2.8 EJECUCIÓN DE SENTENCIAS DE DESCRIPCIÓN DE DATOS.

Normalmente cuando desarrollamos una aplicación JDBC conocemos la estructura de las tablas y datos que estamos manejando, es decir, conocemos, las columnas que tienen y cómo están relacionadas entre sí. Es posible que no conozcamos la estructura de las tablas de una base de datos, en este caso la información de la base de datos se puede obtener a través de los *metaobjetos*, que no son más que objetos que proporcionan información sobre la base de datos.

La interfaz **DatabaseMetaData** proporciona información sobre la base de datos a través de múltiples métodos de los cuales es posible obtener gran cantidad de información.

Muchos de estos métodos devuelven un **ResultSet**, algunos de los que veremos en los siguientes ejemplos son:

|  |  |
| --- | --- |
| **Método** | **Descripción** |
| getTables() | Proporciona información sobre las tablas y vistas de la base de datos. |
| getColumns() | Devuelve información sobre las columnas de una tabla. |
| getPrimaryKeys() | Proporciona información sobre las columnas que forman la clave primaria de una tabla. |
| getExportedKeys() | Devuelve información sobre las claves ajenas que utilizan la clave primaria de una tabla. |
| getImportedKeys() | Devuelve información sobre las claves ajenas existentes en una tabla. |
| getProcedures() | Devuelve información sobre los procedimientos almacenados. |
| ***Más información sobre métodos de DatabaseMetaData:***  https://docs.oracle.com/javase/8/docs/api/java/sql/DatabaseMetaData.html | |

El siguiente ejemplo conecta con la base de datos MySQL de nombre *ejemplo* y muestra información sobre el producto de base de datos, el driver, la URL para acceder a la base de datos, el nombre de usuario y las tablas y vistas del esquema actual (o de todos los esquemas dependiendo del sistema gestor de base de datos), un esquema se corresponde generalmente con un usuario de la base de datos; el método ***getMetaData()*** de la interface **Connection** devuelve un objeto **DataBaseMetaData** que contiene información sobre la base de datos representada por el objeto **Connection**:

import java.sql.\*;

**public class EjemploDatabaseMetadata** {

public static void main(String[] args) {

try

{

Class.forName("com.mysql.jdbc.Driver"); //Cargar el driver

//Establecemos la conexion con la BD

Connection conexion = DriverManager.getConnection

("jdbc:mysql://localhost/ejemplo","ejemplo", "ejemplo");

**DatabaseMetaData dbmd = conexion.getMetaData();**

ResultSet resul = null;

String nombre = dbmd.getDatabaseProductName();

String driver = dbmd.getDriverName();

String url = dbmd.getURL();

String usuario = dbmd.getUserName() ;

System.out.println("INFORMACIÓN SOBRE LA BASE DE DATOS:");

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

System.out.printf("Nombre : %s %n", nombre );

System.out.printf("Driver : %s %n", driver );

System.out.printf("URL : %s %n", url );

System.out.printf("Usuario: %s %n", usuario );

**//Obtener información de las tablas y vistas que hay**

**resul = dbmd.getTables(null, "ejemplo", null, null);**

**//****getTables(catalogo, esquema, patronDeTabla, tipos[])**

while (resul.next()) {

String catalogo = resul.getString(1);//columna 1

String esquema = resul.getString(2); //columna 2

String tabla = resul.getString(3); //columna 3

String tipo = resul.getString(4); //columna 4

System.out.printf("%s - Catalogo: %s, Esquema: %s,

Nombre: %s %n", tipo, catalogo, esquema, tabla);

}

conexion.close(); //Cerrar conexion

}

catch (ClassNotFoundException cn) {cn.printStackTrace();}

catch (SQLException e) {e.printStackTrace();}

}//fin de main

}//fin de la clase

La ejecución del programa visualiza la siguiente información:

INFORMACIÓN SOBRE LA BASE DE DATOS:

===================================

Nombre : MySQL

Driver : MySQL-AB JDBC Driver

URL : jdbc:mysql://localhost/ejemplo

Usuario: ejemplo@localhost

TABLE - Catalogo: ejemplo, Esquema: null, Nombre: departamentos

TABLE - Catalogo: ejemplo, Esquema: null, Nombre: empleados

VIEW - Catalogo: ejemplo, Esquema: null, Nombre: vista

El método ***getTables()*** devuelve un objeto **ResultSet** que proporciona información sobre las tablas y vistas de la base de datos. Consulta la URL para ver todo sobre el método

<https://docs.oracle.com/javase/8/docs/api/java/sql/DatabaseMetaData.html#getTables-java.lang.String-java.lang.String-java.lang.String-java.lang.String:A->

Su sintaxis es:

public abstract **ResultSet** **getTables**(

String catalogo, String esquema,

String patronDeTabla, String tipos[]) throws SQLException

* Primer parámetro: catálogo de la base de datos. El método obtiene las tablas del catálogo indicado, al poner *null*, indicamos el catálogo actual.
* Segundo parámetro: esquema de la base de datos (nombre de usuario). Obtiene las tablas del esquema indicado, el valor *null* indica el esquema actual (o todos los esquemas, dependiendo del SGBD, como en Oracle).
* Tercer parámetro: es un patrón en el que se indica el nombre de las tablas que queremos que obtenga el método. Se puede utilizar el carácter guión bajo o porcentaje, por ejemplo *"de%"* obtendría todas las tablas cuyo nombre empieza por *"de"*.
* El cuarto parámetro es un array de *String*, en el que indicamos qué tipos de objetos queremos queremos obtener, por ejemplo: *TABLE* (para tablas), *VIEW* (para vistas); al poner ***null***, nos devolverá todos los tipos de objetos ya sean tablas o vistas. Los tipos válidos son: ***"TABLE", "VIEW", "SYSTEM TABLE", "GLOBAL TEMPORARY", "LOCAL TEMPORARY", "ALIAS"* y *"SYNONYM****"*. El siguiente ejemplo nos devolvería las tablas y los sinónimos:

String[] tipos = {"TABLE","SYNONYM"};

resul = dbmd.getTables(null, null, null, tipos);

**getTables en las distintas BD:**

**getTables(catalogo, esquema, patronDeTabla, tipos[])**

**.getTables(null, null, null, null );**

Poniendo null indicamos catálogo actual, esquema actual, todas las tablas, todos los tipos

En MYSQL:

ResultSet resul = dbmd.getTables("ejemplo", null, null, null ); // ejemplo es la BD

En ORACLE:

ResultSet resul = dbmd.getTables(null, "EJEMPLO", null, null ); // EJEMPLO es el usuario

En APACHE DERBY:

ResultSet resul = dbmd.getTables(null, "APP", null, null );

// APP es el usuario administrador por defecto de esta BD.

En SQLITE:

ResultSet resul = dbmd.getTables(null, null, null, null); // No se necesita ningún valor

En HSQLDB :

ResultSet resul = dbmd.getTables("PUBLIC", "PUBLIC", null, null );

//PUBLIC es el catálogo y esquema por defecto de esta BD.

Cada fila de **ResultSet** que devuelve ***getTables()*** tiene información sobre una tabla.

La descripción de cada columna tiene las siguientes columnas:

*TABLE\_CAT* (columna 1, el nombre del catálogo al que pertenece la tabla)*,*

*TABLE\_SCHEM,* (columna 2, el nombre del esquema al que pertenece la tabla),

*TABLE\_NAME* (columna 3, el nombre de la tabla o vista),

*TABLE\_TYPE* (columna 4, el tipo TABLE o VIEW),

*REMARKS* (columna 5, comentarios)*,*

*TYPE\_CAT, TYPE\_SCHEM, TYPE\_NAME, SELF\_REFERENCING\_COL\_NAME,* y *REF\_GENERATION*.

