# Chapter 2: Database Design Fundamentals Solutions

## Answers to Review Questions

- An entity is a person, place, thing, or event.
- An attribute is a property of an entity.
- A relationship is an association between tables (entities). A one-to-many relationship between two tables is a relationship in which each row in the first table can be associated with many rows in the second table, but each row in the second table is associated with only one row in the first table.
- A repeating group is multiple entries in a single location in a table.
- A relation is a two-dimensional table in which the entries in the table are single-valued (each location in the table contains a single entry), each column has a distinct name (or attribute name), all values in a column are values of the same attribute, the order of the rows and columns is immaterial, and each row contains unique values.
- A relational database is a collection of relations.
- For each table, you write the name of the table and then within parentheses list all of the columns in the table. Underline the primary keys.

```
BRANCH (BRANCH_NUM, BRANCH_NAME, BRANCH_LOCATION,
NUM_EMPLOYEES) PUBLISHER (PUBLISHER_CODE,
PUBLISHER_NAME, CITY)
AUTHOR (AUTHOR_NUM, AUTHOR_LAST, AUTHOR_FIRST)
BOOK (BOOK_CODE, TITLE, PUBLISHER_CODE, TYPE, PRICE,
PAPERBACK) WROTE (BOOK_CODE, AUTHOR_NUM, SEQUENCE)
INVENTORY (BOOK_CODE, BRANCH_NUM, ON_HAND)
```

- To qualify the name of a field, indicate the table in which the field appears. You do this by preceding the name of the field with the name of the table and a period.
- A column (attribute), B, is functionally dependent on another column, A (or possibly a collection of columns), if at any point in time a value for A determines a single value for B.
- Column A (or a collection of columns) is the primary key for a table if (1) *All* columns in the table are functionally dependent on A and (2) No subcollection of the columns in A (assuming A is a collection of columns and not just a single column) also has property 1. The primary key of the BRANCH table is the BRANCH_NUM column. The primary key of the PUBLISHER table is the PUBLISHER_CODE column. The primary key of the AUTHOR table is the AUTHOR_NUM column. The primary key of the BOOK table is the BOOK_CODE column. The primary key of the WROTE table is the combination of the BOOK_CODE and AUTHOR_NUM

columns. the primary key of the INVENTORY table is the combination of the BOOK_CODE and BRANCH_NUM columns.

- Functional dependencies:

```
DEPARTMENT_NUM ϰ DEPARTMENT_NAME
ADVISOR_NUM ϰ ADVISOR_LAST_NAME, ADVISOR_FIRST_NAME,
DEPARTMENT_NUM COURSE_CODE ϰ DESCRIPTION
STUDENT_NUM ϰ STUDENT_LAST_NAME, STUDENT_FIRST_NAME,
ADVISOR_NUM STUDENT_NUM, COURSE_CODE ϰ GRADE
```

Relations:

```
DEPARTMENT (DEPARTMENT_NUM, DEPARTMENT_NAME)
ADVISOR (ADVISOR_NUM, ADVISOR LAST NAME,
      ADVISOR_FIRST_NAME, DEPARTMENT_NUM)
COURSE (COURSE_CODE, DESCRIPTION)
STUDENT (STUDENT_NUM, STUDENT LAST NAME,
      STUDENT_FIRST_NAME, ADVISOR_NUM
STUDENT_COURSE (STUDENT_NUM, COURSE_CODE, GRADE)
```

DEPARTMENT

Entity-Relationship diagram: (**NOTE**: Your rectangles may be in different positions as long as they are connected by the same arrows.)

ADVISOR

STUDENT

STUDENT_COURSE

COURSE

- A table (relation) is in first normal form (1NF) if it does not contain repeating groups.
- A table (relation) is in second normal form if it is in first normal form and no nonkey column is dependent on only a portion of the primary key. If a table is not in second normal form, the table contains redundancy, which leads to a variety of update anomalies. A change in a value can require not just one change, but several. There is the possibility of inconsistent data. Adding additional data to the database

may not be possible without creating artificial values for part of the key. Finally, deletions of certain items can result in inadvertently deleting crucial information from the database.

