25

Accessing Databases with JDBC

It is a capital mistake to theorize before one has data.
—Arthur Conan Doyle

Now go, write it before them in a table, and note it in a book, that it may be for the time to come for ever and ever.
—The Holy Bible, Isaiah 30:8

Get your facts first, and then you can distort them as much as you please.
—Mark Twain

I like two kinds of men: domestic and foreign.
—Mae West

OBJECTIVES

In this chapter you will learn:

- Relational database concepts.
- To use Structured Query Language (SQL) to retrieve data from and manipulate data in a database.
- To use the JDBC™ API of package `java.sql` to access databases.
- To use the `RowSet` interface from package `javax.sql` to manipulate databases.
- To use JDBC 4.0's automatic JDBC driver discovery.
- To use `PreparedStatement`s to create precompiled SQL statements with parameters.
- How transaction processing makes database applications more robust.

Java™ How to Program, Seventh Edition By H. M. Deitel - Deitel & Associates, Inc., P. J. Deitel - Deitel & Associates, Inc. ISBN: 9780136085676 Publisher: Prentice Hall
Print Publication Date: 2006/12/27

Outline

25.1 Introduction

A **database** is an organized collection of data. There are many different strategies for organizing data to facilitate easy access and manipulation. A **database management system** (DBMS) provides mechanisms for storing, organizing, retrieving and modifying data for many users. Database management systems allow for the access and storage of data without concern for the internal representation of data.

Today's most popular database systems are relational databases, where the data is stored without consideration of its physical structure (Section 25.2). A language called SQL—pronounced "sequel," or as its individual letters—is the international standard language used almost universally with relational databases to perform **queries** (i.e., to request information that satisfies given criteria) and to manipulate data. [*Note:* As you learn about SQL, you will see some authors writing "a SQL statement" (which assumes the pronunciation "sequel") and others writing "an SQL statement" (which assumes that the individual letters are pronounced). In this book we pronounce SQL as "sequel."

Java™ How to Program, Seventh Edition By H. M. Deitel - Deitel & Associates, Inc., P. J. Deitel - Deitel & Associates, Inc. ISBN: 9780136085676 Publisher: Prentice Hall
Print Publication Date: 2006/12/27

Some popular **relational database management systems** (**RDBMSs**) are Microsoft SQL Server, Oracle, Sybase, IBM DB2, Informix, PostgreSQL and MySQL. The JDK now comes with a pure-Java RDBMS called Java DB—Sun's version of Apache Derby. In this chapter, we present examples using MySQL and Java DB.[1]

Java programs communicate with databases and manipulate their data using the **JDBC™ API**. A **JDBC driver** enables Java applications to connect to a database in a particular DBMS and allows you to manipulate that database using the JDBC API.

Software Engineering Observation 25.1

Using the JDBC API enables developers to change the underlying DBMS without modifying the Java code that accesses the database.

Most popular database management systems now provide JDBC drivers. There are also many third-party JDBC drivers available. In this chapter, we introduce JDBC and use it to manipulate MySQL and Java DB databases. The techniques demonstrated here can also be used to manipulate other databases that have JDBC drivers. Check your DBMS's documentation to determine whether your DBMS comes with a JDBC driver. If not, third-party vendors provide JDBC drivers for many DBMSs.

For more information on JDBC, visit

```
java.sun.com/javase/technologies/database/index.jsp
```

This site contains JDBC information including the JDBC specification, FAQs, a learning resource center and software downloads to search for JDBC drivers for your DBMS,

```
developers.sun.com/product/jdbc/drivers/
```

This site provides a search engine to help you locate drivers appropriate for your DBMS.

25.2 Relational Databases

A **relational database** is a logical representation of data that allows the data to be accessed without consideration of its physical structure. A relational database stores data in **tables**. Figure 25.1 illustrates a sample table that might be used in a personnel system. The table name is `Employee`, and its primary purpose is to store the attributes of an employee. Tables are composed of **rows**, and rows are composed of **columns** in which values are stored. This table consists of six rows. The `Number` column of each row in this table is the table's **primary key**—a column (or group of columns) in a table with a unique value that cannot be duplicated in other rows. This guarantees that each row can be identified by its primary key. Good examples of primary key columns are a social security number, an employee ID number and a part number in an inventory system, as values in each of these columns are guaranteed to be unique. The rows in Fig. 25.1 are displayed in order by primary key. In this case, the rows are listed in increasing order, but we could also use decreasing order.

