Multidimensional Database Technology

Multidimensional databases model data as either facts, dimensions, or numerical measures for use in the interactive analysis of large amounts of data for decision-making purposes.

Torben Bach Pedersen
Christian S. Jensen
Aalborg University

The relational data model, which was introduced by E.F. Codd in 1970 and earned him the Turing award a decade later, served as the foundation of today's multibillion dollar database industry. During the past decade, the *multidimensional data model* emerged for use when the objective is to analyze data rather than to perform online transactions. Multidimensional database technology is a key factor in the interactive analysis of large amounts of data for decision-making purposes. In contrast to previous technologies, these databases view data as multidimensional *cubes* that are particularly well suited for data analysis.

Multidimensional models categorize data either as *facts* with associated numerical *measures* or as textual *dimensions* that characterize the facts. In the case of a retail business, a purchase would be a fact and the purchase amount and price would be measures; the type of product being bought and the purchase time and location would be dimensions. Queries aggregate measure values over a range of dimension values to provide results such as total sales per month of a given product. Multidimensional data models have three important application areas within data analysis.

- *Data warehouses* are large repositories that integrate data from several sources in an enterprise for analysis.
- *Online analytical processing* (OLAP) systems provide fast answers for queries that aggregate large amounts of detail data to find overall trends.
- *Data-mining* applications seek to discover knowledge by searching semiautomatically for previously unknown patterns and relationships in multidimensional databases.

Academic researchers have proposed formal mathematical models of multidimensional databases, while industry has implicitly specified proposals via the concrete software tools that implement them.[1,2] The "Multidimensional Database History" sidebar describes the evolution of the multidimensional data model and how it has benefited from the use of semantic as well as scientific and statistical data models.

SPREADSHEETS AND RELATIONS

A spreadsheet such as that shown in Table 1 is a useful tool for analyzing sales data such as product sold, number of purchases, and city of sale. A *pivot table* is a two-dimensional spreadsheet with associated subtotals and totals that supports viewing more complex data by nesting several dimensions on the *x*- or *y*-axis and displaying data on multiple pages. Pivot tables generally support interactively selecting data subsets and changing the displayed level of detail.

Spreadsheets are an inadequate tool for managing and storing multidimensional data because they tie data storage too tightly to the presentation—they do not separate the structural information from the desired views of the information. Thus, adding a third dimension such as time or grouping the data into higher-level product types requires a considerably more complex setup. The obvious solution is to use a separate spreadsheet for each dimension, but this will work only to a limited extent because analyzing the additional values of the extra dimension quickly becomes unwieldy.

0018-9162/01/$17.00 © 2001 IEEE

Using a Structured Query Language database management system offers considerable flexibility in structuring data. However, formulating many desirable computations such as cumulative aggregates (sales in year to date), combining totals and subtotals, or determining rankings such as the top 10 selling products is difficult if not impossible in standard SQL. Also, transposing rows and columns requires manually specifying and combining multiple views. Although SQL extensions such as the data cube operator[3] and query windows[4] will remedy some of these problems, the SQL-based relational model does not handle hierarchical dimensions satisfactorily.

Spreadsheets and relational databases provide adequate support for a small volume of data that has only a few nonhierarchical dimensions, but they do not fully support the requirements for advanced data analysis. The only robust solution is to use database technology that offers inherent support for the full range of multidimensional data modeling.

CUBES

Multidimensional databases view data as cubes that generalize spreadsheets to any number of dimensions. In addition, cubes support hierarchies in dimensions and formulas without duplicating their definitions. A collection of related cubes comprises a multidimensional database or data warehouse.

Because dimensions in a cube are first-class, built-in concepts with associated domains, cubes can easily manage the addition of new dimension values. Although the term implies three dimensions, a cube can theoretically have any number of dimensions; in fact, most real-world cubes have four to 12 dimensions.[3,9] Current tools often experience performance problems when a so-called hypercube contains more than 10 to 15 dimensions.

Combinations of dimension values define a cube's cells. Depending on the specific application, the cells in a cube range from sparse to dense. Cubes tend to become sparser as dimensionality increases and as the dimension values' granularities become finer.

Figure 1 shows a cube capturing the sales for the two Danish cities in Table 1 with the additional dimension of time. The corresponding cells store the number of sales. The example has a fact—a nonempty cell that contains a number of associated numerical measures—for each combination of time, product, and city where at least one sale was made. The cells store numerical values associated with a fact—in this case, the number of sales is the only measure.