- A table is in third normal form if it is in second normal form and if the only determinants it contains are candidate keys. A change in a value can require not just one change, but several. There is the possibility of inconsistent data. Adding certain additional data to the database may not be possible without creating artificial rows in the table. Finally, deletions of certain items can result in inadvertently deleting crucial information from the database.
-

```
STUDENT (STUDENT_NUM, STUDENT_LAST_NAME,
 STUDENT_FIRST_NAME, ADVISOR_NUM)
ADVISOR (ADVISOR_NUM, ADVISOR_LAST_NAME,
ADVISOR_FIRST_NAME) COURSE (COURSE_CODE,
DESCRIPTION)
STUDENT_COURSE  (STUDENT_NUM, COURSE_CODE, GRADE)
```

## Answers to Premiere Products Exercises

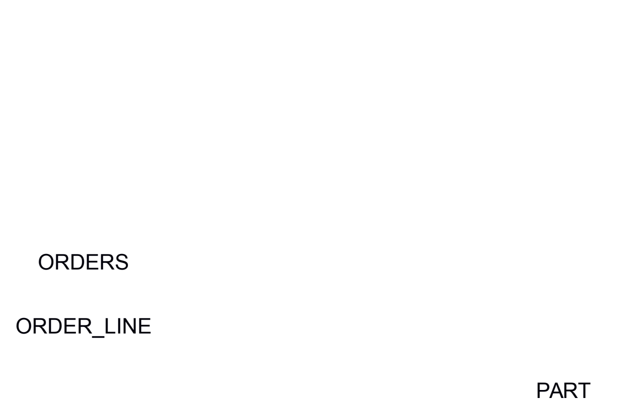
- **NOTES**: The CUSTOMER_REP table in the following lists implements the relationship between customers and reps. If customer 148, for example, is represented by both rep 20 and rep 35, there would be a row in the table in which the customer number is 148 and the rep number is 20 as well as a row in which the customer number is 148 and the rep number is 35. A row would only be allowed in the order table if the combination of the customer number and the rep number match a row in the CUSTOMER_REP table.

```
REP (REP_NUM, LAST_NAME,
      FIRST_NAME, STREET, CITY,
      STATE, ZIP, COMMISSION,
      RATE)
CUSTOMER (CUSTOMER_NUM,
      CUSTOMER_NAME, STREET, CITY,
      STATE, ZIP, BALANCE,
      CREDIT_LIMIT)
CUSTOMER_REP (CUSTOMER_NUM, REP_NUM)
ORDERS (ORDER_NUM, ORDER_DATE,
CUSTOMER_NUM, REP_NUM) ORDER_LINE
(ORDER_NUM, PART_NUM, NUM_ORDERED,
      QUOTED_PRICE)
PART (PART_NUM, DESCRIPTION,
      ON_HAND, CLASS, WAREHOUSE,
      PRICE)
```

Indicate the changes that need to be made to the design of the Premiere Products database to support the following: A customer is not necessarily represented by a single sales rep, but can be represented by several sales reps. When a customer places an order, the sales rep who gets the commission on the order must be one of the collection of sales reps who represent the customer.

Relationships: There are one-to-many relationships from REP to CUSTOMER_REP, CUSTOMER to CUSTOMER_REP, CUSTOMER_REP to ORDERS, ORDERS to ORDER_LINE, and PART to ORDER_LINE.

Entity-Relationship diagram: (**NOTE**: Your rectangles may be in different positions as long as they are connected by the same arrows.)