1. MySQL is one of the most popular open-source database management systems in use today. As of this writing, it does not yet support JDBC 4, which is part of Java SE 6 (Mustang). However, Sun's Java DB, which is based on the open-source Apache Derby database management system and bundled with Sun's JDK 1.6.0, does support JDBC 4. We use MySQL and JDBC 3 in Sections 25.8–25.10, and we use Java DB and JDBC 4 in Section 25.11.

Java™ How to Program, Seventh Edition By H. M. Deitel - Deitel & Associates, Inc., P. J. Deitel - Deitel & Associates, Inc. ISBN: 9780136085676 Publisher: Prentice Hall
Print Publication Date: 2006/12/27

Number	Name	Department	Salary	Location
23603	Jones	413	1100	New Jersey
24568	Kerwin	413	2000	New Jersey
34589	Larson	642	1800	Los Angeles
35761	Myers	611	1400	Orlando
47132	Neumann	413	9000	New Jersey
78321	Stephens	611	8500	Orlando

Row

Primary key

Column

Fig. 25.1 | `Employee` table sample data.

Rows in tables are not guaranteed to be stored in any particular order. As we will demonstrate in an upcoming example, programs can specify ordering criteria when requesting data from a database.

Each column represents a different data attribute. Rows are normally unique (by primary key) within a table, but particular column values may be duplicated between rows. For example, three different rows in the `Employee` table's `Department` column contain number 413.

Different users of a database are often interested in different data and different relationships among the data. Most users require only subsets of the rows and columns. To obtain these subsets, we use queries to specify which data to select from a table. You use SQL to define complex queries that select data from a table. For example, you might select data from the `Employee` table to create a result that shows where each department is located, and present the data sorted in increasing order by department number. This result is shown in Fig. 25.2. SQL queries are discussed in Section 25.4.

Department	Location
413	New Jersey
611	Orlando
642	Los Angeles

Fig. 25.2 | Result of selecting distinct `Department` and `Location` data from table `Employee`.

25.3 Relational Database Overview: The books Database

We now overview relational databases in the context of a sample `books` database we created for this chapter. Before we discuss SQL, we overview the tables of the `books` database. We use this database to introduce various database concepts, including how to use SQL to obtain information from the database and to manipulate the data. We provide a script to create the database. You can find the script in the examples directory for this chapter on the CD that accompanies this book. Section 25.5 explains how to use this script.

The database consists of three tables: `authors`, `authorISBN` and `titles`. The `authors` table (described in Fig. 25.3) consists of three columns that maintain each author's unique ID number, first name and last name. Figure 25.4 contains sample data from the `authors` table of the `books` database.

Java™ How to Program, Seventh Edition By H. M. Deitel - Deitel & Associates, Inc., P. J. Deitel - Deitel & Associates, Inc. ISBN: 9780136085676 Publisher: Prentice Hall
Print Publication Date: 2006/12/27

Column	Description
`authorID`	Author's ID number in the database. In the `books` database, this integer column is defined as **autoincremented**—for each row inserted in this table, the `authorID` value is increased by 1 automatically to ensure that each row has a unique `authorID`. This column represents the table's primary key.
`firstName`	Author's first name (a string).
`lastName`	Author's last name (a string).

Fig. 25.3 | `authors` table from the `books` database.

authorID	firstName	lastName
1	Harvey	Deitel
2	Paul	Deitel
3	Andrew	Goldberg
4	David	Choffnes

Fig. 25.4 | Sample data from the `authors` table.

The `authorISBN` table (described in Fig. 25.5) consists of two columns that maintain each ISBN and the corresponding author's ID number. This table associates authors with their books. Both columns are foreign keys that represent the relationship between the tables `authors` and `titles`—one row in table `authors` may be associated with many rows in table `titles`, and vice versa. Figure 25.6 contains sample data from the `authorISBN`

Column	Description
`authorID`	The author's ID number, a foreign key to the `authors` table.
`isbn`	The ISBN for a book, a foreign key to the `titles` table.

