Andreas Bauer, Wolfgang Hümmer, Wolfgang Lehner

An Alternative Relational OLAP Modeling Approach

Erstveröffentlichung in / First published in:

DaWaK: International Conference on Big Data Analytics and Knowledge Discovery. London, 04.-06.09.2000. Springer, S. 189-198. ISBN 978-3-540-44466-4.

DOI: http://dx.doi.org/10.1007/3-540-44466-1_19

Final edited form was published in "DaWaK: International Conference on Big Data Analytics and Knowledge Discovery. London 2000", S. 189-198, ISBN 978-3-540-44466-4
http://dx.doi.org/10.1007/3-540-44466-1_19

An Alternative Relational OLAP Modeling Approach

Andreas Bauer, Wolfgang Hümmer, Wolfgang Lehner
University of Erlangen-Nuremberg (Database Systems)
Martensstr. 3, Erlangen, 91058, Germany

EMail: {andreas.bauer, huemmer, lehner}@informatik.uni-erlangen.de

Abstract. Schema design is one of the fundamentals in database theory and practice as well. In this paper, we discuss the problem of locally valid dimensional attributes in a classification hierarchy of a typical OLAP scenario. In a first step, we show that the traditional *star and snowflake schema* approach is not feasible in this very natural case of a hierarchy. Therefore, we sketch two alternative modeling approaches resulting in practical solutions and a seamless extension of the traditional *star and snowflake schema* approach: In a pure relational approach, we replace each dimension table of a *star / snowflake schema* by a set of views directly reflecting the classification hierarchy. The second approach takes advantage of the object-relational extensions. Using object-relational techniques in the context for the relational representation of a multidimensional OLAP scenario is a novel approach and promises a clean and smooth schema design.

1 Introduction

In the last few years, *"Online Analytical Processing"* (OLAP, [CoCS93]) has become a major research area in the database community (special data models: [VaSe99]; SQL extensions: [GBLP96], [SQL99]). One consequence of the OLAP fever is the rejuvenation of the multidimensional data model. The *ROLAP* approach ("Relational OLAP") simulates the multidimensionality and performs data access on top of a relational database engine, thus using sophisticated relational base technology to handle, i.e. store and analyze the typical large data volumes of the underlying data warehouses. This approach however needs an adequate relational representation, which is typically a variation of a *star / snowflake schema*.

Based on the experiences from an industrial project, we have seen that the traditional modeling techniques for the relational based solution, *star* and *snowflake schema*, are not always adequate ([ALTK97]). Even considering multiple variations with regard to slowly changing dimensions, factless fact tables, etc. ([Inmo96]), we demonstrate a modeling problem which is not addressed adequately in literature. This paper reviews our schema design problems with the traditional techniques and proposes two more general and therefore alternative modeling approaches.

The key idea of the multidimensional model is that each dimension of a multidimensional data cube, e.g. products, shops, or time, can be seen as part of the primary key, defining the cartesian product with the elements of dimensions. Consequently, any combination of the composite primary key identifies exactly a single cell within the cube.

Final edited form was published in "DaWaK: International Conference on Big Data Analytics and Knowledge Discovery. London 2000", S. 189-198, ISBN 978-3-540-44466-4
http://dx.doi.org/10.1007/3-540-44466-1_19

Each cell may hold a numerical fact value (measure) or a NULL value if no such entry exists. As illustrated in figure 1, based on the dimensional elements, e.g. single articles in the product dimension, classifications can be defined to identify different classes C like product families, groups, or product areas. Each classification node at a specific classification level can be seen as an instance of a corresponding classification attribute (CA_i).

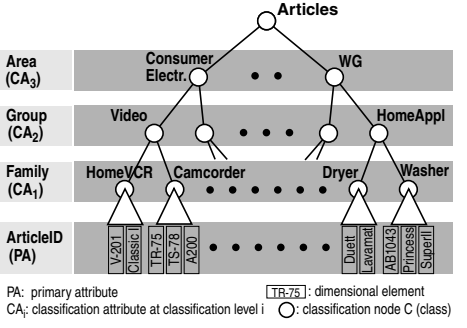

Fig. 1. Sample classification hierarchy

Additionally, dimensional attributes (DA_k) like brand, color, shoptype etc. can be used to enrich the multidimensional analysis process. As depicted in figure 2, these attributes, *characterizing* single dimensional elements, are standing orthogonal with regard to the classification hierarchy, *classifying* dimensional elements.

