PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

CHAPTER 1 SQL and Data

Oracle SQL By Example, Fourth Edition by Alice Rischert. Published by Prentice Hall. Copyright © 2008 by Pearson Education, Inc.

CHAPTER OBJECTIVES

In this chapter, you will learn about:

- Relational Databases
- Data Normalization and Table Relationships
- The STUDENT Schema Diagram

Before getting started writing SQL statements, it is beneficial to introduce some basic database concepts and review the history of databases.

What Is SQL?

SQL (pronounced "*sequel*") is an acronym for *Structured Query Language*, a standardized language used to access and manipulate data. The history of SQL corresponds closely with the development of a relational databases concept published in a paper by Dr. E. F. Codd at IBM in 1970. He applied mathematical concepts to the specification of a method for data storage and access; this

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

specification, which became the basis for relational databases, was intended to overcome the physical dependencies of the then-available database systems. The SQL language (originally called “System R” in the prototype and later called “SEQUEL”) was developed by the IBM Research Laboratory as a standard language to use with relational databases. In 1979 Oracle, then called Relational Software, Inc., introduced the first commercially available implementation of a relational database incorporating the SQL language.

The SQL language evolved with many additional syntax expansions incorporated into the American National Standards Institute (ANSI) SQL standards developed since. Individual database vendors continuously added extensions to the language, which eventually found their way into the latest ANSI standards used by relational databases today. Large-scale commercial implementations of relational database applications started to appear in the mid- to late 1980s, as early implementations were hampered by poor performance. Since then, relational databases and the SQL language have continuously evolved and improved.

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

Why Learn SQL?

Today, SQL is accepted as the universal standard database access language. Databases using the SQL language are entrusted with managing critical information affecting many aspects of our daily lives. Most applications developed today use a relational database, and Oracle continues to be one of the largest and most popular database vendors. Although relational databases and the SQL language are already over 30 years old, there seems to be no slowing down of the popularity of the language. Learning SQL is probably one of the best long-term investments you can make for a number of reasons:

- SQL is used by most commercial database applications.
- Although the language has evolved over the years with a large array of syntax enhancements and additions, most of the basic functionality has remained essentially unchanged.
- SQL knowledge will continue to be a fundamental skill because there is currently no mature and viable alternative language that accomplishes the same functionality.

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

- Learning Oracle's specific SQL implementation allows you great insight into the feature-rich functionality of one of the largest and most successful database vendors.

An Introduction to Databases

Before you begin to use SQL, you must know about data, databases, and relational databases. What is a database? A *database* is an organized collection of data. A *database management system (DBMS)* is software that allows the creation, retrieval, and manipulation of data. You use such systems to maintain patient data in a hospital, bank accounts in a bank, or inventory in a warehouse.

A *relational database management system (RDBMS)* provides this functionality within the context of the relational database theory and the rules defined for relational databases by Codd. These rules, called "Codd's Twelve Rules," later expanded to include additional rules, describe goals for database management systems to cope with ever-challenging and demanding database requirements. Compliance with Codd's Rules has been a major challenge for database vendors, and early versions of relational databases complied with only a handful of the rules.

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

Understanding relational database concepts provides the foundation for understanding the SQL language. Those unfamiliar with relational concepts or interested in a refresher will receive an overview of basic relational theories in the next two labs. If you are already familiar with relational theory, you can skip the first two labs and jump directly to Lab 1.3, which teaches you about the organization of a fictional university's student database called STUDENT. This database is used throughout the exercises in this book.

LAB 1.1 The Relational Database

LAB OBJECTIVES

After this lab, you will be able to:

- Understand Relational Database Concepts
- Define *SQL*. Name the Components of a Relational Database

This lab introduces you to the terminology of relational databases. You will appreciate some of the inherent advantages of relational database design and gain an overview of SQL language commands.

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

What Is Data?

Data is all around you; you make use of it every day. Your hair may be brown, your flight leaves from gate K10, you try to get up in the morning at 6:30 A.M. Storing data in related groups and making the connections among them are what databases are all about.

You interact with a database when you withdraw cash from an ATM, order a book from a Web site, or check stock quotes on the Internet. The switch from the information processing society to the knowledge management society will be facilitated by databases. Databases are a major asset to any organization. They contain information that helps run the business, and the information and technology employed in these database systems represent the backbones of the many technological advances we enjoy today.

Before the availability of relational databases, data was stored in individual files that could not be accessed unless you knew a programming language. Data could not be combined easily, and modifications to the underlying database structures were extremely difficult. The relational model conceived by E. F. Codd provided the

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

framework to solve a myriad of these and many other database problems.

How Is Data Organized?

Relational databases offer *data independence*, meaning a user does not need to know on which hard drive and file a particular piece of information is stored. The RDBMS provides users with *data consistency* and *data integrity*. For example, if an employee works in the finance department, and we know that he can work for only one department, then there should not be duplicate department records or contradicting data in the database. As you work through this lab, you will discover many of these useful and essential features.

A relational database stores data in tables, essentially a two-dimensional matrix consisting of columns and rows. Let's start with a discussion of the terminology used in relational databases.

TABLES

A table typically contains data about a single subject. Each table has a unique name that signifies the contents of the data. A database usually consists of many tables. For example, you may store data about books you read

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

in a table called BOOK and store details about authors in the AUTHOR table.

COLUMNS

Columns in a table organize the data further, and a table consists of at least one column. Each column represents a single, low-level detail about a particular set of data. The name of the column is unique within a table and identifies the data you find in the column. For example, the BOOK table may have a column for the title, publisher, date the book was published, and so on. The order of the columns is unimportant because SQL allows you to display data in any order you choose.

ROWS

Each row usually represents one unique set of data within a table. For example, the row in Figure 1.1 with the title "The Invisible Force" is unique within the BOOK table. All the columns of the row represent respective data for the row. Each intersection of a column and row in a table represents a value, and some do not, as you see in the PUBLISH_DATE column. The value is said to be *null*. Null is an unknown value, so it is not even blank spaces. Nulls cannot be evaluated or compared because they are unknown.

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

FIGURE 1.1 The BOOK table

BOOK_ID	TITLE	PUBLISHER	PUBLISH_DATE
1010	The Invisible Force	Literacy Circle	
1011	Into The Sky	Prentice Hall	10/2008
1012	Making It Possible	Life Books	08/2010

Row

Column

PRIMARY KEY

When working with tables, you must understand how to uniquely identify data within a table. This is the purpose of the *primary key*. This means you find one, and only one, row in the table by looking for the primary key value. Figure 1.2 shows an example of the CUSTOMER table with CUSTOMER_ID as the primary key of the table.

FIGURE 1.2 Primary key example

CUSTOMER_ID	CUSTOMER_NAME	ADDRESS	PHONE	ZIP
2010	Movers, Inc.	123 Park Lane	212-555-1212	10095
2011	Acme Mfg, Ltd.	555 Broadway	212-566-1212	10004
2012	ALR Inc.	50 Fifth Avenue	212-999-1212	10010

Primary Key

At first glance, you might think that the CUSTOMER_NAME column can serve as the primary

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

key of the CUSTOMER table because it is unique. However, it is entirely possible to have customers with the same name. Therefore, the CUSTOMER_NAME column is not a good choice for the primary key. Sometimes the unique key is a system-generated sequence number; this type of key is called a *synthetic*, or *surrogate, key*. The advantage of using a surrogate key is that it is unique and does not have any inherent meaning or purpose; therefore, it is not subject to changes. In this example, the CUSTOMER_ID column is such a surrogate key.

It is best to avoid any primary keys that are subject to updates as they cause unnecessary complexity. For example, the phone number of a customer is a poor example of a primary key column choice. Though it may possibly be unique within a table, phone numbers can change and then cause a number of problems with updates of other columns that reference this column.

A table may have only one primary key, which consists of one or more columns. If the primary key contains multiple columns, it is referred to as a *composite primary key*, or *concatenated primary key*. (Choosing appropriate keys is discussed further in Chapter 12, "Create, Alter, and Drop Tables.") Oracle does not require that every table have a primary key, and there

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

may be cases in which it is not appropriate to have one. However, it is strongly recommended that most tables have a primary key.

FOREIGN KEYS

If you store customers and the customers' order information in one table, each customer's name and address is repeated for each order. Figure 1.3 depicts such a table. Any change to the address requires the update of all the rows in the table for that individual customer.

FIGURE 1.3 Example of CUSTOMER data mixed with ORDER data

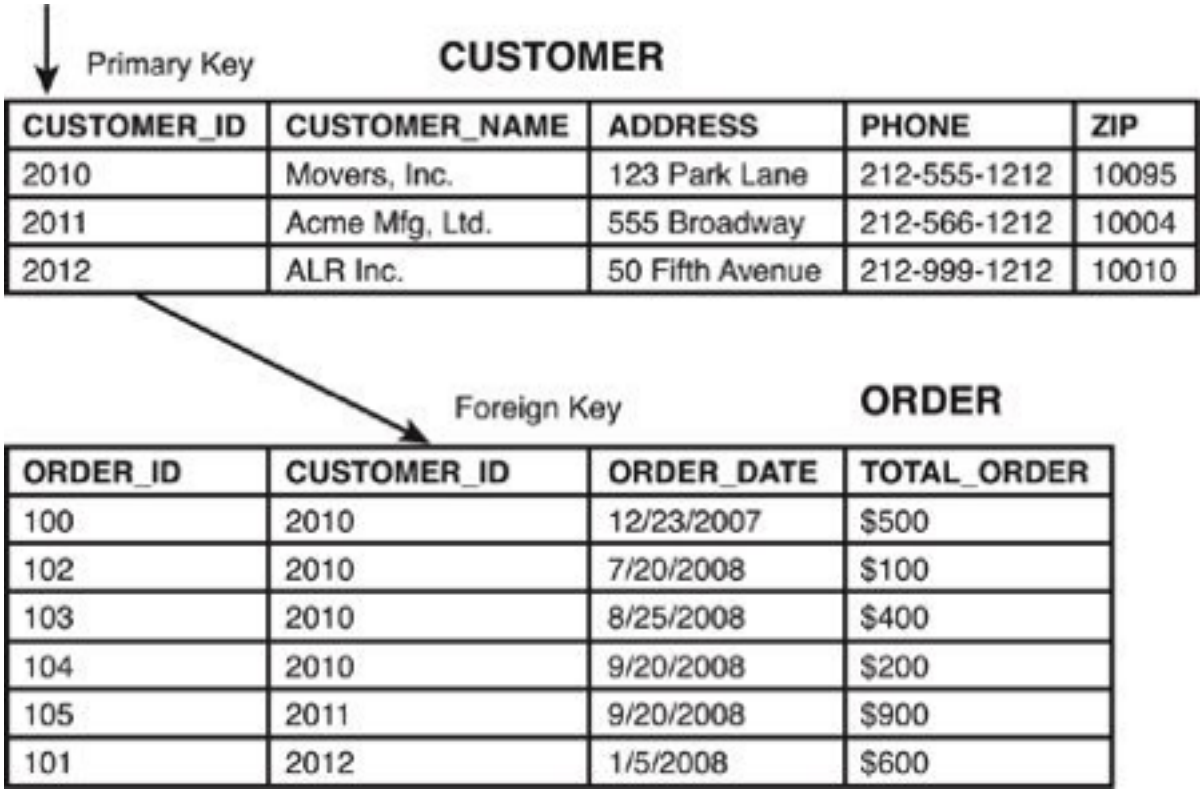
CUSTOMER_ID	CUSTOMER_NAME	ADDRESS	PHONE	ZIP	ORDER_ID	ORDER_DATE	TOTAL_ ORDER
2010	Movers, Inc.	123 ParkLane	212-555-1212	10095	100	12/23/2007	$500
2010	Movers, Inc.	123 ParkLane	212-555-1212	10095	102	7/20/2008	$100
2010	Movers, Inc.	123 ParkLane	212-555-1212	10095	103	8/25/2008	$400
2010	Movers, Inc.	123 ParkLane	212-555-1212	10095	104	9/20/2008	$200
2011	Acme Mfg, Ltd.	555 Broadway	212-566-1212	10004	105	9/20/2008	$900
2012	ALR Inc.	50 Fifth Avenue	212-999-1212	10010	101	1/5/2008	$600

If, however, the data is split into two tables (CUSTOMER and ORDER, as shown in Figure 1.4) and the customer's address needs to be updated, only one row in the CUSTOMER table needs to be updated. Furthermore, separating data this way avoids the

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

possibility of data inconsistency, whereby the data differs between the different rows. Eliminating redundancy is one of the key concepts in relational databases, and this process, referred to as *normalization*, is discussed shortly.

FIGURE 1.4 Primary and foreign key relationship between CUSTOMER and ORDER tables



CUSTOMER_ID	CUSTOMER_NAME	ADDRESS	PHONE	ZIP
2010	Movers, Inc.	123 Park Lane	212-555-1212	10095
2011	Acme Mfg, Ltd.	555 Broadway	212-566-1212	10004
2012	ALR Inc.	50 Fifth Avenue	212-999-1212	10010

ORDER_ID	CUSTOMER_ID	ORDER_DATE	TOTAL_ORDER
100	2010	12/23/2007	$500
102	2010	7/20/2008	$100
103	2010	8/25/2008	$400
104	2010	9/20/2008	$200
105	2011	9/20/2008	$900
101	2012	1/5/2008	$600

Figure 1.4 illustrates how the data is split into two tables to provide data consistency. In this example,

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

CUSTOMER_ID becomes a *foreign key* column in the ORDER table. The foreign key is the column that links the CUSTOMER and ORDER table together. You can find all orders for a particular customer by looking for the particular CUSTOMER_ID in the ORDER table. The CUSTOMER_ID corresponds to a single row in the CUSTOMER table that provides the customer-specific information.

