published in: Proceedings of OES-SEO 2001 Workshop, ISBN 88-88047-21-2, pp 48-54

# A Model for a Temporal Data Warehouse

Johann Eder
*University of Klagenfurt*
*Dep. of Informatics-Systems*
*eder@isys.uni-klu.ac.at*

Christian Koncilia
*University of Klagenfurt*
*Dep. of Informatics-Systems*
*koncilia@isys.uni-klu.ac.at*

Tadeusz Morzy
*Poznan University of Technology*
*Institute of Computing Science*
*morzy@sol.put.poznan.pl*

July 2001

## Abstract

*Data warehouses are a primary means for a consolidated view on the data within an enterprise and frequently a first step in integrating enterprise information systems. Above all, data warehouses are used for analyzing enterprise data online, giving the possibility to aggregate and compare data along dimensions relevant in the application domain. Typically time is one of the dimensions we find in data warehouses allowing comparisons of different periods. The instances of dimensions, however, change over time – countries unite and separate, products emerge and vanish, organizational structures evolve. In current data warehouse technology these changes cannot be represented adequately since all dimensions are (implicitly) considered as orthogonal, putting heavy restrictions on the validity of OLAP queries spanning several periods.*

*In this paper we briefly propose an architecture for temporal data warehouse systems which allows the registration of temporal versions of dimension data and the transfer of data between different temporal versions.*

## 1 Introduction and Motivation

The integration of enterprise information systems is frequently a cumbersome and long-lasting activity. Examples for such situations are cooperate information systems after mergers and acquisitions, or if different software products are employed, which were developed independently, or which were purchased for specific purposes and cannot be replaced so easily. A first step to obtain full integration of enterprise information is to integrate data stemming from various sources, and make these data available for supporting management decisions.

Such an integration of data, where the operational data is left with heterogeneous, partly incompatible information systems, but the data needed for steering the enterprise is extracted from the operational data sources, integrated and purified can be achieved with data warehousing technology. The data kept in data warehouses provides the information in a way that is suitable for steering enterprise processes. And as a result of decision making processes data can be fed back to the operational information systems.

Data warehouses or data marts are usually defined as integrated materialized views over several data sources which can be conventionally structured or semi-structured data. Hence, data warehouses are often used to integrate data.

They do frequently serve as a basis for CRM-Systems, ERP-Systems and other information systems. The most important usage of data warehouses is On-Line Analytical Processing (OLAP) typically using a multi-dimensional view of the data. OLAP tools then allow to aggregate and compare data along dimensions relevant to the application domain. Typical dimensions found frequently in business data warehouses are time, organizational structure (divisions, departments, etc.), space (cities, regions, countries) and product data.

This multi-dimensional view provides long term

data that can be analyzed along the time axis, whereas most OLTP systems only supply snapshots of data at one point of time. Available OLAP systems are therefore prepared to deal with changing values, e. g. , changing profit or turnover. Surprisingly, they are not able to deal with modifications in dimensions, e. g. , if a new branch or division is established, although time is usually explicitly represented as a dimension in data warehouses. Neither did this problem attract much research so far, rare examples are [3, 10].

The reason for this disturbing property of current data warehouse technology is the implicitly underlying assumptions that the dimensions are orthogonal. Orthogonality with respect to the dimension time means the other dimensions ought to be time-invariant. This silent assumptions inhibits the proper treatment of changes in dimension data.

Naturally, it is vital for the correctness of results of OLAP queries that modifications of dimension data is correctly taken into account. E.g. when the economic figures of European countries over the last 20 years are compared on a country level, it is essential to be aware of the re-unification of Germany, the separation of Czechoslovakia, etc. Business structures and even structures in public administration are nowadays subject to highly direct changes. Comparisons of data over several periods, computation of trends, computation of benchmark values from data of previous periods have the necessity to correctly and adequately treat changes in dimension data. Otherwise we face meaningless figures and wrong conclusions triggering bad decisions. From our experience we could cite too many such cases.

## 2 Temporal Data Warehouse

For obtaining correct results of OLAP queries it is essential to keep track of the modifications of dimension data.

