# Business Intelligence and Data Warehouses

13

THIRTEEN

**In this chapter, you will learn:**

- How business intelligence is a comprehensive framework to support business decision making
- How operational data and decision support data differ
- What a data warehouse is, how to prepare data for one, and how to implement one
- What star schemas are and how they are constructed
- What data mining is and what role it plays in decision support
- About online analytical processing (OLAP)
- How SQL extensions are used to support OLAP-type data manipulations

## Preview

Data are crucial raw material in this information age, and data storage and management have become the focus of database design and implementation. Ultimately, the reason for collecting, storing, and managing data is to generate information that becomes the basis for rational decision making. Decision support systems (DSSs) were originally developed to facilitate the decision-making process. However, as the complexity and range of information requirements increased, so did the difficulty of extracting all the necessary information from the data structures typically found in an operational database. Therefore, a new data storage facility, called a data warehouse, was developed. The data warehouse extracts or obtains its data from operational databases as well as from external sources, providing a more comprehensive data pool.

In parallel with data warehouses, new ways to analyze and present decision support data were developed. Online analytical processing (OLAP) provides advanced data analysis and presentation tools (including multidimensional data analysis). Data mining employs advanced statistical tools to analyze the wealth of data now available through data warehouses and other sources and to identify possible relationships and anomalies.

Business intelligence (BI) is the collection of best practices and software tools developed to support business decision making in this age of globalization, emerging markets, rapid change, and increasing regulation. BI encompasses tools and techniques such as data warehouses and OLAP, with a more comprehensive focus on integrating them from a company-wide perspective.

This chapter explores the main concepts and components of business intelligence and decision support systems that gather,generate, and present information for business decision makers, focusing especially on the use of data warehouses.

## 13.1 THE NEED FOR DATA ANALYSIS

Organizations tend to grow and prosper as they gain a better understanding of their environment. Most managers want to be able to track daily transactions to evaluate how the business is performing. By tapping into the operational database, management can develop strategies to meet organizational goals. In addition, data analysis can provide information about short-term tactical evaluations and strategies such as these: Are our sales promotions working? What market percentage are we controlling? Are we attracting new customers? Tactical and strategic decisions are also shaped by constant pressure from external and internal forces, including globalization, the cultural and legal environment, and (perhaps most importantly) technology.

Given the many and varied competitive pressures, managers are always looking for a competitive advantage through product development and maintenance, service, market positioning, sales promotion, and so on. Managers understand that the business climate is dynamic, and thus, mandates their prompt reaction to change in order to remain competitive. In addition, the modern business climate requires managers to approach increasingly complex problems that involve a rapidly growing number of internal and external variables. It should also come as no surprise that interest is growing in creating support systems dedicated to facilitating quick decision making in a complex environment.

Different managerial levels require different decision support needs. For example, transaction-processing systems, based on operational databases, are tailored to serve the information needs of people who deal with short-term inventory, accounts payable, and purchasing. Middle-level managers, general managers, vice presidents, and presidents focus on strategic and tactical decision making. Those managers require detailed information designed to help them make decisions in a complex data and analysis environment.

Companies and software vendors addressed these multilevel decision support needs by creating independent applications to fit the needs of particular areas (finance, customer management, human resources, product support, etc.). Applications were also tailored to different industry sectors such as education, retail, health care, or financial. This approach worked well for some time, but changes in the business world (globalization, expanding markets, mergers and acquisitions, increased regulation, and more) called for new ways of integrating and managing data across levels and sectors. This more comprehensive and integrated decision support framework within organizations became known as business intelligence.

## 13.2 BUSINESS INTELLIGENCE

**Business intelligence (BI)**[1] is a term used to describe a comprehensive, cohesive, and integrated set of tools and processes used to capture, collect, integrate, store, and analyze data with the purpose of generating and presenting information used to support business decision making. As the names implies, BI is about creating intelligence about a business. This intelligence is based on learning and understanding the facts about a business environment. BI is a framework that allows a business to transform data into information, information into knowledge, and knowledge into wisdom. BI has the potential to positively affect a company's culture by creating "business wisdom" and distributing it to all users in an organization. This business wisdom empowers users to make sound business decisions based on the accumulated

[1] In 1989, while working at Gartner Inc., Howard Dresner popularized "BI" as an umbrella term to describe a set of concepts and methods to improve business decision making by using fact-based support systems. Source: http://www.computerworld.com/action/article.do?command=viewArticleBasic&articleId=266298

knowledge of the business as reflected on recorded facts (historic operational data). Table 13.1 gives some real-world examples of companies that have implemented BI tools (data warehouse, data mart, OLAP, and/or data mining tools) and shows how the use of such tools benefited the companies.

**TABLE 13.1 Solving Business Problems and Adding Value with BI Tools**

| COMPANY | PROBLEM | BENEFIT |
|---|---|---|
| MOEN<br>Manufacturer of bathroom and kitchen fixtures and supplies<br>Source: Cognos Corp.<br>*www.cognos.com* | • Information generation very limited and time-consuming.<br>• How to extract data using a 3GL known by only five people.<br>• Response time unacceptable for managers' decision-making purposes. | • Provided quick answers to ad hoc questions for decision making.<br>• Provided access to data for decision-making purposes.<br>• Received in-depth view of product performance and customer margins. |
| NASDAQ<br>Largest U.S. electronic stock market trading organization<br>Source: Oracle<br>*www.oracle.com* | • Inability to provide real-time ad hoc query and standard reporting to executives, business analysts, and other users.<br>• Excessive storage costs for many terabytes of data. | • Reduced storage cost by moving to a multitier storage solution.<br>• Implemented new data warehouse center with support for ad hoc query and reporting and near real-time data access for end users. |
| Sega of America, Inc.<br>Interactive entertainment systems and video games<br>Source: Oracle Corp.<br>*www.oracle.com* | • Needed a way to rapidly analyze a great amount of data.<br>• Needed to track advertising, coupons, and rebates associated with effects of pricing changes.<br>• Used to do it with Excel spreadsheets, leading to human-caused errors. | • Eliminated data-entry errors.<br>• Identified successful marketing strategies to dominate interactive entertainment niches.<br>• Used product analysis to identify better markets/product offerings. |
| Owens and Minor, Inc.<br>Medical and surgical supply distributor<br>Source: *CFO Magazine*<br>*www.cfomagazine.com* | • Lost its largest customer, which represented 10% of its annual revenue ($360 million).<br>• Stock plunged 23%.<br>• Cumbersome process to get information out of antiquated mainframe system. | • Increased earnings per share in just five months.<br>• Gained more business, thanks to opening the data warehouse to its clients.<br>• Managers gained quick access to data for decision-making purposes. |
| Amazon.com<br>Leading online retailer<br>Source: *PC Week Online*<br>*whitepapers.zdnet.com/whitepaper.aspx?docid=241748* | • Difficulty in managing a very rapidly growing data environment.<br>• Existing data warehouse solution not capable of supporting extremely rapid growth.<br>• Needed more flexible and reliable data warehouse solution to protect its investment in data and infrastructure. | • Implemented new data warehouse with superior scalability and performance.<br>• Improved business intelligence.<br>• Improved management of product flow through the entire supply chain.<br>• Improved customer experience. |

BI is a comprehensive endeavor because it encompasses all business processes within an organization. *Business processes* are the central units of operation in a business. Implementing BI in an organization involves capturing not only business data (internal and external) but also the metadata, or knowledge about the data. In practice, BI is a complex proposition that requires a deep understanding and alignment of the business processes, the internal and external data, and the information needs of users at all levels in an organization.

BI is not a product by itself, but a framework of concepts, practices, tools, and technologies that help a business better understand its core capabilities, provide snapshots of the company situation, and identify key opportunities to create competitive advantage. In practice, BI provides a well-orchestrated framework for the management of data that works across all levels of the organization. BI involves the following general steps:

1. Collecting and storing operational data
2. Aggregating the operational data into decision support data
3. Analyzing decision support data to generate information
4. Presenting such information to the end user to support business decisions
5. Making business decisions, which in turn generate more data that is collected, stored, etc. (restarting the process)
6. Monitoring results to evaluate outcomes of the business decisions (providing more data to be collected, stored, etc.)

To implement all these steps, BI uses varied components and technologies. In the following sections, you will learn about the basic BI architecture and implementations.

## 13.3 BUSINESS INTELLIGENCE ARCHITECTURE

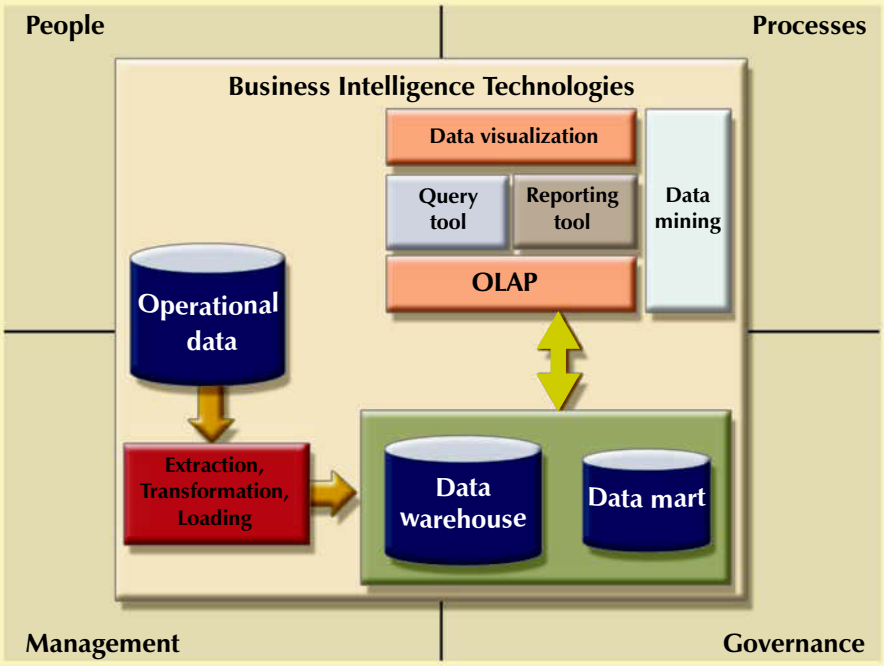
BI covers a range of technologies and applications to manage the entire data life cycle from acquisition to storage, transformation, integration, analysis, monitoring, presentation, and archiving. BI functionality ranges from simple data gathering and extraction to very complex data analysis and presentation applications. There is no single BI architecture; instead, it ranges from highly integrated applications from a single vendor to a loosely integrated, multivendor environment. However, there are some general types of functionality that all BI implementations share.

Like any critical business IT infrastructure, the BI architecture is composed of data, people, processes, technology, and the management of such components. Figure 13.1 depicts how all those components fit together within the BI framework.

Remember that the main focus of BI is to gather, integrate, and store business data for the purpose of creating information. As depicted in Figure 13.1, BI integrates people and processes using technology in order to add value to the business. Such value is derived from how end users use such information in their daily activities, and in particular, their daily business decision making. Also note that the BI technology components are varied. This chapter will explain those components in greater detail in the following sections.

The focus of traditional information systems was on operational automation and reporting; in contrast, BI tools focus on the strategic and tactical use of information. In order to achieve this goal, BI recognizes that technology alone is not enough. Therefore, BI uses an arrangement of best management practices to manage data as a corporate asset. One of the most recent developments in this area is the use of master data management techniques. **Master data management (MDM)** is a collection of concepts, techniques, and processes for the proper identification, definition, and management of data elements within an organization. MDM's main goal is to provide a comprehensive and consistent definition of all data within an organization. MDM ensures that all company resources (people, procedures, and IT systems) that operate over data have uniform and consistent views of the company's data.

**FIGURE 13.1** Business intelligence framework




An added benefit of this meticulous approach to data management and decision making is that it provides a framework for business governance. **Governance** is a method or process of government. In this case, BI provides a method for controlling and monitoring business health and for consistent decision making. Furthermore, having such governance creates accountability for business decisions. In the present age of business flux, accountability is increasingly important. Had governance been as pivotal to business operations a few years back, crises precipitated by the likes of Enron, WorldCom, and Arthur Andersen might have been avoided.

Monitoring a business's health is crucial to understanding where the company is and where it is headed. In order to do this, BI makes extensive use of a special type of metrics known as key performance indicators. **Key performance indicators (KPI)** are quantifiable measurements (numeric or scale based) that assess the company's effectiveness or success in reaching its strategic and operational goals. There are many different KPI used by different industries. Some examples of KPI are:

- *General*. Year-to-year measurements of profit by line of business, same store sales, product turnovers, product recalls, sales by promotion, sales by employee, etc.
- *Finance*. Earnings per share, profit margin, revenue per employee, percentage of sales to account receivables, assets to sales, etc.
- *Human resources*. Applicants to job openings, employee turnover, employee longevity, etc.
- *Education*. Graduation rates, number of incoming freshmen, student retention rates, etc.

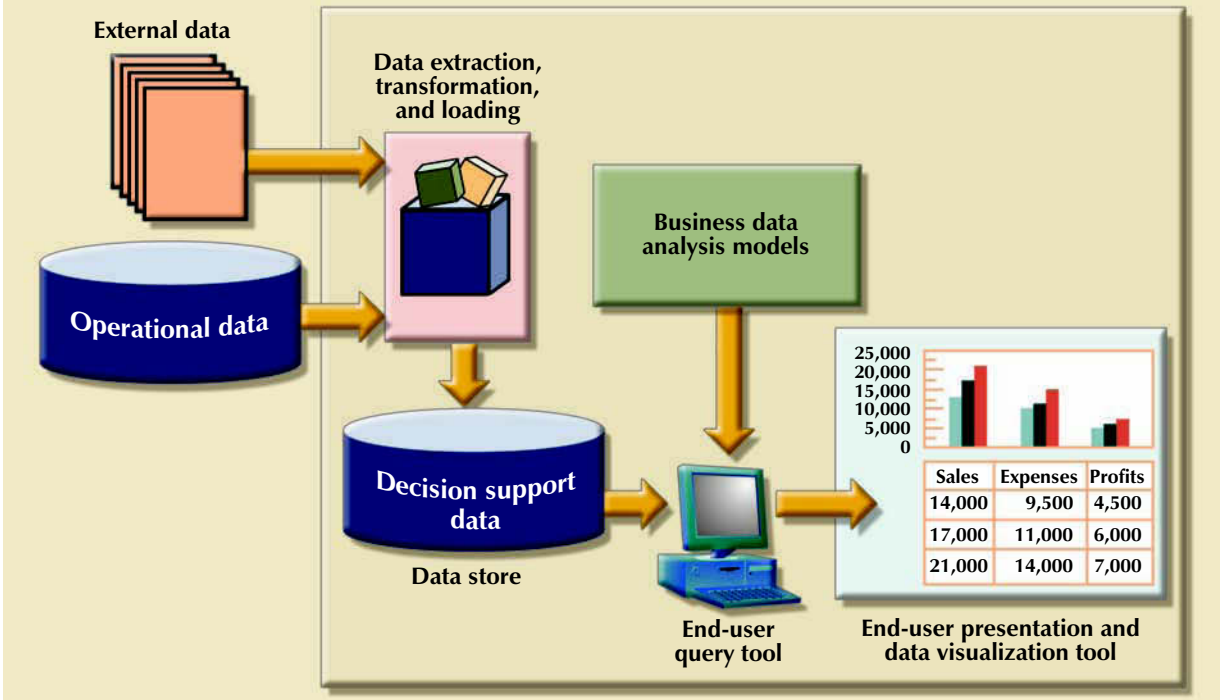
KPIs are determined after the main strategic, tactical, and operational goals for a business are defined. To tie the KPI to the strategic master plan of an organization, a KPI will be compared to a desired goal within a specific time frame. For example, if you are in an academic environment, you might be interested in ways to measure student satisfaction or retention. In this case, a sample goal would be to "Increase the graduating senior average exit exam grades from 9 to 12 by fall, 2010." Another sample KPI would be: "Increase the returning student rate of freshman year to sophomore year from 60% to 75% by 2012." In this case, such performance indicators would be measured and monitored on a year-to-year basis, and plans to achieve such goals would be set in place.

Another way to understand BI architecture is by describing the basic components that form part of its infrastructure. Some of the components have overlapping functionality; however, there are four basic components that all BI environments should provide. These are described in Table 13.2 and illustrated in Figure 13.2.

**TABLE 13.2 Basic BI Architectural Components**

| COMPONENT | DESCRIPTION |
|---|---|
| Data extraction, transformation, and loading (ETL) tools | This component is in charge of collecting, filtering, integrating, and aggregating operational data to be saved into a data store optimized for decision support. For example, to determine the relative market share by selected product lines, you require data from competitors' products. Such data can be located in external databases provided by industry groups or by companies that market the data. As the name implies, this component extracts the data, filters the extracted data to select the relevant records, and packages the data in the right format to be added to the data store component. |
| Data store | The data store is optimized for decision support and is generally represented by a data warehouse or a data mart. The data store contains business data extracted from the operational database and from external data sources. The business data are stored in structures that are optimized for data analysis and query speed. The external data sources provide data that cannot be found within the company but that are relevant to the business, such as stock prices, market indicators, marketing information (such as demographics), and competitors' data. |
| Data query and analysis tools | This component performs data retrieval, data analysis, and data mining tasks using the data in the data store and business data analysis models. This component is used by the data analyst to create the queries that access the database. Depending on the implementation, the query tool accesses either the operational database, or more commonly, the data store. This tool advises the user on which data to select and how to build a reliable business data model. This component is generally represented in the form of an OLAP tool. |
| Data presentation and visualization tools | This component is in charge of presenting the data to the end user in a variety of ways. This component is used by the data analyst to organize and present the data. This tool helps the end user select the most appropriate presentation format, such as summary report, map, pie or bar graph, or mixed graphs. The query tool and the presentation tool are the front end to the BI environment. |

**FIGURE 13.2** Business intelligence components




Each BI component shown in Table 13.2 has generated a fast-growing market for specialized tools. And thanks to the advancement of client/server technologies, those components can interact with other components to form a truly open architecture. As a matter of fact, you can integrate multiple tools from different vendors into a single BI framework. Table 13.3 shows a sample of common BI tools and vendors.

**TABLE 13.3** Sample of Business Intelligence Tools

| TOOL | DESCRIPTION | SAMPLE VENDORS |
|---|---|---|
| Decision support systems | A **decision support system** (**DSS**) is an arrangement of computerized tools used to assist managerial decision making within a business. *Decision support* systems were the precursors of modern BI systems. A DSS typically has a much narrower focus and reach than a BI solution. | SAP<br>Teradata<br>IBM<br>Proclarity |
| Dashboards and business activity monitoring | **Dashboards** use Web-based technologies to present key business performance indicators or information in a single integrated view, generally using graphics in a clear, concise, and easy to understand manner. | *Salesforce*<br>VisualCalc<br>Cognos<br>BusinessObjects<br>Information Builders<br>Actuate |
| Portals | **Portals** provide a unified, single point of entry for information distribution. Portals are a Web-based technology that uses a Web browser to integrate data from multiple sources into a single Web page. Many different types of BI functionality can be accessed through a portal. | Oracle Portal<br>Actuate<br>Microsoft |

**TABLE 13.3 Sample of Business Intelligence Tools (continued)**

| TOOL | DESCRIPTION | SAMPLE VENDORS |
|---|---|---|
| Data analysis and reporting tools | Advanced tools used to query multiple diverse data sources to create single integrated reports. | Mircrosoft Reporting Services<br>Information Builders<br>Eclipse BIRT<br>MicroStrategy<br>SAS WebReportStudio |
| Data mining tools | Tools that provide advanced statistical analysis to uncover problems and opportunities hidden within business data. | MicroStrategy Intelligence Server<br>MS Analytics Services |
| Data warehouses | The data warehouse is the foundation on which a BI infrastructure is built. Data is captured from the OLTP system and placed on the DW on near-real time basis. BI provides companywide integration of data and the capability to respond to business issues in a timely manner. | Microsoft<br>Oracle<br>IBM<br>MicroStrategy |
| OLAP tools | Online analytical processing provides multidimensional data analysis. | Cognos<br>BusinessObjects<br>Oracle<br>Microsoft |
| Data visualization | Tools that provide advanced visual analysis and techniques to enhance understanding of business data. | Advanced Visual Systems<br>Dundas<br>iDashboards |

Although BI has an unquestionably important role in modern business operations, keep in mind that the *manager* must initiate the decision support process by asking the appropriate questions. The BI environment exists to support the manager; it does *not* replace the management function. If the manager fails to ask the appropriate questions, problems will not be identified and solved, and opportunities will be missed. In spite of the very powerful BI presence, the human component is still at the center of business technology.

**NOTE**

Although the term BI includes a variety of components and tools, this chapter focuses on its data warehouse component.

## 13.4 DECISION SUPPORT DATA

Although BI is used at strategic and tactical managerial levels within organizations, *its effectiveness depends on the quality of data gathered at the operational level*. Yet operational data are seldom well suited to the decision support tasks. The differences between operational data and decision support data are examined in the next section.

### 13.4.1 Operational Data vs. Decision Support Data

Operational data and decision support data serve different purposes. Therefore, it is not surprising to learn that their formats and structures differ.

Most operational data are stored in a relational database in which the structures (tables) tend to be highly normalized. Operational data storage is optimized to support transactions that represent daily operations. For example, each time an item is sold, it must be accounted for. Customer data, inventory data, and so on, are in a frequent update mode. To provide effective update performance, operational systems store data in many tables, each with a minimum number of fields. Thus, a simple sales transaction might be represented by five or more different tables (for example, invoice,

invoice line, discount, store, and department). Although such an arrangement is excellent in an operational database, it is not efficient for query processing. For example, to extract a simple invoice, you would have to join several tables. Whereas operational data are useful for capturing daily business transactions, decision support data give tactical and strategic business meaning to the operational data. From the data analyst's point of view, decision support data differ from operational data in three main areas: time span, granularity, and dimensionality.

- *Time span.* Operational data cover a short time frame. In contrast, decision support data tend to cover a longer time frame. Managers are seldom interested in a specific sales invoice to customer X; rather, they tend to focus on sales generated during the last month, the last year, or the last five years.
- *Granularity (level of aggregation).* Decision support data must be presented at different levels of aggregation, from highly summarized to near-atomic. For example, if managers must analyze sales by region, they must be able to access data showing the sales by region, by city within the region, by store within the city within the region, and so on. In that case, summarized data to compare the regions is required, but also data in a structure that enables a manager to **drill down**, or decompose, the data into more atomic components (that is, finer-grained data at lower levels of aggregation). In contrast, when you **roll up** the data, you are aggregating the data to a higher level.
- *Dimensionality.* Operational data focus on representing individual transactions rather than on the effects of the transactions over time. In contrast, data analysts tend to include many data dimensions and are interested in how the data relate over those dimensions. For example, an analyst might want to know how product X fared relative to product Z during the past six months by region, state, city, store, and customer. In that case, both place and time are part of the picture.

Figure 13.3 shows how decision support data can be examined from multiple dimensions (such as product, region, and year), using a variety of filters to produce each dimension. The ability to analyze, extract, and present information in meaningful ways is one of the differences between decision support data and transaction-at-a-time operational data.

