12.D. Ejercicio completo.

Sitio: <u>Formación Profesional a Distancia</u>

Curso: Programación

Libro: 12.D. Ejercicio completo.

Imprimido por: Iván Jiménez Utiel

Día: domingo, 17 de mayo de 2020, 16:28

Tabla de contenidos

- 1. Ejecución de consultas sobre la base de datos.
- 1.1. Recuperación de información (I).
- 1.2. Recuperación de información (II).
- 1.3. Actualización de información.
- 1.4. Adición de información.
- 1.5. Borrado de información.

1. Ejecución de consultas sobre la base de datos.

Para operar con una base de datos ejecutando las consultas necesarias, nuestra aplicación deberá hacer las operaciones siguientes:

- Cargar el conector necesario para comprender el protocolo que usa la base de datos en cuestión.
- Establecer una conexión con la base de datos.
- Enviar consultas SQL y procesar el resultado.
- Liberar los recursos al terminar.
- **Gestionar los errores** que se puedan producir.

Podemos utilizar los siguientes tipos de sentencias:

- Statement: para sentencias sencillas en SQL.
- PreparedStatement: para consultas preparadas, como por ejemplo las que tienen parámetros.
- CallableStatement: para ejecutar procedimientos almacenados en la base de datos.

El API JDBC distingue dos tipos de consultas:

- Consultas: SELECT. Para las sentencias de consulta que obtienen datos de la base de datos, se emplea el método ResultSet executeQuery(String sql). El método de ejecución del comando SQL devuelve un objeto de tipo ResultSet que sirve para contener el resultado del comando SELECT, y que nos permitirá su procesamiento.
- **Actualizaciones**: INSERT, UPDATE, DELETE, sentencias DDL. Para estas sentencias se utiliza el <u>método</u> executeUpdate(String sql)

Autoevaluación

Para poder enviar consultas a la base de datos hemos tenido que conectarnos a ella previamente.

- Verdadero.
- Falso.

1.1. Recuperación de información (I).

Las consultas a la base de datos se realizan con sentencias SQL que van "**embebidas**" en otras sentencias especiales que son propias de Java. Por tanto, podemos decir que las consultas SQL las escribimos como parámetros de algunos métodos Java que reciben el **String** con el texto de la consulta SQL.

Las consultas devuelven un ResultSet, que es una <u>clase</u> java parecida a una lista en la que se aloja el resultado de la consulta. Cada elemento de la lista es uno de los registros de la base de datos que cumple con los requisitos de la consulta.

El ResultSet no contiene todos los datos, sino que los va obteniendo de la base de datos según se van pidiendo. La razón de esto es evitar que una consulta que devuelva una cantidad muy elevada de registros, tarde mucho tiempo en obtenerse y sature la memoria del programa.

Con el ResultSet hay disponibles una serie de métodos que permiten movernos hacia delante y hacia atrás en las filas, y obtener la información de cada fila.

Por ejemplo, para obtener: nif, nombre, apellidos y teléfono de los clientes que están almacenados en la tabla del mismo nombre, de la base de datos *notarbd* que se creó anteriormente, haríamos la siguiente consulta:

```
// Preparamos la consulta y la ejecutamos

Statement s = n.createStatement();

ResultSet rs = s.executeQuery ("SELECT NIF, NOMBRE,"

+ "APELLIDOS, TELÉFONO FROM CLIENTE");
```

El <u>método</u> next() del ResultSet hace que dicho <u>puntero</u> avance al siguiente <u>registro</u>. Si lo consigue, el <u>método</u> next() devuelve true. Si no lo consigue, porque no haya más registros que leer, entonces devuelve false.

1.2. Recuperación de información (II).

El <u>método</u> executeQuery devuelve un <u>objeto</u> ResultSet para poder recorrer el resultado de la consulta utilizando un <u>cursor</u>.

Para obtener una columna del <u>registro</u> utilizamos los métodos <u>get</u>. Hay un <u>método</u> <u>get</u> para cada tipo básico Java y para las cadenas.

Un <u>método</u> interesante es <u>wasNull</u> que nos informa si el último valor leído con un <u>método</u> get es nulo.

Cuando trabajamos con el ResultSet, en cada <u>registro</u>, los métodos getInt(), getString(), getDate(), etc., nos devuelve los valores de los campos de dicho <u>registro</u>. Podemos pasar a estos métodos un índice (que comienza en 1) para indicar qué columna de la tabla de base de datos deseamos, o bien, podemos usar un String con el nombre de la columna (tal cual está en la tabla de base de datos).

