Body, M., M. Miquel, Y. Bédard & A. Tchounikine, 2002, A Multidimensional and Multiversion Structure for OLAP Applications, ACM Fifth International Workshop on Data Warehousing and OLAP,Proceedings, pp. 1-6,

A Multidimensional and Multiversion Structure for OLAP Applications

Mathurin Body
CRG/LISI
Bâtiment 501, 69621
Villeurbanne France
+33 4 72 43 84 83

Maryvonne Miquel
LISI-INSA de Lyon
Bâtiment 501, 69621
Villeurbanne, France
+33 4 72 43 84 83
miquel@if.insa-lyon.fr

Yvan Bédard
CRG
Université Laval, Qc,
G1K7P4, Canada
+1 418 656-5491
yvan.bedard@scg.ulaval.ca

Anne Tchounikine
LISI-INSA de Lyon
Bâtiment 501, 69621
Villeurbanne, France
+33 4 72 43 89 83
atchouni@if.insa-lyon.fr

ABSTRACT

When changes occur on data organization, conventional multidimensional structures are not adapted because dimensions are supposed to be static. In many cases, especially when time covered by the data warehouse is large, dimensions of the hypercube must be redesigned in order to integrate evolutions. We propose an approach allowing to track history but also to compare data, mapped into static structures. We define a conceptual model building a Mutiversion Fact Table from the Temporal Multidimensional Schema and we introduce the notion of temporal modes of representation corresponding to different ways to analyze data and their evolution.

Keywords

Data Warehouse, OLAP, Conceptual Model, Temporal Evolution.

1. INTRODUCTION

Multidimensional models [4,9,10] usually consider that data in fact tables reflect the dynamic aspect of data warehouses, whereas dimension data represent static information [11]. However, in many real-life cases, changes occur on the analysis structures. Recently, literature has brought forward the problem of evolutions in the multidimensional structures and new models have been proposed to handle some of these issues. Some of them, the *updating models* [1,2,7,8], focus on mapping data into the most recent version of the structure, whereas *tracking history models* [3,5,6,11,14] keep trace of evolutions of the system. In this paper, we propose a complete approach taking into account user needs. The remainder of the paper is organized as follows. After motivating examples, section 2 presents a typology of evolutions in multidimensional structures and a study of related work in this research area. Section 3 formally defines our conceptual model. We provide possible adaptations to implement it on current commercial OLAP systems and we present the global architecture that we use for our prototype: section 4 focuses on this implementation and section 5 proposes user interface tools to handle changes in analysis structures. We conclude in section 6.

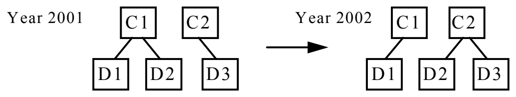
2. MOTIVATIONS AND RELATED WORK

In this section, we present motivating examples and an overview of possible changes on multidimensional structures, which will both give our work guideline. Then, we consider related work and conclude on the position and contribution of our work with respect to the existing approaches.

2.1 Motivating examples

For both of these examples, let us assume that we are working with a multidimensional schema, having a *time* dimension, and a *localization* dimension with two levels *City* and *District*. The observed measure is the number of births, by district and year in its finest granularity.

Example 2.1. Let us start with the following hierarchical structure for the dimension *localization,* where the district D2 is reclassified from C1 to C2 between 2001 and 2002:



If we have the following source data for these two years:

Year 2001		
City	Dist.	Nb births
C1	D1	100
	D2	50
C2	D3	100

Year 2002		
City	Dist.	Nb births
C1	D1	100
C2	D2	100
	D3	50

the query "Total number of births per year and city" could have three possible interpretations:
- The first one returns the number of births considering in which city births appended. This version is called 'Consistent Time'. It returns the following table:

City	2001	2002	Evolution
C1	150	100	↘
C2	100	150	↗

- The second one gives the number of births considering that the districts have always been as they were in 2001:

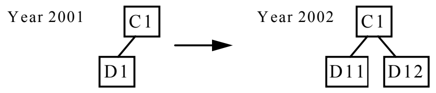
City	2001	2002	Evolution
C1	150	*200*	↗
C2	100	*50*	↘

- The last interpretation returns the following mapped data considering that the districts have always been as they are now (in 2002):

City	2001	2002	Evolution
C1	*100*	100	→
C2	*150*	150	→

This first case-study shows that, depending on the interpretation given to a simple request, results and then conclusions may greatly vary and even be contradictory. It is therefore primordial to let the end-user choose, but also to guide him, throughout all these different interpretations.

