12

Multidimensional Databases[1]

Torben Bach Pedersen
Aalborg University

Christian S. Jensen
Aalborg University

12.1 Introduction

The relational data model, which was introduced by Codd in 1970 and earned him the Turing Award a decade later, was the foundation of today's multi-billion-dollar database industry. During the 1990s, a new type of data model, the *multidimensional data model*, has emerged, which has taken over from the relational model where the objective is to *analyze* data, rather than to perform on-line transactions.

Multidimensional data models are designed expressly to support data analyses. A number of such models have been proposed by researchers from academia and industry. In academia, formal mathematical models have been proposed, while the industrial proposals have typically been specified more or less *implicitly* by the concrete software tools that implement them.

Briefly, multidimensional models categorize data as being either *facts* with associated numerical *measures*, or as being *dimensions* that characterize the facts and are mostly textual. For example, in a retail business, *products* are sold to *customers* at certain *times* in certain *amounts* and at certain *prices*. A typical fact would be a *purchase*. Typical measures would be the amount and price of the purchase. Typical dimensions would be the location of the purchase, the type of product being purchased, and the time of

[1]Based on "Multidimensional Database Technology" by Torben Bach Pedersen and Christian S. Jensen which appeared in IEEE Computer 34(12):40–46. ©2001 IEEE.

0-8493-1985-4/05/$0.00+$1.50

the purchase. Queries then aggregate measure values over ranges of dimension values to produce results such as the total sales per month and product type.

Multidimensional data models have three important application areas within data analysis. First, multidimensional models are used in *data warehousing*. A data warehouse is a large repository of integrated data obtained from several sources in an enterprise for the specific purpose of data analysis. Typically, these data are modeled as being multidimensional, as this best supports data analyses.

Second, multidimensional models lie at the core of *On-Line Analytical Processing* (OLAP) systems. Such systems provide fast answers to queries that aggregate large amounts of so-called detail data to find overall trends, and they present the results in a multidimensional manner. Consequently, a multidimensional data organization is ideal for OLAP.

Third, multidimensional data are increasingly becoming the basis for *data mining*, where the aim is the (semi) automatic discovery of unknown knowledge in large databases, as it turns out that multidimensionally organized databases are particularly well suited for queries posed by data mining tools.

This chapter describes fundamental concepts in multidimensional data models. It aims to communicate the essence of the multitude of proposed models in a clear and easily understood form. In doing so, we hope that the chapter will serve as an introduction of the concepts and benefits of multidimensional databases, which are well known in the database community by now, to the broader community of computer science. In-depth coverage of the features of individual models may be found elsewhere [18, 23]. The chapter also addresses implementation technologies and gives an overview of commercial systems.

12.2 Background

Related Terminology

It is useful to be familiar with a few special terms when studying the literature on issues related to multidimensional databases.

OLAP: OLAP abbreviates *On-Line Analytical Processing*. As opposed to the well-known On-Line *Transaction* Processing (OLTP), the focus is on data analyses rather than transactions. Furthermore, the analyses occur "On-Line," that is, a fast query response is required. OLAP systems always adopt a multidimensional view of data.

Data warehouse: A data warehouse (DW) is a repository of integrated enterprise data that are used specifically for decision support, that is, there is (typically) only one data warehouse in an enterprise. The data in a DW are typically collected from a large number of sources within (and sometimes also outside) the enterprise.

Data mart: A data mart (DM) is a subset of a data warehouse that is specialized to the needs of a special user group, for example, the marketing department.

ETL: Extract-Transform-Load (ETL) is the three-step process that puts data into the DW. First, data are *extracted* from the operational source systems, for example, ERP systems. Second, data are *transformed* from the source system formats into the DW format. This includes combining data from several sources and performing *cleansing* to correct errors such as missing or wrong data. Third, data are *loaded* into the DW. At times, ETL is also referred to as Extract-Transform-Transport (ETT).

Multidimensional History

Multidimensional databases do not have their origin in database technology, but stem from multidimensional matrix algebra, which has been used for (manual) data analyses since the late 19th century.

During the late 1960s, two companies, IRI and Comshare, independently began the development of systems that later evolved into multidimensional database systems. The IRI Express tool became very popular in the marketing analysis area in the late 1970s and early 1980s; it later evolved into a market-leading OLAP tool and was acquired by Oracle. Concurrently, the Comshare system evolved into System W, which was used considerably for financial planning, analysis, and reporting during the 1980s.

In 1991, Arbor was formed with the specific purpose of creating "a multiuser, multidimensional database server," which resulted in the Essbase system. Arbor, now Hyperion, later licensed a basic version of Essbase to IBM for integration into DB2. It was Arbor and Codd who in 1993 coined the term OLAP [2].

Another significant development in the early 1990s was the advent of large *data warehouses* [10], which are typically based on relational *star* or *snowflake* schemas, an approach to implementing multidimensional databases using relational database technology.