Fig. 25.5 | `authorISBN` table from the `books` database.

authorID	isbn	authorID	isbn
1	0131869000	2	0131450913
2	0131869000	1	0131828274
1	0131483986	2	0131828274
2	0131483986	3	0131450913
1	0131450913	4	0131828274

Fig. 25.6 | Sample data from the `authorISBN` table of `books`.

Java™ How to Program, Seventh Edition By H. M. Deitel - Deitel & Associates, Inc., P. J. Deitel - Deitel & Associates, Inc. ISBN: 9780136085676 Publisher: Prentice Hall
Print Publication Date: 2006/12/27

table of the books database. [*Note:* To save space, we have split the contents of this table into two columns, each containing the authorID and isbn columns.] The authorID column is a **foreign key**—a column in this table that matches the primary key column in another table (i.e., authorID in the authors table). Foreign keys are specified when creating a table. The foreign key helps maintain the **Rule of Referential Integrity**: Every foreign-key value must appear as another table's primary-key value. This enables the DBMS to determine whether the authorID value for a particular book is valid. Foreign keys also allow related data in multiple tables to be selected from those tables for analytic purposes—this is known as **joining** the data.

The titles table described in Fig. 25.7 consists of four columns that stand for the ISBN, the title, the edition number and the copyright year. The table is in Fig. 25.8.

There is a one-to-many relationship between a primary key and a corresponding foreign key (e.g., one publisher can publish many books). A foreign key can appear many times in its own table, but can appear only once (as the primary key) in another table. Figure 25.9 is an **entity-relationship** (**ER**) **diagram** for the books database. This diagram shows the database tables and the relationships among them. The first compartment in each box contains the table's name. The names in italic are primary keys. A table's primary key uniquely identifies each row in the table. Every row must have a primary-key value, and that value must be unique in the table. This is known as the **Rule of Entity Integrity**.

Column	Description
isbn	ISBN of the book (a string). The table's primary key. ISBN is an abbreviation for "International Standard Book Number"—a numbering scheme that publishers use to give every book a unique identification number.
title	Title of the book (a string).
editionNumber	Edition number of the book (an integer).
copyright	Copyright year of the book (a string).

Fig. 25.7 | titles table from the books database.

isbn	title	editionNumber	copyright
0131869000	Visual Basic How to Program	3	2006
0131525239	Visual C# How to Program	2	2006
0132222205	Java How to Program	7	2007
0131857576	C++ How to Program	5	2005
0132404168	C How to Program	5	2007
0131450913	Internet & World Wide Web How to Program	3	2004

Fig. 25.8 | Sample data from the titles table of the books database .

Java™ How to Program, Seventh Edition By H. M. Deitel - Deitel & Associates, Inc., P. J. Deitel - Deitel & Associates, Inc. ISBN: 9780136085676 Publisher: Prentice Hall
Print Publication Date: 2006/12/27

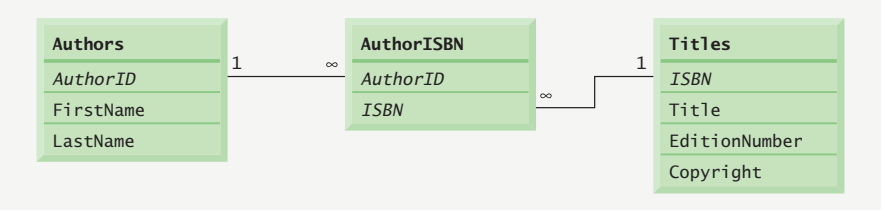

Fig. 25.9 | Table relationships in the `books` database.

Common Programming Error 25.1

Not providing a value for every column in a primary key breaks the Rule of Entity Integrity and causes the DBMS to report an error.

Common Programming Error 25.2

Providing the same value for the primary key in multiple rows causes the DBMS to report an error.

The lines connecting the tables in Fig. 25.9 represent the relationships between the tables. Consider the line between the `authorISBN` and `authors` tables. On the `authors` end of the line, there is a `1`, and on the `authorISBN` end, there is an infinity symbol (∞), indicating a **one-to-many relationship** in which every author in the `authors` table can have an arbitrary number of books in the `authorISBN` table. Note that the relationship line links the `authorID` column in the table `authors` (i.e., its primary key) to the `authorID` column in table `authorISBN` (i.e., its foreign key). The `authorID` column in the `authorISBN` table is a foreign key.