The foreign key column CUSTOMER_ID happens to have the same column name in the ORDER table. This makes it easier to recognize the fact that the tables share common column values. Often the foreign key column and the primary key have identical column names, but it is not required. You will learn more about foreign key columns with the same and different names and how to create foreign key relationships in Chapter 12. Chapter 7, "Equijoins," teaches you how to combine results from the two tables using SQL.

You connect and combine data between tables in a relational database via common columns.

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

Overview of SQL Language Commands

You work with tables, rows, and columns using the SQL language. SQL allows you to query data, create new data, modify existing data, and delete data. Within the SQL language you can differentiate between individual sublanguages, which are a collection of individual commands.

For example, *Data Manipulation Language* (DML) commands allow you to query, insert, update, and delete data. SQL allows you to create new database structures such as tables or modify existing ones; this subcategory of SQL language commands is called the *Data Definition Language* (DDL). Using the SQL language, you can control access to the data using *Data Control Language* (DCL) commands. Table 1.1 shows you different language categories, with examples of their respective SQL commands.

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

TABLE 1.1 Overview of SQL Language Commands

DESCRIPTION	SQL COMMANDS
Data Manipulation	SELECT, INSERT, UPDATE, DELETE, MERGE
Data Definition	CREATE, ALTER, DROP, TRUNCATE, RENAME
Data Control	GRANT, REVOKE
Transaction Control	COMMIT, ROLLBACK, SAVEPOINT

One of the first statements you will use in the exercises is the SELECT command, which allows you to query data. For example, to retrieve the TITLE and PUBLISHER columns from the BOOK table, you may issue a SELECT statement such as the following:

```
SELECT title, publisher
  FROM book
```

The INSERT command lets you add new rows to a table. The next command shows you an example of an INSERT statement that adds a row to the BOOK table. The row contains the values Oracle SQL as a book title, a BOOK_ID of 1020, and a publish date of 12/2011, with Prentice Hall as the publisher.

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

```
INSERT INTO book
    (book_id, title, publisher,
publish_date) VALUES
  (1020, 'Oracle SQL', 'Prentice
Hall', '12/2011')
```

To create new tables, you use the CREATE TABLE command. The following statement illustrates how to create a simple table called AUTHOR with three columns. The first column, AUTHOR_ ID, holds numeric data; the FIRST_NAME and LAST_NAME columns contain alphanumeric character data.

```
CREATE TABLE author
    (author_id NUMBER,
    first_name VARCHAR2(30),
    last_name VARCHAR2(30))
```

You can manipulate the column definitions of a table with the ALTER TABLE command. This allows you to add or drop columns. You can also create primary and foreign key constraints on a table. Constraints enforce business rules within the database. For example, a primary key constraint can enforce the uniqueness of the AUTHOR_ID column in the AUTHOR table.

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

To grant SELECT and INSERT access to the AUTHOR table, you issue a GRANT command. It allows the user Scott to retrieve and insert data in the AUTHOR table.

```
GRANT SELECT, INSERT ON author TO
scott
```

Starting with Chapter 2, "SQL: The Basics," you will learn how to execute the SELECT command against an Oracle database; Chapter 11, "Insert, Update, and Delete," describes the details of data manipulation and transaction control; and Chapter 12 introduces you to the creation of tables and the definition of constraints to enforce the required business rules. Chapter 15, "Security," discusses how to control access to data and the various Oracle database security features.

LAB 1.1 EXERCISES

a) What is SQL, and why is it useful?

b) Try to match each of the SQL commands on the left with a verb from the list on the right.

1. CREATE	**a.** manipulate
2. UPDATE	**b.** define
3. GRANT	**c.** control

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

c) Why is it important to control access to data in a database?

d) How is data organized in a relational database?

e) Is it possible to have a table with no rows at all?

8
9

f) Figure 1.5 displays an EMPLOYEE table and a DEPARTMENT table. Identify the columns you consider to be the primary key and foreign key columns.

g) Would it be possible to insert into the EMPLOYEE table an employee with a DEPT_NO of 10?

FIGURE 1.5 EMPLOYEE and DEPARTMENT tables

EMPLOYEE

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY	DEPT_NO
230	Kyle	Hsu	$80,000	40
231	Kirsten	Soehner	$130,000	50
232	Madeline	Dimitri	$70,000	40
234	Joshua	Hunter	$90,000	20

DEPARTMENT

DEPT_NO	DEPARTMENT_NAME
20	Finance
40	Human Resources
50	Sales
60	Information Systems

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

LAB 1.1 EXERCISE ANSWERS

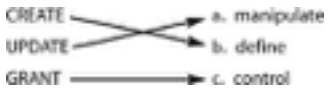
a) What is SQL, and why is it useful?

ANSWER: SQL, the Structured Query Language, is a standardized relational database access language. It is useful because it allows a user to query, manipulate, define, and control data in an RDBMS.

SQL is sanctioned by ANSI, which determines standards on all aspects of SQL, including data types. However, most relational database products, including Oracle, have their own extensions to the ANSI standard, providing additional functionality within their respective products by further extending the use of SQL. By learning SQL, you will be able to use the language on other non-Oracle databases with some minor syntax adjustments.

b) Try to match each of the SQL commands on the left with a verb from the list on the right:

ANSWER: The following shows how these commands match with the appropriate verb.



PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

DML is used to *manipulate* data, with the SELECT, INSERT, UPDATE, and DELETE commands. (Note that in some of Oracle's own documentation, the SELECT command is not part of DML but is considered Data Retrieval Language.) DDL is used to *define* objects such as tables with the CREATE, ALTER, and DROP commands. DCL is used to *control* access privileges in an RDBMS, such as with the GRANT and REVOKE commands to give or remove privileges. These SQL commands are written and executed against the database using a software program. In this workbook, Oracle's SQL Developer and SQL*Plus software tools are used to communicate these commands to the RDBMS. The use of these tools and the SQL commands is covered in Chapter 2.

c) Why is it important to control access to data in a database?

ANSWER: Data can contain sensitive information to which some users should have limited access privileges. Some users may be allowed to query certain data but not change it, while others may be allowed to add data to a

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

database but not delete it. By controlling access to data, the security of the data is assured for all users. You learn about safeguarding your data in Chapter 15.

d) How is data organized in a relational database?

ANSWER: Data is organized by placing similar pieces of information together in a table that consists of columns and rows.

For example, the data found in a library is typically organized in several ways to facilitate finding a book. Figure 1.6 shows information specific to books. The data is organized into columns and rows; the columns represent a type of data (title vs. genre), and the rows contain data. A table in a database is organized in the same way. You might call this table BOOK; it contains information related to books only. Each intersection of a column and row in a table represents a value.

FIGURE 1.6 The BOOK table

TITLE	AUTHOR	ISBN#	GENRE	LOCATION_ID
Magic Gum	Harry Smith	0-11-124456-2	Computer	D11
Desk Work	Robert Jones	0-11-223754-3	Fiction	H24
Beach Life	Mark Porter	0-11-922256-8	Junvenile	J3
From Here to There	Gary Mills	0-11-423356-5	Fiction	H24

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

Searching for a book by location might yield the excerpt of data shown in Figure 1.7. This set of columns and rows represents another database table called LOCATION, with information specific to locations in a library.

FIGURE 1.7 The LOCATION table

LOCATION_ID	FLOOR	SECTION	SHELF
D11	1	3	1
H24	2	2	3
J3	3	1	1

The advantage to storing information about books and their locations separately is that information is not repeated unnecessarily, and maintenance of the data is much easier.

For instance, two books in the BOOK table have the same LOCATION_ID, H24. If the floor, section, and shelf information were also stored in the BOOK table, this information would be repeated for each of the two book rows. In that situation, if the floor of LOCATION_ID H24 changed, both of the rows in the BOOK table would have to change. Instead, by storing the location information separately, the floor

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

information has to change only once in the LOCATION table.

10
11

The two tables (BOOK and LOCATION) have a common column between them, namely LOCATION_ID. In a relational database, SQL can be used to query information from more than one table at a time, making use of the common column they contain by performing a *join*. The join allows you to query both the BOOK and LOCATION tables to return a list of book titles together with floor, section, and shelf information to help you locate the books easily.

e) Is it possible to have a table with no rows at all?

ANSWER: Yes, it is possible, although clearly it is not very useful to have a table with no data.

f) Figure 1.5 displays an EMPLOYEE table and a DEPARTMENT table. Identify the columns you consider to be the primary key and foreign key columns.

ANSWER: The primary key of the EMPLOYEE table is EMPLOYEE_ID. The primary key of the DEPARTMENT table is DEPT_NO. The DEPT_NO is also the foreign key column of the

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

EMPLOYEE table and is common between the two tables.

In the DEPT_NO column of the EMPLOYEE table you can *only* enter values that exist in the DEPARTMENT table. The DEPARTMENT table is the parent table from which the child table, the EMPLOYEE table, gets its DEPT_NO values.

g) Would it be possible to insert into the EMPLOYEE table an employee with a DEPT_NO of 10?

ANSWER: Only valid primary key values from the DEPARTMENT table are allowed in the EMPLOYEE table's foreign key DEPT_NO column. You cannot enter a DEPT_NO of 10 in the EMPLOYEE table if such a value does not exist in the DEPARTMENT table. Establishing a foreign key relationship highlights the benefit of *referential integrity*.

If the foreign key constraint between the two tables is temporarily disabled or no foreign key constraint exists, such an "invalid" entry is possible. We will discuss more on this topic in later chapters.

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

Note that the DEPARTMENT table contains one row with the department number 60, which does not have any corresponding employees. The referential integrity rule allows a parent without child(ren) but does not allow a child without a parent because this would be considered an orphan row. You will learn how to establish primary key and foreign key relationships in Chapter 12.

Lab 1.1 Quiz

In order to test your progress, you should be able to answer the following questions.

1) A university's listing of students and the classes they are enrolled in is an example of a database system.

_____**a)** True

_____**b)** False

2) A table must always contain both columns and rows.

_____**a)** True

_____**b)** False

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

3) Referential integrity ensures that each value in a foreign key column of the child table links to a matching primary key value in the parent table.

_____**a)** True

_____**b)** False

4) A null value means that that the value is unknown.

_____**a)** True

_____**b)** False

ANSWERS APPEAR IN APPENDIX A.

LAB 1.2 Data Normalization and Table Relationships

LAB OBJECTIVES

After this lab, you will be able to:

- Identify Data Normalization Rules and Table Relationships
- Read a Database Schema Diagram
- Understand the Database Development Context

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

Although this is a book about SQL, you must know the basic concepts, terminology, and issues involved in database design to be able to understand why tables are organized in specific ways. This lab introduces you to the practical aspects of designing tables and determining their respective relationships to each other.

Data Normalization

The objective of *normalization* is to eliminate redundancy in tables, therefore avoiding any future data manipulation problems. There are a number of different rules for minimizing duplication of data, which are formulated into the various *normal forms*.

The rules verify that the columns you placed in the tables do in fact belong there. You design your tables, the appropriate columns, and the matching primary and foreign keys to comply with these rules. This process is called *normalization*. The normalization rules will be quite intuitive after you have read through the examples in this lab. Although there are many normalization rules, the *five normal forms* and the *Boyce–Codd normal form* (BCNF) are the most widely accepted. This lab discusses the first three normal forms because programmers and analysts typically don’t bother normalizing beyond third

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

normal form—although experienced database designers sometimes do.

FIRST NORMAL FORM

For a table to be in *first normal form*, all repeating groups must be removed and placed in a new table. The example in Figure 1.8 illustrates the use of repeating groups in the BOOK table. The table has multiple locations of warehouses across the country where the title is stocked. The location is listed in three columns as LOCATION_1, LOCATION_2, and LOCATION_3.

FIGURE 1.8 Repeating group in the BOOK table

BOOK_ID	TITLE	RETAIL PRICE	LOCATION_1	LOCATION_2	LOCATION_3
1010	The Invisible Force	29.95	New York	San Francisco	
1011	Into The Sky	39.95	Chicago		
1012	Making It Possible	59.95	Miami	Austin	New York

Imagine a scenario in which you have more than three locations for a book. To avoid problems, the database designer will move the location information to a separate table named BOOK_LOCATION, as illustrated in Figure 1.9. This design is more flexible and allows the storing of books at an unlimited number of locations.

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

FIGURE 1.9 Tables in first normal form

BOOK

BOOK_ID	TITLE	RETAIL PRICE
1010	The Invisible Force	29.95
1011	Into The Sky	39.95
1012	Making It Possible	59.95

BOOK_LOCATION

BOOK_ID	LOCATION
1010	New York
1010	San Francisco
1011	Chicago
1012	Miami
1012	Austin
1012	New York

SECOND NORMAL FORM

Second normal form states that all nonkey columns must depend on the entire primary key. It applies only to tables that have composite primary keys. Figure 1.10 shows the BOOK_AUTHOR table with both the BOOK_ID and AUTHOR_ID as the composite primary key. In this example, authors with the ID 900 and 901 coauthored the book with the ID 10002. If you add the author's phone number to the table, the second normal

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

form is violated because the phone number is dependent only on the AUTHOR_ID, not on BOOK_ID. ROYALTY_ SHARE is dependent completely on the combination of both columns because the percentage of the royalty varies from book to book and is split among authors.

FIGURE 1.10 Violation of second normal form in the BOOK_AUTHOR table

BOOK_ID	AUTHOR_ID	ROYALITY_SHARE	AUTHOR_PHONE_NO
10001	900	100	212-555-1212
10002	901	75	901-555-1212
10002	900	25	212-555-1212
10003	902	100	899-555-1212

THIRD NORMAL FORM

The *third normal form* goes a step further than the second normal form: It states that every nonkey column must be a fact about the primary key. The third normal form is quite intuitive. Figure 1.11 shows a table that violates third normal form. The PUBLISHER_PHONE_NO column is not dependent on the primary key column BOOK_ID but on the PUBLISHER_NAME column. Therefore, the

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

PUBLISHER_PHONE_NO column should not be part of the BOOK table.