Example 2.2. Let us now suppose that we have the following hierarchical structure for the dimension localization, where the district D1 is split into D11 and D12 in 2002:



Suppose we have the following source data for these two years:

Year 2001		
City	Dist.	Nb births
C1	D1	100

Year 2002		
City	Dist.	Nb births
C1	D11	150
	D12	50

As in the previous example, the query "Total number of births per year and district" has three possible interpretations:

- The first one gives once again the results in "Consistent Time":

District	2001	2002	Evolution
D1	100	-	?
D11	-	150	?
D12	-	50	?

- The second one gives results as if the structure has always been as it is in 2001. The mapping is exact and is obtained by adding the values of the parts of D1:

District	2001	2002	Evolution
D1	100	*200*	↗

- Finally, we may want to see data mapped into the last version of the hierarchical structure, but then, we need additional information to calculate the number of births for D11 and D12 in 2001. This information can be given by percentages estimating the population repartition:

District	2001	2002	Evolution
D11 (~ 40% of D1)	*40*	150	↗
D12 (~ 60% of D1)	*60*	50	↘

This second example points out that in most cases, if the 'Consistent Time' view gives 'true' data, it does not allow the tracing of data evolution. That is why we need to map data in a given structure version. In this case, note that the "older" version is less detailed but more truthful than the "current" one, in which approximations are made. These different points of view on data are complementary. Finally, this example also shows that it is essential to be able to distinguish source from mapped data, and to eventually report data reliability.

2.2 Evolutions on multidimensional structures

We divide possible evolutions on a multidimensional structure into *schema evolution* and *evolution on members*. The evolutions members are more difficult to list exhaustively. Hence, we choose to introduce six *simple operations* that could be combined to build *complex operations*. All these operations should be taken into account in our model.

Dimension schema evolution
Creation and deletion of a dimension
Creation and deletion of a hierarchy
Creation and deletion of a level
Move of a level in the hierarchical schema structure
Evolution members: simple operations
Creation of a member
Deletion of a member
Transformation of a member (change of an attribute, its name or meaning…)
Merging of n members into one member
Splitting of one member into n members
Reclassification of a member in the dimension structure
Evolution on members : Exples of complex operations
Decreasing: splitting and deletion
Increasing: creation and merging
Partial annexation: splitting and merging

2.3 Related works

To take such evolutions into account, most of the current OLAP systems report data in the most recent analysis structure. In this case, research focuses on *updating models* [1,2,7,8]. These models provide a pragmatic way of handling evolutions since they allow the temporal comparison of data and then, drawing conclusions that would not be achievable in a changing analysis structure. But in this case, some data is corrupted, or even lost (e.g. deletion of members that do not exist anymore). Moreover, working only with the latest version hides the existence of evolution and information that may be critical for data analysis. False conclusions may even occur from analysis that does not take into account the explicit evolution of information over time.

As early as 1996, Kimball gave good bases on this issue, by introducing three types of "Slowly Changing Dimensions" [9] (SCDs) that are in fact three possible ways of handling changes in multidimensional structures. The first one is to update the data structure (as is done by *updating models*). But as Kimball underlines it, this only "avoids the real goal", which is the tracking of history. As an answer, the Type Two SCD proposes to keep all 'versions' of members. However, in such a representation, comparisons across the transitions cannot be made, since links between them are not kept, even if evolutions are. That is why Kimball proposes a Type Three SCD, in which all evolutions are kept 'inside' members. Nevertheless, limitations exist also for this solution, since overlapping between versions may occur and cannot be handled. It is also "equipped to handle only changes" on members attributes. Although partial, this first study takes into account most users needs and points out the necessity of both keeping track of history and links between transitions.

