Uso de Statement actualizables con JDBC

Cuando hacemos una consulta a base de datos con un Statement(), obtenemos un ResultSet en el que podemos ir leyendo los resultados de la consulta. Este ResultSet normalmente es solo de lectura (no nos permite modificar la base de datos) y se va recorriendo secuencialmente, desde la primera fila de resultados a la última. Pues bien, podemos cambiar esta configuración al crear el Statement, de forma que obtengamos un ResultSet por el que podamos avanzar y retroceder, además de que nos permita modificar los datos. Lo mismo ocurre con los PreparedStatement.

Importante: esto funciona si la consulta es una consulta simple a una única tabla de base de datos y tiene que tener una clave primaria y que se haya seleccionado.

Como hemos dicho antes, por defecto, los **Result Set** son **no actualizables** y **no desplazables** (no nos podemos mover de arriba a abajo por ellos) si los creamos así:

Statement statement = connection.createStatement();

O si los creamos con un PreparedStatement así:

PreparedStatement statement = connection.prepareStatement(sql);

Por tanto, si queremos cambiar este comportamiento, tenemos que hacerlo pasando los parámetros correspondientes que nos permitan que tengan esas características:

Para un statement:

Statement statement = connection.createStatement(int resultSetType, int resultSetConcurrency);

Y para un prepared statement:

PreparedStatement statement = connection.prepareStatement(

String sql, int resultSetType, int resultSetConcurrency);

Los valores posibles para el tipo de Result Set (resultSetType) están definidos por las constantes siguientes del interfaz ResultSet:

TYPE_FORWARD_ONLY: Es el valor por defecto, no se puede avanzar y retroceder como queramos.

TYPE_SCROLL_INSENSITIVE: Podemos avanzar y retroceder por el result set, pero no es sensible a cambios de terceros en la base de datos. Es decir, no vemos esos datos desde nuestra aplicación cuando avanzamos y/o retrocedemos.

TYPE_SCROLL_SENSITIVE: Podemos avanzar y retroceder por el result set, y es sensible a cambios de terceros en la base de datos. Es decir, mientras no hemos cerrado el Result Set somos capaces de ver cambios de terceros.

Los valores posibles para concurrencia en el Result set (**resultSetConcurrency**):

CONCUR_READ_ONLY: Desde el Result set no se puede actualizar la base de datos. Es el comportamiento por defecto.

CONCUR_UPDATABLE: Se puede usar el result set para modificar la base de datos.

Por ejemplo, para un statement:

Statement statement =

connection.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE, ResultSet.CONCUR_UPDATABLE);

Y para un objeto PreparedStatement:

PreparedStatement statement = connection.prepareStatement(sql,

ResultSet.TYPE_SCROLL_INSENSITIVE, ResultSet.CONCUR_UPDATABLE);

En estos casos anteriores, se puede actualizar la base de datos, pero no se ven los cambios que se hayan producido en la base de datos mientras el result set esté abierto.

Es recomendable comprobar si la base de datos soporta result sets actualizables o no de la siguiente manera:

DatabaseMetaData metadata = connection.getMetaData();

boolean esActualizable = metadata.supportsResultSetConcurrency(

```
ResultSet.TYPE_SCROLL_INSENSITIVE,
ResultSet.CONCUR_UPDATABLE);

if (!esActualizable) {
        // terminamos
}
// continuamos
```

Vamos a ver ejemplos:

Actualizar la fila actual en un Result Set

Para realizar esta operación, tenemos que usar el método **updateXXX**, por ejemplo**:**

updateString(int columnIndex, String x)

updateString(String columnLabel, String x)

updateInt(int columnIndex, int x)

updateInt(String columnLabel, int x)

Como ves, se puede especificar la columna de la fila actual bien con su número de orden, bien con su nombre. Recuerda que el orden es el del result set de tu consulta, no el de la tabla.

Por otra parte, es más recomendable usar los nombres de las columnas que los índices, se entiende mejor lo que estamos haciendo.

Antes de llamar al método updateXXX tenemos que posicionarnos en la fila que queremos modificar. Esto se hace con el método **absolute(posición)**

Después de llamar a updateXXX tenemos que llamar al método **updateRow()** para confirmar (commit) los cambios en la base de datos.