FIGURE 1.11 Violation of third normal form

BOOK_ID	TITLE	PUBLISHER_NAME	PUBLISH_DATE	PUBLISHER_PHONE_NO
1010	The Invisible Force	Literacy Circle	12/07	801-111-1111
1011	Into The Sky	Prentice Hall	10/08	999-888-1212
1012	Making It Possible	Life Books	2/06	777-555-1212
1013	Wonders of the World	Literacy Circle	2/06	801-111-1111

Instead, the publisher's phone number should be stored in a separate table called PUBLISHER. This has the advantage that when a publisher's phone number is updated, it needs to be updated in only one place rather than at all occurrences of this publisher in the BOOK table. Removing the PUBLISHER_PHONE_NO column eliminates redundancy and avoids any possibilities of data inconsistencies (see Figure 1.12).

The BOOK table can also benefit by introducing a surrogate key, such as the PUBLISHER_ID column. Such a key is not subject to changes and is easily referenced in any additional tables that may need to refer to data about the publisher.

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

BOYCE–CODD NORMAL FORM (BCNF), FOURTH NORMAL FORM, AND FIFTH NORMAL FORM

BCNF is an elaborate version of the third normal form and deals with deletion anomalies. The *fourth normal form* tackles potential problems when three or more columns are part of the unique identifier and their dependencies to each other. The *fifth normal form* splits the tables even further apart to eliminate all redundancy. These different normal forms are beyond the scope of this book; for more details, please consult one of the many excellent books on database design.

15

16

FIGURE 1.12 Tables in third normal form

BOOK

BOOK_ID	TITLE	PUBLISHER_ID	PUBLISH_DATE
1010	The Invisible Force	1	12/07
1011	Into The Sky	2	10/08
1012	Making It Possible	3	2/06
1013	Wonders of the World	1	2/06

PUBLISHER

PUBLISHER_ID	PUBLISHER_NAME	PUBLISHER_PHONE_NO
1	Literacy Circle	80-111-1111
2	Prentice Hall	999-888-1212
3	Life Books	777-555-1212

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

Table Relationships

When two tables have a common column or columns, the tables are said to have a *relationship* between them. The *cardinality* of a relationship is the actual number of occurrences between them. We will explore one-to-many, one-to-one, and many-to-many relationships.

ONE-TO-MANY RELATIONSHIP (1:M)

Figure 1.13 shows the CUSTOMER table and the ORDER table. The common column is CUSTOMER_ID. The link between the two tables is a *one-to-many* relationship, the most common type of relationship. This means that one individual customer can have many order rows in the ORDER table. This relationship represents the business rule that "one customer can place one or many orders (or no orders)." Reading the relationship in the other direction, an order is associated with only one customer row (or no customer rows). In other words, "each order may be placed by one and only one customer."

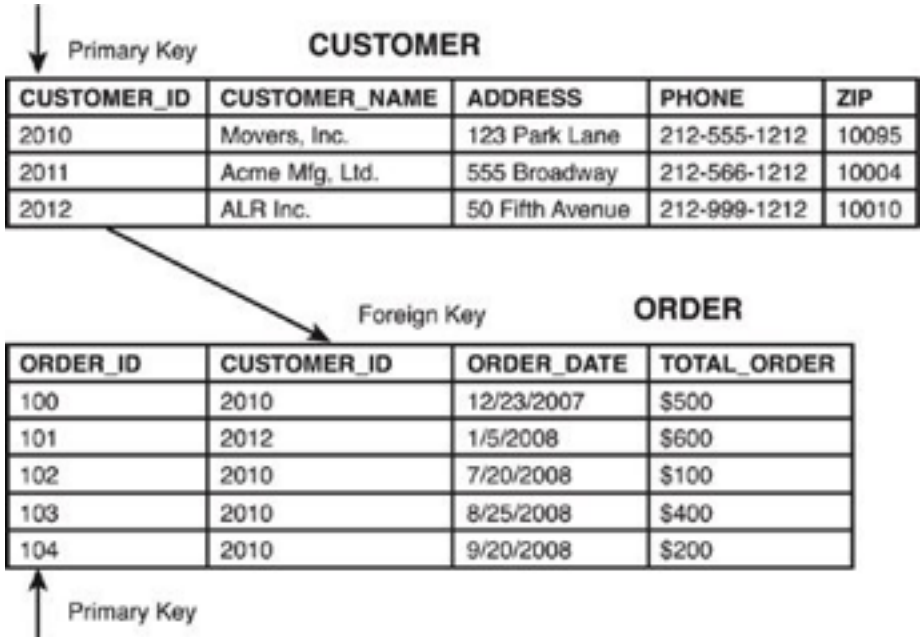
ONE-TO-ONE RELATIONSHIP (1:1)

One-to-one relationships exist in the database world, but they are not typical because most often data from both

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

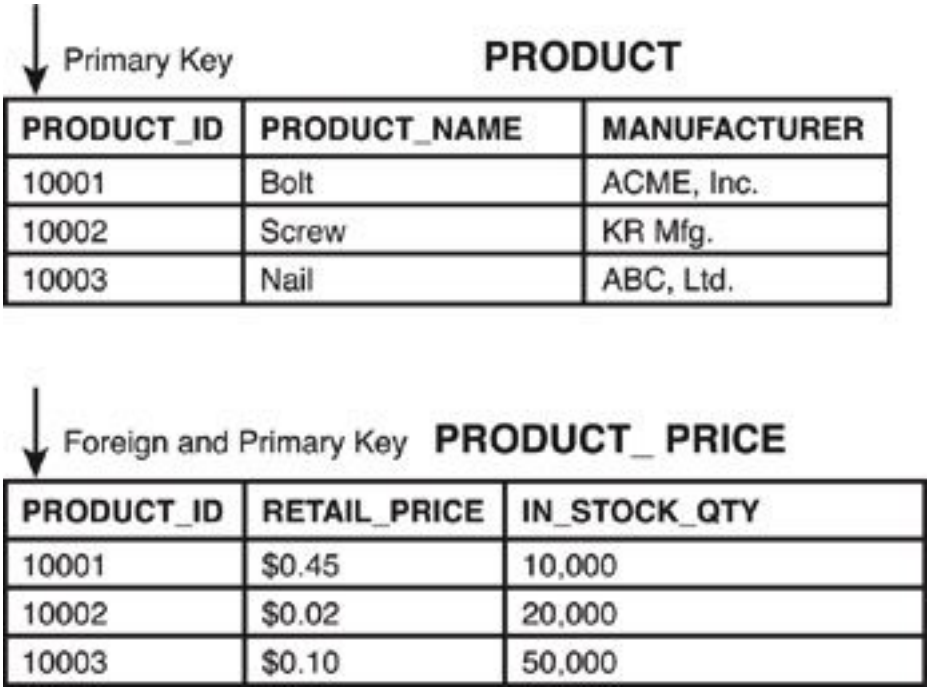
tables are combined into one table for simplicity. Figure 1.14 shows an example of a *one-to-one* relationship between the PRODUCT table and the PRODUCT_PRICE table. For every row in the PRODUCT table, you may find only one matching row in the PRODUCT_PRICE table. And for every row in the PRODUCT_PRICE table, there is one matching row in the PRODUCT table. If the two tables are combined, the RETAIL_PRICE and IN_STOCK_QTY columns can be included in the PRODUCT table.

FIGURE 1.13 One-to-many relationship example between CUSTOMER and ORDER tables



PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

FIGURE 1.14 One-to-one relationship example



PRODUCT_ID	PRODUCT_NAME	MANUFACTURER
10001	Bolt	ACME, Inc.
10002	Screw	KR Mfg.
10003	Nail	ABC, Ltd.

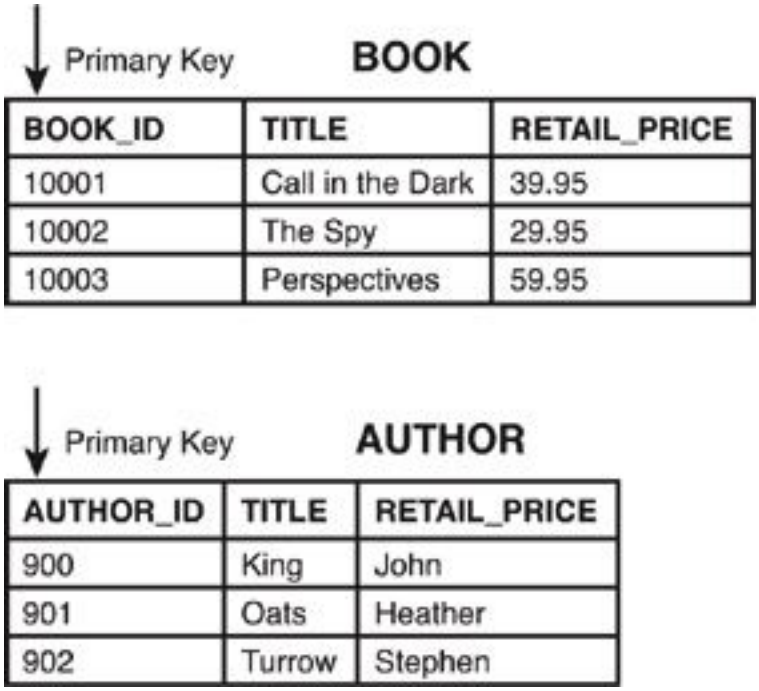
PRODUCT_ID	RETAIL_PRICE	IN_STOCK_QTY
10001	$0.45	10,000
10002	$0.02	20,000
10003	$0.10	50,000

MANY-TO-MANY RELATIONSHIP (M:M)

The examination of Figure 1.15 reveals a *many-to-many* relationship between the BOOK and AUTHOR tables. One book can have one or more authors, and one author can write one or more books. The relational database model requires the resolution of many-to-many relationships into one-to-many relationship tables.

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

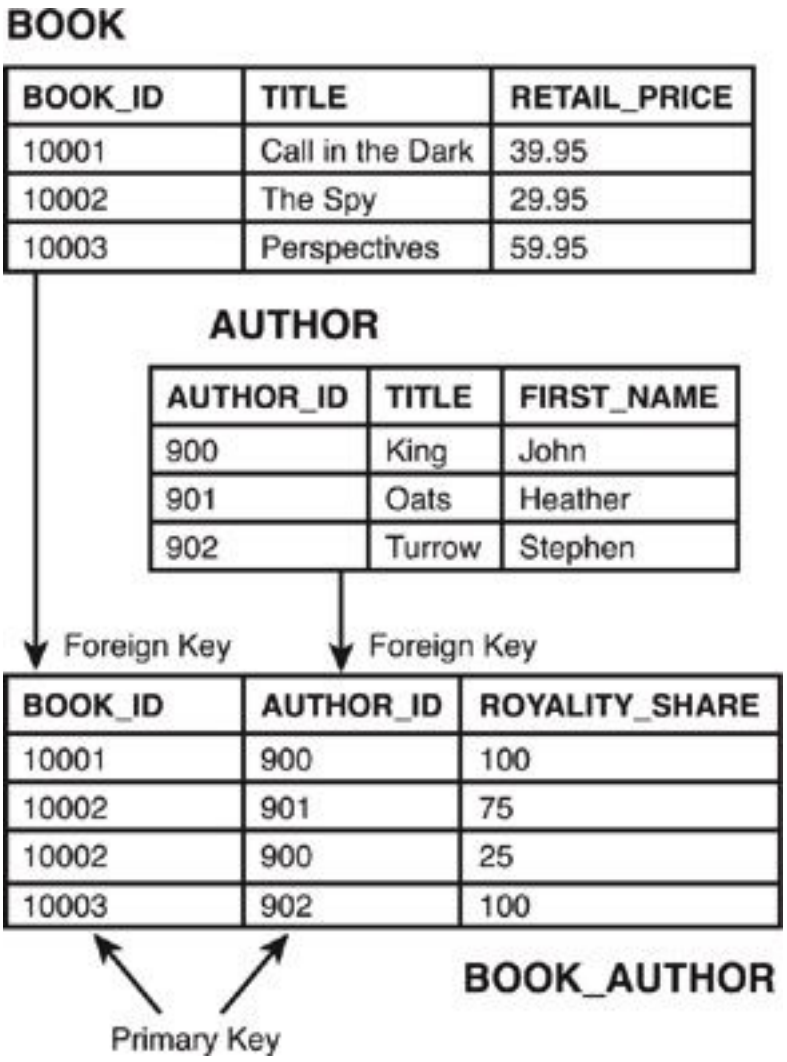
FIGURE 1.15 Many-to-many relationship example



The solution in this example is achieved by creating an *associative table* (also called an *intersection table*) via the BOOK_AUTHOR table. Figure 1.16 shows the columns of this table.

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

FIGURE 1.16 Associative BOOK_AUTHOR table that resolves the many-to-many relationship



18

19

The BOOK_AUTHOR table lists the individual author(s) for each book and shows, for a particular author, the book(s) he or she wrote. The primary key of

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

the BOOK_AUTHOR table is the combination of both columns: the BOOK_ID column and the AUTHOR_ID column. These two columns represent the concatenated primary key that uniquely identifies a row in the table. As you may recall from Lab 1.1, multicolumn primary keys are referred to as a *composite*, or concatenated, primary key. In addition, the BOOK_AUTHOR table has AUTHOR_ID and the BOOK_ID as two individual foreign keys linking back to the AUTHOR table and the BOOK table, respectively.