Generally, a cube supports viewing only two or three dimensions simultaneously, but it can show up to four low-cardinality dimensions by nesting one dimension within another on the axes. Thus, cube dimensionality is reduced at query time by projecting it down to

Multidimensional Database History

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

During the late 1960s, IRI Software and Comshare independently began developing what later became multidimensional database systems. IRI Express, a popular tool for marketing analysis in the late 1970s and early 1980s, became a market-leading online analytical processing tool and was acquired by Oracle. Concurrently, the Comshare system developed into System W, which saw heavy use for financial planning, analysis, and reporting during the 1980s.

In 1991, Arbor Software, now Hyperion Solutions, was formed with the specific purpose of creating a multiuser, multidimensional database server, which resulted in the Essbase system. Arbor later licensed a basic version of Essbase to IBM for integration into DB2.

In 1993, E.F. Codd coined the term *OLAP*.[1] Another significant development in the early 1990s was the advent of large data warehouses, which are typically based on relational star or snowflake schemas, an approach that uses relational database technology to implement multidimensional databases.

In 1998, Microsoft shipped its MS OLAP Server, the first multidimensional system aimed at the mass market, and now multidimensional systems are becoming commodity products, shipped at no extra cost together with leading relational database systems.

Reference

1. E.F. Codd, S.B. Codd, and C.T. Salley, "Providing OLAP (On-Line Analytical Processing) to User-Analysts: An IT Mandate," http://www.hyperion.com/solutions/whitepapers.cfm (current Nov. 2001).

Table 1. Sample sales spreadsheet.

	Number of purchases			
Product	**Aalborg**	**Copenhagen**	**Los Angeles**	**New York City**
Milk	123	555	145	5,001
Bread	102	260	54	2,010
Jeans	20	89	32	345
Light bulbs	22	213	32	9,450

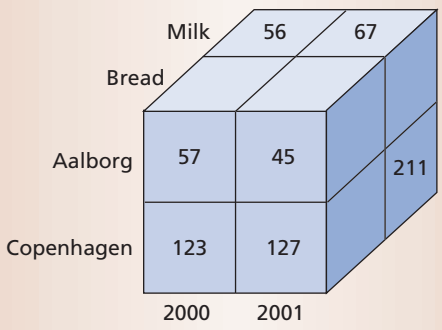

Figure 1. Sample cube capturing sales data. Data cubes support viewing of up to four low-cardinality dimensions simultaneously. In this case, the cube generalizes the spreadsheet from Table 1 to three dimensions.

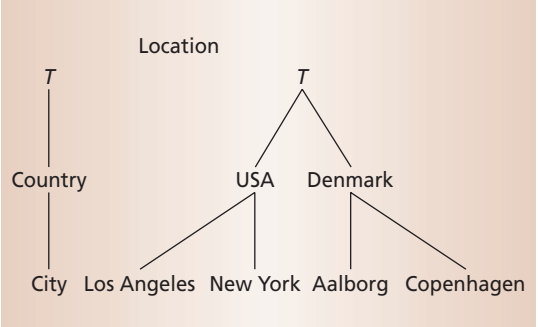

Figure 2. Sample schema and instances of the location dimension. Every dimension value is part of the* T *value.

2D or 3D by aggregating the measure values in the projected-out dimensions, resulting in higher-level measure values for the desired data view. For example, to view sales by city and time we aggregate over the entire product dimension for each combination of city and time. Thus, in Figure 1, adding 127 and 211 yields the total sales for Copenhagen in 2001.

DIMENSIONS

Dimensions are an essential and distinguishing concept in multidimensional databases. An important goal of multidimensional modeling is to use dimensions to provide as much context as possible for facts.[5] In contrast to relational databases, controlled redundancy is generally considered appropriate in multidimensional databases if it increases the data's information value. Because multidimensional cube data is often derived from other sources—for example, a transactional relational system—rather than being "born" in the multidimensional cube, the redundancy problems related to updates can be managed more readily.[5] There is usually no redundancy in the facts, only in the dimensions.

Dimensions are used for selecting and aggregating data at the desired level of detail. A dimension is organized into a containment-like hierarchy composed of numerous levels, each representing a level of detail required by the desired analyses. Each instance of the dimension, or dimension value, belongs to a particular level.

It is sometimes advantageous for multidimensional models to define multiple hierarchies for a dimension—for example, the model can define time as both fiscal year and calendar year. Multiple hierarchies share one or more common lowest levels—for example, day and month—and the model groups them into multiple levels higher up—fiscal quarter and calendar quarter—to allow easy reference to several ways of grouping. To avoid duplicating definitions, the cube or multidimensional database metadata defines the dimension hierarchy.

