Ryan Stephens Ron Plew Arie D. Jones

FIFTH EDITION
Includes Coverage of
Oracle and
Microsoft SQL
Implementations

Sams Teach Yourself

SQL

n 24 Hours

SAMS

FREE SAMPLE CHAPTER

SHARE WITH OTHERS











Ryan Stephens Ron Plew Arie D. Jones

Sams Teach Yourself

SQL



FIFTH EDITION



Sams Teach Yourself SQL in 24 Hours. Fifth Edition

Copyright © 2011 by Pearson Education, Inc.

All rights reserved. No part of this book shall be reproduced, stored in a retrieval system, or transmitted by any means, electronic, mechanical, photocopying, recording, or otherwise, without written permission from the publisher. No patent liability is assumed with respect to the use of the information contained herein. Although every precaution has been taken in the preparation of this book, the publisher and author assume no responsibility for errors or omissions. Nor is any liability assumed for damages resulting from the use of the information contained herein.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

ISBN-13: 978-0-672-33541-9 ISBN-10: 0-672-33541-7

The Library of Congress cataloging-in-publication data is on file.

Printed in the United States of America

Fourth Printing: May 2013

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 authors 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 or from the programs accompanying it.

Bulk Sales

Sams Publishing offers excellent discounts on this book when ordered in quantity for bulk purchases or special sales. For more information, please contact

U.S. Corporate and Government Sales 1-800-382-3419 corpsales@pearsontechgroup.com

For sales outside of the U.S., please contact

International Sales international@pearson.com

Associate Publisher

Mark Taub

Acquisitions Editor

Trina MacDonald

Development Editor

Michael Thurston

Managing Editor

Kristy Hart

Project Editor

Jovana San Nicolas-Shirlev

Copy Editor

The Wordsmithery LLC

Indexer

Lisa Stumpf

Proofreader

Gill Editorial Services

Technical Editor

Benjamin Schupak

Publishing Coordinator

Olivia Basegio

Book Designer

Gary Adair

Composition

Gloria Schurick

Contents at a Glance

Part I: An	SQL Concepts Overview	
HOUR 1	Welcome to the World of SQL	1
Part II: Bu	ilding Your Database	
HOUR 2	Defining Data Structures	21
3	Managing Database Objects	37
4	The Normalization Process	. 61
5	Manipulating Data	73
6	Managing Database Transactions	87
Part III: Ge	etting Effective Results from Queries	
HOUR 7	Introduction to the Database Query	99
8	Using Operators to Categorize Data	115
9	Summarizing Data Results from a Query	141
10	Sorting and Grouping Data	153
11	Restructuring the Appearance of Data	169
12	Understanding Dates and Times	191
Part IV: Bu	uilding Sophisticated Database Queries	
HOUR 13	Joining Tables in Queries	207
14	Using Subqueries to Define Unknown Data	225
15	Combining Multiple Queries into One	239
Part V: SQ	L Performance Tuning	
HOUR 16	Using Indexes to Improve Performance	255
17	Improving Database Performance	267

Teach Yourself SQL in 24 Hours

Part VI: Us	sing SQL to Manage Users and Security	
HOUR 18	Managing Database Users	285
19	Managing Database Security	299
Part VII: S	ummarized Data Structures	
HOUR 20	Creating and Using Views and Synonyms	313
21	Working with the System Catalog	329
Part VIII: A	Applying SQL Fundamentals in Today's World	
HOUR 22	Advanced SQL Topics	339
23	Extending SQL to the Enterprise, the Internet, and the Intranet	355
24	Extensions to Standard SQL	367
Part IX: Ap	pendixes	
A	Common SQL Commands	377
В	Using the Databases for Exercises	383
C	Answers to Quizzes and Exercises	391
D	CREATE TABLE Statements for Book Examples	439
E	INSERT Statements for Data in Book Examples	443
F	Glossary	451
G	Bonus Exercises	455

Table of Contents

Part I: An SQL Concepts Overview

HUU	JR 1: Welcome to the World of SQL	1
	SQL Definition and History	1
	SQL Sessions	8
	Types of SQL Commands	9
	The Database Used in This Book	12
	Summary	17
	Q&A	17
	Workshop	18
Part	t II: Building Your Database	
HOU	JR 2: Defining Data Structures	21
	What Is Data?	21
	Basic Data Types	22
	Summary	
	Q&A.	31
	Workshop	32
HOU	JR 3: Managing Database Objects	37
	What Are Database Objects?	37
	What Is a Schema?	37
	Tables: The Primary Storage for Data	39
	Integrity Constraints	49
	Summary	54
	Q&A	55
	Workshop	55

Teach Yourself SQL in 24 Hours

HOUR 4: The Normalization Process	61
Normalizing a Database	61
Denormalizing a Database	69
Summary	70
Q&A	70
Workshop	71
HOUR 5: Manipulating Data	73
Overview of Data Manipulation	73
Populating Tables with New Data	74
Updating Existing Data	80
Deleting Data from Tables	82
Summary	83
Q&A	83
Workshop	84
HOUR 6: Managing Database Transactions	87
What Is a Transaction?	87
Controlling Transactions	88
Transactional Control and Database Performance	95
Summary	96
Q&A	96
Workshop	97
Part III: Getting Effective Results from Queries	
HOUR 7: Introduction to the Database Query	99
What Is a Query?	99
Introduction to the SELECT Statement	99
Examples of Simple Queries	108
Summary	112
Q&A	112
Workshop	113

Contents

HOUR 8: Using Operators to Categorize Data	115
What Is an Operator in SQL?	115
Comparison Operators	116
Logical Operators	119
Conjunctive Operators	126
Negative Operators	129
Arithmetic Operators	133
Summary	
Q&A	137
Workshop	137
HOUR 9: Summarizing Data Results from a Query	141
What Are Aggregate Functions?	141
Summary	150
Q&A	150
Workshop	150
HOUR 10: Sorting and Grouping Data	153
HOUR 10: Sorting and Grouping Data Why Group Data?	
	153
Why Group Data?	
Why Group Data? The GROUP BY Clause	153 154 159
Why Group Data? The GROUP BY Clause GROUP BY Versus ORDER BY	153 154 159 161
Why Group Data? The GROUP BY Clause GROUP BY Versus ORDER BY CUBE and ROLLUP Expressions	153 154 159 161 164
Why Group Data? The GROUP BY Clause GROUP BY Versus ORDER BY CUBE and ROLLUP Expressions The HAVING Clause	153 154 159 161 164 165
Why Group Data? The GROUP BY Clause GROUP BY Versus ORDER BY CUBE and ROLLUP Expressions The HAVING Clause Summary	153 154 159 161 164 165
Why Group Data? The GROUP BY Clause GROUP BY Versus ORDER BY CUBE and ROLLUP Expressions The HAVING Clause Summary Q&A	153 154 159 161 164 165
Why Group Data? The GROUP BY Clause GROUP BY Versus ORDER BY CUBE and ROLLUP Expressions The HAVING Clause Summary Q&A Workshop	153 154 159 161 164 165 166 166
Why Group Data? The GROUP BY Clause GROUP BY Versus ORDER BY CUBE and ROLLUP Expressions The HAVING Clause Summary Q&A Workshop HOUR 11: Restructuring the Appearance of Data	153 154 159 161 164 165 166 166 169
Why Group Data? The GROUP BY Clause GROUP BY Versus ORDER BY CUBE and ROLLUP Expressions The HAVING Clause Summary Q&A Workshop HOUR 11: Restructuring the Appearance of Data ANSI Character Functions	153 154 159 161 164 165 166 166 169 170
Why Group Data? The GROUP BY Clause GROUP BY Versus ORDER BY CUBE and ROLLUP Expressions The HAVING Clause Summary Q&A Workshop HOUR 11: Restructuring the Appearance of Data ANSI Character Functions Common Character Functions	153 154 159 161 164 165 166 166 169 170 179

Teach Yourself SQL in 24 Hours

	Combining Character Functions	186
	Summary	187
	Q&A	188
	Workshop	188
Н	OUR 12: Understanding Dates and Times	191
	How Is a Date Stored?	191
	Date Functions	193
	Date Conversions	198
	Summary	204
	Q&A	204
	Workshop	205
Н	OUR 13: Joining Tables in Queries	207
Н	OUR 13: Joining Tables in Queries	207
	Selecting Data from Multiple Tables	207
	Understanding Joins	208
	Join Considerations	217
	Summary	221
	Q&A	222
	Workshop	222
Н	OUR 14: Using Subqueries to Define Unknown Data	225
	What Is a Subquery?	225
	Embedded Subqueries	231
	Correlated Subqueries	233
	Subquery Performance	234
	Summary	235
	Q&A	235
	Workshop	236

Contents

HOUR 15: Combining Multiple Queries into One	239
Single Queries Versus Compound Queries	239
Compound Query Operators	240
Using ORDER BY with a Compound Query	246
Using GROUP BY with a Compound Query	248
Retrieving Accurate Data	250
Summary	250
Q&A	250
Workshop	251
Part V: SQL Performance Tuning	
HOUR 16: Using Indexes to Improve Performance	255
What Is an Index?	255
How Do Indexes Work?	256
The CREATE INDEX Command	257
Types of Indexes	258
When Should Indexes Be Considered?	260
When Should Indexes Be Avoided?	261
Altering an Index	263
Dropping an Index	263
Summary	264
Q&A	264
Workshop	265
HOUR 17: Improving Database Performance	267
What Is SQL Statement Tuning?	267
Database Tuning Versus SQL Statement Tuning	268
Formatting Your SQL Statement	268
Full Table Scans	274
Other Performance Considerations	275
Cost-Based Optimization	279
Performance Tools	280
Summary	280

