Contents at a Glance

Part I: An SQL Concepts Overview

HOUR 1 Welcome to the World of SQL ....................................................... 1

Part II: Building 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: Getting 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: Building 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: SQL Performance Tuning

HOUR 16 Using Indexes to Improve Performance .................................. 255
17 Improving Database Performance ............................................. 267
Part VI: Using SQL to Manage Users and Security

**HOUR 18** Managing Database Users ........................................................ 285

**19** Managing Database Security ............................................................. 299

Part VII: Summarized Data Structures

**HOUR 20** Creating and Using Views and Synonyms .................................... 313

**21** Working with the System Catalog ...................................................... 329

Part VIII: 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: Appendixes

**A** Common SQL Commands ................................................................. 377

**B** 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

Index ............................................................................................................. 461
# Table of Contents

## Part I: An SQL Concepts Overview

**HOUR 1: Welcome to the World of SQL**

<table>
<thead>
<tr>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL Definition and History</td>
<td>1</td>
</tr>
<tr>
<td>SQL Sessions</td>
<td>8</td>
</tr>
<tr>
<td>Types of SQL Commands</td>
<td>9</td>
</tr>
<tr>
<td>The Database Used in This Book</td>
<td>12</td>
</tr>
<tr>
<td>Summary</td>
<td>17</td>
</tr>
<tr>
<td>Q&amp;A</td>
<td>17</td>
</tr>
<tr>
<td>Workshop</td>
<td>18</td>
</tr>
</tbody>
</table>

## Part II: Building Your Database

**HOUR 2: Defining Data Structures**

<table>
<thead>
<tr>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>What Is Data?</td>
<td>21</td>
</tr>
<tr>
<td>Basic Data Types</td>
<td>22</td>
</tr>
<tr>
<td>Summary</td>
<td>31</td>
</tr>
<tr>
<td>Q&amp;A</td>
<td>31</td>
</tr>
<tr>
<td>Workshop</td>
<td>32</td>
</tr>
</tbody>
</table>

**HOUR 3: Managing Database Objects**

<table>
<thead>
<tr>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>What Are Database Objects?</td>
<td>37</td>
</tr>
<tr>
<td>What Is a Schema?</td>
<td>37</td>
</tr>
<tr>
<td>Tables: The Primary Storage for Data</td>
<td>39</td>
</tr>
<tr>
<td>Integrity Constraints</td>
<td>49</td>
</tr>
<tr>
<td>Summary</td>
<td>54</td>
</tr>
<tr>
<td>Q&amp;A</td>
<td>55</td>
</tr>
<tr>
<td>Workshop</td>
<td>55</td>
</tr>
</tbody>
</table>
HOUR 4: The Normalization Process

Normalizing a Database ................................................................. 61
Denormalizing a Database ............................................................. 69
Summary ....................................................................................... 70
Q&A .............................................................................................. 70
Workshop ...................................................................................... 71

HOUR 5: Manipulating Data

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

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

What Is a Query? ........................................................................... 99
Introduction to the SELECT Statement .............................................. 99
Examples of Simple Queries ............................................................. 108
Summary ...................................................................................... 112
Q&A .............................................................................................. 112
Workshop ...................................................................................... 113
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Combining Character Functions</td>
<td>186</td>
</tr>
<tr>
<td>Summary</td>
<td>187</td>
</tr>
<tr>
<td>Q&amp;A</td>
<td>188</td>
</tr>
<tr>
<td>Workshop</td>
<td>188</td>
</tr>
<tr>
<td><strong>HOUR 12: Understanding Dates and Times</strong></td>
<td>191</td>
</tr>
<tr>
<td>How Is a Date Stored?</td>
<td>191</td>
</tr>
<tr>
<td>Date Functions</td>
<td>193</td>
</tr>
<tr>
<td>Date Conversions</td>
<td>198</td>
</tr>
<tr>
<td>Summary</td>
<td>204</td>
</tr>
<tr>
<td>Q&amp;A</td>
<td>204</td>
</tr>
<tr>
<td>Workshop</td>
<td>205</td>
</tr>
<tr>
<td><strong>Part IV: Building Sophisticated Database Queries</strong></td>
<td></td>
</tr>
<tr>
<td><strong>HOUR 13: Joining Tables in Queries</strong></td>
<td>207</td>
</tr>
<tr>
<td>Selecting Data from Multiple Tables</td>
<td>207</td>
</tr>
<tr>
<td>Understanding Joins</td>
<td>208</td>
</tr>
<tr>
<td>Join Considerations</td>
<td>217</td>
</tr>
<tr>
<td>Summary</td>
<td>221</td>
</tr>
<tr>
<td>Q&amp;A</td>
<td>222</td>
</tr>
<tr>
<td>Workshop</td>
<td>222</td>
</tr>
<tr>
<td><strong>HOUR 14: Using Subqueries to Define Unknown Data</strong></td>
<td>225</td>
</tr>
<tr>
<td>What Is a Subquery?</td>
<td>225</td>
</tr>
<tr>
<td>Embedded Subqueries</td>
<td>231</td>
</tr>
<tr>
<td>Correlated Subqueries</td>
<td>233</td>
</tr>
<tr>
<td>Subquery Performance</td>
<td>234</td>
</tr>
<tr>
<td>Summary</td>
<td>235</td>
</tr>
<tr>
<td>Q&amp;A</td>
<td>235</td>
</tr>
<tr>
<td>Workshop</td>
<td>236</td>
</tr>
</tbody>
</table>
### HOUR 15: Combining Multiple Queries into One

