Acceso a Datos

UT3. MANEJO DE CONECTORES
BASES DE DATOS RELACIONALES
SENTENCIAS DDL Y DML

6. Ejecución de sentencias DML

- En SQL, las sentencias DML son aquellas que nos permiten trabajar con los datos de una BBDD.
  - **SELECT** - para obtener datos de una base de datos.
  - **INSERT** - para insertar datos a una tabla.
  - **UPDATE** - para modificar datos existentes dentro de una tabla.
  - **DELETE** - elimina todos los registros de la tabla; no borra los espacios asignados a los registros.
- Para ejecutar cualquiera de estas sentencias debemos usar objetos:
  - **Statement** → Permite ejecutar sentencias SQL sin parámetros
  - **PreparedStatement** → Permite ejecutar sentencias SQL con parámetros

6. Ejecución de sentencias DML

La interface Statement

- La interface **Statement** permite crear objetos Statement, al tratarse de una interface no podemos instanciar directamente este tipo de objetos sino que debemos valernos del método createStatement() de la clase Connection.
- Los principales métodos de esta interface son:

Método	Descripción
ResultSet executeQuery(String query)	Se utiliza para ejecutar sentencias SQL que recuperan datos. Devuelve un objeto ResultSet con los datos recuperados.
int executeUpdate(String Query)	Se utiliza para ejecutar sentencias de manipulación como Insert, Update y Delente. Devuelve un entero indicando el número de registros que se han visto afectados.
boolean execute(String Query)	Se puede utilizar para ejecutar cualquier consulta SQL. En caso de que la sentencia devuelva un ResultSet, el método execute() devuelve true y debemos recuperar el objeto ResultSet a través del método getResultSet(). En caso contrario (Cualquier otro tipo de sentencia) nos devolverá false y debemos usar el método getUpdateCount() para recuperar el valor devuelto.

6. Ejecución de sentencias DML

La interface PreparedStatement

- Es muy habitual el uso de variables dentro de una sentencia SQL:
  - Valores a insertar, actualizar o borrar
  - Filtros en consultas de selección.
  - Etc.
- Para este tipo de consultas con parte variable debemos hacer uso de las llamadas Sentencias Preparadas (Prepared Statements)
- La interface PreparedStatement nos va a permitir crear sentencias SQL placeholders (marcadores de posición) que representan los datos que serán agregados posteriormente (Variables). Estos placeHolders se representan mediante signos de interrogación (?)

```
String sql = "INSERT INTO empleados(emp_no, apellido, oficio, fecha_alt, salario, dept_no)"
        + "VALUES (?,?,?,?,?,?)";
```

6. Ejecución de sentencias DML

La interface PreparedStatement

- Cada Placeholder tiene un índice, siendo 1 el primer elemento que se encuentre en la cadena.
- Antes de ejecutar una PreparedStatement es necesario asignar valores a cada uno de los placeholders.
- La ventaja que nos ofrece esto frente a la sentencias tradicionales es el hecho de poder preparar consultas (precompilar) una sola vez y tener la posibilidad de ejecutarlas tantas veces como sea necesario con diferentes valores de entrada. Ofrecen una gran flexibilidad.
- Los principales métodos de PreparedStatement son:

Método	Descripción
ResultSet executeQuery()	Similar a su homónima en la interface Statement
int executeUpdate()	Similar a su homónima en la interface Statement
boolean execute()	Similar a su homónima en la interface Statement

6. Ejecución de sentencias DML

La interface PreparedStatement

- Además, contamos con una serie de métodos par asignar valores a cada uno de los placeholders.

