Chapter 9 – Designing the Database

Table of Contents

Chapter Overview

Database management systems provide designers, programmers, and end users with sophisticated capabilities to store, retrieve, and manage data. Sharing and managing the vast amounts of data needed by a modern organization would not be possible without a database management system.

In Chapter 4, students learned to construct conceptual data models and to develop entity-relationship diagrams (ERDs) for traditional analysis and domain model class diagrams for object-oriented (OO) analysis. To implement an information system, developers must transform a conceptual data model into a more detailed database model and implement that model in a database management system.

In the first sections of this chapter students learn about relational database management systems, and how to convert a data model into a relational database schema. The database sections conclude with a discussion of database architectural issues such as single server databases versus distributed databases which are deployed across multiple servers and multiple sites.

Many system interfaces are electronic transmissions or paper outputs to external agents. Therefore, system developers need to design and implement integrity controls and security controls to protect the system and its data. This chapter discusses techniques to provide the integrity controls to reduce errors, fraud, and misuse of system components. The last section of the chapter discusses security controls and explains the basic concepts of data protection, digital certificates, and secure transactions.

Learning Objectives

After reading this chapter, the student should be able to:

- Explain the responsibilities of the data administrator and database administrator
- Design a relational database schema based on a class diagram

- Evaluate and improve the quality of a database schema
- Describe the different methods for configuring distributed databases
- Explain the importance of and methods for protecting the database

Notes on Opening Case and EOC Cases

Opening Case

Downslope Ski Company: Designing a Secure Supplier System Interface: This case discusses a very real situation that is faced by many companies in today's interconnected world. Downslope Ski Company is a manufacturer of skis and snowboards. To streamline its production process and to reduce raw material costs, it uses a just-in-time production approach. The case highlights the difficulty of responding to many different and varied stakeholders. Each stakeholder has different information needs and the solution must be integrated to meet all these needs. In addition, the data inputs and outputs will range from Web-based systems to automated equipment feeds. These diverse requirements must be integrated into the solution.

EOC Cases

Computer Publishing, Inc.: Writing and publishing a textbook is a complex process. Updating and maintaining a textbook is also complex. Computer Publishing, Inc. is looking for a way to automate the writing, editing, updating, and publishing process. In order to do so, a sophisticated data model of a textbook and all of its association is required. Students are asked to develop this data model and discuss the appropriateness of the relational data model for storing all of this information (including images, test banks, etc.) Students are also asked to discuss the issues relating to maintaining the security of this information since outside contractors will need access to it.

Community Board of Realtors (running case)**:** Community Board of Realtors is a professional organization that supports real estate offices and agents. Students are asked to use the domain model that they developed in Chapter 4 and develop the database schema, including keys, foreign keys, and all table attributes. They are also required to verify that their solution is in third normal form.

Spring Breaks 'R' Us Travel Services (SBRU) (running case)**:** SBRU is an online travel services that books spring break trips to resorts for college students. The students are asked to take the data model developed in Chapter 4 and create a relational database schema. Then they are to verify that it is in first-, second-, and third-normal form.

On the Spot Courier Services (running case)**:** On the Spot is a small, but growing, courier service that needs to track customers, package pickups, package deliveries, and delivery routes. In Chapter 4 students developed and domain model, which they have updated in various later chapters. Students are asked to update the domain model as much as possible with the information from prior chapters and then develop a relational database schema. They are asked to verify that the tables are in third normal form. Students are also asked to discuss which employee(s) could function as the DA and the DBA.

Sandia Medical Devices (running case)**:** Sandia Medical Devices is a company that specializes in medical monitoring through remote, mobile telecommunication devices. This case is divided into two

parts for this chapter.

Part 1. Based on the domain model students created in earlier chapters (3, 4, and 8), and on the additional definition provided in the description, students are required to develop a set of relational database tables and to verify that they are in third normal form.

Part 2. Based on the results from Chapter 6, and additional information provided about cell phone access, students are asked to discuss issues and to modify answers given in Chapter 6 to address security and controls issues.

