**Estándar SQL**

**Nombre de las Clases**

El nombre de las clases será similar al estándar de Java, pero con la sutil diferencia de que este tendrá que tener en su término las iniciales “BD”. Es importante no olvidar además, que una vez creada la clase en java, la clase utilizada para SQL tendrá que poseer el mismo nombre (adjunto “BD” al termino).

<Nombre clase>BD

Ejemplo:

Class Ciente; //nombre de clase utilizada en capa 2

Class ClienteBD; //nombre de clase utilizada en capa 3

**Observación**

Cabe señalar que para el resto de la implementación interna de la clase: métodos, variables, prefijos en variables globales, asignaciones y referencias a variables. Se seguirá el estándar propuesto de Java.

**Dentro de las clases**

Lo primero que debemos hacer dentro de nuestra clase es importar la biblioteca correspondiente a SQL, que nos permitirá realizar la conexión con java.

**import** java.sql.Connection;

Luego, declaramos las variables utilizadas en la clase utilizada para la base de datos, de la siguiente forma.

PreparedStatement <nombre variable>

Ejemplo:

**public** **class** ClienteBD

{

PreparedStatement insert

PreparedStatemen selectAll

A su vez tendremos nuestro constructor, el cual tendrá como parámetro la conexión. A continuación declararemos nuestra variable del tipo String llamada “query”, en la que almacenaremos la respuesta a la consulta realizada, quedando de la siguiente forma.

**public** ClienteDB(Connection connection)

{

**try**

{

String query="";

query = "INSERT INTO persons(rut, name, lastname) "+

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

insert = connection.prepareStatement(query);

query = "SELECT rut, name, lastname " +

"FROM persons;";

selectAll = connection.prepareStatement(query);

}

**catch** (SQLException e)

{

e.printStackTrace();

}

}

Finalmente para insertar nuevos datos, a través de un función que nos lo permita (considerando con los parámetros existentes).

**public** **int** insertCliente(Cliente cliente)

{

**int** result=0;

**try**

{

insert.setString(1, cliente.getRut());

insert.setString(2, cliente.getName());

insert.setString(3, cliente.getLastName());

result= insert.executeUpdate();

}

**catch** (SQLException e)

{

e.printStackTrace();

}

**return** result;

}

O la selección de algún dato desde una lista

A continuación se especifican las distintas acciones en SQL para la manipulación de datos.

**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);

}

**Conexión**

Lo primero que debemos hacer dentro de nuestra clase es importar la biblioteca correspondiente a SQL, que nos permitirá realizar la conexión con java

**import** java.sql.Connection;

Luego, declaramos las variables utilizadas en la clase utilizada para la base de datos, de la siguiente forma

<Nombre clase>BD

Ejemplo:

**public** **class** <nombre clase

{

PreparedStatement <nombre variable>

tendremos una variable del tipo String, en la que almacenaremos la modificación solicitada a realizar en nuestra base de datos.

private String SQL="";

Para finalmente hacer la inserción de la modificación, con los parámetros anteriores.

Mysql.insertSQL(conn, SQL);

O la selección de algún dato desde una lista

public ArrayList<String> getServicios(int numColumnas)throws SQLException{

SQL = "SELECT \* FROM Servicio";

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

}

A continuación se especifican las distintas acciones en SQL para la manipulación de datos.