# Well-formed Data Warehouse Structures

Michel Schneider

LIMOS, Blaise Pascal University, Complexe des Cézeaux
63173 Aubière, France
schneider@isima.fr

**Abstract:** Two main problems arise in modelling data warehouse structures. The first consists in establishing an adequate representation of dimensions in order to facilitate and to control the analysis operations. The second relates to the modelling of various types of architecture. Research work dedicated to the first problem has been conducted, and adequate solutions have been proposed. The second problem has not received so much attention. However, there is a need to apprehend complex structures interconnecting dimensions and facts in various ways. In this paper, we propose a model through which dimensions at different levels can be shared between different facts and various relationships between these facts can be described. Using this model, we then define the notion of well-formed warehouse structures.

**Keywords:** data warehouses, dimension modelling, architecture modelling, graph representation.

## 1 Introduction

The tasks of design and implementation of a data warehouse cannot be achieved without adequate modelling of dimensions and facts.

Concerning the modelling of dimensions, the objective is to find an organization which corresponds to the analysis operations and which provides strict control over how the aggregations can be made. In particular it is important to avoid double-counting or summation of non-additive data. Many studies have been devoted to this problem. Most recommend organizing the members of a given dimension into hierarchies with which the aggregation paths can be explicitly defined. In [12], hierarchies are defined by means of a containment function. In [7], the organization of a dimension results from the functional dependences which exist between its members, and a multi-dimensional normal form is defined. In [6], the functional dependences are also used to design the dimensions and to relate facts to dimensions. In [1], relationships between levels in a hierarchy are apprehended through the Part-Whole semantics. In [14], dimensions are organized around the notion of a dimension path which is a set of drilling relationships. The model is centred on a parent-child (one to many) relationship type. A drilling relationship describes how the members of a children level can be grouped into sets that correspond to members of the parent level. In [13], a dimension is viewed as a lattice and two functions anc and desc are used to perform the roll up and the drill down operations. [11] proposes an extended multidimensional data model which is also based on a lattice structure, and which

provides non-strict hierarchies (i.e. many to many relationships between the different levels in a dimension).

Modelling of facts and their relationships have not received so much attention. Facts are generally considered in a simple fashion which consists in relating a fact with the roots of the dimensions. However, there is a need for considering more sophisticated structures where the same set of dimensions are connected to different fact types and where several fact types are inter-connected. The model described in [11] permits some possibilities in this direction.

Apart from these studies it is important to note various propositions [2, 3, 5, 10] for cubic models where the primary objective is the definition of an algebra for multidimensional analysis. Others works must also be mentioned. In [4], a solution is proposed to derive multidimensional structures from E/R schemas. In [8] are established necessary conditions for summarizability in multidimensional structures. In [9] is suggested a normalised relational model for data warehouses.

Our objective in this paper is to propose a model which can be used to apprehend the sharing of dimensions in various ways and to describe different relationships between fact types. Using this model, we will also define the notion of well-formed warehouse structures. Such structures have desirable properties for applications. We suggest a graph representation for such structures which can help the users in designing and requesting a data warehouse.

The paper is organized as follows: sections 2 and 3 respectively present the modelling of facts and the modelling of dimensions; section 4 presents the typical structures we want to model, and defines the notion of well-formed structures; section 5 shows the ability of our model to describe realistic cases; section 6 discusses relational mapping.

## 2 Modelling facts

**Fact type.** A fact is used to record measures or states concerning an event or a situation. Measures and states can be analysed through different criteria organized in dimensions.

A fact type is a structure

fact_name[(fact_key), (list_of_reference_attributes), (list_of_fact_attributes)]

where

- fact_name is the name of the type;
- fact_key is a list of attribute names; the concatenation of these attributes identifies each instance of the type;
- list_of_reference_attributes is a list of attribute names; each attribute references a member in a dimension or another fact instance;
- list_of_fact_attributes is a list of attribute names; each attribute is a measure for the fact.

The set RD of referenced dimensions comprises the dimensions which are directly referenced through the list_of_reference_attributes, but also the dimensions which are indirectly referenced through other facts.

