December 2001

Managing Changes to Schema of Data Sources in a Data Warehouse

MANAGING CHANGES TO SCHEMA OF DATA SOURCES IN A DATA WAREHOUSE

G. Shankaranarayanan
Department of Management Information Systems
Boston University
gshankar@acs.bu.edu

Abstract

In the data warehouse environment, research has not adequately addressed the management of schema changes to source databases. Such changes have important implications. First the metadata used for managing the warehouse is affected by these changes and second, the applications that interact with the source databases may no longer be able to work with the changed schemas. In short, the data in the warehouse may not be consistent with the data and the structure of the source databases. In this paper we examine the implications for managing changes to the schema (or structure) of the operational databases in the context of data warehouses. First, we propose a framework that uses metadata to manage the impact of changes to the schema of source databases on data warehouse(s) that is dependent on these databases. We then examine the implications for managing and propagating these changes to the data transformation applications. We further describe how all of this fits in with the existing architecture of a data warehouse. To the best of our knowledge, this is the first research that examines the issue of schema changes in a data warehouse environment.

Introduction

An accepted definition of a data warehouse is that it is an integrated, subject-oriented, non-volatile, and time-variant collection of data with support for decision-making [Inmon 1998]. A data warehouse is populated with data from multiple operational and legacy data sources using applications that extract, cleanse, and integrate data from these different sources. We will refer to these applications that interface with the data sources as ETL (extraction, transformation, and loading) applications. Typically, these sources tend to be relational databases but no restrictions are imposed on the inclusion of other types of databases. In fact, Weiner et al state that the data sources for a warehouse are distributed, autonomous, and possibly heterogeneous [Weiner et al.1996]. A great deal of research addressing a variety of issues about the data warehouse has been completed. These issues include design and architecture of data warehouses, use of materialized views for better performance when populating data warehouses, case tools for designing data warehouses and many more.

An important issue not addressed in the literature is the issue of schema changes in the underlying operational databases. A database schema is defined as the description of the database [Elmasri and Navathe 1994] and as the overall design of the database [Silberschatz et al. 1997]. If the schema of any one (or more) operational database changes, it may impact the structure of the data warehouse. A change to the schema of one data source not only affects the schema of and the data in that data source but also affects the inter-relationships between the multiple data sources. As the structure (or schema) of the data warehouse is defined based on the structure of the individual data sources and based on the inter-relationships between these sources, a single schema change has the potential to significantly impact the structure of the warehouse. More importantly, the change to the schema of the data source affects the extraction programs and view-queries that interact with this schema. Further, the data in the warehouse may not be consistent with the data in the data sources when the schema of the data source changes. To date, structural changes to the underlying schema have not been adequately examined.

In this paper we examine the implications for managing changes to the schema (or structure) of the operational databases in the context of data warehouses. First, we propose a framework that uses metadata to manage the impact of changes to the schema of source databases on data warehouse(s) that is dependent on these databases. Metadata in the data warehouse environment includes the logical structure of the warehouse, logical structure of the source databases, and the transformation rules for extracting

data from the sources, integrating this data, and populating the data warehouse [Dolk 2000]. We then examine the implications for managing and propagating these changes to the data transformation applications. We further describe how all of this fits in with the existing architecture of a data warehouse. To the best of our knowledge, this is the first research that examines the issue of schema changes in a data warehouse environment.

The remainder of this paper is organized in the following manner. Relevant work on data warehouses, schema evolution, and metadata management is discussed in section 2. Section 3 presents the requirements for managing schema changes in the data warehouse environment and describes a framework addressing these requirements. We also list the metadata requirements for this framework and describe how the metadata is used to propagate the changes to the ETL applications. How this framework fits in the general architecture of a data warehouse is discussed in section 4. Section 5 presents the conclusions and suggests directions for further research.

Relevant Work

Three research areas relevant to this paper are managing changes in data warehouses and design of data warehouses, managing schema changes (or schema evolution), and modeling and managing metadata. We will each of these here.

A number of data warehouse vendors such as Oracle, IBM, and Microsoft have introduced systems that assist in building data warehouses. All of these help the designer define the data component of the warehouse by selecting data from data sources and then extract (transform, and load) the data into the data warehouse systematically. These systems use metadata extensively to support the design process besides using graphical interfaces to assist in navigation and selection of data elements. The designer must be familiar with the structure of the underlying data sources to use these tools efficiently. These systems do not deal with structural changes to one or more data sources based on which the warehouse is defined.