El mismo código copiable:

```
// Obtener la conexión

Connection con = DriverManager.getConnection(connectionUrl);

// Preparamos la consulta

Statement s = con.createStatement();

ResultSet rs = s.executeQuery ("SELECT NIF, NOMBRE,"

+ "APELLIDOS, TELÉFONO FROM CLIENTE");

// Iteramos sobre los registros del resultado

while (rs.next())

System.out.println (rs.getString("NIF") + " " +

rs.getString (2) + " " +

rs.getString (3) + " " +

rs.getString (4));
```

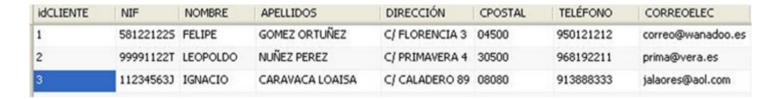
Autoevaluación

Para obtener un entero almacenado en uno de los campos de un <u>registro</u>, trabajando con el <u>ResulSet</u> emplearemos el <u>método</u> getInt().

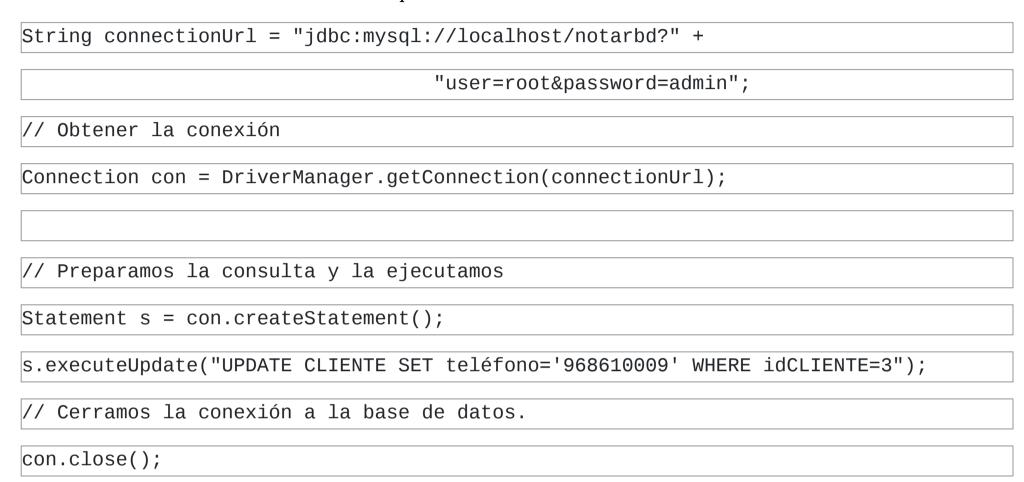
- Verdadero.
- Falso.

1.3. Actualización de información.

Respecto a las consultas de actualización, executeUpdate, retornan el número de registros insertados, registros actualizados o eliminados, dependiendo del tipo de consulta que se trate.



Supongamos que tenemos varios registros en la tabla Cliente, de la base de datos notarbd con la que seguimos trabajando. Si quisiéramos actualizar el teléfono del tercer <u>registro</u>, que tiene idCLIENTE=3 y ponerle como nuevo teléfono el 968610009 tendríamos que hacer:



1.4. Adición de información.

Si queremos añadir un <u>registro</u> a la tabla Cliente, de la base de datos con la que estamos trabajando tendremos que utilizar la sentencia <u>INSERT INTO</u> de SQL. Al igual que hemos visto en el apartado anterior, utilizaremos <u>executeUpdate</u> pasándole como parámetro la consulta, de inserción en este caso.

Así, un ejemplo sería:

```
// Preparamos la consulta y la ejecutamos

Statement s = con.createStatement();
s.executeUpdate( "INSERT INTO CLIENTE" +

" (idCLIENTE, NIF, NOMBRE, APELLIDOS, DIRECCIÓN, CPOSTAL, TELÉFONO, CORREOELEC)" +
 " VALUES (4, '66778998T', 'Alfredo', 'Gates Gates', 'C/ Pirata 23','20400',
 '891222112', 'prueba@eresmas.es' )");
```

Autoevaluación

Al añadir registros a una tabla de una base de datos, tenemos que pasar como parámetro al executeUpdate(), una sentencia SQL del tipo: DELETE...

- Verdadero.
- Falso.

1.5. Borrado de información.

Cuando nos interese eliminar registros de una tabla de una base de datos, emplearemos la sentencia SQL: **DELETE**. Así, por ejemplo, si queremos eliminar el <u>registro</u> a la tabla Cliente, de nuestra base de datos y correspondiente a la persona que tiene el nif: 66778998T, tendremos que utilizar el código siguiente.

```
// Preparamos la consulta y la ejecutamos

Statement s = con.createStatement();

numReg = res.executeUpdate( "DELETE FROM CLIENTE WHERE NIF= '66778998T' " );

// Informamos del número de registros borrados

System.out.println ("\nSe borró " + numReg + " registro\n");
```

Autoevaluación

Al ejecutar el borrado de un <u>registro</u> mediante executeUpdate(...), no podemos saber si el borrado eliminó alguna fila o no.

- Verdadero.
- Falso.