Evolution-Oriented User-Centric Data Warehouse

Darja Solodovnikova and Laila Niedrite

Abstract. Data warehouses tend to evolve, because of changes in data sources and business requirements of users. All these kinds of changes must be properly handled, therefore, data warehouse development is never-ending process. In this paper we propose the evolution-oriented user-centric data warehouse design, which on the one hand allows to manage data warehouse evolution automatically or semi-automatically, and on the other hand it provides users with the understandable, easy and transparent data analysis possibilities. The proposed approach supports versions of data warehouse schemata and data semantics.

1 Introduction

Data warehouses are databases designed for querying and analysing data. The main goal of a data warehouse is to provide the most accurate and historically correct information to users in the most convenient and easy understandable way to support the analysis of business processes and decision making. On the one hand, this means that the data warehouse must reflect all the changes that occur in the analyzed process, and on the other hand users should receive answers to their questions as fast and easy as possible.

Data warehouses integrate information from various data sources that can change in the course of time. Besides, business requirements often evolve at the client level. To reflect this evolution, it is possible to adapt the existing data warehouse schemata and data extraction, transformation and loading (ETL) processes, but this solution can cause a loss of history. This is why it is preferable to keep track of the evolution. This can be realized by data warehouse schema versions. According to [1], '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 paper we propose the approach to supporting data warehouse evolution, creation and management of data warehouse schema versions and versions of data semantics, analysis of data warehouse data by users in easily designed and modified reports that take into account existence of data warehouse versions.

The rest of this paper is organized as follows. In Section 2 the related work is presented. In Section 3 the proposed data warehouse framework is outlined. The Section 4 describes the working example used throughout the rest of the paper for demonstration purposes. The main contribution of this paper is presented in Sections 5 and 6, where the metadata model for a multiversion data warehouse is described and different aspects of creation and execution of reports are discussed. We conclude with directions for future work in Section 7.

2 Related Work

In the literature there are various solutions for the data warehouse evolution problems. In [2] dimension update operators are formally specified, and their effect is studied over materialized views over dimension levels. In [3] the primitive evolution operations that occur over the data warehouse schema are defined. In [4] the generalized data warehouse model formally is defined, which supports extended hierarchies, and the transformation rules of evolution operators are specified.

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 [5], [6]. These approaches are based on mappings or transformations 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 [7], [8] 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.

Several authors [1], [9], [10] propose the data warehouse schema versioning approach to solve the problems of schema evolution. The main idea in [1] is to store augmented schemata together with schema versions to support cross-version querying. Though the metadata of schema versions is mentioned, the details are not explored. In [10] 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 [9] metadata management solutions in a multiversion data warehouse are proposed. Issues related to queries over a multiversion data warehouse are considered in [11], but the translation of queries to SQL is not discussed.

The above mentioned papers consider only one kind of evolution problems, for example, changes in a data warehouse schema raised by evolving business requirements, changes in data sources or data warehouse. In our approach we propose the solution that is able to handle all these kinds of evolution problems. Besides there was too little research on implementation methods of data warehouse evolution and creation and execution of reports on multiple data warehouse schema versions.

3 Data Warehouse Evolution Framework

To handle data warehouse evolution problems, we propose the data warehouse framework. The detailed description of the framework is given in [12]. The framework is composed of the development environment, where the metadata repository is located and ETL processes and change processing is conducted, and the user environment, where reports on one or several data warehouse versions are defined and executed by users.

All operation of the data warehouse framework is based on the metadata, which are used to describe the data warehouse schema versions, their storage in the relational database and semantics of data stored in the data warehouse, and to accumulate information about reports defined by users on schema versions.

The data warehouse evolution framework supports physical, logical and semantic changes that create a new data warehouse schema version. Physical changes operate with database objects (tables, columns), but logical and semantic changes modify mainly schema metadata. The examples of physical changes are renaming, creation or deletion of an attribute, measure, dimension or cube, etc. The examples of logical changes are connection or disconnection of a dimension from a cube, creation or deletion of a hierarchy or level, etc. Semantic changes refer to adaptation of meanings of the same data objects. As a result of a logical change, logical metadata are adapted and there may be no changes at the physical level, except for new keys or key columns. As a result of a physical change, both logical and physical metadata are modified. As a result of semantic change usually the semantic and logical metadata are adjusted.