Método	Tipo SQL
void setString(int index, String valor)	VARCHAR
void setBoolean(int índice, boolean valor)	BIT
void setShort(int índice, short valor)	SMALLINT
void setInt(int indice, int valor)	INTEGER
void setLong(int indice, int valor)	BIGINT
void setFloat(int índice, float valor)	FLOAT
void setDouble(int índice, double valor)	DOUBLE
void setBytes(int índice, byte[] valor)	VARBINARY
void setDate(int índice, Date valor)	DATE
void setTime(int índice, Time valor)	TIME

6. Ejecución de sentencias DML

```
// Preparamos la consulta
String sql = "INSERT INTO empleados(emp_no, apellido, oficio, fecha_alt, salario, dept_no)"
        + "VALUES (?,?,?,?,?,?)";

PreparedStatement sentenciaPreparada = conexion.prepareStatement(sql);

sentenciaPreparada.setInt(1, 20);
sentenciaPreparada.setString(2, "FERNANDEZ");
sentenciaPreparada.setString(3, "ANALISTA");

// String str = "2023-01-01";
// Date date = Date.valueOf(str); // java.sql.Date
// sentenciaPreparada.setDate(4, date);
sentenciaPreparada.setDate(4, new Date(System.currentTimeMillis()));

sentenciaPreparada.setFloat(5, 1500);
sentenciaPreparada.setInt(6, 40);

int resultado = sentenciaPreparada.executeUpdate();

System.out.println(resultado);

sentenciaPreparada.close();
conexion.close();
```

6. Ejecución de sentencias DML

La clase ResultSet

- Mediante un objeto de la clase ResultSet podemos recoger los valores devueltos por una consulta de selección.
- A través de un objeto ResultSet podemos acceder al valor de cualquier columna de la tupla actual a través de su posición o su nombre.
- También podemos obtener información general sobre la consulta (Como el número de columnas, su tipo, etc) mediante el método **getMetada()**.
- Para acceder a cualquiera de las columnas de la tupla actual haremos uso de los métodos getXXX().

6. Ejecución de sentencias DML

La clase ResultSet

Método	Tipo Java
getString(int numeroCol)	String
getBoolean(int numeroCol)	boolean
getShort(int numeroCol)	short
getInt(int numeroCol)	int
getLong(int numeroCol)	long
getFloat(int numeroCol)	float
getDouble(int numeroCol)	double
getBytes(int numeroCol)	byte[]
getDate(int numeroCol)	Date
getTime(int numeroCol)	Time
getTimeStamp(int numeroCol)	Timestamp

Método	Tipo Java
getString(String nombreCol)	String
getBoolean(String nombreCol)	boolean
getShort(String nombreCol)	short
getInt(String nombreCol)	int
getLong(iString nombreCol)	long
getFloat(iString nombreCol)	float
getDouble(String nombreCol)	double
getBytes(String nombreCol)	byte[]
getDate(String nombreCol)	Date
getTime(String nombreCol)	Time
getTimeStamp(String nombreCol)	Timestamp

6. Ejecución de sentencias DML

La clase ResultSet

- Para recorrer un ResultSet haremos uso del método next() como ya vimos en ejemplos anteriores.
- Además del número de columna, podemos utilizar directamente el nombre de los campos.

```
while(resultado.next()) {
    String nif= resultado.getString("nif");
    String nombre= resultado.getString("nombre");
    String apellidos= resultado.getString("Apellidos");
    Double salario= resultado.getDouble("salario");
    System.out.println(nif+" "+ nombre  +" "+apellidos+" "+salario);
}
```

7. Ejecución de sentencias DDL

- Aunque el grueso de las operaciones que cualquier operaciones que se realizan desde una aplicación cliente contra una BBDD sea operaciones de manipulación de datos, pueden darse casos en los que nos veamos obligados a realizar operaciones de definición.
  - Acabamos de instalar una aplicación en un nuevo equipo y necesitamos crear, de forma automatizada, una BBDD local para su funcionamiento (El caso más habitual).
  - Tras una actualización de software la estructura de la BBDD ha cambiado y debemos actualizarla desde nuestra aplicación Java.
  - Desconocemos la estructura de la BBBD y queremos realizar consultas dinámicas sobre la misma.
- En este apartado estudiaremos los mecanismos y procedimientos necesarios para solventar cualquiera de estas situaciones.

