65

Metadata to Support Data Warehouse Evolution

Darja Solodovnikova

Abstract

The focus of this chapter is metadata necessary to support data warehouse evolution. We present the data warehouse framework that is able to track evolution process and adapt data warehouse schemata and data extraction, transformation, and loading (ETL) processes. We discuss the significant part of the framework, the metadata repository that stores information about the data warehouse, logical and physical schemata and their versions. We propose the physical implementation of multiversion data warehouse in a relational DBMS. For each modification of a data warehouse schema, we outline the changes that need to be made to the repository metadata and in the database.

Keywords Data warehouse · Schema evolution · Changes · Metadata · Repository

1. Introduction

Data warehouses often evolve due to changes in data sources and business requirements. These changes influence existing schemata and data extraction, transformation, and loading (ETL) processes of the data warehouse. This is why these changes need to be handled properly. Often the existing schema and ETL processes can be adapted to changes automatically or with minimal manual work.

Simple adaptation of the data warehouse schema can cause a loss of history. Therefore, it is necessary to keep versions of data warehouse schemata. According to [7], 'schema version is a schema that reflects the business requirements during a given time interval, called its validity, that starts upon schema creation and extends until the next version is created.' In this chapter metadata of the multiversion data warehouse are discussed. The metadata include physical and logical information about data warehouse schema versions. The impact of changes of data warehouse schema on metadata is also discussed.

The rest of this chapter is organized as follows. In Section 2 the related work is presented. In Section 3 the proposed data warehouse framework is outlined. The main contribution of this chapter is presented in Sections 4 and 5, where the metadata model for a multiversion data warehouse is described and changes of the data warehouse schema that are supported by the proposed framework and their results are discussed. We conclude with directions for future work in Section 6.

2. 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.

Changes in dimensions of a data warehouse are discussed in [8]. Dimension structural and instance update operators are formally specified and their effect is studied over materialized views over dimension levels.

Darja Solodovnikova • University of Latvia, Raina bulv. 19, Riga, Latvia.

G.A. Papadopoulos et al. (eds.), *Information Systems Development*, DOI 10.1007/b137171_65,

In [1, 9] the primitive evolution operations that occur over the data warehouse schema are defined. In [9] the semantics of the changes for star-and snowflake schemata are given for each operation.

The above-mentioned papers do not address the problems of the data warehouse adaptation after changes in data sources. Several approaches have been proposed for solving these problems [10, 11, 18]. These approaches are based on mappings that specify how one schema is obtained from the other schema. This specification is used to adapt one schema after changes in the other schema.

In several papers [3, 14] 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 all these approaches schema versioning is not supported. Several authors [6, 8, 19] propose versioning approach. The main idea in [7] is to store augmented schemata together with schema versions to support cross-version querying. When a new version is created, for all previous versions, augmented schemata are created and populated with data. Though the physical storage and metadata of schema versions are mentioned, the details are not explored. In [15] 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. 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. In [6] the metadata model that supports schema versioning for data warehouses is introduced. Metadata management solutions in a multiversion data warehouse are also proposed in [19], 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 over a multiversion data warehouse are considered in [12]. These papers do not address the problem of physical storage of multiversion data warehouse.

The above-mentioned papers consider only one kind of evolution problems, for example, changes in a schema 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.

3. Data Warehouse Evolution Framework

To handle data warehouse evolution problems, we propose the data warehouse framework depicted in Fig. 65.1. In this section the general issues of the framework are discussed. The detailed description of the framework is given in [16].

The framework is composed of the development environment and user environment. In the development environment the data warehouse metadata repository and other metadata management components 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 by users.

The metadata of the data warehouse schema versions and database structure are stored in the mapping repository. Also the repository includes metadata, which define the logics of ETL processes, and metadata used for data warehouse adaptation. The description of these metadata and the adaptation process is found in [17]. The schema metadata are presented in Sections 4.1 and 4.2.

4. Data Warehouse Metadata

Common Warehouse Metamodel (CWM) [13] was used as a basis of the proposed metamodel of multidimensional data warehouse. CWM is a metadata standard produced by Object Management Group to simplify metadata interchange between data warehousing applications. CWM consists of packages, which describe different aspects of a data warehouse.