Figure 2 shows the schema and instances of a sample location dimension for the sales data in Table 1. Of the location dimension's three levels, City is the lowest. City-level values are grouped into country-level values—for example, Aalborg and Copenhagen are in Denmark. The ***T*** level represents all of a dimension.

In some multidimensional models, a level has a number of associated properties that hold simple, nonhierarchical information. For example, the package size can be a level property in the product dimension. A package-size dimension could also capture this information. Using the level property does not increase the cube's dimensionality.

Unlike the linear spaces used in matrix algebra, multidimensional models typically do not include ordering or distance metrics for the dimension values. Rather, the only ordering is that higher-level values contain lower-level values. However, for some dimensions such as time, an ordering of the dimension values is used to calculate cumulative information such as total sales to date. Most models require dimension hierarchies to form balanced trees—the hierarchy must have uniform height everywhere and each nontop value has precisely one parent.

FACTS

Facts represent the subject—the interesting pattern or event in the enterprise that must be analyzed to understand its behavior. In most multidimensional data models, facts are implicitly defined by their combination of dimension values; a fact exists only if there is a nonempty cell for a particular combination of values. However, some models treat facts as first-class objects with a separate identity. Most multidimensional models also require mapping each fact to one value at the lowest level in each dimension, but some models relax this mapping requirement.[1]

Each fact has a certain granularity determined by the levels from which its combination of dimension values is drawn—for example, the fact granularity of the cube in Figure 1 is year by product by city. Granularities consisting of higher- or lower-level dimension values than a given granularity—such as year by type by city and day by product by city—are coarser or finer, respectively.

Data warehouses commonly include three types of facts:[5]

- *Events,* at least at the finest granularity, typically model real-world events, with one fact representing the same instance of an underlying phenomenon. Examples include sales, clicks on a Web page, or movement of goods in and out of a warehouse.
- *Snapshots* model an entity's state at a given point in time, such as store and warehouse inventory

levels and the number of Web site users. The same instance of the underlying real-world phenomenon—such as a specific can of beans on a shelf—may occur in several facts at different time points.
- *Cumulative snapshots* handle information about activity up to a certain point in time. For example, 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.

Because they support complementary classes of analyses, a given data warehouse often contains all three types of facts. Indeed, the same base data—for example, the movement of goods in a warehouse—can be included in three different types of cubes: warehouse flow, inventory, and flow in year to date.

MEASURES

A measure consists of two components:

- a fact's numerical property, such as the sales price or profit; and
- a formula, usually a simple aggregation function such as `sum`, that can combine several measure values into one.

In a multidimensional database, measures generally represent the properties of the fact that the user wants to optimize. Measures then take on different values for various combinations of dimension values. The property and formula are chosen to provide a meaningful value for all combinations of aggregation levels. Because the metadata defines the formula, the data is not replicated as in a spreadsheet. Most multidimensional data models have measures, but some rely on using dimension values to make computations at the expense of user friendliness.[1]

Three classes of measures behave quite differently in computations:

- *Additive* measures can be meaningfully combined along any dimension. For example, it makes sense to add total sales for the product, location, and time because this causes no overlap among the real-world phenomena that generated the individual values.
- *Semi-additive* measures cannot be combined along one or more dimensions. For example, summing inventory across products and warehouses is meaningful, but summing inventory levels across time does not make sense because the same physical phenomenon could be counted several times.
- *Nonadditive* measures cannot be combined along any dimension, usually because the chosen formula prevents combining lower-level averages into higher-level averages.

Additive and nonadditive measures can occur for any kind of fact, while semi-additive measures generally occur for snapshot or cumulative snapshot facts.

In a multidimensional database, measures take on different values for various combinations of dimension values.

QUERYING

A multidimensional database naturally lends itself to certain types of queries:

- *Slice-and-dice* queries make selections to reduce a cube. For example, we can slice the cube in Figure 1 by considering only those cells that concern bread, then further reduce this slice by considering only the cells for the year 2000. Selecting a single dimension value reduces the cube's dimensionality, but more general selections are also possible.
- *Drill-down* and *roll-up* queries are inverse operations that use dimension hierarchies and measures to perform aggregations. Rolling up to its top value corresponds with omitting the dimension. For example, rolling from City to Country in Figure 2 aggregates the values for Aalborg and Copenhagen into a single value—Denmark.
- *Drill-across* queries combine cubes that share one or more dimensions. In relational algebraic terms, this operation performs a join.
- *Ranking* or top *n*/bottom *n* queries[6] can return only those cells that appear at the top or bottom of the specified order—for example, the 10 best-selling products in Copenhagen in 2000.
- *Rotating* a cube allows users to see the data grouped by other dimensions.