Therefore, it is necessary to introduce a temporal extension to the well known data warehouse architecture, e. g. [8, 6, 1]. Hence, all dimension members and all hierarchical links between these dimension members have to be time stamped with a time interval $[T_s, T_e]$ representing the valid time where $T_s$ is the beginning of the valid time, $T_e$ is the end of the valid time and $T_e \geq T_s$.

Using temporal projection and temporal selection as defined in the *Consensus Glossary of Temporal Database Concepts* [7] we are now able to create what we call *Structure Versions* (SV) out of the temporal data warehouse. A structure version represents a view on the temporal data warehouse, that holds a structure valid for a certain time interval. Each modification of a dimension member or a hierarchical relation leads to a new structure version, if a structure version for the given time interval does not already exist. Such modifications can be done with the three basic operations `INSERT`, `UPDATE` and `DELETE` and some complex operations, e. g. , `SPLIT` and `MERGE`.

A first ***defensive approach*** to query such a temporal data warehouse can now be accomplished. The idea of the defensive approach is that for each stated query the systems checks whether or not the query crosses the boundaries of a structure version. If the boundaries between structure versions are crossed the system has two alternatives: (a) it could reject the query or (b) it could issue a warning.

Nevertheless, the defensive approach cannot answer all queries correctly although they might be (at least approximatively) answerable. Consider for example that from March 2000 onwards a department $A$ splits up into $A_1$ and $A_2$. If we want to analyze all months of the year 2000, we will, for division $A$, only have the data for January and February, whereas from March onwards we will only have the data for divisions $A_1$ and $A_2$. It is therefore taken for granted that the user of the analysis is in possession of an adequate knowledge of the domain and that he/she knows that the divisions $A$, $A_1$ and $A_2$ are related and how they are related.

Hence, we have to introduce a mechanism to inform the system about dependencies and relations between dimension members in different structure versions.

In the example just given we could define that it is possible to represent the turnover of the division $A_1$ for the periods before March 2000 as a function $turnover(A_1, period) = 30\%\ of\ turnover(A, period)$. We could also show that for all periods from March 2000 onwards the number of employees $M\#$ of the division $A$ corresponds to the function $M\#(A, period) = M\#(A_1, period) + M\#(A_2, period)$. Using such functions, it is possible to assure that a successful analysis can be made even though there might