In this paper we will concentrate on the part of the framework related to the metadata models of data warehouse versions, execution of reports and analysis of data warehouse data.

4 Working Example

Let us consider a working example of a data warehouse, which was designed at our university to store data about course usage in the e-learning environment. The evolution of the data warehouse is depicted in Figure 1. The first version V_1 of the data warehouse was created on 01.09.2004. It contained a cube Activity, where the number of hits, total usage time in hours, and number of active users was accumulated. It was possible to analyze these measures by the course, tool, user role in a course and time.

Using this data warehouse, it became obvious that its' granularity was not satisfactory and on 01.02.2005 the version V_2 was created, where the activity was collected separately for each user and session. The data warehouse was redesigned and two data warehouse versions were created.

The following changes were conducted to implement the new schema version of the data warehouse:

- Physical changes:
  - Addition of new dimensions User and Session;
  - Addition of new dimension attributes Time and Hour of the Time dimension, SubCategory and Category of the Tool dimension;
  - Deletion of measure NumberOfUsers from the Activity cube.

- Logical changes:
  - Connection of the User and Session dimensions to the Activity cube;
  - Addition of new hierarchy ToolHierarchy to the Tool dimension;
  - Addition of new levels ToolName, SubCategory and Category to the ToolHierarchy; levels Time and Hour to the TimeHierarchy;
  - Connection of attributes ToolName, SubCategory, Category, Time and Hour to the appropriate hierarchy levels.

- Semantic change – Change of meaning of the measure TotalTime from ‘Usage time in hours‘ to ‘Usage time in seconds’.

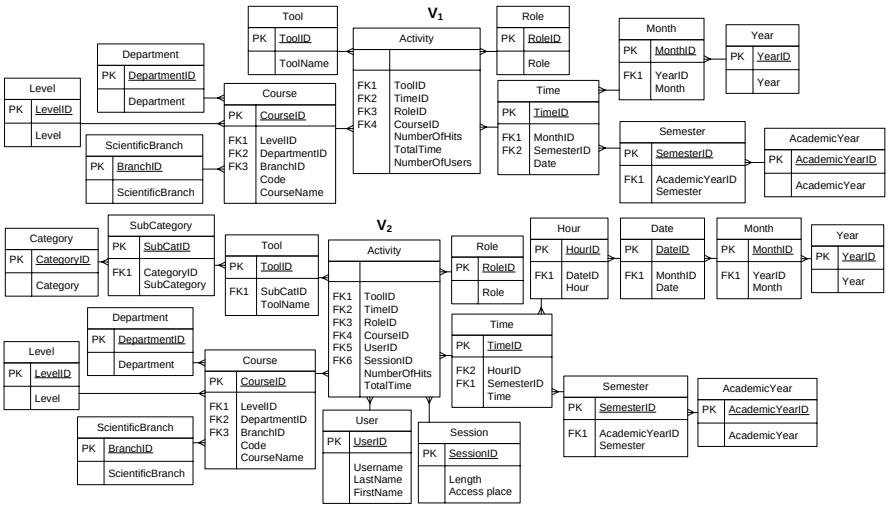

Fig. 1. Data warehouse schema versions

5 Metadata Repository

The metadata repository of the data warehouse evolution framework describes a data warehouse at three levels, namely logical, physical and semantic levels. Be-

sides the repository also contains reporting metadata, which describes the structure of reports generated by users.

Common Warehouse Metamodel (CWM) [13] was used as a basis of the proposed metamodel of multidimensional data warehouse. CWM consists of packages, which describe different aspects of a data warehouse. In the next sections each type of metadata is outlined.

5.1 Logical Metadata

Metadata at the logical level describe the multidimensional data warehouse schema. The logical level metadata are based on the OLAP package of CWM and contains the main objects from this package, such as dimensions and cubes connected by cube-dimension associations, measures, attributes, hierarchies, etc.

To reflect multiple versions of a data warehouse schema, two objects were introduced: SchemaVersion and VersionTransformation (Fig. 2), which are not included in CWM. An object SchemaVersion corresponds to a data warehouse schema version, which is created as a result of some change in a data warehouse schema. Each schema version has a validity period defined by attributes ValidFrom and ValidTill. Each version, except for the first one, has a link to a previous version.

