7-1 Introduction to SQL

Ideally, a database language allows you to create database and table structures, perform basic data management chores (add, delete, and modify), and perform complex queries designed to transform the raw data into useful information. Moreover, a database language must perform such basic functions with minimal user effort, and its command structure and syntax must be easy to learn. Finally, it must be portable; that is, it must conform to some basic standard, so a person does not have to relearn the basics when moving from one RDBMS to another. SQL meets those ideal database language requirements well.

SQL functions fit into several broad categories:

- It is a *data manipulation language (DML)*. SQL includes commands to insert, update, delete, and retrieve data within the database tables. The data manipulation commands you will learn in this chapter are listed in Table 7.1. In this chapter, we will concentrate on the commands to retrieve data in interesting ways.
- It is a *data definition language (DDL)*. SQL includes commands to create database objects such as tables, indexes, and views, as well as commands to define access rights to those database objects. Some common data definition commands you will learn about in Chapter 8, Advanced SQL, are listed in Table 7.2.
- It is a *transaction control language (TCL)*. The DML commands in SQL are executed within the context of a **transaction**, which is a logical unit of work composed of one or more SQL statements, as defined by business rules (see Chapter 10, Transaction Management and Concurrency Control). SQL provides commands to control the processing of these statements an indivisible unit of work. These will be discussed in Chapter 8, after you learn about the DML commands that compose a transaction.
- It is a *data control language (DCL)*. Data control commands are used to control access to data objects, such as giving a one user permission to only view the PRODUCT table, and giving another use permission to change the data in the PRODUCT table. Common TCL and DCL commands are shown in Table 7.3.

SQL is relatively easy to learn. Its basic command set has a vocabulary of fewer than 100 words. Better yet, SQL is a nonprocedural language: you merely command *what* is to be done; you do not have to worry about *how*. For example, a single command creates the complex table structures required to store and manipulate data successfully; end users and programmers do not need to know the physical data storage format or the complex activities that take place when a SQL command is executed.

The American National Standards Institute (ANSI) prescribes a standard SQL. The ANSI SQL standards are also accepted by the International Organization for Standardization (ISO), a consortium composed of national standards bodies of more than 150 countries. Although adherence to the ANSI/ISO SQL standard is usually required in commercial and government contract database specifications, many RDBMS vendors add their own special enhancements. Consequently, it is seldom possible to move a SQL-based application from one RDBMS to another without making some changes.

However, even though there are several different SQL "dialects," their differences are minor. Whether you use Oracle, Microsoft SQL Server, MySQL, IBM DB2, Microsoft Access, or any other well-established RDBMS, a software manual should be sufficient to get you up to speed if you know the material presented in this chapter.

7-1a Data Types

The ANSI/ISO SQL standard defines many different data types. A data type is a specification about the kinds of data that can be stored in an attribute. A more

transaction
A logical unit of work composed of one or more SQL statements.

TABLE 7.1

SQL DATA MANIPULATION COMMANDS

COMMAND, OPTION, OR OPERATOR	DESCRIPTION	COVERED
SELECT	**Selects attributes from rows in one or more tables or views**	**Chapter 7**
FROM	Specifies the tables from which data should be retrieved	Chapter 7
WHERE	Restricts the selection of rows based on a conditional expression	Chapter 7
GROUP BY	Groups the selected rows based on one or more attributes	Chapter 7
HAVING	Restricts the selection of grouped rows based on a condition	Chapter 7
ORDER BY	Orders the selected rows based on one or more attributes	Chapter 7
INSERT	**Inserts row(s) into a table**	**Chapter 8**
UPDATE	**Modifies an attribute's values in one or more table's rows**	**Chapter 8**
DELETE	**Deletes one or more rows from a table**	**Chapter 8**
Comparison operators		**Chapter 7**
=, <, >, <=, >=, <>, !=	Used in conditional expressions	Chapter 7
Logical operators		**Chapter 7**
AND/OR/NOT	Used in conditional expressions	Chapter 7
Special operators	**Used in conditional expressions**	**Chapter 7**
BETWEEN	Checks whether an attribute value is within a range	Chapter 7
IN	Checks whether an attribute value matches any value within a value list	Chapter 7
LIKE	Checks whether an attribute value matches a given string pattern	Chapter 7
IS NULL	Checks whether an attribute value is null	Chapter 7
EXISTS	Checks whether a subquery returns any rows	Chapter 7
DISTINCT	Limits values to unique values	Chapter 7
Aggregate functions	**Used with SELECT to return mathematical summaries on columns**	**Chapter 7**
COUNT	Returns the number of rows with non-null values for a given column	Chapter 7
MIN	Returns the minimum attribute value found in a given column	Chapter 7
MAX	Returns the maximum attribute value found in a given column	Chapter 7
SUM	Returns the sum of all values for a given column	Chapter 7
AVG	Returns the average of all values for a given column	Chapter 7

thorough discussion of data types will wait until Chapter 8, when we discuss the SQL commands to implement entities and attributes as tables and columns. However, a basic understanding of data types is needed before we can discuss how to retrieve data. Data types influence queries that retrieve data because there are slight differences in the syntax of SQL and how it behaves during a query that are based on the data type of the column being retrieved. For now, consider that there are three fundamental types of data: character data, numeric data, and date data. *Character data* is composed of any printable characters such as alphabetic values, digits, punctuation, and special characters. Character data is also often referred to as a "string" because it is a collection of characters threaded together to create the value. *Numeric data* is composed of digits, such that the data has a specific numeric value. *Date data* is composed of date and,

