Complied and Combined by
Saagar Kamrani
Chapter 1 The Database Environment

Chapter Overview

The purpose of this chapter is to introduce students to the database approach to information systems development, and to the important concepts and principles of this approach. This is an important chapter because it should convey a sense of the central importance of databases in today’s information systems environment. The idea of an organizational database is intuitively appealing to most students. However, many students will have little or no background or experience with databases. Others will have had some experience with a PC database (such as Microsoft Access), and consequently will have a limited perspective concerning an organizational approach to databases.

In this chapter we introduce the basic concepts and definitions of databases. We contrast data with information, and introduce the notion of metadata and its importance. We contrast the database approach with older file processing systems, and introduce the Pine Valley Furniture Company case to illustrate these concepts. We describe the range of database applications from personal computer databases to enterprise databases and identify key decisions that must be made for each type of database. We describe both the potential benefits and typical costs of using this approach. We conclude the chapter by tracing the historical evolution of database systems.

Chapter Objectives

Specific student learning objectives are included at the beginning of each chapter. From an instructor’s point of view, the objectives of this chapter are to:

1. Create a sense of excitement concerning the database field, and the types of job opportunities that are available.
2. Introduce the key terms and definitions that describe the database environment.
3. Describe data models and how they are used to capture the nature and relationships among data.
4. Acquaint students with the broad spectrum of database applications, and how organizations are using database applications for competitive advantage.
5. Describe the major components of the database environment, and how these components interact with each other.
6. Introduce the Pine Valley Furniture Company case, which is used throughout the text to illustrate important concepts.
7. Introduce the Mountain View Community Hospital case, which is included at the end of each chapter as a source for student projects.
Key Terms

<table>
<thead>
<tr>
<th>Constraint</th>
<th>Database Management System (DBMS)</th>
<th>Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td>Database</td>
<td>Metadata</td>
</tr>
<tr>
<td>Data independence</td>
<td>Enterprise data model</td>
<td>Relational database</td>
</tr>
<tr>
<td>Data warehouse</td>
<td>Enterprise resource planning (ERP) systems</td>
<td>Repository</td>
</tr>
<tr>
<td>Database</td>
<td></td>
<td>User view</td>
</tr>
<tr>
<td>Database application</td>
<td></td>
<td>Entity</td>
</tr>
</tbody>
</table>

Classroom Ideas

1. Start with a discussion of how students interact with databases on a daily basis (credit card transactions, shopping cards, telephone calls, cell phone contact lists, downloadable music, etc.). If you teach in a classroom with computers, ask students to find examples of websites that appear to be accessing databases.

2. Contrast the terms, data and information. Using Figure 1-1 as a starting point, have the students provide some good examples of data and information from their own experiences. This may well lead to some differences of opinion, and the conclusion that one person’s data may be another person’s information.

3. Introduce the concept of metadata using Table 1-1. Ask the students to suggest other metadata that might be appropriate for this example.

4. Introduce data models using Figure 1-2. Discuss the differences between an enterprise data model and a project data model, using Figure 1-2 (a) and (b).

5. Discuss file processing systems and their limitations, using Figure 1-3 and Table 1-2. Emphasize that many of these systems are still in use today.

6. Introduce the database approach and its advantages, using Figure 1-4.

7. Introduce the students to the major components of the database environment (Figure 1-5). Stress the interfaces between these components and the fact that they can “make or break” a database implementation.

8. Discuss the range of database applications (personal computer to enterprise), using Figures 1-6 through 1-8 and Table 1-6. Ask your students to give other examples of each of these types of databases.

9. Introduce the concept of a data warehouse as a type of enterprise database. This topic is described in detail in Chapter 11.

10. Discuss each of the advantages of the database approach (Table 1-3). Stress that these advantages can only be achieved through strong organizational planning and commitment. Also discuss the costs and risks of the database approach (Table 1-4).

11. Review the evolution of database technologies and the significance of each era (Figure 1-9). Add your own perspective to the directions that this field is likely to take in the future.

12. If time permits, have the students answer several problems and exercises in class.

13. Use the project case to reinforce concepts discussed in class. Students can be assigned to work on this case in class if time permits, or it can be used as a homework assignment.

14. If time permits, use Teradata University Network resources to demonstrate the structure and contents of a relational database for some of the textbook datasets. Demonstrate, or lead students through, some simple SQL retrieval exercises against the textbook databases.
Answers to Review Questions

1. Define each of the following key terms:
   a. **Data.** Stored representations of objects and events that have meaning and importance in the user’s environment.
   b. **Information.** Data that have been processed in such a way as to increase the knowledge of the person who uses it.
   c. **Metadata.** Data that describes the properties or characteristics of end-user data and the context of that data.
   d. **Database application.** An application program (or set of related programs) that is used to perform a series of database activities (create, read, update, and delete) on behalf of database users.
   e. **Data warehouse.** An integrated decision support database whose content is derived from the various operational databases.
   f. **Constraint.** A rule that cannot be violated by database users.
   g. **Database.** An organized collection of logically related data.
   h. **Entity** A person, place, object, event, or concept in the user environment about which the organization wishes to maintain data.
   i. **Database management system.** A software system that is used to create, maintain, and provide controlled access to user databases.

2. Match the following terms and definitions:
   - c data
   - b database application
   - l constraint
   - g repository
   - f metadata
   - m data warehouse
   - a information
   - j user view
   - k database management system
   - h data independence
   - e database
   - i enterprise resource systems planning (ERP)
   - d enterprise data model

3. Contrast the following terms:
   a. **Data dependence; data independence.** With data dependence, data descriptions are included with the application programs that use the data, while with data independence the data descriptions are separated from the application programs.
   b. **Data warehouse; data mining.** A data warehouse is an integrated decision support database, while data mining (described in the chapter introduction) is the process of extracting useful information from databases.
   c. **Data; information.** Data consist of facts, text, and other multimedia objects, while information is data that have been processed in such a way that it can increase the knowledge of the person who uses it.
d. **Repository; database.** A repository is a centralized storehouse for all data definitions, data relationships, and other system components, while a database is an organized collection of logically related data.

e. **Entity; enterprise data model.** An entity is an object or concept that is important to the business, while an enterprise data model is a graphical model that shows the high-level entities for the organization and the relationship among those entities.

f. **Data warehouse; ERP system.** Both use enterprise level data. Data warehouses store historical data at a chosen level of granularity or detail, and are used for data analysis purposes, to discover relationships and correlations about customers, products, and so forth that may be used in strategic decision making. ERP systems integrate operational data at the enterprise level, integrating all facets of the business, including marketing, production, sales, and so forth.

4. Five disadvantages of file processing systems:
   a. Program-data dependence
   b. Duplication of data
   c. Limited data sharing
   d. Lengthy development times
   e. Excessive program maintenance

5. Two ways to convert data to information:
   a. Put data in context by providing structure.
   b. Summarize or process and present them for human interpretation.
6. Five categories of databases are:

<table>
<thead>
<tr>
<th>Category</th>
<th>Definition</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Personal database</td>
<td>- Designed for one user, may be located on desktop or laptop PC, PDAs, or other handheld devices</td>
<td>- Set of data describing patient visits, recorded by a home health-care professional on a handheld device</td>
</tr>
<tr>
<td>Workgroup database</td>
<td>- Designed for a small team of people (&lt; 25) collaborating on projects or applications</td>
<td>- A database that supports the work of several scientists performing research on a new drug</td>
</tr>
<tr>
<td>Department database</td>
<td>- Designed for a functional unit in an organization and services between 25 to 100 people</td>
<td>- A database used by the human resources department of a large hospital</td>
</tr>
<tr>
<td>Enterprise database</td>
<td>- Designed to support the entire organization or enterprise for operations and decision-making; usually supports multiple departments</td>
<td>- The database supporting the SAP (or JD Edwards, etc.) enterprise information system</td>
</tr>
<tr>
<td>Web-enabled database</td>
<td>- Designed to support Web browser access to applications and databases</td>
<td>- The database used by Amazon.com to support customer purchases of books and other items</td>
</tr>
</tbody>
</table>

7. The definition of data has been expanded in today’s environment because databases (and Web servers) are commonly used to store more complex data such as documents, images, and video clips.

8. Following are nine major components in a typical database system environment:
   a. CASE tools: automated tools used to design databases and database applications.
   b. Repository: centralized storehouse of data definitions.
   c. Database management system (DBMS): commercial software used to define, create, maintain, and provide controlled access to the database and the repository.
   d. Database: organized collection of logically related data.
   e. Application programs: computer programs that are used to create and maintain the database.
   f. User interface: languages, menus, and other facilities by which users interact with the various system components.
   g. Data administrators: persons who are responsible for the overall information resources of an organization.
   h. System developers: persons such as systems analysts and programmers who design new application programs.
   i. End users: persons who add, delete, and modify data in the database and who request information from it.
9. Relationships between tables are expressed by identical data values stored in the associated columns of related tables in a relational database.

10. Some key questions for each type of database are the following:
   a. Personal database:
      1. Should the application be purchased or developed internally?
      2. If developed internally, should it be developed by an end user or by an IS professional?
      3. What data are required by the user and how should the database be designed?
      4. What commercial DBMS product should be used for the application?
      5. How should data in the personal database be synchronized with data in other databases?
      6. Who is responsible for the accuracy of the data?
      7. What physical security precautions can be taken to safeguard the data stored in this database?
   b. Workgroup database:
      1. How can the design of the database be optimized for the various group members?
      2. How can the group members use the database concurrently without compromising database integrity?
      3. Which data processing operations should be performed at a client workstation and which on the server?
      4. What data rights (create, read, update, delete) should be assigned to the users of this database?
   c. Department databases:
      1. How can the database and its environment be designed for adequate performance?
      2. How can adequate security be provided to protect against unauthorized disclosure or distribution of sensitive data?
      3. What database and application development tools should be used?
   d. Enterprise databases:
      1. How should the database be distributed among the various locations?
      2. How can the organization develop and maintain acceptable data standards?
      3. What actions must be taken in order to successfully integrate numerous systems, including legacy data from earlier systems that are desired for analysis?
      4. How can adequate security be provided to protect against unauthorized disclosure or distribution of sensitive data?
   e. Web-enabled databases:
      1. What type of security is required for external access to a business database?
      2. How will the database deal with different types of client browsers and hardware platforms?
11. Data independence refers to the separation of data descriptions from the application programs that use the data. It is an important goal because it allows an organization’s data to change and evolve without changing the application programs that use the data. Additionally, data independence allows changes to application programs without requiring changes in data storage structure.

12. Potential benefits of the database approach are:
   a. Program-data independence
   b. Minimal data redundancy
   c. Improved data consistency
   d. Improved data sharing
   e. Increased development productivity
   f. Enforcement standards
   g. Improved data quality
   h. Improved data accessibility and responsiveness
   i. Reduced program maintenance
   j. Improved decision support

13. Five additional costs or risks of the database approach are:
   a. New, specialized personnel
   b. Installation, management cost, and complexity
   c. Conversion costs
   d. Need for explicit backup and recovery
   e. Organizational conflict

14. Evolution of database technology:
   a. 1960s – traditional files.
   b. 1970s – first generation; hierarchical and network databases.
   c. 1980s – second generation; relational databases.
   d. 1990s – third generation; object-oriented and object-relational databases.

15. Internet technologies would benefit Pine Valley Furniture in several ways.
   a. Customer order entry
      The implementation of an order entry application utilizing the Internet would greatly improve Pine Valley’s customer base, streamline the order process, and improve employee efficiency.
   b. Inventory and supplier orders
      In addition to the order entry application, Pine Valley could also consider utilizing the Internet to exchange information with suppliers. For example, when supply levels reach a certain threshold, an order for materials could be prepared and electronically submitted to suppliers.

Some possible problems which might arise:
   a. Additional complexity of systems
      In comparison to an in-house system, you have no way of guaranteeing a persistent connection with an Internet user. Thus, applications would need to be written in such a way to account for this.
b. User expectations
   Since the Internet is available 24 hours a day, seven days a week, customers would expect that they could place orders anytime. This could present difficulties if the system is not always available.

c. Security
   There could also be the added risk of data security being compromised via the Internet.

16. Pine Valley Furniture Company (PVFC) uses a database management system to support its operational functions but this database is not structured in a way that supports timely analysis of trends or historical patterns. PVFC can benefit from a data warehouse that is appropriately structured for questions related to vendor pricing and/or customer order patterns over time. A data warehouse would enable PVFC to summarize data drawn from various operational databases (i.e., personal, workgroup, department, and ERP) into meaningful structures for timely decision-making access.

17. While the Internet is available to anyone, an intranet is available only to employees of a company, most often only from PCs that are located on-site. An intranet does utilize the same protocols as the Internet and can be accessed via a browser.

   An extranet also uses Internet protocols, however, it also allows limited access from the outside into the company’s data. Most often, companies will use this to allow vendors and customers limited access to data.

18. Several trends will continue during the next decade:
   a. Capability to manage increasingly complex data types.
   b. Continued development of universal servers.
   c. Trend toward centralization of databases will continue.
   d. Content-addressable data.
   e. Database access for untrained users will become much easier.
   f. Improved synchronization of small databases.
   g. Increased use of data warehouses, likely leading to more data mining.
   h. Self-tuning database management systems will improve database performance.
   i. Further development of computer forensics.

19. Very large databases are being used to improve customer relationship management (CRM) by creating CRM systems that react to individual customer’s purchase behavior. For example by suggesting other items that a customer may want to purchase based on that customer’s previous purchases. They are also being used to improve employee relationship management by tracking employee skills and sending notice when an internal job opportunity that needs a particular skill that the employee possesses is announced. Online shopping sites are able to carry a large virtual inventory stored in a database for the customer to peruse.

20. For this exercise, have students research sites for vendors such as Oracle, SqlServer, mySQL, DB2, Informix, etc.
Solutions to Problems and Exercises

1. Examples of relationships:
   a. Many-to-Many:
      ![Diagram of Many-to-Many relationship between STUDENT and COURSE](image1)
   b. One-to-Many:
      ![Diagram of One-to-Many relationship between BOOK and BOOK COPY](image2)
   c. One-to-Many:
      ![Diagram of One-to-Many relationship between COURSE and SECTION](image3)
   d. One-to-Many:
      ![Diagram of One-to-Many relationship between ROOM and SECTION](image4)
   e. Many-to-Many:
      ![Diagram of Many-to-Many relationship between INSTRUCTOR and COURSE](image5)

2. In this database, the relationship between CUSTOMER and CONTACT HISTORY is one-to-many:
   ![Diagram of One-to-Many relationship between CUSTOMER and CONTACT HISTORY](image6)

3. Advanced data types have several special requirements:
   a. Storage requirements – multimedia objects (such as images, sound, and video clips) require substantial storage capacity, which needs to be justified.
   b. Content management – this is the problem of storing, locating, and retrieving the multimedia objects. This process requires specialized software not generally available in a relational DBMS or extra effort to create a means to rapidly access multimedia objects (such as keyword indexes).
c. Maintenance – while conventional relational data are easily updated, multimedia objects may require maintaining multiple versions of the data. Usually the whole object needs to be restored because it is treated as a whole rather than a set of parts.

4. Metadata for Class Roster:

Please note that some columns have been omitted in order to save space.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Source</th>
<th>Created</th>
<th>Updated</th>
<th>Responsible Party</th>
</tr>
</thead>
<tbody>
<tr>
<td>Course</td>
<td>Alphanumeric</td>
<td>Course ID and name</td>
<td>Academic Unit</td>
<td>5/10/2004</td>
<td>6/1/2004</td>
<td>Registrar</td>
</tr>
<tr>
<td>Section</td>
<td>Integer</td>
<td>Section number</td>
<td>Registrar</td>
<td>5/10/2004</td>
<td></td>
<td>Registrar</td>
</tr>
<tr>
<td>Semester</td>
<td>Alphanumeric</td>
<td>Semester and year</td>
<td>Registrar</td>
<td>5/10/2004</td>
<td></td>
<td>Registrar</td>
</tr>
<tr>
<td>Name</td>
<td>Alphanumeric</td>
<td>Student name</td>
<td>Student IS</td>
<td>8/07/2003</td>
<td></td>
<td>Student IS</td>
</tr>
<tr>
<td>ID</td>
<td>Integer</td>
<td>Student ID (SSN)</td>
<td>Student IS</td>
<td>8/07/2003</td>
<td></td>
<td>Student IS</td>
</tr>
<tr>
<td>Major</td>
<td>Alphanumeric</td>
<td>Student major</td>
<td>Student IS</td>
<td>8/07/2003</td>
<td>11/15/2003</td>
<td>Student IS</td>
</tr>
<tr>
<td>GPA</td>
<td>Decimal</td>
<td>Student grade point average</td>
<td>Academic Unit</td>
<td>8/07/2003</td>
<td>5/10/2004</td>
<td>Department Chair</td>
</tr>
</tbody>
</table>

5. Some limitations of databases:
   a. Why do organizations create multiple databases?
      There are several reasons. First, because of resource limitations, organizations fund development of their information systems one application at a time. Second, organizations may acquire some of their information systems from outside vendors. This also results in a proliferation of databases. Third, mergers and acquisitions generally result in multiple databases.
   b. What organizational and personal factors lead an organization to have multiple, independently managed databases?
      Perhaps the most common reason is that end-user groups develop their own database applications, rather than wait for the central IS organization to develop a centralized database. Also the pressures associated with rapid business change result in organizations taking a short-term, suboptimal approach rather than a careful, long-term strategy.

6. This is a good in-class interactive exercise for individuals or small groups. For individuals, have each student choose a student club, fraternity/sorority, or other organization to illustrate a “top-down” approach to develop an enterprise data model. For small groups, divide the class into groups and have each group work to develop an enterprise data model for a club, fraternity/sorority, or other organization. Re-convene as a large class to compare/contrast each of the small group enterprise data models. Identify the similarities and differences through class discussion.

7. Typical decisions regarding databases:
   a. Personal databases: Who is responsible for the accuracy of the data?
      Answer: the end user who uses this database is responsible.
   b. Workgroup database: Which data processing operations should be performed at a workstation and which should occur on the server?
Answer: data input and (when possible) data validation should be performed on the workstation. Database queries and updates should be performed on the server.

c. Departmental/divisional database: How can adequate security be provided to protect the data?
   Answer: a layer of security procedures must be designed by database administration and implemented in the department database system. These procedures will include password protection and authorization rules that restrict user access and actions they may take. Other procedures may include authentication schemes, which positively identify potential users, and encryption (or encoding) schemes.

d. Enterprise database: How can the organization develop and maintain adequate data standards?
   Answer: this is a difficult goal whose achievement requires top management commitment and the establishment of a strong, proactive data administration group (see Chapter 12).

e. Web-enabled database: With applets and code modules on different servers and browsers, how can all components have a shared understanding of the meaning of data?
   Answer: all database connectivity is done through a common interface, typically ODBC or JDBC, which enables an application to access data on any type of server without having to know the implementation details of the platform.

8. Use of ERP systems running on database platforms allows companies to significantly decrease the time it takes to ship an order. Information from manufacturing and inventory can be integrated with orders placed through the Web, and packing slips produced automatically. The eCommerce application that accepts the order records each order along with customer and shipping information into a database that can be accessed for inventory and production control planning.

9.
   a. structured data
   b. metadata; fact describing property
   c. unstructured data
   d. unstructured data
   e. structured data
   f. metadata; fact describing context
   g. metadata; fact describing context

10.
   a. one-to-many
   b. many-to-many
11. Suggested metadata for the customer entity:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Length</th>
<th>Min</th>
<th>Max</th>
<th>Description</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>CustomerID</td>
<td>Alphanumeric</td>
<td>10</td>
<td></td>
<td></td>
<td>Customer ID</td>
<td>OE System</td>
</tr>
<tr>
<td>Name</td>
<td>Alphanumeric</td>
<td>30</td>
<td></td>
<td></td>
<td>Customer Name</td>
<td>Customer</td>
</tr>
<tr>
<td>Street</td>
<td>Alphanumeric</td>
<td>25</td>
<td></td>
<td></td>
<td>Customer Street</td>
<td>Customer</td>
</tr>
<tr>
<td>City</td>
<td>Alphanumeric</td>
<td>20</td>
<td></td>
<td></td>
<td>Customer City</td>
<td>Customer</td>
</tr>
<tr>
<td>State</td>
<td>Alphanumeric</td>
<td>2</td>
<td></td>
<td></td>
<td>Customer State</td>
<td>Customer</td>
</tr>
<tr>
<td>ZipCode</td>
<td>Alphanumeric</td>
<td>5</td>
<td></td>
<td></td>
<td>Customer Zip</td>
<td>Customer</td>
</tr>
<tr>
<td>Telephone</td>
<td>Alphanumeric</td>
<td>14</td>
<td></td>
<td></td>
<td>Customer Phone</td>
<td>Customer</td>
</tr>
<tr>
<td>StartDate</td>
<td>Date</td>
<td>8</td>
<td></td>
<td></td>
<td>Date customer started doing business with company</td>
<td>OE System</td>
</tr>
<tr>
<td>CreditLine</td>
<td>Decimal</td>
<td>5</td>
<td>0.0</td>
<td>10000.00</td>
<td>Customer’s Credit Line</td>
<td>OE System</td>
</tr>
</tbody>
</table>

12. One suggested approach would be to create an enterprise database to contain all information about customers, policies, etc. The need for an enterprise database is clear, since policy information would need to be accessed not just by the sales team but also by the actuarial department and the claims department. For inside agents, access to the database would be through an intranet, utilizing a browser-based application as the front-end. Each outside agent would have a personal database on his or her notebook computer with only information for his or her territory. The personal database would then be synchronized periodically with the enterprise database through the use of an extranet.

13. Some common data elements are: patient ID, patient LastName, patient FirstName, patient MI, staff ID, staff LastName, staff FirstName, staff MI, Hospital ID, Hospital Name, Hospital Street Address, Drug code, drug name, drug generic name, drug side effect.
15. Some suggestions to assist students would be to research business journals as well as perform searches on the Internet using search engines such as Google.

16. a. one-to-many  
    b. one-to-many  
    c. There could be a relationship between customer and store. It would be useful if the customer had never purchased a pet, so for example the store could send mailings to prospective customers.

Suggestions for Field Exercises

1. You can accomplish this exercise either by arranging a field trip for your class (preferred), or by inviting the IS manager (or other key IS person) to visit your class. Following are some of the key steps to perform:
   a. Select an organization. We suggest a mid-sized manufacturing company or a familiar organization in the service sector such as a hospital or bank.  
   b. Identify several mainstream applications such as human resource management, material requirements planning, and financial accounting.  
   c. Determine whether these applications were predominantly developed internally, or purchased from an outside vendor or vendors.  
   d. Now determine the mix between file processing and database processing for these applications. If a database approach is used, is the data shared among the applications?  
   e. To draw a figure depicting the files and databases, inquire whether the organization has system flowcharts (or similar documentation) that portray much of this information.

2. This exercise is most easily performed as a continuation of Exercise 1. Arrange to interview a database administrator or key designer as part of the same field trip. Discuss whether the organization maintains user-oriented metadata, or only technical metadata. Where is this metadata maintained: within individual applications, in one or more CASE tool repositories, or elsewhere?

3. This exercise is also most easily performed as a continuation of Exercise 1. If you interview the database administrator, that person can answer the questions in this exercise as well. Be prepared to provide the person you interview with a definition of each type of database (see the summary in Table 1-6). Also be prepared to define what you mean by a data warehouse (see Chapter 11)—that term is used rather loosely today.

4. If the student has selected an established company with a fairly extensive information systems department, they will find that the company is either already very involved in Web-enabling some or many parts of their business, or that they are actively planning to expand the scope of their involvement. You may want to encourage your students to use this question to explore where the company sees itself going with regard to the Internet. Many older companies are struggling very hard to adjust and remain competitive as the Internet dramatically affects business models.
5. One of the important benefits of this exercise is that it encourages students to develop habits of self-study, which are important in a rapidly growing and changing field. We suggest you consider asking your students to maintain the journal in electronic form, perhaps by posting their notes to a Web site.

Project Case

Case Questions

1. A well-managed database might help the hospital achieve its mission by:
   a. Tracking operating costs, and identifying costs that are rising or are above average for similar hospitals.
   b. Providing an electronic medical record for each patient, so that physicians can quickly assemble all relevant information.
   c. Developing historical records of various treatment programs, to identify which programs and procedures are most effective (and which are not).
   d. Providing data and information to track important measures such as average length of patient stay, cost per day for a patient, and so on.

2. Database technology can provide several mechanisms to ensure that data are protected and that the hospital conforms to HIPAA standards. The following table is taken from the standards and highlights some areas that database technology can help with regarding compliance.

<table>
<thead>
<tr>
<th>Standards</th>
<th>Implementation Specifications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Security and Training</td>
<td>Login Monitoring</td>
</tr>
<tr>
<td></td>
<td>Password Management</td>
</tr>
<tr>
<td>Contingency Plan</td>
<td>Data Backup Plan</td>
</tr>
<tr>
<td>Device and Media Controls</td>
<td>Data Backup and Storage</td>
</tr>
<tr>
<td>Access Control</td>
<td>Encryption</td>
</tr>
<tr>
<td>Transmission Security</td>
<td>Encryption</td>
</tr>
</tbody>
</table>

3. Some of the costs and risks of using databases that the hospital must manage carefully are:
   a. New, specialized personnel. This will include database designers, database administrators, and other skills that are in short supply in a tight job market.
   b. Installation and management cost and complexity. This may include installing a new DBMS as well as hardware and operating system upgrades.
   c. Conversion costs. These are primarily the costs of converting older applications to a database environment. Alternatively, the hospital may choose to purchase new systems from a vendor.
d. Need for explicit backup and recovery. These are the system costs associated with backup and recovery procedures.

e. Organizational conflict. The hospital must anticipate the costs of data administration and other activities related to developing a consensus on data definitions, ownership, and maintenance.

4. Data quality is of utmost importance in a hospital environment. This is particularly true when considering applications such as patient care administration and clinical services, since poor data quality could directly affect patient care. Less critical, but still important, is data quality for financial management and administrative services.

5. There are several complex data types that must be managed by a hospital:
   a. Medical scans (MRI, X-ray, etc.)
   b. Clinical test data (blood test, etc.)
   c. Documents (admission forms, physician referrals, etc.)
   d. Unstructured text (doctors’ orders, doctors’ notes, etc.)
   e. Test results (electrocardiograms, ultrasound, etc.)

6. The PATIENT and PATIENT CHARGES tables are linked by the Patient_Number attribute, which is common to both tables. Thus, for example, patient no. 4238 has three separate charges at the present time.

7. Mountain View Community Hospital could use personal databases that reside on PDAs for physicians to have access to databases that contain information on prescribed treatment for various diseases. Also, physicians could enter notes into a database on the PDA which could then be later uploaded to a patient information system. Workgroup databases could be utilized to contain information specific to one ward, for example an extensive database of cancer treatments. Also, workgroup databases might be utilized to collect information for epidemiological studies being conducted by one group of researchers. Departmental databases could be utilized to store records that are only pertinent to that department, but might not become part of the patient’s full medical record. For example, radiology might store all images from a patient x-ray or MRI, while only the ones used for diagnosis would be contained in the medical record. Also, radiology might keep a database of images to help radiologists in diagnosis. An enterprise database could be created to store all information about a patient, such as the medical record, billing, etc. Web-enabled databases could be utilized as described in Question 8.

8. Mountain View Community Hospital could use Web-based applications in several ways. Perhaps one of the greatest areas that it could use this technology would be to establish an Intranet for access to databases by internal hospital personnel. Also, the hospital could investigate an extranet application to do third-party billing directly with insurance companies. Another application would be to enable access to medical databases and prescription drug databases that are available online. Web-based applications have several benefits, including streamlining such processes as third-party billing. One major
risk associated with any Web-based application is security. Even having data available via an Intranet could still pose a security risk to the hospital if network security did not prevent unauthorized access from the outside.

Case Exercises

1. Relationship between PATIENT and PATIENT_CHARGES:

```
PATIENT  Billed_for  Billed_To  PATIENT_CHARGES
```

2. Metadata chart:

<table>
<thead>
<tr>
<th>Data Item</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong></td>
<td><strong>Type</strong></td>
</tr>
<tr>
<td>Patient_Name</td>
<td>Character</td>
</tr>
<tr>
<td>Patient_Number</td>
<td>Integer</td>
</tr>
<tr>
<td>Patient_Address</td>
<td>Character</td>
</tr>
<tr>
<td>Item_Description</td>
<td>Character</td>
</tr>
<tr>
<td>Item_Code</td>
<td>Integer</td>
</tr>
<tr>
<td>Amount</td>
<td>Decimal</td>
</tr>
</tbody>
</table>

**Note to instructor:** Student answers may vary as there are many other attributes that could be listed in this solution.

3. Patient Bill view:

<table>
<thead>
<tr>
<th>Item Code</th>
<th>Item Description</th>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>Room Semi-Priv</td>
<td>1600</td>
</tr>
<tr>
<td>275</td>
<td>Radiology</td>
<td>150</td>
</tr>
<tr>
<td>700</td>
<td>EEG Test</td>
<td>200</td>
</tr>
</tbody>
</table>

Published by www.emagzine.org
4. E-R diagram (simplified) for hospital:

The simplified ER diagram completed for exercise 4 suggests these relationships.

5.

a.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Length</th>
<th>Min</th>
<th>Max</th>
<th>Description</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Employee_ID</td>
<td>Integer</td>
<td>4</td>
<td>1</td>
<td>9999</td>
<td>Employee ID</td>
<td>Human Resources</td>
</tr>
<tr>
<td>Date_Hired</td>
<td>Date</td>
<td>8</td>
<td></td>
<td></td>
<td>Date of Hire</td>
<td>Human Resources</td>
</tr>
<tr>
<td>Employee_Type</td>
<td>Text</td>
<td>15</td>
<td></td>
<td></td>
<td>Type of Employee</td>
<td>Human Resources</td>
</tr>
</tbody>
</table>

b. 1:M Physician to Patient
M:M Patient to Charge

The simplified ER diagram completed for exercise 4 suggests these relationships.
c. The Patient table will contain information about a patient, such as Patient_ID, Patient_LastName, Patient_FirstName, Patient_Middle_Initial. The Patient table is likely to have stored the Physician_ID value for the physician who admitted the patient. In order for this data to be useful for human interpretation, one would need to be able to link the Patient table with other tables, such as the Physician table, Charge table, etc.

d. A sample report might look like the following:

<table>
<thead>
<tr>
<th>Patient Name</th>
<th>Physician</th>
<th>Admit Date</th>
<th>Care Center</th>
<th>Room</th>
</tr>
</thead>
<tbody>
<tr>
<td>Terri Smith</td>
<td>Larry Moxley</td>
<td>Feb. 24, 2006</td>
<td>Cancer Care Center</td>
<td>100</td>
</tr>
<tr>
<td>Chris Bailey</td>
<td>Larry Moxley</td>
<td>Mar 1, 2006</td>
<td>General Care Center</td>
<td>102</td>
</tr>
</tbody>
</table>

Total Patients: 2

6. SELECT *
FROM PATIENT
WHERE Patient_LastName > ‘A’;

Project Assignments

P1. Some examples of entities are:

- PRESCRIPTION – this entity would track prescription drugs that are stocked and prescribed to patients.
- INSURANCE_CARRIER – this entity would contain information about insurance carriers, including address information.
- ITEM - this entity would contain information on items, such as specialized equipment, which might be consumed by patients.
- LABORATORY - this entity would contain information on laboratory tests.

P2. Some other views would be:

- Treatment View – what patients received which treatments.
- Physician View - what patients were treated and/or admitted by which physicians.
- Laboratory View – which patients were tested.
- Pharmacy View – what drugs were prescribed to which patients.
Nurse_Supervisor View – how are patients assigned to nurses.
Care Center View  - which patients are in a particular care center. What staff have been assigned to the care center.
Lab Supervisor View - which technicians performed certain tests.

P3. Preliminary Enterprise Data Model for Mountain View Community Hospital; this two entity segment is presented as an example of the kind of diagram a student might develop. Student answers may vary; what is important is that the entities are appropriate for the case and that the preliminary cardinalities are drawn correctly for the situation.
Chapter 2 The Database Development Process

Chapter Overview

The purpose of this chapter is to introduce students to the database development process within the broader context of information systems development. The chapter presents an expanded description of the systems development life cycle and the role of database development within it. It also presents a description of the prototyping methodology and its impact on database development. The chapter introduces the role of packaged data models and how they are used to simplify the database development process. Chapter 2 presents an updated description of the well-known three-schema architecture and uses it to summarize the various deliverables of database development. The chapter continues to emphasize the information engineering methodology in database development, including the role of the enterprise data model.

Chapter Objectives

Specific student learning objectives are included at the beginning of the chapter. From an instructor's point of view, the objectives of this chapter are to:
1. Illustrate some of the database design and processing skills your students will have after completing a database course using this text.
2. Provide an idea of how to structure a database development project sufficient to begin on a course exercise.
3. Provide a comprehensive overview of various concepts and issues in database management. These will be developed in more detail in later chapters.
4. Discuss the goal of enterprise-wide computing and the need for strategic information systems planning in achieving this goal.
5. Introduce the Zachman Information Systems Architecture and its role in information systems development.
6. Provide a review of systems development methodologies, particularly the systems development life cycle, prototyping, and agile software development; build an understanding of how database development fits with these methodologies.
7. Build an awareness of CASE tools and how they can be used to facilitate the database development process.
8. Describe how packaged data models can be used to shorten the development process and improve the quality of data models.
9. Make students aware of the three-schema architecture and its benefits for database development and design.
10. Develop an understanding of the different roles involved in a database development team.
Key Terms

<table>
<thead>
<tr>
<th>Agile software development</th>
<th>Functional decomposition</th>
<th>Physical schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business function</td>
<td>Incremental commitment</td>
<td>Project</td>
</tr>
<tr>
<td>Client/server architecture</td>
<td>Information engineering</td>
<td>Prototyping</td>
</tr>
<tr>
<td>Computer-aided software engineering (CASE)</td>
<td>Information systems life cycle (SDLC)</td>
<td>Systems development architecture (ISA)</td>
</tr>
<tr>
<td>Conceptual Schema</td>
<td>Logical schema</td>
<td>Top-down planning</td>
</tr>
<tr>
<td>Enterprise data modeling</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Classroom Ideas

1. You can discuss in class the types of systems analysis that might have been done to design the application illustration in this chapter. This is a good opportunity to discuss the linkage between your database course and the systems analysis and design course(s) your students take. You can also mention that a tool like Microsoft Access® is excellent for system prototyping since the database, screens, views, and reports shown in this chapter can be developed easily and quickly. If you are interested in teaching materials on SA&D that are consistent with this database text, the fifth edition of *Modern Systems Analysis and Design*, authored by Hoffer, George, and Valacich, is available from Prentice Hall.

2. Discuss the importance of enterprise-wide computing in today's competitive environment and the difficulties of sharing information given the proliferation of personal computers and databases.

3. Introduce the Zachman Information Systems Architecture Framework. We find in discussing this model it is especially helpful to discuss the different roles associated with each component. An excellent reference to the framework is provided at the Framework Software, Inc. Web site; the link to the PDF is:


4. Discuss the major phases in the information engineering methodology (Tables 2-1 and 2-2). Illustrate sample deliverables from each of the phases using the Pine Valley Furniture Company case.

5. If feasible, provide an in-class demonstration of a CASE tool that supports the information engineering methodology.

6. Review the steps in the database development process (Figure 2-5). Indicate in which row the output for each step relates to the Information Systems Architecture framework as described in the text.

7. Discuss the importance of strategic information systems planning with your students. Have your students give examples of poor results when planning is ignored. Also discuss factors that often interfere with effective planning.

8. Discuss the role of CASE tools and a repository in information systems development. If appropriate, find out what CASE tools your students use in their work environment and their experience with these tools.
9. A quick in-class demo of Microsoft Access or similar product may be used to give the students an initial exposure to an RDBMS and demonstrate a prototyping approach to database development. Consider using the PVFC prototyping request as an example.
10. Your students may have examples from their workplaces to contribute about client/server architectures. Or, you may provide them with an understanding of where the RDBMS software and their data will be stored at your school as an illustration.

Answers to Review Questions

1. Define each of the following key terms:
   a. Information systems architecture (ISA) A conceptual blueprint or plan that expresses the desired future structure for the information systems in an organization.
   b. Systems development life cycle (SDLC) A traditional methodology used to develop, maintain, and replace information systems.
   c. Agile software development An approach to database and software development that emphasizes individuals and interactions over processes and tools, working software over comprehensive documentation, customer collaboration over contract negotiation, and response to change over following a plan.
   d. Client/server architecture A local area network-based environment in which database software on a server (called a database server or database engine) performs database commands sent to it from client workstations, and application programs on each client concentrate on user interface functions.
   e. Incremental commitment A strategy in systems development projects in which the project is reviewed after each phase and continuation of the project is re-justified in each of these reviews.
   f. Enterprise data model The first step in database development, in which the scope and general contents of organizational databases are specified.
   g. Conceptual data model (or schema) A detailed, technology-independent specification of the overall structure of organizational data.
   h. Logical data model (or schema) The representation of data for a particular data management technology (such as the relational model). In the case of a relational data model, elements include tables, columns, rows, primary and foreign keys, as well as constraints.
   i. Physical data model (or schema) A set of specifications that detail how data from a logical data model (or schema) are stored in a computer’s secondary memory for a specific database management system. There is one physical data model (or schema) for each logical data model.

2. Match the following terms and definitions:
   f   conceptual schema
   h   business function
   d   prototyping
   i   systems development life cycle
   b   functional decomposition
   g   top-down planning
3. Contrast the following terms:

a. **Internal schema; conceptual schema**  
   An internal schema consists both of a physical schema and a logical schema; a conceptual schema consists of a single view of the data, combining different external views. When you design the logical schema and the physical schema, it is possible that you will encounter inconsistencies or other issues with the conceptual schema or user views; it is possible to cycle back through these design steps.

b. **Systems development life cycle; prototyping**  
   Both are systems development processes. The SDLC is a methodical, highly structured approach which includes many checks and balances. Consequently, the SDLC is often criticized for the length of time needed until a working system is produced, which occurs only at the end of the process. Increasingly, organizations use more rapid application development (RAD) methods, which follow an iterative process of rapidly repeating analysis, design, and implementation steps until you converge on the system the user wants. Prototyping is one of them. In prototyping, a database and its applications are iteratively refined through a close interaction of systems developers and users.

c. **Top-down planning; functional decomposition**  
   Top-down planning approaches have the advantages of a broad perspective, an ability to look at integration of individual system components, an understanding of the relationship of information systems to business objectives, and an understanding of the impact of information systems across the whole organization. Functional decomposition is a classical process employed in systems analysis in order to simplify problems, isolate attention, and identify components.

d. **Enterprise data modeling; information engineering**  
   In many organizations, database development begins with enterprise data modeling, where the range and general contents of organizational databases are set. An information system’s architecture is developed by information systems planners following a particular methodology for IS planning. One such formal and popular methodology is information engineering.

e. **Repository; computer-aided software engineering**  
   CASE tools automate some portions of the development process and thereby improve system development time and the quality of the delivered products. A repository helps systems and database analysts achieve a seamless integration of data from several CASE tools.

f. **Enterprise data model; conceptual data model**  
   In an enterprise data model, the range and contents of the organizational databases are set. Generally, the enterprise data model represents all of the entities and relationships. The conceptual data model extends the enterprise data model further by combining all of the various user views and then representing the organizational databases using ER diagrams.

g. **Prototyping; Agile software development**  
   Prototyping is a rapid application development (RAD) method where a database and its application(s) are iteratively refined through analysis, design, and implementation cycles with systems developers and end users. Agile software development is a method that shares an emphasis on iterative development with the prototyping method yet further emphasizes the people and rapidity of response in its process.
4. Information engineering steps:
   a. **Planning phase**: development of strategic information systems plans that are linked to strategic business plans. This stage results in developing an enterprise model that includes a decomposition of business functions and a high-level entity-relationship diagram.
   b. **Analysis phase**: Current business situations and information systems are studied to develop detailed specifications for the information systems required to support the organization.
   c. **Design phase**: transformation of the information models developed during analysis to models that conform to the target technology.
   d. **Implementation phase**: construction and installation of the information systems by creating database definitions and applications in the languages of the target systems.

5. Information engineering planning phase steps:
   a. **Identify strategic planning factors**: The purpose of this step is to develop the planning context and to link information systems plans to the strategic business plans.
   b. **Identify corporate planning objects**: The corporate planning objects define the business scope. The scope limits subsequent systems analysis and where information system changes can occur.
   c. **Develop enterprise model**: A comprehensive enterprise model consists of a functional breakdown (or decomposition) model of each business function, an enterprise data model, and various planning matrices.

6. Three information engineering strategic planning factors:
   The strategic planning factors are **organization goals**, **critical success factors**, and **problem areas**. These factors help information system managers to set priorities to address requests for new information systems, and hence the development of databases. For example, the problem area of inaccurate sales forecasts might cause information system managers to include additional historical sales data, new market research data, or data concerning results from test trials of new products in organizational databases.

7. Five key corporate planning objects:
   a. **Organizational units**, which are the various organizational departments
   b. **Organizational locations**, which are the places at which business operations occur
   c. **Business functions**, which are related groups of business processes that support the mission of the organization (Note that business functions are different from organizational units; in fact, a function may be assigned to more than one organizational unit. For example: product development, a function, may be the joint responsibility of the sales and manufacturing departments.)
   d. **Entity types**, which are major categories of data about the people, places, and things managed by the organization
   e. **Information systems**, which are the application software and supporting procedures for handling sets of data

Published by [www.emagazine.org](http://www.emagazine.org)
8. Functional decomposition:

Functional decomposition is a classical process employed in systems analysis in order to simplify problems, isolate attention, and identify components. Often, many databases are necessary to handle the full set of business and supporting functions, whereas a particular database may support only a subset of the supporting functions. It is, however, helpful to have a total, high-level enterprise view so that redundancy of data can be minimized.

9. Using information system planning matrices:

Planning matrices are used to show interrelationships between planning objects. Relationships between data entities and the other organizational planning objects can be represented at a high level by the planning matrices, which can be manipulated to understand patterns of relationships.

10. Five SDLC phases:
   a. **Planning**
      
      **Purpose:** To develop a preliminary understanding of the business situation and how information systems might help solve a problem or make an opportunity possible
      
      **Deliverable:** A written request to study the possible changes to an existing system; the development of a new system that addresses an information systems solution to the business problems or opportunities
   
   b. **Analysis**
   
      **Purpose:** To analyze the business situation thoroughly to determine requirements, to structure those requirements, and to select among competing system features
      
      **Deliverables:** The functional specifications for a system that meets user requirements and is feasible to develop and implement
   
   c. **Design**
   
      **Purpose:** To elicit and structure all information requirements; to develop all technology and organizational specifications
      
      **Deliverables:** Detailed functional specifications of all data, forms, reports, displays, and processing rules; program and database structures, technology purchases, physical site plans, and organizational redesigns
   
   d. **Implementation**
   
      **Purpose:** To write programs, build data files, test and install the new system, train users, and finalize documentation
      
      **Deliverables:** Programs that work accurately and according to specifications, documentation, and training materials
   
   e. **Maintenance**
   
      **Purpose:** To monitor the operation and usefulness of a system; to repair and enhance the system
      
      **Deliverables:** Periodic audits of the system to demonstrate whether the system is accurate and still meets needs
11. Database development activities occur in every phase of the SDLC. Actual database development is most intense in the design, implementation, and maintenance steps of the SDLC.

12. Prototyping steps:
   
   **Identify problem**
   a. Conceptual data modeling
   b. Analyze requirements
   c. Develop preliminary data model

   **Develop initial prototype**
   a. Logical database design
   b. Analyze requirements in detail
   c. Integrate database views into conceptual data model

   **Physical database design and creation**
   a. Define new database contents to DBMS
   b. Decide on physical organization for new data
   c. Design database processing programs

   **Implement and use prototype**
   a. Database implementation
   b. Code database processing
   c. Install new database contents, usually from existing data sources

   **Revise and enhance prototype**
   a. Database maintenance
   b. Analyze database to ensure it meets application needs
   c. Fix errors in database

   **Convert to operational system**
   a. Database maintenance
   b. Tune database for improved performance
   c. Fix errors in database

13. Procedures and processes that are common to SDLC, prototyping, and agile methodologies include:

   - Understanding and analyzing the customer’s business requirements for the system;
   - Translating the customer’s requirements into specifications (logical & physical) for systems development;
   - Developing databases and software programs to meet specifications; and
   - Implementing an operational system.

The methodologies are considered to be different not because of what is done, but because the timing of the methodologies differ. The SDLC methodology is methodical and thorough which makes it well-suited for systems that populate and revise databases. Prototyping, with its rapidly repeating analysis, design, and implementation phases, is well-suited for systems that retrieve data and for helping to refine a customer’s requirements for a new system.
Chapter 2

Agile software development emphasizes quick responses and rests on high-involvement from knowledgeable customers. Agile software development is well-suited to projects with unpredictable and/or rapidly changing requirements and responsible developers (per text citation of Fowler, 2005).

14. Differences between user views, a conceptual schema, and an internal schema:
A conceptual schema defines the whole database without reference to how data are stored in a computer’s secondary memory. A user view (or external schema) is also independent of database technology, but typically contains a subset of the associated conceptual schema, relevant to a particular user or group of users (e.g., an inventory manager or accounts receivable department). An internal schema consists of both a physical schema and a logical schema. A logical schema consists of a representation of the data for a type of data management technology. For example, if the relational model is the technology used, then the logical schema will consist of tables, columns, rows, primary keys, foreign keys and constraints. A physical schema contains the specifications for how data from a logical schema are stored in a computer’s secondary memory.

15. External and conceptual schema design:
External schemas are not necessarily developed before the conceptual schema. In fact, they are typically developed iteratively. Often, a first cut at the conceptual schema is based on the organization’s enterprise data model and the general understanding of database requirements on a project. Then external schemas (user views) are developed for each transaction, report, screen display, and other system uses.

16. Three-tiered database architecture:
Database architecture that allows the data for a given information system to reside in multiple locations or tiers of computers. The purpose is to balance various organizational and technical factors. Also, the processing of data may occur at different locations in order to take advantage of the processing speed, ease of use, or ease of programming on different computer platforms. Although four (and even more) tiers are possible (that is, data on a desktop or laptop microcomputer, workgroup server, department server, a corporate mainframe), three tiers are more commonly considered:
1. **Client tier**  A desktop or laptop computer, which concentrates on managing the user-system interface and localized data—also called the presentation tier
2. **Department (or workgroup) minicomputer server tier**  Performs calculations and provides access to data shared within the workgroup—also called the process services tier
3. **Enterprise server (minicomputer or mainframe) tier**  Performs sophisticated calculations and manages the merging of data from multiple sources across the organization—also called the data services tier

17. Yes, it is possible. The end user machine (in the client tier) — a P.C., for example — might have presentation logic but no database installed on it.
18. Information systems architecture (ISA) components include:
   1. *Data*, which can be represented as an Enterprise Data Model, and provides a high-level view of what will be included in an organization’s databases
   2. *Processes*, which manipulate data (represented by data flow diagrams, object-models with methods, or other notations) and indicate how data may be used in the organization
   3. *Network*, which transports data around the organization or between the organization and its key business partners (shown by a schematic of the network links and topology) and indicates likely interfaces for the organization’s databases with other systems or external organizations
   4. *People*, who perform processes and are the source and receiver of data and information; people can be shown on process models as senders and receivers of data; people will need security and authorization rights to the organization’s databases
   5. *Events and points in time*, when processes are performed (shown by state-transition diagrams and other means) that indicate key parts of business rules governing the data stored in databases
   6. *Reasons*, for events and rules that govern the processing of data (often shown in textural form, but some diagramming tools exist for rules, such as decision tables) and reflect the meaning of the business rules that govern the data stored in databases.

The ISA provides a high-level plan for the information systems in an organization, including a broad view of how the data and information resources will be managed. All of the 6 components of the ISA touch on elements of an organization’s database design at some level. However, the Data component of the ISA is the most central to an organization’s database design efforts. An organization’s enterprise data model supports an organization’s overall data architecture. An organization’s overall data architecture sets the stage for re-use of data throughout the organization as new application requirements are identified and acted upon by systems developers.

19. Entity-relationship diagrams may be drawn as early as during enterprise data modeling, when the range and general contents of organizational databases are determined. They are also drawn during the conceptual design phase. The level of abstraction varies at each stage, with the enterprise-level entity-relationship diagram being more abstract and of broader scope.

20. Three-schema architecture:
   a. external schema
   b. conceptual schema
   c. internal schema

21. Enterprise data models are less detailed; they capture the major categories of data and the relationship between data. They are comprehensive to the whole organization. Project data models are more detailed and more closely match the way that data must be included in the database. Enterprise data models are used to cover the scope of data covered in the database. Project-level data models are used in the process of developing a database and applications which will access the database.
Answers to Problems and Exercises

1. The representation of the systems development life cycle has changed from the original waterfall metaphor. While it is a more compact representation, there are still some problems. For example, it is not purely linear. Also, it is possible to conduct steps in parallel due to time overlaps. One additional problem is the inability to go back from one step to another without completing the entire five-step process.

2. 

<table>
<thead>
<tr>
<th>Data Entity Types</th>
<th>Business Function</th>
<th>Product</th>
<th>Order</th>
<th>Work Center</th>
<th>Equipment</th>
<th>Employee</th>
<th>Raw Material</th>
<th>Customer</th>
<th>Invoice</th>
<th>Work Order</th>
</tr>
</thead>
<tbody>
<tr>
<td>Order fulfillment</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Finance and accounting</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Production operations</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Order shipment</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Materials management</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Sales summarization</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Business planning</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Product development</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

X = data entity (column); is used within business function (row)

The matrix is ordered from those business functions that use the most data entities to those that use the fewest. The entities are also in order from those used in the most business functions to those used in the fewest. Now, if a given function has a priority in systems development, we could easily spot it on the matrix and find out whether the data entities it uses are to be accessed by many or just a few other functions.

3. EMPLOYEE, SUPPLIER, and SHIPMENT might be good examples since all of them represent major categories of data about the entities managed by the organization.

4. 

4a. 

<table>
<thead>
<tr>
<th>Facilities</th>
<th>Capacity planning</th>
<th>Construction/Alteration</th>
<th>Maintenance/security</th>
<th>Allocation/scheduling</th>
</tr>
</thead>
</table>

Published by www.emagazine.org
The representation of functional decomposition may be achieved by following the example below:

```
Curriculum
  - Curriculum design
  - Class scheduling

Instruction
  - Course development
  - Course instruction
  - Performance evaluation

Student life
  - Admissions
  - Course evaluation
  - Registrations
  - Student activities
  - Student advising

Personnel, etc.
```

```
Instruction
  - Course
    - Teach Classes
    - Evaluate Performance
  - Evaluation
```
4b.
4c.

<table>
<thead>
<tr>
<th>FUNCTION</th>
<th>ENTITY TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Other Employees</td>
</tr>
<tr>
<td>Capacity Planning</td>
<td>C</td>
</tr>
<tr>
<td>Construction/Alteration</td>
<td></td>
</tr>
<tr>
<td>Maintenance/Security</td>
<td>U</td>
</tr>
<tr>
<td>Allocation/Scheduling</td>
<td>U</td>
</tr>
<tr>
<td>Curriculum Design</td>
<td></td>
</tr>
<tr>
<td>Class Scheduling</td>
<td>U</td>
</tr>
<tr>
<td>Course Instruction</td>
<td></td>
</tr>
<tr>
<td>Course Evaluation</td>
<td></td>
</tr>
<tr>
<td>Admissions</td>
<td></td>
</tr>
<tr>
<td>Registrations</td>
<td></td>
</tr>
<tr>
<td>Student Activities</td>
<td>U</td>
</tr>
<tr>
<td>Student Advising</td>
<td>U</td>
</tr>
</tbody>
</table>

d. CSFs: Maintain academic standards; Maintain good quality of student life; Hire outstanding faculty; Maintain quality work life for faculty; Maintain current curriculum; Provide quality services within budgetary constraints.

e. Considerations for using a multi-tier architecture: Since much of the data are updated from a large number of different functions (as shown on the matrix in part c), network traffic will be an issue of crucial importance. Processing close to the source data could reduce network traffic. Client technologies however, can be mixed (personal computers with Intel or Motorola processors, network computers, information kiosks, etc.) and yet, share common data. In addition, you can change technologies at any tier with limited impact on the system modules on other tiers. All this will allow for data consistency and maintaining academic standards — a critical success factor for the academic unit.
5.

a. This is a sample list of information systems which could be employed by a fictional student organization:
   - Event Management System
   - General Management Support System
   - Publicity Support System
   - Financial Management System
   - Cost Accounting System, etc.

   The Event Management System could assist the club management in several ways: event production plan preparation (outline the steps necessary to implement the program proposal); setup and catering details arrangement; staffing plan and contingency plan preparation; contract arrangements on performer's pay, travel, and any special requirements; arrangement for technical requirements; event production and evaluation.

b. Sample enterprise data model for fictional student organization

**Notes to instructor:**

1) This data model solution can be used in conjunction with the matrix solutions in parts c and d to stimulate class discussion. For instance, in the data model, we show a STUDENT entity. In the matrices, the STUDENT column is quite empty. Will non-member student data be collected by the organization? If not, do we need the STUDENT entity?

2) This data model solution may also be used to sensitize students to the importance of an organization’s business rules. Suggested questions to discuss with students include:
   - What data would be kept in LOCAL ORGANIZATION?
   - Is STAFF/FACULTY MEMBER data different from STUDENT or MEMBER data for this situation?
   - Does the organization really track who attends a lecture?
   - What reports can be generated from the OFFICER – STUDENT relationship?
   - Can a STUDENT be an OFFICER and not a MEMBER?
c.

<table>
<thead>
<tr>
<th>Information Systems</th>
<th>Data Entity Types</th>
<th>Student</th>
<th>Staff / Faculty Member</th>
<th>Member</th>
<th>Local Organization Officers</th>
<th>Lecture</th>
<th>Guest Speaker</th>
<th>National Convention</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event Management System</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Publicity Support System</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Financial Management System</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cost Accounting System, etc.</td>
<td></td>
<td>R</td>
<td></td>
<td>U</td>
<td>U</td>
<td>U</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

d.

<table>
<thead>
<tr>
<th>Information Systems</th>
<th>Data Entity Types</th>
<th>Student</th>
<th>Staff / Faculty Member</th>
<th>Member</th>
<th>Local Organization Officers</th>
<th>Lecture</th>
<th>Guest Speaker</th>
<th>National Convention</th>
</tr>
</thead>
<tbody>
<tr>
<td>Financial Management System</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Publicity Support System</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cost Accounting System, etc.</td>
<td>R</td>
<td></td>
<td>U</td>
<td>U</td>
<td>U</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Event Management System</td>
<td></td>
<td></td>
<td>C, U, D</td>
<td>C, U, D</td>
<td>C, U, D</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Certain entities would be accessed by the majority of information systems (e.g., MEMBER, OFFICERS, LOCAL ORGANIZATION), while certain information systems may operate on data about the majority of entities. This new rearrangement of the matrix could help us understand that some of the applications would need to access most of the data in a single
session, which would mean that more powerful hardware has to be set up for them. Also, some priorities have to be assigned to the different systems in the case of simultaneous data access attempts.

6. We could rearrange the matrix in order to group the functions and data into subsystems. These subsystems will help us in subdividing the overall development into manageable projects. Since three of the business functions provide the bulk of the use of five of the data entities, these functions would need to have easy access to a large portion of all the data.

7.

<table>
<thead>
<tr>
<th></th>
<th>Data Entity Types</th>
<th>Information Systems</th>
<th>Customer</th>
<th>Product</th>
<th>Raw Material</th>
<th>Order</th>
<th>Work Center</th>
<th>Invoice</th>
<th>Equipment</th>
<th>Employee</th>
</tr>
</thead>
<tbody>
<tr>
<td>Order Tracking System</td>
<td>R</td>
<td>R</td>
<td>R,U, D</td>
<td>R,U</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Plant Scheduling</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R,U</td>
<td>R,U</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Payroll</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R,U</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sales Management</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Inventory Control</td>
<td>R</td>
<td>R,U, D</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Production Scheduling</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The management information systems (e.g., Sales Management System) will tend to retrieve information, while the transaction processing systems will perform frequent updates. The only exception to this might be the case of Inventory Control, which would make changes to the inventory.
8. Even though order fulfillment and product development are both considered business functions, they are not overlapping in nature. Since the two diagrams represent two different subsets, none of the sub-functions would appear on both figures at the same time.

9. Enterprise data models and conceptual data models:
   Enterprise data modeling results in a total picture or explanation of organizational data, not in the design for a particular database. A particular database provides the data for one or more information systems, whereas an enterprise data model, which may encompass many databases, describes the scope of data maintained by the organization. Therefore, the level of abstraction associated with an enterprise data model is higher; conceptual data models will be more detailed.

10. Returning to logical database design during physical database design and creation:
    Database development activities occur in each of the SDLC phases, and feedback may occur which causes a project to return to a prior phase. SDLC activities may find missing elements or errors when designing specific transactions, reports, displays, and inquiries. When a missing element is noticed, for example, it will be necessary to revisit the logical database design.

11. Contrasting database development during conceptual data modeling and logical database design phases:
    It is often said that conceptual data modeling is done in a top-down fashion, driven from a general understanding of the business area, not from specific information processing activities. **Logical database design** approaches database development from two perspectives. First, the conceptual data model is transformed into a standard notation through normalization, based on relational database theory. Then, as each computer program in the information system is designed — including the program’s input and output formats — a detailed review of the transactions, reports, displays, and inquiries supported
by the database is performed. This bottom-up analysis verifies exactly what data are to be maintained in the database and the nature of those data as needed for each transaction, report, and so forth. During logical database design, you combine or integrate the original conceptual data model (more general information) along with the individual user views (more specific information) into a comprehensive design.

12. Prototype development:
Workgroup and department databases are often developed in combination by end users, systems professionals working in business units, and central database professionals. A combination of people is necessary since a wide variety of issues must be balanced in the design of shared databases: processing speed, ease of use, differences in data definitions, and the like. Often, the prototype being developed will be located on a separate workgroup computer or a department computer, where those involved on the project will have easy access to the database and no likelihood of interfering with the organization's production databases.

13. User views of organizational data:
A good approach in developing this problem for a bank might be to carefully select the views to be developed (similar to the example of Figure 2-1) by collecting a transaction slip, monthly statement (representing each type of account), statement of earnings, etc. Examples of data included in each message are customer information, bank information, and transaction data (checks, deposits, service charges, maintenance fees, overdraft protection fees, and so forth). Statement and deposit slip views are given below, as is the combined conceptual data model. Combining the different views could lead to the addition of new attributes or possibly entities and relationships not being shown in the original views.

Statement view:
14. A typical example of potential data duplication in this case would be customer information. Customer data and customer invoices reside on the Accounting Department Server (departmental database). Customer data and their order information reside on the Order Entry Department Server (a machine that hosts another departmental database). The cash flow analyst's PC hosts a local database storing data about customer receipts. If different machines hold different addresses for the same customer, for example, loss of data integrity may be experienced. Data formats could also be incompatible. In the relational model, associations are established by defining foreign keys, which are redundant data, so the inclusion of redundant data does not violate the principles of the database approach. Normalization though, should result in the elimination of unnecessary data redundancy, which does exist in the hypothetical example given here. Coordination of the metadata at the enterprise server will prevent uncontrolled data duplication, limited data sharing, lengthy
development time, and excessive program maintenance. It may also be desirable to intentionally include limited additional redundancy to improve database performance.

15. The single line connection indicates a one-to-one relationship between the entities. In business terms: each order is billed on exactly one invoice, each invoice is written for only one order. In contrast, many payments can be made on one order. Remember to model for the exceptions.

16. 
   a. Each product line name can be up to 40 characters and spaces in length. This variable length is specified by the data type, VARCHAR, and the value in parentheses, 40.
   b. PRODUCT_ID is specified as required in Figure 2-14a (SQL) by the NOT NULL constraint that is included after the data type identification for PRODUCT_ID. This means that for each instance of product information entered, PRODUCT_ID must have a value. This is necessary because PRODUCT_ID is the primary key of the relation. There must be a value entered in the primary key field for each instance. The relation will be indexed and stored based on this value.
   In Figure 2-14b (MS-ACCESS), PRODUCT_ID is also indicated to be the primary key by the small key icon to the left of its specification. The Required property in the bottom half of the screen is set to YES for the same reasons as described above.
   c. The last line establishes a relationship between the PRODUCT table and the PRODUCT_LINE table by designating PRODUCT_LINE_NAME as a foreign key that points back to PRODUCT_LINE_NAME in the PRODUCT_LINE table.

17. 
   a. Sales_to_Date is calculated by multiplying the quantity by the price.
   b. Eliminating "Home Office" from the Criteria line under Product_Line_Name would result in all product line names being displayed, not just the Home Office line.

18. 
   a. The following entities will be needed: Payment, Invoice, Order, Order_Line, Product, Customer
   b. All of the above entities will be needed for the SQL query.

19. 
   a. PRODUCT_LINE_ID is used as the primary key in the Pine Valley database.
   b. Product_Line_Name is used as the primary key in the draft of Helen’s database.
   c. Helen’s version assumes that each product line will have a unique name. Helen’s database also includes two attributes that are not included in Pine Valley Furniture’s database, PL_Prior_Year_Goal and PL_Current_Year_Goal. One conclusion that could be drawn from this is that these two attributes have been generated through her department’s budgeting efforts and are tracked through her department, rather than through the organization-level database. This information is very important to Helen’s management activities and demonstrates the reason why personal databases of this sort are so common.
20.

<table>
<thead>
<tr>
<th></th>
<th>All Entities?</th>
<th>All Attributes?</th>
<th>Technology Independent?</th>
<th>DBMS Independent?</th>
<th>Record Layouts?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enterprise</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>Conceptual</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>Logical</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Physical</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
</tbody>
</table>

**Suggestions for Field Exercises**

1. Students will probably find prototyping being used for smaller database applications. Prototyping is often the systems development methodology of choice where visual programming tools such as Visual Basic, Java, Visual C++, and fourth-generation languages are used because of the ease with which the user-system interface can be modified. Large systems that require a new database are more likely to use the SDLC, requiring a team of information systems professionals, including database designers and programmers. The systems development process for Web applications can be similar, with prototyping being employed. However, students may find that there is a lot more testing involved with Web applications, since one has less control over the client interface.

2. Those organizations which do have a formally recognized information systems architecture are likely to rely on an information systems architecture model developed and integrated with the help of CASE tools. When data format changes occur, or new data is added to the old database — possibly obtained from external data sources such as market research data, general economic indicators, or industry standards — those CASE tools could inform the developers about possible inconsistencies and ensure quality data throughout the organization. This problem is likely to be an eye opener for students if they each investigate a different organization and compare their findings, as many organizations will not have a formally recognized ISA. They may begin to comprehend the differences that exist among organizations as well as the difference between textbook presentations of best practices and actual organization practices.

3. Again, as in the previous field exercise, this exercise will help students to integrate their new textbook knowledge with the realities that organizations must face. Some students will find organizations that are struggling to achieve client/server architecture and also deal with their legacy systems, for example.

4. Students are likely to find some of the following roles on the information systems development teams in larger organizations:
   a. Systems analysts, who analyze the business situation and identify the need for information and information services to meet the problems or opportunities of the business
   b. Database analysts, who concentrate on determining the requirements and design for the database component of the information system
c. Users, who provide assessment of their information needs and monitor appropriateness of the developed system

d. Programmers, who design and write computer programs that have embedded in them commands to maintain and access data in the database

e. Database and data administrators, who have responsibility for existing and future databases and ensure consistency and integrity across databases; as experts on database technology, they provide consulting and training to other project team members

f. Other technical experts, with facility in networking, operating systems, testing, and documentation

g. A senior systems or database analyst, who will be assigned to be project leader, who is responsible for creating detailed project plans as well as staffing and supervising the project team; this good project leader will possess leadership, management, customer relations and communications, technical problem solving, conflict management, team building, and risk and change management skills.

In smaller organizations, students may find that these roles are combined and performed by fewer people because the smaller organizations do not have the additional staff to assign to a diverse team. Thus, fewer people will cover these same functional responsibilities by each person having multiple roles to fulfill. Many employees in small organizations develop deep skill sets in programming, data administration, or technical database administration expertise.

5. **(Note to instructor:** See Figure 2-5 and chapter section on “Managing the people involved in database development” for more background on this exercise.)

If a CASE tool is being used, a repository stores descriptions of the data entities and the business rules, detailed descriptions of CSFs, and objectives. The information systems department analyzes all of them. Throughout the systems development process, CASE tools are used to develop data models and to maintain (in the repository) the metadata for the database and applications. A repository maintains all of the documentation too. Various people might use the CASE tools and associated repository during a database development project: systems analysts, database analysts, users, programmers, database and data administrators, and other technical specialists. As a significant new portion of a project is completed and entries are made in the repository, a review point occurs so that those working on the project and funding the human and capital resources of the project can assess progress and renew commitment based on incremental achievements.
Project Case

Case Questions

1. The tiered location plan will represent the location of databases and different machines throughout the organization. Since it would include information about the data to be hosted on each machine, a suitable approach will be to develop a business function-to-data entity matrix. It would identify which functions maintain instances of data, and which functions only use data about those entities. Then, a location-to-function matrix may be used in order to find out the locations at which those functions are performed. (A unit-to-function matrix may also be a good choice in this regard.) Accordingly, those business units which will maintain the data would need to be the data owner. At the same time, units which would only retrieve the data might have network access to it.

2. a. Quality of medical care:
   1. All staff members have to meet strict educational and certification requirements.
   2. Improve services and reduce the number of customers' complaints.
   3. Maintain library resources of up-to-date medical journals and other types of specialized literature.

   b. Control of capital costs:
      1. Acquire with caution new surgery equipment and electro-diagnosis technology.
      2. Wisely manage capital consuming, long-term projects.

   c. Recruitment and retention of skilled personnel:
      1. Maintain a predetermined number of nursing personnel and volunteers.
         Maintain an attractive employee benefits plan.
      2. Distribute subsidies among university and college-enrolled employees.
      3. Offer flexible schedules.

3. The three activities from the case study are: recruit volunteers, schedule volunteers, and evaluate volunteers. Some additional activities could be: train volunteers, schedule training sessions for volunteers, schedule orientation sessions for volunteers, etc.

4. Hold emergency simulation and evacuation training sessions, prepare a disaster recovery plan, perform system backups, etc.

5. One idea is to show the prepared function-to-entity matrix to employees involved with each function, who can identify any entities that may have been missed. This happens in enterprise modeling (part of the planning phase in the SDLC). To be more exact, enterprise modeling occurs during information systems planning and other activities. During SDLC, activities missing elements or errors may be found. This may occur while designing specific transactions, reports, displays, or inquiries in the analysis phase.
6. The assignment is for the development of an information systems plan for the hospital. The roles included on the business planning study team are: one information systems person, one assistant administrator, the head of surgery, and one consultant. This varied expertise should give the team a broad view over the different business functions. However, they will still need to communicate with the rest of the staff, e.g., human resources, the accounting department, nursing staff, general physicians, etc. Additional or different team members could also be included from any of these constituencies.

7.
   a. Each WARD has any number of PATIENTs assigned to it.
   b. A SUPPLY ITEM may be provided by any number of VENDORs.
   c. A MEDICAL/SURGICAL ITEM may be used by one or many PATIENTs.

8. Both those functions, electro-diagnosis and surgery, require heavy capital investments. In addition, those functions are highly dependent on technology. We could further expand this notation by adding the N status (business function is not of significant importance in achieving CSF). In that way, we could show, for example, that Patient Scheduling or Patient Accounting functions are not significant in achieving capital cost control.
Case Exercises

1.

<table>
<thead>
<tr>
<th>Business Function</th>
<th>Patient</th>
<th>Physician</th>
<th>Laboratory</th>
<th>Ward</th>
<th>Test</th>
<th>Facility</th>
<th>Staff</th>
<th>Medical/Surgical Item</th>
<th>Supply Item</th>
<th>Vendor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rehabilitation</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nursing</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Patient Monitoring</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dietary</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Surgery</td>
<td>R</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Patient Scheduling</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Blood Banking</td>
<td>R</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Patient Registration</td>
<td>M</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Multiphasic Screening</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Patient Accounting</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td></td>
<td></td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Physician Orders</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td></td>
<td></td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Laboratory Reporting</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td></td>
<td></td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Radiology</td>
<td>R</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Electrodiagnosis</td>
<td>R</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Psychiatric Testing</td>
<td>R</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Volunteering</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td></td>
<td></td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>General Accounting</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td></td>
<td></td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cost Accounting</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td></td>
<td></td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Payroll</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td></td>
<td></td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Personnel</td>
<td>M</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Inventory Control</td>
<td>M</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Housekeeping</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Risk Management</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td></td>
<td></td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Purchasing</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td></td>
<td></td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

M = data entity (column); is maintained by business function (row)
R = data entity (column); is used by business function (row)

Certain entities would participate in the majority of business functions, e.g., PATIENTS, PHYSICIANS, LABORATORY, while certain business functions may involve the majority of entities. This fact would be important in the development of this case since with this new rearrangement of the matrix, we could indicate what data are to be accessed most often. Also, which processes will need to access data about the majority of entities.
The business functions, Clinical Services and Patient Care Services (Figure 2-2), need to be expanded with some new subfunctions. The preliminary enterprise data model (Figure 2-3) does not need to be changed at this point. Some functions may be added to business function-to-data entity and business function-to-CSF matrices (Figure 2-4, 2-5), based on what is decided about the sub-functions noted in Figure 2-2.
3. **Note to instructor:** The preliminary enterprise data model in Figure 2-3 properly depicts the stated business rules in the Case Scenario. The diagram is reproduced here solely for reference purposes. Student answers could vary if additional assumptions are made about valid business rules for the Case Scenario (e.g., use of Volunteers, etc.).

![Preliminary Enterprise Data Model for Mountain View Community Hospital](image)

4. This can be achieved by selecting the functions with an E status from the business function-to-CSF matrix (Nursing, Patient Accounting, etc.). Then find out what entities from the business function-to-data entity matrix would need to be maintained or used by those functions. The problem is that almost every entity turns out to be of a great importance for this CSF, and this could happen with the rest of the CSF. Consequently, the quality of all data is essential.
5. We will have to first take a look at the Data Item/Name column of the metadata chart. It does list data items in the form of entity attributes' names. Since the preliminary enterprise data model will not show data attributes, but entities only, we could only verify if a particular entity from the metadata chart is represented on the enterprise data model; no detailed verification can be performed. Also, ITEM code and ITEM description that appear on the patient bill are not sufficient to determine whether a MEDICAL/SURGICAL ITEM or a SUPPLY ITEM is taken into consideration, e.g., some entity data may be similar in nature to other entity data. We need to go back and re-label the attributes in order to avoid misinterpretation.

6. Given the high priority assigned to this project, the obvious strategy is to first determine whether packaged information systems (including associated packaged data models) are available for risk management. Most hospitals are facing the same problems, and it is very likely that hospital-specific data models and systems are available. Of course, the hospital staff will have to determine whether such systems meet their needs and are compatible with existing development efforts. If instead, Mountain View Community Hospital decides to develop the systems internally, the solution could be a combination of the prototyping and life cycle approaches. Once the enterprise model has been agreed upon, it should be possible to identify the entities and relationships involved in the desired risk management system. From there, an interim prototype of the risk management system could be built if the manager's needs are deemed to be so critical that such a step is necessary. Thus, project initiation and planning (including placing this project within the company's information system architecture) can be addressed quickly. Then use an iterative cycle of analysis, design, and implementation to work closely with the manager of the risk management area to develop a working prototype of the system he or she needs. This system probably needs a database of limited scope, which may be used until it is integrated into the completed system.

7. Since we have considered prototyping the system as a preliminary piece of the entire system, but one that will be used quickly, this system may well be developed as a stand-alone database. Then, the unstructured and unpredictable use of data will not interfere with access to the operational databases needed to support efficient transaction processing systems. Also, the confidential information will not need to travel across the network, where possibilities of interception exist.

Project Assignments

P1. It sounds as if the team should be ready to consider the requirements of the proposed expansion of the system to possibly handle the two proposed new departments, Retirement Living and Geriatric Medicine. These requirements will need to be integrated into the existing enterprise model. Once the entire enterprise model is constructed, the proposed subsystems should be identified and prioritized so that plans can be laid to move on to begin the conceptual data model.

P2.

a. Note to instructor: Based on a reading of the Case Scenario, the clearly stated business rules tend to support the preliminary enterprise data model as shown in Figure 2-3 (and
Case Exercise 3 solution above); no modifications are necessary based on the stated business rules. However, a thorough reading of the Case Scenario will reveal the importance of Volunteers as well as a need to include Risk Management issues in this proposed new system and database. One of the events that a Hospital such as this may need to address is the use of an Evacuation Location. Thus, it can be implied that additions of Evacuation Location and Volunteer entities to the enterprise data model would be useful to development efforts of the project team (see revised diagram below). Adding these entities now will assist the team in remembering that the new system may require more data to support the requirements from these areas as the project progresses.
b. A volunteer volunteers at 1 or more facilities.  
A facility has 1 or more volunteers.  
A facility uses 1 evacuation location.  
An evacuation location is used by many facilities.

c. MVCH Context Diagram

![MVCH Context Diagram](chart.png)
Chapter 3  Modeling Data in the Organization

Chapter Overview

The purpose of this chapter is to present a detailed description of the entity-relationship model and the use of this tool within the context of conceptual data modeling. This chapter presents the basic entity-relationship (or E-R) model, while advanced features are presented in Chapter 4.

Chapter Objectives

Specific student learning objectives are included in the beginning of the chapter. From an instructor’s point of view, the objectives of this chapter are to:

1. Emphasize the importance of understanding organizational data, and convince your students that unless they can represent data unambiguously in logical terms, they cannot implement a database that will effectively serve the needs of management.
2. Present the E-R model as a logical data model that can be used to capture the structure and much, although not all, of the semantics (or meaning) of data.
3. Apply E-R modeling concepts to several practical examples including the Pine Valley Furniture Company case.

Key Terms

<table>
<thead>
<tr>
<th>Associative entity</th>
<th>Entity-relationship diagram (E-R diagram)</th>
<th>Multivalued attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attribute</td>
<td></td>
<td>Optional attribute</td>
</tr>
<tr>
<td>Binary relationship</td>
<td>Entity-relationship model (E-R model)</td>
<td>Relationship instance</td>
</tr>
<tr>
<td>Business rule</td>
<td></td>
<td>Relationship type</td>
</tr>
<tr>
<td>Cardinality constraint</td>
<td>Entity type</td>
<td>Required attribute</td>
</tr>
<tr>
<td>Composite attribute</td>
<td>Fact</td>
<td>Simple attribute</td>
</tr>
<tr>
<td>Composite identifier</td>
<td>Identifier</td>
<td>Strong entity type</td>
</tr>
<tr>
<td>Degree</td>
<td>Term</td>
<td>Ternary relationship</td>
</tr>
<tr>
<td>Derived attribute</td>
<td>Identifying owner</td>
<td>Time stamp</td>
</tr>
<tr>
<td>Entity</td>
<td>Identifying relationship</td>
<td>Unary relationship</td>
</tr>
<tr>
<td>Entity instance</td>
<td>Minimum cardinality</td>
<td>Weak entity type</td>
</tr>
<tr>
<td></td>
<td>Maximum cardinality</td>
<td></td>
</tr>
</tbody>
</table>

Classroom Ideas

1. Review the major steps in the database development process (Figure 2-5). Lead a discussion concerning who in the organization is typically most heavily involved in each of the steps and how end users may best participate in the process.
2. Introduce the concept of drawing models to represent information in a concise manner by having your students participate in a small active exercise in map-making. Divide the students into teams of 3-4 each so that you have an even number of teams in the class. Instruct each team to work together to investigate and develop a map to selected campus locations (you develop the list ahead of time; e.g., from this classroom to the library,
from this classroom to a colleague’s office, etc.). Ask each team to verify the map they
draw and then return to the classroom. Pair up each team with a unique location with
another team; ask the teams to exchange maps. Instruct each team to then verify the map
they received by following it and then returning to the classroom. Conduct a debriefing
discussion about how easy/hard it was to follow the maps, how useful were the symbols
used, how easily understood were the symbols, etc. Use this discussion to lead into the
use of E-R notation used to represent data models and why standardization is useful to
systems development activities.

3. Use the sample E-R diagram shown in Figure 3-1 to “jump-start” your students’
understanding. Ask your students to explain the business rules represented in this
diagram.

4. Use Figure 3-2 to summarize the basic E-R notation used in this chapter (and throughout
the remainder of the text).

5. Contrast the terms, entity type, and entity instance (see Figure 3-3). Discuss other
effects: STUDENT (with each student in the classroom as an instance), etc. Warn the
students that the term “entity” is often used either way; the meaning is intended to come
from the context in which it is used.

6. Give examples of common errors in E-R diagramming, including inappropriate entities
(see Figure 3-4). Ask your students for other examples.

7. Compare strong versus weak entities using Figure 3-5. Ask your students for other
examples.

8. Discuss the various types of attributes that are commonly encountered (Figures 3-7
through 3-9). Again, ask your students to think of other examples.

9. Make sure your students understand the difference between relationship types and
relationship instances (Figure 3-10).

10. Introduce the notion of an associative entity by using Figure 3-11. Discuss the four
reasons (presented in the text) for converting a relationship to an associative entity.

11. Discuss unary, binary, and ternary relationships (Figure 3-12). Have the students
brainstorm at least two additional examples for each of these relationship degrees.

12. Discuss the bill-of-materials unary relationship (Figure 3-13). Use a simple and familiar
product (such as a toy) to illustrate this structure.

13. Introduce the concept and notation of cardinality constraints in relationships (Figures 3-
16, 3-17, and 3-18). Emphasize that these constraints are important expressions of
business rules.

14. Introduce the problem of representing time dependent data. Use Figures 3-19 and 3-20 to
illustrate different means of coping with time dependencies.

15. Discuss examples of multiple relationships between entities (Figure 3-21). Ask your
students to suggest other examples.

16. Use the diagram for Pine Valley Furniture Company (Figure 3-22) to illustrate a more
comprehensive E-R diagram. Stress that in real-world situations, E-R diagrams are often
much more complex than this example.

17. As time permits, have your students work in small teams, 2 or 3 students each, to solve
some of the E-R diagramming exercises at the end of the chapter. We have included a
number of new examples for this purpose. Also, you may assign the project case as a
homework exercise.
Answers to Review Questions

1. Define each of the following terms:
   a. **Entity type**  A collection of entities that share common properties or characteristics.
   b. **Entity-relationship model**  A logical representation of the data for an organization or for a business area.
   c. **Entity instance**  A single occurrence of an entity type.
   d. **Attribute**  A property or characteristic of an entity type that is of interest to the organization.
   e. **Relationship type**  A meaningful association between (or among) entity types.
   f. **Identifier**  An attribute (or combination of attributes) that uniquely identifies individual instances of an entity type.
   g. **Multivalued attribute**  An attribute that may take on more than one value for a given entity instance.
   h. **Associative entity**  An entity type that associates the instances of one or more entity types and contains attributes that are peculiar to the relationship between those entity instances.
   i. **Cardinality constraint**  Specifies the number of instances of one entity that can (or must) be associated with each instance of another entity.
   j. **Weak entity**  An entity type whose existence depends on some other entity type.
   k. **Identifying relationship**  The relationship between a weak entity type and its owner.
   l. **Derived attribute**  An attribute whose values can be calculated from related attribute values.
   m. **Multivalued attribute**  See letter g.
   n. **Business rule**  A statement that defines or constrains some aspect of the business.

2. Match the following terms and definitions:
   i. composite attribute
   d. associative entity
   b. unary relationship
   j. weak entity
   h. attribute
   m. entity
   e. relationship type
   c. cardinality constraint
   g. degree
   a. identifier
   f. entity type
   k. ternary
   l. bill-of-materials
3. Contrast the following terms:
   a. **Stored attribute; derived attribute** A stored attribute is one whose values are stored in the database, while a derived attribute is one whose values can be calculated or derived from related stored attributes.
   b. **Simple attribute; composite attribute** A simple attribute is one that cannot be broken down into smaller components, while a composite attribute can be broken down into component parts.
   c. **Entity type; relationship type** An entity type is a collection of entities that share common properties or characteristics, while a relationship type is a meaningful association between (or among) entity types.
   d. **Strong entity type; weak entity type** A strong entity type is an entity that exists independently of other entity types, while a weak entity type depends on some other entity type.
   e. **Degree; cardinality** The degree (of a relationship) is the number of entity types that participate in that relationship, while cardinality is a constraint on the number of instances of one entity that can (or must) be associated with each instance of another entity.
   f. **Required attribute; optional attribute** A required attribute must have a value for each entity instance, whereas an optional attribute may not have a value for every entity instance.
   g. **Composite attribute; multivalued attribute** A composite attribute has component parts that give meaning, whereas a multivalued attribute may take on or more values for an entity instance.

4. Three reasons why data modeling is the most important part of the system development process:
   a. The characteristics of data captured during data modeling are crucial in the design of databases, programs, and other system components. Facts and rules that are captured during this process are essential in assuring data integrity in an information system.
   b. Data, rather than processes, are the most important aspects of many modern information systems and hence, require a central role in structuring system requirements.
   c. Data tend to be more stable than the business processes that use the data. Thus, an information system that is based on a data orientation should have a longer useful life than one based on a process orientation.

5. Four reasons why a business rules approach is advocated as a new paradigm for specifying information systems requirements:
   a. Business rules are a core concept in an enterprise since they are an expression of business policy, and they guide individual and aggregate behavior. Well-structured business rules can be stated in a natural language for end users and in a data model for system developers.
   b. Business rules can be expressed in terms that are familiar to end users. Thus, users can define and then maintain their own rules.
   c. Business rules are highly maintainable: they are stored in a central repository and
each rule is expressed only once, then shared throughout the organization.

d. Enforcement of business rules can be automated through the use of software that can interpret the rules and enforce them using the integrity mechanisms of the database management system.

6. Business rules appear in descriptions of business functions, events, policies, units, stakeholders, and other objects. These descriptions can be found in interview notes from individual and group information systems requirements collection sessions, organizational documents, and other sources. Rules are identified by asking questions about the who, what, when, where, why, and how of the organization.

7. Six general guidelines for naming data objects in a data model:
   a. Data names should relate to business, not technical characteristics.
   b. Data names should be meaningful, almost to the point of being self-documenting.
   c. Data names should be unique from the name used for every other distinct data object.
   d. Data names should be readable, so the name is structured as the concept would most naturally be said.
   e. Data names should be composed of words taken from an approved list.
   f. Data names should be repeatable, meaning that different people or the same person at different times should develop exactly or almost the same name.

8. Four criteria for selecting identifiers for entities:
   a. Choose an identifier that will not change its value over the life of each instance of the entity type.
   b. Choose an identifier such that for each instance of the entity the attribute is guaranteed to have valid values and not be null (or unknown).
   c. Avoid the use of so-called intelligent identifiers (or keys), whose structure indicates classifications, locations, and so on.
   d. Consider substituting single-attribute surrogate identifiers for large composite identifiers.

9. Three conditions that suggest the designer should model a relationship as an associative entity type are:
   a. All of the relationships for the participating entity types are “many” relationships.
   b. The resulting associative entity type has independent meaning to end users, and it preferably can be identified with a single-attribute identifier.
   c. The associative entity has one or more attributes in addition to the identifier.
10. Four types of cardinality constraints are:
   a. Optional one:
      ![Diagram of PERSON to BICYCLE relationship with "Owns"
   b. Mandatory one:
      ![Diagram of TEAM to LEADER relationship with "Lead_By"
   c. Optional many:
      ![Diagram of STUDENT to COURSE relationship with "Registers_For"
   d. Mandatory many:
      ![Diagram of COURSE to TEXTBOOK relationship with "Uses"

11. PHONE CALL (see below) is an example of a weak entity because a phone call must be placed by a PERSON. Because in this simple example, PHONE CALL is related to only one other entity type, it is not necessary to show the identifying relationship; however, if this data model were ever expanded so that PHONE CALL related to other entity types, it is good practice to always indicate the identifying relationship.

   ![Diagram of PERSON to PhoneCall relationship with "Places"
12. The degree of a relationship is the number of entity types that participate in the relationship.

1) Unary (one entity type):

```
    Related_to
    /
PERSON
```

2) Binary (two entity types):

```
PERSON  Attends  EVENT
```

3) Ternary (three entity types):

```
CONSULTANT  Signs  CLIENT
             /
CONTRACT
```
13. Attribute examples:
   a. Derived – distance (rate x time)
   b. Multivalued – spoken language
   c. Composite – flight ID (flight number + date)

14. Examples of relationships:
   a. Ternary

   ![Ternary Relationship Diagram]

   b. Unary

   ![Unary Relationship Diagram]
15. 

```
STUDENT
Student_ID
{Advisor
  (Advisor_Name
  Advise_Start_Date
  Advise_End_Date))
```

16. When the attribute is the identifier or some other characteristic of an entity type in the data model, and multiple entity instances need to share these same attributes.

17. • A relationship name should always be a verb phrase and should state the action taken, as opposed to the result of the action taken.
    • Use descriptive, powerful verb phrases as opposed to vague names.

18. The relationship definition should also explain the following:
    • any optional participation
    • the reason for any explicit maximum cardinality
    • any mutually exclusive relationships
    • any restrictions on participation in the relationship
    • the extent of history that is kept in the relationship
    • whether an entity instance involved in a relationship instance can transfer participation to another relationship instance

19. Presently, the cardinality is one-to-many. One possible scenario is an employee who is supervised by more than one manager. This would make the cardinality many-to-many. Another possibility is that the employee is supervised by one manager, and the manager only supervises one employee. This would result in a one-to-one cardinality. If we take time/history into consideration, the idea of someone being managed currently versus never being managed could affect the cardinality. As we can see here, you cannot always tell what the business rule is by looking at the ERD. These possible scenarios will need to be discussed with the end user to determine the “correct” modeling representation for the business rules at this organization.

20. An entity type can be thought of as a template, defining all of the characteristics of an entity instance. For example, “student” would be an entity type, whereas you are an instance of “student.”
Answers to Problems and Exercises

1. Each answer refers to Figure 3-22 found in the chapter text.

   a) Where is a unary relationship, what does it mean, and for what reasons might the cardinalities on it be different in other organizations?

      A unary relationship is shown with the EMPLOYEE entity; An EMPLOYEE Supervises 0:M EMPLOYEEs, An EMPLOYEE Is_Supervised_By 0:1 EMPLOYEE. This relationship tells us that we can determine what employees are supervised by another employee, as well as determine which employees are supervisors in this company.

      In other organizations, there may be different policies regarding employee supervision that could cause the data relationships among EMPLOYEE instances to be different. For instance, another company might allow an employee to have multiple supervisors (e.g., in an organization with a matrix structure).

   b) Why is Includes a one-to-many relationship and why might this ever be different in some other organization?

