chapter 29

# Overview of Data Warehousing and OLAP

The increasing processing power and sophistication of analytical tools and techniques have resulted in the development of what are known as data warehouses. These data warehouses provide storage, functionality, and responsiveness to queries beyond the capabilities of transaction-oriented databases. Accompanying this ever-increasing power is a great demand to improve the data access performance of databases. As we have seen throughout this book, traditional databases balance the requirement of data access with the need to ensure data integrity. In modern organizations, users of data are often completely removed from the data sources. Many people only need read-access to data, but still need fast access to a larger volume of data than can conveniently be downloaded to the desktop. Often such data comes from multiple databases. Because many of the analyses performed are recurrent and predictable, software vendors and systems support staff are designing systems to support these functions. Presently there is a great need to provide decision makers from middle management upward with information at the correct level of detail to support decision making. *Data warehousing, online analytical processing* (OLAP), and *data mining* provide this functionality. We gave an introduction to data mining techniques in Chapter 28. In this chapter we give a broad overview of data warehousing and OLAP technologies.

## 29.1 Introduction, Definitions, and Terminology

In Chapter 1 we defined a *database* as a collection of related data and a *database system* as a database and database software together. A data warehouse is also a collection of information as well as a supporting system. However, a clear distinction exists. Traditional databases are transactional (relational, object-oriented, network, or hierarchical). *Data warehouses* have the distinguishing characteristic that they are mainly intended for decision-support applications. They are optimized for data retrieval, not routine transaction processing.

Because data warehouses have been developed in numerous organizations to meet particular needs, there is no single, canonical definition of the term data warehouse. Professional magazine articles and books in the popular press have elaborated on the meaning in a variety of ways. Vendors have capitalized on the popularity of the term to help market a variety of related products, and consultants have provided a large variety of services, all under the data warehousing banner. However, data warehouses are quite distinct from traditional databases in their structure, functioning, performance, and purpose.

W. H. Inmon[1] characterized a **data warehouse** as *a subject-oriented, integrated, nonvolatile, time-variant collection of data in support of management's decisions.* Data warehouses provide access to data for complex analysis, knowledge discovery, and decision making. They support high-performance demands on an organization's data and information. Several types of applications—OLAP, DSS, and data mining applications—are supported. We define each of these next.

**OLAP** (**online analytical processing**) is a term used to describe the analysis of complex data from the data warehouse. In the hands of skilled knowledge workers, OLAP tools use distributed computing capabilities for analyses that require more storage and processing power than can be economically and efficiently located on an individual desktop.

**DSS** (**decision-support systems**), also known as **EIS—executive information systems**; not to be confused with enterprise integration systems—support an organization's leading decision makers with higher-level data for complex and important decisions. Data mining (which we discussed in Chapter 28) is used for *knowledge discovery*, the process of searching data for unanticipated new knowledge.

Traditional databases support **online transaction processing** (**OLTP**), which includes insertions, updates, and deletions, while also supporting information query requirements. Traditional relational databases are optimized to process queries that may touch a small part of the database and transactions that deal with insertions or updates of a few tuples per relation to process. Thus, they cannot be optimized for OLAP, DSS, or data mining. By contrast, data warehouses are designed precisely to support efficient extraction, processing, and presentation for analytic and decision-making purposes. In comparison to traditional databases, data warehouses generally contain very large amounts of data from multiple sources that may include databases from different data models and sometimes files acquired from independent systems and platforms.

[1]Inmon (1992) is credited with initially using the term *warehouse*. The latest edition of his work is Inmon (2005).

## 29.2 Characteristics of Data Warehouses

To discuss data warehouses and distinguish them from transactional databases calls for an appropriate data model. The multidimensional data model (explained in more detail in Section 29.3) is a good fit for OLAP and decision-support technologies. In contrast to multidatabases, which provide access to disjoint and usually heterogeneous databases, a data warehouse is frequently a store of integrated data from multiple sources, processed for storage in a multidimensional model. Unlike most transactional databases, data warehouses typically support time-series and trend analysis, both of which require more historical data than is generally maintained in transactional databases.

Compared with transactional databases, data warehouses are nonvolatile. This means that information in the data warehouse changes far less often and may be regarded as non–real-time with periodic updating. In transactional systems, transactions are the unit and are the agent of change to the database; by contrast, data warehouse information is much more coarse-grained and is refreshed according to a careful choice of refresh policy, usually incremental. Warehouse updates are handled by the warehouse's acquisition component that provides all required preprocessing.

We can also describe data warehousing more generally as *a collection of decision support technologies, aimed at enabling the knowledge worker (executive, manager, analyst) to make better and faster decisions.*[2] Figure 29.1 gives an overview of the conceptual structure of a data warehouse. It shows the entire data warehousing process, which includes possible cleaning and reformatting of data before loading it into the warehouse. This process is handled by tools known as ETL (extraction, transformation, and loading) tools. At the back end of the process, OLAP, data mining, and DSS may generate new relevant information such as rules; this information is shown in the figure going back into the warehouse. The figure also shows that data sources may include files.