- **NOTES**: There is no relationship between customers and reps, so there is no REP NUM column in the CUSTOMER table nor is there an additional table like the CUSTOMER REP table in Exercise 1. A row can only exist in the ORDERS table if the customer number matches a row in the CUSTOMER table and the rep number matches a row in the REP table.

```
REP (REP NUM, LAST NAME, FIRST NAME, STREET, CITY,
      STATE, ZIP, COMMISSION, RATE)
CUSTOMER (CUSTOMER_NUM, CUSTOMER_NAME, STREET,
      CITY, STATE, ZIP, BALANCE, CREDIT_LIMIT)
ORDERS (ORDER_NUM, ORDER_DATE, CUSTOMER_NUM,
REP_NUM) ORDER_LINE (ORDER_NUM, PART_NUM,
NUM ORDERED, QUOTED PRICE) PART (PART NUM,
DESCRIPTION, ON_HAND, CLASS, WAREHOUSE, PRICE)
```

Relationships: There are one-to-many relationships from REP to ORDERS, CUSTOMER to ORDERS, ORDERS to ORDER_LINE, and PART to ORDER_LINE.

Entity-Relationship diagram: (**NOTE**: Your rectangles may be in different positions as long as they are connected by the same arrows.)

- **NOTES**: The WAREHOUSE NUM and ON HAND columns do not appear in the PART table. There is a WAREHOUSE table, whose key is WAREHOUSE NUM and which contains the warehouse description. Information about units on hand is stored in a

new table, the PART WAREHOUSE table, whose key is the combination of the part number and warehouse number. If there are 10 units of part AT94 on hand in warehouse 2, for example, there would be a row in PART_WAREHOUSE on which the part number is AT94, the warehouse number is 2, and the number of units on hand is 10.

```
REP (REP_NUM, LAST_NAME, FIRST_NAME, STREET, CITY,
      STATE, ZIP, COMMISSION, RATE)
CUSTOMER (CUSTOMER_NUM, CUSTOMER_NAME, STREET,
      CITY, STATE, ZIP, BALANCE, CREDIT_LIMIT,
      REP_NUM)
ORDERS (ORDER_NUM, ORDER_DATE, CUSTOMER_NUM)
ORDER_LINE (ORDER_NUM, PART_NUM, NUM_ORDERED,
QUOTED_PRICE) PART (PART_NUM, DESCRIPTION,
CLASS, PRICE)
WAREHOUSE (WAREHOUSE_NUM,
WAREHOUSE_DESCRIPTION) PART_WAREHOUSE
(PART_NUM, WAREHOUSE_NUM, ON_HAND)
```

Relationships: There are one-to-many relationships from REP to CUSTOMER, CUSTOMER to ORDERS, ORDERS to ORDER_LINE, PART to ORDER LINE, PART to PART WAREHOUSE, and WAREHOUSE to PART_WAREHOUSE.

REP

Entity-Relationship diagram: (**NOTE**: Your rectangles may be in different positions as long as they are connected by the same arrows.)

CUSTOMER

ORDERS

ORDER_LINE

- Functional Dependencies:

PART_NUM ϰ DESCRIPTION, ON_HAND, CLASS, WAREHOUSE,
PRICE ORDER_NUM ϰ ORDER_DATE, CUSTOMER_NUM
CUSTOMER_NUM ϰ CUSTOMER_NAME
PART_NUM, ORDER_NUM ϰ NUM_ORDERED, QUOTED_PRICE

Relations:

PART (PART_NUM, DESCRIPTION, ON_HAND, CLASS,
WAREHOUSE, PRICE) ORDERS (ORDER_NUM, ORDER_DATE,
CUSTOMER_NUM)

CUSTOMER (CUSTOMER_NUM, CUSTOMER_NAME)
ORDER_LINE (PART_NUM, ORDER_NUM, NUM ORDERED, QUOTED_PRICE)

**NOTE**: The keys for ORDER_LINE could also have been listed as ORDER_NUM, PART_NUM.

## Answers to Henry Books Exercises

- Tables (Relations):

DIRECTOR (DIRECTOR_NUM, DIRECTOR_NAME, YEAR_BORN,
YEAR_DIED) MOVIE (MOVIE_NUM, MOVIE_TITLE,
YEAR_MADE, TYPE, DIRECTOR_NUM,
CRITICS_RATING, MPAA_RATING, NUM_AWARDS_NOM,
NUM_AWARDS_WON) STAR (STAR_NUM, STAR_NAME, BIRTHPLACE,
YEAR_BORN, YEAR_DIED) MOVIE_STAR (MOVIE_NUM, STAR_NUM)

Relationships: There are one-to-many relationships from DIRECTOR to MOVIE, MOVIE to MOVIE_STAR, and STAR to MOVIE_STAR.

