A Bi-Temporal Data Warehouse Model

Christian Koncilia

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

Abstract. In data warehouses measures are analyzed along different dimensions. Although the structures of these dimensions change over time, data warehouse tools currently in use are not able to deal with these modifications in a sophisticated way. In this paper we present a bi-temporal extension of the COMET metamodel. This extension enables us to represent not only the valid time of structural modifications, but also the transaction time.

1 Introduction and Motivation

Data warehouses provide sophisticated features for aggregating, analyzing, and comparing data to support decision making in companies.

The most popular architecture for data warehouses are multidimensional data cubes, where transaction data (called cells, fact data or measures) are described in terms of master data (also called dimension members). Master data is hierarchically organized in dimensions, where the facts of the upper levels are computed from the facts of the lower levels by some consolidation functions.

Available OLAP (On-Line Analytical Processing) systems are therefore able to deal with changing measures, e. g. , changing profit or turnover. They are, however, 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.

In [EKM02] we presented our COMET approach which allows to represent changes of transaction data and structure data. This has been done by introducing a model that stores information about the valid time of data elements. *Valid time* defines the time, in which a fact is true in the real world [EJS98].

However, as OLAP systems are mainly used as decision support systems it would sometimes be helpful to know which knowledge (i.e., which data) was stored in the data warehouse at a given point in time. This would enable us, to reconstruct the data that led to a particular decision.

Transaction time represents the time in which a fact was logically present in the database [EJS98]. Hence, it stores the time point, at which a fact was inserted into the database, and at which it was (logically) deleted from the database.

In this paper, we will present a bi-temporal extension of the COMET metamodel, i.e., a metamodel that supports valid time and transaction time on both the schema and the instance level.

2 A Formal Model for Bi-Temporal Multidimensional Systems

Basically, a multidimensional view on data, e.g. an OLAP cube or a data warehouse, consists of a set of dimensions. Typical examples of dimensions frequently found in multidimensional databases are *Time*, *Facts* or *Products*.

The structure of each dimension is defined by a set of categories, e.g., the dimension *Time* could consist of the categories *Year*, *Quarter* and *Month* that are in the hierarchical relation $Year \rightarrow Quarter \rightarrow Month$, where for example $Year \rightarrow Quarter$ means that a quarter rolls-up to a year.

Each category consists of a set of dimension members. Dimension members define the instances of a data warehouse schema. For instance, *January*, *February* and *March* are dimension members assigned to the category *Month*.

We will now give a formal description of a bi-temporal multidimensional system. The features of this model comprise:

- **Bi-Temporal Model**: The model supports valid time and transaction time in order to keep track of the history of modifications on both, the instance and the schema level of a data warehouse.
- **Fact-Constellation Schemas**: The model allows to build *multi-cube data warehouses*. These cubes may share different dimensions or only parts of different dimensions. Thus, our model enables the administrator to create Fact-Constellation schemas [AV98].
- **Proportional Aggregation**: Furthermore, the model supports *correct aggregation* even if dimension members are not disjoint. This means that one dimension member may belong to several "parents". For instance, the calendar week number 5 (which is a dimension member of the category *Weeks*) of 2002 belongs to the months January and February where four days or 4/7 belong to January and three days or 3/7 belong to February.
- **Generic Dimensionality**: Our model fulfills E.F. Codd's sixth OLAP rule of "*Generic Dimensionality*" [CCS93]. Codd claims that each dimension must be equivalent in both its structure and operational capabilities. For instance, we treat the dimension *Time*, which is usually a part of a data warehouse, like any other dimension. Furthermore, we represent the facts of a data warehouse as a dimension *Facts*. This allows to apply the whole functionality of our approach even to the dimension *Time* or *Facts*.

In the following paragraphs, $[VT_s, VT_e]$ is always used to represent the valid time of an object (e.g., to represent the valid time of a dimension, a category or a dimension member) where VT_s is the beginning of the valid time, VT_e is the end of the valid time and $VT_e \geq VT_s$. $[TT_s, TT_e]$ is used to represent the transaction time interval of an object where TT_s is the beginning of the transaction time, TT_e is the end of the transaction time and $TT_e \geq TT_s$.

The schema of our temporal data warehouse approach is defined by:

i.) A number of dimensions J.

ii.) A set of dimensions $\mathbb{D} = \{D_1, ..., D_J\}$ where $D_i =< ID, D_{Key}, [VT_s, VT_e], [TT_s, TT_e] >$. ID is a unique identifier of the dimension. D_{Key} is a user defined key (e. g. , the name of the dimension) which is unique within the data warehouse for each timepoint $VT_s \leq T \leq VT_e$.

iii.) A number of categories K.

iv.) A set of categories $\mathbb{C} = \{C_1, ..., C_K\}$ where $C_i =< ID, C_{Key}, [VT_s, VT_e], [TT_s, TT_e] >$. ID is a unique identifier of the category. C_{Key} is a user defined key (e. g. , the name of the category) which is unique within the data warehouse for each timepoint $VT_s \leq T \leq VT_e$.

v.) A set of assignments between dimensions and categories $\mathbb{A}_{DC} = \{A^1_{DC}, ..., A^N_{DC}\}$, where $A^i_{DC} =< D.ID, C.ID, [VT_s, VT_e], [TT_s, TT_e] >$. $D.ID$ and $C.ID$ represents the identifier of the corresponding dimension, i.e., of the corresponding category.

