February 1999

A conceptual modelling perspective for data warehouses

Jaroslav Pokorný
University of Prague, CZ, pokorny@ksi.ms.mff.cuni.cz

Peter Sokolowsky
University of Prague, CZ and IKS, Saarbrücken, ps@itm.uni-sb.de

A Conceptual Modelling Perspective for Data Warehouses

Jaroslav Pokorný
University of Prague, CZ (pokorny@ksi.ms.mff.cuni.cz)

Peter Sokolowsky
University of Prague, CZ and IKS, Saarbrücken (ps@itm.uni-sb.de)

Contents

Electronic Business Engineering / 4. Internationale Tagung Wirtschaftsinformatik 1999.
Hrsg.: August-Wilhelm Scheer; Markus Nüttgens. – Heidelberg: Physica-Verlag, 1999

Abstract

The volume of information of the most various types stored electronically in a company is increasing to an ever-greater extent. While in the field of operational systems everything is aimed at achieving the quickest possible throughput, in the dispositive field, questions regarding the total overview or detailed views are of interest. OLAP servers are multidimensionally structured. They are therefore suited for the analysis of multidimensional datastores. The functionality of the OLAP server, such as, creation of forms, drill down, roll up, slice and dice, analysis technology, multidimensional consolidation, etc. demonstrate the advantages of this tool. The analysis of the relevant features of the data warehouse and OLAP is based on both the mainstream literature and on our experience in a two-year project Data Warehouse for Tupperware Inc.

The second problem addressed by this paper is the discussion of recent approaches for a proposition some formal definitions of basic constructs used in so called multidimensional modelling which seems to be an important technique for data warehousing and OLAP. It is different from E-R modelling and offers a number of important advantages that the E-R modelling lacks.

We show a relationship of E-R modelling to the multidimensional modelling and describe a broad class of multidimensional databases based on so called constellation schemes with explicit hierarchies.

1 Introduction

The volume of information of the most various types stored electronically in a company is increasing to an ever-greater extent. The information stored in this manner in the fields of, for example, marketing, design, production, management and even as far as controlling represents an important information potential. However, in most enterprises, it is not possible to utilize the whole of the data capacity. In the areas mentioned above, the data often cannot be utilized fully as, as a rule, it is insufficient or inadequately structured.

Even the possibility of using external data – already possible today – such as, for example, market research data, information from external suppliers, etc. is hardly used or not used at all.

It is known that such in information pool[1] cannot consist of operative data only. By uniting operative, external and historical data from the enterprise, so-called dispositive or flexible data, which form the data warehouse, is created (Aberdeen

[1] The associated software technology is called usually OLTP (Online Transaction Processing).

Group Market Viewpoint, 1995). In section 2 we introduce a data warehouse example at Tupperware Germany. Section 3 describes the most important functionalities of OLAP. OLAP design problems are discussed in section 4. The main part of the paper, section 5, is focused on multidimensional modelling which makes it possible to fulfil the OLAP functionality. The approach presented is based on the theory introduced in (Pokorny 1998a) and (Pokorny 1998b). Finally, section 6 contains some conclusions.

2 An Example of Data Warehouse and OLAP

A data warehouse was 1997 implemented as a prototype (so-called ”Pilot Project”) in Tupperware Germany. It is a part of the Tupperware Company, an enterprise that has been affiliated with the American group of Premark International since 1987. At present, there are some 165 so-called regional trading offices, economically independent enterprises, which, in the same way as the wholesale company, Tupperware Deutschland GmbH, carry out a purely selling function. The regional trading offices form the interface between the consultants who work throughout the whole of Germany, who offer and sell the Tupperware products during home presentations (”Tupper Party”) and the central organization. The regional dealers (RD) are allocated employees according to the regions in which they are located. These employees are, in turn, employees of Tupperware. These consultants are divided into groups. Each of these teams is cared for by a so-called Group Consultant (GC). She is the partner for the RD and for the consultants. The structure of the sales organization described in the foregoing is depicted in Fig. 1.

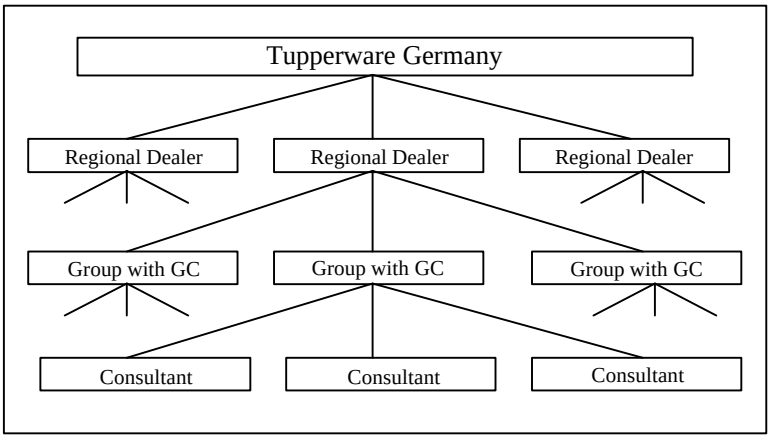

Figure 1: Company organizational sales structure

The software (Fig. 2) which is used to create the data warehouse is ideal for a medium-sized company such as Tupperware Germany. There is no necessity to install a completely new system as the selected products run on a platform-independent basis. In the Tupperware trading organization, the data warehouse was implemented on a NT server. For analysis, standard IBM compatible personal computers are used. It is possible for every department in the company and for management to carry out analyses – taking into account the data release involved – via a network which has been installed within the company.