DIRECTOR

Entity-Relationship diagram: (**NOTE**: Your rectangles may be in different positions as long as they are connected by the same arrows.)

MOVIE

MOVIE_STAR

STAR

- Functional Dependencies:

BOOK_CODE ϰ TITLE, TYPE, PRICE
AUTHOR_NUM ϰ AUTHOR_LAST,
AUTHOR_FIRST

Tables (Relations):

```
BOOK (BOOK_CODE, TITLE, TYPE, PRICE)
AUTHOR (AUTHOR_NUM, AUTHOR_LAST,
AUTHOR_FIRST) ) BOOK_AUTHOR
(BOOK_CODE, AUTHOR_NUM)
```

**NOTE:** The BOOK_AUTHOR relation is necessary to relate books and authors. (You could have assigned it any name you like.) The key could have been listed as AUTHOR_NUM, BOOK_CODE rather than BOOK_CODE, AUTHOR_NUM.

- Functional Dependencies:

```
BOOK_CODE ϰ TITLE, TYPE, PRICE,
PUB_CODE PUB_CODE ϰ PUBLISHER_NAME,
CITY
```

Tables (Relations):

```
BOOK (BOOK_CODE, TITLE, TYPE,
PRICE, PUB_CODE) PUBLISHER
(PUB_CODE, PUBLISHER_NAME, CITY)
```

## Answers to Alexamara Marina Group Exercises

- **NOTES**: There are two ways to handle the MARINA SLIP table, the table that contains information about the slips within each marina. You can add a column to identify each slip, for example SLIP_ID (see Figure 1.10 for an illustration of the use of this column). Alternatively, you could do without this column by making the primary key the combination of the marina number and the slip number. (If the SLIP_ID column were missing from Figure 1.10, the primary key would be the combination of the MARINA NUM and SLIP NUM columns.) Both approaches are legitimate and are illustrated below.

Tables (Relations):

Option 1 (With SlipID added as a unique identifier for MARINA_SLIP):

```
MARINA (MARINA_NUM, NAME, ADDRESS, CITY, STATE, ZIP)
OWNER (OWNER_NUM, LAST_NAME, FIRST_NAME, ADDRESS, CITY,
STATE, ZIP) MARINA_SLIP (SLIP_ID, MARINA_NUM, SLIP_NUM,
LENGTH, RENTAL_FEE,
      BOAT_NAME, BOAT_TYPE, OWNER_NUM)
SERVICE_CATEGORY (CATEGORY_NUM,
CATEGORY_DESCRIPTION)
SERVICE_REQUEST (SERVICE_ID, SLIP_ID, CATEGORY_NUM,
      DESCRIPTION, STATUS, EST_HOURS, SPENT_HOURS,
      NEXT_SERVICE_DATE)
```

Option 2 (Without SlipID):

```
MARINA (MARINA_NUM, NAME, ADDRESS, CITY, STATE, ZIP)
OWNER (OWNER_NUM, LAST_NAME, FIRST_NAME, ADDRESS, CITY,
STATE, ZIP) MARINA_SLIP (MARINA_NUM, SLIP_NUM, LENGTH,
RENTAL_FEE, BOAT_NAME,
      BOAT_TYPE, OWNER_NUM)
SERVICE_CATEGORY (CATEGORY_NUM, CATEGORY_DESCRIPTION)
SERVICE_REQUEST (SERVICE_ID, MARINA_NUM, SLIP_NUM,
CATEGORY_NUM,
      DESCRIPTION, STATUS, EST_HOURS,
      SPENT_HOURS, NEXT_SERVICE_DATE)
```

Relationships: There are one-to-many relationships from MARINA to MARINA_SLIP, OWNER to MARINA_SLIP, SERVICE_CATEGORY to SERVICE_REQUEST, and MARINA_SLIP to SERVICE_REQUEST.