Common Programming Error 25.3

Providing a foreign-key value that does not appear as a primary-key value in another table breaks the Rule of Referential Integrity and causes the DBMS to report an error.

The line between the `titles` and `authorISBN` tables illustrates another one-to-many relationship; a title can be written by any number of authors. In fact, the sole purpose of the `authorISBN` table is to provide a many-to-many relationship between the `authors` and `titles` tables—an author can write any number of books and a book can have any number of authors.

25.4 SQL

We now provide an overview of SQL in the context of our `books` database. You will be able to use the SQL discussed here in the examples later in the chapter and in examples in Chapters 26–28.

The next several subsections discuss the SQL keywords listed in Fig. 25.10 in the context of SQL queries and statements. Other SQL keywords are beyond this text's scope. To learn other keywords, refer to the SQL reference guide supplied by the vendor of the RDBMS you are using. [*Note:* For more information on SQL, refer to the web resources in Section 25.15 and the recommended readings listed at the end of this chapter.]

SQL keyword	Description
SELECT	Retrieves data from one or more tables.
FROM	Tables involved in the query. Required in every SELECT.
WHERE	Criteria for selection that determine the rows to be retrieved, deleted or updated. Optional in a SQL query or a SQL statement.
GROUP BY	Criteria for grouping rows. Optional in a SELECT query.
ORDER BY	Criteria for ordering rows. Optional in a SELECT query.
INNER JOIN	Merge rows from multiple tables.
INSERT	Insert rows into a specified table.
UPDATE	Update rows in a specified table.
DELETE	Delete rows from a specified table.

Fig. 25.10 | SQL query keywords.

25.4.1 Basic SELECT Query

Let us consider several SQL queries that extract information from database `books`. A SQL query "selects" rows and columns from one or more tables in a database. Such selections are performed by queries with the `SELECT` keyword. The basic form of a `SELECT` query is

```
SELECT * FROM tableName
```

in which the asterisk (*) indicates that all columns from the *tableName* table should be retrieved. For example, to retrieve all the data in the `authors` table, use

```
SELECT * FROM authors
```

Most programs do not require all the data in a table. To retrieve only specific columns from a table, replace the asterisk (*) with a comma-separated list of the column names. For example, to retrieve only the columns `authorID` and `lastName` for all rows in the `authors` table, use the query

```
SELECT authorID, lastName FROM authors
```

This query returns the data listed in Fig. 25.11.

authorID	lastName
1	Deitel
2	Deitel
3	Goldberg
4	Choffnes

Fig. 25.11 | Sample `authorID` and `lastName` data from the `authors` table.

Java™ How to Program, Seventh Edition By H. M. Deitel - Deitel & Associates, Inc., P. J. Deitel - Deitel & Associates, Inc. ISBN: 9780136085676 Publisher: Prentice Hall
Print Publication Date: 2006/12/27

Software Engineering Observation 25.2

For most queries, the asterisk (*) should not be used to specify column names. In general, you process results by knowing in advance the order of the columns in the result—for example, selecting `authorID` *and* `lastName` *from table* `authors` *ensures that the columns will appear in the result with* `authorID` *as the first column and* `lastName` *as the second column. Programs typically process result columns by specifying the column number in the result (starting from number 1 for the first column). Selecting columns by name also avoids returning unneeded columns and protects against changes in the actual order of the columns in the table(s).*

Common Programming Error 25.4

If you assume that the columns are always returned in the same order from a query that uses the asterisk (*), the program may process the results incorrectly. If the column order in the table(s) changes or if additional columns are added at a later time, the order of the columns in the result would change accordingly.

