Ryan Stephens

SEVENTH EDITION

New sample database + standard SQL examples: Use these skills anywhere

Sams Teach Yourself SQL in 24 Hours

Pearson

FREE SAMPLE CHAPTER

SHARE WITH OTHERS

Ryan Stephens

Sams Teach Yourself SQL in 24 Hours

Seventh Edition

SAMS

Sams Teach Yourself SQL in 24 Hours, Seventh Edition

ISBN-13: 978-0-13-754312-0
ISBN-10: 0-13-754312-3

Library of Congress Control Number: 2021948117

ScoutAutomatedPrintCode

Trademarks

All terms mentioned in this book that are known to be trademarks or service marks have been appropriately capitalized. Sams Publishing cannot attest to the accuracy of this information. Use of a term in this book should not be regarded as affecting the validity of any trademark or service mark.

Warning and Disclaimer

Every effort has been made to make this book as complete and as accurate as possible, but no warranty or fitness is implied. The information provided is on an "as is" basis. The author and the publisher shall have neither liability nor responsibility to any person or entity with respect to any loss or damages arising from the information contained in this book.

Special Sales

For information about buying this title in bulk quantities, or for special sales opportunities (which may include electronic versions; custom cover designs; and content particular to your business, training goals, marketing focus, or branding interests), please contact our corporate sales department at corpsales@pearsoned.com or (800) 382-3419.

For government sales inquiries, please contact governmentsales@pearsoned.com.

For questions about sales outside the U.S., please contact intlcs@pearson.com.

Editor-in-Chief
Mark Taub

Acquisitions Editor
Malobika Chakraborty

Development Editor
Mark Renfrow

Managing Editor
Sandra Schroeder

Senior Project Editor
Tonya Simpson

Copy Editor
Krista Hansing Editorial Services

Indexer
Cheryl Lenser

Proofreader
Betty Pessagno

Cover Designer
Chuti Prasertsith

Compositor
codemantra

Pearson's Commitment to Diversity, Equity, and Inclusion

Pearson is dedicated to creating bias-free content that reflects the diversity of all learners. We embrace the many dimensions of diversity, including but not limited to race, ethnicity, gender, socioeconomic status, ability, age, sexual orientation, and religious or political beliefs.

Education is a powerful force for equity and change in our world. It has the potential to deliver opportunities that improve lives and enable economic mobility. As we work with authors to create content for every product and service, we acknowledge our responsibility to demonstrate inclusivity and incorporate diverse scholarship so that everyone can achieve their potential through learning. As the world's leading learning company, we have a duty to help drive change and live up to our purpose to help more people create a better life for themselves and to create a better world.

Our ambition is to purposefully contribute to a world where:

- Everyone has an equitable and lifelong opportunity to succeed through learning.
- Our educational products and services are inclusive and represent the rich diversity of learners.
- Our educational content accurately reflects the histories and experiences of the learners we serve.
- Our educational content prompts deeper discussions with learners and motivates them to expand their own learning (and worldview).

While we work hard to present unbiased content, we want to hear from you about any concerns or needs with this Pearson product so that we can investigate and address them.

- Please contact us with concerns about any potential bias at https://www.pearson.com/report-bias.html.

Contents at a Glance

Table of Contents

About the Author

Ryan Stephens is an entrepreneur who has built his career and multiple IT companies around SQL, data, and relational databases. He has shared his knowledge and experience with organizations, students, and IT professionals all over the world. Two of the companies he has co-founded, Perpetual Technologies, Inc. (PTI), and Indy Data Partners, have provided expert database and IT services to large-scale government and commercial clients for more than 25 years. Ryan has authored several books for Pearson, including *Sams Teach Yourself SQL in 24 Hours*, 6th Edition; some of his books have been translated and published internationally. Additionally, Ryan has worked for large organizations and has consulted within the areas of SQL, database design, database management, and project management. He designed and taught a database management program for Indiana University–Purdue University in Indianapolis and currently teaches online SQL and database classes for Pearson Education.

Dedication

This book is dedicated to my daughter,
Charlie Marie, named after my late adoptive mom,
Charlotte Anne Pritchett Stephens, who is the first person I remember
teaching me to write. I love you both.

Acknowledgments

First of all, I would like to thank the contributing authors to previous editions of this book, Ronald Plew and Arie D. Jones.