Drill-down and roll-up queries can be combined with slice-and-dice queries.

IMPLEMENTATION

Multidimensional database implementations take two basic forms:

- Multidimensional online analytical processing stores data on disks in specialized multidimensional structures. MOLAP systems typically include provisions for handling sparse arrays and apply advanced indexing and hashing to locate the data when performing queries.[6]
- Relational OLAP (ROLAP) systems[3] use relational database technology for storing data, and they also employ specialized index structures, such as bit-mapped indices, to achieve good query performance.

MOLAP systems generally provide more space-efficient storage as well as faster query response times.

Achieving Fast Query Response Time

The most essential performance-enhancing techniques in multidimensional databases are *precomputation* and its more specialized cousin, *preaggregation,* which enable response times to queries involving potentially huge amounts of data to be fast enough to allow interactive data analysis.

Computing and storing, or materializing, a product's total sales by country and month is one application of preaggregation. 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 can be derived entirely from the precomputed results without needing to access bulks of data in the data warehouse.

The latest versions of commercial relational database products, as well as dedicated multidimensional systems, offer query optimization based on precomputed aggregates and automatic maintenance of stored aggregates during updating of base data.[1]

Full preaggregation—materializing all combinations of aggregates—is infeasible because it takes too much storage and initial computation time. Instead, modern OLAP systems adopt the practical preaggregation approach of materializing only select combinations of aggregates and then reusing these to efficiently compute other aggregates.[2] Reusing aggregates requires a well-behaved multidimensional data structure.

References

1. R. Winter, "Databases: Back in the OLAP Game," *Intelligent Enterprise Magazine*, vol. 1, no. 4, 1998, pp. 60-64.
2. E. Thomsen, G. Spofford, and D. Chase, *Microsoft OLAP Solutions*, John Wiley & Sons, New York, 1999.

The "Achieving Fast Query Response Time" sidebar outlines some of the techniques used to accomplish this. ROLAP systems typically scale better in the number of facts they can store (although some MOLAP tools are now becoming just as scalable), are more flexible with respect to cube redefinitions, and provide better support for frequent updates. The virtues of the two approaches are combined in the hybrid OLAP approach, which uses MOLAP technology to store higher-level summary data and ROLAP systems to store the detail data.

ROLAP implementations typically employ *star* or *snowflake* schemas,[5] both of which store 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, containing the measure value for the particular fact, as well as a column for each dimension that contains a foreign key referencing a dimension table for the particular dimension.

Star and snowflake schemas differ in how they handle dimensions, and choosing between them largely depends on the desired properties of the system being developed. As Figure 3 shows, a star schema has one table for each dimension. The dimension table contains a key column, one column for each dimension level containing textual descriptions of that level's values, and one column for each level property in the dimension.

The star schema's fact table holds the sales price for one particular sale and its related dimension values. It has a foreign key column for each of the three dimensions: product, location, and time. The dimension tables have corresponding key columns and one column for each dimension level—for example, LocID, City, and Country. No column is necessary for the *T* level, which will always hold the same value. The dimension table's key column is typically a dummy integer key without any semantics. This prevents misuse of keys, offers better storage use, and provides more support for dimension updates than information-bearing keys from the source systems.[5]

Redundancy will occur in higher-level data. For example, because May 2001 has 31 day values, the year value "2001" will be repeated 30 times. Because dimensions typically only take up one to five percent of a cube's total required storage, however, redundancy is not a storage problem. Also, the central handling of dimension updates ensures consistency. Thus, using denormalized dimension tables, which support a simpler formulation of better-performing queries, is often beneficial.

Snowflake schemas contain one table for each dimension level to avoid redundancy, which may be advantageous in some situations. The dimension tables each contain a key, a column holding 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. For example, the day table in Figure 4 contains an integer key, the date, and a foreign key to the month table.

COMPLEX MULTIDIMENSIONAL DATA

Traditional multidimensional data models and implementation techniques assume that

- all facts map directly to the lowest-level dimension values and only to one value in each dimension, and
- dimension hierarchies are balanced trees.