TABLE 7.2

SQL DATA DEFINITION COMMANDS

COMMAND OR OPTION	DESCRIPTION	COVERED
CREATE SCHEMA AUTHORIZATION	**Creates a database schema**	**Chapter 8**
CREATE TABLE	**Creates a new table in the user's database schema**	**Chapter 8**
NOT NULL	Ensures that a column will not have null values	Chapter 8
UNIQUE	Ensures that a column will not have duplicate values	Chapter 8
PRIMARY KEY	Defines a primary key for a table	Chapter 8
FOREIGN KEY	Defines a foreign key for a table	Chapter 8
DEFAULT	Defines a default value for a column (when no value is given)	Chapter 8
CHECK	Validates data in an attribute	Chapter 8
CREATE INDEX	**Creates an index for a table**	**Chapter 8**
CREATE VIEW	**Creates a dynamic subset of rows and columns from one or more tables**	**Chapter 8**
ALTER TABLE	**Modifies a table's definition (adds, modifies, or deletes attributes or constraints)**	**Chapter 8**
CREATE TABLE AS	**Creates a new table based on a query in the user's database schema**	**Chapter 8**
DROP TABLE	**Permanently deletes a table (and its data)**	**Chapter 8**
DROP INDEX	**Permanently deletes an index**	**Chapter 8**
DROP VIEW	**Permanently deletes a view**	**Chapter 8**

TABLE 7.3

OTHER SQL COMMANDS

COMMAND OR OPTION	DESCRIPTION	COVERED
Transaction Control Language		
COMMIT	Permanently saves data changes	Chapter 8
ROLLBACK	Restores data to its original values	Chapter 8
Data Control Language		
GRANT	Gives a user permission to take a system action or access a data object	Chapter 16
REVOKE	Removes a previously granted permission from a user	Chapter 16

occasionally, time values. Although character data may contain digits, the DBMS does not recognize the numeric value of those digits.

7-1b SQL Queries

At the heart of SQL is the query. In Chapter 1, Database Systems, you learned that a query is a spur-of-the-moment question. Actually, in the SQL environment, the word *query* covers both questions and actions. Most SQL queries are used to answer questions such as these: "What products currently held in inventory are priced over $100, and what is the quantity on hand for each of those products?" or "How many employees have been hired since January 1, 2016, by each of the company's departments?" However, many SQL queries are used to perform actions such as adding or deleting table rows or changing attribute values within tables. Still other SQL queries create new tables or indexes. For a DBMS, a *query* is simply a SQL statement that must be executed. In most database-related jobs, retrieving data is by far the most common type of task. Not only do database professionals have to know how to retrieve data from the database, but virtually all application programmers need this skill as well.

Data retrieval is done in SQL using a SELECT query. When you run a SELECT command on a table, the RDBMS returns a set of one or more rows that have the same characteristics as a relational table. This is a very important characteristic of SQL commands. By default, most SQL data manipulation commands operate over an entire table (relation), which is why SQL commands are said to be *set-oriented* commands. A SQL **set-oriented** command works over a set of rows. The set may include one or more columns and zero or more rows from one or more tables. A SELECT query specifies which data should be retrieved and how it should be filtered, aggregated, and displayed. There are many potential clauses, or parts, to a SELECT query, as shown in Table 7.1. Constructing a SELECT query is similar to constructing objects with building blocks. The database programmer has to understand what each building block (clause) does and how the blocks fit together. Then he or she can make a plan for which blocks to use and determine how to assemble those blocks to produce the desired result.

7-1c The Database Model

set-oriented
Dealing with or related to sets, or groups of things. In the relational model, SQL operators are set-oriented because they operate over entire sets of rows and columns at once.

A simple database composed of the following tables is used to illustrate the SQL commands in this chapter: CUSTOMER, INVOICE, LINE, PRODUCT, and VENDOR. This database model is shown in Figure 7.1.

The database model in Figure 7.1 reflects the following business rules:

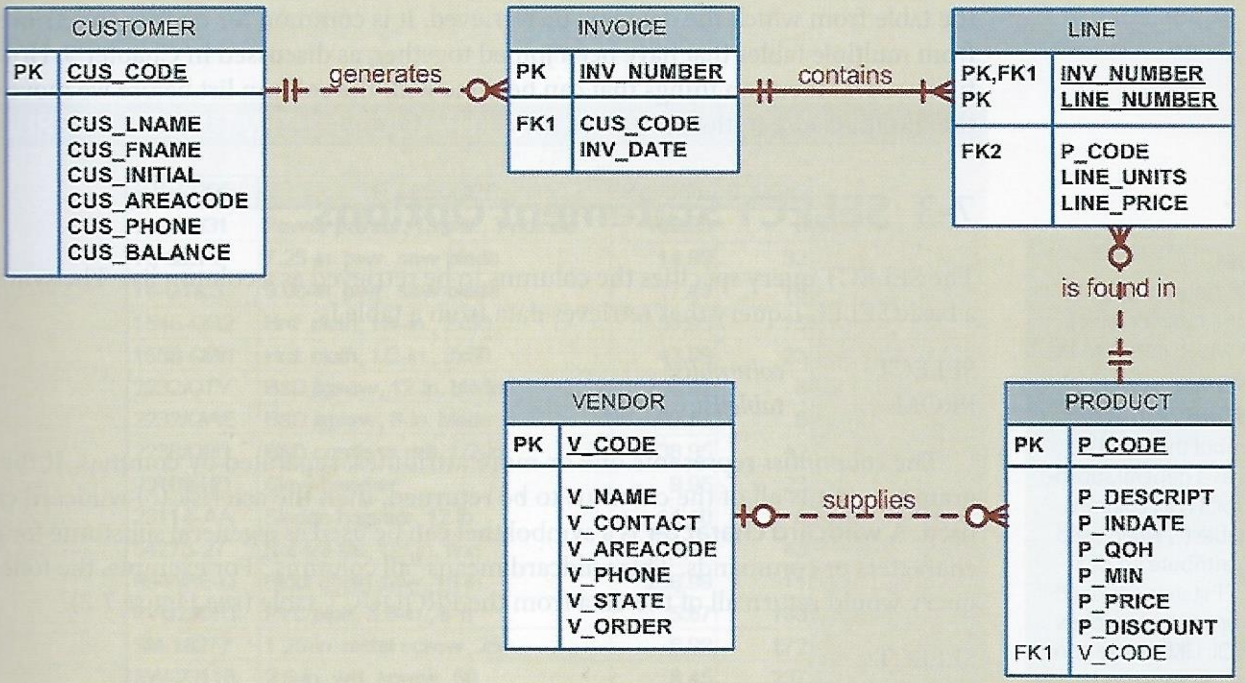
- A customer may generate many invoices. Each invoice is generated by one customer.
- An invoice contains one or more invoice lines. Each invoice line is associated with one invoice.
- Each invoice line references one product. A product may be found in many invoice lines. (You can sell more than one hammer to more than one customer.)
- A vendor may supply many products. Some vendors do not yet supply products. For example, a vendor list may include potential vendors.
- If a product is vendor-supplied, it is supplied by only a single vendor.
- Some products are not supplied by a vendor. For example, some products may be produced in-house or bought on the open market.

Except as noted, the database model shown in Figure 7.1 will be used for the queries in the remainder of the chapter. Recall that when an ERD is implemented as a database, each entity becomes a table in the database, and each attribute within an entity becomes a column in that table.

Online Content
The database model in Figure 7.1 is implemented in the Microsoft Access Ch07_SaleCo database, which is available at *www.cengagebrain.com*. (This database contains a few additional tables that are not reflected in Figure 7.1. These tables are used for discussion purposes only.) Scripts to create these tables in Oracle, MySQL, and SQL Server are also available at *www.cengagebrain.com.*

Note
This chapter focuses on SELECT queries to retrieve data from tables. Chapter 8 will explain how those tables are actually created and how the data is loaded into them. This reflects the experience of most entry-level database positions. As a new hire working with databases, you will likely spend quite a bit of time retrieving data from tables that already exist before you begin creating new tables and modifying the data.

FIGURE 7.1 THE DATABASE MODEL



7-2 Basic SELECT Queries

Each clause in a SELECT query performs a specific function. Understanding the function of each clause is key to developing the skills to construct queries to satisfy the reporting needs of the users. The following clauses will be covered in this chapter (although not in this order).

- SELECT—specifies the attributes to be returned by the query
- FROM—specifies the table(s) from which the data will be retrieved
- WHERE—filters the rows of data based on provided criteria
- GROUP BY—groups the rows of data into collections based on sharing the same values in one or more attributes
- HAVING—filters the groups formed in the GROUP BY clause based on provided criteria
- ORDER BY—sorts the final query result rows in ascending or descending order based on the values of one or more attributes.

Although SQL commands can be grouped together on a single line, complex command sequences are best shown on separate lines, with space between the SQL command and the command's components. Using that formatting convention makes it much easier to see the components of the SQL statements, which in turn makes it easy to trace the SQL logic and make corrections if necessary. The number of spaces used in the indention is up to you. For a SELECT query to retrieve data from the database, it will require at least a SELECT column list and a FROM clause. The **SELECT** column list specifies the relational projection, as discussed in Chapter 3, The Relational Database Model. The column list allows the programmer to specify which columns should be retrieved

SELECT
A SQL command that yields the values of all rows or a subset of rows in a table. The SELECT statement is used to retrieve data from tables.

by the query and the order in which they should be returned. Only columns specified in the column list will appear in the query result. The **FROM** clause is used to specify the table from which the data will be retrieved. It is common for queries to retrieve data from multiple tables that have been joined together, as discussed in Chapter 3. However, first, we will focus on things that can be done with the column list before we move on to the FROM clause options.

7-3 SELECT Statement Options

FROM
A SQL clause that specifies the table or tables from which data is to be retrieved.

wildcard character
A symbol that can be used as a general substitute for: (1) all columns in a table (*) when used in an attribute list of a SELECT statement or (2) zero or more characters in a SQL LIKE clause condition (% and _).

