitude, operator,plugged, district,well_type, other_location_name)
SELECT distinct
core.generate_uuid('TEXAS_TECH','SURFACE_SITE',w.API), dsetUUID,
w.api, well_name || '(' || w.api || ')', field_name,
s_lease_num,
rkb_elev/3.280839895, s_lat, s_long, operator,
status, cast(district as int), type_code,
coalesce('surface bolk: '||s_blk,' ')
FROM staging.tt_well_data w where w.api is not null;

```

```

INSERT INTO core.depth (surface_site_id, dataset_id, depth_date, depth,
true_vertical_depth)
SELECT distinct core.generate_uuid('TEXAS_TECH','SURFACE_SITE',API),
dsetUUID,
to_date(td_date, 'YYYYMMDD'),
to_number(td_md, '999999')/3.280839895,
to_number(tvd, '999999')/3.280839895
FROM staging.tt_well_data where api is not null;

```

```

INSERT INTO core.well_dates (surface_site_id, dataset_id,drl_start_date,status_date)
SELECT distinct
core.generate_uuid('TEXAS_TECH','SURFACE_SITE',API) AS
surface_site_id,
dsetUUID,
to_date(spud_date, 'YYYYMMDD') as drl_start_date,
to_date(st_date, 'YYYYMMDD') as status_date
FROM staging.tt_well_data;

```

/*

-- adding description to relates resources from uncorrected bht

```

INSERT INTO core.site_related_resources (surface_site_id, label, resource_id, url,
dataset_id,"type",description,shape)
SELECT distinct core.generate_uuid('TEXAS_TECH','SURFACE_SITE',API),
file_name,
core.generate_uuid('TEXAS_TECH','WELL_LOG',file_name),
'https://geothermal.smu.edu/gtda/rest/fileSystem/getFile?uuid=' ||
core.generate_uuid('TEXAS_TECH','WELL_LOG',file_name),
dsetUUID, 'WELL_LOG', description, geometryfromtext('POINT('||
s_long || ' ' || s_lat ||)'), 4326)
FROM (select L.api,L.file_name, D.description, W.s_lat,W.s_long from
staging.tt_uncorrected_bht L
LEFT JOIN staging.tt_log_descriptions D ON L.file_name=D.file_name
LEFT JOIN staging.tt_well_data W ON W.api=L.api) Z;

```

```

-- adding description to relates resources from tt_log_reports

INSERT INTO core.site_related_resources (surface_site_id, label, resource_id, url,
dataset_id,"type",description,shape)
    SELECT distinct core.generate_uuid('TEXAS_TECH','SURFACE_SITE',API),
        file_name,
        core.generate_uuid('TEXAS_TECH','WELL_LOG',file_name),
        'https://geothermal.smu.edu/gtda/rest/fileSystem/getFile?uuid=' ||
core.generate_uuid('TEXAS_TECH','WELL_LOG',file_name),
        dsetUUID, 'WELL_LOG', description, geometryfromtext('POINT('||
s_long || ' ' || s_lat ||)'), 4326)
    FROM (select L.api,L.file_name, D.description, W.s_lat,W.s_long from
staging.tt_log_reports L
    LEFT JOIN staging.tt_log_descriptions D ON L.file_name=D.file_name
    LEFT JOIN staging.tt_well_data W ON W.api=L.api) Z;

*/

-- adding uncorrected bht and depth:

INSERT INTO core.bht_measurements (id,surface_site_id, dataset_id, bht,bht_depth)
    SELECT distinct
        core.generate_uuid('TEXAS_TECH','BHT',API || ';' || bht_depth || ';' ||
file_name || ';' || row_number() over (partition by API, bht_depth, file_name)) AS id,
        core.generate_uuid('TEXAS_TECH','SURFACE_SITE',API) AS
surface_site_id,
        dsetUUID,
        bht,
        bht_depth
    FROM(
        SELECT      API, file_name,
            (cast(bht as double precision)-32)*5/9 AS bht,
            cast(depth as double precision)/3.280839895 AS bht_depth
        FROM staging.tt_uncorrected_bht where bht is not null and depth is not null

        UNION

        SELECT      API, file_name,
            (cast(bht1 as double precision)-32)*5/9 AS bht,
            cast(depth1 as double precision)/3.280839895 AS bht_depth
        FROM staging.tt_uncorrected_bht where bht1 is not null and depth1 is not null

        UNION

        SELECT      API, file_name,
            (cast(bht2 as double precision)-32)*5/9 AS bht,
            cast(depth2 as double precision)/3.280839895 AS bht_depth
        FROM staging.tt_uncorrected_bht where bht2 is not null and depth2 is not null

        UNION

```

```

SELECT          API, file_name,
                (cast(bht3 as double precision)-32)*5/9 AS bht,
                cast(depth3 as double precision)/3.280839895 AS bht_depth
FROM staging.tt_uncorrected_bht
where bht3 is not null and depth3 is not null

```

```

UNION
SELECT          API, file_name,
                (cast(bht4 as double precision)-32)*5/9 AS bht,
                cast(depth4 as double precision)/3.280839895 AS bht_depth
FROM staging.tt_uncorrected_bht
where bht4 is not null and depth4 is not null

```

```

UNION
SELECT          API, file_name,
                (cast(bht5 as double precision)-32)*5/9 AS bht,
                cast(depth5 as double precision)/3.280839895 AS bht_depth
FROM staging.tt_uncorrected_bht
where bht5 is not null and depth5 is not null

```

```

UNION
SELECT          API, file_name,
                (cast(bht6 as double precision)-32)*5/9 AS bht,
                cast(depth6 as double precision)/3.280839895 AS bht_depth
FROM staging.tt_uncorrected_bht
where bht6 is not null and depth6 is not null

```

```

UNION
SELECT          API, file_name,
                (cast(bht7 as double precision)-32)*5/9 AS bht,
                cast(depth7 as double precision)/3.280839895 AS bht_depth
FROM staging.tt_uncorrected_bht
where bht7 is not null and depth7 is not null

```

```

UNION
SELECT          API, file_name,
                (cast(bht8 as double precision)-32)*5/9 AS bht,
                cast(depth8 as double precision)/3.280839895 AS bht_depth
FROM staging.tt_uncorrected_bht
where bht8 is not null and depth8 is not null

```

```

UNION
SELECT          API, file_name,
                (cast(bht9 as double precision)-32)*5/9 AS bht,
                cast(depth9 as double precision)/3.280839895 AS bht_depth
FROM staging.tt_uncorrected_bht

```


where bht9 is not null and depth9 is not null

```
UNION
SELECT      API, file_name,
            (cast(bht10 as double precision)-32)*5/9 AS bht,
            cast(depth10 as double precision)/3.280839895 AS bht_depth
FROM staging.tt_uncorrected_bht
where bht10 is not null and depth10 is not null
```

```
UNION
SELECT      API, file_name,
            (cast(bht11 as double precision)-32)*5/9 AS bht,
            cast(depth11 as double precision)/3.280839895 AS bht_depth
FROM staging.tt_uncorrected_bht
where bht11 is not null and depth11 is not null
```

```
UNION
SELECT      API, file_name,
            (cast(bht12 as double precision)-32)*5/9 AS bht,
            cast(depth12 as double precision)/3.280839895 AS bht_depth
FROM staging.tt_uncorrected_bht
where bht12 is not null and depth12 is not null
```

```
UNION
SELECT      API, file_name,
            (cast(bht13 as double precision)-32)*5/9 AS bht,
            cast(depth13 as double precision)/3.280839895 AS bht_depth
FROM staging.tt_uncorrected_bht
where bht13 is not null and depth13 is not null
```

```
UNION
SELECT      API, file_name,
            (cast(bht14 as double precision)-32)*5/9 AS bht,
            cast(depth14 as double precision)/3.280839895 AS bht_depth
FROM staging.tt_uncorrected_bht
where bht14 is not null and depth14 is not null
```

```
UNION
SELECT      API, file_name,
            (cast(bht15 as double precision)-32)*5/9 AS bht,
            cast(depth15 as double precision)/3.280839895 AS bht_depth
FROM staging.tt_uncorrected_bht
where bht15 is not null and depth15 is not null
```

```
UNION
SELECT      API, file_name,
```

```

        (cast(bht16 as double precision)-32)*5/9 AS bht,
        cast(depth16 as double precision)/3.280839895 AS bht_depth
FROM staging.tt_uncorrected_bht
where bht16 is not null and depth15 is not null
UNION
SELECT      API, file_name,
        (cast(bht17 as double precision)-32)*5/9 AS bht,
        cast(depth17 as double precision)/3.280839895 AS bht_depth
FROM staging.tt_uncorrected_bht
where bht17 is not null and depth15 is not null
UNION
SELECT      API, file_name,
        (cast(bht18 as double precision)-32)*5/9 AS bht,
        cast(depth18 as double precision)/3.280839895 AS bht_depth
FROM staging.tt_uncorrected_bht
where bht18 is not null and depth15 is not null
    ) Z where api is not null;

-- adding corrected bht (sub-sea-floor and depth)
INSERT INTO core.bht_measurements (id,surface_site_id, dataset_id,
bht_corrected,bht_depth)
    SELECT distinct core.generate_uuid('TEXAS_TECH','BHT',API || ';' || depth || ';'
|| row_number() over (partition by API, depth)) AS id,
        core.generate_uuid('TEXAS_TECH','SURFACE_SITE',API) AS
surface_site_id,
        dsetUUID,
        vrt AS bht_corrected,
        depth AS bht_depth
FROM
    (SELECT C.api, C.vrt, (to_number(W.w_depth, '999999') +W.rkb_elev +
C.ssf_depth_ft)/3.280839895 as depth
    FROM staging.tt_corrected_bht C, staging.tt_well_data W
    WHERE C.api=W.api AND C.vrt is not null and C.ssf_depth_ft is not
null) Z;

UPDATE core.surface_site SET shape = geometryfromtext('POINT('|| longitude || ' ' || latitude
||)'), 4326) WHERE dataset_id = dsetUUID;
UPDATE core.surface_site SET county_id=c.id FROM core.ref_county c WHERE
ST_Contains(c.geometry,shape) and dataset_id = dsetUUID;

-- update materialized views in staging
perform staging.refresh_matview('staging.tt_well_view_materialized');
perform core.update_all_matviews();

RETURN 0;

```

```
END;  
$$;
```

```
ALTER FUNCTION staging.texas_tech_transformationx() OWNER TO postgres;
```

```
--  
-- Name: und_data_transformationx(); Type: FUNCTION; Schema: staging; Owner: postgres  
--
```

```
CREATE FUNCTION und_data_transformationx() RETURNS integer  
  LANGUAGE plpgsql  
  AS $$  
DECLARE  
    dsetUUID uuid;  
BEGIN  
  
    select core.register_dataset('UND','ALL',NULL) into dsetUUID;  
  
    INSERT INTO core.surface_site (id, dataset_id, latitude, longitude, range, section,  
section_part,township, tectonic_province, site_owner,site_name, elevation_gl)  
    select distinct  
        first_value(id) OVER (PARTITION BY id),  
            dsetUUID as dataset_id,  
                first_value(latitude) OVER (PARTITION BY id),  
                    first_value(longitude)  
OVER (PARTITION BY id),  
  
        first_value(range) OVER (PARTITION BY id),  
  
            first_value(section) OVER (PARTITION BY id),  
  
                first_value(sectionpart) OVER  
(PARTITION BY id),  
  
        first_value(township) OVER (PARTITION BY id),  
  
            first_value(tectonic_province) OVER (PARTITION BY id),  
  
                first_value(site_owner) OVER (PARTITION  
BY id),
```

first_value(site_name) OVER (PARTITION BY id),

first_value(elevation_gl) OVER (PARTITION BY id)

FROM

((select distinct

core.generate_uuid('UND','SURFACE_SITE',
to_char(coalesce(rad.latdegree,tc.LatDegreeWGS84),'999.999') ||
to_char(coalesce(rad.longdegree,tc.LongDegreeWGS84),'999.999')) as id,

coalesce(tc.LatDegreeWGS84, rad.latdegree) as latitude, -- into surface_site

coalesce(tc.LongDegreeWGS84, rad.longdegree) as longitude, -- into
surface_site

rad.township as township, -- ss.township

rad.range as range, -- ss.range

```

ss.section
rad.section as section, --

rad.sectionpart as sectionpart, -- ss.section_part

rad.tectonicprovince as tectonic_province, --ss.tectonic_province,

null as site_owner, -- ss.site_owner

tc.sitelocationname as site_name, -- ss.well_name,

cast(null as double precision) as elevation_gl -
- tds.elevation_per_m

from staging.und_tc_data tc FULL
OUTER JOIN staging.und_rad_data rad ON tc.observationuri=rad.observationuri)

UNION

```

(select distinct

core.generate_uuid('UND','SURFACE_SITE', tds.hole_number || ';' || tds.label_description) as
id,

tds.latitude as latitude, -- into surface_site

tds.longitude as longitude, -- into surface_site

substring(tds.trs from 'T([0-9]+[NSEW]?)/') as
township, -- ss.township

substring(tds.trs from 'R([0-
9]+[NSEW]?)/') as range, -- ss.range

```
from 'S([0-9]+)') as section, -- ss.section
```

```
substring(tds.trs
```

```
substring(tds.trs from 'S[0-9]+([A-Z]+)') as sectionpart, -- ss.section_part
```

```
null as tectonic_province, --ss.tectonic_province,
```

```
tds.company_owner as site_owner, -- ss.site_owner,
```

```
tds.location_name as site_name, -- ss.well_name,
```

```

elevation_gl-- ss.elevation_gl
tds.elevation_per_m as

from

staging.und_td_sites tds)) Z;

```

```

UPDATE
core.surface_site SET shape = geometryfromtext('POINT('|| longitude || ' ' || latitude ||)'), 4326)
WHERE dataset_id = dsetUUID;

```

```

UPDATE
core.surface_site SET county_id=c.id FROM core.ref_county c WHERE
ST_Contains(c.geometry,shape) and dataset_id = dsetUUID;

```



```
INSERT  
INTO core.formation (surface_site_id, dataset_id, formation,formation_type)
```

```
SELECT  
distinct
```

```
core.generate_uuid('UND','SURFACE_SITE',  
to_char(coalesce(rad.latdegree,tc.LatDegreeWGS84),'999.999')
```

```
||  
to_char(coalesce(rad.longdegree,tc.LongDegreeWGS84),'999.999')) as surface_site_id,
```

dsetUUID as dataset_id,

tc.geologicformationname AS formation,

tc.geologicformationtype AS formation_type

from staging.und_tc_data tc FULL OUTER JOIN staging.und_rad_data rad ON
tc.observationuri=rad.observationuri

where tc.geologicformationname is not null;

```
INSERT INTO core.bht_measurements (id,surface_site_id, dataset_id,  
bht,bht_depth, drill_date)
```

```
SELECT distinct
```

```
core.generate_uuid('UND','BHT', tds.hole_number || ';' ||  
tds.label_description || ';' || tds.drill_depth_m) AS id,
```

```
core.generate_uuid('UND','SURFACE_SITE', tds.hole_number || ';' ||  
tds.label_description) as surface_site_id,
```

```
dsetUUID,
```

tds.bht_c as bht,

tds.drill_depth_m AS depth,

tds.log_date AS depth_date

from staging.und_td_sites tds

where (tds.log_date is not null and
tds.drill_depth_m is not null and tds.bht_c is not null);

INSERT INTO core.depth (surface_site_id,
dataset_id, depth)

SELECT distinct

core.generate_uuid('UND','SURFACE_SITE', tds.hole_number || ';' || tds.label_description) as
surface_site_id,

dsetUUID,

-- tds.log_date AS depth_date, -- don't put it in depth

tds.drill_depth_m AS depth

from staging.und_td_sites tds

not null);
where (tds.log_date is not null and tds.drill_depth_m is

INSERT INTO core.thermal_conductivity
(id,surface_site_id, dataset_id, laboratory, measurement_device, measurement_pressure
,measurement_procedure,

sample_collection_date, sample_measurement_date, sample_depth,

sample_height,
sample_width, sample_length, sample_shape, sample_temperature_diff, sample_type,
conductivity, permeability,comments)

SELECT
distinct

core.generate_uuid('UND','THERMAL_CONDUCTIVITY', tc.observationuri) AS id,

core.generate_uuid('UND','SURFACE_SITE',
to_char(coalesce(rad.latdegree,tc.LatDegreeWGS84),'999.999')

```

        || to_char(coalesce(rad.longdegree,tc.LongDegreeWGS84),'999.999')) as
surface_site_id,

dsetUUID,

tc.laboratory as
laboratory ,

tc.measurementdevice as measurement_device,

tc.measurementpressure as measurement_pressure ,

tc.measurementprocedure as
measurement_procedure,

tc.samplecollectiondate
as sample_collection_date,

```

tc.samplemeasurementdate as sample_measurement_date,

tc.sampledepth as sample_depth,

tc.sampledimensionsheight as
sample_height,

tc.sampledimensionswidth as sample_width,

tc.sampledimensionslength as sample_length,

tc.sampleshape as sample_shape,

sample_temperature_diff,

tc.sampletemperature as

tc.sampletype as sample_type,

tc.saturatedsampleconductivity as conductivity,

tc.permeability as permeability,

tc.notes as comments

```
tc FULL OUTER JOIN staging.und_rad_data rad ON tc.observationuri=rad.observationuri
from staging.und_tc_data
```

```
WHERE
(tc.observationuri is not null);
```

```
-- radioactivity info ties to
heat production (which is in heat flow)
```

```
INSERT INTO
core.radioactivity (id, surface_site_id, dataset_id, laboratory, measurement_procedure,
measurement_device, rock, specimen_label, potassium,thorium, uranium,comments)
```

```
SELECT
distinct
```

```
core.generate_uuid('UND','RADIOACTIVITY', rad.observationuri) as id,
```

```
core.generate_uuid('UND','SURFACE_SITE',  
to_char(coalesce(rad.latdegree,tc.LatDegreeWGS84),'999.999')
```

```
|| to_char(coalesce(rad.longdegree,tc.LongDegreeWGS84),'999.999')) as surface_site_id,
```

```
dsetUUID,
```

```
case when  
(length(rad.laboratory)>0) then rad.laboratory else rad.laboratoryuri end as laboratory,
```

```
rad.measurementprocedure as measurement_procedure,
```

```
rad.RadioactivityMeasurementDevice as measurement_device ,
```

```
rad.rockname as rock,
```

specimen_label, rad.specimenlabel as

rad.potassium as potassium,

rad.thorium as thorium,

rad.uranium as uranium,

rad.notes as comments

```
from staging.und_tc_data tc FULL OUTER  
JOIN staging.und_rad_data rad ON tc.observationuri=rad.observationuri
```

```
WHERE (rad.observationuri is not null);
```

```
INSERT INTO core.heat_flow (id,  
thermal_conductivity_id, radioactivity_id, surface_site_id, dataset_id, heat_generation,  
heat_generation_se)
```

```
SELECT distinct
```

```
core.generate_uuid('UND','HEAT_FLOW', rad.observationuri),
```

```
case when (tc.observationuri is not null) then  
core.generate_uuid('UND','THERMAL_CONDUCTIVITY', tc.observationuri)
```

```
else null end AS id,
```

```
core.generate_uuid('UND','RADIOACTIVITY', rad.observationuri) as radioactivity_id,
```

```
core.generate_uuid('UND','SURFACE_SITE',  
to_char(coalesce(rad.latdegree,tc.LatDegreeWGS84),'999.999')
```

```

to_char(coalesce(rad.longdegree,tc.LongDegreeWGS84,'999.999')) as surface_site_id,

```

```

dsetUUID,

```

```

heat_generation,
rad.heatproduction_mwm3 AS

```

```

when (length(rad.heatproductionuncertainty)=0) then null else
cast(split_part(rad.heatproductionuncertainty,' ',1) as double precision) end AS
heat_generation_se
case

```

```
staging.und_tc_data tc FULL OUTER JOIN staging.und_rad_data rad ON
tc.observationuri=rad.observationuri
```

from

```
WHERE (rad.observationuri is not null and rad.heatproduction_mwm3 is not null);
```

```
populate temperature series and precision logs:
```

--

```
INSERT INTO core.precision_log (id, surface_site_id, dataset_id,
precision_log_source_id, log_date, log_depth)
```

```
distinct
```

select

```
core.generate_uuid('UND','TD_CURVE', tds.hole_number || ';' || tds.label_description) as id,
```



```
core.generate_uuid('UND','SURFACE_SITE',  
tds.hole_number || ';' || tds.label_description) as surface_site_id,
```

```
dsetUUID,
```

```
tds.label_description as precision_log_source_id,
```

```
to_char(tds.log_date,'YYYY-MM-DD') as log_date,
```

```
tds.drill_depth_m as log_depth
```

```
from staging.und_td_sites tds;
```

```
INSERT INTO  
core.temperature_series (precision_log_id, dataset_id, depth, temperature)
```

```
select distinct
```

```
core.generate_uuid('UND','TD_CURVE', tds.hole_number || ';' || tds.label_description) as  
precision_log_id,
```

```
dsetUUID,
```

```
tdc.depth,
```

tdc.temperature

from staging.und_td_sites tds

LEFT OUTER JOIN

staging.und_td_curve_points tdc ON (tds.location_name=tdc.location_name OR
tds.hole_number=tdc.hole_number OR tds.label_description=tdc.sheet_name)

WHERE (tds.sites_id is not null and tdc.depth
is not null and tdc.temperature is not null);

-- add td curves to related resources:

```
INSERT INTO core.site_related_resources  
(surface_site_id, dataset_id, label, type, shape, source)
```

select distinct

```
core.generate_uuid('UND','SURFACE_SITE', tds.hole_number || ';' || tds.label_description) as  
surface_site_id,
```

```
dsetUUID as dataset_id,
```

```

        tds.label_description as label,

        'TD_CURVES' as type,

        geometryfromtext('POINT(' || tds.longitude || ' ' ||
tds.latitude || ')', 4326) as shape,

        'UND' as source

    from

staging.und_td_sites tds;

-- update materialized views in staging
perform staging.refresh_matview('staging.und_rad_view_materialized');
perform staging.refresh_matview('staging.und_tc_view_materialized');
perform staging.refresh_matview('staging.und_td_view_materialized');
-- perform core.update_all_matviews();
-- perform core.recreate_all_matviews();

RETURN 0;
END;

$$;

ALTER FUNCTION staging.und_data_transformationx() OWNER TO postgres;

SET search_path = core, pg_catalog;

SET default_tablespace = '';

SET default_with_oids = false;

```

```
--  
-- Name: aquifier; Type: TABLE; Schema: core; Owner: postgres; Tablespace:  
--
```

```
CREATE TABLE aquifier (  
    id bigint NOT NULL,  
    surface_site_id uuid,  
    dataset_id uuid,  
    water_table_depth double precision,  
    aquifier_temperature double precision,  
    aquifier_temperature_se double precision  
);
```

```
ALTER TABLE core.aquifier OWNER TO postgres;
```

```
--  
-- Name: aquifier_id_seq; Type: SEQUENCE; Schema: core; Owner: postgres  
--
```

```
CREATE SEQUENCE aquifier_id_seq  
    START WITH 1  
    INCREMENT BY 1  
    NO MAXVALUE  
    NO MINVALUE  
    CACHE 1;
```

```
ALTER TABLE core.aquifier_id_seq OWNER TO postgres;
```

```
--  
-- Name: aquifier_id_seq; Type: SEQUENCE OWNED BY; Schema: core; Owner: postgres  
--
```

```
ALTER SEQUENCE aquifier_id_seq OWNED BY aquifier.id;
```

```
--  
-- Name: bht_measurements; Type: TABLE; Schema: core; Owner: postgres; Tablespace:  
--
```

```
CREATE TABLE bht_measurements (  
    id uuid NOT NULL,  
    surface_site_id uuid,  
    bht double precision,
```

```

    bht_corrected double precision,
    bht_correction_type_id integer,
    bht_depth double precision,
    bht_timestamp timestamp without time zone,
    drill_date date,
    time_since_circulation double precision,
    estimated boolean DEFAULT false,
    dataset_id uuid,
    run_number integer,
    bht_correction_type character varying(16),
    date_type character varying(16)
);

ALTER TABLE core.bht_measurements OWNER TO postgres;

--
-- Name: bht_measurements_max_temp; Type: VIEW; Schema: core; Owner: postgres
--

CREATE VIEW bht_measurements_max_temp AS
    SELECT bht_measurements.surface_site_id, max(bht_measurements.bht) AS temperature
FROM bht_measurements GROUP BY bht_measurements.surface_site_id;

ALTER TABLE core.bht_measurements_max_temp OWNER TO postgres;

--
-- Name: casing; Type: TABLE; Schema: core; Owner: postgres; Tablespace:
--

CREATE TABLE casing (
    id bigint NOT NULL,
    surface_site_id uuid,
    dataset_id uuid,
    casing_size_inches double precision,
    casing_depth double precision,
    liner_inches double precision,
    liner_top double precision,
    liner_base double precision,
    choke_mm double precision,
    casing_weight double precision,
    casing_top character varying(7),
    casing_left double precision,
    file_date date
);

```

```

ALTER TABLE core.casing OWNER TO postgres;

--
-- Name: casing_id_seq; Type: SEQUENCE; Schema: core; Owner: postgres
--

CREATE SEQUENCE casing_id_seq
    START WITH 1
    INCREMENT BY 1
    NO MAXVALUE
    NO MINVALUE
    CACHE 1;

ALTER TABLE core.casing_id_seq OWNER TO postgres;

--
-- Name: casing_id_seq; Type: SEQUENCE OWNED BY; Schema: core; Owner: postgres
--

ALTER SEQUENCE casing_id_seq OWNED BY casing.id;

--
-- Name: dataset_job_data; Type: TABLE; Schema: core; Owner: postgres; Tablespace:
--

CREATE TABLE dataset_job_data (
    dataset_job_data_id integer NOT NULL,
    dataset_id uuid NOT NULL,
    dataset_name character varying(100) NOT NULL,
    job_name character varying(100) NOT NULL,
    display_name character varying(100) NOT NULL,
    file_name character varying(100) NOT NULL,
    file_format character varying(100) NOT NULL,
    required boolean DEFAULT false,
    type character varying(50)
);

ALTER TABLE core.dataset_job_data OWNER TO postgres;

--

```



```
-- Name: dataset_job_data_dataset_job_data_id_seq; Type: SEQUENCE; Schema: core;  
Owner: postgres  
--
```

```
CREATE SEQUENCE dataset_job_data_dataset_job_data_id_seq  
  START WITH 1  
  INCREMENT BY 1  
  NO MAXVALUE  
  NO MINVALUE  
  CACHE 1;
```

```
ALTER TABLE core.dataset_job_data_dataset_job_data_id_seq OWNER TO postgres;
```

```
--  
-- Name: dataset_job_data_dataset_job_data_id_seq; Type: SEQUENCE OWNED BY;  
Schema: core; Owner: postgres  
--
```

```
ALTER SEQUENCE dataset_job_data_dataset_job_data_id_seq OWNED BY  
dataset_job_data.dataset_job_data_id;
```

```
--  
-- Name: datasets; Type: TABLE; Schema: core; Owner: postgres; Tablespace:  
--
```

```
CREATE TABLE datasets (  
  id uuid NOT NULL,  
  datasource character varying(16) NOT NULL,  
  datatype character varying(16) NOT NULL,  
  resource_id character varying(32)  
);
```

```
ALTER TABLE core.datasets OWNER TO postgres;
```

```
--  
-- Name: datasets_map; Type: TABLE; Schema: core; Owner: postgres; Tablespace:  
--
```

```
CREATE TABLE datasets_map (  
  datasets_map_id integer NOT NULL,  
  dataset_id uuid NOT NULL,  
  table_names character varying(256) NOT NULL  
);
```

```

ALTER TABLE core.datasets_map OWNER TO postgres;

--
-- Name: datasets_map_datasets_map_id_seq; Type: SEQUENCE; Schema: core; Owner:
postgres
--

CREATE SEQUENCE datasets_map_datasets_map_id_seq
  START WITH 1
  INCREMENT BY 1
  NO MAXVALUE
  NO MINVALUE
  CACHE 1;

ALTER TABLE core.datasets_map_datasets_map_id_seq OWNER TO postgres;

--
-- Name: datasets_map_datasets_map_id_seq; Type: SEQUENCE OWNED BY; Schema: core;
Owner: postgres
--

ALTER SEQUENCE datasets_map_datasets_map_id_seq OWNED BY
datasets_map.datasets_map_id;

--
-- Name: depth_id_seq; Type: SEQUENCE; Schema: core; Owner: postgres
--

CREATE SEQUENCE depth_id_seq
  START WITH 1
  INCREMENT BY 1
  NO MAXVALUE
  NO MINVALUE
  CACHE 1;

ALTER TABLE core.depth_id_seq OWNER TO postgres;

--
-- Name: depth; Type: TABLE; Schema: core; Owner: postgres; Tablespace:
--

```

```

CREATE TABLE depth (
  id bigint DEFAULT nextval('depth_id_seq'::regclass) NOT NULL,
  surface_site_id uuid,
  depth_date date,
  depth double precision,
  true_vertical_depth double precision,
  dataset_id uuid,
  bit_diameter double precision,
  mud_weight double precision,
  viscosity double precision
);

```

```

ALTER TABLE core.depth OWNER TO postgres;

```

```

--
-- Name: depth_max_temp; Type: VIEW; Schema: core; Owner: postgres
--

```

```

CREATE VIEW depth_max_temp AS
  SELECT depth.surface_site_id, max(depth.depth) AS depth FROM depth GROUP BY
  depth.surface_site_id;

```

```

ALTER TABLE core.depth_max_temp OWNER TO postgres;

```

```

--
-- Name: field_report_mapping_id_seq; Type: SEQUENCE; Schema: core; Owner: postgres
--

```

```

CREATE SEQUENCE field_report_mapping_id_seq
  START WITH 16
  INCREMENT BY 1
  NO MAXVALUE
  NO MINVALUE
  CACHE 1;

```

```

ALTER TABLE core.field_report_mapping_id_seq OWNER TO postgres;

```

```

--
-- Name: file_upload; Type: TABLE; Schema: core; Owner: postgres; Tablespace:
--

```

```

CREATE TABLE file_upload (
  file_upload_id integer NOT NULL,

```

```

file_name character varying(100),
file_size character varying(25),
file_path character varying(256),
created_date timestamp without time zone,
processed_date timestamp without time zone,
status_code integer,
user_id integer,
error_message character varying(2048),
job_execution_id integer NOT NULL,
datasets_id uuid NOT NULL,
job_type character varying(50),
created_timestamp bigint
);

```

```

ALTER TABLE core.file_upload OWNER TO postgres;

```

```

--
-- Name: file_upload_file_upload_id_seq; Type: SEQUENCE; Schema: core; Owner: postgres
--

```

```

CREATE SEQUENCE file_upload_file_upload_id_seq
START WITH 1
INCREMENT BY 1
NO MAXVALUE
NO MINVALUE
CACHE 1;

```

```

ALTER TABLE core.file_upload_file_upload_id_seq OWNER TO postgres;

```

```

--
-- Name: file_upload_file_upload_id_seq; Type: SEQUENCE OWNED BY; Schema: core;
Owner: postgres
--

```

```

ALTER SEQUENCE file_upload_file_upload_id_seq OWNED BY file_upload.file_upload_id;

```

```

--
-- Name: filter_condition; Type: TABLE; Schema: core; Owner: postgres; Tablespace:
--

```

```

CREATE TABLE filter_condition (
user_id integer,
filter_type name,

```

```

    field_name name,
    operator_name character varying(50),
    static_value character varying(1000),
    condition_comments character varying(1000),
    dataset_id uuid
);

```

```

ALTER TABLE core.filter_condition OWNER TO postgres;

```

```

--
-- Name: fluid; Type: TABLE; Schema: core; Owner: postgres; Tablespace:
--

```

```

CREATE TABLE fluid (
    id uuid NOT NULL,
    surface_site_id uuid,
    dataset_id uuid,
    cntl_no integer,
    lease_id integer,
    well_no character(6),
    field_no integer,
    oper integer,
    uic_class integer,
    bbl_vol_inj integer,
    max_inj_pressure integer,
    top_inj_zone integer,
    bot_inj_zone integer,
    w_status integer,
    sw character(1),
    fw character(1),
    bw character(1),
    air character(1),
    gas character(1),
    lpg character(1),
    co2 character(1),
    polymer character(1),
    cycle_yr date,
    fw_avg_or_max character(1),
    fw_bbls_day integer
);

```

```

ALTER TABLE core.fluid OWNER TO postgres;

```

```

--

```

```
-- Name: fluid_h10; Type: TABLE; Schema: core; Owner: postgres; Tablespace:
--
```

```
CREATE TABLE fluid_h10 (
    id bigint NOT NULL,
    surface_site_id uuid,
    dataset_id uuid,
    injection_id uuid,
    h10_date date,
    h10_avg_inj_pressure integer,
    h10_max_inj_pressure integer,
    h10_total_vol_bbl integer,
    h10_total_vol_mcf integer
);
```

```
ALTER TABLE core.fluid_h10 OWNER TO postgres;
```

```
--
-- Name: fluid_h10_id_seq; Type: SEQUENCE; Schema: core; Owner: postgres
--
```

```
CREATE SEQUENCE fluid_h10_id_seq
    START WITH 1
    INCREMENT BY 1
    NO MAXVALUE
    NO MINVALUE
    CACHE 1;
```

```
ALTER TABLE core.fluid_h10_id_seq OWNER TO postgres;
```

```
--
-- Name: fluid_h10_id_seq; Type: SEQUENCE OWNED BY; Schema: core; Owner: postgres
--
```

```
ALTER SEQUENCE fluid_h10_id_seq OWNED BY fluid_h10.id;
```

```
--
-- Name: fluid_h10h; Type: TABLE; Schema: core; Owner: postgres; Tablespace:
--
```

```
CREATE TABLE fluid_h10h (
    id bigint NOT NULL,
    surface_site_id uuid,
```

```
dataset_id uuid,  
injection_id uuid,  
h10h_date date NOT NULL,  
h10h_max_hydrocarb_psig integer,  
h10h_max_brine_psig integer  
);
```

```
ALTER TABLE core.fluid_h10h OWNER TO postgres;
```

```
--  
-- Name: fluid_h10h_id_seq; Type: SEQUENCE; Schema: core; Owner: postgres  
--
```

```
CREATE SEQUENCE fluid_h10h_id_seq  
START WITH 1  
INCREMENT BY 1  
NO MAXVALUE  
NO MINVALUE  
CACHE 1;
```

```
ALTER TABLE core.fluid_h10h_id_seq OWNER TO postgres;
```

```
--  
-- Name: fluid_h10h_id_seq; Type: SEQUENCE OWNED BY; Schema: core; Owner: postgres  
--
```

```
ALTER SEQUENCE fluid_h10h_id_seq OWNED BY fluid_h10h.id;
```

```
--  
-- Name: formation; Type: TABLE; Schema: core; Owner: postgres; Tablespace:  
--
```

```
CREATE TABLE formation (  
id bigint NOT NULL,  
surface_site_id uuid,  
dataset_id uuid,  
formation character varying(50),  
formation_depth integer,  
formation_type character varying(64)  
);
```

```
ALTER TABLE core.formation OWNER TO postgres;
```

```
--  
-- Name: formation_id_seq; Type: SEQUENCE; Schema: core; Owner: postgres  
--
```

```
CREATE SEQUENCE formation_id_seq  
  START WITH 1  
  INCREMENT BY 1  
  NO MAXVALUE  
  NO MINVALUE  
  CACHE 1;
```

```
ALTER TABLE core.formation_id_seq OWNER TO postgres;
```

```
--  
-- Name: formation_id_seq; Type: SEQUENCE OWNED BY; Schema: core; Owner: postgres  
--
```

```
ALTER SEQUENCE formation_id_seq OWNED BY formation.id;
```

```
--  
-- Name: heat_flow; Type: TABLE; Schema: core; Owner: postgres; Tablespace:  
--
```

```
CREATE TABLE heat_flow (  
  id uuid NOT NULL,  
  surface_site_id uuid,  
  publication_source_id uuid,  
  bht_measurements_id uuid,  
  precision_log_id uuid,  
  thermal_conductivity_id uuid,  
  heat_generation_id bigint,  
  interval_order character varying(16),  
  interval_top_depth double precision,  
  interval_bottom_depth double precision,  
  gradient double precision,  
  gradient_se double precision,  
  gradient_corrected double precision,  
  gradient_corrected_se double precision,  
  gradient_correction_type_id integer,  
  heat_flow double precision,  
  heat_flow_se double precision,  
  heat_flow_corrected double precision,  
  heat_flow_corrected_se double precision,
```



```

    heat_flow_quality_code_id integer,
    heat_flow_quality_method_id integer,
    lithology_information_id integer,
    notes_id uuid,
    dataset_id uuid,
    heat_generation double precision,
    heat_generation_se double precision,
    heat_generation_nb_samples integer,
    radioactivity_id uuid,
    quality character varying(16)
);

```

```

ALTER TABLE core.heat_flow OWNER TO postgres;

```

```

--
-- Name: heat_flow_id_seq; Type: SEQUENCE; Schema: core; Owner: postgres
--

```

```

CREATE SEQUENCE heat_flow_id_seq
    START WITH 1
    INCREMENT BY 1
    NO MAXVALUE
    NO MINVALUE
    CACHE 1;

```

```

ALTER TABLE core.heat_flow_id_seq OWNER TO postgres;

```

```

--
-- Name: heat_flow_publication_links_id_seq; Type: SEQUENCE; Schema: core; Owner:
postgres
--

```

```

CREATE SEQUENCE heat_flow_publication_links_id_seq
    START WITH 1
    INCREMENT BY 1
    NO MAXVALUE
    NO MINVALUE
    CACHE 1;

```

```

ALTER TABLE core.heat_flow_publication_links_id_seq OWNER TO postgres;

```

```

--
-- Name: heat_generation_id_seq; Type: SEQUENCE; Schema: core; Owner: postgres

```

--

```
CREATE SEQUENCE heat_generation_id_seq
  START WITH 1
  INCREMENT BY 1
  NO MAXVALUE
  NO MINVALUE
  CACHE 1;
```

```
ALTER TABLE core.heat_generation_id_seq OWNER TO postgres;
```

--

-- Name: interval; Type: TABLE; Schema: core; Owner: postgres; Tablespace:

--

```
CREATE TABLE "interval" (
  id bigint NOT NULL,
  surface_site_id uuid,
  dataset_id uuid,
  top_depth double precision,
  bottom_depth double precision,
  interval_length double precision,
  water_volume double precision,
  bhp double precision,
  producing_reservoir character varying(64),
  duration_hours double precision
);
```

```
ALTER TABLE core."interval" OWNER TO postgres;
```

--

-- Name: interval_id_seq; Type: SEQUENCE; Schema: core; Owner: postgres

--

```
CREATE SEQUENCE interval_id_seq
  START WITH 1
  INCREMENT BY 1
  NO MAXVALUE
  NO MINVALUE
  CACHE 1;
```

```
ALTER TABLE core.interval_id_seq OWNER TO postgres;
```

```
--  
-- Name: interval_id_seq; Type: SEQUENCE OWNED BY; Schema: core; Owner: postgres  
--
```

```
ALTER SEQUENCE interval_id_seq OWNED BY "interval".id;
```

```
--  
-- Name: lithology_information_id_seq; Type: SEQUENCE; Schema: core; Owner: postgres  
--
```

```
CREATE SEQUENCE lithology_information_id_seq  
  START WITH 1  
  INCREMENT BY 1  
  NO MAXVALUE  
  NO MINVALUE  
  CACHE 1;
```

```
ALTER TABLE core.lithology_information_id_seq OWNER TO postgres;
```

```
--  
-- Name: matviews; Type: TABLE; Schema: core; Owner: postgres; Tablespace:  
--
```

```
CREATE TABLE matviews (  
  mv_name name NOT NULL,  
  v_name name NOT NULL,  
  last_refresh timestamp with time zone  
);
```

```
ALTER TABLE core.matviews OWNER TO postgres;
```

```
--  
-- Name: ngds_heat_flow_mview; Type: TABLE; Schema: core; Owner: postgres; Tablespace:  
--
```

```
CREATE TABLE ngds_heat_flow_mview (  
  "ObservationURI" uuid,  
  "ObservationName" text,  
  "Label" text,  
  "HeaderURI" uuid,  
  point public.geometry,  
  "APINo" character varying(18),  
  "WellName" name,
```

```

"OtherId" character varying(32),
"Owner" character varying(128),
"LeaseName" character varying(128),
"Field" character varying(128),
"County" name,
"State" name,
"LatDegreeWGS84" numeric(10,7),
"LongDegreeWGS84" numeric(10,7),
"LocationUncertaintyStatement" character varying(4000),
"PLSS_Meridians" character varying(64),
"Township" character varying(5),
"Range" character varying(5),
"Section" character varying(5),
"SectionPart" character varying(5),
"TectonicProvince" character varying(4000),
"OtherLocationName" name,
"WellType" character varying(100),
dataset_id uuid
);

```

```

ALTER TABLE core.ngds_heat_flow_mview OWNER TO postgres;

```

```

--
-- Name: notes; Type: TABLE; Schema: core; Owner: postgres; Tablespace:
--

```

```

CREATE TABLE notes (
    id uuid NOT NULL,
    notes character varying(4000),
    surface_site_id uuid,
    dataset_id uuid
);

```

```

ALTER TABLE core.notes OWNER TO postgres;

```

```

--
-- Name: precision_log; Type: TABLE; Schema: core; Owner: postgres; Tablespace:
--

```

```

CREATE TABLE precision_log (
    id uuid NOT NULL,
    surface_site_id uuid,
    precision_log_source_id character varying(256) NOT NULL,
    log_date character varying(10),

```

```

log_depth double precision,
log_deepest_temperature double precision,
log_max_temperature double precision,
time_since_circulation double precision,
log_equipment_id integer,
log_company character varying(64),
dataset_id uuid
);

```

```

ALTER TABLE core.precision_log OWNER TO postgres;

```

```

--
-- Name: COLUMN precision_log.precision_log_source_id; Type: COMMENT; Schema: core;
Owner: postgres
--

```

```

COMMENT ON COLUMN precision_log.precision_log_source_id IS 'The precision log td
curve file name.';

```

```

--
-- Name: COLUMN precision_log.log_date; Type: COMMENT; Schema: core; Owner:
postgres
--

```

```

COMMENT ON COLUMN precision_log.log_date IS 'The precision log date, but someytimes
we only have partial information. A string permit that.';

```

```

--
-- Name: precision_log_max_temp; Type: VIEW; Schema: core; Owner: postgres
--

```

```

CREATE VIEW precision_log_max_temp AS
  SELECT precision_log.surface_site_id, max(precision_log.log_max_temperature) AS
temperature FROM precision_log GROUP BY precision_log.surface_site_id;

```

```

ALTER TABLE core.precision_log_max_temp OWNER TO postgres;

```

```

--
-- Name: publication; Type: TABLE; Schema: core; Owner: postgres; Tablespace:
--

```

```

CREATE TABLE publication (

```

```

publication_id integer NOT NULL,
authors character varying(1024),
title character varying(512),
year character varying(64),
report_number character varying(100),
pages character varying(100),
agency character varying(512),
contract character varying(128),
source_number character varying(64),
source character varying(64),
location character varying(128),
pdf character varying(32),
electronic character varying(32),
url character varying(2000),
month numeric(2,0),
publisher character varying(512),
volume character varying(128),
issue character varying(64),
collection character varying(256),
keywords character varying(1024),
publication_uuid uuid,
ocr_required boolean DEFAULT false,
expected boolean DEFAULT true,
dataset_id uuid,
data_source_id integer,
indexed boolean,
status character(10)
);

```

```

ALTER TABLE core.publication OWNER TO postgres;

```

```

SET search_path = filesystem, pg_catalog;

```

```

--
-- Name: files; Type: TABLE; Schema: filesystem; Owner: postgres; Tablespace:
--

```

```

CREATE TABLE files (
    file_uuid uuid NOT NULL,
    parent_uuid uuid,
    file_size bigint NOT NULL,
    file_path_internal character varying(256) NOT NULL,
    file_upload_date timestamp without time zone,
    file_name character varying(256) NOT NULL,
    file_mimetype character varying(70),

```

```

    file_downloadable boolean DEFAULT true,
    file_path_external character varying(256),
    file_provider character varying(32)
);

ALTER TABLE filesystem.files OWNER TO postgres;

SET search_path = core, pg_catalog;

--
-- Name: publication_metadata; Type: VIEW; Schema: core; Owner: postgres
--

CREATE VIEW publication_metadata AS
    SELECT ('bib '::text || COALESCE(publication.publication_uuid,
generate_uuid(publication.source, 'PUBLICATION '::character varying,
publication.source_number))) AS id, COALESCE(publication.publication_uuid,
generate_uuid(publication.source, 'PUBLICATION '::character varying,
publication.source_number)) AS resource_id, publication.publication_uuid AS metadata_uuid,
publication.title, publication.title AS description, publication.authors AS author,
publication.year AS publication_date, 'Document' AS resource_type, publication.keywords AS
keywords_thematic, 'en' AS resource_languages, files.file_path_external AS resource_url,
now() AS metdata_date, 'en' AS metadata_language, 'SMU Geothermal Laboratory, Roy M.
Huffington Department of Earth Sciences' AS metadata_contact_org_name, 'Maria Richard' AS
metadata_contact_person_name, 'Coordinator, SMU Geothermal Laboratory' AS
metadata_contact_position_name, 'mrichard@mail.smu.edu' AS metadata_contact_email, 'P.O.
Box 75-0395' AS metadata_contact_street_address, 'Dallas' AS metadata_contact_city, 'TX' AS
metadata_contact_state, '75275-0395' AS metadata_contact_zip, '+1 (214) 768 1975' AS
metadata_contact_phone FROM (publication publication LEFT JOIN filesystem.files ON
((files.file_uuid = publication.publication_uuid))) WHERE ((publication.status = '0 '::bpchar)
OR (publication.status = '1 '::bpchar));

ALTER TABLE core.publication_metadata OWNER TO postgres;

--
-- Name: publication_publication_id_seq; Type: SEQUENCE; Schema: core; Owner: postgres
--

CREATE SEQUENCE publication_publication_id_seq
    START WITH 1
    INCREMENT BY 1
    NO MAXVALUE
    NO MINVALUE
    CACHE 1;

```

```

ALTER TABLE core.publication_publication_id_seq OWNER TO postgres;

--
-- Name: publication_publication_id_seq; Type: SEQUENCE OWNED BY; Schema: core;
Owner: postgres
--

ALTER SEQUENCE publication_publication_id_seq OWNED BY publication.publication_id;

--
-- Name: quality_issue_log_id_seq; Type: SEQUENCE; Schema: core; Owner: postgres
--

CREATE SEQUENCE quality_issue_log_id_seq
    START WITH 1
    INCREMENT BY 1
    NO MAXVALUE
    NO MINVALUE
    CACHE 1;

ALTER TABLE core.quality_issue_log_id_seq OWNER TO postgres;

--
-- Name: radioactivity; Type: TABLE; Schema: core; Owner: postgres; Tablespace:
--

CREATE TABLE radioactivity (
    id uuid NOT NULL,
    surface_site_id uuid,
    dataset_id uuid,
    laboratory character varying(64),
    measurement_procedure character varying(64),
    measurement_device character varying(64),
    rock character varying(64),
    specimen_label character varying(32),
    potassium double precision,
    thorium double precision,
    uranium double precision,
    comments character varying(128)
);

```



```
ALTER TABLE core.radioactivity OWNER TO postgres;
```

```
--  
-- Name: ref_county; Type: TABLE; Schema: core; Owner: postgres; Tablespace:  
--
```

```
CREATE TABLE ref_county (  
    id integer NOT NULL,  
    state_id integer,  
    fips_code character varying(5),  
    county_name character varying(50),  
    geometry public.geometry,  
    min_elevation double precision,  
    max_elevation double precision,  
    CONSTRAINT county_enforce_dims_shape CHECK ((public.st_ndims(geometry) = 2)),  
    CONSTRAINT county_enforce_geotype_shape CHECK (((public.geometrytype(geometry)  
= 'MULTIPOLYGON'::text) OR (geometry IS NULL))),  
    CONSTRAINT county_enforce_srid_shape CHECK ((public.st_srid(geometry) = 4326))  
);
```

```
ALTER TABLE core.ref_county OWNER TO postgres;
```

```
--  
-- Name: ref_county_elev; Type: VIEW; Schema: core; Owner: postgres  
--
```

```
CREATE VIEW ref_county_elev AS  
    SELECT c.id, CASE WHEN (c.min_elevation IS NOT NULL) THEN c.min_elevation  
ELSE (0)::double precision END AS min_elevation, CASE WHEN (c.max_elevation IS NOT  
NULL) THEN c.max_elevation ELSE (6100)::double precision END AS max_elevation  
FROM ref_county c;
```

```
ALTER TABLE core.ref_county_elev OWNER TO postgres;
```

```
--  
-- Name: VIEW ref_county_elev; Type: COMMENT; Schema: core; Owner: postgres  
--
```

```
COMMENT ON VIEW ref_county_elev IS 'add a global min and max elevation to the original  
ref_county table. Used in heatflow-r-analyse.xml to generate log of impossible elevations';
```

```
--  
-- Name: ref_data_source; Type: TABLE; Schema: core; Owner: postgres; Tablespace:
```

```

--

CREATE TABLE ref_data_source (
    id integer NOT NULL,
    ref_name character varying(10),
    ref_description character varying(4000)
);

ALTER TABLE core.ref_data_source OWNER TO postgres;

--
-- Name: ref_quality_issue_type; Type: TABLE; Schema: core; Owner: postgres; Tablespace:
--

CREATE TABLE ref_quality_issue_type (
    id integer NOT NULL,
    pid integer,
    ref_name character varying(100),
    ref_description character varying(4000)
);

ALTER TABLE core.ref_quality_issue_type OWNER TO postgres;

--
-- Name: ref_quality_issue_type_h; Type: VIEW; Schema: core; Owner: postgres
--

CREATE VIEW ref_quality_issue_type_h AS
    WITH RECURSIVE t(level, id, child_id, parent_name, child_name, cycle) AS (SELECT 1,
    NULL::integer AS int4, NULL::integer AS int4, "::character varying AS "varchar", "::character
    varying AS "varchar", false AS bool UNION ALL SELECT (t.level + 1), parent.id, child.id,
    parent.ref_name, child.ref_name, true AS bool FROM ((ref_quality_issue_type parent JOIN
    ref_quality_issue_type child ON ((parent.id = child.pid))) CROSS JOIN t) WHERE (NOT
    t.cycle)) SELECT t.level, t.id, t.child_id, t.parent_name, t.child_name, t.cycle FROM t
    WHERE (t.id IS NOT NULL);

ALTER TABLE core.ref_quality_issue_type_h OWNER TO postgres;

--
-- Name: ref_state; Type: TABLE; Schema: core; Owner: postgres; Tablespace:
--

CREATE TABLE ref_state (

```

```

id integer NOT NULL,
fips_code character varying(2),
state_name character varying(32),
state_abbr character varying(2),
sub_region character varying(7),
shape public.geometry,
CONSTRAINT state_enforce_dims_shape CHECK ((public.st_ndims(shape) = 2)),
CONSTRAINT state_enforce_geotype_shape CHECK (((public.geometrytype(shape) =
'MULTIPOLYGON'::text) OR (shape IS NULL))),
CONSTRAINT state_enforce_srid_shape CHECK ((public.st_srid(shape) = 4326))
);

```

```

ALTER TABLE core.ref_state OWNER TO postgres;

```

```

--
-- Name: ref_symbol; Type: TABLE; Schema: core; Owner: postgres; Tablespace:
--

```

```

CREATE TABLE ref_symbol (
    id integer NOT NULL,
    ref_name character varying(10),
    ref_description character varying(4000)
);

```

```

ALTER TABLE core.ref_symbol OWNER TO postgres;

```

```

--
-- Name: reservoir; Type: TABLE; Schema: core; Owner: postgres; Tablespace:
--

```

```

CREATE TABLE reservoir (
    id uuid NOT NULL,
    dataset_id uuid,
    reservoiruri character varying(255),
    reservoirname character varying(255),
    label character varying(255),
    otherid character varying(255),
    othername character varying(255),
    description character varying(255),
    county character varying(255),
    state character varying(255),
    locationname character varying(255),
    latdegree double precision,
    longdegree double precision,

```

srs character varying(255),
 locationuncertainty statement character varying(255),
 temperaturemax_f double precision,
 temperaturemin_f double precision,
 depthtotop_ft double precision,
 depthtobottom_ft double precision,
 dominantformationname character varying(255),
 otherformationname character varying(255),
 dominantlithologyname character varying(255),
 otherlithologyname character varying(255),
 area_acres double precision,
 width_ft double precision,
 length_ft double precision,
 sandvolume_cft double precision,
 blockvolume_cft double precision,
 percentsand double precision,
 averagenetsandthickness_ft double precision,
 wellcount integer,
 faultcount integer,
 faulted character varying(255),
 faulttype character varying(255),
 faultyhydrauliccharacter character varying(255),
 firstproductiondate date,
 lastproductiondate date,
 yearsinproduction integer,
 monthlyproduction_bbl double precision,
 productionreportdate date,
 averagemonthlywaterproduction_bbl double precision,
 minimummonthlywaterproduction_bbl double precision,
 maximummonthlywaterproduction_bbl double precision,
 waterinjection_bbl double precision,
 watersaturation_pct double precision,
 watertds_mgl double precision,
 averagefieldwidepressure_psi double precision,
 averagefieldwidegradient_degfft double precision,
 porositymax_pct double precision,
 porositymin_pct double precision,
 permeabilitymax_md double precision,
 permeabilitymin_md double precision,
 cementationmineralogy character varying(255),
 trap character varying(255),
 drive character varying(255),
 gravity double precision,
 commodityofinterest character varying(255),
 stimulation character varying(255),
 averageperforatedthickness_ft double precision,

```
relatedresource character varying(255),
source character varying(1023),
metadatauri character varying(255),
majorfields character varying(511),
shape public.geometry
);
```

```
ALTER TABLE core.reservoir OWNER TO postgres;
```

```
--
-- Name: site_comments_id_seq; Type: SEQUENCE; Schema: core; Owner: postgres
--
```

```
CREATE SEQUENCE site_comments_id_seq
START WITH 1
INCREMENT BY 1
NO MAXVALUE
NO MINVALUE
CACHE 1;
```

```
ALTER TABLE core.site_comments_id_seq OWNER TO postgres;
```

```
--
-- Name: site_publication_links_id_seq; Type: SEQUENCE; Schema: core; Owner: postgres
--
```

```
CREATE SEQUENCE site_publication_links_id_seq
START WITH 1
INCREMENT BY 1
NO MAXVALUE
NO MINVALUE
CACHE 1;
```

```
ALTER TABLE core.site_publication_links_id_seq OWNER TO postgres;
```

```
--
-- Name: site_related_resources; Type: TABLE; Schema: core; Owner: postgres; Tablespace:
--
```

```
CREATE TABLE site_related_resources (
surface_site_id uuid,
label character varying(128),
url character varying(512),
```

```
type character varying(32),
resource_id uuid,
description character varying(512),
dataset_id uuid,
shape public.geometry,
source character varying(32),
record_id integer NOT NULL
);
```

```
ALTER TABLE core.site_related_resources OWNER TO postgres;
```

```
--
-- Name: site_related_resources_id_seq; Type: SEQUENCE; Schema: core; Owner: postgres
--
```

```
CREATE SEQUENCE site_related_resources_id_seq
START WITH 1
INCREMENT BY 1
NO MAXVALUE
NO MINVALUE
CACHE 1;
```

```
ALTER TABLE core.site_related_resources_id_seq OWNER TO postgres;
```

```
--
-- Name: site_related_resources_record_id_seq; Type: SEQUENCE; Schema: core; Owner:
postgres
--
```

```
CREATE SEQUENCE site_related_resources_record_id_seq
START WITH 1
INCREMENT BY 1
NO MAXVALUE
NO MINVALUE
CACHE 1;
```

```
ALTER TABLE core.site_related_resources_record_id_seq OWNER TO postgres;
```

```
--
-- Name: site_related_resources_record_id_seq; Type: SEQUENCE OWNED BY; Schema:
core; Owner: postgres
--
```

```
ALTER SEQUENCE site_related_resources_record_id_seq OWNED BY
site_related_resources.record_id;
```

```
--
-- Name: surface_site; Type: TABLE; Schema: core; Owner: postgres; Tablespace:
--
```

```
CREATE TABLE surface_site (
    id uuid NOT NULL,
    api character varying(18),
    site_name character varying(64),
    source_id character varying(32),
    site_owner character varying(128),
    lease_name character varying(128),
    field character varying(128),
    county_id integer,
    latitude numeric(10,7),
    longitude numeric(10,7),
    plss_meridians character varying(64),
    township character varying(5),
    range character varying(5),
    section character varying(10),
    section_part character varying(5),
    other_location_name character varying(256),
    elevation_gl double precision,
    elevation_df double precision,
    elevation_kb double precision,
    surface_temperature double precision,
    shape public.geometry,
    dataset_id uuid,
    basement_depth double precision,
    average_heatflow double precision,
    tectonic_province character varying(128),
    well_type character varying(128),
    plugged character varying(64),
    shut_in_date date,
    block character varying(10),
    feet_from_survey integer,
    section_block_survey character varying(52),
    hole_open integer,
    fresh_water_converted integer,
    district integer,
    lithology character varying(255),
    comment character varying(512),
    contributor character varying(32),
```

```

    mintemp numeric(10,7),
    maxtemp numeric(10,7),
    ams character varying(64),
    location_notes character varying(128),
    original_source character varying(16),
    operator character varying(128),
    other_ids character varying(128),
    CONSTRAINT enforce_dims_shape CHECK ((public.st_ndims(shape) = 2)),
    CONSTRAINT enforce_geotype_shape CHECK (((public.geometrytype(shape) =
'POINT'::text) OR (shape IS NULL))),
    CONSTRAINT enforce_srid_shape CHECK ((public.st_srid(shape) = 4326))
);

```

```

ALTER TABLE core.surface_site OWNER TO postgres;

```

```

--
-- Name: COLUMN surface_site.site_name; Type: COMMENT; Schema: core; Owner:
postgres
--

```

```

COMMENT ON COLUMN surface_site.site_name IS 'Can be a well_name';

```

```

--
-- Name: COLUMN surface_site.source_id; Type: COMMENT; Schema: core; Owner:
postgres
--

```

```

COMMENT ON COLUMN surface_site.source_id IS 'Is an external identifier used to uniquely
allocate used uuid';

```

```

--
-- Name: surface_site_county_state_materialized_view; Type: TABLE; Schema: core; Owner:
postgres; Tablespace:
--

```

```

CREATE TABLE surface_site_county_state_materialized_view (
    id character varying,
    uuid character varying,
    name character varying,
    site_name character varying(64),
    other_location_name character varying(256),
    api character varying(18),
    latitude numeric(10,7),

```



```

longitude numeric(10,7),
state_id integer,
state character varying(32),
state_code character varying(2),
county character varying(50),
county_id integer,
bht double precision,
max_temperature double precision,
depth double precision,
shape public.geometry,
datasource character varying(16),
thermal_conductivity double precision,
heat_flow double precision,
precision_log_source_id character varying(256)
);

```

```

ALTER TABLE core.surface_site_county_state_materialized_view OWNER TO postgres;

```

```

--
-- Name: thermal_conductivity; Type: TABLE; Schema: core; Owner: postgres; Tablespace:
--

```

```

CREATE TABLE thermal_conductivity (
    id uuid NOT NULL,
    surface_site_id uuid,
    conductivity double precision,
    conductivity_se double precision,
    conductivity_sample_type_id integer,
    conductivity_nb_samples integer,
    conductivity_measurement_type_id integer,
    conductivity_sample_shape_id integer,
    sample_width double precision,
    sample_height double precision,
    sample_length double precision,
    density double precision,
    density_uncertainty double precision,
    dataset_id uuid,
    resistance double precision,
    porosity double precision,
    laboratory character varying(32),
    measurement_device character varying(32),
    measurement_procedure character varying(128),
    sample_collection_date date,
    sample_measurement_date date,
    sample_depth double precision,

```

```

sample_shape character varying(32),
sample_temperature_diff double precision,
sample_type character varying(32),
permeability double precision,
lithology_info character varying(128),
comments character varying(128),
measurement_pressure double precision
);

```

```

ALTER TABLE core.thermal_conductivity OWNER TO postgres;

```

```

--
-- Name: surface_site_county_state_view; Type: VIEW; Schema: core; Owner: postgres
--

```

```

CREATE VIEW surface_site_county_state_view AS
    SELECT DISTINCT (s.id)::character varying AS id, (s.id)::character varying AS uuid,
    COALESCE(s.site_name, s.api, s.other_location_name) AS name, s.site_name,
    s.other_location_name, s.api, s.latitude, s.longitude, st.id AS state_id, st.state_name AS state,
    st.state_abbr AS state_code, c.county_name AS county, c.id AS county_id, bht.bht,
    (s.maxtemp)::double precision AS max_temperature, GREATEST(d.depth, bht.bht_depth) AS
    depth, s.shape, ds.datasource, tc.conductivity AS thermal_conductivity,
    COALESCE(s.average_heatflow, hf.heat_flow) AS heat_flow, pl.precision_log_source_id
    FROM (((((((surface_site s LEFT JOIN ref_county c ON ((c.id = s.county_id))) LEFT JOIN
    ref_state st ON ((st.id = c.state_id))) LEFT JOIN depth d ON ((d.surface_site_id = s.id))) LEFT
    JOIN bht_measurements bht ON ((bht.surface_site_id = s.id))) LEFT JOIN datasets ds ON
    ((ds.id = s.dataset_id))) LEFT JOIN thermal_conductivity tc ON ((tc.surface_site_id = s.id)))
    LEFT JOIN heat_flow hf ON ((hf.surface_site_id = s.id))) LEFT JOIN precision_log pl ON
    ((pl.surface_site_id = s.id))) WHERE ((bht.estimated IS NULL) OR (bht.estimated = false))
    ORDER BY (s.id)::character varying;

```

```

ALTER TABLE core.surface_site_county_state_view OWNER TO postgres;

```

```

--
-- Name: temperature_series_id_seq; Type: SEQUENCE; Schema: core; Owner: postgres
--

```

```

CREATE SEQUENCE temperature_series_id_seq
    START WITH 1
    INCREMENT BY 1
    NO MAXVALUE
    NO MINVALUE
    CACHE 1;

```

```

ALTER TABLE core.temperature_series_id_seq OWNER TO postgres;

--
-- Name: temperature_series; Type: TABLE; Schema: core; Owner: postgres; Tablespace:
--

CREATE TABLE temperature_series (
    id bigint DEFAULT nextval('temperature_series_id_seq'::regclass) NOT NULL,
    precision_log_id uuid,
    depth double precision,
    temperature double precision,
    dataset_id uuid
);

ALTER TABLE core.temperature_series OWNER TO postgres;

--
-- Name: well_dates; Type: TABLE; Schema: core; Owner: postgres; Tablespace:
--

CREATE TABLE well_dates (
    id bigint NOT NULL,
    surface_site_id uuid,
    dataset_id uuid,
    compl_date date,
    drl_compl_date date,
    water_injection_nbr character varying(6),
    elevation integer,
    elevation_code character varying(2),
    drl_start_date date,
    status_date date
);

ALTER TABLE core.well_dates OWNER TO postgres;

--
-- Name: well_info; Type: TABLE; Schema: core; Owner: postgres; Tablespace:
--

CREATE TABLE well_info (
    id bigint NOT NULL,
    surface_site_id uuid,
    dataset_id uuid,

```

```

    commodity character varying(16),
    well_id character varying(16),
    permit character varying(16)
);

```

```

ALTER TABLE core.well_info OWNER TO postgres;

```

```

--
-- Name: template_borehole; Type: VIEW; Schema: core; Owner: postgres
--

```

```

CREATE VIEW template_borehole AS
    WITH depth AS (SELECT depth.surface_site_id, max(depth.true_vertical_depth) AS
true_vertical_depth, max(depth.depth) AS depth, max(depth.bit_diameter) AS bit_diameter,
array_to_string(array_agg(DISTINCT depth.depth_date), '|'::text) AS depth_date,
max(depth.viscosity) AS viscosity FROM core.depth GROUP BY depth.surface_site_id),
casing AS (SELECT t.surface_site_id, t.casing_depth, CASE WHEN ((t.casing_top)::text ~
'^[0-9]+$'::text) THEN (t.casing_top)::numeric ELSE ((-999999))::numeric END AS
casing_top, t.casing_size_inches, t.casing_weight FROM (core.casing t JOIN (SELECT
casing.surface_site_id, max(casing.file_date) AS max_date, max(casing.casing_depth) AS
max_depth FROM core.casing GROUP BY casing.surface_site_id) x ON (((t.surface_site_id =
x.surface_site_id) AND ((x.max_date IS NULL) OR (t.file_date = x.max_date))) AND
(t.casing_depth = x.max_depth))))), well_info AS (SELECT well_info.surface_site_id,
array_to_string(array_agg(DISTINCT well_info.commodity), '|'::text) AS commodity,
array_to_string(array_agg(DISTINCT well_info.well_id), '|'::text) AS well_id FROM
core.well_info GROUP BY well_info.surface_site_id), well_dates AS (SELECT
well_dates.surface_site_id, array_to_string(array_agg(DISTINCT well_dates.drl_compl_date),
'|'::text) AS drl_compl_date, array_to_string(array_agg(DISTINCT well_dates.drl_start_date),
'|'::text) AS drl_start_date, array_to_string(array_agg(DISTINCT well_dates.status_date),
'|'::text) AS status_date FROM core.well_dates GROUP BY well_dates.surface_site_id),
formation AS (SELECT formation.surface_site_id, array_to_string(array_agg(DISTINCT
(((formation.formation)::text || ' depth='::text) || CASE WHEN (formation.formation_depth IS
NULL) THEN 'NA'::text ELSE ('::text || formation.formation_depth) END)), '|'::text) AS
formation FROM core.formation GROUP BY formation.surface_site_id) SELECT
('https://geothermal.smu.edu/obervation/borehole/'::text || b.id) AS "ObservationURI",
COALESCE(s.site_name, s.api) AS "WellName", s.api AS "APINo",
('https://geothermal.smu.edu/header/'::text || s.id) AS "HeaderURI", wi.well_id AS "OtherID",
('Borehole observation for well '::text || (COALESCE(s.site_name, s.api))::text) AS "Label",
s.operator AS "Operator", s.lease_name AS "LeaseName", s.site_owner AS "LeaseOwner",
wi.well_id AS "LeaseNo", wd.drl_start_date AS "SpudDate", COALESCE(d.depth_date,
('::text || wd.drl_compl_date)) AS "EndedDrillingDate", s.well_type AS welltype, s.plugged AS
"Status", COALESCE(s.well_type, (wi.commodity)::character varying) AS
"CommodityOfInterest", (((('status_date '::text || wd.status_date) || ' shut_in_date '::text) ||
s.shut_in_date) AS "StatusDate", s.field AS "Field", s.other_location_name AS
"OtherLocationName", COALESCE(rc.county_name, 'nil:missing'::character varying) AS

```

```

"County", COALESCE(rs.state_name, 'nil:missing'::character varying) AS "State",
s.plss_meridians AS "PLSS_Meridians", s.township AS "Township", s.range AS "Range",
s.section AS "Section", s.section_part AS "SectionPart", s.latitude AS "LatDegree", s.longitude
AS "LongDegree", 'WGS84'::character varying(6) AS "SRS", s.location_notes AS
"LocationUncertaintyStatement", d.depth AS "DrillerTotalDepth", ">::character varying(1) AS
"DepthReferencePoint", 'm'::character varying(1) AS "LengthUnits", ">::character varying(1) AS
"WellBoreShape", d.true_vertical_depth AS "TrueVerticalDepth", s.elevation_kb AS
"ElevationKB", s.elevation_df AS "ElevationDF", s.elevation_gl AS "ElevationGL",
f.formation AS "FormationTD", d.bit_diameter AS "BitDiameterCollar", 'inches'::character
varying(6) AS "DiameterUnits", b.bht AS "MeasuredTemperature", b.bht_corrected AS
"CorrectedTemperature", 'C'::character varying(1) AS "TemperatureUnits",
b.time_since_circulation AS "TimeSinceCirculation", b.bht_correction_type_id AS
"CorrectionType", b.bht_depth AS "DepthOfMeasurement", b.drill_date AS
"MeasurementDateTime", 'nil:missing'::character varying(16) AS "MeasurementSource",
cs.casing_depth AS "CasingBottomDepthDriller", cs.casing_top AS "CasingTopDepth",
cs.casing_size_inches AS "CasingPipeDiameter", cs.casing_weight AS "CasingWeight",
'lbs/ft'::character varying(6) AS "CasingWeightUnits", d.viscosity AS "Viscosity", CASE
WHEN ((s.contributor IS NULL) AND (s.original_source IS NULL)) THEN
((((ds.datasource)::text || ' '::text) || (ds.datatype)::text)::character varying WHEN (s.contributor
IS NOT NULL) THEN s.contributor ELSE s.original_source END AS "InformationSource",
s.shape AS "Shape" FROM (((((((((surface_site s JOIN bht_measurements b ON
((b.surface_site_id = s.id))) LEFT JOIN datasets ds ON ((s.dataset_id = ds.id))) LEFT JOIN
ref_county rc ON ((s.county_id = rc.id))) LEFT JOIN ref_state rs ON ((rc.state_id = rs.id)))
LEFT JOIN depth d ON ((d.surface_site_id = s.id))) LEFT JOIN formation f ON
((f.surface_site_id = s.id))) LEFT JOIN casing cs ON ((cs.surface_site_id = s.id))) LEFT JOIN
well_info wi ON ((wi.surface_site_id = s.id))) LEFT JOIN well_dates wd ON
((wd.surface_site_id = s.id))) WHERE (b.estimated IS NOT TRUE);

```

```

ALTER TABLE core.template_borehole OWNER TO postgres;

```

```

--
-- Name: template_borehole_materialized; Type: TABLE; Schema: core; Owner: postgres;
Tablespace:
--

```

```

CREATE TABLE template_borehole_materialized (
    "ObservationURI" text,
    "WellName" character varying,
    "APINo" character varying(18),
    "HeaderURI" text,
    "OtherID" text,
    "Label" text,
    "Operator" character varying(128),
    "LeaseName" character varying(128),
    "LeaseOwner" character varying(128),

```

"LeaseNo" text,
"SpudDate" text,
"EndedDrillingDate" text,
welltype character varying(128),
"Status" character varying(64),
"CommodityOfInterest" character varying,
"StatusDate" text,
"Field" character varying(128),
"OtherLocationName" character varying(256),
"County" character varying,
"State" character varying,
"PLSS_Meridians" character varying(64),
"Township" character varying(5),
"Range" character varying(5),
"Section" character varying(10),
"SectionPart" character varying(5),
"LatDegree" numeric(10,7),
"LongDegree" numeric(10,7),
"SRS" character varying(6),
"LocationUncertaintyStatement" character varying(128),
"DrillerTotalDepth" double precision,
"DepthReferencePoint" character varying(1),
"LengthUnits" character varying(1),
"WellBoreShape" character varying(1),
"TrueVerticalDepth" double precision,
"ElevationKB" double precision,
"ElevationDF" double precision,
"ElevationGL" double precision,
"FormationTD" text,
"BitDiameterCollar" double precision,
"DiameterUnits" character varying(6),
"MeasuredTemperature" double precision,
"CorrectedTemperature" double precision,
"TemperatureUnits" character varying(1),
"TimeSinceCirculation" double precision,
"CorrectionType" integer,
"DepthOfMeasurement" double precision,
"MeasurementDateTime" date,
"MeasurementSource" character varying(16),
"CasingBottomDepthDriller" double precision,
"CasingTopDepth" numeric,
"CasingPipeDiameter" double precision,
"CasingWeight" double precision,
"CasingWeightUnits" character varying(6),
"Viscosity" double precision,
"InformationSource" character varying,

```
"Shape" public.geometry  
);
```

```
ALTER TABLE core.template_borehole_materialized OWNER TO postgres;
```

```
--  
-- Name: template_fluidflux; Type: VIEW; Schema: core; Owner: postgres  
--
```

```
CREATE VIEW template_fluidflux AS
```

```
WITH fluidh10 AS (SELECT fluid_h10.id, fluid_h10.surface_site_id, fluid_h10.dataset_id,  
fluid_h10.injection_id, fluid_h10.h10_date, fluid_h10.h10_avg_inj_pressure,  
fluid_h10.h10_max_inj_pressure, fluid_h10.h10_total_vol_bbl, fluid_h10.h10_total_vol_mcf  
FROM fluid_h10 WHERE ((fluid_h10.h10_total_vol_bbl > 0) OR  
(fluid_h10.h10_avg_inj_pressure > 0))), wi AS (SELECT well_info.surface_site_id,  
array_to_string(array_agg(DISTINCT well_info.commodity), '|')::text AS commodity,  
array_to_string(array_agg(DISTINCT well_info.well_id), '|')::text AS well_id,  
array_to_string(array_agg(DISTINCT well_info.permit), '|')::text AS permit FROM well_info  
GROUP BY well_info.surface_site_id), wd AS (SELECT well_dates.surface_site_id,  
max(well_dates.drl_compl_date) AS drl_compl_date FROM well_dates GROUP BY  
well_dates.surface_site_id) SELECT ('https://geothermal.smu.edu/fluidflux/'::text || s.id) AS  
"ObservationURI", ('Fluid Flux for well '::text || (COALESCE(s.site_name, s.api))::text) AS  
"RecordName", 'nil:missing'::character varying(16) AS "FacilityURI", fluid.oper AS  
"Operator", ('https://geothermal.smu.edu/header/'::text || s.id) AS "HeaderURI",  
COALESCE(s.site_name, s.api) AS "WellName", s.api AS "APINo",  
(((COALESCE(s.other_ids, ' '::character varying))::text || COALESCE(wi.well_id, ' '::text)) AS  
"OtherID", COALESCE(rc.county_name, 'nil:missing'::character varying) AS "County",  
COALESCE(rs.state_name, 'nil:missing'::character varying) AS "State", s.field AS "Field",  
s.latitude AS "LatDegreeWGS84", s.longitude AS "LongDegreeWGS84", 'WGS84'::character  
varying(6) AS "SRS", s.location_notes AS "LocationUncertaintyStatement",  
COALESCE(fluidh10.h10_date, '1900-01-01'::date) AS "IntervalStartDateTime",  
'nil:missing'::character varying(16) AS "IntervalEndDateTime", 'nil:missing'::character  
varying(16) AS "DurationDays", COALESCE(wi.permit, 'nil:missing'::text) AS "Permit",  
CASE WHEN (fluid.sw = 'Y'::bpchar) THEN 'salt water'::text WHEN (fluid.fw = 'Y'::bpchar)  
THEN 'fresh water'::text WHEN (fluid.bw = 'Y'::bpchar) THEN 'brackish water'::text WHEN  
(fluid.air = 'Y'::bpchar) THEN 'air'::text WHEN (fluid.gas = 'Y'::bpchar) THEN 'gas'::text  
WHEN (fluid.lpg = 'Y'::bpchar) THEN 'lpg'::text WHEN (fluid.co2 = 'Y'::bpchar) THEN  
'co2'::text WHEN (fluid.polymer = 'Y'::bpchar) THEN 'polymer'::text ELSE 'nil:missing'::text  
END AS "FluidType", COALESCE(fluidh10.h10_total_vol_bbl, (-99999)) AS "FluidVolume",  
'bbl'::character varying(3) AS "FluidVolumeUnits", (-99999) AS "Temperature_F",  
COALESCE(fluidh10.h10_avg_inj_pressure, (-99999)) AS "Pressure_psi", 'average'::character  
varying(7) AS "AggregationType", 'nil:missing'::character varying(16) AS  
"MeasurementMethod", COALESCE(fluid.top_inj_zone, (-99999)) AS "ContactIntervalTop",  
COALESCE(fluid.bot_inj_zone, (-99999)) AS "ContactIntervalBottom", CASE WHEN  
((s.contributor IS NULL) AND (s.original_source IS NULL)) THEN (((ds.datasource)::text || '
```

```
'::text) || (ds.datatype)::text))::character varying WHEN (s.contributor IS NOT NULL) THEN
s.contributor ELSE s.original_source END AS "Source", 'nil:missing'::character varying(16)
AS "MetadataURI", s.shape AS "Shape" FROM ((((((surface_site s JOIN fluid fluid ON
((fluid.surface_site_id = s.id))) JOIN fluidh10 ON ((fluid.id = fluidh10.injection_id))) LEFT
JOIN datasets ds ON ((s.dataset_id = ds.id))) LEFT JOIN ref_county rc ON ((s.county_id =
rc.id))) LEFT JOIN ref_state rs ON ((rc.state_id = rs.id))) LEFT JOIN wi ON
((wi.surface_site_id = s.id))) LEFT JOIN wd ON ((wd.surface_site_id = s.id)));
```

```
ALTER TABLE core.template_fluidflux OWNER TO postgres;
```

```
--
-- Name: template_fluidflux_materialized; Type: TABLE; Schema: core; Owner: postgres;
Tablespace:
--
```

```
CREATE TABLE template_fluidflux_materialized (
    "ObservationURI" text,
    "RecordName" text,
    "FacilityURI" character varying(16),
    "Operator" integer,
    "HeaderURI" text,
    "WellName" character varying,
    "APINo" character varying(18),
    "OtherID" text,
    "County" character varying,
    "State" character varying,
    "Field" character varying(128),
    "LatDegreeWGS84" numeric(10,7),
    "LongDegreeWGS84" numeric(10,7),
    "SRS" character varying(6),
    "LocationUncertaintyStatement" character varying(128),
    "IntervalStartDateTime" date,
    "IntervalEndDateTime" character varying(16),
    "DurationDays" character varying(16),
    "Permit" text,
    "FluidType" text,
    "FluidVolume" integer,
    "FluidVolumeUnits" character varying(3),
    "Temperature_F" integer,
    "Pressure_psi" integer,
    "AggregationType" character varying(7),
    "MeasurementMethod" character varying(16),
    "ContactIntervalTop" integer,
    "ContactIntervalBottom" integer,
    "Source" character varying,
```



```

    "MetadataURI" character varying(16),
    "Shape" public.geometry
);

```

```

ALTER TABLE core.template_fluidflux_materialized OWNER TO postgres;

```

```

--
-- Name: template_heatflow; Type: VIEW; Schema: core; Owner: postgres
--

```

```

CREATE VIEW template_heatflow AS
    WITH formation AS (SELECT formation.surface_site_id,
array_to_string(array_agg(DISTINCT (((formation.formation)::text || ' depth='::text) || CASE
WHEN (formation.formation_depth IS NULL) THEN 'NA'::text ELSE ("::text ||
formation.formation_depth) END))), '|'::text) AS formation FROM core.formation GROUP BY
formation.surface_site_id), aquifier AS (SELECT aquifier.surface_site_id,
max(aquifier.aquifier_temperature) AS aquifier_temperature, max(aquifier.water_table_depth)
AS water_table_depth FROM aquifier GROUP BY aquifier.surface_site_id), depth AS
(SELECT depth.surface_site_id, max(depth.true_vertical_depth) AS true_vertical_depth,
max(depth.depth) AS depth, array_to_string(array_agg(DISTINCT depth.depth_date), '|'::text)
AS depth_date FROM core.depth GROUP BY depth.surface_site_id) SELECT
('https://geothermal.smu.edu/observation/heatflow/'::text || h.id) AS "ObservationURI", ('Heat
flow observation for well '::text || (COALESCE(s.site_name, s.api))::text) AS
"ObservationName", ('Heat flow observation for well '::text || (COALESCE(s.site_name,
s.api))::text) AS "Label", CASE WHEN (((h.interval_order IS NULL) AND
(h.interval_top_depth IS NULL)) AND (h.interval_bottom_depth IS NULL)) OR
((h.interval_order)::text = 'a'::text)) THEN 'Site'::text ELSE 'Interval'::text END AS
"HeatFlowType", ('https://geothermal.smu.edu/header/'::text || s.id) AS "HeaderURI", s.api AS
"APINo", COALESCE(s.site_name, s.api) AS "WellName", ((((((s.other_ids)::text || '|'::text) ||
'SourceID and Interval: '::text) || (s.source_id)::text) || ' : '::text) || (h.interval_order)::text) AS
"OtherID", s.site_owner AS "Owner", s.lease_name AS "LeaseName", s.field AS "Field",
COALESCE(rc.county_name, 'nil:missing'::character varying) AS "County",
COALESCE(rs.state_name, 'nil:missing'::character varying) AS "State", s.latitude AS
"LatDegreeWGS84", s.longitude AS "LongDegreeWGS84", s.location_notes AS
"LocationUncertaintyStatement", s.plss_meridians AS "PLSS_Meridians", s.township AS
"Township", s.range AS "Range", s.section AS "Section_", s.section_part AS "SectionPart",
s.tectonic_province AS "TectonicProvince", (((s.other_location_name)::text || ' ams: '::text) ||
(s.ams)::text) AS "OtherLocationName", 'nil:NA'::character varying(7) AS
"PublicationSource", s.original_source AS "AdditionalSources", CASE WHEN ((s.well_type
IS NULL) OR (length((s.well_type)::text) = 0)) THEN 'nil:missing'::character varying ELSE
s.well_type END AS "WellType", ("::text || bht.drill_date) AS "WellDrillDate", bht.date_type
AS "WellDrillDateType", CASE WHEN (s.elevation_kb IS NOT NULL) THEN s.elevation_kb
WHEN (s.elevation_df IS NOT NULL) THEN s.elevation_df WHEN (s.elevation_gl IS NOT
NULL) THEN s.elevation_gl ELSE NULL::double precision END AS "Elevation", CASE
WHEN (s.elevation_kb IS NOT NULL) THEN 'KB'::text WHEN (s.elevation_df IS NOT

```

```

NULL) THEN 'DF'::text WHEN (s.elevation_gl IS NOT NULL) THEN 'GL'::text ELSE
NULL::text END AS "ElevationType", 'm'::character varying(1) AS "DepthUnits",
COALESCE(d.depth, ((-99999))::double precision) AS "DrillerTotalDepth",
bht.time_since_circulation AS "TimeSinceCirculation", 'C'::character varying(1) AS
"TemperatureUnits", 'BHT'::character varying(7) AS "TemperatureMeasureType",
s.surface_temperature AS "SurfaceTemperature", bht.bht AS "BHT", bht.bht_corrected AS
"BHTCorrectedValue", bht.bht_correction_type AS "BHTCorrectionType", bht.drill_date AS
"EqLog_MeasurementDate", bht.bht_depth AS "EqLog_TotalDepth", s.surface_temperature
AS "EqLogTemperatureTop", s.maxtemp AS "EqLogTemperatureMax", bht.bht AS
"EqLogTemperatureBottom", COALESCE(s.average_heatflow, ((-99999))::double precision)
AS "SiteHeatFlow", 'nil:NA'::character varying(7) AS "SiteValueMethod", (-99999) AS
"Intervals", COALESCE(h.interval_top_depth, ((-99999))::double precision) AS
"IntervalTopDepth", COALESCE(h.interval_bottom_depth, ((-99999))::double precision) AS
"IntervalBottomDepth", 'Watt per meter per Kelvin'::character varying(64) AS
"ThermalConductivityUnits", tc.conductivity AS "IntervalThermalConductivity",
tc.conductivity_se AS "SE_Conductivity", tc.lithology_info AS "ConductivitySampleType",
tc.conductivity_nb_samples AS "ConductivityNumSamples", CASE WHEN (tc.id IS NULL)
THEN 'nil:NA'::text ELSE ('https://geothermal.smu.edu/obervation/thermal_conductivity/'::text
|| tc.id) END AS "ConductivityMeasuredURI", 'C/km'::character varying(4) AS
"GradientUnits", COALESCE(h.gradient, ((-99999))::double precision) AS
"IntervalUncorrectedGradient", h.gradient_se AS "SE_UncorrectedGradient",
h.gradient_corrected AS "IntervalCorrectedGradient", h.gradient_corrected_se AS
"SE_CorrectedGradient", 'mW/m2'::character varying(5) AS "HeatFlowUnits",
COALESCE(h.heat_flow, ((-99999))::double precision) AS "IntervalUncorrectedHeatFlow",
h.heat_flow_se AS "SE_UncorrectedHeatflow", COALESCE(h.heat_flow_corrected, ((-
99999))::double precision) AS "IntervalCorrectedHeatFlow", h.heat_flow_corrected_se AS
"SE_CorrectedHeatFlow", COALESCE(h.quality, ''::character varying) AS
"IntervalHF_QualityCode", 'µW/m3'::character varying(5) AS "HeatGenerationUnits",
h.heat_generation AS "HeatGeneration", h.heat_generation_nb_samples AS
"HG_NumberSamples", a.water_table_depth AS "WaterTableTop", a.aquifier_temperature AS
"WaterTableTemperature", (((('BasementDepth_m'::text || COALESCE(''::text ||
s.basement_depth), 'NA'::text)) || ' '::text) || (s.lithology)::text) || ' '::text) ||
(tc.lithology_info)::text) AS "LithologyInformation", f.formation AS "FormationName",
(((('WellResistance'::text || tc.resistance) || ' '::text) || (s.comment)::text) AS "Notes", s.shape AS
"Shape" FROM (((((((((surface_site s JOIN heat_flow h ON ((h.surface_site_id = s.id))) LEFT
JOIN ref_county rc ON ((s.county_id = rc.id))) LEFT JOIN ref_state rs ON ((rc.state_id =
rs.id))) LEFT JOIN formation f ON ((f.surface_site_id = s.id))) LEFT JOIN acquifier a ON
((a.surface_site_id = s.id))) LEFT JOIN depth d ON ((d.surface_site_id = s.id))) LEFT JOIN
bht_measurements bht ON (((bht.surface_site_id = s.id) AND (bht.id =
h.bht_measurements_id)))) LEFT JOIN thermal_conductivity tc ON (((tc.surface_site_id = s.id)
AND (tc.id = h.thermal_conductivity_id)))));

```

```

ALTER TABLE core.template_heatflow OWNER TO postgres;

```

```

--

```

-- Name: template_heatflow_materialized; Type: TABLE; Schema: core; Owner: postgres;
Tablespace:

--

```
CREATE TABLE template_heatflow_materialized (  
    "ObservationURI" text,  
    "ObservationName" text,  
    "Label" text,  
    "HeatFlowType" text,  
    "HeaderURI" text,  
    "APINo" character varying(18),  
    "WellName" character varying,  
    "OtherID" text,  
    "Owner" character varying(128),  
    "LeaseName" character varying(128),  
    "Field" character varying(128),  
    "County" character varying,  
    "State" character varying,  
    "LatDegreeWGS84" numeric(10,7),  
    "LongDegreeWGS84" numeric(10,7),  
    "LocationUncertaintyStatement" character varying(128),  
    "PLSS_Meridians" character varying(64),  
    "Township" character varying(5),  
    "Range" character varying(5),  
    "Section_" character varying(10),  
    "SectionPart" character varying(5),  
    "TectonicProvince" character varying(128),  
    "OtherLocationName" text,  
    "PublicationSource" character varying(7),  
    "AdditionalSources" character varying(16),  
    "WellType" character varying,  
    "WellDrillDate" text,  
    "WellDrillDateType" character varying(16),  
    "Elevation" double precision,  
    "ElevationType" text,  
    "DepthUnits" character varying(1),  
    "DrillerTotalDepth" double precision,  
    "TimeSinceCirculation" double precision,  
    "TemperatureUnits" character varying(1),  
    "TemperatureMeasureType" character varying(7),  
    "SurfaceTemperature" double precision,  
    "BHT" double precision,  
    "BHTCorrectedValue" double precision,  
    "BHTCorrectionType" character varying(16),  
    "EqLog_MeasurementDate" date,  
    "EqLog_TotalDepth" double precision,
```

```

"EqLogTemperatureTop" double precision,
"EqLogTemperatureMax" numeric(10,7),
"EqLogTemperatureBottom" double precision,
"SiteHeatFlow" double precision,
"SiteValueMethod" character varying(7),
"Intervals" integer,
"IntervalTopDepth" double precision,
"IntervalBottomDepth" double precision,
"ThermalConductivityUnits" character varying(64),
"IntervalThermalConductivity" double precision,
"SE_Conductivity" double precision,
"ConductivitySampleType" character varying(128),
"ConductivityNumSamples" integer,
"ConductivityMeasuredURI" text,
"GradientUnits" character varying(4),
"IntervalUncorrectedGradient" double precision,
"SE_UncorrectedGradient" double precision,
"IntervalCorrectedGradient" double precision,
"SE_CorrectedGradient" double precision,
"HeatFlowUnits" character varying(5),
"IntervalUncorrectedHeatFlow" double precision,
"SE_UncorrectedHeatflow" double precision,
"IntervalCorrectedHeatFlow" double precision,
"SE_CorrectedHeatFlow" double precision,
"IntervalHF_QualityCode" character varying,
"HeatGenerationUnits" character varying(5),
"HeatGeneration" double precision,
"HG_NumberSamples" integer,
"WaterTableTop" double precision,
"WaterTableTemperature" double precision,
"LithologyInformation" text,
"FormationName" text,
"Notes" text,
"Shape" public.geometry
);

```

```

ALTER TABLE core.template_heatflow_materialized OWNER TO postgres;

```

```

--
-- Name: template_radiogenicheat; Type: VIEW; Schema: core; Owner: postgres
--

```

```

CREATE VIEW template_radiogenicheat AS
  WITH formation AS (SELECT formation.surface_site_id,
array_to_string(array_agg(DISTINCT (((formation.formation)::text || ' depth='::text) || CASE

```

```

WHEN (formation.formation_depth IS NULL) THEN 'NA'::text ELSE ('::text ||
formation.formation_depth) END)), '::text) AS formation FROM core.formation GROUP BY
formation.surface_site_id) SELECT
('https://geothermal.smu.edu/obervation/radiogenicheat/'::text || rad.id) AS "ObservationURI",
('Radiogenic Heat Production for well '::text || (COALESCE(s.site_name, s.api, (((s.latitude || '
)::text) || s.longitude))::character varying))::text) AS "ObservationName",
COALESCE(rad.specimen_label, 'nil:missing'::character varying) AS "SpecimenLabel",
f.formation AS "GeologicUnitName", s.lithology AS "LithologyTerm", rad.rock AS
"RockName", 'nil:missing'::character varying(16) AS "SamplingFeatureName",
tc.sample_collection_date AS "SampleCollectionDate", s.tectonic_province AS
"TectonicProvince", COALESCE(rc.county_name, 'nil:missing'::character varying) AS
"County", COALESCE(rs.state_name, 'nil:missing'::character varying) AS "State",
s.plss_meridians AS "PLSS_Meridians", s.township AS "Township", s.range AS "Range",
s.section AS "Section_", s.section_part AS "SectionPart", s.latitude AS "LatDegree",
s.longitude AS "LongDegree", 'WGS84'::character varying(6) AS "SRS",
COALESCE(s.location_notes, 'nil:missing'::character varying) AS
"LocationUncertaintyStatement", tc.sample_depth AS "SampleDepth_m",
tc.sample_measurement_date AS "SampleMeasurementDate", COALESCE(tc.sample_type,
'nil:missing'::character varying) AS "SampleType", ((-1))::double precision AS
"SampleMass_kg", rad.laboratory AS "Laboratory", COALESCE(rad.measurement_device,
'nil:missing'::character varying) AS "RadioactivityMeasurementDevice",
rad.measurement_procedure AS "MeasurementProcedure", 'nil:missing'::character varying(16)
AS "MeasurementCount", COALESCE(rad.uranium, ((-1))::double precision) AS "Uranium",
COALESCE(rad.thorium, ((-1))::double precision) AS "Thorium", COALESCE(rad.potassium,
((-1))::double precision) AS "Potassium", COALESCE(h.heat_generation, ((-1))::double
precision) AS "HeatProduction_mWm3", COALESCE((h.heat_generation_se)::text,
'nil:missing'::text) AS "HeatProductionUncertainty", rad.comments AS "Notes", CASE WHEN
((s.contributor IS NULL) AND (s.original_source IS NULL)) THEN (((ds.datasource)::text || '
)::text) || (ds.datatype)::text)::character varying WHEN (s.contributor IS NOT NULL) THEN
s.contributor ELSE s.original_source END AS "Source", s.shape AS "Shape" FROM
(((((((surface_site s JOIN radioactivity rad ON ((rad.surface_site_id = s.id))) LEFT JOIN
ref_county rc ON ((s.county_id = rc.id))) LEFT JOIN ref_state rs ON ((rc.state_id = rs.id)))
LEFT JOIN datasets ds ON ((s.dataset_id = ds.id))) LEFT JOIN formation f ON
((f.surface_site_id = s.id))) LEFT JOIN heat_flow h ON (((h.surface_site_id = s.id) AND
(h.radioactivity_id = rad.id)))) LEFT JOIN thermal_conductivity tc ON (((tc.surface_site_id =
s.id) AND (tc.id = h.thermal_conductivity_id)))));

```