Ronald Plew is my retired co-founder of Perpetual Technologies and is a great friend. I could not have started my first successful business without him. Ron studied and consulted in the field of relational database technology for more than 20 years, is a published author, and helped me build a database training program at Indiana University–Purdue University (IUPUI) in Indianapolis. Ron taught SQL and database classes at IUPUI for 5 years.

Arie D. Jones is the vice president for Emerging Technologies for Indy Data Partners, Inc. (IDP), in Indianapolis and helped me get that company off to a successful start. Arie leads IDP's team of experts in planning, design, development, deployment, and management of database environments and applications to achieve the best combination of tools and services for each client. He is a regular speaker at technical events and has authored several books and articles on database-related topics.

Thank you also to Angie Gleim, Amy Reeves, Terri Klein, and all of my team members for doing such a professional job running our companies so that I have the freedom to take on projects like this. And as always, thanks to the staff at Pearson and Sams Publishing for your attention to detail and patience. It is always a pleasure working with you.

We Want to Hear from You!

As the reader of this book, *you* are our most important critic and commentator. We value your opinion and want to know what we're doing right, what we could do better, what areas you'd like to see us publish in, and any other words of wisdom you're willing to pass our way.

We welcome your comments. You can email or write to let us know what you did or didn't like about this book—as well as what we can do to make our books better.

Please note that we cannot help you with technical problems related to the topic of this book.

When you write, please be sure to include this book's title and author as well as your name and email address. We will carefully review your comments and share them with the author and editors who worked on the book.

Email: community@informit.com

What This Book Is About

This book is about the Structured Query Language (SQL), the standard language used to communicate with any relational database. Relational databases are one of the most popular data-management solutions, and many vendor software implementations of SQL are available for organizations and individuals. The text does not focus on a specific implementation of SQL, but Oracle is used for most of the examples because it is a leading database-management software product that adheres closely to the SQL standard. In the upcoming hours, you learn how to apply the SQL standard from a practical standpoint, using real-world examples and hands-on exercises. You can use almost any SQL implementation with this book, which attests to the portability of SQL and its popularity.

What You Need for This Book

This book has no prerequisites, other than a desire to learn about relational databases and SQL. The material is adapted from an approach that the author has used for more than 25 years in the consulting and information technology world to teach thousands of students both locally and globally. It starts with a basic approach to understanding databases and both defining and managing data. It also introduces SQL as the standard language for communicating with any relational database. The book is set up to show all the basic concepts and syntax of SQL while also providing real-world examples. Beginners can easily learn from simplified examples, while experienced information technology professionals can peruse the more advanced exercises.

For software, access to a relational database management system is recommended. This book uses Oracle, but many other options also adhere to the SQL standard (including Microsoft SQL Server, MySQL, IBM, and PostgreSQL). If you do not currently have access to a relational database management system, many options are available for free, either as online access or as a download for learning and development purposes. The author provides guidance on software options in the book as well. All the database creation scripts and data are provided to the reader for hands-on exercises and continued learning after completing this book.

Thank you, and enjoy.

Author's Website

Please visit the author's website at www.ryanSQL.com for additional SQL resources, hands-on exercises, and links to other Pearson Education materials created by the author.

HOUR 1

Understanding the Relational Database and SQL

What You'll Learn in This Hour:

- Understand information, data, and databases in organizations
- Examine common database environments
- Learn about the key components of the relational database
- Understand how SQL relates to the relational database

Welcome to the world of Structured Query Language (SQL) and the vast, growing database technologies of today's global organizations. By reading this book, you have taken your first step in building the knowledge base needed to navigate today's world of relational databases and data management. Before you dive into SQL, its role in relational databases, and the specifics of the language, you learn the basic concepts of the relational database in this hour so that you have a solid foundation to build upon for increased knowledge and sustainability.

NOTE

SQL Is the Language to Know in IT

SQL is the standard language used to communicate with any relational database, so this first hour concentrates on the relational database itself. This concept is critical in learning SQL and understanding how to most effectively take advantage of your data. In the long run, truly understanding the relational database puts you in a position to excel beyond your peers and make your organization more competitive in a world that heavily relies on data.

The best approach to fully understanding any topic, including SQL, is to start with the basics and build from there. In this hour, you learn the basics of the relational database, get a little SQL history, and look at the key elements that comprise the relational database. From there, you will have a solid foundation to build upon.

