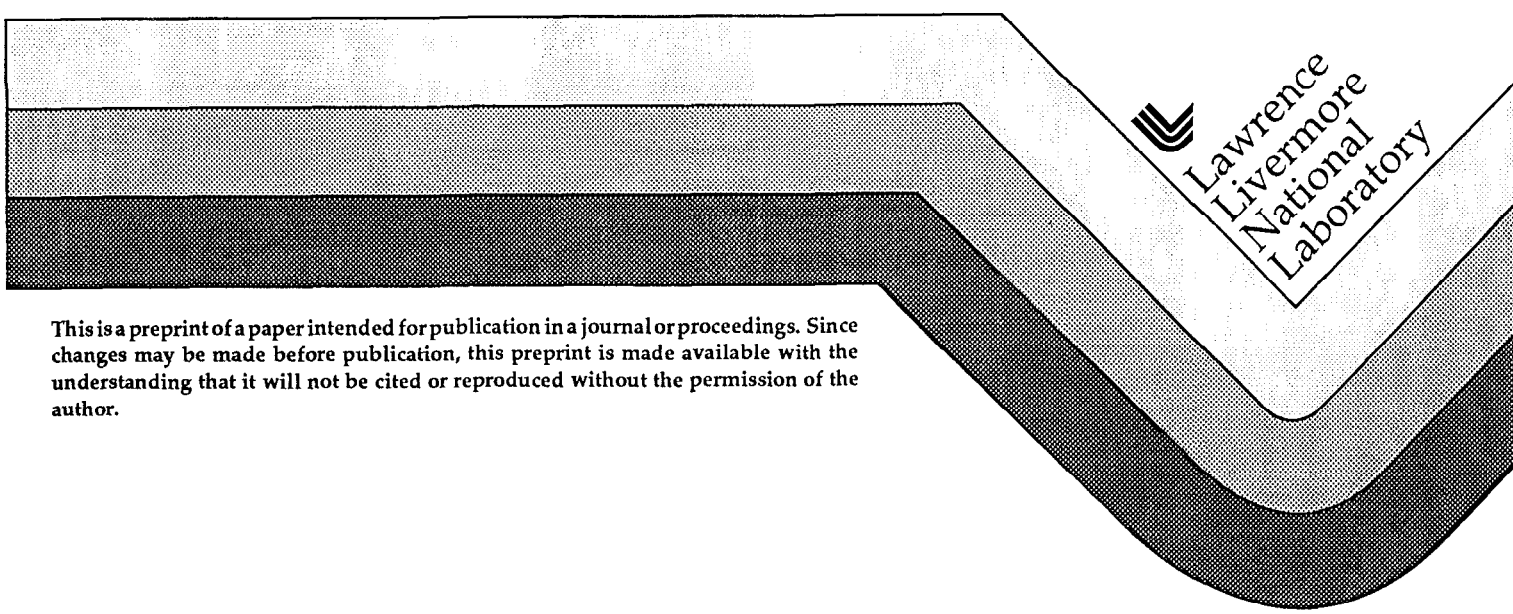
UCRL-JC-129930
PREPRINT

Automatic Generation of Warehouse Mediators Using an Ontology Engine

T. Critchlow, M. Ganesh, R. Musick

This paper was prepared for submittal to the
5th International Workshop on Knowledge Representation
Meets Databases (KRDB'98)
Seattle, WA
May 31, 1998

March 4, 1998



Automatic Generation of Warehouse Mediators Using an Ontology Engine

T. Critchlow, M. Ganesh, R. Musick
Center for Applied Scientific Computing
Lawrence Livermore National Labratory (LLNL)
{critchlow | ganesh | rmusick }@llnl.gov

1. Introduction

The DataFoundry research project at LLNL is investigating data warehousing in highly dynamic scientific environments. Specifically, we are developing a data warehouse to aid structural biologists in genetics research. Upon completion, this warehouse will present a uniform view of data obtained from several heterogeneous data sources containing distinct but related data from various genetics domains. Our warehouse uses a mediated data warehouse architecture in which only some data is represented explicitly in the warehouse; remote access is required to obtain the non-materialized data. ***Mediators*** are used to convert data from the data source representation to the warehouse representation and make it available to the warehouse.

The major challenge we face is reducing the impact of source schema changes on warehouse availability and reliability: based upon previous efforts, we anticipate one source schema modification every 2-4 weeks once all of the desired sources have been integrated. Incorporating these modifications into the mediators using brute force results in an unacceptable amount of warehouse down-time. We believe that extensive use of a carefully designed ontology will allow us to overcome this problem, while providing a useful knowledge base for other applications. In addition to automatically generating the transformation between the data sources and the warehouse, the ontology will be used to guide automatic schema evolution, and provide a high level interface to the warehouse.

