

Ryan Stephens  
Ron Plew  
Arie D. Jones

The Fifth Edition of  
*Sams Teach Yourself  
SQL in 21 Days*

**More than  
48,000 sold!**

Sams **Teach Yourself**

# SQL

in **One Hour** a Day



**SAMS**

# **Sams Teach Yourself SQL in One Hour a Day**

**Copyright © 2009 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.

ISBN-13: 978-0-672-33025-4

ISBN-10: 0-672-33025-3

Library of Congress Cataloging-in-Publication Data:

Stephens, Ryan K.

Sams teach yourself SQL in one hour a day / Ryan Stephens, Ron Plew,  
Arie D. Jones. — 1st ed.

p. cm.

Previously published under title: Sams teach yourself SQL in 24 hours.

Includes indexes.

ISBN 978-0-672-33025-4 (pbk.)

I. SQL (Computer program language) I. Plew, Ronald R. II. Jones, Arie. III. Stephens,  
Ryan K. Sams teach yourself SQL in 24 hours. IV. Title.

QA76.73.S67P554 2009

005.13'3—dc22

2009014482

Printed in the United States of America

First Printing: June 2009

## **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 use of 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**

**Editor-in-Chief**

Mark Taub

**Acquisitions Editor**

Trina MacDonald

**Development Editor**

Songlin Qiu

**Managing Editor**

Kristy Hart

**Senior Project Editor**

Matthew Purcell

**Copy Editor**

Seth Kerney

**Indexer**

Lisa Stumpf

**Proofreader**

Language Logistics,  
LLC

**Technical Editor**

Clayton Rardin

Steven Romero

**Publishing**

**Coordinator**

Cindy Teeters

**Book Designer**

Gary Adair

**Compositor**

Nonie Ratcliff

# Introduction

Over the past decade the landscape of information technology has drastically shifted to a data-centric world. More than ever companies are looking for ways in which they can leverage their own data networks to make intelligent business decisions. This includes the ability to gather, store, and report effectively over possibly large sets of data in multiple formats. So the role of database administrators and developers have become strategically important in the proper implementation and care of these systems.

The cornerstone to any database project is the language that will be used in order to interact with the system. Fortunately, a consortium of entities has enacted a standard query language for database environments known as the ANSI SQL standard. This provides a commonality between all database querying languages by following this known standard and allows developers to learn the standard and then work on any given number of database systems with minor adjustments.

This book takes a focused approach on getting the reader the basics of the SQL language in order to allow them to have a solid foundation for future learning. Often in today's business environment, there is very little time to learn new things as our day-to-day functions consume large amounts of our time. By focusing on smaller lesson plans and logically segmenting the sections in a stepping stone fashion, the book allows readers to learn the SQL language at their own pace and within their own schedules.

## Who Should Read This Book?

This book is for people who want to learn the fundamentals of Structured Query Language (SQL) quickly. Through the use of countless examples, this book depicts all the major components of SQL, as well as options that are available with various database implementations. You should be able to apply what you learn here to relational databases in a traditional business setting.

## How Is This Book Organized?

This book is divided into seven parts, which logically break down the structure of ANSI SQL into easily learnable sections:

- Part I, comprised of the first seven lessons, discusses the basic concepts behind SQL and mainly focuses on the SQL query.

- Part II includes topics on the art of database design, such as creating databases and database objects properly, which is often the foundation of RDBMS application development.
- Part III focuses on data manipulation and using SQL to perform `UPDATES`, `INSERTs`, and `DELETES` of data within your database. These will be the staple commands that you will use on a day-to-day basis.
- Part IV is dedicated to database administration, which covers such topics as security, management, and performance, enabling you to maintain the integrity and performance of your database instance.
- Part V focuses on more advanced SQL objects such as triggers and stored procedures. Using these objects will allow you to perform more sophisticated data manipulation techniques that would otherwise be difficult in standard SQL syntax.
- Part VI covers more advanced SQL programming. Advanced SQL programming will allow you to perform more advanced queries and manipulation of the data within your database.
- Part VII presents you with SQL in various database implementations. SQL extensions such as PL/SQL allow you to take advantage of unique attributes within a particular database environment, such as Oracle.
- This book also contains four appendices, which provide you with not only the answers to the exercises in each lesson but also the code examples to create and populate the tables used in the book. Two additional appendices are located at <http://www.informit.com/store/products.aspx?isbn=0672330253> under the extras tab.

After studying this book, you should have an excellent understanding of SQL and should know how to apply SQL in the real world.

## NOTE

If you are familiar with the basics and history of SQL, we suggest you skim the first lesson and begin in earnest with Lesson 2 “Introducing the Query.”

The syntax of SQL is explained and then brought to life in examples using MySQL, which is the closest implementation of the ANSI SQL standard syntax, as well as Oracle Express edition, which demonstrates some of the extensions to ANSI SQL.

## Conventions Used in This Book

This book uses the following typeface conventions:

- Menu names are separated from menu options by a comma. For example, File, Open means select the Open option from the File menu.
- New terms appear in *italic*.
- In some listings, we've included both the input and output (**Input/Output ▼**). For these, all code that you type in (input) appears in **boldface monospace**. Output appears in standard monospace. The Combination icon indicates that both input and output appear in the code.
- The **Input ▼** and **Output ▼** icons also identify the nature of the code.
- Many code-related terms within the text also appear in monospace.
- Placeholders in code appear in *italic monospace*.
- When a line of code is too long to fit on one line of this book, it is broken at a convenient place and continued to the next line. A code continuation character (↪) precedes the continuation of a line of code. (You should type a line of code that has this character as one long line without breaking it.)
- Paragraphs that begin with the **Analysis ▼** icon explain the preceding code example.
- The **Syntax ▼** icon identifies syntax statements.
- Special design features enhance the text material:

### NOTE

Notes explain interesting or important points that can help you understand SQL concepts and techniques.

### TIP

Tips are little pieces of information that will help you in real-world situations. Tips often offer shortcuts to make a task easier or faster.

### CAUTION

Cautions provide information about detrimental performance issues or dangerous errors. Pay careful attention to Cautions.

## Using MySQL for Hands-on Exercises

We have chosen to use MySQL for hands-on exercises in this edition. In previous editions, we left it up to the reader to obtain access to any SQL implementation. We decided that it would be better to provide the reader with an open-source SQL database that allowed all readers to start on the same level with the same software. We chose MySQL because it is the most popular open-source database available today, and it is easy to download and use.

Unfortunately, MySQL does have its limitations. There are several features of standard SQL that are not supported by MySQL. We have attempted to distinguish between the exercises that support MySQL and those that do not. Those exercises that do not will mainly focus on using Oracle Enterprise edition, instead. The beauty of SQL is that it is a standard language, although each implementation does have its differences. After using MySQL to understand the basic fundamentals of SQL, you should be able to easily apply the concepts you have learned to any SQL implementation.

## About the Book's Source Code

In the appendices, you will find the source code for creating all of the objects used throughout the book. This includes all of the tables and data that is used. Additionally, the source code will be available for download from the publisher's website. This will allow you to simply cut and paste entries into your interface instead of spending the majority of your time typing and enable you to focus more clearly on the material.

# LESSON 3

## **Expressions, Conditions, and Operators**

In Lesson 2, “Introducing the Query,” you used `SELECT` and `FROM` to manipulate data in interesting (and useful) ways. In this lesson, you learn more about `SELECT` and `FROM`. You will expand the basic query with some new terms, a new clause, and a group of handy items called *operators*. By the end of this lesson, you will

- Know what an expression is and how to use it.
- Know what a condition is and how to use it.
- Be familiar with the basic uses of the `WHERE` clause.
- Be able to use arithmetic, comparison, character, logical, and set operators.
- Have a working knowledge of some miscellaneous operators.

### **NOTE**

We used Oracle and MySQL to generate this lesson's examples. Other implementations of SQL might differ slightly in the way in which commands are entered or output is displayed, but the results are basically the same for all implementations that conform to the ANSI standard.

### **NOTE**

This lesson is one of the longest in the book and also one of the most important as it lays the foundation for most of the other lessons. In this lesson we provide many examples for you to absorb. Do not try to remember every specific example but rather learn the concepts behind them. The lessons to follow will give you plenty of practice in implementing what you will learn.

## Working with Query Expressions

The definition of an expression is simple: An *expression* returns a value. Expression types are very broad, covering different data types such as String, Numeric, and Boolean. In fact, pretty much anything following a clause (SELECT or FROM, for example) is an expression. In the following example, AMOUNT is an expression that returns the value contained in the AMOUNT column:

### Syntax ▼

---

```
SELECT AMOUNT FROM CHECKS;
```

Of course, the following is also considered a numerical expression. Remember that the key to an expression is that it returns a value.

### Syntax ▼

---

```
SELECT AMOUNT*10 FROM CHECKS;
```

In the following statement, NAME, ADDRESS, PHONE, and ADDRESSBOOK are expressions:

### Syntax ▼

---

```
SELECT NAME, ADDRESS, PHONE  
FROM ADDRESSBOOK;
```

Now, examine the following WHERE clause:

### Syntax ▼

---

```
WHERE NAME = 'BROWN'
```

It contains a condition, NAME = 'BROWN', which is an example of a Boolean expression. NAME = 'BROWN' will be either TRUE or FALSE, depending on the condition =.

## Placing Conditions on Queries

If you ever want to find a particular item or group of items in your database, you need one or more conditions. Conditions are contained in the WHERE clause. In the preceding example, the condition is

### Syntax ▼

---

```
NAME = 'BROWN'
```



To find everyone in your organization who worked more than 100 hours last month, your condition would be

Syntax ▼

```
NUMBEROFHOURS > 100
```

Conditions enable you to make selective queries. In their most common form, conditions comprise a variable, a constant, and a comparison operator. In the first example, the variable is NAME, the constant is 'BROWN', and the comparison operator is =. In the second example, the variable is NUMBEROFHOURS, the constant is 100, and the comparison operator is >. You need to know about two more elements before you can write conditional queries: the WHERE clause and operators.

The syntax of the WHERE clause is

Syntax ▼

```
WHERE <SEARCH CONDITION>
```

SELECT, FROM, and WHERE are the three most frequently used clauses in SQL. WHERE simply causes your queries to be more selective. Without the WHERE clause, the most useful thing you could do with a query is display all records in the selected table(s)—for example,

Input ▼

```
SQL> SELECT * FROM BIKES;
```

lists all rows of data in the table BIKES.

Output ▼

NAME	FRAMESIZE	COMPOSITION	MILESRIIDEN	TYPE
TREK 2300	22.5	CARBON FIBER	3500	RACING
BURLEY	22	STEEL	2000	TANDEM
GIANT	19	STEEL	1500	COMMUTER
FUJI	20	STEEL	500	TOURING
SPECIALIZED	16	STEEL	100	MOUNTAIN
CANNONDALE	22.5	ALUMINUM	3000	RACING

6 rows selected.

If you wanted a particular bike, you could type

### Input ▼

```
SQL> SELECT *
      2 FROM BIKES
      3 WHERE NAME = 'BURLEY';
```

which would yield only one record:

### Output ▼

NAME	FRAMESIZE	COMPOSITION	MILESRIIDEN	TYPE
BURLEY	22	STEEL	2000	TANDEM

1 rows selected.

These simple examples show how you can place a condition on the data that you want to retrieve.

## Learning How to Use Operators

Operators are the elements you use inside an expression to articulate how you want specified conditions to retrieve data. Operators fall into six groups: arithmetic, comparison, character, logical, set, and miscellaneous. SQL utilizes three types of operators: arithmetic, comparison, and logical.

### Arithmetic Operators

The arithmetic operators are plus (+), minus (-), divide (/), multiply (\*), and modulo (%). The first four are self-explanatory. Modulo returns the integer remainder of a division. Here are two examples:

```
5 % 2 = 1
6 % 2 = 0
```

The modulo operator does not work with data types that have decimals, such as Real or Number.

If you place several of these arithmetic operators in an expression without any parentheses, the operators are resolved in this order: multiplication, division, modulo, addition, and subtraction. For example, the expression

```
2*6+9/3
```

equals

$12 + 3 = 15$

However, the expression

$2 * (6 + 9) / 3$

equals

$2 * 15 / 3 = 10$

Watch where you put those parentheses! Sometimes the expression does exactly what you tell it to do, rather than what you want it to do. The same holds true for SQL.

The following sections examine the arithmetic operators in some detail and give you a chance to write some queries.

## Plus (+)

You can use the plus sign in several ways. Type the following statement to display the PRICE table:

### Input/Output ▼

```
SQL> SELECT * FROM PRICE;
ITEM                WHOLESAL
-----
TOMATOES             .34
POTATOES             .51
BANANAS              .67
TURNIPS              .45
CHEESE               .89
APPLES               .23
6 rows selected.
```

Now type

### Input ▼

```
SQL> SELECT ITEM, WHOLESAL, WHOLESAL + 0.15
2 FROM PRICE;
```

Here the + adds 15 cents to each price to produce the following:

**Output ▼**


---

ITEM	WHOLESALE	WHOLESALE+0.15
TOMATOES	.34	.49
POTATOES	.51	.66
BANANAS	.67	.82
TURNIPS	.45	.60
CHEESE	.89	1.04
APPLES	.23	.38

6 rows selected.

**Analysis ▼**


---

What is this last column with the unattractive column heading `WHOLESALE+0.15`? It's not in the original table. SQL allows you to create a virtual or derived column by combining or modifying existing columns.

Retype the original entry:

**Input ▼**


---

```
SQL> SELECT * FROM PRICE;
```

The following table results:

**Output ▼**


---

ITEM	WHOLESALE
TOMATOES	.34
POTATOES	.51
BANANAS	.67
TURNIPS	.45
CHEESE	.89
APPLES	.23

6 rows selected.

**Analysis ▼**


---

The output confirms that the original data has not been changed and that the column heading `WHOLESALE+0.15` is not a permanent part of it. In fact, the column heading is so unattractive that you should do something about it.

Type the following:

Input ▼

```
SQL> SELECT ITEM, WHOLESale, (WHOLESale + 0.15) RETAIL
      2 FROM PRICE;
```

Here’s the result:

Output ▼

ITEM	WHOLESale	RETAIL
-----	-----	-----
TOMATOES	.34	.49
POTATOES	.51	.66
BANANAS	.67	.82
TURNIPS	.45	.60
CHEESE	.89	1.04
APPLES	.23	.38
6 rows selected.		

Analysis ▼

This is wonderful! Not only can you create new output columns, but you can also rename them on the fly. You can rename any of the columns using the syntax <column\_name> <alias>. (Note the space between the column\_name and alias.)

For example, the query

Input ▼

```
SQL> SELECT ITEM PRODUCE, WHOLESale, WHOLESale + 0.25 RETAIL
      2 FROM PRICE;
```

renames the columns as follows:

Output ▼

PRODUCE	WHOLESale	RETAIL
-----	-----	-----
TOMATOES	.34	.59
POTATOES	.51	.76
BANANAS	.67	.92
TURNIPS	.45	.70
CHEESE	.89	1.14
APPLES	.23	.48
6 rows in set (0.00 sec)		



**NOTE**

Some implementations of SQL use the syntax <column name = alias>. The preceding example would be written as follows:

```
SQL> SELECT ITEM = PRODUCE,
       2 WHOLESALE,
       3 WHOLESALE + 0.25 = RETAIL,
       4 FROM PRICE;
```

Alternatively, the SQL standard allows you to use the AS keyword, which is implemented in many database systems and looks like the following:

```
SQL> SELECT ITEM AS PRODUCE,
       2 WHOLESALE,
       3 WHOLESALE + 0.25 = RETAIL,
       4 FROM PRICE;
```

Check your implementation for the exact syntax.

**NOTE**

MySQL allows you to present your column alias in mixed case.

You might be wondering what use aliasing is if you are not using command-line SQL. Fair enough. Have you ever wondered how report builders work? Some day, when you are asked to write a report generator, you'll remember this and not spend weeks reinventing what Dr. Codd and IBM have wrought.

In some implementations of SQL, the plus sign does double duty as a character operator. You'll see that side of the plus sign a little later in this lesson.

**Minus (-)**

Minus also has two uses. First, it can change the sign of a number. You can use the table HILOW to demonstrate this function.

**Input/Output ▼**

```
SQL> SELECT * FROM HILOW;
STATE      LOWS      HIGHS
-----
CA         -50       120
FL          20       110
LA          15        99
ND         -70       101
NE         -60       100
```

For example, here’s a way to manipulate the data:

**Input/Output ▼**

```
SQL> SELECT STATE, - LOWS, - HIGHS
2 FROM HILOW;
STATE          LOWS      HIGHS
-----
CA              50      -120
FL             -20      -110
LA             -15      -99
ND              70     -101
NE              60     -100
```

**NOTE**

Notice that the minus sign was reversed on the temperatures.

The second (and obvious) use of the minus sign is to subtract one column from another—for example,

**Input/Output ▼**

```
SQL> SELECT STATE,
2   LOWS,
3   HIGHS,
4   (-HIGHS - LOWS) DIFFERENCE
5 FROM HILOW;
STATE          LOWS      HIGHS  DIFFERENCE
-----
CA             -50       120         170
FL              20       110          90
LA              15        99          84
ND             -70       101         171
NE             -60       100         160
```

If you accidentally use the minus sign on a character field, you get something like this:

**Input/Output ▼**

```
SQL> SELECT -STATE FROM HILOW;

ERROR:
ORA-01722: invalid number
no rows selected
```

The exact error message varies with implementation. Here is an example using MySQL:

### Input/Output ▼

```
mysql> select -state
      -> from hilow;
+-----+
| -state |
+-----+
|      0 |
|      0 |
|      0 |
|      0 |
+-----+
4 rows in set (0.00 sec)
```

MySQL evaluated the SELECT statement, but as you can see, the results are rather meaningless.

### Divide (/)

The division operator has only the one obvious meaning. Using the table PRICE, type the following:

### Input/Output ▼

```
SQL> SELECT * FROM PRICE;
ITEM                WHOLESale
-----
TOMATOES             .34
POTATOES             .51
BANANAS              .67
TURNIPS              .45
CHEESE               .89
APPLES               .23
6 rows selected.

mysql> select * from price;
+-----+-----+
| item  | wholesale |
+-----+-----+
| TOMATOES | 0.34 |
| POTATOES | 0.51 |
| BANANAS  | 0.67 |
| TURNIPS  | 0.45 |
| CHEESE   | 0.89 |
| APPLES   | 0.23 |
+-----+-----+

6 rows in set (0.26 sec)
```



You can show the effects of a two-for-one sale by typing the next statement:

**Input/Output ▼**

```
SQL> SELECT ITEM, WHOLESale, (WHOLESale/2) SALEPRICE
      2 FROM PRICE;
ITEM          WHOLESale SALEPRICE
-----
TOMATOES      .34       .17
POTATOES      .51       .255
BANANAS       .67       .335
TURNIPS       .45       .225
CHEESE        .89       .445
APPLES        .23       .115
6 rows selected.
```

The same example in MySQL would be:

**Input/Output ▼**

```
mysql> select ITEM, WHOLESale, (WHOLESale/2) Saleprice
-> from price;
+-----+-----+-----+
| ITEM   | WHOLESale | Saleprice |
+-----+-----+-----+
| TOMATOES | 0.34 | 0.1700 |
| POTATOES | 0.51 | 0.2550 |
| BANANAS  | 0.67 | 0.3350 |
| TURNIPS  | 0.45 | 0.2250 |
| CHEESE   | 0.89 | 0.4450 |
| APPLES   | 0.23 | 0.1150 |
+-----+-----+-----+
6 rows in set (0.26 sec)
```

The use of division in the preceding SELECT statement is straightforward (except that coming up with half pennies can be tough).

**Multiply (\*)**

The multiplication operator is also straightforward. Again, using the PRICE table, type the following:

**Input/Output ▼**

```
SQL> SELECT * FROM PRICE;
ITEM          WHOLESale
-----
TOMATOES      .34
POTATOES      .51
```

```

BANANAS          .67
TURNIPS          .45
CHEESE           .89
APPLES           .23
6 rows selected.

```

The output from this query reflects an across-the-board 10% discount. The actual data in the table has not changed.

### Input/Output ▼

```
SQL> SQL> SELECT ITEM, WHOLESAL, WHOLESAL * 0.9 NEWPRICE
      2 FROM PRICE;
```

ITEM	WHOLESAL	NEWPRICE
TOMATOES	.34	.306
POTATOES	.51	.459
BANANAS	.67	.603
TURNIPS	.45	.405
CHEESE	.89	.801
APPLES	.23	.207

6 rows selected.

The same example in MySQL would be:

### Input/Output ▼

```
mysql> select Item,
      -> Wholesale, Wholesale * 0.9 "New Price"
      -> from price;
```

Item	Wholesale	New Price
TOMATOES	0.34	0.31
POTATOES	0.51	0.46
BANANAS	0.67	0.60
TURNIPS	0.45	0.41
CHEESE	0.89	0.80
APPLES	0.23	0.21

6 rows in set (0.00 sec)

### NOTE

One last thing about aliases: You can give your column a two-word heading by using quotes to surround your aliases. Sometimes this will be single quotes and sometimes it will be double quotes. Please check your specific implementation's documentation to see what it allows.

These operators enable you to perform powerful calculations in a SELECT statement.

**Modulo (%)**

The modulo operator returns the integer remainder of the division operation. Using the table REMAINS, type the following:

**Input/Output ▼**

```
SQL> SELECT * FROM REMAINS;
NUMERATOR  DENOMINATOR
-----
      10           5
       8           3
      23           9
      40          17
     1024          16
      85          34
6 rows selected.
```

The same example in MySQL would be:

**Input/Output ▼**

```
mysql> select * from remains;
+-----+-----+
| numerator | denominator |
+-----+-----+
|      10 |           5 |
|       8 |           3 |
|      23 |           9 |
|      40 |          17 |
|     1024 |          16 |
|      85 |          34 |
+-----+-----+
6 rows in set (0.43 sec)
```

You can also create a new output column, REMAINDER, to hold the values of NUMERATOR % DENOMINATOR:

**Input/Output ▼**

```
SQL> SELECT NUMERATOR,
2  DENOMINATOR,
3  NUMERATOR%DENOMINATOR REMAINDER
4  FROM REMAINS;

NUMERATOR  DENOMINATOR  REMAINDER
```



```
-----
      10      5      0
      8      3      2
     23      9      5
     40     17      6
    1024     16      0
     85     34     17
6 rows selected.
```

The same example in MySQL would be:

**Input/Output ▼**

```
mysql> select numerator, denominator, numerator%denominator remainder
-> from remains;

+-----+-----+-----+
| numerator | denominator | remainder |
+-----+-----+-----+
|      10 |          5 |          0 |
|       8 |          3 |          2 |
|      23 |          9 |          5 |
|      40 |         17 |          6 |
|     1024 |         16 |          0 |
|       85 |         34 |         17 |
+-----+-----+-----+
6 rows in set (0.01 sec)
```

**Analysis ▼**

Some implementations of SQL implement modulo as a function called MOD (see Lesson 7, “Molding Data with Built-in Functions”). The following statement produces results that are identical to the results in the preceding statement:

**Input/Output ▼**

```
SQL> SELECT NUMERATOR,
2  DENOMINATOR,
3  MOD(NUMERATOR,DENOMINATOR) REMAINDER
4  FROM REMAINS;

NUMERATOR  DENOMINATOR  REMAINDER
-----
      10      5      0
      8      3      2
     23      9      5
     40     17      6
    1024     16      0
     85     34     17
6 rows selected.
```

The same example in MySQL would be:

### Input/Output ▼

```
mysql> select numerator, denominator,
-> mod(numerator,denominator) remainder
-> from remains;
```

numerator	denominator	remainder
10	5	0
8	3	2
23	9	5
40	17	6
1024	16	0
85	34	17

6 rows in set (0.00 sec)

### Precedence

*Precedence* is the order in which an implementation will evaluate different operators in the same expression. This section examines the use of precedence in a SELECT statement. Using the table PRECEDENCE, type the following:

### Input/Output ▼

```
SQL> SELECT * FROM PRECEDENCE;
```

N1	N2	N3	N4
1	2	3	4
13	24	35	46
9	3	23	5
63	2	45	3
7	2	1	4

5 rows selected.

```
mysql> select * from precedence;
```

n1	n2	n3	n4
1	2	3	4
13	24	35	46
9	3	23	5
63	2	45	3
7	2	1	4

5 rows in set (0.00 sec)

Use the following code segment to test precedence:

Input/Output ▼

```
SQL> SELECT
  2 N1+N2*N3/N4,
  3 (N1+N2)*N3/N4,
  4 N1+(N2*N3)/N4
  5 FROM PRECEDENCE;
N1+N2*N3/N4 (N1+N2)*N3/N4 N1+(N2*N3)/N4
-----
      2.5          2.25          2.5
31.26087    28.152174    31.26087
      22.8          55.2          22.8
      93           975           93
      7.5          2.25          7.5
```

5 rows selected.

```
mysql> select n1+n2*n3/n4,
-> (n1+n2)*n3/n4,
-> n1+(n2*n3)/n4
-> from precedence;
+-----+-----+-----+
| n1+n2*n3/n4 | (n1+n2)*n3/n4 | n1+(n2*n3)/n4 |
+-----+-----+-----+
|          2.50 |             9 |          2.50 |
|          31.26 |          1295 |          31.26 |
|          22.80 |           276 |          22.80 |
|          93.00 |         2925 |          93.00 |
|           7.50 |             9 |           7.50 |
+-----+-----+-----+
5 rows in set (0.00 sec)
```

Notice that the first and last columns are identical. If you added a fourth column  $N1+N2*(N3/N4)$ , its values would also be identical to those of the current first and last columns. The rules for precedence follow the usual algebraic set in that values are normally executed in the following order moving left to right.

