LEARNING MADE EASY

9th Edition

SQL

for dummies®

A Wiley Brand

Learn all the features in the latest version of SQL

Build a relational database and create a management system

Secure and protect your database from corruption

Allen G. Taylor

Bestselling author of all previous editions of *SQL For Dummies*

SQL

9th Edition

by Allen G. Taylor
Author of SQL All-in-One For Dummies

for dummies®
A Wiley Brand

SQL For Dummies® 9th Edition

Published by: **John Wiley & Sons, Inc.,** 111 River Street, Hoboken, NJ 07030-5774, www.wiley.com

Published simultaneously in Canada

Library of Congress Control Number: 2018960776

ISBN: 978-1-119-52707-7 (pbk); ISBN: 978-1-119-52708-4 (ePDF); ISBN: 978-1-119-52709-1 (ePub)

Manufactured in the United States of America

10 9 8 7 6 5 4 3 2 1

Contents at a Glance

Table of Contents

Introduction

Welcome to database development using SQL, the industry-standard database query language. Many database management system (DBMS) tools run on a variety of hardware platforms. The differences among the tools can be great, but all serious products have one thing in common: They support SQL data access and manipulation. If you know SQL, you can build relational databases and get useful information out of them.

About This Book

Relational database management systems are vital to many organizations. People often think that creating and maintaining these systems must be extremely complex activities — the domain of database gurus who possess enlightenment beyond that of mere mortals. This book sweeps away the database mystique. In this book, you

- Get to the roots of databases.
- Find out how a DBMS is structured.
- Discover the major functional components of SQL.
- Build a database.
- Protect a database from harm.
- Operate on database data.
- Determine how to get the information you want out of a database.

The purpose of this book is to help you build relational databases and get valuable information out of them by using SQL. SQL is the international standard language used to create and maintain relational databases. This edition covers the latest version of the standard, SQL:2016.

This book doesn't tell you how to design a database (I do that in *Database Development For Dummies*, also published by Wiley). Here I assume that you or somebody else has already created a valid design. I then illustrate how you implement that

design by using SQL. If you suspect that you don't have a good database design, then by all means fix your design before you try to build the database. The earlier you detect and correct problems in a development project, the cheaper the corrections will be.

Foolish Assumptions

If you need to store or retrieve data from a DBMS, you can do a much better job with a working knowledge of SQL. You don't need to be a programmer to use SQL, and you don't need to know programming languages, such as Java, C, or BASIC. SQL's syntax is like that of English. If you *are* a programmer, you can incorporate SQL into your programs. SQL adds powerful data manipulation and retrieval capabilities to conventional languages. This book tells you what you need to know to use SQL's rich assortment of tools and features inside your programs.

Icons Used in This Book

When something in this book is particularly valuable, we go out of our way to make sure that it stands out. We use these cool icons to mark text that (for one reason or another) *really* needs your attention. Here's a quick preview of the ones waiting for you in this book and what they mean.


TIP

Tips save you a lot of time and keep you out of trouble.


REMEMBER

Pay attention to the information marked by this icon — you may need it later.


WARNING

Heeding the advice that this icon points to can save you from major grief. Ignore it at your peril.


TECHNICAL STUFF

This icon alerts you to the presence of technical details that are interesting but not absolutely essential to understanding the topic being discussed.

Beyond the Book

In addition to the content in this book, you'll find some extra content available at the www.dummies.com website:

- **For the Cheat Sheet for this book, visit** www.dummies.com/ and search for SQL For Dummies 9E cheat sheet.
- **For updates to this book, if any, visit the** www.dummies.com store and search for SQL For Dummies 9E.

Where to Go from Here

Now for the fun part! Databases are the best tools ever invented for keeping track of the things you care about. After you understand databases and can use SQL to make them do your bidding, you wield tremendous power. Co-workers come to you when they need critical information. Managers seek your advice. Youngsters ask for your autograph. But most importantly, you know, at a very deep level, how your organization really works.

1 Getting Started with SQL

IN THIS PART. . .

The essentials of relational databases

Basic SQL concepts

Fundamental database tools

IN THIS CHAPTER

- Organizing information
- Defining "database" in digital terms
- Deciphering DBMS
- Looking at the evolution of database models
- Defining "*relational* database" (can you relate?)
- Considering the challenges of database design

Chapter 1

Relational Database Fundamentals

SQL (pronounced *ess-que-ell,* not *see'qwl,* though database geeks still argue about that) is a language specifically designed with databases in mind. SQL enables people to create databases, add new data to them, maintain the data in them, and retrieve selected parts of the data. Developed in the 1970s at IBM, SQL has grown and advanced over the years to become the industry standard. It is governed by a formal standard maintained by the International Standards Organization (ISO).