Entity-Relationship diagram: (**NOTE**: Your rectangles may be in different positions as long as they are connected by the same arrows.)

- Functional Dependencies:

```
MARINA_NUM ϰ NAME
MARINA_NUM, SLIP_NUM ϰ LENGTH, RENTAL_FEE, BOAT_NAME
```

Tables (Relations):

```
MARINA (MARINA_NUM, NAME)
MARINA_SLIP (MARINA_NUM, SLIP_NUM, LENGTH, RENTAL_FEE, BOAT_NAME)
```

- Functional Dependencies:

```
ID ϰ MARINA_NUM, SLIP_NUM, LENGTH, RENTAL_FEE,
      BOAT_NAME, BOAT_TYPE, OWNER_NUM, LAST_NAME,
      FIRST_NAME
OWNER_NUM ϰ LAST_NAME, FIRST_NAME
```

Tables (Relations):

```
MARINA_SLIP (ID, MARINA_NUM, SLIP_NUM, LENGTH,
      RENTAL_FEE, BOAT_NAME, BOAT_TYPE,
      OWNER_NUM)
OWNER (OWNER_NUM, LAST_NAME, FIRST_NAME)
```

# Chapter 2
# Database Design Fundamentals

## At a Glance

## Table of Contents

## Lecture Notes

### Overview

In this chapter, students learn about database design. Students examine the important concepts related to databases. They learn how to identify tables and columns and how to identify the relationships between the tables. Students learn how to produce an appropriate database design for a given set of requirements. They examine the process of normalization, a process that identifies and fixes potential problems in a database design. Finally, students learn how to represent a database design in a visual manner.

### Chapter Objectives

- Understand the terms *entity*, *attribute*, and *relationship*

- Understand the terms *relation* and *relational database*
- Understand functional dependencies and be able to identify when one column is functionally dependent of another
- Understand the term *primary key* and identify primary keys in tables
- Design a database to satisfy a set of requirements
- Convert an unnormalized relation to first normal form
- Convert tables from first normal form to second normal form
- Convert tables from second normal form to third normal form
- Create an entity-relationship diagram to represent the design of a database

**Teaching Tips**

- This chapter does not need to be covered in sequence. It can be covered later in the course. If you are using a textbook such as Pratt and Adamski's *Concepts of Database Management, Fifth Edition,* you may want to skip this chapter entirely.
- Be prepared to spend considerable class time on this chapter. The material is complex and it is important that students understand all the concepts presented. The best way for students to learn the material is to work through lots of examples. Use the embedded questions that are included throughout the chapter to test students' understanding.
- Encourage students to bring their texts with them to class so they can review the examples.
- Use the review questions and exercises at the end of the chapter as in-class exercises. Work through some of the assignments with the class and assign groups of students to work on others. Students really benefit from working through the exercises in a group.
- Point out that normalization is a technique that allows us to analyze the design of a relational database to see whether it is bad. It alerts us to update anomalies and provides a method for correcting those problems. The goal of normalization is to start with a table or collection of tables and produce a new collection of tables that is equivalent (represents the same information) but is free of problems.
- For more information on the database concepts presented in this chapter, see http://www.service-architecture.com/database/articles/. In addition, the Pratt

and Adamski's text published by Course Technology is an excellent textbook to use with this textbook for a course requiring that students learn both concepts and MySQL.

## Instructor Notes

### Introduction

The process of determining the particular tables and columns that will comprise a database is known as **database design**. This chapter identifies the concepts and techniques that students need to design databases that are free of potential problems.

### Database Concepts

Before learning how to design a database, you need to be familiar with some important database concepts. This section explains these concepts.