SQL is a database language. It is the standard language used to communicate with any relational database, which is one of the most popular databases in use today (and, in fact, for decades). As you begin to understand the nuts and bolts of a relational database—and even database design, which later chapters cover—you can more easily grasp the concepts of SQL and more quickly apply SQL concepts and syntax to real-world situations in real databases. Hour 2, "Exploring the Components of the SQL Language," focuses on the standard components of the SQL language to provide a preview. Then Hours 3, "Getting to Know Your Data," and 4, "Setting Up Your Database," get you involved with the database used in this book so that you deeply understand both the data and how that data is related in the database. When you have this information, you will understand SQL more easily, the knowledge you gain will stick, and you will have the basic foundation to start applying SQL.

Thriving in a Data-Driven World

Data is all around you: You can see it every day in your personal life and the organizations you work for. Data is on your phone, on your computer, in online stores, inside the walls of physical stores, and in complex databases in organizations all over the world. Truly, the world cannot survive without data.

Data, or information, has been around since the beginning of time and people have used that data to make decisions. The modern world has evolved to better understand how data can be used in everyday life and how to make organizations more competitive.

As the world becomes more competitive, it is your responsibility to learn how to use data effectively. You must be able to understand the information and data you're working with and how to apply it to everyday situations. In today's world, you have access to mountains of information, especially on the Internet. However, information must become usable data: If the data is not usable, it is useless. If it is inaccurate, it is useless. If it contains inconsistencies, it is useless. Databases and other technologies have helped people better take advantage of data, but this has also created a new problem: The volume of data is growing quickly and must be managed carefully. The relational database and SQL, in particular, facilitate easy data management, if done right.

NOTE

Information Must Become Usable Data

Information is all around us. However, a lot of information in this world is incorrect or not consistent with other sources. Even in databases within some of the most respected organizations in the world, data can be inaccurate or difficult to understand. This can create business problems. You must understand how to turn information into useful data for you and your organization. Fortunately, you can use the relational database and SQL to protect data and present it to end users in a relevant manner. Data must be kept safe, clean, and consistent within the database, and SQL offers a primary way to control that.

Organizations, Data, and User Communities

You know by now that data is critical to the success of any organization or individual. Within an organization, many types of individuals depend on data—some of them every day:

- Organizational leaders
- Managers
- Technical users
- Administrators
- Functional and end users
- Stakeholders
- Customers

Figure 1.1 illustrates these individuals and also shows how data sits at the core of any organization. All sorts of users access data that is stored in a database of some sort. Applications are available to end users to manage data in a database and query the data in the database. This data is available to customers and other users to access the database in a limited manner, without jeopardizing the integrity of the data itself. Reports and other useful information can be generated from a database so that key personnel in an organization can use them to make high-level decisions; *business intelligence* is another term for this. Additionally, the administrators of the database are included in the general information technology environment, along with technical users that design, develop, and code the elements within a database and the applications surrounding the data. By looking at this simple diagram, you can easily see how important data is in an organization. SQL is the standard language that brings everything together so that all individuals, whether directly or indirectly, can interact with the database and get both accurate and available data to successfully perform their everyday duties.

Databases Defined

In simple terms, a *database* is a collection of data. You might like to think of a database as an organized mechanism that has the capability to store information that a user can retrieve in an effective and efficient manner.

As previously mentioned, people use databases every day without realizing it. For example, a contact list is a database of names, addresses, emails, telephone numbers, and other important information. The listings are alphabetized or indexed, which enables the user to easily look up a particular contact and quickly get in touch. Ultimately, this data is stored in a database somewhere on a computer.