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single Queries Versus Compound Queries</td>
<td>239</td>
</tr>
<tr>
<td>Compound Query Operators</td>
<td>240</td>
</tr>
<tr>
<td>Using ORDER BY with a Compound Query</td>
<td>246</td>
</tr>
<tr>
<td>Using GROUP BY with a Compound Query</td>
<td>248</td>
</tr>
<tr>
<td>Retrieving Accurate Data</td>
<td>250</td>
</tr>
<tr>
<td>Summary</td>
<td>250</td>
</tr>
<tr>
<td>Q&amp;A</td>
<td>250</td>
</tr>
<tr>
<td>Workshop</td>
<td>251</td>
</tr>
</tbody>
</table>

### Part V: SQL Performance Tuning

### HOUR 16: Using Indexes to Improve Performance

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>What Is an Index?</td>
<td>255</td>
</tr>
<tr>
<td>How Do Indexes Work?</td>
<td>256</td>
</tr>
<tr>
<td>The CREATE INDEX Command</td>
<td>257</td>
</tr>
<tr>
<td>Types of Indexes</td>
<td>258</td>
</tr>
<tr>
<td>When Should Indexes Be Considered?</td>
<td>260</td>
</tr>
<tr>
<td>When Should Indexes Be Avoided?</td>
<td>261</td>
</tr>
<tr>
<td>Altering an Index</td>
<td>263</td>
</tr>
<tr>
<td>Dropping an Index</td>
<td>263</td>
</tr>
<tr>
<td>Summary</td>
<td>264</td>
</tr>
<tr>
<td>Q&amp;A</td>
<td>264</td>
</tr>
<tr>
<td>Workshop</td>
<td>265</td>
</tr>
</tbody>
</table>

### HOUR 17: Improving Database Performance

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>What Is SQL Statement Tuning?</td>
<td>267</td>
</tr>
<tr>
<td>Database Tuning Versus SQL Statement Tuning</td>
<td>268</td>
</tr>
<tr>
<td>Formatting Your SQL Statement</td>
<td>268</td>
</tr>
<tr>
<td>Full Table Scans</td>
<td>274</td>
</tr>
<tr>
<td>Other Performance Considerations</td>
<td>275</td>
</tr>
<tr>
<td>Cost-Based Optimization</td>
<td>279</td>
</tr>
<tr>
<td>Performance Tools</td>
<td>280</td>
</tr>
<tr>
<td>Summary</td>
<td>280</td>
</tr>
</tbody>
</table>
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
### HOUR 21: Working with the System Catalog

- 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

### Part VIII: Applying SQL Fundamentals in Today’s World

#### HOUR 22: Advanced SQL Topics

- Cursors 339
- Stored Procedures and Functions 343
- Triggers 346
- Dynamic SQL 348
- Call-Level Interface 349
- Using SQL to Generate SQL 350
- Direct Versus Embedded SQL 351
- Windowed Table Functions 351
- Working with XML 352

#### HOUR 23: Extending SQL to the Enterprise, the Internet, and the Intranet

- SQL and the Enterprise 355
- Accessing a Remote Database 357
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.

—Poppy

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.
This page intentionally left blank
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
Tables: The Primary Storage for Data

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.