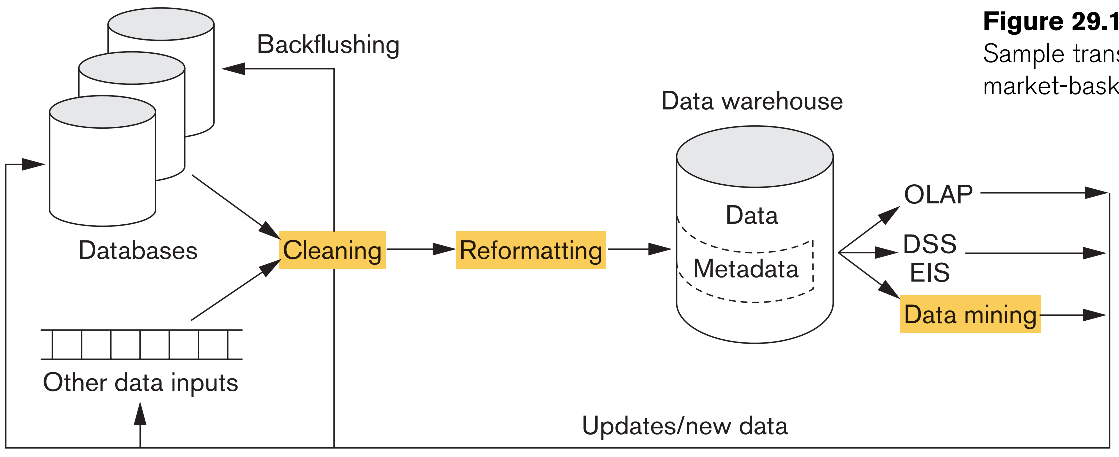


**Figure 29.1**
Sample transactions in market-basket model.

[2]Chaudhuri and Dayal (1997) provide an excellent tutorial on the topic, with this as a starting definition.

Data warehouses have the following distinctive characteristics:[3]

- Multidimensional conceptual view
- Generic dimensionality
- Unlimited dimensions and aggregation levels
- Unrestricted cross-dimensional operations
- Dynamic sparse matrix handling
- Client-server architecture
- Multiuser support
- Accessibility
- Transparency
- Intuitive data manipulation
- Consistent reporting performance
- Flexible reporting

Because they encompass large volumes of data, data warehouses are generally an order of magnitude (sometimes two orders of magnitude) larger than the source databases. The sheer volume of data (likely to be in terabytes or even petabytes) is an issue that has been dealt with through enterprise-wide data warehouses, virtual data warehouses, and data marts:

- **Enterprise-wide data warehouses** are huge projects requiring massive investment of time and resources.
- **Virtual data warehouses** provide views of operational databases that are materialized for efficient access.
- **Data marts** generally are targeted to a subset of the organization, such as a department, and are more tightly focused.

## 29.3 Data Modeling for Data Warehouses

Multidimensional models take advantage of inherent relationships in data to populate data in multidimensional matrices called *data cubes*. (These may be called *hypercubes* if they have more than three dimensions.) For data that lends itself to dimensional formatting, query performance in multidimensional matrices can be much better than in the relational data model. Three examples of dimensions in a corporate data warehouse are the corporation's fiscal periods, products, and regions.

A standard spreadsheet is a two-dimensional matrix. One example would be a spreadsheet of regional sales by product for a particular time period. Products could be shown as rows, with sales revenues for each region comprising the columns. (Figure 29.2 shows this two-dimensional organization.) Adding a time dimension,

[3]Codd and Salley (1993) coined the term OLAP and mentioned these characteristics. We have reordered their original list.

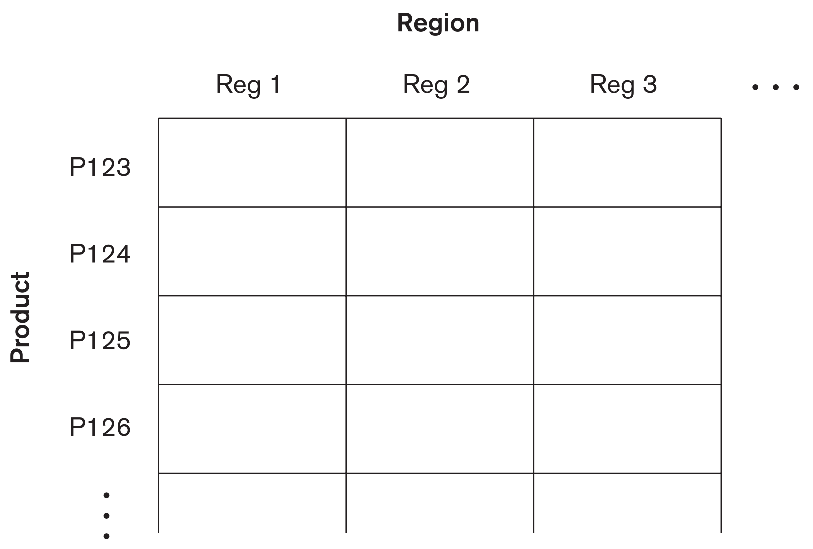


**Figure 29.2**
A two-dimensional matrix model.

such as an organization's fiscal quarters, would produce a three-dimensional matrix, which could be represented using a data cube.

