Multidimensional Modeling

Pedersen, Torben Bach

Published in:
Encyclopedia of Database Systems

Publication date:
2009

Document Version
Early version, also known as pre-print

Multidimensional Modeling

Torben Bach Pedersen, Aalborg University, Denmark, http://www.cs.aau.dk/ tbp

SYNONYMS

Dimensional modeling; Star schema modeling

DEFINITION

Multidimensional modeling is the process of modeling the data in a universe of discourse using the modeling constructs provided by a multidimensional data model. 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 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.

HISTORICAL BACKGROUND

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 turned 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 turned into a market-leading OLAP tool and was acquired by Oracle. Concurrently, the Comshare system developed into System W, which was heavily used for financial planning, analysis, and reporting during the 1980s.

A concurrent development started in the early 1980s in the area of so-called statistical data management which focused on modeling and managing statistical data [1], initially within social science contexts such as census data. Many important concepts of multidimensional modeling such as summarizability (ensuring correct aggregate query results for complex data) have their roots in this area. An overview is found in [15].

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 [5] for storing and analyzing massive amounts of enterprise data. Data warehouses are typically based on relational star schemas or snowflake schemas, an approach to implementing multidimensional databases using relational database technology. The 1996 version of [5] popularized the use of star schema modeling for data warehouses.

From the mid 1990s and beyond, the introduction of the "data cube" operator [4] sparked a considerable research interest in the field of modeling multidimensional databases for use in data warehouses and On-Line Analytical Processing (OLAP).

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

A more in-depth coverage of the history of multidimensional databases is available in the literature [16]. Surveys of multidimensional data models can also be found in the literature [12, 17].

SCIENTIFIC FUNDAMENTALS

First, an overview of the concept of a multidimensional cube is given, then dimensions, facts, and measures are covered in turn.

Data 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*.

A dimensional cube for, e.g., CD sales can be obtained by including additional dimensions apart from just the album and the city where the album was sold. The most pertinent example of an additional dimension is a time dimension, but it is also possible to include other dimensions, e.g., an artist dimension that describes the artists associated with albums. In a cube, the combinations of a dimension value from each dimension define the *cells* of the cube. The actual sales counts are stored in 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 3 dimensions, a cube can have any number of dimensions. It turns out that most real-world cubes have 4–12 dimensions [5, 16]. 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 suggest the high number of dimensions, the term "hypercube" is often used instead of "cube."

Figure 1 illustrates a three-dimensional cube based on the number of CD sales of two particular albums in Aalborg, Denmark, and New York, USA, for 2006 and 2007. The cube then contains sales counts for two cities, two albums, and two years.

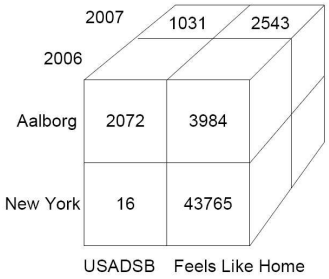

Figure 1: Sales Data Cube

Depending on the specific application, a highly varying percentage of the cells in a cube are non-empty, 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 non-empty cell is called a *fact*. The example has a fact for each combination of time, album, and city where at least one sale was made. A fact has associated with it a number of *measures*. These are numerical values that "live" within the cells. In our case, there is only one measure, the sales count.

Generally, only 2 or 3 dimensions may be viewed at the same time, although for low-cardinality dimensions, up to 4 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 2 or 3 dimensions via *aggregation* of the measure values across the projected-out dimensions. For example, if the user wants to view just sales by City and Time, she aggregates over the entire dimension that characterizes the sales by Album for each combination of City and Time.

An important goal of multidimensional modeling is to "provide as much context as possible for the facts" [5]. 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 is often *derived* from other data sources, e.g., data from a transactional relational system, rather than being "born" as multidimensional data, meaning that updates can more easily be handled [5]. However, there is usually no redundancy in the facts, only in the dimensions.

Having introduced the cube, its principal elements, dimensions, facts, and measures, are now described in more detail.

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

A dimension is organized into a containment-like hierarchy composed of a number of *levels*, each of which represents 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 may have hierarchies for both *Fiscal Year* and *Calendar Year* defined on it. Multiple hierarchies share one or more common lowest level(s), e.g., Day and Month, and then group these into multiple levels higher up, e.g., 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.

