Meta-Data Support for Data Transformations Using Microsoft Repository

Philip A. Bernstein Thomas Bergstraesser
Microsoft Corporation
[philbe, thomberg]@microsoft.com

Abstract

Data warehousing requires facilities for moving and transforming data and meta-data. This paper describes such facilities that are integrated with Microsoft SQL Server 7.0, particularly as they relate to meta-data and information models in its shared repository. It also discusses the use of XML to move meta-data between tools.

1 Introduction

To interpret data, users and applications need meta-data. Therefore, whenever data is moved or transformed, meta-data needs to be involved. Meta-data controls the movement by specifying such information as location, schedule, and source-to-target mappings. It needs to be moved along with the data so that further work can be done on the data after the move. Meta-data that describes the data movement activity is also useful to retrace data movement steps later on.

Like any kind of data, meta-data requires a persistent data store (i.e., a repository) in which to keep it [1]. It also requires an information model (i.e., schema) that describes the meta-data to be managed. The information model supports the integration of relevant tools and enables sharing of meta-data through the repository.

Today, the most common application environment in which data is moved and transformed is the data warehouse, an increasingly important part of an enterprise's information architecture. The amount and diversity of meta-data required in a data warehouse makes it a natural application of repository technology. Schema management for source and target data sources, aggregation information, scheduling of tasks, and system topology are examples of meta-data that needs to be managed.

Data warehousing scenarios are major applications of Microsoft[1] Repository, a meta-data manager that ships with Microsoft SQL Server 7.0. It includes a repository engine that sits on top of Microsoft SQL Server and supports both navigational access via object-oriented interfaces and SQL access to the underlying store. Microsoft Repository also includes the Open Information Model, a repository schema that covers the kinds of meta-data that are needed for data warehouse scenarios. One of the many users of the repository is SQL Server's Data Transformation Service (DTS), a tool that helps users design programs to populate a data warehouse.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

[1] ActiveX, Microsoft and Visual Basic are trademarks of Microsoft Corporation.

This paper describes facilities that support the movement and transformation of data and meta-data, particularly as they relate to the repository functions that support them. Section 2 describes data warehouse scenarios in more detail, and how DTS addresses them. Section 3 discusses the problem of sharing and exchanging meta-data via XML. Section 4 summarizes some future directions.

The first release of Microsoft Repository and the Open Information Model was in early 1997 in Visual Basic 5.0 [2]. The second release, which includes version and workspace management and a much expanded information model, shipped in Visual Studio 6.0 and SQL Server 7.0 in late 1998 [3]. The product also has a software development kit (SDK) that includes tools for designing and extending information models. Details are available at [7].

2 Populating a Data Warehouse

The most labor-intensive parts of creating a data warehouse are cleansing and mapping. Data from many sources is moved through a sometimes complex scrubbing process whose goals are to normalize the representation, filter out the desired information, and preserve information quality in the target system. To make this error prone process traceable, sophisticated systems maintain additional lineage information that links data to meta-data that describes how the data was processed.

Rules must be defined and implemented to identify and resolve inconsistent data formats, missing data, invalid data, semantic inconsistencies, and other sources of poor data quality. Often, data must be merged from multiple data sources and reconciled. Sometimes it is aggregated into summary information. Even a relatively simple data warehouse can require a hundred cleaning and mapping rules, with effort measured in person years.

Anecdotal evidence from consultants and users tells us that data scrubbing is over two-thirds of the work in setting up and managing a data warehouse. For this reason, there is a growing market for tools that simplify this work, a market that many software vendors are trying to fill. As explained earlier, these tools need meta-data: schema descriptions, libraries of transformation rules, descriptions of transformation scripts, etc.

Data warehousing is becoming one of the largest segments of the database business. SQL Server in its 7.0 release addresses this segment with an easy-to-use out-of-the-box data mart solution and a platform for scalable data warehouses. The main built-in components are an OLAP engine for accessing multi-dimensional data (Microsoft Decision Support Services) and a data movement tool (Data Transformation Service (DTS)). Meta-data management and extensibility come via Microsoft Repository and the Open Information Model.

DTS is a graphical tool for constructing and running transformation packages. Each package is a workflow that defines a transformation. It consists of a set of partially ordered steps, each of which invokes a task. Such a task can execute SQL statements, call an ActiveX script or external program, perform a bulk insert, send E-mail, transfer whole database objects, or call a data pump.

