A Multiversion-Based Multidimensional Model

Franck Ravat, Olivier Teste, and Gilles Zurfluh

IRIT (UMR 5505)
118, Route de Narbonne
F-31062 Toulouse cedex 04 (France)
{ravat, teste, zurfluh}@irit.fr

Abstract. This paper addresses the problem of how to specify changes in multidimensional databases. These changes may be motivated by evolutions of user requirements as well as changes of operational sources. The multiversion-based multidimensional model we provide supports both data and structure changes. The approach consists in storing star versions according to relevant structure changes whereas data changes are recorded through dimension instances and fact instances in a star version. The model is able to integrate mapping functions to populate multiversion-based multidimensional databases.

1 Introduction

On-Line Analytical Processing (OLAP) has emerged to support multidimensional data analysis by providing manipulations through aggregations of data drawn from various transactional databases. This approach is often based on a Multidimensional DataBase (MDB). A MDB schema [1] is composed of a fact (subject of analysis) and dimensions (axes of analysis). A fact contains indicators or measures. A measure is the data item of interest. As mentioned in [2], fact data reflect the dynamic aspect whereas dimension data represent more static information. However, sources (transactional databases) may evolve and these changes have an impact on structures and contents of the MDB built on them. In the same way, user requirement evolutions may induce schema changes; *eg*. to create a new dimension or a new "dimension member" [3], to add a new measure,… Changes occur on dimensions as well as facts.

This paper addresses the problem of how to specify changes in a MDB. The changes may be related to contents as well as schema structures. Our work is not limited to represent the mapping data into the most recent version of the schema. We intend to keep trace of changes of multidimensional structures.

1.1 Related Works and Discussion

As mentioned in [3, 4], the approaches to manage changes in a MDB can be classified into two categories.

The first one, also called "updating model" in [1], provides a pragmatic way of handling changes of schema and data. The approaches in this first category support only the most recent MDB version of the schema and its instances. However, working with the latest version of a MDB hides the existence of changes [3]. This category regroups the following works [5, 6, 7].

A Min Tjoa and J. Trujillo (Eds.): DaWaK 2006, LNCS 4081, pp. 65 – 74, 2006.

In the approaches from the second category called "tracking history approaches", changes to a MDB schema are time-stamped in order to create temporal versions. [3] provides an approach for tracking history and comparing data mapped into versions. Their conceptual model builds a multiversion fact table and structural change operators. The proposed mechanism uses one central fact table for storing all permanent time-stamped versions of data. As a consequence, the set of schema changes is limited, and only changes to dimension structure and “dimension instance structure” named hierarchy are supported [4]. The authors detail the related works of “tracking history approaches” and they focus on a model supporting multiversions; *e.g.* each version represents a MDB state at a time period and it is composed of a schema version and its instances [8]. Their approach is not based on a formal representation of versions and their model supports one single subject of analysis. The provided model also does not integrate extracting functions allowing the population of data warehouse version from time-variant transactional sources.

1.2 Paper Contributions and Paper Outline

We intend to represent multidimensional data in a temporally consistent mode of presentation. Our model has the following features.

- The paper deals with the management of constellation changes. A constellation extends star schemata [1]. A constellation regroups several facts and dimensions. Our model supports complex dimensions, which are organised through one or several hierarchies. A constellation may integrate versions, which represents structural changes.
- Our model must support overlapping versions of MDB parts. Each version representing a subject of analysis (fact) and its axes of analysis (dimensions) is time stamped.
- Our model must integrate mapping functions to populate versions.

The remainder of the paper is organized as follows. Section 2 formally defines our conceptual multiversion model dedicated to MDB. Section 3 presents the mapping functions. Section 4 focuses on the prototype implementation.

2 Multidimensional Database Modelling

The conceptual model we define supports temporal changes using multi-versions. The temporal model is based on discrete and linear temporal model. An instant is a time point on the time line whereas an interval represents the time between two instants. We consider in the model valid-time and transaction-time [9]. The valid-time represents time when the information is valid in the real world whereas transaction-time represents time when the information is recording in the MDB. Note that they are various transaction-times at source level and MDB level. At the MDB level, each extraction provides a transaction time point.

2.1 Constellation and Star Version

The model is a conceptual model near user point of views. A MDB is modelled through a constellation, which is composed of star versions modelling schema changes. Each star version is a snapshot of one fact and its dimensions at an extraction time point.