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 `CUSTOMERNAMES`. 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
Tables: The Primary Storage for Data

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.](image)

**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:

```sql
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 `EMPLOYEE_TBL` in the following example using the syntax for MySQL:

```sql
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:

---

By the Way

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?
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.

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:

```sql
CREATE TABLE TEST_INCREMENT(
    ID SERIAL,
    TEST_NAME VARCHAR(20));
```

**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.

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

```sql
CREATE TABLE TEST_INCREMENT(
    ID INT IDENTITY(1,1) NOT NULL,
    TEST_NAME VARCHAR(20));
```

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:

```sql
INSERT INTO TEST_INCREMENT(TEST_NAME)
VALUES ('FRED'), ('JOE'), ('MIKE'), ('TED');
```

```sql
SELECT * FROM TEST_INCREMENT;
```

<table>
<thead>
<tr>
<th>ID</th>
<th>TEST_NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>FRED</td>
</tr>
<tr>
<td>2</td>
<td>JOE</td>
</tr>
<tr>
<td>3</td>
<td>MIKE</td>
</tr>
<tr>
<td>4</td>
<td>TED</td>
</tr>
</tbody>
</table>
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.

By the Way

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

Watch Out!

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 cre-
ating 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;
```

<table>
<thead>
<tr>
<th>PROD_ID</th>
<th>PROD_DESC</th>
<th>COST</th>
</tr>
</thead>
<tbody>
<tr>
<td>11235</td>
<td>WITCH COSTUME</td>
<td>29.99</td>
</tr>
<tr>
<td>222</td>
<td>PLASTIC PUMPKIN 18 INCH</td>
<td>7.75</td>
</tr>
<tr>
<td>13</td>
<td>FALSE PARAFFIN TEETH</td>
<td>1.1</td>
</tr>
<tr>
<td>90</td>
<td>LIGHTED LANTERNES</td>
<td>14.5</td>
</tr>
<tr>
<td>15</td>
<td>ASSORTED COSTUMES</td>
<td>10</td>
</tr>
<tr>
<td>9</td>
<td>CANDY CORN</td>
<td>1.35</td>
</tr>
<tr>
<td>6</td>
<td>PUMPKIN CANDY</td>
<td>1.45</td>
</tr>
<tr>
<td>87</td>
<td>PLASTIC SPIDERS</td>
<td>1.05</td>
</tr>
<tr>
<td>119</td>
<td>ASSORTED MASKS</td>
<td>4.95</td>
</tr>
</tbody>
</table>

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:

```sql
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.

```sql
select *
from products_tmp;
```

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

**Did You Know?**

**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:

```sql
drop table table_name [ restrict | cascade ]
```
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,
...)
```
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.
Integrity Constraints

Study the following example:

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

```sql
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,
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.
HOUR 3: Managing Database Objects

NULL and NOT NULL

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.

By the Way

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:

```sql
ALTER TABLE employee_pay_tbl
ADD CONSTRAINT id_fk FOREIGN KEY (emp_id)
REFERENCES employee_tbl (emp_id);
```

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,
EMP_CITY         VARCHAR2(15) 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,
DATE_LAST_RAISE    DATE   NULL,
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.

Did You Know?

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)?

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

```sql
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:

```sql
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:

```sql
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:

```sql
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.

```bash
SQLCMD -S localhost -U username -Ppassword
```
At the `>` 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>GO
```

Now go to Appendix D to get the DDL for the tables used in this book. At the `>` 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 `>` 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 `>` 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;
This page intentionally left blank
ABS (absolute value), 183
accessing remote databases, 357-358
  JDBC, 358
  ODBC (Open Database Connectivity), 358
  OLE DB, 359
  through web interfaces, 359-360
  vendor connectivity products, 359
adding
  auto-incrementing columns to tables, 45
  columns to tables, 44-45
  time to dates, 196-197
addition, arithmetic operators, 133-134
ADMIN OPTION, 305
aggregate functions, 141-142
  AVG, 146-147
  COUNT, 142-145
  creating groups, 155-158
  MAX, 147-148
  MIN, 148-149
  SUM, 145-146
ALL, 124-126
ALTER ANY TABLE, 302
ALTER DATABASE, 302
ALTER TABLE, 44, 264, 377
ALTER USER, 302
altering
  indexes, 263
  users, 294-295
American National Standards Institute. See ANSI (American National Standards Institute)
AND, 126-127
ANSI (American National Standards Institute), 2
ANSI character functions, 169-170
ANSI SQL, 2
  compliance with, 369
ANSI standard, SELECT, 368
ANY, 124-126
arithmetic operators, 133
addition, 133-134
combinations of, 135-136
division, 135
multiplication, 134
subtraction, 134
ASCII, 182-183
asterisks, 101
AUTHORIZATION keyword, 292
auto-incrementing columns, adding to tables, 45
AVG, 146-147
avoiding
HAVING clause, 277
indexes, 261-263
large sort operations, 277-278
OR operator, performance, 276-277

B
back-end application, SQL and enterprise, 356
BACKUP ANY TABLE, 302
BACKUP DATABASE, 301
base tables, joins, 218-219
batch loads, disabling indexes during, 278-279
BETWEEN, 120-121
BOOLEAN values, 29

C
call-level interface (CLI), 349-350
Cartesian product, joins, 219-221
case sensitivity, 74
SELECT, 106-107
CEILING, 183
character functions, 170
ASCII, 182-183
COALESCE, 181
combining, 186-187
CONCAT, 170-172
DECODE, 178-179
IFNULL, 180
INSTR, 176-177
LENGTH, 179-180
LOWER, 174-175
LPAD, 181-182
LTRIM, 177-178
REPLACE, 173
RPAD, 182
RTRIM, 178
SUBSTR, 175-176
TRANSLATE, 172-173
UPPER, 174
character strings
converting numbers to, 185-186
converting to dates, 203-204
converting to numbers, 184-185
check constraints, 53
CLI (call-level interface), 349-350
client/server model, 5-6
CLOSE, 340
closing cursors, 342-343
COALESCE, 181
collation, 107
columns, 16, 40-41
adding to tables, 44-45
auto-incrementing columns, adding to tables, 45
limited columns, inserting data into, 75-76
modifying, 46
representing column names with numbers, 158-159
updating multiple columns in one or more records, 81-82
value of a single column, 80-81
combining character functions, 186-187
comma separated arguments, 100
COMMIT, 11, 89-91, 377
comparison operators, 116
combinations of, 118-119
equality, 116
less than, greater than, 118
non-equality, 117
composite indexes, 259-260
compound queries
GROUP BY clause, 248-250
ORDER BY clause, 246-248
versus single queries, 239-240
compound query operators, 240
EXCEPT, 245-246
INTERSECT, 244-245
data

conversion functions, 183-184
converting character strings to numbers, 184-185
converting numbers to character strings, 185-186
converting
character strings to dates, 203-204
calendar strings to numbers, 184-185
dates to character strings, 202-203
numbers to character strings, 185-186
correlated subqueries, 233-234
COS, 183
cost-based optimization, performance, 279-280
COUNT, 109, 142-145
COUNT(*), 143
CREATE, 302
CREATE ANY TABLE, 302
CREATE DATABASE, 301
CREATE INDEX, 257, 377
CREATE PROCEDURE, 301
CREATE ROLE, 308-309, 377
CREATE SCHEMA, 292-293
CREATE TABLE AS, 378
CREATE TABLE statement, 41-43, 301, 378
CREATE TRIGGER, 301, 346-348
CREATE TYPE, 378
CREATE USER, 290, 302, 378
CREATE VIEW, 301, 316, 378
CUBE expression, 163-164
CUME_DIST() OVER, 351
current date, 194
cursors, 339-340
closing, 342-343
fetching data from, 341-342
opening, 340-341