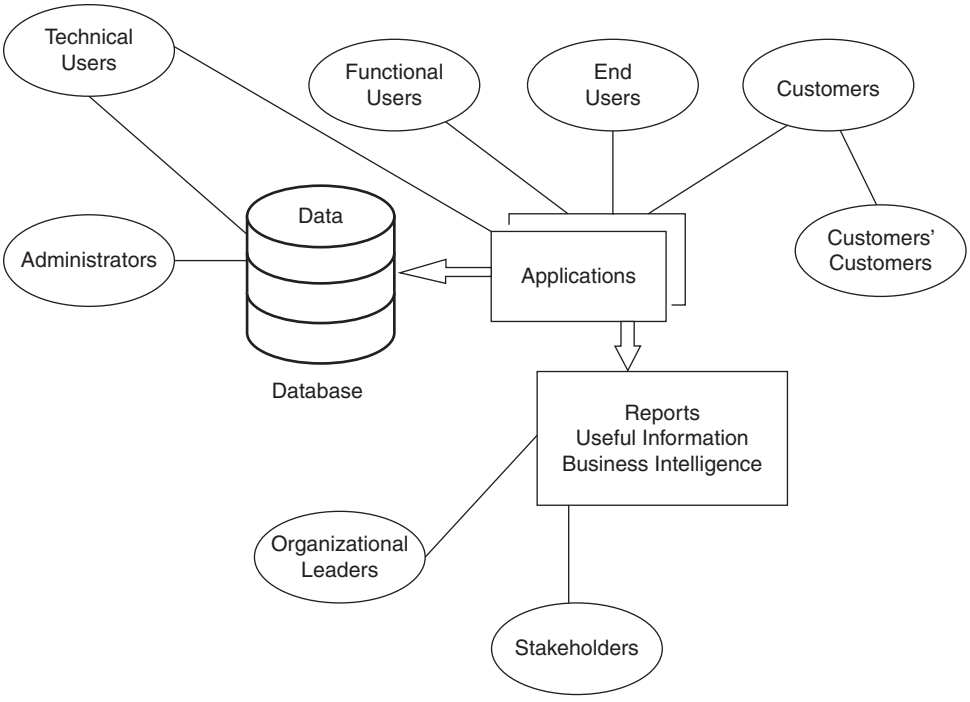

FIGURE 1.1
Data and user communities

Of course, databases must be maintained. As people move to different cities or states, entries have to be added or removed from the contact list. Likewise, entries need to be modified when people change names, addresses, telephone numbers, and so on. Figure 1.2 illustrates a simple database.

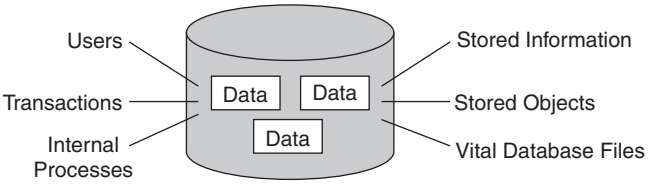

FIGURE 1.2
A typical database

Think of your mobile phone or any mobile device: It contains data and probably has many apps installed that access various databases. Additionally, your phone contains data in the form of contact lists and pictures.

Common Database Environments

In this section, you explore a couple of the most common database environments. A database environment contains all the key components to house a database and store data effectively, from the database itself, to the operating system, the network, and any applications that access the database via these key components. The following two major environments are briefly discussed in this hour:

- Client/server environments
- Web-based environments

Client/Server Environments

In the past, the computer industry was predominately ruled by mainframe computers—large, powerful systems capable of high storage capacity and high data processing capabilities. Users communicated with the mainframe through dumb terminals—terminals that did not think on their own but relied solely on the mainframe's CPU, storage, and memory. Each terminal had a data line attached to the mainframe. The mainframe environment definitely served its purpose and still does in many businesses, but a greater technology was soon introduced: the client/server model.

In the *client/server system*, the main computer, called the server, is accessible from a network—typically, a *local area network (LAN)* or a *wide area network (WAN)*. The server is normally accessed by personal computers (PCs) or other servers instead of dumb terminals. Each PC, called a *client*, is granted access to the network, allowing communication between the client and the server. The main difference between client/server and mainframe environments is that the user's PC in a client/server environment can think on its own and run its own processes using its own CPU and memory, but it is readily accessible to a server computer through a network. In most cases, a client/server system is much more flexible for today's overall business needs.

Modern database systems reside on various types of computer systems with various operating systems. The most common types of operating systems are Windows-based systems, Linux, and command-line systems such as UNIX. Databases reside mainly in client/server and web environments.

Lack of training and experience is the main reason for failed implementations of database systems. Today's businesses need personnel who can work within the client/server model and web-based systems (explained in the next section), can address the rising (and sometimes unreasonable) demands placed on modern organizations, and understand Internet technologies and network computing. Figure 1.3 illustrates the concept of client/server technology.

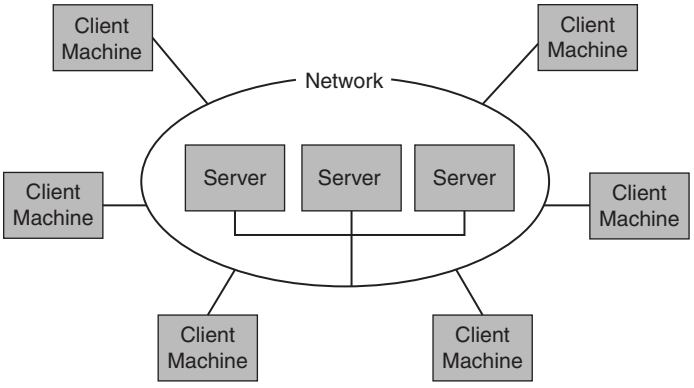

FIGURE 1.3
The client/server model

Web-Based Environments

Business information systems have largely moved toward web integration. Databases have been accessible through the Internet for years, meaning that customers access an organization's information through an Internet browser such as Chrome, Microsoft Edge, or Firefox. Customers (users of data) can order merchandise, check inventories, check the status of their orders, make administrative changes to accounts, transfer money from one account to another, and so on.