Teach Yourself SQL in 24 Hours

Q&A	281
Workshop	281
Part VI: Using SQL to Manage Users and Security	
HOUR 18: Managing Database Users	285
User Management in the Database	285
The Management Process	288
Tools Utilized by Database Users	296
Summary	296
Q&A	297
Workshop	297
HOUR 19: Managing Database Security	299
What Is Database Security?	299
What Are Privileges?	301
Controlling User Access	304
Controlling Privileges Through Roles	308
Summary	310
Q&A	310
Workshop	311
Part VII: Summarized Data Structures	
HOUR 20: Creating and Using Views and Synonyms	313
What Is a View?	313
Creating Views	316
WITH CHECK OPTION	320
Creating a Table from a View	321
Views and the ORDER BY Clause	322
Updating Data Through a View	322
Dropping a View	323
Performance Impact of Using Nested Views	323
What Is a Synonym?	324

Contents

Summary	325
Q&A	326
Workshop	326
HOUR 21: Working with the System Catalog	329
What Is the System Catalog?	329
How Is the System Catalog Created?	331
What Is Contained in the System Catalog?	331
System Catalog Tables by Implementation	333
Querying the System Catalog	334
Updating System Catalog Objects	336
Summary	337
Q&A	337
Workshop	338
HOUR 22: Advanced SQL Topics Cursors	339
Cursors	
Stored Procedures and Functions	
Triggers	
Dynamic SQL	
Call-Level Interface	
Using SQL to Generate SQL	
Direct Versus Embedded SQL	
Windowed Table Functions	351
Working with XML	
Summary	
	353
Q&A	353
Q&A Workshop	353
	353
Workshop	353 353 354 355

Teach Yourself SQL in 24 Hours

	SQL and the Internet	360
	SQL and the Intranet	361
	Summary	362
	Q&A	363
	Workshop	363
НО	UR 24: Extensions to Standard SQL	367
	Various Implementations	367
	Example Extensions	370
	Interactive SQL Statements	373
	Summary	374
	Q&A	375
	Workshop	375
Paı	rt IX: Appendixes	
API	PENDIX A: Common SQL Commands	377
	SQL Statements	377
	SQL Clauses	381
API	PENDIX B: Using the Databases for Exercises	383
	Windows Installation Instructions for MySQL	383
	Windows Installation Instructions for Oracle	386
	Windows Installation Instructions for Microsoft SQL Server	388
API	PENDIX C: Answers to Quizzes and Exercises	391
API	PENDIX D: CREATE TABLE Statements for Book Examples	439
API	PENDIX E: INSERT Statements for Data in Book Examples	443
API	PENDIX F: Glossary	451
API	PENDIX G: Bonus Exercises	455
IND	DEX	461

About the Author

For more than 10 years, the authors have studied, applied, and documented the SQL standard and its application to critical database systems in this book.

Ryan Stephens and **Ron Plew** are entrepreneurs, speakers, and cofounders of Perpetual Technologies, Inc. (PTI), a fast-growing IT management and consulting firm. PTI specializes in database technologies, primarily Oracle and SQL servers running on all UNIX, Linux, and Microsoft platforms. Starting out as data analysts and database administrators, Ryan and Ron now lead a team of impressive technical subject matter experts who manage databases for clients worldwide. They authored and taught database courses for Indiana University-Purdue University in Indianapolis for five years and have authored more than a dozen books on Oracle, SQL, database design, and high availability of critical systems.

Arie D. Jones is the principal technology manager for Perpetual Technologies, Inc. (PTI) in Indianapolis, Indiana. Arie leads PTI'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 pertaining to database-related topics.

Dedication

This book is dedicated to my parents, Thomas and Karlyn Stephens, who always taught me that I can achieve anything if determined. This book is also dedicated to my brilliant son, Daniel, and to my beautiful daughters, Autumn and Alivia; don't ever settle for anything less than your dreams.

—Ryan

This book is dedicated to my family: my wife, Linda; my mother, Betty; my children, Leslie, Nancy, Angela, and Wendy; my grandchildren, Andy, Ryan, Holly, Morgan, Schyler, Heather, Gavin, Regan, Caleigh, and Cameron; and my sons-in-law, Jason and Dallas. Thanks for being patient with me during this busy time. Love all of you.

—Рорру

I would like to dedicate this book to my wife, Jackie, for being understanding and supportive during the long hours that it took to complete this book.

—Arie

Acknowledgments

Thanks to all the people in our lives who have been patient during all editions of this book—mostly to our wives, Tina and Linda. Thanks to Arie Jones for stepping up to the plate and helping so much with this edition. Thanks also to the editorial staff at Sams for all of their hard work to make this edition better than the last. It has been a pleasure to work with each of you.

—Ryan and Ron

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.

You can email or write me directly to let me know what you did or didn't like about this book—as well as what we can do to make our books stronger.

Please note that I cannot help you with technical problems related to the topic of this book, and that due to the high volume of mail I receive, I might not be able to reply to every message.

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

Email: opensource@samspublishing.com

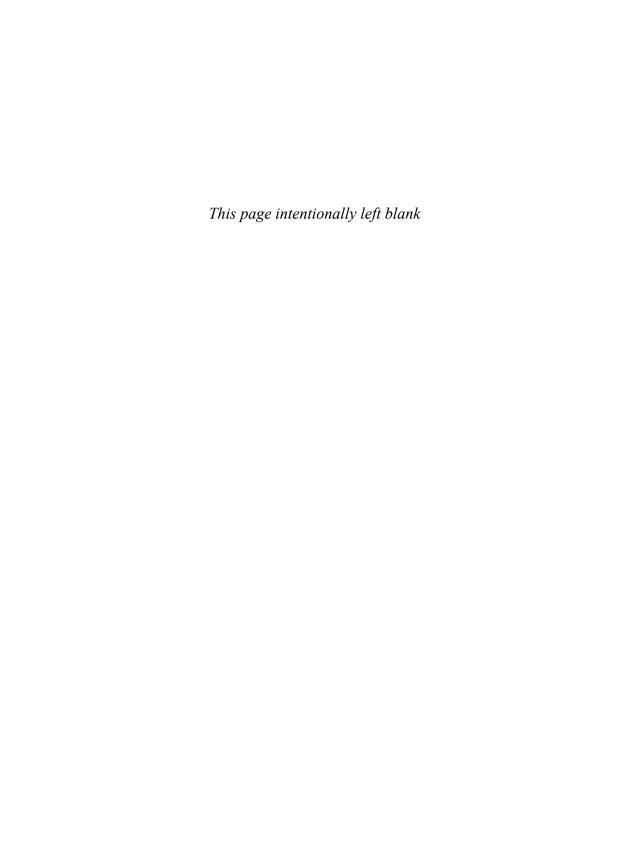
Mail: Mark Taub

Associate Publisher Sams Publishing 800 East 96th Street

Indianapolis, IN 46240 USA

Reader Services

Visit our website and register this book at informit.com/register for convenient access to any updates, downloads, or errata that might be available for this book.



HOUR 3

Managing Database Objects

What You'll Learn in This Hour:

- ► An introduction to database objects
- An introduction to the schema
- ▶ An introduction to the table
- ▶ A discussion of the nature and attributes of tables
- ► Examples for the creation and manipulation of tables
- ► A discussion of table storage options
- ► Concepts on referential integrity and data consistency

In this hour, you learn about database objects: what they are, how they act, how they are stored, and how they relate to one another. Database objects are the logical units that compose the building blocks of the database. The majority of the instruction during this hour revolves around the table, but keep in mind that there are other database objects, many of which are discussed in later hours of study.

What Are Database Objects?

A database object is any defined object in a database that is used to store or reference data. Some examples of database objects include tables, views, clusters, sequences, indexes, and synonyms. The table is this hour's focus because it is the primary and simplest form of data storage in a relational database.

What Is a Schema?

A *schema* is a collection of database objects normally associated with one particular database username. This username is called the *schema owner*, or the owner of the related group of objects. You may have one or multiple schemas in a database. The user is only associated with the schema of the

same name, and often the terms are used interchangeably. Basically, any user who creates an object has just created it in her own schema unless she specifically instructs it to be created in another one. So, based on a user's privileges within the database, the user has control over objects that are created, manipulated, and deleted. A schema can consist of a single table and has no limits to the number of objects that it may contain, unless restricted by a specific database implementation.

Say you have been issued a database username and password by the database administrator. Your username is USER1. Suppose you log on to the database and then create a table called EMPLOYEE_TBL. According to the database, your table's actual name is USER1. EMPLOYEE_TBL. The schema name for that table is USER1, which is also the owner of that table. You have just created the first table of a schema.

The good thing about schemas is that when you access a table that you own (in your own schema), you do not have to refer to the schema name. For instance, you could refer to your table as either one of the following:

```
EMPLOYEE_TBL
USER1.EMPLOYEE_TBL
```

The first option is preferred because it requires fewer keystrokes. If another user were to query one of your tables, the user would have to specify the schema as follows:

```
USER1.EMPLOYEE TBL
```

In Hour 20, "Creating and Using Views and Synonyms," you learn about the distribution of permissions so that other users can access your tables. You also learn about synonyms, which enable you to give a table another name so you do not have to specify the schema name when accessing a table. Figure 3.1 illustrates two schemas in a relational database.

There are, in Figure 3.1, two user accounts in the database that own tables: USER1 and USER2. Each user account has its own schema. Some examples for how the two users can access their own tables and tables owned by the other user follow:

USER1 accesses own TABLE1: TABLE1

USER1 accesses own TEST: TEST

USER1 accesses USER2's TABLE10: USER2.TABLE10

USER1 accesses USER2's TEST: USER2.TEST

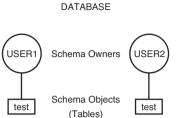


table10

table20

table1

table2

FIGURE 3.1 Schemas in a database.

In this example, both users have a table called TEST. Tables can have the same names in a database as long as they belong to different schemas. If you look at it this way, table names are always unique in a database because the schema owner is actually part of the table name. For instance, USER1.TEST is a different table than USER2.TEST. If you do not specify a schema with the table name when accessing tables in a database, the database server looks for a table that you own by default. That is, if USER1 tries to access TEST, the database server looks for a USER1-owned table named TEST before it looks for other objects owned by USER1, such as synonyms to tables in another schema. Hour 21, "Working with the System Catalog," helps you fully understand how synonyms work.