When these assumptions fail, however, standard models and systems do not adequately support the desired applications. Complex multidimensional data is especially problematic because it is not *summarizable*—higher-level aggregate results cannot be derived from lower-level aggregate results. Queries on lower-level results will provide the wrong results or precomputing, storing, and subsequently reusing lower-level results to compute higher-level results is no longer possible. Aggregates must instead be calculated directly from base data, which considerably increases computational costs.

Summarizability requires distributive aggregate functions and *strict*, *onto*, and *covering* dimension hierarchy values.[1,7] 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. As Figure 5 shows, in the case of irregular dimensions, some lower-level values will be either double-counted or not counted when reusing intermediate query results.

Irregular dimension hierarchies occur in many contexts, including organization hierarchies,[8] medical diagnosis hierarchies,[9] and concept hierarchies for Web portals such as Yahoo! (http://www.yahoo.com). One solution is to *normalize* irregular hierarchies, a process that pads non-onto and noncovering hierarchies with dummy dimension values to make them onto and covering, and fuses sets of parents to remedy the problems with nonstrict hierarchies. This transformation can be accomplished transparently to the user.[10]

Multidimensional database technology has come a long way since its inception more than 30 years ago. It has recently begun to reach the mass market, with major vendors now delivering multidimensional engines along with their relational database offerings, often at no extra cost. Multidimensional technology has also made significant gains in scalability and maturity.

Several exiting trends lie ahead. Data that must be analyzed is becoming increasingly distributed—for example, it is often desirable to perform analyses using Extensible Markup Language data from certain Web sites. The increasing distribution of data in turn calls for techniques that easily integrate new data into multidimensional databases, thus easing the daunting task of building an integrated data warehouse. Examples include the automatic generation of dimensions and cubes from new data sources and methods for easy, on-the-fly data cleansing.

Multidimensional database technology is also being applied to new types of data that current technology often cannot adequately analyze. For example, classic techniques such as preaggregation cannot ensure fast query response times when data—such as from sensors or moving objects such as Global-Positioning-System-equipped vehicles—is continuously changing.

Finally, multidimensional database technology will increasingly be applied where analysis results are fed directly into other systems, thereby eliminating humans from the loop. When coupled with the need for continuous updates, this context poses stringent performance requirements not met by current technology. ✷

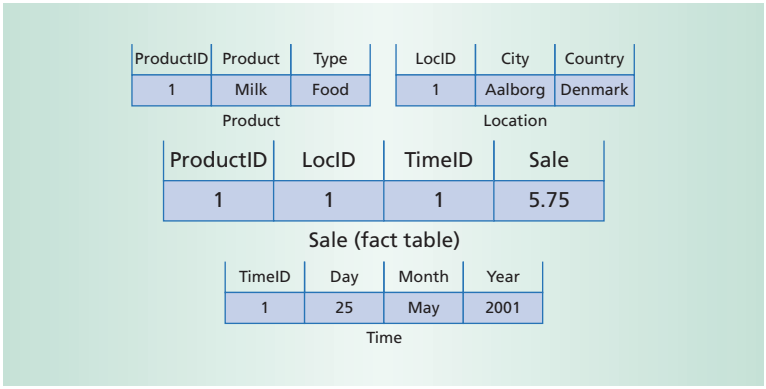

Figure 3. Star schema for sample sales cube. Information from all levels in a dimension is stored in one dimension table—for example, product names and product types are both stored in the Product dimension table.