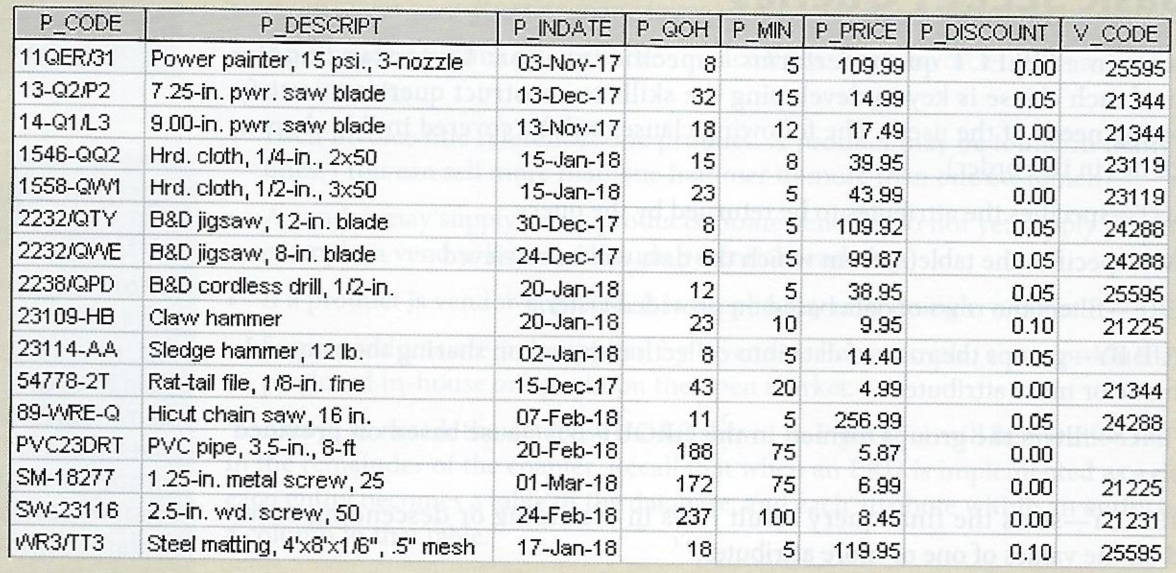
The SELECT query specifies the columns to be retrieved as a column list. The syntax for a basic SELECT query that retrieves data from a table is:

```
SELECT      columnlist
FROM        tablelist;
```

The *columnlist* represents one or more attributes, separated by commas. If the programmer wants all of the columns to be returned, then the asterisk (*) wildcard can be used. A **wildcard character** is a symbol that can be used as a general substitute for other characters or commands. This wildcard means "all columns." For example, the following query would return all of the data from the PRODUCT table (see Figure 7.2).

```
SELECT      *
FROM        PRODUCT;
```