vi.) A set of hierarchical category assignments $\mathbb{HC} = \{HC_1, ..., HC_O\}$ where $HC_i =< ID, C.ID_C, C.ID_P >$. ID is a unique identifier of the hierarchical category assignment. $C.ID_C$ is the identifier of a category, $C.ID_P$ is the category identifier of the parent of $C.ID_C$ or $\emptyset$ if the category is a top-level category.

vii.) A number of user defined attributes L.

viii.) A set of user defined attributes (UDAs) $\mathbb{U} = \{U_1, ..., U_L\}$ where $U_i = < ID, U_{Key}, U_{Type}, [VT_s, VT_e], [TT_s, TT_e], I_{ID} >$. ID is a unique identifier of the UDA. U_{Key} is a user defined key (e. g. , the name of the UDA) which is unique within the data warehouse for each timepoint $VT_s \leq T \leq VT_e$. U_{Type} defines the data type of the corresponding UDA. I_{ID} is the identifier of a dimension or a category.

This identifier I_{ID} defines the set of dimension members for which the corresponding user defined attribute is applicable: Each dimension member assigned to the corresponding dimension and/or to the corresponding category may have an assigned value representing the extension of the corresponding user defined attribute. Hence, this set of user defined attributes defines the name and type of all user defined attributes within our data warehouse.

ix.) A number of cubes I.

x.) A set of cubes $\mathbb{B} = \{B_1, ..., B_I\}$ where $B_i =< ID, B_{Key}, \mathcal{S}, [VT_s, VT_e], [TT_s, TT_e] >$. ID is a unique identifier of the cube (similar to $O_{id's}$ in object-oriented database systems). B_{Key} is a user defined key (e. g. , the name of the cube) which is unique within the data warehouse for each timepoint $VT_s \leq T \leq VT_e$.

$\mathcal{S}$ represents the schema of the cube. The n-tuple $\mathcal{S}$ consists of all dimensions and hierarchical category assignments that are a part of this cube. Therefore $\mathcal{S}$ is defined as $\mathcal{S} = (\mathbb{D}, \mathbb{A})$ where $\mathbb{D} = \{D_1.ID, ..., D_N.ID\}$ and $\mathbb{A} = \{HC_1.ID, ..., HC_M.ID\}$.

The instances of our temporal data warehouse approach are defined by:

i.) A number of dimension members P.

ii.) A set of dimension members $\mathbb{M} = \{M_1, ..., M_P\}$ where $M_i =< ID, M_{Key}, \mathbb{UV}, \mathbb{CA}, [VT_s, VT_e], [TT_s, TT_e] >$. ID is a unique identifier of the dimension member. M_{Key} is a user defined key (e. g. , the name of the dimension member) which is unique within the data warehouse for each timepoint $VT_s \leq T \leq VT_e$.
The set $\mathbb{CA}$ represents the set of categories to which the corresponding dimension member is assigned. If a dimension member is assigned to more than one categories, and if these categories belong to the same cube for at least one point in time then we call it a *shared member*.
$\mathbb{UV}$ is a set of tuples that consists of all user defined attribute values. It is defined as $\mathbb{UV} = \{(U.ID_1, V_1), ..., (U.ID_N, V_N)\}$ where $U.ID_i$ is the identifier of the corresponding user defined attribute, i.e. its definition, and V_i is the value, i.e. its extension.

iii.) A set of hierarchical member assignments $\mathbb{HM} = \{HM_1, ..., HM_O\}$ where $HM_i =< ID, M.ID_C, M.ID_P >$. ID is a unique identifier of the hierarchical member assignment. $M.ID_C$ is the identifier of a dimension member, $M.ID_P$ is the dimension member identifier of the parent of $M.ID_C$ or $\emptyset$ if the dimension member is at the top-level.

iv.) A function $cval : (M_{D_1}, ..., M_{D_N}) \rightarrow value$ which uniquely assigns a value to each vector $(M_{D_1}, ..., M_{D_N})$ where $(M_{D_1}, ..., M_{D_N}) \in \mathcal{M}_{D_1} \times ... \times \mathcal{M}_{D_N}$. The domain of this function is the set of all cell references. The range of this function are all cell values, i.e., measures of a cube.

3 Conclusion

In this paper, we presented a bi-temporal extension of the COMET metamodel. This extension supports valid time and transaction time on both the schema and the instance level. Hence, it allows to represent changes of transaction data and structure data. Furthermore, it enables us to represent knowledge about which data was stored in the data warehouse at a given point in time, and thus led to right or wrong decisions.

References

[AV98] C. Adamson and M. Venerable. *Data Warehousing Design Solutions*. Wiley, New York, 1 edition, 1998.

[CCS93] E. Codd, S. Codd, and C. Smalley. *Providing OLAP to User-Analysts: An IT Mandate*. Hyperion Solutions Corporation, California, 1993. URL: http://www.hyperion.com/.

[EJS98] O. Etzion, S. Jajodia, and S. Sripada, editors. *Temporal Databases: Research and Practise*. Springer-Verlag (LNCS 1399), 1998.

[EKM02] J. Eder, C. Koncilia, and T. Morzy. The COMET Metamodel for Temporal Data Warehouses. In *Proceedings of the 14th International Conference on Advanced Information Systems Engineering (CAISE'02)*, Toronto, Canada, 2002. Springer Verlag (LNCS 2348).