



# PRÁCTICA 2, FASE 2

Archivos y bases de datos

Joel López – joel.lopez.2015  
Alex Pons – ls31155

# Índice

1.	Resumen del enunciado (Fase 1 y Fase 2) .....	2
2.	Conceptos teóricos.....	2
3.	Diagramas de la base de datos .....	3
4.	Extracción de datos por línea de comandos. ....	5
	En LOCAL: .....	5
	En REMOTO: .....	6
5.	Proceso de importación y exportación de la base de datos. ....	9
6.	Explicación detallada de la gestión de usuarios (creación y control de acceso) .....	11
7.	Explicación detallada de como se ha realizado la lógica del buscador .....	14
8.	Modificaciones en la interfaz proporcionada .....	16
9.	Dedicación .....	16
10.	Conclusiones .....	17
11.	Bibliografía .....	17

## 1. Resumen del enunciado (Fase 1 y Fase 2)

### Fase 1

Tenemos que configurar una interface con Java capaz de conectarse con una base de datos que se encuentra subida en el servidor remoto Puigpedros de La Salle. También se nos solicita hacer un buscador de esta base de datos offline, por lo que tendremos que tener una copia de seguridad local de la base de datos.

### Fase 2

Una vez tengamos la base de datos en nuestro programa en Java, el usuario mediante una interface que se nos proporciona, ha de poder crear nuevos usuarios (no se puede crear usuarios si ya existen y no se podrá acceder a la base de datos con un usuario que no exista). Cada vez que se realice el proceso de login, nuestro programa tendrá que refrescarse y decir cuánto tiempo ha pasado desde la última vez que se ha hecho una copia. También nuestro programa constará de un buscador, por el cual, nos ha de permitir poder ver las búsquedas que se hagan en la interfaz.

## 2. Conceptos teóricos

### OLTP (Procesamiento de Transacciones En Línea)

El OLTP es un tipo de procesamiento que facilita y administra aplicaciones transaccionales, usualmente para entrada de datos y recuperación y procesamiento de transacciones. Los paquetes de software para OLTP se basan en la arquitectura cliente-servidor ya que suelen ser utilizados por empresas con una red informática distribuida. El procesamiento de transacciones en línea tiene dos claros beneficios: la simplicidad y la eficiencia. Cuando hablamos de sus inconvenientes se puede decir que en su utilización hay algunas cuestiones en las que se debe pensar ya que pueden suponer un problema: la seguridad y los costes económicos o de tiempo.

### OLAP (Procesamiento Analítico En Línea)

EL OLAP es una solución utilizada en el campo de la llamada Inteligencia de negocios cuyo objetivo es agilizar la consulta de grandes cantidades de datos. Para ello utiliza estructuras de datos diversas, normalmente multidimensionales (Cubos OLAP), que contienen datos resumidos de grandes BBDD o Sistemas Transaccionales (OLTP). Se usa en informes de negocios de ventas, marketing, informes de dirección y áreas similares. La principal característica que potencia, es que es lo más rápido a la hora de ejecutar sentencias SQL de tipo SELECT, en contraposición con OLTP que es la mejor opción para operaciones de tipo INSERT, UPDATE Y DELETE.

### Data Warehouse (almacenamiento de datos)

*Data warehouse* es como se denomina al almacén electrónico donde generalmente una empresa u organización mantiene una gran cantidad de información. Los datos de un *data warehouse* deben almacenarse de forma segura, fiable, fácil de recuperar y fácil de administrar.