Various kinds of databases exist, each adhering to a different model of how the data in the database is organized.

SQL was originally developed to operate on data in databases that follow the *relational model.* Recently, the international SQL standard has incorporated part of the *object model,* resulting in hybrid structures called object-relational databases. In this chapter, I discuss data storage, devote a section to how the relational model

compares with other major models, and provide a look at the important features of relational databases.

Before I talk about SQL, however, I want to nail down what I mean by the term *database.* Its meaning has changed, just as computers have changed the way people record and maintain information.

Keeping Track of Things

Today people use computers to perform many tasks formerly done with other tools. Computers have replaced typewriters for creating and modifying documents. They've surpassed calculators as the best way to do math. They've also replaced millions of pieces of paper, file folders, and file cabinets as the principal storage medium for important information. Compared with those old tools, of course, computers do much more, much faster — and with greater accuracy. These increased benefits do come at a cost, however: Computer users no longer have direct physical access to their data.

When computers occasionally fail, office workers may wonder whether computerization really improved anything at all. In the old days, a manila file folder "crashed" only if you dropped it — then you merely knelt down, picked up the papers, and put them back in the folder. Barring earthquakes or other major disasters, file cabinets never "went down," and they never gave you an error message. A hard-drive crash is another matter entirely: You can't "pick up" lost bits and bytes. Mechanical, electrical, and human failures can make your data go away into the Great Beyond, never to return. Backing up your data frequently is one thing you can do to enhance your peace of mind. Another thing you can do is store your data in the cloud and let your cloud provider do the backing up.

Taking the necessary precautions to protect yourself from accidental data loss allows you to start cashing in on the greater speed and accuracy that computers provide.

If you're storing important data, you have four main concerns:

- Storing data must be quick and easy because you're likely to do it often.
- The storage medium must be reliable. You don't want to come back later and find some (or all) of your data missing.

- Data retrieval must be quick and easy, regardless of how many items you store.
- You need an easy way to separate the exact information you want *now* from the tons of data that you *don't* want right now.

State-of-the-art computer databases satisfy these four criteria. If you store more than a dozen or so data items, you probably want to store those items in a database.

What Is a Database?

The term *database* has fallen into loose use lately, losing much of its original meaning. To some people, a database is any collection of data items (phone books, laundry lists, parchment scrolls . . . whatever). Other people define the term more strictly.

In this book, I define a *database* as a self-describing collection of integrated records. And yes, that does imply computer technology, complete with programming languages such as SQL.

REMEMBER

A *record* is a representation of some physical or conceptual object. Say, for example, that you want to keep track of a business's customers. You assign a record for each customer. Each record has multiple *attributes,* such as name, address, and telephone number. Individual names, addresses, and so on are the *data.*

A database consists of both data and *metadata.* Metadata is the data that describes the data's structure within a database. If you know how your data is arranged, then you can retrieve it. Because the database contains a description of its own structure, it's *self-describing.* The database is *integrated* because it includes not only data items but also the relationships among data items.

The database stores metadata in an area called the *data dictionary,* which describes the tables, columns, indexes, constraints, and other items that make up the database.

Because a flat-file system (described later in this chapter) has no metadata, applications written to work with flat files must contain the equivalent of the metadata as part of the application program.

Database Size and Complexity

Databases come in all sizes, from simple collections of a few records to mammoth systems holding millions of records. Most databases fall into one of three categories, which are based on the size of the database itself, the size of the equipment it runs on, and the size of the organization that is maintaining it:

- A **personal database** is designed for use by a single person on a single computer. Such a database usually has a rather simple structure and a relatively small size.
- A **departmental or workgroup database** is used by the members of a single department or workgroup within an organization. This type of database is generally larger than a personal database and is necessarily more complex; such a database must handle multiple users trying to access the same data at the same time.
- An **enterprise database** can be huge. Enterprise databases may model the critical information flow of entire large organizations.

What Is a Database Management System?

Glad you asked. A *database management system* (DBMS) is a set of programs used to define, administer, and process databases and their associated applications. The database being managed is, in essence, a structure that you build to hold valuable data. A DBMS is the tool you use to build that structure and operate on the data contained within the database.

You can find many DBMS programs on the market today. Some run on large and powerful machines, and some on personal computers, notebooks, and tablets. Some even run on smartphones. A strong trend, however, is for such products to work on multiple platforms or on networks that contain different classes of machines. An even stronger trend is to store data in data centers or even to store it out in the *cloud,* which could be a public cloud run by a large company such as Amazon, Google, or Microsoft, via the Internet, or it could be a private cloud operated by the same organization that is storing the data on its own intranet.