Instructor's Notes

Databases and Database Management Systems

Key Terms

- **database (DB) –** an integrated collection of stored data that is centrally managed and controlled
- **database management system (DBMS) –** a system software component that manages and controls one or more databases
- **schema –** database component that contains descriptive information about the data stored in the physical data store
- **Structured Query Language SQL** – a query language used to access and update the data in a relational database

Lecture Notes

A **database** is managed and controlled by a **database management system (DBMS)**. A DBMS is a system software component that is generally purchased and installed separately from other system software components (e.g., operating systems). The **schema** contains descriptive information about the data stored in the physical data store, including:

- Organization of individual stored data items into higher level groups, such as tables
- Associations among tables or classes (e.g., pointers from customer objects to related sale objects)
- Details of individual data items, including types, lengths, locations, and indexing of data items
- Access and content controls, including allowable values for specific data items, value dependencies among multiple data items, and lists of users allowed to read or update data items

A DBMS has four key components: an application program interface (API), a query interface, an administrative interface, and an underlying set of data access programs and subroutines. **Structured Query Language (SQL)**, as the query language used to access the database data.

Databases and database management systems provide several important data access and management capabilities, including:

- Simultaneous access by many users and application programs
- Access to data without writing application programs (i.e., via a query language)
- Application of uniform and consistent access and content controls
- Integration of data stored on multiple servers distributed across multiple locations

Quick Quiz

Q: What is a database schema and what information is kept in a schema?

> A: A schema is the information about the structure of the database, including the tables, the attributes, the keys, etc.

Q: What are the four key components of every DBMS?

> A: An API, a query interface, an administrative interface, and the underlying programs that read/write to the database.

Q: What does SQL stand for and how is it used?

> A: SQL stands for Structured Query Language. It is the standard language used to access and update the data in a relational DBMS. All RDBMS's utilize SQL.

Database Design and Administration

Key Terms

data administrator (DA) – The person in charge of the structure and integrity of the data

database administrator (DBA) – The person in charge of the safety and operation of the DBMS

Lecture Notes

Before jumping into the actual details of how to design the database, we first address three important issues:

- How does database design integrate into the organization's overall technological environment?
- How does database design integrate into the overall project plan?
- Who is involved in database design?

Technology Environment

Data and the DBMS Environment In most organizations, information systems and databases are deployed in a piecemeal fashion, with new systems purchased or developed to meet specific business requirements, which results in many independent systems with diverse data and DBMS configurations in one company. One of the most important assets of almost any organization is its data. In addition to

replacing or upgrading existing data in the database itself, it is also common that the previous system also fed data to other information systems in the organization. The result is that there are frequently many DBMS's that the project team must be aware of and coordinate with.

Hardware and Network Architecture: The other big issue is the hardware and network configuration in the organization. An important issue is how the database architecture of the new system will interact with the architecture that supports existing systems. More specific questions include whether the new system will use existing DBMSs and servers and whether the existing servers and the network have sufficient unused capacity to support the new system.

Databases can be deployed as the following:

- A single desktop system residing on one computer and used only by one or two users.
- A shared database that resides on a database server for a LAN.
- A larger database requiring multiple servers, but that is contained within a single server farm or data center.
- Finally, when the organization becomes global with many users worldwide (or in multiple locations in a single country), it is often required to have multiple data centers or server farms.

Project Plan and Schedule

Choosing the correct timing for database design and construction within a system development project isn't a simple matter. The basic issue is whether the database design can proceed in iterations or if it is better to complete the database early in the project. Normally in iterative development, the higher-risk elements of a subsystem are developed first. Because the database is the foundation of any information system, it is common to define the domain model and perform the database design in the first few iterations of subsystem development. Unfortunately, even this technique may not be sufficient for a complex system consisting of multiple subsystems with interlocking database requirements.

Database Design Team

Two key positions on the permanent database staff are the data administrator and the database administrator. A **data administrator (DA)** has responsibility for the structure and integrity of the data itself. Specifically, the DA manages important aspects of data definition and database design, including the following:

- Data standards. Naming standards, definition standards, data typing standards, and value edits
- Data use. Ownership of data, accessibility of data, and confidentiality
- Data quality. Validation rules, completeness, currency, consistency, and relevancy

A **database administrator (DBA)** maintains the database after it has been deployed and manages the safety and operation of the database. It is his or her job to ensure that the database is configured correctly for the organization's architecture and performs effectively and efficiently. The DBA's responsibilities include the following:

- Managing a multiple DBMS environment

- Protecting the data and database, including user authentication and attack prevention
- Monitoring and maintaining high levels of performance
- Backing up the database and defining recovery procedures

Quick Quiz

Q: Identify four possible database configurations based on the size of the organization.