From the designer's point of view, the differences between operational and decision support data are as follows:

- Operational data represent transactions as they happen in real time. Decision support data are a snapshot of the operational data at a given point in time. Therefore, decision support data are historic, representing a time slice of the operational data.
- Operational and decision support data are different in terms of transaction *type* and transaction *volume*. Whereas operational data are characterized by update transactions, decision support data are mainly characterized by *query* (read-only) transactions. Decision support data also require *periodic* updates to load new data that are summarized from the operational data. Finally, the concurrent transaction volume in operational data tends to be very high when compared with the low-to-medium levels found in decision support data.
- Operational data are commonly stored in many tables, and the stored data represent the information about a given transaction only. Decision support data are generally stored in a few tables that store data derived from the operational data. The decision support data do not include the details of each operational transaction. Instead, decision support data represent transaction *summaries*; therefore, the decision support database stores data that are integrated, aggregated, and summarized for decision support purposes.
- The degree to which decision support data are summarized is very high when contrasted with operational data. Therefore, you will see a great deal of derived data in decision support databases. For example, rather than storing all 10,000 sales transactions for a given store on a given day, the decision support database might simply store the total number of units sold and the total sales dollars generated during that day. Decision support data might be collected to monitor such aggregates as total sales for each store or for each product. The purpose of the summaries is simple: they are to be used to establish and evaluate sales trends, product sales comparisons, and so on, that serve decision needs. (How well are items selling? Should this product be discontinued? Has the advertising been effective as measured by increased sales?)
- The data models that govern operational data and decision support data are different. The operational database's frequent and rapid data updates make data anomalies a potentially devastating problem. Therefore,

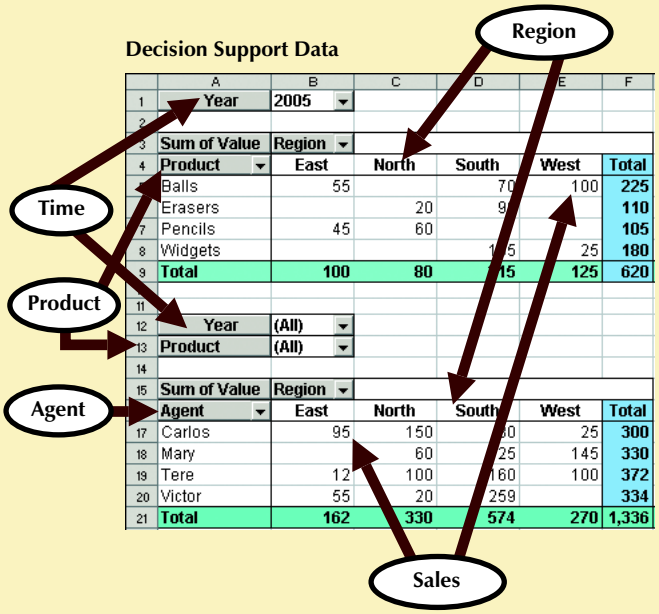
**FIGURE 13.3** **Transforming operational data into decision support data**

**Operational Data**

| | A | B | C | D | E |
|---|---|---|---|---|---|
| 3 | Year | Region | Agent | Product | Value |
| 4 | 2004 | East | Carlos | Erasers | 50 |
| 5 | 2004 | East | Tere | Erasers | 12 |
| 6 | 2004 | North | Carlos | Widgets | 120 |
| 7 | 2004 | North | Tere | Widgets | 100 |
| 8 | 2004 | North | Carlos | Widgets | 30 |
| 9 | 2004 | South | Victor | Balls | 145 |
| 10 | 2004 | South | Victor | Balls | 34 |
| 11 | 2004 | South | Victor | Balls | 80 |
| 12 | 2004 | West | Mary | Pencils | 89 |
| 13 | 2004 | West | Mary | Pencils | 56 |
| 14 | 2005 | East | Carlos | Pencils | 45 |
| 15 | 2005 | East | Victor | Balls | 55 |
| 16 | 2005 | North | Mary | Pencils | 60 |
| 17 | 2005 | North | Victor | Erasers | 20 |
| 18 | 2005 | South | Carlos | Widgets | 30 |
| 19 | 2005 | South | Mary | Widgets | 75 |
| 20 | 2005 | South | Mary | Widgets | 50 |
| 21 | 2005 | South | Tere | Balls | 70 |
| 22 | 2005 | South | Tere | Erasers | 90 |
| 23 | 2005 | West | Carlos | Widgets | 25 |
| 24 | 2005 | West | Tere | Balls | 100 |

**Operational data have a narrow time span, low granularity, and single focus. Such data are usually presented in tabular format, in which each row represents a single transaction. This format often makes it difficult to derive useful information.**

**Decision Support Data**




| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 1 | Year | 2005 | | | | |
| 2 | | | | | | |
| 3 | Sum of Value | Region | | | | |
| 4 | Product | East | North | South | West | Total |
| 5 | Balls | 55 | | 70 | 100 | 225 |
| 6 | Erasers | | 20 | 9 | | 110 |
| 7 | Pencils | 45 | 60 | | | 105 |
| 8 | Widgets | | | 15 | 25 | 180 |
| 9 | Total | 100 | 80 | 15 | 125 | 620 |
| 11 | | | | | | |
| 12 | Year | (All) | | | | |
| 13 | Product | (All) | | | | |
| 14 | | | | | | |
| 15 | Sum of Value | Region | | | | |
| 16 | Agent | East | North | South | West | Total |
| 17 | Carlos | 95 | 150 | 30 | 25 | 300 |
| 18 | Mary | | 60 | 25 | 145 | 330 |
| 19 | Tere | 12 | 100 | 160 | 100 | 372 |
| 20 | Victor | 55 | 20 | 259 | | 334 |
| 21 | Total | 162 | 330 | 574 | 270 | 1,336 |

**Decision support system (DSS) data focus on a broader time span, tend to have high levels of granularity, and can be examined in multiple dimensions. For example, note these possible aggregations:**

- **Sales by product, region, agent, etc.**
- **Sales for all years or only a few selected years.**
- **Sales for all products or only a few selected products.**

## ONLINE CONTENT

The operational data in Figure 13.3 are found in the Student Online Companion for this book. The decision support data in Figure 13.3 shows the output for the solution to Problem 2 at the end of this chapter.

the data requirements in a typical relational transaction (operational) system generally require normalized structures that yield many tables, each of which contains the minimum number of attributes. In contrast, the decision support database is not subject to such transaction updates, and the focus is on querying capability. Therefore, decision support databases tend to be non-normalized and include few tables, each of which contains a large number of attributes.

- Query activity (frequency and complexity) in the operational database tends to be low to allow additional processing cycles for the more crucial update transactions. Therefore, queries against operational data typically are narrow in scope, low in complexity, and speed-critical. In contrast, decision support data exist for the sole purpose of serving query requirements. Queries against decision support data typically are broad in scope, high in complexity, and less speed-critical.
- Finally, decision support data are characterized by very large amounts of data. The large data volume is the result of two factors. First, data are stored in non-normalized structures that are likely to display many data redundancies and duplications. Second, the same data can be categorized in many different ways to represent different snapshots. For example, sales data might be stored in relation to product, store, customer, region, and manager.

Table 13.4 summarizes the differences between operational and decision support data from the database designer's point of view.

**TABLE 13.4 Contrasting Operational and Decision Support Data Characteristics**

| CHARACTERISTIC | OPERATIONAL DATA | DECISION SUPPORT DATA |
|---|---|---|
| Data currency | Current operations<br>Real-time data | Historic data<br>Snapshot of company data<br>Time component (week/month/year) |
| Granularity | Atomic-detailed data | Summarized data |
| Summarization level | Low; some aggregate yields | High; many aggregation levels |
| Data model | Highly normalized<br>Mostly relational DBMS | Non-normalized<br>Complex structures<br>Some relational, but mostly multidimensional DBMS |
| Transaction type | Mostly updates | Mostly query |
| Transaction volumes | High update volumes | Periodic loads and summary calculations |
| Transaction speed | Updates are critical | Retrievals are critical |
| Query activity | Low to medium | High |
| Query scope | Narrow range | Broad range |
| Query complexity | Simple to medium | Very complex |
| Data volumes | Hundreds of megabytes, up to gigabytes | Hundreds of gigabytes, up to terabytes |

The many differences between operational data and decision support data are good indicators of the requirements of the decision support database, described in the next section.

### 13.4.2 Decision Support Database Requirements

A decision support database is a specialized DBMS tailored to provide fast answers to complex queries. There are four main requirements for a decision support database: the database schema, data extraction and loading, the end-user analytical interface, and database size.

**TABLE 13.5 Ten-Year Sales History for a Single-Department, in Millions of Dollars**

| YEAR | SALES |
|---|---|
| 1998 | 8,227 |
| 1999 | 9,109 |
| 2000 | 10,104 |
| 2001 | 11,553 |
| 2002 | 10,018 |
| 2003 | 11,875 |
| 2004 | 12,699 |
| 2005 | 14,875 |
| 2006 | 16,301 |
| 2007 | 19,986 |

#### Database Schema

The decision support database schema must support complex (non-normalized) data representations. As noted earlier, the decision support database must contain data that are aggregated and summarized. In addition to meeting those requirements, the queries must be able to extract multidimensional time slices. If you are using an RDBMS, the conditions suggest using non-normalized and even duplicated data. To see why this must be true, take a look at the 10-year sales history for a single store containing a single department. At this point, the data are fully normalized within the single table, as shown in Table 13.5.

This structure works well when you have only one store with only one department. However, it is very unlikely that such a simple environment has much need for a decision support database. One would suppose that a decision support database becomes a factor when dealing with more than one store, each of which has more than one department. To support all of the decision support requirements, the database must contain data for all of the stores and all of their departments—and the database must be able to support

multidimensional queries that track sales by stores, by departments, and over time. For simplicity, suppose there are only two stores (A and B) and two departments (1 and 2) within each store. Let's also change the time dimension to include yearly data. Table 13.6 shows the sales figures under the specified conditions. Only 1998, 2002, and 2007 are shown; ellipses (...) are used to indicate that data values were omitted. You can see in Table 13.6 that the number of rows and attributes already multiplies quickly and that the table exhibits multiple redundancies.

**TABLE 13.6** Yearly Sales Summaries, Two Stores and Two Departments per Store, in Millions of Dollars

| YEAR | STORE | DEPARTMENT | SALES |
|---|---|---|---|
| 1998 | A | 1 | 1,985 |
| 1998 | A | 2 | 2,401 |
| 1998 | B | 1 | 1,879 |
| 1998 | B | 2 | 1,962 |
| ... | ... | ... | ... |
| 2002 | A | 1 | 3,912 |
| 2002 | A | 2 | 4,158 |
| 2002 | B | 1 | 3,426 |
| 2002 | B | 2 | 1,203 |
| ... | ... | ... | ... |
| 2007 | A | 1 | 7,683 |
| 2007 | A | 2 | 6,912 |
| 2007 | B | 1 | 3,768 |
| 2007 | B | 2 | 1,623 |

Now suppose that the company has 10 departments per store and 20 stores nationwide. And suppose you want to access *yearly* sales summaries. Now you are dealing with 200 rows and 12 monthly sales attributes per row. (Actually, there are 13 attributes per row if you add each store's sales total for each year.)

The decision support database schema must also be optimized for query (read-only) retrievals. To optimize query speed, the DBMS must support features such as bitmap indexes and data partitioning to increase search speed. In addition, the DBMS query optimizer must be enhanced to support the non-normalized and complex structures found in decision support databases.

## Data Extraction and Filtering

The decision support database is created largely by extracting data from the operational database and by importing additional data from external sources. Thus, the DBMS must support advanced data extraction and data filtering tools. To minimize the impact on the operational database, the data extraction capabilities should allow batch and scheduled data extraction. The data extraction capabilities should also support different data sources: flat files and hierarchical, network, and relational databases, as well as multiple vendors. Data filtering capabilities must include the ability to check for inconsistent data or data validation rules. Finally, to filter and integrate the operational data into the decision support database, the DBMS must support advanced data integration, aggregation, and classification.

Using data from multiple external sources also usually means having to solve data-formatting conflicts. For example, data such as Social Security numbers and dates can occur in different formats; measurements can be based on different scales, and the same data elements can have different names. In short, data must be filtered and purified to ensure that only the pertinent decision support data are stored in the database and that they are stored in a standard format.

### End-User Analytical Interface

The decision support DBMS must support advanced data modeling and data presentation tools. Using those tools makes it easy for data analysts to define the nature and extent of business problems. Once the problems have been defined, the decision support DBMS must generate the necessary queries to retrieve the appropriate data from the decision support database. If necessary, the query results may then be evaluated with data analysis tools supported by the decision support DBMS. Because queries yield crucial information for decision makers, the queries must be optimized for speedy processing. The end-user analytical interface is one of the most critical DBMS components. When properly implemented, an analytical interface permits the user to navigate through the data to simplify and accelerate the decision-making process.

### Database Size

Decision support databases tend to be very large; gigabyte and terabyte ranges are not unusual. For example, in 2005, Wal-Mart, the world's largest company, had 260 terabytes of data in its data warehouses. As mentioned earlier, the decision support database typically contains redundant and duplicated data to improve data retrieval and simplify information generation. Therefore, the DBMS must be capable of supporting **very large databases** (**VLDBs**). To support a VLDB adequately, the DBMS might be required to use advanced hardware, such as multiple disk arrays, and even more importantly, to support multiple-processor technologies, such as a symmetric multiprocessor (SMP) or a massively parallel processor (MPP).

The complex information requirements and the ever-growing demand for sophisticated data analysis sparked the creation of a new type of data repository. This repository contains data in formats that facilitate data extraction, data analysis, and decision making. This data repository is known as a data warehouse and has become the foundation for a new generation of decision support systems.

## 13.5 THE DATA WAREHOUSE

Bill Inmon, the acknowledged "father" of the **data warehouse**, defines the term as "an *integrated*, *subject-oriented*, *time-variant*, *nonvolatile* collection of data (italics added for emphasis) that provides support for decision making."[2] To understand that definition, let's take a more detailed look at its components.

- *Integrated*. The data warehouse is a centralized, consolidated database that integrates data derived from the entire organization and from multiple sources with diverse formats. Data integration implies that all business entities, data elements, data characteristics, and business metrics *are described in the same way throughout the enterprise*. Although this requirement sounds logical, you would be amazed to discover how many different measurements for "sales performance" can exist within an organization; the same scenario holds true for any other business element. For instance, the status of an order might be indicated with text labels such as "open," "received," "cancelled," and "closed" in one department and as "1," "2," "3," and "4" in another department. A student's status might be defined as "freshman," "sophomore," "junior," or "senior" in the accounting department and as "FR," "SO," "JR," or "SR" in the computer information systems department. To avoid the potential format tangle, the data in the data warehouse must conform to a common format acceptable throughout the organization. This integration can be time-consuming, but once accomplished, it enhances decision making and helps managers better understand the company's operations. This understanding can be translated into recognition of strategic business opportunities.
- *Subject-oriented*. Data warehouse data are arranged and optimized to provide answers to questions coming from diverse functional areas within a company. Data warehouse data are organized and summarized by topic, such as sales, marketing, finance, distribution, and transportation. For each topic, the data warehouse contains

[2] Inmon, Bill and Chuck Kelley. "The Twelve Rules of Data Warehouse for a Client/Server World," *Data Management Review*, 4(5), May 1994, pp. 6–16.

specific subjects of interest—products, customers, departments, regions, promotions, and so on. This form of data organization is quite different from the more functional or process-oriented organization of typical transaction systems. For example, an invoicing system designer concentrates on designing normalized data structures (relational tables) to support the business process by storing invoice components in two tables: INVOICE and INVLINE. In contrast, the data warehouse has a *subject* orientation. Data warehouse designers focus specifically on the data rather than on the processes that modify the data. (After all, data warehouse data are not subject to numerous real-time data updates!) Therefore, instead of storing an invoice, the data warehouse stores its "sales by product" and "sales by customer" components because decision support activities require the retrieval of sales summaries by product or customer.

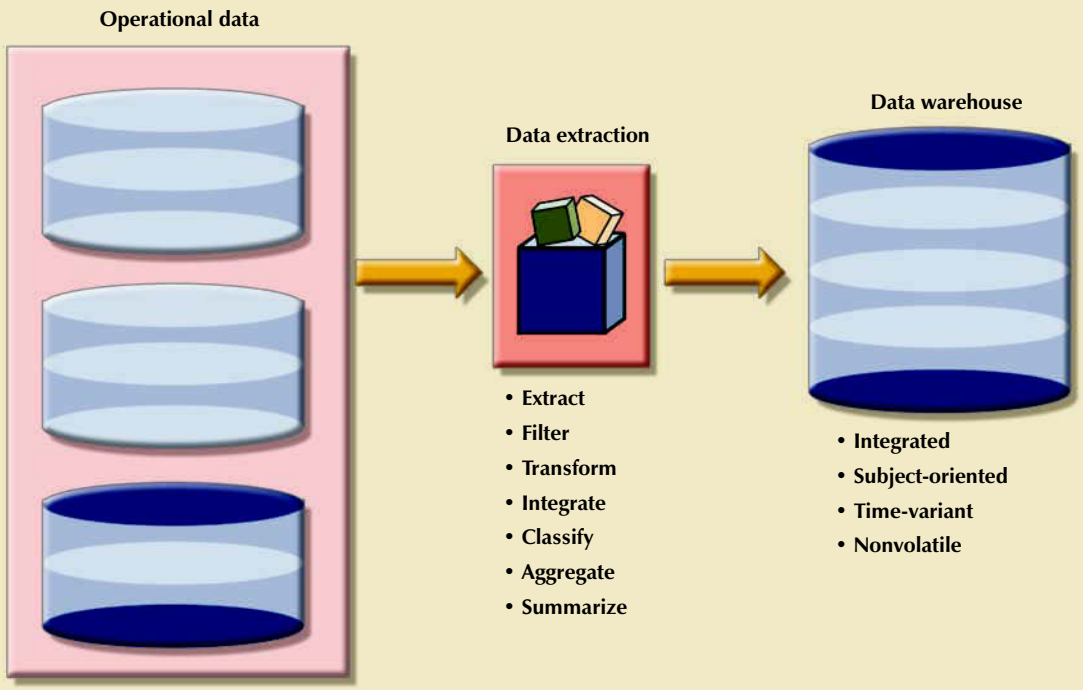
- *Time-variant.* In contrast to operational data, which focus on current transactions, warehouse data represent the flow of data through time. The data warehouse can even contain projected data generated through statistical and other models. It is also time-variant in the sense that once data are periodically uploaded to the data warehouse, all time-dependent aggregations are recomputed. For example, when data for previous weekly sales are uploaded to the data warehouse, the weekly, monthly, yearly, and other time-dependent aggregates for products, customers, stores, and other variables are also updated. Because data in a data warehouse constitute a snapshot of the company history as measured by its variables, the time component is crucial. The data warehouse contains a time ID that is used to generate summaries and aggregations by week, month, quarter, year, and so on. Once the data enter the data warehouse, the time ID assigned to the data cannot be changed.
- *Nonvolatile.* Once data enter the data warehouse, they are never removed. Because the data in the warehouse represent the company's history, the operational data, representing the near-term history, are always added to it. Because data are never deleted and new data are continually added, the data warehouse is always growing. That's why the DBMS must be able to support multigigabyte and even multiterabyte databases, operating on multiprocessor hardware. Table 13.7 summarizes the differences between data warehouses and operational databases.

**TABLE 13.7 Characteristics of Data Warehouse Data and Operational Database Data**

| CHARACTERISTIC | OPERATIONAL DATABASE DATA | DATA WAREHOUSE DATA |
|---|---|---|
| Integrated | Similar data can have different representations or meanings. For example, Social Security numbers may be stored as ###-##-#### or as #########, and a given condition may be labeled as T/F or 0/1 or Y/N. A sales value may be shown in thousands or in millions. | Provide a unified view of all data elements with a common definition and representation for all business units. |
| Subject-oriented | Data are stored with a functional, or process, orientation. For example, data may be stored for invoices, payments, and credit amounts. | Data are stored with a subject orientation that facilitates multiple views of the data and facilitates decision making. For example, sales may be recorded by product, by division, by manager, or by region. |
| Time-variant | Data are recorded as current transactions. For example, the sales data may be the sale of a product on a given date, such as $342.78 on 12-MAY-2008. | Data are recorded with a historical perspective in mind. Therefore, a time dimension is added to facilitate data analysis and various time comparisons. |
| Nonvolatile | Data updates are frequent and common. For example, an inventory amount changes with each sale. Therefore, the data environment is fluid. | Data cannot be changed. Data are added only periodically from historical systems. Once the data are properly stored, no changes are allowed. Therefore, the data environment is relatively static. |

In summary, the data warehouse is usually a read-only database optimized for data analysis and query processing. Typically, data are extracted from various sources and are then transformed and integrated—in other words, passed through a data filter—before being loaded into the data warehouse. Users access the data warehouse via front-end tools and/or end-user application software to extract the data in usable form. Figure 13.4 illustrates how a data warehouse is created from the data contained in an operational database.

**FIGURE 13.4** Creating a data warehouse




Although the centralized and integrated data warehouse can be a very attractive proposition that yields many benefits, managers may be reluctant to embrace this strategy. Creating a data warehouse requires time, money, and considerable managerial effort. Therefore, it is not surprising that many companies begin their foray into data warehousing by focusing on more manageable data sets that are targeted to meet the special needs of small groups within the organization. These smaller data stores are called data marts. A **data mart** is a small, single-subject data warehouse subset that provides decision support to a small group of people. In addition, a data mart could also be created from data extracted from a larger data warehouse with the specific function to support faster data access to a target group or function. That is, data marts and data warehouses can coexist within a business intelligence environment.

Some organizations choose to implement data marts not only because of the lower cost and shorter implementation time, but also because of the current technological advances and inevitable "people issues" that make data marts attractive. Powerful computers can provide a customized decision support system to small groups in ways that might not be possible with a centralized system. Also, a company's culture may predispose its employees to resist major changes, but they might quickly embrace relatively minor changes that lead to demonstrably improved decision support. In addition, people at different organizational levels are likely to require data with different summarization, aggregation, and presentation formats. Data marts can serve as a test vehicle for companies exploring the potential benefits of data warehouses. By migrating gradually from data marts to data warehouses, a specific department's decision support needs can be addressed within a reasonable time frame (six months to one year), as compared to the

longer time frame usually required to implement a data warehouse (one to three years). Information technology (IT) departments also benefit from this approach because their personnel have the opportunity to learn the issues and develop the skills required to create a data warehouse.

The only difference between a data mart and a data warehouse is the size and scope of the problem being solved. Therefore, the problem definitions and data requirements are essentially the same for both. To be useful, the data warehouse must conform to uniform structures and formats to avoid data conflicts and to support decision making. In fact, before a decision support database can be considered a true data warehouse, it must conform to the rules described in the next section.

### 13.5.1 Twelve Rules that Define a Data Warehouse

In 1994, William H. Inmon and Chuck Kelley created 12 rules defining a data warehouse, which summarize many of the points made in this chapter about data warehouses.[3]

1. The data warehouse and operational environments are separated.
2. The data warehouse data are integrated.
3. The data warehouse contains historical data over a long time.
4. The data warehouse data are snapshot data captured at a given point in time.
5. The data warehouse data are subject oriented.
6. The data warehouse data are mainly read-only with periodic batch updates from operational data. No online updates are allowed.
7. The data warehouse development life cycle differs from classical systems development. The data warehouse development is data-driven; the classical approach is process-driven.
8. The data warehouse contains data with several levels of detail: current detail data, old detail data, lightly summarized data, and highly summarized data.
9. The data warehouse environment is characterized by read-only transactions to very large data sets. The operational environment is characterized by numerous update transactions to a few data entities at a time.
10. The data warehouse environment has a system that traces data sources, transformations, and storage.
11. The data warehouse's metadata are a critical component of this environment. The metadata identify and define all data elements. The metadata provide the source, transformation, integration, storage, usage, relationships, and history of each data element.
12. The data warehouse contains a chargeback mechanism for resource usage that enforces optimal use of the data by end users.