Data pump tasks are the main workhorse. They retrieve and store data through OLE DB, which defines a standard interface for access to heterogeneous data [4]. Each data pump task copies and/or transforms data from OLE DB data sources to OLE DB data targets. It consists of a set of transformations, each of which maps a set of input tables and columns to a set of output tables and columns. A transformation ranges from a simple copy operation to the invocation of a custom C++ program or ActiveX script.

The DTS Designer is a graphical tool for creating DTS packages (see Figure 1). Package definitions are meta-data and can be stored in a file, in SQL Server, or in Microsoft Repository. The latter is required if a user wants to track package versions, meta-data, and data lineage, all of which are maintained in the repository as fine-grained objects.

DTS's usage of the repository as a meta-data store is based on two main sub-areas of the Open Information Model (OIM): database models and data transformation models. The core database model (DBM) includes the concepts found in SQL data definitions and similar models of formatted data. Thus, it includes the concepts of database catalog, database schema (e.g., all objects owned by a user), column set (an abstraction of table and

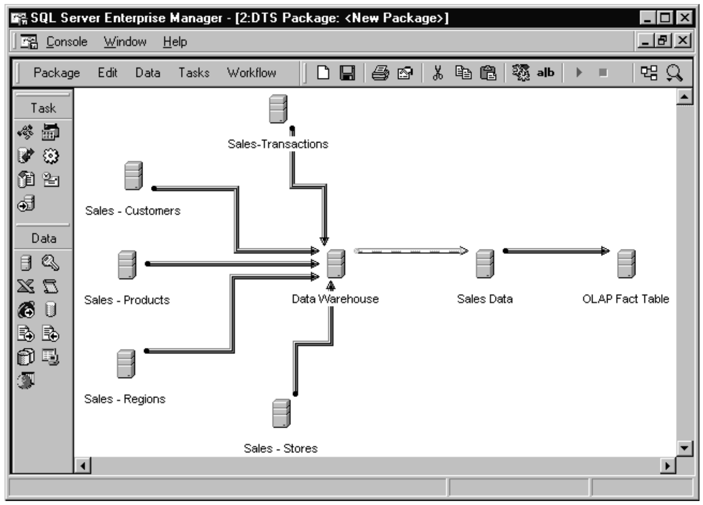

Figure 1: The DTS Designer

view), table, view, constraint, trigger, stored procedure, and column.

OIM is described in the Unified Modeling Language (UML) [5, 6, 8]. It also uses UML as its core model from which sub-models of OIM inherit. Inheriting UML concepts reduces the overall size of the model and promotes sharing between sub-models. For example, a subset of DBM is shown in Figure 2, where gray boxes denote UML core concepts from which DBM concepts inherit: catalog and schema inherit from UML package (UML's general container concept); column set, table, and view inherit from UML type; and column inherits from UML attribute (a structural feature of a UML type). Similarly, constraint inherits from UML constraint and trigger inherits from UML method (not shown in the figure). Some DBM concepts do not have natural analogs in UML, such as key and index, so they inherit from the very abstract UML concept of model element.

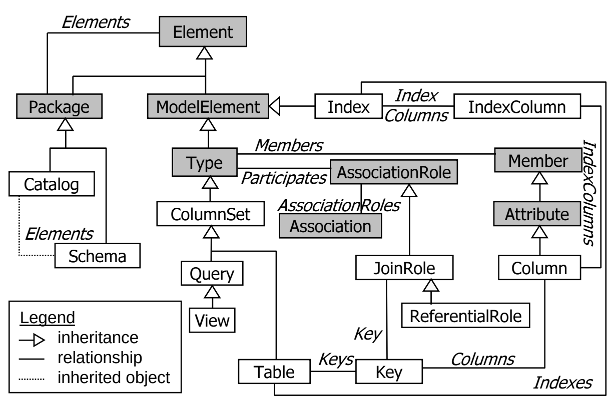

Figure 2: Database Information Model (DBM)

The database information model is populated with schema information by utilities included with SQL Server. The utilities use OLE DB and therefore can import schemas from any OLD DB data source, which include most popular database products (see http://www.microsoft.com/data/oledb/).

The transformation model (TFM) is another sub-model of OIM, which captures information about compound data transformation scripts (see Figure 3). An individual transformation (which is a specialization of UML method) can have relationships to the sources and targets of the transformation. Transformation semantics can be captured by constraints and by code-decode sets for table-driven mappings. Transformations can be grouped into a task (the unit of atomic execution), which is invoked by steps (the unit of scheduling) within a top-level package (which is a specialization of the generic UML package).