Ventajas principales de usar *data warehouse*:

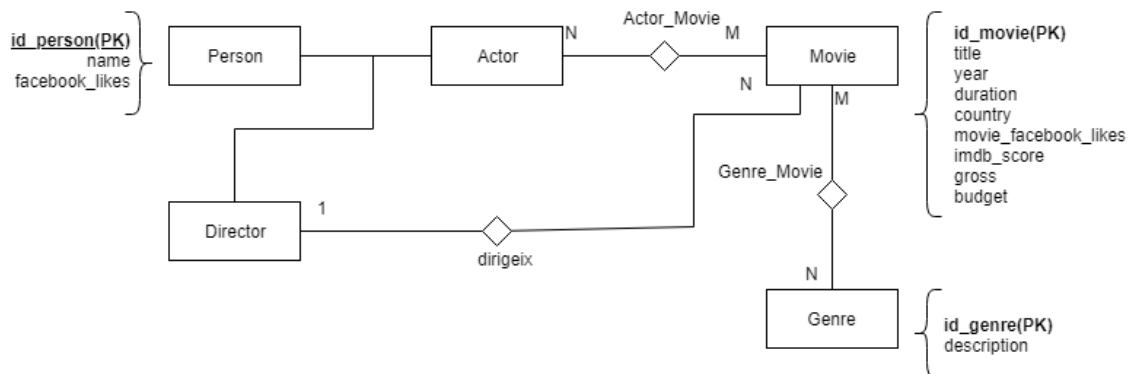
- Los almacenes de datos hacen más fácil el acceso a una gran variedad de datos a los usuarios finales
- Facilitan el funcionamiento de las aplicaciones de los sistemas de apoyo a la decisión tales como informes de tendencia, por ejemplo: obtener los ítems con la mayoría de las ventas en un área en particular dentro de los últimos dos años; informes de excepción, informes que muestran los resultados reales frente a los objetivos planteados a priori.

Primeros inconvenientes:

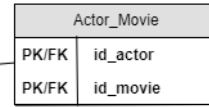
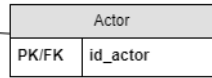
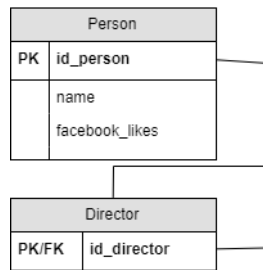
- A lo largo de su vida los almacenes de datos pueden suponer altos costos. El almacén de datos no suele ser estático. Los costos de mantenimiento son elevados.
- Los almacenes de datos se pueden quedar obsoletos relativamente pronto.
- A veces, ante una petición de información estos devuelven una información subóptima, que también supone una pérdida para la organización.

### 3. Diagramas de la base de datos

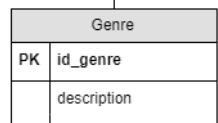
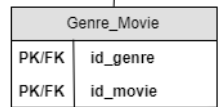
#### Modelo Entidad-Relación