In Figure 2, the schema and instances of a sample *Location* dimension capturing the cities where CDs are sold are shown. The Location dimension has three levels, the City level being the lowest. City level values are grouped into *Country* level

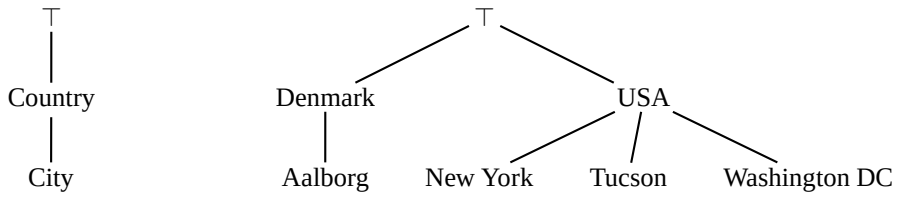

Figure 2: Schema and Instance for the Location Dimension

values, i.e., countries. For example, Aalborg is in Denmark. The ⊤ ("top") level represents *all* of the dimension, i.e., 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, non-hierarchical information. For example, the duration of an album can be a level property in the Album level of the Music dimension. This information could also be captured using an extra Duration 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, e.g., the Time 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 uniform height everywhere, e.g., 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 non-top value has precisely one parent, e.g., a product must belong to exactly one product group. Below, the relaxation of these constraints is discussed.

Facts Facts are the objects that represent the *subject* of the desired analyses, i.e., the interesting "thing," or event or process, 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 non-empty 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 [12].) 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 [12].

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 Album by City." Granularities consisting of higher-level or lower-level dimension levels than a given granularity, e.g., "Year by Album Genre by City" or "Day by Album 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, *state* facts, and *cumulative snapshot* facts [5]. Event facts (at least at the finest granularity) typically model *events in the real world*, meaning that a unique instance, e.g., a particular sale of a given (particular physical instance of a) product in a given store at a given time, of the overall real-world

process that is captured, e.g., sales for a supermarket chain, is represented by one fact. Examples of event facts include sales, clicks on web pages, and movement 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 physical object, e.g., a specific physical instance of a can of beans on a shelf, with which the captured real-world process, e.g., inventory management, is concerned, may be "measured" at several time points, meaning that data related to that particular physical object will occur in several facts at different time points. This is unlike event facts, where a specific physical object such as a particular instance of a can of beans can only be sold once, and will thus only occur in one fact.
Cumulative snapshot facts are used to handle information about *a process up to a certain point in time*. For example, the total sales in the year to date may be considered 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, e.g., the movement of goods in a (real) warehouse, may often find its way into three cubes of different types, e.g., warehouse flow, warehouse inventory, and warehouse flow in year-to-date.

Measures A *measure* has two components: a *numerical property* of a fact, e.g., 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, e.g., 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. Most multidimensional data models provide the built-in concept of measures, but a few models do not. In these models, dimension values are used for computations instead [12].
It is important to distinguish among three classes of measures, namely *additive*, *semi-additive*, and *non-additive* measures, as these behave quite differently in computations.
Additive measure values can be summed meaningfully along any dimension. For example, it makes sense to add the total sales over Album, 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.
Semi-additive measure values cannot be summed along one or more of the dimensions, most often the Time dimension. Semi-additive 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, e.g., a specific physical instance of an album, may be counted several times, but it is meaningful to sum inventory levels across albums and stores.
Non-additive measure values cannot be summed along any dimension, usually because of the chosen formula. For example, this occurs when averages for lower-level values cannot be summed into averages for higher-level values. Non-additive measures can occur for any kind of fact.

The Modeling Process Now, the process to be carried out when doing multidimensional modeling is covered. One difference from "ordinary" data modeling is that the multidimensional modeler should not try to include all the available data and all their relationships in the model, but only those parts which are essential "drivers" of the business. Another difference is that redundancy may be ok (in a few, well-chosen places) if introducing redundancy makes the model more intuitive for the user. For example, time-related information may be stored in both a Calendar time dimension and a Fiscal Year time dimension, or specific customer info may be present both in a person-oriented Customer dimension or a group-oriented Demographics dimension.
Kimball [5, 6] advocates a four-step process when doing multidimensional modeling.

Choose the business process(es) to model

2. Choose the grain of the business process
3. Choose the dimensions
4. Choose the measures

Step 1 refers to the facts that not all business processes may be equally important for the business. For example, in a supermarket, there are business processes for *sales* and *purchases*, but the sales process is probably the one with the largest potential for increasing profits, and should thus be prioritized. Step 2 says that data should be captured at the right grain,

or granularity, compared to the analysis needs. For example, "individual sales items" may be captured, or perhaps (slightly aggregated) "total sales per product per store per day" may be precise enough, enabling performance and storage gains. Step 3 then goes on to refine the schema of each part of the grain into a complete dimension with levels and attributes. For the example above, a Store, a Product, and a Time dimension are specified. Finally, Step 4 chooses the numerical measures to capture for each combination of dimension values, for example dollar sales, unit sales, dollar cost, profit, etc.

