



Romel Cabiling ▾



Home

Home > My courses > 121 - CC106 > MODULE 16: WHAT ARE THE STEPS OF MANIPULATING DATA... > Lesson Proper for Week 16

Lesson Proper for Week 16

What are the steps of Manipulating database using java Database Connectivity?

Java JDBC Driver for Microsoft Access Database

There are several third-party JDBC drivers out there for Microsoft Access database, and we recommend **UCanAccess** - a pure Java JDBC Driver for Access that allows Java developers and JDBC client programs to read/write Microsoft Access databases. UCanAccess supports various Access formats: 2000, 2002/2003, 2007, 2010/2013/2016 (Access 97 is supported for read-only).

UCanAccess is open-source and implemented entirely in Java so it can be used across platforms (Windows, Mac, Linux...). It also provides Maven dependencies so you can integrate it in your existing projects quickly.

To use UCanAccess JDBC Driver for Access, add the following dependency information in your project's pom.xml file:

```
<artifactId>ucanaccess</artifactId>
<version>4.0.4</version>
</dependency>
```

In case you don't use Maven, you have to download UCanAccess distribution and add the following JAR files to the classpath:

- ucanaccess-4.0.4.jar
- hsqldb-2.3.1.jar
- jackcess-2.1.11.jar
- commons-lang-2.6.jar
- commons-logging-1.1.3.jar



Once a connection is obtained we can interact with the database. The JDBC *Statement*, *CallableStatement*, and *PreparedStatement* interfaces define the methods and properties that enable you to send SQL or PL/SQL commands and receive data from your database.

They also define methods that help bridge data type differences between Java and SQL data types used in a database.

Interfaces	Recommended Use
Statement	Use this for general-purpose access to your database. Useful when you are using static SQL statements at runtime. The Statement interface cannot accept parameters.
PreparedStatement	Use this when you plan to use the SQL statements many times. The PreparedStatement interface accepts input parameters at runtime.
CallableStatement	Use this when you want to access the database stored procedures. The CallableStatement interface can also accept runtime input parameters.

The following table provides a summary of each interface's purpose to decide on the interface to use.

The Statement Objects Creating Statement Object Before you can use a Statement object to execute a SQL statement, you need to create one using the Connection object's `createStatement()` method, as in the following example –

```
Statement stmt = null;
try {
    stmt = conn.createStatement();
}
catch (SQLException e) {
}
finally {
}
```

Once you've created a Statement object, you can then use it to execute a SQL statement with one of its three execute methods.

- **boolean execute (String SQL):** Returns a boolean value of true if a ResultSet object can be retrieved; otherwise, it returns false. Use this method to execute SQL DDL statements or when you need to use truly dynamic SQL.
- **int executeUpdate (String SQL):** Returns the number of rows affected by the execution of the SQL statement. Use this method to execute SQL statements for which you expect to get a number of rows affected - for example, an INSERT, UPDATE, or DELETE statement.
- **ResultSet executeQuery (String SQL):** Returns a ResultSet object. Use this method when you expect to get a result set, as you would with a SELECT statement.

Closing Statement Object



Just as you close a Connection object to save database resources, for the same reason you should also close the Statement object.

A simple call to the close() method will do the job. If you close the Connection object first, it will close the Statement object as well. However, you should always explicitly close the Statement object to ensure proper cleanup.

```
Statement stmt = null;
try {
    stmt = conn.createStatement();
}
catch (SQLException e) {
}
finally {
    stmt.close();
}
```

The PreparedStatement Objects

The *PreparedStatement* interface extends the Statement interface, which gives you added functionality with a couple of advantages over a generic Statement object.

This statement gives you the flexibility of supplying arguments dynamically.

Creating PreparedStatement Object

```
PreparedStatement pstmt = null;
try {
    String SQL = "Update Employees SET age = ? WHERE id = ?";
    pstmt = conn.prepareStatement(SQL);
}
catch (SQLException e) {
}
finally {
```

All parameters in JDBC are represented by the ? symbol, which is known as the parameter marker. You must supply values for every parameter before executing the SQL statement.