Figure 29.3 shows a three-dimensional data cube that organizes product sales data by fiscal quarters and sales regions. Each cell could contain data for a specific product,

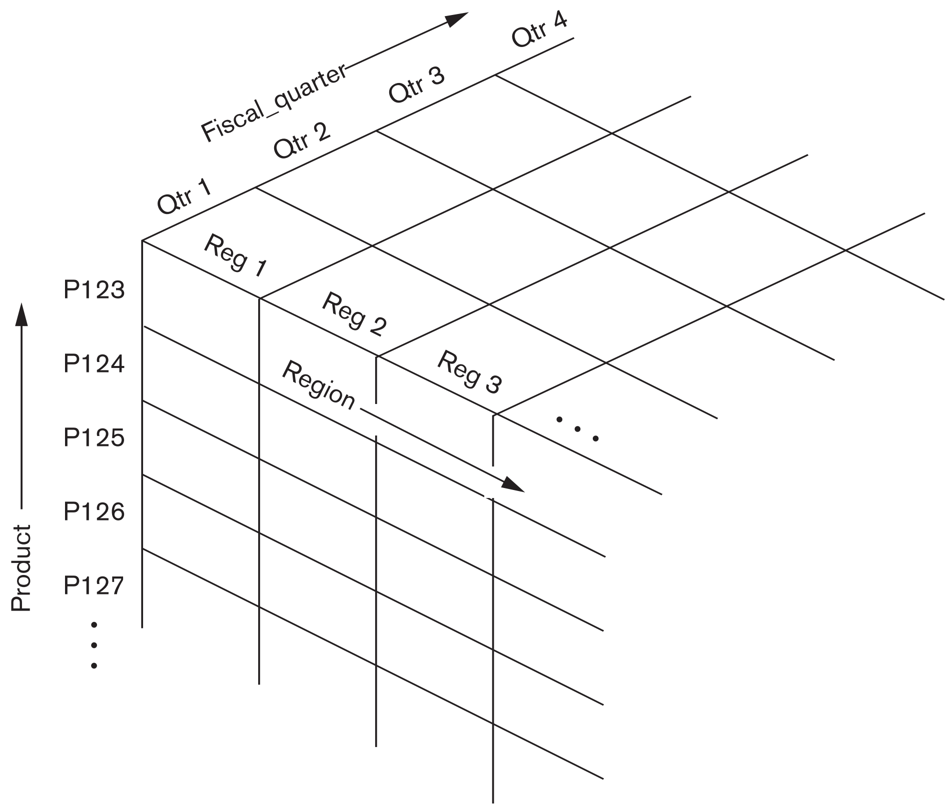


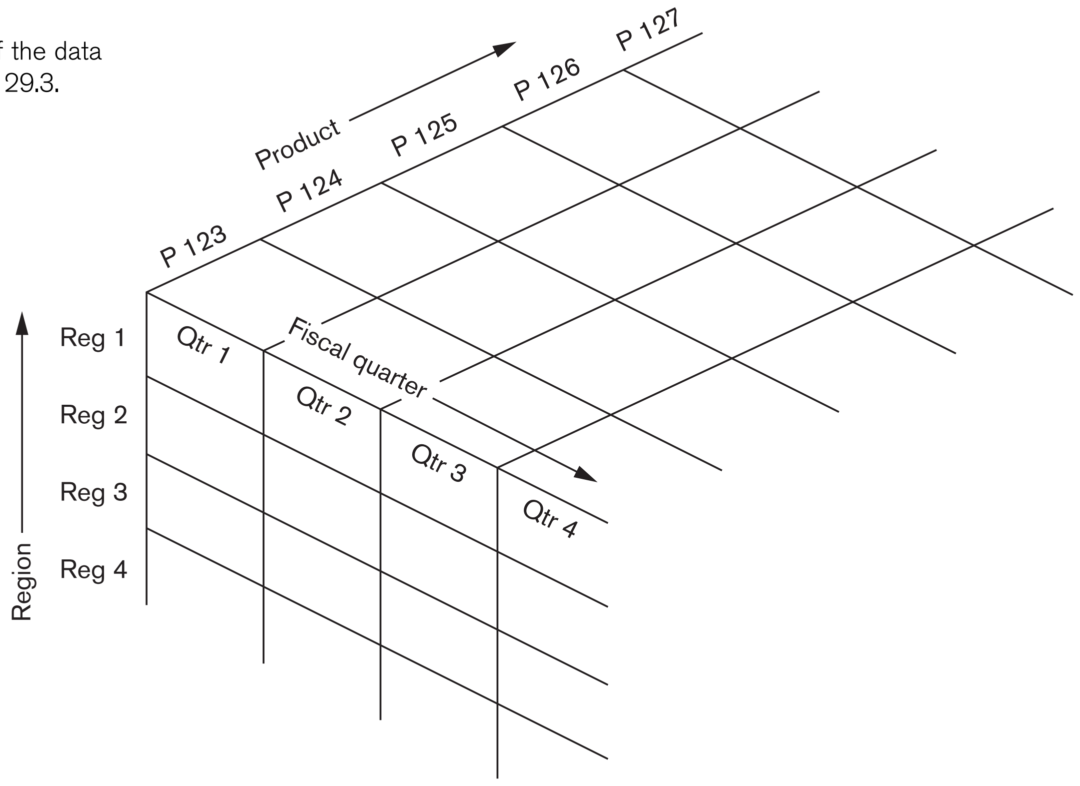
**Figure 29.3**
A three-dimensional data cube model.

specific fiscal quarter, and specific region. By including additional dimensions, a data hypercube could be produced, although more than three dimensions cannot be easily visualized or graphically presented. The data can be queried directly in any combination of dimensions, bypassing complex database queries. Tools exist for viewing data according to the user's choice of dimensions.

