

DAY 2

In the previous class, we understood how to establish the connection between our java program and the database.

Now let us see how we can share data between the java program and the database.

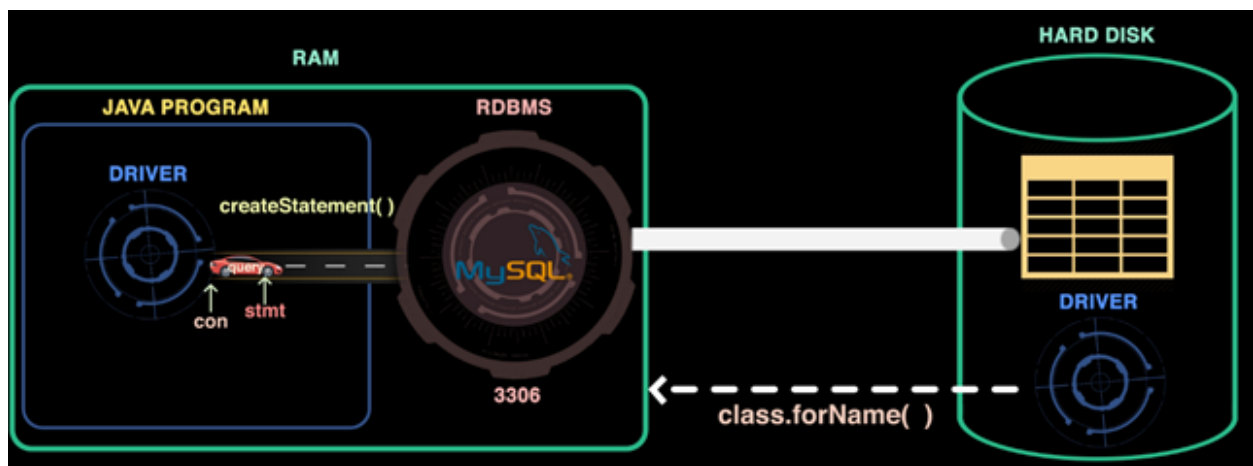
The connection which we have created between our java program and the database, assume to be a **“Road”**, through which we can pass **data(transport)**.

Assume a scenario, where you want to transport some goods from point A to point B, for that we usually use a goods vehicle.

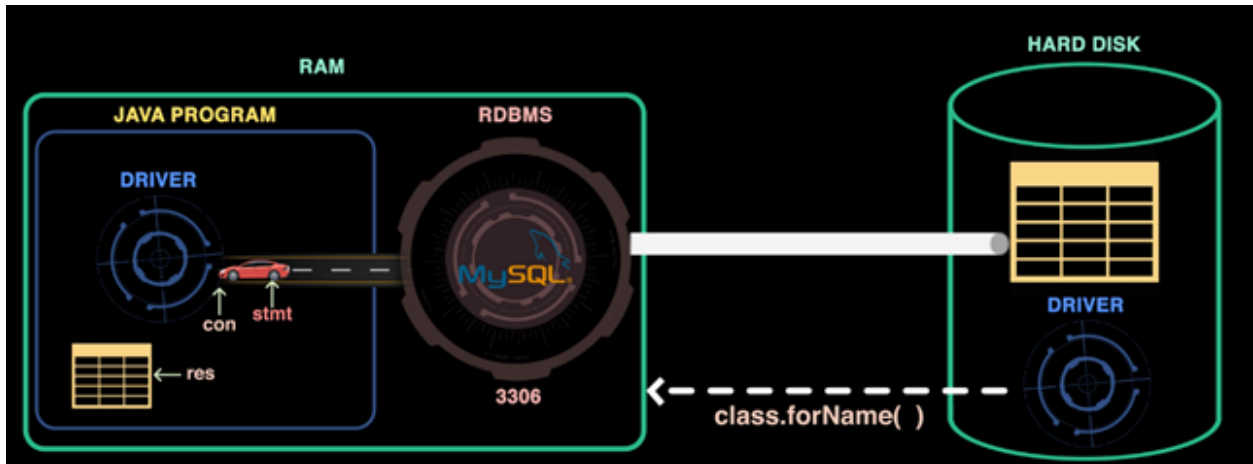
Similar to this analogy, if we want to pass a **“query”** from our java program to the database, we use something known as a **“statement”** in JDBC.

