CHAPTER 20

TRENDS IN DATABASE TECHNOLOGY

CHAPTER OBJECTIVES

- Get a broad, general overview of the current trends in database technology
- Understand the motivation for going beyond relational DBMSs
- Study the concepts of object orientation in information storage and retrieval
- Note how relational and object technologies are merging into object-relational systems
- Survey the use of database technology to provide business intelligence
- Explore the leading trends in special applications of database technology

Where is the database industry heading? What are the hot research interests? What are the new demands for information? Are organizations hungry for new types of information? Are the purposes and usage of information in the modern enterprise changing? If so, in what ways? What are the leading trends? Having gone through the evolution of data systems and having studied database design and development so far, you would like to explore answers to these questions. Database technology is vibrant and dynamic. It is adapting to the newer paradigms. The technology continues to evolve to address newer information requirements. In this chapter, you will examine these exciting trends; you will understand the leading areas of innovation and progress.

Although we have reviewed data models other than the relational model, our main focus has been on relational database systems. For very good reasons, too. Relational database systems still hold a large share of the database market; most enterprises have adopted them; their strengths and features still make them

Database Design and Development: An Essential Guide for IT Professionals by Paulraj Ponniah
ISBN 0-471-21877-4 Copyright © 2003 by John Wiley and Sons, Inc.

superior compared to earlier data systems. Nevertheless, with newer types of applications, relational database systems tend to fall short on some types of information requirements. For applications dealing with multimedia and spatial data, the relational model is not quite adequate. The relational model limits the types of data that can be defined and used. For engineering, scientific, office information, and high-volume data retrieval systems, we need a more flexible and powerful model than the relational model. The trend is toward extending the functionality of relational database systems or going with a different orientation of data definitions and rules for storage and retrieval. Object-oriented database systems and object-relational systems have emerged to address the inadequacies of relational systems.

Another significant area of interest in business circles relates to business intelligence. Traditionally, for two decades or more, database systems helped organizations run their day-to-day business. With amplified complexity of business conditions and increased data volumes, today's organization cannot perform its business functions without database systems. Although information provided by current database systems supports organizations in running everyday operations, these systems cannot provide information in a form suitable for making strategic decisions. Strategic decisions deal with major and far-reaching issues that help companies set directions, make plans, and improve profitability. Database systems are being adapted and used in business intelligence applications such as data warehousing, data mining, and online analytical processing.

Database systems have also evolved to meet very specific requirements and to be fitted with very specific functionalities. These are special-purpose or special-function database systems. Such database systems, built with special properties and functions, include parallel databases, active databases, intelligent databases, and so on. There are also database systems intended for special purposes or special needs: multimedia databases, mobile databases, geographic databases, and so on. We will cover these kinds of specialized trends.

OBJECT-ORIENTED DATABASES

Let us begin with a broad definition. We may generally describe object orientation as follows: software modeling and development methods that simplify construction of complex systems with individual, reusable components. A main attraction of object orientation consists of the ability to construct applications with standard, reusable components.

What then are object-oriented databases? Object-oriented databases integrate object orientation with database technology. Object orientation enables a more precise, truer depiction of real-world objects. Therefore, object-oriented databases provide a better representation of real-world information requirements than earlier data systems. As you will see, object orientation allows a developer to hide the details of implementation, concentrate on true representation, and share objects usefully. Object-oriented databases combine the features of object orientation such as abstract data typing and distinctive object identities with database capabilities such as persistence, transaction processing, concurrency control, recovery, security, and performance.

Object-oriented databases evolved to address some of the limiting features of relational databases. Although relational databases serve well for many purposes, their specific limitations need to be addressed. Here are a few of the problems that object-oriented databases are intended to address:

- Although recent versions of relational database systems support BLOBs (binary large objects), generally they can handle only limited data types. This is no longer adequate in today's environment. Object-oriented database systems include additional data types. Furthermore, you can also have user-defined data types.
- Object-oriented systems also support user-defined functions. In fact, object orientation combines the definition of data and processes.
- In the relational data model, each normalized relation does not necessarily symbolize a real-world entity completely. You may have to combine two or three normalized relations to actually represent a single entity. The object-oriented data model improves the representation of real-world entities.
- The relational model represents entities and relationship by using a single modeling construct—a relation or table. Although this appears to simplify the representation, in reality, it is difficult to show the distinction between entities and relationships just by using a single modeling construct. You cannot express the distinction with distinguishable constructs. Object-orientation enhances the semantic nature of the data model by providing distinct meanings in the constructs.

Basic Concepts

Several features of object orientation make the concept complete and useful. However, three basic notions lay the foundation for the concept. These are data abstraction, object identity, and inheritance. Most of the data manipulation and data modeling in object-oriented database systems rests on these fundamental notions. Let us begin with these notions. If you have not been exposed to object-orientated concepts, please pay special attention.

Data Abstraction Think of trying to define and describe an object. The first aspect of an object relates to what the object is. The second aspect of an object is what the object does. A set of operations may be applied to it for changing its state.

For example, consider an object known as EMPLOYEE. You can describe this object using its characteristics such as name, address, date of birth, salary, and so on. A set of values for these attributes defines the current state of the object. Now you can apply an operation on the object. Let us say that this operation is that of increasing the salary by 5 percent, thereby changing the state of the object.

Apart from the important aspects of what an object is and what an object does, there are other less significant aspects such as how to implement the object in a database or for an application. The notion of abstraction refers to the process of identifying and separating out essential aspects of an object from other less important ones. This leads us to two other basic concepts.

Encapsulation. This means that you can place the important aspects within the definition of an object as in a capsule. An object, therefore, contains both the data structure and the set of operations that can manipulate it.

Information hiding. With abstraction, you can separate the external aspects of implementation from the inner details. The inner aspects of what the object is and what it does may be hidden from the external realm.

What are the effects of encapsulation and information hiding on application processing? The internal details are insulated from the external implementation and processing requirements. Therefore, you may change the internal representation and the definition of internal operations without affecting processing requirements. An object resembles a black box component that can be constructed and changed, apart from the rest of the system. Such modularization simplifies the design and development of applications.

In object orientation, abstract data types (ADTs) provide encapsulation. An object has an interface part and an internal or implementation part. Only the interface part is made visible to users and programs.

Inheritance Through inheritance an object may inherit the internal components from an entity at a higher level. For example, an object known as SalesPerson can inherit the structure representation as well as definitions of internal operations or behavior from an object called EMPLOYEE at a higher level. Even at this stage you can readily appreciate how the property of inheritance results in code sharing and structure sharing. We will cover inheritance in more depth a little later.

Object Identity The identity of an object distinguishes that single object from all other objects. Using an object identity, you can differentiate one object from another through its attributes. The notion of object identity enables one object to refer to another object or one object to contain another object. Figure 20-1 illustrates the use of object identity as applied to an airplane and its component parts.

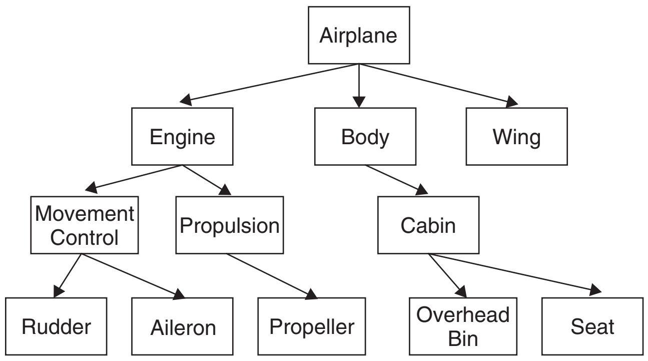

Figure 20-1 Identities for airplane identity.

Objects and Classes

The notions of object and class are keys to the understanding of object-oriented technology. An object is a single thing, and a class represents a collection of things. From this primitive definition, let us build up and review the notions in greater detail.

An Object An object is a single, uniquely identifiable entity that consists of attributes or characteristics that describe the state of the object in the real world, as well as the actions or behavior associated with it. As noted, an object contains information about its current state. An object also holds information about the actions that can take place to change the current state of the object.

One or more attributes describe the current state of an object. The values of the attributes determine the state of an object. If you modify the value of any of its attributes, the object goes from one state to another. Attributes are also called instance variables—relating to a single object instance.

Note the following concepts with regard to objects and attributes:

Simple attribute. A primitive type of attribute such as a string or integer that takes a literal or numeric value.

Complex attribute. Contains a collection of objects or references to other objects.

Reference attribute. Refers to a relationship between objects.

Complex object. Contains one or more complex attributes

Object identifier (OID). A system-generated identifier for identifying an object instance. An OID cannot be altered or reused; it does not depend on the state of the object; it is transparent to the user.

Figure 20-2 presents an example of attributes for an instance of the EMPLOYEE object. Note the different types of attributes and also determine why this is a complex object.

Attributes or representation	Type
EmployeeNumber	Simple, primitive
SocialSecurityNumber	Simple, primitive
EmployeeName	Simple, primitive
EmployeeAddress	Simple, primitive
EmployeePhone	Simple, primitive
Salary	Simple, primitive
DepartmentNumber	Reference (relationship with DEPARTMENT object)
Project	Complex (collection of PROJECT objects)

Figure 20-2 Attributes for employee instance.

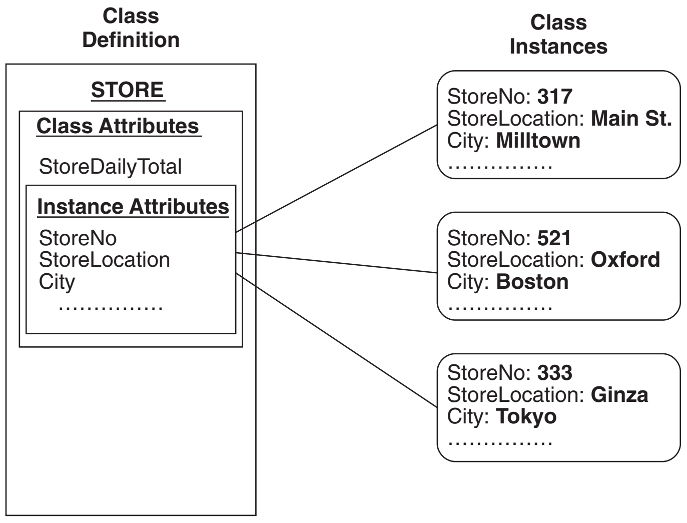

Figure 20-3 Class definition and class instances.

A Class A class is used as a template to create objects of the same structure. Objects having the same attributes and performing the same operations can be put together as a class.

Consider a class STORE. Attributes and actions may be defined once for the class. Then all store objects can be described by the attributes and actions defined for the class STORE. Each such store object is an instance of the STORE class.

A class itself may also be considered as an object and have its own special attributes and operations. These are class attributes describing the general characteristics of the entire class. For example, StoreDailyTotal may represent the total sale amount for all the stores.

Figure 20-3 illustrates a class definition and class instances.

Methods and Messages

In our discussion on encapsulation, you have noted that an object contains both the data structure and operations or functions as a self-contained, independent package.

Methods The functions or operations encapsulated by an object are known as methods. Methods indicate the behavior of the object. We mentioned that the current state of an object is defined by the current values of its attributes. Methods can be utilized to change the values of the attributes and thereby change the state of an object. A method in the CUSTOMER class may be defined to effect an address change to a customer instance. Another method may be stipulated to print the customer name and address.

Figure 20-4 shows the representation of the CUSTOMER class, object instances, and an example of a method defined within the class.

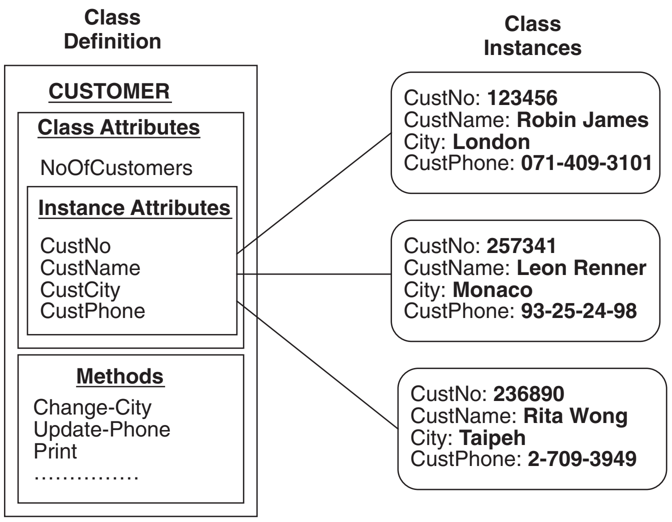

Figure 20-4 Class, class instances, and methods

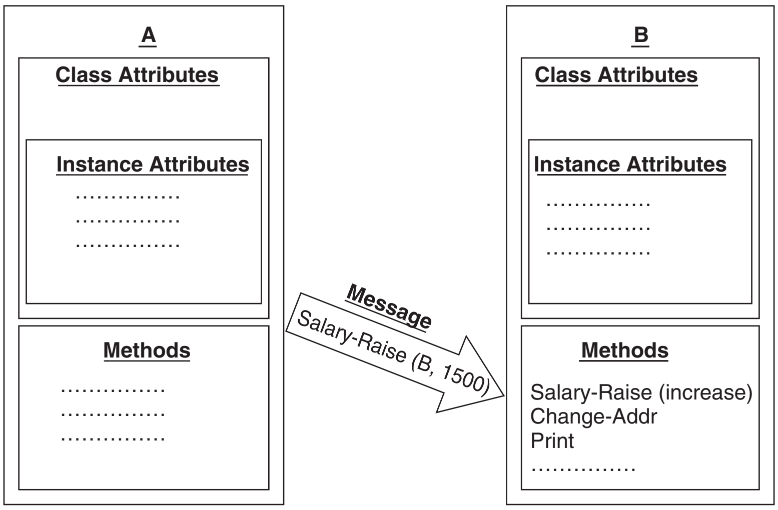

Figure 20-5 Sending of a message.

