D

DATA WAREHOUSE

INTRODUCTION

For a few decades, the role played by database technology in companies and enterprises has only been that of storing operational data, that is data generated by daily, routine operations carried out within business processes (such as selling, purchasing, and billing). On the other hand, managers need to access quickly and reliably the strategic information that supports decision making. Such information is extracted mainly from the vast amount of operational data stored in corporate databases, through a complex selection and summarization process.

Very quickly, the exponential growth in data volumes made computers the only suitable support for the decisional process run by managers. Thus, starting from the late 1980s, the role of databases began to change, which led to the rise of *decision support systems* that were meant as the suite of tools and techniques capable of extracting relevant information from a set of electronically recorded data. Among decision support systems, data warehousing systems are probably those that captured the most attention from both the industrial and the academic world.

A typical decision-making scenario is that of a large enterprise, with several branches, whose managers wish to quantify and evaluate the contribution given from each of them to the global commercial return of the enterprise. Because elemental commercial data are stored in the enterprise database, the traditional approach taken by the manager consists in asking the database administrators to write an ad hoc query that can aggregate properly the available data to produce the result. Unfortunately, writing such a query is commonly very difficult, because different, heterogeneous data sources will be involved. In addition, the query will probably take a very long time to be executed, because it will involve a huge volume of data, and it will run together with the application queries that are part of the operational workload of the enterprise. Eventually, the manager will get on his desk a report in the form of a either summary table, a diagram, or a spreadsheet, on which he will base his decision.

This approach leads to a useless waste of time and resources, and often it produces poor results. By the way, mixing these ad hoc, analytical queries with the operational ones required by the daily routine causes the system to slow down, which makes all users unhappy. Thus, the core idea of data warehousing is to separate analytical queries, which are commonly called OLAP *(On-Line Analytical Processing)* queries, from the operational ones, called OLTP *(On-Line Transactional Processing)* queries, by building a new information repository that integrates the elemental data coming from different sources, organizes them into an appropriate structure, and makes them available for analyses and evaluations aimed at planning and decision making.

Among the areas where data warehousing technologies are employed successfully, we mention but a few: trade, manufacturing, financial services, telecommunications, and health care. On the other hand, the applications of data warehousing are not restricted to enterprises: They also range from epidemiology to demography, from natural sciences to didactics. The common trait for all these fields is the need for tools that enable the user to obtain summary information easily and quickly out of a vast amount of data, to use it for studying a phenomenon and discovering significant trends and correlations—in short, for acquiring useful knowledge for decision support.

BASIC DEFINITIONS

A *data warehousing system* can be defined as a collection of methods, techniques, and tools that support the so-called *knowledge worker* (one who works primarily with information or develops and uses knowledge in the workplace: for instance, a corporate manager or a data analyst) in decision making by transforming data into information. The main features of data warehousing can be summarized as follows:

- Easy access to nonskilled computer users.
- Data integration based on a model of the enterprise.
- Flexible querying capabilities to take advantage of the information assets.
- Synthesis, to enable targeted and effective analysis.
- Multidimensional representation to give the user an intuitive and handy view of information.
- Correctness, completeness, and freshness of information.

At the core of this process, the *data warehouse* is a repository that responds to the above requirements. According to the classic definition by Bill Inmon (see Further Reading), a data warehouse is a collection of data that exhibits the following characteristics:

1. Subject-oriented, which means that all the data items related to the same business object are connected.
2. Time-variant, which means that the history of business is tracked and recorded to enable temporal reports.
3. Nonvolatile, which means that data are read-only and never updated or deleted.
4. Integrated, which means that data from different enterprise applications are collected and made consistent.

Although operational data commonly span a limited time interval, because most business transactions only involve recent data, the data warehouse must support analyses that cover some years. Thus, the data warehouse is refreshed periodically starting from operational data. According to a common metaphor, we can imagine that photographs of operational data are periodically taken; the sequence of photos is then stored in the data warehouse, where a sort of movie is generated that depicts the history of business up to the current time.

Because in principle data are never deleted, and refreshes are made when the system is offline, a data warehouse can be considered basically as a read-only database. This feature, together with the importance given to achieving good querying performances, has two main consequences. First, the database management systems (DBMSs) used to manage the data warehouse do not need sophisticated techniques for supporting transactions. Second, the design techniques used for data warehouses are completely different from those adopted for operational databases.

As mentioned, another relevant difference between operational databases and data warehouses is related to the types of queries supported. OLTP queries on operational databases typically read and write a relatively small number of records from some tables related by simple relationships (e.g., search for customers' data to insert new orders). Conversely, OLAP queries on data warehouses commonly read a huge number of records to compute a few pices of summary information. Most importantly, although the OLTP workload is "frozen" within applications and only occasionally ad hoc queries are formulated, the OLAP workload is intrinsically interactive and dynamic.

ARCHITECTURES

To preserve the separation between transactional and analytical processing, most data warehousing architectures are based on at least two data levels: the data sources and the data warehouse.

Data sources are heterogeneous; they may be part of the corporate information system (operational databases, legacy systems, spreadsheets, flat files, etc.). or even reside outside the company (Web databases, streams, etc.). These data are extracted, cleaned, completed, validated, integrated into a single schema, and loaded into the data warehouse by the so-called *ETL (Extraction, Transformation, and Loading)* tools.

