SAND2019-10824C

Citadel: Tracking Data Provenance and Synchronizing Across Multiple Instances

Aaron Comen

Citadel is a custom code framework designed to quickly roll out multiple data storage applications at Sandia National Laboratories. Citadel helps projects by providing features common to all data system applications. These include: enabling CRUD operations, tracking changes to the data (via a user accessible provenance database), parsing common data types into Parquet files, opening RESTful API's for third party tools to interact with the data (see posters on WASP), and establishing custom access controls to data on a granular level.

Currently the two implementations of Citadel are the SEDS and DataSEA projects. SEDS focuses on replacing a large legacy data system. It has hard-coded data structures and rules for accessing data. DataSEA focuses on quickly rolling out smaller data systems that allow the users to determine their own data structures and custom rules for storing, accessing, and manipulating data.

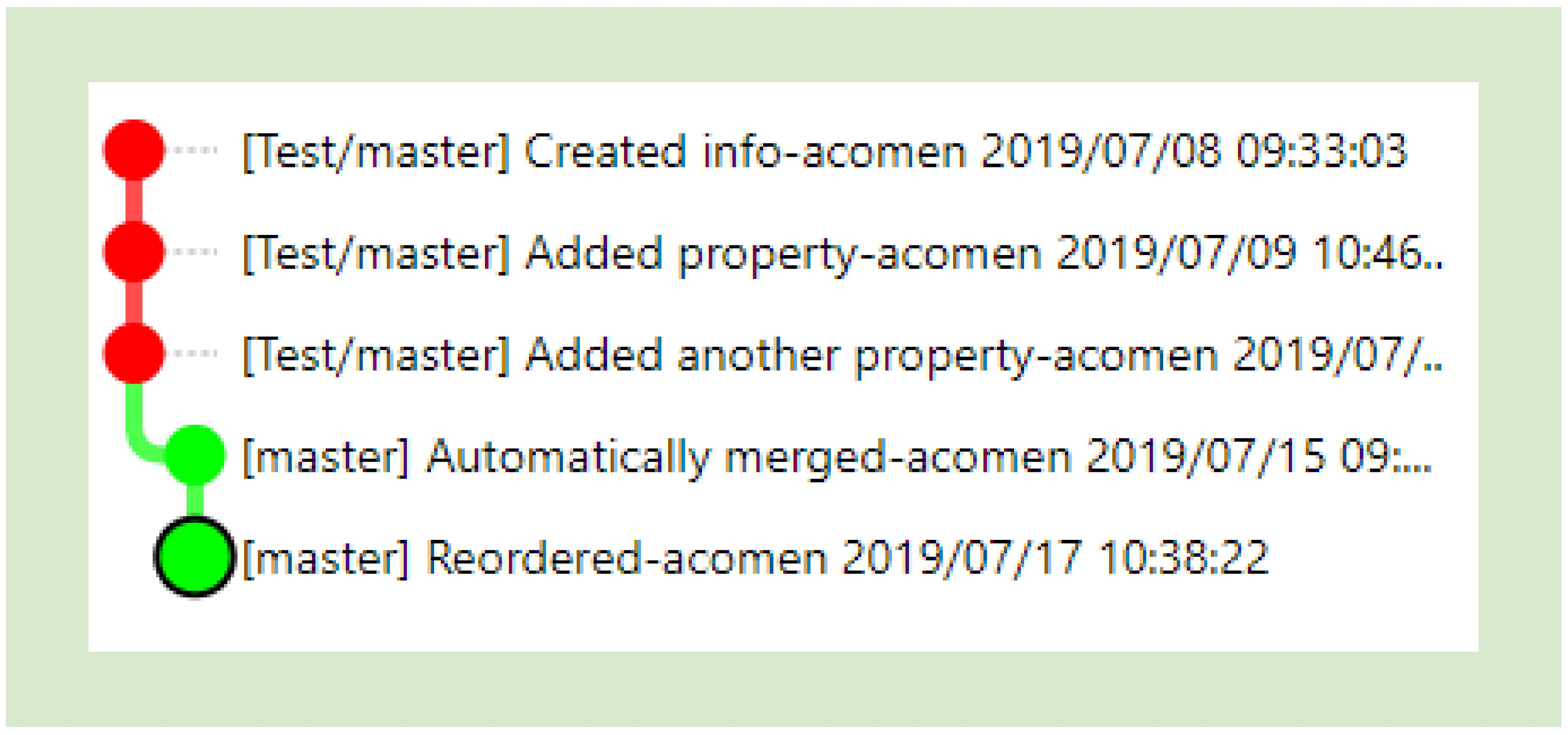
Data Provenance

A key feature of these systems is tracking data provenance for all data stored in the systems. This includes tracking author, create time, and revision history for all data. To accomplish this Citadel leverages the graph database Neo4j to store this provenance information. Citadel stores all data and associated metadata as records within a MongoDB datastore and assigns all created records a universally unique revision ID. Records are immutable and any revision to a record creates a new record in the database. These records are referenced via these revision IDs from nodes stored in the graph database which contain information about the revision such as author and reasoning for the revision. The edges between these nodes express the relationship between these records, and can support a branching structure if needed. Records which are revisions of each other share a provenance ID.