The use and maintenance of materialized views for data warehouses has received considerable attention. Materialized views are very applicable in the data warehouse environment [Gupta and Mumick 1998]. In fact a data warehouse is perceivable as a set (one or more) of materialized views where the base data resides in the data sources [Widom 1995]. A data warehouse (or the materialized views herein) is easier queried and refreshed by retrieving data from materialized views than from the source databases for several important reasons. These include the complex nature of views in a data warehouse, the extensive aggregation and integration of the data required by the warehouse and other reasons stated in [Widom 1995]. As each view is materialized and stored it must be consistent with the data source it is dependent on. Several methods have been suggested for maintaining *data* in these views consistent with the *data* in the associated data sources. The integrator in the warehouse along with the wrapper/monitor associated with each data source [Widom 1995], [Hammer et al. 1995] are critical components of the warehouse architecture that help maintain data consistency between the data source(s) and corresponding view(s). While it addresses the problem of adding a new data source, it does not address issues on maintaining the view consistent with the data source when the schema of an existing data source is changed.

Why are schema changes important? The schema of a database defines the structure to capture data stored in that database. Designing a database schema is an evolutionary activity that requires several iterations and even after it is completed the database schema continues to change over time [Zanilo et al. 1997]. Sjoberg's study revealed that schema changes are common during the design / implementation phase of the database *and* during the operation of the database [Sjoberg 1993]. Database reorganization has been the solution to accommodate schema changes. This is an expensive process and is hence performed infrequently. Research in schema evolution has suggested several methods for dynamically incorporating schema changes without significantly affecting the day-to-day operations of the database [Peters and Ozsu 1997] [Ram et al. 1999]. There are two key activities associated with schema evolution. First, application programs dependent on the schema are rendered obsolete when the schema is changed and hence must be managed so that they may continue to access the schema and associated data. Second, changes to the schema must be propagated to the data in the corresponding database to maintain the data consistent with its schema. Both these activities are important for managing schema changes in a data warehouse. The data transformation applications and the view-generating queries for a data source are defined based on the current schema of that source database. These may not work when the schema changes or worse, may retrieve incorrect data that eventually enters the warehouse. This impairs the quality of the data in the warehouse and significantly impacts decisions based on it. At the minimum, the administrators of the applications must be informed of the schema changes. Better yet, the changes needed must be suggested to the administrator who can then decide how best to implement them. Ideally, the simple changes must be automatically made and more complex changes made with the assistance of the administrator(s). The second activity in schema evolution, the propagation of schema changes to the data associated with it affects the quality and consistency of the metadata used to manage the warehouse.

Metadata is a key component in the warehouse environment [Dolk 2000]. The schema and evolution of metadata has been pointed out as being an important dimension affecting the quality of the warehouse [Jaarke 1999]. Metadata can be classified as technical and business metadata and each support a different class of users [Barquin and Edelstein 1997]. Technical metadata consists of the logical structure of the warehouse, logical structures of the data sources for the warehouse, and transformation rules embodied in the ETL applications [Dolk 2000]. The inter-relationships amongst the above three are also a part of it. The business metadata defines the data warehouse in business terms to enable each warehouse user to operate independently [Barquin and Edelstein 1997]. Data warehouse users must also be familiar with the transformation rules and data element (both source and warehouse) associations so that they can understand where the data came from and how the data was derived. Changes to the schema of one or more source databases can affect both types of metadata.

In the data warehouse environment, research has not addressed the management of schema changes to source databases. Such changes have important implications. First the metadata used for managing the warehouse is affected by these changes and second, the applications that interact with the source databases may no longer be able to work with the changed schemas. In short, the data in the warehouse may not be consistent with the data and the structure of the source databases. In the following section we propose a framework to maintain the data in the warehouse consistent with its sources.

Managing Schema Changes in Source Databases

Before describing the framework let us examine the requirements for managing the data warehouse consistent (structure and data) with its data sources. The following tasks are required to completely manage the data warehouse when changes occur in the schema of one or more source databases.

- Identify and manage schema changes at the source databases. This includes identifying changes to the schema, understanding the implications of each change, incorporating the changes to the schema, identifying changes that occur in other different source databases triggered by a change to the schema of one database (cascading changes), and propagating these changes to the data associated with each corresponding source database. While making changes to the database schema, we need to ensure that the schema is maintained in a correct state.