Definition. A *constellation* C is defined by a set of star versions $\{VS_1,\ldots, VS_U\}$.

Definition. A *star version* $\forall i \in [1..u]$, VS_i is defined by (VF, $\{VD_1,\ldots, VD_V\}$, T)

- VF is a fact version,
- $\forall k \in [1..v]$, VD_k is a dimension version, which is associated to the fact version,
- $T = [T_{Start}, T_{End}]$ is a temporal interval during the star schema version is valid.

Example. The following figure depicts an example of constellation evolutions. At T1 the constellation is composed of two star versions ($VS_{1.1}$ and $VS_{2.1}$). Between times T_1 and T_3, the constellation have one new dimension version noted $VD_{2.1}$, which is associated to a new fact version, noted $VF_{1.2}$. A new dimension version, noted $VD_{3.2}$ is deduced from $VD_{3.1}$. According to the model we provided, this constellation is defined by a set of four star versions $\{VS_{1.1}, VS_{2.1}, VS_{1.2}, VS_{2.2}\}$.

- $VS_{1.1} = (VF_{1.1}, \{VD_{1.1}, VD_{3.1}\}, [T_1,T_3])$
- $VS_{2.1} = (VF_{2.1}, \{VD_{3.1}, VD_{4.1}\}, [T_1,T_3])$
- $VS_{1.2} = (VF_{1.2}, \{VD_{1.1}, VD_{2.1}, VD_{3.2}\}, [T_3,T_{Now}])$
- $VS_{2.2} = (VF_{2.1}, \{VD_{3.2}, VD_{4.1}\}, [T_3,T_{Now}])$

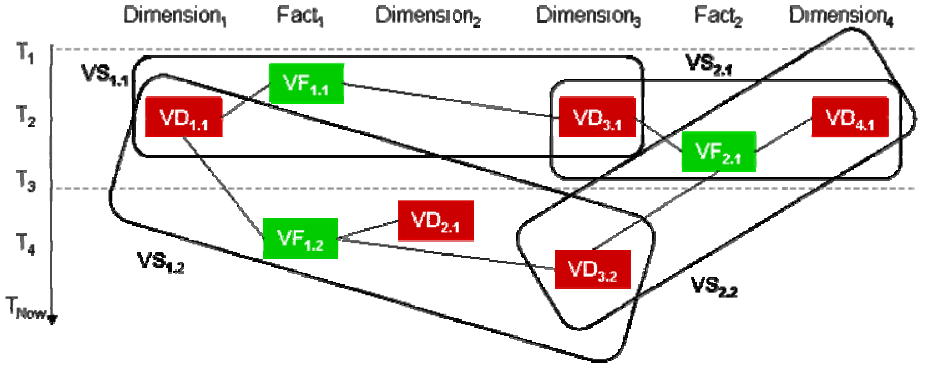

Fig. 1. Example of constellation changes

Note that the model is a multiversion based model because several star versions can be used at a same instant. If source data changes do not require structural change, the current star version is refreshed; *e.g.* new dimension instances and/or fact instances are calculated [10]. If source data changes require structural changes (for example, a hierarchy may be transformed according to new source data), a new star version is defined.

2.2 Star Version Components

Each star version is composed of one fact version and several dimension versions. Each fact version is composed of measures. Each dimension version is composed of properties, which are organised according to one or several hierarchies.

Definition. A *fact version* VF is defined by (N^{VF}, Int^{VF}, Ext^{VF}, Map^{VF})

- N^{VF} is the fact name
- $Int^{VF} = \{f_1(m_1),\ldots, f_p(m_p)\}$ is the fact intention, which is defined by a set of measures (or indicators) associated to aggregate functions,
- $Ext^{VF} = \{i^{VF}_1,\ldots, i^{VF}_x\}$ is the fact extension, which is composed of instances. Each fact instance is defined by $\forall k \in [1..x]$, $i^{VF}_k = [m_1{:}v_1,\ldots, m_p{:}v_p, id^{VD_1}{:}id_1,\ldots, id^{VD_v}{:}id_v, T_{Start}{:}vt, T_{End}{:}vt']$ where $m_1{:}v_1,\ldots, m_p{:}v_p$ are the measure values, $id^{VD_1}{:}id_1,\ldots, id^{VD_v}{:}id_v$ are the linked dimension identifiers and $T_{Start}{:}vt$, $T_{End}{:}vt'$ are transaction-time values,
- Map^{VF} is a mapping function, which populates the fact version.