Using the following metamodel, it is possible to model a data warehouse schema at physical and logical level. In the physical metamodel an implementation of a data warehouse in RDBMS is specified, but

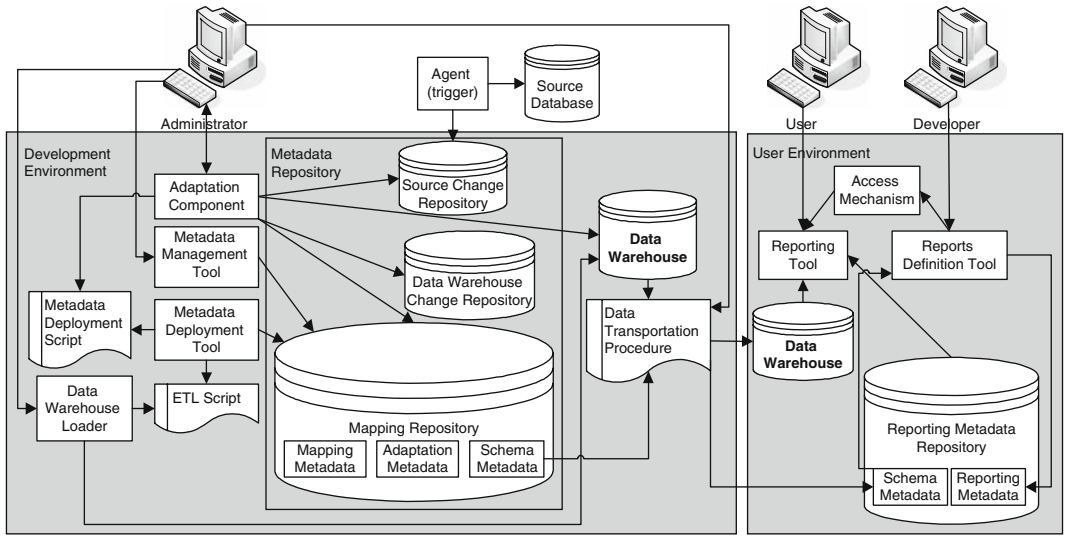

Figure 65.1. Data warehouse evolution framework.

the logical metadata describe schema versions of a data warehouse. The physical level metadata are based on the relational package of CWM. The logical level metadata are based on the OLAP package. These two levels are connected by objects defined in the Transformation package of CWM. Due to space limitations, the description of elements of the physical and logical metadata that are taken from CWM is not given.

4.1. Logical Metadata

Metadata at the logical level describe the multidimensional data warehouse schema. The logical metamodel is depicted in Fig. 65.2.

To reflect multiple versions of a data warehouse schema, two objects were introduced: *schema version* and *version transformation*, which are not included in CWM. If, as a result of any change in a data warehouse schema, a new schema version is created, a new record is inserted in the table schema version. Each schema version has a validity period defined by the columns ValidFrom and ValidTo. The column