Note how those 12 rules capture the complete data warehouse life cycle—from its introduction as an entity separate from the operational data store to its components, functionality, and management processes. The next section illustrates the historical progression of decision support architectural styles. This discussion will help you understand how the data store components evolved to produce the data warehouse.

### 13.5.2 Decision Support Architectural Styles

Several decision support database architectural styles are available. These architectures provide advanced decision support features, and some are capable of providing access to multidimensional data analysis. Table 13.8 summarizes the main architectural styles that you are likely to encounter in the decision support database environment.

[3] Inmon, Bill and Chuck Kelley. "The Twelve Rules of Data Warehouse for a Client/Server World," *Data Management Review*, 4 (5), May 1994, pp. 6–16.

**TABLE 13.8** Decision Support Architectural Styles

| SYSTEM TYPE | SOURCE DATA | DATA EXTRACTION/ INTEGRATION PROCESS | DECISION SUPPORT DATA STORE | END-USER QUERY TOOL | END USER PRESENTATION TOOL |
|---|---|---|---|---|---|
| Traditional mainframe-based online transaction processing (OLTP) | Operational data | None<br>Reports, reads, and summarizes data directly from operational data | None<br>Temporary files used for reporting purposes | Very basic<br>Predefined reporting formats<br>Basic sorting, totaling, and averaging | Very basic<br>Menu-driven, predefined reports, text and numbers only |
| Managerial information system (MIS) with third-generation language (3GL) | Operational data | Basic extraction and aggregation<br>Reads, filters, and summarizes operational data into intermediate data store | Lightly aggregated data in RDBMS | Same as above, in addition to some ad hoc reporting using SQL | Same as above, in addition to some ad hoc columnar report definitions |
| First-generation departmental DSS | Operational data | Data extraction and integration process to populate a DSS data store; is run periodically | First DSS database generation<br>Usually RDBMS | Query tool with some analytical capabilities and predefined reports | Advanced presentation tools with plotting and graphics capabilities |
| First-generation enterprise data warehouse using RDBMS | Operational data<br>External data (census data) | Advanced data extraction and integration tools<br>Features include access to diverse data sources, transformations, filters, aggregations, classifications, scheduling, and conflict resolution | Data warehouse integrated decision support database to support the entire organization<br>Uses RDBMS technology optimized for query purposes<br>Star schema model | Same as above, in addition to support for more advanced queries and analytical functions with extensions | Same as above, in addition to additional multidimensional presentation tools with drill-down capabilities |
| Second-generation data warehouse using multidimensional database management system (MDBMS) | Operational data<br>External data (Industry group data) | Same as above | Data warehouse stores data by using MDBMS technology based on data structures; referred to as cubes with multiple dimensions | Same as above, but uses different query interface to access MDBMS (proprietary) | Same as above, but uses cubes and multidimensional matrixes<br>Limited in terms of cube size |

You might be tempted to think that the data warehouse is just a big summarized database. The previous discussion indicates that a good data warehouse is much more than that. A complete data warehouse architecture includes support for a decision support data store, a data extraction and integration filter, and a specialized presentation interface. In the next section you will learn more about a common decision support architectural style known as Online Analytical Processing (OLAP).

## 13.6 ONLINE ANALYTICAL PROCESSING

The need for more intensive decision support prompted the introduction of a new generation of tools. Those new tools, called **online analytical processing** (**OLAP**), create an advanced data analysis environment that supports decision making, business modeling, and operations research. OLAP systems share four main characteristics:

- They use multidimensional data analysis techniques.
- They provide advanced database support.
- They provide easy-to-use end-user interfaces.
- They support client/server architecture.

Let's examine each of those characteristics.

### 13.6.1 Multidimensional Data Analysis Techniques

The most distinct characteristic of modern OLAP tools is their capacity for multidimensional analysis. In multidimensional analysis, data are processed and viewed as part of a multidimensional structure. This type of data analysis is particularly attractive to business decision makers because they tend to view business data as data that are related to other business data.

To better understand this view, let's examine how, as a business data analyst, you might investigate sales figures. In this case, you are probably interested in the sales figures as they relate to other business variables such as customers and time. In other words, customers and time are viewed as different dimensions of sales. Figure 13.5 illustrates how the operational (one-dimensional) view differs from the multidimensional view of sales.

Note in Figure 13.5 that the tabular (operational) view of sales data is not well suited to decision support, because the relationship between INVOICE and LINE does not provide a business perspective of the sales data. On the other hand, the end user's view of sales data *from a business perspective* is more closely represented by the multidimensional view of sales than by the tabular view of separate tables. Note also that the multidimensional view allows end users to consolidate or aggregate data at different levels: total sales figures by customers and by date. Finally, the multidimensional view of data allows a business data analyst to easily switch business perspectives (dimensions) from sales by customer to sales by division, by region, and so on.

Multidimensional data analysis techniques are augmented by the following functions:

- *Advanced data presentation functions*. 3-D graphics, pivot tables, crosstabs, data rotation, and three-dimensional cubes. Such facilities are compatible with desktop spreadsheets, statistical packages, and query and report packages.
- *Advanced data aggregation, consolidation, and classification functions.* These allow the data analyst to create multiple data aggregation levels, slice and dice data (see Section 13.6.3), and drill down and roll up data across different dimensions and aggregation levels. For example, aggregating data across the time dimension (by week, month, quarter, and year) allows the data analyst to drill down and roll up across time dimensions.
- *Advanced computational functions*. These include business-oriented variables (market share, period comparisons, sales margins, product margins, and percentage changes), financial and accounting ratios (profitability, overhead, cost allocations, and returns), and statistical and forecasting functions. These functions are provided automatically, and the end user does not need to redefine their components each time they are accessed.
- *Advanced data modeling functions*. These provide support for what-if scenarios, variable assessment, variable contributions to outcome, linear programming, and other modeling tools.

**FIGURE 13.5** Operational vs. multidimensional view of sales

**Database name: Ch13_Text**

**Table name: DW_INVOICE**

| INV_NUM | INV_DATE | CUS_NAME | INV_TOTAL |
|---|---|---|---|
| 2034 | 15-May-08 | Dartonik | 1400.00 |
| 2035 | 15-May-08 | Summer Lake | 1200.00 |
| 2036 | 16-May-08 | Dartonik | 1350.00 |
| 2037 | 16-May-08 | Summer lake | 3100.00 |
| 2038 | 16-May-08 | Trydon | 400.00 |

**Table name: DW_LINE**

| INV_NUM | LINE_NUM | PROD_DESCRIPTION | LINE_PRICE | LINE_QUANTITY | LINE_AMOUNT |
|---|---|---|---|---|---|
| 2034 | 1 | Optical Mouse | 45.00 | 20 | 900.00 |
| 2034 | 2 | Wireless RF remote and laser pointer | 50.00 | 10 | 500.00 |
| 2035 | 1 | Everlast Hard Drive, 60 GB | 200.00 | 6 | 1200.00 |
| 2036 | 1 | Optical Mouse | 45.00 | 30 | 1350.00 |
| 2037 | 1 | Optical Mouse | 45.00 | 10 | 450.00 |
| 2037 | 2 | Roadster 56KB Ext. Modem | 120.00 | 5 | 600.00 |
| 2037 | 3 | Everlast Hard Drive, 60 GB | 205.00 | 10 | 2050.00 |
| 2038 | 1 | NoTech Speaker Set | 50.00 | 8 | 400.00 |

**Multidimensional View of Sales**

| | Time Dimension | | |
|---|---|---|---|
| **Customer Dimension** | **15-May-08** | **16-May-08** | **Totals** |
| **Dartonik** | $1,400.00 | $1,350.00 | **$2,750.00** |
| **Summer Lake** | $1,800.00 | $3,100.00 | **$4,900.00** |
| **Trydon** | | $400.00 | **$400.00** |
| **Totals** | **$3,200.00** | **$4,850.00** | **$8,050.00** |

**Sales are located in the intersection of a customer row and time column**

**Aggregations are provided for both dimensions**

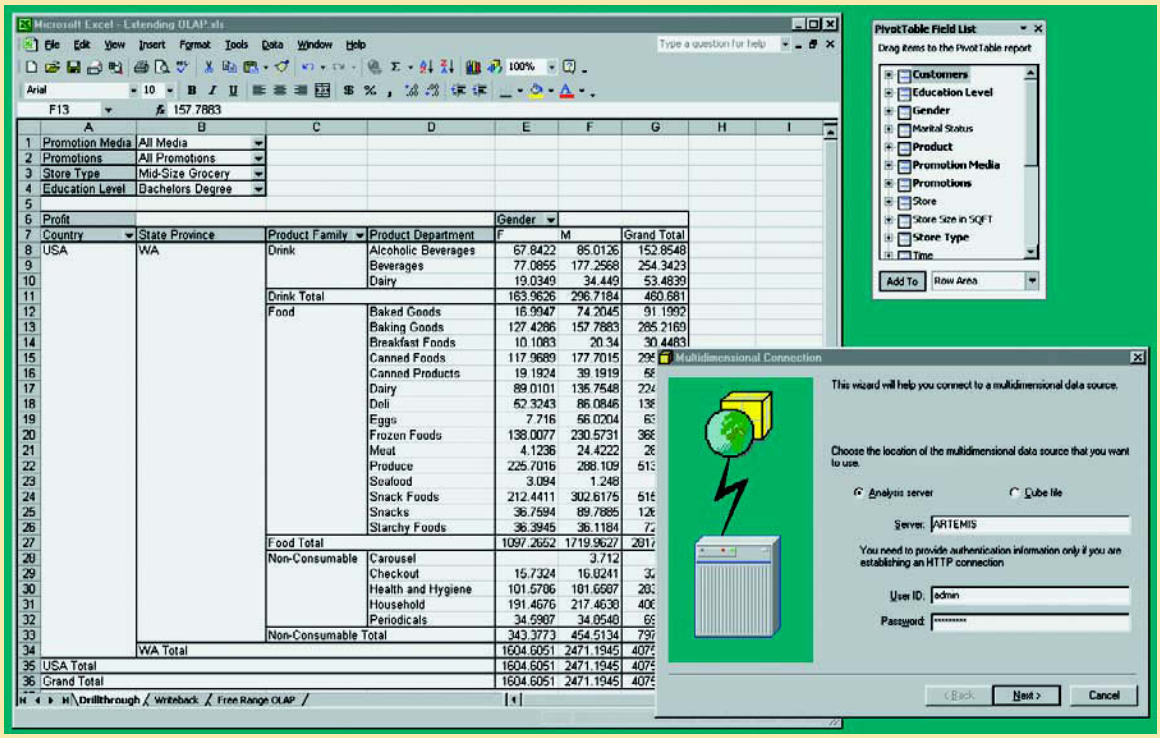
Because many analysis and presentation functions are common to desktop spreadsheet packages, most OLAP vendors have closely integrated their systems with spreadsheets such as Microsoft Excel and IBM Lotus 1-2-3. Using the features available in graphical end-user interfaces such as Windows, the OLAP menu option simply becomes another option within the spreadsheet menu bar, as shown in Figure 13.6. This seamless integration is an advantage for OLAP systems and for spreadsheet vendors because end users gain access to advanced data analysis features by using familiar programs and interfaces. Therefore, additional training and development costs are minimized.

### 13.6.2 Advanced Database Support

To deliver efficient decision support, OLAP tools must have advanced data access features. Such features include:

- Access to many different kinds of DBMSs, flat files, and internal and external data sources.
- Access to aggregated data warehouse data as well as to the detail data found in operational databases.
- Advanced data navigation features such as drill-down and roll-up.
- Rapid and consistent query response times.

**FIGURE 13.6** Integration of OLAP with a spreadsheet program




- The ability to map end-user requests, expressed in either business or model terms, to the appropriate data source and then to the proper data access language (usually SQL). The query code must be optimized to match the data source, regardless of whether the source is operational or data warehouse data.
- Support for very large databases. As already explained, the data warehouse can easily and quickly grow to multiple gigabytes and even terabytes.

To provide a seamless interface, OLAP tools map the data elements from the data warehouse and from the operational database to their own data dictionaries. These metadata are used to translate end-user data analysis requests into the proper (optimized) query codes, which are then directed to the appropriate data source(s).

### 13.6.3 Easy-to-Use End-User Interface

Advanced OLAP features become more useful when access to them is kept simple. OLAP tool vendors learned this lesson early and have equipped their sophisticated data extraction and analysis tools with easy-to-use graphical interfaces. Many of the interface features are "borrowed" from previous generations of data analysis tools that are already familiar to end users. This familiarity makes OLAP easily accepted and readily used.

### 13.6.4 Client/Server Architecture

Client/server architecture provides a framework within which new systems can be designed, developed, and implemented. The client/server environment enables an OLAP system to be divided into several components that define its architecture. Those components can then be placed on the same computer, or they can be distributed among several computers. Thus, OLAP is designed to meet ease-of-use requirements while keeping the system flexible.

**ONLINE CONTENT**

If necessary, review the coverage in **Appendix F, Client/Server Systems** in the Student Online Companion for this book, which provides an in-depth look at client/server system architecture and principles.

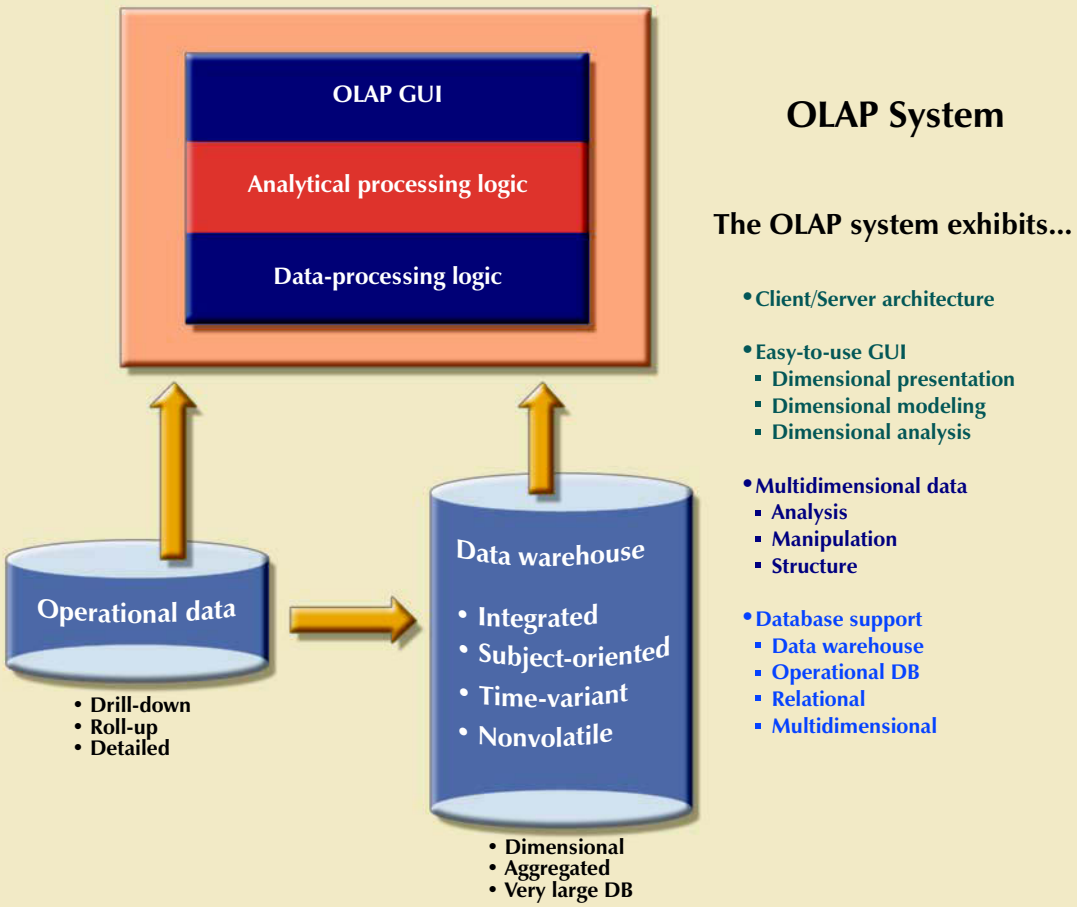
### 13.6.5 OLAP Architecture

OLAP operational characteristics can be divided into three main modules:

- Graphical user interface (GUI).
- Analytical processing logic.
- Data-processing logic.

In the client/server environment, those three OLAP modules make the defining features of OLAP possible: multidimensional data analysis, advanced database support, and an easy-to-use interface. Figure 13.7 illustrates OLAP's client/server components and attributes.

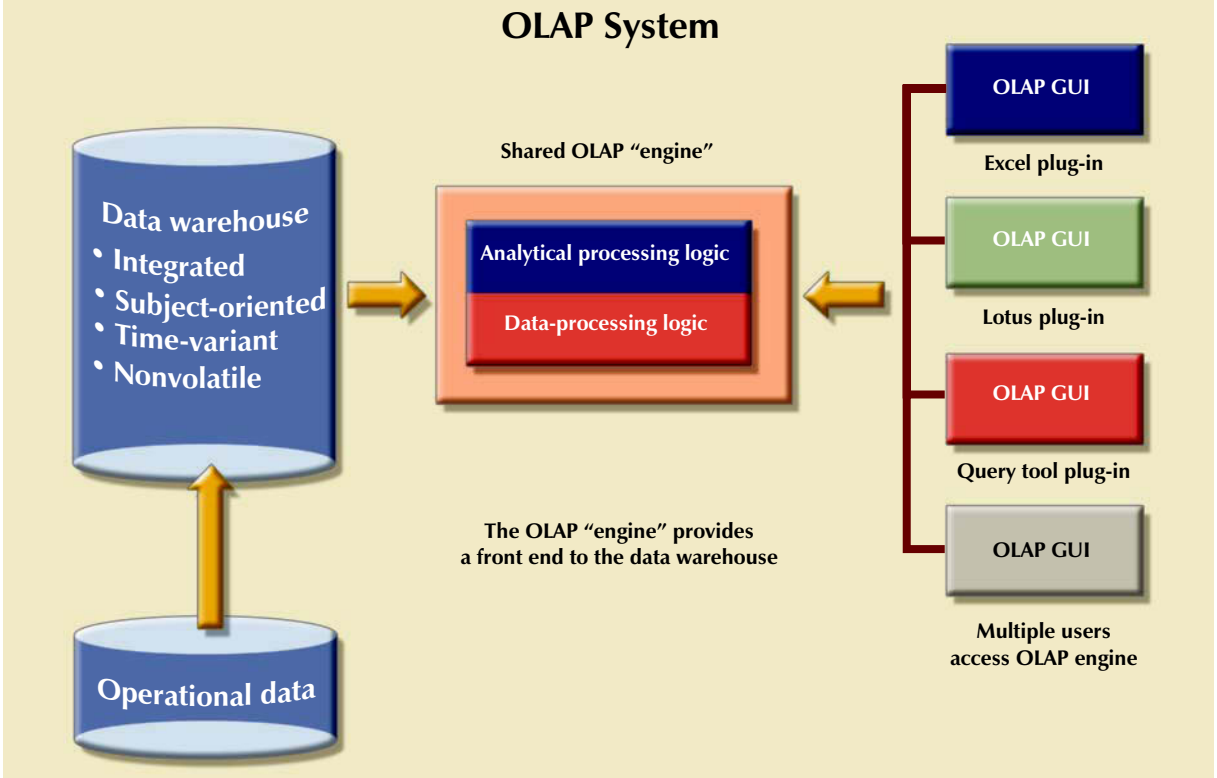
**FIGURE 13.7** OLAP client/server architecture



As Figure 13.7 illustrates, OLAP systems are designed to use both operational and data warehouse data. Figure 13.7 shows the OLAP system components located on a single computer, but this single-user scenario is only one of many. In fact, one problem with the installation shown here is that each data analyst must have a powerful computer to store the OLAP system and perform all data processing locally. In addition, each analyst uses a separate copy of the data. Therefore, the data copies must be synchronized to ensure that analysts are working with the same data. In other words, each end user must have his/her own "private" copy (extract) of the data and programs, thus returning to the *islands of information* problems discussed in Chapter 1, Database Systems. This approach does not provide the benefits of a single business image shared among all users.

A more common and practical architecture is one in which the OLAP GUI runs on client workstations, while the OLAP engine, or server, composed of the OLAP analytical processing logic and OLAP data-processing logic, runs on a shared computer. In that case, the OLAP server will be a front end to the data warehouse's decision support data. This front end or middle layer (because it sits between the data warehouse and the end-user GUI) accepts and processes the data-processing requests generated by the many end-user analytical tools. The end-user GUI might be a custom-made program or, more likely, a plug-in module that is integrated with Lotus 1-2-3, Microsoft Excel, or a third-party data analysis and query tool. Figure 13.8 illustrates such an arrangement.

**FIGURE 13.8** OLAP server arrangement




Note in Figure 13.8 that the data warehouse is created and maintained by a process or software tool that is independent of the OLAP system. This independent software performs the data extraction, filtering, and integration necessary to transform operational data into data warehouse data. This scenario reflects the fact that in most cases, the data warehousing and data analysis activities are handled separately.

At this point, you might ask why you need a data warehouse if OLAP provides the necessary multidimensional data analysis of operational data. The answer lies in the definition of OLAP. OLAP is defined as an "advanced data analysis environment that supports decision making, business modeling, and research activities." The keyword here is *environment*, which includes client/server technology. Environment is defined as "surroundings or atmosphere." And an atmosphere surrounds a nucleus. *In this case, the nucleus is composed of all business activities within an organization as represented by the operational data*. Just as there are several layers within the atmosphere, there are several layers of data processing, each outer layer representing a more aggregated data analysis. The fact is that an OLAP system might access both data storage types (operational or data warehouse) or only one; it depends on the vendor's implementation of the product selected. In any case, multidimensional data analysis requires some type of multidimensional data representation, which is normally provided by the OLAP engine.

In most implementations, the data warehouse and OLAP are interrelated, complementary environments. While the data warehouse holds integrated, subject-oriented, time-variant, and nonvolatile decision support data, the OLAP system provides the front end through which end users access and analyze such data. Yet an OLAP system can also directly access operational data, transforming it and storing it in a multidimensional structure. In other words, the OLAP system can provide a multidimensional data store component, as shown in Figure 13.9.