Messages Object orientation may be considered as an object-message pattern. Objects constitute the computational set in this pattern. Messages are the means of communication between objects. Each object responds to a stipulated set of messages.

A message is a simply a request from a sender object, say A, to a receiver object, say B, asking B to execute one of B's methods. In a special case, an object A may itself be both the sender and the receiver object.

Figure 20-5 illustrates how a message is sent from one object to another so that one of its methods may be executed by the receiver.

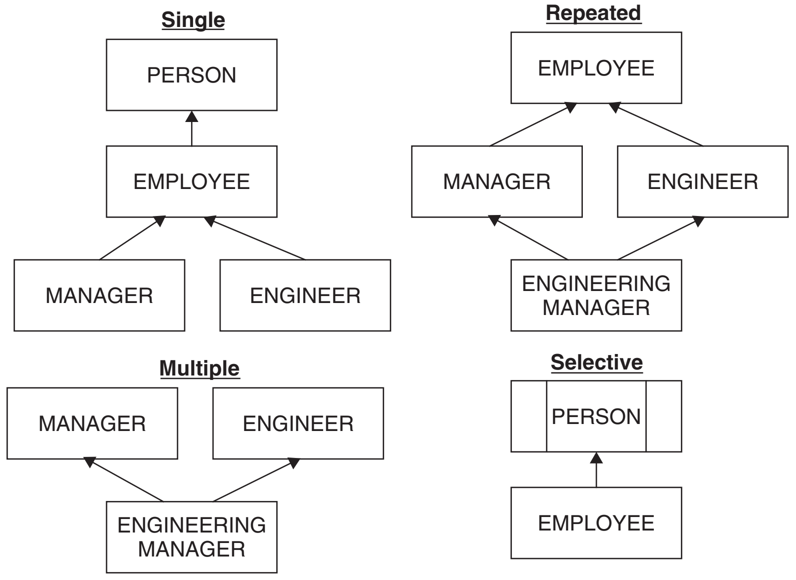

Figure 20-6 Types of inheritance.

Inheritance

Consider two classes, EMPLOYEE and MANAGER. The two classes share a number of attributes and, perhaps, a reasonable number of methods. In addition to the attributes of the class EMPLOYEE, the class MANAGER may have a few more specialized attributes. Nevertheless, the two classes are substantially similar.

A feature of object orientation, known as inheritance, allows a class to be defined as a special case of a more general class. Such special classes are called subclasses, and the general classes are called superclasses. When you form a superclass from a specialized class, the process is known as generalization. Similarly, the process of forming a subclass from a general class is called specialization. The concepts of generalization and specialization discussed here are similar to what we discussed in Chapter 6 for the object-based data model.

Inheritance may be of different forms: single, multiple, repeated, and selective. Figure 20-6 presents the various types of inheritance.

Polymorphism

This is a powerful concept in object orientation. To understand the concept, first consider the general concept of overloading. Consider any specific method defined within a class. The method has a name. Overloading allows the name of any specific method to be reused within the class or across different classes. Do you realize what this powerful technique can accomplish? Just a single message can perform different functions depending on the receiver object and the parameters passed to it. The context of the method determines its function.

Polymorphism (having many forms) is more general than overloading. Three types of polymorphism exist.

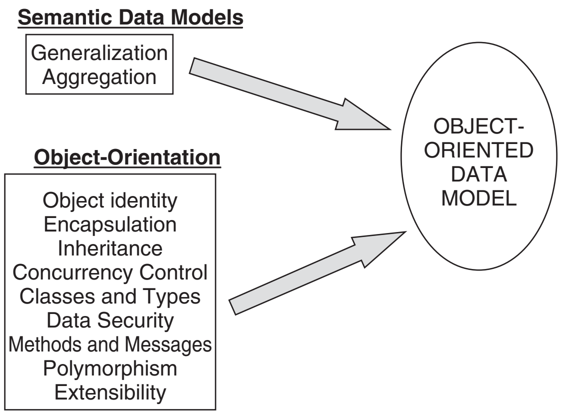

Figure 20-7 Object-oriented data model.

Operation polymorphism. Overloading is a form of operation polymorphism. With a method with one name you can have different operations performed based on the context.

Inclusion polymorphism. Relates to a method defined in one class and inherited by a subclass.

Parametric polymorphism. Uses a type as a parameter in generic type or class declarations.

Object-Oriented Data Model

When we discussed the design of a relational database, we considered the creation of a logical model. You studied the various components that make up the logical data model for traditional database systems. We also went through the steps for creating the logical model.

The detailed steps to create an object-oriented data model are beyond the scope of our discussions here. Let us just consider the features of an object-oriented data model. Basically, an object-oriented data model must represent the structure and meanings of objects defined in the context of object orientation. Figure 20-7 captures the essence of an object-oriented data model.

OODB An object-oriented data model defines an object-oriented database (OODB). Object-oriented databases combine database capabilities with object orientation. Figure 20-8 illustrates the type of integration present in OODBs.

Vendors take different approaches for integrating object orientation and database functionality. Here are the main methods:

- Add database capabilities to object-oriented programming languages.
- Include functions provided by object orientation to existing database languages.

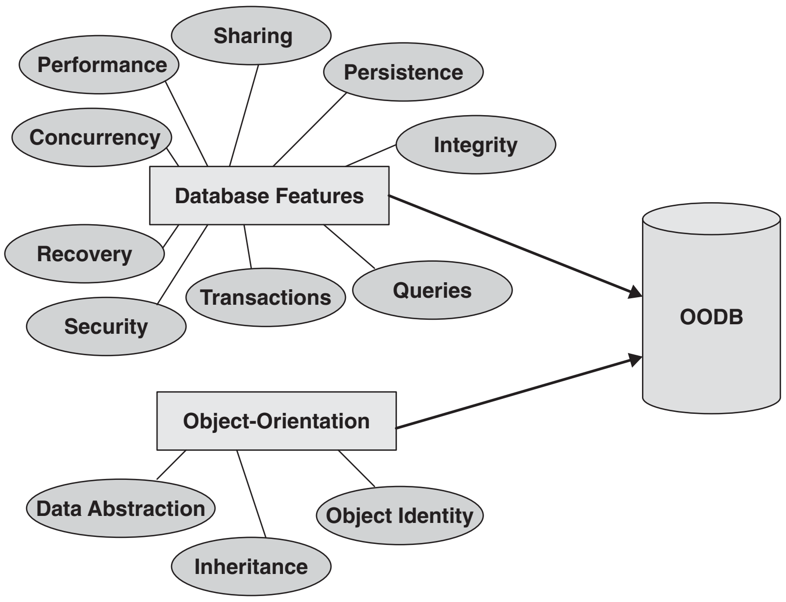

Figure 20-8 OODB: database plus object orientation.

- Take a fresh approach based on integration—not extending language to include database capabilities or the other way around.
- Embed object-oriented language for database access in host languages.
- Develop object-oriented database software for specific applications.

OODBMS Database management systems manage the storing and accessing of data from a database implemented based on a particular data model. For doing so, DBMSs must possess a number of features and support various functions. In addition to the usual features and functions of traditional DBMSs, object-oriented database management systems (OODBMSs) must be based on object orientation also.

Here is a list of functions and features of an OODBMS.

Data model support. Supports the object-oriented data model.

Data persistence. Enables data created by programs to endure and persist in the database long after termination of the program.

Data sharing. Allows data to be shared by multiple user applications.

Query processing. Provides ad hoc query facilities on object classes and subclasses.

Transaction processing. Enables transaction processing by ensuring atomicity of each transaction.

Concurrency control. Guarantees database consistency when transactions execute simultaneously.

Reliability and recovery. Provides the ability to recover from failures.

Security. Has facilities to protect the database from unauthorized access.

Scalability. Has capabilities to perform adequately even when data and transaction volumes increase.

Data distribution. Enables data to be partitioned and distributed among various physical locations.

Let us leave the topic of OODBMSs by listing their advantages and disadvantages.

Advantages

- Enhanced modeling of real-world information based on object orientation
- Ability to build new abstract data types
- Benefits of generalization and specialization resulting from provisions for defining superclasses and subclasses
- More feasible schema evolution because of the tight coupling between data and applications
- Ability to process transactions running for long durations without difficulty
- Capability to support sophisticated applications such as CAD and multimedia systems

Disadvantages

- O-O data model lacks solid theoretical foundation
- Complexity of OODBMS—data model slanted toward the programmer, not the end user
- Lacks the level of usage and experience of other data models
- Standards not firmed up
- Performance compromised by object-level locking for concurrency control
- No support for user views of data as in relational systems
- Security mechanisms not quite adequate

OBJECT-RELATIONAL DATABASES

As you know, relational DBMSs are more widely used than any other types of database systems. Object-oriented database systems started out to support special applications in engineering, telecommunications, and finance. Even though object-oriented databases are preferred for such applications, the extent of usage of these databases is still small compared to that of relational databases.

It appeared that organizations had to make the choice only between relational and object-oriented systems. There was no middle path. This has been changing in recent years. What seem to be needed are database systems that would allow representation of complex data types, user-defined functions to assist in processing, and user-defined operators to carry out complicated functions. SQL-92 is inadequate in this regard; so are pure object-oriented database systems.

An object-relational database system presents a viable solution for handling complex data types. It combines the capabilities of object technology with the features of data integrity, reliability, recovery, and security found in relational technology.

The Driving Forces

Users are moving to object-relational database management systems (ORDBMSs) not so much for object orientation. The need for new data types such as HTML pages, huge unstructured documents, audio, video—all of this within the relational infrastructure—is what drives the transition to ORDBMS.

Michael Stonebraker, a database expert and founder of the ORDBMS vendor Illustra, describes the characteristics of an ORDBMS. In his fundamental book, *Object-Relational DBMSs*, Stonebraker refers to a typical application at the State of California Department of Water Resources (DWR) that manages the waterways, canals, and water projects in that state. DWR maintains a library of nearly half a million pictures. DWR employees access this library several times a day. An employee would need to request for a picture by content with a query such as: show me Lake Cachuma (a Santa Barbara county reservoir) with a low water level. Despite an index system of captions and keywords, retrieval of the right picture within a reasonable time is virtually impossible. Information requirements such as this need a database system that would allow representation of complex data types and mechanisms to query and manipulate these data types.

When you group the various factors that have given rise to the object-relational system, you will note that two major forces propel ORDBMS: computerization of new types of applications and sophistication of business data processing applications. Let us consider these two driving forces.

New Types of Applications What we notice is an ever-growing number of applications in new areas. New types of applications range in variety from satellite imaging, weather forecasting, flood control, engineering design, and bio-engineering to spatial, geographic, and temporal systems. In particular, new types of multimedia applications keep increasing in the following areas.

Web applications Developers are loading up the World Wide Web with complex data at an amazing rate. Using the Web as a transport vehicle, Web applications require the means to run complex queries on complex data types such as multimedia objects.

Digital images Over the next decade, X-ray, ultrasound, and other such systems are expected to store enormous amounts of data on digital storage devices. Useful

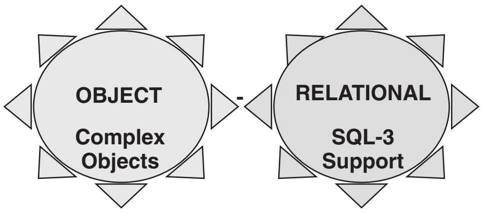

Figure 20-9 Amalgam of object/relational features.

information must be generated by managing all that digital data and combining these with textual and numerical data.

Sophistication of Business Applications Core business applications are becoming more and more sophisticated. The continuing decline in hardware and software costs promotes the enhancement of business applications with diagrams, pictures, images, audio, and video clips.

An insurance application would add pictures to the claims database file. A manufacturing application would augment the parts inventory with intricate diagrams. An auction application would capture digital images of upscale art and move the images around preauction exhibitions worldwide. A decision support system would allow sophisticated queries on complex data.

What is an ORDBMS?

The features of ORDBMSs go well beyond those of OODBMSs and those of relational DBMSs. The features and functions result from an amalgam of the two.

Michael Stonebraker defines ORDBMSs as follows: DBMSs that support a dialect of SQL-3, include nontraditional tools, and optimize for complex SQL-3 queries are called ORDBMSs. Because these support SQL, they are relational; because they support complex data, they are object-oriented.

Figure 20-9 portrays this amalgam between object orientation and relational features.

Feature Highlights

Apart from the strengths gained from the amalgam of object and relational attributes in general, the following distinct features make ORDBMS superior to either OODBMS or RDBMS.

Base Data Type Extension SQL-92 allows simple data types: integer, floating point number, character string, date/time, numerical, and decimal. In organizations with worldwide locations, two-dimensional data types would be required. Some useful data types would include 2-D polygon, 2-D trapezoid, 2-D ellipse, and so on.

Advanced Functions Extensions also cover functions such as

- Contained (point, circle) returns Boolean
- Overlaps(circle, polygon) returns Boolean

Complex Operators In addition to providing advanced functions, complex operators such as the following are also provided:

- Rotate(image, angle) returns image
- Crop (image, rectange) returns image

Create Your Own In an ORDBMS, you can create a new base data type and define your own functions and operators.