#### Relational Databases

A **relational database** is a collection of tables. Formally, tables are called relations. Use Figure 2-1 to emphasize that the Premiere Products database is a collection of tables. Review the Note on page 28.

#### Entities, Attributes, and Relationships

Define and discuss the database terms: entity, attribute, and relationship. An **entity** is a person, place, object, event, or idea for which you want to store and process data. The entities of interest to Premiere Products are customers, orders, parts, and sales reps.

An **attribute** is a characteristic or property of an entity. The terms column and field are used as synonyms in many database systems. For Premiere Products, the attributes of interest for the entity "customer" are such things as customer name, street, city, and so on.

Define relationship and one-to-many relationship. A **relationship** is an association between entities. There is a **one-to-many relationship** between sales reps and customers in the Premiere Products database. One sales rep represents many customers but each customer is associated with only one sales rep.

In a relational database, each entity has its own tables and the attributes of the entity are columns in the table. A one-to-many relationship is handled by using common columns in the two tables.

A **relation** is a two-dimensional table with specific properties. These properties include:

- Entries in the table are single-valued.
- Each column has a distinct name.
- All values in a column are values of the same attribute.

- The order of the columns is immaterial.
- Each row is distinct.
- The order of the rows is immaterial.

If a structure contains entries that are not single-valued (**repeating groups** occur), then the structure is called an unnormalized relation. Use Figure 2-2 to discuss repeating groups. Use Figure 2-3 to discuss the six properties of a relation. See the Note on page 31. Mention that the formal term for a table is relation, and the formal term for a row is **tuple**. A row also is called a **record**. Columns in a table are also called **fields** or attributes.

DBDL (Database Definition Language) is a commonly accepted shorthand notation for showing the structure of a table. After the name of the table, all the columns in the table are listed within a set of parentheses. While each column in a table has a distinct name, the same column name can be used in more than one table within the same database. When two or more tables in a database use the same column name, **qualify** the column name, that is, combine the table name and the column name.

- A(n)______is a person, place, object, event, or idea for which you want to store and process data.
  Answer: entity

- A(n)______is a characteristic or property of an entity. Answer: attribute

- A(n)______is the association between entities. Answer: relationship

## Functional Dependence

Functional dependence is a formal name for what is a simple idea. In a relational database, column B, is **functionally dependent** on another column A (or possibly a collection of columns), if a value for A determines a single value for B at any one time. Another way of defining functional dependence is to say that A **functionally determines** B. Use Figure 2-4 to explain functional dependence. Make sure that students understand what functional dependence is before proceeding, or they will be lost for the remainder of the chapter. Review the embedded Questions and Answers on pages 32 and 33.

Use Figures 2-5 and 2-6 to point out that you cannot determine functional dependencies by looking at sample data. You must understand the users' policies.

## Primary Keys

To make each row distinct, one or more columns must uniquely identify a given row in a table. This column or collection of columns is called the **primary key**.

A more precise definition for a primary key is the following:
Column (attribute) A (or a collection of columns) is the **primary key** for a table (relation), R, if:
    Property 1: All columns in R are functionally dependent on A.

Property 2: No subcollection of the columns in A (assuming that A is a collection of columns and not just a single column) also has Property 1.

Review the embedded Questions and Answers on pages 34 and 35 to make sure students understand the concept of a primary key. Explain that when using the shorthand representation of a database, the primary key is underlined.

Discuss the three Notes on page 36. Point out that a **candidate key** is a column or collection of columns on which all columns in the table are functionally dependent. The definition for a primary key really defines a candidate key as well. If two or more columns in a table are identified as candidate keys, choose one to be the primary key. The decision is usually base on the specific application for which the database will be used.

- A(n) is a two-dimensional table.
  Answer: relation

- In the simplest terms, a(n) ______ is the unique identifier for a table. Answer: primary key

- _____ occur when there are multiple entries in an individual location in a table. Answer: Repeating groups

## Database Design

This section presents a specific method for designing a database, given a set of requirements that the database must support. The determination of the requirements is part of the process known as systems analysis. For more information on how to elicit database requirements, see Pratt and Adamski's *Concepts of Database Management, Fifth Edition.*

### Design Method

Review the design steps given in this section. Use Figure 2-1 as a visual aid as you explain each of these steps and ask the students to identify the items listed in the steps.