## Modelo



## Relacional

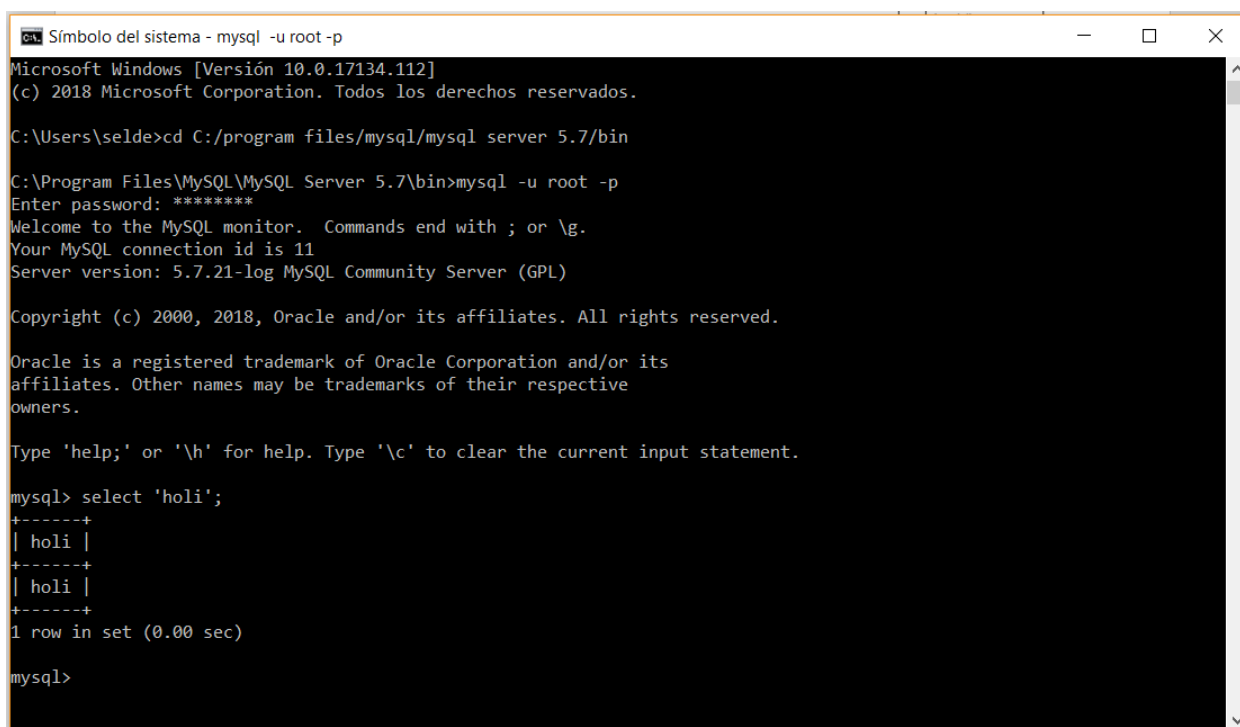


## 4. Extracción de datos por línea de comandos.

Para acceder a la información de la base de datos que debíamos importar debimos utilizar una serie de comandos básicos tras establecer la conexión en mysql desde el canal SSH, y para ver la información también de la base de datos local, donde íbamos a exportar los datos:

En LOCAL:

- **Para abrir el mysql por cmd:** ir hasta el directorio de mysql.server, y una vez allí escribir:  
`mysql -u <usuario> -p`
- *[luego pide contraseña, la introduces y has accedido al mysql desde la cmd]*



```
Símbolo del sistema - mysql -u root -p
Microsoft Windows [Versión 10.0.17134.112]
(c) 2018 Microsoft Corporation. Todos los derechos reservados.

C:\Users\selde>cd C:/program files/mysql/mysql server 5.7/bin

C:\Program Files\MySQL\MySQL Server 5.7\bin>mysql -u root -p
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 11
Server version: 5.7.21-log MySQL Community Server (GPL)

Copyright (c) 2000, 2018, Oracle and/or its affiliates. All rights reserved.

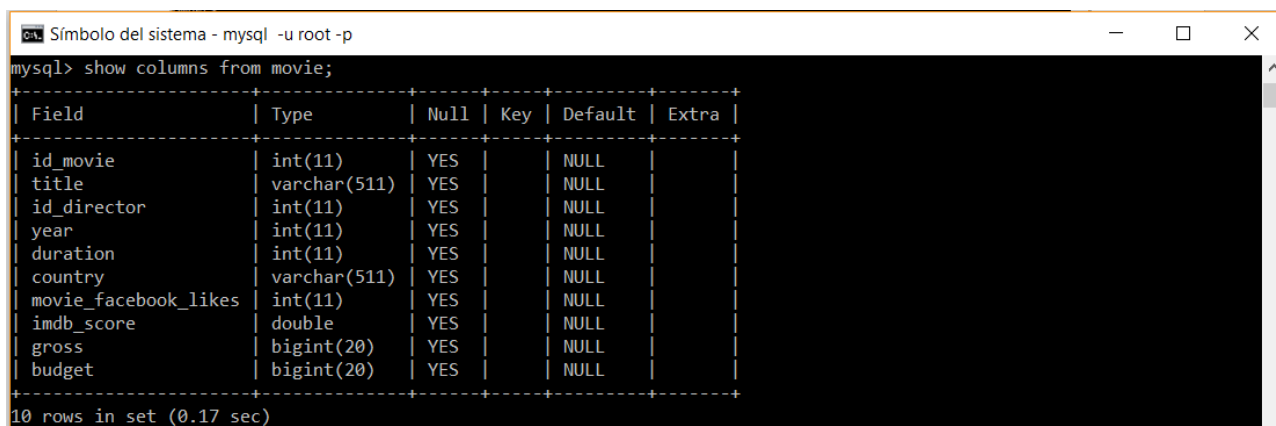
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> select 'holi';
+-----+
| holi |
+-----+
| holi |
+-----+
1 row in set (0.00 sec)

mysql>
```