The BOOK_AUTHOR table contains an additional column, the ROYALTY_SHARE column, which identifies the royalty percentage for each author for an individual book. When there are multiple authors, the percentage of the royalty is split; in the case of a sole author, the share is 100%. This column is appropriately located in the BOOK_AUTHOR table as the values are relevant for the combination of the BOOK_ID and AUTHOR_ID. These two columns uniquely identify both a book and an author and the respective percentage share of the royalty.

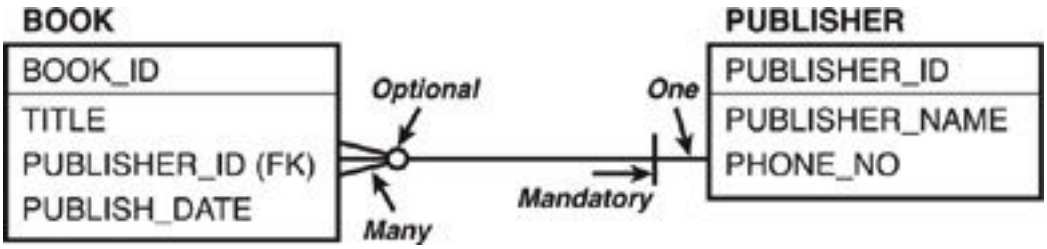
Database Schema Diagrams

For clarity of meaning and conceptual consistency, it is useful to show table relationships using *database schema*

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

diagrams. There are a number of standard notations for this type of diagram. Figure 1.17 illustrates one of the ways to graphically depict the relationship between tables. The convention used in this book for a one-to-many relationship is a line with a "crow's foot" (fork) on one end indicating the "many" side of the relationship; at the other end, a "single line" depicts the "one" side of the relationship. You will see the use of the *crow's-foot notation* throughout this book. Software diagramming programs that support the graphical display of relational database models often allow you to choose your notation preference.

FIGURE 1.17 Crow's-foot notation



CARDINALITY AND OPTIONALITY

The cardinality expresses the ratio of a parent and child table from the perspective of the parent table. It describes how many rows you may find between the two tables for a given primary key value.

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

Graphical relationship lines also indicate the *optionality* of a relationship, whether a row is required or not (mandatory or optional). Specifically, optionality shows whether one row in a table can exist without a row in the related table.

Figure 1.17 illustrates a one-to-many relationship between the PUBLISHER (parent) and the BOOK (child). Examining the relationship line on the "many" end, you notice a circle identifying the *optional relationship* and a crow's foot indicating "many ." The symbols indicate that a publisher *may* publish zero, one, or many books. You use the word *may* to indicate that the relationship is *optional*, and this allows a publisher to exist without a corresponding value in the BOOK table.

The relationship line also reads the other way. The solid line on the PUBLISHER end of the line indicates the "one" side, and a vertical bar intersects it. This bar identifies a *mandatory relationship*. You read this direction of the relationship as "One book *must* be published by one and only one publisher." This means a row in the BOOK table must always have the PUBLISHER_ID value filled in.

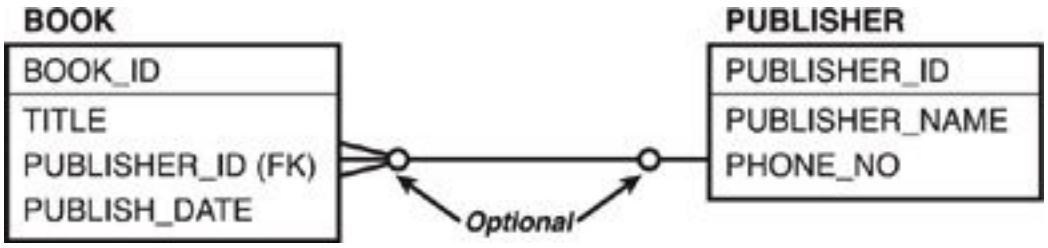
PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

When a column is defined as NOT NULL, it must always contain a value. When a column in a row is defined as allowing NULL values, it means that a column does not need to contain a value.

The "(FK)" symbol next to the PUBLISHER_ID column on the BOOK table indicates that this is a foreign key column. In this diagram, the primary key is separated from the other columns with a line; you can see that BOOK_ID and the PUBLISHER_ID are the primary keys, or unique identifiers.

Figure 1.18 shows an optional relationship on both sides; a book may be published by zero or one publisher. Effectively, this means the value in the PUBLISHER_ID column in BOOK is optional. Reading the relationship from PUBLISHER, you can say that "one publisher may publish zero, one, or many books" (which is identical to Figure 1.17).

FIGURE 1.18 Optional relationship notation on both sides



PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

Figure 1.19 shows some sample data that illustrates the optional relationships. Review the row with the BOOK_ID value of 1012; it does not contain a PUBLISHER_ID column. This means it's possible to have an entry in the BOOK table without the publisher being specified. Furthermore, the BOOK table contains multiple entries for the PUBLISHER_ID 1. This indicates that many books can be published by the same publisher.

On the PUBLISHER table, you see the PUBLISHER_ID 3, which does not exist in the BOOK table. A publisher can exist in the table without having an entry in the BOOK table.

20

21

FIGURE 1.19 Sample data for BOOK and PUBLISHER, illustrating optional relationships

BOOK

BOOK_ID	TITLE	PUBLISHER_ID	PUBLISH_DATE
1010	The Invisible Force	1	12/07
1011	Into The Sky	2	10/08
1012	Making It Possible		2/06
1013	Wonders of the World	1	2/06

PUBLISHER

PUBLISHER_ID	PUBLISHER_NAME	PUBLISHER_PHONE_NO
1	Literacy Circle	801-111-1111
2	Prentice Hall	999-888-1212
3	Life Books	777-555-1212

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

REAL-WORLD BUSINESS PRACTICE

You typically see only two types of relationships: first, mandatory on the "one" side and optional on the "many" end, as in Figure 1.17; and second, optional on both ends, as in Figure 1.18. Only rarely do you find other types of relationships. For example, mandatory relationships on both sides are infrequently implemented; this means that rows must be inserted in both tables simultaneously. Occasionally, you find one-to-one relationships, but most often, the columns from both tables are combined into one table. Many-to-many relationships are not allowed in the relational database; they must be resolved via an associative, or intersection, table into one-to-many relationships.

LABELING RELATIONSHIPS

To clarify and explain the nature of a relationship on a diagram, it is useful to add on the relationship line a label or name with a verb. Figure 1.20 shows an example of a labeled relationship. For the utmost clarity, a relationship should be labeled on both sides. You then read it as: "One PUBLISHER may publish zero, one, or many BOOKs; and one BOOK must be published by one and only one PUBLISHER."

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

Or, in the example shown in Figure 1.18, the label would need to read "One PUBLISHER may publish zero, one, or many BOOKs; and one BOOK may be published by one and only one PUBLISHER." This kind of labeling makes the relationship perfectly clear and states the relationship in terms that a business user can understand.

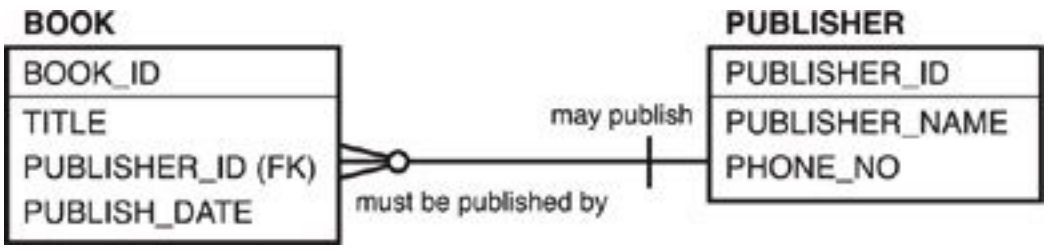
IDENTIFYING AND NONIDENTIFYING RELATIONSHIPS

In an *identifying relationship*, the primary key is propagated to the child entity as part of the primary key. This is in contrast to a *nonidentifying relationship*, in which the foreign key becomes one of the nonkey columns. A nonidentifying relationship may accept a null value in the foreign key column.

21

22

FIGURE 1.20 Labeled relationship between BOOK and PUBLISHER

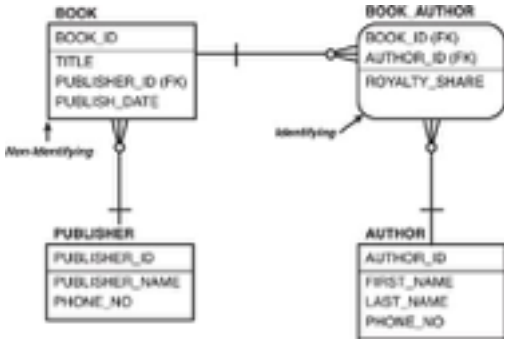


PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

If a graphical representation of a table's box has *rounded edges*, it means that the relationship is *identifying*. Effectively, one of the foreign keys became the primary key or part of the primary key. In the case of the BOOK_AUTHOR table shown in Figure 1.21, both foreign key columns constitute the primary key, and both columns may not be null because a primary key is never null.

The many-to-many relationship between the BOOK and AUTHOR tables was resolved with the associative table called BOOK_AUTHOR. The combination of the two foreign keys forms the unique primary key. This primary key also ensures that there will not be any duplicate entries. In other words, it is not possible to enter the combination of AUTHOR_ID and BOOK_ID twice.

FIGURE 1.21 Identifying and nonidentifying relationships



PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

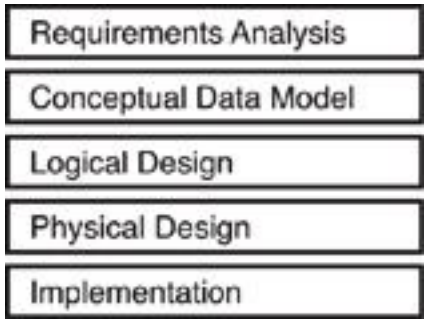
The relationship between the PUBLISHER and BOOK tables is nonidentifying, as indicated by the sharp edges. The foreign key column PUBLISHER_ID is not part of the primary key of the BOOK table. The foreign key columns of a nonidentifying relationship may be either NULL or NOT NULL. In this instance, you can determine if a null is allowed by checking whether the relationship is optional or mandatory. Although the foreign key column allows null values in nonidentifying relationships, here the relationship depicts a single bar on the relationship line. Effectively, for every row in the BOOK table, there must be a corresponding row in the PUBLISHER table, and the PUBLISHER_ID column of the BOOK table cannot be null.

Database Development Context

Now that you are familiar with the some of the terminology and core concepts of relational databases, you are ready to learn about how all this information fits into the context of database development. From the initial idea of an application until the final system implementation, the data model is continuously refined. Figure 1.22 indicates the essential phases of the development project with respect to the database.

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

FIGURE 1.22 Database development and design phases



REQUIREMENTS ANALYSIS

The database development and design process starts with gathering data requirements that identify the needs and wants of the users. One of the outputs of this phase is a list of individual data elements that need to be stored in the database.

CONCEPTUAL DATA MODEL

The *conceptual data model* logically groups the major data elements from the requirements analysis into individual *entities*. An *entity* is something of significance for which you need to store data. For example, all data related to books, such as the title, publish date, and retail price, are placed in the book entity. Data elements such as the author's name and

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

address are part of the author entity. The individual data elements are referred to as *attributes*.

You designate a *unique identifier*, or *candidate key*, that uniquely distinguishes a row in the entity. In this conceptual data model, we use the terms *entity*, *attribute*, and *candidate key* or *unique identifier* instead of *table*, *column*, and *primary key*, respectively.

Noncritical attributes are not included in the model to emphasize the business meaning of those entities, attributes, and relationships. Many-to-many relationships are acceptable and not resolved. The diagram of the conceptual model is useful to communicate the initial understanding of the requirements to business users. The conceptual model gives no consideration to the implementation platform or database software. Many projects skip the conceptual model and go directly to the logical model.

23
24

LOGICAL DATA MODEL

The purpose of the *logical data model* is to show that all the entities, their respective attributes, and the relationship between entities represent the business requirements, without considering technical issues. The focus is entirely on business problems and considers a

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

design that accommodates growth and change. The entities and attributes require descriptive names and documentation of their meaning. Labeling and documenting the relationships between entities clarify the business rules between them.

The diagram may show the data type of an attribute in general terms, such as text, number, and date. In many logical design models, you find foreign key columns identified; in others, they are implied.

The complete model is called the *logical data model*, or *entity relationship diagram* (ERD). At the end of the analysis phase, the entities are fully normalized, the unique identifier for each entity is determined, and any many-to-many relationships are resolved into associative entities.

PHYSICAL DATA MODEL

The *physical data model*, also referred to as the *database schema diagram*, is a graphical model of the physical design implementation of the database. This physical schema diagram is what the programmers and you use to learn about the database and the relationship between the tables. In Lab 1.3 you will be introduced to

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

the STUDENT schema diagram used throughout this book.

This physical data model is derived from the fully normalized logical model. Before the actual implementation (installation) of the physical data model in the database, multiple physical data models may exist. They represent a variety of alternative physical database designs that consider the performance implications and application constraints. One of the physical design models will be implemented in the database. The schema diagram graphically represents the chosen implemented physical data model; it is specific to a particular RDBMS product such as Oracle.