Para obtener estos resultados también podríamos haber puesto en el código anterior el nombre de la columna en lugar del número:

String catalogo = resul.getString("**TABLE\_CAT**"); //columna 1

String esquema = resul.getString("**TABLE\_SCHEM**"); //columna 2

String tabla = resul.getString("**TABLE\_NAME**"); //columna 3

String tipo = resul.getString("**TABLE\_TYPE**"); //columna 4

Actividad 2.8

Prueba el programa anterior para visualizar información de las bases de datos Oracle y SQLite con las que estás trabajando en este tema.

Otros métodos importantes del objeto **DatabaseMetaData** son:

* ***getColumns()*:**

<https://docs.oracle.com/javase/8/docs/api/java/sql/DatabaseMetaData.html#getColumns-java.lang.String-java.lang.String-java.lang.String-java.lang.String->

Devuelve un objeto **ResultSet** con información sobre las columnas de una tabla o tablas. La descripción de cada columna tiene las siguientes columnas: *TABLE\_CAT, TABLE\_SCHEM, TABLE\_NAME, COLUMN\_NAME, DATA\_TYPE, TYPE\_NAME, COLUMN\_SIZE, BUFFER\_LENGTH, DECIMAL\_DIGITS, NUM\_PREC\_RADIX, NULLABLE, REMARKS, COLUMN\_DEF, SQL\_DATA\_TYPE, SQL\_DATETIME\_SUB, CHAR\_OCTET\_LENGTH, ORDINAL\_POSITION, IS\_NULLABLE, SCOPE\_CATALOG, SCOPE\_SCHEMA, SCOPE\_TABLE, SOURCE\_DATA\_TYPE, IS\_AUTOINCREMENT* e *IS\_GENERATEDCOLUMN*. Su sintaxis es:

public abstract **ResultSet** **getColumns**(

String catalogo, String Esquema,

String patronNombreDeTabla, String patronNombreDeColumna)

throws SQLException

Para el patrón de nombre de la tabla y de la columna se puede utilizar el carácter guión bajo o porcentaje. Por ejemplo ***getColumns(null, “ejemplo”, “departamentos”, “d%”)***, obtiene todos los nombres de columna que empiezan por la letra d en la tabla *departamentos* y en el esquema de nombre *ejemplo*.

El valor *null* en los 4 parámetros indica que obtiene información de todas las columnas y tablas del esquema actual. El siguiente ejemplo muestra información sobre todas las columnas de la tabla *departamentos*:

System.out.println("COLUMNAS TABLA DEPARTAMENTOS:");

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

ResultSet columnas=null;

**columnas = dbmd.getColumns(null, "ejemplo", "departamentos", null);**

while (columnas.next()) {

String nombCol = columnas.getString("**COLUMN\_NAME**"); //getString(4)

String tipoCol = columnas.getString("**TYPE\_NAME**"); //getString(6)

String tamCol = columnas.getString("**COLUMN\_SIZE**"); //getString(7)

String nula = columnas.getString("**IS\_NULLABLE**"); //getString(18)

System.out.printf("Columna: %s, Tipo: %s, Tamaño: %s,

¿Puede ser Nula:? %s %n", nombCol, tipoCol, tamCol, nula);

}

Visualiza la siguiente información:

COLUMNAS TABLA DEPARTAMENTOS:

===================================

Columna: dept\_no, Tipo: TINYINT, Tamaño: 3, ¿Puede ser Nula:? NO

Columna: dnombre, Tipo: VARCHAR, Tamaño: 15, ¿Puede ser Nula:? YES

Columna: loc, Tipo: VARCHAR, Tamaño: 15, ¿Puede ser Nula:? YES

* ***getPrimaryKeys()***:

<https://docs.oracle.com/javase/8/docs/api/java/sql/DatabaseMetaData.html#getPrimaryKeys-java.lang.String-java.lang.String-java.lang.String->

devuelve la lista de columnas que forman la clave primaria de la tabla especificada. La descripción de cada columna de la clave primaria tiene las siguientes columnas: *TABLE\_CAT, TABLE\_SCHEM, TABLE\_NAME, COLUMN\_NAM* y *KEY\_SEQ*. La sintasis es la siguiente:

public abstract **ResultSet getPrimaryKeys**(

String catalogo, String esquema, String tabla)

throws SQLException

El siguiente ejemplo muestra la clave primaria de la tabla *departamentos* (ejemplo en MySQL):

