Practica 08

Alumnos:
Castañon Maldonado Carlos Emilio
Chávez Zamora Mauro Emiliano
Gallegos Diego Cristian Ricardo
Navarro Santana Pablo César
Nepomuceno Escarcega Arizdelcy Lizbeth

Facultad de Ciencias
UNAM

Reporte - Práctica 08: JDBC

Para la elaboración de esta práctica, decidimos utilizar las tablas, **Cliente** y **CorreoCliente**:

Tabla **Cliente**:

1. ```
   SELECT *
   FROM Cliente
   ```

   Esta query se encarga de recuperar todos los registros de la relación Cliente, el segmento de SELECT se emplea para elegir cuáles atributos queremos recuperar de la relación, el símbolo * sirve como un comodín para no tener que especificar todos y cada uno de los atributos en nuestras tablas, FROM especifica de qué relación/tabla se extraerá esa información. En lenguaje natural su equivalente sería "Recupera/selecciona todos los registros de la relación Cliente".

2. ```
   SELECT * FROM cliente WHERE idPersona=
   ```

   Esta query se encarga de recuperar un registro específico de la tabla cliente en la base de datos, filtrando por el idPersona. El SELECT * elige recuperar todos los atributos de la tabla para el registro que corresponde a la condición especificada en la cláusula WHERE. El parámetro de la condición busca un registro que tenga el idPersona igual al valor proporcionado, permitiendo seleccionar un registro particular.

3. ```
   INSERT INTO Cliente(idPersona, nombre, paterno, materno, genero)
   VALUES (:idPersona,:nombre,:paterno,:materno,:genero)
   ```

   Esta query se encarga de agregar nuevos registros a la relación Cliente, para eso es importante que especifiquemos qué valor vamos a asignar a cada uno de los atributos para tener nuestra instancia de Cliente creada. INSERT INTO avisa a la base de datos que vamos a crear/insertar un nuevo registro de la tabla Cliente, donde los parámetros entre paréntesis "(idPersona, nombre, paterno, materno, genero)" van a tomar los valores asignados por VALUES (:idPersona, :nombre,:paterno,:materno,:genero), en este caso los ':' sirven para que el SqlParameterSource junto con el KeyHolder puedan proporcionar los datos de una instancia en concreto de la clase Cliente en Java.

4. ```
   UPDATE Cliente
   SET idPersona=:idPersona, nombre=:nombre, paterno=:paterno, materno=:materno,
   genero=:genero
   WHERE idPersona=:idPersona
   ```

   Esta query se encarga de actualizar la tabla Cliente, la instrucción UPDATE informa que se va a modificar un registro existente, pero como lo hacemos a partir de una instancia la clase Cliente debemos actualizar los datos completos, por eso SET ajusta los nuevos parámetros (idPersona=:idPersona, nombre=:nombre, paterno=:paterno, materno=:materno, genero=:genero), los dos puntos son para que el código en Java pueda asignarles valores a través de un objeto de la clase Cliente empleando SqlParameterSource y el KeyHolder, acá para asegurarnos que el registro ya existe la instrucción WHERE modificará aquellos registros que correspondan al IdPersona que estamos pasando como parámetro, debido a que es único y la base de datos lo restringe, esto sólo modificará un único registro.

5. ```
   UPDATE Cliente
   SET idPersona=:idPersona, nombre=:nombre, paterno=:paterno, materno=:materno,
   genero=:genero
   WHERE idPersona=:idPersona
   ```

   Esta query funciona de manera práctiamente idéntica a la anterior.

6. ```
   DELETE FROM Cliente
   WHERE idPersona=:idPersona
   ```

   Esta query se encarga de borrar un registro de la tabla Cliente, al ser único el idPersona no nos basta más que ese dato para efectuar el borrado. DELETE notifica sobre el borrado, FROM Cliente especifica que será de la tabla Cliente y la sentencia WHERE busca que el idPersona de la tabla corresponda con el idPersona proporcionado por el código que genera la petición.

Tabla **CorreoCliente**:

1. ```
   SELECT *
   FROM CorreoCliente
   ```

   Esta query se encarga de recuperar todos los registros de la relación CorreoCliente. La cláusula SELECT es utilizada para elegir qué atributos se recuperarán de la relación, y el símbolo * es un comodín que recupera todos los atributos disponibles en la tabla CorreoCliente. La cláusula FROM especifica de qué relación o tabla se extraerá la información, en este caso, CorreoCliente. En lenguaje natural, su equivalente sería "Recupera/selecciona todos los registros de la relación CorreoCliente".

2. ```
   INSERT INTO correoCliente(idPersona, correo)
   VALUES (:idPersona,:correo)
   ```

   Esta query se encarga de agregar un nuevo registro a la tabla correoCliente. La sentencia INSERT INTO especifica que se creará una nueva entrada en la tabla mencionada. Los valores que se asignarán a los atributos idPersona y correo se definen mediante VALUES (:idPersona, :correo), donde :idPersona y :correo son parámetros que posiblemente se completarán más tarde en la ejecución, quizás desde un objeto en Java. Esto resulta en la inserción de un nuevo registro en la tabla correoCliente, con los valores proporcionados para idPersona y correo.

3. ```
   UPDATE CorreoCliente
   SET idPersona=:idPersona, correo=:correo
   WHERE idPersona=:idPersona
   ```

   Esta query tiene como propósito actualizar un registro existente en la tabla CorreoCliente. La sentencia UPDATE señala la intención de modificar datos existentes. En la cláusula SET, se definen los nuevos valores que se asignarán a los atributos especificados: idPersona=:idPersona y correo=:correo. La cláusula WHERE garantiza que la modificación se aplique solo a los registros en los que el valor de idPersona coincida con el proporcionado, asegurando la actualización de un único registro basado en el identificador de persona.

4. ```
   UPDATE Cliente SET idPersona=:idPersona, correo=:correo
   WHERE idPersona=:idPersona
   ```

   Esta query tiene como objetivo actualizar registros en la tabla Cliente. La sentencia UPDATE indica la intención de modificar datos existentes. La cláusula SET define los nuevos valores que se asignarán a los atributos especificados: en este caso, idPersona=:idPersona y correo=:correo. La cláusula WHERE asegura que la modificación se aplique solo a los registros donde el idPersona coincide con el valor proporcionado, lo que garantiza la actualización de un único registro basado en el identificador de persona.

5. ```
   DELETE FROM correoCliente
   WHERE idPersona=:idPersona
   ```

   Esta query tiene la función de eliminar un registro de la tabla correoCliente. La instrucción DELETE FROM indica la intención de borrar registros de la tabla mencionada. La cláusula WHERE se encarga de especificar las condiciones para la eliminación, en este caso, se busca el registro donde el valor de idPersona coincida con el parámetro :idPersona. Esto permite la eliminación de un registro específico de la tabla correoCliente que posea el identificador de persona proporcionado en la consulta.
   ```