So we will give a reference to this statement as 'stmt', and we have to use the method **'createStatement()'** to create the statement stmt.

Now we will pass a **“query”** to the statement and send it to the database. The database will execute the query and return the output to the statement.



And this result of execution of the query is called **'ResultSet'** in JDBC. To access this ResultSet, we will give it a reference as **'res'**. Remember *res is of type ResultSet*.



Now let us see how we can integrate this into our code.

```
package jdbc;

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

public class Demo {

    public static void main(String[] args) {

        String url = "jdbc:mysql://localhost:3306/employee";
        String un = "root";
        String pwd = "root";

        try
        {
            Class.forName("com.mysql.cj.jdbc.Driver");
            System.out.println("Driver successfully loaded");

            Connection con = DriverManager.getConnection(url, un, pwd);
            System.out.println("Connection established");

            Statement stmt = con.createStatement();

            String query = "select * from emp";

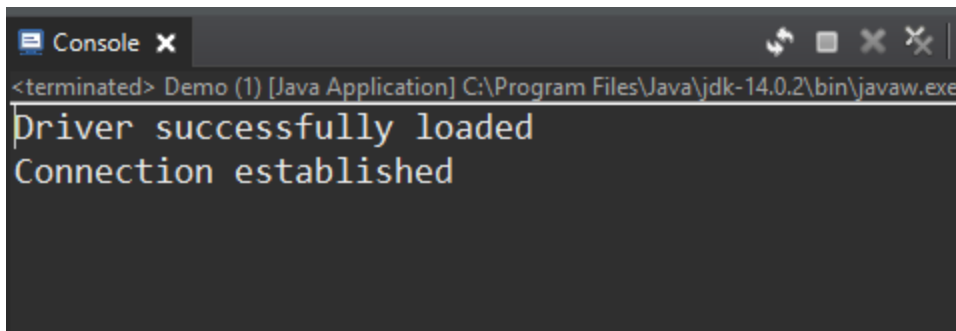
            ResultSet res = stmt.executeQuery(query);
            System.out.println("Query executed");
        }
    }
}
```

```

    }
    catch (ClassNotFoundException e)
    {
        System.out.println("Driver not loaded");
    }
    catch (SQLException e)
    {
        e.printStackTrace();
    }
}
}

```

And if we execute the code, we will the following output-



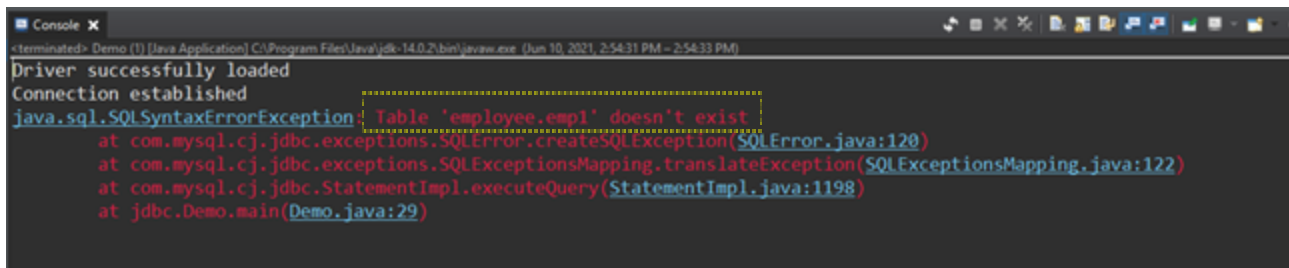
```

Console x
<terminated> Demo (1) [Java Application] C:\Program Files\Java\jdk-14.0.2\bin\javaw.exe
Driver successfully loaded
Connection established

```

So, it means that we were successfully able to establish a connection between the java code and the database and also retrieve the data from the database and save it in the ResultSet 'res'.