      Includes is a one-to-many (1:M) relationship because of the business rules that PVFC has in place: “a product line may group any number of products but must group at least one product; and each product must belong to exactly one product line.” Another organization may have other business rules that could permit a product being assigned to more than one product line (changing Includes to a M:M relationship). Alternatively, another organization might also show Includes as a (1:M) overall relationship but might permit the establishment of a PRODUCT_LINE without identifying PRODUCTS that belong to this group (e.g., thus permitting an optional minimum cardinality on the PRODUCT side of the Includes relationship).

   c) Does Includes allow for a product to be represented in the database before it is assigned to a product line (e.g., while the product is in research and development)?

      No, Figure 3-22 shows that the PRODUCT must be Included in at least 1 PRODUCT_LINE by the mandatory 1 and only 1 cardinality notation near the PRODUCT_LINE portion of the Includes relationship line. The cardinality notation would have to be changed to show optional 1 cardinality in order to represent the research and development situation.

   d) Suppose there were a different customer contact person for each sales territory in which a customer did business, where in the data model would we place this person’s name?

      The Does_Business_In associative entity, that associates a single instance of a SALES_TERRITORY with a single instance of a CUSTOMER, would permit the tracking of a customer contact person name for each sales territory in which a customer did business.

   e) What is the meaning of the Does_Business_In associative entity and why does each Does_Business_In instance have to be associated with exactly one SALES_TERRITORY and CUSTOMER?
The Does_Business_In associative entity associates a single instance of a SALES_TERRITORY with a single instance of a CUSTOMER for the overriding M:M Does_Business_In relationship between SALES_TERRITORY and CUSTOMER. Each Does_Business_In instance must be related to exactly one SALES_TERRITORY and one CUSTOMER because the business rules of PVFC indicate that sales territories have been established for its customers. In particular, the rules are: a SALES_TERRITORY has one-to-many CUSTOMERs; and a CUSTOMER may do business in 0:M SALES_TERRITORIES. When converting this M:M relationship on the ERD, the cardinalities near the originating entities will always be mandatory 1, indicating the exactly one relationship with each entity’s instances and the associative entity’s instance.

f) In what way might Pine Valley change the way it does business that would cause the Supplies associative entity to be eliminated and the relationships around it change?

According to current business practice at PVFC, each RAW_MATERIAL is provided by 1 or more VENDORs and a VENDOR supplies 0, 1, or many RAW_MATERIALs and this is represented by the Supplies associative entity. The PVFC could consider entering into exclusive supplier arrangements with particular vendors such that an instance of RAW_MATERIAL is supplied by only 1 VENDOR. If that situation should occur, then the overall relationship between RAW_MATERIAL and VENDOR would change to 1:M (instead of M:M) and the Supply_Unit_Price attribute could become part of the RAW_MATERIAL entity instance; the Supplies associative entity would no longer need to be on the ERD.

2. Analysis of Figure 3-22:
   2.1. Entities PRODUCT, PRODUCT_LINE; relationship Includes
   2.2. Entities CUSTOMER, ORDER; relationship Submits
   2.3. Entities ORDER, PRODUCT; associative entity Order_Line
   2.4. Entities CUSTOMER, SALES_TERRITORY; associative entity Does_business_in
   2.5. Entities SALESPERSON, SALES_TERRITORY; relationship Serves
   2.6. Entities PRODUCT, RAW_MATERIAL; relationship Uses
   2.7. Entities RAW_MATERIAL, VENDOR; relationship Supplies
   2.8. Entities WORK_CENTER, PRODUCT; associative entity Produces_In
   2.9. Entities EMPLOYEE, WORK_CENTER; associative entity Works_In
   2.10. Entity EMPLOYEE; relationship Supervises, Is_supervised_by

3. Student answers will vary based on the CASE or drawing tool that is used and their personal experiences using the tool. The answers should describe their experiences with the CASE or drawing tool in terms of the requirements of the E-R notation used in the chapter. Expect to see students make reference to noting identifiers, using associative entities, using cardinality constraints properly, indicating required vs. optional attributes, and noting derived/composite/multivalued attributes.
4a) The ERD for City B does not (nor does any ERD) tell us why the cardinality is 1:M. The more restrictive cardinality for City B could be due to a business rule that they want to maintain only current volunteers but it could also be due to only tracking the agency for which the volunteer works the most hours of assistance. More detailed discussions would need to be held with the end users to properly document this business rule; notes should be added to the diagram to depict the appropriate business rule.

4b) The ERD for City A shows that a volunteer may assist one, none, or several agencies.

4c) The native notation used in ERDs does not show whether membership in a relationship can change (i.e., whether a volunteer can change agencies or whether an agency can change its volunteers). Some DBMSs can be told whether membership can change or not, and special notation or textual notes can be added to an ERD to state such business rules. The minimum cardinality next to Agency does address whether a Volunteer must always be associated with an Agency to exist in the database, but none of the cardinalities control whether linkages between specific agencies and volunteers can change. More detailed discussions would need to be held with the end users to properly document this business rule; notes should be added to the diagram to depict the appropriate business rule.

<table>
<thead>
<tr>
<th>Question</th>
<th>City A</th>
<th>City B</th>
<th>Can't Tell</th>
</tr>
</thead>
<tbody>
<tr>
<td>a. Which city maintains data about only those volunteers who currently assist agencies?</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>b. In which city would it be possible for a volunteer to assist more than one agency?</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>c. In which city would it be possible for a volunteer to change which agency or agencies she assists.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>
5.
5a.

Yes, the attribute names do generally follow the guidelines for naming attributes.

5b.

Assignment: All three entities participate in the Assigned relationship that is modeled as an associative entity Assignment, since the Assign_Date for each Chemist’s assignment to a particular project and equipment item must be tracked. However, EQUIPMENT and PROJECT do not need to participate in any assignments. All entities can have multiple assignments.
5c. 

![Database Diagram](image)

**Note:** SECTION is modeled as a weak entity. It could have been modeled as a multivalued attribute; however, using a weak entity is better, since SECTION may have a relationship with another entity. A multivalued attribute could not be used to show this relationship.

5d. 

![Database Diagram](image)

Both Admits and Treats relationships were created since the patient could be treated by different PHYSICIANS than the admitting PHYSICIAN.
5e. **First situation**: credit check can be used by more than 1 request.

Second Situation: CREDIT_CHECK can only be used by 1 CREDIT_REQUEST (2 entities)

Second Situation: CREDIT_CHECK can only be used by 1 request (1 entity)

Using 1 entity type seems much simpler since the credit check and rating only apply to this credit request. However, Credit_Check_Date and Credit_Rating will be blank (null) until the credit check is received.

5f. Starting point diagram:
(5f continued)  Situation 1 - Adding Hourly_Rate attribute: this could be added to the CONSULTANT entity as the business rule is that a CONSULTANT Works for only 1 COMPANY at a time.

Situation 2 – Tracking a consultant’s contract. Note that CONTRACT is added as another entity that participates in a binary relationship with COMPANY and a binary relationship with CONSULTANT. We have moved Hourly_Rate to the CONTRACT entity, which permits a CONSULTANT to vary his/her Hourly_Rate as a function of the particular CONTRACT for a COMPANY. As only current CONTRACTs are tracked, an alternative solution would be to move the CONSULTANT attributes into the CONTRACT entity and eliminate the CONSULTANT entity from the model. The downside to this alternative solution is that Consultant_Name and Consultant_Specialty would occur redundantly in the CONTRACT entity instances.

Situation 3: we want to track historical CONTRACT information. We can create an associative entity for CONTRACT. I’ve also added Contract_ID as a surrogate identifier that is a unique serial number (not a composite identifier, as shown in Situation 2 above).
5g.

**Notes:**

1) ARTWORK is created by 0:1 ARTIST (0 for Unknown ARTIST); alternative design would be to have a valid ARTIST instance with a Name of "Unknown", this would enable you to enforce a business rule that each piece of ARTWORK must have an ARTIST stored in the database and the cardinality would change to mandatory 1 near the ARTIST entity in the diagram.

2) Item_Status attribute of ARTWORK permits designation of ARTWORK as Display (and then a valid value for Museum_Location attribute), Storage, Loan, or Show.

3) An ARTWORK item may participate in a SHOW; however, there is not a way to note on the ERD that an ARTWORK item cannot be a part of two shows with overlapping dates. This constraint will need to be noted as part of the system design documentation.
5h. **NOTES TO INSTRUCTOR for P&E 5h:** This problem and exercise is a good lead-in for Chapter 4 modeling notation for the Extended Entity Relationship Diagram (EERD). The P&E offers several chances to provide better representation in the EERD (with subtyping) than the ERD notation that is provided in Chapter 3. Using EERD notation, a single LEGAL_ENTITY can be shown as a supertype, with subtypes of DEFENDANT and PLAINTIFF. The 'type' (person or Organization) characteristic of both DEFENDANT and PLAINTIFF may also be considered for further subtyping. The solution presented here is a valid answer to the P&E, given the limitations of basic ERD notation and what is currently known about the situation.

This P&E also provides the instructor with an opportunity to discuss how history might be modeled if the business assumption regarding the tracking of Net_Worth for both Plaintiff and Defendant was changed from only being concerned with Net_Worth at the time of the CASE, to wanting to track the Net_Worth over time of each party to the CASE. Refer to the chapter section on “Modeling Time-Dependent Data” and Figure 3-19 for more information on how this ERD could be revised.

**Note:**
1) Def_Type and Plaintiff_Type are used to denote Person or Organization type of legal entity.
2) Net_Worth of both Plaintiff and Defendant is relevant only at the time of the CASE, thus is modeled as attributes of the M:N relationships between CASE and PLAINTIFF, DEFENDANT.
5i.

Note:
No checks are written before the first royalty is paid, thus the minimum cardinality is 0 for the Royalty_Check associative entity.
6. NOTE: The addition of Semester and Year attributes on the Registers_for relationship allows this diagram (and resulting database) to reflect multiple semesters of data.

7. Note: Assume Student_Name is unique and available to be used as the identifier.

\textbf{Note:}
Assume Instructor_Name is unique and thus available to be used as an identifier.
8.

Note: An additional business rule for this scenario is that an EMPLOYEE may Manage only the SALE OFFICE to which s/he Is Assigned.
8. continued

Entities:

Employee: An employee of the firm. An employee works for one sales office and may
manage one sales office. It is not explicitly indicated that the employee can only
manage the office that he/she works for. This would require a business rule.

Sales_Office: The office where real estate is sold.

Property: Buildings for sale, such as houses, condos and apartment buildings.

Owner: The individual who owns one or more properties.

Attributes on Employee:

Employee_ID: A unique identifier for an employee. This attribute must be unique.
Employee_Name: The name of the employee.

Attributes on Sales_Office:

Office_Number: A unique identifier for the office.
Location: The physical location of the sales office. This data may be made up of the city
and state.

Attributes on Property:

Property_ID: The unique identifier for the property.
Location: A composite attribute that consists of the street address, city, state, and Zip
Code.

Attributes on Owner:

Owner_ID: The unique identifier for the owner.
Owner_Name: The name of the owner.

Relationship:

Is_assigned: An employee is assigned to one sales office. A sales office may have many
employees assigned but must have at least one employee.

Manages: An employee may manage one sales office or no sales office. Each sales office
is managed by one employee. A business rule is needed here in order to indicate
that an employee can only manage the sales office in which he or she works.

Lists: Each property is listed by only one sales office. Each sales office can list one,
one, or many properties.

Owns: Each property has one or more owners. Each owner can own one or more
properties. Percent_owned is an attribute on Owns; it tracks the percent of
property that the owner owns.
9. Note: attributes are omitted to save space in the Instructor’s Manual.

a. 

b. 

c.
e. [Diagram showing relations between ITEM, BOM_STRUCTURE, and Has_Components, Used_in_assemblies]

f. [Diagram showing relations between PART, VENDOR, SUPPLY_SCHEDULE, and WAREHOUSE]
10.  
Problem & Exercise 10a

```
PERSON
Person_ID
Person_Name
```

Is_Married_To

Problem & Exercise 10b

```
PERSON
Person_ID
Person_Name
```

Is_Married_To

Problem & Exercise 10c

```
PERSON
Person_ID
Person_Name
```

Is_Married_To

```
Marriage_Date
Dissolution_Date
```
Problem & Exercise 10d

Notes:
1) This solution presumes that Marriage_Date is a partial identifier of the MARRIAGE entity; a full composite identifier will include Marriage_Date and the two Person_IDs involved in the marriage. The solution also assumes that the same two people do not get married, dissolved, and re-married on the same date. Adding a Marriage_Time attribute (also a part of the identifier) would permit this situation to be covered by this model.
2) An alternate solution would be to use a surrogate identifier of License_No instead of the suggested composite identifier of Marriage_Date and the two Person_IDs for the MARRIAGE entity.

Problem & Exercise 10e:

The solution in 10d does not place any restrictions on the number of persons to whom any one person is simultaneously married, thus the 10d solution is sufficient in representing the lack of legal restrictions regarding the number of marriage partners.
11a) A STUDENT Works_For 0:1 SCHOOL; A SCHOOL Employs 0:M STUDENTs

11b) A STUDENT may belong to a CLUB only when located in the SCHOOL s/he Attends

11c) Student answers may vary. Alternative solutions are as follows:

- Since the STUDENT may not Work_For a SCHOOL (the employment is optional), the Works_For relationship is needed in the diagram in order to properly represent this business rule. This solution makes it harder for the database to enforce the business rule that a STUDENT works for the SCHOOL that s/he attends, but opens up the possibility that a STUDENT could work for a SCHOOL that s/he is not currently attending.

- An alternative design would be to remove the Works_For relationship, and add an attribute to STUDENT named Works that would have a binary (Y/N) value to represent whether or not the STUDENT instance is working for the SCHOOL s/he Attends. The advantage of this design is that it would enforce the business rule that a STUDENT can only work for a SCHOOL that s/he is currently attending.
12. Are associative entities also weak entities? Why or why not? If yes, is there anything special about their “weakness”?

A weak entity requires the presence of another entity type; the weak entity does not exist independently from the other entity type and has no business meaning in the ERD without the other entity type. A weak entity will not have its own identifier, but will have a partial identifier attribute that will later be combined with the identifier of its strong entity owner to create a full identifier.

An associative entity is an entity type that associates the instances of one or more entity types and contains attributes specific to the relationship between those entity instances. An associative entity generally has independent business meaning to end users and can be identified with a single-attribute identifier. If an associative entity meets these conditions, then it would not be considered a weak entity.

13. Figure 3-27 shows two diagrams (A and B), both of which are legitimate ways to represent that a stock has a history of many prices. Which of the two diagrams do you consider a better way to model this situation and why?

**NOTE TO INSTRUCTOR:** Student answers may vary. The crux of the answer relies upon what is the purpose of the ER diagram for the modeling situation and how end users in the organization “see” the situation. In particular, do people in the organization have a term for stock_price and refer to it as its own concept? If so, solution B may be the “better” way to model this situation. Instructors may also use solution B to demonstrate an issue related to view integration (topic in chapter 5) where transitive dependencies emerge; solution B makes the model easy to expand so that stock prices may have relationships that do not directly involve the STOCK entity.

Solution A indicates that each STOCK has multiple prices and is well-suited to early discussions with end users about the data needs of a system. Solution B adds the precision of multiple STOCK_PRICE entity instances occurring for each STOCK entity instance. Solution B indicates that STOCK_PRICE is a weak entity whose instances do not exist independently in the database without a corresponding STOCK entity instance. Solution B presents more precise detail of the data relationships that will likely be developed in the logical design of the database; this model may more closely resemble the relational model implementation of this design. Solution B also makes it easy to expand the model so that stock prices may have relationships with other entities that do not directly involve the STOCK entity.

The crux of the answer relies upon what is the purpose of the ER diagram for the modeling situation and how end users in the organization “see” the situation.
A SOLOIST performs one or more COMPOSITIONs at one or more CONCERTs. This is modeled using a ternary relationship, Performs, which is shown as an associative entity PERFORMANCE.
15. **Note to instructor:** Student answers to this problem and exercise will vary based on their life experiences (e.g., do the students actually receive and review monthly/annual credit card statements), the drawing tool used, and the documents chosen. Three alternative solutions are presented and are ordered from the least complex to the most complex scenario. The purpose of this problem and exercise is to begin sensitizing students to the occurrence of synonyms and homonyms when ERDs are created. The actual topic does not show up until Chapter 4, but this problem and exercise can be a good lead-in for this discussion.

**Alternative One: 15a**

Problem & Exercise 15a
American Express
Monthly Statement of Account ERD - initial draft

Notes:
1) Card_Type refers to Standard, Gold, Platinum, Corporate.
2) Activity_Type refers to Purchase or Payment.
3) Activity_Desc is modeled as a composite attribute so that we don't forget to show the details of the Merchant contact information in an Activity instance in the database.
Alternative One: 15b

Problem & Exercise 15b
American Express
Year-End Summary ERD - initial draft

CUSTOMER
Customer_ID
Customer_Name
( Last, Mi, First)
Customer_Address
(Street, City, State, Zip)
Customer_Phone
Customer_Email

ACCOUNT
Account_No
Exp_Date
Card_Type

TRANSACTION
Txn_ID
Txn_Date
Txn_Post_Date
Txn_Desc
Charge_Amount

Notes:
1) Card_Type refers to Standard, Gold, Platinum, Corporate.
2) SPENDING_CATEGORY provides major grouping of CARD_ACCOUNT transactions so
the CUSTOMER can know spending patterns by groups such as TRAVEL, AUTO, etc. Each
SPENDING_CATEGORY includes at least one, and usually more than one, SUBCATEGORY.
SUB_CATEGORIES are unique to each SPENDING_CATEGORY.
3) TxnDesc includes the 40 character text description of each Merchant's charge to the CARD_ACCOUNT.

Alternative One: 15c

Do you find the same entities, attributes, and relationships in the two ERDs you developed for
parts a and b? What differences do you find in modeling the same data entities, attributes, and
relationships between the two ERDs? Can you combine the two ERDs into one ERD for which
the original two are subsets? Do you encounter any issues in trying to combine the ERDs?
Suggest some issues that might arise if two different data modelers had independently developed
the two data models.

Yes, the same entities of CUSTOMER and ACCOUNT are in both sets of ERDs; these
entities also appear to share the same attributes in each ERD. The relationship between
CUSTOMER and ACCOUNT in part a ERD is Owns, while in part b ERD it is Holds.
This would appear to be the same kind of relationship between entity instances in both
ERDs. Also, the TRANSACTION entity in part b appears to be the same as ACTIVITY
in part a.

There appear to be differences in the level of detail that is modeled in the ACTIVITY entity with respect to the description of the activity charge when it is compared to the TRANSACTION entity’s TxnDesc attribute. Additionally, the part b ERD shows additional entities of SPENDING_SUB_CATEGORY and SPENDING_CATEGORY that are related to TRANSACTION; these additional entities are not in evidence in the part a ERD.

It would appear that these two ERDs can be combined into one ERD with minimal confusion. However, further clarification from the end user is necessary to determine the meaning (semantics) of the Activity_Type attribute in part a ERD and the Txn_Desc attribute in part b ERD. Further, some discussion is necessary to determine whether the use of “Activity” or “Transaction” terminology is preferred with the end users so proper decisions can be made about attribute naming conventions.

If two data modelers had independently modeled these user views, it is possible that even greater variance might be evidenced between the entity, attribute, and relationship names. It is also possible that the data modeler working on the Monthly Statement user view might not have been as specific in noting the composition of the Activity_Desc attribute; thus, it would not be apparent that contact information related to the Merchant is part of this data model.

**Alternative One: 15d**
How might you use data naming and definition standards to overcome the issues you identified in part c?

Naming and definition standards could be used to develop common Classes [e.g., Identifier (ID), Number (No), Date (Date), Address (Addr), Transaction (Txn), Description (Desc)] and Qualifiers [Post, Transaction, Activity], as well as how attribute names will be noted (i.e., Account_No vs. AccountNo).
Alternative Two: 15a

Problem & Exercise 15a
Credit Card Receipt ERD - first draft

Notes:
1) Transaction_Type refers to Sale or Refund; a RECEIPT has only 1 Transaction_Type at a time.
2) This ERD refers to a Credit Card Receipt; revisions would be necessary to depict a cash transaction.
3) Card_Type refers to Visa, MasterCard, American Express, Discover, etc.
**Alternative Two: 15b**

**Problem & Exercise 15b**
Monthly Statement of Visa Credit Card Account ERD - first draft

**ACCOUNT**
- AccountNo
- CreditLine
- CashAdvLimit
- AcctStatus
- BillingCycleDays
- BillingCycleDate
- ExpDate
- CurrentBalance

**CUSTOMER**
- CustNo
- Name (Last, MI, First)
- Address (Street, City, State, Zip)
- Phone
- ApprovalDate

**MERCHANT**
- MerchantNo
- MerchantName
- MerchantAddress (Street, City, State, Zip)
- MerchantPhone
- MerchantTxnText
- MerchantApprovalDate

**TRANSACTION**
- TxnID
- TxnDate
- PostDate
- TxnType
- TxnAmount

**TXN_CATEGORY**
- TxnCatCode
- TxnCatDescription

**CHG_TO_CATEGORY**
- ChgToCatCode
- ChgToCatDescription

**Notes:**
1) TxnType refers to Purchase, Cash Advance, Payment, or Adjustment.
2) CHG_TOCATEGORY refers to Finance Charge Categories (e.g., StandardPurchase or StandardCashAdv).
3) TXN_CATEGORY refers to Spending Categories (e.g., Merchandise, Services, Auto Rental, etc.).
4) AcctStatus refers to Active, Inactive, Closed, Overdue.
5) MerchantTxnText refers to the text shown as part of the TxnDescription; if this value is NULL, then the business rule is to show MerchantName, MerchantCity, MerchantState as part of the Transaction Description information on the Monthly Summary.
Alternative Two: 15c

Do you find the same entities, attributes, and relationships in the two ERDs you developed for parts a and b? What differences do you find in modeling the same data entities, attributes, and relationships between the two ERDs? Can you combine the two ERDs into one ERD for which the original two are subsets? Do you encounter any issues in trying to combine the ERDs? Suggest some issues that might arise if two different data modelers had independently developed the two data models.

Yes, when comparing the ERDs in part a and part b, MERCHANT appears to be the same entity in both data models. Additionally, since it is known that the physical Receipt document that was used to generate the part a ERD is actually one of the transactions that is shown on the Visa Monthly Statement, there are common attributes between RECEIPT (part a) and TRANSACTION (part b), although different names have been used in the data models. Additionally, the CC_Account_No from RECEIPT (in part a) is equivalent to the AccountNo from ACCOUNT (in part b).

The two ERDs could be combined into one ERD, however, there would need to be decisions made about how the data that crosses organizational boundaries are maintained in different organization’s databases. For instance, the Receipt_No on the Merchant’s receipts for purchases at the Merchant are relevant to the Merchant’s internal accounting records and may not be of use to the Credit Card Company’s reporting to its account cardholders. Likewise, the Credit Card Company needs to track the date that a particular account transaction is posted to the account, and this level of data is most likely not of interest to the Merchant.

Aside from this larger issue, there are some minor naming issues that will need to be overcome if the data models are combined. Even though the MERCHANT entities are the same, standardization on names for the attributes needs to be resolved (e.g., Merchant_ID vs. MerchantNo; Merchant_Name vs. MerchantName, etc.). Additionally, the business usage of Transactions versus Receipt language needs to be sorted out.

If two different data modelers had developed these ERDs, there would likely be even more variance in how the names of Entities, Attributes, and Relationships would have been established. It’s also possible that the different data modelers would not recognize that the RECEIPT and TRANSACTION entities are similar, if they did not share the sample data from each separate user view with each other.
**Alternative Two: 15d**

How might you use data naming and definition standards to overcome the issues you identified in part c?

Naming and definition standards could be used to develop common Classes [e.g., Number (No), Credit Card (CC), Date (Date), Address (Addr), Transaction (Txn), Description (Desc)] and Qualifiers [Post, Transaction, Activity, BillingCycle], as well as how attribute names will be noted (i.e., Account_No vs. AccountNo).

**Alternative Three: 15a**

**Problem & Exercise 15a**

Cash Register Credit Card Receipt ERD - first draft

<table>
<thead>
<tr>
<th>STORE</th>
<th>RECEIPT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Store_No</td>
<td>Receipt_No</td>
</tr>
<tr>
<td>Store_Name</td>
<td>Date</td>
</tr>
<tr>
<td>Store.Addr</td>
<td>Time</td>
</tr>
<tr>
<td>(Street, City, State, Zip)</td>
<td>Register_No</td>
</tr>
<tr>
<td>Store.Phone</td>
<td>Transaction_Type</td>
</tr>
<tr>
<td></td>
<td>Cashier</td>
</tr>
<tr>
<td></td>
<td>Line_Items_Subtotal</td>
</tr>
<tr>
<td></td>
<td>Total_Tax</td>
</tr>
<tr>
<td></td>
<td>Purchase_Total</td>
</tr>
<tr>
<td></td>
<td>CC_Charge_Amount</td>
</tr>
<tr>
<td></td>
<td>CC_Account_No</td>
</tr>
<tr>
<td></td>
<td>Card_Type</td>
</tr>
<tr>
<td></td>
<td>Auth_Code</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ITEM</th>
<th>Line_Item</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item_No</td>
<td>Line_Item_SeqNo</td>
</tr>
<tr>
<td>Item_Desc</td>
<td>Quantity</td>
</tr>
<tr>
<td>Item_Size</td>
<td>Subtotal</td>
</tr>
<tr>
<td>Item_SubSize</td>
<td></td>
</tr>
<tr>
<td>Item_Color</td>
<td></td>
</tr>
<tr>
<td>Item_Price</td>
<td></td>
</tr>
</tbody>
</table>

**Notes:**

1) Transaction_Type refers to Sale or Refund; a RECEIPT has only 1 Transaction_Type at a time.
2) This ERD refers to a Credit Card Receipt; revisions would be necessary to depict a cash transaction.
3) Cashier refers to the first name of the Cashier and is assumed to be unique. An alternative design would be to use a Cashier_Number and provide a relationship to a CASHIER entity.
4) Card_Type refers to Visa or MasterCard.
5) Line_Item_SeqNo is a partial identifier of the Line_Item associative entity; Receipt_No and Item_No will be needed to provide a full identifier for each Line_Item instance.
Alternative Three: 15b

Problem & Exercise 15b
Monthly Statement of Visa Credit Card Account ERD - first draft

Notes:
1) TxnType refers to Purchase, Cash Advance, Payment, or Adjustment.
2) CHG_TO_CATEGORY refers to Finance Charge Categories (e.g., StandardPurchase or StandardCashAdv).
3) TXN_CATEGORY refers to Spending Categories (e.g., Merchandise, Services, Auto Rental, etc.).
4) AcctStatus refers to Active, Inactive, Closed, Overdue.
5) MerchantTxnText refers to the text shown as part of the TxnDescription; if this value is NULL, then the business rule is to show MerchantName, MerchantCity, MerchantState as part of the Transaction Description information on the Monthly Summary.

Alternative Three: 15c

Do you find the same entities, attributes, and relationships in the two ERDs you developed for parts a and b? What differences do you find in modeling the same data entities, attributes, and relationships between the two ERDs? Can you combine the two ERDs into one ERD for which the original two are subsets? Do you encounter any issues in trying to combine the ERDs? Suggest some issues that might arise if two different data modelers had independently developed the two data models.

The Cash Register Credit Card Receipt ERD was developed from a user view of the Customer purchasing items from a Store, and reflects the entities and attributes present on that user view and sample data available in the actual user document. This data model will provide the Customer with a receipt including details of what was purchased, the
quantity of the item purchased, the price for each item purchased, as well as tax and the total charge to the credit card account. From the Store’s perspective, this data model provides tracking of the Cashier and Register related to the overall sales transaction, as well as credit card processing information (e.g., type of card, charge amount, card account number, and authorization code), and information related to management of the Store’s inventory (e.g., item information and quantities).

The Monthly Statement of a Visa Credit Card Account ERD was developed from a user view sent to the Account Owner of the Visa Credit Card and reflects the entities and attributes present in the data on the sample document. This data model serves both the Account Owner by providing details of all transactions posted against the Credit Card Account, and also the Visa Credit Card Company by providing transaction charges for both customers and merchants served.

When these two ERDs are reviewed, it does not appear that any entities, attributes, or relationships are named the same which seems to indicate that none of these are the same between the two ERDs. However, since both the receipt and the monthly statement are for my own purchases with a credit card, it is known that some of the data underlying both of these data models is the same, although different names have been used. For instance, the monthly statement shows a listing of individual credit card receipts. Although in this case, the individual receipt shows more detail that is shown on the monthly statement, it can be seen that the underlying data is the same. The STORE entity in part a is actually equivalent to the MERCHANT entity in part b. The CC_Charge_Amount, Date (from RECEIPT) in part a is the same as the TxnAmount, TxnDate (from TRANSACTION) in part b. Finally, the CC_Account_No (from RECEIPT) in part a is equivalent to the AccountNo (from ACCOUNT) in part b.

Although it is technically feasible to combine these two ERDs into one ERD, it would not be advisable due to the difference in the level of detail captured (e.g., Store Inventory Management data in part a) in the two models and due to the different purposes (and ultimate end users) of the data. Naming standards would also have to be developed to accomplish the merging of the data models. If two data modelers had developed these ERDs, it is unlikely that the common underlying data would have been identified.

Alternative Three: 15d
How might you use data naming and definition standards to overcome the issues you identified in part c?

Naming and definition standards could be used to develop common Classes [e.g., Number (No), Credit Card (CC), Date (Date), Address (Addr), Transaction (Txn), Description (Desc)] and Qualifiers [Post, Transaction, Activity, BillingCycle], as well as how attribute names will be noted (i.e., Account_No vs. AccountNo). However, these standards would not address the level of detail and purpose issues identified earlier as issues in merging the ERDs.
16. Projects, Inc. ERD

Notes:
- We assume that a Vendor will be tracked in our database even if they have not participated in a Buys_From relationship with a department, hence, the 0:M cardinality next to Department in the diagram. This permits the tracking of a Vendor in our database prior to the first transaction with us.
- We assume that we may set up a department in our company that may not yet have employees assigned to it; thus, the 0:M cardinality next to Employee on the Belongs_To relationship between Employee and Department.

Classes: Number (No), Identifier (ID), Date

Qualifiers: Married, Of_Birth, Last meeting
17. Stillwater Antiques ERD
18. H.I. Topi School of Business ERD

Note: Contact_type refers to mail, email, telephone, fax, or personal discussion.
19. PVFC ERD

Note: A COMPONENT may be used to make 0:M PRODUCTS, as a COMPONENT may be a raw material that is not used immediately in making a PRODUCT.
20. Emerging Electric ERD

```
CUSTOMER
Customer_ID
Customer_Name
Customer.Addr
(Street, City, State, ZipCode)
Telephone
```

```
LOCATION
Location_ID
Location_Type
Location.Addr
(Street, City, State, ZipCode)
```

```
RANGE
Rate_Class
Rate_Per_KWH
```

**Notes:**
1) A RATE may be for one, none, or many LOCATIONs.
2) A LOCATION may have multiple CUSTOMERs.
3) A CUSTOMER may own multiple LOCATIONs.

21. STUDENT and ADVISORs ERD

```
STUDENT
StudentID
Name
Major
```

```
REGISTRATION
Semester
Year
```

```
ADVISOR
Advisor_ID
Name
Department
```

```
COURSE
CourseID
Title
Credits
```

```
Semester
Year
```

```
Assigned
```
22. In Figure 3-22, we have the following associative entities:

- **Does_business_in**: between SALES_TERRITORY and CUSTOMER
  Although this entity has no attributes and no independent meaning, it is the only way that Visio can represent the M:N relationship between SALES_TERRITORY and CUSTOMER.

- **Order_Line**: between PRODUCT and ORDER
  This relationship has an attribute: Ordered_Quantity that reflects the amount of product on each line of the order by the customer. It has independent meaning on the Customer’s Order.

- **Uses**: between PRODUCT and RAW MATERIALS
  This relationship has one attribute, Goes_into_quantity. It also may have independent meaning, although there is no obvious independent identifier.

- **Supplies**: between RAW MATERIALS and VENDOR
  Since there is an attribute on this entity and it can have independent meaning, it might be a good candidate to convert to an associative entity.

- **Produced_in**: between WORK CENTER and PRODUCT:
  Although this entity has no attributes and no independent meaning, it is the only way that Visio can represent the M:N relationship between WORK_CENTER and PRODUCT.

- **Works_in**: between WORK CENTER and EMPLOYEE
  Although this entity has no attributes and no independent meaning, it is the only way that Visio can represent the M:N relationship between WORK_CENTER and EMPLOYEE.

- **Has_Skill**: between EMPLOYEE and SKILL
  Although this entity has no attributes and no independent meaning, it is the only way that Visio can represent the M:N relationship between SKILL and EMPLOYEE.

There are so many associative entities because there are many M:N relationships that have independent meaning and because Visio’s templates cannot represent M:N relationships.
23. Wally's Wonderful World of Wallcoverings ERD:

Note:
The question does not indicate that there is a quantity for the Contains or Consists_of relationships.
24. Peck and Paw ERD:

```
ATTORNEY
  Attorney_ID
  Attorney_Name
  Attorney_Str_Addr
  Attorney_City
  Attorney_State
  Attorney_Zip_Code
  {Bar}
  {Specialty}

CASE
  Case_ID
  Case_Type
  Case_Desc

CLIENT
  Client_ID
  Client_Name
  Client_Str_Addr
  Client_City
  Client_State
  Client_Zip_Code
  Client_Phone
  ClientDOB

Assignment

COURT
  Court_ID
  Court_Name
  Court_City
  Court_State
  Court Zip_Code

Placed_in

Presides_over

JUDGE
  Judge_ID
  Judge_Name
  Years_in_Practice
```
25.

```
EMPLOYEE
  Employee_ID
  Employee_Name (...)
  Birth_Date

COURSE_COMPLETION
  Date_Completed

COURSE
  Course_ID
  Course_Title
  (Topic)

Notification

Date_of_Notification
```
26. a. The address attributes of employee, customer, and vendor do not currently contain the street, city, or state.
26b. There could be more than 1 product finish for a product, which could affect the price.
26c. Yes, this would be possible. For example, a customer could have more than 1 address.

27.

Suggestions for Field Exercises

1. The intent of this exercise is to have your students gain some exposure to standards in the business world. This is a good opportunity for your students to learn the benefits of enforcing naming standards, whether for E-R models or for programming code. If standards do not exist in the organization, have your students come up with some guidelines for naming standards. If standards do exist, your students should ask the database or systems analyst for an opportunity to review these standards to see if they are consistent and uniform.

2. You may choose to use the same organizations for this field exercise that were used in Field Exercise 4 in Chapter 1, or instead choose different organizations. It is likely that some of your students may have contacts in suitable organizations. The main difference that students are likely to find in a manufacturing company (compared to a service...
company) is the complexity encountered in modeling a product structure (or bill of materials). This often results in a recursive unary relationship, which is described in this chapter.

3. This field exercise can be performed in conjunction with Exercise 2 above. Most organizations will probably have examples of each of these types of relationships. Be on the alert to discover ternary relationships that are mistakenly modeled as multiple binary relationships.

4. This field exercise can be combined with Exercise 3 above. It is quite likely the organization will be using E-R notations that are different from the text, but students should be able to accommodate different notations with some explanation.

5. We suggest you combine this with Exercise 4 (and perhaps Exercise 3) above. If time-dependent data is apparent in the models, you might ask, for example, how the organization tracks customer sales over time.

6. Students should build a table to compare features of all products.

**Project Case**

**Case Questions**

1. Mountain View Community Hospital (MVCH) would want to use ER modeling to understand its data requirements because this approach will provide a pictorial depiction of MVCH’s business rules about data and how it is managed in the organization. The ER model provides a representation of these rules so they can be unambiguously understood by system developers and end users. The hospital might also want to model their requirements using the object-oriented model (see later chapter in text). Other possible diagrams might be data flow diagrams (DFD), state-transition diagrams, or use case diagrams.

2. No; Mountain View Community Hospital is an instance of the entity type HOSPITAL. Since there is only one instance, there is no need to model the HOSPITAL entity type.

3. a. BED may be a weak entity because it appears to require a Care_Center_ID attribute (per case description). MVCH may have a business rule requiring a BED to be assigned to a CARE_CENTER in order for the system to track the BED.

   b. There are no multivalued attributes.

   c. Between PATIENT and PHYSICIAN there are two relationships: refers and admits. Between EMPLOYEE and CARE_CENTER there are two relationships: has_assigned and nurse_in_charge.

4. At this stage in our understanding of E-R diagrams, we simply diagram the relationship...
(called Is_assigned) between PATIENT and BED as an optional 0-1 relationship. In Chapter 4 we will learn how to model the subtypes of PATIENT (IN PATIENT and OUT PATIENT) and then create a mandatory relationship between IN PATIENT and BED.

5. The only reason to split ITEM into two separate entities would be to track the use of reusable items. In other words, once an item is purchased and can be reused, one might want to see how frequently an item is used. In this case, one might wish to record the item serial number (or assign a number) and then see specifically how that item was used. However, I still think that it would be of merit to track reusable items in the general sense. For example, you might want to know that you have 100 forceps in stock. If one gets damaged and is thrown away, the inventory is reduced. Once the inventory reaches a certain level, more forceps can be ordered.

6. I would take a look at all user views by examining reports and screens from any existing systems. I would then compare these to the data model and make a determination of whether this data model will support the system’s generation of reports and screens.

Case Exercises

1. Some other questions we might like to ask are the following:
   a. Should we model pharmaceutical items separately from ITEM since such items are prescribed by a physician for a patient?
   b. Is there a need to maintain a historical record of a patient’s relationship with the hospital? If so, how can this be modeled in the E-R diagram?
   c. Need we model the various subtypes of EMPLOYEE (nurses, staff, physicians, etc.)?
   d. Is there a need to model the relationship with other persons such as volunteers and donors?

   You should ask your students to develop additional questions.
2. No. The entity type ITEM has a Unit-Cost attribute, but has no provision to represent a unit cost per day, which would be required for items such as in-room TVs.

Published by www.emagazine.org
Chapter 3

4.

Notes:
1) An EMPLOYEE may serve as a Nurse_in_charge for 0, 1, or many CARE_CENTERs.
2) Visit_ID is unique (surrogate) identifier for ASSESSMENT.
3) Comments attribute for ASSESSMENT is used to record reasons for visit & symptoms.
4) Bed_ID is composite (aggregate) identifier for BED; it is shown as a partial identifier as Care_Center_ID will be needed to complete the unique identifier and Care_Center_ID will be added in a later design step.

Published by www.emagazine.org
5.

Notes:
1) An EMPLOYEE may serve as a Nurse_in_charge for 0, 1, or many CARE_CENTERs.
2) Visit_ID is unique (surrogate) identifier for ASSESSMENT.
3) Comments attribute for ASSESSMENT is used to record reasons for visit & symptoms.
4) Bed_ID is surrogate composite identifier for BED; it is shown as a partial identifier as Room_No will be needed to complete the unique identifier and Room_No will be added in a later design step.

7. Yes. The model records the date, time, and results for each treatment occurrence performed by a physician on behalf of a patient.
Project Assignments
P1.

Notes:
1) An EMPLOYEE may work for 0, 1, or many CARE CENTERS.
2) Bed_ID is unique (surrogate) identifier for ASSESSMENT.
3) Comments attribute for ASSESSMENT is used to record reasons for visit & symptoms.
4) Bed_ID is surrogate composite identifier for BED; it is shown as a partial identifier as Room_No will be needed to complete the unique identifier and Room_No will be added in a later design step.
5) A DIAGNOSTIC_UNIT performs one to many TREATMENTS; a TREATMENT is performed by 1 DIAGNOSTIC_UNIT.
A FACILITY can contain one or more CARE_CENTERS or may contain no CARE_CENTERS. A CARE_CENTER is part of one and only one FACILITY.

A FACILITY may maintain one or more DIAGNOSTIC UNITS or may maintain no DIAGNOSTIC UNITS. A DIAGNOSTIC UNIT is part of only one FACILITY.

A CARE_CENTER has many EMPLOYEES. Each CARE_CENTER has one EMPLOYEE assigned as a nurse_in_charge. Each EMPLOYEE may work for one or more CARE_CENTERS.

A CARE_CENTER will contain one or more ROOMs. Each ROOM is contained in only one CARE_CENTER.

A ROOM may contain one or more beds or may contain no BEDS. A BED is contained in only one ROOM.

A DIAGNOSTIC_UNIT performs one or more TREATMENTS. A TREATMENT is performed by only one DIAGNOSTIC UNIT.

A BED is assigned to one patient or no patients. A PATIENT is assigned to one BED or no BEDS.

A PHYSICIAN admits one or more PATIENTS or admits no PATIENTS. A PATIENT is admitted by only one PHYSICIAN.

A PHYSICIAN may refer one or more PATIENTS or may refer no PATIENTS. A PATIENT must be referred by one PHYSICIAN.

A PATIENT may consume many ITEMS or may consume no ITEMS. An ITEM is consumed by one or more PATIENTS or may be consumed by no PATIENTS.

An ITEM is supplied by one or more VENDORS. A VENDOR may supply one or more items or may supply no ITEMS.

A PHYSICIAN may write one or more ORDERS or may write no ORDERS for one PATIENT. An ORDER is written by one PHYSICIAN.

An ORDER may consist of one or more ITEMS or no ITEMS. An ITEM may be part of one or more ORDERS or may be part of no ORDERS.

An ORDER may consist of one or more TREATMENTS or no TREATMENTS. A TREATMENT may be part of one or more ORDERS.

A PHYSICIAN may complete one or more DIAGNOSES for one or more PATIENTS. A DIAGNOSIS is completed for one PATIENT by one PHYSICIAN.

A VENDOR may supply one or more ITEMS. Each ITEM may be supplied by more than one VENDOR.

An EMPLOYEE completes one, none, or many ASSESSMENTs of a PATIENT. Each PATIENT may have one or many ASSESSMENTs over time at this hospital.

A FACILITY may prepare multiple staffing schedules for its PHYSICIANs. Each SCHEDULE instance is for a single FACILITY and a single PHYSICIAN. A PHYSICIAN may have zero, one, or many SCHEDULEs.
P3. [Sample questions are listed below; student answers may vary]

a. How is patient billing done?
b. What reporting requirements does the administration have?
c. Should there be a distinction between a diagnostic test, a procedure, and a treatment?
d. Can a physician choose which diagnostic unit to use for a test?
e. How will we handle referrals by physicians who are not on staff?
f. How will medical records be modeled?
g. Should the relationship between patient and bed contain a start and end date?
Chapter 4  The Enhanced E-R Model and Business Rules

Chapter Overview

The purpose of this chapter is to present some important extensions to the E-R model (described in Chapter 3) that are useful in capturing additional business meaning. In particular, we describe two types of extensions to the E-R model. First, the enhanced entity-relationship (EER) model includes constructs for supertype/subtype relationships. Second, the inclusion of new notation for business rules allows the designer to capture a broader range of constraints on the data model than were previously available.

Chapter Objectives

Specific student objectives are included in the beginning of the chapter. From an instructor’s point of view, the objectives of this chapter are to:

1. Introduce the concept of supertype/subtype relationships, and prepare the student to recognize when to use these relationships in data modeling.
2. Describe the use of specialization (top-down perspective) and generalization (bottom-up perspective) as complementary techniques for defining supertype/subtype relationships.
3. Introduce notation for specifying both completeness constraints and disjointness constraints when modeling supertype/subtype relationships.
4. Help students gain sufficient perspective so that they recognize when to use (and when not to use) supertype/subtype relationships in realistic business situations.
5. Describe the basic premises of a business rules paradigm.
7. Introduce notation for modeling typical operational constraints that can be incorporated in your EER diagram.
8. Discuss the universal data model.

Key Terms

<table>
<thead>
<tr>
<th>Action</th>
<th>Derived Fact</th>
<th>Specialization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action assertion</td>
<td>Disjoint rule</td>
<td>Structural assertion</td>
</tr>
<tr>
<td>Anchor object</td>
<td>Disjointness constraint</td>
<td>Subtype</td>
</tr>
<tr>
<td>Attribute inheritance</td>
<td>Enhanced entity-relationship</td>
<td>Subtype discriminator</td>
</tr>
<tr>
<td>Completeness constraint</td>
<td>Entity cluster</td>
<td>Supertype</td>
</tr>
<tr>
<td>Corresponding object</td>
<td>Generalization</td>
<td>Supertype/subtype hierarchy</td>
</tr>
<tr>
<td>Derivation</td>
<td>Overlap rule</td>
<td>Total specialization rule</td>
</tr>
<tr>
<td></td>
<td>Partial Specialization rule</td>
<td>Universal data model</td>
</tr>
</tbody>
</table>
Classroom Ideas

1. Introduce the concept of supertypes and subtypes with a familiar example, such as VEHICLE (subtypes are CAR, TRUCK, SUV, etc.).
2. Introduce the basic notation for supertype/subtype relationships (Figure 4-1). Use this notation to represent the example you introduced in (1). Introduce your students to all three types of notation.
3. Discuss the EMPLOYEE example with subtypes (Figure 4-2). Use this figure to introduce the example of attribute inheritance.
4. Use Figure 4-3 to discuss the two major reasons for introducing supertype/subtype relationships: unique attributes among subtypes, and unique subtype relationships.
5. Contrast generalization and specialization using Figures 4-4 and 4-5. Have your students suggest other examples that use each of these techniques.
6. Introduce the completeness constraint using Figure 4-6. Give other examples where either the total specialization rule or the partial specialization rule is more appropriate.
7. Discuss the disjointness constraint and related notation using Figure 4-7. For reinforcement, have the students work Problem 7 (Problems and Exercises) in class.
8. Introduce notation for a subtype discriminator (Figures 4-8 and 4-9). Discuss why a different notation is required for the two cases shown in these figures.
9. Discuss entity clustering and illustrate with Figures 4-13 and 4-14.
10. Review the extended example of a supertype/subtype hierarchy shown in Figure 4-10. For reinforcement, ask the students to work problem 2 (Problems and Exercises) in class.
11. Review universal data models and discuss how these are being used more widely today. Consider inviting an industry guest speaker to discuss how these universal data models are utilized in his/her company. If your students have access to computers during your class session, break the students into small groups and have them complete Problem & Exercise 16 then report back to the large class with their findings.
12. Discuss the classification of business rules shown in Figure 4-18. Give examples for each type of rule.
13. Discuss the notion of structural assertions, and illustrate with examples shown in Figure 4-19.
14. Introduce the notion of action assertions, and illustrate with the examples shown in Figure 4-20. For this example, show how the business rule is captured with notation on the diagram.
15. Ask your students for examples of other business rules they have encountered recently in their work, school, or home experience. See if they can diagram these rules using the notation provided in this chapter.
Answers to Review Questions

1. Define each of the following terms:
   a. **Supertype.** A generic entity type that has a relationship with one or more subtypes.
   b. **Subtype.** A subgrouping of the entities in an entity type that is meaningful to the organization.
   c. **Specialization.** The process of defining one or more subtypes of the supertype, and forming supertype/subtype relationships.
   d. **Entity cluster.** A set of one or more entity types and associated relationships grouped into a single abstract entity type.
   e. **Structural assertion.** A statement that expresses some aspect of the static structure of the organization.
   f. **Anchor object.** A business rule (a fact) on which actions are limited.
   g. **Subtype discriminator.** An attribute of the supertype whose values determine the target supertype or subtypes.
   h. **Total specialization rule.** Specifies that each entity instance of the supertype must be a member of some subtype in the relationship.
   i. **Generalization.** The process of defining a generalized entity type from a set of more specialized entity types.
   j. **Disjoint rule.** Specifies that if an entity instance (of the supertype) is a member of one subtype, it cannot simultaneously be a member of two (or more) subtypes.
   k. **Overlap rule.** Specifies that an entity instance can simultaneously be a member of two (or more) subtypes.
   l. **Action assertion.** A statement of a constraint or control on the actions of the organization.
   m. **Universal data model:** A generic or template data model that can be reused as a starting point for a data modeling project.

2. Match the following terms and definitions:
   h  supertype
   k  entity cluster
   i  structural assertion
   a  subtype
   j  specialization
   d  anchor object
   f  action
   l  subtype discriminator
   c  attribute inheritance
   b  overlap rule
   e  corresponding object
   g  derived fact
3. Contrast the following terms:
   a. **Supertype; subtype.** A supertype is a generalized entity type that has one or more subtypes, while a subtype is a subgrouping of the entities in a supertype.
   b. **Generalization; specialization.** Generalization is the process of defining a generalized entity type from a set of more specialized entity types, while specialization is the process of defining one or more subtypes of the supertype.
   c. **Anchor object; corresponding object.** An anchor object is a business rule on which actions are limited, whereas a corresponding object influences the ability to perform an action on another business rule.
   d. **Disjoint rule; overlap rule.** With the disjoint rule an instance of a supertype must be a member of only one subtype at a given time. With the overlap rule an instance of a supertype may simultaneously be a member of two or more subtypes.
   e. **Structural assertion; action assertion.** A structural assertion expresses the static structure of an organization whereas an action assertion is a statement of control on the actions of an organization.
   f. **Total specialization rule; partial specialization rule.** With the total specialization rule, each instance of the supertype must be a member of some subtype in the relationship. With the partial specialization rule, an instance of the supertype is allowed not to belong to any subtype.
   g. **PARTY;PARTY ROLE:** In a universal data model, PARTY represents persons and organizations independent of the roles they play whereas PARTY ROLE contains information about a party for an associated role.

4. Two conditions that indicate a designer should consider using supertype/subtype relationships:
   a. There are attributes that apply to some (but not all) of the instances of an entity type.
   b. There are relationships that apply to some (but not all) of the instances of an entity type.

5. The reasons for entity clustering are:
   a. Complex enterprise-wide E-R diagram.
   b. Ability to have a hierarchical decomposition of data model.
   c. Desire to focus part of the model on an area of interest to some community of users.
   d. Ability to create several different entity clusters each with a different focus.

6. An example of a supertype/subtype relationship:
The supertype PERSON has many possible subtypes: MALE, FEMALE, INFANT, TEENAGER, etc, assuming these different types of persons have somewhat different attributes or participate in different relationships. In an organizational context, PERSON may have subtypes of EMPLOYEE, CONTRACTOR, CUSTOMER, VENDOR, MANAGER, etc.
7. Give an example of a(n):
   a. Structural assertion: a person is a customer of a store.
   b. Action assertion: a customer must have a credit limit greater than zero in order to charge a purchase.

8. Action assertions can be classified in three ways. Each of the three ways has several types associated with it:
   a. Based on the type of result from the assertion:
      i. Condition: states that if something is true, than another business rule will apply.
      ii. Integrity constraint: states something that must always be true.
      iii. Authorization: states a privilege.
   b. Based on the form of the assertion:
      i. Enabler: if true, leads to the existence of the corresponding object.
      ii. Timer: enables or creates an action.
      iii. Executive: causes the execution of one or more actions.
   c. Based on the rigor of the assertion:
      i. Controlling: states that something must or must not happen.
      ii. Influencing: items of interest for which a notification must occur.

9. An example of an enabler action assertion: an order can be entered into the system once the customer’s credit has been verified.

10. An example of an executive assertion: assign a contract number to a vendor the first time that the vendor supplies raw materials to Pine Valley Furniture.

11. Attribute inheritance is the property that subtype entities inherit values of all attributes of their supertype(s). This property is important because it makes it unnecessary to include supertype attributes redundantly with subtypes.

12. Give examples of:
   a. Supertype/subtype relationship where the disjoint rule applies: PERSON has subtypes MALE and FEMALE.
   b. Supertype/subtype relationship where the overlap rule applies: PERSON has subtypes INSTRUCTOR and STUDENT.

13. The types of business rules that are normally captured in an EER diagram include terms, relationship constraints, and supertype/subtype relationships (see Figure 4-11).

14. The purpose of a subtype discriminator is to determine the target subtype (or subtypes) for each instance of a supertype.

15. A packaged data model is most useful when one can easily customize it to the specific business (that is, the organization is very similar to other organizations for the same industry or purpose or the functional area is roughly the same as that functional area in other organizations). As long as the packaged data model is for the type of business or functional area, then it can generally be customized. The amount of customization depends upon the types of specialized business rules in place for the organization.
16. **Note to instructor**: After registering (free) at Bill Inmon’s site with an e-mail and password, you need to select the Data Models link. Several models are available for review.

Two packaged data models appear as possible candidates for adaptation to the PVFC situation. The PVFC appears to combine elements from both the SALES and MANUFACTURING packaged data models. The SALES data model would need to be modified to include PVFC sales territories and salespersons. The SALES data model might also have to be adjusted to reflect Vendor provision of Raw Materials that PVFC assembles into Products that are sold. The SALES data model would also need to track PVFC employees and skills.

The MANUFACTURING data model appears to capture the PVFC data related to transforming Raw Material into Products, but seems to lack the tracking of the Order and Sales data. Overall, the SALES packaged data model seems to be the most promising for adaptation to PVFC’s situation.

17. A subtype/supertype hierarchy is useful when you have several subtypes that are also supertypes. An example would be for bank accounts. At the first level, you can have savings, checking and loans. Underneath loans, there are several subtypes, including personal, auto, home, etc.

18. A member of a supertype is always a member of at least one subtype when the rule of total specialization applies to an EERD.

19. 
   a. An order contains many products and a product can be part of many orders. This fact is shown as the order_line associative entity in Figure 4-12.
   b. An employee can be either management or union, but not both. This is shown by a subtype/supertype relationship in Figure 4-12.
   c. A union employee can work in many work centers and a work center can have many union employees. This is shown by the Works_in relationship in Figure 4-12.
   d. A regular customer does business in a sales territory, while a national customer does not do business in a particular sales territory. This is shown in the subtype/supertype relationship for Customer in Figure 4-12.
   e. An employee can have many skills. This is shown as a multivalued attribute in Figure 4-12.

20. A few derived facts for Pine Valley Furniture:
   a. An order is supplied by one or more suppliers. This can be derived from the order_line, uses and supplies relationships in Figure 4-12.
   b. The order total is calculated by adding the product of the quantity times the price for each order_line for a given order.
   c. An order is produced in one or more work centers. This can be derived from the produced_in relationship and order_line associative entity in Figure 4-12.
d. An order is worked on by one or more employees. This can be derived from the
produced_in relationship, the order_line associative entity and the works_in
relationship in Figure 4-12.

e. An order includes one or more product lines. This can be derived from the
order_line associative entity and the includes relationship in Figure 4-12.

Solutions to Problems and Exercises

1. An example listing follows for a GRADUATE STUDENT:

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Data Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSN</td>
<td>736-94-1802</td>
</tr>
<tr>
<td>Name</td>
<td>Jessica James</td>
</tr>
<tr>
<td>Address</td>
<td>25 Lake Dr. Medford OR 95106</td>
</tr>
<tr>
<td>Gender</td>
<td>female</td>
</tr>
<tr>
<td>Date_of_Birth</td>
<td>Oct. 23, 1967</td>
</tr>
<tr>
<td>Major_Dept</td>
<td>Computer Science</td>
</tr>
<tr>
<td>Test_Score</td>
<td>986</td>
</tr>
</tbody>
</table>
2.

```
PERSON

SSN
Name
Address
Gender
Date_of_Birth
Person_Type
(Employee?, Student?, Alumnus?)

Employee?="Y"

ALUMNUS

Degree
(Degree, Year, Designation)

Alumnus?="Y"

STUDENT

Major, Dept
Student_Type

Student?="Y"

EMPLOYEE

Date_Hired
Salary
Employee_Type

Employee_Type=

FACULTY

Rank

Faculty"

STAFF

Position

Staff"

GRADUATE

Student
Test_Score

Grad"

UNDERGRAD

Class_Standing

Undergrad"
```
3. Please note that the problem does not explicitly state that Skill is a multivalued attribute. Given the fact that examples in the text have skill as a multivalued attribute, we have made this assumption here also.

EER Notation:
3) Visio Notation:
3) Subtypes inside Supertypes Notation
4. **Note**: Again, we have assumed that Skill is a multivalued attribute.

Standard EER Notation:
4) Visio Notation:
4) Subtypes inside Supertypes Notation:

```
PERSON
SSN
Name
Address
City
State
ZipCode
Telephone
Person_Type

EMPLOYEE
Date_Hired

VOLUNTEER

DONOR

SKILL
Maintains

ITEM
Donates
```
5. No, none of the vehicle classifications has a unique attribute or a unique relationship. Instead, Vehicle category likely should be an attribute of the Vehicle entity type.

6a.

6b.
6c.

```
PERSON

• Camper
• Biker
• Runner
```

6d.

```
PERSON

0

• Camper
• Biker
• Runner
```
7. Standard EER Notation:

7) Visio Notation:
7) Subtype in Supertype Notation:

```
ACCOUNT
Acct_No
Date_Opened
Balance
Account_Type

CHECKING
   Service_Charge

SAVINGS
   Interest_Rate_Sav

LOAN
   Interest_Rate_Loan
   Payment
```
8. **a. Sample definitions:**

- **EMPLOYEE:** A person who has signed an employment agreement or contract with the company.
- **HOURLY EMPLOYEE:** An employee whose pay is based on number of hours worked.
- **SALARIED EMPLOYEE:** An employee who receives a fixed salary each pay period.
- **CONSULTANT:** An employee who has signed an employment contract and whose pay is based on an agreed billing rate.
- **Employee_Number:** An employee’s social security number.
- **Employee_Name:** An employee’s name consisting of first name, middle initial, and last name.
- **Address:** An employee’s home address, consisting of street address, city, state, and zip code.
- **Date_Hired:** The date when an employee signed an employment agreement or contract.
- **Hourly_Rate:** The pay rate ($/hour) for an hourly employee.
- **Annual_Salary:** The base annual salary for a salaried employee.
- **Stock_Option:** The annual compensation (shares/year) of company stock for a salaried employee.
- **Contract_Number:** The number on the employment contract signed by a consultant.
- **Billing_Rate:** The compensation ($/hour or other stated period) on the employment contract signed by a consultant.

**b. Sample integrity-constraint action assertions:**

- **Employee_Number:** Each employee must have a unique employee number.
- **Employee_Name:** Each employee must have a name.
- **Address:** Each employee must have an address.
- **Date_Hired:** Each employee must have a date of hire earlier than or equal to today’s date.
- **Hourly_Rate:** Hourly employees must have an hourly rate which must be between $1 and $100.
- **Annual_Salary:** Salaried employees must have an annual salary between $1 and $999,999.
- **Stock_Option:** Salaried employees must have a value for stock option between 0 and 10,000.
- **Billing_Rate:** Consultants must have a billing rate between $0 and $999.
- **Contract_Number:** Consultants must have a contract number.
9. a. and b.

10. a. Anchor object: resident patient

Published by www.emazine.org
b. Corresponding object: responsible physician

11. Business rule: “A student may attend a concert only if that student has completed his/her homework”:
   a. and b.

   ![Diagram]

   c. Anchor object: student
   Corresponding object: Has_completed (homework)

12. a. Sample definitions:
   PATIENT: a person who has been admitted to the hospital, or to a treatment program administered by the hospital.
   OUTPATIENT: a person who has been admitted to a program of treatment administered by the hospital.
   RESIDENT PATIENT: a person who has been admitted for a stay in the hospital and assigned to a bed location.
   RESPONSIBLE PHYSICIAN: a physician who has formally admitted to patient to the hospital.
   BED: a hospital bed located within a room in the hospital.
   Is_cared_for: the relationship between a physician and a patient admitted to the hospital by that physician.
   Is_Assigned: the relationship between a resident patient and the hospital bed to which that patient is assigned.
   Patient_ID: a patient’s social security number.
   Patient_Name: a patient’s first and last name.
   Admit_Date: the date when a patient was most recently admitted to the hospital or to a treatment program.
   Checkback_Date: the date when an outpatient is scheduled for a return visit.
   Date_Discharged: the date when a resident patient was discharged following the most recent stay in the hospital.
   Physician_ID: a unique identification number for an admitting physician.
Bed_ID: a unique identification number for each hospital bed.
b. Sample integrity constraint action assertion:
   Patient_ID
   Each patient must have a unique patient ID.
   
   Patient_Name
   Each patient must have a first and last name.
   
   Admit_Date
   Each patient must have an admit date which is a date no greater than today.
   
   Checkback_Date
   Each outpatient must have a checkback date which is greater than or equal to today’s date and can be null.
   
   The checkback_date cannot be before the admit_date.
   
   Date_Discharged
   Each resident patient must have a discharge date.
   The date discharged cannot be before the admit_date.
   
   Physician_ID
   Each responsible physician must have a unique physician id.
   
   Bed_ID
   Each bed must have a unique bed ID.
   
13. a. Because only regular customers (as opposed to national customers) do business in a sales territory, then not all instances of the customer entity cluster do business in a selling unit. However, because all sales territories do business with at least one regular customer, then all sales territories do business with at least one instance of a customer entity cluster.
   
   b. The attributes of item would be the attributes of PRODUCT and PRODUCT LINE from Figure 3-22: Product_ID, Product_Description, Product_Finish, Standard_Price, Product_line_ID, Product_Line_Name.
   
   c. The attributes of material would be the attributes of RAW MATERIAL, Supplies, SUPPLIER, and VENDOR from Figure 3-22: Vendor_ID, Vendor_Name, Vendor_Address, Contract_Number, Supply_Unit_Price, Material_ID, Unit_of_Measure, Material_Name, Standard_Cost.
14. E-R Diagram from Chapter 3 Problem 7 with Entity Clusters:
14 (continued)

The Sales_Unit cluster can be used by people only interested in how the business is managed, without concern for the properties listed. The Property_Listing cluster can be used by people who are interested in property that is currently listed or who owns that property.

15. E-R Diagram from Chapter 3, Problem 12 with Entity Clusters:

There are three entity clusters: Project_Detail, Employee_Detail, and Dept_Detail. Project_Detail contains the set of entities that would be used by one interested in the project without concern for the specific employees on the project. An assumption is that the only concern from the project side is to track employee skills and location, not individual employees.

The Employee_Detail cluster would be of most value to the user who was interested in what skills specific employees have as well as location. Other details are available in this cluster, such as marriage. This cluster was chosen since one can then isolate employee information without looking at project information.

The Dept_Detail cluster was chosen since one might not be concerned about vendors, however one might want to know what department a given employee works for. In the same way, one might want specifics about vendors without needing information about employees or projects.
16. **Notes to instructor**: Student answers may vary depending upon their interpretation of the exercise situation. The solutions presented here are representative of possible answers, given certain assumptions. These notes are presented here for your reference:

a) An alternative solution could be to set ORGANIZATION as a supertype with INTERNAL and EXTERNAL subtypes. However, the exercise does not indicate unique attributes or special relationships between the proposed subtypes and any other entities, thus no supertype-subtype hierarchy is shown in these solutions. More questions would need to be asked of the end users to determine the appropriate model for this situation.

b) Another possible solution to this exercise would be to create a supertype of PARTY with subtypes of PERSON and ORGANIZATION. There is significant overlap of attributes between these two entity types that could suggest such a solution.

**EER Notation:**

[Diagram of EER notation showing relationships between PERSON, EMPLOYMENT, POSITION HISTORY, POSITION, and ORGANIZATION entities]
16) Visio Notation:
16) Subtypes within Supertypes Notation:

PERSON
- Person_ID
- Person_Name
- Person_Address
- Person_Phone
- Person_DOB

EMPLOYMENT
- Emp_Start_Date
- Emp_Term_Date
- Emp_Bonus

ORGANIZATION
- Org_ID
- Org_Name
- Org_Address
- Org_Phone
- Org_Budget_No

POSITION_HISTORY
- Hist_Start_Date
- Hist_Term_Date
- Hist_Salary

POSITION
- Position_ID
- Position_Title

Responsible_for
17. **Note to instructor:** This Problem & Exercise has a different written scenario than a similar one in Chapter 3. The plural “requested judgment characteristics” in Chapter 3 is semantically different than this exercise’s “requested judgment characteristic” which results in the alternate model solution shown below. This may be useful to point out to students regarding the importance of paying attention to fine details while modeling data.

![Database Diagram](image)

**Notes:**
1) Person_ or Org attribute denotes Person or Organization type of Legal Entity. There is no reason to show Person and Organization as subtypes of Legal Entity, as there are no special attributes or relationships identified.
2) The same legal entity cannot be both a Plaintiff and Defendant in the same Case.
3) Although DEFENDANT has no other unique attributes, it is required as a subtype to show the parties involved in a CASE. Further, the DEFENDANT subtype is necessary to show the Brought_Against role that is necessary to defining the parties in a CASE.
18. EER Notation
18) **Visio Notation:**

![Database ER Diagram](image-url)
18) **Subtype within Supertype Notation:**
19. **EER Notes (for all notations):**

1) A CONSULTANT is either a Business or Technical consultant, not both.

**EER Notation:**

![EER Diagram](image-url)
19) **Visio notation:**

![Database Diagram](image-url)
19) Subtype within Supertype Notation:

![Database diagram with entities and relationships]
20. a. Sample Definitions

CONSULTANT: a person who has signed an employment agreement or contract with the company, and who is on the company payroll.

BUSINESS CONSULTANT: a consultant who provides an estimate to a customer.

TECHNICAL CONSULTANT: a consultant who provides security services to a customer.

CUSTOMER: a business that requires security services.

LOCATION: one or more places of business for a customer.

SERVICE: a security service that can be performed.

Estimates: a written estimate prepared by a business consultant for a location.

Services_Performed: actual services performed by a technical consultant at one location.

Emp_ID: a consultant’s employee id.

Degree: a consultant’s academic training.

Business Experience: a business consultant’s business experience.

Tech Skills: a technical consultant’s experience.

Coverage: how much of an area a service covers for a given location.

b. Sample integrity-constraint action assertions:

Emp_ID:

- each consultant must have a unique employee id.

Employee_Name:

- each consultant must have a name.

Address:

- each consultant must have an address.

Degree:

- consultants can have one, none or many degrees.

Business Experience:

- business consultants can have one, none or many types of business experience. For each type of experience, we also record the years.

Amount:

- both services performed and estimates can have a dollar value associated with the service or estimate.
21.

a.

b.
e.

```
PART

<table>
<thead>
<tr>
<th>Part_No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
</tr>
<tr>
<td>Location</td>
</tr>
<tr>
<td>Qty_on_Hand</td>
</tr>
<tr>
<td>Part_Type</td>
</tr>
<tr>
<td>(Manufactured?, Purchased?)</td>
</tr>
</tbody>
</table>

Part_Type:

- Manufactured? = "Y"
- Purchased? = "Y"

MANUFACTURED

<table>
<thead>
<tr>
<th>PART</th>
</tr>
</thead>
<tbody>
<tr>
<td>Checkbox_Date</td>
</tr>
</tbody>
</table>

Purchased?

PURCHASED

<table>
<thead>
<tr>
<th>PART</th>
</tr>
</thead>
</table>

SUPPLIER

| Supplier_ID |

SUPPLIES

| Unit_Price |
```
Notes:
1) A PERSON, in his/her EMPLOYMENT, may hold multiple POSITIONs or not yet have an assigned POSITION (this is shown with the 0:M cardinality near POSITION from EMPLOYMENT).
2) A POSITION might initially be unfilled, or over time, may be filled with multiple EMPLOYMENT instances of PERSONs (this is shown with the 0:M cardinality near EMPLOYMENT from POSITION).
23.

**Notes for EERD:**

1) Member_Type values are Golf or Non-Golf.
2) Social and Tennis members are considered Non-Golf members. A Social member has a Golf_Rounds_Limit of 2 and Tennis_Courts?=N. A Tennis member has a Golf_Rounds_Limit of 4 and Tennis_Courts?=Y. A Golf member has a Golf_Rounds_Limit of 999 and Tennis_Courts?=Y.
3) Golf members’ visits are tracked only if they bring a guest.
4) If a Guest becomes a Member, then Guest records are archived out of the database.
5) Member_Date tracks the membership date of the Member.
24.

**Notes for EERD:**
1) Owners wish to know the attendance and price_charged for each TIMESLOT (i.e., there is a charge with an attendance to see everything shown on a SCREEN in the same TIMESLOT).
2) Movie_Seq_No tracks the sequence in which movies are shown in the TIMESLOT (e.g., in a timeslot there might be two trailers, followed by two commercials, followed by a feature film, and closed with a commercial).
Suggestions for Field Exercises

1. Common examples of EER constructs:
   a. Supertype/subtype relationships. Ask your students to try to try to find an example of each of the rules described in the chapter: disjoint, overlapping, partial specialization, and total specialization. Also, for each example, have your students identify a candidate subtype discriminator. Ask your students to justify the use of supertype/subtype relationships for each of these examples, using the guidelines stated in the chapter.
   b. Business rules. For each example, ask your students to first state the rule using structured English (following the format described in the chapter), then draw an E-R diagram segment with the rule superimposed on the diagram.

2. We suggest that you use this exercise as a continuation of Field Exercise 2 in Chapter 3. Ask your students to determine whether supertype/subtype relationships are formally modeled in the corporate E-R diagrams. Also, ask your students to determine how business rules are stated and enforced by each organization.

3. We suggest you assign this exercise in conjunction with Field Exercise 4 in Chapter 3.

4. Following are several questions that can be used to structure this report:
   a. How are business rules defined?
   b. Why are business rules important to an organization?
   c. What are alternative methods for capturing and expressing business rules?
   d. What advantages can an organization realize by formally capturing business rules?

5. One good place for students to begin would be to perform a google search on business rules engine software. This will bring them to several sites, such as www.tcs.com

6. One good Web site for students to go to in their research is www.universaldatamodels.com
Project Case

Case Questions

1. Yes, the ability to model supertype/subtype relationships is likely to be very important for a hospital. A modern hospital is a triumph of specialization. Many hospital entities are likely to have subtypes, for example:
   ITEM: possible subtypes are Supply, Item, and Prescription Item
   PATIENT: possible subtypes are Inpatient and Outpatient
   TEST: possible subtypes are Scan and Blood test
   PROCEDURE: possible subtypes are Biopsy and Surgical

2. Yes, the business rules paradigm can be used for competitive advantage. Business rules allow a business to change its processes and procedures quickly in responding to environmental changes. Hospitals are under intense cost pressures and new government regulations. Process improvements and regulations, if implemented through clear business rules, will make it easier to react to changes.

3. Yes, the entity VISIT (scheduled for outpatients) is clearly a weak entity.

4. A hospital has many business rules. Two examples are the following:
   a. A patient cannot be admitted to the hospital without a referral from a responsible physician.
   b. A nurse can be reassigned to a different care center only by permission of the nurse in charge of the care center where the nurse is presently assigned.

5. A universal data model for Mountain View Community Hospital would work out well, since hospital applications are quite common. Of course, there would still need to be customization. One justification would be to look at the cost versus the cost savings from developing a database application completely from scratch.
Case Exercises

1. [Note to instructor: When assigning this exercise to students, be sure to allow a sufficient amount of time for completion. The case scenario is fairly complex and will encourage students to do a lot of thinking and experimenting prior to developing a workable diagram. Due to the large size of the diagram, and the necessary notes to the diagram, this solution is presented in three parts: notes, diagram (without attributes displayed), and a “diagram” of the entities with all attributes displayed.]

Business Rules/Notes for EERD (diagram on next page):

1. Only one entry for the Patient’s Emergency Contact Information is stored in the database.
2. Only the primary insurance information for the Patient is stored in the database. Secondary insurance information (if provided by the Patient) will be stored in paper files.
3. Referring/Primary Care Physician Contact information is stored as part of the PHYSICIAN entity in the database.
4. MVCH wants to track the history of all volunteer assignments within the facility, thus there is a need to use an associative entity (VOL_SERV_HISTORY) to track over time the various volunteer assignments, and each assignment’s supervisor, as well as the total amount of hours worked on each assignment. Some WORK_UNITs do not have VOLUNTEERs, thus a 0:M cardinality is required on its relationship with the associative entity.
5. A VOLUNTEER may be supervised by an EMPLOYEE or a PHYSICIAN at one time. Only one of these “supervision” relationships is active at one point in time, although both are shown on the diagram.
6. MVCH wants to track VOLUNTEER information at the point of application, thus it is possible that a VOLUNTEER instance may not yet have a corresponding instance of VOL_SERV_HISTORY in the database system (this is why there is a 0:M cardinality nearest the VOL_SERV_HISTORY associative entity on the relationship from VOLUNTEER).
7. A Registered Nurse (RN) may direct one, none, or several Licensed Practical Nurses (LPNs); a single LPN will be directed by only one RN. MVCH wishes to track these RN direction responsibilities for accountability and quality control purposes within the hospital.
8. A Floater nurse is not assigned to a CARE_CENTER, thus this is why there is a 0:M cardinality shown near CARE_CENTER on the Assigned relationship. If discussion with end users indicate additional attributes or relationships that are associated with Floater nurses, then an alternative solution could be to establish a FLOATER subtype entity on the diagram. The addition of this FLOATER subtype could allow the cardinality of the Assigned relationship for the NURSE supertype to be changed to 1:M.
9. Skill is modeled as a multivalued attribute of TECHNICIAN as it only relates to this entity and has no additional characteristics mentioned in the case. Under different assumptions, an alternative solution could be to model Skill as its own entity type with a relationship to TECHNICIAN (and possibly other entity types in the model).
10. Only current STAFF and TECHNICIAN assignments to WORKCENTERs are necessary for this case.
11. Only current BED to RESIDENT associations are necessary in the database.
CE1: EERD (Note: attributes by entity are shown on next page; attributes omitted here to conserve space)
CE1: Attributes listed by entity

<table>
<thead>
<tr>
<th>PERSON</th>
<th>PHYSICIAN</th>
<th>EMPLOYEE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Person_ID</td>
<td>DEA_No</td>
<td>Date_Hired</td>
</tr>
<tr>
<td>Person_Name</td>
<td>Pager_No</td>
<td>Emp_Type</td>
</tr>
<tr>
<td>Person_St_Address</td>
<td>Phys_Specialty</td>
<td></td>
</tr>
<tr>
<td>Person_City</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Person_State</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Person_Zip</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Person_Home_Phone</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Person_Work_Phone</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PersonDOB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Person_Email</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Person_Type (Physician?, Employee?, Volunteer?, Patient?)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PATIENT</th>
<th>VOLUNTEER</th>
<th>TECHNICIAN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contact_Date</td>
<td>Felony?</td>
<td>(Tech_Skill)</td>
</tr>
<tr>
<td>Emer Contact_Info</td>
<td>Felony_Explanation</td>
<td></td>
</tr>
<tr>
<td>(Last_Name, First_Name, Relationship, Address, Phone)</td>
<td>V_Emer Contact_Info</td>
<td></td>
</tr>
<tr>
<td>Insurance_Info</td>
<td>VEC_Last_Name, VEC_First_Name, VEC_Role</td>
<td></td>
</tr>
<tr>
<td>(Company_Name, Policy_No, Group_No, Company_Phone)</td>
<td>VEC_Address, VEC_Phone</td>
<td></td>
</tr>
<tr>
<td>Ins_Subscriber_Info</td>
<td>V Ref Info (VRI_LName, VRI_FName, VRI_Role)</td>
<td></td>
</tr>
<tr>
<td>(Sub_LName, Sub_FName, Sub_Relationship, Sub_Address, Sub_Phone)</td>
<td>V_Employee</td>
<td></td>
</tr>
<tr>
<td>Patient_Type (Outpatient?, Resident?)</td>
<td>V_Employer</td>
<td></td>
</tr>
<tr>
<td></td>
<td>V_Employ_add</td>
<td></td>
</tr>
<tr>
<td></td>
<td>V_Employ_Position</td>
<td></td>
</tr>
<tr>
<td></td>
<td>V_Employ_Start_Date</td>
<td></td>
</tr>
<tr>
<td></td>
<td>V_Employ_End_Date</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MVCH_Service?</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(MVCH_Service_Info)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Volunteer_Exp?</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Volunteer_Exp_Info)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Why_Volunteer?</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Language, Skill, Interest, Volunteer_Preference)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Vol_Availability, (DayOfWeek, PortionOfDay))</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>STAFF</th>
<th>VOL_SERV_HISTORY</th>
<th>WORK_UNIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Job_Class</td>
<td>Service_Begin_Date</td>
<td>Unit_Name</td>
</tr>
<tr>
<td></td>
<td>Service_End_Date</td>
<td>Unit_Location (Facility, Floor)</td>
</tr>
<tr>
<td></td>
<td>Service_Hrs_Worked</td>
<td>Unit_Type</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OUTPATIENT</th>
<th>CARE_CENTER</th>
<th>RN</th>
<th>LPN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date_Admitted</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Date_Discharged</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>VISIT</th>
<th>BED</th>
</tr>
</thead>
<tbody>
<tr>
<td>Visit_No</td>
<td>Bed_ID</td>
</tr>
<tr>
<td>Visit_Date</td>
<td>(Bed_No, Room_No)</td>
</tr>
<tr>
<td>Visit_Time</td>
<td></td>
</tr>
<tr>
<td>Visit_Reason</td>
<td></td>
</tr>
</tbody>
</table>
2. A clear approach is to establish two relationships between RN and CARE_CENTER: Day_in_charge, Night_in_charge. See proposed revision to diagram below.
3. Following are some sample definitions:
   a. **Entity types:**
      - **Physician** - a person who is licensed to practice medicine in this state.
      - **Patient** - a person who has been admitted to Mountain View Community Hospital or who is currently being treated as an outpatient by the hospital.
      - **Employee** - a person who has an employment agreement with the hospital and who is on the hospital payroll.
      - **Care center** - an organizational unit that performs a related set of services directed toward patient care.
   
   b. **Attributes:**
      - **Person_ID** - a unique identifier for each person that does not violate privacy guidelines such as those outlined in HIPAA (e.g., not social security number).
      - **Birth_Date** - month, day, and year a person was born.
      - **Specialty** - a physician’s or nurse’s area of practice.
      - **Location** - the floor number and hospital wing for a work unit in the hospital (e.g., care center or diagnostic unit).
   
   c. **Relationships:**
      - **Provided** - associates a resident patient with a hospital bed.
      - **Scheduled** - associates an outpatient with an instance of a visit to the hospital.

4. **PARTY**: Person and Organization/Facility
   **PARTYROLE**: There are PersonRoles: Patient, Physician, Employee, Volunteer. Within Employee there are additional roles: Nurse, Staff and Technician. Within Patient, there are additional roles: Resident and Outpatient.
   **PARTYRELATIONSHIP**: There are Person-to-Person relationships: Patient to Physician.
   There are also Person to Organization: Nurse to Care_Center; Staff, Technician to Work_Unit.
   **EVENTS**: Communication events include in-person, e-mail and correspondence. Transactions events include orders and tests.
   **PRIORITY_TYPE**: If a patient is acute, then his status will be resident.
   **STATUSTYPE**: When a patient is discharged.
   **EVENTROLE**: Patient, Nurse, Physician.
   **ROLETYPE**: Person or Organization.

5. **A VOLUNTEER** may have one EMERGENCY CONTACT. An EMERGENCY CONTACT may be for more than one VOLUNTEER.
   
   **A VOLUNTEER** must have exactly two REFERENCES. A REFERENCE can act as a reference for more than one VOLUNTEER.
   
   **A VOLUNTEER** must have exactly one last EMPLOYER. An EMPLOYER can employ more than one VOLUNTEER.
   
   **A VOLUNTEER** may have previous volunteer experience.
A VOLUNTEER may have one or more Skills. A VOLUNTEER also may have no Skills.

A VOLUNTEER may have one or more Hobbies. A VOLUNTEER also may have no Hobbies.

A VOLUNTEER may have one or more Interests. A VOLUNTEER also may have no Interests.

A VOLUNTEER may speak one or more Languages (in addition to English).

A VOLUNTEER may have one or more TIMESLOT preferences. A TIMESLOT may be chosen by one or more VOLUNTEERS.

6. The rule can be restated as follows:
   “For a nurse to be appointed nurse-in-charge of a care center, that nurse must possess an RN certificate.”

   ![Diagram](image)

   a. Anchor object: nurse (entity)
   b. Corresponding object: RN Certificate (entity).

7. You can perform a side-by-side comparison to show how the EER diagram provides a more detailed and complete statement of requirements for the hospital. For example, the EER diagram includes information about technicians as well as volunteers.

PROJECT ASSIGNMENTS

P1. A PERSON can be a PATIENT, PHYSICIAN, EMPLOYEE or VOLUNTEER.
An instance of PERSON may be more than one of these.

A PATIENT may be only a RESIDENT PATIENT or an OUTPATIENT and cannot be both.

AN OUTPATIENT is scheduled for one or more VISITS. An OUTPATIENT can also be scheduled for no VISITS. A VISIT is for only one OUTPATIENT.

AN EMPLOYEE may only be a NURSE, STAFF or TECHNICIAN and cannot be more than one of these.

A NURSE may be only a Registered Nurse (RN) or Licensed Practical Nurse (LPN) and cannot be both. An RN will direct the work of one, none, or many LPNs: LPN’s work will be directed by only one RN.

A TECHNICIAN is assigned to one or more DIAGNOSTIC_UNITS. A DIAGNOSTIC_UNIT has one or more TECHNICIANS.

A FACILITY can contain one or more WORK_UNITS or may contain no WORK_UNITS. A WORK_UNIT is part of one and only one FACILITY. Currently defined WORK_UNITS include CARE_CENTERS and DIAGNOSTIC_UNITS.

A CARE_CENTER has at least one, and usually many NURSEs assigned to it. Each CARE_CENTER has one RN assigned as a nurse_in_charge for the day shift, and one RN assigned as a nurse_in_charge for the night shift. Some NURSEs are floaters and are not assigned to a particular CARE_CENTER and will work for more than one CARE_CENTER over time; non-floater NURSEs are assigned to a particular CARE_CENTER.

A CARE_CENTER may contain one or more beds or may contain not BEDS. A BED is contained in only one CARE_CENTER.

A DIAGNOSTIC_UNIT performs one or more TREATMENTS. A TREATMENT is performed by only one DIAGNOSTIC UNIT.

A BED is assigned to one RESIDENT PATIENT or no RESIDENT PATIENTs. A RESIDENT PATIENT is assigned to one BED.

A PHYSICIAN admits one or more PATIENTs or admits no PATIENTs. A PATIENT is admitted by only one PHYSICIAN.

A PHYSICIAN may refer one or more PATIENTs or may refer no PATIENTs. A PATIENT must be referred by one PHYSICIAN.

A PATIENT may consume many ITEMS or may consume no ITEMS. An ITEM is consumed by one or more PATIENTS or may be consumed by no PATIENTS.
An ITEM is supplied by one or more VENDORs. A VENDOR may supply one or more items or may supply no ITEMS.

A PHYSICIAN may write one or more ORDERs or may write no ORDERs for one PATIENT. An ORDER is written by one PHYSICIAN.

An ORDER may consist of one or more ITEMS or no ITEMS. An ITEM may be part of one or more ORDERs or may be part of no ORDERs.

An ORDER may consist of one or more TREATMENTS or no TREATMENTS. A TREATMENT may be part of one or more ORDERs, or no ORDERs.

A PHYSICIAN may complete one or more DIAGNOSES for one or more PATIENTs. A DIAGNOSIS is Completed for one PATIENT by one PHYSICIAN.
P2. [Note to instructor:] When assigning this exercise to students, be sure to allow a sufficient amount of time for completion. The case scenario is fairly complex and will encourage students to do a lot of thinking and experimenting prior to developing a workable diagram. Due to the large size of the diagram, and the necessary notes to the diagram, this solution is presented in three parts: notes, diagram (without attributes displayed), and a “diagram” of the entities with all attributes displayed.]

Business Rules/Notes for EERD (diagram on next page):
1. Only one entry for the Patient’s Emergency Contact Information is stored in the database.
2. Only the primary insurance information for the Patient is stored in the database. Secondary insurance information (if provided by the Patient) will be stored in paper files.
3. Referring/Primary Care Physician Contact information is stored as part of the PHYSICIAN entity in the database.
4. MVCH wants to track the history of all volunteer assignments within the facility, thus there is a need to use an associative entity (VOL_SERV_HISTORY) to track over time the various volunteer assignments, and each assignment’s supervisor, as well as the total amount of hours worked on each assignment. Some WORK_UNITs do not have VOLUNTEERs, thus a 0:M cardinality is required on its relationship with the associative entity.
5. A VOLUNTEER may be supervised by an EMPLOYEE or a PHYSICIAN at one time. Only one of these “supervision” relationships is active at one point in time, although both are shown on the diagram.
6. MVCH wants to track VOLUNTEER information at the point of application, thus it is possible that a VOLUNTEER instance may not yet have a corresponding instance of VOL_SERV_HISTORY in the database system (this is why there is a 0:M cardinality nearest the VOL_SERV_HISTORY associative entity on the relationship from VOLUNTEER).
7. A Registered Nurse (RN) may direct one, none, or several Licensed Practical Nurses (LPNs); a single LPN will be directed by only one RN. MVCH wishes to track these RN direction responsibilities for accountability and quality control purposes within the hospital.
8. A Floater nurse is not assigned to a CARE_CENTER, thus this is why there is a 0:M cardinality shown near CARE_CENTER on the Assigned relationship. If discussion with end users indicate additional attributes or relationships that are associated with Floater nurses, then an alternative solution could be to establish a FLOATER subtype entity on the diagram. The addition of this FLOATER subtype could allow the cardinality of the Assigned relationship for the NURSE supertype to be changed to 1:M.
9. Skill is modeled as a multivalued attribute of TECHNICIAN as it only relates to this entity and has no additional characteristics mentioned in the case. Under different assumptions, an alternative solution could be to model Skill as its own entity type with a relationship to TECHNICIAN (and possibly other entity types in the model).
10. Only current STAFF and TECHNICIAN assignments to WORK_CENTERs are necessary for this case.
11. Only current BED to RESIDENT associations are necessary in the database.
12. A NURSE prepares one, none, or many ASSESSMENTs of PATIENTs; A PATIENT receives one to many ASSESSMENTs over time.
13. Assessment_ID is unique (surrogate) identifier for ASSESSMENT.
14. A DIAGNOSTIC_UNIT is defined to include Labs and other hospital WORK_UNITs that perform procedures, tests, and treatments required by ORDERS. A DIAGNOSTIC_UNIT performs one to many TREATMENTS; A TREATMENT is performed by one DIAGNOSTIC_UNIT.
### PERSON
- Person_ID
- Person_Name
- Person_Str_Address
- Person_City
- Person_State
- Person_Zip
- Person_Home_Phone
- Person_Work_Phone
- PersonDOB
- Person_Email
- Person_Type (Physician?, Employee?, Volunteer?, Patient?)

