Introduction

"I presume you're mortal, and may err."
—James Shirley
The Lady of Pleasure

If you've used a computer more than casually, you have probably used Structured Query Language, or SQL—perhaps without even knowing it. SQL is *the* standard language for communicating with most database systems. Any time you import data into a spreadsheet or perform a merge into a word processing program, you're most likely using SQL in some form or another. Every time you go online to an e-commerce site on the Web and place an order for a book, a recording, a movie, or any of the dozens of other products you can order, there's a very high probability that the code behind the Web page you're using is accessing its databases with SQL. If you need to get information from a database system that uses SQL, you can enhance your understanding of the language by reading this book.

Are You a Mere Mortal?

You might ask, "Who is a *mere mortal*? Me?" The answer is not simple. When we started to write this book, we thought we were experts in the database language called SQL. Along the way, we discovered we were mere mortals too, in several areas. We understood a few specific implementations of SQL very well, but we unraveled many of the complex intricacies of the language as we studied how it is used in many commercial products. So, if you fit any of the following descriptions, you're a mere mortal too!

- If you use computer applications that let you access information from a database system, you're probably a mere mortal. The first time you don't get the information you expected using the query tools built in to your application, you'll need to explore the underlying SQL statements to find out why.

- If you have recently discovered one of the many available desktop database applications but are struggling with defining and querying the data you need, you're a mere mortal.
- If you're a database programmer who needs to "think out of the box" to solve some complex problems, you're a mere mortal.
- If you're a database guru in one product but are now faced with integrating the data from your existing system into another system that supports SQL, you're a mere mortal.

In short, *anyone* who has to use a database system that supports SQL can use this book. As a beginning database user who has just discovered that the data you need can be fetched using SQL, you will find that this book teaches you all the basics and more. For an expert user who is suddenly faced with solving complex problems or integrating multiple systems that support SQL, this book will provide insights into leveraging the complex abilities of the SQL database language.

About This Book

Everything you read in this book is based on the current International Organization for Standardization (ISO) Standard for the SQL database language (document ISO/IEC 9075-2:2003), as currently implemented in most of the popular commercial database systems. The ISO document was also adopted by the American National Standards Institute (ANSI), so this is truly an international standard. The SQL you'll learn here *is not* specific to any particular software product.

As you'll learn in more detail in Chapter 3, A Concise History of SQL, the SQL Standard defines both more and less than you'll find implemented in most commercial database products. Most database vendors have yet to implement many of the more advanced features, but most do support the core of the standard.

We researched a wide range of popular products to make sure that you can use what we're teaching in this book. When we found parts of the core of the language not supported by some major products, we warned you in the text and showed you alternate ways to state your database requests in standard SQL. When we found significant parts of the SQL Standard supported by only a few vendors, we introduced you to the syntax and then suggested alternatives.

We have organized this book into five major sections.

- Part I, Relational Databases and SQL, explains how modern database systems are based on a rigorous mathematical model and provides a brief history of the database query language that has evolved into what we know as SQL. We also discuss some simple rules that you can use to make sure your database design is sound.
- Part II, SQL Basics, introduces you to using the SELECT statement, creating expressions, and sorting information with an ORDER BY clause. You'll also learn how to filter data by using a WHERE clause.
- Part III, Working with Multiple Tables, shows you how to formulate queries that draw data from more than one table. Here we show you how to link tables in a query using the INNER JOIN, OUTER JOIN, and UNION operators, and how to work with subqueries.
- Part IV, Summarizing and Grouping Data, discusses how to obtain summary information and group and filter summarized data. Here is where you'll learn about the GROUP BY and HAVING clauses.
- Part V, Modifying Sets of Data, explains how to write queries that modify a set of rows in your tables. In the chapters in this section, you'll learn how to use the UPDATE, INSERT, and DELETE statements.

At the end of the book in the appendices, you'll find syntax diagrams for all the SQL elements you've learned, layouts of the sample databases, a list of date and time manipulation functions implemented in five of the major database systems, and book recommendations to further your study of SQL. There is also a CD containing all the sample databases used throughout the book in several different formats.

What This Book Is Not

Although this book is based on the 2003 SQL Standard that was current at the time of this writing (a 2007/2008 draft standard is in the works), it does not cover every aspect of the standard. In truth, many features in the 2003 SQL Standard won't be implemented for many years—if at all—in the major database system implementations. The fundamental purpose of this book is to give you a solid grounding in writing queries in SQL. Throughout the book, you'll find us recommending that you "consult your database documentation" for how a specific feature might or might not work. That's not to say we covered only the lowest common denominator for any feature among the major database systems. We do try to caution you when some systems implement a feature differently or not at all.

You'll find it difficult to create other than simple queries using a single table if your database design is flawed. We included a chapter on database design to help you identify when you will have problems, but that one chapter includes only the basic principles. A thorough discussion of database design principles and how to implement a design in a specific database system is beyond the scope of this book.