25.4.2 WHERE Clause

In most cases, it is necessary to locate rows in a database that satisfy certain **selection criteria**. Only rows that satisfy the selection criteria (formally called **predicates**) are selected. SQL uses the optional `WHERE` **clause** in a query to specify the selection criteria for the query. The basic form of a query with selection criteria is

```
SELECT columnName1, columnName2, ... FROM tableName WHERE criteria
```

For example, to select the `title`, `editionNumber` and `copyright` columns from table `titles` for which the `copyright` date is greater than `2005`, use the query

```
SELECT title, editionNumber, copyright
   FROM titles
   WHERE copyright > '2005'
```

Figure 25.12 shows the result of the preceding query. The `WHERE` clause criteria can contain the operators `<`, `>`, `<=`, `>=`, `=`, `<>` and `LIKE`. Operator **`LIKE`** is used for **pattern matching** with wildcard characters **percent** (**`%`**) and **underscore** (`_`). Pattern matching allows SQL to search for strings that match a given pattern.

A pattern that contains a percent character (`%`) searches for strings that have zero or more characters at the percent character's position in the pattern. For example, the next query locates the rows of all the authors whose last name starts with the letter `D`:

```
SELECT authorID, firstName, lastName
   FROM authors
   WHERE lastName LIKE 'D%'
```

This query selects the two rows shown in Fig. 25.13—two of the four authors have a last name starting with the letter `D` (followed by zero or more characters). The `%` in the `WHERE` clause's `LIKE` pattern indicates that any number of characters can appear after the letter `D` in the `lastName`. Note that the pattern string is surrounded by single-quote characters.

Portability Tip 25.1

See the documentation for your database system to determine whether SQL is case sensitive on your system and to determine the syntax for SQL keywords (i.e., should they be all uppercase letters, all lowercase letters or some combination of the two?).

Java™ How to Program, Seventh Edition By H. M. Deitel - Deitel & Associates, Inc., P. J. Deitel - Deitel & Associates, Inc. ISBN: 9780136085676 Publisher: Prentice Hall
Print Publication Date: 2006/12/27

title	editionNumber	copyright
Visual C# How to Program	2	2006
Visual Basic 2005 How to Program	3	2006
Java How to Program	7	2007
C How to Program	5	2007

Fig. 25.12 | Sampling of titles with copyrights after 2005 from table `titles`.

authorID	firstName	lastName
1	Harvey	Deitel
2	Paul	Deitel

Fig. 25.13 | Authors whose last name starts with `D` from the `authors` table.

Portability Tip 25.2

Read your database system's documentation carefully to determine whether your system supports the `LIKE` operator. The SQL we discuss is supported by most RDBMSs, but it is always a good idea to check the features of SQL that are supported by your RDBMS.

An underscore (_) in the pattern string indicates a single wildcard character at that position in the pattern. For example, the following query locates the rows of all the authors whose last names start with any character (specified by _), followed by the letter `o`, followed by any number of additional characters (specified by `%`):

```
SELECT authorID, firstName, lastName
   FROM authors
   WHERE lastName LIKE '_o%'
```

The preceding query produces the row shown in Fig. 25.14, because only one author in our database has a last name that contains the letter `o` as its second letter.

authorID	firstName	lastName
3	Andrew	Goldberg

Fig. 25.14 | The only author from the `authors` table whose last name contains `o` as the second letter.

25.4.3 ORDER BY Clause

The rows in the result of a query can be sorted into ascending or descending order by using the optional `ORDER BY` clause. The basic form of a query with an `ORDER BY` clause is

Java™ How to Program, Seventh Edition By H. M. Deitel - Deitel & Associates, Inc., P. J. Deitel - Deitel & Associates, Inc. ISBN: 9780136085676 Publisher: Prentice Hall
Print Publication Date: 2006/12/27

```
SELECT columnName1, columnName2, ... FROM tableName ORDER BY column ASC
SELECT columnName1, columnName2, ... FROM tableName ORDER BY column DESC
```

where `ASC` specifies ascending order (lowest to highest), `DESC` specifies descending order (highest to lowest) and *column* specifies the column on which the sort is based. For example, to obtain the list of authors in ascending order by last name (Fig. 25.15), use the query

```
SELECT authorID, firstName, lastName
   FROM authors
   ORDER BY lastName ASC
```

Note that the default sorting order is ascending, so `ASC` is optional. To obtain the same list of authors in descending order by last name (Fig. 25.16), use the query

```
SELECT authorID, firstName, lastName
   FROM authors
   ORDER BY lastName DESC
```