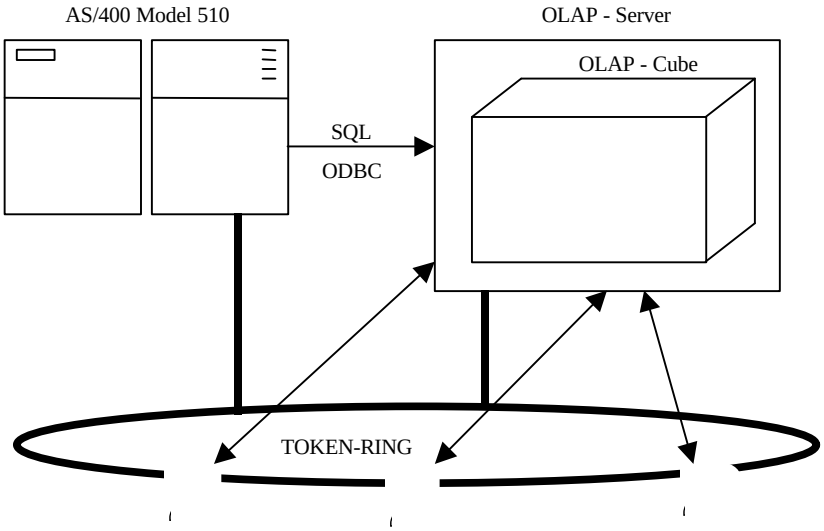

Figure 2 : System structure of data warehouse at Tupperware Germany

The development of a product can be quickly and reliably evaluated, it is necessary to have an analysis tool, which enables intuitive enquiries to be made. At Tupperware, it was decided to use the product with the name of Improptu (from Cognos).

Here, metadata and raw data, which are available on an IBM-Computer, are transferred to the data warehouse via an ODBC interface.

The development is changing from the previously hierarchical order of the information system in an enterprise towards that of the data warehouse. The elementary fact here being that the data warehouse is a data source of all information systems, independent of their hierarchical order (Glassey 1997).

For example, a manager wants to have an analysis of his area of responsibility. For this purpose, he wants to know how many pieces of certain category of product have been purchased by which dealers in a prescribed period of time and wishes to compare the results with the figures for the previous year.

It can be seen that the information must be described in an n-dimensional form. This is referred to as ”multidimensionalism”. Thus, OLAP environment needs more satisfiable design methods. Most of them are based on so called multidimensional (or dimensional) modelling.

There are two basic approaches to multidimensional modelling:

- conceptual structures are based on tables (dimension and fact tables) arranged into so called star schemes,
- conceptual structures are based on so called hypercubes (data cubes, multidimensional arrays) that represent the data as a multidimensional structure.

OLAP[2] servers (Frye 1995), usually running over a warehouse, are multidimensionally structured to store data and relationships among data. As a result of this fact, they are especially suitable for the analysis of multidimensional datastores.

As opposed to relational databases, the strengths of the OLAP servers lie in their powerful analytical functions (Mann and Mehta 1996).

3 Functionalities of OLAP

The advantages of the OLAP server lie in the fact that it is possible to achieve the following functionalities without any additional programming work:

a. The Creation of Formulas

The fields involved cannot always be called up directly. They must be calculated from the data available. This then means that such data, as is the case with all other data from the data warehouse, must be treated with equal priority by the OLAP server for analysis.

b. Multidimensional Consolidation

Hierarchically classified data represent no problem for the OLAP server. Compilations and complex calculations can be implemented easily. For example, the number of articles sold per customer can be compiled and then further aggregated per region. This demonstrates that the OLAP server can support aggregates, consolidations and various types of hierarchies.

c. Drill down Roll up (Fig. 3)

”Drill down” means that the very aggregated results are broken down step by step according to their hierarchy. For example, the total turnover of a company can be subdivided into North and South. From here, in any continent or region selected, a further drill down can be carried out into sales regions and further into individual states and even as far as individual customers.

---

[2] Online Analytical Processing

The ”roll up” has the reverse function. It begins at the data level and aggregates the data step by step.

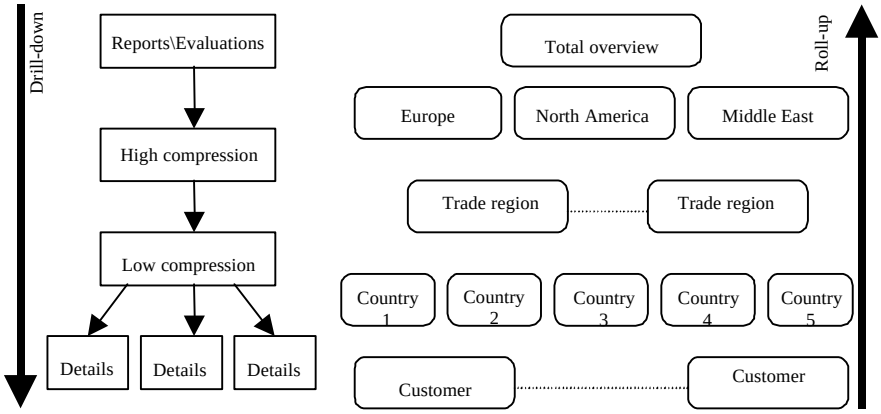

Figure 3: Drill down and roll up

d. Slice and Dice

With this it is possible to present the OLAP cube in specific partial views such as slices or dices. For example, it is possible to see a slice with the turnover of all products. By turning and revolving the cube further, more and more new views are presented (Fig. 4).