In 1998, Microsoft shipped its MS OLAP Server, the first multidimensional system aimed at the mass market. This has led to the current situation, where multidimensional systems are increasingly becoming commodity products that are shipped at no extra cost together with leading relational database management systems.

A more in-depth coverage of the history of multidimensional databases is available in the literature [21].

12.3 Spreadsheets and Relations

Assume that we want to analyze data about sales of products, for which we capture the sales price, the product sold, and the city in which it was sold. A simple example with two dimensions is shown in Table 12.1.

When considering how to analyze such data, *spreadsheets* immediately come to mind as a possibility — Table 12.1 is just a (two-dimensional) spreadsheet.

Our first analysis requirement is that we do not just want to see sales by product and city, but also the two kinds of subtotals, sales by product and sales by city, and the grand total of sales. This means that formulas for producing the (sub)totals must be added to the spreadsheet, each requiring some consideration. It is possible, if rather cumbersome, to add new data to the spreadsheet, for example, if new products are sold. Thus, for two dimensions, we can perhaps somehow manage with spreadsheets.

However, if we go to three dimensions, for example, to include time, we have to consider carefully what to do. The obvious solution is to use separate worksheets to handle the extra dimension, with one worksheet for each dimension value. This will work only when the third dimension has few dimension values and only to some extent. Analyses involving several values of the third dimension are cumbersome, and with many thousands of, say, time dimension values, the solution becomes infeasible. The situation becomes even worse if we need to support four or more dimensions, which in any case will require a very complex setup.

Another problem arises if we want to group, for example, the product into higher-level product types like "food" and "non-food." Then, we must duplicate the grouping information across all worksheets, which results in a system that uses considerable extra space and is very difficult to maintain. The essence of the problem is that spreadsheets tie the storage of data too tightly to the presentation — the *structure* and the *desired views* of the information are not separated. However, spreadsheets are good for *viewing and querying* multidimensional data, for example, using *pivot tables*. A pivot table is a two-dimensional table of data with associated subtotals and totals. For example, if we add subtotals by City and Product and a City/Product grand total to Table 12.1, we obtain an example of a pivot table. To support viewing of more complex data, several dimensions may be nested on the *x*- or *y*-axis, and data may be displayed on multiple pages, for example, one for each product. Pivot tables generally also offer support for interactively selecting subsets of the data and changing the displayed level of detail.

With spreadsheets falling short in meeting our requirements for the management of multidimensional data, we may then consider using an SQL-based, relational system for data management, as the relational

TABLE 12.1 Sales Data

	City			
Product	Aalborg	Copenhagen	Berkeley	New York
Milk	123	555	145	5001
Bread	102	250	54	2010
Jeans	20	89	32	345
Lighbulps	22	213	32	9450

model offers considerable flexibility in the modeling and querying of data. The problem here is that many desirable computations, including cumulative aggregates (sales in year to date), totals and subtotals together, and rankings (top 10 selling products), are difficult or impossible to formulate in standard SQL.

The main underlying issue is that *interrow* computations are difficult to express in SQL — only *intercolumn* computations are easy to specify. Additionally, transpositions of rows and columns are not easily possible, but rather require the manual specification and combination of multiple views. Although extensions of SQL, such as the *data cube operator* [7] and *query windows* [5], advanced by the standards bodies, will remedy some of the problems, the concept of hierarchical dimensions remains to be handled satisfactorily.

To summarize, we have seen that neither spreadsheets nor relational databases fully support the requirements posed by advanced data analyses. To be fair, these technologies may be adequate in more restricted circumstances. For example, if we have only a few dimensions, do not need hierarchical dimensions, and the data volume is small, spreadsheets may provide adequate support. However, the only robust solution to the above problems is to provide data models and database technology that offer inherent support for the full range of multidimensional concepts.

12.4 Cubes

Data cubes provide true multidimensionality. They generalize spreadsheets to any number of dimensions. In addition, hierarchies in dimensions and formulas are first-class, built-in concepts, meaning that these are supported without duplicating their definitions. A collection of related cubes is commonly referred to as a *multidimensional database* or a *multidimensional data warehouse*.

Figure 12.1 shows a cube capturing the product sales from Table 12.1 for the two Danish cities, with the additional dimension Time. The combinations of dimension values define the *cells* of the cube. The actual sales prices are stored within the corresponding cells.

In a cube, dimensions are first-class concepts with associated domains, meaning that the addition of new dimension values is easily handled. Although the term "cube" implies three dimensions, a cube can have any number of dimensions. It turns out that most real-world cubes have 4–12 dimensions [10, 21]. Although there is no theoretical limit to the number of dimensions, current tools often experience performance problems when the number of dimensions is more than 10–15. To better illustrate the high number of dimensions, the term "hypercube" is often used instead of "cube."

FIGURE 12.1 Sales cube.

Depending on the specific application, a highly varying percentage of the cells in a cube are nonempty, meaning that cubes range from *sparse* to *dense*. Cubes tend to become increasingly sparse with increasing dimensionality and with increasingly finer granularities of the dimension values.