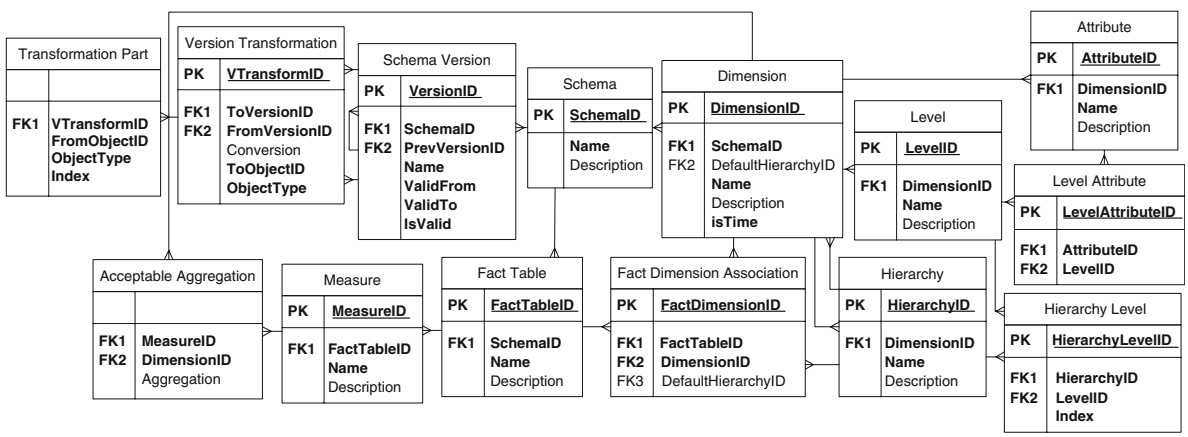

Figure 65.2. Logical metamodel.

IsValid indicates whether a version is currently valid. Each version, except for the first one, has a link to a previous version.

Elements of a version are connected to the table schema version by the table version transformation. The column ToObjectID stores an identifier of an element of the current version. An object can be any element of the logical metamodel, for example, dimension, attribute, hierarchy level, etc. The type of an element is stored in the column ObjectType. If an element remains unchanged it is connected to several versions. In the column Conversion a function that obtains a changed element from elements of other version is stored. Building version transformations, acceptable aggregations of elements are taken into account. The table transformation part stores data about the elements of the previous version that are used to calculate the changed element of a new version.

4.2. Physical Metadata

Metadata at the physical level (Fig. 65.3) describe relational database schema of a data warehouse and mapping of a multidimensional schema to relational database objects. Physical metadata do not include versioning information, because versioning is implemented at the logical level.

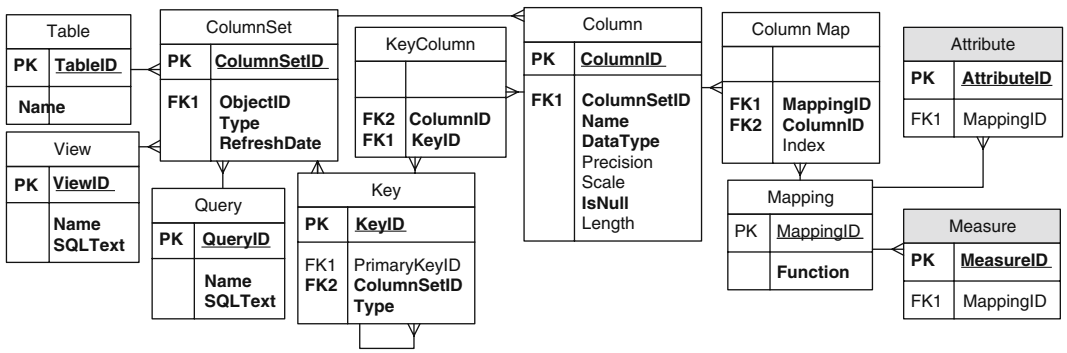

Figure 65.3. Physical metamodel.

CWM metadata were extended by the additional type of column set, a result of SQL *query*. The column RefreshDate was introduced to store the date of the last update of the table or base tables of a view or a query. *Mappings* (FeatureMap in CWM package transformation) specify formulas that obtain attributes and measures from one or several columns. In this chapter it is assumed that attributes and measures are obtained directly from columns by the mapping function "copy."

5. Evolution Support

The evolution framework supports physical changes that operate with tables and columns, and logical changes that modify mainly schema metadata. In this section a result of each change and its' impact on the metadata is described. As a result of a logical change, mainly the logical metadata are adapted. As a result of a physical change, both logical and physical metadata are modified. When a new schema version V_N is created from the previous version V_P, the time of change is recorded in the column ValidTo of the version V_P and the column ValidFrom of the version V_N. In the following sections the corresponding tables of dimensions and fact tables are the tables that include columns that are transformed by a mapping into attributes of dimensions and measures of fact tables in the physical metadata.

5.1. Physical Changes

New Dimension Attribute. When a new attribute A is added to the dimension D, in the database a new column K is added to the corresponding table T. In the physical metadata a new column K is created in the column set T. In the logical metadata a new schema version V_N is constructed from the previous version V_P.