**FIGURE 13.9** **OLAP server with multidimensional data store arrangement**

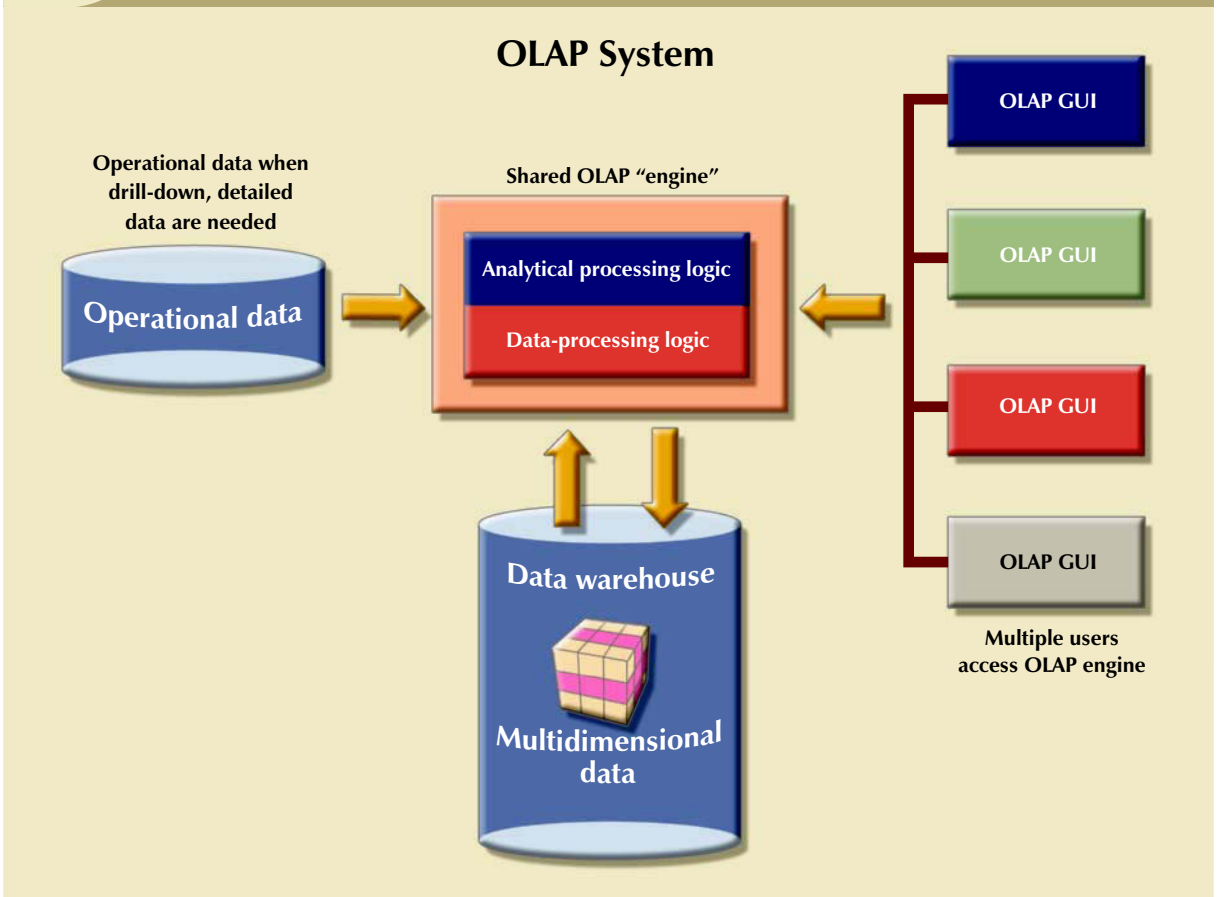
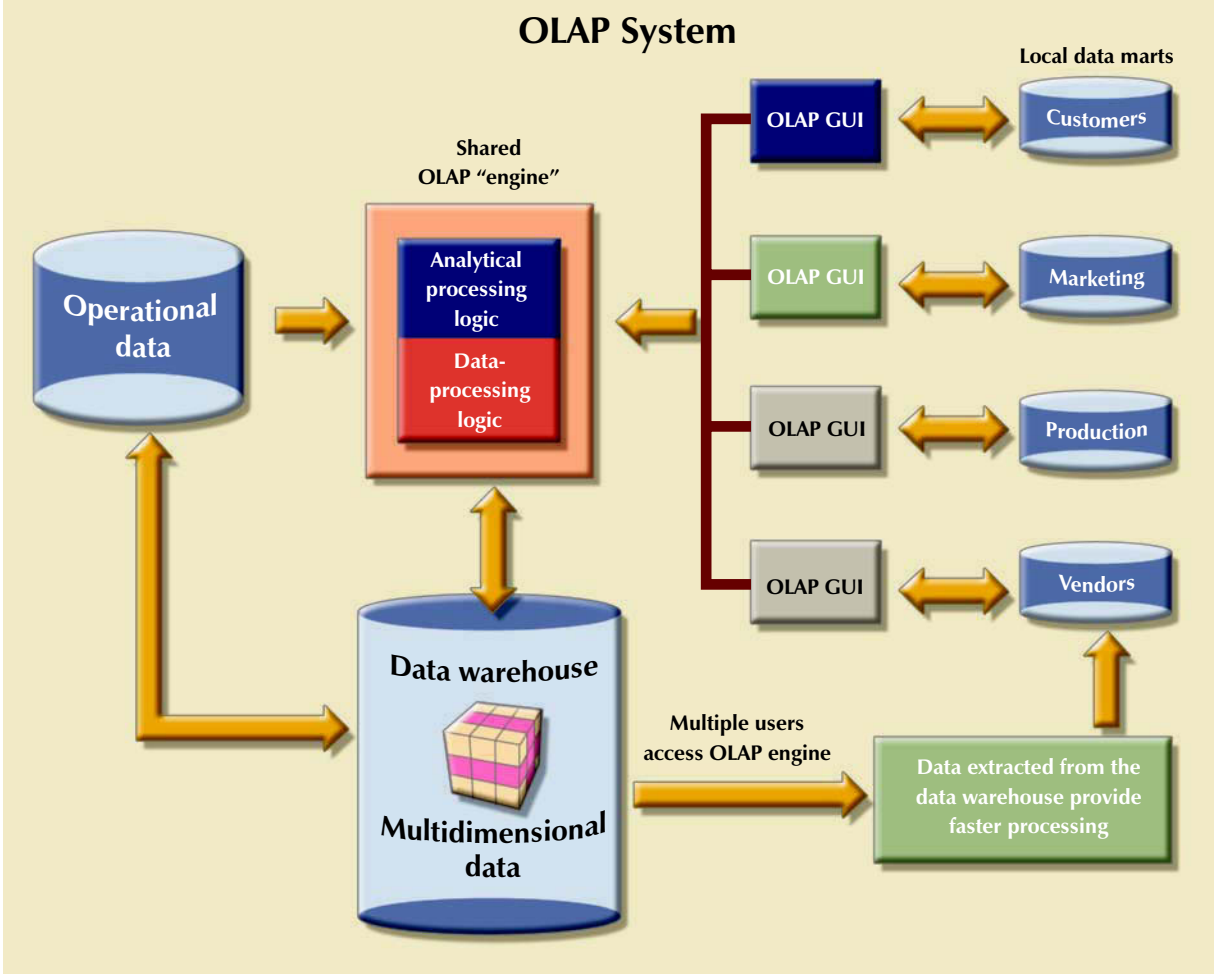



Figure 13.9 represents a scenario in which the OLAP engine extracts data from an operational database and then stores it in a multidimensional structure for further data analysis. The extraction process follows the same conventions used with data warehouses. Therefore, the OLAP provides a mini data-warehouse component that looks remarkably

like the data mart mentioned in previous sections. In this scenario, the OLAP engine has to perform all of the data extraction, filtering, integration, classification, and aggregation functions that the data warehouse normally provides. In fact, when properly implemented, the data warehouse performs all data preparation functions instead of letting OLAP perform those chores; as a result, there is no duplication of functions. Better yet, the data warehouse handles the data component more efficiently than OLAP does; so you can appreciate the benefits of having a central data warehouse serve as the large enterprise decision support database.

To provide better performance, some OLAP systems merge the data warehouse and data mart approaches by storing small extracts of the data warehouse at end-user workstations. The objective is to increase the speed of data access and data visualization (the graphic representations of data trends and characteristics). The logic behind that approach is the assumption that most end users usually work with fairly small, stable data warehouse data subsets. For example, a sales analyst is most likely to work with sales data, whereas a customer representative is likely to work with customer data. Figure 13.10 illustrates that scenario.

**FIGURE 13.10** OLAP server with local mini data marts




Whatever the arrangement of the OLAP components, one thing is certain: multidimensional data must be used. But how are multidimensional data best stored and managed? OLAP proponents are sharply divided. Some favor the use of relational databases to store the multidimensional data; others argue for the superiority of specialized multidimensional databases to store multidimensional data. The basic characteristics of each approach are examined next.

### 13.6.6 RELATIONAL OLAP

**Relational online analytical processing** (**ROLAP**) provides OLAP functionality by using relational databases and familiar relational query tools to store and analyze multidimensional data. That approach builds on existing relational technologies and represents a natural extension to all of the companies that already use relational database management systems within their organizations. ROLAP adds the following extensions to traditional RDBMS technology:

- Multidimensional data schema support within the RDBMS.
- Data access language and query performance optimized for multidimensional data.
- Support for very large databases (VLDBs).

#### Multidimensional Data Schema Support within the RDBMS

Relational technology uses normalized tables to store data. The reliance on normalization as the design methodology for relational databases is seen as a stumbling block to its use in OLAP systems. Normalization divides business entities into smaller pieces to produce the normalized tables. For example, sales data components might be stored in four or five different tables. The reason for using normalized tables is to reduce redundancies, thereby eliminating data anomalies, and to facilitate data updates. Unfortunately, for decision support purposes, it is easier to understand data when they are seen with respect to other data. (See the example in Figure 13.5.) Given that view of the data environment, this book has stressed that decision support data tend to be non-normalized, duplicated, and pre-aggregated. Those characteristics seem to preclude the use of standard relational design techniques and RDBMSs as the foundation for multidimensional data.

Fortunately for those heavily invested in relational technology, ROLAP uses a special design technique to enable RDBMS technology to support multidimensional data representations. This special design technique is known as a star schema, which is covered in detail in Section 13.7.

The star schema is designed to optimize data query operations rather than data update operations. Naturally, changing the data design foundation means that the tools used to access such data will have to change. End users who are familiar with the traditional relational query tools will discover that those tools do not work efficiently with the new star schema. However, ROLAP saves the day by adding support for the star schema when familiar query tools are used. ROLAP provides advanced data analysis functions and improves query optimization and data visualization methods.

#### Data Access Language and Query Performance Optimized for Multidimensional Data

Another criticism of relational databases is that SQL is not suited for performing advanced data analysis. Most decision support data requests require the use of multiple-pass SQL queries or multiple nested SQL statements. To answer this criticism, ROLAP extends SQL so that it can differentiate between access requirements for data warehouse data (based on the star schema) and operational data (normalized tables). In that way, a ROLAP system is able to generate the SQL code required to access the star schema data.

Query performance is also improved because the query optimizer is modified to identify the SQL code's intended query targets. For example, if the query target is the data warehouse, the optimizer passes the requests to the data warehouse. However, if the end user performs drill-down queries against operational data, the query optimizer identifies that operation and properly optimizes the SQL requests before passing them through to the operational DBMS.

Another source of improved query performance is the use of advanced indexing techniques such as bitmapped indexes within relational databases. As the name suggests, a bitmapped index is based on 0 and 1 bits to represent a given condition. For example, if the REGION attribute in Figure 13.3 has only four outcomes—North, South, East, and West—those outcomes may be represented as shown in Table 13.9. (Only the first 10 rows from Figure 13.3 are represented in Table 13.9. The "1" represents "bit on," and the "0" represents "bit off." For example, to represent a row with a REGION attribute = "East," only the "East" bit would be on. Note that each row must be represented in the index table.)

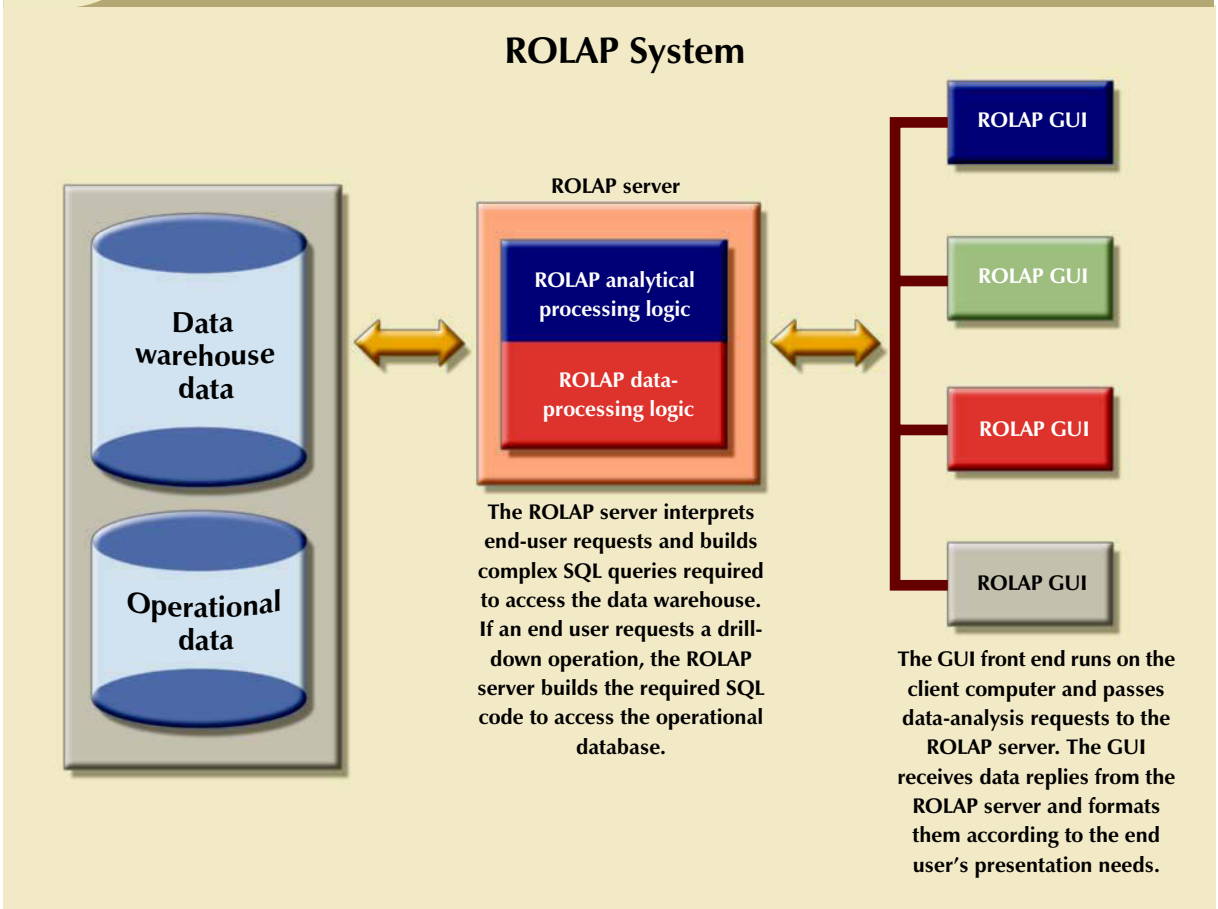
**TABLE 13.9 Bitmap Representation of Region Values**

| NORTH | SOUTH | EAST | WEST |
|---|---|---|---|
| 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 |

Note that the index in Table 13.9 takes a minimum amount of space. Therefore, bitmapped indexes are more efficient at handling large amounts of data than are the indexes typically found in many relational databases. But do keep in mind that bitmapped indexes are primarily used in situations where the number of possible values for an attribute (in other words, the attribute domain) is fairly small. For example, REGION has only four outcomes in this example. Marital status—married, single, widowed, divorced—would be another good bitmapped index candidate, as would gender—M or F.

ROLAP tools are mainly client/server products in which the end-user interface, the analytical processing, and the data processing take place on different computers. Figure 13.11 shows the interaction of the client/server ROLAP components.

**FIGURE 13.11 Typical ROLAP client/server architecture**



## Support for Very Large Databases

Recall that support for VLDBs is a requirement for decision support databases. Therefore, when the relational database is used in a decision support role, it also must be able to store very large amounts of data. Both the storage capability and the process of loading data into the database are crucial. Therefore, the RDBMS must have the proper tools to import, integrate, and populate the data warehouse with data. Decision support data are normally loaded in bulk (batch) mode from the operational data. However, batch operations require that both the source and the destination databases be reserved (locked). The speed of the data-loading operations is important, especially when you realize that most operational systems run 24 hours a day, 7 days a week, 52 weeks a year. Therefore, the window of opportunity for maintenance and batch loading is open only briefly, typically during slack periods.

With an open client/server architecture, ROLAP provides advanced decision support capabilities that are scalable to the entire enterprise. Clearly, ROLAP is a logical choice for companies that already use relational databases for their operational data. Given the size of the relational database market, it is hardly surprising that most current RDBMS vendors have extended their products to support data warehouses.

### 13.6.7 MULTIDIMENSIONAL OLAP

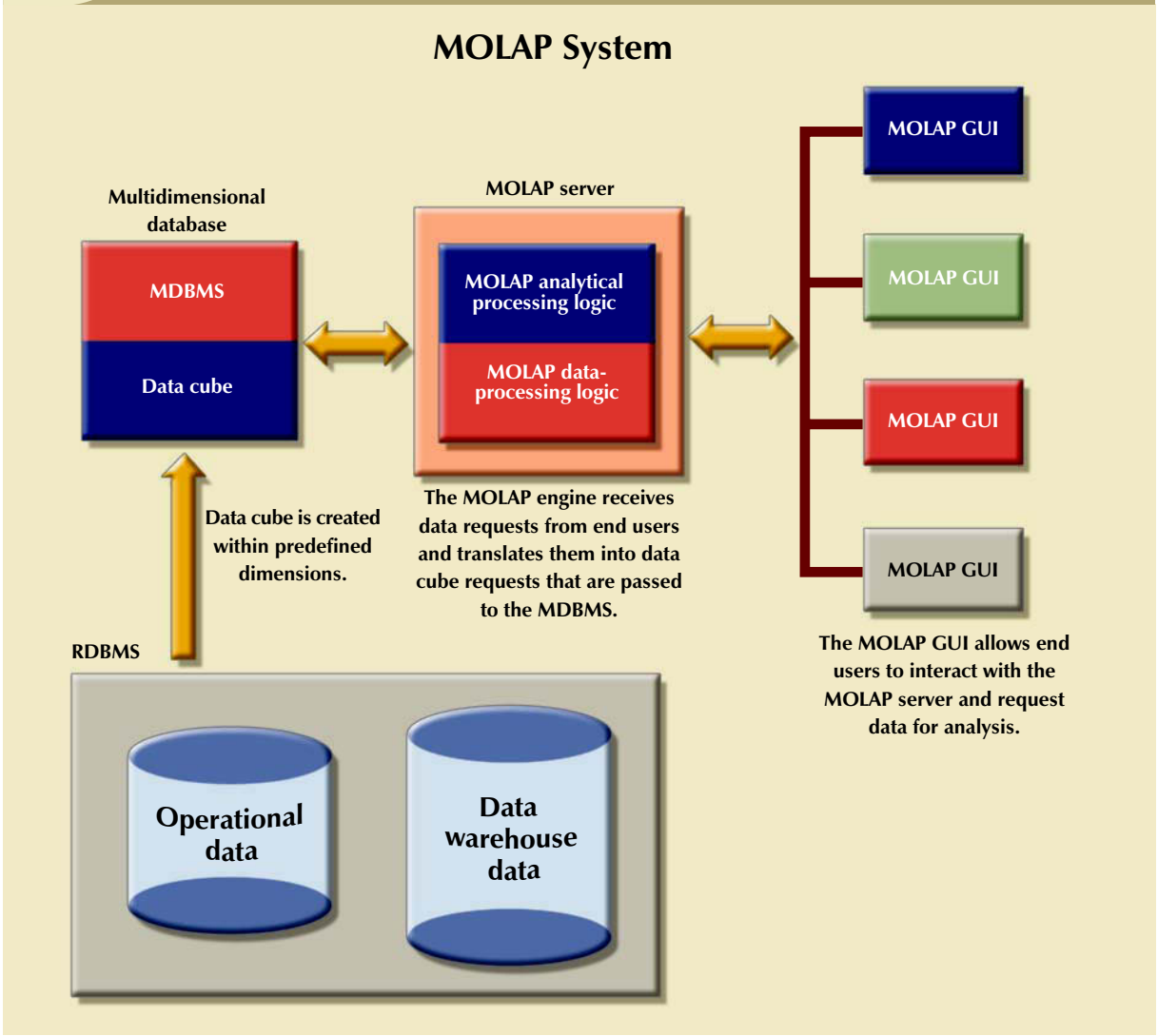
**Multidimensional online analytical processing** (**MOLAP**) extends OLAP functionality to **multidimensional database management systems** (**MDBMSs**). (An MDBMS uses special proprietary techniques to store data in matrix-like *n*-dimensional arrays.) MOLAP's premise is that multidimensional databases are best suited to manage, store, and analyze multidimensional data. Most of the proprietary techniques used in MDBMSs are derived from engineering fields such as computer-aided design/computer-aided manufacturing (CAD/CAM) and geographic information systems (GIS).

Conceptually, MDBMS end users visualize the stored data as a three-dimensional cube known as a **data cube**. The location of each data value in the data cube is a function of the x-, y-, and z-axes in a three-dimensional space. The x-, y-, and z-axes represent the dimensions of the data value. The data cubes can grow to *n* number of dimensions, thus becoming *hypercubes*. Data cubes are created by extracting data from the operational databases or from the data warehouse. One important characteristic of data cubes is that they are static; that is, they are not subject to change and must be created before they can be used. Data cubes cannot be created by ad hoc queries. Instead, you query pre-created cubes with defined axes; for example, a cube for sales will have the product, location, and time dimensions, and you can query only those dimensions. Therefore, the data cube creation process is critical and requires in-depth front-end design work. The front-end design work may be well justified because MOLAP databases are known to be much faster than their ROLAP counterparts, especially when dealing with small to medium data sets. To speed data access, data cubes are normally held in memory in what is called the **cube cache**. (A data cube is only a window to a predefined subset of data in the database. A *data cube* and a *database* are not the same thing.) Because MOLAP also benefits from a client/server infrastructure, the cube cache can be located at the MOLAP server, at the MOLAP client, or in both locations. Figure 13.12 shows the basic MOLAP architecture.

Because the data cube is predefined with a set number of dimensions, the addition of a new dimension requires that the entire data cube be re-created. This re-creation process is time consuming. Therefore, when data cubes are created too often, the MDBMS loses some of its speed advantage over the relational database. And although MDBMSs have performance advantages over relational databases, the MDBMS is best suited to small and medium data sets. Scalability is somewhat limited because the size of the data cube is restricted to avoid lengthy data access times caused by having less work space (memory) available for the operating system and the application programs. In addition, the MDBMS makes use of proprietary data storage techniques that, in turn, require proprietary data access methods using a multidimensional query language.

Multidimensional data analysis is also affected by how the database system handles sparsity. **Sparsity** is a measurement of the density of the data held in the data cube and is computed by dividing the total number of actual values in the cube by the total number of cells in the cube. Because the data cube's dimensions are predefined, not all cells are populated. In other words, some cells are empty. Returning to the sales example, there may be many products

FIGURE 13.12 MOLAP client/server architecture




that are not sold during a given time period in a given location. In fact, you will often find that fewer than 50 percent of the data cube's cells are populated. In any case, multidimensional databases must handle sparsity effectively to reduce processing overhead and resource requirements.