The **setXXX()** methods bind values to the parameters, where **XXX** represents the Java data type of the value you wish to bind to the input parameter. If you forget to supply the values, you will receive an SQLException.

Each parameter marker is referred by its ordinal position. The first marker represents position 1, the next position 2, and so forth. This method differs from that of Java array indices, which starts at 0.

All of the **Statement object's** methods for interacting with the database (a) execute(), (b) executeQuery(), and (c) executeUpdate() also work with the PreparedStatement object. However, the methods are modified to use SQL statements that can input the parameters.

Closing PreparedStatement Object

Just as you close a Statement object, for the same reason you should also close the PreparedStatement object.



A simple call to the close() method will do the job. If you close the Connection object first, it will close the PreparedStatement object as well. However, you should always explicitly close the PreparedStatement object to ensure proper cleanup.

```
PreparedStatement pstmt = null;
try {
    String SQL = "Update Employees SET age = ? WHERE id = ?";
    pstmt = conn.prepareStatement(SQL);
}
catch (SQLException e) {
}
finally {
    pstmt.close();
}
```

The version numbers here may differ than the latest versions you downloaded. Now, let's see how to write a simple Java program to read/write a Microsoft Access database.

The CallableStatement Objects

Just as a Connection object creates the Statement and PreparedStatement objects, it also creates the CallableStatement object, which would be used to execute a call to a database stored procedure.

Creating CallableStatement Object

Suppose, you need to execute the following Oracle stored procedure –

```
CREATE OR REPLACE PROCEDURE getEmpName
(EMP_ID IN NUMBER, EMP_FIRST OUT VARCHAR) AS
BEGIN
    SELECT first INTO EMP_FIRST
    FROM Employees
    WHERE ID = EMP_ID;
END;
```

This module presented samples programs to show how to manipulate MS ACCESS database using JDBC with jdbc.UcanaccessDriver driver. This an open source JDBC driver and easy to use and understand so everybody will be able to run the programs without difficulty.

1. Example program to connect java to database with SELECT query using MsAccess



```

package sampleConnection;

import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;
import java.sql.Connection;
import java.sql.ResultSet;

public class DatabaseConnectionExample {

    public static void main(String[] args) {

        // variables
        Connection connection = null;
        Statement statement = null;
        ResultSet resultSet = null;

        // Step 1: Loading or registering Oracle JDBC driver class
        try {
            Class.forName("net.ucanaccess.jdbc.UcanaccessDriver");
            System.out.println("Loading Success!");
        } //end of try
        catch (ClassNotFoundException cnfex) {
            System.out.println("Problem in loading MS Access JDBC driver");
            cnfex.printStackTrace();
        } //end of catch

        // Step 2: Opening database connection
        try {
            String dbURL =
"jdbc:ucanaccess://C:/Users/MyComputer/Desktop/Student.accdb";

            // Step 2.A: Create and get connection using DriverManager class
            connection = DriverManager.getConnection(dbURL);

            // Step 2.B: Creating JDBC Statement
            statement = connection.createStatement();

            // Step 2.C: Executing SQL & retrieve data into ResultSet
            resultSet = statement.executeQuery("SELECT * FROM PLAYER");

            System.out.println("ID\tName\tAge\tMatches");
            System.out.println("=====\t=====\t=====\t=====");

```



```

// processing returned data and printing into console
while(resultSet.next()) {
    System.out.println(resultSet.getInt(1) + "\t" +
        resultSet.getString(2) + "\t" +
        resultSet.getString(3) + "\t" +
        resultSet.getString(4));
} // end of while loop
} // end of try
catch(SQLException sqlex){
    sqlex.printStackTrace();
} // end of catch
finally {

    // Step 3: Closing database connection
    try {
        if(null != connection) {
            // cleanup resources, once after processing
            resultSet.close();
            statement.close();
            // and then finally close connection
            connection.close();
        }
    } //end of try
    catch (SQLException sqlex) {
        sqlex.printStackTrace();
    } //end of catch
} //end of finally
} //end of main
} //end of DatabaseConnectionExample.java