A new attribute A is attached to the dimension D. A mapping with a function "copy" is created to connect the column K at the physical level with the attribute A at the logical level. If A can be calculated from other attributes of the dimension D, then the version transformation is constructed for A from version V_P to version V_N.

Deletion of Dimension Attribute. When an attribute A is deleted from the dimension D, in the database the corresponding column K of the table T is not removed but is no longer updated by ETL processes. The physical metadata remain unchanged. In the logical metadata a new schema version V_N is constructed from the same objects from the previous version V_P except for the attribute A. If the attribute A can be calculated from the remaining attributes of the dimension D, then the version transformation is constructed for A from version V_N to version V_P.

An attribute can be deleted only if it is not connected to any level. If a connection persists, at first an attribute must be disconnected from all levels. It is not allowed to delete an attribute that corresponds to a primary key column.

Change of Datatype of Attribute. When datatype of the attribute A of the dimension D is changed, a new column K with a new datatype is added to the corresponding table T. The column K is updated instead of the changed column. In the physical metadata a new column K is created in the column set T. In the logical metadata a new schema version V_N is constructed from the previous version V_P, but the datatype of A is changed to a new value. A mapping with a function "copy" is created to connect the column K at the physical level with the attribute A at the logical level. Version transformation, which transforms the attribute A in the version V_P to the attribute A in the version V_P by conversion function is created. This means that semantically the attributes in both versions are identical.

Renaming Dimension Attribute. When an attribute A of the dimension D is renamed, in the database and in the physical metadata a corresponding column K is renamed. In the logical metadata a new schema version V_N is constructed from the previous version V_P, but the attribute A is created with a new name. A mapping with a function "copy" is created to connect the column K at the physical level with the attribute A at the logical level. Version transformation with an empty function, which connects the attribute A in the version V_P to the attribute A in the version V_P is created. This means that semantically the attributes in both versions are identical.

New Dimension. When a new dimension D is added, in the database a new corresponding table T with a primary key is created. In the physical metadata the table T and the corresponding column set is created, metadata about columns and keys of the table T are recorded. In the logical metadata a new schema version V_N is constructed from the previous version V_P. A new dimension D with attributes that correspond to columns of the table T is added. Mappings with a function "copy" are created to connect columns of the table T at the physical level with attributes of the dimension D at the logical level. Version transformation is not constructed. Reports that use the dimension D can run only from the time it was created.

Deletion of Dimension. When a dimension D is deleted, in the database the corresponding table T is not removed but is no longer updated by ETL processes. The physical metadata remain unchanged. In the logical metadata a new schema version V_N is constructed from the same objects from the previous version V_P except for the dimension D, attributes, hierarchies and levels of it.

A dimension can be deleted only if it is not connected to any fact table. If a connection persists, at first a dimension must be disconnected from all fact tables.

Renaming Dimension. When a dimension D is renamed, in the database and in the physical metadata a corresponding table T is renamed. In the logical metadata a new schema version V_N is constructed from the previous version V_P, but the dimension D is created with a new name. A mapping with a function "copy" is created to connect columns of T at the physical level with attributes of D at the logical level. Version

transformation with an empty function which connects the dimension D in the version V_P to the dimension D in the version V_P is created. This means that semantically the dimensions in both versions are identical.

New Measure. When a new measure M is added to the fact table F, in the database a new column K is added to the corresponding table T. In the physical metadata a new column K is created in the column set T. In the logical metadata a new schema version V_N is constructed from the previous version V_P. A new measure M is attached to the fact table F. A mapping with a function "copy" is created to connect the column K at the physical level with the measure M at the logical level. If the measure M can be calculated from other measures of the fact table F, then the version transformation is constructed for M from version V_P to version V_N.

Deletion of Measure. When a measure M is deleted from the fact table F, in the database the corresponding column K of the table T is not removed but is no longer updated by ETL processes. The physical metadata remain unchanged. In the logical metadata a new schema version V_N is constructed from the same objects from the previous version V_P except for the measure M. If the measure M can be calculated from the remaining measures of the fact table F, then the Version transformation is constructed for M from version V_N to version V_P.