This book is also not about how to solve a problem in the most efficient way. As you work through many of the later chapters, you'll find we suggest more than one way to solve a particular problem. In some cases where writing a query in a particular way is likely to have performance problems on any system, we try to warn you about it. But each database system has its own strengths and weaknesses. After you learn the basics, you'll be ready to move on to digging into the particular database system you use to learn how to formulate your query solutions so that they run in a more optimal manner.

How to Use This Book

We have designed the chapters in this book to be read in sequence. Each succeeding chapter builds on concepts taught in earlier chapters. However, you can jump into the middle of the book without getting lost. For example, if you are already familiar with the basic clauses in a SELECT statement and want to learn more about JOINs, you can jump right in to Chapters 7 Thinking in Sets, 8 INNER JOINS, and 9 OUTER JOINS.

At the end of many of the chapters you'll find an extensive set of sample problems, their solutions, and sample result sets. We recommend that you study several of the samples to gain a better understanding of the techniques involved and then try solving some of the later samples yourself without looking at the solutions we propose.

Note that where a particular query returns dozens of rows in the result set, we show you only the first few rows in this book to give you an idea of how the answer should look. You might not see the exact same result on your system, however, because each database system that supports SQL has its own optimizer that figures out the fastest way to solve the query. Also, the first few rows you see returned by your database system might not exactly match the first few we show you unless the query contains an ORDER BY clause that requires the rows to be returned in a specific sequence.

We've also included a complete set of problems for you to solve on your own, which you'll find at the end of most chapters. This gives you the opportunity to really practice what you've just learned in the chapter. Don't worry—the solutions are included in the sample databases on the CD. We've also included hints on those problems that might be a little tricky.

After you have worked your way through the entire book, you'll find the complete SQL diagrams in Appendix A to be an invaluable reference for all the SQL techniques we showed you. You will also be able to use the sample database layouts in Appendix B to help you design your own databases.

Reading the Diagrams Used in This Book

The numerous diagrams throughout the book illustrate the proper syntax for the statements, terms, and phrases you'll use when you work with SQL. Each diagram provides a clear picture of the overall construction of the SQL element currently being discussed. You can also use any of these diagrams as templates to create your own SQL statements or to help you acquire a clearer understanding of a specific example.

All the diagrams are built from a set of core elements and can be divided into two categories: *statements* and *defined terms*. A statement is always a major SQL operation, such as the SELECT statement we discuss in this book, while a defined term is always a component used to build part of a statement, such as a *value expression, a search condition,* or a *conditional expression.* (Don't worry—we'll explain all these terms later in the book.) The only difference between a syntax diagram for a statement and a syntax diagram for a defined term is the manner in which the main syntax line begins and ends. We designed the diagrams with these differences so that you can clearly see whether you're looking at the diagram for an entire statement or a diagram for a term that you might use within a statement. Figure 1 (on page xxviii) shows the beginning and end points for both diagram categories. Aside from this difference, the diagrams are built from the same elements. Figure 2 (on page xxviii) shows an example of each type of syntax diagram and is followed by a brief explanation of each diagram element.