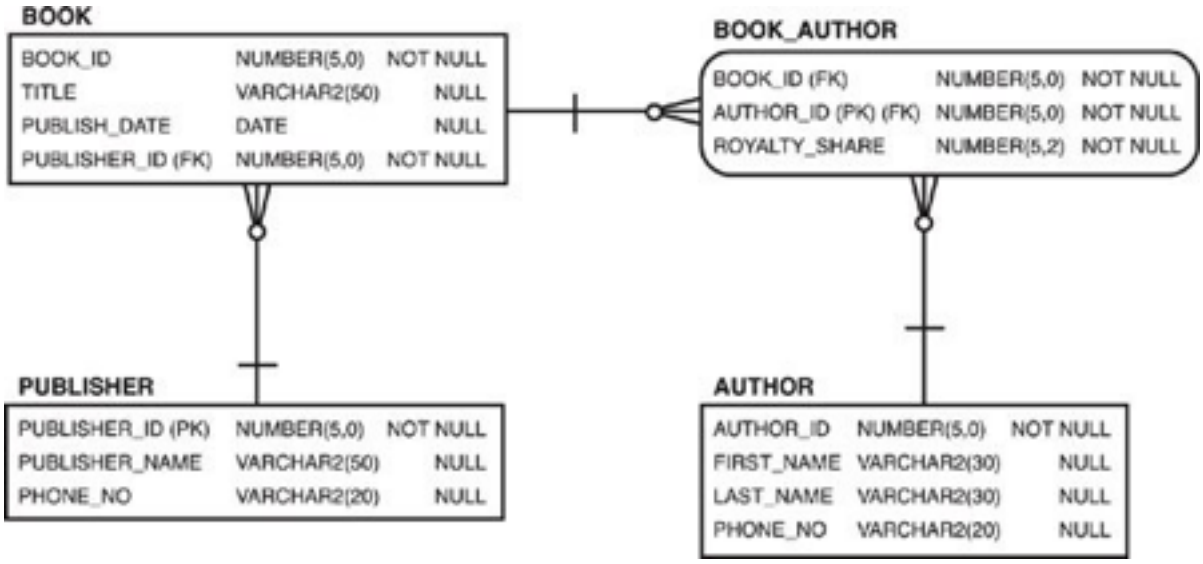
Figure 1.23 depicts the schema diagram of the book publishing database discussed in this chapter. It shows the structure of the tables with their respective columns, and it illustrates the relationships between the tables.

The physical data model has different terminology than the conceptual or logical data model. The physical data model refers to tables instead of entities; the individual pieces of data are columns instead of attributes in the logical model. This diagram graphically depicts how the physical database tables are defined, including the individual column's data type and length and whether

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

they are mandatory (that is, require a value) or allow nulls.

FIGURE 1.23 Book publishing database schema diagram



TRANSFER FROM LOGICAL TO PHYSICAL MODEL

The transfer from the logical to the physical models—which ultimately means the actual implementation in a database as tables, columns, primary keys, foreign keys, indexes, and so on—requires a number of steps and considerations. The entities identified in the logical data model are resolved to physical tables; the entity name is often identical to the table name. Some designers use

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

singular names for entities and plural names for tables; others abbreviate the entity names when implementing the physical model to follow certain business naming standards. Frequently, the physical data model includes additional tables for specific technical implementation requirements and programming purposes, such as a report queue table or an error log table.

As mentioned previously, attributes become columns, with names being either identical or following business naming conventions and abbreviations. The columns are associated with the database software vendor's specific data types, which consider valid column lengths and restrictions. Individual data entry formats are determined (for example, phone numbers must be in numeric format, with dashes between). Rules for maintaining data integrity and consistency are created, and physical storage parameters for individual tables are determined. You will learn about these and many other aspects of creating these restrictions in Chapter 12. Sometimes additional columns are added that were not in the logical design, with the purpose of storing precalculated values; this is referred to as *denormalization*, which we discuss shortly.

Another activity that occurs in the physical data design phase is the design of indexes. *Indexes* are database

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

objects that facilitate speedy access to data with the help of a specific column or combination of columns of a table. Placing indexes on tables is necessary to optimize efficient query performance. However, indexes can have the negative impact of requiring additional time for insert, update, or delete operations. Balancing the trade-offs with the advantages requires careful consideration of these factors, including knowledge in optimizing SQL statements and an understanding of the features of a particular database version. You will learn more about different types of indexes and the success factors of a well-placed index strategy in Chapter 13, "Indexes, Sequences, and Views."

Database designers must be knowledgeable and experienced in many aspects of programming, design, and database administration to fully understand how design decisions affect cost, performance, system interfaces, programming effort, and future maintenance.

Poor physical database design is very costly and difficult to correct.

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

You might wonder how the graphical models you see in this book are produced. Specific software packages allow you to visually design the various models, and they allow you to display different aspects of it, such as showing only table names or showing table names, columns, and their respective data types. Many of these tools even allow you to generate the DDL SQL statements to create the tables. For a list of software tools that allow you to visually produce these diagrams, see Appendix H, "Resources."

DENORMALIZATION

Denormalization is the act of adding redundancy to the physical database design. Typically, logical models are fully normalized or at least in third normal form. When designing the physical model, database designers must weigh the benefit of eliminating all redundancy, with data split into many tables, against potentially poor performance when these many tables are joined.

Therefore, database designers, also called *database architects*, sometimes purposely add redundancy to their physical design. Only experienced database designers should denormalize. Increasing redundancy may greatly increase the overall programming effort because now

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

many copies of the same data must be kept in sync; however, the time it takes to query data may be reduced.

In some applications, particularly data warehousing applications, where massive amounts of detailed data are stored and summarized, denormalization is required. *Data warehouse applications* are database applications that benefit users who need to analyze large data sets from various angles and use this data for reporting and decision-making purposes. Typically, the source of the data warehouse is historical transaction data, but it can also include data from various other sources for the purpose of consolidating data. For example, the purchasing department of a supermarket chain could determine how many turkeys to order for a specific store the week before Thanksgiving or use the data to determine what promotional offers have the largest sales impact on stores with certain customer demographics.

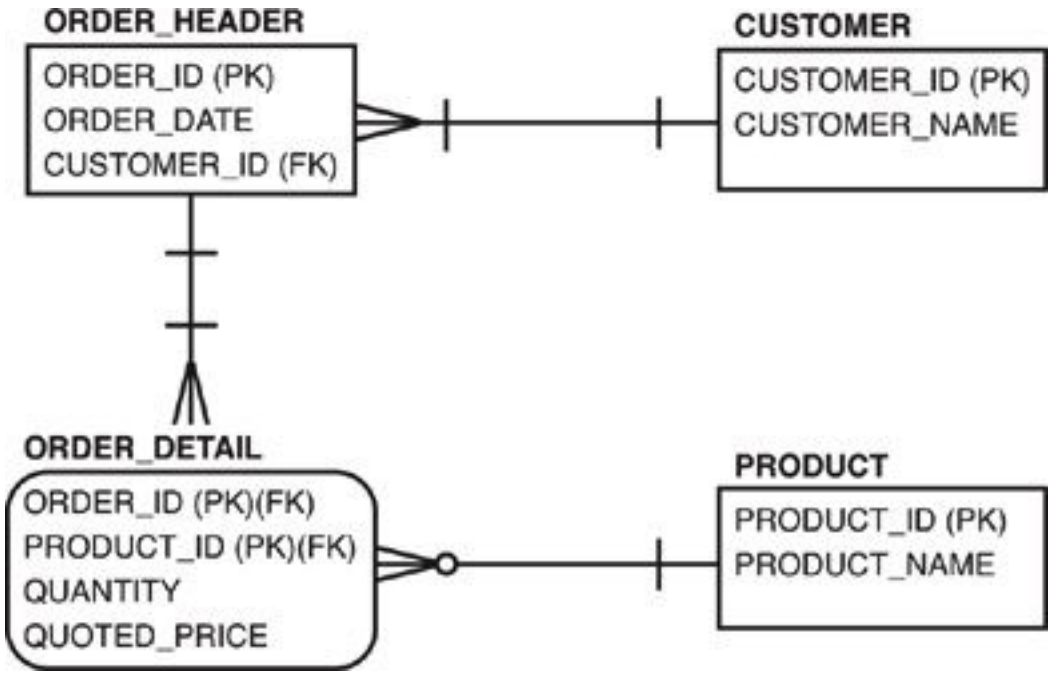
The primary purpose of a data warehouse is to query, report, and analyze data. Therefore, redundancy is encouraged and necessary for queries to perform efficiently.

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

LAB 1.2 EXERCISES

a) Describe the nature of the relationship between the ORDER_HEADER table and the ORDER_DETAIL table in Figure 1.24.

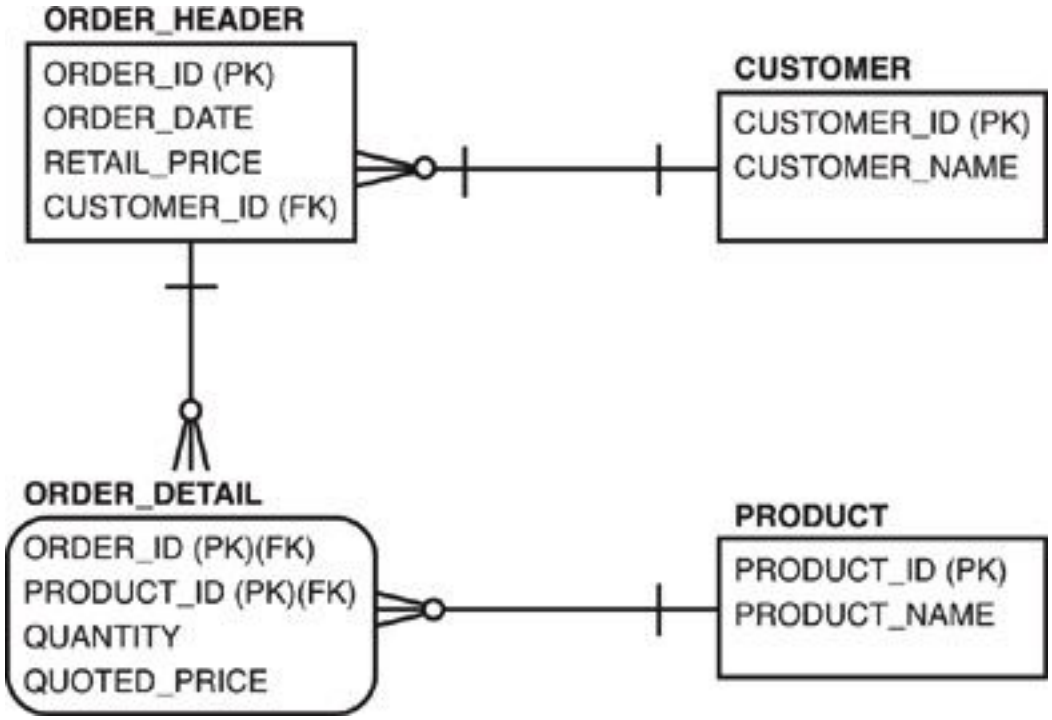
FIGURE 1.24 Order tables



b) One of the tables in Figure 1.25 is not fully normalized. Which normal form is violated? Draw a new diagram.

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

FIGURE 1.25 A table that is not fully normalized



27

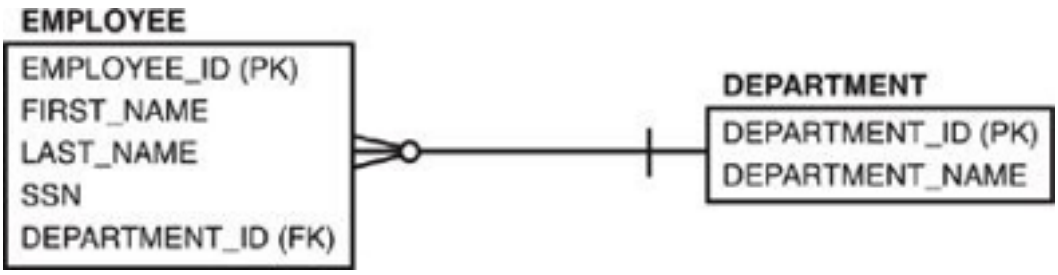
28

c) How would Figure 1.25 need to be changed to add information about the sales representative who took the order?

d) How would Figure 1.26 need to be changed if an employee does not have to belong to a department?

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

FIGURE 1.26 EMPLOYEE and DEPARTMENT relationship



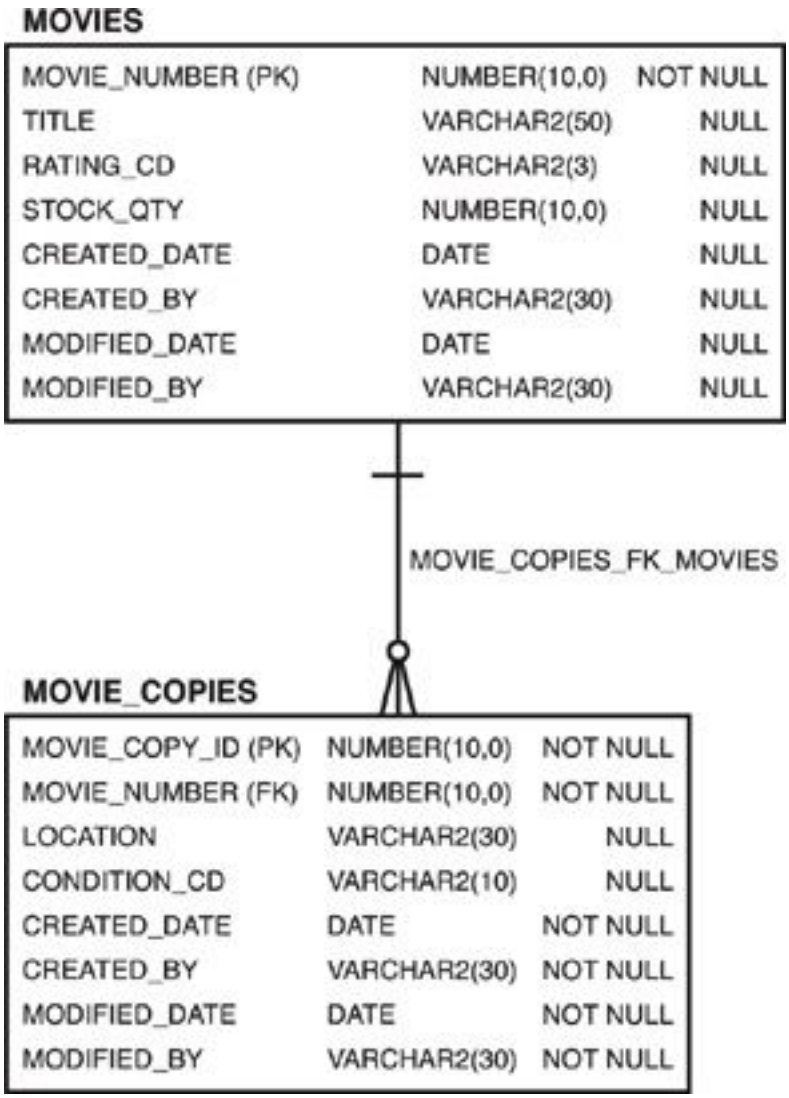
e) Based on Figure 1.26, would the Social Security number (SSN) column be a better primary key column than the EMPLOYEE_ID column?

f) Figures 1.27 and 1.28 depict the logical and physical model of a fictional movie rental database. What differences exist between the entity relationship diagram and the physical schema diagram?