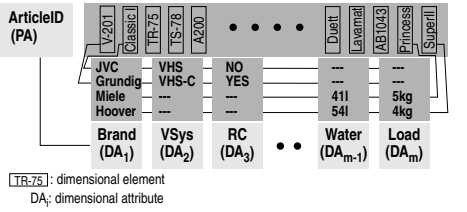

Fig. 2. Sample dimensional attributes

Thus, a typical question according to a multidimensional scenario could be as follows: *Give me the total sales of consumer electronics goods for Europe and the first quarter of 1997 by different brands and different shop types*.

It is worth mentioning here that such a structure with special (better: locally valid) dimensional attributes for different classes within the classification hierarchy reflects the basic idea of classification. What else should be the reason for classification? Classifications are generally used to hide specialities of subclasses and perform an abstraction process when going from subclasses to super-classes. We think that this idea should be adequately reflected in the relational schema design of a multidimensional scenario.

2 The Traditional Relational OLAP Approach

To illustrate the failure of the traditional ROLAP approach and to motivate our alternative approach, we refer to a sample scenario, stemming from a joint research project with a worldwide operating retail research company. In their business, facts like sales, stock, or turnover values of single articles in single shops at a specific period of time are monitored and collected to form the raw database. For example:

```
Facts(ArticleID, ShopID,  Period,  SALES,  STOCK,  TURNOVER)
      TR-75       203      05/97    121     78       333
      TS-78       203      05/97    112     63       121
      .......
```

Generally, this raw database is commonly called a *fact table* and consists of two main components: A set of dimensions, we denote as 'primary attributes' PA_i ($1 \le i \le n$) forming the composite primary key of the table and a set of measures $\{f_1, ..., f_k\}$ denoting the

Final edited form was published in "DaWaK: International Conference on Big Data Analytics and Knowledge Discovery. London 2000", S. 189-198, ISBN 978-3-540-44466-4
http://dx.doi.org/10.1007/3-540-44466-1_19

figures being analyzed. The number of primary attributes n determines the dimensionality of the problem:

$$Facts(\underline{PA_1},...\underline{1},\underline{PA_n}, f_1,..., f_k)$$

In a further step of the retail research evaluation phase, the raw database is analyzed in two ways: On the one hand, the data is aggregated along a predefined classification hierarchy (like the one in figure 1). On the other hand, the data is split into characteristic features of the single articles or shops. For example, each shop holds country specific information about its purchase class or its shop type ("cash&carry", "retail", "hypermarket"). In the product dimension, each article of the 250.000 monitored products belongs to one of 400 product families. Furthermore, each article is characterized by five attributes valid for all products ('*brand*', '*package type*', ...) and about 15 attributes which are valid only in the product family or product group to which the article belongs to ('*video system*' only for the product group *Video* equipment, '*water*' usage only for the product family *Washers*).

2.1 Star Schema

The simplest traditional way to model this qualifying information skeleton used during the analysis process is to use a single *dimension table* D^i ($1 \le i \le n$) for each dimension to resolve high-level terms according to the classification hierarchy and to represent dimensional attributes. Since each dimension is connected to the corresponding primary key of the fact table, the whole scenario looks like a 'Star'. Figure 3 illustrates the star schema for the ongoing example. Formally, the schema of the dimension table for the dimension i consists of the primary attribute PA_i, all classification attributes CA_j ($1 \le j \le p$) and the complete set of dimensional attributes DA_k ($1 \le k \le m$).

$$D^i(\underline{PA_i}, DA_1, ...,DA_m, CA_1,...,CA_p)$$

It is worth to note that (mainly for performance reasons) the classification hierarchy is modeled as a set of functionally dependent attributes ($CA_j \rightarrow CA_{j+1}$ ($1 \le j < p$)). Furthermore, the distinction of dimensional elements organized in hierarchies and dimensional attributes further characterizing the elements explicitly prescribed in the multidimensional model gets totally lost. Figure 4 shows the product dimension table for the market research example.