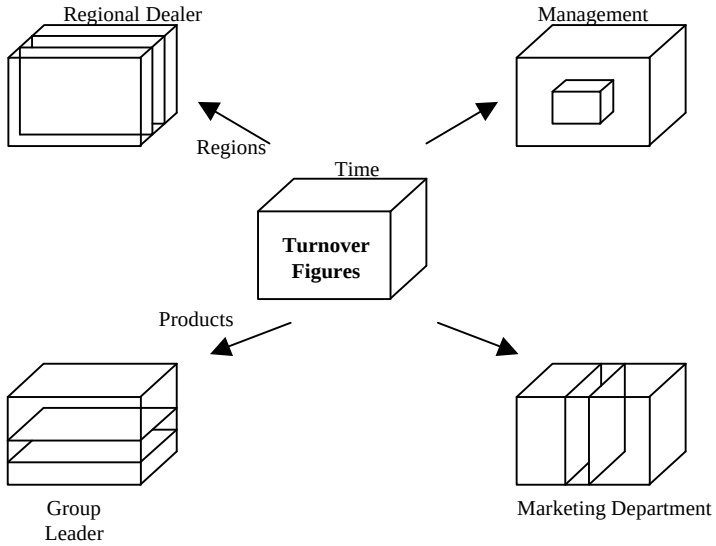

Figure 4: Slicing and Dicing

e. Data retention in the OLAP server

Multidimensional data is stored in aggregated or compressed form. The compression of the data is adapted to suit the data warehouse. The storage of matrices with a large number of ”zero cells” is treated separately from matrices with a low volume of ”zero cells”. This leads to a minimization of storage space, thus making the analysis of great volumes of data possible. By means of this special form of storage, access to the hard disk is avoided which in turn has a positive effect on performance. In the OLAP server itself, only greatly aggregated data is to be found. If, something which occurs only very seldom, the analysis of the data goes back as far as an individual booking, the server accesses the operational datastores. This means that the gigantic volumes of booking data remain in the operative system but are, nevertheless, available at all times (Weldon 1995).

f. Analysis technology

If an analysis is started, the OLAP server ”knows” exactly what the status of the last analysis is. If a further reaching analysis is now started, then the server does not begin from scratch, but instead uses the data already selected (Glassey 1997).

With the new technology of data storage and the ”intelligence” of the OLAP server it is possible for the user to design analyses and enquiries intuitively. With the implementation of a data warehouse on an OLAP server, valuable data can be extracted from the masses of data, which, in its quality, can create particularly great competitive advantages. It can be said that the OLAP technology will assert itself in the future.

g. The heterogeneous environment (Fig. 5)

The relevant data is often obtained from heterogeneous data sources. The structures and access possibilities can be from anything from relational databases to simple ASCII files. In order to arrive at an efficient analysis, it is necessary to administer these in the complex data warehouse (Aberdeen Group Market Viewpoint, 1995).

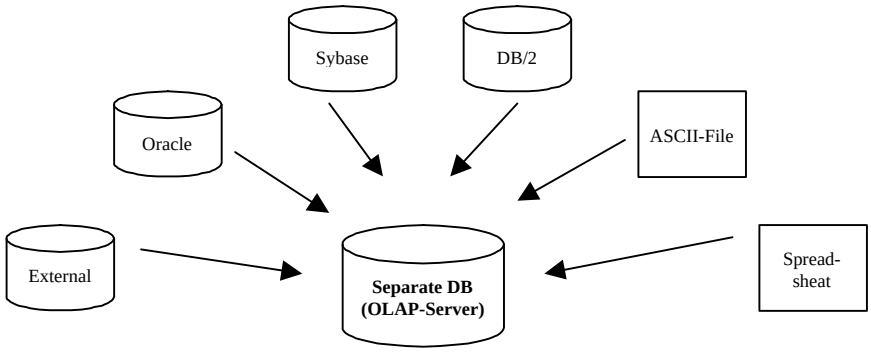

Figure 5: Heterogeneous data sources for OLAP server

In order to ”selectively process” a data warehouse in all of the functionalities already mentioned, special software is required which is able to correspond with the server.

There is already a large selection of such tools on the market. The main characteristics of the software are its ability to generate reports without any programming effort of its own. Software packages with a report generator, an analysis program and an application to present this data in logically arranged tables with graphics are offered, among others, by the following leading software manufacturers (Fig. 6).

	Business Objects 3.1	*Impromptu 3.0*	*ShowCase Vista*	*Express*
Manufacturer	Business Objects	Cognos	ShowCase	ORACLE
Configuration	Very good	very good	very good	very good
Safety	Good	very good	good	excellent
Simple report in column form	Good	good	good	very good
Complex report with several tables	Good	very good	satisfactory	very good
Features for power user	Satisfactory	very good	good	good
Speed	Poor	very good	good	good
Documentation	Satisfactory	good	satisfactory	good
Support	Satisfactory	good	satisfactory	very good
Slice & Dice	✓	✓	✓	✓
Drill down	✓	✓	✓	✓
HW platform-independent	✓	✓	✓	✓
DB platform	Tm l server Fact Gab. …	ORACLE SYBASE- SQL. …	DB2/400	ORACLE Access to all standard SQL databases
Operating system	Windows NT, UNIX	Windows 95, Windows NT	AS 400	Windows 95, Windows NT
Object-oriented	✓	✓	✓	✓
Internet connection	–	–	–	–
ODBC interface	✓	✓	✓	✓
MOLAP	✓	✓	–	✓
32bit application	✓	–	–	✓

Figure 6: Comparison of OLAP tools selected for the Tupperware Inc.

The picture developing for the future is one of the enterprises being connected to its subsidiaries and partners via Internet or Intranet. These problems are solved in (Sokolowsky et al 1997).