A nonempty cell is called a *fact*. The example has a fact for each combination of time, product, and city where at least one sale was made. A fact has a number of *measures* associated with it. These are numerical values that "live" within the cells. In our case, we have one measure, the sales price.

Generally, only two or three dimensions may be viewed at the same time, although for low-cardinality dimensions, up to four dimensions can be shown by nesting one dimension within another on the axes. Thus, the dimensionality of a cube is reduced at query time by *projecting* it down to two or three dimensions via *aggregation* of the measure values across the projected-out dimensions. For example, if we want to view just sales by City and Time, we aggregate over the entire Product dimension for each combination of City and Time. In our example, we obtain the total sales for Copenhagen in 2001 by adding 127 and 211.

An important goal of multidimensional modeling is to "provide as much context as possible for the facts" [10]. The concept of *dimension* is the central means of providing this context. One consequence of this is a different view on *data redundancy* than in relational databases. In multidimensional databases, controlled redundancy is generally considered appropriate, as long as it considerably increases the information value of the data. One reason to allow redundancy is that multidimensional data are often *derived* from other data sources, for example, data from a transactional relational system, rather than being "born" as multidimensional data, meaning that updates can be handled more easily [10]. However, there is usually no redundancy in the facts, only in the dimensions.

Having introduced the cube, we describe its principal elements, dimensions, facts, and measures, in more detail.

12.5 Dimensions

The notion of a dimension is an essential and distinguishing concept in multidimensional databases. Dimensions are used for two purposes: *selection* of data and *grouping* of data at the desired level of detail.

A dimension is organized into a containment-like hierarchy, composed of a number of *levels* that each represent a level of detail that is of interest to the analyses to be performed. The instances of the dimension are typically called *dimension values*. Each such value belongs to a particular level. In some cases, it is advantageous for a dimension to have *multiple hierarchies* defined on it, for example, a Time dimension that has hierarchies for both *Fiscal Year* and *Calendar Year*. Multiple hierarchies share one or more common lowest level(s), for example, Day and Month, and then group these into multiple levels higher up, for example, Fiscal Quarter and Calendar Quarter to allow for easy reference to several ways of grouping. Most multidimensional models allow multiple hierarchies. A dimension hierarchy is defined in the metadata of the cube, or the metadata of the multidimensional database, if dimensions can be shared. This means that the problem of duplicate hierarchy definitions as discussed in Section 12.3 is avoided.

In Figure 12.2, the schema and instances of a sample *Location* dimension for the data in Table 12.1 are shown.

The Location dimension has three levels, the City level being the lowest. City-level values are grouped into *Country*-level values, that is, countries. For example, Aalborg is in Denmark. The ⊤ ("top") level represents *all* of the dimension, that is, every dimension value is part of the ⊤ ("top") value.

In some multidimensional models, a level may have associated with it a number of *level properties* that are used to hold simple, nonhierarchical, information. For example, the package size for a given product can be a level property in the Product level of the Product dimension. This information could also be captured using an extra Package Size dimension. Using the level property has the effect of not increasing the dimensionality of the cube.

Unlike the linear spaces used in matrix algebra, there is typically no ordering and/or distance metric on the dimension values in multidimensional models. Rather, the only ordering is the containment of lower-level values in higher-level values. However, for some dimensions, for example, the Time

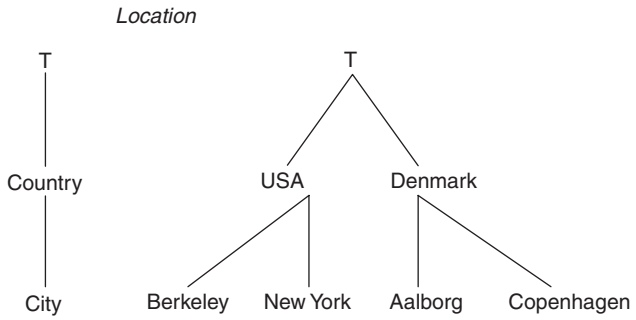

FIGURE 12.2 Schema and instances for the Location dimension.

dimension, an ordering of the dimension values is available and is used for calculating cumulative information such as "total sales in year to date."

Most models require dimension hierarchies to form *balanced trees*. This means that the dimension hierarchy must have a uniform height everywhere, for example, all departments, even small ones, must be subdivided into project groups. Additionally, direct links between dimension values can only go between immediate parent–child levels, and not jump two or more levels. For example, all cities are first grouped into states and then into countries; cities cannot be grouped directly under countries (as is the case in Denmark, which has no states). Finally, each nontop value has precisely one parent, for example, a product must belong to exactly one product group. In Section 12.10, we discuss the relaxation of these constraints.

12.6 Facts

Facts are the objects that represent the *subject* of the desired analyses, that is, the interesting "thing," or event or process, in the enterprise that is to be analyzed to better understand its behavior.