```

Program Output

```

Loading Success!
ID          Name          Age    Course
==          =====    ==    =====
2019-MN-1   Lopez, Gerald   20     BSCS
2019-MN-2   Reyes, Luigi    18     BSIT
2019-MN-3   Legaspi, Sean   19     BSIS
2019-MN-4   Abandia, Bea   21     BSCE
2019-MN-5   Vergara, Kate   20     BSA

```

2. Example program with SELECT and INSERT QUERY using MsAccess



```

package sampleConnection;

import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.sql.Statement;
import java.sql.Connection;
import java.sql.ResultSet;
import java.util.Scanner;

public class InsertToDatabaseExample {
    public static void main(String[] args) {

        //variables
        Connection connection = null;
        Statement statement = null;
        PreparedStatement preparedstatement = null; //variable for insert statement
        ResultSet resultSet = null;
        Scanner sc = new Scanner(System.in); // variable for scanner
        String studentId = ""; //variable for new student ID
        String studentName = ""; //variable for new student name
        int age = 0; //variable for new age
        String course = ""; //variable for new course

        // Step 1: Loading or registering Oracle JDBC driver class
        try {
            Class.forName("net.ucanaccess.jdbc.UcanaccessDriver");
            System.out.println("Loading Success!");
        } //end of try
        catch(ClassNotFoundException cnfex) {
            System.out.println("Problem in loading MS Access JDBC driver");
            cnfex.printStackTrace();
        } //end of catch

        // Step 2: Opening database connection
        try {
            String dbURL =
                "jdbc:ucanaccess://C:/Users/MyComputer/Desktop/Student.accdb";

            // Step 2.A: Create and get connection using DriverManager class
            connection = DriverManager.getConnection(dbURL);
            statement = connection.createStatement();

```



```

resultSet = statement.executeQuery("SELECT * FROM STUDENT");

System.out.println(); //for new line in console
System.out.println("Old records");
System.out.println("ID\tName\tAge\tCourse");

System.out.println("=====\t=====\t=====\t=====");

// Retrieving old data and printing into console
while(resultSet.next()) {
    System.out.println(resultSet.getString(1) + "\t" +
        resultSet.getString(2) + "\t" +
        resultSet.getInt(3) + "\t" +
        resultSet.getString(4));
} // end of while loop

// Step 2.B: Creating JDBC Statement for insert statement and data to be
inserted
System.out.println(); //for new line in console
System.out.println(); //for new line in console
System.out.print("Enter new Student ID: ");
studentId = sc.nextLine();
System.out.print("Enter student name: ");
studentName = sc.nextLine();
System.out.print("Enter student course: ");
course = sc.nextLine();
System.out.print("Enter student age: ");
age = sc.nextInt();

//Executing the insert query
preparedstatement = connection.prepareStatement("INSERT INTO
    STUDENT(StudentId,StudentName,Age,Course)VALUES(?,?,?,?)");
preparedstatement.setString(1,studentId);
preparedstatement.setString(2, studentName);
preparedstatement.setInt(3,age);
preparedstatement.setString(4, course);
preparedstatement.execute();

System.out.println("Data added succesfully");

// Step 2.C: Creating a JDBC statement for Select statement
// Executing SQL & retrieve data into ResultSet
statement = connection.createStatement();
resultSet = statement.executeQuery("SELECT * FROM STUDENT");

System.out.println("Records after insert query.");
System.out.println("ID\tName\tAge\tCourse");

```




```

System.out.println("====\t\t====\t\t====\t\t====");

// processing returned data and printing into console
while(resultSet.next()) {
    System.out.println(resultSet.getString(1) + "\t" +
        resultSet.getString(2) + "\t" +
        resultSet.getInt(3) + "\t" +
        resultSet.getString(4));
}

// end of while loop
// end of try
catch(SQLException sqlex){
    sqlex.printStackTrace();
}

// end of catch
finally {
    // Step 3: Closing database connection
    try {
        if(null != connection) {
            // cleanup resources, once after processing
            resultSet.close();
            statement.close();

            // and then finally close connection
            connection.close();
        }
    }

    //end of try
    catch (SQLException sqlex) {
        sqlex.printStackTrace();
    }

    //end of catch
}

//end of finally
//end of main
//end of DatabaseConnectionExample.java

```

PROGRAM OUTPUT

Loading Success!