Multiple columns can be used for sorting with an `ORDER BY` clause of the form

```
ORDER BY column1 sortingOrder, column2 sortingOrder, ...
```

where *sortingOrder* is either `ASC` or `DESC`. Note that the *sortingOrder* does not have to be identical for each column. The query

```
SELECT authorID, firstName, lastName
   FROM authors
   ORDER BY lastName, firstName
```

authorID	firstName	lastName
4	David	Choffnes
1	Harvey	Deitel
2	Paul	Deitel
3	Andrew	Goldberg

Fig. 25.15 | Sample data from table `authors` in ascending order by `lastName`.

authorID	firstName	lastName
3	Andrew	Goldberg
1	Harvey	Deitel
2	Paul	Deitel
4	David	Choffnes

Fig. 25.16 | Sample data from table `authors` in descending order by `lastName`.

Java™ How to Program, Seventh Edition By H. M. Deitel - Deitel & Associates, Inc., P. J. Deitel - Deitel & Associates, Inc. ISBN: 9780136085676 Publisher: Prentice Hall
Print Publication Date: 2006/12/27

sorts all the rows in ascending order by last name, then by first name. If any rows have the same last name value, they are returned sorted by first name (Fig. 25.17).

The `WHERE` and `ORDER BY` clauses can be combined in one query, as in

```
SELECT isbn, title, editionNumber, copyright
   FROM titles
   WHERE title LIKE '%How to Program'
   ORDER BY title ASC
```

which returns the `isbn`, `title`, `editionNumber` and `copyright` of each book in the `titles` table that has a `title` ending with `"How to Program"` and sorts them in ascending order by `title`. A portion of the query results are shown in Fig. 25.18.

authorID	firstName	lastName
4	David	Choffnes
1	Harvey	Deitel
2	Paul	Deitel
4	Andrew	Goldberg

Fig. 25.17 | Sample data from `authors` in ascending order by `lastName` and `firstName`.

isbn	title	editionNumber	copyright
0132404168	C How to Program	5	2007
0131857576	C++ How to Program	5	2005
0131450913	Internet and World Wide Web How to Program	3	2004
0132222205	Java How to Program	7	2007
0131869000	Visual Basic 2005 How to Program	3	2006
013152539	Visual C# How to Program	2	2006

Fig. 25.18 | Sampling of books from table `titles` whose titles end with `How to Program` in ascending order by `title`.

25.4.4 Merging Data from Multiple Tables: INNER JOIN

Database designers often split related data into separate tables to ensure that a database does not store data redundantly. For example, the `books` database has tables `authors` and `titles`. We use an `authorISBN` table to store the relationship data between authors and their corresponding titles. If we did not separate this information into individual tables, we would need to include author information with each entry in the `titles` table. This

Java™ How to Program, Seventh Edition By H. M. Deitel - Deitel & Associates, Inc., P. J. Deitel - Deitel & Associates, Inc. ISBN: 9780136085676 Publisher: Prentice Hall
Print Publication Date: 2006/12/27

would result in the database storing duplicate author information for authors who wrote multiple books. Often, it is necessary to merge data from multiple tables into a single result. Referred to as joining the tables, this is specified by an `INNER JOIN` operator in the query. An `INNER JOIN` merges rows from two tables by matching values in columns that are common to the tables. The basic form of an `INNER JOIN` is:

```
SELECT columnName1, columnName2, ...
FROM table1
INNER JOIN table2
   ON table1.columnName = table2.columnName
```

The **`ON` clause** of the `INNER JOIN` specifies the columns from each table that are compared to determine which rows are merged. For example, the following query produces a list of authors accompanied by the ISBNs for books written by each author:

```
SELECT firstName, lastName, isbn
FROM authors
INNER JOIN authorISBN
   ON authors.authorID = authorISBN.authorID
ORDER BY lastName, firstName
```

The query merges the `firstName` and `lastName` columns from table `authors` with the `isbn` column from table `authorISBN`, sorting the result in ascending order by `lastName` and `firstName`. Note the use of the syntax *tableName*`.`*columnName* in the `ON` clause. This syntax, called a **qualified name**, specifies the columns from each table that should be compared to join the tables. The "*tableName*`.`" syntax is required if the columns have the same name in both tables. The same syntax can be used in any query to distinguish columns in different tables that have the same name. In some systems, table names qualified with the database name can be used to perform cross-database queries. As always, the query can contain an `ORDER BY` clause. Figure 25.19 depicts a portion of the results of the preceding query, ordered by `lastName` and `firstName`. [*Note:* To save space, we split the result of the query into two columns, each containing the `firstName`, `lastName` and `isbn` columns.]