This paper focuses on the use of the ontology to automatically generate mediators, because reducing the effect of source changes is a critical step in providing reliable access to heterogeneous data sources. An overview of the role mediators play in this process is provided in the next section. Section 3 briefly describes the ontology, and Section 4 outlines how it is used to generate the mediators.

2. The Role of Mediators

Figure 1 outlines our approach to loading the warehouse: parsing the data, transforming it to the desired format, and entering it into the warehouse. In practice, these steps are not always distinct. Often, a single program will parse the input file, and transform the data before storing it in an internal specification. This internal representation can then be directly entered into the warehouse. This intermingling of the parser and mediator is permitted because the mediator API is rarely defined. However, a carefully designed API is critical to reducing the maintenance requirements of the warehouse; if the API does not remove or redefine methods once they are created, the ontology and warehouse can evolve without affecting the parser.

We have defined a set of ontologies that describe the data sources and the warehouse, the abstract concepts being represented, and the transformations required to

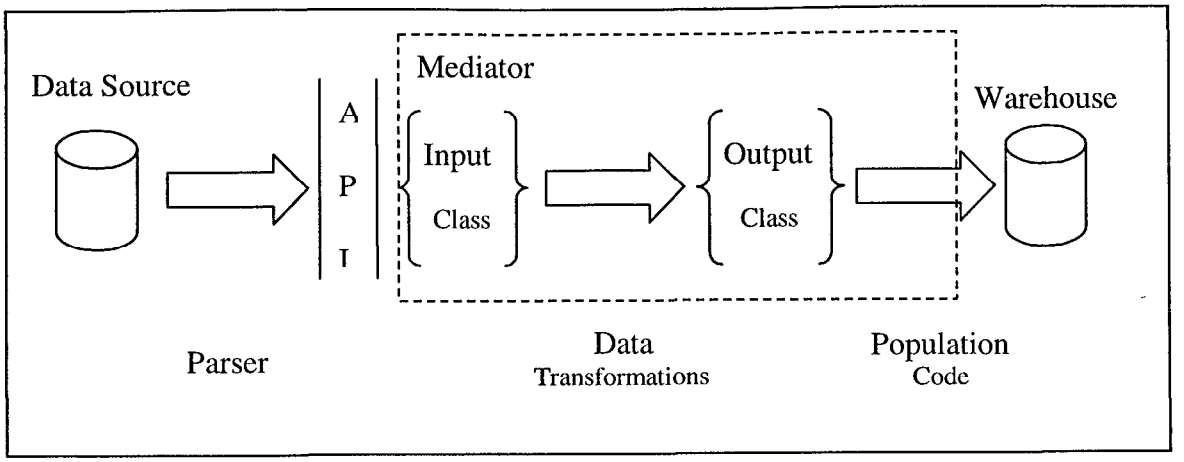


Figure 1 The Integration Process

map between them. In Section 4, we discuss how our **ontology engine** (OE) automatically constructs the API and mediators based on these ontologies. Unfortunately, we are not yet able to either automatically generate the initial parser for a data source, or modify it when a source schema changes.

3. The Ontology

Our ontologies are represented as a single repository defined in Ontilingua [1]. It contains abstractions of domain specific concepts, database descriptions, mappings between them, and transformation functions to resolve differences in representation. Figure 2 contains an example of the first three types of information, which are discussed below. The transformation functions are not shown, but are simply methods associated with an abstraction. Strict naming conventions ensure the source and target representations are easily identifiable.

Abstractions are the heart of the ontology. They contain the aggregate of all information known about a particular domain concept. Each concept is represented by a collection of attributes representing the various components of the concept. These attributes can be grouped into characteristics, combining related attributes or alternate representations. For example, the abstraction *atoms* has a characteristic representing its *position*, and that characteristic has three attributes for the cartesian coordinates that make up its 3-D position.

A database description consists of language independent class definitions that closely mirror the physical layout of a relational database. This information may be automatically obtained from the metadata associated with most DBMSs; the table name is followed by a list of the column names, data types, and arity. Unfortunately, most flat file data sources do not maintain any metadata, so this information must be manually entered for them.