When doing multidimensional modeling "in the large" for many types of data (many cubes) and several user groups, the most important task is to ensure that analysis results are comparable across cubes, i.e., that the cubes are somehow "compatible." This is ensured by (as far as possible) picking dimensions and measures from a set of common so-called "conformed" dimensions and measures [5, 6] rather than "re-definining" the same concept, e.g., product, each time it occurs in a new context. New cubes can then be put onto the common "DW bus" [6] and used together. This sounds easier than it is, since it often requires quite a struggle with different parts of an organisation to define for example a common Product dimension that can be used by everyone.

Complex Multidimensional Modeling Multidimensional data modeling is not always as simple as described above. A complexity that is almost always present is that of handling *change* in the dimension values. Kimball [5, 6] calls this the problem of *slowly changing dimensions*. For example, customer addresses, product category names, and the way products are categorized may change over time. This must be handled to ensure correct results both for current and historical data. Kimball advises three types of slowly changing dimensions: Type 1 (overwrite previous value with current value), Type 2 (keep versions of dimension rows), and Type 3 (keep previous and current value in different columns). Finally, the concept of *minidimensions* [6] advocates the separation of relatively static information (customer name, etc) and dynamic information (income, number of kids, etc.) into separate dimensions. Please read the Data Warehouse Maintenance, Evolution, and Versioning entry for details on slowly changing dimensions.

The traditional multidimensional data models and implementation techniques assume that the data being modeled is quite regular. 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 fail.

In such situations, the support offered by "standard" multidimensional models and systems is inadequate, and more advanced concepts and techniques are called for. Now, the impact of irregular hierarchies on the performance enhancing technique known as partial, or practical, pre-computation, is reviewed.

Complex multidimensional data are problematic as they are not summarizable. Intuitively, data is *summarizable* if the results of higher-level aggregates can be derived from the results of lower-level aggregates. Without summarizability, users will either get wrong query results, if they base them on lower-level results, or the system cannot use pre-computed lower-level results to compute higher-level results. When it is no longer possible to pre-compute, store, and subsequently reuse lower-level results for the computation of higher-level results, aggregates must instead be calculated directly from base data, which leads to considerable increases in 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* [7, 12]. 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.

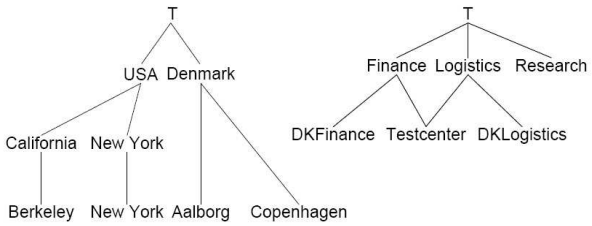

Figure 3: Irregular Dimensions

Figure 3 contains two dimension hierarchies: a Location hierarchy including a State level, and the hierarchy for the

Organization dimension for some company. The hierarchy to the left is *non-covering*, as Denmark has no states. If aggregates at the State level are pre-computed, there will be no values for Aalborg and Copenhagen, meaning that facts mapped to these cities will not be counted when computing country totals.
To the right in figure, the hierarchy is non-onto because the Research department has no further subdivision. If aggregates are materialized at the lowest level, facts mapping directly to the Research department will not be counted. The hierarchy is also non-strict as the TestCenter is shared between Finance and Logistics. If aggregates are materialized at the middle level, data for TestCenter will be counted twice, for both Finance and Logistics, which is, in fact, what is desired at this level. However, this means that data will be double-counted if these aggregates are then combined into the grand total.
There exists several design solutions that aims to solve the problems associated with irregular hierarchies by altering the dimension schemas or hierarchies [8, 11].

KEY APPLICATIONS*
Multidimensional data models have three important application areas within data analysis. First, multidimensional models are used in *data warehousing*. Briefly, 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, this data is modeled as being multidimensional, as this offers good support for 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 fashion. Consequently, a multidimensional data organization has proven to be particularly well suited for OLAP. The widely acknowledged "OLAP Report" company [9] provides an "acid test" for OLAP by defining OLAP as "Fast Analysis of Shared Multidimensional Information" (FASMI). In this definition, "Fast" refers to the expectation of response times that are within a few seconds, "Analysis" refers to the need for easy-to-use support for business logic and statistical analyses, "Shared" suggests a need for security mechanisms and concurrency control for multiple users, "Multidimensional" refers to the expectation that a data model with hierarchical dimensions is used, and "Information" suggests that the system must be able to manage all the required data and derived information.
Third, multidimensional data are increasingly becoming the basis for *data mining*, where the aim is to (semi-) automatically discover unknown knowledge in large databases. Indeed, it turns out that multidimensionally organized data are also particularly well suited for the queries posed by data mining tools.