4 OLAP Database Design

There are significant differences between OLTP and OLAP[3] databases (see, e.g. Fig. 7 adapted from (Codd 1993)). Whereas in OLTP databases indexing, precalculated fields, and data duplication are avoided, OLAP databases keep derived tables composed of data placed in other tables. The reason for it is easy. OLAP databases are optimized for the purpose of easy querying rather than for inserting and updating data as the OLTP databases do it. Notice that DW and, consequently, OLAP databases store relatively stable data that are updated rather periodically than immediately.

In OLAP, the most important aspect of database design is focused on how we will to need analyze the data. Well-known modelling methodologies with the most common E-R diagrams as the leading one in OLTP environment are inappropriate in OLAP environment. The main difference between E-R modelling and OLAP application design is representation of understanding of processing logic. E-R schema assumes that data are processed by programs and represents only the relationships between objects.

Criteria	*OLAP*	*OLTP*
Enquiries	In part, not predictable, (answer time: seconds to minutes)	Predictable (answer time: 0-5 seconds)
Data contents	Several years, Deduced and aggregated data	Current periods, Possibly, short histories
Data organization	The investigation can extend to cover the whole of the enterprise	Application oriented
Dimensionality	Frequently multi-dimensional	Two dimensional
Use of data	Mostly unstructured, the investigation is at the core	High degree of structuring (transaction oriented and enables location of individual data records)
Information types	Formatted or, resp., unformatted and internal/external information	Formatted and internal information
Redundancy	Monitored redundancy (star and snowflake)	Minor
Access	Mainly reading	Reading and writing

Figure 7: Comparison between OLAP and OLTP

On the other hand, the calculation of derived data is crucial to designing OLAP databases.

We have already mentioned two approaches to the multidimensional modelling as a method supporting so-called data centric data processing. Examples of both

[3] Among other typical OLAP features we also include visualizing aggregations in a graphical way.

approaches from a car business subject area are in Fig. 8a and Fig. 8b respectively.

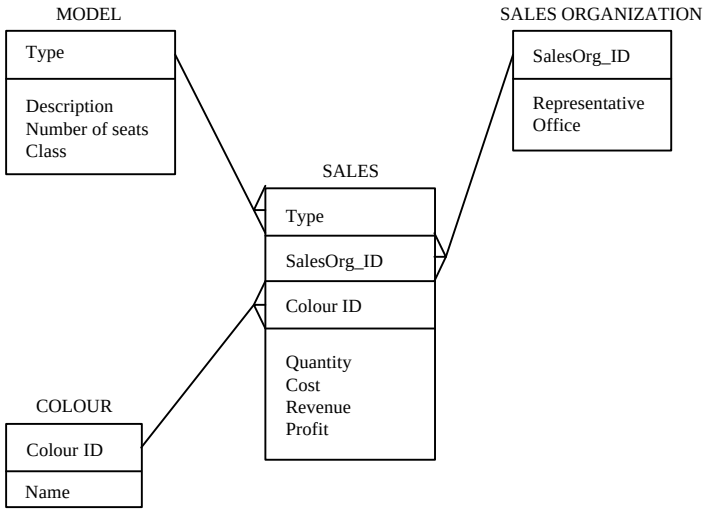

Figure 8a: Star schema

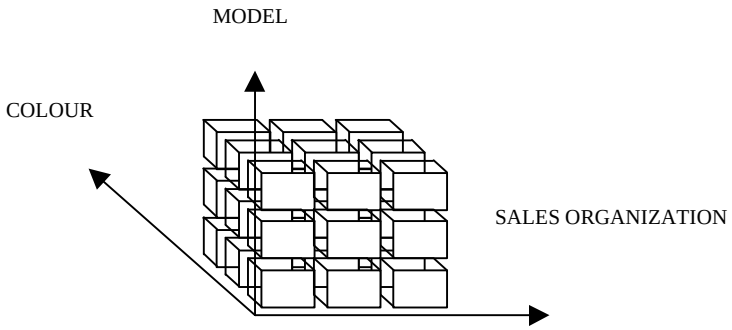

Figure 8b: Hypercube

The latter shows data in cells arranged by dimension of the data. There is a close relationship between both approaches. In (Gyssens, Lakshmanan, and Subrimanian 1996), a tabular data model is developed which provides a unified apparatus to star-like multidimensional databases and to hypercubes. We will focus rather on the first approach.

5 Multidimensional Modelling

Multidimensional or, shortly, *dimensional modelling* (DM) is a logical design technique that uses the relational data model with some important restrictions. The basic components of DM are facts, dimensions, attributes. Similarly to the E-R model, which has many variants, DM has also no referential set of constructs. The same we can say about its formal fundamentals.

We could begin with the approach advocated by R. Kimball, the author of the seminal book (Kimball 1996) and the material (Kimball 1997). Each dimensional model is composed of one table with a multi-part key, called the *fact table*, and a set of tables called *dimension tables*. Each dimension table has a single-part primary key that corresponds exactly to one of the components of the multi-part key in the fact table. Tables are conceived here in a similar way as the tables in SQL, e.g. each table has rows whose components are elements of associated column domains. As usually, each table is a subset of the Cartesian product of its column domains. The set of tables is also consistent in the sense of the referential integrity induced by its star schema.

The following definitions highlight the clean distinction made in the general conceptual modelling between table schemes and tables (the latter are sometimes called table instances or states in the literature). We use upper letters D, F, … for table schemes and D^*,F^*, … for tables.

The schema expressing mentioned tables has a characteristic ”star-like” structure and is called the *star schema*. For its associated diagram we will use the term *DM-diagram* in the multidimensional conceptual modelling (see, e.g. Fig. 8a). Table schemes are connected with lines, which express many-to-many cardinalities of relationships that are modelled by the fact table. In terminology of the E-R model, the notion of star schema is used only for special types of relationships.