Because an abstraction contains aggregate information about a concept, including all alternative representations used by the data sources, there is always a direct mapping between database attributes and attributes of the corresponding abstraction. However, due to representational differences this mapping may require data from multiple classes (i.e. a join). Our example demonstrates how information associated with the *atoms* abstraction is obtained from both the warehouse *atom* and *res_in_model* classes.

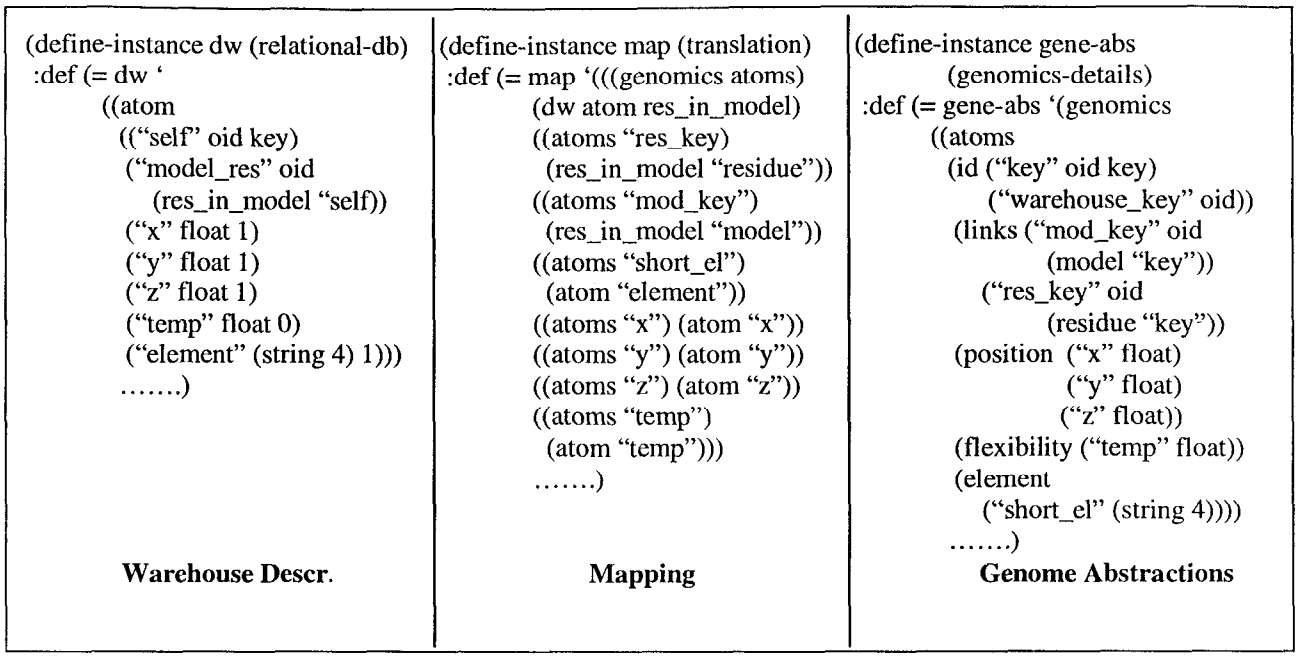


Figure 2 Examples of Ontology Data

4. Generating the Mediator

Figure 3 outlines how the information expressed in the ontology relates to the various components of the mediator. The entire interface and the vast majority of code will be automatically generated from the ontology using the OE – these components are shown with a solid frame. The mediator functionality is decomposed into two components: the translation library and the mediator class. The API available to the parser is a combination of the individual APIs. The translation library is a C++ library containing a set of classes corresponding to, and automatically derived from, the abstractions; of course, the transformation methods must be explicitly entered. The mediator uses the abstraction classes and mapping ontology to perform the transformation from the input data format, obtained from the parser, to the warehouse representation, as described by the ontology. While converting data to the target representation may require multiple steps (based on the methods available) the naming convention makes this a relatively straightforward search process. Once the data transformation has been performed, the SQL interface is used to load the data into the warehouse.

Incorporating a new data source requires the DBA to describe the data source, map the source attributes to corresponding abstraction attributes, ensure that all applicable transformation methods are defined, and create the parser. The OE then creates the new mediator class, and expands the data class API if needed. Once a database has been integrated, adapting to minor source schema changes often requires only modifying the parser to read the new format. Significant changes in the data representation may require the ontology to be modified and a new mediator created.