Change of Datatype of Measure. When datatype of the measure M of the fact table F is changed, a new column K with a new datatype is added to the corresponding table T. The column K is updated instead of the changed column. In the physical metadata a new column K is created in the column set T. In the logical metadata a new schema version V_N is constructed from the previous version V_P, but the datatype of M is changed to a new value. A mapping with a function "copy" is created to connect the column K at the physical level with the measure M at the logical level. Version transformation which transforms the measure M in the version V_P to the measure M in the version V_P by conversion function is created. This means that semantically the measures in both versions are identical.

Renaming Measure. When a measure M of the fact table F is renamed, in the database and in the physical metadata a corresponding column K is renamed. In the logical metadata a new schema version V_N is constructed from the previous version V_P, but the measure M is created with a new name. A mapping with a function "copy" is created to connect the column K at the physical level with the measure M at the logical level. Version transformation with an empty function, which connects M in the version V_P to the measure M in the version V_P is created. This means that semantically the measures in both versions are identical.

New Fact Table. When a new fact table F is added, in the database a new corresponding table T is created. In the physical metadata the table T and the corresponding column set with the type "table" is created, metadata about columns of the table T are recorded. In the logical metadata a new schema version V_N is constructed from the previous version V_P. A new fact table F with measures that correspond to columns of the table T is added. Mappings with a function "copy" are created to connect columns of the table T at the physical level with measures of the fact table F at the logical level. Version transformation is not constructed. Reports that use the fact table F can run only from the time it was created.

Deletion of Fact Table. When a fact table F is deleted, in the database the corresponding table T is not removed but is no longer updated by ETL processes. The physical metadata remain unchanged. In the logical metadata a new schema version V_N is constructed from the same objects from the previous version V_P except for the fact table F, its measures and connections with dimensions.

Renaming Fact Table. When a fact table F is renamed, in the database and in the physical metadata a corresponding table T is renamed. In the logical metadata a new schema version V_N is constructed from the previous version V_P, but the fact table F is created with a new name. A mapping with a function "copy" is created to connect columns of the table T at the physical level with measures of the fact table F at the logical level. Version transformation with an empty function which connects the fact table F in the version V_P to the fact table F in the version V_P is created. This means that semantically the fact tables in both versions are identical.

5.2. Logical Changes

Connection of Dimension to Fact Table. To connect a dimension D to a fact table F, in the database and the physical metadata a foreign key column K (or columns) is added to the corresponding table T_F,

which will be connected to the primary key column (or columns) of the table T_D that corresponds to the dimension D. In the database in the table T_D a new fictive record (for example, with data "all together") is created with an identifier I. If there are data in the table T_F the new column K is filled with I. In the database and the physical metadata a foreign key is created to connect the column K to the primary key of the table T_D or column set T_D. Depending on requirements, in the database and in the physical metadata the column K can be included in the primary key of the table T_F and column set T_F or a new primary key can be created if there is no primary key in the table T_F.

In the logical metadata a new schema version V_N is constructed from the previous version V_P. A fact dimension association is created between the fact table F and dimension D. Version transformations are created for measures of the fact table F if it is possible. One transformation with the function that transforms (divides) measure data to correspond to the version V_N is recorded. Another transformation with the function that transforms (aggregates) measure data in respect of the dimension D to correspond to the version V_P is recorded. If conversion is not possible, reports that span both versions can be run only until or after the change of a schema using only one of the versions V_P or V_N.

Disconnection of Dimension from Fact Table. To disconnect a dimension D from a fact table F, in the database in the table T_D that corresponds to the dimension D a new fictive record (for example, with data "all together") is created with an identifier I. During ETL processes the identified I is stored in the table T_F that corresponds to the fact table F in the foreign key column that was connected to the table T_D. The physical metadata remain unchanged. In the logical metadata a new schema version V_N is constructed from the same objects from the previous version V_P, except for the fact dimension association between the fact table F and the dimension D. Version transformations are created for measures of the fact table F if it is possible. One transformation with the function that transforms (aggregates) measure data in respect of the dimension D to correspond to the version V_N is recorded. Another transformation with the function that transforms (divides) measure data to correspond to the version V_P is recorded. If conversion is not possible, reports that span both versions can be run only until or after the change of a schema using only one of the versions V_P or V_N.