In most multidimensional data models, the facts are *implicitly* defined by their combination of dimension values. If a nonempty cell exists for a particular combination, a fact exists; otherwise, no fact exists. (Some other models treat facts as first-class objects with a separate identity [18].) Next, most multidimensional models require that each fact be mapped to precisely one dimension value at the lowest level in each dimension. Other models relax this requirement [18].

A fact has a certain *granularity*, determined by the levels from which its combination of dimension values are drawn. For example, the fact granularity in our example cube is "Year by Product by City." Granularities consisting of higher- or lower-level dimension levels than a given granularity, for example, "Year by product Type by City" or "Day by Product by City" for our example, are said to be *coarser* or *finer* than the given granularity, respectively.

It is commonplace to distinguish among three kinds of facts: *event* facts, *snapshot* facts, and *cumulative snapshot* facts [10]. Event facts (at least at the finest granularity) typically model *events in the real world*, meaning that a unique instance, for example, a particular sale of a given product in a given store at a given time, of the overall real-world process that is captured, for example, sales for a supermarket chain, is represented by one fact. Examples of event facts include sales, clicks on web pages, and movements of goods in and out of (real) warehouses (flow).

A snapshot fact models the *state* of a given process at a given point in time. Typical examples of snapshot facts include the inventory levels in stores and warehouses, and the number of users using a web site. For snapshot facts, the same object, for example, a specific can of beans on a shelf, with which the captured real-world process, for example, inventory management, is concerned may occur in several facts at different time points.

Cumulative snapshot facts are used to handle information about *a process up to a certain point in time*. For example, we may consider the total sales in year to date as a fact. Then the total sales up to and including the current month this year can be easily compared to the figure for the corresponding month last year.

Often, all three types of facts can be found in a given data warehouse, as they support complementary classes of analyses. Indeed, the same base data, for example, the movement of goods in a (real) warehouse, may often find its way into three cubes of different types, for example, warehouse flow, warehouse inventory, and warehouse flow in year-to-date.

12.7 Measures

A *measure* has two components: a *numerical property* of a fact, for example, the sales price or profit, and a *formula* (most often, a simple aggregation function such as SUM) that can be used to combine several measure values into one. In a multidimensional database, measures generally represent the properties of the chosen facts that the users want to study, for example, with the purpose of optimizing them.

Measures then take on different values for different combinations of dimension values. The property and formula are chosen such that the value of a measure is meaningful for all combinations of aggregation levels. The formula is defined in the metadata and thus not replicated as in the spreadsheet example. Although most multidimensional data models have measures, some do not. In these, dimension values are also used for computations, thus obviating the need for measures, but at the expense of some user-friendliness [18].

It is important to distinguish among three classes of measures, namely *additive*, *semiadditive*, and *nonadditive* measures, as these behave quite differently in computations.

Additive measure values can be combined meaningfully along any dimension. For example, it makes sense to add the total sales over Product, Location, and Time, as this causes no overlap among the real-world phenomena that caused the individual values. Additive measures occur for any kind of fact.

Semiadditive measure values cannot be combined along one or more of the dimensions, most often the Time dimension. Semiadditive measures generally occur when the fact is of type snapshot or cumulative snapshot. For example, it does not make sense to sum inventory levels across time, as the same inventory item, for example, a specific can of beans, may be counted several times, but it is meaningful to sum inventory levels across products and warehouses.

Nonadditive measure values cannot be combined along any dimension, usually because of the chosen formula. For example, this occurs when averages for lower-level values cannot be combined into averages for higher-level values. Nonadditive measures can occur for any kind of fact.

12.8 Querying

A multidimensional database naturally lends itself toward certain types of queries.

The queries known as *slice and dice* perform selections on a cube, thus reducing the cube. Their effect is similar to that of preparing an onion for cooking. For example, we may slice the cube in Figure 12.1 by considering only those cells that concern "Bread," and we may further slice this resulting slice, for example, by considering only the cells for year "2000." When we select a single value in a dimension, we, in a sense, reduce the dimensionality of the cube (strictly speaking, a degenerate dimension remains), but more general selections are also possible.

The operations known as *drill-down* and *roll-up* are inverses of each other and make use of dimension hierarchies and measures to perform aggregations. Consider the location dimension in Figure 12.2 together with the cube, where the (additive) measure is the count of sales together with the function SUM. We may roll-up from City level to Country level. This results in the measure values for all cities in the same country being combined into one by the associated formula. In our cube, values for "Aalborg" and "Copenhagen" are added to obtain a single "Denmark" value. For the "Bread" slice, this yields two cells, with values 180 (for year 2000) and 172 (for year 2001).

Slicing and dicing may be combined with drill-down and roll-up. Rolling up to the top value in a dimension corresponds in a sense (slightly different from the one above) to omitting the dimension.

When a multidimensional database consists of several cubes that share one or more dimensions, it is possible to combine the cubes via the shared dimensions, via a so-called *drill-across* operation. In terms of relational algebra, this operation performs a join.

