

**PRACTICA III**  
**BASES DE DATOS**  
**MEMORIA**



Tomás Machín, Paloma Pérez, Juan García

# **INDICE**

## **1.- Introducción.**

## **2.- Consultas.**

- Consultas predefinidas.
- Consultas simples.
- Consultas complejas.

## **3.- Aplicación / Interfaz**

- Métodos usados.
- Interfaz.
- Uso.

## **4.- Interpretación de resultados.**

## **5.- Conclusión.**

## **1.- INTRODUCCIÓN**

El objetivo de la practica es: **acceder a una base de datos *MySQL* desde una aplicación *Java* desarrollada a tal efecto. Dicha aplicación deberá medir tiempos de respuesta del sistema, con el fin de poder realizar un análisis comparativo de los mismos.**

Para poder abordar la práctica y obtener una solución a esta, nos hemos decantado por usar la aplicación de *NetBeans* para el desarrollo dela aplicación Java requerida para esta, además del uso obligatorio de las bases de datos previamente proporcionadas por el profesor (**Sakila y World**).

Estas bases de datos son de ejemplo y de ellas deberemos realizar múltiples consultas para la medición del tiempo de ejecuciónde todas ellas. Para ello se necesitan tres tipos de consultas, dos diferentes por tipo. Las consultas proporcionadas por los integrantes del grupo han sido realizadas previamente para la verificación de su funcionamiento.

Finalmente se ha desarrollado una interfaz simple (opcional) para la visualización de los datos obtenidos.

## 2.- CONSULTAS

Para esta práctica hemos usado las bases de datos Sakila y World requeridas por el enunciado de esta. Además, se ha requerido de seis consultas por base de datos, tres pares de ellas en las que se recopilan información de consultas predefinidas, consultas simples y consultas complejas.

### 1. Consultas usada para Sakila:

- Consultas predefinidas:
  - Seleccionar los valores máximos y mínimos del campo **length** de la tabla **film**.  
**SELECT MAX(length), MIN(length) FROM film;**
  - Seleccionar el número de actores unicos de la tabla **actor**.  
**SELECT COUNT(DISTINCT first\_name, last\_name) FROM actor;**
- Consultas simples:
  - Seleccionar todos los campos y valores de la tabla **actor**.  
**SELECT \* FROM actor;**
  - Seleccionar todos los campos y valores de la tabla **customer** con la condición de que el cliente sea inactivo (**active = 0**).  
**SELECT \* FROM customer WHERE active = 0;**
- Consultas complejas:
  - Seleccionar el nombre de la película (**film\_title**) de la unión de la tablas **film**, **film\_actor**, **actor**, **inventory** y **rental**. Los cuales deben estar ordenador por el id de la película (**film.film\_id**) teniendo un conteo del campo **rental.rental\_id > 5**.  
**SELECT film.title FROM film  
INNER JOIN film\_actor ON film.film\_id = film\_actor.film\_id  
INNER JOIN actor ON film\_actor.actor\_id = actor.actor\_id  
INNER JOIN inventory ON film.film\_id = inventory.film\_id  
INNER JOIN rental; ON inventory.inventory\_id =  
rental.inventory\_id  
GROUP BY film.film\_id  
HAVING COUNT(rental.rental\_id) > 5;**

- Seleccionar el nombre (*customer.first\_name*) y apellido (*customer.last\_name*) de cliente y el conteo de los alquileres de estos (*rental.rental\_id*). La información es sacada de la unión de las tablas *customer* y *rental*. Además, esta deberá estar ordenada por el id del cliente (*customer.customer\_id*)

```
SELECT customer.first_name, customer.last_name,
COUNT(rental.rental_id)
FROM customer
INNER JOIN rental ON customer.customer_id =
rental.customer_id
GROUP BY customer.customer_id;
```

## 2. Consultas usada para World:

- Consultas predefinidas:

- Seleccionar la suma de los valores del campo *SurfaceArea* de la tabla *country*.

```
SELECT SUM(SurfaceArea) FROM country;
```