FIGURE 1.27 Logical data model

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

FIGURE 1.28 Physical data model



28

29

LAB 1.2 EXERCISE ANSWERS

a) Describe the nature of the relationship between the ORDER_HEADER table and the ORDER_DETAIL table in Figure 1.24.

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

ANSWER: The relationship depicts a mandatory one-to-many relationship between the ORDER_HEADER and ORDER_DETAIL tables. The ORDER_HEADER table contains data found only once for each order, such as ORDER_ID, CUSTOMER_ID, and ORDER_DATE. The ORDER_DETAIL table holds information about the individual order lines of an order. One row in the ORDER_HEADER table must have one or many order details. One ORDER_DETAIL row must have one and only one corresponding row in the ORDER_HEADER table.

MANDATORY RELATIONSHIP ON BOTH ENDS

The mandatory relationship from ORDER_HEADER to ORDER_ DETAIL indicates that a row in the ORDER_HEADER table cannot exist unless a row in ORDER_DETAIL is created simultaneously. This is a "chicken and egg" problem, whereby a row in the ORDER_HEADER table cannot be created without an ORDER_DETAIL row and vice versa. In fact, it really doesn't matter as long as you create the rows within one

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

transaction. Furthermore, you must make sure that every row in the ORDER_HEADER table has at least one row in the ORDER_DETAIL table and that each row in the ORDER_DETAIL table has exactly one corresponding row in the ORDER_HEADER table. There are various ways to physically implement this relationship.

Another example of a mandatory relationship on the figure is the relationship between ORDER_HEADER and CUSTOMER. You can see the bar on the "many side" of the relationship as an indication for the mandatory row. That means a customer must have placed an order before a row in the CUSTOMER table is saved, and an order can be placed only by a customer.

However, for most practical purposes, a mandatory relationship on both ends is rarely implemented unless there is a very specific and important requirement.

NO DUPLICATES ALLOWED

On the previous diagrams, such as Figure 1.24, you noticed that some foreign keys are part of the primary key. This is frequently the case in associative entities; in this particular example, it requires the combination of ORDER_ID and PRODUCT_ID to be unique. Ultimately, the effect is that a single order containing the

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

same product twice is not allowed. Figure 1.29 lists sample data in the ORDER_DETAIL table for ORDER_ID 345 and PRODUCT_ID P90, which violates the primary key. Instead, you must create one order with a quantity of 10 or create a second order with a different ORDER_ID. You will learn about how Oracle responds with error messages when you attempt to violate the primary key constraint and other types of constraints in Chapter 11.

b) One of the tables in Figure 1.25 is not fully normalized. Which normal form is violated? Draw a new diagram.

ANSWER: The third normal form is violated on the ORDER_HEADER table. The RETAIL_PRICE column belongs to the PRODUCT table instead (see Figure 1.30).

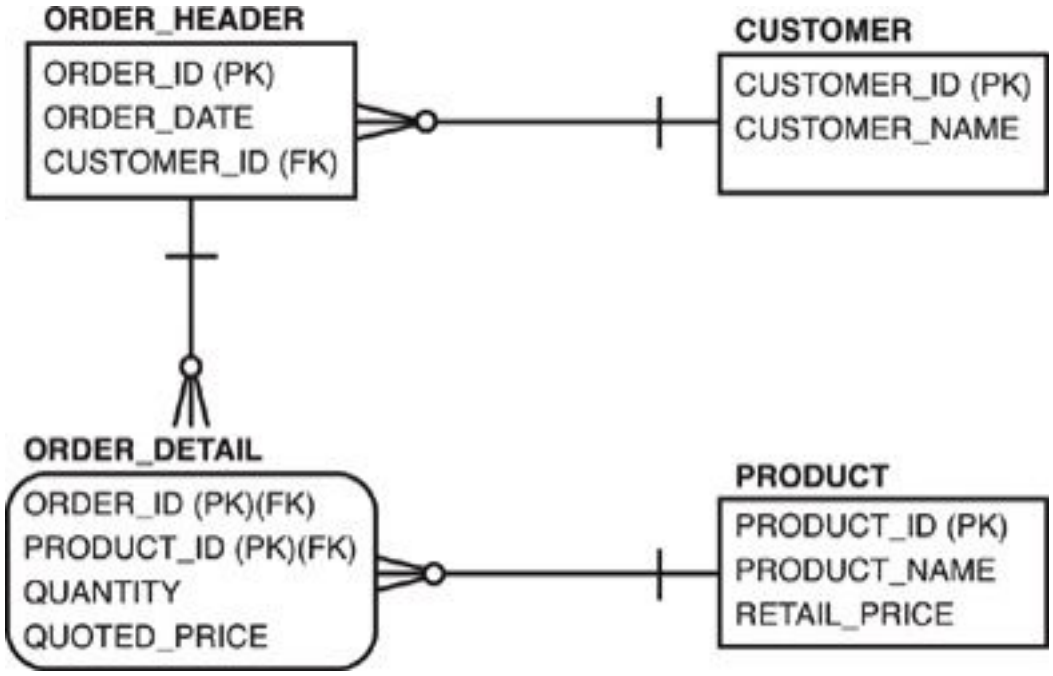
FIGURE 1.29 Sample data of the ORDER_DETAIL table

ORDER_DETAIL

ORDER_ID	PRODUCT_ID	QUANTITY	QUOTED_PRICE
123	P90	5	$50
234	S999	9	$12
345	P90	7	$50
345	X85	3	$10
345	P90	3	$50

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

FIGURE 1.30 Fully normalized tables



Third normal form states that every nonkey column must be a fact about the primary key column, which is the ORDER_ID column in the ORDER_HEADER table. This is clearly not the case with the RETAIL_PRICE column as it is not a fact about ORDER_HEADER and does not depend on ORDER_ID; it is a fact about PRODUCT. The QUOTED_PRICE column is included in the ORDER_DETAIL table because the price may vary over time, from order to order, and from customer to customer. (If you want to track any

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

changes in the retail price, you might want to create a separate table, called PRODUCT_PRICE_HISTORY, to keep track of the retail price per product and the effective date of each price change.) Table 1.2 provides a review of the normal forms.

TABLE 1.2 The Three Normal Forms

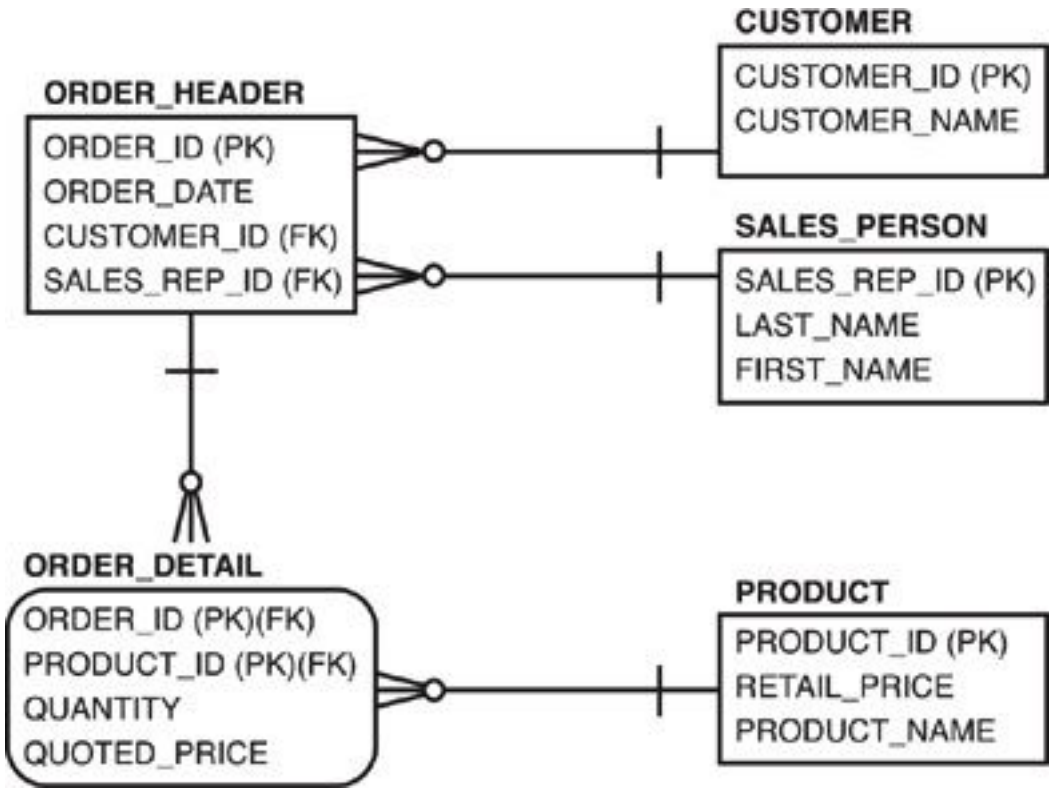
DESCRIPTION	RULE
First normal form (1NF)	No repeating groups are permitted.
Second normal form (2NF)	No partial key dependencies are permitted.
Third normal form (3NF)	No nonkey dependencies are permitted.

c) How would Figure 1.25 need to be changed to add information about the sales representative who took the order?

ANSWER: As you see in Figure 1.31, you need to add another table that contains the sales representative's name, SALES_REP_ID, and any other relevant information. SALESREP_ID then becomes a foreign key in the ORDER_HEADER table.

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

FIGURE 1.31 ORDER_HEADER with SALES_REP_ID

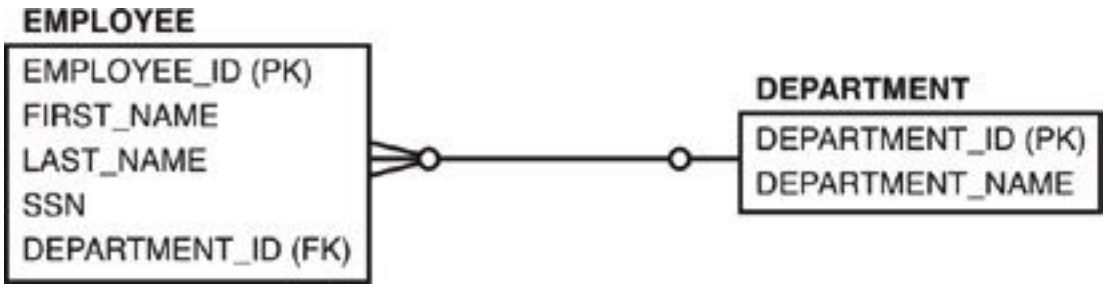


d) How would Figure 1.26 need to be changed if an employee does not have to belong to a department?

ANSWER: You change the relationship line on the DEPARTMENT table end to make it optional. This has the effect that the DEPARTMENT_ID column on the EMPLOYEE table can be null; that is, a value is not required (see Figure 1.32).

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

FIGURE 1.32 EMPLOYEE to DEPARTMENT with optional relationship line



e) Based on Figure 1.26 would the Social Security number (SSN) column be a better primary key column than the EMPLOYEE_ID column?

31

32

ANSWER: The requirement for a primary key is that it is unique, not subject to updates, and not null.

Although the SSN is unique, there have been incidents (though rare) of individuals with the same SSN or individuals who had to change their SSN. Even more common are data entry errors in which numbers are transposed, in which case all the tables containing the incorrect SSN would need to be changed. It is conceivable to have an employee without a SSN assigned yet, hence the

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

column is null. There are a myriad of reasons for not using a SSN; therefore it's best to create a surrogate, or artificial, key.

f) Figures 1.27 and 1.28 depict the logical and physical model of a fictional movie rental database. What differences exist between the entity relationship diagram and the physical schema diagram?

ANSWER: You can spot a number of differences between the logical model (entity relational diagram) and the physical model (database schema diagram). While some logical and physical models are identical, these figures exhibit distinguishing differences you may find in the real world.

The entity name of the logical model is singular versus plural for the table name on the physical model. Some table names have special prefixes that denote the type of application the table belongs to. For example, if a table belongs to the purchase order system, it may be prefixed with PO_; if it belongs to the accounts payable system, the prefix is AP_; and so on. In the logical model, spaces are allowed for table and column names. Typically, in Oracle implementations, table names

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

are defined in uppercase and use the underscore (_) character to separate words.

Although the logical model may include the data types, here the data types (for example, DATE, VARCHAR2, NUMBER) show on the physical model only. The physical model also indicates whether a column allows NULL values.

