Conexión a bases de datos

Parte tres

```
public boolean consultarCliente(Connection conn, String rfc){
		String query = "select nombre,apellido1, apellido2";
		query += " from scestacionamiento.cliente ";
		query += " natural join scestacionamiento.clientepfisica ";
		query += " where rfccliente =  ?  or  rfccliente =.  ?  ";
		PreparedStatement stmt = null;
		try {
				stmt = conn.prepareStatement(query);
				stmt.setString(1,rfc);
				stmt.setString(2,"HERL960327MOC");
				ResultSet rs = stmt.executeQuery();
				ResultSetMetaData rsmd = rs.getMetaData();
				int columnCount = rsmd.getColumnCount();
				// The column count starts from 1
				for (int i = 1; i <= columnCount; i++ ) {
						String name = rsmd.getColumnName(i);
						System.out.print(name + "\t");
				}
				System.out.print("\n");
				while (rs.next()) {
						for (int i = 1; i <= columnCount; i++) {
								System.out.print(rs.getString(i) +
"\t");
						}
						System.out.print("\n");
				}
				rs.close();
				stmt.close();
				// c.close();
		} catch ( Exception e ) {
				System.err.println( e.getClass().getName()+": "+
e.getMessage() );
				System.exit(0);
		}
		System.out.println("Operation ConsultarCliente done successfully");
		return true;
}
```

Para realizar una consulta en SQL se ejeuta a través del método executeQuery() de la clase PreparedStatement.
Para crear la instancia de PreparedStatement, lo hacemos a través de la instacia de nuestra conexión con el método perareStatement, este método recibe como parámetro un String con la cadena del código SQL con la consulta y regresa una instancia de tipo PreparedStatement.

Observa que en la cadena de la consulta se incluyen símbolos ?, esto significa que en ese espacio se agregarán variables, las cuales por el orden en que fueron agregados se pueden identificar por números de 1 en adelante
Los valores de esas variables se agregan a través de los método setString, setInt, setFloat, setDate, etc (depende del tipo de dato del valor a insertar) de la instancia de PreparedStatement, estos métodos reciben 2 parámetros, el primero es el número consecutivo que le corresponde al signo ? Por su posición y el segundo es el valor de la variable.

El método executeQuery() regresa una instancia ResultSet, con el conjunto de tuplas resultantes de la consulta SQL que ha sido ejecutada.
Se puede obtener la cantidad de columnas del resultado y su nombre a través del una instancia de tipo ResultSetMetaData, valor que retorna el método getMetaData() de la instancia de tipo ResultSet.

```
public boolean consultarCliente(Connection conn, String rfc){
		String query = "select nombre,apellido1, apellido2";
		query += " from scestacionamiento.cliente ";
		query += " natural join scestacionamiento.clientepfisica ";
		query += " where rfccliente =   ?    or    rfccliente =.  ?    ";
		PreparedStatement stmt = null;
		try {
				stmt = conn.prepareStatement(query);
				stmt.setString(1,rfc);
				stmt.setString(2,"HERL960327MOC");
				ResultSet rs = stmt.executeQuery();
				ResultSetMetaData rsmd = rs.getMetaData();
				int columnCount = rsmd.getColumnCount();
				// The column count starts from 1
				for (int i = 1; i <= columnCount; i++ ) {
						String name = rsmd.getColumnName(i);
						System.out.print(name + "\t");
				}
				System.out.print("\n");
				while (rs.next()) {
						for (int i = 1; i <= columnCount; i++) {
								System.out.print(rs.getString(i) +
"\t");
						}
						System.out.print("\n");
				}
				rs.close();
				stmt.close();
				 //c.close();
		} catch ( Exception e ) {
				System.err.println( e.getClass().getName()+": "+
e.getMessage() );
				System.exit(0);
		}
		System.out.println("Operation ConsultarCliente done successfully");
		return true;
}
```

Con la instancia de ResultSetMetaData, invocando su método getColumnCount() podemos saber la cantidad de columnas de resultado de la consulta SQL y con getColumnName(i) el nombre de la columna con el índice i. La primera columna del resultado tiene la posición 1.

Las tupla (registros o filas) de ResultSet, las podemos recorrer a través del método next(), si aún hay filas disponibles retorna verdadero, en otro caso, si la consulta está vacía o se llegó al final, regresará falso.

Dentro de cada tupla se puede recorrer cada columna a través del número de columna y los métodos getString(i), getInt(i), getDate(i), etc., dependiendo del tipo de dato que se va a leer.

Una vez que se obtuvo la consulta y se leyó el resultado y antes de salir es necesario cerrar la instancia ResultSet y PreparedStatement a través de su método close()

Si ya la conexión no se va a utilizar, es necesario cerrarla temibén.

```
public boolean insertarClientepfisica(Connection conn, Clientepfisica c){
    String sqlInsertCliente = "insert into scestacionamiento.cliente (rfccliente, correoe,
fecharegistro, tipocliente) values (?, ?, ?, 'f')";
    PreparedStatement stmt = null;
    int tuplas = 0;
    java.sql.Date sqlDate = new java.sql.Date(c.getfecharegistro().getTime());
    try {
                stmt = conn.prepareStatement(sqlInsertCliente);
                stmt.setString(1,c.getrfccliente());
                stmt.setString(2,c.getcorreoe());
                stmt.setDate(3,sqlDate);
                tuplas = stmt.executeUpdate();
   } catch ( Exception e ) {
                System.err.println( e.getClass().getName()+": "+ e.getMessage() );
                System.exit(0);
    } System.out.println("Operation done successfully" + tuplas);
     return true;
}
```

Cuando insertamos una nueva tupla (insert), actualizamos datos (update) o eliminamos tuplas (delete), se cambia la sintaxis.

Se construye la sentencia SQL que se va a ejecutar, observa que al igual que en la consulta, se incluyen ? Para agregar parámetros de la consulta.

También se crea una instancia de tipo PreparedStatement

Ahora en lugar del método executeQuery(), se invoca el método **executeUpdate(),** el cual nos regresa un número entero con la cantidad de tuplas afectadas por la sentencia SQL.

- Pero por qué utilizar ? En lugar de concatenar directamente en la cadena el valor de las variables.
- Es sencillo, esto nos permite evitar que se haga una inyección de código SQL en tiempo de ejecución en nuestro programa.
- Si hay un formulario que tiene por ejemplo un cuadro de texto, nuestro usuario podría escribir ahí completamente una sentencia SQL que pueda eliminar todas nuestras tuplas, al concatenar directamente, podría hacerlo.

- DUDAS?, recuerda entrar al foro creado para eso