### PHYSICIAN
- DEA_No
- Pager_No
- Phys_Specialty

### PHYSICIAN_DX
- Assessment_ID
- Assessment_Date
- Assessment_Time
- Comments
- Patient_Weight
- Patient_BP
- Patient_pulse
- Patient_temperature

### STAFF
- Job_Class

### NURSE
- Cert_Degree
- State_License_No
- Nurse_Specialty (Field_Certification)
- Nurs_Type

### TECHNICIAN
- Tech_Skill

### ORDER
- Order_ID
- Order_Date
- Order_Time

### VENDOR
- Vendor_ID
- Vendor_Name

### CARE_CENTER
- Facility_ID
- Facility_Name

### FACILITY
- Facility_ID
- Facility_Name

### DIAGNOSIS
- Diagnosis_Code
- Diagnosis_Name

### DIAGNOSTIC_UNIT
- Treatment_Code
- Treatment_Name

### ITEM
- Item_No
- Item_Desc
- Item_Unit_Cost

### ITEM_BILLING
- Start_Date
- End_Date
- Cost

### ITEM_CONSUMPTION
- Consume_Date
- Consume_Time
- Consume_Qty
  
### SCHEDULE
- Schedule_ID
- Sched_Begin
- Sched_End

### VOL_SERV_HISTORY
- Service_Begin_Date
- Service_End_Date
- Service_Hrs_Worked

### VOLUNTEER
- First_Name
- Last_Name
- Middle_Name
- Vol_svc_Full_Name
- Vol_svc_Last_Name
- Vol_svc_First_Name
- Vol_svc_Middle_Name
- Vol_svc_Ext_Name
- Vol_svc_Suffix
- Vol_svc_Rank
- Vol_svc_Rating
- Vol_svc_Rating_Remarks
- Vol_svc_Rating_Remarks
- Vol_svc_Rating_Remarks
- Vol_svc_Rating_Remarks
- Vol_svc_Rating_Remarks
- Vol_svc_Rating_Remarks
- Vol_svc_Rating_Remarks
- Vol_svc_Rating_Remarks
- Vol_svc_Rating_Remarks
- Vol_svc_Rating_Remarks
- Vol_svc_Rating_Remarks
- Vol_svc_Rating_Remarks
- Vol_svc_Rating_Remarks
- Vol_svc_Rating_Remarks
- Vol_svc_Rating_Remarks
- Vol_svc_Rating_Remarks
- Vol_svc_Rating_Remarks
- Vol_svc_Rating_Remarks
- Vol_svc_Rating_Remarks
- Vol_svc_Rating_Remarks
- Vol_svc_Rating_Remarks
- Vol_svc_Rating_Remarks
- Vol_svc_Rating_Remarks
- Vol_svc_Rating_Remarks
- Vol_svc_Rating_Remarks
- Vol_svc_Rating_Remarks
- Vol_svc_Rating_Remarks
- Vol_svc_Rating_Remarks
- Vol_svc_Rating_Remarks
- Vol_svc_Rating_Remarks
- Vol_svc_Rating_Remarks
- Vol_svc_Rating_Remarks
- Vol_svc_Rating_Remarks
- Vol_svc_Rating_Remarks
- Vol_svc_Rating_Remarks
- Vol_svc_Rating_Remarks
- Vol_svc_Rating_Remarks
- Vol_svc_Rating_Remarks
- Vol_svc_Rating_Remarks
- Vol_svc_Rating_Remarks
- Vol_svc_Rating_Remarks
- Vol_svc_Rating_Remarks
- Vol_svc_Rating_Remarks
- Vol_svc_Rating_Remarks
- Vol_svc_Rating_Remarks
- Vol_svc_Rating_Remarks
- Vol_svc_Rating_Remarks
- Vol_svc_Rating_Remarks
- Vol_svc_Rating_Remarks
- Vol_svc_Rating_Remarks
- Vol svc Ratin
P3. **[Note to instructors]**: Student answers may vary slightly; these issues are presented as a possible representation of issues which may be identified. An in-class comparison/contrast of selected student answers to this assignment and that from the prior chapter would be a good hands-on exercise for exploring the upcoming chapter 5 topic of merging issues (homonyms, synonyms, transitive dependencies, and supertype/subtype relationships). Comparing/contrasting the diagrams could lead to a useful discussion of the importance of **understanding the meaning of the data** in order to resolve issues and ensure appropriate capture of the end user’s data requirements.[

Some issues that come up during the merging:
- Should there be a relationship between DIAGNOSTIC_UNIT and TECHNICIAN?
- Are items consumed by both resident patients as well as outpatients? (diagram shows current assumption is ‘Yes’)
- What needs to be done to the data model to allow for follow up care for discharged patients.
- We kept the Admits relationship between PATIENT and PHYSICIAN in place of the Responsible used in the original EER model (Case Exercise 1, Chapter 4); is this a correct understanding of the relationship, or is another relationship required to show the Responsible association?
- We assumed that the NURSE subtype (both RN and LPN) participates in the ASSESSMENT of PATIENTs and reflects the same relationship as shown by the EMPLOYEE relationship to ASSESSMENT in Chapter 3. Do other EMPLOYEE subtypes (e.g., STAFF or TECHNICIAN) prepare such ASSESSMENTs? Do both types of NURSES prepare ASSESSMENTs?
- We assumed that a DIAGNOSTIC_UNIT is defined to include Labs and other hospital WORK_UNITs that perform procedures, tests, and treatments required by ORDERs. Is this a correct assumption or are there other WORK_UNITs that need to be tracked in the database?
- In chapter 3, a business rule was indicated that EMPLOYEES are Assigned to CARE_CENTERS and that Hours_Worked needed to be tracked for each such assignment in the database. We interpreted this to mean the NURSE subtype in the chapter 4 EER diagram and have added the Hours_Worked attribute to the M:N Assigned relationship in the diagram.
- In chapter 3, the identifier for the BED weak entity was described as a composite identifier including Bed_No, Care_Center_ID, and Room_No. In chapter 4, the identifier for the BED weak entity was described as a composite identifier including Bed_No and Room_No. The correct identifier needs to be validated with the end users and finalized in the EER diagram listing of entities.
Chapter 5  Logical Database Design and the Relational Model

Chapter Overview

The purpose of this chapter is to describe in depth the major steps in logical database design, with more emphasis on the relational model. Logical database design is the process of transforming the conceptual data model (described in Chapters 3 and 4) into a logical data model. First, we provide a concise description of the relational data model, including the properties of relations. Next, we describe and illustrate the various types of integrity constraints associated with the relational model. This section introduces SQL table definitions and the concept of well-structured relations. We then provide a detailed description of the process of transforming EER diagrams into relations. Next, we define normalization and describe the steps in normalizing relations. The chapter concludes with a discussion of merging relations and techniques for dealing with typical issues that arise during this process.

Chapter Objectives

Specific student learning objectives are included in the beginning of the chapter. From an instructor’s point of view, the objectives of this chapter are to:

1. Show students the position of logical database design within the overall database development process. This is a key chapter in the textbook since students will begin to see how their databases will be implemented.
2. Provide students with a solid understanding of the relational data model including the properties of relations, integrity constraints, and well-structured relations.
3. Discuss the principles and detailed steps involved in mapping EER diagrams to relations. Computer-assisted techniques are often used to speed up this process, but students should still understand the principles involved.
4. Provide students with a firm grasp on the principles of functional dependencies, determinants, and related concepts of normalization.
5. Emphasize why normalization is important to stable database design with the relational model, and then present a concise description of the various normal forms and the normalization process.
6. Discuss some of the anomalies that arise when merging relations, and discuss how to apply the principles we have learned to address these anomalies.
Key Terms

<table>
<thead>
<tr>
<th>Alias</th>
<th>Foreign key</th>
<th>Recursive foreign key</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anomaly</td>
<td>Functional dependency</td>
<td>Referential integrity constraint</td>
</tr>
<tr>
<td>Candidate key</td>
<td>Homonym</td>
<td>Relation</td>
</tr>
<tr>
<td>Composite key</td>
<td>Normal form</td>
<td>Second normal form</td>
</tr>
<tr>
<td>Determinant</td>
<td>Normalization</td>
<td>Surrogate primary key</td>
</tr>
<tr>
<td>Enterprise key</td>
<td>Null</td>
<td>Synonyms</td>
</tr>
<tr>
<td>Entity integrity rule</td>
<td>Partial functional dependency</td>
<td>Third normal form</td>
</tr>
<tr>
<td>First normal form</td>
<td>Primary key</td>
<td>Well-structured relation</td>
</tr>
</tbody>
</table>

Classroom Ideas

1. Motivate the need for logical database design. We sometimes start by showing students the conceptual data model (E-R diagrams) for Pine Valley Furniture Company (Figure 3-22). Emphasize that this E-R diagram must be transformed through logical database design before it can be implemented.

2. Review the position of logical database design in the overall database development process (see Figure 2-5). You might want to discuss who in the organization is usually responsible for this step and what CASE tools might be appropriate.

3. Discuss the relational data model using Figures 5-1 through 5-4 as examples.

4. Introduce the important integrity constraints in the relational model using Figure 5-5 and Table 5-1. Emphasize that these constraints will be enforced by the DBMS, but must first be specified by the designer.

5. Introduce SQL table definitions (Figure 5-6). Show how these definitions specify the referential integrity constraints that are diagrammed in Figure 5-5.

6. Illustrate how anomalies can occur when relations are not well structured, using Figures 5-2b and 5-7. Emphasize the fact that much real-world data (including relational data) are not well structured.

7. Discuss the process of transforming EER diagrams to relations (Figures 5-8 through 5-21). We suggest you reinforce these concepts by asking your students (in teams of two) to perform Exercise 6a in class immediately following the discussion.

8. Preview the steps in normalization using Figure 5-22. You will want to use this figure again to summarize normalization at the end of your discussion.

9. Discuss the concepts of functional dependencies, determinants, and candidate keys. Start with your own examples on the board, and then have your students give additional examples. Summarize using Figure 5-22.

10. Discuss first through third normal forms, using Figures 5-25 through 5-29. Additional normal forms (BCNF and 4NF) are presented in Appendix B, if time permits.

11. Discuss merging relations and view integration problems.

12. Use Figure 5-31 to discuss enterprise keys and what they are used for.

13. We strongly suggest for you to ask your students to work in small teams on one or more end-of-chapter exercises. (Exercises 3 and 4 work well for this purpose.)
Answers to Review Questions

1. Define each of the following terms:
   a. **Determinant** The attribute on the left-hand side of the arrow in a functional dependency.
   b. **Functional dependency** A constraint between two attributes or two sets of attributes.
   c. **Transitive dependency** A functional dependency between two (or more) nonkey attributes.
   d. **Recursive foreign key** A foreign key in a relation that references the primary key values of that same relation.
   e. **Normalization** The process of decomposing relations with anomalies to produce smaller, well-structured relations.
   f. **Composite key** A primary key that consists of more than one attribute.
   g. **Relation** A named, two-dimensional table of data.
   h. **Normal form** A state of a relation that results from applying simple rules regarding functional dependencies (or relationships between attributes) to that relation.
   i. **Partial functional dependency** A functional dependency in which one or more nonkey attributes (such as Name) are functionally dependent on part (but not all) of the primary key.
   j. **Enterprise key** A primary key whose value is unique across all relations.
   k. **Surrogate primary key** A serial number or other system assigned primary key for a relation.

2. f well-structured relation
e anomaly
a functional dependency
j determinant
g composite key
d 1NF
h 2NF
i 3NF
c recursive foreign key
k relation
b transitive dependency

3. Contrast the following terms:
   a. **Normal form; normalization** Normal form is a state of a particular relation regarding functional dependencies, while normalization is the process of decomposing relations with anomalies to produce smaller, well-structured relations.
   b. **Candidate key; primary key** A primary key is an attribute (or combination of attributes) that uniquely identifies a row in a relation. When a relation has more than one such attribute (or combination of attributes), each is called a candidate key. The primary key is then the one chosen by users to uniquely identify the rows in the relation.
c. **Partial dependency; transitive dependency** A partial functional dependency exists when a nonkey attribute is functionally dependent on part (but not all) of a composite primary key; a transitive dependency is a functional dependency between two or more nonkey attributes.

d. **Composite key; recursive foreign key** A composite key is a primary key that consists of more than one attribute, while a recursive foreign key is a foreign key in a relation that references the primary key values of that same relation.

e. **Determinant; candidate key** A determinant is the attribute on the left-hand side of the arrow in a functional dependency, while a candidate key uniquely identifies a row in a relation.

f. **Foreign key; primary key** A primary key uniquely identifies each row in a relation, while a foreign key is a primary key in another table.

g. **Enterprise key; surrogate primary key** An enterprise key is a primary key whose value is unique across all relations in the whole database and is likely to hold no business meaning. A surrogate primary key is a primary key whose value is a serial number or other system assigned value and is unique to the relation.

4. Six important properties of relations are:
   a. Each relation in a database has a unique name.
   b. An entry at the intersection of each row and column is atomic (or single valued).
   c. Each row is unique.
   d. Each attribute within a table has a unique name.
   e. The sequence of columns is insignificant.
   f. The sequence of rows is insignificant.

5. Describe two properties that must be satisfied by candidate keys:
   a. Unique identification: For every row, the value of the key must uniquely identify that row.
   b. Nonredundancy: No attribute in the key can be deleted without destroying the property of unique identification.

6. Three types of anomalies in tables:
   a. Insertion anomaly: A new row cannot be inserted unless all primary key values are supplied.
   b. Deletion anomaly: Deleting a row results in the loss of important information not stored elsewhere.
   c. Modification anomaly: A simple update must be applied to multiple rows.
7. Fill in the blanks.
   a. second
   b. first
   c. third

8. A well-structured relation is one that contains a minimum amount of redundancy and allows users to insert, modify, and delete the rows in a table without errors or inconsistency. Well-structured relations are important because they promote database integrity.

9. Describe how the following components of an E-R diagram are transformed into relations:
   a. Regular entity type: Each entity type is transformed into a simple relation. Each simple attribute of the entity type becomes an attribute of the relation.
   b. Relationship (1:M): A relation is created for each of the two entity types participating in the relationship. The primary key attribute of the entity on the one-side of the relationship becomes a foreign key in the relation on the many-side of the relationship.
   c. Relationship (M:N): A new relation is created to represent this relationship. The primary key for each of the participating entity types is included in this new relation.
   d. Relationship (supertype/subtype): A separate relation is created for the supertype and each of its subtypes. The primary key of the supertype is assigned to each subtype, as well as attributes that are unique to the subtype.
   e. Multivalued attribute: A new relation is created to replace the multivalued attribute. The primary key of this new relation consists of two attributes: the primary key of the original relation, plus the multivalued attribute itself.
   f. Weak entity: A new relation is created corresponding to the weak entity. The primary key of this relation consists of the primary key of the owner relation, plus the partial identifier of the weak entity type.
   g. Composite attribute: The simple component attributes of the composite attribute are included in the new relation.

10. Four typical problems in merging relations:
    a. Synonyms: Two (or more) attributes have different names but the same meaning.
        Solution: Convince users to standardize on a single name.
    b. Homonyms: A single attribute has more than one meaning.
        Solution: Create new attribute names that capture the separate meanings.
    c. Transitive dependency: Merging relations produces transitive dependencies.
        Solution: Create 3 NF relations by removing the transitive dependency.
    d. Supertype/subtype: May be implied by content of existing relations.
        Solution: Create new relations that explicitly recognize this relationship.

11. Three conditions that imply a relation is in second normal form:
    a. The primary key consists of a simple attribute
    b. No nonkey attributes exist in the relation
c. Every nonkey attribute is functionally dependent on the full set of primary key attributes

12. Integrity constraints enforced in SQL CREATE TABLE commands:
   a. Entity integrity: enforced by NOT NULL clause
   b. Referential integrity: enforced by FOREIGN KEY REFERENCES statement

13. Relationships between entities are represented by foreign key values in one relation that match primary key values in another relation.

14. A 1:M unary relationship is represented by a recursive foreign key whose values reference the primary key values of the same relation.

15. An M:N ternary relationship is represented by a new associative relation whose primary key consists of the primary key attributes of the participating entity types.

16. A new relation (called the associative relation) is created to represent the associative entity. The default primary key consists of the primary key values from the relations that participate in the association.

17. All of the nonkey attributes of a relation are functionally dependent on the primary key of that relation.

18. A foreign key must not be null when the minimum cardinality of the relationship implied by the foreign key is one.

19. The only way to eliminate ripple effects is to create an enterprise key from the very beginning of database development.

20. A unary 1:M relationship always utilizes a recursive foreign key, whereas an M:N recursive relationship dictates that a second table must be created for the relationship.

21. The following three conditions are adequate to suggest that a surrogate key should be developed for a relation:
   - the presence of a composite primary key
   - the natural primary key (the one initially identified by users during conceptual modeling activities) is too long or made up of more than two attributes which will cause performance issues with database software
   - the natural primary key cannot be guaranteed to be unique over time (e.g., due to duplicates or re-use over time)

Answers to Problems and Exercises

Note to Instructors: In this section of the Instructor’s Manual, we have adopted a number of notational conventions to indicate the Primary and Foreign Keys in the relations. Primary Key(s) are noted with an underlined name. Foreign Key(s) are noted in Italics. Please be aware that due
to the shortcomings of the drawing tool used to represent the relations, the underscore used to connect multi-part names of an attribute is sometimes hard to see when the attribute name is underlined to depict a Primary Key (e.g., Employee_ID in Problem and Exercise 1 below). Instructors may wish to: (1) modify attribute names when relations are created (e.g., Employee_ID becomes EmployeeID); (2) modify notation for Primary Keys (e.g., use **Bold** rather than *Underline* to depict Primary Keys when using a software tool); or (3) acknowledge the shortcomings of the software tools and accept the graphical representation as valid.

1. Transforming E-R diagrams to relations:
   a. The derived attribute Years_Employed is not going to be an actual column, so it is not displayed here.

![Diagram](image)

   EMPLOYEE
   - Employee_ID
   - Employee_Name
   - Address
   - Date_Employed

   EMPLOYEE SKILL
   - Employee_ID
   - Skill

   FLIGHT
   - Flight_No
   - Date
   - No_of_Passengers
c.
d.

```
EMPLOYEE

Employee_ID  Employee_Name  DOB

CERTIFICATE

Certificate_No  Employee_ID  Course_ID  Date_Completed

COURSE

COURSE_ID  Course_Title

COURSE_TOPIC

COURSE_ID  Topic

TOPIC

TOPIC
```
e. COURSE

   Course_ID    Course_Title

   PREREQUISITE

   Course_ID    Prereq_ID

f. EMPLOYEE

   Employee_ID    Employee_Name

   EMPLOYEE_SKILL

   Employee_ID    Skill_ID

   SKILL

   Skill_ID    Skill_Title    Skill_Type
g. MOVIE

   Movie_Name

   VIDEO TAPE

   Copy_No  Movie_Name

h. PRODUCT

   Product_ID

   HISTORY

   Product_ID  Effective_Date  Price
2. Transforming EER diagrams to relations:
   a.

   VEHICLE
<table>
<thead>
<tr>
<th>Vehicle_ID</th>
<th>Price</th>
<th>Make</th>
<th>Model</th>
<th>Engine_Displacement</th>
</tr>
</thead>
</table>

   CAR
<table>
<thead>
<tr>
<th>C_Vehicle_ID</th>
<th>No_of_Passengers</th>
</tr>
</thead>
</table>

   TRUCK
<table>
<thead>
<tr>
<th>T_Vehicle_ID</th>
<th>Cab_Type</th>
<th>Capacity</th>
</tr>
</thead>
</table>
b.

**RESPONSIBLE_PHYSICIAN**

Physician_ID

**PATIENT**

Patient_ID  Patient_Name  Admit_Date  Physician_ID

**OUTPATIENT**

O_Patient_ID  Checkback_Date

**RESIDENT_PATIENT**

R_Patient_ID  Date_Discharged  Bed_ID

**BED**

Bed_ID
c.

PART

| Part_No | Description | Location | Quantity_on_Hand | Manufactured? | Purchased? |

MANUFACTURED PART

| M_Part_No | Routing_Number |

PURCHASED_PART

| P_Part_No |

SUPPLIES

| P_Part_No | Supplier_ID | Unit_Price |

SUPPLIER

| Supplier_ID |
d.

(Solution to exercise 2e is shown on the next page)
3. The normal form for the relations are:
   a. 3NF
   b. 3NF
   c. 2NF
      CLASS (Course_No, Section_No, Room)
      ROOM (Room, Capacity)
   d. 1NF
      COURSE (Course_No, Course_Name)
      CLASS (Course_No, Section_No, Room)
      ROOM (Room, Capacity)
4. 3NF relations for Millennium College are:
   OBJECT (OID, Object_Type)
   INSTRUCTOR (OID, Instructor_Name, Instructor_Location)
   COURSE (OID, Course_No, Course_Title, Instructor_Name)
   STUDENT (OID, Student_No, Student_Name, Major)
   OUTCOME (OID, Student_No, Course_No, Grade)
5. Transforming an E-R diagram to relations (parts a and b)
6. Transforming Table 5-2 to relations:

a. PART SUPPLIER

<table>
<thead>
<tr>
<th>Part_No</th>
<th>Description</th>
<th>Vendor_Name</th>
<th>Address</th>
<th>Unit_Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>1234</td>
<td>Logic Chip</td>
<td>Fast Chips</td>
<td>Cupertino</td>
<td>10.00</td>
</tr>
<tr>
<td>1234</td>
<td>Logic Chip</td>
<td>Smart Chips</td>
<td>Phoenix</td>
<td>8.00</td>
</tr>
<tr>
<td>5678</td>
<td>Memory Chip</td>
<td>Fast Chips</td>
<td>Cupertino</td>
<td>3.00</td>
</tr>
<tr>
<td>5678</td>
<td>Memory Chip</td>
<td>Quality Chips</td>
<td>Austin</td>
<td>2.00</td>
</tr>
<tr>
<td>5678</td>
<td>Memory Chip</td>
<td>Smart Chips</td>
<td>Phoenix</td>
<td>5.00</td>
</tr>
</tbody>
</table>
b.  

<table>
<thead>
<tr>
<th>Part_No</th>
<th>Description</th>
<th>Vendor_Name</th>
<th>Address</th>
<th>Unit_Cost</th>
</tr>
</thead>
</table>

**Part Supplier**

- **Part_No**: Primary key for the PART table
- **Description**: Part description
- **Vendor_Name**: Name of the vendor
- **Address**: Address of the vendor
- **Unit_Cost**: Unit cost of the part

**Part**

- **Part_No**: Primary key for the PART table
- **Description**: Part description

**Part Supplier**

- **Part_No**: Foreign key to PART table
- **Vendor_Name**: Foreign key to VENDOR table
- **Unit_Cost**: Unit cost of the part

**VENDOR**

- **Vendor_Name**: Foreign key to PART Supplier table
- **Address**: Address of the vendor

c. Insert anomaly: We cannot insert a new vendor unless we also include a part number that the supplier provides.
Delete anomaly: If we delete the last part information for a given supplier, we also lose information about that vendor.
Modification anomaly: If a vendor’s address changes, we have to modify all records (or rows) for that vendor.

d.  

![Diagram](attachment:part_supplier.png)

e. 1NF

f. PART

![Diagram](attachment:part_vendor.png)
7. Transforming Table 5-3 to relations:
   a.

   b. 1NF
c.

<table>
<thead>
<tr>
<th>STUDENT</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Student_ID</td>
<td>Student_Name</td>
<td>Campus_Address</td>
<td>Major</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>REGISTRATION</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Student_ID</td>
<td>Course_ID</td>
<td>Grade</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>COURSE</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Course_ID</td>
<td>Course_Title</td>
<td>Instructor_Name</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>INSTRUCTOR</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Instructor_Name</td>
<td>Instructor_Location</td>
</tr>
</tbody>
</table>
d.

[Diagram of database schema]

- **STUDENT**
  - Student_ID
  - Student_Name
  - Campus_Address
  - Major

- **REGISTRATION**
  - Student_ID
  - Course_ID
  - Grade

- **COURSE**
  - Course_ID
  - Course_Title
  - Instructor_Name

- **INSTRUCTOR**
  - Instructor_Name
  - Instructor_Location
8.

a. Total weight is a derived attribute and can be calculated from weight and quantity. Therefore it is not shown here as part of the relational schema.

b. First normal form
c.

**Shipment**

```
| Shipment ID | Shipment Date | Expected Arrival Date | Origin | Destination | Ship Number | Captain ID |
```

**Shipment Line**

```
| Shipment ID | Item Number | Quantity |
```

**Item**

```
| Item Number | Type | Description | Weight |
```

**Captain**

```
| Captain ID | Captain Name |
```

Published by www.emagazine.org
d.

Shipments

Item

Captain
e.
9.

**CAPTAIN**
- Capt_ID
- Captain_Name

**SHIPMENT**
- Shipment_ID
- Shipment_Date
- Expected_Arrival_Date
- Origin
- Destination
- Ship_No

**SHIPMENT_LINE**
- Quantity

**ITEM**
- Item_Number
- Description
- Type
- Weight

The diagram shows the relationships between the entities and attributes:
- **Capt_ID** is the primary key for the **CAPTAIN** entity.
- **Shipment_ID** is the primary key for the **SHIPMENT** entity.
- **Item_Number** is a unique identifier for each item in the **ITEM** entity.
- The **SHIPMENT** entity operates on the **CAPTAIN** entity through the **Operated_By** relationship.
- Each **SHIPMENT** has many **SHIPMENT_LINE** entities.
- Each **SHIPMENT_LINE** has an **ITEM**.

Published by [www.emagzine.org](http://www.emagzine.org)
10.

Attribute version:

**EMPLOYEE**

<table>
<thead>
<tr>
<th>Employee_ID</th>
<th>Employee_Name</th>
</tr>
</thead>
</table>

**SKILL**

<table>
<thead>
<tr>
<th>Employee_ID</th>
<th>Skill_Code</th>
<th>Skill_Title</th>
<th>Skill_Type</th>
</tr>
</thead>
</table>

Relationship version:

**EMPLOYEE**

<table>
<thead>
<tr>
<th>Employee_ID</th>
<th>Employee_Name</th>
</tr>
</thead>
</table>

**POSSESES**

<table>
<thead>
<tr>
<th>Employee_ID</th>
<th>Skill_Code</th>
</tr>
</thead>
</table>

**SKILL**

<table>
<thead>
<tr>
<th>Skill_Code</th>
<th>Skill_Title</th>
<th>Skill_Type</th>
</tr>
</thead>
</table>

The attribute version of the 3NF relations is similar to Figure 5-10. However, we have a much clearer definition of a primary key in this version. One main advantage of the relationship version is that we do not have to store Skill_Title and Skill_Type many times. If a skill title was changed or types were reclassified, this would make things much easier since update anomalies are eliminated.
11.

a. Parking ticket table

<table>
<thead>
<tr>
<th>St_ID</th>
<th>L_Name</th>
<th>F_Name</th>
<th>Phone_No</th>
<th>St_Lic</th>
<th>Lic_No</th>
<th>Ticket#</th>
<th>Date</th>
<th>Code</th>
<th>Fine</th>
</tr>
</thead>
<tbody>
<tr>
<td>38249</td>
<td>Brown</td>
<td>Thomas</td>
<td>111-7804</td>
<td>FL</td>
<td>BRY</td>
<td>15634</td>
<td>10/17/08</td>
<td>2</td>
<td>$25</td>
</tr>
<tr>
<td>38249</td>
<td>Brown</td>
<td>Thomas</td>
<td>111-7804</td>
<td>FL</td>
<td>BRY</td>
<td>16017</td>
<td>11/13/08</td>
<td>1</td>
<td>$15</td>
</tr>
<tr>
<td>82453</td>
<td>Green</td>
<td>Sally</td>
<td>391-1689</td>
<td>AL</td>
<td>TRE</td>
<td>14987</td>
<td>10/05/08</td>
<td>3</td>
<td>$100</td>
</tr>
<tr>
<td>82453</td>
<td>Green</td>
<td>Sally</td>
<td>391-1689</td>
<td>AL</td>
<td>TRE</td>
<td>16293</td>
<td>11/18/08</td>
<td>1</td>
<td>$15</td>
</tr>
<tr>
<td>82453</td>
<td>Green</td>
<td>Sally</td>
<td>391-1689</td>
<td>AL</td>
<td>TRE</td>
<td>17892</td>
<td>12/13/08</td>
<td>2</td>
<td>$25</td>
</tr>
</tbody>
</table>

The determinants are: ST_ID, Ticket#, Code.

b.

c. An insertion anomaly occurs when you try to insert a student who has never had a parking ticket.

A deletion anomaly would occur if you were to delete a record of any student with only one parking ticket since all student information will be lost.

A modification anomaly would occur whenever you want to update the Phone_No (or name, State License, or License Number) data of a student.
d.

**STUDENT**

- **ST_ID**
- **L_Name**
- **F_Name**
- **Phone_No**
- **ST_Lic**
- **Lic_No**

**TICKET**

- **Ticket#**
- **Date**
- **Code**
- **ST_ID**

**TICKET_CODE**

- **Code**
- **Fine**
- **Violation**
e. 

```
STUDENT
- ST_ID
- L_Name
- F_Name
- Phone_No
- Lic_No
- St_Lic
```

```
TICKET
- Ticket_No
- Date
```

```
TICKET_CODE
- Code
- Violation
- Fine
```

f. 

```
STUDENT
<table>
<thead>
<tr>
<th>PK</th>
<th>ST_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>L_Name</td>
<td></td>
</tr>
<tr>
<td>F_Name</td>
<td></td>
</tr>
<tr>
<td>Phone_No</td>
<td></td>
</tr>
<tr>
<td>Lic_No</td>
<td></td>
</tr>
<tr>
<td>St_Lic</td>
<td></td>
</tr>
</tbody>
</table>
```

```
TICKET
<table>
<thead>
<tr>
<th>PK</th>
<th>Ticket#</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>FK1</td>
<td>ST_ID</td>
</tr>
<tr>
<td></td>
<td>Date</td>
</tr>
<tr>
<td>FK2</td>
<td>Code</td>
</tr>
</tbody>
</table>
```

```
TICKET_CODE
<table>
<thead>
<tr>
<th>PK</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Violation</td>
<td></td>
</tr>
<tr>
<td>Fine</td>
<td></td>
</tr>
</tbody>
</table>
```

Published by www.emagzine.org
12.

a.

b. MATERIAL

- Material_ID
- Material_Name
- Unit_Of_Measure
- Standard_Cost
- Vendor_ID
- Vendor_Name
- Unit_Price
- Terms_Code
- Terms

- SuppliedBy
  - Material_ID
  - Vendor_ID
  - Unit_Price

- VENDOR
  - Vendor_ID
  - Vendor_Name
  - Terms_Code

- TERM
  - Terms_Code
  - Terms
c.

```
MATERIAL
Material_ID
Material_Name
Unit_of_Measure
Standard_Cost

SuppliedBy
Unit_Price

VENDOR
Vendor_ID
Vendor_Name

TERM
Terms_Code
Terms

Uses
```

d.

```
MATERIAL
PK Material_ID
Material_Name
Unit_of_Measure
Standard_Cost

SuppliedBy
PK,FK1 Material_ID
PK,FK2 Vendor_ID
Unit_Price

TERM
PK Term_Code
Terms

VENDOR
PK Vendor_ID
Name
FK1 Term_Code

Uses
```
13. 

a. 

b. Second normal form. There is a transitive dependency between origin/destination and distance.

[Note to instructor: An interesting discussion point for you to make the students aware of is the issue of symmetry in this problem. By symmetry, we mean: is the distance from A to B the same as from B to A? If it is, then there is an interesting issue about whether the Distances table potentially has redundancy. You may wish to consider assigning an advanced exercise to remove the potential redundancy from the relations under these circumstances.]
c.

**SHIPMENT**

<table>
<thead>
<tr>
<th>Shipment#</th>
<th>Origin</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>409</td>
<td>Seattle</td>
<td>Denver</td>
</tr>
<tr>
<td>618</td>
<td>Chicago</td>
<td>Dallas</td>
</tr>
<tr>
<td>723</td>
<td>Boston</td>
<td>Atlanta</td>
</tr>
<tr>
<td>824</td>
<td>Denver</td>
<td>Los Angeles</td>
</tr>
<tr>
<td>629</td>
<td>Seattle</td>
<td>Denver</td>
</tr>
</tbody>
</table>

**Distances**

<table>
<thead>
<tr>
<th>Origin</th>
<th>Destination</th>
<th>Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seattle</td>
<td>Denver</td>
<td>1537</td>
</tr>
<tr>
<td>Chicago</td>
<td>Dallas</td>
<td>1058</td>
</tr>
<tr>
<td>Boston</td>
<td>Atlanta</td>
<td>1214</td>
</tr>
<tr>
<td>Denver</td>
<td>Los Angeles</td>
<td>975</td>
</tr>
</tbody>
</table>

d.

**SHIPMENT**

<table>
<thead>
<tr>
<th>PK</th>
<th>Shipment#</th>
</tr>
</thead>
<tbody>
<tr>
<td>FK1</td>
<td>Origin</td>
</tr>
<tr>
<td>FK1</td>
<td>Destination</td>
</tr>
</tbody>
</table>

**DISTANCES**

<table>
<thead>
<tr>
<th>PK</th>
<th>Origin</th>
</tr>
</thead>
<tbody>
<tr>
<td>PK</td>
<td>Destination</td>
</tr>
<tr>
<td></td>
<td>Distance</td>
</tr>
</tbody>
</table>
14.

a.

RENTER

<table>
<thead>
<tr>
<th>Renter_ID</th>
<th>Last_Name</th>
<th>First_Name</th>
<th>Middle_Initial</th>
<th>Address</th>
<th>Phone#</th>
<th>E_Mail</th>
</tr>
</thead>
</table>

RENTAL_AGREEMENT

<table>
<thead>
<tr>
<th>Agreement_ID</th>
<th>Begin_Date</th>
<th>End_Date</th>
<th>Rental_Amount</th>
<th>Renter_ID</th>
<th>Property_ID</th>
</tr>
</thead>
</table>

PROPERTY

<table>
<thead>
<tr>
<th>Property_ID</th>
<th>Street_Address</th>
<th>City_State</th>
<th>Zip</th>
<th>No_Rooms</th>
<th>Base_Rate</th>
<th>Property_Type</th>
</tr>
</thead>
</table>

BEACHPROPERTY

<table>
<thead>
<tr>
<th>B_Property_ID</th>
<th>Blocks_to_Beach</th>
</tr>
</thead>
</table>

MOUNTAINPROPERTY

<table>
<thead>
<tr>
<th>M_Property_ID</th>
<th>Property_Activity</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>M_Property_ID</th>
<th>Activity_ID</th>
</tr>
</thead>
</table>

Activity

<table>
<thead>
<tr>
<th>Activity_ID</th>
<th>Activity</th>
</tr>
</thead>
</table>
b. All relations are in third normal form.

c. All relations are in third normal form.
d. An action assertion could be used such as:
A property shall have one and only one rental agreement in force at one time, where “in force” means that a given date falls between Begin_Date and End_Date.

15. Note: for this situation, we assume that Client information is stored even if there is not a corresponding instance of Retention with an Attorney and a Case. We also assume that Attorney information is stored even if there is not a corresponding instance of Retention with a Client and a Case.

All relations are in 3NF since there are no partial functional dependencies or transitive dependencies. It could be argued that zip code is determined by city and state (also by address with 9-digit zip codes). This could be a good point to discuss in class.
16.

**Notes to instructor:**
- Purchase date information refers to a Customer’s purchase of a pet from the store.
- Delivery date information refers to a Store’s receipt of a shipment of a pet from the pet’s supplier.
- Date of Last Purchase can be determined by viewing the data stored in the delivery date fields.
- Last_Customer_Name can be determined by viewing the data stored in the Customer_Name fields.
- The problem states that “Peter Corona wants to implement a Web-enabled database application” for use by his branch stores for entering inventory, ordering, and sales data. This is a good opportunity to discuss with students the issues involved in converting a legacy, flat-file “system” that mixes together inventory, customer purchases, and store purchases into a relational design that will support a web application and future growth. Stress the importance of re-designing the data structure to support the tracking of all customer purchases and store deliveries, rather than just the four presented in the flat-file. You might also note that there appears to be some data missing from the “old” system that would be useful in managing inventory and sales data at the stores (e.g., quantity sold, quantity purchased, quantity received, etc.).
Solution to 16a
b. This structure is not a relation nor is it in any normal form due to the presence of Delivery_Date1-4, Purchase_Date1-4, and Customer_Name1-4 attributes. These attributes appear to suggest that they are repeating groups. In order to show a relation in 1NF, repeating groups must be removed. Peter might change his structure to the following 1NF solution, which will result in having more data instances in his structure to store the same data that is currently being stored in his flat-file “system”. The attributes of Last_Customer_Name, Date_Last_Purchase, and Date_Last_Delivery are not stored as part of the proposed 1NF solution as this data can be determined by viewing the stored data in the revised relation below.
17.
18.
Revised normalized data model:

```
STORE
- Store_ID
- Store_Name

PET
- Pet_ID
- Pet_Name
- Pet_Description
- Price
- Cost

Delivery
- Store_ID
- Pet_ID
- Delivery_Date

SUPPLIER
- Supplier_ID
- Supplier_Name

Supplies
- Supplier_ID
- Pet_ID
- Shipping_Time

Purchase
- Customer_ID
- Store_ID
- Pet_ID
- Purchase_Date

CUSTOMER
- Customer_ID
- Customer_Name
```
Revised E-R Diagram:
19. a.
Solution to 19b

**MENU**

- **Menu_ID**
- **Menu_Type**
- **Menu_Description**

**DISH**

- **Dish_ID**
- **Dish_Name**
- **Prep_Time**

**EVENT**

- **Event_ID**
- **Event_Date**
- **Event_Location**
- **Event_Time**
- **Menu_ID**

**STAFF**

- **Emp_ID**
- **Name**
- **Salary**
- **Supervisor_ID**

**Skills**

- **Emp_ID**
- **Skill**
c. All relations are in 3NF.

20.

a. Chapter 3, Problems & Exercise 5b Relations

![Database Diagram]

- CHEMIST
  - Employee_ID
  - Name
  - Phone_No

- PROJECT
  - Project_ID
  - Start_Date

- ASSIGNMENT
  - Assign_ID
  - Assign_Date
  - Employee_ID
  - Project_ID
  - Serial_No

- EQUIPMENT
  - Serial_No
  - Cost
b. Chapter 3, Problems & Exercise 5g Relations

ARTIST

<table>
<thead>
<tr>
<th>Artist_ID</th>
<th>Artist_Name</th>
<th>Artist DOB</th>
<th>Artist Death Date</th>
</tr>
</thead>
</table>

ARTWORK

<table>
<thead>
<tr>
<th>Item Code</th>
<th>Item Title</th>
<th>Item Type</th>
<th>Height</th>
<th>Width</th>
<th>Weight</th>
<th>Museum Location</th>
<th>Item Status</th>
<th>Artist ID</th>
</tr>
</thead>
</table>

LOAN

<table>
<thead>
<tr>
<th>Loan Code</th>
<th>Item Loan Date</th>
<th>Item Return Date</th>
<th>Gallery ID</th>
<th>Item Code</th>
</tr>
</thead>
</table>

GALLERY

<table>
<thead>
<tr>
<th>Gallery ID</th>
<th>Gallery Name</th>
<th>Gallery City</th>
</tr>
</thead>
</table>

PARTICIPATION

<table>
<thead>
<tr>
<th>Participation Code</th>
<th>Show ID</th>
<th>Item Code</th>
</tr>
</thead>
</table>

SHOW

<table>
<thead>
<tr>
<th>Show ID</th>
<th>Show City</th>
<th>Show Start Date</th>
<th>Show End Date</th>
</tr>
</thead>
</table>
c. Chapter 3, Problems & Exercise 5h Relations

DEFENDANT

DEFENSE_CLAIM

CASE

PLAINTIFF_CLAIM

PLAINTIFF

REQUESTED_JUDGMENT

Published by www.emagzine.org
d. Chapter 3, Problems & Exercise 5i Relations

![Diagram of database relations]
Chapter 5

Chapter 3, Problems & Exercise 14 Relations

CONCERT SEASON

CONCERT

CONCERT_DATE

PERFORMANCE

SOLO-PERFORM

SOLOIST

COMPOSITION

MOVEMENT

Published by www.emagzine.org
f. Chapter 3, Problems & Exercise 17 Relations
g. Chapter 4, Problems & Exercise 17 Relations
h. Chapter 4, Problems & Exercise 18 Relations
i. Chapter 4, Problems & Exercise 24 Relations

THEATRE

```
| Theatre_ID | Theatre_Name | Location |
```

SCREEN

```
| Screen_No   | Seating_Capacity | Theatre_ID |
```

TIMESLOT

```
| Slot_ID     | Screen_No       | Slot_Date  | Slot_Start_Time | Slot_End_Time | Price_Charged | Attendance |
```

SCHEDULE

```
| Schedule_ID | Movie_Seq_No | Slot_ID  | Screen_No | Movie_ID |
```

MOVIE

```
| Movie_ID   | Movie_Title   | Movie_Duration | Movie_Type |
```
Suggestions for Field Exercises

1. For this exercise, we suggest you interview at least two organizations: a manufacturing company and a service sector organization. (You may choose to combine this exercise with Field Exercise 2 in Chapter 4). First, determine what methodology, if any, each uses for conceptual design: E-R diagrams, object diagrams, etc. Then determine how these models are transformed into logical data models (relational schema, object-oriented designs, etc.). To what extent are these activities automated through the use of CASE tools? If the target data model is relational, determine the role of normalization: who is responsible for normalization, to what level is it performed, and how are users involved (if at all) in these activities?

2. We suggest you first perform this as an in-class exercise with student participation in the process. Bring a copy of your own document to class, and ask the students to volunteer a document as well. This provides students with valuable “hands-on” experience in the bottom-up design process.

3. (You may choose to combine this exercise with Exercise 1 above). Student answers will vary depending on the organizations and the person who is the subject of the interview. We suggest that you use the student interview notes as a means of stimulating class discussion regarding the amount of variation in practice regarding the extent of normalization activities.

4. This exercise is really a continuation of Exercise 2 above, now possibly applied to a more complex document. Use a report or other document that has detail lines and requires the use of normalization skills.

Project Case

Case Questions

1. Mountain View Community Hospital will continue to use relational technology for several reasons:
   a. The present IS staff is trained and experienced in using this technology.
   b. The present relational systems are stable and support existing operations quite well.
   c. Conversion to newer technology would be costly and would entail a number of risks.

2. Yes, Mountain View Community Hospital should use normalization in designing its relational database. Normalization helps avoid anomalies that impair data quality. Normalization shows data in its most simple, atomic form, but does not constrain physical database design. (See Chapter 6.)
3. Entity integrity helps assure that two real-world entities (such as patient or tests) are not confused. Also, it makes sure that no two instances of the same type of entity are confused. Certainly, primary keys are not null. Foreign keys which point to optional members in a relationship may be null. Often, when master data (strong entities) are entered, not all data are known, so many attributes may be null (e.g., Patient Home Address). For transactional data, often associative entities, most attributes of the transaction will be known (e.g., the date a test was done).

4. Referential integrity helps assure that one real-world entity (such as a test result) is not lost or disassociated from its owner entity (such as patient).

5. The attribute chosen to be a relation’s primary key must be unique, unlikely to change over time, and should not be an intelligent key. Although Social Security number, physician license number, DEA registration number, and a hospital assigned Physician_ID will all meet the uniqueness criterion of a primary key, there are other issues that should be reviewed. One scenario to consider is that of whether a physician’s license number or DEA registration number could possibly change over time – for instance, what happens if a license or DEA registration number is suspended or revoked? Are those numbers re-used by these external-to-the-hospital organizations that issued these numbers to physicians? Another issue to consider is that of physician privacy rights and possible identity theft – each of the non-hospital assigned attributes (Social Security number, license number, DEA registration number) is sensitive information about the physician that could potentially be abused if stolen or not kept securely within the hospital records. Given these concerns, it would seem a better course of action to use the hospital-assigned Physician_ID as the primary key of the Physician relation.

6. An enterprise key will enable MVCH to have a primary key that does not have business meaning. This could be important since identifying information for a particular entity (such as Physician) could change easily in the medical environment. Also, if MVCH expands by merger, other acquired hospitals may have their own keys, and an enterprise key would make it easier to consolidate the data.

7. Often times, we may need to revise the EER model based upon normalization. Normalization suggests missing data elements from the original EER models. Also, because of the passing of time from one phase to the next, new requirements may emerge which need to be added to all data models.
Case Exercises
1. In this answer, we skip from an unnormalized relation to the answer in part b.
   a. i. Patient Bill
      1) Initial:

      ![Diagram of initial relation]

      2) After normalization:

      ![Diagram of normalized relations]
ii. Room Utilization Report

1) Initial

```
| Location | Date_Admitted | Accom | Patient_No | Patient_Name | Exp_Discharge_Date |
```

2) After normalization

```
ROOM

| Location | Accom |

| Occupies |

| Location | Date_Admitted | Patient_No |

PATIENT

| Patient_No | Patient_Name |

DISCHARGE

| Patient_No | Date_Admitted | Exp_Discharge_Date |
```
iii. Patient Display
1) Initial

2) After normalization
   PATIENT

   Admission

   ROOM

iv. Physician Report
1) Initial
2) After normalization:

PHYSICIAN

PATIENT

Treats

PROC_LOCATION

Published by www.emagazine.org
c. Merged relations:

```
invoice
  +------------------+
  | Invoice_No | Account_No | Invoice_Date | Due_Date | Date_Admitted | Date_Discharged |
  +------------------+
  | account
  +------------------+
  | Account_No | Patient_No |
  +------------------+
  | physician
  +------------------+
  | Physician_ID | Physician_Name |
  +------------------+
  | treat
  +------------------+
  | Patient_No | Patient_Name | Patient_Address | City | State | Zip | Insurance |
  +------------------+
  | item
  +------------------+
  | Item_Code | Description |
  +------------------+
  | room
  +------------------+
  | Location | Account | Extension |
  +------------------+
  | occupies
  +------------------+
  | Location | Date_Admitted | Patient_No |
  +------------------+
  | admission
  +------------------+
  | Patient_No | Date_Admitted | Date_Discharged | Location | Exp_Discharge_Date |
  +------------------+
```

d. One suggestion:
   Include a surrogate primary key for the Treats relation.

e. You could add a time column to the Treats relation. Alternately, you could assign a surrogate primary key to Treats that would be different for each treatment.
2.
   a. I would suggest a composite primary key made up of Patient#, Visit_Date and Visit_Time. This would take into consideration two visits on the same day.

   b. This table is not a relation since it is not even in 1NF due to repeating groups.

   c. There is a deletion anomaly since Social_Worker is not stored in another relation. For example, if we delete all records for patient # 4211, any reference to social worker Lynn Riley is gone.

      There is an update anomaly since a name change of the patient would need to be replicated across many relations.

      There is an insert anomaly since a patient cannot be added before visiting the MS Center.

   d. This assumes that the social worker always stays the same for the patient. If this were not the case, then the figure below would represent the functional dependencies:

   e.
If we consider that a social worker could be different for each visit, the 3NF relations would be as follows:

- **Patient**
  - **PK:** Patient_No
  - Name
  - First_Seen

- **Visit**
  - **PK:** Patient_No, Visit_Date, Visit_Time
  - Visit_Date
  - Visit_Time
  - Visit_Reason
  - New_Symptoms
  - Pain_Level
  - Social_Worker

- **Schedules**

```
f.
create table patient
    (patient_no        varchar2(5) primary key,
     name              varchar2(35),
     first_seen        date,
     social_worker     varchar2(35));

create table visit
    (patient_no        varchar2(5) references patient(patient_no),
     visit_date        date,
     visit_time        varchar2(5),
     visit_reason      varchar2(40),
     new_symptoms      varchar2(50),
     pain_level        integer,
     constraint visit_pk primary key (patient_no,visit_date,visit_time);```
3.

a.

b.
c.

[Database Schema Diagram]
Project Assignments

Notes to instructor:
The Chapter 4 EER is very complex to support the richly described Case situation. When the EER is mapped to a relational schema, there are over 40 relations that are created. Because of the degree of detail (and multiple pages) needed to show all of these relations, we have taken the following steps in preparing the solution to the Project Assignments in Chapter 5:

- For Project assignment 1, we have listed all the relations in shorthand form. We display primary keys with an underlined name (though attribute names with an underscore character are not easy to see in this case, for example Emp_ID) and foreign keys in Italics. Attributes that are both primary keys and foreign keys (e.g., a subtype’s primary key, or a weak entity’s primary key) are shown with both italics and underline.

- For all Project assignments, we developed surrogate keys for the relations necessary to support the multivalued attributes and repeating groups shown in the EER.

- For Project assignment 2, we show only the referential integrity dependency arrows in the solution. Additionally, we show the relations in shorthand form rather than use the traditional dependency diagram “boxes” in order to save space in the solution. The same notation for PK and FK as in Project assignment 1 is used in Project assignment 2.

- For Project assignment 3, we show only the referential integrity dependency arrows for the foreign keys relation-specific Object_IDs back to the primary key relation-specific Object_IDs (e.g., VOID in VOL_MVCH_SERVICE is a FK back to the PK of VOID in VOLUNTEER). It should be noted that the primary key relation-specific Object_IDs are also serving a foreign key role for referential integrity with the Object_ID in the OBJECT relation (e.g., VOID in VOLUNTEER is FK back to PK of OID in OBJECT).

P1. All relations mapped from Chapter 4 EERD.

[Note to Instructor: In this section of the solution, the primary key (PK) of each relation is underlined, though this tends to overlay the “_” character in each PK’s attribute name. Foreign Keys are shown in Italics, rather than with a double underline. PKs that are both a primary key and a foreign key are noted in Italics with an underline. Relation names are shown in Bold.]

PERSON (Person_ID, Person_Name, Person_Str_Address, Person_City, Person_State, Person_Zip, Person_Home_Phone, Person_Work_Phone, PersonDOB, Person_Email, Is_Physician, Is_Employee, Is_Volunteer, Is_Patient)
VOLUNTEER (Vol_ID, Had_Felony, Felony_Explanation, VEC_Last_Name,
VEC_First_Name, VEC_Relationship, VEC_Address, VEC_Phone, V_Employer,
V_Employ_Addr, V_Employ_Position, V_Employ_Start_Date,
V_Employ_End_Date, Had_MVCH_Service, Had_Vol_Exp, Why_Volunteer)

VOL_MVCH_SERVICE (VMS_ID, MVCH_Service_Info, Vol_ID)

VOL_REF_INFO (VRI_ID, VRI_Last_Name, VRI_First_Name, VRI_Relationship,
VRI_Phone, VRI_Address, VRI_City, VRI_State, VRI_Zip, Vol_ID)

VOL_EXPERIENCE (VE_ID, Volunteer_Exp_Info, Vol_ID)

VOL_LANGUAGE (VL_ID, Language, Vol_ID)

VOL_SKILL (VS_ID, Skill, Vol_ID)

VOL_INTEREST (VI_ID, Interest, Vol_ID)


VOL_SERV_HISTORY (VSH_ID, Service_Begin_Date, Service_End_Date,
Service_Hrs_Worked, Unit_Name, Vol_ID, Physician_ID, Emp_ID)

PHYSICIAN (Physician_ID, DEA_No, Pager_No, Specialty)

PHYSICIAN_DX (PD_ID, Diagnosis_Date, Diagnosis_Time, Physician_ID,
Patient_ID, Diagnosis_Code)

DIAGNOSIS (Diagnosis_Code, Diagnosis_Name)

PATIENT (Patient_ID, Contact_Date, EC_Last_Name, EC_First_Name,
EC_Relationship, EC_Address, EC_Phone, Company_Name, Policy_No,
Group_No, Company_Phone, Sub_Last_Name, Sub_First_Name,
Sub_Relationship, Sub_Address, Sub_Phone, Is_Outpatient, Is_Inpatient,
Admit_Phys, Refer_Phys)

RESIDENT (R_Patient_ID, Date_Admitted, Date_Discharged, Bed_No, Room_No)

OUTPATIENT (O_Patient_ID)

VISIT (Visit_No, Visit_Date, Visit_Time, Visit_Reason, O_Patient_ID)

BED (Room_No, Bed_No)

ROOM (Room_No, CC_Unit_Name)

CC_ASSIGNMENT (CCA_ID, Assign_Start, Assign_End, Hrs_Worked,
CC_Unit_Name, Nurse_ID)

CARE_CENTER (CC_Unit_Name, Day_in_Charge, Night_in_Charge)

RN (RN_ID)

LPN (LPN_ID, Supervisor)

NURSE (Nurse_ID, Cert_Degree, State_License_No, Nurse_Specialty, Nurse_Type)

FIELD_CERTIFICATION (FC_ID, FC_Description, Nurse_ID)

EMPLOYEE (Emp_ID, Date_Hired, Emp_Type)

Published by www.emagzine.org
STAFF (Staff_ID, Job_Class, Unit_Name)
TECHNICIAN (Technician_ID, Unit_Name)
TECHNICIAN_SKILL (TS_ID, Technician_ID, TS_Skill)
WORK_UNIT (Unit_Name, Facility_ID, Floor, Unit_Type)
FACILITY (Facility_ID, Facility_Name)
DIAGNOSTIC_UNIT (DX_Unit_Name)
TREATMENT (Trt_Code, Trt_Name, DX_Unit_Name)
TREATMENT_ORDER (TO_ID, Trt_Date, Trt_Time, Results, Trt_Code, Order_ID)
ORDER (Order_ID, Order_Date, Order_Time, Patient_ID, Physician_ID, Item_No)
ITEM (Item_No, Item_Desc, Item_Unit_Cost)
INVENTORY (Inv_ID, Item_No, Vendor_ID)
VENDOR (Vendor_ID, Vendor_Name)
ITEM_BILLING (IB_ID, Start_Date, End_Date, Cost, Item_No, Room_No, Patient_ID)
ITEM_CONSUMPTION (IC_ID, Consume_Date, Consume_Time, Consume_Qty, Item_No, Patient_ID)
ASSESSMENT (Assessment_ID, Assessment_Date, Assessment_Time, Comments, Patient_Weight, Patient_BP, Patient_pulse, Patient_temperature, Patient_ID, Nurse_ID)
SCHEDULE (Schedule_ID, Sched_Begin, Sched_End, Facility_ID, Physician_ID)
P2. Referential integrity functional dependency arrows for all relations – shown in 3 pages. Primary keys are underlined; foreign keys are in italics; attributes that are both primary and foreign keys are underlined and italicized.

Part 1: Person, Volunteer, associated Volunteer relations, and Employee
Part 2: Person (repeated), Physician, Patient, associated relations for Physician & Patient, Care_Center, Room, Bed, Nurse, Employee (repeated)
Part 3: Facility, Work_Unit, Treatment, Order, Item, Vendor and remaining relations

P3. Relations shown after adding an enterprise key to all relations – shown in 3 pages as in P2. Note that Primary Keys (PKs) in all relations serve a double purpose as a foreign key back to the OID primary key in the OBJECT relation. Primary keys are underlined; foreign keys are in italics; attributes that are both primary and foreign keys are underlined and italicized.

Part 1: Person, Volunteer, associated Volunteer relations, and Employee
Part 2: Person (repeated), Physician, Patient, associated relations for Physician & Patient, Care_Center, Room, Bed, Nurse, Employee (repeated)
Part 3: Facility, Work_Unit, Treatment, Order, Item, Vendor and remaining relations

P4. The affected sections of the EERD are shown over two pages for this solution. The first page shows the additional entities and any changes to entities, along with the relationships that arise out of resolving multivalued attributes and repeating groups shown in the chapter 4 EERD. The second page shows the entities and their respective attributes.
Figure IM5-PA2-1: Functional Dependencies of Project Assignment 2 (Part 1 of 3)

PERSON
(Person_ID, Person_Name, Person_Str_Address, Person_City, Person_State, Person_Zip, Person_Home_Phone, Person_Work_Phone, Person_DOB, Person_Email, Is_Physician, Is_Employee, Is_Volunteer, Is_Patient)

VOLUNTEER
(Vol_ID, Had_Feisty, Felony_Explanation, VEC_Last_Name, VEC_First_Name, VEC_Relationship, VEC_Address, VEC_Phone, V_Employer, V_Employer_Add, V_Employ_Position, V_Employ_Start_Date, V_Employ_End_Date, Had_MVCH_Serv, Had_Vol_Exp, Why_Volunteer)

VOL_MVCH_SERVICE
(VMS_ID, Vol_ID, MVCH_Service_Info)

VOL_REF_INFO
(VRI_ID, Vol_ID, VRI_Last_Name, VRI_First_Name, VRI_Relationship, VRI_Phone, VRI_Address, VRI_City, VRI_State, VRI_Zip)

VOL_EXPERIENCE
(VE_ID, Vol_ID, Volunteer_Exp_Info)

VOL_LANGUAGE
(VL_ID, Vol_ID, Language)

VOL_SKILL
(VS_ID, Vol_ID, Skill)

VOL_INTEREST
(VI_ID, Vol_ID, Interest)

VOL_AVAILABILITY

VOL_SERV_HISTORY
(VSH_ID, Service_Begin_Date, Service_End_Date, Service_Hrs_Worked, Vol_ID, Emp_ID, Unit_Name, Physician_ID)

EMPLOYEE
(Emp_ID, Date_Hired, Emp_Type)
Figure IM5-PA2-3: Functional Dependencies of Project Assignment 2 (Part 3 of 3)

[Diagram showing the relationships between entities such as FACILITY, SCHEDULE, WORK_UNIT, DIAGNOSTIC_UNIT, TREATMENT, STAFF, TECHNICIAN, TECHNICIAN_SKILL, TREATMENT_ORDER, ORDER, ITEM, INVENTORY, VENDOR, ITEM_BILLING, ITEM_CONSUMPTION, ASSESSMENT, and their dependencies, with arrows indicating the direction of the relationships.]
Figure IM5-PA3-2: Enterprise Keys of Project Assignment 3 (Part 2 of 3)
Chapter 5 Case Project Assignment P4
Recommended Revisions to EERD - Part 2: Entities and Attributes

**VOLUNTEER**
- Felony?
- Felony_Explanation
- V_Emer_Cntct_Info
- VEC_Last_Name
- VEC_First_Name
- VEC_Relationship
- VEC_Address, VEC_Phone
- V_Employer
- V_Employ_Addr
- V_Employ_Position
- V_Employ_Start_Date
- V_Employ_End_Date
- MVCH_Service?
- Volunteer_Exp?
- Why_Volunteer?
- Volunteer_Preference

**VOL_MVCH_SERVICE**
- VMS_ID
- MVCH_Service_Info

**VOL_REF_INFO**
- VRI_ID
- VRI_LName
- VRI_FName
- VRI_Relationship
- VRI_Phone
- VRI_Address
- VRI_City
- VRI_State
- VRI_Zip

**VOL_EXPERIENCE**
- VE_ID
- Volunteer_Exp_Info

**VOL_LANGUAGE**
- VL_ID
- Language

**VOL_SKILL**
- VS_ID
- Skill

**VOL_INTEREST**
- VI_ID
- Interest

**VOL_AVAILABILITY**
- VA_ID
- Day_of_Week
- Portion_of_Day

**VISIT**
- Visit_No
- Visit_Date
- Visit_Time
- Visit_Reason

**FIELD_CERTIFICATION**
- FC_ID
- FC_Description

**TECHNICIAN**
- Technician_ID

**SKILL**
- TS_ID
- TS_Skill

**CC_ASSIGNMENT**
- CCA_ID
- Assign_Start
- Assign_End
- Hrs_Worked

**TREATMENT_ORDER**
- TO_ID
- Trt_Date
- Trt_Time
- Results

**INVENTORY**
- Inv_ID

Published by www.emagzine.org
Chapter 6 Physical Database Design and Performance

Chapter Overview

This chapter presents the basic steps that are required to develop an effective physical database design. Physical database design is very important since it immediately impacts those factors that are important to the end user: data integrity and security, response times, user friendliness, and so on. First, we present a simple approach to estimating the volume of data in a database, as well as the probable data usage patterns. Next, we discuss issues associated with defining fields, including data type determination, coding and compression techniques, controlling data integrity, and handling missing data. We then discuss designing physical records and include an expanded section on denormalization. Vertical and horizontal partitioning are covered next. We describe the basic file organizations and the trade-offs that are typically involved in selecting a file organization. We examine the use of indexes and have added bitmap indexes to this section. File access performance is discussed, including a new discussion of RAID technology. The chapter concludes with a comparison of available database architectures. The chapter continues to emphasize the physical design process and the goals of that process.

Chapter Objectives

Specific student learning objectives are included at the beginning of the chapter. From an instructor's point of view, the objectives of this chapter are to:

1. Present physical database design as a critical element in achieving overall database objectives, rather than as an afterthought.
2. Ensure that students understand the factors that must be considered in distributing data effectively and how a simple model can be used to obtain at least a first-cut distribution.
3. Provide students with a sound understanding of the use of indexes and the trade-offs that must be considered in their use.
4. Ensure students understand that denormalization must be used with great care and for specific reasons.

Key Terms

<table>
<thead>
<tr>
<th>Bitmap Index</th>
<th>Hashed file organization</th>
<th>Physical record</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blocking factor</td>
<td>Hashing algorithm</td>
<td>Pointer</td>
</tr>
<tr>
<td>Data type</td>
<td>Horizontal partitioning</td>
<td>Redundant Array of Inexpensive Disks (RAID)</td>
</tr>
<tr>
<td>Denormalization</td>
<td>Index</td>
<td>Secondary key</td>
</tr>
<tr>
<td>Extent</td>
<td>Indexed file organization</td>
<td>Sequential file organization</td>
</tr>
<tr>
<td>Field</td>
<td>Join index</td>
<td>Stripe</td>
</tr>
<tr>
<td>File organization</td>
<td>Page</td>
<td>Tablespace</td>
</tr>
<tr>
<td>Hash index table</td>
<td>Physical file</td>
<td>Vertical partitioning</td>
</tr>
</tbody>
</table>
Classroom Ideas
1. Point out to your students that physical database design is the last step in the database design process. Suggest to your students that this step is where the "rubber meets the road," since regardless of how well previous steps may have been completed, users will not be happy with a sloppy physical design.
2. Discuss data volume and usage analysis, using Figure 6-1.
3. The Y2K problem can be used to highlight the process of selecting data types. The DATE data type is particularly interesting, both because of this problem, and because of its capability to calculate using date arithmetic in most RDBMSs.
4. Using examples of each type of constraint can bring the importance of data integrity home. For instance, you can encourage the students to think through on their own or in small groups, situations where default values will improve data integrity, and situations where using them may decrease data integrity. Range controls, null value controls, and referential integrity can also be addressed this way.
5. Discuss denormalization and the conditions under which this approach may be considered, referring to the discussion in the text. Be sure to describe the trade-offs in denormalization.
6. Start a discussion of indexes by having the students give common examples (index at the end of their database text, Yellow Pages, card catalog, etc.). Then describe and illustrate the use of each type of index using Figure 6-7.
7. Review the basic types of file organizations. Ask your students to give examples other than those described in the text for each organization.
8. Encourage a discussion of the comparative desirability of the various levels of RAID. Students will probably be surprised to figure out that the answer to this question will not be in sequential order, and they should understand that the ordering would be affected by the performance required by the systems.
9. Similarly, a comparison of the database architectures is a good opportunity to emphasize that hierarchical and network models are still in use, particularly for transaction processing systems, and that object-oriented databases are becoming much more common.
10. Query optimization is a fascinating subject that is only touched on lightly in the book. Further examples, perhaps taken from the work of Joe Celko (Joe Celko's SQL Puzzles and Answers, and so forth), will be interesting to the students if time permits.

Review Questions
1. Define each of the following terms:
   a. File organization A technique for physically arranging the records of a file on secondary storage devices.
   b. Sequential file organization Records in the file are stored in sequence according to a primary key value.
   c. Indexed file organization Records are either stored sequentially or non-sequentially, and an index is created that allows software to locate individual records.
   d. Hashing file organization The address for each record is determined using a hashing algorithm.
e. **Denormalization** The process of transforming normalized relations into unnormalized physical record specifications.

f. **Composite key** A key made up of more than one column.

g. **Secondary key** One or a combination of fields for which more than one record may have the same combination of values.

h. **Data type** Each unit of a detailed coding scheme recognized by system software, such as a DBMS, for representing organizational data.

i. **Bitmap index** A table of bits in which each row represents the distinct values of a key and each column is a bit, which when on indicates that the record for that bit column position has the associated field value.

j. **Redundant Arrays of Inexpensive Disks (RAID)** A set, or array, of physical disk drives that appear to the database user (and programs) as if they form one large, logical storage unit.

k. **Join index** An index on columns from two or more tables that come from the same domain of values.

l. **Stripe** The set of pages on all disks in a RAID that are the same relative distance from the beginning of the disk drive.

m. **Explain plan** A command available in most database management systems which displays how the query optimizer will access indexes, use parallel servers and join tables to prepare query results.

2. Match the following terms to the appropriate definitions:

   d. bitmap index
   f. hashing algorithm
   a. page
   g. physical record
   e. pointer
   b. blocking factor
   c. physical file

3. **Contrast the following terms:**

   a. **Horizontal partitioning; vertical partitioning** Horizontal partitioning is very similar to creating a supertype/subtype relationship because different types of the entity (where the subtype discriminator is the field used for segregating rows) are involved in different relationships, hence different processing. Neither horizontal nor vertical partitioning prohibits the ability to treat the original relation as a whole.

   b. **Physical file; tablespace** A physical file is a named portion of secondary memory (magnetic tape, hard disk) allocated for the purpose of storing records; a tablespace is a named set of disk storage elements in which physical files for database tables may be stored.

   c. **Physical record; physical file** A physical file is a named portion of secondary memory (a magnetic tape or hard disk) allocated for the purpose of storing physical records. Physical records of several types can be clustered together into one physical file in order to place records frequently used together close to one another in secondary memory.

   d. **Page; physical record** Often a page will store multiple physical records. Depending on the computer system, a lengthy physical record may or may not be allowed to span two pages.
e. **Secondary key; primary key** A secondary key is one or a combination of fields for which more than one record may have the same combination of values; the primary key is one or a combination of fields for which every record has a unique value. Hence, the primary key is a unique identifier for a row.

4. Three major inputs to physical design:
   a. Logical database structures developed during logical design
   b. User processing requirements identified during requirements definition
   c. Characteristics for the DBMS and other components of the computer operating environment

5. Key decisions in physical database design:
   a. Choosing the storage format (called *data type*) for each attribute from the logical data model: the format is chosen to minimize storage space and to maximize data integrity.
   b. Grouping attributes from the logical data model into *physical records*: you will discover that although the columns of a relational table are a natural definition for the contents of a physical record, this is not always the most desirable grouping of attributes.
   c. Arranging similarly structured records in secondary memory (primarily hard disks) so that individual and groups of records can be stored, retrieved, and updated rapidly (called *file organizations*): consideration must also be given to protecting data and recovering data after errors are found.
   d. Selecting structures for storing and connecting files to make retrieving related data more efficient (called *indexes* and *database architectures*).
   e. Preparing strategies for handling queries against the database that will optimize performance and take advantage of the file organizations and indexes that you have specified: efficient database structures will be of benefit only if queries and the database management systems that handle those queries are tuned to intelligently use those structures.

6. Composite usage maps:
   Figure 6-1 illustrates a composite usage map, which shows both data volume and access frequencies for the illustrative Pine Valley Furniture inventory system. The volume and frequency statistics are generated during the systems analysis phase of the systems development process when systems analysts are studying current and proposed data processing and business activities. The data volume statistics represent the size of the business, and should be calculated assuming business growth over a several year period. The access frequencies are estimated from the timing of events, transaction volumes, and reporting and querying activities. Since many databases support ad hoc accesses, and such accesses may change significantly over time, the access frequencies tend to be less certain than the volume statistics. Fortunately, precise numbers are not necessary. What is crucial is the relative size of the numbers, which will suggest where the greatest attention needs to be given in order to achieve the best possible performance. It might also be helpful to know if an access results in data creation, retrieval, update, or deletion. Such a refined description of access frequencies can be handled by additional notation on a diagram such as in Figure 6-1 or by text and tables kept in other documentation.
7. Developing field specifications:
   a. Define the data type used to represent values of the field.
   b. Establish data integrity controls for the field, including default values, ranges, null value controls, and referential integrity.
   c. Determine how missing values for the field will be handled.
   d. Other field specifications, such as display format, must be made as part of the total specification of the information system; those specifications are typically handled by programs rather than by the DBMS.

8. Selecting a field data type:
   These four objectives will have varying relative importance for different applications:
   a. Minimize storage space
   b. Represent all possible values
   c. Improve data integrity
   d. Support all data manipulations

9. Coding or compressing field values:
   Where attributes have a sparse set of values or a volume so large that considerable storage space will be consumed, possibilities for coding or compressing field values should be considered. Large data fields mean that data are further apart, which yields slower data processing. Where the set of valid values is small, translation into a code that requires less space is a possibility. Data compression techniques also use coding to reduce the storage space required for commonly recurring patterns of data.

10. Controlling field integrity:
    a. Specify default values.
    b. Specify a range or list of permissible values.
    c. Set null value permissions.
    d. Establish referential integrity.

11. Three ways to handle missing field values:
    a. *Substitute an estimate of the missing value:* for example, for a missing sales value when computing monthly product sales, use a formula involving the mean of the existing monthly sales values for that product indexed by total sales for that month across all products. Such estimates must be marked so users know that these are not actual values.
    b. *Track missing data* so that special reports and other system elements cause people to quickly resolve unknown values. Setting up a trigger in the database can accomplish this. A trigger is a routine that will automatically execute when some event occurs or time period passes. One trigger could log the missing entry to a file when a null or other missing value is stored, and another trigger could run periodically to create a report of the contents of this log file.
    c. *Perform sensitivity testing* so that missing data are ignored unless knowing a value might significantly change results. For example, if total monthly sales for a particular salesperson were almost over a threshold that would make a difference in that
person’s compensation, then attention would be drawn to the missing value. Otherwise, it would be ignored. This is the most complex of the methods mentioned; it requires the most sophisticated programming, which must be written in application programs since DBMSs do not have the sophistication to handle this method.

12. Effect of normalizing relations on physical record storage:
One goal of physical record design is efficient use of storage space. In most cases, the second goal of physical record design—efficient data processing—dominates the design process. Efficient processing of data, just like efficient accessing of books in a library, depends on how close together related data (or books) are. Often, all the attributes that appear within a relation are not used together, and data from different relations are needed together to answer a query or produce a report. Thus, although normalized relations solve data maintenance anomalies, normalized relations, if implemented one for one as physical records, may not yield efficient data processing.

13. Situations that suggest the possibility of denormalization:
   a. Two entities with a one-to-one relationship: Even if one of the entities is an optional participant, if the matching entity exists most of the time, then it may be wise to combine these two relations into one record definition (especially if the access frequency between these two entity types is high). Figure 6-3 shows student data with optional data from a standard scholarship application a student may complete. In this case, one record could be formed with four fields from the STUDENT and SCHOLARSHIP APPLICATION normalized relations. (Note: In this case, fields from the optional entity must have null values allowed.)
   b. A many-to-many relationship (associative entity) with nonkey attributes: Rather than joining three files to extract data from the two basic entities in the relationship, it may be advisable to combine attributes from one of the entities into the record representing the many-to-many relationship, thus avoiding one join in many data access modules. Again, this would be most advantageous if this joining occurs frequently. Figure 6-4 shows price quotes for different items from different vendors. In this case, fields from ITEM and PRICE QUOTE relations might be combined into one record to avoid having to join all three files together. (Note: This may create considerable duplication of data—in the example, the ITEM fields, such as Description, would repeat for each price quote—and excessive updating if duplicated data changes.)
   c. Reference data: Reference data exists in an entity on the one-side of a one-to-many relationship, and this entity participates in no other database relationships. You should seriously consider merging the two entities in this situation into one record definition when there are few instances of the entity on the many-side for each entity instance on the one-side. See Figure 6-5 in which several ITEMS have the same STORAGE INSTRUCTIONS, and STORAGE INSTRUCTIONS only relate to ITEMS. In this case, the storage instruction data could be stored in the ITEM record, creating, of course, redundancy and potential for extra data maintenance.
14. Advantages and disadvantages of horizontal and vertical partitioning:

**Advantages of partitioning:**

a. Efficiency: Data used together are stored close to one another and separate from data not used together.

b. Local Optimization: Each partition of data can be stored to optimize performance for its own use.

c. Security: Data not relevant to one group of users can be segregated from data they are allowed to use.

d. Recovery and uptime: Smaller files will take time to recover, and other files are still accessible if one file is damaged, so the effects of damage are isolated.

e. Load balancing: Files can be allocated to different storage areas (disks or other media), which minimize contention for access to the same storage area or even allows for parallel access to the different areas.

**Disadvantages of partitioning:**

a. Inconsistent access speed: Different partitions may yield different access speeds, thus confusing users. Also, when data must be combined across partitions, users may have to deal with significantly slower response times.

b. Complexity: Partitioning is usually not transparent to programmers, who will have to write more complex programs due to violations of third normal form.

c. Anomalies: Insertion, deletion, and update anomalies are possible, and special programming is required to avoid these problems.

d. Extra space and update time: Data may be duplicated across the partitions, taking extra storage space, compared to storing all the data in normalized files. Updates, which affect data in multiple partitions, can take more time than if one file were used.

15. Seven criteria for selecting a file organization:

a. Data retrieval speed

b. Data input and maintenance transaction processing throughput rate

c. Storage efficiency

d. Failure or data loss protection level

e. Frequency of data reorganization required

f. Ability to accommodate growth

g. Security protection provided

16. The desirability of a bitmap index:

A bitmap is ideal for attributes that have few possible values, which is not true for conventional tree indexes. A bitmap also often requires less storage space (possibly as little as 25 percent) than a conventional tree index (Schumacher, 1997), but for an attribute with many distinct values, a bitmap index can exceed the storage space of a conventional tree index. One bitmap can be used for multiple keys in order to perform searches on elements that would satisfy more than one condition at a time.

17. The benefits of a hash index table:

Using a hashing algorithm allows for rows stored independently of the address, so that whatever file organization makes sense can be used for storage. Also, because index tables
are much smaller than a data table, the index can be more easily designed to reduce the likelihood of key collisions or overflows.

18. The purpose of clustering data in a file:
Some database systems allow physical files to contain records with different structures, e.g., rows from different tables may be stored in the same disk area. This clustering reduces the time to access related records compared to the normal allocation of different files to different areas of a disk. Time is reduced since related records will be closer to each other than if the records are stored in separate files in separate areas of the disk.

19. Seven rules of thumb for choosing indexes for relational databases.
   a. Indexes are more useful on larger tables.
   b. Specify a unique index for the primary key of each table.
   c. Indexes are more useful for columns that frequently appear in WHERE clauses of SQL commands, either to qualify the rows to select (e.g., WHERE FINISH = ‘Oak’, for which an index on Finish would speed retrieval) or for linking (joining) tables (e.g., WHERE PRODUCT.PRODUCT_ID = ORDER_LINE.PRODUCT_ID, for which a secondary key index on Product_ID in the Order_Line table and a primary key index on Product_ID in the Product table would improve retrieval performance). In this second case, the index is on a foreign key in the Order_Line table that is used in joining tables.
   d. Use an index for attributes referenced in ORDER BY (sorting) and GROUP BY (categorizing) clauses. You do have to be careful, though, about these clauses. Be sure that the DBMS will, in fact, use indexes on attributes listed in these clauses (e.g., Oracle uses indexes on attributes in ORDER BY clauses but not GROUP BY clauses).
   e. Use an index when there is significant variety in the values of an attribute. Oracle suggests that an index is not useful when there are fewer than 30 different values for an attribute, and an index is clearly useful when there are 100 or more different values for an attribute. Similarly, an index will be helpful only if the results of a query that uses that index do not exceed roughly 20 percent of the total number of records in the file (Schumacher, 1997).
   f. Check your DBMS for the limit, if any, on the number of indexes allowable per table. Many systems permit no more than 16 indexes and may limit the size of an index key value (e.g., no more than 2000 bytes for each composite value). So, you may have to choose those secondary keys that will most likely lead to improved performance.
   g. Be careful about indexing attributes that have null values. For many DBMSs, rows with a null value will not be referenced in the index (so they cannot be found from an index search of ATTRIBUTE = NULL). Such a search will have to be done by scanning the file.

20. Two views of multidimensional databases:
   a. Multidimensional table: Each cell contains one or more simple attributes, and the dimensions are ways to categorize the raw data. These categories, or dimensions, are the factors on which users want to summarize or segment the data, such as time periods, geography, lines of business, or people. A cell contains data relevant to the
intersection of all of its dimension values. For example, a cell might hold the number of units sold attribute for a given time period, location, line of business, and salesperson.

b. **Star-schema**: At the center is a fact table, equivalent to the cell in the multidimensional view. This table contains all the raw attributes and a composite key made up of the primary keys of all the surrounding dimension tables. The surrounding dimension tables define each of the ways to categorize data, such as all the description data about each salesperson.

21. How can the use of the EXPLAIN command help in writing a more efficient query: EXPLAIN plan will show exactly how a query will be submitted to the DBMS for processing. We can see indexes used, servers used, and how tables will be joined. Different execution plans for the query written in several different ways will help identify the least-cost execution for a desired query. The query with the best performance can then be chosen.

22. Four options for optimizing query performance:
   a. The most common approach is to replicate the query so that each copy works against a portion of the database, usually a horizontal partition (sets of rows). The partitions need to be defined in advance by the database designer. The same query is run against each portion in parallel on separate processors, and the intermediate results from each processor are combined to create the final query result as if the query were run against the whole database.

   b. A second option for using parallel scanning of a table occurs when the query is written. When the DBMS parses a query, it decides, based on statistics it keeps about the database structure, the number of distinct values for fields; and the query, what the best strategy is—called a query plan—for accessing each index and table. The DBMS includes rules that consider these statistics, and the DBMS uses these rules to pick the best query plan. The module of the DBMS that does these calculations is called the cost-based optimizer. The query plan calculated by the cost-based optimizer says what table or index to use first and how to use it, then what table or index to use next, and so on. Oracle includes an ANALYZE command which collects these statistics and stores them for use by the DBMS. It is possible in some DBMS to give the DBMS a hint or suggestion within the query to force the DBMS to process the query in a certain way.

   c. Avoid the use of subqueries. SQL allows nesting of queries, or writing one query inside another. The types of queries are less efficient than queries that retrieve the same data set with the subquery.

   d. Break complex queries into multiple, simple parts. Because a DBMS may only have one index per query, it is often good to break a complex query down into multiple, simpler parts which each use an index. Then combine the results of the smaller queries together.
Answers to Problems and Exercises

1. a. STUDENT_ID in STUDENT because it is a primary key and the index would enforce uniqueness of the key; also, STUDENT_ID in STUDENT and in REGISTRATION is used in a WHERE clause for joining the STUDENT and REGISTRATION tables, so it likely makes sense to create an index on STUDENT_ID in REGISTRATION as well.

   GPA in STUDENT because it is a nonkey cluster attribute used to qualify record retrieval

   STUDENT_NAME in STUDENT because it is a nonkey attribute used to sort records

   STUDENT_ID, COURSE_ID in REGISTRATION because it is a concatenated primary key and the index would enforce uniqueness of the key

   b. CREATE UNIQUE INDEX STUPKINDX ON STUDENT (STUDENT_ID);
   CREATE INDEX STUDREGINDX ON REGISTRATION (STUDENT_ID);
   CREATE INDEX CLUST_INDX
   ON STUDENT (GPA)
   CLUSTER;
   CREATE INDEX NAMEINDX ON STUDENT (STUDENT_NAME);
   CREATE UNIQUE INDEX REGSPKINDX ON REGISTRATION (STUDENT_ID, COURSE_ID);

2. Assuming that Vendor_ID and Price hold data that fits the two byte field size:
   Vendor_ID    SMALLINT
   Address    VARCHAR(40)
   Contact_Name   VARCHAR(25)
   Item_ID    INTEGER
   Description   VARCHAR(35)
   Price     SMALLINT

3. The answer to this question will vary for different universities, but common situations are used here. Since the majority of students are accepted in the first year after high school graduation and university attendance is increasing, the average age of students in the first year of college would be a good choice for a default value. Often students are accepted at the different schools within the university after the second year. Therefore we need to add two years to the first answer if we design the system for a business school, for example. Degree seeking students are generally younger than non-degree seeking students, and the default value for this field might be a higher number for non-degree seeking students. Graduate students have already completed a degree, which usually takes at least four years, so a graduate university would also use a higher default value.

4. Since every student who hasn't explicitly declared a major of his or her choice would be assigned this value, it should be considered a default value. The null value is an empty value.
5. **EMPLOYEE_SCHEDULE**  
(Department_ID, Employee_ID, Where_Work, Employee_Name, Employee_Address, Date)

A many-to-many relationship (associative entity) with nonkey attributes: Rather than joining three files to extract data from the two basic entities in the relationship, it may be advisable to combine attributes from one of the entities into the record representing the relation in the many-to-many relationship, thus avoiding one join in many data access modules. This approach is advantageous as this joining will occur frequently.

**DEPARTMENT**  
(Department_ID, Manager_ID, Sales_Goal, Store_ID, Region, Manager_ID, Square_Feet)

This reference data denormalization option wouldn't be recommended since the table STORE is further related to a table MANAGER, and there are probably more than just a few departments in each STORE.

6. Sixteen records will fit on one page, using 3840 of the 4000 bytes available per page. This means that 4000 - 3840 = 160 bytes will be lost from each page since records may not span pages. A total of 1000/16 or 62.5 sets of 16 bytes will be needed—or 63 pages. Sixty-three pages will take up 252,000 bytes of storage.

7. Disadvantages of partitioning:
   a. *Inconsistent access speed:* Different partitions may yield different access speeds, which may confuse users.
   b. *Complexity:* Partitioning is usually not transparent to programmers, who will have to write more complex programs due to the violations of third normal form.
   c. *Anomalies:* The violations of third normal form will lead to anomalies.
   d. *Extra space and update time.* Duplication of data across the partitions will use more storage space than data stored in normalized files. Updates may well affect data in multiple partitions.

   Conditions that influence the use of partitioning:
   a. Criticalness of fast and consistent record access times to users
   b. Sophistication of in-house programming talent
   c. Storage space availability

8. A sequential file organization arranges the records in physical sequence based on one sorting criterion, so scanning the file in that sequence is possible and efficient. For sequential media, really the only practical scanning sequence is the sequence in which the records are stored. If the sequential file is, however, on a random access device, then although the storage sequence is still the fastest scanning sequence, other sequential scans are possible by using pointers to link the records in order for the sequences in addition to the physical sequence.
9.  
   a. Records can be accessed sequentially in two directions: from start to end and vice versa. As a corollary, it is easy to find the next or prior record in sequence if you now the address of a particular record.  
   b. What is described is a simple, bi-directional pointer --which will allow traversal through the records in two directions forward and backward—but does not allow for different sequences. However, by adding a pair of bi-directional pointers for each sequence, multiple sequences can be maintained without repeating the data.  

10. A row selection qualification clause will be used:  

    WHERE (Major = "MIS" or Major = "Computer Science") And Age > 25 And Marital_Status = "single") Or (Major = "Computer Engineering" And Marital_Status = "single" And Home_Zipcode = 45462). Indexes on these fields can be used to considerable advantage in this situation. Assume that each index qualification (e.g., Major = “MIS”) produces a list of record numbers for the records satisfying that qualification. Lists can be merged to process OR operators, and lists can be intersected to process AND operators. Indexes may be scanned in main memory, and the list operations also done without accessing secondary memory, thus composing the list of qualified records very quickly. Only then does secondary memory need to be accessed for only those records that satisfy the whole query.  

11. 

    | Marital_Status | 101 | 102 | 103 | 104 | 105 | 106 | 107 | 108 | 109 | 110 | 111 |
    |----------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
    | Single         | 0   | 0   | 0   | 1   | 1   | 1   | 1   | 0   | 1   | 0   | 1   |
    | Married        | 1   | 0   | 1   | 0   | 0   | 0   | 1   | 0   | 0   | 0   | 0   |
    | Other          | 0   | 1   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 1   | 0   |

A bitmap is ideal for attributes that have a few possible values. It requires less storage space (possibly as little as 25 percent) than a conventional tree index.
12. Consider Figure 6-7c.

When one of the leaves is full and a new record needs to be added to that leaf, the leaf node may be turned into an intermediate parent node. This node would then serve as an index to those records, and an additional leaf node (containing the actual records) will be attached to that intermediate node. The parent node will contain pointers to the records in the leaf node.

13. There are two different multidimensional models in Figure 6-12. Both view facts (e.g., sales) qualified by different dimensions (e.g., product, region, and month). The cube is really a way to display the data, and can be represented via the star-schema view in the database. A skeleton of the star schema view for the cube would be the following:

PRODUCT (Product_ID, other data describing products)
REGION (Region_ID, other data describing regions)
DATE (Date_ID, other data describing dates [months])
SALES (Product_ID, Region_ID, Date_ID, dollar sales, unit sales, other such facts)

14. A cluster is defined by the tables and the column or columns by which the tables are usually joined. The column by which they are joined (foreign key) would need to have the same
value in the two tables for the adjacent records. If the tables are populated with data before clustering occurs, this is much harder to achieve. Hence, in Oracle, tables are assigned to a cluster at the time of their creation.

15. In general, the answer depends on the number of processors available. For the general structure mentioned in the problem, each set of conditions within parentheses (called a conjunction) could be given to a separate processor, then the results from each processor would be intersected to obtain the final result. If more processors are available, then each condition within a conjunction could be assigned to its own processor.

As an illustration of what might typically happen, consider the qualification clause from Problem and Exercise 10:

```
SELECT STUDENT_ID, STUDENT_NAME
FROM STUDENT
WHERE (Major = "MIS" or Major = "Computer Science") And Age > 25 And
Marital_Status = "single" Or (Major = "Computer Engineering" And Marital_Status = "single" And Home_Zipcode = 45462);
```

There are two general approaches for parallel processing of a query:

1. To ensure that subsequent scans of this table are performed in parallel using at least three processors, you would first alter the structure of the table with the SQL command:

```
ALTER TABLE STUDENT PARALLEL 3,
```

and then run the query itself.

2. The second option would be to give the DBMS a hint within the query. This will force it to process the query in a certain way. In Oracle:

```
SELECT /*+ FULL(STUDENT) PARALLEL(STUDENT,3) */ COUNT(*)
FROM STUDENT
WHERE (Major = "MIS" or Major = "Computer Science") And Age > 25
And Marital_Status = "single" Or (Major = "Computer Engineering" And Marital_Status = "single" And Home_Zipcode = 45462);
```

would enforce a full scan on the table STUDENT, and its processing in parallel, by three CPUs. (Note: In Oracle, parallel processing is possible only when a table is scanned, not when it is accessed through an index.)
16.  

<table>
<thead>
<tr>
<th>CustRowID</th>
<th>OrderRowID</th>
<th>Cust#</th>
</tr>
</thead>
<tbody>
<tr>
<td>10001</td>
<td>30004</td>
<td>C2027</td>
</tr>
<tr>
<td>10002</td>
<td>30002</td>
<td>C1062</td>
</tr>
<tr>
<td>10003</td>
<td>30003</td>
<td>C1062</td>
</tr>
<tr>
<td>10004</td>
<td>30001</td>
<td>C3861</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

17.  

Notes to instructor: This problem & solution references an older version of Figure 6-1, where the QUOTATION table below is the same as the SUPPLIES table in 9e version of the textbook. Also note that the (40) and 40 numbers around QUOTATION are due to assumptions regarding raw number analysis of queries against the database. An alternative solution to the one below would allow for 90 accesses of QUOTATION/SUPPLIES due to the increase in the direct accesses to PURCHASED_PART (now 150+75=225; was 140+60=200 with 80 accesses).
18.
a.

It might also be useful to ask your students what type of additional usage data might need to be gathered to make a complete analysis.

b. Since only employees are using the system, one possibility might be to eliminate the regular customer and national account customer subtypes and only have one customer type. We might also assume that only national account customers would be serviced by employees. Depending on how much National Account Customer data there is, this and the Order Line entities might be merged.

19.
a. An index on the State field of the customer table would help with grouping and sorting. An index on Order_Date in the order table would help with qualifying to the desired period. If
rows are sorted by description within state, it would seem that a index on Product_Description in the product table might be helpful, but such an index would likely not be used by most DBMSs because the number of rows to sort within each category is so few.

b. An index on Product_Finish in the Product table would help to compute the totals by product finish.

c. An index on Order_Date would help to quickly find the desired orders, and an index on Order_ID in the order line table would speed joining.

d. An index on Product_Line_ID in product would speed grouping, and an index on Product_ID in the order line table would speed joining.

Suggestions for Field Exercises

1. A good starting point for the purposes of this assignment would be to identify any DBMSs that support complex data types like graphics, video, and sound. Object-oriented databases will stand out for their abilities in this regard. This is the newest DBMS technology and larger organizations are gaining experience with it by selectively using it when complex data or event-driven programming is appropriate for the application.

2. Students who investigate this question may become interested in understanding the difference between symmetric multiprocessing (SMP) and massively parallel processing (MPP), topics that are covered in more depth in Chapter 9. In a typical SMP architecture, the machine has up to a few dozen processors, and each processor shares all hardware resources, including memory, disks, and the system bus. Because each processor can see all of the available memory, communicating between processors is straightforward: one processor can easily view the results generated by another processor simply by looking at the appropriate section of memory. MPP machines support multiple nodes that each have their own private memory and that communicate via passing messages over an interconnect. SMP systems seem to be best suited for either mission-critical or OLTP applications, where the application's growth rate is slow and steady at less than 20 percent annually and the amount of raw data is in the range of 10 to 100GB. MPP systems are best suited for either complex analytical or very large decision-support applications.

Students may also discover that several options exist for breaking apart a query into modules that can be processed in parallel. All options are not available with every DBMS, and each DBMS often has unique options due to its underlying design.

3. Student answers will vary based on the organization, its standards, and the people they actually contact within the organization. Answers should evidence a belief (fully normalized or denormalized databases) as well as a rationale for the belief.
4. In choosing a file organization for a particular file in a database, students should find that database designers consider many of these seven factors:
   a. Fast data retrieval
   b. High throughput for processing data input and maintenance transactions
   c. Efficient use of storage space
   d. Protection from failures or data loss
   e. Minimizing need for reorganization
   f. Accommodating growth
   g. Security from unauthorized use

Secondary key indexes are important for supporting many reporting requirements and for providing rapid ad hoc data retrieval. Indexed sequential in comparison to indexed nonsequential allows more efficient use of space and faster sequential data processing without much, if any, sacrifice in random data record accessing speed. A bitmap is ideal for attributes that have even a few possible values, which is not true for conventional tree indexes. Indexes may be deleted because of storage space constraints or to reduce maintenance overhead.

5. Redundant Arrays of Inexpensive Disks (RAID) is hardware and software technology that helps the database designer accomplish parallel database processing. The result of parallel processing of data is that several input/output operations, when done in parallel, take as long as only one such operation. RAID-1 is best for fault-tolerant, database maintenance applications (those requiring a high percentage of up time) or when only two disk drives are affordable. RAID-5 is best for read-intensive applications against very large data sets and when at least three (and more typically five or more) disk drives are affordable. DBAs favor RAID-1 because of the fast recovery time when a problem develops with a disk, but they may not always be able to afford the required storage space.

Project Case

Case Questions
1. a. Data volume estimates (data volume and frequency-of-use statistics, representing them by adding notation to the EER diagram)
   b. Definitions of each attribute
   c. Descriptions of where and when data are used: entered, retrieved, deleted, and updated (including frequencies)
   d. Expectations or requirements for response time and data security, backup, recovery, retention, and integrity
   e. Descriptions of the technologies (database management systems) used for implementing the database. The efficient use of secondary storage is influenced both by the size of the physical record and the structure of secondary storage. Hence, we need to know the page size, whether a physical record is allowed to span two pages, and the blocking factor, etc.
2. Some types of data that are collected include medical records (e.g., scanned from manual systems), lab results, test results, handwritten doctor notes, doctor dictations, etc. Some data, such as audio clips, MRI images, or x-rays could not be captured using standard data types. Some items can be converted to alphanumeric datatypes (e.g., dictation can be transcribed). One option would be to use other datatypes such as binary large objects (BLOBS). Increasingly, jpeg, mpeg, and other rich media objects need to be accommodated.

3. Data partitioning would be beneficial if the database is going to be distributed among different machines. Horizontal partitioning distributes all the rows of a table in separate files, based upon common column values. When data are needed to be viewed together the SQL union operator may be used to display all rows of data in one table. Vertical partitioning, or distributing the columns of a relation in separate files by repeating the primary key for each file, would be another possibility. By joining the tables together, all data may be viewed together. Where the natural divisions are to partition the data are not clear. Some labs may run rather independently, so some data only for use in one or a few labs could be stored in a separate partition. However, the push for integrated, electronic medical records encourages one integrated database of medical data. Some hospitals use different applications for medical records and for practice management (the back office functions). Thus, these applications may be supported by separate partitions (possibly with some planned redundancy).

4. A bitmapped index could be used on fields such as Skill in the Volunteer and Technician relations, Specialty in the Physician relation and Job_Class in the Staff relation. Bitmapped indexes would work well in all of these cases because each column being indexed has a small number of possible values. One possible join index would be Care_Center_ID in the NURSE relation and the BED relation. This might be useful in a data warehousing application, particularly if you would like to report on what nurses could potentially work in what rooms and care for patients in what beds.

5. a. The following secondary indexes are recommended:
   - Treatment_Date on the order_treatment table since this indicates the date of treatment
   - Treatment_Code on the order_treatment table since we will group by treatment_code
   - P_Person_ID on the order_treatment table since this will link to the physician table
   - Physician_Name on the physician table since we will also want to search by physician_name

   b. The statements are:
      - `create index tdate on order_treatment(treatment_date);`
      - `create index tcode on order_treatment(treatment_code);`
      - `create index rphys on order_treatment(p_person_id);`
      - `create index rphys_name on physician(physician_name);`
6. RAID technology would be useful for the hospital, especially since it is critical to keep access to the data available 24 by seven. With RAID, we can be sure that there is some redundancy built in.

7. 
   a. MVCH could benefit from voluntary compliance since this would increase the quality of the data. While MVCH would have to provide more detailed reporting on financials, it still would stand to gain much in terms of quality which would be looked upon favorably by accrediting agencies such as JCAHCO. Also, SOX compliance might help MVCH justify some quality awards, which could be used for competitive advantage in promoting the hospital against alternative healthcare providers.

   b. Accuracy and completeness of MCVH is attained through including some range controls and other integrity controls during physical database design.

      Elimination of duplicates and data inconsistencies will be achieved by using a data model in 3NF (or with planned denormalization) as well as including appropriate referential integrity.

      MVCH data will be more understandable if some physical data naming conventions and standards are used.

Case Exercises

1. 
   a. Yes, I would change all TEXT to VARCHAR since this would dynamically allocate memory space for the fields and will help to minimize storage space.

   b. Yes, I would include lookup tables for Social_Worker, Pain_Level and reason for visit, for example.

   c. Patient_No, Name, and Social_Worker in the patient table should have values. First_seen could be null, since the patient info may be entered before the patient is seen.

      In the Visit table, new_symptoms could be null.

   d. I would write a trigger to write to an exceptions table. This table would then be used to generate a report. I would not use a default value for social worker. For reason_for_visit, I might use routine as the default value. Alternatively, applications that populate these fields could force these data to be included, otherwise a transaction (e.g., recording a visit) cannot be entered. It might be possible to reuse the most recent value for reason for a visit, assuming the most likely reason is a continuation of the most recent reason. However, in this case, or any case of entering a value for a missing field, it is best to indicate that the value stored is an estimate, and that the real value is unknown, until the value can be verified.
2.

a. Yes. I would create a UDT called test_results which contains the test_name, last date, and result. This would eliminate separate fields in the PATIENT table for each type test (MRI, FSS, etc.).

b. Yes, presenting symptoms and active meds in the visit table both could be coded with a list. Stage in the patient could also be coded with a list.

c. Active_meds could take on a null value, since the patient does not always need to be on meds. The same is true for last_MRI_results and MRI_Results since some patients will not have an MRI. The same would be true for FSS, EDSS and neuro assessment.

d. I do not see any possibilities for denormalization here.

e. Yes, one possibility is stage, since there would be a limited number.

f. I could see using a join index across all tables since MRN is a common field.
3. I would recommend clustering order and order_detail, using the order_id as the cluster key due to data volumes. SQL Server does support clustering through a cluster key, which physically rearranges the order of the records. The Oracle implementation actually stores the two or more tables next to each other on the disk.

4. VARCHAR is not available in Access. Also, the date datatype is a datetime datatype in Access.
6. a. Here is a table-by-table review:

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Design Changes/Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assignment_T</td>
<td>CareCenter_ID and Employee_ID should be of type text to all IDs that start with 0.</td>
</tr>
<tr>
<td></td>
<td>Hours_Worked could have a default value of 37.5 and should not allow negative values.</td>
</tr>
<tr>
<td>Bed_T</td>
<td>Bed_ID, CareCenter_ID and Room_ID should be text to allow IDs that start with 0.</td>
</tr>
<tr>
<td>Care_Center_T</td>
<td>CareCenter_ID and Employee_ID should be of type text to all IDs that start with 0.</td>
</tr>
<tr>
<td></td>
<td>CareCenter_Type could use a lookup table.</td>
</tr>
<tr>
<td></td>
<td>In_Charge should have a range of 0 or 1.</td>
</tr>
<tr>
<td>Consumes_T</td>
<td>Item_ID and Patient_ID should be of type text to all IDs that start with 0.</td>
</tr>
<tr>
<td></td>
<td>Time_Received could be stored inside of date_received since this is a date/time field.</td>
</tr>
<tr>
<td></td>
<td>Quantity should be required and should not allow anything less than 0.</td>
</tr>
<tr>
<td>Employee_T</td>
<td>Employee_ID and Person_ID should be of type text to all IDs that start with 0.</td>
</tr>
<tr>
<td></td>
<td>Employee_Type could use a lookup table.</td>
</tr>
<tr>
<td>Item_T</td>
<td>Item_ID should be of type text to all IDs that start with 0.</td>
</tr>
<tr>
<td></td>
<td>Item_cost should be greater than 0.</td>
</tr>
<tr>
<td>Patient_T</td>
<td>Patient_ID, Person_ID, and Bed should be of type text to all IDs that start with 0.</td>
</tr>
<tr>
<td></td>
<td>Contact_Time could be incorporated into Contact_Date since this is a date/time field.</td>
</tr>
<tr>
<td></td>
<td>Contact_Date should be today or before.</td>
</tr>
<tr>
<td></td>
<td>Admission_Type could use a lookup table.</td>
</tr>
<tr>
<td>Performs_T</td>
<td>Patient_ID, Physician_ID, and Treatment_ID should be of type text to all IDs that start with 0.</td>
</tr>
<tr>
<td></td>
<td>Treatment_time could be incorporated into Treatment_Date since this is a date/time field.</td>
</tr>
<tr>
<td>Person_T</td>
<td>Person_Number should be of type text to all IDs that start with 0.</td>
</tr>
<tr>
<td></td>
<td>Person_Birthdate should be today or earlier.</td>
</tr>
</tbody>
</table>

(6a solution continued on next page)
<table>
<thead>
<tr>
<th>Table Name</th>
<th>Design Changes/Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Physician_T</td>
<td>Person_Number should be of type text to all for IDs that start with 0.</td>
</tr>
<tr>
<td></td>
<td>Speciality could use a lookup table.</td>
</tr>
<tr>
<td>Treatment_T</td>
<td>Treatment_ID should be of type text to all for IDs that start with 0.</td>
</tr>
<tr>
<td>Volunteer_T</td>
<td>Volunteer_ID and Person_ID should be of type text to all for IDs that start with 0.</td>
</tr>
<tr>
<td></td>
<td>Skill could use a lookup table</td>
</tr>
</tbody>
</table>
c. A couple of possibilities:  
Consumes_T with Item_T using Item_ID  
Performs_T with Treatment_T using Treatment_ID
Project Assignments

P1. The data types for each field in the database are defined in P2 solution (the data dictionary). Following is a discussion of each of the raised issues:

P1.1 Opportunities for user-defined data types: User-defined data types are discussed in the context of evolving SQL standards in chapter 8 of the textbook. There do not appear to be any opportunities to use user-defined data types in this case exercise.

P1.2 Coding possibilities. The following fields are candidates for coding:

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Field Name</th>
<th>Coding Scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vol_Language</td>
<td>Language</td>
<td>List most common (e.g., French, Spanish, etc.)</td>
</tr>
<tr>
<td>Vol_Skill</td>
<td>Skill</td>
<td>List out possible values</td>
</tr>
<tr>
<td>Vol_Interest</td>
<td>Interest</td>
<td>List out possible values</td>
</tr>
<tr>
<td>Vol_Availability</td>
<td>Day_of_Week</td>
<td>List</td>
</tr>
<tr>
<td>Vol_Availability</td>
<td>Portion_of_Day</td>
<td>List</td>
</tr>
<tr>
<td>Employee</td>
<td>Emp_Type</td>
<td>Staff, Technician, Nurse</td>
</tr>
<tr>
<td>Nurse</td>
<td>Nurse_Type</td>
<td>RN, LPN</td>
</tr>
<tr>
<td>Physician</td>
<td>Physician_Specialty</td>
<td>List those recognized by professional medical associations</td>
</tr>
<tr>
<td>Nurse</td>
<td>Nurse_Specialty</td>
<td>List those recognized by professional medical associations</td>
</tr>
<tr>
<td>Work_Unit</td>
<td>Unit_Type</td>
<td>CC, DU</td>
</tr>
<tr>
<td>Technician_Skill</td>
<td>TS_Skill</td>
<td>List recognized professional skills</td>
</tr>
<tr>
<td>Staff</td>
<td>Job_Class</td>
<td>List recognized job classes for MVCH</td>
</tr>
</tbody>
</table>
P1.3 **Null values.** The following fields could have null values (selected examples), however it might be a good design idea to set up default values for a “not applicable” value for text-based data rather than leaving fields null:

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Field Name</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assessment</td>
<td>Comments</td>
<td>There could be no comments at time of assessment.</td>
</tr>
<tr>
<td>Treatment_Order</td>
<td>Results</td>
<td>A treatment’s results might not be filled in yet.</td>
</tr>
<tr>
<td>Item_Billing</td>
<td>End_Date</td>
<td>It might not be the end of an item’s usage for billing yet.</td>
</tr>
<tr>
<td>Person</td>
<td>Person_Work_Phone</td>
<td>A patient that is a child/Minor may not have a work phone</td>
</tr>
<tr>
<td>Volunteer</td>
<td>Felony_Explanation</td>
<td>No felony to report.</td>
</tr>
<tr>
<td></td>
<td>V_Employer_Addr, Position,</td>
<td>Volunteer may not be working, or with current work may not have an end-date</td>
</tr>
<tr>
<td></td>
<td>Start_Date, End_Date</td>
<td></td>
</tr>
<tr>
<td>Vol_Serv_History</td>
<td>Service_End_Date</td>
<td>May be currently volunteering, thus no end date yet.</td>
</tr>
<tr>
<td>Resident</td>
<td>Date_Discharged</td>
<td>Not yet discharged</td>
</tr>
<tr>
<td>CC_Assignment</td>
<td>Assign_End</td>
<td>Assignment not yet completed</td>
</tr>
</tbody>
</table>

P1.4 **Indexed fields:** Primary key and Foreign Key fields should be indexed. Additional suggestions for secondary indexes are noted in the solution to P2.
P2. Please see the table-by-table solution for the data dictionary; the solution assumes usage of Oracle data types for fields, there may be slight differences for MS SQL Server, MySQL, or MS Access implementation specifics.

**Table: Person**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Length</th>
<th>Min</th>
<th>Max</th>
<th>Description</th>
<th>Index</th>
<th>PK</th>
<th>FK</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>Person_ID</td>
<td>Varchar</td>
<td>5</td>
<td></td>
<td></td>
<td>Person ID</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Person_Name</td>
<td>Varchar</td>
<td>35</td>
<td></td>
<td></td>
<td>Name</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Person_Str_Address</td>
<td>Varchar</td>
<td>20</td>
<td></td>
<td></td>
<td>Address</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Person_City</td>
<td>Varchar</td>
<td>20</td>
<td></td>
<td></td>
<td>City</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Person_State</td>
<td>Varchar</td>
<td>2</td>
<td></td>
<td></td>
<td>State</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Person_Zip</td>
<td>Varchar</td>
<td>10</td>
<td></td>
<td></td>
<td>ZipCode</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Person_Home_Phone</td>
<td>Varchar</td>
<td>14</td>
<td></td>
<td></td>
<td>Phone Number</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Person_Work_Phone</td>
<td>Varchar</td>
<td>14</td>
<td></td>
<td></td>
<td>Phone Number</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Person_DOB</td>
<td>Date</td>
<td></td>
<td></td>
<td></td>
<td>Date of Birth</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Person_Email</td>
<td>Varchar</td>
<td>25</td>
<td></td>
<td></td>
<td>E-mail address</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Is_Physician</td>
<td>Char</td>
<td>1</td>
<td>“N”</td>
<td>“Y”</td>
<td>Subtype Discriminator</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Is_Employee</td>
<td>Char</td>
<td>1</td>
<td>“N”</td>
<td>“Y”</td>
<td>Subtype Discriminator</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Is_Volunteer</td>
<td>Char</td>
<td>1</td>
<td>“N”</td>
<td>“Y”</td>
<td>Subtype Discriminator</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Is_Patient</td>
<td>Char</td>
<td>1</td>
<td>“N”</td>
<td>“Y”</td>
<td>Subtype Discriminator</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table: Physician**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Length</th>
<th>Min</th>
<th>Max</th>
<th>Description</th>
<th>Index</th>
<th>PK</th>
<th>FK</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>Physician_ID</td>
<td>Varchar</td>
<td>5</td>
<td></td>
<td></td>
<td>Person ID</td>
<td>Y</td>
<td>Y</td>
<td></td>
<td>Person(Person_ID)</td>
</tr>
<tr>
<td>DEA_No</td>
<td>Varchar</td>
<td>20</td>
<td></td>
<td></td>
<td>DEA number</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pager_no</td>
<td>Varchar</td>
<td>14</td>
<td></td>
<td></td>
<td>Number for pager</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specialty</td>
<td>Varchar</td>
<td>20</td>
<td></td>
<td></td>
<td>Physician medical specialty</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Table: Physician_DX

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Length</th>
<th>Min</th>
<th>Max</th>
<th>Description</th>
<th>Index</th>
<th>PK</th>
<th>FK</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>PD_ID</td>
<td>Number</td>
<td>5</td>
<td></td>
<td></td>
<td>Unique ID for Physician &amp; Diagnosis instance</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Diagnosis_Date</td>
<td>Date</td>
<td></td>
<td></td>
<td></td>
<td>Date of diagnosis</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Diagnosis_Time</td>
<td>Timestamp</td>
<td></td>
<td></td>
<td></td>
<td>Time of diagnosis</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Physician_ID</td>
<td>Varchar</td>
<td>5</td>
<td></td>
<td></td>
<td>Physician ID who made the diagnosis</td>
<td>Y</td>
<td></td>
<td>Y</td>
<td>Physician(Physician_ID)</td>
</tr>
<tr>
<td>Patient_ID</td>
<td>Varchar</td>
<td>5</td>
<td></td>
<td></td>
<td>Patient receiving diagnosis</td>
<td>Y</td>
<td></td>
<td>Y</td>
<td>Patient(Patient_ID)</td>
</tr>
<tr>
<td>Diagnosis_Code</td>
<td>Varchar</td>
<td>5</td>
<td></td>
<td></td>
<td>Diagnosis Code</td>
<td>Y</td>
<td></td>
<td>Y</td>
<td>Diagnosis(Diagnosis_Code)</td>
</tr>
</tbody>
</table>

### Table: Diagnosis

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Length</th>
<th>Min</th>
<th>Max</th>
<th>Description</th>
<th>Index</th>
<th>PK</th>
<th>FK</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>Diagnosis_Code</td>
<td>Varchar</td>
<td>5</td>
<td></td>
<td></td>
<td>Hospital recognized codes for medical diagnoses</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Diagnosis_Name</td>
<td>Varchar</td>
<td>60</td>
<td></td>
<td></td>
<td>Name of diagnosis</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table: Employee

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Length</th>
<th>Min</th>
<th>Max</th>
<th>Description</th>
<th>Index</th>
<th>PK</th>
<th>FK</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>Emp_ID</td>
<td>Varchar</td>
<td>5</td>
<td></td>
<td></td>
<td>Person ID</td>
<td>Y</td>
<td></td>
<td>Y</td>
<td>Person(Person_ID)</td>
</tr>
<tr>
<td>Date_Hired</td>
<td>Date</td>
<td></td>
<td></td>
<td></td>
<td>Hire date of employee</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Emp_Type</td>
<td>Char</td>
<td>1</td>
<td>“N”</td>
<td>“T”</td>
<td>Subtype discriminator; values of N, S or T only</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Table: Technician

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Length</th>
<th>Min</th>
<th>Max</th>
<th>Description</th>
<th>Index</th>
<th>PK</th>
<th>FK</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>Technician_ID</td>
<td>Varchar</td>
<td>5</td>
<td></td>
<td></td>
<td>Person ID</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Employee(Emp_ID)</td>
</tr>
<tr>
<td>Unit_Name</td>
<td>Varchar</td>
<td>20</td>
<td></td>
<td></td>
<td>FK to Work_Unit where technician assigned</td>
<td></td>
<td>Y</td>
<td></td>
<td>Work_Unit(Unit_Name)</td>
</tr>
</tbody>
</table>

### Table: Technician_Skill

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Length</th>
<th>Min</th>
<th>Max</th>
<th>Description</th>
<th>Index</th>
<th>PK</th>
<th>FK</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>TS_ID</td>
<td>Varchar</td>
<td>5</td>
<td></td>
<td></td>
<td>Unique ID for Technician skill instance</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Technician_ID</td>
<td>Varchar</td>
<td>5</td>
<td></td>
<td></td>
<td>FK to Technician</td>
<td>Y</td>
<td></td>
<td></td>
<td>Technician(Technician_ID)</td>
</tr>
<tr>
<td>TS_Skill</td>
<td>Varchar</td>
<td>20</td>
<td></td>
<td></td>
<td>Specific recognized medical skill</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table: Staff

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Length</th>
<th>Min</th>
<th>Max</th>
<th>Description</th>
<th>Index</th>
<th>PK</th>
<th>FK</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>Staff_ID</td>
<td>Varchar</td>
<td>5</td>
<td></td>
<td></td>
<td>Person ID</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Employee(Emp_ID)</td>
</tr>
<tr>
<td>Job_Class</td>
<td>Varchar</td>
<td>3</td>
<td></td>
<td></td>
<td>Job classification code</td>
<td>Y</td>
<td></td>
<td></td>
<td>bitmapped</td>
</tr>
<tr>
<td>Unit_Name</td>
<td>Varchar</td>
<td>20</td>
<td></td>
<td></td>
<td>FK to Work_Unit where staff assigned</td>
<td></td>
<td>Y</td>
<td></td>
<td>Work_Unit(Unit_Name)</td>
</tr>
</tbody>
</table>

### Table: Work_Unit

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Length</th>
<th>Min</th>
<th>Max</th>
<th>Description</th>
<th>Index</th>
<th>PK</th>
<th>FK</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unit_Name</td>
<td>Varchar</td>
<td>20</td>
<td></td>
<td></td>
<td>Name of Work_Unit in hospital</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Floor</td>
<td>Varchar</td>
<td>3</td>
<td></td>
<td></td>
<td>Floor where work_unit is located</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Facility_ID</td>
<td>Varchar</td>
<td>10</td>
<td></td>
<td></td>
<td>FK to Facility</td>
<td>Y</td>
<td></td>
<td></td>
<td>Facility(Facility_ID)</td>
</tr>
<tr>
<td>Unit_Type</td>
<td>Char</td>
<td>2</td>
<td>“CC”</td>
<td>“DU”</td>
<td>Subtype discriminator; values of CC or DU only</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table: **Volunteer**

<table>
<thead>
<tr>
<th>Data Item</th>
<th>Type</th>
<th>Length</th>
<th>Min</th>
<th>Max</th>
<th>Description</th>
<th>Index</th>
<th>PK</th>
<th>FK</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vol_ID</td>
<td>Varchar</td>
<td>5</td>
<td></td>
<td></td>
<td>Person ID</td>
<td>Y</td>
<td>Y</td>
<td></td>
<td>Person(Person_ID)</td>
</tr>
<tr>
<td>Had_Felony</td>
<td>Char</td>
<td>1</td>
<td>“N”</td>
<td>“Y”</td>
<td>Yes/no for felony question</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Felony_Explanation</td>
<td>Varchar</td>
<td>50</td>
<td></td>
<td></td>
<td>Text explanation</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VEC_Last_Name</td>
<td>Varchar</td>
<td>20</td>
<td></td>
<td></td>
<td>Last name of emergency contact</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VEC_First_Name</td>
<td>Varchar</td>
<td>20</td>
<td></td>
<td></td>
<td>First name of emergency contact</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VEC_Relationship</td>
<td>Varchar</td>
<td>15</td>
<td></td>
<td></td>
<td>Relationship of emergency contact to volunteer</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VEC_Address</td>
<td>Varchar</td>
<td>55</td>
<td></td>
<td></td>
<td>Street, city, zip address of emergency contact</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VEC_Phone</td>
<td>Varchar</td>
<td>14</td>
<td></td>
<td></td>
<td>Phone number of emergency contact</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>V_Employer</td>
<td>Varchar</td>
<td>25</td>
<td></td>
<td></td>
<td>Name of volunteer’s employer</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>V_Employer_Addr</td>
<td>Varchar</td>
<td>55</td>
<td></td>
<td></td>
<td>Address of volunteer’s employer</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>V_Employ_Position</td>
<td>Varchar</td>
<td>20</td>
<td></td>
<td></td>
<td>Volunteer’s employment position</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>V_Employ_Start_Date</td>
<td>Date</td>
<td></td>
<td></td>
<td></td>
<td>Starting date of volunteer employment</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>V_Employ_End_Date</td>
<td>Date</td>
<td></td>
<td></td>
<td></td>
<td>Ending date of volunteer employment</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Had_MVCH_Serv</td>
<td>Char</td>
<td>1</td>
<td>“N”</td>
<td>“Y”</td>
<td>Yes/no answer to volunteer having served at MVCH</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Had_Vol_Exp</td>
<td>Char</td>
<td>1</td>
<td>“N”</td>
<td>“Y”</td>
<td>Yes/no answer to volunteer having experience</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Why_Volunteer</td>
<td>Char</td>
<td>50</td>
<td></td>
<td></td>
<td>Text explanation</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Table: **Vol_MVCH_Service**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Length</th>
<th>Min</th>
<th>Max</th>
<th>Description</th>
<th>Index</th>
<th>PK</th>
<th>FK</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>VMS_ID</td>
<td>Varchar</td>
<td>5</td>
<td></td>
<td></td>
<td>Unique ID for Volunteer’s MVCH Service experience</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MVCH_Service_Info</td>
<td>Varchar</td>
<td>25</td>
<td></td>
<td></td>
<td>Service information</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vol_ID</td>
<td>Varchar</td>
<td>5</td>
<td></td>
<td></td>
<td>FK to Volunteer</td>
<td>Y</td>
<td></td>
<td>Volunteer(Vol_ID)</td>
<td></td>
</tr>
</tbody>
</table>

### Table: **Vol_Ref_Info**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Length</th>
<th>Min</th>
<th>Max</th>
<th>Description</th>
<th>Index</th>
<th>PK</th>
<th>FK</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>VRI_ID</td>
<td>Varchar</td>
<td>5</td>
<td></td>
<td></td>
<td>Unique ID for Volunteer’s Reference Info instance</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VRI_Last_Name</td>
<td>Varchar</td>
<td>20</td>
<td></td>
<td></td>
<td>Last Name of Volunteer’s Reference person</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VRI_First_Name</td>
<td>Varchar</td>
<td>20</td>
<td></td>
<td></td>
<td>First Name of Volunteer’s Reference person</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VRI_Relationship</td>
<td>Varchar</td>
<td>15</td>
<td></td>
<td></td>
<td>Relationship of volunteer’s reference</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VRI_Phone</td>
<td>Varchar</td>
<td>14</td>
<td></td>
<td></td>
<td>Phone number of volunteer’s reference</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VRI_Address</td>
<td>Varchar</td>
<td>20</td>
<td></td>
<td></td>
<td>Street address of volunteer’s reference</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VRI_City</td>
<td>Varchar</td>
<td>20</td>
<td></td>
<td></td>
<td>City of volunteer’s reference</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VRI_State</td>
<td>Varchar</td>
<td>2</td>
<td></td>
<td></td>
<td>State of volunteer’s reference</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VRI_Zip</td>
<td>Varchar</td>
<td>10</td>
<td></td>
<td></td>
<td>Zip code of volunteer’s reference</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vol_ID</td>
<td>Varchar</td>
<td>5</td>
<td></td>
<td></td>
<td>FK to Volunteer</td>
<td>Y</td>
<td></td>
<td>Volunteer(Vol_ID)</td>
<td></td>
</tr>
</tbody>
</table>

### Table: **Vol_Experience**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Length</th>
<th>Min</th>
<th>Max</th>
<th>Description</th>
<th>Index</th>
<th>PK</th>
<th>FK</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>VE_ID</td>
<td>Varchar</td>
<td>5</td>
<td></td>
<td></td>
<td>Unique ID for Volunteer’s experience instance</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Volunteer_Exp_Info</td>
<td>Varchar</td>
<td>25</td>
<td></td>
<td></td>
<td>Experience information</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vol_ID</td>
<td>Varchar</td>
<td>5</td>
<td></td>
<td></td>
<td>FK to Volunteer</td>
<td>Y</td>
<td></td>
<td>Volunteer(Vol_ID)</td>
<td></td>
</tr>
</tbody>
</table>
### Table: Vol_Language

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Length</th>
<th>Min</th>
<th>Max</th>
<th>Description</th>
<th>Index</th>
<th>PK</th>
<th>FK</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>VL_ID</td>
<td>Varchar</td>
<td>5</td>
<td></td>
<td></td>
<td>Unique ID for Volunteer’s Language instance</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Language</td>
<td>Varchar</td>
<td>15</td>
<td></td>
<td></td>
<td>Language spoken by volunteer</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vol_ID</td>
<td>Varchar</td>
<td>5</td>
<td></td>
<td></td>
<td>FK to Volunteer</td>
<td>Y</td>
<td></td>
<td></td>
<td>Volunteer(Vol_ID)</td>
</tr>
</tbody>
</table>

### Table: Vol_Skill

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Length</th>
<th>Min</th>
<th>Max</th>
<th>Description</th>
<th>Index</th>
<th>PK</th>
<th>FK</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>VS_ID</td>
<td>Varchar</td>
<td>5</td>
<td></td>
<td></td>
<td>Unique ID for Volunteer’s skill instance</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Skill</td>
<td>Varchar</td>
<td>25</td>
<td></td>
<td></td>
<td>Skill information for volunteer</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vol_ID</td>
<td>Varchar</td>
<td>5</td>
<td></td>
<td></td>
<td>FK to Volunteer</td>
<td>Y</td>
<td></td>
<td></td>
<td>Volunteer(Vol_ID)</td>
</tr>
</tbody>
</table>

### Table: Vol_Interest

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Length</th>
<th>Min</th>
<th>Max</th>
<th>Description</th>
<th>Index</th>
<th>PK</th>
<th>FK</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>VI_ID</td>
<td>Varchar</td>
<td>5</td>
<td></td>
<td></td>
<td>Unique ID for Volunteer’s interest instance</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interest</td>
<td>Varchar</td>
<td>25</td>
<td></td>
<td></td>
<td>Interest information</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vol_ID</td>
<td>Varchar</td>
<td>5</td>
<td></td>
<td></td>
<td>FK to Volunteer</td>
<td>Y</td>
<td></td>
<td></td>
<td>Volunteer(Vol_ID)</td>
</tr>
</tbody>
</table>

### Table: Vol_Availability

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Length</th>
<th>Min</th>
<th>Max</th>
<th>Description</th>
<th>Index</th>
<th>PK</th>
<th>FK</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>VA_ID</td>
<td>Varchar</td>
<td>5</td>
<td></td>
<td></td>
<td>Unique ID for Volunteer’s availability instance</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Day_of_Week</td>
<td>Varchar</td>
<td>8</td>
<td></td>
<td></td>
<td>Day of the week</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Portion_of_Day</td>
<td>Varchar</td>
<td>15</td>
<td></td>
<td></td>
<td>Timeslot of availability</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vol_ID</td>
<td>Varchar</td>
<td>5</td>
<td></td>
<td></td>
<td>FK to Volunteer</td>
<td>Y</td>
<td></td>
<td></td>
<td>Volunteer(Vol_ID)</td>
</tr>
</tbody>
</table>
### Table: `Vol_Serv_History`

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Length</th>
<th>Min</th>
<th>Max</th>
<th>Description</th>
<th>Index</th>
<th>PK</th>
<th>FK</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>VSH_ID</td>
<td>Varchar</td>
<td>5</td>
<td></td>
<td></td>
<td>Unique ID for Volunteer’s Service history instance</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Service_Beg_Date</td>
<td>Date</td>
<td></td>
<td></td>
<td></td>
<td>Date of service (start)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Service_End_Date</td>
<td>Date</td>
<td></td>
<td></td>
<td></td>
<td>Date of service (end)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Service_Hrs_Worked</td>
<td>Varchar</td>
<td>3</td>
<td></td>
<td></td>
<td>Aggregate of hours worked for this particular service instance</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vol_ID</td>
<td>Varchar</td>
<td>5</td>
<td></td>
<td></td>
<td>FK to Volunteer</td>
<td>Y</td>
<td></td>
<td></td>
<td>Volunteer(Vol_ID)</td>
</tr>
<tr>
<td>Unit_Name</td>
<td>Varchar</td>
<td>15</td>
<td></td>
<td></td>
<td>FK to Work_Unit</td>
<td>Y</td>
<td></td>
<td></td>
<td>Work_Unit(Unit_Name)</td>
</tr>
<tr>
<td>Physician_ID</td>
<td>Varchar</td>
<td>5</td>
<td></td>
<td></td>
<td>FK to Physician; supervisor</td>
<td>Y</td>
<td></td>
<td></td>
<td>Physician(Physician_ID)</td>
</tr>
<tr>
<td>Emp_ID</td>
<td>Varchar</td>
<td>5</td>
<td></td>
<td></td>
<td>FK to Employee; supervisor</td>
<td>Y</td>
<td></td>
<td></td>
<td>Employee(Emp_ID)</td>
</tr>
</tbody>
</table>

### Table: `Outpatient`

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Length</th>
<th>Min</th>
<th>Max</th>
<th>Description</th>
<th>Index</th>
<th>PK</th>
<th>FK</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>O_Patient_ID</td>
<td>Varchar</td>
<td>5</td>
<td></td>
<td></td>
<td>Person ID</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Patient(Patient_ID)</td>
</tr>
</tbody>
</table>

### Table: `Resident`

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Length</th>
<th>Min</th>
<th>Max</th>
<th>Description</th>
<th>Index</th>
<th>PK</th>
<th>FK</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>R_Patient_ID</td>
<td>Varchar</td>
<td>5</td>
<td></td>
<td></td>
<td>Person ID</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Patient(Patient_ID)</td>
</tr>
<tr>
<td>Date_Admitted</td>
<td>Date</td>
<td></td>
<td></td>
<td></td>
<td>Date of Admission</td>
<td>Y Secondary</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Date_Discharged</td>
<td>Date</td>
<td></td>
<td></td>
<td></td>
<td>Date of discharge</td>
<td>Y Secondary</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bed_No</td>
<td>Varchar</td>
<td>3</td>
<td></td>
<td></td>
<td>Bed Number</td>
<td>Y</td>
<td></td>
<td></td>
<td>Bed(Room_No, Bed_No)</td>
</tr>
<tr>
<td>Room_No</td>
<td>Varchar</td>
<td>5</td>
<td></td>
<td></td>
<td>Room Number</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table: Visit

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Length</th>
<th>Min</th>
<th>Max</th>
<th>Description</th>
<th>Index</th>
<th>PK</th>
<th>FK</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>Visit_No</td>
<td>Varchar</td>
<td>5</td>
<td></td>
<td></td>
<td>Unique ID for patient’s visit instance</td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
</tr>
<tr>
<td>O_Patient_ID</td>
<td>Varchar</td>
<td>5</td>
<td></td>
<td></td>
<td>Outpatient Person ID</td>
<td></td>
<td>Y</td>
<td>Outpatient(O_Patient_ID)</td>
<td></td>
</tr>
<tr>
<td>Visit_Date</td>
<td>Date</td>
<td></td>
<td></td>
<td></td>
<td>Date of Visit</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Visit_Time</td>
<td>Timestamp</td>
<td></td>
<td></td>
<td></td>
<td>Time of Visit</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Visit_Reason</td>
<td>Varchar</td>
<td>50</td>
<td></td>
<td></td>
<td>Reason for visit</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table: Patient

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Length</th>
<th>Min</th>
<th>Max</th>
<th>Description</th>
<th>Index</th>
<th>PK</th>
<th>FK</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>Patient_ID</td>
<td>Varchar</td>
<td>5</td>
<td></td>
<td></td>
<td>Person ID</td>
<td></td>
<td>Y</td>
<td>Y</td>
<td>Person(Person_ID)</td>
</tr>
<tr>
<td>Contact_Date</td>
<td>Date</td>
<td></td>
<td></td>
<td></td>
<td>Date of first contact</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EC_Last_Name</td>
<td>Varchar</td>
<td>20</td>
<td></td>
<td></td>
<td>Emergency Contact (EC) Last Name</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EC_First_Name</td>
<td>Varchar</td>
<td>20</td>
<td></td>
<td></td>
<td>EC First Name</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EC_Relationship</td>
<td>Varchar</td>
<td>15</td>
<td></td>
<td></td>
<td>Relationship of EC to patient</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EC_Address</td>
<td>Varchar</td>
<td>55</td>
<td></td>
<td></td>
<td>Street, City, State, Zip address of EC</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Company_Name</td>
<td>Varchar</td>
<td>25</td>
<td></td>
<td></td>
<td>Name of Insurance Company</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Policy_No</td>
<td>Varchar</td>
<td>20</td>
<td></td>
<td></td>
<td>Insurance Policy number</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Group_No</td>
<td>Varchar</td>
<td>15</td>
<td></td>
<td></td>
<td>Group Number of Ins. Policy</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Company_Phone</td>
<td>Varchar</td>
<td>14</td>
<td></td>
<td></td>
<td>Phone for Insurance Company</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EC_Phone</td>
<td>Varchar</td>
<td>14</td>
<td></td>
<td></td>
<td>Phone for EC</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sub_Last_Name</td>
<td>Varchar</td>
<td>20</td>
<td></td>
<td></td>
<td>Insurance Subscriber Last Name</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sub_First_Name</td>
<td>Varchar</td>
<td>20</td>
<td></td>
<td></td>
<td>Insurance Subscriber First Name</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sub_Relationship</td>
<td>Varchar</td>
<td>15</td>
<td></td>
<td></td>
<td>Relationship of Insurance subscriber to patient</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sub_Address</td>
<td>Varchar</td>
<td>55</td>
<td></td>
<td></td>
<td>Street, City, State, Zip address of Ins. Subscriber</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sub_Phone</td>
<td>Varchar</td>
<td>14</td>
<td></td>
<td></td>
<td>Phone for Ins. Subscriber</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Is_Outpatient</td>
<td>Char</td>
<td>1</td>
<td>“N”</td>
<td>“Y”</td>
<td>Subtype Discriminator</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Is_Resident</td>
<td>Char</td>
<td>1</td>
<td>“N”</td>
<td>“Y”</td>
<td>Subtype Discriminator</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Admit_Phys</td>
<td>Varchar</td>
<td>5</td>
<td></td>
<td></td>
<td>FK to Admitting Physician</td>
<td>Y</td>
<td></td>
<td></td>
<td>Physician(Physician_ID)</td>
</tr>
<tr>
<td>Refer_Phys</td>
<td>Varchar</td>
<td>5</td>
<td></td>
<td></td>
<td>FK to Referring Physician</td>
<td>Y</td>
<td></td>
<td></td>
<td>Physician(Physician_ID)</td>
</tr>
</tbody>
</table>
### Table: Bed

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Length</th>
<th>Min</th>
<th>Max</th>
<th>Description</th>
<th>Index</th>
<th>PK</th>
<th>FK</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bed_No</td>
<td>Varchar</td>
<td>5</td>
<td></td>
<td></td>
<td>Bed Number</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Room_No</td>
<td>Varchar</td>
<td>5</td>
<td></td>
<td></td>
<td>Room Number</td>
<td>Y</td>
<td>Y</td>
<td>Room(Room_No)</td>
<td></td>
</tr>
</tbody>
</table>

### Table: Room

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Length</th>
<th>Min</th>
<th>Max</th>
<th>Description</th>
<th>Index</th>
<th>PK</th>
<th>FK</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>Room_No</td>
<td>Varchar</td>
<td>5</td>
<td></td>
<td></td>
<td>Room Number</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CC_Unit_Name</td>
<td>Varchar 20</td>
<td></td>
<td></td>
<td></td>
<td>FK to Care Center table</td>
<td>Y</td>
<td></td>
<td></td>
<td>Care Center(CC_Unit_Name)</td>
</tr>
</tbody>
</table>

### Table: Care_Center

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Length</th>
<th>Min</th>
<th>Max</th>
<th>Description</th>
<th>Index</th>
<th>PK</th>
<th>FK</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>CC_Unit_Name</td>
<td>Varchar   20</td>
<td></td>
<td></td>
<td></td>
<td>Care Center Name</td>
<td>Y</td>
<td>Y</td>
<td></td>
<td>Work_Unit(Unit_Name)</td>
</tr>
<tr>
<td>Day_in_Charge</td>
<td>Varchar   5</td>
<td></td>
<td></td>
<td></td>
<td>FK to RN; shows daytime in-</td>
<td>Y</td>
<td></td>
<td></td>
<td>RN(RN_ID)</td>
</tr>
<tr>
<td>Night_in_Charge</td>
<td>Varchar  5</td>
<td></td>
<td></td>
<td></td>
<td>FK to RN; shows nighttime-in-</td>
<td>Y</td>
<td></td>
<td></td>
<td>RN(RN_ID)</td>
</tr>
</tbody>
</table>

### Table: RN

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Length</th>
<th>Min</th>
<th>Max</th>
<th>Description</th>
<th>Index</th>
<th>PK</th>
<th>FK</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>RN_ID</td>
<td>Varchar</td>
<td>5</td>
<td></td>
<td></td>
<td>ID of RN</td>
<td>Y</td>
<td>Y</td>
<td></td>
<td>Nurse(Nurse_ID)</td>
</tr>
</tbody>
</table>

### Table: LPN

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Length</th>
<th>Min</th>
<th>Max</th>
<th>Description</th>
<th>Index</th>
<th>PK</th>
<th>FK</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>LPN_ID</td>
<td>Varchar</td>
<td>5</td>
<td></td>
<td></td>
<td>ID of LPN</td>
<td>Y</td>
<td>Y</td>
<td></td>
<td>Nurse(Nurse_ID)</td>
</tr>
<tr>
<td>Supervisor</td>
<td>Varchar</td>
<td>5</td>
<td></td>
<td></td>
<td>FK to RN</td>
<td>Y</td>
<td>Y</td>
<td></td>
<td>RN(RN_ID)</td>
</tr>
</tbody>
</table>
### Table: CC_Assignment

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Length</th>
<th>Min</th>
<th>Max</th>
<th>Description</th>
<th>Index</th>
<th>PK</th>
<th>FK</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCA_ID</td>
<td>Varchar</td>
<td>5</td>
<td></td>
<td></td>
<td>Unique ID for Care Center assignment instance</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Assign_Start</td>
<td>Date</td>
<td></td>
<td></td>
<td></td>
<td>Start date of assignment to Care Center</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Assign_End</td>
<td>Date</td>
<td></td>
<td></td>
<td></td>
<td>End date of assignment to Care Center</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hrs_Worked</td>
<td>Number</td>
<td>4,2</td>
<td></td>
<td></td>
<td>Aggregate number of hours worked during assignment</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CC_Unit_Name</td>
<td>Varchar</td>
<td>20</td>
<td></td>
<td></td>
<td>FK to Care_Center of assignment</td>
<td>Y</td>
<td></td>
<td></td>
<td>Care_Center(CC_Unit_Name)</td>
</tr>
<tr>
<td>Nurse_ID</td>
<td>Varchar</td>
<td>5</td>
<td></td>
<td></td>
<td>FK to Nurse; shows Nurse assigned</td>
<td>Y</td>
<td></td>
<td></td>
<td>Nurse(Nurse_ID)</td>
</tr>
</tbody>
</table>

### Table: Nurse

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Length</th>
<th>Min</th>
<th>Max</th>
<th>Description</th>
<th>Index</th>
<th>PK</th>
<th>FK</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nurse_ID</td>
<td>Varchar</td>
<td>5</td>
<td></td>
<td></td>
<td>Person ID</td>
<td>Y</td>
<td>Y</td>
<td></td>
<td>Employee(Emp_ID)</td>
</tr>
<tr>
<td>Cert_Degree</td>
<td>Varchar</td>
<td>5</td>
<td></td>
<td></td>
<td>Degree preparation</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>State_License_No</td>
<td>Varchar</td>
<td>15</td>
<td></td>
<td></td>
<td>License No.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nurse_Specialty</td>
<td>Varchar</td>
<td>20</td>
<td></td>
<td></td>
<td>Nurse medical specialty</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nurse_Type</td>
<td>Char</td>
<td>1</td>
<td>“L”</td>
<td>“R”</td>
<td>Subtype discriminator; L or R values</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table: Field_Certification

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Length</th>
<th>Min</th>
<th>Max</th>
<th>Description</th>
<th>Index</th>
<th>PK</th>
<th>FK</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>FC_ID</td>
<td>Varchar</td>
<td>5</td>
<td></td>
<td></td>
<td>Unique ID for Field certification instance</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FC_Description</td>
<td>Varchar</td>
<td>30</td>
<td></td>
<td></td>
<td>Description</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nurse_ID</td>
<td>Varchar</td>
<td>5</td>
<td></td>
<td></td>
<td>FK to Nurse</td>
<td>Y</td>
<td></td>
<td></td>
<td>Nurse(Nurse_ID)</td>
</tr>
</tbody>
</table>
### Table: Facility

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Length</th>
<th>Min</th>
<th>Max</th>
<th>Description</th>
<th>Index</th>
<th>PK</th>
<th>FK</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>Facility_ID</td>
<td>Varchar</td>
<td>10</td>
<td></td>
<td></td>
<td>Facility ID</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Facility_Name</td>
<td>Varchar</td>
<td>40</td>
<td></td>
<td></td>
<td>Name of Facility</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table: Diagnostic_Unit

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Length</th>
<th>Min</th>
<th>Max</th>
<th>Description</th>
<th>Index</th>
<th>PK</th>
<th>FK</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>DX_Unit_Name</td>
<td>Varchar</td>
<td>20</td>
<td></td>
<td></td>
<td>Unit Name</td>
<td>Y</td>
<td>Y</td>
<td></td>
<td>Work_Unit(Unit_Name)</td>
</tr>
</tbody>
</table>

### Table: Treatment

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Length</th>
<th>Min</th>
<th>Max</th>
<th>Description</th>
<th>Index</th>
<th>PK</th>
<th>FK</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trt_Code</td>
<td>Varchar</td>
<td>5</td>
<td></td>
<td></td>
<td>Treatment Code</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Treatment_Name</td>
<td>Varchar</td>
<td>30</td>
<td></td>
<td></td>
<td>Name of Treatment</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DX_Unit_Name</td>
<td>Varchar</td>
<td>20</td>
<td></td>
<td></td>
<td>ID of Unit performing treatment</td>
<td>Y</td>
<td></td>
<td></td>
<td>Diagnostic_Unit(DX_Unit_Name)</td>
</tr>
</tbody>
</table>

### Table: Treatment_Order

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Length</th>
<th>Min</th>
<th>Max</th>
<th>Description</th>
<th>Index</th>
<th>PK</th>
<th>FK</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>TO_ID</td>
<td>Varchar</td>
<td>5</td>
<td></td>
<td></td>
<td>Unique ID for treatment order instance</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Trt_Code</td>
<td>Varchar</td>
<td>5</td>
<td></td>
<td></td>
<td>Treatment Code FK</td>
<td>Y</td>
<td></td>
<td></td>
<td>Treatment(Trt_Code)</td>
</tr>
<tr>
<td>Results</td>
<td>Varchar</td>
<td>50</td>
<td></td>
<td></td>
<td>Results of treatment</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Trt_Date</td>
<td>Date</td>
<td></td>
<td></td>
<td></td>
<td>Date of Treatment</td>
<td>Y</td>
<td></td>
<td></td>
<td>Secondary</td>
</tr>
<tr>
<td>Trt_Time</td>
<td>Timestamp</td>
<td></td>
<td></td>
<td></td>
<td>Time of Treatment</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Order_ID</td>
<td>Varchar</td>
<td>5</td>
<td></td>
<td></td>
<td>FK to Order</td>
<td>Y</td>
<td></td>
<td></td>
<td>Order(Order_ID)</td>
</tr>
</tbody>
</table>
### Table: Order

<table>
<thead>
<tr>
<th>Data Item</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Type</td>
</tr>
<tr>
<td>Order_ID</td>
<td>Varchar</td>
</tr>
<tr>
<td>Patient_ID</td>
<td>Varchar</td>
</tr>
<tr>
<td>Physician_ID</td>
<td>Varchar</td>
</tr>
<tr>
<td>Item_No</td>
<td>Varchar</td>
</tr>
<tr>
<td>Order_Date</td>
<td>Date</td>
</tr>
<tr>
<td>Order_Time</td>
<td>Timestamp</td>
</tr>
</tbody>
</table>

### Table: Item

<table>
<thead>
<tr>
<th>Data Item</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Type</td>
</tr>
<tr>
<td>Item_No</td>
<td>Varchar</td>
</tr>
<tr>
<td>Item_Description</td>
<td>Varchar</td>
</tr>
<tr>
<td>Item_Unit_Cost</td>
<td>Number</td>
</tr>
</tbody>
</table>

### Table: Inventory

<table>
<thead>
<tr>
<th>Data Item</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Type</td>
</tr>
<tr>
<td>Inv_ID</td>
<td>Varchar</td>
</tr>
<tr>
<td>Item_No</td>
<td>Varchar</td>
</tr>
<tr>
<td>Vendor_ID</td>
<td>Varchar</td>
</tr>
</tbody>
</table>

### Table: Vendor

<table>
<thead>
<tr>
<th>Data Item</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Type</td>
</tr>
<tr>
<td>Vendor_ID</td>
<td>Varchar</td>
</tr>
<tr>
<td>Vendor_Name</td>
<td>Varchar</td>
</tr>
</tbody>
</table>
### Table: Item_Consumption

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Length</th>
<th>Min</th>
<th>Max</th>
<th>Description</th>
<th>Index</th>
<th>PK</th>
<th>FK</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>IC_ID</td>
<td>Varchar</td>
<td>5</td>
<td></td>
<td></td>
<td>Unique ID for consumption instance</td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Consume_Date</td>
<td>Date</td>
<td></td>
<td></td>
<td></td>
<td>Date of Consumption</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Consume_Time</td>
<td>Timestamp</td>
<td></td>
<td></td>
<td></td>
<td>Time of consumption</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Consume_Qty</td>
<td>Number</td>
<td>3</td>
<td>1</td>
<td></td>
<td>Quantity Consumed</td>
<td></td>
<td>Y</td>
<td></td>
<td>Secondary</td>
</tr>
<tr>
<td>Item_No</td>
<td>Varchar</td>
<td>5</td>
<td></td>
<td></td>
<td>Item Number FK</td>
<td></td>
<td></td>
<td></td>
<td>Item(Item_No)</td>
</tr>
<tr>
<td>Patient_ID</td>
<td>Varchar</td>
<td>5</td>
<td></td>
<td></td>
<td>Patient ID FK</td>
<td></td>
<td></td>
<td></td>
<td>Patient(Patient_ID)</td>
</tr>
</tbody>
</table>

### Table: Item_Billing

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Length</th>
<th>Min</th>
<th>Max</th>
<th>Description</th>
<th>Index</th>
<th>PK</th>
<th>FK</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>IB_ID</td>
<td>Varchar</td>
<td>5</td>
<td></td>
<td></td>
<td>Unique ID for billing instance</td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Start_Date</td>
<td>Date</td>
<td></td>
<td></td>
<td></td>
<td>Date of Consumption</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>End_Date</td>
<td>Date</td>
<td></td>
<td></td>
<td></td>
<td>Time of consumption</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cost</td>
<td>Number</td>
<td>9,2</td>
<td>.01</td>
<td>.01</td>
<td>Billing cost</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Item_No</td>
<td>Varchar</td>
<td>5</td>
<td></td>
<td></td>
<td>Item Number FK</td>
<td></td>
<td>Y</td>
<td></td>
<td>Item(Item_No)</td>
</tr>
<tr>
<td>Patient_ID</td>
<td>Varchar</td>
<td>5</td>
<td></td>
<td></td>
<td>Patient ID FK</td>
<td></td>
<td></td>
<td></td>
<td>Patient(Patient_ID)</td>
</tr>
<tr>
<td>Room_No</td>
<td>Varchar</td>
<td>5</td>
<td></td>
<td></td>
<td>Room_No FK</td>
<td></td>
<td></td>
<td></td>
<td>Room(Room_No)</td>
</tr>
</tbody>
</table>

### Table: Schedule

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Length</th>
<th>Min</th>
<th>Max</th>
<th>Description</th>
<th>Index</th>
<th>PK</th>
<th>FK</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schedule_ID</td>
<td>Varchar</td>
<td>5</td>
<td></td>
<td></td>
<td>Unique ID for schedule instance</td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Physician_ID</td>
<td>Varchar</td>
<td>5</td>
<td></td>
<td></td>
<td>Physician Person ID; FK</td>
<td></td>
<td></td>
<td></td>
<td>Physician(Physician_ID)</td>
</tr>
<tr>
<td>Sched_Begin</td>
<td>Date</td>
<td></td>
<td></td>
<td></td>
<td>Start date of schedule block</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sched_End</td>
<td>Date</td>
<td></td>
<td></td>
<td></td>
<td>End date of schedule block</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Facility_ID</td>
<td>Varchar</td>
<td>10</td>
<td></td>
<td></td>
<td>Facility where Physician is assigned to work the schedule block; FK</td>
<td></td>
<td>Y</td>
<td></td>
<td>Facility(Facility_ID)</td>
</tr>
</tbody>
</table>
### Table: Assessment

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Length</th>
<th>Min</th>
<th>Max</th>
<th>Description</th>
<th>Index</th>
<th>PK</th>
<th>FK</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assessment_ID</td>
<td>Varchar</td>
<td>5</td>
<td></td>
<td></td>
<td>Unique ID for patient’s assessment instance</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Patient_ID</td>
<td>Varchar</td>
<td>5</td>
<td></td>
<td></td>
<td>Patient Person ID</td>
<td></td>
<td>Y</td>
<td></td>
<td>Patient(Patient_ID)</td>
</tr>
<tr>
<td>Assessment_Date</td>
<td>Date</td>
<td></td>
<td></td>
<td></td>
<td>Date of Assessment</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Assessment_Time</td>
<td>Timestamp</td>
<td></td>
<td></td>
<td></td>
<td>Time of Assessment</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Comments</td>
<td>Varchar</td>
<td>50</td>
<td></td>
<td></td>
<td>Comments for visit</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Patient_Weight</td>
<td>Number</td>
<td>3</td>
<td>1</td>
<td></td>
<td>Weight</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Patient_BP</td>
<td>Varchar</td>
<td>7</td>
<td></td>
<td></td>
<td>Blood pressure reading</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Patient_pulse</td>
<td>Number</td>
<td>4</td>
<td></td>
<td></td>
<td>Pulse reading</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Patient_temperature</td>
<td>Number</td>
<td>(3,2)</td>
<td></td>
<td></td>
<td>Temperature reading</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nurse_ID</td>
<td>Varchar</td>
<td>5</td>
<td></td>
<td></td>
<td>Nurse who provided patient assessment; FK</td>
<td>Y</td>
<td></td>
<td></td>
<td>Nurse(Nurse_ID)</td>
</tr>
</tbody>
</table>
P3. **Note to instructor:** Selected examples of elements of the physical data model are shown in the following two images, to illustrate interesting data types, choice of primary key, etc. A full solution was not provided in order to save space in the 9e Instructor’s Manual, due to the complexity of the full data model now necessary to depict the rich case information.
P4.

a. Nurses assigned to each care_center
Assume this report is run 7 times per day, there are 7 Care Centers and 250 Nurses.

(PA4a) Nurses assigned to each Care Center
Data Volume Analysis Diagram

b. 5 most common diagnoses
Again, assume 7 runs per day, 50 different diagnoses and 1,000,000 diagnostic records.

(PA4b) Five most common diagnoses
Data Volume Analysis Diagram
c. Items consumed by each patient; assume run 10 times a day.

(PA4c) Items consumed by each patient
Data Volume Analysis Diagram
d. Number of items provided by vendors; run 10 times per day; assume 500 vendors.

(PA4d) Number of items provided by vendors
Data Volume Analysis Diagram
e. Number of patient admissions per physician; run 10 times per day

(Phase 4e) Number of patient admissions per physician
Data Volume Analysis Diagram

```
PHYSICIAN
250

PATIENT
20,000
```
Chapter 7  Introduction to SQL

Chapter Overview

This chapter describes in detail what has become the standard query language for relational database management systems, SQL. Although SQL is illustrated in this chapter primarily through the Oracle10g SQL*Plus version (MySQL and Microsoft Access SQL are also shown), SQL is a portable language that is available on all classes of computers and with many different DBMSs.

This chapter introduces SQL, DDL, and DML. Single table queries are covered in this chapter, and multiple-table queries are covered in Chapter 8.

Several chapters contain important prerequisite material for this chapter. Chapter 5 introduces the relational model and provides much of the background for this chapter. The discussion in Chapter 6 on indexes is also important because choosing primary and secondary key indexes is one of the few, but crucial, internal database design choices for users of relational systems.

Chapter Objectives

Specific student learning objectives are included at the beginning of the chapter. From an instructor’s point of view, the objectives of this chapter are to:

1. Explain SQL and show the basic operators so that the student can anticipate the capabilities of particular SQL-based systems.
2. Provide a historical perspective of the development of SQL and its continuing development. This perspective illustrates the benefits and risks of adopting a standard query language.
3. Show that SQL, although a standard and a high-level language, does have some flaws, and that SQL must evolve to include additional features.
4. Explain and illustrate the power of relational views for simplifying relational database processing.
5. Illustrate data definition language (DDL) commands for creating tables and views as well as modifying and dropping tables.
6. Provide examples of single table SQL queries.
7. Provide some examples of the use of functions within SQL queries.
8. Show how to establish referential integrity using SQL.
9. Illustrate the use of the group by and order by clauses.

Key Terms

<table>
<thead>
<tr>
<th>Base table</th>
<th>Data manipulation language (DML)</th>
<th>Relational DBMS (RDBMS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Catalog</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data control language (DCL)</td>
<td>Dynamic view</td>
<td>Scalar aggregate</td>
</tr>
<tr>
<td></td>
<td>Materialized view</td>
<td>Schema</td>
</tr>
<tr>
<td>Data definition language (DDL)</td>
<td>Referential integrity</td>
<td>Vector aggregate</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Virtual table</td>
</tr>
</tbody>
</table>
Classroom Ideas

1. Depending on how you choose to sequence the chapters of this book, you may want to begin lecturing on this chapter by reviewing normalization principles and discussing why one might want to implement a fully normalized database. See the section “Denormalization” in Chapter 6 for background on this topic. The point is to emphasize the difference between logical and physical database designs.

2. The capabilities and benefits of views are not as obvious to students as you might expect. First, many relational systems place limitations on operations that may be performed on views (for example, no or restricted modification). Second, views are not just the relational term for temporary table or for subschemas (from the network model); rather, views provide a capability for simpler query writing for casual users. Finally, views may have negative performance implications compared to temporary, intermediate tables explicitly created by a prewritten procedure. These points should be emphasized in lectures or assignments. Consider having students create both a view and a temporary table (or materialized view) in SQL. In most SQL systems, a temporary table can be created by the use of a SAVE or KEEP clause at the end of a query, or one can use a CREATE TABLE command that has a query (rather than column definitions) as its object. Then, the same query can be run and timed against both the view and temporary tables. As another alternative, have students derive the view table as part of the FROM clause of a query. One way to estimate the timing of each approach, given small data sets, is to use the EXPLAIN command (or similar command with your RDBMS) on each version of the query to show students how the query would be processed using each approach.

3. The simplicity of SQL syntax leads some students to become overconfident of their SQL prowess. It may help to develop some examples to demonstrate that logically and syntactically correct SQL queries can be constructed that do not accomplish the intended data manipulation and may be dangerously incorrect. For example, two tables may be joined using common domains (for instance, positive integers) but noncommon roles (for example, two tables can be joined on equal values of quantity-on-hand of a product and number of employees in a department). Such potential problems arise because the relational data model uses implicit relationships, not explicit ones. Stress the importance of testing a query on a small, tractable set of data before using it in a production environment.

4. With the variety of SQL implementations now available, we strongly recommend that you illustrate some SQL system in class and have the students use such a system. The Pine Valley Furniture and Mountain View Community Hospital cases in the text provide rich contexts in which to develop examples that emphasize those aspects of SQL the instructor considers most important. We have found that students learn SQL best by seeing and doing many examples and experimentation, so we encourage teaching SQL by example. When we teach SQL, we often spend about 15 minutes giving an overview of the evolution of SQL and providing some
basic syntax and terminology. Then we spend between two and six hours in live
demonstrations showing simple to complex queries. In a computer-enabled
classroom where an SQL system can be demonstrated to and used by students, this
time can be an effective, interactive learning experience.

All the SQL queries in this chapter can be found in a Microsoft Access database
available from the publisher; you can show in class how Access SQL varies from
standard SQL, and you can show variations on the queries from this chapter. In
addition, a set of 40 lab exercise queries is available on
www.teradatauniversitynetwork.com for use with the Pine Valley Furniture
Company case. Many of these lab exercises are included already in this text. These
queries are designed to work with a database that is close to, but not exactly, the data
model in Figure 3-22.

5. It can be helpful to use one case study for homework assignments. The students
become very familiar with the tables used in the case study and they seem to learn
the SQL syntax much quicker when this is done by the instructor.

6. Because many SQL systems include other modules (report writers, business graphics
routines, screen painters, and so forth), this chapter can be expanded to discuss
fourth-generation languages and systems prototyping. Explain that most packages
include capabilities and development tools that are more than simply a relational
DBMS and query language. If your school has joined the Oracle Academic
Initiative, it will be possible to demonstrate additional tools such as Designer/2000.

7. It should be emphasized that SQL is still fundamentally a programming language for
the professional programmer. That is, SQL is not really a suitable language for end-
user system development. This does not diminish its importance because it achieves
tremendous productivity gains for a professional programming staff and is used
extensively by database administrators. Many of your students are likely to hold
these positions during their careers. Other user interfaces that might permit access to
a relational database (like QBE) are more appropriate for end users.

Answers to Review Questions

1. Define each of the following key terms:
   a. Base table A table in the relational data model containing inserted raw data that
      is likely to correspond to one physical file in secondary storage. (The base table
      is also referred to as one of the objects—such as the base tables, views,
      constraints, and so on.—which have been defined for a particular database by a
      particular user, who owns the objects in the schema.)
   b. Data definition language Those commands used to define a database, including
      creating, altering, and dropping tables and establishing constraints.
   c. Data manipulation language Those commands used to maintain and query a
      database, including updating, inserting, modifying, and querying data.
d. **Dynamic view** A virtual table that is created dynamically upon request by a user. Not a temporary table, its definition is stored in the system catalog. The contents of the view are materialized as a result of an SQL query that uses the view.

e. **Materialized view** Copies or replicas of data based on SQL queries, in the same manner that dynamic views are created. However, a materialized view exists as a table and care must be taken to keep it synchronized with its associated base tables.

f. **Referential integrity** An integrity constraint specifying that the value or existence of an attribute in one relation depends on the value or existence of a primary key in the same or another relation. Referential integrity means that a value in the matching column on the many-side must correspond to a value in the primary key for some row in the table on the one-side, or be null.

g. **Relational DBMS (RDBMS)** A database management system that manages data as a collection of tables in which all data relationships are represented by common values (not links) in related tables.

h. **Schema** A structure that contains descriptions of objects created by a user, such as base tables, views, constraints, and so on, as part of a database.

i. **Virtual table** A table constructed automatically as needed by a DBMS. Virtual tables are not maintained as real data. Sometimes a virtual table is referred to as a dynamic view.

2. Match the following terms to the appropriate definitions:
   
   d. view
   h. referential integrity
   c. dynamic view
   i. materialized view
   j. SQL-200n
   e. null value
   k. scalar aggregate
   a. vector aggregate
   b. catalog
   f. schema
   g. host language

3. Contrast the following terms:
   
a. **Base table; view** A view is a virtual table and is often part of an external database. In contrast to a defined base table (relation), a view is not permanently represented in storage. A view definition is stored, and the contents of the view are calculated each time the view is referenced in a query. A view may join multiple tables or views together and may contain derived (or virtual) columns, while base tables cannot. In comparison to a temporary real table, a view consumes very little storage space. A view is costly (compared to a dynamic, materialized view), however, because its contents must be calculated each time that it is requested.

b. **Dynamic view; materialized view** Dynamic views are not a temporary table but are materialized from an SQL query that uses the view definition that is stored in
the system catalog. Materialized views exist as a table and, thus, must be kept synchronized with their associated base tables.

c. **Catalog; schema** Schema is a structure that contains descriptions of objects created by a user, such as base tables, views, constraints, and so on. These objects have been defined for a particular database by a particular user, who owns those objects in the schema. The catalog is a set of schemas, which, when put together, constitute a description of a database. (If more than one user has created objects in the database, combining information about all their schemas will yield information for the entire database.)

4. History leading up to SQL-200n:

a. To provide direction for the development of RDBMSs, the American National Standards Institute (ANSI) and the International Organization for Standardization (IOS) approved a standard for the SQL relational query language functions and syntax proposed originally by the X3H2 Technical Committee on Database (Technical Committee X3H2 Database, 1986; ISO, 1987). This standard is often referred to as SQL/86. The 1986 standards have been extended to include an optional Integrity Enhancement Feature (IEF), often referred to as SQL/89. The ISO and ANSI committees created SQL-92 (Technical Committee X3H2 Database, 1989; ISO, 1989, 1991), which was a more extensive expansion of SQL/86. This standard was ratified in late 1992, and is known as International Standard ISO/IEC 9075:1992, *Database Language SQL*.

b. SQL-99 is a significant extension beyond SQL-92. SQL-99 established Core-level conformance, which must be met before any other level of conformance can be achieved. Eight additional types of enhanced conformance have been specified so far, including Active Database, Enhanced Integrity Management, Enhanced Datetime Facilities, Basic Object Support, Enhanced Object Support, Active Database, OLAP Facilities, Persistent Stored Modules, and Call-Level Interface.

c. SQL200n provides many enhancements to SQL-99 including predefined data types, type constructors, scalar, and table expressions. Also included as part of this standard is SQL/XML.

5. Describe a relational DBMS (RDBMS), its underlying data model, data storage structures, and manner of establishing data relationships:

a. A relational DBMS (or RDMBS) is a data management system that implements a relational data model.

b. Data are stored in a collection of tables, and the data relationships are represented by common values, not links.

c. The relational data model assumes that a table is a logical construct rather than a physical construct, so a table need not correspond to a physical file of contiguous records. The same data model may have many different possible physical implementation structures. The storage of the database is dependent on both the hardware and software environment. It is usually the concern of the system administrator.
d. The power of the RDBMS is realized through relationships existing between tables. These relationships are established by including a common column(s) in each table where a relationship is needed.

6. Six potential benefits of achieving an SQL standard:
   a. *Reduced training costs* Training in an organization can concentrate on SQL, and a large labor pool of IS professionals trained in a common language reduces retraining when hiring new employees.
   b. *Productivity* IS professionals can learn SQL thoroughly and become proficient with it from continued use, the organization can afford to invest in tools to help IS professionals become more productive, and programmers can more quickly maintain existing programs because they are familiar with the language in which programs are written.
   c. *Application portability* Applications can be moved from machine to machine when each machine uses SQL. Further, it is economical for the computer software industry to develop off-the-shelf application software when there is a standard language.
   d. *Application longevity* A standard language tends to remain so for a long time, so there will be little pressure to rewrite old applications. Rather, applications will simply be updated as the standard language is enhanced or new versions of DBMSs are introduced.
   e. *Reduced dependence on a single vendor* When a nonproprietary language is used, it is easier to use different vendors for the DBMS, training and educational services, application software, and consulting assistance. Further, the market for such vendors will be more competitive, which may lower prices and improve service.
   f. *Cross-system communication* Different DBMSs and application programs can more easily communicate and cooperate in managing data and processing user programs.

7. The components and structure of a typical SQL environment:
The SQL environment includes an instance of an SQL DBMS along with accessible databases and associated users and programs. Each database is included in a catalog and has a schema that describes the database objects. Information contained in the catalog is maintained by the DBMS itself, rather than by the users of the DBMS.

8. Distinguish among data definition commands, data manipulation commands, and data control commands:
   a. The data definition language (DDL) commands of SQL are used to define a database, including its creation and the creation of its tables, indexes, and views. Referential integrity is also established through DDL commands. CREATE/DROP DICTIONARY, CREATE/DROP TABLE, ALTER TABLE, CREATE/DROP INDEX, CREATE/DROP VIEW are examples of DDL commands.
b. The data manipulation (DML) commands of SQL are used to load, update, and query the database through the use of the SELECT command. (START TRANSACTION, COMMIT WORK, ROLLBACK WORK, INSERT, UPDATE, and DELETE are examples of DML commands.)

c. Data control language (DCL) commands are used to establish user access to the database through the GRANT, ADD USER, and REVOKE commands.

9. Establishing referential integrity using an SQL:1999 compliant database and the differences among the ON UPDATE RESTRICT, ON UPDATE CASCADE, and ON UPDATE SET NULL clauses, and results of declaring ON DELETE CASCADE:

a. The SQL REFERENCES clause is used to establish referential integrity and prevents a foreign key value from being added if it is not already a valid value in the referenced primary key column.

b. The ON UPDATE RESTRICT clause allows a row to be deleted in a parent table only if no record in the child table references the primary key value to be deleted in the parent table. The ON UPDATE CASCADE clause will cause a change in a primary key value of a parent table to be passed through and update the foreign key value in the related child table. The ON UPDATE SET NULL option allows the update on the parent table, but changes the involved foreign key value in the child table to NULL. Using the SET NULL option would result in losing the connection between the parent and child tables, not a desired effect.

c. With DELETE CASCADE, removing the primary key value in the parent table also removes all associated records from the child table.

10. Reasons to create a view using SQL and how a view can be used to reinforce data security:

a. Views may simplify query commands, provide valuable data security, and enhance programming productivity for a database.

b. Tables and columns that are not included in a view will not be obvious to the user of the view. Restricting access to a view with GRANT and REVOKE statements (security statements) adds a further layer of security. It should not be regarded as the primary security layer.

11. Update limitations on data changes when referencing data through a view:

   In general, update operations to data in a view are permitted as long as the update is unambiguous in terms of data modification in the base table. However, when the CREATE VIEW statement contains any of the following situations, that view may not be updated directly.

a. The SELECT clause includes the keyword DISTINCT.

b. The SELECT clause contains expressions, including derived columns, aggregates, statistical functions, and so forth.

c. The FROM clause, a subquery, or a UNION clause references more than one table.

d. The FROM clause or a subquery references another view, which is not updateable.
12. Saving reprogramming effort by using views:
   When more than one program uses a view on the same base table, changes in all
   the applications that relate to this particular base table will require only re-
   creation of the view (outside the applications themselves). Views require
   considerable run-time computer processing because the virtual table of a view is
   recreated each time the view is referenced. Therefore, referencing a base table
   through a view rather than directly, can add considerable time to query
   processing, depending on the query. This additional operational cost must be
   balanced against the potential reprogramming savings from a view. A view also
   simplifies query writing because queries written against the view refer to only one
   (virtual) table, rather than several base tables.

13. Factors to be considered in deciding whether to create a key index for a table in SQL:
   a. A key index on one column or a concatenation of columns enables rapid access
      to the rows of a table in a sequence or randomly by key value.
   b. Choosing to index primary and/or secondary keys may increase the speed of row
      selection, table joining, and row ordering.
   c. Dropping indexes will increase the speed of table updating.

14. Qualifying the ownership of a table in SQL:
   Placing the ID of the owner prior to the table name and attribute name indicates
   ownership of a table. For example, if the user with the User_ID of ORTEGA
   owns the CUSTOMER_T table, that ownership would be indicated as
   ORTEGA.CUSTOMER_T.

15. Changing attribute order and column heading labels in a result table:
   a. Attributes are displayed in columns in the sequence in which they are listed in the
      SELECT list unless the SELECT list is *, in which case all attributes from the
      referenced tables are displayed in the sequence in which they are defined.
   b. Use AS to specify column heading labels, e.g., SELECT CUST.
      CUSTOMER_NAME AS NAME will result in a column heading label NAME
      instead of CUSTOMER_NAME.

16. COUNT, COUNT DISTINCT, and COUNT(*) in SQL and the results generated
    when using these:
   a. COUNT tallies only those rows that contain a value; it ignores all null values.
   b. COUNT DISTINCT does not return a count on all values; it only tallies once if
      more than one row has equal values.
   c. COUNT (*) counts all rows regardless of whether any of the rows contain null
      values.
   d. If we never use NULL values, COUNT and COUNT(*) will return the same
      results. If our table includes no duplicates in the considered attribute values,
      COUNT and COUNT DISTINCT will have the same meaning.
17. Evaluation order for the Boolean operators (AND, OR, NOT) in an SQL command; getting the operators to work in the order that you want:
   a. If multiple Boolean operators are used in an SQL statement, NOT is evaluated first, then AND, then OR.
   b. With the use of parentheses around statements using standard mathematical notation, a set of statements may be given a user-chosen specific order of evaluation.

18. Limitations on attributes that can be selected when an SQL statement contains GROUP BY:
   Only those columns that have a single value for each group can be included.

19. The HAVING clause is useful when you need to select results based upon qualifications on an aggregation, such as a characteristic of a group. For example, if we have a table with transactional sales data for salespeople, we may want to sum up the total sales for a given period of time (an aggregate). We may then want to produce a list of only salespeople who sold more than a certain amount, say $1,000. This can be done using the HAVING clause.

20. The OR operator can be used to perform the same operation as the IN operator. For example, if part of our Where clause is State in ('MA', 'NH'), this could be substituted with State = 'MA' OR State = 'NH'.

21. Yes, according to www.wisecorp.com/SQL2003Features.pdf, products conforming to Core SQL:1999 are likely to conform to Core SQL:200n.

22. CREATE TABLE LIKE is a quick way to clone a table.

23. The identity column creates a numeric sequence. For example, if we create the following table:

   ```sql
   CREATE TABLE test1
   (id smallint identity(5,2),...)
   ```

   Then the first record added to test1 will have 5 as its id value. Subsequent record ids will be incremented by 2.

   This is beneficial because it allows a way to automatically generate a number, such as an ID.
24. Prior to SQL:200n, one had to use both *insert* and *update* to update and merge data into a table. The *Merge* command allows one to do this in a single step. A common use for this would be updating master table from a transaction table where one might want to update records or create new records in the master table based upon what was done in the transaction table. Let’s take a look at an example: Suppose that we have an item table and a shipment table. The item table contains all of the items and the shipment table contains items received by the company.

<table>
<thead>
<tr>
<th>Item Table</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Item_No</strong></td>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>342</td>
<td>Apple  PowerBook</td>
</tr>
<tr>
<td>345</td>
<td>IBM PC</td>
</tr>
<tr>
<td>346</td>
<td>256mb Memory</td>
</tr>
<tr>
<td>347</td>
<td>80gb HD</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Shipment Table</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Item_No</strong></td>
</tr>
<tr>
<td>342</td>
</tr>
<tr>
<td>345</td>
</tr>
<tr>
<td>347</td>
</tr>
<tr>
<td>352</td>
</tr>
</tbody>
</table>

Notice that there is one additional item in the Shipment table. Now, let’s write the merge statement to update the Item table. If the record exists in *item*, then we will add to the quantity on hand. If it does not, then we will add the record.

```
MERGE INTO  ITEM AS ITM
USING (SELECT ITEM_NO, DESCRIPTION, QUANTITY_RECEIVED FROM SHIPMENT)
AS SH
ON (ITM.ITEM_NO = SH.ITEM_NO)
WHEN MATCHED THEN UPDATE
SET QUANTITY_ON_HAND = ITM.QUANTITY_ON_HAND +
    SH.QUANTITY_RECEIVED
WHEN NOT MATCHED THEN INSERT
(ITEM_NO, DESCRIPTION, QUANTITY_ON_HAND)
VALUES (SH.ITEM_NO, SH.DESCRIPTION, SH.QUANTITY_RECEIVED):
```
Answers to Problems and Exercises

Note: The solutions that include SQL statements are not intended as the definitive answer to the questions, but as possible solutions. Instructors and students will approach the problems using different SQL capabilities, achieving results that are also correct.

1. Database descriptions:
   Note: A particular SQL system may restrict the length of column names or may not permit embedded spaces in names. The following answer may not be acceptable to the SQL system you use, but you may modify the grammar accordingly.

   CREATE TABLE STUDENT
   (STUDENT_ID  NUMBER   NOT NULL,
   STUDENT_NAME  VARCHAR2(25),
   CONSTRAINT STUDENT_PK PRIMARY KEY (STUDENT_ID));

   CREATE TABLE FACULTY
   (FACULTY_ID  NUMBER   NOT NULL,
   FACULTY_NAME  VARCHAR2(25),
   CONSTRAINT FACULTY_PK PRIMARY KEY (FACULTY_ID));

   CREATE TABLE COURSE
   (COURSE_ID   CHAR(8)  NOT NULL,
   COURSE_NAME  VARCHAR2(15),
   CONSTRAINT COURSE_PK PRIMARY KEY (COURSE_ID));

   CREATE TABLE SECTION
   (SECTION_NO  NUMBER   NOT NULL,
   SEMESTER    CHAR(7)   NOT NULL,
   COURSE_ID   CHAR(8),
   CONSTRAINT SECTION_PK PRIMARY KEY(COURSE_ID,SECTION_NO, SEMESTER),
   CONSTRAINT SECTION_FK FOREIGN KEY (COURSE_ID) REFERENCES COURSE (COURSE_ID));

   CREATE TABLE QUALIFIED
   (FACULTY_ID  NUMBER   NOT NULL ,
   COURSE_ID   CHAR(8)   NOT NULL,
   DATE_QUALIFIED        DATE,
   CONSTRAINT IS_QUALIFIED_PK PRIMARY KEY (FACULTY_ID, COURSE_ID),
   CONSTRAINT QUALIFIED_FACULTY_FK FOREIGN KEY (FACULTY_ID) REFERENCES FACULTY (FACULTY_ID),
   CONSTRAINT QUALIFIED_COURSE_FK FOREIGN KEY (COURSE_ID) REFERENCES COURSE (COURSE_ID));

Published by www.emagzine.org
CREATE TABLE REGISTRATION
(STUDENT_ID NUMBER NOT NULL,
SECTION_NO NUMBER NOT NULL,
SEMESTER CHAR(7) NOT NULL,
CONSTRAINT IS_REGISTERED_PK PRIMARY KEY (STUDENT_ID,
SECTION_NO, SEMESTER),
CONSTRAINT STUDENT_IS_REGISTERED_FK
FOREIGN KEY(STUDENT_ID)
REFERENCES STUDENT(STUDENT_ID),
CONSTRAINT COURSE_IS_REGISTERED_FK
FOREIGN KEY (SECTION_NO, SEMESTER)
REFERENCES SECTION(SECTION_ID, SEMESTER));

2. CREATE VIEW STUDENT_V AS
SELECT STUDENT_ID, STUDENT_NAME FROM STUDENT;

3. See the SECTION_FK constraint in the DDL for the SECTION table in Problem and Exercise 1 above.

4. a. ALTER TABLE STUDENT
   ADD CLASS VARCHAR2(5);
b. DROP TABLE REGISTRATION;
c. ALTER TABLE FACULTY
   MODIFY FACULTY_NAME VARCHAR2(40);

Note: CHANGE instead of MODIFY also usually works.

5. a. INSERT INTO STUDENT (STUDENT_ID, STUDENT_NAME)
    VALUES (65798,'Lopez');
    INSERT INTO STUDENT VALUES (65798,'Lopez');
b. DELETE FROM STUDENT WHERE STUDENT_ID = 65798;
c. UPDATE COURSE
   SET COURSE_NAME = 'Introduction to Relational Databases'
   WHERE COURSE_ID = 'ISM 4212';

6. a. SELECT STUDENT_ID, STUDENT_NAME
    FROM STUDENT
    WHERE STUDENT_ID < 50000;
b. SELECT FACULTY_NAME
    FROM FACULTY
    WHERE FACULTY_ID = 4756;
c. SELECT MIN(SECTION_ID)
   FROM REGISTRATION
   WHERE SEMESTER = 'I-2008';

7.
   a. SELECT COUNT(*)
      FROM REGISTRATION
      WHERE SECTION = 2714
      AND SEMESTER = 'I-2008';
   b. SELECT FACULTY_ID,COURSE_ID,DATE_QUALIFIED
      FROM QUALIFIED
      WHERE DATE_QUALIFIED >= '01-JAN-1993';

8.  
   a. We assume all the question wishes in the result set are the student IDs. The Database course is ISM 4212, which is section 2714 in the REGISTRATION table, and the Networking course is ISM 4930, which is section 2715 in the REGISTRATION table:
      SELECT STUDENT_ID,COUNT(*)
      FROM REGISTRATION
      WHERE SECTION_NO IN (2714,2715)
      GROUP BY STUDENT_ID
      HAVING COUNT(*) > 1;
   b. In answering this question, we assume we are not interested in seeing those instructors who can teach neither course, but rather only those who can teach one but not the other course. To find those instructors who cannot teach either course requires SQL capabilities introduced in Chapter 8:
      SELECT INSTRUCTOR_ID,COUNT(*)
      FROM QUALIFIED
      WHERE COURSE_ID IN ('ISM 3113','ISM 3112')
      GROUP BY INSTRUCTOR_ID
      HAVING COUNT(*) = 1;

9.  
   a. SELECT DISTINCT COURSE_ID
      FROM SECTION;
   b. SELECT STUDENT_NAME
      FROM STUDENT
      ORDER BY STUDENT_NAME;
   c. SELECT SECTION_NO,SEMESTER,STUDENT_ID
      FROM REGISTRATION
      WHERE SEMESTER = 'I-2008'
      ORDER BY SECTION_NO,SEMESTER,STUDENT_ID;
d.  SELECT COURSE_ID, COURSE_NAME
    FROM COURSE
    ORDER BY COURSE_ID;

10.  a. Query:
    SELECT COUNT(TUTOR_ID) AS NUMTSTOP
    FROM TUTOR
    WHERE STATUS = 'Temp Stop';

    b. Query:
    SELECT TUTOR_ID
    FROM TUTOR
    WHERE STATUS = 'Active';

11.  Query:
    SELECT TUTOR_ID
    FROM TUTOR
    WHERE CERT_DATE BETWEEN '01-JAN-2008' AND '31-JAN-2008';

12.  Query:
    SELECT COUNT(STUDENT_ID)
    FROM MATCH_HISTORY
    WHERE START_DATE BETWEEN '01-JAN-2008' AND '31-MAY-2008';

13.  Query:
    SELECT READ, STUDENT_ID
    FROM STUDENT
    ORDER BY READ DESC;
For #14 and #15, be sure the student realizes that a null value in END_DATE means that a student is still studying in the program and should be included in the calculation. This is a good opportunity for the student to discover that a calculated field can be quite useful where null values can be expected. You may want to emphasize that this query will not update the base table and these null values will remain until a student leaves the program.

14. This answer is MS ACCESS SQL and uses the NZ function, which is not covered in the text, to deal with the null dates. If no date is entered as an end date in the dialog box, today’s date is inserted automatically for null values. As the NZ function is used here, it will return the number of days the student has studied into the calculated field.

**Query:**

```sql
SELECT STUDENT_ID, START_DATE, END_DATE,
    DateDiff('d',[START_DATE],NZ([END_DATE], NZ
        [Enter date for blank records], Date()))) AS TOTAL_DAYS
FROM MATCH_HISTORY;
```

15. This query needs to use the TOTAL_DAYS information calculated in #14, but as the average function aggregates all rows, a two-step query will be needed to complete the problem. This problem also demonstrates using a query as the base for another query rather than working from a base table.

a. **Query:** (Answer assumes this query is named CH7P15_First Query.)