Recently, in tracking history approaches, some authors [3,5] choose to represent data in the temporally consistent mode of presentation as seen in motivating example. They therefore present the same limits as the Type Two SCD of Kimball, i.e. to not be able to draw comparisons across time. The three most recent models of Mendelzon [11], Pedersen [14] and Eder [6], aware of the users needs of both accurately tracking history and comparing data, provide a way of mapping data in an "unchanged structure", chosen by the user.

Eder [6] proposes mapping functions that allow conversions between structure versions. These mapping functions are used to store the links across transitions (e.g. links between an 'old' member and its new version). They are based on knowledge around evolution operations. Yet, it provides a partial solution, which neither takes schema evolution and time consistent presentation into account, nor considers complex dimension structures.

Pedersen[14] proposes a conceptual model focusing on imprecision and complex dimension structures. Handling evolutions is considered as one aspect. Their approach is however closed to the one previously proposed by Mendelzon [11]. This later approach gives a good overview of the end-user needs. The authors define a temporal multidimensional model that reuses contributions of the temporal databases research area: mainly, the timestamps on the elements of their multidimensional database. Using these valid times, they build the TOLAP Query Language that lets the user choose in his request the way he wants data to be aggregated. He can therefore choose between a temporally consistent representation and the last version. Moreover, they also extend their model to track the links between transitions, and then be able to handle merging and splitting evolutions. Nonetheless, their model does not provide the means of reporting data in any other version than the latest one and to only partially take merges and splits evolutions into account, in comparison with the approach of Eder. In light of the above, the Mendelzon, and Eder approaches seem complementary. The first one lays down bases for handling evolutions in multidimensional structures, whereas the second one gives solutions to exploit knowledge on evolutions in order to map data in a given representation. Using these notions, we introduce a conceptual temporal multidimensional model in order to build a multiversion fact table.

3. CONCEPTUAL MODEL

In this section, our temporal multidimensional model is developed in detail. We then introduce the evolution operators that allow one to take the evolutions of multidimensional structures into account.

3.1 A Temporally Consistent Fact Table

As most of the proposed multidimensional models, our approach is based on a fact table and dimensions. We redefine these elements with valid times, used to define a Temporally Consistent Fact Table.

Definition 1 (Member Version). A Member is an object or an abstract entity of special interest for the end-user. Because changes may occur on this member, a Member Version can be seen as a state of a member, unchanged and coherent over a given time slice. It is represented by a tuple *<MVid, Name, [A], [Level], ti, tf>*, where *MVid* is an identifier for each Member Version, *Name* is the name of the associated member. *A* may be a set of user-defined attributes and values for this Member Version. *Level* is optional and can be used to explicitly define the schema structure (see definition 4). Finally, *[ti, tf]* defines the valid time of this Member Version. Note that a Member may have several valid Member Versions for a given time (when valid times overlap).

Definition 2 (Temporal Relationship). A Temporal Relationship establishes an explicit hierarchical link between two Member Versions and represents a rollup function. It consists of a tuple *<Id_from, Id_to, ti, tf>* where *Id_from* is the identifier of the Member Version child in the relationship and *Id_to* is the one of the Member Version, parent in it. *[ti, tf]* gives the valid time of this relationship. Note that this valid time has to be included in the intersection of the valid times of both Member Versions.

Definition 3 (Temporal Dimension). A Temporal Dimension D is a tuple *<Did, Dname, D, G>* where *Did* is a unique identifier for the dimension and *Dname* its name. *D* is a set of Member Versions and *G*, a set of Temporal Relationships, defining the hierarchical structure of this dimension. Then, a Temporal Dimension can be seen as a directed graph, where nodes are Member Versions of D and arcs are relationships of G.

For any instant *t*, let *D(t)* be *<Did, Dname, D(t), G(t)>* where $D(t) = \{d \in D \mid t \in [t_i^d ; t_f^d]\}$ and $G(t) = \{g \in D \mid t \in [t_i^g ; t_f^g]\}$. More explicitly, *D(t)* (resp. *G(t)*) is the restriction of *D* (resp. *G*) to its elements valid at time *t*. *D(t)* must be a Directed Acyclic Graph (DAG) that represents the dimension structure at this instant *t*.

In the rest of the paper we will call Leaf Member Versions, all Member Versions that have no children at, at least, one instant.