Elements of a version are connected to the SchemaVersion by the VersionTransformation. The association ToElement connects an element of the current version. A Schema Element can be any element of the logical metamodel, for example, Measure, Cube, etc. If an element remains unchanged it is connected to several versions through the VersionTransformation. The attribute Conversion stores a function that obtains a changed element from elements of other version. Elements of other version, which are used to calculate the changed element of a new version, are connected to VersionTransformation by the association FromElements and the corresponding version is connected by the association FromVersion.

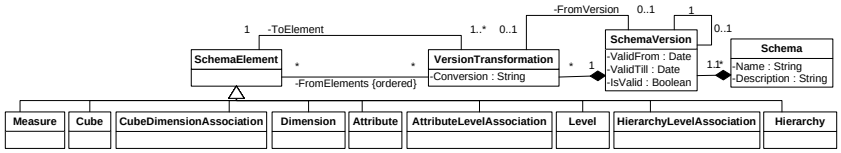

Fig. 2. Schema versions in logical metadata

For the example data warehouse (Fig. 1) in the logical metadata, two schema versions V_1 and V_2 are constructed. Both schema versions include dimensions, cubes, attributes, measures, hierarchies and corresponding associations, according to the schemata in Figure 1. The elements, which are common to both versions, are connected by version transformations with empty conversion functions to ver-

sions V_1 and V_2. The measure NumberOfUsers is connected only to version V_1. Since the granularity of other measures is different in both versions, in the logical metadata the two different measure objects are constructed for each of the measures NumberOfHits and TotalTime and connected to each of the versions. The dimensions User and Session and the corresponding cube-dimension associations between these dimensions and cube Activity are connected only to version V_2. Also the new attributes of the dimensions Time and Tool and corresponding hierarchies made of these attributes exist only in the new version V_2.

The version transformations given in Table 1 are generated for measures. The transformations are created from version V_2 to version V_1, because the transformations of measures in other direction are not possible. Also transformations are possible for new Tool dimension attributes Subcategory and Category.

Table 1. Version transformations

FromVersion	ToVersion	ToElement	Conversion
V_2	V_1	NumberOfUsers	COUNT(DISTINCT Activity.UserID)
V_2	V_1	NumberOfHitsV1	SUM(Activity.NumberOfHitsV2)
V_2	V_1	TotalTimeV1	SUM(Activity.TotalTimeV2/3600)

5.2 Physical Metadata

Metadata at the physical level describe relational database schema of a data warehouse and mapping of a multidimensional schema to relational database objects from the logical level. The model of physical level metadata is shown in Figure 3. Physical metadata do not include versioning information because in the database there is only one schema version and versioning is implemented at the logical level. The physical level metadata are based on the Relational package of CWM. The objects of physical and logical levels are connected by objects defined in the Transformation package of CWM. This means that attributes of dimensions and measures of cubes are defined by Mappings, which specify formulas that obtain attributes and measures from one or several columns of physical tables and views.

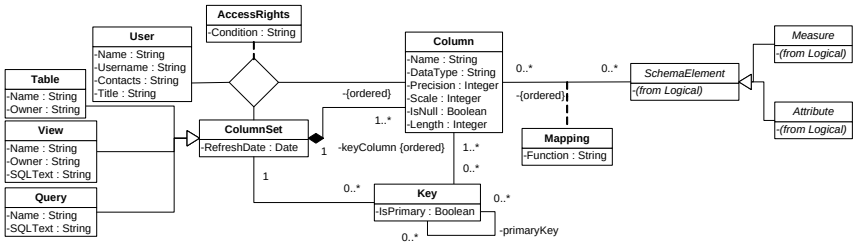

Fig. 3. Physical metadata

For the example data warehouse physically in the database all schema versions are stored in one physical schema. The schema of the first version of the data warehouse is created according to the model V_1 in Figure 1. When the version V_2 is created, the new tables User and Session are created. In these tables new fictive records with data "All together" are created with an identifier U and S respectively. The columns User_ID and Session_ID are added to the table Activity and filled with U and S for all existing data. The table Time and Tool are supplemented with columns Time, Hour and Subcategory, Category respectively. The new columns Subcategory and Category are updated with data for the existing records in tables to form corresponding hierarchies, because these columns correspond to new hierarchy levels of smaller detail. But the new columns of the Time dimension correspond to new hierarchy levels of grater detail, therefore, they are updated with fictive data, for example, time '00:00'. The column NumberOfUsers is not removed, but is no longer updated by ETL processes. For the new versions of other two measures additional columns are not created, i.e. both versions of each measure are stored in one table column.