Changing from one-dimensional hierarchy (orientation) to another is easily accomplished in a data cube with a technique called **pivoting** (also called *rotation*). In this technique the data cube can be thought of as rotating to show a different orientation of the axes. For example, you might pivot the data cube to show regional sales revenues as rows, the fiscal quarter revenue totals as columns, and the company's products in the third dimension (Figure 29.4). Hence, this technique is equivalent to having a regional sales table for each product separately, where each table shows quarterly sales for that product region by region.

Multidimensional models lend themselves readily to hierarchical views in what is known as roll-up display and drill-down display. A **roll-up display** moves up the hierarchy, grouping into larger units along a dimension (for example, summing weekly data by quarter or by year). Figure 29.5 shows a roll-up display that moves from individual products to a coarser-grain of product categories. Shown in Figure 29.6, a **drill-down display** provides the opposite capability, furnishing a finer-grained view, perhaps disaggregating country sales by region and then regional sales by subregion and also breaking up products by styles.

**Figure 29.4**
Pivoted version of the data cube from Figure 29.3.



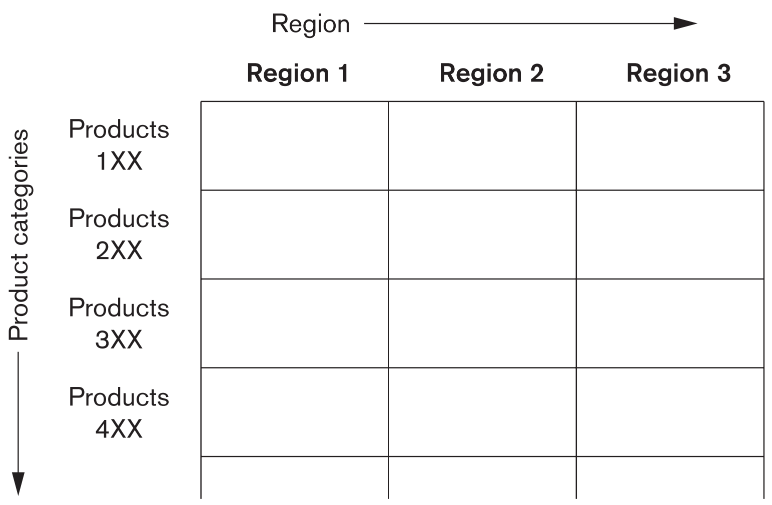


**Figure 29.5**
The roll-up operation.

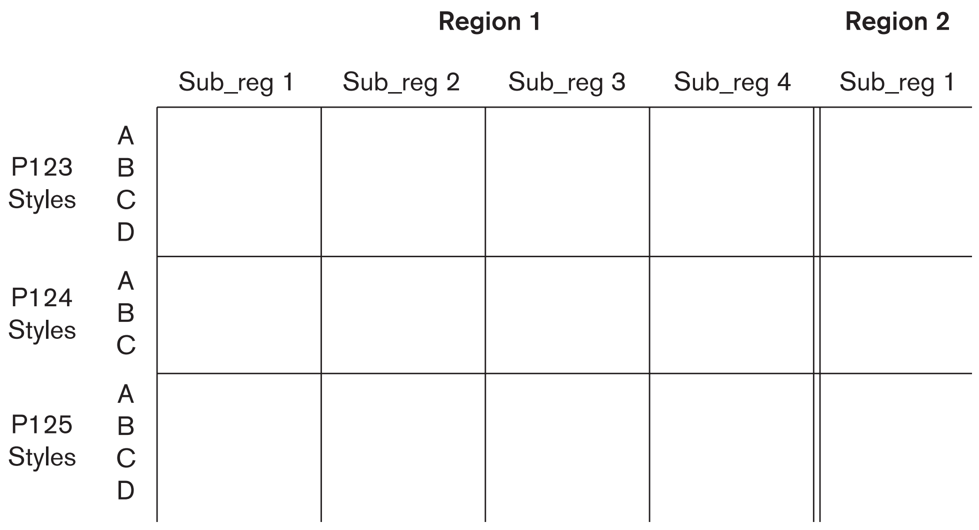


**Figure 29.6**
The drill-down operation.

The multidimensional storage model involves two types of tables: dimension tables and fact tables. A **dimension table** consists of tuples of attributes of the dimension. A **fact table** can be thought of as having tuples, one per a recorded fact. This fact contains some measured or observed variable(s) and identifies it (them) with pointers to dimension tables. The fact table contains the data, and the dimensions identify each tuple in that data. Figure 29.7 contains an example of a fact table that can be viewed from the perspective of multiple dimension tables.