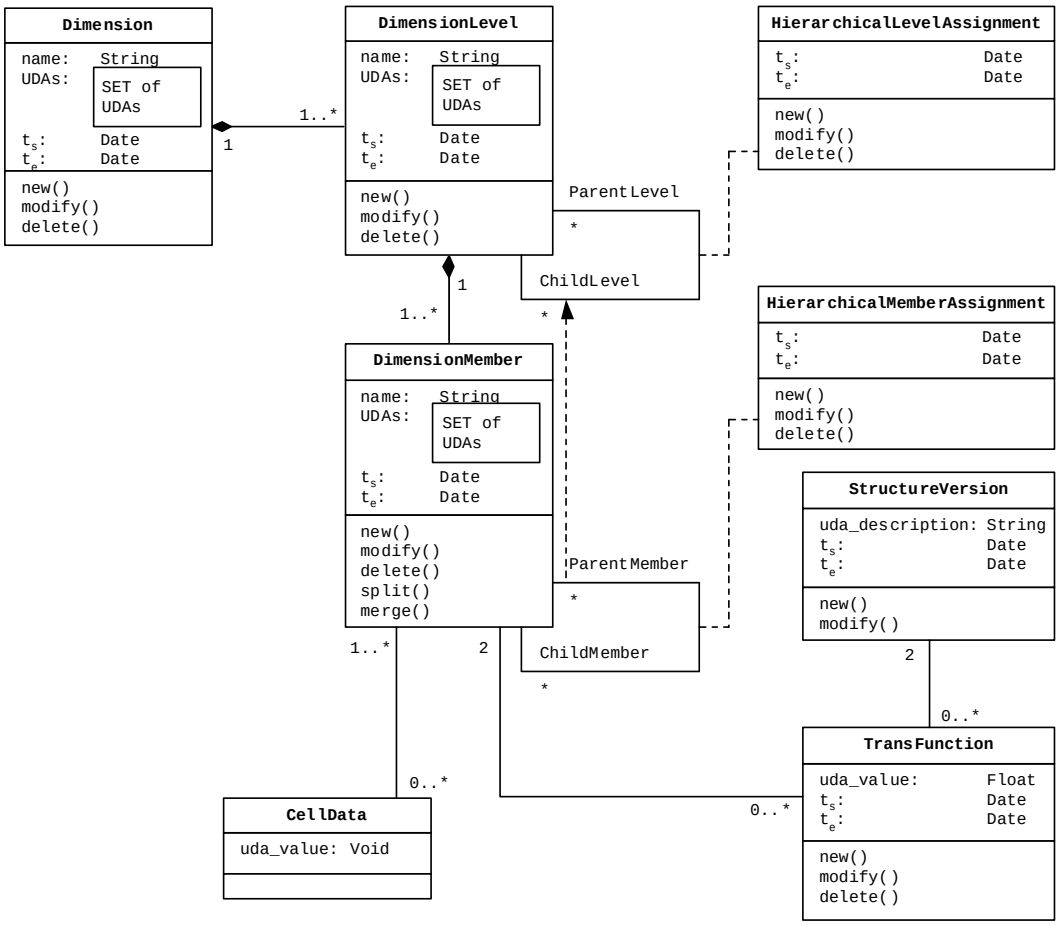


Figure 1: Generic meta-model for a temporal data warehouse system

be changes in the structure.

We call these functions *transformation functions* and introduce an operation *MapF* to define such transformation functions.

Our temporal data warehouse model proposed in [4] enables us to deal with changes on the instance level. It mainly consists of three parts (see Fig. 3):

- **Structure Data**: Holds the necessary information regarding all structures of our data warehouse, i. e. , it stores all dimension data and their hierarchical structure time stamped.
- **Structural Mappings Data**: Stores all defined *MapF* functions, i. e. , all transformation functions that are necessary to map data from one structure version into another.
- **Fact Data**: Consists of all fact data, i. e. , measures, in our data warehouse. In other words, it stores all cell values.

Using this information we are able to query data with two different ***offensive approaches*** that will be discussed in Sect. 3.

We are currently working on an extension to our temporal data warehouse model that enables us to deal with changes on both the instance and the schema level. We denote the dimensions and dimension levels of a data warehouse as its schema and the dimension members of a data warehouse as its instance.

The main goal of this UML model was to define a

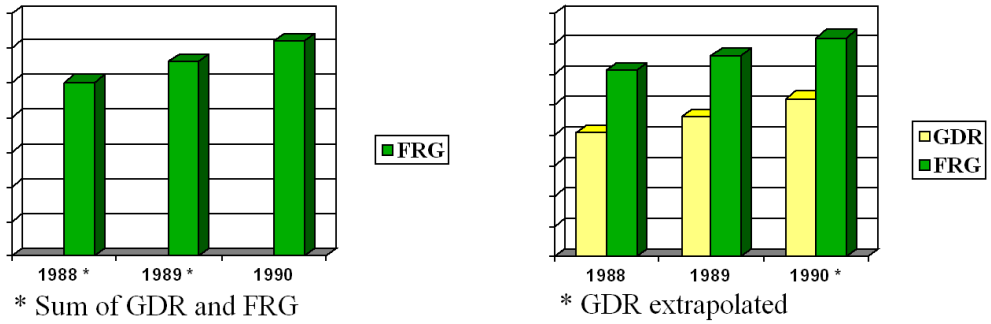


Figure 2: Result of a query with a) $SV_2$ or $SV_3$ and b) $SV_1$ as base structure version

meta-model that allows us to describe data warehouse applications in an evolving environment. The meta-model for a generic, temporal data warehouse is given in Fig. 1.