The data warehouse is the centralized repository for the integrated information. Here, different from the sources, data are stored in multidimensional form, and their structure is optimized to guarantee good performance for OLAP queries. In practice, most often, the data warehouse is replaced physically by a set of *data marts* that include the portion of information that is relevant to a specific area of business, division of the enterprise, and category of users. Note the presence of a *metadata repository* that contains the "data about data," for example, a description of the logical organization of data within the sources, the data warehouse, and the data marts.

Finally, the information in the data warehouse is accessed by users by means of different types of tools: reporting tools, OLAP tools, data-mining tools, and what-if analysis tools.

Some architectures include an additional level called the *reconciled level* or *operational data-store*. It materializes the operational data obtained by extracting and cleaning source data: Thus, it contains integrated, consistent, correct, detailed, and current data. These reconciled data are then used to feed the data warehouse directly. Although the reconciled level introduces a significant redundancy, it also bears some notable benefits. In fact, it defines a reference data model for the whole company, and at the same time, it introduces a clear separation between the issues related to data extraction, cleaning and integration and those related to data warehouse loading. Remarkably, in some cases, the reconciled level is also used to better accomplish some operational tasks (such as producing daily reports that cannot be prepared satisfactorily using the corporate applications).

In the practice, these ingredients are blended differently to give origin to the five basic architectures commonly recognized in the literature:

- Independent data marts architecture
- Bus architecture
- Hub-and-spoke architecture
- Centralized data warehouse architecture
- Federated architecture

In the *independent data mart architecture*, different data marts are designed separately and built in a nonintegrated fashion (Fig. 1). This architecture, although sometimes initially adopted in the absence of a strong sponsorship toward an enterprise-wide warehousing project or when the organizational divisions that make up the company are coupled loosely, tends to be soon replaced by other architectures that better achieve data integration and cross-reporting.

The *bus architecture* is apparently similar to the previous one, with one important difference: A basic set of conformed dimension and facts, derived by a careful analysis of the main enterprise processes, is adopted and shared as a common design guideline to ensure logical integration of data marts and an enterprise-wide view of information.

In the *hub-and-spoke architecture*, much attention is given to scalability and extensibility and to achieving an enterprise-wide view of information. Atomic, normalized data are stored in a reconciled level that feeds a set of data marts containing summarized data in multidimensional form (Fig. 2). Users mainly access the data marts, but they occasionally may query the reconciled level.

The *centralized architecture* can be viewed as a particular implementation of the hub-and-spoke architecture where the reconciled level and the data marts are collapsed into a single physical repository.

Finally, *the federated architecture* is sometimes adopted in contexts where preexisting data warehouses/data marts are to be integrated noninvasively to provide a single, cross-organization decision support environment (e.g., in the case of mergers and acquisitions). Each data warehouse/data mart is either virtually or physically integrated with the

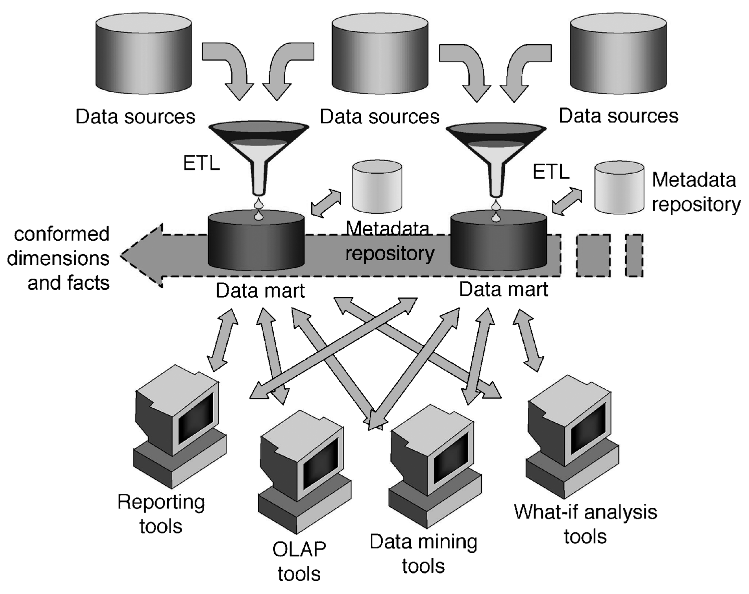

Figure 1. Independent data marts and bus architectures (without and with conformed dimensions and facts).

others by leaning on a variety of advanced techniques such as distributed querying, ontologies, and metadata interoperability.

ACCESSING THE DATA WAREHOUSE

This section discusses how users can exploit information stored in the data warehouse for decision making. In the following subsection, after introducing the particular features of the multidimensional model, we will survey the two main approaches for analyzing information: reporting and OLAP.

The Multidimensional Model

The reasons why the multidimensional model is adopted universally as the paradigm for representing data in data warehouses are its simplicity, its suitability for business analyses, and its intuitiveness for nonskilled computer users, which are also caused by the widespread use of spreadsheets as tools for individual productivity. Unfortunately, although some attempts have been made in the literature to formalize the multidimensional model (e.g., Ref. 1), none of them has emerged as a standard so far.

The multidimensional model originates from the observation that the decisional process is ruled by the *facts* of the business world, such as sales, shipments, bank transactions, and purchases. The occurrences of a fact correspond to *events* that occur dynamically: For example, every sale or shipment made is an event. For each fact, it is important to know the values of a set of *measures* that quantitatively describe the events: the revenue of a sale, the quantity shipped, the amount of a bank transaction, and the discount on a purchase.