- 1. Parentheses
- 2. Multiplication/division
- 3. Addition/subtraction

Analysis ▼

Quite simply, values inside parentheses are computed first, then multiplication or division operations are performed, and lastly addition and subtraction operations are

performed. These rules are important to remember as you start to write more complicated calculations to analyze data.

## Comparison Operators

True to their name, comparison operators compare expressions and return one of three values: TRUE, FALSE, or UNKNOWN. Wait a minute! Unknown? TRUE and FALSE are self-explanatory, but what is UNKNOWN?

To understand how you could get an UNKNOWN, you need to know a little about the concept of NULL. In database terms, NULL is the absence of data in a field. It does not mean that a column has a zero or a blank in it. A zero or a blank is a value. NULL means nothing is in that field.

If you make a comparison such as `Field = 9` and the only acceptable value for `Field` is NULL, the comparison will come back UNKNOWN. Because UNKNOWN is an uncomfortable condition, most flavors of SQL change UNKNOWN to FALSE and provide a special operator, `IS NULL`, to test for a NULL condition.

Here’s an example of NULL: Suppose an entry in the `PRICE` table does not contain a value for `WHOLESALE`. The results of a query might look like this:

### Input/Output ▼

```
SQL> SELECT * FROM PRICE;
```

ITEM	WHOLESALE
-----	-----
TOMATOES	.34
POTATOES	.51
BANANAS	.67
TURNIPS	.45
CHEESE	.89
APPLES	.23
ORANGES	

7 rows selected.

### Analysis ▼

Notice that no value appears in the `WHOLESALE` field position for `ORANGES`. The value of the field `WHOLESALE` for `ORANGES` is NULL. The NULL is noticeable in this case because it is in a numeric column. However, if the NULL appeared in the `ITEM` column, it would be impossible to tell the difference between NULL and a blank.



Try to find the NULL:

### Input/Output ▼

---

```
SQL> SELECT *
      2 FROM PRICE
      3 WHERE WHOLESALE IS NULL;
ITEM          WHOLESALE
-----
ORANGES

1 rows selected.
```

As you can see by the output, ORANGES is the only item whose value for WHOLESALE is NULL, or does not contain a value. What if you use the equal sign (=) instead?

### Input/Output ▼

---

```
SQL>SELECT *
      2 FROM PRICE
      3 WHERE WHOLESALE = NULL;

no rows selected
```

### Analysis ▼

---

You wouldn't find anything because the comparison `WHOLESALE = NULL` returned a FALSE—the result was unknown. It would be more appropriate to use an `IS NULL` instead of `=`, changing the WHERE statement to `WHERE WHOLESALE IS NULL`. In this case, you would get all the rows where a NULL existed.

This example also illustrates both the use of the most common comparison operator (`=`) and the playground of all comparison operators, the WHERE clause. You already know about the WHERE clause, so here's a brief look at the equal sign.

### Equal Sign (=)

Earlier today you saw how some implementations of SQL use the equal sign in the SELECT clause to assign an alias. In the WHERE clause, the equal sign is the most commonly used comparison operator. Used alone, the equal sign is a very convenient way of selecting one value out of many. Try this:

### Input/Output ▼

---

```
SQL> SELECT * FROM FRIENDS;
LASTNAME      FIRSTNAME      AREACODE PHONE    ST ZIP
-----
BUNDY         AL             100 555-1111 IL 22333
```



```

MEZA          AL          200 555-2222 UK
MERRICK       BUD         300 555-6666 CO 80212
MAST          JD          381 555-6767 LA 23456
BULHER        FERRIS      345 555-3223 IL 23332
5 rows selected.

```

Let's find JD's row. (On a short list this task appears trivial, but you might have more friends than we do—or you might have a list with thousands of records.)

## Input/Output ▼

```

SQL> SELECT *
      2 FROM FRIENDS
      3 WHERE FIRSTNAME = 'JD';

```

LASTNAME	FIRSTNAME	AREACODE	PHONE	ST	ZIP
MAST	JD	381	555-6767	LA	23456

1 rows selected.

```

mysql> select * from friends
      -> where firstname = 'JD';

```

lastname	firstname	areacode	phone	st	zip
MAST	JD	381	555-6767	LA	23456

1 row in set (0.37 sec)

We got the result that we expected. Try this:

## Input/Output ▼

```

SQL> SELECT *
      2 FROM FRIENDS
      3 WHERE FIRSTNAME = 'AL';

```

LASTNAME	FIRSTNAME	AREACODE	PHONE	ST	ZIP
BUNDY	AL	100	555-1111	IL	22333
MEZA	AL	200	555-2222	UK	

2 rows selected.

## NOTE

Here you see that = can pull in multiple records. Notice that ZIP is blank on the second record. ZIP is a character field (you learn how to create and populate tables in Lesson 9, "Creating and Maintaining Tables"), and in this particular record, the NULL demonstrates that a NULL in a character field is impossible to differentiate from a blank field.

Here's another very important lesson concerning case sensitivity:

### Input/Output ▼

---

```
SQL> SELECT FIRSTNAME FROM FRIENDS
      2 WHERE FIRSTNAME = 'BUD';
```

```
FIRSTNAME
```

```
-----
```

```
BUD
```

```
1 row selected.
```

```
mysql> select firstname from friends where firstname = 'BUD';
```

```
+-----+
```

```
| firstname |
```

```
+-----+
```

```
| BUD      |
```

```
+-----+
```

```
1 row in set (0.00 sec)
```

Now try this:

### Input/Output ▼

---

```
SQL> select FIRSTNAME from friends
      2 where firstname = 'Bud';
```

```
no rows selected.
```

```
mysql> select firstname
```

```
    -> from friends
```

```
    -> where firstname = 'bud';
```

```
+-----+
```

```
| firstname |
```

```
+-----+
```

```
| BUD      |
```

```
+-----+
```

```
1 row in set (0.01 sec)
```

### Analysis ▼

---

Even though SQL syntax is not case sensitive, data within it is, at least in some implementations. As you can see in the preceding examples, data stored in an Oracle database (SQL\*Plus) is case sensitive, whereas the MySQL example demonstrates the opposite.

Most companies prefer to store data in uppercase to provide data consistency. I recommend that you always store data either in all uppercase or in all lowercase, regardless of what type of database you are working in. Mixing case might create difficulties when you try to retrieve accurate data through comparisons in the WHERE clause.

Greater Than (>) and Greater Than or Equal To (>=)

The greater than operator (>) works like this:

Input/Output ▼

```
SQL> SELECT *
      2 FROM FRIENDS
      3 WHERE AREACODE > 300;
LASTNAME      FIRSTNAME      AREACODE PHONE      ST ZIP
-----
MAST          JD              381 555-6767 LA 23456
BULHER        FERRIS         345 555-3223 IL 23332
```

2 rows selected.  
This example found all the area codes greater than (but not including) 300. To include 300, type this:

Input/Output ▼

```
SQL> SELECT *
      2 FROM FRIENDS
      3 WHERE AREACODE >= 300;
LASTNAME      FIRSTNAME      AREACODE PHONE      ST ZIP
-----
MERRICK        BUD              300 555-6666 CO 80212
MAST          JD              381 555-6767 LA 23456
BULHER        FERRIS         345 555-3223 IL 23332
```

3 rows selected.

```
mysql> select * from friends
-> where areacode >= 300;
+-----+-----+-----+-----+-----+-----+
| lastname | firstname | areacode | phone | st | zip |
+-----+-----+-----+-----+-----+-----+
| MERRICK | BUD      | 300      | 555-6666 | CO | 80212 |
| MAST    | JD       | 381      | 555-6767 | LA | 23456 |
| BULHER  | FERRIS   | 345      | 555-3223 | IL | 23332 |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.34 sec)
```

With this change you get area codes starting at 300 and going up. You could achieve the same results with the statement AREACODE > 299.

NOTE

Notice that no quotes surround 300 in either of the two prior SQL statements. Number-defined fields do not require quotes.

## Less Than (<) and Less Than or Equal To (<=)

As you might expect, these comparison operators work the same way as > and >= work, only in reverse:

### Input/Output ▼

```
SQL> SELECT *
      2 FROM FRIENDS
      3 WHERE ST < 'LA';
```

LASTNAME	FIRSTNAME	AREACODE	PHONE	ST	ZIP
BUNDY	AL	100	555-1111	IL	22333
MERRICK	BUD	300	555-6666	CO	80212
BULHER	FERRIS	345	555-3223	IL	23332

3 rows selected.

```
mysql> select * from friends where st < 'LA';
```

lastname	firstname	areacode	phone	st	zip
BUNDY	AL	100	555-1111	IL	22333
MERRICK	BUD	300	555-6666	CO	80212
BULHER	FERRIS	345	555-3223	IL	23332

3 rows in set (0.00 sec)

### NOTE

In an Oracle database, if the column has only two characters, the column name is shortened to two characters in the returned rows. If the column name had been COWS, it would come out CO. The widths of AREACODE and PHONE are wider than their column names, so they are not truncated.

### Analysis ▼

Wait a minute. Did you just use < on a character field? Of course you did. You can use any of these operators on any data type. The result varies by data type. For example, use lowercase in the following state search:

### Input/Output ▼

```
SQL> SELECT *
      2 FROM FRIENDS
      3 WHERE STATE < 'la';
```

LASTNAME	FIRSTNAME	AREACODE	PHONE	ST	ZIP
BUNDY	AL	100	555-1111	IL	22333
MEZA	AL	200	555-2222	UK	

```

MERRICK      BUD      300 555-6666 CO 80212
MAST         JD       381 555-6767 LA 23456
BULHER       FERRIS   345 555-3223 IL 23332
5 rows selected.

```

```
mysql> select * from friends where st < 'la';
```

lastname	firstname	areacode	phone	st	zip
BUNDY	AL	100	555-1111	IL	22333
MERRICK	BUD	300	555-6666	CO	80212
BULHER	FERRIS	345	555-3223	IL	23332

```
3 rows in set (0.00 sec)
```

Uppercase is usually sorted before lowercase; therefore, the uppercase codes returned are less than la. Again, to be safe, check your implementation.

### TIP

To be sure of how these operators will behave, check your language tables. Most PC implementations use the ASCII tables.

To include the state of Louisiana in the original search, type

### Input/Output ▼

```
SQL> SELECT *
```

```
2 FROM FRIENDS
```

```
3 WHERE STATE <= 'LA';
```

LASTNAME	FIRSTNAME	AREACODE	PHONE	ST	ZIP
BUNDY	AL	100	555-1111	IL	22333
MERRICK	BUD	300	555-6666	CO	80212
MAST	JD	381	555-6767	LA	23456
BULHER	FERRIS	345	555-3223	IL	23332

4 rows selected.

```
mysql> select * from friends where st <= 'LA';
```

lastname	firstname	areacode	phone	st	zip
BUNDY	AL	100	555-1111	IL	22333
MERRICK	BUD	300	555-6666	CO	80212
MAST	JD	381	555-6767	LA	23456
BULHER	FERRIS	345	555-3223	IL	23332

```
4 rows in set (0.00 sec)
```

## Inequalities (< > or !=)

When you need to find everything except for certain data, use the inequality symbol, which can be either < > or !=, depending on your SQL implementation. For example, to find everyone who is not AL, type this:

### Input/Output ▼

```
SQL> SELECT *
      2 FROM FRIENDS
      3 WHERE FIRSTNAME <> 'AL';
```

LASTNAME	FIRSTNAME	AREACODE	PHONE	ST	ZIP
MERRICK	BUD	300	555-6666	CO	80212
MAST	JD	381	555-6767	LA	23456
BULHER	FERRIS	345	555-3223	IL	23332

3 rows selected.

```
mysql> select * from friends where firstname <> 'AL';
```

lastname	firstname	areacode	phone	st	zip
MERRICK	BUD	300	555-6666	CO	80212
MAST	JD	381	555-6767	LA	23456
BULHER	FERRIS	345	555-3223	IL	23332

4 rows in set (0.00 sec)

To find everyone not living in California, type this:

### Input/Output ▼

```
SQL> SELECT *
      2 FROM FRIENDS
      3 WHERE STATE != 'CA';
```

LASTNAME	FIRSTNAME	AREACODE	PHONE	ST	ZIP
BUNDY	AL	100	555-1111	IL	22333
MEZA	AL	200	555-2222	UK	
MERRICK	BUD	300	555-6666	CO	80212
MAST	JD	381	555-6767	LA	23456
BULHER	FERRIS	345	555-3223	IL	23332

5 rows selected.

```
mysql> select * from friends where st != 'CA';
```

lastname	firstname	areacode	phone	st	zip
BUNDY	AL	100	555-1111	IL	22333
MEZA	AL	200	555-2222	UK	

MERRICK	BUD	300	555-6666	CO	80212
MAST	JD	381	555-6767	LA	23456
BULHER	FERRIS	345	555-3223	IL	23332

5 rows in set (0.00 sec)

NOTE

Notice that both symbols, <> and !=, can express “not equal” in the two implementations we have shown you.

Character Operators

You can use character operators to manipulate the way character strings are represented, both in the output of data and in the process of placing conditions on data to be retrieved. This section describes two character operators: the LIKE operator and the || operator, the latter of which conveys the concept of character concatenation.

LIKE

What if you wanted to select parts of a database that fit a pattern but weren’t quite exact matches? You could use the equal sign and run through all the possible cases, but that process would be boring and time-consuming. Instead, you can use LIKE. Consider the following:

Input/Output ▼

```
SQL> SELECT * FROM PARTS;
```

NAME	LOCATION	PARTNUMBER
-----	-----	-----
APPENDIX	MID-STOMACH	1
ADAMS APPLE	THROAT	2
HEART	CHEST	3
SPINE	BACK	4
ANVIL	EAR	5
KIDNEY	MID-BACK	6

6 rows selected.

How can you find all the parts located in the back? A quick visual inspection of this simple table shows that it has two parts, but unfortunately the locations have slightly different names. Try this:

Input/Output ▼

```
SQL> SELECT *
2 FROM PARTS
3 WHERE LOCATION LIKE '%BACK%';
```

NAME	LOCATION	PARTNUMBER
SPINE	BACK	4
KIDNEY	MID-BACK	6

2 rows selected.

## Analysis ▼

You can see the use of the percent sign (%) in the statement after LIKE. When used inside a LIKE expression, % is a wildcard. What you asked for was any occurrence of BACK in the column location. If you queried

## Input ▼

```
SQL> SELECT *
      2 FROM PARTS
      3 WHERE LOCATION LIKE 'BACK%';
```

you would get any occurrence that started with BACK:

## Input/Output ▼

NAME	LOCATION	PARTNUMBER
SPINE	BACK	4

1 rows selected.

```
mysql> select * from parts where location like 'BACK%';
```

```
+-----+-----+-----+
| name | location | partnumber |
+-----+-----+-----+
| SPINE | BACK    | 4          |
+-----+-----+-----+
```

1 row in set (0.00 sec)

If you queried

## Input ▼

```
SQL> SELECT *
      2 FROM PARTS
      3 WHERE NAME LIKE 'A%';
```

you would get any name that starts with A:



**Output ▼**

NAME	LOCATION	PARTNUMBER
APPENDIX	MID-STOMACH	1
ADAMS APPLE	THROAT	2
ANVIL	EAR	5

3 rows selected.

Is LIKE case sensitive in *both* Oracle and MySQL? Try the next query to find out.

**Input/Output ▼**

```
SQL> SELECT *
      2 FROM PARTS
      3 WHERE NAME LIKE 'a%';
```

no rows selected

```
mysql> select * from parts where name like 'a%';
```

name	location	partnumber
APPENDIX	MID-STOMACH	1
ADAMS APPLE	THROAT	2
ANVIL	EAR	5

3 rows in set (0.00 sec)

The answer is yes in Oracle and no in MySQL. References to data are dependent upon the implementation you are working with.

What if you want to find data that matches all but one character in a certain pattern? In this case you could use a different type of wildcard: the underscore.

**Underscore (\_)**

The underscore is the single-character wildcard. Using a modified version of the table FRIENDS, type this:

**Input/Output ▼**

```
SQL> SELECT * FROM FRIENDS;
LASTNAME      FIRSTNAME      AREACODE  PHONE      ST ZIP
-----
BUNDY         AL             100 555-1111 IL 22333
MEZA          AL             200 555-2222 UK
MERRICK       BUD            300 555-6666 CO 80212
MAST          JD             381 555-6767 LA 23456
```

```
BULHER          FERRIS          345 555-3223 IL 23332
PERKINS         ALTON           911 555-3116 CA 95633
BOSS            SIR             204 555-2345 CT 95633
7 rows selected.
```

To find all the records where ST starts with C, type the following:

Input/Output ▼

---

```
SQL> SELECT *
      2 FROM FRIENDS
      3 WHERE ST LIKE 'C_';
LASTNAME      FIRSTNAME      AREACODE PHONE    ST ZIP
-----
MERRICK       BUD            300 555-6666 CO 80212
PERKINS       ALTON          911 555-3116 CA 95633
BOSS          SIR            204 555-2345 CT 95633
3 rows selected.
```

```
mysql> select * from friends where st like 'C_';
+-----+-----+-----+-----+-----+-----+
| lastname | firstname | areacode | phone   | st | zip |
+-----+-----+-----+-----+-----+-----+
| MERRICK  | BUD       | 300      | 555-6666 | CO | 80212 |
| PERKINS  | ALTON     | 911      | 555-3116 | CA | 95633 |
| BOSS     | SIR       | 204      | 555-2345 | CT | 95633 |
+-----+-----+-----+-----+-----+-----+
3 row in set (0.00 sec)
```

You can use several underscores in a statement:

Input/Output ▼

---

```
SQL> SELECT *
      2 FROM FRIENDS
      3 WHERE PHONE LIKE '555-6_ _';
LASTNAME      FIRSTNAME      AREACODE PHONE    ST ZIP
-----
MERRICK       BUD            300 555-6666 CO 80212
MAST          JD             381 555-6767 LA 23456
2 rows selected.
```

The previous statement could also be written as follows:

Input/Output ▼

---

```
SQL> SELECT *
      2 FROM FRIENDS
      3 WHERE PHONE LIKE '555-6%';
```

LASTNAME	FIRSTNAME	AREACODE	PHONE	ST	ZIP
MERRICK	BUD	300	555-6666	CO	80212
MAST	JD	381	555-6767	LA	23456

2 rows selected.

Notice that the results are identical. These two wildcards can be combined. The next example finds all records with L as the second character:

### Input/Output ▼

```
SQL> SELECT *
      2 FROM FRIENDS
      3 WHERE FIRSTNAME LIKE '_L%';
```

LASTNAME	FIRSTNAME	AREACODE	PHONE	ST	ZIP
BUNDY	AL	100	555-1111	IL	22333
MEZA	AL	200	555-2222	UK	
PERKINS	ALTON	911	555-3116	CA	95633

3 rows selected.

### Concatenation ( || )

The || (double pipe) symbol concatenates two strings. Try this:

### Input/Output ▼

```
SQL> SELECT FIRSTNAME || LASTNAME ENTIRENAME
      2 FROM FRIENDS;
```

ENTIRENAME
AL BUNDY
AL MEZA
BUD MERRICK
JD MAST
FERRIS BULHER
ALTON PERKINS
SIR BOSS

7 rows selected.

### Analysis ▼

Notice that || is used instead of +. If you use + to try to concatenate the strings, the SQL interpreter used for this example (Oracle) returns the following error:

## Input/Output ▼

```
SQL> SELECT FIRSTNAME + LASTNAME ENTIRENAME
      2 FROM FRIENDS;
```

```
ERROR:
ORA-01722: invalid number
```

It is looking for two numbers to add and throws the error invalid number when it doesn't find any.

### NOTE

Some implementations of SQL, such as Microsoft SQL Server, use the plus sign to concatenate strings. Check your implementation.

### NOTE

MySQL can be set up to allow the `||` for concatenation; however, this is not the default when MySQL is installed. `concat()` is the default. Any number of variables may be passed to the function `concat()`, and it is quite easy to use. Should you desire to change the parameters in MySQL to allow the use of the `||` for concatenation, first please research the subject in the documentation provided with MySQL.

## Input/Output ▼

```
mysql> select concat(firstname," ",lastname)Entirename from friends;
+-----+
| Entirename |
+-----+
| AL BUNDY   |
| BUD MERRICK|
| JD MAST    |
| FERRIS BULHER|
| AL MEZA    |
| ALTON PERKINS|
| SIR BOSS   |
+-----+
7 rows in set (0.00 sec)
```

Here's a more practical example using concatenation:

## Input/Output ▼

```
SQL> SELECT LASTNAME || ' ' || FIRSTNAME NAME
       2 FROM FRIENDS;
```

NAME

```
-----
BUNDY      , AL
MEZA       , AL
MERRICK    , BUD
MAST       , JD
BULHER     , FERRIS
PERKINS    , ALTON
BOSS       , SIR
7 rows selected.
```

```
mysql> select concat(lastname," "," ",firstname)Name from friends;
```

```
+-----+
| Name |
+-----+
| BUNDY, AL |
| MEZA, AL |
| MERRICK, BUD |
| MAST, JD |
| BULHER, FERRIS |
| PERKINS, ALTON |
| BOSS, SIR |
+-----+
```

```
7 rows in set (0.00 sec)
```

The Oracle statement inserted a comma between the last name and the first name. This was done because Oracle (and other implementations) accounts for the entire length that a column may be when it concatenates to the other string. This creates a natural spacing between the values of the columns/strings. The MySQL statement inserted a comma and a space between the two columns. MySQL automatically runs the values of the columns/strings into one; thus, any “natural” spacing between the values is lost.

### NOTE

More on this space issue: Notice the extra spaces between the first name and the last name in the Oracle examples. These spaces are actually part of the data. With certain data types, spaces are right-padded to values less than the total length allocated for a field. See your implementation. Data types will be discussed in Lesson 9. Additionally, if you try to concatenate a NULL value to a string, the result will be a NULL value for the entire expression. In these instances, you would possibly want to use a built-in function to remove the NULL values. This will be discussed in Lesson 7.

So far you have performed the comparisons one at a time. This method is fine for some problems, but what if you need to find all the people at work with last names starting with P who have less than three days of vacation time? Logical operators can help in this case.

Logical Operators

Logical operators separate two or more conditions in the WHERE clause of a SQL statement.

Vacation time is always a hot topic around the workplace. Say you designed a table called VACATION for the accounting department:

Input/Output ▼

```
SQL> SELECT * FROM VACATION;
```

LASTNAME	EMPLOYEEENUM	YEARS	LEAVETAKEN
ABLE	101	2	4
BAKER	104	5	23
BLEDSON	107	8	45
BOLIVAR	233	4	80
BOLD	210	15	100
COSTALES	211	10	78

6 rows selected.

Suppose your company gives each employee 12 days of leave each year. Using what you have learned and a logical operator, find all the employees whose name starts with B and who have more than 50 days of leave coming.

Input/Output ▼

```
SQL> SELECT LASTNAME,
2 YEARS * 12 - LEAVETAKEN REMAINING
3 FROM VACATION
4 WHERE LASTNAME LIKE 'B%'
5 AND
6 YEARS * 12 - LEAVETAKEN > 50;
```

LASTNAME	REMAINING
BLEDSON	51
BOLD	80

2 rows selected.

```
mysql> select lastname,
```

```

-> years*12 - leavetaken remaining
-> from vacation
-> where lastname like 'B%'
-> and years*12 - leavetaken > 50;
+-----+-----+
| lastname | remaining |
+-----+-----+
| BLEDSOE  |         51 |
| BOLD     |         80 |
+-----+-----+
2 rows in set (0.00 sec)

```

## Analysis ▼

This query is the most complicated you have done to date. The `SELECT` clause (lines 1 and 2) uses arithmetic operators to determine how many days of leave each employee has remaining. The normal precedence computes `YEARS * 12 - LEAVETAKEN`. (A clearer approach would be to write `(YEARS * 12) - LEAVETAKEN`.)

`LIKE` is used in line 4 with the wildcard `%` to find all the B names. Line 5 uses the `>` to find all occurrences greater than 50.

The new element is on line 5. You used the logical operator `AND` to ensure that you found records that met the criteria in lines 4 *and* 5.

## AND

`AND` requires that both expressions on either side be true to return `TRUE`. If either expression is false, `AND` returns `FALSE`. For example, to find out which employees have been with the company for 5 or fewer years and have taken more than 20 days leave, try this:

## Input/Output ▼

```

SQL> SELECT LASTNAME
      2 FROM VACATION
      3 WHERE YEARS <= 5
      4 AND
      5 LEAVETAKEN > 20 ;
LASTNAME
-----
BAKER
BOLIVAR
2 rows selected.

```

```

mysql> select lastname from vacation
-> where years <= 5
-> and leavetaken > 20;

```

```

+-----+
| lastname |
+-----+
| BAKER    |
| BOLIVAR  |
+-----+
2 rows in set (0.00 sec)

```

If you want to know which employees have been with the company for 5 years or more and have taken less than 50 percent of their leave, you could write:

### Input/Output ▼

---

```

SQL> SELECT LASTNAME WORKAHOLICS
      2 FROM VACATION
      3 WHERE YEARS >= 5
      4 AND
      5 ((YEARS *12)-LEAVETAKEN)/(YEARS * 12) < 0.50;
WORKAHOLICS
-----
BOLD
COSTALES

```

2 rows selected.

```

mysql> select lastname Workaholics
      -> from vacation
      -> where years >= 5
      -> and ((years * 12) - leavetaken) / (years * 12) < 0.50;
+-----+
| Workaholics |
+-----+
| BOLD        |
| COSTALES    |
+-----+
2 rows in set (0.00 sec)

```

Check these people for burnout. Also check out how we used the **AND** to combine these two conditions.