CUSTOMER_TBL, 440-441, 444-445
Oracle, 447-448
data
defined, 21
deleting from tables, 82-83
grouping, 153-154
CUBE expression, 163-164
GROUP BY clause, 154-155
GROUP BY clause versus ORDER BY clause, 159-161
HAVING clause, 164-165
ROLLUP expression, 161-162
inserting
into limited columns of tables, 75-76
NULL values, 78-80
from other tables, 76-78
into tables, 74-75
manipulating, 10, 73
retrieving, 250
selecting, 10
from multiple tables, 207-208

How can we make this index more useful? Email us at indexes@samspublishing.com
data

selecting data from another user's table, 110
updating, 80
  multiple columns in one or more records, 81-82
through views, 322-323
value of a single column, 80-81
used in this book, 13-15
data access, simplifying with views, 314-315
data administration commands, 9, 11
Data Control Language. See DCL (Data Control Language)
Data Definition Language. See DDL (Data Definition Language)
Data Manipulation Language. See DML (Data Manipulation Language)
Data Query Language. See DQL (Data Query Language)
data redundancy, logical database design, normalization, 63-64
data types, 22
  BOOLEAN values, 29
date and time data types, 27
  DATETIME data types, 192
decimal values, 25-26
domains, 30
fixed-length strings, 23
floating-point decimals, 26-27
integers, 26
large object types, 24
limitations on, 43
literal strings, 28
NULL values, 28-29
numeric values, 24-25
user-defined types, 29-30
varying-length strings, 23
database administrator. See DBA
database design information, system catalogs, 332
database management system. See DBMS (database management system)
database objects, 37
database performance, transactional control and, 95-96
database security, 299-300
database structures, DDL (Data Definition Language), 9-10
database tuning versus SQL statement tuning, 268
database vendors, 7-8
databases
defined, 4-5
  relational databases, 5
  web-based database systems, 6-7
date conversions, 198-199
  converting character strings to dates, 203-204
  converting dates to character strings, 202-203
date pictures, 199-202
date functions, 193
  adding time to dates, 196-197
dates, converting to character strings, 202-203
DATETIME data types, 27, 192
  implementation-specific data types, 193
DATETIME elements, 192
DB_DATAWRITER, 308
DB_DDLADMIN, 307
DBA (database administrator), 21, 300, 307
DBMS (database management system), 1
DCL (Data Control Language), 9, 10-11
DDL (Data Definition Language), 9-10
decimal values, 25-26
  floating-point decimals, 26-27
DECODE, 178-179
default storage, 191
DELETE statements, 82, 379
  subqueries, 230
deleting data from tables, 82-83
denormalization, 69-70
DENSE_RANK() OVER, 351
direct SQL versus embedded SQL, 351
disabling indexes during batch loads, 278-279
DISCONNECT, 8-9
DISTINCT
  aggregate functions, 149
  SELECT, 102
division, arithmetic operators, 135
DML (Data Manipulation Language), 9-10
domains, data types, 30
DQL (Data Query Language), 9-10
DROP 302
DROP INDEX, 379
DROP ROLE, 309
DROP TABLE, 302, 379
DROP TRIGGER, 348
DROP USER, 302, 379
DROP VIEW, 323, 379
dropping
constraints, 54
indexes, 263
schemas, 293-294
synonyms, 325
tables, 48-49
triggers, 348
views, 323
dynamic SQL, 348-349
enterprises, SQL and, 355
back-end applications, 356
front-end applications, 356-357
equality, 116
equality of joins, 208-210
EXCEPT, 245-246
EXEC SQL, 349
EXECUTE, 301
EXISTS, 123-124
EXIT, 8-9
EXP (exponential values), 183
EXPLAIN PLAN, 280
extensions
implementations, 369-370
MySQL, 372-373
PL/SQL, 371-372
SQL extensions, 370
Transact-SQL, 371
FROM clause, 381
arranging tables, 271
SELECT, 102
front-end applications, SQL and enterprise, 356-357
full table scans, 257, 274-275
functions
aggregate functions. See aggregate functions
ANSI character functions, 169-170
character functions, 170
ASCII, 182-183
COALESCE, 181
CONCAT, 170-172
DECODING, 178-179
IFNULL, 180
INSTR, 176-177
LENGTH, 179-180
LOWER, 174-175
LPAD, 181-182
LTRIM, 177-178
REPLACE, 173
RPAD, 182
RTRIM, 178
SUBSTR, 175-176
TRANSLATE, 172-173
UPPER, 174
conversion functions, 183-184
functions

date functions, 193
adding time to dates, 196-197
current date, 194