Two common multidimensional schemas are the star schema and the snowflake schema. The **star schema** consists of a fact table with a single table for each dimension (Figure 29.7). The **snowflake schema** is a variation on the star schema in which

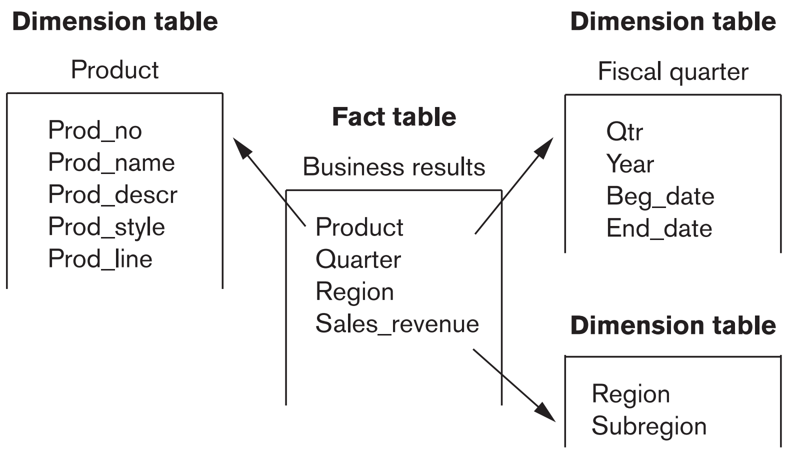
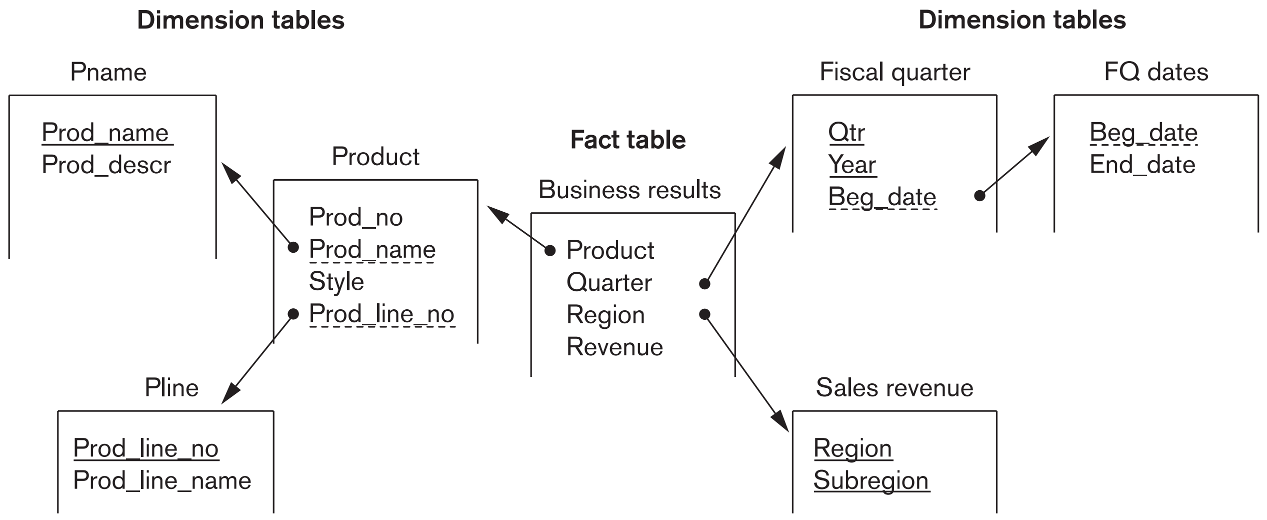


**Figure 29.7**
A star schema with fact and dimensional tables.

the dimensional tables from a star schema are organized into a hierarchy by normalizing them (Figure 29.8). Some installations are normalizing data warehouses up to the third normal form so that they can access the data warehouse to the finest level of detail. A **fact constellation** is a set of fact tables that share some dimension tables. Figure 29.9 shows a fact constellation with two fact tables, business results and business forecast. These share the dimension table called product. Fact constellations limit the possible queries for the warehouse.

Data warehouse storage also utilizes indexing techniques to support high-performance access (see Chapter 18 for a discussion of indexing). A technique called **bitmap indexing** constructs a bit vector for each value in a domain (column)

**Figure 29.8**
A snowflake schema.



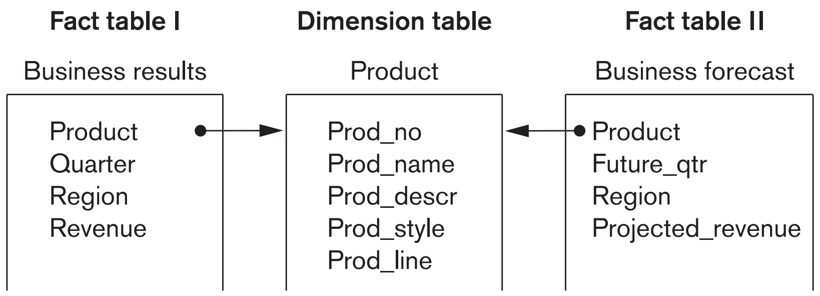


**Figure 29.9**
A fact constellation.

being indexed. It works very well for domains of low cardinality. There is a 1 bit placed in the *j*th position in the vector if the *j*th row contains the value being indexed. For example, imagine an inventory of 100,000 cars with a bitmap index on car size. If there are four car sizes—economy, compact, mid-size, and full-size—there will be four bit vectors, each containing 100,000 bits (12.5K) for a total index size of 50K. Bitmap indexing can provide considerable input/output and storage space advantages in low-cardinality domains. With bit vectors a bitmap index can provide dramatic improvements in comparison, aggregation, and join performance.

