Administración y Programación de Bases de Datos
Laboratorio JDBC

Ingeniería Civil Informática / DCCTI / **Universidad del Bío-Bío**

Profesor: Gilberto Gutiérrez R.

Otoño 2019

1 Prepared Statement

Cada vez que se ejecuta una sentencia del tipo *statement* esta es compilada en el servidor. En cambio una sentencia del tipo *preparedStatement* se mantiene precompilada en el servidor, lo que significa que cuando se ejecuta no es necesario volver a compilarla. Como resultado, este tipo de sentecias permite ahorrar tiempo de ejecución, cuando tenemos que ejecutar repetidamente una sentencia SQL, cambiando solamente algunos parámetros. Por ejemplo, un programa necesita realizar varias inserciones en una tabla.

El ejemplo siguiente muestra como se crea un objeto del tipo *preparedStatement*. Suponemos que una aplicación repetidamente actualiza salarios de empleados y que le son proporcionados el nuevo monto y el ID (rut) del empleado.

```
PreparedStatement ps = con.prepareStatement("update empleado set salario = ? where ID = ?");
 ps.setDouble(1, 153833.00);
 ps.setInt(2, 110592);
 ps.executeUpdate();
```

En una sentencia *preparedStatement* los parámetros (o valores a proporcionar) se indican con el caracter `?` y se enumeran en el orden de aparición de izquierda a derecha. En el ejemplo de arriba el primer valor que hay que suministrar es el salario y el segundo el ID del empleado lo que se indica mediante los métodos `ps.setDouble(1, 153833.00)` y `ps.setInt(2, 110592)` respectivamente. Una vez establecidos los parámetros en la sentencia, ésta se ejecuta con `ps.executeUpdate()`.

1. Escriba un programa java (que use *preparedStatement*) insertar amigos a la tabla `amigos`. El programa debe ir leyendo desde la consola los valores de los atributos tal como se muestra en el ejemplo e ir insertándolos en la tabla `amigos`.

   ```
   ---- datos del amigo -----
   ID (-1 --> termina) :33
   Nombre :Raul
   Celular : 86867933
   ```

```
Fecha Nacimiento (aaaa-mm-dd) : 1975-2-1
Sexo (M/F): M
---- datos del amigo -----
ID (-1 --> termina) :-1
Terminado ...
[ggutierr@oracle ~]$
```

Para leer desde la consola puede usar la clase `Scanner`. Aqui un ejemplo de un programa para leer de la consola con `Scanner`.

```
import java.util.Scanner;
class leeConsola {
public static void main (String args [])
 {
   Scanner in = new Scanner(System.in);
   System.out.print("ID :");
   int elId = in.nextInt();
   System.out.print("Nombre :");
   String elNombre = in.next();
   System.out.println("ID = " + elId + "  Nombre = " + elNombre);
 }
}
```

Con su programa inserte los siguientes amigos:

```
 2000 Lorena  0896981 1970-10-3 F
 2001 Luciano  0796981 1980-11-3 M
 2000 Beatriz  0895767 1978-9-6 F
 2000 Rene    0896981 1968-10-7 M
 2000 Sergio  0896981 1971-10-12 M
```

2. Por error las coordenadas x de la tabla RECTANGULO (vista en clases clases anteriores) se han insertado amplificadas por 10. Escriba un programa JAVA para corregir este error, es decir, actualizar los atributos XL y XH a $XL/10$ y $XH/10$ respectivamente. La estructura de la tabla RECTANGULO es:

```
Name                                      Null?    Type
----------------------------------------- -------- ----------------------------
RID                                                NUMBER(38)
XL                                                 NUMBER
YL                                                 NUMBER
XH                                                 NUMBER
YH                                                 NUMBER
```

2 Procedimientos almacenados con JDBC

Con JDBC es posible llamar a procedimientos o funciones construidos en PL/SQL. A continuación se presenta un ejemplo muy sencillo. Cree (SQLPLUS) la siguiente función:

```
create or replace function foo (val1 char)
 return char as
 begin
   return val1 || 'suffix';
 end;
```

La función recibe como entrada un string y retorna el string concatenado con el string 'suffix'.

El siguiente trozo de programa JAVA realiza la llamada a la función `foo`.

```
import java.sql.*;
class foo {
public static void main (String args []) throws SQLException
 {
   DriverManager.registerDriver (new oracle.jdbc.driver.OracleDriver());
   Connection conn = DriverManager.getConnection (
     "jdbc:oracle:thin:@oracle.localdomain:1521:orcl", "user", "passwd");

   CallableStatement cs = conn.prepareCall ("begin ? := foo(?); end;");
   cs.registerOutParameter(1,Types.CHAR);
   cs.setString(2, "aa");
   cs.executeUpdate();

   String result = cs.getString(1);
   System.out.println("---> " + result);
 }
} // end clase foo
```

La instrucción `CallableStatement cs = conn.prepareCall ("begin ? := foo(?); end;");` crea un objeto de tipo `CallableStatement` el cual será utilizado para enviar la instrucción con la llamada a la función. Notar que la llamada a la función `foo` va entre los delimitadores `begin` y `end`. El primer símbolo ? indica el valor retornado por la función, mientras que en el segundo símbolo ? se indica el parámetro que es pasado a la función. Las restantes sentencias del programa JAVA son fáciles de comprender.

1. Escriba un programa en JAVA para llamar a la función `Area()`, vista en clases pasada. Si no la recuerda, aqui tiene una copia. Su programa debe obtener todos rectángulos cuya área es mayor a a, con a un parámetro de su programa.

   ```
   create or replace FUNCTION Area(x1 in  number, y1 in number, x2 in number, y2 in number)
     return  number IS
   dx number;
   ```

```
dy number;
BEGIN
 dx := x2 - x1;
 dy := y2 - y1;
 return (dx * dy);
END;
```