FIGURE 7.2 SELECT AN ENTIRE TABLE

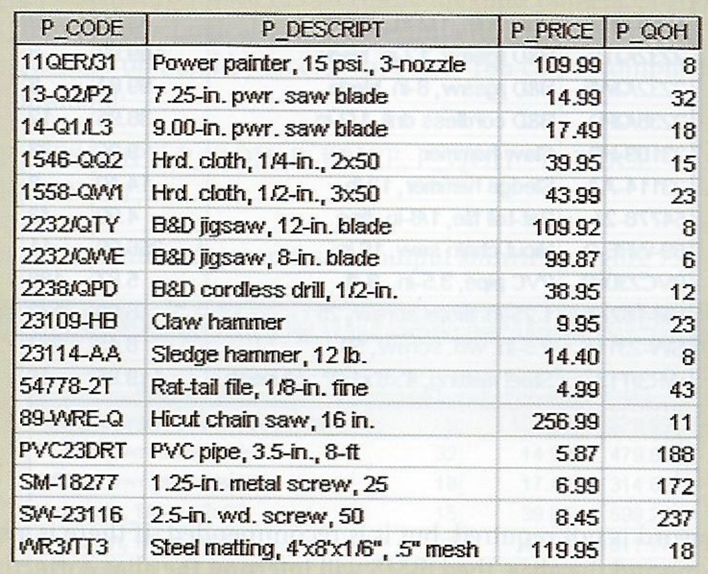


P_CODE	P_DESCRIPT	P_INDATE	P_QOH	P_MIN	P_PRICE	P_DISCOUNT	V_CODE
11QER/31	Power painter, 15 psi., 3-nozzle	03-Nov-17	8	5	109.99	0.00	25595
13-Q2/P2	7.25-in. pwr. saw blade	13-Dec-17	32	15	14.99	0.05	21344
14-Q1/L3	9.00-in. pwr. saw blade	13-Nov-17	18	12	17.49	0.00	21344
1546-QQ2	Hrd. cloth, 1/4-in., 2x50	15-Jan-18	15	8	39.95	0.00	23119
1558-QW1	Hrd. cloth, 1/2-in., 3x50	15-Jan-18	23	5	43.99	0.00	23119
2232/QTY	B&D jigsaw, 12-in. blade	30-Dec-17	8	5	109.92	0.05	24288
2232/QWE	B&D jigsaw, 8-in. blade	24-Dec-17	6	5	99.87	0.05	24288
2238/QPD	B&D cordless drill, 1/2-in.	20-Jan-18	12	5	38.95	0.05	25595
23109-HB	Claw hammer	20-Jan-18	23	10	9.95	0.10	21225
23114-AA	Sledge hammer, 12 lb.	02-Jan-18	8	5	14.40	0.05	
54778-2T	Rat-tail file, 1/8-in. fine	15-Dec-17	43	20	4.99	0.00	21344
89-WRE-Q	Hicut chain saw, 16 in.	07-Feb-18	11	5	256.99	0.05	24288
PVC23DRT	PVC pipe, 3.5-in., 8-ft	20-Feb-18	188	75	5.87	0.00	
SM-18277	1.25-in. metal screw, 25	01-Mar-18	172	75	6.99	0.00	21225
SW-23116	2.5-in. wd. screw, 50	24-Feb-18	237	100	8.45	0.00	21231
WR3/TT3	Steel matting, 4'x8'x1/6", .5" mesh	17-Jan-18	18	5	119.95	0.10	25595

In this query, the column list indicates that all columns (and by default all of the rows) should be returned. The FROM clause specifies that the data from the PRODUCT table is to be used. Recall from Chapter 3 that projection does not limit the rows being returned. To limit the rows being returned, relational selection (or restriction) must be used. The column list allows the programmer to specify which columns should be returned, as shown in the next query (see Figure 7.3).

```
SELECT      P_CODE, P_DESCRIPT, P_PRICE, P_QOH
FROM        PRODUCT;
```

FIGURE 7.3 SELECT WITH A COLUMN LIST



P_CODE	P_DESCRIPT	P_PRICE	P_QOH
11QER/31	Power painter, 15 psi., 3-nozzle	109.99	8
13-Q2/P2	7.25-in. pwr. saw blade	14.99	32
14-Q1/L3	9.00-in. pwr. saw blade	17.49	18
1546-QQ2	Hrd. cloth, 1/4-in., 2x50	39.95	15
1558-QW1	Hrd. cloth, 1/2-in., 3x50	43.99	23
2232/QTY	B&D jigsaw, 12-in. blade	109.92	8
2232/QWE	B&D jigsaw, 8-in. blade	99.87	6
2238/QPD	B&D cordless drill, 1/2-in.	38.95	12
23109-HB	Claw hammer	9.95	23
23114-AA	Sledge hammer, 12 lb.	14.40	8
54778-2T	Rat-tail file, 1/8-in. fine	4.99	43
89-WRE-Q	Hicut chain saw, 16 in.	256.99	11
PVC23DRT	PVC pipe, 3.5-in., 8-ft	5.87	188
SM-18277	1.25-in. metal screw, 25	6.99	172
SW-23116	2.5-in. wd. screw, 50	8.45	237
WR3/TT3	Steel matting, 4'x8'x1/6", .5" mesh	119.95	18

This query specifies that the data should come from the PRODUCT table, and that only the product code, description, price, and quantity on hand columns should be included. Notice that only the requested columns are returned and that the columns are in the same order in the output as they were listed in the query. To display the columns in a different order, simply change the order of the columns in the column list.

7-3a Using Column Aliases

Recall that the attribute within an entity is implemented as a column in the table. The attribute name becomes the name of that column. When that column is retrieved in a query, the attribute name is used as a label, or column heading, in the query output by default. If the programmer wants a different name to be used as the label in the output, a new name can be specified. The new name is referred to as an **alias**. For example, aliases are used in the following query (see Figure 7.4).

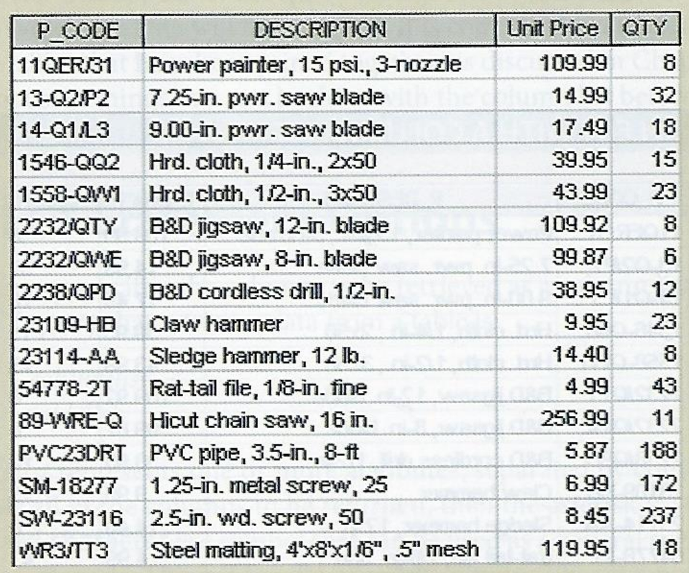
```
SELECT      P_CODE, P_DESCRIPT AS DESCRIPTION, P_PRICE AS “Unit Price”,
            P_QOH QTY
FROM        PRODUCT;
```

In this query and its output in Figure 7.4, the DESCRIPT attribute P_ is given the alias DESCRIPTION, P_PRICE is given the alias Unit Price, and P_QOH is given the alias QTY. There are a few things of interest about the use of these aliases:

- Not all columns in a query must use an alias
- AS is optional, but recommended
- Aliases that contain a space must be inside a delimiter (quotes)

alias
An alternative name for a column or table in a SQL statement.