These days, *cloud* is a buzzword that is bandied about incessantly in techie circles. Like the puffy white things up in the sky, it has indistinct edges and seems to float somewhere out there. In reality, it is a collection of computing resources that is accessible via a browser, either over the Internet or on a private intranet. The thing that distinguishes the computing resources in the cloud from similar

computing resources in a physical data center is the fact that the resources are accessible via a browser rather than an application program that directly accesses those resources.

REMEMBER

A DBMS that runs on platforms of multiple classes, large and small, is called *scalable.*

Whatever the size of the computer that hosts the database — and regardless of whether the machine is connected to a network — the flow of information between database and user is always the same. Figure 1-1 shows that the user communicates with the database through the DBMS. The DBMS masks the physical details of the database storage so that the application need only concern itself with the logical characteristics of the data, not with how the data is stored.

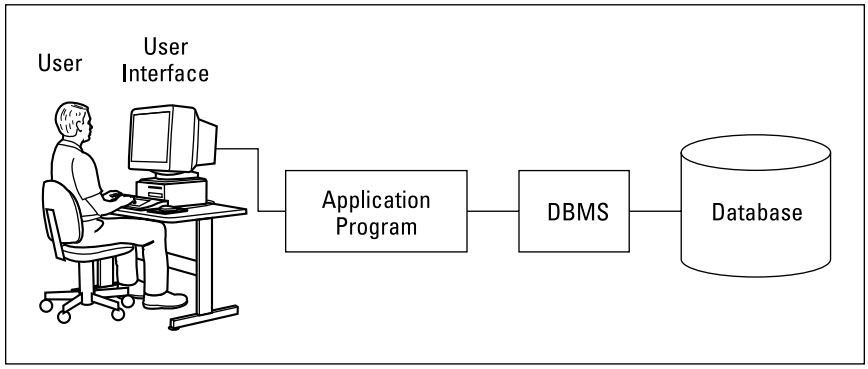

FIGURE 1-1: A block diagram of a DBMS-based information system.

THE VALUE IS NOT IN THE DATA, BUT IN THE STRUCTURE

Years ago, some clever person calculated that if you reduce human beings to their components of carbon, hydrogen, oxygen, and nitrogen atoms (plus traces of others), they would be worth only 97 cents. However droll this assessment, it's misleading. People aren't composed of mere isolated collections of atoms. Our atoms combine into enzymes, proteins, hormones, and many other substances that would cost millions of dollars per ounce on the pharmaceutical market. The precise structure of these combinations of atoms is what gives them greater value. By analogy, database structure makes possible the interpretation of seemingly meaningless data. The structure brings to the surface patterns, trends, and tendencies in the data. Unstructured data — like uncombined atoms — has little or no value.

Flat Files

Where structured data is concerned, the flat file is as simple as it gets. No, a flat file isn't a folder that's been squashed under a stack of books. *Flat files* are so called because they have minimal structure. If they were buildings, they'd barely stick up from the ground. A flat file is simply a collection of data records, one after another, in a specified format — the data, the whole data, and nothing but the data — in effect, a list. In computer terms, a flat file is simple. Because the file doesn't store structural information (metadata), its overhead (stuff in the file that is not data but takes up storage space) is minimal.

Say that you want to keep track of the names and addresses of your company's customers in a flat file system. The system may have a structure something like this:

```
Harold Percival  26262 S. Howards Mill Rd  Westminster  CA92683
Jerry Appel      32323 S. River Lane Rd    Santa Ana    CA92705
Adrian Hansen    232 Glenwood Court        Anaheim      CA92640
John Baker       2222 Lafayette St         Garden Grove CA92643
Michael Pens     77730 S. New Era Rd       Irvine       CA92715
Bob Michimoto    25252 S. Kelmsley Dr      Stanton      CA92610
Linda Smith      444 S.E. Seventh St       Costa Mesa   CA92635
Robert Funnell   2424 Sheri Court          Anaheim      CA92640
Bill Checkal     9595 Curry Dr             Stanton      CA92610
Jed Style        3535 Randall St           Santa Ana    CA92705
```

As you can see, the file contains nothing but data. Each field has a fixed length (the `Name` field, for example, is always exactly 15 characters long), and no structure separates one field from another. The person who created the database assigned field positions and lengths. Any program using this file must "know" how each field was assigned, because that information is not contained in the database itself.

Such low overhead means that operating on flat files can be very fast. On the minus side, however, application programs must include logic that manipulates the file's data at a very detailed level. The application must know exactly where and how the file stores its data. Thus, for small systems, flat files work fine. The larger a system is, however, the more cumbersome a flat-file system becomes.

TIP

Using a database instead of a flat-file system eliminates duplication of effort. Although database files themselves may have more overhead, the applications can be more portable across various hardware platforms and operating systems. A database also makes writing application programs easier because the programmer doesn't need to know the physical details of where and how the data is stored.