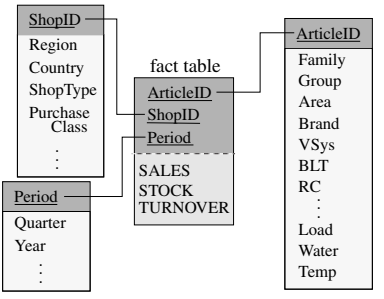

Fig. 3. Sample Star Schema

	PA	DA_1	DA_2			. . .			DA_m	CA_1	CA_2	CA_3
Articles	ArticleID,	Brand,	VSys,	BLT,	RC,	... Load,	Water,	Temp,	Family,	Group,	Area)	
	TR-75	Sony	HI8	2h	NULL	... NULL	NULL	NULL	Camcorder	Video	ConsElectr	
	TS-78	Sony	HI8	2h	NULL	... NULL	NULL	NULL	Camcorder	Video	ConsElectr	
	A200	JVC	N8	3h	NULL	... NULL	NULL	NULL	Camcorder	Video	ConsElectr	
	V-201	JVC	VHS	NULL	Yes	... NULL	NULL	NULL	HomeVCR	Video	ConsElectr	
	ClassicI	Grundig	VHS-C	NULL	No	... NULL	NULL	NULL	HomeVCR	Video	ConsElectr	
	....											
	AB1043	Ariston	NULL	NULL	NULL	... 5kg	451	NULL	Washer	HomeAppl	WhiteGoods	
	Princess	Miele	NULL	NULL	NULL	... 5kg	411	NULL	Washer	HomeAppl	WhiteGoods	
	SuperII	Hoover	NULL	NULL	NULL	... 4kg	541	NULL	Washer	HomeAppl	WhiteGoods	
	Duett	Miele	NULL	NULL	NULL	... 6kg	NULL	37°C	Dryer	HomeAppl	WhiteGoods	
	Lavamat	AEG	NULL	NULL	NULL	... 6kg	NULL	39°C	Dryer	HomeAppl	WhiteGoods	
	....											

Fig. 4. Sample star schema dimension table for the product dimension

Final edited form was published in "DaWaK: International Conference on Big Data Analytics and Knowledge Discovery. London 2000", S. 189-198, ISBN 978-3-540-44466-4
http://dx.doi.org/10.1007/3-540-44466-1_19

2.2 Snowflake Schema

Elimination of the functional dependencies in the single dimension tables, i.e. normalizing the *star schema*, leads to small satellite tables representing the dimensional hierarchy. Therefore, the set of dimension tables for a single dimension is modeled in the following manner:

$$D^i(\underline{PA_i}, DA_1, ..., DA_{n_1}, CA_1)$$
$$D^i_1(\underline{CA_1}, DA_{n_1+1}, ..., DA_{n_2}, CA_2)$$
$$\ddots$$
$$D^i_{p-1}(\underline{CA_{p-1}}, DA_{n_{p-1}+1}, ..., DA_m, CA_p)$$

Again, PA_i $(1 \le i \le n)$ denotes the primary attribute for the i-th dimension for joining the fact table. The classification attribute CA_j forming the hierarchy acts as a foreign key in classification level j-1 and as a primary key in level j. Furthermore, all dimensional attributes DA_k at level j are fully dependent on CA_j. To adopt the dimension table for the ongoing example, the classification attributes 'Group' and 'Area' are shifted into two new relations. The relational schema for the product dimension of the current example is depicted in figure 5:

Articles(ArticleID,	Brand,	VSys,	BLT,	RC,	...	Load,	Water,	Temp,	Family)
TR-75	Sony	HI8	2h	NULL	...	NULL	NULL	NULL	Camcorder
TS-78	Sony	HI8	2h	NULL	...	NULL	NULL	NULL	Camcorder
A200	JVC	N8	3h	NULL	...	NULL	NULL	NULL	Camcorder
V-201	JVC	VHS	NULL	Yes	...	NULL	NULL	NULL	HomeVCR
ClassicI	Grundig	VHS-C	NULL	No	...	NULL	NULL	NULL	HomeVCR
....									
AB1043	Ariston	NULL	NULL	NULL	...	5kg	45l	NULL	Washer
Princess	Miele	NULL	NULL	NULL	...	5kg	41l	NULL	Washer
SuperII	Hoover	NULL	NULL	NULL	...	4kg	54l	NULL	Washer
Duett	Miele	NULL	NULL	NULL	...	6kg	NULL	37°C	Dryer
Lavamat	AEG	NULL	NULL	NULL	...	6kg	NULL	39°C	Dryer
....									

Families(Family,	Group)
Camcorder	Video
HomeVCR	Video
....	
Washer	HomeAppl
Dryer	HomeAppl

Groups(Group,	Area)
Video	ConsElectr
....	
HomeAppl	WhiteGoods

Fig. 5. Sample snowflake schema dimension tables for the product dimension

2.3 Summary and Conclusion

The *star / snowflake schema* approach allows modeling a wide range of simple multidimensional scenarios. From a performance point of view the *star schema* avoids a lot of lookup joins with the satellite tables but it is reprehensible from a schema design point of view.

Both traditional approaches however fail from an implementation and schema design point of view, if the *existence of dimensional attributes* depends on *values of the dimensional elements* in the classification hierarchy. As we have seen in the market research scenario, this problem becomes even worse, e.g. dimensional attributes '*Water*' and '*Load*' are only applicable for *washers* and not for *video equipment*.The Alternative Relational OLAP Approach

Final edited form was published in "DaWaK: International Conference on Big Data Analytics and Knowledge Discovery. London 2000", S. 189-198, ISBN 978-3-540-44466-4
http://dx.doi.org/10.1007/3-540-44466-1_19

With our solution, we adopt the idea of [SmSm77] introducing a special class of attributes, called *discriminating attributes*. This kind of attributes holds relation names as their attribute values, which allows a real hierarchical representation of the problem of mapping a pyramid of concepts [Lore87], i.e. the classification hierarchy, to the relational data model.

In a first step, we can model each product group, i.e. leaf node of the classification tree, in a separate relation with all the node specific dimensional attributes. For example, the four sample product families from figure 1 are modeled in the following way:

Camcorder	ArticleID	Brand	VSys	BLT
	TR-75	Sony	HI8	2h
	TS-78	Sony	HI8	2h
	A200	JVC	N8	3h

Washer	ArticleID	Brand	Load	Water
	AB1043	Ariston	5kg	45l
	Princess	Miele	5kg	41l
	SuperII	Hoover	4kg	54l

HomeVCR	ArticleID	Brand	VSys	RC
	V-201	JVC	VHS	Yes
	ClassicI	Grundig	VHS-C	No

Dryer	ArticleID	Brand	Load	Temp
	Duett	Miele	6kg	37°C
	Lavamat	AEG	6kg	39°C

More formally, the schema of a classification node C at the lowest, i.e. first classification level with the classification attribute CA_1 is denoted by:

$$C(\underline{PA}, DA_1, ..., DA_m)$$

As usual, PA is the primary attribute for the join with the fact table and the set of DA_k ($1 \le k \le m$) denotes the dimensional attributes which are applicable to the classification node C.

The construction of the classification hierarchy is made in a bottom-up fashion, i.e. sets of classification nodes are grouped into a new high-level term, i.e. a new classification node. Suppose, in the j-th step, the classification nodes $\{C^1, ..., C^q\}$ with the set of locally valid dimensional attributes $\{DA_1^i, ..., DA_{m_i}^i\}$ for each C^i corresponding to the classification attribute CA_{j-1} are subsumed by the higher level node C corresponding to the classification attribute CA_j. The set of valid dimensional attributes is achieved by intersecting all attribute sets of the subsumed nodes.

$$\{DA_1, ..., DA_m\} := \bigcap_{i=1}^{q} \left\{ DA_1^i, ..., DA_{m_i}^i \right\}$$

For example, *Camcorders* and *HomeVCR* are classified into the class *Video*. *Washers* and *Dryers* are subsumed by the new higher level classification node *Home Appliances*. Furthermore, only those dimensional attributes are propagated to the new parent node, which are still valid there. Hence, the specific dimensional attributes '*BLT*' (for *Camcorder*) and '*RC*' (for *HomeVCR*) are lost, whereas the attributes '*VSys*' and '*Brand*' are propagated to the *video* class.