- Seleccionar los valores máximos y mínimos de la población (*population*) dividida por la superficie (*SurfaceArea*) de la tabla *country* con la condición de que el continente sea Asia (*continent = "Asia"*).

```
SELECT MAX(population/ SurfaceArea), MIN(population/
SurfaceArea) FROM country WHERE continent = "Asia";
```

- Consultas simples:

- Seleccionar todos los campos y valores de la tabla *country*.

```
SELECT * FROM country;
```

- Seleccionar todos los campos y valores de la tabla *city* con la condición de que el nombre de la ciudad empiece por la letra *S*..

```
SELECT * FROM city WHERE name LIKE "S%";
```

- Consultas complejas:

- Seleccionar el nombre del país (*country.Name*) de la unión de la tablas *country* y *countrylanguage*. Con la condición de que el idioma sea oficial (*countrylanguage.IsOfficial = "T"*), esté la información ordenada por el código del país (*country.Code*) en los países en los que se hablan más de tres idiomas.

```
SELECT country.Name FROM country
INNER JOIN countrylanguage ON country.Code =
countrylanguage.CountryCode
WHERE countrylanguage.IsOfficial = "T"
GROUP BY country.Code
HAVING COUNT(countrylanguage.Language) > 3;
```

- Seleccionar el nombre del país (**country.Name**), el nombre de la ciudad (**city.Name**), el idioma del país (**countrylanguage.Language**) y si el idioma es oficial o no (**countrylanguage.IsOfficial**). Con las condiciones de que el porcentaje del idioma hablado sea mayor al 50% (**countrylanguage.Percentage > 50.0**) y que seleccione solo aquellos países que empiecen por las letras **M** hasta la **Z**. Además, deberán estar ordenados por el nombre del país de manera ascendente.

```
SELECT country.Name, city.Name, countrylanguage.Language,
countrylanguage.IsOfficial
FROM country
INNER JOIN city ON country.Code = city.CountryCode
INNER JOIN countrylanguage ON country.Code =
countrylanguage.CountryCode
WHERE countrylanguage.Percentage > 50.0
AND country.Name REGEXP "[m-z]%"
ORDER BY country.Name ASC;
```

### 3.- APLICACIÓN / INTERFAZ

La interfaz utilizada y la implementación del código de la aplicación Java es simple.

Se han usado diferentes clases llamas: **sakilaConnect.java** y **worldConnect.java** en las cuales hemos proporcionado el código necesario para poder realizar la conexión a ambas bases de datos (**Sakila** y **World**).

Además de realizar la conexión a ambas bases de datos, se requería de la obtención del tiempo de ejecución de una serie de consultas (3 tipos: predefinidas, simples y complejas). Para ello hemos creado un metodo dentro del ambas clases **sakilaConnect.java** y **worldConnect.java** que apliquen el mismo código pero con la sintaxis apropiada para cada clase.

```
public class worldConnect {
    String url = "jdbc:mysql://localhost:3306/world";
    String usuario;
    String password;
    Statement statement;
    ResultSet resultado;
    Connection conexion;
```

(Creación de *Strings* y objeto de tipo *Connection* para la conexión a las bases de datos con *usuario* y *contraseña*, además de objetos de tipo *Statement* y *ResultSet* para las consultas y ejecución de estas)

```

public long medirTiempoConsulta(String consulta, String usuario2, String password2) {
    long tiempo = 0;
    usuario = usuario2;
    password = password2;

    try {

        conexion = DriverManager.getConnection(url, usuario, password);
        statement = conexion.createStatement();
        long inicio = System.nanoTime();
        resultado = statement.executeQuery(consulta);
        long fin = System.nanoTime();
        tiempo = fin - inicio;

        System.out.println("Tiempo de ejecución de la consulta: " + tiempo + " ms");

    } catch (SQLException e) {
        System.err.println("Error al conectar a la base de datos: " + e.getMessage());
    }

    return tiempo;
}
}

```

Hemos hecho uso de nuestros conocimientos de la primera parte de la asignatura de Bases de Datos I para realizar la conexión y la ejecución de las consultas y hemos añadido un metodo del sistema para poder medir el tiempo de ejecución de las consultas en nanosegundos. De ahí hemos obtenido el tiempo restando lo que tarda al inicio con respecto al final.