- The above step will result in changes to the source data elements, or more specifically, changes to the structure of the source objects that define the source data elements. We next need to identify the ETL applications affected by the changes to one or more of these source data elements, the transformation rules that apply to the source data elements. We also need to identify the dependencies defined between source data elements in one source and other data elements in other sources. These are the dependencies defined to help the integrator combine one or more data elements from one source with related data elements from other different sources.

- We then need to identify the warehouse data elements affected by changes to source data elements. We also need to know the set of ETL applications that extract these elements as well as the applicable set of transformation rules.

- The metadata in the warehouse corresponding to all the above must be updated to reflect the changes.

The above information will help the warehouse administrator understand the following:

- The warehouse elements that are affected by some change to the schema of the source database

- The ETL applications and transformation rules associated with these elements and affected by the change. The set of data elements that are extracted (cleansed, and transformed) by these applications along with the applicable set(s) of transformation rules which are also affected. The applications and the transformation rules may need to be modified/redefined depending on how the schema changes in the data source.

- The metadata in the warehouse that needs updating to reflect the changes and the information needed to update it.

All of the information (sources, objects, associations, applications, transformation rules, and dependency definitions) must be part of the metadata repository in a data warehouse. In typical data warehouse implementations, most of the metadata required is captured and is part of the data warehouse. We propose a simple classification (or layering) of this metadata as shown in figure 1. A schematic representation of the metadata objects and the relationships amongst these objects is shown in figure 2. The bi-directional arrows represent the inter-dependencies that exist between these objects. Consider each rectangle in figure 2 to represent an entity class that defines the metadata (i.e. one entity class represents the data sources, another the objects in the data

sources, and the third the data elements in the data sources). In terms of metadata, each bi-directional arrow corresponds to a set of ordered pairs ***<u, v>*** where ***'u'*** is an instance of the entity class at one end of this arrow, and ***'v'*** an instance of the entity class at the other end. A clear understanding of these inter-dependencies is necessary to understand how changes in the source schema impact other parts of the metadata in the warehouse. This layered representation also helps in explaining the methodology for propagating source schema changes to the other parts of the warehouse and in managing the warehouse consistent with the modified source schemas. This is described next.

The Source Schema Layer: This is the layer at which schema changes are initiated. These changes may be necessary because of any one or more of the reasons such as changes to the real world represented by that database, changes in user perceptions and requirements, as well as changes stemming from the application domain [Zanilo et al. 1997][Ram et al. 1999]. The source databases may be a heterogeneous set including data sources that are flat files and data sources that are relational/object-oriented/object-relational databases. A comprehensive framework for dynamically managing schema evolution in such an environment has been proposed in [Ram et al. 1999]. This includes incorporating schema changes into the source database, identifying and incorporating cascading changes in other databases in the heterogeneous set, and propagating the changes to the data in the corresponding underlying databases. The schema of the heterogeneous set of data sources, the global/federated schema, is represented using a semantic model. The schema is then mapped onto a graph-based representation. Changes are incorporated into the schema using a set of graph-based operators. The changed graph is converted back to an evolved global/federated schema. To propagate the changes to the set of databases, mapping information (metadata) is maintained which captures the associations between the schema objects and the database objects and between the database objects and corresponding databases. We propose the use of this framework to manage schema changes to the source schemas. The metadata used for managing these changes is part of the metadata in the warehouse, captured in the source schema layer shown in figure 1 and 2.