Relational proponents also argue that using proprietary solutions makes it difficult to integrate the MDBMS with other data sources and tools used within the enterprise. Although it takes a substantial investment of time and effort to integrate the new technology and the existing information systems architecture, MOLAP may be a good solution for those situations in which small- to medium-sized databases are the norm and application software speed is critical.

### 13.6.8 Relational vs. Multidimensional OLAP

Table 13.10 summarizes some OLAP and MOLAP pros and cons. Keep in mind, too, that the selection of one or the other often depends on the evaluator's vantage point. For example, a proper evaluation of OLAP must include price, supported hardware platforms, compatibility with the existing DBMS, programming requirements, performance, and availability of administrative tools. The summary in Table 13.10 provides a useful starting point for comparison.

**TABLE 13.10 Relational vs. Multidimensional OLAP**

| CHARACTERISTIC | ROLAP | MOLAP |
|---|---|---|
| Schema | Uses star schema<br>Additional dimensions can be added dynamically | Uses data cubes<br>Additional dimensions require re-creation of the data cube |
| Database size | Medium to large | Small to medium |
| Architecture | Client/server<br>Standards-based<br>Open | Client/server<br>Proprietary |
| Access | Supports ad hoc requests<br>Unlimited dimensions | Limited to predefined dimensions |
| Resources | High | Very high |
| Flexibility | High | Low |
| Scalability | High | Low |
| Speed | Good with small data sets; average for medium to large data sets | Faster for small to medium data sets; average for large data sets |

ROLAP and MOLAP vendors are working toward the integration of their respective solutions within a unified decision support framework. Many OLAP products are able to handle tabular and multidimensional data with the same ease. For example, if you are using Excel OLAP functionality, as shown earlier in Figure 13.6, you can access relational OLAP data in a SQL server as well as cube (multidimensional data) in the local computer. In the meantime, relational databases successfully use the star schema design to handle multidimensional data, and their market share makes it unlikely that their popularity will fade anytime soon.

## 13.7 STAR SCHEMAS

The **star schema** is a data modeling technique used to map multidimensional decision support data into a relational database. In effect, the star schema creates the near equivalent of a multidimensional database schema from the existing relational database. The star schema was developed because existing relational modeling techniques, ER, and normalization did not yield a database structure that served advanced data analysis requirements well.

Star schemas yield an easily implemented model for multidimensional data analysis while still preserving the relational structures on which the operational database is built. The basic star schema has four components: facts, dimensions, attributes, and attribute hierarchies.

### 13.7.1 Facts

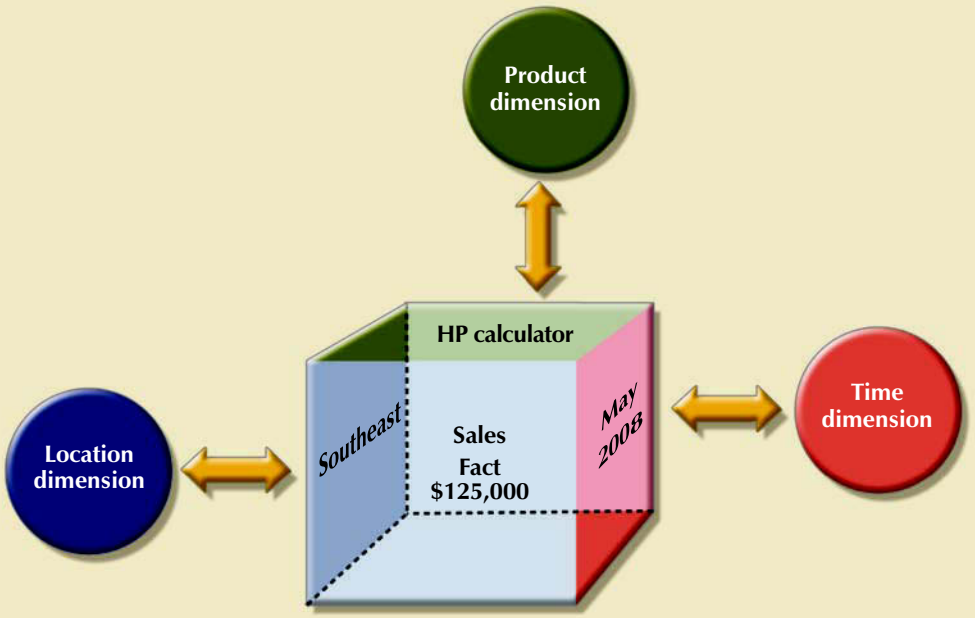
**Facts** are numeric measurements (values) that represent a specific business aspect or activity. For example, sales figures are numeric measurements that represent product and/or service sales. Facts commonly used in business data analysis are units, costs, prices, and revenues. Facts are normally stored in a fact table that is the center of the star schema. The **fact table** contains facts that are linked through their dimensions, which are explained in the next section.

Facts can also be computed or derived at run time. Such computed or derived facts are sometimes called **metrics** to differentiate them from stored facts. The fact table is updated periodically (daily, weekly, monthly, and so on) with data from operational databases.

### 13.7.2 Dimensions

**Dimensions** are qualifying characteristics that provide additional perspectives to a given fact. Recall that dimensions are of interest because *decision support data are almost always viewed in relation to other data*. For instance, sales might be compared by product from region to region and from one time period to the next. The kind of problem typically addressed by a BI system might be to make a comparison of the sales of unit X by region for the first quarters of 1998 through 2007. In that example, sales have product, location, and time dimensions. In effect, dimensions are the magnifying glass through which you study the facts. Such dimensions are normally stored in **dimension tables**. Figure 13.13 depicts a star schema for sales with product, location, and time dimensions.

**FIGURE 13.13 Simple star schema**




### 13.7.3 Attributes

Each dimension table contains attributes. Attributes are often used to search, filter, or classify facts. *Dimensions provide descriptive characteristics about the facts through their attributes*. Therefore, the data warehouse designer must define common business attributes that will be used by the data analyst to narrow a search, group information, or describe dimensions. Using a sales example, some possible attributes for each dimension are illustrated in Table 13.11.

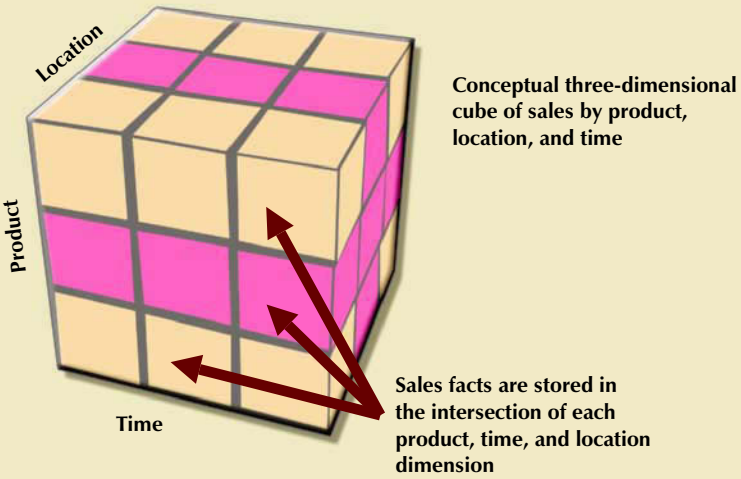
**TABLE 13.11 Possible Attributes for Sales Dimensions**

| DIMENSION NAME | DESCRIPTION | POSSIBLE ATTRIBUTES |
|---|---|---|
| Location | Anything that provides a description of the location. For example, Nashville, Store 101, South Region, and TN | Region, state, city, store, and so on |
| Product | Anything that provides a description of the product sold. For example, hair care product, shampoo, Natural Essence brand, 5.5-oz. bottle, and blue liquid | Product type, product ID, brand, package, presentation, color, size, and so on |
| Time | Anything that provides a time frame for the sales fact. For example, the year 2008, the month of July, the date 07/29/2008, and the time 4:46 p.m. | Year, quarter, month, week, day, time of day, and so on |

These product, location, and time dimensions add a business perspective to the sales facts. The data analyst can now group the sales figures for a given product, in a given region, and at a given time. The star schema, through its facts and dimensions, can provide the data in the required format when the data are needed. And it can do so without imposing the burden of the additional and unnecessary data (such as order number, purchase order number, and status) that commonly exist in operational databases.

Conceptually, the sales example's multidimensional data model is best represented by a three-dimensional cube. Of course, this does not imply that there is a limit on the number of dimensions that can be associated to a fact table. There is no mathematical limit to the number of dimensions used. However, using a three-dimensional model makes it easy to visualize the problem. In this three-dimensional example, the multidimensional data analysis terminology, the cube illustrated in Figure 13.14 represents a view of sales dimensioned by product, location, and time.

**FIGURE 13.14** **Three-dimensional view of sales**



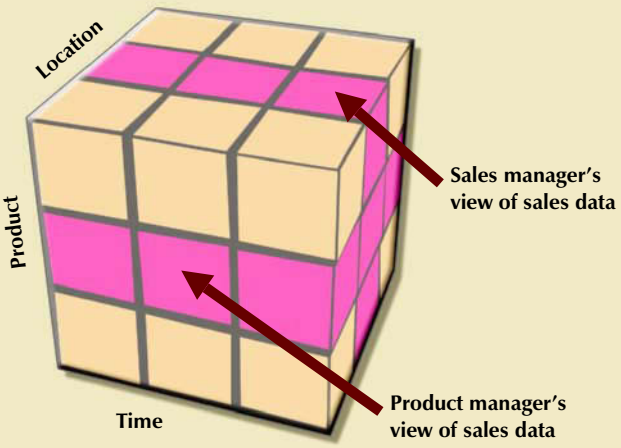

Note that each sales value stored in the cube in Figure 13.14 is associated with the location, product, and time dimensions. However, keep in mind that this cube is only a *conceptual* representation of multidimensional data, and it does not show how the data are physically stored in a data warehouse. A ROLAP engine stores data in an RDBMS and uses its own data analysis logic and the end-user GUI to perform multidimensional analysis. A MOLAP system stores data in an MDBMS, using proprietary matrix and array technology to simulate this multidimensional cube.

Whatever the underlying database technology, one of the main features of multidimensional analysis is its ability to focus on specific "slices" of the cube. For example, the product manager may be interested in examining the sales of a product while the store manager is interested in examining the sales made by a particular store. In multidimensional terms, the ability to focus on slices of the cube to perform a more detailed analysis is known as **slice and dice**. Figure 13.15 illustrates the slice-and-dice concept. As you look at Figure 13.15, note that each cut across the cube yields a slice. Intersecting slices produce small cubes that constitute the "dice" part of the "slice-and-dice" operation.

To slice and dice, it must be possible to identify each slice of the cube. That is done by using the values of each attribute in a given dimension. For example, to use the location dimension, you might need to define a STORE_ID attribute in order to focus on a particular store.

Given the requirement for attribute values in a slice-and-dice environment, let's reexamine Table 13.11. Note that each attribute adds an additional perspective to the sales facts, thus setting the stage for finding new ways to search, classify, and possibly aggregate information. For example, the location dimension adds a geographic perspective of where the

**FIGURE 13.15** Slice-and-dice view of sales

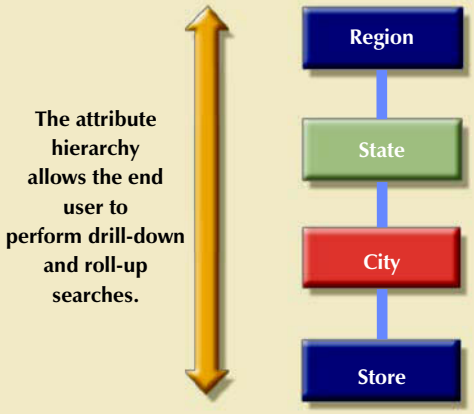



sales took place: in which region, state, city, store, and so on. All of the attributes are selected with the objective of providing decision support data to the end users so that they can study sales by each of the dimension's attributes.

Time is an especially important dimension. The time dimension provides a framework from which sales patterns can be analyzed and possibly predicted. Also, the time dimension plays an important role when the data analyst is interested in looking at sales aggregates by quarter, month, week, and so on. Given the importance and universality of the time dimension from a data analysis perspective, many vendors have added automatic time dimension management features to their data warehousing products.

### 13.7.4 Attribute Hierarchies

**FIGURE 13.16** Location attribute hierarchy



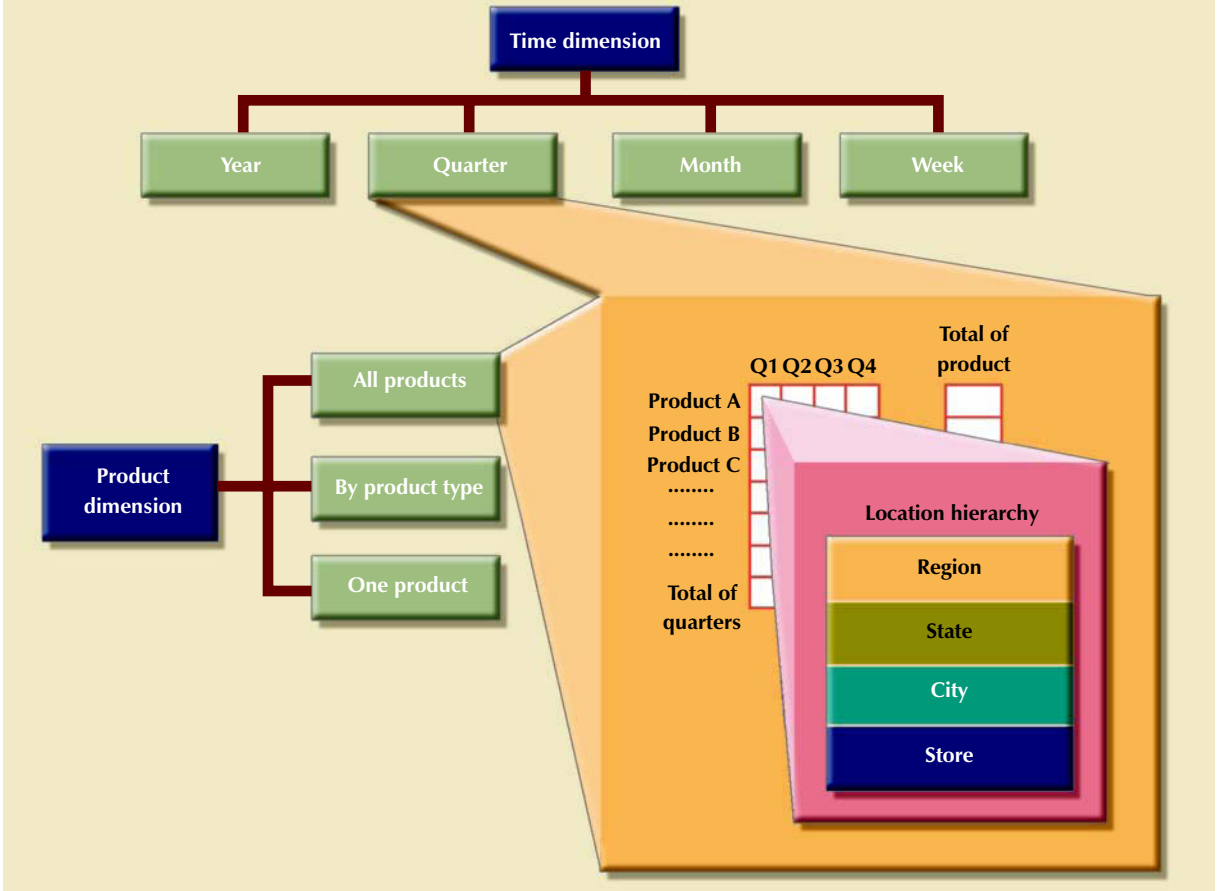

Attributes within dimensions can be ordered in a well-defined attribute hierarchy. The **attribute hierarchy** provides a top-down data organization that is used for two main purposes: aggregation and drill-down/roll-up data analysis. For example, Figure 13.16 shows how the location dimension attributes can be organized in a hierarchy by region, state, city, and store.

The attribute hierarchy provides the capability to perform drill-down and roll-up searches in a data warehouse. For example, suppose a data analyst looks at the answers to the query, How does the 2007 month-to-date sales performance compare to the 2008 month-to-date sales performance? The data analyst spots a sharp sales decline for March 2008. The data analyst might decide to drill down inside the month of March to see how sales by regions compared to the previous year. By doing that, the analyst can determine whether the low March sales were reflected in all regions or in only a particular region. This type of drill-down operation can even be extended until the data analyst identifies the store that is performing below the norm.

The March sales scenario is possible because the attribute hierarchy allows the data warehouse and OLAP systems to have a defined path that will identify how data are to be decomposed and aggregated for drill-down and roll-up operations. It is not necessary for all attributes to be part of an attribute hierarchy; some attributes exist merely to provide narrative descriptions of the dimensions. But keep in mind that the attributes from different dimensions can be grouped to form a hierarchy. For example, after you drill down from city to store, you might want to drill down using the product dimension so the manager can identify slow products in the store. The product dimension can be based on the product group (dairy, meat, and so on) or on the product brand (Brand A, Brand B, and so on).

Figure 13.17 illustrates a scenario in which the data analyst studies sales facts, using the product, time, and location dimensions. In this example, the product dimension is set to "All products," meaning that the data analyst will see all products on the y-axis. The time dimension (x-axis) is set to "Quarter," meaning that the data are aggregated by quarters (for example, total sales of products A, B, and C in Q1, Q2, Q3, and Q4). Finally, the location dimension is initially set to "Region," thus ensuring that each cell contains the total regional sales for a given product in a given quarter.

**FIGURE 13.17** Attribute hierarchies in multidimensional analysis




The simple data analysis scenario illustrated in Figure 13.17 provides the data analyst with three different information paths. On the product dimension (the y-axis), the data analyst can request to see all products, products grouped by type, or just one product. On the time dimension (the x-axis), the data analyst can request time-variant data at different levels of aggregation: year, quarter, month, or week. Each sales value initially shows the total sales, by region, of each product. When a GUI is used, clicking on the region cell enables the data analyst to drill down to see sales by states within the region. Clicking again on one of the state values yields the sales for each city in the state, and so forth.

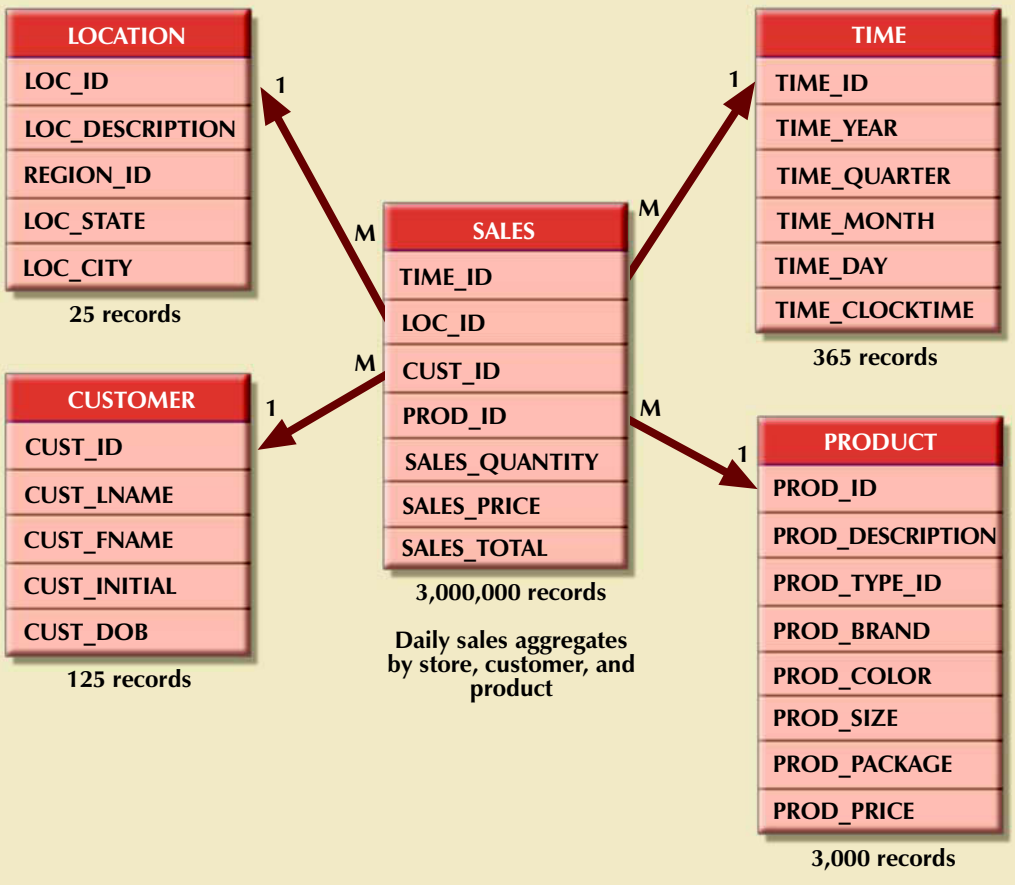
As the preceding examples illustrate, attribute hierarchies determine how the data in the data warehouse are extracted and presented. The attribute hierarchy information is stored in the DBMS's data dictionary and is used by the OLAP tool to access the data warehouse properly. Once such access is ensured, query tools must be closely integrated with the data warehouse's metadata and they must support powerful analytical capabilities.

### 13.7.5 Star Schema Representation

Facts and dimensions are normally represented by physical tables in the data warehouse database. The fact table is related to each dimension table in a many-to-one (M:1) relationship. In other words, many fact rows are related to each dimension row. Using the sales example, you can conclude that each product appears many times in the SALES fact table.

Fact and dimension tables are related by foreign keys and are subject to the familiar primary key/foreign key constraints. The primary key on the "1" side, the dimension table, is stored as part of the primary key on the "many" side, the fact table. *Because the fact table is related to many dimension tables, the primary key of the fact table is a composite primary key*. Figure 13.18 illustrates the relationships among the sales fact table and the product, location, and time dimension tables. To show you how easily the star schema can be expanded, a customer dimension has been added to the mix. Adding the customer dimension merely required including the CUST_ID in the SALES fact table and adding the CUSTOMER table to the database.