In a star schema, dimensional data can be indexed to tuples in the fact table by **join indexing**. Join indexes are traditional indexes to maintain relationships between primary key and foreign key values. They relate the values of a dimension of a star schema to rows in the fact table. For example, consider a sales fact table that has city and fiscal quarter as dimensions. If there is a join index on city, for each city the join index maintains the tuple IDs of tuples containing that city. Join indexes may involve multiple dimensions.

Data warehouse storage can facilitate access to summary data by taking further advantage of the nonvolatility of data warehouses and a degree of predictability of the analyses that will be performed using them. Two approaches have been used: (1) smaller tables including summary data such as quarterly sales or revenue by product line, and (2) encoding of level (for example, weekly, quarterly, annual) into existing tables. By comparison, the overhead of creating and maintaining such aggregations would likely be excessive in a volatile, transaction-oriented database.

## 29.4 Building a Data Warehouse

In constructing a data warehouse, builders should take a broad view of the anticipated use of the warehouse. There is no way to anticipate all possible queries or analyses during the design phase. However, the design should specifically support **ad-hoc querying**, that is, accessing data with any meaningful combination of values for the attributes in the dimension or fact tables. For example, a marketing-intensive consumer-products company would require different ways of organizing the data warehouse than would a nonprofit charity focused on fund raising. An appropriate schema should be chosen that reflects anticipated usage.

Acquisition of data for the warehouse involves the following steps:

1. The data must be extracted from multiple, heterogeneous sources, for example, databases or other data feeds such as those containing financial market data or environmental data.
2. Data must be formatted for consistency within the warehouse. Names, meanings, and domains of data from unrelated sources must be reconciled. For instance, subsidiary companies of a large corporation may have different fiscal calendars with quarters ending on different dates, making it difficult to aggregate financial data by quarter. Various credit cards may report their transactions differently, making it difficult to compute all credit sales. These format inconsistencies must be resolved.
3. The data must be cleaned to ensure validity. Data cleaning is an involved and complex process that has been identified as the largest labor-demanding component of data warehouse construction. For input data, cleaning must occur before the data is loaded into the warehouse. There is nothing about cleaning data that is specific to data warehousing and that could not be applied to a host database. However, since input data must be examined and formatted consistently, data warehouse builders should take this opportunity to check for validity and quality. Recognizing erroneous and incomplete data is difficult to automate, and cleaning that requires automatic error correction can be even tougher. Some aspects, such as domain checking, are easily coded into data cleaning routines, but automatic recognition of other data problems can be more challenging. (For example, one might require that City = 'San Francisco' together with State = 'CT' be recognized as an incorrect combination.) After such problems have been taken care of, similar data from different sources must be coordinated for loading into the warehouse. As data managers in the organization discover that their data is being cleaned for input into the warehouse, they will likely want to upgrade their data with the cleaned data. The process of returning cleaned data to the source is called **backflushing** (see Figure 29.1).
4. The data must be fitted into the data model of the warehouse. Data from the various sources must be installed in the data model of the warehouse. Data may have to be converted from relational, object-oriented, or legacy databases (network and/or hierarchical) to a multidimensional model.
5. The data must be loaded into the warehouse. The sheer volume of data in the warehouse makes loading the data a significant task. Monitoring tools for loads as well as methods to recover from incomplete or incorrect loads are required. With the huge volume of data in the warehouse, incremental updating is usually the only feasible approach. The refresh policy will probably emerge as a compromise that takes into account the answers to the following questions:
   - How up-to-date must the data be?
   - Can the warehouse go offline, and for how long?
   - What are the data interdependencies?

- What is the storage availability?
- What are the distribution requirements (such as for replication and partitioning)?
- What is the loading time (including cleaning, formatting, copying, transmitting, and overhead such as index rebuilding)?

As we have said, databases must strike a balance between efficiency in transaction processing and supporting query requirements (ad hoc user requests), but a data warehouse is typically optimized for access from a decision maker's needs. Data storage in a data warehouse reflects this specialization and involves the following processes:

- Storing the data according to the data model of the warehouse
- Creating and maintaining required data structures
- Creating and maintaining appropriate access paths
- Providing for time-variant data as new data are added
- Supporting the updating of warehouse data
- Refreshing the data
- Purging data

Although adequate time can be devoted initially to constructing the warehouse, the sheer volume of data in the warehouse generally makes it impossible to simply reload the warehouse in its entirety later on. Alternatives include selective (partial) refreshing of data and separate warehouse versions (requiring double storage capacity for the warehouse!). When the warehouse uses an incremental data refreshing mechanism, data may need to be periodically purged; for example, a warehouse that maintains data on the previous twelve business quarters may periodically purge its data each year.