A: Single desktop, LAN configuration, single server farm, global database across multiple farms

Q: What are the duties of a data administrator?

A: Define and maintain structure and integrity of the data such as data standards, data quality, and how the data can be used.

Q: What are the duties of a database administrator?

A: Maintain the safety and integrity of the database and the DBMS, including manage the DBMS environment, protect the database, monitor performance and tune the database, ensure it is backed up.

Relational Databases

Key Terms

- **relational database management system (RDBMS) –** a DBMS that organizes data in tables or relations
- **table –** a two-dimensional data structure of columns and rows
- **row –** one horizontal group of data attribute values in a table
- **attribute –** one vertical group of data attribute values in a table
- **attribute value –** the value held in a single table cell
- **key –** an attribute or set of attributes, the values of which occur only once in all the rows of the table
- **candidate key** – an attribute or set of attributes that are unique identifiers and could serve as the primary key
- **primary key –** the key chosen by a database designer to represent relationships among rows in different tables
- **foreign key –** an attribute that duplicates the primary key of a different (or foreign) table
- **referential integrity –** a consistent state among foreign key and primary key values
- **referential integrity constraint –** a constraint, stored in the schema, that the DBMS uses to

automatically enforce referential integrity

- **normalization –** a formal technique for evaluating and improving the quality of a relational database schema
- **first normal form (1NF) –** restriction that all rows of a table must contain the same number of columns
- **functional dependency –** a one-to-one association between the values of two attributes
- **second normal form (2NF) –** restriction that a table is in 1NF and that each non-key attribute is functionally dependent on the entire primary key
- **third normal form (3NF) –** restriction that a table is in 2NF and that no non-key attribute is functionally dependent on any other non-key attribute
- **data type –** the storage format and allowable content of a program variable, class attribute, or relational database attribute or column
- **primitive data –** type a data type supported directly by computer hardware or a programming language
- **complex data type –** combinations of or extensions to primitive data types that are supported by programming languages, operating systems, and DBMSs

Lecture Notes

Definitions

Relational database tables are similar to conventional tables, however, relational database terminology is somewhat different from conventional table and file terminology. A single row of a table is called a **row**, **tuple**, or record, and a column of a table is called an **attribute** or field. A single cell in a table is called an **attribute value**, field value, or data element.

Each table in a relational database must have a unique **key**. Sometimes there may be multiple sets of attributes that uniquely identify each row. These are called **candidate keys**. Key attributes may be natural or invented. **Primary key**s are critical elements of relational database design because they are the basis for representing relationships among tables. Keys are the "glue" that binds rows of one table to rows of another table—in other words, keys relate tables to each other. A **foreign key** is an attribute that duplicates the primary key of a different (or foreign) table.

Designing Relational Databases

For database design, the preferred starting point is the domain model class diagram, rather than the design class diagram, because it omits many design details that aren't relevant to database design. To create a relational database schema from a domain model class diagram, follow these steps:

1. Create a table for each class.
2. Choose a primary key for each table (invent one, if necessary).
3. Add foreign keys to represent one-to-many associations.

4. Create new tables to represent many-to-many associations.
5. Represent classification hierarchies.
6. Define referential integrity constraints.
7. Evaluate schema quality and make necessary improvements.
8. Choose appropriate data types.
9. Incorporate integrity and security controls.

Creating tables from domain classes: The first step in creating a relational database schema is to create a table for each class on the class diagram. The attributes of each table will be the same as those defined for the corresponding class in the class diagram.

Choosing Primary Keys: After creating tables for each class, the designer selects a primary key for each table. If a table already has an attribute or set of attributes that are guaranteed to be unique, then the designer can choose that attribute or set of attributes as the primary key. If the table contains no possible keys, then the designer must invent a new key attribute. Because key creation and management are critical functions in databases and information systems, many relational DBMSs automate key creation.