Generally, the schema of a new classification node C for the classification attribute CA_j ($1 < j \le p$) is algorithmically determined by:

$$C(\underline{PA}, DA_1, ..., DA_m, CA_1, ..., CA_{j-1}, CA_j) = \bigcup_{i=1}^{q} \pi_{(\underline{PA}, DA_1, ..., DA_m, CA_1, ..., CA_{j-1}, 'C^i')} C^i$$

The key point of this technique is that *each classification node* C^i *is added as a constant value for the new classification attribute* CA_j *of the new classification node* C.

In building the higher level classes, we intensively use the view mechanism of the relational database system at the implementation side. Below, the view definitions of the

Final edited form was published in "DaWaK: International Conference on Big Data Analytics and Knowledge Discovery. London 2000", S. 189-198, ISBN 978-3-540-44466-4
http://dx.doi.org/10.1007/3-540-44466-1_19

product groups *Video* and *Home Appliances* are illustrated. In analogy to the formal description, each relation name appears as a constant value in the new view:

```
create view Video
  (ArticleID, Brand, VSys, Family) as
select ArticleID, Brand, VSys, 'Camcorder'
from Camcorder
union
select ArticleID, Brand,  VSys, 'HomeVCR'
from HomeVCR
```

Video(ArticleID,	Brand,	VSys,	Family)
TR-75	Sony	HI8	Camcorder
TS-78	Sony	HI8	Camcorder
A200	JVC	N8	Camcorder
V-201	JVC	VHS	HomeVCR
ClassicI	Grundig	VHS-C	HomeVCR

```
create view HomeAppl
  (ArticleID, Brand, Load, Family) as
select ArticleID, Brand, Load, 'Washer'
from Washer
union
select ArticleID, Brand, Load, 'Dryer'
from Dryer
```

HomeAppl(ArticleID,	Brand,	Load,	Family)
AB1043	Ariston	5kg	Washer
Princess	Miele	5kg	Washer
SuperII	Hoover	4kg	Washer
Duett	Miele	6kg	Dryer
Lavamat	AEG	6kg	Dryer

Each view holds the primary attribute, the applicable dimensional attributes and (in analogy to the star schema) all classification attributes. Furthermore, each view builds the basis for defining higher level classification nodes. This recursivly definition is shown below for the classification hierarchy depicted in figure 1.

```
create view ConsElectr
  (ArticleID, Brand, Family, Group) as
select ArticleID, Brand, Family, 'Video'
from Video
union ....

create view WhiteGoods
  (ArticleID, Brand, Family, Group) as
select
ArticleID, Brand, Family, 'HomeAppl'
from HomeAppl
union ....
```

```
create view Articles
  (ArticleID, Brand, Family, Group, Area) as
select ArticleID, Brand, Family, 'ConsElectr'
from ConsElectr
union
....
union
select ArticleID, Brand, Family, Group, 'WhiteGoods'
from WhiteGoods
```

In comparison to the traditional modeling approach (figure 4), only those attributes are available in the dimension table which are applicable for all dimensional elements. To address specific features, the corresponding dimensional sub-tables have to be used, whose names are specified as instances of the classification attributes. Figure 6 summarizes the modeling of the classification hierarchy using the proceeding, illustrated in this section. In analogy to figure 5 (snowflake schema), our approach can be straightforwardly normalized, too.