The attribute and column names differ between the two models. For example, the RATING attribute changed to RATING_CD, which indicates that the values are encoded (for example, “PG” rather than a descriptive “Parental Guidance” value). Designers create or follow established naming conventions and abbreviations for consistency. Naming conventions can help describe the type of values stored in the column.

STOCK_QTY is another example of using the abbreviation QTY to express that the column holds a quantity of copies. This column is absent from the logical model; it is a *derived column*. The quantity of movies for an individual movie title could be determined from the MOVIE_COPIES table. The database designer deliberately denormalized the table by adding this column. This

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

simplifies any queries that determine how many copies of this particular title exist. Rather than issuing another query that counts the number of rows in MOVIE_COPIES for the specific title, this column stores the information.

Adding a derived column to a table requires that the value stay in sync with the data in the related table (MOVIE_COPIES, in this case). The synchronization can be accomplished by writing a program that is executed from the end user's screen. Alternatively, the developer could write a *trigger*, a stored database program, that executes upon a change to the table and automatically updates the STOCK_QTY value. Whenever a new row is added or data is modified on the MOVIE_COPIES table for each individual title, the quantity column can be updated. (For an example of a table trigger, refer to Chapter 12.)

The schema diagram prominently exhibits columns that did not exist in the logical data model, namely CREATED_DATE, MODIFIED_DATE, CREATED_BY, and MODIFIED_BY. Collectively, these columns are sometimes referred to as *audit columns*. They keep information about

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

when a row was created and last changed together with the respective user who executed this action.

32
33

On the logical data model, the relationship is labeled in both directions. On the physical model, the name of the foreign key constraint between the tables is listed instead. You may find that some physical models depict no label at all. There are no set standards for how a physical or logical model must graphically look, and therefore the diagrams produced by various software vendors that offer diagramming tools not only look different but also allow a number of different display options.

33
34

Lab 1.2 Quiz

In order to test your progress, you should be able to answer the following questions.

1) An entity relationship diagram depicts entities, attributes, and tables.

_____**a)** True

_____**b)** False

2) The crow's foot depicts the M of a 1:M relationship.

_____**a)** True

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

_____**b)** False

3) Repeating groups are a violation of the first normal form.

_____**a)** True

_____**b)** False

4) The logical model is derived from the schema diagram.

_____**a)** True

_____**b)** False

5) The concept of denormalization deals with eliminating redundancy.

_____**a)** True

_____**b)** False

6) When you issue a SQL statement, you are concerned with the logical design of the database.

_____**a)** True

_____**b)** False

7) In a mandatory relationship, null values are not allowed in the foreign key column.

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

_____**a)** True

_____**b)** False

8) A nonidentifying relationship means that the foreign key is propagated as a nonkey attribute in the child entity or child table.

_____**a)** True

_____**b)** False

ANSWERS APPEAR IN APPENDIX A.

LAB 1.3 The STUDENT Schema Diagram

LAB OBJECTIVES

After this lab, you will be able to:

- Understand the STUDENT Schema Diagram
- Understand Recursive Relationships
- Describe Surrogate and Natural Keys
- Identify Table Relationships

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

Throughout this book, the database for a school's computer education program is used as a case study on which all exercises are based. If you have worked through the previous two labs, you know that the schema diagram is a model of data that reflects the relationships among data in a database. The name of the case study schema diagram is STUDENT. Before you begin to write SQL statements against the database, it is important to familiarize yourself with the diagram. You can find this graphical representation in Appendix D, "STUDENT Database Schema."

You will frequently be referring to the STUDENT schema diagram shown in Appendix D. Rather than flip back and forth, you might find it more convenient to print out the schema diagram from the companion Web site to this book, located at **www.oraclesqlbyexample.com.**

The STUDENT Table

Examine the STUDENT schema diagram and locate the STUDENT table. This table contains data about each individual student, such as his or her name, address,

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

employer, and the date the student registered in the program.

DATA TYPES

Next to each column name in the diagram, you find the data type of the column. Each column contains a different kind of data, which can be classified by a data type. The FIRST_NAME column is of data type VARCHAR2(25). This means that a variable length of (with a maximum of 25) alphanumeric characters (letters or numbers) may be stored in this column.

Another data type, the CHAR data type, also stores alphanumeric data but is a fixed-length data type. Unlike the VARCHAR2 data type, the CHAR data types pads any unused space in the column with blanks until it reaches the defined column length.

The STUDENT_ID column is of data type NUMBER, with a maximum number of eight integer digits and no decimal place digits; the column is the primary key, as the “(PK)” symbol indicates.

Oracle also provides a DATE data type (as seen on the CREATED_DATE and MODIFIED_DATE columns) that stores both the date and time.

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

You will learn more details about all the various data types in the next chapter. Next to each column, the schema diagram indicates whether a column allows NULL values. A NULL value is an unknown value. A space or value of zero is not the same as NULL. When a column in a row is defined as allowing NULL values, it means that a column does not need to contain a value. When a column is defined as NOT NULL, it must always contain a value.

You will observe that the STUDENT table does not show the city and state. This information can be looked up via the foreign key column ZIP, as indicated with the "(FK)" symbol after the column name. The ZIP column is a NOT NULL column and requires that every student row have a corresponding zip code entered.

The COURSE Table

The COURSE table lists all the available courses that a student may take. The primary key of the table is the COURSE_NO column. The DESCRIPTION column shows the course description, and the COST column lists the dollar amount charged for the enrollment in the course. The PREREQUISITE column displays the course number of a course that must be taken as a prerequisite to

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

this course. This column is a foreign key column, and its values refer to the COURSE_NO column. Only valid COURSE_NO values may be listed in this column. The relationship line of the COURSE table to itself represents a *recursive*, or *self-referencing, relationship*.

RECURSIVE RELATIONSHIP

As the term *recursive*, or *self-referencing, relationship* implies, a column in the COURSE table refers to another column in the same table. The PREREQUISITE column refers to the COURSE_NO column, which provides the list of acceptable values (also referred to as a *domain*) for the PREREQUISITE column. Because the relationship is optional, the foreign key column PREREQUISITE column allows null. Recursive relationships are always optional relationships; otherwise, there is no starting point in the hierarchy.

Figure 1.33 lists an excerpt of data from the COURSE table. The courses with the COURSE_NO column values 10 and 20 do not have a value in the PREREQUISITE column: Those are the courses that a student must take to be able to take any subsequent courses (unless equivalent experience can be substituted). Course number 20 is a prerequisite course for course number 100, Hands-On Windows, and course

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

number 140, Systems Analysis. You will learn more about the intricacies of recursive relationships in Chapter 16, "Regular Expressions and Hierarchical Queries."

FIGURE 1.33 Data from the COURSE table

COURSE_NO	DESCRIPTION	PREREQUISITE	...
10	Technology Concepts		...
20	Intro to Information Systems		...
100	Hands-On Windows	20	...
140	Systems Analysis	20	...
25	Intro to Programming	140	...
...	...	...	...

The SECTION Table

The SECTION table includes all the individual sections a course may have. An individual course may have zero, one, or many sections, each of which can be taught in different rooms, at different times, and by different instructors. The primary key of the table is SECTION_ID. The foreign key that links to the COURSE table is the COURSE_NO column.

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

NATURAL AND SURROGATE KEYS

The SECTION_NO column identifies the individual section number. For example, for the first section of a course, it contains the number 1; the second section lists the number 2, and so on. The two columns, COURSE_NO and SECTION_NO, also uniquely identify a row. This is called a *natural key*. The natural key was not chosen to be the primary key of the table. Instead, a new key column, named SECTION_ID, was created. This SECTION_ID column is called a *surrogate*, or *synthetic, key*.

This surrogate key is system generated and therefore ensures uniqueness. Database designers typically prefer surrogate keys. Imagine a scenario in which the courses of another school are merged into the database. It is conceivable that the combination of COURSE_NO and SECTION_NO are no longer unique. Users will never see or query the surrogate key because it has no business meaning. Instead, users will probably retrieve data using the COURSE_NO and SECTION_NO columns, because these columns represent meaningful data.

The column START_DATE_TIME shows the date and time the section meets for the first time. The

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

LOCATION column lists the classroom. The CAPACITY column shows the maximum number of students that may enroll in this section.

RELATIONSHIPS TO OTHER TABLES

The INSTRUCTOR_ID column is another foreign key column in the SECTION table; it links to the INSTRUCTOR table. The relationship between the SECTION table and the INSTRUCTOR table indicates that an instructor must always be assigned to a section. The INSTRUCTOR_ID column of the SECTION table may never be null, and when you read the relationship from the opposite end, you can say that an individual instructor may teach zero, one, or multiple sections.

The relationship line leading from the COURSE table to the SECTION table means that a course may have zero, one, or multiple sections. Conversely, every individual section *must* have a corresponding row in the COURSE table.

Relationships between tables are based on *business rules*. In this case, the business rule is that a course can exist without a section, but a section cannot exist unless it is assigned to a course. As mentioned, this is indicated with the bar (|) on the other end of the relationship line.

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

Most of the child relationships on the schema diagram are considered mandatory relationships (with two exceptions); this dictates that the foreign key columns in the child table must contain a value (that is, must be NOT NULL), and the value must correspond to a row in the parent table via its primary key value.

The INSTRUCTOR Table

The INSTRUCTOR table lists information related to an individual instructor, such as name, address, phone, and zip code. The primary key of this table is the INSTRUCTOR_ID.

The ZIP column is the foreign key column to the ZIPCODE table. The relationship between INSTRUCTOR and ZIPCODE is an optional relationship, so a null value in the ZIP column is allowed. For a given ZIP column value, there is one and only one value in the ZIPCODE table. For a given ZIP value in the ZIPCODE table, you may find zero, one, or many of the same value in the INSTRUCTOR table.

Another foreign key relationship exists to the SECTION table: An instructor may teach zero, one, or multiple sections, and an individual section can be taught by one and only one instructor.

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

The ZIPCODE Table

The primary key of ZIPCODE table is the ZIP column. For an individual zip code, it allows you to look up the corresponding CITY and STATE column values. This ensures that these values are not repeated in the other tables. The tables have been normalized to avoid redundancy of the data.

The data type of the ZIP column is VARCHAR2 and not NUMBER, because it allows you to enter leading zeros. Both the STUDENT table and the INSTRUCTOR table reference the ZIPCODE table. The relationship between the ZIPCODE and STUDENT tables is mandatory: For every ZIP value in the STUDENT table, there must be a corresponding value in the ZIPCODE table, and for one given zip code, there may be zero, one, or multiple students with that zip code. In contrast, the relationship between the INSTRUCTOR and ZIPCODE tables is optional; the ZIP column of the INSTRUCTOR table may be null.

WHAT ABOUT DELETE OPERATIONS?

Referential integrity does not allow deletion of a primary key value in a parent table that exists in a child as a foreign key value. This would create orphan rows in

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

the child table. There are many ways to handle deletions, and you will learn about this topic and the effects of deletions on other tables in Chapter 11.

The ENROLLMENT Table

The ENROLLMENT table is an *intersection* table between the STUDENT and the SECTION table. It lists the students enrolled in the various sections. The primary key of the table is a composite primary key consisting of the STUDENT_ID and SECTION_ID columns. This unique combination prevents a student from registering for the same section twice.

The ENROLL_DATE column contains the date the student registered for the section. The FINAL_GRADE column lists the student's final grade. The final grade is to be computed from individual grades, such as quizzes, homework assignments, and so on. The column is a derived column because the data to compute the final grade is found in other tables. However, for simplicity, the value is stored here and not computed each time. This simplifies the querying of the final grade information.

The relationship line between the ENROLLMENT and STUDENT tables indicates that one student may be enrolled in zero, one, or many sections. For one row of

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

the ENROLLMENT table, you can find one and only one corresponding row in the STUDENT table. The relationship between the ENROLLMENT and SECTION tables shows that a section may have zero, one, or multiple enrollments. A single row in the ENROLLMENT table always links back to one and only one row in the SECTION table.

The GRADE_TYPE Table

The GRADE_TYPE table is a lookup table for other tables as it relates to grade information. The table's primary key is the GRADE_TYPE_CODE column that lists the unique category of grade, such as MT, HW, PA, and so on. The DESCRIPTION column describes the abbreviated code. For example, for the GRADE_TYPE_CODE MT, you find the description Midterm, and for HW, you see Homework.

The GRADE Table

In this table, you find all the grades related to the section in which a student is enrolled. For example, the listed grades may include the midterm grade, individual quiz grades, final examination grade, and so on. For some grades (for example, quizzes, homework assignments), there may be multiple grades, and the sequence number is

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

shown in the GRADE_CODE_OCCUR-RENCE column. Figure 1.34 displays an excerpt of data from the GRADE table.

The NUMERIC_GRADE column lists the actual grade received. This grade may be converted to a letter grade with the help of the GRADE_CONVERSION table discussed later.

39
40

FIGURE 1.34 Data from the GRADE table

STUDENT_ID	SECTION_ID	GRADE_ TYPE_ CODE	GRADE_ CODE_ OCCURRENCE	NUMERIC_ GRADE	...
221	104	FI	1	77	...
221	104	HM	1	76	...
221	104	HM	2	76	...
221	104	HM	3	86	...
221	104	HM	4	96	...
221	104	MT	1	90	...
221	104	PA	1	83	...
221	104	QZ	1	84	...
221	104	QZ	2	83	...
...	...	...	...	...	...

The primary key columns are STUDENT_ID, SECTION_ID, GRADE_TYPE_CODE, and GRADE_CODE_OCCURRENCE. From the relationship between the ENROLLMENT and GRADE tables, you can learn that rows exist in the GRADE table only if the student is actually enrolled in the section listed in the

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

ENROLLMENT table. In other words, it is not possible for a student to have grades for a section in which he or she is not enrolled. The foreign key columns STUDENT_ID and SECTION_ID from the ENROLLMENT table enforce this relationship.