Old records

ID	Name	Age	Course
2019-MN-1	Lopez, Gerald	20	BSCS
2019-MN-2	Reyes, Luigi	18	BSIT
2019-MN-3	Legaspi, Sean	19	BSIS
2019-MN-4	Abandia, Bea	21	BSA
2019-MN-5	Vergara, Kate	20	BSCE

Enter new Student ID: 2019-MN-6

Enter student name: Atienza, Faye

Enter student course: BSME

Enter student age: 18

Data added successfully

Records after insert query.

ID	Name	Age	Course
2019-MN-1	Lopez, Gerald	20	BSCS
2019-MN-2	Reyes, Luigi	18	BSIT
2019-MN-3	Legaspi, Sean	19	BSIS
2019-MN-4	Abandia, Bea	21	BSA
2019-MN-5	Vergara, Kate	20	BSCE
2019-MN-6	Atienza, Faye	18	BSME

3. Example program with SELECT and UPDATE query using MsAccess



```

package sampleConnection;

import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.sql.Statement;
import java.sql.Connection;
import java.sql.ResultSet;
import java.util.Scanner;

public class UpdateToDatabaseExample {
    public static void main(String[] args) {

        // variables
        Connection connection = null;
        Statement statement = null;
        PreparedStatement preparedstatement = null; // variable for insert statement
        ResultSet resultSet = null;

        Scanner sc = new Scanner(System.in); // variable for scanner
        String studentId = ""; //variable for selected student ID
        String course = ""; //variable for new course

        // Step 1: Loading or registering Oracle JDBC driver class

        try {
            Class.forName("net.ucanaccess.jdbc.UcanaccessDriver");
            System.out.println("Loading Success!");
        } //end of try
        catch(ClassNotFoundException cnfex) {
            System.out.println("Problem in loading MS Access JDBC driver");
            cnfex.printStackTrace();
        } //end of catch

        // Step 2: Opening database connection

        try {
            String dbURL =
                "jdbc:ucanaccess://C:/Users/MyComputer/Desktop/Student.accdb";

            // Step 2.A: Create and get connection using DriverManager class
            connection = DriverManager.getConnection(dbURL);

            statement = connection.createStatement();
            resultSet = statement.executeQuery("SELECT * FROM STUDENT");

            System.out.println("Records before update query.");
            System.out.println("ID\tName\tAge\tCourse");

            System.out.println("=====\t=====\t=====\t=====");

            // processing returned data and printing into console
            while(resultSet.next()) {
                System.out.println(resultSet.getString(1) + "\t" +
                    resultSet.getString(2) + "\t" +
                    resultSet.getInt(3) + "\t" +
                    resultSet.getString(4));
            } // end of while loop

            // Step 2.B: Creating JDBC Statement for update statement and selecting data
            // to be updated
            System.out.println(); //for new line in console

```





```

Loading Success!
Records before update query.
ID          Name          Age      Course
==          =====      ==      =====
2019-MN-1   Lopez, Gerald   20      BSCS
2019-MN-2   Reyes, Luigi    18      BSIT
2019-MN-3   Legaspi, Sean   19      BSIS
2019-MN-4   Abandia, Bea   21      BSA
2019-MN-5   Vergara, Kate   20      BSCE
2019-MN-6   Atienza, Faye   18      BSME

```

```

Enter Student ID to be updated: 2019-MN-4
Enter new course: BSIT

```

```

Update Successful!
Records after update query.
ID          Name          Age      Course
==          =====      ==      =====
2019-MN-1   Lopez, Gerald   20      BSCS
2019-MN-2   Reyes, Luigi    18      BSIT
2019-MN-3   Legaspi, Sean   19      BSIS
2019-MN-4   Abandia, Bea   21      BSIT
2019-MN-5   Vergara, Kate   20      BSCE
2019-MN-6   Atienza, Faye   18      BSME