Por ejemplo, si queremos modificar datos en una tabla estudiantes modificando la línea tercera:

```
        String sql = "SELECT * FROM estudiantes";

        Statement statement = conn.createStatement(
        ResultSet.TYPE_SCROLL_INSENSITIVE,
        ResultSet.CONCUR_UPDATABLE);
```

```
ResultSet result = statement.executeQuery(sql);

result.absolute(3); // Nos posicionamos en la fila tercera

result.updateString("name", "Nombre nuevo");
result.updateString("email", "correo nuevo");

result.updateRow();
```

Insertar una nueva fila en el Result Set

Ahora vamos a ver un ejemplo de inserción de nuevos valores

```
result.moveToInsertRow();

result.updateString("name", "Nombre nuevo");
result.updateString("email", "correo nuevo");

result.insertRow();

result.moveToCurrentRow();
```

Como ves, lo primero es mover el cursor a la posición adecuada de inserción, para ello usamos el método **moveToInsertRow()**. Después usamos updateXXX()para especificar los valores que queremos introducir en los campos de la fila y luego, llamamos a **insertRow()** para grabar los cambios en la base de datos.

El método **moveToCurrentRow()** mueve el curso para que mueva el cursor a la fila en el ResultSet que era la fila actual antes de que se realizase la operación de inserción..

Borrar la fila actual en el Resul Set

Para borrar la **fila actual** tenemos que llamar al método **deleteRow**():

```
result.deleteRow();
```

NOTA:

Tanto updateRow() como insertRow() y deleteRow() lanzan SQLException si ocurre algún error en la base de datos o el result set está cerrado o en un modo que no permite las operaciones.

Veamos ahora como podemos movernos por el result set

```
Statement stmt = con.createStatement(
        ResultSet.TYPE_SCROLL_INSENSITIVE,
        ResultSet.CONCUR_READ_ONLY);

ResultSet rs = stmt.executeQuery("SELECT * FROM alumnos;");

rs.last();
while (rs.previous())
   System.out.println(rs.getString(1));
```

Comprueba qué ocurre si ponemos la característica de solo poder avanzar hacia delante.

Información sobre la Base de Datos (metadatos)

Hasta ahora asumíamos que el programador conocía la estructura de la BD a la que estaba accediendo (el nombre de las tablas, su tipo de datos, etc.). Sin embargo, en determinadas aplicaciones es posible que necesitemos obtener esa información directamente de la propia BD. Esta información se conoce con el nombre de **Metadatos**, datos sobre la estructura de la base de datos. Para obtener y manejar esta información disponemos del interfaz **DatabaseMetaData**.

Su uso es el siguiente:

```
DatabaseMetaData dbmd = con.getMetaData();
```

Ya tenemos toda la información de la base de datos apuntada por dbmd. Ahora debemos manejarla. Para ello disponemos de más de 100 métodos en la clase DatabaseMetaData. Vamos a ver alguno.

Los siguientes métodos proporcionan información genérica de la base de datos:

getDatabaseProductName	Devuelve el nombre de la BD
getDatabaseProductVersion	Ídem versión
getDriverName	Ídem nombre del driver
supportsResultSetType(int)	Pasándole por parámetro algunos de los tipos de ResultSet (TYPE_FORWARD_ONLY, TYPE_SCROLL_SENSITIVE, etc.), nos devuelve si la BD los soporta

Podemos obtener toda la información referente a las tablas en la BD. Para ello llamamos al siguiente método:

```
String[] tipos = {"TABLE"};
ResultSet resul = dbmd.getTables(null, null, null, tipos);
```

Este método devuelve un objeto de la clase ResultSet. Los parámetros pasados al método sirven para:

- Los dos primeros para especificar el catálogo y el esquema de los que se obtendrá la información.
- El tercer parámetro es el nombre de la tabla a obtener.
- Y el último los tipos (en nuestro caso, TABLE, también puede ser "VIEW","SYSTEM TABLE", etc.)

Los tres primeros parámetros son de tipo cadena y permiten utilizar comodines. Por ejemplo, si quisiéramos obtener todas las tablas cuyo nombre empiece por Nom, pasaríamos el parámetro "Nom%".

Al devolver un objeto ResultSet podemos visitarlo tal como hemos hecho siempre.

Vemos el ejemplo del ejercicio Empresa y otro ejemplo sobre la base de datos instituto

Como vemos, en el último ejemplo hemos accedido a todas las tablas de la BD. Para cada tabla obtenemos la información de sus columnas, que también es devuelta mediante un ResultSet. Visitamos todas las columnas y obtenemos la información de cada una de ellas y por último obtenemos las claves primarias de la tabla.
También podemos obtener los metadatos de una consulta, es decir, directamente de un ResultSet obtenido al llamar a un método executeQuery.