Example 1. In the second motivating example we presented earlier, we have a district D1 that is split into D11 and D12, on and after 2002. In our approach, we have three member versions: *<D1_id, 'D1', District, 01/2001, 12/2001>*, *<D11_id, 'D11', District, 01/2002, Now>* and *<D12_id, 'D12', District, 01/2002, Now>*. These three districts belong to the city C1 defined by the Member Version *<C1_id, 'C1', City, 01/2001, Now>*. The temporal relationships are: *<D1_id, C1_id, 01/2001, 12/2001>*, *<D11_id, C1_id, 01/2002, Now>* and *<D12_id, C1_id, 01/2002, Now>*. A *Loc* dimension is defined as *<Loc_id, Loc, D, G>* where *D* only contains the three district member versions and the city C1, and *G* only has the three temporal relationships previously defined. Figure 1 represents the *Loc* dimension with its temporal elements.

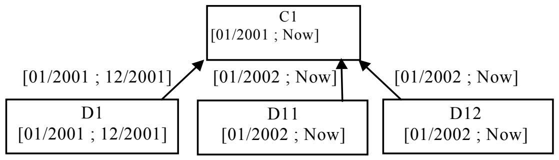


Figure 1. The *Loc* Dimension

Definition 4 (Levels in a dimension). Given a Temporal Dimension D, a level in this dimension is a set of Member Versions that may be defined in two ways. First, if the optional *Level* field is given for all Member Versions of D, then the levels are defined by the sets of Member Versions having the same value for this field. Levels are actually equivalence classes of *D*, for the relation "has same *level* field as". Otherwise, if this option is not defined for all Member Versions, levels are the set of Member Versions of same depth in the DAG of D*(t)*. Note that the notion of level can emerge from the members of D and that it evolves then over time. Thus hierarchy schema is implicit, and be encoded in a relational parent-child table, as in [14]. This allows the representation of non-standard dimensions (non-onto, non-covering, many-to-many hierarchy).

Definition 5 (Temporally Consistent Fact Table). Given *n* temporal dimensions $D_1, ..., D_n$, a Time Dimension *T* and a set of *m* measures $M=\{m_1, .., m_m\}$, a Temporally Consistent Fact Table *f* can be seen as a function :

$$f : D_1 \times ... \times D_n \times T \rightarrow dom(m_1),...,dom(m_m)$$
$$d_1,...,d_n, t \mapsto v_1,...,v_m$$

where D_i is the set of Member Versions of each dimension D_i and $dom(m_k)$ is the range for the measure m_k. This function associates a set of Leaf Member Versions d_i, valid at time *t*, with the values v_k obtained for the measure m_k.

3.2 Design of a MultiVersion Fact Table

In order to keep the links across transitions, i.e. between Members Versions, we introduce Mapping Relationships and Confidence Factor. This will be used to build a Multiversion Fact Table.

Definition 6 (Confidence Factor). A **Confidence Factor** is a value that describes the reliability of data. It allows to distinguish source from mapped data.
Moreover, an aggregate function $\otimes_{cf}$ has to be defined by the designer to obtain this notion of confidence factor for aggregated data in the cube. This aggregate function can either be defined by a function, in case of quantitative Confidence Factors, or by a truth table, if Confidence Factors are given in a qualitative way.

Example 2. In our example, we define the following range of Confidence Factors: CF={*sd*, *em*, *am*, *uk*} where *sd* stands for source data, or temporally consistent data. *em* and *am* mean respectively exact and approximated mapped data, in order to translate that, and how, data is mapped. Finally, *uk* indicates that the mapping relationship is unknown.

Definition 7 (Mapping Relationship). A Mapping Relationship is defined as a tuple <*Id_from, Id_to, F,* F^{-1}>, where *Id_from* is the Leaf Member Version identifier of the Member before changing, and *Id_to*, the identifier of the one after changing. *F* is a set of *m* pairs <fm_k, cf_k> where fm_k is a mapping function from $dom(m_k)$ into itself that specifies how the measure m_k must be mapped ; and cf_k is a confidence factor for this mapping function. Finally, F^{-1} is also a set of *m* pairs <fm'_k, cf'_k>, describing the reverse mapping from *Id_to* to *Id_from*. This lets us deal with the problem exposed by Kimball in his Type Three Slowly Changing Dimensions: keeping links between transitions.
Notice that these mapping relationships are only relevant for Member Versions of the Temporally Consistent Fact Table, i.e. Leaf Member Version. For all non-Leaf Member Versions, mappings will be calculated from the aggregation of their children values (eventually mapped). Confidence Factors are linked here to mapping functions. They will be associated to data later on, once data have been mapped using the mapping functions.