### OR

You can also use **OR** to sum up a series of conditions. If any of the comparisons are true, **OR** returns **TRUE**. To illustrate the difference, run the last query with **OR** instead of with **AND**:



## Input/Output ▼

```
SQL> SELECT LASTNAME WORKAHOLICS
      2 FROM VACATION
      3 WHERE YEARS >= 5
      4 OR
      5 ((YEARS *12)-LEAVETAKEN)/(YEARS * 12) < 0.50;
```

WORKAHOLICS

-----

BAKER  
BLEDSON  
BOLD  
COSTALES

4 rows selected.

```
mysql> select lastname
      -> from vacation
      -> where years >= 5
      -> OR ((years*12)-leavetaken)/(years*12) < 0.50;
```

```
+-----+
| lastname |
+-----+
| BAKER    |
| BLEDSON  |
| BOLD     |
| COSTALES |
+-----+
```

4 rows in set (0.00 sec)

The original names are still in the list, but you have three new entries (who would probably resent being called workaholics). These three new names made the list because they satisfied one of the conditions. OR requires only that one of the conditions be true for data to be returned.

## NOT

NOT means just that. If the condition it applies to evaluates to TRUE, NOT makes it FALSE. If the condition after the NOT is FALSE, it becomes TRUE. For example, the following SELECT returns the only two names not beginning with B in the table:

## Input/Output ▼

```
SQL> SELECT *
      2 FROM VACATION
      3 WHERE LASTNAME NOT LIKE 'B%';
```

LASTNAME	EMPLOYEEENUM	YEARS	LEAVETAKEN
ABLE	101	2	4
COSTALES	211	10	78

2 rows selected.

```
mysql> select * from vacation
-> where lastname not like 'B%';
+-----+-----+-----+-----+
| lastname | employeeenum | years | leavetaken |
+-----+-----+-----+-----+
| ABLE     | 101          | 2     | 4          |
| COSTALES | 211          | 10    | 78         |
+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

NOT can also be used with the operator IS when applied to NULL. Recall the PRICE table where we put a NULL value in the WHOLESALE column opposite the item ORANGES.

**Input/Output ▼**

---

```
SQL> SELECT * FROM PRICE;
```

ITEM	WHOLESALE
TOMATOES	.34
POTATOES	.51
BANANAS	.67
TURNIPS	.45
CHEESE	.89
APPLES	.23
ORANGES	

7 rows selected.

To find the non-NULL items, type this:

**Input/Output ▼**

---

```
SQL> SELECT *
2 FROM PRICE
3 WHERE WHOLESALE IS NOT NULL;
ITEM      WHOLESALE
-----
TOMATOES  .34
POTATOES  .51
BANANAS   .67
TURNIPS   .45
```

```
CHEESE          .89
APPLES         .23
6 rows selected.
```

## Set Operators

In Lesson 1, “Getting Started with SQL,” you learned that SQL is based on the theory of sets. The following sections examine set operators. *Set operators* are used to combine different sets of data returned by different queries into one query, and ultimately, one data set. There are various set operators available in SQL that allow you to combine different data sets to meet your data processing needs.

### UNION and UNION ALL

UNION returns the results of two queries minus the duplicate rows. The following two tables represent the rosters of teams:

#### Input/Output ▼

---

```
SQL> SELECT * FROM FOOTBALL;
```

```
NAME
-----
ABLE
BRAVO
CHARLIE
DECON
EXITOR
FUBAR
GOOBER
7 rows selected.
SQL> SELECT * FROM SOFTBALL;
```

```
NAME
-----
ABLE
BAKER
CHARLIE
DEAN
EXITOR
FALCONER
GOOBER
7 rows selected.
```

How many different people play on one team or another?

### Input/Output ▼

---

```
SQL> SELECT NAME FROM SOFTBALL
      2 UNION
      3 SELECT NAME FROM FOOTBALL;
NAME
-----
ABLE
BAKER
BRAVO
CHARLIE
DEAN
DECON
EXITOR
FALCONER
FUBAR
GOOBER
10 rows selected.
```

UNION returns 10 distinct names from the two lists. How many names are on both lists (including duplicates)?

### Input/Output ▼

---

```
SQL> SELECT NAME FROM SOFTBALL
      2 UNION ALL
      3 SELECT NAME FROM FOOTBALL;
NAME
-----
ABLE
BAKER
CHARLIE
DEAN
EXITOR
FALCONER
GOOBER
ABLE
BRAVO
CHARLIE
DECON
EXITOR
FUBAR
GOOBER
14 rows selected.
```

### Analysis ▼

---

The combined list—courtesy of the UNION ALL statement—has 14 names. UNION ALL works just like UNION except that it does not eliminate duplicates. You need to remember

that the UNION and UNION ALL statements will only work if all SELECT statements have the same columns. Otherwise, an error message will be returned. Now show me a list of players who are on both teams. You can't do that with UNION—you need to learn INTERSECT.

## INTERSECT

INTERSECT returns only the rows found by both queries. The next SELECT statement shows the list of players who play on both teams:

### Input/Output ▼

---

```
SQL> SELECT * FROM FOOTBALL
      2 INTERSECT
      3 SELECT * FROM SOFTBALL;
NAME
-----
ABLE
CHARLIE
EXITOR
GOOBER
```

4 rows selected.

In this example, INTERSECT finds the short list of players who are on both teams by combining the results of the two SELECT statements. INTERSECT has the same limitations as the UNION and UNION ALL statement, in as much as the SELECT statements that it is binding must contain the same columns.

## MINUS (Difference)

MINUS returns the rows from the first query that were not present in the second. For example:

### Input/Output ▼

---

```
SQL> SELECT * FROM FOOTBALL
      2 MINUS
      3 SELECT * FROM SOFTBALL;
NAME
-----
BRAVO
DECON
FUBAR
```

3 rows selected.

The preceding query shows the three football players who are not on the softball team. If you reverse the order, you get the three softball players who aren't on the football team:

**Input/Output ▼**

---

```
SQL> SELECT * FROM SOFTBALL
      2 MINUS
      3 SELECT * FROM FOOTBALL;
NAME
-----
BAKER
DEAN
FALCONER
```

3 rows selected.

**Miscellaneous Operators: IN and BETWEEN**

The two operators IN and BETWEEN provide a shorthand for functions you already know how to do. If you wanted to find friends in Colorado, California, and Louisiana, you could type the following:

**Input/Output ▼**

---

```
SQL> SELECT *
      2 FROM FRIENDS
      3 WHERE ST= 'CA'
      4 OR
      5 ST ='CO'
      6 OR
      7 ST = 'LA';
```

LASTNAME	FIRSTNAME	AREACODE	PHONE	ST	ZIP
-----	-----	-----	-----	-----	-----
MERRICK	BUD	300	555-6666	CO	80212
MAST	JD	381	555-6767	LA	23456
PERKINS	ALTON	911	555-3116	CA	95633

3 rows selected.

Or you could type this:

**Input/Output ▼**

---

```
SQL> SELECT *
      2 FROM FRIENDS
      3 WHERE ST IN('CA', 'CO', 'LA');
```

LASTNAME	FIRSTNAME	AREACODE	PHONE	ST	ZIP
MERRICK	BUD	300	555-6666	CO	80212
MAST	JD	381	555-6767	LA	23456
PERKINS	ALTON	911	555-3116	CA	95633

3 rows selected.

```
mysql> select * from friends
-> where st in ('CA','CO','LA');
```

lastname	firstname	areacode	phone	st	zip
MERRICK	BUD	300	555-6666	CO	80212
MAST	JD	381	555-6767	LA	23456
PERKINS	ALTON	911	555-3116	CA	95633

2 rows in set (0.20 sec)

The second example is shorter and more readable than the first. You never know when you might have to go back and work on something you wrote months ago. IN also works with numbers. Consider the following, where the column AREACODE is a number:

### Input/Output ▼

```
SQL> SELECT *
2 FROM FRIENDS
3 WHERE AREACODE IN(100,381,204);
```

LASTNAME	FIRSTNAME	AREACODE	PHONE	ST	ZIP
BUNDY	AL	100	555-1111	IL	22333
MAST	JD	381	555-6767	LA	23456
BOSS	SIR	204	555-2345	CT	95633

3 rows selected.

If you needed a range of data from the PRICE table, you could write the following:

### Input/Output ▼

```
SQL> SELECT *
2 FROM PRICE
3 WHERE WHOLESale > 0.25
4 AND
5 WHOLESale < 0.75;
```

ITEM	WHOLESale
TOMATOES	.34

```
POTATOES      .51
BANANAS       .67
TURNIPS       .45
4 rows selected.
```

Or using BETWEEN, you would write this:

## Input/Output ▼

```
SQL> SELECT *
      2 FROM PRICE
      3 WHERE WHOLESALE BETWEEN 0.25 AND 0.75;
ITEM          WHOLESALE
-----
TOMATOES      .34
POTATOES      .51
BANANAS       .67
TURNIPS       .45
4 rows selected.
```

```
mysql> select * from price
-> where wholesale between .25 and .75;
+-----+-----+
| item   | wholesale |
+-----+-----+
| TOMATOES | 0.34      |
| POTATOES | 0.51      |
| BANANAS  | 0.67      |
| TURNIPS  | 0.45      |
+-----+-----+
4 rows in set (0.08 sec)
```

Again, the second example is a cleaner, more readable solution than the first.

### NOTE

If a WHOLESALE value of 0.25 existed in the PRICE table, that record would have been retrieved also. Parameters used with BETWEEN are inclusive.

## Summary

At the beginning of this lesson, you knew how to use the basic SELECT and FROM clauses. Now you know how to use a host of operators that enable you to fine-tune your requests to the database. You learned how to use arithmetic, comparison, character, logical, and



set operators. This powerful set of tools provides the cornerstone of your SQL knowledge. In Lesson 4, you learn to increase the data-mining power of the SQL query by integrating other clauses such as the WHERE clause into your queries to perform operations involving grouping and ordering.

## Q&A

**Q How does all this information apply to me if I am not using SQL from the command line as depicted in the examples?**

**A** Whether you use SQL in COBOL as Embedded SQL or in Microsoft's Open Database Connectivity (ODBC), you use the same basic constructions. You will use what you learned in these first lessons repeatedly as you work with SQL.

**Q Why are you constantly telling me to check my implementation? I thought there was a standard!**

**A** There is an ANSI standard (the most recent version was released in late 2008); however, most vendors modify it somewhat to suit their databases. The basics are similar if not identical, and each instance has extensions that other vendors copy and improve. We have chosen to use ANSI as a starting point and to point out the differences as we go along.

3

## Workshop

The Workshop provides quiz questions to help solidify your understanding of the material covered, as well as exercises to provide you with experience in using what you have learned. Try to answer the quiz and exercise questions before checking the answers in Appendix A, "Answers."

Here are the CREATE TABLE statements and INSERT statements for the FRIENDS and PRICE tables. Type the following code into MySQL if you have not already done so.

```
create table friends
(lastname      varchar(15)      not null,
firstname     varchar(15)      not null,
areacode      numeric(3)       null,
phone         varchar(9)       null,
st            char(2)          not null,
zip           varchar(5)       not null);

insert into friends values
('BUNDY', 'AL', '100', '555-1111', 'IL', '22333');

insert into friends values
('MEZA', 'AL', '200', '555-2222', 'UK', NULL);
```

```

insert into friends values
('MERRICK', 'BUD', '300', '555-6666', 'CO', '80212');

insert into friends values
('MAST', 'JD', '381', '555-6767', 'LA', '23456');

insert into friends values
('BULHER', 'FERRIS', '345', '555-3223', 'IL', '23332');

insert into friends values
('PERKINS', 'ALTON', '911', '555-3116', 'CA', '95633');

insert into friends values
('BOSS', 'SIR', '204', '555-2345', 'CT', '95633');

create table price
(item          varchar(15)      not null,
 wholesale     decimal(4,2)    not null);

insert into price values
('TOMATOES', '.34');

insert into price values
('POTATOES', '.51');

insert into price values
('BANANAS', '.67');

insert into price values
('TURNIPS', '.45');

insert into price values
('CHEESE', '.89');

insert into price values
('APPLES', '.23');

```

## Quiz

Use the FRIENDS table to answer the following questions.

LASTNAME	FIRSTNAME	AREACODE	PHONE	ST	ZIP
-----	-----	-----	-----	---	-----
BUNDY	AL	100	555-1111	IL	22333
MEZA	AL	200	555-2222	UK	
MERRICK	BUD	300	555-6666	CO	80212
MAST	JD	381	555-6767	LA	23456
BULHER	FERRIS	345	555-3223	IL	23332
PERKINS	ALTON	911	555-3116	CA	95633
BOSS	SIR	204	555-2345	CT	95633

1. Write a query that returns everyone in the database whose last name begins with M.
2. Write a query that returns everyone who lives in Illinois with a first name of AL.
3. Given two tables (PART1 and PART2) containing columns named PARTNO, how would you find out which part numbers are in both tables? Write the query.
4. What shorthand could you use instead of `WHERE a >= 10 AND a <=30`?
5. What will this query return?  

```
SELECT FIRSTNAME
FROM FRIENDS
WHERE FIRSTNAME = 'AL'
    AND LASTNAME = 'BULHER';
```
6. What is the main difference in the result set when using UNION versus UNION ALL?
7. What is the primary difference between using INTERSECT and MINUS?

## Exercises

1. Using the FRIENDS table, write a query that returns the following:

NAME	ST
-----	-----
AL	FROM IL

2. Using the FRIENDS table, write a query that returns the following:

NAME	PHONE
-----	-----
MERRICK, BUD	300-555-6666
MAST, JD	381-555-6767
BULHER, FERRIS	345-555-3223

3. Select all columns from the PRICE table where the column WHOLESALE is greater than .50.
4. What results do you get from the following query?  

```
mysql> select *
-> from price
-> where item like '%ATO%';
```

5. Does MySQL support set operators such as UNION, UNION ALL, INTERSECT, and MINUS?

6. What is wrong with the following query?