```

ALTER TABLE core.template_radiogenicheat OWNER TO postgres;

```

```

--
-- Name: template_radiogenicheat_materialized; Type: TABLE; Schema: core; Owner:
postgres; Tablespace:
--

```

```

CREATE TABLE template_radiogenicheat_materialized (

```

```

"ObservationURI" text,
"ObservationName" text,
"SpecimenLabel" character varying,
"GeologicUnitName" text,
"LithologyTerm" character varying(255),
"RockName" character varying(64),
"SamplingFeatureName" character varying(16),
"SampleCollectionDate" date,
"TectonicProvince" character varying(128),
"County" character varying,
"State" character varying,
"PLSS_Meridians" character varying(64),
"Township" character varying(5),
"Range" character varying(5),
"Section_" character varying(10),
"SectionPart" character varying(5),
"LatDegree" numeric(10,7),
"LongDegree" numeric(10,7),
"SRS" character varying(6),
"LocationUncertaintyStatement" character varying,
"SampleDepth_m" double precision,
"SampleMeasurementDate" date,
"SampleType" character varying,
"SampleMass_kg" double precision,
"Laboratory" character varying(64),
"RadioactivityMeasurementDevice" character varying,
"MeasurementProcedure" character varying(64),
"MeasurementCount" character varying(16),
"Uranium" double precision,
"Thorium" double precision,
"Potassium" double precision,
"HeatProduction_mWm3" double precision,
"HeatProductionUncertainty" text,
"Notes" character varying(128),
"Source" character varying,
"Shape" public.geometry
);

```

```

ALTER TABLE core.template_radiogenicheat_materialized OWNER TO postgres;

```

```

--
-- Name: template_reservoir; Type: VIEW; Schema: core; Owner: postgres
--

```

```

CREATE VIEW template_reservoir AS

```

```

SELECT COALESCE(r.reservoiruri, 'nil:missing'::character varying) AS "ReservoirURI",
COALESCE(r.reservoirname, 'nil:missing'::character varying) AS "ReservoirName", CASE
WHEN ((r.label IS NULL) OR (length((r.label)::text) = 0)) THEN 'nil:missing'::character
varying ELSE r.label END AS "Label", r.otherid AS "OtherID", r.othername AS "OtherName",
r.description AS "Description", COALESCE(r.county, 'nil:missing'::character varying) AS
"County", COALESCE(r.state, 'nil:missing'::character varying) AS "State", r.locationname AS
"LocationName", r.latdegree AS "LatDegree", r.longdegree AS "LongDegree", r.srs AS "SRS",
CASE WHEN ((r.locationuncertaintystatement IS NULL) OR
(length((r.locationuncertaintystatement)::text) = 0)) THEN 'nil:missing'::character varying
ELSE r.locationuncertaintystatement END AS "LocationUncertaintyStatement",
r.temperaturemax_f AS "TemperatureMax_F", r.temperaturemin_f AS "TemperatureMin_F",
r.depthtotop_ft AS "DepthToTop_ft", r.depthtobottom_ft AS "DepthToBottom_ft",
r.dominantformationname AS "DominantFormationName", r.otherformationname AS
"OtherFormationName", r.dominantlithologyname AS "DominantLithologyName",
r.otherlithologyname AS "OtherLithologyName", r.area_acres AS "Area_acres", r.width_ft AS
"Width_ft", r.length_ft AS "Length_ft", r.sandvolume_cft AS "SandVolume_cft",
r.blockvolume_cft AS "BlockVolume_cft", r.percentsand AS "PercentSand",
r.averagenetsandthickness_ft AS "AverageNetSandThickness_ft", r.wellcount AS "WellCount",
r.faultcount AS "FaultCount", r.faulted AS "Faulted", r.faulttype AS "FaultType",
r.faultydrauliccharacter AS "FaultHydraulicCharacter", r.firstproductiondate AS
"FirstProductionDate", r.lastproductiondate AS "LastProductionDate", r.yearsinproduction AS
"YearsInProduction", r.monthlyproduction_bbl AS "MonthlyProduction_bbl",
r.productionreportdate AS "ProductionReportDate", r.averagemonthlywaterproduction_bbl AS
"AverageMonthlyWaterProduction_bbl", r.minimummonthlywaterproduction_bbl AS
"MinimumMonthlyWaterProduction_bbl", r.maximummonthlywaterproduction_bbl AS
"MaximumMonthlyWaterProduction_bbl", r.waterinjection_bbl AS "WaterInjection_bbl",
r.watersaturation_pct AS "WaterSaturation_pct", r.watertds_mgl AS "WaterTDS_mgl",
r.averagefieldwidepressure_psi AS "AverageFieldWidePressure_psi",
r.averagefieldwidegradient_degfft AS "AverageFieldWideGradient_degfft", r.porositymax_pct
AS "PorosityMax_pct", r.porositymin_pct AS "PorosityMin_pct", r.permeabilitymax_md AS
"PermeabilityMax_md", r.permeabilitymin_md AS "PermeabilityMin_md",
r.cementationmineralogy AS "CementationMineralogy", r.trap AS "Trap", r.drive AS "Drive",
r.watertds_mgl AS "WaterDensity", r.commodityofinterest AS "CommodityOfInterest",
r.stimulation AS "Stimulation", r.averageperforatedthickness_ft AS
"AveragePerforatedThickness_ft", r.relatedresource AS "RelatedResource", r.source AS
"Source", r.metadatauri AS "MetadataURI", r.shape AS "Shape" FROM reservoir r;

```

```

ALTER TABLE core.template_reservoir OWNER TO postgres;

```

```

--
-- Name: template_reservoir_materialized; Type: TABLE; Schema: core; Owner: postgres;
Tablespace:
--

```

```

CREATE TABLE template_reservoir_materialized (

```

"ReservoirURI" character varying,
"ReservoirName" character varying,
"Label" character varying,
"OtherID" character varying(255),
"OtherName" character varying(255),
"Description" character varying(255),
"County" character varying,
"State" character varying,
"LocationName" character varying(255),
"LatDegree" double precision,
"LongDegree" double precision,
"SRS" character varying(255),
"LocationUncertaintyStatement" character varying,
"TemperatureMax_F" double precision,
"TemperatureMin_F" double precision,
"DepthToTop_ft" double precision,
"DepthToBottom_ft" double precision,
"DominantFormationName" character varying(255),
"OtherFormationName" character varying(255),
"DominantLithologyName" character varying(255),
"OtherLithologyName" character varying(255),
"Area_acres" double precision,
"Width_ft" double precision,
"Length_ft" double precision,
"SandVolume_cft" double precision,
"BlockVolume_cft" double precision,
"PercentSand" double precision,
"AverageNetSandThickness_ft" double precision,
"WellCount" integer,
"FaultCount" integer,
"Faulted" character varying(255),
"FaultType" character varying(255),
"FaultHydraulicCharacter" character varying(255),
"FirstProductionDate" date,
"LastProductionDate" date,
"YearsInProduction" integer,
"MonthlyProduction_bbl" double precision,
"ProductionReportDate" date,
"AverageMonthlyWaterProduction_bbl" double precision,
"MinimumMonthlyWaterProduction_bbl" double precision,
"MaximumMonthlyWaterProduction_bbl" double precision,
"WaterInjection_bbl" double precision,
"WaterSaturation_pct" double precision,
"WaterTDS_mgl" double precision,
"AverageFieldWidePressure_psi" double precision,
"AverageFieldWideGradient_degfft" double precision,


```

"PorosityMax_pct" double precision,
"PorosityMin_pct" double precision,
"PermeabilityMax_mD" double precision,
"PermeabilityMin_mD" double precision,
"CementationMineralogy" character varying(255),
"Trap" character varying(255),
"Drive" character varying(255),
"WaterDensity" double precision,
"CommodityOfInterest" character varying(255),
"Stimulation" character varying(255),
"AveragePerforatedThickness_ft" double precision,
"RelatedResource" character varying(255),
"Source" character varying(1023),
"MetadataURI" character varying(255),
"Shape" public.geometry
);

```

```

ALTER TABLE core.template_reservoir_materialized OWNER TO postgres;

```

```

--
-- Name: template_thermalconductivity; Type: VIEW; Schema: core; Owner: postgres
--

```

```

CREATE VIEW template_thermalconductivity AS
  WITH formation AS (SELECT formation.surface_site_id,
    array_to_string(array_agg(DISTINCT formation.formation), '::text') AS formation,
    array_to_string(array_agg(DISTINCT formation.formation_type), '::text') AS formation_type
  FROM core.formation GROUP BY formation.surface_site_id), depth AS (SELECT
    depth.surface_site_id, max(depth.true_vertical_depth) AS true_vertical_depth,
    max(depth.depth) AS depth, array_to_string(array_agg(DISTINCT depth.depth_date), '::text')
  AS depth_date FROM core.depth GROUP BY depth.surface_site_id), wellinfo AS (SELECT
    well_info.surface_site_id, array_to_string(array_agg(DISTINCT well_info.commodity),
    '::text') AS commodity, array_to_string(array_agg(DISTINCT well_info.well_id), '::text') AS
    well_id FROM well_info GROUP BY well_info.surface_site_id) SELECT
    ('https://geothermal.smu.edu/obervation/conductivity/'::text || tc.id) AS "ObservationURI",
    ('Thermal conductivity observation for well '::text || (COALESCE(s.site_name, s.api))::text) AS
    "ObservationName", ('Thermal conductivity observation for well '::text ||
    (COALESCE(s.site_name, s.api))::text) AS "Label", ((COALESCE(s.other_ids, ' '::character
    varying))::text || COALESCE(wi.well_id, ' '::text)) AS "OtherID",
    COALESCE(tc.sample_depth, ((-99999))::double precision) AS "SampleDepth", 'm'::character
    varying(1) AS "MeasurementUnits", ('https://geothermal.smu.edu/header/'::text || s.id) AS
    "SiteLocationURI", COALESCE(s.site_name, s.api) AS "SiteLocationName",
    COALESCE(rc.county_name, 'nil:missing'::character varying) AS "County",
    COALESCE(rs.state_name, 'nil:missing'::character varying) AS "State", s.latitude AS
    "LatDegreeWGS84", s.longitude AS "LongDegreeWGS84", 'WGS84'::character varying(6) AS

```

```

"SRS", COALESCE(s.location_notes, 'nil:missing':character varying) AS
"LocationUncertaintyStatement", tc.sample_collection_date AS "SampleCollectionDate",
tc.sample_measurement_date AS "SampleMeasurementDate", COALESCE(tc.sample_type,
'nil:missing':character varying) AS "SampleType", COALESCE(tc.sample_shape,
'nil:missing':character varying) AS "SampleShape", tc.sample_height AS
"SampleDimensionsHeight", tc.sample_width AS "SampleDimensionsWidth",
tc.sample_length AS "SampleDimensionsLength", 'cm':character varying(2) AS
"SampleDimensionsUnits", tc.density AS "SampleDensity", tc.density_uncertainty AS
"UncertaintyDensity", tc.laboratory AS "Laboratory", COALESCE(tc.measurement_device,
'nil:missing':character varying) AS "MeasurementDevice", tc.measurement_procedure AS
"MeasurementProcedure", tc.measurement_pressure AS "MeasurementPressure",
tc.sample_temperature_diff AS "SampleTemperature", 'nil:missing':character varying(16) AS
"MeasurementRun", tc.conductivity AS "SaturatedSampleConductivity", tc.conductivity AS
"SiteThermalConductivity", tc.conductivity_se AS "UncertaintyTC", 'Watt per meter per
Kelvin':character varying(64) AS "UnitsTC", f.formation AS "GeologicFormationName",
'nil:missing':character varying(16) AS "GeologicFormationURI", f.formation_type AS
"GeologicFormationType", 'nil:missing':character varying(16) AS "DatumElevation", CASE
WHEN (s.elevation_kb IS NOT NULL) THEN s.elevation_kb WHEN (s.elevation_df IS NOT
NULL) THEN s.elevation_df WHEN (s.elevation_gl IS NOT NULL) THEN s.elevation_gl
ELSE NULL::double precision END AS "Elevation", COALESCE(tc.permeability, ((-
999999))::double precision) AS "Permeability", tc.porosity AS "Porosity", tc.comments AS
"Notes", CASE WHEN ((s.contributor IS NULL) AND (s.original_source IS NULL)) THEN
((((ds.datasource)::text || ' '::text) || (ds.datatype)::text)::character varying WHEN (s.contributor
IS NOT NULL) THEN s.contributor ELSE s.original_source END AS "Source", s.shape AS
"Shape" FROM ((((((thermal_conductivity tc JOIN surface_site s ON ((tc.surface_site_id =
s.id))) LEFT JOIN datasets ds ON ((s.dataset_id = ds.id))) LEFT JOIN ref_county rc ON
((s.county_id = rc.id))) LEFT JOIN ref_state rs ON ((rc.state_id = rs.id))) LEFT JOIN
formation f ON ((f.surface_site_id = s.id))) LEFT JOIN depth d ON ((d.surface_site_id = s.id)))
LEFT JOIN wellinfo wi ON ((wi.surface_site_id = s.id)));

```

```

ALTER TABLE core.template_thermalconductivity OWNER TO postgres;

```

```

--
-- Name: template_thermalconductivity_materialized; Type: TABLE; Schema: core; Owner:
postgres; Tablespace:
--

```

```

CREATE TABLE template_thermalconductivity_materialized (
    "ObservationURI" text,
    "ObservationName" text,
    "Label" text,
    "OtherID" text,
    "SampleDepth" double precision,
    "MeasurementUnits" character varying(1),
    "SiteLocationURI" text,

```

```

"SiteLocationName" character varying,
"County" character varying,
"State" character varying,
"LatDegreeWGS84" numeric(10,7),
"LongDegreeWGS84" numeric(10,7),
"SRS" character varying(6),
"LocationUncertaintyStatement" character varying,
"SampleCollectionDate" date,
"SampleMeasurementDate" date,
"SampleType" character varying,
"SampleShape" character varying,
"SampleDimensionsHeight" double precision,
"SampleDimensionsWidth" double precision,
"SampleDimensionsLength" double precision,
"SampleDimensionsUnits" character varying(2),
"SampleDensity" double precision,
"UncertaintyDensity" double precision,
"Laboratory" character varying(32),
"MeasurementDevice" character varying,
"MeasurementProcedure" character varying(128),
"MeasurementPressure" double precision,
"SampleTemperature" double precision,
"MeasurementRun" character varying(16),
"SaturatedSampleConductivity" double precision,
"SiteThermalConductivity" double precision,
"UncertaintyTC" double precision,
"UnitsTC" character varying(64),
"GeologicFormationName" text,
"GeologicFormationURI" character varying(16),
"GeologicFormationType" text,
"DatumElevation" character varying(16),
"Elevation" double precision,
"Permeability" double precision,
"Porosity" double precision,
"Notes" character varying(128),
"Source" character varying,
"Shape" public.geometry
);

```

```

ALTER TABLE core.template_thermalconductivity_materialized OWNER TO postgres;

```

```

--
-- Name: template_wellfluid; Type: VIEW; Schema: core; Owner: postgres
--

```

```

CREATE VIEW template_wellfluid AS
  WITH wellinfo AS (SELECT well_info.surface_site_id,
    array_to_string(array_agg(DISTINCT well_info.commodity), '|')::text AS commodity,
    array_to_string(array_agg(DISTINCT well_info.well_id), '|')::text AS well_id FROM well_info
  GROUP BY well_info.surface_site_id) SELECT (((('https://geothermal.smu.edu/wellfluid/'::text
  || s.id) || ';'::text) || "interval".top_depth) AS "ObservationURI", (((COALESCE(s.site_name,
  s.api))::text || 'top depth: '::text) || "interval".top_depth) AS "RecordName", s.operator AS
  "Operator", ('https://geothermal.smu.edu/header/'::text || s.id) AS "HeaderURI",
  COALESCE(s.site_name, s.api) AS "WellName", s.api AS "APINo", wi.well_id AS
  "OtherID", COALESCE(rc.county_name, 'nil:missing'::character varying) AS "County",
  COALESCE(rs.state_name, 'nil:missing'::character varying) AS "State", s.field AS "Field",
  s.other_location_name AS "OtherLocationName", s.plss_meridians AS "PLSS_Meridians",
  s.township AS "TWP", s.range AS "RGE", s.section AS "Section", s.section_part AS
  "SectionPart", s.latitude AS "LatDegreeWGS84", s.longitude AS "LongDegreeWGS84",
  'WGS84'::character varying(6) AS "SRS", COALESCE(s.location_notes,
  'nil:missing'::character varying) AS "LocationUncertaintyStatement", 'nil:missing'::character
  varying(16) AS "IntervalStartDateTime", 'nil:missing'::character varying(16) AS
  "IntervalEndDateTime", COALESCE(("interval".duration_hours / (24)::double precision),
  (0)::double precision) AS "DurationDays", s.lease_name AS "Permit", 'water'::character
  varying(5) AS "FluidType", "interval".water_volume AS "FluidVolume", 'barrels per
  day'::character varying(32) AS "FluidVolumeUnits", (-99999) AS "Temperature_F",
  COALESCE("interval".bhp, ((-99999))::double precision) AS "Pressure_psi",
  'nil:missing'::character varying(16) AS "AggregationType", 'nil:missing'::character varying(16)
  AS "MeasurementMethod", "interval".top_depth AS "ContactIntervalTop",
  "interval".bottom_depth AS "ContactIntervalBottom", CASE WHEN ((s.contributor IS NULL)
  AND (s.original_source IS NULL)) THEN (((ds.datasource)::text || ' '::text) ||
  (ds.datatype)::text)::character varying WHEN (s.contributor IS NOT NULL) THEN
  s.contributor ELSE s.original_source END AS "Source",
  ('https://geothermal.smu.edu/header/'::text || s.id) AS "RelatedResources", s.shape AS "Shape"
  FROM (((((surface_site s JOIN "interval" "interval" ON (("interval".surface_site_id = s.id)))
  LEFT JOIN datasets ds ON ((s.dataset_id = ds.id))) LEFT JOIN ref_county rc ON
  ((s.county_id = rc.id))) LEFT JOIN ref_state rs ON ((rc.state_id = rs.id))) LEFT JOIN wellinfo
  wi ON ((wi.surface_site_id = s.id)));

```

```

ALTER TABLE core.template_wellfluid OWNER TO postgres;

```

```

--
-- Name: template_wellfluid_materialized; Type: TABLE; Schema: core; Owner: postgres;
Tablespace:
--

```

```

CREATE TABLE template_wellfluid_materialized (
  "ObservationURI" text,
  "RecordName" text,
  "Operator" character varying(128),

```

```

"HeaderURI" text,
"WellName" character varying,
"APINo" character varying(18),
"OtherID" text,
"County" character varying,
"State" character varying,
"Field" character varying(128),
"OtherLocationName" character varying(256),
"PLSS_Meridians" character varying(64),
"TFW" character varying(5),
"RGE" character varying(5),
"Section" character varying(10),
"SectionPart" character varying(5),
"LatDegreeWGS84" numeric(10,7),
"LongDegreeWGS84" numeric(10,7),
"SRS" character varying(6),
"LocationUncertaintyStatement" character varying,
"IntervalStartDateTime" character varying(16),
"IntervalEndDateTime" character varying(16),
"DurationDays" double precision,
"Permit" character varying(128),
"FluidType" character varying(5),
"FluidVolume" double precision,
"FluidVolumeUnits" character varying(32),
"Temperature_F" integer,
"Pressure_psi" double precision,
"AggregationType" character varying(16),
"MeasurementMethod" character varying(16),
"ContactIntervalTop" double precision,
"ContactIntervalBottom" double precision,
"Source" character varying,
"RelatedResources" text,
"Shape" public.geometry
);

```

```

ALTER TABLE core.template_wellfluid_materialized OWNER TO postgres;

```

```

--
-- Name: template_wellheader; Type: VIEW; Schema: core; Owner: postgres
--

```

```

CREATE VIEW template_wellheader AS
  WITH depth AS (SELECT depth.surface_site_id, max(depth.true_vertical_depth) AS
true_vertical_depth, max(depth.depth) AS depth, max(depth.bit_diameter) AS bit_diameter,
array_to_string(array_agg(DISTINCT depth.depth_date), '|')::text AS depth_date FROM

```

```

core.depth GROUP BY depth.surface_site_id), formation AS (SELECT
formation.surface_site_id, array_to_string(array_agg(DISTINCT (((formation.formation)::text
|| ' depth='::text) || CASE WHEN (formation.formation_depth IS NULL) THEN 'NA'::text
ELSE ('::text || formation.formation_depth) END))), '|::text) AS formation FROM
core.formation GROUP BY formation.surface_site_id), casing AS (SELECT t.surface_site_id,
t.casing_depth, CASE WHEN ((t.casing_top)::text ~ '^[0-9]+$'::text) THEN
(t.casing_top)::numeric ELSE ((-999999))::numeric END AS casing_top, t.casing_size_inches,
t.casing_weight FROM (core.casing t JOIN (SELECT casing.surface_site_id,
max(casing.file_date) AS max_date, max(casing.casing_depth) AS max_depth FROM
core.casing GROUP BY casing.surface_site_id) x ON (((t.surface_site_id = x.surface_site_id)
AND ((x.max_date IS NULL) OR (t.file_date = x.max_date))) AND (t.casing_depth =
x.max_depth))))), wellinfo AS (SELECT well_info.surface_site_id,
array_to_string(array_agg(DISTINCT well_info.commodity), '|::text) AS commodity,
array_to_string(array_agg(DISTINCT well_info.well_id), '|::text) AS well_id FROM well_info
GROUP BY well_info.surface_site_id), welldates AS (SELECT well_dates.surface_site_id,
array_to_string(array_agg(DISTINCT well_dates.drl_compl_date), '|::text) AS drl_compl_date,
array_to_string(array_agg(DISTINCT well_dates.drl_start_date), '|::text) AS drl_start_date
FROM well_dates GROUP BY well_dates.surface_site_id), "interval" AS (SELECT
z.surface_site_id, z.producing_reservoir FROM (SELECT "interval".surface_site_id,
array_to_string(array_agg(DISTINCT "interval".producing_reservoir), '|::text) AS
producing_reservoir FROM core."interval" GROUP BY "interval".surface_site_id) z) SELECT
('https://geothermal.smu.edu/header/'::text || s.id) AS "HeaderURI", COALESCE(s.site_name,
s.api) AS "WellName", s.api AS "APINo", COALESCE(s.site_name, s.api) AS "Label",
(((COALESCE(s.other_ids, ' '::character varying))::text || COALESCE(wi.well_id, ' '::text)) AS
"OtherID", s.operator AS "Operator", s.site_owner AS "MineralOwner", s.lease_name AS
"LeaseName", COALESCE('::text || wd.drl_start_date)) AS "SpudDate",
COALESCE(d.depth_date, ('::text || wd.drl_compl_date)) AS "EndedDrillingDate", s.well_type
AS welltype, s.plugged AS "Status", COALESCE(s.well_type, (wi.commodity)::character
varying) AS "CommodityOfInterest", ('shut_in_date'::text || s.shut_in_date) AS "StatusDate",
iv.producing_reservoir AS "ProducingInterval", COALESCE(rc.county_name,
'nil:missing'::character varying) AS "County", COALESCE(rs.state_name,
'nil:missing'::character varying) AS "State", s.field AS "Field",
((((COALESCE(s.tectonic_province, ' '::character varying))::text || ';'::text) ||
(COALESCE(s.ams, ' '::character varying))::text) || ';'::text) ||
(COALESCE(s.other_location_name, ' '::character varying))::text) AS "OtherLocationName",
s.plss_meridians AS "PLSS_Meridians", s.township AS "Township", s.range AS "Range",
s.section AS "Section", s.section_part AS "SectionPart", s.latitude AS "LatDegreeWGS84",
s.longitude AS "LongDegreeWGS84", 'WGS84'::character varying(6) AS "SRS",
COALESCE(s.location_notes, 'nil:missing'::character varying) AS
"LocationUncertaintyStatement", d.depth AS "DrillerTotalDepth", '::character varying(1) AS
"DepthReferencePoint", 'm'::character varying(1) AS "LengthUnits", '::character varying(1) AS
"WellBoreShape", d.true_vertical_depth AS "TrueVerticalDepth", s.elevation_kb AS
"ElevationKB", s.elevation_df AS "ElevationDF", s.elevation_gl AS "ElevationGL",
f.formation AS "FormationTD", d.bit_diameter AS "BitDiameterCollar", 'inches'::character
varying(6) AS "DiameterUnits", s.comment AS "Notes", cs.casing_depth AS
"CasingBottomDepthDriller", cs.casing_top AS "CasingTopDepth", cs.casing_size_inches AS

```

```
"CasingPipeDiameter", cs.casing_weight AS "CasingWeight", 'lbs/ft'::character varying(6) AS
"CasingWeightUnits", CASE WHEN ((s.contributor IS NULL) AND (s.original_source IS
NULL)) THEN (((ds.datasource)::text || ' '::text) || (ds.datatype)::text)::character varying
WHEN (s.contributor IS NOT NULL) THEN s.contributor ELSE s.original_source END AS
"InformationSource", s.shape AS "Shape" FROM (((((((((surface_site s LEFT JOIN datasets ds
ON ((s.dataset_id = ds.id))) LEFT JOIN ref_county rc ON ((s.county_id = rc.id))) LEFT JOIN
ref_state rs ON ((rc.state_id = rs.id))) LEFT JOIN depth d ON ((d.surface_site_id = s.id)))
LEFT JOIN formation f ON ((f.surface_site_id = s.id))) LEFT JOIN casing cs ON
((cs.surface_site_id = s.id))) LEFT JOIN wellinfo wi ON ((wi.surface_site_id = s.id))) LEFT
JOIN welldates wd ON ((wd.surface_site_id = s.id))) LEFT JOIN "interval" iv ON
((iv.surface_site_id = s.id)));
```

```
ALTER TABLE core.template_wellheader OWNER TO postgres;
```

```
--
-- Name: template_wellheader_materialized; Type: TABLE; Schema: core; Owner: postgres;
Tablespace:
--
```

```
CREATE TABLE template_wellheader_materialized (
    "HeaderURI" text,
    "WellName" character varying,
    "APINo" character varying(18),
    "Label" character varying,
    "OtherID" text,
    "Operator" character varying(128),
    "MineralOwner" character varying(128),
    "LeaseName" character varying(128),
    "SpudDate" text,
    "EndedDrillingDate" text,
    welltype character varying(128),
    "Status" character varying(64),
    "CommodityOfInterest" character varying,
    "StatusDate" text,
    "ProducingInterval" text,
    "County" character varying,
    "State" character varying,
    "Field" character varying(128),
    "OtherLocationName" text,
    "PLSS_Meridians" character varying(64),
    "Township" character varying(5),
    "Range" character varying(5),
    "Section" character varying(10),
    "SectionPart" character varying(5),
    "LatDegreeWGS84" numeric(10,7),
```

```

"LongDegreeWGS84" numeric(10,7),
"SRS" character varying(6),
"LocationUncertaintyStatement" character varying,
"DrillerTotalDepth" double precision,
"DepthReferencePoint" character varying(1),
"LengthUnits" character varying(1),
"WellBoreShape" character varying(1),
"TrueVerticalDepth" double precision,
"ElevationKB" double precision,
"ElevationDF" double precision,
"ElevationGL" double precision,
"FormationTD" text,
"BitDiameterCollar" double precision,
"DiameterUnits" character varying(6),
"Notes" character varying(512),
"CasingBottomDepthDriller" double precision,
"CasingTopDepth" numeric,
"CasingPipeDiameter" double precision,
"CasingWeight" double precision,
"CasingWeightUnits" character varying(6),
"InformationSource" character varying,
"Shape" public.geometry
);

```

```

ALTER TABLE core.template_wellheader_materialized OWNER TO postgres;

```

```

--
-- Name: template_welllog; Type: VIEW; Schema: core; Owner: postgres
--

```

```

CREATE VIEW template_welllog AS
  WITH depth AS (SELECT depth.surface_site_id, max(depth.true_vertical_depth) AS
true_vertical_depth, max(depth.depth) AS depth, max(depth.bit_diameter) AS bit_diameter,
array_to_string(array_agg(DISTINCT depth.depth_date), '::text) AS depth_date FROM
core.depth GROUP BY depth.surface_site_id) SELECT
('http://geothermal.smu.edu/gtda/rest/fileSystem/getFile?uuid='::text || res.resource_id) AS
"LogURI", COALESCE(res.label, 'nil:missing'::character varying) AS "DisplayName",
(((res.type)::text || ':'::text) || (COALESCE(res.description, ''::character varying))::text) AS
"Notes", 'nil:missing'::character varying(11) AS "LogTypeTerm", 'nil:missing'::character
varying(11) AS "WellBoreURI", s.api AS "APINo", 'nil:missing'::character varying(11) AS
"WellType", COALESCE(rc.county_name, 'nil:missing'::character varying) AS "County",
COALESCE(rs.state_name, 'nil:missing'::character varying) AS "State", s.field AS "Field",
s.other_location_name AS "OtherLocationName", s.plss_meridians AS "PLSS_Meridians",
s.township AS "TWP", s.range AS "RGE", s.section AS "Section_", s.section_part AS
"SectionPart", s.latitude AS "LatDegree", s.longitude AS "LongDegree", 'WGS84'::character

```



```

varying(6) AS "SRS", COALESCE(s.location_notes, 'nil:missing'::character varying) AS
"LocationUncertaintyStatement", COALESCE((d.depth * (3.280839895)::double precision), ((-
999999)::double precision) AS "DrillerTotalDepth_ft", COALESCE((s.elevation_gl *
(3.280839895)::double precision), ((-999999)::double precision) AS "ElevationGL_ft", ((-
999999)::double precision AS "TopLoggedInterval_ft", ((-999999)::double precision AS
"BottomLoggedInterval_ft", 'nil:NA'::character varying(7) AS "DateTimeLogRun",
ds.datasource AS "PaperCopyLocation",
('http://geothermal.smu.edu/gtda/rest/fileSystem/getFile?uuid='::text || res.resource_id) AS
"ScannedFileURL", CASE WHEN ((s.contributor IS NULL) AND (s.original_source IS
NULL)) THEN (((ds.datasource)::text || ' '::text) || (ds.datatype)::text)::character varying
WHEN (s.contributor IS NOT NULL) THEN s.contributor ELSE s.original_source END AS
"Source", s.shape AS "Shape" FROM (((((surface_site s JOIN site_related_resources res ON
(((s.id = res.surface_site_id) OR public.st_contains(res.shape, s.shape)))) LEFT JOIN datasets
ds ON ((s.dataset_id = ds.id))) LEFT JOIN ref_county rc ON ((s.county_id = rc.id))) LEFT
JOIN ref_state rs ON ((rc.state_id = rs.id))) LEFT JOIN depth d ON ((d.surface_site_id =
s.id))) WHERE ((res.resource_id IS NOT NULL) AND ((res.type)::text = ANY
(ARRAY[('WELL_LOG'::character varying)::text, ('FIELDNOTE'::character varying)::text,
('TD_CURVES'::character varying)::text, ('WELL_LOG_HEADER'::character varying)::text,
('WellLog'::character varying)::text]]));

```

```

ALTER TABLE core.template_welllog OWNER TO postgres;

```

```

--
-- Name: template_welllog_materialized; Type: TABLE; Schema: core; Owner: postgres;
Tablespace:
--

```

```

CREATE TABLE template_welllog_materialized (
    "LogURI" text,
    "DisplayName" character varying,
    "Notes" text,
    "LogTypeTerm" character varying(11),
    "WellBoreURI" character varying(11),
    "APINo" character varying(18),
    "WellType" character varying(11),
    "County" character varying,
    "State" character varying,
    "Field" character varying(128),
    "OtherLocationName" character varying(256),
    "PLSS_Meridians" character varying(64),
    "TWP" character varying(5),
    "RGE" character varying(5),
    "Section_" character varying(10),
    "SectionPart" character varying(5),
    "LatDegree" numeric(10,7),

```

```

    "LongDegree" numeric(10,7),
    "SRS" character varying(6),
    "LocationUncertaintyStatement" character varying,
    "DrillerTotalDepth_ft" double precision,
    "ElevationGL_ft" double precision,
    "TopLoggedInterval_ft" double precision,
    "BottomLoggedInterval_ft" double precision,
    "DateTimeLogRun" character varying(7),
    "PaperCopyLocation" character varying(16),
    "ScannedFileURL" text,
    "Source" character varying,
    "Shape" public.geometry
);

```

```

ALTER TABLE core.template_welllog_materialized OWNER TO postgres;

```

```

--
-- Name: well_test; Type: TABLE; Schema: core; Owner: postgres; Tablespace:
--

```

```

CREATE TABLE well_test (
    id bigint NOT NULL,
    surface_site_id uuid,
    dataset_id uuid,
    well_info character varying(16),
    g10_id character varying(2),
    g10_rrcid integer NOT NULL,
    g10_tst_date date,
    g10_dly_gas integer,
    g10_dly_cond numeric(6,1),
    g10_dly_wtr integer,
    g10_siwh_press integer,
    g10_shut_cd character varying(1),
    g10_flow_press integer,
    g10_bhp integer,
    g10_shut_press integer,
    g10_opr character varying(6),
    g10_wh_pres_test_date date
);

```

```

ALTER TABLE core.well_test OWNER TO postgres;

```

```

--
-- Name: template_welltest; Type: VIEW; Schema: core; Owner: postgres

```

--

```
CREATE VIEW template_welltest AS
  SELECT (('https://geothermal.smu.edu/obervation/welltest/'::text || s.id) || wt.g10_tst_date)
  AS "ObservationURI", ('https://geothermal.smu.edu/header/'::text || s.id) AS "WellHeaderURI",
  COALESCE(s.site_name, s.api) AS "WellName", s.api AS "APINo",
  COALESCE(s.site_name, s.api, wt.well_info) AS "Label", wt.well_info AS "OtherID",
  'G10'::character varying(3) AS "TestName", 'Injection'::character varying(9) AS "TestType",
  wt.g10_opr AS "TestOperator", wt.g10_tst_date AS "TestDateTime", s.field AS "Field",
  COALESCE(rc.county_name, 'nil:missing'::character varying) AS "County",
  COALESCE(rs.state_name, 'nil:missing'::character varying) AS "State", s.plss_meridians AS
  "PLSS_Meridians", s.township AS "Township", s.range AS "Range", s.section AS "Section_",
  s.section_part AS "SectionPart", s.latitude AS "LatDegree", s.longitude AS "LongDegree",
  'WGS84'::character varying(6) AS "SRS", s.location_notes AS
  "LocationUncertaintyStatement", 'nil:missing'::character varying(12) AS "TargetFormation", ((-
  999999))::double precision AS "DepthTopOpenZone", ((-999999))::double precision AS
  "DepthBottomOpenZone", 'nil:missing'::character varying(12) AS "UnitsOfMeasure",
  'nil:missing'::character varying(12) AS "FlowHistory", ((-999999))::double precision AS
  "PressureInitialShutIn", ((-999999))::double precision AS "TimeInitialShutIn_min", ((-
  999999))::double precision AS "PreferredPressure", wt.g10_shut_press AS
  "PressureFinalShutIn", ((-999999))::double precision AS "TimeFinalShutIn_min",
  'PSI'::character varying(3) AS "Units_Pressure", ((-999999))::double precision AS
  "HydrostaticPressure", 'nil:missing'::character varying(12) AS "HydrostaticPressureMethod",
  ((-999999))::double precision AS "EquilibriumPressure", wt.g10_flow_press AS
  "FlowPressureMin", wt.g10_flow_press AS "FlowPressureMax", ((-999999))::double precision
  AS "TemperatureCorrection", CASE WHEN ((s.contributor IS NULL) AND (s.original_source
  IS NULL)) THEN (((ds.datasource)::text || ' '::text) || (ds.datatype)::text)::character varying
  WHEN (s.contributor IS NOT NULL) THEN s.contributor ELSE s.original_source END AS
  "Source", s.shape AS "Shape" FROM (((surface_site s JOIN well_test wt ON
  ((wt.surface_site_id = s.id))) LEFT JOIN datasets ds ON ((s.dataset_id = ds.id))) LEFT JOIN
  ref_county rc ON ((s.county_id = rc.id))) LEFT JOIN ref_state rs ON ((rc.state_id = rs.id)));
```

```
ALTER TABLE core.template_welltest OWNER TO postgres;
```

--

```
-- Name: template_welltest_materialized; Type: TABLE; Schema: core; Owner: postgres;
Tablespace:
```

--

```
CREATE TABLE template_welltest_materialized (
  "ObservationURI" text,
  "WellHeaderURI" text,
  "WellName" character varying,
  "APINo" character varying(18),
  "Label" character varying,
```

```

"OtherID" character varying(16),
"TestName" character varying(3),
"TestType" character varying(9),
"TestOperator" character varying(6),
"TestDateTime" date,
"Field" character varying(128),
"County" character varying,
"State" character varying,
"PLSS_Meridians" character varying(64),
"Township" character varying(5),
"Range" character varying(5),
"Section_" character varying(10),
"SectionPart" character varying(5),
"LatDegree" numeric(10,7),
"LongDegree" numeric(10,7),
"SRS" character varying(6),
"LocationUncertaintyStatement" character varying(128),
"TargetFormation" character varying(12),
"DepthTopOpenZone" double precision,
"DepthBottomOpenZone" double precision,
"UnitsOfMeasure" character varying(12),
"FlowHistory" character varying(12),
"PressureInitialShutIn" double precision,
"TimeInitialShutIn_min" double precision,
"PreferredPressure" double precision,
"PressureFinalShutIn" integer,
"TimeFinalShutIn_min" double precision,
"Units_Pressure" character varying(3),
"HydrostaticPressure" double precision,
"HydrostaticPressureMethod" character varying(12),
"EquilibriumPressure" double precision,
"FlowPressureMin" integer,
"FlowPressureMax" integer,
"TemperatureCorrection" double precision,
"Source" character varying,
"Shape" public.geometry
);

```

```

ALTER TABLE core.template_welltest_materialized OWNER TO postgres;

```

```

--
-- Name: temprature_series_mapping_id_seq; Type: SEQUENCE; Schema: core; Owner:
postgres
--

```

```
CREATE SEQUENCE temprature_series_mapping_id_seq
  START WITH 1
  INCREMENT BY 1
  NO MAXVALUE
  NO MINVALUE
  CACHE 1;
```

```
ALTER TABLE core.temprature_series_mapping_id_seq OWNER TO postgres;
```

```
--
-- Name: well_dates_id_seq; Type: SEQUENCE; Schema: core; Owner: postgres
--
```

```
CREATE SEQUENCE well_dates_id_seq
  START WITH 1
  INCREMENT BY 1
  NO MAXVALUE
  NO MINVALUE
  CACHE 1;
```

```
ALTER TABLE core.well_dates_id_seq OWNER TO postgres;
```

```
--
-- Name: well_dates_id_seq; Type: SEQUENCE OWNED BY; Schema: core; Owner: postgres
--
```

```
ALTER SEQUENCE well_dates_id_seq OWNED BY well_dates.id;
```

```
--
-- Name: well_info_id_seq; Type: SEQUENCE; Schema: core; Owner: postgres
--
```

```
CREATE SEQUENCE well_info_id_seq
  START WITH 1
  INCREMENT BY 1
  NO MAXVALUE
  NO MINVALUE
  CACHE 1;
```

```
ALTER TABLE core.well_info_id_seq OWNER TO postgres;
```

```
--
```

```
-- Name: well_info_id_seq; Type: SEQUENCE OWNED BY; Schema: core; Owner: postgres
--
```

```
ALTER SEQUENCE well_info_id_seq OWNED BY well_info.id;
```

```
--
-- Name: well_test_id_seq; Type: SEQUENCE; Schema: core; Owner: postgres
--
```

```
CREATE SEQUENCE well_test_id_seq
  START WITH 1
  INCREMENT BY 1
  NO MAXVALUE
  NO MINVALUE
  CACHE 1;
```

```
ALTER TABLE core.well_test_id_seq OWNER TO postgres;
```

```
--
-- Name: well_test_id_seq; Type: SEQUENCE OWNED BY; Schema: core; Owner: postgres
--
```

```
ALTER SEQUENCE well_test_id_seq OWNED BY well_test.id;
```

```
SET search_path = filesystem, pg_catalog;
```

```
--
-- Name: directories; Type: TABLE; Schema: filesystem; Owner: postgres; Tablespace:
--
```

```
CREATE TABLE directories (
  dir_uuid uuid NOT NULL,
  parent_uuid uuid,
  dir_name character varying(128) NOT NULL,
  dir_creation_date timestamp without time zone
);
```

```
ALTER TABLE filesystem.directories OWNER TO postgres;
```

```
SET search_path = staging, pg_catalog;
```

```
--
```

```
-- Name: countyp020_gid_seq; Type: SEQUENCE; Schema: staging; Owner: postgres
--
```

```
CREATE SEQUENCE countyp020_gid_seq
  START WITH 6138
  INCREMENT BY 1
  NO MAXVALUE
  NO MINVALUE
  CACHE 1;
```

```
ALTER TABLE staging.countyp020_gid_seq OWNER TO postgres;
```

```
--
-- Name: counties; Type: TABLE; Schema: staging; Owner: postgres; Tablespace:
--
```

```
CREATE TABLE counties (
  gid integer DEFAULT nextval('countyp020_gid_seq'::regclass) NOT NULL,
  area double precision,
  perimeter double precision,
  countyp020 double precision,
  state character varying(2),
  county character varying(50),
  state_fips character varying(2),
  square_mil double precision,
  the_geom public.geometry,
  vcount integer DEFAULT 0 NOT NULL,
  CONSTRAINT enforce_dims_the_geom CHECK ((public.st_ndims(the_geom) = 2)),
  CONSTRAINT enforce_geotype_the_geom CHECK (((public.geometrytype(the_geom) =
'MULTIPOLYGON'::text) OR (the_geom IS NULL))),
  CONSTRAINT enforce_srid_the_geom CHECK ((public.st_srid(the_geom) = 4326))
);
```

```
ALTER TABLE staging.counties OWNER TO postgres;
```

```
--
-- Name: states; Type: TABLE; Schema: staging; Owner: postgres; Tablespace:
--
```

```
CREATE TABLE states (
  gid integer NOT NULL,
  name character varying(25),
  fips_code character varying(2),
  sub_region character varying(7),
```

```

abbreviation character varying(2),
shape public.geometry,
vcount integer DEFAULT 0 NOT NULL,
CONSTRAINT enforce_dims_the_geom CHECK ((public.st_ndims(shape) = 2)),
CONSTRAINT enforce_geotype_the_geom CHECK (((public.geometrytype(shape) =
'MULTIPOLYGON'::text) OR (shape IS NULL))),
CONSTRAINT enforce_srid_the_geom CHECK ((public.st_srid(shape) = 4326))
);

```

```

ALTER TABLE staging.states OWNER TO postgres;

```

```

--
-- Name: beg_reservoir_data; Type: TABLE; Schema: staging; Owner: postgres; Tablespace:
--

```

```

CREATE TABLE beg_reservoir_data (
    reservoiruri character varying(255),
    reservoirname character varying(255),
    label character varying(255),
    otherid character varying(255),
    othername character varying(255),
    description character varying(255),
    county character varying(255),
    state character varying(255),
    locationname character varying(255),
    latdegree double precision,
    longdegree double precision,
    srs character varying(255),
    locationuncertainty statement character varying(255),
    temperaturemax_f double precision,
    temperaturemin_f double precision,
    depthtotop_ft double precision,
    depthtobottom_ft double precision,
    dominantformationname character varying(255),
    otherformationname character varying(255),
    dominantlithologyname character varying(255),
    otherlithologyname character varying(255),
    area_acres double precision,
    width_ft double precision,
    length_ft double precision,
    sandvolume_cft double precision,
    blockvolume_cft double precision,
    percentsand double precision,
    averagenetsandthickness_ft double precision,
    wellcount integer,

```



```

    faultcount integer,
    faulted character varying(255),
    faulttype character varying(255),
    faultyhydrauliccharacter character varying(255),
    firstproductiondate date,
    lastproductiondate date,
    yearsinproduction integer,
    monthlyproduction_bbl double precision,
    productionreportdate date,
    averagemonthlywaterproduction_bbl double precision,
    minimummonthlywaterproduction_bbl double precision,
    maximummonthlywaterproduction_bbl double precision,
    waterinjection_bbl double precision,
    watersaturation_pct double precision,
    watertds_mgl double precision,
    averagefieldwidepressure_psi double precision,
    averagefieldwidegradient_degfft double precision,
    porositymax_pct double precision,
    porositymin_pct double precision,
    permeabilitymax_md double precision,
    permeabilitymin_md double precision,
    cementationmineralogy character varying(255),
    trap character varying(255),
    drive character varying(255),
    gravity double precision,
    commodityofinterest character varying(255),
    stimulation character varying(255),
    averageperforatedthickness_ft double precision,
    relatedresource character varying(255),
    source character varying(1023),
    metadatauri character varying(255),
    majorfields character varying(511),
    cordinate_nw_long double precision,
    cordinate_nw_lat double precision,
    cordinate_ne_long double precision,
    cordinate_ne_lat double precision,
    cordinate_sw_long double precision,
    cordinate_sw_lat double precision,
    cordinate_se_long double precision,
    cordinate_se_lat double precision
);

```

```

ALTER TABLE staging.beg_reservoir_data OWNER TO postgres;

```

```

--

```

-- Name: beg_reservoir_view; Type: VIEW; Schema: staging; Owner: postgres
--

CREATE VIEW beg_reservoir_view AS

SELECT beg_reservoir_data.reservoiruri, beg_reservoir_data.reservoirname,
beg_reservoir_data.label, beg_reservoir_data.otherid, beg_reservoir_data.othername,
beg_reservoir_data.description, beg_reservoir_data.county, beg_reservoir_data.state,
beg_reservoir_data.locationname, beg_reservoir_data.latdegree, beg_reservoir_data.longdegree,
beg_reservoir_data.srs, beg_reservoir_data.locationuncertaintystatement,
beg_reservoir_data.temperaturemax_f, beg_reservoir_data.temperaturemin_f,
beg_reservoir_data.depthtotop_ft, beg_reservoir_data.depthtobottom_ft,
beg_reservoir_data.dominantformationname, beg_reservoir_data.otherformationname,
beg_reservoir_data.dominantlithologyname, beg_reservoir_data.otherlithologyname,
beg_reservoir_data.area_acres, beg_reservoir_data.width_ft, beg_reservoir_data.length_ft,
beg_reservoir_data.sandvolume_cft, beg_reservoir_data.blockvolume_cft,
beg_reservoir_data.percentsand, beg_reservoir_data.averagenetsandthickness_ft,
beg_reservoir_data.wellcount, beg_reservoir_data.faultcount, beg_reservoir_data.faulted,
beg_reservoir_data.faulttype, beg_reservoir_data.faultyhydrauliccharacter,
beg_reservoir_data.firstproductiondate, beg_reservoir_data.lastproductiondate,
beg_reservoir_data.yearsinproduction, beg_reservoir_data.monthlyproduction_bbl,
beg_reservoir_data.productionreportdate,
beg_reservoir_data.averagemonthlywaterproduction_bbl,
beg_reservoir_data.minimummonthlywaterproduction_bbl,
beg_reservoir_data.maximummonthlywaterproduction_bbl,
beg_reservoir_data.waterinjection_bbl, beg_reservoir_data.watersaturation_pct,
beg_reservoir_data.watertds_mgl, beg_reservoir_data.averagefieldwidepressure_psi,
beg_reservoir_data.averagefieldwidegradient_degfft, beg_reservoir_data.porositymax_pct,
beg_reservoir_data.porositymin_pct, beg_reservoir_data.permeabilitymax_md,
beg_reservoir_data.permeabilitymin_md, beg_reservoir_data.cementationmineralogy,
beg_reservoir_data.trap, beg_reservoir_data.drive, beg_reservoir_data.gravity,
beg_reservoir_data.commodityofinterest, beg_reservoir_data.stimulation,
beg_reservoir_data.averageperforatedthickness_ft, beg_reservoir_data.relatedresource,
beg_reservoir_data.source, beg_reservoir_data.metadatauri, beg_reservoir_data.majorfields,
beg_reservoir_data.cordinate_nw_long, beg_reservoir_data.cordinate_nw_lat,
beg_reservoir_data.cordinate_ne_long, beg_reservoir_data.cordinate_ne_lat,
beg_reservoir_data.cordinate_sw_long, beg_reservoir_data.cordinate_sw_lat,
beg_reservoir_data.cordinate_se_long, beg_reservoir_data.cordinate_se_lat FROM
beg_reservoir_data;

ALTER TABLE staging.beg_reservoir_view OWNER TO postgres;

--
-- Name: beg_reservoir_view_materialized; Type: TABLE; Schema: staging; Owner: postgres;
Tablespace:
--

```

CREATE TABLE beg_reservoir_view_materialized (
  reservoiruri character varying(255),
  reservoirname character varying(255),
  label character varying(255),
  otherid character varying(255),
  othername character varying(255),
  description character varying(255),
  county character varying(255),
  state character varying(255),
  locationname character varying(255),
  latdegree double precision,
  longdegree double precision,
  srs character varying(255),
  locationuncertainty statement character varying(255),
  temperaturemax_f double precision,
  temperaturemin_f double precision,
  depthtotop_ft double precision,
  depthtobottom_ft double precision,
  dominantformationname character varying(255),
  otherformationname character varying(255),
  dominantlithologyname character varying(255),
  otherlithologyname character varying(255),
  area_acres double precision,
  width_ft double precision,
  length_ft double precision,
  sandvolume_cft double precision,
  blockvolume_cft double precision,
  percentsand double precision,
  averagenetsandthickness_ft double precision,
  wellcount integer,
  faultcount integer,
  faulted character varying(255),
  faulttype character varying(255),
  faulthydrauliccharacter character varying(255),
  firstproductiondate date,
  lastproductiondate date,
  yearsinproduction integer,
  monthlyproduction_bbl double precision,
  productionreportdate date,
  averagemonthlywaterproduction_bbl double precision,
  minimummonthlywaterproduction_bbl double precision,
  maximummonthlywaterproduction_bbl double precision,
  waterinjection_bbl double precision,
  watersaturation_pct double precision,
  watertds_mgl double precision,

```

```

averagefieldwidepressure_psi double precision,
averagefieldwidegradient_degfft double precision,
porositymax_pct double precision,
porositymin_pct double precision,
permeabilitymax_md double precision,
permeabilitymin_md double precision,
cementationmineralogy character varying(255),
trap character varying(255),
drive character varying(255),
gravity double precision,
commodityofinterest character varying(255),
stimulation character varying(255),
averageperforatedthickness_ft double precision,
relatedresource character varying(255),
source character varying(1023),
metadatauri character varying(255),
majorfields character varying(511),
cordinate_nw_long double precision,
cordinate_nw_lat double precision,
cordinate_ne_long double precision,
cordinate_ne_lat double precision,
cordinate_sw_long double precision,
cordinate_sw_lat double precision,
cordinate_se_long double precision,
cordinate_se_lat double precision
);

ALTER TABLE staging.beg_reservoir_view_materialized OWNER TO postgres;

--
-- Name: beg_well_data; Type: TABLE; Schema: staging; Owner: postgres; Tablespace:
--

CREATE TABLE beg_well_data (
    api bigint,
    county character varying(64),
    lease character varying(64),
    formation character varying(64),
    fm_at_td character varying(64),
    lat double precision,
    long double precision,
    tvd double precision,
    depth double precision,
    bht double precision,
    depth_m double precision,

```

bht_c double precision,
 bht_c_cor double precision,
 bht_f_cor double precision,
 gradient_per_1000ft double precision,
 source character varying(64),
 wellname character varying(64),
 wellid character varying(64),
 logid character varying(32),
 number_run integer,
 log_date date,
 location character varying(256),
 operator_n character varying(64),
 accession_log_beg character varying(32),
 state_name character varying(32),
 casing_size_inches double precision,
 casing_depth double precision,
 liner_inches double precision,
 liner_top double precision,
 liner_base double precision,
 log_types character varying(128),
 in_core_db character varying(16),
 accession_core_beg character varying(32),
 prod_reservoir_1 character varying(64),
 prod_reservoir_2 character varying(64),
 prod_reservoir_3 character varying(64),
 prod_reservoir_4 character varying(64),
 prod_reservoir_5 character varying(64),
 prod_reservoir_6 character varying(64),
 prod_reservoir_7 character varying(64),
 prod_reservoir_8 character varying(64),
 sum_of_perfs double precision,
 top_ft_1 double precision,
 bottom_ft_1 double precision,
 interval_ft_1 double precision,
 top_ft_2 double precision,
 bottom_ft_2 double precision,
 interval_ft_2 double precision,
 top_ft_3 double precision,
 bottom_ft_3 double precision,
 interval_ft_3 double precision,
 top_ft_4 double precision,
 bottom_ft_4 double precision,
 interval_ft_4 double precision,
 top_ft_5 double precision,
 bottom_ft_5 double precision,
 interval_ft_5 double precision,

```

top_ft_6 double precision,
bottom_ft_6 double precision,
interval_ft_6 double precision,
top_ft_7 double precision,
bottom_ft_7 double precision,
interval_ft_7 double precision,
top_ft_8 double precision,
bottom_ft_8 double precision,
interval_ft_8 double precision,
watrvol_1_bbl double precision,
watrvol_2_bbl double precision,
watrvol_3_bbl double precision,
watrvol_4_bbl double precision,
watrvol_5_bbl double precision,
watrvol_6_bbl double precision,
watrvol_7_bbl double precision,
watrvol_8_bbl double precision,
bhp_1 double precision,
bhp_2 double precision,
bhp_3 double precision,
bhp_4 double precision,
bhp_5 double precision,
bhp_6 double precision,
bhp_7 double precision,
bhp_8 double precision,
durtn_hrs_1 double precision,
durtn_hrs_2 double precision,
durtn_hrs_3 double precision,
durtn_hrs_4 double precision,
durtn_hrs_5 double precision,
durtn_hrs_6 double precision,
durtn_hrs_7 double precision,
durtn_hrs_8 double precision,
testtype character varying(16),
choke_mm double precision
);

```

```

ALTER TABLE staging.beg_well_data OWNER TO postgres;

```

```

--
-- Name: beg_well_view; Type: VIEW; Schema: staging; Owner: postgres
--

```

```

CREATE VIEW beg_well_view AS

```

```

SELECT core.generate_uuid('BEG'::character varying, 'SURFACE_SITE'::character
varying, (((beg_well_data.lat || ','::text) || beg_well_data.long))::character varying) AS id,
beg_well_data.api, beg_well_data.county, beg_well_data.lease, beg_well_data.formation,
beg_well_data.fm_at_td, beg_well_data.lat, beg_well_data.long, beg_well_data.tvd,
beg_well_data.depth, beg_well_data.bht, beg_well_data.depth_m, beg_well_data.bht_c,
beg_well_data.bht_c_cor, beg_well_data.bht_f_cor, beg_well_data.gradient_per_1000ft,
beg_well_data.source, beg_well_data.wellname, beg_well_data.wellid, beg_well_data.logid,
beg_well_data.number_run, beg_well_data.log_date, beg_well_data.location,
beg_well_data.operator_n, beg_well_data.accession_log_beg, beg_well_data.state_name,
beg_well_data.casing_size_inches, beg_well_data.casing_depth, beg_well_data.liner_inches,
beg_well_data.liner_top, beg_well_data.liner_base, beg_well_data.log_types,
beg_well_data.in_core_db, beg_well_data.accession_core_beg,
beg_well_data.prod_reservoir_1, beg_well_data.prod_reservoir_2,
beg_well_data.prod_reservoir_3, beg_well_data.prod_reservoir_4,
beg_well_data.prod_reservoir_5, beg_well_data.prod_reservoir_6,
beg_well_data.prod_reservoir_7, beg_well_data.prod_reservoir_8,
beg_well_data.sum_of_perfs, beg_well_data.top_ft_1, beg_well_data.bottom_ft_1,
beg_well_data.interval_ft_1, beg_well_data.top_ft_2, beg_well_data.bottom_ft_2,
beg_well_data.interval_ft_2, beg_well_data.top_ft_3, beg_well_data.bottom_ft_3,
beg_well_data.interval_ft_3, beg_well_data.top_ft_4, beg_well_data.bottom_ft_4,
beg_well_data.interval_ft_4, beg_well_data.top_ft_5, beg_well_data.bottom_ft_5,
beg_well_data.interval_ft_5, beg_well_data.top_ft_6, beg_well_data.bottom_ft_6,
beg_well_data.interval_ft_6, beg_well_data.top_ft_7, beg_well_data.bottom_ft_7,
beg_well_data.interval_ft_7, beg_well_data.top_ft_8, beg_well_data.bottom_ft_8,
beg_well_data.interval_ft_8, beg_well_data.watrvol_1_bbl, beg_well_data.watrvol_2_bbl,
beg_well_data.watrvol_3_bbl, beg_well_data.watrvol_4_bbl, beg_well_data.watrvol_5_bbl,
beg_well_data.watrvol_6_bbl, beg_well_data.watrvol_7_bbl, beg_well_data.watrvol_8_bbl,
beg_well_data.bhp_1, beg_well_data.bhp_2, beg_well_data.bhp_3, beg_well_data.bhp_4,
beg_well_data.bhp_5, beg_well_data.bhp_6, beg_well_data.bhp_7, beg_well_data.bhp_8,
beg_well_data.durtn_hrs_1, beg_well_data.durtn_hrs_2, beg_well_data.durtn_hrs_3,
beg_well_data.durtn_hrs_4, beg_well_data.durtn_hrs_5, beg_well_data.durtn_hrs_6,
beg_well_data.durtn_hrs_7, beg_well_data.durtn_hrs_8, beg_well_data.testtype,
beg_well_data.choke_mm FROM beg_well_data;

```

```

ALTER TABLE staging.beg_well_view OWNER TO postgres;

```

```

--
-- Name: beg_well_view_materialized; Type: TABLE; Schema: staging; Owner: postgres;
Tablespace:
--

```

```

CREATE TABLE beg_well_view_materialized (
    id uuid,
    api bigint,
    county character varying(64),

```

lease character varying(64),
formation character varying(64),
fm_at_td character varying(64),
lat double precision,
long double precision,
tvd double precision,
depth double precision,
bht double precision,
depth_m double precision,
bht_c double precision,
bht_c_cor double precision,
bht_f_cor double precision,
gradient_per_1000ft double precision,
source character varying(64),
wellname character varying(64),
wellid character varying(64),
logid character varying(32),
number_run integer,
log_date date,
location character varying(256),
operator_n character varying(64),
accession_log_beg character varying(32),
state_name character varying(32),
casing_size_inches double precision,
casing_depth double precision,
liner_inches double precision,
liner_top double precision,
liner_base double precision,
log_types character varying(128),
in_core_db character varying(16),
accession_core_beg character varying(32),
prod_reservoir_1 character varying(64),
prod_reservoir_2 character varying(64),
prod_reservoir_3 character varying(64),
prod_reservoir_4 character varying(64),
prod_reservoir_5 character varying(64),
prod_reservoir_6 character varying(64),
prod_reservoir_7 character varying(64),
prod_reservoir_8 character varying(64),
sum_of_perfs double precision,
top_ft_1 double precision,
bottom_ft_1 double precision,
interval_ft_1 double precision,
top_ft_2 double precision,
bottom_ft_2 double precision,
interval_ft_2 double precision,


```
top_ft_3 double precision,  
bottom_ft_3 double precision,  
interval_ft_3 double precision,  
top_ft_4 double precision,  
bottom_ft_4 double precision,  
interval_ft_4 double precision,  
top_ft_5 double precision,  
bottom_ft_5 double precision,  
interval_ft_5 double precision,  
top_ft_6 double precision,  
bottom_ft_6 double precision,  
interval_ft_6 double precision,  
top_ft_7 double precision,  
bottom_ft_7 double precision,  
interval_ft_7 double precision,  
top_ft_8 double precision,  
bottom_ft_8 double precision,  
interval_ft_8 double precision,  
watrvol_1_bbl double precision,  
watrvol_2_bbl double precision,  
watrvol_3_bbl double precision,  
watrvol_4_bbl double precision,  
watrvol_5_bbl double precision,  
watrvol_6_bbl double precision,  
watrvol_7_bbl double precision,  
watrvol_8_bbl double precision,  
bhp_1 double precision,  
bhp_2 double precision,  
bhp_3 double precision,  
bhp_4 double precision,  
bhp_5 double precision,  
bhp_6 double precision,  
bhp_7 double precision,  
bhp_8 double precision,  
durtn_hrs_1 double precision,  
durtn_hrs_2 double precision,  
durtn_hrs_3 double precision,  
durtn_hrs_4 double precision,  
durtn_hrs_5 double precision,  
durtn_hrs_6 double precision,  
durtn_hrs_7 double precision,  
durtn_hrs_8 double precision,  
testtype character varying(16),  
choke_mm double precision  
);
```

```
ALTER TABLE staging.beg_well_view_materialized OWNER TO postgres;
```

```
--
```

```
-- Name: cornell_well_data; Type: TABLE; Schema: staging; Owner: postgres; Tablespace:
```

```
--
```

```
CREATE TABLE cornell_well_data (  
  api_wellno character varying(16),  
  bht_f double precision,  
  bht_c double precision,  
  harrison_correction double precision,  
  harrison_bht_c double precision,  
  log_depth_f double precision,  
  log_depth_m double precision,  
  tvd_f double precision,  
  tvd_m double precision,  
  calc_depth_m double precision,  
  harrison_corr_grad_c_km double precision,  
  uncorr_grad double precision,  
  county character varying(64),  
  cosuna_col character varying(8),  
  town character varying(64),  
  permit_number character varying(16),  
  operator character varying(128),  
  farm_name character varying(128),  
  td_date date,  
  log_date date,  
  comp_date date,  
  well_type character varying(64),  
  well_status character varying(64),  
  well_nm character varying(64),  
  elevation double precision,  
  field character varying(64),  
  pool character varying(64),  
  municipality character varying(64),  
  quad character varying(64),  
  map_section integer,  
  tdform character varying(64),  
  interval_id character varying(64),  
  site_heat_flow double precision,  
  site_value_method character varying(64),  
  interval_quality_code character varying(16),  
  decimal_longitude double precision,  
  decimal_latitude double precision,  
  sed_thickness_m double precision,
```

```

    heat_flow_mw_m2 double precision,
    average_conductivity double precision,
    data_source character varying(512),
    calc_basement_heat_prod double precision,
    temp_est_1500m double precision,
    temp_est_3000m double precision,
    temp_est_4500m double precision,
    temp_est_6000m double precision,
    temp_est_7500m double precision,
    temp_est_10000m double precision,
    est_depth_150c double precision,
    est_depth_80c double precision
);

```

```

ALTER TABLE staging.cornell_well_data OWNER TO postgres;

```

```

--
-- Name: cornell_well_view; Type: VIEW; Schema: staging; Owner: postgres
--

```

```

CREATE VIEW cornell_well_view AS
    SELECT core.generate_uuid('CORNELL'::character varying, 'SURFACE_SITE'::character
    varying, cornell_well_data.api_wellno) AS id, cornell_well_data.api_wellno,
    cornell_well_data.bht_f, cornell_well_data.bht_c, cornell_well_data.harrison_correction,
    cornell_well_data.harrison_bht_c, cornell_well_data.log_depth_f,
    cornell_well_data.log_depth_m, cornell_well_data.tvd_f, cornell_well_data.tvd_m,
    cornell_well_data.calc_depth_m, cornell_well_data.harrison_corr_grad_c_km,
    cornell_well_data.uncorr_grad, cornell_well_data.county, cornell_well_data.cosuna_col,
    cornell_well_data.town, cornell_well_data.permit_number, cornell_well_data.operator,
    cornell_well_data.farm_name, cornell_well_data.td_date, cornell_well_data.log_date,
    cornell_well_data.comp_date, cornell_well_data.well_type, cornell_well_data.well_status,
    cornell_well_data.well_nm, cornell_well_data.elevation, cornell_well_data.field,
    cornell_well_data.pool, cornell_well_data.municipality, cornell_well_data.quad,
    cornell_well_data.map_section, cornell_well_data.tdform, cornell_well_data.interval_id,
    cornell_well_data.site_heat_flow, cornell_well_data.site_value_method,
    cornell_well_data.interval_quality_code, cornell_well_data.decimal_longitude,
    cornell_well_data.decimal_latitude, cornell_well_data.sed_thickness_m,
    cornell_well_data.heat_flow_mw_m2, cornell_well_data.average_conductivity,
    cornell_well_data.data_source, cornell_well_data.calc_basement_heat_prod,
    cornell_well_data.temp_est_1500m, cornell_well_data.temp_est_3000m,
    cornell_well_data.temp_est_4500m, cornell_well_data.temp_est_6000m,
    cornell_well_data.temp_est_7500m, cornell_well_data.temp_est_10000m,
    cornell_well_data.est_depth_150c, cornell_well_data.est_depth_80c FROM cornell_well_data;

```

```
ALTER TABLE staging.cornell_well_view OWNER TO postgres;
```

```
--  
-- Name: cornell_well_view_materialized; Type: TABLE; Schema: staging; Owner: postgres;  
Tablespace:  
--
```

```
CREATE TABLE cornell_well_view_materialized (  
    id uuid,  
    api_wellno character varying(16),  
    bht_f double precision,  
    bht_c double precision,  
    harrison_correction double precision,  
    harrison_bht_c double precision,  
    log_depth_f double precision,  
    log_depth_m double precision,  
    tvd_f double precision,  
    tvd_m double precision,  
    calc_depth_m double precision,  
    harrison_corr_grad_c_km double precision,  
    uncorr_grad double precision,  
    county character varying(64),  
    cosuna_col character varying(8),  
    town character varying(64),  
    permit_number character varying(16),  
    operator character varying(128),  
    farm_name character varying(128),  
    td_date date,  
    log_date date,  
    comp_date date,  
    well_type character varying(64),  
    well_status character varying(64),  
    well_nm character varying(64),  
    elevation double precision,  
    field character varying(64),  
    pool character varying(64),  
    municipality character varying(64),  
    quad character varying(64),  
    map_section integer,  
    tdform character varying(64),  
    interval_id character varying(64),  
    site_heat_flow double precision,  
    site_value_method character varying(64),  
    interval_quality_code character varying(16),  
    decimal_longitude double precision,  
    decimal_latitude double precision,
```

```

    sed_thickness_m double precision,
    heat_flow_mw_m2 double precision,
    average_conductivity double precision,
    data_source character varying(512),
    calc_basement_heat_prod double precision,
    temp_est_1500m double precision,
    temp_est_3000m double precision,
    temp_est_4500m double precision,
    temp_est_6000m double precision,
    temp_est_7500m double precision,
    temp_est_10000m double precision,
    est_depth_150c double precision,
    est_depth_80c double precision
);

ALTER TABLE staging.cornell_well_view_materialized OWNER TO postgres;

--
-- Name: field_reports_id_seq; Type: SEQUENCE; Schema: staging; Owner: postgres
--

CREATE SEQUENCE field_reports_id_seq
    START WITH 4978
    INCREMENT BY 1
    NO MAXVALUE
    NO MINVALUE
    CACHE 1;

ALTER TABLE staging.field_reports_id_seq OWNER TO postgres;

--
-- Name: field_reports_mapping_id_seq; Type: SEQUENCE; Schema: staging; Owner:
postgres
--

CREATE SEQUENCE field_reports_mapping_id_seq
    START WITH 4978
    INCREMENT BY 1
    NO MAXVALUE
    NO MINVALUE
    CACHE 1;

ALTER TABLE staging.field_reports_mapping_id_seq OWNER TO postgres;

```

```

--
-- Name: matviews; Type: TABLE; Schema: staging; Owner: postgres; Tablespace:
--

CREATE TABLE matviews (
    mv_name name NOT NULL,
    v_name name NOT NULL,
    last_refresh timestamp with time zone
);

ALTER TABLE staging.matviews OWNER TO postgres;

--
-- Name: mlk_bht_data; Type: TABLE; Schema: staging; Owner: postgres; Tablespace:
--

CREATE TABLE mlk_bht_data (
    api42 integer,
    county_nr integer,
    drillertotaldepth double precision,
    operator character varying(128),
    leasename character varying(128),
    field character varying(128),
    measurementdatetime date,
    measuredtemperaturef double precision,
    casingpipediameter double precision,
    bitdiametercollar double precision,
    circulationduration double precision,
    mud_weight double precision,
    viscosity double precision
);

ALTER TABLE staging.mlk_bht_data OWNER TO postgres;

--
-- Name: publication; Type: TABLE; Schema: staging; Owner: postgres; Tablespace:
--

CREATE TABLE publication (
    publication_id integer NOT NULL,
    authors character varying(1024),
    title character varying(512),
    year character varying(64),

```

```
report_number character varying(100),
pages character varying(100),
agency character varying(512),
contract character varying(128),
source_number character varying(64),
source character varying(64),
location character varying(128),
pdf character varying(32),
electronic character varying(32),
url character varying(256),
month numeric(2,0),
publisher character varying(512),
volume character varying(128),
issue character varying(64),
collection character varying(256),
keywords character varying(1024),
publication_uuid uuid,
ocr_required boolean DEFAULT false,
expected boolean DEFAULT true,
dataset_id uuid,
data_source_id integer,
indexed boolean
);
```

```
ALTER TABLE staging.publication OWNER TO postgres;
```

```
--
-- Name: publication_publication_id_seq; Type: SEQUENCE; Schema: staging; Owner:
postgres
--
```

```
CREATE SEQUENCE publication_publication_id_seq
  START WITH 1
  INCREMENT BY 1
  NO MAXVALUE
  NO MINVALUE
  CACHE 1;
```

```
ALTER TABLE staging.publication_publication_id_seq OWNER TO postgres;
```

```
--
-- Name: publication_publication_id_seq; Type: SEQUENCE OWNED BY; Schema: staging;
Owner: postgres
--
```

```
ALTER SEQUENCE publication_publication_id_seq OWNED BY publication.publication_id;
```

```
--
```

```
-- Name: smu2_hf_heatflow; Type: TABLE; Schema: staging; Owner: postgres; Tablespace:
```

```
--
```

```
CREATE TABLE smu2_hf_heatflow (  
    gid character varying(255),  
    interval_id character varying(255),  
    depth double precision,  
    bottom double precision,  
    avg_tcu double precision,  
    avg_tcu_sym character varying(5),  
    avg_tcu_std_err double precision,  
    bulk_tcu double precision,  
    bulk_tcu_repeats numeric(7,0),  
    porosity double precision,  
    geoth_grad double precision,  
    geoth_grad_sym character varying(5),  
    geoth_grad_std_err double precision,  
    cgeoth_grad double precision,  
    cgeoth_grad_sym character varying(5),  
    cgeoth_grad_std_err double precision,  
    heat_flow double precision,  
    heat_flow_sym character varying(5),  
    heat_flow_std_err double precision,  
    cheat_flow double precision,  
    cheat_flow_sym character varying(5),  
    cheat_flow_std_err double precision,  
    heat_production double precision,  
    heat_production_sym character varying(5),  
    heat_production_std_err double precision,  
    num_hg double precision,  
    aquifier_temp double precision,  
    aquifier_temp_sym character varying(5),  
    aquifier_temp_std_err double precision,  
    quality_code character varying(10)  
);
```

```
ALTER TABLE staging.smu2_hf_heatflow OWNER TO postgres;
```

```
--
```

```
-- Name: smu2_hf_point; Type: TABLE; Schema: staging; Owner: postgres; Tablespace:
```


--

```
CREATE TABLE smu2_hf_point (  
  gid character varying(64) NOT NULL,  
  area_or_ams_map character varying(255),  
  hole_name character varying(255),  
  pub_ref character varying(255),  
  new_pub_ref character varying(255),  
  twn_rng_sec character varying(255),  
  county character varying(255),  
  state character varying(255),  
  tect_prov character varying(255),  
  w_long numeric(10,7),  
  n_lat numeric(10,7),  
  location_notes character varying(255),  
  meas_date date,  
  drill_date date,  
  collar_elevation double precision,  
  drill_depth double precision,  
  water_table double precision,  
  water_symbold character varying(255),  
  surf_temp double precision,  
  min_temp double precision,  
  max_temp double precision,  
  bot_temp double precision,  
  lit_info character varying(255),  
  comment_info character varying(255)  
);
```

```
ALTER TABLE staging.smu2_hf_point OWNER TO postgres;
```

--

-- Name: smu2_hf_ref; Type: TABLE; Schema: staging; Owner: postgres; Tablespace:

--

```
CREATE TABLE smu2_hf_ref (  
  pub_ref character varying(256),  
  new_pub_ref character varying(256),  
  info text,  
  smu_id character varying(256),  
  link character varying(512),  
  copyrighted character varying(8),  
  second_link character varying(256)  
);
```

```

ALTER TABLE staging.smu2_hf_ref OWNER TO postgres;

--
-- Name: smu_bht_data; Type: TABLE; Schema: staging; Owner: postgres; Tablespace:
--

CREATE TABLE smu_bht_data (
    database character varying(128),
    api character varying(128),
    permit character varying(128),
    longitude numeric(10,7),
    latitude numeric(10,7),
    surface_interval_id character varying(64),
    depth double precision,
    uncor_bht double precision,
    harrison_correction character varying(128),
    bhtcorrected_temp double precision,
    surface_temp double precision,
    harrison_gradient double precision,
    reference_data_set character varying(512),
    state character varying(128),
    k_source character varying(128),
    k_section character varying(128),
    resistance double precision,
    k double precision,
    q_calculated double precision,
    quality character varying(128),
    basement_depth double precision,
    correction character varying(128),
    bht_well double precision,
    well_ave_hf double precision,
    file_number character varying(128),
    operation_name character varying(128),
    drilling_start date,
    drilling_complete date,
    tsc double precision,
    state_plane_meter_x_coordinate character varying(128),
    state_plane_meter_y_coordinate character varying(128),
    company_name character varying(128),
    well_type character varying(128),
    well_status character varying(128),
    field_name character varying(128),
    formation character varying(128),
    elevation_measured character varying(128),
    elevation_ft double precision,

```

```

    elevation_m double precision,
    county_name character varying(128),
    surface_id character varying(128),
    bottom_id character varying(128)
);

```

```

ALTER TABLE staging.smu_bht_data OWNER TO postgres;

```

```

--
-- Name: smu_bht_view; Type: VIEW; Schema: staging; Owner: postgres
--

```

```

CREATE VIEW smu_bht_view AS
    SELECT core.generate_uuid('SMU'::character varying, 'SURFACE_SITE'::character
    varying, ((to_char(smu_bht_data.longitude, '999.9999'::text) || to_char(smu_bht_data.latitude,
    '999.9999'::text)))::character varying) AS id, smu_bht_data.database, smu_bht_data.api,
    smu_bht_data.permit, smu_bht_data.longitude, smu_bht_data.latitude,
    smu_bht_data.surface_interval_id, smu_bht_data.depth, smu_bht_data.uncor_bht,
    smu_bht_data.harrison_correction, smu_bht_data.bhtcorrected_temp,
    smu_bht_data.surface_temp, smu_bht_data.harrison_gradient,
    smu_bht_data.reference_data_set, smu_bht_data.state, smu_bht_data.k_source,
    smu_bht_data.k_section, smu_bht_data.resistance, smu_bht_data.k,
    smu_bht_data.q_calculated, smu_bht_data.quality, smu_bht_data.basement_depth,
    smu_bht_data.correction, smu_bht_data.bht_well, smu_bht_data.well_ave_hf,
    smu_bht_data.file_number, smu_bht_data.operation_name, smu_bht_data.drilling_start,
    smu_bht_data.drilling_complete, smu_bht_data.tsc,
    smu_bht_data.state_plane_meter_x_coordinate, smu_bht_data.state_plane_meter_y_coordinate,
    smu_bht_data.company_name, smu_bht_data.well_type, smu_bht_data.well_status,
    smu_bht_data.field_name, smu_bht_data.formation, smu_bht_data.elevation_measured,
    smu_bht_data.elevation_ft, smu_bht_data.elevation_m, smu_bht_data.county_name,
    smu_bht_data.surface_id, smu_bht_data.bottom_id FROM smu_bht_data;

```

```

ALTER TABLE staging.smu_bht_view OWNER TO postgres;

```

```

--
-- Name: smu_bht_view_materialized; Type: TABLE; Schema: staging; Owner: postgres;
Tablespace:
--

```

```

CREATE TABLE smu_bht_view_materialized (
    id uuid,
    database character varying(128),
    api character varying(128),
    permit character varying(128),

```

```

longitude numeric(10,7),
latitude numeric(10,7),
surface_interval_id character varying(64),
depth double precision,
uncor_bht double precision,
harrison_correction character varying(128),
bhtcorrected_temp double precision,
surface_temp double precision,
harrison_gradient double precision,
reference_data_set character varying(512),
state character varying(128),
k_source character varying(128),
k_section character varying(128),
resistance double precision,
k double precision,
q_calculated double precision,
quality character varying(128),
basement_depth double precision,
correction character varying(128),
bht_well double precision,
well_ave_hf double precision,
file_number character varying(128),
operation_name character varying(128),
drilling_start date,
drilling_complete date,
tsc double precision,
state_plane_meter_x_coordinate character varying(128),
state_plane_meter_y_coordinate character varying(128),
company_name character varying(128),
well_type character varying(128),
well_status character varying(128),
field_name character varying(128),
formation character varying(128),
elevation_measured character varying(128),
elevation_ft double precision,
elevation_m double precision,
county_name character varying(128),
surface_id character varying(128),
bottom_id character varying(128)
);

```

```

ALTER TABLE staging.smu_bht_view_materialized OWNER TO postgres;

```

```
--
```

```
-- Name: smu_hf_view; Type: VIEW; Schema: staging; Owner: postgres
```

--

CREATE VIEW smu_hf_view AS

```
SELECT core.generate_uuid('SMU'::character varying, 'SURFACE_SITE'::character
varying, pt.gid) AS id, pt.gid, pt.n_lat AS latitude, pt.w_long AS longitude,
pt.area_or_ams_map, pt.hole_name, pt.twn_rng_sec, pt.county, pt.state, pt.tect_prov,
pt.location_notes, pt.meas_date, pt.drill_date, pt.collar_elevation, pt.drill_depth, pt.water_table,
pt.water_symbold, pt.surf_temp, pt.min_temp, pt.max_temp, pt.bot_temp, pt.lit_info,
pt.comment_info, hf.interval_id, hf.depth, hf.bottom, hf.avg_tcu, hf.avg_tcu_sym,
hf.avg_tcu_std_err, hf.bulk_tcu, hf.bulk_tcu_repeats, hf.porosity, hf.geoth_grad,
hf.geoth_grad_sym, hf.geoth_grad_std_err, hf.cgeoth_grad, hf.cgeoth_grad_sym,
hf.cgeoth_grad_std_err, hf.heat_flow, hf.heat_flow_sym, hf.heat_flow_std_err, hf.cheat_flow,
hf.cheat_flow_sym, hf.cheat_flow_std_err, hf.heat_production, hf.heat_production_sym,
hf.heat_production_std_err, hf.num_hg, hf.aquifier_temp, hf.aquifier_temp_sym,
hf.aquifier_temp_std_err, hf.quality_code, ref.pub_ref, ref.new_pub_ref, ref.info, ref.smu_id,
ref.link, ref.copyrighted, ref.second_link FROM ((smu2_hf_point pt LEFT JOIN
smu2_hf_heatflow hf ON (((pt.gid)::text = (hf.gid)::text))) LEFT JOIN smu2_hf_ref ref ON
(((pt.new_pub_ref)::text = (ref.new_pub_ref)::text)));
```

ALTER TABLE staging.smu_hf_view OWNER TO postgres;

--

-- Name: smu_hf_view_materialized; Type: TABLE; Schema: staging; Owner: postgres;
Tablespace:

--

CREATE TABLE smu_hf_view_materialized (

```
id uuid,
gid character varying(64),
latitude numeric(10,7),
longitude numeric(10,7),
area_or_ams_map character varying(255),
hole_name character varying(255),
twn_rng_sec character varying(255),
county character varying(255),
state character varying(255),
tect_prov character varying(255),
location_notes character varying(255),
meas_date date,
drill_date date,
collar_elevation double precision,
drill_depth double precision,
water_table double precision,
water_symbold character varying(255),
surf_temp double precision,
```

```

min_temp double precision,
max_temp double precision,
bot_temp double precision,
lit_info character varying(255),
comment_info character varying(255),
interval_id character varying(255),
depth double precision,
bottom double precision,
avg_tcu double precision,
avg_tcu_sym character varying(5),
avg_tcu_std_err double precision,
bulk_tcu double precision,
bulk_tcu_repeats numeric(7,0),
porosity double precision,
geoth_grad double precision,
geoth_grad_sym character varying(5),
geoth_grad_std_err double precision,
cgeoth_grad double precision,
cgeoth_grad_sym character varying(5),
cgeoth_grad_std_err double precision,
heat_flow double precision,
heat_flow_sym character varying(5),
heat_flow_std_err double precision,
cheat_flow double precision,
cheat_flow_sym character varying(5),
cheat_flow_std_err double precision,
heat_production double precision,
heat_production_sym character varying(5),
heat_production_std_err double precision,
num_hg double precision,
aquifier_temp double precision,
aquifier_temp_sym character varying(5),
aquifier_temp_std_err double precision,
quality_code character varying(10),
pub_ref character varying(256),
new_pub_ref character varying(256),
info text,
smu_id character varying(256),
link character varying(512),
copyrighted character varying(8),
second_link character varying(256)
);

```

```

ALTER TABLE staging.smu_hf_view_materialized OWNER TO postgres;

```

```
--  
-- Name: states_gid_seq; Type: SEQUENCE; Schema: staging; Owner: postgres  
--
```

```
CREATE SEQUENCE states_gid_seq  
  START WITH 1  
  INCREMENT BY 1  
  NO MAXVALUE  
  NO MINVALUE  
  CACHE 1;
```

```
ALTER TABLE staging.states_gid_seq OWNER TO postgres;
```

```
--  
-- Name: states_gid_seq; Type: SEQUENCE OWNED BY; Schema: staging; Owner: postgres  
--
```

```
ALTER SEQUENCE states_gid_seq OWNED BY states.gid;
```

```
--  
-- Name: tt_corrected_bht; Type: TABLE; Schema: staging; Owner: postgres; Tablespace:  
--
```

```
CREATE TABLE tt_corrected_bht (  
  objectid integer,  
  block character varying(16),  
  api character varying(16),  
  ssf_depth double precision,  
  vrt double precision,  
  ssf_depth_ft double precision,  
  vrt_f double precision  
);
```

```
ALTER TABLE staging.tt_corrected_bht OWNER TO postgres;
```

```
--  
-- Name: tt_uncorrected_bht; Type: TABLE; Schema: staging; Owner: postgres; Tablespace:  
--
```

```
CREATE TABLE tt_uncorrected_bht (  
  objectid integer,  
  file_name character varying(64),  
  lease_area character varying(32),
```

```
addition character varying(32),
block character varying(32),
api character varying(16),
depth character varying(32),
bht character varying(32),
depth1 character varying(32),
bht1 character varying(32),
depth2 character varying(32),
bht2 character varying(32),
depth3 character varying(32),
bht3 character varying(32),
depth4 character varying(32),
bht4 character varying(32),
depth5 character varying(32),
bht5 character varying(32),
depth6 character varying(32),
bht6 character varying(32),
depth7 character varying(32),
bht7 character varying(32),
depth8 character varying(32),
bht8 character varying(32),
depth9 character varying(32),
bht9 character varying(32),
depth10 character varying(32),
bht10 character varying(32),
depth11 character varying(32),
bht11 character varying(32),
depth12 character varying(32),
bht12 character varying(32),
depth13 character varying(32),
bht13 character varying(32),
depth14 character varying(32),
bht14 character varying(32),
depth15 character varying(32),
bht15 character varying(32),
depth16 character varying(32),
bht16 character varying(32),
depth17 character varying(32),
bht17 character varying(32),
depth18 character varying(32),
bht18 character varying(32)
);
```

```
ALTER TABLE staging.tt_uncorrected_bht OWNER TO postgres;
```



```
--  
-- Name: tt_well_data; Type: TABLE; Schema: staging; Owner: postgres; Tablespace:  
--
```

```
CREATE TABLE tt_well_data (  
    objectid numeric,  
    shape character varying(128),  
    api character varying(16),  
    name_suf character varying(16),  
    well_name character varying(16),  
    field_name character varying(16),  
    operator character varying(16),  
    lease_num character varying(16),  
    spud_date character varying(16),  
    rkb_elev double precision,  
    td_md character varying(16),  
    s_northing character varying(16),  
    tvd character varying(16),  
    s_easting character varying(16),  
    s_ns character varying(16),  
    s_area character varying(16),  
    s_ew character varying(16),  
    b_northing character varying(16),  
    s_blk character varying(16),  
    b_easting character varying(16),  
    b_ns character varying(16),  
    b_area character varying(16),  
    b_ew character varying(16),  
    b_blk character varying(16),  
    st_date character varying(16),  
    td_date character varying(16),  
    district character varying(16),  
    type_code character varying(16),  
    status character varying(16),  
    s_long numeric(10,7),  
    w_depth character varying(16),  
    b_long numeric(10,7),  
    s_lat numeric(10,7),  
    b_lat numeric(10,7),  
    complex_id character varying(16),  
    s_lease_num character varying(16),  
    struc_id character varying(16),  
    order_status character varying(32)  
);
```

```
ALTER TABLE staging.tt_well_data OWNER TO postgres;
```

```
--  
-- Name: tt_well_view; Type: VIEW; Schema: staging; Owner: postgres  
--
```

```
CREATE VIEW tt_well_view AS  
  SELECT DISTINCT core.generate_uuid('TEXAS_TECH'::character varying,  
'SURFACE_SITE'::character varying, c.api) AS id, w.s_lat AS latitude, w.s_long AS longitude,  
  w.api, w.name_suf, w.well_name, w.field_name, w.operator, w.lease_num, w.spud_date,  
  w.rkb_elev, w.td_md, w.s_northing, w.tvd, w.s_easting, w.s_ns, w.s_area, w.s_ew,  
  w.b_northing, w.s_blk, w.b_easting, w.b_ns, w.b_area, w.b_ew, w.b_blk, w.st_date, w.td_date,  
  w.district, w.type_code, w.status, w.w_depth, w.b_long, w.b_lat, w.complex_id,  
  w.s_lease_num, w.struc_id, w.shape, w.order_status, u.file_name, u.lease_area, u.addition,  
  u.block, u.depth, u.bht, u.depth1, u.bht1, u.depth2, u.bht2, u.depth3, u.bht3, u.depth4, u.bht4,  
  u.depth5, u.bht5, u.depth6, u.bht6, u.depth7, u.bht7, u.depth8, u.bht8, u.depth9, u.bht9,  
  u.depth10, u.bht10, u.depth11, u.bht11, u.depth12, u.bht12, u.depth13, u.bht13, u.depth14,  
  u.bht14, u.depth15, u.bht15, u.depth16, u.bht16, u.depth17, u.bht17, u.depth18, u.bht18,  
  c.ssf_depth, c.vrt, c.ssf_depth_ft, c.vrt_f FROM ((tt_well_data w LEFT JOIN  
  tt_uncorrected_bht u ON (((w.api)::text = (u.api)::text))) LEFT JOIN tt_corrected_bht c ON  
  (((w.api)::text = (c.api)::text)));
```

```
ALTER TABLE staging.tt_well_view OWNER TO postgres;
```

```
--  
-- Name: tt_well_view_materialized; Type: TABLE; Schema: staging; Owner: postgres;  
Tablespace:  
--
```

```
CREATE TABLE tt_well_view_materialized (  
  id uuid,  
  latitude numeric(10,7),  
  longitude numeric(10,7),  
  api character varying(16),  
  name_suf character varying(16),  
  well_name character varying(16),  
  field_name character varying(16),  
  operator character varying(16),  
  lease_num character varying(16),  
  spud_date character varying(16),  
  rkb_elev double precision,  
  td_md character varying(16),  
  s_northing character varying(16),  
  tvd character varying(16),  
  s_easting character varying(16),
```

s_ns character varying(16),
s_area character varying(16),
s_ew character varying(16),
b_northing character varying(16),
s_blk character varying(16),
b_easting character varying(16),
b_ns character varying(16),
b_area character varying(16),
b_ew character varying(16),
b_blk character varying(16),
st_date character varying(16),
td_date character varying(16),
district character varying(16),
type_code character varying(16),
status character varying(16),
w_depth character varying(16),
b_long numeric(10,7),
b_lat numeric(10,7),
complex_id character varying(16),
s_lease_num character varying(16),
struc_id character varying(16),
shape character varying(128),
order_status character varying(32),
file_name character varying(64),
lease_area character varying(32),
addition character varying(32),
block character varying(32),
depth character varying(32),
bht character varying(32),
depth1 character varying(32),
bht1 character varying(32),
depth2 character varying(32),
bht2 character varying(32),
depth3 character varying(32),
bht3 character varying(32),
depth4 character varying(32),
bht4 character varying(32),
depth5 character varying(32),
bht5 character varying(32),
depth6 character varying(32),
bht6 character varying(32),
depth7 character varying(32),
bht7 character varying(32),
depth8 character varying(32),
bht8 character varying(32),
depth9 character varying(32),

```

bht9 character varying(32),
depth10 character varying(32),
bht10 character varying(32),
depth11 character varying(32),
bht11 character varying(32),
depth12 character varying(32),
bht12 character varying(32),
depth13 character varying(32),
bht13 character varying(32),
depth14 character varying(32),
bht14 character varying(32),
depth15 character varying(32),
bht15 character varying(32),
depth16 character varying(32),
bht16 character varying(32),
depth17 character varying(32),
bht17 character varying(32),
depth18 character varying(32),
bht18 character varying(32),
ssf_depth double precision,
vrt double precision,
ssf_depth_ft double precision,
vrt_f double precision
);

ALTER TABLE staging.tt_well_view_materialized OWNER TO postgres;

--
-- Name: und_rad_data; Type: TABLE; Schema: staging; Owner: postgres; Tablespace:
--

CREATE TABLE und_rad_data (
  observationuri character varying(255),
  observationname character varying(255),
  specimenlabel character varying(255),
  specimenuri character varying(255),
  geologicunitname character varying(255),
  featureofinteresturi character varying(255),
  lithologyterm character varying(255),
  rockname character varying(255),
  samplingfeatureuri character varying(255),
  samplingfeaturename character varying(255),
  samplecollectiondate date,
  localityterms character varying(255),
  tectonicprovince character varying(255),

```

```

county character varying(255),
state character varying(255),
plss_meridians character varying(255),
township character varying(255),
range character varying(255),
section character varying(255),
sectionpart character varying(255),
latdegree double precision,
longdegree double precision,
srs character varying(255),
locationuncertainty statement character varying(255),
elevationdatum_m double precision,
sampledepth_m double precision,
samplemeasurementdate date,
sampletype character varying(255),
samplemass_kg double precision,
overburdenthickness_m double precision,
sampledensity_g_cc double precision,
uncertaintydensity character varying(255),
laboratory character varying(255),
laboratoryuri character varying(255),
radioactivitymeasurementdevice character varying(255),
measurementprocedure character varying(255),
samplecureduration character varying(255),
measurementcount character varying(255),
uranium double precision,
thorium double precision,
potassium double precision,
heatproduction_mwm3 double precision,
heatproductionuncertainty character varying(255),
notes character varying(255),
metadatauri character varying(255),
source character varying(255),
identifier character varying(255)
);

```

```

ALTER TABLE staging.und_rad_data OWNER TO postgres;

```

```

--
-- Name: und_tc_data; Type: TABLE; Schema: staging; Owner: postgres; Tablespace:
--

```

```

CREATE TABLE und_tc_data (
    observationuri character varying(255),
    observationname character varying(255),

```

label character varying(255),
relatedsampleuri character varying(255),
otherid character varying(255),
sampledepth double precision,
measurementunits character varying(32),
sitelocationuri character varying(255),
sitelocationname character varying(255),
localityterms character varying(255),
county character varying(32),
state character varying(32),
latdegrewgs84 double precision,
longdegrewgs84 double precision,
srs character varying(32),
locationuncertaintystatement character varying(255),
samplecollectiondate date,
samplemeasurementdate date,
sampletype character varying(32),
sampleshape character varying(32),
sampledimensionsheight double precision,
sampledimensionswidth double precision,
sampledimensionslength double precision,
sampledimensionsunits character varying(32),
sampledensity double precision,
uncertaintydensity character varying(32),
sampledensityunits character varying(32),
laboratory character varying(255),
measurementdevice character varying(255),
measurementprocedure character varying(255),
measurementpressure double precision,
measurementpressureunits character varying(255),
sampletemperature double precision,
sampletemperatureunits character varying(255),
measurementduration double precision,
measurementdirection character varying(255),
measurementrun integer,
bulkmixingformula character varying(255),
saturationfluid character varying(255),
saturatedsampleconductivity double precision,
drysampleconductivity double precision,
averagethermalconductivity double precision,
uncertaintytc character varying(32),
correctiontc double precision,
correctiondescription character varying(255),
unitstc character varying(32),
specificeat double precision,
uncertaintyspecificeat double precision,