But if we change the table name to **'emp1'**, i.e., a table that does not exists, we will get the following error-



```

Console x
<terminated> Demo (1) [Java Application] C:\Program Files\Java\jdk-14.0.2\bin\javaw.exe (Jun 10, 2021, 2:54:31 PM)
Driver successfully loaded
Connection established
java.sql.SQLException: Table 'employee.emp1' doesn't exist
    at com.mysql.cj.jdbc.exceptions.SQLError.createSQLException(SQLError.java:120)
    at com.mysql.cj.jdbc.exceptions.SQLExceptionsMapping.translateException(SQLExceptionsMapping.java:122)
    at com.mysql.cj.jdbc.StatementImpl.executeQuery(StatementImpl.java:1198)
    at jdbc.Demo.main(Demo.java:29)

```

As we can see, Driver was **successfully loaded** and the **connection was also established**, but since the table **'emp1'** does not exist in the **'employee'** database, we get **"SQLSyntaxErrorException"**.

Now that we have got our ResultSet '**res**', let us now see how we can retrieve data from the ResultSet.

The reference '**res**' is like a cursor, which we can move and access the data from the table. This cursor will always be pointing over the 1st row, so if we want to access the data, we have to move the cursor.



The diagram illustrates a cursor named 'res' pointing to the first row of a table. The table has four columns: 'id', 'name', 'designation', and 'salary'. The first row contains the values '1', 'Rohit', 'CEO', and '30000'. The second row contains '2', 'Somanna', 'CTO', and '25000'. The third row contains '3', 'Rakshith', 'COO', and '20000'. The cursor 'res' is shown as an arrow pointing to the first row.

id	name	designation	salary
1	Rohit	CEO	30000
2	Somanna	CTO	25000
3	Rakshith	COO	20000

To make this possible we have a method known as 'next()', so if we want to move the cursor to the 1st row we have to call '**res.next()**', since res is the reference we have given to the ResultSet.

Now that the cursor is pointing to the 1st row, let us see how we can access the elements present in the row.

To access the elements, present in a row in two ways –

- By using the column number
- By using the column name

By using the column number:

If we want to access the element with the help of column number, then the syntax would be –

```
res.getDataType(column number)
```

Ex: **res.getInt(1)**

```
try
{
    Class.forName("com.mysql.cj.jdbc.Driver");
    System.out.println("Driver successfully loaded");

    Connection con = DriverManager.getConnection(url, un, pwd);
    System.out.println("Connection established");

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

```
String query = "select * from emp";

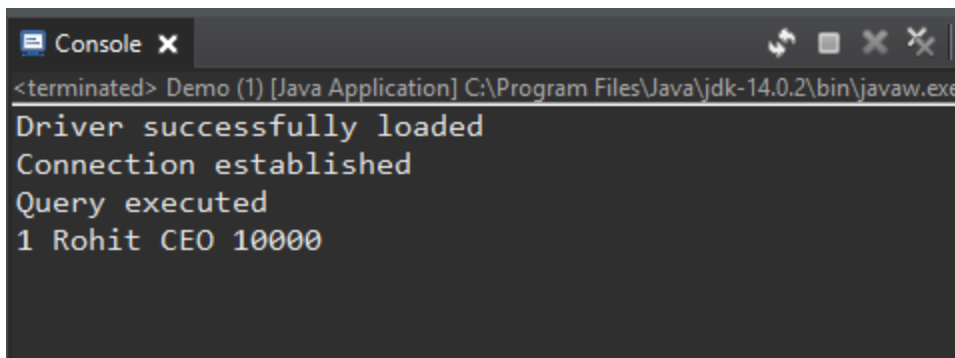
ResultSet res = stmt.executeQuery(query);
System.out.println("Query executed");

res.next();

System.out.println(res.getInt(1) + " " + res.getString(2) + " "
                  + res.getString(3) + " " + res.getInt(4));

}
catch (ClassNotFoundException e)
{
    System.out.println("Driver not loaded");
}
catch (SQLException e)
{
    e.printStackTrace();
}
```

Output:



```
Console X
<terminated> Demo (1) [Java Application] C:\Program Files\Java\jdk-14.0.2\bin\javaw.exe
Driver successfully loaded
Connection established
Query executed
1 Rohit CEO 10000
```

As we can clearly see, with the help of column number we can access the elements of the database.

By using the column name:

If we want to access the element with the help of column name, then the syntax would be –

```
res.getDatatype(column name)
```

Ex: `res.getInt("id")`

```
try
{
    Class.forName("com.mysql.cj.jdbc.Driver");
```

```

System.out.println("Driver successfully loaded");

Connection con = DriverManager.getConnection(url, un, pwd);
System.out.println("Connection established");

Statement stmt = con.createStatement();

String query = "select * from emp";

ResultSet res = stmt.executeQuery(query);
System.out.println("Query executed");

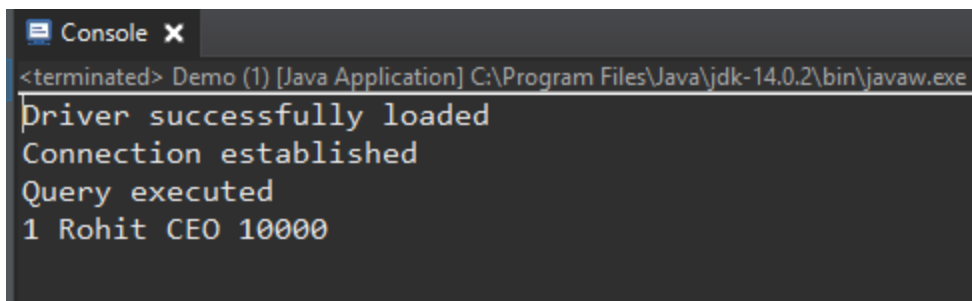
res.next();

System.out.println(res.getInt("id") + " " + res.getString("name") + " "
                  + res.getString("dsig") + " " + res.getInt("salary"));

}
catch (ClassNotFoundException e)
{
    System.out.println("Driver not loaded");
}
catch (SQLException e)
{
    e.printStackTrace();
}

```

Output:



```

Console X
<terminated> Demo (1) [Java Application] C:\Program Files\Java\jdk-14.0.2\bin\javaw.exe
Driver successfully loaded
Connection established
Query executed
1 Rohit CEO 10000

```

And we can also access the elements of the database with the help of column names.

So, if we want to access the next row, then we have to repeat the same code again. But if we had to access some 100s of rows, then repeating the lines of code would be a waste of time.

There must be an easier way to achieve this right?

Yes, we can make use of java loops to access the elements.

In this case, we will use “while loop”, as we want to continue till the condition becomes false i.e., we want to access till there are no more rows. In other words, till the condition (**res.next() == true**) is true, we will fetch the elements and as soon as the condition becomes false, it means that there are no more rows and the loop will terminate.

So, after modifying the code as shown below -

```
Connection con = DriverManager.getConnection(url, un, pwd);
System.out.println("Connection established");

Statement stmt = con.createStatement();

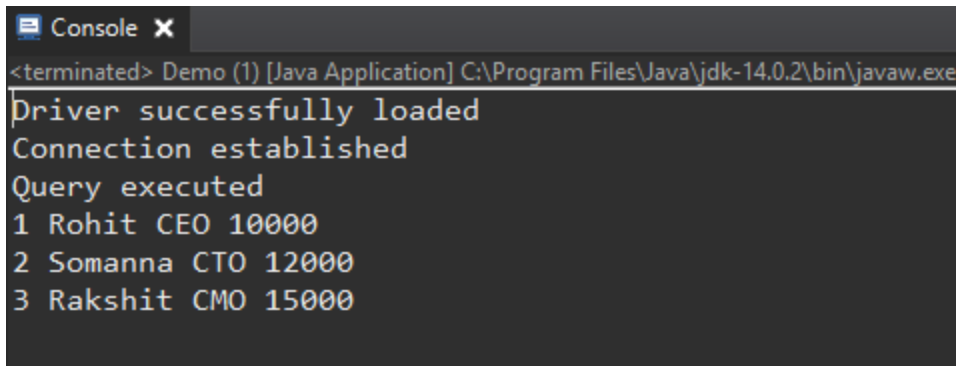
String query = "select * from emp";