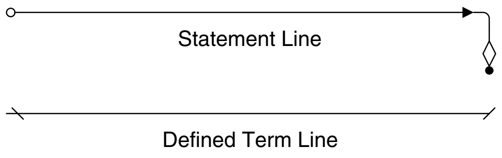

Figure 1 *Syntax line end points for statements and defined terms*

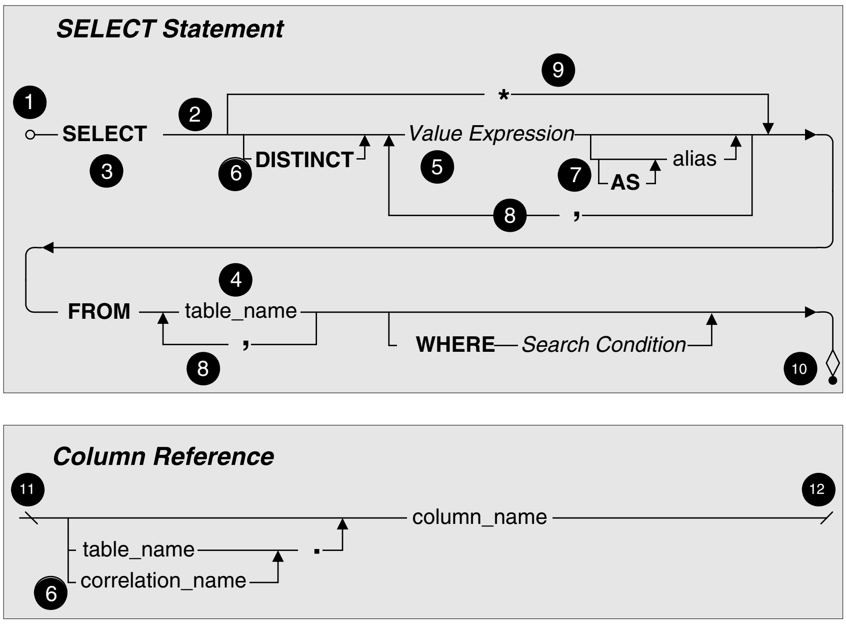

Figure 2 *Sample statement and defined term diagrams*

1. ***Statement start point***—denotes the beginning of the main syntax line for a statement. Any element that appears *directly on* the main syntax line is a *required element*, and any element that appears *below* it is an *optional element*.
2. ***Main syntax line***—determines the order of all required and optional elements for the statement or defined term. Follow this line from left to right (or in the direction of the arrows) to build the syntax for the statement or defined term.
3. ***Keyword(s)***—indicates a major word in SQL grammar that is a required part of the syntax for a statement or defined term. In a diagram, keywords are formatted in capital letters and bold font. (You don't have to worry about typing a keyword in capital letters when you actually write the statement in your database program, but it does make the statement easier to read.)

4. ***Literal entry***—specifies the name of a value you explicitly supply to the statement. A literal entry is represented by a word or phrase that indicates the type of value you need to supply. Literal entries in a diagram are formatted in all lowercase letters.
5. ***Defined term***—denotes a word or phrase that represents some operation that returns a final value to be used in this statement. We'll explain and diagram every defined term you need to know as you work through the book. Defined terms are always formatted in italic letters.
6. ***Optional element***—indicates any element or group of elements that appears below the main syntax line. An optional element can be a statement, keyword, defined term, or literal value and, for purposes of clarity, is placed on its own line. In some cases, you can specify a set of values for a given option, with each value separated by a comma (see number 8). Also, several optional elements have a set of sub-optional elements (see number 7). In general, you read the syntax line for an optional element from left to right, in the same manner that you read the main syntax line. Always follow the directional arrows and you'll be in good shape. Note that some options allow you to specify multiple values or choices, so the arrow will flow from right to left. After you've entered all the items you need, however, the flow will return to normal from left to right. Fortunately, all optional elements work the same way. After we show you how to use an optional element later in the book, you'll know how to use any other optional element you encounter in a syntax diagram.
7. ***Sub-optional element***—denotes any element or group of elements that appears below an optional element. Sub-optional elements allow you to fine-tune your statements so that you can work with more complex problems.
8. ***Option list separator***—indicates that you can specify more than one value for this option and that each value must be separated with a comma.
9. ***Alternate option***—denotes a keyword or defined term that can be used as an alternative to one or more optional elements. The syntax line for an alternate option will bypass the syntax lines of the optional elements it is meant to replace.
10. ***Statement end point***—denotes the end of the main syntax line for a statement.
11. ***Defined term start point***—denotes the beginning of the main syntax line for a defined term.
12. ***Defined term end point***—denotes the end of the main syntax line for a defined term.

Now that you're familiar with these elements, you'll be able to read all the syntax diagrams in the book. And on those occasions when a diagram requires further explanation, we provide you with the information you need to read the diagram clearly and easily. To help you better understand how the diagrams work, here's a sample SELECT statement that we built using Figure 2.

```
SELECT FirstName, LastName, City, DOB AS DateOfBirth
FROM Students
WHERE City = 'El Paso'
```

This SELECT statement retrieves four columns from the Students table, as we've indicated in the SELECT and FROM clauses. As you follow the main syntax line from left to right, you see that you have to indicate at least one *value expression*. A value expression can be a column name, an expression created using column names, or simply a constant (literal) value that you want to display. You can indicate as many columns as you need with the value expression's *option list separator* (a comma). This is how we were able to use four column names from the Student table. We were concerned that some people viewing the information returned by this SELECT statement might not know what DOB means, so we assigned an *alias* to the DOB column with the value expression's AS sub-option. Finally, we used the WHERE clause to make certain the SELECT statement shows only those students who live in El Paso. (If this doesn't quite make sense to you just now, there's no cause for alarm. You'll learn all this in great detail throughout the remainder of the book.)

You'll find a full set of syntax diagrams in Appendix A. They show the complete and proper syntax for all the statements and defined terms we discuss in the book. If you happen to refer to these diagrams as you work through each chapter, you'll notice a slight disparity between some of the diagrams in a given chapter and the corresponding diagrams in the appendix. The diagrams in the chapters are just simplified versions of the diagrams in the appendix. These simplified versions allow us to explain complex statements and defined terms more easily and give us the ability to focus on particular elements as needed. But don't worry—all the diagrams in the appendix will make perfect sense after you work through the material in the book.