miscellaneous, 197-198
time zones, 194-195
mathematical functions, 183
stored procedures and, 343-346
TRANSLATE, 170
windowed table functions, 351-352

G-H

generating SQL with SQL, 350
GRANT, 302, 379
controlling user access, 304-305
GRANT OPTION, 305
granting privileges, 303-304
GROUP BY clause, 154-155, 381
compound queries, 248-250
creating groups with aggregate functions, 155-158
group functions, 155
grouping selected data, 155
versus ORDER BY, 159-161
representing column names with numbers, 158-159

group functions, GROUP BY clause, 155

GROUP BY clause, 154-155
GROUP BY clause versus ORDER BY clause, 159-161
HAVING clause, 164-165
ROLLUP expression, 161-162
queries, 128
selected data, GROUP BY clause, 155
groups, creating with aggregate functions, 155-158

groups of privileges, controlling user access, 306-308
GUI tools, 296
HAVING clause, 164-165, 381
avoiding, 277
IFNULL, 180
implementations, 367
compliance with ANSI SQL, 369
differences between, 367-369
extensions to SQL, 369-370
implementation-specific data types, 193
implicit indexes, 260
IN, 121-122
indexes, 255-256
altering, 263
avoiding, 261-263
composite indexes, 259-260
considering, 260-261
creating groups, 258-257
creating with CREATE INDEX, 257
disabling indexes during batch loads, 278-279
dropping, 263
how they work, 256-257
implicit indexes, 260
single-column indexes, 258
unique indexes, 258-259
INSERT, 303, 379
subqueries, 228-229
inserting data
into limited columns of tables, 75-76
NULL values, 78-80
from other tables, 76-78
into tables, 74-75
INSERT...SELECT, 380

installing
Microsoft SQL Server, 388-390
MySQL, 383-385
Oracle, 386-387
INSTR, 176-177

integers, 26

integrity constraints, 49
check constraints, 53
foreign key constraints, 51-52
NOT NULL constraints, 52
primary key constraints, 49-50
unique constraints, 50-51

interactive SQL statements, 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
table aliases, 210

keywords
AUTHORIZATION, 292
SELECT, 100

L
large object types, 24
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
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

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 instructions, 388-390
MIN, 148-149
modifying
columns in tables, 46
elements of tables, 44
multiple keys, joining, 216-217
multiplication, arithmetic operators, 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

How can we make this index more useful? Email us at indexes@samspublishing.com
stored procedures, 344
triggers, creating, 346
Windows installation instructions, 383-385

naming conventions
normalization, 67
tables, 43
naming objects, 39
negative operators, 129
  IS NOT NULL, 132
  NOT BETWEEN, 130-131
  NOT EQUAL, 130
  NOT EXISTS, 133
  NOT IN, 131
  NOT LIKE, 131-132
nested views, performance, 323-324
non-equality
  comparison operators, 117
  joins, 211-212
normal forms, 64
  first normal form, 64-65
  second normal form, 65-66
  third normal form, 67
normalization, 61-62
  benefits of, 68-69
  drawbacks of, 69
  logical database design, 62-63
  data redundancy, 63-64
  end user needs, 63
naming conventions, 67
normal forms, 64
  first normal form, 64-65
  second normal form, 65-66
  third normal form, 67
raw databases, 62
NOT BETWEEN, 130-131
NOT EQUAL, 130
NOT EXISTS, 133
NOT IN, 131
NOT LIKE, 131-132
NOT NULL constraints, 52
NULL value checker, 180
NULL values, 16, 28-29
  inserting in tables, 78-80
numbers
  converting character strings to, 184-185
  converting to character strings, 185-186
numeric values, 24-25

O
object privileges, 302-303
objects, naming, 39
ODBC (Open Database Connectivity), 358
OLE DB, 359
OPEN, 340
opening cursors, 340-341
operators, 115
  arithmetic operators, 133
    addition, 133-134
combinations, 135-136
division, 135
multiplication, 134
subtraction, 134
comparison operators, 116
  combinations of, 118-119
  equality, 116
  less than, greater than, 118
  non-equality, 117
conjunctive operators, 126
  AND, 126-127
  OR, 127-129
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
negative operators, 129
  IS NOT NULL, 132
  NOT BETWEEN, 130-131
  NOT EQUAL, 130
  NOT EXISTS, 133
  NOT IN, 131