You must be careful to understand the distinction between objects in your own schema and those objects in another schema. If you do not provide a schema when performing operations that alter the table, such as a DROP command, the database assumes that you mean a table in your own schema. This could possibly lead to your unintentionally dropping the wrong object. So you must always pay careful attention as to which user you are currently logged into the database with.

Object Naming Rules Differ Between Systems

Every database server has rules concerning how you can name objects and elements of objects, such as field names. You must check your particular implementation for the exact naming conventions or rules.



Tables: The Primary Storage for Data

The table is the primary storage object for data in a relational database. In its simplest form, a table consists of row(s) and column(s), both of which hold the data. A table takes up physical space in a database and can be permanent or temporary.

Columns

A *field*, also called a *column* in a relational database, is part of a table that is assigned a specific data type. The data type determines what kind of data the column is allowed to hold. This enables the designer of the table to help maintain the integrity of the data.

Every database table must consist of at least one column. Columns are those elements within a table that hold specific types of data, such as a person's name or address. For example, a valid column in a customer table might be the customer's name. Figure 3.2 illustrates a column in a table.

FIGURE 3.2 An example of a column.



Generally, a column name must be one continuous string and can be limited to the number of characters used according to each implementation of SQL. It is typical to use underscores with names to provide separation between characters. For example, a column for the customer's name can be named CUSTOMER_NAME instead of CUSTOMERNAME. This is normally done to increase the readability of database objects. There are other naming conventions that you can utilize, such as Camel Case, to fit your specific preferences. As such, it is important for a database development team to agree upon a standard naming convention and stick to it so that order is maintained within the development process.

The most common form of data stored within a column is string data. This data can be stored as either uppercase or lowercase for character-defined fields. The case that you use for data is simply a matter of preference, which should be based on how the data will be used. In many cases, data is stored in uppercase for simplicity and consistency. However, if data is stored in different case types throughout the database (uppercase, lowercase, and mixed case), functions can be applied to convert the data to either uppercase or lowercase if needed. These functions are covered in Hour 11, "Restructuring the Appearance of Data."

Columns also can be specified as NULL or NOT NULL, meaning that if a column is NOT NULL, something must be entered. If a column is specified as NULL, nothing has to be entered. NULL is different from an empty set, such as

an empty string, and holds a special place in database design. As such, you can relate a NULL value to a lack of any data in the field.

Rows

A row is a record of data in a database table. For example, a row of data in a customer table might consist of a particular customer's identification number, name, address, phone number, and fax number. A row is composed of fields that contain data from one record in a table. A table can contain as little as one row of data and up to as many as millions of rows of data or records. Figure 3.3 illustrates a row within a table.

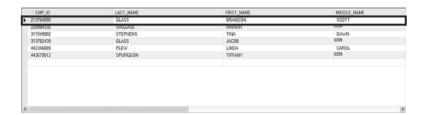


FIGURE 3.3 Example of a table row.

The CREATE TABLE Statement

The CREATE TABLE statement in SQL is used to create a table. Although the very act of creating a table is quite simple, much time and effort should be put into planning table structures before the actual execution of the CREATE TABLE statement. Carefully planning your table structure before implementation saves you from having to reconfigure things after they are in production.

Types We Use in This Hour

In this hour's examples, we use the popular data types CHAR (constant-length character), VARCHAR (variable-length character), NUMBER (numeric values, decimal, and nondecimal), and DATE (date and time values).



Some elementary questions need to be answered when creating a table:

- ▶ What type of data will be entered into the table?
- ▶ What will be the table's name?
- ▶ What column(s) will compose the primary key?
- ▶ What names shall be given to the columns (fields)?

- ▶ What data type will be assigned to each column?
- ▶ What will be the allocated length for each column?
- ▶ Which columns in a table can be left as a null value?



Existing Systems Often Have Existing Naming Rules

Be sure to check your implementation for rules when naming objects and other database elements. Often database administrators adopt a *naming convention* that explains how to name the objects within the database so you can easily discern how they are used.

After these questions are answered, the actual CREATE TABLE statement is simple.

The basic syntax to create a table is as follows:

```
CREATE TABLE table_name
( field1 data_type [ not null ],
  field2 data_type [ not null ],
  field3 data_type [ not null ],
  field4 data_type [ not null ],
  field5 data type [ not null ] );
```

Note that a semicolon is the last character in the previous statement. Also, brackets indicate portions that are optional. Most SQL implementations have some character that terminates a statement or submits a statement to the database server. Oracle, Microsoft SQL Server, and MySQL use the semicolon. Although Transact-SQL, Microsoft SQL Server's ANSI SQL version, has no such requirement, it is considered best practice to use it. This book uses the semicolon.

Create a table called $\mbox{EMPLOYEE_TBL}$ in the following example using the syntax for \mbox{MySQL} :

```
CREATE TABLE EMPLOYEE TBL
(EMP ID CHAR(9)
                        NOT NULL,
EMP NAME VARCHAR (40) NOT NULL,
EMP ST ADDR VARCHAR (20) NOT NULL,
EMP_CITY VARCHAR (15)
                         NOT NULL,
EMP ST
           CHAR(2)
                         NOT NULL,
EMP_ZIP
           INTEGER(5)
                         NOT NULL,
EMP PHONE
            INTEGER(10)
                         NULL,
EMP PAGER INTEGER (10)
                         NULL);
```

The following code would be the compatible code for both Microsoft SQL Server and Oracle:

```
CREATE TABLE EMPLOYEE TBL
(EMP ID CHAR(9)
                          NOT NULL,
EMP NAME
           VARCHAR (40)
                          NOT NULL,
EMP_ST_ADDR VARCHAR (20)
                          NOT NULL,
EMP CITY VARCHAR (15)
                          NOT NULL,
EMP ST
           CHAR(2)
                          NOT NULL,
EMP_ZIP
            INTEGER
                          NOT NULL.
EMP PHONE
            INTEGER
                          NULL,
EMP PAGER
            INTEGER
                          NULL);
```

Eight different columns make up this table. Notice the use of the underscore character to break the column names up into what appears to be separate words (EMPLOYEE ID is stored as EMP_ID). This is a technique that is used to make table or column name more readable. Each column has been assigned a specific data type and length, and by using the NULL/NOT NULL constraint, you have specified which columns require values for every row of data in the table. The EMP_PHONE is defined as NULL, meaning that NULL values are allowed in this column because there might be individuals without a telephone number. The information concerning each column is separated by a comma, with parentheses surrounding all columns (a left parenthesis before the first column and a right parenthesis following the information on the last column).

Limitations on Data Types Vary

Check your particular implementation for name length limits and characters that are allowed; they could differ from implementation to implementation.

Watch

```
Each record, or row of data, in this table consists of the following: 
EMP_ID, EMP_NAME, EMP_ST_ADDR, EMP_CITY, EMP_ST, EMP_ZIP, EMP_PHONE, EMP_PAGER
```

In this table, each field is a column. The column EMP_ID could consist of one employee's identification number or many employees' identification numbers, depending on the requirements of a database query or transaction.

Naming Conventions

When selecting names for objects, specifically tables and columns, make sure the name reflects the data that is to be stored. For example, the name for a table pertaining to employee information could be named EMPLOYEE_TBL. Names for columns should follow the same logic. When storing an employee's phone number, an obvious name for that column would be PHONE NUMBER.

The ALTER TABLE Command

You can modify a table after the table has been created by using the ALTER TABLE command. You can add column(s), drop column(s), change column definitions, add and drop constraints, and, in some implementations, modify table STORAGE values. The standard syntax for the ALTER TABLE command follows:

```
alter table table_name [modify] [column column_name][datatype | null not
null]
[restrict | cascade]
[drop] [constraint constraint_name]
[add] [column] column definition
```

Modifying Elements of a Table

The *attributes* of a column refer to the rules and behavior of data in a column. You can modify the attributes of a column with the ALTER TABLE command. The word *attributes* here refers to the following:

- ▶ The data type of a column
- ▶ The length, precision, or scale of a column
- ▶ Whether the column can contain NULL values

The following example uses the ALTER TABLE command on EMPLOYEE_TBL to modify the attributes of the column EMP_ID:

```
ALTER TABLE EMPLOYEE_TBL MODIFY EMP_ID VARCHAR(10); Table altered.
```

The column was already defined as data type VARCHAR (a varying-length character), but you increased the maximum length from 9 to 10.

Adding Mandatory Columns to a Table

One of the basic rules for adding columns to an existing table is that the column you are adding cannot be defined as NOT NULL if data currently exists in the table. NOT NULL means that a column must contain some value for every row of data in the table. So, if you are adding a column defined as NOT NULL, you are contradicting the NOT NULL constraint right off the bat if the preexisting rows of data in the table do not have values for the new column.

There is, however, a way to add a mandatory column to a table:

1. Add the column and define it as NULL. (The column does not have to contain a value.)

- **2.** Insert a value into the new column for every row of data in the table
- 3. Alter the table to change the column's attribute to NOT NULL.

Adding Auto-Incrementing Columns to a Table

Sometimes it is necessary to create a column that auto-increments itself to give a unique sequence number for a particular row. You could do this for many reasons, such as not having a natural key for the data, or wanting to use a unique sequence number to sort the data. Creating an auto-incrementing column is generally quite easy. In MySQL, the implementation provides the SERIAL method to produce a truly unique value for the table. Following is an example:

Using NULL for Table Creation

NULL is a default attribute for a column; therefore, it does not have to be entered in the CREATE TABLE statement. NOT NULL must always be specified.

By the Way

In Microsoft SQL Server, we are provided with an IDENTITY column type. The following is an example for the SQL Server implementation:

Oracle does not provide a direct method for an auto-incrementing column. However, there is one method using an object called a SEQUENCE and a TRIGGER that simulates the effect in Oracle. This technique is discussed when we talk about TRIGGERs in Hour 22, "Advanced SQL Topics."