Complex Objects ORDBMS provides for complex objects. These objects may be composed by putting together base and user-defined data types.

Inheritance Inheritance consists of data inheritance and function inheritance. As with base type extension and complex objects, inheritance allows you to define new data types.

Rules In ORDBMS, the rule system is much more flexible, with the general form “on event, do action.” Thus the rule system demands the ORDBMS to watch for a given event and execute the appropriate action just before or just after the event.

Managing Large Objects Facilities are available to define and manage very large objects including audio, video, and text.

BLOB (binary large object)
CLOB (character large object)
NCLOB (fixed-width multibyte CLOB)
BFILE (binary file stored outside the database)

SQL-3: Object-Relational Support

Support features include the following:

SQL-3 provides for user-defined data types, constructors for data types, and user-defined functions and routines and also offers support for large objects and database triggers. In SQL-3, two types of objects exist: (1) row type whose instances are tuples in tables and (2) abstract data type indicative of any general type used as components of rows.

DATABASES FOR DECISION SUPPORT

The database system in an organization functions as a hub around which all business activities take place. Do you need to process an order from a customer? Do you want check inventory? Do you need to calculate the price? Do you have to check the customer’s credit status? Do you want to arrange for the shipping? Do you need to bill the customer and send an invoice? Where do you get the information for all these processes? Databases, particularly relational databases, drive the day-to-day business of many organizations.

	OPERATIONAL	INFORMATIONAL
Data Content	Current values	Archived, derived, summarized
Data Structure	Optimized for transactions	Optimized for complex queries
Access Frequency	High	Medium to low
Access Type	Read, update, delete	Read
Usage	Predictable, repetitive	Ad hoc, random, heuristic
Response Time	Sub-seconds	Several seconds to minutes
Users	Large number	Relatively small number

Figure 20-10 Operational and informational systems.

Such operational applications keep businesses alive. However, executives, managers, and analysts need a different type of informational applications providing information in different formats, to review past performance, spot future trends, and make strategic decisions. Whereas operational systems enable you to make the wheels of business turn, informational systems enable you to watch the wheels of business turn. Informational systems are decision-support applications. Figure 20-10 contrasts the features of operational and informational systems.

Decision-support systems provide business intelligence to decision makers. Data warehousing, online analytical processing (OLAP), and data mining are major decision-support applications. Database systems for decision support must provide fast processing of queries, must enable sophisticated analysis along many business dimensions, and must be able to process large volumes of data.

We will broadly discuss these decision-support systems and consider their purposes, features, and functions. Specifically, we will examine the types of database systems that are needed for these applications. You will note how these database systems differ from those for operational systems. You will also observe how decision-support database systems are designed for analysis whereas operational database systems are geared for performance.

Data Warehousing

Data warehousing is a new system environment that provides strategic information for analysis and decision making. Organizations that are building data warehouses are actually building this new system environment. This new environment is kept separate from the system environment supporting day-to-day operations. A data warehouse essentially holds the business intelligence for an organization to aid its strategic decision making. Figure 20-11 shows the nature and composition of business intelligence provided by the data warehouse environment.

At a high level of interpretation, a data warehouse contains critical measurements of the business processes stored along business dimensions. For example, a data warehouse might contain units of sales, by product, by day, by customer group, by sales district, by sales region, and by promotion. Here the business dimensions are product, day, customer group, sales district, sales region, and promotion.

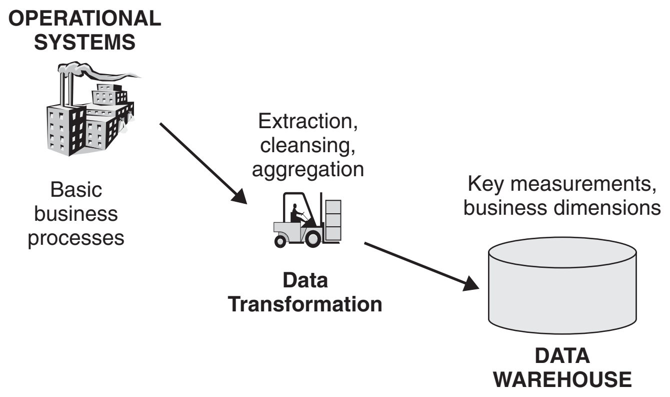

Figure 20-11 Business intelligence at the data warehouse.

From where does the data warehouse get its data? The data are derived from the operational systems that support the basic business processes of the organization. Between the operational systems and the data warehouse, there is a data staging area. In this staging area, operational data are cleansed and transformed into a form suitable to be placed in the data warehouse for easy retrieval and quick analysis.

Data Warehouse: A New Type of System Environment You have noted that the data warehouse environment is not the same as the operational environment supporting the day-to-day business operations. Here are the defining features of the data warehouse environment:

- Database designed for analytical tasks
- Data from multiple applications
- Easy to use and conducive to long interactive sessions by users
- Read-intensive data usage
- Direct interaction with the system by the users without IT assistance
- Content updated periodically and stable
- Content to include current and historical data
- Ability for users to run queries and get results online
- Ability for users to initiate reports

So how can you define a data warehouse? The data warehouse is an informational environment that

- Provides an integrated and total view of the enterprise,
- Makes the enterprise's current and historical information easily available for decision making,
- Makes decision support transactions possible without hindering operational systems,
- Renders the organization's information consistent, and
- Presents a flexible and interactive source of strategic information.

Most of the processing in the data warehouse environment for strategic information is analytical. Four levels of analytical processing are possible:

1. Running simple queries and reports against current and historical data
2. Performing "what if" analysis is many different ways
3. Querying, stepping back, analyzing, and then continuing the process to any desired length
4. Spotting historical trends and applying them for future results

Data Warehouse: Underlying Principles

A Simple Concept In the final analysis, data warehousing is a simple concept. It is born out of the need for strategic information; it is the result of the search for a new way to provide strategic information. The old methods using the operational computing environment, all through the last two decades, were unsatisfactory. The new concept is not to generate fresh data but to make use of the large volumes of existing data and to transform the data into forms suitable for providing strategic information.

The data warehouse exists to answer questions users have about the business, the performance of the various operations, the business trends, and about what can be done to improve the business. The data warehouse exists to provide business users with direct access to data, to provide a single unified version of the performance indicators, to record the past accurately, and to provide the ability to view data from many different perspectives. In short, the data warehouse is there to support decisional processes.

When everything is said about data warehousing, you will realize that it is really a simple concept. Take all the data you already have in the organization, clean and transform the data, and then provide useful strategic information. What could sound simpler than that?

An Environment, Not a Product A data warehouse is not a single software or hardware product you purchase to provide strategic information. It is rather a computing environment. It is an environment where users can find strategic information. It is an environment where users are put directly in touch with the data they need to make better decisions. It is a user-centric environment.

Let us summarize the characteristics of this new computing environment called data warehouse created for the users:

- An ideal environment for data analysis and decision support
- Fluid, flexible, and interactive
- 100 Percent user-driven
- Very responsive and conducive to ask-answer-ask-again pattern
- Provides ability to discover answers to complex, unpredictable questions

A Blend of Many Technologies Although a simple concept, data warehousing involves different functions. It involves the function of data extraction, the function of loading the data, the function of transforming the data, the function of storing

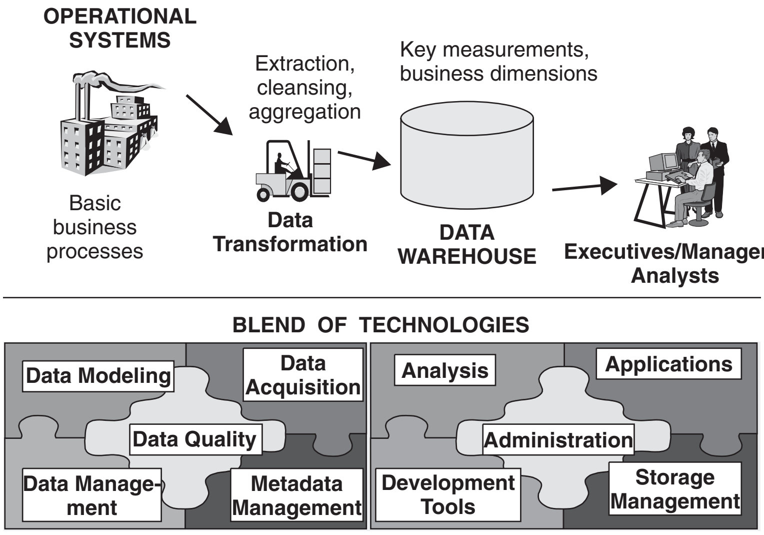

Figure 20-12 Data warehouse: a blend of technologies.

the data, and the function of providing user interfaces. Different technologies are therefore needed to support these functions. Figure 20-12 illustrates how various technologies blend together in the data warehouse environment.

Although many technologies are in use, they all work together in a data warehouse. The end result is the creation of a new computing environment for the purpose of providing strategic information every enterprise needs desperately. Several vendor tools are available in each of these technologies. You do not have to build your data warehouse from scratch.

Nature of the Data It is important to consider the nature of the data in the data warehouse. How are these data different from the data in any operational system? Why does they have to be different? How is the data content in the data warehouse used?

Subject-Oriented In operational systems, we store data by individual applications. For a banking institution, data sets for a consumer loans application contain data for that particular application. Data sets for other distinct applications of checking accounts and savings accounts relate to those specific applications. Again, in an insurance company, different data sets support individual applications such as automobile insurance, life insurance, and workers' compensation insurance.

In striking contrast, in the data warehouse, data are stored by subjects, not by applications. What are business subjects? Business subjects differ from organization to organization. These are the subjects critical for the enterprise. For a manufacturing company, sales, shipments, and inventory are critical business subjects. For a retail store, sales at the check-out counter is a critical subject.

In a data warehouse, there is no application flavor. The data in a data warehouse cut across applications.

Integrated For proper decision making, you need to pull all the relevant data from the various applications. The data in the data warehouse come from several operational systems. Source data are in different databases, files, and data segments. These are disparate applications. The operational platforms could be different. The operating systems could be different. File layouts, character code representations, field naming conventions—all of these could be different.

In addition to data from internal operational systems, for many enterprises, data from outside sources are likely to be very important. Companies such as Metro Mail, A. C. Nielsen, and IRI specialize in providing vital data on a regular basis. Your data warehouse may need data from such sources. This is one more variation in the mix of source data for a data warehouse.

Before the data from various disparate sources can be usefully stored in a data warehouse, you have to remove the inconsistencies. You have to standardize the various data elements; you have to make sure of the meanings of data names in each source application. Before moving the data into the data warehouse, you have to go through a process of transformation, consolidation, and integration of the source data.

Time-Variant For an operational system, the stored data contain the *current* values. In an accounts receivable system, the balance is the *current* outstanding balance in the customer's account. In an order entry system, the status of an order is the *current* status of the order. In a consumer loans application, the balance amount owed by the customer is the *current* amount. Of course, we store some past transactions in operational systems. But, essentially, operational systems reflect *current* information because these systems support day-to-day *current* operations.

On the other hand, the data in the data warehouse are meant for analysis and decision making. If a user is looking at the buying pattern of a specific customer, the user needs data not only about the current purchase but on the past purchases as well. When a user wants to find out the reason for the drop in sales in the North East division, the user needs all the sales data for that division over a period extending back in time. When an analyst in a grocery chain wants to promote two or more products together, that analyst wants sales of the selected products over a number of past quarters.

A data warehouse, because of the very nature of its purpose, has to contain historical data, not just current values. Data are stored as snapshots over past and current periods. Every data structure in the data warehouse contains the time element. You will find historical snapshots of the operational data in the data warehouse. This aspect of the data warehouse is quite significant for both the design and implementation phases.

For example, in a data warehouse containing units of sale, the quantity stored in each file record or a table row relates to a specific time element. Depending on the level of the details in the data warehouse, the sales quantity in a record may relate to a specific date, a specific week, a specific month, or a specific quarter.

The time-variant nature of the data in a data warehouse allows for analysis of the past, relates information to the present, and enables forecasts for the future.

Nonvolatile Data extracted from the various operational systems and pertinent data obtained from outside sources are transformed, integrated, and stored in the data warehouse. The data in the data warehouse are not intended to run the day-to-day business. When you want to process the next order received from a customer, you do not look into the data warehouse to find the current stock status. The operational order entry application is meant for that purpose. In the data warehouse you keep the extracted stock status data as snapshots over time. You do not update the data warehouse every time you process a single order.

Data from the operational systems are moved into the data warehouse at specific intervals. Depending on the requirements of the business, these data movements take place twice a day, once a day, once a week, or once in two weeks. In fact, in a typical data warehouse, data movements to different data sets may take place at different frequencies. The changes to the attributes of the products may be moved once a week. Any revisions to the geographical setup may be moved once a month. The units of sales may be moved once a day. You plan and schedule the data movements or data loads based on the requirements of your users.

Not every business transaction updates the data in the data warehouse. The business transactions update the operational system databases in real time. We add, change, or delete data from an operational system as each transaction happens. You do not usually update the data in the data warehouse. You do not delete the data in the data warehouse in real time. Once the data are captured in the data warehouse, you do not run individual transactions to change the data there. Data updates are commonplace in an operational database; not so in a data warehouse. The data in a data warehouse are not as volatile as the data in an operational database. The data in a data warehouse are primarily for query and analysis.

