Estándar de Consultas a la Base de Datos

**Estándar Métodos**

Dentro del proyecto de Java contemplaremos los siguientes paquetes para la manipulación de datos:

**1. AdministracionBD:** En este paquete se almacenarán los distintos métodos de cada clase descritos en el archivo “Diagrama de clases”, como una clase. La implementación de estas clases nos permitirá implementar las distintas consultas SQL para el manejo de datos.

* **Nombre:** El nombre de cada clase corresponderá al nombre mismo del método existente (pero con mayúscula al inicio) con las iniciales “BD” al termino, es decir de la siguiente forma: <nombre\_método>BD.java.

**Ejemplo:** Dentro de la clase *MantenerUsuario* tenemos el método *addUsuario*, por lo tanto la nueva clase que crearemos dentro del paquete *Administración* será *AddUsuarioBD.java.*

* **Librerías:** Las librerías que importaremos para cada clase corresponderán a las clases utilizadas desde el paquete: TransferObjects.

**Ejemplo:** Para la clase *AddUsuarioBD.java* importaremos *TransferObjects.Usuario*.

* **Clase:** Lo primero que haremos dentro de nuestra clase, será crear la función que nos permita conectarnos para la obtención de datos, esta clase tendrá el mismo nombre de la clase.

**Ejemplo:**

public AddUsuarioBD(Connection connection)

{

this.connection = connection;

}

Luego, agregaremos la función encargada de realizar las acciones correspondientes al manejo de datos solicitado, esta llevara el nombre del método de la clase

Ejemplo: *addUsuario.*

* **Función:** Finalmente, dentro de la función correspondiente al método de la clase, declararemos la variable utilizada para la consulta llamada “query”, esta variable nos recibirá el resultado arrojado por la base de datos, para posteriormente ser utilizada en el manejo de los datos según sea la acción que se desee realizar.

**Ejemplo:**

query = "INSERT INTO Usuario (nombre,aPaterno,aMaterno,usuario,cargo,contrasena,servicio,pRegistrar,pEditar,pEliminar,pPurgar) "+

"VALUES (?, ?, ?, ?,?,? ,?,?,? ,?,?);";

insert = connection.prepareStatement(query);

**2. Administración:** Análogamente al paquete *AdministracionBD;* en este paquete se almacenarán los distintos métodos de cada clase descritos en el archivo “Diagrama de clases”, como una clase. La implementación de estas clases nos permitirá instanciarlas posteriormente al momento de realizar las acciones correspondientes al manejo de datos.

* **Nombre:** El nombre de cada clase corresponderá al nombre mismo del método existente (pero con mayúscula al inicio), es decir de la siguiente forma: <nombre\_método>.java.

**Ejemplo:** Dentro de la clase *MantenerUsuario* tenemos el método *addUsuario*, por lo tanto la nueva clase que crearemos dentro del paquete *Administración* será *AddUsuario.java.*

* **Librerías:** Las librerías que importaremos para cada clase corresponderán a las clases utilizadas desde los paquetes: TransferObjects y AdministracionBD.

**Ejemplo:** Para la clase *AddUsuario.java* importaremos *TransferObjects.Usuario* y *AdministracionBD.AddUsuarioBD*.

* **Clase:** Dentro de cada clase crearemos una función *public stactic* con su tipo correspondiente; el nombre será el mismo correspondiente al método de la clase.

**Ejemplo:**

public class AddUsuario {

public static String addUsuario(Usuario u) throws SQLException{

}

}

* **Función:** A su vez, dentro de la función insertaremos la conexión que realizaremos con la base de datos, correspondiente a la clase que estamos utilizando para poder acceder a los datos.

**Ejemplo:**

Connection connection=DBConnectionManager.getConnection();

AddUsuarioBD aubd = new AddUsuarioBD(connection);

return aubd.addUsuario(u);

Continuando con el manejo de datos al interior de una función, basándonos en el archivo “Diagrama de clases”, esta puede realizar las siguientes acciones: add, purg, set, del. Para donde cada uno de ellos se define de la siguiente forma.

* **Add:** Este tipo de método será representado por la variable “insert” al interior de la función. Una vez declarada por medio de *PreparedStatement,* hacemos uso de ella.

Ejemplo:

PreparedStatement insert;

Ejemplo función:

insert.setString(1, u.nombre);

insert.setString(2, u.apellidoMaterno);

insert.setString(3, u.apellidoMaterno);

insert.setString(4, u.usuario);

insert.setString(5, u.cargo);

insert.setString(6, u.contrasena);

insert.setString(7, u.servicio);

insert.setBoolean(8, u.permisoRegistrar);

insert.setBoolean(9, u.permisoEditar);

insert.setBoolean(10, u.permisoEliminar);

insert.setBoolean(11, u.permisoPurgar);

* **Get:** Este tipo de métodos será representado por la variable “select” con el nombre correspondiente al dato al final, select<dato>.

Ejemplo:

PreparedStatement selectRut;

A su vez, para la obtención de estos datos crearemos una función que nos retorne una lista con la información, para luego ser utilizada. El nombre de esta función será getAll<dato>.

**Ejemplo:**

public List<Login> getAllPersons()

**Ejemplo función:**

public List<Login> getAllPersons()

{

List<Login> persons=new ArrayList<Login>();

Login person;

try

{

ResultSet result = selectAll.executeQuery();

while(result.next())

{

person= new Login();

person.setRut(result.getString(1));

person.setName(result.getString(2));

persons.add(person);

}

}

catch (SQLException e)

{

e.printStackTrace();

}

return persons;

}

* **Set:** Este tipo de métodos será representado por la variable “set” con el nombre correspondiente al dato al final, set<dato>.