```sql
SELECT DateDiff('d',[START_DATE],NZ([END_DATE], NZ
    [Enter date for blank records], Date()))) AS TOTAL_DAYS
FROM MATCH_HISTORY;
```

b. **Query:**

```sql
SELECT AVG(TOTAL_DAYS) AS [Average time]
FROM CH7P15_FirstQuery;
```
Note to instructor: Problems and Exercises 16-37 are based on the extended version of the Pine Valley Furniture Company database. (BigPVFC.mdb is the MS Access file version of this database; this is also available on Teradata student resources.) Please note that this version of the database has a different structure than that in the textbook version of the database (e.g., the salesperson information is in the extended version but not in the textbook version). Some of the field names may also have changed due to the version of the database you are using due to the reserved words of the DBMS. When you first use the database, check the table definitions to see what the exact field names and table structures are for the DBMS that you are using.

16. ALTER TABLE PRODUCT_T
    ADD QTY_ON_HAND NUMBER(5) CHECK (QTY_ON_HAND >=0);

17. Students should answer this exercise using a series of UPDATE commands to SET values for this new field in each existing row of the PRODUCT_T table. If using Oracle 10g and the CHECK option, students should receive error messages if they attempt to enter negative values for QTY_ON_HAND values.

18. Students should answer this exercise using INSERT commands to enter new rows into the ORDER_t and ORDER_LINE_t tables.

19. Although the data in this table is sparsely populated and the answers to these questions can be found by simple inspection of the table values, encourage the students to write SQL to derive the results:

   a. SELECT COUNT(*)
      FROM WORK_CENTER_T;

   b. SELECT WORK_CENTER_LOCATION
      FROM WORK_CENTER_T;

20. Display the product line ID and the average standard price for all products in each product line:

    Query:
    SELECT Product_Line_Id, AVG(Standard_Price)
    FROM PRODUCT_t
    GROUP BY Product_Line_Id;
21. For every product that has been ordered, display the product ID and the total quantity ordered (label this result Total_Ordered). List the most popular product first and the least popular last:

**Query:**
```sql
SELECT Product_Id, SUM(Ordered_Quantity) AS Total_Ordered
FROM Order_line_t
GROUP BY Product_Id
ORDER BY SUM(Ordered_Quantity) DESC;
```

22. Display the product ID and the number of orders placed for each product. Show the results in decreasing order:

**Query:**
```sql
SELECT Product_Id, COUNT(Product_ID) AS NumOrders
FROM Order_line_t
GROUP BY Product_Id
ORDER BY COUNT(Product_ID) DESC;
```

23. Note: Because subqueries have not been covered yet, the student should just order the query by the total orders and report on the top product_id (or product_ids if more than one):

**Query:**
```sql
SELECT product_id, count(*) AS NumOrders
FROM order_line_t
GROUP BY product_id
ORDER by count(*) DESC;
```

An alternate approach in SQL-Server or MS-Access is to use the Top predicate:

**Query:**
```sql
SELECT TOP 1 product_id, count(*) AS NumOrders
FROM order_line_t
GROUP BY product_id;
```

24. Employees whose last name begins with “L”:

**Query:**
```sql
SELECT EMPLOYEE_ID,EMPLOYEE_NAME
FROM EMPLOYEE_T
WHERE EMPLOYEE_NAME LIKE 'L%';
```
25. Employees hired during 1999:

**Query:**

```sql
SELECT EMPLOYEE_NAME
FROM EMPLOYEE_T
WHERE EMPLOYEE_DATE_HIRED BETWEEN '01-JAN-1999'
     AND '31-DEC-1999';
```

26. Customers who live in California or Washington, ordered by descending zip code:

**Query:**

```sql
SELECT CUSTOMER_ID,CUSTOMER_NAME
FROM CUSTOMER_T
WHERE CUSTOMER_STATE IN ('WA','CA')
ORDER BY POSTAL_CODE
```

27. Customer ID and total number of orders placed:

**Query:**

```sql
SELECT CUSTOMER_ID,COUNT(ORDER_ID) AS TOT_ORDERS
FROM ORDER_T
GROUP BY CUSTOMER_ID
```

28. Customer ID and total orders placed in 2008:

In MS-Access, the query would be written as follows:

```sql
SELECT order_t.Customer_Id, Count(order_t.Order_Id) AS Totorders
FROM order_t
WHERE (((order_t.Order_Date) Between #1/1/2008# And #12/31/2008#))
GROUP BY order_t.Customer_Id;
```

In Oracle 10g, the query would be written as follows:

```sql
SELECT Customer_Id, Count(Order_Id) AS Totorders
FROM order_t
WHERE Order_Date Between '01-Jan-2008' and '31-Dec-2008'
GROUP BY Customer_Id;
```

29. Customer ID and total orders placed if more than two orders were placed:

**Query:**

```sql
SELECT Customer_Id, COUNT(Order_ID) as Totorders
FROM order_t
GROUP BY Customer_Id
HAVING Count(Order_ID) > 2;
```
30. All sales territories that have more than one salesman: (Note: We also show the number of salespeople for each territory.)

Query:
```
SELECT territory_id, COUNT(salesperson_name) AS NumSalesPeople
FROM salesperson_t
GROUP BY territory_id
HAVING COUNT(salesperson_name) > 1;
```

31. All raw materials made of cherry and 12 x 12 dimensions: (Note: The query shown here is an MS-Access query. If using Oracle, simply replace the double quotes with single quotes.)

Query:
```
SELECT material_id, material_name
FROM raw_material_t
WHERE material = "Cherry" AND thickness ="12" AND width = "12";
```

32. Material ID, name, material, standard price, and thickness for raw materials made of cherry, pine, or walnut. Order list by material, standard price, and thickness: (Note: The query shown here is an MS-Access query. If using Oracle, simply replace the double quotes with single quotes.)

Query:
```
SELECT material_id,material_name,material,standard_price, thickness
FROM  raw_material_t
WHERE material IN ("Cherry","Pine","Walnut")
ORDER BY material,standard_price,thickness;
```

33. Total number of orders for each salesman:

Query:
```
SELECT salesperson_id,count(order_id) AS Total_Orders
FROM order_t
GROUP BY salesperson_id;
```

34. Each salesperson’s number of orders placed for each month of 2008. The solution is shown here in both MS-Access and Oracle:

**MS-Access**
```
SELECT salesperson_id,month(order_date) AS Month,
COUNT(order_id) as Total_Orders
FROM order_t
WHERE order_date BETWEEN #01/01/2008# and #12/31/2008#
GROUP BY salesperson_id,month(order_date)
ORDER BY salesperson_id,month(order_date);
```
Oracle
SELECT salesperson_id, TO_CHAR(order_date,'MON') AS Month,
       COUNT(order_id) AS Total_Orders
FROM order_t
WHERE order_date BETWEEN '01-Jan-2008' and '31-Dec-2008'
GROUP BY salesperson_id,TO_CHAR(order_date,'MON')
ORDER BY salesperson_id,TO_CHAR(order_date,'MON');

35. Each salesperson’s list of customers, by ID:

Query:
SELECT DISTINCT salesperson_id,customer_id
FROM order_t
ORDER BY salesperson_id;

36. Number of orders for each salesperson except salespersons 3, 5, and 9:

Query:
SELECT SalesPerson_Id, COUNT(*)
FROM Order_t
WHERE SalesPerson_Id NOT IN (3, 5, 9)
GROUP BY SalesPerson_ID;

37. For every territory having more than one salesman, display Territory ID and the
   number of salespersons in the territory:

Query:
SELECT Territory_Id, COUNT(*) AS NumSalesPersons
FROM Salesperson_t
GROUP BY Territory_Id
HAVING NumSalesPersons > 1;

38. Based on the symptoms described in the situation, it would appear that the alarm
    system company’s database maintains multiple address fields in the database that
    are not cross-checked with any potential customer promotions. It would appear
    that the mass mailing advertising list that is used to produce the mailings for
    potential customers (and may have been purchased from an outside-of-the-alarm-
    system-company mailing list) is not being cross-checked against a list of the
    company’s current customers’ mailing or physical addresses. Thus, existing
    customers who have a mailing address that is different from the physical address
    of the alarm system are receiving “junk” mail (or spam, if delivered electronically
    by email).
Suggestions for Field Exercises

1. Students may need to be advised that most of the work in conducting an interview is completed before the actual meeting. Some of the tips below may help them conduct a successful interview and increase their self-confidence.

a. Before the interview:
   - Identify the correct person to interview. Job titles do not always denote the same set of responsibilities. The typical database administrator is responsible for configuring and installing systems, setting up the environment for development and support, performing emergency restoration, and ensuring system security, capacity, and performance. The database administrator is a person having central control over data and programs accessing that data.
   - Prepare the interview questions carefully. Phrase the questions so that they are open-ended and cannot be answered by a simple “yes” or “no.”
   - Set up the interview several days in advance. Arrange the exact place and time of the interview.
   - Present yourself professionally, both on the telephone and in person. Explain what topics will be included and specify the length of the interview.

b. During the interview:
   - Begin the interview with a brief, non-business conversation to establish rapport. Objects in the person’s office may offer clues as to the person’s interests or background (school affiliation, particular sports interest, and so forth) and may be a good basis for opening conversation.
   - Try to make the interview conversational. Use the prepared questions as a guideline rather than as a script.
   - Taking notes to review after the interview will help you to get the maximum benefit from the interview.
   - Encourage the presentation of more detailed information if the person being interviewed is speaking in generalities.
   - Clarify any statements that you do not quite understand.
   - At the end of an interview, ask for a contact phone number in case a question would happen to come up.

c. Issues relevant to this particular interview:
   - Most companies will need tools to support three general categories of users:
     1. Query/Reporting (Q/R) tools support queries and reports against operational data and data warehouses. They are often used by knowledge workers.
     2. On-line Analytical Processing (OLAP) tools provide the summarized information for middle management and business analysts.
     3. Executive Information System (EIS) tools, often implemented by adding a front end to OLAP tools, provide high-level reporting and monitoring capabilities for senior management.
Applications that use embedded SQL statements to access and manipulate data are developed using different host languages (Ada, COBOL, C, etc.). There are different means for SQL integration into a language, direct embedded support like PowerBuilder, and indirect embedded support through a pre-processor like zTools (an automatic code generation precompiler for SYBASE that automatically creates source code to link the application program to stored procedures on the SYBASE server using DB-Library calls), or database API support. The downside to Application Programming Interface (API) support is that it takes many lines of tedious C code to equal each line of embedded SQL. To further complicate the issue, many companies are building intranets—private networks on the World Wide Web—to provide their employees, customers, and partners access to information in corporate data warehouses. Decision support tools need to accommodate these users, too.

2. Use the strategies proposed in Exercise 1.

Note: With PL/SQL, you can use SQL statements to manipulate Oracle data and flow-of-control statements to process the data. Moreover, you can declare constants and variables, define subprograms (procedures and functions), and trap runtime errors. Thus, PL/SQL combines the data manipulating power of SQL with the data processing power of procedural languages.

3. Use the strategies proposed in Exercise 1. Student answers will vary based on the company chosen, the background and experience of the interviewee, and the kinds of DBMS that the interviewee has used.

Project

Case Questions

1. The student will need to establish what platform and version of SQL will be used to complete the project case. The solution offered here is based on SQL*Plus, Oracle9i.

2. The student will need to identify any CASE tool such as Visible Analyst or Designer/2000.

3. One possible solution would be to write a program in a high-level language to generate ASCII files, as well as to generate ID fields randomly, and come up with a set of other standard fields. An even better solution would be to use a commercial product, such as TurboData, to automatically populate the tables. Some RDBMSs come with data load utilities to batch load many rows from an ASCII file.
4. The values used for test data help you to test the functionality of the database by checking for situations where things like range checks and other constraints (like primary key and referential integrity) do not make sense.

**Case Exercises**

1. a. create table patient
   (patient_no varchar(5) primary key,
    name varchar(35),
    first_seen date,
    social_worker varchar(35));

   create table visit
   (patient_no varchar(5) references patient(patient_no),
    visit_date date,
    visit_time varchar(5),
    visit_reason varchar(40),
    new_symptoms varchar(50),
    pain_level integer,
    constraint visit_pk primary key (patient_no,visit_date,visit_time);

b. This is left as an exercise for the student.

c. i. select patient_no,name
    from patient
    where social_worker = 'John Smith';

ii. select social_worker, count(*)
    from patient
    group by social_worker;

iii. select min(pain_level)
    from visit;

    select avg(pain_level)
    from visit;

2. a. Information from only one of the tables:

SELECT EMPLOYEE_ID, DATE_HIRED FROM EMPLOYEE_T
ORDER BY EMPLOYEE_ID;
b. Aggregate information from one attribute in a table:

```sql
SELECT COUNT (*)
FROM ITEM_T
WHERE ITEM_DESCRIPTION = 'Television';
```

```sql
SELECT AVERAGE(TOTAL_COST) FROM CONSUMES_T;
```

c. Various functions:

```sql
SELECT MIN (ITEM_COST)
FROM ITEM_T;
```

d. Qualify results by category:

```sql
SELECT PHYSICIAN_ID, COUNT(*)
FROM PERFORMS_T
GROUP BY PHYSICIAN_ID
```

**Project Assignments**

P1.

a. The CREATE TABLE commands suggested here are simple versions and do not contain examples of setting additional parameters such as TABLESPACE, STORAGE, PCTFREE, or PCTUSED. These are optional, but would be used in defining tables for production systems in Oracle.

Further, NOT NULL constraints have not been implemented in this solution, but could be used in the table definitions to ensure data integrity. Also, please note that if you are not using Oracle 10g (or greater), the CHECK clause will cause an error in the execution of the SQL noted below.

```sql
CREATE TABLE PERSON(
  Person_ID   VARCHAR(5) Constraint PER_PERSID_PK PRIMARY KEY,
  Person_Name   VARCHAR(35),
  Person_Str_Address VARCHAR(20),
  Person_City   VARCHAR(20),
  Person_State  CHAR(2),
  Person_Zip   VARCHAR(10),
  Person_Home_Phone  VARCHAR(14),
  Person_Work_Phone  VARCHAR(14),
  Person_DOB   DATE,
  Person_EMail  VARCHAR(25),
  Is_Physician  CHAR(1)  check (Is_Physician in (‘Y’,’N’)),
  Is_Employee   CHAR(1)  check (Is_Employee in (‘Y’,’N’)),
  Is_Volunteer  CHAR(1)  check (Is_Volunteer in (‘Y’,’N’)),
  Is_Patient   CHAR(1)  check (Is_Patient in (‘Y’,’N’));
```
CREATE TABLE PHYSICIAN (  
Physician_ID Varchar(5) Constraint PHY_PHYSID_PK primary key,  
  Constraint PHY_PHYSID_FK references Person(Person_ID),  
DEA_No Varchar(20),  
Pager_No Varchar(14),  
Specialty Varchar(20));

CREATE TABLE PATIENT (  
Patient_ID Varchar(5) Constraint PAT_PATID_PK primary key,  
  Constraint PAT_PATID_FK references Person(Person_ID),  
Contact_Date Date,  
EC_Last_Name Varchar(20),  
EC_First_Name Varchar(20),  
EC_Relationship Varchar(15),  
EC_Address Varchar(55),  
EC_Phone Varchar(14),  
Company_Name Varchar(25),  
Policy_No Varchar(20),  
Group_No Varchar(15),  
Company_Phone Varchar(14),  
Sub_Last_Name Varchar(20),  
Sub_First_Name Varchar(20),  
Sub_Relationship Varchar(15),  
Sub_Address Varchar(55),  
Sub_Phone Varchar(14),  
Is_Outpatient Char(1) check (Is_Outpatient in ('Y','N'))),  
Is_Resident Char(1) check (Is_Resident in ('Y','N'))),  
Admit_phys Varchar(5)  
  Constraint PAT_ADPHYS_PK references Physician(Physician_ID),  
Refer_phys Varchar(5)  
  Constraint PAT_REFPHYS_PK references Physician(Physician_ID));

CREATE TABLE EMPLOYEE (  
Emp_ID Varchar(5) Constraint EMP_EMPID_PK primary key,  
  Constraint EMP_EMPID_FK references Person(Person_ID),  
Date_Hired Date,  
Emp_Type Char(1) check (Emp_Type in ('N','S','T')));

CREATE TABLE VOLUNTEER (  
Vol_ID Varchar(5) Constraint VOL_VOLID_PK primary key,  
  Constraint VOL_VOLID_PK references Person(Person_ID),  
Had_Felony Char(1) check (Had_Felony in ('Y','N'))),  
Felony_Explation Varchar(50),  
VEC_Last_Name Varchar(20),  
VEC_First_Name Varchar(20),  
VEC_Relationship Varchar(15),  
VEC_Address Varchar(55),  
VEC_Phone Varchar(14),  
VEmployer Varchar(25),  
VEmployeer_addr Varchar(55),  
VEmployer_Position Varchar(20),  
VEmployeer_start_Date Date,  
VEmployeer_end_Date Date,  
Had_MVCH_Svc Char(1) check (Had_MVCH_Svc in ('Y','N'))),  
Had_Vol_Exp Char(1) check (Had_Vol_Exp in ('Y','N'))),  
Why_Volunteer Varchar(50));
CREATE TABLE FACILITY (  
    Facility_ID   Varchar(10) Constraint FAC_FID_PK primary key,  
    Facility_Name Varchar(40));

CREATE TABLE VENDOR (  
    Vendor_ID   Varchar(5) Constraint VEN_VID_PK primary key,  
    Vendor_Name Varchar(40));

CREATE TABLE ITEM (  
    Item_No   Varchar(5) Constraint ITM_INO_PK primary key,  
    Item_Desc Varchar(40),  
    Unit_Cost Number(7,2));

CREATE TABLE DIAGNOSIS (  
    Diagnosis_Code Varchar(5) Constraint DX_DXCODE_PK primary key,  
    Diagnosis_Name Varchar(60));

CREATE TABLE OUTPATIENT (  
    O_Patient_ID Varchar(5) Constraint OP_OPID_PK primary key  
        Constraint OP_OPID_FK references Patient(Patient_ID));

CREATE TABLE VISIT (  
    Visit_No   Varchar(5) Constraint VIS_VNO_PK primary key,  
    O_Patient_ID Varchar(5)  
        Constraint VIS_OPID_FK references Outpatient(O_Patient_ID),  
    Visit_Date Date,  
    Visit_Time Timestamp,  
    Visit_Reason Varchar(50));

CREATE TABLE NURSE (  
    Nurse_ID   Varchar(5) Constraint NUR_NID_PK primary key  
        Constraint NUR_NID_FK references Employee(Emp_ID),  
    Cert_Degree Varchar(5),  
    State_License_No Varchar(15),  
    Nurse_Specialty Varchar(20),  
    Nurse_Type Char(1) check (Nurse_Type in ('R','L')));

CREATE TABLE RN (  
    RN_ID   Varchar(5) Constraint RN_RNID_PK primary key  
        Constraint RN_RNID_FK references Nurse(Nurse_ID));

CREATE TABLE LPN (  
    LPN_ID   Varchar(5) Constraint LPN_LPNID_PK primary key  
        Constraint LPN_LPNID_FK references Nurse(Nurse_ID)  
        Supervisor Varchar(5) Constraint LPN_RNID_FK references RN(RN_ID));

CREATE TABLE WORK_UNIT (  
    Unit_Name Varchar(20) Constraint WU_UName_PK primary key,  
    Floor    Varchar(3),  
    Facility_ID Varchar(10)  
        Constraint WU_FID_PK references Facility(Facility_ID),  
    Unit_Type Char(2) check (Unit_Type in ('CC','DU')));
CREATE TABLE CARE_CENTER(
CC_Unit_Name Varchar(20) CC_CCNAME_PK primary key,
    Constraint CC_CCNAME_PK references Work_Unit(Unit_Name),
Day_in_Charge Varchar(5) Constraint CC_AMID_FK references RN(RN_ID),
Night_in_Charge Varchar(5) Constraint CC_PMID_FK references RN(RN_ID));

CREATE TABLE ROOM(
    Room_No Varchar(5) Constraint ROOM_RNO_PK primary key,
CC_Unit_Name Varchar(20)
    Constraint ROOM_CC_FK references CARE_CENTER(CC_Unit_Name));

CREATE TABLE BED(
    Bed_No Varchar(5),
    Room_No Varchar(5) Constraint BED_RNO_FK references ROOM(Room_No),
    Constraint Bed_PK primary key(Room_No, Bed_No));

CREATE TABLE RESIDENT(
    R_Patient_ID Varchar(5) Constraint RES_RID_PK primary key,
    Date_Admitted Date,
    Date_Discharged Date,
    Bed_No Varchar(3),
    Room_No Varchar(5),
    Constraint RES_ROOMBED_FK foreign key(Room_No, Bed_No)
        references Bed(Room_No, Bed_No));

CREATE TABLE PHYSICIAN_DX(
    PD_ID Varchar(5) Constraint PDX_PDID_PK primary key,
    Diagnosis_Date Date,
    Diagnosis_Time Timestamp,
    Diagnosis_Code Varchar(5)
    Constraint PDX_DXCODE_FK references Diagnosis(Diagnosis_Code),
Physician_ID Varchar(5)
    Constraint PDX_PHYID_FK references Physician(Physician_ID),
Patient_ID Varchar(5)
    Constraint PDX_PATID_FK references Patient(Patient_ID));

CREATE TABLE CC_ASSIGNMENT(
    CCA_ID Varchar(5) Constraint CCA_CCAID_PK primary key,
    Assign_Start Date,
    Assign_End Date,
    Hrs_Worked Number (4,2),
    Nurse_ID Varchar(5) Constraint CCA_NID_FK references Nurse(Nurse_ID),
CC_Unit_Name Varchar(20)
    Constraint CCA_CCNAME_FK references Care_Center(CC_Unit_Name));

CREATE TABLE FIELD_CERTIFICATION(
    FC_ID Varchar(5) Constraint FC_FCID_PK primary key,
FC_Description Varchar(30),
    Nurse_ID Varchar(5) Constraint FC_NID_FK references Nurse(Nurse_ID));
CREATE TABLE ASSESSMENT (  Assessment_ID Varchar(5) Constraint AS_ASID_PK primary key,  Assessment_Date Date,  Assessment_Time Timestamp,  Comments Varchar(50),  Patient_Weight Number(3),  Patient_BP Varchar(7),  Patient_pulse Number(4),  Patient_temperature Number(3,2)  Patient_ID Varchar(5)  Constraint AS_PATID_PK references Patient(Patient_ID),  Nurse_ID Varchar(5)  Constraint AS_NID_PK references Nurse(Nurse_ID));

CREATE TABLE STAFF (  Staff_ID Varchar(5) Constraint STF_STID_PK primary key  Constraint STF_STID_FK references Employee(Emp_ID),  Job_Class Varchar(3),  Unit_Name Varchar(20)  Constraint STF_UName_FK references Work_Unit(Unit_Name));

CREATE TABLE TECHNICIAN (  Technician_ID Varchar(5) Constraint T_TID_PK primary key  Constraint T_TID_FK references Employee(Emp_ID),  Unit_Name Varchar(20)  Constraint T_UName_FK references Work_Unit(Unit_Name));

CREATE TABLE TECHNICIAN_SKILL (  TS_ID Varchar(5) Constraint TS_TSID_PK primary key,  Technician_ID Varchar(5)  Constraint TS_TID_FK references Technician(Technician_ID),  TS_Skill Varchar(20));

CREATE TABLE DIAGNOSTIC_UNIT (  DX_Unit_Name Varchar(20) Constraint DU_DXName_PK primary key  Constraint DU_DXName_FK references Work_Unit(Unit_Name));

CREATE TABLE TREATMENT (  Trt_Code Varchar(5) Constraint TRT_TCode_PK primary key,  Treatment_Name Varchar(30),  DX_Unit_Name Varchar(20)  Constraint TRT_DXName_FK references Diagnostic_Unit(DX_Unit_Name));

CREATE TABLE ORDER (  Order_ID Varchar(5) Constraint OR_ORDID_PK primary key,  Patient_ID Varchar(5)  Constraint OR_PATID_PK references Patient(Patient_ID),  Physician_ID Varchar(5)  Constraint OR_PHYID_PK references Physician(Physician_ID),  Item_NO Varchar(5)  Constraint OR_ITNO_PK references Item(Item_No),  Order_Date Date,  Order_Time Timestamp);
CREATE TABLE TREATMENT_ORDER (  
TO_ID Varchar(5) Constraint TO_TOID_PK primary key,  
Trt_Code Varchar(5)  
Constraint TO_TCODE_FK references Treatment(Trt_Code),  
Results Varchar(50),  
Trt_Date Date,  
Trt_Time Timestamp,  
Order_ID Varchar(5)  
Constraint TO_ORID_FK references Order(Order_ID));

CREATE TABLE ITEM_CONSUMPTION (  
IC_ID Varchar(5) Constraint IC_ICID_PK primary key,  
Consume_Date Date,  
Consume_Time Timestamp,  
Consume_Qty Number(3),  
Order_ID Varchar(5) references Orders(Order_ID),  
Item_No Varchar(5) Constraint IC_ITNO_FK references  
Item(Item_No),  
Patient_ID Varchar(5)  
Constraint IC_PATID_FK references Patient(Patient_ID));

CREATE TABLE ITEM_BILLING (  
IB_ID Varchar(5) Constraint IB_IBID_PK primary key,  
Start_Date Date,  
End_Date Date,  
Cost Number (9,2),  
Item_No Varchar(5) Constraint IB_ITNO_FK references  
Item(Item_No),  
Room_No Varchar(5) Constraint IB_RNO_FK references  
Room(Room_No),  
Patient_ID Varchar(5)  
Constraint IB_PATID_FK references Patient(Patient_ID));

CREATE TABLE INVENTORY (  
INV_ID Varchar (5) Constraint INV_INVID_PK primary key,  
Item_No Varchar(5) Constraint INV_ITNO_FK references  
Item(Item_No),  
Vendor_ID Varchar(5) Constraint INV_VID_FK references  
Vendor(Vendor_ID));

CREATE TABLE SCHEDULE (  
Schedule_ID Varchar(5) Constraint SCH_SCHID_PK primary key,  
Sched_Begin Date,  
Sched_End Date,  
Physician_ID Varchar(5)  
Constraint SCH_PHYID_FK references Physician(Physician_ID),  
Facility_ID Varchar(10)  
Constraint SCH_FID_FK references Facility(Facility_ID));

CREATE TABLE VOL_MVCH_SERVICE (  
VMS_ID Varchar(5) Constraint VMS_VMSID_PK primary key,  
MVCH_Service_Info Varchar(25),  
Vol_ID Varchar (5)  
Constraint VMS_VID_FK references Volunteer(Vol_ID));
CREATE TABLE VOL_REF_INFO (  VRI_ID Varchar(5) Constraint VRI_VRID_PK primary key,  VRI_Last_Name Varchar(20),  VRI_First_Name Varchar(20),  VRI_Relationship Varchar(15),  VRI_Phone Varchar(14),  VRI_Address Varchar(20),  VRI_City Varchar(20),  VRI_State Varchar(2),  VRI_Zip Varchar(10),  Vol_ID Varchar(5)  Constraint VRI_VID_FK references Volunteer(Vol_ID));

CREATE TABLE VOL_EXPERIENCE (  VE_ID Varchar(5) Constraint VE_VEID_PK primary key,  Volunteer_Exp_Info Varchar(25),  Vol_ID Varchar(5)  Constraint VE_VID_FK references Volunteer(Vol_ID));

CREATE TABLE VOL_LANGUAGE (  VL_ID Varchar(5) Constraint VL_VLID_PK primary key,  Language Varchar(15),  Vol_ID Varchar(5)  Constraint VL_VID_FK references Volunteer(Vol_ID));

CREATE TABLE VOL_SKILL (  VS_ID Varchar(5) Constraint VS_VSID_PK primary key,  Skill Varchar(25),  Vol_ID Varchar(5)  Constraint VS_VID_FK references Volunteer(Vol_ID));

CREATE TABLE VOL_INTEREST (  VI_ID Varchar(5) Constraint VI_VID_PK primary key,  Interest Varchar(25),  Vol_ID Varchar(5)  Constraint VI_VID_FK references Volunteer(Vol_ID));


CREATE TABLE VOL_SERV_HISTORY (  VSH_ID Varchar(5) Constraint VSH_VSHID_PK primary key,  Service_Begin_Date Date,  Service_End_Date Date,  Service_Hrs_Worked Varchar(3),  Vol_ID Varchar(5)  Constraint VSH_VID_FK references Volunteer(Vol_ID),  Unit_Name Varchar(15)  Constraint VSH_UName_FK references Work_Unit(Unit_Name),  Physician_ID Varchar(5)  Constraint VSH_PhyID_FK references Physician(Physician_ID),  Emp_ID Varchar(5)  Constraint VSH_EmpID_FK references Employee(Emp_ID));
P1b. The following commands will create secondary indexes based upon specifications from chapter 6:

```
create index dr_dx_idx on physician_dx(physician_id);
create index dxcode_idx on physician_dx(diagnosis_code);
create index patient_dx_idx on physician_dx(patient_id);
create index res_date_adm_idx on resident(date_admitted);
create index res_date_dischg_idx on resident(date_discharged);
create index trtorder_date_idx on treatment_order(trt_date);
create index odate_index on order(order_date);
create index item_cost_idx on item(item_unit_cost);
create index consume_qty_idx on item_consumption(consume_qty);
```

P2 and P3. These are left as student exercises. Be sure that students include queries that utilize the GROUP BY clause as well as aggregate functions.
Chapter 8  Advanced SQL

Chapter Overview

Chapter 8 follows from Chapter 7, from single table queries to multi-table joins, subqueries (both non-correlated and correlated), establishing referential integrity, and derived tables. Triggers, stored procedures, functions, embedded SQL, dynamic SQL, and Persistent Stored Modules are also covered. This chapter also contains a detailed discussion of transaction integrity as well as the SQL-200n enhancements and extensions to SQL and an overview of data dictionaries. Chapter 7 is a prerequisite for this chapter.

Chapter Objectives

Specific student learning objectives are included at the beginning of the chapter. From an instructor’s point of view, the objectives of this chapter are to:

1. Build the student’s SQL skills and an appreciation of SQL through many examples of relational queries from SQL; demonstrate capabilities such as multiple-table data retrieval (join and other operators such as difference, union, and intersection), explicit and implicit joining, and built-in functions.
2. Illustrate the differences between the joining and subquery approaches to manipulating multiple tables in SQL.
3. Introduce the transaction and concurrency control features of relational DBMSs.
4. Discuss the SQL-200n enhancements to SQL.
5. Briefly discuss the data dictionary facilities available in Oracle 10g.
6. Discuss triggers and stored procedures and provide examples of how these might be used.
7. Briefly discuss dynamic and embedded SQL.
8. Understand OLTP and OLAP and how SQL is used in writing queries for the two approaches.

Key Terms

<table>
<thead>
<tr>
<th>Correlated subquery</th>
<th>Natural join</th>
<th>Persistent Stored Modules (SQL/PSM)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dynamic SQL</td>
<td>Online analytical processing (OLAP)</td>
<td></td>
</tr>
<tr>
<td>Embedded SQL</td>
<td>Online transaction processing (OLTP)</td>
<td>Procedure</td>
</tr>
<tr>
<td>Equi-join</td>
<td>Outer join</td>
<td>Trigger</td>
</tr>
<tr>
<td>Function</td>
<td></td>
<td>User-defined data type (UDT)</td>
</tr>
<tr>
<td>Join</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Classroom Ideas

1. Have students program in a system that supports SQL along with this chapter. The nuances of joining multiple tables, nesting subqueries, properly qualifying built-in functions, and so forth are really only learned by writing a wide variety of non-trivial queries. There are exercises at the end of the chapter that will provide such practice for students.

2. If students have access to Oracle, have them take a look at the various data dictionary views available to them as a user. You may also want to discuss the various DBA views available and show these to the students during your lecture. Remember that Teradata University supports Oracle for classroom use, and that you may set up access for yourself and your students. The databases from the text are available, as are much larger datasets that you may want to use. Teradata University’s home page is www.teradatastudentnetwork.com.

3. When discussing multiple table queries, always emphasize that there is more than one way to write a query. For example, show the students a query using a join and then the same query using subqueries.

4. Emphasize the cases when a subquery is needed and cannot be substituted with a join. A good example of this would be the case where one needs to find all customers who have never purchased a product (using a subquery with the NOT IN qualifier).

5. Develop an exercise for the students to explore the effects of a trigger. Have them create and populate some tables, then write an insert trigger for one of the tables that might impact other tables. You could then have the students insert some records and see the results. It is important to show the effects of triggers through examples that the students can try out, followed by problems that they would have to solve by writing triggers. The same can be said for stored procedures.

6. The discussion on SQL/PSM might be a good place to introduce PL/SQL before introducing triggers and stored procedures.

Answers to Review Questions

1. Define each of the following key terms:
   a. **Dynamic SQL** The process of making an application capable of generating specific SQL code on the fly, as the application is processed.
   b. **Correlated subquery** This type of subquery is processed outside in, rather than inside out. That is, the inner query is executed for each row in the outer query, and the inner query depends in some way on values from the current row in the outer query.
   c. **Embedded SQL** The process of including hard-coded SQL statements in a program written in another language such as C or Java.
   d. **Procedure** A collection of procedural and SQL statements that are assigned a unique name within the schema and stored in the database.
   e. **Join** The most frequently used relational operation, which brings together data from two or more related tables into one result table.
Chapter 8

f. **Equi-join** A join in which the joining condition is based on equality between values in the common columns. It produces a table of rows composed of columns from two other tables, where common columns appear (redundantly) in the result table.

g. **OLTP** A system used in transaction-oriented applications that involves real-time processing of SQL transactions. It is characterized by fast data entry and retrieval in multiuser environments.

h. **Outer join** A join in which rows that do not have matching values in common columns are nevertheless included in the result table. Outer joins return all the values in one of the tables included in the join, regardless of whether a match exists in the other table(s) or not.

i. **Function** A stored subroutine that returns one value and has only input parameters.

j. **Persistent Stored Modules (SQL/PSM)** Extensions defined in SQL-99 that include the capability to create and drop modules of code stored in the database schema across user sessions.

2. Match the following terms to the appropriate definitions:
   
e. equi-join
   
i. natural join
   
d. outer join
   
j. trigger
   
k. procedure
   
g. Embedded SQL
   
b. UDT
   
f. COMMIT
   
c. SQL/PSM
   
h. Dynamic SQL
   
a. ROLLBACK

3. Using an outer join instead of a natural join:

   Outer joins are often used in database maintenance to find rows that do not have matching values in common columns. Null values appear in columns where there is no match between tables. Another example would be a query that returns all customers—whether they have placed orders in the last four months or not—along with the date of the most recent order placed within the last four months. Customers who have not placed an order would be returned with a null value under most recent order.

4. Explain the processing order of a correlated subquery:

   Correlated subqueries use the result of the outer query to determine the processing of the inner query. Thus, the inner query may be somewhat different for each row referenced in the outer query.
5. Explain that any query that can be written using the subquery approach can also be written using the joining approach, but not vice versa:

While SQL*PLUS allows a subquery to return more than one column, most systems allow pairwise joining of *one and only one column* in an inner query with *one column* in an outer query. (An exception to this is when a subquery is used with the EXISTS keyword.) You can display data only from the table(s) referenced in the outer query. If you want to include data from the subquery in the result, then you would have to use the join technique because data from the subquery cannot be included in the results. The joining technique is useful when data from several relations are to be retrieved and displayed, and the relationships are not necessarily nested.

6. Purpose of the COMMIT command; its relation to the notion of a business transaction:
   a. SQL transactions terminate by executing either a COMMIT or ROLLBACK operation. COMMIT [WORK] takes the contents of the log file and applies them to the database and then empties the log file. There is also an AUTOCOMMIT (ON/OFF) command in many RDBMSs that specifies whether changes are made permanent after each data modification command (ON) or only when work is explicitly made permanent (OFF) by the COMMIT WORK command. These commands are necessary to maintain a valid database and are transparent to the user in most interactive SQL situations.
   b. SQL transactions are logical units of work. Either all of the operations performed in the SQL transaction will be committed, or none of the operations will be committed to the database. An SQL transaction may be more involved than an accounting transaction. For example, the entry of a customer order may also trigger inventory adjustment. Executing the COMMIT command will either make permanent changes to all relations involved in the logical unit of work, or it will make changes to none of them.

7. Hidden triggers: They are hard to see coming until they fail to fire. They may fire without notification. Cascading triggers and endless loop triggers are also possible.

8. Three parts of a trigger are identified: the event, condition, and action sections. The event defines the change about to be made, such as an UPDATE or DELETE of a record. The condition section examines the record(s) about to be affected. For each record that meets that condition, the action to be taken by the trigger begins.

9. Use UNION when you want to combine the output from multiple queries together. However, each query involved must output the same number of rows, and they must be UNION-compatible.

10. Adds a set of analytical functions. Makes it easier to work with results that are in a particular order, e.g., find the top ten salesmen. More efficient processing with
optimized functions. Will add another clause to SQL, the WINDOW clause, where the partitioning, ordering, and aggregation grouping for the window can be specified.

11. Make SQL computationally complete (database applications and so forth).

12. Flexibility, efficiency, sharability, applicability.

13. **Embedded SQL** To create an application where you know exactly what the SQL syntax is that you will need to use.

   **Dynamic SQL** Use where you need to create SQL on the fly, identifying exact parameter values, tables, and so forth at runtime.

14. The CASE keyword would be useful in a situation where you might want to assign categories, for example a discount level based upon sales. In this example, one could use the CASE keyword to check the sales level and assign a discount level, such as level 1 for sales < 1000, level 2 for sales >= 1000 and <5000, etc.

15. Derived tables are used to create a temporary table that is treated as if it were an actual table. This table is not persistent in that it goes away after the query in which it was created is run.

16. One example of the use of a derived table would be to find all ships that were loaded beyond capacity. In this example, a shipment’s weight is calculated by computing the sum of the quantity order times the weight. The query follows:
   ```sql
   SELECT ship.ship_no
   FROM ship, shipment,
   (SELECT shipment_line.shipment_id,
       SUM(item.weight*shipment_line.quantity) AS tweight
   FROM shipment_line,item
   WHERE shipment_line.item_no = item.item_no
   GROUP BY shipment_id) AS ship_wt
   WHERE ship.ship_no = shipment.ship_no
   AND shipment.shipment_id = ship_wt.shipment_id
   AND ship.capacity < ship_wt.tweight;
   ```

17. A view can be used in place of a derived table. A derived table is better in cases where you run the query infrequently because you do not have to remember to delete the view. Views are better when the view will be accessed frequently. They may be used to create a limited view of a database for a specialized end-user, such as an Accounts Payable clerk.

18. One possibility would be to convert one of the data types. For example, if one data type is a character and the other numeric, you could use a function such as Oracle’s `TO_CHAR` to convert the numeric to a character. Another option is to decide which
tables might be involved in UNION operations and make sure that the data types are compatible.

19. The outer join is not easily implemented for more than two tables. The results vary by RDBMS vendor and should be thoroughly tested before implementing.

20. This is left as an exercise for students. The textbook shows examples of different system table names between Oracle and Microsoft SQL Server, which would be one difference in the data dictionary facilities between vendors. Other differences that might be apparent if students have hands-on access to both vendor DBMSs could be what system tables the students/users have access to based on the database authorization and security setup. Comparisons of the major DBMS vendors and their adherence to SQL standards related to data dictionary functions are detailed as a work-in-progress at http://troels.arvin.dk/db/rdbms (accessed 27 December 2007).

Answers to Problems and Exercises

Note: The solutions, which include SQL statements, are not intended as the definitive answer to the questions, but as possible solutions. Instructors and students will approach the problems using different SQL capabilities, achieving results that are also correct.

1. 
   a. Display the course ID and course name for all courses with an ISM prefix:

   **Query:**
   ```sql
   SELECT COURSE_ID, COURSE_NAME
   FROM COURSE
   WHERE COURSE_ID LIKE 'ISM%';
   ```

   b. Display all courses for which Professor Berndt has been qualified:

   **Query:**
   ```sql
   SELECT FACULTY_NAME, COURSE.COURSE_ID, COURSE_NAME
   FROM FACULTY, COURSE, QUALIFIED
   WHERE QUALIFIED.FACULTY_ID = 3467
   AND FACULTY.FACULTY_ID= QUALIFIED.FACULTY_ID
   AND COURSE.COURSE_ID=QUALIFIED.COURSE_ID;
   ```
c. Display the class roster, including student name, for all students enrolled in section 2714 of ISM 4212:

**Query:**

SELECT STUDENT.STUDENT_ID, STUDENT_NAME, SECTION.COURSE_ID, REGISTRATION.SECTION_NO, SEMESTER FROM STUDENT, REGISTRATION, SECTION WHERE SECTION.SECTION_NO = REGISTRATION.SECTION_NO AND STUDENT.STUDENT_ID = REGISTRATION.STUDENT_ID AND REGISTRATION.SECTION_NO = 2714 AND SECTION.SEMESTER = 'I-2008' AND REGISTRATION.SEMESTER = 'I-2008' ORDER BY STUDENT_NAME;

2. Which instructors are qualified to teach ISM 3113?

**Query:**

SELECT FACULTY.FACULTY_NAME FROM FACULTY, QUALIFIED WHERE QUALIFIED.FACULTY_ID = FACULTY.FACULTY_ID AND QUALIFIED.COURSE_ID = 'ISM 3113';

3. Is any instructor qualified to teach ISM 3113 and not qualified to teach ISM 4930?

**Query:**

SELECT FACULTY.FACULTY_NAME FROM FACULTY, QUALIFIED WHERE QUALIFIED.FACULTY_ID = FACULTY.FACULTY_ID AND QUALIFIED.COURSE_ID = 'ISM 3113' AND NOT (QUALIFIED.COURSE_ID = 'ISM 4930');

4. a. How many students are enrolled in section 2714 during semester I-2008?

**Query:**

SELECT COUNT(DISTINCT (STUDENT_ID)) FROM REGISTRATION WHERE SECTION_ID = 2714 AND SEMESTER = 'I-2008';
b. How many students are enrolled in ISM 3113 during semester I-2008?

**Query:**

```
SELECT COUNT (DISTINCT (STUDENT_ID))
FROM SECTION, REGISTRATION
WHERE SECTION.SECTION_NO = REGISTRATION.SECTION_NO
AND COURSE_ID = 'ISM 3113'
AND SEMESTER = 'I-2008';
```

5. Which students were not enrolled in any courses during semester I-2008?

**Query:**

```
SELECT DISTINCT STUDENT_ID, STUDENT_NAME
FROM STUDENT
WHERE NOT EXISTS
  (SELECT * FROM REGISTRATION
   WHERE STUDENT.STUDENT_ID =
   REGISTRATION.STUDENT_ID
   AND SEMESTER='I-2008');
```

**Note to instructor:** Problems and Exercises 6–14 are based on Figure 8-14 and continue a problem set from Chapter 7 (problems 10–15, based on Figure 7-12). Please note that the chapter 8 problem set alters the design of the database from the earlier design shown in chapter 7. It may be useful for students to build this small database in a particular DBMS environment and populate the tables with sample data as they work on the requested queries. The SQL illustrated in this sample solution is based on MS Access SQL.

6. List primary and foreign keys for all entities in Figure 8-14:

**TUTOR**
- Primary key: TUTOR_ID
- Foreign key: none

**STUDENT**
- Primary key: STUDENT_ID
- Foreign key: none

**MATCH_HISTORY**
- Primary key: MATCH_ID
- Foreign keys: TUTOR_ID references TUTOR(TUTOR_ID)
  STUDENT_ID references STUDENT(STUDENT_ID)

**TUTOR_REPORT**
- Primary key: MATCH_ID + MONTH (composite PK)
- Foreign key: MATCH_ID references MATCH_HISTORY(MATCH_ID)
7. Write SQL to add Math_Score to Student table:

**SQL:**
ALTER TABLE STUDENT
ADD COLUMN MATH_SCORE NUMBER(2,1);

8. Write SQL to add Subject to Tutor table:

**SQL:**
ALTER TABLE TUTOR
ADD COLUMN SUBJECT VARCHAR(7)
CHECK (SUBJECT IN ('Reading', 'Math', 'ESL'));

9. One approach is to adjust the database design to allow tutors the ability to declare more than one subject preference for tutoring. Based on the prior specifications (including problem and exercise 8), the business rule appeared to be that tutors only tutored in one subject, thus the subject could be stored with tutor information. Under this new requirement, the database needs the ability to track more than one subject per tutor. This can be accomplished by adding two tables to the current design (Tutor_Subject, Subject) and by altering the structure of the MATCH_HISTORY table to track the Subject involved in each match of Tutor and Student, as noted in the revised ERD.

It is also possible to assume that (1) each tutor is certified only once or (2) that a tutor must be certified in each of the subjects. Assumption 2 would require that CERT_DATE be moved to the Tutor_Subject relation. This is a good opportunity to show how making an assumption without interviewing the database user may result in an incorrect data model because the correct business rule has not been uncovered.

Some students will be stimulated by adding the subjects “Reading, Math, and ESL” to the teachers’ certifications to ask if one would not also then track what the students were studying. This can be used in class as an example of how an analyst can uncover additional business rules that have not been initially elicited from the client in interviews. You may expand this question to track student assessment scores for math and ESL in addition to the existing READ score. This should lead to consideration that, just as tutors may only be certified in one area, students may elect to study in one to three areas, depending on their needs. The data model would need to be expanded further to handle this.
10. (Answer provided in MS ACCESS SQL). Students who build a practice database to answer this problem should use a DATE format and may need to add a day-of-month value to the data provided. Note that a left outer join is necessary to pick up tutors who have never submitted a report.

First, a query named *CH8Q10* returns a list of all reports ever submitted for each currently active student. The query is sorted by tutor and report dates for that tutor.

```
SELECT MATCH_HISTORY.MATCH_ID, MATCH_HISTORY.END_DATE,
MATCH_HISTORY.TUTOR_ID, TUTOR_REPORT.MONTH
FROM MATCH_HISTORY LEFT JOIN TUTOR_REPORT ON
MATCH_HISTORY.MATCH_ID = TUTOR_REPORT.MATCH_ID
WHERE (((MATCH_HISTORY.END_DATE) Is Null)) OR
(((MATCH_HISTORY.END_DATE)>#6/30/2008#) AND
((TUTOR_REPORT.MONTH) Is Null))
ORDER BY MATCH_HISTORY.MATCH_ID,
TUTOR_REPORT.MONTH;
```

Next, a query is built to retrieve data from query *Ch8Q10*. This second query returns just those tutors with active students who have not submitted a report for July.

```
SELECT MATCH_HISTORY.MATCH_ID, MATCH_HISTORY.END_DATE,
MATCH_HISTORY.TUTOR_ID, TUTOR_REPORT.MONTH
FROM MATCH_HISTORY LEFT JOIN TUTOR_REPORT ON
MATCH_HISTORY.MATCH_ID = TUTOR_REPORT.MATCH_ID
WHERE (((MATCH_HISTORY.END_DATE) Is Null)) OR
(((MATCH_HISTORY.END_DATE)>#6/30/2008#) AND
((TUTOR_REPORT.MONTH) Is Null))
ORDER BY MATCH_HISTORY.MATCH_ID,
TUTOR_REPORT.MONTH;
```

Students should be encouraged to modify or add records to the example in the book in order to test their query thoroughly. As given, only Tutor 104 with a new student who started in June has not submitted a report for July. In order to test their query more completely, data should be created for tutors who have active students and who have previously submitted monthly reports but have not submitted one in June.
SELECT CH8P10.TUTOR_ID, CH8P10.MONTH
FROM CH8P10
WHERE (((CH8P10.MONTH) Is Null)) OR (((CH8P10.MONTH)>=#6/30/2008#
And (CH8P10.MONTH) Not Between #7/1/2008# And #7/31/2008#));

Motivated students may add parameters to this query so it will work for any month.

11. Note: This solution assumes the creation of a PERSON table to store the common contact data for STUDENTs and TUTORs, and the linking of the STUDENT and TUTOR table to the PERSON table. Individual student answers may vary from this proposed solution due to this assumption.

CREATE TABLE PERSON (  
  Person_ID   Varchar(5)
  CONSTRAINT PER_PERSID_PK PRIMARY KEY,
  Last_Name   Varchar(15),
  First_Name  Varchar(15),
  Middle_Init Varchar(1),
  Person_Str_Address Varchar(20),
  Person_City Varchar(20),
  Person_State Char(2),
  Person_Zip  Varchar(10),
  Person_Phone Varchar(14),
  Person_EMail Varchar(25),
  Person_Type Char(1));

ALTER TABLE STUDENT (  
  ADD COLUMN PERSON_ID VARCHAR(5));

ALTER TABLE STUDENT (  
  ADD CONSTRAINT ST_PERSONID_FK
  PERSON_ID REFERENCES PERSON(PERSON_ID));

ALTER TABLE TUTOR (  
  ADD COLUMN PERSON_ID VARCHAR(5));

ALTER TABLE TUTOR (  
  ADD CONSTRAINT T_TUTORID_FK
  REFERENCES PERSON(PERSON_ID));

12. List all active students in June by name, including number of hours tutored and number of lessons completed:

Query:
SELECT STUDENT.STUDENT_ID, MATCH_HISTORY.END_DATE,
  PERSON.Last_Name, Sum(TUTOR_REPORT.HOURS) AS [Total Hours],
  Sum(TUTOR_REPORT.LESSONS) AS [Total Lessons]
FROM (PERSON INNER JOIN STUDENT ON PERSON.PERSON_ID =
  STUDENT.STUDENT_ID) INNER JOIN MATCH_HISTORY LEFT
JOIN TUTOR_REPORT ON MATCH_HISTORY.MATCH_ID =

13. Which tutors, by name, are available to tutor?

This answer assumes any active tutor may be available to accept a new student:

Query:
```
SELECT PERSON.Last_Name, PERSON.First_Name, TUTOR.STATUS
FROM PERSON INNER JOIN TUTOR ON
PERSON.PERSON_ID = TUTOR.PERSON_ID
WHERE (((TUTOR.STATUS)="Active"));
```

This answer assumes a tutor is available only if currently unassigned a student:

```
SELECT T.TUTOR_ID, PERSON.Last_Name, PERSON.First_Name
FROM PERSON INNER JOIN TUTOR AS T ON
PERSON.PERSON_ID = T.PERSON_ID
WHERE (((T.TUTOR_ID) In (SELECT MH.TUTOR_ID  FROM MATCH_HISTORY MH WHERE END_DATE IS NOT NULL) And
(T.TUTOR_ID) Not In (SELECT MH.TUTOR_ID  FROM MATCH_HISTORY MH WHERE END_DATE IS  NULL)) AND ((T.STATUS)="Active"));
```

14. Which tutor needs to be reminded to turn in reports?

Based on the answer to Question 10 (just pick up the tutor’s name from PERSON).

Query:
```
SELECT [Ch8P10_part_2_query].TUTOR_ID, PERSON.Last_Name,
PERSON.First_Name
FROM PERSON INNER JOIN (TUTOR INNER JOIN [Ch8P10_part_2_query]
ON TUTOR.TUTOR_ID = [Ch8P10_part_2_query].TUTOR_ID) ON
PERSON.PERSON_ID = TUTOR.PERSON_ID;
```

Note to instructor: Problems and Exercises 15–35 are based on the extended version of the Pine Valley Furniture Company database (BigPVFC.mdb is the MS Access file version of this database; this extended database version is also available on Teradata student resources). Please note that this version of the database has a different structure than that in the textbook version of the database (e.g., the salesperson information is in the extended version but not in the textbook version). Some of the field names may also have changed due to the version of the database you are using due to the reserved words.
of the DBMS. When you first use the database, check the table definitions to see what the exact field names and table structures are for the DBMS that you are using. Also note that, where possible, solutions are presented in both MS Access and Oracle SQL syntax (although an Oracle version of the extended PVFC database is not provided in the SQL scripts).

15. Display order number, customer number, order date and items ordered for order #1:

**Microsoft Access Query:**

```sql
SELECT Order_t.Order_ID, Order_t.Customer_ID, Order_t.Order_Date,
       Order_line_t.Product_ID, PRODUCT_t.Product_Description,
       Order_line_t.Ordered_quantity
FROM (PRODUCT_t INNER JOIN Order_t ON PRODUCT_t.Product_ID =
       Order_t.Product_ID) INNER JOIN Order_line_t ON Order_t.Order_ID =
       Order_line_t.Order_ID
WHERE (((Order_t.Order_ID)= 1));
```

**Oracle Query:**

```sql
SELECT ORDER_t.Order_ID, ORDER_t.Customer_ID, ORDER_t.Order_Date,
       Order_line_t.Product_ID, PRODUCT_T.Product_Description,
       Order_line_t.Ordered_Quantity
FROM Order_line_t, ORDER_t, PRODUCT_t
WHERE ORDER_t.Order_ID=Order_line_t.Order_ID AND
       ORDER_line_t.Product_ID=PRODUCT_t.Product_ID AND
       ORDER_t.Order_ID = 1;
```

16. Display each item ordered for order #1, its standard price, and total price for each item ordered:

**Microsoft Access Query:**

```sql
SELECT Order_line_t.Product_ID, Product_t.Standard_Price,
       Sum(Order_line_t.Ordered_quantity)*Product_t.Standard_Price AS Total
FROM Product_t INNER JOIN Order_line_t ON Product_t.Product_ID =
       Order_line_t.Product_ID
GROUP BY Order_line_t.Product_ID, Product_t.Standard_Price,
       Order_line_t.Order_ID
HAVING (((Order_line_t.Order_ID)= 1));
```

**Oracle Query:**

```sql
SELECT Order_line_t.Product_ID, Product_t.Standard_Price,
       Sum(Order_line_t.Ordered_quantity)*Product_t.Standard_Price
       AS Total_Price
FROM Product_t,order_line_t
WHERE Product_t.Product_ID = Order_line_t.Product_ID
```
GROUP BY Order_line_t.Product_ID, Product_t.Standard_Price,
Order_line_t.Order_ID
HAVING Order_line_t.Order_ID=1;

17. Total the cost of the order for order #1:

This solution uses the results of Question 16, saved as a query or view named order_1:

Both MS-Access and Oracle are the same.

**Query:**
Select sum(order_1.total_price) as Total_Cost
from order_1;

18. Find customers who have not placed any orders:

**Microsoft Access Query:**
SELECT customer_t.Customer_Id
FROM customer_t
WHERE (((customer_t.Customer_Id) Not In (select customer_id from order_t)));

**Oracle Query:**
SELECT customer_t.Customer_Id
FROM customer_t
WHERE customer_t.Customer_Id Not In (select customer_id from order_t);

19. Produce a list of all the products (show product description) and the number of times each product has been ordered:

This query requires an outer join because some products may not have been ordered. Because many SQL systems do not have an outer join operator, often this type of query must use the UNION command. The following answer uses this second approach because it will work with almost any system. Also, note that the question wants the number of times a product has been ordered, not the total quantity ordered:
Microsoft Access Query:

```
SELECT Product_t.Product_ID, Product_Description, COUNT(*) as TimesOrdered
FROM Product_t INNER JOIN Order_line_t ON Product_t.Product_ID = Order_line_t.Product_ID
GROUP BY Product_t.Product_ID, Product_Description
UNION
SELECT Product_ID, Product_Description, 0
FROM Product_t
WHERE (EXISTS
  (SELECT * FROM Order_line_t
   WHERE (Order_line_t.Product_ID = Product_t.Product_ID))=FALSE);
```

Oracle Query:

```
SELECT Product_t.Product_ID, Product_Description, COUNT(*) as TimesOrdered
FROM Product_t, Order_line_t
WHERE Product_t.Product_ID = Order_line_t.Product_ID
GROUP BY Product_t.Product_ID, Product_Description
UNION
SELECT Product_ID, Product_Description, 0
FROM Product_t
WHERE NOT EXISTS
  (SELECT * FROM Order_line_t
   WHERE Order_line_t.Product_ID = Product_t.Product_ID);
```

20. Calculate the total raw material cost (label TotCost) for each product compared to its standard product price and display product ID, product description, standard price, and the total cost in the result:

Query:
```
SELECT P.Product_ID, Product_Description, P.Standard_Price,
       SUM(Goes_into_Quantity*R.Standard_Price)as TotCost
FROM Product_t as P, Uses as U, Raw_Material_t as R
WHERE P.Product_ID = U.Product_ID
  and U.Material_ID = R.Material_ID
GROUP BY P.Product_ID, Product_Description, P.Standard_Price;
```

Or here is another interesting approach using a derived table in the SELECT list:
SELECT Product_t.Product_Id, Product_Description, Product_t.Standard_Price,
    TotCost
FROM Product_t, (SELECT Product_Id,
    SUM(Standard_Price*Goes_into_Quantity) as TotCost
    FROM Uses, Raw_Material_t
    WHERE Uses.Material_Id = Raw_Material_t.Material_id
    GROUP BY Product_Id) as Cost_t
WHERE Product_t.Product_Id = Cost_t.Product_id;

21. For every order that has been received, display the order ID, the total dollar
    amount owed on that order (you’ll have to calculate this total from attributes in
    one or more tables; label this result Total_Due), and the amount received in
    payments on that order (assume there is only one payment made on each order).
    To make this query a little simpler, you don’t have to include those orders for
    which no payment has yet been received. List the results in decreasing order by
    the difference between total due and amount paid:

    Query:
    Just to help to verify the result, the following shows all 11 rows of the payment
    table, and as assumed, there is only one payment per order, but not all orders have
    payments. Note: Dates in this database sometimes change between editions of the
    associated textbook, so your results may vary in terms of dates:

<table>
<thead>
<tr>
<th>Pay_id</th>
<th>Order_Id</th>
<th>Date_c</th>
<th>Type_c</th>
<th>Amount</th>
<th>Comment_c</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>24</td>
<td>2004-03-10D</td>
<td>D</td>
<td>25</td>
<td>cash</td>
</tr>
<tr>
<td>5</td>
<td>32</td>
<td>2004-03-11D</td>
<td>D</td>
<td>3000</td>
<td>Cashiers Check</td>
</tr>
<tr>
<td>7</td>
<td>39</td>
<td>2004-03-11D</td>
<td>D</td>
<td>600</td>
<td>chk 1003</td>
</tr>
<tr>
<td>4</td>
<td>28</td>
<td>2004-03-10D</td>
<td>D</td>
<td>25</td>
<td>cash</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2004-03-01D</td>
<td>D</td>
<td>1000</td>
<td>chk101</td>
</tr>
<tr>
<td>9</td>
<td>51</td>
<td>2004-03-11D</td>
<td>D</td>
<td>150</td>
<td>cash</td>
</tr>
<tr>
<td>11</td>
<td>69</td>
<td>2004-03-11D</td>
<td>D</td>
<td>200</td>
<td>chk3001</td>
</tr>
<tr>
<td>10</td>
<td>54</td>
<td>2004-03-11D</td>
<td>D</td>
<td>2650</td>
<td>Check # 343</td>
</tr>
<tr>
<td>3</td>
<td>26</td>
<td>2004-03-10D</td>
<td>D</td>
<td>222</td>
<td>cash</td>
</tr>
<tr>
<td>6</td>
<td>34</td>
<td>2004-03-11D</td>
<td>D</td>
<td>575</td>
<td>Chk1201</td>
</tr>
<tr>
<td>8</td>
<td>48</td>
<td>2004-03-11D</td>
<td>D</td>
<td>1000</td>
<td>chk2301</td>
</tr>
</tbody>
</table>

Now the query:
SELECT Order_line_t.Order_Id,
    SUM(Ordered_Quantity*Standard_Price) AS Total_Due, Amount
FROM Order_line_t, Product_t, Payment_t
WHERE Order_line_t.Product_Id = Product_t.Product_Id
    and Order_line_t.Order_Id = Payment_t.Order_Id
GROUP BY Order_line_t.Order_Id, Amount
ORDER BY Total_Due - Amount DESC;
22. List the order number and order quantity for all customer orders for which the order quantity is greater than the average order quantity of that product: (Hint: This involves a correlated subquery.)

**Microsoft Access Query:**

```
SELECT order_11.Order_Id, order_11.Ordered_Quantity, order_11.Product_Id
FROM PRODUCT_t INNER JOIN Order_line_t AS order_11 ON
    PRODUCT_t.Product_Id = order_11.Product_Id
WHERE (order_11.Ordered_Quantity)>(SELECT avg(ordered_quantity) FROM order_line_t x1 WHERE x1.product_id = order_11.product_id)
AND ((PRODUCT_t.Product_Id)=[order_11].[Product_ID]));
```

**Oracle Query:**

```
FROM PRODUCT_t, Order_line_t order_11
WHERE PRODUCT_t.Product_ID = order_11.Product_ID and order_11.ordered_quantity >
(SELECT avg(ordered_quantity) FROM order_line_t x1 WHERE x1.product_id = order_11.product_id);
```

23. List the names and number of employees supervised (label this value HeadCount) for all the supervisors who supervise more than two employees:

**Query:**

```
SELECT S.Employee_Name, COUNT(E.Employee_ID) AS HeadCount
FROM Employee_t S, Employee_t E
WHERE S.Employee_Id = E.Employee_Supervisor
GROUP BY S.Employee_Name
HAVING HeadCount > 2;
```

**Result:**

<table>
<thead>
<tr>
<th>Employee_Name</th>
<th>HeadCount</th>
</tr>
</thead>
<tbody>
<tr>
<td>Robert Lewis</td>
<td>3</td>
</tr>
</tbody>
</table>

You can verify this by running a simpler query that computes the number of employees each employee supervises. From this query you would see that there are three supervisors (those who supervise anyone), and the other two supervisors supervise only one employee each.
24. List each salesperson who has sold computer desks and the number of units sold by each salesperson:

**Microsoft Access Query:**

```sql
SELECT order_t.SalesPerson_Id, salesperson_t.SalesPerson_Name,
       SUM(order_line_t.Ordered_Quantity) AS UnitsSold
FROM (order_line_t INNER JOIN (order_t INNER JOIN salesperson_t ON
                     order_t.SalesPerson_Id = salesperson_t.SalesPerson_Id) ON
                      order_line_t.Order_Id = order_t.Order_Id) INNER JOIN product_t ON
                      order_line_t.Product_Id = product_t.Product_Id
WHERE (((order_t.Order_Id)=[order_line_t].[order_id])
           AND ((order_line_t.Product_Id)=[product_t].[product_id])
           AND ((product_t.Product_Description)='Oak Computer Desk')
           AND ((order_t.SalesPerson_Id)=[salesperson_t].[salesperson_id]))
GROUP BY order_t.SalesPerson_Id, salesperson_t.SalesPerson_Name;
```

**Oracle Query:**

```sql
SELECT order_t.salesperson_id, salesperson_t.salesperson_name,
       SUM(ordered_quantity) as UnitsSold
FROM order_line_t,order_t,product_t,salesperson_t
WHERE order_t.order_id = order_line_t.order_id
     AND order_line_t.product_id  = product_t.product_id
     AND product_description = 'Oak Computer Desk'
     AND order_t.salesperson_id = salesperson_t.salesperson_id
GROUP BY order_t.salesperson_id,salesperson_name;
```

25. List the salesperson who has sold the most computer desks:

**Microsoft Access Query:**

It is easiest to create a subquery first. The first query we will call tsales:

```sql
SELECT salesperson_t.SalesPerson_ID,
       Sum(order_line_t.Ordered_Quantity) AS totsales
FROM ((order_line_t INNER JOIN order_t ON order_line_t.Order_ID =
          order_t.Order_ID) INNER JOIN product_t ON
             order_line_t.Product_ID = product_t.Product_ID) INNER JOIN
                (salesperson_t INNER JOIN does_business_in_t ON
                 salesperson_t.Territory_Id = does_business_in_t.Territory_Id) ON
                    order_t.Customer_ID = does_business_in_t.Customer_Id
WHERE (((order_t.Order_ID)=[order_line_t].[order_id])
       AND ((order_line_t.Product_ID)=[product_t].[product_id])
       AND ((product_t.Product_Description)='Oak Computer Desk')
       AND ((order_t.Customer_ID)=[does_business_in_t].[customer_id])
       AND ((salesperson_t.Territory_Id)=[does_business_in_t].[territory_id]))
GROUP BY salesperson_t.SalesPerson_ID;
```
Next, find the SalesPerson_ID for the salesperson who had the highest sales:

```
SELECT salesperson_id
FROM tsales
WHERE totsales = (select max(totsales) from tsales);
```

**Oracle Query:**

```
SELECT salesperson_t.salesperson_id, sum(ordered_quantity) as totsales
FROM order_line_t,order_t,product_t,does_business_in_t,salesperson_t
WHERE order_t.order_id = order_line_t.order_id
  and order_line_t.product_id = product_t.product_id
  and product_description = 'Oak Computer Desk'
  and order_t.customer_id = does_business_in_t.customer_id
  and salesperson_t.territory_id = does_business_in_t.territory_id
GROUP BY salesperson_t.salesperson_id;
```

Save the result as tsales and run to find the salesperson with the most computer desk sales:

```
SELECT salesperson_id
FROM tsales
WHERE totsales = (select max(totsales) from tsales);
```

26. List in alphabetical order the names of all employees (managers) who are now managing people with skill ID BS12. List each such manager’s name only once, even if that manager manages several people with this skill:

**Query:**

```
SELECT DISTINCT M.Employee_Name
FROM Employee_t AS M, Employee_t AS E, Employee_Skills_t AS ES
WHERE Skill_Id = 'BS12'
  and ES.Employee_Id = E.Employee_Id
  and E.Employee_Supervisor = M.Employee_Id
ORDER BY 1;
```

27. Display the salesperson name, product finish, and total quantity sold (label as Tot_Sales) for each finish by each salesperson:

**Query:**

```
SELECT DISTINCT Salesperson_Name, Product_Finish,
SUM(Ordered_Quantity) AS Tot_Sales
FROM Salesperson_t, Order_line_t, Product_t, Order_t
WHERE Salesperson_t.Salesperson_Id = Order_t.Salesperson_Id
  AND Order_t.Order_Id = Order_line_t.Order_Id
  AND Order_line_t.Product_Id = Product_t.Product_Id
GROUP BY Salesperson_Name, Product_Finish;
```
28. Display the customer ID, name, and order ID for all customer orders. For those customers who do not have any orders, include them in the display once by showing an order ID of 0:

**Query:**
```sql
SELECT Customer_t.Customer_Id, Customer_Name, Order_Id
FROM Customer_t, Order_t
WHERE Customer_t.Customer_Id = Order_t.Customer_Id
UNION
SELECT Customer_Id, Customer_Name, 0
FROM Customer_t
WHERE NOT EXISTS
  (SELECT * FROM Order_t
   WHERE Order_t.Customer_Id = Customer_t.Customer_Id);
```

Or replace last three lines above with:

WHERE Customer_Id NOT IN (SELECT Customer_Id FROM Order_t);

Or using an outer join:

```sql
SELECT c.Customer_Id, Customer_Name, ZEROIFNULL(Order_Id)
FROM Customer_t c LEFT OUTER JOIN Order_t o
ON c.Customer_Id = o.Customer_Id;
```

29. Write a query to list the number of products produced in each work center (label this result ‘TotalProducts’). If a work center does not produce any products, display the result with a total of 0:

**Query:**
```sql
SELECT Work_Center.Work_Center_Id, COUNT(Product_Id) as TotalProducts
FROM Work_Center LEFT OUTER JOIN Produced_in_t
ON Work_Center.Work_Center_Id = Produced_in_t.Work_Center_Id
GROUP BY Work_Center.Work_Center_Id;
```

30. The production manager at PVFC is concerned about support for purchased parts in products owned by customers. A simple analysis he wants done is to determine for each customer how many vendors are in the same state as that customer. Develop a list of all the PVFC customers by name with the number of vendors in the same state as that customer (label this computed result NumVendors):

**Query:**
```sql
SELECT Customer_Name, COUNT(Vendor_Id) AS NumVendors
FROM Customer_t C LEFT OUTER JOIN Vendor_t V
ON C.Customer_State = V.Vendor_State
GROUP BY Customer_Name;
```
31. Display the Employee ID and Employee Name for those employees who do not possess the skill Router. Display the results in order by EmployeeName:

**Query:**
SELECT Employee_Id, Employee_Name from Employee_t
 WHERE Employee_Id NOT IN
  (SELECT ES.Employee_Id from Employee_Skills_t as ES, Skill_t as S
   WHERE Skill_Description = 'Router'
   and ES.Skill_Id = S.Skill_Id)
 ORDER BY Employee_Name;

32. Show the customer ID and name for all the customers who have ordered both products with IDs 3 and 4 on the same order:

**Query:**
SELECT C.Customer_Id, Customer_Name
 FROM Customer_t C, Order_t O1, Order_line_t OL1
 WHERE C.Customer_Id = O1.Customer_Id
   and O1.Order_Id = OL1.Order_Id
   and OL1.Product_Id = 3
   and O1.Order_Id IN
    (SELECT Order_Id from Order_line_t OL2
     WHERE OL2.Product_Id = 4);

Or here is another interesting approach using derived tables (color for some parentheses helps to show the nesting of SELECTs):

SELECT distinct(Customer_Id), Customer_Name
 FROM Customer_t
 WHERE Customer_Id IN
  (SELECT Customer_Id
   FROM (SELECT p3.Order_Id
      FROM (SELECT Order_Id FROM Order_line_t where Product_Id = 3) as p3,
      (SELECT Order_Id from Order_line_t where Product_Id = 4) as p4
      WHERE p3.Order_Id = p4.Order_Id) as Orders, Order_t
   WHERE Orders.Order_Id = Order_t.Order_Id);

33. The head of marketing is interested in some opportunities for cross-selling of products. She thinks that the way to identify cross-selling opportunities is to know for each product how many other products are sold to the same customer on the same order. So, for example, a product that is bought by a customer in the same order with lots of other products is a better candidate for cross-selling than a product bought by itself.
a. To help the marketing manager, first list the IDs for all the products that have sold in total more than 20 units across all orders (these are popular products, which are the only products she wants to consider as triggers for potential cross-selling):

**Query for part a.:**

```sql
SELECT X.Product_Id from Order_line_t X
GROUP BY X.Product_Id
HAVING SUM(X.Ordered_Quantity) > 20;
```

**Result for part a.:**

<table>
<thead>
<tr>
<th>Product_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
</tbody>
</table>

b. Next, write a new query that lists all the IDs for the orders that include products that satisfy this first query along with the number of products on those orders (see result below for an example). Only orders with three or more products on them are of interest to the marketing manager. Write this query as general as possible to cover any answer to the first query, which might change over time. To clarify, if product X is one of the products that is in the answer set from part a., then in part b., we want to see the desired order information for orders that include product X:

**Query for part b.:**

```sql
SELECT Y.Order_Id, count(Y.Product_Id) as NumProductsOnOrder
FROM Order_line_t Y
WHERE Y.Order_Id IN
  (SELECT T.Order_Id FROM Order_line_t T
   where T.Product_Id IN
     (SELECT X.Product_Id FROM Order_line_t X
      GROUP BY X.Product_Id
      HAVING SUM(X.Ordered_Quantity) > 20))
GROUP BY Y.Order_Id
HAVING NumProductsOnOrder >= 3;
```

**Result for part b.:**

<table>
<thead>
<tr>
<th>Order_ID</th>
<th>NumProductsOnOrder</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>

c. Finally, the marketing manager needs to know what are the (other) products sold on the orders that are in the result for part b. (again, write this query for the general, not specific result, to the query in part b.). These are products that are sold, for example, with product X from part a., and these are the ones that
if people buy that product, we would want to try to cross-sell them product X because history says they are likely to buy it along with whatever else they are buying. Write a query to identify these other products by ID and description. It is okay to include product X in your result (i.e., you don’t need to exclude the products that were in the result of part a.):

**Query for part c.:**

```sql
SELECT DISTINCT P.Product_Id, P.Product_Description
FROM Order_line_t as O, Product_t as P
WHERE P.Product_Id = O.Product_Id
and O.Order_Id IN
  (SELECT Y.Order_Id
   FROM Order_line_t Y
   WHERE Y.Order_Id IN
     (SELECT T.Order_Id FROM Order_line_t T
      WHERE T.Product_Id IN
        (SELECT X.Product_Id FROM Order_line_t X
         GROUP BY X.Product_Id
         HAVING SUM(X.Ordered_Quantity) > 20))
   GROUP BY Y.Order_Id
HAVING COUNT(Y.Product_Id) >= 3);
```

**Result for part c.:**

<table>
<thead>
<tr>
<th>Product_ID</th>
<th>Product_Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Birch coffee table</td>
</tr>
<tr>
<td>3</td>
<td>Oak computer desk</td>
</tr>
<tr>
<td>4</td>
<td>Entertainment center</td>
</tr>
<tr>
<td>5</td>
<td>Writer’s desk</td>
</tr>
<tr>
<td>6</td>
<td>8-Drawer dresser</td>
</tr>
<tr>
<td>8</td>
<td>48&quot; bookcase</td>
</tr>
<tr>
<td>10</td>
<td>96&quot; bookcase</td>
</tr>
<tr>
<td>14</td>
<td>Writer’s Desk</td>
</tr>
</tbody>
</table>

34. For each product display in ascending order by product ID the product ID and description along with the customer ID and name for the customer who has bought the most of that product; also show the total quantity ordered by that customer (who has bought the most of that product). Use a correlated subquery:

**Query:**

```sql
SELECT P1.Product_Id, Product_Description, C1.Customer_Id, Customer_Name,
       SUM(OL1.Ordered_Quantity) as TotOrdered
FROM Customer_t as C1, Product_t as P1, Order_line_t as OL1, Order_t as O1
WHERE C1.Customer_Id = O1.Customer_Id
GROUP BY P1.Product_Id, Product_Description, C1.Customer_Id, Customer_Name
ORDER BY P1.Product_Id ASC;
```

Published by [www.emagazine.org](http://www.emagazine.org)
and O1.Order_Id = OL1.Order_Id
and OL1.Product_Id = P1.Product_Id
GROUP BY P1.Product_Id, Product_Description, C1.Customer_Id,
Customer_Name
HAVING TotOrdered >= ALL
(SELECT SUM(OL2.Ordered_Quantity)
  FROM Order_line_t as OL2, Order_t as O2
  WHERE OL2.Product_Id = P1.Product_Id
  and OL2.Order_Id = O2.Order_Id
  and O2.Customer_Id <> C1.Customer_Id
GROUP BY O2.Customer_Id)
ORDER BY P1.Product_Id;

Result:

<table>
<thead>
<tr>
<th>Product_Id</th>
<th>Product_Description</th>
<th>Customer_Id</th>
<th>Customer_Name</th>
<th>TotOrdered</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Cherry end table</td>
<td>4</td>
<td>Eastern Furniture</td>
<td>9</td>
</tr>
<tr>
<td>2</td>
<td>Birch Coffee table</td>
<td>4</td>
<td>Eastern Furniture</td>
<td>26</td>
</tr>
<tr>
<td>3</td>
<td>Oak Computer desk</td>
<td>3</td>
<td>Home Furnishings</td>
<td>12</td>
</tr>
<tr>
<td>4</td>
<td>Entertainment center</td>
<td>16</td>
<td>ABC Furniture Co.</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>Writer’s desk</td>
<td>15</td>
<td>Janet’s Collection</td>
<td>10</td>
</tr>
<tr>
<td>6</td>
<td>68-Drawer dresser</td>
<td>4</td>
<td>Eastern Furniture</td>
<td>4</td>
</tr>
<tr>
<td>7</td>
<td>48&quot; bookcase</td>
<td>4</td>
<td>Eastern Furniture</td>
<td>4</td>
</tr>
<tr>
<td>8</td>
<td>848&quot; bookcase</td>
<td>3</td>
<td>Home Furnishings</td>
<td>2</td>
</tr>
<tr>
<td>10</td>
<td>1096&quot; bookcase</td>
<td>4</td>
<td>Eastern Furniture</td>
<td>9</td>
</tr>
<tr>
<td>13</td>
<td>Nightstand</td>
<td>13</td>
<td>Ikards</td>
<td>2</td>
</tr>
<tr>
<td>14</td>
<td>Writer’s desk</td>
<td>15</td>
<td>Janet’s Collection</td>
<td>10</td>
</tr>
<tr>
<td>17</td>
<td>High back leather chair</td>
<td>1</td>
<td>Contemporary Casuals</td>
<td>5</td>
</tr>
<tr>
<td>20</td>
<td>Amiore</td>
<td>8</td>
<td>Dunkins Furniture</td>
<td>1</td>
</tr>
</tbody>
</table>
35. Display in product ID order the product ID and total amount ordered of that product by the customer who has bought the most of that product; use a derived table in a FROM clause to answer this query. Note that result of this query is a subset (first and last columns) of the prior query result.

**Query:**

```sql
SELECT F1.Product_Id, MAX(F1.ProdCustTotal) as TotOrdered
FROM
  (SELECT Product_Id, Customer_Id, sum(Ordered_Quantity) as ProdCustTotal
   FROM Order_line_t as OL2, Order_t as O2
   WHERE OL2.Order_Id = O2.Order_Id
   GROUP BY Product_Id, Customer_Id) as F1
GROUP BY F1.Product_Id
ORDER BY F1.Product_Id;
```

**Result:**

<table>
<thead>
<tr>
<th>Product_ID</th>
<th>TotOrdered</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>9</td>
</tr>
<tr>
<td>2</td>
<td>26</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>10</td>
<td>9</td>
</tr>
<tr>
<td>13</td>
<td>2</td>
</tr>
<tr>
<td>14</td>
<td>10</td>
</tr>
<tr>
<td>17</td>
<td>5</td>
</tr>
<tr>
<td>20</td>
<td>1</td>
</tr>
</tbody>
</table>

**Suggestions for Field Exercises**

1. This question is self-explanatory.

2. Three differences that you may want to help the students discover:
   - Primary and foreign keys are specified with the CREATE TABLE or ALTER TABLE commands in SQL*PLUS. MS Access uses a graphical interface to create tables and to establish primary and foreign keys.
   - MS Access SQL does not include the CREATE VIEW or DROP VIEW commands. Instead, the query that would be used as part of the CREATE VIEW syntax is saved as a query, which can be accessed later through new select statements by simply specifying the query name.
• MS Access has a keyword, DISTINCTROW, which is not found in other DBMS SQL implementations.

Five similarities between Oracle SQL and MS Access SQL:
• Both use the semicolon to mark the end of a statement and cause it to execute.
• Basic SQL syntax is very similar, using the keywords SELECT, FROM, WHERE, GROUP BY, ORDER BY, and HAVING in the same fashion.
• Both use brackets to specify the order of expression evaluation.
• Subqueries and correlated subqueries can be written in both.
• Both are insensitive to spacing and line breaks within a statement.

Project Case

Case Questions

1. The student will need to identify the capabilities of the DBMS.

2. DDL Triggers can be written to update an audit log whenever records are added, deleted or updated in any table. This audit log could contain the table name, fields affected, user, date, and time.

Case Exercises

1. Info from two tables: List all the details of all the visits of a patient:

   Query:
   ```sql
   SELECT patient.name, visit.visit_date, visit.visit_time, visit.visit_reason, visit.new_symptoms, visit.pain_level
   FROM patient INNER JOIN visit
   ON patient.patient_no = visit.patient_no
   WHERE patient.name = "John Smith";
   ```

b. Query with subquery syntax: List all patients who reported pain that exceeded the average pain for all visits:

   Query:
   ```sql
   SELECT patient.name
   FROM patient INNER JOIN visit
   ON patient.patient_no = visit.patient_no
   WHERE pain_level > (SELECT avg(pain_level) FROM visit);
   ```
c. Result table to produce report: listing of patients assigned to specific social worker:

**Query:**

```
SELECT patient_name
from patient
where social_worker = "Jane Klein";
```

2. Students should be encouraged to add data and adjust the data model of their prototype database in order to improve it and to completely test the queries they are writing. Student answers will vary based on how their prototype is designed and implemented.

a. For a given physician, which treatments has that physician performed on each patient who was referred by that physician to the hospital?

**Query:**

```
Select distinct person_t.person_first_name,
person_t.person_last_name, treatment_name
from treatment_t, performs_t, patient_t, physician_t, person_t
where performs_t.patient_id = patient_t.patient_id
and performs_t.treatment_id = treatment_t.treatment_id
and physician_t.physician_id = performs_t.physician_id
and patient_t.physician_id = physician_t.physician_id
and physician_t.person_id = person_t.person_id;
```

b. For query in part a., also include physicians who have not referred patients to the hospital:

**Query:**

```
Select distinct person_t.person_first_name,
person_t.person_last_name, treatment_name
from treatment_t, performs_t, patient_t, physician_t, person_t
where performs_t.patient_id = patient_t.patient_id
and performs_t.treatment_id = treatment_t.treatment_id
and physician_t.physician_id = performs_t.physician_id
and patient_t.physician_id = physician_t.physician_id
and physician_t.person_id = person_t.person_id
union
Select person_t.person_first_name, person_t.person_last_name, 'None'
from person_t
where person_t.person_id = physician_t.person_id
and physician_t.physician_id not in (select physician_id from patient_t);
```
c. For each patient, what is the average number of treatments performed on him or her by each physician who has treated that patient?

This query could be a bit confusing to students. Make sure that you explain to students that they should find the total treatments done for each patient by physician. Once this is done, the next step is to compute an average of the total number of treatments done by all physicians for a patient. For example, if we had the following performs_t table (here, I only show three fields):

<table>
<thead>
<tr>
<th>Patient_ID</th>
<th>Physician_ID</th>
<th>Treatment_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>321</td>
<td>45</td>
</tr>
<tr>
<td>12</td>
<td>321</td>
<td>56</td>
</tr>
<tr>
<td>12</td>
<td>421</td>
<td>5</td>
</tr>
<tr>
<td>12</td>
<td>456</td>
<td>23</td>
</tr>
<tr>
<td>12</td>
<td>456</td>
<td>56</td>
</tr>
<tr>
<td>13</td>
<td>321</td>
<td>32</td>
</tr>
<tr>
<td>13</td>
<td>342</td>
<td>45</td>
</tr>
</tbody>
</table>

We would end up with a derived table (treat_counts in the query following) as follows:

<table>
<thead>
<tr>
<th>Patient_ID</th>
<th>Physician_ID</th>
<th>Total_Treatments</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>321</td>
<td>2</td>
</tr>
<tr>
<td>12</td>
<td>421</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>456</td>
<td>2</td>
</tr>
<tr>
<td>13</td>
<td>321</td>
<td>1</td>
</tr>
<tr>
<td>13</td>
<td>342</td>
<td>1</td>
</tr>
</tbody>
</table>

We would then calculate the average from the treat_counts table as follows:

<table>
<thead>
<tr>
<th>Patient_ID</th>
<th>Average Treatments</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>1.66</td>
</tr>
<tr>
<td>13</td>
<td>1</td>
</tr>
</tbody>
</table>
Query in Oracle:

```
SELECT person_t.Person_Last_Name,
       Avg(treat_count.total_treatments) AS AvgOftotal_treatments
FROM person_t, patient_t,
     (SELECT performs_t.patient_id, performs_t.physician_id,
         count(*)  as total_treatments
      FROM performs_t
      GROUP BY performs_t.patient_id,performs_t.physician_id)
     AS treat_count
WHERE person_t.Person_ID=patient_t.person_id  AND
  patient_t.Patient_id=treat_count.patient_id
GROUP BY person_t.Person_Last_Name;
```

In Access, we could either use the Oracle query (shown previously) in SQL view or first create the total_treatments derived table as a query:

```
SELECT PERFORMS_t.Patient_id, PERFORMS_t.Physician_ID,
       Count(PERFORMS_t.Treatment_ID) AS CountOfTreatment_ID
FROM PERFORMS_t
GROUP BY PERFORMS_t.Patient_id, PERFORMS_t.Physician_ID;
```

We can now use this query for our final query:

```
SELECT PERSON_t.Person_Last_Name,
       Avg(treat_count.[Total Treatments])  AS [AvgOfTotal Treatments]
FROM (PERSON_t INNER JOIN PATIENT_t ON
       PERSON_t.Person_ID =  PATIENT_t.Person_ID) INNER JOIN treat_count
ON PATIENT_t.Patient_ID =  treat_count.Patient_id
GROUP BY PERSON_t.Person_Last_Name;
```

d. List all patients who have received no treatments:

Oracle Query:

```
select person_t.person_last_name,person_t.person_first_name
from person_t,patient_t
where person_t.person_id = patient_t.person_id
and patient_id not in (select patient_id from performs_t);
```

MS Access Query:

```
SELECT PERSON_t.Person_Last_Name, PERSON_t.Person_First_Name
FROM PERSON_t INNER JOIN
      PATIENT_t ON PERSON_t.Person_ID =  PATIENT_t.Person_ID
where patient_id not in (select patient_id from performs_t)
```
e. For each nurse-in-charge, what is the total number of hours worked by all employees who work in the care center that that nurse supervises?

**Oracle Query:**
```
select in_charge, sum(hours_worked)
from care_center_t, assignment_t
where care_center_t.carecenter_id = assignment_t.carecenter_id
group by in_charge;
```

**MS Access Query:**
```
SELECT CARE_CENTER_t.In_Charge, Sum(Assignment_t.Hours) AS SumOfHours
FROM CARE_CENTER_t INNER JOIN Assignment_t ON
CARE_CENTER_t.CareCenter_ID = Assignment_t.CareCenter_ID
GROUP BY CARE_CENTER_t.In_Charge;
```

f. Which technicians have more than one skill? Which technicians have no skills listed?

Unless we create a separate table with skills or change the primary key of the technician_t table, currently, each technician can have only one skill.

To find technicians with no skills, we would look for any technicians without a record in the technician_t table:

**Oracle Query:**
```
Select person_t.person_last_name, person_t.person_first_name
from person_t, employee_t
where person_t.person_id = employee_t.employee_id
and entitytypes = 'Technician'
and employee_id not in (select technician_id from technician_t);
```

**MS Access Query:**
```
SELECT PERSON_t.Person_Last_Name, PERSON_t.Person_First_Name
FROM PERSON_t INNER JOIN EMPLOYEE_t ON
PERSON_t.Person_ID = EMPLOYEE_t.Employee_ID
where employeeetype = 'Technician' and
employee_id not in (select technician_id from technician_t)
```

g. Determine whether any outpatients were accidentally assigned to resident beds:

**Oracle Query:**
```
Select count(*) From patient_t Where admission_type = 'Outpatient' and
bed_id is not null;
```
h. Determine which item is consumed most:

**Oracle Query:****

```
SELECT CONSUMES_t.Item_ID, Count(*)
FROM CONSUMES_t
GROUP BY CONSUMES_t.Item_ID;
```

**MS Access Query:****

```
SELECT top 1 CONSUMED_t.Item_ID, Count(*)
FROM CONSUMED_t
GROUP BY CONSUMED_t.Item_ID;
```

i. Determine which physicians prescribe the most expensive item:

In the MVCH database on the Teradata Web site, there is no table that links the physicians to items unless we assume that the admitting physician is the physician prescribing all items. In this case, we need to first determine the most expensive item, then use that result table in a subsequent query:

**Oracle Query:****

```
select item_id
from item_t
where item_cost = (select max(item_cost) from item_t);
```

We can use the previous query as a derived table (we will call it top_item).

```
SELECT PERSON_t.Person_Last_Name,
       PERSON_t.Person_First_Name
FROM (select item_id from item_t where item_cost =
       (select max(item_cost) from item_t))
       top_item, PERSON_t, PHYSICIAN_t, PATIENT_t,
       CONSUMES_t
where person_t.person_id = physician_t.person_id
    and physician_t.physician_id = patient_t.physician_id
    and consumes_t.patient_id = patient_t.patient_id
    and consumes_t.item_id = top_item.item_id
```
MS Access Query:

We can create a query and call it top_item:

```sql
SELECT TOP 1 items_id, item_cost
FROM item_t;
```

We can then use the query created above to determine physicians who prescribed the most expensive items.