Data Warehouse Components Figure 20-13 presents the basic components of a data warehouse.

Let us briefly review the components illustrated in the figure.

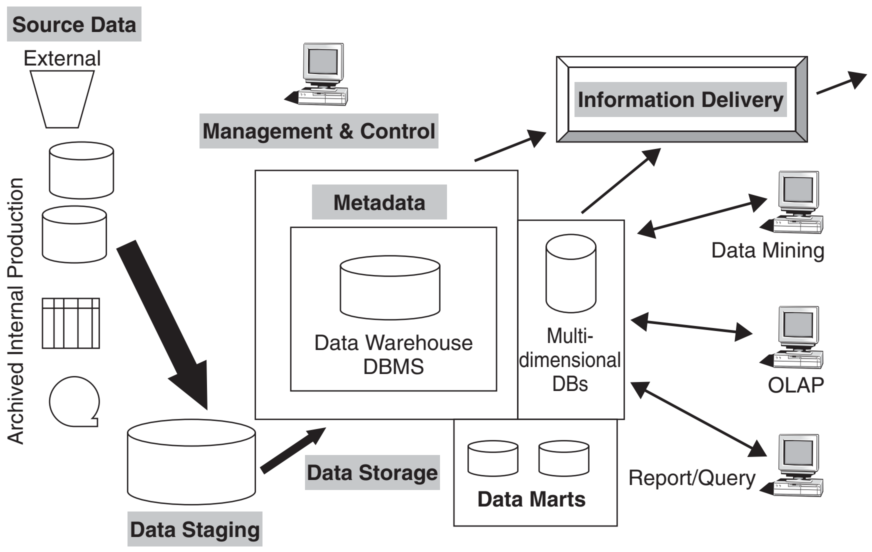

Figure 20-13 Data warehouse components.

Source Data Source data coming into the data warehouse may be grouped into four broad categories:

Production data. Come from the various operational systems of the organization. The significant characteristic of production data is its wide disparity.

Internal data. These data relate to "private" spreadsheets, documents, customer profiles, and sometimes even departmental databases. These are internal data, parts of which could be useful in a data warehouse.

Archived data. These are the historical data separated out from current data and stored in archived media. Historical data are essential in a data warehouse.

External data. This type of data emanates from outside the organization. Many executives use performance indicators, competitor's market shares, financial parameters, and so on from outside sources.

Data Staging This component of the data warehouse refers to the processes and the area where preparation of the data takes place. In the data staging area, data are cleansed, transformed, and prepared for loading.

Data Storage This component includes all data storage in the overall data warehouse, data marts, and other multidimensional databases. In the top-down method of design, the overall data warehouse feeds dependent data marts. In a practical approach, a collection of conformed data marts, by themselves, constitute the organization's data warehouse.

Metadata The metadata component in a data warehouse environment is similar to the data dictionary or system catalog of a DBMS. But the metadata component has more functions than a routine data dictionary. Metadata serves as a road map for users to navigate within a data warehouse and examine the contents.

Information Delivery This component includes a wide variety of methods for delivery of information to users. Users obtain information on-line by e-mail or through the Internet or an intranet. Information delivery mechanisms include ad hoc reports, complex queries, multidimensional analysis, statistical analysis, data mining applications, and executive information systems.

Management and Control This component sits on top of all other components and coordinates their services and functions within the data warehouse.

Dimensional Data Model You are familiar with data modeling for operational or OLTP systems. We adopt the entity-relationship (E-R) modeling technique to create a data model. Operational systems possess the following characteristics:

- Capture details of events or transactions
- Focus on individual events

- Are windows into microlevel transactions
- Need data at detailed level to run the business
- Suitable only for questions at level of individual transactions
- Data consistency, nonredundancy, and efficient data storage are critical

These characteristics make E-R modeling technique appropriate for OLTP systems. The E-R modeling technique removes data redundancy, ensures data consistency, and expresses microscopic relationships.

On the other hand, data warehousing systems possess different characteristics and serve other purposes. Here is a summarized list of data warehousing features:

- Meant to answer questions on the overall processes
- Focuses on how managers view the business
- Reveals business trends
- Information centered on a business process
- Answers show how business processes are monitored and measured
- Enables measures to be analyzed and studied along several business dimensions

Fact and Dimension Tables For modeling data warehouses, E-R modeling technique is not suitable. We need to use a different technique. For modeling data warehouses, we use the dimensional modeling technique. For a moment, consider how a manager or an analyst performs a typical analysis. For example, if an analyst wants to analyze sales, how does he or she perform the analysis? He or she needs data such as sales quantities, sales dollars, sales price, sales cost, and margin. These pieces of information are known as facts—metrics to be analyzed. How are the facts going to be analyzed?

Let us consider a typical query: How much revenue did sales of product Widget-1 to married customers generate for the past three years, in the northeast territory? In the query, the fact being analyzed is the revenue or margin. The revenue is being analyzed along four business dimensions—product, customer, time, and salesperson's territory.

The dimensional modeling technique enables us to model facts and dimensions. The result is a star schema.

STAR schema Figure 20-14 presents a STAR schema for the subject of sales. Notice the fact table in the middle surrounded by dimension tables.

Inside the fact table, you will find all the measures or facts to be used for analysis. In each dimension table, you will find the attributes of the particular business dimension. Notice the primary keys of each table. Each dimension table is in a one-to-many relationship with the fact table in the middle. With such an arrangement in the STAR schema with the fact table in the middle, query executions are optimized.

Database Software Considerations Examine the features of the leading commercial relational DBMSs (RDBMSs). As data warehousing becomes more preva-

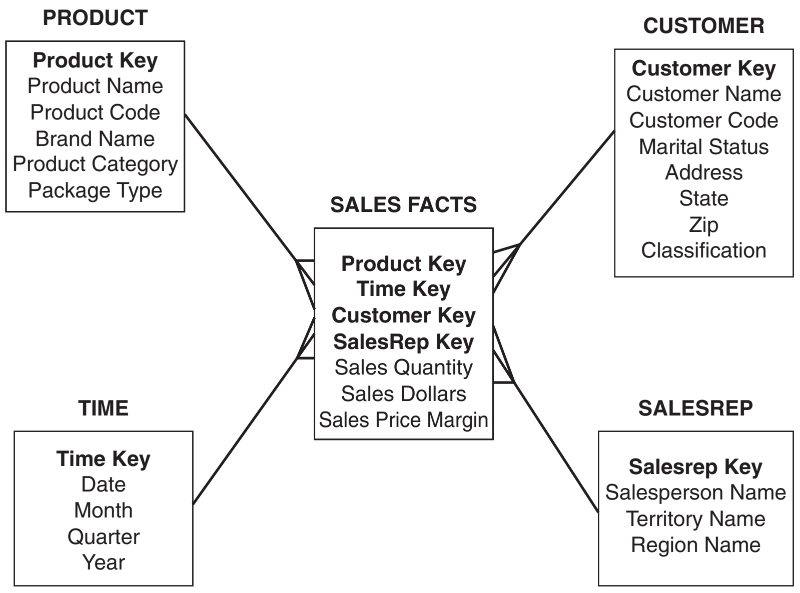

Figure 20-14 STAR schema for sales.

lent, you would expect to see data warehouse features being included in the software products. That is exactly what the database vendors are doing. Data warehouse-related add-ons are becoming parts of the database offerings. The database software that started out for use in operational OLTP systems is being enhanced to cater to decision support systems. DBMSs have also scaled up to support very large databases.

Some RDBMS products now include support for the data acquisition area of the data warehouse. Mass loading and retrieval of data from other database systems have become easier. Some vendors have paid special attention to the data transformation function. Replication features have been reinforced to assist in bulk refreshes and incremental loading of the data warehouse.

Bitmapped indexes could be very effective in a data warehouse environment to index on fields that have smaller number of distinct values. For example in a database table containing geographic regions, the number of distinct region codes is low. However, frequently queries involve selection by regions. In this case, retrieval by a bitmapped index on the region code values can be very fast. Vendors have strengthened this type of indexing.

Apart from these enhancements, the more important enhancements relate to load balancing and query performance. These two features are critical in a data warehouse. Your data warehouse is query-centric. Everything that can be done to improve query performance is most desirable. The DBMS vendors are providing parallel processing features to improve query performance. Let us briefly review the parallel processing options within the DBMS that can take full advantage of parallel server hardware.

Parallel Processing Options Parallel processing options in database software are intended only for machines with multiple processors. Most of the current database software can parallelize a large number of operations. These operations include the following: mass loading of data, full table scans, queries with exclusion conditions, queries with grouping, selection with distinct values, aggregation, sorting, creation of tables using subqueries, creating and rebuilding indexes, inserting rows into one table from other tables, enabling constraints, star transformation, which is an optimization technique when processing queries against a STAR schema, and so on. Note that this is an impressive list of operations that the RDBMS can process in parallel.

Let us now examine what happens when a user initiates a query at the workstation. Each session accesses the database through a server process. The query is sent to the DBMS, and data retrieval takes place from the database. Data are retrieved and the results are sent back, all under the control of the dedicated server process. The query dispatcher software is responsible for splitting the work and distributes the units to be performed among the pool of available query server processes to balance the load. Finally, the results of the query processes are assembled and returned as a single, consolidated result set.

Interquery Parallelization In this method, several server processes handle multiple requests simultaneously. Multiple queries may be serviced based on your server configuration and the number of available processors. You may take advantage of this feature of the DBMS successfully on SMP systems, thereby increasing throughput and supporting more concurrent users.

However, interquery parallelism is limited. Let us see what happens here. Multiple queries are processed concurrently, but each query is still being processed serially by a single server process. Suppose that a query consists of index read, data read, sort, and join operations; then these operations are carried out in this order. Each operation must finish before the next one can begin. Parts of the same query do not execute in parallel. To overcome this limitation, many DBMS vendors have come up with versions of their products that provide intraquery parallelization.

Intraquery Parallelization Using the intraquery parallelization technique, the DBMS splits the query into the lower-level operations of index read, data read, join, and sort. Then each one of these base operations is executed in parallel on a single processor. The final result set is the consolidation of the intermediary results. Let us review three ways a DBMS can provide intraquery parallelization, that is, parallelization of parts of the operations within the same query itself.

Horizontal parallelism. The data are partitioned across multiple disks. Parallel processing occurs within each single task in the query, for example, data read, which is performed on multiple processors concurrently on different sets of data to be read spread across multiple disks. After the first task is completed from all of the relevant parts of the partitioned data, the next task of that query is carried out, and then the next one after that task, and so on.

Vertical parallelism. This kind of parallelism occurs among different tasks, not just a single task in a query as in the case of horizontal parallelism. All component query operations are executed in parallel, but in a pipelined manner. This assumes that the RDBMS has the capability to decompose the query into subtasks where each subtask has all the operations of index read, data read, join, and sort. Then each subtask executes on the data in a serial fashion. In this approach, ideally the database records are processed by one step and immediately given to the next step for processing, thus avoiding wait times. Of course, in this method, the DBMS must possess a very high level of sophistication in decomposing tasks.

Hybrid method. In this method, the query decomposer partitions the query both horizontally and vertically. Naturally, this approach produces the best results. You will realize the greatest utilization of resources, optimal performance, and high scalability.

Selection of the DBMS Our discussions of the server hardware and the DBMS parallel processing options must have convinced you that selection of the DBMS is most crucial. You must choose the server hardware with the appropriate parallel architecture. You choice of the DBMS must match with the selected server hardware. These are critical decisions for your data warehouse.

Broadly, the following elements of business requirements affect the choice of the DBMS:

Level of user experience

If the users are totally inexperienced with database systems, the DBMS must have features to monitor and control runaway queries. On the other hand, if many of your users were power users, then they would be formulating their own queries. In this case, the DBMS must support easy SQL-type language interface.

Types of queries

The DBMS must have a powerful optimizer if most of the queries are complex and produce large result sets. Alternatively, if there is an even mix of simple and complex queries, there must be some sort of query management in the database software to balance the query execution.

Need for openness

The degree of openness depends on the back-end and front-end architectural components, and those, in turn, depend on the business requirements.

Data loads

The data volumes and load frequencies determine the strengths in the areas of data loading, recovery, and restart.

Metadata management

If your metadata component need not be elaborate, then a DBMS with an active data dictionary may be sufficient. Let your requirements definition reflect the type and extent of the metadata framework.

Data repository locations

Is your data warehouse going to reside in one central location, or is it going to be distributed? The answer to this question will establish whether the selected DBMS must support distributed databases.

Data warehouse growth

Your business requirements definition must contain information on the estimated growth in the number of users and in the number and complexity of queries. The growth estimates will have a direct relation to how the selected DBMS supports scalability.

In addition to the criteria that the selected DBMS must have load balancing and parallel processing options, we list below other key features you need to consider when selecting the DBMS for your data warehouse:

Query governor	To anticipate and abort runaway queries
Query optimizer	To parse and optimize user queries
Query management	To balance the execution of different types of queries
Load utility	For high-performance data loading, recovery, and restart
Metadata management	With an active data catalog or dictionary
Scalability	In terms of both number of users and data volumes
Extensibility	Having hybrid extensions to OLAP databases
Portability	Across platforms
Query tool APIs	For tools from leading vendors
Administration	Providing support for all DBA functions

Online Analytical Processing (OLAP)

In today's business conditions, users need to go beyond basic facilities for analysis provided by the data warehouse. Data warehousing is query-centric and designed for querying and analysis. However, in the changed and competitive business environment, users must have the capability to perform far more complex analysis in less time.