Each fact attribute can be analysed along each of the dimensions of RD. Analysis is achieved through the computing of aggregate functions on the values of this attribute. As in [6], we distinguish three levels of summarizability for a fact attribute

relative to a dimension: S which means that SUM and all the other aggregate functions are possible; O (others) which means that all the aggregate functions are possible except SUM; C which means that only the COUNT function is possible. Indicators of summarizability are placed in parentheses after the name of each fact attribute.

There may be no fact attribute; in this case a fact records the occurrence of an event or a situation. In such cases analysis consists in counting occurrences satisfying a certain number of conditions.

For the needs of an application, it is possible to introduce different fact types sharing certain dimensions and having references between them.

A fact attribute is said to be basic if it cannot be derived from the other fact attributes.

Two dimensions are independent if there is no relationship between a member of the first and a member of the second.

A dimension is degenerated in a fact type if its reference attribute is replaced by a value attribute. In other words the analysis is achieved by direct use of the values of this attribute.

**Definition 1 (well-formed fact type).** A fact type is well-formed if there is at least one dimension (possibly degenerated), if the referenced dimensions are independent of each other, if the fact attributes are all basic, and if analysis of each fact attribute is possible through any of the referenced dimensions.

**Example 1.** Let us consider the following fact type for memorizing the sales in a set of stores.

Sales[(ticket_number, product_key), (time_key, product_key, store_key),
(price_per_unit(O,O,O), quantity(O,O,O))]

Sales is a well-formed fact type. The key is (ticket_number, product_key). This means that there is an instance of Sales for each different product of a ticket. There are three dimension references : time_key, product_key, store_key. There are two fact attributes : price_per_unit, quantity. Indicators of summarizability are all equal to O.

## 3 Modelling dimensions

**Member of a dimension.** The different criteria which are needed to conduct analysis along a dimension are introduced through members. A member is a specific attribute (or a group of attributes as we will see later) taking its values on a well defined domain. For example, the dimension TIME can include members such as DAY, MONTH, YEAR, … . Analysing a fact attribute A along a member M means that we are interested in computing aggregate functions on the values of A for any grouping defined by the values of M. In the paper we will also use the notation $M_{ij}$ for the j-th member of i-th dimension.

**Organization of members.** Members of a dimension are generally organized in a hierarchy which is a conceptual representation of the hierarchies of their occurrences. Hierarchy in dimensions is a very useful concept that can be used to impose constraints on member values and to guide the analysis. Hierarchies of occurrences

result from various relationships which can exist in the real world: categorization, membership of a subset, mereology. Figure 1 illustrates typical situations which can occur. Cases (a) and (b) represents the same members but organized differently. In case (a) there are hierarchical relationships between time_key and month and between month and year. Time_key, for example, is a date which encodes the day, the month and the year; the month has values such as February/2000 which identifies each month from all the months of the total period. In these conditions, the amount of sales for all the months of all the years, is obtained with a group_by just on the month. In case (b) month and year are both hierarchically dependent of time_key but are independent of each other (month, for example, is a value such as January identifying a month independently from the value of year). The expression of the previous request would now involve a group_by on month+year. Case (c) represents a hierarchy where the two paths share the same root type (cust_key) and the same leaf type (region). This configuration has precise semantics: for a given occurrence of cust_key, whether the town path or the demozone path is used, one always obtains the same occurrence of region.

We will model these different cases according to a hierarchical relationship (HR) which links a child member $M_{ij}$ (i.e. town) to a parent member Mik (i.e. region) and we will use the notation $M_{ij} \rightarrow M_{ik}$. For the following we consider only situations where a child occurrence is linked to a unique parent occurrence in a type. However, a child occurrence, as in case (b) or (c), can have several parent occurrences but each of different types. We will also suppose that HR is reflexive, antisymmetric and transitive. This kind of relationship covers the great majority of real situations [1]. Existence of this HR is very important since it means that the members of a dimension can be organized into levels and correct aggregation of fact attribute values along levels can be guaranteed.

**Definition 2 (cover graph of a dimension).** A cover graph of a dimension is a minimal cover for the directed graph defined by the HR between the member attributes of this dimension.

**Definition 3 (well-formed dimension).** A dimension is well-formed relative to a cover graph when this graph has a unique connected component and is acyclic.

