EJECUCIÓN DE SENTENCIAS DE MANIPULACIÓN DE DATOS

Javier García-Retamero Redondo

EJECUCIÓN DE SENTENCIAS DE MANIPULACIÓN DE DATOS

CONSULTAS

SENTENCIAS PREPARADAS

- El objetivo principal es ejecutar consultas sobre los datos almacenados.
- Utilizaremos la interfaz **PreparedStatement**
- Podemos utilizar marcadores de posición para evitar tener que concatenar cadenas de caracteres a la hora de formar la cadena SQL.
- De esta forma precompilaremos la sentencia una vez y le pasaremos los valores en tiempo de ejecución.
- Con Statement normales, se compila la sentencia cada vez que se ejecuta.

SENTENCIAS PREPARADAS

- **Preparamos el string para la llamada:**

  String sql = "SELECT * FROM departamentos WHERE dept_no=?";

  1

- **Creamos un objeto llamando al método prepareStatement:**

  PreparedStatement sentencia=conexion.prepareStatement(sql);

  …La sentencia se precompila…

- **Le damos valor a los parámetros de entrada:**

  sentencia.setInt(1, dep);

- **Ejecutamos la sentencia ya precompilada con los valores suministrados:**

  Optional <ResultSet> rs = sentencia.executeQuery();

- **Cerramos el ResultSet y el PreparedStatement:**

  rs.close();

  sentencia.close();

Si la variable del paréntesis no fuera del mismo tipo, utilizaríamos:

Integer.parseInt(v),
…
Float.parseFloat(v)

SENTENCIAS PREPARADAS

Método	Tipo SQL
setString (i,String)	VARCHAR
setBoolean(i,boolean)	BIT
setByte(i,byte)	TINYINT
setShort(i,short)	SMALLINT
setInt (i,int)	INTEGER
setLong (i,long)	BIGINT
setFloat (i,float)	FLOAT
setDouble(i,double)	DOUBLE
setDate (i,Date)	DATE
setNull(i, int tipoSQL)	Pone a null una posición de un tipo especificado en java.sql.Types
setObject	Cualquier objeto

RECORRER EL RESULTADO

- Pasos para recorrer el resultado (ResulSet) de una consulta:
  - Al ejecutar la sentencia el resultado se devolverá en un objeto que es similar a una tabla con los datos.
  - Comienza con un puntero que está antes del primer registro.
  - Para avanzar al siguiente utilizaremos: **next()**.
  - Para ir al último registro utilizaremos: **last()**.
  - Para situar el puntero al comienzo. **beforeFirst()**.
  - Para obtener los valores de las columnas utilizaremos los métodos "**get**" indicando el nombre de la columna.

```
if (rs.isPresent())
        while (rs.get().next()) {
                System.out.printf("%d, %s, %s %n",
                        rs.getInt("dept_no"),
                        rs.getString("dnombre"),
                        rs.getString("loc"));
        }
}
```

EJECUCIÓN DE SENTENCIAS DE MANIPULACIÓN DE DATOS

INSERCIÓN, MODIFICACIÓN Y BORRADO

INSERCIÓN, MODIFICACIÓN, BORRADO

- **int executeUpdate(String):**
  - Se utiliza para:
    - DML (insert, delete y update): Devuelve el número de filas afectadas.

INSERCIÓN

- **Preparamos el string para la llamada:** 1 2 3

  String sql = "INSERT INTO departamentos VALUES (?,?,?)";

- **Creamos un objeto llamando al método prepareStatement:**

  PreparedStatement sentencia=
  conexion.prepareStatement(sql, PreparedStatement.RETURN_GENERATED_KEYS);

  …La sentencia se precompila…

- **Le damos valor a los parámetros de entrada:**

  sentencia.setInt(1, dep);

  sentencia.setString (2 , dnombre);

  sentencia.setString (3 , loc);

RETURN_GENERATED_KEYS:
Nos permitirá obtener el ID en caso de que la clave sea un campo autoincremental. Si no es incremental, no es necesario ponerlo.

Para obtener el valor generado podemos ejecutar:

preparedStatement.getGeneratedKeys()

- **Ejecutamos la sentencia ya precompilada con los valores suministrados:**

  int numeroFilas= sentencia.**executeUpdate**();

MODIFICACIÓN

- **Preparamos el string para la llamada:**

  String sql = "UPDATE departamentos SET dnombre = ?, loc = ? WHERE dept_no=?";

  1 2 3

- **Creamos un objeto llamando al método prepareStatement:**

  PreparedStatement sentencia=conexion.prepareStatement(sql);

  …La sentencia se precompila…

- **Le damos valor a los parámetros de entrada:**

  sentencia.setString (1 , dnombre);

  sentencia.setString (2 , loc);

  sentencia.setInt(3, dep);

- **Ejecutamos la sentencia ya precompilada con los valores suministrados:**

  int numeroFilas= sentencia.**executeUpdate**();

BORRADO

- **Preparamos el string para la llamada:**

  String sql = "delete from Departamentos where dept_no =(?)";

- **Creamos un objeto llamando al método prepareStatement:**

  PreparedStatement sentencia=conexion.prepareStatement(sql);

  …La sentencia se precompila…

- **Le damos valor a los parámetros de entrada:**

  sentencia.setInt(1, dep);

- **Ejecutamos la sentencia ya precompilada con los valores suministrados:**

  int numeroFilas= sentencia.**executeUpdate**();

EJECUCIÓN DE SENTENCIAS DE MANIPULACIÓN DE DATOS

DDL

DDL

- **int executeUpdate(String):**
  - Se utiliza para:
    - DDL (create, drop, alter): Devuelve 0

DDL

- **Preparamos el string para la llamada:**

  String sql = "ALTER TABLE departamentos ADD (tlfn NUMBER(9))";

- **Creamos un objeto llamando al método createStatement:**

  Statement sentencia = conexion.createStatement();

- **Ejecutamos la sentencia:**

  int resultado= sentencia.**executeUpdate**(sql);

EJECUCIÓN DE SENTENCIAS DE MANIPULACIÓN DE DATOS

CUALQUIER SENTENCIA

CUALQUIER SENTENCIA

- **boolean execute(String):**
  - Vale para ejecutar cualquier sentencia SQL.
  - Devuelve TRUE: Cuando devuelve un Resulset
    - Utilizar getResulset() para obtener el resulset devuelto
  - Devuelve FALSE: Cuando devuelve un recuento de actualizaciones o no hay resultados.
    - Utilizar getUpdateCount() para recuperar el valor devuelto

CUALQUIER SENTENCIA

```
String sql="SELECT * FROM departamentos";
Connection conexion = DriverManager.getConnection(……);
Statement sentencia = conexion.createStatement ();

Boolean valor = sentencia.execute(sql);

if (valor) {
        Resulset rs = sentencia.getResulset();
        ……Recorremos el resulset
} else {
        int f = sentencia.getUpdateCount();
}
```

GESTIÓN DE ERRORES

CONSIDERACIONES

CONSIDERACIONES

- **Con SQLException podemos acceder a cierta información:**

```
catch (SQLException e) {
        System.out.println ("Ha ocurrido un error:");
        System.out.println ("Mensaje: " +e.getMessage());
        System.out.println ("SQL Estado: " +e.getSQLState());
        System.out.println ("Código de error: " +e.getErrorCode());
}
```

Método	Función
getMessage()	Mensaje que describe el error.
getSQLState()	Estado definido por el estándar X/OPEN SQL
getErrorCode()	Muestra el código de error que se ha producido en la BBDD. Por ejemplo en Oracle correspondería con el número que aparece en los ORA-