The events that happen in the enterprise world are obviously too many to be analyzed one by one. Thus, to make them easily selectable and groupable, we imagine arranging them within an n-dimensional space whose axes, called *dimensions* of analysis, define different perspectives for their identification. Dimensions commonly are discrete, alphanumeric attributes that determine the minimum granularity for analyzing facts. For instance, the sales in a chain of stores can be represented within a three-dimensional space whose dimensions are the products, the stores, and the dates.

The concepts of dimension gave birth to the well-known *cube* metaphor for representing multidimensional data. According to this metaphor, events correspond to cells of a cube whose edges represents the dimensions of analysis. A cell of the cube is determined uniquely by assigning a value to every dimension, and it contains a value for each measure. Figure 3 shows an intuitive graphical representation of a cube centered on the sale fact. The dimensions are **product, store**, and **date**. An event corresponds to the selling of a given product in a given store on a given day, and it is described by two measures: the quantity sold and the revenue. The figure emphasizes that the cube is sparse, i.e., that several events did not happen at all: Obviously, not all products are sold every day in every store.

Normally, each dimension is structured into a *hierarchy* of *dimension levels* (sometimes called *roll-up hierarchy*) that group its values in different ways. For instance, products may be grouped according to their type and their brand, and types may be grouped additionally into categories. Stores are grouped into cities, which in turn are grouped into regions and nations. Dates are grouped into months and years. On top of each hierarchy, a final level exists that groups together all possible values of a hierarchy (all products, all stores, and all dates). Each dimension level may be described even more by one or more *descriptive attributes* (e.g., a product may be described by its name, its color, and its weight).

A brief mention to some alternative terminology used either in the literature or in the commercial tools is useful.

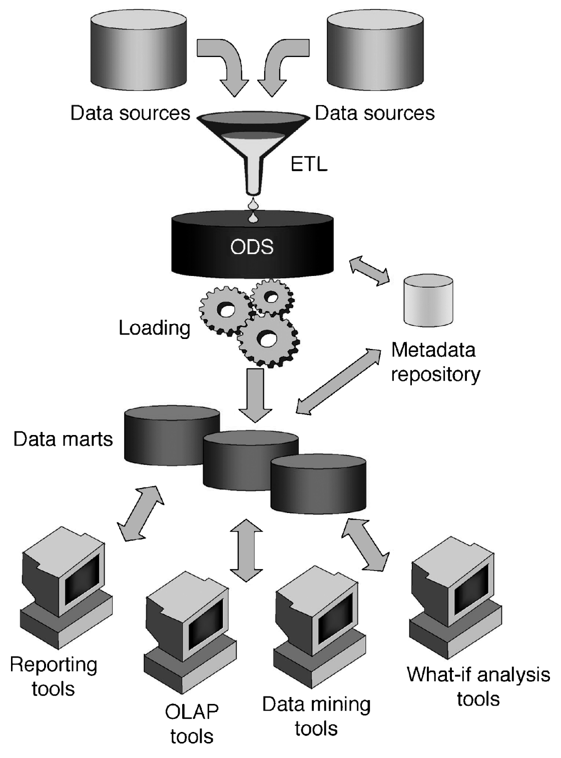

Figure 2. Hub-and-spoke architecture; ODS stands for operational data store.

Although with the term *dimension* we refer to the attribute that determines the minimum fact granularity, sometimes the whole hierarchies are named as dimensions. Measures are sometimes called *variables, metrics, categories, properties*, or *indicators*. Finally, dimension levels are sometimes called *parameters* or *attributes*.

We now observe that the cube cells and the data they contain, although summarizing the elemental data stored within operational sources, are still very difficult to analyze because of their huge number: Two basic techniques are used, possibly together, to reduce the quantity of data and thus obtain useful information: *restriction* and *aggregation*. For both, hierarchies play a fundamental role because they determine how events may be aggregated and selected.

Restricting data means cutting out a portion of the cube to limit the scope of analysis. The simplest form of restriction is *slicing*, where the cube dimensionality is reduced by focusing on one single value for one or more dimensions. For instance, as depicted in Fig. 4, by deciding that only sales of store “S-Mart” are of interest, the decision maker actually cuts a slice of the cube obtaining a two-dimensional subcube. *Dicing* is a generalization of slicing in which a subcube is determined by posing Boolean conditions on hierarchy levels. For instance, the user may be interested in sales of products of type “Hi-Fi” for the stores in Rome during the days of January 2007 (see Fig. 4).

Although restriction is used widely, aggregation plays the most relevant role in analyzing multidimensional data. In fact, most often users are not interested in analyzing events at the maximum level of detail. For instance, it may be interesting to analyze sale events not on a daily basis but by month. In the cube metaphor, this process means grouping, for each product and each store, all cells corresponding to the days of the same month into one macro-cell. In the aggregated cube obtained, each macro-cell represents a synthesis of the data stored in the cells it aggregates: in our example, the total number of items sold in each month and the total monthly revenue, which are calculated by summing the values of Quantity and Revenue through the corresponding cells. Eventually, by aggregating along the time hierarchy, an aggregated cube is obtained in which each macro-cell represents the total sales over the whole time period for each product and store. Aggregation can also be operated along two or more hierarchies. For instance, as shown in Fig. 5, sales can be aggregated by month, product type, and city.