```
SELECT FIRSTNAME, LASTNAME FROM FRIENDS_1
UNION
SELECT FIRSTNAME FROM FRIENDS_2;
```

# Index

## Symbols

**& (ampersand), 614**

SQL\*Plus variables, 611

**\* (asterisk)**

columns, 180

line numbers, 590

queries, writing, 26-27

**@ (at symbol), variables, 668**

**^ (caret) wildcard operator (T-SQL), 680**

**/ \* / (comments), 624**

**- (dash), numeric values, formatting, 541**

**{ } (curly brackets), 672**

**/ (division sign) arithmetic operator, 48-49**

**. (dot), table names, 248**

**-- (double dashes), 546**

**- - (double hyphens), comments, 624**

**|| (double pipe) concatenation character operator, 67-70, 338, 509**

**" "(double quotation marks) aliases, 50**

literal strings, 620

**= (equal sign), 540**

tables, joining, 137

**= (equal sign) comparison operator, 56-58**

**= (equal sign) relational operator, subqueries, 159**

**/ (forward slash)**

PL/SQL, 642

SQL\*Plus buffer, 590

table names, 248

**<or !=> (inequalities)**

comparison operator, 62-63

**> (greater than sign)**

comparison operator, 59

**>= (greater than or equal to sign) comparison operator, 59**

**< (less than sign)**

comparison operator, 60-61

**<= (less than or equal to sign) comparison operator, 60-61**

**% (modulo sign) arithmetic operator, 51-53**

**\* (multiplication sign)**

arithmetic operator, 49-51

**= operator, values**

comparing, 538

**() (parentheses), 43**

columns, 180

numeric values, formatting, 541

subqueries, 155

**% (percent sign), 64**

numeric values, increasing, 542

**% (percent sign) wildcard operator (T-SQL), 680**

**+ (plus sign) arithmetic operator, 43-46**

**+ (plus sign), joining tables, 143**

**# (pound sign)**

MySQL on UNIX, 687

tables, 468

**" (quotation marks), number-defined fields, 59**

**;(semicolon)**

commands, 691

queries, writing, 28

SQL statements, 516, 690

**' (single quotation marks)**

character data types, 290

NULL values, 290

SQL scripts, 511

**\_ (underscore) character operator, 65-67**

**\_ (underscore) wildcard operator (T-SQL), 679**

**3GLs (third-generation languages), 5**

**4GL (fourth-generation language), 12**

**12 rules (Codd's), relational databases, 6-10**

## A

**abbreviating SQL\*Plus commands, 589**

**abbreviations, time zones, 315**

**ABS function, 195**

**ACCEPT command, SQL\*Plus variables, 612-614**

access, users of Oracle data dictionary, 443-449

Access, import/export tools, 303-304

Access (Microsoft) relational database management system, 414

Accessing databases

from Java, 581

T-SQL, 665

ACCOUNT ID field (BILLS table), indexes (creating), 373-377

accounts

ISQL (InterBase SQL), creating, 577

Oracle8, creating, 577

accuracy of data entry, 268

ADD MONTHS/ADD DATE function, 188-190

adding

lines in code, 591

time to dates, 315-318

administering MySQL, 686

administrators, 401. *See also* DBA

database, security, 413-414

advanced reports, creating, 624-626

age (individual's), computing from date of birth, 532-533

aggregate functions

ANSI standard, 179

AVG, 182-183

COUNT, 180-181

data, summarizing, 180

MAX, 184

MIN, 185

SELECT statement, 349

STDDEV, 186-187

subqueries, 160-161

SUM, 181-182

VARIANCE, 186

aliases

columns, 133

in MySQL, 46

“ ” (double quotation marks), 50

queries, 46

tables, 133, 157, 391

ALL keyword, 33

subqueries, embedding, 169-174

ALL TAB PRIVS view, 447

ALL TABLES view, 445-446

ALL\_CATALOG view, 444

ALL\_USERS view, 441

ALTER ANY CLUSTER system privilege, 423

ALTER ANY DATABASE system privilege, 423

ALTER ANY INDEX system privilege, 423

ALTER ANY PROCEDURE system privilege, 423

ALTER ANY ROLE system privilege, 423

ALTER ANY SEQUENCE system privilege, 423

ALTER ANY SNAPSHOT system privilege, 423

ALTER ANY TABLE system privilege, 423

ALTER ANY TRIGGER system privilege, 423

ALTER ANY TYPE system privilege, 423

ALTER ANY USER system privilege, 423

ALTER SESSION command, 356

ALTER SYSTEM command, 356

ALTER TABLE command, 440

constraints (on data), 275-276

CREATE INDEX statement, 373

ALTER TABLE statement

CHANGE option, syntax, 259

constraints, 279

primary keys, 273

syntax, 258

table structures, modifying, 257-261

ALTER TYPE statement, syntax, 493-494

ALTER USER command, 418

American National Standards Organization. *See* ANSI

AMOUNT field (BILLS table), indexes, creating, 377-378

ampersand (&), 614

SQL\*Plus variables, 611

AND logical operator, 71-72

anonymous users (MySQL), 688

ANSI (American National Standards Organization), 6

SQL extensions, 662-663

SQL3 standard, 6, 18

standards, 86

ANSI SQL standard, 1

ANSI standard

aggregate functions, 179

CAST operator, data types, converting, 321

data types, dates and time, 310-311

DATE data type, 310

TIME data type, 310-311

TIMESTAMP data type, 310-311

ANSO SQL3 syntax

ALTER TYPE statement, 493-494

ALTER TYPETYPE statement, 494

CREATE TYPE statement, 493-494

DROP statement, 493-494

**ANY** keyword, embedding subqueries, 169-174

**APIs (Application Programming Interfaces)**, 16

**APPEND** command, 592

**APPEND** text command, 588

**application programming**

ISQL (InterBase SQL), 577-580

Java, 581-583

JDBC, 581

.NET, 583-584

Oracle8, 577-580

SQL, embedding, 17-18, 575

**Application Programming Interfaces (APIs)**, 16

**applications**

APIs, 16

banking, 355-356

client/server development, 13

development tools, 575-576

**architectures, ODBC (Open Database Connectivity)**, 16

**arithmetic operations, functions**, 195

ABS, 195

CEIL, 196

EXP, 196-197

FLOOR, 196

LN, 197-198

LOG, 197-198

MOD, 198

POWER, 199

SIGN, 199-200

SQRT, 200-201

**arithmetic operators**, 42-43

% (modulo sign), 51-53

\* (multiplication sign), 49-51

+ (plus sign), 43-46

- (minus sign), 46-48

/ (division sign), 48-49

precedence, 53-54

**arranging query elements**, 393-395

**arrows, DOWN ARROW** key (cursors), 472

**articles, "Relational Model of Data for Large Shared Data Banks (A),"** 6

**Artists Cursor** result set, cursors (scrolling), 474

**ARTISTS** table, 469

cursors, creating, 473

**assigning**

constants of DECLARE section (PL/SQL blocks), 633

variables of DECLARE section (PL/SQL blocks), 632-633

**asterisk (\*)**

columns, 180

line numbers, 590

queries, writing, 26-27

**attributes**

%ROWCOUNT of DECLARE section (PL/SQL blocks), 634-635

%ROWTYPE of DECLARE section (PL/SQL blocks), 634

%TYPE of DECLARE section (PL/SQL blocks), 633-634

UDTs (User Defined Types), 495

**%attributes"NOTFOUND,"** 639

**AUTOCOMMIT** option (SET TRANSACTION statement), 358

**AVG** function, 182-183

groups, 608

subqueries, embedding, 160-161

**AVG(ANNUALLEAVE)** clause, 113-115

**AVG(SALARY)** clause, 113-115

**AVG(SICKLEAVE)** clause, 113-115

## B

**B-tree** index, 371

**backing up** tables, 523

**BALANCES** table, transaction control, 355

**BANK ACCOUNTS** table, data, 254, 334

**banking applications**

BALANCES table, 355

CUSTOMERS table, 354

data, 355

transaction control, 354-356

transactions

beginning, 356-359

canceled, 361-364

finishing, 359-361

savepoints, 364-366

**base** tables, 135

**BASEBALL** database (T-SQL), 665-668

**batch loads and OLTP (online transactional processing)**, comparing, 398-400

**batch mode**, MySQL terminal monito, 692-693

**batch transactions, COMMIT** statement, 401-402

**BATTERS** table, 666

**BCP (bulk copy)** tool, 304

**BEGIN** statement

of PROCEDURE section (PL/SQL blocks), 635

program flow control, 672

**BEGIN TRANSACTION** command, 357

**BEGIN TRANSACTION statement, 480****beginning transactions, 356-359**

syntax, 356

**benefits of databases, normalizing, 236-237****BETWEEN operator, 78-80****BILLS table**ACCOUNT ID field,  
indexes, creating,  
373-377AMOUNT field, creating  
indexes, 377-378

data, 253-254, 333

data breakdown, 247

**binary data types (T-SQL), 664****binary distribution file, installing MySQL, 686-687****binary strings data type (T-SQL), 664****bit data types (T-SQL), 665****BLOB data types, 250, 498****block structure (PL/SQL), 630-631**DECLARE section,  
631-635EXCEPTION section,  
631, 640blocks, executing,  
642-643comments,  
inserting, 642exceptions, handling,  
641-642

exceptions, raising, 641

output to users,  
displaying, 643-644script files,  
executing, 643PROCEDURE section, 631,  
635

BEGIN statement, 635

CLOSE command, 637

conditional statements,  
637-640cursor control  
commands, 635-637DECLARE  
command, 636

END statement, 635

FETCH command,  
636-637

FOR-LOOP, 640

IF, THEN  
statement, 638

LOOP, 639

loops, 638-640

OPEN command, 636

WHILE-LOOP,  
639-640**blocks, PL/SQL, 646-652**

executing, 642-643

starting, 642

**Boar, Bernard H., 12****brackets, curly ({}), 672****brackets ([ ]) wildcard  
operator (T-SQL), 680****BREAK command, program  
flow control, 677****BREAK ON command,  
creating report and group  
summaries, 607-608****BTITLE command, formatting  
reports, 604****buffers, SQL\*Plus, 588,  
591-593**

APPEND command, 592

\* (asterisk), 590

CHANGE command,  
syntax, 590CLEAR BUFFER  
command, 592CLEAR command, 592  
commands, 588

abbreviating, 589

case sensitivity, 589

contents, clearing, 592

/ (forward slash), 590

INPUT command, 591

line numbers, 589-590

lines, 591

LIST command, 588

SQL statement, 588

**building**indexes, rebuilding,  
402-404SQL queries (complex),  
546-547

subqueries, 153-160

tables, rebuilding, 402-404

**built-in database tuning  
tools, 409****built-in functions, 179****bulk copy (BCP) tool, 304****BY password option,  
Personal Oracle database  
security, 416****BYTES, 449****bytes, converting to kilobytes  
and megabytes, 536****C****C functions, Static SQL,  
483-484****call-level interfaces, 18****canceling transactions,  
361-364****capitalization, queries, 22****caret (^) wildcard operator  
(T-SQL), 680****Cartesian product, 567-568**cross joining tables,  
123-128**CASCADE option,  
Personal Oracle database  
security, 419****case, sorting, 61****CASE (computer-aided  
software engineering)  
tools, 245, 439****case sensitivity**

commands, 22

data, storing, 58

keywords, 665



- LIKE operator, 65
- MySQL commands, 691
- SQL\*Plus commands, 589
- CAST operator, datatypes, converting, 321**
- CEIL function, 196**
- CHANGE command, syntax, 590**
- CHANGE option (ALTER TABLE statement), syntax, 259**
- CHANGE/old value/new value command, 588**
- changing users for Personal Oracle, 418**
- @char convert variable, 668**
- char data type (T-SQL), 663**
- CHAR(size) data type, 249**
- character data types, ' ' (single quotation marks), 290**
- character operators**
  - || (double pipe) concatenation, 67-70
  - LIKE, 63-65
  - \_ (underscore), 65-67
- character strings**
  - converting to dates, 326
  - data types (T-SQL), 663
  - dates, converting to, 325
- characters**
  - concatenation (||), 509
  - functions, 201
    - CHR, 201-202
    - CONCAT, 202-203
    - INITCAP, 203
    - INSTR, 214
    - LENGTH, 215
    - LOWER, 203-205
    - LPAD, 205-206
    - LTRIM, 206-207
    - REPLACE, 207-209
    - RPAD, 205-206
    - RTRIM, 206-207
    - SUBSTR, 209-213
    - SUBSTR/MID, 213
    - TRANSLATE, 213-214
    - UPPER, 203-205
  - functions (MySQL), 219
    - INSTR, 220
    - LEFT, 221
    - LENGTH, 219
    - LOCATE, 219-220
    - LPAD, 220
    - LTRIM, 222
    - RIGHT, 221
    - RPAD, 220
    - RTRIM, 222
    - SUBSTRING, 221
    - TRIM, 222
  - table names, limit of, 248
- check constraints (on data), 276-277**
- CHECKS table, 25**
- child**
  - parent/child relationships, deleting records, 279
  - parent/child table relationships, 275-276
  - tables, Oracle SQL\*Plus referential integrity reports, 280-281
- CHR function, 201-202**
- clauses**
  - AVG(ANNUALLEAVE), 113-115
  - AVG(SALARY), 113-115
  - AVG(SICKLEAVE), 113-115
  - DELETE, 299
  - DISTINCT, views, creating, 345
  - FROM keyword, 87
  - GROUP BY, 98-105, 116-117
    - SELECT statement, 343
    - subqueries, correlated, 168-169
  - HAVING, 105-111, 116-117
    - subqueries, correlated, 168-169
  - IDENTIFIED BY, 417
  - MODIFY, 261
  - NAME, 113
  - ORDER BY, 89-98, 114-116
    - indexes, 377
    - SELECT statement, 343
  - PAYEE, 112
    - in queries, 85
      - AVG(ANNUAL LEAVE), 113-115
      - AVG(SALARY), 113-115
      - AVG(SICKLEAVE), 113-115
    - combining, 112
    - errors, 109
    - GROUP BY, 98-105, 116-117
    - HAVING, 105-111, 116-117
    - NAME, 113
    - ORDER BY, 89-98, 114-116
    - PAYEE, 112
    - REMARKS, 112
    - SELECT, 100
      - SELECT statement, syntax, 85-86
      - syntax, 86
      - TEAM, 113-115
      - WHERE, 87-88, 115-116
    - REMARKS, 112
    - SELECT, 100
    - SELECT keyword, 87
    - storage in CREATE TABLE statement, 254-255
    - TEAM, 113-115

- WHERE, 87-88, 115-116
  - DELETE
    - statement, 300
  - most restrictive condition, 393-395
  - SQL statements, 390, 393
  - syntax, 41-42
  - tables, joining, 126
  - UPDATE
    - statement, 295
  - WITH GRANT OPTION, Personal Oracle security, 433-434
- CLEAR BUFFER**
  - command, 592
- CLEAR command, 592, 609**
  - SQL\*Plus settings, deleting, 603
- @@client csid variable, 668**
- @@client curname**
  - variable, 668
- client/server application development, 13**
- client/server computing, 12**
- client/server database systems, 11**
- client/server development, 11**
- CLOB data type, 250**
- CLOSE command of PROCEDURE section (PL/SQL blocks), 637**
- closing**
  - cursors, 475
  - database cursors, 472
- clustered indexes, 373, 384-385**
- Codd, E.F.**
  - RDBMS, 8
  - relational databases, 12
  - rules, 6
- code**
  - case, sorting, 61
  - CREATE INDEX
    - statement, 372
  - CREATE TABLE statement
    - example, 248
  - lines, 591
  - mailing list tables, 403-404
  - ORDERS table, creating, 152-153
  - PART table, creating, 152-153
  - SQL statements, output, 26-27
  - table lists, viewing, 256-257
  - tables
    - creating, 731-738
    - populating, 743-755
- COLUMN command, 614**
  - columns, formatting, 605-606
  - dates, 620
- column name = alias**
  - syntax, 46
- columns**
  - aliases, 133
    - "" (double quotation marks), 50
    - in MySQL, 46
  - \* (asterisk), 180
  - changing from NOT NULL to NULL, syntax, 259
  - changing from NULL to NOT NULL, 260-261
  - composite indexes, 381
  - finding, 128-129
  - foreign keys, constraints (on data), 274
  - formatting, 605-606
  - indexes, 378
  - individual, selecting, 28
  - invalid column names, errors, 555-556
  - lengths, increasing or decreasing, 258
  - names, 336
    - shortening, 60
  - NOT NULL keywords, 250
  - null, inserting spaces, 289
  - NULL values, indexes, 379
  - numeric values, finding highest, 542-544
  - order, changing, 29-32
  - ordering, 97
  - ( ) (parentheses), 180
  - pseudocolumn, SYSDAYE function, 314
  - relational databases, 8
  - renaming, 45, 337-338
  - selecting and placing, 336
  - virtual, updating, 345
- combining**
  - clauses, 112
  - SELECT and CREATE VIEW statements, 341
- command-line history, MySQL terminal monitor, 692**
- command-line options, MySQL terminal monitor, 689-690**
- command-line SQL, 14**
- comments, 624**
  - inserting, 642
- COMMIT command, 294**
  - PL/SQL transactional control, 644
  - transactions, 359-360
- COMMIT statement**
  - batch transactions, 401-402
  - transactions, 359-360
    - canceling, 363
- COMMIT WORK command, transactions, 360**
- commits, transactions, 364**
- COMPANY table, data, 254, 334**
- comparing**
  - batch loads and OLTP, 398-400
  - dates and time periods, 320

- OLAP and OLTP databases, 397
- OUTER JOINS and INNER JOINS, 139-143
- PL/SQL and Java stored procedures, 657
- values, 538
- comparison operators, 55-56, 63**
  - = (equal sign), 56-58
  - FALSE value, 55
  - > (greater than sign), 59
  - >= (greater than or equal to sign), 59
  - < or != (inequalities), 62-63
  - < (less than sign), 60-61
  - <= (less than or equal to sign), 60-61
  - TRUE value, 55
  - UNKNOWN value, 55
- “comparison operators”NULL, 55-57
- complex queries, 529. See also queries (complex)**
  - simplifying with views, 347-348
- components, T-SQL, 662-663**
- composite indexes, 379-381, 393**
- COMPUTE command, creating report and group summaries, 608-610**
- computer-aided software engineering (CASE) tools, 245, 439**
- computing**
  - client/server, 12
  - individual’s age from date of birth, 532-533
- CONCAT function, 202-203**
- concatenation, || (double pipe) character operator, 67-70**
- concatenation (||) character, 509**
- concepts of queries, applying, 25-26**
- conditional statements**
  - FOR-LOOP, 640
  - IF, THEN, 638
  - LOOP, 639
  - loops, 638-640
  - of PROCEDURE section (PL/SQL blocks), 637-640
  - WHILE-LOOP, 639-640
- conditions**
  - most restrictive (WHERE clause), 393-395
  - queries, 40-41
    - WHERE clause, syntax, 41-42
- conflicts of table names, 635**
- CONNECT role, 442**
- Connect role, creating for Personal Oracle, 420**
- connecting to databases with MySQL terminal monitor, 689**
- connections**
  - ODBC (Open Database Connectivity), 16-17
    - creating, 580
  - SQL Server, logging off, 471
- @connections variable, 669**
- constants of DECLARE section (PL/SQL blocks), assigning, 633**
- constraints**
  - ALTER TABLE statement, 279
  - CREATE TABLE statement, 279
  - scripts, maintaining, 279
  - table, disabling, 516-517
- constraints (on data)**
  - ALTER TABLE command, 275-276
  - check, 276-277
  - correct order of, 278-279
  - creating, 279-280
  - data integrity, 267-268
  - definition, 267
  - foreign key, 274-275
  - managing, 278
  - NOT NULL, 269-271
  - Oracle SQL\*Plus referential integrity reports, 280-283
  - primary key, 271-273
  - types of, 269
  - unique, 273-274
  - usefulness of, 268
- contents, data dictionary, 439**
- CONTINUE command, program flow control, 677-678**
- control, 354, 672. See also program flow control; transaction control**
- controlling**
  - data integrity, 267
  - transactions, 353
- conventions, 234. See also naming conventions**
- conversion functions, 215**
  - TO CHAR, 215-217
  - TO NUMBER, 217
- conversions**
  - DATE, 619-623
  - dates (T-SQL), 680-681
- CONVERT command, 680**
- converting**
  - bytes to kilobytes and megabytes, 536
  - character strings to dates, 326
  - data types with CAST operator, 321
  - date formats, 321-326
  - units with views, 346-347

correlated subqueries,  
166-169

**COUNT** function, 180-181

groups, 608

subqueries, embedding,  
160-161

**COUNT(\*)** function, 507

counting table rows,  
507-511

@@cpu busy variable, 669

**CREATE ANY INDEX** system  
privilege, 423

**CREATE ANY PROCEDURE**  
system privilege, 423

**CREATE ANY TABLE** system  
privilege, 423

**CREATE ANY TRIGGER**  
system privilege, 423

**CREATE ANY VIEW** system  
privilege, 423

**CREATE DATABASE**  
statement, 242

data dictionaries, creating,  
244-245

data, breaking down, 247

database design, 244

key fields, creating,  
246-247

options, 243

syntax, 242

**CREATE INDEX** statement,  
369, 373-375

ALTER TABLE command,  
373

code, 372

MySQL database, 373

UNIQUE keyword,  
381-382

**CREATE** keyword, 577

**CREATE PROCEDURE** system  
privilege, 423

**CREATE PROFILE** system  
privilege, 423

**CREATE PUBLIC SYNONYM**  
privilege, 517

**CREATE ROLE** statement,  
488-490

**CREATE ROLE** system  
privilege, 423

**CREATE SESSION** privilege,  
441-442

**CREATE SESSION** system  
privilege, 424

**CREATE** statements,  
syntax, 241

**CREATE SYNONYM** system  
privilege, 423

**CREATE TABLE** command,  
255, 335

**CREATE TABLE** statement,  
247-248

code example, 248

constraints, 279

examples, 737

fields

data types, 249-250

names, 249

NULL value, 250-252

unique, 252-254

storage clause, 254-255

tables

creating, 255-257

names, 248-249

sizing, 254-255

storing, 254-255

**create table** statements,  
269, 529-532, 731-738,  
740-742

NOT NULL  
constraints, 271

**CREATE TABLE** system  
privilege, 423

**CREATE TRIGGER** statement,  
syntax, 491

**CREATE TRIGGER** system  
privilege, 424

**CREATE TYPE** statement,  
492-496

object-orientation, 492

syntax, 493-494

UDTs, creating, 494-495

**CREATE USER** system  
privilege, 424

**CREATE VIEW**  
statement, 332

columns, selecting and  
placing, 336

and SELECT statement,  
combining, 341

syntax, 332

**CREATE VIEW** system  
privilege, 424

**CREATE VIEW:SELECT**  
statement, 343

cross joining tables  
(Cartesian product),  
123-128

cross joins, 126

cross-product language  
(SQL), 12-13

.CTL file extension, 305

Ctrl+D keyboard  
shortcuts, 689

**CURDATE** function, 225

curly brackets ({}), 672

current dates for queries,  
313-314

**cursors**, 472

closing, 475

control commands of  
PROCEDURE section  
(PL/SQL blocks),  
635-637

creating

based on ARTISTS  
table, 473

with Oracle SQL  
syntax, 473

in sessions, 476

in stored  
procedures, 476

with T-SQL syntax, 473

in triggers, 476

databases

creating, using,  
closing, 472

records, examining, 472

- DEALLOCATE
    - command, 475
  - DECLARE cursor\_name
    - CURSOR statement, 473
  - of DECLARE section
    - (PL/SQL blocks), 633
  - DOWN ARROW key, 472
  - memory, 476
  - %NOTFOUND
    - attribute, 639
  - opening, 473
  - query results, saving, 472
  - result sets, creating, 472
  - scope, 475-476
  - scrolling
    - through Artists Cursor
      - result set, 474
    - through result sets,
      - 473-474
  - status, testing, 475
  - tables, scrolling with
    - WHILE loop, 678-679
  - testing status of, 474
  - CUSTOMERS table,**
    - transaction control, 354**
  - customizing SQL\*Plus work environment, 599-603**
- D**
- daemons, mysqld, 687-688**
  - dash (-), formatting numeric values, 541**
  - dashes, double (--), 546**
  - data**
    - BANK ACCOUNTS table,
      - 254, 334
    - BILLS table, 253-254, 333
      - data breakdown, 247
    - breaking down, 247
    - COMMIT command, 294
    - COMPANY table, 254, 334
    - constraints
      - ALTER TABLE
        - command, 275-276
      - check, 276-277
      - correct order of,
        - 278-279
      - creating, 279-280
      - data integrity, 267-268
      - definition, 267
      - foreign key, 274-276
      - managing, 278
      - NOT NULL, 269-271
      - Oracle SQL\*Plus
        - referential integrity
          - reports, 280-283
      - parent/child table
        - relationships, 275-276
      - primary key, 271-273
      - types of, 269
      - unique, 273-274
      - usefulness of, 268
    - data-manipulation
      - statements, 285-286
    - DELETE statement, 285,
      - 298-302
    - dictionaries
      - CASE tools, 245
      - RDBMS packages, 245
    - entering
      - accurately, 268
      - into tables, 577
    - exporting from foreign
      - sources, 303-305
    - filtering for views, 336
    - importing from foreign
      - sources, 303-305
    - INSERT statements,
      - 285-286
      - examples, 36, 118
      - NULL values, inserting,
        - 289-290
      - records, entering with
        - INSERT, SELECT
          - statement, 292-294
        - records, entering with
          - INSERT, VALUES
            - statement, 286-289
            - unique values, inserting,
              - 291-292
      - integrity, controlling, 267
      - loading, disabling table
        - constraints, 516-517
      - manipulating, 285
      - merging, 233
      - Microsoft Access
        - import/export tools,
          - 303-304
      - Microsoft SQL Server
        - import/export tools, 304
      - modifying in views,
        - 343-345
      - MySQL import/export
        - tools, 305
      - Personal Oracle
        - import/export tools, 305
      - PL/SQL tables, 645
      - preventing problems
        - with, 571
      - querying with views, 337
      - redundancy, 231
        - normalization, 229
      - retrieving
        - from banking
          - applications, 355
        - indexes, 378
        - into local variables, 671
      - storing
        - case sensitivity, 58
        - using variables, 670
      - summarizing
        - aggregate
          - functions, 180
        - AVG function, 182-183
        - COUNT function,
          - 180-181
        - MAX function, 184
        - MIN function, 185
        - STDDEV function,
          - 186-187

- SUM function, 181-182
  - from tables, 349
- VARIANCE
  - function, 186
- Sybase SQL Server
  - import/export tools, 304
- UPDATE statement, 285, 295-298
  - updating from banking applications, 355
- Data Definition Language (DDL), 437**
- data definition statements, 241**
- data dictionaries, 437-438**
  - contents, 439
  - creating, 244-245
  - DBAs, 439
  - MySQL, 440
  - Oracle, 439-440
    - DBA views, 449-450
    - DBA, dynamic performance views, 458-461
    - DBA, growth, 456-457
    - DBA, objects, 452-456
    - DBA, security, 451-452
    - DBA, space allocation, 457-458
    - DBA, users, 450-451
    - INFORMATION\_SCHEMA, 461-463
    - MySQL table commands, 460-461
    - sessions, 458-459
    - user access, 443-449
    - user privileges, 442-443
    - user views, 440
    - users, identifying, 441-442
  - system engineers, 439
  - tables, 437-438
  - users, identifying, 438-439
- Data Dictionary Language. See DDL**
- data integrity, constraints, 267-268**