Next, operations are also available that enable the user to manipulate the visualization of a cube, for example, by *rotating* it to make a different set of dimensions "face the user," that is, to see the data grouped by other dimensions.

Queries involving order are very important in data analyses and are thus supported by all multidimensional data analysis tools. These may order cells in results, and they may return only those cells that appear at the top or bottom of the specified order. For example, one may wish to retrieve the ten best-selling products in "Copenhagen" in the year 2000. Such queries are often referred to as *ranking* or *TOP N/BOTTOM N* queries [21].

12.9 Implementation Technologies

One may distinguish between two major approaches to implementing multidimensional databases, termed multidimensional vs. relational on-line analytical processing. We first contrast the two and then consider the latter in additional detail.

Multidimensional vs. Relational OLAP

Multidimensional OLAP (MOLAP) systems [21] store data on disk in specialized multidimensional structures. These typically include provisions for handling sparse arrays, and they apply advanced indexing and hashing to locate the data when performing queries [21]. In contrast, *Relational OLAP* (ROLAP) systems [10] use relational database technology for storing the data. In order to achieve good query performance, ROLAP systems employ specialized index structures such as bit-mapped indices [6], as well as materialized views [24]. We offer more information regarding DW performance at the end of this section.

Generally, MOLAP systems provide faster query response and more space-efficient storage, while ROLAP systems scale better in the number of facts, are more flexible with respect to cube redefinitions, and provide better support for frequent updates.

The virtues of the two approached are combined in the *Hybrid OLAP* (HOLAP) approach, which generally stores higher-level summary data using MOLAP technology, while using ROLAP technology to store the detailed data. Some HOLAP systems allow even more flexibility, giving users a choice between MOLAP and ROLAP at the level of individual cubes and/or materialized aggregates.

Relational OLAP Schemas

ROLAP is the most commonly used technology for the detailed data in the DW; hence, we go on to describe the typical schemas found in ROLAP DWs.

ROLAP implementations typically use *star* or *snowflake* schemas [10], both of which store the data in *fact tables* and *dimension tables*. A fact table holds one row for each fact in the cube. It has a column for each measure. In a row, this column then contains the measure value for the fact that the row represents. A fact table also has one column for each dimension. In a row, these columns contain foreign keys that reference dimension tables.

The difference between star and snowflake schemas lies in their handling of dimensions. A star schema has one dimension table for each dimension. The dimension table contains a key column. It also has one column for each level in the dimension. In a row, such a column contains a textual description of a dimension value at its level. Finally, the dimension table contains one column for each level property in the dimension. An example star schema for the Sales cube can be found in Figure 12.3.

The fact table of the star schema holds the sales price for one particular sale. The fact table also has a foreign key column for each of the three dimensions, Product, Location, and Time. The dimension tables have

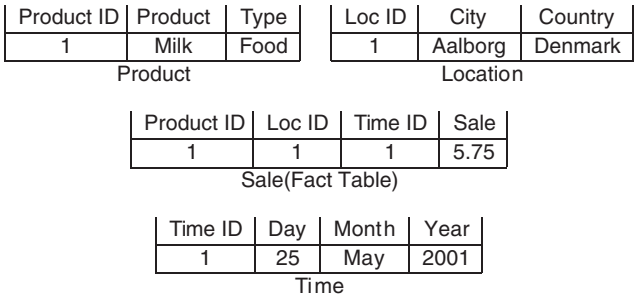

FIGURE 12.3 Star schema for sales cube.

corresponding key columns and one column for each of their levels, for example, LocID, City, and Country for Location. No column is needed for the $\top$ level, as that column would always hold the same value. The key column in a dimension table is typically a "dummy" integer key without any semantics. This has several advantages over the option of using information-bearing keys from the source systems, including better storage use, prevention of problems associated with key reuse, and better support for dimension updates [10].

It can be seen that there will be a redundancy in higher-level data. For example, with 31 day values in May 2001, the year value "2001" will be repeated 30 times for this month alone. However, as dimensions typically take up only 1–5% of the total storage required for a cube, redundancy is not a problem space-wise; and since the updates of dimensions are handled centrally, it is also possible to ensure consistency. Thus, it is often a good idea to use redundant dimension tables because this supports a simpler formulation of (better-performing) queries.

Snowflake schemas contain several dimension tables for each dimension, namely one table for each level. This means that redundancy is avoided, which may be advantageous in some situations. The dimension tables contain a key, a column having textual descriptions of the level values, and possibly columns for level properties. Tables for lower levels also contain a foreign key to the containing level.

Figure 12.4 shows a snowflake schema for the Sales cube. For example, the Day table in Figure 12.3 contains an integer key, a date, and a foreign key to the Month table. Note that no year values are replicated.

The choice of star vs. snowflake schemas depends highly on the desired properties of the system being developed. Due to space constraints, we do not provide a full discussion here.

