X-Time: Schema Versioning and Cross-Version Querying in Data Warehouses

Stefano Rizzi Matteo Golfarelli
DEIS, University of Bologna
Viale Risorgimento 2, 40136 Bologna, Italy
{*srizzi* | *mgolfarelli*}*@deis.unibo.it*

Abstract

In this demo we present X-Time, a prototype for managing schema versioning in relational data warehouses, specifically oriented to support the formulation of cross-version queries, i.e., queries whose temporal horizon spans multiple versions. The key issue to increase querying flexibility is the introduction of augmented schemata that properly extend previous schema versions.

1 Introduction

The problem of schema versioning in data warehouses (DWs) is gaining more and more attention by the research and the industrial communities. In fact, business requirements evolve quickly, so that new dimensions, properties, and measures may become necessary, while others may become obsolete. On the other hand, accountability requires access to past schema versions, thus there is a need for an approach to *schema versioning* where past schema definitions are retained, so that all data may be accessed both retrospectively and prospectively through user-definable schema interfaces. Though some approaches have been devised in the literature (e.g., [1, 2]), on the commercial side the problem has only marginally been addressed and no tool offers a comprehensive solution.

In this demonstration we present X-Time, a prototype for managing schema versioning in relational DWs, specifically oriented to support the formulation of *cross-version queries*, i.e., queries whose temporal horizon spans multiple versions. X-Time is based on the approach proposed in [2].

2 Approach Overview

We call a *version* a schema that reflects the business requirements during a given time interval, called its *validity*. A version is populated with the events occurring during its validity and can be queried by the user. In order to capture the core of the multidimensional model we represent each version by a *schema graph*, i.e. a directed graph S, rooted in the fact of interest, whose nodes represent attributes (dimensions, dimension properties, and measures) and whose arcs represent simple functional dependencies (FDs) between them [2].

An example of the schema graph S_0 for a fact modeling the shipments of parts to customers all over the world is shown in Figure 1. Nodes Date, Part, Customer, Deal, and ShipFrom are the fact dimensions, QtyShipped and ShippingCosts are its measures. Each dimension is the root for a hierarchy of properties linked by simple FDs.

A new version is the result of a *schema transaction*, i.e., a sequence of modification operations. We provide four visual modification operations: add a new attribute, delete an existing attribute, add an arc (i.e., an FD) between two existing attributes, and remove an existing arc [2]. Intermediate schemata obtained after applying single schema modifications are invisible for querying purposes.

In our example, at time t_1 the shipment schema graph undergoes a major revision aimed at better fulfilling some changing business requirements. For instance, in the new version S_1 the temporal granularity is changed from Date to Month, a measure Discount is added, a classification into subcategories is inserted into the part hierarchy, and the ShipMode dimension is substituted with a ShipFrom dimension. Within a system not supporting versioning, at t_1 all previous data would be destructively migrated to the new version. In X-Time data migration is non-destructive, so both S_0 and S_1 are still available for querying together with the data recorded during their validity times.

Augmentation is the technique we adopt to allow designers to increase flexibility in cross-version querying. When creating a new version S_n at time t, the designer may create *augmented schemata* that extend previous versions to reflect the current schema extension. Given the differences between S_n and each previous version S_i, the system proposes to the designer a set of possible *augmentation actions* on the data recorded under S_i; these actions may entail checking for additional constraints or inserting new data. The set of