All fact versions having the same fact name (N^{VF}) depict one fact; *eg*. Each fact version represents a state occurring during its lifetime cycle.

Example. The case study is taken from commercial domain. Let us consider the following fact versions:

- $VF_{1.1}$ = (ORDER, {SUM(Quantity)}, Ext^{VF11}, Map^{VF11})
- $VF_{1.2}$ = (ORDER, {SUM(Quantity), SUM(Amount)}, Ext^{VF12}, Map^{VF12})
- $VF_{2.1}$ = (DELIVER, {SUM(Quantity)}, Ext^{VF21}, Map^{VF21})

$VF_{1.1}$ and $VF_{1.2}$ are two versions of the same fact named ORDER whereas $VF_{2.1}$ is one version of the fact called DELIVER. The following tables show examples of fact instances.

Table 1. Fact instances of Ext^{VF12}

measures		*linked dimension identifiers*			*transaction-time values*	
SUM(Quantity)	**SUM(Amount)**	**IDP**	**IDT**	**IDC**	$\mathbf{T_{Start}}$	$\mathbf{T_{End}}$
200	1500.00	p_1	2006/01/01	c_1	T_1	T_2
250	1800.00	p_1	2006/01/01	c_1	T_2	T_{now}
150	900.00	p_2	2006/01/01	c_1	T_1	T_{now}

Remarks. $\forall i \in [1..u]$, VS_i,=(VF^i, $\{VD^i_1,\ldots, VD^i_V\}$, $[T_{Start}{}^i, T_{End}{}^i]$), $\forall k \in [1..x]$, $i^{VF}_k \in Ext^{VFi}$, then $T_{Start}{}^i \leq T_{Start}{}^{VFi}_k \wedge T_{End}{}^{VFi}_k \leq T_{End}{}^i \wedge T_{Start}{}^{VFi}_k \leq T_{End}{}^{VFi}_k$. In the same way, the transaction time of fact versions or dimension versions may be calculated.

Note that a new fact version is defined when new measures are created or old measures are removed. In the previous example, one new measure, noted SUM(Amount), is created between $VF_{1.1}$ and $VF_{1.2}$ versions.

Definition. A *dimension version* VD is defined by (N^{VD}, Int^{VD}, Ext^{VD}, Map^{VD})

- N^{VD} is the dimension name,
- $Int^{VD} = (A^{VD}, H^{VD})$ is the dimension intention composed of attributes, $A^{VD} = \{a_1,\ldots, a_q\} \cup \{id^{VD}, All\}$, which are organised through hierarchies, $H^{VD} = \{H^{VD}{}_1,\ldots, H^{VD}{}_W\}$,
- $Ext^{VD} = \{i^{VD}{}_1,\ldots, i^{VD}{}_Y\}$ is the dimension extension, which is composed of instances. Each dimension instance is defined by $\forall k \in [1..Y]$, $i^{VD}{}_k = [a_1{:}v_1,\ldots, a_q{:}v_q, T_{Start}{:}vt, T_{End}{:}vt']$ where $a_1{:}v_1,\ldots, a_q{:}v_q$ are dimension attribute values and $T_{Start}{:}vt$, $T_{End}{:}vt'$ are transaction-time values,
- Map^{VD} is a mapping function defining. It defines the ETL process, which populates the dimension (see section 3 for more details).

Definition. A *hierarchy* $H^{VD}{}_i$ is defined by ($N^{VD}{}_i$, $P^{VD}{}_i$, $WA^{VD}{}_i$)

- $N^{VD}{}_i$ is the hierarchy name,
- $P^{VD}{}_i = \langle p_1,\ldots,p_s\rangle$ is an ordered set of dimension attributes, called parameters, $\forall k \in [1..s]$, $p_k \in A^{VD}$, $p_1 = id^{VD}$ is the *root parameter*, p_s = All is the *extremity parameter*,
- $WA^{VD}{}_i : P^{VD}{}_i \rightarrow 2^{AVD}$ is a function associating each parameter to a set of *weak attributes*, which add information to the parameter.