**Example 2.** Let us consider again the dimensions illustrated in figure 1. They are all well-formed since their cover graphs have a unique component and are acyclic. Restricting the cover graph to a unique component is very important in practice. If the graph comprises, for example, two components, the dimension must be divided into two distinct dimensions.

**Aggregation levels in a well-formed dimension.** Since the cover graph of a well-formed dimension is acyclic, it is possible to distribute its members into levels. Each level represents a level of aggregation. Each time we follow a directed edge, the level increases (by one or more depending on the used path). This action corresponds to a ROLLUP operation (corresponding to the semantics of the HR) and the opposite operation to a DRILL DOWN. Starting from the reference to a dimension D in a fact

type F, we can then roll up in the hierarchy of dimension D by following a path of the cover graph of D.

**Entries and roots in a dimension.** Each member of a dimension can be an entry for this dimension i.e. can be referenced from a fact type. This possibility is very important since it means that dimensions between several fact types can be shared in various ways. In particular, it is possible to reference a dimension at different levels of granularity. A dimension root represents a standard entry. For the three dimensions in figure 1, there is a single root. However, definition 3 authorizes several roots.

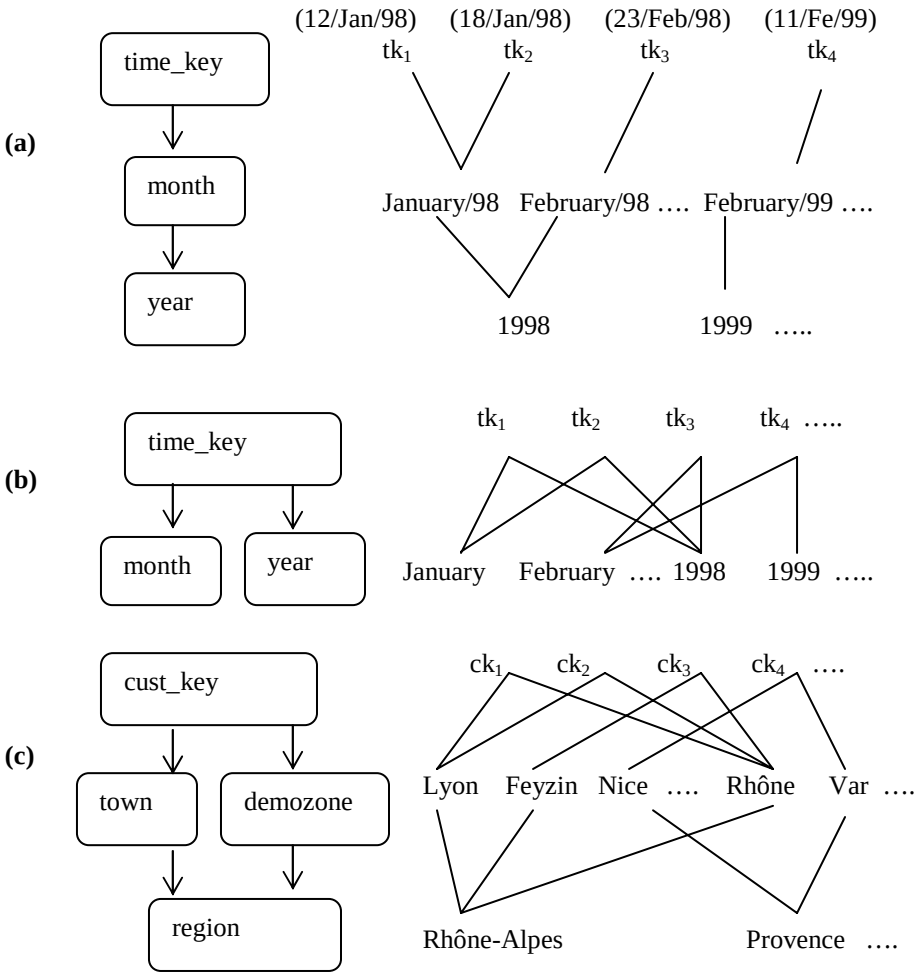


**Fig. 1.** Different hierarchies in dimensions

**Property attributes in a dimension.** As in other studies [6], we consider property attributes in a dimension which is used to describe the members. A property attribute is linked to its member through a functional dependence, but does not introduce a new member and a new level of aggregation. For example the member *town* in the *customer* dimension may have property attributes such as population, administrative position, … . Such attributes can be used in the selection predicates of requests to filter certain groups.