The GRADE_TYPE_WEIGHT Table

The GRADE_TYPE_WEIGHT table aids in computation of the final grade a student receives for an individual section. This table details how the final grade for an individual section is computed. For example, the midterm may constitute 50 percent of the final grade, all the quizzes 10 percent, and the final examination 40 percent. If there are multiple grades for a given GRADE_TYPE_CODE, the lowest grade may be dropped if the column DROP_LOWEST contains the value Y.

The final grade is determined by using the individual grades of the student and section in the GRADE table in conjunction with this table. This computed final grade value is stored in the FINAL_GRADE column of the ENROLLMENT table discussed previously. (The FINAL_GRADE column is a derived column. As mentioned earlier, the values to compute this number are available in the GRADE and GRADE_TYPE_WEIGHT

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

tables, but because the computation of this value is complex, it is stored to simplify queries.)

The primary key of the GRADE_TYPE_WEIGHT table consists of the SECTION_ID and GRADE_TYPE_CODE columns. A particular GRADE_TYPE_CODE value may exist zero, one, or multiple times in the table. For every row of the GRADE_TYPE_WEIGHT table, you find one and only one corresponding GRADE_TYPE_CODE value in the GRADE_TYPE table.

The relationship between the GRADE_TYPE_WEIGHT table and the SECTION table indicates that a section may have zero, one, or multiple rows in the GRADE_TYPE_WEIGHT table for a given SECTION_ID value. For one SECTION_ID value in the GRADE_TYPE_WEIGHT table, there must always be one and only one corresponding value in the SECTION table.

The GRADE_CONVERSION Table

The purpose of the GRADE_CONVERSION table is to convert a number grade to a letter grade. The table does not have any relationship with any other tables. The column LETTER_GRADE contains the unique letter grades, such as A+, A, A-, B, and so forth. For each of

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

these letter grades, there is an equivalent number range. For example, for the letter B, the range is 83 through 86 and is listed in the MIN_GRADE and MAX_GRADE columns.

You can find the individual table and column descriptions of all the tables listed in the STUDENT schema diagram in Appendix E, "Table and Column Descriptions."

LAB 1.3 EXERCISES

a) What does the STUDENT schema diagram represent?

b) Does the STUDENT schema diagram tell you where a student lives? Explain.

c) What four columns are common to all tables in the STUDENT schema diagram?

d) What is the primary key of the COURSE table?

e) How many primary keys does the ENROLLMENT table have? Name the column(s).

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

f) How many foreign keys does the SECTION table have?

g) Will a foreign key column in a table accept any data value? Explain, using the STUDENT and ZIPCODE tables.

h) To make the relationship between the ZIPCODE and STUDENT tables optional, what would have to change in the STUDENT table?

i) From what domain of values (that is, what column in what table) does the PREREQUISITE column of the COURSE table get its values?

j) Explain the relationship(s) the ENROLLMENT table has to other table(s).

LAB 1.3 EXERCISE ANSWERS

a) What does the STUDENT schema diagram represent?

ANSWER: The STUDENT schema diagram is a graphical representation of tables in a relational database.

A schema diagram is a useful tool during the software development lifecycle. English-like

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

words should be used to name tables and columns so that anyone, whether developer or end user, can look at a schema diagram and grasp the meaning of data and the relationships. Developers study a schema diagram to understand the design of a database long before they put hands on the keyboard to develop a system, and end users can use the schema diagram to understand the relationship between the data elements.

41
42

b) Does the STUDENT schema diagram tell you where a student lives? Explain.

ANSWER: No. The STUDENT schema diagram tells you how data is organized in a relational database: the names of tables, the columns in those tables, and the relationship among them. It cannot tell you what actual data looks like. You use the SQL language to interact with a relational database to view, manipulate, and store the data in the tables.

c) What four columns are common to all tables in the STUDENT schema diagram?

ANSWER: The four columns are CREATED_BY, CREATED_DATE, MODIFIED_BY, and MODIFIED_DATE.

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

Database tables often contain columns similar to these four to create an audit trail. These columns are designed to identify who first created or last modified a row of a table and when the action occurred. You typically find these columns only on the physical schema diagram, not on the logical model. Some of these values in the columns can be filled in automatically by writing triggers. You will see an example of a table trigger in Chapter 12. (Triggers are described in further detail in another book in this series: *Oracle PL/SQL by Example*, 4th edition, by Benjamin Rosenzweig and Elena Silvestrova Rakhimov; Prentice Hall, 2008.)

d) What is the primary key of the COURSE table?

ANSWER: The primary key of the COURSE table is the column COURSE_NO.

You can identify the primary key because it has a PK symbol listed next to the column. In general, a primary key uniquely identifies a row in a table, and the column or columns of the primary key are defined as NOT NULL.

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

e) How many primary keys does the ENROLLMENT table have? Name the column(s).

ANSWER: A table can have only one primary key. The primary key of the ENROLLMENT table consists of the two columns STUDENT_ID and SECTION_ID.

As mentioned earlier, a primary key uniquely identifies a single row in a table. In the case of the ENROLLMENT table, two columns uniquely identify a row and create a composite primary key.

In the schema diagram, these two columns are also foreign keys. The STUDENT_ID column is the foreign key to the STUDENT table, and SECTION_ID is the foreign key to the SECTION table. Both foreign key relationships are identifying relationships.

f) How many foreign keys does the SECTION table have?

ANSWER: Two. The foreign keys of the SECTION table are COURSE_NO and INSTRUCTOR_ID.

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

g) Will a foreign key column in a table accept any data value? Explain, using the STUDENT and ZIPCODE tables.

ANSWER: No. A foreign key must use the values of the primary key it references as its domain of values.

The ZIP column is the primary key in the ZIPCODE table. The STUDENT table references this column with the foreign key ZIP column. Only values that exist in the ZIP column of the ZIPCODE table can be valid values for the ZIP column of the STUDENT table. If you attempt to create a row or change an existing row in the STUDENT table with a zip code not found in the ZIPCODE table, the foreign key constraint on the STUDENT table will reject it.

In general, a foreign key is defined as being a column, or columns, in the child table. This column refers to the primary key of another table, referred to as the parent table.

The primary key values are the domain of values for the foreign key column. A *domain* is a set of values that shows the possible values a column

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

can have. The primary key values of the parent table are the only acceptable values that can appear in the foreign key column in the other table. (Domains are not only used in context with primary key and foreign key relationships but can also be used for a list of values that may be stored in a table. For example, common domains include Yes/No, Gender: Male/Female/Unknown, Weekday: Sun/Mon/Tue/Wed/Thu/Fri/Sat.)

h) To make the relationship between the ZIPCODE and STUDENT tables optional, what would have to change in the STUDENT table?

ANSWER: The foreign key column ZIP in the STUDENT table would have to be defined as allowing NULL values. It is currently defined as NOT NULL. The relationship should be indicated as optional instead of mandatory, as shown in Figure 1.35.

FIGURE 1.35 The relationships of the ZIPCODE table

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

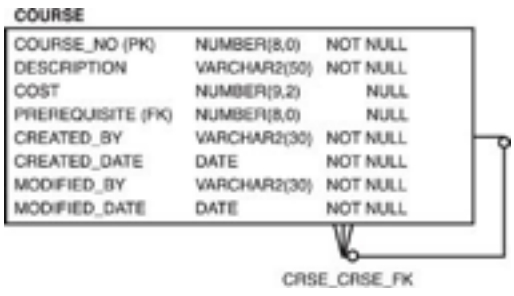
There is such an optional relationship between the INSTRUCTOR and ZIPCODE tables. All the NOT NULL values of ZIP in the INSTRUCTOR table must be found in the ZIPCODE table.

i) From what domain of values (that is, what column in what table) does the PREREQUISITE column of the COURSE table get its values?

ANSWER: From the COURSE_NO column in the COURSE table.

In this case, the PREREQUISITE column refers to the COURSE_NO column, which provides the domain of values for the PREREQUISITE column. A prerequisite is valid only if it is also a valid course number in the COURSE table. This relationship is shown in Figure 1.36.

FIGURE 1.36 The self-referencing relationship of the COURSE table



PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

j) Explain the relationship(s) the ENROLLMENT table has to other table(s).

ANSWER: The STUDENT table and the SECTION table are the parent tables of the ENROLLMENT table. The ENROLLMENT table is one of the parent tables of the GRADE table.

As shown in Figure 1.37, the relationship between the STUDENT and SECTION tables signifies that a student may be enrolled in zero, one, or many sections. One individual student can be enrolled in one specific section only once; otherwise, the unique combination of the two columns in the ENROLLMENT table would be violated. The combination of these two foreign key columns represents the primary key of the ENROLLMENT table.

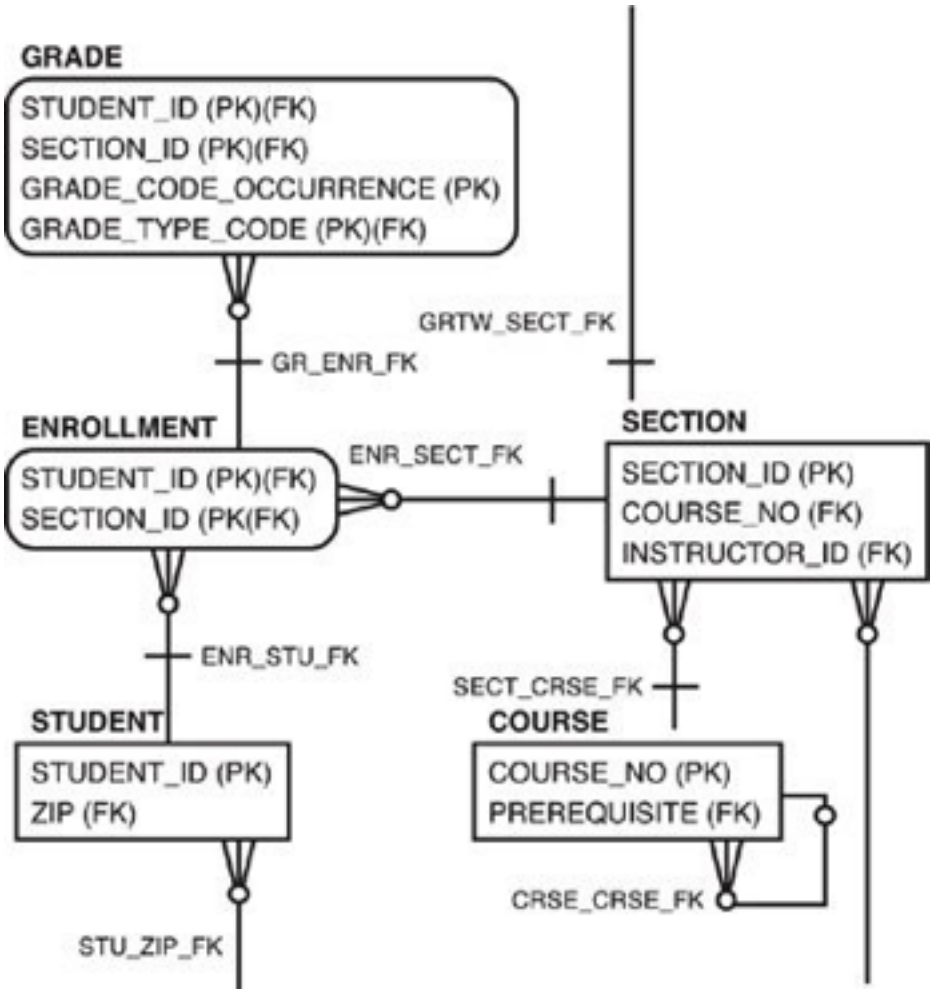
The relationship of the ENROLLMENT table as the parent of the GRADE table shows that for an individual student and her or his enrolled section, there may be zero, one, or many grades. The primary key columns of the ENROLLMENT table (STUDENT_ID and SECTION_ID) are foreign keys in the GRADE table that become

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

part of the GRADE table's composite primary key. Therefore, only enrolled students may have rows in the GRADE, as indicated with the optional line. If a row in GRADE exists, it must be for one specific enrollment in a section, for one specific student.

44
45

FIGURE 1.37 The relationships of the ENROLLMENT table



PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

It is important to note that in some cases, the foreign keys become part of a table's primary key, as in the ENROLLMENT table or the GRADE table. If a composite primary key contains many columns (perhaps more than four or five), a surrogate key consisting of one column may be considered for simplicity. The decision to use a surrogate key is based on the database designer's understanding of how data is typically accessed by the application programs.

Lab 1.3 Quiz

In order to test your progress, you should be able to answer the following questions.

1) What role(s) does the STUDENT_ID column play in the GRADE table? Select all that apply.

_____**a)** Part of composite primary key

_____**b)** Primary key

_____**c)** Foreign key

2) The GRADE_TYPE table does not allow values to be NULL in any column.

_____**a)** True

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

_____**b)** False

3) The number of columns in a table matches the number of rows in that table.

_____**a)** True

_____**b)** False

4) The SECTION table has no foreign key columns.

_____**a)** True

_____**b)** False

5) A table can contain 10 million rows.

_____**a)** True

_____**b)** False

6) A primary key may contain NULL values.

_____**a)** True

_____**b)** False

7) A column name must be unique within a table.

_____**a)** True

_____**b)** False

PRINTED BY: Monica Gonzalez <monicazalez1@hotmail.com>. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

8) If a table is a child table in three different one-to-many relationships, how many foreign key columns does it have?

_____**a)** One

_____**b)** Exactly three

_____**c)** Three or more

9) Referential integrity requires the relationship between foreign key and primary key to maintain values from the same domain.

_____**a)** True

_____**b)** False