New Dimension Hierarchy. When a new hierarchy H is added to the dimension D, in the logical metadata a new schema version V_N is constructed from the previous version V_P. A new hierarchy H is attached to the dimension D. Version transformation is not constructed. Reports that use the hierarchy H can run only from the time it was created.

Deletion of Dimension Hierarchy. When a hierarchy H is deleted from the dimension D, in the logical metadata a new schema version V_N is constructed from the same objects from the previous version V_P except for the hierarchy H. Version transformation is not constructed. Reports that use the hierarchy H can run only until the time it was deleted.

New Hierarchy Level. When a new level L is added to the hierarchy H, in the logical metadata a new schema version V_N is constructed from the previous version V_P. A new level L is attached to the dimension D, which contains the hierarchy H. The level L is connected to the hierarchy H and the corresponding index of the level L is recorded. If there are any other levels in the hierarchy H that have the index, which is the same or bigger than the index of the level L, their index is increased by 1. Version transformation with an empty conversion is constructed for all changed levels, except for the level L. Version transformation is not constructed for the level L. Reports that use the level L can run only from the time it was created.

Deletion of Level from Hierarchy. When a level L is deleted from the hierarchy H, in the logical metadata a new schema version V_N is constructed from the same objects from the previous version V_P except for the connection between the hierarchy H and the level L. If there are any other levels in the hierarchy H that have the index which is the bigger than the index of the level L, their index is decreased by 1. Version transformation is not constructed. Reports that use the connection between the hierarchy H and the level L can run only until the time it was removed.

Deletion of Level from Dimension. To delete a level L from a dimension D, at first it must be deleted from all hierarchies where it is used. Then in the logical metadata a new schema version V_N is constructed from the same objects from the previous version V_P except for the level L. Version transformation is not constructed. Reports that use the level L can run only until the time it was deleted.

Connection of Level to Hierarchy. To connect a level L to a hierarchy H, in the logical metadata a new schema version V_N is constructed from the previous version V_P and the level L is connected to the hierarchy H and the corresponding index of the level L is recorded. If there are any other levels in the hierarchy H that have the index which is the same or bigger than the index of the level L, their index is increased by 1. Version transformation with an empty conversion is constructed for all changed levels, except for the level L. Version transformation is not constructed for the connection between the hierarchy H and the level L. Reports that use the connection between the hierarchy H and the level L can run only from the time it was created.

Connection of Attribute to Level. To connect an attribute A to a level L, in the logical metadata a new schema version V_N is constructed from the previous version V_P and the attribute A is connected to the level L. Version transformation is not constructed for the connection between the attribute A and the level L. Reports that use this connection can run only from the time it was created.

Disconnection of Attribute from Level. To disconnect an attribute A from a level L, in the logical metadata a new schema version V_N is constructed from the same objects from the previous version V_P, except for the connection between the attribute A and the level L. Version transformation is not constructed for the connection between the attribute A and the level L. Reports that use the connection between the level L and the attribute A can run only until the time it was removed.

6. Conclusions and Future Work

The main contribution of this chapter is a data warehouse repository that describes data warehouse schema at the logical and physical level. The model is based on the CWM standard, which was supplemented with metadata to describe data warehouse schema versions. The proposed metamodel is used in the framework that supports data warehouse evolution caused by changes of business requirements and data.

The physical implementation of the multiversion data warehouse in relational DBMS is proposed. It allows to store all schema versions in one physical schema. Changes that can happen with multidimensional data warehouse and create new schema versions are discussed. Necessary modifications in the proposed metamodel and in the database are given. It would be desirable to evaluate the completeness of the supported changes.

The future research could be connected with querying multiple data warehouse schema versions. The open issues are the construction of SQL queries based on the proposed metadata, processing of results of these queries, and presentation of reports. It is planned to develop a reporting system to allow to execute reports on one or several data warehouse versions using metadata from the presented data warehouse metamodel.