Now we can insert values into the newly created table without specifying a value for our auto-incrementing column:

Modifying Columns

You need to consider many things when modifying existing columns of a table. Following are some common rules for modifying columns:

- ► The length of a column can be increased to the maximum length of the given data type.
- ▶ The length of a column can be decreased only if the largest value for that column in the table is less than or equal to the new length of the column.
- The number of digits for a number data type can always be increased.
- ► The number of digits for a number data type can be decreased only if the value with the most number of digits for that column is less than or equal to the new number of digits specified for the column.
- The number of decimal places for a number data type can either be increased or decreased.
- ▶ The data type of a column can normally be changed.

Some implementations might actually restrict you from using certain ALTER TABLE options. For example, you might not be allowed to drop columns from a table. To do this, you have to drop the table itself and then rebuild the table with the desired columns. You could run into problems by dropping a column in one table that is dependent on a column in another table or dropping a column that is referenced by a column in another table. Be sure to refer to your specific implementation documentation.



Creating Tables for Exercises

You will create the tables that you see in these examples at the end of this hour in the "Exercises" section. In Hour 5, "Manipulating Data," you will populate the tables you create in this hour with data.

Creating a Table from an Existing Table



Altering or Dropping Tables Can Be Dangerous

Take heed when altering and dropping tables. If you make logical or typing mistakes when issuing these statements, you can lose important data.

You can create a copy of an existing table using a combination of the CREATE TABLE statement and the SELECT statement. The new table has the same column definitions. You can select any or all columns. New columns that you create via functions or a combination of columns automatically assume the size necessary to hold the data. The basic syntax for creating a table from another table is as follows:

```
create table new_table_name as
select [ *|column1, column2 ]
from table_name
[ where ]
```

Notice some new keywords in the syntax, particularly the SELECT keyword. SELECT is a database query and is discussed in more detail in Chapter 7, "Introduction to the Database Query." However, it is important to know that you can create a table based on the results from a query.

Both MySQL and Oracle support the CREATE TABLE AS SELECT method of creating a table based on another table. Microsoft SQL Server, however, uses a different statement. For that database implementation, you use a SELECT ... INTO statement. This statement is used like this:

```
select [ *|column1, columnn2]
into new_table_name
from table_name
[ where ]
```

Here you'll examine some examples of using this method.

First, do a simple query to view the data in the PRODUCTS_TBL table:

select * from products tbl;

PROD_ID	PROD_DESC	COST
11235	WITCH COSTUME	29.99
222	PLASTIC PUMPKIN 18 INCH	7.75
13	FALSE PARAFFIN TEETH	1.1
90	LIGHTED LANTERNS	14.5
15	ASSORTED COSTUMES	10
9	CANDY CORN	1.35
6	PUMPKIN CANDY	1.45
87	PLASTIC SPIDERS	1.05
119	ASSORTED MASKS	4.95

Next, create a table called PRODUCTS_TMP based on the previous query:

```
create table products_tmp as
select * from products_tbl;
```

Table created.

In SQL Server, the same statement would be written as such:

```
select *
into products_tmp
from products_tbl;
```

Table created.

Now if you run a query on the PRODUCTS_TMP table, your results appear the same as if you had selected data from the original table.

select *

from prod	ucts_tmp;	
PROD_ID	PROD_DESC	COST
11235	WITCH COSTUME	29.99
222	PLASTIC PUMPKIN 18 INCH	7.75
13	FALSE PARAFFIN TEETH	1.1
90	LIGHTED LANTERNS	14.5
15	ASSORTED COSTUMES	10
9	CANDY CORN	1.35
6	PUMPKIN CANDY	1.45
87	PLASTIC SPIDERS	1.05
119	ASSORTED MASKS	4.95



What the * Means

SELECT * selects data from all fields in the given table. The * represents a complete row of data, or record, in the table.



Default STORAGE Attributes for Tables

When creating a table from an existing table, the new table takes on the same STORAGE attributes as the original table.

Dropping Tables

Dropping a table is actually one of the easiest things to do. When the RESTRICT option is used and the table is referenced by a view or constraint, the DROP statement returns an error. When the CASCADE option is used, the drop succeeds and all referencing views and constraints are dropped. The syntax to drop a table follows:

```
drop table table_name [ restrict | cascade ]
```

SQL Server does not allow for the use of the CASCADE option. So for that particular implementation, you must ensure that you drop all objects that reference the table you are removing to ensure that you are not leaving an invalid object in your system.

In the following example, you drop the table that you just created: drop table products tmp:

Table dropped.

Be Specific When Dropping a Table

Whenever you're dropping a table, be sure to specify the schema name or owner of the table before submitting your command. You could drop the incorrect table. If you have access to multiple user accounts, ensure that you are connected to the database through the correct user account before dropping tables.



Integrity Constraints

Integrity constraints ensure accuracy and consistency of data in a relational database. Data integrity is handled in a relational database through the concept of referential integrity. Many types of integrity constraints play a role in *referential integrity (RI)*.

Primary Key Constraints

Primary key is the term that identifies one or more columns in a table that make a row of data unique. Although the primary key typically consists of one column in a table, more than one column can comprise the primary key. For example, either the employee's Social Security number or an assigned employee identification number is the logical primary key for an employee table. The objective is for every record to have a unique primary key or value for the employee's identification number. Because there is probably no need to have more than one record for each employee in an employee table, the employee identification number makes a logical primary key. The primary key is assigned at table creation.

The following example identifies the EMP_ID column as the PRIMARY KEY for the EMPLOYEES table:

```
CREATE TABLE EMPLOYEE_TBL
(EMP_ID CHAR(9) NOT NULL PRIMARY KEY,
```

```
EMP_NAMEVARCHAR (40)NOT NULL,EMP_ST_ADDRVARCHAR (20)NOT NULL,EMP_CITYVARCHAR (15)NOT NULL,EMP_STCHAR(2)NOT NULL,EMP_ZIPINTEGER(5)NOT NULL,EMP_PHONEINTEGER(10)NULL,EMP_PAGERINTEGER(10)NULL);
```

This method of defining a primary key is accomplished during table creation. The primary key in this case is an implied constraint. You can also specify a primary key explicitly as a constraint when setting up a table, as follows:

```
CREATE TABLE EMPLOYEE_TBL

(EMP_ID CHAR(9) NOT NULL,

EMP_NAME VARCHAR (40) NOT NULL,

EMP_ST_ADDR VARCHAR (20) NOT NULL,

EMP_CITY VARCHAR (15) NOT NULL,

EMP_ST CHAR(2) NOT NULL,

EMP_ZIP INTEGER(5) NOT NULL,

EMP_PHONE INTEGER(10) NULL,

EMP_PAGER INTEGER(10) NULL,

PRIMARY KEY (EMP_ID));
```

The primary key constraint in this example is defined after the column comma list in the CREATE TABLE statement.

You can define a primary key that consists of more than one column by either of the following methods, which demonstrate creating a primary key in an Oracle table:

```
CREATE TABLE PRODUCT_TST

(PROD_ID VARCHAR2(10) NOT NULL,

VEND_ID VARCHAR2(10) NOT NULL,

PRODUCT VARCHAR2(30) NOT NULL,

COST NUMBER(8,2) NOT NULL,

PRIMARY KEY (PROD_ID, VEND_ID));

ALTER TABLE PRODUCTS_TST

ADD CONSTRAINT PRODUCTS_PK PRIMARY KEY (PROD_ID, VEND_ID);
```

Unique Constraints

A unique column constraint in a table is similar to a primary key in that the value in that column for every row of data in the table must have a unique value. Although a primary key constraint is placed on one column, you can place a unique constraint on another column even though it is not actually for use as the primary key.

Study the following example:

```
CREATE TABLE EMPLOYEE TBL
          CHAR(9) NOT NULL
VARCHAR (40) NOT NULL,
(EMP ID
                                        PRIMARY KEY,
EMP NAME
EMP_ST_ADDR VARCHAR (20) NOT NULL,
EMP CITY
            VARCHAR (15) NOT NULL,
EMP ST
            CHAR(2)
                           NOT NULL,
EMP ZIP
            INTEGER(5)
                          NOT NULL,
EMP_PHONE
            INTEGER (10)
                                        UNIQUE.
                           NULL
EMP PAGER
            INTEGER(10)
                           NULL);
```

The primary key in this example is EMP_ID, meaning that the employee identification number is the column ensuring that every record in the table is unique. The primary key is a column that is normally referenced in queries, particularly to join tables. The column EMP_PHONE has been designated as a UNIQUE value, meaning that no two employees can have the same telephone number. There is not a lot of difference between the two, except that the primary key provides an order to data in a table and, in the same respect, joins related tables.

Foreign Key Constraints

A foreign key is a column in a child table that references a primary key in the parent table. A foreign key constraint is the main mechanism that enforces referential integrity between tables in a relational database. A column defined as a foreign key references a column defined as a primary key in another table.

Study the creation of the foreign key in the following example:

```
CREATE TABLE EMPLOYEE_PAY_TST

(EMP_ID CHAR(9) NOT NULL,

POSITION VARCHAR2(15) NOT NULL,

DATE_HIRE DATE NULL,

PAY_RATE NUMBER(4,2) NOT NULL,

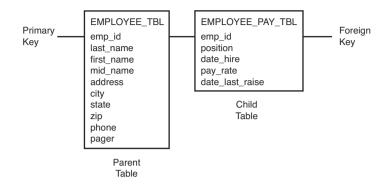
DATE_LAST_RAISE DATE NULL,

CONSTRAINT EMP_ID_FK FOREIGN KEY (EMP_ID) REFERENCES EMPLOYEE_TBL (EMP_ID));
```

The EMP_ID column in this example has been designated as the foreign key for the EMPLOYEE_PAY_TBL table. This foreign key, as you can see, references the EMP_ID column in the EMPLOYEE_TBL table. This foreign key ensures that for every EMP_ID in the EMPLOYEE_PAY_TBL, there is a corresponding EMP_ID in the EMPLOYEE_TBL. This is called a *parent/child relationship*. The parent table is the EMPLOYEE_TBL table, and the child table is the EMPLOYEE_PAY_TBL table. Study Figure 3.4 for a better understanding of the parent table/child table relationship.