**Ejemplo:**

PreparedStatement setRut;

Cabe mencionar que para “Set” debemos obtener los datos nuevamente para luego modificarlos, por lo tanto al igual que “Get” haremos el llamado a una función que nos retorne una lista con todos los datos solicitados.

* **Del:** Este tipo de métodos será representado por la variable “delet” con el nombre correspondiente al dato al final, delet<dato>.

**Ejemplo:**

PreparedStatmente deletRut;

* **Purg:** Este tipo de métodos será representado por la variable “purgar” con el nombre correspondiente al dato al final, purgar<dato>.

**Ejemplo variable:**

PreparedStatmente purgarRut;

**Ejemplo función:**

public int deleteMascota(int id){

int result=0;

try{

delete.setInt(1, id);

result = delete.executeUpdate();

}

catch (SQLException e){

e.printStackTrace();

}

return result;

}

**Estándar SQL**

## 1. Select

La recuperación de los datos en el lenguaje SQL se realiza mediante la sentencia SELECT, seleccionar. Esta sentencia permite indicar al SGBD la información que se quiere recuperar. Esta es la sentencia SQL, con diferencia, más habitual. La sentencia SELECT consta de cuatro partes básicas:

* La cláusula SELECT seguida de la descripción de lo que se desea ver, los nombres de las columnas a seleccionar. Esta parte es obligatoria.
  + Las columnas a seleccionar se enumeran sin más en la cláusula SELECT. Si se desea seleccionar todas las columnas de una tabla se puede hacer enumerando a todas las columnas o colocando un asterisco, \*, en su lugar.
* La cláusula FROM seguida de la especificación de las tablas de las que se han de obtener los datos. Esta parte es obligatoria.
  + La cláusula FROM define las tablas de las que se van a seleccionar las columnas.
* La cláusula WHERE seguida por un criterio de selección, una condición. Esta parte es opcional.
  + Es aquí donde debemos proponer la condición que han de cumplir todas las filas para salir en el resultado de la consulta
* La cláusula ORDER BY seguida por el criterio de ordenación. Esta parte es opcional.
  + Se utiliza para especificar el criterio de ordenación de la respuesta a la consulta. Por defecto la ordenación es ascendente, aunque se puede especificar un orden descendente

Una primera aproximación a la sintaxis de la sentencia SELECT puede mostrarnos la siguiente expresión:

SELECT {\* | {columna,}+}

FROM {tabla,}+

[WHERE condición]

[ORDER BY {expresiónColumna [ASC | DESC],}+];

**Ejemplo**

public ArrayList<String> getCita(int id,int numColumnas ) throws SQLException {

SQL = "SELECT \* FROM Cita WHERE Cita.idCita = '"+id+"';";

return Mysql.selectSQL(conn, SQL, numColumnas);

}

## 2. Actualizar

Otra de las operaciones más comunes es la modificación de la información almacenada en las tablas. Para ello se utiliza el comando UPDATE cuya sintaxis se muestra a continuación.

UPDATE tabla SET {columna = expresión,}+ [WHERE condición];

Se especificará en la cláusula SET las columnas que se actualizarán y con qué valores. La cláusula WHERE indica las filas con las que se va a trabajar. Si se omite la actualización afectará a todas las filas de la tabla.

**Ejemplo**

public void setServicio(Servicio s) throws SQLException {

SQL = "UPDATE Servicio SET Servicio.descripcion = '"+s.getDescripcion()+"', Servicio.Admistrador\_idAdmistrador='"+s.getAdministrador()+"';";

Mysql.insertSQL(conn, SQL);

}

## 3. Borrar

Con insertar y modificar, la otra operación que completa el trio es la de borrado de filas. La sintaxis es la que sigue:

DELETE FROM tabla [WHERE condición];

Borrará todas las filas que cumplan la condición especificada en la cláusula WHERE. Si esta cláusula no se fija, se borrarán todas las filas de la tabla. Aquí cabe decir que aunque con DELETE borremos todas las filas de una tabla, no borramos la definición de la tabla del diccionario y podemos insertar datos posteriormente en la tabla. Esta es una diferencia con la sentencia DROP TABLE, que produce la eliminación tanto del contenido de la tabla como de la definición de la misma.

**Ejemplo**

public void delServicio(int id) throws SQLException {

SQL = "DELETE FROM Servicio WHERE Servicio.idServicio = '"+id+"';";

Mysql.insertSQL(conn, SQL);

}

## 4. Insertar

El comando que permite insertar filas en las tablas es el siguiente.

INSERT INTO tabla [({columna,}\*)] VALUES ({expresión,}+);

Sólo especificaremos las columnas donde insertar y su orden cuando no insertemos datos en todas ellas o no lo hagamos en el mismo orden en que definimos la tabla. La asociación columna-valor es posicional. Los valores deben cumplir con los tipos de datos definidos. Los valores de tipo caracter y fecha deben ir encerrados entre comillas simples, ('').

**Ejemplo**

public void addServicio(Servicio s) throws SQLException {

SQL = "INSERT INTO Servicio VALUES ('"+s.getIdServicio()+"','"+ s.getNombre()+"','"+s.getCosto() +"','"+ s.getIdAdministrador()+"','"+s.getDescripcion()+");";

Mysql.insertSQL(conn, SQL);

}