



# Práctica 1

## Statement vs. PreparedStatement y Pool de conexiones

### Objetivos

Los objetivos perseguidos por esta práctica son:

- Repasar conceptos ya conocidos de JDBC
- Comparativa Statement/PreparedStatement
- Conexiones a diversos SGBD: Oracle, HSQLdb. Comparativa de tiempos.
- Uso de un Pool de conexiones

### Preparación

- Arrancar el servidor HSQLdb. Para ello, acceder al campus virtual, descargar y descomprimir el archivo "HSQLDB Carworkshop database" y ejecutar data/startup.
- Conectarse al servidor de Oracle. Ejecutar el "Script para la creación de tablas y datos en Oracle".
- Descargar los drivers necesarios para las aplicaciones java y las librerías para el pool de conexiones.
- Crear un directorio RI19-20/Sesion1/workspace en tu HOME
- Dentro de su directorio workspaces, crear un primer espacio de trabajo llamado JDBC que contendrá los proyectos con las soluciones a los siguientes ejercicios. No olvide configurar su build path para enlazar los drivers (para hsqldb y Oracle y, en el tercer ejercicio, añadir las librerías para el pool de conexiones).

### Datos de conexión

- **Oracle**
  - URL = "jdbc:oracle:thin:@156.35.94.99:1521:DESA"; //(puertos:1521 o 5850)
  - USER = "XXXXX"; //Identificador UO
  - PASS = "password";
- **HSQLDB**
  - URL = "jdbc:hsqldb:hsq://localhost";
  - USER = "sa";
  - PASS = "";

### Ejercicios

#### 1. Statement vs. PreparedStatement: comparación de tiempos

Crear un programa java que nos permita comparar el rendimiento de **Statement** frente al de **PreparedStatement**. Para ello se realizará un bucle de **1000** repeticiones con la **misma consulta** de una y



otra manera y se medirán tiempos para compararlos. Para que la diferencia sea significativa debe utilizarse una consulta de cierta complejidad que vaya cambiando en cada iteración del bucle. Como ejemplo, se desarrollará un programa que **cuenta el número de vehículos cuya matrícula contiene el número de iteración**.

En el caso del Statement se construirá la sentencia SQL y se ejecutará (concatenando cadenas, por ejemplo), y en el caso del PreparedStatement se deben utilizar *placeholders* “?”.

Para medir tiempos se puede utilizar `System.currentTimeMillis()` que nos devuelve la fecha/hora en número de milisegundos desde el 01/01/1970

Realizar la comparativa conectándose tanto a Oracle como a HSQLDB.

Se puede ver una discusión muy interesante al respecto de lo trabajado en esta práctica en: [http://asktom.oracle.com/pls/asktom/f?p=100:11:115361867455944:::P11\\_QUESTION\\_ID:1993620575194](http://asktom.oracle.com/pls/asktom/f?p=100:11:115361867455944:::P11_QUESTION_ID:1993620575194)

## 2. Comprobación de costes: apertura y cierre de conexiones

Crear un programa que nos permita ver el coste que tiene el abrir conexiones a la base de datos. Para ello se procederá de dos formas distintas midiendo los tiempos y comparándolos:

1. Se realizará un bucle de 100 repeticiones y dentro del bucle se abrirá una conexión, se realizará una consulta y se cerrará la conexión.
  - a. Repetir 100 veces
    - i. Abrir conexión
    - ii. Hacer algo (ej: contar los vehículos de la tabla *tvehiculos*)
    - iii. Cerrar conexión
2. Similar al anterior, pero la apertura y cierre de la conexión se realizará fuera del bucle.
  - a. Abrir conexión
    - i. Repetir 100 veces
      1. Hacer algo (lo mismo que en el caso anterior)
  - b. Cerrar conexión

## 3. Pool de Conexiones

Para esta práctica utilizaremos el pool de conexiones c3p0 cuyos drivers están en la carpeta lib (descargados de <http://www.mchange.com/projects/c3p0/>).

La práctica consiste en crear un pool de conexiones con las siguientes características:

- Máximo de conexiones (*maxPoolSize*): 30
- Mínimo de conexiones (*minPoolSize*): 3
- Tamaño inicial del pool (*initialPoolSize*): 3

Se debe probar tanto con Oracle como con HSQLDB.



Para comprobar el funcionamiento correcto haremos como en el ejercicio anterior en el que medimos el coste en tiempo de abrir y cerrar conexiones dentro de un bucle. Se debe realizar el mismo bucle pero comparando entre utilizar una conexión del pool o una conexión obtenida sin pool. (Ej hazAlgo: contar el número de vehículos de la tabla tvehiculos)

```
for (int i=0; i<1000; i++){  
    crearConexion();  
    hazAlgo();  
    cerrarConexion();  
}
```

A modo de ejemplo de la utilización de c3p0 se copia aquí un extracto de código disponible en la web de c3p0 ([http://www.mchange.com/projects/c3p0/#using\\_combopooleddatasource](http://www.mchange.com/projects/c3p0/#using_combopooleddatasource)). Con el propio driver viene código de ejemplo. El código siguiente es para iniciar un Pooled DataSource. Una vez creado se podrán obtener Connection.

```
DataSource ds_unpooled =  
DataSources.unpooledDataSource("jdbc:postgresql://localhost/testdb",  
                                "swaldman",  
                                "test-password");  
  
Map overrides = new HashMap();  
overrides.put("maxStatements", "200");  
overrides.put("maxPoolSize", new Integer(50));  
ds_pooled = DataSources.pooledDataSource(ds_unpooled, overrides );  
//ya está inicializado  
//Para obtener conexión  
con = ds_pooled.getConnection();
```

La ejecución de hazAlgo debe realizarse con hilos, de forma concurrente, de forma que se reciban 100 intentos de conexión simultáneos y no secuenciales.

Tomar los tiempos después de crear el pool y al finalizar la ejecución. Compararlos con los tiempos anteriores. Tratar de explicar la diferencia.