**FIGURE 13.18** Star schema for SALES



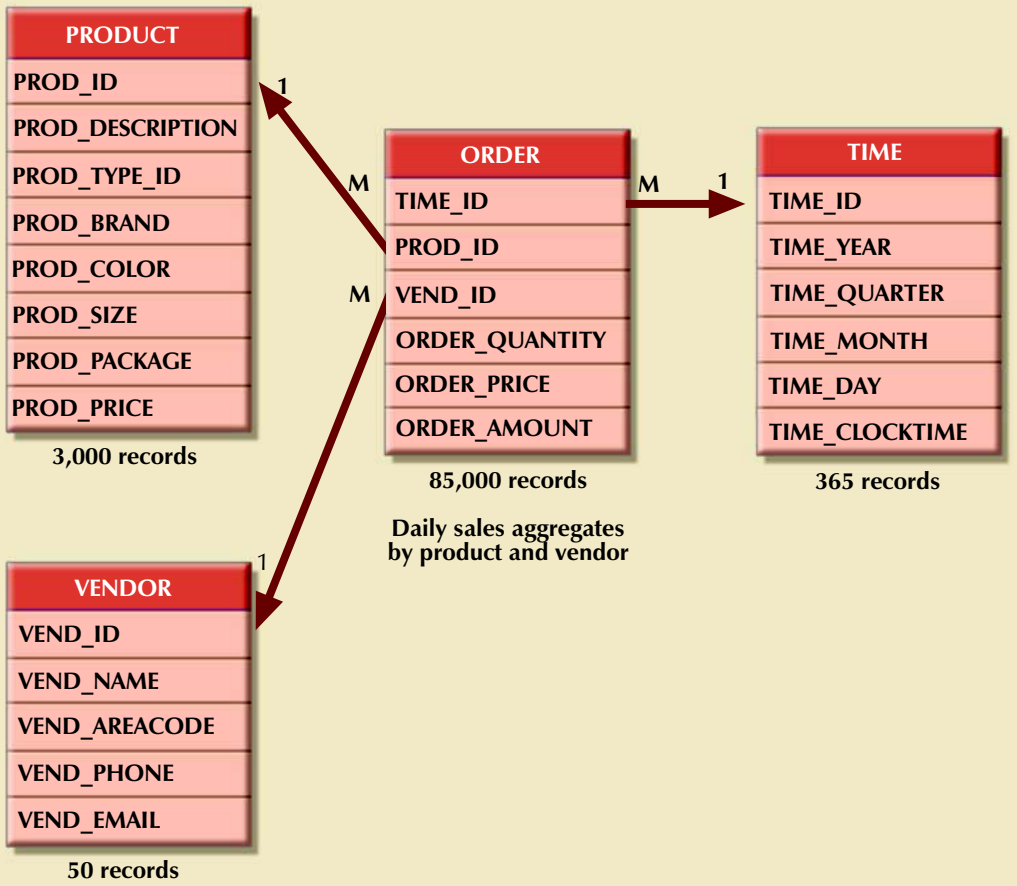
The composite primary key for the SALES fact table is composed of TIME_ID, LOC_ID, CUST_ID, and PROD_ID. Each record in the SALES fact table is uniquely identified by the combination of values for each of the fact table's foreign keys. *By default, the fact table's primary key is always formed by combining the foreign keys pointing to the dimension tables to which they are related*. In this case, each sales record represents each product sold to a specific customer, at a specific time, and in a specific location. In this schema, the TIME dimension table represents daily periods, so the SALES fact table represents daily sales aggregates by product and by customer. Because fact tables contain the actual values used in the decision support process, those values are repeated many times in the fact tables. Therefore, the fact tables are always the largest tables in the star schema. Because the dimension tables contain only nonrepetitive information (all unique salespersons, all unique products, and so on), the dimension tables are always smaller than the fact tables.

In a typical star schema, each dimension record is related to thousands of fact records. For example, "widget" appears only once in the product dimension, but it has thousands of corresponding records in the SALES fact table. That characteristic of the star schema facilitates data retrieval functions because most of the time the data analyst will look at the facts through the dimension's attributes. Therefore, a data warehouse DBMS that is optimized for decision support first searches the smaller dimension tables before accessing the larger fact tables.

Data warehouses usually have many fact tables. Each fact table is designed to answer specific decision support questions. For example, suppose you develop a new interest in orders while maintaining your original interest in sales. In that scenario, you should maintain an ORDERS fact table and a SALES fact table in the same data warehouse. If orders are considered to be an organization's key interest, the ORDERS fact table should be the center of a star schema that might have vendor, product, and time dimensions. In that case, an interest in vendors yields a new vendor dimension, represented by a new VENDOR table in the database. The product dimension is represented by the same product table used in the initial sales star schema. However, given the interest in orders as well as sales, the time dimension now requires special attention. If the orders department uses the same time periods as the sales department, time can be represented by the same time table. If different time periods are used, you must create another table, perhaps named ORDER_TIME, to represent the time periods used by the orders department. In Figure 13.19, the orders star schema shares the product, vendor, and time dimensions.

Multiple fact tables also can be created for performance and semantic reasons. The following section explains several performance-enhancing techniques that can be used within the star schema.

FIGURE 13.19 Orders star schema




### 13.7.6 Performance-Improving Techniques for the Star Schema

The creation of a database that provides fast and accurate answers to data analysis queries is the data warehouse design's prime objective. Therefore, performance-enhancement actions might target query speed through the facilitation of SQL code as well as through better semantic representation of business dimensions. Four techniques are often used to optimize data warehouse design:

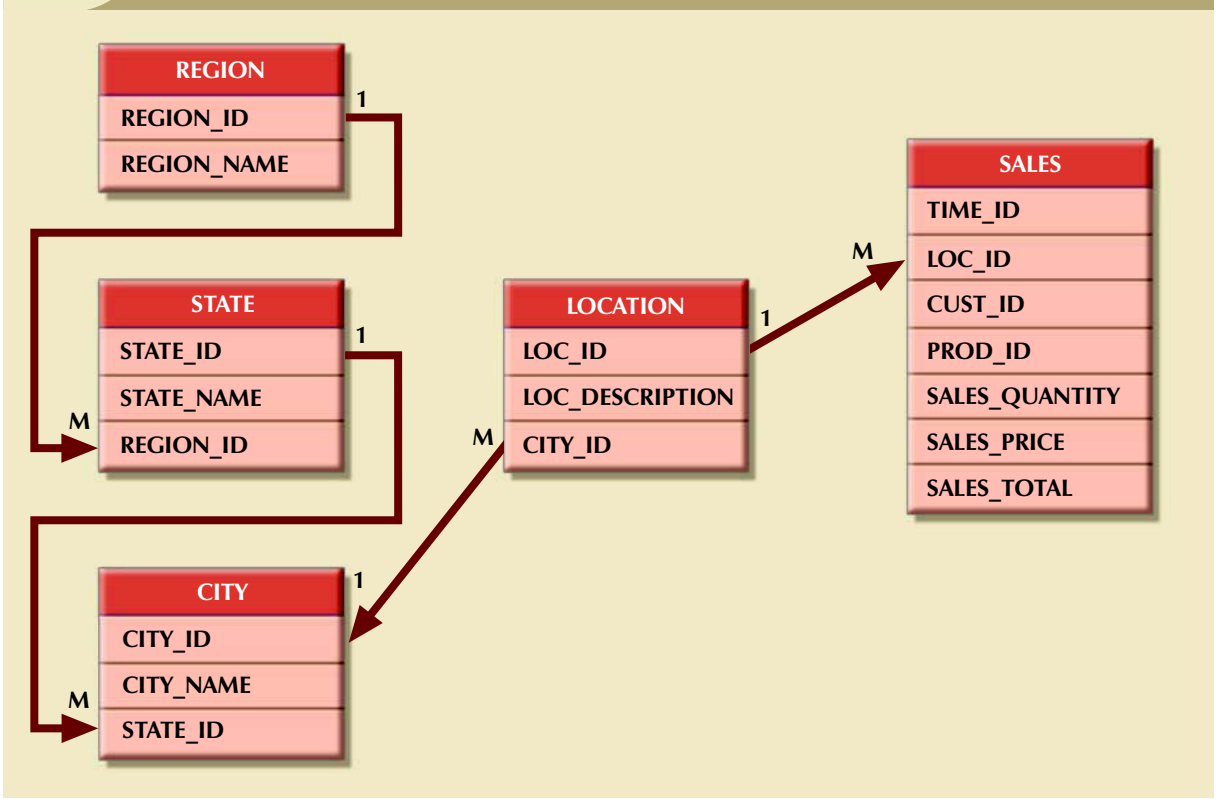
- Normalizing dimensional tables.
- Maintaining multiple fact tables to represent different aggregation levels.
- Denormalizing fact tables.
- Partitioning and replicating tables.

## Normalizing Dimensional Tables

Dimensional tables are normalized to achieve semantic simplicity and facilitate end-user navigation through the dimensions. For example, if the location dimension table contains transitive dependencies among region, state, and city, you can revise those relationships to the 3NF (third normal form), as shown in Figure 13.20. (If necessary, review

normalization techniques in Chapter 5, Normalization of Database Tables.) The star schema shown in Figure 13.20 is known as a **snowflake schema**, which is a type of star schema in which the dimension tables can have their own dimension tables. The snowflake schema is usually the result of normalizing dimension tables.

**FIGURE 13.20** Normalized dimension tables




By normalizing the dimension tables, you simplify the data-filtering operations related to the dimensions. In this example, the region, state, city, and location contain very few records compared to the SALES fact table. Only the location table is directly related to the sales fact table.

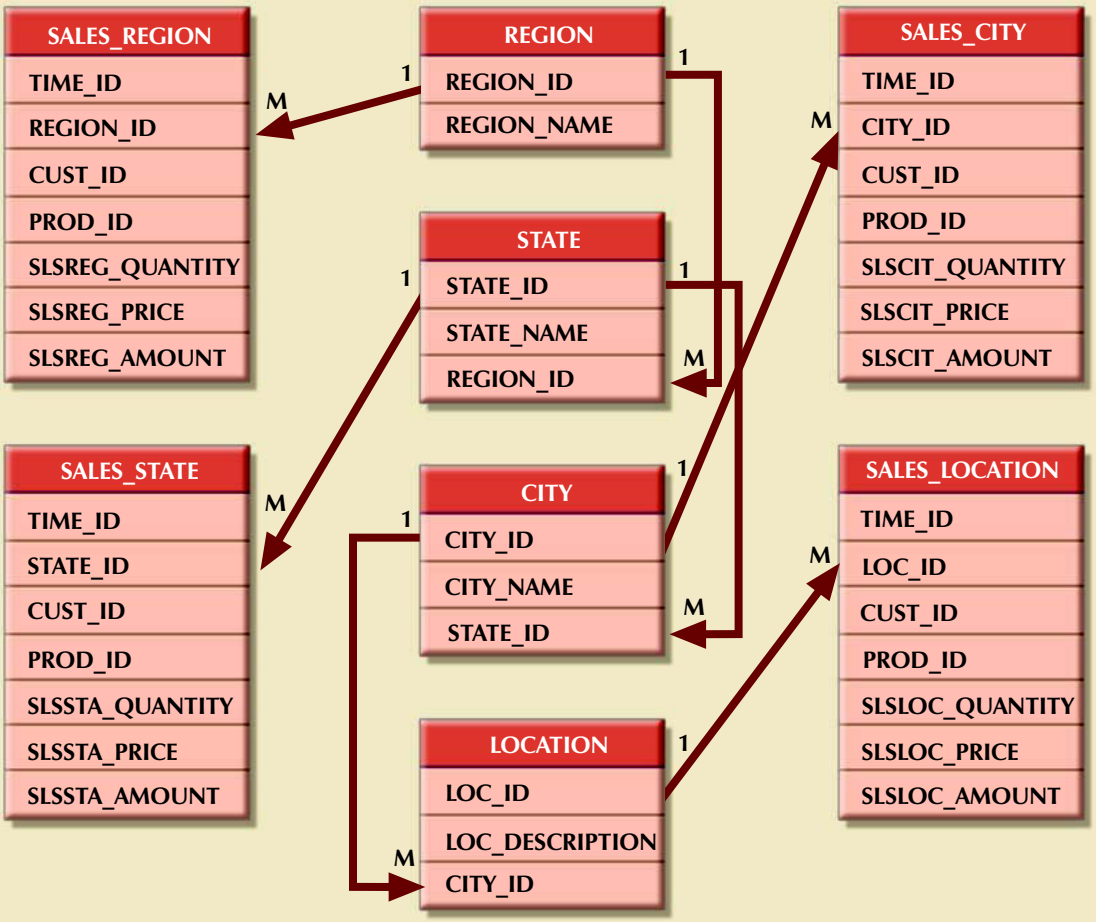
**NOTE**

Although using the dimension tables shown in Figure 13.20 gains structural simplicity, there is a price to pay for that simplicity. For example, if you want to aggregate the data by region, you must use a four-table join, thus increasing the complexity of the SQL statements. The star schema in Figure 13.18 uses a LOCATION dimension table that greatly facilitates data retrieval by eliminating multiple join operations. This is yet another example of the trade-offs that designers must consider.

## Maintaining Multiple Fact Tables that Represent Different Aggregation Levels

You can also speed up query operations by creating and maintaining multiple fact tables related to each level of aggregation (region, state, and city) in the location dimension. These aggregate tables are precomputed at the data-loading phase rather than at run time. The purpose of this technique is to save processor cycles at run time, thereby speeding up data analysis. An end-user query tool optimized for decision analysis then properly accesses the summarized fact tables instead of computing the values by accessing a lower level of detail fact table. This technique is illustrated in Figure 13.21, which adds aggregate fact tables for region, state, and city to the initial sales example.

FIGURE 13.21 Multiple fact tables




The data warehouse designer must identify which levels of aggregation to precompute and store in the database. These multiple aggregate fact tables are updated during each load cycle in batch mode. And because the objective is to minimize access and processing time, according to the expected frequency of use and the processing time required to calculate a given aggregation level at run time, the data warehouse designer must select which aggregation fact tables to create.

## Denormalizing Fact Tables

Denormalizing fact tables improves data access performance and saves data storage space. The latter objective, however, is becoming less of an issue. Data storage costs decrease almost daily, and DBMS limitations that restrict database and table size limits, record size limits, and the maximum number of records in a single table have far more negative effects than raw storage space costs.

Denormalization improves performance by using a single record to store data that normally take many records. For example, to compute the total sales for all products in all regions, you might have to access the region sales aggregates and summarize all of the records in this table. If you have 300,000 product sales, you could be summarizing at least 300,000 rows. Although this might not be a very taxing operation for a DBMS, a comparison of, say, 10 years' worth of previous sales begins to bog down the system. In such cases, it is useful to have special aggregate tables that are

denormalized. For example, a YEAR_TOTALS table might contain the following fields: YEAR_ID, MONTH_1, MONTH_2 ... MONTH_12, and each year's total. Such tables can easily be used to serve as a basis for year-to-year comparisons at the top month level, the quarter level, or the year level. Here again, design criteria, such as frequency of use and performance requirements, are evaluated against the possible overload placed on the DBMS to manage the denormalized relations.

### Partitioning and Replicating Tables

Because table partitioning and replication were covered in detail in Chapter 12, Distributed Database Management Systems, those techniques are discussed here only as they specifically relate to the data warehouse. Table partitioning and replication are particularly important when a BI system is implemented in dispersed geographic areas. **Partitioning** splits a table into subsets of rows or columns and places the subsets close to the client computer to improve data access time. **Replication** makes a copy of a table and places it in a different location, also to improve access time.

No matter which performance-enhancement scheme is used, time is the most common dimension used in business data analysis. Therefore, it is very common to have one fact table for each level of aggregation defined within the time dimension. For example, in the sales example, you might have five aggregate sales fact tables: daily, weekly, monthly, quarterly, and yearly. Those fact tables must have an implicit or explicit periodicity defined. **Periodicity**, usually expressed as current year only, previous years, or all years, provides information about the time span of the data stored in the table.

At the end of each year, daily sales for the current year are moved to another table that contains previous years' daily sales only. This table actually contains all sales records from the beginning of operations, with the exception of the current year. The data in the current year and previous years' tables thus represent the complete sales history of the company. The previous years' sales table can be replicated at several locations to avoid remote access to the historic sales data, which can cause slow response time. The possible size of this table is enough to intimidate all but the bravest of query optimizers. Here is one case in which denormalization would be of value!

## 13.8 IMPLEMENTING A DATA WAREHOUSE

Organization-wide information system development is subject to many constraints. Some of the constraints are based on available funding. Others are a function of management's view of the role played by an IS department and of the extent and depth of the information requirements. Add the constraints imposed by corporate culture, and you understand why no single formula can describe perfect data warehouse development. Therefore, rather than proposing a single data warehouse design and implementation methodology, this section identifies a few factors that appear to be common to data warehousing.

### 13.8.1 The Data Warehouse as an Active Decision Support Framework

Perhaps the first thing to remember is that a data warehouse is not a static database. Instead, it is a dynamic framework for decision support that is, almost by definition, always a work in progress. Because it is the foundation of a modern BI environment, the design and implementation of the data warehouse means that you are involved in the design and implementation of a complete database-system-development infrastructure for company-wide decision support. Although it is easy to focus on the data warehouse database as the BI central data repository, you must remember that the decision support infrastructure includes hardware, software, people, and procedures, as well as data. The argument that the data warehouse is the only *critical* BI success component is as misleading as the argument that a human being needs only a heart or a brain to function. The data warehouse is a critical component of a modern BI environment, but it is certainly not the only critical component. Therefore, its design and implementation must be examined in light of the entire infrastructure.

### 13.8.2 A Company-Wide Effort That Requires User Involvement

Designing a data warehouse means being given an opportunity to help develop an integrated data model that captures the data that are considered to be essential to the organization, from both end-user and business perspectives. Data warehouse data cross departmental lines and geographical boundaries. Because the data warehouse represents an attempt to model all of the organization's data, you are likely to discover that organizational components (divisions, departments, support groups, and so on) often have conflicting goals, and it certainly will be easy to find data inconsistencies and damaging redundancies. Information is power, and the control of its sources and uses is likely to trigger turf battles, end-user resistance, and power struggles at all levels. Building the perfect data warehouse is not just a matter of knowing how to create a star schema; it requires managerial skills to deal with conflict resolution, mediation, and arbitration. In short, the designer must:

- Involve end users in the process.
- Secure end users' commitment from the beginning.
- Solicit continuous end-user feedback.
- Manage end-user expectations.
- Establish procedures for conflict resolution.

### 13.8.3 Satisfy the Trilogy: Data, Analysis, and Users

Great managerial skills are not, of course, solely sufficient. The technical aspects of the data warehouse must be addressed as well. The old adage of input-process-output repeats itself here. The data warehouse designer must satisfy:

- Data integration and loading criteria.
- Data analysis capabilities with acceptable query performance.
- End-user data analysis needs.

The foremost technical concern in implementing a data warehouse is to provide end-user decision support with advanced data analysis capabilities—at the right moment, in the right format, with the right data, and at the right cost.

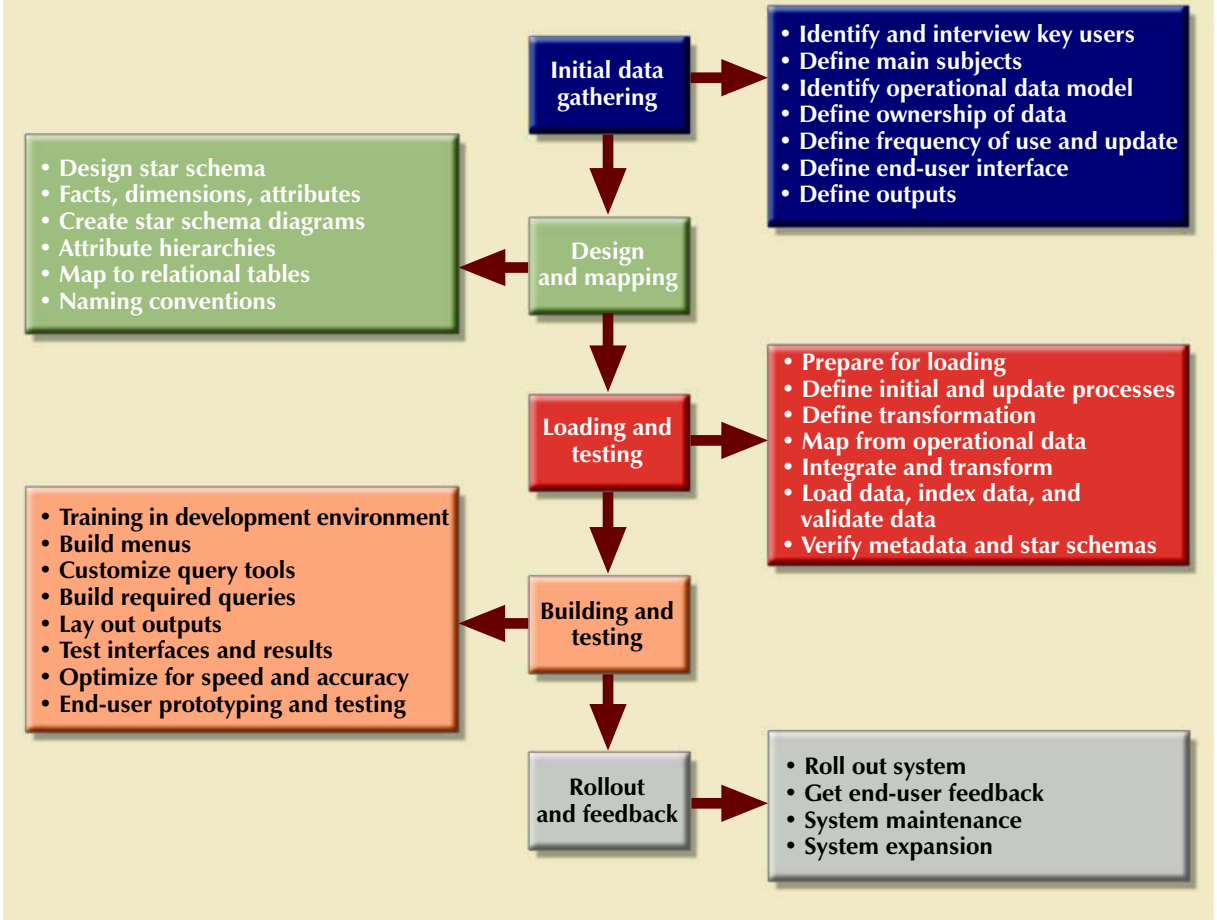
### 13.8.4 Apply Database Design Procedures

You learned about the database life cycle and the database design process in Chapter 9, Database Design, so perhaps it is wise to review the traditional database design procedures. These design procedures must then be adapted to fit the data warehouse requirements. If you remember that the data warehouse derives its data from operational databases, you will understand why a solid foundation in operational database design is important. (It's difficult to produce good data warehouse data when the operational database data are corrupted.) Figure 13.22 depicts a simplified process for implementing the data warehouse.

As noted, developing a data warehouse is a company-wide effort that requires many resources: human, financial, and technical. Providing company-wide decision support requires a sound architecture based on a mix of people skills, technology, and managerial procedures that is often difficult to find and implement. For example:

- The sheer and often mind-boggling quantity of decision support data is likely to require the latest hardware and software—that is, advanced computers with multiple processors, advanced database systems, and large-capacity storage units. In the not-too-distant past, those requirements usually prompted the use of a mainframe-based system. Today's client/server technology offers many other choices to implement a data warehouse.
- Very detailed procedures are necessary to orchestrate the flow of data from the operational databases to the data warehouse. Data flow control includes data extraction, validation, and integration.
- To implement and support the data warehouse architecture, you also need people with advanced database design, software integration, and management skills.