**Member type.** We now define the notion of member type, which incorporates the different features presented above.

A member type is a structure:

member_name[(member_key), dimension_name, (list_of_reference_attributes), (list_of_property_attributes)]

where

- member_name is the name of the type;
- member_key is a list of attribute names; the concatenation of these attributes identifies each instance of the type;
- dimension_name is the name of the dimension to which the type belongs;
- list_of_reference_attributes is a list of attribute names where each attribute is a reference to the successors of the member instance in the cover graph of the dimension;
- list_of_property_attributes is a list of attribute names where each attribute is a property for the member.

Only the member_key and the dimension_name are mandatory.

**Example 3.** Using this model, the representation of the members of dimension *customer* in figure 1 is the following:

cust_root[(cust_key), customer, (town_name, zone_name), ()]
town[(town_name), customer, (region_name), (population, airport_type)]
demozone[(zone_name), customer, (region_name), ()]
region[(region_name), customer, (), (population)]
Here is an occurrence of town :
town [(Lyon), customer, (Rhône-Alpes), (1200000)]

**Similarities between fact and member.** A fact type has a very similar structure to that of a member type. Moreover, property attributes of a member can be seen as fact attributes and can be analysed along the successors of this member acting as roots of partial dimensions. For example, in the case of the dimension *customer* (cf example 3), the attribute *population* of *town* can be analysed along the successor *region*: one can calculate aggregates on *population* with groupings on *region_name*. Note that the result is not necessarily equal to the value of *population* in *region* since all the towns of the region are not necessarily stored in the data warehouse.

## 4 Well-formed structures

In this section we explain how the fact types and the member types can be interconnected in order to model various warehouse structures.

First, a fact can directly reference any member of a dimension. Usually a dimension is referenced through one of its roots (as we saw above, a dimension can have several roots). But it is also interesting and useful to have references to members other than the roots. This means that a dimension can be used by different facts with different granularities. For example, a fact can directly reference *town* in the *customer* dimension and another can directly reference *region* in the same dimension. This second reference corresponds to a coarser granule of analysis than the first.

Moreover, a fact $F_1$ can reference any other fact $F_2$. This type of reference is necessary to model certain situations (see section 5). This means that a fact attribute of $F_1$ can be analysed by using the key of $F_2$ (acting as the grouping attribute of a normal member) and also by using the dimensions referenced by $F_2$.

To formalise the interconnection between facts and dimensions, we thus suggest extending the HR relationship of section 3 to the representation of the associations between fact types and the associations between fact types and member types. We impose the same properties (reflexivity, anti-symmetry, transitivity). We also forbid cyclic interconnections. This gives a very uniform model since fact types and member types are considered equally. To maintain a traditional vision of the data warehouses, we also ensure that the members of a dimension cannot reference facts.

