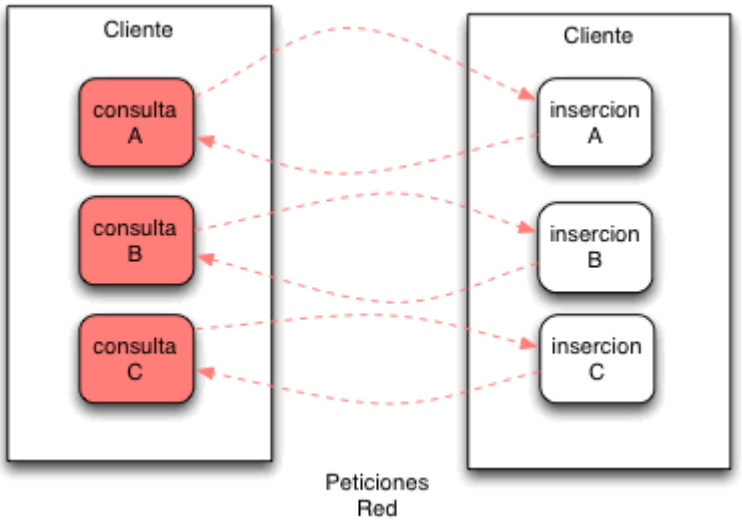
JDBC Batch , o ejecuciones batch es una de las características que muchas veces nos olvidamos que están disponibles en JDBC y pueden mejorar el rendimiento de las actualizaciones que ejecutemos. Normalmente cuando uno trabaja con JDBC o tecnologías similares puede ejecutar varias consultas de inserción seguidas.

CURSO SPRING BOOT
GRATIS
APUNTATE!!

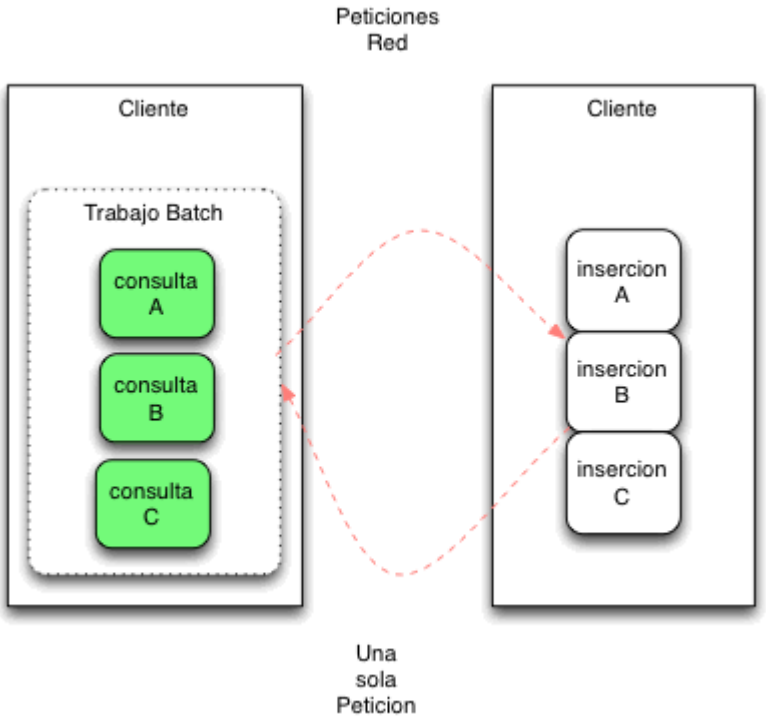
```
sentencia.executeQuery(“insert into persona (nombre,apellidos,edad)
values ('pepe','perez',30)");
sentencia.executeQuery(“insert into persona (nombre,apellidos,edad)
values ('maria','sanchez',20)");
sentencia.executeQuery(“insert into persona (nombre,apellidos,edad)
values ('gema','alvarez',15)");
```

Aunque esto nos parezca correcto, dependiendo de la circunstancia puede no serlo, ya que cada vez que realizamos una llamada al servidor realizaremos una única consulta de inserción y en rendimiento debido al tráfico de red bajará.



JDBC Batch

Para solventar este problema JDBC soporta los trabajos Batch. Estos trabajos agrupan un conjunto de consultas para ejecutarlas todas de forma simultanea.



Vamos a ver el código :

```
package com.arquitecturajava.clase1;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;

public class ProgramaMain {
public static void main(String[] args) {

Connection conexion = null;
Statement sentencia = null;
try {
Class.forName("com.mysql.jdbc.Driver");
String cadenaConexion = "jdbc:mysql://localhost:3306/curso";
conexion = DriverManager.getConnection(cadenaConexion, "root", "");
sentencia = conexion.createStatement();
sentencia.addBatch("insert into persona (nombre,apellidos,edad) values
('pepe','perez',30)");
sentencia.addBatch("insert into persona (nombre,apellidos,edad) values
('maria','sanchez',20)");
sentencia.addBatch("insert into persona (nombre,apellidos,edad) values
('gema','alvarez',15)");
sentencia.executeBatch();

} catch (ClassNotFoundException e) {
// TODO Auto-generated catch block
e.printStackTrace();
} catch (SQLException e) {
```

```
// TODO Auto-generated catch block
e.printStackTrace();

} finally {

if (sentencia != null)
try {
sentencia.close();
} catch (SQLException e) {
e.printStackTrace();
}

if (conexion != null)
try {
conexion.close();
} catch (SQLException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}

}
}
}
```

Como podemos observar el método de addBatch va añadiendo las diferentes consultas al grupo. Por último el método executeBatch las ejecuta de golpe contra la base de datos. Este método devuelve un array de enteros con las filas afectadas por cada consulta.

CURSO SPRING BOOT

GRATIS

APUNTATE!!

Otros artículos relacionados :

Introducción a JPA

EntityManager

Introducción a JTA