Noticeably, not every measure can be aggregated consistently along all dimensions using the sum operator. In some cases, other operators (such as average or minimum) can be used instead, whereas in other cases, aggregation is not possible at all. For details on the two related problems of *additivity* and *summarizability*, the reader is referred to Ref. 2.

Reporting

Reporting is oriented to users who need to access periodically information structured in a fixed way. For instance, a hospital must send monthly reports of the costs of patient stays to a regional office. These reports always have the same form, so the designer can write the query that generates the report and “freeze” it within an application so that it can be executed at the users’ needs.

A report is associated with a query and a presentation. The query typically entails selecting and aggregating multidimensional data stored in one or more facts. The presentation can be in tabular or graphical form (a diagram, a histogram, a cake, etc.). Most reporting tools also allow for automatically distributing periodic reports to interested users by e-mail on a subscription basis or for posting reports in the corporate intranet server for downloading.

OLAP

OLAP, which is probably the best known technique for querying data warehouses, enables users to explore interactively and analyze information based on the multidimensional model. Although the users of reporting tools essentially play a passive role, OLAP users can define actively a complex analysis session where each step taken follows from the results obtained at previous steps. The impromptu character of OLAP sessions, the deep knowledge of data required, the complexity of the possible queries, and the orientation toward users not skilled in computer science maximize the importance of the employed tool, whose interface necessarily has to exhibit excellent features of flexibility and friendliness.

An OLAP session consists in a “navigation path” that reflects the course of analysis of one or more facts from different points of view and at different detail levels. Such a path is realized into a sequence of queries, with each differentially expressed with reference to the previous

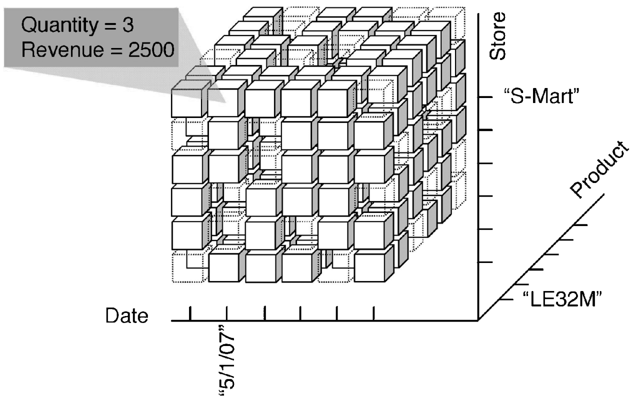

Figure 3. The three-dimensional cube that models the sales in a chain of shops. In the S-Mart store, on 5/1/2007, three LE32M TVs were sold, for a total revenue of $2500.

query. Query results are multidimensional; like for reporting, OLAP tools typically represent data in either tabular or graphical form.

Each step in the analysis session is marked by the application of an *OLAP operator* that transforms the previous query into a new one. The most common OLAP operators are as follows:

- *Roll-up*, which aggregates data even more (e.g., from sales by product, nation, and month to sales by category, nation, and year).
- *Drill-down*, which adds detail to data (e.g., from sales by category, nation, and year to sales by category, city, and year).
- *Slice-and-dice*, which selects data by fixing values or intervals for one or more dimensions (e.g., sales of products of type "Hi-Fi" for stores in Italy).
- *Pivoting*, which changes the way of visualizing the results by rotating the cube (e.g., swaps rows with columns).
- *Drill-across*, which joins two or more correlated cubes to compare their data (e.g., join the sales and the promotions cubes to compare revenues with discounts).

We close this section by observing that, in several applications, much use is made of an intermediate approach commonly called *semistatic reporting*, in which only a reduced set of OLAP navigation paths are enabled to avoid obtaining inconsistent or wrong results by incorrectly using aggregation, while allowing for some flexibility in manipulating data.

IMPLEMENTATIONS OF THE MULTIDIMENSIONAL MODEL

Two main approaches exist for implementing a data warehouse: *ROLAP*, which stands for *relational OLAP*, and *MOLAP*, which stands for *multidimensional OLAP*. Recently a third, intermediate approach has been adopted in some commercial tools: *HOLAP*, that is, *hybrid OLAP*.

Relational OLAP

On a ROLAP platform, the relational technology is employed to store data in multidimensional form. This approach is motivated by the huge research work made on the relational model, by the widespread knowledge of relational databases and their administration, and by the excellent level of performance and flexibility achieved by relational DBMSs. Of course, because the expressiveness of the relational model does not include the basic concepts of the multidimensional model, it is necessary to adopt specific schema structures that allow the multidimensional model to be mapped onto the relational model. Two main such structures are commonly adopted: the *star schema* and the *snowflake schema*.

The star schema is a relational schema composed of a set of relations called *dimension tables* and one relation called a *fact table*. Each dimension table models a hierarchy; it includes a surrogate key (i.e., a unique progressive number generated by the DBMS) and one column for each level and descriptive attribute of the hierarchy. The fact table includes a set of foreign keys, one that references each dimension table, which together define the primary key, plus one column for each measure. Figure 6 shows a star schema for the sales example. Noticeably, dimension tables are denormalized (they are not in the third normal form); this is aimed at reducing the number of relational joins to be computed when executing OLAP queries, so as to improve performance.

