Data Warehouse Evolution Framework

© Darja Solodovnikova

Department of Computer Science, University of Latvia
darja.solodovnikova@lu.lv

Abstract

In this paper a data warehouse framework that supports data warehouse evolution is presented. The framework is able to handle not only changes in data sources, but also direct changes in a data warehouse schema. In the framework the data warehouse versions are supported in the development environment as well as in reports in the user environment.

1 Introduction

Data warehouses integrate information from various distributed and autonomous data sources that can change in the course of time. Therefore a data warehouse has to be adaptable to any changes that can happen in underlying data sources. Besides business requirements often change at the client level. That can cause changes to the data warehouse model. All these changes in data sources or business requirements can invalidate existing schemata and data extraction, transformation and loading (ETL) processes of the data warehouse. This is why these changes need to be handled properly. In many cases the existing data warehouse can be adapted to changes.

Simple adaptation of the data warehouse schema can cause a loss of history when some previously available data structures are deleted. To solve problems of history losses, it is necessary to keep data warehouse versions. Schema versioning means that a change in the data warehouse schema creates a new schema version that is assigned a timestamp or other user-defined identifier.

In this paper a data warehouse framework is discussed. The framework supports data warehouse schema evolution that can happen for different reasons, including cases when schemas of data sources are changed. The supported changes are insertion, deletion and renaming of a source relation, insertion, deletion, renaming and change of a type of a source relation attribute. The proposed framework not only automates the evolution of a data warehouse schema or creation of a new version, but also allows to adapt ETL processes and existing reports on a data warehouse schema.

The rest of this paper is organized as follows. In Section 2 the motivating example that demonstrates the necessity to adapt data warehouse is given. In Section 3 the related work is presented. In Section 4 the proposed data warehouse evolution framework that supports changes in data sources and propagates them to the data warehouse is discussed. We conclude with directions for future work in Section 5.

Proceedings of the Spring Young Researcher's Colloquium On Database and Information Systems SYRCoDIS, Moscow, Russia, 2007

2 Motivating Example

Data warehouse schema evolves frequently when business requirements are changed or extended or a schema is adapted after changes in data sources.

As an example, let us consider a data warehouse that stores information about students' activities in a learning management system (LMS). This data warehouse contained one fact table with measures: hits and time, which records the duration of students' activity. These measures could be analyzed by the used course, a tool in this course and time, when the activity occurred. The activity of all students was summarized.

During the operation of the aforementioned data warehouse the users complained that the information available in it is insufficient because the existing scheme did not satisfy the desirable granularity. Besides, it was decided to store also data about the activities of lecturers of courses. Therefore, the new dimensions that describe the particular user and his or her role in a course were created.

To solve the evolution problems, the administrator had to create a new data warehouse schema and ETL processes. It required much time and resources, but finally the second data warehouse version was created. But still there was an open question how to automate the data warehouse adaptation and how users can work with two data warehouse versions, because traditional reporting tools and query languages do not support the concurrent work with many schema versions.

3 Related work

In the literature there are various solutions for the data warehouse evolution problems, which are the data warehouse adaptation after the changes in source data and schemata as well as business requirements. In [1] the primitive evolution operations that occur over the data warehouse schema are defined. The necessary adaptation activities of the data warehouse schema and instances are formally specified for each operation. This paper only considers changes raised by alterations of business requirements.

In [15] the existing techniques for schema evolution are integrated in the new quality-oriented framework. The author proposes schema evolution operations (e.g. attribute insertion or deletion) and describes quality factors that they affect.

In [10] the evolution operations that change the data warehouse schema are considered. For each operation, the formal semantics of the changes for star and snowflake schemata are given.

The above mentioned papers do not address the problems of the data warehouse adaptation after changes in data sources. One of the approaches for solving these problems is adaptation of the data warehouse schema and ETL processes. In [11] the author proposes the solution, which is based on the transformation of schemata of data sources by transformation primitives. When the data source schema changes the information in the transformation specification is used to adapt the data warehouse schema and ETL processes.

In [19] the authors consider mapping adaptation after the changes in data source schemata. Here mapping specifies how data instances of one schema correspond to data instances of another. The authors propose the algorithm that detects mappings affected by changes in data sources and generates rewritings that are consistent with the semantics of the mapped schemata.

In the paper [12] both as view approach for data integration is proposed. The basis of this approach is schema transformation primitives, which specify how the global schema is obtained from local schemata. From these transformations it is also possible to infer, how local schemata can be obtained from the global schema. The evolution of local and global schemata is also discussed. For schema adaptation, similarly as in the previously mentioned papers, the information about transformations is used.