We define our temporal data warehouse as a set of dimensions, for example the dimensions *Time*, *Facts* and *Products*. It should be emphasized that in contrast to most publications in the field of data warehousing, e. g. [9] and [2], we do not distinguish between dimension and facts. In fact, we claim that facts can be simply described using one dimension *Facts*. Hence we are able to deal with facts that are in a hierarchical order. Just as we are able to describe a dimension *Market* where stores belong to regions and regions to states we are able to similarly describe a dimension *Facts* where there exists a hierarchical order between *Profit*, *Turnover* and *Costs*.

Each dimension is a composition of dimension levels, e. g. the dimension levels *Day*, *Month* and *Year* belong to the dimension *Time*. The hierarchical structure between these dimension levels is defined through a set of hierarchical assignments (*HierarchicalLevelAssignment*), e. g. the dimension levels for the dimension *Time* are in the hierarchical order *Day* $\rightarrow$ *Month* $\rightarrow$ *Year*. The instance of a dimension level is called dimension member, e. g. *June* is an instance of the dimension level *Month*. These dimension members are again in a hierarchical order (*HierarchicalMemberAssignment*). Each measure, i. e. cell data is referenced by a set of dimension members. Furthermore our system manages transformation functions and structure version as detailed above. Of course, the shown meta-model also consists of some integrity constraints which can not be discussed in this work for sake of limited space.

# 3 Querying a Temporal Data Warehouse

In [4] we briefly discussed how a user can state queries in our temporal data warehouse model. The basic idea is that he/she first specifies a certain base structure. This base structure version determines which structure has to be used for the analysis. In most cases this will be the current structure version. However, in some cases it will be of interest to use an "older" structure version. Suppose a temporal data warehouse that holds measures since 1988 about all countries of the European Union. Within this temporal data warehouse two modifications of structure happened: the re-unification of Germany in 1990 and the participation of Sweden, Austria and Finland in 1995. Therefore, we can distinguish three structure versions that are valid for the time periods as shown in the following table ($T_s$ is the beginning of the valid time and $T_e$ is the end of the valid time of the corresponding structure version):

| Structure Version | $T_s$ | $T_e$ |
|---|---|---|
| $SV_1$ | 1988 | 1989 |
| $SV_2$ | 1990 | 1994 |
| $SV_3$ | 1995 | $\infty$ |