```

specificheatunits character varying(32),
thermaldiffusivity double precision,
uncertaintythermaldiffusivity double precision,
unitsthermaldiffusivity character varying(32),
geologicformationname character varying(255),
geologicformationuri character varying(255),
geologicformationtype character varying(255),
datumelevation date,
elevation double precision,
permeability double precision,
unitspermeability character varying(32),
porosity double precision,
notes character varying(255),
metadatauri character varying(255),
source character varying(255),
identifier character varying(255)
);

```

```

ALTER TABLE staging.und_tc_data OWNER TO postgres;

```

```

--
-- Name: und_rad_view; Type: VIEW; Schema: staging; Owner: postgres
--

```

```

CREATE VIEW und_rad_view AS

```

```

    SELECT core.generate_uuid('UND'::character varying, 'SURFACE_SITE'::character
varying, ((to_char(COALESCE(rad.latdegree, tc.latdegreewgs84), '999.999'::text) ||
to_char(COALESCE(rad.longdegree, tc.longdegreewgs84), '999.999'::text)))::character
varying) AS id, rad.latdegree AS latitude, rad.longdegree AS longitude, rad.observationuri,
rad.observationname, rad.specimenlabel, rad.specimenuri, rad.geologicunitname,
rad.featureofinteresturi, rad.lithologyterm, rad.rockname, rad.samplingfeatureuri,
rad.samplingfeaturename, rad.samplecollectiondate, rad.localityterms, rad.tectonicprovince,
rad.county, rad.state, rad.plss_meridians, rad.township, rad.range, rad.section, rad.sectionpart,
rad.latdegree, rad.longdegree, rad.srs, rad.locationuncertaintystatement, rad.elevationdatum_m,
rad.sampledepth_m, rad.samplemeasurementdate, rad.sampletype, rad.samplemass_kg,
rad.overburdenthickness_m, rad.sampledensity_g_cc, rad.uncertaintydensity, rad.laboratory,
rad.laboratoryuri, rad.radioactivitymeasurementdevice, rad.measurementprocedure,
rad.samplecureduration, rad.measurementcount, rad.uranium, rad.thorium, rad.potassium,
rad.heatproduction_mwm3, rad.heatproductionuncertainty, rad.notes, rad.metadatauri,
rad.source, rad.identifier FROM (und_tc_data tc FULL JOIN und_rad_data rad ON
(((tc.observationuri)::text = (rad.observationuri)::text))) WHERE ((rad.latdegree IS NOT
NULL) AND (rad.longdegree IS NOT NULL));

```

```

ALTER TABLE staging.und_rad_view OWNER TO postgres;

```

```
--  
-- Name: und_rad_view_materialized; Type: TABLE; Schema: staging; Owner: postgres;  
Tablespace:  
--
```

```
CREATE TABLE und_rad_view_materialized (  
  id uuid,  
  latitude double precision,  
  longitude double precision,  
  observationuri character varying(255),  
  observationname character varying(255),  
  specimenlabel character varying(255),  
  specimenuri character varying(255),  
  geologicunitname character varying(255),  
  featureofinteresturi character varying(255),  
  lithologyterm character varying(255),  
  rockname character varying(255),  
  samplingfeatureuri character varying(255),  
  samplingfeaturename character varying(255),  
  samplecollectiondate date,  
  localityterms character varying(255),  
  tectonicprovince character varying(255),  
  county character varying(255),  
  state character varying(255),  
  plss_meridians character varying(255),  
  township character varying(255),  
  range character varying(255),  
  section character varying(255),  
  sectionpart character varying(255),  
  latdegree double precision,  
  longdegree double precision,  
  srs character varying(255),  
  locationuncertaintystatement character varying(255),  
  elevationdatum_m double precision,  
  sampledepth_m double precision,  
  samplemeasurementdate date,  
  sampletype character varying(255),  
  samplemass_kg double precision,  
  overburdenthickness_m double precision,  
  sampledensity_g_cc double precision,  
  uncertaintydensity character varying(255),  
  laboratory character varying(255),  
  laboratoryuri character varying(255),  
  radioactivitymeasurementdevice character varying(255),  
  measurementprocedure character varying(255),
```



```

samplecureduration character varying(255),
measurementcount character varying(255),
uranium double precision,
thorium double precision,
potassium double precision,
heatproduction_mwm3 double precision,
heatproductionuncertainty character varying(255),
notes character varying(255),
metadatauri character varying(255),
source character varying(255),
identifier character varying(255)
);

```

```

ALTER TABLE staging.und_rad_view_materialized OWNER TO postgres;

```

```

--
-- Name: und_tc_view; Type: VIEW; Schema: staging; Owner: postgres
--

```

```

CREATE VIEW und_tc_view AS

```

```

    SELECT core.generate_uuid('UND')::character varying, 'SURFACE_SITE'::character
    varying, ((to_char(COALESCE(rad.latdegree, tc.latdegreewgs84), '999.999')::text) ||
    to_char(COALESCE(rad.longdegree, tc.longdegreewgs84), '999.999')::text))::character
    varying) AS id, tc.latdegreewgs84 AS latitude, tc.longdegreewgs84 AS longitude,
    tc.observationuri, tc.observationname, tc.label, tc.relatedsampleuri, tc.otherid, tc.sampledepth,
    tc.measurementunits, tc.sitelocationuri, tc.sitelocationname, tc.localityterms, tc.county, tc.state,
    tc.latdegreewgs84, tc.longdegreewgs84, tc.srs, tc.locationuncertaintystatement,
    tc.samplecollectiondate, tc.samplemeasurementdate, tc.sampletype, tc.sampleshape,
    tc.sampledimensionsheight, tc.sampledimensionswidth, tc.sampledimensionslength,
    tc.sampledimensionsunits, tc.sampledensity, tc.uncertaintydensity, tc.sampledensityunits,
    tc.laboratory, tc.measurementdevice, tc.measurementprocedure, tc.measurementpressure,
    tc.measurementpressureunits, tc.sampletemperature, tc.sampletemperatureunits,
    tc.measurementduration, tc.measurementdirection, tc.measurementrun, tc.bulkmixingformula,
    tc.saturationfluid, tc.saturatedsampleconductivity, tc.drysampleconductivity,
    tc.averagethermalconductivity, tc.uncertaintytc, tc.correctiontc, tc.correctiondescription,
    tc.unitstc, tc.specificeat, tc.uncertaintyspecificeat, tc.specificeatunits, tc.thermaldiffusivity,
    tc.uncertaintythermaldiffusivity, tc.unitsthermaldiffusivity, tc.geologicformationname,
    tc.geologicformationuri, tc.geologicformationtype, tc.datumelevation, tc.elevation,
    tc.permeability, tc.unitspermeability, tc.porosity, tc.notes, tc.metadatauri, tc.source, tc.identifier
    FROM (und_tc_data tc FULL JOIN und_rad_data rad ON (((tc.observationuri)::text =
    (rad.observationuri)::text))) WHERE ((tc.longdegreewgs84 IS NOT NULL) AND
    (tc.latdegreewgs84 IS NOT NULL));

```

```

ALTER TABLE staging.und_tc_view OWNER TO postgres;

```

```
--  
-- Name: und_tc_view_materialized; Type: TABLE; Schema: staging; Owner: postgres;  
Tablespace:  
--
```

```
CREATE TABLE und_tc_view_materialized (  
  id uuid,  
  latitude double precision,  
  longitude double precision,  
  observationuri character varying(255),  
  observationname character varying(255),  
  label character varying(255),  
  relatedsampleuri character varying(255),  
  otherid character varying(255),  
  sampledepth double precision,  
  measurementunits character varying(32),  
  sitelocationuri character varying(255),  
  sitelocationname character varying(255),  
  localityterms character varying(255),  
  county character varying(32),  
  state character varying(32),  
  latdegrewgs84 double precision,  
  longdegrewgs84 double precision,  
  srs character varying(32),  
  locationuncertaintystatement character varying(255),  
  samplecollectiondate date,  
  samplemeasurementdate date,  
  sampletype character varying(32),  
  sampleshape character varying(32),  
  sampledimensionsheight double precision,  
  sampledimensionswidth double precision,  
  sampledimensionslength double precision,  
  sampledimensionsunits character varying(32),  
  sampledensity double precision,  
  uncertaintydensity character varying(32),  
  sampledensityunits character varying(32),  
  laboratory character varying(255),  
  measurementdevice character varying(255),  
  measurementprocedure character varying(255),  
  measurementpressure double precision,  
  measurementpressureunits character varying(255),  
  sampletemperature double precision,  
  sampletemperatureunits character varying(255),  
  measurementduration double precision,  
  measurementdirection character varying(255),
```

```

measurementrun integer,
bulkmixingformula character varying(255),
saturationfluid character varying(255),
saturatedsampleconductivity double precision,
drysamplconductivity double precision,
averagethermalconductivity double precision,
uncertaintytc character varying(32),
correctiontc double precision,
correctiondescription character varying(255),
unitstc character varying(32),
specificeat double precision,
uncertaintyspecificeat double precision,
specificeatunits character varying(32),
thermaldiffusivity double precision,
uncertaintythermaldiffusivity double precision,
unitsthermaldiffusivity character varying(32),
geologicformationname character varying(255),
geologicformationuri character varying(255),
geologicformationtype character varying(255),
datumelevation date,
elevation double precision,
permeability double precision,
unitspermeability character varying(32),
porosity double precision,
notes character varying(255),
metadatauri character varying(255),
source character varying(255),
identifier character varying(255)
);

```

```

ALTER TABLE staging.und_tc_view_materialized OWNER TO postgres;

```

```

--
-- Name: und_td_curve_points; Type: TABLE; Schema: staging; Owner: postgres; Tablespace:
--

```

```

CREATE TABLE und_td_curve_points (
    curve_point_id integer NOT NULL,
    depth double precision,
    temperature double precision,
    comments_info text,
    sheet_name character varying(64),
    location_name character varying(64),
    hole_number character varying(64),
    operator_name character varying(64),

```

```

log_date timestamp without time zone,
latitude double precision,
longitude double precision,
trs character varying(128),
elevation_per_m double precision,
spud_date timestamp without time zone,
quadrangle_map character varying(128),
map_1_250k character varying(128),
probe_cable_bridge character varying(128),
identifier character varying(100)
);

ALTER TABLE staging.und_td_curve_points OWNER TO postgres;

--
-- Name: und_td_curve_points_curve_point_id_seq; Type: SEQUENCE; Schema: staging;
Owner: postgres
--

CREATE SEQUENCE und_td_curve_points_curve_point_id_seq
START WITH 1
INCREMENT BY 1
NO MAXVALUE
NO MINVALUE
CACHE 1;

ALTER TABLE staging.und_td_curve_points_curve_point_id_seq OWNER TO postgres;

--
-- Name: und_td_curve_points_curve_point_id_seq; Type: SEQUENCE OWNED BY;
Schema: staging; Owner: postgres
--

ALTER SEQUENCE und_td_curve_points_curve_point_id_seq OWNED BY
und_td_curve_points.curve_point_id;

--
-- Name: und_td_sites; Type: TABLE; Schema: staging; Owner: postgres; Tablespace:
--

CREATE TABLE und_td_sites (
sites_id integer NOT NULL,
identifier character varying(100) NOT NULL,

```

```

location_name character varying(64),
hole_number character varying(64),
label_description character varying(64),
latitude double precision,
longitude double precision,
srs character varying(64),
bht_c double precision,
depth_per_m double precision,
county character varying(64),
state character varying(32),
trs character varying(128),
elevation_per_m double precision,
company_owner character varying(256),
drill_depth_m double precision,
log_date timestamp without time zone,
map character varying(128),
quadrangle_map character varying(128),
probe_cable_bridge character varying(128),
operator_name character varying(64)
);

```

```

ALTER TABLE staging.und_td_sites OWNER TO postgres;

```

```

--
-- Name: und_td_sites_sites_id_seq; Type: SEQUENCE; Schema: staging; Owner: postgres
--

```

```

CREATE SEQUENCE und_td_sites_sites_id_seq
    START WITH 1
    INCREMENT BY 1
    NO MAXVALUE
    NO MINVALUE
    CACHE 1;

```

```

ALTER TABLE staging.und_td_sites_sites_id_seq OWNER TO postgres;

```

```

--
-- Name: und_td_sites_sites_id_seq; Type: SEQUENCE OWNED BY; Schema: staging;
Owner: postgres
--

```

```

ALTER SEQUENCE und_td_sites_sites_id_seq OWNED BY und_td_sites.sites_id;

```

```
--
-- Name: und_td_view; Type: VIEW; Schema: staging; Owner: postgres
--
```

```
CREATE VIEW und_td_view AS
```

```
    SELECT core.generate_uuid('UND'::character varying, 'SURFACE_SITE'::character
varying, ((to_char(tds.latitude, '999.999'::text) || to_char(tds.longitude,
'999.999'::text)))::character varying) AS id, tds.latitude, tds.longitude, tds.sites_id,
tds.identifier, tds.location_name AS sitelocationname, tds.hole_number AS siteholenumber,
tds.label_description, tds.latitude AS tdslatitude, tds.longitude AS tdslongitude, tds.srs AS
tdssrs, tds.bht_c, tds.depth_per_m, tds.county AS tdscounty, tds.state AS tdsstate, tds.trs,
tds.company_owner, tds.elevation_per_m AS siteelevation, tds.drill_depth_m, tds.log_date AS
sitelogdate, tds.map, tds.quadrangle_map AS siteqmap, tds.probe_cable_bridge AS
siteprobecablebridge, tds.operator_name AS siteoperatorname, tdc.sheet_name,
tdc.location_name AS curvelocationname, tdc.hole_number AS curveholenumber,
tdc.operator_name AS curveoperatorname, tdc.log_date AS curvelogdate, tdc.latitude AS
tdclatitude, tdc.longitude AS tdclongitude, tdc.trs AS tdc_srs, tdc.elevation_per_m AS
curveelevation, tdc.spud_date, tdc.quadrangle_map AS curveqmap, tdc.map_1_250k,
tdc.probe_cable_bridge AS curveprobecablebridge, tdc.depth, tdc.temperature,
tdc.comments_info FROM (und_td_sites tds LEFT JOIN und_td_curve_points tdc ON
((((tds.location_name)::text = (tdc.location_name)::text) OR ((tds.hole_number)::text =
(tdc.hole_number)::text)) OR ((tds.label_description)::text = (tdc.sheet_name)::text))))
WHERE (((tds.sites_id IS NOT NULL) AND (tdc.depth IS NOT NULL)) AND
(tdc.temperature IS NOT NULL));
```

```
ALTER TABLE staging.und_td_view OWNER TO postgres;
```

```
--
-- Name: und_td_view_materialized; Type: TABLE; Schema: staging; Owner: postgres;
Tablespace:
--
```

```
CREATE TABLE und_td_view_materialized (
```

```
    id uuid,
    latitude double precision,
    longitude double precision,
    sites_id integer,
    identifier character varying(100),
    sitelocationname character varying(64),
    siteholenumber character varying(64),
    label_description character varying(64),
    tdslatitude double precision,
    tdslongitude double precision,
    tdssrs character varying(64),
    bht_c double precision,
```

```

depth_per_m double precision,
tdscounty character varying(64),
tdsstate character varying(32),
trs character varying(128),
company_owner character varying(256),
siteelevation double precision,
drill_depth_m double precision,
sitelogdate timestamp without time zone,
map character varying(128),
siteqmap character varying(128),
siteprobecablebridge character varying(128),
siteoperatorname character varying(64),
sheet_name character varying(64),
curvelocationname character varying(64),
curveholenumber character varying(64),
curveoperatorname character varying(64),
curvelogdate timestamp without time zone,
tdclatitude double precision,
tdclongitude double precision,
tdcsrs character varying(128),
curveelevation double precision,
spud_date timestamp without time zone,
curveqmap character varying(128),
map_1_250k character varying(128),
curveprobecablebridge character varying(128),
depth double precision,
temperature double precision,
comments_info text
);

```

```

ALTER TABLE staging.und_td_view_materialized OWNER TO postgres;

```

```

--
-- Name: well_log_headers_id_seq; Type: SEQUENCE; Schema: staging; Owner: postgres
--

```

```

CREATE SEQUENCE well_log_headers_id_seq
  START WITH 4978
  INCREMENT BY 1
  NO MAXVALUE
  NO MINVALUE
  CACHE 1;

```

```

ALTER TABLE staging.well_log_headers_id_seq OWNER TO postgres;

```

```
--  
-- Name: well_log_headers_mapping_id_seq; Type: SEQUENCE; Schema: staging; Owner:  
postgres  
--
```

```
CREATE SEQUENCE well_log_headers_mapping_id_seq  
  START WITH 4978  
  INCREMENT BY 1  
  NO MAXVALUE  
  NO MINVALUE  
  CACHE 1;
```

```
ALTER TABLE staging.well_log_headers_mapping_id_seq OWNER TO postgres;
```

```
SET search_path = core, pg_catalog;
```

```
--  
-- Name: id; Type: DEFAULT; Schema: core; Owner: postgres  
--
```

```
ALTER TABLE aquifier ALTER COLUMN id SET DEFAULT  
nextval('aquifier_id_seq'::regclass);
```

```
--  
-- Name: id; Type: DEFAULT; Schema: core; Owner: postgres  
--
```

```
ALTER TABLE casing ALTER COLUMN id SET DEFAULT  
nextval('casing_id_seq'::regclass);
```

```
--  
-- Name: dataset_job_data_id; Type: DEFAULT; Schema: core; Owner: postgres  
--
```

```
ALTER TABLE dataset_job_data ALTER COLUMN dataset_job_data_id SET DEFAULT  
nextval('dataset_job_data_dataset_job_data_id_seq'::regclass);
```

```
--  
-- Name: datasets_map_id; Type: DEFAULT; Schema: core; Owner: postgres  
--
```



```
ALTER TABLE datasets_map ALTER COLUMN datasets_map_id SET DEFAULT
nextval('datasets_map_datasets_map_id_seq'::regclass);
```

```
--
-- Name: file_upload_id; Type: DEFAULT; Schema: core; Owner: postgres
--
```

```
ALTER TABLE file_upload ALTER COLUMN file_upload_id SET DEFAULT
nextval('file_upload_file_upload_id_seq'::regclass);
```

```
--
-- Name: id; Type: DEFAULT; Schema: core; Owner: postgres
--
```

```
ALTER TABLE fluid_h10 ALTER COLUMN id SET DEFAULT
nextval('fluid_h10_id_seq'::regclass);
```

```
--
-- Name: id; Type: DEFAULT; Schema: core; Owner: postgres
--
```

```
ALTER TABLE fluid_h10h ALTER COLUMN id SET DEFAULT
nextval('fluid_h10h_id_seq'::regclass);
```

```
--
-- Name: id; Type: DEFAULT; Schema: core; Owner: postgres
--
```

```
ALTER TABLE formation ALTER COLUMN id SET DEFAULT
nextval('formation_id_seq'::regclass);
```

```
--
-- Name: id; Type: DEFAULT; Schema: core; Owner: postgres
--
```

```
ALTER TABLE "interval" ALTER COLUMN id SET DEFAULT
nextval('interval_id_seq'::regclass);
```

```
--
-- Name: publication_id; Type: DEFAULT; Schema: core; Owner: postgres
```

--

```
ALTER TABLE publication ALTER COLUMN publication_id SET DEFAULT
nextval('publication_publication_id_seq'::regclass);
```

--

-- Name: record_id; Type: DEFAULT; Schema: core; Owner: postgres

--

```
ALTER TABLE site_related_resources ALTER COLUMN record_id SET DEFAULT
nextval('site_related_resources_record_id_seq'::regclass);
```

--

-- Name: id; Type: DEFAULT; Schema: core; Owner: postgres

--

```
ALTER TABLE well_dates ALTER COLUMN id SET DEFAULT
nextval('well_dates_id_seq'::regclass);
```

--

-- Name: id; Type: DEFAULT; Schema: core; Owner: postgres

--

```
ALTER TABLE well_info ALTER COLUMN id SET DEFAULT
nextval('well_info_id_seq'::regclass);
```

--

-- Name: id; Type: DEFAULT; Schema: core; Owner: postgres

--

```
ALTER TABLE well_test ALTER COLUMN id SET DEFAULT
nextval('well_test_id_seq'::regclass);
```

```
SET search_path = staging, pg_catalog;
```

--

-- Name: publication_id; Type: DEFAULT; Schema: staging; Owner: postgres

--

```
ALTER TABLE publication ALTER COLUMN publication_id SET DEFAULT
nextval('publication_publication_id_seq'::regclass);
```

```

--
-- Name: gid; Type: DEFAULT; Schema: staging; Owner: postgres
--

ALTER TABLE states ALTER COLUMN gid SET DEFAULT
nextval('states_gid_seq'::regclass);

--
-- Name: curve_point_id; Type: DEFAULT; Schema: staging; Owner: postgres
--

ALTER TABLE und_td_curve_points ALTER COLUMN curve_point_id SET DEFAULT
nextval('und_td_curve_points_curve_point_id_seq'::regclass);

--
-- Name: sites_id; Type: DEFAULT; Schema: staging; Owner: postgres
--

ALTER TABLE und_td_sites ALTER COLUMN sites_id SET DEFAULT
nextval('und_td_sites_sites_id_seq'::regclass);

SET search_path = core, pg_catalog;

--
-- Name: aquifier_pkey; Type: CONSTRAINT; Schema: core; Owner: postgres; Tablespace:
--

ALTER TABLE ONLY aquifier
  ADD CONSTRAINT aquifier_pkey PRIMARY KEY (id);

--
-- Name: bht_pkey; Type: CONSTRAINT; Schema: core; Owner: postgres; Tablespace:
--

ALTER TABLE ONLY bht_measurements
  ADD CONSTRAINT bht_pkey PRIMARY KEY (id);

--
-- Name: casing_pkey; Type: CONSTRAINT; Schema: core; Owner: postgres; Tablespace:

```

```

--

ALTER TABLE ONLY casing
  ADD CONSTRAINT casing_pkey PRIMARY KEY (id);

--
-- Name: county_pkey; Type: CONSTRAINT; Schema: core; Owner: postgres; Tablespace:
--

ALTER TABLE ONLY ref_county
  ADD CONSTRAINT county_pkey PRIMARY KEY (id);

--
-- Name: dataset_job_data_pkey; Type: CONSTRAINT; Schema: core; Owner: postgres;
Tablespace:
--

ALTER TABLE ONLY dataset_job_data
  ADD CONSTRAINT dataset_job_data_pkey PRIMARY KEY (dataset_job_data_id);

--
-- Name: datasets_map_pkey; Type: CONSTRAINT; Schema: core; Owner: postgres;
Tablespace:
--

ALTER TABLE ONLY datasets_map
  ADD CONSTRAINT datasets_map_pkey PRIMARY KEY (datasets_map_id);

--
-- Name: datasets_pkey; Type: CONSTRAINT; Schema: core; Owner: postgres; Tablespace:
--

ALTER TABLE ONLY datasets
  ADD CONSTRAINT datasets_pkey PRIMARY KEY (id);

--
-- Name: depth_pkey; Type: CONSTRAINT; Schema: core; Owner: postgres; Tablespace:
--

ALTER TABLE ONLY depth
  ADD CONSTRAINT depth_pkey PRIMARY KEY (id);

```

```

--
-- Name: file_upload_pkey; Type: CONSTRAINT; Schema: core; Owner: postgres;
Tablespace:
--

ALTER TABLE ONLY file_upload
  ADD CONSTRAINT file_upload_pkey PRIMARY KEY (file_upload_id);

--
-- Name: fluid_h10_pkey; Type: CONSTRAINT; Schema: core; Owner: postgres; Tablespace:
--

ALTER TABLE ONLY fluid_h10
  ADD CONSTRAINT fluid_h10_pkey PRIMARY KEY (id);

--
-- Name: fluid_h10h_pkey; Type: CONSTRAINT; Schema: core; Owner: postgres; Tablespace:
--

ALTER TABLE ONLY fluid_h10h
  ADD CONSTRAINT fluid_h10h_pkey PRIMARY KEY (id);

--
-- Name: fluid_pkey; Type: CONSTRAINT; Schema: core; Owner: postgres; Tablespace:
--

ALTER TABLE ONLY fluid
  ADD CONSTRAINT fluid_pkey PRIMARY KEY (id);

--
-- Name: formation_pkey; Type: CONSTRAINT; Schema: core; Owner: postgres; Tablespace:
--

ALTER TABLE ONLY formation
  ADD CONSTRAINT formation_pkey PRIMARY KEY (id);

--
-- Name: heatflow_pkey; Type: CONSTRAINT; Schema: core; Owner: postgres; Tablespace:
--

```

```

ALTER TABLE ONLY heat_flow
    ADD CONSTRAINT heatflow_pkey PRIMARY KEY (id);

--
-- Name: interval_pkey; Type: CONSTRAINT; Schema: core; Owner: postgres; Tablespace:
--

ALTER TABLE ONLY "interval"
    ADD CONSTRAINT interval_pkey PRIMARY KEY (id);

--
-- Name: matviews_pkey; Type: CONSTRAINT; Schema: core; Owner: postgres; Tablespace:
--

ALTER TABLE ONLY matviews
    ADD CONSTRAINT matviews_pkey PRIMARY KEY (mv_name);

--
-- Name: notes_pkey; Type: CONSTRAINT; Schema: core; Owner: postgres; Tablespace:
--

ALTER TABLE ONLY notes
    ADD CONSTRAINT notes_pkey PRIMARY KEY (id);

--
-- Name: precision_log_pkey; Type: CONSTRAINT; Schema: core; Owner: postgres;
Tablespace:
--

ALTER TABLE ONLY precision_log
    ADD CONSTRAINT precision_log_pkey PRIMARY KEY (id);

--
-- Name: publication_pkey; Type: CONSTRAINT; Schema: core; Owner: postgres;
Tablespace:
--

ALTER TABLE ONLY publication
    ADD CONSTRAINT publication_pkey PRIMARY KEY (publication_id);

```

```

--
-- Name: radioactivity_pkey; Type: CONSTRAINT; Schema: core; Owner: postgres;
Tablespace:
--

ALTER TABLE ONLY radioactivity
  ADD CONSTRAINT radioactivity_pkey PRIMARY KEY (id);

--
-- Name: ref_data_source_pkey; Type: CONSTRAINT; Schema: core; Owner: postgres;
Tablespace:
--

ALTER TABLE ONLY ref_data_source
  ADD CONSTRAINT ref_data_source_pkey PRIMARY KEY (id);

--
-- Name: ref_quality_issue_type_pkey; Type: CONSTRAINT; Schema: core; Owner: postgres;
Tablespace:
--

ALTER TABLE ONLY ref_quality_issue_type
  ADD CONSTRAINT ref_quality_issue_type_pkey PRIMARY KEY (id);

--
-- Name: ref_symbol_pkey; Type: CONSTRAINT; Schema: core; Owner: postgres;
Tablespace:
--

ALTER TABLE ONLY ref_symbol
  ADD CONSTRAINT ref_symbol_pkey PRIMARY KEY (id);

--
-- Name: reservoir_pkey; Type: CONSTRAINT; Schema: core; Owner: postgres; Tablespace:
--

ALTER TABLE ONLY reservoir
  ADD CONSTRAINT reservoir_pkey PRIMARY KEY (id);

--

```

-- Name: site_related_resources_pkey; Type: CONSTRAINT; Schema: core; Owner: postgres;
Tablespace:

--

ALTER TABLE ONLY site_related_resources
ADD CONSTRAINT site_related_resources_pkey PRIMARY KEY (record_id);

--

-- Name: state_pkey; Type: CONSTRAINT; Schema: core; Owner: postgres; Tablespace:

--

ALTER TABLE ONLY ref_state
ADD CONSTRAINT state_pkey PRIMARY KEY (id);

--

-- Name: surface_site_pkey; Type: CONSTRAINT; Schema: core; Owner: postgres;
Tablespace:

--

ALTER TABLE ONLY surface_site
ADD CONSTRAINT surface_site_pkey PRIMARY KEY (id);

--

-- Name: temperature_series_pkey; Type: CONSTRAINT; Schema: core; Owner: postgres;
Tablespace:

--

ALTER TABLE ONLY temperature_series
ADD CONSTRAINT temperature_series_pkey PRIMARY KEY (id);

--

-- Name: thermal_conductivity_pkey; Type: CONSTRAINT; Schema: core; Owner: postgres;
Tablespace:

--

ALTER TABLE ONLY thermal_conductivity
ADD CONSTRAINT thermal_conductivity_pkey PRIMARY KEY (id);

--

-- Name: well_dates_pkey; Type: CONSTRAINT; Schema: core; Owner: postgres; Tablespace:

--


```

ALTER TABLE ONLY well_dates
  ADD CONSTRAINT well_dates_pkey PRIMARY KEY (id);

--
-- Name: well_info_pkey; Type: CONSTRAINT; Schema: core; Owner: postgres; Tablespace:
--

ALTER TABLE ONLY well_info
  ADD CONSTRAINT well_info_pkey PRIMARY KEY (id);

--
-- Name: well_test_pkey; Type: CONSTRAINT; Schema: core; Owner: postgres; Tablespace:
--

ALTER TABLE ONLY well_test
  ADD CONSTRAINT well_test_pkey PRIMARY KEY (id);

SET search_path = filesystem, pg_catalog;

--
-- Name: file_system_dir_pkey; Type: CONSTRAINT; Schema: filesystem; Owner: postgres;
-- Tablespace:
--

ALTER TABLE ONLY directories
  ADD CONSTRAINT file_system_dir_pkey PRIMARY KEY (dir_uuid);

--
-- Name: file_system_files_pkey; Type: CONSTRAINT; Schema: filesystem; Owner: postgres;
-- Tablespace:
--

ALTER TABLE ONLY files
  ADD CONSTRAINT file_system_files_pkey PRIMARY KEY (file_uuid);

SET search_path = staging, pg_catalog;

--
-- Name: countyp020_pkey; Type: CONSTRAINT; Schema: staging; Owner: postgres;
-- Tablespace:

```

```

--

ALTER TABLE ONLY counties
  ADD CONSTRAINT countyp020_pkey PRIMARY KEY (gid);

--
-- Name: matviews_pkey; Type: CONSTRAINT; Schema: staging; Owner: postgres;
-- Tablespace:
--

ALTER TABLE ONLY matviews
  ADD CONSTRAINT matviews_pkey PRIMARY KEY (mv_name);

--
-- Name: smu2_hf_point_pkey; Type: CONSTRAINT; Schema: staging; Owner: postgres;
-- Tablespace:
--

ALTER TABLE ONLY smu2_hf_point
  ADD CONSTRAINT smu2_hf_point_pkey PRIMARY KEY (gid);

--
-- Name: states_pkey; Type: CONSTRAINT; Schema: staging; Owner: postgres; Tablespace:
--

ALTER TABLE ONLY states
  ADD CONSTRAINT states_pkey PRIMARY KEY (gid);

--
-- Name: stg_publication_pkey; Type: CONSTRAINT; Schema: staging; Owner: postgres;
-- Tablespace:
--

ALTER TABLE ONLY publication
  ADD CONSTRAINT stg_publication_pkey PRIMARY KEY (publication_id);

--
-- Name: und_td_curve_points_pkey; Type: CONSTRAINT; Schema: staging; Owner:
-- postgres; Tablespace:
--

```

```
ALTER TABLE ONLY und_td_curve_points
  ADD CONSTRAINT und_td_curve_points_pkey PRIMARY KEY (curve_point_id);
```

```
--
-- Name: und_td_sites_pkey; Type: CONSTRAINT; Schema: staging; Owner: postgres;
-- Tablespace:
```

```
ALTER TABLE ONLY und_td_sites
  ADD CONSTRAINT und_td_sites_pkey PRIMARY KEY (sites_id);
```

```
SET search_path = core, pg_catalog;
```

```
--
-- Name: bht_site_idx; Type: INDEX; Schema: core; Owner: postgres; Tablespace:
--
```

```
CREATE INDEX bht_site_idx ON bht_measurements USING btree (surface_site_id);
```

```
--
-- Name: county_id_idx; Type: INDEX; Schema: core; Owner: postgres; Tablespace:
--
```

```
CREATE INDEX county_id_idx ON ref_county USING btree (id);
```

```
--
-- Name: county_state_idx; Type: INDEX; Schema: core; Owner: postgres; Tablespace:
--
```

```
CREATE INDEX county_state_idx ON ref_county USING btree (state_id);
```

```
--
-- Name: dataset_id_idx; Type: INDEX; Schema: core; Owner: postgres; Tablespace:
--
```

```
CREATE UNIQUE INDEX dataset_id_idx ON surface_site USING btree (id);
```

```
--
-- Name: dataset_surface_county_id_idx; Type: INDEX; Schema: core; Owner: postgres;
-- Tablespace:
```

```

--

CREATE INDEX dataset_surface_county_id_idx ON surface_site USING btree (county_id);

--
-- Name: depth_site_idx; Type: INDEX; Schema: core; Owner: postgres; Tablespace:
--

CREATE INDEX depth_site_idx ON depth USING btree (surface_site_id);

--
-- Name: id_idx; Type: INDEX; Schema: core; Owner: postgres; Tablespace:
--

CREATE INDEX id_idx ON ref_state USING btree (id);

--
-- Name: precision_log_site_idx; Type: INDEX; Schema: core; Owner: postgres; Tablespace:
--

CREATE INDEX precision_log_site_idx ON precision_log USING btree (surface_site_id);

--
-- Name: ref_county_geom_gist; Type: INDEX; Schema: core; Owner: postgres; Tablespace:
--

CREATE INDEX ref_county_geom_gist ON ref_county USING gist (geometry);

--
-- Name: ref_county_shape; Type: INDEX; Schema: core; Owner: postgres; Tablespace:
--

CREATE INDEX ref_county_shape ON ref_county USING gist (geometry);

--
-- Name: ref_state_shape; Type: INDEX; Schema: core; Owner: postgres; Tablespace:
--

CREATE INDEX ref_state_shape ON ref_state USING gist (shape);

```

```

--
-- Name: shape_gix; Type: INDEX; Schema: core; Owner: postgres; Tablespace:
--

CREATE INDEX shape_gix ON surface_site USING gist (shape);


SET search_path = staging, pg_catalog;


--
-- Name: countyp020_the_geom_gist; Type: INDEX; Schema: staging; Owner: postgres;
-- Tablespace:
--

CREATE INDEX countyp020_the_geom_gist ON counties USING gist (the_geom);


--
-- Name: states_the_geom_gist; Type: INDEX; Schema: staging; Owner: postgres; Tablespace:
--

CREATE INDEX states_the_geom_gist ON states USING gist (shape);


SET search_path = core, pg_catalog;


--
-- Name: dataset_id; Type: FK CONSTRAINT; Schema: core; Owner: postgres
--

ALTER TABLE ONLY datasets_map
  ADD CONSTRAINT dataset_id FOREIGN KEY (dataset_id) REFERENCES datasets(id)
  MATCH FULL;


--
-- Name: dataset_id; Type: FK CONSTRAINT; Schema: core; Owner: postgres
--

ALTER TABLE ONLY dataset_job_data
  ADD CONSTRAINT dataset_id FOREIGN KEY (dataset_id) REFERENCES datasets(id)
  MATCH FULL;


--

```

```

-- Name: fluid_dataset; Type: FK CONSTRAINT; Schema: core; Owner: postgres
--

ALTER TABLE ONLY fluid
  ADD CONSTRAINT fluid_dataset FOREIGN KEY (dataset_id) REFERENCES datasets(id)
  MATCH FULL;

--

-- Name: fluid_h10_dataset; Type: FK CONSTRAINT; Schema: core; Owner: postgres
--

ALTER TABLE ONLY fluid_h10
  ADD CONSTRAINT fluid_h10_dataset FOREIGN KEY (dataset_id) REFERENCES
  datasets(id) MATCH FULL;

--

-- Name: fluid_h10h_dataset; Type: FK CONSTRAINT; Schema: core; Owner: postgres
--

ALTER TABLE ONLY fluid_h10h
  ADD CONSTRAINT fluid_h10h_dataset FOREIGN KEY (dataset_id) REFERENCES
  datasets(id) MATCH FULL;

--

-- Name: heat_flow_notes; Type: FK CONSTRAINT; Schema: core; Owner: postgres
--

ALTER TABLE ONLY heat_flow
  ADD CONSTRAINT heat_flow_notes FOREIGN KEY (notes_id) REFERENCES notes(id)
  MATCH FULL;

--

-- Name: heat_flow_plog; Type: FK CONSTRAINT; Schema: core; Owner: postgres
--

ALTER TABLE ONLY heat_flow
  ADD CONSTRAINT heat_flow_plog FOREIGN KEY (precision_log_id) REFERENCES
  precision_log(id) MATCH FULL;

--

-- Name: heat_flow_radioactivity; Type: FK CONSTRAINT; Schema: core; Owner: postgres

```

```

--

ALTER TABLE ONLY heat_flow
  ADD CONSTRAINT heat_flow_radioactivity FOREIGN KEY (radioactivity_id)
  REFERENCES radioactivity(id) MATCH FULL;

--

-- Name: heat_flow_thermal; Type: FK CONSTRAINT; Schema: core; Owner: postgres
--

ALTER TABLE ONLY heat_flow
  ADD CONSTRAINT heat_flow_thermal FOREIGN KEY (thermal_conductivity_id)
  REFERENCES thermal_conductivity(id) MATCH FULL;

--

-- Name: job_execution_id_fk; Type: FK CONSTRAINT; Schema: core; Owner: postgres
--

ALTER TABLE ONLY file_upload
  ADD CONSTRAINT job_execution_id_fk FOREIGN KEY (job_execution_id)
  REFERENCES gtdabatch.batch_job_execution(job_execution_id);

--

-- Name: temperature_series_plog; Type: FK CONSTRAINT; Schema: core; Owner: postgres
--

ALTER TABLE ONLY temperature_series
  ADD CONSTRAINT temperature_series_plog FOREIGN KEY (precision_log_id)
  REFERENCES precision_log(id) MATCH FULL;

--

-- Name: well_dates_dataset; Type: FK CONSTRAINT; Schema: core; Owner: postgres
--

ALTER TABLE ONLY well_dates
  ADD CONSTRAINT well_dates_dataset FOREIGN KEY (dataset_id) REFERENCES
  datasets(id) MATCH FULL;

--

-- Name: well_info_dataset; Type: FK CONSTRAINT; Schema: core; Owner: postgres
--

```

```

ALTER TABLE ONLY well_info
  ADD CONSTRAINT well_info_dataset FOREIGN KEY (dataset_id) REFERENCES
datasets(id) MATCH FULL;

--
-- Name: well_test_dataset; Type: FK CONSTRAINT; Schema: core; Owner: postgres
--

ALTER TABLE ONLY well_test
  ADD CONSTRAINT well_test_dataset FOREIGN KEY (dataset_id) REFERENCES
datasets(id) MATCH FULL;

SET search_path = filesystem, pg_catalog;

--
-- Name: file_system_dir_fkey; Type: FK CONSTRAINT; Schema: filesystem; Owner:
postgres
--

ALTER TABLE ONLY directories
  ADD CONSTRAINT file_system_dir_fkey FOREIGN KEY (parent_uuid) REFERENCES
directories(dir_uuid);

--
-- Name: file_system_files_fkey; Type: FK CONSTRAINT; Schema: filesystem; Owner:
postgres
--

ALTER TABLE ONLY files
  ADD CONSTRAINT file_system_files_fkey FOREIGN KEY (parent_uuid) REFERENCES
directories(dir_uuid) MATCH FULL;

--
-- Name: core; Type: ACL; Schema: -; Owner: gtda
--

REVOKE ALL ON SCHEMA core FROM PUBLIC;
REVOKE ALL ON SCHEMA core FROM gtda;
GRANT ALL ON SCHEMA core TO gtda;
GRANT USAGE ON SCHEMA core TO ranalytics;

```



```

--
-- Name: staging; Type: ACL; Schema: -; Owner: postgres
--

REVOKE ALL ON SCHEMA staging FROM PUBLIC;
REVOKE ALL ON SCHEMA staging FROM postgres;
GRANT ALL ON SCHEMA staging TO postgres;
GRANT USAGE ON SCHEMA staging TO ranalytics;


SET search_path = core, pg_catalog;


--
-- Name: aquifier; Type: ACL; Schema: core; Owner: postgres
--

REVOKE ALL ON TABLE aquifier FROM PUBLIC;
REVOKE ALL ON TABLE aquifier FROM postgres;
GRANT ALL ON TABLE aquifier TO postgres;
GRANT SELECT,INSERT,DELETE,UPDATE ON TABLE aquifier TO gtta;
GRANT SELECT ON TABLE aquifier TO ranalytics;


--
-- Name: bht_measurements; Type: ACL; Schema: core; Owner: postgres
--

REVOKE ALL ON TABLE bht_measurements FROM PUBLIC;
REVOKE ALL ON TABLE bht_measurements FROM postgres;
GRANT ALL ON TABLE bht_measurements TO postgres;
GRANT SELECT,INSERT,DELETE,TRUNCATE,UPDATE ON TABLE bht_measurements
TO gtta;
GRANT SELECT ON TABLE bht_measurements TO ranalytics;


--
-- Name: casing; Type: ACL; Schema: core; Owner: postgres
--

REVOKE ALL ON TABLE casing FROM PUBLIC;
REVOKE ALL ON TABLE casing FROM postgres;
GRANT ALL ON TABLE casing TO postgres;
GRANT SELECT,INSERT,DELETE,UPDATE ON TABLE casing TO gtta;
GRANT SELECT ON TABLE casing TO ranalytics;

```

```

--
-- Name: dataset_job_data; Type: ACL; Schema: core; Owner: postgres
--

REVOKE ALL ON TABLE dataset_job_data FROM PUBLIC;
REVOKE ALL ON TABLE dataset_job_data FROM postgres;
GRANT ALL ON TABLE dataset_job_data TO postgres;
GRANT SELECT,INSERT,DELETE,UPDATE ON TABLE dataset_job_data TO gtda;
GRANT SELECT ON TABLE dataset_job_data TO ranalytics;


--
-- Name: datasets; Type: ACL; Schema: core; Owner: postgres
--

REVOKE ALL ON TABLE datasets FROM PUBLIC;
REVOKE ALL ON TABLE datasets FROM postgres;
GRANT ALL ON TABLE datasets TO postgres;
GRANT SELECT ON TABLE datasets TO ranalytics;


--
-- Name: datasets_map; Type: ACL; Schema: core; Owner: postgres
--

REVOKE ALL ON TABLE datasets_map FROM PUBLIC;
REVOKE ALL ON TABLE datasets_map FROM postgres;
GRANT ALL ON TABLE datasets_map TO postgres;
GRANT SELECT,INSERT,DELETE,UPDATE ON TABLE datasets_map TO gtda;
GRANT SELECT ON TABLE datasets_map TO ranalytics;


--
-- Name: depth_id_seq; Type: ACL; Schema: core; Owner: postgres
--

REVOKE ALL ON SEQUENCE depth_id_seq FROM PUBLIC;
REVOKE ALL ON SEQUENCE depth_id_seq FROM postgres;
GRANT ALL ON SEQUENCE depth_id_seq TO postgres;
GRANT USAGE ON SEQUENCE depth_id_seq TO gtda;


--
-- Name: depth; Type: ACL; Schema: core; Owner: postgres
--

```

```
REVOKE ALL ON TABLE depth FROM PUBLIC;
REVOKE ALL ON TABLE depth FROM postgres;
GRANT ALL ON TABLE depth TO postgres;
GRANT SELECT,INSERT,DELETE,TRUNCATE,UPDATE ON TABLE depth TO gtta;
GRANT SELECT ON TABLE depth TO ranalytics;
```

```
--
-- Name: field_report_mapping_id_seq; Type: ACL; Schema: core; Owner: postgres
--
```

```
REVOKE ALL ON SEQUENCE field_report_mapping_id_seq FROM PUBLIC;
REVOKE ALL ON SEQUENCE field_report_mapping_id_seq FROM postgres;
GRANT ALL ON SEQUENCE field_report_mapping_id_seq TO postgres;
GRANT USAGE ON SEQUENCE field_report_mapping_id_seq TO gtta;
```

```
--
-- Name: file_upload; Type: ACL; Schema: core; Owner: postgres
--
```

```
REVOKE ALL ON TABLE file_upload FROM PUBLIC;
REVOKE ALL ON TABLE file_upload FROM postgres;
GRANT ALL ON TABLE file_upload TO postgres;
GRANT SELECT,INSERT,DELETE,UPDATE ON TABLE file_upload TO gtta;
GRANT SELECT ON TABLE file_upload TO ranalytics;
```

```
--
-- Name: filter_condition; Type: ACL; Schema: core; Owner: postgres
--
```

```
REVOKE ALL ON TABLE filter_condition FROM PUBLIC;
REVOKE ALL ON TABLE filter_condition FROM postgres;
GRANT ALL ON TABLE filter_condition TO postgres;
GRANT SELECT,INSERT,DELETE,TRUNCATE,UPDATE ON TABLE filter_condition TO
gtta;
GRANT SELECT ON TABLE filter_condition TO ranalytics;
```

```
--
-- Name: heat_flow; Type: ACL; Schema: core; Owner: postgres
--
```

```
REVOKE ALL ON TABLE heat_flow FROM PUBLIC;
```

```
REVOKE ALL ON TABLE heat_flow FROM postgres;
GRANT ALL ON TABLE heat_flow TO postgres;
GRANT SELECT,INSERT,DELETE,TRUNCATE,UPDATE ON TABLE heat_flow TO gtta;
GRANT SELECT ON TABLE heat_flow TO ranalytics;
```

```
--
-- Name: heat_flow_id_seq; Type: ACL; Schema: core; Owner: postgres
--
```

```
REVOKE ALL ON SEQUENCE heat_flow_id_seq FROM PUBLIC;
REVOKE ALL ON SEQUENCE heat_flow_id_seq FROM postgres;
GRANT ALL ON SEQUENCE heat_flow_id_seq TO postgres;
GRANT USAGE ON SEQUENCE heat_flow_id_seq TO gtta;
```

```
--
-- Name: heat_flow_publication_links_id_seq; Type: ACL; Schema: core; Owner: postgres
--
```

```
REVOKE ALL ON SEQUENCE heat_flow_publication_links_id_seq FROM PUBLIC;
REVOKE ALL ON SEQUENCE heat_flow_publication_links_id_seq FROM postgres;
GRANT ALL ON SEQUENCE heat_flow_publication_links_id_seq TO postgres;
GRANT USAGE ON SEQUENCE heat_flow_publication_links_id_seq TO gtta;
```

```
--
-- Name: heat_generation_id_seq; Type: ACL; Schema: core; Owner: postgres
--
```

```
REVOKE ALL ON SEQUENCE heat_generation_id_seq FROM PUBLIC;
REVOKE ALL ON SEQUENCE heat_generation_id_seq FROM postgres;
GRANT ALL ON SEQUENCE heat_generation_id_seq TO postgres;
GRANT USAGE ON SEQUENCE heat_generation_id_seq TO gtta;
```

```
--
-- Name: interval; Type: ACL; Schema: core; Owner: postgres
--
```

```
REVOKE ALL ON TABLE "interval" FROM PUBLIC;
REVOKE ALL ON TABLE "interval" FROM postgres;
GRANT ALL ON TABLE "interval" TO postgres;
GRANT SELECT,INSERT,DELETE,UPDATE ON TABLE "interval" TO gtta;
GRANT SELECT ON TABLE "interval" TO ranalytics;
```

```

--
-- Name: lithology_information_id_seq; Type: ACL; Schema: core; Owner: postgres
--

REVOKE ALL ON SEQUENCE lithology_information_id_seq FROM PUBLIC;
REVOKE ALL ON SEQUENCE lithology_information_id_seq FROM postgres;
GRANT ALL ON SEQUENCE lithology_information_id_seq TO postgres;
GRANT USAGE ON SEQUENCE lithology_information_id_seq TO gtda;

--
-- Name: ngds_heat_flow_mview; Type: ACL; Schema: core; Owner: postgres
--

REVOKE ALL ON TABLE ngds_heat_flow_mview FROM PUBLIC;
REVOKE ALL ON TABLE ngds_heat_flow_mview FROM postgres;
GRANT ALL ON TABLE ngds_heat_flow_mview TO postgres;
GRANT SELECT,INSERT,DELETE,TRUNCATE,UPDATE ON TABLE
ngds_heat_flow_mview TO gtda;
GRANT SELECT ON TABLE ngds_heat_flow_mview TO ranalytics;

--
-- Name: notes; Type: ACL; Schema: core; Owner: postgres
--

REVOKE ALL ON TABLE notes FROM PUBLIC;
REVOKE ALL ON TABLE notes FROM postgres;
GRANT ALL ON TABLE notes TO postgres;
GRANT SELECT,INSERT,DELETE,TRUNCATE,UPDATE ON TABLE notes TO gtda;
GRANT SELECT ON TABLE notes TO ranalytics;

--
-- Name: precision_log; Type: ACL; Schema: core; Owner: postgres
--

REVOKE ALL ON TABLE precision_log FROM PUBLIC;
REVOKE ALL ON TABLE precision_log FROM postgres;
GRANT ALL ON TABLE precision_log TO postgres;
GRANT SELECT,INSERT,DELETE,TRUNCATE,UPDATE ON TABLE precision_log TO
gtda;
GRANT SELECT ON TABLE precision_log TO ranalytics;

```

```

--
-- Name: publication; Type: ACL; Schema: core; Owner: postgres
--

REVOKE ALL ON TABLE publication FROM PUBLIC;
REVOKE ALL ON TABLE publication FROM postgres;
GRANT ALL ON TABLE publication TO postgres;
GRANT SELECT,INSERT,DELETE,TRUNCATE,UPDATE ON TABLE publication TO
gt-da;
GRANT SELECT ON TABLE publication TO ranalytics;

SET search_path = filesystem, pg_catalog;

--
-- Name: files; Type: ACL; Schema: filesystem; Owner: postgres
--

REVOKE ALL ON TABLE files FROM PUBLIC;
REVOKE ALL ON TABLE files FROM postgres;
GRANT ALL ON TABLE files TO postgres;
GRANT SELECT,INSERT,DELETE,TRUNCATE,UPDATE ON TABLE files TO gt-da;

SET search_path = core, pg_catalog;

--
-- Name: publication_publication_id_seq; Type: ACL; Schema: core; Owner: postgres
--

REVOKE ALL ON SEQUENCE publication_publication_id_seq FROM PUBLIC;
REVOKE ALL ON SEQUENCE publication_publication_id_seq FROM postgres;
GRANT ALL ON SEQUENCE publication_publication_id_seq TO postgres;
GRANT USAGE ON SEQUENCE publication_publication_id_seq TO gt-da;

--
-- Name: quality_issue_log_id_seq; Type: ACL; Schema: core; Owner: postgres
--

REVOKE ALL ON SEQUENCE quality_issue_log_id_seq FROM PUBLIC;
REVOKE ALL ON SEQUENCE quality_issue_log_id_seq FROM postgres;
GRANT ALL ON SEQUENCE quality_issue_log_id_seq TO postgres;
GRANT USAGE ON SEQUENCE quality_issue_log_id_seq TO gt-da;
GRANT USAGE ON SEQUENCE quality_issue_log_id_seq TO ranalytics;

```

```

--
-- Name: radioactivity; Type: ACL; Schema: core; Owner: postgres
--

REVOKE ALL ON TABLE radioactivity FROM PUBLIC;
REVOKE ALL ON TABLE radioactivity FROM postgres;
GRANT ALL ON TABLE radioactivity TO postgres;
GRANT SELECT,INSERT,DELETE,UPDATE ON TABLE radioactivity TO gtda;
GRANT SELECT ON TABLE radioactivity TO ranalytics;


--
-- Name: ref_county; Type: ACL; Schema: core; Owner: postgres
--

REVOKE ALL ON TABLE ref_county FROM PUBLIC;
REVOKE ALL ON TABLE ref_county FROM postgres;
GRANT ALL ON TABLE ref_county TO postgres;
GRANT SELECT,INSERT,DELETE,TRUNCATE,UPDATE ON TABLE ref_county TO
gtda;
GRANT SELECT ON TABLE ref_county TO ranalytics;


--
-- Name: ref_county_elev; Type: ACL; Schema: core; Owner: postgres
--

REVOKE ALL ON TABLE ref_county_elev FROM PUBLIC;
REVOKE ALL ON TABLE ref_county_elev FROM postgres;
GRANT ALL ON TABLE ref_county_elev TO postgres;
GRANT SELECT,INSERT,DELETE,TRUNCATE,UPDATE ON TABLE ref_county_elev TO
gtda;
GRANT SELECT ON TABLE ref_county_elev TO ranalytics;


--
-- Name: ref_data_source; Type: ACL; Schema: core; Owner: postgres
--

REVOKE ALL ON TABLE ref_data_source FROM PUBLIC;
REVOKE ALL ON TABLE ref_data_source FROM postgres;
GRANT ALL ON TABLE ref_data_source TO postgres;
GRANT SELECT,INSERT,DELETE,TRUNCATE,UPDATE ON TABLE ref_data_source TO
gtda;
GRANT SELECT ON TABLE ref_data_source TO ranalytics;

```

```
--  
-- Name: ref_quality_issue_type; Type: ACL; Schema: core; Owner: postgres  
--
```

```
REVOKE ALL ON TABLE ref_quality_issue_type FROM PUBLIC;  
REVOKE ALL ON TABLE ref_quality_issue_type FROM postgres;  
GRANT ALL ON TABLE ref_quality_issue_type TO postgres;  
GRANT SELECT,INSERT,DELETE,TRUNCATE,UPDATE ON TABLE  
ref_quality_issue_type TO gtda;  
GRANT SELECT ON TABLE ref_quality_issue_type TO ranalytics;
```

```
--  
-- Name: ref_quality_issue_type_h; Type: ACL; Schema: core; Owner: postgres  
--
```

```
REVOKE ALL ON TABLE ref_quality_issue_type_h FROM PUBLIC;  
REVOKE ALL ON TABLE ref_quality_issue_type_h FROM postgres;  
GRANT ALL ON TABLE ref_quality_issue_type_h TO postgres;  
GRANT SELECT,INSERT,DELETE,TRUNCATE,UPDATE ON TABLE  
ref_quality_issue_type_h TO gtda;  
GRANT SELECT ON TABLE ref_quality_issue_type_h TO ranalytics;
```

```
--  
-- Name: ref_state; Type: ACL; Schema: core; Owner: postgres  
--
```

```
REVOKE ALL ON TABLE ref_state FROM PUBLIC;  
REVOKE ALL ON TABLE ref_state FROM postgres;  
GRANT ALL ON TABLE ref_state TO postgres;  
GRANT SELECT,INSERT,DELETE,TRUNCATE,UPDATE ON TABLE ref_state TO gtda;  
GRANT SELECT ON TABLE ref_state TO ranalytics;
```

```
--  
-- Name: ref_symbol; Type: ACL; Schema: core; Owner: postgres  
--
```

```
REVOKE ALL ON TABLE ref_symbol FROM PUBLIC;  
REVOKE ALL ON TABLE ref_symbol FROM postgres;  
GRANT ALL ON TABLE ref_symbol TO postgres;  
GRANT SELECT,INSERT,DELETE,TRUNCATE,UPDATE ON TABLE ref_symbol TO  
gtda;
```



```
GRANT SELECT ON TABLE ref_symbol TO ranalytics;
```

```
--  
-- Name: site_comments_id_seq; Type: ACL; Schema: core; Owner: postgres  
--
```

```
REVOKE ALL ON SEQUENCE site_comments_id_seq FROM PUBLIC;  
REVOKE ALL ON SEQUENCE site_comments_id_seq FROM postgres;  
GRANT ALL ON SEQUENCE site_comments_id_seq TO postgres;  
GRANT USAGE ON SEQUENCE site_comments_id_seq TO gtda;
```

```
--  
-- Name: site_publication_links_id_seq; Type: ACL; Schema: core; Owner: postgres  
--
```

```
REVOKE ALL ON SEQUENCE site_publication_links_id_seq FROM PUBLIC;  
REVOKE ALL ON SEQUENCE site_publication_links_id_seq FROM postgres;  
GRANT ALL ON SEQUENCE site_publication_links_id_seq TO postgres;  
GRANT USAGE ON SEQUENCE site_publication_links_id_seq TO gtda;
```

```
--  
-- Name: site_related_resources_id_seq; Type: ACL; Schema: core; Owner: postgres  
--
```

```
REVOKE ALL ON SEQUENCE site_related_resources_id_seq FROM PUBLIC;  
REVOKE ALL ON SEQUENCE site_related_resources_id_seq FROM postgres;  
GRANT ALL ON SEQUENCE site_related_resources_id_seq TO postgres;  
GRANT USAGE ON SEQUENCE site_related_resources_id_seq TO gtda;
```

```
--  
-- Name: surface_site; Type: ACL; Schema: core; Owner: postgres  
--
```

```
REVOKE ALL ON TABLE surface_site FROM PUBLIC;  
REVOKE ALL ON TABLE surface_site FROM postgres;  
GRANT ALL ON TABLE surface_site TO postgres;  
GRANT SELECT,INSERT,DELETE,TRUNCATE,UPDATE ON TABLE surface_site TO  
gtda;  
GRANT SELECT ON TABLE surface_site TO ranalytics;
```

```
--
```

-- Name: surface_site_county_state_materialized_view; Type: ACL; Schema: core; Owner: postgres

--

REVOKE ALL ON TABLE surface_site_county_state_materialized_view FROM PUBLIC;
REVOKE ALL ON TABLE surface_site_county_state_materialized_view FROM postgres;
GRANT SELECT ON TABLE surface_site_county_state_materialized_view TO postgres;

--

-- Name: thermal_conductivity; Type: ACL; Schema: core; Owner: postgres

--

REVOKE ALL ON TABLE thermal_conductivity FROM PUBLIC;
REVOKE ALL ON TABLE thermal_conductivity FROM postgres;
GRANT ALL ON TABLE thermal_conductivity TO postgres;
GRANT SELECT,INSERT,DELETE,TRUNCATE,UPDATE ON TABLE thermal_conductivity TO gtda;
GRANT SELECT ON TABLE thermal_conductivity TO ranalytics;

--

-- Name: surface_site_county_state_view; Type: ACL; Schema: core; Owner: postgres

--

REVOKE ALL ON TABLE surface_site_county_state_view FROM PUBLIC;
REVOKE ALL ON TABLE surface_site_county_state_view FROM postgres;
GRANT SELECT ON TABLE surface_site_county_state_view TO postgres;

--

-- Name: temperature_series_id_seq; Type: ACL; Schema: core; Owner: postgres

--

REVOKE ALL ON SEQUENCE temperature_series_id_seq FROM PUBLIC;
REVOKE ALL ON SEQUENCE temperature_series_id_seq FROM postgres;
GRANT ALL ON SEQUENCE temperature_series_id_seq TO postgres;
GRANT USAGE ON SEQUENCE temperature_series_id_seq TO gtda;

--

-- Name: temperature_series; Type: ACL; Schema: core; Owner: postgres

--

REVOKE ALL ON TABLE temperature_series FROM PUBLIC;
REVOKE ALL ON TABLE temperature_series FROM postgres;

```
GRANT ALL ON TABLE temperature_series TO postgres;
GRANT SELECT,INSERT,DELETE,TRUNCATE,UPDATE ON TABLE temperature_series
TO gtda;
GRANT SELECT ON TABLE temperature_series TO ranalytics;
```

```
--
-- Name: template_borehole; Type: ACL; Schema: core; Owner: postgres
--
```

```
REVOKE ALL ON TABLE template_borehole FROM PUBLIC;
REVOKE ALL ON TABLE template_borehole FROM postgres;
GRANT SELECT ON TABLE template_borehole TO postgres;
```

```
--
-- Name: template_borehole_materialized; Type: ACL; Schema: core; Owner: postgres
--
```

```
REVOKE ALL ON TABLE template_borehole_materialized FROM PUBLIC;
REVOKE ALL ON TABLE template_borehole_materialized FROM postgres;
GRANT SELECT ON TABLE template_borehole_materialized TO postgres;
```

```
--
-- Name: template_fluidflux; Type: ACL; Schema: core; Owner: postgres
--
```

```
REVOKE ALL ON TABLE template_fluidflux FROM PUBLIC;
REVOKE ALL ON TABLE template_fluidflux FROM postgres;
GRANT SELECT ON TABLE template_fluidflux TO postgres;
```

```
--
-- Name: template_fluidflux_materialized; Type: ACL; Schema: core; Owner: postgres
--
```

```
REVOKE ALL ON TABLE template_fluidflux_materialized FROM PUBLIC;
REVOKE ALL ON TABLE template_fluidflux_materialized FROM postgres;
GRANT SELECT ON TABLE template_fluidflux_materialized TO postgres;
```

```
--
-- Name: template_heatflow; Type: ACL; Schema: core; Owner: postgres
--
```

```
REVOKE ALL ON TABLE template_heatflow FROM PUBLIC;
REVOKE ALL ON TABLE template_heatflow FROM postgres;
GRANT SELECT ON TABLE template_heatflow TO postgres;
```

```
--
-- Name: template_heatflow_materialized; Type: ACL; Schema: core; Owner: postgres
--
```

```
REVOKE ALL ON TABLE template_heatflow_materialized FROM PUBLIC;
REVOKE ALL ON TABLE template_heatflow_materialized FROM postgres;
GRANT SELECT ON TABLE template_heatflow_materialized TO postgres;
```

```
--
-- Name: template_radiogenicheat; Type: ACL; Schema: core; Owner: postgres
--
```

```
REVOKE ALL ON TABLE template_radiogenicheat FROM PUBLIC;
REVOKE ALL ON TABLE template_radiogenicheat FROM postgres;
GRANT SELECT ON TABLE template_radiogenicheat TO postgres;
```

```
--
-- Name: template_radiogenicheat_materialized; Type: ACL; Schema: core; Owner: postgres
--
```

```
REVOKE ALL ON TABLE template_radiogenicheat_materialized FROM PUBLIC;
REVOKE ALL ON TABLE template_radiogenicheat_materialized FROM postgres;
GRANT SELECT ON TABLE template_radiogenicheat_materialized TO postgres;
```

```
--
-- Name: template_reservoir; Type: ACL; Schema: core; Owner: postgres
--
```

```
REVOKE ALL ON TABLE template_reservoir FROM PUBLIC;
REVOKE ALL ON TABLE template_reservoir FROM postgres;
GRANT SELECT ON TABLE template_reservoir TO postgres;
```

```
--
-- Name: template_reservoir_materialized; Type: ACL; Schema: core; Owner: postgres
--
```

```
REVOKE ALL ON TABLE template_reservoir_materialized FROM PUBLIC;
```

```
REVOKE ALL ON TABLE template_reservoir_materialized FROM postgres;  
GRANT SELECT ON TABLE template_reservoir_materialized TO postgres;
```

```
--  
-- Name: template_thermalconductivity; Type: ACL; Schema: core; Owner: postgres  
--
```

```
REVOKE ALL ON TABLE template_thermalconductivity FROM PUBLIC;  
REVOKE ALL ON TABLE template_thermalconductivity FROM postgres;  
GRANT SELECT ON TABLE template_thermalconductivity TO postgres;
```

```
--  
-- Name: template_thermalconductivity_materialized; Type: ACL; Schema: core; Owner:  
postgres  
--
```

```
REVOKE ALL ON TABLE template_thermalconductivity_materialized FROM PUBLIC;  
REVOKE ALL ON TABLE template_thermalconductivity_materialized FROM postgres;  
GRANT SELECT ON TABLE template_thermalconductivity_materialized TO postgres;
```

```
--  
-- Name: template_wellfluid; Type: ACL; Schema: core; Owner: postgres  
--
```

```
REVOKE ALL ON TABLE template_wellfluid FROM PUBLIC;  
REVOKE ALL ON TABLE template_wellfluid FROM postgres;  
GRANT SELECT ON TABLE template_wellfluid TO postgres;
```

```
--  
-- Name: template_wellfluid_materialized; Type: ACL; Schema: core; Owner: postgres  
--
```

```
REVOKE ALL ON TABLE template_wellfluid_materialized FROM PUBLIC;  
REVOKE ALL ON TABLE template_wellfluid_materialized FROM postgres;  
GRANT SELECT ON TABLE template_wellfluid_materialized TO postgres;
```

```
--  
-- Name: template_wellheader; Type: ACL; Schema: core; Owner: postgres  
--
```

```
REVOKE ALL ON TABLE template_wellheader FROM PUBLIC;
```

```
REVOKE ALL ON TABLE template_wellheader FROM postgres;  
GRANT SELECT ON TABLE template_wellheader TO postgres;
```

```
--  
-- Name: template_wellheader_materialized; Type: ACL; Schema: core; Owner: postgres  
--
```

```
REVOKE ALL ON TABLE template_wellheader_materialized FROM PUBLIC;  
REVOKE ALL ON TABLE template_wellheader_materialized FROM postgres;  
GRANT SELECT ON TABLE template_wellheader_materialized TO postgres;
```

```
--  
-- Name: template_welllog; Type: ACL; Schema: core; Owner: postgres  
--
```

```
REVOKE ALL ON TABLE template_welllog FROM PUBLIC;  
REVOKE ALL ON TABLE template_welllog FROM postgres;  
GRANT SELECT ON TABLE template_welllog TO postgres;
```

```
--  
-- Name: template_welllog_materialized; Type: ACL; Schema: core; Owner: postgres  
--
```

```
REVOKE ALL ON TABLE template_welllog_materialized FROM PUBLIC;  
REVOKE ALL ON TABLE template_welllog_materialized FROM postgres;  
GRANT SELECT ON TABLE template_welllog_materialized TO postgres;
```

```
--  
-- Name: template_welltest; Type: ACL; Schema: core; Owner: postgres  
--
```

```
REVOKE ALL ON TABLE template_welltest FROM PUBLIC;  
REVOKE ALL ON TABLE template_welltest FROM postgres;  
GRANT SELECT ON TABLE template_welltest TO postgres;
```

```
--  
-- Name: template_welltest_materialized; Type: ACL; Schema: core; Owner: postgres  
--
```

```
REVOKE ALL ON TABLE template_welltest_materialized FROM PUBLIC;  
REVOKE ALL ON TABLE template_welltest_materialized FROM postgres;
```

```

GRANT SELECT ON TABLE template_welltest_materialized TO postgres;

--
-- Name: temprature_series_mapping_id_seq; Type: ACL; Schema: core; Owner: postgres
--

REVOKE ALL ON SEQUENCE temprature_series_mapping_id_seq FROM PUBLIC;
REVOKE ALL ON SEQUENCE temprature_series_mapping_id_seq FROM postgres;
GRANT ALL ON SEQUENCE temprature_series_mapping_id_seq TO postgres;
GRANT USAGE ON SEQUENCE temprature_series_mapping_id_seq TO gtda;

SET search_path = filesystem, pg_catalog;

--
-- Name: directories; Type: ACL; Schema: filesystem; Owner: postgres
--

REVOKE ALL ON TABLE directories FROM PUBLIC;
REVOKE ALL ON TABLE directories FROM postgres;
GRANT ALL ON TABLE directories TO postgres;
GRANT SELECT,INSERT,DELETE,TRUNCATE,UPDATE ON TABLE directories TO gtda;

SET search_path = staging, pg_catalog;

--
-- Name: countyp020_gid_seq; Type: ACL; Schema: staging; Owner: postgres
--

REVOKE ALL ON SEQUENCE countyp020_gid_seq FROM PUBLIC;
REVOKE ALL ON SEQUENCE countyp020_gid_seq FROM postgres;
GRANT ALL ON SEQUENCE countyp020_gid_seq TO postgres;
GRANT USAGE ON SEQUENCE countyp020_gid_seq TO gtda;

--
-- Name: counties; Type: ACL; Schema: staging; Owner: postgres
--

REVOKE ALL ON TABLE counties FROM PUBLIC;
REVOKE ALL ON TABLE counties FROM postgres;
GRANT ALL ON TABLE counties TO postgres;
GRANT SELECT,INSERT,DELETE,TRUNCATE,UPDATE ON TABLE counties TO gtda;
GRANT SELECT ON TABLE counties TO ranalytics;

```

```

--
-- Name: states; Type: ACL; Schema: staging; Owner: postgres
--

REVOKE ALL ON TABLE states FROM PUBLIC;
REVOKE ALL ON TABLE states FROM postgres;
GRANT ALL ON TABLE states TO postgres;
GRANT SELECT,INSERT,DELETE,TRUNCATE,UPDATE ON TABLE states TO gtda;
GRANT SELECT ON TABLE states TO ranalytics;


--
-- Name: beg_reservoir_view; Type: ACL; Schema: staging; Owner: postgres
--

REVOKE ALL ON TABLE beg_reservoir_view FROM PUBLIC;
REVOKE ALL ON TABLE beg_reservoir_view FROM postgres;
GRANT SELECT ON TABLE beg_reservoir_view TO postgres;


--
-- Name: beg_reservoir_view_materialized; Type: ACL; Schema: staging; Owner: postgres
--

REVOKE ALL ON TABLE beg_reservoir_view_materialized FROM PUBLIC;
REVOKE ALL ON TABLE beg_reservoir_view_materialized FROM postgres;
GRANT SELECT ON TABLE beg_reservoir_view_materialized TO postgres;


--
-- Name: beg_well_view; Type: ACL; Schema: staging; Owner: postgres
--

REVOKE ALL ON TABLE beg_well_view FROM PUBLIC;
REVOKE ALL ON TABLE beg_well_view FROM postgres;
GRANT SELECT ON TABLE beg_well_view TO postgres;


--
-- Name: beg_well_view_materialized; Type: ACL; Schema: staging; Owner: postgres
--

REVOKE ALL ON TABLE beg_well_view_materialized FROM PUBLIC;
REVOKE ALL ON TABLE beg_well_view_materialized FROM postgres;

```



```

GRANT SELECT ON TABLE beg_well_view_materialized TO postgres;

--
-- Name: cornell_well_view; Type: ACL; Schema: staging; Owner: postgres
--

REVOKE ALL ON TABLE cornell_well_view FROM PUBLIC;
REVOKE ALL ON TABLE cornell_well_view FROM postgres;
GRANT SELECT ON TABLE cornell_well_view TO postgres;

--
-- Name: cornell_well_view_materialized; Type: ACL; Schema: staging; Owner: postgres
--

REVOKE ALL ON TABLE cornell_well_view_materialized FROM PUBLIC;
REVOKE ALL ON TABLE cornell_well_view_materialized FROM postgres;
GRANT SELECT ON TABLE cornell_well_view_materialized TO postgres;

--
-- Name: field_reports_id_seq; Type: ACL; Schema: staging; Owner: postgres
--

REVOKE ALL ON SEQUENCE field_reports_id_seq FROM PUBLIC;
REVOKE ALL ON SEQUENCE field_reports_id_seq FROM postgres;
GRANT ALL ON SEQUENCE field_reports_id_seq TO postgres;
GRANT USAGE ON SEQUENCE field_reports_id_seq TO gtda;

--
-- Name: field_reports_mapping_id_seq; Type: ACL; Schema: staging; Owner: postgres
--

REVOKE ALL ON SEQUENCE field_reports_mapping_id_seq FROM PUBLIC;
REVOKE ALL ON SEQUENCE field_reports_mapping_id_seq FROM postgres;
GRANT ALL ON SEQUENCE field_reports_mapping_id_seq TO postgres;
GRANT USAGE ON SEQUENCE field_reports_mapping_id_seq TO gtda;

--
-- Name: publication; Type: ACL; Schema: staging; Owner: postgres
--

REVOKE ALL ON TABLE publication FROM PUBLIC;

```

```

REVOKE ALL ON TABLE publication FROM postgres;
GRANT ALL ON TABLE publication TO postgres;
GRANT SELECT,INSERT,DELETE,UPDATE ON TABLE publication TO gtda;
GRANT SELECT ON TABLE publication TO ranalytics;

--
-- Name: smu2_hf_heatflow; Type: ACL; Schema: staging; Owner: postgres
--

REVOKE ALL ON TABLE smu2_hf_heatflow FROM PUBLIC;
REVOKE ALL ON TABLE smu2_hf_heatflow FROM postgres;
GRANT ALL ON TABLE smu2_hf_heatflow TO postgres;
GRANT SELECT,INSERT,DELETE,UPDATE ON TABLE smu2_hf_heatflow TO gtda;
GRANT SELECT ON TABLE smu2_hf_heatflow TO ranalytics;

--
-- Name: smu2_hf_point; Type: ACL; Schema: staging; Owner: postgres
--

REVOKE ALL ON TABLE smu2_hf_point FROM PUBLIC;
REVOKE ALL ON TABLE smu2_hf_point FROM postgres;
GRANT ALL ON TABLE smu2_hf_point TO postgres;
GRANT SELECT,INSERT,DELETE,UPDATE ON TABLE smu2_hf_point TO gtda;
GRANT SELECT ON TABLE smu2_hf_point TO ranalytics;

--
-- Name: smu2_hf_ref; Type: ACL; Schema: staging; Owner: postgres
--

REVOKE ALL ON TABLE smu2_hf_ref FROM PUBLIC;
REVOKE ALL ON TABLE smu2_hf_ref FROM postgres;
GRANT ALL ON TABLE smu2_hf_ref TO postgres;
GRANT SELECT,INSERT,DELETE,UPDATE ON TABLE smu2_hf_ref TO gtda;
GRANT SELECT ON TABLE smu2_hf_ref TO ranalytics;

--
-- Name: smu_bht_view; Type: ACL; Schema: staging; Owner: postgres
--

REVOKE ALL ON TABLE smu_bht_view FROM PUBLIC;
REVOKE ALL ON TABLE smu_bht_view FROM postgres;
GRANT SELECT ON TABLE smu_bht_view TO postgres;

```

```
--  
-- Name: smu_bht_view_materialized; Type: ACL; Schema: staging; Owner: postgres  
--
```

```
REVOKE ALL ON TABLE smu_bht_view_materialized FROM PUBLIC;  
REVOKE ALL ON TABLE smu_bht_view_materialized FROM postgres;  
GRANT SELECT ON TABLE smu_bht_view_materialized TO postgres;
```

```
--  
-- Name: smu_hf_view; Type: ACL; Schema: staging; Owner: postgres  
--
```

```
REVOKE ALL ON TABLE smu_hf_view FROM PUBLIC;  
REVOKE ALL ON TABLE smu_hf_view FROM postgres;  
GRANT ALL ON TABLE smu_hf_view TO postgres;
```

```
--  
-- Name: smu_hf_view_materialized; Type: ACL; Schema: staging; Owner: postgres  
--
```

```
REVOKE ALL ON TABLE smu_hf_view_materialized FROM PUBLIC;  
REVOKE ALL ON TABLE smu_hf_view_materialized FROM postgres;  
GRANT SELECT ON TABLE smu_hf_view_materialized TO postgres;
```

```
--  
-- Name: states_gid_seq; Type: ACL; Schema: staging; Owner: postgres  
--
```

```
REVOKE ALL ON SEQUENCE states_gid_seq FROM PUBLIC;  
REVOKE ALL ON SEQUENCE states_gid_seq FROM postgres;  
GRANT ALL ON SEQUENCE states_gid_seq TO postgres;  
GRANT USAGE ON SEQUENCE states_gid_seq TO gtda;
```

```
--  
-- Name: tt_well_view; Type: ACL; Schema: staging; Owner: postgres  
--
```

```
REVOKE ALL ON TABLE tt_well_view FROM PUBLIC;  
REVOKE ALL ON TABLE tt_well_view FROM postgres;  
GRANT SELECT ON TABLE tt_well_view TO postgres;
```

```

--
-- Name: tt_well_view_materialized; Type: ACL; Schema: staging; Owner: postgres
--

REVOKE ALL ON TABLE tt_well_view_materialized FROM PUBLIC;
REVOKE ALL ON TABLE tt_well_view_materialized FROM postgres;
GRANT SELECT ON TABLE tt_well_view_materialized TO postgres;


--
-- Name: und_rad_view; Type: ACL; Schema: staging; Owner: postgres
--

REVOKE ALL ON TABLE und_rad_view FROM PUBLIC;
REVOKE ALL ON TABLE und_rad_view FROM postgres;
GRANT SELECT ON TABLE und_rad_view TO postgres;


--
-- Name: und_rad_view_materialized; Type: ACL; Schema: staging; Owner: postgres
--

REVOKE ALL ON TABLE und_rad_view_materialized FROM PUBLIC;
REVOKE ALL ON TABLE und_rad_view_materialized FROM postgres;
GRANT SELECT ON TABLE und_rad_view_materialized TO postgres;


--
-- Name: und_tc_view; Type: ACL; Schema: staging; Owner: postgres
--

REVOKE ALL ON TABLE und_tc_view FROM PUBLIC;
REVOKE ALL ON TABLE und_tc_view FROM postgres;
GRANT SELECT ON TABLE und_tc_view TO postgres;


--
-- Name: und_tc_view_materialized; Type: ACL; Schema: staging; Owner: postgres
--

REVOKE ALL ON TABLE und_tc_view_materialized FROM PUBLIC;
REVOKE ALL ON TABLE und_tc_view_materialized FROM postgres;
GRANT SELECT ON TABLE und_tc_view_materialized TO postgres;

```

```

--
-- Name: und_td_curve_points; Type: ACL; Schema: staging; Owner: postgres
--

REVOKE ALL ON TABLE und_td_curve_points FROM PUBLIC;
REVOKE ALL ON TABLE und_td_curve_points FROM postgres;
GRANT ALL ON TABLE und_td_curve_points TO postgres;
GRANT SELECT,INSERT,DELETE,UPDATE ON TABLE und_td_curve_points TO gtda;
GRANT SELECT ON TABLE und_td_curve_points TO ranalytics;


--
-- Name: und_td_sites; Type: ACL; Schema: staging; Owner: postgres
--

REVOKE ALL ON TABLE und_td_sites FROM PUBLIC;
REVOKE ALL ON TABLE und_td_sites FROM postgres;
GRANT ALL ON TABLE und_td_sites TO postgres;
GRANT SELECT,INSERT,DELETE,UPDATE ON TABLE und_td_sites TO gtda;
GRANT SELECT ON TABLE und_td_sites TO ranalytics;


--
-- Name: und_td_view; Type: ACL; Schema: staging; Owner: postgres
--

REVOKE ALL ON TABLE und_td_view FROM PUBLIC;
REVOKE ALL ON TABLE und_td_view FROM postgres;
GRANT SELECT ON TABLE und_td_view TO postgres;


--
-- Name: und_td_view_materialized; Type: ACL; Schema: staging; Owner: postgres
--

REVOKE ALL ON TABLE und_td_view_materialized FROM PUBLIC;
REVOKE ALL ON TABLE und_td_view_materialized FROM postgres;
GRANT SELECT ON TABLE und_td_view_materialized TO postgres;


--
-- Name: well_log_headers_id_seq; Type: ACL; Schema: staging; Owner: postgres
--

REVOKE ALL ON SEQUENCE well_log_headers_id_seq FROM PUBLIC;

```

```
REVOKE ALL ON SEQUENCE well_log_headers_id_seq FROM postgres;
GRANT ALL ON SEQUENCE well_log_headers_id_seq TO postgres;
GRANT USAGE ON SEQUENCE well_log_headers_id_seq TO gtda;

--
-- Name: well_log_headers_mapping_id_seq; Type: ACL; Schema: staging; Owner: postgres
--

REVOKE ALL ON SEQUENCE well_log_headers_mapping_id_seq FROM PUBLIC;
REVOKE ALL ON SEQUENCE well_log_headers_mapping_id_seq FROM postgres;
GRANT ALL ON SEQUENCE well_log_headers_mapping_id_seq TO postgres;
GRANT USAGE ON SEQUENCE well_log_headers_mapping_id_seq TO gtda;

--
-- PostgreSQL database dump complete
--
```

Tables and Fields in the Staging Database

Pulled on April 7, 2014 using the following command:

```
psql -h sgtl1d1 -U postgres gtda -c '\d+ staging.*' > gt-da-staging-tables.txt 2> /dev/null
```

Required signing onto sgtl1p1 with PuTTY. Also required the password for username 'postgres' on sgtl1d1.

Staging Tables and Fields Contents

Tables and Fields in the Staging Database.....	217
BEG Reservoir.....	219
Table "staging.beg_reservoir_data"	219
View "staging.beg_reservoir_view"	221
Table "staging.beg_reservoir_view_materialized"	223
BEG Well Data	224
Table "staging.beg_well_data"	224
View "staging.beg_well_view"	226
Table "staging.beg_well_view_materialized"	228
Cornell Well Data	230
Table "staging.cornell_well_data"	230
View "staging.cornell_well_view"	231
Table "staging.cornell_well_view_materialized"	232
Counties	233
Table "staging.counties"	233
Sequence "staging.countyp020_gid_seq"	234
Index "staging.countyp020_pkey"	234
Index "staging.countyp020_the_geom_gist"	234
Field Reports ID	234
Sequence "staging.field_reports_id_seq"	234
Sequence "staging.field_reports_mapping_id_seq"	234
Mat Views	235
Table "staging.matviews"	235
Index "staging.matviews_pkey"	235

MLKay BHT Data.....	235
Table "staging.mlk_bht_data"	235
Publications.....	235
Table "staging.publication".....	235
Sequence "staging.publication_publication_id_seq"	236
SMU Heat Flow (EQ Data – Part 1 of 3)	236
Table "staging.smu2_hf_heatflow"	236
SMU Heat Flow (EQ Data – Part 2 of 3)	237
Table "staging.smu2_hf_point"	237
Index "staging.smu2_hf_point_pkey"	238
SMU Heat Flow (EQ Data – Part 3 of 3)	238
Table "staging.smu2_hf_ref"	238
SMU BHT Heat Flow	238
Table "staging.smu_bht_data"	238
View "staging.smu_bht_view"	239
Table "staging.smu_bht_view_materialized"	240
SMU Heat Flow (EQ Data – All 3 of 3).....	241
View "staging.smu_hf_view"	241
Table "staging.smu_hf_view_materialized"	242
States	244
Table "staging.states"	244
Sequence "staging.states_gid_seq"	244
Index "staging.states_pkey"	244
Index "staging.states_the_geom_gist"	244
Publications.....	244
Index "staging.stg_publication_pkey"	244
TTU Well Data (Part 1 of 3)	245
Table "staging.tt_corrected_bht"	245
TTU Well Data (Part 2 of 3)	245
Table "staging.tt_uncorrected_bht"	245
TTU Well Data (Part 3 of 3)	246
Table "staging.tt_well_data"	246

TTU Well Data (All 3 of 3).....	247
View "staging.tt_well_view"	247
Table "staging.tt_well_view_materialized"	248
UND Radiogenic Heat Data	250
Table "staging.und_rad_data"	250
View "staging.und_rad_view"	251
Table "staging.und_rad_view_materialized"	252
UND Thermal Conductivity Data	253
Table "staging.und_tc_data"	253
View "staging.und_tc_view"	255
Table "staging.und_tc_view_materialized"	256
UND Temperature Depth Data	258
Table "staging.und_td_curve_points"	258
Sequence "staging.und_td_curve_points_curve_point_id_seq"	258
Index "staging.und_td_curve_points_pkey"	258
Table "staging.und_td_sites"	258
Index "staging.und_td_sites_pkey"	259
Sequence "staging.und_td_sites_sites_id_seq"	259
View "staging.und_td_view"	259
Table "staging.und_td_view_materialized"	261
Well Log Headers	261
Sequence "staging.well_log_headers_id_seq"	261
Sequence "staging.well_log_headers_mapping_id_seq"	262

BEG Reservoir

Table "staging.beg_reservoir_data"

Column	Type	Modifiers	Storage	Description
reservoiruri	character varying(255)		extended	
reservoirname	character varying(255)		extended	
label	character varying(255)		extended	
otherid	character varying(255)		extended	
othername	character varying(255)		extended	
description	character varying(255)		extended	
county	character varying(255)		extended	
state	character varying(255)		extended	

locationname	character varying(255)	extended
latdegree	double precision	plain
longdegree	double precision	plain
srs	character varying(255)	extended
locationuncertaintystatement	character varying(255)	extended
temperaturemax_f	double precision	plain
temperaturemin_f	double precision	plain
depthtotop_ft	double precision	plain
depthtobottom_ft	double precision	plain
dominantformationname	character varying(255)	extended
otherformationname	character varying(255)	extended
dominantlithologyname	character varying(255)	extended
otherlithologyname	character varying(255)	extended
area_acres	double precision	plain
width_ft	double precision	plain
length_ft	double precision	plain
sandvolume_cft	double precision	plain
blockvolume_cft	double precision	plain
percentsand	double precision	plain
averagenetsandthickness_ft	double precision	plain
wellcount	integer	plain
faultcount	integer	plain
faulted	character varying(255)	extended
faulttype	character varying(255)	extended
faulthydrauliccharacter	character varying(255)	extended
firstproductiondate	date	plain
lastproductiondate	date	plain
years in production	integer	plain
monthlyproduction_bbl	double precision	plain
productionreportdate	date	plain
averagemonthlywaterproduction_bbl	double precision	plain
minimummonthlywaterproduction_bbl	double precision	plain
maximummonthlywaterproduction_bbl	double precision	plain
waterinjection_bbl	double precision	plain
watersaturation_pct	double precision	plain
watertds_mgl	double precision	plain
averagefieldwidepressure_psi	double precision	plain
averagefieldwidegradient_degfft	double precision	plain
porositymax_pct	double precision	plain
porositymin_pct	double precision	plain
permeabilitymax_md	double precision	plain
permeabilitymin_md	double precision	plain
cementationmineralogy	character varying(255)	extended
trap	character varying(255)	extended
drive	character varying(255)	extended
gravity	double precision	plain
commodityofinterest	character varying(255)	extended
stimulation	character varying(255)	extended
averageperforatedthickness_ft	double precision	plain
relatedresource	character varying(255)	extended
source	character varying(1023)	extended
metadataauri	character varying(255)	extended
majorfields	character varying(511)	extended
cordinate_nw_long	double precision	plain
cordinate_nw_lat	double precision	plain
cordinate_ne_long	double precision	plain

cordinate_ne_lat	double precision	plain
cordinate_sw_long	double precision	plain
cordinate_sw_lat	double precision	plain
cordinate_se_long	double precision	plain
cordinate_se_lat	double precision	plain
Has OIDs: no		

View "staging.beg_reservoir_view"

Column	Type	Modifiers	Storage	Description
reservoiruri	character varying(255)		extended	
reservoirname	character varying(255)		extended	
label	character varying(255)		extended	
otherid	character varying(255)		extended	
othername	character varying(255)		extended	
description	character varying(255)		extended	
county	character varying(255)		extended	
state	character varying(255)		extended	
locationname	character varying(255)		extended	
latdegree	double precision		plain	
longdegree	double precision		plain	
srs	character varying(255)		extended	
locationuncertaintystatement	character varying(255)		extended	
temperaturemax_f	double precision		plain	
temperaturemin_f	double precision		plain	
depthtop_ft	double precision		plain	
depthtobottom_ft	double precision		plain	
dominantformationname	character varying(255)		extended	
otherformationname	character varying(255)		extended	
dominantlithologyname	character varying(255)		extended	
otherlithologyname	character varying(255)		extended	
area_acres	double precision		plain	
width_ft	double precision		plain	
length_ft	double precision		plain	
sandvolume_cft	double precision		plain	
blockvolume_cft	double precision		plain	
percentsand	double precision		plain	
averagenetsandthickness_ft	double precision		plain	
wellcount	integer		plain	
faultcount	integer		plain	
faulted	character varying(255)		extended	
faulttype	character varying(255)		extended	
faulthydrauliccharacter	character varying(255)		extended	
firstproductiondate	date		plain	
lastproductiondate	date		plain	
years in production	integer		plain	
monthlyproduction_bbl	double precision		plain	
productionreportdate	date		plain	
averagemonthlywaterproduction_bbl	double precision		plain	
minimummonthlywaterproduction_bbl	double precision		plain	
maximummonthlywaterproduction_bbl	double precision		plain	
waterinjection_bbl	double precision		plain	
watersaturation_pct	double precision		plain	
water tds_mgl	double precision		plain	
averagefieldwidepressure_psi	double precision		plain	
averagefieldwidegradient_degfft	double precision		plain	

porositymax_pct	double precision	plain
porositymin_pct	double precision	plain
permeabilitymax_md	double precision	plain
permeabilitymin_md	double precision	plain
cementationmineralogy	character varying(255)	extended
trap	character varying(255)	extended
drive	character varying(255)	extended
gravity	double precision	plain
commodityofinterest	character varying(255)	extended
stimulation	character varying(255)	extended
averageperforatedthickness_ft	double precision	plain
relatedresource	character varying(255)	extended
source	character varying(1023)	extended
metadadatauri	character varying(255)	extended
majorfields	character varying(511)	extended
cordinate_nw_long	double precision	plain
cordinate_nw_lat	double precision	plain
cordinate_ne_long	double precision	plain
cordinate_ne_lat	double precision	plain
cordinate_sw_long	double precision	plain
cordinate_sw_lat	double precision	plain
cordinate_se_long	double precision	plain
cordinate_se_lat	double precision	plain

View definition:

```

SELECT beg_reservoir_data.reservoiruri, beg_reservoir_data.reservoirname, beg_reservoir_data.label,
beg_reservoir_data.otherid, beg_reservoir_data.othername, beg_reservoir_data.description,
beg_reservoir_data.county, beg_reservoir_data.state, beg_reservoir_data.locationname,
beg_reservoir_data.latdegree, beg_reservoir_data.longdegree, beg_reservoir_data.srs,
beg_reservoir_data.locationuncertaintystatement, beg_reservoir_data.temperaturemax_f,
beg_reservoir_data.temperaturemin_f, beg_reservoir_data.depthtotop_ft, beg_reservoir_data.depthtobottom_ft,
beg_reservoir_data.dominantformationname, beg_reservoir_data.otherformationname,
beg_reservoir_data.dominantlithologyname, beg_reservoir_data.otherlithologyname,
beg_reservoir_data.area_acres, beg_reservoir_data.width_ft, beg_reservoir_data.length_ft,
beg_reservoir_data.sandvolume_cft, beg_reservoir_data.blockvolume_cft, beg_reservoir_data.percentsand,
beg_reservoir_data.averagenetsandthickness_ft, beg_reservoir_data.wellcount, beg_reservoir_data.faultcount,
beg_reservoir_data.faulted, beg_reservoir_data.faulttype, beg_reservoir_data.faulthydrauliccharacter,
beg_reservoir_data.firstproductiondate, beg_reservoir_data.lastproductiondate,
beg_reservoir_data.yearsinproduction, beg_reservoir_data.monthlyproduction_bbl,
beg_reservoir_data.productionreportdate, beg_reservoir_data.averagemonthlywaterproduction_bbl,
beg_reservoir_data.minimummonthlywaterproduction_bbl,
beg_reservoir_data.maximummonthlywaterproduction_bbl, beg_reservoir_data.waterinjection_bbl,
beg_reservoir_data.watersaturation_pct, beg_reservoir_data.watertds_mgl,
beg_reservoir_data.averagefieldwidepressure_psi, beg_reservoir_data.averagefieldwidegradient_degfft,
beg_reservoir_data.porositymax_pct, beg_reservoir_data.porositymin_pct,
beg_reservoir_data.permeabilitymax_md, beg_reservoir_data.permeabilitymin_md,
beg_reservoir_data.cementationmineralogy, beg_reservoir_data.trap, beg_reservoir_data.drive,
beg_reservoir_data.gravity, beg_reservoir_data.commodityofinterest, beg_reservoir_data.stimulation,
beg_reservoir_data.averageperforatedthickness_ft, beg_reservoir_data.relatedresource, beg_reservoir_data.source,
beg_reservoir_data.metadadatauri, beg_reservoir_data.majorfields, beg_reservoir_data.cordinate_nw_long,
beg_reservoir_data.cordinate_nw_lat, beg_reservoir_data.cordinate_ne_long, beg_reservoir_data.cordinate_ne_lat,
beg_reservoir_data.cordinate_sw_long, beg_reservoir_data.cordinate_sw_lat,
beg_reservoir_data.cordinate_se_long, beg_reservoir_data.cordinate_se_lat
FROM staging.beg_reservoir_data;
```

Table "staging.beg_reservoir_view_materialized"

Column	Type	Modifiers	Storage	Description
reservoiruri	character varying(255)		extended	
reservoirname	character varying(255)		extended	
label	character varying(255)		extended	
otherid	character varying(255)		extended	
othername	character varying(255)		extended	
description	character varying(255)		extended	
county	character varying(255)		extended	
state	character varying(255)		extended	
locationname	character varying(255)		extended	
latdegree	double precision		plain	
longdegree	double precision		plain	
srs	character varying(255)		extended	
locationuncertaintystatement	character varying(255)		extended	
temperaturemax_f	double precision		plain	
temperaturemin_f	double precision		plain	
depthtotop_ft	double precision		plain	
depthtobottom_ft	double precision		plain	
dominantformationname	character varying(255)		extended	
otherformationname	character varying(255)		extended	
dominantlithologyname	character varying(255)		extended	
otherlithologyname	character varying(255)		extended	
area_acres	double precision		plain	
width_ft	double precision		plain	
length_ft	double precision		plain	
sandvolume_cft	double precision		plain	
blockvolume_cft	double precision		plain	
percentsand	double precision		plain	
averagenetsandthickness_ft	double precision		plain	
wellcount	integer		plain	
faultcount	integer		plain	
faulted	character varying(255)		extended	
faulttype	character varying(255)		extended	
faulthydrauliccharacter	character varying(255)		extended	
firstproductiondate	date		plain	
lastproductiondate	date		plain	
years in production	integer		plain	
monthlyproduction_bbl	double precision		plain	
productionreportdate	date		plain	
averagemonthlywaterproduction_bbl	double precision		plain	
minimummonthlywaterproduction_bbl	double precision		plain	
maximummonthlywaterproduction_bbl	double precision		plain	
waterinjection_bbl	double precision		plain	
watersaturation_pct	double precision		plain	
water tds_mgl	double precision		plain	
averagefieldwidepressure_psi	double precision		plain	
averagefieldwidegradient_degfft	double precision		plain	
porositymax_pct	double precision		plain	
porositymin_pct	double precision		plain	
permeabilitymax_md	double precision		plain	
permeabilitymin_md	double precision		plain	
cementationmineralogy	character varying(255)		extended	
trap	character varying(255)		extended	
drive	character varying(255)		extended	
gravity	double precision		plain	

commodityofinterest	character varying(255)	extended
stimulation	character varying(255)	extended
averageperforatedthickness_ft	double precision	plain
relatedresource	character varying(255)	extended
source	character varying(1023)	extended
metadatauri	character varying(255)	extended
majorfields	character varying(511)	extended
cordinate_nw_long	double precision	plain
cordinate_nw_lat	double precision	plain
cordinate_ne_long	double precision	plain
cordinate_ne_lat	double precision	plain
cordinate_sw_long	double precision	plain
cordinate_sw_lat	double precision	plain
cordinate_se_long	double precision	plain
cordinate_se_lat	double precision	plain
Has OIDs: no		

BEG Well Data

Table "staging.beg_well_data"

Column	Type	Modifiers	Storage	Description
api	bigint		plain	
county	character varying(64)		extended	
lease	character varying(64)		extended	
formation	character varying(64)		extended	
fm_at_td	character varying(64)		extended	
lat	double precision		plain	
long	double precision		plain	
tvd	double precision		plain	
depth	double precision		plain	
bht	double precision		plain	
depth_m	double precision		plain	
bht_c	double precision		plain	
bht_c_cor	double precision		plain	
bht_f_cor	double precision		plain	
gradient_per_1000ft	double precision		plain	
source	character varying(64)		extended	
wellname	character varying(64)		extended	
wellid	character varying(64)		extended	
logid	character varying(32)		extended	
number_run	integer		plain	
log_date	date		plain	
location	character varying(256)		extended	
operator_n	character varying(64)		extended	
accession_log_beg	character varying(32)		extended	
state_name	character varying(32)		extended	
casing_size_inches	double precision		plain	
casing_depth	double precision		plain	
liner_inches	double precision		plain	
liner_top	double precision		plain	
liner_base	double precision		plain	
log_types	character varying(128)		extended	
in_core_db	character varying(16)		extended	
accession_core_beg	character varying(32)		extended	
prod_reservoir_1	character varying(64)		extended	

prod_reservoir_2	character varying(64)	extended
prod_reservoir_3	character varying(64)	extended
prod_reservoir_4	character varying(64)	extended
prod_reservoir_5	character varying(64)	extended
prod_reservoir_6	character varying(64)	extended
prod_reservoir_7	character varying(64)	extended
prod_reservoir_8	character varying(64)	extended
sum_of_perfs	double precision	plain
top_ft_1	double precision	plain
bottom_ft_1	double precision	plain
interval_ft_1	double precision	plain
top_ft_2	double precision	plain
bottom_ft_2	double precision	plain
interval_ft_2	double precision	plain
top_ft_3	double precision	plain
bottom_ft_3	double precision	plain
interval_ft_3	double precision	plain
top_ft_4	double precision	plain
bottom_ft_4	double precision	plain
interval_ft_4	double precision	plain
top_ft_5	double precision	plain
bottom_ft_5	double precision	plain
interval_ft_5	double precision	plain
top_ft_6	double precision	plain
bottom_ft_6	double precision	plain
interval_ft_6	double precision	plain
top_ft_7	double precision	plain
bottom_ft_7	double precision	plain
interval_ft_7	double precision	plain
top_ft_8	double precision	plain
bottom_ft_8	double precision	plain
interval_ft_8	double precision	plain
watrvol_1_bbl	double precision	plain
watrvol_2_bbl	double precision	plain
watrvol_3_bbl	double precision	plain
watrvol_4_bbl	double precision	plain
watrvol_5_bbl	double precision	plain
watrvol_6_bbl	double precision	plain
watrvol_7_bbl	double precision	plain
watrvol_8_bbl	double precision	plain
bhp_1	double precision	plain
bhp_2	double precision	plain
bhp_3	double precision	plain
bhp_4	double precision	plain
bhp_5	double precision	plain
bhp_6	double precision	plain
bhp_7	double precision	plain
bhp_8	double precision	plain
durtn_hrs_1	double precision	plain
durtn_hrs_2	double precision	plain
durtn_hrs_3	double precision	plain
durtn_hrs_4	double precision	plain
durtn_hrs_5	double precision	plain
durtn_hrs_6	double precision	plain
durtn_hrs_7	double precision	plain
durtn_hrs_8	double precision	plain

testtype	character varying(16)	extended
choke_mm	double precision	plain
Has OIDs: no		

View "staging.beg_well_view"

Column	Type	Modifiers	Storage	Description
id	uuid		plain	
api	bigint		plain	
county	character varying(64)		extended	
lease	character varying(64)		extended	
formation	character varying(64)		extended	
fm_at_td	character varying(64)		extended	
lat	double precision		plain	
long	double precision		plain	
tvd	double precision		plain	
depth	double precision		plain	
bht	double precision		plain	
depth_m	double precision		plain	
bht_c	double precision		plain	
bht_c_cor	double precision		plain	
bht_f_cor	double precision		plain	
gradient_per_1000ft	double precision		plain	
source	character varying(64)		extended	
wellname	character varying(64)		extended	
wellid	character varying(64)		extended	
logid	character varying(32)		extended	
number_run	integer		plain	
log_date	date		plain	
location	character varying(256)		extended	
operator_n	character varying(64)		extended	
accession_log_beg	character varying(32)		extended	
state_name	character varying(32)		extended	
casing_size_inches	double precision		plain	
casing_depth	double precision		plain	
liner_inches	double precision		plain	
liner_top	double precision		plain	
liner_base	double precision		plain	
log_types	character varying(128)		extended	
in_core_db	character varying(16)		extended	
accession_core_beg	character varying(32)		extended	
prod_reservoir_1	character varying(64)		extended	
prod_reservoir_2	character varying(64)		extended	
prod_reservoir_3	character varying(64)		extended	
prod_reservoir_4	character varying(64)		extended	
prod_reservoir_5	character varying(64)		extended	
prod_reservoir_6	character varying(64)		extended	
prod_reservoir_7	character varying(64)		extended	
prod_reservoir_8	character varying(64)		extended	
sum_of_perfs	double precision		plain	
top_ft_1	double precision		plain	
bottom_ft_1	double precision		plain	
interval_ft_1	double precision		plain	
top_ft_2	double precision		plain	
bottom_ft_2	double precision		plain	
interval_ft_2	double precision		plain	

top_ft_3	double precision	plain
bottom_ft_3	double precision	plain
interval_ft_3	double precision	plain
top_ft_4	double precision	plain
bottom_ft_4	double precision	plain
interval_ft_4	double precision	plain
top_ft_5	double precision	plain
bottom_ft_5	double precision	plain
interval_ft_5	double precision	plain
top_ft_6	double precision	plain
bottom_ft_6	double precision	plain
interval_ft_6	double precision	plain
top_ft_7	double precision	plain
bottom_ft_7	double precision	plain
interval_ft_7	double precision	plain
top_ft_8	double precision	plain
bottom_ft_8	double precision	plain
interval_ft_8	double precision	plain
watrvol_1_bbl	double precision	plain
watrvol_2_bbl	double precision	plain
watrvol_3_bbl	double precision	plain
watrvol_4_bbl	double precision	plain
watrvol_5_bbl	double precision	plain
watrvol_6_bbl	double precision	plain
watrvol_7_bbl	double precision	plain
watrvol_8_bbl	double precision	plain
bhp_1	double precision	plain
bhp_2	double precision	plain
bhp_3	double precision	plain
bhp_4	double precision	plain
bhp_5	double precision	plain
bhp_6	double precision	plain
bhp_7	double precision	plain
bhp_8	double precision	plain
durtn_hrs_1	double precision	plain
durtn_hrs_2	double precision	plain
durtn_hrs_3	double precision	plain
durtn_hrs_4	double precision	plain
durtn_hrs_5	double precision	plain
durtn_hrs_6	double precision	plain
durtn_hrs_7	double precision	plain
durtn_hrs_8	double precision	plain
testtype	character varying(16)	extended
choke_mm	double precision	plain

View definition:

```
SELECT core.generate_uuid('BEG'::character varying, 'SURFACE_SITE'::character varying, ((beg_well_data.lat || ','::text) || beg_well_data.long)::character varying) AS id, beg_well_data.api, beg_well_data.county, beg_well_data.lease, beg_well_data.formation, beg_well_data.fm_at_td, beg_well_data.lat, beg_well_data.long, beg_well_data.tvd, beg_well_data.depth, beg_well_data.bht, beg_well_data.depth_m, beg_well_data.bht_c, beg_well_data.bht_c_cor, beg_well_data.bht_f_cor, beg_well_data.gradient_per_1000ft, beg_well_data.source, beg_well_data.wellname, beg_well_data.wellid, beg_well_data.logid, beg_well_data.number_run, beg_well_data.log_date, beg_well_data.location, beg_well_data.operator_n, beg_well_data.accession_log_beg, beg_well_data.state_name, beg_well_data.casing_size_inches, beg_well_data.casing_depth,
```

beg_well_data.liner_inches, beg_well_data.liner_top, beg_well_data.liner_base, beg_well_data.log_types,
 beg_well_data.in_core_db, beg_well_data.accession_core_beg, beg_well_data.prod_reservoir_1,
 beg_well_data.prod_reservoir_2, beg_well_data.prod_reservoir_3, beg_well_data.prod_reservoir_4,
 beg_well_data.prod_reservoir_5, beg_well_data.prod_reservoir_6, beg_well_data.prod_reservoir_7,
 beg_well_data.prod_reservoir_8, beg_well_data.sum_of_perfs, beg_well_data.top_ft_1,
 beg_well_data.bottom_ft_1, beg_well_data.interval_ft_1, beg_well_data.top_ft_2, beg_well_data.bottom_ft_2,
 beg_well_data.interval_ft_2, beg_well_data.top_ft_3, beg_well_data.bottom_ft_3, beg_well_data.interval_ft_3,
 beg_well_data.top_ft_4, beg_well_data.bottom_ft_4, beg_well_data.interval_ft_4, beg_well_data.top_ft_5,
 beg_well_data.bottom_ft_5, beg_well_data.interval_ft_5, beg_well_data.top_ft_6, beg_well_data.bottom_ft_6,
 beg_well_data.interval_ft_6, beg_well_data.top_ft_7, beg_well_data.bottom_ft_7, beg_well_data.interval_ft_7,
 beg_well_data.top_ft_8, beg_well_data.bottom_ft_8, beg_well_data.interval_ft_8, beg_well_data.watrvol_1_bbl,
 beg_well_data.watrvol_2_bbl, beg_well_data.watrvol_3_bbl, beg_well_data.watrvol_4_bbl,
 beg_well_data.watrvol_5_bbl, beg_well_data.watrvol_6_bbl, beg_well_data.watrvol_7_bbl,
 beg_well_data.watrvol_8_bbl, beg_well_data.bhp_1, beg_well_data.bhp_2, beg_well_data.bhp_3,
 beg_well_data.bhp_4, beg_well_data.bhp_5, beg_well_data.bhp_6, beg_well_data.bhp_7, beg_well_data.bhp_8,
 beg_well_data.durtn_hrs_1, beg_well_data.durtn_hrs_2, beg_well_data.durtn_hrs_3, beg_well_data.durtn_hrs_4,
 beg_well_data.durtn_hrs_5, beg_well_data.durtn_hrs_6, beg_well_data.durtn_hrs_7, beg_well_data.durtn_hrs_8,
 beg_well_data.testtype, beg_well_data.choke_mm

FROM staging.beg_well_data;

Table "staging.beg_well_view_materialized"

Column	Type	Modifiers	Storage	Description
id	uuid		plain	
api	bigint		plain	
county	character varying(64)		extended	
lease	character varying(64)		extended	
formation	character varying(64)		extended	
fm_at_td	character varying(64)		extended	
lat	double precision		plain	
long	double precision		plain	
tvd	double precision		plain	
depth	double precision		plain	
bht	double precision		plain	
depth_m	double precision		plain	
bht_c	double precision		plain	
bht_c_cor	double precision		plain	
bht_f_cor	double precision		plain	
gradient_per_1000ft	double precision		plain	
source	character varying(64)		extended	
wellname	character varying(64)		extended	
wellid	character varying(64)		extended	
logid	character varying(32)		extended	
number_run	integer		plain	
log_date	date		plain	
location	character varying(256)		extended	
operator_n	character varying(64)		extended	
accession_log_beg	character varying(32)		extended	
state_name	character varying(32)		extended	
casing_size_inches	double precision		plain	
casing_depth	double precision		plain	

liner_inches	double precision	plain
liner_top	double precision	plain
liner_base	double precision	plain
log_types	character varying(128)	extended
in_core_db	character varying(16)	extended
accession_core_beg	character varying(32)	extended
prod_reservoir_1	character varying(64)	extended
prod_reservoir_2	character varying(64)	extended
prod_reservoir_3	character varying(64)	extended
prod_reservoir_4	character varying(64)	extended
prod_reservoir_5	character varying(64)	extended
prod_reservoir_6	character varying(64)	extended
prod_reservoir_7	character varying(64)	extended
prod_reservoir_8	character varying(64)	extended
sum_of_perfs	double precision	plain
top_ft_1	double precision	plain
bottom_ft_1	double precision	plain
interval_ft_1	double precision	plain
top_ft_2	double precision	plain
bottom_ft_2	double precision	plain
interval_ft_2	double precision	plain
top_ft_3	double precision	plain
bottom_ft_3	double precision	plain
interval_ft_3	double precision	plain
top_ft_4	double precision	plain
bottom_ft_4	double precision	plain
interval_ft_4	double precision	plain
top_ft_5	double precision	plain
bottom_ft_5	double precision	plain
interval_ft_5	double precision	plain
top_ft_6	double precision	plain
bottom_ft_6	double precision	plain
interval_ft_6	double precision	plain
top_ft_7	double precision	plain
bottom_ft_7	double precision	plain
interval_ft_7	double precision	plain
top_ft_8	double precision	plain
bottom_ft_8	double precision	plain
interval_ft_8	double precision	plain
watrvol_1_bbl	double precision	plain
watrvol_2_bbl	double precision	plain
watrvol_3_bbl	double precision	plain
watrvol_4_bbl	double precision	plain
watrvol_5_bbl	double precision	plain
watrvol_6_bbl	double precision	plain
watrvol_7_bbl	double precision	plain
watrvol_8_bbl	double precision	plain
bhp_1	double precision	plain
bhp_2	double precision	plain
bhp_3	double precision	plain
bhp_4	double precision	plain
bhp_5	double precision	plain
bhp_6	double precision	plain
bhp_7	double precision	plain
bhp_8	double precision	plain
durtn_hrs_1	double precision	plain

durtn_hrs_2	double precision	plain
durtn_hrs_3	double precision	plain
durtn_hrs_4	double precision	plain
durtn_hrs_5	double precision	plain
durtn_hrs_6	double precision	plain
durtn_hrs_7	double precision	plain
durtn_hrs_8	double precision	plain
testtype	character varying(16)	extended
choke_mm	double precision	plain
Has OIDs: no		

Cornell Well Data

Table "staging.cornell_well_data"

Column	Type	Modifiers	Storage	Description
api_wellno	character varying(16)		extended	
bht_f	double precision		plain	
bht_c	double precision		plain	
harrison_correction	double precision		plain	
harrison_bht_c	double precision		plain	
log_depth_f	double precision		plain	
log_depth_m	double precision		plain	
tvf_f	double precision		plain	
tvf_m	double precision		plain	
calc_depth_m	double precision		plain	
harrison_corr_grad_c_km	double precision		plain	
uncorr_grad	double precision		plain	
county	character varying(64)		extended	
cosuna_col	character varying(8)		extended	
town	character varying(64)		extended	
permit_number	character varying(16)		extended	
operator	character varying(128)		extended	
farm_name	character varying(128)		extended	
td_date	date		plain	
log_date	date		plain	
comp_date	date		plain	
well_type	character varying(64)		extended	
well_status	character varying(64)		extended	
well_nm	character varying(64)		extended	
elevation	double precision		plain	
field	character varying(64)		extended	
pool	character varying(64)		extended	
municipality	character varying(64)		extended	
quad	character varying(64)		extended	
map_section	integer		plain	
tdform	character varying(64)		extended	
interval_id	character varying(64)		extended	
site_heat_flow	double precision		plain	
site_value_method	character varying(64)		extended	
interval_quality_code	character varying(16)		extended	
decimal_longitude	double precision		plain	
decimal_latitude	double precision		plain	
sed_thickness_m	double precision		plain	
heat_flow_mw_m2	double precision		plain	
average_conductivity	double precision		plain	

data_source	character varying(512)	extended
calc_basement_heat_prod	double precision	plain
temp_est_1500m	double precision	plain
temp_est_3000m	double precision	plain
temp_est_4500m	double precision	plain
temp_est_6000m	double precision	plain
temp_est_7500m	double precision	plain
temp_est_10000m	double precision	plain
est_depth_150c	double precision	plain
est_depth_80c	double precision	plain
Has OIDs: no		

View "staging.cornell_well_view"

Column	Type	Modifiers	Storage	Description
id	uuid		plain	
api_wellno	character varying(16)		extended	
bht_f	double precision		plain	
bht_c	double precision		plain	
harrison_correction	double precision		plain	
harrison_bht_c	double precision		plain	
log_depth_f	double precision		plain	
log_depth_m	double precision		plain	
tvf_f	double precision		plain	
tvf_m	double precision		plain	
calc_depth_m	double precision		plain	
harrison_corr_grad_c_km	double precision		plain	
uncorr_grad	double precision		plain	
county	character varying(64)		extended	
cosuna_col	character varying(8)		extended	
town	character varying(64)		extended	
permit_number	character varying(16)		extended	
operator	character varying(128)		extended	
farm_name	character varying(128)		extended	
td_date	date		plain	
log_date	date		plain	
comp_date	date		plain	
well_type	character varying(64)		extended	
well_status	character varying(64)		extended	
well_nm	character varying(64)		extended	
elevation	double precision		plain	
field	character varying(64)		extended	
pool	character varying(64)		extended	
municipality	character varying(64)		extended	
quad	character varying(64)		extended	
map_section	integer		plain	
tdform	character varying(64)		extended	
interval_id	character varying(64)		extended	
site_heat_flow	double precision		plain	
site_value_method	character varying(64)		extended	
interval_quality_code	character varying(16)		extended	
decimal_longitude	double precision		plain	
decimal_latitude	double precision		plain	
sed_thickness_m	double precision		plain	
heat_flow_mw_m2	double precision		plain	
average_conductivity	double precision		plain	

data_source	character varying(512)	extended
calc_basement_heat_prod	double precision	plain
temp_est_1500m	double precision	plain
temp_est_3000m	double precision	plain
temp_est_4500m	double precision	plain
temp_est_6000m	double precision	plain
temp_est_7500m	double precision	plain
temp_est_10000m	double precision	plain
est_depth_150c	double precision	plain
est_depth_80c	double precision	plain

View definition:

```
SELECT core.generate_uuid('CORNELL'::character varying, 'SURFACE_SITE'::character varying,
cornell_well_data.api_wellno) AS id, cornell_well_data.api_wellno, cornell_well_data.bht_f,
cornell_well_data.bht_c, cornell_well_data.harrison_correction, cornell_well_data.harrison_bht_c,
cornell_well_data.log_depth_f, cornell_well_data.log_depth_m, cornell_well_data.tvd_f, cornell_well_data.tvd_m,
cornell_well_data.calc_depth_m, cornell_well_data.harrison_corr_grad_c_km, cornell_well_data.uncorr_grad,
cornell_well_data.county, cornell_well_data.cosuna_col, cornell_well_data.town,
cornell_well_data.permit_number, cornell_well_data.operator, cornell_well_data.farm_name,
cornell_well_data.td_date, cornell_well_data.log_date, cornell_well_data.comp_date, cornell_well_data.well_type,
cornell_well_data.well_status, cornell_well_data.well_nm, cornell_well_data.elevation, cornell_well_data.field,
cornell_well_data.pool, cornell_well_data.municipality, cornell_well_data.quad, cornell_well_data.map_section,
cornell_well_data.tdform, cornell_well_data.interval_id, cornell_well_data.site_heat_flow,
cornell_well_data.site_value_method, cornell_well_data.interval_quality_code,
cornell_well_data.decimal_longitude, cornell_well_data.decimal_latitude, cornell_well_data.sed_thickness_m,
cornell_well_data.heat_flow_mw_m2, cornell_well_data.average_conductivity, cornell_well_data.data_source,
cornell_well_data.calc_basement_heat_prod, cornell_well_data.temp_est_1500m,
cornell_well_data.temp_est_3000m, cornell_well_data.temp_est_4500m, cornell_well_data.temp_est_6000m,
cornell_well_data.temp_est_7500m, cornell_well_data.temp_est_10000m, cornell_well_data.est_depth_150c,
cornell_well_data.est_depth_80c
FROM staging.cornell_well_data;
```

Table "staging.cornell_well_view_materialized"

Column	Type	Modifiers	Storage	Description
id	uuid		plain	
api_wellno	character varying(16)		extended	
bht_f	double precision		plain	
bht_c	double precision		plain	
harrison_correction	double precision		plain	
harrison_bht_c	double precision		plain	
log_depth_f	double precision		plain	
log_depth_m	double precision		plain	
tvd_f	double precision		plain	
tvd_m	double precision		plain	
calc_depth_m	double precision		plain	
harrison_corr_grad_c_km	double precision		plain	
uncorr_grad	double precision		plain	
county	character varying(64)		extended	
cosuna_col	character varying(8)		extended	
town	character varying(64)		extended	
permit_number	character varying(16)		extended	
operator	character varying(128)		extended	
farm_name	character varying(128)		extended	
td_date	date		plain	

log_date	date	plain
comp_date	date	plain
well_type	character varying(64)	extended
well_status	character varying(64)	extended
well_nm	character varying(64)	extended
elevation	double precision	plain
field	character varying(64)	extended
pool	character varying(64)	extended
municipality	character varying(64)	extended
quad	character varying(64)	extended
map_section	integer	plain
tdform	character varying(64)	extended
interval_id	character varying(64)	extended
site_heat_flow	double precision	plain
site_value_method	character varying(64)	extended
interval_quality_code	character varying(16)	extended
decimal_longitude	double precision	plain
decimal_latitude	double precision	plain
sed_thickness_m	double precision	plain
heat_flow_mw_m2	double precision	plain
average_conductivity	double precision	plain
data_source	character varying(512)	extended
calc_basement_heat_prod	double precision	plain
temp_est_1500m	double precision	plain
temp_est_3000m	double precision	plain
temp_est_4500m	double precision	plain
temp_est_6000m	double precision	plain
temp_est_7500m	double precision	plain
temp_est_10000m	double precision	plain
est_depth_150c	double precision	plain
est_depth_80c	double precision	plain
Has OIDs: no		

Counties

Table "staging.counties"

Column	Type	Modifiers	Storage	Description
gid	integer	not null default nextval('staging.countyp020_gid_seq' ::regclass)	plain	
area	double precision		plain	
perimeter	double precision		plain	
countyp020	double precision		plain	
state	character varying(2)		extended	
county	character varying(50)		extended	
state_fips	character varying(2)		extended	
square_mil	double precision		plain	
the_geom	geometry		main	
vcount	integer	not null default 0	plain	

Indexes:

"countyp020_pkey" PRIMARY KEY, btree (gid)

"countyp020_the_geom_gist" gist (the_geom)

Check constraints:

"enforce_dims_the_geom" CHECK (st_ndims(the_geom) = 2)

"enforce_geotype_the_geom" CHECK (geometrytype(the_geom) = 'MULTIPOLYGON'::text OR the_geom IS NULL)

"enforce_srid_the_geom" CHECK (st_srid(the_geom) = 4326)

Has OIDs: no

Sequence "staging.countyp020_gid_seq"

Column	Type	Value	Storage	Description
sequence_name	name	countyp020_gid_seq	plain	
last_value	bigint	6138	plain	
start_value	bigint	6138	plain	
increment_by	bigint	1	plain	
max_value	bigint	9223372036854775807	plain	
min_value	bigint	1	plain	
cache_value	bigint	1	plain	
log_cnt	bigint	1	plain	
is_cycled	boolean	f	plain	
is_called	boolean	f	plain	

Index "staging.countyp020_pkey"

Column	Type	Storage	Description
gid	integer	plain	
primary key, btree, for table "staging.counties"			

Index "staging.countyp020_the_geom_gist"

Column	Type	Storage	Description
the_geom	box2d	plain	
gist, for table "staging.counties"			

Field Reports ID

Sequence "staging.field_reports_id_seq"

Column	Type	Value	Storage	Description
sequence_name	name	field_reports_id_seq	plain	
last_value	bigint	4978	plain	
start_value	bigint	4978	plain	
increment_by	bigint	1	plain	
max_value	bigint	9223372036854775807	plain	
min_value	bigint	1	plain	
cache_value	bigint	1	plain	
log_cnt	bigint	1	plain	
is_cycled	boolean	f	plain	
is_called	boolean	f	plain	

Sequence "staging.field_reports_mapping_id_seq"

Column	Type	Value	Storage	Description
sequence_name	name	field_reports_mapping_id_seq	plain	
last_value	bigint	4978	plain	

start_value	bigint	4978	plain
increment_by	bigint	1	plain
max_value	bigint	9223372036854775807	plain
min_value	bigint	1	plain
cache_value	bigint	1	plain
log_cnt	bigint	1	plain
is_cycled	boolean	f	plain
is_called	boolean	f	plain

Mat Views

Table "staging.matviews"

Column	Type	Modifiers	Storage	Description
mv_name	name	not null	plain	
v_name	name	not null	plain	
last_refresh	timestamp with time zone		plain	

Indexes:

"matviews_pkey" PRIMARY KEY, btree (mv_name)

Has OIDs: no

Index "staging.matviews_pkey"

Column	Type	Storage	Description
mv_name	cstring	plain	

primary key, btree, for table "staging.matviews"

MLKay BHT Data

Table "staging.mlk_bht_data"

Column	Type	Modifiers	Storage	Description
api42	integer		plain	
county_nr	integer		plain	
drillertotaldepth	double precision		plain	
operator	character varying(128)		extended	
lease_name	character varying(128)		extended	
field	character varying(128)		extended	
measurementdatetime	date		plain	
measuredtemperaturef	double precision		plain	
casingpipediameter	double precision		plain	
bitdiametercollar	double precision		plain	
circulationduration	double precision		plain	
mud_weight	double precision		plain	
viscosity	double precision		plain	

Has OIDs: no

Publications

Table "staging.publication"

Column	Type	Modifiers	Storage	Description
publication_id	integer	not null default nextval('staging.publication_p ublication_id_seq'::regclass)	plain	

authors	character varying(1024)		extended
title	character varying(512)		extended
year	character varying(64)		extended
report_number	character varying(100)		extended
pages	character varying(100)		extended
agency	character varying(512)		extended
contract	character varying(128)		extended
source_number	character varying(64)		extended
source	character varying(64)		extended
location	character varying(128)		extended
pdf	character varying(32)		extended
electronic	character varying(32)		extended
url	character varying(256)		extended
month	numeric(2,0)		main
publisher	character varying(512)		extended
volume	character varying(128)		extended
issue	character varying(64)		extended
collection	character varying(256)		extended
keywords	character varying(1024)		extended
publication_uuid	uuid		plain
ocr_required	boolean	default false	plain
expected	boolean	default true	plain
dataset_id	uuid		plain
data_source_id	integer		plain
indexed	boolean		plain

Indexes:

"stg_publication_pkey" PRIMARY KEY, btree (publication_id)

Has OIDs: no

Sequence "staging.publication_publication_id_seq"

Column	Type	Value	Storage	Description
sequence_name	name	publication_publication_id_seq	plain	
last_value	bigint	1	plain	
start_value	bigint	1	plain	
increment_by	bigint	1	plain	
max_value	bigint	9223372036854775807	plain	
min_value	bigint	1	plain	
cache_value	bigint	1	plain	
log_cnt	bigint	1	plain	
is_cycled	boolean	f	plain	
is_called	boolean	f	plain	

SMU Heat Flow (EQ Data - Part 1 of 3)

Table "staging.smu2_hf_heatflow"

Column	Type	Modifiers	Storage	Description
gid	character varying(255)		extended	
interval_id	character varying(255)		extended	
depth	double precision		plain	
bottom	double precision		plain	
avg_tcu	double precision		plain	
avg_tcu_sym	character varying(5)		extended	
avg_tcu_std_err	double precision		plain	

bulk_tcu	double precision	plain
bulk_tcu_repeats	numeric(7,0)	main
porosity	double precision	plain
geoth_grad	double precision	plain
geoth_grad_sym	character varying(5)	extended
geoth_grad_std_err	double precision	plain
cgeoth_grad	double precision	plain
cgeoth_grad_sym	character varying(5)	extended
cgeoth_grad_std_err	double precision	plain
heat_flow	double precision	plain
heat_flow_sym	character varying(5)	extended
heat_flow_std_err	double precision	plain
cheat_flow	double precision	plain
cheat_flow_sym	character varying(5)	extended
cheat_flow_std_err	double precision	plain
heat_production	double precision	plain
heat_production_sym	character varying(5)	extended
heat_production_std_err	double precision	plain
num_hg	double precision	plain
aquifier_temp	double precision	plain
aquifier_temp_sym	character varying(5)	extended
aquifier_temp_std_err	double precision	plain
quality_code	character varying(10)	extended
Has OIDs: no		

SMU Heat Flow (EQ Data – Part 2 of 3)

Table "staging.smu2_hf_point"

Column	Type	Modifiers	Storage	Description
gid	character varying(64)	not null	extended	
area_or_ams_map	character varying(255)		extended	
hole_name	character varying(255)		extended	
pub_ref	character varying(255)		extended	
new_pub_ref	character varying(255)		extended	
tw_nrg_sec	character varying(255)		extended	
county	character varying(255)		extended	
state	character varying(255)		extended	
tect_prov	character varying(255)		extended	
w_long	numeric(10,7)		main	
n_lat	numeric(10,7)		main	
location_notes	character varying(255)		extended	
meas_date	date		plain	
drill_date	date		plain	
collar_elevation	double precision		plain	
drill_depth	double precision		plain	
water_table	double precision		plain	
water_symbold	character varying(255)		extended	
surf_temp	double precision		plain	
min_temp	double precision		plain	
max_temp	double precision		plain	
bot_temp	double precision		plain	
lit_info	character varying(255)		extended	
comment_info	character varying(255)		extended	

Indexes:

"smu2_hf_point_pkey" PRIMARY KEY, btree (gid)

Has OIDs: no

Index "staging.smu2_hf_point_pkey"

Column	Type	Storage	Description
gid	character varying(64)	extended	
primary key, btree, for table "staging.smu2_hf_point"			

SMU Heat Flow (EQ Data – Part 3 of 3)

Table "staging.smu2_hf_ref"

Column	Type	Modifiers	Storage	Description
pub_ref	character varying(256)		extended	
new_pub_ref	character varying(256)		extended	
info	text		extended	
smu_id	character varying(256)		extended	
link	character varying(512)		extended	
copyrighted	character varying(8)		extended	
second_link	character varying(256)		extended	
Has OIDs: no				

SMU BHT Heat Flow

Table "staging.smu_bht_data"

Column	Type	Modifiers	Storage	Description
database	character varying(128)		extended	
api	character varying(128)		extended	
permit	character varying(128)		extended	
longitude	numeric(10,7)		main	
latitude	numeric(10,7)		main	
surface_interval_id	character varying(64)		extended	
depth	double precision		plain	
uncor_bht	double precision		plain	
harrison_correction	character varying(128)		extended	
bhtcorrected_temp	double precision		plain	
surface_temp	double precision		plain	
harrison_gradient	double precision		plain	
reference_data_set	character varying(512)		extended	
state	character varying(128)		extended	
k_source	character varying(128)		extended	
k_section	character varying(128)		extended	
resistance	double precision		plain	
k	double precision		plain	
q_calculated	double precision		plain	
quality	character varying(128)		extended	
basement_depth	double precision		plain	
correction	character varying(128)		extended	
bht_well	double precision		plain	
well_ave_hf	double precision		plain	
file_number	character varying(128)		extended	
operation_name	character varying(128)		extended	
drilling_start	date		plain	

drilling_complete	date	plain
tsc	double precision	plain
state_plane_meter_x_coordinate	character varying(128)	extended
state_plane_meter_y_coordinate	character varying(128)	extended
company_name	character varying(128)	extended
well_type	character varying(128)	extended
well_status	character varying(128)	extended
field_name	character varying(128)	extended
formation	character varying(128)	extended
elevation_measured	character varying(128)	extended
elevation_ft	double precision	plain
elevation_m	double precision	plain
county_name	character varying(128)	extended
surface_id	character varying(128)	extended
bottom_id	character varying(128)	extended
Has OIDs: no		

View "staging.smu_bht_view"

Column	Type	Modifiers	Storage	Description
id	uuid		plain	
database	character varying(128)		extended	
api	character varying(128)		extended	
permit	character varying(128)		extended	
longitude	numeric(10,7)		main	
latitude	numeric(10,7)		main	
surface_interval_id	character varying(64)		extended	
depth	double precision		plain	
uncor_bht	double precision		plain	
harrison_correction	character varying(128)		extended	
bhtcorrected_temp	double precision		plain	
surface_temp	double precision		plain	
harrison_gradient	double precision		plain	
reference_data_set	character varying(512)		extended	
state	character varying(128)		extended	
k_source	character varying(128)		extended	
k_section	character varying(128)		extended	
resistance	double precision		plain	
k	double precision		plain	
q_calculated	double precision		plain	
quality	character varying(128)		extended	
basement_depth	double precision		plain	
correction	character varying(128)		extended	
bht_well	double precision		plain	
well_ave_hf	double precision		plain	
file_number	character varying(128)		extended	
operation_name	character varying(128)		extended	
drilling_start	date		plain	
drilling_complete	date		plain	
tsc	double precision		plain	
state_plane_meter_x_coordinate	character varying(128)		extended	
state_plane_meter_y_coordinate	character varying(128)		extended	
company_name	character varying(128)		extended	
well_type	character varying(128)		extended	
well_status	character varying(128)		extended	
field_name	character varying(128)		extended	

formation	character varying(128)	extended
elevation_measured	character varying(128)	extended
elevation_ft	double precision	plain
elevation_m	double precision	plain
county_name	character varying(128)	extended
surface_id	character varying(128)	extended
bottom_id	character varying(128)	extended

View definition:

```
SELECT core.generate_uuid('SMU'::character varying, 'SURFACE_SITE'::character varying,
(to_char(smu_bht_data.longitude, '999.9999'::text) || to_char(smu_bht_data.latitude, '999.9999'::text))::character
varying) AS id, smu_bht_data.database, smu_bht_data.api, smu_bht_data.permit, smu_bht_data.longitude,
smu_bht_data.latitude, smu_bht_data.surface_interval_id, smu_bht_data.depth, smu_bht_data.uncor_bht,
smu_bht_data.harrison_correction, smu_bht_data.bhtcorrected_temp, smu_bht_data.surface_temp,
smu_bht_data.harrison_gradient, smu_bht_data.reference_data_set, smu_bht_data.state, smu_bht_data.k_source,
smu_bht_data.k_section, smu_bht_data.resistance, smu_bht_data.k, smu_bht_data.q_calculated,
smu_bht_data.quality, smu_bht_data.basement_depth, smu_bht_data.correction, smu_bht_data.bht_well,
smu_bht_data.well_ave_hf, smu_bht_data.file_number, smu_bht_data.operation_name,
smu_bht_data.drilling_start, smu_bht_data.drilling_complete, smu_bht_data.tsc,
smu_bht_data.state_plane_meter_x_coordinate, smu_bht_data.state_plane_meter_y_coordinate,
smu_bht_data.company_name, smu_bht_data.well_type, smu_bht_data.well_status, smu_bht_data.field_name,
smu_bht_data.formation, smu_bht_data.elevation_measured, smu_bht_data.elevation_ft,
smu_bht_data.elevation_m, smu_bht_data.county_name, smu_bht_data.surface_id, smu_bht_data.bottom_id

FROM staging.smu_bht_data;
```

Table "staging.smu_bht_view_materialized"

Column	Type	Modifiers	Storage	Description
id	uuid		plain	
database	character varying(128)		extended	
api	character varying(128)		extended	
permit	character varying(128)		extended	
longitude	numeric(10,7)		main	
latitude	numeric(10,7)		main	
surface_interval_id	character varying(64)		extended	
depth	double precision		plain	
uncor_bht	double precision		plain	
harrison_correction	character varying(128)		extended	
bhtcorrected_temp	double precision		plain	
surface_temp	double precision		plain	
harrison_gradient	double precision		plain	
reference_data_set	character varying(512)		extended	
state	character varying(128)		extended	
k_source	character varying(128)		extended	
k_section	character varying(128)		extended	
resistance	double precision		plain	
k	double precision		plain	
q_calculated	double precision		plain	
quality	character varying(128)		extended	

basement_depth	double precision	plain
correction	character varying(128)	extended
bht_well	double precision	plain
well_ave_hf	double precision	plain
file_number	character varying(128)	extended
operation_name	character varying(128)	extended
drilling_start	date	plain
drilling_complete	date	plain
tsc	double precision	plain
state_plane_meter_x_coordinate	character varying(128)	extended
state_plane_meter_y_coordinate	character varying(128)	extended
company_name	character varying(128)	extended
well_type	character varying(128)	extended
well_status	character varying(128)	extended
field_name	character varying(128)	extended
formation	character varying(128)	extended
elevation_measured	character varying(128)	extended
elevation_ft	double precision	plain
elevation_m	double precision	plain
county_name	character varying(128)	extended
surface_id	character varying(128)	extended
bottom_id	character varying(128)	extended
Has OIDs: no		

SMU Heat Flow (EQ Data – All 3 of 3)

View "staging.smu_hf_view"

Column	Type	Modifiers	Storage	Description
id	uuid		plain	
gid	character varying(64)		extended	
latitude	numeric(10,7)		main	
longitude	numeric(10,7)		main	
area_or_ams_map	character varying(255)		extended	
hole_name	character varying(255)		extended	
twm_rmg_sec	character varying(255)		extended	
county	character varying(255)		extended	
state	character varying(255)		extended	
tect_prov	character varying(255)		extended	
location_notes	character varying(255)		extended	
meas_date	date		plain	
drill_date	date		plain	
collar_elevation	double precision		plain	
drill_depth	double precision		plain	
water_table	double precision		plain	
water_symbold	character varying(255)		extended	
surf_temp	double precision		plain	
min_temp	double precision		plain	
max_temp	double precision		plain	
bot_temp	double precision		plain	
lit_info	character varying(255)		extended	
comment_info	character varying(255)		extended	
interval_id	character varying(255)		extended	
depth	double precision		plain	
bottom	double precision		plain	
avg_tcu	double precision		plain	

avg_tcu_sym	character varying(5)	extended
avg_tcu_std_err	double precision	plain
bulk_tcu	double precision	plain
bulk_tcu_repeats	numeric(7,0)	main
porosity	double precision	plain
geoth_grad	double precision	plain
geoth_grad_sym	character varying(5)	extended
geoth_grad_std_err	double precision	plain
cgeoth_grad	double precision	plain
cgeoth_grad_sym	character varying(5)	extended
cgeoth_grad_std_err	double precision	plain
heat_flow	double precision	plain
heat_flow_sym	character varying(5)	extended
heat_flow_std_err	double precision	plain
cheat_flow	double precision	plain
cheat_flow_sym	character varying(5)	extended
cheat_flow_std_err	double precision	plain
heat_production	double precision	plain
heat_production_sym	character varying(5)	extended
heat_production_std_err	double precision	plain
num_hg	double precision	plain
aquifier_temp	double precision	plain
aquifier_temp_sym	character varying(5)	extended
aquifier_temp_std_err	double precision	plain
quality_code	character varying(10)	extended
pub_ref	character varying(256)	extended
new_pub_ref	character varying(256)	extended
info	text	extended
smu_id	character varying(256)	extended
link	character varying(512)	extended
copyrighted	character varying(8)	extended
second_link	character varying(256)	extended

View definition:

```

SELECT core.generate_uuid('SMU'::character varying, 'SURFACE_SITE'::character varying, pt.gid) AS id,
pt.gid, pt.n_lat AS latitude, pt.w_long AS longitude, pt.area_or_ams_map, pt.hole_name, pt.twn_rng_sec,
pt.county, pt.state, pt.tect_prov, pt.location_notes, pt.meas_date, pt.drill_date, pt.collar_elevation, pt.drill_depth,
pt.water_table, pt.water_symbol, pt.surf_temp, pt.min_temp, pt.max_temp, pt.bot_temp, pt.lit_info,
pt.comment_info, hf.interval_id, hf.depth, hf.bottom, hf.avg_tcu, hf.avg_tcu_sym, hf.avg_tcu_std_err, hf.bulk_tcu,
hf.bulk_tcu_repeats, hf.porosity, hf.geoth_grad, hf.geoth_grad_sym, hf.geoth_grad_std_err, hf.cgeoth_grad,
hf.cgeoth_grad_sym, hf.cgeoth_grad_std_err, hf.heat_flow, hf.heat_flow_sym, hf.heat_flow_std_err,
hf.cheat_flow, hf.cheat_flow_sym, hf.cheat_flow_std_err, hf.heat_production, hf.heat_production_sym,
hf.heat_production_std_err, hf.num_hg, hf.aquifier_temp, hf.aquifier_temp_sym, hf.aquifier_temp_std_err,
hf.quality_code, ref.pub_ref, ref.new_pub_ref, ref.info, ref.smu_id, ref.link, ref.copyrighted, ref.second_link
FROM staging.smu2_hf_point pt
LEFT JOIN staging.smu2_hf_heatflow hf ON pt.gid::text = hf.gid::text
LEFT JOIN staging.smu2_hf_ref ref ON pt.new_pub_ref::text = ref.new_pub_ref::text
WHERE length(pt.gid::text) > 0;

```

Table "staging.smu_hf_view_materialized"

Column	Type	Modifiers	Storage	Description
id	uuid		plain	
gid	character varying(64)		extended	
latitude	numeric(10,7)		main	

longitude	numeric(10,7)	main
area_or_ams_map	character varying(255)	extended
hole_name	character varying(255)	extended
twm_rng_sec	character varying(255)	extended
county	character varying(255)	extended
state	character varying(255)	extended
tect_prov	character varying(255)	extended
location_notes	character varying(255)	extended
meas_date	date	plain
drill_date	date	plain
collar_elevation	double precision	plain
drill_depth	double precision	plain
water_table	double precision	plain
water_symbold	character varying(255)	extended
surf_temp	double precision	plain
min_temp	double precision	plain
max_temp	double precision	plain
bot_temp	double precision	plain
lit_info	character varying(255)	extended
comment_info	character varying(255)	extended
interval_id	character varying(255)	extended
depth	double precision	plain
bottom	double precision	plain
avg_tcu	double precision	plain
avg_tcu_sym	character varying(5)	extended
avg_tcu_std_err	double precision	plain
bulk_tcu	double precision	plain
bulk_tcu_repeats	numeric(7,0)	main
porosity	double precision	plain
geoth_grad	double precision	plain
geoth_grad_sym	character varying(5)	extended
geoth_grad_std_err	double precision	plain
cgeoth_grad	double precision	plain
cgeoth_grad_sym	character varying(5)	extended
cgeoth_grad_std_err	double precision	plain
heat_flow	double precision	plain
heat_flow_sym	character varying(5)	extended
heat_flow_std_err	double precision	plain
cheat_flow	double precision	plain
cheat_flow_sym	character varying(5)	extended
cheat_flow_std_err	double precision	plain
heat_production	double precision	plain
heat_production_sym	character varying(5)	extended
heat_production_std_err	double precision	plain
num_hg	double precision	plain
aquifier_temp	double precision	plain
aquifier_temp_sym	character varying(5)	extended
aquifier_temp_std_err	double precision	plain
quality_code	character varying(10)	extended
pub_ref	character varying(256)	extended
new_pub_ref	character varying(256)	extended
info	text	extended
smu_id	character varying(256)	extended
link	character varying(512)	extended
copyrighted	character varying(8)	extended
second_link	character varying(256)	extended

Has OIDs: no

States

Table "staging.states"

Column	Type	Modifiers	Storage	Description
gid	integer	not null default nextval('staging.states_gid_seq'::regclass)	plain	
name	character varying(25)		extended	
fips_code	character varying(2)		extended	
sub_region	character varying(7)		extended	
abbreviation	character varying(2)		extended	
shape	geometry		main	
vcount	integer	not null default 0	plain	

Indexes:

"states_pkey" PRIMARY KEY, btree (gid)

"states_the_geom_gist" gist (shape)

Check constraints:

"enforce_dims_the_geom" CHECK (st_ndims(shape) = 2)

"enforce_geotype_the_geom" CHECK (geometrytype(shape) = 'MULTIPOLYGON'::text OR shape IS NULL)

"enforce_srid_the_geom" CHECK (st_srid(shape) = 4326)

Has OIDs: no

Sequence "staging.states_gid_seq"

Column	Type	Value	Storage	Description
sequence_name	name	states_gid_seq	plain	
last_value	bigint	1	plain	
start_value	bigint	1	plain	
increment_by	bigint	1	plain	
max_value	bigint	9223372036854775807	plain	
min_value	bigint	1	plain	
cache_value	bigint	1	plain	
log_cnt	bigint	1	plain	
is_cycled	boolean	f	plain	
is_called	boolean	f	plain	

Index "staging.states_pkey"

Column	Type	Storage	Description
gid	integer	plain	
primary key, btree, for table "staging.states"			

Index "staging.states_the_geom_gist"

Column	Type	Storage	Description
shape	box2d	plain	
gist, for table "staging.states"			

Publications

Index "staging.stg_publication_pkey"

Column	Type	Storage	Description

publication_id	integer	plain
primary key, btree, for table "staging.publication"		

TTU Well Data (Part 1 of 3)

Table "staging.tt_corrected_bht"

Column	Type	Modifiers	Storage	Description
objectid	integer		plain	
block	character varying(16)		extended	
api	character varying(16)		extended	
ssf_depth	double precision		plain	
VRT	double precision		plain	
ssf_depth_ft	double precision		plain	
VRT_f	double precision		plain	
Has OIDs: no				

TTU Well Data (Part 2 of 3)

Table "staging.tt_uncorrected_bht"

Column	Type	Modifiers	Storage	Description
objectid	integer		plain	
file_name	character varying(64)		extended	
lease_area	character varying(32)		extended	
addition	character varying(32)		extended	
block	character varying(32)		extended	
api	character varying(16)		extended	
depth	character varying(32)		extended	
bht	character varying(32)		extended	
depth1	character varying(32)		extended	
bht1	character varying(32)		extended	
depth2	character varying(32)		extended	
bht2	character varying(32)		extended	
depth3	character varying(32)		extended	
bht3	character varying(32)		extended	
depth4	character varying(32)		extended	
bht4	character varying(32)		extended	
depth5	character varying(32)		extended	
bht5	character varying(32)		extended	
depth6	character varying(32)		extended	
bht6	character varying(32)		extended	
depth7	character varying(32)		extended	
bht7	character varying(32)		extended	
depth8	character varying(32)		extended	
bht8	character varying(32)		extended	
depth9	character varying(32)		extended	
bht9	character varying(32)		extended	
depth10	character varying(32)		extended	
bht10	character varying(32)		extended	
depth11	character varying(32)		extended	
bht11	character varying(32)		extended	
depth12	character varying(32)		extended	
bht12	character varying(32)		extended	
depth13	character varying(32)		extended	
bht13	character varying(32)		extended	

depth14	character varying(32)	extended
bht14	character varying(32)	extended
depth15	character varying(32)	extended
bht15	character varying(32)	extended
depth16	character varying(32)	extended
bht16	character varying(32)	extended
depth17	character varying(32)	extended
bht17	character varying(32)	extended
depth18	character varying(32)	extended
bht18	character varying(32)	extended
Has OIDs: no		

TTU Well Data (Part 3 of 3)

Table "staging.tt_well_data"

Column	Type	Modifiers	Storage	Description
objectid	numeric		main	
shape	character varying(128)		extended	
api	character varying(16)		extended	
name_suf	character varying(16)		extended	
well_name	character varying(16)		extended	
field_name	character varying(16)		extended	
operator	character varying(16)		extended	
lease_num	character varying(16)		extended	
spud_date	character varying(16)		extended	
rkb_elev	double precision		plain	
td_md	character varying(16)		extended	
s_northing	character varying(16)		extended	
tvd	character varying(16)		extended	
s_easting	character varying(16)		extended	
s_ns	character varying(16)		extended	
s_area	character varying(16)		extended	
s_ew	character varying(16)		extended	
b_northing	character varying(16)		extended	
s_blk	character varying(16)		extended	
b_easting	character varying(16)		extended	
b_ns	character varying(16)		extended	
b_area	character varying(16)		extended	
b_ew	character varying(16)		extended	
b_blk	character varying(16)		extended	
st_date	character varying(16)		extended	
td_date	character varying(16)		extended	
district	character varying(16)		extended	
type_code	character varying(16)		extended	
status	character varying(16)		extended	
s_long	numeric(10,7)		main	
w_depth	character varying(16)		extended	
b_long	numeric(10,7)		main	
s_lat	numeric(10,7)		main	
b_lat	numeric(10,7)		main	
complex_id	character varying(16)		extended	
s_lease_num	character varying(16)		extended	
struc_id	character varying(16)		extended	
order_status	character varying(32)		extended	
Has OIDs: no				

TTU Well Data (All 3 of 3)

View "staging.tt_well_view"

Column	Type	Modifiers	Storage	Description
id	uuid		plain	
latitude	numeric(10,7)		main	
longitude	numeric(10,7)		main	
api	character varying(16)		extended	
name_suf	character varying(16)		extended	
well_name	character varying(16)		extended	
field_name	character varying(16)		extended	
operator	character varying(16)		extended	
lease_num	character varying(16)		extended	
spud_date	character varying(16)		extended	
rkb_elev	double precision		plain	
td_md	character varying(16)		extended	
s_northing	character varying(16)		extended	
tvd	character varying(16)		extended	
s_easting	character varying(16)		extended	
s_ns	character varying(16)		extended	
s_area	character varying(16)		extended	
s_ew	character varying(16)		extended	
b_northing	character varying(16)		extended	
s_blk	character varying(16)		extended	
b_easting	character varying(16)		extended	
b_ns	character varying(16)		extended	
b_area	character varying(16)		extended	
b_ew	character varying(16)		extended	
b_blk	character varying(16)		extended	
st_date	character varying(16)		extended	
td_date	character varying(16)		extended	
district	character varying(16)		extended	
type_code	character varying(16)		extended	
status	character varying(16)		extended	
w_depth	character varying(16)		extended	
b_long	numeric(10,7)		main	
b_lat	numeric(10,7)		main	
complex_id	character varying(16)		extended	
s_lease_num	character varying(16)		extended	
struc_id	character varying(16)		extended	
shape	character varying(128)		extended	
order_status	character varying(32)		extended	
file_name	character varying(64)		extended	
lease_area	character varying(32)		extended	
addition	character varying(32)		extended	
block	character varying(32)		extended	
depth	character varying(32)		extended	
bht	character varying(32)		extended	
depth1	character varying(32)		extended	
bht1	character varying(32)		extended	
depth2	character varying(32)		extended	
bht2	character varying(32)		extended	
depth3	character varying(32)		extended	
bht3	character varying(32)		extended	

depth4	character varying(32)	extended
bht4	character varying(32)	extended
depth5	character varying(32)	extended
bht5	character varying(32)	extended
depth6	character varying(32)	extended
bht6	character varying(32)	extended
depth7	character varying(32)	extended
bht7	character varying(32)	extended
depth8	character varying(32)	extended
bht8	character varying(32)	extended
depth9	character varying(32)	extended
bht9	character varying(32)	extended
depth10	character varying(32)	extended
bht10	character varying(32)	extended
depth11	character varying(32)	extended
bht11	character varying(32)	extended
depth12	character varying(32)	extended
bht12	character varying(32)	extended
depth13	character varying(32)	extended
bht13	character varying(32)	extended
depth14	character varying(32)	extended
bht14	character varying(32)	extended
depth15	character varying(32)	extended
bht15	character varying(32)	extended
depth16	character varying(32)	extended
bht16	character varying(32)	extended
depth17	character varying(32)	extended
bht17	character varying(32)	extended
depth18	character varying(32)	extended
bht18	character varying(32)	extended
ssf_depth	double precision	plain
vrt	double precision	plain
ssf_depth_ft	double precision	plain
vrt_f	double precision	plain

View definition:

```

SELECT DISTINCT core.generate_uuid('TEXAS_TECH'::character varying, 'SURFACE_SITE'::character
varying, c.api) AS id, w.s_lat AS latitude, w.s_long AS longitude, w.api, w.name_suf, w.well_name,
w.field_name, w.operator, w.lease_num, w.spud_date, w.rkb_elev, w.td_md, w.s_northing, w.tvd, w.s_easting,
w.s_ns, w.s_area, w.s_ew, w.b_northing, w.s_blk, w.b_easting, w.b_ns, w.b_area, w.b_ew, w.b_blk, w.st_date,
w.td_date, w.district, w.type_code, w.status, w.w_depth, w.b_long, w.b_lat, w.complex_id, w.s_lease_num,
w.struc_id, w.shape, w.order_status, u.file_name, u.lease_area, u.addition, u.block, u.depth, u.bht, u.depth1,
u.bht1, u.depth2, u.bht2, u.depth3, u.bht3, u.depth4, u.bht4, u.depth5, u.bht5, u.depth6, u.bht6, u.depth7, u.bht7,
u.depth8, u.bht8, u.depth9, u.bht9, u.depth10, u.bht10, u.depth11, u.bht11, u.depth12, u.bht12, u.depth13, u.bht13,
u.depth14, u.bht14, u.depth15, u.bht15, u.depth16, u.bht16, u.depth17, u.bht17, u.depth18, u.bht18, c.ssf_depth,
c.vrt, c.ssf_depth_ft, c.vrt_f
FROM staging.tt_well_data w
LEFT JOIN staging.tt_uncorrected_bht u ON w.api::text = u.api::text
LEFT JOIN staging.tt_corrected_bht c ON w.api::text = c.api::text;

```

Table "staging.tt_well_view_materialized"

Column	Type	Modifiers	Storage	Description
id	uuid		plain	
latitude	numeric(10,7)		main	

longitude	numeric(10,7)	main
api	character varying(16)	extended
name_suf	character varying(16)	extended
well_name	character varying(16)	extended
field_name	character varying(16)	extended
operator	character varying(16)	extended
lease_num	character varying(16)	extended
spud_date	character varying(16)	extended
rkb_elev	double precision	plain
td_md	character varying(16)	extended
s_northing	character varying(16)	extended
tv_d	character varying(16)	extended
s_easting	character varying(16)	extended
s_ns	character varying(16)	extended
s_area	character varying(16)	extended
s_ew	character varying(16)	extended
b_northing	character varying(16)	extended
s_blk	character varying(16)	extended
b_easting	character varying(16)	extended
b_ns	character varying(16)	extended
b_area	character varying(16)	extended
b_ew	character varying(16)	extended
b_blk	character varying(16)	extended
st_date	character varying(16)	extended
td_date	character varying(16)	extended
district	character varying(16)	extended
type_code	character varying(16)	extended
status	character varying(16)	extended
w_depth	character varying(16)	extended
b_long	numeric(10,7)	main
b_lat	numeric(10,7)	main
complex_id	character varying(16)	extended
s_lease_num	character varying(16)	extended
struc_id	character varying(16)	extended
shape	character varying(128)	extended
order_status	character varying(32)	extended
file_name	character varying(64)	extended
lease_area	character varying(32)	extended
addition	character varying(32)	extended
block	character varying(32)	extended
depth	character varying(32)	extended
bht	character varying(32)	extended
depth1	character varying(32)	extended
bht1	character varying(32)	extended
depth2	character varying(32)	extended
bht2	character varying(32)	extended
depth3	character varying(32)	extended
bht3	character varying(32)	extended
depth4	character varying(32)	extended
bht4	character varying(32)	extended
depth5	character varying(32)	extended
bht5	character varying(32)	extended
depth6	character varying(32)	extended
bht6	character varying(32)	extended
depth7	character varying(32)	extended
bht7	character varying(32)	extended

depth8	character varying(32)	extended
bht8	character varying(32)	extended
depth9	character varying(32)	extended
bht9	character varying(32)	extended
depth10	character varying(32)	extended
bht10	character varying(32)	extended
depth11	character varying(32)	extended
bht11	character varying(32)	extended
depth12	character varying(32)	extended
bht12	character varying(32)	extended
depth13	character varying(32)	extended
bht13	character varying(32)	extended
depth14	character varying(32)	extended
bht14	character varying(32)	extended
depth15	character varying(32)	extended
bht15	character varying(32)	extended
depth16	character varying(32)	extended
bht16	character varying(32)	extended
depth17	character varying(32)	extended
bht17	character varying(32)	extended
depth18	character varying(32)	extended
bht18	character varying(32)	extended
ssf_depth	double precision	plain
vrt	double precision	plain
ssf_depth_ft	double precision	plain
vrt_f	double precision	plain
Has OIDs: no		

UND Radiogenic Heat Data

Table "staging.und_rad_data"

Column	Type	Modifiers	Storage	Description
observationuri	character varying(255)		extended	
observationname	character varying(255)		extended	
specimenlabel	character varying(255)		extended	
specimenuri	character varying(255)		extended	
geologicunitname	character varying(255)		extended	
featureofinteresturi	character varying(255)		extended	
lithologyterm	character varying(255)		extended	
rockname	character varying(255)		extended	
samplingfeatureuri	character varying(255)		extended	
samplingfeaturename	character varying(255)		extended	
samplecollectiondate	date		plain	
localityterms	character varying(255)		extended	
tectonicprovince	character varying(255)		extended	
county	character varying(255)		extended	
state	character varying(255)		extended	
plss_meridians	character varying(255)		extended	
township	character varying(255)		extended	
range	character varying(255)		extended	
section	character varying(255)		extended	
sectionpart	character varying(255)		extended	
latdegree	double precision		plain	
longdegree	double precision		plain	

srs	character varying(255)	extended
locationuncertaintystatement	character varying(255)	extended
elevationdatum_m	double precision	plain
sampledepth_m	double precision	plain
samplemeasurementdate	date	plain
sampletype	character varying(255)	extended
samplemass_kg	double precision	plain
overburdenthickness_m	double precision	plain
sampldensity_g_cc	double precision	plain
uncertaintydensity	character varying(255)	extended
laboratory	character varying(255)	extended
laboratoryuri	character varying(255)	extended
radioactivitymeasurementdevice	character varying(255)	extended
measurementprocedure	character varying(255)	extended
samplecureduration	character varying(255)	extended
measurementcount	character varying(255)	extended
uranium	double precision	plain
thorium	double precision	plain
potassium	double precision	plain
heatproduction_mwm3	double precision	plain
heatproductionuncertainty	character varying(255)	extended
notes	character varying(255)	extended
metadatauri	character varying(255)	extended
source	character varying(255)	extended
identifier	character varying(255)	extended
Has OIDs: no		

View "staging.und_rad_view"

Column	Type	Modifiers	Storage	Description
id	uuid		plain	
latitude	double precision		plain	
longitude	double precision		plain	
observationuri	character varying(255)		extended	
observationname	character varying(255)		extended	
specimenlabel	character varying(255)		extended	
specimenuri	character varying(255)		extended	
geologicunitname	character varying(255)		extended	
featureofinteresturi	character varying(255)		extended	
lithologyterm	character varying(255)		extended	
rockname	character varying(255)		extended	
samplingfeatureuri	character varying(255)		extended	
samplingfeaturename	character varying(255)		extended	
samplecollectiondate	date		plain	
localityterms	character varying(255)		extended	
tectonicprovince	character varying(255)		extended	
county	character varying(255)		extended	
state	character varying(255)		extended	
plss_meridians	character varying(255)		extended	
township	character varying(255)		extended	
range	character varying(255)		extended	
section	character varying(255)		extended	
sectionpart	character varying(255)		extended	
latdegree	double precision		plain	
longdegree	double precision		plain	
srs	character varying(255)		extended	

locationuncertaintystatement	character varying(255)	extended
elevationdatum_m	double precision	plain
sampledepth_m	double precision	plain
samplemeasurementdate	date	plain
sampletype	character varying(255)	extended
samplemass_kg	double precision	plain
overburdenthickness_m	double precision	plain
sampldensity_g_cc	double precision	plain
uncertaintydensity	character varying(255)	extended
laboratory	character varying(255)	extended
laboratoryuri	character varying(255)	extended
radioactivitymeasurementdevice	character varying(255)	extended
measurementprocedure	character varying(255)	extended
samplecureduration	character varying(255)	extended
measurementcount	character varying(255)	extended
uranium	double precision	plain
thorium	double precision	plain
potassium	double precision	plain
heatproduction_mwm3	double precision	plain
heatproductionuncertainty	character varying(255)	extended
notes	character varying(255)	extended
metadatauri	character varying(255)	extended
source	character varying(255)	extended
identifier	character varying(255)	extended

View definition:

```
SELECT core.generate_uuid('UND'::character varying, 'SURFACE_SITE'::character varying,
(to_char(COALESCE(rad.latdegree, tc.latdegreewgs84), '999.999'::text) || to_char(COALESCE(rad.longdegree,
tc.longdegreewgs84), '999.999'::text))::character varying) AS id, rad.latdegree AS latitude, rad.longdegree AS
longitude, rad.observationuri, rad.observationname, rad.specimenlabel, rad.specimenuri, rad.geologicunitname,
rad.featureofinteresturi, rad.lithologyterm, rad.rockname, rad.samplingfeatureuri, rad.samplingfeaturename,
rad.samplecollectiondate, rad.localityterms, rad.tectonicprovince, rad.county, rad.state, rad.plss_meridians,
rad.township, rad.range, rad.section, rad.sectionpart, rad.latdegree, rad.longdegree, rad.srs,
rad.locationuncertaintystatement, rad.elevationdatum_m, rad.sampledepth_m, rad.samplemeasurementdate,
rad.sampletype, rad.samplemass_kg, rad.overburdenthickness_m, rad.sampldensity_g_cc, rad.uncertaintydensity,
rad.laboratory, rad.laboratoryuri, rad.radioactivitymeasurementdevice, rad.measurementprocedure,
rad.samplecureduration, rad.measurementcount, rad.uranium, rad.thorium, rad.potassium,
rad.heatproduction_mwm3, rad.heatproductionuncertainty, rad.notes, rad.metadatauri, rad.source, rad.identifier
FROM staging.und_tc_data tc
FULL JOIN staging.und_rad_data rad ON tc.observationuri::text = rad.observationuri::text
WHERE rad.latdegree IS NOT NULL AND rad.longdegree IS NOT NULL;
```

Table "staging.und_rad_view_materialized"

Column	Type	Modifiers	Storage	Description
id	uuid		plain	
latitude	double precision		plain	
longitude	double precision		plain	
observationuri	character varying(255)		extended	
observationname	character varying(255)		extended	
specimenlabel	character varying(255)		extended	
specimenuri	character varying(255)		extended	
geologicunitname	character varying(255)		extended	
featureofinteresturi	character varying(255)		extended	

lithologyterm	character varying(255)	extended
rockname	character varying(255)	extended
samplingfeatureuri	character varying(255)	extended
samplingfeaturename	character varying(255)	extended
samplecollectiondate	date	plain
localityterms	character varying(255)	extended
tectonicprovince	character varying(255)	extended
county	character varying(255)	extended
state	character varying(255)	extended
plss_meridians	character varying(255)	extended
township	character varying(255)	extended
range	character varying(255)	extended
section	character varying(255)	extended
sectionpart	character varying(255)	extended
latdegree	double precision	plain
longdegree	double precision	plain
srs	character varying(255)	extended
locationuncertaintystatement	character varying(255)	extended
elevationdatum_m	double precision	plain
sampledepth_m	double precision	plain
samplemeasurementdate	date	plain
sampletype	character varying(255)	extended
samplemass_kg	double precision	plain
overburdenthickness_m	double precision	plain
sampldensity_g_cc	double precision	plain
uncertaintydensity	character varying(255)	extended
laboratory	character varying(255)	extended
laboratoryuri	character varying(255)	extended
radioactivitymeasurementdevice	character varying(255)	extended
measurementprocedure	character varying(255)	extended
samplecureduration	character varying(255)	extended
measurementcount	character varying(255)	extended
uranium	double precision	plain
thorium	double precision	plain
potassium	double precision	plain
heatproduction_mwm3	double precision	plain
heatproductionuncertainty	character varying(255)	extended
notes	character varying(255)	extended
metadatauri	character varying(255)	extended
source	character varying(255)	extended
identifier	character varying(255)	extended
Has OIDs: no		

UND Thermal Conductivity Data

Table "staging.und_tc_data"

Column	Type	Modifiers	Storage	Description
observationuri	character varying(255)		extended	
observationname	character varying(255)		extended	
label	character varying(255)		extended	
relatedsampleuri	character varying(255)		extended	
otherid	character varying(255)		extended	
sampledepth	double precision		plain	
measurementunits	character varying(32)		extended	
sitelocationuri	character varying(255)		extended	

sitelocationname	character varying(255)	extended
localityterms	character varying(255)	extended
county	character varying(32)	extended
state	character varying(32)	extended
latdegreewgs84	double precision	plain
longdegreewgs84	double precision	plain
srs	character varying(32)	extended
locationuncertaintystatement	character varying(255)	extended
samplecollectiondate	date	plain
samplemeasurementdate	date	plain
sampletype	character varying(32)	extended
sampleshape	character varying(32)	extended
sampledimensionsheight	double precision	plain
sampledimensionswidth	double precision	plain
sampledimensionslength	double precision	plain
sampledimensionsunits	character varying(32)	extended
sampledensity	double precision	plain
uncertaintydensity	character varying(32)	extended
sampledensityunits	character varying(32)	extended
laboratory	character varying(255)	extended
measurementdevice	character varying(255)	extended
measurementprocedure	character varying(255)	extended
measurementpressure	double precision	plain
measurementpressureunits	character varying(255)	extended
sampletemperature	double precision	plain
sampletemperatureunits	character varying(255)	extended
measurementduration	double precision	plain
measurementdirection	character varying(255)	extended
measurementrun	integer	plain
bulkmixingformula	character varying(255)	extended
saturationfluid	character varying(255)	extended
saturatedsampleconductivity	double precision	plain
drysampleconductivity	double precision	plain
averagethermalconductivity	double precision	plain
uncertaintytc	character varying(32)	extended
correctiontc	double precision	plain
correctiondescription	character varying(255)	extended
unitstc	character varying(32)	extended
specificheat	double precision	plain
uncertaintyspecificheat	double precision	plain
specificheatunits	character varying(32)	extended
thermaldiffusivity	double precision	plain
uncertaintythermaldiffusivity	double precision	plain
unitsthermaldiffusivity	character varying(32)	extended
geologicformationname	character varying(255)	extended
geologicformationuri	character varying(255)	extended
geologicformationtype	character varying(255)	extended
datumelevation	date	plain
elevation	double precision	plain
permeability	double precision	plain
unitspermeability	character varying(32)	extended
porosity	double precision	plain
notes	character varying(255)	extended
metadatauri	character varying(255)	extended
source	character varying(255)	extended
identifier	character varying(255)	extended

Has OIDs: no

View "staging.und_tc_view"

Column	Type	Modifiers	Storage	Description
id	uuid		plain	
latitude	double precision		plain	
longitude	double precision		plain	
observationuri	character varying(255)		extended	
observationname	character varying(255)		extended	
label	character varying(255)		extended	
relatedsampleuri	character varying(255)		extended	
otherid	character varying(255)		extended	
sampledepth	double precision		plain	
measurementunits	character varying(32)		extended	
sitelocationuri	character varying(255)		extended	
sitelocationname	character varying(255)		extended	
localityterms	character varying(255)		extended	
county	character varying(32)		extended	
state	character varying(32)		extended	
latdegreewgs84	double precision		plain	
longdegreewgs84	double precision		plain	
srs	character varying(32)		extended	
locationuncertaintystatement	character varying(255)		extended	
samplecollectiondate	date		plain	
samplemeasurementdate	date		plain	
sampletype	character varying(32)		extended	
sampleshape	character varying(32)		extended	
sampldimensionsheight	double precision		plain	
sampldimensionswidth	double precision		plain	
sampldimensionslength	double precision		plain	
sampldimensionsunits	character varying(32)		extended	
sampldensity	double precision		plain	
uncertaintydensity	character varying(32)		extended	
sampldensityunits	character varying(32)		extended	
laboratory	character varying(255)		extended	
measurementdevice	character varying(255)		extended	
measurementprocedure	character varying(255)		extended	
measurementpressure	double precision		plain	
measurementpressureunits	character varying(255)		extended	
sampletemperature	double precision		plain	
sampletemperatureunits	character varying(255)		extended	
measurementduration	double precision		plain	
measurementdirection	character varying(255)		extended	
measurementrun	integer		plain	
bulkmixingformula	character varying(255)		extended	
saturationfluid	character varying(255)		extended	
saturatedsampleconductivity	double precision		plain	
drysampleconductivity	double precision		plain	
averagethermalconductivity	double precision		plain	
uncertaintytc	character varying(32)		extended	
correctiontc	double precision		plain	
correctiondescription	character varying(255)		extended	
unitstc	character varying(32)		extended	
specificheat	double precision		plain	
uncertaintyspecificheat	double precision		plain	

specificheatunits	character varying(32)	extended
thermaldiffusivity	double precision	plain
uncertaintythermaldiffusivity	double precision	plain
unitsthermaldiffusivity	character varying(32)	extended
geologicformationname	character varying(255)	extended
geologicformationuri	character varying(255)	extended
geologicformationtype	character varying(255)	extended
datumelevation	date	plain
elevation	double precision	plain
permeability	double precision	plain
unitspermeability	character varying(32)	extended
porosity	double precision	plain
notes	character varying(255)	extended
metadatauri	character varying(255)	extended
source	character varying(255)	extended
identifier	character varying(255)	extended

View definition:

```

SELECT core.generate_uuid('UND'::character varying, 'SURFACE_SITE'::character varying,
(to_char(COALESCE(rad.latdegree, tc.latdegrewgs84), '999.999'::text) || to_char(COALESCE(rad.longdegree,
tc.longdegrewgs84), '999.999'::text))::character varying) AS id, tc.latdegrewgs84 AS latitude,
tc.longdegrewgs84 AS longitude, tc.observationuri, tc.observationname, tc.label, tc.relatedsampleuri, tc.otherid,
tc.sampledepth, tc.measurementunits, tc.sitelocationuri, tc.sitelocationname, tc.localityterms, tc.county, tc.state,
tc.latdegrewgs84, tc.longdegrewgs84, tc.srs, tc.locationuncertaintystatement, tc.samplecollectiondate,
tc.samplemeasurementdate, tc.sampletype, tc.sampleshape, tc.sampledimensionsheight, tc.sampledimensionswidth,
tc.sampledimensionslength, tc.sampledimensionsunits, tc.sampledensity, tc.uncertaintydensity,
tc.sampledensityunits, tc.laboratory, tc.measurementdevice, tc.measurementprocedure, tc.measurementpressure,
tc.measurementpressureunits, tc.sampletemperature, tc.sampletemperatureunits, tc.measurementduration,
tc.measurementdirection, tc.measurementrun, tc.bulkmixingformula, tc.saturationfluid,
tc.saturatedsampleconductivity, tc.drysampleconductivity, tc.averagethermalconductivity, tc.uncertaintytc,
tc.correctiontc, tc.correctiondescription, tc.unitstc, tc.specificheat, tc.uncertaintyspecificheat, tc.specificheatunits,
tc.thermaldiffusivity, tc.uncertaintythermaldiffusivity, tc.unitsthermaldiffusivity, tc.geologicformationname,
tc.geologicformationuri, tc.geologicformationtype, tc.datumelevation, tc.elevation, tc.permeability,
tc.unitspermeability, tc.porosity, tc.notes, tc.metadatauri, tc.source, tc.identifier
FROM staging.und_tc_data tc
FULL JOIN staging.und_rad_data rad ON tc.observationuri::text = rad.observationuri::text
WHERE tc.longdegrewgs84 IS NOT NULL AND tc.latdegrewgs84 IS NOT NULL;

```

Table "staging.und_tc_view_materialized"

Column	Type	Modifiers	Storage	Description
id	uuid		plain	
latitude	double precision		plain	
longitude	double precision		plain	
observationuri	character varying(255)		extended	
observationname	character varying(255)		extended	
label	character varying(255)		extended	
relatedsampleuri	character varying(255)		extended	
otherid	character varying(255)		extended	
sampledepth	double precision		plain	
measurementunits	character varying(32)		extended	
sitelocationuri	character varying(255)		extended	
sitelocationname	character varying(255)		extended	
localityterms	character varying(255)		extended	

county	character varying(32)	extended
state	character varying(32)	extended
latdegrewgs84	double precision	plain
longdegrewgs84	double precision	plain
srs	character varying(32)	extended
locationuncertaintystatement	character varying(255)	extended
samplecollectiondate	date	plain
samplemeasurementdate	date	plain
sampletype	character varying(32)	extended
sampleshape	character varying(32)	extended
sampledimensionsheight	double precision	plain
sampledimensionswidth	double precision	plain
sampledimensionslength	double precision	plain
sampledimensionsunits	character varying(32)	extended
sampledensity	double precision	plain
uncertaintydensity	character varying(32)	extended
sampledensityunits	character varying(32)	extended
laboratory	character varying(255)	extended
measurementdevice	character varying(255)	extended
measurementprocedure	character varying(255)	extended
measurementpressure	double precision	plain
measurementpressureunits	character varying(255)	extended
sampletemperature	double precision	plain
sampletemperatureunits	character varying(255)	extended
measurementduration	double precision	plain
measurementdirection	character varying(255)	extended
measurementrun	integer	plain
bulkmixingformula	character varying(255)	extended
saturationfluid	character varying(255)	extended
saturatedsampleconductivity	double precision	plain
drysampleconductivity	double precision	plain
averagethermalconductivity	double precision	plain
uncertaintytc	character varying(32)	extended
correctiontc	double precision	plain
correctiondescription	character varying(255)	extended
unitstc	character varying(32)	extended
specifichat	double precision	plain
uncertaintyspecifichat	double precision	plain
specifichatunits	character varying(32)	extended
thermaldiffusivity	double precision	plain
uncertaintythermaldiffusivity	double precision	plain
unitsthermaldiffusivity	character varying(32)	extended
geologicformationname	character varying(255)	extended
geologicformationuri	character varying(255)	extended
geologicformationtype	character varying(255)	extended
datumelevation	date	plain
elevation	double precision	plain
permeability	double precision	plain
unitspermeability	character varying(32)	extended
porosity	double precision	plain
notes	character varying(255)	extended
metadatauri	character varying(255)	extended
source	character varying(255)	extended
identifier	character varying(255)	extended
Has OIDs: no		

UND Temperature Depth Data

Table "staging.und_td_curve_points"

Column	Type	Modifiers	Storage	Description
curve_point_id	integer	not null default nextval('staging.und_td_curve_points_curve_point_id_seq'::regclass)	plain	
depth	double precision		plain	
temperature	double precision		plain	
comments_info	text		extended	
sheet_name	character varying(64)		extended	
location_name	character varying(64)		extended	
hole_number	character varying(64)		extended	
operator_name	character varying(64)		extended	
log_date	timestamp without time zone		plain	
latitude	double precision		plain	
longitude	double precision		plain	
trs	character varying(128)		extended	
elevation_per_m	double precision		plain	
spud_date	timestamp without time zone		plain	
quadrangle_map	character varying(128)		extended	
map_1_250k	character varying(128)		extended	
probe_cable_bridge	character varying(128)		extended	
identifier	character varying(100)		extended	

Indexes:

"und_td_curve_points_pkey" PRIMARY KEY, btree (curve_point_id)

Has OIDs: no

Sequence "staging.und_td_curve_points_curve_point_id_seq"

Column	Type	Value	Storage	Description
sequence_name	name	und_td_curve_points_curve_point_id_seq	plain	
last_value	bigint	11857	plain	
start_value	bigint	1	plain	
increment_by	bigint	1	plain	
max_value	bigint	9223372036854775807	plain	
min_value	bigint	1	plain	
cache_value	bigint	1	plain	
log_cnt	bigint	28	plain	
is_cycled	boolean	f	plain	
is_called	boolean	t	plain	

Index "staging.und_td_curve_points_pkey"

Column	Type	Storage	Description
curve_point_id	integer	plain	
primary key, btree, for table "staging.und_td_curve_points"			

Table "staging.und_td_sites"

Column	Type	Modifiers	Storage	Description

sites_id	integer	not null default nextval('staging.und_td_sites _sites_id_seq'::regclass)	plain
identifier	character varying(100)	not null	extended
location_name	character varying(64)		extended
hole_number	character varying(64)		extended
label_description	character varying(64)		extended
latitude	double precision		plain
longitude	double precision		plain
srs	character varying(64)		extended
bht_c	double precision		plain
depth_per_m	double precision		plain
county	character varying(64)		extended
state	character varying(32)		extended
trs	character varying(128)		extended
elevation_per_m	double precision		plain
company_owner	character varying(256)		extended
drill_depth_m	double precision		plain
log_date	timestamp without time zone		plain
map	character varying(128)		extended
quadrangle_map	character varying(128)		extended
probe_cable_brid ge	character varying(128)		extended
operator_name	character varying(64)		extended

Indexes:
"und_td_sites_pkey" PRIMARY KEY, btree (sites_id)
Has OIDs: no

Index "staging.und_td_sites_pkey"

Column	Type	Storage	Description
sites_id	integer	plain	
primary key, btree, for table "staging.und_td_sites"			

Sequence "staging.und_td_sites_sites_id_seq"

Column	Type	Value	Storage	Description
sequence_name	name	und_td_sites_sites_id_seq	plain	
last_value	bigint	32	plain	
start_value	bigint	1	plain	
increment_by	bigint	1	plain	
max_value	bigint	9223372036854775807	plain	
min_value	bigint	1	plain	
cache_value	bigint	1	plain	
log_cnt	bigint	32	plain	
is_cycled	boolean	f	plain	
is_called	boolean	t	plain	

View "staging.und_td_view"

Column	Type	Modifiers	Storage	Description
id	uuid		plain	
latitude	double precision		plain	
longitude	double precision		plain	
sites_id	integer		plain	

identifier	character varying(100)	extended
sitelocationname	character varying(64)	extended
siteholenumber	character varying(64)	extended
label_description	character varying(64)	extended
tdslatitude	double precision	plain
tdslongitude	double precision	plain
tdssrs	character varying(64)	extended
bht_c	double precision	plain
depth_per_m	double precision	plain
tdscounty	character varying(64)	extended
tdsstate	character varying(32)	extended
trs	character varying(128)	extended
company_owner	character varying(256)	extended
siteelevation	double precision	plain
drill_depth_m	double precision	plain
sitelogdate	timestamp without time zone	plain
map	character varying(128)	extended
siteqmap	character varying(128)	extended
siteprobecablebridge	character varying(128)	extended
siteoperatorname	character varying(64)	extended
sheet_name	character varying(64)	extended
curvelocationname	character varying(64)	extended
curveholenumber	character varying(64)	extended
curveoperatorname	character varying(64)	extended
curvelogdate	timestamp without time zone	plain
tdclatitude	double precision	plain
tdclongitude	double precision	plain
tdcsrs	character varying(128)	extended
curveelevation	double precision	plain
spud_date	timestamp without time zone	plain
curveqmap	character varying(128)	extended
map_1_250k	character varying(128)	extended
curveprobecablebridge	character varying(128)	extended
depth	double precision	plain
temperature	double precision	plain
comments_info	text	extended

View definition:

```

SELECT core.generate_uuid('UND'::character varying, 'SURFACE_SITE'::character varying,
(to_char(tds.latitude, '999.999'::text) || to_char(tds.longitude, '999.999'::text))::character varying) AS id,
tds.latitude, tds.longitude, tds.sites_id, tds.identifier, tds.location_name AS sitelocationname, tds.hole_number AS
siteholenumber, tds.label_description, tds.latitude AS tdslatitude, tds.longitude AS tdslongitude, tds.srs AS tdssrs,
tds.bht_c, tds.depth_per_m, tds.county AS tdscounty, tds.state AS tdsstate, tds.trs, tds.company_owner,
tds.elevation_per_m AS siteelevation, tds.drill_depth_m, tds.log_date AS sitelogdate, tds.map,
tds.quadrangle_map AS siteqmap, tds.probe_cable_bridge AS siteprobecablebridge, tds.operator_name AS
siteoperatorname, tdc.sheet_name, tdc.location_name AS curvelocationname, tdc.hole_number AS
curveholenumber, tdc.operator_name AS curveoperatorname, tdc.log_date AS curvelogdate, tdc.latitude AS
tdclatitude, tdc.longitude AS tdclongitude, tdc.trs AS tdc_srs, tdc.elevation_per_m AS curveelevation,
tdc.spud_date, tdc.quadrangle_map AS curveqmap, tdc.map_1_250k, tdc.probe_cable_bridge AS
curveprobecablebridge, tdc.depth, tdc.temperature, tdc.comments_info
FROM staging.und_td_sites tds
LEFT JOIN staging.und_td_curve_points tdc ON tds.location_name::text = tdc.location_name::text OR
tds.hole_number::text = tdc.hole_number::text OR tds.label_description::text = tdc.sheet_name::text
WHERE tds.sites_id IS NOT NULL AND tdc.depth IS NOT NULL AND tdc.temperature IS NOT NULL;
```

Table "staging.und_td_view_materialized"

Column	Type	Modifiers	Storage	Description
id	uuid		plain	
latitude	double precision		plain	
longitude	double precision		plain	
sites_id	integer		plain	
identifier	character varying(100)		extended	
sitelocationname	character varying(64)		extended	
siteholenumber	character varying(64)		extended	
label_description	character varying(64)		extended	
tdslatitude	double precision		plain	
tdslongitude	double precision		plain	
tdssrs	character varying(64)		extended	
bht_c	double precision		plain	
depth_per_m	double precision		plain	
tdscounty	character varying(64)		extended	
tdsstate	character varying(32)		extended	
trs	character varying(128)		extended	
company_owner	character varying(256)		extended	
siteelevation	double precision		plain	
drill_depth_m	double precision		plain	
siteogdate	timestamp without time zone		plain	
map	character varying(128)		extended	
siteqmap	character varying(128)		extended	
siteprobecablebridge	character varying(128)		extended	
siteoperatorname	character varying(64)		extended	
sheet_name	character varying(64)		extended	
curvelocationname	character varying(64)		extended	
curveholenumber	character varying(64)		extended	
curveoperatorname	character varying(64)		extended	
curvelogdate	timestamp without time zone		plain	
tdclatitude	double precision		plain	
tdclongitude	double precision		plain	
tdcsrs	character varying(128)		extended	
curveelevation	double precision		plain	
spud_date	timestamp without time zone		plain	
curveqmap	character varying(128)		extended	
map_1_250k	character varying(128)		extended	
curveprobecablebridge	character varying(128)		extended	
depth	double precision		plain	
temperature	double precision		plain	
comments_info	text		extended	
Has OIDs: no				

Well Log Headers

Sequence "staging.well_log_headers_id_seq"

Column	Type	Value	Storage	Description
sequence_name	name	well_log_headers_id_seq	plain	
last_value	bigint	4978	plain	
start_value	bigint	4978	plain	
increment_by	bigint	1	plain	
max_value	bigint	9223372036854775807	plain	
min_value	bigint	1	plain	
cache_value	bigint	1	plain	

log_cnt	bigint	1	plain
is_cycled	boolean	f	plain
is_called	boolean	f	plain

Sequence "staging.well_log_headers_mapping_id_seq"

Column	Type	Value	Storage	Description
sequence_name	name	well_log_headers_mapping_id_seq	plain	
last_value	bigint	4978	plain	
start_value	bigint	4978	plain	
increment_by	bigint	1	plain	
max_value	bigint	9223372036854775807	plain	
min_value	bigint	1	plain	
cache_value	bigint	1	plain	
log_cnt	bigint	1	plain	
is_cycled	boolean	f	plain	
is_called	boolean	f	plain	

Tables and Fields in the Core Database

Pulled on April 7, 2014 using the following command:

```
psql -h sgtl1d1 -U postgres gtda -c '\d+ core.*' > gtda-core-tables.txt 2> /dev/null
```

Required signing onto sgtl1p1 with PuTTY. Also required the password for username 'postgres' on sgtl1d1.

Core Tables and Fields Contents

Tables and Fields in the Core Database	263
Aquifer	269
Table "core.aquifier"	269
Sequence "core.aquifier_id_seq"	269
Index "core.aquifier_pkey"	269
BHT Measurements	269
Table "core.bht_measurements"	269
View "core.bht_measurements_max_temp"	270
Index "core.bht_pkey"	270
Index "core.bht_site_idx"	270
Casing	270
Table "core.casing"	270
Sequence "core.casing_id_seq"	271
Index "core.casing_pkey"	271
County	271
Index "core.county_id_idx"	271
Index "core.county_pkey"	271
Index "core.county_state_idx"	271
Dataset ID	271
Index "core.dataset_id_idx"	271
Dataset Job	272
Table "core.dataset_job_data"	272
Sequence "core.dataset_job_data_dataset_job_data_id_seq"	272
Index "core.dataset_job_data_pkey"	272

County.....	272
Index "core.dataset_surface_county_id_idx"	272
Datasets	273
Table "core.datasets".....	273
Table "core.datasets_map"	273
Sequence "core.datasets_map_datasets_map_id_seq"	273
Index "core.datasets_map_pkey".....	274
Index "core.datasets_pkey"	274
Depth (& Mudweight Viscosity).....	274
Table "core.depth".....	274
Sequence "core.depth_id_seq"	274
View "core.depth_max_temp"	275
Index "core.depth_pkey".....	275
Index "core.depth_site_idx"	275
Field Report Mapping ID	275
Sequence "core.field_report_mapping_id_seq"	275
File Upload	275
Table "core.file_upload"	275
Sequence "core.file_upload_file_upload_id_seq"	276
Index "core.file_upload_pkey"	276
Filter Condition	276
Table "core.filter_condition"	276
Fluid.....	276
Table "core.fluid"	276
Table "core.fluid_h10"	277
Sequence "core.fluid_h10_id_seq"	277
Index "core.fluid_h10_pkey"	278
Table "core.fluid_h10h".....	278
Sequence "core.fluid_h10h_id_seq"	278
Index "core.fluid_h10h_pkey"	278
Index "core.fluid_pkey"	278
Formation.....	279

Table "core.formation"	279
Sequence "core.formation_id_seq"	279
Index "core.formation_pkey"	279
Heat Flow	279
Table "core.heat_flow"	279
Sequence "core.heat_flow_id_seq"	280
Heat Flow Publication Link Ids	280
Sequence "core.heat_flow_publication_links_id_seq"	280
Radiogenic Heat Generation IDs	281
Sequence "core.heat_generation_id_seq"	281
Heat Flow	281
Index "core.heatflow_pkey"	281
Ref_State	281
Index "core.id_idx"	281
Intervals	281
Table "core.interval"	281
Sequence "core.interval_id_seq"	282
Index "core.interval_pkey"	282
Lithology	282
Sequence "core.lithology_information_id_seq"	282
Materialized Views	282
Table "core.matviews"	282
Index "core.matviews_pkey"	283
Table "core.ngds_heat_flow_mview"	283
Heat Flow Notes	283
Table "core.notes"	283
Index "core.notes_pkey"	283
Precision TD Logs	284
Table "core.precision_log"	284
View "core.precision_log_max_temp"	284
Index "core.precision_log_pkey"	284
Index "core.precision_log_site_idx"	285

Publications.....	285
Table "core.publication"	285
Publication Metadata	285
View "core.publication_metadata"	285
Index "core.publication_pkey"	286
Sequence "core.publication_publication_id_seq"	286
Quality Issue Log ID.....	287
Sequence "core.quality_issue_log_id_seq"	287
Radioactivity.....	287
Table "core.radioactivity"	287
Index "core.radioactivity_pkey"	287
County.....	287
Table "core.ref_county"	287
Elevation	288
View "core.ref_county_elev"	288
County.....	288
Index "core.ref_county_geom_gist"	288
Index "core.ref_county_shape"	288
Data Source.....	289
Table "core.ref_data_source"	289
Index "core.ref_data_source_pkey"	289
Quality Issue Type	289
Table "core.ref_quality_issue_type"	289
View "core.ref_quality_issue_type_h"	289
Index "core.ref_quality_issue_type_pkey"	290
State	290
Table "core.ref_state"	290
Index "core.ref_state_shape"	290
Table "core.ref_symbol"	290
Index "core.ref_symbol_pkey"	291
Geologic Reservoir	291
Table "core.reservoir"	291

Index "core.reservoir_pkey"	292
Shape	292
Index "core.shape_gix"	292
Site Comments.....	292
Sequence "core.site_comments_id_seq"	292
Publication Links & Related Resources	293
Sequence "core.site_publication_links_id_seq"	293
Table "core.site_related_resources"	293
Sequence "core.site_related_resources_id_seq"	293
Index "core.site_related_resources_pkey"	294
Sequence "core.site_related_resources_record_id_seq"	294
Aggregated Well Data Tier 2 (Also called Surface Site County State).....	294
Index "core.state_pkey"	294
Table "core.surface_site"	294
Table "core.surface_site_county_state_materialized_view"	295
View "core.surface_site_county_state_view"	296
Index "core.surface_site_pkey"	296
Temperature Series (TD Curves)	297
Table "core.temperature_series"	297
Sequence "core.temperature_series_id_seq"	297
Index "core.temperature_series_pkey"	297
Borehole Temperature Observation Content Model	297
View "core.template_borehole"	297
Table "core.template_borehole_materialized"	300
Fluid Flux Injection Disposal Content Model	301
View "core.template_fluidflux"	301
Table "core.template_fluidflux_materialized"	303
Heat Flow Observation Content Model	303
View "core.template_heatflow"	303
Table "core.template_heatflow_materialized"	306
Radiogenic Heat Production Content Model	308
View "core.template_radiogenicheat"	308

Table "core.template_radiogenicheat_materialized"	309
Geologic Reservoir Content Model.....	310
View "core.template_reservoir"	310
Table "core.template_reservoir_materialized"	312
Thermal Conductivity Observation Content Model.....	313
View "core.template_thermalconductivity"	313
Table "core.template_thermalconductivity_materialized"	315
Well Fluid Production Observation Content Model	316
View "core.template_wellfluid"	316
Table "core.template_wellfluid_materialized"	318
Well Header Observation Content Model	318
View "core.template_wellheader"	318
Table "core.template_wellheader_materialized"	321
Well Log Observation Content Model	322
View "core.template_welllog"	322
Table "core.template_welllog_materialized"	323
Well Test Observation Content Model	324
View "core.template_welltest"	324
Table "core.template_welltest_materialized"	325
Temperature Series Mapping ID	326
Sequence "core.temprature_series_mapping_id_seq"	326
Thermal Conductivity.....	326
Table "core.thermal_conductivity"	326
Index "core.thermal_conductivity_pkey"	327
Well Dates	327
Table "core.well_dates"	327
Sequence "core.well_dates_id_seq"	327
Index "core.well_dates_pkey"	328
Well Info.....	328
Table "core.well_info"	328
Well Info ID	328
Sequence "core.well_info_id_seq"	328

Index "core.well_info_pkey"	328
Well Test	329
Table "core.well_test"	329
Sequence "core.well_test_id_seq"	329
Index "core.well_test_pkey"	329

Aquifer

Table "core.aquifier"

Column	Type	Modifiers	Storage	Description
id	bigint	not null default nextval('core.aquifier_id_seq'::regclass)	plain	
surface_site_id	uuid		plain	
dataset_id	uuid		plain	
water_table_depth	double precision		plain	
aquifier_temperature	double precision		plain	
aquifier_temperature_se	double precision		plain	

Indexes:

"aquifier_pkey" PRIMARY KEY, btree (id)

Has OIDs: no

Sequence "core.aquifier_id_seq"

Column	Type	Value	Storage	Description
sequence_name	name	aquifier_id_seq	plain	
last_value	bigint	59576	plain	
start_value	bigint	1	plain	
increment_by	bigint	1	plain	
max_value	bigint	9223372036854775807	plain	
min_value	bigint	1	plain	
cache_value	bigint	1	plain	
log_cnt	bigint	19	plain	
is_cycled	boolean	f	plain	
is_called	boolean	t	plain	

Index "core.aquifier_pkey"

Column	Type	Storage	Description
id	bigint	plain	
primary key, btree, for table "core.aquifier"			

BHT Measurements

Table "core.bht_measurements"

Column	Type	Modifiers	Storage	Description
id	uuid	not null	plain	
surface_site_id	uuid		plain	

bht	double precision		plain
bht_corrected	double precision		plain
bht_correction_type_id	integer		plain
bht_depth	double precision		plain
bht_timestamp	timestamp without time zone		plain
drill_date	date		plain
time_since_circulation	double precision		plain
estimated	boolean	default false	plain
dataset_id	uuid		plain
run_number	integer		plain
bht_correction_type	character varying(16)		extended
date_type	character varying(16)		extended

Indexes:

"bht_pkey" PRIMARY KEY, btree (id)

"bht_site_idx" btree (surface_site_id)

Has OIDs: no

View "core.bht_measurements_max_temp"

Column	Type	Modifiers	Storage	Description
surface_site_id	uuid		plain	
temperature	double precision		plain	

View definition:

```
SELECT bht_measurements.surface_site_id, max(bht_measurements.bht) AS temperature
FROM core.bht_measurements
GROUP BY bht_measurements.surface_site_id;
```

Index "core.bht_pkey"

Column	Type	Storage	Description
id	uuid	plain	

primary key, btree, for table "core.bht_measurements"

Index "core.bht_site_idx"

Column	Type	Storage	Description
surface_site_id	uuid	plain	

btree, for table "core.bht_measurements"

Casing

Table "core.casing"

Column	Type	Modifiers	Storage	Description
id	bigint	not null default nextval('core.casing_id_seq'::regclass)	plain	
surface_site_id	uuid		plain	
dataset_id	uuid		plain	
casing_size_inches	double precision		plain	
casing_depth	double precision		plain	
liner_inches	double precision		plain	
liner_top	double precision		plain	
liner_base	double precision		plain	
choke_mm	double precision		plain	

casing_weight	double precision	plain
casing_top	character varying(7)	extended
casing_left	double precision	plain
file_date	date	plain

Indexes:

"casing_pkey" PRIMARY KEY, btree (id)

Has OIDs: no

Sequence "core.casing_id_seq"

Column	Type	Value	Storage	Description
sequence_name	name	casing_id_seq	plain	
last_value	bigint	11350288	plain	
start_value	bigint	1	plain	
increment_by	bigint	1	plain	
max_value	bigint	9223372036854775807	plain	
min_value	bigint	1	plain	
cache_value	bigint	1	plain	
log_cnt	bigint	7	plain	
is_cycled	boolean	f	plain	
is_called	boolean	t	plain	

Index "core.casing_pkey"

Column	Type	Storage	Description
id	bigint	plain	

primary key, btree, for table "core.casing"

County

Index "core.county_id_idx"

Column	Type	Storage	Description
id	integer	plain	

btree, for table "core.ref_county"

Index "core.county_pkey"

Column	Type	Storage	Description
id	integer	plain	

primary key, btree, for table "core.ref_county"

Index "core.county_state_idx"

Column	Type	Storage	Description
state_id	integer	plain	

btree, for table "core.ref_county"

Dataset ID

Index "core.dataset_id_idx"

Column	Type	Storage	Description
id	uuid	plain	

unique, btree, for table "core.surface_site"

Dataset Job

Table "core.dataset_job_data"

Column	Type	Modifiers	Storage	Description
dataset_job_data_id	integer	not null default nextval('core.dataset_job_data_data aset_job_data_id_seq'::regclass)	plain	
dataset_id	uuid	not null	plain	
dataset_name	character varying(100)	not null	extended	
job_name	character varying(100)	not null	extended	
display_name	character varying(100)	not null	extended	
file_name	character varying(100)	not null	extended	
file_format	character varying(100)	not null	extended	
required	boolean	default false	plain	
type	character varying(50)		extended	

Indexes:

"dataset_job_data_pkey" PRIMARY KEY, btree (dataset_job_data_id)

Foreign-key constraints:

"dataset_id" FOREIGN KEY (dataset_id) REFERENCES core.datasets(id) MATCH FULL

Has OIDs: no

Sequence "core.dataset_job_data_dataset_job_data_id_seq"

Column	Type	Value	Storage	Description
sequence_name	name	dataset_job_data_dataset_job_data_id_seq	plain	
last_value	bigint	14	plain	
start_value	bigint	1	plain	
increment_by	bigint	1	plain	
max_value	bigint	9223372036854775807	plain	
min_value	bigint	1	plain	
cache_value	bigint	1	plain	
log_cnt	bigint	0	plain	
is_cycled	boolean	f	plain	
is_called	boolean	t	plain	

Index "core.dataset_job_data_pkey"

Column	Type	Storage	Description
dataset_job_data_id	integer	plain	
primary key, btree, for table "core.dataset_job_data"			

County

Index "core.dataset_surface_county_id_idx"

Column	Type	Storage	Description
county_id	integer	plain	
btree, for table "core.surface_site"			

Datasets

Table "core.datasets"

Column	Type	Modifiers	Storage	Description
id	uuid	not null	plain	
datasource	character varying(16)	not null	extended	
datatype	character varying(16)	not null	extended	
resource_id	character varying(32)		extended	

Indexes:

"datasets_pkey" PRIMARY KEY, btree (id)

Referenced by:

TABLE "core.datasets_map" CONSTRAINT "dataset_id" FOREIGN KEY (dataset_id) REFERENCES core.datasets(id) MATCH FULL

TABLE "core.dataset_job_data" CONSTRAINT "dataset_id" FOREIGN KEY (dataset_id) REFERENCES core.datasets(id) MATCH FULL

TABLE "core.fluid" CONSTRAINT "fluid_dataset" FOREIGN KEY (dataset_id) REFERENCES core.datasets(id) MATCH FULL

TABLE "core.fluid_h10" CONSTRAINT "fluid_h10_dataset" FOREIGN KEY (dataset_id) REFERENCES core.datasets(id) MATCH FULL

TABLE "core.fluid_h10h" CONSTRAINT "fluid_h10h_dataset" FOREIGN KEY (dataset_id) REFERENCES core.datasets(id) MATCH FULL

TABLE "core.well_dates" CONSTRAINT "well_dates_dataset" FOREIGN KEY (dataset_id) REFERENCES core.datasets(id) MATCH FULL

TABLE "core.well_info" CONSTRAINT "well_info_dataset" FOREIGN KEY (dataset_id) REFERENCES core.datasets(id) MATCH FULL

TABLE "core.well_test" CONSTRAINT "well_test_dataset" FOREIGN KEY (dataset_id) REFERENCES core.datasets(id) MATCH FULL

Has OIDs: no

Table "core.datasets_map"

Column	Type	Modifiers	Storage	Description
datasets_map_id	integer	not null default nextval('core.datasets_map_datasets_map_id_seq'::regclass)	plain	
dataset_id	uuid	not null	plain	
table_names	character varying(256)	not null	extended	

Indexes:

"datasets_map_pkey" PRIMARY KEY, btree (datasets_map_id)

Foreign-key constraints:

"dataset_id" FOREIGN KEY (dataset_id) REFERENCES core.datasets(id) MATCH FULL

Has OIDs: no

Sequence "core.datasets_map_datasets_map_id_seq"

Column	Type	Value	Storage	Description
sequence_name	name		plain	
		datasets_map_datasets_map_id_seq		
last_value	bigint	17	plain	
start_value	bigint	1	plain	
increment_by	bigint	1	plain	

max_value	bigint	9223372036854775807	plain
min_value	bigint	1	plain
cache_value	bigint	1	plain
log_cnt	bigint	0	plain
is_cycled	boolean	f	plain
is_called	boolean	t	plain

Index "core.datasets_map_pkey"

Column	Type	Storage	Description
datasets_map_id	integer	plain	
primary key, btree, for table "core.datasets_map"			

Index "core.datasets_pkey"

Column	Type	Storage	Description
id	uuid	plain	
primary key, btree, for table "core.datasets"			

Depth (& Mudweight Viscosity)

Table "core.depth"

Column	Type	Modifiers	Storage	Description
id	bigint	not null default nextval('core.depth_id_seq'::reg class)	plain	
surface_site_id	uuid		plain	
depth_date	date		plain	
depth	double precision		plain	
true_vertical_depth	double precision		plain	
dataset_id	uuid		plain	
bit_diameter	double precision		plain	
mud_weight	double precision		plain	
viscosity	double precision		plain	

Indexes:

"depth_pkey" PRIMARY KEY, btree (id)

"depth_site_idx" btree (surface_site_id)

Has OIDs: no

Sequence "core.depth_id_seq"

Column	Type	Value	Storage	Description
sequence_name	name	depth_id_seq	plain	
last_value	bigint	3592037	plain	
start_value	bigint	1	plain	
increment_by	bigint	1	plain	
max_value	bigint	9223372036854775807	plain	
min_value	bigint	1	plain	
cache_value	bigint	1	plain	
log_cnt	bigint	22	plain	
is_cycled	boolean	f	plain	
is_called	boolean	t	plain	

View "core.depth_max_temp"

Column	Type	Modifiers	Storage	Description
surface_site_id	uuid		plain	
depth	double precision		plain	

View definition:

```
SELECT depth.surface_site_id, max(depth.depth) AS depth
FROM core.depth
GROUP BY depth.surface_site_id;
```

Index "core.depth_pkey"

Column	Type	Storage	Description
id	bigint	plain	
primary key, btree, for table "core.depth"			

Index "core.depth_site_idx"

Column	Type	Storage	Description
surface_site_id	uuid	plain	
btree, for table "core.depth"			

Field Report Mapping ID

Sequence "core.field_report_mapping_id_seq"

Column	Type	Value	Storage	Description
sequence_name	name	field_report_mapping_id_seq	plain	
last_value	bigint	16	plain	
start_value	bigint	16	plain	
increment_by	bigint	1	plain	
max_value	bigint	9223372036854775807	plain	
min_value	bigint	1	plain	
cache_value	bigint	1	plain	
log_cnt	bigint	1	plain	
is_cycled	boolean	f	plain	
is_called	boolean	f	plain	

File Upload

Table "core.file_upload"

Column	Type	Modifiers	Storage	Description
file_upload_id	integer	not null default nextval('core.file_upload_file_upload_id_seq'::regclass)	plain	
file_name	character varying(100)		extended	
file_size	character varying(25)		extended	
file_path	character varying(256)		extended	
created_date	timestamp without time zone		plain	
processed_date	timestamp without time zone		plain	
status_code	integer		plain	
user_id	integer		plain	

error_message	character varying(2048)		extended
job_execution_id	integer	not null	plain
datasets_id	uuid	not null	plain
job_type	character varying(50)		extended
created_timestamp	bigint		plain

Indexes:

"file_upload_pkey" PRIMARY KEY, btree (file_upload_id)

Foreign-key constraints:

"job_execution_id_fk" FOREIGN KEY (job_execution_id) REFERENCES
gtdabatch.batch_job_execution(job_execution_id)

Has OIDs: no

Sequence "core.file_upload_file_upload_id_seq"

Column	Type	Value	Storage	Description
sequence_name	name	file_upload_file_upload_id_seq	plain	
last_value	bigint	189	plain	
start_value	bigint	1	plain	
increment_by	bigint	1	plain	
max_value	bigint	9223372036854775807	plain	
min_value	bigint	1	plain	
cache_value	bigint	1	plain	
log_cnt	bigint	32	plain	
is_cycled	boolean	f	plain	
is_called	boolean	t	plain	

Index "core.file_upload_pkey"

Column	Type	Storage	Description
file_upload_id	integer	plain	
primary key, btree, for table "core.file_upload"			

Filter Condition

Table "core.filter_condition"

Column	Type	Modifiers	Storage	Description
user_id	integer		plain	
filter_type	name		plain	
field_name	name		plain	
operator_name	character varying(50)		extended	
static_value	character varying(1000)		extended	
condition_comments	character varying(1000)		extended	
dataset_id	uuid		plain	

Has OIDs: no

Fluid

Table "core.fluid"

Column	Type	Modifiers	Storage	Description
id	uuid	not null	plain	
surface_site_id	uuid		plain	

dataset_id	uuid	plain
cntl_no	integer	plain
lease_id	integer	plain
well_no	character(6)	extended
field_no	integer	plain
oper	integer	plain
uic_class	integer	plain
bbl_vol_inj	integer	plain
max_inj_pressure	integer	plain
top_inj_zone	integer	plain
bot_inj_zone	integer	plain
w_status	integer	plain
sw	character(1)	extended
fw	character(1)	extended
bw	character(1)	extended
air	character(1)	extended
gas	character(1)	extended
lpg	character(1)	extended
co2	character(1)	extended
polymer	character(1)	extended
cycle_yr	date	plain
fw_avg_or_max	character(1)	extended
fw_bbbls_day	integer	plain

Indexes:

"fluid_pkey" PRIMARY KEY, btree (id)

Foreign-key constraints:

"fluid_dataset" FOREIGN KEY (dataset_id) REFERENCES core.datasets(id) MATCH FULL

Has OIDs: no

Table "core.fluid_h10"

Column	Type	Modifiers	Storage	Description
id	bigint	not null default nextval('core.fluid_h10_id_seq'::regclass)	plain	
surface_site_id	uuid		plain	
dataset_id	uuid		plain	
injection_id	uuid		plain	
h10_date	date		plain	
h10_avg_inj_pressure	integer		plain	
h10_max_inj_pressure	integer		plain	
h10_total_vol_bbl	integer		plain	
h10_total_vol_mcf	integer		plain	

Indexes:

"fluid_h10_pkey" PRIMARY KEY, btree (id)

Foreign-key constraints:

"fluid_h10_dataset" FOREIGN KEY (dataset_id) REFERENCES core.datasets(id) MATCH FULL

Has OIDs: no

Sequence "core.fluid_h10_id_seq"

Column	Type	Value	Storage	Description
sequence_name	name	fluid_h10_id_seq	plain	
last_value	bigint	26159040	plain	
start_value	bigint	1	plain	

increment_by	bigint	1	plain
max_value	bigint	9223372036854775807	plain
min_value	bigint	1	plain
cache_value	bigint	1	plain
log_cnt	bigint	0	plain
is_cycled	boolean	f	plain
is_called	boolean	t	plain

Index "core.fluid_h10_pkey"

Column	Type	Storage	Description
id	bigint	plain	
primary key, btree, for table "core.fluid_h10"			

Table "core.fluid_h10h"

Column	Type	Modifiers	Storage	Description
id	bigint	not null default nextval('core.fluid_h10h_id_seq'::regclass)	plain	
surface_site_id	uuid		plain	
dataset_id	uuid		plain	
injection_id	uuid		plain	
h10h_date	date	not null	plain	
h10h_max_hydrocarb_psig	integer		plain	
h10h_max_brine_psig	integer		plain	

Indexes:

"fluid_h10h_pkey" PRIMARY KEY, btree (id)

Foreign-key constraints:

"fluid_h10h_dataset" FOREIGN KEY (dataset_id) REFERENCES core.datasets(id) MATCH FULL

Has OIDs: no

Sequence "core.fluid_h10h_id_seq"

Column	Type	Value	Storage	Description
sequence_name	name	fluid_h10h_id_seq	plain	
last_value	bigint	224078	plain	
start_value	bigint	1	plain	
increment_by	bigint	1	plain	
max_value	bigint	9223372036854775807	plain	
min_value	bigint	1	plain	
cache_value	bigint	1	plain	
log_cnt	bigint	0	plain	
is_cycled	boolean	f	plain	
is_called	boolean	t	plain	

Index "core.fluid_h10h_pkey"

Column	Type	Storage	Description
id	bigint	plain	
primary key, btree, for table "core.fluid_h10h"			

Index "core.fluid_pkey"

Column	Type	Storage	Description

id	uuid	plain

primary key, btree, for table "core.fluid"

Formation

Table "core.formation"

Column	Type	Modifiers	Storage	Description
id	bigint	not null default nextval('core.formation_id_seq'::regc lass)	plain	
surface_site_id	uuid		plain	
dataset_id	uuid		plain	
formation	character varying(50)		extended	
formation_depth	integer		plain	
formation_type	character varying(64)		extended	

Indexes:

"formation_pkey" PRIMARY KEY, btree (id)

Has OIDs: no

Sequence "core.formation_id_seq"

Column	Type	Value	Storage	Description
sequence_name	name	formation_id_seq	plain	
last_value	bigint	8174607	plain	
start_value	bigint	1	plain	
increment_by	bigint	1	plain	
max_value	bigint	9223372036854775807	plain	
min_value	bigint	1	plain	
cache_value	bigint	1	plain	
log_cnt	bigint	22	plain	
is_cycled	boolean	f	plain	
is_called	boolean	t	plain	

Index "core.formation_pkey"

Column	Type	Storage	Description
id	bigint	plain	

primary key, btree, for table "core.formation"

Heat Flow

Table "core.heat_flow"

Column	Type	Modifiers	Storage	Description
id	uuid	not null	plain	
surface_site_id	uuid		plain	
publication_source_id	uuid		plain	
bht_measurements_id	uuid		plain	
precision_log_id	uuid		plain	
thermal_conductivity_id	uuid		plain	
heat_generation_id	bigint		plain	
interval_order	character varying(16)		extended	
interval_top_depth	double precision		plain	

interval_bottom_depth	double precision	plain
gradient	double precision	plain
gradient_se	double precision	plain
gradient_corrected	double precision	plain
gradient_corrected_se	double precision	plain
gradient_correction_type_id	integer	plain
heat_flow	double precision	plain
heat_flow_se	double precision	plain
heat_flow_corrected	double precision	plain
heat_flow_corrected_se	double precision	plain
heat_flow_quality_code_id	integer	plain
heat_flow_quality_method_id	integer	plain
lithology_information_id	integer	plain
notes_id	uuid	plain
dataset_id	uuid	plain
heat_generation	double precision	plain
heat_generation_se	double precision	plain
heat_generation_nb_samples	integer	plain
radioactivity_id	uuid	plain
quality	character varying(16)	extended

Indexes:

"heatflow_pkey" PRIMARY KEY, btree (id)

Foreign-key constraints:

"heat_flow_notes" FOREIGN KEY (notes_id) REFERENCES core.notes(id) MATCH FULL

"heat_flow_plog" FOREIGN KEY (precision_log_id) REFERENCES core.precision_log(id) MATCH FULL

"heat_flow_radioactivity" FOREIGN KEY (radioactivity_id) REFERENCES core.radioactivity(id) MATCH FULL

"heat_flow_thermal" FOREIGN KEY (thermal_conductivity_id) REFERENCES core.thermal_conductivity(id) MATCH FULL

Has OIDs: no

Sequence "core.heat_flow_id_seq"

Column	Type	Value	Storage	Description
sequence_name	name	heat_flow_id_seq	plain	
last_value	bigint	1	plain	
start_value	bigint	1	plain	
increment_by	bigint	1	plain	
max_value	bigint	9223372036854775807	plain	
min_value	bigint	1	plain	
cache_value	bigint	1	plain	
log_cnt	bigint	1	plain	
is_cycled	boolean	f	plain	
is_called	boolean	f	plain	

Heat Flow Publication Link Ids

Sequence "core.heat_flow_publication_links_id_seq"

Column	Type	Value	Storage	Description
sequence_name	name	heat_flow_publication_links_id_seq	plain	
last_value	bigint	1	plain	
start_value	bigint	1	plain	
increment_by	bigint	1	plain	

max_value	bigint	9223372036854775807	plain
min_value	bigint	1	plain
cache_value	bigint	1	plain
log_cnt	bigint	1	plain
is_cycled	boolean	f	plain
is_called	boolean	f	plain

Radiogenic Heat Generation IDs

Sequence "core.heat_generation_id_seq"

Column	Type	Value	Storage	Description
sequence_name	name	heat_generation_id_seq	plain	
last_value	bigint	1	plain	
start_value	bigint	1	plain	
increment_by	bigint	1	plain	
max_value	bigint	9223372036854775807	plain	
min_value	bigint	1	plain	
cache_value	bigint	1	plain	
log_cnt	bigint	1	plain	
is_cycled	boolean	f	plain	
is_called	boolean	f	plain	

Heat Flow

Index "core.heatflow_pkey"

Column	Type	Storage	Description
id	uuid	plain	
primary key, btree, for table "core.heat_flow"			

Ref_State

Index "core.id_idx"

Column	Type	Storage	Description
id	integer	plain	
btree, for table "core.ref_state"			

Intervals

Table "core.interval"

Column	Type	Modifiers	Storage	Description
id	bigint	not null default nextval('core.interval_id_seq'::regc lass)	plain	
surface_site_id	uuid		plain	
dataset_id	uuid		plain	
top_depth	double precision		plain	
bottom_depth	double precision		plain	
interval_length	double precision		plain	
water_volume	double precision		plain	
bhp	double precision		plain	

producing_reservoir	character varying(64)	extended
duration_hours	double precision	plain

Indexes:
 "interval_pkey" PRIMARY KEY, btree (id)
 Has OIDs: no

Sequence "core.interval_id_seq"

Column	Type	Value	Storage	Description
sequence_name	name	interval_id_seq	plain	
last_value	bigint	596067	plain	
start_value	bigint	1	plain	
increment_by	bigint	1	plain	
max_value	bigint	9223372036854775807	plain	
min_value	bigint	1	plain	
cache_value	bigint	1	plain	
log_cnt	bigint	16	plain	
is_cycled	boolean	f	plain	
is_called	boolean	t	plain	

Index "core.interval_pkey"

Column	Type	Storage	Description
id	bigint	plain	

primary key, btree, for table "core.interval"

Lithology

Sequence "core.lithology_information_id_seq"

Column	Type	Value	Storage	Description
sequence_name	name	lithology_information_id_seq	plain	
last_value	bigint	1	plain	
start_value	bigint	1	plain	
increment_by	bigint	1	plain	
max_value	bigint	9223372036854775807	plain	
min_value	bigint	1	plain	
cache_value	bigint	1	plain	
log_cnt	bigint	1	plain	
is_cycled	boolean	f	plain	
is_called	boolean	f	plain	

Materialized Views

Table "core.matviews"

Column	Type	Modifiers	Storage	Description
mv_name	name	not null	plain	
v_name	name	not null	plain	
last_refresh	timestamp with time zone		plain	

Indexes:
 "matviews_pkey" PRIMARY KEY, btree (mv_name)
 Has OIDs: no

Index "core.matviews_pkey"

Column	Type	Storage	Description
mv_name	cstring	plain	
primary key, btree, for table "core.matviews"			

Table "core.ngds_heat_flow_mview"

Column	Type	Modifiers	Storage	Description
ObservationURI	uuid		plain	
ObservationName	text		extended	
Label	text		extended	
HeaderURI	uuid		plain	
point	geometry		main	
APINo	character varying(18)		extended	
WellName	name		plain	
OtherId	character varying(32)		extended	
Owner	character varying(128)		extended	
LeaseName	character varying(128)		extended	
Field	character varying(128)		extended	
County	name		plain	
State	name		plain	
LatDegreeWGS84	numeric(10,7)		main	
LongDegreeWGS84	numeric(10,7)		main	
LocationUncertaintyStatement	character varying(4000)		extended	
PLSS_Meridians	character varying(64)		extended	
Township	character varying(5)		extended	
Range	character varying(5)		extended	
Section	character varying(5)		extended	
SectionPart	character varying(5)		extended	
TectonicProvince	character varying(4000)		extended	
OtherLocationName	name		plain	
WellType	character varying(100)		extended	
dataset_id	uuid		plain	
Has OIDs: no				

Heat Flow Notes

Table "core.notes"

Column	Type	Modifiers	Storage	Description
id	uuid	not null	plain	
notes	character varying(4000)		extended	
surface_site_id	uuid		plain	
dataset_id	uuid		plain	

Indexes:

"notes_pkey" PRIMARY KEY, btree (id)

Referenced by:

TABLE "core.heat_flow" CONSTRAINT "heat_flow_notes" FOREIGN KEY (notes_id) REFERENCES core.notes(id) MATCH FULL

Has OIDs: no

Index "core.notes_pkey"

Column	Type	Storage	Description

id	uuid	plain
primary key, btree, for table "core.notes"		

Precision TD Logs

Table "core.precision_log"

Column	Type	Modifiers	Storage	Description
id	uuid	not null	plain	
surface_site_id	uuid		plain	
precision_log_source_id	character varying(256)	not null	extended	The precision log td curve file name.
log_date	character varying(10)		extended	The precision log date, but someytimes we only have partial information. A string permit that.
log_depth	double precision		plain	
log_deepest_temperature	double precision		plain	
log_max_temperature	double precision		plain	
time_since_circulation	double precision		plain	
log_equipment_id	integer		plain	
log_company	character varying(64)		extended	
dataset_id	uuid		plain	

Indexes:

"precision_log_pkey" PRIMARY KEY, btree (id)

"precision_log_site_idx" btree (surface_site_id)

Referenced by:

TABLE "core.heat_flow" CONSTRAINT "heat_flow_plog" FOREIGN KEY (precision_log_id)

REFERENCES core.precision_log(id) MATCH FULL

TABLE "core.temperature_series" CONSTRAINT "temperature_series_plog" FOREIGN KEY

(precision_log_id) REFERENCES core.precision_log(id) MATCH FULL

Has OIDs: no

View "core.precision_log_max_temp"

Column	Type	Modifiers	Storage	Description
surface_site_id	uuid		plain	
temperature	double precision		plain	

View definition:

```
SELECT precision_log.surface_site_id, max(precision_log.log_max_temperature) AS temperature
FROM core.precision_log
GROUP BY precision_log.surface_site_id;
```

Index "core.precision_log_pkey"

Column	Type	Storage	Description
id	uuid	plain	
primary key, btree, for table "core.precision_log"			

Index "core.precision_log_site_idx"

Column	Type	Storage	Description
surface_site_id	uuid	plain	
btree, for table "core.precision_log"			

Publications

Table "core.publication"

Column	Type	Modifiers	Storage	Description
publication_id	integer	not null default nextval('core.publication_publication_id_seq'::regclass)	plain	
authors	character varying(1024)		extended	
title	character varying(512)		extended	
year	character varying(64)		extended	
report_number	character varying(100)		extended	
pages	character varying(100)		extended	
agency	character varying(512)		extended	
contract	character varying(128)		extended	
source_number	character varying(64)		extended	
source	character varying(64)		extended	
location	character varying(128)		extended	
pdf	character varying(32)		extended	
electronic	character varying(32)		extended	
url	character varying(2000)		extended	
month	numeric(2,0)		main	
publisher	character varying(512)		extended	
volume	character varying(128)		extended	
issue	character varying(64)		extended	
collection	character varying(256)		extended	
keywords	character varying(1024)		extended	
publication_uuid	uuid		plain	
ocr_required	boolean	default false	plain	
expected	boolean	default true	plain	
dataset_id	uuid		plain	
data_source_id	integer		plain	
indexed	boolean		plain	
status	character(10)		extended	

Indexes:

"publication_pkey" PRIMARY KEY, btree (publication_id)

Has OIDs: no

Publication Metadata

View "core.publication_metadata"

Column	Type	Modifiers	Storage	Description
id	text		extended	
resource_id	uuid		plain	
metadata_uuid	uuid		plain	
title	character varying(512)		extended	
description	character varying(512)		extended	

author	character varying(1024)	extended
publication_date	character varying(64)	extended
resource_type	unknown	plain
keywords_thematic	character varying(1024)	extended
resource_languages	unknown	plain
resource_url	character varying(256)	extended
metadata_date	timestamp with time zone	plain
metadata_language	unknown	plain
metadata_contact_org_name	unknown	plain
metadata_contact_person_name	unknown	plain
metadata_contact_position_name	unknown	plain
metadata_contact_email	unknown	plain
metadata_contact_street_address	unknown	plain
metadata_contact_city	unknown	plain
metadata_contact_state	unknown	plain
metadata_contact_zip	unknown	plain
metadata_contact_phone	unknown	plain

View definition:

```
SELECT 'bib':::text
COALESCE(publication.publication_uuid, core.generate_uuid(publication.source, 'PUBLICATION':character
varying, publication.source_number)) AS id, COALESCE(publication.publication_uuid,
core.generate_uuid(publication.source, 'PUBLICATION':character varying, publication.source_number)) AS
resource_id, publication.publication_uuid AS metadata_uuid, publication.title, publication.title AS description,
publication.authors AS author, publication.year AS publication_date, 'Document' AS resource_type,
publication.keywords AS keywords_thematic, 'en' AS resource_languages, files.file_path_external AS
resource_url, now() AS metadata_date, 'en' AS metadata_language, 'SMU Geothermal Laboratory, Roy M.
Huffington Department of Earth Sciences' AS metadata_contact_org_name, 'Maria Richard' AS
metadata_contact_person_name, 'Coordinator, SMU Geothermal Laboratory' AS
metadata_contact_position_name, 'mrichard@mail.smu.edu' AS metadata_contact_email, 'P.O. Box 75-0395' AS
metadata_contact_street_address, 'Dallas' AS metadata_contact_city, 'TX' AS metadata_contact_state, '75275-
0395' AS metadata_contact_zip, '+1 (214) 768 1975' AS metadata_contact_phone
FROM core.publication publication
LEFT JOIN filesystem.files ON files.file_uuid = publication.publication_uuid
WHERE publication.status = '0':bpchar OR publication.status = '1':bpchar;
```

Index "core.publication_pkey"

Column	Type	Storage	Description
publication_id	integer	plain	
primary key, btree, for table "core.publication"			

Sequence "core.publication_publication_id_seq"

Column	Type	Value	Storage	Description
sequence_name	name	publication_publication_id_seq	plain	
last_value	bigint	5302	plain	
start_value	bigint	1	plain	
increment_by	bigint	1	plain	
max_value	bigint	9223372036854775807	plain	
min_value	bigint	1	plain	
cache_value	bigint	1	plain	
log_cnt	bigint	32	plain	
is_cycled	boolean	f	plain	

is_called	boolean	t	plain

Quality Issue Log ID

Sequence "core.quality_issue_log_id_seq"

Column	Type	Value	Storage	Description
sequence_name	name	quality_issue_log_id_seq	plain	
last_value	bigint	1	plain	
start_value	bigint	1	plain	
increment_by	bigint	1	plain	
max_value	bigint	9223372036854775807	plain	
min_value	bigint	1	plain	
cache_value	bigint	1	plain	
log_cnt	bigint	1	plain	
is_cycled	boolean	f	plain	
is_called	boolean	f	plain	

Radioactivity

Table "core.radioactivity"

Column	Type	Modifiers	Storage	Description
id	uuid	not null	plain	
surface_site_id	uuid		plain	
dataset_id	uuid		plain	
laboratory	character varying(64)		extended	
measurement_procedure	character varying(64)		extended	
measurement_device	character varying(64)		extended	
rock	character varying(64)		extended	
specimen_label	character varying(32)		extended	
potassium	double precision		plain	
thorium	double precision		plain	
uranium	double precision		plain	
comments	character varying(128)		extended	

Indexes:

"radioactivity_pkey" PRIMARY KEY, btree (id)

Referenced by:

TABLE "core.heat_flow" CONSTRAINT "heat_flow_radioactivity" FOREIGN KEY (radioactivity_id)

REFERENCES core.radioactivity(id) MATCH FULL

Has OIDs: no

Index "core.radioactivity_pkey"

Column	Type	Storage	Description
id	uuid	plain	
primary key, btree, for table "core.radioactivity"			

County

Table "core.ref_county"

Column	Type	Modifiers	Storage	Description
id	integer	not null	plain	

state_id	integer	plain
fips_code	character varying(5)	extended
county_name	character varying(50)	extended
geometry	geometry	main
min_elevation	double precision	plain
max_elevation	double precision	plain

Indexes:

"county_pkey" PRIMARY KEY, btree (id)
 "county_id_idx" btree (id)
 "county_state_idx" btree (state_id)
 "ref_county_geom_gist" gist (geometry)
 "ref_county_shape" gist (geometry)

Check constraints:

"county_enforce_dims_shape" CHECK (st_ndims(geometry) = 2)
 "county_enforce_geotype_shape" CHECK (geometrytype(geometry) = 'MULTIPOLYGON'::text OR geometry

IS NULL)

"county_enforce_srid_shape" CHECK (st_srid(geometry) = 4326)

Has OIDs: no

Elevation

View "core.ref_county_elev"

Column	Type	Modifiers	Storage	Description
id	integer		plain	
min_elevation	double precision		plain	
max_elevation	double precision		plain	

View definition:

```
SELECT c.id,
       CASE
         WHEN c.min_elevation IS NOT NULL THEN c.min_elevation
         ELSE 0::double precision
       END AS min_elevation,
       CASE
         WHEN c.max_elevation IS NOT NULL THEN c.max_elevation
         ELSE 6100::double precision
       END AS max_elevation
FROM core.ref_county c;
```

County

Index "core.ref_county_geom_gist"

Column	Type	Storage	Description
geometry	box2d	plain	
gist, for table "core.ref_county"			

Index "core.ref_county_shape"

Column	Type	Storage	Description
geometry	box2d	plain	
gist, for table "core.ref_county"			

Data Source

Table "core.ref_data_source"

Column	Type	Modifiers	Storage	Description
id	integer	not null	plain	
ref_name	character varying(10)		extended	
ref_description	character varying(4000)		extended	

Indexes:

"ref_data_source_pkey" PRIMARY KEY, btree (id)

Has OIDs: no

Index "core.ref_data_source_pkey"

Column	Type	Storage	Description
id	integer	plain	
primary key, btree, for table "core.ref_data_source"			

Quality Issue Type

Table "core.ref_quality_issue_type"

Column	Type	Modifiers	Storage	Description
id	integer	not null	plain	
pid	integer		plain	
ref_name	character varying(100)		extended	
ref_description	character varying(4000)		extended	

Indexes:

"ref_quality_issue_type_pkey" PRIMARY KEY, btree (id)

Has OIDs: no

View "core.ref_quality_issue_type_h"

Column	Type	Modifiers	Storage	Description
level	integer		plain	
id	integer		plain	
child_id	integer		plain	
parent_name	character varying		extended	
child_name	character varying		extended	
cycle	boolean		plain	

View definition:

```
WITH RECURSIVE t(level, id, child_id, parent_name, child_name, cycle) AS (
    SELECT 1, NULL::integer AS int4, NULL::integer AS int4, "::character varying AS "varchar",
    "::character varying AS "varchar", false AS bool
    UNION ALL
    SELECT t.level + 1, parent.id, child.id, parent.ref_name, child.ref_name, true AS bool
    FROM core.ref_quality_issue_type parent
    JOIN core.ref_quality_issue_type child ON parent.id = child.pid
    CROSS JOIN t
    WHERE NOT t.cycle
)
SELECT t.level, t.id, t.child_id, t.parent_name, t.child_name, t.cycle
FROM t
WHERE t.id IS NOT NULL;
```

Index "core.ref_quality_issue_type_pkey"

Column	Type	Storage	Description
id	integer	plain	
primary key, btree, for table "core.ref_quality_issue_type"			

State

Table "core.ref_state"

Column	Type	Modifiers	Storage	Description
id	integer	not null	plain	
fips_code	character varying(2)		extended	
state_name	character varying(32)		extended	
state_abbr	character varying(2)		extended	
sub_region	character varying(7)		extended	
shape	geometry		main	

Indexes:

"state_pkey" PRIMARY KEY, btree (id)

"id_idx" btree (id)

"ref_state_shape" gist (shape)

Check constraints:

"state_enforce_dims_shape" CHECK (st_ndims(shape) = 2)

"state_enforce_geotype_shape" CHECK (geometrytype(shape) = 'MULTIPOLYGON'::text OR shape IS NULL)

"state_enforce_srid_shape" CHECK (st_srid(shape) = 4326)

Has OIDs: no

Index "core.ref_state_shape"

Column	Type	Storage	Description
shape	box2d	plain	
gist, for table "core.ref_state"			

Table "core.ref_symbol"

Column	Type	Modifiers	Storage	Description

id	integer	not null	plain
ref_name	character varying(10)		extended
ref_description	character varying(4000)		extended

Indexes:

"ref_symbol_pkey" PRIMARY KEY, btree (id)

Has OIDs: no

Index "core.ref_symbol_pkey"

Column	Type	Storage	Description
id	integer	plain	
primary key, btree, for table "core.ref_symbol"			

Geologic Reservoir

Table "core.reservoir"

Column	Type	Modifiers	Storage	Description
id	uuid	not null	plain	
dataset_id	uuid		plain	
reservoiruri	character varying(255)		extended	
reservoirname	character varying(255)		extended	
label	character varying(255)		extended	
otherid	character varying(255)		extended	
othername	character varying(255)		extended	
description	character varying(255)		extended	
county	character varying(255)		extended	
state	character varying(255)		extended	
locationname	character varying(255)		extended	
latdegree	double precision		plain	
longdegree	double precision		plain	
srs	character varying(255)		extended	
locationuncertaintystatement	character varying(255)		extended	
temperaturemax_f	double precision		plain	
temperaturemin_f	double precision		plain	
depthtotop_ft	double precision		plain	
depthtobottom_ft	double precision		plain	
dominantformationname	character varying(255)		extended	
otherformationname	character varying(255)		extended	
dominantlithologyname	character varying(255)		extended	
otherlithologyname	character varying(255)		extended	
area_acres	double precision		plain	
width_ft	double precision		plain	
length_ft	double precision		plain	
sandvolume_cft	double precision		plain	
blockvolume_cft	double precision		plain	
percentsand	double precision		plain	
averagenetsandthickness_ft	double precision		plain	
wellcount	integer		plain	
faultcount	integer		plain	
faulted	character varying(255)		extended	
faulttype	character varying(255)		extended	
faulthydrauliccharacter	character varying(255)		extended	
firstproductiondate	date		plain	
lastproductiondate	date		plain	

years in production	integer	plain
monthly production_bbl	double precision	plain
production report date	date	plain
average monthly water production_bbl	double precision	plain
minimum monthly water production_bbl	double precision	plain
maximum monthly water production_bbl	double precision	plain
water injection_bbl	double precision	plain
water saturation_pct	double precision	plain
water tds_mgl	double precision	plain
average field wide pressure_psi	double precision	plain
average field wide gradient_degfft	double precision	plain
porosity max_pct	double precision	plain
porosity min_pct	double precision	plain
permeability max_md	double precision	plain
permeability min_md	double precision	plain
cementation mineralogy	character varying(255)	extended
trap	character varying(255)	extended
drive	character varying(255)	extended
gravity	double precision	plain
commodity of interest	character varying(255)	extended
stimulation	character varying(255)	extended
average perforated thickness_ft	double precision	plain
related resource	character varying(255)	extended
source	character varying(1023)	extended
metadata auri	character varying(255)	extended
major fields	character varying(511)	extended
shape	geometry	main

Indexes:

"reservoir_pkey" PRIMARY KEY, btree (id)

Has OIDs: no

Index "core.reservoir_pkey"

Column	Type	Storage	Description
id	uuid	plain	
primary key, btree, for table "core.reservoir"			

Shape

Index "core.shape_gix"

Column	Type	Storage	Description
shape	box2d	plain	
gist, for table "core.surface_site"			

Site Comments

Sequence "core.site_comments_id_seq"

Column	Type	Value	Storage	Description
sequence_name	name	site_comments_id_seq	plain	
last_value	bigint	1	plain	
start_value	bigint	1	plain	
increment_by	bigint	1	plain	

max_value	bigint	9223372036854775807	plain
min_value	bigint	1	plain
cache_value	bigint	1	plain
log_cnt	bigint	1	plain
is_cycled	boolean	f	plain
is_called	boolean	f	plain

Publication Links & Related Resources

Sequence "core.site_publication_links_id_seq"

Column	Type	Value	Storage	Description
sequence_name	name	site_publication_links_id_seq	plain	
last_value	bigint	1	plain	
start_value	bigint	1	plain	
increment_by	bigint	1	plain	
max_value	bigint	9223372036854775807	plain	
min_value	bigint	1	plain	
cache_value	bigint	1	plain	
log_cnt	bigint	1	plain	
is_cycled	boolean	f	plain	
is_called	boolean	f	plain	

Table "core.site_related_resources"

Column	Type	Modifiers	Storage	Description
surface_site_id	uuid		plain	
label	character varying(128)		extended	
url	character varying(512)		extended	
type	character varying(32)		extended	
resource_id	uuid		plain	
description	character varying(512)		extended	
dataset_id	uuid		plain	
shape	geometry		main	
source	character varying(32)		extended	
record_id	integer	not null default nextval('core.site_related_resourc es_record_id_seq'::regclass)	plain	

Indexes:

"site_related_resources_pkey" PRIMARY KEY, btree (record_id)

Has OIDs: no

Sequence "core.site_related_resources_id_seq"

Column	Type	Value	Storage	Description
sequence_name	name	site_related_resources_id_seq	plain	
last_value	bigint	1	plain	
start_value	bigint	1	plain	
increment_by	bigint	1	plain	
max_value	bigint	9223372036854775807	plain	
min_value	bigint	1	plain	
cache_value	bigint	1	plain	
log_cnt	bigint	1	plain	
is_cycled	boolean	f	plain	
is_called	boolean	f	plain	

Index "core.site_related_resources_pkey"

Column	Type	Storage	Description
record_id	integer	plain	
primary key, btree, for table "core.site_related_resources"			

Sequence "core.site_related_resources_record_id_seq"

Column	Type	Value	Storage	Description
sequence_name	name	site_related_resources_record_id_seq	plain	
last_value	bigint	128279	plain	
start_value	bigint	1	plain	
increment_by	bigint	1	plain	
max_value	bigint	9223372036854775807	plain	
min_value	bigint	1	plain	
cache_value	bigint	1	plain	
log_cnt	bigint	4	plain	
is_cycled	boolean	f	plain	
is_called	boolean	t	plain	

Aggregated Well Data Tier 2 (Also called Surface Site County State)

Index "core.state_pkey"

Column	Type	Storage	Description
id	integer	plain	
primary key, btree, for table "core.ref_state"			

Table "core.surface_site"

Column	Type	Modifiers	Storage	Description
id	uuid	not null	plain	
api	character varying(18)		extended	
site_name	character varying(64)		extended	Can be a well_name
source_id	character varying(32)		extended	Is an external identifier used to uniquely allocate used uuid
site_owner	character varying(128)		extended	
lease_name	character varying(128)		extended	
field	character varying(128)		extended	
county_id	integer		plain	
latitude	numeric(10,7)		main	
longitude	numeric(10,7)		main	
plss_meridians	character varying(64)		extended	
township	character varying(5)		extended	
range	character varying(5)		extended	
section	character varying(10)		extended	
section_part	character varying(5)		extended	
other_location_name	character varying(256)		extended	
elevation_gl	double precision		plain	
elevation_df	double precision		plain	
elevation_kb	double precision		plain	

surface_temperature	double precision	plain
shape	geometry	main
dataset_id	uuid	plain
basement_depth	double precision	plain
average_heatflow	double precision	plain
tectonic_province	character varying(128)	extended
well_type	character varying(128)	extended
plugged	character varying(64)	extended
shut_in_date	date	plain
block	character varying(10)	extended
feet_from_survey	integer	plain
section_block_survey	character varying(52)	extended
hole_open	integer	plain
fresh_water_converted	integer	plain
district	integer	plain
lithology	character varying(255)	extended
comment	character varying(512)	extended
contributor	character varying(32)	extended
mintemp	numeric(10,7)	main
maxtemp	numeric(10,7)	main
ams	character varying(64)	extended
location_notes	character varying(128)	extended
original_source	character varying(16)	extended
operator	character varying(128)	extended
other_ids	character varying(128)	extended

Indexes:

"surface_site_pkey" PRIMARY KEY, btree (id)
"dataset_id_idx" UNIQUE, btree (id)
"dataset_surface_county_id_idx" btree (county_id)
"shape_gix" gist (shape)

Check constraints:

"enforce_dims_shape" CHECK (st_ndims(shape) = 2)
"enforce_geotype_shape" CHECK (geometrytype(shape) = 'POINT'::text OR shape IS NULL)
"enforce_srid_shape" CHECK (st_srid(shape) = 4326)

Has OIDs: no

Table "core.surface_site_county_state_materialized_view"

Column	Type	Modifiers	Storage	Description
id	character varying		extended	
uuid	character varying		extended	
name	character varying		extended	
site_name	character varying(64)		extended	
other_location_name	character varying(256)		extended	
api	character varying(18)		extended	
latitude	numeric(10,7)		main	
longitude	numeric(10,7)		main	
state_id	integer		plain	
state	character varying(32)		extended	
state_code	character varying(2)		extended	
county	character varying(50)		extended	
county_id	integer		plain	
bht	double precision		plain	
max_temperature	double precision		plain	

depth	double precision	plain
shape	geometry	main
datasource	character varying(16)	extended
thermal_conductivity	double precision	plain
heat_flow	double precision	plain
precision_log_source_id	character varying(256)	extended
Has OIDs: no		

View "core.surface_site_county_state_view"

Column	Type	Modifiers	Storage	Description
id	character varying		extended	
uuid	character varying		extended	
name	character varying		extended	
site_name	character varying(64)		extended	
other_location_name	character varying(256)		extended	
api	character varying(18)		extended	
latitude	numeric(10,7)		main	
longitude	numeric(10,7)		main	
state_id	integer		plain	
state	character varying(32)		extended	
state_code	character varying(2)		extended	
county	character varying(50)		extended	
county_id	integer		plain	
bht	double precision		plain	
max_temperature	double precision		plain	
depth	double precision		plain	
shape	geometry		main	
datasource	character varying(16)		extended	
thermal_conductivity	double precision		plain	
heat_flow	double precision		plain	
precision_log_source_id	character varying(256)		extended	

View definition:

```
SELECT DISTINCT s.id::character varying AS id, s.id::character varying AS uuid, COALESCE(s.site_name,
s.api, s.other_location_name) AS name, s.site_name, s.other_location_name, s.api, s.latitude, s.longitude, st.id AS
state_id, st.state_name AS state, st.state_abbr AS state_code, c.county_name AS county, c.id AS county_id,
bht.bht, s.maxtemp::double precision AS max_temperature, GREATEST(d.depth, bht.bht_depth) AS depth,
s.shape, ds.datasource, tc.conductivity AS thermal_conductivity, COALESCE(s.average_heatflow, hf.heat_flow)
AS heat_flow, pl.precision_log_source_id
FROM core.surface_site s
LEFT JOIN core.ref_county c ON c.id = s.county_id
LEFT JOIN core.ref_state st ON st.id = c.state_id
LEFT JOIN core.depth d ON d.surface_site_id = s.id
LEFT JOIN core.bht_measurements bht ON bht.surface_site_id = s.id
LEFT JOIN core.datasets ds ON ds.id = s.dataset_id
LEFT JOIN core.thermal_conductivity tc ON tc.surface_site_id = s.id
LEFT JOIN core.heat_flow hf ON hf.surface_site_id = s.id
LEFT JOIN core.precision_log pl ON pl.surface_site_id = s.id
WHERE bht.estimated IS NULL OR bht.estimated = false
ORDER BY s.id::character varying;
```

Index "core.surface_site_pkey"

Column	Type	Storage	Description
id	uuid	plain	

primary key, btree, for table "core.surface_site"

Temperature Series (TD Curves)

Table "core.temperature_series"

Column	Type	Modifiers	Storage	Description
id	bigint	not null default nextval('core.temperature_series_id_seq'::reg class)	plain	
precision_log_id	uuid		plain	
depth	double precision		plain	
temperature	double precision		plain	
dataset_id	uuid		plain	

Indexes:

"temperature_series_pkey" PRIMARY KEY, btree (id)

Foreign-key constraints:

"temperature_series_plog" FOREIGN KEY (precision_log_id) REFERENCES core.precision_log(id) MATCH
FULL

Has OIDs: no

Sequence "core.temperature_series_id_seq"

Column	Type	Value	Storage	Description
sequence_name	name	temperature_series_id_seq	plain	
last_value	bigint	949783	plain	
start_value	bigint	1	plain	
increment_by	bigint	1	plain	
max_value	bigint	9223372036854775807	plain	
min_value	bigint	1	plain	
cache_value	bigint	1	plain	
log_cnt	bigint	17	plain	
is_cycled	boolean	f	plain	
is_called	boolean	t	plain	

Index "core.temperature_series_pkey"

Column	Type	Storage	Description
id	bigint	plain	

primary key, btree, for table "core.temperature_series"

Borehole Temperature Observation Content Model

View "core.template_borehole"

Column	Type	Modifiers	Storage	Description
ObservationURI	text		extended	
WellName	character varying		extended	
APINo	character varying(18)		extended	
HeaderURI	text		extended	
OtherID	text		extended	
Label	text		extended	
Operator	character varying(128)		extended	

LeaseName	character varying(128)	extended
LeaseOwner	character varying(128)	extended
LeaseNo	text	extended
SpudDate	text	extended
EndedDrillingDate	text	extended
welltype	character varying(128)	extended
Status	character varying(64)	extended
CommodityOfInterest	character varying	extended
StatusDate	text	extended
Field	character varying(128)	extended
OtherLocationName	character varying(256)	extended
County	character varying	extended
State	character varying	extended
PLSS_Meridians	character varying(64)	extended
Township	character varying(5)	extended
Range	character varying(5)	extended
Section	character varying(10)	extended
SectionPart	character varying(5)	extended
LatDegree	numeric(10,7)	main
LongDegree	numeric(10,7)	main
SRS	character varying(6)	extended
LocationUncertaintyStatement	character varying(128)	extended
DrillerTotalDepth	double precision	plain
DepthReferencePoint	character varying(1)	extended
LengthUnits	character varying(1)	extended
WellBoreShape	character varying(1)	extended
TrueVerticalDepth	double precision	plain
ElevationKB	double precision	plain
ElevationDF	double precision	plain
ElevationGL	double precision	plain
FormationTD	text	extended
BitDiameterCollar	double precision	plain
DiameterUnits	character varying(6)	extended
MeasuredTemperature	double precision	plain
CorrectedTemperature	double precision	plain
TemperatureUnits	character varying(1)	extended
TimeSinceCirculation	double precision	plain
CorrectionType	integer	plain
DepthOfMeasurement	double precision	plain
MeasurementDateTime	date	plain
MeasurementSource	character varying(16)	extended
CasingBottomDepthDriller	double precision	plain
CasingTopDepth	numeric	main
CasingPipeDiameter	double precision	plain
CasingWeight	double precision	plain
CasingWeightUnits	character varying(6)	extended
Viscosity	double precision	plain
InformationSource	character varying	extended
Shape	geometry	main

View definition:
WITH depth AS (
SELECT depth.surface_site_id, max(depth.true_vertical_depth) AS true_vertical_depth, max(depth.depth)
AS depth, max(depth.bit_diameter) AS bit_diameter, array_to_string(array_agg(DISTINCT depth.depth_date),
"::text) AS depth_date, max(depth.viscosity) AS viscosity
FROM core.depth
GROUP BY depth.surface_site_id


```

), casing AS (
  SELECT t.surface_site_id, t.casing_depth,
    CASE
      WHEN t.casing_top::text ~ '^[0-9]+$'::text THEN t.casing_top::numeric
      ELSE (-999999)::numeric
    END AS casing_top, t.casing_size_inches, t.casing_weight
  FROM core.casing t
  JOIN ( SELECT casing.surface_site_id, max(casing.file_date) AS max_date, max(casing.casing_depth) AS
max_depth
        FROM core.casing
        GROUP BY casing.surface_site_id) x ON t.surface_site_id = x.surface_site_id AND (x.max_date IS
NULL OR t.file_date = x.max_date) AND t.casing_depth = x.max_depth
), well_info AS (
  SELECT well_info.surface_site_id, array_to_string(array_agg(DISTINCT well_info.commodity), '|'::text)
AS commodity, array_to_string(array_agg(DISTINCT well_info.well_id), '|'::text) AS well_id
  FROM core.well_info
  GROUP BY well_info.surface_site_id
), well_dates AS (
  SELECT well_dates.surface_site_id, array_to_string(array_agg(DISTINCT well_dates.drl_compl_date),
'|'::text) AS drl_compl_date, array_to_string(array_agg(DISTINCT well_dates.drl_start_date), '|'::text) AS
drl_start_date, array_to_string(array_agg(DISTINCT well_dates.status_date), '|'::text) AS status_date
  FROM core.well_dates
  GROUP BY well_dates.surface_site_id
), formation AS (
  SELECT formation.surface_site_id, array_to_string(array_agg(DISTINCT (formation.formation::text ||
depth)::text) ||
    CASE
      WHEN formation.formation_depth IS NULL THEN 'NA'::text
      ELSE "::text || formation.formation_depth
    END), '|'::text) AS formation
  FROM core.formation
  GROUP BY formation.surface_site_id
)
SELECT 'https://geothermal.smu.edu/obervation/borehole/'::text || b.id AS "ObservationURI",
COALESCE(s.site_name, s.api) AS "WellName", s.api AS "APINo", 'https://geothermal.smu.edu/header/'::text ||
s.id AS "HeaderURI", wi.well_id AS "OtherID", 'Borehole observation for well '::text || COALESCE(s.site_name,
s.api)::text AS "Label", s.operator AS "Operator", s.lease_name AS "LeaseName", s.site_owner AS
"LeaseOwner", wi.well_id AS "LeaseNo", wd.drl_start_date AS "SpudDate", COALESCE(d.depth_date, "::text ||
wd.drl_compl_date) AS "EndedDrillingDate", s.well_type AS welltype, s.plugged AS "Status",
COALESCE(s.well_type, wi.commodity::character varying) AS "CommodityOfInterest", (('status_date '::text ||
wd.status_date) || ' shut_in_date '::text) || s.shut_in_date AS "StatusDate", s.field AS "Field",
s.other_location_name AS "OtherLocationName", COALESCE(rc.county_name, 'nil:missing'::character varying)
AS "County", COALESCE(rs.state_name, 'nil:missing'::character varying) AS "State", s.plss_meridians AS
"PLSS_Meridians", s.township AS "Township", s.range AS "Range", s.section AS "Section", s.section_part AS
"SectionPart", s.latitude AS "LatDegree", s.longitude AS "LongDegree", 'WGS84'::character varying(6) AS
"SRS", s.location_notes AS "LocationUncertaintyStatement", d.depth AS "DrillerTotalDepth", "::character
varying(1) AS "DepthReferencePoint", 'm'::character varying(1) AS "LengthUnits", "::character varying(1) AS
"WellBoreShape", d.true_vertical_depth AS "TrueVerticalDepth", s.elevation_kb AS "ElevationKB",
s.elevation_df AS "ElevationDF", s.elevation_gl AS "ElevationGL", f.formation AS "FormationTD",
d.bit_diameter AS "BitDiameterCollar", 'inches'::character varying(6) AS "DiameterUnits", b.bht AS
"MeasuredTemperature", b.bht_corrected AS "CorrectedTemperature", 'C'::character varying(1) AS
"TemperatureUnits", b.time_since_circulation AS "TimeSinceCirculation", b.bht_correction_type_id AS
"CorrectionType", b.bht_depth AS "DepthOfMeasurement", b.drill_date AS "MeasurementDateTime",
'nil:missing'::character varying(16) AS "MeasurementSource", cs.casing_depth AS "CasingBottomDepthDriller",
cs.casing_top AS "CasingTopDepth", cs.casing_size_inches AS "CasingPipeDiameter", cs.casing_weight AS
"CasingWeight", 'lbs/ft'::character varying(6) AS "CasingWeightUnits", d.viscosity AS "Viscosity",

```

```

CASE
  WHEN s.contributor IS NULL AND s.original_source IS NULL THEN ((ds.datasource::text || ' '::text) ||
ds.datatype::text)::character varying
  WHEN s.contributor IS NOT NULL THEN s.contributor
  ELSE s.original_source
END AS "InformationSource", s.shape AS "Shape"
FROM core.surface_site s
JOIN core.bht_measurements b ON b.surface_site_id = s.id
LEFT JOIN core.datasets ds ON s.dataset_id = ds.id
LEFT JOIN core.ref_county rc ON s.county_id = rc.id
LEFT JOIN core.ref_state rs ON rc.state_id = rs.id
LEFT JOIN depth d ON d.surface_site_id = s.id
LEFT JOIN formation f ON f.surface_site_id = s.id
LEFT JOIN casing cs ON cs.surface_site_id = s.id
LEFT JOIN well_info wi ON wi.surface_site_id = s.id
LEFT JOIN well_dates wd ON wd.surface_site_id = s.id
WHERE b.estimated IS NOT TRUE;

```

Table "core.template_borehole_materialized"

Column	Type	Modifiers	Storage	Description
ObservationURI	text		extended	
WellName	character varying		extended	
APINo	character varying(18)		extended	
HeaderURI	text		extended	
OtherID	text		extended	
Label	text		extended	
Operator	character varying(128)		extended	
LeaseName	character varying(128)		extended	
LeaseOwner	character varying(128)		extended	
LeaseNo	text		extended	
SpudDate	text		extended	
EndedDrillingDate	text		extended	
welltype	character varying(128)		extended	
Status	character varying(64)		extended	
CommodityOfInterest	character varying		extended	
StatusDate	text		extended	
Field	character varying(128)		extended	
OtherLocationName	character varying(256)		extended	
County	character varying		extended	
State	character varying		extended	
PLSS_Meridians	character varying(64)		extended	
Township	character varying(5)		extended	
Range	character varying(5)		extended	
Section	character varying(10)		extended	
SectionPart	character varying(5)		extended	
LatDegree	numeric(10,7)		main	
LongDegree	numeric(10,7)		main	
SRS	character varying(6)		extended	
LocationUncertaintyStatement	character varying(128)		extended	
DrillerTotalDepth	double precision		plain	
DepthReferencePoint	character varying(1)		extended	
LengthUnits	character varying(1)		extended	
WellBoreShape	character varying(1)		extended	
TrueVerticalDepth	double precision		plain	

ElevationKB	double precision	plain
ElevationDF	double precision	plain
ElevationGL	double precision	plain
FormationTD	text	extended
BitDiameterCollar	double precision	plain
DiameterUnits	character varying(6)	extended
MeasuredTemperature	double precision	plain
CorrectedTemperature	double precision	plain
TemperatureUnits	character varying(1)	extended
TimeSinceCirculation	double precision	plain
CorrectionType	integer	plain
DepthOfMeasurement	double precision	plain
MeasurementDateTime	date	plain
MeasurementSource	character varying(16)	extended
CasingBottomDepthDriller	double precision	plain
CasingTopDepth	numeric	main
CasingPipeDiameter	double precision	plain
CasingWeight	double precision	plain
CasingWeightUnits	character varying(6)	extended
Viscosity	double precision	plain
InformationSource	character varying	extended
Shape	geometry	main
Has OIDs: no		

Fluid Flux Injection Disposal Content Model

View "core.template_fluidflux"

Column	Type	Modifiers	Storage	Description
ObservationURI	text		extended	
RecordName	text		extended	
FacilityURI	character varying(16)		extended	
Operator	integer		plain	
HeaderURI	text		extended	
WellName	character varying		extended	
APINo	character varying(18)		extended	
OtherID	text		extended	
County	character varying		extended	
State	character varying		extended	
Field	character varying(128)		extended	
LatDegreeWGS84	numeric(10,7)		main	
LongDegreeWGS84	numeric(10,7)		main	
SRS	character varying(6)		extended	
LocationUncertaintyStatement	character varying(128)		extended	
IntervalStartDateTime	date		plain	
IntervalEndDateTime	character varying(16)		extended	
DurationDays	character varying(16)		extended	
Permit	text		extended	
FluidType	text		extended	
FluidVolume	integer		plain	
FluidVolumeUnits	character varying(3)		extended	
Temperature_F	integer		plain	
Pressure_psi	integer		plain	
AggregationType	character varying(7)		extended	
MeasurementMethod	character varying(16)		extended	
ContactIntervalTop	integer		plain	

ContactIntervalBottom	integer	plain
Source	character varying	extended
MetadataURI	character varying(16)	extended
Shape	geometry	main

View definition:

```

WITH fluidh10 AS (
    SELECT fluid_h10.id, fluid_h10.surface_site_id, fluid_h10.dataset_id, fluid_h10.injection_id,
    fluid_h10.h10_date, fluid_h10.h10_avg_inj_pressure, fluid_h10.h10_max_inj_pressure,
    fluid_h10.h10_total_vol_bbl, fluid_h10.h10_total_vol_mcf
    FROM core.fluid_h10
    WHERE fluid_h10.h10_total_vol_bbl > 0 OR fluid_h10.h10_avg_inj_pressure > 0
), wi AS (
    SELECT well_info.surface_site_id, array_to_string(array_agg(DISTINCT well_info.commodity), '|')::text
    AS commodity, array_to_string(array_agg(DISTINCT well_info.well_id), '|')::text AS well_id,
    array_to_string(array_agg(DISTINCT well_info.permit), '|')::text AS permit
    FROM core.well_info
    GROUP BY well_info.surface_site_id
), wd AS (
    SELECT well_dates.surface_site_id, max(well_dates.drl_compl_date) AS drl_compl_date
    FROM core.well_dates
    GROUP BY well_dates.surface_site_id
)
SELECT 'https://geothermal.smu.edu/fluidflux/'::text || s.id AS "ObservationURI", 'Fluid Flux for well '::text ||
COALESCE(s.site_name, s.api)::text AS "RecordName", 'nil:missing'::character varying(16) AS "FacilityURI",
fluid.oper AS "Operator", 'https://geothermal.smu.edu/header/'::text || s.id AS "HeaderURI",
COALESCE(s.site_name, s.api) AS "WellName", s.api AS "APINo", COALESCE(s.other_ids, ' '::character
varying)::text || COALESCE(wi.well_id, ' '::text) AS "OtherID", COALESCE(rc.county_name,
'nil:missing'::character varying) AS "County", COALESCE(rs.state_name, 'nil:missing'::character varying) AS
"State", s.field AS "Field", s.latitude AS "LatDegreeWGS84", s.longitude AS "LongDegreeWGS84",
'WGS84'::character varying(6) AS "SRS", s.location_notes AS "LocationUncertaintyStatement",
COALESCE(fluidh10.h10_date, '1900-01-01'::date) AS "IntervalStartDateTime", 'nil:missing'::character
varying(16) AS "IntervalEndDateTime", 'nil:missing'::character varying(16) AS "DurationDays",
COALESCE(wi.permit, 'nil:missing'::text) AS "Permit",
CASE
    WHEN fluid.sw = 'Y'::bpchar THEN 'salt water'::text
    WHEN fluid.fw = 'Y'::bpchar THEN 'fresh water'::text
    WHEN fluid.bw = 'Y'::bpchar THEN 'brackish water'::text
    WHEN fluid.air = 'Y'::bpchar THEN 'air'::text
    WHEN fluid.gas = 'Y'::bpchar THEN 'gas'::text
    WHEN fluid.lpg = 'Y'::bpchar THEN 'lpg'::text
    WHEN fluid.co2 = 'Y'::bpchar THEN 'co2'::text
    WHEN fluid.polymer = 'Y'::bpchar THEN 'polymer'::text
    ELSE 'nil:missing'::text
END AS "FluidType", COALESCE(fluidh10.h10_total_vol_bbl, (-99999)) AS "FluidVolume",
'bbl'::character varying(3) AS "FluidVolumeUnits", (-99999) AS "Temperature_F",
COALESCE(fluidh10.h10_avg_inj_pressure, (-99999)) AS "Pressure_psi", 'average'::character varying(7) AS
"AggregationType", 'nil:missing'::character varying(16) AS "MeasurementMethod",
COALESCE(fluid.top_inj_zone, (-99999)) AS "ContactIntervalTop", COALESCE(fluid.bot_inj_zone, (-99999))
AS "ContactIntervalBottom",
CASE
    WHEN s.contributor IS NULL AND s.original_source IS NULL THEN ((ds.datasource::text || ' '::text) ||
ds.datatype::text)::character varying
    WHEN s.contributor IS NOT NULL THEN s.contributor
    ELSE s.original_source
END AS "Source", 'nil:missing'::character varying(16) AS "MetadataURI", s.shape AS "Shape"
FROM core.surface_site s

```

```

JOIN core.fluid fluid ON fluid.surface_site_id = s.id
JOIN fluidh10 ON fluid.id = fluidh10.injection_id
LEFT JOIN core.datasets ds ON s.dataset_id = ds.id
LEFT JOIN core.ref_county rc ON s.county_id = rc.id
LEFT JOIN core.ref_state rs ON rc.state_id = rs.id
LEFT JOIN wi ON wi.surface_site_id = s.id
LEFT JOIN wd ON wd.surface_site_id = s.id;

```

Table "core.template_fluidflux_materialized"

Column	Type	Modifiers	Storage	Description
ObservationURI	text		extended	
RecordName	text		extended	
FacilityURI	character varying(16)		extended	
Operator	integer		plain	
HeaderURI	text		extended	
WellName	character varying		extended	
APINo	character varying(18)		extended	
OtherID	text		extended	
County	character varying		extended	
State	character varying		extended	
Field	character varying(128)		extended	
LatDegreeWGS84	numeric(10,7)		main	
LongDegreeWGS84	numeric(10,7)		main	
SRS	character varying(6)		extended	
LocationUncertaintyStatement	character varying(128)		extended	
IntervalStartDateTime	date		plain	
IntervalEndDateTime	character varying(16)		extended	
DurationDays	character varying(16)		extended	
Permit	text		extended	
FluidType	text		extended	
FluidVolume	integer		plain	
FluidVolumeUnits	character varying(3)		extended	
Temperature_F	integer		plain	
Pressure_psi	integer		plain	
AggregationType	character varying(7)		extended	
MeasurementMethod	character varying(16)		extended	
ContactIntervalTop	integer		plain	
ContactIntervalBottom	integer		plain	
Source	character varying		extended	
MetadataURI	character varying(16)		extended	
Shape	geometry		main	
Has OIDs: no				

Heat Flow Observation Content Model

View "core.template_heatflow"

Column	Type	Modifiers	Storage	Description
ObservationURI	text		extended	
ObservationName	text		extended	
Label	text		extended	
HeatFlowType	text		extended	
HeaderURI	text		extended	

APINo	character varying(18)	extended
WellName	character varying	extended
OtherID	text	extended
Owner	character varying(128)	extended
LeaseName	character varying(128)	extended
Field	character varying(128)	extended
County	character varying	extended
State	character varying	extended
LatDegreeWGS84	numeric(10,7)	main
LongDegreeWGS84	numeric(10,7)	main
LocationUncertaintyStatement	character varying(128)	extended
PLSS_Meridians	character varying(64)	extended
Township	character varying(5)	extended
Range	character varying(5)	extended
Section_	character varying(10)	extended
SectionPart	character varying(5)	extended
TectonicProvince	character varying(128)	extended
OtherLocationName	text	extended
PublicationSource	character varying(7)	extended
AdditionalSources	character varying(16)	extended
WellType	character varying	extended
WellDrillDate	text	extended
WellDrillDateType	character varying(16)	extended
Elevation	double precision	plain
ElevationType	text	extended
DepthUnits	character varying(1)	extended
DrillerTotalDepth	double precision	plain
TimeSinceCirculation	double precision	plain
TemperatureUnits	character varying(1)	extended
TemperatureMeasureType	character varying(7)	extended
SurfaceTemperature	double precision	plain
BHT	double precision	plain
BHTCorrectedValue	double precision	plain
BHTCorrectionType	character varying(16)	extended
EqLog_MeasurementDate	date	plain
EqLog_TotalDepth	double precision	plain
EqLogTemperatureTop	double precision	plain
EqLogTemperatureMax	numeric(10,7)	main
EqLogTemperatureBottom	double precision	plain
SiteHeatFlow	double precision	plain
SiteValueMethod	character varying(7)	extended
Intervals	integer	plain
IntervalTopDepth	double precision	plain
IntervalBottomDepth	double precision	plain
ThermalConductivityUnits	character varying(64)	extended
IntervalThermalConductivity	double precision	plain
SE_Conductivity	double precision	plain
ConductivitySampleType	character varying(128)	extended
ConductivityNumSamples	integer	plain
ConductivityMeasuredURI	text	extended
GradientUnits	character varying(4)	extended
IntervalUncorrectedGradient	double precision	plain
SE_UncorrectedGradient	double precision	plain
IntervalCorrectedGradient	double precision	plain
SE_CorrectedGradient	double precision	plain
HeatFlowUnits	character varying(5)	extended

IntervalUncorrectedHeatFlow	double precision	plain
SE_UncorrectedHeatflow	double precision	plain
IntervalCorrectedHeatFlow	double precision	plain
SE_CorrectedHeatFlow	double precision	plain
IntervalHF_QualityCode	character varying	extended
HeatGenerationUnits	character varying(5)	extended
HeatGeneration	double precision	plain
HG_NumberSamples	integer	plain
WaterTableTop	double precision	plain
WaterTableTemperature	double precision	plain
LithologyInformation	text	extended
FormationName	text	extended
Notes	text	extended
Shape	geometry	main

View definition:

```

WITH formation AS (
    SELECT formation.surface_site_id, array_to_string(array_agg(DISTINCT (formation.formation::text || '
depth='::text) ||
    CASE
        WHEN formation.formation_depth IS NULL THEN 'NA'::text
        ELSE "''::text || formation.formation_depth
    END), ' '::text) AS formation
    FROM core.formation
    GROUP BY formation.surface_site_id
), aquifier AS (
    SELECT aquifier.surface_site_id, max(aquifier.aquifier_temperature) AS aquifier_temperature,
max(aquifier.water_table_depth) AS water_table_depth
    FROM core.aquifier
    GROUP BY aquifier.surface_site_id
), depth AS (
    SELECT depth.surface_site_id, max(depth.true_vertical_depth) AS true_vertical_depth, max(depth.depth)
AS depth, array_to_string(array_agg(DISTINCT depth.depth_date), ' '::text) AS depth_date
    FROM core.depth
    GROUP BY depth.surface_site_id
)
SELECT 'https://geothermal.smu.edu/obervation/heatflow/'::text || h.id AS "ObservationURI", 'Heat flow
observation for well '::text || COALESCE(s.site_name, s.api)::text AS "ObservationName", 'Heat flow observation
for well '::text || COALESCE(s.site_name, s.api)::text AS "Label",
    CASE
        WHEN h.interval_order IS NULL AND h.interval_top_depth IS NULL AND h.interval_bottom_depth IS
NULL OR h.interval_order::text = 'a'::text THEN 'Site'::text
        ELSE 'Interval'::text
    END AS "HeatFlowType", 'https://geothermal.smu.edu/header/'::text || s.id AS "HeaderURI", s.api AS
"APINo", COALESCE(s.site_name, s.api) AS "WellName", (((s.other_ids::text || ' '::text) || 'SourceID and
Interval: '::text) || s.source_id::text) || ' '::text || h.interval_order::text AS "OtherID", s.site_owner AS "Owner",
s.lease_name AS "LeaseName", s.field AS "Field", COALESCE(rc.county_name, 'nil:missing'::character varying)
AS "County", COALESCE(rs.state_name, 'nil:missing'::character varying) AS "State", s.latitude AS
"LatDegreeWGS84", s.longitude AS "LongDegreeWGS84", s.location_notes AS "LocationUncertaintyStatement",
s.plss_meridians AS "PLSS_Meridians", s.township AS "Township", s.range AS "Range", s.section AS
"Section_", s.section_part AS "SectionPart", s.tectonic_province AS "TectonicProvince",
(s.other_location_name::text || ' ams: '::text) || s.ams::text AS "OtherLocationName", 'nil:NA'::character varying(7)
AS "PublicationSource", s.original_source AS "AdditionalSources",
    CASE
        WHEN s.well_type IS NULL OR length(s.well_type::text) = 0 THEN 'nil:missing'::character varying
        ELSE s.well_type
    END AS "WellType", "''::text || bht.drill_date AS "WellDrillDate", bht.date_type AS "WellDrillDateType",

```

```

CASE
  WHEN s.elevation_kb IS NOT NULL THEN s.elevation_kb
  WHEN s.elevation_df IS NOT NULL THEN s.elevation_df
  WHEN s.elevation_gl IS NOT NULL THEN s.elevation_gl
  ELSE NULL::double precision
END AS "Elevation",
CASE
  WHEN s.elevation_kb IS NOT NULL THEN 'KB'::text
  WHEN s.elevation_df IS NOT NULL THEN 'DF'::text
  WHEN s.elevation_gl IS NOT NULL THEN 'GL'::text
  ELSE NULL::text
END AS "ElevationType", 'm'::character varying(1) AS "DepthUnits", COALESCE(d.depth, (-
99999)::double precision) AS "DrillerTotalDepth", bht.time_since_circulation AS "TimeSinceCirculation",
'C'::character varying(1) AS "TemperatureUnits", 'BHT'::character varying(7) AS "TemperatureMeasureType",
s.surface_temperature AS "SurfaceTemperature", bht.bht AS "BHT", bht.bht_corrected AS "BHTCorrectedValue",
bht.bht_correction_type AS "BHTCorrectionType", bht.drill_date AS "EqLog_MeasurementDate", bht.bht_depth
AS "EqLog_TotalDepth", s.surface_temperature AS "EqLogTemperatureTop", s.maxtemp AS
"EqLogTemperatureMax", bht.bht AS "EqLogTemperatureBottom", COALESCE(s.average_heatflow, (-
99999)::double precision) AS "SiteHeatFlow", 'nil:NA'::character varying(7) AS "SiteValueMethod", (-99999) AS
"Intervals", COALESCE(h.interval_top_depth, (-99999)::double precision) AS "IntervalTopDepth",
COALESCE(h.interval_bottom_depth, (-99999)::double precision) AS "IntervalBottomDepth", 'Watt per meter
per Kelvin'::character varying(64) AS "ThermalConductivityUnits", tc.conductivity AS
"IntervalThermalConductivity", tc.conductivity_se AS "SE_Conductivity", tc.lithology_info AS
"ConductivitySampleType", tc.conductivity_nb_samples AS "ConductivityNumSamples",
CASE
  WHEN tc.id IS NULL THEN 'nil:NA'::text
  ELSE 'https://geothermal.smu.edu/obervation/thermal_conductivity/'::text || tc.id
END AS "ConductivityMeasuredURI", 'C/km'::character varying(4) AS "GradientUnits",
COALESCE(h.gradient, (-99999)::double precision) AS "IntervalUncorrectedGradient", h.gradient_se AS
"SE_UncorrectedGradient", h.gradient_corrected AS "IntervalCorrectedGradient", h.gradient_corrected_se AS
"SE_CorrectedGradient", 'mW/m2'::character varying(5) AS "HeatFlowUnits", COALESCE(h.heat_flow, (-
99999)::double precision) AS "IntervalUncorrectedHeatFlow", h.heat_flow_se AS "SE_UncorrectedHeatflow",
COALESCE(h.heat_flow_corrected, (-99999)::double precision) AS "IntervalCorrectedHeatFlow",
h.heat_flow_corrected_se AS "SE_CorrectedHeatFlow", COALESCE(h.quality, ''::character varying) AS
"IntervalHF_QualityCode", 'µW/m3'::character varying(5) AS "HeatGenerationUnits", h.heat_generation AS
"HeatGeneration", h.heat_generation_nb_samples AS "HG_NumberSamples", a.water_table_depth AS
"WaterTableTop", a.aquifer_temperature AS "WaterTableTemperature", (((('BasementDepth_m'::text ||
COALESCE(''::text || s.basement_depth, 'NA'::text)) || ' '::text) || s.lithology::text) || ' '::text) ||
tc.lithology_info::text AS "LithologyInformation", f.formation AS "FormationName", (('WellResistance'::text ||
tc.resistance) || ' '::text) || s.comment::text AS "Notes", s.shape AS "Shape"
FROM core.surface_site s
JOIN core.heat_flow h ON h.surface_site_id = s.id
LEFT JOIN core.ref_county rc ON s.county_id = rc.id
LEFT JOIN core.ref_state rs ON rc.state_id = rs.id
LEFT JOIN formation f ON f.surface_site_id = s.id
LEFT JOIN aquifier a ON a.surface_site_id = s.id
LEFT JOIN depth d ON d.surface_site_id = s.id
LEFT JOIN core.bht_measurements bht ON bht.surface_site_id = s.id AND bht.id = h.bht_measurements_id
LEFT JOIN core.thermal_conductivity tc ON tc.surface_site_id = s.id AND tc.id = h.thermal_conductivity_id;

```

Table "core.template_heatflow_materialized"

Column	Type	Modifiers	Storage	Description
ObservationURI	text		extended	
ObservationName	text		extended	

Label	text	extended
HeatFlowType	text	extended
HeaderURI	text	extended
APINo	character varying(18)	extended
WellName	character varying	extended
OtherID	text	extended
Owner	character varying(128)	extended
LeaseName	character varying(128)	extended
Field	character varying(128)	extended
County	character varying	extended
State	character varying	extended
LatDegreeWGS84	numeric(10,7)	main
LongDegreeWGS84	numeric(10,7)	main
LocationUncertaintyStatement	character varying(128)	extended
PLSS_Meridians	character varying(64)	extended
Township	character varying(5)	extended
Range	character varying(5)	extended
Section_	character varying(10)	extended
SectionPart	character varying(5)	extended
TectonicProvince	character varying(128)	extended
OtherLocationName	text	extended
PublicationSource	character varying(7)	extended
AdditionalSources	character varying(16)	extended
WellType	character varying	extended
WellDrillDate	text	extended
WellDrillDateType	character varying(16)	extended
Elevation	double precision	plain
ElevationType	text	extended
DepthUnits	character varying(1)	extended
DrillerTotalDepth	double precision	plain
TimeSinceCirculation	double precision	plain
TemperatureUnits	character varying(1)	extended
TemperatureMeasureType	character varying(7)	extended
SurfaceTemperature	double precision	plain
BHT	double precision	plain
BHTCorrectedValue	double precision	plain
BHTCorrectionType	character varying(16)	extended
EqLog_MeasurementDate	date	plain
EqLog_TotalDepth	double precision	plain
EqLogTemperatureTop	double precision	plain
EqLogTemperatureMax	numeric(10,7)	main
EqLogTemperatureBottom	double precision	plain
SiteHeatFlow	double precision	plain
SiteValueMethod	character varying(7)	extended
Intervals	integer	plain
IntervalTopDepth	double precision	plain
IntervalBottomDepth	double precision	plain
ThermalConductivityUnits	character varying(64)	extended
IntervalThermalConductivity	double precision	plain
SE_Conductivity	double precision	plain
ConductivitySampleType	character varying(128)	extended
ConductivityNumSamples	integer	plain
ConductivityMeasuredURI	text	extended
GradientUnits	character varying(4)	extended
IntervalUncorrectedGradient	double precision	plain
SE_UncorrectedGradient	double precision	plain

IntervalCorrectedGradient	double precision	plain
SE_CorrectedGradient	double precision	plain
HeatFlowUnits	character varying(5)	extended
IntervalUncorrectedHeatFlow	double precision	plain
SE_UncorrectedHeatflow	double precision	plain
IntervalCorrectedHeatFlow	double precision	plain
SE_CorrectedHeatFlow	double precision	plain
IntervalHF_QualityCode	character varying	extended
HeatGenerationUnits	character varying(5)	extended
HeatGeneration	double precision	plain
HG_NumberSamples	integer	plain
WaterTableTop	double precision	plain
WaterTableTemperature	double precision	plain
LithologyInformation	text	extended
FormationName	text	extended
Notes	text	extended
Shape	geometry	main
Has OIDs: no		

Radiogenic Heat Production Content Model

View "core.template_radiogenicheat"

Column	Type	Modifiers	Storage	Description
ObservationURI	text		extended	
ObservationName	text		extended	
SpecimenLabel	character varying		extended	
GeologicUnitName	text		extended	
LithologyTerm	character varying(255)		extended	
RockName	character varying(64)		extended	
SamplingFeatureName	character varying(16)		extended	
SampleCollectionDate	date		plain	
TectonicProvince	character varying(128)		extended	
County	character varying		extended	
State	character varying		extended	
PLSS_Meridians	character varying(64)		extended	
Township	character varying(5)		extended	
Range	character varying(5)		extended	
Section_	character varying(10)		extended	
SectionPart	character varying(5)		extended	
LatDegree	numeric(10,7)		main	
LongDegree	numeric(10,7)		main	
SRS	character varying(6)		extended	
LocationUncertaintyStatement	character varying		extended	
SampleDepth_m	double precision		plain	
SampleMeasurementDate	date		plain	
SampleType	character varying		extended	
SampleMass_kg	double precision		plain	
Laboratory	character varying(64)		extended	
	character varying		extended	
RadioactivityMeasurementDevice				
MeasurementProcedure	character varying(64)		extended	
MeasurementCount	character varying(16)		extended	
Uranium	double precision		plain	
Thorium	double precision		plain	
Potassium	double precision		plain	

HeatProduction_mWm3	double precision	plain
HeatProductionUncertainty	text	extended
Notes	character varying(128)	extended
Source	character varying	extended
Shape	geometry	main

View definition:

```
WITH formation AS (
    SELECT formation.surface_site_id, array_to_string(array_agg(DISTINCT (formation.formation::text || '
depth='::text) ||
    CASE
        WHEN formation.formation_depth IS NULL THEN 'NA'::text
        ELSE "::text || formation.formation_depth
    END), '|'::text) AS formation
    FROM core.formation
    GROUP BY formation.surface_site_id
)
SELECT 'https://geothermal.smu.edu/obervation/radiogenicheat/'::text || rad.id AS "ObservationURI", 'Radiogenic
Heat Production for well '::text || COALESCE(s.site_name, s.api, ((s.latitude || ' '::text) || s.longitude)::character
varying)::text AS "ObservationName", COALESCE(rad.specimen_label, 'nil:missing'::character varying) AS
"SpecimenLabel", f.formation AS "GeologicUnitName", s.lithology AS "LithologyTerm", rad.rock AS
"RockName", 'nil:missing'::character varying(16) AS "SamplingFeatureName", tc.sample_collection_date AS
"SampleCollectionDate", s.tectonic_province AS "TectonicProvince", COALESCE(rc.county_name,
'nil:missing'::character varying) AS "County", COALESCE(rs.state_name, 'nil:missing'::character varying) AS
"State", s.plss_meridians AS "PLSS_Meridians", s.township AS "Township", s.range AS "Range", s.section AS
"Section_", s.section_part AS "SectionPart", s.latitude AS "LatDegree", s.longitude AS "LongDegree",
'WGS84'::character varying(6) AS "SRS", COALESCE(s.location_notes, 'nil:missing'::character varying) AS
"LocationUncertaintyStatement", tc.sample_depth AS "SampleDepth_m", tc.sample_measurement_date AS
"SampleMeasurementDate", COALESCE(tc.sample_type, 'nil:missing'::character varying) AS "SampleType", (-
1)::double precision AS "SampleMass_kg", rad.laboratory AS "Laboratory",
COALESCE(rad.measurement_device, 'nil:missing'::character varying) AS "RadioactivityMeasurementDevice",
rad.measurement_procedure AS "MeasurementProcedure", 'nil:missing'::character varying(16) AS
"MeasurementCount", COALESCE(rad.uranium, (-1)::double precision) AS "Uranium", COALESCE(rad.thorium,
(-1)::double precision) AS "Thorium", COALESCE(rad.potassium, (-1)::double precision) AS "Potassium",
COALESCE(h.heat_generation, (-1)::double precision) AS "HeatProduction_mWm3",
COALESCE(h.heat_generation_se::text, 'nil:missing'::text) AS "HeatProductionUncertainty", rad.comments AS
"Notes",
CASE
    WHEN s.contributor IS NULL AND s.original_source IS NULL THEN ((ds.datasource::text || ' '::text) ||
ds.datatype::text)::character varying
    WHEN s.contributor IS NOT NULL THEN s.contributor
    ELSE s.original_source
END AS "Source", s.shape AS "Shape"
FROM core.surface_site s
JOIN core.radioactivity rad ON rad.surface_site_id = s.id
LEFT JOIN core.ref_county rc ON s.county_id = rc.id
LEFT JOIN core.ref_state rs ON rc.state_id = rs.id
LEFT JOIN core.datasets ds ON s.dataset_id = ds.id
LEFT JOIN formation f ON f.surface_site_id = s.id
LEFT JOIN core.heat_flow h ON h.surface_site_id = s.id AND h.radioactivity_id = rad.id
LEFT JOIN core.thermal_conductivity tc ON tc.surface_site_id = s.id AND tc.id = h.thermal_conductivity_id;
```

Table "core.template_radiogenicheat_materialized"

Column	Type	Modifiers	Storage	Description
ObservationURI	text		extended	

ObservationName	text	extended
SpecimenLabel	character varying	extended
GeologicUnitName	text	extended
LithologyTerm	character varying(255)	extended
RockName	character varying(64)	extended
SamplingFeatureName	character varying(16)	extended
SampleCollectionDate	date	plain
TectonicProvince	character varying(128)	extended
County	character varying	extended
State	character varying	extended
PLSS_Meridians	character varying(64)	extended
Township	character varying(5)	extended
Range	character varying(5)	extended
Section_	character varying(10)	extended
SectionPart	character varying(5)	extended
LatDegree	numeric(10,7)	main
LongDegree	numeric(10,7)	main
SRS	character varying(6)	extended
LocationUncertaintyStatement	character varying	extended
SampleDepth_m	double precision	plain
SampleMeasurementDate	date	plain
SampleType	character varying	extended
SampleMass_kg	double precision	plain
Laboratory	character varying(64)	extended
RadioactivityMeasurementDevice	character varying	extended
MeasurementProcedure	character varying(64)	extended
MeasurementCount	character varying(16)	extended
Uranium	double precision	plain
Thorium	double precision	plain
Potassium	double precision	plain
HeatProduction_mWm3	double precision	plain
HeatProductionUncertainty	text	extended
Notes	character varying(128)	extended
Source	character varying	extended
Shape	geometry	main
Has OIDs: no		

Geologic Reservoir Content Model

View "core.template_reservoir"

Column	Type	Modifiers	Storage	Description
ReservoirURI	character varying		extended	
ReservoirName	character varying		extended	
Label	character varying		extended	
OtherID	character varying(255)		extended	
OtherName	character varying(255)		extended	
Description	character varying(255)		extended	
County	character varying		extended	
State	character varying		extended	
LocationName	character varying(255)		extended	
LatDegree	double precision		plain	
LongDegree	double precision		plain	
SRS	character varying(255)		extended	
LocationUncertaintyStatement	character varying		extended	
TemperatureMax_F	double precision		plain	

TemperatureMin_F	double precision	plain
DepthToTop_ft	double precision	plain
DepthToBottom_ft	double precision	plain
DominantFormationName	character varying(255)	extended
OtherFormationName	character varying(255)	extended
DominantLithologyName	character varying(255)	extended
OtherLithologyName	character varying(255)	extended
Area_acres	double precision	plain
Width_ft	double precision	plain
Length_ft	double precision	plain
SandVolume_cft	double precision	plain
BlockVolume_cft	double precision	plain
PercentSand	double precision	plain
AverageNetSandThickness_ft	double precision	plain
WellCount	integer	plain
FaultCount	integer	plain
Faulted	character varying(255)	extended
FaultType	character varying(255)	extended
FaultHydraulicCharacter	character varying(255)	extended
FirstProductionDate	date	plain
LastProductionDate	date	plain
YearsInProduction	integer	plain
MonthlyProduction_bbl	double precision	plain
ProductionReportDate	date	plain
AverageMonthlyWaterProduction_bbl	double precision	plain
MinimumMonthlyWaterProduction_bbl	double precision	plain
MaximumMonthlyWaterProduction_bbl	double precision	plain
WaterInjection_bbl	double precision	plain
WaterSaturation_pct	double precision	plain
WaterTDS_mgl	double precision	plain
AverageFieldWidePressure_psi	double precision	plain
AverageFieldWideGradient_degfft	double precision	plain
PorosityMax_pct	double precision	plain
PorosityMin_pct	double precision	plain
PermeabilityMax_mD	double precision	plain
PermeabilityMin_mD	double precision	plain
CementationMineralogy	character varying(255)	extended
Trap	character varying(255)	extended
Drive	character varying(255)	extended
WaterDensity	double precision	plain
CommodityOfInterest	character varying(255)	extended
Stimulation	character varying(255)	extended
AveragePerforatedThickness_ft	double precision	plain
RelatedResource	character varying(255)	extended
Source	character varying(1023)	extended
MetadataURI	character varying(255)	extended
Shape	geometry	main

View definition:

```

SELECT COALESCE(r.reservoiruri, 'nil:missing'::character varying) AS "ReservoirURI",
COALESCE(r.reservoirname, 'nil:missing'::character varying) AS "ReservoirName",
CASE
    WHEN r.label IS NULL OR length(r.label::text) = 0 THEN 'nil:missing'::character varying
    ELSE r.label
END AS "Label", r.otherid AS "OtherID", r.othername AS "OtherName", r.description AS "Description",
COALESCE(r.county, 'nil:missing'::character varying) AS "County", COALESCE(r.state, 'nil:missing'::character

```

```

varying) AS "State", r.locationname AS "LocationName", r.latdegree AS "LatDegree", r.longdegree AS
"LongDegree", r.srs AS "SRS",
CASE
    WHEN r.locationuncertaintystatement IS NULL OR length(r.locationuncertaintystatement::text) = 0 THEN
nil:missing'::character varying
    ELSE r.locationuncertaintystatement
END AS "LocationUncertaintyStatement", r.temperaturemax_f AS "TemperatureMax_F", r.temperaturemin_f
AS "TemperatureMin_F", r.depthtotop_ft AS "DepthToTop_ft", r.depthtobottom_ft AS "DepthToBottom_ft",
r.dominantformationname AS "DominantFormationName", r.otherformationname AS "OtherFormationName",
r.dominantlithologyname AS "DominantLithologyName", r.otherlithologyname AS "OtherLithologyName",
r.area_acres AS "Area_acres", r.width_ft AS "Width_ft", r.length_ft AS "Length_ft", r.sandvolume_cft AS
"SandVolume_cft", r.blockvolume_cft AS "BlockVolume_cft", r.percentsand AS "PercentSand",
r.averagenetsandthickness_ft AS "AverageNetSandThickness_ft", r.wellcount AS "WellCount", r.faultcount AS
"FaultCount", r.faulted AS "Faulted", r.faulttype AS "FaultType", r.faulthydrauliccharacter AS
"FaultHydraulicCharacter", r.firstproductiondate AS "FirstProductionDate", r.lastproductiondate AS
"LastProductionDate", r.yearsinproduction AS "YearsInProduction", r.monthlyproduction_bbl AS
"MonthlyProduction_bbl", r.productionreportdate AS "ProductionReportDate",
r.averagemonthlywaterproduction_bbl AS "AverageMonthlyWaterProduction_bbl",
r.minimummonthlywaterproduction_bbl AS "MinimumMonthlyWaterProduction_bbl",
r.maximummonthlywaterproduction_bbl AS "MaximumMonthlyWaterProduction_bbl", r.waterinjection_bbl AS
"WaterInjection_bbl", r.watersaturation_pct AS "WaterSaturation_pct", r.watertds_mgl AS "WaterTDS_mgl",
r.averagefieldwidepressure_psi AS "AverageFieldWidePressure_psi", r.averagefieldwidegradient_degfft AS
"AverageFieldWideGradient_degfft", r.porositymax_pct AS "PorosityMax_pct", r.porositymin_pct AS
"PorosityMin_pct", r.permeabilitymax_md AS "PermeabilityMax_mD", r.permeabilitymin_md AS
"PermeabilityMin_mD", r.cementationmineralogy AS "CementationMineralogy", r.trap AS "Trap", r.drive AS
"Drive", r.watertds_mgl AS "WaterDensity", r.commodityofinterest AS "CommodityOfInterest", r.stimulation AS
"Stimulation", r.averageperforatedthickness_ft AS "AveragePerforatedThickness_ft", r.relatedresource AS
"RelatedResource", r.source AS "Source", r.metadatauri AS "MetadataURI", r.shape AS "Shape"
FROM core.reservoir r;

```

Table "core.template_reservoir_materialized"

Column	Type	Modifiers	Storage	Description
ReservoirURI	character varying		extended	
ReservoirName	character varying		extended	
Label	character varying		extended	
OtherID	character varying(255)		extended	
OtherName	character varying(255)		extended	
Description	character varying(255)		extended	
County	character varying		extended	
State	character varying		extended	
LocationName	character varying(255)		extended	
LatDegree	double precision		plain	
LongDegree	double precision		plain	
SRS	character varying(255)		extended	
LocationUncertaintyStatement	character varying		extended	
TemperatureMax_F	double precision		plain	
TemperatureMin_F	double precision		plain	
DepthToTop_ft	double precision		plain	
DepthToBottom_ft	double precision		plain	
DominantFormationName	character varying(255)		extended	
OtherFormationName	character varying(255)		extended	
DominantLithologyName	character varying(255)		extended	
OtherLithologyName	character varying(255)		extended	
Area_acres	double precision		plain	

Width_ft	double precision	plain
Length_ft	double precision	plain
SandVolume_cft	double precision	plain
BlockVolume_cft	double precision	plain
PercentSand	double precision	plain
AverageNetSandThickness_ft	double precision	plain
WellCount	integer	plain
FaultCount	integer	plain
Faulted	character varying(255)	extended
FaultType	character varying(255)	extended
FaultHydraulicCharacter	character varying(255)	extended
FirstProductionDate	date	plain
LastProductionDate	date	plain
YearsInProduction	integer	plain
MonthlyProduction_bbl	double precision	plain
ProductionReportDate	date	plain
AverageMonthlyWaterProduction_bbl	double precision	plain
MinimumMonthlyWaterProduction_bbl	double precision	plain
MaximumMonthlyWaterProduction_bbl	double precision	plain
WaterInjection_bbl	double precision	plain
WaterSaturation_pct	double precision	plain
WaterTDS_mgl	double precision	plain
AverageFieldWidePressure_psi	double precision	plain
AverageFieldWideGradient_degfft	double precision	plain
PorosityMax_pct	double precision	plain
PorosityMin_pct	double precision	plain
PermeabilityMax_mD	double precision	plain
PermeabilityMin_mD	double precision	plain
CementationMineralogy	character varying(255)	extended
Trap	character varying(255)	extended
Drive	character varying(255)	extended
WaterDensity	double precision	plain
CommodityOfInterest	character varying(255)	extended
Stimulation	character varying(255)	extended
AveragePerforatedThickness_ft	double precision	plain
RelatedResource	character varying(255)	extended
Source	character varying(1023)	extended
MetadataURI	character varying(255)	extended
Shape	geometry	main
Has OIDs: no		

Thermal Conductivity Observation Content Model

View "core.template_thermalconductivity"

Column	Type	Modifiers	Storage	Description
ObservationURI	text		extended	
ObservationName	text		extended	
Label	text		extended	
OtherID	text		extended	
SampleDepth	double precision		plain	
MeasurementUnits	character varying(1)		extended	
SiteLocationURI	text		extended	
SiteLocationName	character varying		extended	
County	character varying		extended	
State	character varying		extended	

LatDegreeWGS84	numeric(10,7)	main
LongDegreeWGS84	numeric(10,7)	main
SRS	character varying(6)	extended
LocationUncertaintyStatement	character varying	extended
SampleCollectionDate	date	plain
SampleMeasurementDate	date	plain
SampleType	character varying	extended
SampleShape	character varying	extended
SampleDimensionsHeight	double precision	plain
SampleDimensionsWidth	double precision	plain
SampleDimensionsLength	double precision	plain
SampleDimensionsUnits	character varying(2)	extended
SampleDensity	double precision	plain
UncertaintyDensity	double precision	plain
Laboratory	character varying(32)	extended
MeasurementDevice	character varying	extended
MeasurementProcedure	character varying(128)	extended
MeasurementPressure	double precision	plain
SampleTemperature	double precision	plain
MeasurementRun	character varying(16)	extended
SaturatedSampleConductivity	double precision	plain
SiteThermalConductivity	double precision	plain
UncertaintyTC	double precision	plain
UnitsTC	character varying(64)	extended
GeologicFormationName	text	extended
GeologicFormationURI	character varying(16)	extended
GeologicFormationType	text	extended
DatumElevation	character varying(16)	extended
Elevation	double precision	plain
Permeability	double precision	plain
Porosity	double precision	plain
Notes	character varying(128)	extended
Source	character varying	extended
Shape	geometry	main

View definition:

```

WITH formation AS (
    SELECT formation.surface_site_id, array_to_string(array_agg(DISTINCT formation.formation), '::text') AS
formation, array_to_string(array_agg(DISTINCT formation.formation_type), '::text') AS formation_type
    FROM core.formation
    GROUP BY formation.surface_site_id
), depth AS (
    SELECT depth.surface_site_id, max(depth.true_vertical_depth) AS true_vertical_depth, max(depth.depth)
AS depth, array_to_string(array_agg(DISTINCT depth.depth_date), '::text') AS depth_date
    FROM core.depth
    GROUP BY depth.surface_site_id
), wellinfo AS (
    SELECT well_info.surface_site_id, array_to_string(array_agg(DISTINCT well_info.commodity), '::text')
AS commodity, array_to_string(array_agg(DISTINCT well_info.well_id), '::text') AS well_id
    FROM core.well_info
    GROUP BY well_info.surface_site_id
)
SELECT 'https://geothermal.smu.edu/obervation/conductivity/'::text || tc.id AS "ObservationURI", "Thermal
conductivity observation for well '::text || COALESCE(s.site_name, s.api)::text AS "ObservationName", "Thermal
conductivity observation for well '::text || COALESCE(s.site_name, s.api)::text AS "Label",
COALESCE(s.other_ids, ' '::character varying)::text || COALESCE(wi.well_id, ' '::text) AS "OtherID",
COALESCE(tc.sample_depth, (-99999)::double precision) AS "SampleDepth", 'm '::character varying(1) AS

```



```

"MeasurementUnits", 'https://geothermal.smu.edu/header/::text || s.id AS "SiteLocationURI",
COALESCE(s.site_name, s.api) AS "SiteLocationName", COALESCE(rc.county_name, 'nil:missing':character
varying) AS "County", COALESCE(rs.state_name, 'nil:missing':character varying) AS "State", s.latitude AS
"LatDegreeWGS84", s.longitude AS "LongDegreeWGS84", 'WGS84':character varying(6) AS "SRS",
COALESCE(s.location_notes, 'nil:missing':character varying) AS "LocationUncertaintyStatement",
tc.sample_collection_date AS "SampleCollectionDate", tc.sample_measurement_date AS
"SampleMeasurementDate", COALESCE(tc.sample_type, 'nil:missing':character varying) AS "SampleType",
COALESCE(tc.sample_shape, 'nil:missing':character varying) AS "SampleShape", tc.sample_height AS
"SampleDimensionsHeight", tc.sample_width AS "SampleDimensionsWidth", tc.sample_length AS
"SampleDimensionsLength", 'cm':character varying(2) AS "SampleDimensionsUnits", tc.density AS
"SampleDensity", tc.density_uncertainty AS "UncertaintyDensity", tc.laboratory AS "Laboratory",
COALESCE(tc.measurement_device, 'nil:missing':character varying) AS "MeasurementDevice",
tc.measurement_procedure AS "MeasurementProcedure", tc.measurement_pressure AS "MeasurementPressure",
tc.sample_temperature_diff AS "SampleTemperature", 'nil:missing':character varying(16) AS "MeasurementRun",
tc.conductivity AS "SaturatedSampleConductivity", tc.conductivity AS "SiteThermalConductivity",
tc.conductivity_se AS "UncertaintyTC", 'Watt per meter per Kelvin':character varying(64) AS "UnitsTC",
f.formation AS "GeologicFormationName", 'nil:missing':character varying(16) AS "GeologicFormationURI",
f.formation_type AS "GeologicFormationType", 'nil:missing':character varying(16) AS "DatumElevation",
CASE
    WHEN s.elevation_kb IS NOT NULL THEN s.elevation_kb
    WHEN s.elevation_df IS NOT NULL THEN s.elevation_df
    WHEN s.elevation_gl IS NOT NULL THEN s.elevation_gl
    ELSE NULL::double precision
END AS "Elevation", COALESCE(tc.permeability, (-99999)::double precision) AS "Permeability",
tc.porosity AS "Porosity", tc.comments AS "Notes",
CASE
    WHEN s.contributor IS NULL AND s.original_source IS NULL THEN ((ds.datasources::text || ' '::text) ||
ds.datatype::text)::character varying
    WHEN s.contributor IS NOT NULL THEN s.contributor
    ELSE s.original_source
END AS "Source", s.shape AS "Shape"
FROM core.thermal_conductivity tc
JOIN core.surface_site s ON tc.surface_site_id = s.id
LEFT JOIN core.datasets ds ON s.dataset_id = ds.id
LEFT JOIN core.ref_county rc ON s.county_id = rc.id
LEFT JOIN core.ref_state rs ON rc.state_id = rs.id
LEFT JOIN formation f ON f.surface_site_id = s.id
LEFT JOIN depth d ON d.surface_site_id = s.id
LEFT JOIN wellinfo wi ON wi.surface_site_id = s.id;

```

Table "core.template_thermalconductivity_materialized"

Column	Type	Modifiers	Storage	Description
ObservationURI	text		extended	
ObservationName	text		extended	
Label	text		extended	
OtherID	text		extended	
SampleDepth	double precision		plain	
MeasurementUnits	character varying(1)		extended	
SiteLocationURI	text		extended	
SiteLocationName	character varying		extended	
County	character varying		extended	
State	character varying		extended	
LatDegreeWGS84	numeric(10,7)		main	
LongDegreeWGS84	numeric(10,7)		main	

SRS	character varying(6)	extended
LocationUncertaintyStatement	character varying	extended
SampleCollectionDate	date	plain
SampleMeasurementDate	date	plain
SampleType	character varying	extended
SampleShape	character varying	extended
SampleDimensionsHeight	double precision	plain
SampleDimensionsWidth	double precision	plain
SampleDimensionsLength	double precision	plain
SampleDimensionsUnits	character varying(2)	extended
SampleDensity	double precision	plain
UncertaintyDensity	double precision	plain
Laboratory	character varying(32)	extended
MeasurementDevice	character varying	extended
MeasurementProcedure	character varying(128)	extended
MeasurementPressure	double precision	plain
SampleTemperature	double precision	plain
MeasurementRun	character varying(16)	extended
SaturatedSampleConductivity	double precision	plain
SiteThermalConductivity	double precision	plain
UncertaintyTC	double precision	plain
UnitsTC	character varying(64)	extended
GeologicFormationName	text	extended
GeologicFormationURI	character varying(16)	extended
GeologicFormationType	text	extended
DatumElevation	character varying(16)	extended
Elevation	double precision	plain
Permeability	double precision	plain
Porosity	double precision	plain
Notes	character varying(128)	extended
Source	character varying	extended
Shape	geometry	main
Has OIDs: no		

Well Fluid Production Observation Content Model

View "core.template_wellfluid"

Column	Type	Modifiers	Storage	Description
ObservationURI	text		extended	
RecordName	text		extended	
Operator	character varying(128)		extended	
HeaderURI	text		extended	
WellName	character varying		extended	
APINo	character varying(18)		extended	
OtherID	text		extended	
County	character varying		extended	
State	character varying		extended	
Field	character varying(128)		extended	
OtherLocationName	character varying(256)		extended	
PLSS_Meridians	character varying(64)		extended	
TWP	character varying(5)		extended	
RGE	character varying(5)		extended	
Section	character varying(10)		extended	
SectionPart	character varying(5)		extended	
LatDegreeWGS84	numeric(10,7)		main	

LongDegreeWGS84	numeric(10,7)	main
SRS	character varying(6)	extended
LocationUncertaintyStatement	character varying	extended
IntervalStartDateTime	character varying(16)	extended
IntervalEndDateTime	character varying(16)	extended
DurationDays	double precision	plain
Permit	character varying(128)	extended
FluidType	character varying(5)	extended
FluidVolume	double precision	plain
FluidVolumeUnits	character varying(32)	extended
Temperature_F	integer	plain
Pressure_psi	double precision	plain
AggregationType	character varying(16)	extended
MeasurementMethod	character varying(16)	extended
ContactIntervalTop	double precision	plain
ContactIntervalBottom	double precision	plain
Source	character varying	extended
RelatedResources	text	extended
Shape	geometry	main

View definition:

```

WITH wellinfo AS (
    SELECT well_info.surface_site_id, array_to_string(array_agg(DISTINCT well_info.commodity), '|')::text
AS commodity, array_to_string(array_agg(DISTINCT well_info.well_id), '|')::text AS well_id
    FROM core.well_info
    GROUP BY well_info.surface_site_id
)
SELECT (('https://geothermal.smu.edu/wellfluid/'::text || s.id) || ';'::text) || "interval".top_depth AS
"ObservationURI", (COALESCE(s.site_name, s.api)::text || 'top depth: '::text) || "interval".top_depth AS
"RecordName", s.operator AS "Operator", 'https://geothermal.smu.edu/header/'::text || s.id AS "HeaderURI",
COALESCE(s.site_name, s.api) AS "WellName", s.api AS "APINo", wi.well_id AS "OtherID",
COALESCE(rc.county_name, 'nil:missing'::character varying) AS "County", COALESCE(rs.state_name,
'nil:missing'::character varying) AS "State", s.field AS "Field", s.other_location_name AS "OtherLocationName",
s.plss_meridians AS "PLSS_Meridians", s.township AS "TWP", s.range AS "RGE", s.section AS "Section",
s.section_part AS "SectionPart", s.latitude AS "LatDegreeWGS84", s.longitude AS "LongDegreeWGS84",
"WGS84"::character varying(6) AS "SRS", COALESCE(s.location_notes, 'nil:missing'::character varying) AS
"LocationUncertaintyStatement", 'nil:missing'::character varying(16) AS "IntervalStartDateTime",
'nil:missing'::character varying(16) AS "IntervalEndDateTime", COALESCE("interval".duration_hours /
24::double precision, 0::double precision) AS "DurationDays", s.lease_name AS "Permit", 'water'::character
varying(5) AS "FluidType", "interval".water_volume AS "FluidVolume", 'barrels per day'::character varying(32)
AS "FluidVolumeUnits", (-99999) AS "Temperature_F", COALESCE("interval".bhp, (-99999)::double precision)
AS "Pressure_psi", 'nil:missing'::character varying(16) AS "AggregationType", 'nil:missing'::character varying(16)
AS "MeasurementMethod", "interval".top_depth AS "ContactIntervalTop", "interval".bottom_depth AS
"ContactIntervalBottom",
CASE
    WHEN s.contributor IS NULL AND s.original_source IS NULL THEN ((ds.datasource::text || ' '::text) ||
ds.datatype::text)::character varying
    WHEN s.contributor IS NOT NULL THEN s.contributor
    ELSE s.original_source
END AS "Source", 'https://geothermal.smu.edu/header/'::text || s.id AS "RelatedResources", s.shape AS
"Shape"
FROM core.surface_site s
JOIN core."interval" "interval" ON "interval".surface_site_id = s.id
LEFT JOIN core.datasets ds ON s.dataset_id = ds.id
LEFT JOIN core.ref_county rc ON s.county_id = rc.id
LEFT JOIN core.ref_state rs ON rc.state_id = rs.id
LEFT JOIN wellinfo wi ON wi.surface_site_id = s.id;

```

Table "core.template_wellfluid_materialized"

Column	Type	Modifiers	Storage	Description
ObservationURI	text		extended	
RecordName	text		extended	
Operator	character varying(128)		extended	
HeaderURI	text		extended	
WellName	character varying		extended	
APINo	character varying(18)		extended	
OtherID	text		extended	
County	character varying		extended	
State	character varying		extended	
Field	character varying(128)		extended	
OtherLocationName	character varying(256)		extended	
PLSS_Meridians	character varying(64)		extended	
TWP	character varying(5)		extended	
RGE	character varying(5)		extended	
Section	character varying(10)		extended	
SectionPart	character varying(5)		extended	
LatDegreeWGS84	numeric(10,7)		main	
LongDegreeWGS84	numeric(10,7)		main	
SRS	character varying(6)		extended	
LocationUncertaintyStatement	character varying		extended	
IntervalStartDateTime	character varying(16)		extended	
IntervalEndDateTime	character varying(16)		extended	
DurationDays	double precision		plain	
Permit	character varying(128)		extended	
FluidType	character varying(5)		extended	
FluidVolume	double precision		plain	
FluidVolumeUnits	character varying(32)		extended	
Temperature_F	integer		plain	
Pressure_psi	double precision		plain	
AggregationType	character varying(16)		extended	
MeasurementMethod	character varying(16)		extended	
ContactIntervalTop	double precision		plain	
ContactIntervalBottom	double precision		plain	
Source	character varying		extended	
RelatedResources	text		extended	
Shape	geometry		main	
Has OIDs: no				

Well Header Observation Content Model

View "core.template_wellheader"

Column	Type	Modifiers	Storage	Description
HeaderURI	text		extended	
WellName	character varying		extended	
APINo	character varying(18)		extended	
Label	character varying		extended	
OtherID	text		extended	
Operator	character varying(128)		extended	
MineralOwner	character varying(128)		extended	

LeaseName	character varying(128)	extended
SpudDate	text	extended
EndedDrillingDate	text	extended
welltype	character varying(128)	extended
Status	character varying(64)	extended
CommodityOfInterest	character varying	extended
StatusDate	text	extended
ProducingInterval	text	extended
County	character varying	extended
State	character varying	extended
Field	character varying(128)	extended
OtherLocationName	text	extended
PLSS_Meridians	character varying(64)	extended
Township	character varying(5)	extended
Range	character varying(5)	extended
Section	character varying(10)	extended
SectionPart	character varying(5)	extended
LatDegreeWGS84	numeric(10,7)	main
LongDegreeWGS84	numeric(10,7)	main
SRS	character varying(6)	extended
LocationUncertaintyStatement	character varying	extended
DrillerTotalDepth	double precision	plain
DepthReferencePoint	character varying(1)	extended
LengthUnits	character varying(1)	extended
WellBoreShape	character varying(1)	extended
TrueVerticalDepth	double precision	plain
ElevationKB	double precision	plain
ElevationDF	double precision	plain
ElevationGL	double precision	plain
FormationTD	text	extended
BitDiameterCollar	double precision	plain
DiameterUnits	character varying(6)	extended
Notes	character varying(512)	extended
CasingBottomDepthDriller	double precision	plain
CasingTopDepth	numeric	main
CasingPipeDiameter	double precision	plain
CasingWeight	double precision	plain
CasingWeightUnits	character varying(6)	extended
InformationSource	character varying	extended
Shape	geometry	main

View definition:

```

WITH depth AS (
    SELECT depth.surface_site_id, max(depth.true_vertical_depth) AS true_vertical_depth, max(depth.depth)
    AS depth, max(depth.bit_diameter) AS bit_diameter, array_to_string(array_agg(DISTINCT depth.depth_date),
    '::text) AS depth_date
    FROM core.depth
    GROUP BY depth.surface_site_id
), formation AS (
    SELECT formation.surface_site_id, array_to_string(array_agg(DISTINCT (formation.formation::text || '
depth='::text) ||
    CASE
        WHEN formation.formation_depth IS NULL THEN 'NA'::text
        ELSE "::text || formation.formation_depth
    END), '::text) AS formation
    FROM core.formation
    GROUP BY formation.surface_site_id

```

```

), casing AS (
  SELECT t.surface_site_id, t.casing_depth,
    CASE
      WHEN t.casing_top::text ~ '^[0-9]+$'::text THEN t.casing_top::numeric
      ELSE (-999999)::numeric
    END AS casing_top, t.casing_size_inches, t.casing_weight
  FROM core.casing t
  JOIN ( SELECT casing.surface_site_id, max(casing.file_date) AS max_date, max(casing.casing_depth) AS
max_depth
        FROM core.casing
        GROUP BY casing.surface_site_id) x ON t.surface_site_id = x.surface_site_id AND (x.max_date IS
NULL OR t.file_date = x.max_date) AND t.casing_depth = x.max_depth
), wellinfo AS (
  SELECT well_info.surface_site_id, array_to_string(array_agg(DISTINCT well_info.commodity), '|')::text
AS commodity, array_to_string(array_agg(DISTINCT well_info.well_id), '|')::text AS well_id
  FROM core.well_info
  GROUP BY well_info.surface_site_id
), welldates AS (
  SELECT well_dates.surface_site_id, array_to_string(array_agg(DISTINCT well_dates.drl_compl_date),
'|')::text AS drl_compl_date, array_to_string(array_agg(DISTINCT well_dates.drl_start_date), '|')::text AS
drl_start_date
  FROM core.well_dates
  GROUP BY well_dates.surface_site_id
), "interval" AS (
  SELECT z.surface_site_id, z.producing_reservoir
  FROM ( SELECT "interval".surface_site_id, array_to_string(array_agg(DISTINCT
"interval".producing_reservoir), '|')::text) AS producing_reservoir
  FROM core."interval"
  GROUP BY "interval".surface_site_id) z
)
SELECT 'https://geothermal.smu.edu/header/'::text || s.id AS "HeaderURI", COALESCE(s.site_name, s.api) AS
"WellName", s.api AS "APINo", COALESCE(s.site_name, s.api) AS "Label", COALESCE(s.other_ids, '
'::character varying)::text || COALESCE(wi.well_id, ' '::text) AS "OtherID", s.operator AS "Operator",
s.site_owner AS "MineralOwner", s.lease_name AS "LeaseName", COALESCE('::text || wd.drl_start_date) AS
"SpudDate", COALESCE(d.depth_date, '::text || wd.drl_compl_date) AS "EndedDrillingDate", s.well_type AS
welltype, s.plugged AS "Status", COALESCE(s.well_type, wi.commodity::character varying) AS
"CommodityOfInterest", 'shut_in_date'::text || s.shut_in_date AS "StatusDate", iv.producing_reservoir AS
"ProducingInterval", COALESCE(rc.county_name, 'nil:missing'::character varying) AS "County",
COALESCE(rs.state_name, 'nil:missing'::character varying) AS "State", s.field AS "Field",
(((COALESCE(s.tectonic_province, ' '::character varying)::text || ' '::text) || COALESCE(s.ams, ' '::character
varying)::text) || ' '::text) || COALESCE(s.other_location_name, ' '::character varying)::text AS
"OtherLocationName", s.plss_meridians AS "PLSS_Meridians", s.township AS "Township", s.range AS "Range",
s.section AS "Section", s.section_part AS "SectionPart", s.latitude AS "LatDegreeWGS84", s.longitude AS
"LongDegreeWGS84", 'WGS84'::character varying(6) AS "SRS", COALESCE(s.location_notes,
'nil:missing'::character varying) AS "LocationUncertaintyStatement", d.depth AS "DrillerTotalDepth", '::character
varying(1) AS "DepthReferencePoint", 'm'::character varying(1) AS "LengthUnits", '::character varying(1) AS
"WellBoreShape", d.true_vertical_depth AS "TrueVerticalDepth", s.elevation_kb AS "ElevationKB",
s.elevation_df AS "ElevationDF", s.elevation_gl AS "ElevationGL", f.formation AS "FormationTD",
d.bit_diameter AS "BitDiameterCollar", 'inches'::character varying(6) AS "DiameterUnits", s.comment AS
"Notes", cs.casing_depth AS "CasingBottomDepthDriller", cs.casing_top AS "CasingTopDepth",
cs.casing_size_inches AS "CasingPipeDiameter", cs.casing_weight AS "CasingWeight", 'lbs/ft'::character
varying(6) AS "CasingWeightUnits",
  CASE
    WHEN s.contributor IS NULL AND s.original_source IS NULL THEN ((ds.datasource::text || ' '::text) ||
ds.datatype::text)::character varying
    WHEN s.contributor IS NOT NULL THEN s.contributor

```

```

ELSE s.original_source
END AS "InformationSource", s.shape AS "Shape"
FROM core.surface_site s
LEFT JOIN core.datasets ds ON s.dataset_id = ds.id
LEFT JOIN core.ref_county rc ON s.county_id = rc.id
LEFT JOIN core.ref_state rs ON rc.state_id = rs.id
LEFT JOIN depth d ON d.surface_site_id = s.id
LEFT JOIN formation f ON f.surface_site_id = s.id
LEFT JOIN casing cs ON cs.surface_site_id = s.id
LEFT JOIN wellinfo wi ON wi.surface_site_id = s.id
LEFT JOIN welldates wd ON wd.surface_site_id = s.id
LEFT JOIN "interval" iv ON iv.surface_site_id = s.id;

```

Table "core.template_wellheader_materialized"

Column	Type	Modifiers	Storage	Description
HeaderURI	text		extended	
WellName	character varying		extended	
APINo	character varying(18)		extended	
Label	character varying		extended	
OtherID	text		extended	
Operator	character varying(128)		extended	
MineralOwner	character varying(128)		extended	
LeaseName	character varying(128)		extended	
SpudDate	text		extended	
EndedDrillingDate	text		extended	
welltype	character varying(128)		extended	
Status	character varying(64)		extended	
CommodityOfInterest	character varying		extended	
StatusDate	text		extended	
ProducingInterval	text		extended	
County	character varying		extended	
State	character varying		extended	
Field	character varying(128)		extended	
OtherLocationName	text		extended	
PLSS_Meridians	character varying(64)		extended	
Township	character varying(5)		extended	
Range	character varying(5)		extended	
Section	character varying(10)		extended	
SectionPart	character varying(5)		extended	
LatDegreeWGS84	numeric(10,7)		main	
LongDegreeWGS84	numeric(10,7)		main	
SRS	character varying(6)		extended	
LocationUncertaintyStatement	character varying		extended	
DrillerTotalDepth	double precision		plain	
DepthReferencePoint	character varying(1)		extended	
LengthUnits	character varying(1)		extended	
WellBoreShape	character varying(1)		extended	
TrueVerticalDepth	double precision		plain	
ElevationKB	double precision		plain	
ElevationDF	double precision		plain	
ElevationGL	double precision		plain	
FormationTD	text		extended	
BitDiameterCollar	double precision		plain	
DiameterUnits	character varying(6)		extended	

Notes	character varying(512)	extended
CasingBottomDepthDriller	double precision	plain
CasingTopDepth	numeric	main
CasingPipeDiameter	double precision	plain
CasingWeight	double precision	plain
CasingWeightUnits	character varying(6)	extended
InformationSource	character varying	extended
Shape	geometry	main
Has OIDs: no		

Well Log Observation Content Model

View "core.template_welllog"

Column	Type	Modifiers	Storage	Description
LogURI	text		extended	
DisplayName	character varying		extended	
Notes	text		extended	
LogTypeTerm	character varying(11)		extended	
WellBoreURI	character varying(11)		extended	
APINo	character varying(18)		extended	
WellType	character varying(11)		extended	
County	character varying		extended	
State	character varying		extended	
Field	character varying(128)		extended	
OtherLocationName	character varying(256)		extended	
PLSS_Meridians	character varying(64)		extended	
TWP	character varying(5)		extended	
RGE	character varying(5)		extended	
Section_	character varying(10)		extended	
SectionPart	character varying(5)		extended	
LatDegree	numeric(10,7)		main	
LongDegree	numeric(10,7)		main	
SRS	character varying(6)		extended	
LocationUncertaintyStatement	character varying		extended	
DrillerTotalDepth_ft	double precision		plain	
ElevationGL_ft	double precision		plain	
TopLoggedInterval_ft	double precision		plain	
BottomLoggedInterval_ft	double precision		plain	
DateTimeLogRun	character varying(7)		extended	
PaperCopyLocation	character varying(16)		extended	
ScannedFileURL	text		extended	
Source	character varying		extended	
Shape	geometry		main	

View definition:

```
WITH depth AS (
    SELECT depth.surface_site_id, max(depth.true_vertical_depth) AS true_vertical_depth, max(depth.depth)
    AS depth, max(depth.bit_diameter) AS bit_diameter, array_to_string(array_agg(DISTINCT depth.depth_date),
    ' '::text) AS depth_date
    FROM core.depth
    GROUP BY depth.surface_site_id
)
SELECT 'http://geothermal.smu.edu/gtda/rest/fileSystem/getFile?uuid='::text || res.resource_id AS "LogURI",
COALESCE(res.label, 'nil:missing'::character varying) AS "DisplayName", (res.type::text || ' ': ::text) ||
COALESCE(res.description, ' '::character varying)::text AS "Notes", 'nil:missing'::character varying(11) AS
"LogTypeTerm", 'nil:missing'::character varying(11) AS "WellBoreURI", s.api AS "APINo",
```



```

'nil:missing'::character varying(11) AS "WellType", COALESCE(rc.county_name, 'nil:missing'::character varying)
AS "County", COALESCE(rs.state_name, 'nil:missing'::character varying) AS "State", s.field AS "Field",
s.other_location_name AS "OtherLocationName", s.plss_meridians AS "PLSS_Meridians", s.township AS
"TPW", s.range AS "RGE", s.section AS "Section_", s.section_part AS "SectionPart", s.latitude AS "LatDegree",
s.longitude AS "LongDegree", 'WGS84'::character varying(6) AS "SRS", COALESCE(s.location_notes,
'nil:missing'::character varying) AS "LocationUncertaintyStatement", COALESCE(d.depth * 3.280839895::double
precision, (-99999)::double precision) AS "DrillerTotalDepth_ft", COALESCE(s.elevation_gl *
3.280839895::double precision, (-99999)::double precision) AS "ElevationGL_ft", (-99999)::double precision AS
"TopLoggedInterval_ft", (-99999)::double precision AS "BottomLoggedInterval_ft", 'nil:NA'::character varying(7)
AS "DateTimeLogRun", ds.datasources AS "PaperCopyLocation",
'http://geothermal.smu.edu/gtda/rest/fileSystem/getFile?uuid='::text || res.resource_id AS "ScannedFileURL",
CASE
    WHEN s.contributor IS NULL AND s.original_source IS NULL THEN ((ds.datasources::text || ' '::text) ||
ds.datatype::text)::character varying
    WHEN s.contributor IS NOT NULL THEN s.contributor
    ELSE s.original_source
END AS "Source", s.shape AS "Shape"
FROM core.surface_site s
JOIN core.site_related_resources res ON s.id = res.surface_site_id OR st_contains(res.shape, s.shape)
LEFT JOIN core.datasets ds ON s.dataset_id = ds.id
LEFT JOIN core.ref_county rc ON s.county_id = rc.id
LEFT JOIN core.ref_state rs ON rc.state_id = rs.id
LEFT JOIN depth d ON d.surface_site_id = s.id
WHERE res.resource_id IS NOT NULL AND (res.type::text = ANY (ARRAY['WELL_LOG'::character
varying::text, 'FIELDNOTE'::character varying::text, 'TD_CURVES'::character varying::text,
'WELL_LOG_HEADER'::character varying::text, 'WellLog'::character varying::text]));

```

Table "core.template_welllog_materialized"

Column	Type	Modifiers	Storage	Description
LogURI	text		extended	
DisplayName	character varying		extended	
Notes	text		extended	
LogTypeTerm	character varying(11)		extended	
WellBoreURI	character varying(11)		extended	
APINo	character varying(18)		extended	
WellType	character varying(11)		extended	
County	character varying		extended	
State	character varying		extended	
Field	character varying(128)		extended	
OtherLocationName	character varying(256)		extended	
PLSS_Meridians	character varying(64)		extended	
TPW	character varying(5)		extended	
RGE	character varying(5)		extended	
Section_	character varying(10)		extended	
SectionPart	character varying(5)		extended	
LatDegree	numeric(10,7)		main	
LongDegree	numeric(10,7)		main	
SRS	character varying(6)		extended	
LocationUncertaintyStatement	character varying		extended	
DrillerTotalDepth_ft	double precision		plain	
ElevationGL_ft	double precision		plain	
TopLoggedInterval_ft	double precision		plain	
BottomLoggedInterval_ft	double precision		plain	
DateTimeLogRun	character varying(7)		extended	

PaperCopyLocation	character varying(16)	extended
ScannedFileURL	text	extended
Source	character varying	extended
Shape	geometry	main
Has OIDs: no		

Well Test Observation Content Model

View "core.template_welltest"

Column	Type	Modifiers	Storage	Description
ObservationURI	text		extended	
WellHeaderURI	text		extended	
WellName	character varying		extended	
APINo	character varying(18)		extended	
Label	character varying		extended	
OtherID	character varying(16)		extended	
TestName	character varying(3)		extended	
TestType	character varying(9)		extended	
TestOperator	character varying(6)		extended	
TestDateTime	date		plain	
Field	character varying(128)		extended	
County	character varying		extended	
State	character varying		extended	
PLSS_Meridians	character varying(64)		extended	
Township	character varying(5)		extended	
Range	character varying(5)		extended	
Section_	character varying(10)		extended	
SectionPart	character varying(5)		extended	
LatDegree	numeric(10,7)		main	
LongDegree	numeric(10,7)		main	
SRS	character varying(6)		extended	
LocationUncertaintyStatement	character varying(128)		extended	
TargetFormation	character varying(12)		extended	
DepthTopOpenZone	double precision		plain	
DepthBottomOpenZone	double precision		plain	
UnitsOfMeasure	character varying(12)		extended	
FlowHistory	character varying(12)		extended	
PressureInitialShutIn	double precision		plain	
TimeInitialShutIn_min	double precision		plain	
PreferredPressure	double precision		plain	
PressureFinalShutIn	integer		plain	
TimeFinalShutIn_min	double precision		plain	
Units_Pressure	character varying(3)		extended	
HydrostaticPressure	double precision		plain	
HydrostaticPressureMethod	character varying(12)		extended	
EquilibriumPressure	double precision		plain	
FlowPressureMin	integer		plain	
FlowPressureMax	integer		plain	
TemperatureCorrection	double precision		plain	
Source	character varying		extended	
Shape	geometry		main	

View definition:

```
SELECT ('https://geothermal.smu.edu/obervation/welltest/'::text || s.id) || wt.g10_tst_date AS "ObservationURI",
'https://geothermal.smu.edu/header/'::text || s.id AS "WellHeaderURI", COALESCE(s.site_name, s.api) AS
"WellName", s.api AS "APINo", COALESCE(s.site_name, s.api, wt.well_info) AS "Label", wt.well_info AS
```

```

"OtherID", 'G10'::character varying(3) AS "TestName", 'Injection'::character varying(9) AS "TestType",
wt.g10_opr AS "TestOperator", wt.g10_tst_date AS "TestDateTime", s.field AS "Field",
COALESCE(rc.county_name, 'nil:missing'::character varying) AS "County", COALESCE(rs.state_name,
'nil:missing'::character varying) AS "State", s.plss_meridians AS "PLSS_Meridians", s.township AS "Township",
s.range AS "Range", s.section AS "Section_", s.section_part AS "SectionPart", s.latitude AS "LatDegree",
s.longitude AS "LongDegree", 'WGS84'::character varying(6) AS "SRS", s.location_notes AS
"LocationUncertaintyStatement", 'nil:missing'::character varying(12) AS "TargetFormation", (-99999)::double
precision AS "DepthTopOpenZone", (-99999)::double precision AS "DepthBottomOpenZone",
'nil:missing'::character varying(12) AS "UnitsOfMeasure", 'nil:missing'::character varying(12) AS "FlowHistory",
(-99999)::double precision AS "PressureInitialShutIn", (-99999)::double precision AS "TimeInitialShutIn_min", (-
99999)::double precision AS "PreferredPressure", wt.g10_shut_press AS "PressureFinalShutIn", (-99999)::double
precision AS "TimeFinalShutIn_min", 'PSI'::character varying(3) AS "Units_Pressure", (-99999)::double precision
AS "HydrostaticPressure", 'nil:missing'::character varying(12) AS "HydrostaticPressureMethod", (-99999)::double
precision AS "EquilibriumPressure", wt.g10_flow_press AS "FlowPressureMin", wt.g10_flow_press AS
"FlowPressureMax", (-99999)::double precision AS "TemperatureCorrection",
CASE
    WHEN s.contributor IS NULL AND s.original_source IS NULL THEN ((ds.datasource::text || ' '::text) ||
ds.datatype::text)::character varying
    WHEN s.contributor IS NOT NULL THEN s.contributor
    ELSE s.original_source
END AS "Source", s.shape AS "Shape"
FROM core.surface_site s
JOIN core.well_test wt ON wt.surface_site_id = s.id
LEFT JOIN core.datasets ds ON s.dataset_id = ds.id
LEFT JOIN core.ref_county rc ON s.county_id = rc.id
LEFT JOIN core.ref_state rs ON rc.state_id = rs.id;

```

Table "core.template_welltest_materialized"

Column	Type	Modifiers	Storage	Description
ObservationURI	text		extended	
WellHeaderURI	text		extended	
WellName	character varying		extended	
APINo	character varying(18)		extended	
Label	character varying		extended	
OtherID	character varying(16)		extended	
TestName	character varying(3)		extended	
TestType	character varying(9)		extended	
TestOperator	character varying(6)		extended	
TestDateTime	date		plain	
Field	character varying(128)		extended	
County	character varying		extended	
State	character varying		extended	
PLSS_Meridians	character varying(64)		extended	
Township	character varying(5)		extended	
Range	character varying(5)		extended	
Section_	character varying(10)		extended	
SectionPart	character varying(5)		extended	
LatDegree	numeric(10,7)		main	
LongDegree	numeric(10,7)		main	
SRS	character varying(6)		extended	
LocationUncertaintyStatement	character varying(128)		extended	
TargetFormation	character varying(12)		extended	
DepthTopOpenZone	double precision		plain	
DepthBottomOpenZone	double precision		plain	

UnitsOfMeasure	character varying(12)	extended
FlowHistory	character varying(12)	extended
PressureInitialShutIn	double precision	plain
TimeInitialShutIn_min	double precision	plain
PreferredPressure	double precision	plain
PressureFinalShutIn	integer	plain
TimeFinalShutIn_min	double precision	plain
Units_Pressure	character varying(3)	extended
HydrostaticPressure	double precision	plain
HydrostaticPressureMethod	character varying(12)	extended
EquilibriumPressure	double precision	plain
FlowPressureMin	integer	plain
FlowPressureMax	integer	plain
TemperatureCorrection	double precision	plain
Source	character varying	extended
Shape	geometry	main
Has OIDs: no		

Temperature Series Mapping ID

Sequence "core.temperature¹_series_mapping_id_seq"

Column	Type	Value	Storage	Description
sequence_name	name	temprature_series_mapping_id_seq	plain	
last_value	bigint	325	plain	
start_value	bigint	1	plain	
increment_by	bigint	1	plain	
max_value	bigint	9223372036854775807	plain	
min_value	bigint	1	plain	
cache_value	bigint	1	plain	
log_cnt	bigint	0	plain	
is_cycled	boolean	f	plain	
is_called	boolean	t	plain	

Thermal Conductivity

Table "core.thermal_conductivity"

Column	Type	Modifiers	Storage	Description
id	uuid	not null	plain	
surface_site_id	uuid		plain	
conductivity	double precision		plain	
conductivity_se	double precision		plain	
conductivity_sample_type_id	integer		plain	
conductivity_nb_samples	integer		plain	
conductivity_measurement_type_id	integer		plain	
conductivity_sample_shape_id	integer		plain	
sample_width	double precision		plain	
sample_height	double precision		plain	
sample_length	double precision		plain	
density	double precision		plain	
density_uncertainty	double precision		plain	
dataset_id	uuid		plain	

¹ sic

resistance	double precision	plain
porosity	double precision	plain
laboratory	character varying(32)	extended
measurement_device	character varying(32)	extended
measurement_procedure	character varying(128)	extended
sample_collection_date	date	plain
sample_measurement_date	date	plain
sample_depth	double precision	plain
sample_shape	character varying(32)	extended
sample_temperature_diff	double precision	plain
sample_type	character varying(32)	extended
permeability	double precision	plain
lithology_info	character varying(128)	extended
comments	character varying(128)	extended
measurement_pressure	double precision	plain

Indexes:

"thermal_conductivity_pkey" PRIMARY KEY, btree (id)

Referenced by:

TABLE "core.heat_flow" CONSTRAINT "heat_flow_thermal" FOREIGN KEY (thermal_conductivity_id)

REFERENCES core.thermal_conductivity(id) MATCH FULL

Has OIDs: no

Index "core.thermal_conductivity_pkey"

Column	Type	Storage	Description
id	uuid	plain	
primary key, btree, for table "core.thermal_conductivity"			

Well Dates

Table "core.well_dates"

Column	Type	Modifiers	Storage	Description
id	bigint	not null default nextval('core.well_dates_id_seq' eq '::regclass)	plain	
surface_site_id	uuid		plain	
dataset_id	uuid		plain	
compl_date	date		plain	
drl_compl_date	date		plain	
water_injection_nbr	character varying(6)		extended	
elevation	integer		plain	
elevation_code	character varying(2)		extended	
drl_start_date	date		plain	
status_date	date		plain	

Indexes:

"well_dates_pkey" PRIMARY KEY, btree (id)

Foreign-key constraints:

"well_dates_dataset" FOREIGN KEY (dataset_id) REFERENCES core.datasets(id) MATCH FULL

Has OIDs: no

Sequence "core.well_dates_id_seq"

Column	Type	Value	Storage	Description
sequence_name	name	well_dates_id_seq	plain	

last_value	bigint	4897207	plain
start_value	bigint	1	plain
increment_by	bigint	1	plain
max_value	bigint	9223372036854775807	plain
min_value	bigint	1	plain
cache_value	bigint	1	plain
log_cnt	bigint	12	plain
is_cycled	boolean	f	plain
is_called	boolean	t	plain

Index "core.well_dates_pkey"

Column	Type	Storage	Description
id	bigint	plain	
primary key, btree, for table "core.well_dates"			

Well Info

Table "core.well_info"

Column	Type	Modifiers	Storage	Description
id	bigint	not null default nextval('core.well_info_id_seq'::reg class)	plain	
surface_site_id	uuid		plain	
dataset_id	uuid		plain	
commodity	character varying(16)		extended	
well_id	character varying(16)		extended	
permit	character varying(16)		extended	

Indexes:

"well_info_pkey" PRIMARY KEY, btree (id)

Foreign-key constraints:

"well_info_dataset" FOREIGN KEY (dataset_id) REFERENCES core.datasets(id) MATCH FULL

Has OIDs: no

Well Info ID

Sequence "core.well_info_id_seq"

Column	Type	Value	Storage	Description
sequence_name	name	well_info_id_seq	plain	
last_value	bigint	3290246	plain	
start_value	bigint	1	plain	
increment_by	bigint	1	plain	
max_value	bigint	9223372036854775807	plain	
min_value	bigint	1	plain	
cache_value	bigint	1	plain	
log_cnt	bigint	16	plain	
is_cycled	boolean	f	plain	
is_called	boolean	t	plain	

Index "core.well_info_pkey"

Column	Type	Storage	Description

id	bigint	plain

primary key, btree, for table "core.well_info"

Well Test

Table "core.well_test"

Column	Type	Modifiers	Storage	Description
id	bigint	not null default nextval('core.well_test_id_seq' '::regclass)	plain	
surface_site_id	uuid		plain	
dataset_id	uuid		plain	
well_info	character varying(16)		extended	
g10_id	character varying(2)		extended	
g10_rrcid	integer	not null	plain	
g10_tst_date	date		plain	
g10_dly_gas	integer		plain	
g10_dly_cond	numeric(6,1)		main	
g10_dly_wtr	integer		plain	
g10_siwh_press	integer		plain	
g10_shut_cd	character varying(1)		extended	
g10_flow_press	integer		plain	
g10_bhp	integer		plain	
g10_shut_press	integer		plain	
g10_opr	character varying(6)		extended	
g10_wh_pres_test_date	date		plain	

Indexes:

"well_test_pkey" PRIMARY KEY, btree (id)

Foreign-key constraints:

"well_test_dataset" FOREIGN KEY (dataset_id) REFERENCES core.datasets(id) MATCH FULL

Has OIDs: no

Sequence "core.well_test_id_seq"

Column	Type	Value	Storage	Description
sequence_name	name	well_test_id_seq	plain	
last_value	bigint	5681691	plain	
start_value	bigint	1	plain	
increment_by	bigint	1	plain	
max_value	bigint	9223372036854775807	plain	
min_value	bigint	1	plain	
cache_value	bigint	1	plain	
log_cnt	bigint	0	plain	
is_cycled	boolean	f	plain	
is_called	boolean	t	plain	

Index "core.well_test_pkey"

Column	Type	Storage	Description
id	bigint	plain	

primary key, btree, for table "core.well_test"

File System

Pulled on April 7, 2014 using the following command:

```
psql -h sgtl1d1 -U postgres gtda -c '\d+ filesystem.*' > gtda-filesystem-tables.txt 2> /dev/null
```

Required signing onto sgtl1p1 with PuTTY. Also required the password for username 'postgres' on sgtl1d1.

Table "filesystem.directories"

Column	Type	Modifiers	Storage	Description
dir_uuid	uuid	not null	plain	
parent_uuid	uuid		plain	
dir_name	character varying(128)	not null	extended	
dir_creation_date	timestamp without time zone		plain	

Indexes:

"file_system_dir_pkey" PRIMARY KEY, btree (dir_uuid)

Foreign-key constraints:

"file_system_dir_fkey" FOREIGN KEY (parent_uuid) REFERENCES filesystem.directories(dir_uuid)

Referenced by:

TABLE "filesystem.directories" CONSTRAINT "file_system_dir_fkey" FOREIGN KEY (parent_uuid) REFERENCES filesystem.directories(dir_uuid)

TABLE "filesystem.files" CONSTRAINT "file_system_files_fkey" FOREIGN KEY (parent_uuid) REFERENCES filesystem.directories(dir_uuid) MATCH FULL

Has OIDs: no

Index "filesystem.file_system_dir_pkey"

Column	Type	Storage	Description
dir_uuid	uuid	plain	
primary key, btree, for table "filesystem.directories"			

Index "filesystem.file_system_files_pkey"

Column	Type	Storage	Description
file_uuid	uuid	plain	
primary key, btree, for table "filesystem.files"			

Table "filesystem.files"

Column	Type	Modifiers	Storage	Description
file_uuid	uuid	not null	plain	
parent_uuid	uuid		plain	
file_size	bigint	not null	plain	
file_path_internal	character varying(256)	not null	extended	
file_upload_date	timestamp without time zone		plain	
file_name	character varying(256)	not null	extended	
file_mimetype	character varying(70)		extended	
file_downloadable	boolean	default true	plain	
file_path_external	character varying(256)		extended	
file_provider	character varying(32)		extended	

Indexes:

"file_system_files_pkey" PRIMARY KEY, btree (file_uuid)

Foreign-key constraints:

"file_system_files_fkey" FOREIGN KEY (parent_uuid) REFERENCES filesystem.directories(dir_uuid)

MATCH FULL

Has OIDs: no

Staging Transform Operations

Pulled April 7, 2014 using the following command:

```
psql -h sgtl1d1 -U postgres gtda -c '\df+ staging.*' > gtda-staging-functions.txt 2> /dev/null
```

Required signing in to sgtl1p1 with PuTTY. Also requires the userid for user name 'postgres' on sgtl1d1.

The command generates a very wide table format and was been modified for readability by removing the two right most columns, "Source Code" and "Description". The description column was not populated. The information appearing in the Source Code column now appears underneath the headings for each of the 14 different staging transform functions. The 14 sets of operations include:

Schema	Name	Result data type	Argument data types	Type	Volatility	Owner	Language
staging	beg_data_transformationx	integer		normal	volatile	postgres	plpgsql
staging	beg_reservoir_transformationx	integer		normal	volatile	postgres	plpgsql
staging	cornell_transformationx	integer		normal	volatile	postgres	plpgsql
staging	create_matview	void	name, name	normal	volatile	postgres	plpgsql
staging	drop_matview	void	name	normal	volatile	postgres	plpgsql
staging	mlkay_bht_transformationx	integer		normal	volatile	postgres	plpgsql
staging	mlkay_uic_transformationx	integer		normal	volatile	postgres	plpgsql
staging	mlkay_wb_transformationx	integer		normal	volatile	postgres	plpgsql
staging	recreate_all_matviews	integer		normal	volatile	postgres	plpgsql
staging	refresh_matview	void	name	normal	volatile	postgres	plpgsql
staging	smu_bht_transformationx	integer		normal	volatile	postgres	plpgsql
staging	smu_hf_transformationx	integer		normal	volatile	postgres	plpgsql
staging	texas_tech_transformationx	integer		normal	volatile	postgres	plpgsql
staging	und_data_transformationx	integer		normal	volatile	postgres	plpgsql

Source Code for BEG Well Transform Operations

```
: \r
: DECLARE \r
:     dsetUUID uuid;\r
: BEGIN\r
: \r
:     select core.register_dataset('BEG','ALL',NULL) into dsetUUID;\r
:     \r
:     INSERT INTO core.surface_site (id, dataset_id ,api, latitude, longitude, \r
:         lease_name, other_location_name, site_name,operator,contributor,comment)\r
:     select distinct * from (SELECT \r
:         first_value(core.generate_uuid('BEG','SURFACE_SITE',lat || ','|| long)) OVER (PARTITION BY lat,long) AS id,\r
:         dsetUUID as dataset_id ,\r
:         first_value(api) OVER (PARTITION BY lat,long) as api, \r
:         first_value(lat) OVER (PARTITION BY lat,long) as latitude,\r
:         first_value(long) OVER (PARTITION BY lat,long) as longitude,\r
:         first_value(lease) OVER (PARTITION BY lat,long) as lease_name, \r
:         first_value("location") OVER (PARTITION BY lat,long) as other_location_name,\r
:         first_value(wellname) OVER (PARTITION BY lat,long) as site_name, \r
:         first_value(operator_n) OVER (PARTITION BY lat,long) as operator,\r
:         first_value('BEG: '||source) OVER (PARTITION BY lat,long) as contributor,\r
:         first_value('BEG accession core database: '||accession_log_beg) OVER (PARTITION BY lat,long) as comment\r
```

```

:      FROM staging.beg_well_data) Z;\r
:\r
:      UPDATE core.surface_site SET shape = geometryfromtext('POINT('|| longitude || ' ' || latitude ||)'), 4326) WHERE
dataset_id = dsetUUID;\r
:      UPDATE core.surface_site SET county_id=c.id FROM core.ref_county c WHERE ST_Contains(c.geometry,shape) and
dataset_id = dsetUUID;\r
:\r
:\r
:      INSERT INTO core.formation (surface_site_id, dataset_id, formation,formation_depth)\r
:      SELECT distinct\r
:          core.generate_uuid('BEG','SURFACE_SITE',lat || ','|| long),\r
:          dsetUUID as dataset_id, \r
:          formation,\r
:          depth_m\r
:          FROM staging.beg_well_data \r
:          where length(formation)>0;\r
:\r
:      INSERT INTO core.formation (surface_site_id, dataset_id, formation,formation_depth)\r
:      SELECT distinct\r
:          core.generate_uuid('BEG','SURFACE_SITE',lat || ','|| long),\r
:          dsetUUID as dataset_id, \r
:          fm_at_td,\r
:          tvd/3.280839895 -- converts to meters\r
:          FROM staging.beg_well_data \r
:          where length(fm_at_td)>0;\r
:\r
:\r
:      INSERT INTO core.well_info (surface_site_id, dataset_id, well_id)\r
:      SELECT distinct\r
:          core.generate_uuid('BEG','SURFACE_SITE',lat || ','|| long) AS surface_site_id,\r
:          dsetUUID as dataset_id,\r
:          wellid\r
:          FROM staging.beg_well_data \r
:          where wellid is not null;\r
:\r
:      INSERT INTO core.depth (surface_site_id, dataset_id, true_vertical_depth)\r
:      SELECT distinct\r
:          core.generate_uuid('BEG','SURFACE_SITE',lat || ','|| long), \r
:          dsetUUID,\r
:          tvd/3.280839895 -- converts to meters\r
:          FROM staging.beg_well_data WHERE tvd is not null;\r
:\r
:      INSERT INTO core.bht_measurements (id,surface_site_id, dataset_id, run_number,
bht_depth,bht,bht_corrected,drill_date)\r

```

```

:      SELECT distinct\r
:      core.generate_uuid('BEG','BHT',lat || ',' || long || ';' || coalesce(depth_m,0) || ';' || row_number() OVER (partition by
lat, long, depth_m)) AS id, \r
:      core.generate_uuid('BEG','SURFACE_SITE',lat || ',' || long) AS surface_site_id, \r
:      dsetUUID, \r
:      number_run AS run_number, \r
:      depth_m AS bht_depth, \r
:      bht_c AS bht, \r
:      bht_c_cor AS bht_corrected, \r
:      log_date AS drill_date \r
:      FROM staging.beg_well_data; \r
: \r
:      INSERT INTO core.heat_flow (id,bht_measurements_id,surface_site_id, dataset_id, gradient_corrected) \r
:      SELECT distinct\r
:      core.generate_uuid('BEG','HEAT_FLOW',lat || ',' || long || ';' || coalesce(depth_m,0) || ';' || row_number() OVER
(partition by lat, long, depth_m)) AS id, \r
:      core.generate_uuid('BEG','BHT',lat || ',' || long || ';' || coalesce(depth_m,0) || ';' || row_number() OVER (partition by
lat, long, depth_m)) AS bht_measurements_id, \r
:      core.generate_uuid('BEG','SURFACE_SITE',lat || ',' || long) AS surface_site_id, \r
:      dsetUUID, \r
:      (gradient_per_1000ft*5/9)*3.280839895 AS gradient_corrected -- convert to delta(C)/km from delta(F)/1000ft \r
:      FROM staging.beg_well_data WHERE gradient_per_1000ft is not null; \r
: \r
: \r
: \r
: -- add well logs: \r
: \r
: /* \r
: -- query to get linking info: \r
: select substring(L.log_image_addr from '[a-zA-Z0-9]+.tif+$') as filename, 'WELL_LOG' as "type", L.log_types as
description, core.generate_uuid('BEG','SURFACE_SITE',lat || ',' || long) as site_identfier, W.api, W.lat as lat, W.long as long \r
:      from staging.beg_logs_with_images L JOIN staging.beg_well_data W ON L.wellid=W.wellid \r
:      WHERE W.wellid is not null \r
:      \r
: -- not used anymore \r
:      INSERT INTO core.site_related_resources (surface_site_id, label, resource_id,url, dataset_id,"type",shape) \r
:      SELECT distinct \r
:      core.generate_uuid('BEG','SURFACE_SITE',lat || ',' || long), \r
:      file_name, \r
:      core.generate_uuid('BEG','WELL_LOG',file_name), \r
:      'https://geothermal.smu.edu/gtda/rest/fileSystem/getFile?uuid=' ||
core.generate_uuid('BEG','WELL_LOG',file_name), \r
:      dsetUUID, \r
:      'WELL_LOG', \r
:      geometryfromtext('POINT(' || long || ' ' || lat || ')', 4326) \r

```

```

:      FROM (\r
:      select W.lat as lat, W.long as long, L.log_image_addr as file_name \r
:      from staging.beg_logs_with_images L JOIN staging.beg_well_data W ON L.wellid=W.wellid \r
:      WHERE W.wellid is not null\r
:      ) Z;\r
: */\r
: \r
: \r
: --- add casing info:\r
:      INSERT INTO core.casing (surface_site_id, dataset_id, casing_size_inches, casing_depth, liner_inches, liner_top,
:      liner_base, choke_mm)\r
:      SELECT distinct\r
:      core.generate_uuid('BEG','SURFACE_SITE',lat || ',' || long) AS surface_site_id,\r
:      dsetUUID,\r
:      casing_size_inches,\r
:      casing_depth, \r
:      liner_inches, \r
:      liner_top, \r
:      liner_base, \r
:      choke_mm          \r
:      FROM staging.beg_well_data;\r
: \r
: --- add interval info up to 8 times:\r
:      INSERT INTO core.interval (surface_site_id, dataset_id, top_depth, bottom_depth, interval_length, water_volume, bhp,
:      duration_hours, producing_reservoir)\r
:      SELECT distinct\r
:      core.generate_uuid('BEG','SURFACE_SITE',lat || ',' || long) AS surface_site_id,\r
:      dsetUUID,\r
:      top_ft_1/3.280839895,\r
:      bottom_ft_1/3.280839895,\r
:      interval_ft_1/3.280839895,\r
:      watrvol_1_bbl,\r
:      bhp_1,\r
:      durtn_hrs_1,\r
:      prod_reservoir_1      \r
:      FROM staging.beg_well_data\r
:      WHERE top_ft_1 is not null and bottom_ft_1 is not null;\r
: \r
: INSERT INTO core.interval (surface_site_id, dataset_id, top_depth, bottom_depth, interval_length, water_volume, bhp,
:      duration_hours, producing_reservoir)\r
:      SELECT distinct\r
:      core.generate_uuid('BEG','SURFACE_SITE',lat || ',' || long) AS surface_site_id,\r
:      dsetUUID,\r
:      top_ft_2/3.280839895,\r

```

```

:         bottom_ft_2/3.280839895,\r
:         interval_ft_2/3.280839895,\r
:         watrvol_2_bbl,\r
:         bhp_2,\r
:         durtn_hrs_2,\r
:         prod_reservoir_2          \r
:     FROM staging.beg_well_data\r
:     WHERE top_ft_2 is not null and bottom_ft_2 is not null;\r
:\r
: INSERT INTO core.interval (surface_site_id, dataset_id,top_depth, bottom_depth, interval_length, water_volume, bhp,
duration_hours, producing_reservoir)\r
:     SELECT distinct\r
:         core.generate_uuid('BEG','SURFACE_SITE',lat || ','|| long) AS surface_site_id,\r
:         dsetUUID,\r
:         top_ft_3/3.280839895,\r
:         bottom_ft_3/3.280839895,\r
:         interval_ft_3/3.280839895,\r
:         watrvol_3_bbl,\r
:         bhp_3,\r
:         durtn_hrs_3,\r
:         prod_reservoir_3          \r
:     FROM staging.beg_well_data\r
:     WHERE top_ft_3 is not null and bottom_ft_3 is not null;\r
:\r
: INSERT INTO core.interval (surface_site_id, dataset_id,top_depth, bottom_depth, interval_length, water_volume, bhp,
duration_hours, producing_reservoir)\r
:     SELECT distinct\r
:         core.generate_uuid('BEG','SURFACE_SITE',lat || ','|| long) AS surface_site_id,\r
:         dsetUUID,\r
:         top_ft_4/3.280839895,\r
:         bottom_ft_4/3.280839895,\r
:         interval_ft_4/3.280839895,\r
:         watrvol_4_bbl,\r
:         bhp_4,\r
:         durtn_hrs_4,\r
:         prod_reservoir_4          \r
:     FROM staging.beg_well_data\r
:     WHERE top_ft_4 is not null and bottom_ft_4 is not null;\r
:\r
:\r
: INSERT INTO core.interval (surface_site_id, dataset_id,top_depth, bottom_depth, interval_length, water_volume, bhp,
duration_hours, producing_reservoir)\r
:     SELECT distinct\r
:         core.generate_uuid('BEG','SURFACE_SITE',lat || ','|| long) AS surface_site_id,\r

```

```

:      dsetUUID,\r
:      top_ft_5/3.280839895,\r
:      bottom_ft_5/3.280839895,\r
:      interval_ft_5/3.280839895,\r
:      watrvol_5_bbl,\r
:      bhp_5,\r
:      durtn_hrs_5,\r
:      prod_reservoir_5      \r
:  FROM staging.beg_well_data\r
:  WHERE top_ft_5 is not null and bottom_ft_5 is not null;\r
: \r
: \r
: INSERT INTO core.interval (surface_site_id, dataset_id,top_depth, bottom_depth, interval_length, water_volume, bhp,
duration_hours, producing_reservoir)\r
:
:  SELECT distinct\r
:
:      core.generate_uuid('BEG','SURFACE_SITE',lat || ','|| long) AS surface_site_id,\r
:      dsetUUID,\r
:      top_ft_6/3.280839895,\r
:      bottom_ft_6/3.280839895,\r
:      interval_ft_6/3.280839895,\r
:      watrvol_6_bbl,\r
:      bhp_6,\r
:      durtn_hrs_6,\r
:      prod_reservoir_6      \r
:  FROM staging.beg_well_data\r
:  WHERE top_ft_6 is not null and bottom_ft_6 is not null;\r
: \r
: \r
: INSERT INTO core.interval (surface_site_id, dataset_id,top_depth, bottom_depth, interval_length, water_volume, bhp,
duration_hours, producing_reservoir)\r
:
:  SELECT distinct\r
:
:      core.generate_uuid('BEG','SURFACE_SITE',lat || ','|| long) AS surface_site_id,\r
:      dsetUUID,\r
:      top_ft_7/3.280839895,\r
:      bottom_ft_7/3.280839895,\r
:      interval_ft_7/3.280839895,\r
:      watrvol_7_bbl,\r
:      bhp_7,\r
:      durtn_hrs_7,\r
:      prod_reservoir_7      \r
:  FROM staging.beg_well_data\r
:  WHERE top_ft_7 is not null and bottom_ft_7 is not null;\r
: \r
: \r

```

```

: INSERT INTO core.interval (surface_site_id, dataset_id,top_depth, bottom_depth, interval_length, water_volume, bhp,
duration_hours, producing_reservoir)\r
:
:      SELECT distinct\r
:
:      core.generate_uuid('BEG','SURFACE_SITE',lat || ','|| long) AS surface_site_id,\r
:
:      dsetUUID,\r
:
:      top_ft_8/3.280839895,\r
:
:      bottom_ft_8/3.280839895,\r
:
:      interval_ft_8/3.280839895,\r
:
:      watrvol_8_bbl,\r
:
:      bhp_8,\r
:
:      durtn_hrs_8,\r
:
:      prod_reservoir_8          \r
:
:      FROM staging.beg_well_data\r
:
:      WHERE top_ft_8 is not null and bottom_ft_8 is not null;\r
: \r
: -- update materialized views in staging:\r
: perform staging.refresh_matview('staging.beg_well_view_materialized');\r
: perform core.update_all_matviews();\r
: \r
: RETURN 0;\r
: END;\r
:

```

Source Code for BEG Reservoir Transform Operations

```

:
: DECLARE
:
:      dsetUUID uuid;
: BEGIN
:
:      RAISE INFO 'PostgreSQL is now starting staging.beg_reservoir_transformationx';
:      select core.register_dataset('BEG','RESERVOIR',NULL) into dsetUUID;
:
:
:      RAISE INFO 'PostgreSQL is now inserting into core.reservoir';
:      INSERT INTO core.reservoir
:
:      select distinct
:
:      core.generate_uuid('BEG','RESERVOIR',reservoirname),
:
:      dsetUUID,
:
:      reservoiruri,
:
:      reservoirname,
:
:      label,
:
:      otherid,
:
:      othername,
:
:      description,
:
:      county,

```


- : state,
- : locationname,
- : latdegree,
- : longdegree,
- : srs,
- : locationuncertaintystatement,
- : temperaturemax_f,
- : temperaturemin_f,
- : depthtotop_ft,
- : depthtobottom_ft,
- : dominantformationname,
- : otherformationname,
- : dominantlithologyname,
- : otherlithologyname,
- : area_acres,
- : width_ft,
- : length_ft,
- : sandvolume_cft,
- : blockvolume_cft,
- : percentsand,
- : averagenetsandthickness_ft,
- : wellcount,
- : faultcount,
- : faulted,
- : faulttype,
- : faultyhydrauliccharacter,
- : firstproductiondate,
- : lastproductiondate,
- : yearsinproduction,
- : monthlyproduction_bbl,
- : productionreportdate,
- : averagemonthlywaterproduction_bbl,
- : minimummonthlywaterproduction_bbl,
- : maximummonthlywaterproduction_bbl,
- : waterinjection_bbl,
- : watersaturation_pct,
- : watertds_mgl,
- : averagefieldwidepressure_psi,
- : averagefieldwidegradient_degfft,
- : porositymax_pct,
- : porositymin_pct,
- : permeabilitymax_md,
- : permeabilitymin_md,

```

:      cementationmineralogy,
:      trap,
:      drive,
:      gravity,
:      commodityofinterest,
:      stimulation,
:      averageperforatedthickness_ft,
:      relatedresource,
:      source,
:      metadatauri,
:      majorfields,
:      GeometryFromText('POLYGON((' ||
:      cordinate_nw_lat || ' ' || cordinate_nw_long || ',' ||
:      cordinate_ne_lat || ' ' || cordinate_ne_long || ',' ||
:      cordinate_se_lat || ' ' || cordinate_se_long || ',' ||
:      cordinate_sw_lat || ' ' || cordinate_sw_long || ',' ||
:      cordinate_nw_lat || ' ' || cordinate_nw_long ||
:      '))', 4326) as shape
:      FROM staging.beg_reservoir_data;
:      RAISE INFO 'PostgreSQL has completed inserting into core.reservoir';
:      --UPDATE core.surface_site SET shape = geometryfromtext('POINT(' || longitude || ' ' || latitude ||)'), 4326) WHERE
dataset_id = dsetUUID;
:      --UPDATE core.surface_site SET county_id=c.id FROM core.ref_county c WHERE ST_Contains(c.geometry,shape)
and dataset_id = dsetUUID;
:
: -- update materialized views in staging
: RAISE INFO 'PostgreSQL is now refreshing staging.beg_reservoir_view_materialized';
: perform staging.refresh_matview('staging.beg_reservoir_view_materialized');
: RAISE INFO 'PostgreSQL has completed refreshing staging.beg_reservoir_view_materialized';
:
: --perform core.update_all_matviews();
: RAISE INFO 'PostgreSQL is now refreshing core.template_reservoir_materialized';
: perform core.refresh_matview('core.template_reservoir_materialized');
: RAISE INFO 'PostgreSQL has completed refreshing core.template_reservoir_materialized';
:
: RAISE INFO 'PostgreSQL has completed staging.beg_reservoir_transformationx';
:
: RETURN 0;
: END;
:

```

Source Code for Cornell Transform Operations

```

: DECLARE \r
:      dsetUUID uuid;\r

```

```

: BEGIN\r
:\r
:   select core.register_dataset('CORNELL','ALL',NULL) into dsetUUID;\r
:   \r
:   UPDATE staging.cornell_well_data SET WELL_NM=null where WELL_NM="";\r
:\r
:\r
:   INSERT INTO core.surface_site (id, dataset_id ,api, site_name, operator, longitude, latitude,
other_location_name,field,section,basement_depth,plugged,\r
:   well_type,lithology,comment, average_heatflow)\r
:   select distinct * from (SELECT \r
:       first_value(core.generate_uuid('CORNELL','SURFACE_SITE',API_WELLNO)) OVER (PARTITION BY
API_WELLNO) AS id, \r
:       dsetUUID AS dataset_id,\r
:       first_value(API_WELLNO) OVER (PARTITION BY API_WELLNO) AS api,\r
:       first_value(WELL_NM) OVER (PARTITION BY API_WELLNO) AS site_name,\r
:       first_value("operator") OVER (PARTITION BY API_WELLNO) AS operator,\r
:       first_value(decimal_longitude) OVER (PARTITION BY API_WELLNO) AS longitude,\r
:       first_value(decimal_latitude) OVER (PARTITION BY API_WELLNO) AS latitude,          \r
:       --first_value(permit_number) OVER (PARTITION BY API_WELLNO) AS lease_name,\r
:       first_value('town=' || town || '; municipality=' || municipality || '; farm=' || farm_name || '; pool=' || pool || ';
quad=' || quad) OVER (PARTITION BY API_WELLNO) AS other_location_name,\r
:       first_value(field) OVER (PARTITION BY API_WELLNO) AS field,\r
:       first_value(map_section) OVER (PARTITION BY API_WELLNO) AS section,\r
:       first_value(sed_thickness_m) OVER (PARTITION BY API_WELLNO) AS basement_depth,\r
:       first_value(well_status) OVER (PARTITION BY API_WELLNO) AS plugged,\r
:       first_value(well_type) OVER (PARTITION BY API_WELLNO) AS well_type,\r
:       first_value('cosuna=' || cosuna_col) OVER (PARTITION BY API_WELLNO) AS lithology,\r
:       first_value(data_source) OVER (PARTITION BY API_WELLNO) AS comment,\r
:       first_value(site_heat_flow) OVER (PARTITION BY API_WELLNO) as average_heatflow
\r
:       FROM (select * from staging.cornell_well_data order by API_WELLNO, log_date) A) B);\r
:\r
:\r
:   UPDATE core.surface_site SET shape = geometryfromtext('POINT('|| longitude || ' ' || latitude ||)'), 4326) WHERE
dataset_id = dsetUUID;\r
:   UPDATE core.surface_site SET county_id=c.id FROM core.ref_county c WHERE ST_Contains(c.geometry,shape) and
dataset_id = dsetUUID;\r
:\r
:\r
:   INSERT INTO core.formation (surface_site_id, dataset_id, formation,formation_depth)\r
:       SELECT distinct core.generate_uuid('CORNELL','SURFACE_SITE',API_WELLNO), dsetUUID,\r
:       tdform, tvd_m\r
:       FROM staging.cornell_well_data where length(tdform)>0;\r
:\r
:\r

```

```

:      INSERT INTO core.well_dates ( surface_site_id, dataset_id, compl_date, elevation)  \r
:      SELECT distinct \r
:      core.generate_uuid('CORNELL','SURFACE_SITE',API_WELLNO), dsetUUID,\r
:      compl_date,\r
:      elevation/3.280839895 -- converts to meters\r
:      FROM staging.cornell_well_data where (compl_date is not null) or (elevation is not null);\r
: \r
:      INSERT INTO core.well_info ( surface_site_id, dataset_id, permit)  \r
:      SELECT distinct \r
:      core.generate_uuid('CORNELL','SURFACE_SITE',API_WELLNO), \r
:      dsetUUID,\r
:      permit_number \r
:      FROM staging.cornell_well_data where (permit_number is not null) AND (length(permit_number)>0);\r
: \r
: \r
:      INSERT INTO core.depth (surface_site_id, dataset_id, depth_date, true_vertical_depth)\r
:      SELECT distinct core.generate_uuid('CORNELL','SURFACE_SITE',API_WELLNO), dsetUUID,\r
:      td_date, tvd_m\r
:      FROM staging.cornell_well_data;\r
: \r
:      INSERT INTO core.bht_measurements (id,surface_site_id, dataset_id, bht,bht_corrected,drill_date,bht_depth,\r
bht_correction_type)\r
:      SELECT distinct * FROM\r
:      (SELECT\r
:      core.generate_uuid('CORNELL','BHT',API_WELLNO || ';' || coalesce(log_depth_m,0) || ';' || row_number()\r
OVER (partition by API_WELLNO, log_depth_m)) AS id, \r
:      core.generate_uuid('CORNELL','SURFACE_SITE',API_WELLNO) AS surface_site_id, \r
:      dsetUUID,\r
:      bht_c AS bht, \r
:      harrison_bht_c AS bht_corrected,\r
:      log_date AS drill_date,\r
:      log_depth_m AS bht_depth,\r
:      cast('Harrison' as varchar(16)) as bht_correction_type\r
:      FROM staging.cornell_well_data order by API_WELLNO, log_depth_m) Z;\r
: \r
:      INSERT INTO core.thermal_conductivity (id,surface_site_id, dataset_id,conductivity)\r
:      SELECT distinct * FROM\r
:      (SELECT \r
:      core.generate_uuid('CORNELL','THERMAL_CONDUCTIVITY',API_WELLNO || ';' ||\r
coalesce(log_depth_m,0) || ';' || row_number() OVER (partition by API_WELLNO, log_depth_m)) AS id, \r
:      core.generate_uuid('CORNELL','SURFACE_SITE',API_WELLNO) AS surface_site_id, \r
:      dsetUUID,\r
:      average_conductivity\r
:      FROM staging.cornell_well_data \r

```

```

:      where (average_conductivity is not null)\r
:      order by API_WELLNO, log_depth_m) Z;\r
:\r
:      INSERT INTO core.heat_flow (id, bht_measurements_id,thermal_conductivity_id,surface_site_id,dataset_id, gradient,
gradient_corrected, heat_flow, \r
:      interval_bottom_depth, interval_order, quality)\r
:      SELECT distinct * FROM\r
:      (SELECT\r
:      core.generate_uuid('CORNELL','HEAT_FLOW',API_WELLNO || ';' || coalesce(log_depth_m,0) || ';' ||
row_number() OVER (partition by API_WELLNO, log_depth_m)) AS id, \r
:      core.generate_uuid('CORNELL','BHT',API_WELLNO || ';' || coalesce(log_depth_m,0) || ';' || row_number()
OVER (partition by API_WELLNO, log_depth_m)) AS bht_measurements_id, \r
:      core.generate_uuid('CORNELL','THERMAL_CONDUCTIVITY',API_WELLNO || ';' ||
coalesce(log_depth_m,0) || ';' || row_number() OVER (partition by API_WELLNO, log_depth_m)) AS
thermal_conductivity_id,\r
:      core.generate_uuid('CORNELL','SURFACE_SITE',API_WELLNO) AS surface_site_id, \r
:      dsetUUID,\r
:      uncorr_grad AS gradient, \r
:      harrison_corr_grad_c_km AS gradient_corrected,\r
:      heat_flow_mw_m2 AS heat_flow,\r
:      log_depth_m as interval_bottom_depth,\r
:      interval_id as interval_order,\r
:      interval_quality_code as quality\r
:      FROM staging.cornell_well_data order by API_WELLNO, log_depth_m) Z;\r
:\r
:\r
:      INSERT INTO core.bht_measurements (id,surface_site_id, dataset_id, bht,bht_depth,estimated)\r
:      (SELECT \r
:      core.generate_uuid('CORNELL','BHT',API_WELLNO || ';'1500' || ';' || random()) AS id, \r
:      core.generate_uuid('CORNELL','SURFACE_SITE',API_WELLNO) AS surface_site_id, \r
:      dsetUUID,\r
:      temp_est_1500m AS bht, \r
:      1500 AS bht_depth,\r
:      true as estimated\r
:      FROM staging.cornell_well_data);\r
:\r
:      INSERT INTO core.bht_measurements (id,surface_site_id, dataset_id, bht,bht_depth,estimated)\r
:      (SELECT \r
:      core.generate_uuid('CORNELL','BHT',API_WELLNO || ';'3000' || ';' || random()) AS id, \r
:      core.generate_uuid('CORNELL','SURFACE_SITE',API_WELLNO) AS surface_site_id, \r
:      dsetUUID,\r
:      temp_est_3000m AS bht, \r
:      3000 AS bht_depth,\r
:      true as estimated\r
:      FROM staging.cornell_well_data);\r

```

```

:      \r
:  INSERT INTO core.bht_measurements (id,surface_site_id, dataset_id, bht,bht_depth,estimated)\r
:  (SELECT \r
:      core.generate_uuid('CORNELL','BHT',API_WELLNO || ';'4500' || ';' || random()) AS id, \r
:      core.generate_uuid('CORNELL','SURFACE_SITE',API_WELLNO) AS surface_site_id, \r
:      dsetUUID,\r
:      temp_est_4500m AS bht, \r
:      4500 AS bht_depth,\r
:      true as estimated\r
:  FROM staging.cornell_well_data);\r
: \r
:  INSERT INTO core.bht_measurements (id,surface_site_id, dataset_id, bht,bht_depth,estimated)\r
:  (SELECT \r
:      core.generate_uuid('CORNELL','BHT',API_WELLNO || ';'6000' || ';' || random()) AS id, \r
:      core.generate_uuid('CORNELL','SURFACE_SITE',API_WELLNO) AS surface_site_id, \r
:      dsetUUID,\r
:      temp_est_6000m AS bht, \r
:      6000 AS bht_depth,\r
:      true as estimated\r
:  FROM staging.cornell_well_data);\r
:      \r
:  INSERT INTO core.bht_measurements (id,surface_site_id, dataset_id, bht,bht_depth,estimated)\r
:  (SELECT \r
:      core.generate_uuid('CORNELL','BHT',API_WELLNO || ';'7500' || ';' || random()) AS id, \r
:      core.generate_uuid('CORNELL','SURFACE_SITE',API_WELLNO) AS surface_site_id, \r
:      dsetUUID,\r
:      temp_est_7500m AS bht, \r
:      7500 AS bht_depth,\r
:      true as estimated\r
:  FROM staging.cornell_well_data);\r
: \r
:  INSERT INTO core.bht_measurements (id,surface_site_id, dataset_id, bht,bht_depth,estimated)\r
:  (SELECT \r
:      core.generate_uuid('CORNELL','BHT',API_WELLNO || ';'10000' || ';' || random()) AS id, \r
:      core.generate_uuid('CORNELL','SURFACE_SITE',API_WELLNO) AS surface_site_id, \r
:      dsetUUID,\r
:      temp_est_10000m AS bht, \r
:      10000 AS bht_depth,\r
:      true as estimated\r
:  FROM staging.cornell_well_data);\r
: \r
:  INSERT INTO core.bht_measurements (id,surface_site_id, dataset_id, bht,bht_depth,estimated)\r
:  (SELECT \r

```

```

:         core.generate_uuid('CORNELL','BHT',API_WELLNO || ';' || random()) AS id, \r
:         core.generate_uuid('CORNELL','SURFACE_SITE',API_WELLNO) AS surface_site_id, \r
:         dsetUUID, \r
:         150 AS bht, \r
:         est_depth_150c AS bht_depth, \r
:         true as estimated \r
:         FROM staging.cornell_well_data); \r
: \r
:     INSERT INTO core.bht_measurements (id,surface_site_id, dataset_id, bht,bht_depth,estimated) \r
:     (SELECT \r
:         core.generate_uuid('CORNELL','BHT',API_WELLNO || ';' || random()) AS id, \r
:         core.generate_uuid('CORNELL','SURFACE_SITE',API_WELLNO) AS surface_site_id, \r
:         dsetUUID, \r
:         80 AS bht, \r
:         est_depth_80c AS bht_depth, \r
:         true as estimated \r
:         FROM staging.cornell_well_data); \r
: \r
: -- update materialized views in staging \r
: perform staging.refresh_matview('staging.cornell_well_view_materialized'); \r
: perform core.update_all_matviews(); \r
: \r
: RETURN 0; \r
: END; \r
: \r
:

```

Source Code for Create Materialized Views Transform Operations

```

: DECLARE
:     matview ALIAS FOR $1;
:     view_name ALIAS FOR $2;
:     entry staging.matviews%ROWTYPE;
: BEGIN
:     SELECT * INTO entry FROM staging.matviews WHERE mv_name = matview;
:
:     IF FOUND THEN
:         RAISE EXCEPTION 'Materialized view "%" already exists.',
:             matview;
:     END IF;
:
:     EXECUTE 'REVOKE ALL ON ' || view_name || ' FROM postgres';
:     EXECUTE 'GRANT SELECT ON ' || view_name || ' TO postgres';
:     EXECUTE 'CREATE TABLE ' || matview || ' AS SELECT * FROM ' || view_name;

```

```

: EXECUTE 'REVOKE ALL ON ' || matview || ' FROM postgres';
: EXECUTE 'GRANT SELECT ON ' || matview || ' TO postgres';
:
: INSERT INTO staging.matviews (mv_name, v_name, last_refresh)
:   VALUES (matview, view_name, CURRENT_TIMESTAMP);
:
: RETURN;
: END
:

```

Source Code for Drop Materialized Views Transform Operations

```

: DECLARE
:   matview ALIAS FOR $1;
:   entry staging.matviews%ROWTYPE;
: BEGIN
:   SELECT * INTO entry FROM staging.matviews WHERE mv_name = matview;
:
:   IF NOT FOUND THEN
:     RAISE EXCEPTION 'Materialized view % does not exist.', matview;
:   END IF;
:
:   EXECUTE 'DROP TABLE ' || matview;
:   DELETE FROM staging.matviews WHERE mv_name=matview;
:
:   RETURN;
: END
:

```

Source Code for MLKay BHT Transform Operations

```

:
: DECLARE
:
:   dsetUUID uuid;
: BEGIN
:
:
:   select core.register_dataset('MLKay','WB',NULL) into dsetUUID;
:
:
:
:   INSERT INTO core.bht_measurements (id, surface_site_id, dataset_id, bht,bht_depth, drill_date,

```



```

time_since_circulation)

:      SELECT distinct

:      core.generate_uuid('MLKay','WB',s.api || ';' || (drillertotaldepth/3.280839895) || ';' ||
coalesce(to_char(measurementdatetime,'YYYY-MM-DD'),')) AS id,

:      s.id AS surface_site_id,

:      dsetUUID as dataset_id,

:      (measuredtemperaturef-32)*5/9 AS bht, -- converts to C

:      drillertotaldepth/3.280839895 AS bht_depth, -- coverts to meters

:      measurementdatetime AS drill_date,

:      circulationduration AS time_since_circulation

:      FROM core.surface_site s , (select distinct * from staging.mlk_bht_data) b

:      where s.api=('42' || lpad('' || b.api42, 8,'0') || '0000')

:      and s.dataset_id=dsetUUID;

:

:

:      INSERT INTO core.casing (surface_site_id, dataset_id, casing_size_inches)

:      SELECT distinct

:      core.generate_uuid('MLKay','WB',s.api) AS surface_site_id,

:      dsetUUID,

:      casingpipediameter

:      FROM core.surface_site s , (select distinct * from staging.mlk_bht_data) b

:      where s.api=('42' || lpad('' || b.api42, 8,'0') || '0000')

:      and s.dataset_id=dsetUUID;

:

:      INSERT INTO core.depth (surface_site_id, dataset_id, depth, depth_date, bit_diameter, mud_weight, viscosity )

:      SELECT distinct

:      core.generate_uuid('MLKay','WB',s.api) AS surface_site_id,

:      dsetUUID,

:      drillertotaldepth/3.280839895 AS bht_depth, -- coverts to meters

```

```

:      measurementdatetime AS drilldate,
:
:      bitdiametercollar as bit_diameter,
:
:      mud_weight as mud_weight,
:
:      viscosity as viscosity
:
:      FROM core.surface_site s , (select distinct * from staging.mlk_bht_data) b
:
:      where s.api=('42' || lpad('' || b.api42, 8,'0') || '0000')
:
:      and s.dataset_id=dsetUUID;
:
: RETURN 0;
: END;
:

```

Source Code for MLKay UIC Transform Operations

```

:
: DECLARE
:      dsetUUID uuid;
: BEGIN
:
:      select core.register_dataset('MLKay','WB',NULL) into dsetUUID;
:
:      INSERT INTO core.fluid (id, surface_site_id, dataset_id, cntl_no, lease_id, well_no, field_no, oper, uic_class,
bbl_vol_inj, max_inj_pressure,
:      top_inj_zone, bot_inj_zone, w_status, sw, fw, bw, air, gas, lpg, co2, polymer, cycle_yr, fw_avg_or_max,
fw_bbls_day)
:      SELECT distinct
:      core.generate_uuid('MLKay','WB','42' || lpad('' || r.api_number, 8,'0') || '0000;' || r.cntl_no) AS id,
:      core.generate_uuid('MLKay','WB','42' || lpad('' || r.api_number, 8,'0') || '0000') AS surface_site_id,
:      dsetUUID as dataset_id,
:      r.cntl_no,
:      r.lease_id,
:      r.well_no,
:      r.field_no,
:      r.oper,
:      r.uic_class,
:      r.bbl_vol_inj,
:      r.max_inj_pressure,
:      r.top_inj_zone,
:      r.bot_inj_zone,

```

```

:      m.w_status,
:      m.sw,
:      m.fw,
:      m.bw,
:      m.air,
:      m.gas,
:      m.lpg,
:      m.co2,
:      m.polymer,
:      m.cycle_yr,
:      m.fw_avg_or_max,
:      m.fw_bbls_day
:      FROM uic.uic_root r, core.surface_site s, uic.uic_monitor m
:      where s.dataset_id=dsetUUID and s.api=('42' || lpad('' || r.api_number, 8,'0') || '0000') and r.cntl_no=m.cntl_no;
:
:      INSERT INTO core.fluid_h10 (injection_id,surface_site_id, dataset_id, h10_date, h10_avg_inj_pressure,
h10_max_inj_pressure, h10_total_vol_bbl, h10_total_vol_mcf)
:      SELECT distinct
:      core.generate_uuid('MLKay','WB','42' || lpad('' || r.api_number, 8,'0') || '0000;' || r.cntl_no) AS injection_id,
:      core.generate_uuid('MLKay','WB','42' || lpad('' || r.api_number, 8,'0') || '0000') AS surface_site_id,
:      dsetUUID as dataset_id,
:      h10_date,
:      h10_avg_inj_pressure,
:      h10_max_inj_pressure,
:      h10_total_vol_bbl,
:      h10_total_vol_mcf
:      FROM uic.uic_root r, core.surface_site s, uic.uic_h10 h10
:      where s.dataset_id=dsetUUID and s.api=('42' || lpad('' || r.api_number, 8,'0') || '0000') and r.cntl_no=h10.cntl_no;
:
:
:      INSERT INTO core.fluid_h10h (injection_id,surface_site_id, dataset_id, h10h_date, h10h_max_hydrocarb_psig,
h10h_max_brine_psig)
:      SELECT distinct
:      core.generate_uuid('MLKay','WB','42' || lpad('' || r.api_number, 8,'0') || '0000;' || r.cntl_no) AS injection_id,
:      core.generate_uuid('MLKay','WB','42' || lpad('' || r.api_number, 8,'0') || '0000') AS surface_site_id,
:      dsetUUID as dataset_id,
:      h10h_date,
:      h10h_max_hydrocarb_psig,
:      h10h_max_brine_psig
:      FROM uic.uic_root r, core.surface_site s, uic.uic_h10h h10h
:      where s.dataset_id=dsetUUID and s.api=('42' || lpad('' || r.api_number, 8,'0') || '0000') and r.cntl_no=h10h.cntl_no;
:
:
:

```

```

: -- --we don't need these, since no new surface sites are added --
: --UPDATE core.surface_site SET shape = geometryfromtext('POINT(' || longitude || ' ' || latitude ||)'), 4326) WHERE
dataset_id = dsetUUID;
: --UPDATE core.surface_site SET county_id=c.id FROM core.ref_county c WHERE ST_Contains(c.geometry,shape) and
dataset_id = dsetUUID;
:
: RETURN 0;
: END;
:

```

Source Code for MLKay WB (Well Bore) Transform Operations

```

:
: DECLARE
:     dsetUUID uuid;
: BEGIN
:
:     select core.register_dataset('MLKay','WB',NULL) into dsetUUID;
:
:     INSERT INTO core.surface_site (id, dataset_id ,api, district, plugged,fresh_water_converted,shut_in_date,
block,section,
:         feet_from_survey, latitude,longitude,lease_name,section_block_survey,hole_open, contributor)
:     SELECT distinct
:     core.generate_uuid('MLKay','WB','42' || lpad('' || nl.api_number, 8,'0') || '0000') AS id,
:     dsetUUID as dataset_id,
:     '42' || lpad('' || nl.api_number, 8,'0') || '0000' as api,
:     r.field_district as district,
:     r.plug_flag as plugged,
:     case when r.fresh_water_flag='Y' then 1 else 0 end as fresh_water_converted,
:     r.shut_in_date as shut_in_date,
:     nl.block_number as block,
:     nl.section as section,
:     nl.feet_from_sur_sect_1 as feet_from_survey,
:     nl.wgs84_latitude as latitude,
:     -1*nl.wgs84_longitude as longitude,
:     ol.lease_name as lease_name,
:     ol.sec_blk_survey_loc as sec_block_survey,
:     case when z.open_hole_code='OH' then 1 else 0 end as hole_open,
:     case when (-1*nl.wgs84_longitude > 97.15) then 'Bureau of Economic Geology' else 'MLKay Technologies' end as
contributor
:     FROM  wb.wb_new_location_seg nl
:     LEFT JOIN wb.wb_root r ON  r.api_number=nl.api_number
:     LEFT JOIN wb.wb_old_location_seg ol ON  r.api_number=ol.api_number
:     LEFT JOIN (select distinct api_number, open_hole_code from wb.wb_perf_seg where open_hole_code='OH') Z
ON r.api_number=z.api_number;
:

```

```

:
: INSERT INTO core.depth (surface_site_id, dataset_id, depth_date,depth)
:     SELECT distinct
:     core.generate_uuid('MLKay','WB','42' || lpad('' || r.api_number, 8,'0') || '0000') AS surface_site_id,
:     dsetUUID as dataset_id,
:     r.orig_compl_date as completion_date,
:     r.total_depth/3.280839895 as depth -- converts to meters
:     FROM wb.wb_new_location_seg nl
:     LEFT JOIN wb.wb_root r ON r.api_number=nl.api_number
:     where r.total_depth>0;
:
: INSERT INTO core.formation (surface_site_id, dataset_id, formation,formation_depth)
: SELECT distinct
:     core.generate_uuid('MLKay','WB','42' || lpad('' || nl.api_number, 8,'0') || '0000') AS id,
:     dsetUUID as dataset_id,
:     fs.formation_name,
:     fs.formation_depth/3.280839895 -- converts to meters
:     FROM wb.wb_new_location_seg nl
:     INNER JOIN wb.wb_formation_seg fs ON nl.api_number=fs.api_number;
:
:
: INSERT INTO core.casing (surface_site_id, dataset_id, casing_size_inches, casing_depth, casing_weight,
casing_top,casing_left,file_date)
:     SELECT distinct
:     core.generate_uuid('MLKay','WB','42' || lpad('' || nl.api_number, 8,'0') || '0000') AS surface_site_id,
:     dsetUUID,
:     cs.casing_size_data,
:     cs.casing_depth_set,
:     cs.casing_weight_lbs_ft[1],
:     cs.top_of_cement_casing,
:     cs.amount_casing_left,
:     file_date
:     FROM (wb.wb_new_location_seg nl INNER JOIN wb.wb_casing_seg cs ON nl.api_number=cs.api_number);
:
:
: INSERT INTO core.well_info (surface_site_id, dataset_id, commodity, well_id, permit)
:     SELECT distinct
:     core.generate_uuid('MLKay','WB','42' || lpad('' || seg.api_number, 8,'0') || '0000') AS surface_site_id,
:     dsetUUID as dataset_id,
:     oil as commodity,
:     case when oil='O' then (seg.oil_district || '-' || seg.oil_lse_number || ' ' || seg.oil_well_number) else cast(gas_rrcid as
text) end as well_id,
:     permit_number as permit
:     FROM wb.wb_new_location_seg nl INNER JOIN wb.wb_well_id_seg seg ON nl.api_number=seg.api_number;

```

```

:
:
:
:   INSERT INTO core.well_dates ( surface_site_id, dataset_id, compl_date, drl_compl_date, water_injection_nbr,
elevation, elevation_code)
:
:   SELECT distinct
:
:   core.generate_uuid('MLKay','WB','42' || lpad('' || seg.api_number, 8,'0') || '0000') AS surface_site_id,
:
:   dsetUUID as dataset_id,
:
:   seg.compl_date,
:
:   seg.drl_compl_date,
:
:   seg.water_injection_nbr,
:
:   seg.elevation/3.280839895, -- converts to meters
:
:   seg.elevation_code
:
:   from wb.wb_file_date_seg seg, wb.wb_new_location_seg nl
:
:   where nl.api_number=seg.api_number
:
:   and (elevation >0 OR water_injection_nbr!=null OR compl_date is not null OR drl_compl_date is not null);
:
:
:
:   INSERT INTO core.well_test (surface_site_id, dataset_id, well_info, g10_id, g10_rrcid, g10_tst_date, g10_dly_gas,
g10_dly_cond, g10_dly_wtr,
:
:   g10_siwh_press, g10_shut_cd, g10_flow_press, g10_bhp, g10_shut_press, g10_opr,
g10_wh_pres_test_date)
:
:   SELECT distinct
:
:   core.generate_uuid('MLKay','WB','42' || lpad('' || W.api_number, 8,'0') || '0000') AS surface_site_id,
:
:   dsetUUID,
:
:   gas_rrcid,
:
:   G.g10_id,
:
:   G.g10_rrcid,
:
:   G.g10_tst_date,
:
:   G.g10_dly_gas,
:
:   G.g10_dly_cond,
:
:   G.g10_dly_wtr,
:
:   G.g10_siwh_press,
:
:   G.g10_shut_cd,
:
:   G.g10_flow_press,
:
:   G.g10_bhp,
:
:   G.g10_shut_press,
:
:   g.g10_opr,
:
:   G.g10_wh_pres_test_date
:
:   from wb.gse10m G, (select distinct seg.gas_rrcid, seg.api_number from wb.wb_well_id_seg seg,
wb.wb_new_location_seg nl where gas='G' and nl.api_number=seg.api_number) as W
:
:   where G.g10_rrcid=W.gas_rrcid;
:
:
:
:
: UPDATE core.surface_site SET shape = geometryfromtext('POINT(' || longitude || ' ' || latitude ||)'), 4326) WHERE dataset_id

```

```

= dsetUUID;
: UPDATE core.surface_site SET county_id=c.id FROM core.ref_county c WHERE ST_Contains(c.geometry,shape) and
dataset_id = dsetUUID;
:
: RETURN 0;
: END;
:

```

Source Code for Re-Creation of All Materialized Views Transform Operations