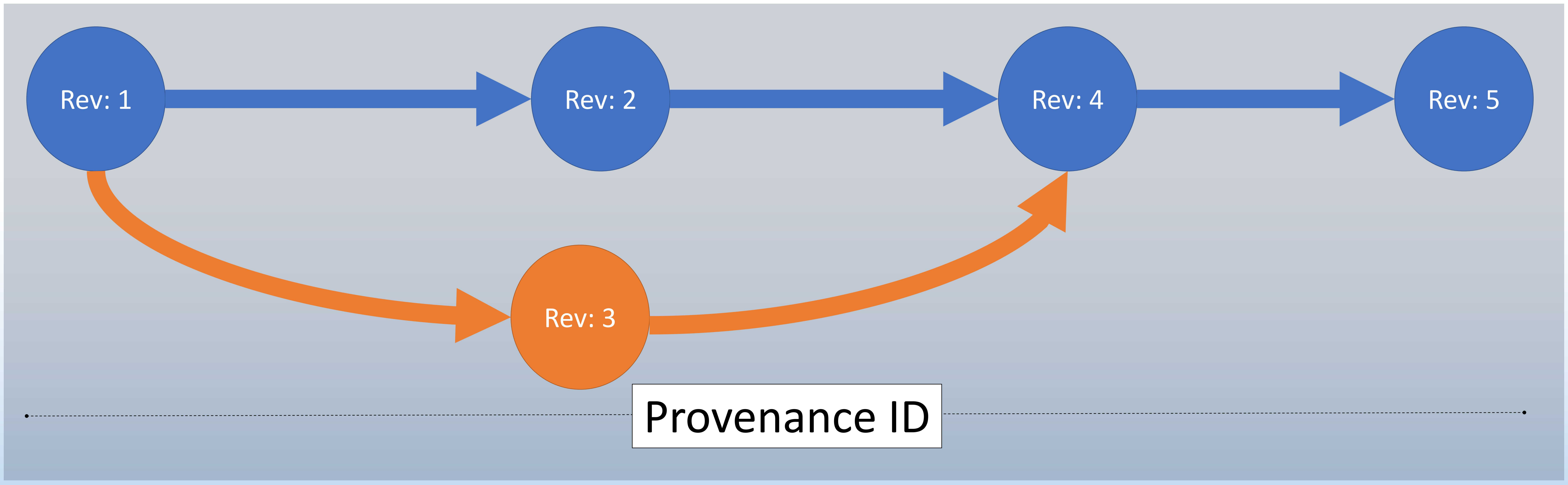
Using the information in this graph database, citadel can construct a Directed Acyclic Graph (DAG) representing the provenance of the data, which allows for easy inspection of changes to data over time. This DAG is rendered into a graph which clearly explains this history of the data and allows the users to see previous revisions, giving end users insight into who is changing their data, how it is changing, and why it is changing.

Data Synchronization

Data which is inputted into citadel instances can be produced in a variety of environments, including standalone environments running locally on laptops which are not connected to a network, where the data must later be synchronized to a main instance running on a server. The history of this data should still be tracked, and the source of this data should be made clear to users on the main system. When data is synchronized, all new records are placed in the datastore using their unique revision IDs as identifiers. When merging revision histories of data, edits done on a remote host are represented as a separate branch. If possible, the branch is merged back into the main branch when data is synchronized. This creates revision histories which easily display where data was modified and who modified it.



```
[Test/master] Created info-acomen 2019/07/08 09:33:03
[Test/master] Added property-acomen 2019/07/09 10:46..
[Test/master] Added another property-acomen 2019/07/..
[master] Automatically merged-acomen 2019/07/15 09:...
[master] Reordered-acomen 2019/07/17 10:38:22
```


Rev: 1
Rev: 2
Rev: 4
Rev: 5
Rev: 3
Provenance ID