In many papers [3,6,16] a data warehouse is defined as a set of materialized views over data sources. These papers study the problems of how to rewrite a view definition and adapt view extent after changes in source data and schemata. In [16] the authors study reasons for schema changes and possible data warehouse adaptation issues for dynamic sources. The framework of the evolvable view environment is presented, which adapts view definition and extent after changes in data sources.

Several authors [7,8,20] propose the data warehouse schema versioning approach to solve the problems of schema evolution. In [8] the authors propose to store augmented schemata together with schema versions. When schema changes occur, firstly the new schema version is produced and then, for the previous versions, augmented schemata are created and populated with data.

In [7] the metadata model that supports schema versioning for data warehouses is introduced. Metadata management solutions in a multiversion data warehouse are also proposed in [20], where one of the discussed issues is metadata support for detection of changes in sources and propagation of them to the designated data warehouse version. Issues related to queries to a multiversion data warehouse are considered in [14].

In [17] the definition of a multidimensional schema that supports schema versioning is given. This definition is very similar to the one given in [2], the difference is that the first one supports versioning. The version evolution operations that result in versioning of the data warehouse schema are formalized.

Structural and content changes in dimensions of a data warehouse are discussed in [9]. A multidimensional model and its' instances are defined. Dimension structural and instance update operators are formally specified and their effect is studied over materialized views over dimension levels.

In [13] a temporal multidimensional data model is proposed, which allows to track history of dimension updates. In the model elements of dimension schemas and/or instances are assigned the timestamp when they were the part of a dimension. The query language TOLAP is also presented that supports queries over the proposed data model both over data and metadata.

In [4,5] a method to support data and structure versions of dimensions is proposed. The method allows tracking history and comparing data, using temporal modes of presentation that is data mapping into the particular structure version. The authors define the conceptual model based on the multiversion fact table.

The above mentioned papers consider only one kind of evolution problems, for example, changes in a schema of a data warehouse raised by evolving business requirements, adaptation of a data warehouse after changes in data sources or data warehouse versioning and querying multiversion data warehouse. In our approach we propose the framework that is able to solve all these kinds of evolution problems.

4 Data Warehouse Evolution Framework

To support the data warehouse adaptation after changes in source schemata and versioning, we propose the data warehouse framework depicted in Figure 1.

4.1 Components of the Framework

The framework is composed of the development environment and user environment. In the development environment the data warehouse metadata repository and other components, which will be described later, are located and ETL processes and change processing is conducted. In the user environment reports on one or several data warehouse versions are defined and executed.

The basic element of the development environment is the adaptation component that processes changes in relations and attributes of source schemata, identifies the potential changes in a data warehouse and possible new versions, adapts a data warehouse schema or creates a new version according to the choice of the data warehouse administrator, creates the necessary version metadata and adapts ETL processes. To realize this functionality, the adaptation component uses data from the metadata repository.