Sample Databases Used in This Book

Bound into the back of the book, you'll find a CD-ROM containing five sample databases that we use for the example queries throughout the book. We've also included diagrams of the database structures in Appendix B: Schema for the Sample Databases.

1. **Sales Orders**. This is a typical order entry database for a store that sells bicycles and accessories. (Every database book needs at least *one* order entry example, right?)
2. **Entertainment Agency**. We structured this database to manage entertainers, agents, customers, and bookings. You would use a similar design to handle event bookings or hotel reservations.
3. **School Scheduling**. You might use this database design to register students at a high school or community college. This database tracks not only class registrations but also which instructors are assigned to each class and what grades the students received.
4. **Bowling League**. This database tracks bowling teams, team members, the matches they played, and the results.
5. **Recipes**. You can use this database to save and manage all your favorite recipes. We even added a few that you might want to try.

On the sample CD, you can find all five sample databases in four different formats.

- Because of the great popularity of the Microsoft Office Access desktop database, we created one set of databases (.mdb file extension) using Microsoft Access 2000 (Version 9.0). We chose Version 9 of this product because it closely supports the current ISO/IEC SQL Standard, and you can open database files in this format using Access 2000, 2002 (XP), 2003, and 2007. You can find these files in the MSAccess subfolder.
- The second format consists of database files (.mdf file extension) created using Microsoft SQL Server 2000. We have also included SQL command files (.sql file extension) and batch files (.bat file extension) that you can use to attach the samples to a Microsoft SQL Server catalog. You can also attach these files to a Microsoft SQL Server 2005 server. You can find these files in the MSSQLServer subfolder. You can obtain a free copy of Microsoft SQL Server 2005 Express Edition at http://msdn.microsoft.com/vstudio/express/sql/download/default.aspx.
- We created the third set of databases using the popular open-source MySQL version 5 database system. You can either point your InnoDB data directory to the MySQL subfolder or use the scripts (.sql file extension) you can also find in that folder to create the database structure, load the data, and create the sample views in your own MySQL data folder. You can obtain a free copy of the community edition of the MySQL database system at http://www.mysql.com/.

- The fourth format is a series of SQL scripts that you can modify and use with any major database system that supports SQL. You can find scripts to define the schema (the tables) of each database, to load the data using INSERT statements, and to create the queries using CREATE VIEW statements in the SQLScripts subfolder. Although we created these scripts using utilities in Microsoft SQL Server, we simplified them to make them generic for use with most database systems.

To install the sample files, see the file ReadMe.txt in the root folder of the sample CD. If you mount the sample CD on an Apple Macintosh system, you will find only the sample files for MySQL and the SQL scripts.

❖ **Note** Although we were very careful to use the most common and simplest syntax for the CREATE TABLE, CREATE INDEX, CREATE CONSTRAINT, and INSERT commands in the sample SQL scripts, you (or your database administrator) might need to modify these files slightly to work with your database system. If you're working with a database system on a remote server, you might need to gain permission from your database administrator to build the samples from the SQL commands we supplied.

For the chapters in Parts II, III, and IV that focus on the SELECT statement, you'll find all the example statements and solutions in the "example" version of each sample database (e.g., SalesOrdersExample, EntertainmentAgency Example). Because the examples in Part V modify the sample data, we created "modify" versions of each of the sample databases (e.g., SalesOrdersModify, EntertainmentAgencyModify). The sample databases for Part V also include additional columns and tables not found in the SELECT examples that enable us to demonstrate certain features of UPDATE, INSERT, and DELETE queries.

"Follow the Yellow Brick Road"

—Munchkin to Dorothy in *The Wizard of Oz*

Now that you've read through the Introduction, you're ready to start learning SQL, right? Well, maybe. At this point, you're still in the house, it's still being tossed about by the tornado, and you haven't left Kansas.

Before you make that jump to Chapter 4, Creating a Sample Query, take our advice and read through the first three chapters. Chapter 1, What Is Relational?, will give you an idea of how the relational database was conceived and how it has grown to be the most widely used type of database in the industry today. We hope this will give you some amount of insight into the database system you're currently using. In Chapter 2, Ensuring Your Database Structure Is Sound, you'll learn how to fine-tune your data structures so that your data is reliable and, above all, accurate. You're going to have a tough time working with some of the SQL statements if you have poorly designed data structures, so we suggest you read this chapter carefully.

Chapter 3 is literally the beginning of the "yellow brick road." Here you'll learn the origins of SQL and how it evolved into its current form. You'll also learn about some of the people and companies who helped pioneer the language and why there are so many varieties of SQL. Finally, you'll learn how SQL came to be a national and international standard and what the outlook for SQL will be in the years to come.

After you've read these chapters, consider yourself well on your way to Oz. Just follow the road we've laid out through each of the remaining chapters. When you've finished the book, you'll find that you've found the wizard—and he is you.