

Midiendo Tiempos de Ejecución

Practica 2 de BBDD II

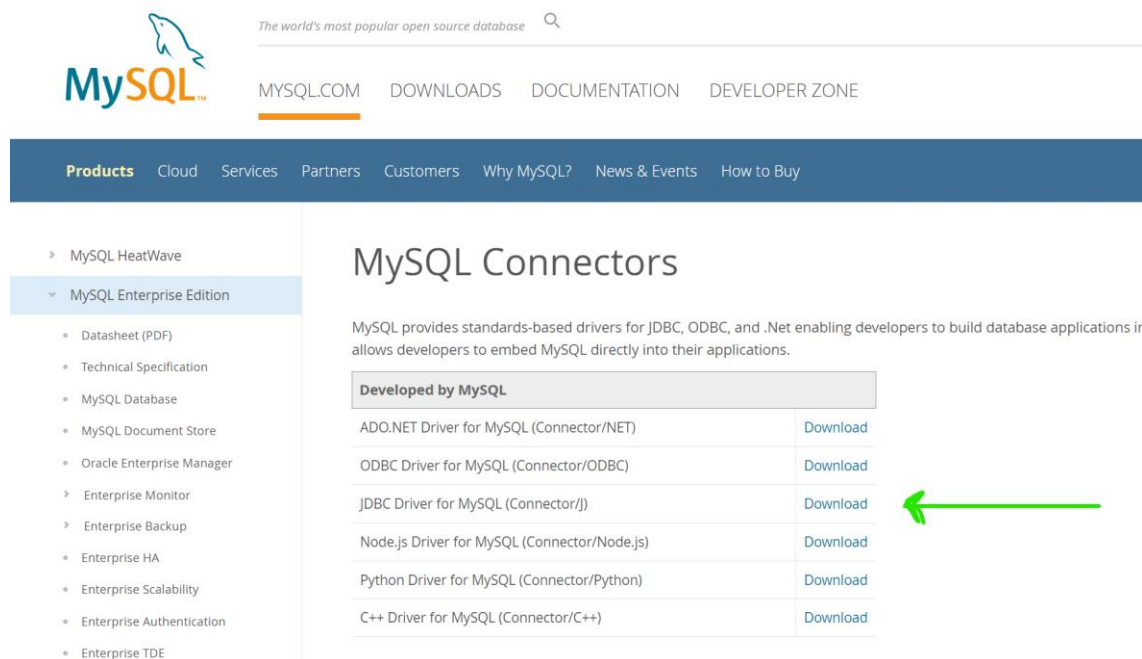
Rodrigo De Jesus Meléndez Molina

Raimundo Baca Mbang

Ignacio Arvilla De Caralt

1. Conexión con MySQL

Lo primero que debemos hacer para conectarse a MySQL con java es descargarnos un archivo .jar JDBC connector, y meterlo en la carpeta del proyecto java.



The screenshot shows the MySQL website. The top navigation bar includes links for MySQL.COM, DOWNLOADS, DOCUMENTATION, and DEVELOPER ZONE. The main content area is titled 'MySQL Connectors' and describes the standard-based drivers for JDBC, ODBC, and .Net. A table lists the available connectors, with a green arrow pointing to the 'Download' link for the JDBC Driver for MySQL (Connector/J).

Developed by MySQL	
ADO.NET Driver for MySQL (Connector/.NET)	Download
ODBC Driver for MySQL (Connector/ODBC)	Download
JDBC Driver for MySQL (Connector/J)	Download
Node.js Driver for MySQL (Connector/Node.js)	Download
Python Driver for MySQL (Connector/Python)	Download
C++ Driver for MySQL (Connector/C++)	Download

Hecho esto, desde el código guardamos en variables, la dirección de nuestro servicio MySQL en localhost, nuestro usuario que por defecto es el root, nuestra contraseña, y la dirección del driver descargado anteriormente.

```
static String url = "jdbc:mysql://localhost:3306/";
static String user = "root";
static String passwd = "contraseña";
static String driver = "com.mysql.cj.jdbc.Driver";
static Connection cx;

public static Connection conectar(String bd) {
    try{
        Class.forName(className: driver);
        cx = DriverManager.getConnection(url+bd, user, password:passwd);
        System.out.println("Se ha conectado a la Base de Datos " + bd);
    }catch(ClassNotFoundException | SQLException ex) {
        System.out.println("No se pudo conectar a la Base de Datos " + bd);
    }
    return cx;
}
```