Articles(ArticleID,	Brand,	Family,	Group,	Area)
TR-75	Sony	Camcorder	Video	ConsElectr
TS-78	Sony	Camcorder	Video	ConsElectr
A200	JVC	Camcorder	Video	ConsElectr
V-201	JVC	HomeVCR	Video	ConsElectr
ClassicI	Grundig	HomeVCR	Video	ConsElectr
...				
AB1043	Ariston	Washer	HomeAppl	WhiteGoods
Princess	Miele	Washer	HomeAppl	WhiteGoods
SuperII	Hoover	Washer	HomeAppl	WhiteGoods
Duett	Miele	Dryer	HomeAppl	WhiteGoods
Lavamat	AEG	Dryer	HomeAppl	WhiteGoods

Fig. 6. Sample dimension table for the product dimension (alternative approach)

To put it into a nutshell, in the case of the traditional approach, the (senseless) query asking for total sales of *Home Appliances* by '*video system*' would result in a table scan of the dimension table, resulting in an empty join partner for the fact table and at last in a numerical zero. In our alternative approach, the query would be rejected, since *Home Appliances* does not contain a dimensional attribute '*video system*'.

Final edited form was published in "DaWaK: International Conference on Big Data Analytics and Knowledge Discovery. London 2000", S. 189-198, ISBN 978-3-540-44466-4
http://dx.doi.org/10.1007/3-540-44466-1_19

3 Object-Relational Design

Another alternative and novel approach to overcome the limitations of the traditional star-/snowflake schema pattern is to use object-relational techniques. Since object-relational concepts are supported only to a certain degree and implemented in a very system-specific manner, we propose an object-relational schema design based on the capabilities of the IBM DB2/UDB V6.1 database system.

The design of an object-relational schema in DB2 is divided into two phases. In a first step, we have to define the type hierarchy and references based on types. Referring to these types, we are then able to 'instantiate' regular tables (also called: typed tables). Moreover, in opposite to the approach shown in the previous section, we have to proceed top-down when defining the object-relational schema of the dimensional structures.

3.1 Type Definitions

The super-type of a dimension holds only the most generic dimensional attributes and all possible classification attributes. For the ongoing example of the product dimension, the following DDL statement introduces the generic type of *Articles_T*, where the single articles are identified by the *ArticleID* attribute.

```
create type Articles_T as (brandvarchar(30),
           area   varchar(30),
           group  varchar(30),
           family varchar(30) );
```

Analogous to the classical concept of inheritance, special classes of products are derived from the more general classes of products and specific dimensional attributes are added to the derived type. Below are the SQL statements to define the necessary subtypes of the product classification hierarchy. For each sub-type, the keyword UNDER denotes the corresponding super-type.

```
create type ConsElectr_T under Articles_T as ( ... );
create type WhiteGoods_T under Articles_T as ( ... );
create type Video_T      under ConsElectr_T as(vidsysvarchar(30) );
create type HomeAppl_T   under WhiteGoods_T as(loadvarchar(30) );
create type HomeVCR_T    under Video_T as    (RC    char(1) );
create type Camcorder_T  under Video_T as    (BLT   varchar(5) );
create type Washer_T     under HomeAppl_T as (Water varchar(5) );
create type Dryer_T      under HomeAppl_T as (Temp  short) ;
```

Once we have defined the types of the dimensional structures, e.g. for the *Products* dimension and the *Shops* dimension, we are able to create a type for the fact table. Although this is not a mandatory step to design the schema of the fact table using OR technology, we are able to achieve some advantage when querying the database later. Therefore, the type of a fact table consists of two references to the super-types of the participating dimensions.

```
create type Facts_T as (
        ArticleID REF(Articles_T),
        ShopID REF(Shops_T),
        Period date,
```

Final edited form was published in "DaWaK: International Conference on Big Data Analytics and Knowledge Discovery. London 2000", S. 189-198, ISBN 978-3-540-44466-4
http://dx.doi.org/10.1007/3-540-44466-1_19

```
Sales  integer,
Stock  integer,
Price  integer);
```

It is worth mentioning here that this construction yields the following advantage: Consider the situation where we have to setup a data mart for a specific product group, e.g. video equipment. We are now able to create a type of a customized fact table allowing only references to the articles belonging to the video class. Therefore, we would substitute the reference to the generic articles type by a reference to the more specific type of *Video_T (... ArticleID REF(Video_T), ...)*. This design ensures already at the schema level (!!) that the fact table for this specific data mart will never contain a product other than a video article.