Example 3. In order to map measures from one district to another, let us introduce the following mapping relationships: <*D1_id, D11_id,* {($x \mapsto 0.4x$, *am*)}, {($x \mapsto x$, *em*)}> and <*D1_id, D12_id,* {($x \mapsto 0.6x$, *am*)}, {($x \mapsto x$, *em*)} >, which mean that all values obtained for D11 or D12 will be mapped (just as they are and with the *em* confidence factor) to D1, whereas data associated to D1 will approximately be reported to D11 and D12, by a given proportion.

Definition 8 (Temporal Multidimensional Schema). A Temporal Multidimensional Schema, denoted *TMD*, is a tuple <{D_1, ..., D_n, *T*}, *MR*, *f* >, where D_i are temporal dimensions, *T* is the Time Dimension, *MR* is a set of Mapping Relationships and *f* is a Temporally Consistent Fact Table.

Definition 9 (Structure Version) . A Structure Version *V* of a Temporal Multidimensional Schema *TMD* is a tuple <VS_{id}, {$D_{1,VSid}$, ..., $D_{n,VSid}$}, t_i, t_f>, where VS_{id} is an identifier and [t_i, t_f] defines the valid time of this Structure Version. Each $D_{i,Vsid}$ is the restriction of D_i to its elements (Member Versions and Temporal Relationships) valid for all *t* in [t_i, t_f]. We can see a Structure Version as a valid, unchanged structure over its given valid time.
Note that the Structure Versions of a given Temporal Multidimensional Schema *TMD* partition history and that they can be inferred from *TMD* Schema, as the intersections of the valid time intervals of all Member Versions and Temporal Relationships.
Multidimensional requests may have several ways of presentation: either in the temporally consistent mode or in one of the available Structure Versions. That is why we introduce here the concept of Temporal Mode of Presentation.

Definition 10 (Temporal Mode of Presentation). Given *N* structure versions V_1, …, V_N (deduced from a Temporal Multidimensional Schema *TMD*), we define *TMP* = {*tcm*, VM_1, .., VM_N} as the set of Temporal Modes of Presentation for the data. *tcm* stands for the temporally consistent mode of presentation, and VM_i denotes the temporal mode where data is mapped into the structure version V_i.

Example 4. With the evolution of the district D1, we obtain two Structures Versions: the first one laying from 01/2001 to 12/2001, in which only the district D1 exists; the second one laying from 01/2002 to Now, and where D11 and D12 are valid. We have three temporal modes of presentation: these two structure versions, plus the temporally consistent mode*, tcm*.

Definition 11 (MultiVersion Fact Table). Given a Temporal Multidimensional Schema and its associated set *TMP* of Temporal Modes of Presentation, the MultiVersion Fact Table is a function f' such as:

$$\begin{aligned} f' : D_1 \times \ldots \times D_n \times T \times TMP &\rightarrow dom(m_1) \times \ldots \times dom(m_m) \times CF^m \\ d_1, \ldots, d_n, t, tmp &\mapsto v_1, \ldots, v_m, cf_1, \ldots, cf_m \end{aligned}$$

where *CF* is the range of the Confidence Factors. This function associates a set of values v_k and their respective confidence cf_k, to a set of Leaf Member Versions d_i, valid for the given mode *tmp* (but not necessarily for the given time *t,* if *tmp* is different than the temporally consistent mode), a time *t* and a given Temporal Mode of Presentation *tmp*.
Note that the Temporally Consistent Fact Table is included into this MultiVersion Fact Table. We have indeed:

$$f'\big|_{D_1 \times \ldots \times D_n \times T \times \{tcm\}} = f \times \{sd\}^m$$

where *tcm* is the temporally consistent mode of presentation and *sd* is the confidence for source data.
This MultiVersion Fact Table can be inferred from the Temporal Multidimensional Schema. It can then be automatically calculated from the temporal dimensions, Mapping Relationships and the Temporally Consistent Fact Table.