Achieving Fast Query Response Time

The most essential performance-enhancing technique in multidimensional databases is *precomputation* and its more specialized cousin, *preaggregation*. In ROLAP systems, this is referred to as *materialized views*. This technique enables the delivery of response times to queries involving potentially large amounts of data that are fast enough to allow interactive data analysis.

As an example application of preaggregation, we may compute and store (materialize) the total sales of a product by country and month. This enables fast answers to queries that ask for the total sales, for example, by month alone, by country alone, or by quarter and country in combination. These answers may be derived from the precomputed results alone; access to the bulks of detail data in the data warehouse is unnecessary.

Preaggregation has attracted substantial attention in the research community [17], and recent versions of commercial relational database products, as well as the dedicated multidimensional systems, offer query optimization based on precomputed aggregates, as well as automatic maintenance of the stored aggregates when the detailed data on which the aggregates are based are updated [24].

Full preaggregation, where all combinations of aggregates are materialized, is infeasible as it takes too much storage and initial computation time. Instead, modern OLAP systems adopt the *practical*, or partial, preaggregation approach of materializing only select combinations of aggregates and then reuse these

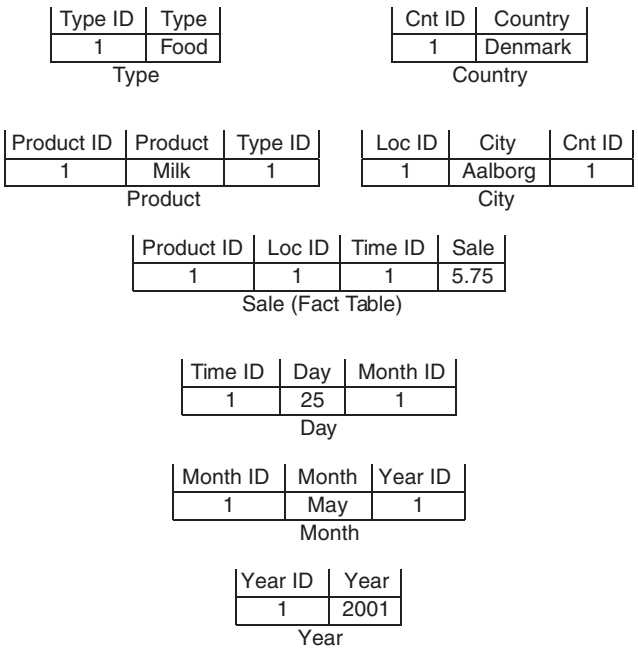

FIGURE 12.4 Snowflake schema for sales cube.

to efficiently compute other aggregates [21, 22]. This reuse of aggregates requires a well-behaved structure of the multidimensional data.

12.10 Complex Multidimensional Data

The traditional multidimensional data models and implementation techniques assume that the data being modeled conform to a quite rigid regime. Specifically, it is typically assumed that all facts map (directly) to dimension values at the lowest levels of the dimensions and only to one value in each dimension. Further, it is assumed that the dimension hierarchies are simply balanced trees. In many cases, this is adequate to support the desired applications satisfactorily. However, situations occur where these assumptions are too rigid for comfort.

In such situations, the support offered by "standard" multidimensional models and systems is inadequate, and more advanced concepts and techniques are called for. A more comprehensive treatment of complex multidimensional data is available in the literature [4]. We proceed to review the impact of irregular hierarchies on practical precomputation.

Complex multidimensional data are problematic as they are not summarizable. Intuitively, data are *summarizable* if the results of higher-level aggregates can be derived from the results of lower-level aggregates. Without summarizability, users will either obtain wrong query results, if they base them on lower-level results, or computation may be prohibitively time consuming because we cannot use precomputed lower-level results to compute higher-level results. When it is no longer possible to precompute, store, and subsequently reuse lower-level results for the computation of higher-level results, aggregates must instead be calculated directly from base data, which is what leads to the increased computational costs.

It has been shown that summarizability requires that aggregate functions be distributive and that the ordering of dimension values be *strict*, *onto*, and *covering* [11, 18]. Informally, a dimension hierarchy is *strict* if no dimension value has more than one (direct) parent, *onto* if the hierarchy is balanced, and *covering* if no containment path skips a level. Intuitively, this means that dimension hierarchies must be balanced trees. If this is not the case, some lower-level values will be either double-counted or not counted when reusing intermediate query results.

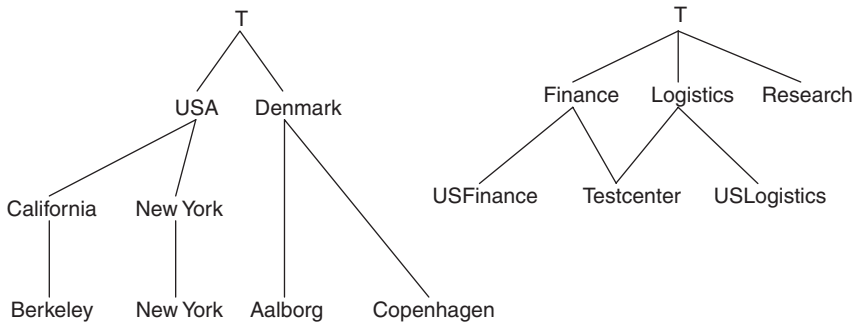

FIGURE 12.5 Irregular dimensions.