Let us quickly examine how the traditional methods of analysis provided in a data warehouse are not sufficient and perceive the nature of the more effective analytical system demanded by users.

Need for Multidimensional Analysis Let us quickly review the business model of a large retail operation. If you just look at daily sales, you soon realize that the sales are interrelated to many business dimensions. Daily sales are meaningful only when they are related to the dates of the sales, the products, the distribution channels, the stores, the sales territories, the promotions, and a few more dimensions. Multidimensional views are inherently representative of any business model. Very few models are limited to three dimensions or fewer. For planning and making strategic decisions, managers and executives probe into business data by scenarios.

For example, they compare actual sales against targets and against sales in prior periods. They examine the breakdown of sales by product, by store, by sales territory, by promotion, and so on.

Decision makers are no longer satisfied with one-dimensional queries such as "How many units of Product A did we sell in the store in Milltown, New Jersey?" Consider the following query, which is more useful: "How much revenue did the new Product X generate during the last three months, broken down by individual months, in the South Central territory, by individual stores, broken down by promotions, compared to estimates, and compared to the previous version of the product?" The analysis does not stop with this single multidimensional query. The user continues to ask for further comparisons with similar products, comparisons among territories, and views of the results by rotating the presentation between columns and rows.

For effective analysis, your users must have easy methods for performing complex analysis along several business dimensions. They need an environment that presents a multidimensional view of data providing the foundation for analytical processing through easy and flexible access to information. Decision makers must be able to analyze data along any number of dimensions, at any level of aggregation, with the capability of viewing results in a variety of ways. They must have the ability to drill down and roll up along the hierarchies of every dimension. Without a solid system for true multidimensional analysis, your data warehouse is incomplete.

In any analytical system, time is a critical dimension. Hardly any query is executed without having time as one of the dimensions along which analysis is performed. Furthermore, time is a unique dimension because of its sequential nature. November of a year always comes after October of that year. Users monitor performance over time, as for example, performance this month compared to last month, or performance this month compared with performance in the same month last year.

Another point about the uniqueness of the time dimension is the way in which the hierarchies of the dimension work. A user may look for sales in March and may also look for sales for the first four months of the year. In the second query for sales for the first four months, the implied hierarchy at the next higher level is an aggregation taking into account the sequential nature of time. No user looks for sales of the first four stores or the last three stores. There is no implied sequence in the store dimension. True analytical systems must recognize the sequential nature of time.

Need for Fast Access and Powerful Calculations Whether a user's request is for monthly sales of all products along all geographical regions or for year-to-date sales in a region for a single product, the query and analysis system must have consistent response times. Users must not be penalized for the complexity of their analysis. Both the size of the effort to formulate a query or the amount of time to receive the result sets must be consistent irrespective of the query types.

Let us take an example to understand how the speed of the analysis process matters to users. Imagine a business analyst looking for reasons why profitability dipped sharply in the recent months in the entire enterprise. The analyst starts this analysis by querying for the overall sales for the last five months for the entire company broken down by individual months. The analyst notices that, although the sales do not show a drop, profitability exhibits a sharp reduction for the last three months. Now the analysis proceeds further when the analyst wants to find out which countries show reductions. The analyst requests a breakdown of sales by major worldwide regions and notes that the European region reveals the reduction in profitability. Now the analyst senses that clues become more pronounced. So the analyst looks for a breakdown of the European sales by individual countries. The analyst finds that profitability is increased for a few countries and decreased sharply for some other countries and has stayed the course for the rest. At this point, the analyst introduces another dimension into the analysis. Now the analyst wants the breakdown of profitability for the European countries by country, by month, and by product. This step brings the analyst closer to the reason for the decline in the profitability. The analyst observes that the countries in the European Union show a very sharp decline in profitability for the last two months. Further queries reveal that manufacturing and other direct costs remained at the usual levels but the indirect costs shot up. The analyst is now able to determine that the decline in profitability is because of the additional tax levies on some products in the European Union countries. The analyst has also determined the exact effect of the levies so far. Strategic decisions follow on how to deal with the decline in profitability.

Now look at Figure 20-15 showing the steps through the single analysis session.

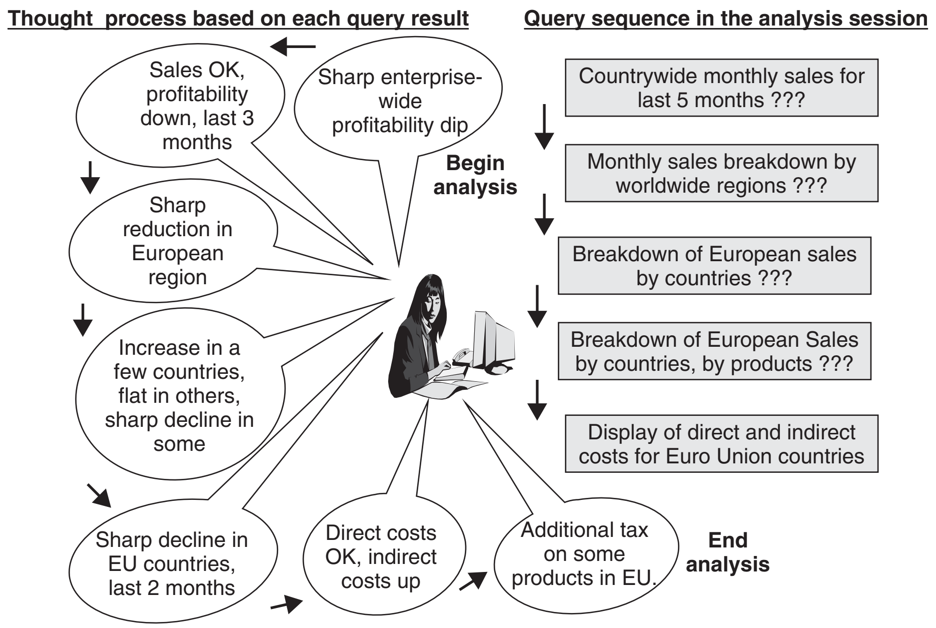
Thought process based on each query result
Query sequence in the analysis session
Sharp enterprise-wide profitability dip
Begin analysis
Sales OK, profitability down, last 3 months
Sharp reduction in European region
Increase in a few countries, flat in others, sharp decline in some
Sharp decline in EU countries, last 2 months
Direct costs OK, indirect costs up
Additional tax on some products in EU.
End analysis
Countrywide monthly sales for last 5 months ???
Monthly sales breakdown by worldwide regions ???
Breakdown of European sales by countries ???
Breakdown of European Sales by countries, by products ???
Display of direct and indirect costs for Euro Union countries

Figure 20-15 Query steps in an analysis session.

How many steps are there? Many steps, but a single analysis session. Many steps, but a single train of thought. Each step in this train of thought constitutes a query. The analyst formulates each query, executes the query, waits for the result set to appear on the screen, and studies the result set. Each query is interactive because the result set from one query forms the basis for the next query. In this manner of querying, the user cannot maintain the train of thought unless the momentum is preserved. Fast access is absolutely essential for an effective analytical processing environment.

Did you notice that none of the queries in the above analysis session included any serious calculations? This is not typical. In a real-world analysis session, many of the queries require calculations, sometimes, complex calculations. What is the implication here? An effective analytical processing environment must not only be fast and flexible, it must also support complex and powerful calculations.

What follows is a list of typical calculations that get included in the query requests:

- Roll-ups to provide summaries and aggregations along the hierarchies of the dimensions
- Drill-downs from the top level to the lowest along the hierarchies of the dimensions, in combinations among the dimensions
- Simple calculations such as computation of margins (sales minus costs)
- Share calculations to compute the percentage of parts to the whole
- Algebraic equations involving key performance indicators
- Moving averages and growth percentages
- Trend analysis using statistical methods

Features and Functions OLAP provides users with the ability to perform multidimensional analysis with complex calculations.

Here is a summary list of features and functions of OLAP:

- Enables analysts, executives, and managers to gain useful insights from the presentation of data.
- Can reorganize metrics along several dimensions and allow data to be viewed from different perspectives.
- Supports multidimensional analysis.
- Is able to drill down or roll up within each dimension.
- Is capable of applying mathematical formulas and calculations to measures.
- Provides fast response, facilitating speed-of-thought analysis.
- Complements the use of other information delivery techniques such as data mining.
- Improves the comprehension of result sets through visual presentations using graphs and charts.
- Can be implemented on the Web.
- Designed for highly interactive analysis.

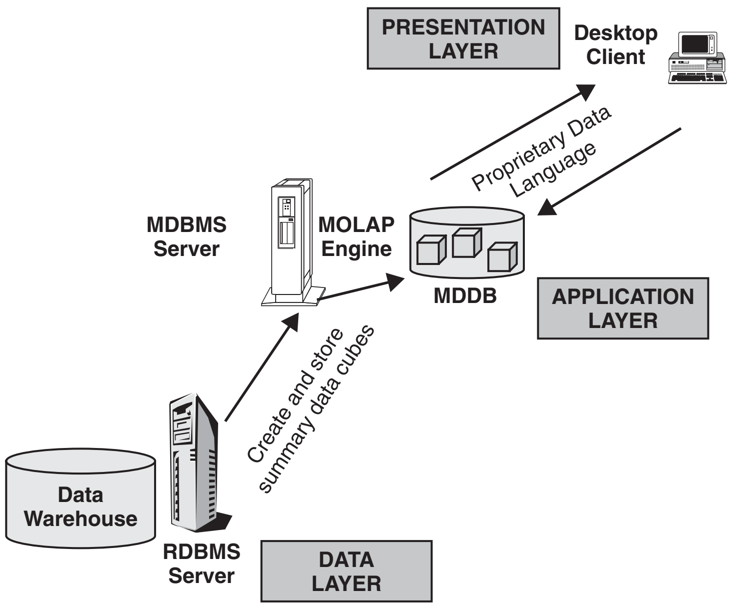

Figure 20-16 The MOLAP model.

OLAP Models OLAP systems provide multidimensionality in one of two ways: multidimensional OLAP and relational OLAP. The method of storing data differentiates the two methods. We will now examine these two methods.

The MOLAP Model In the multidimensional OLAP (MOLAP) model, data for analysis are stored in specialized multidimensional databases. Large multidimensional arrays form the storage structures. For example, to store sales number of 500 units for product ProductA, in month number 2001/01, in store StoreS1, under distributing channel Channel05, the sales number of 500 is stored in an array represented by the values (ProductA, 2001/01, StoreS1, Channel05).

The array values indicate the location of the cells. These cells are intersections of the values of dimension attributes. If you note how the cells are formed, you will realize that not all cells have values of metrics. If a store is closed on Sundays, then the cells representing Sundays will all be nulls.

Let us now consider the architecture for the MOLAP model. Please go over each part of Figure 20-16 carefully.

Note the three layers in the multitier architecture. Precalculated and prefabricated multidimensional data cubes are stored in multidimensional databases. The MOLAP engine in the application layer pushes the multidimensional view of the data from the MDDBs to the users.

Multidimensional database management systems are proprietary software systems. These systems provide the capability to consolidate and fabricate summarized cubes during the process that loads data into the MDDBs from the main data warehouse. The users who need summarized data enjoy fast response times from the preconsolidated data.

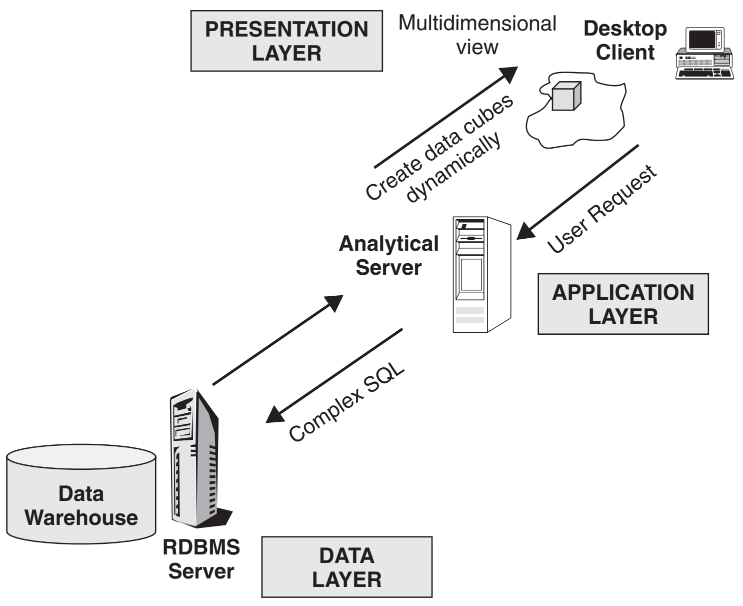

Figure 20-17 The ROLAP model.

The ROLAP Model In the relational OLAP (ROLAP) model, data are stored as rows and columns in the relational format. This model presents data to the users in the form of business dimensions. To hide the storage structure to the user and present data multidimensionally, a semantic layer of metadata is created. The metadata layer supports the mapping of dimensions to the relational tables. Additional metadata support summarizations and aggregations. You may store the metadata in relational databases.

Now take a look at Figure 20-17. This figure shows the architecture of the ROLAP model.

What you see is a three-tier architecture. The analytical server in the middle tier application layer creates multidimensional views on the fly. The multidimensional system at the presentation layer provides a multidimensional view of the data to the users. When the users issue complex queries based on this multidimensional view, the queries are transformed into complex SQL directed to the relational database. Unlike the MOLAP model, static multidimensional structures are not created and stored.