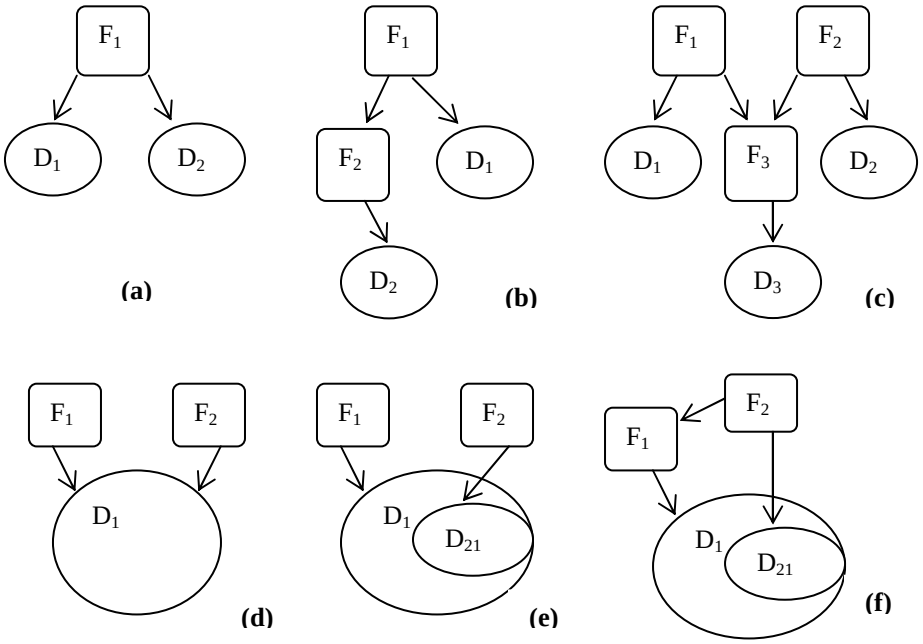


**Fig. 2.** Typical warehouse structures

Figure 2 illustrates the typical structures we want to model. Case (a) corresponds to the simple case, also known as star structure, where there is a unique fact type $F_1$ and several separate dimensions $D_1$, $D_2$, … . Cases (b) and (c) correspond to the notion of facts of fact. Cases (d), (e) and (f) correspond to the sharing of the same

dimension. In case (f) there can be two different paths starting at $F_2$ and reaching the same member M of the sub-dimension $D_{21}$. So analysis using these two paths cannot give the same results when reaching M. To pose this problem we introduce the DWG and the path coherence constraint.

**Data Warehouse Graph (DWG).** To represent data warehouse structures, we suggest using a graph representation called DWG (data warehouse graph). It consists in representing each type (fact type or member type) by a node containing the main information about this type, and representing each reference by a directed edge.

**Path coherence constraint.** Suppose that in the DWG graph, there are two different paths $P_1$ and $P_2$ starting from the same fact type F, and reaching the same member type M. We can analyse instances of F by using $P_1$ or $P_2$. The path coherence constraint is satisfied if we obtain the same results when reaching M.

Note that this problem is the same as that discussed in section 3 concerning two different paths in the hierarchy of a dimension with the same starting node and the same ending node (Figure 1 (c)).

We are now able to introduce the notion of well-formed structures.

**Definition 4 (well-formed warehouse structures).** A warehouse structure is well-formed when the DWG is acyclic, fact types are well-formed, dimensions are well-formed, members do not reference facts, and the path coherence constraint is satisfied for any couple of paths having the same starting node and the same ending node.

A well-formed warehouse structure can thus have several roots. The different paths from the roots can be always divided into two sub-paths: the first one with only fact nodes and the second one with only member nodes. So roots are fact types.

## 5 Illustrating the modelling of realistic cases with well-formed structures

In this section we show how different realistic cases can be described with well-formed structures.

**Star and snowflake structures.** We have a star or snowflake structure when :

- there is a unique fact type;
- each dimension has a unique root;
- each reference in the fact type points towards the root of a dimension.

Our model does not differentiate star structures from snowflake structures. The difference will appear with the mapping towards the relational model (see section 6). The DWG of a star-snowflake structure is represented in figure 3. This representation is well-formed. Such a representation can be very useful to a user for formulating requests. Facts are clearly differentiated from members, reference to dimensions are shown explicitly, indicators of summarizability are outlined, dimension organization appears immediately.