To design a database for a set of requirements:

- Read the requirements, identify the entities (objects) involved, and name the entities.
- Identify the unique identifiers for the entities identified in step 1.
- Identify the attributes for all the entities.
- Identify the functional dependencies that exist among the attributes.
- Use the functional dependencies to identify the tables by placing each attribute with the attribute or minimum combination of attributes on which it is functionally dependent.
- Identify any relationships between tables.

### Database Design Requirements

This section reviews the requirements that the database for Premiere Products must support. The database must store specific data about sales reps, customers, parts, orders, and order lines. Additionally, there are certain constraints, such as, "there is only one customer per order" that the database must enforce. Again, use

Figure 2-1 to illustrate the requirements. You also can refer to Figure 1-1, which shows a sample order for Premiere Products.

**Database Design Process Example**
Once the requirements for a database are known, you can apply the six steps given in the Design Method section on pages 36 and 37 to produce the appropriate database design: This section gives an example of using the six steps to create a database design for Premiere
Products. Be sure to point out the functional dependencies discussed in step 4. It is in this step that students often have trouble. Review the embedded Questions and Answers on pages 41 and 42.

## Normalization
Stress that database design is an iterative process. Once you create an initial database design, you must analyze it for potential problems. **Normalization** is process in which you identify the existence of potential problems such as data duplication and redundancy, and implement ways to correct these problems. The goal of normalization is to convert unnormalized relations into various types of normal forms. An **unnormalized relation** is a relation (table) that contains a repeating group. A table in a particular **normal form** possesses a certain desirable collections of properties. Normalization is a process in which a table that is in first normal form is better than a table that is not in first normal form, a table in second normal form is better than a table in first normal form and so on. The goal of normalization is to take an initial collection of tables and produce a new collection of tables that represents the same information but is free of problems.

### First Normal Form
An unnormalized relation is a relation (table) that contains a repeating group. A table (relation) is in **first normal form (1NF)** if it does not contain repeating groups. Use Figures 2-7 and 2-8 to explain converting an unnormalized table to 1NF. In general, when converting a non-first normal form table to first normal form, the primary key usually will include the original primary key concatenated with the key to the repeating group.

### Second Normal Form
Use Figure 2-9 to illustrate a relation that is in first normal form but not in second normal form. Point out the **redundancy**, that is, duplication of data in Figure 2-9. This duplication can cause update anomalies. **Update anomalies** fall into four categories:

| | |
|---|---|
| Update | Instead of changing one row, it is necessary to update multiple rows. |
| Inconsistent data | If the same value appears in more than one row, for example, part description, an update could change one row without changing the other rows. |
| Additions | Cannot add a record correctly. |
| Deletions | Cannot delete a record correctly. |

These anomalies occur when a column is dependent on only a portion of the primary key. Emphasize the fact that much real-world data (including relational data) are not well structured and have update anomalies.

Second normal form eliminates update anomalies caused by partial dependencies. A table (relation) is in **second normal form (2NF)** if it is in first normal form and no nonkey column is dependent on only a portion of the primary key. A column is a **nonkey column** if it is not a part of the primary key. Point out again that you cannot determine functional dependence by looking at sample data. Use Figure 2-10 to explain converting to 2NF. Mention the Note on page 47. If a relation has a single-column primary key , it automatically is in 2NF.

**Third Normal Form**
Relations that are in 2NF can still cause potential problems. Use Figure 2-11 to illustrate update anomalies with a table in 2NF. Any column or collection of columns that determines another column is called a **determinant**. A candidate key is a column or collection of columns that could function as the primary key. Update anomalies also can occur when one nonkey column determines another nonkey column. A table (relation) is in **third normal form (3NF)** if it is in second normal form and the only determinants it contains are candidate keys. Use Figure 5.11 to review the dependencies in the Customer table. Use Figure 2-12 to explain converting to 3NF. Show students how each progressive normal form solves update problems of the previous normal form. Mention the Note on page 50. The definition used in this text for 3NF is really the definition for **Boyce-Codd normal form (BCNF).** Review the embedded Question and Answer on page 53.