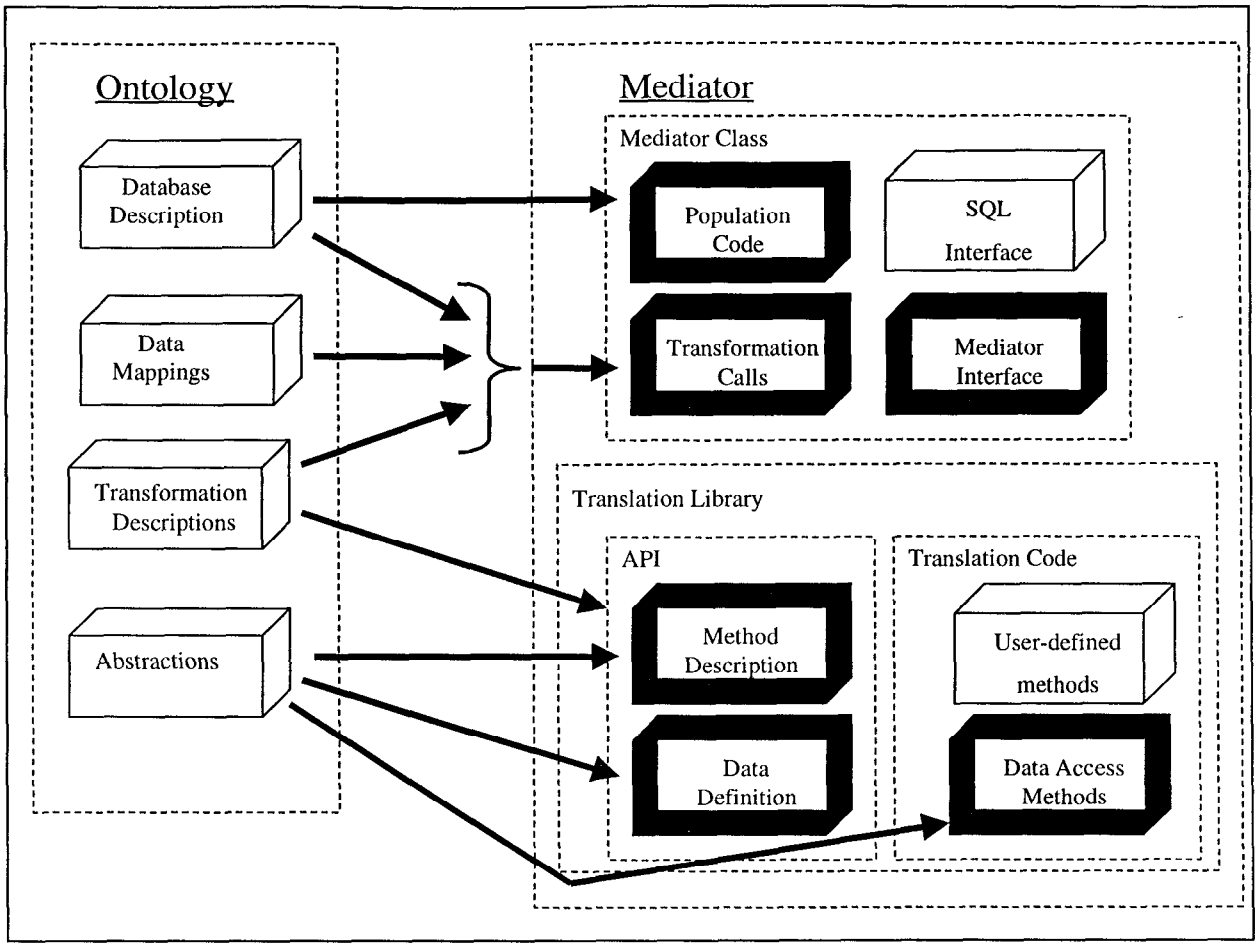


Figure 3 The Ontology and Mediator

5. Conclusion

In a dynamic scientific environment, maintaining the consistency and availability of a data warehouse requires quickly adapting to changes in the source schemata. We believe our extensive use of an ontology to represent information about the participating databases will dramatically reduce the effort required to manage these changes. We have defined the ontology data, and are currently coding the OE. We expect to have a functional prototype in place by July 1998, at which time we will begin exploring other uses for the ontology information.

References

[1] T. Gruber. Ontolingua: A Mechanism to Support Portable Ontologies. Stanford. Knowledge Systems Laboratory. Tech Report KSL-91-66. November 1992.

Work performed under the auspices of the U.S. DOE by LLNL under contract No. W-7405-ENG-48

Technical Information Department • **Lawrence Livermore National Laboratory**
University of California • Livermore, California 94551