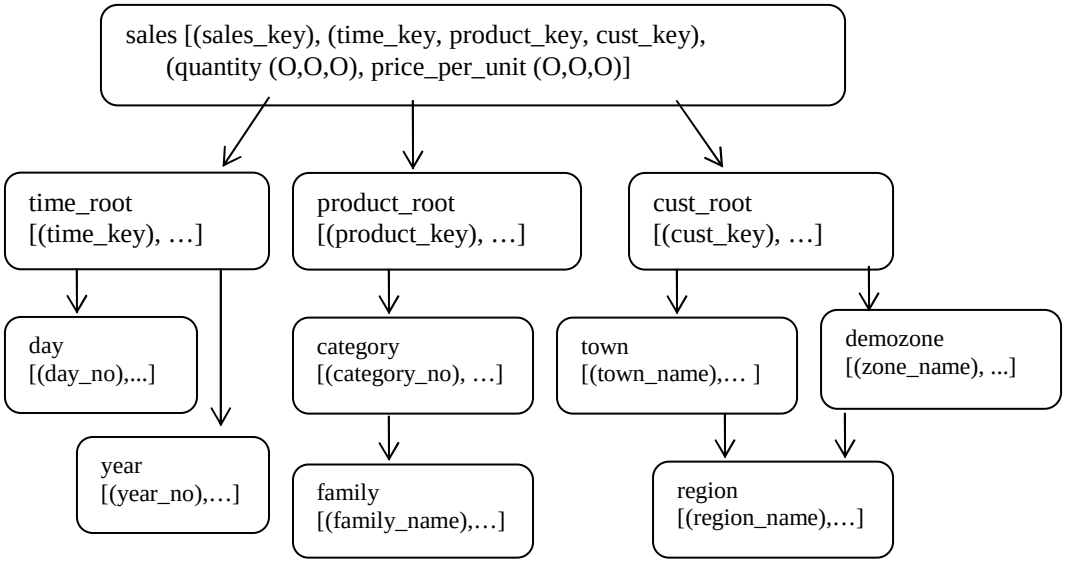


**Fig. 3.** The DWG of a star-snowflake structure

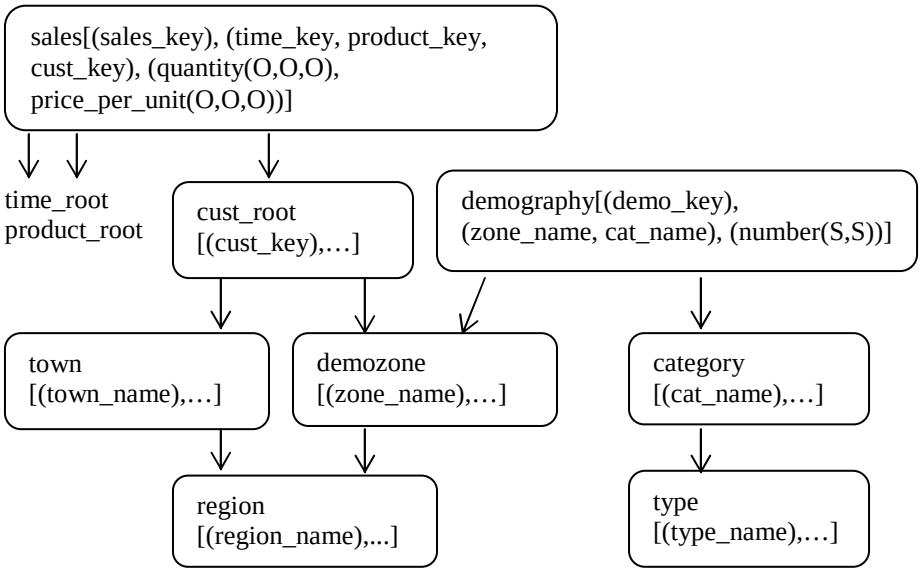


**Fig. 4.** The DWG for a constellation warehouse

**Sharing of a dimension (constellation structure).** The constellation structure appears when :

- there are at least two different fact types;
- two different fact types share the same dimension;
- a fact type does not reference another fact type.

Using the notion of DWG, figure 4 shows an example with the fact type *sales* from figure 3 and a new fact type *demography* memorizing demographic facts for a given category in a demographic zone. *Demography* has a reference to the member *demozone* of the dimension *customer*. So, the dimension *customer* is shared partly between the two fact types.

The DWG clearly shows how the two fact types can be exploited separately or simultaneously. We can explore the graph from one of its two roots and use it as a single rooted graph. We can also simultaneously exploit the two fact types. For example, to the node *demozone*, one can associate different aggregates from the *demography* occurrences and use them for the analysis of the *sales* facts, or vice-versa.