Representing Associations: Associations are represented within a relational database by foreign keys. Which foreign keys should be placed in which tables depends on the type of association being represented. The rules for representing one-to-many and many-to-many associations are as follows:

- One-to-many associations—Add the primary key attribute(s) of the "one" class to the table that represents the "many" class.
- Many-to-many associations—If no association class exists, create a new table to represent the association. Add the primary key attribute(s) of the associated classes to the table that represents the association. The concatenation of the keys of the associated classes is always a valid candidate key of the association class.

Representing Classification Hierarchies: Classification hierarchies, such as the association among Sale, InStoreSale, TelephoneSale, and WebSale, are a special case in relational database design. This inheritance can be represented in multiple ways, including:

- Combining all the tables into a single table containing the superset of all classes
- Using separate tables to represent the child classes and using the primary key of the parent class table as the primary key of the child class tables

Enforcing Referential Integrity: For relational databases, the term **referential integrity** describes a consistent state among foreign key and primary key values. In most cases, a database designer wants to ensure that these references are consistent. That is, foreign key values that appear in one table must also appear as the primary key value of the related table. The DBMS usually enforces referential integrity automatically once the schema designer identifies primary and foreign keys.

Database Normalization

Databases embedded within information systems often survive several generations of programs. Because databases are difficult to change once they are populated with data, analysts take extra steps to ensure a high-quality database design. A high-quality relational database schema has these features:

1. Allows flexibility in implementing future data model changes
2. Contains a minimum of redundant data
3. Prevents insertion, deletion, and update anomalies

Normalization is a formal technique for evaluating and improving the quality of a relational database schema. It determines whether a database schema is flexible and whether it contains any of the "wrong" kinds of redundancy. It also defines specific methods to eliminate redundancy and improve flexibility.

First Normal Form: A table is in **first normal form (1NF)** if all rows contain the same number of columns. In other words if there are not multiply occurring attributes that differ between rows.

Functional Dependency: A **functional dependency** is a one-to-one association between the values of two attributes. The association is formally stated as follows:

> *Attribute A* is functionally dependent on *attribute B* if for each value of *attribute B* there is only one corresponding value of *attribute A.*

The most precise way to determine whether functional dependency exists is to pick two attributes in a table and insert their names in the italic portions of the previous statement and ask whether the result is true.

> *Description* is functionally dependent on *ProductItemID* if for each value of *ProductItemID* there is only one corresponding value of *Description*.

Second Normal Form: A table is in **second normal form (2NF)** if it is in 1NF and if each non-key attribute is functionally dependent on the entire primary key. A table violates 2NF when a non-key attribute is functionally dependent on only part of the primary key, which is only possible if the primary key contains multiple attributes. When a table's primary key consists of two or more attributes, the analyst must examine functional dependency of non-key attributes on each part of the primary key. The simplest way to test for 2NF is to test for functional dependency of non-key attributes on each subset of the primary key.

Third Normal Form: A table is in third normal form (3NF) if it is in 2NF and if no non-key attribute is functionally dependent on any other non-key attribute. To verify that a table is in 3NF, we must check the functional dependency of each non-key attribute on every other non-key attribute. A common example of a 3NF violation is an attribute that can be computed by a formula or algorithm that uses other stored values as inputs. Common examples of computable attributes include subtotals, totals, and taxes.

Data Types

A **data type** defines the storage format and allowable content of a program variable, class attribute, or

relational database attribute or column. **Primitive data types** are supported directly by computer hardware and programming languages and include integers, single characters, and real numbers (floating-point numbers). **Complex data types** are combinations of or extensions to primitive data types that are supported by programming languages, operating systems, and DBMSs.

Quick Quiz

Q: How are classes represented in a RDBMS?

A: Each table becomes a class.

Q: How are associations represented in a RDBMS?

A: One-to-many are represented by a foreign key. Many-to-many are represented by a table or if there is already an association class, then they are already in the RDBMS.

Q: What is another name for row?

A: tuple

Q: What is another name for a column?

A: Attribute

Q: What is a key?

A: A unique identifier for each row in a table.

Q: How are classification hierarchies (generalization/specialization) represented?

A: Either by a single table with all attributes of all classes, or by a table for each class, or some combination.

