Chapter 17 Lab
Databases

Lab Objectives

- Be able to write SQL statements.
- Be able to retrieve information from a database.
- Be able to add information to a database.
- Be able to change information in multiple tables of the database.

Introduction

A recreation director needs to keep track of summer T-ball teams, games, and season records. He has hired you to write a computer program to make his life easier. You have created a menu driven program that allows the director to use a database to easily manipulate teams, games, and scores. It also allows the database to be refreshed for a new year.

Complete the tasks below to enable all the functionality that is required for this application.

Task #1 Database Connection and Initialization

1. Copy the file *TeamDB.java* (see Code Listing 17.1) from the Student CD or as directed by your instructor.
2. Modify the `DB_URL` constant at the beginning of the `main` method as directed by your instructor to connect with your database correctly.
3. Create and initialize the database by running the program and selecting option 6.
4. If, at any point after the initialization, you wish to start over, you can select option 6, which will remove the database tables and recreate and initialize them.
5. The `Teams` table created and initialized in the database is as follows:

TeamName	Wins	Losses	Ties
Astros	0	0	0
Brewers	0	0	0
Cubs	0	0	0
Marlins	0	0	0

6. The `Games` table created an initialized in the database is as follows:

GameNumber	HomeTeam	HomeTeamScore	VisitorTeam	VisitorTeamScore
1	Astros	0	Brewers	0
2	Brewers	0	Cubs	0
3	Cubs	0	Astros	0

Task #2 Retrieve Information Using SQL

1. Select option 1 to view the teams. Your output should be as follows:
```
Team Name           Win       Lose      Tie
Astros                0          0        0
Brewers               0          0        0
Cubs                  0          0        0
Marlins               0          0        0
```
2. Write the implementation for the `viewSchedule` method.
   a. Write an SQL statement to select all columns from the `Games` table.
   b. Execute the query by sending the SQL statement to the DBMS.
   c. Use a `while` loop to display the contents of the result set to the monitor.
   d. Compile and run the program. Select option 2 and your output should be as follows:
```
GameID    Home      Score     Visitor   Score
1         Astros        0     Brewers       0
2         Brewers       0     Cubs          0
3         Cubs          0     Astros        0
```

Task #3 Add Information Using SQL

1. Write the body of the loop in the `addTeams` method. You should prompt the user for the new team name, write the SQL statement and update the `Teams` table with the new team name.
2. Compile and run the program. Select option 3 and enter **Braves**. Enter **N** to stop entering names. Select option 1 again to view the teams. Your output should now look like:
```
Team Name           Win       Lose      Tie
Astros                0          0        0
Braves                0          0        0
Brewers               0          0        0
Cubs                  0          0        0
Marlins               0          0        0
```

Task #4 Change Information in Multiple Tables Using SQL

1. Edit the `enterScores` method. Execute a query using the appropriate SQL statement to get the result set containing all information for the `gameNumber` the user entered.
2. After the user inputs the scores for both teams, execute updates to the `Games` table using the appropriate SQL statements.
3. In each branch of the logic determining winner, loser or tie: retrieve the number from the appropriate column (losses, wins, or ties) from the `Teams` table for one of the teams, increment it, and update the table for that team. Do the same for the opposing team.

4. Select option 5 to enter scores. Enter scores as follows:

Game 1	Astros	5	Brewers	3
Game 2	Brewers	4	Cubs	4
Game 3	Cubs	1	Astros	6

5. Select option 1 to view the team standings and option 2 to see the scores. Your output should now be:

```
Team Name           Win        Lose       Tie
Astros                2           0         0
Braves                0           0         0
Brewers               0           1         1
Cubs                  0           1         1
Marlins               0           0         0

List of games and scores:
GameID    Home      Score     Visitor   Score
1         Astros        5     Brewers       3
2         Brewers       4     Cubs          4
3         Cubs          1     Astros        6
```