5.1 Preliminaries

More formally, a *star schema* is a triple <**D, F, CC**>, where **D** is a set of *dimension table schemes* D_i with attributes A_i, i=1,...,n. **F** is a *fact table schema*, and **CC** is a set of cardinality constraints. One attribute of each table D_i is called the *key* of D_i and is denoted as KD_i. The key of F table is the union of KD_i, i=1,...,n. Other (non-key) attributes of F are called *facts*.

The *cardinality constraint* CC_i for F and D_i, i=1,...,n, is defined as follows. Let F^* and D_i^* be a fact and a dimension table, respectively. Then the cardinality constraint is *satisfied* by these tables, when for each row *u* from F^* there is only one row *v* in D_i^*, such that

$$u.KD_i = v.KD_i$$

Informally, rows of the fact and dimension tables are in many-to-one relationship. In more precise notation could be this fact expressed with the help of min-max pairs as

<F:(1,1), D_i:(0,n)>

i.e., some rows from D_i^* are associated with no row from F^*. Thus, dimensions are independent on facts, facts can not exist without dimensions. Cardinality constraints also imply that each KD is a foreign key in F. In contrary to the SQL, the value of any foreign key in F^* must not be NULL.

Now we can define a multidimensional database. Let S be a star schema. A *multidimensional database* over S is a set of tables D_i^*, i=1,…,n, and F^* that satisfy all cardinality constraints from **CC**.

Single-star schema environment is regarded as easy to understand but also a little limited. In the OLAP environment we can distinguish another approach in which a more star scheme is defined on the conceptual level. We obtain so-called *multi-star* or, better, *constellation schema.*

Thus, one dimension table schema can be common for more fact table schemes. The notion of *multidimensional database* over a constellation schema is a natural extension of this notion specified in its previous definition.

Regardless of tables as the basic construct, DM expresses the conceptual level. Certainly, multidimensional databases can be implemented in relational databases. In the simplest DM, the basic star schema has a natural relational representation. Each dimension is described by its own table, and the facts are arranged in a single large table in which parts of the multi-part key are foreign keys referring to particular dimension tables. On the other hand, a multidimensional database can have its own implementation (e.g. RedBrick Warehouse).

5.2 Dimensions

Dimensions are the classes of descriptors of facts. If the name of the fact table is SALES, the dimensions might be COLOUR, MODEL, and SALES_ORGANIZATION. Dimensions are described by attributes some of which are descriptive, e.g. Description, Name, within the others may be included business-oriented, enterprise-specific *dimension hierarchies,* e.g.

Item → Class (H1)

An important dimension is TIME structured usually into the hierarchy

date → month → quarter → year (H2)

Other attribute hierarchy is:

Office → district → region (H3)

Attributes in a dimension hierarchy are called *members* of the hierarchy. In more rigorous approach, particular members of each such hierarchy are classes of

entities. Some hierarchies can be multiple in one dimension. For example, in the TIME dimension the hierarchy date → week is a separate hierarchy from (H2). An *extension* of a hierarchy can be defined e.g. as a set of trees. An example of an extension for hierarchy (H3) appears in Fig. 13.

A typical usage of hierarchies is in various possibilities of aggregation. Beginning with office we can roll up to geographically higher wholes and aggregate the associated data.

The members of dimensions can be further described by another descriptive attributes. For example, each region has a Regional_Manager, a store has a city and state, a month could be described by Ending_Date and Starting_Date. As a consequence of this approach, we can obtain highly denormalized dimension tables. In fact, attribute hierarchies define naturally sets of functional dependencies, other functional dependencies are induced by the existence of descriptive attributes. It results in the conclusion that dimension tables are not in 3NF. This does no problems because the assumption behind the star schema is that the associated database is static, i.e. no updates are performed on-line.

Star schemes contain mostly slowly changing dimensions. This fact can imply some important decisions on the implementation level of the multidimensional database.

5.3 Facts

The facts are usually numeric quantities that describe, e.g., how many cars of a given model have been sold and the money received for the cars. This numeric data can be summed when a group of fact rows is selected. Since aggregation is an additive process, it is best if facts are limited to additive, numeric values.

A special case offer factless fact tables. These tables have the set of non-key attributes empty. They record, e.g., events. In a student tracking system, each record in the fact table detects student attendance event each day. The second kind of factless fact tables is called *coverage table* with which the problem of *sparsity* can be solved. For example, the table SALES contains facts concerning the sales of cars. This table cannot answer the question ”Which models were offered that did not sell?”. The coverage table in this case keeps rows (model, sales organization ID, colour ID) representing the current offer of cars (of given models, colours, and from given sales organisations). The same effect could be reached by allowing the fact table with partial non-key attributes. For some fraction of the Cartesian product of dimension keys, values of non-key attributes (facts) would be non-defined, i.e. set up as NULL. Any multidimensional database can handle sparsity in the way in which it does not record rows where some elements of the Cartesian product give invalid combinations of values.

Storing a hierarchy into one dimension table can pose problems in DM. We would like to keep in the fact table aggregated data e.g. such as sales dollars for a region, for a given model, and for a given colour. Thus, the question is how to

construct the key of SALES_ORGANIZATION. Suppose, the table schema contains all attributes in the hierarchy, i.e. office, district, region. Then these attributes have to participate in the table primary key. So called *generated* (artificial) *keys* and a special attribute Level may be introduced. An example explaining this problem is depicted in Fig. 9. The generated key is SO_key.

