Bases de datos con JDBC

Sesión 2: Consulta a una BD

Índice

- Restricciones, movimientos y actualizaciones en el *ResultSet*
- Sentencias de actualización
- Otras llamadas a la BD

Restricciones en la llamada

- Cuando interrogamos una BD, el resultado devuelto puede ser extremadamente grande
- Podemos limitar el número de registros a devolver
- Disponemos de dos métodos en la clase *Statement*, *getMaxRows* y *setMaxRows* que nos devuelve y cambia el máximo número de filas
- Por defecto está a 0 (no hay restricciones)
- Si cambiamos el valor (p.e. 30), una consulta no devolverá más de 30 registros

Movimientos en el *ResultSet*

- Hasta ahora hemos utilizado el método *next* para movernos por el *ResultSet*
- Podemos crear un *ResultSet* arrastable que nos permita movernos de forma no lineal
- Primero tenemos que crear un objeto *Statement* de la siguiente manera:
  - Statement createStatement (int resultSetType, int resultSetConcurrency)

Valores de resultSetType

- ResultSet.TYPE_FORWARD_ONLY Valor por defecto. Sólo permite el desplazamiento hacia delante
- ResultSet.TYPE_SCROLL_INSENSITIVE Permite el desplazamiento. Si se cambian los datos que estamos visualizando en la BD, los datos mostrados no cambian
- ResultSet.TYPE_SCROLL_SENSITIVE Permite el desplazamiento y cualquier cambio en la BD afecta a los datos visualizados
- El mostrar los cambios que se produzcan en la BD dependerá de la BD y del driver que estemos utilizando

Valores de resultSetConcurrency

- ResultSet.CONCUR_READ_ONLY Valor por defecto. Cualquier cambio en el *ResultSet* no tiene efecto en la BD
- ResultSet.CONCUR_UPDATABLE Permite que los cambios efectuados en el *ResultSet* tengan efecto en la BD
- Para poder actualizar los datos de la BD debemos crear un *Statement* con, al menos, TYPE_FORWARD_SENSITIVE y CONCUR_UPDATABLE

Movimientos en el *ResultSet*

- Una vez realizada la consulta y obtenido el *ResultSet* arrastable, podemos usar:

next	**Pasa a la siguiente fila**
previous	**Ídem fila anterior**
last	**Ídem última fila**
first	**Ídem primera fila**
absolute(int fila)	**Pasa a la fila número fila**
relative(int fila)	**Pasa a la fila número fila desde la actual**
getRow	**Devuelve la número de fila actual**
isLast	**Devuelve si la fila actual es la última**
isFirst	**Ídem la primera**

Modificación del *ResultSet*

- Para modificar un campo del registro actual, usaremos *updateXXXX* (igual que *getXXXX*)
- *updateXXXX* recibe dos parámetros, nombre del campo a modificar y nuevo valor del campo
- Para que los cambios tengan efecto debemos llamar a *updateRow*
  - rs.updateString("nombre","manolito");
    rs.updateRow();

Modificación del *ResultSet*

- Para desechar los cambios del registro actual, antes de llamar a *updateRow,* llamaremos a *cancelRowUpdates*
- Para borrar el registro actual, usaremos *deleteRow*. La llamada a este método deja una fila vacía en el *ResultSet*. Si intentamos acceder a ese registro se producirá una excepción
- El método *rowDeleted* nos dice si el registro ha sido eliminado

Restricciones

- Para poder hacer uso de un *ResultSet* arrastable, la sentencia SELECT que lo genera debe:
  - Referenciar sólo una tabla
  - No contener una claúsula *join* o *group by*
  - Seleccionar la clave primaria de la tabla
- En el *ResultSet* disponemos de un registro especial, llamado de inserción.
- Nos permite introducir nuevos registros en la tabla

Sentencias de actualización

- La clase *Statement* incorpora un método para realizar actualizaciones: *executeUpdate*
- Recibe una cadena que es la sentencia SQL a ejecutar:
  - CREATE (creación de tablas)
  - INSERT (inserción de datos)
  - DELETE (borrado de datos)
- El método *executeUpdate* devuelve un entero que indica el número de registros afectados (CREATE devuelve siempre 0)

Ejemplos

- String st_crea = "CREATE TABLE ALUMNOS (exp INTEGER, nombre VARCHAR(32), sexo CHAR(1), PRIMARY KEY (exp))";
  stmt.executeUpdate(st_crea);
- String st_inserta = "INSERT INTO ALUMNOS (exp, nombre) VALUES(1285, 'Manu', 'M')";
  stmt.executeUpdate(st_inserta);
- String st_actualiza = "UPDATE FROM ALUMNOS SET sexo = ‘H' WHERE exp = 1285";
  stmt.executeUpdate(st_actualiza);
- String st_borra = "DELETE FROM ALUMNOS WHERE exp = 1285";
  stmt.executeUpdate(st_borra);

Otras llamadas a la BD

- Si no conocemos de antemano el tipo de consulta (la ha introducido el usuario), podemos utilizar el método *execute* de la clase *Statement*.
- El método devuelve un valor booleano, siendo cierto si hay resultados y falso en el caso de una sentencia de actualización
- Si es falso, podemos llamar al método *getUpdateCount* de *Statement* que nos dice el número de registros afectados

Otras llamadas a la BD

- Si hay resultados, los podemos obtener con el método *getResultSet* de *Statement*. Este método devuelve un *ResultSet*
- Si hemos ejecutado un procedimiento en la BD, es posible que tengamos más de un *ResultSet*
- El método *getMoreResult* nos devuelve cierto si existen más resultados. Después de esta llamada podemos volver a llamar a *getResultSet* para conseguir el siguiente resultado

Otras llamadas a la BD

- Si queremos ejecutar varias sentencias SQL a la vez, podemos utilizar el método *executeBatch*
- No permite sentencias de tipo SELECT
- Para añadir sentencias usaremos el método *addBatch*
- *executeBatch* devuelve un array de enteros indicando el número de registros afectados en cada sentencia

Ejemplo

- stmt.addBatch("INSERT INTO ALUMNOS(exp, nombre) VALUES(1285, 'Manu', 'M')");
- stmt.addBatch("INSERT INTO ALUMNOS(exp, nombre) VALUES(1299, 'Miguel', 'M')");
- int[] res = stmt.executeBatch();

¿Preguntas...?