<u>Code Listing 17.1 (`TeamDB.java`)</u>

```
import java.util.Scanner;
import java.sql.*;

/**
   This program uses a database to keep track of
   summer T-ball teams, games, and season records.
*/

public class TeamDB
{
   public static void main(String[] args)
   {
      // Create a named constant for the database URL
      // NOTE: This value is specific for Java DB
      final String DB_URL = "jdbc:derby:TeamDB;" +
                            "create=true";

      Connection conn;

      try
      {
         // Create a connection to the database
         conn = DriverManager.getConnection(URL);

         char choice;
         Scanner keyboard = new Scanner(System.in);
         System.out.println("Welcome to the Sports " +
                            "Teams Database Manager!");
         do
         {
            printMenu();
            choice = keyboard.nextLine().charAt(0);
            switch(choice)
            {
               case '0':
                  // Close the connection
                  conn.close();
                  break;
               case '1':
                  viewTeams(conn);
                  break;
               case '2':
                  viewSchedule(conn);
                  break;
               case '3':
                  addTeams(conn);
                  break;
               case '4':
                  addGames(conn);
                  break;
               case '5':
                  enterScores(conn);
                  break;
               case '6':
                  beginNewSeason(conn);
                  break;
            }
         } while(choice != '0');
      }

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

   /**
      The printMenu method displays the menu choices
      for the user to work with the database.
   */
```

```java
public static void printMenu()
{
   System.out.println();
   System.out.println("Select from the following " +
                      "options:");
   System.out.println("1. View team standings");
   System.out.println("2. View the schedule");
   System.out.println("3. Add a team");
   System.out.println("4. Add a game to the schedule");
   System.out.println("5. Enter game scores");
   System.out.println("6. Begin a new season");
   System.out.println("0. Exit the program");
}

/**
   The beginNewSeason method is a utility method that
   removes the tables and allows the user to reset the
   database for a new season.
   @param conn A connection to the database.
*/

public static void beginNewSeason(Connection conn)
{
   try
   {
      Statement stmt = conn.createStatement();

      // Remove the tables if they already exist
      // Throws an exception if the tables do not exist
      stmt.execute("DROP TABLE Games");
      stmt.execute("DROP TABLE Teams");

      // Once the tables have been removed, call the
      // method to create and initialize the tables
      System.out.println("Reinitializing database " +
                         "for a new season");
      createTeamDB(conn);
   }

   catch(Exception ex)
   {
      // Create the tables if they do not exist
      System.out.println("Creating database for " +
                         "the first time");
      createTeamDB(conn);
   }
}
```

```
/**
   The createTeamDB method is a utility method that
   creates the tables and initializes the database
   with teams and games.
   @param conn A connection to the database.
*/

public static void createTeamDB(Connection conn)
{
   try
   {
      Statement stmt = conn.createStatement();

      // Create the table of teams
      stmt.execute("CREATE TABLE Teams (" +
                   "TeamName CHAR(15) " +
                   "NOT NULL PRIMARY KEY, " +
                   "Wins INT, " + "Losses INT, " +
                   "Ties INT" + ")");

      // Add some teams
      stmt.executeUpdate("INSERT INTO Teams " +
                         "(TeamName) " +
                         "VALUES ('Astros')");

      stmt.executeUpdate("INSERT INTO Teams " +
                         "(TeamName) " +
                         "VALUES ('Marlins')");

      stmt.executeUpdate("INSERT INTO Teams " +
                         "(TeamName) " +
                         "VALUES ('Brewers')");

      stmt.executeUpdate("INSERT INTO Teams " +
                         "(TeamName) " +
                         "VALUES ('Cubs')");

      // Create a listing of the games to be played
      stmt.execute("CREATE TABLE Games (" +
                   "GameNumber INT " +
                   "NOT NULL PRIMARY KEY, " +
                   "HomeTeam CHAR(15) " +
                   "NOT NULL REFERENCES " +
                   "Teams (TeamName), " +
                   "HomeTeamScore INT, " +
                   "VisitorTeam CHAR(15) NOT NULL " +
                    "REFERENCES Teams (TeamName), " +
                    "VisitorTeamScore INT" + ")");

         stmt.executeUpdate("INSERT INTO Games " +
                            "(GameNumber, HomeTeam, " +
                            "VisitorTeam) " +
                            "VALUES (1, 'Astros', " +
                            "'Brewers')");

         stmt.executeUpdate("INSERT INTO Games " +
                            "(GameNumber, HomeTeam, " +
                            "VisitorTeam) " +
                            "VALUES (2, 'Brewers', " +
                            "'Cubs')");

         stmt.executeUpdate("INSERT INTO Games " +
                            "(GameNumber, HomeTeam, " +
                            "VisitorTeam) " +
                            "VALUES (3, 'Cubs', " +
                            "'Astros')");
      }

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

   /**
      The addTeams method allows the user to add
      more teams to the database.
      @param conn A connection to the database.
   */

   public static void addTeams(Connection conn)
   {
      Scanner keyboard = new Scanner(System.in);

      try
      {
         char ans;
         String teamName;

         Statement stmt = conn.createStatement();
         do
         {
            // ADD LINES FOR TASK #3 HERE
            // Prompt the user for a new team name
            // Update the Teams table

            System.out.print("Do you want to enter " +
                             "another team: ");
            ans = keyboard.nextLine().charAt(0);

         } while(ans == 'Y'|| ans == 'y');
      }

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

   /**
      The addGames method allows the user to add games to
      the schedule. A unique game number is created for
      each game on the schedule.The user will need to
      supply a home team name and a visitor team name