FIGURE 7.4 SELECT WITH COLUMN ALIASES



P_CODE	DESCRIPTION	Unit Price	QTY
11QER/31	Power painter, 15 psi., 3-nozzle	109.99	8
13-Q2/P2	7.25-in. pwr. saw blade	14.99	32
14-Q1/L3	9.00-in. pwr. saw blade	17.49	18
1546-QQ2	Hrd. cloth, 1/4-in., 2x50	39.95	15
1558-QW1	Hrd. cloth, 1/2-in., 3x50	43.99	23
2232/QTY	B&D jigsaw, 12-in. blade	109.92	8
2232/QWE	B&D jigsaw, 8-in. blade	99.87	6
2238/QPD	B&D cordless drill, 1/2-in.	38.95	12
23109-HB	Claw hammer	9.95	23
23114-AA	Sledge hammer, 12 lb.	14.40	8
54778-2T	Rat-tail file, 1/8-in. fine	4.99	43
89-WRE-Q	Hicut chain saw, 16 in.	256.99	11
PVC23DRT	PVC pipe, 3.5-in., 8-ft	5.87	188
SM-18277	1.25-in. metal screw, 25	6.99	172
SW-23116	2.5-in. wd. screw, 50	8.45	237
WR3/TT3	Steel matting, 4'x8'x1/6", .5" mesh	119.95	18

The AS keyword is not required, but it is recommended. If there is a space between the column name and the alias, the DBMS will interpret the alias correctly. However, as we shall soon see, it is possible to embed formulas and functions within the column list, and you will generally want an alias for the columns produced. In those cases, having the AS keyword makes it much easier to read the query and understand that the alias is just an alias and not a part of the formula. Finally, the DBMS expects an alias to appear as a single word. If there are any spaces in the alias, then the programmer must use a delimiter to indicate where the alias begins and ends. In Figure 7.4, a double-quote delimiter was used around the Unit Price alias because it contains a space. Most DBMS products allow double quotes around a column alias.

Note

Using delimiters with column aliases even when the alias does not contain a space can serve other purposes. In some DBMSs, if the column alias is not placed inside a delimiter, it is automatically converted to uppercase letters. In those cases, using the delimiter allows the programmer to control the capitalization of the column alias. Using delimiters also allows a column alias to contain a special character, such as "+", or a SQL keyword, such as "SELECT." In general, using special characters and SQL keywords in column aliases is discouraged, but it is possible.

Note

MySQL uses a special delimiter, the back tick " ` " (usually found to the left of the number 1 on a standard keyboard) as a delimiter for column aliases if you want to refer to that alias elsewhere within the query, such as the ORDER BY clause covered later in this chapter.

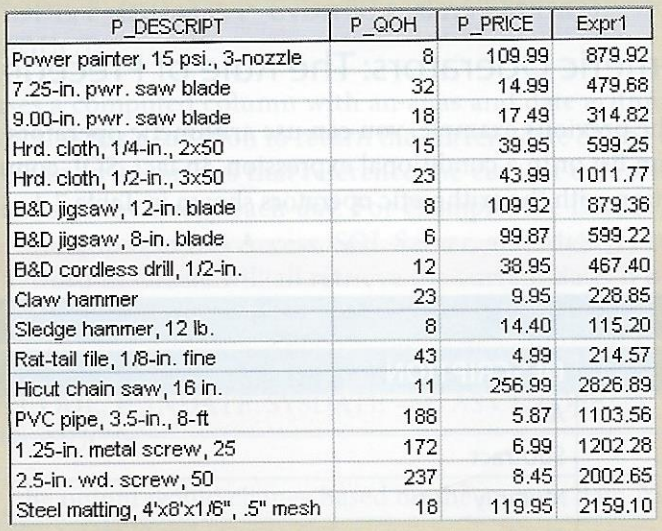
7-3b Using Computed Columns

A computed column (also called a calculated column) represents a derived attribute, as discussed in Chapter 4, Entity Relationship Modeling. Recall from Chapter 4 that a derived attribute may or may not be stored in the database. If the decision is made not to store the derived attribute, then the attribute must be calculated when it is needed. For example, suppose that you want to determine the total value of each of the products currently held in inventory. Logically, that determination requires the multiplication of each product's quantity on hand by its current price. You can accomplish this task with the following command:

```
SELECT    P_DESCRIPT, P_QOH, P_PRICE, P_QOH * P_PRICE
FROM      PRODUCT;
```

Entering the SQL command generates the output shown in Figure 7.5.