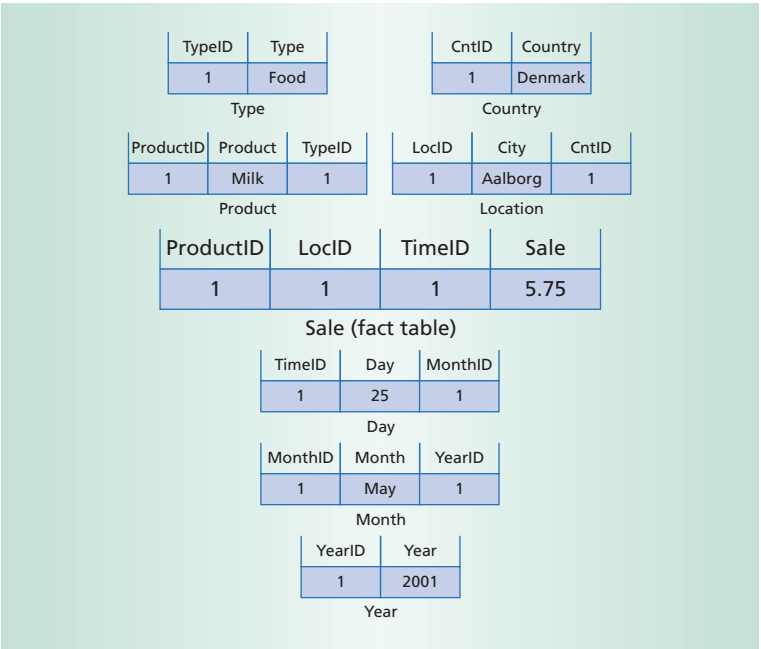

Figure 4. Snowflake schema for sample sales cube. Information from different levels in a dimension is stored in different dimension tables—for example, product names and product types are stored in the Product and Type dimension tables, respectively.

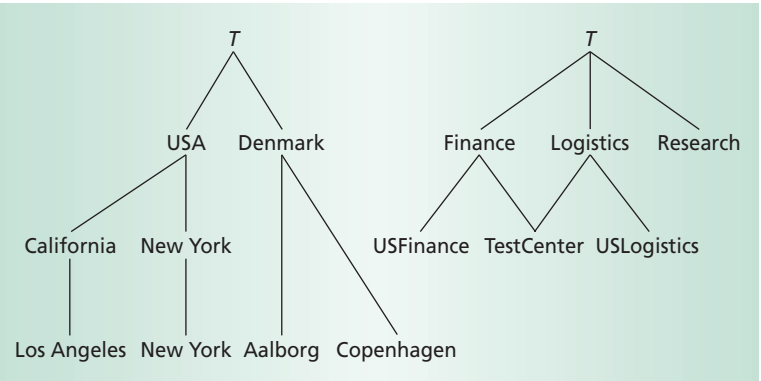

Figure 5. Irregular dimensions. The location hierarchy to the left is noncovering because Denmark has no states. The hierarchy is also nonstrict as Finance and Logistics share the TestCenter.

References

1. T.B. Pedersen, C.S. Jensen, and C.E. Dyreson, "A Foundation for Capturing and Querying Complex Multidimensional Data," *Information Systems*, vol. 26, no. 5, 2001, pp. 383-423.
2. P. Vassiliadis and T.K. Sellis, "A Survey of Logical Models for OLAP Databases," *ACM SIGMOD Record*, vol. 28, no. 4, 1999, pp. 64-69.
3. J. Gray et al., "Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab and Sub-Totals," *Data Mining and Knowledge Discovery*, vol. 1, no. 1, 1997, pp. 29-54.
4. A. Eisenberg and J. Melton, "SQL Standardization: The Next Steps," *ACM SIGMOD Record*, vol. 29, no. 1, 2000, pp. 63-67.
5. R. Kimball, *The Data Warehouse Toolkit: Practical Techniques for Building Dimensional Data Warehouses,* John Wiley & Sons, New York, 1996.
6. E. Thomsen, *OLAP Solutions: Building Multidimensional Information Systems*, John Wiley & Sons, New York, 1997.
7. H-J. Lenz and A. Shoshani, "Summarizability in OLAP and Statistical Data Bases," *Proc.9th Int'l Conf. Scientific and Statistical Database Management,* IEEE CS Press, Los Alamitos, Calif., 1997, pp. 39-48.
8. T. Zurek and M. Sinnwell, "Data Warehousing Has More Colours Than Just Black and White," *Proc. 25th Int'l Conf. Very Large Databases*, Morgan Kaufmann, San Mateo, Calif., 1999, pp. 726-729.
9. UK National Health Service, *Read Codes Version* 3, Sept. 1999, http://www.cams.co.uk/readcode.htm (current Nov. 2001).
10. T.B. Pedersen, C.S. Jensen, and C.E. Dyreson, "Extending Practical Pre-Aggregation in On-Line Analytical Processing," *Proc. 25th Int'l Conf. Very Large Databases*, Morgan Kaufmann, San Mateo, Calif., 1999, pp. 663-674.

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 a PhD in computer science from Aalborg University. He is a member of the IEEE, the IEEE Computer Society, and the ACM. Contact him at tbp@cs.auc.dk.*

Christian S. Jensen *is a professor of computer science at Aalborg University, Denmark. His research interests include multidimensional databases, data warehousing, temporal and spatiotemporal databases, and location-based services. He received a PhD and a DrTechn in computer science from Aalborg University. He is a member of the IEEE Computer Society and the ACM and is a senior member of the IEEE. Contact him at csj@cs.auc.dk.*