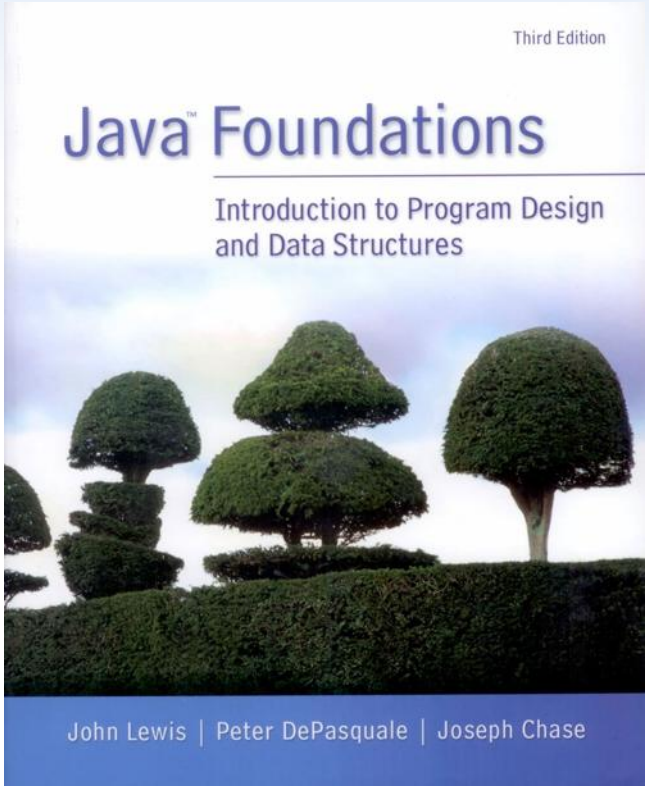

Chapter 25

Databases

Chapter Scope

- Database concepts
- Tables and queries
- SQL statements
- Managing data in a database

Databases

- A *database* is a potentially large repository of data, organized for quick search and manipulation
- A *database management system* (DBMS) is software that interacts with the database
- Four primary operations: create, read, update, and delete (CRUD)
- There are several underlying ways to organize a database
- The most common type: a *relational database*

Database Tables

- Two examples of database tables:

Person			
personID	firstName	LastName	locationID
0	Matthew	Williamson	0
1	Peter	DePasquale	0
2	John	Lewis	1
3	Jason	Smithson	2

Location		
locationID	city	state
0	Portsmouth	RI
1	Blacksburg	VA
2	Maple Glen	PA
3	San Jose	CA

Database Tables

- These tables are related using the `locationID` field
- One location can be associated with multiple people without duplicating data
- This saves space, reduces inconsistencies, and makes updates easier
- We can *query* the database (ask it a question) to determine things like
  - How many people live in Maple Glen?

Database Connections

- Before we can interact with a database in a Java program, we must first establish a connection
- The Java Database Connectivity (JDBC) API provides software to establish the connection
- In addition, we need software called a *driver* for the specific type of database used
- For example, if you're using a MySQL databse, you'll need an appropriate MySQL driver
- SQL (Structured Query Language) is a language for interacting with databases

```
import java.sql.*;

/**
 * Demonstrates the establishment of a JDBC connector.
 *
 * @author Java Foundations
 * @version 4.0
 */
public class DatabaseConnector
{
    /**
     * Establishes the connection to the database and prints an
     * appropriate confirmation message.
     */
    public static void main (String args[])
    {
        try
        {
            Connection conn = null;

            // Loads the class object for the mysql driver into the DriverManager.
            Class.forName("com.mysql.jdbc.Driver");
```

```
            // Attempt to establish a connection to the specified database via the
            // DriverManager
            conn = DriverManager.getConnection("jdbc:mysql://comtor.org/" +
                "javafoundations?user=jf2e&password=hirsch");

            if (conn != null)
            {
                System.out.println("We have connected to our database!");
                conn.close();
            }

        } catch (SQLException ex) {
          System.out.println("SQLException: " + ex.getMessage());
          ex.printStackTrace();
        } catch (Exception ex) {
          System.out.println("Exception: " + ex.getMessage());
          ex.printStackTrace();
        }
    }
}
```

Creating Databases

- A SQL statement called CREATE TABLE can be used to add a new table to the database
- You specify the field names and sizes, among other things

```
CREATE TABLE Student (student_ID INT UNSIGNED NOT
NULL AUTO_INCREMENT, PRIMARY KEY (student_ID),
firstName varchar(255), lastName varchar(255))
```

- This establishes that the `student_ID` field will be the *primary key*, which must be unique

Creating Databases

- To execute this statement from a Java program, you store it as a string and invoke the `execute` method of the `Statement` class (from the JDBC)

```
String myCommand = "CREATE TABLE …";
Statement stmt = conn.createStatement();
boolean result = stmt.execute(myCommand);
```

Altering Tables

- The structure of a database table can be changed with the ALTER TABLE statement
- To add `age` and `gpa` fields to the `Student` table:

```
ALTER TABLE Student ADD COLUMN (age tinyint UNSIGNED,
gpa FLOAT (3,2) unsigned)
```

- To eliminate a field and remove all of its data:

```
ALTER TABLE Student DROP COLUMN firstName
```

Queries

- Data from a query is returned in a `ResultSet` object
- The SHOW COLUMNS statement returns the structure (a list of fields and their attributes) of a table

```
ResultSet rSet = stmt.executeQuery("SHOW COLUMNS
FROM Student");
```

- We can extract meta data from the result set:

```
ResultSetMetaData rsmd = rSet.getMetaData();
int numColumns = rsmd.getColumnCount();
```

Queries

- The metadata for the `Student` table:

Field	Type	Null	Key	Default	Extra
student_ID	int(10) unsigned	NO	PRI	auto_increment	
lastName	varchar(255)	YES			
age	tinyint(3) unsigned	YES			
gpa	float(3, 2) unsigned	YES			

Inserting Data

- The INSERT command is used to add a row of data to a table
- It specifies the columns and their corresponding values:

```
INSERT Student (lastName, age, gpa) VALUES
("Campbell", 19, 3.79)
```

- The `studentID` field is automatically incremented, and therefore not specified

Inserting Data

- Some sample data for the `Student` class:

lastName	age	gpa
Campbell	19	3.79
Garcia	28	2.37
Fuller	19	3.18
Cooper	26	2.13
Walker	27	2.14
Griego	31	2.10

Retrieving Data

- The SELECT … FROM command is used to retrieve specific data from the database
- To get a list of all student's names and GPAs:

```
SELECT lastName, gpa FROM Student
```

- A * can be used to get all fields
- The WHERE clause further specifies which data is sought:

```
SELECT * FROM Student WHERE age >= 21 && gpa <= 3.0
```

Updating Data

- To update existing data in a table, we:
  - Obtain a `ResultSet` and navigate to the row(s) to update
  - Update the `ResultSet` value(s)
  - Update the database with the revised `ResultSet`

```
ResultSet rSet = stmt.executeQuery("SELECT * FROM
Student WHERE lastName = \"Jones\"");
rSet.first()
rSet.updateFloat("gpa", 3.41f);
rSet.UpdateRow();
```

```
import java.sql.*;

/**
 * Demonstrates interaction between a Java program and a database.
 *
 * @author Java Foundations
 * @version 4.0
 */
public class DatabaseModification
{
    /**
     * Carries out various CRUD operations after establishing the
     * database connection.
     */
    public static void main (String args[])
    {
        Connection conn = null;
        try
        {
            // Loads the class object for the mysql driver into the DriverManager.
            Class.forName("com.mysql.jdbc.Driver");

            // Attempt to establish a connection to the specified database via the
            // DriverManager
            conn = DriverManager.getConnection("jdbc:mysql://comtor.org/" +
                "javafoundations?user=jf2e&password=hirsch");
```

```
        // Check the connection
        if (conn != null)
        {
            System.out.println("We have connected to our database!");

            // Create the table and show the table structure
            Statement stmt = conn.createStatement();
            boolean result = stmt.execute("CREATE TABLE Student " +
                " (student_ID INT UNSIGNED NOT NULL AUTO_INCREMENT, " +
                "  PRIMARY KEY (student_ID), lastName varchar(255), " +
                "  age tinyint UNSIGNED, gpa FLOAT (3,2) unsigned)");

            System.out.println("\tTable creation result: " + result);
            DatabaseModification.showColumns(conn);

            // Insert the data into the database and show the values in the table
            Statement stmt2 = conn.createStatement(ResultSet.TYPE_FORWARD_ONLY,
                ResultSet.CONCUR_UPDATABLE);
            int rowCount = stmt2.executeUpdate("INSERT Student " +
                "(lastName, age, gpa) VALUES (\"Campbell\", 19, 3.79)");
            DatabaseModification.showValues(conn);

            // Close the database
            conn.close();
        }
```

```
    } catch (SQLException ex) {
        System.out.println("SQLException: " + ex.getMessage());
        ex.printStackTrace();
    } catch (Exception ex) {
        System.out.println("Exception: " + ex.getMessage());
        ex.printStackTrace();
    }
}

/**
 * Obtains and displays a ResultSet from the Student table.
 */
public static void showValues(Connection conn)
{
    try
    {
        Statement stmt = conn.createStatement();
        ResultSet rset = stmt.executeQuery("SELECT * FROM Student");
        DatabaseModification.showResults("Student", rset);
    } catch (SQLException ex) {
        System.out.println("SQLException: " + ex.getMessage());
        ex.printStackTrace();
    }
}
```

```
/**
 * Displays the structure of the Student table.
 */
public static void showColumns(Connection conn)
{
    try
    {
        Statement stmt = conn.createStatement();
        ResultSet rset = stmt.executeQuery("SHOW COLUMNS FROM Student");
        DatabaseModification.showResults("Student", rset);
    } catch (SQLException ex) {
        System.out.println("SQLException: " + ex.getMessage());
        ex.printStackTrace();
    }
}
```

```
/**
 * Displays the contents of the specified ResultSet.
 */
public static void showResults(String tableName, ResultSet rSet)
{
    try
    {
        ResultSetMetaData rsmd = rSet.getMetaData();
        int numColumns = rsmd.getColumnCount();
        String resultString = null;
        if (numColumns > 0)
        {
            resultString = "\nTable: " + tableName + "\n" +
              "=======================================================\n";
            for (int colNum = 1; colNum <= numColumns; colNum++)
                resultString += rsmd.getColumnLabel(colNum) + "     ";
        }
        System.out.println(resultString);
        System.out.println(
            "=======================================================");
```

```
            while (rSet.next())
            {
                resultString = "";
                for (int colNum = 1; colNum <= numColumns; colNum++)
                {
                    String column = rSet.getString(colNum);
                    if (column != null)
                        resultString += column + "     ";
                }
                System.out.println(resultString + '\n' +
                    "-------------------------------------------------------");
            }
        } catch (SQLException ex) {
            System.out.println("SQLException: " + ex.getMessage());
            ex.printStackTrace();
        }
    }
}
```

Deleting Data

- To delete specific data from a table:

```
DELETE FROM Student WHERE age >= 30
```

- To delete an entire table:

```
DROP TABLE Student
```