1-4244-0803-2/07/$20.00 ©2007 IEEE

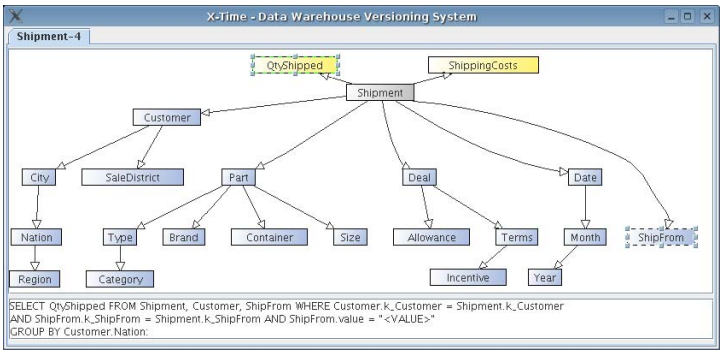

Figure 1. Schema graph for the shipment fact

actions the designer decides to undertake on data leads to defining an augmented schema S_i^{AUG} that will be used instead of S_i, transparently to the final user, to answer queries spanning the validity interval of S_i. E.g., in the shipments case, when measure Discount is added to the shipment fact, the designer could decide to provide values for Discount for the events recorded under S_0 by deriving an estimate based on the values of the other measures: e.g., if the discount applied depends on the shipped quantity, its values for past events may be easily estimated from Qty shipped.

As concerns cross-version querying, let q be an OLAP query and T_q be the temporal interval covered by the data to be analyzed in q. The *formulation context* for q is defined as the schema graph corresponding to the largest schema that retains its validity throughout T_q. If T_q spans a single version v, the formulation context is obviously the schema graph describing v. When T_q spans two or more versions, its formulation context is the intersection of all the augmented schema graphs whose validity overlaps with T_q; the intersection between two versions S_i and S_j, formally defined in [2], intuitively includes only the attributes belonging to both S_i and S_j, as well as their common FDs. In the light of this, the query formulation cycle can be described as follows: (1) the user defines the temporal interval that next query q will span; (2) the formulation context for q is visualized in the form of a schema graph; (3) the user visually formulates q on the schema graph as a GPSJ query; (4) the corresponding SQL query is executed on each augmented schema involved; (5) the result, union of the partial results, is displayed in tabular form. Steps 2-4 are illustrated in Figure 1.

3 X-Time Architecture and Demonstration

X-Time was developed in Java relying on the DOM4J library to manage the XML meta-data, on JDBC to connect with the RDBMS hosting the data, on the JGraph library to visualize graphs. A *multi-pool* approach is adopted for physically implementing version histories, i.e., each version is associated with a "private" extensional data pool. The overall architecture is sketched in Figure 2. To the best of our knowledge, the only other research prototype for DW versioning is the one presented in [3], that while effectively supporting cross-version queries, does not provide any augmentation technique.

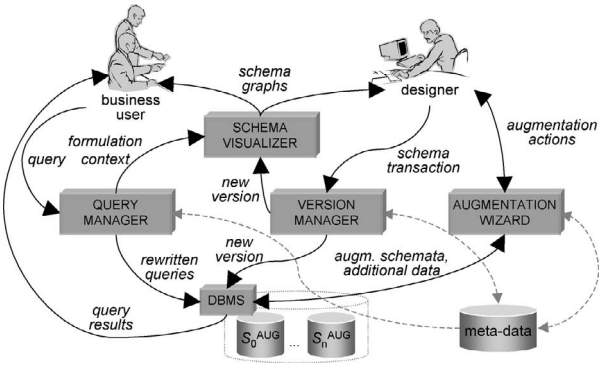

Figure 2. X-Time architecture

The demonstration will focus on showing how versions are managed, how the designer is supported in defining and populating augmented schemata, and how the business user is enabled to visually formulate cross-version queries.

References

[1] M. Blaschka. *FIESTA - A framework for schema evolution in multidimensional databases*. PhD thesis, Technische Universitat Munchen, Germany, 2000.

[2] M. Golfarelli, J. Lechtenbörger, S. Rizzi, and G. Vossen. Schema versioning in data warehouses: Enabling cross-version querying via schema augmentation. *Data and Knowledge Engineering*, 59(2):435–459, 2006.

[3] R. Wrembel and T. Morzy. Managing and querying versions of multiversion data warehouse. In *Proc. EDBT*, pages 1121–1124, Munich, Germany, 2006.