**ResultSet pk = dbmd.getPrimaryKeys(null, "ejemplo", "departamentos");**

String pkDep="", separador="";

while (pk.next()) {

pkDep = pkDep + separador +

pk.getString("**COLUMN\_NAME**");//getString(4)

separador="+";

}

System.out.println("Clave Primaria: " + pkDep);

* ***getExportedKeys()*:**

<https://docs.oracle.com/javase/8/docs/api/java/sql/DatabaseMetaData.html#getExportedKeys-java.lang.String-java.lang.String-java.lang.String->

devuelve la lista de todas las claves ajenas que utilizan la clave primaria de la tabla especificada. La descripción de cada columna de clave ajena tiene las siguientes columnas: *PKTABLE\_CAT, PKTABLE\_SCHEM, PKTABLE\_NAME, PKCOLUMN\_NAME, FKTABLE\_CAT, FKTABLE\_SCHEM, FKTABLE\_NAME, FKCOLUMN\_NAME, KEY\_SEQ, UPDATE\_RULE, DELETE\_RULE, FK\_NAME, PK\_NAME y DEFERRABILITY*. La sintaxis es:

public abstract **ResultSet getExportedKeys**

(String catalogo, String esquema, String tabla) throws SQLException

El siguiente ejemplo muestra las tablas y sus claves ajenas que referencian a la tabla *departamentos*, en este caso sólo la tabla *empleados*:

**ResultSet fk = dbmd.getExportedKeys(null, "ejemplo", "departamentos");**

while (fk.next()) {

String fk\_name = fk.getString("**FKCOLUMN\_NAME**");

String pk\_name = fk.getString("**PKCOLUMN\_NAME**");

String pk\_tablename = fk.getString("**PKTABLE\_NAME**");

String fk\_tablename = fk.getString("**FKTABLE\_NAME**");

System.out.printf("Tabla PK: %s, Clave Primaria: %s %n",

pk\_tablename, pk\_name);

System.out.printf("Tabla FK: %s, Clave Ajena: %s %n",

fk\_tablename, fk\_name);

}

Visualiza la siguiente información:

Tabla PK: departamentos, Clave Primaria: dept\_no

Tabla FK: empleados, Clave Ajena: dept\_no

El método no devuelve nada si queremos ver las claves ajenas que referencian a la tabla *empleados*, ***dbmd.getExportedKeys(null, "ejemplo", "empleados")***, ya que la tabla *empleados* no es referenciada por ninguna clave ajena.

A la hora de crear una tabla es recomendable definir las restricciones de clave ajena asignándolas un nombre, usando la cláusula *CONSTRAINT nombre FOREIGN KEY (col1, col2,…) REFERENCES tabla(col1,col2,…)*. De esta manera el método ***getExportedKeys()*** nos devolverá la información deseada.

* ***getImportedKeys():***

<https://docs.oracle.com/javase/8/docs/api/java/sql/DatabaseMetaData.html#getImportedKeys-java.lang.String-java.lang.String-java.lang.String->

devuelve la lista de claves ajenas existentes en la tabla indicada. Se utiliza igual que el método anterior, en este caso ***dbmd.getImportedKeys(null, "ejemplo", "empleados")*** devuelve la salida anterior, en cambio ***dbmd.getImportedKeys(null, "ejemplo", "departamentos")*** no devuelve nada ya que no tiene claves ajenas. La sintaxis es:

public abstract **ResultSet getImportedKeys**

(String catalogo, String esquema, String tabla)

throws SQLException

* ***getProcedures()***:

<https://docs.oracle.com/javase/8/docs/api/java/sql/DatabaseMetaData.html#getProcedures-java.lang.String-java.lang.String-java.lang.String->

devuelve la lista de procedimientos almacenados. Cada descripción de procedimiento tiene las siguientes columnas: *PROCEDURE\_CAT* (columna 1), *PROCEDURE\_SCHEM* (columna 2), *PROCEDURE\_NAME* (columna 3), *REMARKS* (columna 7), *PROCEDURE\_TYPE* (columna 8) y *SPECIFIC\_NAME* (columna 9). La sintaxis es:

public abstract **ResultSet getProcedures**

(String catalogo, esquema, String procedure) throws SQLException

Para probar el método ***getProcedures()*** creamos algunos procedimientos y funciones. Por ejemplo, creamos la función de nombre *SUMAR* que recibe dos números y devuelve la suma:

**Ejemplo de función en Oracle:**

CREATE OR REPLACE FUNCTION SUMAR (N1 NUMBER, N2 NUMBER)

RETURN NUMBER AS

BEGIN

RETURN N1 + N2;

END SUMAR;

/

Para probarla escribimos: *SELECT SUMAR(2,22) FROM DUAL;*

**Ejemplo de función en MySQL:**

DELIMITER //

CREATE FUNCTION SUMAR (N1 INT, N2 INT) RETURNS INT

BEGIN

RETURN N1 + N2;

END;

//

Para probarla escribimos: *SELECT SUMAR(2,22)*

Ejemplo de creación de un procedimiento de nombre *SUBIDA* que sube 100 euros el salario de los empleados del departamento 30:

**Ejemplo en Oracle:**

CREATE OR REPLACE PROCEDURE SUBIDA AS

BEGIN

UPDATE EMPLEADOS SET SALARIO = SALARIO +100 WHERE DEPT\_NO=30;

COMMIT;

END SUBIDA;

Para probarla escribimos: exec SUBIDA();

**Ejemplo en MySQL:**

DELIMITER //

CREATE PROCEDURE SUBIDA()

BEGIN

UPDATE EMPLEADOS SET SALARIO = SALARIO + 100 WHERE DEPT\_NO=30;

COMMIT;

END;

//

Para probarla escribimos: CALL SUBIDA();

El siguiente ejemplo muestra los procedimientos y funciones que tiene el esquema de nombre *ejemplo*:

**ResultSet proc = dbmd.getProcedures(null, "ejemplo", null);**

while (proc.next()) {

String proc\_name = proc.getString("**PROCEDURE\_NAME**");

String proc\_type = proc.getString("**PROCEDURE\_TYPE**");

System.out.printf("Nombre Procedimiento: %s - Tipo: %s %n",

proc\_name, proc\_type);

}

2.8.1 ResultSetMetaData.

Se pueden obtener metadatos (datos sobre los datos) a partir de un objeto **ResultSet** mediante la interfaz **ResultSetMetaData**; es decir podemos obtener más información sobre los tipos y propiedades de las columnas de los objetos **ResultSet**, como por ejemplo el número de columnas devueltas, el tipo, el nombre, etc. El siguiente trozo de código muestra el uso de la interfaz para conocer más información acerca de las columnas devueltas por esta consulta *SELECT \* FROM departamentos*; en este caso al usar \* en la *SELECT* desconocemos el nombre de las columnas devueltas.

Usaremos el método ***getMetadata()*** del objeto **ResultSet** que devuelve una referencia a un objeto **ResultSetMetaData** con el que se obtendrá la información acerca de las columnas devueltas:

import java.sql.\*;

**public class EjemploResultsetmetadata** {

public static void resulsetmetadata(Connection conexion, String consulta)

{

//public static void main(String[] args) {

try {

Class.forName("com.mysql.jdbc.Driver"); // Cargar el driver

Connection conexion = DriverManager.getConnection(

"jdbc:mysql://localhost/ejemplo", "ejemplo", "ejemplo");

Statement sentencia = conexion.createStatement();

**ResultSet rs = sentencia**

**.executeQuery("SELECT \* FROM departamentos");**

**ResultSetMetaData rsmd = rs.getMetaData();**

**int nColumnas = rsmd.getColumnCount();**

String nula;