SO_Key	SalesOrg_ ID	Office	Repre- sentative	District	Region	Region manager	Level
234	STO3276	BUICI	Jones	Idaho	North	Smith	Office
235	STO3189	BMI	Hover	Florida	South	Navara	Office
236	STY5478	AUD4	Archwood	Idaho	North	Smith	Office
237	STQ6781	AUD8	Seaman	Florida	South	Navara	Office
238	NULL	NULL	NULL	Florida	South	Navara	District
390	NULL	NULL	NULL	Idaho	North	Smith	District
240	NULL	NULL	NULL	NULL	North	Smith	Region
241	NULL	NULL	NULL	NULL	North	Smith	Region

Figure 9: Dimension table with a hidden hierarchy

Often emphasized property of star schemes is that they are built for simplicity and speed. However, the level indicator can limit its flexibility. Moreover, summary data in the fact table can yield poorer performance, dimension tables are huge. More structured approaches make it possible

- to split the fact table into more fact tables according to a dimension hierarchy,
- or, to build hierarchies as paths of separate tables.

5.4 Fact Constellation Schema

For each star schema it is possible to construct so-called *fact constellation schema*. We will observe that these structures create a proper subset of constellation schemes specified in Section 5.1. Here we only extend one star schema along one selected hierarchy (roll up). The basic fact table of this schema contains data aggregated by the lowest member of the hierarchy. For example, for the hierarchy (H3), it is office. Obviously, the generated key is not necessary in this approach. Then, particular new fact tables can be built, i.e. SALES_D and SALES R for aggregation by district and region, respectively. Fig. 10 shows adding the SALES D fact table to the original star schema. Obviously, the Level attribute is not necessary here.

Notice that the fact tables in Fig. 10 are done only for the lowest hierarchy members of dimensions remaining the DM-diagram. Similarly it is possible to extend the schema in other dimensions. But, when we need to aggregate data, e.g., by district and class, it is necessary to build other fact table.

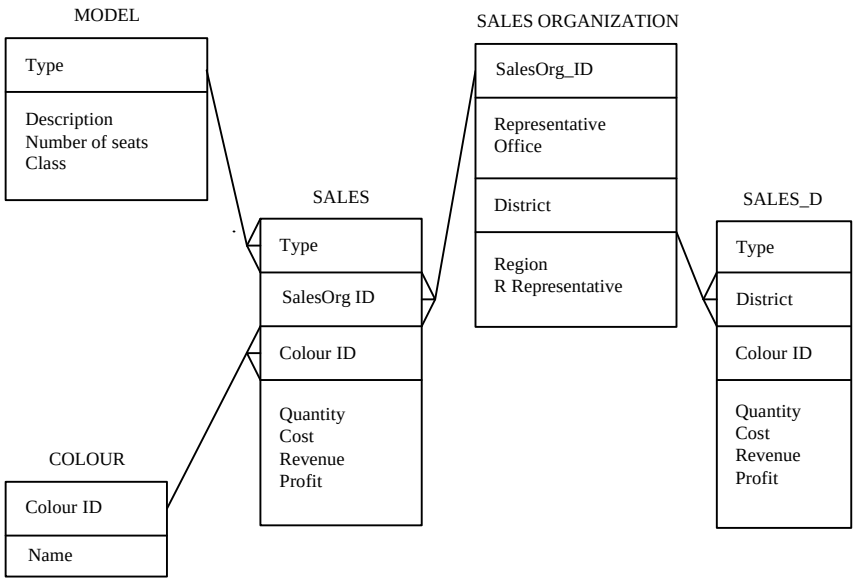

Figure 10: The fact constellation schema

The main disadvantage of the fact constellation schema is a more complicated design because many variants for particular kinds of aggregation must be considered and selected. Moreover, dimension tables are still large.

In more general approach to constellation schemes specified in Section 5.7, we will consider different fact tables built generally over different sets of dimensions.

Another alternative to the star schema is to denormalize the dimension tables according to its associated hierarchy. The fact table is split into different fact tables as before. Keys of such tables point to the smaller dimension tables. By ”snowflaking” we mean here an explosion of the original star schema into more star schemes each of them describes facts on another level of dimension hierarchies.

5.5 Constellation Schemes with Explicit Hierarchies

In (Meredith and Khader 1996), snowflakes are replaced by explicitly expressed hierarchies. Separate fact tables are again built for different kinds of aggregations accordingly to connections with appropriate members of the dimension hierarchies. An example is given in Fig. 11 Notice that SALE_C_D does not contain the fact Quantity that probably offers not useful information in this context.