A dimension is depicted by several dimension versions having the same name. Note that a new dimension version is defined when its structure changes; *eg*. when dimension attributes are creating or deleting, hierarchies are adding, deleting or modifying [11].

Example. The facts named ORDER and DELIVER can be analysed according to products. We define two versions of the dimension, named PRODUCT:

- $VD_{3.1}$ = (PRODUCT, ({IDP, Category_Name, Sector_Name, All}, $\{H^{VD31}{}_1\}$) Ext^{VD31}, Map^{VD31}),
- $VD_{3.2}$ = (PRODUCT, ({IDP, Product_Desc, Brand_Desc, Category_Name, Sector_Name, All}, $\{H^{VD32}{}_1, H^{VD32}{}_2\}$), Ext^{VD32}, Map^{VD32}).

These two dimension versions are composed of three hierarchies, which are defined as follows.

- $H^{VD31}{}_1$ = (HSector, <IDP, Category_Name, Sector_Name, All>, {})
- $H^{VD32}{}_1$ = (HSector, <IDP, Category_Name, Sector_Name, All>, {(IDP, Product_Desc)})
- $H^{VD32}{}_2$ = (HBrand, <IDP, Brand_Desc, All>, {(IDP, Product_Desc)})

The following tables show examples of dimension instances. In Ext^{VD31}, we find two products, denoted p_1 and p_2; two dimension instances represent p_1 because its category name changed at t_2.

Table 2. Dimension instances of Ext^{VD31}

IDP	Category_Name	Sector_Name	All	T_{Start}	T_{End}
p_1	Tv	video	all	T_1	T_2
p_1	Television	video	all	T_2	T_3
p_2	Dvd	video	all	T_1	T_3

Table 3. Dimension instances of Ext^{VD32}

IDP	Product_Desc	Category_Name	Sector_Name	Brand_Desc	All	T_{Start}	T_{End}
p_1	14PF7846	television	video	Philips	all	T_3	T_4
p_1	Flat TV 14PF7846	television	video	Philips	all	T_3	T_{now}
p_2	DVP3010	dvd	video	Philips	all	T_3	T_{now}

Transaction-time is an interval associated to the fact instances and dimension instances. Note that the valid time is modelled by temporal dimension in the MDB.

3 Mapping Function

The approach we present is based on decisional systems, which are composed of three levels: (1) operational data sources, (2) data warehouse and (3) multiversion-based multidimensional data marts, noted MDB. In this context, the data warehouse aims to store relevant decisional data and it supports historical data [12]. Usually a data warehouse is implemented in relational database management systems.

This paper focuses on MDB level, which is modeled through constellations. A constellation is composed of fact versions and dimension versions. These versions are populated from data warehouse tables. The mapping functions of these versions (MAP) model the data extraction. We use the relational algebra for defining the extraction process of relational data warehouse data.

Example. The next figure depicts a relational data warehouse schema. This schema is used for populating star versions.

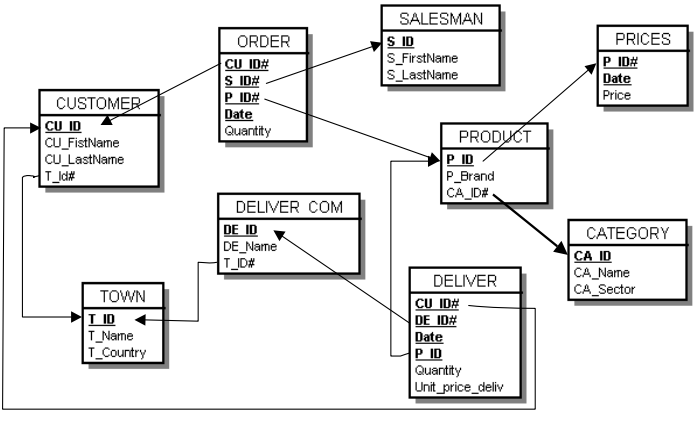

Fig. 2. Example of relational data warehouse schema

From this data warehouse, figure Fig. 3 depicts a constellation schema at T_3 time. This constellation is composed of two star versions

- $VS_{1.2} = (VF_{1.2}, \{VD_{1.1}, VD_{2.1}, VD_{3.2}\}, [T_3, T_{Now}])$, and
- $VS_{2.2} = (VF_{2.1}, \{VD_{3.2}, VD_{4.1}\}, [T_3, T_{Now}])$.

