The ORDER BY clause is used to arrange the contents of the result set in the order it uses the keywords, ASC representing ascending order and DESC representing descending order. If you do not mention either of these, the contents will be arranged in ascending order by default.

Syntax

Following is the syntax of the ORDER BY clause −

SELECT \* FROM table\_name ORDER BY column\_name ASC|DESC.

Example

Suppose we have a table named Employees in the database with the following records −

ID |NAME |SALARY |LOCATION

------------------------------------------------------------------------------

1 |Amit |30000 |Vijayawada

2 |Kalyan |40000 |Vishakhapatnam

3 |Renuka |50000 |Delhi

4 |Archana |15000 |Vijayawada

5 |Trupthi |45000 |Kochin

6 |Suchatra |33000 |Vijayawada

7 |Rahul |39000 |Lucknow

The following query arranges the contents of the table in ascending order based on the name of the Employee.

ij> SELECT \* FROM Employees ORDER BY Name;

This will generate the following output −

ID |NAME |SALARY |LOCATION

---------------------------------------------------------------

1 |Amit |30000 |Hyderabad

4 |Archana |15000 |Mumbai

2 |Kalyan |40000 |Vishakhapatnam

7 |Rahul |39000 |Lucknow

3 |Renuka |50000 |Delhi

6 |Suchatra |33000 |Pune

5 |Trupthi |45000 |Kochin

7 rows selected

Similarly, following query arranges the contents of the table in descending order based on the salary of the Employee −

ij> SELECT \* FROM Employees ORDER BY Salary DESC;

This will generate the following output −

ID |NAME |SALARY |LOCATION

---------------------------------------------------------------

3 |Renuka |50000 |Delhi

5 |Trupthi |45000 |Kochin

2 |Kalyan |40000 |Vishakhapatnam

7 |Rahul |39000 |Lucknow

6 |Suchatra |33000 |Pune

1 |Amit |30000 |Hyderabad

4 |Archana |15000 |Mumbai

7 rows selected

Sorting Data using JDBC program

This section teaches you how to sort the contents of a table in Derby using JDBC. You can arrange records in order using the ORDER BY clause and the keywords ASC (denoting ascending order), and DSC (denoting descending order).

If you want to request the Derby network server using network client, make sure that the server is up and running. The class name for the Network client driver is org.apache.derby.jdbc.ClientDriver and the URL is jdbc:derby://localhost:1527/**DATABASE\_NAME;**create=true;user=**USER\_NAME;**passw ord=**PASSWORD**".

Follow the steps given below to sort the records of a table in Apache Derby −

Step 1: Register the driver

To communicate with the database, first of all, you need to register the driver. The **forName()** method of the class **Class** accepts a String value representing a class name loads it in to the memory, which automatically registers it. Register the driver using this method.

Step 2: Get the connection

In general, the first step we do to communicate to the database is to connect with it. The **Connection** class represents the physical connection with a database server. You can create a connection object by invoking the **getConnection()** method of the **DriverManager** class. Create a connection using this method.

Step 3: Create a statement object

You need to create a **Statement** or **PreparedStatement** or, **CallableStatement** objects to send SQL statements to the database. You can create these using the methods **createStatement(), prepareStatement() and, prepareCall()** respectively. Create either of these objects using the appropriate method.

Step 4: Execute the query

After creating a statement, you need to execute it. The **Statement** class provides various methods to execute a query like the **execute()** method to execute a statement that returns more than one result set. The **executeUpdate()** method executes queries like INSERT, UPDATE, DELETE. The **executeQuery()** method returns data. Use either of these methods and execute the statement created previously.

Example

Following JDBC example demonstrates how to sort the records of a table in Apache Derby using JDBC program. Here, we are connecting to a database named sampleDB (will create if it does not exist) using the embedded driver.

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

public class SortData {

public static void main(String args[]) throws SQLException, ClassNotFoundException {

//Registering the driver

Class.forName("org.apache.derby.jdbc.EmbeddedDriver");

//Getting the Connection object

String URL = "jdbc:derby:SampleDB;create=true";

Connection conn = DriverManager.getConnection(URL);

//Creating the Statement object

Statement stmt = conn.createStatement();

//Creating a table and populating it

String query = "CREATE TABLE Employees("

+ "Id INT NOT NULL GENERATED ALWAYS AS IDENTITY, "

+ "Name VARCHAR(255), Salary INT NOT NULL, "

+ "Location VARCHAR(255), "

+ "PRIMARY KEY (Id))";

String query = "INSERT INTO Employees("

+ "Name, Salary, Location) VALUES "

+ "('Amit', 30000, 'Hyderabad'), "

+ "('Kalyan', 40000, 'Vishakhapatnam'), "

+ "('Renuka', 50000, 'Delhi'), "

+ "('Archana', 15000, 'Mumbai'), "

+ "('Trupthi', 45000, 'Kochin'), "

+ "('Suchatra', 33000, 'Pune'), "

+ "('Rahul', 39000, 'Lucknow'), "

+ "('Trupti', 45000, 'Kochin')";

//Executing the query

String query = "SELECT Location, SUM(Salary) " + "from Employees GROUP BY Location";

ResultSet rs = stmt.executeQuery(query);

while(rs.next()) {

System.out.println("Salary: "+rs.getString(1));

System.out.println("Location: "+rs.getString(2));

System.out.println(" ");

}

}

}

Output

On executing the above program, you will get the following output −

Salary: Chennai

Location: 43000

Salary: Delhi

Location: 50000

Salary: Hyderabad

Location: 30000

Salary: Kochin

Location: 45000

Salary: Lucknow

Location: 39000

Salary: Mumbai

Location: 15000

Salary: Pune

Location: 33000