System.out.printf("Número de columnas recuperadas: %d%n",

nColumnas);

for (int i = 1; i <= nColumnas; i++) {

System.out.printf("Columna %d: %n ", i);

System.out.printf(" Nombre: %s %n Tipo: %s %n ",

**rsmd.getColumnName(i), rsmd.getColumnTypeName(i)**);

if (**rsmd.isNullable(i)** == 0)

nula = "NO";

else

nula = "SI";

System.out.printf(" Puede ser nula?: %s %n ", nula);

System.out.printf(" Máximo ancho de la columna: %d %n",

**rsmd.getColumnDisplaySize(i)**);

}// for

sentencia.close();

rs.close();

conexion.close();

} catch (ClassNotFoundException cn) {

cn.printStackTrace();

} catch (SQLException e) {

e.printStackTrace();

}

}// fin de main

}

Obtiene la siguiente información:

Número de columnas recuperadas: 3

Columna 1:

Nombre: dept\_no

Tipo: TINYINT

Puede ser nula?: NO

Máximo ancho de la columna: 2

Columna 2:

Nombre: dnombre

Tipo: VARCHAR

Puede ser nula?: SI

Máximo ancho de la columna: 15

Columna 3:

Nombre: loc

Tipo: VARCHAR

Puede ser nula?: SI

Máximo ancho de la columna: 15

Los métodos usados son los siguientes:

|  |  |
| --- | --- |
| **Método** | **Descripción** |
| int getColumnCount() | Devuelve el número de columnas devueltas por la consulta. |
| String getColumnName(int índiceColumna) | Devuelve el nombre de la columna cuya posición se indica *índiceColumna*. |
| String getColumnTypeName(int índiceColumna) | Devuelve el nombre del tipo de dato específico del sistema de bases de datos que contiene la columna indicada en *índiceColumna.* |
| int isNullable(int índiceColumna) | Devuelve 0 si la columna puede contener valores nulos. |
| int getColumnDisplaySize(int índiceColumna) | Devuelve el máximo ancho en caracteres de la columna indicada en *índiceColumna.* |
| ***Más información sobre métodos de ResultSetMetaData:***  [*https://docs.oracle.com/javase/8/docs/api/java/sql/ResultSetMetaData.html*](https://docs.oracle.com/javase/8/docs/api/java/sql/ResultSetMetaData.html) | |

Actividad 2.9

Visualiza información sobre las columnas de la tabla *empleados*.

**PROBAR JTABLE**

**Plugin WB Eclipse**

Abrir Eclipse Market Place y buscar Windows builder, aparecerá la versión 1.9….

Si así no se instala descargar el plugin de : http://www.eclipse.org/windowbuilder/download.php

Descargar el zip

Y para instalarlo Menú Help/Install new Software y Botón Add para seleccionar el archivo comprimido

Ver el documento Crear JTAble.docx para obtener los datos de una consulta en un control visual **Jtable**

**Para que devuelva el número de filas de la select, sin hacer consulta con count\*(), en todas las bases de datos, podemos utilizar una api nueva.**

**Estos métodos no funcionan en todas las BD**

**resul.beforeFirst(); resul.last(); resul.first(); (por ejemplo Oracle)**

Utilizamos un nuevo jar, lo añadiremos al build path, **javax.servlet.jsp.jstl-api-1.2.2.jar**

El jar si no lo tenéis se descarga de esta url

<https://jar-download.com/artifacts/javax.servlet.jsp.jstl/javax.servlet.jsp.jstl-api/1.2.2/source-code>

Y utilizaremos 2 clases nuevas ***Result*** y ***ResultSupport***

**import** javax.servlet.jsp.jstl.sql.\*;

Y las instrucciones son estas:

String sql = "SELECT \* FROM departamentos";

ResultSet resul = sentencia.executeQuery(sql);

**Result** result1 = **ResultSupport**.*toResult*(**resul**);

int numfilas = result1.getRowCount();

System.*out*.println("Num filas:"+ numfilas);