Each star version regroups one fact version and its linked dimension versions. The textual definitions of these versions are:

- $VF_{1.2}$ = (ORDER, {SUM(Quantity), SUM(Amount)}, Ext^{VF12}, Map^{VF12}),
- $VF_{2.1}$ = (DELIVER, {SUM(Quantity)}, Ext^{VF21}, Map^{VF21}),
- $VD_{1.1}$ = (TIME, ({IDD, Month_Name, Month_Number, Year, All}, {$H^{VD11}{}_1$}), Ext^{VD11}, Map^{VD11}),
- $VD_{2.1}$ = (CUSTOMER, ({IDC, Firstname, Lastname, City, Country, All}, {$H^{VD21}{}_1$}), Ext^{VD21}, Map^{VD21}),
- $VD_{3.2}$ = (PRODUCT, ({IDP, Product_Desc, Brand_Desc, Category_Name, Sector_Name, All}, {$H^{VD32}{}_1$, $H^{VD32}{}_2$}), Ext^{VD32}, Map^{VD32}),
- $VD_{4.1}$ = (COMPAGNY, ({IDCP, CName, CCountry All}, {$H^{VD41}{}_1$}), Ext^{VD41}, Map^{VD41}).

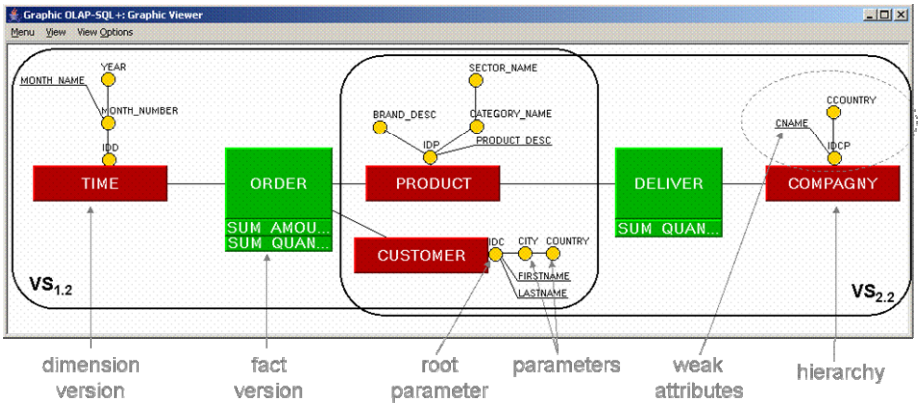

Fig. 3. Graphical representation of a constellation

The extensions of the fact version and its dimension versions of $VS_{1.2}$ are populated from the following map functions.

- Map^{VD32} = **π**(⋈(PRODUCT; CATEGORY; CA_ID=CA_ID); {P_ID **AS** IDP, CA_NAME **AS** Category_Name, CA_SECTOR **AS** Sector_Name, P_BRAND **AS** Brand_Desc})
- Map^{VD11} = **π**(ORDER; {Date **AS** IDC, **TO_CHAR**(Date, 'mm') **AS** Month_Number}, **TO_CHAR**(Date, 'month') **AS** Month_Name, **TO_CHAR**(Date, 'yyyy') **AS** Year})
- Map^{VD21} = **π**(⋈(CUSTOMER; TOWN; T_ID= T_ID); {CU_ID **AS** IDC, CU_FirstName **AS** Firstname, CU_LastName **AS** Lastname, T_Name **AS** City, T_Country **AS** Country})

- Map^{VF12} = **SUM(**⋈**(**⋈**(**⋈**(**ORDER; PRODUCT; P_ID=P_ID**)**; PRICES; P_ID=P_ID**)**; PRODUCT; P_ID=P_ID**)**; ORDER.P_ID, ORDER.Date, ORDER.CU_ID; ORDER.Quantity **AS** Quantity, PRICES.Price**x**ORDER.Quantity **AS** Amount**)**

As illustrating in the following figure, some instances can be calculated from data warehouse data, but others instances may be "derived" from instances of MDB (note that MDB components such as facts and dimensions are viewed as relations). MAP_1 and MAP_4 are mapping functions defining instances (i_{v1} and i_{v4}) from data warehouse data whereas MAP_2 and MAP_3 are mapping functions defining instances (i_{v2} and i_{v3}) from instances of the MDB. This mechanism may be interesting for limiting the extraction process; *e.g.* i_{v1} is calculated from data warehouse data, but i_{v2}, which is an alternative instance at the same time instant, is calculated from i_{v1}.