7. Ejecución de sentencias DDL

Definición y modificación de estructuras

Las sentencias DDL tradicionales no dejan de ser sentencias SQL y, por tanto, las podremos ejecutar de la forma que ya hemos estudiado a través de la interfaz **Statement/PreparedStatement**.

```
Connection conexion=
        DriverManager.getConnection("jdbc:mysql://localhost", "root", "");
Statement sentencia= conexion.createStatement();
String sql= "Sentencia DDL";
int resultado= sentencia.executeUpdate(sql);
System.out.println(resultado);
conexion.close();
sentencia.close();
```

7. Ejecución de sentencias DDL

Definición y modificación de estructuras

- Podemos hacer uso del modificador IF NOT EXISTS dentro de nuestro código SQL para asegurarnos de que la base de datos se crea antes de iniciar los procesos CRUD de nuestra aplicación.

```
Connection conexion=
        DriverManager.getConnection("jdbc:mysql://localhost/animales", "root", "");
Statement sentencia= conexion.createStatement();
String sql= "Create table if not exists mascotas("
        + "codigo varchar(3),"
        + "nombre varchar(20),"
        + "Constraint mascotas_PK PRIMARY KEY(codigo)"
        + ");";
int resultado= sentencia.executeUpdate(sql);
System.out.println(resultado);
conexion.close();
sentencia.close();
```

7. Ejecución de sentencias DDL

Definición y modificación de estructuras

- A través de este mecanismo podríamos crear bases de datos y sus correspondientes estructuras (Tablas, restricciones, vistas, índices…).
- Es una práctica muy habitual cuando vamos a usar una BBDD local, en la actualidad es muy empleado en aplicaciones para dispositivos móviles, aplicaciones locales de gestión, gestores de contenido web, etc.
- Incluso, a través de este diseño podríamos crear un cliente de bases de datos similar a MySQLWorkbench o el propio HeidiSQL.

7. Ejecución de sentencias DDL

Definición y modificación de estructuras

- En ocasiones puede darse el caso de que no conozcamos la estructura de una BBDD, por suerte esta información se encuentra almacenada en los denominados metaobjetos de la base de datos.
- La interface DatabaseMetaData proporciona información sobre la base de datos a través de múltiples métodos de los cuales es posible obtener una gran cantidad de información.

Método	Descripción
ResultSet getTables()	Proporciona información sobre diferentes tipos de objeto de la base de datos
ResultSet getColumns()	Devuelve información sobre las columnas de una tabla
ResultSet getPrimaryKeys()	Devuelve las claves primarias.
ResultSet getExportedKeys()	Devuelve las claves ajenas que apuntan a una tabla determinada.
ResultSet getImportedKeys()	Devuelve las claves ajenas de una tabla.

7. Ejecución de sentencias DDL

Definición y modificación de estructuras

- ResultSet getTables(String catálogo, String esquema, String patronDeTabla, String tipos)
  - Catálogo → Catálogo de la BBDD al que estamos apuntando. Con null indicamos que estamos apuntando al actual.
  - Esquema → Esquema del cual queremos obtener información. Con null indicamos que queremos información del esquema actual.
  - Patrón → Permite refinar la búsqueda indicando mediante un patrón los nombres o parte de los nombres que estamos buscando. Similar al funcionamiento de la cláusula like.
  - Tipos → Debemos indicar que tipos de objeto queremos obtener, estos tipos pueden ser: TABLE, VIEW, SYSTEM TABLE, GLOBAL TEMPORARY, LOCAL TEMPORARY, ALIAS y SYNONYM.

7. Ejecución de sentencias DDL

Definición y modificación de estructuras

- ResultSet getTables(String catálogo, String esquema, String patronDeTabla, String tipos)

```
DatabaseMetaData dbmd=  conexion.getMetaData();
String tipos[]= {"TABLE", "VIEW"};
ResultSet tablas=dbmd.getTables(null, null, null, tipos);
while(tablas.next()){
    System.out.println(
            tablas.getString("TABLE_CAT")+
            tablas.getString("TABLE_SCHEM")+
            tablas.getString("TABLE_NAME")+
            tablas.getString("TABLE_TYPE"));
}
```

Dudas y preguntas