- data loads, indexes, dropping, 400-401**
- Data Manipulation Language. See DML**
- data manipulation statements, 241**
- data retrieval, 23-24**
- Data Sources (ODBC), 580**
- data types**
  - ANSI standard, dates and time, 310-311
  - BLOB, 250, 498
  - CHAR(size), 249
  - CLOB, 250
  - converting CAST operator, 321
  - DATE, 249, 310-312
  - DATETIME, 309-312
  - of fields, 249-250
  - implementing, 312
  - INTEGER, 250
  - INTEGER(n), 250
  - LONG, 250
  - LONG RAW, 250
  - LONG RAWBinary, 249
  - LONG VARCHAR, 250
  - LONGVARIABLE, 249
  - MLSLABEL, 250
  - NCLOB, 250
  - NUMBER, 249
  - Personal Oracle support, 250
  - Personal Oracle8 support, 249-250
  - RAW MLSLABEL, 249
  - RAW(size), 249
  - ROWID, 250
  - SMALLDATETIME, 312
  - SMALLINT, 249
  - TIME, 310-312
  - TIMESTAMP, 310-312
  - VARCHAR, 250
  - VARCHAR2(size), 250
  - YEAR, 312
- data types (T-SQL), 663-664**
- database access (T-SQL), 665**
  - BASEBALL database, 665-668
  - PRINT command, 671-672
  - variables, 671
    - data storage, 670
  - DECLARE
    - keyword, 670
  - global, declaring, 668-669
  - local, declaring, 668
  - local, retrieving data, 671
- database administrators. See DBAs**
- database management system. See DBMS**
- database manipulation language (DML) commands, 490**
- database security, 413, 415**
  - database administrators, 413-414
  - Microsoft Access relational database management system, 414
  - Microsoft FoxPro database management system, 414
  - MySQL server, 415
  - Oracle relational database management system, 414
  - Personal Oracle. *See* Personal Oracle
  - products, 414-415
  - Sybase SQL Server, 415

**databases**

- accessing from Java, 581
- BASEBALL (T-SQL), 665-668
- CASE (computer-aided software engineering) tools, 245
- client/server database systems, 11
- Codd, E.F. (12 rules), 6-10
- connecting to with MySQL terminal monitor, 689
- CREATE DATABASE statement, 242-247
- creating, 577-580
- current technologies, 11-12
- cursors, 472
- dates of birth, storing, 320
- DBA, 255
- DBMS, 6
- deleting, 263
- denormalizing, 237-238
- design, 244
- disk space, 244
- DML commands, 490
- DROP DATABASE statement, 262-263
- dynamic database environment, 402-404
- dynamic performance views, Oracle data dictionary, 458-461
- EXPLAIN PLAN tool, 409
- fragmentation reports, 537
- growth, Oracle data dictionary, 456-457
- history, 6
- indexes, 379
  - defragmenting, 403
- mailing list tables, code, 403-404
- MUSIC
  - ARTISTS table, 469
  - MEDIA table, 470

**RECORDINGS**

- table, 470
- MySQL, CREATE INDEX statement, 373
- MySQL table commands, Oracle data dictionary, 460-461
- normalizing, 127, 229, 235, 244, 268
  - benefits, 236-237
  - data redundancy, 231
  - denormalizing, 237-238
  - drawbacks, 237
  - end user needs, 230-231
  - first normal form, 232
  - foreign keys, 233
  - logical database design, 230
  - naming conventions, 234
  - normal forms, 231
  - primary keys, 233
  - raw databases, 229-230
  - referential integrity, 235
  - second normal form, 233
  - third normal form, 234
- NULL, 55-57
- objects
  - Oracle data dictionary, 452-456
  - scripts, maintaining, 279
- ODBC. *See* ODBC
- OLAP, 397-398
- OLTP, 397-398
- overflow, 298
- PAYMENTS, table structure, 246
- performance enhancements, disks, 405-407
- performance obstacles, identifying, 407-408
- physical structure, 230

- queries, PL/SQL, 631
- RDBMS. *See* RDBMS
- records, examining with cursors, 472
- referential integrity, 302
- relational, 6-10
  - Codd's 12 rules, 6-10
  - dates, 309
  - DEPENDENTS and EMPLOYEE tables, retrieving fields, 10
  - DEPENDENTS table records, 10
  - EMPLOYEE table records, 8-9
  - JOIN, 9
  - joins, 9
  - table columns and fields, 8
  - time, 309
  - UNION, 9
  - unions, 9
- security, Oracle data dictionary, 451-452
- SELECT statements, 504
- sessions, Oracle data dictionary, 458-459
- SHOW DATABASES command, 256
- space allocation, Oracle data dictionary, 457-458
- SQL statements, generating, 504
- tables
  - CREATE TABLE statement examples, 737
  - creating, 577
  - data, entering, 577
  - defragmenting, 403
  - foreign keys, 235-236
  - INSERT statement examples, 36, 118
  - lookup, 292

- primary keys, 235
  - temporary, 292
- tablespaces, dropping into, 523-524
- TEMPDB, temporary tables, creating, 471
- TKPROF tool, 409
- transaction control, 354
- transactions
  - beginning, 356-359
  - canceling, 361-364
  - finishing, 359-361
  - rollback segments, 401-402
  - savepoints, 364-366
- truncation, 298
- tuning, 405-407
  - tools (built-in), 409
- users of Oracle data dictionary, 450-451
- datatypes, date, format, 748, 752**
- date and time functions, 187-188**
  - ADD MONTHS/ADD DATE, 188-190
  - LAST DATE, 190-191
  - LAST DAY, 190-191
  - leap years, LAST DAY function, 191
  - MONTHS BETWEEN, 191-193
  - NEXT DAY, 193
  - SYSDATE, 193-195
- DATE conversions, 619-623**
- date conversions (T-SQL), 680-681**
- DATE data types, 249, 310-312**
  - T-SQL, 664
- DATE FORMAT function, 223-224, 321**
- DATE HIRE function, 316**
- date picture, parts of, 620**
- DATEADD function, 316**
- DATEADD/DATE\_ADD function, 316**
- DATEDIFF function, 321**
- DATENAME function, 321, 325-326**
- DATEPART function, 321**
- dates, 309**
  - ANSI standard data types, 310-311
  - character strings, converting to dates, 326
  - COLUMN command, 620
  - CONVERT command, 680
  - converting to character strings, 325
  - current dates, 313-314
  - data types, implementing, 312
  - date pictures, 322-324
  - DATETIME elements, 311
  - format strings, specifiers, 223-224
  - formats, converting, 321-322
  - formatting, 538-539
  - functions
    - applying to queries, 312
    - current dates, 313-314
    - dates and time periods, comparing, 320
    - dates, subtracting, 318-320, 713
    - of MySQL, 321
    - of SQL Server, 320
    - time zones, 315
    - time, adding to dates, 315-318
  - functions (MySQL), 223-225
  - storing, 326
  - subtracting, 318-320, 713
  - time, adding to dates, 315-318
  - and time periods, comparing, 320
  - time zones, 315
  - TO CHAR function, 619
  - TO DATE function, 622
  - values, storing, 310
- dates datatype format, 748, 752**
- dates of birth, storing, 320**
- DATETIME data type, 309-312**
- datetime data type (T-SQL), 664**
- DATETIME elements, 311**
- DATETIME value, 315**
- DATE\_ADD function, 316**
- DAYNAME function, 321**
- days, breaking into hours, minutes, seconds, 533-535**
- DBA CATALOG view, 453**
- DBA DATA FILES view, 457**
- DBA EXTENTS view, 457**
- DBA INDEXES view, 454**
- DBA role, creating for Personal Oracle, 421**
- DBA ROLE PRIVS view, 451**
- DBA roles, security for Personal Oracle, 433**
- DBA SEGMENTS view, 456-457**
- DBA SYS PRIVS view, 452**
- DBA TABLES view, 453**
- DBA TABLESPACES view, 455**
- DBAs (database administrators), 255, 401, 439**
  - batch transactions, COMMIT statement, 401-402
  - data dictionary, 439
  - of Oracle data dictionary
    - dynamic performance views, 458-461
    - growth, 456-457



- INFORMATION\_SCHEMA, 461-463
- MySQL table
  - commands, 460-461
  - objects, 452-456
  - security, 451-452
  - sessions, 458-459
  - space allocation, 457-458
  - users, 450-451
  - views, 449-450
- DBA\_ROLES view, 451**
- DBA\_ROLE\_PRIVS view, 451**
- DBA\_SYS\_PRIVS view, 451**
- DBA\_USERS view, 451**
- DBMS (database management system), 6, 241, 413**
  - MODIFY clause, 261
- DBMS OUTPUT package, 643**
- DDL (Data Dictionary Language), 631**
  - commands, PL/SQL, 631
- DEALLOCATE command, 475**
- DEALLOCATE statement, syntax, 475**
- decimal values, deleting, 533**
- DECLARE command, of PROCEDURE section (PL/SQL blocks), 636**
- DECLARE cursor\_name CURSOR statement, 473**
- DECLARE keyword, variables, 670**
- DECLARE section (PL/SQL blocks), 631-632**
  - constant assignment, 633
  - cursor definitions, 633
  - %ROWCOUNT attribute, 634-635
  - %ROWTYPE attribute, 634
  - %TYPE attribute, 633-634
  - variable assignment, 632-633
- declaring**
  - global variables, 668-669
  - local variables, 668
  - variables, table name conflicts, 635
- DECODE function, 616-619**
- default storage parameters, large tables, 569**
- DEFINE command, SQL\*Plus variables, 611-612**
- DEL command, 588**
- DELETE, INSERT, VALUES statement, 298**
- DELETE ANY TABLE system privilege, 424**
- DELETE clause, 299**
- DELETE command, 332, 418, 440**
  - views, 343
- DELETE event, 490**
- DELETE operation, tables (triggers), 479**
- DELETE statement, 285**
  - data, deleting, 298-302
  - table views, 345
  - WHERE clause, 300
- deleting. *See also* removing**
  - databases, 263
  - decimal values, 533
  - information, 298-302
  - lines in code, 591
  - object privileges, 425
  - records, parent/child relationships, 279
  - roles, 420
  - SQL\*Plus settings, 603
  - stored procedures, 478-479
  - tables, 262
  - users from Personal Oracle, 419
- delimited text files, exporting, 303**
- denormalizing databases, 237-238**
- DEPENDENTS table, 10**
- DESC command, table structure, viewing, 593**
- DESC operator, sorting indexes, 378**
- DESCRIBE command**
  - table structure, 645
  - table structure, viewing, 593-594
  - UDTs (User Defined Types), attributes, 495
- designing**
  - databases, 230-231
  - triggers, 479
- designs, databases, 244**
- Destination dialog box, 303**
- development tools for applications, 575**
  - Java, 576
  - .NET, 577
  - ODBC, 576
  - Personal Oracle, 576
- diagnostic tools, SQL Server, 681**
- diagrams, syntax, 86**
- dictionaries. *See also* data dictionary**
  - data
    - CASE tools, 245
    - creating, 244-245
    - RDBMS packages, 245
  - MySQL data dictionary, 440
  - Oracle data dictionary. *See* Oracle, data dictionary
- Direct Access Method, 370**
- direct invocation, 17**
- directories, installing MySQL, 687**

disabling table constraints,  
516-517

disconnecting from terminal  
monitor (MySQL), 689

disk space for  
databases, 244

disks, enhancing database  
performance, 405-407

displaying

output to users, 643-644

session settings, 594-595

DISTINCT, selecting multiple  
columns, 566

DISTINCT clause, views,  
creating, 345

DISTINCT function, 453

DISTINCT keyword, 33

distributions, installing  
MySQL binary distribution  
file, 686-687

division sign (/) arithmetic  
operator, 48-49

DLL (Data Definition  
Language), 437

DML (Data Manipulation  
Language), 537

commands, 490

PL/SQL, 631

subqueries, 537

dot (.), table names, 248

double dashes (--), 546

double hyphens (- -),  
comments, 624

double pipe (||)  
concatenation character  
operator, 67-70

double pipe (||)  
operator, 338

double quotation marks (" "),  
literal strings, 620

double quotation marks (" "),  
aliases, 50

DOWN ARROW key  
(cursors), 472

downloading

MySQL, 14, 686

MySQL 3.23, 686

Personal Oracle8, 15

Dr. Codd's 12 rules for  
relational databases, 6-10

drawbacks to normalizing  
databases, 237

driving tables, 135

DROP ANY INDEX system  
privilege, 424

DROP ANY PROCEDURE  
system privilege, 424

DROP ANY ROLE system  
privilege, 424

DROP ANY SYNONYM  
system privilege, 424

DROP ANY TABLE system  
privilege, 424

DROP ANY TRIGGER system  
privilege, 424

DROP ANY VIEW system  
privilege, 424

DROP command, stored  
procedures, 478

DROP DATABASE statement,  
262-263

DROP INDEX statement,  
375-376

DROP statement, syntax,  
493-494

DROP TABLE, 567

DROP TABLE command, 262

DROP TABLE statement,  
261-263

DROP TRIGGER statement,  
syntax, 491

DROP USER command, 419

DROP USER system  
privilege, 424

DROP VIEW command, 350

DROP VIEW statement,  
removing views, 350

dropping

indexes, 375-376

from data loads,  
400-401

MAX HITS table, 261

roles, 421

synonyms, syntax, 432

tables, 377

tablespaces into databases,  
523-524

views, 350

dropping unqualified tables,  
566-567

DUAL table, 314, 615-616

dynamic database  
environment

definition, 402

indexes, rebuilding,  
402-404

tables, rebuilding, 402-404

dynamic performance views,  
DBA of Oracle data  
dictionary, 458-461

dynamic SQL (Structured  
Query Language), 482-483

dynamic SQL, call-level  
interfaces, 18

## E

echos

SET ECHO OFF  
command, 505, 509

SET ECHO ON  
command, 505

ED FILENAME command,  
507-509

EDIT command

files, editing, 595-596

script files, creating, 623

EDIT PROD.LST command,  
query output, spooling, 599

editing output files, 509

**elements**

DATETIME, 311  
of queries, arranging,  
393-395

**embedded SQL, 17, 481**

Dynamic SQL, 482-483  
Static SQL, 482-484

**embedding SQL, 17-18**

in application  
programming, 575

**embedding subqueries, 151-153****EMPLOYEE table**

and DEPENDENTS table,  
retrieving fields, 10  
records, 8-9  
SELECT statement, 13

**END statement**

of PROCEDURE section  
(PL/SQL blocks), 635  
program flow control, 672

**end users, database design, 230-231****ending. See finishing****engineers, system (data dictionaries), 439****entering**

data, INSERT  
statement, 286

MySQL terminal monitor  
commands, 690-692

records

INSERT, SELECT  
statement, 292-294

INSERT, VALUES  
statement, 286-289

**entering data, accuracy of, 268****environments, dynamic database, 402****equal sign (=), 540**

tables, joining, 137

**equal sign (=) comparison operator, 56-58****equal sign (=) relational operator, subqueries, 159****equality**

equit joins, joining tables,  
129-137

non-equi-joins (non-  
equality) of tables,  
137-139

**equi-joins (tables), 129-137****@@error variable, 669****errors**

common logical mistakes

allowing large tables to  
take default storage  
parameters, 569

Cartesian product,  
567-568

dropping unqualified  
tables, 566-567

failure to budget system  
resources, 570

failure to compress  
large backup files, 570

failure to enforce file  
system structure  
conventions, 568

failure to enforce input  
standards, 568

placing objects in the  
system tablespace,  
569-570

use of DISTINCT when  
selecting multiple  
columns, 566

use of public synonyms  
in multischema  
databases, 567

using reserved words in  
your SQL statement,  
564-566

common SQL errors

cannot create operating  
system files, 564

columns ambiguously  
defined, 558-559

commands not properly  
ended, 559

escape character in your  
statement—invalid  
character, 564

FROM keyword not  
specified, 553-554

group function not  
allowed, 554-555

inserted value too large  
for columns, 562-563

insufficient privileges  
during grants,  
563-564

integrity constraints  
violated—parent key  
not found, 561

invalid column names,  
555-556

invalid usernames or  
passwords, 553

missing commas, 558

missing expressions,  
559-560

missing keywords, 556

missing left parenthesis,  
556-557

missing right  
parenthesis, 557-558

not enough arguments  
for functions, 560

not enough values,  
560-561

Oracle not  
available, 562

table or view that does  
not exist, 552

TNS: Listener could  
not resolve SID  
given in connect  
descriptor, 563

exceptions, definition, 640

query clauses, 109

undoing with ROLLBACK  
command, 292, 298

**escape characters, 564****events, 490**

**EXCEPTION section (PL/SQL blocks), 631, 640**

blocks, executing, 642-643  
 comments, inserting, 642  
 exceptions, 641-642  
 output to users, displaying, 643-644

**EXCEPTION section (PL/SQL script files), executing script files, 643****exceptions**

definition, 640  
 handling, 641-642  
 raising, 641  
 ZERO DIVIDE, 643

**EXECUTE ANY PROCEDURE system privilege, 424****EXECUTE command, creating stored procedures, 477****EXECUTE statement, Print Artists Name procedure, executing, 477****executing**

PL/SQL blocks, 642-643  
 PL/SQL script files, 643

**EXISTS keyword**

program flow control, 675  
 subqueries, embedding, 169-174

**EXP function, 196-197****EXPLAIN PLAN tool, 409****exploring views, 335-337****exporting**

data from foreign sources, 303-305  
 delimited text files, 303

**expressions**

queries, 40  
 regular expressions, 497-498

**extensions, ANSI SQL, 662-663****extensions of files**

.CTL, 305  
 .FMT, 304  
 .sql, 597

**EXTERNALLY option, Personal Oracle database security, 416****F****failure to budget system resources, 570****failure to enforce file system structure conventions, 568****failure to enforce input standards, 568****FALSE value, comparison operators, 55****feedback**

SET FEEDBACK OFF  
 command, 506, 509  
 SET FEEDBACK ON  
 command, 506

**FEEDBACK commands, SQL\*Plus work environment, customizing, 600****FETCH command**

cursors, scrolling, 473-474  
 of PROCEDURE section (PL/SQL blocks), 636-637

**fields**

ACCOUNT ID (BILLS table), creating indexes, 373-377  
 AMOUNT (BILLS table), creating indexes, 377-378  
 data types, 249-250  
 foreign key, 246  
     NULL values, 252  
 indexing on, 379-381  
 key, creating, 246-247  
 names, 249  
 NULL value, 250-252  
 number-defined, " (quotation marks), 59  
 primary key, 246  
     NULL values, 252  
 relational databases, 8

retrieving from  
 DEPENDENTS and  
 EMPLOYEE tables, 10

ROWID, 252

SALARY (SALARIES table), updating, 427  
 unique, 252-254

**File menu commands, Open, 303****filenames**

ED FILENAME command, 507, 509  
 SPOOL FILENAME command, 506  
 START FILENAME command, 506

**files**

CTL [edit, period before] extension, 305  
 delimited text, exporting, 303  
 EDIT command, 595-596  
 EDIT PROD.LST command, 599  
 .FMT extension, 304  
 GET command, 595-596  
 manipulating with file commands, 595  
 MySQL binary distribution, installing, 686-687  
 output, editing, 509  
 PL/SQL script, 646-652  
     executing, 643  
 query output, spooling, 598-599  
 RUN command, 598  
 SAVE command, 595-596  
 script  
     creating with EDIT command, 623  
     tables, creating and entering data, 577-578  
 SPOOL command, 598  
 SPOOL OFF command, 599

- SQL (Structured Query Language), running, 623-624
- .sql extension, 597
- START command, 598
- starting, 596-598
- TAR command, 687
- filtering data for views, 336**
- finding columns, 128-129**
- finishing transactions, 359-361**
- First Federal Financial Bank.**  
See **banking applications**
- first normal form, normalizing databases, 232**
- float data type (T-SQL), 664**
- FLOOR function, 196, 536**
- flow control (programs), T-SQL, 672**
  - BEGIN statement, 672
  - BREAK command, 677
  - CONTINUE command, 677-678
  - END statement, 672
  - EXISTS keyword, 675
  - FOR loop, 676
  - IF, ELSE statement, 673-675
  - query results, testing, 675-676
  - WHILE loop, 676-679
- .FMT file extension, 304**
- FOR loop, program flow control, 676**
- FOR-LOOP, conditional statements, 640**
- foreign key fields, NULL values, 252**
- foreign keys, 246**
  - constraints (on data), 274-275
  - parent/child table relationships, 275-276
  - databases, normalizing, 233
  - field, 246
  - tables, 235-236
- foreign sources, importing and exporting data, 303-305**
- FORMAT command, formatting columns, 605-606**
- format strings, specifiers, 223-224**
- formats**
  - date datatype, 748, 752
  - of dates. See **dates**
- formatting**
  - columns, 605-606
  - dates, 538-539
  - reports, 604
  - SQL\*Plus output, 603
- forms, normal**
  - database normalization, 231
  - first, 232
  - second, 233
  - third, 234
- forward slash (/)**
  - PL/SQL, 642
  - SQL\*Plus buffer, 590
  - table names, 248
- fourth-generation language (4GL), 12**
- fragmentation reports for databases, 537**
- FROM keyword, 23-24, 87**
  - errors, 553-554
- full-table scans, 371**
  - avoiding, 391-392
- functions**
  - ABS, 195
  - ADD MONTHS/ADD DATE, 188-190
  - aggregate
    - ANSI standard, 179
    - AVG, 182-183
    - COUNT, 180-181
    - data, summarizing, 180
    - MAX, 184
    - MIN, 185
    - SELECT statement, 349
    - STDDEV, 186-187
    - subqueries, embedding, 160-161
    - SUM, 181-182
    - VARIANCE, 186
  - arithmetic operations, 195
  - ABS, 195
  - CEIL, 196
  - EXP, 196-197
  - FLOOR, 196
  - LN, 197-198
  - LOG, 197-198
  - MOD, 198
  - POWER, 199
  - SIGN, 199-200
  - SQRT, 200-201
  - AVG, 182-183
  - groups, 608
  - subqueries, embedding, 160-161
  - built-in, 179
  - C, Static SQL, 483-484
  - CEIL, 196
  - character, 201
    - CHR, 201-202
    - CONCAT, 202-203
    - INITCAP, 203
    - INSTR, 214
    - LENGTH, 215
    - LOWER, 203-205
    - LPAD, 205-206
    - LTRIM, 206-207
    - REPLACE, 207-209
    - RPAD, 205-206
    - RTRIM, 206-207
    - SUBSTR, 209-213
    - SUBSTR/MID, 213
    - TRANSLATE, 213-214
    - UPPER, 203-205

character (MySQL),  
     219-222  
 CHR, 201-202  
 CONCAT, 202-203  
 conversion, 215-217  
 COUNT, 180-181  
     groups, 608  
     subqueries, embedding,  
         160-161  
 COUNT(\*), 507  
 CURDATE, 225  
 date  
     applying to queries, 312  
     current dates, 313-314  
     dates and time periods,  
         comparing, 320  
     dates, subtracting, 318-  
         320, 713  
     of MySQL, 321  
     of SQL Server, 320  
     time zones, 315  
     time, adding to dates,  
         315-318  
 date and time, 187-188  
     ADD MONTHS/ADD  
         DATE, 188-190  
     LAST DAY, 190-191  
     MONTHS BETWEEN,  
         191-193  
     NEXT DAY, 193  
     SYSDATE, 193-195  
 DATE FORMAT,  
     223-224, 321  
 DATE HIRE, 316  
 DATEADD, 316  
 DATEADD/DATE\_ADD,  
     316  
 DATEDIFF, 321  
 DATENAME, 321, 325  
 DATEPART, 321  
 dates (MySQL), 223-224  
 DATE\_ADD, 316  
 DAYNAME, 321

DECODE, 616-619  
 DISTINCT, 453  
 EXP, 196-197  
 FLOOR, 196, 536  
 GETDATE( ), 313, 321  
 GREATEST, 217-218  
 INITCAP, 203  
 INSTR, 214, 220  
 LAST DAY, 190-191  
 LEAST, 217-218  
 LEFT, 221  
 LENGTH, 215, 219  
 LN, 197-198  
 LOCATE, 219-220  
 LOG, 197-198  
 LOWER, 203-205  
 LPAD, 205-206, 220  
 LTRIM, 206-207, 222  
 MAX, 184, 539, 676  
     subqueries, embedding,  
         160-161  
 MIN, 185  
     subqueries, embedding,  
         160-161  
 miscellaneous, 217-219  
 MOD, 52, 198, 536  
 MONTHS BETWEEN,  
     191-193, 321  
 NEXT DAY, 193, 321  
 PERIOD DIFF, 321  
 POWER, 199  
 QUARTER, 321  
 REPLACE, 207-209  
 RIGHT, 221  
 RPAD, 205-206, 220, 538  
 RTRIM, 206-207, 222  
 SIGN, 199-200  
 SQRT, 200-201  
 STDDEV, 186-187  
 SUBSTR, 209-213  
 SUBSTR/MID, 213

SUBSTRING, 221  
 SUM, 181-182, 341  
     groups, 608  
     subqueries, embedding,  
         160-161  
 SYSDATE, 193-195, 314  
     pseudocolumn, 314  
 SYSDATE (in an Oracle  
     database), 193  
 TIME FORMAT, 224  
 TO CHAR, 215-217, 544  
     dates, 619  
 TO DATE, 326, 622  
 TO NUMBER, 217  
 TRANSLATE, 213-214  
 TRIM, 222  
 TRUNC, 319  
 UPDATE, views, 345  
 UPPER, 203-205  
 USER, 218-219  
 VARIANCE, 186

## G

**generating shell scripts,**  
**523-524**

**GET command, 614**  
     files, getting, 595-596

**GET filename**  
**command, 588**

**GETDATE( ) function,**  
**313, 321**

**global variables**

    declaring, 668-669  
     @@rowcount global,  
         cursors, testing  
         status, 475  
     @@sqlstatus global,  
         cursors, testing  
         status, 475

**go command, 666**

**GRANT ANY PRIVILEGE**  
**system privilege, 424**

**GRANT ANY ROLE system privilege, 424**

**GRANT statements, 512-513**

syntax, 433

tables, 514

**WITH GRANT OPTION clause, 433-434**

**Graphical User Interface (GUI) tool, 440**

**greater than or equal to sign (>=) comparison operator, 59**

**greater than sign (>) comparison operator, 59**

**GREATEST function, 217-218**

**GROUP BY clause, 98-105, 116-117**

SELECT statement, 343

subqueries, correlated, 168-169

**group functions. See aggregate functions**

**groups**

AVG function, 608

COUNT function, 608

set-oriented, 6

SUM function, 608

summaries

BREAK ON command, 607-608

COMPUTE command, 608-610

creating, 606-610

**growth, DBA of Oracle data dictionary, 456-457**

**GUI (Graphical User Interface) tool, 440**

## H

**handling exceptions, 641-642**

**HAVING clause, 105-111, 116-117**

subqueries, correlated, 168-169

**HEADING command, formatting columns, 605-606**

**headings**

SET HEADING OFF command, 506, 509

SET HEADING ON command, 506

**histories**

command-line, MySQL terminal monitor, 692

databases, 6

SQL, 5-6

**hours, breaking days into, 533-535**

**hyphens, -- (double) (comments), 624**

## I

**IBM, RDBMS (relational database management system), 5**

**IBM DB2, 16**

**IDENTIFIED BY clause, 417**

**identifying**

database performance obstacles, 407-408

users

for data dictionary, 438-439

of Oracle data dictionary, 441-442

**@@identity variable, 669**

**@@idle variable, 669**

**IF, ELSE statement, program flow control, 673-675**

**IF, THEN statement, 638**

**IL (intermediary language), 583**

**implementations, SQL, 14**

**Implementing Client/Server Computing, 12**

**Import/Export Setup dialog box, 303-304**

**import/export tools, 303-305**

**importing data from foreign sources, 303-305**

**IN keyword, embedding subqueries, 173**

**IN operator, 78-80**

values, comparing, 538

**indexes, 370-371**

adding to tables, 393

B-tree, 371

clustered, 373, 384-385

column data, 378

columns

composite indexes, 381

NULL values, 379

composite, 379-381, 393

**CREATE INDEX statement, 369**

code, 372

MySQL database, 373

UNIQUE keyword, 381-382

creating, 369, 373-377

on ACCOUNT ID field (BILLS table), 373-377