**Facts of fact.** Sometimes one fact type, called primary fact type, can be characterized by several other fact types, called secondary fact types. Let us consider for example the case of a medical operation. It is characterized by primary fact attributes such as quotation, duration, ... . Following this operation, a prescription including several drugs can be made out. Secondary fact attributes are associated to each drug such as the number of times it should be taken per day, or the number of days concerned by the prescription. It is not adequate to memorize these secondary facts in the primary type (there is a many-to-many association between operations and drugs). One solution consists in placing them in a secondary fact type refering to the primary fact type. Our model caters for the description of such a solution. It consists in specifying two different fact types *prescription* and *operation*, and installing in *prescription* a reference to *operation* (figure 5). It should also be noted that *prescription* has a normal reference to the root of the *drug* dimension. There is an instance of *prescription* for each different drug in a prescription.

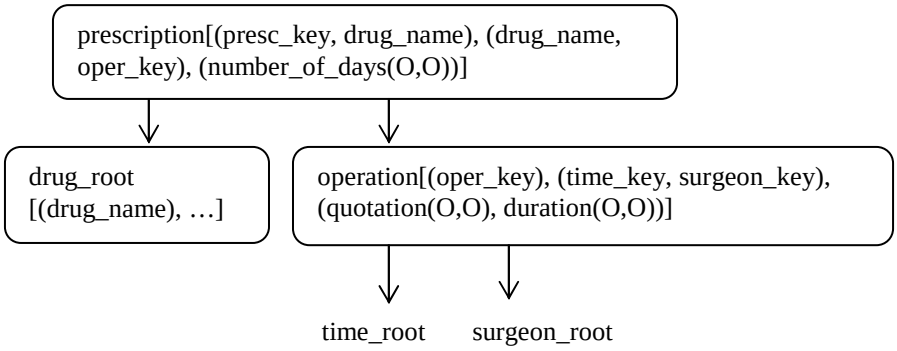


**Fig. 5.** Modelling facts of fact

For the secondary fact type, the primary fact type acts as a multi-dimension. So, all the dimensions of the primary type can be also used as dimensions of the secondary type. This clearly appears in the DWG of the global structure (figure 5). For example, quotation and duration can be analysed using the *time* and *surgeon* dimensions and *number_days* can be analysed using *drug* dimension but also *time* and *surgeon* dimensions. This last fact can also be analysed using *oper_key* alone acting as a degenerated dimension. For example we can calculate the average number of drugs per operation.

## 6 Mapping to the relational model

One way to implement warehouses is to use relational DBMS. So it is necessary to be able to map our well-formed structures in accordance with the relational structure. In this section we provide a certain number of guidelines for this mapping.

**Relational mapping with split dimensions.** This solution, which is straightforward, consists in mapping each type $P_i$ (fact or member) into a table $T_i$. The key of $P_i$ becomes the primary key of $T_i$. References between types are implemented via foreign keys. This solution offers a simple way to memorize precalculated aggregates by adding supplementary attributes in member types. Its drawback is well-known: navigating through dimensions necessitates many joins which can burden the performances. For a structure like that described in figure 3, this solution leads to the relational snowflake warehouse structure.

**Relational mapping with regrouped dimensions.** This solution consists in mapping each fact type into a specific table and to group all the member types of a dimension in a unique table. This solution is only possible when the dimension has a unique root and when references are only made to this root. For a structure like that described in figure 3, this solution leads to the relational star warehouse structure. With this mapping the structure of the dimension hierarchies are embedded into the data.

**Hybrid relational mapping**. The previously described mapping is not possible when several entries are used in a dimension, whether these entries are roots or not. This is because each entry must be the key of a table in order to install the references correctly. The hybrid mapping thus consists in inserting each member entry into a specific table. Members which are accessible only from one entry can be stored in the table of this entry. Others must be stored in separate tables.

For the structure of figure 4, this mapping gives the following tables (primary keys are marked in bold, foreign keys in italics):

sales(**sales_key**, *time_key*, *product_key*, *cust_key*, quantity, price_per_unit)
demography(**demo_key**, *zone_name*, *cat_name*, number)
time(**time_key**, …)
product(**product_key**, …)
cust_root(**cust_key**,…, town_name,…, *region_name*, *zone_name*)
demozone(**zone_name**,…, *region_name*)
region (**region_name**,...)
category(**cat_name**,…, type_name,…)

Note that the member types *town* and *type* have been encapsulated into the tables *cust_root* and *category* respectively.