5.3 Reporting Metadata

Reporting metadata describe structure of reports. CWM contains a package Information Visualization that describes how data warehouse elements are rendered. These metadata are very general and not sufficient, therefore, a new metamodel (Figure 4) was developed. Basically reports are worksheets that contain data items defined by calculations, which specify computation formulas from parameters and table columns, which usually correspond to schema elements. Reports also consist of user-defined conditions and joins between tables.

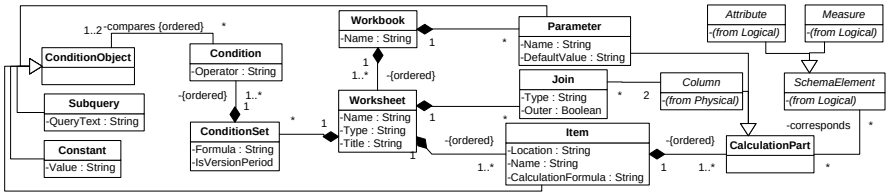

Fig. 4. Reporting metadata

5.4 Semantic Metadata

Data warehouse users must understand the semantics of data that appear in reports from business perspective. They also must be able to analyse these data using all necessary features, including OLAP operations drill-down and roll-up, using hier-

archies. Besides, it is desirable that users can modify or construct reports themselves from elements, which are familiar to them, so that reports creation becomes transparent. For these purposes, it is necessary to describe each element of the data warehouse model in business language. This description could also be used by users to express their requirements for information and changes in requirements making the understanding between users and developers of data warehouse clearer. The description of data warehouse elements in business language is stored in semantic metadata.

In CWM there is the package Business Nomenclature, which can be used to represent business metadata. This package was used as a basis for semantic metadata depicted in Figure 5. The main classes that are used for description of data warehouse elements are Terms and Concepts, which are united in Glossaries and Taxonomies respectively. A concept is the semantic meaning or notion of some data warehouse element or data stored in some element, but a term is particular word or phrase used by users to refer to a concept. Terms define items used in a report. There may be preferred terms and synonym terms to identify a concept. Also terms may be related to each other.

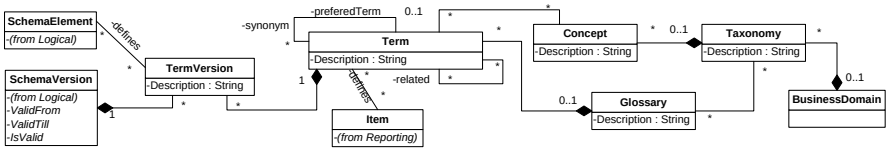

Fig. 5. Semantic metadata

The CWM metamodel was supplemented with the class TermVersion, which is connected to the particular schema version. The new term version reflects the meaning of some data warehouse schema element, which was valid during the particular period of time defined by attributes ValidFrom and ValidTill of the corresponding schema version.

Different versions of terms may be created in case of changes of the data warehouse business requirements. For example, in the working example data warehouse the measure TotalTime is defined by two term versions ‘Total time of all users in hours’ and ‘Total time of each user in seconds’.

6 Reports on Multiversion Data Warehouse

In the evolution data warehouse framework reports are constructed by users, who select desirable terms and term versions from the semantic metadata. The structure of concepts and terms is displayed as a graph, which shows taxonomies of concepts, their associated terms, relations between terms and term versions. The report items, which correspond to selected terms and term versions, are automati-

cally created in the reporting metadata. Users also can define new report items and their corresponding terms, which are saved in the semantic metadata and become available to other users. Users must also specify the location of selected terms in the report. There is also a possibility to define report parameters and conditions. All necessary reporting metadata are created automatically, according to requirements of users.

Example Report. To demonstrate the execution of reports in the following sections, according to the approach presented in the paper, let us use the most popular report executed in the example data warehouse. This report was first developed in the first version of the data warehouse schema. The report is widely used by lecturers of particular courses, who are interested in the detailed analysis of the usage of taught courses. It displays the number of students (NumberOfUsers), who used the specific tool (ToolName) in their course during some period of time and the duration of this tool usage (TotalTimeV1).

6.1 Building Queries on Multiple Versions

6.1.1 Version Selection

When a user runs a report, at first the list of data warehouse versions, which contain report items, is obtained. For each attribute or measure used in the report, all versions that contain it and that were valid in the time period of the report are collected from the logical metadata. If only one version remains then report data are presented according to this version. If more then one version remains, further analysis is necessary.