FIGURE 3.4

The parent/child table relationship.



In this figure, the EMP_ID column in the child table references the EMP_ID column in the parent table. For a value to be inserted for EMP_ID in the child table, a value for EMP_ID in the parent table must exist. Likewise, for a value to be removed for EMP_ID in the parent table, all corresponding first values for EMP_ID must be removed from the child table. This is how referential integrity works.

You can add a foreign key to a table using the ALTER TABLE command, as shown in the following example:

```
alter table employee_pay_tbl
add constraint id_fk foreign key (emp_id)
references employee_tbl (emp_id);
```

By the Way

ALTER TABLE Variations

The options available with the ALTER TABLE command differ among implementations of SQL, particularly when dealing with constraints. In addition, the actual use and definitions of constraints vary, but the concept of referential integrity should be the same with all relational databases.

NOT NULL Constraints

Previous examples use the keywords NULL and NOT NULL listed on the same line as each column and after the data type. NOT NULL is a constraint that you can place on a table's column. This constraint disallows the entrance of NULL values into a column; in other words, data is required in a NOT NULL column for each row of data in the table. NULL is generally the default for a column if NOT NULL is not specified, allowing NULL values in a column.

Check Constraints

You can utilize check (CHK) constraints to check the validity of data entered into particular table columns. Check constraints provide back-end database edits, although edits are commonly found in the front-end application as well. General edits restrict values that can be entered into columns or objects, whether within the database or on a front-end application. The check constraint is a way of providing another protective layer for the data.

The following example illustrates the use of a check constraint in Oracle:

```
CREATE TABLE EMPLOYEE CHECK TST
(EMP_ID
             CHAR(9)
                         NOT NULL,
EMP NAME
             VARCHAR2(40) NOT NULL,
EMP ST ADDR VARCHAR2(20) NOT NULL,
           VARCHAR2(15)
EMP CITY
                            NOT NULL,
EMP ST
             CHAR(2)
                            NOT NULL,
EMP ZIP
            NUMBER(5)
                            NOT NULL,
EMP PHONE
             NUMBER (10)
                            NULL,
EMP PAGER
             NUMBER (10)
                            NULL,
PRIMARY KEY (EMP ID),
CONSTRAINT CHK EMP ZIP CHECK ( EMP ZIP = '46234'));
```

The check constraint in this table has been placed on the EMP ZIP column, ensuring that all employees entered into this table have a ZIP Code of '46234'. Perhaps that is a little restricting. Nevertheless, you can see how it works

If you wanted to use a check constraint to verify that the ZIP Code is within a list of values, your constraint definition could look like the following:

```
CONSTRAINT CHK EMP ZIP CHECK ( EMP ZIP in ('46234', '46227', '46745') );
```

If there is a minimum pay rate that can be designated for an employee, you could have a constraint that looks like the following:

```
CREATE TABLE EMPLOYEE PAY TBL
(EMP ID
                   CHAR(9)
                                  NOT NULL,
POSITION
                   VARCHAR2(15) NOT NULL,
DATE HIRE
                   DATE
                                  NULL,
PAY RATE
                   NUMBER(4,2)
                                  NOT NULL,
                                  NULL,
DATE LAST RAISE
                  DATE
CONSTRAINT EMP ID FK FOREIGN KEY (EMP ID) REFERENCES EMPLOYEE TBL
(EMP ID),
CONSTRAINT CHK PAY CHECK ( PAY RATE > 12.50 ) );
```

In this example, any employee entered into this table must be paid more than \$12.50 an hour. You can use just about any condition in a check constraint, as you can with an SQL query. You learn more about these conditions in Hours 5 and 7.

Dropping Constraints

Using the ALTER TABLE command with the DROP CONSTRAINT option, you can drop any constraint that you have defined. For example, to drop the primary key constraint in the EMPLOYEES table, you can use the following command:

ALTER TABLE EMPLOYEES DROP CONSTRAINT EMPLOYEES PK;

Table altered.

Some implementations provide shortcuts for dropping certain constraints. For example, to drop the primary key constraint for a table in MySQL, you can use the following command:

ALTER TABLE EMPLOYEES DROP PRIMARY KEY;

Table altered.



Other Ways of Dealing with Constraints

Instead of permanently dropping a constraint from the database, some implementations allow you to temporarily disable constraints and then enable them later.

Summary

You have learned a little about database objects in general, but you have specifically learned about the table. The table is the simplest form of data storage in a relational database. Tables contain groups of logical information, such as employee, customer, or product information. A table is composed of various columns, with each column having attributes; those attributes mainly consist of data types and constraints, such as NOT NULL values, primary keys, foreign keys, and unique values.

You learned the CREATE TABLE command and options, such as storage parameters, that might be available with this command. You also learned how to modify the structure of existing tables using the ALTER TABLE command. Although the process of managing database tables might not be the most basic process in SQL, if you first learn the structure and nature of tables, you will more easily grasp the concept of accessing the tables, whether through data manipulation operations or database queries. In later hours, you learn about the management of other objects in SQL, such as indexes on tables and views.

Q&A

- Q. When I name a table that I am creating, is it necessary to use a suffix such as _TBL?
- **A.** Absolutely not. You do not have to use anything. For example, a table to hold employee information could be named something similar to the following, or anything else that would refer to what type of data is to be stored in that particular table:

EMPLOYEE
EMP_TBL
EMPLOYEE_TBL
EMPLOYEE_TABLE
WORKER

- Q. Why is it so important to use the schema name when dropping a table?
- A. Here's a true story about a new DBA who dropped a table. A programmer had created a table under his schema with the same name as a production table. That particular programmer left the company. His database account was being deleted from the database, but the DROP USER statement returned an error because he owned outstanding objects. After some investigation, it was determined that his table was not needed, so a DROP TABLE statement was issued.

It worked like a charm, but the problem was that the DBA was logged in as the production schema when the DROP TABLE statement was issued. The DBA should have specified a schema name, or owner, for the table to be dropped. Yes, the wrong table in the wrong schema was dropped. It took approximately eight hours to restore the production database.

Workshop

The following workshop is composed 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 are intended to afford you the opportunity to apply the concepts discussed during the current hour, as well as build upon the knowledge acquired in previous hours of study. Please take time to complete the quiz questions and exercises before continuing. Refer to Appendix C, "Answers to Quizzes and Exercises," for answers.

Quiz

1. Does the following CREATE TABLE statement work? If not, what needs to be done to correct the problem(s)? Are there limitations as to what database implementation it works in (MySQL, Oracle, SQL Server)?

```
Create table EMPLOYEE_TABLE as:
  (ssn number(9) not null,
  last_name varchar2(20) not null,
  first_name varchar2(20) not null,
  middle_name varchar2(20) not null,
  st address varchar2(30) not null,
  city char(20) not null,
  state char(2) not null,
  zip number(4) not null,
  date hired date);
```

- 2. Can you drop a column from a table?
- **3.** What statement would you issue to create a primary key constraint on the preceding EMPLOYEE_TABLE?
- **4.** What statement would you issue on the preceding EMPLOYEE_TABLE to allow the MIDDLE_NAME column to accept NULL values?
- 5. What statement would you use to restrict the people added into the preceding EMPLOYEE_TABLE to only reside in the state of New York ('NY')?
- **6.** What statement would you use to add an auto-incrementing column called EMPID to the preceding EMPLOYEE_TABLE using both the MySQL and SQL Server syntax?

Exercises

In the following exercise, you will be creating all the tables in the database to set up the environment for later. Additionally, you will be executing several commands that will allow you to investigate the table structure in an existing database. For thoroughness we have provided instructions for each of the three implementations (MySQL, Microsoft SQL Server, and Oracle) because each is slightly different in its approach.

MySQL

Bring up a command prompt and use the following syntax to log onto your local MySQL instance, replacing *username* with your username and *password* with your password. Ensure that you do not leave a space between -p and your password.

```
Mysql -h localhost -u username -ppassword
```

At the mysql> command prompt, enter the following command to tell MySQL that you want to use the database you created previously:

```
use learnsql;
```

Now go to Appendix D, "CREATE TABLE Statements for Book Examples," to get the DDL for the tables used in this book. At the mysql> prompt, enter each CREATE TABLE statement. Be sure to include a semicolon at the end of each CREATE TABLE statement. The tables that you create are used throughout the book.

At the mysql> prompt, enter the following command to get a list of your tables:

```
show tables;
```

At the mysql> prompt, use the DESCRIBE command (desc for short) to list the columns and their attributes for each one of the tables you created. For example:

```
describe employee_tbl;
describe employee_pay tbl;
```

If you have errors or typos, simply re-create the appropriate table(s). If the table was successfully created but has typos (perhaps you did not properly define a column or forgot a column), drop the table, and issue the CREATE TABLE command again. The syntax of the DROP TABLE command is as follows:

```
drop table orders_tbl;
```

Microsoft SQL Server

Bring up a command prompt and use the following syntax to log onto your local SQL Server instance, replacing username with your username and password with your password. Ensure that you do not leave a space between -p and your password.

```
SQLCMD -S localhost -U username -Ppassword
```

At the 1> command prompt, enter the following command to tell SQL Server that you want to use the database you created previously. Remember that with SQLCMD you must use the keyword GO to tell the command tool that you want the previous lines to execute.

```
1>use learnsql;
2>G0
```

Now go to Appendix D to get the DDL for the tables used in this book. At the 1> prompt, enter each CREATE TABLE statement. Be sure to include a semicolon at the end of each CREATE TABLE statement and follow up with the keyword GO to have your statement execute. The tables that you create are used throughout the book.