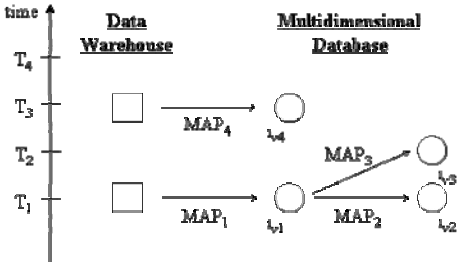

Fig. 4. Mechanism for calculating extensions

4 Implementation

In order to validate the model, we have implemented a prototype, called Graphic OLAPSQL. This tool is developed using Java 1.5 (and additional packages called JavaCC and Oracle JDBC) over Oracle10g Application Server. In [13] we presented the prototype and associated languages and interfaces. The MDB schema is displayed as a graph where nodes represent facts and dimensions while links represent the associations between facts and dimensions (see Fig. 3). These notations are based on notations introduced by [14].

We are extending this prototype for managing versions. Users can display a constellation at one time instant. If several versions (deriving versions) are defined at this time instant, the user chooses its working version, which is displayed. Users express their queries by manipulating the graphical representation of the choosing constellation. The query result is represented through a dimensional-table.

The management of multiversion MDB is based on a metabase. Its schema is represented as follows. A constellation is composed of several star schemata. Each star schema is composed of one fact version and several dimension versions. A dimension version regroups hierarchies which organize attributes of the dimension. A fact version is composed of measures. A same fact version (or dimension version) may be integrated in different star schema. Each fact version (or dimension version) is

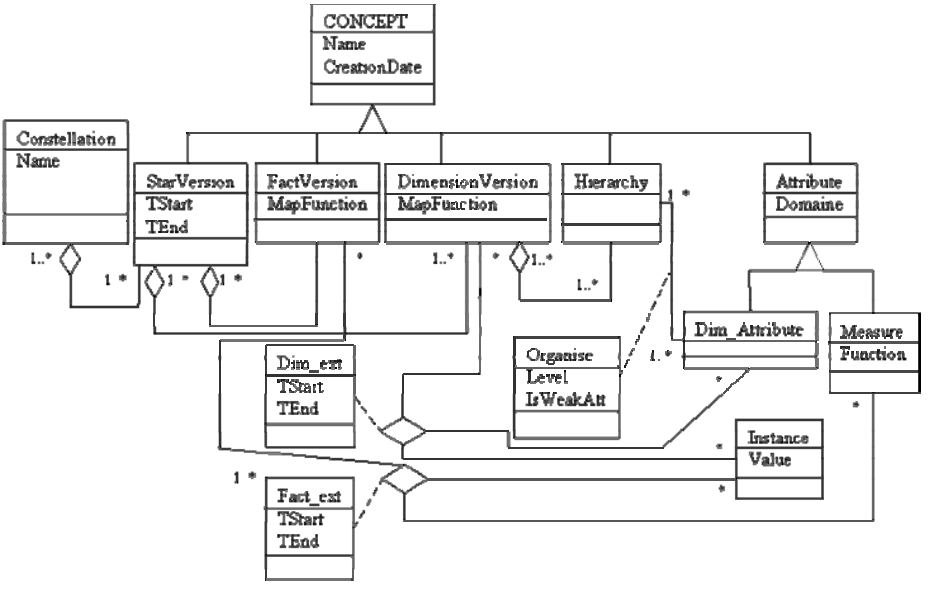

Fig. 5. Metaschema for managing multiversion-based MDB

characterized by a mapping function, its extension (classes Dim_ext or Fact_ext) and its intention.

5 Concluding Remarks

In this paper, we provide solutions for managing data changes of a MDB. The multidimensional model intends to manage several subjects of analysis studied through several axes of analysis. The solution we present is a constellation based on several facts related to dimensions composed of multi-hierarchies.

For supporting changes, a constellation is defined as a set of star versions. A star version is associated to a temporal interval and it is composed of dimension versions (one version per dimension which is composed of a schema and its instances) associated to a fact version (defined by a schema and its instances). A fact version or a dimension version is defined through a mapping function. This function is formalised with a relational algebraic expression on relational data warehouse data to populate the versions. In order to validate our specifications, we are implementing a prototype supporting a multiversion-based constellation.