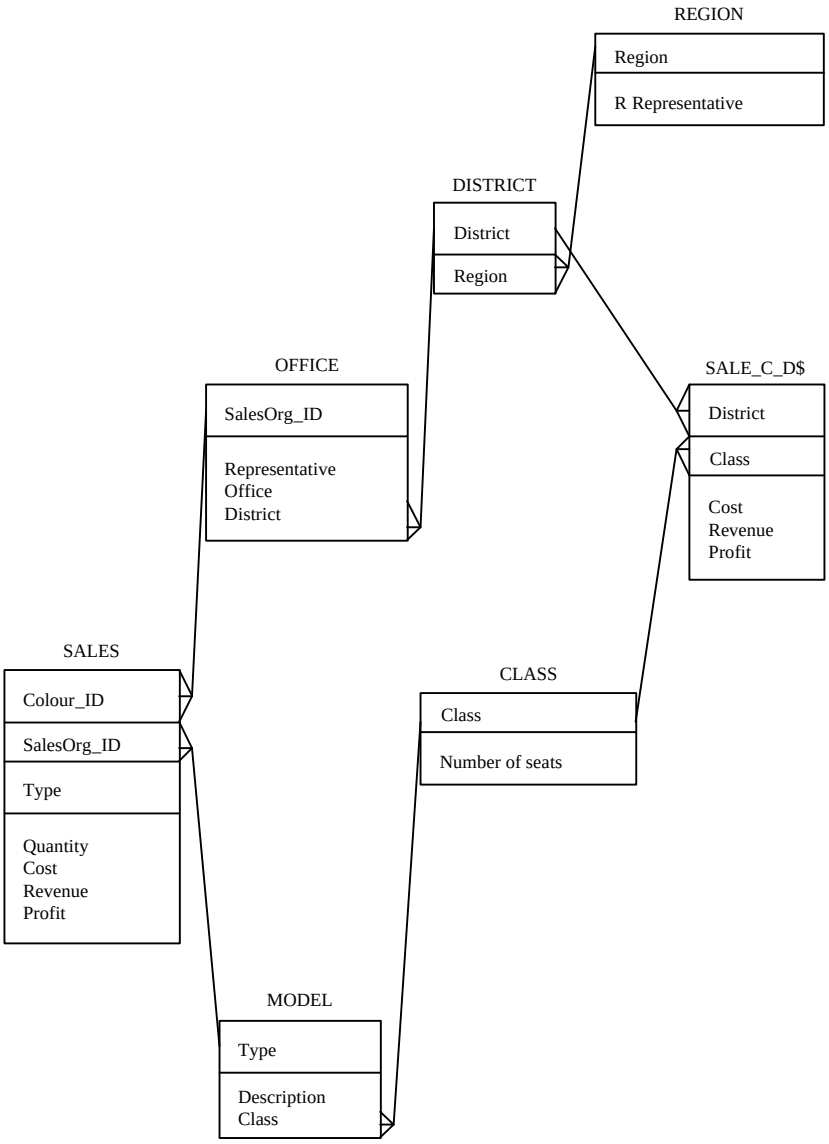

Figure 11: Explicit hierarchies of dimensions

If we create all possible aggregates, the total number of associated tables grows rapidly. For example, for two dimension hierarchies with the number of respective hierarchy members *p* and *q*, we could obtain *p* * *q* fact tables. Thus, a certain caution must be kept in deciding how to design the conceptual schema.

At first glance, it seems reasonable to try to define constellation schemes with explicit hierarchies. This approach has a number of advantages from the

conceptual modelling point of view. First, dimensions are structurally visible in the schema. Second, different fact tables are explicitly assigned to those dimensions, which are for given facts relevant. This is important, e.g. in cases when some facts are associated with days in the time dimension and facts from other fact table are associated with moths (i.e. with days they do not make of sense). We use here a definition developed in (Pokorny 1998a).

A *constellation schema with explicit dimension hierarchies* is a <**D,F,H,CC**>, where **D** is a set of *dimension table schemes* D_i with attributes A_i, i=1,…,n, **F** is a set of *fact table schemes*, **H** (dimension hierarchies) is a subset **D** x **D**, and **CC** is a set of cardinality constraints.

- Dimension tables are structured into dimension hierarchies. A *dimension hierarchy* is a sequence $\{D_{il},\ldots,D_{ik}\}$, k>1, where $(D_{ij}, Di_{i(j+1)}) \in$ **H**, j=1,…,k-1, or {D}, D ∈ **D**, such that four conditions hold:
  - (a) all dimension table schemes in the sequence are different.
  - (b) there are no two dimension tables schemes D´and D´´, such that (D', D_{il}) and (D_{ik}, D'') are in **H**,
  - (c) if (D_j, D_k) ∈ **H**, KD_k is the key of D_k, then KD_k is also an attribute of D_j,
  - (d) each element of **D** and **H** participates at least in one dimension hierarchy,
  - (e) if {D} is dimension hierarchy, then D is not a member an any couple from **H**.
- For each fact table F from **F**, there are subsets $\mathbf{D}_F \subseteq \mathbf{D}$ and $\mathbf{CC_k} \subseteq \mathbf{CC}$, such that <$\mathbf{D}_F$, F, $\mathbf{CC}_F$> is a star schema.
- The set **CC** is the union of two sets of integrity constraints IC_D and IC_F where
  - (f) IC_D is the set of cardinality constraints CC_{ij} defined for cach pair (D_i, D_j) in **H**.
  - (g) $IC_F = \cup_{F \in \mathbf{F}} \mathbf{CC}_F$.

A *multidimensional database* over a constellation scheme with explicit hierarchies S is a set of dimensional and fact tables that satisfy all cardinality constraints from **CC**.

We can observe from the condition (c) that again, KD_i in D_i is a foreign key in the same sense as there is a connection of a dimension table to a fact table. The condition (a) in the dimension hierarchy definition implies its acyclicity, the condition (b) guarantees its maximum length. With (d) we can model ”isolated” single dimensions.

This general definition supports most of meaningful situation in multidimensional conceptual modelling. Facts can be modelled on the lowest level of aggregation and any aggregates requiring other fact tables may be given explicitly or, alternatively, via view in the same way as in the SQL language.

