Java ResultSetMetaData Interface

The metadata means data about data i.e. we can get further information from the data.

If you have to get metadata of a table like total number of column, column name, column type etc. , ResultSetMetaData interface is useful because it provides methods to get metadata from the ResultSet object.

Commonly used methods of ResultSetMetaData interface

Method	Description
public int getColumnCount()throws SQLException	it returns the total number of columns in the ResultSet object.
public String getColumnName(int index)throws SQLException	it returns the column name of the specified column index.
public String getColumnTypeName(int index)throws SQLException	it returns the column type name for the specified index.
public String getTableName(int index)throws SQLException	it returns the table name for the specified column index.

How to get the object of ResultSetMetaData:

The getMetaData() method of ResultSet interface returns the object of ResultSetMetaData. Syntax:

```
public ResultSetMetaData getMetaData()throws SQLException
```

Example of ResultSetMetaData interface :

```
import java.sql.*;
class Rsmd{
public static void main(String args[]){
try{
Class.forName("oracle.jdbc.driver.OracleDriver");
Connection con=DriverManager.getConnection(
"jdbc:oracle:thin:@localhost:1521:xe","system","oracle");

PreparedStatement ps=con.prepareStatement("select * from emp");
ResultSet rs=ps.executeQuery();
ResultSetMetaData rsmd=rs.getMetaData();

System.out.println("Total columns: "+rsmd.getColumnCount());
System.out.println("Column Name of 1st column: "+rsmd.getColumnName(1));
System.out.println("Column Type Name of 1st column: "+rsmd.getColumnTypeName(1));

con.close();
}catch(Exception e){ System.out.println(e);}
}
}
```

```
Output:Total columns: 2
       Column Name of 1st column: ID
       Column Type Name of 1st column: NUMBER
```

Java DatabaseMetaData interface

DatabaseMetaData interface provides methods to get meta data of a database such as database product name, database product version, driver name, name of total number of tables, name of total number of views etc.

Commonly used methods of DatabaseMetaData interface

- **public String getDriverName()throws SQLException:** it returns the name of the JDBC driver.
- **public String getDriverVersion()throws SQLException:** it returns the version number of the JDBC driver.
- **public String getUserName()throws SQLException:** it returns the username of the database.
- **public String getDatabaseProductName()throws SQLException:** it returns the product name of the database.
- **public String getDatabaseProductVersion()throws SQLException:** it returns the product version of the database.
- **public ResultSet getTables(String catalog, String schemaPattern, String tableNamePattern, String[] types)throws SQLException:** it returns the description of the tables of the specified catalog. The table type can be TABLE, VIEW, ALIAS, SYSTEM TABLE, SYNONYM etc.

How to get the object of DatabaseMetaData:

The getMetaData() method of Connection interface returns the object of DatabaseMetaData.
Syntax:

```
public DatabaseMetaData getMetaData()throws SQLException
```

Simple Example of DatabaseMetaData interface :

```
import java.sql.*;
class Dbmd{
public static void main(String args[]){
try{
Class.forName("oracle.jdbc.driver.OracleDriver");

Connection con=DriverManager.getConnection(
"jdbc:oracle:thin:@localhost:1521:xe","system","oracle");
DatabaseMetaData dbmd=con.getMetaData();

System.out.println("Driver Name: "+dbmd.getDriverName());
System.out.println("Driver Version: "+dbmd.getDriverVersion());
System.out.println("UserName: "+dbmd.getUserName());
System.out.println("Database Product Name: "+dbmd.getDatabaseProductName());
System.out.println("Database Product Version: "+dbmd.getDatabaseProductVersion());

con.close();
}catch(Exception e){ System.out.println(e);}
}
}
```

```
Output:Driver Name: Oracle JDBC Driver
       Driver Version: 10.2.0.1.0XE
       Database Product Name: Oracle
       Database Product Version: Oracle Database 10g Express Edition
                                 Release 10.2.0.1.0 -Production
```

download this example

Example of DatabaseMetaData interface that prints total number of tables :

```
import java.sql.*;
class Dbmd2{
public static void main(String args[]){
try{
Class.forName("oracle.jdbc.driver.OracleDriver");

Connection con=DriverManager.getConnection(
"jdbc:oracle:thin:@localhost:1521:xe","system","oracle");

DatabaseMetaData dbmd=con.getMetaData();
String table[]={"TABLE"};
ResultSet rs=dbmd.getTables(null,null,null,table);

while(rs.next()){
System.out.println(rs.getString(3));
}

con.close();

}catch(Exception e){ System.out.println(e);}

}
}
```