Figure 12.5 contains two dimension hierarchies: a Location dimension hierarchy including a State level, and an Organization dimension hierarchy that describes the organization in some company. The hierarchy to the left is *noncovering* because Denmark has no states. If we precompute aggregates at the State level, we will have no values for Aalborg and Copenhagen, meaning that facts mapped to these cities will not be taken into account when computing country aggregates from precomputed State-level aggregates.

The hierarchy to the right in the figure is *nononto* because the Research department has no further subdivision. If we materialize aggregates at the lowest level, facts mapping directly to the Research department will not be counted. The hierarchy is also *nonstrict* because the TestCenter is shared between Finance and Logistics. If we materialize aggregates at the middle level, data for TestCenter will be used twice, for both Finance and Logistics, which is what we want at this level. However, this means that the data will also be used twice if we combine these aggregates into the grand total.

Irregular dimension hierarchies occur in many contexts, including organization hierarchies [26], medical diagnosis hierarchies [14], and concept hierarchies for web portals [25]. A solution to the problems with irregular hierarchies is to *normalize* the hierarchies, a process that pads nononto and noncovering hierarchies with "dummy" dimension values to make them onto and covering, and fuses sets of parents in order to remedy the problems with nonstrict hierarchies. This transformation may be accomplished transparently to the user, rendering preaggregation applicable also to the more generalized types of dimension hierarchies discussed here [17].

12.11 Commercial Systems

As noted in Section 12.2, the development of multidimensional database systems began more than 35 years ago. However, it is only in the last 5–6 years that multidimensional technology has begun to enter the mainstream. Below, we introduce some of the most common commercial systems in the area. We have focused on the offerings from the "big-3" software vendors (IBM, Microsoft, Oracle) as well as the systems having a market share of at least 5% in the 2002 OLAP market, as measured by the acknowledged, independent OLAP market research report "The OLAP Report" [15].

Generally, commercial multidimensional systems fall into two groups: tools that are primarily multidimensional database servers and client-oriented analysis and reporting tools.

The multidimensional database server market is dominated by three products, each of which is backed by one of the "big-3" vendors. The number one server is Microsoft's Analysis Services [12, 22], which comes bundled with the MS SQL Server RDBMS. The number two server is Hyperion Essbase [8], long the market leader. A version of Essbase is sold along with IBM DB2 as DB2 OLAP Services [9]. The number three server is Oracle 9i OLAP R2 [16], which is based on the Oracle RDBMS and the late Express MOLAP server. All three servers offer flexible HOLAP technology, with flexible storage of data using ROLAP and/or MOLAP technology. All three servers generally provide a good query and load performance, as well as advanced query functionality.

The three largest vendors of client-oriented tools are Cognos [3], Business Objects [1], and Microstrategy [13]. They all provide tools for both "real" interactive OLAP analyses as well as more traditional static reporting, including paper-based reports. Another common functionality is support for building web portals that integrate analysis results with other data from the enterprise. They also include embedded server components that are primarily intended as back-ends for the client tools rather than as stand-alone OLAP servers.

The OLAP server vendors also have some client-side offerings for data analysis. Microsoft ships the Data Analyzer OLAP client and an Excel OLAP plug-in for OLAP analysis, and Reporting Services for traditional reports. Hyperion (IBM) has a large suite of client tools and a large market share in customized OLAP applications. Oracle offers both a range of general analysis tools and more specialized systems allowing easy analysis of data from the Oracle Applications ERP system. SAP offers the similar mySAP Business Intelligence and SAP Business Information Warehouse (BIW) products [20], which allow easy DW integration and subsequent analysis of data from the SAP ERP system, as well as from other source systems.

In summary, the most popular server and client tools generally offer sufficient functionality and performance to meet the needs of most customers. The choice of tools should thus be based on criteria such as existing system portfolio, the overall IT strategy, vendor support, and any specialized requirements, to name but a few criteria.

12.12 Summary

The continued advances in key information technology areas as well as the increasing electronic capture of business data have enabled the collection of very large volumes of business data. Advances in software technologies have contributed significantly to the provision of systems with response times that enable interactive analyses of such data.

At the core of these software technologies lies a type of data model that espouses a multidimensional view of data. While this type of model originally evolved from multidimensional matrix algebra, it has in recent years been influenced and enriched by the insights gained in the areas of semantic as well as scientific and statistical data models.

Multidimensional models view data as consisting of business facts, with associated measures, that are characterized by descriptive data values organized in multiple dimensions. Dimension values are organized in containment-type hierarchical structures that enable the computation of aggregate queries at different levels of granularity. This data organization lends itself toward graphical formulation of aggregate queries and graphical display of their results, and it is also conducive to the efficient evaluation of aggregate queries.