To determine options of report data presentation, a special relationship matrix is being constructed. The columns of the matrix correspond to versions, but rows correspond to schema elements, selected for the report. If a schema element exists in a version (it is obtained by a version transformation without conversion), then '1' is recorded in the corresponding matrix cell. If a schema element does not exist in a version, but it is obtainable with the transformation version by conversion function from other elements, then the corresponding matrix cell is filled with '2'. If a schema element does not exist in a version and can not be obtained with any conversion function (no version transformation), then in the corresponding matrix cell '0' is recorded. Depending on the values of cells of the relationship matrix, the following options of report presentation are available:

1. The report can be presented *in accordance with one particular version*, if all matrix cells of the column, which corresponds to that version, are filled with '1' and all other cells are filled either with '1' or '2'.

2. If neither of versions does contain all elements, then *elements from different versions can be presented in one report*, if all matrix cells contain '1' or '2'.
3. If any matrix cell contains '0', then the report can only be displayed *separately for each version*.

If the example report is executed for the time period that spans both schema versions, then the corresponding matrix is depicted in Figure 6 (a), where it can be seen that the report can be presented according to the first version. If we change a little the definition of the report to display data by tool Subcategory, the report then can be presented only with elements from various versions (Figure 6 (b).

Schema Element	Version V_1	Version V_2
CourseName (Course)	1	1
TotalTimeV1 (Activity)	1	2
ToolName (Tool)	1	1
Date (Time)	1	1
NumberOfUsers (Activity)	1	2

(a)

Schema Element	Version V_1	Version V_2
CourseName (Course)	1	1
TotalTimeV1 (Activity)	1	2
ToolName (Tool)	1	1
Date (Time)	1	1
NumberOfUsers (Activity)	1	2
SubCategory (Tool)	2	1

(b)

Fig. 6. Relationships Matrix

6.1.2 Query Generation

When a user chooses any option of report presentation, an SQL query is built based on a report definition in reporting metadata, and its result is displayed to a user. An SQL query is constructed, according to special algorithm consisting of the following steps:

1. Analysis of chosen items and determination of used column sets;
2. Analysis of joins;
3. Generation of list of conditions;
4. Grouping and construction of conditions with aggregates functions;
5. Adding restrictions of user rights;
6. Simplification and optimization of the query;
7. Supplementation of a query with version transformations.

The details of the algorithm were published in the paper [14]. According to the query construction algorithm, the query displayed in Figure 7 is constructed for the example report.

```
/*Q1*/SELECT TOOL_NAME, DATE1, SUM(NUMBER_OF_USERS), SUM(TOTAL_TIME) FROM COURSE, TOOL, TIME, ACTIVITY, ROLE
WHERE ROLE.ID=ROLE_ID AND COURSE.ID=COURSE_ID AND TOOL.ID=TOOL_ID AND TIME.ID=TIME_ID
AND DATE1 BETWEEN TO_DATE(:"from",'dd.mm.yyyy') AND TO_DATE(:"until",'dd.mm.yyyy')
AND DATE1 BETWEEN TO_DATE('01.09.2004','dd.mm.yyyy') AND TO_DATE('31.01.2005','dd.mm.yyyy')
AND COURSE_NAME=:"course" AND O140169.ROLE='Student' GROUP BY TOOL_NAME, DATE1
UNION
/*Q2*/SELECT TOOL_NAME, DATE1, COUNT(DISTINCT USER_ID), SUM(TOTAL_TIME) FROM COURSE, TOOL, TIME, ACTIVITY, ROLE
WHERE ROLE.ID=ROLE_ID AND COURSE.ID=COURSE_ID AND TOOL.ID=TOOL_ID AND TIME.ID=TIME_ID
AND DATE1 BETWEEN TO_DATE(:"from",'dd.mm.yyyy') AND TO_DATE(:"until",'dd.mm.yyyy')
AND DATE1 BETWEEN TO_DATE('01.02.2005','dd.mm.yyyy') AND sysdate
AND COURSE_NAME=:"course" AND O140169.ROLE='Student' GROUP BY TOOL_NAME, DATE1;
```

Fig. 7. Query on multiple schema versions

6.2 Using Hierarchy Versions

Data warehouse reports should support high interactivity with a user including OLAP operations: roll-up, drill-down, etc. This is why the reporting tool allows to transform data in the report according to user needs. One of the possibilities to analyze data from different points of view is to use hierarchies defined in the logical metadata. In the metadata, different versions of hierarchies, levels and associations between attributes and hierarchy levels can be created. When a user runs a report, the query construction algorithm also identifies all hierarchies and their structure that exist in the particular data warehouse version, which is selected by a user to present report data. If a user chooses to present elements from different versions in one report, then only hierarchies, which exist in all versions, are available as well as hierarchies that can be transformed by version transformations.