FIGURE 7.5 SELECT STATEMENT WITH A COMPUTED COLUMN



P_DESCRIPT	P_QOH	P_PRICE	Expr1
Power painter, 15 psi., 3-nozzle	8	109.99	879.92
7.25-in. pwr. saw blade	32	14.99	479.68
9.00-in. pwr. saw blade	18	17.49	314.82
Hrd. cloth, 1/4-in., 2x50	15	39.95	599.25
Hrd. cloth, 1/2-in., 3x50	23	43.99	1011.77
B&D jigsaw, 12-in. blade	8	109.92	879.36
B&D jigsaw, 8-in. blade	6	99.87	599.22
B&D cordless drill, 1/2-in.	12	38.95	467.40
Claw hammer	23	9.95	228.85
Sledge hammer, 12 lb.	8	14.40	115.20
Rat-tail file, 1/8-in. fine	43	4.99	214.57
Hicut chain saw, 16 in.	11	256.99	2826.89
PVC pipe, 3.5-in., 8-ft	188	5.87	1103.56
1.25-in. metal screw, 25	172	6.99	1202.28
2.5-in. wd. screw, 50	237	8.45	2002.65
Steel matting, 4'x8'x1/6", .5" mesh	18	119.95	2159.10

SQL accepts any valid expressions (or formulas) in the computed columns. Such formulas can contain any valid mathematical operators and functions that are applied to attributes in any of the tables specified in the FROM clause of the SELECT statement. Different DBMS products vary in the column headings that are displayed for the computed column.

Note

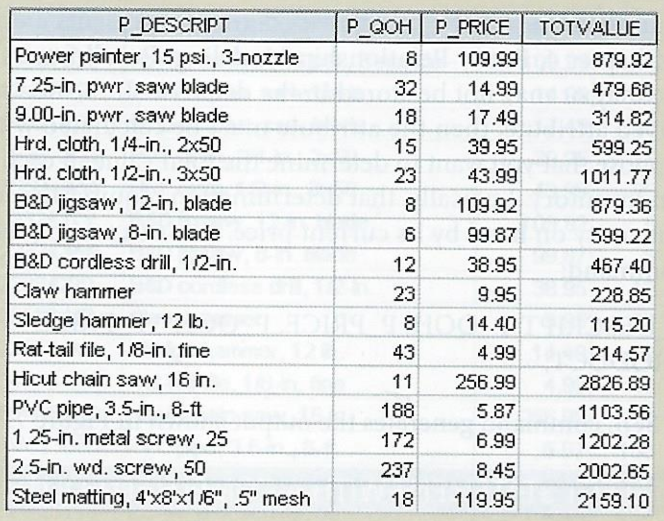
MS Access automatically adds an Expr label to all computed columns when an alias is not specified. (The first computed column would be labeled Expr1; the second, Expr2; and so on.) Oracle uses the actual formula text as the label for the computed column. Other DBMSs return the column without a heading label.

To make the output more readable, an alias is typically used for any computed fields. For example, you can rewrite the previous SQL statement as follows:

```
SELECT    P_DESCRIPT, P_QOH, P_PRICE, P_QOH * P_PRICE AS TOTVALUE
FROM      PRODUCT;
```

The output of the command is shown in Figure 7.6.

FIGURE 7.6 SELECT STATEMENT WITH A COMPUTED COLUMN AND AN ALIAS



P_DESCRIPT	P_QOH	P_PRICE	TOTVALUE
Power painter, 15 psi., 3-nozzle	8	109.99	879.92
7.25-in. pwr. saw blade	32	14.99	479.68
9.00-in. pwr. saw blade	18	17.49	314.82
Hrd. cloth, 1/4-in., 2x50	15	39.95	599.25
Hrd. cloth, 1/2-in., 3x50	23	43.99	1011.77
B&D jigsaw, 12-in. blade	8	109.92	879.36
B&D jigsaw, 8-in. blade	6	99.87	599.22
B&D cordless drill, 1/2-in.	12	38.95	467.40
Claw hammer	23	9.95	228.85
Sledge hammer, 12 lb.	8	14.40	115.20
Rat-tail file, 1/8-in. fine	43	4.99	214.57
Hicut chain saw, 16 in.	11	256.99	2826.89
PVC pipe, 3.5-in., 8-ft	188	5.87	1103.56
1.25-in. metal screw, 25	172	6.99	1202.28
2.5-in. wd. screw, 50	237	8.45	2002.65
Steel matting, 4'x8'x1/6", .5" mesh	18	119.95	2159.10

7-3c Arithmetic Operators: The Rule of Precedence

As you saw in the previous example, you can use arithmetic operators with table attributes in a column list or in a conditional expression. In fact, SQL commands are often used in conjunction with the arithmetic operators shown in Table 7.4.

TABLE 7.4

THE ARITHMETIC OPERATORS

OPERATOR	DESCRIPTION
+	Add
-	Subtract
*	Multiply
/	Divide
^	Raise to the power of (some applications use ** instead of ^)

Do not confuse the multiplication symbol (*) with the wildcard symbol used by some SQL implementations, such as MS Access. The wildcard symbol is used only in string comparisons, while the multiplication symbol is used in conjunction with mathematical procedures.

As you perform mathematical operations on attributes, remember the mathematical rules of precedence. As the name suggests, the **rules of precedence** are the rules that establish the order in which computations are completed. For example, note the order of the following computational sequence:

1. Perform operations within parentheses.
2. Perform power operations.
3. Perform multiplications and divisions.
4. Perform additions and subtractions.

rules of precedence
Basic algebraic rules that specify the order in which operations are performed. For example, operations within parentheses are executed first, so in the equation 2 + (3 × 5), the multiplication portion is calculated first, making the correct answer 17.