Q: What is first normal form?

A: When all the rows have the same number of attributes. In other words, when no row needs to have extra columns to hold some multiply occurring data field.

Distributed Database Architectures

Key Terms

- **decentralized database** – a database stored at multiple locations without needing to be interconnected through a network or synchronized
- **homogeneous distributed database** – a database distributed across multiple locations with the same DBMS, and all database access coordinated by a global schema
- **heterogeneous distributed database** – a database distributed across multiple locations with different DBMSs and with local access allowed without global schema coordination
- **database synchronization –** updating one database copy with changes made to other database copies

Lecture Notes

As an organization becomes very large, with a user base that spans the globe, it becomes more beneficial to locate the data at locations that are closer to the users. There are three scenarios for distributing the data:

- **Decentralized database.** In some situations, a global company may have pockets of database users who share data with each other, but do not have the need to share the data globally. In this situation, the data is purely local, even though the database configuration may be the same.
- **Homogeneous distributed database.** In the situation where the data need to be shared or at least available throughout the reach of the organization, a homogeneous distributed database is the correct structure. In this configuration, the global schema is available to every database user issuing database queries.
- **Heterogeneous distributed database.** This configuration combines the features of the two previous configurations, namely, that there are some users and queries that are purely local, combined with other users and queries that require global access.

Implementation Approaches for Distributed Database

Once the need for a distributed database is recognized, decisions must be made on how to partition the database and what the configuration will be at each location. The following sections briefly consider four distribution strategies.

Data Replication: Complete database copies are hosted at each location or server farm and maintained by cooperating DBMSs. The servers are usually distributed across geographic locations. Applications can direct access requests to any available server, with preference given to the nearest server.

To keep data current on all servers, each database copy must periodically be updated with changes from other database servers. This process is called **database synchronization**. The time delay between an update to a database copy and the propagation of that update to other database copies is an important database design decision. The proper synchronization strategy is a complex trade-off among cost, hardware and network capacity, and the need of application programs and users for current data.

Horizontal Partitioning: Horizontal partitioning of the database occurs when a table is split by storing some rows or records at one location and other rows or records at another location. Even though this is a fairly straightforward method to partition the database, it does get more complex as customers also become global and have accounts or loans in multiple locations throughout the world. Hence, even with horizontal partitioning there may be the need to maintain duplicate data or to synchronize data. To reconstruct the complete base tables requires that data from all locations be combined together.

Vertical Partitioning: Vertical partitioning of the database occurs when complete tables or only specific columns of a base table are stored at distinct locations. Distributing entire base tables to distinct locations is fairly straightforward. Distributing only columns of the same table to distinct locations is more complex, and is more complex than horizontal partitioning.

Combination of Replication, Horizontal, and Vertical Partitions: All of the preceding three techniques can be combined to provide the right data at the right place in the right form. The design and maintenance of this complex configuration would be designed under the direction of the data

administrator and the database administrator roles, as explained previously. A single project team would not take responsibility for designing and administering this scenario.

RMO Distributed Database Architecture

The starting point for designing a distributed database architecture is information about the data needs of geographically dispersed users.

- Warehouse staff members (Portland, Salt Lake City, and Albuquerque) need to check inventory levels, query orders, record back orders and order fulfillment, and record order returns.
- Phone-order staff members (Salt Lake City) need to check inventory levels; create, query, update, and delete orders; query customer account information; and query catalogs.
- Customers using the online sales system and sales associates in retail stores need the same access capabilities as phone-order staff.
- Marketing staff members (Park City) need to query and adjust orders, query and adjust customer accounts, and create and query product and promotion information.

A single-server architecture is infeasible for the CSMS. There are many accesses from many locations at many different times. A more complex alternative that addresses the risk is shown in Figure 12-18. Each remote location employs a combination of database partitioning and replication. The primary advantages of this architecture are fault tolerance and reduced WAN capacity requirements. The primary disadvantages to the distributed architecture are cost and complexity.

So does the proposed architecture make sense for RMO? The distributed architecture would provide higher performance and reliability but at substantially increased cost. Management must determine whether the extra cost is worth the expected benefits.

Quick Quiz

Q: What is the difference between a homogeneous distributed database and a heterogeneous distributed database?