firstName	lastName	isbn	firstName	lastName	isbn
David	Choffnes	0131828274	Paul	Deitel	0131525239
Harvey	Deitel	0131525239	Paul	Deitel	0132404168
Harvey	Deitel	0132404168	Paul	Deitel	0131869000
Harvey	Deitel	0131869000	Paul	Deitel	0132222205
Harvey	Deitel	0132222205	Paul	Deitel	0131450913
Harvey	Deitel	0131450913	Paul	Deitel	0131525239
Harvey	Deitel	0131525239	Paul	Deitel	0131857576
Harvey	Deitel	0131857576	Paul	Deitel	0131828274
Harvey	Deitel	0131828274	Andrew	Goldberg	0131450913

Fig. 25.19 | Sampling of authors and ISBNs for the books they have written in ascending order by `lastName` and `firstName`.

Java™ How to Program, Seventh Edition By H. M. Deitel - Deitel & Associates, Inc., P. J. Deitel - Deitel & Associates, Inc. ISBN: 9780136085676 Publisher: Prentice Hall
Print Publication Date: 2006/12/27

Software Engineering Observation 25.3

If a SQL statement includes columns with the same name from multiple tables, the statement must precede those column names with their table names and a dot (e.g., `authors.authorID`*).*

Common Programming Error 25.5

Failure to qualify names for columns that have the same name in two or more tables is an error.

25.4.5 INSERT Statement

The `INSERT` statement inserts a row into a table. The basic form of this statement is

```
INSERT INTO tableName ( columnName1, columnName2, ..., columnNameN )
   VALUES ( value1, value2, ..., valueN )
```

where *tableName* is the table in which to insert the row. The *tableName* is followed by a comma-separated list of column names in parentheses (this list is not required if the `INSERT` operation specifies a value for every column of the table in the correct order). The list of column names is followed by the SQL keyword `VALUES` and a comma-separated list of values in parentheses. The values specified here must match the columns specified after the table name in both order and type (e.g., if *columnName1* is supposed to be the `firstName` column, then *value1* should be a string in single quotes representing the first name). Always explicitly list the columns when inserting rows. If the table's column order changes or a new column is added, using only `VALUES` may cause an error. The `INSERT` statement

```
INSERT INTO authors ( firstName, lastName )
   VALUES ( 'Sue', 'Smith' )
```

inserts a row into the `authors` table. The statement indicates that values are provided for the `firstName` and `lastName` columns. The corresponding values are `'Sue'` and `'Smith'`. We do not specify an `authorID` in this example because `authorID` is an autoincremented column in the `authors` table. For every row added to this table, MySQL assigns a unique `authorID` value that is the next value in the autoincremented sequence (i.e., 1, 2, 3 and so on). In this case, Sue Smith would be assigned `authorID` number 5. Figure 25.20 shows the `authors` table after the `INSERT` operation. [*Note:* Not every database management system supports autoincremented columns. Check the documentation for your DBMS for alternatives to autoincremented columns.]

authorID	firstName	lastName
1	Harvey	Deitel
2	Paul	Deitel
3	Andrew	Goldberg
4	David	Choffnes
5	Sue	Smith

Fig. 25.20 | Sample data from table `Authors` after an `INSERT` operation.

Java™ How to Program, Seventh Edition By H. M. Deitel - Deitel & Associates, Inc., P. J. Deitel - Deitel & Associates, Inc. ISBN: 9780136085676 Publisher: Prentice Hall
Print Publication Date: 2006/12/27

Common Programming Error 25.6

It is normally an error to specify a value for an autoincrement column.

Common Programming Error 25.7

SQL uses the single-quote (') character as a delimiter for strings. To specify a string containing a single quote (e.g., O'Malley) in a SQL statement, the string must have two single quotes in the position where the single-quote character appears in the string (e.g., `'O''Malley'`). The first of the two single-quote characters acts as an escape character for the second. Not escaping single-quote characters in a string that is part of a SQL statement is a SQL syntax error.