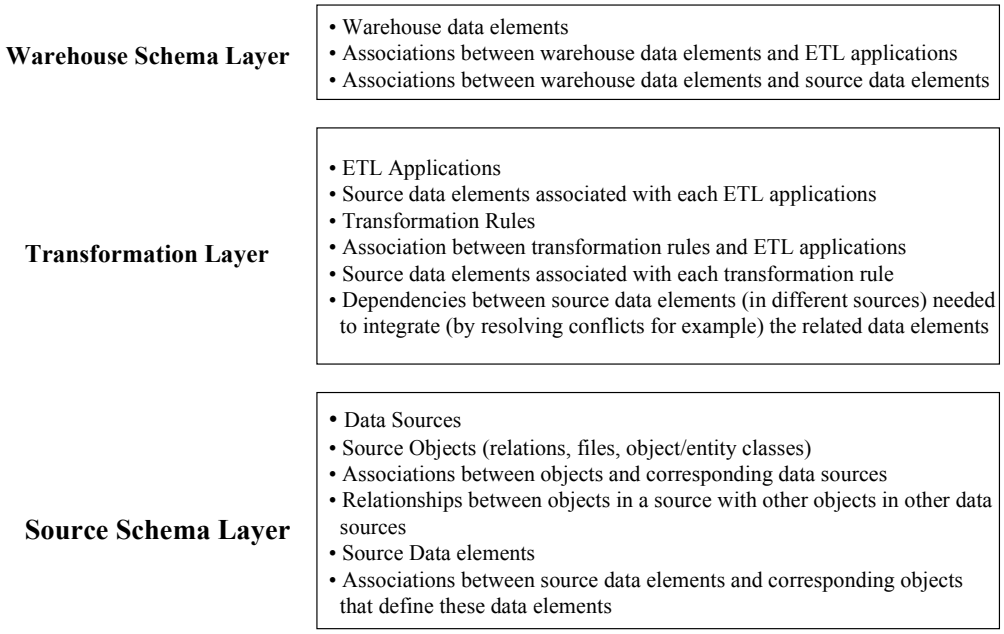

Figure 1. Layered View of the Metadata in a Data Warehouse

Transformation and Warehouse Schema Layers: The metadata maintained in these layers is about the data transformation applications, transformation rules, and the dependencies defined to merge related source data elements. It also includes the associations between each of the above and the source data elements that are associated with each. Besides these, metadata on the warehouse data elements is also captured in these layers. The objects (ETL applications, transformation rules, dependency definitions, and warehouse data elements) affect by a change to a source schema (and all of its cascading changes) can be identified using the metadata (inter-dependency ordered pairs) captured in this layer. The administrator can be informed about this set of affected objects and the users can be informed about the inconsistencies. Having identified the affected objects, how do we propagate corresponding changes to these objects? Can these changes be propagated automatically? We are currently looking into solution methods to address these questions.

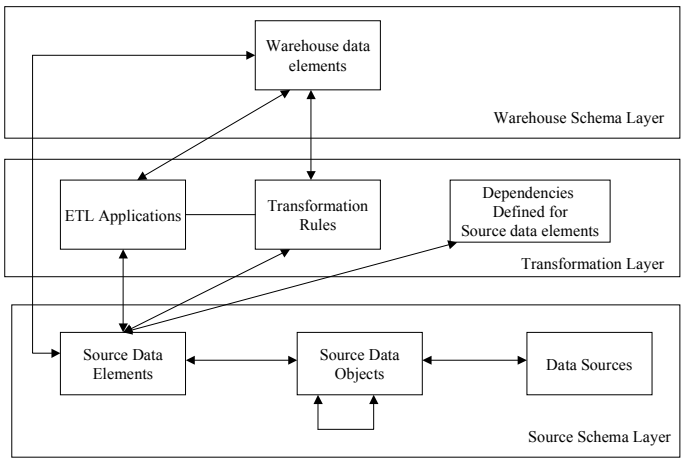

Figure 2. Schematic Representation of the Metadata and Interdependencies

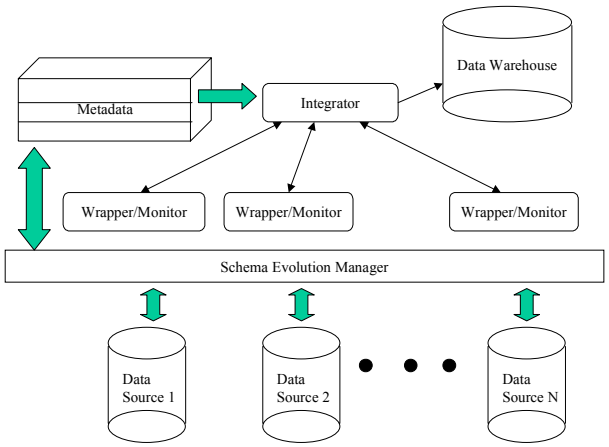

Figure 3. Schematic of a DW with Support for Schema Changes (adapted from Widom 1995)

Architecture to Support Schema Changes

How does all of this tie-in with the overall architecture of a data warehouse? Figure 3 shows the general architecture of a data warehouse adapted from [Widom 1995]. This figure is modified to show the added components for managing schema evolution in the data warehouse. The *Schema Evolution Manager* module manages schema evolution in the source databases. If the databases were completely independent and autonomous, each database would have its own manager module. This module updates and retrieves metadata that resides in the *source schema layer* (in figures 1 and 2). It is also responsible for propagating the schema changes to the data in the source databases. The administrator is informed about the affected objects that need modifications such as the ETL application(s) and transformation rules. The administrator then needs to modify the applications/rules and the metadata reflecting the changes must be updated in the metadata repository. The wrappers/monitors detect the changes to data in the source databases and trigger the process to regenerate the data in the materialized views. This maintains data in the views consistent with the changed data in the data sources. The integrator, using the metadata, plays an important role in the process. As views are queried to populate the warehouse, the data in the warehouse is consistent with the data in the source databases.