Data warehouses must also be designed with full consideration of the environment in which they will reside. Important design considerations include the following:

- Usage projections
- The fit of the data model
- Characteristics of available sources
- Design of the metadata component
- Modular component design
- Design for manageability and change
- Considerations of distributed and parallel architecture

We discuss each of these in turn. Warehouse design is initially driven by usage projections; that is, by expectations about who will use the warehouse and how they will use it. Choice of a data model to support this usage is a key initial decision. Usage projections and the characteristics of the warehouse's data sources are both taken into account. Modular design is a practical necessity to allow the warehouse to evolve with the organization and its information environment. Additionally, a well-built data warehouse must be designed for maintainability, enabling the warehouse managers to plan for and manage change effectively while providing optimal support to users.

You may recall the term *metadata* from Chapter 1; metadata was defined as the description of a database including its schema definition. The **metadata repository** is a key data warehouse component. The metadata repository includes both technical and business metadata. The first, **technical metadata**, covers details of acquisition processing, storage structures, data descriptions, warehouse operations and maintenance, and access support functionality. The second, **business metadata**, includes the relevant business rules and organizational details supporting the warehouse.

The architecture of the organization's distributed computing environment is a major determining characteristic for the design of the warehouse.

There are two basic distributed architectures: the distributed warehouse and the federated warehouse. For a **distributed warehouse**, all the issues of distributed databases are relevant, for example, replication, partitioning, communications, and consistency concerns. A distributed architecture can provide benefits particularly important to warehouse performance, such as improved load balancing, scalability of performance, and higher availability. A single replicated metadata repository would reside at each distribution site. The idea of the **federated warehouse** is like that of the federated database: a decentralized confederation of autonomous data warehouses, each with its own metadata repository. Given the magnitude of the challenge inherent to data warehouses, it is likely that such federations will consist of smaller scale components, such as data marts. Large organizations may choose to federate data marts rather than build huge data warehouses.

## 29.5 Typical Functionality of a Data Warehouse

Data warehouses exist to facilitate complex, data-intensive, and frequent ad hoc queries. Accordingly, data warehouses must provide far greater and more efficient query support than is demanded of transactional databases. The data warehouse access component supports enhanced spreadsheet functionality, efficient query processing, structured queries, ad hoc queries, data mining, and materialized views. In particular, enhanced spreadsheet functionality includes support for state-of-the-art spreadsheet applications (for example, MS Excel) as well as for OLAP applications programs. These offer preprogrammed functionalities such as the following:

- **Roll-up.** Data is summarized with increasing generalization (for example, weekly to quarterly to annually).
- **Drill-down.** Increasing levels of detail are revealed (the complement of roll-up).
- **Pivot.** Cross tabulation (also referred to as *rotation*) is performed.
- **Slice and dice.** Projection operations are performed on the dimensions.
- **Sorting.** Data is sorted by ordinal value.

- **Selection.** Data is available by value or range.
- **Derived (computed) attributes.** Attributes are computed by operations on stored and derived values.

Because data warehouses are free from the restrictions of the transactional environment, there is an increased efficiency in query processing. Among the tools and techniques used are query transformation; index intersection and union; special **ROLAP** (relational OLAP) and **MOLAP** (multidimensional OLAP) functions; SQL extensions; advanced join methods; and intelligent scanning (as in piggy-backing multiple queries).

Improved performance has also been attained with parallel processing. Parallel server architectures include symmetric multiprocessor (SMP), cluster, and massively parallel processing (MPP), and combinations of these.

Knowledge workers and decision makers use tools ranging from parametric queries to ad hoc queries to data mining. Thus, the access component of the data warehouse must provide support for structured queries (both parametric and ad hoc). Together, these make up a managed query environment. Data mining itself uses techniques from statistical analysis and artificial intelligence. Statistical analysis can be performed by advanced spreadsheets, by sophisticated statistical analysis software, or by custom-written programs. Techniques such as lagging, moving averages, and regression analysis are also commonly employed. Artificial intelligence techniques, which may include genetic algorithms and neural networks, are used for classification and are employed to discover knowledge from the data warehouse that may be unexpected or difficult to specify in queries. (We treat data mining in detail in Chapter 28.)

## 29.6 Data Warehouse versus Views

Some people have considered data warehouses to be an extension of database views. Earlier we mentioned materialized views as one way of meeting requirements for improved access to data (see Section 5.3 for a discussion of views). Materialized views have been explored for their performance enhancement. Views, however, provide only a subset of the functions and capabilities of data warehouses. Views and data warehouses are alike in that they both have read-only extracts from databases and subject orientation. However, data warehouses are different from views in the following ways:

- Data warehouses exist as persistent storage instead of being materialized on demand.
- Data warehouses are not usually relational, but rather multidimensional. Views of a relational database are relational.
- Data warehouses can be indexed to optimize performance. Views cannot be indexed independent of the underlying databases.
- Data warehouses characteristically provide specific support of functionality; views cannot.

- Data warehouses provide large amounts of integrated and often temporal data, generally more than is contained in one database, whereas views are an extract of a database.