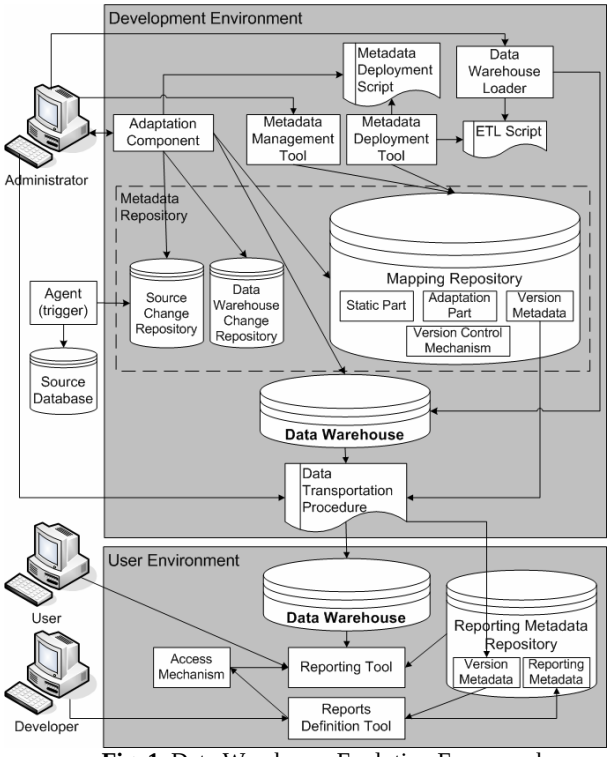

Fig. 1. Data Warehouse Evolution Framework

The metadata management tool that incorporates the graphical user interface client tool is used by administrator or developer to design a data warehouse schema and specify ETL processes. The metadata management tool maintains the static part of the mapping repository of the metadata repository, where the metadata of the last data warehouse version and mappings, which define the logics of ETL processes, are stored. In addition the mapping repository includes also three another parts. In the adaptation part the adaptation component stores information about dependencies of data warehouse elements and source elements used for a data warehouse adaptation. The version control mechanism defines rules for necessity and logics of creation of a new data warehouse version. The version metadata stores information about data warehouse versions, which is necessary for definition and execution of reports, including links between different versions. The detailed description of the metadata repository as well as the operation of the adaptation component with adaptation options for each change and presentation of the implemented prototype of the adaptation framework is found in the paper [18].

The metadata repository includes also the data warehouse change repository, which accumulates the potential changes of a data warehouse schema and version creation options. The administrator chooses the most suitable options that are applied. Special agents are incorporated into data sources. These agents track changes in source schemata and accumulate them in the source change repository.

ETL processes are generated by the metadata deployment tool that uses the metadata from the static part of the mapping repository. The data warehouse loader executes generated ETL scripts. The data transportation procedure transfers data warehouse data from the development environment into the user environment and version metadata into the reporting metadata repository.

In addition to the data warehouse, in the user environment there is also the reporting metadata repository that contains the version metadata, which are transferred from the mapping repository of the development environment, and the reporting metadata, which are created by a data warehouse developer by the reports definition tool and are used by the reporting tool for generation of reports. Data warehouse users work with the reporting tool that allows to define ad-hoc queries, display reports as tables and graphs and analyze data using hierarchies.

Using links between data warehouse versions in the metadata repository, the reporting tool can run queries on multiple data warehouse schema versions or one version. In case of many versions, a user can choose which version will be used to display results of a query.

In the user environment an access mechanism is also implemented. It is the metadata that defines which reports can be used by the particular user. These metadata are used by the reporting tool. The access mechanism is set by the developer by the reports definition tool.

4.2 Framework Operation

The proposed framework is able to handle source changes that can influence a data warehouse as well as other changes of a data warehouse schema. If the data warehouse schema is changed by the administrator then all changes are conducted by the metadata management tool, which allows to create a new data warehouse version or alter an old version. The metadata of ETL processes in the mapping repository are adapted according to a new data warehouse version.

The source changes are processed before the execution of ETL processes in the development environments. Initially the adaptation component analyzes the changes in the source change repository and detects changes that affect a data warehouse schema and ETL processes. The adaptation component processes these changes using data from the adaptation part of the mapping repository and the version control mechanism and, for each change, generates solutions that create a new data warehouse version or adapt the data warehouse schema and ETL processes. The administrator is informed about all changes and their adaptation and version options. The administrator chooses the most suitable solutions that must be implemented according to the business requirements.

If the administrator decides to create a new data warehouse version, the adaptation component changes the version metadata in the mapping repository to reflect the new data warehouse version. If the administrator chooses to conduct adaptation without creation of a new version, the adaptation component does not need to change the version metadata.

The adaptation component adjusts the metadata of the data source, data warehouse and specification of

ETL processes in the adaptation part of the mapping repository according to the chosen solutions. The adaptation component also creates a new data warehouse version or adapts the existing data warehouse directly in the database. The adaptation component generates the metadata deployment script, which is executed by the metadata deployment tool that generates the executable ETL process script. The ETL process is executed by the data warehouse loader.

When a data warehouse schema is changed, the reporting and version metadata, which describe a data warehouse, in the reporting metadata repository in the user environment become inadequate to a changed schema or new schema version and reports on a data warehouse can not run any more. Therefore, during the data transfer from the development environment into the user environment the data transportation procedure updates also the reporting and version metadata to reflect the new data warehouse schema.

5 Conclusions and Future Work

We proposed the data warehouse evolution framework. This framework was developed by expanding the data warehouse adaptation framework [18], which was previously designed and implemented. The adaptation framework could automatically detect changes in schemata of data sources and adapt a data warehouse schema and ETL processes, according to the administrator's decision.

Unlike the adaptation framework, the evolution framework is able to handle not only changes in data sources, but also direct changes in a data warehouse schema. The second important difference is the fact that in the evolution framework the data warehouse versions are supported in the development environment as well as in reports in the user environment.