Our future works consist in specifying a logical model of a multiversion constellation in a relational context [4]. This R-OLAP model must limit data redundancies in order to accelerate OLAP analysis. Moreover, we intend to specify and to implement a query language [15]. In our context, this language must allow the querying of the current star version, or a set of star versions or a specific version. In this paper the mapping functions are based on a single relational data warehouse. We plan to integrate more complex process such as ETL processes [16].

References

1. Kimball, R.: The Data Warehouse Toolkit: Practical Techniques for Building Dimensional Data Warehouses. John Wiley & Sons, Inc, New York, USA, 1996.
2. Vaisman, A.A., Mendelzon, A.O.: A temporal query language for OLAP: Implementation and a case study. 8th Biennial Workshop on Data Bases and Programming Languages - DBPL 2001, Rome, Italy, September 2001.
3. Body, M., Miquel, M., Bédard, Y., Tchounikine, A.: A multidimensional and multiversion structure for OLAP Applications. 5th International Workshop on Data Warehousing and OLAP - DOLAP'02, USA, Nov. 2002.
4. Wrembel, R., Morzy, T.: Multiversion Data Warehouses: Challenges and Solutions. IEEE Conference on Computational Cybernetics - ICCC'05, Mauritius, 2005.
5. Blaska, M., Sapia, C., Hoflin, G.: On schema evolution in multidimensional databases. 1st International Conference on Data Warehousing and Knowledge Discovery - DaWaK'99, pp 153-164, Florence (Italy), August 30–Sept. 1, 1999.
6. Hurtado, C.A., Mendelzon, A.O., Vaisman, A.A.: Maintaining Data cubes under dimension updates. 15th International Conference on Data Engineering - ICDE'99, pp 346-355, Sydney (Australia), March 23-26, 1999.
7. Vaisman, A.A., Mendelzon, A.O., Ruaro, W., Cymerman, S.G.: Supporting dimension updates in an OLAP Server. CAISE'02, Canada, 2002.
8. Bebel, B., Eder, J., Koncilia, C., Morzy, T., Wrembel, R.: Creation and Management of Versions in Multiversion Data Warehouse. ACM Symposium on Applied Computing, pp. 717-723, Nicosia (Cyprus), March 14-17, 2004.
9. Bertino, E., Ferrari, E., Guerrini, G.: A formal temporal object-oriented data model. 5th International Conference on Extending Database Technology - EDBT'96, pp342-356, Avignon (France), March 25-29, 1996.
10. Eder J., Koncilia C., Mitsche D., « Automatic Detection of Structural Changes in Data Warehouses", 5th International Conference on Data Warehousing and Knowledge Discovery – DAWAK'03, LNCS 2737, pp. 119-128, Czech Republic, 2003.
11. Eder, J., Koncilia, C.: Cahnges of Dimension Data in Temporal Data Warehouses. 3rd Int. Conf. on Data Warehousing and Knowledge Discovery – DAWAK'01,LNCS 2114, Munich (Germany), 2001.
12. Ravat, F., Teste, O., Zurfluh, G.: Towards the Data Warehouse Design. 8th Int. Conf. On Information Knowledge Managment- CIKM'99, Kansas City (USA), 1999.
13. Ravat, F., Teste, O. et Zurfluh, G.: Constraint-Based Multi-Dimensional Databases Chapter XI of "Database Modeling for Industrial Data Management", Zongmin Ma, IDEA Group (ed.), pp.323-368.
14. Golfarelli, M., Maio, D., Rizzi, S.: Conceptual design of data warehouses from E/R schemes. 31st Hawaii International Conference on System Sciences, 1998.
15. Morzy, T., Wrembel, R.: On Querying Versions of Multiversion Data Warehouse. 7th International Workshop on Data Warehousing and OLAP - DOLAP'04, pp.92-101, Washington DC (USA), Nov. 12-13 2004.
16. Simitsis, A., Vassiliadis, P., Terrovitis, M., Skiadopoulos, S.: Graph-Based Modeling of ETL Activities with Multi-level Transformations and Updates. 7th International Conference on Data Warehousing and Knowledge Discovery – DaWak'05, LNCS 3589, pp43-52, 2005.