A customer simply invokes an Internet browser, goes to the organization's website, logs in (if required by the organization), and uses an application built into the organization's web page to access data. Most organizations first require users to register and then issue them a login and password.

Of course, many actions happen behind the scenes when a database is accessed through a web browser. For instance, the web application can execute SQL that accesses the organization's database, returns data to the web server, and then returns that data to the customer's Internet browser.

From the user's standpoint, the basic structure of a web-based database system is similar to that of a client/server system (refer to Figure 1.3). Each user has a client machine that is connected to the Internet and contains a web browser. The network in Figure 1.3 (for a web-based database) just happens to be the Internet instead of a local network. For the most part, a client is still accessing a server for information; it doesn't matter that the server might exist in another state or even another country. The main point of web-based database systems is to expand the potential customer base of a database system that knows no physical location bounds, thus increasing data availability and an organization's customer base.

NOTE

Data Integration Across Multiple Environments

These days, organizations and individuals alike use a variety of databases and applications to access data that is often spread among multiple environments. With modern technology, data located in a variety of environments, vendor implementations, and even types of databases can be seamlessly integrated.

Understanding the Relational Database

A *relational database* is a database that is divided into logical units called tables. These tables are related to one another within the database. A relational database allows data to be broken down into logical, smaller, manageable units, facilitating easier maintenance and providing more optimal database performance according to the level of organization. In Figure 1.4, you can see that tables are related to one another through a common key (data value) in a relational database.

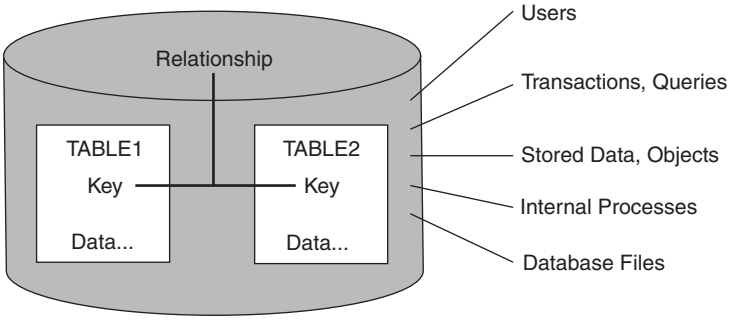

FIGURE 1.4
The relational database

Again, tables are related in a relational database so that adequate data can be retrieved in a single query (although the desired data might exist in more than one table). Having common *keys*, or *fields*, among relational database tables allows data from multiple tables to be joined to form one large set of data. As you venture deeper into this book, you see more of a relational database's advantages, including overall performance and easy data access.

NOTE

Relationships Within a Relational Database

The following sections provide an overview of relationships between data in a RDBMS. Sample data illustrates how data can be related to one another and how easy it is to bring common data together. Future hours expand upon this through numerous examples and hands-on exercises. You will even get the opportunity to design a portion of the sample database in this book and define relationships of your own.

Taking a Glimpse into a Sample Database

All databases have a simple reason for existing: They store and maintain valuable data. In this section, you look at a simplified example data set that illustrates how data might look in a relational database and also shows how data is related through key relationships. These relationships, which are also rules for how data is stored, hold the key to the value of a relational database.

Figure 1.5 shows a table called `EMPLOYEES`. A table is the most basic type of object where data is stored within a relational database. A database object is a defined structure that physically contains data or has an association to data stored in the database.

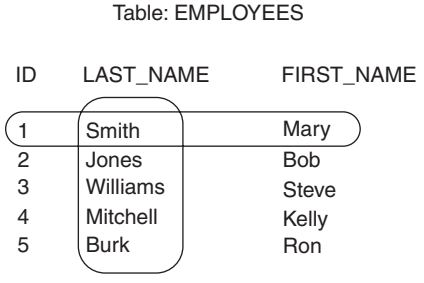

FIGURE 1.5
Table structure

Fields

Every table is divided into smaller entities called fields. A *field* is a column in a table that is designed to maintain specific information about every record in the table. The fields in the `EMPLOYEES` table consist of `ID`, `LAST_NAME`, and `FIRST_NAME`. They categorize the specific information that is maintained in a given table. Obviously, this is a simplistic example of the data that might be stored in a table such as `EMPLOYEES`.