To support data lineage, package executions are also modeled. When a package is executed, DTS stores a description of that package execution in the repository. It also optionally tags each data warehouse row by the identifier of the package execution that produced it. This identifier enables traceability from the row in a data table to the repository's description of package execution that populated or updated it. From there, one can trace back to the package transformations, the sources and targets of the transformation, etc., to get a complete description of where that row came from.

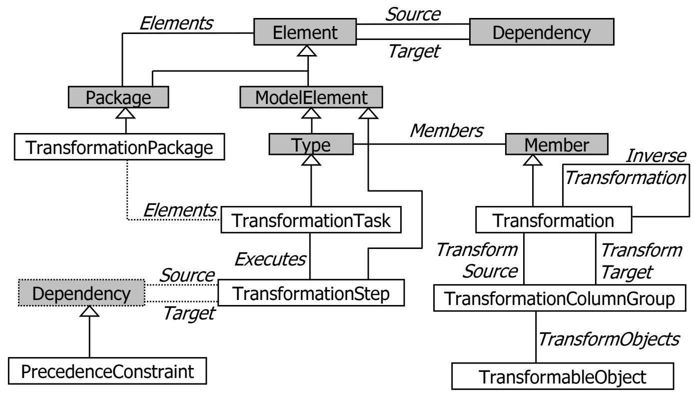

Figure 3: Transformation Model (TFM)

The DBM and DTS models were developed in collaboration with several vendors and reviewed by hundreds of others. Many of these vendors ship products that store or retrieve meta-data that are instances of these models. This enables them to ”upsell” to customers whose data warehouse problems outgrow the complexity or scale that Microsoft's framework can handle using the built-in tools.

OIM groups models into generic ones that cover the most common meta-data types in each domain and tool-specific ones that model private data structures. For example, the generic TFM model is specialized to the DTS model, which is a product-specific sub-model for the DTS tool. Primarily, the DTS model captures detailed properties of DTS's specializations of TFM script objects. For example, it includes properties of data pump tasks, transfer object tasks, and send mail tasks, all of which inherit from TFM transformation task, and of transformations, steps, and step executions.

Other sub-models of OIM that are relevant to data warehouse scenarios are the OLAP information model (OLP) for multi-dimensional schemas and Semantic Information Model (SIM) for semantics to drive a natural language query tool. The repository integrates these tools by importing a multi-dimensional schema as OLP objects from Microsoft OLAP Services and semantic information as SIM objects from Microsoft English Query.

3 Using XML

XML (extensible Markup Language) and its family of technologies are standards managed by the World Wide Web Council (W3C) for representing information as structured documents [9]. The structure of a document is expressed by a DTD (Document Type Definition) that consists of tag definitions and rules for how tagged elements can be nested. The basic idea behind XML is to embed tags into a character stream so that semantic structures can be easily recognized. Figure 4 shows an ASCII stream containing the address of a company, a DTD that captures the structure of an address definition, and the ASCII stream with embedded XML tags that conforms to the DTD.

Tagged structures like the one in Figure 4 can be embedded in Web pages to allow more efficient indexing and searching, an important application area for XML. However, there is an equally broad scope for XML in the area of data exchange and transformation. XML will become the standard way to define the structure of ASCII data streams in a world of heterogeneous systems. Agreeing on a simple DTD for the exchange allows systems to interpret ASCII data, to map it onto database tables, or to convert it into different formats.

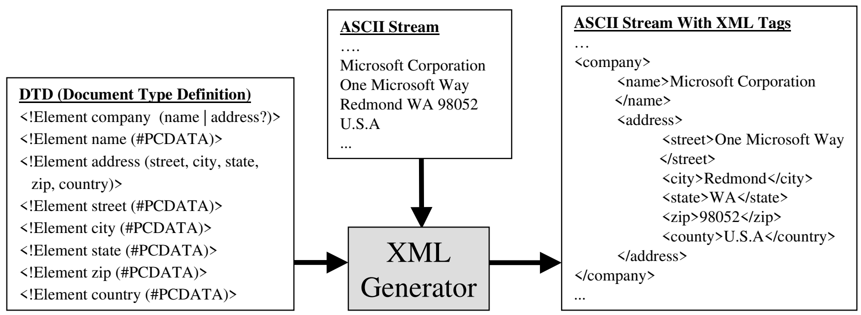

Figure 4: XML Example

It is therefore very natural to use XML to interchange meta-data – highly structured data described by an information model. To use XML to exchange meta-data, one needs to agree on models that describe the semantics and structure of meta-data to be exchanged. OIM is one such model. Because of its formal description in UML, we have been able to automatically map each sub-model of OIM into a DTD that guides the encoding of instances of that sub-model (see Figure 5). The mapping is a set of simple rules for generating DTD statements from the meta-data specifications.