The proposed framework differs from other solutions of data warehouse evolution problems presented in the literature by the fact that it supports many evolution problems at once, not just one problem.

Further it is planned to transform the developed prototype of the adaptation framework to conform to the aforementioned evolution framework.

References

[1] M. Blaschka. FIESTA: A Framework for Schema Evolution in Multidimensional Databases, PhD thesis, Technische Universitat Munchen, Germany (2000)

[2] M. Blaschka, C. Sapia, G. Hofling. On Schema Evolution in Multidimensional Databases. In *LNCS*, Vol. 1676, Springer-Verlag (1999) 153–164

[3] Z. Bellahsene. Schema Evolution in Data Warehouses. In *Knowledge and Information Systems*, 4, Springer-Verlag, (2002) 283-304

[4] M. Body, M. Miquel, Y. Bedard, A. Tchounikine. A Multidimensional and Multiversion Structure for OLAP Applications. In *Proc. of the 5th ACM Intl. Workshop DOLAP*, McLean, Virginia (2002) 1-6

[5] M. Body, M. Miquel, Y. Bedard, A. Tchounikine. Handling Evolutions in Multidimensional Structures. In *Proc. of the 19th Intl. Conference on Data Engineering*, Bangalore, India (2003)

[6] S. Chen, X. Zhang, E.A. Rundensteiner. A Compensation-based Approach for Materialized View Maintenance in Distributed Environments. In *Computer Science Technical Report*, Worcester Polytechnic Institute, Worcester, MA, USA (2004)

[7] J. Eder, C. Koncilia, T. Morzy. The COMET Metamodel for Temporal Data Warehouses. In *LNCS*, Vol. 2348, Springer-Verlag, (2002) 83–99

[8] M. Golfarelli, J. Lechtenbörger, S. Rizzi, G. Vossen. Schema Versioning in Data Warehouses. In *LNCS*, Vol. 3289, (2004) 415–428

[9] C.A. Hurtado, A. O. Mendelzon, A.A. Vaisman. Maintaining Data Cubes under Dimension Updates. In *Proc. of the 15th Intl. Conference on Data Engineering*, Washington, DC (1999) 346-357

[10] C.E. Kaas, T.B. Pedersen, B.D. Rasmussen. Schema Evolution for Stars and Snowflakes. In *Proc. of the 6th Intl. Conference ICEIS*, Porto, Portugal (2004) 425-433

[11] A. Marotta. Data Warehouse Design and Maintenance through Schema Transformations, Master thesis, Universidad de la República Uruguay, (2000).

[12] P. McBrien, A. Poulovassilis. Data Integration by Bi-Directional Schema Transformation Rules. In *Proc. of the 19th Intl. Conference ICDE*, Bangalore, India (2003) 227-238

[13] A.O. Mendelzon, A.A. Vaisman. Temporal Queries in OLAP. In *Proc. of the 26th Intl. Conference VLDB*, Cairo (2000) 242-253

[14] T. Morzy, R. Wrembel. On Querying Versions of Multiversion Data Warehouse. In *Proc. of the 7th ACM Intl. Workshop DOLAP*, Washington, DC, USA (2004) 92–101

[15] C. Quix. Repository Support for Data Warehouse Evolution. In *Proc. of the Intl. Workshop DMDW*, Heidelberg, Germany (1999)

[16] E.A. Rundensteiner, A. Koeller, X. Zhang. Maintaining Data Warehouses over Changing Information Sources. In *Communications of the ACM*, Vol. 43, New York, NY, USA (2000) 57-62

[17] M.K. Shahzad, J.A. Nasir, M.A. Pasha. CEV-DW: Creation and Evolution of Versions in Data Warehouse. In *Asian Journal of Information Technology*, 4(10) (2005) 910-917

[18] D. Solodovnikova, L. Niedrite. Data Warehouse Adaptation after the Changes in Source Schemata. In *Proc. of the 7th Intl. Baltic Conference on Databases and Information Systems*, Vilnius, Lithuania (2006) 52-63

[19] A.A. Vaisman, A.O. Mendelzon, W. Ruaro, S.G. Cymerman. Supporting Dimension Updates in an OLAP Server. In *LNCS*, Vol. 2348, Springer-Verlag (2002) 67-82

[20] R. Wrembel, B. Bebel. Metadata Management in a Multiversion Data Warehouse. In *LNCS*, Vol. 3761, Springer-Verlag (2005) 1347-1364