**FIGURE 13.22** Data warehouse design and implementation road map




## 13.9 DATA MINING

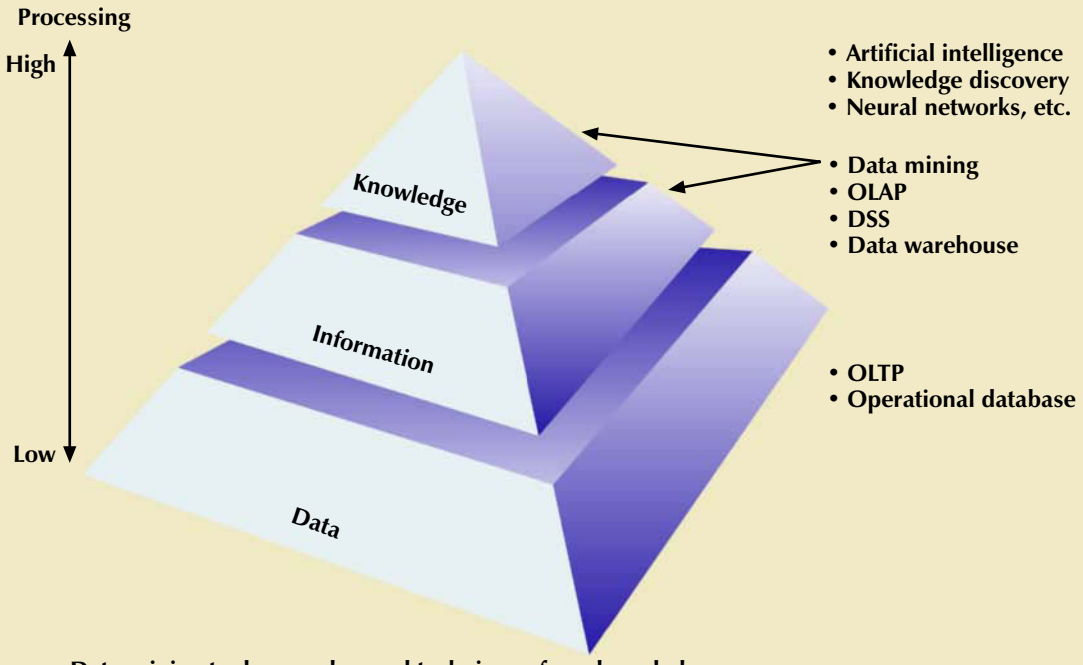
The purpose of data analysis is to discover previously unknown data characteristics, relationships, dependencies, or trends. Such discoveries then become part of the information framework on which decisions are built. *A typical data analysis tool relies on the end users to define the problem, select the data, and initiate the appropriate data analyses to generate the information that helps model and solve problems that the end users uncover*. In other words, the end user reacts to an external stimulus—the discovery of the problem itself. If the end user fails to detect a problem, no action is taken. Given that limitation, some current BI environments now support various types of automated alerts. The alerts are software agents that constantly monitor certain parameters, such as sales indicators and inventory levels, and then perform specified actions (send e-mail or alert messages, run programs, and so on) when such parameters reach predefined levels.

In contrast to the traditional (reactive) BI tools, data mining is *proactive*. Instead of having the end user define the problem, select the data, and select the tools to analyze the data, *data-mining tools automatically search the data for anomalies and possible relationships, thereby identifying problems that have not yet been identified by the end user*. In other words, **data mining** refers to the activities that analyze the data, uncover problems or opportunities hidden in the data relationships, form computer models based on their findings, and then use the models to predict business behavior—requiring minimal end-user intervention. Therefore, the end user is able to use the system's findings

to gain knowledge that might yield competitive advantages. Data mining describes a new breed of specialized decision support tools that automate data analysis. In short, data-mining tools *initiate* analyses to create knowledge. Such knowledge can be used to address any number of business problems. For example, banks and credit card companies use knowledge-based analysis to detect fraud, thereby decreasing fraudulent transactions.

To put data mining in perspective, look at the pyramid in Figure 13.23, which represents how knowledge is extracted from data. *Data* form the pyramid base and represent what most organizations collect in their operational databases. The second level contains *information* that represents the purified and processed data. Information forms the basis for decision making and business understanding. *Knowledge* is found at the pyramid's apex and represents highly specialized information.

**FIGURE 13.23 Extracting knowledge from data**




**Data-mining tools use advanced techniques from knowledge discovery, artificial intelligence, and other fields to obtain "knowledge" and apply it to business needs. Knowledge is then used to make predictions of events or forecasts of values such as sales returns. Several OLAP tools have integrated at least some of these data-mining features in their products.**

It is difficult to provide a precise list of characteristics of data-mining tools. For one thing, the current generation of data-mining tools contains many design and application variations to fit data-mining requirements. Additionally, the many variations exist because there are no established standards that govern the creation of data-mining tools. Each data-mining tool seems to be governed by a different approach and focus, thus generating families of data-mining tools that focus on market niches such as marketing, retailing, finance, healthcare, investments, insurance, and banking. Within a given niche, data-mining tools can use certain algorithms, and those algorithms can be implemented in different ways and/or applied over different data.

In spite of the lack of precise standards, data mining is subject to four general phases:

1. Data preparation.
2. Data analysis and classification.
3. Knowledge acquisition.
4. Prognosis.

In the *data preparation phase*, the main data sets to be used by the data mining operation are identified and cleansed of any data impurities. Because the data in the data warehouse are already integrated and filtered, the data warehouse usually is the target set for data mining operations.

The *data analysis and classification phase* studies the data to identify common data characteristics or patterns. During this phase, the data-mining tool applies specific algorithms to find:

- Data groupings, classifications, clusters, or sequences.
- Data dependencies, links, or relationships.
- Data patterns, trends, and deviations.

The *knowledge acquisition phase* uses the results of the data analysis and classification phase. During the knowledge acquisition phase, the data-mining tool (with possible intervention by the end user) selects the appropriate modeling or knowledge acquisition algorithms. The most common algorithms used in data mining are based on neural networks, decision trees, rules induction, genetic algorithms, classification and regression trees, memory-based reasoning, and nearest neighbor and data visualization. A data-mining tool may use many of these algorithms in any combination to generate a computer model that reflects the behavior of the target data set.

Although many data-mining tools stop at the knowledge-acquisition phase, others continue to the *prognosis phase*. In that phase, the data mining findings are used to predict future behavior and forecast business outcomes. Examples of data mining findings can be:

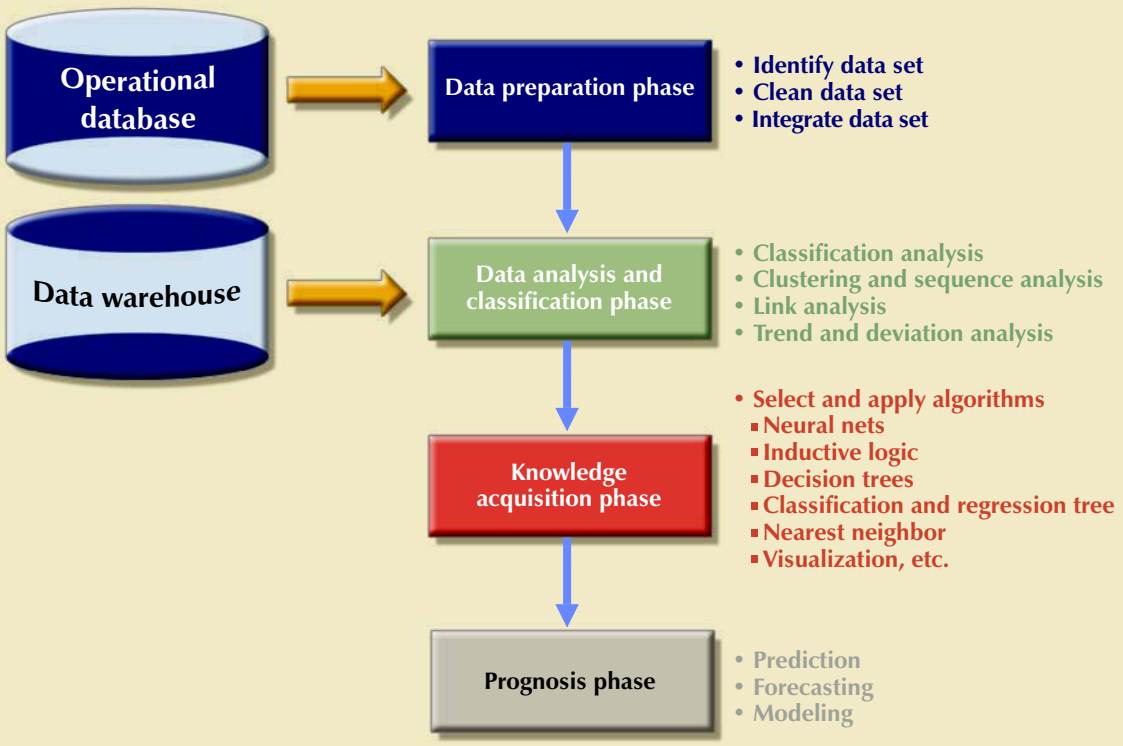
- Sixty-five percent of customers who did not use a particular credit card in the last six months are 88 percent likely to cancel that account.
- Eighty-two percent of customers who bought a 27-inch or larger TV are 90 percent likely to buy an entertainment center within the next four weeks.
- If age < 30 and income <= 25,000 and credit rating < 3 and credit amount > 25,000, then the minimum loan term is 10 years.

The complete set of findings can be represented in a decision tree, a neural net, a forecasting model, or a visual presentation interface that is used to project future events or results. For example, the prognosis phase might project the likely outcome of a new product rollout or a new marketing promotion. Figure 13.24 illustrates the different phases of the data mining techniques.

Because data mining technology is still in its infancy, some of the data mining findings might fall outside the boundaries of what business managers expect. For example, a data-mining tool might find a close relationship between a customer's favorite brand of soda and the brand of tires on the customer's car. Clearly, that relationship might not be held in high regard among sales managers. (In regression analysis, those relationships are commonly described by the label "idiot correlation.") Fortunately, data mining usually yields more meaningful results. In fact, data mining has proved to be very helpful in finding practical relationships among data that help define customer buying patterns, improve product development and acceptance, reduce healthcare fraud, analyze stock markets, and so on.

Ideally, you can expect the development of databases that not only store data and various statistics about data usage, but also have the ability to learn about and extract knowledge from the stored data. Such database management systems, also known as inductive or intelligent databases, are the focus of intense research in many laboratories. Although those databases have yet to lay claim to substantial commercial market penetration, both "add-on" and DBMS-integrated data mining tools have proliferated in the data warehousing database market.

FIGURE 13.24 Data–mining phases



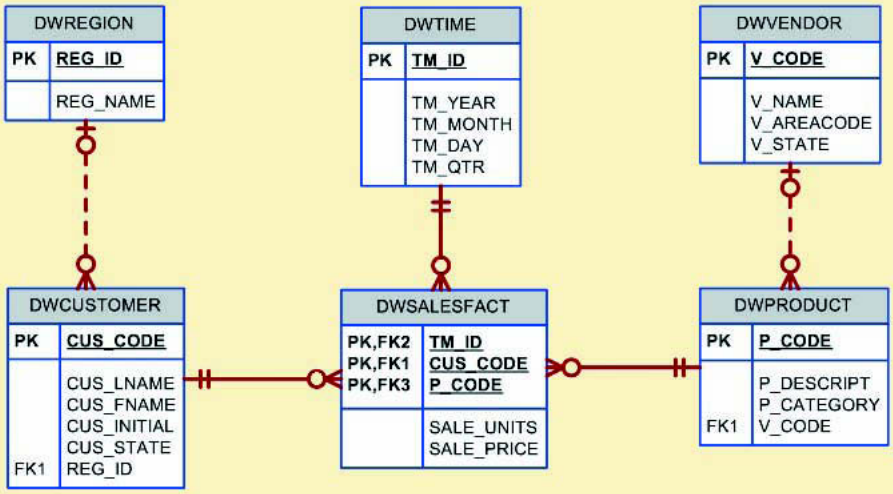

## 13.10 SQL EXTENSIONS FOR OLAP

The proliferation of OLAP tools has fostered the development of SQL extensions to support multidimensional data analysis. Most SQL innovations are the result of vendor-centric product enhancements. However, many of the innovations have made their way into standard SQL. This section introduces some of the new SQL extensions that have been created to support OLAP-type data manipulations.

The SaleCo snowflake schema shown in Figure 13.25 will be used to demonstrate the use of the SQL extensions. Note that this snowflake schema has a central DWSALESFACT fact table and three dimension tables: DWCUSTOMER, DWPRODUCT, and DWTIME. The central fact table represents daily sales by product and customer. However, as you examine the star schema shown in Figure 13.25 more carefully, you will see that the DWCUSTOMER and DWPRODUCT dimension tables have their own dimension tables: DWREGION and DWVENDOR.

Keep in mind that a database is at the core of all data warehouses. Therefore, all SQL commands (such as CREATE, INSERT, UPDATE, DELETE, and SELECT) will work in the data warehouse as expected. However, most queries you run in a data warehouse tend to include a lot of data groupings and aggregations over multiple columns. That's why this section introduces two extensions to the GROUP BY clause that are particularly useful: ROLLUP and CUBE. In addition, you will learn about using materialized views to store preaggregated rows in the database.

**FIGURE 13.25** SaleCo snowflake schema




## ONLINE CONTENT

The script files used to populate the database and run the SQL commands are available in the Student Online Companion.

**NOTE**

This section uses the Oracle RDBMS to demonstrate the use of SQL extensions to support OLAP functionality. If you use a different DBMS, consult the documentation to verify whether the vendor supports similar functionality and what the proper syntax is for your DBMS.

### 13.10.1 THE ROLLUP EXTENSION

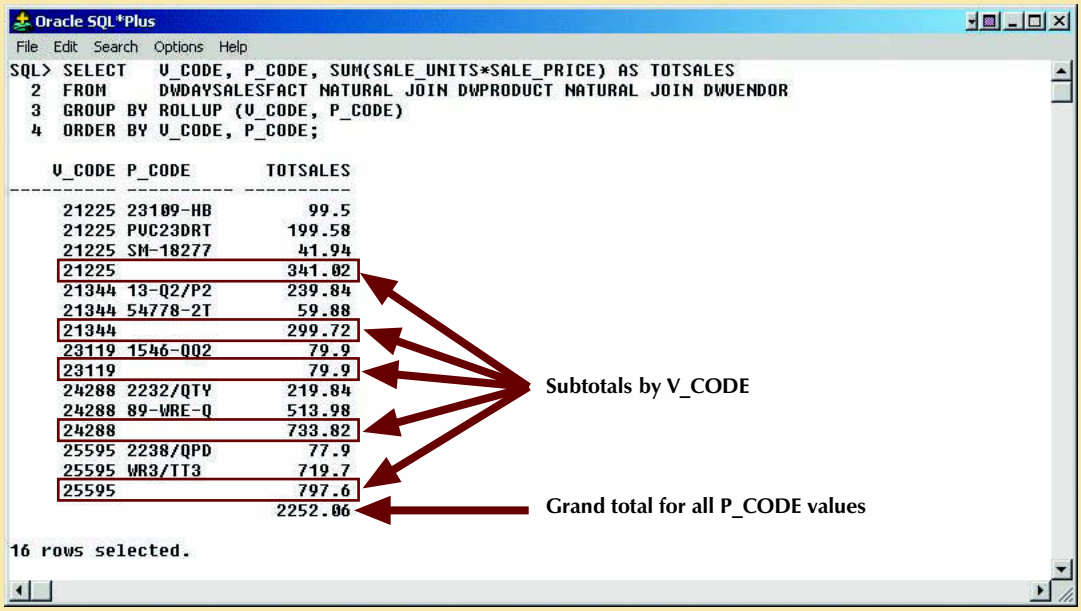
The ROLLUP extension is used with the GROUP BY clause to generate aggregates by different dimensions. As you know, the GROUP BY clause will generate only one aggregate for each new value combination of attributes listed in the GROUP BY clause. The ROLLUP extension goes one step further; it enables you to get a subtotal for each column listed except for the last one, which gets a grand total instead. The syntax of the GROUP BY ROLLUP is as follows:

```
SELECT      column1, column2 [, ...], aggregate_function(expression)
FROM        table1 [,table2, ...]
[WHERE      condition]
GROUP BY    ROLLUP (column1, column2 [, ...])
[HAVING     condition]
[ORDER BY   column1 [, column2, ...]]
```

The order of the column list within the GROUP BY ROLLUP is very important. The last column in the list will generate a grand total. All other columns will generate subtotals. For example, Figure 13.26 shows the use of the ROLLUP extension to generate subtotals by vendor and product.

Note that Figure 13.26 shows the subtotals by vendor code and a grand total for all product codes. Contrast that with the normal GROUP BY clause that will generate only the subtotals for each vendor and product combination rather than the subtotals *by vendor* and the grand total for *all products*. The ROLLUP extension is particularly useful when you want to obtain multiple nested subtotals for a dimension hierarchy. For example, within a location hierarchy, you can use ROLLUP to generate subtotals by region, state, city, and store.

**FIGURE 13.26** ROLLUP extension




## 13.10.2 The CUBE Extension

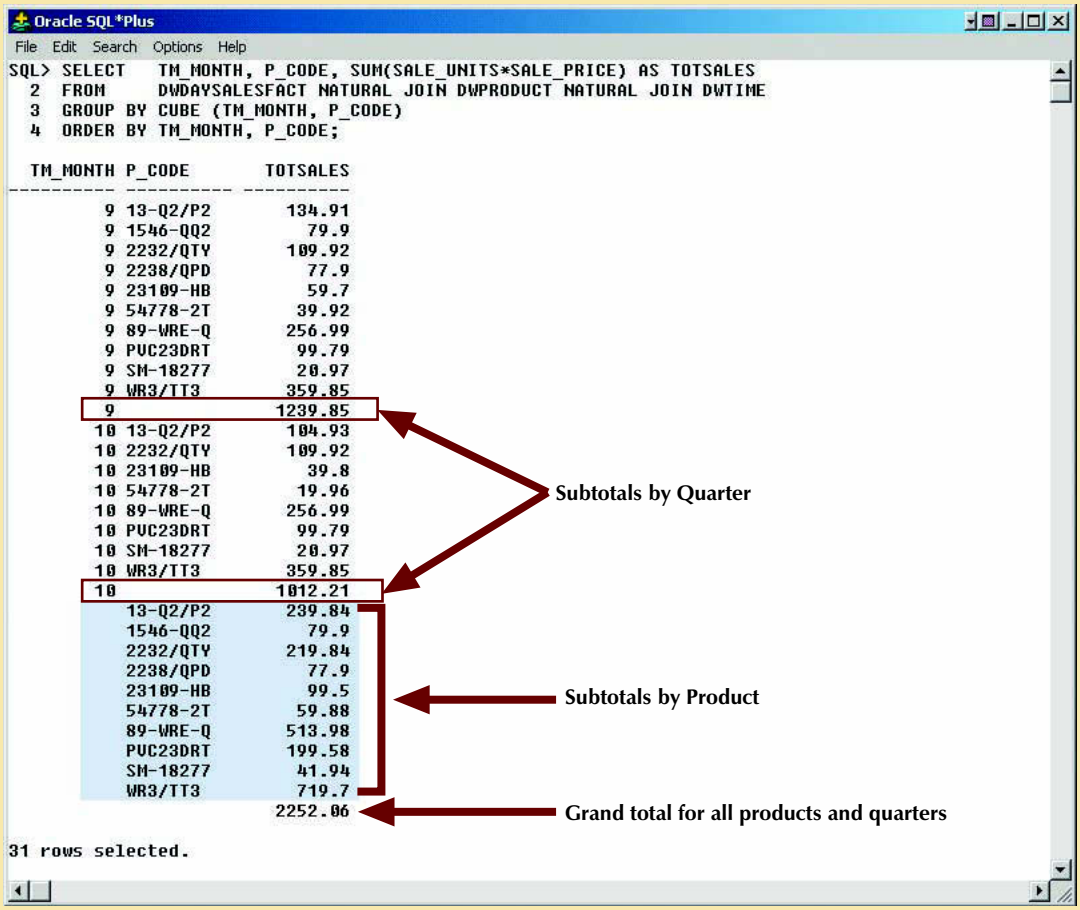
The CUBE extension is also used with the GROUP BY clause to generate aggregates by the listed columns, including the last one. The CUBE extension will enable you to get a subtotal for each column listed in the expression, in addition to a grand total for the last column listed. The syntax of the GROUP BY CUBE is as follows:

```
SELECT      column1 [, column2, ...], aggregate_function(expression)
FROM        table1 [,table2, ...]
[WHERE      condition]
GROUP BY    CUBE (column1, column2 [, ...])
[HAVING     condition]
[ORDER BY   column1 [, column2, ...]]
```

For example, Figure 13.27 shows the use of the CUBE extension to compute the sales subtotals by month and by product, as well as a grand total.

In Figure 13.27, note that the CUBE extension generates the subtotals for each combination of month and product, in addition to subtotals by month and by product, as well as a grand total. The CUBE extension is particularly useful when you want to compute all possible subtotals within groupings based on multiple dimensions. Cross-tabulations are especially good candidates for application of the CUBE extension.

**FIGURE 13.27** CUBE extension




### 13.10.3 Materialized Views

The data warehouse normally contains fact tables that store specific measurements of interest to an organization. Such measurements are organized by different dimensions. The vast majority of OLAP business analysis of "everyday activities" is based on comparisons of data that are aggregated at different levels, such as totals by vendor, by product, and by store.

Because businesses normally use a predefined set of summaries for benchmarking, it is reasonable to predefine such summaries for future use by creating summary fact tables. (See Section 13.5.6 for a discussion of additional performance-improving techniques.) However, creating multiple summary fact tables that use GROUP BY queries with multiple table joins could become a resource-intensive operation. In addition, data warehouses must also be able to maintain up-to-date summarized data at all times. So what happens with the summary fact tables after new sales data have been added to the base fact tables? Under normal circumstances, the summary fact tables are re-created. This operation requires that the SQL code be run again to re-create all summary rows, even when only a few rows needed updating. Clearly, this is a time-consuming process.

To save query processing time, most database vendors have implemented additional "functionality" to manage aggregate summaries more efficiently. This new functionality resembles the standard SQL views for which the SQL code is predefined in the database. However, the added functionality difference is that the views also store the

preaggregated rows, something like a summary table. For example, Microsoft SQL Server provides indexed views, while Oracle provides materialized views. This section explains the use of materialized views.

A **materialized view** is a dynamic table that not only contains the SQL query command to generate the rows, but also stores the actual rows. The materialized view is created the first time the query is run and the summary rows are stored in the table. The materialized view rows are automatically updated when the base tables are updated. That way, the data warehouse administrator will create the view but will not have to worry about updating the view. The use of materialized views is totally transparent to the end user. The OLAP end user can create OLAP queries, using the standard fact tables, and the DBMS query optimization feature will automatically use the materialized views if those views provide better performance.