2. Consultas de selección y medir tiempos de respuesta del servidor

En el código fuente, cada consulta realizada está hecha en un método propio llamado consulta1(), consulta2(), ... , consulta12() ya que hay 12 consultas al pedirnos 2 consultas por cada base de datos en cada apartado.

```
public static void consulta4(Connection cn){
    long startTime = System.currentTimeMillis();

    String consulta = "select * from city where CountryCode = 'ESP'";
    Statement stmt;
    ResultSet rs;

    int id;
    String nombre;
    String pais;
    String distrito;
    int poblacion;

    try{
        stmt = cn.createStatement();
        rs = stmt.executeQuery(consulta);

        while(rs.next()){
            id = rs.getInt("ID");
            nombre = rs.getString("Name");
            pais = rs.getString("CountryCode");
            distrito = rs.getString("District");
            poblacion = rs.getInt("Population");

            System.out.println(id + " " + nombre + " | " + pais + " " + distrito + " | " + poblacion);
        }

        long endTime = System.currentTimeMillis();
        long totalTime = endTime - startTime;

        System.out.println("Tiempo de respuesta del servidor: " + totalTime + " milisegundos");

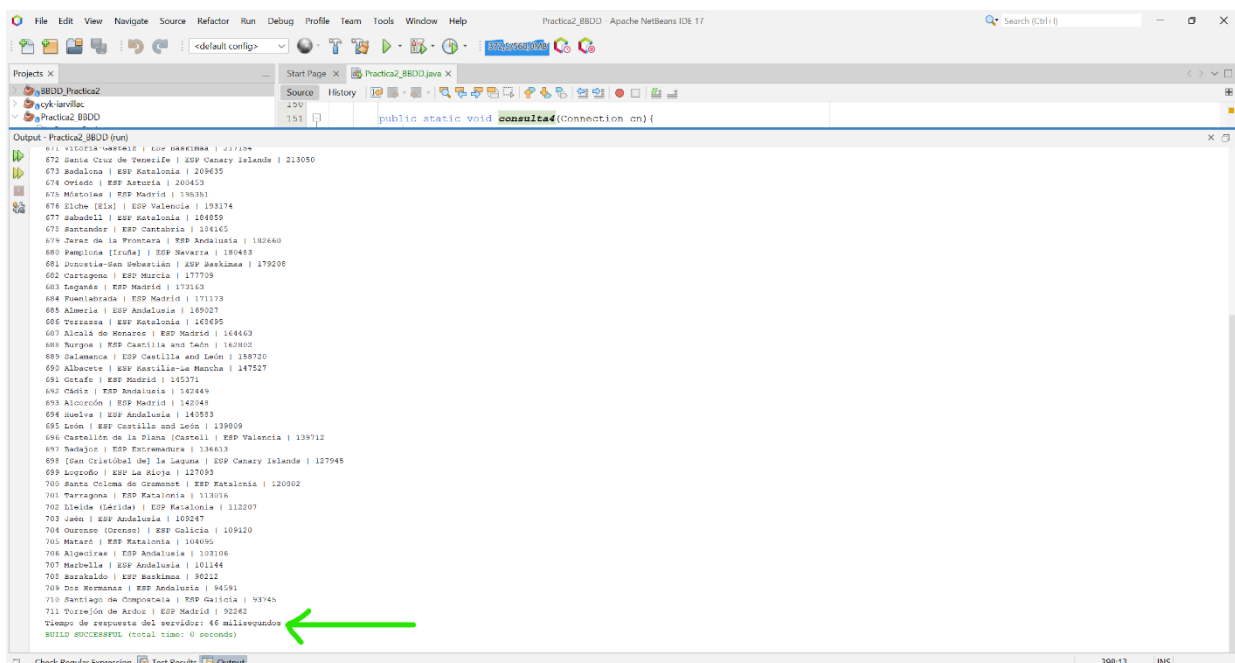
    }catch(Exception e){
        System.out.println(e, "No se ha podido realizar la consulta");
    }
}
```

En cuanto al código necesario para realizar la consulta, creamos la variable stmt la cual será una sentencia SQL, y otra variable rs que será el resultado de cada registro de devuelva esa sentencia.