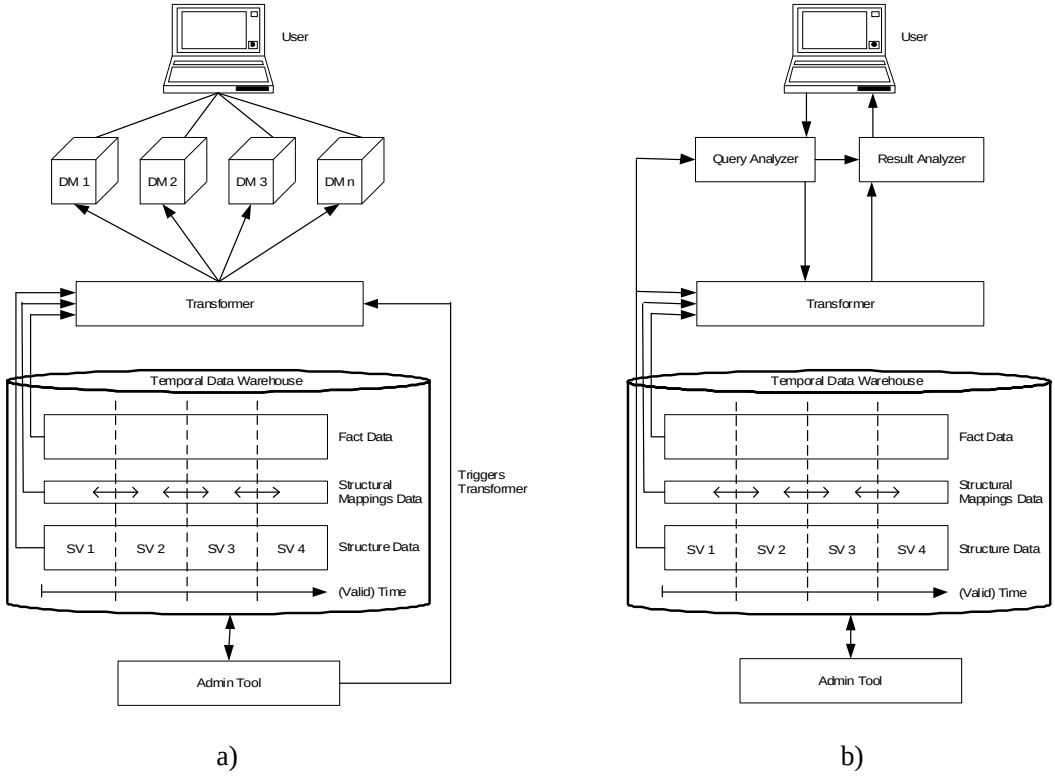


Figure 3: Architectures of a) the Indirect Approach and b) the Direct Approach

We might assume the user chooses $SV_3$ as his or her base structure version and requests data for 1995 and 1994 for his analysis. In this case the system needs functions to map data which is valid for the structure version $SV_2$ into the structure version $SV_3$. The same analysis however could also be made with $SV_2$ as base structure version. For this query the system needs functions to map data from $SV_3$ to $SV_2$.

Our system should not only be able to correctly answer queries spanning multiple periods and perhaps different versions of dimension data, but also to inform the user what kind of structural modifications took place. E. g. , if a user states the query "return the number of inhabitants for Germany for 1998, 1999 and 1990" we could inform the user that a modification of the structure for the dimension *State* happened in 1990, namely the re-unification of Germany. Depending on the chosen base structure version the result of the query could look like shown in Fig. 2.

We are currently working on two different approaches to accomplish this. Both approaches have three parts in common:

**Admin Tool** : The Admin Tool allows an administrator to import new cell data into the temporal data warehouse and, of course, to perform modifications of the multidimensional structure. The Admin Tool is implemented with the Java programming language.

**Temporal Data Warehouse** : The Temporal Data Warehouse holds the required information about structure versions, cell data and transformation functions as described in Sect. 2. We use Oracle 8. 1 as basis for our temporal data warehouse.

**Transformer** : The Transformer transforms all required cell values from all required structure versions into the chosen base structure version by using the defined transformation functions, i. e. , by

using all necessary *MapF* functions. The Transformer is implemented with the Java programming language.

### 3.1 Indirect Approach

As a first step we will implement the indirect approach. This approach is easier to implement as the direct approach due to the fact that it requires no implementation of an front end for analyses.

The main idea of the indirect approach is, as shown in Fig. 3 a), that the Transformer generates one data mart for each structure version needed by the user. In most cases, this will only be the actual structure version. Each data mart consists of all fact data that are valid for the same time interval as the corresponding structure version plus it consists of all fact data that could be transformed by the defined *MapF* functions from all other structure versions.

Therefore, the user defines his/her base structure version by selecting a specific data mart.

We call this approach indirect because the Transformer is triggered by the Admin Tool and not (directly) by the user. Or, in other words, the Transformer starts to generate data marts only after a new structure version has been generated or new measures have been imported. In both cases the Transformer has to recalculate all existing data marts in order to return consistent information.

We are implementing this approach with Oracle 8.1i as basis for our temporal data warehouse and Hyperion Essbase (Release 6) as front end that holds our data marts.

As we use a standard OLAP database for each data mart, the main advantage of the indirect approach is that each data mart offers the whole OLAP functionality, e.g., drill-down, roll-up, slice, dice, etc.

### 3.2 Direct Approach

The indirect approach has its limitations. The main drawback is that beside of giving the user a tool to state queries even after structural changes we would also like to inform the user about what kind of structural changes had an impact on the stated query. Hence, we have to enrich the result of a query with some query information as shown in Fig. 2.