on AMOUNT field (BILLS table), 377-378

on views, 379

SQL syntax, 371

data retrieval, 378

database space, 379

defragmenting, 403

Direct Access Method, 370

dropping, 375-376

from data loads, 400-401

fields, indexing on, 379

full-table scans, 371

joins, 382-384

performance improvement, 378

- pointers, 370
- primary keys, 382
- purpose, 370
- queries, 378
  - timing, 383
- rebuilding, 402-404
- scripts, maintaining, 279
- Sequential Access Method, 370
- sorting DESC operator, 378
- storing, 379
- trees, 370-371
- indexing on fields, 379-381**
- individual columns, selecting, 28**
- individual's age, computing from date of birth, 532-533**
- individuals, storing dates of birth, 320**
- inequalities (< or !=) comparison operator, 62-63**
- information, deleting, 298-302**
- INFORMATION\_SCHEMA, 461-462**
  - user privileges, 462-463
- INITCAP function, 203**
- initial access privileges, MySQL on UNIX-based systems, 688**
- INITIAL SIZE (tables), 255**
- INNER JOINs and OUTER JOINs, comparing, 139-143**
- INPUT command, 591, 612**
- INSERT ANY TABLE system privilege, 424**
- INSERT command, 332, 335, 440**
  - views, 343
- INSERT event, 490**
- INSERT keyword, 577**
- INSERT operation, tables (triggers), 479**

- INSERT statements, 285**
  - data, entering, 286
  - date datatype format, 748, 752
  - examples, 36, 118
  - NULL values, inserting, 289-290
  - records
    - entering with INSERT, SELECT statement, 292-294
    - entering with INSERT, VALUES statement, 286-289
  - tables, populating, 333, 743-755, 760-761
  - unique values, inserting, 291-292
  - views, 345
- INSERT, SELECT statement, 292-294**
- INSERT, VALUES statement**
  - DELETE, 298
  - records, entering, 286-289
  - syntax, 286
- inserting**
  - comments, 642
  - NULL values, 289-290
  - unique values, 291-292
- installing**
  - Linux, 764-765
  - MySQL binary distribution file, 686-687
  - MySQL directories, 687
  - MySQL on UNIX-based systems, 686-687
  - Windows, 763-764
- INSTR function, 214, 220**
- int data type (T-SQL), 663**
- INTEGER data type, 250**
- INTEGER(n) data type, 250**
- integrity**
  - Oracle SQL\*Plus
    - referential integrity reports, 280-283

- referential
  - databases, normalizing, 235
  - of databases, 302
  - triggers, 479-480

- integrity constraints violated—parent key not found, 561**

- integrity of data, controlling, 267**

## interfaces

- APIs, 16
- call-level, 18
- GUI tool, 440

- intermediary language (IL), 583**

- International Standards Organization (ISO), 6**

- INTERSECT set operator, 77**

- invalid characters, troubleshooting, 564**

- invalid column names, errors, 555-556**

- @@io busy variable, 669**

- isamchk utility, MySQL on UNIX-based systems, 694**

- ISO (International Standards Organization), 6**

- @@isolation variable, 669**

- ISQL, 577-580**

## J

### Java

- application programming, 581-583
- applications, developing, 576
- databases, accessing, 581
- JDBC, 581
- and PL/SQL stored procedures, comparing, 657

- JDBC, 581**

- JOIN, relational databases, 9**



**JOIN ON syntax, 140**

**join operations, 233**

**JOIN statement, 121**

**join views, handling records, 345**

**joining tables, 121**

columns, finding, 128-129

cross joining (Cartesian product), 123-128

equi-joins (equality), 129-137

non-equi-joins (non-equality), 137-139

OUTER JOINS and INNER JOINS, comparing, 139-143

in SELECT statements, 121-123

self joins, 143-146

WHERE clause, 126

**joins**

cross, 126

indexes, 382-384

OUTER JOINS and INNER JOINS, comparing, 139-143

relational databases, 9

self joins (tables), 143-146

## K

**key fields**

creating, 246-247

foreign, 246

primary, 246

**keyboard shortcuts, Ctrl+D, 689**

**keys**

DOWN ARROW (cursors), 472

foreign, 246

constraints (on data), 274-276

databases, normalizing, 233

parent/child relationships, 275-276

tables, 235-236

foreign key fields, NULL values, 252

primary, 246

ALTER TABLE statement, 273

constraints (on data), 271-273

databases, normalizing, 233

indexes, 382

NULL values, 252

records, 271

tables, 232-235

**keywords**

ALL, 33

subqueries, embedding, 169-174

ANY, embedding subqueries, 169-174

case sensitivity, 665

CREATE, 577

DECLARE, variables, 670

DISTINCT, 33

EXISTS

program flow control, 675

subqueries, embedding, 169-174

FROM, 23-24, 87

FROM:errors, 553-554

IN, embedding subqueries, 173

INSERT, 577

missing keywords, errors, 556

NOT NULL, 250

PUBLIC, granting privileges, 422

queries, 23

SELECT, 23-24, 87

SOME, embedding subqueries, 172

UNIQUE, CREATE INDEX statement, 381-382

**kilobytes, converting bytes to, 536**

## L

**@@language variable, 669**

**languages. See also DDL; PL/SQL; SQL**

3GLs (third-generation languages), 5

4GL (fourth-generation language), 12

cross-product (SQL), 12-13

DLL, 437

DML, 631

subqueries, 537

non procedural. *See* SQL

**@@languid variable, 669**

**LANs (Local Area Networks), 11**

**LAST DAY function, 190-191**

**leap years, LAST DAY function, 191**

**LEAST function, 217-218**

**LEFT function, 221**

**LENGTH function, 215, 219**

**less than or equal to sign (<=) comparison operator, 60-61**

**less than sign (<) comparison operator, 60-61**

**licensing MySQL, 685**

**LIKE character operator, 63-65**

**line numbers**

\* (asterisk), 590

SQL\*Plus buffer, 589

**lines in code, 591**

**LINESIZE option (SQL\*Plus), 601-602**

**Linux, installing, 764-765**

**LIST command, SQL\*Plus buffer, 588**

**LIST line number command, 588**

**lists, viewing tables, 256-257**  
tables, viewing, 256-257

**literal strings, “ ” (double quotation marks), 620**

**LN function, 197-198**

**loading data, disabling table constraints, 516-517**

**loads**

batch and OLTP,  
comparing, 398-400  
data, dropping indexes,  
400-401

**local area network (LAN), 11**

**local variables**

data, retrieving, 671  
declaring, 668

**LOCATE function, 219-220**

**LOCK ANY TABLE system privilege, 424**

**LOCK TABLE command, 356**

**LOG function, 197-198**

**logging off SQL Server connections, 471**

**logical data types, bit (T-SQL), 665**

**logical database design, 230-231**

**logical model, 230**

**logical operators, 70-71, 693**

AND, 71-72

NOT, 73-75

OR, 72-73

avoiding in queries,  
396-397

**logs, transactional, 398**

**LONG data type, 250**

**LONG RAW data type, 250**

**LONG RAWBinary data type, 249**

**LONG VARCHAR data type, 250**

**LONGVARIABLE data type, 249**

**lookup tables, 292**

**LOOP, conditional statements, 639**

**loops**

FOR, program flow control, 676

FOR-LOOP, 640

LOOP, 639

of PROCEDURE section (PL/SQL blocks),  
638-640

WHILE

program flow control, 676

tables, scrolling,  
678-679

WHILE-LOOP, 639-640

**LOWER function, 203-205**

**LPAD function, 205-206, 220**

**LTRIM function, 206-207, 222**

## M

**mailing list tables, code, 403-404**

**maintaining tables, 241**

**management information systems. See MIS**

**managing**

constraints (on data),  
278-283

security with roles, 488-489

**manipulating data, 285**

**MAX BYTES, 449**

**@@max connections variable, 669**

**MAX function, 184, 539, 676**

subqueries, embedding,  
160-161

**@@maxcharlen variable, 669**

**MAXEXTENTS (tables), 255**

maximum values,  
subqueries, 539-540

**MAX\_HITS table, dropping, 261**

**MEDIA table, 470**

**megabytes, converting bytes to, 536**

**memory, cursors, 476**

**merging data, 233**

**Microsoft**

Access, import/export tools, 303-304

SQL Server, import/export tools, 304

**Microsoft Access relational database management system, 414**

**Microsoft FoxPro database management system, 414**

**MIN function, 185**

subqueries, embedding,  
160-161

**MINEXTENTS (tables), 255**

**MINUS (difference) set operator, 77-78**

**minus sign (-) arithmetic operator, 46-48**

**minutes, breaking days into, 533-535**

**MIS (management information systems), 11**

**miscellaneous functions, 217**

**misisng keywords, errors, 556**

**missing left parenthesis, errors, 556-557**

**missing right parenthesis, errors, 557-558**

**mistakes. See errors**

**MLSLABEL data type, 250**

**MOD function, 52, 198, 536**

**modes, batch (MySQL terminal monitor), 692-693**

- MODIFY clause, 261**
- module language, 17**
- modulo sign (%) arithmetic operator, 51-53**
- money data type (T-SQL), 664**
- money data types (T-SQL), 664**
- monitors, 688. *See also* terminal monitor**
- MONTHS BETWEEN function, 191-193, 321**
- most restrictive condition (WHERE clause), 393-395**
- multiplication sign (\*) arithmetic operator, 49-51**
- multischema databases, public synonyms, 567**
- MUSIC database**
  - ARTISTS table, 469
  - cursors, creating, 473
  - MEDIA table, 470
  - RECORDINGS table, 470
- MySQL, 4, 14**
  - 3.23 (current version),
    - downloading, 686
  - anonymous users, 688
  - binary distribution file, installing, 686-687
  - character functions, 219-222
  - CHECKS table, 25
  - columns, aliases, 46
  - commands, 691
  - data dictionary, 440
  - data functions, 321
  - database, CREATE INDEX statement, 373
  - DATE data type, 312
  - date datatype format, 748, 752
  - date functions, 223-225
  - date pictures, 323
  - DATETIME data type, 312
  - directories, installing, 687
  - downloading, 14, 686
  - import/export tools, 305
  - INSERT statements, date datatype format, 748, 752
  - licensing, 685
  - MYSQLADMIN
    - EXTENDED-STATUS command, 694
  - MYSQLADMIN
    - PROCESSLIST command, 694
  - MYSQLADMIN
    - VARIABLES command, 694
  - MYSQLSHOW -K DB NAME TBLE NAME command, 694
  - MYSQLSHOW
    - command, 694
  - MYSQLSHOW DB NAME TBLE NAME command, 694
  - passwords, changing, 688
  - PROMPTs, writing queries, 24
  - queries, 39
  - root, passwords, 688
  - SHOW COLUMNS
    - command, 694
  - SHOW DATABASES
    - command, 694
  - SHOW FIELDS
    - command, 694
  - SHOW INDEX
    - command, 694
  - SHOW MYSQLSHOW — STATUS DB NAME command, 694
  - SHOW MYSQLSHOW DB NAME
    - command, 694
  - SHOW PROCESSLIST
    - command, 694
  - SHOW STATUS
    - command, 694
  - SHOW TABLE STATUS
    - command, 694
  - SHOW TABLES
    - command, 694
  - SHOW VARIABLES
    - command, 694
  - SQL statements, ; (semicolon), 690
  - SQLSTATISTICS
    - command, 694
  - table commands, Oracle data dictionary, 460-461
  - tables (code examples)
    - creating, 731-738
    - populating, 743-751
  - TIME data type, 312
  - TIMESTAMP data type, 312
  - UDTs (User Defined Types), 493
  - on UNIX-based systems, 685
    - administration, 686
    - initial access privileges, 688
    - installing, 686-687
    - isamchk utility, 694
    - mysqlaccess utility, 694
    - mysqladmin utility, 694
    - mysqldump utility, 694
    - mysqlimport utility, 694
    - mysqlshow utility, 694
    - # (pound sign), 687
    - starting, 687-688
    - stopping, 687-688
    - terminal monitor, 688
    - terminal monitor, batch mode, 692-693
    - terminal monitor, command-line history, 692
    - terminal monitor, command-line options, 689-690
    - terminal monitor, connecting to databases, 689

- terminal monitor,
  - disconnecting from, 689
- terminal monitor,
  - entering commands, 690-692
- terminal monitor,
  - prompts changing, 692
- terminal monitor,
  - SHOW commands, 693-694
- utilities, 694
- use database command,
  - tables, populating, 743
- user root, 688
- YEAR data type, 312
- MySQL server, database security, 415**
- mysqlaccess utility, MySQL on UNIX-based systems, 694**
- MYSQLADMIN EXTENDED-STATUS command, 694**
- MYSQLADMIN PROCESSLIST command, 694**
- mysqladmin utility, MySQL on UNIX-based systems, 694**
- MYSQLADMIN VARIABLES command, 694**
- mysqld daemon, starting, 687**
- mysqld daemon, stopping, 688**
- mysqldump utility, MySQL on UNIX-based systems, 694**
- mysqlimport utility, MySQL on UNIX-based systems, 694**
- MYSQLSHOW -K DB NAME TBLE NAME command, 694**
- MYSQLSHOW command, 694**
- MYSQLSHOW DB NAME TBLE NAME command, 694**
- mysqlshow utility, MySQL on UNIX-based systems, 694**

## N

### NAME clause, 113

#### names

- of columns, 336
  - shortening, 60
- of fields, 249
- spaces between, 69
- of tables, 248-249
  - conflicts, 635

### naming columns, 337-338

### naming conventions, normalizing databases, 234

### @@ncharsize variable, 669

### NCLOB data type, 250

### @@nestlevel variable, 669

### nested transactions, 357-358

#### nesting

- subqueries, 162-165
- triggers, 481

#### .NET

- application programming, 583-584
- applications,
  - developing, 577

### networks, LANs (Local Area Networks), 11

### NEW VALUE command, SQL\*Plus variables, 614-615

### NEXT DAY function, 193, 321

### NEXT SIZE (tables), 255

### non procedural language.

See SQL

### non-equi-joins (tables), 137-139

### nonprocedural, 5

### normal forms, 231-234

#### normalization

- data redundancy, 229
- databases, 244

### normalizing databases, 127, 229, 235, 268

- benefits, 236-237
- data redundancy, 231
- denormalizing, 237-238
- drawbacks, 237
- end user needs, 230-231
- first normal form, 232
- foreign keys, 233
- logical database design, 230
- naming conventions, 234
- normal forms, 231
- primary keys, 233
- raw databases, 229-230
- referential integrity, 235
- second normal form, 233
- third normal form, 234

### NOT logical operator, 73-75

### NOT NULL constraints (on data), 269-271

### NOT NULL keywords, 250

### NOT NULL to NULL, syntax to change, 259

### NOTFOUND [edit, % in front] cursor attribute, 639

### NULL, 55

- databases, 55-57

### null columns, inserting spaces, 289

### NULL to NOT NULL, changing, 260-261

### NULL values, 7

- ‘ ’ (single quotation marks), 290
- of fields, 250-252
- indexes, 379
- inserting, 289-290
- SQL queries (complex), 544-546
- subqueries, 541

### NUMBER data type, 249

### number-defined fields, " (quotation marks), 59

numbers. *See also*  
arithmetic operators

line, 589-590

numeric data types (T-SQL),  
663-664

numeric values, 541-544

## O

object privileges, 424-425

object-orientation, 492

objects

of databases, scripts,  
maintaining, 279

DBA of Oracle data  
dictionary, 452-456

placing in system  
tablespaces, 569-570

obstacles of database  
performance, identifying,  
407-408

ODBC (Open Database  
Connectivity), 16-17, 576

APIs, 16

Applications,  
developing, 576

architecture, 16

connections, creating, 580

Data Sources, 580

OLAP (online analytical  
processing), 397

OLAP (online analytical  
processing) databases,  
397-398

OracleDB, 577

OLLBACK command, 401

OLTP (online transactional  
processing), 397

and batch loads,  
comparing, 398-400

OLTP databases

and OLAP databases,  
comparing, 397

tuning, 397-398

OMMIT command, 401

online analytical processing  
(OLAP), 397-398

OPEN command of  
PROCEDURE section  
(PL/SQL blocks), 636

Open command (File  
menu), 303

Open Database Connectivity.  
*See* ODBC

opening cursors, 473

operations

arithmetic

ABS function, 195

CEIL function, 196

EXP function, 196-197

FLOOR function, 196

functions, 195

LN function, 197-198

LOG function, 197-198

MOD function, 198

POWER function, 199

SIGN function, 199-200

SQRT function,  
200-201

DELETE, tables, 479

INSERT, tables, 479

join, 233

UPDATE, tables, 479

operators

arithmetic, 42-43

/ (division sign), 48-49

= (equal sign), 538

- (minus sign), 46-48

% (modulo sign), 51-53

\* (multiplication sign),  
49-51

+ (plus sign), 43-46

precedence, 53-54

BETWEEN, 78-80

CAST, converting data  
types, 321

character

|| (double pipe)  
concatenation, 67-70

LIKE, 63-65

\_ (underscore), 65-67

comparison, 55-56, 63

= (equal sign), 56-58

> (greater than sign), 59

>= (greater than or  
equal to sign), 59

< or != (inequalities),  
62-63

< (less than sign), 60-61

<= (less than or equal to  
sign), 60-61

FALSE value, 55

NULL, 55-57

TRUE value, 55

UNKNOWN value, 55

DESC, sorting indexes, 378

|| (double pipe), 338

IN, 78-80

values, comparing, 538

logical, 70-71, 693

AND, 71-72

NOT, 73-75

OR, 72-73

OR logical, avoiding in  
queries, 396-397

queries, 42

relational, = (equal  
sign), 159

set, 75

INTERSECT, 77

MINUS (difference),  
77-78

UNION, 75-77

UNION ALL, 75-77

UNION, SELECT  
statement, 343

wildcards (T-SQL),  
679-680

**OR logical operator, 72-73**

avoiding in queries,  
396-397

**Oracle**

- CHECKS table, 25
- data dictionary, 439-440
  - DBA views, 449-450
  - DBA, dynamic performance views, 458-461
  - DBA, growth, 456-457
  - DBA, objects, 452-456
  - DBA, security, 451-452
  - DBA, space allocation, 457-458
  - DBA, users, 450-451
- INFORMATION\_SCH  
EMA, 461-463
- MySQL table
  - commands, 460-461
  - sessions, 458-459
  - user access, 443-449
  - user privileges, 442-443
  - user views, 440
  - users, identifying, 441-442
- DATE data type, 312
- date pictures, 323-324
- Personal
  - applications, developing, 576
  - import/export tools, 305
  - object privileges, 424
  - transactions, beginning (syntax), 356
- Personal Oracle8, queries, 39
- PROMPTs, writing queries, 24
- ROLLBACK statement (syntax), 361
- savepoints, creating (syntax), 364
- SQL\*Plus, 577
- SQL\*Plus commands, 505
- syntax, creating tables, 255

- tables (code examples)
  - creating, 731-738
  - populating, 743-751, 753-755, 757-758, 760-761
- TO CHAR function, 325
- TO DATE function, 326
- transactions, finishing (syntax), 359
- ZERO\_DIVIDE exception, 643
- Oracle (Personal) database security, 416**
  - By password option, 416
  - CASCADE option, 419
  - Connect role, creating, 420
  - DBA roles, 433
    - creating, 421
  - EXTERNALLY option, 416
  - Resource role, creating, 421
  - roles, creating, 419-420
  - synonyms replacing views, 430-432
  - tables
    - creating, 425-426
    - qualifying, 427-429
    - solution to qualifying, 429-430
  - users
    - access to views, 422-429
    - changing, 418
    - creating, 416-419, 426
    - deleting, 419
    - passwords, 417
    - privileges, 421-422
  - views, 429-432
    - solution to qualifying, 429-430
- WITH ADMIN OPTION option, 420, 422

- WITH GRANT OPTION clause, 433-434

**Oracle Corporation**

- SQL\*Plus, 15
- Web site, downloading Personal Oracle8, 15

**Oracle PL/SQL. See PL/SQL****Oracle relational database management system, 414****Oracle SQL**

- cursors,
  - creating, 473
  - scrolling, 474
  - triggers, creating, 479

**Oracle SQL\*Plus, referential integrity reports (constraints on data), 280-283****Oracle, 587. See also SQL\*Plus****Oracle8**

- accounts, creating, 577
- databases, creating, 577-580

**Oracle8 (Personal), 14-15****ORDER BY clause, 89-98, 114-116**

- indexes, 377
- SELECT statement, 343

**order of columns, changing, 29-32****order of constraints (on data), 278-279****ordering columns, 97****ORDERS table, creating, 152-153****OUTER JOINs and INNER JOINs, comparing, 139-143****output**

- queries, 598-599
- SQL statements, 26-27

**output files, editing, 509****output of SQL\*Plus, formatting, 603**

output to users, displaying,  
643-644

overflow (databases), 298

## P

@@pack received  
variable, 669

@@pack sent variable, 669

packages

DBMS OUTPUT, 643

definition, 652

PL/SQL, 652-654

@@packet errors  
variable, 669

PAGESIZE option  
(SQL\*Plus), 602

parent key not found, 561

parent tables, Oracle  
SQL\*Plus referential  
integrity reports, 281-283

parent/child relationships,  
records, deleting, 279

parent/child table  
relationships, 275-276

parentheses ( ), 43

numeric values,  
formatting, 541

subqueries, 155

parentheses (( )),  
columns, 180

PART table, creating,  
152-153

passwords

invalid passwords, 553

MySQL, changing, 688

root, 688

users of Personal  
Oracle, 417

PAY STATUS TABLE, 645,  
647-648

truncating, 647

PAY TABLE, 648-652

PAYEE clause, 112

PAYMENT TABLE, 645, 648

PAYMENTS database, table  
structure, 246

PCT INCREASE view, 448

PCTINCREASE (tables), 255

percent sign (%), 64

numeric values,  
increasing, 542

percent sign (%) wildcard  
operator (T-SQL), 680

performance

of databases, disks to  
enhance, 405-407

indexes, 378

obstacles of databases,  
identifying, 407-408

PERIOD DIFF function, 321

Personal Oracle

applications,  
developing, 576

data types, 250

import/export tools, 305

object privileges, 424

transactions, beginning  
(syntax), 356

Personal Oracle database  
security, 416

BY password option, 416

CASCADE option, 419

Connect role, creating, 420

DBA roles, 433

creating, 421

EXTERNALLY  
option, 416

Resource role,  
creating, 421

roles, creating, 419-420

synonyms replacing views,  
430-432

tables

creating, 425-426

qualifying, 427-429

solution to qualifying,  
429-430

users

access to views,  
422-429

changing, 418

creating, 416-419, 426

deleting, 419

passwords, 417

privileges, 421-422

views, 429-432

solution to qualifying,  
429-430

WITH ADMIN OPTION  
option, 420, 422

WITH GRANT OPTION  
clause, 433-434

Personal Oracle8, 14-15

data types, 249-250

downloading, 15

queries, 39

physical structures,  
databases, 230

pictures, dates, 322-324

pipes, || (double pipe)  
operator, 338

concatenation  
character, 509

PITCHERS table, 666-667

PL/SQL (Oracle), 629, 644

/ (forward slash), 642

and Java stored procedures,  
comparing, 657

block structures. *See* block  
structures (PL/SQL)

blocks, 646-652

executing, 642-643

starting, 642

database queries, 631

DBMS OUTPUT  
package, 643

DDL commands, 631

DML commands, 631

- packages, 652, 654
- procedural language, 630
- programmers, 630
- script files, 646-652
- script files, executing, 643
- SQL statements, 629-630
- stored procedures, 652-654
- tables, data, 645
- transactional control, 644
- triggers, 652, 655-656
- placing columns, 336**
- plus sign (+), joining tables, 143**
- plus sign (+) arithmetic operator, 43-46**
- pointers, indexes, 370**
- populating tables, 333**
  - code examples, 743-755, 757-758, 760-761
  - use database command, 743
  - with INSERT statement, 333
- pound sign (#)**
  - MySQL on UNIX, 687
  - tables, 468
- POWER function, 199**
- precedence, arithmetic operators, 53-54**
- preventing problems with data, 571**
- primary key fields, NULL values, 252**
- primary keys, 246**
  - ALTER TABLE statement, 273
  - constraints (on data), 271-273
  - databases, normalizing, 233
  - field, 246
  - indexes, 382
  - records, 271
  - tables, 232-233, 235
- Print Artists Name procedure, executing, 477**
- PRINT command, T-SQL (Transact-SQL), 671-672**
- private synonyms, 430**
- privileges**
  - CREATE PUBLIC SYNONYM, 517
  - CREATE SESSION, 441-442
  - initial access, MySQL on UNIX-based systems, 688
  - object
    - deleting, 425
    - dropping, 421
    - Personal Oracle, 424
  - PUBLIC keyword, 422
  - roles, 488
  - SELECT\_ANY\_TABLE, 449-450
  - system, 423-424
    - granting, 512-513
  - tables, granting, 514-516
  - user
    - of Oracle data dictionary, 442-443
    - for Personal Oracle, 421-422
    - privileges, INFORMATION\_SCHEMA, 462-463
- problems, modifying data in views, 345**
- procedural languages. See PL/SQL**
- PROCEDURE section (PL/SQL blocks), 631, 635**
  - BEGIN statement, 635
  - CLOSE command, 637
  - conditional statements, 637-640
  - cursor control commands, 635-637
  - DECLARE command, 636
  - END statement, 635
  - FETCH command, 636-637
  - FOR-LOOP, 640
  - IF, THEN statement, 638
  - LOOP, 639
  - loops, 638-640
  - OPEN command, 636
  - WHILE-LOOP, 639-640
- procedures. See also stored procedures**
  - definition, 652
  - non procedural language. *See* SQL
  - queries, 395-396
- processing SQL views, 338-342**
- @@procid variable, 669**
- products. See also Cartesian product**
  - cross-product language (SQL), 12-13
  - database security, 414-415
- programmers, PL/SQL, 630**
- programming applications**
  - .NET, 583-584
  - ISQL (InterBase SQL), 577-580
  - Java, 581-583
  - JDBC, 581
  - Oracle8, 577-580
  - SQL, embedding, 17-18, 575
- programs, flow control (T-SQL), 672**
  - BEGIN statement, 672
  - BREAK command, 677
  - CONTINUE command, 677-678
  - END statement, 672
  - EXISTS keyword, 675
  - FOR loop, 676
  - IF, ELSE statement, 673-675
  - query results, testing, 675-676
  - WHILE loop, 676, 678-679



**prompts, MySQL terminal monitor, 692**

**PROMPTS, writing queries, 24**

**pseudocolumn, SYSDAYE function, 314**

**PUBLIC keyword, 422**

**public synonyms, 430**

CREATE PUBLIC  
SYNONYM  
privilege, 517

in multischema  
databases, 567

## Q

**QA (quality assurance), 568**

**qualifying tables, 519**

for Personal Oracle,  
427-429

**quality assurance (QA), 568**

**QUARTER function, 321**

**queries. See also SQL queries**

aliases, 46

ALL keyword, 33

arithmetic operators, 42-43

/ (division sign), 48-49

% (modulo sign), 51-53

\* (multiplication sign),  
49-51

- (minus sign), 46-48

+ (plus sign), 43-46

precedence, 53-54

BETWEEN operator, 78-80

[ ] (brackets) wildcard  
operator (T-SQL), 680

^ (caret) wildcard operator  
(T-SQL), 680

character operators

LIKE, 63-65

|| (double pipe)  
concatenation, 67-70

\_ (underscore), 65-67

CHECKS table, 25

clauses, 85

AVG(ANNUALLEAVE),  
113-115

AVG(SALARY),  
113-115

AVG(SICKLEAVE),  
113-115

combining, 112

errors, 109

GROUP BY, 98-105,  
116-117

HAVING, 105-111,  
116-117

NAME, 113

ORDER BY, 89-98,  
114-116