Records, or Rows of Data

A *record*, also called a *row* of data, is a horizontal entry in a table. Looking at the last table, `EMPLOYEES`, consider the following first record in that table:

```
1       Smith      Mary
```

The record consists of an employee identification, employee last name, and employee first name. For every distinct employee, there should be a corresponding record in the `EMPLOYEES` table.

A *row of data* is an entire record in a relational database table.

Columns

A *column* is a vertical entity in a table that contains all the information associated with a specific field in a table. For example, a column in the EMPLOYEES table for the employee's last name consists of the following:

```
Smith
Jones
William
Mitchell
Burk
```

This column is based on the field LAST_NAME, the employee's last name. A column pulls information about a certain field from every record within a table.

Referential Integrity

Referential integrity is the hallmark of any relational database. Figure 1.6 shows two tables, EMPLOYEES and DEPENDENTS, that are related to one another in our imaginary database. The DEPENDENTS table is simply a table that contains information about dependents of each employee in the database, such as spouses and children. As in the EMPLOYEES table, the DEPENDENTS table has an ID. The ID in the DEPENDENTS table references, or is related to, the ID in the EMPLOYEES table. Again, this is a simplistic example to show how relationships work in a database and to help you understand referential integrity.

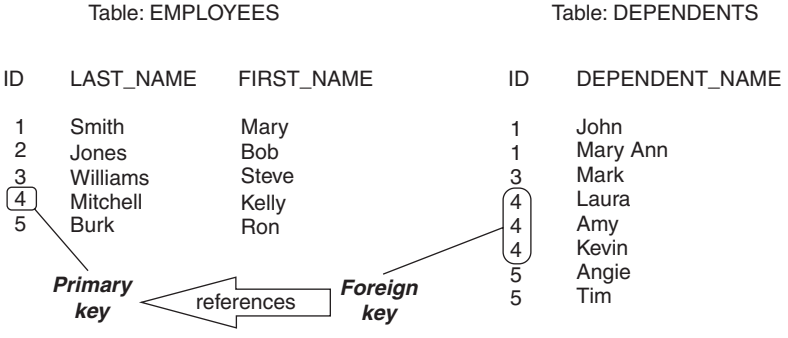

FIGURE 1.6
Table relationships

The key point to note in Figure 1.6 is that there is a relationship between the two tables through the ID field. The ID field in EMPLOYEES is related to the ID field in DEPENDENTS. The ID field in EMPLOYEES is a primary key, whereas the ID field in DEPENDENTS is a foreign key. These types of keys are critical to any relational database structure and to referential integrity.

Primary Keys

A *primary key* is a column that makes each row of data in the table unique in a relational database. In Figure 1.6, the primary key in the `EMPLOYEES` table is `ID`, which is typically initialized during the table creation process. The primary key ensures that all employee identifications are unique, so each record in the `EMPLOYEES` table has its own `ID`. Primary keys alleviate the possibility of a duplicate record in a table and are used in other ways, which you learn more about as you progress throughout the book.

Primary keys are typically initialized during the table creation process, although a primary key can be added later as long as duplicate data does not already exist in the table.

Foreign Keys

A *foreign key* is a column in a table that references a column in another table. Primary and foreign keys establish relationships between tables in a relational database.

Columns identified as foreign keys have these characteristics:

- Do not have to have unique values
- Ensure that each entry in the column has a corresponding entry in the referenced table
- Ensure that column data (child records) are never deleted if corresponding data (parent records) is found in the table referenced (the primary key column)

In Figure 1.6, the foreign key in the `DEPENDENTS` table is `ID`, which is the column that contains the employee `ID` of the corresponding table in the `EMPLOYEES` table. Again, this is a simplistic example: Ideally, the `DEPENDENTS` table would have an `ID` for each dependent and an `ID` for each employee referenced. So for learning purposes, the `ID` in the `DEPENDENTS` table is a foreign key that references the `ID` in the `EMPLOYEES` table. The record, or row of data, in the `EMPLOYEES` table, is a parent record because of the primary key, which might have child records within the database. Likewise, the `ID` in the `DEPENDENTS` table is a foreign key, or child record, which requires a relationship to a parent record, or primary key, somewhere in the database.