At the 1> prompt, enter the following command to get a list of your tables. Follow this command with the keyword GO:

```
Select name from sys.tables;
```

At the 1> prompt, use the sp_help stored procedure to list the columns and their attributes for each one of the tables you created. For example:

```
Sp_help_ employee_tbl;
Sp_help employee_pay_tbl;
```

If you have errors or typos, simply re-create the appropriate table(s). If the table was successfully created but has typos (perhaps you did not properly define a column or forgot a column), drop the table and issue the CREATE TABLE command again. The syntax of the DROP TABLE command is as follows:

```
drop table orders tbl;
```

Oracle

Bring up a command prompt, and use the following syntax to log onto your local Oracle instance. You are prompted to enter your username and password.

```
sqlplus
```

Now go to Appendix D to get the DDL for the tables used in this book. At the SQL> prompt, enter each CREATE TABLE statement. Be sure to include a semicolon at the end of each CREATE TABLE statement. The tables that you create are used throughout the book.

At the SQL> prompt, enter the following command to get a list of your tables:

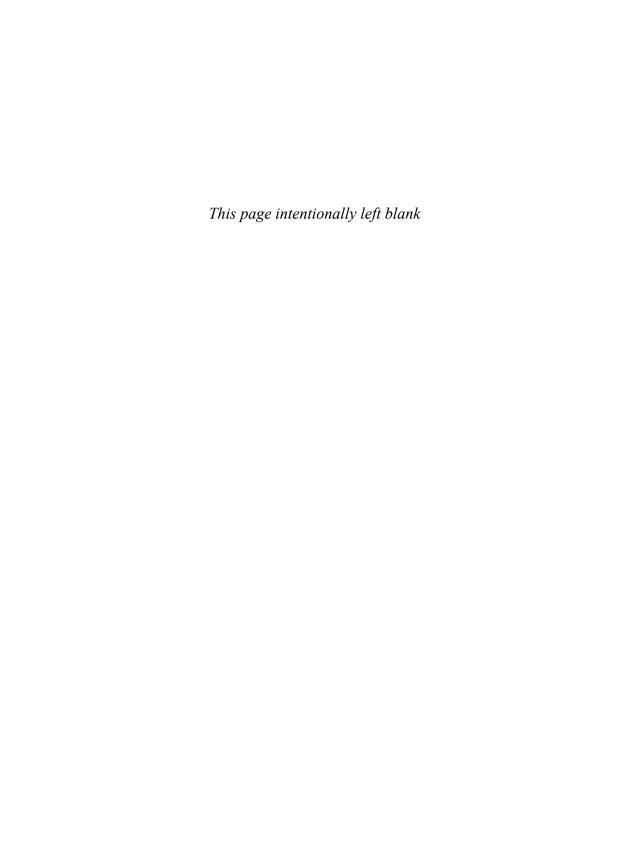
```
Select * from cat;
```

At the SQL> prompt, use the DESCRIBE command (desc for short) to list the columns and their attributes for each one of the tables you created. For example:

```
describe employee_tbl;
describe employee pay tbl;
```

If you have errors or typos, simply re-create the appropriate table(s). If the table was successfully created but has typos (perhaps you did not properly define a column or forgot a column), drop the table, and issue the CREATE TABLE command again. The syntax of the DROP TABLE command is as follows:

```
drop table orders tbl;
```



Index

A	creating groups, 155-158	
	MAX, 147-148	
ABS (absolute value), 183	MIN, 148-149	
accessing remote databases,	SUM, 145-146	
357-358	ALL, 124-126	
JDBC, 358	ALTER ANY TABLE, 302	
ODBC (Open Database	ALTER DATABASE, 302	
Connectivity), 358	ALTER TABLE, 44, 264, 377	
OLE DB, 359	ALTER USER, 302	
through web interfaces,	altering	
359-360	indexes, 263	
vendor connectivity products, 359	users, 294-295	
	American National Standards	
adding	Institute. See ANSI (American	
auto-incrementing columns to tables, 45	National Standards Institute)	
columns to tables, 44-45	AND, 126-127	
,	ANSI (American National	
time to dates, 196-197	Standards Institute), 2	
addition, arithmetic operators,	ANSI character functions,	
133-134	169-170	
ADMIN OPTION, 305	ANSI SQL, 2	
aggregate functions, 141-142	compliance with, 369	
AVG, 146-147	ANSI standard, SELECT, 368	
COUNT, 142-145		

ANY

ANY, 124-126

arithmetic operators, 133 COALESCE, 181 addition, 133-134 call-level interface (CLI), 349-350 collation, 107 combinations of, 135-136 Cartesian product, joins, 219-221 columns, 16, 40-41 division, 135 case sensitivity, 74 adding to tables, 44-45 multiplication, 134 SELECT. 106-107 auto-incrementing columns, adding to tables, 45 subtraction, 134 CEILING, 183 limited columns, inserting ASCII. 182-183 character functions, 170 data into, 75-76 asterisks, 101 ASCII. 182-183 modifying, 46 AUTHORIZATION keyword, 292 COALESCE, 181 representing column names auto-incrementing columns, combining, 186-187 with numbers, 158-159 adding to tables, 45 CONCAT. 170-172 updating AVG, 146-147 DECODE, 178-179 multiple columns in one or avoiding IFNULL, 180 more records, 81-82 HAVING clause, 277 INSTR. 176-177 value of a single column, indexes, 261-263 LENGTH, 179-180 80-81 large sort operations, LOWER, 174-175 combining character functions, 277-278 LPAD. 181-182 186-187 OR operator, performance, LTRIM, 177-178 comma separated arguments, 276-277 100 REPLACE, 173 RPAD, 182 COMMIT, 11, 89-91, 377 **RTRIM, 178** comparison operators, 116 R combinations of, 118-119 SUBSTR, 175-176 **TRANSLATE. 172-173** equality, 116 back-end application, SQL and UPPER, 174 less than, greater than, 118 enterprise, 356 character strings non-equality, 117 **BACKUP ANY TABLE, 302** converting numbers to, composite indexes, 259-260 BACKUP DATABASE, 301 185-186 compound queries base tables, joins, 218-219 converting to dates, 203-204 GROUP BY clause, 248-250 batch loads, disabling indexes converting to numbers, ORDER BY clause, 246-248 during, 278-279 184-185 versus single queries, BETWEEN, 120-121 check constraints, 53 239-240 **BOOLEAN** values, 29 CLI (call-level interface), 349-350 compound query operators, 240 client/server model, 5-6 EXCEPT, 245-246 **CLOSE, 340** INTERSECT, 244-245

C

closing cursors, 342-343

data

retrieving accurate data, 250 conversion functions, 183-184 cursors, 339-340 UNION, 240-243 converting character strings closing, 342-343 to numbers, 184-185 UNION ALL, 243-244 fetching data from, 341-342 converting numbers to char-CONCAT. 170-172 opening, 340-341 acter strings, 185-186 CUSTOMER TBL. 440-441. concatenation, 170 converting 444-445 conjunctive operators, 126 character strings to dates, Oracle, 447-448 AND, 126-127 203-204 OR. 127-129 character strings to numbers, **CONNECT, 8, 307** 184-185 constraints D dates to character strings, dropping, 54 202-203 integrity constraints. See data numbers to character strings. integrity constraints defined, 21 185-186 controlling deleting from tables, 82-83 correlated subqueries, 233-234 privileges, 308 COS, 183 grouping, 153-154 CREATE ROLE, 308-309 CUBE expression. cost-based optimization, perfor-DROP ROLE, 309 163-164 mance, 279-280 SET ROLE, 309 GROUP BY clause. COUNT, 109, 142-145 154-155 transactions, 88 COUNT(*), 143 GROUP BY clause versus COMMIT. 89-91 CREATE, 302 ORDER BY clause. RELEASE SAVEPOINT. CREATE ANY TABLE, 302 159-161 94-95 CREATE DATABASE, 301 HAVING clause, 164-165 ROLLBACK, 91-92 CREATE INDEX257, 377 ROLLUP expression. ROLLBACK TO SAVEPOINT. CREATE PROCEDURE, 301 161-162 93-94 CREATE ROLE, 308-309, 377 inserting SAVEPOINT, 92-93 CREATE SCHEMA, 292-293 into limited columns of SET TRANSACTION, 95 CREATE TABLE AS. 378 tables, 75-76 user access, 304 CREATE TABLE statement, 41-43, NULL values, 78-80 GRANT, 304-305 301, 378 from other tables, 76-78 GRANT OPTION, 305 CREATE TRIGGER, 301, 346-348 into tables, 74-75 groups of privileges, CREATE TYPE.378 manipulating, 10, 73 306-308 CREATE USER, 290, 302, 378 retrieving, 250 on individual columns, CREATE VIEW, 301, 316, 378 selecting, 10 306 CUBE expression, 163-164 from multiple tables, PUBLIC database, 306 CUME_DIST() OVER, 351 207-208 REVOKE, 305-306