```

4. Example program with SELECT and DELETE query using MsAccess

```

package sampleConnection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.sql.Statement;
import java.sql.Connection;
import java.sql.ResultSet;
import java.util.Scanner;

public class DeleteInDatabaseExample {
    public static void main(String[] args) {

        // variables
        Connection connection = null;
        Statement statement = null;
        PreparedStatement preparedstatement = null; // variable for delete statement
        ResultSet resultSet = null;

```



```
Scanner sc = new Scanner(System.in); // variable for scanner
String studentId = ""; //variable for selected student ID to be deleted
```

// Step 1: Loading or registering Oracle JDBC driver class

```
try {
    Class.forName("net.ucanaccess.jdbc.UcanaccessDriver");
    System.out.println("Loading Success!");
} //end of try
catch(ClassNotFoundException cnfex) {
    System.out.println("Problem in loading MS Access JDBC driver");
    cnfex.printStackTrace();
} //end of catch
```

// Step 2: Opening database connection

```
try {
    String dbURL =
"jdbc:ucanaccess://C:/Users/MyComputer/Desktop/Student.accdb";
```

// Step 2.A: Create and get connection using DriverManager class

```
connection = DriverManager.getConnection(dbURL);
```

```
statement = connection.createStatement();
resultSet = statement.executeQuery("SELECT * FROM STUDENT");
```

```
System.out.println("Records before delete query.");
System.out.println("ID\tName\tAge\tCourse");
```

```
System.out.println("=====\t=====\t=====\t=====");
```

```
// processing returned data and printing into console
```

```
while(resultSet.next()) {
    System.out.println(resultSet.getString(1) + "\t" +
        resultSet.getString(2) + "\t" +
        resultSet.getInt(3) + "\t" +
        resultSet.getString(4));
} // end of while loop
```

// Step 2.B: Creating JDBC Statement for delete statement and selecting data to be deleted

```
System.out.println(); //for new line in console
System.out.println(); //for new line in console
System.out.print("Enter Student ID to be deleted: ");
studentId = sc.nextLine();
preparedstatement = connection.prepareStatement("DELETE FROM STUDENT WHERE
```



```
StudentId = "" + studentId + """);
preparedstatement.execute();

System.out.println(); //for new line in console
System.out.println("Update Successful!");

// Step 2.C: Creating a JDBC statement for Select statement
// Executing SQL & retrieve data into ResultSet

resultSet = statement.executeQuery("SELECT * FROM STUDENT");

System.out.println("Records after update query.");
System.out.println("ID\tName\tAge\tCourse");
System.out.println("=====================\t=====\t=====");
// processing returned data and printing into console
while(resultSet.next()) {
    System.out.println(resultSet.getString(1) + "\t" +
                        resultSet.getString(2) + "\t" +
                        resultSet.getInt(3) + "\t" +
                        resultSet.getString(4));
}
// end of while loop
}
// end of try
catch(SQLException sqlex){
    sqlex.printStackTrace();
}
// end of catch
finally {
    // Step 3: Closing database connection
    try {
        if(null != connection) {
            // cleanup resources, once after processing
            resultSet.close();
            statement.close();
            // and then finally close connection
            connection.close();
        }
        //end of if
    }
    //end of try
    catch (SQLException sqlex) {
        sqlex.printStackTrace();
    }
    //end of catch
}
//end of finally
}
//end of main
}
//end of DeleteInDatabaseExample.java
```

PROGRAM OUTPUT

```

Loading Success!
Records before delete query.
ID          Name          Age      Course
==          =====      ==      =====
2019-MN-1   Lopez, Gerald   20      BSCS
2019-MN-2   Reyes, Luigi    18      BSIT
2019-MN-3   Legaspi, Sean  19      BSIS
2019-MN-4   Abandia, Bea  21      BSIT
2019-MN-5   Vergara, Kate  20      BSCE
2019-MN-6   Atienza, Faye  18      BSME

```

Enter Student ID to be deleted: 2019-MN-3

```
Update Successful!
Records after update query.
ID          Name          Age    Course
==          =====    ==    =====
2019-MN-1   Lopez, Gerald    20     BSCS
2019-MN-2   Reyes, Luigi     18     BSIT
2019-MN-4   Abandia, Bea    21     BSIT
2019-MN-5   Vergara, Kate    20     BSCE
2019-MN-6   Atienza, Fave    18     BSME
```



EXERCISE 1

The registrar group is to create a Enrolment system. A student database is needed. Explain the importance of the student database.