The application of the rules of precedence will tell you that 8 + 2 * 5 = 8 + 10 = 18, but (8 + 2) * 5 = 10 * 5 = 50. Similarly, 4 + 5^2 * 3 = 4 + 25 * 3 = 79, but (4 + 5)^2 * 3 = 81 * 3 = 243, while the operation expressed by (4 + 5^2) * 3 yields the answer (4 + 25) * 3 = 29 * 3 = 87.

7-3d Date Arithmetic

Date data in the column list can be interesting when used in computed fields. Internally, the DBMS stores a date value in a numeric format. Although the details can be complicated, essentially, a date is stored as a day number, that is, the number of days that have passed since some defined point in history. Exactly what that point in history is varies from one DBMS to another. However, because the values are stored as a number of days, it is possible to perform date arithmetic in a query. For example, if today's date in some DBMS is the day number "250,000," then tomorrow will be "250,001," and yesterday was "249,999." Adding or subtracting a number from a date that is stored in a date data type returns the date that is the specified number of days from the given date. Subtracting one date value from another yields the number of days between those dates.

Suppose that a manager wants a list of all products, the dates they were received, and the warranty expiration date (90 days from receiving the product). To generate that list, you would make the following query:

```
SELECT    P_CODE, P_INDATE, P_INDATE + 90 AS EXPDATE
FROM      PRODUCT;
```

This query uses a computed column with an alias and date arithmetic in a single query. The DBMS also has a function to return the current date on the database server, making it possible to write queries that reference the current date without having to change the contents of the query each day. For example, the DATE(), GETDATE(), and CURDATE() functions in MS Access, SQL Server, and MySQL, respectively, and the SYSDATE keyword in Oracle will all retrieve the current date. If a manager wants to a list of products and the warranty cutoff date for products, the query in Oracle would be:

```
SELECT     P_CODE, P_INDATE, SYSDATE – 90 AS CUTOFF
FROM       PRODUCT;
```

In this query, the output would change based on the current date. You can use these functions anywhere a date literal is expected.

7-3e Listing Unique Values

How many *different* vendors are currently represented in the PRODUCT table? A simple listing (SELECT) is not very useful if the table contains several thousand rows and you have to sift through the vendor codes manually. Fortunately, SQL's **DISTINCT** clause produces a list of only those values that are different from one another. For example, the command

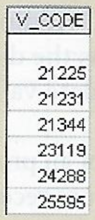
```
SELECT    DISTINCT V_CODE
FROM      PRODUCT;
```

yields only the different vendor codes (V_CODE) in the PRODUCT table, as shown in Figure 7.7. The DISTINCT keyword only appears once in the query, and that is immediately following the SELECT keyword. Note that the first output row shows a null. Rows may contain a null for the V_CODE attribute if the product is developed in-house or if it is purchased directly from the manufacturer. As discussed in Chapter 3, nulls can be problematic because it is difficult to know what the null means in the business environment. Nulls can also be problematic when writing SQL code. Different operators and functions treat nulls differently. For example, the DISTINCT keyword considers

DISTINCT
A SQL clause that produces only a list of values that are different from one another.

null to be a value, and it considers all nulls to be the same value. In later sections, we will encounter functions that ignore nulls, and we will see comparisons that consider all nulls to be different. As a SQL developer, you must understand how nulls will be treated by the code you are writing.

FIGURE 7.7 A LISTING OF DISTINCT V_CODE VALUES IN THE PRODUCT TABLE



V_CODE
21225
21231
21344
23119
24288
25595

7-4 FROM Clause Options

The **FROM** clause of the query specifies the table or tables from which the data is to be retrieved. In the following query, the data is being retrieved from only the PRODUCT table.

```
SELECT    P_CODE, P_DESCRIPT, P_INDATE, P_QOH, P_MIN, P_PRICE,
          P_DISCOUNT, V_CODE
FROM      PRODUCT;
```

In practice, most SELECT queries will need to retrieve data from multiple tables. In Chapter 3, we looked at JOIN operators that are used to combine data from multiple tables in meaningful ways. The database design process that led to the current database was in many ways a process of decomposition—the designer took an integrated set of data related to a business problem and decomposed that data into separate entities to create a flexible, stable structure for storing and manipulating that data. Now, through the use of joins, the programmer reintegrates pieces of the data to satisfy the users' information needs. *Inner joins* return only rows from the tables that match on a common value. *Outer joins* return the same matched rows as the inner join, plus unmatched rows from one table or the other. (The various types of joins are presented in Chapter 3.)

The join condition is generally composed of an equality comparison between the foreign key and the primary key of related tables. For example, suppose that you want to join the two tables VENDOR and PRODUCT. Because V_CODE is the foreign key in the PRODUCT table and the primary key in the VENDOR table, the link is established on V_CODE. (See Table 7.5.)

TABLE 7.5

CREATING LINKS THROUGH FOREIGN KEYS

TABLE	ATTRIBUTES TO BE SHOWN	LINKING ATTRIBUTE
PRODUCT	P_DESCRIPT, P_PRICE	V_CODE
VENDOR	V_NAME, V_CONTACT, V_AREACODE, V_PHONE	V_CODE