download this example

Example of DatabaseMetaData interface that prints total number of views :

```
import java.sql.*;
class Dbmd3{
public static void main(String args[]){
try{
Class.forName("oracle.jdbc.driver.OracleDriver");

Connection con=DriverManager.getConnection(
"jdbc:oracle:thin:@localhost:1521:xe","system","oracle");

DatabaseMetaData dbmd=con.getMetaData();
String table[]={"VIEW"};
ResultSet rs=dbmd.getTables(null,null,null,table);

while(rs.next()){
System.out.println(rs.getString(3));
}

con.close();

}catch(Exception e){ System.out.println(e);}

}
}
```

download this example

Example to store image in Oracle database

You can store images in the database in java by the help of **PreparedStatement** interface.

The **setBinaryStream()** method of PreparedStatement is used to set Binary information into the parameterIndex.

Signature of setBinaryStream method

The syntax of setBinaryStream() method is given below:

```
1) public void setBinaryStream(int paramIndex,InputStream stream)
throws SQLException
2) public void setBinaryStream(int paramIndex,InputStream stream,long length)
throws SQLException
```

For storing image into the database, BLOB (Binary Large Object) datatype is used in the table. For example:

```
CREATE TABLE  "IMGTABLE"
   (    "NAME" VARCHAR2(4000),
    "PHOTO" BLOB
   )
/
```

Let's write the jdbc code to store the image in the database. Here we are using d:\\d.jpg for the location of image. You can change it according to the image location.

Java Example to store image in the database

```
import java.sql.*;
import java.io.*;
public class InsertImage {
public static void main(String[] args) {
```

```
try{
Class.forName("oracle.jdbc.driver.OracleDriver");
Connection con=DriverManager.getConnection(
"jdbc:oracle:thin:@localhost:1521:xe","system","oracle");

PreparedStatement ps=con.prepareStatement("insert into imgtable values(?,?)");
ps.setString(1,"sonoo");

FileInputStream fin=new FileInputStream("d:\\g.jpg");
ps.setBinaryStream(2,fin,fin.available());
int i=ps.executeUpdate();
System.out.println(i+" records affected");

con.close();
}catch (Exception e) {e.printStackTrace();}
}
}
```

If you see the table, record is stored in the database but image will not be shown. To do so, you need to retrieve the image from the database which we are covering in the next page.

Example to retrieve image from Oracle database

By the help of **PreparedStatement** we can retrieve and store the image in the database.

The **getBlob()** method of PreparedStatement is used to get Binary information, it returns the instance of Blob. After calling the **getBytes()** method on the blob object, we can get the array of binary information that can be written into the image file.

Signature of getBlob() method of PreparedStatement

```
public Blob getBlob()throws SQLException
```

Signature of getBytes() method of Blob interface

```
public  byte[] getBytes(long pos, int length)throws SQLException
```

We are assuming that image is stored in the imgtable.

```
CREATE TABLE  "IMGTABLE"
  (    "NAME" VARCHAR2(4000),
   "PHOTO" BLOB
  )
/
```

Now let's write the code to retrieve the image from the database and write it into the directory so that it can be displayed.

In AWT, it can be displayed by the Toolkit class. In servlet, jsp, or html it can be displayed by the img tag.

```
import java.sql.*;
import java.io.*;
public class RetrieveImage {
public static void main(String[] args) {
try{
Class.forName("oracle.jdbc.driver.OracleDriver");
Connection con=DriverManager.getConnection(
"jdbc:oracle:thin:@localhost:1521:xe","system","oracle");

PreparedStatement ps=con.prepareStatement("select * from imgtable");
ResultSet rs=ps.executeQuery();
if(rs.next()){//now on 1st row

Blob b=rs.getBlob(2);//2 means 2nd column data
byte barr[]=b.getBytes(1,(int)b.length());//1 means first image

FileOutputStream fout=new FileOutputStream("d:\\sonoo.jpg");
fout.write(barr);

fout.close();
}//end of if
System.out.println("ok");

con.close();
}catch (Exception e) {e.printStackTrace();  }
}
}
```

Now if you see the d drive, sonoo.jpg image is created.