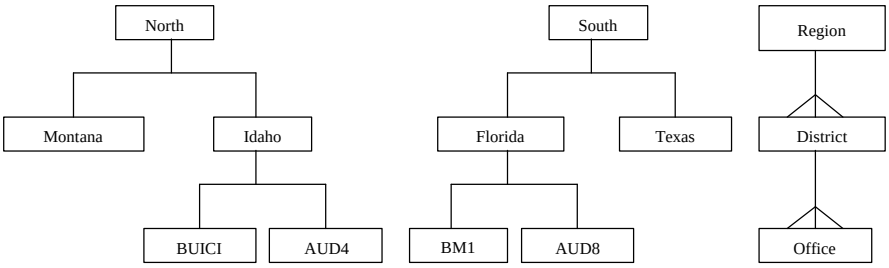

Figure 12: A dimension and its database

Fig. 12 shows a database and its scheme containing a hierarchy of dimensions. However, our definition allows to use very general dimensions. For example, a non-empty intersection of two dimensions is allowed. Fig. 13 shows four dimension hierarchies: {A, B, C, E, F}, {A, B, D, E, F}, {H, G, F}, and {H, I}.

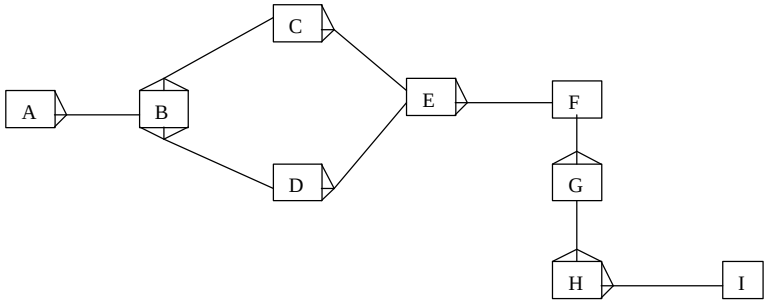

Figure 13: Dimension hierarchies

Sometimes we need subtler distinguishing properties of dimension hierarchies. In all our examples, extensions of dimension hierarchies have been unbalanced, i.e. some elements of members in higher levels of the hierarchy had no associated element on the previous level of the hierarchy. For example, elements Montana and Texas are the case in Fig. 12. We will call these dimension hierarchies *incomplete*. Dimension hierarchies that are not incomplete will be called *complete*. We can observe that simple star schemes can represent only complete hierarchies. The reason for it is easy. For example, the denormalized table SALES ORGANIZATION in Fig. 9 is not able to represent information about regions without empty primary keys. Always it is necessary to have at least one office and one district to represent a region. On the other hand, our constellation schemes approach this problem in a natural way. Tables representing a dimension hierarchy are independent down to the lowest level. It follows from a general concept of the cardinality constraint defined in Section 5.1. The relationships to E-R modelling and representation of the schemes in a relational environment are solved in (Pokorny, 1998a).

6 Conclusions

With the implementation of the data warehouse, an intuitive analysis is possible. The waiting period for the results of inquiries made of the data warehouse is reduced to a minimum. With the OLAP tools available on the market, it is not only possible to produce a presentation in the form of tables, but also a clear graphical presentation of the data.

We have proposed in this paper some fundamentals of multidimensional modelling. We have shown different approaches based on notions of fact table, dimension table, constellation schema. The formalism developed offer how to describe a wide class of conceptual multidimensional schemes and how to prove some of their properties.

In order to meet the ever increasing competitive pressure on the market, information is required from the most varied of areas. For example, it may be of great interest to compare the development on the market in another country with one's own country. The World Wide Web offers an almost bottomless pool of data. On the Internet one can find all possible data which can be taken into consideration in analyses without any problem whatsoever.

The future research could be focused on questions how to integrate multidimensional schemes, what query languages are possible to design. It seems that e.g. SQL is not too beneficial for these purposes. The other question is how to prove an information capacity of multidimensional schemes with aggregate data. A special range of questions appears in connection with various methodologies associated with different approaches to multidimensional modelling.

References

Aberdeen Group Market Viewpoint (1995): Data Warehouse Query Tools : Evolving to Relational OLAP. (1995), 8, 8.

Frye, C. (1995): Big Flap Over OLAP. Client / Server Computing (1995).

Glassey, K. (1997): The Keys to the Data Warehouse: Access Tools for End Users. Brio Technology. Inc. 1997.

Mann, C./Mehta, R. (1996): Selecting Data Warehouse End-User Access Tools. Data Management Review. July/August 1996.

Weldon, J. (1995): Managing Multidimensional Data: Harnessing the Power. Database Programming & Design, August, 1995.

Codd, E.F. et al (1993): Beyond Decision Support. Computerworld (1993), July 26.

Gyssens, M./Lakshmanan, V.S./Subramanian, I.N. (1996): Tables As a Paradigm for Quering and Restructuring. Proc. ACM Symp. On Principles of Database Systems, Montreal 1996.

Meredith, M.E./Khader, A. (1996): Designing Large Warehouses. Database Programming & Design, June 1996.

Kimball, R. (1996): The Data Warehouse Toolkit: Practical Techniques for Building Dimensional Data Warehouses. John Willey, 1996.

Kimball, R. (1997): A Dimensional Manifesto. DBMS, August 1997.

Sokolowsky, P. et al (1997): Use of Data Warehouse presented in the example of Tupperware Inc.. Proc. Of 4th International Conference System integration (Ed, J. Pour at J. Vorisek), Prague, June 1997, S. 321-333.

Pokorny, J. (1998a): Conceptual modelling in OLAP. Proc. Of 6th European Conference on Information Systems (ECIS98), (Ed. W.R.J. Baets), Aix-en-Provence, 1998, S. 273 –288.

Pokorny, J. (1998b): Data Warehouses: a Modelling Perspective. In: Evolution and Challenges in System Development (Eds. W.G.Wojtkowski, S. Wrycza,/J. **Zupanèiè),** Proc. of 7th Int. Conf. on Information Systems, Bled, Slovenia, 1999. (to appear in Plenum Press, January 1999).