> A: Homogeneous distributed database has the same schema at all locations. A heterogeneous distributed database may have some part of the schema that is common, but will also have portions of a local schema that are unique.

Q: What are the three primary methods used to distribute a database? Explain how each works.

> A: Data replication, horizontal partitioning and vertical partitioning. Replication is simply to duplicate the data at every location. Horizontal partitioning means that the table schemas are the same at different locations, but the data may be different. Vertical partitioning means that the table schemas are different (unique columns), but some tables are split by columns.

Q: Why would RMO decide on a distributed database solution?

> A: Due to the very distributed nature of the business, and the desire to have redundancy across all data centers.

Protecting the Database

Key Terms

transaction logging – a technique by which any update to the database is logged with such audit information as user ID, date, time, input data, and type of update

transaction – a piece of work with several steps that must all be completed to be valid

database lock – the technique of applying exclusive control to a part of the database so that one user at a time can use the data

shared or read lock – a lock where other transactions are allowed to read the data

exclusive or write lock – a lock where no other activity is allowed, neither reading nor writing the data

Lecture Notes

Protecting the database is a serious issue in the development and deployment of information systems, and a thorough treatment of the subject is well beyond the scope of this textbook.

Authentication and authorization apply directly to the protection of the data in the database and are used extensively before allowing access to the database. Additional techniques, such as data encryption, are also a frequently used technique to keep the data protected from unauthorized persons and programs. The responsibility for designing and deploying a secure database is shared between the project team, that is, the system development team, and the DBA.

In addition to the security issues for protecting the database, there is the need to protect the data in the database from catastrophes ranging from simple power outages to major natural disasters.

Transaction Logging

Transaction logging is a technique by which any update to the database is logged with such audit information as user ID, date, time, input data, and type of update. The fundamental idea is to create an audit trail of all database updates and, therefore, track any errors or problems that occur. Most DBMSs include transaction logging as part of the DBMS software.

Transaction logging achieves two objectives. First, it helps discourage fraudulent transactions or malicious database changes. Second, a logging system provides a recovery mechanism for erroneous transactions.

Currency and Complex Update Controls

Large, active databases often have tens of thousands of users every minute who need to read and update the data. Not only do databases need to be extremely fast, but they also need to be designed so that multiple users do not damage each other's data inadvertently when they are accessing or updating the same information at the same time.

One of the problems that potentially can occur when multiple people are updating the database at the same time is called the "lost update problem." To solve this problem, techniques that implement two

concepts are used. The first concept is of a **transaction**. A transaction is a piece of work that has several steps, including several reads and writes to the database, that must all be completed to be valid.

The second important concept that is widely used to protect against lost updates is to apply a database lock. A **database lock** is the technique where a portion of the database is locked by one user so that no other users (or programs) can use that portion of the database until the first user is finished and releases the lock.

There are two types of locks that are used, a shared lock, also called a read lock, and an exclusive lock, also called a write lock. If the user application only needs to read the data, with no need to update it, it can issue a shared lock. A **shared lock** allows other users to also read the same data, but not to update it. If the user application needs to read and then rewrite the data, then an **exclusive lock** is issued. No other user can even read the data that has an exclusive lock.

Quick Quiz

Q: What are two primary objectives of transaction logging?

> A: First it provides an audit trail, which will discourage fraudulent activity. Second, it provides a backup and recovery mechanism.

Q: What are the characteristics of a transaction?

> A: A transaction is a sequence of steps, usually more than one, that must all be completed as a unit. If not all steps are completed, then none of the steps should be completed, or left hanging.

Q: What is the difference between a shared lock and an exclusive lock?

> A: With a shared lock, other users can still read the data, but not update it. With an exclusive lock, no other users can access the data.

Classroom Activities

Classroom activities will depend on the level of competence required of the students. For most courses, students must be able to design relational database. A good classroom activity is to provide a problem domain class diagram and have the students, either singly, in pairs, or in small teams generate a relational database schema. Then select two or three groups to present their solution. To get a variation of solutions, it is a good idea when asking for the next volunteer to ask someone whose solution is different than the one already observed. Usually students enjoy showing their solutions, even when they are different than previously observed solutions. As with any student presentation, you can provide encouragement and recognition on parts that work well, but also be prepared to ask probing questions to help students identify and understand the weaknesses and errors in their solutions. This activity is a great learning experience where students both get to have hands-on learning, and observe other students' solutions.