```sql
SELECT PERSON_t.Person_Last_Name, PERSON_t.Person_First_Name
FROM top_item INNER JOIN (PERSON_t INNER JOIN PHYSICIAN_t
ON PERSON_t.Person_ID = PHYSICIAN_t.Person_ID) INNER JOIN
(PATIENT_t INNER JOIN CONSUMED_t ON PATIENT_t.Patient_ID =
CONSUMED_t.Patient_ID) ON PHYSICIAN_t.Physician_ID =
PATIENT_t.Physician_ID) ON top_item.items_id =
CONSUMED_t.Item_ID;
```

j. Return a result table that could be used to produce a hospital report, such as nursing staff assigned to each care center:

Oracle Query:

```sql
SELECT CARE_CENTER_t.CareCenter_ID,
PERSON_t.Person_First_Name, PERSON_t.Person_Last_Name
FROM PERSON_t,EMPLOYEE_t,CARE_CENTER_t,NURSE_t
where CARE_CENTER_t.CareCenter_ID = NURSE_t.CareCenter_ID
and EMPLOYEE_t.Employee_ID = NURSE_t.Employee_ID
and PERSON_t.Person_ID = EMPLOYEE_t.Employee_ID;
```

MS Access Query:

```sql
SELECT CARE_CENTER_t.CareCenter_ID,
PERSON_t.Person_First_Name, PERSON_t.Person_Last_Name
FROM PERSON_t INNER JOIN (EMPLOYEE_t INNER JOIN
(CARE_CENTER_t INNER JOIN NURSE_t ON
CARE_CENTER_t.CareCenter_ID = NURSE_t.CareCenter_ID) ON
EMPLOYEE_t.Employee_ID = NURSE_t.Employee_ID) ON
PERSON_t.Person_ID = EMPLOYEE_t.Employee_ID;
```
k. Use the UNION statement to provide a combined listing of care center names and their locations as well as laboratories and their location. This list should be sorted by location in ascending order. (You should use aliases to rename the fields in this query):

Query:
```sql
select carecenter_name as name, carecenter_location as location
from care_center_t
union all
select laboratory_id as name, laboratory_location as location
from laboratory_t;
```

Note: The ORDER BY clause cannot be used with a literal field.

Project Assignments

P1.

a. Nurses assigned to each care_center:

Query:
```sql
SELECT CC.CC_Unit_Name, N.Nurse_ID
FROM CC_Assignment CCA, NURSE N, CARE_CENTER CC
WHERE N.Nurse_ID = CCA.Nurse_ID AND
CC.CC_Unit_Name = CCA.CC_Unit_Name
ORDER by CC.CC_Unit_Name;
```

b. Five most common diagnoses:

Query:
```sql
SELECT top 5 DIAGNOSIS.DIAGNOSIS_NAME,
COUNT(PHYSICIAN_DX.DIAGNOSIS_CODE) AS CountOfDXCode
FROM DIAGNOSIS INNER JOIN PHYSICIAN_DX ON
DIAGNOSIS.DIAGNOSIS_CODE =
PHYSICIAN_DX.DIAGNOSIS_CODE
GROUP BY DIAGNOSIS.DIAGNOSIS_NAME;
```

c. Items consumed by each patient:

Query:
```sql
SELECT PERSON_NAME, I.ITEM_NO, ITEM_DESC
FROM PERSON PER, PATIENT PAT, ITEM I, ITEM_CONSUMPTION IC
WHERE PAT.PATIENT_ID = PER.PERSON_ID AND
PAT.PATIENT_ID = IC.PATIENT_ID AND
IC.ITEM_NO = I.ITEM_NO
ORDER BY PERSON_NAME;
```
d. Number of items provided by vendors:

**Query:**

```sql
SELECT VENDOR_NAME, COUNT(INV.VENDOR_ID) AS NBRVENITEMS
FROM VENDOR V, INVENTORY INV, ITEM I
WHERE V.VENDOR_ID = INV.VENDOR_ID AND
      I.ITEM_NO = INV.ITEM_NO;
```

e. Number of admissions per physician:

**Query:**

```sql
SELECT PERSON_NAME, COUNT(PAT.ADMIT_PHYS) AS NBRADMITS
FROM PERSON P, PATIENT PAT, PHYSICIAN DR
WHERE DR.PHYSICIAN_ID = P.PERSON_ID AND
      PAT.ADMIT_PHYS = DR.PHYSICIAN_ID
ORDER BY PERSON_NAME
GROUP BY PERSON_NAME;
```

P2. Create one DDL trigger for the database:

If we were to add a quantity-on-hand column to item, then whenever a row would be added to consume, we would reduce the quantity on hand in item by that amount.

**SQL:**

```sql
Create or replace trigger adjust_qoh
After insert on consume
For each row
Update item
Set qoh = qoh - :new.quantity
Where item_no = :new.item_no;
```
Chapter 9 The Client/Server Database Environment

Chapter Overview

This chapter describes all of the components of a client/server system and defines all the terms that students need to master to function within the client/server environment. This chapter emphasizes the impact of client/server technology and how to best take advantage of the resources and choices available. This chapter also addresses why some client/server implementations fail or are not as effective as they should be; it outlines ways to avoid these scenarios.

From here, we explore the use of ODBC (Open Database Connectivity) and JDBC (Java Database Connectivity) and their use in connecting to external databases.

Chapter Objectives

Specific student learning objectives are included at the beginning of the chapter. From an instructor’s point of view, the objectives of this chapter are:

1. To impart to the student a comprehensive view of the possibilities of client/server computing and the advantages and disadvantages of different architectural structures.
2. To provide a framework for discussion of tiered architectures and the vocabulary that goes along with it.
3. To illustrate the uses of application partitioning across the different segments of the network and the client/server system; form a basis for understanding the decision-making process in implementing various configurations.
4. To provide the student with the conceptual underpinnings of connections to remote databases.

Key Terms

<table>
<thead>
<tr>
<th>Application partitioning</th>
<th>Fat client</th>
<th>Three-tier architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application program interface (API)</td>
<td>File server</td>
<td>Middleware</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Open database connectivity (ODBC) standard</td>
</tr>
<tr>
<td>Client/server systems</td>
<td>Stored procedure</td>
<td></td>
</tr>
<tr>
<td>Database server</td>
<td>Thin-client</td>
<td></td>
</tr>
</tbody>
</table>

Classroom Ideas

1. Create a network structure of servers and clients. Give students a simulated design of a business with a file server architecture and estimated number of transactions each day. Have the students map out the traffic between pieces of the system and observe the system load in a normal business day. Ask the students to reassess their observations if the business has a 10 percent growth factor. Now have them follow through the same transactions on a
client/server system: first a two-tier and then a three-tier. Observe again with the 10 percent growth factor. Ask for their observations for each of the three architectures. Make sure that they try different application partitioning for the two- and three-tiered operations. Request a recommendation for the most advantageous approach to use. Students should defend their answers.

2. Have the students take a section of code and rewrite it for remote access using JDBC.

3. Have the students write a query and time its execution. Now have them save it as a stored procedure and time it.

Answers to Review Questions

1. Define each of the following key terms:
   a. Application partitioning Assigning portions of application code to client or server partitions in order to achieve better performance and interoperability.
   b. Application program interface (API) Type of software that allows a specific front-end program development platform to communicate with a particular back-end database server, even when the front end and back end were not built to be compatible.
   c. Client/server system A common solution for hardware and software organization that implements the idea of distributed computing. Many client/server environments use a local area network (LAN) to support a network of personal computers—each with its own storage—that are also able to share common devices (such as a hard disk or printer) and software (such as a DBMS) attached to the LAN. Several client/server architectures have evolved; they can be distinguished by the distribution of application logic components across clients and servers.
   d. Fat client A client PC that is responsible for processing, including presentation logic, and extensive application logic and business rules logic. A thin client would be responsible for much less processing.
   e. File server A device that manages file operations and is shared by each of the client PCs that are attached to the LAN.
   f. Middleware A type of software that allows an application to interoperate with other software without requiring the user to understand and code the low-level operations.
   g. Stored procedure A module of code, usually written in a proprietary language such as Oracle’s PL/SQL or Sybase’s Transact-SQL. It implements application logic or a business rule and is stored on the server, where it runs when called.
   h. Three-tier architecture A client/server configuration that includes three layers: a client layer and two server layers. Although the nature of the server layers differs, common configurations include an application server or a transaction server.
   i. JDBC (Java Database Connectivity) JDBC enables Java programs to execute SQL statements and connect to database servers. It is similar to ODBC but is designed specifically for Java applications.
   j. Remote procedure call (RPC) A client/server infrastructure that allows an application to be distributed over many platforms. The programmer does not have to know the details of the operating system or network interfaces, but instead calls procedures on distributed systems using function calls.
2.  
   e. client/server system  
   g. application program interface (API)  
   a. fat client  
   d. database server  
   f. file server  
   c. middleware  
   h. three-tiered architecture  
   b. thin client  

3. With their ability to accept an open systems approach, client/server architectures have offered businesses opportunities to better fit their computer systems to their needs. Their major advantages are:  
   a. Functionality can be delivered in stages to the end-users. Thus, it arrives more quickly as the first pieces of the project are deployed.  
   b. The GUI interfaces common in client/server environments encourage users to utilize the applications’ functionality.  
   c. The flexibility and scalability of client/server solutions facilitate business process re-engineering.  
   d. More processing can be performed close to the source of data being processed, thereby improving response times and reducing network traffic.  
   e. Client/server architectures allow the development of Web-enabled applications, facilitating the ability of organizations to communicate effectively internally and to conduct external business over the Internet.  

4. Contrast the following terms:  
   a. File server; database server; three-tier architecture  
      File servers manage file operations and are shared by each client PC that is attached to their LAN. The database server architecture makes the client manage the user interface while the database server manages database storage and access. An important distinction is that in response to a request for data, a file server transmits an entire file of data, while a database server transmits only selected data specified in the request. Three-tier architectures include another server in addition to the client and database server layers; they allow application code to be stored on the additional server.  
   b. Client/server computing; mainframe computing  
      Client/server architectures implement distributed approaches to computing and open systems development, while mainframe systems are a centralized approach to computing.  
   c. Fat client; thin client  
      A distinction among client capabilities that is based on processing capabilities. A fat client is responsible for more processing—including presentation logic, extensive application logic, and business rules logic—while a thin client is responsible for much less processing.  
   d. ODBC; JDBC  
      While ODBC is a language independent application programming interface for accessing and processing SQL databases, JDBC is language specific and designed for JAVA.
5. Limitations of file servers:
   a. Create a heavy network load: The server does very little work, the client is busy with extensive data manipulation, and the network is transferring large blocks of data.
   b. Require a full version of the DBMS on each client: This means that there is less room for an application program on the client PC or a PC with more RAM is needed. Further, because the client workstation does most of the work, each client must be rather powerful to provide a suitable response time.
   c. Require complex programming in order to manage shared database integrity: In addition, each application program must recognize items such as locks and take care to initiate the proper locks. Thus, application programmers must be rather sophisticated to understand various subtle conditions that can arise in a multiple-user database environment.
   d. Programming is more complex because you have to program each application with the proper concurrency, recovery, and security controls.

6. Some advantages of database servers:
   a. Reduced network traffic: With this architecture, the database is stored on the server, not on the client. Because less data are sent across the LAN, the communication load on the network is reduced.
   b. Reduced processing power required for each client: With this architecture, only the database server requires processing power adequate to handle the database. The client computers do not require additional RAM or processing power to work with the database. Further, the database server can be tuned to optimize database processing performance.
   c. Centralized user authorization, integrity checking, data dictionary maintenance, and query and update processing on the database server: This is possible through the use of stored procedures (modules of code that implement application logic), which are included on the database server. For example, data integrity can be improved as multiple applications access the same stored procedure.

Some disadvantages of database servers:
   a. Writing stored procedures take more time than using tools such as Visual Basic to create an application.
   b. Stored procedures’ proprietary nature reduces their portability and may make it difficult to change DBMSs without having to rewrite the stored procedures.
   c. Each client must be loaded with the applications that will be used at that location. Upgrades to an application will require that each client be upgraded separately.

7. Some advantages of three-tier and n-tier architectures (Thompson, 1997):
   a. Scalability Three-tier architectures are more scalable than two-tier architectures. For example, the middle tier can be used to reduce the load on a database server by using a TP monitor to reduce the number of connections to a server.
   b. Technological flexibility It is easier to change DBMS engines with a three-tier architecture, though triggers and stored procedures will need to be rewritten. The middle tier can even be moved to a different platform. Simplified presentation services make it easier to implement various desired interfaces such as Web browsers or kiosks.
c. **Lower long-term costs** Use of off-the-shelf components or services in the middle tier can reduce costs as can substitution of modules within an application.
d. **Better match of systems to business needs** New modules can be built to support specific business needs rather than building more general, complete applications.
e. **Improved customer service** Multiple interfaces can access the same business processes.
f. **Competitive advantage** The ability to react to business changes quickly (by changing small modules of code rather than entire applications) can be used to gain a competitive advantage.
g. **Reduced risk** Again, the ability to implement small modules of code quickly and combine them with code purchased from vendors limits the risk assumed with a large-scale development project.

Some disadvantages of three-tier and n-tier architectures:
a. **High short-term costs** Implementing a three-tier architecture requires that the presentation component be split from the process component. Accomplishing this split requires more programming in a 3GL language (such as C) than is required in implementing a two-tier architecture.
b. **Incompatible standards** There are few proposed standards for TP monitors as yet. There are several competing standards proposed for distributed objects, but it is not yet clear which standard will prevail.

8. **Using application partitioning to tailor applications:**

Partitioning applications gives developers the opportunity to write application code they can later place either on a client workstation or on a server, depending on which location will give the best performance. This flexibility allows the developers to tailor each application more effectively.

9. **Six categories of middleware:**
a. **Asynchronous remote procedure calls (RPC)** The client requests services, but does not wait for a response. It will typically establish a point-to-point connection with the server and perform other processing while it waits for the response. If the connection is lost, the client must re-establish the connection and send the request again. This type of middleware has high scalability but low recoverability.
b. **Publish/subscribe** This type of middleware monitors activity and pushes information to subscribers. It is asynchronous: the clients, or subscribers, perform other activities between notifications from the server. The subscribers notify the publisher of information that they wish to receive, and when an event occurs that contains such information, it is sent to the subscriber who can then elect to receive the information or not. For example, you can supply the electronic bookstore www.amazon.com with keywords of topics that interest you. Whenever Amazon adds a book title that is keyword coded with one of your topics, information about that title will be automatically forwarded to you for consideration. This type of middleware is very useful for monitoring situations where actions need to be taken when particular events occur.
c. **Message-oriented middleware (MOM)** MOM is also asynchronous software, sending
messages that are collected and stored until they are acted upon while the client continues with other processing. Workflow applications such as insurance policy applications, which often involve several processing steps, can benefit from MOM. The queue where the requests are stored can be journalized, thus providing some recoverability.

d. **Object request brokers (ORBs)** This type of middleware makes it possible for applications to send objects and request services in an object-oriented system. The ORB tracks the location of and routes requests to each object. Current ORBs are synchronous, but asynchronous ORBs are being developed.

e. **SQL-oriented data access** Connecting applications to databases over networks is achieved by using SQL-oriented data access middleware. This middleware also has the capability to translate generic SQL into the SQL specific to the database. Database vendors and companies that have developed multidatabase access middleware dominate this middleware segment.

f. **Synchronous RPC** A distributed program using synchronous RPC may call services available on different computers. This middleware makes it possible to establish this facility without undertaking the detailed coding usually necessary to write an RPC. Examples would include Microsoft Transaction Server and IBM’s CICS.

10. **Effects of the Web on data distribution patterns:**

   The thin clients represented by the browser interfaces move application logic to more centralized servers. The Internet backbone carries application architectures in addition to the messaging and mailing it now supports, in place of LANs or WANs. Thus, on the Web, tasks will have to be implemented in modules that can be controlled asynchronously (Panttaja, 1996). While application development and systems control may become more centralized, businesses are also deploying systems that reach outside their organization: to business partners, customers, and suppliers.

11. **The purpose of the ODBC specification is to enable an application program, written in any language, to access a relational database using a common application-programming interface (API). It is important because it allows for the development of applications that access different database products easily.**

12. **There are several key components to ensuring success:**
   a. Developing a plan for distribution, including an analysis of how the distributed system will interact with other systems.
   b. An analysis of network performance, including projections of load placed on the system by the client/server application: once an analysis has been performed, a plan can be implemented to upgrade the network to handle the increased load.
   c. Tuning of the production system once in place.
   d. Implementation of troubleshooting procedures.
   e. Code management during the development phase.

13. **Answers may vary, depending upon the information that students acquire in their research. The following is a sample answer:**

   Published by [www.emagazine.org](http://www.emagazine.org)
Advantages:
   a. Useful for integration of legacy systems because innovation of objects in different languages is very easy.
   b. Easy to extend to new languages.
   c. Level of abstraction provided by IDL layer.
   d. ORBs can communicate.
   e. Standards developed by international consortium.

Disadvantages:
   a. Limited developer acceptance.
   b. Difficulty with interoperability between ORBs.
   c. Slow performance.

14. Student exercise.

Answers to Problems and Exercises

1. Business and technology characteristics to consider when reaching a client/server adoption decision:
   a. **Accurate business problem analysis** Just as is the case with other computing architectures, it is critical to develop a sound application design and architecture for a new client/server system. Accurately define the scope of the problem and do an accurate requirement determination; use that information to select the technology.
   b. **Detailed architecture analysis** It is also important to specify the details of the client/server architecture. Building a client/server solution involves connecting many components that may not work together easily. One of the often touted advantages of client/server computing, the ability to accept an open systems approach, can be very detrimental if the heterogeneous components chosen are difficult to connect. In addition to specifying the client workstations, server(s), network, and DBMS, analysts should also specify network infrastructure, the middleware layer, and the application development tools to be used. At each juncture, analysts should take steps to assure that the tools will connect with the middleware, database, network, and so forth.
   c. **Avoiding tool-driven architectures** As above, project requirements should be determined before software tools are chosen, not the reverse. When a tool is chosen first and then applied to the problem, one runs the risk of a poor fit between problem and tool. Tools used in this manner are more likely to have been chosen based on an emotional appeal rather than on the appropriate functionality of the tool.
   d. **Achieving appropriate scalability** Moving to a multi-tier solution allows client/server systems to scale to any number of users and handle diverse processing loads. However, multi-tiered solutions are significantly more expensive and difficult to build. The tools to develop a multi-tier environment are still limited, too. Architects should avoid moving to a multi-tier solution when it is not really needed. Usually, multi-tier makes sense in environments of more than 100 concurrent users, high-volume OLTP systems, or for real-time processing. Smaller, less intense environments can frequently run more efficiently on traditional two-tier systems, especially if triggers and procedures are used.
to manage the processing.

e. **Appropriate placement of services**  Again, a careful analysis of the business problem being addressed is important when making decisions about the placement of processing services. The move toward thin clients and fat servers is not always the appropriate solution. Moving the application logic to a server, thus creating a fat server, can affect capacity as end users all attempt to use the application now located on the server. Sometimes it is possible to achieve better scaling by moving application processing to the client. Fat servers do tend to reduce network load because the processing takes place close to the data, and fat servers do lessen the need for powerful clients. Understanding the business problem intimately should help the architect to distribute the logic appropriately.

f. **Network analysis**  The most common bottleneck in distributed systems is still the network. Therefore, architects ignore the bandwidth capabilities of the network that the system must use at their peril. If the network cannot handle the amount of information that needs to pass between client and server, response time will suffer badly, and the system is likely to fail.

g. **Be aware of hidden costs**  Client/server implementation problems go beyond the analysis, development, and architecture problems listed previously (Atre, 1995). For example, systems that are intended to use existing hardware, networks, operating systems, and DBMSs are often stymied by the complexities of integrating these heterogeneous components together to build the client/server system. Training is a significant and recurring expense that is often overlooked. The complexities of working in a multi-vendor environment can be very costly.

2. **Managerial issues in introducing client/server:**

A variety of new opportunities and competitive pressures are driving the trend toward these database technologies. Corporate restructuring, such as mergers, acquisitions, and consolidations, make it necessary to connect, integrate, or replace existing stand-alone applications. Similarly, corporate downsizing has given individual managers a broader span of control, thus requiring access to a wider range of data. Applications are being downsized from expensive mainframes to networked microcomputers and workstations that are much more user-friendly and sometimes more cost-effective. Handling network traffic, which may become excessive with some architectures, is a key issue in developing successful client/server applications, especially as organizations move to place mission-critical applications in distributed environments. Establishing a good balance between centralized and decentralized systems is a matter of much current discussion as organizations strive to gain maximum benefits from both client/server and mainframe-based DBMSs.

3. **Web effects on client/server database systems:**

The Web is changing the distribution patterns of data. The thin clients represented by the browser interfaces move application logic to more centralized servers. The Internet backbone carries application architectures—in addition to messaging and mailing—in place of LANs or WANs. Thus, on the Web, tasks will have to be implemented in

Published by [www.emagazine.org](http://www.emagazine.org)
modules that can be controlled asynchronously (Panttaja, 1996). While application development and systems control may become more centralized, businesses are also beginning to deploy systems that reach outside their organization to business partners, customers, and suppliers.

4. The importance of ODBC:

Open Database Connectivity (ODBC) is similar to API, but it is designed for Windows-based client/server applications. It is most useful for accessing relational data and not well suited for accessing other types of data such as ISAM files. ODBC has been well accepted because it allows programmers to make connections to almost any vendor’s database without learning proprietary code specific to that database.

5. Movement to client/server databases systems:

Mission-critical systems that were resident on mainframe systems a decade ago have tended to remain on mainframe systems. Less mission-critical, frequently workgroup level, systems have been developed using client/server architectures. However, the popularity of client/server architectures and the strong desire to achieve more effective computing in more distributed environments as business perspectives became broader and more global has led to the deployment of mission-critical systems onto client/server architectures. We expect that each organization will need to achieve a balance between mainframe and client/server platforms, between centralized and distributed solutions, that are closely tailored to the nature of their data and location of business users of the data. As Hurwitz (1996) suggests, data that do not need to be moved often can be centralized on a mainframe. Data to which users need frequent access, complex graphics, and the user interface should be kept close to the users’ workstations.

6. Advantages of middleware:
   a. Middleware allows an application to interoperate with other software without requiring the user to understand and code the low-level operations required to achieve interoperability (Hurwitz, 1998).
   b. When APIs exist for several program development tools, you have considerable independence to develop client applications in the most convenient, front-end programming environment; yet you can still draw data from a common server database.
   c. ODBC has been well accepted even though it is difficult to program and implement. It allowed programmers to make connections to almost any vendor’s database without learning proprietary code specific to that database.
   d. Java Database Connectivity (JDBC) classes can be used to help an applet access any number of databases without understanding the native features of each database.
   e. OLE-DB adds value to the ODBC standard by providing a single point of access to multiple databases (Linthicum, 1997).

In selecting middleware, organizations tend to focus on scalability and recoverability. Unfortunately, these two variables have an inverse relationship. The more recoverable an application is, the less scalability it is likely to have. This is because of the extensive
journalizing, handshaking, and confirmation facilities that are necessary to establish recoverability (Hurwitz, 1998).

Suggestions for Field Exercises

1-3. Students may be helped when structuring their approach to any of these questions by being encouraged to ask questions that briefly describe the historical trends. This gives the interviewee an idea of the student’s level of knowledge, and it provides a starting point to discuss the university’s, department’s, or organization’s situation. Those that already own some legacy systems tend to further utilize them because development and implementation costs have been paid. Students may find that each retains its mainframes but moves strategically important applications to front-end distributed systems. Common problems in this process include poor integration of products from multiple vendors, inadequate performance, lack of support for security and database integrity, flawed communications programs, and lack of network-management facilities.

4. Examples of such sites may be those that allow visitors or customers to browse inventory records and place orders. There are questions we could ask in order to evaluate the overall functionality of the site. Some examples of such questions could be: Is the Web site content organized to emphasize specific areas? If so, is there easy access to those areas? How many clicks are necessary to locate the needed information? A good approach to working on this problem could be to find out which of the answers were highly influenced by the database connectivity level. Also, the site’s developer may be asked for the results of the site testing. Tests are usually performed with a specially developed tool like Mercury Interactive’s LoadRunner, a product that can simulate 50 million hits a day or 3,000 simultaneous users. Questions 4 and 5 allow the instructor to discuss the concept of usability, if he or she wishes.

5. Students should analyze the capabilities of each system, investigating and comparing features such as security, performance, and so forth. For example, if a system features improved customization capabilities to provide users with more flexibility, they can easily adapt the system to address their own view of the data. Ease of use is often misinterpreted as ease of learning: a system’s characteristic that would show how easy it is for the first-time user to understand and utilize the software. In general, GUI interfaces are considered to be easier to use than line mode interfaces. In general, the desktop packages such as Access are not as powerful as Oracle, Sybase, or Informix. However, the ease of use and inclusion of programming capabilities in the desktop products have led many organizations to use a desktop package such as Microsoft Access. Students may find architectures that employ a desktop package as a front-end to a more robust database.

6. Factors—such as size, throughput requirements, load capacity requirements, data distribution needs, employee expertise, legacy systems that need to be supported, and so forth—all play a role in decisions about database platforms and architectures. Different organizations will place varying amounts of importance on each of these factors. Student reports about their findings will be of interest to the class, and it will be useful to point
out the many important issues to be considered.

Case Questions

1. While the Mountain View IT staff may be able to undertake the project of moving toward an integrated environment, some concerns may be higher short-term costs, including acquiring advanced tools and incurring training costs. Also, it is not possible to know at the time when the decision is made to adopt, what the final product would look like. Other issues include compliance with HIPAA regulations. It may be better to outsource the project to a consulting firm that has had more experience developing an integrated environment. The hospital will also need to make a commitment to the IT staff to provide training that will enable them to understand the range of choices. They can work effectively with any outside consultants or vendors to reach the decision and then perform the implementation successfully. Internal expertise must be good enough to enable a wise business decision as vendors will be more focused on selling their product than meeting the business needs of Mountain View.

2. The four choices seem to offer a wide range of options for MVCH. One additional possibility is parallel processing.

3. One area that would improve substantially is data quality because all systems would be integrated. Data security should also be improved in an integrated system if the security policy is carefully devised and implemented so that access to patient records is centrally controlled.

Case Exercises

1. The advantages of moving toward an integrated environment would be the ability to leverage the existing systems while also utilizing some new technologies. This should improve performance. The downside to this is that the data quality would be only as good as the legacy systems’ data quality. Migration to a new system provides opportunity to perform data scrubbing. Often the most difficult challenge is integrating the legacy systems with the three-tier architecture. Discuss specific recommendations for technology in class after students have done some research on the Internet.

2. The fully integrated approach would provide many advantages such as commonalities in user interfaces; a stable hardware platform across the enterprise; and the possibility of integrating existing systems using a phased, rollout approach. Once again, data quality is a challenge as well as migration of existing legacy data. Also, the legacy systems may not be well documented, so there may be a steep learning curve for the staff that performs the migration.

3. This is an excellent question for an in-class debate between two teams of students. Each team should be assigned one of the approaches and should develop a defense for that approach. Another portion of the class could serve in the role of business sponsor and
determine, based upon the arguments presented, which solution would be the best way to
go.

4. This exercise is left to the students. This might also be an excellent in-class presentation
for students to make. Small groups of students could each investigate one local hospital.

5. The advantages to thin clients include cost savings as well as easier maintenance because
the application itself does not reside on the client. Perhaps the most beneficial thin client
device would be a PC utilizing an intranet and a browser-based application because this
does not require any special software other than a browser and network drivers. One
could also imagine utilizing wireless devices in order to access browser-based
applications. The hard-wired clients using a secure intranet would be the best approach
in order to ensure compliance with HIPAA privacy and security regulations. I would
recommend that MVCH pursues a thin client strategy because there would be substantial
cost savings.

6. Many applications that were designed to run on thin clients have been converted to
browser-based applications that could run on any platform (Windows, Mac, Linux, Unix,
etc.) as long as the client can run a browser. So effectively, any client can be a thin
client, and the Web-enabled application resides on a Web server and is centrally
maintained.

**Project Assignments**

P1. This project assignment depends upon the type of RDBMS being used by the class.

P2.
   a. You could establish the second database as a remote database server.
   b. You might want to have a separate schema for Dr. Z’s system, then export the
      original set of tables and import these into the MVCH database instance.
Chapter 10 The Internet Database Environment

Chapter Overview

This chapter highlights the importance of the Internet environment and the role of the database within it. It precisely defines key terms used in the Internet database environment to familiarize the student with the language used. In this edition, words that have become common usage and are now often taught in introductory MIS classes have been assumed.

The importance of databases in this rapidly growing field is evident. Students learn a variety of ways that databases are attached to Web pages in order to facilitate the conduct of business. A basic understanding of the database-enabled intranet/Internet environment is discussed to provide a framework for understanding how the different portions work together.

A historical perspective on languages used, their evolvement, and the advantages and disadvantages of each are explored. We then progress from knowing the architecture to knowing the theory of the languages to seeing actual code that implements the database.

Chapter Objectives

Specific student learning objectives are included at the beginning of the chapter. From an instructor’s point of view, the objectives of this chapter are to:
1. Emphasize the importance of new and emerging technologies that will carry businesses forward in a constantly evolving environment.
2. Define the different constructs of the Internet and the Web-enabled database, and give a comprehensive view of how they work together.
3. Establish the concepts of how clients pull up remote applications and data so that as the technology grows the student can make the transition from one means to another.

Key Terms

<table>
<thead>
<tr>
<th>ActiveX</th>
<th>Reverse proxy</th>
<th>World Wide Web Consortium (W3C)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application Program Interface (API)</td>
<td>Semantic Web</td>
<td>XHTML</td>
</tr>
<tr>
<td>Common Gateway Interface (CGI)</td>
<td>Server-side extension</td>
<td>XLink</td>
</tr>
<tr>
<td>Cookie</td>
<td>Service-oriented architecture (SOA)</td>
<td>XML</td>
</tr>
<tr>
<td>Domain name server (DNS) balancing</td>
<td>Simple Object Access Protocol (SOAP)</td>
<td>XML Schema</td>
</tr>
<tr>
<td>Document Structure Description (DSD)</td>
<td>Software and hardware load balancing</td>
<td>XPath</td>
</tr>
<tr>
<td>Java servlet</td>
<td>Universal Description, Discovery, and Integration (UDDI)</td>
<td>XPointer</td>
</tr>
<tr>
<td>Plug-ins</td>
<td>Web services</td>
<td>XSL</td>
</tr>
<tr>
<td>RELAX NG</td>
<td>Web Services Description Language (WSDL)</td>
<td>XSLT</td>
</tr>
</tbody>
</table>
Classroom Ideas

1. Have the students in the class draw the architecture for an intranet/Internet environment. Have them label all portions of the diagram.

2. Ask for volunteers to play the role of each of the pieces in the structure. Have another student act as a client request for data and watch the request go through the structure in sequence as each student-representative of the structure explains his or her role.

3. Have the students use a simple database application that has not yet been converted to run on the Internet. Divide the students into groups and have each group Web-enable the application in a different way or language. Have each group make a presentation to the class on how they accomplished their goal. Have each group turn in a hard copy of their code. Make these into a handout to distribute to the class in the next meeting.

4. Emphasize the risks of conducting business on the Internet. Discuss the levels at which security is carried out. Ask the students to discuss why security needs to be in place and what kinds of security. Point out the costs of both implementing the security (in terms of designing and implementing security structures and monitoring their use) and the costs of not implementing the security (stolen data, angry customers, system damage, repair time, etc.).

5. Discuss the advantages and disadvantages of having privacy policies in place in a company on the Internet. Check out the Internet for Web sites that focus on privacy issues. Ask the students to find policies that are in place in other countries and bring them to class.

Answers to Review Questions

1. Define each of the following key terms:
   a. XML Schema A language used for defining XML databases that has been recommended by the WWW Consortium.
   b. Cookie A block of data stored on a client by a Web server. When a user returns to the site later, the contents of the cookie are sent back to the Web server and may be used to identify the user and return a customized Web page.
   c. DNS balancing/Domain Name Space A load-balancing approach where the DNS server for the hostname of the site returns multiple IP addresses for the site.
   d. Electronic business (e-business) A technology-enabled business that is using Internet-related technology to facilitate the development of more integrated relationships with customers and suppliers.
   e. API Routines that an application uses to direct the performance of procedures.
   f. Server-side extension A software program that interacts directly with a Web server to handle requests.
   g. Web services Set of emerging standards that define protocols for communication between software programs over the Web.
   h. XSL A language used to develop style sheets specific to how an XML document can be displayed.
i. **SOAP/Simple Object Access Protocol** An XML-based communication protocol used for sending messages between applications via the Internet.

2. Match the following terms and definitions:

   f  CGI
   c  DSD
   g  SOA
   h  XHTML
   e  servlet
   a  reverse proxy
   d  Semantic web
   b  W3C

3. Contrast the following terms:

   a. **CGI; API** CGI is a web server interface for specifying the transfer of information from a Web server to a CGI Program. An API is a set of procedures that an application uses to direct the performance of routines in the operating system.

   b. **Internet; intranet; extranet** Attaching a database to the Internet or an intranet involves the same logic and processes. However, the Internet is an external network, while an intranet is local, usually safe within an organizational firewall. With an extranet, there is access and exchange of data in an external network, but it is not as universal in its access.

   c. **HTML; XML; XHTML** HTML is a scripting language used for documents displayed through Web browsers; it is similar to SGML (Standard Generalized Markup Language). XML is another scripting language based on SGML that allows the creation of customized tags that facilitate transmission and sharing of data across organizations. XHTML is an extension of HTML that makes HTML XML-compliant.

   d. **DNS balancing; software and hardware load balancing; reverse proxy** These are different approaches to load balancing. With DNS balancing, a server is able to handle more hits by placing multiple copies of the site on separate but identical servers. This approach is simple, but it doesn’t guarantee balancing because the IP addresses chosen may not be balanced. With software and hardware load balancing, the requests are distributed more evenly across servers by routing requests to a single IP address among multiple servers. This approach is generally more successful than DNS balancing. Reverse proxy reduces the load on a Web site by storing the responses to a client from a server in its own cache. This means that client requests can sometimes be serviced by the information in the cache rather than a return to the Web server.

   e. **SQL; XML** SQL is a de facto standard language for creating and querying relational databases. XML is a rapidly developing scripting language that permits the capturing of data structures in Web-based documents. XML provides a means of transferring, exchanging, and manipulating data with HTML by describing the data, which is often retrieved from relational databases.

   f. **Web services; SOA** Web services are a set of standards that define protocols for communication between software programs over the Web using XML. SOA is an architecture with services that pass information between one another. While Web services are XML based, SOA services are not necessarily. An example of an SOA is CORBA.
4. Attaching a database to a Web page:

Attaching a database to a Web page is important in the facilitation of e-business because it allows the querying of live data and storage of user input. Customers may determine availability of desired products, place orders, and pay for them without requiring a live employee. Purchases may be made at any time from an Internet connection. Pages without databases or static Web pages are fixed in time and cannot receive user input.

5. The components of an environment needed to establish database-enabled connectivity:

In order to establish database-enabled connectivity, we need to have a language in which to write our application, such as HTML; a Web server interface to allow the Web server to interact with the external program, such as CGI (Common Gateway Interface); a Web server on which to store our data and programs; a database to store and serve up answers to user queries, such as Oracle; a network such as the Internet; and addresses for those Web servers on the Internet to identify the site to clients.

6. Why integrate XML data structure descriptions with relational database navigation?

Relational databases are well-established in organizations—both business and non-business—and are unlikely to disappear anytime soon. As more commerce and interaction with the public is pushed to Web-based solutions, there is an increasing need to provide Web access to the data that is stored in these relational databases. Although Web access to data can be provided by programming language solutions (e.g., Java), this access is not easily obtained. XML provides a hierarchically based description of data structures that is platform-independent and extensible. By integrating XML data structure descriptions with relational database navigation, organizations will more easily be able to meet their business needs for Web access of data.

7. According to the XML Query Working Group, “XQuery is a standardized language for combining documents, databases, Web pages, and almost anything else. It is very widely implemented. It is powerful and easy to learn. XQuery is simpler to work with and easier to maintain than many other alternatives.” It would appear that due to its similarity to SQL syntax and structure, as well as the aspects mentioned by the XML Query Working Group, that XQuery is popular for these reasons. It can be easily grasped by someone with prior SQL experience due to its similarity in syntax and structure, yet it is easy enough to learn for someone new to manipulating data with a standard language.

8. W3C: The World Wide Web Consortium is the chief standards body for HTTP (Hypertext Transfer Protocol) and HTML. Founded in 1994, it is an international consortium of companies intent on developing open standards that will allow Web documents to be consistently displayed across all platforms.

9. Web services are an emerging standard that defines protocols for automatic communication between software programs over the Web. They are XML-based and run in the background,
thus providing transparent communication between companies. The benefits of widespread deployment are substantial because businesses are able to exchange information in a much easier way. For example, a developer does not have to know the technical details of an application being integrated into a B2B system. Deployment also results in a substantial decrease in development time for integrated systems and enhances productivity in many sectors. The biggest concerns about adopting Web services are transaction speed, security, and reliability.

10. The purpose of ColdFusion and ASP.net:

Many middleware applications exist that ease the connection of databases to Web applications. ColdFusion and ASP.net are two of the most popular.

11. Compare and contrast CGI, API and Java servlets:

CGI is a Web server interface that specifies the transfer of information between a Web server and a CGI program. CGI programs accept and return data and may be written in C, C++, Perl, Java, or Visual Basic. CGI programs can be run under different information servers interchangeably. APIs are more efficient than CGI; they are implemented as shared code or DLLs (dynamic link libraries). No external code is needed to process a request. An API can be used by an application to direct the operating system’s performance of procedures. APIs use a shared connection to the server, unlike CGI, which establishes a new link each time. APIs are specific to an operating system and must be rewritten to run with other systems. Java servlets are small programs that execute from within another application rather than the operating system. They are stored on the server rather than with the application. Servlets allow a client program to upload additional program code to a server where it executes. They remain in active memory after execution, unlike CGI scripts that close after execution.

12. Describe three methods for balancing server loads:

DNS balancing, software and hardware load balancing, and reverse proxy are different approaches to load balancing. With DNS balancing, a server is able to handle more hits by placing multiple copies of the site on separate but identical servers. This approach is simple, but it doesn’t guarantee balancing because the IP addresses chosen may not be balanced. With software and hardware load balancing, the requests are distributed more evenly across servers by routing requests to a single IP address among multiple servers. This approach is generally more successful than DNS balancing. Reverse proxy reduces the load on a Web site by storing the responses to a client from a server in its own cache. This means that client requests can sometimes be serviced by the information in the cache rather than a return to the Web server.

13. A cookie is a block of data stored on a client by a Web server. When a user returns to the site later, the contents of the cookie are sent back to the Web server and may be used to identify the user and return a customized Web page.
14. A PHP program that enables a dynamic Web site is going to contain a header.inc file that contains HTML code to set up a generic page as well as embedded SQL code.

15. An XML schema is a step ahead from a DTD because it allows data types to be denoted.

16. Define a plug-in and provide an example:

A plug-in is a hardware or software module that extends the capabilities of a browser by adding features. An example of a plug-in is the Adobe Acrobat Reader.

17. Describe ActiveX controls:

ActiveX controls are a set of controls defined by Microsoft that allow manipulation of data inside the browser.

18. Discuss some concerns that businesses have about adopting a Web services approach: Currently, the security of Web services is not as well developed as that of traditional applications.

19. Discuss UDDI:

Universal Description, Discovery and Integration is a technical specification for creating a distributed registry of Web services and businesses that are open to communicating through Web services. The white pages category contains general company information. The yellow pages category includes classification information about companies and/or Web services. The green pages section contains technical information about Web services, including how to call them up.

20. Discuss why the development of standards for Web services is being hampered:

One problem has been the existence of four standards bodies: OASIS, WS-I, W3C, and the Liberty Alliance. In addition to these bodies, IBM and Microsoft have been working jointly to establish standards. There has been little cooperation toward developing more sophisticated standards.
Answers to Problems and Exercises

1. The difference between a static and a dynamic Web site:

   A static Web page is fixed in time and cannot receive user input. A dynamic page allows the querying of live data and storage of user input. Using dynamic Web pages is important in the facilitation of e-business because it allows the conduct of business to take place quickly and easily, often without the intervention of costly customer support.

2. Languages associated with Internet application development:

   There are scripting or markup languages, such as HTML, XHTML, and XML. These scripting languages are intended for handling layout and display of documents, rather than for programming functions. General purpose, object-oriented languages were brought into use for these activities. Java is an example of one of these. Small Java programs, called Java applets, download from the Web server to the client and run in a Java-compatible Web browser. JavaScript is a scripting language based on Java, but it is easier to learn. JavaScript can be used to achieve interoperability and dynamic content. VBScript, based upon Visual Basic but easier to learn, can be used to create interactive Web pages. Cascading style sheets (CSS) have been developed to allow both a developer and a user to create style sheets that can define any Web page.

3. This is left as a student exercise.

4. Note to instructor: Students may find it helpful to prototype the TUTOR relation. Note that the TUTOR table used here is not identical to the one used in chapters 7 and 8, where a supertype, PERSON, was used.

   Note: This solution is based on a MS Access 2003 database with sample data entered for the Tutor table; your student answers will vary based on the table structure they have developed and the test data they have created. MS Access permits you to export a table as XML; MS Access creates a HTML document (.htm), an XML document, an XML schema (.xsd), and the XML stylesheet (.xsl) for the table you select to Export as XML.

   The XSD schema (required—showing structure) and the XML document (optional—showing record values) are presented as part of the solution following.
XSD SCHEMA (TUTOR.XSD)

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:od="urn:schemas-microsoft-com:officedata">
    <xsd:element name="dataroot">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="TUTOR" minOccurs="0" maxOccurs="unbounded"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="TUTOR">
        <xsd:annotation>
            <xsd:appinfo>
                <od:index index-name="PrimaryKey" index-key="TUTOR_ID " primary="yes"
                    unique="yes" clustered="no"/>
                <od:index index-name="TUTOR_ID" index-key="TUTOR_ID " primary="no"
                    unique="no" clustered="no"/>
            </xsd:appinfo>
        </xsd:annotation>
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="TUTOR_ID" minOccurs="1" od:jetType="longinteger"
                    od:sqlSType="int" od:nonNullable="yes" type="xsd:int"/>
                <xsd:element name="T_LNAME" minOccurs="1" od:jetType="text"
                    od:sqlSType="nvarchar" od:nonNullable="yes">
                    <xsd:simpleType>
                        <xsd:restriction base="xsd:string">
                            <xsd:maxLength value="50"/>
                        </xsd:restriction>
                    </xsd:simpleType>
                </xsd:element>
                <xsd:element name="T_FNAME" minOccurs="0" od:jetType="text"
                    od:sqlSType="nvarchar">
                    <xsd:simpleType>
                        <xsd:restriction base="xsd:string">
                            <xsd:maxLength value="50"/>
                        </xsd:restriction>
                    </xsd:simpleType>
                </xsd:element>
                <xsd:element name="T_PHONE" minOccurs="0" od:jetType="text"
                    od:sqlSType="nvarchar">
                    <xsd:simpleType>
                        <xsd:restriction base="xsd:string">
                            <xsd:maxLength value="14"/>
                        </xsd:restriction>
                    </xsd:simpleType>
                </xsd:element>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
```
<xsd:element name="T_EMAIL" minOccurs="0" od:jetType="text" od:sqlSType="nvarchar">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:maxLength value="25"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>

<xsd:element name="CERT_DATE" minOccurs="1" od:jetType="datetime" od:sqlSType="datetime" od:nonNullable="yes" type="xsd:dateTime"/>
<xsd:element name="STATUS" minOccurs="1" od:jetType="text" od:sqlSType="nvarchar" od:nonNullable="yes">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:maxLength value="10"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
</xsd:complexType>
</xsd:element>
</xsd:schema>
Problem and Exercise 10-4, continued (optional display of XML, with record values shown)

XML DOCUMENT (TUTOR.XML)

```xml
<?xml version="1.0" encoding="UTF-8"?>
<dataroot xmlns:od="urn:schemas-microsoft-com:officedata"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="TUTOR.xsd" generated="2007-12-30T15:37:43">
  <TUTOR>
    <TUTOR_ID>100</TUTOR_ID>
    <T_LNAME>Lupin</T_LNAME>
    <T_FNAME>Remus</T_FNAME>
    <T_PHONE>906-487-1500</T_PHONE>
    <T_EMAIL>lupin@hogwarts.edu</T_EMAIL>
    <CERT_DATE>2008-01-05T00:00:00</CERT_DATE>
    <STATUS>Active</STATUS>
  </TUTOR>
  <TUTOR>
    <TUTOR_ID>101</TUTOR_ID>
    <T_LNAME>Moody</T_LNAME>
    <T_FNAME>Mad-Eye</T_FNAME>
    <T_PHONE>906-487-1500</T_PHONE>
    <T_EMAIL>moody@hogwarts.edu</T_EMAIL>
    <CERT_DATE>2008-01-05T00:00:00</CERT_DATE>
    <STATUS>Temp Stop</STATUS>
  </TUTOR>
  <TUTOR>
    <TUTOR_ID>102</TUTOR_ID>
    <T_LNAME>Umbridge</T_LNAME>
    <T_FNAME>Dolores</T_FNAME>
    <T_PHONE>906-487-1500</T_PHONE>
    <T_EMAIL>prettyinpink@hogwarts.edu</T_EMAIL>
    <CERT_DATE>2008-01-05T00:00:00</CERT_DATE>
    <STATUS>Dropped</STATUS>
  </TUTOR>
  <TUTOR>
    <TUTOR_ID>103</TUTOR_ID>
    <T_LNAME>Dumbledore</T_LNAME>
    <T_FNAME>Albus</T_FNAME>
    <T_PHONE>906-487-1500</T_PHONE>
    <T_EMAIL>fawkes@hogwarts.edu</T_EMAIL>
    <CERT_DATE>2008-05-22T00:00:00</CERT_DATE>
    <STATUS>Active</STATUS>
  </TUTOR>
  <TUTOR>
    <TUTOR_ID>104</TUTOR_ID>
    <T_LNAME>Sprout</T_LNAME>
    <T_FNAME>Pomona</T_FNAME>
    <T_PHONE>906-487-1500</T_PHONE>
    <T_EMAIL>sprout@hogwarts.edu</T_EMAIL>
    <CERT_DATE>2008-05-22T00:00:00</CERT_DATE>
    <STATUS>Active</STATUS>
  </TUTOR>
</dataroot>
```
5. Write an XPath expression that will list all the tutors.

   Using the MS Access generated XML from problem #4, the XPath expression to list all tutors follows (using a wildcard, as illustrated in the chapter):

   /TUTOR/*

6. Expand XPath expression into a FLOWR XQuery. Note: The where is not needed as there is no criterion for selection of a subset of Tutors:

   for $p in doc("TUTOR.XML")/TUTOR/T_LNAME
   order by $p/T_LNAME
   return $p/T_LNAME

7. Two other combinations of products:

   One combination is Oracle, Apache Tomcat, and JSP. JSP allows one to embed Java code in HTML pages. Tomcat then generates java class files from the JSP and executes these on the Tomcat Web server.
Another combination is Internet Information Services (IIS), MS SQL Server, and ASP.net. ASP.net is similar to JSP, except that it is Microsoft-platform specific.

```html
<!DOCTYPE HTML PUBLIC
"-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html401/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>Wine Table Structure</title>
</head>
<body><pre>
<?php
// Open a connection to the server and USE the winestore
$connection = mysql_connect("localhost","fred","shhh");
mysql_select_db("winestore", $connection);

// Run a query on the wine table in the winestore database to retrieve
// one row
$result = mysql_query ("SELECT * FROM wine LIMIT 1", $connection);

// Output a header, with headers spaced by padding
print str_pad("Field", 20) .
    str_pad("Type", 14) .
    str_pad("Null", 6) .
    str_pad("Key", 5) .
    str_pad("Extra", 12) . "\n";

// How many attributes are there?
$x = mysql_num_fields($result);

// For each of the attributes in the result set
for($y=0;$y<$x;$y++)
{
    // Get the meta-data for the attribute
    $info = mysql_fetch_field ($result);

    // Print the attribute name
    print str_pad($info->name, 20);

    // Print the data type
    print str_pad($info->type, 6);

    // Print the field length in brackets e.g.(2)
    print str_pad("({$info->max_length})", 8);

    // Print out YES if attribute can be NULL
    if ($info->not_null != 1)
        print " YES ";
    else
        print "     ";

    // Print out selected index information
    if ($info->primary_key == 1)
        print " PRI ";
    elseif ($info->multiple_key == 1)
```
9. `<?php #gu_byemail.php`  

// This script retrieves a record once an email has been specified.

$page_title = 'Lookup a User';
include ('templates/header.inc');

require_once ('../mysql_connect.php'); // Connect to the db.
function escape_data ($data) {
    global $dbc; // Need the connection.
    if (ini_get('magic_quotes_gpc')) {
        $data = stripslashes($data);
    }
    return mysql_real_escape_string($data, $dbc);
} // End of function.

// Make the query.

// Check for an email address.
    if (empty($_POST['email'])) {
        $e = FALSE;
        $message .= '<p>You forgot to enter the email address!</p>);
    } else {
        $e = escape_data($_POST['email']);
    }

$query = "SELECT CONCAT(last_name, ', ', first_name) AS name, DATE_FORMAT(registration_date, '%M %d, %Y') AS dr FROM users where email = '$e";  
$result = @mysql_query ($query); // Run the query.
$num = mysql_num_rows ($result); // How many users are there?

if ($num > 0) { // If it ran OK, display the records.
Answers to Field Exercises

1. This answer will vary depending upon what plug-ins a user has installed on his or her machine.

2. This answer will also vary depending on what cookies have been installed on the user’s machine. Places to look for cookies include temp directories under an Internet provider or the machine’s operating system.
3. This answer will vary depending upon the software chosen and the type of network and Web site chosen.

4. This is left as a student exercise, based upon the type of data access required.

5. This is left as a student exercise.

**Project Case Study**

**Case Questions**

1. This is a question that should be thoroughly considered by the planning committee. The issues around patient care and confidentiality are different from the issues faced by other businesses. Patients with certain diseases are often treated prejudicially if their privacy is violated, and multi-million-dollar lawsuits have arisen over these issues. In addition, there are some diseases a patient needs to hear diagnosed in person, not over a Web page. The importance of addressing privacy and security concerns cannot be overstated to your students.

2. The hospital might also consider using an intranet with a Web-enabled database application or applications accessible only to personnel while working onsite. This would be in essence a thin client solution.

3. Data entry in a health care system can be a sticky issue. Highly trained and specialized medical staff such as doctors and nurses often balk at low-level “administrative” tasks and don’t want to do them. Data-entry staff are often not adequately medically trained to decipher terminology and abbreviations used in patient care. Transcription errors can occur even when training is conducted. While there are many options that minimize hand entry, such as voice recognition, optical-character recognition (OCR), and some handwriting software available, these are often relatively unreliable in a field that demands precision. Any data entry design would need to include training for all who will use the equipment, both in the hardware and the software and in the specialized language of patient care. Due to the sensitivity of the data, there should be strict controls to ensure that patient confidentiality is maintained. Due to the importance of the data, there should be policies to review anything added into the system and verify the content back to the originator of the data.

4. The healthcare industry has been slower to embrace Web services due to the mission critical nature of healthcare applications as well as the very specific proprietary healthcare information systems that are still in widespread use today. If a hospital has to face using Web services to integrate many different proprietary systems for various departments, the integration challenge is substantial. Mountain View Community Hospital would view Web services as a success if there could be a guarantee that the data provided by the system was accurate and could be delivered in a timely fashion. Also, there must be assurance of privacy and security.
5. The potential implementation of Web-based solutions should be treated as a strategy issue because the impact of a quality system reaches far beyond those who use the system in healthcare but also to patients and the community as a whole.

Case Exercises

1. This is an excellent student exercise. Some possible services that students should look for are:
   - Finding a physician online
   - Scheduling appointments online
   - Health information online

2. Web services could be used in several ways at MVCH:
   - Submitting insurance claims online
   - Providing medical records to other facilities
   - Implementing supply-chain management

   Of course, all of these would have some security considerations as well as data quality and accuracy.

3. Mr. Heller would need to show that the hospital is going to save money as well as improve patient care. For each of the three items mentioned in #2, there are cost/benefits as follows:
   - Submitting insurance claims online  There will be the costs of people to implement and maintain the system. The benefits will include increased accessibility of data, decreased data entry time, and greater ability to cross-reference.

   - Providing medical records to other facilities  Such a system would have a tremendous benefit in that patients could go to any number of hospitals in a region, and staff would have access to their entire medical records. The costs for security could be substantial.

   - Implementing supply-chain management  The costs will be greatly outweighed by the benefits. Secure online ordering, minimized data entry, and up-to-the-minute price and account information will make the organization much more efficient.

4. 4.1 Submitting insurance claims online:
   A. Security and confidentiality concerns  Only billing staff need access to this information. Files containing this information can be protected with passwords and database privileges. How likely the proposed system is to be compromised will depend upon the systems chosen.

   B. Data entry requirements  Each division should handle its own data entry, which will motivate each department to be accurate (because they will be the ones using the data). This will also avoid resentment between departments being made to do another’s data entry.
C. **Benefits and costs** There will be the costs of people to implement and maintain the system. The benefits will include increased accessibility of data, decreased data entry time, and greater ability to cross-reference.

4.2 **Providing clinical information to patients online:**

A. **Security and confidentiality concerns** Patients should be able to access only their own information. Sensitive information, such as a diagnosis of a terminal disease, should not be placed online until after a physician has counseled the patient. Security should be strong because of the risk of misuse of the data.

B. **Data entry requirements** Appointment clerks should enter the initial data for new patients and all information relating to new appointments. Billing staff should enter billing and diagnosis information. Patient care specialists, such as medical stenographers, should be assigned to transcribe physicians’ notes. While each division will be entering their own data, friction should be minimized unless one division is more careless than another, and another department has to correct the mistakes.

C. **Benefits and costs** The benefits can include decreased time on the telephone for physicians answering patient questions and patients receiving results faster. Costs will include the cost of the network and the security mechanisms required.

4.3 **Implementing supply-chain management online:**

A. **Security and confidentiality concerns** These concerns will not be as pressing as for the patient care functions.

B. **Data entry requirements** These will also be minimized since generally suppliers today have their own Web sites, online ordering, and often EDI.

C. **Benefits and costs** The costs will be greatly outweighed by the benefits. Secure online ordering, minimized data entry, and up-to-the-minute price and account information will make the organization much more efficient.

4.4 **Providing medical records to other facilities:**

A. **Security and confidentiality concerns** These would be a major concern because electronic transmission could be intercepted. The strictest security policies would have to be implemented, including such things as encryption. Also, patients would have to sign a written release form.

B. **Data entry requirements** It is assumed that all of the data would be extracted from existing systems.

C. **Benefits and costs** Such a system would have a tremendous benefit in that patients could go to any number of hospitals in a region, and staff would have access to their entire medical records. The costs for security could be substantial.

4.5 **Implementing an online medical knowledgebase**

A. **Security and confidentiality concerns** The hospital would have to decide whether or not to allow public access to the knowledge base.

B. **Data entry requirements** It is reasonable to assume that much of the knowledge base information would be purchased.
C. Benefits and costs The benefits to the patients and staff would be substantial. The greatest cost would be the actual content.

Implementing the submission of insurance claims online would be recommended because it would have the greatest direct benefit in streamlining the billing process.

Note to instructor: The solution to Case Exercise 5 is based on a pre–ninth edition database of the Mountain Valley Community Hospital and is presented for illustrative purposes only. The solution presented here does not correspond to the suggested design solutions in earlier chapters of the ninth edition due to the extensive updating of the case in the ninth edition.

5. a. Patient_t table as an XML file:

```xml
<PATIENT_t>
  <Patient_ID>13</Patient_ID>
  <Person_ID>12</Person_ID>
  <Physician_ID>9723</Physician_ID>
  <Admission_Type>Outpatient</Admission_Type>
  <Contact_Date>2001-04-22T00:00:00</Contact_Date>
  <Contact_Time>1899-12-30T14:40:00</Contact_Time>
</PATIENT_t>
<PATIENT_t>
  <Patient_ID>14</Patient_ID>
  <Person_ID>13</Person_ID>
  <Physician_ID>9801</Physician_ID>
  <Admission_Type>Resident</Admission_Type>
  <Contact_Date>2001-04-22T00:00:00</Contact_Date>
  <Contact_Time>1899-12-30T14:45:00</Contact_Time>
  <Bed_ID>AE-102-1</Bed_ID>
</PATIENT_t>
<PATIENT_t>
  <Patient_ID>15</Patient_ID>
  <Person_ID>14</Person_ID>
  <Physician_ID>9945</Physician_ID>
  <Admission_Type>Resident</Admission_Type>
  <Contact_Date>2001-04-22T00:00:00</Contact_Date>
  <Contact_Time>1899-12-30T15:10:00</Contact_Time>
  <Bed_ID>AE-103-1</Bed_ID>
</PATIENT_t>
<PATIENT_t>
  <Patient_ID>16</Patient_ID>
  <Person_ID>15</Person_ID>
  <Physician_ID>9624</Physician_ID>
  <Admission_Type>Outpatient</Admission_Type>
  <Contact_Date>2001-04-22T00:00:00</Contact_Date>
</PATIENT_t>
```
b. Query related to PATIENT table as an XML file. Both the XML data as well as the schema are shown following:

**XML**
```xml
<?xml version="1.0" encoding="UTF-8"?>
<dataroot xmlns:od="urn:schemas-microsoft-com:officedata"
xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="CH8-a1.xsd">
<CH8-a1>
<Patient_ID>11</Patient_ID>
<Patient_Name>Bronson Chuck</Patient_Name>
<Bed_ID>AE-100-1</Bed_ID>
<Admission_Type>Resident</Admission_Type>
</CH8-a1>
<CH8-a1>
<Patient_ID>12</Patient_ID>
<Patient_Name>Freeman Rita</Patient_Name>
<Bed_ID>AE-102-1</Bed_ID>
<Admission_Type>Outpatient</Admission_Type>
</CH8-a1>
<CH8-a1>
<Patient_ID>13</Patient_ID>
<Patient_Name>Grost Anita</Patient_Name>
<Bed_ID>AE-102-1</Bed_ID>
<Admission_Type>Resident</Admission_Type>
</CH8-a1>
<CH8-a1>
<Patient_ID>14</Patient_ID>
<Patient_Name>Danger Johnny</Patient_Name>
<Bed_ID>AE-103-1</Bed_ID>
<Admission_Type>Resident</Admission_Type>
</CH8-a1>
</dataroot>
```
<Patient_x0020_ID>15</Patient_x0020_ID>  
<Patient_x0020_Name>Nickolson Steven</Patient_x0020_Name>  
<Admission_Type>Outpatient</Admission_Type>  
</CH8-a1>  
<CH8-a1>  
<Patient_x0020_ID>17</Patient_x0020_ID>  
<Patient_x0020_Name>Slinestone Wilma</Patient_x0020_Name>  
<Bed_ID>GC-100-1</Bed_ID>  
<Admission_Type>Resident</Admission_Type>  
</CH8-a1>  
</dataroot>  

Schema  
<?xml version="1.0" encoding="UTF-8"?>  
<xsd:schema xmlns:xsd="http://www.w3.org/2000/10/XMLSchema"  
xmlns:od="urn:schemas-microsoft-com:officedata">  
<xsd:element name="dataroot">  
<xsd:complexType>  
<xsd:choice maxOccurs="unbounded">  
<xsd:element ref="CH8-a1"/>  
</xsd:choice>  
</xsd:complexType>  
</xsd:element>  
<xsd:element name="CH8-a1">  
<xsd:annotation>  
<xsd:appinfo/>  
</xsd:annotation>  
<xsd:complexType>  
<xsd:sequence>  
<xsd:element name="Patient_x0020_ID" od:jetType="autonumber" od:sqlSType="int"  
od:autoUnique="yes" od:nonNullable="yes">  
<xsd:simpleType>  
<xsd:restriction base="xsd:integer"/>  
</xsd:simpleType>  
</xsd:element>  
<xsd:element name="Patient_x0020_Name" minOccurs="0" od:jetType="text"  
od:sqlSType="nvarchar">  
<xsd:simpleType>  
<xsd:restriction base="xsd:string">  
<xsd:maxLength value="255"/>  
</xsd:restriction>  
</xsd:simpleType>  
</xsd:element>  
<xsd:element name="Bed_ID" minOccurs="0" od:jetType="text" od:sqlSType="nvarchar">  
<xsd:simpleType>  
<xsd:restriction base="xsd:string">  
<xsd:maxLength value="12"/>  
</xsd:restriction>  
</xsd:simpleType>  
</xsd:element>  
</xsd:sequence>  
</xsd:complexType>  
</xsd:element>  
</xsd:schema>
c. For this exercise, a simple report in Access was created to show treatment information. The report is named treatpers:
A module was created within MS-Access 2003 to generate the XML code:

Option Compare Database

Private Sub ExportReport()
    ' Purpose: Exports the Invoice table as well as
    ' the presentation and image files. In addition,
    ' a file containing the ReportML is created as
    ' denoted by setting the OtherFlags flag equal
    ' to 16.
    Const CREATE_REPORTML = 16

    Application.ExportXML
        ObjectType:=acExportReport, _
        DataSource:="treatpers", _
        DataTarget:="C:\test\treatpers.xml", _
        PresentationTarget:="C:\test\treatpers.xsl", _
        ImageTarget:="C:\test\Images", _
        OtherFlags:=CREATE_REPORTML
End Sub

This generates four files: treatpers.xml, treatpers.xsl, treatpers.html, and treatpers_report.xml

In order to run the report in a browser, simply open the html file.

**Project Assignments**

**P1.** For this assignment, a user login screen is created. This assumes that a MySQL table
called users is already in existence. In order to access MySQL, mysql_connect.php must
reside in the directory above where the php code is going to be placed. Also, an images
file is needed where a .jpg file can be placed to display.

```php
<?php # - mysql_connect.php

// This file contains the database access information. This file also
establishes a connection to MySQL and selects the database.

// Set the database access information as constants.
DEFINE ('DB_USER', 'shoretoshore');
DEFINE ('DB_PASSWORD', 'shore');
DEFINE ('DB_HOST', 'localhost');
DEFINE ('DB_NAME', 'test');

// Make the connection and then select the database.
$dbc = @mysql_connect (DB_HOST, DB_USER, DB_PASSWORD) OR die ('Could not
    connect to MySQL: ' . mysql_error() );
mysql_select_db (DB_NAME) OR die ('Could not select the database: ' .
    mysql_error() );
?>
```

Published by [www.emagazine.org](http://www.emagazine.org)
Chapter 10

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/2000/REC-xhtml1-20000126/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
<meta http-equiv="content-type" content="text/html; charset=iso-8859-1" />
<title><?php echo $page_title; ?></title>
</head>
<body>
<!-- - header.inc -->
<table border="0" cellspacing="0" cellpadding="4">
<tr> <!-- TOP ROW -->
<td rowspan="2" bgcolor="#999966"><img src="images/mvch.jpg" alt="hospital" width = 200" height="300" /></td>
<td width="*" bgcolor="#999966"><font color="#FFFFFF" size="+2" face="Courier New, Courier, mono">Mountain View Community Hospital<br>Patient Information System<br>Login Screen<br></font></td>
<td width="10" rowspan="2" bgcolor="#999966">&nbsp;</td>
</tr>
<tr> <!-- NAVIGATIONAL ROW -->
<td bgcolor="#CC9933">
<table width="100%" border="0" cellspacing="2" cellpadding="2">
<tr>
<td width="20%" align="center" bgcolor="#FFCC66"><a href="index.php">Home</a></td>
<td width="20%" align="center" bgcolor="#FFCC66"><a href="register.php">Register</a></td>
<td width="20%" align="center" bgcolor="#FFCC66"><a href="login.php">Login</a></td>
</tr>
</table>
</td>
</tr>
<tr> <!-- CONTENT ROW -->
<td bgcolor="#999966">&nbsp;</td>
<td bgcolor="#FFFFFF"><!-- PAGE SPECIFIC CONTENT STARTS HERE -->

MVCHHeader.inc

<!-- PAGE CONTENT ENDS HERE --></td>
</tr>
</table>
</body>
</html>

Published by www.emagazine.org
<td bgcolor="#999966">
<font size="-1">/a</font>/</td>
<td bgcolor="#999966">&nbsp;</td>
</tr>
</table> <!-- Script 6.2 - footer.inc -->
</body>
</html>

**MVCHFooter.inc**

```php
<?php # MVCHLogin.php

// Set the page title and include the HTML header.
$page_title = 'Change Your Password';
include ('templates/mvchheader.inc');

if (isset($_POST['submit'])) { // Handle the form.
    require_once ('../mysql_connect.php'); // Connect to the db.

    // Create a function for escaping the data.
    function escape_data ($data) {
        global $dbc; // Need the connection.
        if (ini_get('magic_quotes_gpc')) {
            $data = stripslashes($data);
        }
        return mysql_real_escape_string($data, $dbc);
    } // End of function.

    $message = NULL; // Create an empty new variable.

    // Check for a username.
    if (empty($_POST['username'])) {
        $u = FALSE;
        $message .= '<p>You forgot to enter your username!</p>';
    } else {
        $u = escape_data($_POST['username']);
    }

    // Check for an existing password.
    if (empty($_POST['password'])) {
        $p = FALSE;
        $message .= '<p>You forgot to enter your password!</p>';
    } else {
        $p = escape_data($_POST['password']);
    }

    if ($u && $p) { // If everything's OK.
        $query = "SELECT user_id FROM users WHERE (username='$u' AND password=PASSWORD('$p'))";
        $result = @mysql_query ($query);
        $num = mysql_num_rows ($result);
        if ($num == 1) {
            echo '<p><b>Welcome to the MVCH System.</b></p>
```
include ('templates/footer.inc'); // Include the HTML footer.
exit(); // Quit the script.

} else {
$message = '<p>Your username and password do not match our records.</p>';
}
mysql_close(); // Close the database connection.

} else {
$message .= '<p>Please try again.</p>';
}

} // End of the main Submit conditional.

// Print the error message if there is one.
if (isset($message)) {
    echo '<font color="red">', $message, '</font>';
}

<?php
include ('templates/mvchfooter.inc'); // Include the HTML footer.
?>

MVCHLogin.php

P2. For this assignment, the person table with a small amount of data is used.

<?xml version="1.0" encoding="UTF-8"?>
<dataroot xmlns:od="urn:schemas-microsoft-com:officedata"
xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="Patient.xsd">
<Patient>
<PT_Person_ID>21</PT_Person_ID>
<MRN>3412</MRN>
<P_Person_ID>12</P_Person_ID>
</Patient>
</dataroot>
<Patient>
  <PT_Person_ID>34</PT_Person_ID>
  <MRN>1231-23</MRN>
  <P_Person_ID>212</P_Person_ID>
</Patient>

<Patient>
  <PT_Person_ID>56</PT_Person_ID>
  <MRN>2352</MRN>
  <P_Person_ID>435</P_Person_ID>
</Patient>

<Patient>
  <PT_Person_ID>235</PT_Person_ID>
  <MRN>342</MRN>
  <P_Person_ID>466</P_Person_ID>
</Patient>

<Patient>
  <PT_Person_ID>97</PT_Person_ID>
  <MRN>28</MRN>
  <P_Person_ID>2181</P_Person_ID>
</Patient>
</dataroot>

person.xml
P3. This solution uses a simple report to print out the visits.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<dataroot xmlns:od="urn:schemas-microsoft-com:officedata"
xmns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="Visit.xsd">

<Visit>
<OP_PT_Person_ID>21</OP_PT_Person_ID>
<Date>2001-01-01T00:00:00</Date>
</Visit>

<Visit>
<OP_PT_Person_ID>23</OP_PT_Person_ID>
<Date>2001-01-03T00:00:00</Date>
</Visit>

<Visit>
<OP_PT_Person_ID>23</OP_PT_Person_ID>
<Date>2006-01-05T00:00:00</Date>
</Visit>
</dataroot>
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2000/10/XMLSchema"
xmns:od="urn:schemas-microsoft-com:officedata">

<xsd:element name="dataroot">
<xsd:complexType>
<xsd:choice maxOccurs="unbounded">
<xsd:element ref="Visit"/>
</xsd:choice>
</xsd:complexType>
</xsd:element>

<xsd:element name="Visit">
<xsd:annotation>
<xsd:appinfo>
<od:index index-name="PrimaryKey" index-key="OP_PT_Person_ID Date " primary="yes" unique="yes" clustered="no"/>
<od:index index-name="OP_PT_Person_ID" index-key="OP_PT_Person_ID " primary="no" unique="no" clustered="no"/>
</xsd:appinfo>
</xsd:annotation>
<xsd:complexType>
<xsd:sequence>
<xsd:element name="OP_PT_Person_ID" minOccurs="0" od:jetType="text" od:sqlSType="nvarchar">
<xsd:simpleType>
<xsd:restriction base="xsd:string">
<xsd:maxLength value="5"/>
</xsd:restriction>
</xsd:simpleType>
</xsd:element>
<xsd:element name="Date" minOccurs="0" od:jetType="datetime" od:sqlSType="datetime" type="xsd:timeInstant"/>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:schema>
```
Chapter 11 Data Warehousing

Chapter Overview

The purpose of this chapter is to introduce students to the rationale and basic concepts of data warehousing from a database management point of view. We contrast operational and informational processing, and we discuss the reasons why so many organizations are seeking to exploit data warehouses for competitive advantage. We discuss alternative data warehouse architectures (especially the database architectures) and techniques for populating a warehouse.

Chapter Objectives

Specific student objectives are included in the beginning of the chapter. *From an instructor’s point of view, the objectives of this chapter are to:*

1. Establish the fact that many organizations today are experiencing an information gap. That is, they are drowning in data but starving for information.
2. Define data warehousing and describe four characteristics of a data warehouse.
3. Describe two major factors that drive the need for data warehousing as well as several advances in the field of information systems that have enabled data warehousing.
4. Contrast operational systems and informational systems from the viewpoint of data management.
5. Describe the basic architectures that are most often used with data warehouses.
6. Contrast transient and periodic data, and discuss how data warehouses are used to build a historical record of an organization.
7. Discuss the purposes of populating a data warehouse and the problems of data reconciliation.
8. Contrast data warehouses and data marts.
9. Describe and illustrate the dimensional data model (or star schema) that is often used in data warehouse design.
Key Terms

<table>
<thead>
<tr>
<th>Conformed dimension</th>
<th>Grain</th>
<th>Periodic data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data mart</td>
<td>Independent data mart</td>
<td>Real-time data warehouse</td>
</tr>
<tr>
<td>Data mining</td>
<td>Informational system</td>
<td>Reconciled data</td>
</tr>
<tr>
<td>Data visualization</td>
<td>Logical data mart</td>
<td>Relational OLAP (ROLAP)</td>
</tr>
<tr>
<td>Data warehouse</td>
<td>Multidimensional OLAP (MOLAP)</td>
<td>Snowflake schema</td>
</tr>
<tr>
<td>Dependent data mart</td>
<td>Online analytical processing (OLAP)</td>
<td>Star schema</td>
</tr>
<tr>
<td>Derived data</td>
<td>Operational data store (ODS)</td>
<td>Transient data</td>
</tr>
<tr>
<td>Enterprise data warehouse (EDW)</td>
<td>Operational system</td>
<td></td>
</tr>
</tbody>
</table>

Classroom Ideas

1. Discuss the importance of data warehousing in organizations today. Over 90 percent of large (Fortune 1000) companies have completed data warehouses or have a warehousing project underway. Ask your students to suggest reasons for this popularity.

2. Discuss job opportunities in data warehousing, business intelligence, and data mining. Numerous Web sites have job listings as well as newspaper advertisements.

3. Emphasize that a successful data warehousing project requires the integration of everything the students have learned throughout the database course (in fact, everything in the IS curriculum).

4. Discuss the idea of heterogeneous data (use Figure 11-1). Ask your students for reasons why such data are so commonplace and what problems they present.

5. Compare operational and informational systems using Table 11-1. Ask your students for examples of each type of system.

6. Compare the two-layer (Figure 11-2), independent data mart (Figure 11-3), dependent data mart and operational data store (Figure 11-4), and logical data mart (Figure 11-5) architectures.

7. Discuss the three-layer data architecture (Figure 11-6). Ask your students why it might be necessary to have both a reconciled data layer and a derived data layer.

8. Compare transient data (Figure 11-8) with periodic data (Figure 11-9). Explain how periodic data provide a historical record of events.

9. Discuss the steps in data reconciliation (Figure 11-10). Emphasize that this is generally considered to be the most complex challenge in data warehousing.

10. Discuss some of the typical data transformation functions (use Figures 11-11 and 11-12). Have your students suggest other practical examples.

11. Introduce components of a star schema (Figure 11-13) and discuss the example shown in Figures 11-14 and 11-15. Have your students help you diagram another example (university, football team, etc.).

Published by www.emagzine.org
12. Discuss some variations of the star schema (Figure 11-18).
13. Discuss conformed dimensions and how these could be used (Figure 11-17).
14. Discuss normalizing dimension tables (Figures 11-19 and 11-20).
15. Discuss slowly changing dimensions and some ways to handle this. Use Figure 11-21 as an example of one possible solution.
16. Discuss some of the ways end users can use a data warehouse or data mart (use Figures 11-22 and 11-23). Ask your students to suggest some advantages of these user interfaces.
17. Introduce the topic of data mining (use Table 11-4). If time permits, have your students read a recent article on data mining in a publication such as *DM Review* (available at www.dmreview.com).
18. Have your students register for the Teradata Student Network and show them how they can access extensive material on data warehousing and business intelligence.

**Answers to Review Questions**

1. Define each of the following terms:

   a. **Data warehouse** A subject-oriented, integrated, time-variant, non-volatile collection of data used in support of management decision-making processes (Inmon and Hackathorn, 1994).
   b. **Data mart** A data warehouse that is limited in scope, whose data is obtained by selecting and (where appropriate) summarizing data from the enterprise data warehouse.
   c. **Reconciled data** Detailed, historical data that are intended to be the single, authoritative source for all decision support applications and not generally intended to be accessed directly by end users.
   d. **Derived data** Data that have been selected, formatted, and aggregated for end-user decision support applications.
   e. **Online analytical processing** (OLAP) The use of a set of graphical tools that provides users with multidimensional views of their data and allows them to analyze the data using simple windowing techniques.
   f. **Data mining** Knowledge discovery using a sophisticated blend of techniques from traditional statistics, artificial intelligence, and computer graphics (Weldon 1996).
   g. **Star schema** A simple database design in which dimensional data are separated from fact or event data. A *dimensional model* is another name for star schema.
   h. **Snowflake schema** An expanded version of a star schema in which all of the tables are fully normalized.
   i. **Grain** The length of time (or other meaning) associated with each record in the table.
   j. **Conformed dimension** One or more dimension tables associated with two or more fact tables for which the dimension tables have the same business meaning and primary key with each fact table.
2. Match the following terms and definitions:

   a. transient data
   b. reconciled data
   c. periodic data
   d. data mart
   e. star schema
   f. data mining
   g. dependent data mart
   h. snowflake schema
   i. data visualization

3. Contrast the following terms:

   a. *Transient data; periodic data*  In transient data, changes to existing records are written over previous records, thus destroying the previous data content. In periodic data, the data is never physically altered or deleted once they have been added to the store.

   b. *Data warehouse; data mart; operational data store*  A data warehouse is an integrated and consistent store of subject-oriented data that are obtained from a variety of sources and formatted into a meaningful context to support decision making in an organization. A data mart is a data warehouse that is limited in scope and whose data are obtained by selecting and (where appropriate) summarizing data from the enterprise data warehouse. An operational data store is much different from a data warehouse or data mart because it is updatable, has a limited amount of historical data, and is available to operational users for use in decision support.

   c. *Reconciled data; derived data*  Reconciled data are intended to be the single, authoritative source for all decision-support applications and not generally intended to be accessed by end users; derived data have been selected, formatted, and aggregated for end-user decision support applications.

   d. *Fact table; dimension table*  Fact tables contain factual or quantitative data about a business such as units sold, orders booked, and so on. Dimensional tables hold descriptive data about the business.

   e. *Star schema; snowflake schema*  A star schema is a simple database design in which dimensional data are separated from fact or event data, while a snowflake schema is an expanded version of a star schema in which all of the tables are fully normalized.

   f. *Independent data mart; dependent data mart; logical data mart*  An independent data mart is populated with data extracted from the operational environment without the benefit of a reconciled data layer; a dependent data mart is populated exclusively from the enterprise data warehouse and its reconciled data layer. A logical data mart is created from a relational view of a data warehouse.
4. Five major trends that necessitate data warehousing in many organizations today:
   a. No single system of record
   b. Multiple systems are not synchronized
   c. Organizations want to analyze the activities in a balanced way
   d. Customer relationship management
   e. Supplier relationship management

5. Major components of a data warehouse architecture:
   a. *Operational data* Stored in the various operational systems throughout the organization (and sometimes in external systems)
   b. *Reconciled data* The type of data stored in the enterprise data warehouse
   c. *Derived data* The type of data stored in each of the data marts

6. List three types of metadata that appear in a three-layer data warehouse architecture, and briefly describe the purpose of each type:
   a. *Operational metadata* These are metadata that describe the data in the various operational systems (as well as external data) that feed the enterprise data warehouse. Operational metadata typically exist in a number of different formats, and they are unfortunately, often of poor quality.
   b. *Enterprise data warehouse (EDW) metadata* These metadata are derived from (or at least are consistent with) the enterprise data model. They describe the reconciled data layer. EDW metadata also describe the rules that are used to transform operational data to reconciled data.
   c. *Data mart metadata* These metadata describe the derived data layer. They also describe the rules that are used to transform reconciled data to derived data.

7. Four characteristics of a data warehouse:
   a. *Subject-oriented* A data warehouse is organized around the key subjects (or high-level entities) of the enterprise. Major subjects may include customers, patients, students, products, and time.
   b. *Integrated* The data housed in the data warehouse are defined using consistent naming conventions, formats, encoding structures, and related characteristics gathered from several internal systems of record and also often from sources external to the organization. This means that the data warehouse holds the one version of “the truth.”
   c. *Time-variant* Data in the data warehouse contain a time dimension so that they may be used to study trends and changes.
   d. *Nonupdatable* Data in the data warehouse are loaded and refreshed from operational systems, but cannot be updated by end users.
8. Five claimed limitations of independent data marts:

   a. A separate ETL process is developed for each data mart. This can yield costly redundant data and efforts.
   b. A clear, enterprise-wide view of data may not be provided because data marts may not be consistent with one another.
   c. Analysis is limited because there is no capability to drill down into greater detail or into related facts in other data marts.
   d. Scaling costs are excessive as each new application creates a separate data mart, which repeats all the extract and load steps.
   e. Attempting to make the separate data marts consistent generates a high cost to the organization.

9. Two claimed benefits of independent data marts:

   a. Allow for the concept of a data warehouse to be proved by working on a series of small, fairly independent projects.
   b. A reduction in the amount of time until a benefit from data warehousing is perceived by the organization, so that there is not a delay until all data are centralized.

10. Three types of operations that can be easily performed with OLAP tools:

    a. Slicing a cube
    b. Drill-down
    c. Data mining

11. List four objectives of derived data:

    a. Provide ease of use for decision support applications
    b. Provide fast response for predefined user queries or requests for information
    c. Customize data for particular target user groups
    d. Support ad-hoc queries and data mining and other analytic applications

12. Is the star schema a relational data model? Why or why not?

    The star schema is a denormalized implementation of the relational data model. The fact table plays the role of a normalized n-ary associative entity that links together the instances of the various dimensions. Usually, the dimension tables are in second normal form or possibly (but rarely) in third normal form. The dimension tables are denormalized and because they are not updated nor joined with one another, provide an optimized user view for specific information needs but could not be used for operational purposes.

13. Explain how the volatility of a data warehouse is different from the volatility of a database for an operational information system:
A major difference between a data warehouse and an operational system is the type of data stored. An operational system most often stores transient data, which are overwritten when changes to the data occur. Thus, the data in an operational system are very volatile. On the other hand, a data warehouse usually contains periodic data, which are never overwritten once they have been added to the store. A data warehouse contains a history of the varying values for important (dimensional) data.

14. Explain the pros and cons of logical data marts:

Pros:
   a. New data marts can be created quickly because no physical database or database technology needs to be acquired or created. Also, loading routines do not need to be written.
   b. Data marts are always up-to-date because data in a view are created when the view is referenced. Views can be materialized.

Con:
   Logical data marts are only practical for moderate-sized data warehouses or when high performance data warehousing technology is used.

15. What is a helper table and why is it often used to help organize derived data?

A star schema data mart is comprised of fact and dimension tables. Fact tables are completely normalized because each fact depends on the whole composite primary key and nothing but the composite primary key. Dimension tables may not be fully normalized. Helper tables in the data warehouse world act as associative entities in the conceptual model world to link instances of data in M:N relationships. The helper table acts as a way to normalize the relationship between the dimension data and the fact data, such as in the case of a multivalued dimension situation explained in Figure 11-15 in the text.

16. The characteristics of a surrogate key as used in a data warehouse or data mart:

All keys used to join the fact table to the dimension tables should be system assigned. The key should be simple as compared to the production or composite key. It is best to maintain the same length and format for all surrogate keys across the entire data warehouse, regardless of the business dimensions involved.

17. Time is almost always a dimension in a data warehouse or data mart because data marts and data warehouses record facts about dimensions over time. Date and time are almost always included as a dimension table, and a date surrogate key is usually one of the components of the primary key of the fact table. The time dimension is critical to most of the reporting and analysis needs that end users of the data warehouse have. Often, users will want to view how facts (such as sales) have changed over time or may want to compare one time period against another.
18. What is the purpose of conformed dimensions for different star schemas within the same data warehousing environment?

A conformed dimension is one or more dimension tables associated with two or more fact tables for which the dimension tables have the same business meaning and primary keys. Thus, conformed dimensions are important when there are multiple fact tables (often because there are multiple data marts) to be able to have consistent results across the marts and to be able to write queries that cut across the different marts.

Conformed dimensions allow users to:
   a. Share nonkey dimension data
   b. Query across fact tables with consistency
   c. Work on facts and business subjects for which all users have the same meaning

19. Can a fact table have no nonkey attributes?

Yes, this would be an example of a factless fact table. There are two general situations in which this might be useful: to track events and to inventory the set of possible occurrences.

20. In what way are dimension tables often not normalized?

Most dimension tables are not normalized so that for a given user group the dimension data are only one join away from associated facts. One example might be multivalued data, in which one could store multiple values by using several different fields. Another example would be the incorporation of data from other tables that are not part of the star schema but might be needed for analysis.

21. What is a hierarchy as it relates to a dimension table?

A dimension table often has a natural hierarchy among the rows. Some examples might be geographical hierarchies (markets within a state, states within a region) and product hierarchies (products within a product line). These hierarchies can be handled in two ways:
   a. Include all information for each level of the hierarchy in a single, denormalized table with a helper table (Figure 11-20)
   b. Normalize the dimension into a nested set of tables (one for each level of the hierarchy) with 1:M relationships between them

22. What is the meaning of the phrase “slowly changing dimension”?

Although data warehouses track data over time, the business does not remain static. We need to keep track of the history of values in order to record the history...
of facts with correct dimensional descriptions when the facts occurred. Dimension data changes slower than transactional data, thus we can consider dimensions to be slowly changing dimensions.

23. Explain the most common approach used to handle slowly changing dimensions.

Create a new dimension table row (with a new key) each time the dimension object changes and this new row will contain all the dimension characteristics. A fact row is associated with the key whose attributes apply at the time of the fact. This approach allows us to create as many dimensional object changes as necessary. It can become unwieldy if rows change frequently. We may also want to store the surrogate key value for the original object in the dimension row so that we can relate changes back to the original object.

24. One of the claimed characteristics of a data warehouse is that it is nonupdatable. What does this mean?

Nonupdatable means that data, once put in the data warehouse, are never changed (except to correct errors), but rather new versions of the same data may be stored.

25. In what ways are a data staging area and an enterprise data warehouse different? The data staging area contains only current, consolidated data from source systems whereas an enterprise data warehouse (EDW) contains time-stamped history.

Answers to Problems and Exercises

1. A possible field list for the new table could be: Student_No, Last_Name, First_Name, MI, Address, Telephone, Status, Dept, Hours, Insurance.

2. a. Transient (06/21)

<table>
<thead>
<tr>
<th>Key</th>
<th>Name</th>
<th>Major</th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td>Amy</td>
<td>Music</td>
</tr>
<tr>
<td>002</td>
<td>Tom</td>
<td>Business</td>
</tr>
<tr>
<td>003</td>
<td>Sue</td>
<td>Art</td>
</tr>
<tr>
<td>004</td>
<td>Joe</td>
<td>Business</td>
</tr>
<tr>
<td>006</td>
<td>Jim</td>
<td>Phys Ed</td>
</tr>
</tbody>
</table>

Transient (06/22)

<table>
<thead>
<tr>
<th>Key</th>
<th>Name</th>
<th>Major</th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td>Amy</td>
<td>Music</td>
</tr>
<tr>
<td>002</td>
<td>Tom</td>
<td>Business</td>
</tr>
</tbody>
</table>
b. It should be noted that the actual PK of the rows of this table is a combination of the original Key and the Date fields.

<table>
<thead>
<tr>
<th>Key</th>
<th>Date</th>
<th>Name</th>
<th>Major</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td>06/20</td>
<td>Amy</td>
<td>Music</td>
<td>C</td>
</tr>
<tr>
<td>002</td>
<td>06/20</td>
<td>Tom</td>
<td>Business</td>
<td>C</td>
</tr>
<tr>
<td>003</td>
<td>06/20</td>
<td>Sue</td>
<td>Art</td>
<td>C</td>
</tr>
<tr>
<td>004</td>
<td>06/20</td>
<td>Joe</td>
<td>Math</td>
<td>C</td>
</tr>
<tr>
<td>004</td>
<td>06/21</td>
<td>Joe</td>
<td>History</td>
<td>U</td>
</tr>
<tr>
<td>005</td>
<td>06/20</td>
<td>Ann</td>
<td>Engineering</td>
<td>C</td>
</tr>
<tr>
<td>005</td>
<td>06/21</td>
<td>Ann</td>
<td>Engineering</td>
<td>D</td>
</tr>
<tr>
<td>006</td>
<td>06/21</td>
<td>Jim</td>
<td>Phys Ed</td>
<td>C</td>
</tr>
</tbody>
</table>

Periodic (06/22)

<table>
<thead>
<tr>
<th>Key</th>
<th>Date</th>
<th>Name</th>
<th>Major</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td>06/20</td>
<td>Amy</td>
<td>Music</td>
<td>C</td>
</tr>
<tr>
<td>002</td>
<td>06/20</td>
<td>Tom</td>
<td>Business</td>
<td>C</td>
</tr>
<tr>
<td>003</td>
<td>06/20</td>
<td>Sue</td>
<td>Art</td>
<td>C</td>
</tr>
<tr>
<td>003</td>
<td>06/22</td>
<td>Sue</td>
<td>History</td>
<td>U</td>
</tr>
<tr>
<td>004</td>
<td>06/20</td>
<td>Joe</td>
<td>Math</td>
<td>C</td>
</tr>
<tr>
<td>004</td>
<td>06/21</td>
<td>Joe</td>
<td>History</td>
<td>U</td>
</tr>
<tr>
<td>005</td>
<td>06/20</td>
<td>Ann</td>
<td>Engineering</td>
<td>C</td>
</tr>
<tr>
<td>005</td>
<td>06/21</td>
<td>Ann</td>
<td>Engineering</td>
<td>D</td>
</tr>
<tr>
<td>006</td>
<td>06/21</td>
<td>Jim</td>
<td>Phys Ed</td>
<td>C</td>
</tr>
<tr>
<td>006</td>
<td>06/22</td>
<td>Jim</td>
<td>Bskt Weav</td>
<td>U</td>
</tr>
</tbody>
</table>
3. Millennium College
   a. Star Schema
   
   ![Star Schema Diagram](image)

   b. 500 course sections x 40 students per section x 30 periods (i.e., 3 semesters per year) = 600,000 rows (assuming 1 professor per course section)

d. There are several possible options that can be considered if a star schema is not mandated. The following suggestions are all aimed at creating a more normalized data model:

   - Professor does not have to relate directly to the fact table. Professor could snowflake off of the Section (who are the professors teaching a particular section?). Then, there would be no Professor PK in the fact table.
   - Course information could snowflake off of Section (put Course ID, Course Name, and Units in a separate table related to Section). This would normalize Course and Section data.
   - Similarly, Department information (Department ID and Department Name) could be snowflaked off of Professor.

   e. Dimensional segmentation is recommended. The student, section, and professor dimensions would be divided into two segments each, one for constant attributes and one for attributes that change. For example, the room
might change frequently for the section. This approach would eliminate a large number of rows in the dimension tables if the changing attributes were to change each semester.

4. Snowflake schema for Millennium College

This schema will have no impact on the size of the fact table, which remains the same as for the star schema.
5. Simplified Automobile Insurance Company
   a. Star schema:

   **INSURED PARTY**
   - Ins_Party_ID
   - Ins_Party_Name

   **COVERAGE ITEM**
   - Coverage_Key
   - Description

   **AGENT**
   - Agent_ID
   - Agent_Name

   **POLICY FACTS**
   - Ins_Party_ID
   - Coverage_Key
   - Agent_ID
   - Policy_ID
   - Date_Key
   - Policy_Premium
   - Deductible
   - No_of_Trans

   **POLICY**
   - Policy_ID
   - Type

   **PERIOD**
   - Date_Key
   - Fiscal_Period

   b. To correctly estimate the number of rows, we must make some additional assumptions regarding the fiscal periods and the frequency of changes to policies. We assume the following:
   1. The length of a fiscal period is one month
   2. The data mart will contain five years of historical data
   3. Approximately 5 percent of the policies experience some type of change each month

   Therefore, the expected number of rows is:
   
   \[ \text{0.05} \times 1,000,000 \times 10 \text{ coverage items} \times 2 \text{ insured parties} \times 5 \text{ years} \times 12 \text{ months per year} = 60,000,000 \]

   c. Total size of fact table:
   
   \[ 60,000,000 \text{ rows} \times 8 \text{ fields} \times 5 \text{ bytes} = 2,400,000,000 \text{ (2.4 Gigabytes)} \]
6. Simplified Automobile Insurance Company:
a. An initial version of the extended star schema:

![Extended Star Schema Diagram]

b. Unfortunately, the policy fact table in this schema has a design issue because premium and deductible are not dependent on CLAIM. This would result in two different “grains” or levels of detail in the same table, which is not permitted. A good solution is to create a separate star schema for claim facts (in this case, only claim_total) with all six dimension tables, along with the original star schema. We can then calculate the number of rows in the claims fact table as follows:

60 months x 2,000 claims per month = 120,000 rows in this additional fact table
7. Another dimension table was added with the department information. This will create a snowflake schema. The rationale for doing this is that users may also eventually want similar information for the Professor dimension. If we added the department information only to the Course_Section dimension, then this information would need to be replicated in the Professor dimension later.

8. While Ralph Kimball advocates for a dimensional model to solve all of the decision support needs of an organization, Inmon and Armstrong advocate for the creation of one central data warehouse and dependent data marts built from this. Kimball’s position is that a dimensional model, with fact tables and denormalized dimension tables, can be created to meet the needs of an entire organization; this model is far better than the traditional ER model. The opposing view holds that normalized tables are the way to build a data warehouse, because these tables can then be used to feed many different data marts tailored to individual users. Inmon states that building a dimensional model for an enterprise-wide application will result in much duplication of data and inconsistent data. While the dimensional model could be built for a data mart, the data mart must be dependent upon a data warehouse as a central repository.

9. In the initial solution, we assume that the customer is purchasing from a storage location. We also assume that there could be internal transfers both between plants and from plants to storage. Because not all dimensions are used in every movement event, the best solution for this exercise is a multiple fact-table star schema.
a. Initial solution:

b. Using generalization to simplify solution:

We could simplify this star schema substantially by re-designing the fact table to act more generically. Essentially, the re-designed fact table contains an origin key and a destination key. For example, if a customer was to purchase some items, the origin key would be a storage ID and the destination key would be the CustomerID. OriginRole and DestinationRole will be the name of the role for each fact related to the Location.
ObjectID (e.g., customer, vendor, plant). TxnType will label the type of transaction that occurred (e.g., sale, return, etc.). Also, note the use of an ObjectID as the surrogate key for the Location dimension.
11. Answers to problems 3, 5, 6, and 9 using dimensional modeling tool:

Problem 3: 

Problem 5:
Problem 6:

Problem 9 - initial solution:
Problem 9 - simplified solution:

Usefulness of the tool:
This modeling tool is useful and easy to use; it is very intuitive. There is a simple Help file that explains the interface and functions of the program. There were a few issues encountered that could not be surmounted: (1) Only one relationship between a dimension table and the fact table can be drawn; the tool will not permit multiple relationships. (2) There did not appear to be a way to adjust the size of the tables so that longer names could be used for table names; the tool just cuts off the display of longer named tables. (3) Attribute names were automatically placed in alphabetical order; there did not appear to be a way to place them in a designer-preferred order. Drawing tools for other diagrams (e.g., Visio, SmartDraw) now seem to come loaded with many sample or standard templates; it would be nice if this tool could also have standard templates for such schemas.
12. Because location is a hierarchy on Customer, it is best to snowflake Location off Customer with a helper table, though Location itself is a hierarchy (location within location), the approach in Figure 11-16 is not sufficient and a solution similar to that shown in Figure 11-17a is necessary.
b. In order to keep the history of the changes in Customer information, the model must be adjusted to store date elements in the Customer, helper (even the hierarchy could change over time), and location tables.

The chapter covers several methods for handling slowly changing dimensions. The preferred approach is to expand the PK of the Customer and Location dimension tables, and probably add one non-key attribute. The new PK would become the original surrogate key plus the Date/Time Stamp for when the change reflected by the values in the row of the dimension table occurred. The new non-key could be the date when the row is no longer true (end date). End date would be null or some very large value for the current row. The fact table still has just the original surrogate keys. The Period PK in the fact table is used to figure out which row of the Customer (or Location) table is relevant for that fact (i.e., the Period falls between the start and end dates of the relevant associated Customer row). The new date key in Customer would serve the same purpose as part of the foreign key in Location for linking a customer to changes in characteristics of its location.
13. Star schema for Fitchwood Insurance:

a. The time dimension is handled with the Date Dimension table, which includes a surrogate key, and attributes of DayOfWeek, Day, Month, Year, Qtr, and WeekdayFlag.

b. The multivalued attribute of customer address is handled in this schema with a slowly changing dimension (SCD) element of a start date as part of the surrogate key for the customer dimension table.

c. Territory is snowflaked off of Agent.

d. TransactionID indicates whether the transaction is initial commission, monthly commission, write policy, or discontinue policy.
14. Is further snowflaking required for the Fitchwood solution in question 13?

The answer to question 13 handles the changing customer address as a slowly changing dimension (SCD), thus there is no reason to snowflake to normalize further for the slowly changing dimension.

15. Revision of Exercise 13 and 14 to address the issue of agents changing territories over time. The solution uses a start date as part of the surrogate key for the agent dimension table to design for this slowly changing dimension (SCD).
16. Revision of Exercise 15 to address the issue of customers having relationships with other customers (e.g., spouse, etc.). The solution uses a helper table as shown in Figure 11-17 to accommodate the relationships in the new design.

17. The use of an OLAP tool is recommended. The OLAP cube could be built from the existing star schema. Specific scripts would have to be written in order to build the cube to accommodate drill-down capabilities. It would also be necessary to store aggregate data.

18. There is a vast amount of data mining that could be done using this case study. For example, what age groups purchase which policies? What age group of customers is most likely to keep a policy in force? What states have the best sales of policies, and how does this compare with the age demographics for the customers in that state?

The exercise of researching data-mining tools is left to the student.
Answers to Field Exercises

1. One of two approaches can be used for this exercise:
   
a. Arrange a field trip and interview several persons to obtain answers to the questions stated. This is the preferred approach because it enables students to obtain different perspectives concerning these questions.
   
b. Invite one of the key participants to visit your class and make a short presentation, followed by student questions. The author used this approach in a recent class and found it reasonably effective, although the answers represented the perspective of only the one person.

2. These Web sites contain a wealth of information concerning data warehousing. You can focus the search around the following issues:
   
a. What are some of the key data warehouse implementation issues today, and what advice do the data warehouse gurus give concerning these issues?
   
b. What jobs have been spawned by data warehousing, and what are typical salary ranges for these jobs?
   
c. What types of software are available to support data warehouse design, implementation, and maintenance?
   
d. Cite several case examples of data warehouse implementations and document the following: benefits obtained, lessons learned, and pitfalls to avoid.

Project Case Study

Case Questions

1. Some of the advantages that a hospital might realize from a data warehouse are:
   
a. Single, organization-wide view of data
   
b. Improved data quality and consistency
   
c. Faster and easier access to data
   
d. Use of powerful data analysis and data-mining tools

   All of these more technical advantages can lead directly to the improvements outlined in the question (e.g., data quality can vastly improve patient safety; data analysis and mining can improve clinical research; faster and easier access to data can directly help treatment efficiency).

2. A data mart could be used in the emergency room to keep patient information, including test results. It could also be used to keep a knowledge base of medical information. In general, a data mart can be optimized to support the specific needs of the emergency room (which likely has a need for more rapid access to
Chapter 11

3. For reasons explained in the text, Mountain View Community Hospital should avoid developing a series of independent data marts. However, an organization can develop a pilot (or prototype) data mart to investigate “proof of concept,” provided that this development is part of a well-orchestrated plan for data warehouse development. The advantage of this approach is that it can help secure the commitment of top managers and user groups. The disadvantage of this approach is that it may be difficult to avoid short-term pressures that may result in developing independent data marts.

4. One possible way to address this would be to code all records so that the identifying information for a patient is not stored with actual data in the same set of tables. Encryption can also be used, however, tight authentication mechanisms will have to be in place. (Look ahead to Chapter 13 for information on the administration of security approaches for databases.)

5. OLAP tools would be helpful, especially since one could drill-down into such things as treatment detail records from high-level balanced scorecard metrics. ROLAP would be a good choice because it provides the greatest flexibility. OLAP’s value is the ability to easily “slice and dice” and “drill-down” a view of data. Thus, OLAP helps to create high-level views of hospital operations and then the ability to isolate where data shows abnormalities or interesting results. This might be in quality of care, financials, customer satisfaction, or other metric areas.

6. There is an opportunity for several types of data-mining applications. For example, regression analysis could be used to look at historical data and discover trends. Sequence association could be employed to identify trends such as seasonal demands for services. Case-based reasoning could be used to help with determining what was most successful for diagnoses for various ailments.

7. A data mart or data warehouse would help to ensure accuracy of financial data because there would be a process to verify data before loading it into the warehouse. Systematic errors made in source systems would be identified. Reports would be timely once the warehouse was developed because front-end tools could be deployed to enable management to generate reports at any time. Also, an EIS application could be deployed on top of the data mart or data warehouse. Trend information could be obtained from data mining. A data warehouse also shows history (whereas operational systems often only show current state), and history provides an audit trail essential for compliance. Overall, a data warehouse forces an organization to have better documentation (metadata), which is critical for compliance.
Case Exercises

1. Summary data mart:
   Star schema:

   We assume that 150 treatment instances occur per month.

   Expected size:
   150 treatments x 36 periods = 5,400 rows
   No. of bytes per row = 5 + 3 + 2 + 3 + 5 = 18
   Total bytes = 5,400 x 18 = 97,200
Detailed data mart:

Star schema:

We assume the average patient census is meant to be 100, so two treatments per day per patient gives you 200 treatments per day.

Expected size:

200 treatments per day x 1,000 days = 200,000 rows
Bytes per row = 3 + 5 + 5 + 2 + 4 + 20 + 10 = 49
200,000 rows x 49 bytes per row = 9,800,000 bytes (9.8 Megabytes)

c. The primary key structure is based on this assumption. One way to overcome this limitation is to add the time of day when a treatment is performed to the fact table. Or not be concerned with treatment transactions in the fact table and be satisfied with a summary value of the total costs to apply the treatment, possibly multiple times, by the same physician to the same patient in the same day and with the final treatment result being all that matters.

d. MVCH should implement the detailed data mart as the data will be able to be summarized from the detail, and there would not be any issues encountered with trying to keep the detail and summary data marts synchronized.
e. For the summary data mart:

Find the total treatments performed by each physician in 2008:

Select physician.physician_name, sum(treatment_facts.monthly_total)
from physician, treatment_facts, period
where physician.physician_id = treatment_facts.physician_id
and period.period_id = treatment_facts.period_id
and period.year = '2008'
group by physician.physician_name;

For the detailed data mart:

Find the total cost of treatments for each patient:

Select patient.patient_name, sum(treatment_cost)
From patient, treatment_facts
Where patient.patient_id = treatment_facts.patient_id
Group by patient_name;

2. a.
Dimensions and Facts

<table>
<thead>
<tr>
<th>Table</th>
<th>Column</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Surgeon Dimension</td>
<td>Surgeon_ID</td>
<td>Primary Key</td>
</tr>
<tr>
<td></td>
<td>Surgeon_Name</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Department</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Years_Experience</td>
<td></td>
</tr>
<tr>
<td>Disposition Dimension</td>
<td>Disp_ID</td>
<td>Primary Key</td>
</tr>
<tr>
<td></td>
<td>Disp_Code</td>
<td>This field will contain several codes such as</td>
</tr>
<tr>
<td></td>
<td></td>
<td>cancelled, patient died or had a reaction to</td>
</tr>
<tr>
<td></td>
<td></td>
<td>blood</td>
</tr>
<tr>
<td></td>
<td>Reason</td>
<td>Reason for cancellation</td>
</tr>
<tr>
<td>Visit Dimension</td>
<td>Visit_ID</td>
<td>Primary Key</td>
</tr>
<tr>
<td></td>
<td>Patient_ID</td>
<td>FK to patient dimension (snowflake schema)</td>
</tr>
<tr>
<td></td>
<td>Patient_Age</td>
<td>Age at time of visit</td>
</tr>
<tr>
<td></td>
<td>Visit_Type</td>
<td>Outpatient or inpatient</td>
</tr>
<tr>
<td></td>
<td>Insurance_Co</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Acuity</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Diagnosis</td>
<td>Diagnosis for this visit</td>
</tr>
<tr>
<td>Patient Dimension</td>
<td>Patient_ID</td>
<td>Primary key</td>
</tr>
<tr>
<td></td>
<td>Patient_Name</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Patient_City</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Patient_State</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Patient_Zip</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Patient DOB</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Patient_Gender</td>
<td></td>
</tr>
</tbody>
</table>
### Operating_Room Dimension

<table>
<thead>
<tr>
<th>OR_ID</th>
<th>Primary Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location</td>
<td></td>
</tr>
</tbody>
</table>

### Surgery_Type Dimension

<table>
<thead>
<tr>
<th>Surgery_Key</th>
<th>Primary Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Type of surgery</td>
</tr>
<tr>
<td>Category</td>
<td>i.e. heart, ENT, etc.</td>
</tr>
</tbody>
</table>

### Time Dimension

<table>
<thead>
<tr>
<th>Timekey</th>
<th>Primary Key</th>
</tr>
</thead>
</table>

| Year |            |
| Quarter | |
| Month |            |
| WeekNumber | Number of week in year |
| WeekofMonth | Number of week in month (1-4) |
| DayofWeek | Sun-Sat |
| Day | Actual day of month (1-31) |

### Surgery_Fact

<table>
<thead>
<tr>
<th>Surgeon_ID</th>
<th>Foreign key to surgeon dimension</th>
</tr>
</thead>
<tbody>
<tr>
<td>Visit_ID</td>
<td>Foreign key to Visit dimension</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OR_ID</th>
<th>Foreign key to Operating_Room dimension</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disp_ID</td>
<td>Foreign key to disposition</td>
</tr>
<tr>
<td>Surgery_Key</td>
<td>Foreign key to surgery_type dimension</td>
</tr>
<tr>
<td>Timekey</td>
<td>Foreign key to time dimension</td>
</tr>
<tr>
<td>Surgery Duration</td>
<td>Length of the surgery</td>
</tr>
<tr>
<td>Surgery_Start</td>
<td></td>
</tr>
<tr>
<td>Surgery_end</td>
<td></td>
</tr>
<tr>
<td>Aesthia Duration</td>
<td></td>
</tr>
</tbody>
</table>
b. Star Schema:

![Star Schema Diagram]