Posteriormente hemos realizado una interfaz para poder mostrar los resultados para esta práctica. Para ello hemos creado funciones para las dos bases y la ejecución de sus respectivas consultas y poder imprimir el tiempo de ejecución.

```

private void imprimirSakila() {
    sakilaConnect miConexion = new sakilaConnect();

    long tiempoConsultaPredefinidaSakila1 = miConexion.medirTiempoConsulta("SELECT MAX(length), MIN(length) FROM film:", usuario1, password1);
    long tiempoConsultaPredefinidaSakila2 = miConexion.medirTiempoConsulta("SELECT COUNT(DISTINCT first_name, last_name) FROM actor:", usuario1, password1);
    String textoS1 = String.valueOf(tiempoConsultaPredefinidaSakila1) + " ns" + "\n" + String.valueOf(tiempoConsultaPredefinidaSakila2) + " ns";
    JTextArea4.setText(textoS1);
}

private void imprimirWorld() {
    worldConnect miConexion = new worldConnect();

    long tiempoConsultaPredefinidaWorld1 = miConexion.medirTiempoConsulta("SELECT SUM(SurfaceArea) FROM country:", usuario1, password1);
    long tiempoConsultaPredefinidaWorld2 = miConexion.medirTiempoConsulta("SELECT MAX(population/SurfaceArea), MIN(population/SurfaceArea) FROM country WHERE continent = 'Asia':", usuario1, password1);
    String textoW1 = String.valueOf(tiempoConsultaPredefinidaWorld1) + " ns" + "\n" + String.valueOf(tiempoConsultaPredefinidaWorld2) + " ns";
    JTextArea5.setText(textoW1);
}

```

Este es un ejemplo de las múltiples consultas realizadas (todas con la misma estructura de código)

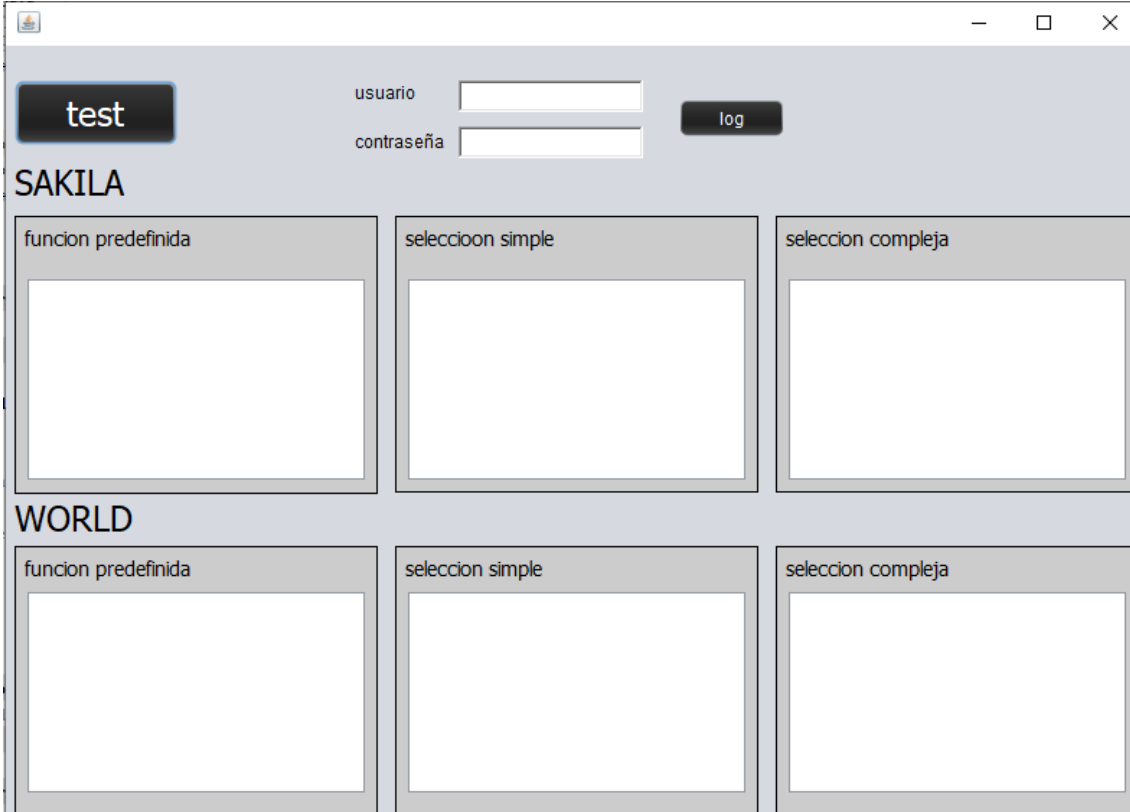
Luego hemos creado una función para realizar ambas funciones a la vez y que resulte más fácil de implementar en la interfaz.

```

private void imprimirGlobal() {
    imprimirSakila();
    imprimirWorld();
}