EXERCISE 2

The registrar group is to create Enrolment system. A student database is needed. Enumerate the fields of attributes to be included in the database.

◀ Preliminary Activity for Week 16

Jump to...



Analysis, Application, and Exploration for Week 16 ▶



Navigation

Home

 Dashboard

Site pages

My courses

121 - CC106

Participants

 Grades

General

MODULE 1: WHAT IS APPLICATION DEVELOPMENT?

MODULE 2: WHAT ARE THE TECHNICAL SKILLS REQUIRED I...

MODULE 3: WHAT ARE THE PROGRAMMING LANGUAGES USED ...

MODULE 4: WHAT IS JAVA PROGRAMMING LANGUAGE AS APP...

MODULE 5: HOW TO WRITE JAVA PROGRAMMING LANGUAGE A...



MODULE 6: PRELIMINARY EXAMINATION

MODULE 7: HOW TO WRITE JAVA PROGRAM USING INTEGRAT...

MODULE 8: WHAT ARE THE BUILDING BLOCKS OF OBJECT-O...

MODULE 9: WHAT ARE THE BASIC CONCEPTS OF INHERITAN...

MODULE 10: WHAT ARE THE BASIC CONCEPTS OF ENCAPSUL...

MODULE 11: WHAT ARE THE BASIC CONCEPTS OF POLUMORP...

Week 12: Midterm Examination

MODULE 13: WHAT ARE THE BASIC CONCEPTS OF ABSTRACT...


MODULE 14: HOW TO WRITE JAVA PROGRAM USING ABSTRAC...

MODULE 15: WHAT IS JAVA DATABASE CONNECTIVITY (JDB...


MODULE 16: WHAT ARE THE STEPS OF MANIPULATING DATA...


 Preliminary Activity for Week 16

 **Lesson Proper for Week 16**

 Analysis, Application, and Exploration for Week 16

 Generalization for Week 16

 Evaluation for Week 16

 Assignment for Week 16

MODULE 17: EMERGING TECHNOLOGIES

121 - BPM101 / DM103

121 - OAELEC2

121 - ITE3

121 - MUL101

121 - ITSP2B

121 - WEB101 / CCS3218

Courses

Fair Warning

NOTICE: Please be reminded that it has come to the attention of the Publishing Team of eLearning Commons that learning materials published and intended for ***free use only by students and faculty members within the eLearning Commons network were UNLAWFULLY uploaded in other sites without due and proper permission.***

PROSECUTION: Under Philippine law (Republic Act No. 8293), copyright infringement is punishable by the following: Imprisonment of between 1 to 3 years and a fine of between 50,000 to 150,000 pesos for the first offense. Imprisonment of 3 years and 1 day to six years plus a fine of between 150,000 to 500,000 pesos for the second offense.

COURSE OF ACTION: Whoever has maliciously uploaded these concerned materials are hereby given an ultimatum to take it down within 24-hours. Beyond the 24-hour grace period, our Legal Department shall initiate the proceedings in coordination with the National Bureau of Investigation for IP Address tracking, account owner identification, and filing of cases for prosecution.



2nd Semester Enrollment



visit www.bcp.edu.ph

Enrollment registration is now Ongoing

For 2nd Semester SY 2021 - 2022





We are accepting new students, returnees and transferees.

"Be trained to be the best,
Be linked to success"

 bcp-inquire@bcp.edu.ph  (8)442-8601 | (8)518-8050

The banner features a blue-tinted image of a multi-story building with a 'BCP' sign on the roof. A large white diagonal banner contains the enrollment information. The bottom right corner includes the college's motto and a circular seal.

Activities

-  Assignments
-  Forums
-  Quizzes
-  Resources

Bestlink College of the Philippines
College Department

Powered by [eLearning Commons](#)