  NOT LIKE, 131-132
OR, 127-129
avoiding, 276-277
Oracle
  creating users, 289-290
cursors, 340
  closing, 343
queries, 99

compound queries
GROUP BY clause, 248-250
ORDER BY clause, 246-248
compound query operators, 240
EXCEPT, 245-246
INTERSECT, 244-245
UNION, 240-243
UNION ALL, 243-244
grouping, 128
simple queries

column aliases, 111
counting records in tables, 109-110
examples, 108-109
selecting data from another user’s table, 110
single versus compound, 239-240
querying system catalogs, 334-336

P

parameters
Microsoft SQL Server, 374
Oracle, 373
PERCENT_RANK() OVER, 351
performance
avoiding
HAVING clause, 277

querying system catalogs

PRODUCTS_TBL, 440, 442, 446
Oracle, 449
PUBLIC database, 307

Q

queries, 99

compound queries
GROUP BY clause, 248-250
ORDER BY clause, 246-248
compound query operators, 240
EXCEPT, 245-246
INTERSECT, 244-245
UNION, 240-243
UNION ALL, 243-244
grouping, 128
simple queries

column aliases, 111
counting records in tables, 109-110
examples, 108-109
selecting data from another user’s table, 110
single versus compound, 239-240
querying system catalogs, 334-336

P

parameters
Microsoft SQL Server, 374
Oracle, 373
PERCENT_RANK() OVER, 351
performance
avoiding
HAVING clause, 277

How can we make this index more useful? Email us at indexes@samspublishing.com
R

RANK() OVER, 351
raw databases, normalization, 62
RDBMS (relational database management system), 2
READ WRITE, 95
readability, formatting SQL statements, 269-271
records, 15-16
  counting records in tables, simple queries, 109-110
REFERENCES, 303
relational database management system. See RDBMS (relational database management system)
relational databases, 5
RELEASE SAVEPOINT, 92-93, 380
RELOAD, 302
remote databases, accessing, 357-358
  JDBC, 358
  ODBC (Open Database Connectivity), 358
  OLE DB, 359
through web interfaces, 359-360
  vendor connectivity products, 359
removing user access, 295-296
REPLACE, 173
representing column names with numbers, 158-159
RESOURCE, 307
restrictive conditions, SQL statements, 272-273
retrieving data, 250
REVOKE, 380
  controlling user access, 305-306
revoking privileges, 303-304
ROLLBACK, 11, 91-92, 380
ROLLBACK TO SAVEPOINT, 93-94
ROLLUP expression, 161-162
ROUND, 183
ROW_NUMBER() OVER, 351
rows, 41
rows of data, 15-16
RPAD, 182
RTRIM, 178
SAVEPOINT, 11, 92-93, 380
schemas, 37-39
  creating, 292-293
  dropping, 293-294
  versus users, 288
second normal form, 65-66
security
  controlling privileges, 308
    CREATE ROLE, 308-309
    DROP ROLE, 309
    SET ROLE, 309
  controlling user access, 304
    GRANT, 304-305
    GRANT OPTION, 305
  groups of privileges, 306-308
  on individual columns, 306
PUBLIC database, 306
REVOKE, 305-306
database security, 299-300
Internet, 361
privileges, 301
  object privileges, 302-303
  system privileges, 301-302
views, 315
security information, system catalogs, 332
SELECT, 10, 73, 99-102, 303, 380-381
  ANSI standard, 368
  case sensitivity, 106-107
  FROM clause, 102
  creating groups, 155-158
  DISTINCT, 102
  Microsoft SQL Server, 368
  Oracle, 368-369
  ORDER BY clause, 104-106
  subqueries, 227-228
  WHERE clause, 103-104
SELECT ANY TABLE, 302
selecting
data, 10
  from multiple tables, 207-208
data from another table, 110
self joins, 215-216
SET ROLE, 309
SET TRANSACTION, 11, 95
SHUTDOWN, 302
SIGN (sign values), 183
simple queries
  column aliases, 111
  counting records in tables, 109-110
  examples, 108-109
  selecting data from another user's table, 110
simplifying data access with views, 314-315
SIN, 183
single queries versus compound queries, 239-240
single quotation marks, 74
single-column indexes, 258
SOME, 124-126
sort operations, avoiding, 277-278
SQL (Structured Query Language), 2
direct versus embedded, 351
enterprises and, 355
back-end applications, 356
front-end applications, 356-357
generating SQL, 350
Internet and, 360
making data available to customers worldwide, 360-361
making data available to employees and privileged customers, 361
intranets and, 361-362
SQL commands, 9
data administration commands, 9, 11
  DCL (Data Control Language), 10-11
  DDL (Data Definition Language), 9-10