True ROLAP has three distinct characteristics:

- Supports all the basic OLAP features and functions discussed above.
- Stores data in a relational form.
- Supports some form of aggregation.

Local storing of multidimensional cubes is a variation of ROLAP provided by vendors. This is how it works:

The user issues a query.

The results of the query get stored in a small, local multidimensional database.

The user performs analysis against this local database.

If additional data are required to continue the analysis, the user issues another query. And the analysis continues.

Data Mining

Most certainly you have heard about data mining. Most of you know that the technology has something to do with discovering knowledge. Some of you possibly know that data mining is used in applications such as marketing, sales, credit analysis, and fraud detection. All of you know vaguely that data mining is somehow connected to data warehousing. You are right. Data mining is used in just about every area of business from sales and marketing to new product development, inventory management, and human resources.

There are perhaps as many variations in the definition for data mining as there are vendors and proponents. Some experts include a whole range of tools and techniques from simple query mechanisms to statistical analysis as data mining. Others simply restrict data mining to mean knowledge discovery techniques. A workable data warehouse, although not a prerequisite, will give a practical boost to the data mining process.

Why is data mining being put to use in more and more businesses? Here are some basic reasons:

- In today's world, an organization generates more information in a week than most people can read in a lifetime. It is humanly impossible to study, decipher, and interpret all those data to find useful competitive information.
- A data warehouse pools all the data after proper transformation and cleansing into well-organized data structures. Nevertheless, the sheer volume of data makes it impossible for anyone to use analysis and query tools to discern useful patterns.
- In recent times, many data mining tools have appeared in the market that are suitable for a wide range of applications. We are seeing the maturity of the tools and products.
- Data mining needs substantial computing power. Parallel hardware, databases, and other powerful components are becoming very affordable.
- As you are aware, organizations are placing enormous emphasis on building sound customer relationships, and for good reasons. Companies want to know how they can sell more to existing customers. Organizations are interested in determining which of the customers will prove to be of long-term value to them. Companies need to discover any existing natural classifications among their customers so that the classifications may be properly targeted with products and services. Data mining enables companies to find answers and discover patterns in their customer data.
- Finally, competitive considerations weigh heavily on your company to get into data mining. Perhaps your company's competition is already into data mining.

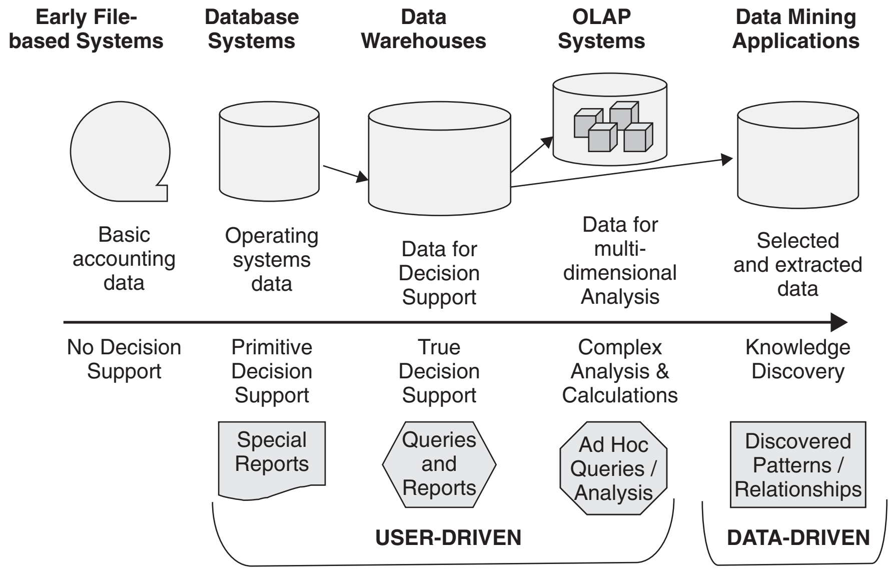

Figure 20-18 Decision support progresses to data mining.

What is Data Mining? Before getting into some formal definitions of data mining, let us try to understand the technology in a business context. Like all decision support systems, data mining delivers information. Please refer to Figure 20-18 showing the progression of decision support.

Note the earliest approach when primitive types of decision support systems existed. Next, we move to database systems providing more useful decision support information. In the 1990s, data warehouses began to be the primary valuable source of decision support information. Query and report tools assist the users to retrieve the types of decision support information they need. For more sophisticated analysis, OLAP tools became available. Up to this point the approach for obtaining information was driven by the users.

But the sheer volume of data renders it impossible for anyone to use analysis and query tools to discern all useful patterns. For example, in marketing analysis, it is almost impossible to think through all the probable associations and to gain insights by querying and drilling down into the data warehouse. You need a technology that can learn from past associations and results and predict customer behavior. You need a tool that will by itself accomplish the discovery of knowledge. You want a data-driven approach and not a user-driven one. This is where data mining steps in and takes over from the users.

Progressive organizations gather the enterprise data from the source operational systems, move the data through a transformation and cleansing process, and store the data in data warehouses in a form suitable for multidimensional analysis. Data mining takes the process a giant step further.

Knowledge Discovery Process Data mining is, therefore, a knowledge discovery process. Data mining discovers knowledge or information that you never knew existed in your data. What about this knowledge? How does it show up? Usually,

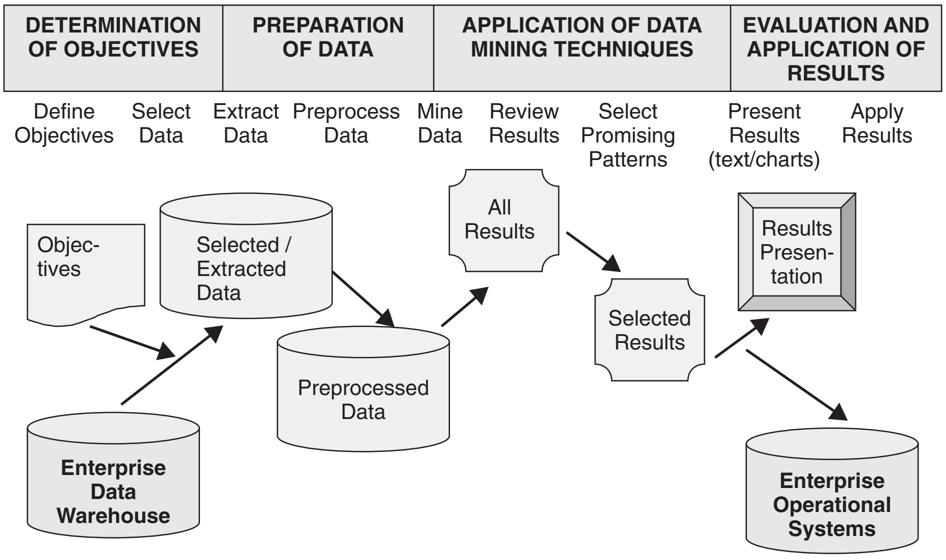
DETERMINATION OF OBJECTIVES
PREPARATION OF DATA
APPLICATION OF DATA MINING TECHNIQUES
EVALUATION AND APPLICATION OF RESULTS
Define Objectives
Select Data
Extract Data
Preprocess Data
Mine Data
Review Results
Select Promising Patterns
Present Results (text/charts)
Apply Results
Objec-tives
Selected / Extracted Data
Enterprise Data Warehouse
Preprocessed Data
All Results
Selected Results
Results Presen-tation
Enterprise Operational Systems

Figure 20-19 Knowledge discovery process.

the uncovered hidden knowledge manifests itself as relationships or patterns. Figure 20-19 illustrates the knowledge recovery process.

OLAP Versus Data Mining As you know, with OLAP queries and analysis, users are able to obtain results from complex queries and derive interesting patterns. Data mining also enables the users to uncover interesting patterns, but there is an essential difference in the way the results are obtained. Figure 20-20 clarifies the basic difference by means of a simple diagram.

When an analyst works with OLAP in an analysis session, the analyst has some prior knowledge of what he or she is looking for. The analyst starts with assumptions deliberately considered and thought out. In the case of data mining, however, the analyst has no prior knowledge of what the results are likely to be.

Users drive OLAP queries. Each query may lead to a more complex query and so on. The user needs prior knowledge of the expected results. It is completely different in data mining.

Data Mining in Data Warehouse Environment Data mining fits well in the data warehouse environment. It plays a significant role in the environment. A clean and complete data warehouse forms the bedrock for data mining. The data warehouse enables data mining operations to take place. The two technologies support each other. These are some of the major factors:

- Data mining algorithms need large amounts of data, more so at the detailed level. Most data warehouses contain data at the lowest level of granularity.
- Data mining flourishes on integrated and cleansed data. If your data extracts from source systems, data transformation functions, and data cleansing proce-

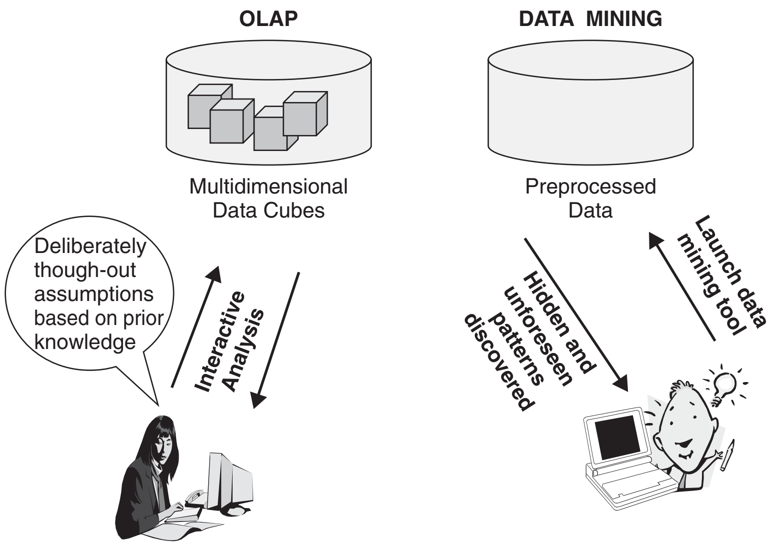
OLAP
DATA MINING
Multidimensional Data Cubes
Preprocessed Data
Deliberately though-out assumptions based on prior knowledge
Interactive Analysis
Hidden and unforeseen patterns discovered
Launch data mining tool

Figure 20-20 OLAP and data mining.

dures were carried out properly, your data warehouse contains data well suited to data mining.

- Already the infrastructure for data warehouses is robust, with parallel processing technology and powerful relational database systems. Because such scalable hardware is already in place, no new investment is needed to support data mining.

Let us point out one difference in the way data are used from the data warehouse for traditional analysis and data mining. When an analyst wants to analyze, say with an OLAP tool, the analyst begins with summary data at a high level. Then the analysis continues through the lower levels by means of drill-down techniques. On many occasions the analyst need not go down to the detail levels. This is because he or she finds suitable results at the higher levels for deriving conclusions. But data mining is different. Because data mining is searching for trends and patterns, it deals with lots of detailed data. For example, if the data mining algorithm is looking for customer buying patterns, it certainly needs detailed data at the level of the individual customer.

So what is a compromise approach? What is the level of granularity you need to provide in the data warehouse? Unless it is a huge burden to keep detailed data at the lowest level of granularity, strive to store detailed data. Otherwise, for data mining engagements, you may have to extract detailed data directly from the operational systems. This calls for additional steps of data consolidation, cleansing, and transformation. You may also keep light summaries in the data warehouse for traditional queries. Most of the summarized data along various sets of dimensions may reside in the OLAP systems.

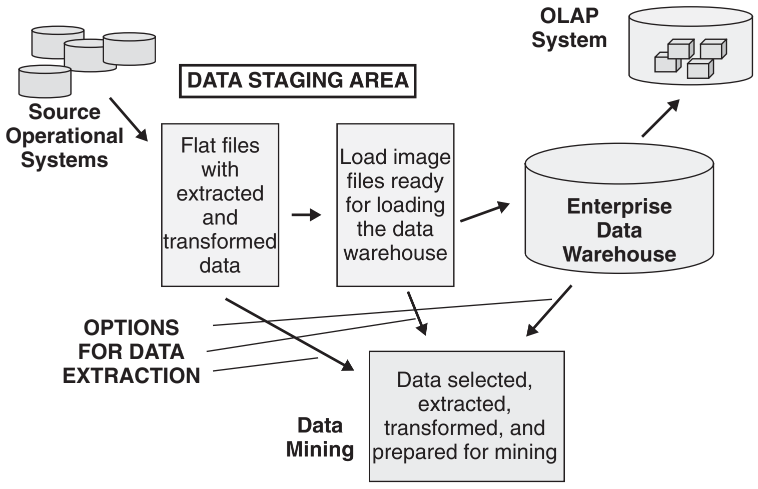
OLAP System
DATA STAGING AREA
Source Operational Systems
Flat files with extracted and transformed data
Load image files ready for loading the data warehouse
Enterprise Data Warehouse
OPTIONS FOR DATA EXTRACTION
Data Mining
Data selected, extracted, transformed, and prepared for mining

Figure 20-21 Data mining in data warehouse environment.

The data warehouse is a valuable and easily available data source for data mining operations. Data extractions for data mining tools to work on come from the data warehouse. Figure 20-21 illustrates how data mining fits in the data warehouse environment.