Definition 12 (Data Aggregation). Suppose given an aggregate function $\oplus_{m_k}$ for each measure m_k, $\otimes_{cf}$ the aggregate function of the Confidence Factors, *tmp* an element of *TMP* and *t* an instant of the time axis. Data Aggregation in the cube can easily be processed, using the MultiVersion Fact Table and Temporal Relationships between Members Versions. As an example, let us assume that we want to aggregate data for a non-Leaf Member Version *d* from the dimension D_1. From G_1, set of temporal relationships of D_1, we can find $d^1, \ldots, d^J$ the children of *d*. Let us also suppose that we have :

$$\forall j \in [1, J], \quad f'(d^j, d_2 \ldots, d_n, t, tmp) = v_1^j, \ldots, v_m^j, cf_1^j, \ldots, cf_m^j$$

We can then obtain the aggregated values for *d* as:

$$f'(d, d_2, \ldots, d_n, t, tmp) = \bigoplus_{m_1}{}_{j=1}^{J} v_1^j, \ldots, \bigoplus_{m_m}{}_{j=1}^{J} v_m^j, \bigotimes_{cf}{}_{j=1}^{J} cf_1^j, \ldots, \bigotimes_{cf}{}_{j=1}^{J} cf_m^j$$

Such operations will have to be proceeded in order to build an OLAP cube.

3.3 Structural Evolution Operators

To modify the structure of a Temporal Multidimensional Schema, we provide four basic operators: *Insert*, *Exclude*, *Associate* and *Reclassify*. The administrator integrates changes in the structure by means of these operators.

Insert(Did, mvID, mName, [A] , [level], t_i, [t_f], P, C) inserts a new Member Version *mvID* with the name *mName* in dimension *Did* as *<mvID, mName, [A], [level], t_i, [t_f]>*. Its position into the hierarchical structure of the dimension is specified by *P* and *C*, respectively set of its parents and children. Temporal Relationships are created from t_i [to t_f] to integrate this change. Note that t_f is optional and, if omitted, is replaced by *Now*.

Exclude(Did, mvID, t_f) excludes the Member Version *mvID* from dimension *Did* on and after t_f. It means that the end time of *mvID* and all Temporal Relationships involving this Member Version is set to t_f*-1*.

Associate(Rmap). Rmap is a Mapping Relationship that defines a new transition link between two versions of a member. It will simply be checked for consistency and added to *MR*, the set of Mapping Relationships.

Reclassify(Did, mvID, t_i, [t_f], OldParents, NewParents) changes the position of member version *mvID* in the hierarchical structure of dimension *Did,* by redefining the set of its parents. *NewParents* (resp. *OldParents*) is a set of the member versions that are (resp. are no more) parents of *mvID* on and after t_i. This operator updates the end time of all temporal relationships concerned (i.e. involving member versions from *OldParents* and *mvID*) and/or inserts new ones. Note that *OldParents* or *NewParents* may be empty.

Using these four basic operators, we cover most of the evolution operations introduced in 2.2. By combining them we are indeed able to translate simple and complex operations into a sequence of basic operators.

4. ARCHITECTURE AND PROTOTYPE

4.1 Adaptation of the conceptual model

Current multidimensional systems are only made of dimensions and fact table(s). Therefore, integrating our notions of temporal mode presentation and the confidence factor in commercial tools seems difficult. But two main adaptations can be done to handle these features and emerged from the definition of the MultiVersion Fact Table.

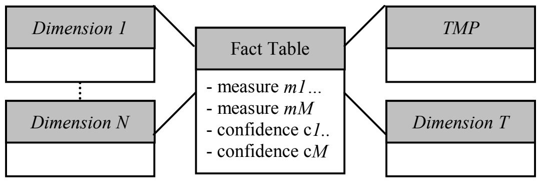


Figure 2: Logical model

Temporal modes of presentation: we represent the set *TMP* of the temporal modes of presentation, as a 'flat' dimension, i.e. without hierarchical structure. This choice offers all the flexibility provided by a usual dimension, during cubes exploration (comparing different structure versions, switching between temporal modes, rotating…).