Acknowledgments

This work was supported by the European Social Fund (ESF).

References

1. Blaschka, M.: FIESTA: A Framework for Schema Evolution in Multidimensional Databases. PhD thesis, Technische Universitat Munchen, Germany (2000)
2. Blaschka, M., Sapia, C., Hofling, G.: On Schema Evolution in Multidimensional Databases. In: DaWaK 1999. LNCS, Vol. 1676, pp. 153–164. Springer, Heidelberg (1999)
3. Bellahsene, Z.: Schema Evolution in Data Warehouses. Knowl. Inf. Syst. 4, 283–304 (2002)
4. Body, M., Miquel, M., Bedard, Y., Tchounikine, A.: A Multidimensional and Multiversion Structure for OLAP Applications. In: ACM 5th International Workshop on Data Warehousing and OLAP, pp. 1–6. ACM, McLean, VA (2002)
5. Body, M., Miquel, M., Bedard, Y., Tchounikine, A.: Handling Evolutions in Multidimensional Structures. In: 19th International Conference on Data Engineering, pp. 581–594. IEEE Computer Society, Los Alamitos, CA (2003)

6. Eder, J., Koncilia, C., Morzy, T.: The COMET Metamodel for Temporal Data Warehouses. In: CAiSE 2002. LNCS, Vol. 2348, pp. 83–99. Springer, Heidelberg (2002)
7. Golfarelli, M., Lechtenbörger, J., Rizzi, S., Vossen, G.: Schema Versioning in Data Warehouses: Enabling Cross-Version Querying Via Schema Augmentation. Data Knowl. Eng. 59(2), 435–459 (2006)
8. Hurtado, C. A., Mendelzon, A. O., Vaisman, A. A.: Maintaining Data Cubes Under Dimension Updates. In: 15th International Conference on Data Engineering, pp. 346–357. IEEE Computer Society, Sydney (1999)
9. Kaas, C. E., Pedersen, T. B., Rasmussen, B. D.: Schema Evolution for Stars and Snowflakes. In: 6th International Conference on ICEIS, pp. 425–433. Porto, Portugal (2004)
10. Marotta, A.: Data Warehouse Design and Maintenance through Schema Transformations. Master thesis, Universidad de la República Uruguay (2000).
11. McBrien, P., Poulovassilis, A.: Data Integration by Bi-directional Schema Transformation Rules. In: 19th International Conference on Data Engineering, pp. 227–238. IEEE Computer Society, Bangalore (2003)
12. Morzy, T., Wrembel, R.: On Querying Versions of Multiversion Data Warehouse. In: ACM 7th International Workshop on Data Warehousing and OLAP, pp. 92–101. ACM, Washington, DC (2004)
13. Object Management Group. Common Warehouse Metamodel Specification, v1.1 http://www.omg.org/cgi-bin/doc?formal/03-03-02
14. Rundensteiner, E. A., Koeller, A., Zhang, X.:Maintaining Data Warehouses over Changing Information Sources. Commun. ACM. 43(6), 57–62 (2000)
15. Shahzad, M. K., Nasir, J. A., Pasha, M. A.:CEV-DW: Creation and Evolution of Versions in Data Warehouse. Asian J Information Technol. 4(10), 910–917 (2005)
16. Solodovnikova, D.: Data Warehouse Evolution Framework. In: Spring Young Researcher's Colloquium on Database and Information Systems, Moscow, Russia (2007)
17. Solodovnikova, D., Niedrite, L.: Data Warehouse Adaptation after the Changes in Source Schemata. In: 7th International Balt. Conference DB & IS, pp. 52–63, Vilnius, Lithuania (2006)
18. Velegrakis, Y., Miller, R.J., Popa, L.: Mapping Adaptation under Evolving Schemas. In: 29th International Conference VLDB, pp. 584–595. Morgan Kaufmann, Berlin, Germany (2003)
19. Wrembel, R., Bebel, B.: Metadata Management in a Multiversion Data Warehouse. In: OTM Conferences (2) LNCS, Vol. 3761, 1347–1364. Springer, Heidelberg (2005)