NOTE

Relationships Within a Relational Database

Primary and foreign key relationships, or parent/child relationships, are the most basic relationships in a relational database. This is the most basic rule that can be set and is the only way to facilitate the ease and efficiency that a relational database provides for effective data management.

When manipulating data in the database, keep the following points in mind:

- You add data into a column identified as a foreign key unless corresponding data already exists as a primary key in another table.

- You cannot delete data from a column identified as a foreign key unless you first remove any corresponding data from columns in tables with primary keys that are referenced by the foreign key.

Getting Data from Multiple Tables

In Figure 1.6, you should understand the primary and foreign keys that have been defined at this point. When these keys are defined upon table creation or modification, the database knows how to help maintain the integrity of data within and also knows how that data is referenced between tables.

Let's say you want to know the dependents of Kelly Mitchell. You have likely already used a common-sense process while looking at Figure 1.6. However, this was probably your thought process:

- You see that Kelly Mitchell's ID is 4.
- You look for Kelly Mitchell's ID of 4 in the `DEPENDENTS` table.
- You see that Kelly Mitchell's ID of 4 corresponds to three records in the `DEPENDENTS` table.
- Therefore, Kelly Mitchell's dependents are Laura, Amy, and Kevin.

This common-sense process of finding information is almost exactly the process that occurs behind the scenes when you use SQL to "ask" a relational database about the data it contains.

`NULL` Values

`NULL` is the term used to represent a missing value. A *NULL value* in a table is a value in a field that appears to be blank; that field has no value. It is important to understand that a `NULL` value is different from a zero value or a field that contains spaces: A field with a `NULL` value has intentionally been left blank during record creation. For example, a table that contains a column called `MIDDLE_NAME` might allow null or missing values because not every person has a middle name. Records in tables that do not have an entry for a particular column signify a `NULL` value.

Additional table elements are discussed in detail during the next two hours.

Logical vs. Physical Database Elements

Within a relational database, both logical and physical elements exist throughout the database lifecycle. Logical elements are typically conceived as database structures during the planning and design phase. Physical structures are objects that are created later; they comprise the database itself that stores data that SQL and various applications access.

For example, logical elements might include the following elements during conception:

- Entities
- Attributes
- Relationships
- Information/data

Those logical elements later become the following physical elements during database creation:

- Tables
- Fields/columns
- Primary and foreign key constraints (built-in database rules)
- Usable data

Database Schemas

A *schema* is a group of related objects in a database. A schema is owned by a single database user, and objects in the schema can be shared with other database users. Multiple schemas can exist in a database. Figure 1.7 illustrates a database schema.

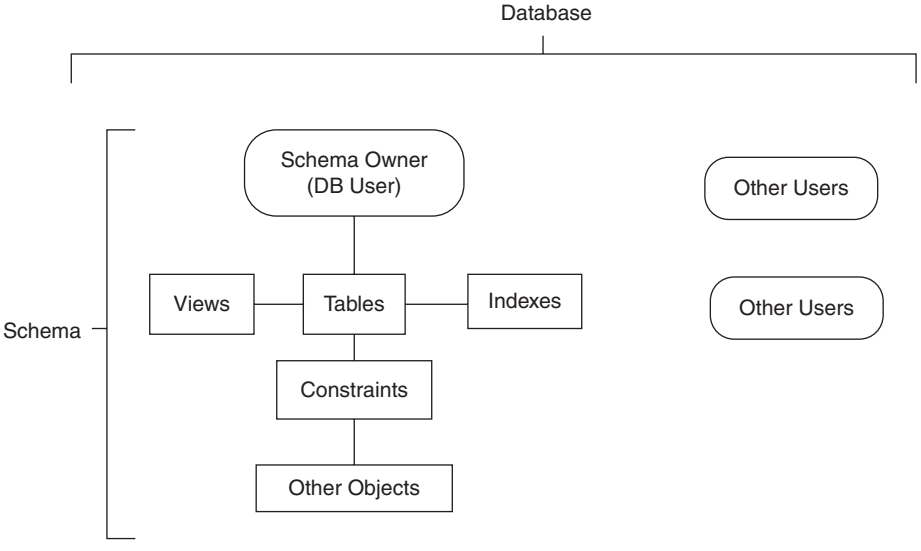

FIGURE 1.7
A schema

The Relational Database Continues to Lead the Way

The relational database has been the superior database choice for effectively managing data for several decades, and it continues to dominate the market share. This is true for many reasons:

- A well-designed relational database provides a simple, organized, easy-to-manage mechanism for data.
- A relational database is scalable as data grows and objects are added to the database.
- Linking data in multiple tables is easy.
- Maintaining the integrity of data is simple using built-in features such as referential integrity (primary and foreign key constraints).
- The overall management of data and use of SQL to communicate with the database is simplified.
- Relevant and useful data is easy to retrieve.