```

: BEGIN
:
: -- recreate all materialized views in core:
: RAISE INFO 'PostgreSQL is now dropping staging.beg_reservoir_view_materialized';
: perform staging.drop_matview('staging.beg_reservoir_view_materialized');
: RAISE INFO 'staging.beg_reservoir_view_materialized has been dropped.';
:
: RAISE INFO 'PostgreSQL is now dropping staging.beg_well_view_materialized';
: perform staging.drop_matview('staging.beg_well_view_materialized');
: RAISE INFO 'staging.beg_well_view_materialized has been dropped.';
:
: RAISE INFO 'PostgreSQL is now dropping staging.cornell_well_view_materialized';
: perform staging.drop_matview('staging.cornell_well_view_materialized');
: RAISE INFO 'staging.cornell_well_view_materialized has been dropped.';
:
: RAISE INFO 'PostgreSQL is now dropping staging.smu_bht_view_materialized';
: perform staging.drop_matview('staging.smu_bht_view_materialized');
: RAISE INFO 'staging.smu_bht_view_materialized has been dropped.';
:
: RAISE INFO 'PostgreSQL is now dropping staging.smu_hf_view_materialized';
: perform staging.drop_matview('staging.smu_hf_view_materialized');
: RAISE INFO 'staging.smu_hf_view_materialized has been dropped.';
:
: RAISE INFO 'PostgreSQL is now dropping staging.tt_well_view_materialized';
: perform staging.drop_matview('staging.tt_well_view_materialized');
: RAISE INFO 'staging.tt_well_view_materialized has been dropped.';
:
: RAISE INFO 'PostgreSQL is now dropping staging.und_rad_view_materialized';
: perform staging.drop_matview('staging.und_rad_view_materialized');
: RAISE INFO 'staging.und_rad_view_materialized has been dropped.';
:
: RAISE INFO 'PostgreSQL is now dropping staging.und_tc_view_materialized';
: perform staging.drop_matview('staging.und_tc_view_materialized');
: RAISE INFO 'staging.und_tc_view_materialized has been dropped.';
:

```

```

: RAISE INFO 'PostgreSQL is now dropping staging.und_td_view_materialized';
: perform staging.drop_matview('staging.und_td_view_materialized');
: RAISE INFO 'staging.und_td_view_materialized has been dropped.';
:
:
:
: RAISE INFO 'PostgreSQL is now creating matview staging.beg_reservoir_view_materialized';
: perform staging.create_matview('staging.beg_reservoir_view_materialized','staging.beg_reservoir_view');
: RAISE INFO 'staging.beg_reservoir_view_materialized has been created';
:
:
: RAISE INFO 'PostgreSQL is now creating matview staging.beg_well_view_materialized';
: perform staging.create_matview('staging.beg_well_view_materialized','staging.beg_well_view');
: RAISE INFO 'staging.beg_well_view_materialized has been created';
:
:
: RAISE INFO 'PostgreSQL is now creating matview staging.cornell_well_view_materialized';
: perform staging.create_matview('staging.cornell_well_view_materialized','staging.cornell_well_view');
: RAISE INFO 'staging.cornell_well_view_materialized has been created';
:
:
: RAISE INFO 'PostgreSQL is now creating matview staging.smu_bht_view_materialized';
: perform staging.create_matview('staging.smu_bht_view_materialized','staging.smu_bht_view');
: RAISE INFO 'staging.smu_bht_view_materialized has been created';
:
:
: RAISE INFO 'PostgreSQL is now creating matview staging.smu_hf_view_materialized';
: perform staging.create_matview('staging.smu_hf_view_materialized','staging.smu_hf_view');
: RAISE INFO 'staging.smu_hf_view_materialized has been created';
:
:
: RAISE INFO 'PostgreSQL is now creating matview staging.tt_well_view_materialized';
: perform staging.create_matview('staging.tt_well_view_materialized','staging.tt_well_view');
: RAISE INFO 'staging.tt_well_view_materialized has been created';
:
:
: RAISE INFO 'PostgreSQL is now creating matview staging.und_rad_view_materialized';
: perform staging.create_matview('staging.und_rad_view_materialized','staging.und_rad_view');
: RAISE INFO 'staging.und_rad_view_materialized has been created';
:
:
: RAISE INFO 'PostgreSQL is now creating matview staging.und_tc_view_materialized';
: perform staging.create_matview('staging.und_tc_view_materialized','staging.und_tc_view');
: RAISE INFO 'staging.und_tc_view_materialized has been created';
:
:
: RAISE INFO 'PostgreSQL is now creating matview staging.und_td_view_materialized';
: perform staging.create_matview('staging.und_td_view_materialized','staging.und_td_view');
: RAISE INFO 'staging.und_td_view_materialized has been created';
:
:

```



```

: RETURN 0;
: END;
:

```

Source Code for Refresh Materialized Views Transform Operations

```

: DECLARE
:   matview ALIAS FOR $1;
:   entry staging.matviews%ROWTYPE;
: BEGIN
:
:   SELECT * INTO entry FROM staging.matviews WHERE mv_name = matview;
:
:   IF NOT FOUND THEN
:     RAISE EXCEPTION 'Materialized view % does not exist.', matview;
:   END IF;
:
:   EXECUTE 'DELETE FROM ' || matview;
:   EXECUTE 'INSERT INTO ' || matview || ' SELECT * FROM ' || entry.v_name;
:
:   UPDATE staging.matviews
:     SET last_refresh=CURRENT_TIMESTAMP
:     WHERE mv_name=matview;
:
:   RETURN;
: END
:

```

Source Code for SMU BHT Heat Flow Transform Operations

```

: DECLARE \r
:   dsetUUID uuid;\r
: BEGIN\r
: \r
:   select core.register_dataset('SMU','BHT',NULL) into dsetUUID;\r
: \r
:   INSERT INTO core.surface_site (id, dataset_id ,api, field, latitude, longitude, operator, surface_temperature, \r
:     elevation_gl,elevation_df,elevation_kb,basement_depth, average_heatflow, plugged, \r
: well_type,other_ids,site_name,"comment")\r
:     select distinct * FROM (SELECT \r
:       first_value(core.generate_uuid('SMU','SURFACE_SITE', to_char(longitude,'999.9999')) || \r
: to_char(latitude,'999.9999')) OVER (PARTITION BY latitude,longitude) AS id, \r
:       dsetUUID as dataset_id,\r
:       first_value(api) OVER (PARTITION BY latitude,longitude) AS api, \r
:       first_value(field_name) OVER (PARTITION BY latitude,longitude) as field_name, \r
:       first_value(latitude) OVER (PARTITION BY latitude,longitude) as latitude, \r

```

```

:         first_value(longitude) OVER (PARTITION BY latitude,longitude) as longitude, \r
:         first_value(company_name) OVER (PARTITION BY latitude,longitude) as operator, \r
:         first_value(surface_temp ) OVER (PARTITION BY latitude,longitude) as surface_temp, \r
:         first_value(case when elevation_measured='GL' then elevation_m else null end) OVER (PARTITION
BY latitude,longitude) as elevation_gl,\r
:         first_value(case when elevation_measured='DF' then elevation_m else null end) OVER (PARTITION
BY latitude,longitude) as elevation_df,\r
:         first_value(case when elevation_measured='KB' then elevation_m else null end) OVER (PARTITION
BY latitude,longitude) as elevation_kb,\r
:         first_value(basement_depth ) OVER (PARTITION BY latitude,longitude) as basement_depth,\r
:         first_value(well_ave_hf ) OVER (PARTITION BY latitude,longitude) as average_heatflow,\r
:         first_value(well_status) OVER (PARTITION BY latitude,longitude) as plugged,\r
:         first_value(well_type ) OVER (PARTITION BY latitude,longitude) as well_type,\r
:         first_value('permit ' || permit || '; FileNumber: ' || file_number || '; OperationName: ' || operation_name)
OVER (PARTITION BY latitude,longitude) as other_ids,\r
:         first_value(split_part(surface_interval_id,-,1) || '-BHT-' || split_part(surface_interval_id,-,3)) OVER
(PARTITION BY latitude,longitude) as site_name,\r
:         first_value(reference_data_set) OVER (PARTITION BY split_part(surface_interval_id,-
,1),split_part(surface_interval_id,-,3)) as original_source\r
:     FROM staging.smu_bht_data) Z;\r
: \r
:     UPDATE core.surface_site SET shape = geometryfromtext('POINT(' || longitude || ' ' || latitude ||)'), 4326) WHERE
dataset_id = dsetUUID;\r
:     UPDATE core.surface_site SET county_id=c.id FROM core.ref_county c WHERE ST_Contains(c.geometry,shape) and
dataset_id = dsetUUID;\r
: \r
: \r
:     INSERT INTO core.formation (surface_site_id, dataset_id, formation,formation_depth)\r
:     SELECT distinct\r
:         core.generate_uuid('SMU','SURFACE_SITE', to_char(longitude,'999.9999') || to_char(latitude,'999.9999')),\r
:         dsetUUID as dataset_id, \r
:         formation,\r
:         depth\r
:         FROM staging.smu_bht_data\r
:         where length(formation)>0;\r
: \r
:     INSERT INTO core.thermal_conductivity (id,surface_site_id, dataset_id,conductivity, resistance, lithology_info)\r
:     SELECT \r
:         core.generate_uuid('SMU','THERMAL_CONDUCTIVITY', to_char(longitude,'999.9999') ||
to_char(latitude,'999.9999') || ';' || coalesce(depth,0) || ';' || \r
:         row_number() OVER (partition by latitude, longitude, depth)) AS id, \r
:         core.generate_uuid('SMU','SURFACE_SITE', to_char(longitude,'999.9999') || to_char(latitude,'999.9999'))
AS surface_site_id, \r
:         dsetUUID,\r
:         K ,\r
:         resistance,\r
:         k_source \r
:     FROM staging.smu_bht_data\r

```

```

:      WHERE (K is not null) or (resistance is not null);\r
: \r
: \r
: \r
:      INSERT INTO core.bht_measurements (id,surface_site_id, dataset_id, bht,bht_corrected,bht_depth, drill_date,
date_type, time_since_circulation, bht_correction_type)\r
:      SELECT \r
:      core.generate_uuid('SMU','BHT', to_char(longitude,'999.9999') || to_char(latitude,'999.9999') || ';' ||
coalesce(depth,0) || ';' || \r
:      row_number() OVER (partition by latitude, longitude, depth)) AS id, \r
:      core.generate_uuid('SMU','SURFACE_SITE', to_char(longitude,'999.9999') || to_char(latitude,'999.9999'))
AS surface_site_id, \r
:      dsetUUID,\r
:      uncor_bht AS bht, \r
:      bhtcorrected_temp AS bht_corrected,\r
:      depth AS bht_depth,\r
:      drilling_complete AS drilldate,\r
:      'Completion' as date_type,\r
:      tsc AS time_since_circulation,\r
:      correction AS bht_correction_type\r
:      FROM staging.smu_bht_data\r
:      WHERE (uncor_bht is not null) or (bhtcorrected_temp is not null);\r
:      \r
:      INSERT INTO core.depth (surface_site_id, dataset_id,depth, depth_date)\r
:      SELECT distinct\r
:      core.generate_uuid('SMU','SURFACE_SITE', to_char(longitude,'999.9999') || to_char(latitude,'999.9999'))
AS surface_site_id, \r
:      dsetUUID,\r
:      depth as depth,\r
:      drilling_complete AS depth_date\r
:      FROM staging.smu_bht_data\r
:      WHERE (depth is not null);\r
:      \r
:      INSERT INTO core.well_dates (surface_site_id, dataset_id,drl_compl_date,drl_start_date)\r
:      SELECT distinct\r
:      core.generate_uuid('SMU','SURFACE_SITE', to_char(longitude,'999.9999') || to_char(latitude,'999.9999'))
AS surface_site_id, \r
:      dsetUUID,\r
:      drilling_complete as drl_compl_date,\r
:      drilling_start as drl_start_date\r
:      FROM staging.smu_bht_data\r
:      WHERE (depth is not null);\r
:      \r
:      -- need to have bht and tcu data \r
:      INSERT INTO core.heat_flow (id,bht_measurements_id, thermal_conductivity_id, surface_site_id, dataset_id,
gradient_corrected, heat_flow, interval_order, quality,\r

```

```

:         interval_top_depth, interval_bottom_depth)\r
:     SELECT \r
:         core.generate_uuid('SMU','HEAT_FLOW', to_char(longitude,'999.9999') || to_char(latitude,'999.9999') || ';' ||
coalesce(depth,0) || ';' || \r
:         row_number() OVER (partition by latitude, longitude, depth)) AS id, \r
:         (case when ((uncor_bht is not null) or (bhtcorrected_temp is not null)) then \r
:         core.generate_uuid('SMU','BHT', to_char(longitude,'999.9999') || to_char(latitude,'999.9999') || ';' ||
coalesce(depth,0) || ';' || \r
:         row_number() OVER (partition by latitude, longitude, depth)) \r
:         else null end) AS bht_measurements_id, \r
:         (case when ((K is not null) or (resistance is not null)) then \r
:         core.generate_uuid('SMU','THERMAL_CONDUCTIVITY', to_char(longitude,'999.9999') ||
to_char(latitude,'999.9999') || ';' || coalesce(depth,0) || ';' || \r
:         row_number() OVER (partition by latitude, longitude, depth)) \r
:         else NULL end) AS thermal_conductivity_id, \r
:         core.generate_uuid('SMU','SURFACE_SITE', to_char(longitude,'999.9999') || to_char(latitude,'999.9999'))
AS surface_site_id, \r
:         dsetUUID,\r
:         harrison_gradient AS gradient_corrected,\r
:         q_calculated AS heat_flow_corrected,\r
:         split_part(surface_interval_id,'-',4) AS interval_order,\r
:         quality as quality,\r
:         0 as interval_top_depth,\r
:         depth as interval_bottom_depth\r
:     FROM staging.smu_bht_data\r
:     WHERE ((harrison_gradient is not null) or (q_calculated is not null)) and ((K is not null) or (resistance is not null))
\r
:     and ((uncor_bht is not null) or (bhtcorrected_temp is not null));\r
: \r
: -- update materialized views in staging\r
: perform staging.refresh_matview('staging.smu_bht_view_materialized');\r
: perform core.update_all_matviews();\r
: \r
: RETURN 0;\r
: END;\r
: \r
:

```

Source Code for SMU EQ Heat Flow Transform Operations

```

: DECLARE \r
:     dsetUUID uuid;\r
: BEGIN\r
: \r
:     select core.register_dataset('SMU','HF',NULL) into dsetUUID;\r
: \r

```

```

:      INSERT INTO core.surface_site (id, dataset_id ,latitude, longitude, site_name, township, range, section, other_ids,
surface_temperature, elevation_gl,\r
:      tectonic_province,lithology,comment, ams, mintemp, maxtemp,location_notes, other_location_name)\r
:      SELECT \r
:      core.generate_uuid('SMU','SURFACE_SITE', gid) id, \r
:      dsetUUID as dataset_id,\r
:      n_lat as latitude,\r
:      w_long as longitude, \r
:      gid as site_name,\r
:      (regexp_split_to_array(twn_rng_sec,E'[-/*\s]'))[1] as township,\r
:      (regexp_split_to_array(twn_rng_sec,E'[-/*\s]'))[2] as range, \r
:      (regexp_split_to_array(twn_rng_sec,E'[-/*\s]'))[3] as section,\r
:      ('SMU: ' || gid) as other_ids, \r
:      surf_temp as surface_temperature,\r
:      collar_elevation as elevation_gl,\r
:      tect_prov as tectonic_province,\r
:      lit_info as lithology,\r
:      coalesce(comment_info,'') || ' ' || coalesce('water_table_depth: '||water_table,'') as comment,\r
:      area_or_ams_map as ams,\r
:      min_temp as mintemp,\r
:      max_temp as maxtemp,\r
:      location_notes as location_notes,\r
:      ('SMU: ' || gid || ' ' || hole_name) as other_location_name\r
:      FROM staging.smu2_hf_point WHERE length(gid::text) > 0;\r
: \r
:      UPDATE core.surface_site SET shape = geometryfromtext('POINT('|| longitude || ' ' || latitude ||)'), 4326) WHERE
dataset_id = dsetUUID;\r
:      UPDATE core.surface_site SET county_id=c.id FROM core.ref_county c WHERE ST_Contains(c.geometry,shape) and
dataset_id = dsetUUID;\r
: \r
:      INSERT INTO core.bht_measurements (id,surface_site_id, dataset_id, bht,bht_depth, drill_date)\r
:      SELECT distinct\r
:      core.generate_uuid('SMU','BHT', gid || ';' || coalesce(drill_depth,0)) AS id, \r
:      core.generate_uuid('SMU','SURFACE_SITE', gid) AS surface_site_id, \r
:      dsetUUID,\r
:      bot_temp AS bht, \r
:      drill_depth AS bht_depth,\r
:      meas_date AS drilldate \r
:      FROM staging.smu2_hf_point\r
:      WHERE (drill_depth is not null) and length(gid::text) > 0;\r
: \r
: \r
:      INSERT INTO core.depth (surface_site_id, dataset_id,depth_date,depth)\r
:      SELECT distinct\r

```

```

:         core.generate_uuid('SMU','SURFACE_SITE', gid) AS surface_site_id, \r
:         dsetUUID,\r
:         drill_date AS depth_date,\r
:         drill_depth as depth\r
:     FROM staging.smu2_hf_point\r
:     WHERE (drill_depth is not null) and length(gid::text) > 0;\r
:\r
:     INSERT INTO core.thermal_conductivity (id,surface_site_id, dataset_id,conductivity, conductivity_se,
conductivity_nb_samples, porosity)\r
:     SELECT distinct\r
:         core.generate_uuid('SMU','THERMAL_CONDUCTIVITY', gid || ';' || interval_id) AS id, \r
:         core.generate_uuid('SMU','SURFACE_SITE', gid) AS surface_site_id, \r
:         dsetUUID,\r
:         avg_tcu,\r
:         avg_tcu_std_err,\r
:         bulk_tcu_repeats, \r
:         porosity \r
:     FROM staging.smu2_hf_heatflow\r
:     where (avg_tcu is not null) and length(gid::text) > 0;\r
:\r
:\r
:     INSERT INTO core.aquifier (surface_site_id, dataset_id, water_table_depth, aquifier_temperature,
aquifier_temperature_se)\r
:     SELECT distinct\r
:         core.generate_uuid('SMU','SURFACE_SITE', P.gid) AS surface_site_id, \r
:         dsetUUID,\r
:         P.water_table,\r
:         H.aquifier_temp ,\r
:         H.aquifier_temp_std_err \r
:     FROM staging.smu2_hf_heatflow H JOIN staging.smu2_hf_point P ON h.gid=p.gid\r
:     where (P.water_table is not null) and length(P.gid::text) > 0;\r
:\r
:\r
:     \r
:     INSERT INTO core.heat_flow (id,surface_site_id, thermal_conductivity_id, bht_measurements_id,dataset_id,
interval_order,interval_top_depth, interval_bottom_depth,\r
:         gradient, gradient_se, gradient_corrected, gradient_corrected_se, heat_flow, heat_flow_se, heat_flow_corrected,
heat_flow_corrected_se,\r
:         heat_generation, heat_generation_se,heat_generation_nb_samples, quality)\r
:     SELECT distinct\r
:         core.generate_uuid('SMU','HEAT_FLOW', H.gid || ';' || H.interval_id || ';' || coalesce(H.depth,0) || ';' ||
coalesce(H.bottom,0)) AS id, \r
:         core.generate_uuid('SMU','SURFACE_SITE', H.gid) AS surface_site_id, \r
:         case when (H.avg_tcu is not null) then \r
:             core.generate_uuid('SMU','THERMAL_CONDUCTIVITY', H.gid || ';' || H.interval_id)\r
:         else null end, \r

```

```

:         case when (P.drill_depth is not null) then \r
:             core.generate_uuid('SMU','BHT', H.gid || ';' || coalesce(P.drill_depth,0)) \r
:             else null end,\r
:
:         dsetUUID,\r
:         H.interval_id,\r
:         H.depth,\r
:         H.bottom,\r
:         H.geoth_grad ,\r
:         H.geoth_grad_std_err ,\r
:         H.cgeoth_grad ,\r
:         H.cgeoth_grad_std_err ,\r
:         H.heat_flow ,\r
:         H.heat_flow_std_err ,\r
:         H.cheat_flow ,\r
:         H.cheat_flow_std_err,\r
:         H.heat_production ,\r
:         H.heat_production_std_err,\r
:         H.num_hg,\r
:         H.quality_code as quality \r
:     FROM staging.smu2_hf_heatflow H JOIN staging.smu2_hf_point P on H.gid=P.gid where length(P.gid::text) >
0;\r
: \r
: \r
: -- add references to site_related_resources\r
:     INSERT INTO core.site_related_resources (surface_site_id,dataset_id,shape,resource_id,"type",description,url,
source)\r
:     SELECT distinct\r
:         core.generate_uuid('SMU','SURFACE_SITE', Z.gid) as surface_site_id,\r
:         dsetUUID,\r
:         geometryfromtext('POINT('|| Z.w_long || ' ' || Z.n_lat ||)'), 4326) as shape,\r
:         PB.publication_uuid as resource_id,\r
:         'Publication' as "type",\r
:         substring(Z.info from 1 for 128) as description,\r
:         substring(Z.link from 1 for 128) as url,\r
:         'SMU' as source\r
:     FROM (select P.gid, P.w_long, P.n_lat, split_part(R.smu_id,' ',1) as source, trim( leading '0' from split_part(R.smu_id,'
',2)) as source_number, R.* \r
:         from (select \r
:             trim( leading ' ' from split_part(new_pub_ref,'/',1)) as ref1,\r
:             trim( leading ' ' from split_part(new_pub_ref,'/',2)) as ref2,\r
:             trim( leading ' ' from split_part(new_pub_ref,'/',3)) as ref3,\r
:             * from staging.smu2_hf_point where new_pub_ref!='#N/A' and length(gid::text) > 0) as P,
staging.smu2_hf_ref R\r
:         where (P.ref1=R.new_pub_ref or P.ref2=R.new_pub_ref or P.ref3=R.new_pub_ref) and
length(R.new_pub_ref)>0) Z, \r

```

```

:         core.publication PB\r
:     WHERE Z.source=PB.source and Z.source_number=PB.source_number;\r
: \r
: -- update materialized views in staging\r
: perform staging.refresh_matview('staging.smu_hf_view_materialized');\r
: perform core.recreate_all_matviews();\r
: \r
: RETURN 0;\r
: END;\r
: \r
:

```

Source Code for Texas Tech University Transform Operations

```

: \r
: DECLARE \r
:     dsetUUID uuid;\r
: BEGIN\r
: \r
:     select core.register_dataset('TEXAS_TECH','ALL',NULL) into dsetUUID;\r
:     \r
:     UPDATE staging.tt_well_data SET well_name=null where well_name="";\r
: \r
:     INSERT INTO core.surface_site (id, dataset_id ,api, site_name, field, lease_name, elevation_kb, latitude, longitude,
operator,plugged, district,well_type, other_location_name)\r
:         SELECT distinct core.generate_uuid('TEXAS_TECH','SURFACE_SITE',w.API), dsetUUID,\r
:             w.api, well_name || '(' || w.api || ')', field_name, s_lease_num, \r
:             rkb_elev/3.280839895, s_lat, s_long, operator, status, cast(district as int), type_code,\r
:             coalesce('surface bolk: '||s_blk,' ') \r
:             FROM staging.tt_well_data w where w.api is not null;\r
:     \r
:     INSERT INTO core.depth (surface_site_id, dataset_id, depth_date, depth, true_vertical_depth)\r
:         SELECT distinct core.generate_uuid('TEXAS_TECH','SURFACE_SITE',API), dsetUUID,\r
:             to_date(td_date, 'YYYYMMDD'), \r
:             to_number(td_md, '999999')/3.280839895, \r
:             to_number(tvd, '999999')/3.280839895\r
:             FROM staging.tt_well_data where api is not null;\r
: \r
: \r
:     INSERT INTO core.well_dates (surface_site_id, dataset_id,drl_start_date,status_date)\r
:         SELECT distinct\r
:             core.generate_uuid('TEXAS_TECH','SURFACE_SITE',API) AS surface_site_id, \r
:             dsetUUID,\r
:             to_date(spud_date, 'YYYYMMDD') as drl_start_date,\r

```



```

:         to_date(st_date, 'YYYYMMDD') as status_date\r
:     FROM staging.tt_well_data;\r
:\r
: /*\r
: -- adding description to relates resources from uncorrected bht\r
: INSERT INTO core.site_related_resources (surface_site_id, label, resource_id, url, dataset_id,"type",description,shape)\r
:     SELECT distinct core.generate_uuid('TEXAS_TECH','SURFACE_SITE',API),\r
:         file_name,\r
:         core.generate_uuid('TEXAS_TECH','WELL_LOG',file_name),\r
:         'https://geothermal.smu.edu/gtda/rest/fileSystem/getFile?uuid=' ||\r
core.generate_uuid('TEXAS_TECH','WELL_LOG',file_name),\r
:         dsetUUID, 'WELL_LOG', description, geometryfromtext('POINT('|| s_long || ' ' || s_lat ||)'), 4326)\r
\r
:     FROM (select L.api,L.file_name, D.description, W.s_lat,W.s_long from staging.tt_uncorrected_bht L \r
:     LEFT JOIN staging.tt_log_descriptions D ON L.file_name=D.file_name\r
:     LEFT JOIN staging.tt_well_data W ON W.api=L.api) Z;\r
:\r
: -- adding description to relates resources from tt_log_reports      \r
:\r
: INSERT INTO core.site_related_resources (surface_site_id, label, resource_id, url, dataset_id,"type",description,shape)\r
:     SELECT distinct core.generate_uuid('TEXAS_TECH','SURFACE_SITE',API),\r
:         file_name,\r
:         core.generate_uuid('TEXAS_TECH','WELL_LOG',file_name),\r
:         'https://geothermal.smu.edu/gtda/rest/fileSystem/getFile?uuid=' ||\r
core.generate_uuid('TEXAS_TECH','WELL_LOG',file_name),\r
:         dsetUUID, 'WELL_LOG', description, geometryfromtext('POINT('|| s_long || ' ' || s_lat ||)'), 4326)\r
\r
:     FROM (select L.api,L.file_name, D.description, W.s_lat,W.s_long from staging.tt_log_reports L \r
:     LEFT JOIN staging.tt_log_descriptions D ON L.file_name=D.file_name\r
:     LEFT JOIN staging.tt_well_data W ON W.api=L.api) Z;\r
: */\r
: -- adding uncorrected bht and depth:\r
:\r
: INSERT INTO core.bht_measurements (id,surface_site_id, dataset_id, bht,bht_depth)\r
:     SELECT distinct \r
:         core.generate_uuid('TEXAS_TECH','BHT',API || ';' || bht_depth || ';' || file_name || ';' || row_number() over\r
(partition by API, bht_depth, file_name)) AS id, \r
:         core.generate_uuid('TEXAS_TECH','SURFACE_SITE',API) AS surface_site_id, \r
:         dsetUUID, \r
:         bht, \r
:         bht_depth\r
:     FROM(\r
:         SELECT  API, file_name, \r
:             (cast(bht as double precision)-32)*5/9 AS bht, \r
:             cast(depth as double precision)/3.280839895 AS bht_depth\r

```

```

:      FROM staging.tt_uncorrected_bht where bht is not null and depth is not null\r
: \r
:      UNION \r
:      SELECT  API, file_name,          \r
:              (cast(bht1 as double precision)-32)*5/9 AS bht, \r
:              cast(depth1 as double precision)/3.280839895 AS bht_depth\r
:      FROM staging.tt_uncorrected_bht where bht1 is not null and depth1 is not null\r
: \r
:      UNION\r
:      SELECT      API, file_name,          \r
:              (cast(bht2 as double precision)-32)*5/9 AS bht, \r
:              cast(depth2 as double precision)/3.280839895 AS bht_depth\r
:      FROM staging.tt_uncorrected_bht where bht2 is not null and depth2 is not null\r
: \r
:      UNION \r
:      SELECT      API, file_name,          \r
:              (cast(bht3 as double precision)-32)*5/9 AS bht, \r
:              cast(depth3 as double precision)/3.280839895 AS bht_depth\r
:      FROM staging.tt_uncorrected_bht\r
:      where bht3 is not null and depth3 is not null\r
: \r
:      UNION \r
:      SELECT      API, file_name, \r
:              (cast(bht4 as double precision)-32)*5/9 AS bht, \r
:              cast(depth4 as double precision)/3.280839895 AS bht_depth\r
:      FROM staging.tt_uncorrected_bht\r
:      where bht4 is not null and depth4 is not null\r
: \r
:      UNION\r
:      SELECT      API, file_name,          \r
:              (cast(bht5 as double precision)-32)*5/9 AS bht, \r
:              cast(depth5 as double precision)/3.280839895 AS bht_depth\r
:      FROM staging.tt_uncorrected_bht\r
:      where bht5 is not null and depth5 is not null\r
: \r
:      UNION\r
:      SELECT      API, file_name,          \r
:              (cast(bht6 as double precision)-32)*5/9 AS bht, \r
:              cast(depth6 as double precision)/3.280839895 AS bht_depth\r
:      FROM staging.tt_uncorrected_bht\r
:      where bht6 is not null and depth6 is not null\r
: \r
:      UNION\r

```

```

:      SELECT      API, file_name,      \r
:      (cast(bht7 as double precision)-32)*5/9 AS bht, \r
:      cast(depth7 as double precision)/3.280839895 AS bht_depth\r
:      FROM staging.tt_uncorrected_bht\r
:      where bht7 is not null and depth7 is not null\r
: \r
:      UNION\r
:      SELECT      API, file_name,      \r
:      (cast(bht8 as double precision)-32)*5/9 AS bht, \r
:      cast(depth8 as double precision)/3.280839895 AS bht_depth\r
:      FROM staging.tt_uncorrected_bht\r
:      where bht8 is not null and depth8 is not null\r
: \r
:      UNION\r
:      SELECT      API, file_name,      \r
:      (cast(bht9 as double precision)-32)*5/9 AS bht, \r
:      cast(depth9 as double precision)/3.280839895 AS bht_depth\r
:      FROM staging.tt_uncorrected_bht\r
:      where bht9 is not null and depth9 is not null\r
: \r
:      UNION\r
:      SELECT      API, file_name,      \r
:      (cast(bht10 as double precision)-32)*5/9 AS bht, \r
:      cast(depth10 as double precision)/3.280839895 AS bht_depth\r
:      FROM staging.tt_uncorrected_bht\r
:      where bht10 is not null and depth10 is not null\r
: \r
:      UNION\r
:      SELECT      API, file_name,      \r
:      (cast(bht11 as double precision)-32)*5/9 AS bht, \r
:      cast(depth11 as double precision)/3.280839895 AS bht_depth\r
:      FROM staging.tt_uncorrected_bht\r
:      where bht11 is not null and depth11 is not null\r
: \r
:      UNION\r
:      SELECT      API, file_name,      \r
:      (cast(bht12 as double precision)-32)*5/9 AS bht, \r
:      cast(depth12 as double precision)/3.280839895 AS bht_depth\r
:      FROM staging.tt_uncorrected_bht\r
:      where bht12 is not null and depth12 is not null\r
: \r
: \r
:      UNION\r

```

```

:      SELECT      API, file_name,          \r
:      (cast(bht13 as double precision)-32)*5/9 AS bht, \r
:      cast(depth13 as double precision)/3.280839895 AS bht_depth\r
:      FROM staging.tt_uncorrected_bht\r
:      where bht13 is not null and depth13 is not null\r
: \r
:      UNION\r
:      SELECT      API, file_name,          \r
:      (cast(bht14 as double precision)-32)*5/9 AS bht, \r
:      cast(depth14 as double precision)/3.280839895 AS bht_depth\r
:      FROM staging.tt_uncorrected_bht\r
:      where bht14 is not null and depth14 is not null\r
: \r
:      UNION\r
:      SELECT      API, file_name,          \r
:      (cast(bht15 as double precision)-32)*5/9 AS bht, \r
:      cast(depth15 as double precision)/3.280839895 AS bht_depth\r
:      FROM staging.tt_uncorrected_bht\r
:      where bht15 is not null and depth15 is not null\r
:      UNION\r
:      SELECT      API, file_name,          \r
:      (cast(bht16 as double precision)-32)*5/9 AS bht, \r
:      cast(depth16 as double precision)/3.280839895 AS bht_depth\r
:      FROM staging.tt_uncorrected_bht\r
:      where bht16 is not null and depth15 is not null\r
:      UNION\r
:      SELECT      API, file_name,          \r
:      (cast(bht17 as double precision)-32)*5/9 AS bht, \r
:      cast(depth17 as double precision)/3.280839895 AS bht_depth\r
:      FROM staging.tt_uncorrected_bht\r
:      where bht17 is not null and depth15 is not null\r
:      UNION\r
:      SELECT      API, file_name,          \r
:      (cast(bht18 as double precision)-32)*5/9 AS bht, \r
:      cast(depth18 as double precision)/3.280839895 AS bht_depth\r
:      FROM staging.tt_uncorrected_bht\r
:      where bht18 is not null and depth15 is not null\r
:      ) Z where api is not null;\r
: \r
: -- adding corrected bht (sub-sea-floor and depth)\r
: INSERT INTO core.bht_measurements (id,surface_site_id, dataset_id, bht_corrected,bht_depth)\r
:      SELECT distinct core.generate_uuid('TEXAS_TECH','BHT',API || ';' || depth || ';' || row_number() over (partition by
API, depth)) AS id, \r

```

```

:         core.generate_uuid('TEXAS_TECH','SURFACE_SITE',API) AS surface_site_id, \r
:         dsetUUID,\r
:         vrt AS bht_corrected,\r
:         depth AS bht_depth\r
:     FROM \r
:     (SELECT C.api, C.vrt, (to_number(W.w_depth, '999999') + W.rkb_elev + C.ssf_depth_ft)/3.280839895 as depth \r
:     FROM staging.tt_corrected_bht C, staging.tt_well_data W\r
:     WHERE C.api=W.api AND C.vrt is not null and C.ssf_depth_ft is not null) Z;\r
:\r
:\r
: UPDATE core.surface_site SET shape = geometryfromtext('POINT('|| longitude || ' ' || latitude ||)'), 4326) WHERE dataset_id = dsetUUID;\r
: UPDATE core.surface_site SET county_id=c.id FROM core.ref_county c WHERE ST_Contains(c.geometry,shape) and dataset_id = dsetUUID;\r
:\r
: -- update materialized views in staging\r
: perform staging.refresh_matview('staging.tt_well_view_materialized');\r
: perform core.update_all_matviews();\r
:\r
: RETURN 0;\r
: END;\r
:

```

Source Code for UND Transform Operations

```

: DECLARE
:     dsetUUID uuid;
: BEGIN
:
:     select core.register_dataset('UND','ALL',NULL) into dsetUUID;
:
:     INSERT INTO core.surface_site (id, dataset_id, latitude, longitude, range, section, section_part, township,
:     tectonic_province, site_owner, site_name, elevation_gl)
:     select distinct
:         first_value(id) OVER (PARTITION BY id),
:         dsetUUID as dataset_id,
:         first_value(latitude) OVER (PARTITION BY id),
:         first_value(longitude) OVER (PARTITION BY id),
:         first_value(range) OVER (PARTITION BY id),
:         first_value(section) OVER (PARTITION BY
: id),
:         first_value(sectionpart)
:     OVER (PARTITION BY id),
:     first_value(township) OVER (PARTITION BY id),
:     first_value(tectonic_province) OVER (PARTITION BY id),

```

```

:
first_value(site_owner) OVER (PARTITION BY id),
:
first_value(site_name) OVER (PARTITION BY id),
:
first_value(elevation_gl) OVER (PARTITION BY id)
:
FROM
:
((select distinct
:
core.generate_uuid('UND','SURFACE_SITE', to_char(coalesce(rad.latdegree,tc.LatDegreeWGS84),'999.999') ||
to_char(coalesce(rad.longdegree,tc.LongDegreeWGS84),'999.999')) as id,
:
coalesce(tc.LatDegreeWGS84, rad.latdegree) as latitude, -- into surface_site
:
coalesce(tc.LongDegreeWGS84, rad.longdegree) as longitude, -- into surface_site
:
rad.township as township, -- ss.township
:
rad.range as range, -- ss.range
:
rad.section as section, -- ss.section
:
rad.sectionpart as sectionpart, -- ss.section_part
:
rad.tectonicprovince as tectonic_province, --ss.tectonic_province,
:
null as site_owner, -- ss.site_owner
:
tc.sitelocationname as site_name, -- ss.well_name,
:
cast(null as double precision) as elevation_gl -- tds.elevation_per_m
:
from staging.und_tc_data tc FULL OUTER JOIN staging.und_rad_data rad ON tc.observationuri=rad.observationuri)
:
UNION
:
(select distinct
:
core.generate_uuid('UND','SURFACE_SITE', tds.hole_number || ';' || tds.label_description) as id,
:
tds.latitude as latitude, -- into surface_site
:
tds.longitude as longitude, -- into surface_site
:
substring(tds.trs from 'T([0-9]+[NSEW]?)/') as township, -- ss.township
:
substring(tds.trs from 'R([0-9]+[NSEW]?)/') as range, -- ss.range
:
substring(tds.trs from 'S([0-9]+)') as section, -- ss.section
:
substring(tds.trs from 'S[0-9]+([A-Z]+)') as sectionpart, -- ss.section_part
:
null as tectonic_province, --ss.tectonic_province,
:
tds.company_owner as site_owner, -- ss.site_owner,
:
tds.location_name as site_name, -- ss.well_name,

```

```

:
tds.elevation_per_m as elevation_gl-- ss.elevation_gl
:
from staging.und_td_sites tds)) Z;
:
:
UPDATE core.surface_site SET shape = geometryfromtext('POINT('|| longitude || ' ' || latitude ||)'), 4326) WHERE dataset_id =
dsetUUID;
:
UPDATE core.surface_site SET county_id=c.id FROM core.ref_county c WHERE ST_Contains(c.geometry,shape) and
dataset_id = dsetUUID;
:
:
:
INSERT INTO core.formation (surface_site_id, dataset_id, formation,formation_type)
:
SELECT distinct
:
core.generate_uuid('UND','SURFACE_SITE', to_char(coalesce(rad.latdegree,tc.LatDegreeWGS84),'999.999')
:
|| to_char(coalesce(rad.longdegree,tc.LongDegreeWGS84),'999.999')) as surface_site_id,
:
dsetUUID as dataset_id,
:
tc.geologicformationname AS formation,
:
tc.geologicformationtype AS formation_type
:
from staging.und_tc_data tc FULL OUTER JOIN staging.und_rad_data rad ON tc.observationuri=rad.observationuri
:
where tc.geologicformationname is not null;
:
:
:
INSERT INTO core.bht_measurements (id,surface_site_id, dataset_id, bht,bht_depth, drill_date)
:
SELECT distinct
:
core.generate_uuid('UND','BHT', tds.hole_number || ';' || tds.label_description || ';' || tds.drill_depth_m) AS id,
:
core.generate_uuid('UND','SURFACE_SITE', tds.hole_number || ';' || tds.label_description) as surface_site_id,
:
dsetUUID,
:
tds.bht_c as bht,
:
tds.drill_depth_m AS depth,
:
tds.log_date AS depth_date
:
from staging.und_td_sites tds
:
where (tds.log_date is not null and tds.drill_depth_m is not null and tds.bht_c is not null);
:
:
INSERT INTO core.depth (surface_site_id, dataset_id, depth)

```

```

:
SELECT distinct
:
core.generate_uuid('UND','SURFACE_SITE', tds.hole_number || ';' || tds.label_description) as surface_site_id,
:
dsetUUID,
:
-- tds.log_date AS depth_date, -- don't put it in depth
:
tds.drill_depth_m AS depth
:
from staging.und_td_sites tds
:
where (tds.log_date is not null and tds.drill_depth_m is not null);
:
:
:
:
:
INSERT INTO core.thermal_conductivity (id,surface_site_id, dataset_id, laboratory, measurement_device,
measurement_pressure ,measurement_procedure,
:
sample_collection_date, sample_measurement_date, sample_depth,
:
sample_height, sample_width, sample_length, sample_shape, sample_temperature_diff, sample_type, conductivity,
permeability,comments)
:
SELECT distinct
:
core.generate_uuid('UND','THERMAL_CONDUCTIVITY', tc.observationuri) AS id,
:
core.generate_uuid('UND','SURFACE_SITE', to_char(coalesce(rad.latdegree,tc.LatDegreeWGS84),'999.999')
:
|| to_char(coalesce(rad.longdegree,tc.LongDegreeWGS84),'999.999')) as surface_site_id,
:
dsetUUID,
:
tc.laboratory as laboratory ,
:
tc.measurementdevice as measurement_device,
:
tc.measurementpressure as measurement_pressure ,
:
tc.measurementprocedure as measurement_procedure,
:
tc.samplecollectiondate as sample_collection_date,
:
tc.samplemeasurementdate as sample_measurement_date,
:
tc.sampledepth as sample_depth,
:
tc.sampledimensionsheight as sample_height,
:
tc.sampledimensionswidth as sample_width,
:
tc.sampledimensionslength as sample_length,
:
:

```



```

tc.sampleshape as sample_shape,
:
tc.sampletemperature as sample_temperature_diff,
:
tc.sampletype as sample_type,
:
tc.saturatedsampleconductivity as conductivity,
:
tc.permeability as permeability,
:
tc.notes as comments
:
from staging.und_tc_data tc FULL OUTER JOIN staging.und_rad_data rad ON tc.observationuri=rad.observationuri
:
WHERE (tc.observationuri is not null);
:
:
:
-- radioactivity info ties to heat production (which is in heat flow)
:
INSERT INTO core.radioactivity (id, surface_site_id, dataset_id, laboratory, measurement_procedure, measurement_device,
rock, specimen_label, potassium,thorium, uranium,comments)
:
SELECT distinct
:
core.generate_uuid('UND','RADIOACTIVITY', rad.observationuri) as id,
:
core.generate_uuid('UND','SURFACE_SITE', to_char(coalesce(rad.latdegree,tc.LatDegreeWGS84),'999.999')
:
|| to_char(coalesce(rad.longdegree,tc.LongDegreeWGS84),'999.999')) as surface_site_id,
:
dsetUUID,
:
case when (length(rad.laboratory)>0) then rad.laboratory else rad.laboratoryuri end as laboratory,
:
rad.measurementprocedure as measurement_procedure,
:
rad.RadioactivityMeasurementDevice as measurement_device ,
:
rad.rockname as rock,
:
rad.specimenlabel as specimen_label,
:
rad.potassium as potassium,
:
rad.thorium as thorium,
:
rad.uranium as uranium,
:
rad.notes as comments
:
from staging.und_tc_data tc FULL OUTER JOIN staging.und_rad_data rad ON tc.observationuri=rad.observationuri
:
WHERE (rad.observationuri is not null);
:
:
:
INSERT INTO core.heat_flow (id, thermal_conductivity_id, radioactivity_id, surface_site_id, dataset_id, heat_generation,

```

```

heat_generation_se)
:
SELECT distinct
:
core.generate_uuid('UND','HEAT_FLOW', rad.observationuri),
:
case when (tc.observationuri is not null) then core.generate_uuid('UND','THERMAL_CONDUCTIVITY', tc.observationuri)
:
else null end AS id,
:
core.generate_uuid('UND','RADIOACTIVITY', rad.observationuri) as radioactivity_id,
:
core.generate_uuid('UND','SURFACE_SITE', to_char(coalesce(rad.latdegree,tc.LatDegreeWGS84),'999.999')
:
|| to_char(coalesce(rad.longdegree,tc.LongDegreeWGS84),'999.999')) as surface_site_id,
:
dsetUUID,
:
rad.heatproduction_mwm3 AS heat_generation,
:
case when (length(rad.heatproductionuncertainty)=0) then null else cast(split_part(rad.heatproductionuncertainty,',',1) as
double precision) end AS heat_generation_se
:
from staging.und_tc_data tc FULL OUTER JOIN staging.und_rad_data rad ON tc.observationuri=rad.observationuri
:
WHERE (rad.observationuri is not null and rad.heatproduction_mwm3 is not null);
:
:
:
-- populate temperature series and precision logs:
:
INSERT INTO core.precision_log (id, surface_site_id, dataset_id, precision_log_source_id, log_date, log_depth)
:
select distinct
:
core.generate_uuid('UND','TD_CURVE', tds.hole_number || ';' || tds.label_description) as id,
:
core.generate_uuid('UND','SURFACE_SITE', tds.hole_number || ';' || tds.label_description) as surface_site_id,
:
dsetUUID,
:
tds.label_description as precision_log_source_id,
:
to_char(tds.log_date,'YYYY-MM-DD') as log_date,
:
tds.drill_depth_m as log_depth
:
from staging.und_td_sites tds;
:
:
INSERT INTO core.temperature_series (precision_log_id, dataset_id, depth, temperature)
:
select distinct
:
core.generate_uuid('UND','TD_CURVE', tds.hole_number || ';' || tds.label_description) as precision_log_id,
:
dsetUUID,

```

```

:
tdc.depth,
:
tdc.temperature
:
from staging.und_td_sites tds
:
LEFT OUTER JOIN staging.und_td_curve_points tdc ON (tds.location_name=tdc.location_name OR
tds.hole_number=tdc.hole_number OR tds.label_description=tdc.sheet_name)
:
WHERE (tds.sites_id is not null and tdc.depth is not null and tdc.temperature is not null);
:
:
-- add td curves to related resources:
:
INSERT INTO core.site_related_resources (surface_site_id, dataset_id, label, type,shape, source)
:
select distinct
:
core.generate_uuid('UND','SURFACE_SITE', tds.hole_number || ';' || tds.label_description) as surface_site_id,
:
dsetUUID as dataset_id,
:
tds.label_description as label,
:
'TD_CURVES' as type,
:
geometryfromtext('POINT('|| tds.longitude || ' ' || tds.latitude ||)'), 4326) as shape,
:
'UND' as source
:
from staging.und_td_sites tds;
:
:
-- update materialized views in staging
: perform staging.refresh_matview('staging.und_rad_view_materialized');
: perform staging.refresh_matview('staging.und_tc_view_materialized');
: perform staging.refresh_matview('staging.und_td_view_materialized');
: -- perform core.update_all_matviews();
: -- perform core.recreate_all_matviews();
:
: RETURN 0;
: END;
:
:

```

Core Transform Operations

Pulled April 7, 2014 using the following command:

```
psql -h sgtl1d1 -U postgres gtdata -c '\df+ core.*' > gtdata-core-functions.txt 2> /dev/null
```

Required signing in to sgtl1p1 with PuTTY. Also requires the userid for user name 'postgres' on sgtl1d1.

The command generates a very wide table format and was been modified for readability by removing the two right most columns, "Source Code" and "Description". The description column was not populated. The information appearing in the Source Code column now appears underneath the headings for each of the 11 different staging transform functions. The 11 sets of core operations include:

Schema	Name	Result data type	Argument data types	Type	Volatility	Owner	Language
core	bounding_box	TABLE(name character varying, site_name character varying, latitude character varying, longitude character varying, state_code character varying, county character varying, max_temperature character varying, depth character varying, datasource character varying, thermal_conductivity character varying, heat_flow character varying)	lower_latitude integer, lower_longitude integer, upper_latitude integer, upper_longitude integer	normal	volatile	postgres	plpgsql
core	create_matview	void	name, name	normal	volatile	postgres	plpgsql
core	drop_matview	void	name	normal	volatile	postgres	plpgsql
core	generate_uuid	uuid	datasource character varying, datatype character varying, resource_id character varying	normal	volatile	postgres	plpgsql
core	populateprecisionlog	boolean	uuid character varying, logsourceid character varying	normal	volatile	postgres	plpgsql
core	recreate_all_matviews	integer		normal	volatile	postgres	plpgsql
core	refresh_matview	void	name	normal	volatile	postgres	plpgsql
core	register_dataset	uuid	datasource character varying, datatype character varying, resource_id character varying	normal	volatile	postgres	plpgsql

core	remove_dataset	integer	datasource character varying, datatype character varying, resource_id character varying	normal	volatile	postgres	plpgsql
core	update_all_matviews	integer		normal	volatile	postgres	plpgsql
core	update_surface_site_detailed	boolean		normal	volatile	postgres	plpgsql

Source Code for bounding_box Transform Operations

```

: BEGIN
:   RETURN QUERY SELECT DISTINCT COALESCE(s.site_name, s.api), s.site_name, s.latitude,
s.longitude, st.state_abbr, c.county_name, bht.bht, GREATEST(d.depth, bht.bht_depth),
ds.datasource, tc.conductivity, hf.heat_flow
:   FROM core.surface_site s
:   LEFT JOIN core.ref_county c ON c.id = s.county_id
:   LEFT JOIN core.ref_state st ON st.id = c.state_id
:   LEFT JOIN core.depth d ON d.surface_site_id = s.id
:   LEFT JOIN core.bht_measurements bht ON bht.surface_site_id = s.id
:   LEFT JOIN core.datasets ds ON ds.id = s.dataset_id
:   LEFT JOIN core.thermal_conductivity tc ON tc.surface_site_id = s.id
:   LEFT JOIN core.heat_flow hf ON hf.surface_site_id = s.id
:   LEFT JOIN core.precision_log pl ON pl.surface_site_id = s.id
:   where s.latitude > $1 and s.longitude > $2 and s.latitude < $3 and s.longitude < $4;
:
: END;
:

```

Source Code for create_matview Transform Operations

```

: DECLARE
:   matview ALIAS FOR $1;
:   view_name ALIAS FOR $2;
:   entry core.matviews%ROWTYPE;
: BEGIN
:   SELECT * INTO entry FROM core.matviews WHERE mv_name = matview;
:
:   IF FOUND THEN
:     RAISE EXCEPTION 'Materialized view "%" already exists.',
:       matview;
:   END IF;
:
:   EXECUTE 'REVOKE ALL ON '
:   EXECUTE 'GRANT SELECT ON '
:   EXECUTE 'CREATE TABLE '
:   EXECUTE 'REVOKE ALL ON '

```

```

: EXECUTE 'GRANT SELECT ON '
:
: INSERT INTO core.matviews (mv_name, v_name, last_refresh)
: VALUES (matview, view_name, CURRENT_TIMESTAMP);
:
: RETURN;
: END
:

```

Source Code for drop_matview Transform Operations

```

: DECLARE
:   matview ALIAS FOR $1;
:   entry core.matviews%ROWTYPE;
: BEGIN
:   SELECT * INTO entry FROM core.matviews WHERE mv_name = matview;
:
:   IF NOT FOUND THEN
:     RAISE EXCEPTION 'Materialized view % does not exist.', matview;
:   END IF;
:
:   EXECUTE 'DROP TABLE '
:   DELETE FROM core.matviews WHERE mv_name=matview;
:
:   RETURN;
: END
:

```

Source Code for generate_uuid Transform Operations

```

: DECLARE
:   uri VARCHAR(128)
: BEGIN
:   IF (datasource IS NOT NULL)
:     THEN uri
:   END IF;
:   IF (datatype IS NOT NULL)
:     THEN uri
:   END IF;
:   IF (resource_id IS NOT NULL)
:     THEN uri
:   END IF;
:   RETURN uuid3(uri);
: END;
:

```

Source Code for populateprecisionlog Transform Operations

```
: DECLARE
:   precision_log_id uuid;
:   sourceid character;
:   temp_dataset_id uuid;
:   temp_log_deepest_depth double precision;
:   temp_log_deepest_temperature double precision;
:   temp_log_max_temperature double precision;
: BEGIN
:   precision_log_id=$1;
:   sourceid = $2;
:   select td.precision_log_id , td.dataset_id,max(td.depth) OVER (PARTITION BY td.precision_log_id),
:   last_value(td.temperature) OVER (PARTITION BY td.precision_log_id ORDER by
:   td.depth),max(td.temperature) OVER (PARTITION BY td.precision_log_id)
:   into precision_log_id, temp_dataset_id, temp_log_deepest_depth, temp_log_deepest_temperature,
:   temp_log_max_temperature from core.temperature_series td
:   where td.precision_log_id = precision_log_id;
:   IF NOT FOUND THEN
:     RETURN FALSE;
:   ELSE
:     -- SET AGGREGATE VALUES FOR LOG
:     UPDATE core.precision_log
:     SET dataset_id = temp_dataset_id,
:       precision_log_source_id = sourceid,
:       log_depth = temp_log_deepest_depth,
:       log_deepest_temperature = temp_log_deepest_temperature,
:       log_max_temperature = temp_log_max_temperature
:     WHERE id = precision_log_id;
:
:     -- LINK LOG TO SURFACE_SITE (WELL) IF MAPPING IS KNOWN
:     UPDATE core.precision_log l
:     SET surface_site_id = s.id
:     FROM core.ref_temperature_series_mapping m, core.surface_site s
:     WHERE l.precision_log_source_id = m.precision_log_source_id
:       AND l.dataset_id = m.data_source_id
:       AND s.dataset_id = m.data_source_id
:       AND s.source_id = m.surface_site_source_id
:       AND l.id = precision_log_id;
:     RETURN true;
:   END IF;
: END;
:
```

Source Code for recreate_all_matviews Transform Operations

```
: BEGIN
:
: -- recreate all materialized views in core
: RAISE INFO 'PostgreSQL is now dropping core.surface_site_county_state_materialized_view';
: perform core.drop_matview('core.surface_site_county_state_materialized_view');
: RAISE INFO 'core.surface_site_county_state_materialized_view has been dropped.';
:
: RAISE INFO 'PostgreSQL is now dropping core.template_borehole_materialized';
: perform core.drop_matview('core.template_borehole_materialized');
: RAISE INFO 'core.template_borehole_materialized has been dropped.';
:
: RAISE INFO 'PostgreSQL is now dropping core.template_fluidflux_materialized';
: perform core.drop_matview('core.template_fluidflux_materialized');
: RAISE INFO 'core.template_fluidflux_materialized has been dropped.';
:
: RAISE INFO 'PostgreSQL is now dropping core.template_heatflow_materialized';
: perform core.drop_matview('core.template_heatflow_materialized');
: RAISE INFO 'core.template_heatflow_materialized has been dropped.';
:
: RAISE INFO 'PostgreSQL is now dropping core.template_radiogenicheat_materialized';
: perform core.drop_matview('core.template_radiogenicheat_materialized');
: RAISE INFO 'core.template_radiogenicheat_materialized has been dropped.';
:
: RAISE INFO 'PostgreSQL is now dropping core.template_reservoir_materialized';
: perform core.drop_matview('core.template_reservoir_materialized');
: RAISE INFO 'core.template_reservoir_materialized has been dropped.';
:
: RAISE INFO 'PostgreSQL is now dropping core.template_thermalconductivity_materialized';
: perform core.drop_matview('core.template_thermalconductivity_materialized');
: RAISE INFO 'core.template_thermalconductivity_materialized has been dropped.';
:
: RAISE INFO 'PostgreSQL is now dropping core.template_wellfluid_materialized';
: perform core.drop_matview('core.template_wellfluid_materialized');
: RAISE INFO 'core.template_wellfluid_materialized has been dropped.';
:
: RAISE INFO 'PostgreSQL is now dropping core.template_wellheader_materialized';
: perform core.drop_matview('core.template_wellheader_materialized');
: RAISE INFO 'core.template_wellheader_materialized has been dropped.';
:
: RAISE INFO 'PostgreSQL is now dropping core.template_welllog_materialized';
: perform core.drop_matview('core.template_welllog_materialized');
: RAISE INFO 'core.template_welllog_materialized has been dropped.';
```



```

:
: RAISE INFO 'PostgreSQL is now dropping core.template_welltest_materialized';
: perform core.drop_matview('core.template_welltest_materialized');
: RAISE INFO 'core.template_welltest_materialized has been dropped.';
:
: RAISE INFO 'PostgreSQL is now creating matview core.surface_site_county_state_materialized_view';
: perform
core.create_matview('core.surface_site_county_state_materialized_view','core.surface_site_county_state_view');
: RAISE INFO 'core.surface_site_county_state_materialized_view has been created';
:
: RAISE INFO 'PostgreSQL is now creating matview core.template_borehole_materialized';
: perform core.create_matview('core.template_borehole_materialized','core.template_borehole');
: RAISE INFO 'core.template_borehole_materialized has been created';
:
: RAISE INFO 'PostgreSQL is now creating matview core.template_fluidflux_materialized';
: perform core.create_matview('core.template_fluidflux_materialized','core.template_fluidflux');
: RAISE INFO 'core.template_fluidflux_materialized has been created';
:
: RAISE INFO 'PostgreSQL is now creating matview core.template_heatflow_materialized';
: perform core.create_matview('core.template_heatflow_materialized','core.template_heatflow');
: RAISE INFO 'core.template_heatflow_materialized has been created';
:
: RAISE INFO 'PostgreSQL is now creating matview core.template_radiogenicheat_materialized';
: perform core.create_matview('core.template_radiogenicheat_materialized','core.template_radiogenicheat');
: RAISE INFO 'core.template_radiogenicheat_materialized has been created';
:
: RAISE INFO 'PostgreSQL is now creating matview core.template_reservoir_materialized';
: perform core.create_matview('core.template_reservoir_materialized','core.template_reservoir');
: RAISE INFO 'core.template_reservoir_materialized has been created';
:
: RAISE INFO 'PostgreSQL is now creating matview core.template_thermalconductivity_materialized';
: perform core.create_matview('core.template_thermalconductivity_materialized','core.template_thermalconductivity');
: RAISE INFO 'core.template_thermalconductivity_materialized has been created';
:
: RAISE INFO 'PostgreSQL is now creating matview core.template_wellfluid_materialized';
: perform core.create_matview('core.template_wellfluid_materialized','core.template_wellfluid');
: RAISE INFO 'core.template_wellfluid_materialized has been created';
:
: RAISE INFO 'PostgreSQL is now creating matview core.template_wellheader_materialized';
: perform core.create_matview('core.template_wellheader_materialized','core.template_wellheader');
: RAISE INFO 'core.template_wellheader_materialized has been created';
:
: RAISE INFO 'PostgreSQL is now creating matview core.template_welllog_materialized';

```

```

: perform core.create_matview('core.template_welllog_materialized','core.template_welllog');
: RAISE INFO 'core.template_welllog_materialized has been created';
:
: RAISE INFO 'PostgreSQL is now creating matview core.template_welltest_materialized';
: perform core.create_matview('core.template_welltest_materialized','core.template_welltest');
: RAISE INFO 'core.template_welltest_materialized has been created';
:
: RETURN 0;
: END;
:

```

Source Code for refresh_matview Transform Operations

```

: DECLARE
:   matview ALIAS FOR $1;
:   entry core.matviews%ROWTYPE;
: BEGIN
:
:   SELECT * INTO entry FROM core.matviews WHERE mv_name = matview;
:
:   IF NOT FOUND THEN
:     RAISE EXCEPTION 'Materialized view % does not exist.', matview;
:   END IF;
:
:   EXECUTE 'DELETE FROM '
:   EXECUTE 'INSERT INTO '
:
:   UPDATE core.matviews
:     SET last_refresh=CURRENT_TIMESTAMP
:     WHERE mv_name=matview;
:
:   RETURN;
: END
:

```

Source Code for register_dataset Transform Operations

```

: DECLARE
:   dset core.datasets%rowtype;
:   id UUID;
: BEGIN
:   IF datasource IS NULL
:     THEN RAISE EXCEPTION 'Missing information for parameter data source name';
:   END IF;
:

```

```

: SELECT * FROM core.datasets d
:   WHERE datasource = d.datasource
:   AND coalesce(d.datatype, "") = coalesce(datatype, d.datatype, "")
:   AND coalesce(d.resource_id, "") = coalesce(resource_id, d.resource_id, "")
: INTO dset;
:
: IF (dset.id IS null)
:   THEN
:     id
:     INSERT INTO core.datasets VALUES (id, datasource, datatype, resource_id);
:   ELSE
:     id
:   END IF;
:
: RETURN id;
: END;
:

```

Source Code for remove_dataset Transform Operations

```

: DECLARE
:   datasets CURSOR FOR SELECT d.id FROM core.datasets d
:   WHERE datasource = d.datasource
:   AND coalesce(d.datatype, "") = coalesce(datatype, d.datatype, "")
:   AND coalesce(d.resource_id, "") = coalesce(resource_id, d.resource_id, "");
: BEGIN
:   IF datasource IS NULL
:     THEN RAISE EXCEPTION 'Missing information for parameter data source name';
:   END IF;
:   --OPEN datasets;
:
:   FOR dsetid IN datasets LOOP
:     BEGIN DELETE FROM core.heat_flow WHERE dataset_id=dsetid.id; END;
:     BEGIN DELETE FROM core.radioactivity WHERE dataset_id=dsetid.id; END;
:     BEGIN DELETE FROM core.depth WHERE dataset_id=dsetid.id; END;
:     --BEGIN DELETE FROM core.surface_site_detailed WHERE dataset_id=dsetid.id; END;
:     --BEGIN DELETE FROM core.site_comments WHERE dataset_id=dsetid.id; END;
:     BEGIN DELETE FROM core.notes WHERE dataset_id=dsetid.id; END;
:     BEGIN DELETE FROM core.thermal_conductivity WHERE dataset_id=dsetid.id; END;
:     --BEGIN DELETE FROM core.filter_condition WHERE dataset_id=dsetid.id; END;
:     --BEGIN DELETE FROM core.ngds_heat_flow_mview WHERE dataset_id=dsetid.id; END;
:     --BEGIN DELETE FROM core.quality_issue_log WHERE dataset_id=dsetid.id; END;
:     BEGIN DELETE FROM core.publication WHERE dataset_id=dsetid.id; END;
:     --BEGIN DELETE FROM core.publication_reference WHERE dataset_id=dsetid.id; END;
:

```

```

:   --BEGIN DELETE FROM core.site_publication_links WHERE dataset_id=dsetid.id; END;
:   BEGIN DELETE FROM core.formation WHERE dataset_id=dsetid.id; END;
:
:   BEGIN DELETE FROM core.well_test WHERE dataset_id=dsetid.id; END;
:   BEGIN DELETE FROM core.well_dates WHERE dataset_id=dsetid.id; END;
:   BEGIN DELETE FROM core.well_info WHERE dataset_id=dsetid.id; END;
:
:   BEGIN DELETE FROM core.fluid_h10 WHERE dataset_id=dsetid.id; END;
:   BEGIN DELETE FROM core.fluid_h10h WHERE dataset_id=dsetid.id; END;
:   BEGIN DELETE FROM core.fluid WHERE dataset_id=dsetid.id; END;
:
:   BEGIN DELETE FROM core.temperature_series WHERE dataset_id=dsetid.id; END;
:   BEGIN DELETE FROM core.precision_log WHERE dataset_id=dsetid.id; END;
:
:   BEGIN DELETE FROM core.site_related_resources WHERE dataset_id=dsetid.id; END;
:
:
:   --BEGIN DELETE FROM core.heat_flow_publication_links WHERE dataset_id=dsetid.id; END;
:   BEGIN DELETE FROM core.bht_measurements WHERE dataset_id=dsetid.id; END;
:   BEGIN DELETE FROM core.casing WHERE dataset_id=dsetid.id; END;
:   BEGIN DELETE FROM core.aquifier WHERE dataset_id=dsetid.id; END;
:   BEGIN DELETE FROM core.interval WHERE dataset_id=dsetid.id; END;
:   BEGIN DELETE FROM core.surface_site WHERE dataset_id=dsetid.id; END;
:
:   BEGIN DELETE FROM core.reservoir WHERE dataset_id=dsetid.id; END;
: END LOOP;
:
: RETURN 0;
: END;
:

```

Source Code for update_all_matviews Transform Operations

```

: DECLARE
:   datasets CURSOR FOR SELECT d.id FROM core.datasets d
:   WHERE datasource = d.datasource
:   AND coalesce(d.datatype, '') = coalesce(datatype, d.datatype, '')
:   AND coalesce(d.resource_id, '') = coalesce(resource_id, d.resource_id, '');
: BEGIN
:   IF datasource IS NULL
:   THEN RAISE EXCEPTION 'Missing information for parameter data source name';
:   END IF;
:   --OPEN datasets;
:

```

```

:   FOR dsetid IN datasets LOOP
:       BEGIN DELETE FROM core.heat_flow WHERE dataset_id=dsetid.id; END;
:       BEGIN DELETE FROM core.radioactivity WHERE dataset_id=dsetid.id; END;
:       BEGIN DELETE FROM core.depth WHERE dataset_id=dsetid.id; END;
:       --BEGIN DELETE FROM core.surface_site_detailed WHERE dataset_id=dsetid.id; END;
:       --BEGIN DELETE FROM core.site_comments WHERE dataset_id=dsetid.id; END;
:       BEGIN DELETE FROM core.notes WHERE dataset_id=dsetid.id; END;
:       BEGIN DELETE FROM core.thermal_conductivity WHERE dataset_id=dsetid.id; END;
:       --BEGIN DELETE FROM core.filter_condition WHERE dataset_id=dsetid.id; END;
:       --BEGIN DELETE FROM core.ngds_heat_flow_mview WHERE dataset_id=dsetid.id; END;
:       --BEGIN DELETE FROM core.quality_issue_log WHERE dataset_id=dsetid.id; END;
:       BEGIN DELETE FROM core.publication WHERE dataset_id=dsetid.id; END;
:       --BEGIN DELETE FROM core.publication_reference WHERE dataset_id=dsetid.id; END;
:       --BEGIN DELETE FROM core.site_publication_links WHERE dataset_id=dsetid.id; END;
:       BEGIN DELETE FROM core.formation WHERE dataset_id=dsetid.id; END;
:
:       BEGIN DELETE FROM core.well_test WHERE dataset_id=dsetid.id; END;
:       BEGIN DELETE FROM core.well_dates WHERE dataset_id=dsetid.id; END;
:       BEGIN DELETE FROM core.well_info WHERE dataset_id=dsetid.id; END;
:
:       BEGIN DELETE FROM core.fluid_h10 WHERE dataset_id=dsetid.id; END;
:       BEGIN DELETE FROM core.fluid_h10h WHERE dataset_id=dsetid.id; END;
:       BEGIN DELETE FROM core.fluid WHERE dataset_id=dsetid.id; END;
:
:       BEGIN DELETE FROM core.temperature_series WHERE dataset_id=dsetid.id; END;
:       BEGIN DELETE FROM core.precision_log WHERE dataset_id=dsetid.id; END;
:
:       BEGIN DELETE FROM core.site_related_resources WHERE dataset_id=dsetid.id; END;
:
:
:       --BEGIN DELETE FROM core.heat_flow_publication_links WHERE dataset_id=dsetid.id; END;
:       BEGIN DELETE FROM core.bht_measurements WHERE dataset_id=dsetid.id; END;
:       BEGIN DELETE FROM core.casing WHERE dataset_id=dsetid.id; END;
:       BEGIN DELETE FROM core.aquifier WHERE dataset_id=dsetid.id; END;
:       BEGIN DELETE FROM core.interval WHERE dataset_id=dsetid.id; END;
:       BEGIN DELETE FROM core.surface_site WHERE dataset_id=dsetid.id; END;
:
:       BEGIN DELETE FROM core.reservoir WHERE dataset_id=dsetid.id; END;
:   END LOOP;
:
:   RETURN 0;
: END;
:

```

Source Code for update_surface_site_detailed Transform Operations