A snowflake schema is a star schema in which one or more dimension tables have been partially or totally normalized to reduce redundancy. Thus, a dimension table can be split into one *primary* dimension table (whose surrogate key is references by the fact table) and one or more

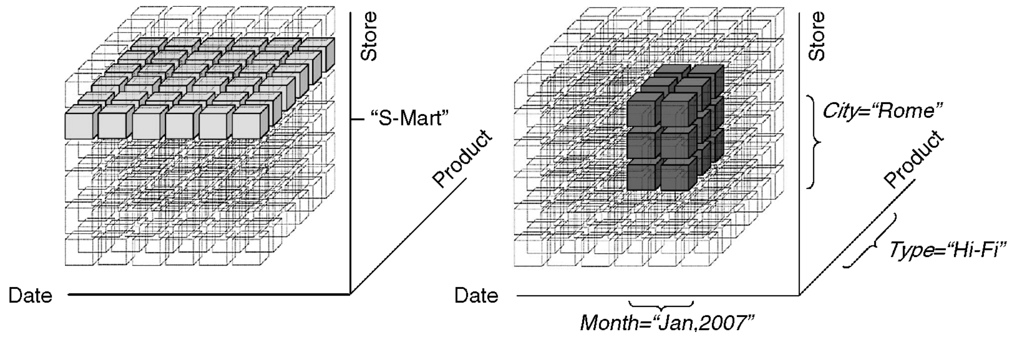

Figure 4. Slicing (left) and dicing (right) on the sales cube.

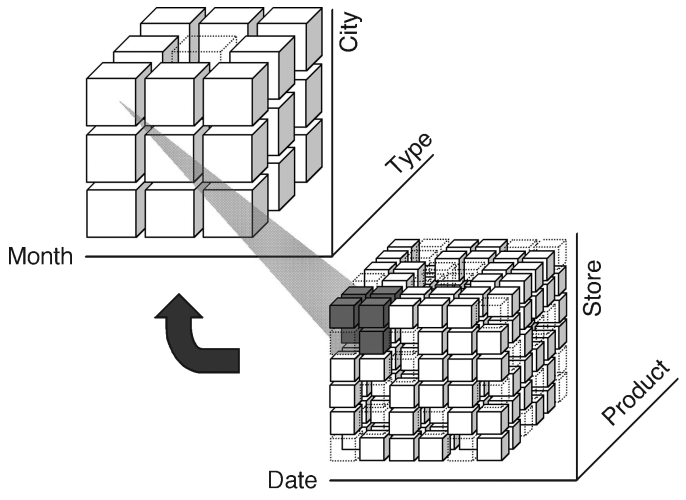

Figure 5. Aggregation on the sales cube.

secondary dimension tables (each including a surrogate key and referencing the key of another dimension table). Figure 7 shows an example for the sale schema, in which the product dimension has been normalized partially.

Multidimensional OLAP

Differently from ROLAP, a MOLAP system is based on a native logical model that directly supports multidimensional data and operations. Data are stored physically into multidimensional arrays, and positional techniques are used to access them.

The great advantage of MOLAP platforms is that OLAP queries can be executed with optimal performances, without resorting to complex and costly join operations. On the other hand, they fall short when dealing with large volumes of data, mainly because of the problem of sparseness: In fact, when a large percentage of the cube cells are empty, a lot of memory space is wasted uselessly unless ad hoc compression techniques are adopted.

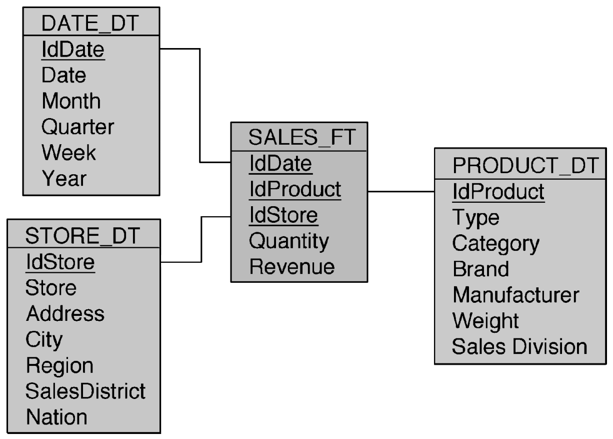

Figure 6. Star schema for the sales example (primary keys are underlined).

Hybrid OLAP

HOLAP can be viewed as an intermediate approach between ROLAP and MOLAP, because it tries to put together their advantages into a single platform. Two basic strategies are pursued in commercial tools to achieve this goal. In the first strategy, detailed data are stored in a relational database, where as a set of useful preaggregates are stored on proprietary multidimensional structures. In the second strategy, cubes are partitioned into dense and sparse subcubes, with the former being stored in multidimensional form, and the latter in relational form.

DESIGN TECHNIQUES

Despite the basic role played by a well-structured methodological framework in ensuring that the data warehouse designed fully meets the user expectations, a very few *comprehensive* design methods have been devised so far (e.g., Refs. 3 and 4). None of them has emerged as a standard, but all agree on one point: A bottom-up approach is preferable to a top-down approach, because it significantly reduces the risk of project failure. Although in a top-down approach the data warehouse is planned and designed initially in its entirety, in a bottom-up approach, it is built incrementally by designing and prototyping one data mart at a time, starting from the one that plays the most strategic business role. In general terms, the macro-phases for designing a data warehouse can be stated as follows:

- *Planning*, based on a feasibility study that assesses the project goals, estimates the system borders and size, evaluates costs and benefits, and analyzes risks and users' expectations.
- *Infrastructure design*, aimed at comparing the different architectural solutions, at surveying the available technologies and tools, and at preparing a draft design of the whole system.
- *Data mart development*, which iteratively designs, develops, tests and deploys each data mart and the related applications.