current date, 194

data

selecting data from another NULL values, 28-29 date pictures, 199-202 user's table, 110 numeric values, 24-25 dates, converting to character updating, 80 strings, 202-203 user-defined types, 29-30 multiple columns in one or DATETIME data types, 27, 192 varying-length strings, 23 more records, 81-82 implementation-specific data database administrator, See DBA through views, 322-323 types, 193 database design information, sysvalue of a single column, **DATETIME elements. 192** tem catalogs, 332 80-81 **DB DATAWRITER, 308** database management system. used in this book, 13-15 See DBMS (database manage-DB DDLADMIN, 307 data access, simplifying with ment system) DBA (database administrator). views, 314-315 database objects, 37 21, 300, 307 data administration commands, database performance, transac-**DBMS** (database management 9.11 tional control and, 95-96 system), 1 Data Control Language. See DCL DCL (Data Control Language), 9, database security, 299-300 (Data Control Language) 10-11 database structures, DDL (Data Data Definition Language. See Definition Language), 9-10 DDL (Data Definition Language), DDL (Data Definition Language) 9-10 database tuning versus SOL Data Manipulation Language. See decimal values, 25-26 statement tuning, 268 DML (Data Manipulation database vendors, 7-8 floating-point decimals, 26-27 Language) databases **DECODE. 178-179** Data Query Language. See DQL defined, 4-5 default storage, 191 (Data Query Language) relational databases, 5 DELETE statements, 82, 379 data redundancy, logical dataweb-based database syssubqueries, 230 base design, normalization, tems. 6-7 deleting data from tables, 82-83 63-64 date conversions, 198-199 denormalization, 69-70 data types, 22 converting character strings DENSE_RANK() OVER, 351 BOOLEAN values, 29 to dates, 203-204 direct SQL versus embedded SQL, date and time data types, 27 converting dates to character 351 DATETIME data types, 192 strings, 202-203 disabling indexes during batch decimal values, 25-26 date pictures, 199-202 loads, 278-279 domains, 30 date functions, 193 DISCONNECT, 8-9 fixed-length strings, 23 adding time to dates, DISTINCT floating-point decimals, 26-27 196-197 aggregate functions, 149 integers, 26 current date, 194 SELECT. 102 large object types, 24 miscellaneous, 197-198 division, arithmetic operators, limitations on, 43 time zones, 194-195 135 literal strings, 28

functions

DML (Data Manipulation enterprises, SQL and, 355 arranging tables in FROM Language), 9-10 clause, 271 back-end applications, 356 domains, data types, 30 ordering join conditions, front-end applications, 271-272 DQL (Data Query Language), 356-357 for readability, 269-271 9-10 equality, 116 DROP. 302 restrictive conditions. equality of joins, 208-210 272-273 DROP INDEX, 379 EXCEPT, 245-246 FROM clause, 381 DROP ROLE, 309 EXEC SQL, 349 arranging tables, 271 DROP TABLE, 302, 379 EXECUTE. 301 SELECT, 102 DROP TRIGGER, 348 EXISTS, 123-124 front-end applications, SQL and DROP USER, 302, 379 EXIT, 8-9 enterprise, 356-357 DROP VIEW, 323, 379 EXP (exponential values), 183 full table scans, 257, 274-275 dropping **EXPLAIN PLAN, 280** functions constraints, 54 extensions aggregate functions. See indexes, 263 implementations, 369-370 aggregate functions schemas, 293-294 MySQL, 372-373 ANSI character functions. synonyms, 325 PL/SQL, 371-372 169-170 tables, 48-49 SQL extensions, 370 character functions, 170 triggers, 348 Transact-SQL, 371 ASCII, 182-183 views, 323 COALESCE, 181 dynamic SQL, 348-349 CONCAT, 170-172 F DECODE, 178-179 IFNULL, 180 E FETCH, 340 INSTR, 176-177 fetching data from cursors, LENGTH, 179-180 embedded SQL versus direct SQL, 341-342 LOWER. 174-175 351 fields. 15 LPAD, 181-182 embedded subqueries, 231-233 first normal form, 64-65 LTRIM, 177-178 EMPLOYEE_PAY_TBL, 439, 441, fixed-length strings, 23 REPLACE, 173 444 FLOAT, 26 RPAD, 182 Oracle, 447 floating-point decimals, 26-27 RTRIM, 178 EMPLOYEE_TBL, 439-440, 443 FL00R, 183 SUBSTR. 175-176 Oracle, 446-447 FOR EACH ROW, triggers, 348 **TRANSLATE**, 172-173 end user needs, logical database foreign key constraints, 51-52 UPPER, 174 design, normalization, 63 formatting SQL statements, conversion functions, 268-269 183-184

functions

group functions, GROUP BY

CUBE expression,

163-164

clause, 155

data, 153-154

grouping

date functions, 193 GROUP BY clause, considering, 260-261 154-155 adding time to dates. creating groups, 258-257 196-197 GROUP BY clause versus creating with CREATE INDEX, ORDER BY clause. current date, 194 257 159-161 miscellaneous, 197-198 disabling indexes during HAVING clause, 164-165 batch loads, 278-279 time zones, 194-195 ROLLUP expression, dropping, 263 mathematical functions, 183 161-162 how they work, 256-257 stored procedures and, aueries, 128 343-346 implicit indexes, 260 selected data, GROUP BY TRANSLATE, 170 single-column indexes, 258 clause, 155 windowed table functions, unique indexes, 258-259 groups, creating with aggregate 351-352 INSERT, 303, 379 functions, 155-158 subqueries, 228-229 groups of privileges, controlling inserting data user access, 306-308 into limited columns of G-H GUI tools, 296 tables, 75-76 NULL values, 78-80 generating SQL with SQL, 350 HAVING clause, 164-165, 381 from other tables, 76-78 GRANT, 302, 379 avoiding, 277 into tables, 74-75 controlling user access, 304-305 INSERT...SELECT. 380 **GRANT OPTION, 305** installing ı granting privileges, 303-304 Microsoft SQL Server, 388-390 GROUP BY clause, 154-155, 381 IFNULL, 180 MySQL, 383-385 compound queries, 248-250 implementations, 367 Oracle, 386-387 creating groups with aggrecompliance with ANSI SQL, INSTR, 176-177 gate functions, 155-158 369 group functions, 155 integers, 26 differences between, 367-369 grouping selected data, 155 integrity constraints, 49 extensions to SQL, 369-370 versus ORDER BY, 159-161 check constraints, 53 implementation-specific data foreign key constraints, 51-52 representing column names types, 193 with numbers, 158-159 NOT NULL constraints, 52 implicit indexes, 260

IN, 121-122

indexes, 255-256

altering, 263

avoiding, 261-263

composite indexes, 259-260

primary key constraints,

unique constraints, 50-51

interactive SQL statements,

49-50

373-374

International Standards Organization (ISO), 2

Internet, SQL and, 360

making data available to customers worldwide, 360-361 making data available to

employees and privileged customers, 361

INTERSECT, 244-245 intranets, SQL and, 361-362 IS NOT NULL, 132 IS NULL, 120 ISO (International Standards

Organization), 2

J-K

JDBC (Java Database Connectivity), 358 join conditions, ordering, 271-272 joins, 208

base tables, 218-219
Cartesian product, 219-221
components of a join condition, 208
equality of, 208-210
multiple keys, 216-217
non-equality, 211-212
outer joins, 212-215
self joins, 215-216

keywords

AUTHORIZATION, 292 SELECT, 100

table aliases, 210

L

LENGTH, 179-180
less than, greater than, comparison operators, 118
LIKE, 122-123, 275-276
limitations on data types, 43
literal strings, 28
logical database design, normalization, 62-63
data redundancy, 63-64

large object types, 24

end user needs, 63 logical operators, 119-120

ALL, 124-126

ANY, 124-126 BETWEEN, 120-121

EXISTS, 123-124

IN, 121-122

IS NULL, 120

LIKE, 122-123

SOME, 124-126

LOWER, 174-175

LPAD, 181-182

LTRIM, 177-178

M

major implementation system catalog objects, 333-334 managing users, 285-287 manipulating data, 10, 73 mathematical functions, 183 MAX, 147-148

Microsoft SQL Server

creating users, 290-291 cursors, 340

closing, 343

CUSTOMER_TBL, 441,

444-445

EMPLOYEE_PAY_TBL, 441,

444

EMPLOYEE_TBL, 440, 443

ORDERS_TBL, 441, 445

parameters, 374

PRODUCTS_TBL, 442, 446

SELECT, 368

stored procedures, 344-345

triggers, creating, 346

Windows installation instruc-

tions, 388-390

MIN, 148-149

modifying

columns in tables, 46

elements of tables, 44

multiple keys, joining, 216-217

multiplication, arithmetic opera-

tors, 134

MySQL

creating users, 291

cursors, 340

closing, 343

CUSTOMER_TBL, 440,

444-445

EMPLOYEE_PAY_TBL, 439,

444

EMPLOYEE_TBL, 439, 443

extensions, 372-373

ORDERS TBL, 440, 445

PRODUCTS_TBL, 440, 446

MySQL

combinations, 135-136 stored procedures, 344 naming conventions, 67 triggers, creating, 346 normal forms, 64 division, 135 Windows installation instrucfirst normal form, 64-65 multiplication, 134 tions, 383-385 second normal form. subtraction, 134 65-66 comparison operators, 116 third normal form, 67 combinations of, 118-119 raw databases, 62 N equality, 116 **NOT BETWEEN, 130-131** less than, greater than, NOT EOUAL, 130 118 naming conventions NOT EXISTS, 133 non-equality, 117 normalization, 67 **NOT IN, 131** conjunctive operators, 126 tables, 43 NOT LIKE, 131-132 AND. 126-127 naming objects, 39 NOT NULL constraints, 52 OR, 127-129 negative operators, 129 NULL value checker, 180 logical operators, 119-120 IS NOT NULL, 132 NULL values, 16, 28-29 ALL, 124-126 NOT BETWEEN, 130-131 inserting in tables, 78-80 ANY. 124-126 NOT EQUAL, 130 numbers BETWEEN, 120-121 NOT EXISTS, 133 converting character strings EXISTS, 123-124 **NOT IN. 131** to. 184-185 IN. 121-122 NOT LIKE, 131-132 converting to character IS NULL, 120 nested views, performance. strings, 185-186 323-324 LIKE, 122-123 numeric values, 24-25 SOME. 124-126 non-equality comparison operators, 117 negative operators, 129 joins, 211-212 IS NOT NULL, 132 O NOT BETWEEN, 130-131 normal forms, 64 first normal form, 64-65 NOT EQUAL, 130 object privileges, 302-303 second normal form, 65-66 NOT EXISTS, 133 objects, naming, 39 third normal form, 67 **NOT IN. 131** ODBC (Open Database normalization, 61-62 NOT LIKE, 131-132 Connectivity), 358 benefits of, 68-69 OR. 127-129 OLE DB, 359 drawbacks of, 69 avoiding, 276-277 **OPEN, 340** logical database design, Oracle opening cursors, 340-341 62-63 creating users, 289-290 operators, 115 data redundancy, 63-64 cursors, 340 arithmetic operators, 133 end user needs, 63 closing, 343

