UA04. GESTIÓN DE LA INFORMACIÓN

4.5. Sentencias de modificación de datos

Entendemos como sentencias de modificación de datos todas aquellas que actualicen o modifiquen de algún modo uno o más registros en una tabla. Este tipo de comando SQL contempla la manipulación de los datos presentes en la base de datos, y serán prácticamente la gran mayoría de los comandos conocidos por SQL:

- INSERT: añadirá una fila en la tabla de base de datos indicada.
- DELETE: se encarga de borrar información de una tabla de la base de datos.
- UPDATE: se encarga de modificar los registros de una tabla de la base de datos.

Estas sentencias nos permitirán borrar datos, insertar nuevos registros o actualizar uno existente. El procedimiento para crear estas sentencias dentro de nuestra aplicación Java es parecido a los ejemplos anteriores, cambiará la consulta.

Inserción de datos

Esta es una sentencia que nos permitirá insertar nuevos registros en una tabla de nuestra base de datos. Esta será una de las opciones más usadas. La implementación en nuestra aplicación será de forma muy parecida a los ejemplos ya explicados con anterioridad.

```
//Para insertar datos a la bbdd
INSERT INTO nombre_tabla (nombre_columna, nombre_columna) VALUE
```

Para realizar un nuevo registro también tenemos que crear un objeto **Statement** que nos servirá para llamar al método **createStatement()**. La sintaxis del **INSERT** la crearemos mediante un string, tal y como veníamos haciendo hasta ahora, y se lo pasaremos al método **executeUpdate()**.

```
Connection conn = null;
PreparedStatement stmt = null;
try {
    //Utiliza la clase auxiliar que hemos creado para establece
    conn = conector.conector();
    System.out.println("Nos hemos conectado a la BBDD");
    String sql = "INSERT INTO Estudiante (id, dni, nombre, apel
    stmt.executeUpdate(sql);
}catch(SQLException se){
    se.printStackTrace();
}catch(Exception e){
    e.printStackTrace();
}finally{
    try{
        if(stmt!=null)
            stmt.close();
            conn.close();
    }catch(SQLException se){
        System.out.println("No se ha podido cerrar la conexión.
    }
}
```

Existe la posibilidad de realizar sentencias SQL dinámicas. La estructura será muy parecida a lo que estábamos usando hasta ahora, pero en lugar del objeto **Statement** usaremos el objeto PreparedStatement. Este tipo de objeto está especialmente diseñado para poder realizar operaciones con sentencias dinámicas.

```
Connection conn = null;
PreparedStatement stmt = null;
```

```
    try {
        //Utiliza la clase auxiliar que hemos creado para establece
        conn = conector.conector();
        System.out.println("Nos hemos conectado a la BBDD");
        String sql = "INSERT INTO Estudiantes (id, dni, nombre, ape
        //Imaginemos que viene con datos
        Estudiante estudiante = new Estudiante();
        //Prepararemos la query para que coja los datos de manera c
        stmt = conn.prepareStatement(sql);
        stmt.setInt(1, estudiante.getId());
        stmt.setString(2, estudiante.getDni());
        stmt.setString(3, estudiante.getNombre());
        stmt.setString(4, estudiante.getApellido());
        stmt.setInt(5, estudiante.getEdad());
        stmt.executeUpdate(sql);
    }catch(SQLException se){
        se.printStackTrace();
    }catch(Exception e){
        e.printStackTrace();
    }finally{
        try{
            if(stmt!=null)
                stmt.close();
            conn.close();
        }catch(SQLException se){
                System.out.println("No se ha podido cerrar la conex
        }
    }
```

Actualización de datos

La sentencia UPDATE es conocida por permitir actualizar valores de una tabla concreta de la base de datos. Recordamos cómo se realiza un UPDATE:

```
    //Para actualizar datos a la bbdd
    UPDATE nombre_tabla SET nombre_columna =valor, nombre_columna=v
```

Este tipo de sentencia se puede ejecutar del mismo modo que en el ejemplo del INSERT. Tenemos dos posibilidades: con una sentencia ya preescrita en nuestro código o con datos dinámicos. Así se implementaría:

```
Connection conn = null;
Statement stmt = null;
try {
    // Paso 1: Realizamos la conexión
    //Paso 2. Crear objeto y llamar a la conexión
    conn = conector.conector();
    System.out.println("Nos hemos conectado a la BBDD");
    //Paso 3. Crear estructura de la sentencia
    String sql = "UPDATE estudiante SET dni = '00000000T' WHERE
    //Paso 4. Ejecución
    stmt = conn.createStatement();
}catch(SQLException se){
    //Gestionamos los posibles errores que puedan surgir durant
    se.printStackTrace();
}catch(Exception e){
    //Gestionamos los posibles errores
    e.printStackTrace();
}finally{
    //Paso 5. Cerrar objetos abiertos
    try{
        if(stmt!=null)
            stmt.close();
        conn.close();
    }catch(SQLException se){
        System.out.println("No se ha podido cerrar la conexión.
    }
}
```

Para un **UPDATE** con sentencia fija, usaremos el objeto **Statement**, ya que es una operación sencilla. El procedimiento es exactamente igual que con otras sentencias: deberemos meter todo el código entre el try-catch para poder controlar los errores que puedan surgir. Para un **UPDATE** con datos dinámicos

usaremos un objeto **PreparedStatement**, que nos va a permitir añadir dinámicamente los valores que nos interesen a nuestra sentencia.