Conclusions and Directions for Future Work

Research in the data warehouse has not addressed managing the warehouse environment when the schema of the source databases change. The metadata used for managing the data warehouse and the data transformation applications that populate the warehouse are both affected by such schema changes. In this paper we propose a framework for a data warehouse environment to manage schema changes in source databases. The contributions of this paper are as follows: (1) it describes a mechanism that utilizes the existing metadata and exploits this to the fullest extent for managing the changes. (2) The method can be incorporated into existing warehouse architecture without significant changes to it. (3) The schema evolution mechanisms are linked to but maintained as a logically distinct unit that only interacts with the metadata in the warehouse and does not affect the rest of the warehouse. This facilitates implementation. (4) To the best of our knowledge, it is the first research that addresses schema (structural) changes in source databases of a data warehouse.

An important issue not completely addressed is the propagation of changes to the metadata. As described, the administrator(s) needs to make the changes to the metadata and to modify the affected applications. We need to examine whether this can be automated and if so how best to do it. We are currently examining these issues in the context of a large data warehouse that supports decision-making in a large financial institution.

References

Barquin, R. & Edelstein, H. (Editors) – *Building, Using, and Managing the Data Warehouse,* Prentice Hall PTR, Upper Saddle River, NJ, 1997

Dolk, Daniel R. – Integrated Model Management in the Data Warehouse Era, *European Journal of Operational Research,* 2000, vol.1, no.22, pp. 199-218

Elmasri, R. and Navathe, S. - *Fundamentals of Database Systems*, Benjamin Cummings Publishing Company, Inc., Redwood City, CA, 1994

Gupta, A & Mumick, I. (Editors) – *Materialized Views: Techniques, Implementations, and Applications,* MIT Press, Cambridge MA, 1998.

Hammer, J., Garcia-Molina, H., Widom, J., Labio, W.J., Zhuge, Y. – The Stanford Data Warehousing Project, *IEEE Data Engineering Bulletin,* June 1995.

Inmon, W. H. – *Building the Data Warehouse*, Second Edition, John Wiley and Sons, March 1996

Jarke, M., Jeusfeld, M., Quix, C., & Vassiliadis, P. – Architecture and Quality in Data Warehouses: An Extended Repository Approach, *Information Systems,* 1999, Vol. 24, No. 3, pp. 229-253

Peters, R. J., & Ozsu, T. - An Axiomatic Model of Dynamic Schema Evolution in Objectbase Systems, *ACM Transactions on Database Systems,* vol. 22, no. 1, pp. 75-114, March 1997

Ram, S., Shankaranarayan, G., & Hartman, B. - *Automating Dynamic Schema Evolution in a Heterogeneous Database Environment,* Working Paper #99-15, Information Systems Department, Boston University, 1999.

Silberschatz, A., Korth, H. F., & Sudharshan, S. – Database System Concepts, McGraw Hill Inc., 1997

Sjoberg, D - *Quantifying Schema Evolution"*, Information and Software Technology, vol. 35, no. 1, pp. 35-44, 1993

Weiner, J.L., Gupta, H., Labio W.J., Zhuge, Y., Garcia-Molina, H., & Widom, J. – A System Prototype for Warehouse View Maintenance, *Proceedings of the ACM Workshop on Materialized Views: Techniques and Applications,* June 1996, Montreal, Canada, pp. 26-33

Widom, Jennifer – Research Problems in Data Warehousing, in *Proceedings of the 4th International Conference on Information and Knowledge Management* (CIKM'95), November 1995, pp. 25-30

Zanilo, S. Ceri, C. Faloutsos, R. T. Snodgrass, V. Subrahmanian, & R. Zicari - "*Advanced Database Systems"*, 1997, Morgan Kaufmann Publishers Inc., San Francisco, California, Chapter 17, pp. 437-438.