The basic syntax for the materialized view is:

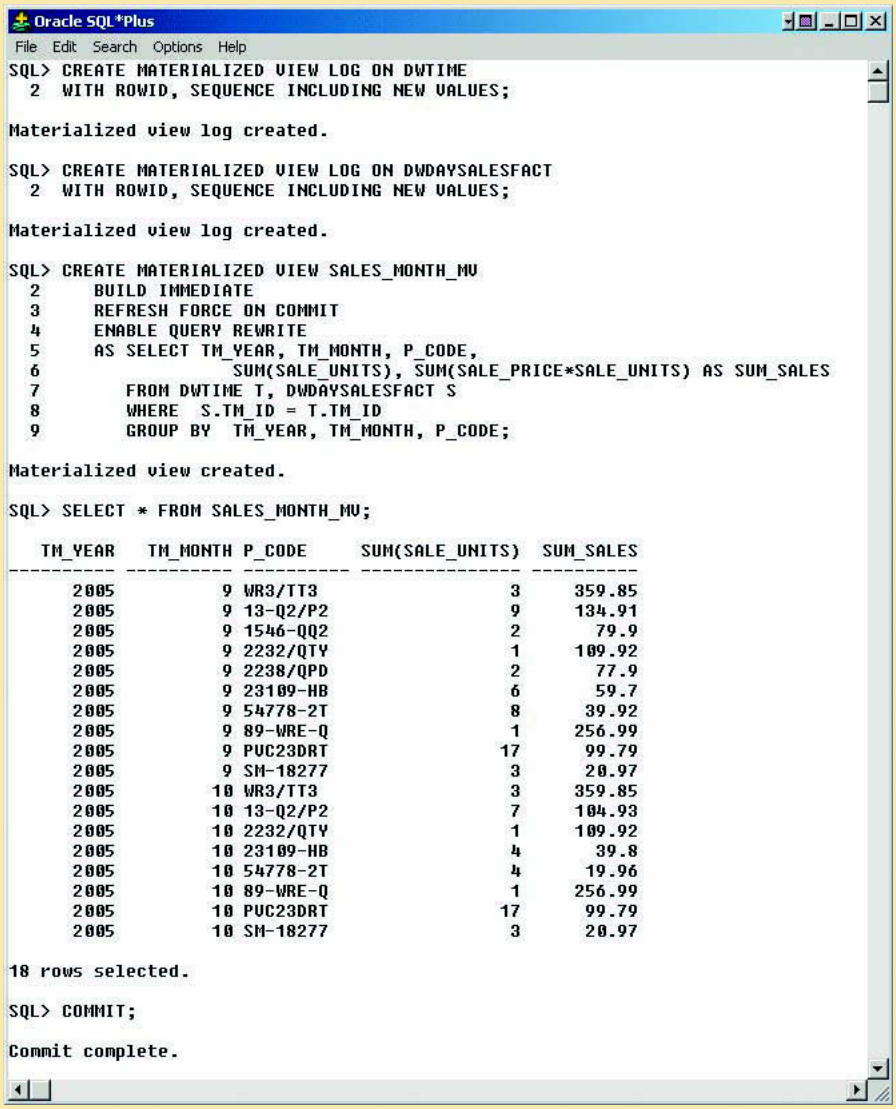
```
CREATE MATERIALIZED VIEW view_name
BUILD {IMMEDIATE | DEFERRED}
REFRESH {[FAST | COMPLETE | FORCE]} ON COMMIT
[ENABLE QUERY REWRITE]
AS select_query;
```

The BUILD clause indicates when the materialized view rows are actually populated. IMMEDIATE indicates that the materialized view rows are populated right after the command is entered. DEFERRED indicates that the materialized view rows will be populated at a later time. Until then, the materialized view is in an "unusable" state. The DBMS provides a special routine that an administrator runs to populate materialized views.

The REFRESH clause lets you indicate when and how to update the materialized view when new rows are added to the base tables. FAST indicates that whenever a change is made in the base tables, the materialized view updates only the affected rows. COMPLETE indicates that a complete update will be made for all rows in the materialized view when the select query on which the view is based is rerun. FORCE indicates that the DBMS will first try to do a FAST update; otherwise, it will do a COMPLETE update. The ON COMMIT clause indicates that the updates to the materialized view will take place as part of the commit process of the underlying DML statement, that is, as part of the commit of the DML transaction that updated the base tables. The ENABLE QUERY REWRITE option allows the DBMS to use the materialized views in query optimization.

To create materialized views, you must have specified privileges and you must complete specified prerequisite steps. As always, you must defer to the DBMS documentation for the latest updates. In the case of Oracle, you must create materialized view logs on the base tables of the materialized view. Figure 13.28 shows the steps required to create the MONTH_SALES_MV materialized view in the Oracle RDBMS.

**FIGURE 13.28** Creating a materialized view




The materialized view in Figure 13.28 computes the monthly total units sold and the total sales aggregates by product. The SALES_MONTH_MV materialized view is configured to automatically update after each change in the base tables. Note that the last row of SALES_MONTH_MV indicates that during October, the sales of product 'SM-18277' are three units, for a total of $20.97. Figure 13.29 shows the effects of an update to the DWDAYSALESFACT base table.

**FIGURE 13.29** Refreshing a materialized view

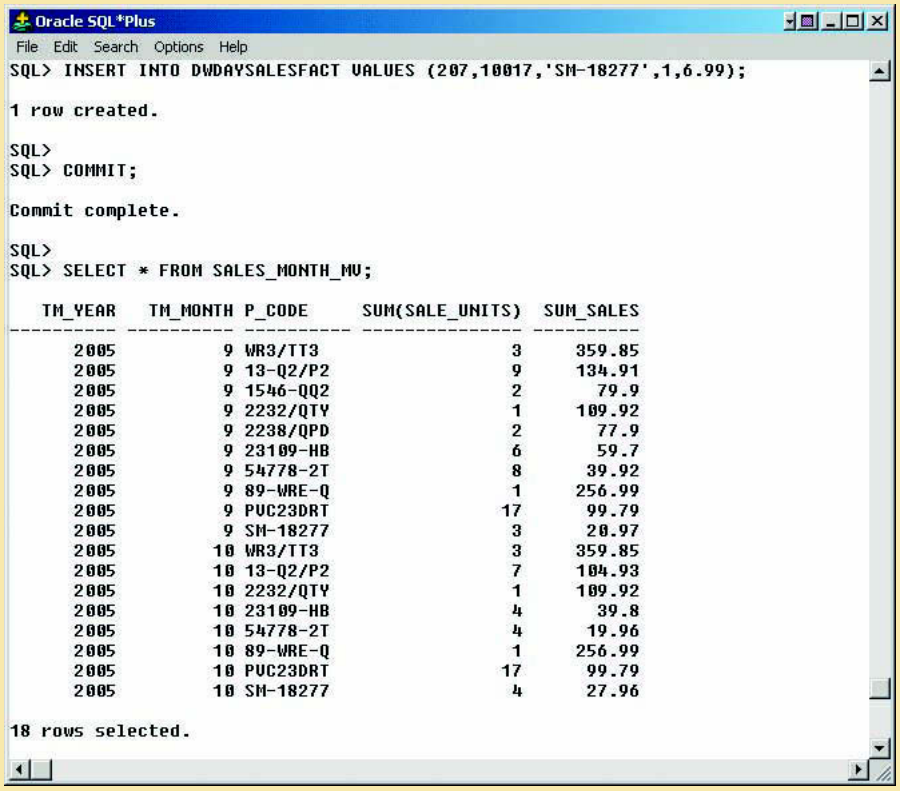
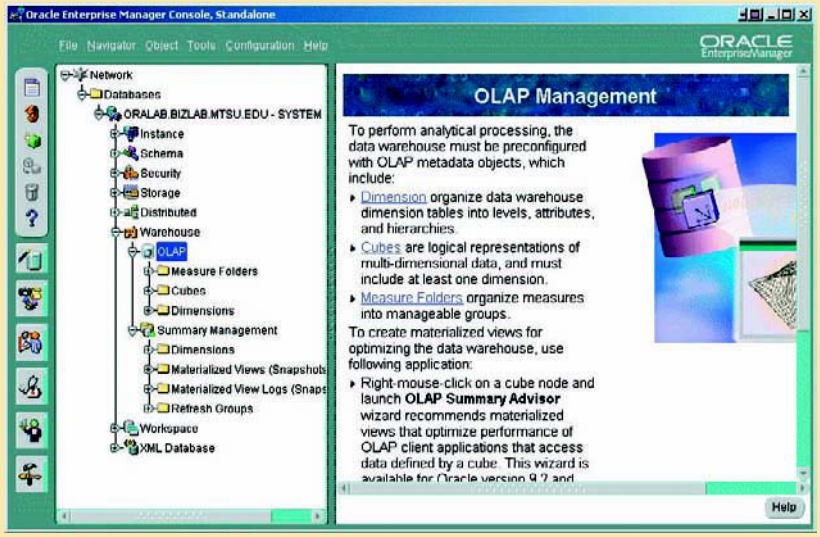


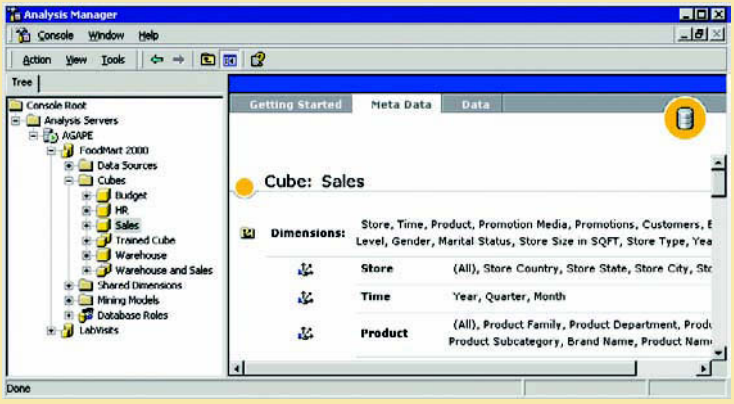

Figure 13.29 shows how the materialized view was automatically updated after the insertion of a new row in the DWDAYSALESFACT table. Note that the last row of the SALES_MONTH_MV now shows that in October, the sales of product 'SM-18277' are four units, for a total of $27.96.

Although all of the examples in this section focus on SQL extensions to support OLAP reporting in an Oracle DBMS, you have seen just a small fraction of the many business intelligence features currently provided by most DBMS vendors. For example, most vendors provide rich graphical user interfaces to manipulate, analyze, and present the data in multiple formats. Figure 13.30 shows two sample screens, one for Oracle and one for Microsoft OLAP products.

**FIGURE 13.30** **Sample OLAP applications**




**Oracle DBMS OLAP Services**




**Microsoft SQL Server Analysis Services**

## SUMMARY

- Business intelligence (BI) is a term used to describe a comprehensive, cohesive, and integrated set of applications used to capture, collect, integrate, store, and analyze data with the purpose of generating and presenting information used to support business decision making.
- BI covers a range of technologies and applications to manage the entire data life cycle from acquisition to storage, transformation, integration, analysis, monitoring, presentation, and archiving. BI functionality ranges from simple data gathering and extraction to very complex data analysis and presentation.
- Decision support systems (DSS) refers to an arrangement of computerized tools used to assist managerial decision making within a business. DSS were the original precursor of current generation BI systems.
- Operational data are not well-suited for decision support. From the end-user point of view, decision support data differ from operational data in three main areas: time span, granularity, and dimensionality.
- The requirements for a decision support DBMS are divided into four main categories: database schema, data extraction and loading, end-user analytical interface, and database size requirements.
- The data warehouse is an integrated, subject-oriented, time-variant, nonvolatile collection of data that provides support for decision making. The data warehouse is usually a read-only database optimized for data analysis and query processing. A data mart is a small, single-subject data warehouse subset that provides decision support to a small group of people.
- Online analytical processing (OLAP) refers to an advanced data analysis environment that supports decision making, business modeling, and operations research. OLAP systems have four main characteristics: use of multidimensional data analysis techniques, advanced database support, easy-to-use end-user interfaces, and client/server architecture.
- Relational online analytical processing (ROLAP) provides OLAP functionality by using relational databases and familiar relational query tools to store and analyze multidimensional data. Multidimensional online analytical processing (MOLAP) provides OLAP functionality by using multidimensional database management systems (MDBMSs) to store and analyze multidimensional data.
- The star schema is a data-modeling technique used to map multidimensional decision support data into a relational database with the purpose of performing advanced data analysis. The basic star schema has four components: facts, dimensions, attributes, and attribute hierarchies. Facts are numeric measurements or values representing a specific business aspect or activity. Dimensions are general qualifying categories that provide additional perspectives to a given fact. Conceptually, the multidimensional data model is best represented by a three-dimensional cube. Attributes can be ordered in well-defined attribute hierarchies. The attribute hierarchy provides a top-down organization that is used for two main purposes: to permit aggregation and to provide drill-down/roll-up data analysis.
- Four techniques are generally used to optimize data warehouse design: normalizing dimensional tables, maintaining multiple fact tables representing different aggregation levels, denormalizing fact tables, and partitioning and replicating tables.
- Data mining automates the analysis of operational data with the intention of finding previously unknown data characteristics, relationships, dependencies, and/or trends. The data mining process has four phases: data preparation, data analysis and classification, knowledge acquisition, and prognosis.
- SQL has been enhanced with extensions that support OLAP-type processing and data generation.

## KEY TERMS

attribute hierarchy, 544
cube cache, 539
dashboard, 519
data cube, 539
data mart, 527
data mining, 553
data store, 518
data warehouse, 525
decision support system (DSS), 519
dimensions, 542
dimension tables, 542
drill down, 521
facts, 541
fact table, 541
governance, 517
key performance indicators (KPI), 517
master data management (MDM), 516
materialized view, 560
metrics, 541
multidimensional database management system (MDBMS), 539
multidimensional online analytical processing (MOLAP), 539
online analytical processing (OLAP), 530
partitioning, 551
periodicity, 551
relational online analytical processing (ROLAP), 537
replication, 551
roll up, 521
slice and dice, 543
snowflake schema, 549
sparsity, 539
star schema, 541
very large databases (VLDBs), 525

## ONLINE CONTENT

Answers to selected Review Questions and Problems for this chapter are contained in the Student Online Companion for this book.

## REVIEW QUESTIONS

1. What is business intelligence?
2. Describe the BI framework.
3. What are decision support systems, and what role do they play in the business environment?
4. Explain how the main components of the BI architecture interact to form a system.
5. What are the most relevant differences between operational and decision support data?
6. What is a data warehouse, and what are its main characteristics?
7. Give three examples of problems likely to be encountered when operational data are integrated into the data warehouse.

**Use the following scenario to answer Questions 8–14.**

While working as a database analyst for a national sales organization, you are asked to be part of its data warehouse project team.

8. Prepare a high-level summary of the main requirements for evaluating DBMS products for data warehousing.
9. Your data warehousing project group is debating whether to prototype a data warehouse before its implementation. The project group members are especially concerned about the need to acquire some data warehousing skills before implementing the enterprise-wide data warehouse. What would you recommend? Explain your recommendations.

10. Suppose you are selling the data warehouse idea to your users. How would you define multidimensional data analysis for them? How would you explain its advantages to them?
11. Before making a commitment, the data warehousing project group has invited you to provide an OLAP overview. The group's members are particularly concerned about the OLAP client/server architecture requirements and how OLAP will fit the existing environment. Your job is to explain to them the main OLAP client/server components and architectures.
12. One of your vendors recommends using an MDBMS. How would you explain this recommendation to your project leader?
13. The project group is ready to make a final decision, choosing between ROLAP and MOLAP. What should be the basis for this decision? Why?
14. The data warehouse project is in the design phase. Explain to your fellow designers how you would use a star schema in the design.
15. Briefly discuss the decision support architectural styles and their evolution. What major technologies influenced this evolution?
16. What is OLAP, and what are its main characteristics?
17. Explain ROLAP and give the reasons you would recommend its use in the relational database environment.
18. Explain the use of facts, dimensions, and attributes in the star schema.
19. Explain multidimensional cubes and describe how the slice-and-dice technique fits into this model.
20. In the star schema context, what are attribute hierarchies and aggregation levels, and what is their purpose?
21. Discuss the most common performance improvement techniques used in star schemas.
22. Explain some of the most important issues in data warehouse implementation.
23. What is data mining, and how does it differ from traditional decision support tools?
24. How does data mining work? Discuss the different phases in the data mining process.

## PROBLEMS

### ONLINE CONTENT

The databases used for this problem set are found in the Student Online Companion for this book. These databases are stored in Microsoft Access 2000 format. The databases, named **Ch13_P1.mdb**, **Ch13_P3.mdb**, and **Ch13_P4.mdb**, contain the data for Problems 1, 3, and 4, respectively. The data for Problem 2 are stored in Microsoft Excel format in the Student Online Companion for this book. The spreadsheet filename is **Ch13_P2.xls**.

1. The university computer lab's director keeps track of lab usage, measured by the number of students using the lab. This particular function is important for budgeting purposes. The computer lab director assigns you the task of developing a data warehouse in which to keep track of the lab usage statistics. The main requirements for this database are to:
   - Show the total number of users by different time periods.
   - Show usage numbers by time period, by major, and by student classification.
   - Compare usage for different majors and different semesters.

Use the **Ch13_P1.mdb** database, which includes the following tables:

- USELOG contains the student lab access data.
- STUDENT is a dimension table containing student data.

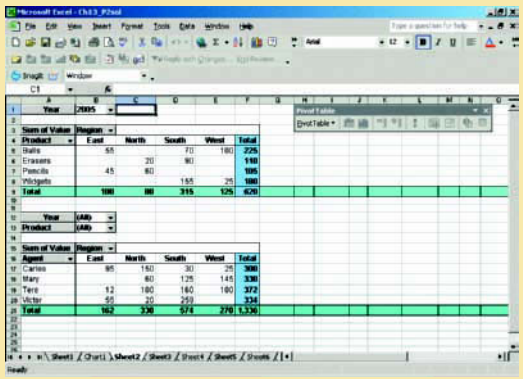
Given the three bulleted requirements and using the **Ch13_P1.mdb** data, complete Problems 1a–1g.

a. Define the main facts to be analyzed. (*Hint*: These facts become the source for the design of the fact table.)
b. Define and describe the appropriate dimensions. (*Hint*: These dimensions become the source for the design of the dimension tables.)
c. Draw the lab usage star schema, using the fact and dimension structures you defined in Problems 1a and 1b.
d. Define the attributes for each of the dimensions in Problem 1b.
e. Recommend the appropriate attribute hierarchies.
f. Implement your data warehouse design, using the star schema you created in Problem 1c and the attributes you defined in Problem 1d.
g. Create the reports that will meet the requirements listed in this problem's introduction.

2. Ms. Victoria Ephanor manages a small product distribution company. Because the business is growing fast, Ms. Ephanor recognizes that it is time to manage the vast information pool to help guide the accelerating growth. Ms. Ephanor, who is familiar with spreadsheet software, currently employs a small sales force of four people. She asks you to develop a data warehouse application prototype that will enable her to study sales figures by year, region, salesperson, and product. (This prototype is to be used as the basis for a future data warehouse database.)

   Using the data supplied in the **Ch13_P2.xls** file, complete the following seven problems:

**FIGURE P13.2E** Using a pivot table



a. Identify the appropriate fact table components.
b. Identify the appropriate dimension tables.
c. Draw a star schema diagram for this data warehouse.
d. Identify the attributes for the dimension tables that will be required to solve this problem.
e. Using a Microsoft Excel spreadsheet (or any other spreadsheet capable of producing pivot tables), generate a pivot table to show the sales by product and by region. The end user must be able to specify the display of sales for any given year. (The sample output is shown in the first pivot table in Figure P13.2E.)
f. Using Problem 2e as your base, add a second pivot table (see Figure P13.2E) to show the sales by salesperson and by region. The end user must be able to specify sales for a given year or for all years and for a given product or for all products.

**FIGURE P13.2G** 3-D bar graph showing the relationships among agent, product, and region

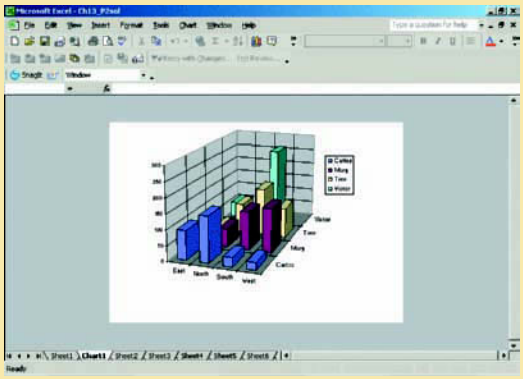
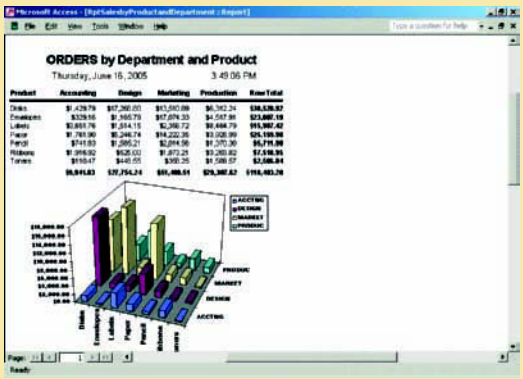


**FIGURE P13.3** Crosstab report: orders by product and department




g. Create a 3-D bar graph to show sales by salesperson, by product, and by region. (See the sample output in Figure P13.2G.)

3. Mr. David Suker, the inventory manager for a marketing research company, is interested in studying the use of supplies within the different company departments. Mr. Suker has heard that his friend, Ms. Ephanor, has developed a small spreadsheet-based data warehouse model (see Problem 2) that she uses to analyze sales data. Mr. Suker is interested in developing a small data warehouse model like Ms. Ephanor's so he can analyze orders by department and by product. He will use Microsoft Access as the data warehouse DBMS and Microsoft Excel as the analysis tool.
   a. Develop the order star schema.
   b. Identify the appropriate dimensions attributes.
   c. Identify the attribute hierarchies required to support the model.
   d. Develop a crosstab report (in Microsoft Access), using a 3-D bar graph to show orders by product and by department. (The sample output is shown in Figure P13.3.)
4. ROBCOR, whose sample data are contained in the database named **Ch13_P4.mdb**, provides "on-demand" aviation charters, using a mix of different aircraft and aircraft types. Because ROBCOR has grown rapidly, its owner has hired you to be its first database manager. (The company's database, developed by an outside consulting team, already has a charter database in place to help manage all of its operations.) Your first critical assignment is to develop a decision support system to analyze the charter data. (Review Problems 30–34 in Chapter 3, The Relational Database Model, in which the operations have been described.) The charter operations manager wants to be able to analyze charter data such as cost, hours flown, fuel used, and revenue. She would also like to be able to drill down by pilot, type of airplane, and time periods.

   Given those requirements, complete the following:

   a. Create a star schema for the charter data.
   b. Define the dimensions and attributes for the charter operation's star schema.
   c. Define the necessary attribute hierarchies.
   d. Implement the data warehouse design, using the design components you developed in Problems 4a–4c.
   e. Generate the reports that will illustrate that your data warehouse meets the specified information requirements.

   Using the data provided in the SaleCo snowflake schema in Figure 13.25, solve the following problems.

## ONLINE CONTENT

The script files used to populate the database are available in the Student Online Companion. The script files assume an Oracle RDBMS. If you use a different DBMS, consult the documentation to verify whether the vendor supports similar functionality and what the proper syntax is for your DBMS.

5. What is the SQL command to list the total sales by customer and by product, with subtotals by customer and a grand total for all product sales? (*Hint*: Use the ROLLUP command.)
6. What is the SQL command to list the total sales by customer, month, and product, with subtotals by customer and by month and a grand total for all product sales? (*Hint*: Use the ROLLUP command.)
7. What is the SQL command to list the total sales by region and customer, with subtotals by region and a grand total for all sales? (*Hint*: Use the ROLLUP command.)
8. What is the SQL command to list the total sales by month and product category, with subtotals by month and a grand total for all sales? (*Hint*: Use the ROLLUP command.)
9. What is the SQL command to list the number of product sales (number of rows) and total sales by month, with subtotals by month and a grand total for all sales? (*Hint*: Use the ROLLUP command.)
10. What is the SQL command to list the number of product sales (number of rows) and total sales by month and product category, with subtotals by month and product category and a grand total for all sales? (*Hint*: Use the ROLLUP command.)
11. What is the SQL command to list the number of product sales (number of rows) and total sales by month, product category, and product, with subtotals by month and product category and a grand total for all sales? (*Hint*: Use the ROLLUP command.)
12. Using the answer to Problem 10 as your base, what command would you need to generate the same output but with subtotals in all columns? (*Hint*: Use the CUBE command.)