The architecture of the direct approach consists of five parts as shown in Fig. 3 b):

**Query Analyzer** : The Query Analyzer takes a query stated by the user as input and analyzes which data out of which structure version is necessary to answer the query. The result of this analysis is passed to the Transformer and to the Result Analyzer.

**Transformer** : The Transformer works as described in Sect. 3. In contrast to the indirect approach, the Transformer is triggered by the user or, in other words, for each stated query the Transformer transforms all required cell values to answer the query.

**Result Analyzer** : The Result Analyzer takes its input from the Query Analyzer and from the Transformer. It enriches the result of the Transformer with further user information, i. e. , with information about what structural modifications had an impact on the stated query. The Result Analyzer is a subject of ongoing research.

**Temporal Data Warehouse & Admin Tool**: Both as described in Sect. 3.

We expect that the model we prosed for the direct approach improves not only the correctness of OLAP queries after modifications of the multidimensional structure, but also the interpretation of answers to OLAP queries.

## 4 Conclusion

We presented a sketch of a temporal data warehouse model designed to represent changes in the instances of dimension data of data warehouses, by introducing temporal extension, structure-versioning and transformation functions.

Furthermore, we extended this model to represent not only modifications on the instance level but also on the schema level. Thus, our system is able to return correct results to OLAP queries even after dimension levels have changed (e.g., if the dimension level *Quarter* is inserted between *Month* and *Year*) or a whole dimension changed (e.g., if a new dimension *Branches* becomes part of the schema).

We propose two architectures to implement this temporal data warehouse model. Both have an underlying temporal data warehouse - they differ in the way of answering queries. The *direct approach* uses a query analyzer and a result analyzer to enrich the result of each stated query with information about the impact of structural modifications on the query. The *indirect approach* generates a data mart for a particular structure version and transforms the data from all other structure versions into this data mart using the given specification of transformation functions. While the *direct approach* offers greater flexibility, the *indirect approach* is superior in terms of response time, once the data mart has been generated.

Further research will compare both approaches regarding performance, space requirements, etc. Another important research area will be to provide transformation functions not only between dimension members, but also between dimensions and dimension levels. Of course, introducing transformation functions on the schema level is far more complex than introducing transformation functions on the instance level of a temporal data warehouse.

We expect that the model we prosed here improves the correctness of interpretation of answers to OLAP queries and relieves the user from the need to have detailed knowledge about the change history of dimension data. In particular, our approach provides for multi-period comparisons of facts which currently requires stability in dimension data.

# References


[1] C. Adamson and M. Venerable. *Data Warehousing Design Solutions*. Wiley, New York, 1 edition, 1998.

[2] M. Blaschka, C. Sapia, and G. Höfling. On Schema Evolution in Multidimensional Databases. In *Proc. of the DaWak99 Conference*, Florence, Italy, 1999.

[3] P. Chamoni and S. Stock. Temporal Structures in Data Warehousing. In *Data Warehousing and Knowledge Discovery (DaWaK) 1999*, pages 353–358, Italy, 1999.

[4] J. Eder and C. Koncilia. Changes of Dimension Data in Temporal Data Warehouses. In *Proc. of the DaWak 2001 Conference*, Munich, Germany, 2001.

[5] O. Etzion, S. Jajodia, and S. Sripada, editors. *Temporal Databases: Research and Practise* Number LNCS 1399. Springer-Verlag, 1998.

[6] W. Inmon. *Building the Data Warehouse*. John Wiley and Sons, New York, 2 edition, 1996.

[7] C. S. Jensen and C. E. Dyreson, editors. *A consensus Glossary of Temporal Database Concepts - Feb. 1998 Version*, pages 367–405. Springer-Verlag, 1998. in [EJS98].

[8] W. Martin, editor. *Data Warehousing - Data Mining - OLAP*. Thomson Publishing, Bonn, 1 edition, 1998.

[9] A. Mendelzon and A. Vaisman. Temporal Queries in OLAP. In *Proc. of the 26th VLDB Conference, Egypt*, 2000.

[10] SAP America, Inc. and SAP AG. Data Modelling with BW - ASAP for BW Accelerator. 1998. White Paper: URL: http://www.sap.com.