FUTURE DIRECTIONS
A pressing need for multidimensional modeling is the aspect of standardization, i.e., agreeing on a common data model, a graphical notation for it, and support by tools. Also, better integration between ordinary "operational modeling" and multidimensional modeling is needed. Another future research line is the modeling of important system aspects such as security, quality, requirements, evolution, and interoperability [14]. This will be extended to also cover the modeling of business intelligence applications such as data mining, patterns, Extraction-Transformation-Loading (ETL), What-if Analysis, and Business Process Modeling [14]. Finally, an important line of research will cover the modeling of more (complex) types of data, including integrating multidimensional data with text data, semistructured/XML/web data and spatial/spatio-temporal/mobile data [13].

CROSS REFERENCE*
Business Intelligence; Cube; Data Warehouse; Data Warehouse Maintenance, Evolution, and Versioning; Data warehousing systems: foundations and architectures; Dimension; Hierarchy; Measure; On-Line Analytical Processing; Statistical Data Management; Summarizability; What-if Analysis;

RECOMMENDED READING
Between 5 and 15 citations to important literature, e.g., in journals, conference proceedings, and websites.

[1] P. Chan and A. Shoshani. SUBJECT: A Directory Driven System for Organizing and Accessing Large Statistical Databases. *In Proceedings of VLDB*, pp. 553–563, 1983.
[2] E. F. Codd. *Providing OLAP (on-line analytical processing) to user-analysts: An IT mandate*. E.F. Codd and Assoc, 1993.
[3] C. E. Dyreson, T. B. Pedersen, and C. S. Jensen. Incomplete information in multidimensional databases. In M. Rafanelli, editor, *Multidimensional databases: Problems and Solutions*. Idea Group Publishing, 2003.
[4] J. Gray, S. Chaudhuri, A. Bosworth, A. Layman, M. Venkatrao D. Reichart, F. Pellow, and H. Pirahesh. Data cube: A relational aggregation operator generalizing group-by, cross-tab and sub-totals. *Data Mining and Knowledge Discovery*, 1(1):29–54, 1997.
[5] R. Kimball and M. Ross. *The Data Warehouse Toolkit, 2nd Ed.*. Wiley Computer Publishing, 2002.
[6] R. Kimball et al. *The Data Warehouse Lifecycle Toolkit*. Wiley Computer Publishing, 1998.
[7] H. Lenz and A. Shoshani. Summarizability in OLAP and Statistical Data Bases. In *Proceedings of SSDBM*, pages 39–48, 1997.
[8] T. Niemi, J. Nummenmaa, and P. Thanisch. Logical Multidimensional Database Design for Ragged and Unbalanced Aggregation. In *Proceedings of DMDW*, no. 7, 2001.
[9] The OLAP Report web page. http://www.olapreport.com. Current as of November 22, 2007.
[10] T. B. Pedersen and C. S. Jensen. Multidimensional database technology. *IEEE Computer*, 34(12):40–46, 2001.
[11] T. B. Pedersen, C. S. Jensen, and C. E. Dyreson. Extending Practical Pre-Aggregation in On-Line Analytical Processing. *In Proceedings of the Twenty-Fifth International Conference on Very Large Databases*, pp. 663–674, 1999.
[12] T. B. Pedersen, C. S. Jensen, and C. E. Dyreson. A foundation for Capturing and Querying Complex multidimensional data. *Information Systems*, 26(5):383–423, 2001.
[13] T. B. Pedersen. Warehousing the World: A Few Remaining Challenges. *In Proceedings of DOLAP*, pp. TODO, 2007
[14] S. Rizzi, A. Abello, J. Lechtenbrger, and J. Trujillo. Research in Data Warehouse Modeling and Design: Dead or Alive? *In Proceedings of DOLAP*,pp. 3–10, 2006.
[15] A. Shoshani. OLAP and Statistical Databases: Similarities and Differences. *In Proceedings of PODS*, pp. 185–196, 1997.
[16] E. Thomsen. *OLAP Solutions: Building Multidimensional Information Systems*. Wiley, 1997.
[17] P. Vassiliadis and T. K. Sellis. A Survey of Logical Models for OLAP Databases. *SIGMOD Record*, 28(4):64–69, 1999.