- ➔ A partir de aquí, las comandas a utilizar son las mismas que si estuviéramos en MySQL (*use Movies;* para ver la base de datos; *show tables;* para ver las tablas de la base de datos que estamos utilizando; ***show columns from <tabla>;*** para ver todos los atributos de una tabla y sus respectivos tipos, además de otra información no tan relevante... -esto fue útil para ver cómo son las tablas de la base de datos remota!).

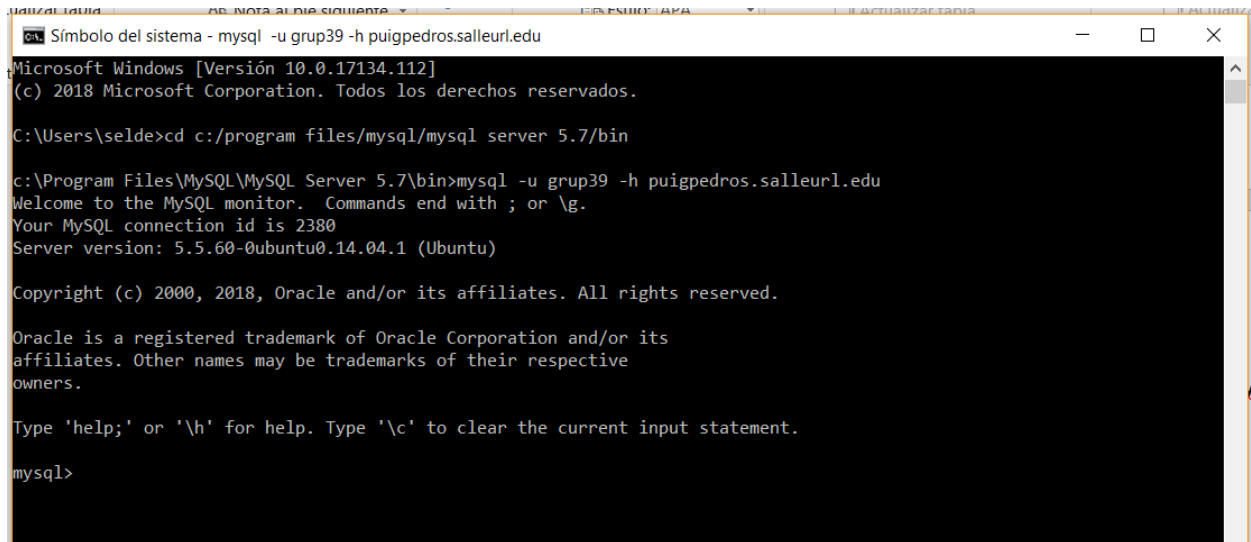


```
Símbolo del sistema - mysql -u root -p
mysql> show columns from movie;
+-----+-----+-----+-----+-----+-----+
| Field          | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id_movie       | int(11)       | YES  |     | NULL    |       |
| title          | varchar(511)  | YES  |     | NULL    |       |
| id_director    | int(11)       | YES  |     | NULL    |       |
| year           | int(11)       | YES  |     | NULL    |       |
| duration       | int(11)       | YES  |     | NULL    |       |
| country        | varchar(511)  | YES  |     | NULL    |       |
| movie_facebook_likes | int(11)       | YES  |     | NULL    |       |
| imdb_score     | double        | YES  |     | NULL    |       |
| gross          | bigint(20)    | YES  |     | NULL    |       |
| budget         | bigint(20)    | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
10 rows in set (0.17 sec)
```