Note how the data warehouse environment supports data mining. Note the levels of data held in the data warehouse and the OLAP system. Also, observe the flow of data from the data warehouse for the knowledge discovery process.

Major Data Mining Techniques and Applications Figure 20-22 shows examples of application areas, mining functions, mining processes, and techniques.

Using the figure, try to understand the connections on the basis of the following remarks:

- Data mining algorithms are part of data mining techniques.
- Data mining techniques are used to carry out the data mining functions. While performing specific data mining functions, you are involved in getting this done through mining processes.
- A specific data mining function is generally suitable to a given application area.
- Each of the application areas is a major area in business where data mining is actively used now.

LEADING TRENDS: BASIC OVERVIEW

In this section, we consider a few leading trends in database technology. This is not an in-depth coverage of each topic. The intention is just to highlight the main points and expose you to what is out there and what you should be watching for.

Application Area	Examples of Mining Functions	Mining Processes	Mining Techniques
Fraud Detection	Credit card frauds	Determination of variations from norms	Data Visualization
	Internal audits		Memory-based Reasoning
	Warehouse pilferage		
Risk Assessment	Credit card upgrades	Detection and analysis of links	Decision Trees
	Mortgage Loans		Memory-based Reasoning
	Customer Retention		Neural Network
	Credit Ratings		
Market Analysis	Market basket analysis	Predictive Modeling	Cluster Detection
	Target marketing		Decision Trees
	Cross selling	Database segmentation	Link Analysis
	Customer Relationship Marketing		Genetic Algorithms

Figure 20-22 Data mining functions and application areas.

For more detailed study of any trends that pique your interest, please consult the appropriate literature. A few of the books mentioned in the Reference section will give you more information.

Parallel Databases

During the past decade, we have observed the emergence of parallel databases as a significant phenomenon. Several commercial vendors have established the feasibility of splitting large queries and running them in parallel modes. If you look at transaction processing at every organization, the enormous surge in the number and complexity of transactions becomes clear. The last 10–15 years has also experienced a great demand for decision support systems. These systems thrive on complex, multidimensional queries on large data sets. Without parallel hard-ware systems and parallel database software, decision support systems cannot be effective.

Parallel databases run on parallel hardware infrastructure. Three types of parallel architecture are commonly adopted, as illustrated by Figure 20-23.

Shared Memory

Each CPU has full access to the shared memory. Multiple CPUs are interconnected through a common network bus.

Shared Disk

Each CPU or a cluster of CPUs in a node has its own local main memory. Memory is not shared among CPUs. Communication occurs over a high-speed bus. Each node has access to the common set of disks.

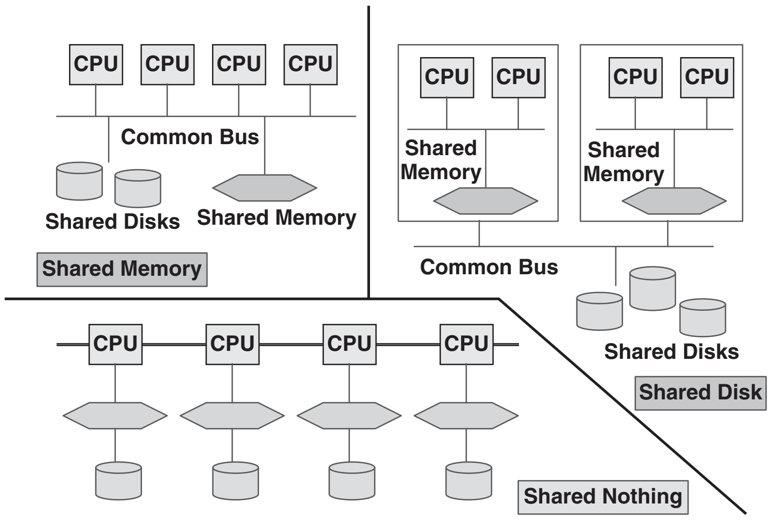

Figure 20-23 Parallel architecture options.

Shared Nothing

Each CPU has its own main memory and set of disks. Works well with an operating system that supports transparent disk access. If a table or database is located on a particular disk, access to it depends entirely on the CPU that owns it. Internode communication is through CPU-to-CPU connection.

Parallel database systems speed up queries and transactions by executing them in parallel. These systems support the following types of parallelism techniques.

I/O Parallelism Database relations are partitioned horizontally or vertically, and the data content is spread across multiple disks. Horizontal partitioning works well for I/O partitioning. You may partition a relation into a number of horizontal partitions and distribute the partitions to disks adopting the following techniques: round-robin, hash distribution, or range partitioning.

When a query or transaction performs a retrieval of a set of rows or writing a set of rows, performing the operation in parallel becomes possible through the I/O parallelism technique. Full table scans, range selections, and value selections work differently for different methods of distribution. For example, the round-robin method is suited for sequential full table scans; value selection works well with hash distribution; range partitioning applies well to range selections.

Interquery Parallelism This form of parallelism increases system throughput. Several queries or transactions execute in parallel. The queries are distributed among several CPUs for running in parallel. Obviously, interquery parallelism cannot be possible with single-processor systems. However, interquery parallelism does not speed up individual queries or transactions. Each query by itself takes as much time as it would in a single-processor system.

Shared-memory architecture works well for interquery parallelism. Supporting interquery parallelism in the other architectures is more difficult.

Intraquery Parallelism This method requires the dividing up of a query into its various operations. Then these individual operations of a single query may be distributed among the processors for execution in parallel. After the completion of all operations, the result sets are assembled together to produce the final result set. This method calls for pipelining the output of one operation as the input for the next.

Intraquery parallelism may be applied to a query in two ways:

Intraoperation parallelism. A transaction may consist of operations such as select, sort, project, or join. Each operation itself may be parallelized. For example, if the transaction has to sort 100,000 records, the sort itself may be split into groups of records and shared by multiple processors.

Interoperation parallelism. You can accelerate the processing of a single query by executing the various operations in parallel. Let us say that a query consists of select, sort, and join. The select operation can run on one processor, the sort on another, and the join on a third.

Hybrid Parallelism This method is a combination of interquery parallelism and intraquery parallelism.

Active Databases

Assume that you have implemented a database for your organization and one of the applications supported by the database is inventory control. Suppose your organization sells a few critical products that contribute to bulk of the revenue. Your marketing vice president is very concerned about ensuring that sufficient inventory of these critical products is always available . How can you satisfy the requirements of this executive?

If your database is a passive database, then periodically you should run queries to determine the inventory levels of these critical products. On the other hand, if your database is an active database, it can automatically alert the marketing vice president when inventory levels fall below prescribed minimum quantities. Active databases do not just store data; they can perform defined actions based on stipulated events. This concept is catching on. Many relational systems support this principle and can be used to behave as active databases.

Events that trigger actions are usually changes to data values in the database. Inserts, updates, and deletes are events that can set into action a particular response. Also, actions can be triggered by temporal events such as the arrival of a certain day or time. When an event occurs, it can initiate application of defined rules. The database system verifies the conditions of the rules and triggers appropriate actions. Rules may be applied for monitoring events. They may also be applied for enforcing integrity constraints.

Active databases work on the event-condition-action paradigm as noted below:

```
ON      event
IF      condition
THEN    action
```

Rules are stored in the database itself. The programs or modules to be executed on the occurrence of the events are also stored in the database. The programs are generally known as triggers or trigger programs. In relational databases, triggers are specialized versions of stored procedures.

Here are a few considerations for designing triggers:

- Ensure that a trigger does not create a chain reaction of triggers.
- Set priority for actions if several actions are possible for the same event.
- Depending on the nature of a particular event, set the trigger to start executing as soon as an event is sensed or immediately after an event completes.
- Address runtime errors that may result from the execution of a trigger.

Intelligent Databases

In the evolution of database systems, as of now, intelligent databases are perhaps the culmination of the search for better methods of storage and access of data. Intelligent databases provide a natural way to use them; they handle seamless movements of huge data volumes; they provide an appropriate set of user tools. Intelligent database systems integrate sophisticated technologies such as object orientation, hypermedia, expert systems, text-based searches, and online information retrieval with database technology.

Intelligent databases offer services at three levels: database engine at the foundational level, user interface at the middle level, and tools at the highest level. Figure 20-24 presents the architecture at these three levels.

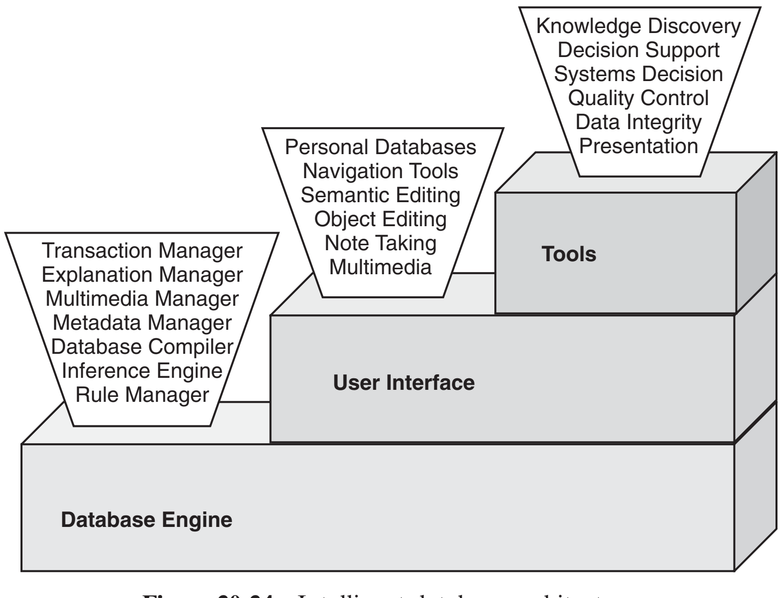

Figure 20-24 Intelligent database architecture.

Deductive Databases

Interest in deductive databases began in research establishments in the late 1970s in an effort to combine logic and database technology. During that period, the promises of artificial intelligence and expert systems attracted a lot of attention. Research projects abounded in Europe, Japan, and the U.S. Japan launched the Fifth Generation Project to encompass a number of technologies including logic systems and database technology. Research institutions proposed several experimental deductive database systems. A few commercial systems were also deployed.

Major objectives that propelled the effort included:

- Extend the relational model and develop a new type of database system.
- Improve the functionality of such a database system to perform as a deductive DBMS.
- In addition, enable the new type of DBMS to support general-purpose applications

Deductive databases combine the functionality of logic programming, methods to specify rules about data, and an inference engine. A declarative language expresses the rules. Logic similar to that used in Prolog language is applied. The inference engine performs as a deduction tool to deduce new facts from the database. Datalog, a variation of Prolog, defines the rules as they relate to relations in the database.

In a deductive database, facts and rules are specified.

Facts. Defined similar to relations in a relational database, except that attribute names are not included. A relation represents a real-world object, and its attributes attempt to describe the object, perhaps only partially. In a deductive database, the position of an attribute within the tuple determines what the value of an attribute means.

Rules. Views in a relational database system define virtual relations, not what is actually stored. In a sense, rules are like views. They specify virtual units that can be derived from the facts by applying inference techniques. Defined similar to data views in a relational database, except that attribute names are not included. Views cannot include recursive definitions; however, rules may involve recursion.

Backward chaining and forward chaining methods are used in deductive database systems to evaluate query or transaction results.

Figure 20-25 presents Prolog notation to express facts, rules, and queries. This is not an explanation of how queries work in the deductive database system—just a sample to indicate the notation.

Multimedia Databases

Traditional database systems store and manage textual and numeric data. The introduction of object-relational database systems meant a big leap forward by the inclusion of new data types. With UDTs, users can define new data types. Still this is not

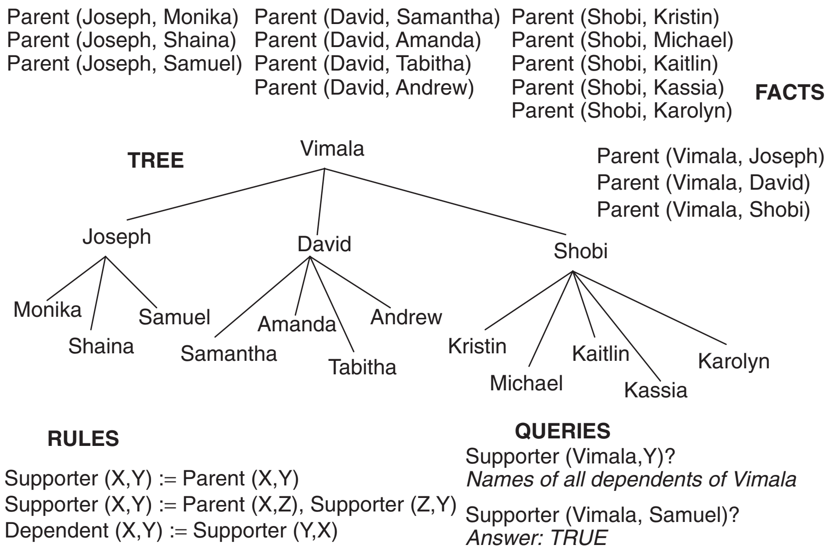

Figure 20-25 Prolog notation: facts, rules, queries.

enough. Modern requirements call for a more inclusive type of database system that can accommodate the ever-increasing newer types of objects.

If the number of multimedia objects is small, you can probably get by if you store the objects in a file and the descriptions in a database and somehow relate the two. Storage and retrieval based on searches can be managed without much difficulty. However, if your organization deals with a large assortment of multimedia objects, as many modern organizations do, then you need a multimedia database system.