## 7 Conclusion

In this paper we propose a model which can describe various data warehouse structures. It extends existing models for sharing dimensions and for representing relationships between facts. It allows for different entries in a dimension corresponding to different granularities. A dimension can also have several roots corresponding to different views and uses. It is possible to apprehend the concept of facts of fact which is very frequently encountered in the real world. Using this model, we define the notion of well-formed warehouse structures which guarantees desirable properties.

We have also proposed the concept of Data Warehouse Graph (DWG) to represent well-formed structures. The DWG gathers the main information from the structure of the warehouse. It can be very useful to users for formulating requests. We believe that the DWG can be used as an efficient support for a graphical interface to manipulate multidimensional structures through a graphical language.

We have also shown how well-formed structures can be mapped to the relational model in different ways. To represent the reference we have used values of reference attributes. Instead, we can adopt an object-oriented model. Identifiers and references would then be represented through OIDs. This would make it possible to define mapping to the object relational model. It appears that this model has a natural place between the conceptual schema of the application and a relational implementation of the warehouse. It can thus serve as a helping support for the design of relational data warehouses.

For the future, we plan to incorporate the notion of generalization / specialization into our model. The notion of role for a dimension relative to a fact will be also useful.

## References

1. Abello, A., Samos, J., Saltor, F. : Understanding Analysis Dimensions in a Multidimensional Object-Oriented Model. Proc. of Intl Workshop on Design and Management of Data Warehouses (DMDW'2001), Interlaken, Switzerland, June 4, 2001.
2. Agrawal, R., Gupta, A., Sarawagi, S. : Modelling Multidimensional Databases. ICDE'97, 13th International Conference on Data Engineering, Birmingham, UK, pp. 232-243, April 7-11, 1997.
3. Datta, A., Thomas, H. : The Cube Data Model : A Conceptual Model and Algebra for on-line Analytical Processing in Data Warehouses. Decision Support Systems, pp. 289-301, 27(3), December 1999.
4. Golfarelli, M., Maio, D., Rizzi, S. : Conceptual Design of Data Warehouses from E/R Schemes. Proc. of the 32th HICSS, 1998.
5. Gyssens, M., Lakshmanan, V.S. : A Foundation for Multi-dimensional Databases. Proc. of the 23rd Conference on Very Large Databases, pp. 106-115, 1997.

6. Hùsemann, B., Lechtenbörger, J., Vossen, G. : Conceptual Data Warehouse Design. Proc. of Intl Workshop on Design and Management of Data Warehouses (DMDW'2000), Stockholm, Sweden, June 5-6, 2000.
7. Lehner, W., Albrecht, J., Wedekind, H. : Normal Forms for Multidimensional Data Bases. l0th Intl Conference on Scientific and Statistical Data Management (SSDBM'98), Capri, Italy, July 1-3, pp. 63-72, 1998.
8. Lenz, H.J., Shoshani, A. : Summarizability in OLAP and Statistical Data Bases. Proc. of the 9th SSDBM, pp. 132-143, 1997.
9. Levene, M., Loizou, G. : Why is the Star Schema a Good Data Warehouse Design. http://citeseer.ni.nec.com/457156.html, April 1999.
10. Li, C., Wang, X.S. : A Data Model for Supporting on-line Analytical Processing. Proc. of the Fifth International Conference on Information and Knowledge Management, pp. 81-88, 1996.
11. Pedersen, T.B., Jensen, C.S. : Multidimensional Data Modelling for Complex Data. In Proc. ICDE' 99, Intl Conference on Data Engineering, pp. 336-345, 1999.
12. Pourabbas, E., Rafanelli, M. : Characterization of Hierarchies and some Operators in OLAP Environment. DOLAP'99, Kansas City, USA, pp. 54-59, 1999.
13. Vassiliadis, P., Skiadopoulos, S. : Modelling and Optimisation Issues for Multidimensional Databases. Proc. of the 12th Intl Conference CAISE, Stockholm, Sweden, pp. 482-497, 2000.
14. Tsois, A., Karayannidis, N., Sellis, T. : MAC: Conceptual Data Modeling for OLAP. Proc. of the Intl Workshop on Design and Management of Data Warehouses (DMDW'2001), Interlaken, Switzerland, June 4, 2001.