PAYEE, 112

REMARKS, 112

SELECT, 100

SELECT statement,  
syntax, 85-86

syntax, 86

TEAM, 113-115

WHERE, 87-88,  
115-116

<column name = alias>  
syntax, 46

columns

individual, selecting, 28

order, changing, 29-32

commands, case  
sensitivity, 22

comparison operators,  
55-56, 63

= (equal sign), 56-58

> (greater than sign), 59

>= (greater than or  
equal to sign), 59

< or != (inequalities),  
62-63

< (less than sign), 60-61

<= (less than or equal to  
sign), 60-61

NULL, 55-57

complex, simplifying with  
views, 347-348

concepts, applying, 25-26

conditions, 40-41

database, PL/SQL, 631

date functions

applying, 312

current dates, 313-314

dates and time periods,  
comparing, 320

dates, subtracting,  
318-320, 713

time, 315-318

dates

current, 313-314

subtracting,  
318-320, 713

and time periods,  
comparing, 320

DISTINCT keyword, 33

elements, arranging,  
393-395

expressions, 40

FROM keyword, 23-24

full-table scans, 371

IN operator, 78-80

indexes, 378

logical operators,  
70-71, 693

AND, 71-72

NOT, 73-75

OR, 72-73

MySQL, 39

NULL, 55-57

operators, 42

OR logical operator,  
avoiding, 396-397

output, 598-599

% (percent sign) wildcard  
operator (T-SQL), 680

Personal Oracle8, 39

procedures, 395-396

PROMPTS, 24

## results

saving with cursors, 472

testing for program flow control, 675-676

SELECT keyword, 23-24

set operators, 75-78

SQL (Structured Query Language), 21-22

SQL statements, 26-28

## subqueries

aggregate functions, 160-161

ALL keyword, 169-174

ANY keyword, 169-174

building, 153-160

correlated, referencing outside, 166-169

embedding, 151-153

EXISTS keyword, 169-174

IN keyword, 173

nesting, 162-165

SOME keyword, 172

syntax, 22-23

tables, selecting, 31

time, 315-318

timing, 383

\_ (underscore) wildcard operator (T-SQL), 679

values, selecting, 32-34

writing, 26-28

### querying data with views, 337

### quotation marks

double (" "), literal strings, 620

number-defined fields, 59

single ( ' '), 290

SQL scripts, 511

## R

raising exceptions, 641

raw databases, 229-230

RAW MLSLABEL data type, 249

RAW(size) data type, 249

RDBMS (relational database management system), 5

Codd, E.F., 8

data dictionaries, 245

DEPENDENTS and EMPLOYEE tables, retrieving fields, 10

SQL3, 487

transaction control, 354-356

## transactions

beginning, 356-359

canceling, 361-364

finishing, 359-361

savepoints, 364-366

READ ONLY option (SET TRANSACTION statement), 356

readability, SQL statements, 390-391

real data type (T-SQL), 664

rebuilding indexes and tables, 402-404

RECORDINGS table, 470

triggers, transactions, 480

## records

DEPENDENTS table, 10

EMPLOYEE table, 8-9

## entering

with INSERT, SELECT statement, 292-294

with INSERT, VALUES statement, 286-289

= (equal sign) comparison operator, 57

examining with cursors, 472

join views, 345

parent/child relationships, deleting, 279

primary keys, 271

redundancies, data, 231

normalization, 229

referencing with correlated subqueries, 166-169

referential integrity, 280

databases, normalizing, 235

of databases, 302

triggers, 479-480

REGEXP, 497

regular expressions, 497-498

relational database management system. *See* RDBMS

## relational databases

Codd's 12 rules, 6-10

dates, 309

DEPENDENTS and EMPLOYEE tables, retrieving fields, 10

DEPENDENTS table, records, 10

EMPLOYEE table, records, 8-9

JOIN, 9

Microsoft Access relational database management system, 414

Oracle relational database management system, 414

tables, 8

time, 309

UNION, 9

"Relational Model of Data for Large Shared Data Banks (A)", 6

relational operators, = (equal sign), 159

REMARK, comments, 624

REMARKS clause, 112

removing views with **DROP VIEW** statement, 350. *See also* deleting

renaming columns, 337-338

**REPLACE** function, 207-209

reports. *See also* **SQL\*Plus**

creating, 624-626

database  
fragmentation, 537

formatting, 604

Oracle **SQL\*Plus**,  
referential integrity  
(constraints on data,  
280-283

summaries

**BREAK ON** command,  
607-608

**COMPUTE** command,  
608-610

creating, 606-610

reserved words, using in  
**SQL** statements (errors),  
564-566

Resource role, creating for  
Personal Oracle, 421

resources

*Implementing Client/Server  
Computing*, 12

“Relational Model of Data  
for Large Shared Data  
Banks (A)”, 6

restrictions

most restrictive condition  
(**WHERE** clause),  
393-395

**SELECT** statement, 343

restrictions on triggers, 481

result sets, 472-474

retrieving

data

from banking  
applications, 355

**FROM** keyword, 23-24

indexes, 378

into local variables, 671

**SELECT** keyword,  
23-24

fields from **DEPENDENTS**  
and **EMPLOYEE**  
tables, 10

**REVOKE** command

object privileges,  
deleting, 425

roles, deleting, 420-421

**RIGHT** function, 221

**RLIKE**, 497

roles

**CONNECT**, 442

Connect, creating for  
Personal Oracle, 420

**CREATE ROLE** statement,  
488-490

creating for Personal  
Oracle, 419-420

**DBA**

creating for Personal  
Oracle, 421

Personal Oracle  
security, 433

defined, 488

deleting, 420

dropping, 421

privileges, granting, 488

Resource, creating for  
Personal Oracle, 421

Security, managing,  
488-489

**ROLLBACK** command

errors, undoing, 292, 298

**PL/SQL** transactional  
control, 644

rollback segments, 401-402

**ROLLBACK** statement

syntax, 361

transactions, canceling,  
361-364

rollbacks, transactions, 361

roots, 688

**%ROWCOUNT** attribute of  
**DECLARE** section (**PL/SQL**  
blocks), 634-635

**@rowcount** global  
variable, 475

**@@rowcount** variable, 669

**ROWID** data type, 250

**ROWID** field, 252

**ROWNUM** view, 453

rows

**COUNT(\*)** function, 507

counting, 507-511

deleting with views, 344

**%ROWTYPE** attribute of  
**DECLARE** section (**PL/SQL**  
blocks), 634

**RPAD** function, 205-206,  
220, 538

**RTRIM** function,  
206-207, 222

**RUN** command, starting  
files, 598

running **SQL** (Structured  
Query Language) files,  
623-624

## S

**SALARIES** tables, 425-427

**SALARY** field (**SALARIES**  
table), updating, 427

**SAVE** command, files, saving,  
595-596

**SAVE newfile** command, 588

**SAVEPOINT** command,  
**PL/SQL** transactional  
control, 644

savepoints, 364-366

**SAVE\_IT** savepoint, 365

saving query results with  
cursors, 472

scans, full-table, 371

avoiding, 391-392

**schemas, truncating tables,**  
**522-523**

**scope of cursors, 475-476**

**screens, ODBC Data  
Sources, 580**

**script files**

PL/SQL, executing, 643  
tables, creating and  
entering data, 577-578

**scripts**

files, creating with EDIT  
command, 623  
maintaining, 279  
Oracle SQL\*Plus  
referential integrity  
reports (constraints on  
data), 280-283  
PL/SQL script file, 646-652  
shell, generating, 523-524  
SQL (Structured Query  
Language)  
comments, adding, 624  
' (single quotation  
mark), 511

**scrolling**

cursors, through result sets,  
473-474  
tables with WHILE loop,  
678-679

**second normal form,  
normalizing  
databases, 233**

**seconds, breaking days into,  
533-535**

**sections**

DECLARE (PL/SQL  
blocks), 631-632  
EXCEPTION (PL/SQL  
blocks), 631, 640  
PROCEDURE (PL/SQL  
blocks), 631, 635

**security**

databases, 413-415  
DBA of Oracle data  
dictionary, 451-452

managing with roles,  
488-489

Personal Oracle, 416

Connect role,  
creating, 420  
DBA role, creating, 421  
DBA roles, 433  
Resource role,  
creating, 421

roles, creating, 419-420

synonyms replacing  
views, 430-432

tables, creating,  
425-426

tables, qualifying,  
427-429

tables, solution to  
qualifying, 429-430

user access to views,  
422-429

user privileges, 421-422

users, changing, 418

users, creating,  
416-419, 426

users, deleting, 419

users, passwords, 417

views, 429-432

views, solution to  
qualifying, 429-430

WITH GRANT  
OPTION clause,  
433-434

with views, 346

**segments, rollback, 401-402**

**SELECT ANY SEQUENCE  
system privilege, 424**

**SELECT ANY TABLE system  
privilege, 424**

**SELECT clause, 100**

**SELECT command, 332, 356**

**SELECT keyword, 23-24, 87**

**SELECT statement, 13, 333**

aggregate functions, 349

and CREATE VIEW  
statement,  
combining, 341

GROUP BY clause, 343

ORDER BY clause, 343

indexes, 377

restrictions, 343

syntax, 85-86

UNION operator, 343

**SELECT statements**

databases, 504

queries, 22

tables

columns, finding,  
128-129

cross joining, 123-128

joining, 121-123

V\$SESSION, 459

**selecting**

columns, 336

query values, 32-34

tables, 31

**SELECT\_ANY\_TABLE  
privilege, 449-450**

**self joins (tables), 143-146**

**self-contained  
subqueries, 166**

**semicolon (;)**

commands, 691

queries, writing, 28

SQL statements, 516, 690

**Sequential Access  
Method, 370**

**@servername variable, 669**

**servers. See also SQL Server**

client/server application  
development, 13

client/server computing, 12

client/server database  
systems, 11

client/server  
development, 11

MySQL, database  
security, 415

Sybase SQL Server,  
database security, 415

**sessions**

- cursors, creating, 476
- Oracle data dictionary, 458-459
- settings, 594-595

**SET AUTOCOMMIT ON**  
command, 358**SET command**, 358, 409**SET commands**

- SQL Server diagnostic tools, 681
- SQL\*Plus work environment, customizing, 599-603

**SET ECHO OFF** command, 505, 509**SET ECHO ON** command, 505, 511**SET FEEDBACK OFF**  
command, 506, 509**SET FEEDBACK OFF**  
commands, 601**SET FEEDBACK ON**  
command, 506**SET HEADING OFF**  
command, 506, 509**SET HEADING ON** command, 506, 511**SET NOCOUNT ON**  
command, 681**SET NOEXEC ON**  
command, 681**set operators**, 75

- INTERSECT, 77
- MINUS (difference), 77-78
- UNION, 75-77
- UNION ALL, 75-77

**SET PARSONLY ON**  
command, 681**SET PASSWORD**  
statement, 688**SET ROLE** command, 356**SET ROWCOUNT n** [edit, n in italics] command, 681**SET SHOWPLAN ON**  
command, 681**SET STATISTICS IO ON**  
command, 681**SET STATISTICS TIME ON**  
command, 681**SET TIMING ON**  
command, 409**SET TRANSACTION**  
statement, 356-358**set-oriented** (sets or groups), 6**shells scripts**, generating, 523-524**SHOW ALL** command, 594**SHOW COLUMNS**  
command, 694**SHOW commands**, 460

- MySQL terminal monitor, 693-694

- session settings, displaying, 594-595

- syntax, 693

**SHOW DATABASES**  
command, 256, 694**SHOW ERROR**  
command, 595**SHOW FIELDS**  
command, 694**SHOW INDEX** command, 694**SHOW MYSQLSHOW —**  
STATUS DB NAME  
command, 694**SHOW MYSQLSHOW DB**  
NAME command, 694**SHOW PROCESSLIST**  
command, 694**SHOW STATUS**  
command, 694**SHOW TABLE STATUS**  
command, 694**SHOW TABLES**  
command, 694**SHOW VARIABLES**  
command, 694**SIGN** function, 199-200**simplifying complex queries**  
with views, 347-348**single quotation marks** ( ' ') 

- character data types, 290
- NULL values, 290
- SQL scripts, 511

**sizes, tables**, 255**sizing tables**, 254-255**slashes, /** (forward)

- PL/SQL, 642
- SQL\*Plus buffer, 590
- table names, 248

**SMALLDATETIME** data  
type, 312**smalldatetime** data type  
(T-SQL), 664**SMALLINT** data type, 249**smallint** data type  
(T-SQL), 663**smallmoney** data types  
(T-SQL), 664**solution to qualifying for**  
Personal Oracle (tables  
and views), 429-430**SOME** keyword, embedding  
subqueries, 172**sorting**

- case, 61
- indexes, DESC  
operator, 378

**sources, importing and**  
exporting foreign data,  
303-305**space allocation, DBA of**  
Oracle data dictionary,  
457-458**spaces between names**, 69**spacing, queries**, 22**specifiers, format strings**,  
223-224**@spid** variable, 669**SPOOL** command, 514

- query output, spooling, 598

**SPOOL FILENAME**  
command, 506**SPOOL OFF** command,  
506, 509

- query output, spooling, 599

### spooling query output, 598-599

### SQL (Structured Query Language), 5, 481. *See* *also* T-SQL

- ANSI extensions, 662-663
- ANSI SQL3 standard, 18
- call-level interfaces, 18
- command-line, 14
- cross-product language,  
12-13
- databases
  - Codd, E.F. (12 rules),  
6-10
  - current technologies,  
11-12
  - history, 6
  - relational, 6-10
- embedded, 481-484
- embedding in application  
programming, 17-18, 575
- files, running, 623-624
- history, 5-6
- IBM DB2, 16
- implementations, 14
- indexes, syntax to  
create, 371
- MySQL, 14
- nonprocedural language,  
5-6
- ODBC, 16-17
- Oracle
  - cursors, creating, 473
  - cursors, scrolling, 474
  - triggers, creating, 479
- overview, 13-14
- Personal Oracle8, 14-15
- queries, 21-22
- RDBMS, 5
- scripts
  - comments, adding, 624
  - ' (single quotation  
mark, 511

- set-oriented (sets or  
groups), 6

- statements

- ; (semicolon), 690

- PL/SQL, 629-630

- Sybase, 15

- view processing, 338-342

### SQL queries (complex)

- building, 546-547

- bytes, converting to  
kilobytes and  
megabytes, 536

- creating, 529

- databases, fragmentation  
reports, 537

- dates, formatting, 538-539

- days, breaking into hours,  
minutes, seconds,  
533-535

- individual's age, computing  
from date of birth,  
532-533

- NULL values, 544-546

- numeric values, 541-544

- subqueries, 540-541

- DML (Data  
Manipulation  
Language), 537

- maximum values,  
539-540

- NULL values, 541

- tables, 529

- CREATE TABLE  
statements, 529-532,  
740-742

### SQL Server

- connections, logging  
off, 471

- CONVERT command, 680

- data functions, 320

- DATENAME function, 325

- diagnostic tools, 681

- import/export tools  
(Microsoft and  
Sybase), 304

- SET commands, 681

- SET NOEXEC ON  
command, 681

- SET PARSONLY ON  
command, 681

- SET SHOWPLAN ON  
command, 681

- SET STATISTICS IO ON  
command, 681

- SET STATISTICS TIME  
ON command, 681

- temporary tables,  
creating, 471

### SQL statements

- ; (semicolon), 516

- batch loads and OLTP  
(online transactional  
processing), comparing,  
398-400

- batch transactions,  
COMMIT statement,  
401-402

- data loads, dropping  
indexes, 400-401

- databases

- EXPLAIN PLAN  
tool, 409

- performance  
enhancement, disks,  
405-407

- performance obstacles,  
identifying, 407-408

- TKPROF tool, 409

- tuning, 405-407

- tuning tools  
(built-in), 409

- full-table scans, avoiding,  
391-392

- generating, 503-505

- indexes

- adding to tables, 393
  - rebuilding, 402-404

- OLAP, 397

- OLAP databases, 397-398

- OLTP, 397

- OLTP databases, 397-398
- output, 26-27
- practical applications, 524-525
- queries
  - elements, arranging, 393-395
  - OR logical operator, avoiding, 396-397
  - procedures, 395-396
- readability, 390-391
- schema tables, truncating, 522-523
- shell scripts, generating, 523-524
- SQL\*Plus buffer, 588
- SQL\*Plus commands, 505
  - ED FILENAME, 507-509
  - SET ECHO OFF, 505, 509
  - SET ECHO ON, 505, 511
  - SET FEEDBACK OFF, 506, 509
  - SET FEEDBACK ON, 506
  - SET HEADING OFF, 506, 509
  - SET HEADING ON, 506, 511
  - SPOOL, 514
  - SPOOL FILENAME, 506
  - SPOOL OFF, 506, 509
  - START, 511
  - START FILENAME, 506
- streamlining, 389-390
- synonyms, creating, 517-520
- system privileges, granting, 512-513
- table constraints, disabling, 516-517
- table privileges, granting, 514-516
- table views, creating, 520-522
- tables
  - rebuilding, 402-404
  - rows, counting, 507-511
- terminating, 28
- transaction logs, 398
- transactions, rollback segments, 401-402
- WHERE clause, 390, 393
- .sql file extension, 597**
- SQL\*Loader dialog box, 305**
- SQL\*Plus, 15, 577, 587**
  - APPEND text command, 588
  - buffer, 588, 591-593
    - \* (asterisk), 590
  - APPEND command, 592
  - CHANGE command, syntax, 590
  - CLEAR BUFFER command, 592
  - CLEAR command, 592
  - commands, 588-589
  - contents, clearing, 592
  - / (forward slash), 590
  - INPUT command, 591
  - line numbers, 589-590
  - lines, 591
  - LIST command, 588
  - SQL statement, 588
  - CHANGE/old value/new value command, 588
  - COLUMN command, 620
  - columns, 605-606
  - commands. *See* SQL\*Plus commands
  - DATE conversions, 619-623
  - date picture, parts of, 620
  - DECODE function, 616-619
  - DEL command, 588
  - DUAL table, 615-616
  - files
    - EDIT command, 595-596
    - EDIT PROD.LST command, 599
    - GET command, 595-596
    - manipulating with file commands, 595
    - query output, spooling, 598-599
    - RUN command, 598
    - SAVE command, 595-596
    - SPOOL command, 598
    - SPOOL OFF command, 599
    - START command, 598
    - starting, 596-598
  - GET filename command, 588
  - groups
    - BREAK ON command, 607-608
    - COMPUTE command, 608-610
    - summaries, creating, 606-610
  - LIST line number command, 588
  - output, formatting, 603
  - reports, 587
    - BREAK ON command, 607-608
    - BTITLE command, 604
    - COMPUTE command, 608-610
    - creating, 624-626
    - formatting, 604
    - summaries, creating, 606-610
    - TTITLE command, 604

- SAVE newfile
  - command, 588
- session settings, displaying, 594-595
- settings, 603
- SQL files, running, 623-624
- SQL scripts, adding
  - comments, 624
- tables, 593-594
- TO CHAR function, 619
- TO DATE function, 622
- variables, 610
  - & (ampersand), 611
  - ACCEPT command, 612-614
  - DEFINE command, 611-612
  - NEW VALUE
    - command, 614-615
- work environment
  - customizing, 599-603
  - FEEDBACK
    - commands, 600
  - LINESIZE option, 601-602
  - PAGESIZE option, 602
  - SET commands, 599-603
  - SET FEEDBACK OFF
    - commands, 601
  - TIME option, 602
- SQL\*Plus commands, 505**
  - ED FILENAME command, 507, 509
  - SET ECHO OFF
    - command, 505, 509
  - SET ECHO ON command, 505, 511
  - SET FEEDBACK OFF
    - command, 506, 509
  - SET FEEDBACK ON
    - command, 506
  - SET HEADING OFF
    - command, 506, 509
  - SET HEADING ON
    - command, 506, 511
  - SPOOL command, 514
  - SPOOL FILENAME
    - command, 506
  - SPOOL OFF command, 506, 509
  - START command, 511
  - START FILENAME
    - command, 506
- SQL3, 487**
  - ALTER TYPE statement, 493-494
  - ANSI, 18
  - CREATE ROLE statement, 488-490
  - CREATE TYPE statement, 492-496
  - DROP statement, 493-494
  - object-orientation, 492
  - RDBMS, 487
  - triggers, creating, 490-492
- SQL3 standard (ANSI), 6**
- SQLBase, 312**
- SQLSTATISTICS**
  - command, 694
- @@sqlstatus global**
  - variable, **cursors (testing status), 475**
- @@sqlstatus variable, 669**
- SQRT function, 200-201**
- standards**
  - ANSI, 86
    - aggregate
      - functions, 179
    - date and time data
      - types, 310-311
  - ANSI SQL3, 6, 18
  - ISO, 6
- START command, 511, 613**
  - files, starting, 598
- START FILENAME**
  - command, 506
- starting**
  - files, 596-598
  - MySQL on UNIX-based
    - systems, 687-688
  - mysqld daemon, 687
  - PL/SQL blocks, 642
- statements. See also SQL statements**
  - % (percent sign), 64
  - ALTER TABLE
    - CHANGE option,
      - syntax, 259
    - constraints, 279
    - primary keys, 273
    - syntax, 258
    - table structures,
      - modifying, 257-261
  - ALTER TYPE, syntax, 493-494
  - BEGIN
    - of PROCEDURE
      - section (PL/SQL blocks), 635
    - program flow
      - control, 672
  - BEGIN
    - TRANSACTION, 480
  - columns, aliases, 133
  - COMMIT
    - batch transactions,
      - 401-402
    - transactions, 359-360
    - transactions,
      - cancelling, 363
  - conditional, of
    - PROCEDURE section
      - (PL/SQL blocks), 637-640
  - CREATE, 241
  - CREATE DATABASE, 242
    - data dictionaries,
      - creating, 244-245
    - data, breaking
      - down, 247
    - database design, 244



- key fields, creating, 246-247
- options, 243
- syntax, 242
- CREATE INDEX, 369, 373-375
  - ALTER TABLE
    - command, 373
  - code, 372
  - MySQL database, 373
  - UNIQUE keyword, 381-382
- CREATE ROLE, 488-490
- CREATE TABLE, 247-248, 269, 529-532, 731-742
  - code example, 248
  - constraints, 279
  - examples, 737
  - field data types, 249-250
  - field names, 249
  - field NULL value, 250-252
  - field, unique, 252-254
  - NOT NULL
    - constraints, 271
  - storage clause, 254-255
  - table names, 248-249
  - tables, 254-257
- CREATE TRIGGER, 491
- CREATE TYPE, 492-496
- CREATE VIEW, 332
  - columns, selecting and placing, 336
- CREATE VIEW:
  - SELECT, 343
- data definition, 241
- data manipulation, 241
- data-manipulation, 285-286
- DEALLOCATE, 475
- DECLARE cursor\_name
  - CURSOR, 473
- DELETE, 285
  - data, deleting, 298-302
  - table views, 345
  - WHERE clause, 300
- DROP, 493-494
- DROP DATABASE, 262-263
- DROP INDEX, 375-376
- DROP TABLE, 261-263
- DROP TRIGGER, 491
- DROP VIEW, 350
- END
  - of PROCEDURE
    - section (PL/SQL blocks), 635
  - program flow
    - control, 672
- EXECUTE, executing
  - Print Artists Name
    - procedure, 477
- GRANT, 512-513
  - syntax, 433
  - tables, 514
  - WITH GRANT
    - OPTION clause, 433-434
- IF, ELSE, program flow
  - control, 673-675
- IF, THEN, 638
- INSERT, 285
  - data, entering, 286
  - date datatype format, 748, 752
  - examples, 36, 118
  - NULL values, inserting, 289-290
  - records, entering with
    - INSERT, SELECT
      - statement, 292-294
  - records, entering with
    - INSERT, VALUES
      - statement, 286-289
  - tables, populating, 333, 743-755, 757-758, 760-761
  - unique values, inserting, 291-292
  - views, 345
- INSERT, SELECT, 292-294
- INSERT, VALUES
  - DELETE, 298
  - records, entering, 286-289
  - syntax, 286
- JOIN, 121
- MODIFY clause, 261
- number-defined fields, "
  - (quotation marks), 59
- ROLLBACK, 361-364
- SELECT, 13, 333
  - aggregate
    - functions, 349
  - columns, finding, 128-129
  - databases, 504
  - GROUP BY clause, 343
  - ORDER BY clause, 343
  - indexes, 377
  - queries, 22
  - restrictions, 343
  - syntax, 85-86
  - tables, cross joining, 123-128
  - tables, joining, 121-123
  - UNION operator, 343
  - V\$SESSION, 459
- SELECT and CREATE
  - VIEW, combining, 341
- SET PASSWORD, 688
- SET TRANSACTION, 356-358
- SQL
  - ED FILENAME
    - command, 507, 509
  - generating, 503-505
  - output, 26-27
  - PL/SQL, 629-630

- practical applications, 524-525
- schema tables,
  - truncating, 522-523
- ;(semicolon), 516, 690
- SET ECHO OFF
  - command, 505, 509
- SET ECHO ON
  - command, 505, 511
- SET FEEDBACK OFF
  - command, 506, 509
- SET FEEDBACK ON
  - command, 506
- SET HEADING OFF
  - command, 506, 509
- SET HEADING ON
  - command, 506, 511
- shell scripts, generating, 523-524
- SPOOL command, 514
- SPOOL FILENAME
  - command, 506
- SPOOL OFF command, 506, 509
- SQL\*Plus buffer, 588
- SQL\*Plus
  - commands, 505
- START command, 511
- START FILENAME
  - command, 506
- synonyms, creating, 517-520
- system privileges,
  - granting, 512-513
- table constraints,
  - disabling, 516-517
- table privileges,
  - granting, 514-516
- table rows, counting, 507-511
- table views, creating, 520-522
- terminating, 28
- syntax, 86
- tables, aliases, 133

- UPDATE, 285
  - data, modifying, 295-298
  - syntax, 295
  - WHERE clause, 295
- Static SQL (Structured Query Language), 482-483**
  - in C functions, 483-484
- statistics, user statistics tables, 438**
- status of cursors, testing, 475**
- STDDEV function, 186-187**
- stopping**
  - MySQL on UNIX-based systems, 687-688
  - MySQL commands, 691
  - mysqld daemon, 688
- storage clauses in CREATE TABLE statement, 254-255**
- stored procedures**
  - creating, 476-478
  - cursors, creating, 476
  - deleting, 478-479
  - DROP command, 478
  - PL/SQL, 652-654
    - and Java, comparing, 657
  - Print Artists Name
    - procedure, executing, 477
- storing**
  - data, case sensitivity, 58
  - date values, 310
  - dates, 326
    - dates of birth, 320
  - indexes, 379
  - tables, 379
  - time values, 310
- storing tables, 254-255**
- streamlining SQL statements, 389-390**

- strings**
  - binary strings data type (T-SQL), 664
  - character
    - converting to dates, 326
    - data types (T-SQL), 663
  - connection, converting to dates, 325
  - format, specifiers, 223-224
  - literal, “ ” (double quotation marks), 620
  - || (double pipe)
    - concatenation character operator, 67-70
- Structured Query Language. See SQL**
- structures**
  - databases, 230
  - of tables
    - modifying, 257-261
  - PAYMENTS
    - database, 246
  - PL/SQL blocks, 630-631
    - DECLARE section, 631-632
    - EXCEPTION section, 631, 640
    - PROCEDURE section, 631, 635
  - SALARIES table, 425
  - tables, 593-594
  - USER tables, 417-418
- subqueries, 540-541**
  - aggregate functions, 160-161
  - ALL keyword, 169-174
  - ANY keyword, 169-174
  - building, 153-160
  - correlated, 166-169
  - DML (Data Manipulation Language), 537
  - embedding, 151-153
  - = (equal sign) relational operator, 159

- EXISTS keyword, 169-174
- IN keyword, 173
- maximum values, 539-540
- nesting, 162-165
- NULL values, 541
- parentheses ( ), 155
- self-contained, 166
- SOME keyword, 172
- SUBSTR function, 209-213**
- SUBSTR/MID function, 213**
- SUBSTRING function, 221**
- subtracting dates, 318-320, 713**
- SUM function, 181-182, 341**
  - groups, 608
  - subqueries, embedding, 160-161
- summaries**
  - of groups
    - BREAK ON command, 607-608
    - COMPUTE command, 608-610
    - creating, 606-610
  - of reports
    - BREAK ON command, 607-608
    - COMPUTE command, 608-610
    - creating, 606-610
- summarizing data**
  - aggregate functions, 180