As concerns the design of each data mart, the methodology proposed in Ref. 3 encompasses eight closely related, but not necessarily strictly sequential, phases:

1. *Data source analysis*. The source schemata are analyzed and integrated to produce a reconciled schema describing the available operational data.
2. *Requirement analysis*. Business users are interviewed to understand and collect their goals and needs, so as to generate requirement glossaries and a preliminary specification of the core workload.
3. *Conceptual design*. Starting from the user requirements and from the reconciled schema, a conceptual schema that describes the data mart in an implementation-independent manner is derived.
4. *Schema validation*. The preliminary workload is better specified and tested against the conceptual schema to validate it.

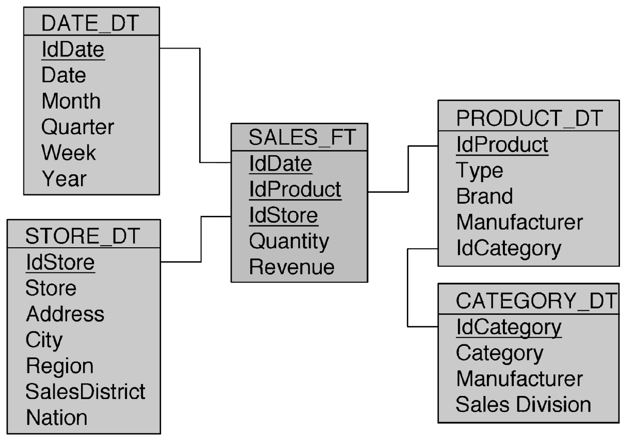

Figure 7. Snowflake schema for the sales example.

5. *Logical design*. The conceptual schema is translated into a logical schema according to the target logical model (relational or multidimensional), considering the expected workload and possible additional constraints related to space occupation and querying performances.
6. *ETL design*. The ETL procedures used to feed the data mart starting from the data sources via the reconciled level are designed.
7. *Physical design*. Physical optimization of the logical schema is done, depending on the specific characteristic of the platform chosen for implementing the data mart.
8. *Implementation*. The physical schema of the data mart is deployed, ETL procedures are implemented, and the applications for data analysis are built and tested.

Several techniques for supporting single phases of design have been proposed in the literature; a brief survey of the most relevant approaches is reported in the following subsections.

Data Source Analysis

A huge literature about schema integration has been accumulating over the last two decades. Integration methodologies have been proposed (e.g., Ref. 5), together with formalisms to code the relevant information (e.g., Ref. 6). However, the integration tools developed so far [such as TSIMMIS (7) and MOMIS (8)] should still be considered research prototypes rather than industrial tools, with the notable exception of Clio (9), which is supported by IBM.

Requirement Analysis

A careful requirement analysis is one of the keys to reduce dramatically the risk of failure for warehousing projects. From this point of view, the approaches to data warehouse design usually are classified in two categories:

- *Supply-driven* (or *data-driven*) approaches design the data warehouse starting from a detailed analysis of the data sources (e.g., Ref. 10). User requirements impact design by allowing the designer to select which chunks of data are relevant for the decision-making process and by determining their structuring according to the multidimensional model.
- *Demand-driven* (or *requirement-driven*) approaches start from determining the information requirements of business users (like in Ref. 11). The problem of mapping these requirements onto the available data sources is faced only *a posteriori*, by designing proper ETL routines, and possibly by accommodating data sources to accomplish the information needs.

A few *mixed* approaches were also devised (12,13), where requirement analysis and source inspection are carried out in parallel, and user requirements are exploited to reduce the complexity of conceptual design.

Conceptual Design

Although no agreement exists on a standard conceptual model for data warehouses, most authors agree on the importance of a conceptual design phase providing a high level of abstraction in describing the multidimensional schema of the data warehouse aimed at achieving independence of implementation issues. To this end, conceptual models typically rely on a graphical notation that facilitates writing, understanding, and managing conceptual schemata by designers and users.

The existing approaches may be framed into three categories: extensions to the entity-relationship model (e.g, Ref. 14), extensions to UML (e.g., Ref. 15), and ad hoc models (e.g., Ref. 16). Although all models have the same core expressivity, in that they all allow the basic concepts of the multidimensional model to be represented graphically, they significantly differ as to the possibility of representing more advanced concepts such as irregular hierarchies, many-to-many associations, and additivity.

Logical design

The goal of logical design is to translate a conceptual schema into a logical schema for the data mart. Although on MOLAP platforms this task is relatively simple, because the target logical model is multidimensional like the source conceptual one, on ROLAP platforms, two different models (multidimensional and relational) have to be matched. This is probably the area of data warehousing where research has focused the most during the last decade (see, for instance, Ref. 17); in particular, a lot has been written about the so-called view materialization problem.

View materialization is a well-known technique for optimizing the querying performance of data warehouses by physically materializing a set of (redundant) tables, called *views*, that store data at different aggregation levels. In the presence of materialized views, an ad hoc component of the underlying DBMS (often called *aggregate navigator*) is entrusted with the task of choosing, for each query formulated by the user, the view(s) on which the query can be answered most cheaply. Because the number of potential views is exponential in the number of hierarchy levels, materializing all views would be prohibitive. Thus, research has focused mainly on effective techniques for determining the optimal subset of views to be materialized under different kinds of constraints (e.g., Ref. 18).