25.4.6 UPDATE Statement

An `UPDATE` statement modifies data in a table. The basic form of the `UPDATE` statement is

```
UPDATE tableName
   SET columnName1 = value1, columnName2 = value2, ..., columnNameN = valueN
   WHERE criteria
```

where *tableName* is the table to update. The *tableName* is followed by keyword `SET` and a comma-separated list of column name/value pairs in the format *columnName* = *value*. The optional `WHERE` clause provides criteria that determine which rows to update. Though not required, the `WHERE` clause is typically used, unless a change is to be made to every row. The `UPDATE` statement

```
UPDATE authors
   SET lastName = 'Jones'
   WHERE lastName = 'Smith' AND firstName = 'Sue'
```

updates a row in the `authors` table. The statement indicates that `lastName` will be assigned the value `Jones` for the row in which `lastName` is equal to `Smith` and `firstName` is equal to `Sue`. [*Note:* If there are multiple rows with the first name "Sue" and the last name "Smith," this statement will modify all such rows to have the last name "Jones."] If we know the `authorID` in advance of the `UPDATE` operation (possibly because we searched for it previously), the `WHERE` clause can be simplified as follows:

```
WHERE AuthorID = 5
```

Figure 25.21 shows the `authors` table after the `UPDATE` operation has taken place.

authorID	firstName	lastName
1	Harvey	Deitel
2	Paul	Deitel
3	Andrew	Goldberg
4	David	Choffnes
5	Sue	Jones

Fig. 25.21 | Sample data from table `authors` after an `UPDATE` operation.

Java™ How to Program, Seventh Edition By H. M. Deitel - Deitel & Associates, Inc., P. J. Deitel - Deitel & Associates, Inc. ISBN: 9780136085676 Publisher: Prentice Hall
Print Publication Date: 2006/12/27

25.4.7 DELETE Statement

A SQL `DELETE` statement removes rows from a table. The basic form of a `DELETE` is

```
DELETE FROM tableName WHERE criteria
```

where *tableName* is the table from which to delete. The optional `WHERE` clause specifies the criteria used to determine which rows to delete. If this clause is omitted, all the table's rows are deleted. The `DELETE` statement

```
DELETE FROM authors
   WHERE lastName = 'Jones' AND firstName = 'Sue'
```

deletes the row for Sue Jones in the `authors` table. If we know the `authorID` in advance of the `DELETE` operation, the `WHERE` clause can be simplified as follows:

```
WHERE authorID = 5
```

Figure 25.22 shows the `authors` table after the `DELETE` operation has taken place.

authorID	firstName	lastName
1	Harvey	Deitel
2	Paul	Deitel
3	Andrew	Goldberg
4	David	Choffnes

Fig. 25.22 | Sample data from table `authors` after a `DELETE` operation.

25.5 Instructions for installing MySQL and MySQL Connector/J

MySQL 5.0 Community Edition is an open-source database management system that executes on many platforms, including Windows, Solaris, Linux, and Macintosh. Complete information about MySQL is available from `www.mysql.com`. The examples in Section 25.8 and Section 25.9 manipulate MySQL databases.

Installing MySQL

To install MySQL Community Edition:

1. To learn about the installation requirements for your platform, visit the site `dev.mysql.com/doc/refman/5.0/en/general-installation-issues.html`.
2. Visit `dev.mysql.com/downloads/mysql/5.0.html` and download the installer for your platform. For the MySQL examples in this chapter, you need only the Windows Essentials package on Microsoft Windows, or the Standard package on most other platforms. [*Note:* For these instructions, we assume you are running Microsoft Windows. Complete installation instructions for other platforms are available at `dev.mysql.com/doc/refman/5.0/en/installing.html`.]
3. Double click `mysql-essential-5.0.27-win32.msi` to start the installer. [*Note:* This file name may differ based on the current version of MySQL 5.0.]

Java™ How to Program, Seventh Edition By H. M. Deitel - Deitel & Associates, Inc., P. J. Deitel - Deitel & Associates, Inc. ISBN: 9780136085676 Publisher: Prentice Hall
Print Publication Date: 2006/12/27