```
: DECLARE
:     row core.surface_site%ROWTYPE;
:     temp_deepest_temperature double precision;
:     temp_depth double precision;
: BEGIN
:
:     DELETE FROM core.surface_site_detailed;
:     FOR row IN SELECT * FROM core.surface_site
:     LOOP
:         SELECT depth
:         INTO temp_depth
:         FROM core.depth where surface_site_id=row.id;
:         SELECT max(log_deepest_temperature) OVER (PARTITION BY surface_site_id)
:         INTO temp_deepest_temperature
:         FROM core.precision_log where surface_site_id=row.id;
:         INSERT INTO core.surface_site_detailed(id, data_source_id,site_name ,source_id ,depth ,
deepest_temperature )
:         VALUES(row.id ,row.data_source_id ,row.site_name,row.source_id ,temp_depth
,temp_deepest_temperature);
:
:
:     END LOOP;
:     RETURN TRUE;
: END;
:
```

Appendix H: Reports testing of the SMU Node for Compliance

GEOSERVER REPORT

Operator: Paul Edwards and Dan Getman (NREL)

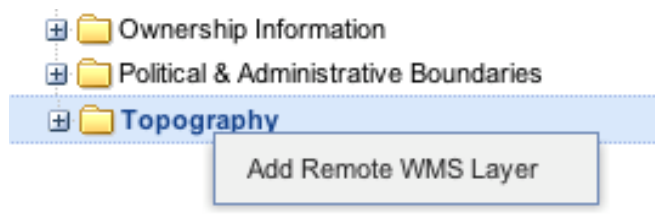
Date: Friday, February 07, 2014 1:52 PM

Report Title: SMU Data Layer – Compatibility Testing

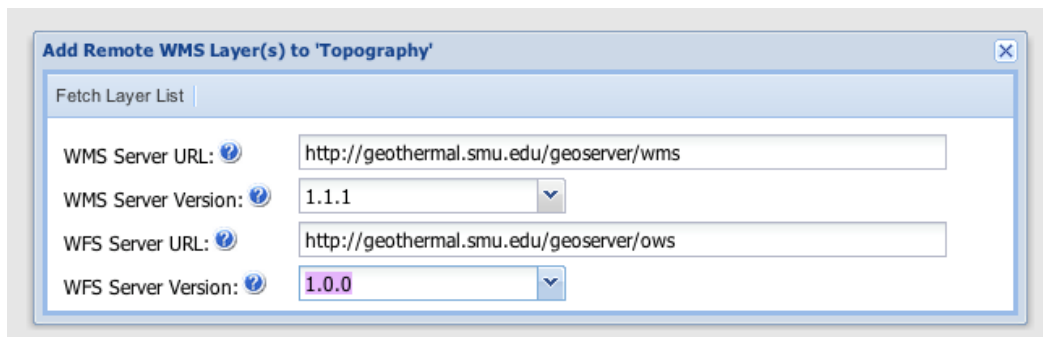
The layers identified on the web page at <http://geothermal.smu.edu/static/DownloadFilesButtonPage.htm> were tested with respect to integration with the Geothermal Prospector at http://maps.nrel.gov/gt_prospector. The tests were performed using the freely available client tools within the GT Prospector user interface. A brief description of the methods used is given, followed by a table of results for each layer

Method of Testing

Add the external layers to the Geothermal Prospector by right clicking on a layer in the layer tree and selecting “Add Remote WMS Layer”.



When presented with the form, complete the input fields as follows:



The click the ‘Fetch Layer List’ button in the upper left-hand corner. A GetCapabilities query is generated for the provide WMS server. The response is parsed and a list of layers is generated. The following list of available layers was presented.

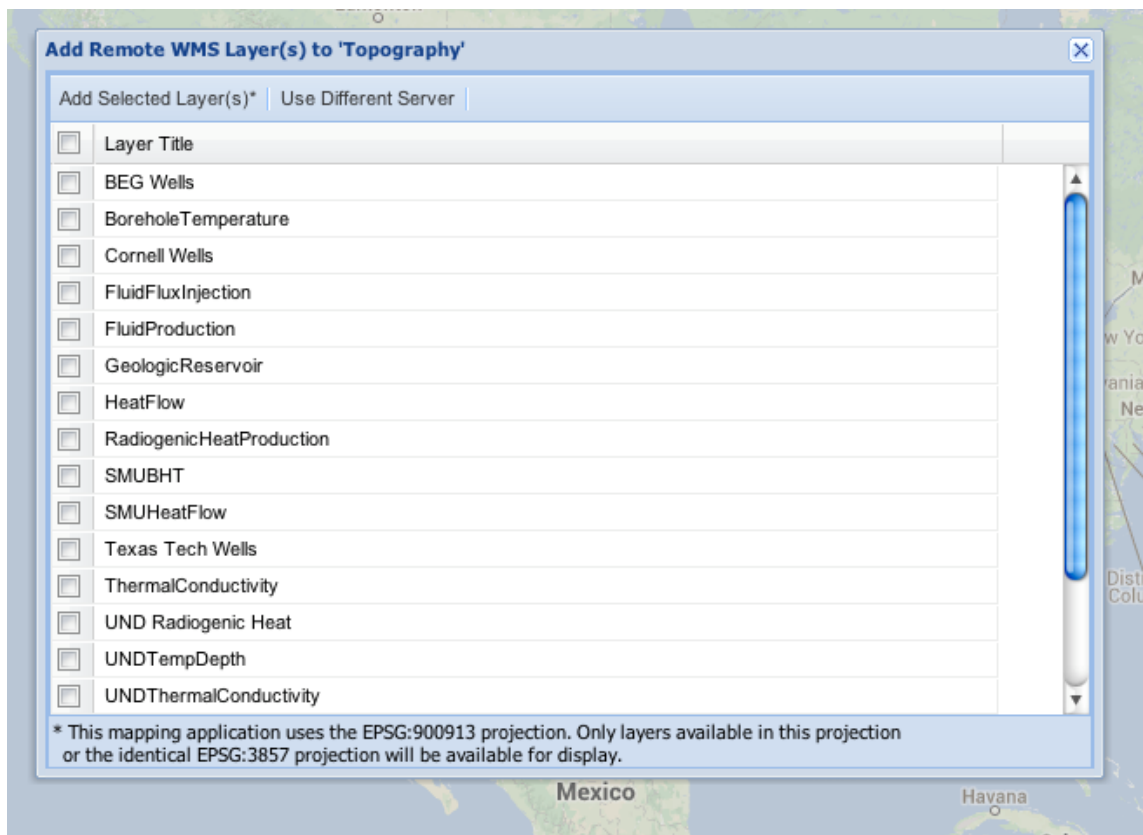


Figure 1 List of available layers

Select all and click the 'Add Selected Layer(s)' button. This will add the selected layers to the layer tree where they can be selected for viewing. Once visible on the map, layers may also be queried.

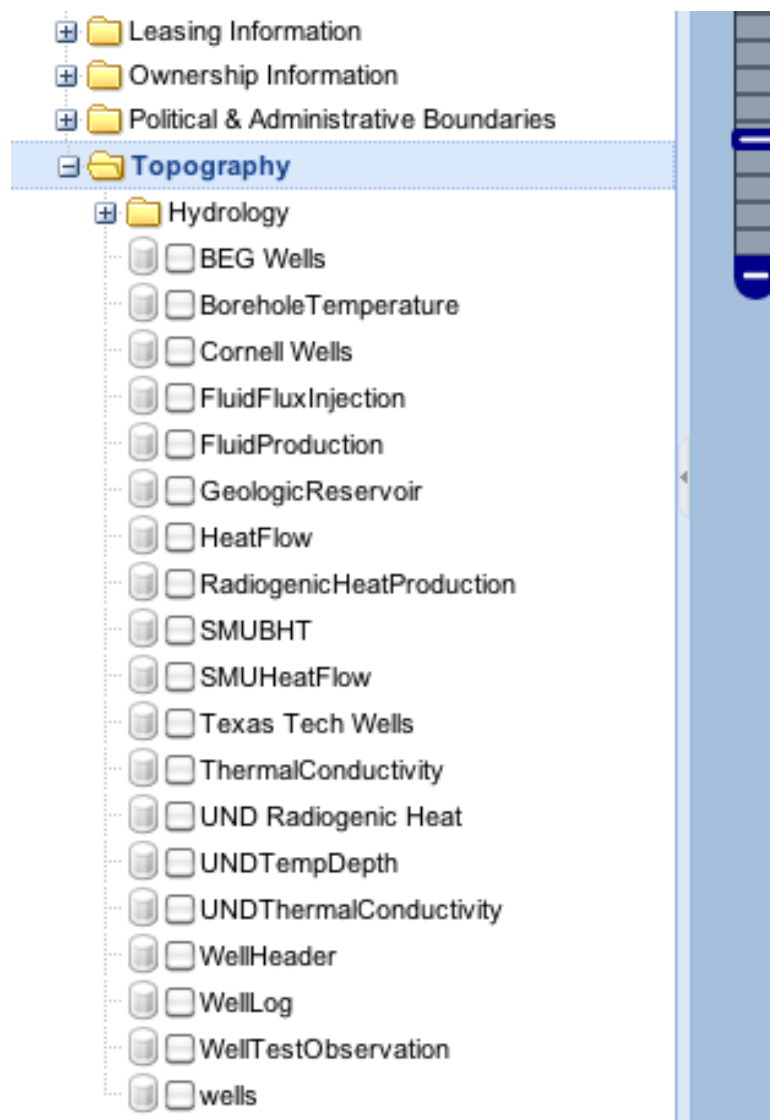


Figure 2 The layers after being added to the layer tree

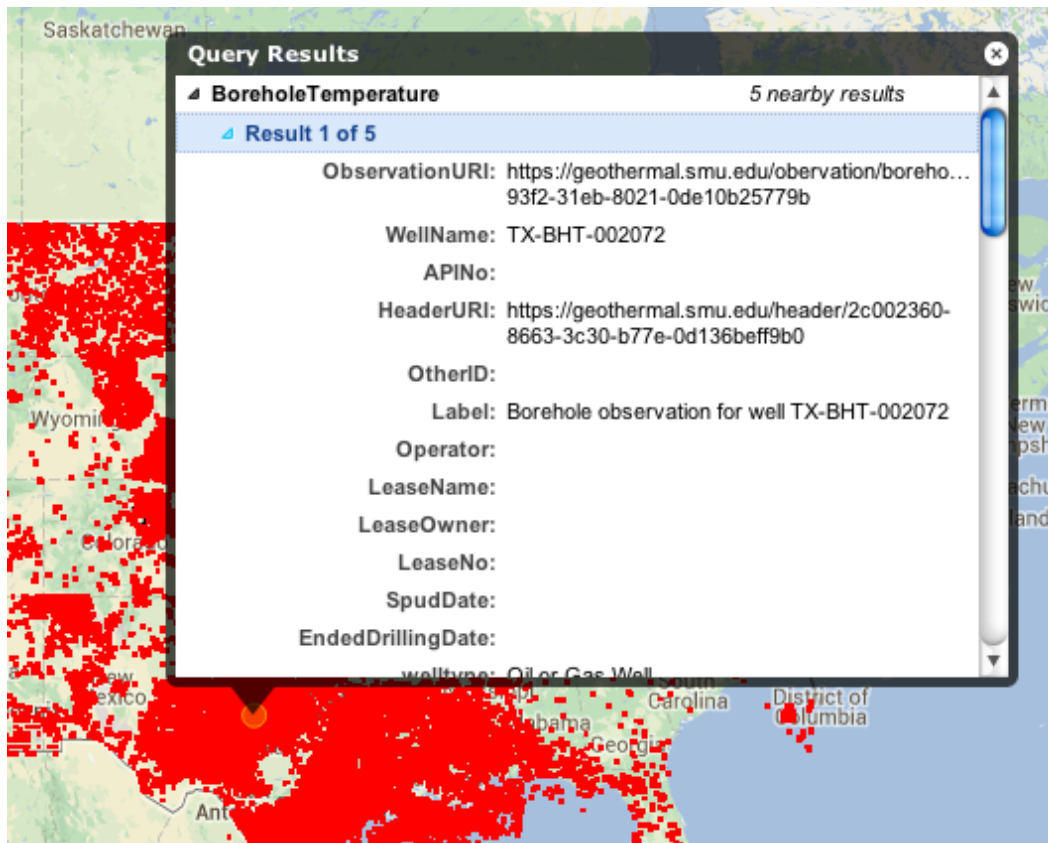


Figure 3 A successful query response

In instances where a layer failed to display or to query properly, the underlying requests and responses were interrogated to determine the cause of failure. The results of which are presented in the next section.

Test Results

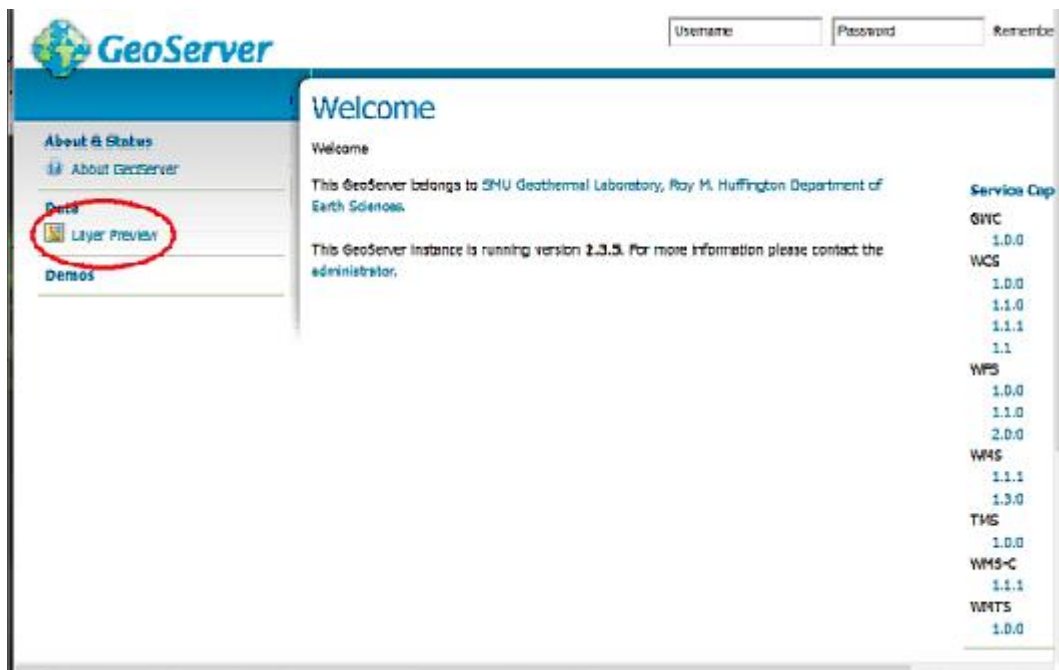
Description of Files Available for Download	Responds to GetCapabilities	Viewable (WMS GetMap)	Queryable (WFS GetFeature)
Aggregated Well Data	Y	Y	Y
Content Model Files Available for Download			
Borehole Temperature Observation	Y	Y	Y
Fluid Flux Injection and Disposal	Y	Y	Y
Geologic Reservoir	Y	Y	Y
Heat Flow Observation	Y	Y	Y
Radiogenic Heat Production	Y	Y	Y
Thermal Conductivity Observation	Y	Y	Y
Well Fluid Production Observation	Y	Y	Y
Well Header Observation	Y	Y	Y

Well Log Observation	Y	Y	Y
Well Test	Y	Y	Y
Contribution/Submission Files Available for Download (‘Tier 2’ Structured Data; not in Content Model Format)			
SMU’s Heat Flow Database based on Equilibrium log data and geothermal wells (note: also includes contributions from University of North Dakota)	Y	Y	Y
SMU’s Heat Flow Database from BHT Data	Y	Y	Y
BEG at University of Texas Consolidated Well Database Contribution	Y	Y	Y
BEG at University of Texas Reservoir Analysis Contributions	Y	Y	Y
Cornell University’s Heat Flow Database (NY and PA) Contributions	Y	Y	Y
Texas Tech University’s Off Shore Wells Database Contributions	Y	Y	Y
University of North Dakota’s Radiogenic Heat Generation Contributions.	Y	Y	Y
University of North Dakota’s Thermal Conductivity Data Contributions.	Y	Y	Y
University of North Dakota’s Temperature at Depth Data Contributions.	Y	Y	Y

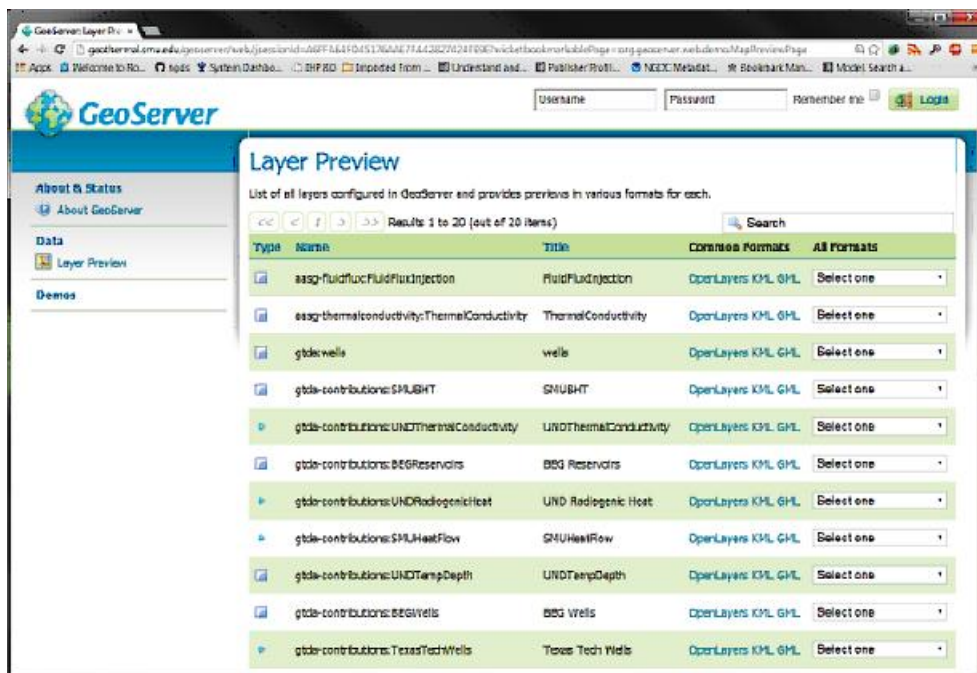
End of Report February 7, 2014.

HOW TO TEST SMU NODE DATA IN GEOSERVER

A convenient way for testing the SMU Node is to visit Geoserver's Layer Preview Page:
<http://geothermal.smu.edu:9000/geoserver/web/>



When clicking on the Layer Preview Link one can see a list of all datasets provided by the GTDA system.



WFS:

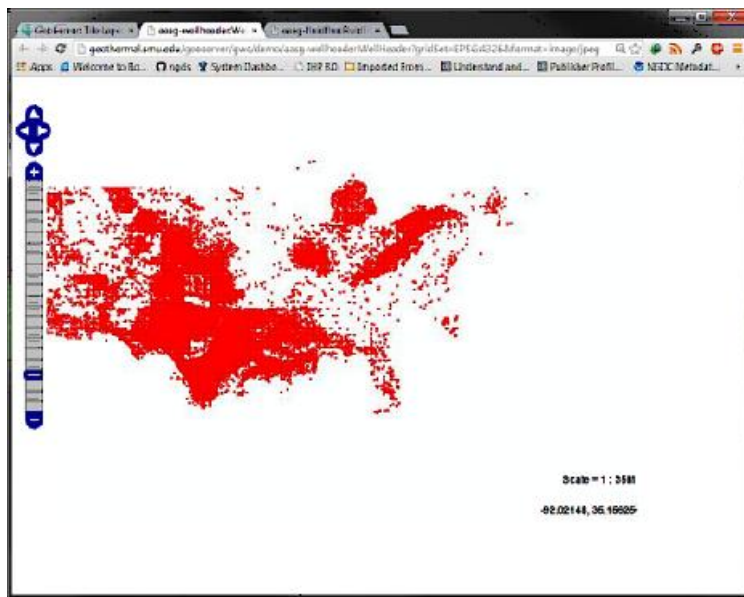
In order to check the availability of the WFS one may click on the Link "GML" of the individual datasets. This shows the first 50 entries of the dataset as XML. In order to review the data in a more human-readable form one may select the CSV output form from the "All Formats" from the pulldown selection.

In addition one may use the WFS command DescribeFeatureType in order to check the availability of a feature type description. In the following you find an example:
<http://geothermal.smu.edu:9000/geoserver/ows?service=wfs&request=DescribeFeatureType&version=1.0.0&typename=gtnda-contributions:BEGWells>

WMS:

Note that the datasets are partially very big. Therefore rendering them just-in-time is impossible. In order to receive acceptable answering times one has to access the cached version of the layers. The following command shows an example how to view a layer from the GeoCache:

<http://geothermal.smu.edu:9000/geoserver/gwc/demo/aasg-wellheader:WellHeader?gridSet=EPSG:4326&format=image/jpeg>



Appendix I: Response to DOE Close-Out Assessment

This Appendix I was written approximately 3 months after the rest of the final report and other appendices. It is primarily a response to the Close-Out Assessment conducted by the DOE and provided to SMU. The Close-Out Assessment summarized observations of the Technical Monitoring Team (TMT) regarding how the project goals and objectives of Project DE-EE0002852 for the Aggregation of Geothermal Data for Submission to the National Geothermal Data System were accomplished. This Appendix I is organized into the following sections:

- A. A response to observations in the assessment that may warrant additional clarification,
- B. Additional detail with regard to quality reviews of the data performed by the team,
- C. Specifics on some limitations of the system such that users or systems programmers will be informed, and
- D. Concluding statement.

A. Response to the Assessment

Overall, the assessment was very favorable, as stated in the opening pages:

“Overall, with one notable exception, performance on this project has been exemplary. As a result of this project large amounts of data of great value to the geothermal community have become freely available through NGDS and an application hosted at SMU.

The exception marring the otherwise superior performance of the project concerns the software delivered to manage and provide access to the data. The SOPO clearly states that the software would be “open, extensible and scalable,” however it is actually closed-source and rights to it are retained by the subcontractor that developed it. Thus it appears that the only means to extend or otherwise modify the software is to hire the same subcontractor to do so. In the opinion of the TMT this clearly contradicts the clear intent of the “open, extensible ...” phrasing in the SOPO. This issue primarily affects SOPO tasks 3, 14 and 16.”

‘Open’ Discussion

The GTDA system is founded on open source products. Indeed, the architecture was modified in the early months to make use of open source operating system (from Windows to Linux), database (from Oracle to Postgres), and other key components (GeoServer, Apache Redhat, Solr, GeoPortal). Data are delivered from the system using OGC protocols and standards (WMS, WFS) none of which require purchase of proprietary software by the users or the data submitters. The datasets and publications use the OGC implementation for catalog services for discovering data across the network (CSW). Clearly, in hindsight, both the initial proposal and the award should have been more explicit in word choice since ‘open and scalable’ meant ‘open source’ to one of the Technical Monitoring Team members (TMT), but did not to the project team during contract negotiations. Siemens is not contractually bound to provide the code as ‘open source’ but is considering designating it as such. SMU Geothermal Lab personnel do have access to the server where the Java scripts and SQL commands reside. A point of clarification regarding the impact of this warranted: even if Siemens does designate the code as ‘open source’, SMU Geothermal Laboratory personnel are still not in a position to modify the code without significant assistance from

professional software programmers. Even if designated ‘open source’, code modifications by *any* organization are unlikely without ongoing financial support. The ARRA funding that sponsored this project disallowed funding beyond the stated term. If future funding to support ongoing software development or code modifications was available, this would be a more important issue.

Final Report Location

The assessment makes reference to the electronic location of the final report on an SMU Departmental FTP site. This is where the draft version was placed for use during the assessment process. The final report will be uploaded into the SMU Node of the NGDS and will be accessible via http://geothermal.smu.edu/static/DE-EE0002852SMU_NGDS_Final_Report_2014.pdf. The final version, which includes this Appendix I, will also be available via OSTI. Searching on the contract number of ‘DE-EE0002852’ and ‘final report’ should be sufficient.

Data Quality

The data quality work effort was presented in an online meeting on October 31, 2013. During the meeting, SMU walked DOE personnel through specific examples of the review process, identification of errors in data, and identification of errors in mapping. As requested in the assessment (see quote below), this Appendix I will expand upon the discussion of quality in the main body of the final report to include the information from the October presentation. From the assessment:

“Task 2. Definition of data quality standards

Each team will formulate quality standards and domain knowledge for their data sources that can be translated to automated quality assurance mechanisms such as rules and statistical tests. In addition to mechanisms to identify quality problems, the experts should also identify potential remedies such as removal of values, interpolation from neighboring data items, etc. One milestone will be a report submitted to the NGDS that specifies what data will be submitted to them and how it will be quality controlled.

Observations:

In general it appears there has been considerable effort expended in ensuring the quality of the data provided by this project. This included reconciliation of overlapping data, deduplication to reduce the occurrence of true duplicate publications and datasets, and application of a number of algorithms to assist in assessing the quality of the various data. These efforts will help lead to higher confidence in the data from this provider.

The data inventory and quality report is provided in Appendix I of the project final report.

[5]”

B. Data Inventory and Quality Report

“Quality” is mentioned throughout the final report (Task 2. Definition of data quality standards; Task 6. Quality control and improvement of data; Key Achievements: Data Quality Advances, Lessons Learned part 11: Data Quality Assessment is never ‘done’; Opportunities for Additional Research or Activities, etc.). If we have learned anything, it is that Quality Assurance (QA) is not a task that is ‘completed’ but is rather a process. While software tools and metrics can facilitate the identification of potential errors, we must not become reliant on such tools as a substitute for the human examination of the end product being fully submitted properly and outputted to the end user.

Quality of the data products offered by the SMU Geothermal Laboratory and our consortium is taken quite seriously. And, ‘quality’ doesn’t mean the same thing to everyone. When the geothermal researcher from a lab is collecting and analyzing data, they generally refer to data quality in the context of whether the information they submitted into the system is ‘reliable’. For instance, in realm of heat flow values, the source of the data is a key factor:

- If the heat flow value is from some sort of publication: is the source known, complete and reputable? This latter point is of course a matter of opinion, but if the full reference is available and provided, there is the possibility for future researchers to make their own determination. The same is true of published temperature measurements and thermal conductivity values, which are the two primary inputs in calculating heat flow. Thermal conductivity determination for a well based on a combination of a published value for different rock types combined with knowledge of the stratigraphy of the area are subject to judgment of the researcher.
- If a heat flow value is based on field or lab measurements: were the procedures and units of measure documented and the results repeatable? For instance, the SMU Geothermal Lab has ‘field notes’ going back into the 1960’s explaining conditions under which wells were logged for temperature. The thermal conductivity rock samples measured on the lab divided bar are conducted according to a defined procedure and are compared to control samples and run multiple times, often with different orientations to determine standard error.
- If a heat flow value is based on an interpretation of a measured value: were the interpretations and/or ‘corrections’ applied in a consistent well documented manner, such that future researchers could modify the interpretation of the raw data, if warranted? For example, geothermal gradients based on Bottom Hole Temperature values collected during hydrocarbon drilling are often ‘corrected’ to approximate in situ temperatures. The mathematical formula may differ depending upon the geology of the area (deep sedimentary basins versus areas with steep terrain are handled differently). The correction formula may need to be modified to bring BHT estimated gradients in sync with more reliable equilibrium logged wells in the vicinity.

In addition to the source of information (published, measured, interpreted) the density of the data is another key parameter in assessing the reliability of data. Again, in the context of heat flow values, equilibrium temperature measurements are generally accepted as more reliable than temperature gradients based on a single BHT value. In addition to continuous measurements throughout the wellbore, they are less subject to the effects of drilling. Consider however a shallow equilibrium logged well of 40 meters with no other nearby wells: regardless of the accuracy of the gradient of the first 40 meters, extrapolating that gradient to a kilometer, no matter how accurate for the first 40 meters, might

not be more reliable than a gradient based on a BHT kilometers deep with other values collected at different depths within a narrowly confined area.

Published Papers on Quality

Initial endeavors to assign quality metrics with an automated program based on multiple criteria were met with interest. SMU's Maria Richards created a program in [MatLab](#) to assign a numerical quality rating to heat flow data based on the type of temperature reading (BHT or equilibrium), the depth(s) of temperature readings, data density of gradient values, corrections applied, thermal conductivity reliability (published values, regional estimates based on stratigraphy, measured samples), etc. A 2012 Richards et al. GRC paper summarizing this effort is also [available](#). By making the program available for download, other labs may use the code to analyze their heat flow values, providing a means to offer a consistent, documented reliability assignment. Even if they choose not to use the automatic program, it provides an opportunity for the researchers to consider the many factors that impact reliability.

The SMU Geothermal Laboratory has prepared a paper focused on data quality of NGDS data for presentation at this fall's 2014 GRC Annual Meeting in Portland. It reviews data from multiple NGDS participants as well as the original source documents for a region in the Cascades. The conclusion is that this improved availability of information provides researchers with both an opportunity and a responsibility to review and improve upon the data.

Screen Shot Images from October 2013 Presentation

The SMU Geothermal Laboratory presented the quality review process on October 31, 2013 in an online meeting in which all participants could see a workbook containing an example of the Quality Assurance Process. The DOE specifically requested that the contents of that presentation be incorporated into the Data Inventory and Quality Report. The process identifies error in the data submission as well as in the data delivery out of the system to the NGDS. The workbook had 5 worksheets:

Tab 1: Process

Tab 2: Content Model Format

Tab 3: SMU BHT Submission

Tab 4: SMU Side By Side Example

Tab 5: Cornell Summary Example

The first tab contained only text, and is thus easy to display in report format. The worksheets on tabs 2-5 do not lend themselves well to report format, so screen images from portions of those worksheets will be used to describe the intent.

Tab 1: Process

Quality Analysis (QA) is an iterative process involving the review and validation of data submitted by the different providers, as well as a review and validation of data delivered as output from the system. When items requiring change or correction are identified and corrected, the process must be repeated to ensure completion and to verify that no additional issues are present.

The SMU Node of the NGDS supports 10 different content models populated from a database that contains aggregated data from up to 10 or more incoming file structures. Content models populated by a single file from a single data contributor are relatively straightforward to review, but never-the-less time

consuming to review line by line. However, most of the content models have source data coming from multiple providers, in multiple formats which are considerably more challenging to review.

Potential errors are possible at many stages of collection and submission:

- a) at the time of the data collection and analysis (an incorrectly keyed in value, a formula error, a mis-sorted spreadsheet, etc.),
- b) at the time of data submission (reversing two columns of data, uploading a version without the most recent information, formulas rather than numerical values in a field, an incompatible field such as “N/A” or “?” in a field expected to contain numerical data, etc.),
- c) at the time of parameter mapping¹ the submitted files to the tables of aggregated data within the GTDA database,
- d) at the time of mapping information from the tables of aggregated data within the GTDA database to the various supported content models,
- e) in the delivered implementation of the content models (conformity to the content models with regard to required fields, layout, definition, etc.).

The errors related to either of the two mapping exercises (c and d in the list above) can be particularly complex because often these are not a question of an accidental association of one field to another, such as a reversal of latitude and longitude, but rather a miscommunication of what a particular field of submitted data is intended to communicate. When this is the case, the resolution can be complex, especially if the parameter in question is one that populates multiple WFS² content model scripts. Much of the table structure within the GTDA software was determined before the content models were decided upon, meaning that there is frequently not a 1:1 relationship between the fields of data within GTDA and the expected field for WFS content model output.

On the surface it may seem that the incoming datasets contain the same information, but perhaps with slightly different names, but in fact we found that is not always the case. One submitter might label a field as ‘depth’ meaning one of multiple intermediate logged depths, whereas another might be referring to the total vertical depth (TVD), while another might be the actual total length (versus depth) of the wellbore. Still others have submitted calculated depths that approximate TVD, but are not the official TVDs as reported on a well header. Thus, when GTDA reports a ‘maximum depth’ for a given point, the question becomes from which of the potential depth fields from each of the providers should that be collected? Dates have been another area with considerable variation in the number collected per submitter, the naming conventions used to communicate the types of dates, the meaning intended for a given date field, etc.

SMU, with participation from Siemens and each of the data contributors, reviewed the data output files for each of the various content models. Reservoir Analysis was among the easiest to review:

- it has only one data contributor (BEG),
- the submitted file layout is nearly identical to the content model layout meaning there is little ambiguity about which fields from the incoming file should populate the WFS for the content model, and
- the reservoir analysis input file from BEG only feeds into the reservoir analysis content model.

¹ In this context, ‘mapping’ means associating a parameter (data field) from the provider’s submission dataset with a data field in the GTDA database, or from the GTDA database to a data field in a content model.

² Web Feature Services

This is in contrast to the majority of the supported content models, which have:

- Input files from multiple data contributors aggregated into the content model output
- Each submitter's input file has a unique file layout, no two of which contain the same fields, or the same order, or even the same vocabulary for file headings
- The same input files populate multiple WFS content models in a “many to many” relationship (example: ‘depth’ populates well header, heat flow, borehole temperature, etc.)

The Borehole Temperature content model is a good example. This content model contains fields such as: location, depth, temperature, date, well owner/operator information etc. These fields are populated by values aggregated from SMU's BHT heat flow dataset, SMU/UND's Equilibrium well log heat flow dataset, Cornell's NY and PA BHT dataset, MLKay Technologies' oil and gas industry database for Texas, BEG's Master Wells dataset for Texas, Texas Tech's off shore BHT data set, and values from UND's temperature depth files. Every one of these incoming datasets uses different terminology, layout, order, etc. which are reconciled within the GTDA application. Additionally, these same input files also populate the well header, heat flow, and other templates. Thus, when a modification to the mapping for one submitter's depth parameter is required, it generally affects multiple content model output files.

While time intensive, the approach the team took was successful in identifying and correcting numerous issues with both the data submissions as well as the mapping. Steps included:

1. From the CSV file of the WFS output for a given content model, select a subset of records, ensuring that multiple rows from each contributing dataset are included. Select a variety of sites, making sure to include a mix of sites with single entries per site as well as those with multiple records for the same site. In our experience, the latter tend to be more prone to mapping issues than single entry sites. Copy these rows into a spreadsheet for further review.
2. Using location identifiers, pull the submitted data for the same sites from the various submission files. Copy these rows into the same spreadsheet as the selected WFS output records from step 1.
3. Review the selected rows to identify null values. Check the WFS output and the data input files to pull additional rows as needed, to ensure all columns are represented in the selected subset. Depending upon the number of null values in the data submission files, it generally requires a sample of at least 15 rows to find all columns populated for both the single row per site and multiple rows per site. Thus to review the borehole temperature content model, which is populated by 7 submitted files, over 100 rows from the Borehole Temperature WFS output CSV file are needed.
4. The columns from the input files will be in a different order and using different column headings than the output files. Compare the data for a single provider at a time, to determine if the intended input data appears in the expected field of the content model. This requires reviewing each field in the input file to assess if it should be represented in that particular content model as well as the reverse – that is, reviewing each field of the content model output to determine if that type of information was contained within that data provider's submission. Note that there will be instances where certain input fields are intentionally omitted from mapping. For example, if depth in meters and feet are both provided, it is unnecessary to read depth in feet into GTDA as a separate value. Similarly, state names are usually provided in the

input files, but are generated automatically using coordinates, and are thus not necessary to import into the GTDA tables. In contrast, some fields should not be omitted, either because they are required by the content model, or because the information was an important part of the data submission and should be accommodated within one of the more open ended content model fields, in the form of 'additional notes' or 'sources'.

5. Document where unexpected values are encountered:
 - a. the content model field is not populated even though the data submitted contained the requested information, perhaps with a different name.
 - b. the content model field is populating with an incorrect value of some sort.
 - c. the content model is populating one field from multiple input fields concatenated together, some of which are missing or incorrect.
6. To help understand if the issue is within the submitted file, in the parameter mapping of the submitted file into the GTDA, or from the GTDA tables into the content models:
 - a. check other entries from the same data submission to determine if the problem occurs throughout the entire dataset or only a subset.
(This may require a repeat of steps 1 and 2.)
 - b. check the affected field of the content model with values coming from other data providers to determine if the issue is limited to a given data submitter or all submissions.
 - c. Finally, check other content models making use of the same input value.
 - d. Complete all three of these checks (6a, 6b and 6c), as there are instances where there has been more than one source of a problem in the same record.
7. Errors in the data submission are reported to the data submitter for correction. In columns where an error is detected, use of filtering and sorting of the large original submission file often highlights other similar errors.
8. Errors to the parameter mapping are reported to Siemens. The mapping could potentially be from the submission file into the GTDA software, from the GTDA software into the content model, or a field omission.
9. Following correction of data submission and/or the mapping within GTDA, the file must be re-uploaded and revalidated. Not only does this ensure correction has taken place, but often times 'new' issues become apparent following a correction. The 'new' error could result from a mistake in the correction process, such as when only a portion of a spreadsheet is sorted, without all of the rows or columns selected. Or, the 'new' error could be one that was present initially, but not detectable until the initial error was corrected.

It is hard to determine ahead of time how long the QA process will take; the length of time for a single correction cycle is dependent upon many factors, including:

- the number of errors detected,
- the difficulty level in ascertaining where the source(s) of the problem is(are),
- the complexity associated with correcting the problem(s), and

- the availability of personnel able to perform each of the given tasks.

Tab 2: Content Model Format

Sample 62 Rows from WFS of Heat Flow Content Model Output (See Figure 1).

- 67 Columns are displayed (through CC). The screen capture in Figure 1 is just a portion of the view.
- Column A was not part of the content model output but was added to keep track of the contributing group for this exercise. Note that the example includes rows of data from 4 different datasets as reflected in Column A: Cornell, BEG, and two different SMU datasets.
- 14 columns are highlighted yellow and are empty of data. Each content models contain some optional fields (cardinality $\neq 1$). Columns were inserted into the pull of the GTDA content model output for Heat Flow so that it would contain the same number of fields as the content model description. At the time of this exercise 14 fields were in the content model but not in the GTDA output. Some of these have since been added, including all required columns.
- While hard to read in this report, the comment pop-ups were ‘first cut’ observations of items that need further review. They include things such as noticing that there are multiple entries for the same API, but both rows are listed as ‘site’ rather than ‘interval’ heat flow observations.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	
1	Sample Rows from WFS of Heat Flow Content Model Output																			
2																				
3	column A added to keep track of contributing group. Would be nice to have a better means to know without having to be so familiar. Perhaps might be addressed after CSW? Not sure how that will work....																			
4																				
5	Yellow (empty) columns added to include all possible fields from content model (some are optional to include).																			
6																				
7	GTDA WFS c	FID	ObservationU	ObservationNa	Label	HeatFlow	Type	HeaderURI	APINo	WellNa	OtherID	Owner	LeaseNa	Field	County	State	LatDeg	LongDegree	WG	PI
8	SMU BHT	HeatFlow.fid	https://geothe	Heat flow observat	Heat flow	Site		https://geothe	AR-BHT-000105						Columbia	Arkansas	33.231	-93.11		
9	SMU BHT	HeatFlow.fid	https://geothe	Heat flow observat	Heat flow	Site		https://geothe	FL-BHT-000121						Holmes C Florida		30.983	-85.86		
10	SMU BHT	HeatFlow.fid	https://geothe	Heat flow observat	Heat flow	Site		https://geothe	IL-BHT-000010						McDonon	Illinois	40.34	-90.89		
11	SMU BHT	HeatFlow.fid	https://geothe	Heat flow observat	Heat flow	Site		https://geothe	LA-BHT-000650						Tensas F Louisiana		31.862	-91.4		
12	SMU BHT	HeatFlow.fid	https://geothe	Heat flow observat	Heat flow	Site		https://geothe	LA-BHT-000650						Tensas F Louisiana		31.862	-91.4		
13	SMU BHT	HeatFlow.fid	https://geothe	Heat flow observat	Heat flow	Site		https://geothe	MT-BHT-000622						Rosebud Montana		45.3	-106.7		
14	SMU Heat flow	HeatFlow.fid	https://geothe	Heat flow observat	Heat flow	Site		https://geothe	NV-04906						Pershing Nevada		40.013	-117.7		
15	Cornell	HeatFlow.fid	https://geothe	Heat flow observat	Heat flow	Site		https://geothe	NY-BHT-000964						Chautau New York		42.009	-73.62		
16	SMU BHT	HeatFlow.fid	https://geothe	Heat flow observat	Heat flow	Site		https://geothe	NY-BHT-000964						Schuyler New York		42.471	-77		
17	Cornell	HeatFlow.fid	https://geothe	Heat flow observat	Heat flow	Site		https://geothe	NY-BHT-000964						Cattaraug New York		42.496	-78.57		
18	Cornell	HeatFlow.fid	https://geothe	Heat flow observat	Heat flow	Site		https://geothe	NY-BHT-000964						Chautau New York		42.164	-73.54		
19	Cornell	HeatFlow.fid	https://geothe	Heat flow observat	Heat flow	Site		https://geothe	Calabro T2	Fortuna Energy Inc.		New Field	Schuyler New York		42.289	-77.07				
20	Cornell	HeatFlow.fid	https://geothe	Heat flow observat	Heat flow	Site		https://geothe	Calabro T2	Fortuna Energy Inc.		New Field	Schuyler New York		42.289	-77.07				
21	Cornell	HeatFlow.fid	https://geothe	Heat flow observat	Heat flow	Site		https://geothe	Calabro T2	Fortuna Energy Inc.		New Field	Schuyler New York		42.289	-77.07				
22	SMU BHT	HeatFlow.fid	https://geothe	Heat flow observat	Heat flow	Site		https://geothe	OK-BHT-000940						Allegany New York		42.084	-77.8		
23	Cornell	HeatFlow.fid	https://geothe	Heat flow observat	Heat flow	Site		https://geothe	OK-BHT-000940						Okfuskee Oklahoma		35.427	-96.13		
24	SMU BHT	HeatFlow.fid	https://geothe	Heat flow observat	Heat flow	Site		https://geothe	PA-BHT-000260						Crawford Pennsylv		41.814	-80.19		
25	SMU BHT	HeatFlow.fid	https://geothe	Heat flow observat	Heat flow	Site		https://geothe	PA-BHT-000260						Indiana C Pennsylv		40.556	-78.92		
26	Cornell	HeatFlow.fid	https://geothe	Heat flow observat	Heat flow	Site		https://geothe	PA-BHT-000260						Venango Pennsylv		41.534	-79.84		
27	Cornell	HeatFlow.fid	https://geothe	Heat flow observat	Heat flow	Site		https://geothe	PA-BHT-000260						McKean Pennsylv		41.843	-78.67		
28	Cornell	HeatFlow.fid	https://geothe	Heat flow observat	Heat flow	Site		https://geothe	PA-BHT-000260						Mercer C Pennsylv		41.441	-80.2		
29	BEG Master	HeatFlow.fid	https://geothe	Heat flow observat	Heat flow	Site		https://geothe	PA-BHT-000260						Armstrong Pennsylv		40.838	-79.33		
30	SMU BHT	HeatFlow.fid	https://geothe	Heat flow observat	Heat flow	Site		https://geothe	TX-BHT-004387						Zapata C Texas		28.977	-99.14		
31	BEG Master	HeatFlow.fid	https://geothe	Heat flow observat	Heat flow	Site		https://geothe	TX-BHT-004387						Refugio C Texas		28.4	-97.23		
32	BEG Master	HeatFlow.fid	https://geothe	Heat flow observat	Heat flow	Site		https://geothe	TX-BHT-004387						Wheeler C Texas		35.515	-100.1		
33	BEG Master	HeatFlow.fid	https://geothe	Heat flow observat	Heat flow	Site		https://geothe	TX-BHT-004387						Roberts C Texas		35.811	-100.6		
34	BEG Master	HeatFlow.fid	https://geothe	Heat flow observat	Heat flow	Site		https://geothe	TX-BHT-004387						Johnson Texas		32.408	-97.44		
35	BEG Master	HeatFlow.fid	https://geothe	Heat flow observat	Heat flow	Site		https://geothe	TX-BHT-004387						Wagner Texas		27.244	-99.13		
36	BEG Master	HeatFlow.fid	https://geothe	Heat flow observat	Heat flow	Site		https://geothe	TX-BHT-004387						Titus Cou Texas		33.363	-95.12		
37	BEG Master	HeatFlow.fid	https://geothe	Heat flow observat	Heat flow	Site		https://geothe	TX-BHT-004387						Webb Co Texas		27.409	-99.38		
38	BEG Master	HeatFlow.fid	https://geothe	Heat flow observat	Heat flow	Site		https://geothe	TX-BHT-004387						Henderson Texas		32.146	-96.08		
39	BEG Master	HeatFlow.fid	https://geothe	Heat flow observat	Heat flow	Site		https://geothe	TX-BHT-004387						Wharton Texas		29.351	-96.23		
40	BEG Master	HeatFlow.fid	https://geothe	Heat flow observat	Heat flow	Site		https://geothe	TX-BHT-004387						Wharton Texas		29.351	-96.23		
41	BEG Master	HeatFlow.fid	https://geothe	Heat flow observat	Heat flow	Site		https://geothe	TX-BHT-004387						La Salle C Texas		28.084	-98.98		
42	BEG Master	HeatFlow.fid	https://geothe	Heat flow observat	Heat flow	Site		https://geothe	TX-BHT-004387						Zapata C Texas		27.025	-99.2		
43	BEG Master	HeatFlow.fid	https://geothe	Heat flow observat	Heat flow	Site		https://geothe	TX-BHT-004387						Freestone Texas		31.512	-96.29		
44	SMU BHT	HeatFlow.fid	https://geothe	Heat flow observat	Heat flow	Site		https://geothe	TX-BHT-007130						Goliad C Texas		28.892	-97.45		
45	SMU BHT	HeatFlow.fid	https://geothe	Heat flow observat	Heat flow	Site		https://geothe	TX-BHT-007130						San Jacinto Texas		30.64	-95.2		
46	SMU BHT	HeatFlow.fid	https://geothe	Heat flow observat	Heat flow	Site		https://geothe	TX-BHT-007130						San Jacinto Texas		30.64	-95.2		
47	BEG Master	HeatFlow.fid	https://geothe	Heat flow observat	Heat flow	Site		https://geothe	TX-BHT-007130						San Jacinto Texas		30.64	-95.2		
48	BEG Master	HeatFlow.fid	https://geothe	Heat flow observat	Heat flow	Site		https://geothe	TX-BHT-007130						Wise Cou Texas		33.171	-97.46		
49	BEG Master	HeatFlow.fid	https://geothe	Heat flow observat	Heat flow	Site		https://geothe	TX-BHT-007130						Zapata C Texas		27.012	-99.06		
50	SMU BHT	HeatFlow.fid	https://geothe	Heat flow observat	Heat flow	Site		https://geothe	TX-BHT-007257						Titus Cou Texas		33.042	-95.1		

Process

Content Model Format

SMU BHT Submission

SMU Side by Side Example

Cornell Summary Example

+

Figure 1: Tab 2, GTDA Output in Heat Flow Content Model Format. Yellow cells are column headings that are in the Content Model structure, but not in GTDA's output as of October 31, 2013.

Tab 3: SMU BHT Submission

The worksheet has data arranged in three 'blocks' (Figure 2):

1. First Block is from the submission file. 18 of the 67 sample rows content model rows in worksheet one were from SMU's BHT dataset. Using the site identifier, the same 18 rows were pulled from SMU's BHT (google project) file submission. There is large overlap in the type of information in the SMU BHT submission and in the Cornell BHT submission. The blue highlights are indicating items that SMU BHT submission contains that are not part of the Cornell BHT submission. This will become relevant later when looking at tab 5: Cornell Summary Example, but is not part of this stage. See Figure 3.
2. Second Block is from the WFS delivery of the submitted files. The column order is the same as the top block of rows, except for insertion of first two columns, added by the system (FID and id columns). The first 3 rows in the second block are also from the group in block 1 (and thus worksheet 2). The remaining 12 rows were selected because they contained data in columns that were not populated in the initial batch of 18 rows. For instance, the Virginia wells contained file number, operation name, drilling dates and other information that was unavailable for the other sites. Some Texas wells with company name, field name, elevation, and other information were selected as well. It is important that enough rows are selected such that all

3. Third Block contains the WFS Heat Flow output for the same 15 sites in block 2, plus a few more from block 1. Columns headings for fields that are in the template but not in the GTDA delivery of the heat flow template were inserted and are highlighted yellow. See Figures 6-8.

Figure 2: Image zoomed out to see overall organization of worksheet on **Tab 3**.

Figure 3: Capture from **Block 1, Tab 3.**

Appendix I: 10

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
32	Before Integration																							
33	FID	id	database api	permit	longitude		latitude	surface_interval_id	depth	uncor_bt	harrison	bhtcorrec	surface	harrison	reference	state	k_source	k_sector	resistanc	k	q_calcul	quality	basemer	correctio
34	SMUBHT	004e3e3	BHT Heat Flow			-93.1128	33.231	AR-BHT-000105-a	2408.5	96.1	13.9		110	0	37.8	America	AR	Blackwell Gulf Coast K Model	2.15	81	BHT-C		Harrison	
35	SMUBHT	003bcb6	BHT Heat Flow			-85.8601	30.983	FL-BHT-000121-a	1195.7	43.3	2	45.3	20.3	20.9	America	FL	Blackwell Gulf Coast K Model	2.06	43	BHT-C		Harrison		
51	SMUBHT	00406ad	BHT Heat Flow			-80.1042	33.105	WV-BHT-001430-a	1325.6	39.4	3.6		43	13.4	22.3	Hendry	F WV	COSUNA section -	426.3	3.11	69	BHT-C		6173.1 Harrison
52																								
53	SMUBHT	736cd4b	BHT Heat Flow	160205		-87.9522	37.849	IN-BHT-000008-a	2569.2	61.1	14.9		76	14.3	24	Indiana C	IN	COSUNA section -	1138.9	2.26	54	BHT-C		3859.9 Harrison
54	SMUBHT	10d9156	BHT Heat Flow			-81.7643	37.265	VA-BHT-000259-a	1566.8	46.7	6.6	53.3	15.5	23.8	Virginia D	VA	COSUNA section -	504.8	3.14	75	BHT-C		3735.2 Harrison	
55	SMUBHT	e631754	BHT Heat Flow			-81.7272	37.198	VA-BHT-000263-a	1708.1	36.4	7.9	44.3	15.5	16.9	Virginia D	VA	COSUNA section -	545.2	3.13	53	BHT-C		3756.7 Harrison	
56	SMUBHT	9a24ee3	BHT Heat Flow			-82.8458	36.927	VA-BHT-000025-a	2001	41.1	10.7	51.8	15.5	18.1	Virginia D	VA	COSUNA section -	643.3	3.11	56	BHT-C		3889.2 Harrison	
57	SMUBHT	7a555ed	BHT Heat Flow			-83.354	36.618	VA-BHT-000003-a	2437.8	50	14.1	64.1	15.5	19.9	Virginia D	VA	COSUNA section -	968.1	2.52	50	BHT-C		3642.8 Harrison	
58	SMUBHT	7a555ed	BHT Heat Flow			-83.354	36.618	VA-BHT-000003-b	912	22.8	-1.8	21	15.5	6	Virginia D	VA	COSUNA section -	239.3	3.05	18	BHT-X		3642.8 Harrison	
59																								
60	SMUBHT	601242c	BHT Hea	4E+09		-96.3388	28.673	TX-BHT-005821-a	2329.9	71.1	13.3	84.4	24.3	25.8	Blackwel	TX	Blackwell Gulf Coast K Model	1.92	50	BHT-C		Harrison		
61	SMUBHT	601242c	BHT Hea	4E+09		-96.3388	28.673	TX-BHT-005821-b	2823.2	96.1	16.4	112.5	24.3	31.2	Blackwel	TX	Blackwell Gulf Coast K Model	1.92	60	BHT-C		Harrison		
62	SMUBHT	601242c	BHT Hea	4E+09		-96.3388	28.673	TX-BHT-005821-c	3165.4	112.8	17.9	130.7	24.3	33.4	Blackwel	TX	Blackwell Gulf Coast K Model	1.92	64	BHT-C		Harrison		
63	SMUBHT	d277064	BHT Hea	4236130399D1		-93.8012	29.98	TX-BHT-008761-a	3622	105.6	18.9	124.5	22.3	28.2	Blackwel	TX	Blackwell Gulf Coast K Model	1.9	54	BHT-C		Harrison		
64	SMUBHT	d277064	BHT Hea	4236130399D1		-93.8012	29.98	TX-BHT-008761-b	3551.8	100	18.8	118.8	22.3	27.2	Blackwel	TX	Blackwell Gulf Coast K Model	1.9	52	BHT-C		Harrison		
65	SMUBHT	d277064	BHT Hea	4236130399D1		-93.8012	29.98	TX-BHT-008761-c	3157.3	81.7	17.8	99.5	22.3	24.5	Blackwel	TX	Blackwell Gulf Coast K Model	1.9	47	BHT-C		Harrison		
66																								
67																								
68	After Integration																							

Figure 4: Capture from **Block 2, Tab 3**.

	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ	AK	AL	AM	AN	AO	AP	AQ	AR	AS	AT
32																								
33	basemer correctio bht_well well Ave file_num operatio drilling_drilling_tsc state_plc state_plc company well_type well_sta field_nai formatio elevatior elevatior elevatior county_n surface_bottom_shape																							
34		Harrison	1	81																			POINT (-93.1128 33	
35		Harrison	1	43																			POINT (-85.8601 30	
51	6173.1	Harrison	1	69																			POINT (-80.1042 39	
52																								
53	3859.9	Harrison	1	54																			POINT (-87.9522 37	
54	3735.2	Harrison	1	75	BU-0085	9139-T	#####	1/3/1962		3210521	1108431	United Fr Gas	Plugged/Abandoned										POINT (-81.7643 37	
55	3756.7	Harrison	1	53	TA-0008	9342	#####	#####		3213550	1100795	XTO Ener Gas	Producing										POINT (-81.7272 37	
56	3889.2	Harrison	1	56	WS-0097	VP133597	#####	1/5/1984		3112954	1074833	EQT Prod Gas	Producing										POINT (-82.8458 36	
57	3642.8	Harrison	2	50	LE-0018	L. S. BALE	1/7/1964	#####	28	3065960	1042700	Shell Oil Oil	Plugged/Abandoned										POINT (-83.354 36.6	
58	3642.8	Harrison	2	50	LE-0018	L. S. BALE	1/7/1964	#####	8	3065960	1042700	Shell Oil Oil	Plugged/Abandoned										POINT (-83.354 36.6	
59																								
60		Harrison	3	58					7.5			Brigham Oil & Gas, LP	Eagle Ne Frio	KB	25	7.62					Jackson	POINT (-96.3388 28		
61		Harrison	3	58					7.5			Brigham Oil & Gas, LP	Eagle Ne Frio	KB	25	7.62					Jackson	POINT (-96.3388 28		
62		Harrison	3	58					7.5			Brigham Oil & Gas, LP	Eagle Ne Frio	KB	25	7.62					Jackson	POINT (-96.3388 28		
63		Harrison	3	51					6			Houston Oil & Minerals Cor	North Sal Hackberr	KB	26	7.93					Orange	POINT (-93.8012 29		
64		Harrison	3	51					6			Houston Oil & Minerals Cor	North Sal Hackberr	KB	26	7.93					Orange	POINT (-93.8012 29		
65		Harrison	3	51					6			Houston Oil & Minerals Cor	North Sal Hackberr	KB	26	7.93					Orange	POINT (-93.8012 29		
66																								
67																								

Figure 5: Scrolling to the right on **Block 2, Tab 3**, illustrating that some rows having different columns populated.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
66																					
67																					
68	After Integration																				
69	Nov2012	template	Observat	Observat	Label	HeatFlowType	Related	HeaderURI	APINo	WellNam	OtherID	Owner	LeaseNa	Field	County	State	LatDegree	LongDeg	Location	PLSS_Mer	Township
70	GTDA	WF FID	Observat	Observat	Label	HeatFlowType		HeaderURI	APINo	WellNam	OtherID	Owner	LeaseNa	Field	County	State	LatDegree	LongDeg	FreeWGS84	PLSS_Mer	Township
71	SMU	BHT	HeatFlow	https://g	Heat flow	Heat flow Site		https://geothermal.smu.edu/	IA-BHT-000105						Columbia	Arkansas	33.2308	-93.1128			
72	SMU	BHT	HeatFlow	https://g	Heat flow	Heat flow Site		https://geothermal.smu.edu/	FL-BHT-000121						Holmes	Florida	30.9831	-85.8601			
85	SMU	BHT	HeatFlow	https://g	Heat flow	Heat flow Site		https://geothermal.smu.edu/	TX-BHT-000278						Parmer	C Texas	34.7359	-102.592			
86	SMU	BHT	HeatFlow	https://g	Heat flow	Heat flow Interval		https://geothermal.smu.edu/	TX-BHT-000278						Parmer	C Texas	34.7359	-102.592			
87	SMU	BHT	HeatFlow	https://g	Heat flow	Heat flow Site		https://geothermal.s	#####	TX-BHT-008353		EOG Resources	Carthage	Polanco	C Texas	32.0108	-94.3523				
88	SMU	BHT	HeatFlow	https://g	Heat flow	Heat flow Site		https://geothermal.smu.edu/	WV-BHT-001430						Barbour	West Vir	39.1048	-80.1042			
89																					
90	SMU	BHT	HeatFlow	https://g	Heat flow	Heat flow Site		https://geothermal.smu.edu/	IN-BHT-000008				160205	Posey	Co Indiana	37.8489	-87.9522				
91	SMU	BHT	HeatFlow	https://g	Heat flow	Heat flow Site		https://geothermal.smu.edu/	VA-BHT-000259			United Fuel	(Columbia)	Buchana	Virginia	37.2653	-81.7643				
92	SMU	BHT	HeatFlow	https://g	Heat flow	Heat flow Site		https://geothermal.smu.edu/	VA-BHT-000263			XTO Energy, Inc.		Tazewell	Virginia	37.1975	-81.7272				
93	SMU	BHT	HeatFlow	https://g	Heat flow	Heat flow Site		https://geothermal.smu.edu/	VA-BHT-000025			EQT Production Company		Wise	Co Virginia	36.9274	-82.8458				
94	SMU	BHT	HeatFlow	https://g	Heat flow	Heat flow Site		https://geothermal.smu.edu/	VA-BHT-000003			Shell Oil Co		Lee Coun	Virginia	36.6175	-83.354				
95	SMU	BHT	HeatFlow	https://g	Heat flow	Heat flow Interval		https://geothermal.smu.edu/	VA-BHT-000003			Shell Oil Co		Lee Coun	Virginia	36.6175	-83.354				
96																					
97																					
98		HeatFlow	https://g	Heat flow	Heat flow Site			https://geothermal.s	4.2E+09	TX-BHT-005821		Brigham Oil & Gas	Eagle Ne Calhoun	Texas	28.6733	-96.3388					
99		HeatFlow	https://g	Heat flow	Heat flow Interval			https://geothermal.s	4.2E+09	TX-BHT-005821		Brigham Oil & Gas	Eagle Ne Calhoun	Texas	28.6733	-96.3388					
100		HeatFlow	https://g	Heat flow	Heat flow Interval			https://geothermal.s	4.2E+09	TX-BHT-005821		Brigham Oil & Gas	Eagle Ne Calhoun	Texas	28.6733	-96.3388					
101		HeatFlow	https://g	Heat flow	Heat flow Site			https://geothermal.s	42361303	TX-BHT-008761		Houston Oil & Min	North Sal Orange	C Texas	29.9802	-93.8012					
102		HeatFlow	https://g	Heat flow	Heat flow Interval			https://geothermal.s	42361303	TX-BHT-008761		Houston Oil & Min	North Sal Orange	C Texas	29.9802	-93.8012					
103		HeatFlow	https://g	Heat flow	Heat flow Interval			https://geothermal.s	42361303	TX-BHT-008761		Houston Oil & Min	North Sal Orange	C Texas	29.9802	-93.8012					
104																					
105																					

Figure 6: **Block 3, Tab 3**, showing the same data reformatted into content model layout. Yellow highlights are content model fields (some of which are optional) that were not in the GTDA output.

	AC	AD	AE	AF	AG	AH	AI	AJ	AK	AL	AM	AN	AO	AP	AQ	AR	AS	AT	AU	AV	AW	AX	AY	AZ
69	Related	WellType	WellDrill	WellDrill	Elevation	Elevation	DepthUn	DrillerTo	TimeSinc	Tempera	Tempera	SurfaceT	BHT	BHTCorre	BHTCorre	EqLog_M	EqLog_Tc	EqLogTer	EqLogTer	EqLogTer	SiteHeat	SiteValue	Intervals	IntervalT
70		WellType	WellDrill	Date	Elevation	Elevation	DepthUn	DrillerTo	TimeSinc	Tempera	Tempera	SurfaceT	BHT	BHTCorre	BHTCorre	EqLog_M	EqLog_Tc	EqLogTer	EqLogTer	EqLogTer	SiteHeat	SiteValue	Intervals	IntervalT
71		nil:missing					m	-99999		C	Unknownr	0	96.1	110							81	nil:NA	-99999	-99999
72		nil:missing					m	-99999		C	Unknownr	20.3	43.3	45.3							43	nil:NA	-99999	-99999
85		nil:missing					m	-99999		C	Unknownr	14.4	48.3	65.2							40	nil:NA	-99999	-99999
86		nil:missing					m	-99999		C	Unknownr	14.4	47.8	55.5							56	nil:NA	-99999	-99999
87		Gas Well			102.44	KB	m	-99999	4.25	C	Unknownr	21.3	98.9	115.6							66	nil:NA	-99999	-99999
88		nil:missing					m	-99999		C	Unknownr	13.4	39.4	43							69	nil:NA	-99999	-99999
89																								
90		nil:missing					m	-99999		C	Unknownr	14.3	61.1	76							54	nil:NA	-99999	-99999
91		Gas	1/3/1962				m	-99999		C	Unknownr	15.5	46.7	53.3							75	nil:NA	-99999	-99999
92		Gas	#####				m	-99999		C	Unknownr	15.5	36.4	44.3							53	nil:NA	-99999	-99999
93		Gas	1/5/1984				m	-99999		C	Unknownr	15.5	41.1	51.8							56	nil:NA	-99999	-99999
94		Oil	#####				m	-99999	28	C	Unknownr	15.5	50	64.1							50	nil:NA	-99999	-99999
95		Oil	#####				m	-99999	8	C	Unknownr	15.5	22.8	21							18	nil:NA	-99999	-99999
96																								
97																								
98		nil:missing			7.62	KB	m	-99999	7.5	C	Unknownr	24.3	71.1	84.4							50	nil:NA	-99999	-99999
99		nil:missing			7.62	KB	m	-99999	7.5	C	Unknownr	24.3	96.1	112.5							60	nil:NA	-99999	-99999
100		nil:missing			7.62	KB	m	-99999	7.5	C	Unknownr	24.3	112.8	130.7							64	nil:NA	-99999	-99999
101		nil:missing			7.93	KB	m	-99999	6	C	Unknownr	22.3	105.6	124.5							54	nil:NA	-99999	-99999
102		nil:missing			7.93	KB	m	-99999	6	C	Unknownr	22.3	100	118.8							52	nil:NA	-99999	-99999
103		nil:missing			7.93	KB	m	-99999	6	C	Unknownr	22.3	81.7	99.5							47	nil:NA	-99999	-99999
104																								

Figure 7: Scrolling to the right on **Block 3, Tab 3**, illustrating that some of the missing content model columns were for equilibrium data. These columns were added in later, after October 31st, but the core database structure feeding into the content model layout does not distinguish between the two temperature types. This was not yet understood until this exercise.

	BK	BL	BM	BN	BO	BP	BQ	BR	BS	BT	BU	BV	BW	BX	BY	BZ	CA	CB	CC	CD	CE
69	Interval	Corrected	SE_Cor	Interval	Corrected	SE_Unco	Corrected	SE_Cor	Interval	HF_Q	HF_Q	HF_Q	HF_Q	HF_Q	HF_Q	HF_Q	HF_Q	HF_Q	HF_Q	HF_Q	HF_Q
70	Interval	Corrected	SE_Cor	Interval	Corrected	SE_Unco	Corrected	SE_Cor	Interval	HF_Q	HF_Q	HF_Q	HF_Q	HF_Q	HF_Q	HF_Q	HF_Q	HF_Q	HF_Q	HF_Q	HF_Q
71	37.8			mW/m2	81	-99999	BHT-C														
72	20.9			mW/m2	43	-99999	BHT-C														
85	17.3			mW/m2	40	-99999	BHT-C														
86	24.2			mW/m2	56	-99999	BHT-C														
87	32.7			mW/m2	66	-99999	BHT-C														
88	22.3			mW/m2	69	-99999	BHT-C														
89																					
90	24			mW/m2	54	-99999	BHT-C														
91	23.8			mW/m2	75	-99999	BHT-C														
92	16.9			mW/m2	53	-99999	BHT-C														
93	18.1			mW/m2	56	-99999	BHT-C														
94	19.9			mW/m2	50	-99999	BHT-C														
95	6			mW/m2	18	-99999	BHT-X														
96																					
97																					
98	25.8			mW/m2	50	-99999	BHT-C														
99	31.2			mW/m2	60	-99999	BHT-C														
100	33.4			mW/m2	64	-99999	BHT-C														
101	28.2			mW/m2	54	-99999	BHT-C														
102	27.2			mW/m2	52	-99999	BHT-C														
103	24.5			mW/m2	47	-99999	BHT-C														
104																					

Figure 8: Scrolling further right on **Block 3, Tab 3**. This example is from SMU BHT submission file. When the same exercise was done with the SMU Heat Flow Database from Equilibrium Wells, we learned that some columns were not being pulled in from the submission file (water table top in column BX, for example). That was because these items were not part of the core database – only the staging database. This led to the decision to make the submission files pulled from the staging database available as tier 2 datasets, so that all columns submitted would be viewable. In the case of water table top, Siemens did add that to the database schema and it does now populate the heat flow content model for equilibrium data.

Tab 4: SMU Side By Side Example

This tab places the columns from the SMU input versus Heat Flow Content Model delivery from GTDA vertically with two sections, one on the left (columns A-G, Figure 9) and one on the right (columns H-P, Figure 10).

1. Columns A-G relate to the SMU BHT Submission File:

Column A lists the column headings used in the submission file as pulled from the staging database. Columns B-F contain data from the submission file for 5 entries. Column G contains

a textual description of how that field should map to the heat flow content model. Cells highlighted red in column G warranted further discussion. In some cases a correction is needed and in other cases it was indicating need for more analysis. See Figure 9.

2. Columns H-P relate to the GTDA Heat Flow Content Model:

Column H contains the content model field descriptions. Column I indicates whether the field is required (cardinality of 1 = required). Column J contains a textual description of how that field should map to the SMU BHT submission file. Column K contains the Heat Flow Content Model field names. Columns L-P contain data from the content model output file for the same 5 sites represented in B-F. See Figure 10.

	A	B	C	D	E	F	G
1	Side by Side Comparison of SMU input versus Heat Flow Content Model output from SMU						
2							HEAT FLOW
3							Description of how SMU BHT column names map to Heat Flow Template columns.
4	FID	SMUBHT.fid--7c271b74_1419462ff30_-817	SMUBHT.fid--7c271b74_1419462ff30_-817	SMUBHT.fid--7c271b74_1419462ff30_-817	SMUBHT.fid--7c271b74_1419462ff30_-817	SMUBHT.fid--7c271b74_1419462ff30_-817	
5	id	00570d52-e57e-3475-00570d52-0059970d-ad98-00406ad8-736cd4b6-7526-35be-ab35-74e10e39a1b1	00570d52-e57e-3475-00570d52-0059970d-ad98-00406ad8-736cd4b6-7526-35be-ab35-74e10e39a1b1	00570d52-e57e-3475-00570d52-0059970d-ad98-00406ad8-736cd4b6-7526-35be-ab35-74e10e39a1b1	00570d52-e57e-3475-00570d52-0059970d-ad98-00406ad8-736cd4b6-7526-35be-ab35-74e10e39a1b1	00570d52-e57e-3475-00570d52-0059970d-ad98-00406ad8-736cd4b6-7526-35be-ab35-74e10e39a1b1	
6	database	BHT Heat Flow	BHT Heat	BHT Heat Flow	BHT Heat	BHT Heat	not part of the the template.
7	api			4236534645			direct match to APIno
8	permit					160205	currently populates LeaseName. Verify ok. Cornell's Permit numbers are also going to LeaseName. Could perhaps go in OtherID. Maybe discuss with MLKay.
9	longitude	-102.5921	-102.592	-94.3523	-80.1042	-87.9522	direct match to LongDegreeWGS84.
10	latitude	34.7359	34.7359	32.0108	39.1048	37.8489	direct match to LatDegreeWGS84.
11	surface_interval_id	TX-BHT-000278-b	TX-BHT-00	TX-BHT-008353-z	WV-BHT-0	IN-BHT-0	mapped to WellName, without the "-letter" at the end
12	depth	1695.7	2932.3	2882.9	1325.6	2569.2	currently not used. Should populate IntervalBottomDepth.
13	uncor_bht	47.8	48.3	98.9	39.4	61.1	direct match to BHT.
14	harrison_correction	7.7	16.9	16.7	3.6	14.9	unused. Only pulls in the corrected value.
15	bhtcorrected_temp	55.5	65.2	115.6	43	76	direct match to BHTCorrectedValue.
16	surface_temp	14.4	14.4	21.3	13.4	14.3	direct match to SurfaceTemperature
17	harrison_gradient	24.2	17.3	32.7	22.3	24	direct match to IntervalCorrectedGradient
Process Content Model Format SMU BHT Submission SMU Side by Side Example Cornell Summary							
READY							

Figure 9: **Tab 4, Column A** lists the column headings used in the submission file as pulled from the staging database. **Columns B-F** contain data from the submission file for 5 entries. **Column G** contains a textual description of how that field

should map to the heat flow content model. Cells highlighted red in column G warranted further discussion. In some cases a correction is needed and in other cases it was indicating need for more analysis.

	H	I	J	K	L	M	N	O	P
1									
2	HEAT FLOW	HEAT FLOW	HEAT FLOW	HEAT FLOW					
3			Description of how Heat Flow Template columns map to SMU BHT columns.						
4				Nov2012 template	SMU BHT	SMU BHT	SMU BHT	SMU BHT	SMU BHT
5	Element Description	Cardinality			HeatFlow.fid	HeatFlow	HeatFlow	HeatFlow	HeatFlow.f
6	Unique identifier for this ob	1	Confirm with Siemens that the S in https is valid. GTDA is correctly listing same headerURI for wells with multiple entries for the same borehole.	ObservationURI	https://geothe	https://g	https://g	https://g	https://g
7	unique identifier for this ob	1	GTDA automatically generates for the well ID (without the interval designation)	ObservationName	Heat flow obs	Heat flow	Heat flow	Heat flow	Heat flow
8	Short text string for labelin	1	GTDA using same as observation name, which is fine.	Label	Heat flow obs	Heat flow	Heat flow	Heat flow	Heat flow
9	Valid terms are SiteHF or	1	(note: GTDA correctly designates anything besides an 'a' to the interval ID as interval and those with 'a' are considered site. However, should use SiteHF or IntervalHF for term.	HeatFlowType	Site	Interval	Site	Site	Site
10	If Heat Flow value is base	0..1	not in SMU's delivery but is in template. Supposed to list the URIs for the interval data.	RelatedHeatFlowIntervalsURI					
11	Unique identification value	1	Confirm with Siemens that the S in https is valid. GTDA is correctly listing same headerURI for wells with multiple entries for the same borehole.	HeaderURI	https://geothe	https://g	https://g	https://g	https://g
12	API number for the the we	0..1	direct match to api	APINo			#####		
13	Common name to identify t	0..1	mapped to surface_interval_id without the "-letter" - Think it is probably ok. Need to find one with another name to double check against.	WellName	TX-BHT-000278	TX-BHT-0	TX-BHT-0	WV-BHT-0	IN-BHT-000
14	Other well identifiers. Use	0..1	Currently not populated. Probably ok. The values in operation_name sound more like they go here than in notes, but not a big problem.	OtherID					
15	The company, land and/or	0..1	currently pulling company_name. Per Maria, that is correct.	Owner			EOG Resources		
16	Name of lease, usually inc	0..1	currently pulling from permit. Ask Maria if this should pull in another field instead - perhaps operation_name?	LeaseName					160205
17	Name for a geothermal res	0..1	direct match to field_name.	Field			Carthage		
	Process	Content Model Format	SMU BHT Submission	SMU Side by Side Example			Cornell Summary Example		

Figure 10: **Tab 4, Column H** contains the content model field descriptions. **Column I** indicates whether the field is required (cardinality of 1 = required). **Column J** contains a textual description of how that field should map to the SMU BHT submission file. If highlighted red, additional discussion was warranted. **Column K** lists the Heat Flow Content Model field names. **Columns L-P** contain data from the content model output file for the same 5 sites represented in B-F in figure 9. Yellow columns were in the content model description, but not in the GTDA implementation as of October 31, 2014.

As indicated by the amount of red on the zoomed out screen capture of most of the worksheet below (Figure 11), there was a significant amount of discussion needed. Keep in mind this is only one of 5 different files that fed into the Heat Flow content model. And, the SMU BHT submission file feeds into many different content models (borehole temperature, well header, heat flow, thermal conductivity,

etc.). Making a change to just one field in the database structure has a wide ripple effect. In some cases, a change to correct where submitted data populates one content model field causes another content model field to become incorrect.

Figure 11: While too small to actually read, an image of the spreadsheet on tab 4 was captured zoomed out to illustrate the number of fields needing discussion (red).

Tab 5: Cornell Summary Example

Using the same process as just outlined, the Cornell submission file data was compared to the GTDA output for *both* the Heat Flow Content Model and for the Borehole Temperature Content Model. A summary of the analysis (minus the rows of sample data) appears on tab 5, organized vertically. 4 different tables are on tab 5 (Figures 12-15).

1. Table 1 is on the left side of the worksheet (columns A-D). Cornell's column names are listed vertically in column A. Column B lists whether those columns were used to populate the Heat Flow Content Model and/or the Borehole Temperature Content Model. 'Match' means it was a direct match to a content model and that it was matched correctly. 'Used' indicates it was not populating either content model. 'Unused' and highlighted red indicates that it is a problem that it is not currently populating either content model. 'Mismatched' means that the field was used to populate the content model but that there is some issue with how it is mapped. Columns C and D describe specifically how the Cornell field is currently matched/mismatched. See Figure 12a, 12b and 12c.
2. Table 2 (columns F and G) are similar, but in reverse, from the perspective of the Heat Flow Content Model to Cornell's data fields. This is necessary, as described in the process, because it identifies required items that may be missing from the content model format. For example, as indicated on Figure 13, Cornell needed to specify if their data was a 'site' value versus an 'interval' value. It also prompted a discussion to verify that Cornell's coordinate values were in WGS84 format, as stipulated by the content model. Highlighted cells indicate discussion and/or action needed. See Figure 13a, 13b, 13c, and 13d.
3. Table 3 (columns I and J) is organized like Table 2 on Figure 13, but is from the Borehole Temperature Content Model to Cornell's data fields. See Figure 14a, 14b, 14c, and 14d.
4. Table 4 (columns L-P) are simply the Borehole Temperature Content Model fields names with descriptions. This same information was provided for the Heat Flow Content Model on the previous Tab 4 (Columns H on Figure 10). It is helpful to have the element descriptions and the cardinality within easy viewing, to help clarify the precise meaning of field names. See Figure 15.

	A	B	C	D
1	Summary of Cornell Submission versus Heat Flow Content Model Output AND versus Borehole Temperature Output			
2				
3				
4	TABLE 1- Correlates the Cornell column names to the template fields.			
5	Cornell's Labels/Order	Matched or Unused by HF and/or Borehole Temp. Templates	Template- Lists the template field name that correlates to the Cornell name appearing in column A.	Comments re: Borehole Temperature Template- Lists the template field name that correlates to the Cornell name appearing in column A.
6	FID	unused?	generated by GTDA.	generated by GTDA.
7	api_wellno	used for API & Name	being used for API. Also used for WellName if none other given.	direct match to APINo. Also used for WellName if Cornell's well_nm is not populated.
8	bht_f	unused	only metric pulled in.	only metric pulled in.
9	bht_c	match	direct match to BHT	direct match to MeasuredTemperature.
10	harrison_correction	unused	only corrected value being pulled in- not amount of correction.	only corrected value being pulled in- not amount of correction.
11	harrison_bht_c	match	direct match to BHTCorrectedValue	direct match to CorrectedTemperature.
12	log_depth_f	unused	only metric pulled in.	only metric pulled in.
13	log_depth_m	Mismatched	went in as DrillerTotalDepth. Should go in for IntervalBottomDepth. Needs correcting.	direct match to DepthOfMeasurement.
14	tvd_f	unused	only metric pulled in.	only metric pulled in.
15	tvd_m	unused HF/Used Borehole	only using calc_depth_m instead for DrillerTotalDepth. Borehole template has tvd spot.	direct match to TrueVerticalDepth.
16	calc_depth_m	unused	should use for DrillerTotalDepth. Needs discussion with Cornell.	should use for DrillerTotalDepth. Needs correcting.
17	harrison_corr_grad_c km	matched	direct match to IntervalCorrectedGradient	not in borehole template

Figure 12a, Tab 5, Table 1 (columns A-D), Top Portion (Rows 6-16). Comparing Cornell's submission file column headings to the column headings to both the Heat Flow Content Model and to the Borehole Temperature Content Model. Red indicates need for discussion.

	A	B	C	D
4	TABLE 1- Correlates the Cornell column names to the template fields.			
5	Cornell's Labels/Order	Matched or Unused by HF and/or Borehole Temp. Templates	Template- Lists the template field name that correlates to the Cornell name appearing in column A.	Comments re: Borehole Temperature Template- Lists the template field name that correlates to the Cornell name appearing in column A.
17	harrison_corr_grad_c_km	matched	direct match to IntervalCorrectedGradient.	not in borehole template.
18	uncorr_grad	matched	direct match to IntervalUncorrectedGradient	not in borehole template.
19	state	unused	generated by GTDA from coordinates.	generated by GTDA from coordinates.
20	county	unused	generated by GTDA from coordinates.	generated by GTDA from coordinates.
21	cosuna_coi	unused	Need to add to either FormationName or LithologyInformation. Needs correcting.	not in borehole template.
22	town	unused	Need to add to OtherLocationName. Needs Correcting	Need to add to OtherLocationName. Needs Correcting
23	permit_number	matched/possible mismatch for borehole	Currently using for LeaseName. Verify ok. Other option would be OtherID. Ask Cornell/MLKag if as intended.	Currently using for LeaseName. Should probably go to LeaseNo instead. Ask Cornell/MLKag if as intended.
24	operator	mismatched	currently using for Owner. Not sure where to place. Template doesn't offer operator field.	Currently using for Owner in Borehole WFS but AASG version of Borehole template does include field called operator.
25	farm_name	matched	part of OtherLocationName	part of OtherLocationName
26	td_date	unused for HF/ unused for borehole	needs description if using for 'wellDrillDate. Not sure how to handle if GTDA combines the dates. Heat Flow template does not accommodate multiple dates so recommending use the log_date for 'wellDrillDate with description that the 'wellDrillDate Type is 'log measurement	not in borehole template. Has 'status date' instead. Use to generate? Don't know that GTDA has a means to identify what kind of date something is other than the bht measurement date...
27	log_date		needs description if using for 'wellDrillDate. Recommending use this date with 'wellDrillDateType of "log Measurement Date"	Currently not being used, but should be direct match to MeasurementDateTime. Needs Correcting.
28	comp_date	unused	Heat flow template does not accommodate multiple dates so recommending only use log-date to populate 'wellDrillDate and supply 'measurement date' as the	not in borehole template. Has 'status date' instead. Use to generate?
29	well_type	matched / missing for borehole	direct match to 'wellType but Cornell should resubmit using term list names instead of abbreviations.	Currently not being used, but has direct match in AASG version of template called 'wellType. Needs Correcting.
30	well_status	unused HF/Used Borehole	Not in HF template - check other templates	Direct match to Status. Status currently also has 'plugged' prepopulated in addition to what is in well_status. Needs correcting.
31	well_nm	matched	direct match to 'wellName	direct match to 'wellName
32	elevation	unused HF / unused in borehole	Currently not reading in elevation. direct match to Elevation but unsure if units are meters. Cornell needs to verify units are meters. If Cornell can tell us if KB, GL, or DF, can populate accordingly.	Currently not being used, since don't know where measured from (KB, etc.)
◀ ▶ ... SMU BHT Submission SMU Side by Side Example Cornell Summary Example ⊕				

Figure 12b, Tab 5, Table 1 (columns A-D), Middle Portion (Rows 17-32). Comparing Cornell's submission file column headings to the column headings to both the Heat Flow Content Model and to the Borehole Temperature Content Model. Red indicates need for discussion.

	A	B	C	D
4	TABLE 1- Correlates the Cornell column names to the template fields.			
5	Cornell's Labels/Order	Matched or Unused by HF and/or Borehole Temp. Templates	Template- Lists the template field name that correlates to the Cornell name appearing in column A.	Comments re: Borehole Temperature Template- Lists the template field name that correlates to the Cornell name appearing in column A.
33	field	matched	part of OtherLocationName	part of OtherLocationName
34	pool	matched	part of OtherLocationName	part of OtherLocationName
35	municipality	unused	Need to add to OtherLocationName. Needs Correcting	Need to add to OtherLocationName. Needs Correcting
36	quad	matched	part of OtherLocationName	part of OtherLocationName
37	map_section	matched	Direct match to Section	Direct match to Section.
38	tdform	potential error	Direct match to FormationName, but has error in formula appended to it.	Currently direct match to FormationTD in combination with tvd_m, but giving error where no
39	longitude	unused	only pulled in decimal coordinates.	only pulled in decimal coordinates.
40	longitude_offset	unused	but could use in LocationUncertaintyStatement (if populated)	but could use in LocationUncertaintyStatement (if populated)
41	latitude	unused	only pulled in decimal coordinates.	only pulled in decimal coordinates.
42	latitude_offset	unused	but could use in LocationUncertaintyStatement (if populated)	but could use in LocationUncertaintyStatement (if populated)
43	decimal_longitude	match	direct match to LongDegree'WGS84. confirm 'WGS84	Direct match to LongDegree. Confirm 'WGS84 with cornell
44	decimal_latitude	match	direct match to LatDegree'WGS84. confirm 'WGS84	Direct match to LatDegree. Confirm 'WGS84 with Cornell
45	sed_thickness_m	unused	Need to add to LithologyInformation. Needs correcting.	not in borehole template.
46	heat_flow_mw_m2	match & mismatch	currently populating SiteHeatFlow and IntervalUncorrectedHeatFlow. Should be IntervalCorrectedHeatFlow. Usually site, too, but not always.	
47	average_conductivity	unused	Need to double check against most recent template, but think it could be used to populate IntervalThermalConductivity?	not in borehole template.
48	data_source	unused	Need to add to PublicationSource. Also needs expanded info from Cornell.	Currently not being used, but should be direct match to Information_Source, along with prepopulating Cornell. Needs correcting
49	calc_basement_heat_prod	unused	Verify if okay to use in HeatGeneration.	not in borehole template.

Figure 12c, Tab 5, Table 1 (columns A-D), Bottom Portion (Rows 33-49). Comparing Cornell's submission file column headings to the column headings to both the Heat Flow Content Model and to the Borehole Temperature Content Model. Red indicates need for discussion. Table 1 continues on for 9 more rows with estimated temperatures at depth fields (not part of either content model)..

	F	G
4	TABLE 2 - Correlates the Heat flow column names to the Cornell column names.	
5	<u>Heat Flow WFS Labels/Order</u>	Comments on how generated and/or which of Cornell's field names match to the Template names. Purple need discussion and/or review. (didn't distinguish who the owner of the action item is at this point)
6	Yellow fields are in 1.23 but not GTDA	Potential additional field to Cornell Spreadsheet
7	Nov2012 template includes yellow	Some Change Suggested - or at least review Cornell->HE
8		
9	ObservationURI	Generated by GTDA-not sure of convention
10	ObservationName	"Heat flow observation for" Wellname (well_nm or API)
11	Label	label identical to ObservationName
12	HeatFlowType	Cornell should provide site vs interval
13	RelatedHeatFlowIntervalsURI	caught by system?
14	HeaderURI	"https://geothermal.smu.edu/header/" FID without contributor name. Consider keeping contributor name?
15	APINo	direct match with api_wellno.
16	WellName	using well_nm unless empty, then using API number
17	OtherID	Currently not populated. Could use permitID here if not okay to use for LeaseName.
18	Owner	Using Operator now. Not equivalent. Discuss if okay to leave blank and not use operator. Where would operator go instead?
19	LeaseName	Verify it is ok to use permit number for lease name. If not, could use for OtherID.
20	Field	Direct match with field. Repeated where available for same API but was blank. (good!)
21	County	using coordiantes
22	State	using coordiantes
23	LatDegreeWGS84	direct match - verify Cornell used WGS 84
24	LongDegreeWGS84	direct match - verify Cornell used WGS 84
25	LocationUncertaintyStatement	Not in GTDA's version, but should be using with either offset info or prepopulated stmt.
◀ ▶ ... SMU BHT Submission SMU Side by Side Example Cornell Summary Ex		

Figure 13a, Tab 5, Table 2 (columns F-G), Top Portion (Rows 9-25). Comparing column headings from the Heat Flow Content Model to Cornell's input file. Shading indicates need for discussion or action.

	F	G
4	TABLE 2 - Correlates the Heat flow column names to the Cornell column names.	
5	<u>Heat Flow WFS Labels/Order</u>	Comments on how generated and/or which of Cornell's field names match to the Template names. Purple need discussion and/or review. (didn't distinguish who the owner of the action item is at this point)
6	Yellow fields are in 1.23 but not GTDA	Potential additional field to Cornell Spreadsheet
7	Nov2012 template includes yellow	<u>Some Change Suggested – or at least review Cornell->HE</u>
26	PLSS_Meridians	not provided/not needed
27	Township	municipality and town provided, but not certain enough to use here. Ok to leave unpopulated for Cornell dataset.
28	Range	not provided/not needed
29	Section_	Currently not populating, but was provided by Cornell's spreadsheet. 931 entries in June2012 ver. Need to use.
30	SectionPart	not provided/not needed (unless map_section is really SectionPart)
31	TectonicProvince	not provided/not needed
32	OtherLocationName	Currently pulling farm, pool and quad. Pull town and municipality, too.
33	PublicationSource	Currently not populated, but should pull in data_source. Cornell should provide more than there now in follow up submission. Include info on well record info, surface temp, lithology sources, etc. Maybe use Cornell name in description, too, if unable to accommodate having provider name elsewhere.
34	AdditionalSources	Cornell should have provided. Could use Publication Source and Notes if can't add now.
35	RelatedResource	prepopulate with URI for their estimated temp at depth map and their estimated values
36	WellType	Populating, but should use term list. Could have Cornell resubmit using standardized terms instead of abbreviations.
37	WellDrillDate	not sure which it's using?? Seems off on entries with multiple dates and multiple entries per api.
38	WellDrillDateType	could provide from column heading? How to handle multiple date entries, though?
39	Elevation	currently not populated, but is a direct match, but need to verify with Cornell really is in meters.
40	ElevationType	not provided. Should we prepopulate with 'unknown'?
41	DepthUnits	Already prepopulated.
		Potential mismatch. Currently pulling in log depth, not
◀ ▶ ... SMU BHT Submission SMU Side by Side Example Cornell Summary Exa		

Figure 13b, Tab 5, Table 2 (columns F-G), Middle Portion (Rows 26-41). Comparing column headings from the Heat Flow Content Model to Cornell's input file. Shading indicates need for discussion or action.

	F	G
4	TABLE 2 - Correlates the Heat flow column names to the Cornell column names.	
5	<u>Heat Flow WFS Labels/Order</u>	Comments on how generated and/or which of Cornell's field names match to the Template names. Purple need discussion and/or review. (didn't distinguish who the owner of the action item is at this point)
6	Yellow fields are in 1.23 but not GTDA	Potential additional field to Cornell Spreadsheet
7	Nov2012 template includes yellow	Some Change Suggested - or at least review Cornell->HF
42	DrillerTotalDepth	Potential mismatch. Currently pulls in log_depth_m and should pull in calc_depth_m. In reviewing ones with multiple records per API, seems to pull largest and repeat for all records. TVD is actual deeper on some. It may be pulling the deepest between log_depth and calc_depth? Need to discuss.
43	TimeSinceCirculation	not provided by Cornell.
44	TemperatureUnits	Already prepopulated.
45	TemperatureMeasureType	Ok to Prepopulate - but need to change to BHT
46	SurfaceTemperature	not provided by Cornell. (but they probably should have). If too late to add, could include with source/notes info.
47	BHT	direct match
48	BHTCorrectedValue	direct match
49	BHTCorrectionType	consider prepopulating with 'Harrison'
50	EqLog_MeasurementDate	not provided/not needed for Cornell's dataset
51	EqLog_TotalDepth	not provided/not needed for Cornell's dataset
52	EqLogTemperatureTop	not provided/not needed for Cornell's dataset
53	EqLogTemperatureMax	not provided/not needed for Cornell's dataset
54	EqLogTemperatureBottom	not provided/not needed for Cornell's dataset
55	SiteHeatFlow	Currently pulling 'heat_flow_mw_m2' (cornell) into SiteHeatFlow (template), even if there are multiple entries for the same site - with different values.
56	SiteValueMethod	consider prepopulating with 'single BHT unless otherwise stated in notes'?
57	Intervals	discuss - most will be 1, but not all.
◀ ▶ ... SMU BHT Submission SMU Side by Side Example Cornell Summary Exam READY		

Figure 13c, Tab 5, Table 2 (columns F-G), Middle Portion, Continued (Rows 42-57). Comparing column headings from the Heat Flow Content Model to Cornell's input file. Shading indicates need for discussion or action.

	F	G
4	TABLE 2 - Correlates the Heat flow column names to the Cornell column names.	
5	Heat Flow WFS Labels/Order	Comments on how generated and/or which of Cornell's field names match to the Template names. Purple need discussion and/or review. (didn't distinguish who the owner of the action item is at this point)
6	Yellow fields are in 1.23 but not GTDA	Potential additional field to Cornell Spreadsheet
7	Nov2012 template includes yellow	Some Change Suggested - or at least review Cornell->HF
58	IntervalTopDepth	discuss - use "null" or "0" for surface?
59	IntervalBottomDepth	discuss - use log_depth_m here.
60	ThermalConductivityUnits	Already prepopulated.
61	IntervalThermalConductivity	discuss - expect to use average_conductivity.
62	SE_Conductivity	not provided, but should have. Discuss prepopulating with a set value or referencing notes or source section?
63	ConductivitySampleType	consider prepopulating with 'estimated from AAPG_COSUNASection'
64	ConductivityNumSamples	not provided/not needed for Cornell's dataset
65	ConductivityMeasuredURL	not provided/not needed for Cornell's dataset
66	GradientUnits	Already prepopulated.
67	IntervalUncorrectedGradient	direct match.
68	SE_UncorrectedGradient	not provided, but should have. Discuss prepopulating with a set value or referencing notes or source section?
69	IntervalCorrectedGradient	direct match.
70	SE_CorrectedGradient	not provided, but should have. Discuss prepopulating with a set value or referencing notes or source section?
71	IntervalCorrectedGradientMethod	not provided, but should have. Discuss prepopulating with a set value or referencing notes or source section?
72	HeatFlowUnits	Already prepopulated.
73	IntervalUncorrectedHeatFlow	Currently pulling 'heat_flow_mw_m2' field into IntervalUncorrectedHeatFlow but should probably leave blank and instead put 'heat_flow_mw_m2' into IntervalCorrectedHeatFlow
74	SE_UncorrectedHeatFlow	not provided, but should have. Discuss prepopulating with a set value or referencing notes or source section?
75	IntervalCorrectedHeatFlow	Currently not populated. It is pulling 'heat_flow_mw_m2' field into IntervalUncorrectedHeatFlow but should probably leave blank and instead put 'heat_flow_mw_m2' into IntervalCorrectedHeatFlow
76	SE_CorrectedHeatFlow	not provided, but should have. Discuss prepopulating with a set value or referencing notes or source section?
77	IntervalHF_QualityCode	not provided, but should have. Discuss prepopulating with a set value or referencing notes or source section?
78	HF_QualityStatisticsMethod	not provided, but should have. Discuss prepopulating with a set value or referencing notes or source section?
79	HeatGenerationUnits	Already prepopulated.
80	HeatGeneration	Currently not populated, but expect to use calc_basement_heat_prod. Verify equivalent since not measured value.
81	HG_NumberSamples	consider prepopulating with 'estimated', if free text permitted by template
82	WaterTableTop	not provided by Cornell.
83	WaterTableTemperature	not provided by Cornell.
84	LithologyInformation	Currently not populated, but could use cosuna_section and sed_thickness_m
85	FormationName	Currently populated using tdfom for formation name followed by a formula based on tvd_m. that doesn't work properly if no tvd given. Can omit depth here.
86	Notes	Cornell should have provided. Could use Publication Source and Notes if can't add now.
◀ ▶ ... SMU BHT Submission SMU Side by Side Example Cornell Summary Example		

Figure 13d, Tab 5, Table 2 (columns F-G), Bottom Portion (Rows 58-86). Comparing column headings from the Heat Flow Content Model to Cornell's input file. Shading indicates need for discussion or action.

	I	J	K
4	TABLE 3- Correlates the Borehole Temperature Column Names to the Cornell names		
5	GTDA WFSOutput for BoreholeTemperature		
6	(have not checked against standard for template)	Some Change Suggested – or at least review Cornell->Borehole	
7	Yellow means in official content model but not in GTDA.		
8	FID	Does FID = OBJECTID?	required
9	ObservationURI	Generated by GTDA-not sure of convention. Content model description sounds like it was to reference either API or otherid. Verify ok.	required
10	WellName	direct match to well_nm and if blank, uses API.	required
11	APINo	direct match to api_wellno.	optional
12	HeaderURI	"https://geothermal.smu.edu/header/" FID without contributor name. Also, is the 's' in https okay? not in the Cornell dataset, but sounds like this is where the SMUID number or the BEG number etc would go.	required
13	OtherID	The metadata is supposed to describe the naming	optional
14	OtherName		optional
15	BoreholeName		conditionally r
16	Label	"borehole observation for well" WellName (well_nm or API)	required
17	Operator	currently not used, but is direct match with operator.	optional
18	LeaseName	Currently populated with permitID. Ask if better to use for LeaseNo instead.	optional
19	LeaseOwner	Currently populated with operator. Not equivalent.	optional
20	LeaseNo	Currently not populated. Ask if permitID should go here instead of under LeaseName.	optional
21	SpudDate		optional
22	EndedDrillingDate	Direct match with td_date.	optional
23	WellType	Not in GTDA delivery for borehole temperature, but is a direct match with Cornell's well_type.	optional
24	Status	any value provided in well_status. Only use well_status. If unavailable, state 'unknown'.	required
25	CommodityOfInterest	Currently not populated. (which is permissible) Providing well type instead, when known.	optional
		Discuss how to handle the 3 provided. Content model	
◀ ▶ ... SMU BHT Submission SMU Side by Side Example Cornell Summary Exam			

Figure 14a, Tab 5, Table3 (columns I-J), Top Portion (Rows 9-25). Comparing column headings from the Borehole Temperature Content Model to Cornell's input file. Shading indicates need for discussion or action. The words 'required' or 'optional' refer to whether the field is a required element of the Borehole Temperature Content Model, per the NGDS standards.

	I	J	K
4	TABLE 3- Correlates the Borehole Temperature Column Names to the Cornell names		
5	GTDA WFSOutput for BoreholeTemperature		
6	(have not checked against standard for template)	Some Change Suggested – or at least review Cornell->Borehole	
7	Yellow means in official content model but not in GTDA.		
26	StatusDate	Discuss how to handle the 3 provided. Content model states this date should be the date of the status	optional
27	Function		optional
28	Production		optional
29	ProducingInterval		optional
30	ReleaseDate		optional
31	Field	Direct match with field. Repeated where available for same API but was blank. (good!)	optional
32	OtherLocationName	Currently pulling farm, pool and quad. Please pull town and municipality, too.	optional
33	County	using coordiantes	required
34	State	using coordiantes	required
35	PLSS_Meridians	not provided/not needed	optional
36	Township	municipality and town provided, but not certain enough to use here. Ok to leave unpopulated for Cornell dataset.	optional
37	Range	not provided/not needed	optional
38	Section	Direct match to map_section.	optional
39	SectionPart	not provided/not needed	optional
40	Parcel		optional
41	UTM_E		optional
	UTM_N		optional
◀ ▶ ... SMU BHT Submission SMU Side by Side Example Cornell Summary Exar			
READY			

Figure 14b, Tab 5, Table3 (columns I-J), Middle Portion (Rows 26-41). Comparing column headings from the Borehole Temperature Content Model to Cornell's input file. Shading indicates need for discussion or action. The words 'required' or 'optional' refer to whether the field is a required element of the Borehole Temperature Content Model, per the NGDS standards.

	I	J	K
4	TABLE 3- Correlates the Borehole Temperature Column Names to the Cornell names		
5	GTDA \WFSOutput for BoreholeTemperature		
6	(have not checked against standard for template)	Some Change Suggested – or at least review Cornell->Borehole	
7	Yellow means in official content model but not in GTDA.		
	UTM_N		
42			optional
	UTMDatumZone		
43			optional
44	LatDegree	direct match with decimal_latitude; verify with Cornell that they used WGS 84	required
45	LongDegree	direct match with decimal_longitude; verify with Cornell that they used WGS 84	required
46	SRS	Already prepopulated. However, 1. Cornell needs to confirm they used WGS84 AND 2. content model specifies use EPSG:4326 to indicate WGS84.	required
47	LocationUncertaintyStatement	Not in GTDA's version, but should be using with either offset info or prepopulated stmt. Note is a required	required
48	LocationUncertaintyCode		optional
49	LocationUncertaintyRadius		optional
50	DrillerTotalDepth	Potential mismatch. Currently pulls in log_depth_m and should pull in calc_depth_m. In reviewing ones	optional
51	DepthReferencePoint	not provided by Cornell. Okay to leave empty.	optional
52	LengthUnits	Already prepopulated.	optional
53	WellBoreShape	generate if logged depth far exceeded vertical depth? Not sure that would be worth doing.	optional
54	TrueVerticalDepth	direct match to tvd_m. Discuss if appropriate to use calculated depth where tvd_m is not populated. Think probably not put anything other than tvd_m there.	optional
55	ElevationKB	Currently not populated since Cornell elevation reference unknown. Need to find out where their elevations are from.	optional
56	ElevationDF	Currently not populated since Cornell elevation reference unknown. Need to find out where their elevations are from.	optional
57	ElevationGL	Currently not populated since Cornell elevation reference unknown. Need to find out where their	optional
◀ ▶ ... SMU BHT Submission SMU Side by Side Example Cornell Summary Ex			

Figure 14c, **Tab 5, Table3** (columns I-J), **Middle Portion, Continued** (Rows 42-57). Comparing column headings from the Borehole Temperature Content Model to Cornell's input file. Shading indicates need for discussion or action. The words 'required' or 'optional' refer to whether the field is a required element of the Borehole Temperature Content Model, per the NGDS standards.

I		J	K
4	TABLE 3- Correlates the Borehole Temperature Column Names to the Cornell names		
5	GTDA WFSOutput for BoreholeTemperature		
6	(have not checked against standard for template)	Some Change Suggested – or at least review Cornell- >Borehole	
7	GTDA.		
58	FormationTD	Currently direct match to tdfom with "=" and the total vertical depth rounded. Some populating with -1 if had formation name in FormationTD but nothing in the TVD column. Discussed with Dmtry to modify the -1.	optional
59	BitDiameterCollar	not provided, thus not populated.	optional
60	BitDiameterTD		optional
61	DiameterUnits	Already prepopulated with inches.	optional
62	Notes	Not in GTDA version, but helpful to have.	optional
63	MaximumRecordedTemperature	'0'.)	optional
64	MeasuredTemperature	Direct match to bht_c.	required
65	CorrectedTemperature	Direct match to harrison_bht_c.	optional
66	TemperatureUnits	Already prepopulated.	required
67	TimeSinceCirculation	not provided, thus not populated.	optional
68	CirculationDuration	not provided, thus not populated.	optional
69	MeasurementProcedure	Currently not provided and not in the GTDA WFS borehole temp WFS layout. Note a required field. Need to check with Cornell, but could prepopulate with a statement similar to "BHT values from oil and gas well log headers. Temperatures corrected for drilling disturbances using the SMU/Harrison methodology. More detailed explanation available at...Explanation of the Quality Analysis review, too."	required
70	CorrectionType	Currently not populated, but consider prepopulating with 'Harrison'. Content model is unclear about the 'controlled vocabulary'.	optional
71	DepthOfMeasurement	Direct match to logdepth_m.	optional
72	MeasurementDateTime	Currently not populating, but should be direct match to log_date.	optional
73	MeasurementFormation		optional
74	MeasurementSource	currently not provided and not in GTDA borehole temp Layout. Note a required field. Need to check with Cornell, but could prepopulate with one	required
75	RelatedResource	Not in GTDA's version, but shouldn't we be redferencing the heat flow data and/or the estimated temp - depth info and/or maps?	optional
76	CasingLogger	not provided, thus not populated.	optional
77	CasingBottomDepthDriller	not provided, thus not populated.	optional
78	CasingTopDepth	not provided, thus not populated.	optional
79	CasingPipeDiameter	not provided, thus not populated.	optional
80	CasingWeight	not provided, thus not populated.	optional
81	CasingWeightUnits	Already prepopulated.	optional
82	CasingThickness	not provided, thus not populated.	optional
83	DrillingFluid	not provided, thus not populated.	optional
84	Salinity	not provided, thus not populated.	optional
85	MudResistivity	not provided, thus not populated.	optional
86	Density	not provided, thus not populated.	optional
SMU BHT Submission		SMU Side by Side Example	Cornell Summary Example
READY			

Figure 14d, Tab 5, Table3 (columns I-J), Bottom Portion (Rows 58-85). Comparing column headings from the Borehole Temperature Content Model to Cornell's input file. Shading indicates need for discussion or action. The table continues on for 6 more rows, all optional fields that were not provided and thus not populated.

	L	M	N	O	P
4	Field List				
5					
6	Interchange Content Element	Data Type	Implementation	Cardinality	Element Description
7					
8	OBJECTID	int	int	1	Added by GIS software.
9	ObservationURI	URI	string 255	1	Unique identifier for this observation record. This is the identifier (Uniform Resource Identifier) that will be used to cross-reference logs, samples, and other observations to the well or borehole represented by this header. If available, the API (American Petroleum Institute) assigned number is the preferred well name reference. If an APINo is not available, observationURI should include one of the 'OtherID' values (below).
10	WellName	free text	string	1	Common or human-readable name by which the feature is known. Recommend using only web-safe characters (a-z A-Z 0-9 _.) in the name. Be consistent in the naming convention used within any given dataset.
11	APINo	free text	string	0..1	API number for the wellbore/borehole identified by the HeaderURI. If an API number is unavailable this field is empty. Usage of hyphens is optional, according to standard practice in each state. API number refers to the American Petroleum Institute number. Each oil, gas, or geothermal well drilled in the United States has a unique API number; this number identifies all boreholes or recompletions associated with the original wellbore. See http://www.snowla.org/technical/api-technical for more information about
12	HeaderURI	URI	string 255	1	Unique identifier for this observation. This is the identifier (Uniform Resource Identifier) that will be used to cross-reference data, samples, and other observations to the observation represented by this record. Best practice is to define an http URI that will dereference to a normative description of the observation. USGIN URI's may be redirected to any URL using the http://resources.usgin.org redirector; contact the USGIN development team at AZGS for instructions.
13	OtherID	free text	string	0..1	Alternative identifier(s) for the feature. Indicate the authority from which the OtherID is derived with a short (lower case) prefix delimited by a colon ':'. If identifiers from multiple different authorities are available, delimit these identifiers with the pipe ' ' character. For example, if a well were known to the Arizona Geological Survey (AZGS) by the id number 1337, the prefix would be azgs: and the identifier string would be azgs:1337. Any such prefixes used to scope identifiers should be explained in the dataset metadata.
14	OtherName	free text	string	0..1	Other human-intelligible labels for feature such as secondary name or common name, other spellings. Indicate the authority from which the OtherName is derived with a short (lower case) prefix delimited by a colon ':'. If identifiers from multiple different authorities are available, delimit these identifiers with the pipe character ' '. For example, if a well were known to the Arizona Geological Survey (AZGS) by the id number 1337, the identifier string would be azgs:1337. Any such prefixes used to scope identifiers should
15	BoreholeName	free text	string	0..1	Human-intelligible name by which the borehole identified by the HeaderURI is known. Convention for wellName is: Name = Operator + Well No. + Lease Owner. A well may consist of multiple boreholes (wellbores). If these are distinguished by different records in this table, they should have the same well names, but distinct BoreholeNames, and at least one identifier (APINo or OtherID) that is unique to the borehole. Conditional obligation - if the HeaderURI identifies a borehole which is one of several within a well
16	Label	free text	string	1	Short text string for labeling the feature on maps. Can be the same as feature name. A label may be an OtherLocationName, OtherID or the Name or some combination.
17	Operator	free text	string	0..1	Name of company/organization that is operating the well during the reporting period.
	LeaseName	free text	string	0..1	Name or number of the lease of the well that contains the borehole identified by the HeaderURI. Formatting and punctuation are not proscribed, but should be consistent within a dataset.
◀ ▶ ... SMU BHT Submission SMU Side by Side Example Cornell Summary Example ⊕					

Figure 15: Partial listing of the **Borehole Temperature Content Model Field List**, describing each field name.

As stated before, Quality Assurance Process, is just that a ‘process’. The SMU Geothermal Laboratory and each of the project participants discovered previously unrecognized errors as a result of the exercise. While time consuming, it was a necessary component. Not all of the parameter mapping issues were fully resolved, leading to some of the Tier 3 (Content Model Formatted) datasets having data population that is not precisely correct. This issue is discussed further in the system limitations section of this appendix.

C. System Limitations

While the overall GTDA system meets the requirements of serving as a repository of geothermal data which can be shared with other nodes on the NGDS and can be added to over time, there are some aspects of the system that are lacking in either flexibility or functionality. The purpose of this section is to summarize these limitations such that users and systems programmers have an accurate understanding.

Data Submissions

One of the primary requirements for the system is to allow for updates to the system through the submission of refreshed data and/or new data. Import routines were written for all major datasets currently in the system, with the exception of MLKay submissions, since the MLKay submission was limited to a one time contribution for this project because of the volume of data and comparably limited updates since. While import routines were created, they are limited to whom may submit that type of data. For example, since UND was the only group to initially submit radioactivity data, only UND userids have the capability to submit radioactivity data; SMU cannot upload radioactivity data, even if formatted identically to the UND layout. There are two known workarounds to this issue: first, using a userid assigned to UND, SMU may append additional rows of data to the last UND submission using the UND data formatting. The 'distributor' text file containing the dataset metadata is updated to include additional contributor details. An alternative option would be for SMU to submit the data in content model format into the Geothermal Data Repository (GDR). In the case of radioactivity data, SMU opted for appending the data to UND's dataset and updating the metadata to reflect it is a joint submission.

Content Model Mapping

The system supports ten different 'tier 3' content models that are able to be shared across the NGDS, but there are errors in the 'mapping' for several. For example, the heat flow content model as described by the NGDS, has different fields for temperature data so that BHT values are distinguishable from equilibrium logged temperature values. SMU's submissions include both types of temperature values, but they are all currently displayed in the tier 3 delivery of the heat flow content model as BHT temperatures. Some mapping issues are relatively simple for a programmer to correct, and others would require significant effort. In the instance described above, the core database of GTDA does not distinguish between the two types of temperatures and thus there is no simple fix to correcting the content model output at this juncture. There are three known workarounds: first, the original submission files can be made available as 'tier 2' datasets prior to processing into the core GTDA database. This has been enabled, along with descriptions of the tier 2 formats used. Second, the submission files can be manually manipulated into the content model formats and submitted into the Geothermal Data Repository (GDR). Finally, SMU may benefit from a copy of the Node-In-A-Box software being developed by AZGS. While the tier 2 datasets have already been made available, the team is evaluating whether it would be better to remove the incorrectly mapped tier 3 datasets or to permit access but with a list of explanations regarding their use.

Related Resources

The system supports linking related resources, such as references, reports, field notes, well logs, and other information to specific wells and/or geographic regions, with some limitations.

- Items uploaded by well identifier, state and/or county are functioning properly, but users have been unsuccessful linking items based on coordinates (point and/or a bounding box). The root source of the problem is still not known and there is currently no alternative.

Fortunately, the majority of the items to be linked can be associated to one or more specific wells or to a county.

- A second limitation is the inability to add to the batch of related resources that currently appear in the well log content model. There are 2,102 SMU field notes and 3,639 SMU temperature depth curves appearing in the well log content model. SMU has been able to add field notes, but they do not appear in the content model; only those field notes initially loaded into the GTDA system by Siemens, prior to the existence of the upload interface, are output to the content model. All field notes added by SMU are able to be downloaded from the system via interactive use of the GTDA system, but this will not help users coming from other nodes of the NGDS realize there are additional material available. Additionally, we may associate files from the initial batch loaded into the GTDA by Siemens with new well sites and the system will update the content model output. This limitation is particularly concerning. The metadata for the Well Log Content Model delivery will be updated to alert users that they need to use the interactive interface to determine if additional field notes are available. It is not known if Texas Tech will be able to add to the 49,945 well logs they have already provided, as we do not have additional well logs to test this capability.
- The BEG's core log library links (269,722 URLs) do not appear in the well log content model output, but they do appear through use of the interactive interface. Additionally, they can be made available as a discrete tier 2 dataset for download. The files and links are also directly available on the BEG website.
- Several system requirements of the upload interface not previously known to the submitters were discovered in May 2014, which will necessitate an update of the documentation. For instance, when uploading any related resources, the submitter must provide a file that associates the files with either a well, a series of wells, or a geographic region. The user must specify the 'type' of related resource in a column titled 'type'. There are only 3 valid choices: Field Note, Related Resources, and TD Curves. The submitters did not realize that Field Note was singular and that Related Resources was plural until many failed attempts. It was also not initially understood how a type of 'Field Note' differed from a type of 'Related Resources'. Similarly, when uploading items to be associated with a county, the user must state the word 'county' after the county name within the column titled 'county'. These, and other procedures, are now better understood and previously submitted files are being re-uploaded for the GTDA system to access them properly. It is possible that the system's inability to upload related resources associated with coordinates is due to some procedural idiosyncrasy of data entry requirements.
- Unfortunately, the system does not indicate the nature of the submission problem to the user, and the 'staged' files appears as though the upload has worked properly, when often it has not. For quality control and to make sure all desired data and files get into the GTDA system, it is imperative for the submitter to take the time to review within the GTDA interface that the recently submitted data/files are discoverable. When performing the check, it should be done for multiple locations listed within the uploaded file. Submission errors are also made by the submitter and it is only through the final checking process that these can be determined.

Publications – CSW Records

The system supports uploading publications, searching for relevant publications, access to the publications when permissible, and sharing of bibliographic data with the NGDS, with limitations.

As best we can determine, all publications loaded into the system by Siemens seem to be accessible without issue. Many of those uploaded by SMU using the GTDA upload interface are also accessible, but many are not.

- There are over 24,000 CSW records for publications that have been shared by the SMU node to the central NGDS node and are now accessible via geothermaldata.org. However, the full bibliographic information is not displayed and thus, if it is a copyrighted publication, the user might not have enough information to locate the item in a library. Many authors have written papers with similar titles in the same year and frequently the report number or journal name and volume are needed to distinguish them. A workaround that seems to mitigate this is to include this bibliographic reference information into the keyword section, which *usually* shows up in the CSW information and is thus discoverable to users on geothermaldata.org.
- In the above statement the word ‘*usually*’ is specified because system testing revealed that the keyword information, and indeed any bibliographic data, cannot be updated by SMU or the other project partners if the record is ‘owned’ by another organization. In the interest of not filling the CSW with duplicate entries from GRC, OSTI, Stanford, EGI, Geoheat Center, SMU, Cornell, UND, and BEG, the software goes through the list in a predetermined order. Because OSTI is higher in the ordered list than SMU, any updates SMU makes to a bibliographic record that is a duplicate entry of another group are not read into the CSW record for that item. SMU discovered this when trying to determine why updates to the reference for “Thermal Ground Water Flow Systems in the Thrust Zone in Southeastern Idaho” were not showing up in the CSW record on <http://geothermaldata.org>. Review of the GTDA database revealed this same publication appears in bibliographies from 6 different organizations, one of which was OSTI. Only the bibliographic data provided by OSTI is being passed through for creation of CSW record for the publication. Unfortunately, the OSTI entry happens to be missing the second author. Because of the deduplication order, a NGDS user cannot see the report number data SMU added to the keywords to help find the item, and we do not have a means to correct the missing second author. The list of bibliographic entries is currently 117,923, so clearly the list needs to be de-duplicated, but this presents a problem if the ‘owning’ organization has a less than complete record. We do not have a workaround at this time. Perhaps if SMU was ordered first in the list, it would enable corrections, but we do not know this to be the case at this time.
- The GTDA system uses an open source product called solr to index the publications. Only the unique items are to be indexed and have CSW records created for them. Of the 117,923 bibliographic entries in the GTDA database, 24,632 are indexed. In reviewing the bibliographic entries from *just* the SMU Geothermal Laboratory (not the full team) there are 2367 unique records, only 364 of which are indexed. SMU has uploaded more than 364 publication PDFs. Some were likely not indexed because another organization ‘owned’ the record, as described in the previous paragraph. Others were known to be unique to our list, and were still not indexed. They were resubmitted with again with hopes that the removal of some records containing dashes in a numeric field will allow the rest of the SMU publications to index properly. We will need to re-verify that all submitted publications are indexing properly.
- Another limitation with CSW entries for publications relates to those where a URL to the publication was provided, rather than a PDF of the publication. SMU collected over 500 URLs for links to publications, a large portion of which do not appear to have CSW entries associated with them. An example is SMU publication number 87, a GSA Bulletin article from 1956 entitled “Wrench Fault Tectonics”. Since it was copyrighted, we provided a link only, but because it was not indexed, there is no CSW entry in the NGDS, nor is it available from the

interactive interface of GTDA. Of those that do have CSW entries, the URL is not in the CSW record available on <http://geothermaldata.org>; instead, there is just a generic warning that SMU does not have the distribution rights to the file. An example is SMU publication 227, “Geology Along US Highway 93 in Idaho”. A URL to this 1963 pamphlet ([www.idahogeology.org/PDF/Pamphlets_\(P\)/P-130.pdf](http://www.idahogeology.org/PDF/Pamphlets_(P)/P-130.pdf)) was provided in the bibliographic record uploaded by SMU, but the URL is not coming through in the CSW entry from SMU to geothermaldata.org, nor does it appear on the interactive interface in GTDA.

Publications – Searching within GTDA Interface

The interactive interface of GTDA has the capability to perform full text searches of the uploaded publications. There are some limitations in the searching capability of which users should be cognizant. For instance,

- The search logic does not recognize partial words, without use of an asterisk (*). Thus searching ‘volcano’ will only find the singular instance and will not find ‘volcanoes’, unless an asterisk is used as in ‘volcano*’. This is problematic when searching for author names, as a large number of papers have a superscript after the author’s name, identifying the institution of the author. Searching for ‘Frone’ will not be able to find ‘Frone²’, unless the user specifies ‘Frone*’ or ‘Frone²’ in the search field. In the case of GRC papers with many authors, the lead author appears on each page followed by et al, but the other authors only appear once followed by the superscript.
- Selecting the ‘full text’ search button yields different results than pressing the ‘enter’ button. This extends to the use of the * described above.
- It is difficult to sort through publications containing commonly found words, as it is not a keyword search but whatever text is contained in the article. However, keywords are passed along in the CSW records, and are thus discoverable via keyword search using the geothermaldata.org interface. Items retrieved may be sorted by title, date, or author, but some queries pull in thousands of results.
- The bibliography is searchable as a discrete dataset, which helps narrow the search considerably. However, only title, author and year are queried; keywords or report names are not searched, even when contained in the bibliographic information in the database. Again, keywords are passed along in the CSW records, and are thus discoverable via keyword search using the geothermaldata.org interface. Unfortunately, because only one bibliographic record per publication is indexed and thus searchable on the NGDS, the one with report numbers or journal names in the keywords may not be the instance of the bibliographic record that is in the NGDS catalog.
- The search functionality does not make use of synonyms identified in the keyword list created by the team. This is apparent when searching for plural versions of words (‘volcano’ versus ‘volcanoes’) or phrases commonly referred to with acronyms (‘EGS’, ‘Enhanced Geothermal Systems’ or ‘Engineered Geothermal Systems’).

Despite these limitations, the system is so much more powerful than a physical library and has enabled us to discover important, but obscure, research papers. SMU has written [‘Frequently Asked Questions’](#) to assist the users and will continue to augment these with helpful hints for use.

Corrupted Sites

Earlier in 2014, during a period of code changes in GTDA, the wells in SMU heat flow data submission briefly became identified by their secondary, non-unique name, rather than by their primary unique

identifier. Related resources uploaded during this period corrupted at least one site, such that it is no longer viewable in the interactive interface of the GTDA system (505 error). Fortunately, very few sites were impacted, but there is no known means for SMU to restore or remove it. SMU will create a record with the same latitude and longitude, the same shorthand non-unique name, but a different unique identifier name. It will be uploaded with a field note explaining that it is to replace the inaccessible site. If users search for the corrupted site using the non-unique shorthand name we believe it should pull up both sites.

Distributor Text Files Currency

The 20 datasets shared with the NGDS each have a CSW record such that they are discoverable and accessible. The metadata for each file is generated by information contained in text files that are submitted at the time of data upload. These files are named distributor.txt and each data submitter may modify the metadata (authors, contact information, abstract, keywords, etc.) at their discretion. The previous uploaded dates set are accessible to the user at the time of their new data submission, along with the distributor file used at the time of the last upload. Overall, this is a sound approach, as submitting organizations will undoubtedly have turnover and this prevents submitters from making updates to a version of their data that is multiple iterations old. We have learned that there is the potential for the distributor file to become out of date for the three organizations that submit more than one dataset: BEG, SMU, and UND. For example, BEG has a file called 'master wells' and another called 'reservoir', both of which are uploaded with their distributor.txt file containing metadata about both files. If they upload a refreshed version of the master wells dataset with modifications to the distributor.txt file, there is potential to become out of sync with the next refresh of their reservoir dataset at some point in the future. We discovered this when SMU wanted to submit radiogenic heat data, appended to the UND dataset. SMU did not realize that UND had a more recent version of their distributor.txt file than the one associated with UND's last radiogenic heat data submission. An outdated version was submitted and data import problems followed. Now that we are aware of this potential issue, the documentation will be updated to instruct data submitters to check the timestamps for all datasets and pull the distributor.txt file used with the latest time stamped submission, even if it was a different dataset.

D. Conclusion

The SMU Geothermal Laboratory and our partners have worked very diligently over the last 4 years collecting and analyzing 40+ years' worth of heat flow and other geothermal research. Every day the number of users increases along with favorable responses. Just this week the SMU Geothermal Laboratory received a question from an individual working for the USGS looking for temperature versus depth plots to estimate reservoir temperatures in the Permian Basin. We directed him to the SMU Node of the NGDS, where he happily reported being able to find the information he needed. Success!