Confidence factor: each confidence factor, characterizing a value, may be seen as a measure in the fact table, associated to the same members in the multidimensional structure. Moreover, the aggregate function becomes a 'usual' aggregate function, as it must be defined for all measures in a multidimensional structure.

Logical model: finally, using the well-known star schema, we can represent the logical model as figure 2.

4.2 Implementation

The most recent commercial tools now handle several ways of structuring dimensions into data warehouses with a "normalized" or "denormalized" representation. To implement our model, we choose the Parent-Child Dimension in Microsoft SQL Server 2000 [13] where dimension does not have explicit hierarchy and where hierarchy is only deduced from links between members. To implement our model on existing tools (SQL Server 2000 Analysis Services, OLE DB for OLAP [12] and ProClarity 4.0 Components) we choose an architecture divided into three parts:

- **A Temporal Data Warehouse**, that contains the Temporal Multidimensional Schema (temporally consistent data), and metadata related to it, including Mapping Relationships.

- **A MultiVersion Data Warehouse** in which the 'temporal mode of presentation' dimension has been proceeded and the MultiVersion fact table has been inferred from the Temporally Consistent fact table and Mapping Relationships.

- An **OLAP cube** built from the MultiVersion Data Warehouse, using aggregations, and that allows requests to integrate the temporal modes of presentation concept.

Up to now, to make our system run on current OLAP tools we have to duplicate the values in all versions. This obviously implies a high level of useless redundancies inside the MultiVersion Data Warehouse and the OLAP Cube, since we could only store differences between versions instead of replicating all values.

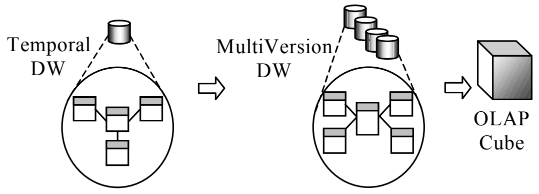


Figure 3. Architecture of the temporal multidimensional system.

4.3 Tools for the end-user interface

Most of our work is motivated by end-user needs. Following this guideline, we give here some ways of providing a user-friendly interface handling temporal evolutions in OLAP structures. For our prototype, we develop a Microsoft Visual Basic 6.0 interface based on ProClarity 4.0 components and our own components for exploring the cube. Figure 4 shows the interface with the main window (A) and the Proclarity component for the user queries (D) and a resulting grid (E).

Confidence Factor: different confidence factors are represented by different background colors. Therefore, the user is able to see at a glance, which data are mapped (with approximation or not) from those that are source data. (figure 4, E).

Highlighting differences between temporal modes: the user may indeed cross two or more temporal modes in a request (figure

4, D). Thus we choose to color the values changing from one version to another.

Metadata: we give the user a full description of the evolutions in the analysis structure by a simple right click on any cell of the grid (figure 4, B). Thus we provide a full and simple access to the metadata describing all evolutions of member versions, how the data have been mapped, what the confidence factor means in a particular case, how it has been aggregated and so on.

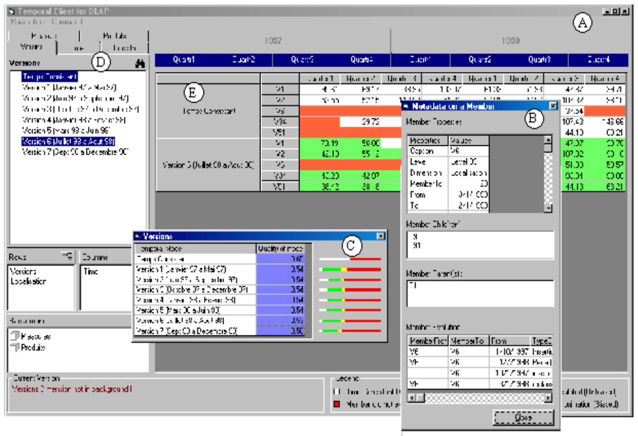


Figure 4: User Interface

Guidance for navigating: the user can be helped in his navigation through the cube and mainly between different temporal versions. It may append that the number of temporal modes of presentation become quite important. It may be fastidious to reformulate the same request in all possible modes to find the most appropriate one. That is why we propose a way of guiding the user in this task. As the notion of "best version" is completely user dependent, we let him parameter the weight given to each type of confidence. Once a request is built, we evaluate by a global factor the quality of the result in each version, based on the number of sources, approximately and exactly mapped data (figure 4, C). The system is able to rank the temporal modes of presentation according to the user's own wishes.