SQL and the relational database management system go hand in hand; you cannot have one without the other. SQL is an English-like language that enables you to create and manage a relational database and to then easily and effectively retrieve data from the database for a specific purpose.

Examples and Exercises

Most of the exercises that you encounter in this book use Oracle, a dominant leader in the market that adheres closely to the SQL standard but also offers many enhancements.

Oracle (as well as many implementations of SQL, or the relational database), make freely distributed versions of its database available. You can then select an implementation of your choice, install it, and follow along with the exercises in the book. Note that because these databases are not 100% compliant with SQL:2016, the exercises might have slight variations of the exact syntax suggested in the ANSI standard. However, if you learn the basics of the ANSI standard, you can generally translate your skills between different database implementations.

Summary

In this hour, you got a brief introduction to the SQL language and the relational database. SQL is the standard language for communicating with any relational database. It is important to first understand what comprises the relational database, identify the key elements, know what data

looks like, and understand how the data in the relational database is related to other data within the database. When you fully grasp these concepts, your journey into the SQL language will take flight sooner and the concepts in this book will make more sense. The goal of this hour was to introduce some basic database concepts and give you a foundation for thinking about data at a higher level.

Simply put, a relational database is a database that is logically organized into multiple tables that might or might not be related to one another. A table consists of one or more columns, or fields, and one or more rows of data. A table can have just a few rows of data or millions upon millions of rows of data. Tables are related through common columns, or fields, that are defined in a relational database using constraints such as primary keys and foreign keys. These keys are built-in features of a relational database that maintain referential integrity within the database. This is the key to the relational database and one of the main components of SQL. Subsequent chapters cover all these concepts in examples and hands-on exercises.

Q&A

Q. If I learn SQL, can I use any of the implementations that use SQL?

A. Yes, you can communicate with a database whose implementation is compliant with ANSI SQL. If an implementation is not completely compliant, you likely will pick it up quickly with some adjustments.

Q. In a client/server environment, is the personal computer the client or the server?

A. The personal computer is the client in a client/server environment, although a server can also serve as a client.

Q. In the context of a database, how do information and usable data differ?

A. Information and data are conceptually the same. However, when you are creating a database, it is important to gather all required information and work to form that information into usable data. Usable data comes from reliable sources and is accurate, consistent, clean, and up-to-date. Usable data is critical for business operations and when making critical decisions. Data that is derived from inaccurate sources or that has not properly been entered or evolved with the database is not usable.

Q. Can SQL be used to integrate databases among multiple environments, platforms, or types of databases?

A. When working with a relational database, SQL is the primary language and tool for working with data, including integrating data among multiple sources. SQL is also necessary when migrating data between sources. Medium to large organizations commonly have many different sources of data, only some of which are SQL based. SQL is an important component for pulling everything together.

Workshop

The following workshop consists of a series of quiz questions and practical exercises. The quiz questions are designed to test your overall understanding of the current material. The practical exercises give you the opportunity to apply the concepts discussed during the current hour, as well as build on the knowledge you acquired in previous hours of study. Be sure to complete the quiz questions and exercises before continuing to the next hour. Refer to Appendix C, “Answers to Quizzes and Exercises,” for answers.

Quiz

1. What does the acronym SQL stand for?
2. What is a schema? Give an example.
3. How do logical and physical components within a relational database differ? How are they related?
4. What keys are used to define and enforce referential integrity in a relational database?
5. What is the most basic type of object in a relational database?
6. What elements comprise this object?
7. Must primary key column values be unique?
8. Must foreign key column values be unique?

Exercises

For the following exercises, refer to Figure 1.6, earlier in this hour.

1. Who are Mary Smith's dependents?
2. How many employees do not have dependents?
3. How many duplicate foreign key values exist in the `DEPENDENTS` table?
4. Who is Tim's parent or guardian?
5. Which employee can be deleted without first having to delete any dependent records?

Index

SYMBOLS

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

S

T

U

V

W

Y