Another optimization technique that is sometimes adopted to improve the querying performance is *fragmentation* (also called *partitioning* or *striping*). In particular, in *vertical* fragmentation, the fact tables are partitioned into smaller tables that contain the key and a subset of measures that are often accessed together by the workload queries (19).

ETL Design

This topic has earned some research interest only in the last few years. The focus here is to model the ETL process either from the functional, the dynamic, or the static point of view. In particular, besides techniques for conceptual design of ETL (20), some approaches are aimed at automating (21) and optimizing (22) the logical design of ETL. Although the research on ETL modeling is probably less mature than that on multidimensional modeling, it will probably have a very relevant impact on improving the overall reliability of the design process and on reducing its duration.

Physical Design

Physical design is aimed at filling the gap between the logical schema and its implementation on the specific target platform. As such, it is concerned mainly with the problem of choosing which types of indexes should be created on which columns of which tables. Like the problem of view selection, this problem has exponential complexity. A few papers on the topic can be found in the literature: For instance, Ref. 23 proposes a technique that jointly optimizes view and index selection, where as Ref. 24 selects the optimal set of indexes for a given set of views in the presence of space constraints. The problem is made even more complex by the fact that ROLAP platforms typically offer, besides classic B-trees, other types of indexes, such as star indexes, projection indexes, and bitmap indexes (25).

Note that, although some authors consider both view selection and fragmentation as part of physical design, we prefer to include them into logical design for similarity with the design of operational databases (26).

ADVANCED TOPICS

Several other topics besides those discussed so far have been addressed by the data warehouse literature. Among them we mention:

- *Query processing*. OLAP queries are intrinsically different from OLTP queries: They are read-only queries requiring a very large amount of data, taken from a few tables, to be accessed and aggregated. In addition, DBMSs oriented to data warehousing commonly support different types of specialized indexes besides B-trees. Finally, differently from the OLTP workload, the OLAP workload is very dynamical and subject to change, and very fast response times are needed. For all these reasons, the query processing techniques required by data warehousing systems are significantly different from those traditionally implemented in relational DBMSs.
- *Security*. Among the different aspects of security, confidentiality (i.e., ensuring that users can only access the information they have privileges for) is particularly relevant in data warehousing, because business information is very sensitive. Although the classic security models developed for operational databases are used widely by data warehousing tools, the particularities of OLAP applications ask for more specific models centered on the main concepts of multidimensional modeling—facts, dimensions, and measures.
- *Evolution*. The continuous evolution of the application domains is bringing to the forefront the dynamic aspects related to describing how the information stored in the data warehouse changes over time. As concerns changes in values of hierarchy data (the so-called *slowly changing dimensions*), several approaches have been devised, and some commercial systems allow us to track changes and to query cubes effectively based on different temporal scenarios. Conversely, the problem of managing changes on the schema level has only been explored partially, and no dedicated commercial tools or restructuring methods are available to the designer yet.
- *Quality*. Because of the strategic importance of data warehouses, it is absolutely crucial to guarantee their quality (in terms of data, design, technology, business, etc.) from the early stages of a project. Although some relevant work on the quality of data has been carried out, no agreement still exists on the quality of the design process and its impact on decision making.
- *Interoperability*. The wide variety of tools and software products available on the market has lead to a broad diversity in metadata modeling for data warehouses. In practice, tools with dissimilar metadata are integrated by building complex metadata bridges, but some information is lost when translating from one form of metadata to another. Thus, a need exists for a standard definition of metadata in order to better support data warehouse interoperability and integration, which is particularly relevant in the recurrent case of mergers and acquisitions. Two industry standards developed by multivendor organizations have originated in this context: the Open Information Model (OIM) by the Meta Data Coalition (MDC) and the Common Warehouse Metamodel (CWM) by the OMG.
- *New architectures and applications*. Advanced architectures for business intelligence are emerging to support new kinds of applications, possibly involving new and more complex data types. Here we cite *spatial data warehousing, web warehousing, real-time data warehousing, distributed data warehousing, and scientific data warehousing*. Thus, it becomes necessary to adapt

and specialize the existing design and optimization techniques to cope with these new applications.

See Ref. 26 for an up-to-date survey of open research themes.