3.2 Typed Table Definitions

Once we have defined the type hierarchy for the classification schema, we are now in the position to 'instantiate' these types resulting in so-called 'typed tables'. Analogous to the type definition, we proceed top-down. For instantiation of the super-type, we need to introduce a object identifier attribute. For the ongoing example, we use the *ArticleID* as reference attribute (or primary key attribute in relational terminology), which content is given as user generated (in opposite to system generated). All dependent sub-tables are instantiated according to their type and referring to their direct super-type.

```
create table Articles of Articles_T (REF IS ArticleID user generated);
create table ConsElectr of ConsElectr_T under Articles inherit select privileges;
create table WhiteGoods of WhiteGoods_T under Articles inherit select privileges;
create table Video of Video_T under ConsElectr inherit select privileges;
create table HomeAppl of HomeAppl_T under WhiteGoods inherit select privileges;
create table HomeVCR of HomeVCRF_T under Video inherit select privileges;
create table Camcorder of Camcorder_T under Video inherit select privileges;
create table Washer of Washer_T under HomeAppl inherit select privileges;
create table Dryer of Dryer_T under HomeAppl inherit select privileges;
```

Again it is worth mentioning that this construction provides a huge advantage when dealing with changing dimensions. Consider again the product dimension. New articles may be added, some articles may be re-classified, and other articles may be deleted because they are no longer sold or their sales are no longer monitored. Since, however, all articles are of the same type, we can simply instantiate a specific type multiple times. Each resulting table reflects then a valid state of the classification hierarchy an can be used to analyze the fact data under certain valid time perspectives.

After creating the dimensional hierarchies, we can instantiate the fact table from the type *Facts_T*. Two aspects have to be considered explicitly. First of all, each fact gets a system generated object identifier *FactID*. More important is the instantiation of the references defined in the type specification. Each reference is pointing to an appropriate table, e.g. *ArticleID* is referencing the *Articles* table, not longer the *Articles_T* type.

```
create table Facts of Facts_T (ref is FactID system generated,
               ArticleID with options scope Articles,
               ShopID with options scope Shops);
```

Final edited form was published in "DaWaK: International Conference on Big Data Analytics and Knowledge Discovery. London 2000", S. 189-198, ISBN 978-3-540-44466-4
http://dx.doi.org/10.1007/3-540-44466-1_19

3.3 Data Manipulation

The object-relational design of a data warehouse database has also some impact on data manipulation. Consider a simple insert statement for the product dimension. The only difference to a 'pure relational' insert is the casting of the article identification to the specific object identifier. Below is an example for a camcorder and a *Dryer* article.

```
insert into Camcorders ( ArticleID, brand, vidsys, blt, family, group, area) values
(Camcorders_T('TR-75'), 'Sony', 'HI8', '2h', 'Camcorder', 'Video', 'ConsElectr');
insert into Dryers ( ArticleID, brand, load, temp, family, group, area) values
(Dryers_T('Duett'), 'Miele', '6kg', '37oC', 'Dryer', 'HomeAppl', 'WhiteGoods');
```

To keep the consistency of references, we are now able to insert facts into the fact table. Again, we have to cast the values of the references to their corresponding type of the dimension tables, i.e. article identifiers are casted to *Articles_T*, shop names are casted to *Shops_T*.

```
insert into Facts (FactID, ArticleID, ShopID, Period, Sales, Stock, Price) values
(Facts_T('1'), Article_T('TR-75'), Shops_T('TeVi'), '1999-12-02', 45, 22, 998);
```

When querying the database we can take advantage of the references which may be visualized as predefined join paths. For example, grouping fact data by region (from the Shops table) and Products groups (from the Articles table) may be specified using the '->' operator without any explicit join.

```
select f.ShopID->region, f.ArticleID->group, SUM(sales)
   from Facts f
   group by f.ShopID->region, f.ArticleID->group
```