```

La interfaz se compone de diferentes botones, paneles y casillas de texto para la diferente información:



The image shows a graphical user interface for testing database queries. At the top, there is a 'test' button on the left. To its right are two input fields: 'usuario' and 'contraseña', followed by a 'log' button. Below this header, the interface is divided into two main sections: 'SAKILA' and 'WORLD'. Each section contains three panels: 'funcion predefinida', 'seleccioon simple' (note the typo), and 'seleccion compleja'. Each panel has a large white rectangular area for displaying results.

Para poder obtener los tiempos es necesario primero introducir el usuario y la contraseña en las casillas que indicamos con **usuario** y **contraseña** en su lado izquierdo. Una vez introducidos hacemos click en el botón **log** y si las credenciales son correctas estaremos conectados a las bases de datos.

Los datos recomendados a introducir son: el usuario **root** y la contraseña que tenga aquella persona para conectarse a su **MySQL**.

En caso de no seguir queriendo esas indicaciones puede crear un usuario con una contraseña y usarlos para conectarse a la base de datos.

Una vez conectados, para ejecutar el programa hacemos click en el botón **test** y estaría todo listo, en pocos segundos deberían aparecer diferentes tiempos en nanosegundos (**ns**) en las casillas blancas que indican que tipo de consulta se ha ejecutado. Saldrán dos tiempos, uno encima de otro, lo que indica que se han ejecutado dos consultas de ese tipo.

Los resultados se muestran como indicamos en el siguiente punto (4).

#### 4.- INTERPRETACIÓN DE LOS RESULTADOS

Una vez ejecutado el programa y haber introducido el usuario y la contraseña para poder establecer conexión con las bases de datos, ejecutamos el test. El test ejecuta las consultas explicadas en el punto 2 de la memoria (consultar en caso de ser necesario).



La primera vez que se ejecuta el test salen datos (tiempos) parecidos a estos:

The screenshot shows a web application window titled "test" with a "log" button. Below the header, the "SAKILA" database results are displayed in a 3x3 grid. The first column, "funcion predefinida", shows two values: 43327300ns and 3088300ns. The second column, "seleccioon simple", shows 5742000ns and 2549000ns. The third column, "seleccion compleja", shows 207802100ns and 11478100ns. Below this, the "WORLD" database results are shown in a similar 3x3 grid. The first column shows 2517600ns and 2517600ns. The second column shows 3952900ns and 6758000ns. The third column shows 5007100ns and 12693400ns.

funcion predefinida	seleccioon simple	seleccion compleja
43327300ns	5742000ns	207802100ns
3088300ns	2549000ns	11478100ns

funcion predefinida	seleccion simple	seleccion compleja
2517600ns	3952900ns	5007100ns
2517600ns	6758000ns	12693400ns

Los datos después de ejecutar el test un par de veces los tiempos son diferentes con respecto a la primera ejecución. Esto ocurre ya que en la primera ejecución se tiene que obtener todo por primera vez por lo que al ejecutarlo más veces los tiempos serán menores al saber ya como conseguir la información de las consultas.