BIBLIOGRAPHY

1. H. Lenz and A. Shoshani, Summarizability in OLAP and statistical data bases, *Proc. Int. Conf. on Scientific and Statistical Database Management,* Olympia, WA, 1997, pp. 132–143.
2. R. Agrawal, A. Gupta, and S. Sarawagi, Modeling multidimensional databases, IBM Research Report, IBM Almaden Research Center, 1995.
3. M. Golfarelli and S. Rizzi, A methodological framework for data warehouse design, *Proc. Int. Workshop on Data Warehousing and OLAP,* Washington DC, 1998, pp. 3–9.
4. S. Luján-Mora and J. Trujillo, A comprehensive method for data warehouse design, *Proc. Int. Workshop on Design and Management of Data Warehouses,* Berlin, Germany, 2003, pp. 1.1–1.14.
5. D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, and R. Rosati, A principled approach to data integration and reconciliation in data warehousing, *Proc. Int. Workshop on Design and Management of Data Warehouses,* Heidelberg, Germany, 1999, pp. 16.1–16.11.
6. D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, and R. Rosati, Description logic framework for information integration, *Proc. Int. Conf. on Principles of Knowledge Representation and Reasoning,* Trento, Italy, 1998, pp. 2–13.
7. S. Chawathe, H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman, and J. Widom, The TSIMMIS project: integration of heterogeneous information sources, *Proc. Meeting of the Inf. Processing Soc. of Japan,* Tokyo, Japan, 1994, pp. 7–18.
8. D. Beneventano, S. Bergamaschi, S. Castano, A. Corni, R. Guidetti, G. Malvezzi, M. Melchiori, and M. Vincini, Information integration: the MOMIS project demonstration, *Proc. Int. Conf. on Very Large Data Bases,* Cairo, Egypt, 2000, pp. 611–614.
9. L. M. Haas, M. A. Hernández, H. Ho, L. Popa, and M. Roth, Clio grows up: from research prototype to industrial tool, *Proc. SIGMOD Conf.,* Baltimore, MD, 2005, pp. 805–810.
10. M. Golfarelli, D. Maio, and S. Rizzi, The dimensional fact model: a conceptual model for data warehouses, *Int. J. Coope. Informat. Sys.*, **7**(2-3): 215–247, 1998.
11. N. Prakash and A. Gosain, Requirements driven data warehouse development, *CAiSE Short Paper Proc.,* Klagenfurt/Velden, Austria, 2003, pp. 13–16.
12. A. Bonifati, F. Cattaneo, S. Ceri, A. Fuggetta, and S. Paraboschi, Designing data marts for data warehouses, ACM Trans. Softw. *Engineer. Methodol.*, **10**(4): 452–483, 2001.
13. P. Giorgini, S. Rizzi, and M. Garzetti, Goal-oriented requirement analysis for data warehouse design, *Proc. Int. Workshop on Data Warehousing and OLAP,* Bremen, Germany, 2005, pp. 47–56.
14. C. Sapia, M. Blaschka, G. Höfling, and B. Dinter, Extending the E/R model for the multidimensional paradigm, *Proc. Int. Workshop on Design and Management of Data Warehouses, Singapore, 1998*, pp. 105–116.
15. S. Luján-Mora, J. Trujillo, and I. Song, A UML profile for multidimensional modeling in data warehouses, *Data Know. Engineer.*, **59**(3): 725–769, 2006.
16. S. Rizzi, Conceptual modeling solutions for the data warehouse, in R. Wrembel and C. Koncilia (eds.), *Data Warehouses and OLAP: Concepts, Architectures and Solutions*, IRM Press: 2006, pp. 1–26.
17. J. Lechtenbörger and G. Vossen, Multidimensional normal forms for data warehouse design, *Informat. Sys.*, **28**(5): 415–434, 2003.
18. D. Theodoratos and M. Bouzeghoub, A general framework for the view selection problem for data warehouse design and evolution, *Proc. Int. Workshop on Data Warehousing and OLAP,* Washington DC, 2000, pp. 1–8.
19. M. Golfarelli, V. Maniezzo, and S. Rizzi, Materialization of fragmented views in multidimensional databases, *Data Knowl. Engineer.*, **49**(3): 325–351, 2004.
20. P. Vassiliadis, A. Simitsis, and S. Skiadopoulos, Conceptual modeling for ETL processes, *Proc. Int. Workshop on Data Warehousing and OLAP,* McLean, 2002, pp. 14–21.
21. A. Simitsis, Mapping conceptual to logical models for ETL processes, *Proc. Int. Workshop on Data Warehousing and OLAP,* Bremen, Germany, 2005, pp. 67–76.
22. A. Simitsis, P. Vassiliadis, and T. K. Sellis, Optimizing ETL processes in data warehouses, *Proc. Int. Conf. on Data Engineering, Tokyo, Japan, 2005*, pp. 564–575.
23. H. Gupta, V. Harinarayan, A. Rajaraman, and J. Ullman, Index selection for OLAP, *Proc. Int. Conf. on Data Engineering,* Birmingham, UK, 1997, pp. 208–219.
24. M. Golfarelli, S. Rizzi, and E. Saltarelli, Index selection for data warehousing, *Proc. Int. Workshop on Design and Management of Data Warehouses,* Toronto, Canada, 2002, pp. 33–42.
25. P. O'Neil and D. Quass, Improved query performance with variant indexes, *Proc. SIGMOD Conf.,* Tucson, AZ, 1997, pp. 38–49.
26. S. Rizzi, A. Abell, J. Lechtenbörger, and J. Trujillo, Research in data warehouse modeling and design: Dead or alive? *Proc. Int. Workshop on Data Warehousing and OLAP,* Arlington, VA, 2006, pp. 3–10.

FURTHER READING

B. Devlin, *Data Warehouse: From Architecture to Implementation*, Reading, MA: Addison-Wesley Longman, 1997.

W. H. Inmon, *Building the Data Warehouse*, 4th ed. New York: John Wiley & Sons, 2005.

M. Jarke, M. Lenzerini, Y. Vassiliou, and P. Vassiliadis, *Fundamentals of Data Warehouse*, New York: Springer, 2000.

R. Kimball, L. Reeves, M. Ross, and W. Thornthwaite, *The Data Warehouse Lifecycle Toolkit*, New York: John Wiley & Sons, 1998.

R. Mattison, *Data Warehousing*, New York: McGraw-Hill, 1996.

STEFANO RIZZI
University of Bologna
Bologna, Italy