## Quick Quizzes

- A relation is in normal form if no repeating groups exist. Answer: first

- If the primary key of a relation contains only a single column, then the relation is automatically in____normal form.
  Answer: second

- Any column (or collection of columns) that determines another column is called a(n)
  _____.
  Answer: determinant

### Diagrams for Database Design
There is an old adage that “a picture is worth a thousand words.” For many people, a database design is easier to understand if it is depicted in graphical form. One graphical model for representing a database design is the **entity-relationship (E-R) diagram**. In and E-R diagram, a rectangle represents an entity or table. One-to-many relationships between entities are drawn as lines between the corresponding rectangles. There are several different styles of E-R diagrams. Use Figures 2-13 through 2-15 to illustrate the different styles of E-R diagrams.

## Classroom Activities
Ask students for other examples of relations (tables) that could have more than one candidate key. Some examples are: state data that contain both state

abbreviation and state name; and inventory data that contain both a tag number and a serial number.

## Discussion Questions

Have students read the second Note on page 36. Ask them how they feel about using social security number as a primary key? Also, ask them for additional databases where social security number is being used as a primary key other than banks and places of employment. Some examples are: insurance company, doctor's office, dentist's office.

## Projects to Assign

Place students in teams. Have them design a database to meet the requirements for a student activity database. The database must keep track of information about the student as well as the student's participation in campus activities Attributes such as number of years in activity as well as any office held are important. A student may engage in more than one activity.

## Key Terms

- **attribute :** A characteristic or property of an entity
- **Boyce-Codd normal form (BCNF):** A relation is in Boyce-Codd normal form if it is in second normal form and the only determinants it contains are candidate keys; also called third normal form
- **candidate key:** A minimal collection of columns in a table
- **concatenation:** A combination of columns
- **database design:** Process of determining the particular tables and columns that will comprise a database
- **determinant:** A column in a table that determines at least one other column
- **entity :** A person, place, object, event, or idea for which you want to store and process data
- **entity-relationship (E-R) diagram:** A graphical illustration for database design that uses rectangles for entities and arrows for relationships
- **field :** An attribute
- **first normal form (1NF):** A table that does not contain any repeating groups
- **functionally dependent:** Column B is functionally dependent on column A (or on a collection of columns) if a value for A determines a single values for B at any one time
- **functionally determines:** Column A functionally determines column B if B is functionally dependent on A
- **nonkey column:** A column that is not part of the primary key
- **normal form:** A progression that proceeds from first normal form to second normal form to third normal form. A table in a particular normal form possesses a certain desirable collection of properties.
- **normalization:** A process that analyzes a database design to identify the existence of potential problems and implements ways to correct these problems
- **one-to-many relationship:** A relationship in which one entity is associated with many other entities
- **primary key :** The column or collection of columns that uniquely identifies a given row in a table
- **qualify:** To combine a column name with a table name

- **record :** A row in a table
- **redundancy:** Duplication of data
- **relation :** A two dimensional table in which the entries are single valued; each column has a distinct name (or attribute name); all values in a column are values of the same attribute; the order of the rows and columns is immaterial; and each row contains unique values
- **relational database :** A collection of relations
- **relationship :** The association between entities
- **repeating group :** Multiple entries in an individual location in a table
- **second normal form (2NF):** A table that is in first normal form and where no nonkey column is dependent on only a portion of the primary key

- **third normal form (3NF):** A table that is in second normal form and the only determinants are candidate keys
- **tuple :** A row in a table
- **unnormalized relation:** A table that satisfies the definition of a relation except that they might contain repeating groups
- **update anomaly:** An update problem that can occur in a database as a result of a faulty design