However, we can not retrieve specific attributes using this construction. The grouping by *Video Systems* for all video equipment would be expressed with the '*Video*' as the correct dimension table as follows:

```
select f.ShopID->region, a.vidsys, SUM(sales)
   from Facts f, Articles a
   where f.ArticleID = a.ArticleID
   group by f.ShopID->region, a.vidsys
```

In summary, object-relational technology provides a powerful tool for schema design in data warehouse environments. Especially, the design of a real classification hierarchy with classification and dimensional attributes finds an adequately representation using inheritance mechanisms on types and typed tables. Extending the object-relational representation to the fact table enables the designer to define some kind of structural constraints already at the schema level.

4 Summary and Conclusion

This paper addresses the problem of locally valid dimensional attributes within a classification hierarchy in the context of a multidimensional schema design. We show that the traditional way of a relational representation is not feasible and give two practicable solutions to this problem. The basic idea of the first proposed mechanism is based on the article of [SmSm77]. Building a pyramid of concepts in a bottom up manner using regular relational views may be seen as a seamless extension of the traditional *star /*

Final edited form was published in "DaWaK: International Conference on Big Data Analytics and Knowledge Discovery. London 2000", S. 189-198, ISBN 978-3-540-44466-4
http://dx.doi.org/10.1007/3-540-44466-1_19

snowflake schema approach. The second top-down approach is based on object-relational techniques. We demonstrate how we can take advantage of object-relational concepts like types, typed subtables, inheritance on types and tables, and references. Again this method results in a flexible schema design.

References

ALTK97 Albrecht, J.; Lehner, W.; Teschke, M.; Kirsche, T.: Building a Real Data Warehouse for Market Research, to appear in: *8th International Conference and Workshop on Database and Expert System Applications* (DEXA'97, Sept. 1-5, 1997)

CoCS93 Codd, E.F.: Codd, S.B.; Salley, C.T.: *Providing OLAP (On-line Analytical Processing) to User Analysts: An IT Mandate*, White Paper, Arbor Software Corporation, 1993

GBLP96 Gray, J.; Bosworth A.; Layman A.; Pirahesh, H.: Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Total, in: *12th IEEE International Conference on Data Engineering* (ICDE´96, New Orleans, Louisiana, Feb. 26 -Mar. 1, 1996), pp. 152-159

GuMu95 Gupta, A.; Mumick, I.: Maintenance of Materialized Views: Problems, Techniques, and Applications, in: *IEEE Data Engineering Bulletin, Special Issue on Materialized Views & Data Warehousing* 18(1995)2, pp. 3-18

Hype99 N.N.: *Hyperion Essbase OLAP Server*, Hyperion Software Corporation, 1999 (http://www.hyperion.com/essbaseolap.cfm)

Inmo96 Inmon, W.H.: *Building the Data Warehouse*, 2nd edition, New York, John Wiley & Sons, 1996

Kimb96 Kimball, R.: *The Data Warehouse Toolkit*, New York, John Wiley & Sons, 1996

Lore87 Lorenzen, P.; *Constructive philosophy*, Amherst, Univ. of Massachusetts Press, 1987

Micr95 N.N.: *Microstrategy 6*, MicroStrategy Inc., 1999 (http://www.microstrategy.com/Products/index.asp)

Shos97 Shoshani, A.: OLAP and Statistical Databases: Similarities and Differences, in: *Proceedings of the 16th Symposium on Principles of Database Systems* (PODS'97, Tuscon, Arizona, May 12-14, 1997), pp. 185-196

SmSm77 Smith, J.M.; Smith, D.C.P.: Database Abstractions: Aggregation and Generalization, *ACM Transactions on Database Systems 2(1977)2*, pp. 105-133

SQL99 N.N.: *ISO/IEC 9075*: 1999, Informatik Technology - Database Languages - SQL, 1999

VaSe99 Vassiliadis, P.; Sellis, T.: A Survey of Logical Models for OLAP Databases. In: *SIGMOD Record 28(1999)4*