Antes de hacer el código probamos la consulta pensada antes para comprobar las columnas que devuelve, ya que deberemos crear variables por cada columna para poder imprimir los resultados después. Entonces, igualamos cada variable a cada columna e imprimimos por pantalla.

En cuanto al código para medir el tiempo de respuesta, creamos dos variables antes y después del código de la consulta, y cada una de estas la igualamos a System.currentTimeMillis(), que devuelve el tiempo actual en milisegundos.

Tras esto, restamos el tiempo al final y al principio y da como resultado el tiempo de respuesta.



```
Output - Practica2_BBDD (run)
671 Vitoria-Gasteiz | ESP Euzkadi | 211154
672 Santa Cruz de Tenerife | ESP Canary Islands | 213050
673 Badajoz | ESP Extremadura | 209455
674 Oviedo | ESP Asturias | 200453
675 Móstoles | ESP Madrid | 190381
676 Elche (Elis) | ESP Valencia | 193174
677 Sabadell | ESP Catalonia | 164059
678 Santander | ESP Cantabria | 104165
679 Jerez de la Frontera | ESP Andalusia | 182660
680 Pamplona (Iruña) | ESP Navarre | 180483
681 Donostia-San Sebastián | ESP Basque | 179208
682 Cartagena | ESP Murcia | 177709
683 Sagunto | ESP Madrid | 172563
684 Puenlabrada | ESP Madrid | 171173
685 Almería | ESP Andalusia | 169027
686 Terresana | ESP Catalonia | 167695
687 Alcalá de Henares | ESP Madrid | 164463
688 Nájera | ESP Castile and León | 162802
689 Salamanca | ESP Castile and León | 158720
690 Alavete | ESP Castile-La Mancha | 147527
691 Getafe | ESP Madrid | 145371
692 Cádiz | ESP Andalusia | 142449
693 Alcorcón | ESP Madrid | 142048
694 Jueve | ESP Andalusia | 140553
695 León | ESP Castile and León | 139009
696 Castellón de la Plana (Castell) | ESP Valencia | 138712
697 Badajoz | ESP Extremadura | 136633
698 [San Cristóbal de] La Laguna | ESP Canary Islands | 127945
699 Logroño | ESP La Rioja | 127093
700 Santa Coloma de Gramenet | ESP Catalonia | 120302
701 Tarragona | ESP Catalonia | 113016
702 Lleida (Sakide) | ESP Catalonia | 112207
703 Jawn | ESP Andalusia | 109247
704 Ourense (Orense) | ESP Galicia | 109120
705 Huesca | ESP Catalonia | 104000
706 Almería | ESP Andalusia | 103108
707 Melilla | ESP Andalusia | 101144
708 Badajoz | ESP Extremadura | 96212
709 Dos Hermanas | ESP Andalusia | 94591
710 Santiago de Compostela | ESP Galicia | 93745
711 Torrejón de Ardoz | ESP Madrid | 92262
Tiempo de respuesta del servidor: 66 milisegundos
BUILD SUCCESSFUL (total time: 0 seconds)
```

3. Análisis de los resultados al medir tiempos

La principal diferencia entre los distintos tiempos de consulta que vemos depende de la complejidad de la consulta que se hace. Por esto las consultas que pide el apartado c tienen un tiempo de respuesta más largo, concretamente de unos 1200 milisegundos de media. En cambio, en las demás consultas ninguna se acerca a este número.

También, al hacer consultas que devuelven muchos registros, tardan más que otras consultas con menos registros.

Al comparar ambas bases de datos, sakila y world. Aunque la diferencia sea mínima, la base de datos sakila tiene en media tiempos de respuesta mas altos que la base de datos world. Esto se debe principalmente a que la base de datos sakila tiene un tamaño mucho

más grande en comparación a world ya que tiene un mayor número de tablas y de relaciones.