Another good classroom activity is to normalize a set of tables. This activity can be done in the same manner as the database design, i.e. in small groups and then have a couple of groups present their solution and explain how they arrived at it.

Troubleshooting Tips

The concepts and skills in this chapter are fairly straightforward and the students generally seem to grasp these ideas without too much trouble. Designing the database schema is a skill that comes with practice. Thus we recommend that you provide amble opportunity for the students to practice designing a database schema and observing the solutions from other students.

The concept that is more difficult are the ideas associated with normalization. Sometimes it is difficult for students to understand functional dependencies. They will sometimes get the functional dependencies backwards. These two statements are the exact reverse of each other:

A functionally determines B. ←→ B is functionally dependent on A

It is normally a good idea to practice this in class with some tables. Here are some classroom examples that you might want to use:

1. Consider the following relation in abstract form (capitalized letters are attribute names, lower-case letters and numbers are values):

X

A	B	C	D	E
a1	b2	c1	d3	e2
a3	b2	c3	d2	e4
a1	b3	c1	d1	e2
a2	b4	c1	d4	e2

Identify the functional dependencies that seem to apply to X from the list below. (Example: A → B is false, because the value of A in row 1 is a1, and the value of B is b2, but in row 3, the value of A is again a1, but the value of B is b3. So the value of A does *not* determine the value of B.)
With Solution:
a.x A → C b. x D → E c. C → A d. E → B
e. E → A f. C → B g. B → D h. B → A

2. Consider the following relation (capitalized letters are attribute names, lower-case letters and numbers are values):

Y

A	B	C	D	E
a1	b2	c1	d3	e2
a2	b2	c3	d3	e4
a1	b3	c2	d1	e2
a2	b4	c5	d1	e5

Identify the functional dependencies that seem to apply to Y from the following list.
With Solution.
a. A → C b. D → E c.x C → A d. E → B
e.x E → A f.x C → B g.x B → D h. B → A

Discussion Questions

1. Career choices of Data Administrator versus Database Administrator

Data Administrator requires a skill set consisting of both business analytical skills with technical skills. A business person with strong technical skills can fulfill the Data Administration position. Database Administration, on the other hand, requires heavy technical skills. Database configuration, database tuning, and backup and recovery all requires strong technical background. As mentioned in the textbook, many companies will merge these two jobs into one position.

Class discussion could focus on the availability of these positions. How to prepare for them. How to seek them. Sometimes these positions require years of experience, so questions regarding how to gain the necessary experience make an interesting discussion.

2. Databases and Database Management Systems

Databases are such an integral part of all information systems that the importance of correct design and deployment is always a critical factor. Discussion can center around two major themes: Database design, which is the primary topic of this chapter, and database deployment and operation, which is not covered in this chapter, but is obviously an important part of a total solution.

Database design issues: Who should be in charge of designing the database – team members, who understand the application requirements, or a database design expert, who may not understand the problem domain issues but does understand good database design? How can the team verify that the solution is correct and, more difficult, that it is efficient? Another interesting combination of both database and integrity is how to design and implement so that the database is secure from injection attacks? What kind of encryption should be used within the database? What data needs to be secure?

Database deployment: Questions about deployment involve such issues as which RDBMS to choose? What criteria should be involved in choosing an RDBMS? After the system is deployed, what kind of statistics should be gathered? What kind of ongoing evaluation might be necessary to verify that the database is working correctly and that it is efficient? What kind of backup and recovery should be set up? How do backup and transaction logging work together?

3. RMO Distributed Database Architecture

RMO has two options for distributed database architecture, single server database architecture or replicated and partitioned database server architecture. Both alternatives have stated advantages and disadvantages. Data related to network traffic and experience is required to make this decision. How can these estimates be made without a working system to prototype? How do network analysts gain the experience needed to help them make these types of decisions?

If you wanted to improve performance in the single server architecture, which improvements would you make? If you wanted to improve performance in the replicated and partitioned database server architecture, which improvements would you make?