## 29.7 Difficulties of Implementing Data Warehouses

Some significant operational issues arise with data warehousing: construction, administration, and quality control. Project management—the design, construction, and implementation of the warehouse—is an important and challenging consideration that should not be underestimated. The building of an enterprise-wide warehouse in a large organization is a major undertaking, potentially taking years from conceptualization to implementation. Because of the difficulty and amount of lead time required for such an undertaking, the widespread development and deployment of data marts may provide an attractive alternative, especially to those organizations with urgent needs for OLAP, DSS, and/or data mining support.

The administration of a data warehouse is an intensive enterprise, proportional to the size and complexity of the warehouse. An organization that attempts to administer a data warehouse must realistically understand the complex nature of its administration. Although designed for read access, a data warehouse is no more a static structure than any of its information sources. Source databases can be expected to evolve. The warehouse's schema and acquisition component must be expected to be updated to handle these evolutions.

A significant issue in data warehousing is the quality control of data. Both quality and consistency of data are major concerns. Although the data passes through a cleaning function during acquisition, quality and consistency remain significant issues for the database administrator. Melding data from heterogeneous and disparate sources is a major challenge given differences in naming, domain definitions, identification numbers, and the like. Every time a source database changes, the data warehouse administrator must consider the possible interactions with other elements of the warehouse.

Usage projections should be estimated conservatively prior to construction of the data warehouse and should be revised continually to reflect current requirements. As utilization patterns become clear and change over time, storage and access paths can be tuned to remain optimized for support of the organization's use of its warehouse. This activity should continue throughout the life of the warehouse in order to remain ahead of demand. The warehouse should also be designed to accommodate the addition and attrition of data sources without major redesign. Sources and source data will evolve, and the warehouse must accommodate such change. Fitting the available source data into the data model of the warehouse will be a continual challenge, a task that is as much art as science. Because there is continual rapid change in technologies, both the requirements and capabilities of the warehouse will change considerably over time. Additionally, data warehousing technology itself will continue to evolve for some time so that component structures and functionalities will continually be upgraded. This certain change is excellent motivation for having fully modular design of components.

Administration of a data warehouse will require far broader skills than are needed for traditional database administration. A team of highly skilled technical experts with overlapping areas of expertise will likely be needed, rather than a single individual. Like database administration, data warehouse administration is only partly technical; a large part of the responsibility requires working effectively with all the members of the organization with an interest in the data warehouse. However difficult that can be at times for database administrators, it is that much more challenging for data warehouse administrators, as the scope of their responsibilities is considerably broader.

Design of the management function and selection of the management team for a database warehouse are crucial. Managing the data warehouse in a large organization will surely be a major task. Many commercial tools are available to support management functions. Effective data warehouse management will certainly be a team function, requiring a wide set of technical skills, careful coordination, and effective leadership. Just as we must prepare for the evolution of the warehouse, we must also recognize that the skills of the management team will, of necessity, evolve with it.

## 29.8 Summary

In this chapter we surveyed the field known as data warehousing. Data warehousing can be seen as a process that requires a variety of activities to precede it. In contrast, data mining (see Chapter 28) may be thought of as an activity that draws knowledge from an existing data warehouse. We introduced key concepts related to data warehousing and we discussed the special functionality associated with a multidimensional view of data. We also discussed the ways in which data warehouses supply decision makers with information at the correct level of detail, based on an appropriate organization and perspective.

## Review Questions

**29.1.** What is a data warehouse? How does it differ from a database?

**29.2.** Define the terms: OLAP (online analytical processing), ROLAP (relational OLAP), MOLAP (multidimensional OLAP), and DSS (decision-support systems).

**29.3.** Describe the characteristics of a data warehouse. Divide them into functionality of a warehouse and advantages users derive from it.

**29.4.** What is the multidimensional data model? How is it used in data warehousing?

**29.5.** Define the following terms: star schema, snowflake schema, fact constellation, data marts.

**29.6.** What types of indexes are built for a warehouse? Illustrate the uses for each with an example.

**29.7.** Describe the steps of building a warehouse.

**29.8.** What considerations play a major role in the design of a warehouse?

**29.9.** Describe the functions a user can perform on a data warehouse and illustrate the results of these functions on a sample multidimensional data warehouse.

**29.10.** How is the concept of a relational view related to a data warehouse and data marts? In what way are they different?

**29.11.** List the difficulties in implementing a data warehouse.

**29.12.** List the open issues and research problems in data warehousing.

## Selected Bibliography

Inmon (1992, 2005) is credited for giving the term wide acceptance. Codd and Salley (1993) popularized the term online analytical processing (OLAP) and defined a set of characteristics for data warehouses to support OLAP. Kimball (1996) is known for his contribution to the development of the data warehousing field. Mattison (1996) is one of the several books on data warehousing that gives a comprehensive analysis of techniques available in data warehouses and the strategies companies should use in deploying them. Ponniah (2002) gives a very good practical overview of the data warehouse building process from requirements collection to deployment maintenance. Bischoff and Alexander (1997) is a compilation of advice from experts. Chaudhuri and Dayal (1997) give an excellent tutorial on the topic, while Widom (1995) points to a number of outstanding research problems.