A wide variety of commercial systems exist that support multidimensional databases, both on the client and the server side. In recent years, multidimensional database functionality has become available as an integrated part of the leading relational database management systems.

References

[1] Business Objects Corporation, http://www.businessobjects.com/, current as of July 18th, 2003.
[2] Codd, E.F., *Providing OLAP (On-Line Analytical Processing) to User-analysts: An IT Mandate*, E.F. Codd and Assoc., Sunnyvale, CA, USA, 1993.
[3] Cognos Corporation, http://www.cognos.com/, current as of July 18th, 2003.
[4] Dyreson, C.E., T.B. Pedersen, and C.S. Jensen, Incomplete information in multidimensional databases, in *Multidimensional Databases: Problems and Solutions*, *Rafanelli, M., Ed.,* Idea Group Publishing, Hershey, PA, USA, 2003.
[5] Eisenberg, A. and J. Melton, SQL standardization: the next steps, *SIGMOD Record,* 29, 63–67, 2000.
[6] H. Garcia-Molina, J. Ullman, and J. Widom, *Database Systems: The Complete Book*, Prentice-Hall, Englewood Cliffs, NJ, 2002.

[7] J. Gray et al., Data cube: a relational aggregation operator generalizing group-by, cross-tab and sub-totals, *Data Mining and Knowledge Discovery*, 1, 29–54, 1997.
[8] Hyperion Corporation, http://www.hyperion.com/, current as of July 18th, 2003.
[9] IBM Corporation, DB2 OLAP Server, http://www.ibm.com/software/data/db2/db2olap/, current as of July 18th, 2003.
[10] R. Kimball, *The Data Warehouse Toolkit*, Wiley Computer Publishing, New York, 1996.
[11] Lenz, H. and A. Shoshani, Summarizability in OLAP and Statistical Data Bases, in *Proceedings of SSDBM*, 1997, pp. 39–48.
[12] Microsoft Corporation, SQL Server Analysis Services, http://www.microsoft.com/sql/evaluation/bi/bianalysis.asp, current as of July 18th, 2003.
[13] Microstrategy Corporation, http://www.microstrategy.com, current as of July 18th, 2003.
[14] National Health Service (NHS), Read Codes version 3, NHS, September 1999.
[15] The OLAP report, 2002 Market Shares, http://www.olapreport.com/Market.htm, current as of July 18th, 2003.
[16] Oracle Corporation, Oracle 9i OLAP R2, http://www.oracle.com/olap/, current as of July 18th, 2003.
[17] Pedersen, T.B., C.S. Jensen, and C. E. Dyreson, Extending Practical Pre-Aggregation in On-Line Analytical Processing, in *Proceedings of VLDB*, 1999, pp. 663–674.
[18] Pedersen, T.B., C.S. Jensen, and C. E. Dyreson, A foundation for capturing and querying complex multidimensional data, *Information Systems,* 26:383–423, 2001 *(Special Issue: Data Warehousing).*
[19] Pedersen T.B., and C.S. Jensen, Multidimensional database technology, *IEEE Computer Magazine, 34:40–46,* 2001 *(Special Issue: Data Warehouses).*
[20] SAP Corporation, mySAP Business Intelligence, http://www.sap.com/solutions/bi/, current as of July 18th, 2003.
[21] Thomsen, E., *OLAP Solutions: Building Multidimensional Information Systems*, Wiley, New York, 1997.
[22] Thomsen, E., G. Spofford, and D. Chase, *Microsoft OLAP Solutions*, Wiley, New York, 1999.
[23] Vassiliadis, P. and T. K. Sellis. A survey of logical models for OLAP databases., *SIGMOD Record* 28:64–69, 1999.
[24] Winter, R., Databases: back in the OLAP game, *Intelligent Enterprise Magazine*, 1:60–64, 1998.
[25] Yahoo! Corporation, www.yahoo.com, current as of July 18th, 2003.
[26] Zurek, T. and M. Sinnwell, Data Warehousing Has More Colours Than Just Black and White, in *Proceedings of VLDB*, 1999, pp. 726–729.

Torben Bach Pedersen is an associate professor of Computer Science at Aalborg University, Denmark. His research interest includes multidimensional databases, OLAP, data warehousing, federated databases, and location-based services. He received his Ph.D. degree in Computer Science from Aalborg University. He is a member of the IEEE, the IEEE Computer Society, and the ACM. He can be contacted at tbp@cs.aau.dk.

Christian S. Jensen is a professor of Computer Science at Aalborg University, Denmark. His research interests include multidimensional databases, data warehousing, temporal and spatio-temporal databases, and location-based services. He received his Ph.D. and Dr.Techn. degrees in Computer Science from Aalborg University. He is a senior member of the IEEE and a member of the IEEE Computer Society and the ACM. He can be contacted at csj@cs.aau.dk.