5. CONCLUSION

We propose a novel temporal multidimensional model for supporting evolutions on multidimensional structures. We introduce the notion of temporal mode of presentation to let the user choose which interpretation he wants to give to his request. We also provide confidence factors that allow the user to detect at a glance, values that are mapped, and then altered somehow, from those that come from source data. Moreover, we extend this notion to allow the taking into account approximated, exact and unknown mappings. With respect to users needs, we provide several tools to guide the user and help him to deal with evolutions in analysis structures. Our attention is also on the designer needs, which are mainly: handling most of the possible evolution operations and allowing to have non trivial hierarchical structures in dimensions, to match with real-life cases and as was pointed out by Pedersen.

We propose successive adaptations to make our formal conceptual model compatible with current commercial tools, leading to a prototype. We hav focus on the validation of our approach and the user interface facilities we have introduced.

We would now like to use the potentials of this first approach for extending it. It could indeed be relevant to let the user choose, in an advanced mode, which temporal mode of presentation he wants to have for each dimension. This will be part of our future work.

6. REFERENCES

[1] M. Blaschka, C. Sapia and G. Höfling. On Schema Evolution in Multidimensional Databases. In *Proceedings of the DaWak'99 Conference*, Florence, Italy, 1999.

[2] M. Blaschka. FIESTA: A Framework for Schema Evolution in Multidimensional Information Systems. In *Proceedings of 6th Doctoral Consortium,* Germany, 1999.

[3] R. Bliujute, S. Saltenis, G. Slivinskas, C.S. Jensen. Systematic Change Managment in Dimensional Data warehousing. In *Proceedings of the 3rd International Baltic Workshop on DB and IS*, 1998.

[4] L. Cabibbo and R. Torlone. A Logical Approach to Multidimensional Databases. In *Proocedings of the 6th International Conference on Extending Database Technology (EDBT'98)*, Valencia, Spain, 1998.

[5] P. Chamoni and S. Stock. Temporal Structures in Data warehousing. In *Proceedings of the DaWaK'99 Conference*, Florence, Italy, 1999.

[6] J. Eder and C. Koncilia (2001). Evolution of Dimension Data in Temporal Data Warehouses. In *Proceedings of the DaWaK'01 Conference,* Munich, Germany, 2001.

[7] C. Hurtado, A.O. Mendelzon and A. Vaisman. Maintaining Data Cubes Under Dimension Updates. In *Proceedings of the IEEE/ICDE'99 Conference*, 1999.

[8] C. Hurtado, A.O. Mendelzon and A. Vaisman. Updating OLAP Dimensions. In *Proceedings of the ACM Second Int. Workshop on Data Warehousing and OLAP*, USA, 1999.

[9] R. Kimball. *The Data Warehouse Toolkit*. J.Wiley and Sons, Inc, 1996.

[10] W. Lehner. Modeling large OLAP scenarios. In *Proceedings of the 1998 International Conference on Extending Database Technology*, Valencia, Spain, 1998.

[11] A.O. Mendelzon and A. Vaisman. Temporal Queries in OLAP. In *Proceedings of the 26th VLDB'00 Conference*, Cairo, Egypt, 2000.

[12] Microsoft Corporation, *OLE DB for OLAP Specification Version 2.0*. Microsoft Technical Document, 2001.

[13] J. Mundy. *Microsoft SQL Server 2000 as a "Dimensionally Friendly System"*. Microsoft Corporation, 2001.

[14] T.B. Pedersen, C.S. Jensen and C.E. Dyreson. A foundation for capturing and querying complex multidimensional data. In *Information Systems, Vol 26, No 5, Special Issue: Data Warehousing*, 2001