```
c. Three star queries:

List the number of surgeries per week per OR:
select operating_room.location, time.week, count(*)
from operating_room, surgery_fact, time
where operating_room.or_id = surgery_fact.or_id
and time.timekey = surgery_fact.timekey
group by operating_room.location, time.week;

List the average surgery time per OR:
Select operating_room.location,
average(surgery_fact.surgery_duration)
From operating_room, surgery_fact
Where operating_room.or_id = surgery_fact.or_id;
```
List the number of negative patient reactions to blood transfusions by surgeon:
Here, we will assume that the disposition dimension table has a Disp_code of ‘nb’ to indicate this.

```sql
Select surgeon.surgeon_name, count(*)
From surgeon, surgery_fact, disposition
Where surgeon.surgeon_id = surgery_fact.surgeon_id
And disposition.disp_id = surgery_fact.disp_id
And disposition.disp_code = 'nb'
Group by surgeon.surgeon_name;
```

d. The development of a business case for this scenario is left to the student. One thing to focus on is cost savings as well as increased patient quality due to the ability to get accurate, timely reports from the data mart.
3. Emergency Room:

<table>
<thead>
<tr>
<th>Table</th>
<th>Column</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Physician Dimension</td>
<td>Physician_ID</td>
<td>Primary Key</td>
</tr>
<tr>
<td></td>
<td>Physician_Name</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Department</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Years_Experience</td>
<td></td>
</tr>
<tr>
<td>Disposition Dimension</td>
<td>Disp_ID</td>
<td>Primary Key</td>
</tr>
<tr>
<td></td>
<td>Disp_Code</td>
<td>This field will contain several codes such as discharged patient admitted, patient died, etc.</td>
</tr>
<tr>
<td></td>
<td>Reason</td>
<td>Any reasons</td>
</tr>
<tr>
<td>Visit_Fact</td>
<td>Physician_ID</td>
<td>FK to physician</td>
</tr>
<tr>
<td></td>
<td>Patient_ID</td>
<td>FK to patient dimension (snowflake schema)</td>
</tr>
<tr>
<td></td>
<td>TR_ID</td>
<td>FK to treatment room</td>
</tr>
<tr>
<td></td>
<td>Disp_id</td>
<td>FK to disposition</td>
</tr>
<tr>
<td></td>
<td>Timekey</td>
<td>FK to time</td>
</tr>
<tr>
<td></td>
<td>Visit Start</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Visit_End</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Patient_Age</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Diagnosis</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Insurance_Co</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Symptoms</td>
<td></td>
</tr>
<tr>
<td>Patient Dimension</td>
<td>Patient_ID</td>
<td>Primary key</td>
</tr>
<tr>
<td></td>
<td>Patient_Name</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Patient_City</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Patient_State</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Patient_Zip</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Patient_DOB</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Patient_Gender</td>
<td></td>
</tr>
<tr>
<td>Treatment_Room Dimension</td>
<td>TR_ID</td>
<td>Primary Key</td>
</tr>
<tr>
<td></td>
<td>Location</td>
<td></td>
</tr>
<tr>
<td>Treatment_Fact</td>
<td>Physician_ID</td>
<td>FK to physician</td>
</tr>
<tr>
<td></td>
<td>Patient_ID</td>
<td>FK to patient</td>
</tr>
<tr>
<td></td>
<td>Treatment_ID</td>
<td>FK to treatment</td>
</tr>
<tr>
<td></td>
<td>Timekey</td>
<td>FK to time</td>
</tr>
<tr>
<td></td>
<td>results</td>
<td></td>
</tr>
<tr>
<td>Time Dimension</td>
<td>Timekey</td>
<td>Primary Key</td>
</tr>
<tr>
<td></td>
<td>Year</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Quarter</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Month</td>
<td></td>
</tr>
<tr>
<td></td>
<td>WeekNumber</td>
<td>Number of week in year</td>
</tr>
<tr>
<td></td>
<td>WeekofMonth</td>
<td>Number of week in month (1–4)</td>
</tr>
<tr>
<td></td>
<td>DayofWeek</td>
<td>Sun – Sat</td>
</tr>
<tr>
<td></td>
<td>Day</td>
<td>Actual day of month (1–31)</td>
</tr>
</tbody>
</table>
Star Schema:

![Star Schema Diagram]
Dr. Z’s MS Center

<table>
<thead>
<tr>
<th>Table</th>
<th>Column</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Physician Dimension</td>
<td>Physician_ID</td>
<td>Primary Key</td>
</tr>
<tr>
<td></td>
<td>Physician_Name</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Department</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Years_Experience</td>
<td></td>
</tr>
<tr>
<td>Medications Dimension</td>
<td>Med_ID</td>
<td>Primary Key</td>
</tr>
<tr>
<td></td>
<td>Type</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>Visit_Fact</td>
<td>Physician_ID</td>
<td>FK to physician</td>
</tr>
<tr>
<td></td>
<td>Patient_ID</td>
<td>FK to patient dimension (snowflake schema)</td>
</tr>
<tr>
<td></td>
<td>Timekey</td>
<td>FK to time</td>
</tr>
<tr>
<td></td>
<td>Visit Start</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Visit_End</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Patient_Age</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Insurance_Co</td>
<td></td>
</tr>
<tr>
<td>Med_Fact</td>
<td>Patient_ID</td>
<td>FK to patient</td>
</tr>
<tr>
<td></td>
<td>Med_ID</td>
<td>FK to medications</td>
</tr>
<tr>
<td></td>
<td>Dosage</td>
<td></td>
</tr>
<tr>
<td>Patient Dimension</td>
<td>Patient_ID</td>
<td>Primary key</td>
</tr>
<tr>
<td></td>
<td>Patient_Name</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Patient_City</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Patient_State</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Patient_Zip</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Patient_DOB</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Patient_Gender</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Last_MRI_Date</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Last_MRI_Res</td>
<td></td>
</tr>
<tr>
<td>Time Dimension</td>
<td>Timekey</td>
<td>Primary Key</td>
</tr>
<tr>
<td></td>
<td>Year</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Quarter</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Month</td>
<td></td>
</tr>
<tr>
<td></td>
<td>WeekNumber</td>
<td>Number of week in year</td>
</tr>
<tr>
<td></td>
<td>WeekofMonth</td>
<td>Number of week in month (1-4)</td>
</tr>
<tr>
<td></td>
<td>DayofWeek</td>
<td>Sun – Sat</td>
</tr>
<tr>
<td></td>
<td>Day</td>
<td>Actual day of month (1-31)</td>
</tr>
<tr>
<td>Symptom Dimension</td>
<td>Symptom_ID</td>
<td>PK</td>
</tr>
<tr>
<td></td>
<td>Description</td>
<td></td>
</tr>
</tbody>
</table>
### Star Schema:

<table>
<thead>
<tr>
<th>Table</th>
<th>Patient_ID</th>
<th>FK to patient</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lipid_Fact</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Timekey</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LDL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Trig</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HDL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chol</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clinic_Fact</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Timekey</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Blood_pressure</td>
<td></td>
<td></td>
</tr>
<tr>
<td>weight</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Advisory_Fact</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Timekey</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Advisory</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Symptom_Fact</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Patient_id</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Timekey</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Symptom_ID</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Advisory</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lipid_Fact</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Timekey</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LDL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Trig</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HDL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chol</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clinic_Fact</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Timekey</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Blood_pressure</td>
<td></td>
<td></td>
</tr>
<tr>
<td>weight</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Advisory_Fact</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Timekey</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Advisory</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Symptom_Fact</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Patient_id</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Timekey</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Symptom_ID</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Advisory</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Published by <www.emagzine.org>
Chapter 12  Data Quality and Integration

Chapter Overview

This chapter reviews the importance of data quality and integration in today’s modern organizations. The chapter begins with an in-depth discussion of data quality: what it is, what the state of data quality is in most organizations, and how data quality can be improved. The data quality discussion forms the foundation for understanding how disparate sources of data are consolidated into an integrated view for decision making and business intelligence activities. The chapter also covers data integration in terms of data federation, data propagation, and data consolidation [via extract-transform-load (ETL) processes used in data warehousing]. The chapter concludes with a brief summary of application software used in data reconciliation activities for data integration.

This chapter refers to foundational material covered in Chapter 11 Data Warehousing, as well as to data modeling (covered in Chapters 3 through 6) and SQL (covered in Chapters 7 and 8). Students will gain the most from this chapter’s topics if these other chapters are covered before beginning study of data quality and integration.

Chapter Objectives

Specific student learning objectives are included at the beginning of the chapter. From an instructor’s point of view, the objectives of this chapter are to:

1. Impart to the student a greater appreciation for the importance of data quality in organizational information systems.
2. Provide a framework for developing a data quality program in an organization.
3. Understand the critical importance of data quality and some of the key steps that can be taken to improve data quality.
4. Provide examples and explanations regarding the challenges involved in presenting a consolidated view of data in organizations and to explain why many organizations are “drowning in data but starving from information.”
5. Provide a foundation to understanding data integration in general, and the extract-transfer-load (ETL) process in particular as an example of specific data integration for data warehousing.
6. Discuss the problems of data reconciliation.

Key Terms

<table>
<thead>
<tr>
<th>Aggregation</th>
<th>Data steward</th>
<th>Refresh mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Changed data capture (CDC)</td>
<td>Data transformation</td>
<td>Selection</td>
</tr>
<tr>
<td>Data federation</td>
<td>Incremental extract</td>
<td>Static extract</td>
</tr>
<tr>
<td>Data governance</td>
<td>Joining</td>
<td>Update mode</td>
</tr>
<tr>
<td>Data scrubbing</td>
<td>Master data management (MDM)</td>
<td></td>
</tr>
</tbody>
</table>
Classroom Ideas

1. Inexperienced students are likely to underestimate the problems associated with establishing adequate data quality. A discussion of the various areas that must be considered, along with examples, should be conducted. Encourage students to provide examples of both poor and good data quality instances from current events or news reports. Examine through discussion how these instances of poor data quality might have been prevented with more attention to data quality in the organization(s) or what some of the contributing factors were to good data quality in this situation.

2. If possible, find video/movie clips to illustrate the issues of data quality or data integration in organizations. (For example, in *Star Trek II: The Wrath of Khan*, Captain James Kirk uses the bridge computers of the Enterprise to control the computers on the starship Reliant hijacked by Khan. This act is only possible through the integrated data and control systems of Federation starships). Use the visual medium as a means of introducing the issue and raising awareness of data quality concerns in organizations to your students.

3. Ask your students to find movies that provide examples of data quality or data integration and to share these movie titles with the rest of the class. Break the class into small teams to discuss how the poor examples might have been avoided, or what factors might have contributed to the success of the good examples. Encourage teams to produce a small skit showing the results of their discussion.

4. Discuss the steps in data reconciliation (Figure 12-2). Emphasize that this is generally considered to be the most complex challenge in data warehousing (covered previously in Chapter 11).

5. Discuss some of the typical data transformation functions (use Figures 12-3 and 12-4). Have your students suggest other practical examples.

6. Everyone has likely seen poor quality data. Ask students to bring examples to class (e.g., redundant credit card application mailings, possibly with inaccurate name spellings; downloaded sports statistics with obvious, spurious values; personal databases, spreadsheets, etc.). In class, with the permission of a student, you might do a simple data profiling (distribution, end points, descriptive statistics) of some attributes to help find poor quality data.

Answers to Review Questions

1. Define each of the following key terms:
   a. **Static extract** A method of capturing a snapshot of the required source data at a point in time.
   b. **Incremental extract** A method of capturing only the changes that have occurred in the source data since the last capture.
   c. **Data steward** A person assigned the responsibility of ensuring that organizational applications properly support the organization’s enterprise goals.
   d. **Master data management** The disciplines, technologies, and methods to ensure that currency, meaning, and quality of reference data within and across various subject areas.
e. **Refresh mode** An approach to filling the data warehouse that employs bulk rewriting of the target data at periodic intervals.

2. Match the following terms and definitions:
   a. data transformation
   b. data scrubbing
   c. selection
   d. data steward
   e. changed data capture

3. Contrast the following terms:
   a. *Static extract; incremental extract* A static extract provides a capture of the source data at a point in time, while an incremental extract provides the changes that have occurred in the source data since the last capture.
   b. *Data scrubbing; data transformation* Data scrubbing uses pattern recognition and other artificial intelligence to upgrade the quality of raw data before they are transformed and moved to the data warehouse. Data transformation converts data from the format of the source operational systems to the format of the enterprise data warehouse.
   c. *Consolidation; federation* Consolidation is a technique for bringing together many disparate data sources into a single, authoritative source for data that support decision making. Consolidation is typified by the extract-transform-load (ETL) process used by most data warehouse technologies. Federation is a technique for data integration that provides a virtual view of integrated data without actually creating one centralized database.
   d. *ETL; master data management* ETL is the extract-transform-load process used by most data warehouse technologies to consolidate many data sources into a single, consolidated data warehouse. Master data management (MDM) refers to the disciplines, technologies, and methods to ensure the currency, meaning, and quality of reference data within and across various subject areas. MDM determines the best source for each piece of reference data, and then affords access to this “golden record” to the applications that need to use it. MDM uses a subset of integrated data while ETL usually provides much more integrated data in a data warehouse.

4. Characteristics of quality data:
   - **Uniqueness** Each entity exists no more than once within the database and there is a key that can be used to uniquely access each entity.
   - **Accuracy** The degree to which any datum correctly represents the real-life object it models.
   - **Consistency** Values for data in one data set (database) are in agreement with the values for related data in another data set (database).
   - **Completeness** All data that must have a value does have an assigned value.
   - **Timeliness** Meeting the expectation for the time between when data are expected and when they are readily available for use.
   - **Currency** The degree to which data is recent enough to be useful.
Conformance  Whether the data is stored, exchanged, or presented in a format that is as specified by its metadata.

Referential integrity  Data referring to other data are unique and satisfy requirements to exist (that is, satisfies any mandatory one or optional one cardinalities).

5. The effect of Sarbanes-Oxley Act on the need for organizations to improve data quality:

In a nutshell, the Sarbanes-Oxley Act (SOX) requires organizations to be accountable to their stakeholders regarding their operations and financial decisions; at the present time, this applies mainly to publicly-traded organizations. As some experts have indicated, information technologies and systems provide a means for organizations to comply with the various regulatory sections of SOX. Data are the heart and lifeblood of most information systems. In particular, various sections of the SOX act yield requirements for organizations to measure and improve metadata quality; ensure data security; measure and improve data accessibility and ease of use; measure and improve data availability, timeliness, and relevance; measure and improve accuracy, completeness, and understandability of general ledger data; and identify and eliminate duplicates and data inconsistencies. The SOX act could provide motivation for many organization executives to treat data quality as an important strategic and operational goal.

6. Characterize the state of data quality in most organizations.

The state of data quality in most organizations is “problematic, or even unacceptable,” according to expert assessments. Four reasons have contributed to the decline of data quality in organizations: (1) extensive and continued use of external data sources where the organization lacks control over the incoming data quality; (2) redundant data storage and inconsistent metadata; (3) data entry problems created through poor data capture controls; and (4) lack of organizational commitment to data quality as an organizational issue. Without a strong organizational commitment to data quality, organizations will experience the consequences of the industry adage: garbage in, garbage out (GIGO).

7. Explain four reasons for poor data quality in organizations:

External data sources  Organizations continue to rely on many outside sources for data—Web forms, XML channels from business-to-business sources, and databases from external organizations (e.g., mailing lists, census data, etc.). Organizations have little to no control over the completeness, accuracy, timeliness, or compatibility of this data with their internal data. Thus, organizations experience issues with data quality in their internal systems when this external data is brought into the organization without some form of data quality audit or control.

Redundant data storage and inconsistent metadata  Often, organizations have data stored in many different formats across the organization: spreadsheets, desktop databases, legacy databases, data marts, data warehouses, and other data repositories. This varied
data may be redundant, inconsistent, and incompatible. If the metadata are wrong (for example, a bad algorithm or formula in a spreadsheet), the data will be wrong as well.

Data Entry: Organizations have not always taken advantage of placing integrity controls, valid value controls, and other data quality controls within the database definitions. Thus, the control for data integrity has typically fallen to the user interface of various applications, which may or may not take advantage of automatically filling in stored data or using drop-down selection boxes.

Lack of organizational commitment: Organizations have failed to recognize the importance of data quality or doubt that there will be a positive return on investment if efforts to improve data quality are undertaken. Thus, organizations have not made the commitment or invested resources to address data quality in organizational systems.

8. What is data profiling and what role does it play in a data quality program?

A data profile is a statistical profile of the data values for each field in each table in the database. The statistical profile will allow people to view the patterns of data values (for example, distribution, outliers, frequencies) and to analyze them to see if the distribution makes sense. Data values can also be reviewed for obscure or extreme values. Data values can also be reviewed against known (and supposedly, operating) business rules as a check against data entry and processing routines.

Data profiling is one step of a data quality audit and a data quality program in an organization. Specialized tools (such as Informatica’s Power Center) are available to permit periodic data profiling of an organization’s database and applications. Data profiling serves in the role of a check against organizational process controls over data entry and maintenance activities.

9. How can data capture processes be improved to improve data quality?

According to Inmon (2004), there are several actions that can be taken at the original data capture step:

a. Enter as much of the data as possible via automatic, not human, means (e.g., retrieval from card swipe, current records, etc.).

b. Data that must be entered manually should be selected from preset options (e.g., drop-down boxes, etc.).

c. Create consistent screen layouts, easy to follow navigation paths, clear data entry masks and formats, and minimally use obscure codes in the user interface.

d. Entered data should immediately be checked for quality against data in the existing database. If there are issues, immediate and understandable feedback should be given to the human operator.
10. What is the function of a data steward?

A data steward manages a specific logical data resource or entity (e.g., customer, product, or facility) for all business functions and data processing systems that originate or use data about the assigned entity. A data steward coordinates all data definitions, quality controls, and improvement programs, access authorization, and planning for the data entity, for which he or she is responsible. Data stewards have the responsibility to ensure that organizational applications are properly supporting the organization’s enterprise goals. They must also ensure that the data captured are accurate and consistent throughout the organization so that users can rely on them.

11. How does data stewardship relate to data governance?

Data governance is a comprehensive program in organizations that usually guides data quality initiatives, data architecture, data integration and master data management, data warehousing, business intelligence, and other data-related matters. Data governance oversees data stewardship in an organization. Data stewardship is a specialized task, whereas data governance exerts guidance over activities to develop a unified management of data across the enterprise and participated in by enterprise decision makers.

12. What are the characteristics of a high-quality data model?

a. Entity types should represent, and be named after, the underlying nature of an object, not the role it plays in a particular context.
b. Entity types should be part of a subtype/supertype hierarchy in order to define a universal context for the data model.
c. Activities and associations should be represented by (event) entity types, not relationships. This principle allows all relationships to be many-to-many and data carrying, so that the association is represented by an associative or intersection-entity type.
d. Relationships should be used to represent only the involvement of entity types with activities or associations.
e. Candidate attributes should be suspected of representing relationships to other entity types. This principle says that any attribute may be a reference (foreign key) to another entity type that contains that attribute, along with others.
f. Entity types should have a single attribute as their primary unique identifier (e.g., a surrogate ID). This should be artificial, and not changeable by users.

13. What are the major differences between the data federation and the data propagation forms of data integration?
Data federation provides a virtual view of integrated data without actually bringing all of the data into one physical, centralized database. Data propagation duplicates data across databases, usually with near-real time update of changes throughout the organization.

14. What distinguishes master data management from other forms of data integration?

A distinguishing feature of master data management (MDM) is that it focuses on only the key reference data that is commonly referenced more frequently than other data in the organization. In essence, MDM operates on only a subset of the data in an organization.

15. List six typical characteristics of reconciled data:

[Note to instructor: This use of the term reconciled data refers to the result of the extract-transform-load (ETL) process]

a. **Detailed** The data are at a detailed level. This provides maximum flexibility for various user communities to structure the data to best suit their needs.

b. **Historical** The data are periodic when they are intended to provide a historical perspective.

c. **Normalized** The data are fully normalized. Data that are normalized provide greater flexibility of use than denormalized data. Denormalization is not necessary to improve performance because reconciled data are usually accessed periodically using batch processes.

d. **Comprehensive** Reconciled data reflect an enterprise-wide perspective whose design conforms to the enterprise data model.

e. **Timely** Except for real-time data warehousing, data need not be near-real time, however, data must be current enough so that decision making can react in time.

f. **Quality controlled** Reconciled data must be of unquestioned quality and integrity because they are summarized into the data marts and used for decision making.

16. List and briefly describe the steps in the data reconciliation process:

The overall steps in the data reconciliation process are as follows:

**Mapping and metadata management** The data needed in the data warehouse are identified and mapped back to the source data to be used to create the data warehouse.

**Capture/extract** The data for the data warehouse are actually obtained from the data sources.

**Cleanse/scrub** The data for the data warehouse are processed to identify errors or problems in the data. Data with errors are rejected from the ETL process and messages/flags are sent to the source systems for fixing of the errors. The cleansing/scrubbing process precedes any transformations of the source data into formats required by the data warehouse.
Transform The data for the warehouse are converted from the format of the source operational databases into the format of the enterprise data warehouse. (Note: A data warehousing trend is for the transformation process to follow the loading process, thus making the overall process more of an extract-load-transform process rather than extract-transform-load process).

Load and index Data for the warehouse are now loaded into the warehouse and indexed.

17. List five errors and inconsistencies that are often found in operational data:
   a. Misspelled names and addresses
   b. Impossible or erroneous dates of birth
   c. Fields used for purposes in which they were never intended
   d. Mismatched addresses and area codes
   e. Missing data

18. Explain how the phrase “extract, transform, and load” relates to the data reconciliation process:

   Reconciled data are the result of the extract, transform, and load process. The process results in a single source for data that support decision making within the organization. The data layer of the warehouse that results from this process is ideally detailed, historical, normalized, comprehensive, and timely. In addition, all of the data are quality controlled through the extract, transform, and load process.

19. List the common tasks performed during data cleansing:
   a. Decoding data to make them understandable for data warehousing applications
   b. Reformatting and changing data types and performing other functions to put data from each source into the standard data warehouse format ready for transformation
   c. Adding time stamps to distinguish values for the same attributes over time
   d. Converting between different units of measure
   e. Generating primary keys for each row of a table
   f. Matching or merging separate extractions into one table or file
   g. Logging errors detected, fixing those errors, and reprocessing corrected data without creating duplicate entries
   h. Finding missing data to complete the batch of data necessary for subsequent loading

Student answers will vary depending on what each student observed during this game. This can be a good exercise to use in-class with students as a fun way to motivate issues involved in checking data quality.

21. Describe some field-level and record-level data transformations that often occur during the ETL process for loading a data warehouse:

Field-level transformations are conversions of data from a source record format to a different format in a target record. Field-level transformations can be single-field or multifield. An example of a single-field transformation is the conversion of an English measurement (e.g., inches) to a metric representation (e.g., centimeters). An example of a multifield transformation is the conversion of a single product code into its two components of product and brand code.

Record-level functions include selection, joining, normalization, and aggregation. Selection, also known as subsetting, refers to getting a subset of the source data into the warehouse according to pre-set criteria, such as a date. Joining brings together data from various sources into a single table or view. Normalization refers to restructuring relations to remove problems with adding, deleting, or updating instances of data. Aggregation focuses on summarizing data at the detailed level into meaningful combinations for decision making (e.g., monthly sales total by store).

Answers to Problems and Exercises

Problems 1 through 5 are based on the Fitchwood Insurance case study, originally introduced in the textbook coverage of Data Warehousing (Chapter 11).

1. The OLTP system data for the Fitchwood Insurance Company is in a series of flat files. What process do you envision would be needed in order to extract the data and create the ERD shown in Figure 11-26? How often should the extraction process be performed? Should it be a static extract or an incremental extract?

Based upon the description of the case study, the extraction would have to be done on a weekly basis. The extraction would be done after the backup of the system on Friday nights. Essentially, the flat files would be exported to comma-delimited ASCII files containing only the information needed for the ERD in Figure 11-26. These ASCII files then would be transferred via FTP to the Unix system. Once this is done, they could then be used to load an initial set of tables using SQL Loader. From this set of tables, the star schema could be populated. Regarding incremental versus static extract, it is difficult to determine without knowing the volume of data. However, based upon my experience with legacy systems and the inconsistencies in such a system, it might be best to perform a static extract each weekend because data that you would not expect to change might.
2. What types of data pollution/cleansing problems might occur with the Fitchwood OLTP system data?

Most of the data pollution problems mentioned in the chapter could occur with this data set. The most likely concerns are missing and duplicate data as well as inconsistencies (for example, different PKs for the same policies or different hire dates for different agents that might be legitimate because they were “hired” on different dates to work in different product lines). It is also possible that different systems have different rules for creating computed values (for example, different insurance products might have different rules for using face value and commissions to calculate the amount paid agents). Territories might have different geographical boundaries across the source systems. Even more issues are possible.

3. Research some tools that perform data scrubbing. What tool would you recommend for the Fitchwood Insurance Company?

In the chapter, Table 12-4 has a good list of tools available for data reconciliation. Students might also want to perform a search using a search engine such as Google to obtain information about other tools. Web sites for organizations such as the Data Warehousing Institute and journals such as *DM Review* and *Intelligent Enterprise* also often have materials about vendors related to the topics addressed by these organizations.

4. What types of data transformations might be needed in order to build the Fitchwood data mart?

Record-level selection functions would have to be performed as part of the extraction process. Most of the aggregation would be done through queries to the star schema. If Fitchwood operates in several countries, language and currency conversions will be necessary; currency conversions are necessary to transform all currency values into one monetary unit. Some of the data pollution issues raised in Problem and Exercise 2 suggest some transformations that are needed, for example, to get all agents into a consistent view of territories. Some attributes, like LastRedistrict, may have to be computed from raw transactional data in the source systems.

5. After some further analysis, you discover that the commission field in the Policies table is updated on a yearly basis to reflect changes in the yearly commission paid to agents on existing policies. Would this change the way in which you extract and load data into the data mart from the OLTP system?

This new information would change the way that we extract and load data only if we use an incremental extract. In this case, we might want to perform a complete extract when the policy table changes yearly. If commission is treated as a dimensional value (not a fact-table value), then it will have to be managed as a slowly changing dimension (SCD) using one of the schemes outlined in the chapter on data warehousing.
6. The Pine Valley databases for this textbook (one small version illustrated in queries throughout the text and a larger version) are available to your instructor to download from the book’s Web site. Your instructor can make those databases available to you. Alternatively, these and other databases are available at www.teradatastudentnetwork.com (your instructor will tell you the login password and you will need to register, then create an SQL Assistant login for the parts of this question). There may actually be another database your instructor wants you to use for this series of questions. No matter how you gain access to a database, answer the following exercises for that database.

[Note to instructor: Specific student answers will vary depending on the database used as a basis of this exercise, however the basic elements of each answer will be similar to the sample solution below. The sample solution is based on the small Pine Valley Furniture Company database produced in MS Access.]

a. Develop a plan for performing a data quality audit on this database. Base your plan on the seven characteristics of quality data, on other concepts introduced in the chapter, and on a set of business rules you will need to create for this database. Justify your plan.

Plan for data quality audit on PVFC Database (small):

1. Review the characteristics of quality data and compare the implemented database against each characteristic. Document results of this review and be sure to highlight any management concerns or issues.

Uniqueness Does each entity exist only once? Does each entity have a unique key? If an entity exists more than once, is there system or design documentation that explains why and how the data is synchronized between the multiple instances?

Accuracy Are order sales values accurate according to accounting records? Does the assignment of salespeople to territories reflect current human-resource assignments? Are employee records current with human-resource and payroll records? Are vendor records accurate according to accounting records?

Consistency Are the supervisory relationships between employees properly noted in the database and are the employee characteristics consistent between instances of the records? Does the Product Finish value in the Product table map consistently to the proper Product Line in the Product Line table?

Completeness All data that must have a value does have an assigned value. Have the raw materials for each product been identified and accurately entered in the database? Are all employee supervisors noted in the Employee table? Are employee skills current in the database? Do the skills noted in the skills table completely reflect the HR needs of PVFC?

Timeliness How quickly are orders posted to the database? What span of order dates are in the current database? How often are order records archived for historical storage (or used in the data mart or data warehouse)?

Currency When were customer and vendor addresses last checked for currency?
How frequently is the standard cost of Raw Materials reviewed? How do raw-materials standard costs compare to product standard prices and vendor-supply unit prices?

**Conformance** Does the data presented in the reports, queries, and screens, match the format specified by its metadata? Are currency values within acceptable ranges? Are quantity values within acceptable ranges? Are state codes representative of valid state values?

**Referential integrity** Does the data in each associative entity table properly correspond to the tables on either side of it? Does a table with a foreign key properly refer back to an existing record in its reference table? For example, does each order line properly refer to existing products in the product table and orders in the order table? The following tables should also be reviewed for referential integrity: Produced_In (Work_Center, Product), Works_In (Work_Center, Employee), Has_Skill (Employee, Skill), Does_Business_In (Customer, Sales_Territory), Supplies (Vendor, Raw_Material), Uses (Product, Raw_Material), Product (Product_Line), Salesperson (Sales_Territory).

2. Inspect the database tables for valid values for fields. If possible, utilize software to perform statistical profiling of the data values so that patterns may be analyzed for issues in the data (e.g., extreme values, missing values, outliers, etc.). Document results of inspection and software analysis. Highlight issues for management review.

3. Review business rules of PVFC and compare the data found in the database to the business rules (Refer to Chapters 3, 4, and 5 for documented business rules of this database. One example is that each employee has one supervisor, except for managers, who have no supervisor). Document results of review and highlight issues for management review.

4. Render an assessment of the state of data quality in this database. List and briefly explain the data quality concerns for management action in terms of highest risk to lowest risk.

**Justification for data quality audit plan on PVFC Database (small):**

This data quality audit plan provides a comprehensive review of the quality of the data in the PVFC database. Guidance is provided regarding the expectations of data quality in the database and a clear framework is provided for comparing the data contents against the data quality expectations. At each step of the plan, documentation of the results of the audit is produced for management review. Additionally, at the end of the audit, an assessment of the data quality is provided along with a management action list.

b. Perform your data quality audit plan for one of the tables in the database (pick the table you think might be the most vulnerable to data quality issues). Develop an audit report on the quality of data in this table.
Table in PVFC Database (small): Product

1. Characteristics of quality data:

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Comments</th>
<th>+ or -</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uniqueness</td>
<td>Product table exists only once in database; it has unique PK</td>
<td>+</td>
</tr>
<tr>
<td>Accuracy</td>
<td>Product standard price is accurate according to accounting department audit of purchase invoices. Description of Product ID 2 appears to be misspelled (coffe table rather than coffee table?)</td>
<td>-</td>
</tr>
<tr>
<td>Consistency</td>
<td>Existing product finish values match the product finish expectations for the Product Line assigned to each product. Current records are consistent, but there are no entry controls to maintain consistency in future.</td>
<td>+</td>
</tr>
<tr>
<td>Completeness</td>
<td>Current records are complete but there are no entry controls to require fields to be entered other than Product_ID</td>
<td>-</td>
</tr>
<tr>
<td>Timeliness</td>
<td>Ok</td>
<td>+</td>
</tr>
<tr>
<td>Currency</td>
<td>Ok</td>
<td>+</td>
</tr>
<tr>
<td>Conformance</td>
<td>Ok</td>
<td>+</td>
</tr>
<tr>
<td>Referential Integrity</td>
<td>Current products in the Product table are properly referenced in the Product_Line, Produced_in, and Order_Line tables. As there are no records in the Raw_Materials table, there are no records to match products to in the Uses table. Referential integrity controls appear to be functioning properly</td>
<td>+</td>
</tr>
</tbody>
</table>

A review of the Product table against the characteristics of quality data shows that the data is of fairly-high quality at the moment. PVFC management should pay attention to instituting some data entry controls to ensure complete entry of information beyond the primary key field for each record in this table. Address values should be reviewed periodically for currency and accuracy, as these can often change over time and may lead to inconsistent data throughout the organization. Additionally, PVFC may wish to automate some form of spell-checking on product description values or schedule a regular review process for this field to ensure errors are caught early in the process.

2. Inspection of data values of Product table:
Data values for the product table appear to be within acceptable ranges of values, with the exception of the spelling error for the Product ID 2 item.

3. Review of business rules:

The Product table appears to be operating in conformance with known PVFC business rules.

4. Assessment of data quality in Product table:

The data quality of the current Product table is good to very good. There are some areas of concern regarding completeness and accuracy that need attention to ensure that future data in this table (and the database) will maintain such a good level of data quality. Following is a brief list of action items for PVFC management:

• Institute required entry of fields in the product table beyond the primary key field. The assignment of price should use a default value if a standard price is not known at the time of entering a new product item.

• Consider automating the description of products along agreed upon dimensions. For example, if you know you only produce desks, tables, and entertainment centers, you may wish to break up the product description field into smaller subsets, like, type=desk, kind=computer, that can be selected from a drop-down list (or other automated selection mechanism) so that errors in data entry may be eliminated or at least mitigated.

c. Execute your data profile plan for a set of three or four related tables. Develop an audit report on the quality of data in these tables.

Tables in PVFC Database (small): Product, Product Line, Uses, Raw_Materials

1. Characteristics of quality data:

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Comments</th>
<th>+ or -</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uniqueness</td>
<td>Each table exists only once in database; each table has unique PK</td>
<td>+</td>
</tr>
<tr>
<td>Accuracy</td>
<td>Data appear to be accurate, but low in volume</td>
<td>-</td>
</tr>
<tr>
<td>Consistency</td>
<td>Current records are consistent, but there are no entry controls to maintain consistency in future</td>
<td>-</td>
</tr>
<tr>
<td>Completeness</td>
<td>Current records are complete but there are no entry controls to require fields to be entered other than primary keys.</td>
<td>-</td>
</tr>
</tbody>
</table>
Raw_Materials and Uses tables are not filled with much data.

<table>
<thead>
<tr>
<th>Timeliness</th>
<th>Ok</th>
<th>+</th>
</tr>
</thead>
<tbody>
<tr>
<td>Currency</td>
<td>Raw_Materials and Uses tables do not appear to be filled with current data</td>
<td>-</td>
</tr>
<tr>
<td>Conformance</td>
<td>Ok</td>
<td>+</td>
</tr>
<tr>
<td>Referential Integrity</td>
<td>Referential integrity appears to be in force</td>
<td>+</td>
</tr>
</tbody>
</table>

The data quality of these four related tables appears to be below average. In particular, the missing data in the Raw_Materials and the Uses tables is troubling and hard to interpret. PVFC management should pay attention to instituting some data entry controls to ensure complete entry of information beyond the primary key field for each record in all tables, in order to maintain consistency and completeness.

2. Inspection of data values of related tables:

The current data values for these related tables appear to be within acceptable ranges of values, with the exception of issues raised earlier in the review of the Product table.


4. Assessment of data quality in related tables: The data quality of these four tables is below average. There are some areas of concern regarding completeness, accuracy, consistency, and currency that need management attention. Following is a brief list of action items for PVFC management:

Investigate why the Raw_Materials and Uses tables are under-populated. Was data entered and deleted? Was data never entered originally? Are we not capturing this data in the application systems?

Institute required entry of fields in all tables beyond the primary key field. The assignment of price should use a default value if a standard price is not known at the time of entering a new product item. Consider automating the description of products along agreed upon dimensions. For example, if you know you only produce desks, tables, and entertainment centers you may wish to break up the product description field into smaller subsets, like type=desk, kind=computer), that can be selected from a drop-down list (or other automated selection mechanism) so that errors in data entry may be eliminated or at least mitigated.

d. Based on the potential errors you discover in the data from the previous two exercises (assuming you find some potential errors), recommend some ways the capture of the
erroneous data could be improved to prevent errors in future data entry for this type of data.

Please see final assessment and action item listing of each of the prior exercises for details on addressing future data entry errors.

e. Evaluate the ERD for the database. You may have to reverse engineer the ERD if one is not available with the database. Is this a high-quality data model? If not, how should it be changed to make it a high-quality data model?

Yes, the ERD for the database appears to be a high-quality data model. It might be improved with a bit more attention to the design of fields, for example, the product description example in prior exercises, and the use of single field surrogate primary keys for the associative entity tables, but it essentially follows all the guidelines for a high-quality data model according to Hay (2005).

f. Assuming your are working with a Pine Valley Furniture Company database in this exercise, consider both the large and small PVFC databases as two different source systems within PVFC. What type of approach would you recommend (consolidation, federation, propagation, master data management), and why, for data integration across these two databases? Presume that you do not know a specific list of queries or reports that need the integrated database and design your data integration approach to support any requirements against any data from these databases.

A consolidation approach for the data integration of the large and small PVFC databases is recommended. Based on a current review of the structure of both databases, there are enough structural differences that the federation, propagation or master data management approaches would work. The consolidation approach could foster organizational agreement on standard understandings of the data in the organization, and may result in a unified view of data for decision making. The consolidation approach may enable PVFC to make revisions in the small database version such that in the future a federation, propagation, or master data management approach may work.
Answers for Field Exercises

1 through 4: Student answers will vary based upon the outside resources consulted. If you assign the interview options (questions 3 and 4), you may wish to provide students with the guides to interviewing that were provided in earlier chapters in the Instructors’ Manual.

Case Questions

1. Do you think that data quality at MVCH is a strategic issue? Why or why not?

Data quality is a strategic issue because poor data quality can result in poor patient care, which will directly impact the growth of the hospital. Poor data quality means increased costs as well.

2. In light of HIPAA and other regulations, securing and protecting patient records is a primary requirement for MVCH. Examine the organization chart in Chapter 2 (MVCH Figure 2-1). Who would be the best choice for a data steward for patient data? Please explain your answer. What recommendations would you make for establishing a data governance committee for MVCH? Who should be on that committee?

Based on reviewing the MVCH organization chart in Figure 2-1 and the expectations of a data steward in this chapter, Mr. Clay should be placed in the role of data steward for patient data. A data steward needs to be a subject-matter expert, have a strong interest in managing information as a corporate resource, an in-depth understanding of the business of the organization, and good negotiation skills. As Mr. Clay is the current person charged with responsibility and accountability for Admissions and Patient Accounts, it seems like he would fit the requirements of this role.

Membership of the data governance committee should include members at the executive level of the organization. At minimum, the data governance committee at MVCH should include Dr. Browne (Chief of Staff), Mr. Lopez (CFO), Ms. Knight (Chief Nursing Officer), and Mr. Heller (CIO). If more executives can accommodate this additional responsibility, then membership should be extended to the entire executive team, including Ms. Price (Clinical Services) and Ms. Baddekar (Chief Administrative Officer).

3. Refer to the MVCH case in Chapter 11 and your answers to case questions and exercises there. How can a data warehouse help improve data quality at MVCH? Can it? Under what circumstances would a data warehouse improve data quality?

A data warehouse at MVCH would improve data quality substantially because much of the data used for reporting and decision-support systems would be centralized across the enterprise. This would cause all parts of the organization to work from the same data and metadata, achieving consistency. Before the data warehouse is loaded, all data would have to be cleansed and scrubbed (reconciled), which would help to insure quality data. A data warehouse would also force the organization to address metadata management, which is core to achieving quality data.
4. Refer to the MVCH case in Chapter 11 and your answers to case questions and exercises there. Which data quality challenges may arise if MVCH develops a data warehouse and/or data mart(s)? Do you think that there is a need for data scrubbing? If so, on all tables or just some?

There is definitely a need for data scrubbing of all tables. To do so may encounter some challenges. There are strict privacy regulations in the healthcare field, which can make it difficult to share data without patient permission. Sharing data across systems, from doctors to hospital to urgent care units, will be essential for scrubbing. There will also likely be considerable missing data. Filling in missing values will have to be done carefully (estimates may not be suitable) and proactively. Medical conditions of a patient can also change rapidly, so data quality controls need to be automated as much as possible.

5. There are commercial off-the-shelf (COTS) packages for EMR, which would replace all of the data systems that would have to be integrated to form an EMR system in-house. (You might want to research a few as background to this question.) Develop a list of pros and cons for purchasing a COTS EMR system versus developing a program for data integration to provide EMR capabilities on top of the existing disparate data sources within MVCH.

This exercise is left to the students. This might also be an excellent in-class presentation for students to make. Small groups of students could investigate one COTS EMR each to gain some perspective on the pros/cons of purchasing versus developing an integrated system.

To some extent, there will be similarities in the generic pros/cons of purchase or development decisions (the IT “make or buy” decision). Some of the more common pros/cons are presented below:

Pros:
- Gain vendor expertise in installation and support
- Standardized field definitions integrated across the system
- Vendor must maintain certification with HIPAA, Medicare, other government or regulatory requirements
- Vendor support and training
- Do not lose time developing new software

Cons:
- Will need to convert internal systems to new system—may require re-working of business processes to fit the commercial system processing
- Conversion of current data to new system format
- New knowledge for internal IT staff
- Financial viability of vendor can be a concern; needs to be researched thoroughly
- May have limited customizability for system stakeholders (e.g., doctors, nurses, clinicians, etc.)

**Case Exercises**

1. This exercise is left to the students. This might also be an excellent in-class presentation for students to make. Small groups of students could investigate one local hospital each to gain some perspective on the questions for this exercise.

2. This is an excellent question for an in-class debate between teams of students. Each team should be assigned one of the approaches and should develop a defense for that approach. Another portion of the class could serve in the role of business sponsor and determine, based upon the arguments presented, which solution would be the best way to go.
Chapter 13 Data and Database Administration

Chapter Overview

The purpose of this chapter is to introduce students to the principles and tools for the administration of data. Data and database administration assume even greater importance as organizations distribute their data and processing across the enterprise.

The chapter emphasizes the changing roles and approaches of data and database administration. The chapter continues to emphasize the critical importance of data and database management in managing data as a corporate asset. It includes the major issues of data security, concurrency control, and backup and recovery that occupy database administrators’ attention on an ongoing basis.

Chapter Objectives

Specific learning objectives are included at the beginning of the chapter. From an instructor’s point of view, the objectives of this chapter are to:

1. Ensure that the students understand the importance of data administration and its relationship to information resource management.
2. Develop an understanding of the changes that are occurring in the data administrator and database administrator roles.
3. Understand the problem of lost updates and the use of both pessimistic and optimistic concurrency control mechanisms.
4. Define the major functions and components of the principal data management software: database management systems and information repositories.
5. Emphasize the problems of database security, and gain an understanding of techniques used to enhance database security.
6. Provide a sound understanding of database recovery including possible situations requiring recovery and the available procedures and facilities to use when recovery is necessary.
7. Understand areas such as installation decisions, management of I/O contention, and so forth that can be addressed when tuning a database.
Key Terms

<table>
<thead>
<tr>
<th>Aborted transaction</th>
<th>Database security</th>
<th>Restore/rerun</th>
</tr>
</thead>
<tbody>
<tr>
<td>After-image</td>
<td>Deadlock</td>
<td>Shared lock (S lock, read lock)</td>
</tr>
<tr>
<td>Authorization rules</td>
<td>Deadlock prevention</td>
<td></td>
</tr>
<tr>
<td>Backup facilities</td>
<td>Deadlock resolution</td>
<td>Smart card</td>
</tr>
<tr>
<td>Backward recovery</td>
<td>Encryption</td>
<td>System catalog</td>
</tr>
<tr>
<td>(rollback)</td>
<td>Exclusive lock (X lock, write lock)</td>
<td>Transaction</td>
</tr>
<tr>
<td>Before-image</td>
<td>Forward recovery</td>
<td>Transaction boundaries</td>
</tr>
<tr>
<td>Checkpoint facility</td>
<td>(rollforward)</td>
<td>Transaction log</td>
</tr>
<tr>
<td>Concurrency control</td>
<td>Heartbeat query</td>
<td>Two-phase locking protocol</td>
</tr>
<tr>
<td>Data administration</td>
<td>Inconsistent read problem</td>
<td>User-defined procedures</td>
</tr>
<tr>
<td>Data archiving</td>
<td>Information Repository</td>
<td></td>
</tr>
<tr>
<td>Data dictionary</td>
<td>Dictionary System (IRDS)</td>
<td>Versioning</td>
</tr>
<tr>
<td>Database administration</td>
<td>Locking</td>
<td></td>
</tr>
<tr>
<td>Database change log</td>
<td>Locking level (lock granularity)</td>
<td></td>
</tr>
<tr>
<td>Database destruction</td>
<td>Open-source DBMS</td>
<td></td>
</tr>
<tr>
<td>Database recovery</td>
<td>Recovery manager</td>
<td></td>
</tr>
</tbody>
</table>

Classroom Ideas

1. If possible, schedule a classroom visit by a data administrator or database administrator from a local company or organization. Discuss the agenda in advance, and allow plenty of time for questions.

2. Discuss the functions of data administration and database administration (Figure 13-1). Indicate that this is an interesting career field that requires both behavioral and technical skills. If possible, bring a few job announcements in the database field to class. Show salary data. Discuss the current situation for outsourcing of database administration and the difficulty of outsourcing data administration.

3. Discuss the selection of a data administrator and the placement of the data administration function. Discuss changing expectations of data administrators and alternative organization placements of data administration responsibilities.

4. Discuss the role of an information repository and its relationship to CASE tools. Compare alternate uses of a repository: passive, active in development, and active in production.

5. Have the students seek examples of computer security problems or “white collar” crime in the press and bring them to class. Alternatively, bring an example or two yourself. Then discuss various security threats, security procedures, and the procedures’ relative effectiveness (Figure 13-3).

6. Discuss database recovery and the various recovery techniques. Stress that failures are inevitable, and an organization must be able to recover if it is to survive. If the students
have related work experience, they can often provide concrete examples of failures that will make the problem more real to other students.

7. Discuss the problem of lost updates and concurrency control (Figure 13-11). One way to dramatize this process is to conduct a manual demonstration: one student is “keeper of the data,” two others are “user X” and “user Y.”

8. Now discuss locks and the problems that can result from their use (Figures 13-12 through 13-14). Locks are an imperfect (but effective) mechanism for multi-user databases.

9. Discuss versioning and optimistic concurrency control (Figure 13-15). Compare pessimistic and optimistic concurrency control and the conditions under which each is likely to be effective or preferred.

10. Inexperienced students are likely to underestimate the problems associated with establishing adequate data quality. A discussion of the various areas that must be considered, along with examples, should be conducted.

11. Try to give the students a basic understanding of the different aspects that may be manipulated when tuning a database. They should understand that it is not possible to tune a database and then assume that it will stay that way.

12. Review the SQL commands for database administration (various CREATE with all the subclauses for journalizing and other features, GRANT/REVOKE, and physical data storage control). Show how these relate to the topics of this chapter.

13. Access the ICCP or DAMA International Web sites and discuss with them the DBA certification exam these organizations sponsor. Talk with your local DAMA chapter to solicit a guest speaker to discuss the value of this vendor-neutral certification. This exam is actually based on this chapter. You could even arrange to be a proctor for administering the exam to your students.

Answers to Review Questions

1. Define the following key terms:

   a. **Data administration** A high-level function that is responsible for the overall management of data resources in an organization, including maintaining corporate-wide definitions and standards.

   b. **Database administration** A technical function that is responsible for physical database design and for dealing with technical issues such as security enforcement, database performance, and backup and recovery.

   c. **Two-phase locking protocol** A procedure for acquiring the necessary locks for a transaction where all necessary locks are acquired before any locks are released, resulting in a growing phase, when locks are acquired, and a shrinking phase, when they are released.

   d. **Information repository** Stores metadata that describe an organization’s data and data processing resources. Manages the total information-processing environment. Combines information about an organization’s business information and its application portfolio.

   e. **Locking** Concurrency control mechanisms that deny access to other users in a multi-user environment while an update is completed or aborted.
f. **Versioning**  Concurrency control mechanism that doesn’t use record locking. Each transaction is restricted to a view of the database as of the time that transaction started, and when a transaction modifies a record, the DBMS creates a new record version instead of overwriting the old record.

g. **Deadlock**  An impasse that results when two or more transactions have locked a common resource, and each waits for the other to unlock that resource.

h. **Transaction**  A discrete unit of work that must be completely processed or not processed at all within a computer system.

i. **Encryption**  The coding (or scrambling) of data so that humans cannot read them.

j. **Data availability**  The concept of data being available to users when needed. Ensuring data availability is of utmost importance to the DBA.

k. **Data archiving**  The process of moving inactive data to another location where it can be accessed when needed.

l. **Heartbeat query**  A query submitted by the DBA to test the current performance of the database. A heartbeat query is also called a canary query.

2. d backup facilities  
f biometric device  
j checkpoint facility  
g database recovery  
a database security  
i granularity  
h recovery manager  
b rollback  
e rollforward  
c system catalog  

3. Compare and contrast the following terms:

   a. **Data administration; database administration**  Data administration is the overall management of data resources. The second function, that of database administration, has been regarded as responsible for physical database design and for dealing with the technical issues, such as security enforcement, database performance, backup, and recovery, associated with managing a database.

   b. **Repository; data dictionary**  While data dictionaries are simple, data-element documentation tools, information repositories are used by data administrators and other information specialists to manage the total information-processing environment.

   c. **Deadlock prevention; deadlock resolution**  When deadlock prevention is employed, user programs must lock all records they will require at the beginning of a transaction, rather than one at a time. Deadlock resolution allows deadlocks to occur, but builds mechanisms into the DBMS for detecting and breaking the deadlocks.

   d. **Backward recovery; forward recovery**  With backward recovery (also called rollback), the DBMS backs out of or undoes unwanted changes to the database. Before-images of the records that have been changed are applied to the database. As a result, the database is returned to an earlier state; the unwanted changes are eliminated. With forward recovery (also called rollforward), the DBMS starts with an earlier copy of the database.
By applying after-images (the results of good transactions), the database is quickly moved forward to a later state (Figure 12-7).

e. *Active data dictionary; passive data dictionary* An active data dictionary is managed automatically by the database management software. Active systems are always consistent with the current structure and definition of the database because they are system maintained. A passive data dictionary is managed by the user(s) of the system, and it is modified whenever the structure of the database is changed. Because this modification must be performed manually by the user, it is possible that the data dictionary will not be current with the current structure of the database.

f. *Optimistic concurrency control; pessimistic concurrency control* Pessimistic approaches use record locking procedures, while optimistic approaches use versioning to achieve concurrency control.

g. *Shared lock; exclusive lock* Placing a shared lock on a record prevents another user from placing an exclusive lock on that record. Placing an exclusive lock on a record prevents another user from placing any type of lock on that record.

h. *Before-image; after-image* A before-image is simply a copy of a record before it has been modified, and an after-image is a copy of the same record after it has been modified.

i. *Two-phase locking protocol; versioning* The two-phase locking protocol relates to the cautious approach of locking the record so that other programs cannot use it. In reality, in most cases other users will not request the same records, or they may only want to read them, which is not a problem (Celko, 1992). By using versioning, a DBMS assumes that most of the time other users do not want the same record, or if they do, they only want to read (but not update) the record. With versioning, there is no form of locking.

j. *Authorization; authentication* Authentication schemes positively identify a person attempting to gain access to a database. For example, a person has to first supply a particular password (or another required proof of identity, according to the authentication scheme in use), and after successfully completing the authentication procedure (if any) may be authorized to read any record in a database.

k. *Data backup; data archiving* Data backup is a process that creates a backup copy of data which can be used to recover lost data due to hardware and/or software failures. Data archiving is a process, which moves inactive data to another place for storage.

4. **Open-source DBMS:**

An open-source DBMS is a free or nearly free database software whose source code is publicly available and runs on the most popular operating systems. The free DBMS source code software provided the core functionality of an SQL-compliant DBMS. Leading examples include MySQL (www.mysql.com) and PostgreSQL (www.postgresql.org).

5.

a. Managing the data repository – data administrator
b. Installing and upgrading the DBMS – database administrator
c. Conceptual data modeling – data administrator
d. Managing data security and privacy – database administrator
6. The changing roles of the data administrator and database administrator:

Data administration is a high-level function that is responsible for the overall management of data resources in an organization, including maintaining corporate-wide data definitions and standards. Typically, the role of database administration is taken to be a more hands-on, physical involvement with the management of a database or databases. Database administration is a technical function that is responsible for physical database design and for dealing with technical issues such as security enforcement, database performance, and backup and recovery. As business practices change, the roles are also changing within organizations. There are, however, a core set of database administration functions, which must be met in every organization, regardless of the database administration chosen. On one hand, these functions may be spread across data administrators and database administrators. At the other extreme, all of these functions may be handled by a single DBA.

7. Four common problems of ineffective data administration:

a. Multiple definitions of the same data entity and/or inconsistent representations of the same data elements
b. Missing key data elements, whose loss eliminates the value of existing data
c. Low data quality levels due to inappropriate sources of data or timing of data transfers from one system to another, thus reducing the reliability of the data
d. Inadequate familiarity with existing data, including awareness of data location and meaning of stored data, thus reducing the capability to use the data to make effective strategic or planning decisions

8. Four job skills necessary for system data administration or data administrators:

a. Communication skills Must interact with top management, users, and computer applications specialists.
b. Must be a decision maker Must play a significant role in deciding where data will be stored and managed.
c. Flexibility To be capable of resolving differences that normally arise when a significant change is introduced into an organization.
d. Managerial and technical skills Capable of managing a technical staff and dealing with technical issues.

Four job skills necessary for project data administration or database administrators:

a. A sound understanding of current hardware architectures and capabilities.
b. A solid understanding of data processing and database development life cycle, including traditional and prototyping approaches.
c. Strong design and data modeling skills are essential at the conceptual, logical, and physical levels.
d. Managerial skills are also critical.

9. Briefly describe four new DBA roles that are emerging today:
   a. *Procedural DBAs* The procedural DBA is responsible for ensuring that all stored procedures, triggers, and Persistent Stored Modules are effectively planned, tested, implemented, shared, and reused.
   b. *e-DBA* An e-DBA is a person who has a full range of DBA skills and in addition can manage applications and databases that are Internet-enabled.
   c. *PDA DBA* Because PDA usage is growing, more users are putting smaller-footprint databases on their PDAs. As such, a new type of DBA is needed to support these users.
   d. *Data warehouse administrator* The DWA is responsible for building an environment to support a data warehouse or data mart. He or she has a similar role as the DBA of a transaction processing system; however, he or she is concerned with the coordination of metadata and data across many data sources.

10. Changes in data administration procedures that will decrease development and implementation time:
   a. *Database planning* Improve technology selection through selective evaluation of possible products. Consider each technology’s fit with the enterprise data model, reducing time required in later stages by effective selection of technology at the database planning stage.
   b. *Database analysis* Work on physical design in parallel with development of the logical and physical models. Prototyping the application now may well lead to changes in the logical and physical data models earlier in the development process.
   c. *Database design* Prioritize application transactions by volume, importance, and complexity. These transactions are going to be most critical to the application, and specifications for them should be reviewed as quickly as the transactions are developed. Logical data modeling, physical database modeling, and prototyping may occur in parallel. DBAs should strive to provide adequate control of the database environment while allowing the developers space and opportunity to experiment.
   d. *Database implementation* Institute database change control procedures so that development and implementation are supported rather than slowed. Wherever possible, segment the model into modules that can be analyzed and implemented more quickly. Find ways to test the system more quickly without compromising quality. Testing may be moved earlier in the development; use testing and change control tools to build and manage the test and production environments.
   e. *Operation and maintenance* Review all timesaving measures that have been taken to ensure that database quality has not been compromised. Consider using third-party tools and utilities wherever possible to save work; other tools, such as Lotus Notes, may reduce the need for meetings, thus saving time.
11. Five areas where threats to data security may occur:

   a. *Accidental losses, including human error, software, and hardware-caused breaches*
   Establishing operating procedures, such as user authorization, uniform software installation procedures, and hardware maintenance schedules, are examples of actions that may be taken to address threats from accidental losses. As in any effort that involves human beings, some losses are inevitable, but well thought-out policies and procedures should reduce the amount and severity of losses.

   b. *Theft and fraud* These activities are going to be perpetrated by people, quite possibly through electronic means, and may or may not alter data. Attention here should focus on each possible location shown in Figure 12-3. For example, control of physical security, so that unauthorized personnel are not able to gain access to the machine room, should be established. Data access policies that restrict altering data immediately prior to a payroll run will help to secure the data. Establishment of a firewall to protect unauthorized access to inappropriate parts of the database through outside communication links is another example of a security procedure that will hamper people who are intent on theft or fraud.

   c. *Loss of privacy or confidentiality* Loss of privacy is usually taken to mean loss of protection of data about individuals, while loss of confidentiality is usually taken to mean loss of protection of critical organizational data that may have strategic value to the organization. Failure to control privacy of information may lead to blackmail, bribery, public embarrassment, or use of user passwords. Failure to control confidentiality may lead to loss of competitiveness.

   d. *Loss of data integrity* When data integrity is compromised, data will be invalid or corrupted. Unless data integrity can be restored through established backup and recovery procedures, an organization may suffer serious losses or make incorrect and expensive decisions based on the invalid data.

   e. *Loss of availability* Sabotage of hardware, networks, or applications may cause the data to become unavailable to users, which again may lead to severe operational difficulties.

12. How creating a view may increase data security, and why not to rely completely on using views to enforce data security:

   The advantage of a view is that it can be built to present only the data to which the user requires access, thus effectively preventing the user from viewing other data (not produced by the SQL for the view) that may be private or confidential. The user may be granted the right to access the view, but not to access the base tables upon which the view is based. For most views, data may not be updated, thus eliminating the possibility of unauthorized changes to the base data. However, views are not adequate security measures because unauthorized persons may gain knowledge of or access to a particular view. In addition, with high-level query languages, an unauthorized person may gain access to data through simple experimentation. Views also do not protect data from access outside the DBMS by hackers. Thus, views provide more convenience than security. *See also the answer to Review Question 27.*
13. Integrity controls in database security:

Integrity controls protect data from unauthorized use and update. Integrity controls limit the values a field may hold, limit the actions that can be performed, or trigger the execution of some procedure (such as placing an entry into a log to record which users have performed which actions on which data). A domain is an example of an integrity constraint that can be used to define a user-defined data type. If this type ever needs to be changed, it can be changed in only one place, and all fields using this domain will be updated automatically. Assertion constraints enforce desirable database conditions and check automatically whenever transactions are run. Triggers can be used for events or conditions, and actions needed to be tracked against a database. Triggers cannot be circumvented.

14. The difference between an authentication scheme and an authorization scheme:

Authorization rules are primarily concerned with protection of the data themselves. They are controls incorporated in the data management system that restrict access to data and also restrict the actions that people may take when they access data. For example, a person who can supply a particular password may be authorized to read any record in a database but cannot necessarily modify any of those records. Authentication schemes establish operating procedures that positively identify a person attempting to gain access to a computer or its resources. Hence their primary concentration is the user’s identification. A user may gain access to the database through an authentication procedure, but the actions he or she may undertake later will depend further on the authorization rules.

15. The advantage of optimistic concurrency control compared with pessimistic concurrency control:

The main advantage of versioning over locking is performance improvement. Read-only transactions can run concurrently with updating transactions without loss of database consistency.

16. The difference between shared locks and exclusive locks:

Shared locks enable others to read a record that is locked by a given user, but not to update a record until it is unlocked. Exclusive locks cause a record to be unavailable for reading until it is unlocked.

17. The difference between deadlock prevention and deadlock resolution:

Deadlock prevention requires all users to lock all records they will require at the beginning of a transaction as opposed to one at a time. Unfortunately, it is often difficult to predict in advance all records that will be needed at the beginning of a transaction. Deadlock resolution requires DBMS mechanisms that detect and break deadlocks. A matrix of a resource usage is maintained that enables deadlock detection.
18. Four DBMS facilities that are required for database backup and recovery:
   
   a. **Backup facilities** provide periodic backup copies of the entire database.
   b. **Journalizing facilities** maintain an audit trail of transactions and database changes.
   c. **Checkpoint facility** allows periodic suspension of all processing and synchronization of a database’s files and journals.
   d. **Recovery manager** allows the DBMS to restore the database to a correct condition and restart processing transactions.

19. Transaction integrity:

   A transaction is the identification of the sequence of steps that constitute a well-defined business activity. Transaction integrity includes those actions taken to enforce the commitment of all of the steps that constitute a transaction. In processing a transaction, we want any changes to the database to be made only if the entire transaction is processed successfully. If the transaction aborts at any point, no changes can be allowed, or else the database will not reflect the data needed to manage the firm.

20. Four common types of database failure:

   a. **Aborted transaction** Aborting a transaction in progress due to some abnormal condition, such as the loss of transmission in a communication link while a transaction is in progress.
   b. **Incorrect data** The database is updated with incorrect, but valid data: for example, an incorrect grade recorded for a student.
   c. **System failure** Some component of the system fails, but the database is not damaged. Causes include power loss, operator failure, loss of communications transmission, or system software failure.
   d. **Database loss or destruction** The database itself is lost or destroyed or cannot be read.

21. Four threats to high availability:

   a. **Hardware failure** Any component of a system, such as memory or a disk drive, can become a point of failure. The usual solution is to provide some measure of redundancy. One example might be having the workload spread across several servers. If one server fails, the others can take over its workload.
   b. **Loss or corruption of data** Data can be lost or become corrupt for many reasons, including hardware and network issues. The best solution to prevent this from interrupting service is to have a backup database with an exact copy of the data. Also, appropriate backup and recovery procedures must be in place.
   c. **Maintenance downtime** Before systems were needed 24/7, there could be planned downtime for maintenance tasks. Generally, this can no longer be done for many systems. Nondisruptive utilities are available which allow routine maintenance to be done on a database without having to take the system off-line.
   d. **Network-related problems** Both internal networks as well as the Internet can cause problems for availability. Organizations should employ the latest in firewalls and other
network technologies as well as have a response plan in place for any large spikes in activity.


IRDS is a computer software tool that is used to manage and control access to the information repository. It provides facilities for recording, storing, and processing descriptions of an organization’s significant data and data processing resources (Lefkovitz, 1985). When systems are IRDS compliant, it is possible to transfer data definitions among the data dictionaries generated by the various products. IRDS has been adopted as a standard by the International Standards Organization (1990), and it includes a set of rules for storing data dictionary information and for accessing it.

23. The ACID properties of a transaction:

Atomic, meaning it cannot be further subdivided. This means a transaction must be completed in its entirety or not at all. If the transaction fails at any point, the whole transaction must be rolled back. Once the whole transaction executes, all changes are committed to the database.

Consistent, meaning that any constraints true before the transaction was applied are true after the transaction is committed.

Isolated, meaning that changes to the database are not revealed to users until the whole transaction is committed. No intermediate values are allowed to be shown.

Durable, meaning that changes are permanent. No subsequent failure of the database can reverse the effect of the transaction.

24. Two common forms of encryption:

One key or data encryption standard (DES). Both the sender and the receiver know the key that is used to scramble the message being transmitted.

Two key or dual key, public key, or asymmetric encryption. This uses both a private key and a public key. All users will know each other’s public keys, but only each individual knows their private key.

25. Four components of a disaster recovery plan:

a. Develop a detailed disaster recovery plan.
b. Choose and train a multidisciplinary team to carry out the plan.
c. Establish a backup data center at an off-site location.
d. Send backup copies of the databases to the backup data center on a scheduled basis.

26. Heartbeat queries are simple queries, which are run several times throughout the day to access the performance of the RDBMS by the database administrator. If a heartbeat query takes a long time to run, then there may be a resource problem such as inefficient queries or an inappropriate mix of jobs.
27. A view is a dynamic result table that is created at run time from a query run over base tables in the database. A view can be constructed to present only the data a particular user needs to access (e.g., certain rows and/or columns) and the user can be granted access only to that view, rather than the base tables that feed the view. A view can be used as part of a data security approach in an organization, but will not be sufficient to control all access to the data stored in databases. Often, views are shared among persons in the organization but people may have different levels of authorization over this same view—a few may be able to update the data while others are restricted to read-only access. It may be easy for unauthorized users to gain knowledge of, or access to, a particular view or person who has greater access authority than the unauthorized person. Generally speaking, additional data security measures must be used along with views to provide an appropriate level of protection over organizational data. See also the answer to Review Question 12.

28. The purpose of the GRANT and REVOKE SQL commands is to control the access to data, and the actions that can be taken on the data, within a database management system. The GRANT and REVOKE commands are generally operated against User accounts within a database management system. The DBMS vendors may vary in their ability to control the authorization rules within a particular DBMS. In Oracle, authorization rules are handled with privileges on User accounts at the database or the table level, as well as INSERT and UPDATE privileges on a column level. Oracle privileges that can be GRANTed or REVOKEd from a user include: SELECT, INSERT, UPDATE, DELETE, ALTER, INDEX, REFERENCES, EXECUTE.
Answers to Problems and Exercises

1.

<table>
<thead>
<tr>
<th></th>
<th>Inventory Records</th>
<th>Receivables Records</th>
<th>Payroll Records</th>
<th>Customer Records</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Insert</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Modify</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Delete</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
</tbody>
</table>

Authorizations for Inventory Clerks

<table>
<thead>
<tr>
<th></th>
<th>Salespersons</th>
<th>A/R Personnel</th>
<th>Inventory Clerks</th>
<th>Carpenters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Insert</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Modify</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>Delete</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
</tr>
</tbody>
</table>

Authorizations for Inventory Records

2.
   a. Aborted transaction; use backward recovery. Of course, the transaction then needs to be re-entered.
   b. Database destruction; use forward recovery from the backup copy of the database.
   c. System failure; use forward recovery from most recent checkpoint.
   d. Incorrect data; several alternatives are possible. First, you could do a selective backward recovery to undo the incorrect transaction, and then reprocess the transaction using the correct amount. If this is not possible, then enter a compensating transaction and notify the student of the error.
   e. If the system is mirrored, it may be possible to switch to the mirrored data and rebuild the corrupted data on a new disk. If the system is not mirrored, it may not be possible to restart because status information in main memory has been lost or damaged. The safest approach is to restart from the most recent checkpoint before the system failure. Because no database change log is created, the database cannot be rolled forward by applying after-images. Transactions have to be reprocessed manually.

3.
   a. Because the merchandise return credit of $50 processed last, the actual balance after the last transaction was $200.
   b. Payment of $250 reduces the balance to $0. A purchase on credit increases the balance to $100. A merchandise return reduces the balance to $50.
4. a. Encryption
   b. Authentication schemes
   c. Authorization rules
   d. Authentication schemes

5. This is a discussion question with no one correct answer. One solution is given below, but the ranking of the first two candidates is the most difficult and could lead to a rousing class discussion.

1. **Jim Reedy**’s professional background shows a combination of both managerial and technical skills, a key requirement for the position of the data administrator. As a well-respected insider, he is likely to be capable of enlisting cooperation from users. Jim Reedy is very familiar with Metro’s current systems environment, and this fact will be a plus in managing the technical staff and dealing with technical issues.

2. **Gerald Bruester** ranks second. Data administration is a high-level function that is responsible for the overall management of data resource. Gerald Bruester is a senior database administrator, but he also has knowledge related to data warehousing—a fact that gives him a competitive advantage over Monica Lopez. Also, he has managed a marketing-oriented database, which will be a most preferred experience for a warehouse storing customer information for marketing purposes, rather than financial information (as in the case with Monica Lopez).

3. **Marie Weber** ranks third because her expertise includes management of a similar data warehouse implementation, which stores customer information. Overall management of data resources has already been one of her responsibilities.

4. **Monica Lopez**’s management of a financial database for a global banking firm for five years in a constantly changing contemporary environment is great proof of professionalism. Despite her expertise as a senior database administrator, however, Monica Lopez has been involved in a different type of business—a major disadvantage over the other candidates.

6. This is a discussion question with no one correct answer. One solution is given below:

1. **Marie Weber** would be the preferred candidate because her expertise includes management of a similar data warehouse implementation, which stores customer information. Her experience with a Red Brick-based application would be a strong asset in the process of building the new data warehouse.

2. **Gerald Bruester** would be the second preferred candidate. He has managed a marketing-oriented database, which will be a most preferred experience for a warehouse storing customer information for marketing purposes, rather than financial information (as in the case with Monica Lopez).

3. **Jim Reedy** is very familiar with Metro’s current systems environment, and this fact will be a plus in managing the technical staff and dealing with technical issues.
4. **Monica Lopez** (as in Problem 5), despite her expertise as a senior database administrator, Monica Lopez has been involved in a different type of business—a major disadvantage over the other candidates.

7. This is a discussion question with no one correct answer. One answer is given below.

1. **Gerald Bruester** would be the preferred candidate. Gerald Bruester is a senior database administrator. His managerial and technical experience with a marketing-oriented database will be very suitable for a company involved in storing customer information for marketing purposes.

2. **Jim Reedy** has technical expertise and in addition is very familiar with Metro’s current systems environment. This fact will be a substantial plus for him.

3. **Monica Lopez** has managed a financial database for five years, probably in a dynamic environment. This expertise gives her a competitive advantage over Marie Weber.

4. **Marie Weber** would be the least preferred candidate, as she has no experience as a database administrator.

8. There are no established security procedures and policies, or the implementation and enforcement of those procedures and policies is lacking. As a result there are two major concerns:

   a. *Loss of privacy or confidentiality* Failure to control privacy of information may lead to blackmail, bribery, public embarrassment, or use of user passwords. Failure to control confidentiality may lead to loss of competitiveness.

   b. *Theft and fraud* These activities are going to be perpetrated by people able to gain access to the machines to alter or obtain data without authorization.

9. To reduce I/O contention, data files need to be separated that are being accessed together. The large database objects that will be accessed concurrently may be striped across disks. This would also reduce I/O contention. Alternatively, it may be possible to reschedule batch jobs that use files on the one drive under contention so that the jobs do not run concurrently. Finally, it might be possible to reorganize the database to allow for more efficient processing by using better indexing schemes.

10. Optimal solutions to this situation will depend on a more detailed understanding of the exact situation and the RDBMS being used, but here are some possibilities based on typical Oracle environments:

   a. Replicate the data on the corporate machine, taking advantage of Oracle’s data caching capabilities. The data will then become a part of the local corporate instance, allowing additional processing of the data beyond producing the regional monthly sales report if desired, without incurring additional loads on the network. If necessary, the snapshot of the data can be refreshed at predetermined intervals. Snapshots can be scheduled at off-peak load times.

   b. Create summary tables at each regional site that contain the summarized data necessary to prepare the monthly regional sales reports at corporate headquarters. This approach
would greatly diminish the network load, as only the summary tables would need to be transmitted. This option does not necessarily provide the flexibility for using the data for other processing that is possible from the previous approach.

c. Because the reports are regional, move the processing of the reports out to the regional offices. This approach would eliminate any network load problems occurring as a result of the existing procedures. However, this approach will raise software management issues unless the company can manage the software used to produce the reports centrally.

d. Moving to a three-tier client/server architecture might also be a possibility. Then the reporting software could reside on the middle tier and the software management issues inherent in the previous approach might be greatly reduced.

11. 

a. According to Table 13-2, the average cost per hour of downtime is $90,000. According to Table 13-3, a system that has 99.9 percent availability has approximately 8.77 hours downtime per year. So, the annual cost of downtime is $789,300.

b. According to Table 13-3, 99.99 percent downtime is equivalent to 0.88 hours a year. Therefore, the total annual cost of downtime would be $79,200. The net realized savings would be $710,100. The cost of the vendor’s proposal would be $300,000 per year. The company would realize a total net savings of $410,100. This would be justifiable because it is a significant cost savings.

12. For the retail brokerage firm, the expected annual cost of downtime for 99.9% availability is $56.6 million. For 99.5 percent availability it is $284 million. Neither of these levels would be acceptable.

13. The estimated loss would be $720,000.

14. The average increase in revenue will be $5 million (100 * 0.05 * 1,000,000).

15. While one primary concern would be ensuring that data were secure once the data warehouse was built, the extraction and transformation process also need to be secure so that access cannot be gained to the data by unauthorized users and subsequently altered before being loaded into the data warehouse. Perhaps the biggest security gap in this whole process is the use of FTP. One way to insure better data security would be to utilize a file transfer package with encryption or to encrypt the files before performing the file transfer.

16. If Fitchwood’s data mart were made available to customers via the Internet, security procedures would have to be in place to insure that customers could only access data on their policies. If Internet access were made available to agents as well, then another layer of security would be needed. One might question the merit of allowing any user to access the data mart via the Internet. It makes more sense for a user to access account information in an online transaction processing system.
17. Any time an external data source is used to bring data into a system, one cannot be sure how accurate the data is. One concern of Web services is security. Until this is addressed, it may not be wise to build a B2B system using Web services.

18. Left as an exercise for students. This question can be answered, in part, by review of the material in Chapter 12.

19. Left as an exercise for students.

20. Because an OLTP system has insertions, deletions, and updates being done constantly, there needs to be some mechanism for locking records and providing resolution of deadlocks. A data warehouse on the other hand, contains static data that generally does not change while the users are accessing the warehouse. All users of the warehouse are given read-only access.

**Answers to Field Exercises**

1. This assignment is an alternative to having a DBA or data administrator visit your class. The evaluation will depend on the organization visited but should target issues raised in this chapter.

2. No additional suggestions included.

3. This assignment is an alternative to having a DBA or data administrator visit your class. The evaluation will depend on the organization visited.

4. This assignment is an alternative to having a DBA or data administrator visit your class. The person who is most likely to be in charge of capacity planning is the database administrator. The DBA role has become more specialized lately, evolving into specialties such as distributed database/net work capacity planning, and it is possible your students will find a person whose responsibilities are defined in this manner.

5. This assignment is an alternative to having a DBA or data administrator visit your class.

6. This assignment will vary depending upon the topic chosen from the Web site.

7. This is a good exercise for students to gain some insight into open-source implementations. Students should particularly make note of how an open-source platform is administered as compared to a traditional platform such as Oracle.
Project Case Study

Case Questions

1. This question could result in a spirited classroom discussion about the value of bringing such a sophisticated information system to a relatively small hospital such as Mountain View. If such systems were to be implemented successfully, they hold the obvious possibility of improving the quality of care by integrating the patient’s records and allowing for rapid determination of a patient’s medical history and current situation. The CPOE system may streamline the entire process of order entry as well as provide a means of mining order data for decision support applications. These systems might also provide the types of security controls required by HIPAA, and controls in a consolidated system could go far to eliminate data quality issues. Any system they choose would have to be scalable to support their growth. Try to get the students to consider the cost containment issues relative to the size of the hospital and the number of records that will be included.

2. (a) Data security issues Consider focusing discussion on the issues of patient record confidentiality. Help the students to discuss the types of information that each of the various users (physicians, insurance clerks, laboratories, and so forth) should be able to access, read, and use. Discuss ways to establish these kinds of access and the importance of establishing policies and procedures to protect the patients’ confidentiality. Suggesting any of the following scenarios may foster excellent classroom discussion related to data security issues: where records of HIV patients are released; issues related to parents’ rights to their children’s medical records, especially if the “child” is away at college or part of the armed services deployment in a foreign country; disclosure of patient medical records to a potential employer; hackers stealing patient identity information to gain access to prescription medications.

(b) Data security techniques MVCH needs to be sure to have data policies, procedures, and standards in place regarding security of data records, especially those of patients to protect patient confidentiality and privacy rights. Additionally, specific steps should be taken to ensure adequate security for client/server, server, network, and Web-enabled databases (if any). In terms of specific database software data security features, MVCH should consider implementing views (to restrict access to particular tables and rows/columns within tables), authorization rules, encryption procedures for sensitive data, and authentication schemes for database access. Above all, MVCH needs to make sure that all parties who have access to patient records (in any form) are properly educated as to the sensitive nature of the information and how to properly secure it against unauthorized access by others who may try to use social engineering to procure the information (e.g., similar to the opening case on ChoicePoint at the beginning of the chapter).

3. Perhaps the best way to prevent access problems such as those described by Dr. Z would be to incrementally roll out the CPOE system and provide a lot of training and support. Also, standard operating procedures should be implemented along with database security in order to prevent things like the pharmacy unilaterally changing orders. Detailed authorization
matrices will be needed to identify who can do what with different pieces of data. Possibly approval mechanisms for changing data will be necessary. Thorough audit trails will be necessary to document all data entry, including who entered the data. Default values on electronic forms could be set for each user (e.g., to accommodate standards by physician).

4. It would not be recommended to develop a full-fledged CPOE system internally due to the complexity of such a system, testing requirements, etc. It would be much more prudent to purchase a system from a vendor who has fully tested the system and can support it. Perhaps the vendor’s system could be integrated with the SQL Server tables. Some customization of a package will be necessary, especially for reporting and decision support, and this is where the IT staff at MVCH can concentrate their efforts.

5. Physicians may resist an EMR system if it added more work for them or they feel a loss of control. Critical success factors would be adequate training as well as on-site support staff. Perhaps an implementation could be executed where the EMR incrementally replaced paper-based records. Other critical success factors would be to gain the full support of senior physicians who champion the project. Those physicians who are accepted as leaders at MVCH could be taken to other hospitals for demonstrations and discussions with physicians there who can speak to the merits of such a system. This is a good question to have students go back to an introductory MIS course where the topic of user acceptance of information systems was likely discussed.

6. Mountain View Community Hospital should adopt a CDP system because it would provide a record of every transaction in the system. If the CDP system utilizes disk for backup, recovery will be much faster than a tape backup and can easily be used to restore a clean copy of any data that may be corrupted. Given the mission-critical nature of the hospital’s data, this level of backup would be most ideal. Unlike a cold backup, the system could be running even as the data is being backed up. Of course the hospital could also explore other backup strategies, such as incremental warm backups performed periodically throughout the day, coupled with a cold backup at night. The disadvantage to this approach is that the hospital must maintain access to its systems 24 hours a day.

7. Data storage definitely should be treated as a strategic issue. If data security is compromised or data is lost, the hospital will face dire consequences because accessibility to data can directly impact patient care. In addition, the hospital needs to look at data storage as a long-range issue when considering growth of the organization. Strategic planning in this area would examine projected growth against existing data storage capacity.

8. The long-range business and information systems plan most likely will address growth as well as adaptation to new treatment modalities. The information systems portion of this plan will need to address data storage requirements and capacity requirements, backup and recovery, as well as security issues. An enterprise data model can be used to show the major categories of data for the assignment of data stewards. A corporate data policy and plan must be coordinated with long-range information systems plans.
Case Exercises

1. The case text specifically indicates only physicians, but it also implies that laboratory/clinical staff and all medical personnel would need access. Students may well suggest additional relevant groups such as nurses, care-giving staff, pharmacists (both internal and external to the hospital), and the patients themselves. Systems maintenance personnel and administrators will also need access. Management likely does not need direct access to the EMR for operational purposes, but there may be a need for summary reports for decision making about capacity planning, workloads, costs, and other planning and control purposes.

2. Be sure that students understand that any RDBMS used for such a sensitive system should include the capability to establish specific permissions at the table and record level. That is, few users should be given complete read, insert, delete, and modify privileges. Physicians, for example, should only be able to read and modify patients’ records for which they are responsible. Nurses should be able to read, insert, and modify records for patients assigned to their station only (possibly by limited access to workstations only in that station with proper user identification). External users would have more restricted access, usually having access to certain tables, records, and certain attributes within each record. All access needs to be tracked via an audit trail showing user identification, data accessed, location, and time of day, as well as before and/or after images for recovery.

3. RFID could be utilized with the EMR system in several ways. In 2004, the FDA began final approval of using RFID tags implanted under the skin to identify patients. RFID tags could also be utilized to track supplies, medications, as well as lab work. If RFID was used, the data storage requirements would increase because each individual supply, lab specimen, etc. would have a unique identifier. This could be a substantial increase in required storage capacity.

4. This is a good opportunity to discuss with students various storage technologies as well as backup strategies (such as CDP). A good case study can be found at www.eweek.com/article2/0,1895,1964610,00.asp. Emphasize to students in discussions that they need to consider such things as data backup and security.

5. This is a good student exercise also. It might be worthwhile to break the class up into teams to tackle portions of the security requirements.
Project Assignments

P1. This is left to the student as an exercise.

P2. The following is a sample matrix.

<table>
<thead>
<tr>
<th></th>
<th>Physicians</th>
<th>Secretary</th>
<th>Manager</th>
<th>Nurse</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Insert</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>Modify</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>Delete</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
</tbody>
</table>

P3. This is also left to the student as an exercise.
Chapter 14 Distributed Databases

Chapter Overview

Please note that the material for this chapter is based upon the Web version of Chapter 14, not the abbreviated version of Chapter 14 in the text. The purpose of this chapter is to discuss distributed database management. The trend toward distributed databases is being driven by the continued evolution of distributed database management systems software, the increasing importance of workgroup computing, and the globalization of commerce. This chapter, along with Chapter 13 on data and database administration, provides thorough coverage of database concurrent access controls.

Chapter Objectives

Specific student learning objectives are included at the beginning of each chapter. From an instructor’s point of view, the objectives of this chapter are to:
1. Discuss the various options that are available for distributing data in organizations.
2. Discuss the potential advantages and risks associated with distributed databases.
3. Enable the student to compare distributed database design strategies with respect to reliability, expandability, communications overhead costs, manageability, and data consistency.
4. Discuss the four types of transparency: location, replication, failure, and concurrency.
5. Consider both the need for and the accomplishment of query optimization.

Key Terms

<table>
<thead>
<tr>
<th>Asynchronous distributed database</th>
<th>Global transaction</th>
<th>Semijoin</th>
</tr>
</thead>
<tbody>
<tr>
<td>Commit protocol</td>
<td>Local autonomy</td>
<td>Synchronous distributed database</td>
</tr>
<tr>
<td>Concurrency transparency</td>
<td>Local transaction</td>
<td>Timestamping</td>
</tr>
<tr>
<td>Decentralized database</td>
<td>Location transparency</td>
<td>Transaction manager</td>
</tr>
<tr>
<td>Distributed database</td>
<td>Replication</td>
<td>Two-phase commit</td>
</tr>
<tr>
<td>Failure transparency</td>
<td>Transparency</td>
<td></td>
</tr>
</tbody>
</table>

Classroom Ideas

1. Assign your students the task of looking for examples of distributed database applications in the press. Discuss the issues in these examples based on material in the text.
2. Discuss the major options for distributing databases. Ask your students to help build a list of the advantages and disadvantages for each of these approaches.
3. Discuss the architecture of a distributed DBMS (Figures 14-2 and 14-3). Walk through the steps for both a local and remote database request.
4. Conduct a discussion of the four types of transparency: location, replication, failure, and concurrency. Relate this discussion to the database shown in Figure 14-10.
Comment on what support (if any) is provided for each type of transparency by the DBMS your students are using in the course.

5. Explain locking, deadlock, and timestamping in a distributed database environment.

6. Discuss the problem of query optimization in a distributed database environment using Table 14-2. This table shows the dramatic differences in query processing times depending on the strategy used.

7. Emphasize that the concepts illustrated in this chapter are not necessarily established in distributed database architectures. A review of current database magazines may produce comparisons of distributed database products which you can use to make students aware of current capabilities.

8. Discuss how simple distributed databases are created using middleware. For example, show how MS Access and Oracle can be linked via ODBC drivers, and an Access database can actually contain tables on both the local device running Access and a remote device running Oracle. Discuss the distributed database issues for this environment and how they should be managed.

Answers to Review Questions

1. Define each of the following terms:

   a. Distributed database A single logical database that is spread physically across computers in multiple locations that are connected by a data communications link.
   b. Location transparency A design goal for a distributed database that says that a user (or a user program), using data need not know the location of the data.
   c. Two-phase commit An algorithm for coordinating updates in a distributed database.
   d. Global transaction A transaction that requires reference to data at one or more nonlocal sites to satisfy the request.
   e. Local autonomy A design goal for a distributed database which says that a site can independently administer and operate its database when connections to other nodes have failed.
   f. Timestamping A concurrency control mechanism that assigns a globally unique timestamp (date and time) to each transaction. Timestamping is an alternative to the use of locks in distributed databases.
   g. Transaction manager A software module that maintains a log of all transactions and maintains an appropriate concurrency control scheme.

2. c replication transparency  
   e unit of work  
   d global transaction  
   b concurrency transparency  
   f replication  
   a failure transparency
3. Contrast the following terms:

a. **Distributed database; decentralized database** Both distributed and decentralized databases are stored on computers at multiple locations. In a decentralized database, however, a network does not interconnect the computers; users at various sites cannot share data. Thus, it is best regarded as a collection of independent databases rather than having the geographical distribution of a single database.

b. **Homogeneous distributed database; heterogeneous distributed database** In a homogeneous distributed database, the same DBMS is used at each node; in a heterogeneous distributed database potentially different DBMSs are used at each node. It is difficult in most organizations to force a homogeneous environment, yet heterogeneous environments are much more difficult to manage.

c. **Location transparency; local autonomy** In a distributed database, the network must allow users to share the data as transparently as possible (location transparency), yet it must allow each node to operate autonomously (local autonomy) when network linkages are broken or specific nodes fail.

d. **Asynchronous distributed database; synchronous distributed database** Synchronous technology ensures data integrity and minimizes the complexity of knowing where the most recent copy of data is located. Synchronous technology may result in unsatisfactorily slow response time because the distributed DBMS is spending considerable time in checking that an update is accurately and completely propagated across the network. Asynchronous technology tends to have acceptable response times because updates happen locally and data replicas are synchronized in batches at predetermined intervals. It may be more complex to plan and design to ensure exactly the right level of data integrity and consistency across the nodes.

e. **Horizontal partition; vertical partition** The advantages and disadvantages of both techniques are similar with the exception that combining data across vertical partitions is more difficult than across horizontal partitions. This difficulty arises from the need to match primary keys or other qualifications to join rows across partitions. Horizontal partitions support an organizational design in which functions are replicated, often on a regional basis; while vertical partitions are typically applied across organizational functions with reasonably separate data requirements.

f. **Full refresh; differential refresh** In full refresh, a snapshot of the replicated portion of the database is sent to each site that carries a copy of the replicated database. In differential refresh, only those pages that have changed because the last snapshot are sent to each site that carries a copy of the replicated database. In this case, a snapshot log for each replicated table is joined with the associated base to form a set of changed rows to be sent to the replicated sites.

g. **Push replication; pull replication** With push replication, the central master site decides when a local site is updated. Examples include snapshot replication and near real-time replication. With pull replication, the target, not the source node, controls when a local database is updated. The local database determines when it needs to be refreshed and requests a snapshot or the emptying of an update.
message queue. Pull strategies have the advantage that the local site controls when it needs and can handle updates. Thus synchronization is less disruptive and occurs when needed by each site.

h. **Local transaction; global transaction** A local transaction requires reference only to data that are stored at the site where the transaction originates, while a global transaction needs a reference to data at one or more nonlocal sites to satisfy the request.

4. Six business conditions that encourage using distributed databases:
   a. **Distributed and autonomous business units** Divisions, departments, and business units in modern organizations are often geographically (and possibly internationally) distributed. Often each unit has the autonomy to create its own information systems, and often these units want local data over which they can have controls.
   b. **Need for data sharing** Even moderately complex business decisions require sharing data over business units, so it must be convenient to consolidate data across local databases on demand.
   c. **Need to contain data communications costs and reliability** The cost to ship large quantities of data across a communications network or to handle a large volume of transactions from remote sources can be high. Also, dependencies on data communications can be risky; keeping local copies or fragments of data can be a reliable way to support the need for a rapid access to data across the organizations.
   d. **Multiple application vendor environments** Today, many organizations purchase packaged application software from several different vendors. Each “best in breed” package is designed to work with its own database and possibly with different database management systems. A distributed database can possibly be defined to provide functionality that cuts across the separate applications.
   e. **Database recovery** Replicating data on separate computers is one strategy for insuring that a damaged database can be quickly recovered and users can have access to data while the primary site is being restored. Replicating data across multiple computer sites is one natural form of a distributed database.
   f. **Satisfying both transaction and analytical processing** The requirements for database management vary across OLTP and OLAP applications. Yet, the same data are in common between the two databases supporting each type of application. Distributed database technology can be helpful in synchronizing data across OLTP and OLAP platforms.

5. Two types of homogeneous distributed databases:
   a. **Autonomous** Each DBMS works independently, passing messages back and forth to share data updates.
   b. **Nonautonomous** A central, or master, DBMS coordinates database accesses and updates across the network.
6. Five characteristics of homogeneous distributed databases:
   a. Data are distributed across all the nodes.
   b. The same DBMS is used at each location.
   c. All data are managed by the distributed DBMS (so there is no exclusively local data).
   d. All users access the database through one exclusive schema or database definition.
   e. The global schema is simply the union of all the local database schemas.

7. Four characteristics of heterogeneous distributed databases:
   a. Data are distributed across all the nodes.
   b. Different DBMS is used at each location.
   c. Some users require only local access to databases, which can be accomplished using only the local DBMS and schema.
   d. A global schema exists, which allows local users to access remote data.

8. Five advantages of distributed databases compared to centralized databases:
   a. Increased availability and reliability When a centralized system fails, the database is unavailable to all users. A distributed system will continue to function at some reduced level however, even when a component fails.
   b. Local control Distributing the data encourages local groups to exercise greater control over “their” data, which promotes improved data integrity and administration. At the same time, users can access nonlocal data if necessary.
   c. Modular growth Suppose that an organization expands to a new location or adds a new work group. It is often easier and more economical to add a local computer and its associated data to the distributed network than to expand a large central computer. Also, there is less chance of disruption to existing users than in the case of a central system.
   d. Lower communication costs With a distributed database, data can be located closer to point of use. This can reduce communication costs, compared to a central system.
   e. Faster response Depending on how data are distributed, most requests for data by the local users can be satisfied by data stored at the local site. This speeds up query processing because communication and central computer delays are minimized.

9. Four costs and disadvantages of distributed databases:
   a. Software cost and complexity More complex software (especially the DBMS) is required for a distributed database.
   b. Higher processing overhead The various sites must exchange messages and perform additional calculations to ensure proper coordination among data at the different sites.
c. **Data integrity maintenance** A by-product of the increased complexity and need for coordination is the additional exposure to improper updating and other problems of data integrity.

d. **Slow response** If the data are not distributed properly according to their usage, or if queries are not formulated correctly, response to requests for data can be extremely slow.

10. Five advantages of the data replication form of distributed databases:

a. **Reliability** If one of the sites containing a relation (or the database) fails, a copy can always be found at another site without network traffic delays. Also, available copies can all be updated as soon as possible when transactions occur, and unavailable nodes will be updated once they return to service.

b. **Fast response** Each site that has a full copy can process queries locally so queries can be processed rapidly.

c. **Possible avoidance of complicated distributed transaction integrity routines** Replicated databases are usually refreshed at scheduled intervals, so most forms of replication are used when some relaxing of synchronization across the database copies is acceptable.

d. **Node decoupling** Each transaction may proceed without coordination across the network. Thus, if nodes are down, busy, or disconnected (e.g., in the case of mobile personal computers), a transaction is handled when the user desires. In the place of real-time synchronization of updates, a behind-the-scenes process coordinates all data copies.

e. **Reduced network traffic at prime time** Often, updating data happens during prime business hours, when network traffic is highest and the demands for rapid response greatest. Replication, with delayed updating of copies of data, moves network traffic for sending updates to other nodes to non-prime-time hours.

11. Two disadvantages of the data replication form of distributed databases:

a. **Storage requirements** Each site that has a full copy must have the same storage capacity that would be required if data were stored centrally. Each copy requires storage space, and processing time is required to update each copy on each node.

b. **Complexity and cost of updating** Whenever a relation is updated, it must eventually be updated at each site that holds a copy. Synchronizing these updates near real-time requires careful coordination.

12. A snapshot replication strategy works best when:

The environment has a single updater. In this case, effects of the multiple updates are effectively batched for the read-only sites. Product catalogs, price lists, and other reference data for a mobile sales force are likely to be appropriate for the snapshot replication approach.
13. A near real-time replication works best when:

The environment has multiple updaters. Each database update event can be handled individually with the use of triggers.

14. Five factors that influence the viability of data replication:

a. Data timeliness Applications that can tolerate out-of-date data (whether this be for a few seconds or a few hours) are better candidates for replication.

b. DBMS capabilities An important DBMS capability is whether it will support a query that references data from more than one node. If not, then replication is a better candidate than partitioning schemes.

c. Performance implications Replication means that each node is periodically refreshed. When this refreshing occurs, the distributed node may be very busy handling a large volume of updates. If the refreshing occurs by event triggers (for example, when a certain volume of changes accumulate), refreshing could occur at a time when the remote node is busy doing local work.

d. Heterogeneity in the network Replication can be complicated if different nodes use different operating systems, DBMSs, or more commonly, different database designs. Mapping changes from one site to n other sites could mean n different routines to translate the changes from the originating node into the scheme for processing at the other nodes.

e. Communications network capabilities: Transmission speeds and capacity in a data communications network may prohibit frequent, complete refreshing of very large tables. Replication does not require a dedicated communications collection, however, so less expensive, shared networks could be used for database snapshot transmission.

15. Advantages and disadvantages of horizontal partitioning:

Advantages:
- Efficiency Data are stored close to where they are used and separate from other data used by other users or applications.
- Local optimization Data can be stored to optimize performance for local access.
- Security Data not relevant to usage at a particular site are made unavailable.
- Ease of querying Combining data across horizontal partitions is easy because rows are simply merged by unions across the partitions.

Disadvantages:
- Inconsistent access speed When data from several partitions are required, the access time can be significantly different from local-only data access.
- Backup vulnerability Because data are not replicated, when data at one site become inaccessible or damaged, usage cannot switch to another site where a copy exists; data may be lost if proper backup is not performed at each site.
16. Advantages and disadvantages of vertical partitioning:

The advantages and disadvantages of vertical partitioning are identical to those of horizontal partitioning, with the exception that combining data across vertical partitions is more difficult than across horizontal partitions. This difficulty arises from the need to match primary keys or other qualifications to join rows across partitions.

17. Five factors that influence the selection of a distributed database design strategy:

   a. **Organizational forces** Funding availability, autonomy of organizational units, and the need for security.
   
   b. **Frequency and location or clustering of reference to data** In general, data should be located close to the applications that use those data.
   
   c. **Need for growth and expansion** The availability of processors on the network will influence where data may be located and applications may be run; this may indicate the need for expansion of the network.
   
   d. **Technological capabilities** Capabilities at each node and for DBMSs coupled with the costs for acquiring and managing technology must be considered. Storage costs tend to be low, but the costs for managing complex technology can be great.
   
   e. **Need for reliable service** Mission-critical applications and very frequently required data encourage replication schemes.

18. Six unique functions of a distributed DBMS:

   a. Keep track of where data are located in a distributed dictionary.
   
   b. Determine the location from which to retrieve requested data and the location at which to process each part of a distributed query.
   
   c. If necessary, translate the request at one node using a local DBMS into the proper request to another node using a different DBMS and data model; return data to the requesting node in the format accepted for that node.
   
   d. Provide data management functions such as security, concurrency and deadlock control, query optimization, and failure recovery.
   
   e. Provide consistency among copies of data across the remote sites.
   
   f. Present a single logical database that is physically distributed.

19. Effect of location transparency:

With location transparency, the user of an ad hoc data query need not be aware that required data exist at various sites, and therefore, that this is a global (rather than local) transaction.
20. Effect of replication transparency:

With replication transparency, the user of an ad hoc data query need not be aware that identical data are stored at other sites. If a read request originates at a site that does not contain the requested data, the user need not be aware that the request has to be routed to another site, resulting in a global, rather than local, transaction.

21. How a two-phase commit can still fail to create a completely consistent distributed database:

If a transaction fails during the commit phase as it attempts commits at the involved remote sites, the transaction will be in limbo; this may result in an inconsistent database, as some commits have occurred, but not all.

22. Three improvements to the two-phase commit:

a. **Read-only commit optimization**  This approach identifies read-only portions of a transaction and eliminates the need for confirmation messages on these portions. For example, a transaction might include checking an inventory balance before entering a new order. The reading of the inventory balance within the transaction boundaries can occur without the callback confirmation.

b. **Lazy commit optimization**  This approach allows those sites which can update to proceed with updating, and other sites which cannot immediately update are allowed to “catch up” later.

c. **Linear commit optimization**  This approach permits each part of a transaction to be committed in sequence rather than hold up a whole transaction when subtraction parts are delayed from being processed.

23. Three steps in distributed query processing:

a. **Query decomposition**  In this step, the query is simplified and rewritten into a structured, relational algebra form.

b. **Data localization**  Here, the query is transformed from a query referencing data across the network as if the database were in one location, into one or more fragments that each explicitly reference data at only one site.

c. **Global optimization**  In this final step, decisions are made about the order in which to execute query fragments, where to move data between sites, and where parts of the query will be executed.

24. Conditions that suggest faster distributed query processing by using a semijoin:

In a semijoin, only the joining attribute is sent from one site to another, and then only the required rows are returned. If only a small percentage of the rows participate in the join, then the amount of data being transferred is minimal. Clearly, the semijoin
saves network traffic, which can be a major contributing factor to the overall time to respond to a user’s query.

Answers to Problems and Exercises

1.
   a. Concurrency transparency
   b. Replication transparency
   c. Failure transparency
   d. Location transparency

2.
   a. UPDATE PART
      SET UNIT_PRICE = UNIT_PRICE * 1.10
      WHERE PART_NUMBER = 56789;
   b. i. Not acceptable. The remote unit of work allows updates at a single remote computer. Thus, updates cannot be made simultaneously at more than one location—three in the case of Figure 14-9.
      ii. Not acceptable. The distributed unit of work allows various statements within a unit of work to refer to multiple remote locations. Updates however, cannot be made simultaneously at more than one location because all tables in a single SQL statement must be at the same location.
      iii. Acceptable. A distributed request allows a single SQL statement to refer to tables at more than one remote DBMS.

3.  a. UPDATE PART
     SET BALANCE = BALANCE * 1.10
     WHERE PART_NUMBER = 56789 AND LOCATION = "San Mateo";
     UPDATE PART
     SET BALANCE = BALANCE * .9
     WHERE PART_NUMBER = 56789 AND LOCATION = "New York";
   b. i. Not acceptable. The transaction in 3a., is essentially a sequence of SQL statements originated at one location, but attempted to be executed by multiple remote DBMSs. The remote unit of work protocol would support only transactions attempted to be executed at a single remote location, and in addition, the same DBMS has to run on the remote computer.
      ii. Acceptable. The distributed unit of work protocol does support protected updates involving multiple sites, provided that each SQL statement refers to table(s) at one site only.
      iii. Cannot determine. The distributed request protocol may or may not support failure transparency.
4. Heterogeneous distributed databases are very difficult database environments because of the necessity to maintain communication, data integrity, efficiency, optimization, and security. Databases at different locations may be based on different models, perhaps relational and network. Products that make it possible to communicate among mixed DBMSs and data models are just beginning to be available. It is a challenging task to optimize a database that runs on one platform; optimizing one across platforms can be a nightmare. The goals of developing open systems without such barriers continue to be elusive. Building and maintaining a production database in a heterogeneous environment is one of the most challenging information systems tasks in businesses now.

5. The drastically different results of the queries shown in Table 14-2 come primarily from the way each query results in data being moved in order to be processed. In general, it is often advisable to break a query in a distributed database environment into components that are isolated at different sites. Then determine which site has the potential to yield the fewest qualified records, and move these results to another site where additional work is performed.

6. The fifth strategy in Table 14-2 utilizes a semijoin. The joining attribute in this case is represented by a composite primary key and is sent to the remote location (Chicago). The Detroit computer however, would not yield the fewest qualified records for this particular query (Table 14-2). (The 1 million SHIPMENT records in Detroit, joined with the 10,000 supplier records, would yield more qualified records than the 1 million records stored in Chicago.) A large percentage of the returned rows participate in this join.

7. a. SELECT PART.PART_NUMBER, COLOR
   FROM PART, SUPPLIER, SHIPMENT
   WHERE SUPPLIER.CITY <> "Columbus"
   AND SHIPMENT.PART_NUMBER = PART.PART_NUMBER

   b. 1. Move PART relation to Detroit, and process the whole query at the Detroit computer.
   2. Join the SUPPLIER and SHIPMENT at the Detroit computer, PROJECT these down to suppliers not from Columbus, and move the result to Chicago for matching with PARTS.
   3. Join the SUPPLIER and SHIPMENT at the Detroit computer, PROJECT just SUPPLIER_NUMBER and PART_NUMBER for suppliers not from Columbus, and move this qualified projection to Chicago for matching with PARTS.
c.  
1. 100,000 records in PART * 100 char = 10,000,000 char to be transferred.  
   10,000,000 char / 10,000 char/sec = 1,000 seconds = 16.7 min.  

2. After the JOIN operation is completed, the result table will have a total of  
   1,000,000 records. Each of those records will be essentially the  
   SHIPMENT relation’s record, extended with more information about its  
   supplier, i.e., CITY. The join operation is a theta join over the Cartesian  
   product of the two relations, where the qualifying expression is equality.  
   Another way to look at it is to consider it a selection over the Cartesian  
   product, where the expression is an equality between  
   SUPPLIER.Supplier_Number and SHIPMENT.Supplier_Number. The  
   worst case is the maximum number of tuples in SHIPMENT. We have to  
   exclude tuples with Supplier.City <> Columbus (which we assume are  
   100,000). Each record will be 150 char each because there will be three  
   attributes in the table: Supplier_Number, Part_Number, and City.  
   
   \[(1,000,000 - 100,000) \text{ records} \times 150 \text{ char} = 135,000,000 \text{ char}.\]  
   135,000,000 char / 10,000 char/sec = 13,500 seconds = 225 min = 3.75 hrs.  

3. After the JOIN operation’s completion, the result table will have  
   1,000,000 records. We have to exclude tuples with Supplier.City <>  
   Columbus (which we assume are 100,000). Each record will be 100 char each because there will be only two  
   attributes in the table: Supplier_Number, Part_Number, and City.  
   \[(1,000,000 - 100,000) \text{ records} \times 100 \text{ char} = 90,000,000 \text{ char}.\]  
   90,000,000 char / 10,000 char/sec = 9,000 seconds = 150 min = 2.5 hrs.  

<table>
<thead>
<tr>
<th>Method</th>
<th>Time</th>
</tr>
</thead>
</table>
| 1. Move PART relation to Detroit, and process the whole query  
   at the Detroit computer. | 16.7 minutes |
| 2. Join the SUPPLIER and SHIPMENT at the Detroit computer,  
   PROJECT these down to suppliers not from Columbus, and  
   move the result to Chicago for matching with PARTS. | 3.75 hours*   |
| 3. Join the SUPPLIER and SHIPMENT at the Detroit computer,  
   PROJECT just SUPPLIER_NUMBER and  
   PART_NUMBER for suppliers not from Columbus, and  
   move this qualified projection to Chicago for matching with  
   PARTS. | 2.5 hours*    |

* We assume a history of 100,000 shipments from Columbus.  

D. Because SHIPMENTS from Cleveland are 100,000, SHIPMENTS from Columbus  
   could not exceed the number of 900,000. Or, to view the problem from another
perspective, all PART records will always be less as a volume than SHIPMENTS not from Columbus. Hence, the first strategy is the optimal solution. Because the least number of records need to be transferred, the timing is the best among the three alternatives.

e. **Data replication** Each site that will have a full copy of the database can process the queries locally, so they can be processed rapidly.  
   **Horizontal partitioning** Would allow isolation at multiple sites for SUPPLIERS according to their location. Hence, we wouldn’t have to scan all records in the SUPPLIER table in order to select those that are not Columbus-located SUPPLIERS. Combining those records would be easy because they could be merged by unions across the partitions.  
   **Vertical partitioning** Would be impossible in the case of 7.a. because all the tables in the example consist of records of two attributes only.

8. 
   a. The distributed database solution will have several advantages compared to a centralized system. When a centralized system fails, the database is unavailable to all users, regardless of their location. A distributed system will continue to function at some reduced level, however, even when a component fails. For a large retail chain with a large number of stores and store departments, availability is a serious consideration. The distributed database solution makes it easier to add a new location to the distributed network because it would be more economical to add a new local computer than to expand the existing central machine. There is less chance of disruption to existing users too. Slow response time, however, will still be a concern because schedule updates at each store are made five times per hour. A decentralized solution would not provide an opportunity for the corporation to control the information at the various locations because the local databases will not be connected in a corporate network.

   b. Partially or totally replicated across geographically distributed sites, with each copy periodically updated with snapshots.

9. Because the data is only updated weekly, a distributed database is recommended. We don’t have information on the telecommunications cost associated with connecting to systems in Florida; however, we can assume that it would be less expensive to send data weekly to Florida than have users constantly access a central database in the main office.

10. The best data replication strategy for this scenario would be snapshot replication. Because the entire data mart is refreshed every weekend, we would need to perform a full refresh of the replicated database. Depending upon the time and cost to do this, we may need to rethink how we refresh the data mart. An incremental refresh of the data mart might be more efficient because an incremental refresh could also be done of the replicated database.
11. Because data is static at all sites, the data mart would be refreshed weekly at the central location. If we assume that a full refresh is performed (from Chapter 11), then the replicated database would also need a full refresh. As mentioned in the answer for Problems and Exercises 10, if the data mart’s method for loading data was changed to an incremental refresh, then the replicated database could also be incrementally refreshed.

12. Because there are ten different sites, changing the data mart refresh policy to employ an incremental refresh on a weekly basis is recommended. This would enable the distributed database to utilize snapshot replication with incremental refresh.

13. Horizontal partitioning could be used. Each office would receive only a copy of records relevant to the agents employed by that office. This would be accomplished by partitioning the fact table as well as the dimension tables of the Fitchwood star schema (see Problems and Exercises in Chapter 11).

14. Depending upon the volume of transactions, snapshot replication would generally not be the best approach because there could be some delays in updating. The best approach would be to utilize synchronized replication if some delays could be tolerated. If the system could not tolerate delays, then a centralized system may need to be implemented.

15. Left as a student exercise.

16. A distributed database employing synchronized replication is recommended. Because order fulfillment could tolerate some delays, this approach would work well. A centralized approach would be more cost prohibitive because both facilities would always have to be connected.

17. If a centralized approach was used, there would be no delay in updating such data as inventory levels. Also, there would not be a need for replicating data across the distributed DBMS. However, depending upon the distance between facilities and the volume of data transmitted, there could be substantial telecommunication costs. Also, the replicated approach provides some level of redundancy should there be a hardware or database failure at one site.

18. A synchronized replication approach is recommended. This approach is applicable because a small amount of delay between updates could be tolerated, particularly if updates to inventory levels were timed to correspond to when actual orders were fulfilled.

19. Horizontal partitioning is recommended. The remote order processing facility would still be replicated, but only with data on items available to customers west of the Mississippi.
20. An additional table, SHIPORIGIN_T, should be added which contains information about the warehouse as well as the manufacturing facility, including a unique ID for both. The Product_On_Hand column from the PRODUCT_T table must be removed. A new column, build_type, must be added to PRODUCT_T. This column will indicate whether the item is custom built or a speciality item. A new table, PRODUCT_QUANTITY_T, which contains the product_id, shiporigin_id, and quantity_on_hand, will be needed. This table will keep track of the quantity of products on-hand at both the warehouse and the manufacturing facility. Regarding distribution strategies, the order processing facility as well as the warehouse should have replicated copies of the database utilizing horizontal partitioning of the customer_t table to exclude international and East Coast customers. The best strategy for replication would be synchronized replication.

Answers to Field Exercises

1. a. In order for the database to be considered distributed, the DBMS at each site must perform the following functions in addition to managing the local database:
   1. Keep track of where data are located in a distributed dictionary.
   2. Determine the location from which to retrieve requested data and the location at which to process each part of a distributed query.
   3. If necessary, translate the request at one node using a local DBMS into the proper request to another node using a different DBMS and data model, and return data to the requesting node in the format accepted for that node.
   4. Provide data management functions such as security, concurrency and deadlock control, query optimization, and failure recovery.
   5. Provide consistency among copies of data across the remote sites.

b. Some important features of those products are summarized in Table 14-3 in the text. It is very difficult to select among those products and features because the exact capabilities of the product must match the organization’s needs. It is impossible to outline general principles for managing a distributed database.

c. Following you will find a detailed description of some problems related to each of the terms. A good approach to preparing the questions to be asked, is to put an emphasis on those aspects.

Location transparency In general, it is easier to update data on one remote site than having to update data stored at multiple sites. To achieve location transparency, the distributed DBMS must have access to a current data dictionary that indicates the location(s) of all data across the network.

Replication transparency Problems arise when one or more users attempt to update replicated data. The updates will have to be accomplished accurately and concurrently at all the sites holding a replicated copy.
Failure transparency  It is achieved by overcoming failure problems, including system failures such as erroneous data, disk head crash, etc. Communications link failures are also a problem.

Concurrency control  Three basic approaches in achieving concurrency transparency are locking, versioning, and time-stamping.

Query optimization  Both the user’s formulation and the intelligence of the DBMS to develop a sensible plan for processing affect query processing.

d. Current releases of DBMSs do not provide all of the known features of distributed database products. For example, some products provide location transparency for read-only transactions, but they do not yet support global updates.

e. The following factors determine the choice of a data distribution strategy:
   1. **Organizational forces** Funding availability, autonomy of organizational units, and the need for security.
   2. **Frequency and locality, or clustering of reference to data** In general, data should be located close to the applications that use those data.
   3. **Need for growth and expansion** The availability of processors on the network will influence where data may be located and when applications may be run and may indicate the need for expansion of the network.
   4. **Technological capabilities** Capabilities at each node and for DBMSs coupled with the costs for acquiring and managing technology must be considered. Storage costs tend to be low, but the costs for managing complex technology can be great.
   5. **Need for reliable service** Mission-critical applications and very frequently required data encourage replication schemes.

2. This exercise will stimulate various research strategies, and answers received will depend on the current state of the products that students investigate.

3.

a. In client/server environments, it is easy to define a database with tables on several nodes in a local or wide area network. Once a user program establishes a linkage with each remote site and suitable database middleware is loaded, full location transparency is achieved. So in client/server database form, distributed databases are readily available to any information systems developer, and heterogeneity of DBMS is possible.

b. The following benefits of distributed systems have to be considered:
   - Increased availability and reliability
   - Local control
   - Modular growth
   - Lower communication cost
   - Faster response
Chapter 15 Object-Oriented Data Modeling

Chapter Overview

Please note that the material for this chapter is based upon the Web version of Chapter 15, not the abbreviated version of Chapter 15 in the text.

The purpose of this chapter is to introduce object-oriented modeling concepts. This chapter presents the UML notation and defines the key terms associated with object-oriented modeling. Most of the concepts learned in Chapters 3 and 4 correspond to concepts in object-oriented modeling, but as this chapter presents, the object-oriented modeling approach builds upon and extends the EER model.

Chapter Objectives

Specific student learning objectives are included in the beginning of the chapter. From an instructor’s point of view, the objectives of this chapter are to:

1. Explore the similarities between the major steps in database development and relevant elements in the object-oriented development process.
2. Present the object-oriented model technique using the UML notation to model real-world situations.
3. Emphasize the importance of building a model and thinking in abstract terms rather than worrying about implementation details in the analysis phase. Focus on describing what the intended system must do, rather than how it will be done.
4. Introduce the terminology associated with object-oriented modeling and recognize when to use generalization, aggregation, and composition relationships to more accurately represent real-world systems.

Key Terms

<table>
<thead>
<tr>
<th>Abstract class</th>
<th>Class-scope attribute</th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract operation</td>
<td>Class-scope operation</td>
<td>Object diagram</td>
</tr>
<tr>
<td>Aggregation</td>
<td>Composition</td>
<td>Operation</td>
</tr>
<tr>
<td>Association</td>
<td>Concrete class</td>
<td>Overriding</td>
</tr>
<tr>
<td>Association class</td>
<td>Constructor operation</td>
<td>Polymorphism</td>
</tr>
<tr>
<td>Association role</td>
<td>Encapsulation</td>
<td>Query operation</td>
</tr>
<tr>
<td>Behavior</td>
<td>Method</td>
<td>State</td>
</tr>
<tr>
<td>Class</td>
<td>Multiple classification</td>
<td>Update operation</td>
</tr>
<tr>
<td>Class diagram</td>
<td>Multiplicity</td>
<td></td>
</tr>
</tbody>
</table>
Classroom Ideas

1. Review the major steps in the object-oriented development life cycle (Figure 15-1).
2. Use the sample object-oriented diagram shown in Figure 15-2 to “jump-start” your students’ understanding.
3. Use Figure 15-18 to demonstrate how complex business systems can be represented using the object-oriented notation.
4. Present the object-oriented concept of encapsulation.
5. Discuss the use of generalization and its role in simplifying complex systems. Reference Figure 15-9. Also discuss how this object-oriented feature should facilitate easier maintenance of the system.
6. Discuss the use of overriding and the difference between overriding for extension versus restriction.
7. Discuss the use of aggregation versus association.
8. If your students are familiar with some object-oriented development environment, you can have them investigate (or you can show them in class) the various tools available in the environment to do the type of data modeling illustrated in this chapter.

Answers to Review Questions

1. Define each of the following key terms:

   a. **Class** An entity that has a well-defined role in the application domain about which the organization wishes to maintain state, behavior, and identity.
   b. **State** Encompasses an object’s properties (attributes and relationships) and the values those properties have.
   c. **Behavior** Represents how an object acts and reacts.
   d. **Encapsulation** The technique of hiding the internal implementation details of an object from its external view.
   e. **Operation** A function or a service that is provided by all the instances of a class.
   f. **Method** The implementation of an operation.
   g. **Constructor operation** An operation that creates a new instance of a class.
   h. **Query operation** An operation that accesses the state of an object but does not alter the state.
   i. **Update operation** An operation that alters the state of an object.
   j. **Abstract class** A class that has no direct instances, but whose descendants may have direct instances.
   k. **Concrete class** A class that can have direct instances.
   l. **Abstract operation** Defines the form or protocol of the operation, but not its implementation.
   m. **Multiplicity** Indicates how many objects participate in a given relationship.
   n. **Class-scope attribute** An attribute of a class that specifies a value common to an entire class rather than a specific value of an instance.
   o. **Association class** An association that has attributes or operations of its own or participates in relationships with other classes.
p. Polymorphism  The same operation may apply to two or more classes in different ways.
q. Overriding  The process of replacing a method inherited from a superclass by a more specific implementation of that method in a subclass.
r. Multiple classification  An object is an instance of more than one class.
s. Composition  A part object which belongs to only one whole object and which lives and dies with the whole object.
t. Recursive aggregation  A part-of relationship between a component object and itself, treating itself as the aggregate object.

2. Match the following terms to the appropriate definitions:

```plaintext
c  concrete class  
b  abstract operation  
f  aggregation  
e  overriding  
a  polymorphism  
h  association class  
d  composition  
g  class  
```

3. Contrast the following terms:

a. Class; object  Class refers to the structure of an object, whereas an object refers to the individual instances.
b. Attribute; operation  Attribute refers to an object’s data; operation refers to a function or a service that is provided by all the instances of a class.
c. State; behavior  State refers to an object’s properties and is characterized by its attributes; behavior refers to how an object acts and reacts.
d. Operation; method  An operation refers to an object’s functions or services that are provided by all the instances of a class. A method refers to how these functions or services are implemented.
e. Query operation; update operation  Both are operations, but the update operation alters the state of an object.
f. Abstract class; concrete class  Instances of objects can be created from concrete classes; they cannot be created from abstract classes.
g. Class diagram; object diagram  A class diagram shows the static structure of an object-oriented model, whereas the object diagram depicts instances of objects.
h. Association; aggregation  An association indicates a relationship between object classes. An aggregation indicates that an object class is part of another class.
i. Generalization; aggregation  Generalization relates object classes through the process of abstracting common attributes, operations, and relationships of a set of object classes; whereas aggregation relates distinct object classes with a part-of association.
j. Aggregation; composition  Aggregation relates the part-of relationship between objects that may or may not exist. A composition refers to objects that live and die with the whole object.
k. **Overriding for extension; overriding for restriction** Both types of overriding reference the overridden method. Overriding for extension adds new processing to the existing method, whereas overriding for restriction adds new restrictions that must be met in order to process the existing method.

4. Activities involved in the object-oriented development life cycle phases:

a. **Object-oriented analysis:** Develop a model of the real-world application showing its important properties.

b. **Object-oriented design** Define how the application-oriented analysis model will be realized in the implementation environment.

c. **Object-oriented implementation** Implement the design using a programming language and/or a database management system.

5. Compare and contrast the object-oriented model with the enhanced E-R model.

An object-oriented model is built around classes of objects, just as the E-R model is built around entity types. However, an object encapsulates both data and behavior, implying that we can use the object-oriented approach not only for data modeling, but also for process and event modeling.

6. When to model an association relationship as an association class:

When an association itself has attributes or operations of its own or when it participates in relationships with other classes, it is useful to model the association as an association class.

7. Class diagrams of unary, binary, and ternary relationships, specifying multiplicities:
8. Role names for Review Question 7 association relationships:

**Binary**

**Unary**

**Ternary**
9. Add operations to some of the classes you identified in Review Question 7

![Class diagram with operations](image)

10. An example of generalization:

![Generalization diagram](image)
11. Extension of Review Question 10 adding at least one abstract class with at least one abstract operation, and indicating which features of a class are inherited by other classes:

```
<<abstract>>
Employee
  ssn
  earnings
<<abstract>> computePay()
```

```
Hourly Employee
  hourlyRate
  overtimeRate
  computePay()

Salared Employee
  weeklyRate
  computePay()

Exempt Employee
  monthlyRate
  computePay()
```

All features from the abstract class are inherited

12. Extension of Review Question 11, giving an example of polymorphism. The specific details of the implementation of the method computePay() depend on the subclass even though they all receive the same message from the classes that request the computePay() service.

```
<<abstract>>
Employee
  ssn
  earnings
<<abstract>> computePay()
```

```
Hourly Employee
  hourlyRate
  overtimeRate
  computePay()

Salared Employee
  weeklyRate
  computePay()

Exempt Employee
  monthlyRate
  computePay()
```

Published by www.emagazine.org
13. Aggregation example:

![Aggregation Diagram]

14. The concepts of encapsulation, inheritance, and polymorphism make object-oriented modeling a powerful tool for developing complex systems.

15. Because Vehicle is an abstract class, it cannot have any direct instances.

16. In order to effectively model a complex system, it is necessary to have a small set of independent views illustrating the problem from multiple perspectives. The UML provides many different types of diagrams in order to provide these different perspectives.

17. The assignment class is an association class.

18. A unary relationship would have to be represented as an association class if the relationship has attributes, such as quantity or begin and end dates.

19. /AvailBalance is a derived attribute. /purchases is a derived relationship. /AvailBalance can be computed from the valued for creditLine and balance. /purchases can be determined by examining the association between customers and items.

20. CheckFee and MonthlyFee are examples of class-scope attributes. CalcFee is an abstract operation.

21. The class diagram shown in Figure 15-23 is an example of multiple-inheritance.

22. The class diagram shown in Figure 15-24 is an example of aggregation. Although faculty could be represented as an aggregate of department, there really is not a need to do this because there will not be operations on department that will carry through to faculty. Also, faculty has independent relationships with other objects, such as courses.
Answers to Problems and Exercises

1. Class diagrams:

   a. **Note:** The Association Class in the following diagram should be named “Assignment.” This change will be incorporated into future editions of the Instructor’s Manual.

```
<table>
<thead>
<tr>
<th>Employee</th>
</tr>
</thead>
<tbody>
<tr>
<td>employeeID</td>
</tr>
<tr>
<td>name</td>
</tr>
<tr>
<td>address</td>
</tr>
<tr>
<td>birthDate</td>
</tr>
<tr>
<td>mailCheck()</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Project</th>
</tr>
</thead>
<tbody>
<tr>
<td>projectName</td>
</tr>
<tr>
<td>startDate</td>
</tr>
</tbody>
</table>

1..*  *

billingRate
hours
calcPayAmt()
```

b. 

```
<table>
<thead>
<tr>
<th>Course</th>
</tr>
</thead>
<tbody>
<tr>
<td>courseNumber</td>
</tr>
<tr>
<td>courseName</td>
</tr>
<tr>
<td>units</td>
</tr>
<tr>
<td>addPrereq(prereq)</td>
</tr>
<tr>
<td>dropPrereq(prereq)</td>
</tr>
</tbody>
</table>

1 Has-Prerequisites
```
c. 

\[ \text{Chemist} \]

- name
- phoneNumber
- avgHours(equipment)

\[ \text{Project} \]

- projectName
- startDate

\[ \text{Equipment} \]

- serialNo
- cost
- type

\[ \text{Assignment} \]

- assignDate
- totalHours

\[ \text{Course} \]

- courseId
- courseName
- units
- findNumSections(semester)

\[ \text{Section} \]

- sectionNumber
- semester

**Note:** Because primary keys are not included on class diagrams, there is no need to consider weak classes in this solution.
2. **Note**: The attribute names in the following diagram should begin with lower case letters (e.g., Address should be address). The Association Class should be named “Participation.” These changes will be incorporated into future editions of the Instructor’s Manual.
3. Class diagram with subset constraint:

4. Sample class diagram with at least four association roles:
6.
7. This is a simple library solution; more classes could be added such as author, categories, etc. itemType includes Paperback, Hardback, audiobook CD, audiobook tape, audio CD, audio tape, video DVD, and videotape; this could also be set up as reference data to the itemType in Item class. Based on reading the problem description, there do not appear to be special attributes or relationships based on the itemType thus no subclasses for itemType are presented in this solution.
8.  

9. **Note**: Account should be marked as an abstract class. This change will be incorporated into future editions of the Instructor’s Manual
A patient assigned to a bed must be first assigned to be treated by a physician.
11. a.
11. b.
11. c.
12.
13. Please note for this exercise that the composite attribute of Address could be a class also, because both customer and location have common address attributes.

**Note:** The attribute names in the following diagram should begin with lower case letters (e.g., CustomerID should be customerID). The Association Class should be named “LocationRates.” The “TimeOfDay” attribute should be broken into two attributes of “startTime” and “endTime.” These changes will be incorporated into future editions of the Instructor’s Manual.
14. **Note**: The attribute names in the following diagram should begin with lower case letters (e.g., EmpID should be empID). The “Consists_of” association should be named “Consists_of.” These changes will be incorporated into future editions of the Instructor’s Manual.
15. **Note:** The attribute names in the following diagram should begin with lower case letters (e.g., Address should be address). Attribute “Court_Name” should be “courtName”; “Specialty_Name” should be “specialtyName.” The “updateyears()” operation should be named “updateYears().” The association class should be named “Assignment.” These changes will be incorporated into future editions of the Instructor’s Manual.

16. **Note:** The attribute names in the following diagram should begin with lower case letters (e.g., EmpID should be empID). The Association Class should be named “Assignment.” These changes will be incorporated into future editions of the Instructor’s Manual.
16. Class diagram
17. Please note in this solution that Estimates and ServicesPerformed both are related to location rather than customer. Although the exercise does not explicitly state this as a business rule, it has been assumed that an estimate or service performed would be for only one location, even if a customer owned multiple locations.

**Note:** The attribute names in the following diagram should begin with lower case letters (e.g., EmpID should be empID). The operation names should begin with lower case letters. The Association Class should be named “Expertise.” The multiplicity at the “Technical” end for the “Technical”–“Technical Skills” association should be 0…*. These changes will be incorporated into future editions of the Instructor’s Manual.

The class diagram solution is below.
17. Class diagram

```
Degree
---
DegreeCompleted
Institution
DateCompleted

Consultant
---
EmpID
Name
DOB
Street
City
State
Zip
Age
Date_Age()

Business
---
Completed

Technical
---
 Attained

BusinessExperience
---
Type
Description

Estimates
---
Date
Amount

Customer
---
CostID
CompanyName
ContactName
ContactTitle
ContactTelephone
BusinessType
Street
City
State
ZipCode
NumberofEmployees
updateCost(amount)
updateContact()

Location
---
LocID
Street
City
State
Zip
Telephone
StatePhone

Service
---
ServiceID
Description
Cost
Coverage
Expiration
updateCost()
updateCoverage()

Published by www.emagazine.org
```
18. **Note**: The attribute names in the following diagram should begin with lower case letters (e.g., Street should be street). These changes will be incorporated into future editions of the Instructor’s Manual.
19. This solution assumes that a single issue might have several “history instances” associated with each issue.
20. **Note:** In the following diagram, the “numberOfItem(assembly)” operation should be named “getNumberOfItems(assembly).” This change will be made in a future edition of the Instructor’s Manual.
21. **Note:** This solution refers to Figure 15-18 in the text. For this solution, the receivePaymt method has been overloaded for cases where an electronic payment is needed.
22. This solution identifies several derived associations (Customer purchases Product, Salesperson sells Product, Union Employee produces Product, Supplier supplies for Product), although there are more to be found.
Answers to Field Exercises

1. An example of a superclass/subclass relationship is with equity securities. Equity securities represent ownership shares in a corporation. Equity securities have attributes such as name, symbol, exchange, and price. A particular type of equity security is common stock. Preferred stock is another type equity security that offers the holders fixed dividends each year. An attribute for preferred stock would be dividend rate.

2. Examples of superclass/subclass relationships should be available in both service and manufacturing businesses. In the service business you will find several classes of employees, ranging from part-time to full-time with benefits. In the manufacturing business you might find several types of engineers (electrical, mechanical). Numerous operational business rules can be found in both types of businesses. In manufacturing, rules exist regarding quality standards and how overhead should be allocated to jobs. In the service industry, rules related to processes, such as how an item is purchased, represent sequences in the object-oriented models.

3. The translation of the EER (or E-R) diagram should map the entity types and attributes into classes. EER diagrams can represent generalizations that also map to class diagrams. EER and E-R notation are different than UML. Association relationships remain the same. E-R diagrams usually show keys. Keys can be represented with class attribute names followed by the {key} notation.

4. The main differences will most likely be in the notation used to describe the model. Students may find systems analysts using Booch, OMT, or their own custom notation. Students should see if the notations in use by systems analysis can represent use cases and class diagrams as shown in the chapter.
Chapter 16 Using Relational Databases to Provide Object Persistence

Chapter Overview

Please note that the material for this chapter is based upon the Web version of Chapter 16, not the abbreviated version of Chapter 16 in the text.

The purpose of this chapter is to introduce the integration between object-oriented application development models and relational databases. This chapter presents background to understand the broader context of object-oriented thinking and how it is materializing in the current industry marketplace in relation to databases. The chapter presents the conceptual differences between object-oriented and relational approaches to modeling, general characteristics of different mechanisms to bridge the so-called “object-relational mismatch,” and a comprehensive example of Hibernate (a widely-used object-relational mapping technology).

Chapter Objectives

Specific student learning objectives are included in the beginning of the chapter. From an instructor’s point of view, the objectives of this chapter are to:

1. Provide background understanding for the “object-relational mismatch” situation found in industry.
2. Explain the different approaches to providing persistence for objects using relational databases. Review decision criteria for selecting an approach for the business situation.
3. Provide a practical example of object-relational mapping with a widely-used object relational mapping technology (Hibernate) from a conceptual standpoint.
4. Introduce the Hibernate query language (HQL) and provide fundamental practice in developing HQL query statements.

Key Terms

<table>
<thead>
<tr>
<th>Accessor method</th>
<th>N+1 selects problem</th>
<th>Pooling of database connections</th>
</tr>
</thead>
<tbody>
<tr>
<td>Call-level application</td>
<td>Object identity</td>
<td>Separation of concerns program interface</td>
</tr>
<tr>
<td>Declarative mapping schema</td>
<td>Object-relational mapping</td>
<td>Serialization</td>
</tr>
<tr>
<td>Entity class</td>
<td>Object-relational mismatch</td>
<td>Transparent persistence</td>
</tr>
<tr>
<td>Fetching strategy</td>
<td>Persistence</td>
<td>Value type</td>
</tr>
</tbody>
</table>
Classroom Ideas

1. Use Figure 16-1 to illustrate the difference in how an Address is represented in an object-oriented solution versus a relational solution.
2. Present the object-oriented concept of association and show the o-o code example of how these are implemented. Use Figure 16-2 to walk through the use of the accessor method.
3. Break the class up into small teams. Ask each small team to briefly research one of the options for providing persistence for objects: call-level application interfaces, SQL query mapping frameworks, and object-relational mapping frameworks. Ask students to find examples of these approaches being used in practice and to report back to the class as a whole with these examples and their research.
4. Complete Problems and Exercises 1 and 2, or 3 and 4, or 5 and 6 in a class session. Review answers together as a class.

Answers to Review Questions

1. Define each of the following key terms:

   a. **Object-relational mismatch**  The conceptual differences between the object-oriented approach to application design and the relational model for database design and implementation.
   b. **Object-relational mapping**  Defining structural relationships between object-oriented and relational representations of data, typically to enable the use of a relational database to provide persistence for objects.
   c. **Persistence**  An object’s capability to maintain its state between application execution sessions.
   d. **Call-level application program interface**  A mechanism that provides an application program with access to an external service, such as a database management system.
   e. **Transparent persistence**  A persistence solution that hides the underlying storage technology.
   f. **JDBC**  Java database connectivity; an industry standard call-level application program interface that enables Java programs to connect with SQL databases.
   g. **iBATIS**  One of the best-known industry tools that provides linking of classes in an object-oriented solution to parameters and results of SQL queries; an example of an SQL Query Mapping Framework that provides object persistence with relational databases.
   h. **Hibernate**  A comprehensive object-relational mapping (ORM) framework that has the longest history of all ORM frameworks; this framework hides the relational data access methods from the object-oriented applications.
   i. **N+1 selects problem**  A performance problem caused by too many SELECT statements generated by an ORM framework.
2. Compare and contrast the following terms:

a. **Object identity; primary key value of a row in a database table**  An object identity is the property of an object separating it from other objects based on its existence, not on the values of any of the attributes of the object. The primary key value of a row in a database table serves to provide the identity of row in the database table.

b. **Entity class; value type**  An entity class represents a real-world entity; a value type is a class specification for a value associated with entity instances.

c. **OODBMS; RDBMS**  OODBMS stands for an object-oriented database management system and provides direct, transparent persistence for objects in o-o applications. RDBMS stands for relational database management systems that store data according to relational theory and does not provide direct, transparent persistence for objects in o-o applications.

d. **Many-to-one association; one-to-many association**  These are both common binary relationships between entities/classes in the relational and o-o design worlds. In the use of ORM, these are mapped differently based on the side from which one is observing the association.

e. **Lazy loading; eager loading**  Both of these are fetching strategies for an ORM framework’s retrieval of persistent objects to run-time memory during navigation processes. A lazy loading strategy retrieves objects from the database only when needed (Hibernate’s default). An eager loading strategy retrieves all associated objects from the database when the object to which they are linked is retrieved.

3. Reasons why OODBMS did not become popular:

A large reason is organizational inertia, influenced by several factors:

- **Investment in relational database management systems**  The amount of money spent since the 1980s for RDBMS is not trivial. Moving to a new DBMS is more difficult (and likely more costly) than moving to a new application development environment.

- **Weak query capabilities in OODBMS**  The query features of OODBMS did not live up to the query features available in RDBMS.

- **OODBMS developers were small market players**  Smaller market players did not have the financial backing to convince user organizations of the scalability and reliability of OODBMS versus RDBMS. User organizations did not find that the OODBMS benefits to outweigh the costs of conversion.

4. Factors contributing to the object-relational mismatch (refer to Table 16-1 and text discussion):

- nature and granularity of data types
- structural relationships (inheritance structures, representation of associations)
- defining the identity of objects/entity instances
- methods of accessing persistent data
- focus on data (relational databases) versus integrated data and behavior (the o-o approach)
- architectural styles
support for managing transactions

5. How do the object-oriented and relational approaches to accessing data differ?

In the o-o world, a typical way to access a data item is to call the accessor method associated with a specific attribute. The navigation to the specific attribute is often accessed through a public accessor method within another object due to encapsulation of the attribute. In practice, the o-o programming language will use some way to iterate over a collection of objects to retrieve the data. In a relational database, a data item is accessed via queries; often, this means that the entire set of values can be specified in advance and retrieved at the same time.

6. Specify the key difference in how entity instance identities are defined in the o-o and relational worlds.

In the relational world, every row in a relational table has a unique primary key value that determines the identity of the row. In the o-o world, each object has its own identity based on its existence (fundamentally, location in memory) and is not dependent on the values of any of the attributes of the object.

7. Why is it essential that relational databases can be effectively used to provide persistence for objects in applications developed using the o-o paradigm?

The o-o paradigm has reached dominance in the application development marketplace. At the same time, tools that provide long-term persistence for organizational data are typically relational databases. Thus, the common organizational usage of relational databases must work with the newly developed o-o applications in order for organizational information systems to meet the needs of the organization.

8. Why can the link between o-o applications and relational databases not be built simply by using JDBC or some other call-level application program interface?

Call-level application program interfaces, such as JDBC, have the advantages of low overhead and the highest level of control over the database connection. However, call-level APIs have significant weaknesses as a solution: they expose the database structure to application developers; require that the developers understand the underlying database and SQL; create a lot of code that cannot usually be re-used and is hard to maintain; and they violate the idea of separation of concerns (e.g., persistence). Also, the use of call-level APIs is very labor-intensive that is prone to errors, especially in large applications.

9. Explain the main conceptual difference between iBATIS and Hibernate.

iBATIS is an example of a SQL Query Mapping Framework tool. Tools such as iBATIS do not create a conceptual connection between classes (in o-o world) and tables (in relational database). Hibernate is an example of an Object Relational Mapping (ORM) Framework tool. Tools such as Hibernate permit the conceptual mapping of classes and associations in
the o-o world to the tables and relationships in the relational database, and this mapping generally occurs only once.

10. Explain the criteria you might use to select between iBATIS and Hibernate

SQL query mapping frameworks such as iBATIS are particularly strong solutions when there is a complex, potentially nonstandard existing structure and the task requires the execution of sophisticated queries resulting in a large number of rows. Object-relational mapping framework tools such as Hibernate are strong solutions when you have an opportunity to create a new database schema to provide persistence to your objects and the required database operations are not hugely complex. Thus, two criteria for deciding between the use of iBATIS or Hibernate might be (1) how complex are the required query/database operations; and (2) is the database structure/schema new or old (legacy).

11. Why is transparent persistence so important from the perspective of application developers?

Persistence is an object’s capability to maintain its state between application execution sessions. Transparent persistence is a persistence solution that hides the underlying storage technology from the object-oriented applications. Thus, for an o-o application developer, an approach for addressing the object-relational mismatch that provides transparent persistence enables the developer to concentrate on the application, and not so much on the underlying database technology. In principle, a developer who is using an ORM framework that provides transparent persistence does not have to write SQL queries or have an in-depth understanding of the underlying database structure.

12. Some developers are concerned about the overhead that SQL query mapping frameworks and ORM frameworks add to call-level APIs. Why?

In some situations the overhead from SQL query mapping and ORM frameworks may impose a performance penalty to applications that use them.

13. What is the relationship between Hibernate and JPA?

JPA is the Java Persistence API, which is part of the Enterprise Java Bean 3.0 standard. JPA is a subset of Hibernate, which is the de facto industry standard for ORM frameworks.

14. What is the purpose of the <Class name>.hbm.xml files in Hibernate?

The <Class name>.hbm.xml files are the XML mapping files that defines the relationship between the object-oriented classes and relational tables.

15. How is Hibernate configured for a specific DBMS environment?

The characteristics of which DBMS, database, and database connection specifics are included in a configuration file named hibernate.cfg.xml. This XML file includes information about the driver to be used, the URL for the database connection string, and
the username and password to connect to the database. Additionally, the XML configuration file includes a listing of the <Class name>.hbm.xml files that define the mapping and the parameters necessary to specify the pooling of database connections.

16. How are attributes specified in the Hibernate configuration files?

Attributes are specified in the mapping resource files (e.g., the <Class name>.hbm.xml files).

17. What is the purpose of the <set> tag in the Hibernate configuration files?

The <set> tag identifies the mapping of 1:M or M:1 relationships among the o-o classes.

18. Explain how primary keys of the database tables are specified within the Hibernate environment

Primary keys of the database tables are specified within an <id> tag element in the XML mapping file.

19. What is the purpose of the SchemaExport tool in Hibernate?

The SchemaExport tool in Hibernate will produce SQL Data Definition Language scripts for creating a relational database schema described in a specific set of mapping files.

20. Referring to the SQL code in Figure 16-7, explain why Student, Faculty, and Registration do not have autogenerated primary keys.

Student and Faculty get their primary keys from Person. The primary key of Registration is a composite of the primary keys from Section and Student.

21. Explain the importance of pooling database connections.

Pooling of database connections refers to the process of using a limited number of database connections that are shared by multiple applications and users. Hibernate permits specification of parameters that enable pooling of database connections to occur. By pooling database connections, a new connection to the DBMS does not have to be made each time an application wishes to interact with the database, thus saving costs.

22. Briefly describe the four different ways in which an inheritance structure can be mapped to a relational schema.

a. **Table per subclass** Requires a table for each class and subclass.
b. **Table per concrete class with implicit polymorphism** Attributes from superclass are included in all tables representing the subclasses (no table for superclass).
23. Explain why it makes sense to differentiate between many-to-one and one-to-many associations in the object-oriented world.

Differentiation is based on the side from which you are observing the directional association. Because navigation to the objects occurs through the objects in the o-o world, it makes sense to know whether you were attempting to access data from the M:1 or 1:M perspective. The o-o programming language can then choose appropriate iterations or routines based on knowing the direction of the association.

24. What is the practical impact of specifying an association as composition from the perspective of object-relational mapping?

Composition means that one side of the association has been specified as the “whole,” and this manages the life cycle of its “parts” to the extent that the “parts” cannot exist without the “whole.” In practice, this means that the foreign key attribute in the relational table that is mapped to an object must be defined as NOT NULL.

25. Explain the importance of well-designed fetching strategies.

Fetching strategies define when and how the ORM framework retrieves persistent objects to run-time memory during a navigation process. Well-designed fetching strategies allow an o-o application to run with minimal performance issues when accessing databases with an ORM framework approach.

26. When is the SELECT keyword necessary in HQL?

The SELECT keyword is necessary when you want to view a collection of attributes instead of whole objects in the source class.

27. What is an implicit association join?

An implicit association join is when you do not have to specify how the tables need to be linked together in the HQL. The implicit association join draws this information from the XML mapping files in Hibernate, and allows you to use object-oriented navigation in the queries.
28. Analyze the queries that include explicit joins in the HQL queries and their SQL counterparts. What is the main difference between these two query types?

The main difference is that the HQL query does not need to specify how the linking occurs whereas in the SQL query the specifics of the foreign key to primary key link is detailed in the SQL syntax.

Answers to Problems and Exercises

1. Set of <class name>.hbm.xml files for Figure 5-5

PVFC.Customer.hbm.xml

```xml
<class name="pvfc.Customer" table="Customer">
  <id name="id" type = "long" column = "Customer_ID">
    <generator class="native"/>
  </id>
  <property name="customerName" type="string" column="Customer_Name"/>
  <property name="customerAddress" type="string" column="Customer_Address"/>
  <property name="city" type="string" column="City"/>
  <property name="state" type="string" column="State"/>
  <property name="postalcode" type="string" column="Postal_Code"/>
  <set name = "orders" inverse = "true" table="Order">
    <key column = "Customer_ID"/>
    <one-to-many class="pvfc.Order"/>
  </set>
</class>
```

PVFC.Product.hbm.xml

```xml
<class name="pvfc.Product" table="Product">
  <id name = "id" type = "long" column = "productID">
    <generator class="native"/>
  </id>
  <property name="productDesc" type = "string" column="Product_Description"/>
  <property name="productFinish" type = "string" column="Product_Finish"/>
  <property name="standardPrice" type = "float" column="Standard_Price"/>
  <property name="productLineID" type = "long" column="Product_Line_ID"/>
</class>
```
PVFC.Order.hbm.xml

```xml
<class name="pvfc.Order" table="Order">
  <id name="id" type = "long" column = "Order_ID">
    <generator class="native"/>
  </id>
  <property name ="orderDate" type = "date" column="Order_Date"/>
  <many-to-one name = "customer"
    class = "pvfc.Customer"
    column = "Customer_ID"
    not-null = "true"/>
  <set name = "orderItems" table="OrderLine">
    <key column = "Order_ID"/>
    <composite-element class = "pvfc.OrderLine">
      <parent name = "Order"/>
      <many-to-one name="product" column="Product_ID"
        class = "pvfc.Product" not-null = "true"/>
      <property name = "orderedQty" type = "integer" column = "Ordered_Quantity"/>
    </composite-element>
  </set>
</class>
```
2. Domain model corresponding to xml files created in Problems and Exercises 1
3. Domain model for EER model in Figure 5-33

![Domain model diagram]

4. Relational schema corresponding to domain model in Problems and Exercises 3

<table>
<thead>
<tr>
<th>Table</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customer</td>
<td>(Customer_ID, Cust_Name, Cust_Address)</td>
</tr>
<tr>
<td>CardAccount</td>
<td>(Account_ID, Exp_Date, Card_Type, Customer_ID)</td>
</tr>
<tr>
<td>debitCard</td>
<td>(DCaccount_ID, Bank_No)</td>
</tr>
<tr>
<td>creditCard</td>
<td>(CCaccount_ID, Cur_Bal)</td>
</tr>
<tr>
<td>Charges</td>
<td>(Charge_ID, Charge_Date, Charge_Time, Amount, CCaccount_ID, Merch_ID)</td>
</tr>
<tr>
<td>Merchant</td>
<td>(Merch_ID, Merch_Addr)</td>
</tr>
</tbody>
</table>

Note: PK are shown in bold, underline; FK shown in italics.
4. <class name>.hbm.xml files corresponding to domain model in Problems and Exercises 3

cards.Customer.hbm.xml

```xml
<class name="cards.Customer" table="Customer">
  <id name = "id" type = "long" column = "Customer_ID">
    <generator class="native"/>
  </id>
  <property name="custName" type = "string" column="Cust_Name"/>
  <property name="custAddress" type = "string" column="Cust_Address"/>
  <set name = "accounts" inverse = "true" table="CardAccount">
    <key column = "Customer_ID"/>
    <one-to-many class="cards.Account"/>
  </set>
</class>
```

cards.Account.hbm.xml

```xml
<class name="cards.Account" table="CardAccount">
  <id name = "id" type = "long" column = "accountID">
    <generator class="native"/>
  </id>
  <property name="expDate" type = "date" column="Exp_Date"/>
  <property name="cardType" type = "string" column="Card_Type"/>
  <joined-subclass name ="cards.Debit" table="DebitCard">
    <key column="DCaccount_ID"/>
    <property name="bankNo" type = "string" column="Bank_No"/>
  </joined-subclass>
  <joined-subclass name ="cards.Credit" table="CreditCard">
    <key column="CCaccount_ID"/>
    <property name="curBal" type = "float" column="Cur_Bal"/>
    <set name = "charges" table="Charges">
      <key column = "CCaccount_ID"/>
      <composite-element class = "cards.Charges">
        <parent name = "CreditCard"/>
        <many-to-one name = "merch" column = "Merch_ID" class = "cards.Merchant" not-null = "true"/>
        <property name = "chargeDate" type = "date" column = "Charge_Date"/>
        <property name = "chargeTime" type = "time" column = "Charge_Time"/>
        <property name = "amount" type = "float" column = "Amount"/>
      </composite-element>
    </set>
  </joined-subclass>
</class>
```
cards.Merchant.hbm.xml

```xml
<class name="cards.Merchant" table="Merchant">
  <id name="id" type="string" column="Merch_ID">
    <generator class="native"/>
  </id>
  <property name="merchAddr" type="string" column="Merch_Addr"/>
</class>
```
5. Domain model corresponding to EER in Figure 5-36
6. Relational schema corresponding to domain model in Problems and Exercises 5

Note: PK are shown in bold, underline; FK shown in italics

Menu (Menu_ID, Menu_Type, Menu_Description)
Menu_Dish (Menu_ID, Dish_ID)
Dish (Dish_ID, Dish_Name, Prep_Time)
Dish_Ingredient (Dish_ID, Ingredient_ID)
Ingredient (Ingredient_ID, Ingredient_Name)
Event (Event_ID, Event_Date, Event_Location, Event_Time, Menu_ID)
Work_Schedule (Event_ID, Emp_ID, Position, Start_Time, End_Time)
Staff (Emp_ID, Name, Salary, Supervisor_ID)
Expertise (Emp_ID, Skill_ID)
Skill (Skill_ID, Skill_Name)

<Menu.hbm.xml corresponding to domain model in Problems and Exercises 5>

<Menu>
  <id name="id" type="long" column="Menu_ID">
    <generator class="native"/>
  </id>
  <property name="menuDesc" type="string" column="Menu_Description"/>
  <property name="menuType" type="string" column="Menu_Type"/>
  <set name="events" inverse="true" table="Event">
    <key column="Event_ID"/>
    <one-to-many class="dining.Event"/>
  </set>
  <set name="menuDish" table="Menu_Dish">
    <key column="Menu_ID"/>
    <composite-element class="dining.MenuDish">
      <parent name="Menu"/>
      <many-to-one name="dish" column="Dish_ID" class="dining.Dish" not-null="true"/>
    </composite-element>
  </set>
</Menu>
Dish.hbm.xml

```xml
<class name="dining.Dish" table="Dish">
  <id name = "id" type = "long" column = "Dish_ID">
    <generator class="native"/>
  </id>
  <property name="dishName" type = "string" column="Dish_Name"/>
  <property name="prepTime" type = "float" column="Prep_Time"/>

  <set name = "dishIngredient" table="Dish_Ingredient">
    <key column = "Dish_ID"/>
    <composite-element class = "dining.DishIngredient">
      <parent name = "Dish"/>
      <many-to-one name = "ingredient" column = "Ingredient_ID"
        class = "dining.Ingredient" not-null = "true"/>
    </composite-element>
  </set>
</class>
```

Ingredient.hbm.xml

```xml
<class name="dining.Ingredient" table="Ingredient">
  <id name = "id" type = "long" column = "Ingredient_ID">
    <generator class="native"/>
  </id>
  <property name="ingredientName" type = "string" column="Ingredient_Name"/>
</class>
```
Event.hbm.xml

```xml
<class name="dining.Event" table="Event">
  <id name="id" type="long" column="Event_ID">
    <generator class="native"/>
  </id>
  <property name="eventDate" type="date" column="Event_Date"/>
  <property name="eventTime" type="time" column="Event_Time"/>
  <property name="eventLocation" type="string" column="Event_Location"/>
  <many-to-one name="menu" class="dining.Menu" column="Menu_ID"/>
  <set name="workSchedule" table="Work_Schedule">
    <key column="Event_ID"/>
    <composite-element class="dining.WorkSchedule">
      <parent name="Event"/>
      <many-to-one name="staff" column="Emp_ID" class="dining.Staff" not-null="true"/>
      <property name="startTime" type="time" column="Start_Time"/>
      <property name="endTime" type="time" column="End_Time"/>
      <property name="position" type="string" column="Position"/>
    </composite-element>
  </set>
</class>
```

Staff.hbm.xml

```xml
<class name="dining.Staff" table="Staff">
  <id name="id" type="long" column="Emp_ID">
    <generator class="native"/>
  </id>
  <property name="name" type="string" column="Name"/>
  <property name="salary" type="string" column="Salary"/>
  <many-to-one name="supervisor" class="dining.Staff" column="Supervisor"/>
  <set name="expertise" table="Expertise">
    <key column="Emp_ID"/>
    <composite-element class="dining.Expertise">
      <parent name="Staff"/>
      <many-to-one name="skill" column="Skill_ID" class="dining.Skill" not-null="true"/>
    </composite-element>
  </set>
</class>
```
7. HQL for finding the titles and numbers of all 400-level IS courses

```
select c.courseTitle, c.courseNbr
from Course c
where substring(c.courseNbr,1,4) = 'IS 4'
```

8. HQL for finding the titles and numbers of all 400-level IS courses offered in spring 2008

```
select c.courseNbr, c.courseTitle
from Section s
join s.course c
where substring(c.courseNbr,1,4) = 'IS 4' and
  s.semester = 'Fall 2007'
```

9. HQL for finding the names of all students taking at least one course with a faculty member

```
select distinct f.lastName, f.firstName, st.lastName, st.firstName
from Section s
join s.course c
join s.faculty f
join s.enrolledStudents reg
join reg.student st
order by f.lastName, f.firstName
```

10. HQL for finding the names and matriculation year for all students enrolled in IS240 for spring 2008

```
select st.lastName, st.firstName, st.yearMatriculated
from Section s
join s.course c
join s.enrolledStudents reg
join reg.student st
where c.courseNbr='IS 240' and s.semester = 'Spring 2008'
```
11. HQL for finding the average grade earned in IS 350 by students who matriculated in 2006, regardless of when they took the course

```sql
select avg(reg.numGrade)
from Section s
join s.course c
join s.enrolledStudents reg
join reg.student st
where c.courseNbr='IS 350' and st.yearMatriculated=2006
```

12. HQL for finding the names and office locations for faculty who taught IS 350 in Fall 2007

```sql
select f.lastName, f.firstName, f.office
from Section s
join s.course c
join s.faculty f
where c.courseNbr = 'IS 350' and
s.semester = 'Fall 2007'
```

13. HQL for finding the names and office locations for faculty who taught more than one course in Fall 2007

```sql
select f.lastName, f.firstName, f.office
from Section s
join s.course c
join s.faculty f
where s.semester='Fall 2007'
group by f.facultyID
having count(s.sectionNbr)>1
```

**Answers to Field Exercises**

1. to 3. Students are encouraged to use the interviewing tips mentioned in earlier textbook chapters to complete these exercises.
Appendix A E-R Modeling Tools and Notation

Appendix Overview

The purpose of this appendix is to introduce one of the most popular methods of modeling relational database systems. There are a variety of software tools available to assist in the modeling of these types of systems. This appendix will assist in comparing the notation used in the text with the notation used in four commonly used packages: Computer Associates’ AllFusion ERWin Data Modeler 4.1 SP1, Oracle Designer 10g, Sybase Powerdesigner 11.1, and Visio Pro 2003.

Appendix Objectives

*From an instructor’s point of view, the objectives of this appendix are to:*

1. Expose students to the look and feel of different modeling tools.
2. Illustrate the differences in notation of various modeling tools.
3. Review the entity symbols and relationships available with different modeling tools.
4. Discuss/review relationship and cardinality types:
   a. 1:1
   b. 1:M
   c. M:N
   d. Optional participation
   e. Mandatory participation
5. Discuss and review types:
   a. Supertype
   b. Subtype
6. Discuss and review attributes:
7. a. Key and nonkey attributes
   b. Null and not null definitions
8. Discuss and review keys:
   a. Candidate keys
   b. Primary keys
   c. Foreign keys
   d. Secondary keys

Classroom Ideas

1. Many of the products shown in this appendix have limited time trial versions. Visit the product Web sites to find the trial versions, which can then be used in class for illustrations or by students to explore the capabilities of these tools.
2. Be sure to review the definitions of entities; entity types; entity instances; relationship structures; cardinality and participation constraints; primary, secondary, and foreign keys; and ways of modeling business rules and constraints.
3. Students always learn best by seeing examples. Provide numerous examples using familiar situations: students, courses, and instructors; customers, orders, and
products; waiters, meals, and bills. Show each situation with several different data modeling tools. If there are capabilities one product has that another does not have, point out these differences.

4. Emphasize the importance of this model as a tool, both to help the designer understand the system, and to show the users that the system is all-encompassing.

5. Divide the class into teams and have each team take the Pine Valley model for one tool and develop comments/observations about the data model produced using that tool.

6. Divide the class into teams and have each team develop a data model for a case study using different tools. Have the teams then compare their results and briefly discuss the strengths and weaknesses of each tool used.

7. You may have other tools that you want students to use other than those illustrated in this appendix or students may know of other tools available for data modeling. Use this appendix as a “jumping-off” point for an exploration session of available tools and a discussion of how students might evaluate such options for an employer looking for a recommendation on what package to use in the “real world.”

8. This appendix is a good opportunity to prepare students for the “real world,” By pointing out the differences in the tools they may have at their disposal at work, the instructor can demonstrate the importance of understanding general database concepts and developing flexibility in how they will be able to accomplish the tasks they are assigned at work.
Appendix B Advanced Normal Forms

Appendix Overview

The purpose of this appendix is to introduce normalized forms beyond third normal form (3NF). 3NF is adequate for most practical applications, but some exposure to higher forms is advantageous. When anomalies in the 3NF model are encountered, students need to know how to approach and solve these issues. The higher forms covered in this appendix are Boyce-Codd Normal Form (BCNF), a stronger 3NF and fourth normal form (4NF). Interested readers are pointed to additional sources for background on Domain-key Normal Form (DKNF) and fifth normal form (5NF).

Appendix Objectives

From an instructor’s point of view, the objectives of this appendix are to:
1. Expose students to the definitions of higher forms.
2. Explain what circumstances require the use of a higher normal form.
3. Illustrate these circumstances with concrete examples.
4. Define what a multivalued dependency is.

Key Terms

| Boyce-Codd normal form (BCNF) | Fourth normal form (4NF) | Multivalued dependency |

Classroom Ideas

1. Be sure to review the definitions of anomalies and why they cause problems in the relational model.
2. Define/review all the terms used in normalization: primary, secondary, and foreign keys; full functional dependency, transitive dependency, partial-key dependencies, and multivalued dependencies.
3. Discuss and review the first three normal forms for context.
4. Emphasize the importance of testing proposed databases for anomalies in 3NF, i.e., populating tables to identify problems in the data set. Use a variety of record instances.
5. Students learn best by example. Do several examples in class using familiar scenarios: students, courses, and instructors; students, advisors, and majors; students, food, and drink preferences.
6. Separate the class into groups. Have each group try to come up with an example of all or one assigned advanced normal form. Have each group present the problem, define the anomaly, and present and defend the solution.
Appendix C Data Structures

Appendix Overview

The purpose of this appendix is to discuss basic data structures. Students who have had a data structures course or covered data structures in a programming course will find this appendix to be a good review. Those who have not had any exposure to data structures can use this appendix to gain a basic understanding of data structures. The appendix begins with a discussion of pointers and the three types of pointers that are generally used. Next, the building blocks of data structures are discussed. A discussion of linear data structures is followed by a discussion of trees.

Appendix Objectives

From an instructor’s point of view, the objectives of this appendix are to:
1. Review pointers and the three different types of pointers.
2. Discuss the building blocks of data structures.
3. Discuss/review linear data structures:
   a. stacks
   b. queues
   c. sorted lists
   d. multilists
4. Discuss the hazards of chain structures.
5. Discuss trees, particularly balanced trees.

Classroom Ideas
1. If your students have had exposure to data structures in another course (possibly a programming course), review pointers, particularly in light of their uses within a database management system.
2. Students always learn best by examples. Provide several examples of pointers in a language that students are familiar with, such as C++ or C.
3. Discuss the differences between the three types of pointers, using Figure C-1. Discuss the various characteristics described in Table C-1.
4. Emphasize that all data structures are built around building blocks. Contrast address sequential with pointer sequential connection methods. Compare data direct placement with data indirect placement. Once again, showing students with actual examples will reinforce the concepts.
5. Review and/or discuss stacks, queues, sorted lists, and linked lists. Show some examples in a language familiar to the students.
6. If your students have had a course in data structures, for a review divide the class into teams. Have each team design a small database using one type of data structure. You might include linked lists, double linked lists, sorted lists, and stacks. The students should write code for basic functionality, such as the ability to add a record, delete a record, lookup a record and modify a record. Have each team present their mini-projects and then discuss the merits of each approach.
7. Discuss problems that can occur when using chain structures.
8. Discuss trees and how they can be used for database management systems.
9. Have the students discuss in class (or write a brief paper on) why it is important for a database administrator to know about data structures. Ask them for what tasks performed by a DBA would it be helpful to know about the data structures shown in this appendix. Ask them specifically about whether issues of database security, integrity, and recoverability relate to this appendix.