ResultSet res = stmt.executeQuery(query);
System.out.println("Query executed");

res.next();

while(res.next() == true)
{
    System.out.println(res.getInt("id") + " " + res.getString("name") + " "
        + res.getString("dsig") + " " + res.getInt("salary"));
}
```

The output will be -

A screenshot of a Java console window titled "Console" with a close button. The window shows the output of a Java application. The first line is "<terminated> Demo (1) [Java Application] C:\Program Files\Java\jdk-14.0.2\bin\javaw.exe". The subsequent lines are: "Driver successfully loaded", "Connection established", "Query executed", and then three lines of data: "1 Rohit CEO 10000", "2 Somanna CTO 12000", and "3 Rakshit CMO 15000".

```
<terminated> Demo (1) [Java Application] C:\Program Files\Java\jdk-14.0.2\bin\javaw.exe
Driver successfully loaded
Connection established
Query executed
1 Rohit CEO 10000
2 Somanna CTO 12000
3 Rakshit CMO 15000
```

The cursor will always move in the forward direction, i.e., it is not a “**scrollable cursor**” but what if we wanted to move it in the reverse direction or we wanted to access the 3rd row directly from the 1st row. Can we do that? Let us try and see-

```
res.first();
System.out.println(res.getInt("id") + " " + res.getString("name") + " "
    + res.getString("dsig") + " " + res.getInt("salary"));
```

And if we try to execute the above code, we will get the an SQLException-

```
Console x
<terminated> Demo (1) [Java Application] C:\Program Files\Java\jdk-14.0.2\bin\javaw.exe (Jun 12, 2021, 9:46:10 AM - 9:46:13 AM)
Driver successfully loaded
Connection established
Query executed
1 Rohit CEO 10000
2 Somanna CTO 12000
3 Rakshit CMO 15000
java.sql.SQLException: Operation not allowed for a result set of type ResultSet.TYPE_FORWARD_ONLY.
    at com.mysql.cj.jdbc.exceptions.SQLExceptionsMapping.translateException(SQLExceptionsMapping.java:122)
    at com.mysql.cj.jdbc.result.ResultSetImpl.first(ResultSetImpl.java:609)
    at jdbc.Demo.main(Demo.java:38)
```

This proves that this is a non-scrollable result set and once the cursor is moved forward, we cannot move it in the reverse direction nor can it jump from one position to another position.

So, what is the solution for this? How can we make our cursor scrollable?

We can resolve this by creating a statement that is scrollable and we can achieve this like shown below-

```
Statement stmt = con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,0);
```

Now our result set is scroll-sensitive and if we now re-execute the code, we can see that we will get the output without any exception i.e., we can scroll through our result set.

```
Console x
<terminated> Demo (1) [Java Application] C:\Program Files\Java\jdk-14.0.2\bin\javaw.exe
Driver successfully loaded
Connection established
Query executed
1 Rohit CEO 10000
2 Somanna CTO 12000
3 Rakshit CMO 15000
1 Rohit CEO 10000
```

If we want to access any row, then we can use the **absolute()**, which accepts one integer parameter, which is the row number.

In real-time, the database could be in one computer and the java code may be in a different computer and we may or may not know the database content, then how do we get to know the data inside the table?

If you don't know the information of the table then what you will do i.e if you don't know the **metadata** (**Metadata in simple words described as data about data.**).

To get the metadata about the table

Once you get the ResultSet along with that metadata will be available

```
ResultSetMetaData metaData = res.getMetaData();  
System.out.println(metaData.getColumnCount());
```

To get Column type

```
for(int i=1; i < metaData.getColumnCount(); i++) {  
    System.out.println(metaData.getColumnName(i)+" "+  
        metaData.getColumnTypeName(i));  
}
```