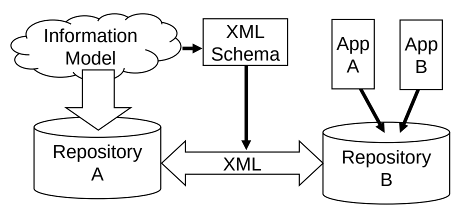

Figure 5: Using OIM XML to Transfer Meta-Data

An example of the OIM XML format is shown in Figure 6. Each element tag includes the OIM sub-model and type within the model. For example, dbm:Table describes the type Table within the database model, DBM. XML import and export programs are in the Microsoft Repository SDK [7]. Using this format, tool vendors can share information in the repository and therefore with DTS, with Microsoft-supplied importers, and each other.

Longer term, we expect unification of data and meta-data transfer via a rich set of efficient XML-based tools. This will strengthen the tool sets currently available for meta-data transfer. And it will simplify data transfer by making accompanying meta-data transfer more readily available.

4 Summary and Futures

Moving and transforming data with its meta-data in an enterprise environment is a complex, error phone, and expensive activity. Data, meta-data that describes data, and meta-data that guides the processing of data are all closely related. They cannot be treated separately if the data and transformation processes are to be manageable and comprehensible.

Microsoft and partner companies have developed the Open Information Model to provide an integration platform for data movement and processing tools. The next step is to extend the model to capture more of the semantics of transformations and to provide a more formal representation of business rules behind transformations.

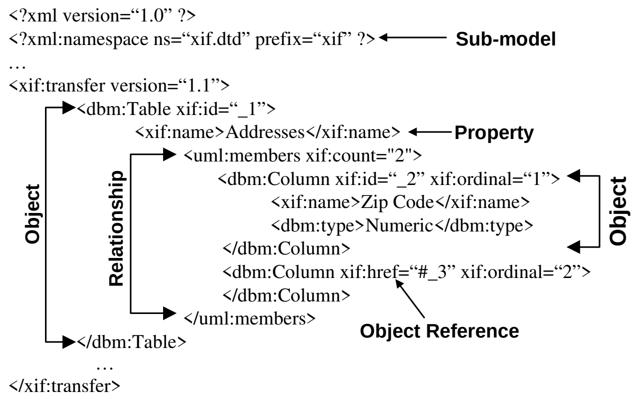

Figure 6: Transferring a Table Definition Using XML

This includes a more formal representation of semantic mappings between different models and their encoding, especially in XML.

To broaden the scope of the OIM as a description of meta-data about data, Microsoft and the Meta Data Coalition (MDC), a four-year-old industry consortium devoted to meta-data standards, have recently announced the forthcoming transfer of OIM to the MDC [7]. The MDC will maintain and evolve the OIM from its current COM orientation into a technology-independent and vendor-neutral meta-data standard. This will allow repository and tool vendors to use the OIM with their products independent of Microsoft and, combined with XML, move this meta-data between their products

References

[1] Bernstein, P.A. "Repositories and Object-Oriented Databases" In *Proceedings of BTW '97*, Springer, pp. 34-46 (1997). (Reprinted in *ACM SIGMOD Record 27, 1* (March 1998)).

[2] Bernstein, P.A., B. Harry, P.J. Sanders, D. Shutt, J. Zander, "The Microsoft Repository," *Proc. of 23rd Int'l Conf. on Very Large Data Bases*, Morgan Kaufmann Publishers, 1997, pp. 3-12.

[3] Bernstein, P.A., T. Bergstraesser, J. Carlson, S. Pal, P.J. Sanders, D. Shutt, "Microsoft Repository Version 2 and the Open Information Model," *Information Systems 24(2)*, 1999, to appear.

[4] Blakeley, J., "Data Access for the Masses through OLE DB," *Proc. 1996 ACM SIGMOD Conf.*, pp. 161-172.

[5] Booch, G., J. Rumbaugh, I. Jacobson, *The Unified Modeling Language User Guide*, Addison-Wesley, Reading, MA, 1998.

[6] Fowler, M., and K. Scott, *UML Distilled: Applying the Standard Object Modeling Language*. Addison-Wesley, Reading, MA, 1997.

[7] Microsoft Corp., Microsoft Repository web site, http://www.microsoft.com/repository

[8] Object Management Group, OMG Technical Library, at http://www.omg.org.

[9] World Wide Web Consortium, XML Language Specification and related documents, at http://www.w3c.org/TR/REC-xml.