Types of Multimedia A multimedia database system stores and manages the following types of objects:

Text. Formatted or free-form character or numeric strings. Includes HTML pages.

Graphics. Diagrams, drawings, and charts in descriptive encoded formats.

Images. Pictures, photographs, scanned images, and so on, digitized in standard formats.

Audio. As bit strings in digitized form. Also includes structured audio with distinct components of note, tone, sound level, and duration.

Video. Sequenced photographs to be presented at precise rates.

Animation. Images or graphs in time sequence.

Mixed media. As for example, synchronized audio and video.

Issues and Challenges Because of the variety of multimedia objects and the potential size of individual objects, multimedia databases are faced with difficult issues and challenges. Here are a few that call for serious consideration.

Modeling and design. Although all types of multimedia may be considered objects, because of the variations, creating a satisfactory database model is difficult. Design issues are also hard to address.

Storage. Even when you have to store a large number of rows in relational database systems, each individual row is of manageable size. On the other hand, multimedia database systems must store and manage individually large objects. Most databases define multimedia objects as a data type called BLOB (binary large object). Storage of large objects requires good compression techniques.

Selection and retrieval. Users must be able to retrieve multimedia objects by specifying selection conditions based on content and similarity. For example, pictures and images that are very similar must be able to be construed as the same object and retrieved.

Query type. Query formulation and execution are different from those in relational database systems, in which they are based on query languages and indexes.

Video-on-demand. Provision of this service for users by dialing into a server is gaining momentum. The server must be able to handle special types of real-time constraints and synchronization of video delivery at server and client.

Applications Although no comprehensive commercial systems are yet available, the following types of applications will greatly benefit from multimedia database systems.

- Real-time alert, control, and monitoring
- Entertainment and travel
- Marketing and advertising
- Education and training
- Knowledge management
- Document storage and retrieval

Mobile Databases

Mobile computing has become pervasive in today's business environment. You see users with handheld devices, portable laptops, and notebooks transmitting data to databases held in remote sites. The FedEx employee, delivering a package at your front door, updates the package-tracking database from his handheld device. The cop who pulled you over on a highway verifies your driver's license against an a central database. The inventory taker at a large warehouse inputs the on-hand quantities into the main inventory database from his handheld computing system. There are many more applications of mobile computing including emergency response services, sales call assistance, and travel information services.

The enormous progress in portable computing devices and tremendous improvements in wireless communications have created a new paradigm of remote access to databases. However, a wide array of issues must be addressed before the new type of remote database access can be fully successful. We will explore the nature of mobile computing, its special needs, and the pertinent database requirements.

Nature of Mobile Computing Mobile computing is not the same as distributed computing. Mobile users keep moving, and along with them moves the computing equipment, whereas in distributed computing, although the computing devices are dispersed, they stay in fixed locations.

The following points highlight the basic nature and considerations of mobile computing.

- Allows users to communicate with one another and perform their functions while on the move.
- Machines accessing the database do not have fixed locations or constant network addresses.
- Machines tend not to stay connected with the database server; they are off-line for long periods. Maintaining transaction consistency becomes difficult.
- Because the battery power of remote units is usually low, receiving information through separate, individual queries is not efficient.
- Most information received by remote units is through broadcasts from the database, on fixed or changeable schedules. Broadcasts are an efficient method because they can reach a large number of remote units.
- Query results sent to remote units are often dependent on the location of the user, not necessarily the current location. For example, a travel information system providing hotel and other such services must be based not on the current location of the user but on future locations.
- Downloaded data are maintained in local files to be used while working offline. Local files need to be kept synchronized with the source data.

Mobile Computing Model and Architecture Because of its nature, a mobile computing model must be based on a distributed architecture. Figure 20-26 presents a typical architecture.

Examine the details shown in the figure and note the descriptions mentioned below.

Fixed host A computing device at a fixed location.

Mobile unit Also called a mobile host, provides remote computing services.

Support station Connects a set of fixed hosts and support stations. Support stations are also known as base stations. Base stations have wireless connections with the mobile units they support.

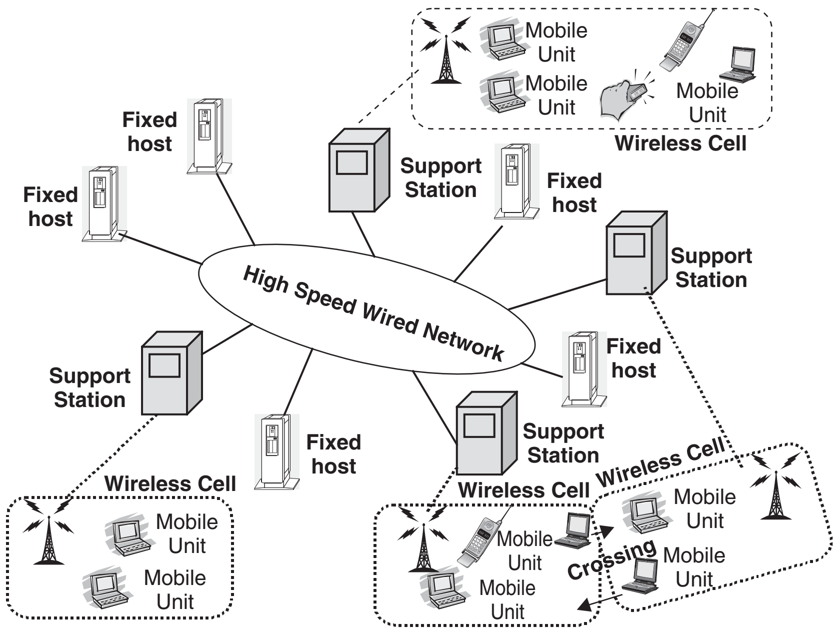

Figure 20-26 Mobile computing architecture.

Wired network Connects a set of fixed hosts and support stations (base stations).

Communication Support stations and fixed hosts are reachable over the wired network. Mobile hosts and support stations communicate through wireless connections.

Downlink Used for sending data from a support station to a mobile unit.

Uplink Used for sending data from a mobile unit to a support station.

Mobility domain The entire domain in which a mobile unit can move.

Cell A part of the mobility domain where wireless connectivity with the controlling support station is possible.

Movements Mobile units move from cell to cell. When a mobile unit moves from one cell to another, a handoff procedure transfers connectivity with the mobile unit from one cell to another.

Access contiguity Guarantees that movement from cell to cell will not affect data access.

Mobile Database Considerations As you can observe, the database considerations for mobile computing relate many different issues not present in centralized or even distributed databases. Note the following considerations.

Types of data content In a mobile computing environment, you can have private data owned, managed, and restricted to a single user. Public data is maintained at one unit, such as stock prices, airline fares, and so on. Public data is available for access by any user. Another type of data content may be read and updated by any user. Product inventory falls in this shared category.

Application types Vertical applications are restricted within a cell. For example, local information about emergency services and accommodation information is valuable only within small areas. On the other hand, horizontal applications are meant for users in all cells.

Data distribution On method is to distribute the data only among the fixed hosts and support stations, using full or partial replication techniques. This is the popular method. The other method is to distribute data among all units—fixed and mobile. Data management is more difficult in this approach.

Data synchronization Frequently, users download relevant data from the common database and keep the downloaded data for off-line work. This adds another dimension to data management. The downloaded data must be kept synchronized with the main database.

Query processing Query processing costs depend on the location of the mobile unit. Another consideration of query processing relates to the requirement that complete and correct query results must be provided even while the mobile unit is moving from cell to cell.

Transaction processing A transaction emanating from a mobile unit may be executed through several support stations and perhaps on different copies of the database fragments. Traditional ACID properties of transactions do not apply to the mobile computing environment.

Recovery Recovery from failures becomes quite involved. Failures at mobile units commonly result from battery failure. Support station failures cause routing readjustments. In this environment, recovery has to address many types of failure—mobile unit, fixed host, support station, network links, wireless links, and transactions themselves.

Geographic Databases

Geographic databases store and manage spatial data. Examples of spatial data in geographic databases include maps with political boundaries and administrative boundaries, digital satellite images showing natural contours, transportation networks, highways, rivers, and so on. These databases also hold another type of data that is nonspatial but related to the spatial data. This second type of data includes census counts, population distribution by economic factors, and sales data.

Geographic databases support a wide variety of applications. They range extensively in scope and use. Your latest automobile is probably fitted with a GPS (Global

Positioning System) unit so that you have no excuse for ever getting lost. This is an application of geographic databases.

Here is a random sample of the types of applications:

- Vehicle navigation
- Distribution of households by demographics
- Market analysis by region
- Consumption of utility services
- Distribution of economic indicators
- Pollution studies
- Water resource management
- Wildlife studies
- Soil evaluation studies
- Landscape surveys
- Parks management
- Crop and produce yield analysis

Geographic Data Formats Although geographic data cover a broad range of objects, they comprises two types. First, geographic data contain information about location or spatial objects. These are rivers, lakes, oceans, land masses, boundaries, regions, cities, towns, and so on. Next, geographic data contain information associated with these spatial units. For example, geographical data contain spatial information about the regions in the north as well as the annual rainfall statistics for these regions.

Let us formalize these two data formats in geographic databases.

Vector format Vector data are created from geometric objects such as points, lines, triangles, squares, and polygons. These are used to represent objects in two dimensions. For example, a river may be represented as a series of line segments, a region as a combination of polygons, and an exit on the highway as a point. For topological information such height and vertical contours, three-dimensional geometric objects such as cylinders, spheres, or polyhedrons may be used.

Raster format Raster data represent information associated with spatial data. Raster data constitute an array or collection of points. Each point represents a value for the information being represented. Suppose that a region is represented by a polygon—a vector format. Now you want to show the income distribution in the region. That would be done with the raster format. If a point in the raster format represents annual income level of 25,000 dollars, then you would need four points to represent 100,000 dollars at the appropriate place in the region. Raster data is kept as n-dimensional arrays where each entry is a unit representing an attribute.

Spatial Query Types How do you use geographic databases? How do you retrieve information from these databases? We are not discussing the different indexing techniques needed for geographic databases. B-tree and bitmapped indexes cannot apply here. However, let us just cover a few types of queries in a geographic database system.

Region query Relates to whole or partial regions. Query may request for objects partially or fully inside a region. Example: Show me the rivers flowing through the state of Louisiana.

Proximity query Requests for objects near a given place. Example: Show me the Italian restaurant nearest to the intersection of Georges Road and Fifth Street.

Intersection and union query Requests of information obtained through intersection or union. Example: Show me the areas in the metropolitan region with low crime rate and low population density.

CHAPTER SUMMARY

- Data abstraction, inheritance, and object identity form the basis for object orientation.
- In object technology, an object is a single, uniquely identifiable entity. Objects having the same attributes and performing the same operations constitute a class. Methods indicate the behavior of an object. Messages are the means for communication between objects.
- Object-oriented database systems address some of the limitations of relational database systems. Object-oriented databases combine database capabilities with object orientation.
- Object-relational databases are emerging from a combination of object orientation and relational database technology.
- A certain set of distinct features makes ORDBMS better than RBDMS and OODBMS: base data type extension, support for complex objects, support for large objects, ability to define new data types and functions, and a flexible rule system.
- Special types of database systems provide information for decision support systems such as data warehousing, OLAP, and data mining.
- A data warehouse is a new type of system environment designed for strategic information. OLAP enables highly multidimensional analysis and complex calculations. Data mining is a knowledge discovery process uncovering patterns and relationships from data.
- Leading trends in database technology: parallel databases, active databases, intelligent databases, deductive databases, multimedia databases, mobile databases, and geographic databases.

REVIEW QUESTIONS

1. Briefly describe the concept of data abstraction in object technology.
2. Describe objects and classes with examples.
3. List the advantages and disadvantages of OODBMS.
4. Define an ORDBMS. What are its major features?

5. What is a STAR schema? How is it applicable as a data model for data warehouses?
6. Explain the need for OLAP. What are the two common OLAP models? Describe briefly any one of the models.
7. How is data mining different from OLAP?
8. What types of parallelism do parallel databases provide?
9. What are multimedia databases? Why are they essential in the modern environment?
10. What are geographic databases? List any six types of applications these databases support.

EXERCISES

1. Indicate whether true or false:
   A. In object orientation, abstract data types provide encapsulation.
   B. Methods and messages are the same in object-oriented systems.
   C. The E-R modeling technique is highly suitable for data warehouses.
   D. Parallel databases work on the event-condition-action paradigm.
   E. Deductive databases combine logic programming and an inference engine.
   F. In a mobile computing environment, fixed hosts are connected to support or base stations.
   G. In geographic databases, raster data are created from geometric objects such as points, lines, and so on.
   H. In an object-relational database environment, users can define their own data types and functions.
   I. Horizontal parallelism works best if data are partitioned across multiple disks.
   J. In a STAR schema, the fact table contains metrics for analysis.
2. Compare and contrast the functions and features of RDBMS, OODBMS, and ORDBMS.
3. Describe the basic components of a data warehouse.
4. Write a short description of data mining. List any four application areas and explain how data mining is useful in these areas.
5. You are a manager of special projects for a large distributor of canned food items. Your company distributes these products to all leading supermarket chains and other convenience stores in the northeast region of the U.S. The region is divided into many districts, each district having a team of salespersons. Plan a mobile computing system to cater to the needs of the traveling sale force. They would need customer, order, product, and inventory information from the proposed mobile database. Prepare a plan outline, considering the overall architecture, data distribution, data access requirements, and other major aspects.