  - AVG function, 182-183
  - COUNT function, 180-181
  - from tables, 349
  - MAX function, 184
  - MIN function, 185
  - STDDEV function, 186-187
  - SUM function, 181-182
  - VARIANCE function, 186
- Sybase, 15**
  - DATETIME data type, 312
  - savepoints, creating (syntax), 364
  - SMALLDATETIME data type, 312
  - SQL Server, import/export tools, 304
- Sybase SQL Server, database security, 415**
- synonyms**
  - CREATE PUBLIC SYNONYM, 517
  - creating, 517-520
  - dropping, 432
  - private, 430
  - public, 430
    - CREATE PUBLIC SYNONYM privilege, 517
  - public synonyms in multischema databases, 567
- synonyms replacing views, Personal Oracle security, 430-432**
- syntax**
  - ALTER TABLE statement, 258
    - CHANGE option, 259
  - ALTER TYPE statement, 493-494
  - CHANGE command, 590
  - <column name = alias> syntax, 46
  - columns
    - changing from NOT NULL to NULL, 259
    - changing from NULL to NOT NULL, 260-261
  - CREATE statements, 241
  - CREATE TABLE statement, examples, 737
  - CREATE TRIGGER statement, 491
  - CREATE TYPE statement, 493-494
  - CREATE VIEW statement, 332
  - DEALLOCATE statement, 475
  - diagrams, 86
  - DROP DATABASE statement, 262
  - DROP statement, 493-494
  - DROP TABLE statement, 261
  - DROP TRIGGER statement, 491
  - GRANT statement, 433
  - INSERT, SELECT statement, 293
  - INSERT, VALUES statement, 286
  - JOIN ON, 140
  - Oracle
    - ROLLBACK statement, 361
    - savepoints, creating, 364
    - tables, creating, 255
    - transactions, finishing, 359
  - Oracle SQL
    - cursors, creating, 473
    - triggers, creating, 479
  - Personal Oracle, transactions, 356
  - public synonyms, 430
  - queries, 22-23
  - SELECT statement, 85-86
  - SHOW commands, 693
  - SQL, creating indexes, 371
  - statements, 86
  - Sybase, creating savepoints, 364
  - synonyms, dropping, 432

**T-SQL**

cursors, 473

triggers, 479

UPDATE statement, 295

WHERE clause, 41-42

**syntax 1, creating temporary tables, 468, 471****SYSDATE function, 193-195, 314**

pseudocolumn, 314

**system catalog. See data dictionaries****system engineers, data dictionaries, 439****system privileges, granting, 512-513****system resources, failure to budget, 570****SYSTEM tablespace, 454****system tablespaces, placing objects in, 569-570****systems, 423-424**

client/server computing, 12

client/server database, 11

client/server

development, 11

resource contentions, 399

**T****T-SQL (Transact-SQL), 473, 661**

[ ] (brackets) wildcard operator, 680

^ (caret) wildcard operator, 680

@ @char convert variable, 668

@ @client csid variable, 668

@ @client csnam variable, 668

@ @connections variable, 669

@ @cpu busy variable, 669

@ @error variable, 669

@ @identity variable, 669

@ @idle variable, 669

@ @io busy variable, 669

@ @isolation variable, 669

@ @language variable, 669

@ @languid variable, 669

@ @max connections variable, 669

@ @maxcharlen variable, 669

@ @ncharsize variable, 669

@ @nestlevel variable, 669

@ @pack received variable, 669

@ @pack sent variable, 669

@ @packet errors variable, 669

% (percent sign) wildcard operator, 680

@ @procid variable, 669

@ @rowcount variable, 669

@ @servername variable, 669

@ @spid variable, 669

@ @sqlstatus variable, 669

@ @textsize variable, 669

@ @thresh hysteresis variable, 669

@ @timeticks variable, 669

@ @total errors variable, 669

@ @total read variable, 669

@ @total write variable, 669

@ @tranchained variable, 669

@ @trancount variable, 669

@ @translate variable, 669

@ @version variable, 669

ANSI SQL extensions, 662-663

BASEBALL database, 665-668

components, 662-663

cursors

creating, 473

status, testing, 475

data types, 663-665

database access, 665

date conversions, 680-681

FETCH command,

scrolling cursors, 473-474

PRINT command, 671-672

program flow control, 672

BEGIN statement, 672

BREAK command, 677

CONTINUE command, 677-678

END statement, 672

EXISTS keyword, 675

FOR loop, 676

IF, ELSE statement, 673-675

query results, testing, 675-676

WHILE loop, 676-679

SET NOCOUNT ON command, 681

SET ROWCOUNT *n* command, 681

SQL Server, 681

stored procedures, creating, 477

triggers, creating, 479

\_ (underscore) wildcard operator, 679

users, 662

variables, 668-671

wildcard operators, 679-680

**table commands, MySQL, 460-461**

**tables. See also columns**

- aliases, 133, 157, 391
- ALTER TABLE command,
  - constraints (on data), 275-276
- ALTER TABLE statement
  - CHANGE option,
    - syntax, 259
  - table structures,
    - modifying, 257-261
- ARTISTS, 469
  - cursors, creating, 473
- backing up, 523
- BALANCES, transaction
  - control, 355
- BANK ACCOUNTS, data, 254, 334
- base, 135
- BATTERS, 666
- BILLS
  - ACCOUNT ID field,
    - creating indexes, 373-377
  - AMOUNT field,
    - creating indexes, 377-378
  - data, 253-254, 333
  - data breakdown, 247
- CASE tools, 245
- CHECKS, 25
- child, Oracle SQL\*Plus
  - referential integrity reports, 280-281
- columns
  - changing from NOT NULL to NULL,
    - syntax, 259
  - changing from NULL to NOT NULL to NULL, 260-261
  - finding, 128-129
  - lengths, increasing or decreasing, 258
  - names, 336

- in relational
  - databases, 8
- renaming, 45, 337-338
- selecting and
  - placing, 336
- COMPANY, data, 254, 334
- composite indexes, 393
- constraints, disabling, 516-517
- COUNT(\*) function, 507
- CREATE DATABASE
  - statement, 242
  - data dictionaries,
    - creating, 244-245
  - data, breaking
    - down, 247
  - database design, 244
  - key fields, creating, 246-247
  - options, 243
  - syntax, 242
- CREATE statements, 241
- CREATE TABLE
  - command, 255, 335
- CREATE TABLE
  - statement, 247-248
  - code example, 248
  - field data types,
    - 249-250
  - field names, 249
  - field NULL value,
    - 250-252
  - fields (unique), 252-254
  - storage clause, 254-255
  - table names, 248-249
  - tables, creating,
    - 255-257
  - tables, storing and
    - sizing, 254-255
- CREATE TABLE statement
  - examples, 737
- create table statements, 269, 731-738

- creating, 241, 255-257, 495-496, 577
  - code examples, 731-738
  - for Personal Oracle, 425-426
- creating and
  - populating, 333
- CUSTOMERS, transaction
  - control, 354
- data
  - entering, 577
  - inserting with
    - INSERT, VALUES statement, 287
  - summarizing with
    - views, 349
- data definition
  - statements, 241
- data dictionary, 437
- data manipulation
  - statements, 241
- data, merging, 233
- DBMS, 241
- defragmenting, 403
- DELETE operation,
  - triggers, 479
- deleting, 262
- DEPENDENTS,
  - records, 10
- DEPENDENTS and
  - EMPLOYEE, retrieving fields, 10
- DESCRIBE command, 645
- disk space, 244
- driving, 135
- DROP DATABASE
  - statement, 262-263
- DROP TABLE
  - command, 262
- DROP TABLE statement, 261-263
- dropping, 377
- DUAL, 314, 615-616

- EMPLOYEE
  - records, 8-9
  - SELECT statement, 13
- EMPLOYEE and DEPENDENTS, retrieving fields, 10
- = (equal sign), tables, 137
- fields in relational databases, 8
- first normal form, 232
- foreign key field, 246
- foreign keys, 233-236, 246
  - constraints (on data), 274
- full-table scans, avoiding, 391-392
- GRANT statements, 514
- indexes, 370-371
  - adding, 393
  - creating, 369-377
    - Direct Access Method, 370
  - dropping, 375-376
  - full-table scans, 371
  - Sequential Access Method, 370
  - storing, 379
- INITIAL SIZE, 255
- INSERT command, 335
- INSERT operation, triggers, 479
- INSERT statements, 345, 743-755, 757-758, 760-761
- JOIN ON syntax, 140
- join operations, 233
- joining, 121
  - columns, finding, 128-129
  - cross joining (Cartesian product), 123-128
  - equi-joins (equality), 129-137
    - in SELECT statements, 121-123
    - non-equi-joins (non-equality), 137-139
  - OUTER JOINS and INNER JOINS, comparing, 139-143
  - self joins, 143-146
  - WHERE clause, 126
- lists, viewing, 256-257
- lookup, 292
- mailing list, code, 403-404
- maintaining, 241
- MAX HITS, dropping, 261
- MAXEXTENTS, 255
- MEDIA, 470
- MINEXTENTS, 255
- names, 248-249
  - conflicts, 635
- naming conventions, 234
- NEXT SIZE, 255
- Oracle syntax, 255
- ORDERS, creating, 152-153
- Parent, Oracle SQL\*Plus referential integrity reports, 281-283
- parent/child, relationships, 275-276
- PART, creating, 152-153
- PAY STATUS TABLE, 645, 647-648
  - truncating, 647
- PAY TABLE, 648-652
- PAYMENT TABLE, 645, 648
- PCTINCREASE, 255
- PITCHERS, 666-667
- PL/SQL, 645
- + (plus sign), tables, 143
- populating
  - code examples, 743-755, 757-758, 760-761
  - use database command, 743
  - with INSERT statement, 333
- # (pound sign), 468
- primary key field, 246
- primary keys, 232-233, 235, 246
- privileges, granting, 514-516
- qualifying, 519
  - for Personal Oracle, 427-429
- rebuilding, 402-404
- RECORDINGS, 470
  - triggers and transactions, 480
- records, join views, 345
- referential integrity, triggers, 479-480
- rows
  - counting, 507-511
  - deleting with views, 344
- SALARIES, 425-427
- schema, truncating, 522-523
- scripts, maintaining, 279
- scrolling with WHILE loop, 678-679
- second normal form, 233
- selecting, 31
- sizes of, 392
- sizing, 254-255
- solution to qualifying, for Personal Oracle, 429-430
- SQL queries (complex), 529
  - CREATE TABLE statements, 529-532, 740-742
- storing, 254-255

- structure
  - modifying, 257-261
  - PAYMENTS database, 246
  - viewing with DESC command, 593
  - viewing with DESCRIBE command, 593-594
- SUM function, 341
- TEAMS, 667-668
- temporary, 292
  - creating, 468-470
  - SQL Server, creating in, 471
  - syntax 1, creating with, 468, 471
  - TEMPDB database, creating in, 471
- third normal form, 234
- TRANSACTION, 655
- Triggers, 479-481
- truncating, backing up, 523
- unqualified tables, dropping, 566-567
- UPDATE operation, triggers, 479
- USER, structure, 417-418
- user statistics, 438
- views, 335
  - creating, 520-522
  - DELETE statements, 345
- virtual (views), 331
- tablespaces**
  - dropping into databases, 523-524
  - SYSTEM, 454
  - USERS, 441, 454
- TAR command, 687**
- TEAM clause, 113-115**
- TEAMS table, 667-668**
- technologies, databases, 11-12**
- TEMPDB database, creating temporary tables, 471**
- temporary tables, 292**
  - creating, 468-471
- terminal monitor, MySQL on UNIX-based systems, 688**
  - batch mode, 692-693
  - command-line history, 692
  - command-line options, 689-690
  - commands, entering, 690-692
  - databases, connecting to, 689
  - disconnecting from, 689
  - prompts, changes, 692
  - SHOW commands, 693-694
- terminating SQL statements, 28**
- testing**
  - cursors, status of, 474-475
  - query results, for program flow control, 675-676
- text data type (T-SQL), 663**
- text files, delimited (exporting), 303**
- @@textsize variable, 669**
- third normal form, normalizing databases, 234**
- third-generation languages (3GLs), 5**
- @@thresh hysteresis variable, 669**
- time, 309. See also date and time functions**
  - adding to dates, 315-318
  - ANSI standard data types, 310-311
  - data types, implementing, 312
  - DATETIME data type, 309-310
  - DATETIME elements, 311
  - days, breaking into hours, minutes, seconds, 533-535
  - values, storing, 310
- TIME data type, 310-312**
- TIME FORMAT function, 224**
- TIME option (SQL\*Plus), 602**
- time periods and dates, comparing, 320**
- time zones, 315**
- TIMESTAMP data type, 310-312**
- @@timeticks variable, 669**
- timing queries, 383**
- tinyint data type (T-SQL), 663**
- TKPROF tool, 409**
- TNS: Listener could not resolve SID given in connect descriptor, 563**
- TO CHAR function, 215-217, 325**
  - dates, 619
- TO DATE function, 622**
- TO NUMBER function, 217**
- tools. See also development tools for applications**
  - BCP (bulk copy), 304
  - built-in tools, database tuning, 409
  - CASE (computer-aided software engineering), 245, 439
  - diagnostic, SQL Server, 681
  - EXPLAIN PLAN, 409
  - import/export, 303-305
  - TKPROF, 409
  - WinMySQLadmom, 440
- TO\_CHAR function, 544**
- @@total errors variable, 669**

**@@total read variable, 669**

**@@total write variable, 669**

**@@tranchained variable, 669**

**@@trancount variable, 669**

**Transact-SQL. See T-SQL**

**transaction control (of databases), 354**

banking applications,  
354-356

transactions

beginning, 356-359

canceling, 361-364

finishing, 359-361

savepoints, 364-366

**transaction logs, 398**

**transaction management.**

**See transaction control**

**TRANSACTION table, 655**

**transactional control  
(PL/SQL), 644**

**transactions**

batch, COMMIT statement,  
401-402

BEGIN TRANSACTION  
statement, 480

beginning, 356

COMMIT command,  
359-360

COMMIT statement,  
359-360, 363

COMMIT WORK  
command, 360

commit, 364

controlling, 353

finishing, 359

nested, 357-358

rollback segments, 401-402

ROLLBACK statement,  
361-364

rollbacks, 361

savepoints, 364-365

SAVE\_IT savepoint, 365

**SET TRANSACTION**

statement, 356-358

triggers, 480

unit of work, 354

**TRANSLATE function,  
213-214**

**@@translate variable, 669**

**trees, 370-371**

**triggers**

CREATE TRIGGER

statement, 491

creating, 490-492

Oracle SQL syntax, 479

T-SQL syntax, 479

cursors, creating, 476

definition, 652

DELETE event, 490

designing, 479

DML commands, 490

DROP TRIGGER

statement, 491

events, 490

INSERT event, 490

nesting, 481

parts of, 490

PL/SQL, 652, 655-656

referential integrity,  
479-480

restrictions, 481

tables, 479

transactions, 480

UPDATE event, 490

**TRIM function, 222**

**troubleshooting errors**

allowing large tables to  
take default storage  
parameters, 569

cannot create operating  
system files, 564

Cartesian product, 567-568

columns ambiguously  
defined, 558-559

commands not properly  
ended, 559

dropping unqualified tables,  
566-567

escape character in your  
statement—invalid  
character, 564

failure to budget system  
resources, 570

failure to compress large  
backup files, 570

failure to enforce file  
system structure  
conventions, 568

failure to enforce input  
standards, 568

FROM keyword not  
specified, 553-554

group function not allowed,  
554-555

inserted value too large for  
column, 562-563

insufficient privileges  
during grants, 563-564

integrity constraints  
violated—parent key not  
found, 561

invalid column names,  
555-556

invalid usernames or  
passwords, 553

missing commas, 558

missing expressions,  
559-560

missing keywords, 556

missing left parenthesis,  
556-557

missing right parenthesis,  
557-558

not enough arguments for  
function, 560

not enough values, 560-561

Oracle not available, 562

placing objects in the  
system tablespace,  
569-570



- public synonyms in multischema databases, 567
  - table or view that does not exist, 552
  - TNS: Listener could not resolve SID given in connector descriptor, 563
  - use of DISTINCT when selecting multiple columns, 566
  - using reserved words in your SQL statement, 564-566
  - TRUE value, comparison operators, 55**
  - TRUNC command, 533**
  - TRUNC function, 319**
  - truncating**
    - PAY STATUS table, 647
    - schema tables, 522-523
    - tables, backing up, 523
  - truncation (databases), 298**
  - TTITLE command, formatting reports, 604**
  - tuning**
    - databases, 405-407
      - tools (built-in), 409
    - OLAP databases, 398
    - OLTP databases, 397-398
  - %TYPE attribute of DECLARE section (PL/SQL blocks), 633-634**
  - types. See also data types**
    - ALTER TYPE statement, 493-494
    - CREATE TYPE statement, 492-496
    - of constraints (on data), 269
      - check, 276-277
      - foreign key, 274-276
      - NOT NULL, 269-271
    - parent/child table relationships, 275-276
    - primary key, 271-273
    - unique, 273-274
- ## U
- UDTs (User Defined Types), 493**
    - attributes, 495
    - creating, 494-495
    - tables, creating, 495-496
  - underscore (\_) character operator, 65-67**
  - underscore (\_) wildcard operator (T-SQL), 679**
  - UNION, relational databases, 9**
  - UNION ALL set operator, 75-77**
  - UNION operator, SELECT statement, 343**
  - UNION set operator, 75-77**
  - unions, relational databases, 9**
  - unique constraints (on data), 273-274**
  - unique fields, 252-254**
  - UNIQUE keyword, CREATE INDEX statement, 381-382**
  - unique values, inserting, 291-292**
  - unit of work (transactions), 354**
  - units, converting with views, 346-347**
  - UNIX, MySQL, 685**
    - administration, 686
    - initial access privileges, 688
    - installing, 686-687
    - isamchk utility, 694
    - mysqlaccess utility, 694
    - mysqladmin utility, 694
    - mysqldump utility, 694
    - mysqlimport utility, 694
    - mysqlshow utility, 694
    - # (pound sign), 687
    - starting, 687-688
    - stopping, 687-688
    - terminal monitor, 688-694
    - utilities, 694
  - UNKNOWN value, comparison operators, 55**
  - unqualified tables, dropping, 566-567**
  - UPDATE ANY TABLE system privilege, 424**
  - UPDATE command, 332, 418, 440**
    - views, 343-344
  - UPDATE event, 490**
  - UPDATE function, views, 345**
  - UPDATE operation, tables (triggers), 479**
  - UPDATE statement, 285**
    - data, modifying, 295-298
    - syntax, 295
    - WHERE clause, 295
  - updating**
    - data from banking applications, 355
    - virtual columns, 345
  - UPPER function, 203-205**
  - use database command, populating tables, 743**
  - USE ROLLBACK SEGMENT option (SET TRANSACTION statement), 357**
  - usefulness of constraints (on data), 268**
  - USER CATALOG view, 453**
  - User Defined Types. See UDTs**

**USER function, 218-219**  
**user privileges,**  
**INFORMATION\_SCHEMA,**  
**462-463**  
**user roots (MySQL), 688**  
**USER SEGMENTS view, 446**  
**user statistics tables, 438**  
**USER tables, structure,**  
**417-418**  
**USER TABLESPACES**  
**view, 448**  
**USER TS QUOTAS view, 448**  
**usernames, invalid user**  
**names, 553**  
**users**  
    access to views for  
        Personal Oracle, 422-429  
    ALTER USER  
        command, 418  
    changing for Personal  
        Oracle, 418  
    creating for Personal  
        Oracle, 416-419, 426  
    database design, 230-231  
    DBA of Oracle data  
        dictionary, 450-451  
    defined, 419  
    DELETE command, 418  
    deleting from Personal  
        Oracle, 419  
    DROP USER  
        command, 419  
    IDENTIFIED BY  
        clause, 417  
    identifying for data  
        dictionary, 438-439  
    MySQL anonymous, 688  
    of Oracle data dictionary  
        access, 443-449  
        identifying, 441-442  
        privileges, 442-443  
        views, 440  
    output, displaying, 643-644  
    passwords for Personal  
        Oracle, 417

privileges for Personal  
    Oracle, 421-422  
system privileges, granting,  
    512-513  
T-SQL (Transact-  
SQL), 662  
table constraints, disabling,  
    516-517  
table privileges, granting,  
    514-516  
tables, qualifying, 519  
UPDATE command, 418  
**USERS tablespace, 441, 454**  
**USER\_CATALOG view,**  
**443-444**  
**USER\_OBJECTS view, 445**  
**USER\_ROLE\_PRIVS view,**  
**442-443**  
**USER\_SYS\_PRIVS view, 442**  
**USER\_TABLES view, 445**  
**USER\_USERS view, 441**  
**utilities, MySQL on UNIX-**  
**based systems, 694**

## V

### V\$SESSION, SELECT statements, 459

#### values

comparing, 538  
dates, storing, 310  
DATETIME, 315  
decimal, deleting, 533  
FALSE, comparison  
    operators, 55  
maximum, subqueries,  
    539-540  
NULL  
    of fields, 250-252  
    indexes, 379  
    inserting, 289-290  
    ‘ ’ (single quotation  
        marks), 290

SQL queries (complex),  
    544-546  
    subqueries, 541  
numeric, 541-544  
of queries, selecting, 32-34  
time, storing, 310  
TRUE, comparison  
    operators, 55  
unique, inserting, 291-292  
UNKNOWN, comparison  
    operators, 55  
**VARCHAR data type, 250**  
**varchar data type**  
**(T-SQL), 663**  
**VARCHAR2(size) data**  
**type, 250**  
**variables**  
    @ (at symbol), 668  
    @@char convert  
        variable, 668  
    @@client csid  
        variable, 668  
    @@client csname  
        variable, 668  
    @@connections  
        variable, 669  
    @@cpu busy variable, 669  
    @@error variable, 669  
    @@identity variable, 669  
    @@idle variable, 669  
    @@io busy variable, 669  
    @@isolation variable, 669  
    @@language variable, 669  
    @@languid variable, 669  
    @@max connections  
        variable, 669  
    @@maxcharlen  
        variable, 669  
    @@ncharsize variable, 669  
    @@nestlevel variable, 669  
    @@pack received  
        variable, 669  
    @@pack sent variable, 669

- @@packet errors variable, 669
- @@procid variable, 669
- @@rowcount
  - cursors, testing status, 475
- @@rowcount variable, 669
- @@servername variable, 669
- @@spid variable, 669
- @@sqlstatus
  - cursors, testing status, 475
- @@sqlstatus variable, 669
- @@ symbol, 668
- @@textsize variable, 669
- @@thresh hysteresis variable, 669
- @@timeticks variable, 669
- @@total errors variable, 669
- @@total read variable, 669
- @@total write variable, 669
- @@tranchained variable, 669
- @@trancount variable, 669
- @@translate variable, 669
- @@version variable, 669
- data, storing, 670
- DECLARE keyword, 670
- of DECLARE section (PL/SQL blocks), assigning, 632-633
- declaring, table name conflicts, 635
- global variables, declaring, 668-669
- local
  - data, retrieving, 671
  - declaring, 668
- SQL\*Plus, 610
  - ACCEPT command, 612-614
  - & (ampersand), 611
  - DEFINE command, 611-612
  - NEW VALUE command, 614-615
  - T-SQL, 671
- VARIANCE function, 186**
- @@version variable, 669**
- viewing**
  - table lists, 256-257
  - table structure, 593-594
- views, 332**
  - ALL TAB PRIVS, 447
  - ALL TABLES, 445-446
  - ALL USERS, 441
  - ALL\_CATALOG, 444
  - BANK ACCOUNTS table, 334
  - BILLS table, 333
  - columns
    - names, 336
    - renaming, 337-338
  - COMPANY table, 334
  - complex queries, simplifying, 347-348
  - CREATE VIEW statement, 332
    - columns, selecting and placing, 336
    - syntax, 332
  - CREATE VIEW[colon]SELECT statement, 343
  - creating, 331, 336
    - with DISTINCT clause, 345
  - data
    - filtering, 336
    - modifying, 343-345
    - modifying, problems, 345
    - querying, 337
    - summarizing from tables, 349
  - DBA (database administrator) of Oracle data dictionary, 449-450
  - DBA CATALOG, 453
  - DBA DATA FILES, 457
  - DBA EXTENTS, 457
  - DBA INDEXES, 454
  - DBA ROLE PRIVS, 451
  - DBA ROLES, 451
  - DBA SEGMENTS, 456-457
  - DBA SYS PRIVS, 451-452
  - DBA TABLES, 453
  - DBA TABLESPACES, 455
  - DBA USERS, 451
  - DELETE command, 332, 343
  - DROP VIEW command, 350
  - dropping, 350
  - dynamic performance, DBA of Oracle data dictionary, 458-461
  - exploring, 335-337
  - indexes, creating, 379
  - INSERT command, 332, 343
  - INSERT statements, 345
  - join, handling records, 345
  - PCT INCREASE, 448
  - removing with DROP VIEW statement, 350
  - replacing with synonyms, Personal Oracle security, 430-432
  - ROWNUM, 453
  - rows, deleting, 344

- security
  - for Personal Oracle, 429-432
  - providing, 346
- SELECT command, 332
- SELECT statement, 333
  - aggregate functions, 349
  - restrictions, 343
- solution to qualifying for Personal Oracle, 429-430
- SQL view processing, 338-342
- tables, 335
  - creating, 520-522
  - DELETE statements, 345
  - populating, 333
- tasks performed, 346
- units, converting, 346-347
- UPDATE command, 332, 343-344
- UPDATE function, 345
- user access, for Personal Oracle, 422-429
- USER CATALOG, 453
- USER OBJECTS, 445
- USER ROLE PRIVS, 442-443
- USER SEGMENTS, 446
- USER SYS PRIVS, 442
- USER TABLES, 445
- USER TABLESPACES, 448
- USER TS QUOTAS, 448
- USER USERS, 441
- users of Oracle data dictionary, 440
- USER\_CATALOG, 443-444
- virtual columns, updating, 345
- virtual tables, 331

- || (double pipe) operator, 338

- virtual columns, updating, 345

- virtual tables (views), 331

## W

### Web sites, downloading

- MySQL, 14, 686
- MySQL 3.23, 686
- Oracle Corporation, Personal Oracle8, 15
- Personal Oracle8, 15

- WHEN OTHERS** command, 642

- WHERE clause, 87-88, 115-116**

- DELETE statement, 300
- most restrictive condition, 393-395
- SQL statements, 390, 393
- syntax, 41-42
- tables, joining, 126
- UPDATE statement, 295

### WHILE loop

- program flow control, 676
- tables, scrolling, 678-679

- WHILE-LOOP, conditional statements, 639-640**

- wildcard operators (T-SQL), 679-680

- wildcards, \_ (underscore) character operator, 65-67

- Windows, installation instructions, 763-764

- WinMySQLadmin tool, 440

- WITH ADMIN OPTION** option, Personal Oracle database security, 420, 422

- WITH GRANT OPTION** clause, Personal Oracle security, 433-434

### work environment of SQL\*Plus

- customizing, 599-603
- FEEDBACK commands, 600
- LINESIZE option, 601-602
- PAGESIZE option, 602
- SET commands, 599-603
- SET FEEDBACK OFF commands, 601
- TIME option, 602

### writing queries, 26

- asterisk (\*), 26-27
- columns
  - individual, selecting, 28
  - order, changing, 29-32
- semicolon (;), 28
- SQL statements, 26-28
- Tables, selecting, 31

## X-Y-Z

- XML (Extensible Markup Language), 499-500**

- YEAR** data type, 312

- ZERO\_DIVIDE** exception, 643

- zones, time zones, 315**