      from the keyboard.
      @param conn A connection to the database.
   */

   public static void addGames(Connection conn)
   {
      Scanner keyboard = new Scanner (System.in);

      try
      {
         char ans;
         String homeTeam;
         String visitingTeam;
         int gameNumber = 1;

         Statement stmt = conn.createStatement();

         // Retrieve the data and count the number of
         // games already scheduled so that you can
         // add a unique game number
         ResultSet result = stmt.executeQuery("SELECT * " +
                                              "FROM " +
                                              "Games");
      while(result.next())
      {
         gameNumber++;
      }

      do
      {
         System.out.print("Enter the home " +
                          "team name: ");
         homeTeam = keyboard.nextLine();

         System.out.print("Enter the visiting " +
                          "team name: ");
         visitingTeam = keyboard.nextLine();

         stmt.executeUpdate("INSERT INTO Games " +
                            "(GameNumber, HomeTeam, " +
                            "VisitorTeam) " +
                            "VALUES (" + gameNumber +
                            ", '" + homeTeam + "', '" +
                            visitingTeam + "')");


         System.out.print("Do you want to enter " +
                          "another game: ");
         ans = keyboard.nextLine().charAt(0);

      } while(ans == 'Y'|| ans == 'y');
   }

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

/**
   The viewTeams method displays a table listing the
   team names and season records. Since teams have not
   yet played, all numbers are zero.
   @param conn A connection to the database.
*/
```

```
public static void viewTeams(Connection conn)
{
   try
   {
      // Create a Statement object
      Statement stmt = conn.createStatement();

      // Send the statement to the DBMS
      ResultSet result = stmt.executeQuery("SELECT * " +
                                           "FROM " +
                                           "Teams");

      System.out.printf("%-15s %10s %10s %10s\n",
                        "Team Name", "Win", "Lose",
                        "Tie");

      // Display the contents of the result set
      // The result set will have 5 columns
      while(result.next())
      {
         System.out.printf("%-15s %10d %10d %10d\n",
                           result.getString("TeamName"),
                           result.getInt("Wins"),
                           result.getInt("Losses"),
                           result.getInt("Ties"));
      }
   }

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

/**
   The viewSchedule method retrieves and displays the
   teams and scores for all games.
   @param conn A connection to the database.
*/

public static void viewSchedule(Connection conn)
{
   try
   {
      // Create a Statement object
      Statement stmt = conn.createStatement();
      // ADD LINES FOR TASK #2 HERE
      // Create a string with a SELECT statement
      // Send the statement to the DBMS

      // This is a suggested column headings display
      System.out.println("List of games and scores:");
      System.out.printf("%-6s %-20s %6s     " +
                        "%-20s %6s\n",
                        "GameID", "Home", "Score",
                        "Visitor", "Score");

      // ADD LINES FOR TASK #2 HERE
      // Use a while loop to display the result set
      // The result set will have five columns
   }

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

/**
   The enterScores method allows user to enter scores
   for both teams. The method will update the Games
   table with the scores entered. It will also compare
   the scores to determine the winning and losing teams
   (or tie) and update the appropriate column in the
   Teams table for each team involved in the game.
   @param conn A connection to the database.
*/

public static void enterScores(Connection conn)
{
   Scanner keyboard = new Scanner(System.in);
   try
   {
      char ans;
      int gameNumber;
      String homeTeam;
      String visitingTeam;
      int score1;
      int score2;
      String sqlStatement;
      ResultSet result = null;

      Statement stmt = conn.createStatement();
```

```
do
{
   viewSchedule(conn);

   System.out.print("Enter the game ID: ");
   gameNumber = keyboard.nextInt();

   // ADD LINES FOR TASK #4 HERE
   // Get the result set from a query that
   // selects all information for the gameNumber
   // the user entered

   if(result.next())
   {
      homeTeam = result.getString("HomeTeam");
      visitingTeam = result.getString("Visitor" +
                                      "Team");

      System.out.print("Enter the score " +
                       "for the " + homeTeam);
      score1 = keyboard.nextInt();

      System.out.print("Enter the score " +
                       "for the " + visitingTeam);
      score2 = keyboard.nextInt();
      keyboard.nextLine();

      // ADD LINES FOR TASK #4 HERE
      // Execute an update to the Games table to
      // store the score for each team of that
      // game number

      if(score1 < score2)
      {
         // ADD LINES FOR TASK #4 HERE
         // Retrieve the number from the
         // appropriate column
         // (wins, losses, or ties) for the
         // home team, increment, and update
         // that team's record.
         // Do the same for the visiting team
      }
```

```
               else if(score2 < score1)
               {
                  // ADD LINES FOR TASK #4 HERE
                  // Retrieve the number from the
                  // appropriate column
                  // (wins, losses, or ties) for the
                  // home team, increment, and update
                  // that team's record.
                  // Do the same for the visiting team
               }
               else
               {
                  // ADD LINES FOR TASK #4 HERE
                  // Retrieve the number from the
                  // appropriate column
                  // (wins, losses, or ties) for the
                  // home team, increment, and update
                  // that team's record.
                  // Do the same for the visiting team
               }
            }

            System.out.print("Do you want to enter " +
                             "another game: ");
            ans = keyboard.nextLine().charAt(0);

         } while(ans == 'Y'|| ans == 'y');
      }

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