For the example report executed in accordance with the first version, the following hierarchies are available: all Course dimension hierarchies, Time hierarchy, which consists of only three levels Date, Month and Year. But if the report includes tool Subcategory, then additional Tool hierarchy can be used to analyse data, because it exists in the second version and the version transformation can be constructed to map lower hierarchy level ToolName to higher hierarchy levels.

6.3 Using Term Versions

In reports term versions are used to separate different meanings of the same schema element. If any schema element has multiple term versions, when a user runs a report, which includes this schema element, he is informed that for the same schema element two term versions exist. The user has to choose the preferred term version and then the appropriate schema element version is included in the report.

In the example report, the measure TotalTime was used. The semantics of this measure is different in two versions of the data warehouse. In the first version TotalTime was accumulated in hours and summarized for all users of the e-learning environment, but in the second version this measure was accumulated in seconds for each user and session. So to execute the example report the user must select one of two term versions ‘Total time of all users in hours’ or ‘Total time of each user in seconds’ and the appropriate data are presented in the report.

7 Conclusions and Future Work

In this paper we have presented an approach to data warehouse design. This approach is evolution-oriented because it supports adaptation of data sources and

business requirements and allows to propagate different changes in data warehouse creating versions of schemata and data semantics. This approach is user-centric, because users themselves can design reports on multiple data warehouse versions using terms, which are familiar to them. Besides changes in users' requirements are taken into account.

Several directions for future research related to the presented issue are personalization of reports built on multiple data warehouse versions and support of automatic adaptation of data warehouse according to user needs expressed using terms and term versions from semantic metadata without or with minimal participation of a data warehouse administrator.

Acknowledgments This work has been supported by ESF projects No. 2009/0216/1DP/1.1.1.2.0/09/APIA/VIAA/044 and 2009/0138/1DP/1.1.2.1.2/09/IPIA/VIAA/004.

References

1. 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)
2. 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)
3. Blaschka, M.: FIESTA: A Framework for Schema Evolution in Multidimensional Databases. PhD thesis, Technische Universitat Munchen, Germany (2000)
4. Banerjee S., Davis, K.C.: Modeling Data Warehouse Schema Evolution over Extended Hierarchy Semantics. Journal on Data Semantics XIII, LNCS, vol. 5530, 72—96. Springer, Heidelberg (2009)
5. Marotta, A.: Data Warehouse Design and Maintenance through Schema Transformations. Master thesis, Universidad de la República Uruguay (2000).
6. Velegrakis, Y., Miller, R.J., Popa, L.: Mapping Adaptation under Evolving Schemas. In: 29th Int. Conf. VLDB, pp. 584—595. Morgan Kaufmann, Berlin, Germany (2003)
7. Bellahsene, Z.: Schema Evolution in Data Warehouses. Knowl. Inf. Syst. 4, 283–304 (2002)
8. Rundensteiner, E.A., Koeller, A., Zhang, X.: Maintaining Data Warehouses over Changing Information Sources. Commun. ACM. 43(6), 57—62 (2000)
9. Wrembel, R., Bebel, B.: Metadata Management in a Multiversion Data Warehouse. In: OTM Conferences (2) LNCS, vol. 3761, 1347—1364. Springer, Heidelberg (2005)
10. 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)
11. 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)
12. Solodovnikova, D.: Data Warehouse Evolution Framework. In: Spring Young Researcher's Colloquium on Database and Information Systems, Moscow, Russia (2007)
13. Object Management Group. Common Warehouse Metamodel Specification, v1.1 http://www.omg.org/cgi-bin/doc?formal/03-03-02
14. Solodovnikova, D.: Building Queries on Multiple Versions of Data Warehouse. In: 8th Int. Balt. Conf. DB&IS, pp. 75—86, Tallinn, Estonia (2008)