  DML (Data Manipulation Language), 10
  DQL (Data Query Language), 10
  transaction control commands, 9, 11
SQL extensions, 370
SQL sessions, 8
  CONNECT, 8
  DISCONNECT, 8-9
  EXIT, 8-9
SQL statement tuning, 267
  versus database tuning, 268
SQL statements formatting, 268-269
  arranging tables in FROM clause, 271
  ordering join conditions, 271-272
  for readability, 269-271
  restrictive conditions, 272-273
  interactive SQL statements, 373-374
SQL-2008, 3-4
SQRT (square root), 183
standards, table-naming standards, 12-13
storage, default storage, 191
stored procedures
  functions and, 343-346
  performance, 278
strings
  fixed-length strings, 23
  literal strings, 28
  varying-length strings, 23
Structured Query Language. See SQL (Structured Query Language)
subqueries, 225-227
  correlated, 233-234
  DELETE statements, 230
  inserted, 231-233
  INSERT statements, 228-229
  performance, 234-235
  SELECT statements, 227-228
  UPDATE statements, 229-230
SUBSTR, 175-176
substrings, 170
subtraction, arithmetic operators, 134
SUM, 145-146
summarized data, maintaining with views, 315-316
synonyms, 324
  creating, 324-325
  dropping, 325
  simple queries, 111
SYS, 331
system catalog objects, updating, 336
system catalogs, 327-330
  contents of, 331-332
  creating, 331
  database design information, 332
  performance statistics, 332-333
system catalogs

querying, 334-336
security information, 332
tables by implementation, 333-334
user data, 332
system privileges, 301-302

T

table aliases, 210
table-naming standards, 12-13
tables, 15, 39
adding
auto-incrementing columns, 45
columns, 44-45
ALTER TABLE, 44
arranging in FROM clause, 271
columns, 16, 40-41
modifying, 46
CREATE TABLE statement, 41-43
creating from existing tables, 46-48
creating from views, 321-322
data
deleting, 82-83
inserting, 74-75
dropping, 48-49
fields, 15
inserting data from another table, 76-78
modifying elements of, 44
naming conventions, 43
NULL values, 16
populating with new data, 74
primary keys, 16
records, 15-16
rows, 41
TAN, 183
TEXT data type, 24
third normal form, 67
TIME, 192
time zones, date functions, 194-195
TIMESTAMP, 192
TKPROF, 280
tools, database users, 296
transaction control commands, 9, 11
transactional control, database performance and, 95-96
transactions, 87-88
controlling, 88
COMMIT, 89-91
RELEASE SAVEPOINT, 94-95
ROLLBACK, 91-92
ROLLBACK TO SAVEPOINT, 93-94
SAVEPOINT, 92-93
SET TRANSACTION, 95
Transact-SQL, extensions, 371
TRANSLATE, 172-173
TRANSLATE function, 170
triggers, 346
creating, 346-348
dropping, 348
FOR EACH ROW, 348

U

UNION, 240-243
UNION ALL, 243-244
unique constraints, 50-51
unique indexes, 258-259
UPDATE statements, 303, 380
subqueries, 229-230
updating
data, 80
multiple columns in one or more records, 81-82
through views, 322-323
value of a single column, 80-81
system catalog objects, 336
UPPER, 174
USAGE, 303
user access, controlling, 304
GRANT, 304-305
GRANT OPTION, 305
groups of privileges, 306-308
on individual columns, 306
PUBLIC database, 306
REVOKE, 305-306
user data, system catalogs, 332
user-defined types, 29-30
user management, 288
creating users, 289
in Microsoft SQL Server, 290-291
in MySQL, 291
in Oracle, 289-290
user sessions, 295
users
altering, 294-295
creating, 289
in Microsoft SQL Server, 290-291
in MySQL, 291
creating groups in Oracle, 289-290
managing, 285-287
place in databases, 287
removing access, 295-296
versus schemas, 288
types of, 286

V
varying-length strings, 23
vendor connectivity products, 359
vendors, database vendors, 7-8
views, 313-314
creating, 316
from multiple tables, 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
simplifying data access, 314-315
updating data, 322-323
WITH CHECK OPTION, 320

W–Z
web-based database systems, 6-7
web interfaces, accessing remote databases, 359-360
WHERE, 82, 381
SELECT, 103-104
wildcards, performance, 275-276
windowed table functions, 351-352
Windows installation instructions
Microsoft SQL Server, 388-390
for MySQL, 383-385
for Oracle, 386-387
WITH CHECK OPTION, views, 320

XML, 352

How can we make this index more useful? Email us at indexes@samspublishing.com