addition, 133-134

querying system catalogs

CUSTOMER TBL,441, large sort operations, PRODUCTS TBL, 440, 442, 446 447-448 277-278 Oracle, 449 EMPLOYEE_PAY_TBL, 441, OR operator, 276-277 PUBLIC database, 307 447 cost-based optimization. EMPLOYEE TBL, 440, 279-280 446-447 disabling indexes during ORDERS TBL, 441, 448-449 batch loads, 278-279 parameters, 373 full table scans, 274-275 queries, 99 PRODUCTS TBL, 442, 449 LIKE operator, 275-276 compound queries SELECT. 368-369 nested views, 323-324 GROUP BY clause. stored procedures, 344-345 SQL statement tuning, 267 248-250 triggers, creating, 346 versus database tuning, ORDER BY clause, 268 Windows installation instruc-246-248 tions, 386-387 stored procedures, 278 compound query operators, Oracle Fusion Middleware, 359 subqueries, 234-235 240 ORDER BY clause, 382 wildcards, 275-276 EXCEPT. 245-246 compound queries, 246-248 performance statistics, system INTERSECT, 244-245 catalogs, 332-333 versus GROUP BY clause. UNION, 240-243 159-161 performance tools, 280 UNION ALL. 243-244 SELECT. 104-106 **PL/SQL, 370** grouping, 128 views, 322 extensions, 371-372 simple queries ordering join conditions, 271-272 populating tables with new data, column aliases, 111 74 ORDERS TBL, 440-441, 445 counting records in tables, **POWER. 183** Oracle, 448-449 109-110 primary key constraints, 49-50 outer joins, 212-215 examples, 108-109 primary keys, 16 selecting data from anothprivileges, 301 er user's table, 110 controlling, 308 single versus compound, P CREATE ROLE, 308-309 239-240 DROP ROLE, 309 querying system catalogs, parameters SET ROLE, 309 334-336 Microsoft SQL Server, 374 granting, 303-304 Oracle, 373

object privileges, 302-303

system privileges, 301-302

revoking, 303-304

PERCENT_RANK() OVER, 351

HAVING clause, 277

performance

avoiding

RANK() OVER

R	retrieving data, 250	on individual columns, 306
	REVOKE, 380	333
RANK() OVER, 351	controlling user access,	PUBLIC database, 306
raw databases, normalization, 62	305-306	REVOKE, 305-306
RDBMS (relational database management system), 2	revoking privileges, 303-304	database security, 299-300
	ROLLBACK, 11, 91-92, 380	Internet, 361
READ WRITE, 95	ROLLBACK TO SAVEPOINT, 93-94	privileges, 301
readability, formatting SQL state-	ROLLUP expression, 161-162	object privileges, 302-303
ments, 269-271	ROUND, 183	system privileges,
records, 15-16	ROW_NUMBER() OVER, 351	301-302
counting records in tables,	rows, 41	views, 315
simple queries, 109-110	rows of data, 15-16	security information, system cata
REFERENCES, 303	RPAD, 182	logs, 332
relational database management system. See RDBMS (relational	RTRIM, 178	SELECT, 10, 73, 99-102, 303, 380-381
database management system)		ANSI standard, 368
relational databases, 5		case sensitivity, 106-107
RELEASE SAVEPOINT, 94-95	5	FROM clause, 102
RELOAD, 302		creating groups, 155-158
remote databases, accessing,	SAVEPOINT, 11, 92-93, 380	DISTINCT, 102
357-358	schemas, 37-39	Microsoft SQL Server, 368
JDBC, 358	creating, 292-293	Oracle, 368-369
ODBC (Open Database	dropping, 293-294	ORDER BY clause, 104-106
Connectivity), 358	versus users, 288	subqueries, 227-228
OLE DB, 359	second normal form, 65-66	WHERE clause, 103-104
through web interfaces,	security	SELECT ANY TABLE, 302
359-360	controlling privileges, 308	selecting
vendor connectivity products,	CREATE ROLE, 308-309	data, 10
359	DROP ROLE, 309	from multiple tables,
removing user access, 295-296	SET ROLE, 309	207-208
REPLACE, 173	controlling user access, 304	data from another table, 110
representing column names with	GRANT, 304-305	self joins, 215-216
numbers, 158-159	GRANT OPTION, 305	SET ROLE, 309
RESOURCE, 307	groups of privileges,	SET TRANSACTION, 11, 95
restrictive conditions, SQL state- ments, 272-273	306-308	SHUTDOWN, 302
		SIGN (sign values), 183
		(- 5

system catalogs

simple queries DCL (Data Control Language), strings 10-11 column aliases, 111 fixed-length strings, 23 DDL (Data Definition counting records in tables, literal strings, 28 Language), 9-10 109-110 varying-length strings, 23 DML (Data Manipulation examples, 108-109 Structured Ouery Language, See Language), 10 selecting data from another SOL (Structured Ouerv DQL (Data Query Language), user's table, 110 Language) simplifying data access with subqueries, 225-227 transaction control comviews, 314-315 correlated, 233-234 mands. 9. 11 SIN. 183 DFI FTF statements, 230 SQL extensions, 370 single queries versus compound embedded, 231-233 SQL sessions, 8 queries, 239-240 INSERT statements, 228-229 CONNECT. 8 single quotation marks, 74 performance, 234-235 DISCONNECT, 8-9 single-column indexes, 258 SELECT statements, 227-228 EXIT, 8-9 SOME, 124-126 UPDATE statements, 229-230 SQL statement tuning, 267 sort operations, avoiding, SUBSTR. 175-176 277-278 versus database tuning, 268 substrings, 170 SQL (Structured Query **SQL** statements subtraction, arithmetic operators, Language), 2 formatting, 268-269 134 direct versus embedded, 351 arranging tables in FROM SUM. 145-146 enterprises and, 355 clause, 271 summarized data, maintaining ordering join conditions, back-end applications, with views, 315-316 356 271-272 synonyms, 324 front-end applications, for readability, 269-271 creating, 324-325 356-357 restrictive conditions. dropping, 325 272-273 generating SQL, 350 simple queries, 111 Internet and, 360 interactive SQL statements, SYS. 331 373-374 making data available to system catalog objects, updating, customers worldwide. SQL-2008, 3-4 336 360-361 SQRT (square root), 183 system catalogs, 327-330 making data available to standards, table-naming stancontents of, 331-332 employees and prividards, 12-13 leged customers, 361 creating, 331 storage, default storage, 191 intranets and, 361-362 database design information, stored procedures 332 SQL commands, 9 functions and, 343-346 performance statistics, data administration comperformance, 278 332-333 mands, 9, 11

system catalogs

naming conventions, 43

NULL values, 16 querying, 334-336 U security information, 332 populating with new data, 74 tables by implementation, primary keys, 16 UNION, 240-243 333-334 records, 15-16 UNION ALL, 243-244 user data, 332 rows, 41 unique constraints, 50-51 system privileges, 301-302 TAN. 183 unique indexes, 258-259 TEXT data type, 24 UPDATE statements, 303, 380 third normal form, 67 subqueries, 229-230 TIME, 192 updating time zones, date functions, data, 80 194-195 table aliases, 210 multiple columns in one or TIMESTAMP, 192 more records, 81-82 table-naming standards, 12-13 TKPROF, 280 through views, 322-323 tables, 15, 39 tools, database users, 296 value of a single column, adding 80-81 transaction control commands, auto-incrementing 9. 11 system catalog objects, 336 columns, 45 transactional control, database **UPPER. 174** columns, 44-45 performance and, 95-96 USAGE, 303 ALTER TABLE, 44 transactions, 87-88 user access, controlling, 304 arranging in FROM clause, controlling, 88 GRANT, 304-305 271 COMMIT, 89-91 columns, 16, 40-41 GRANT OPTION, 305 RELEASE SAVEPOINT, modifying, 46 groups of privileges, 306-308 94-95 CREATE TABLE statement. on individual columns, 306 ROLLBACK, 91-92 41-43 PUBLIC database, 306 ROLLBACK TO SAVEPOINT, creating from existing tables, REVOKE, 305-306 93-94 46-48 user data, system catalogs, 332 SAVEPOINT, 92-93 creating from views, 321-322 user-defined types, 29-30 SET TRANSACTION, 95 data user management, 288 Transact-SQL, extensions, 371 deleting, 82-83 creating users, 289 **TRANSLATE, 172-173** inserting, 74-75 in Microsoft SQL Server, TRANSLATE function, 170 dropping, 48-49 290-291 triggers, 346 fields, 15 in MySQL, 291 creating, 346-348 inserting data from another in Oracle, 289-290 table, 76-78 dropping, 348 user sessions, 295 modifying elements of, 44 FOR EACH ROW, 348

simplifying data access. users 314-315 altering, 294-295 updating data, 322-323 creating, 289 WITH CHECK OPTION, 320 in Microsoft SQL Server, 290-291 in MySQL, 291 creating groups in Oracle, W-Z289-290 managing, 285-287 web-based database systems, 6-7 place in databases, 287 web interfaces, accessing remote removing access, 295-296 databases, 359-360 versus schemas, 288 WHERE, 82, 381 types of, 286 SELECT, 103-104 wildcards, performance, 275-276 windowed table functions, 351-352 Windows installation instructions Microsoft SQL Server, varying-length strings, 23 388-390 vendor connectivity products, 359 for MySQL, 383-385 vendors, database vendors, 7-8 for Oracle, 386-387 views, 313-314 WITH CHECK OPTION, views, 320 creating, 316 from multiple tables, XML, 352 318-319 from a single table, 316-318 from views, 319-320 creating tables from, 321-322 dropping, 323 as a form of security, 315 maintaining summarized data, 315-316 nested views, performance, 323-324

ORDER BY clause, 322