The screenshot shows the same web application window as above, but with different benchmark results. The "SAKILA" database results are: "funcion predefinida" (1267300ns, 2949600ns), "seleccioon simple" (1246500ns, 2176600ns), and "seleccion compleja" (207152900ns, 8867600ns). The "WORLD" database results are: "funcion predefinida" (29214500ns, 29214500ns), "seleccion simple" (1819600ns, 3558100ns), and "seleccion compleja" (2476600ns, 7328600ns).

funcion predefinida	seleccioon simple	seleccion compleja
1267300ns	1246500ns	207152900ns
2949600ns	2176600ns	8867600ns

funcion predefinida	seleccion simple	seleccion compleja
29214500ns	1819600ns	2476600ns
29214500ns	3558100ns	7328600ns

Los tiempos difieren por multiples factores como lo pueden ser: el número de operaciones o funciones a realizar (en caso de tener), el número de tablas del cual se obtienen los datos, la cantidad de datos que se piden imprimir, la cantidad de datos entre los que hay que buscar, la cantidad de líneas de datos impresas o la cantidad de datos necesaria para poder ejecutar las condiciones de ordenación o unión de tablas por ejemplo.

Es por ello que:

**(En todas ellas influirá en el tiempo de ejecución: la cantidad de datos impresos, datos entre los que hay que buscar los correctos y la cantidad de datos que se quieren obtener).**

En Sakila:

#### Consultas predefinidas:

Los tiempos de estas consultas difieren seguramente por las operaciones a realizar, ya que la primera consulta hace uso de dos funciones (**MAX()** y **MIN()**) y en cambio la segunda solo tiene una operación (**COUNT()**).

El tiempo es menor en la primera consulta ya que las funciones son simples pero en la segunda consulta tiene una condición extra dentro de la función la cual debe realizar.

#### Consultas simples:

Los tiempos de estas consultas difieren por la condiciones que se les imponen a estas. En ambas se pide mostrar todos los datos que contiene su respectiva tabla aunque a la segunda consulta se impone una condición simple que a la primera de ellas no se le impone.

Por ello el tiempo de ejecución de la segunda consulta es mayor que el de la primera.

#### Consultas complejas:

Los tiempos de estas consultas difieren debido a la cantidad de tablas requeridas para poder obtener la información (la primera consulta requiere de 5 tablas y la segunda de 2 nada más). Además, la primera consulta tiene una condición extra que la segunda no tiene.

Por ello el tiempo de ejecución de la primera consulta compleja es mayor que el de la segunda.

En World:

#### Consultas predefinidas:

El tiempo de ejecución de estas consultas no difiere y no entendemos exactamente por qué. Comparando ambas consultas podemos observar que en la primera de ellas se ejecuta una sola operación de suma (**SUM()**) mientras que en la segunda, se realizan dos de ellas (**MAX()** y **MIN()**) además de que se le atribuye una condición que la primera consulta prescinde.

En caso de que los tiempos fueran diferentes, entenderíamos que el tiempo de la primera consulta fuera menos a la de la segunda.

#### Consultas simples:

Los tiempos de estas consultas difieren por la posible razón de que a pesar de que ambas consultas obtengan todo lo que contienen sus tablas a una de ellas se le atribuye una condición simple.

Por ello el tiempo de ejecución de la primera consulta es menor con respecto a la segunda consulta.

#### Consultas complejas:

Los tiempos de ejecución de estas son diferentes ya que hacen uso de un número diferente de tablas y las diferentes condiciones impuestas. La primera consulta hace uso de uso de 2 tablas frente a las 3 que usa la segunda consulta para la obtención de los datos. Además, a pesar de tener un mismo número de condiciones, estas son diferentes por lo que el tiempo de ejecución de estas puede influir.

Por estas razones el tiempo de ejecución de la primera consulta es menor que el de la segunda.

### **5.- CONCLUSIÓN**

Podemos concluir que el tiempo de ejecución de las consultas puede variar por muchos motivos como la cantidad de datos impresos y de los diferentes tipos que se quieren obtener de estos, entre los mencionados anteriormente en el punto 4.

Además, podemos concluir que el tiempo de ejecución de una consulta es mayor cuanto mayor cantidad de operaciones deba realizar, o de condiciones a tener en cuenta. Por lo que consultas simples o “cortas” pueden ser un índice de que su tiempo sea mínimo.