### En REMOTO:

Podemos ir a la carpeta de mysql server, como antes, o abrir directamente desde el **mysql Shell**, y escribir:

- **Para conectarnos al servidor remoto de puigpedros por línea de comandos:**  
`mysql -u <usuario1> -h puigpedros.salleurl.edu`
- En caso de pedirnos contraseña (no la pide, pero bueno), al no tener ninguna simplemente pulsamos enter y ya estaríamos dentro:



```
Símbolo del sistema - mysql -u grup39 -h puigpedros.salleurl.edu
Microsoft Windows [Versión 10.0.17134.112]
(c) 2018 Microsoft Corporation. Todos los derechos reservados.

C:\Users\selde>cd c:/program files/mysql/mysql server 5.7/bin

c:\Program Files\MySQL\MySQL Server 5.7\bin>mysql -u grup39 -h puigpedros.salleurl.edu
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2380
Server version: 5.5.60-0ubuntu0.14.04.1 (Ubuntu)

Copyright (c) 2000, 2018, Oracle and/or its affiliates. All rights reserved.

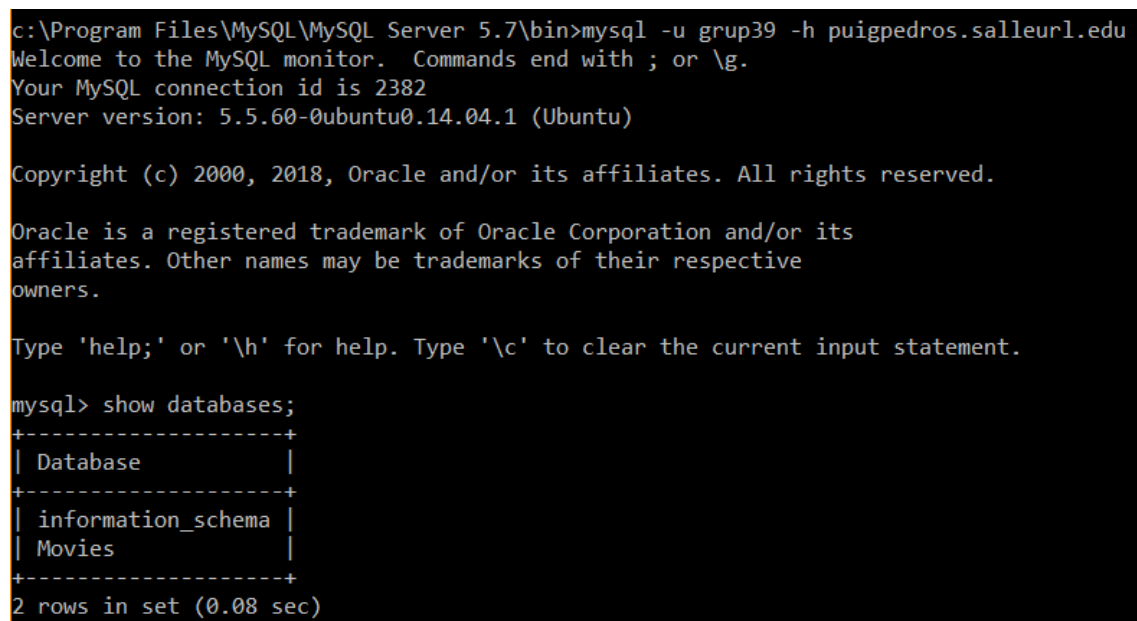
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

Ahora que ya estamos conectados al servidor, podemos utilizar las mismas comandas que en la conexión local para encontrar información sobre la base de datos a importar. Nuestras comandas más utilizadas para esto han sido (en orden de uso):

- ➔ `show databases;` (para ver qué bases de datos había en el servidor remoto)



```
c:\Program Files\MySQL\MySQL Server 5.7\bin>mysql -u grup39 -h puigpedros.salleurl.edu
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2382
Server version: 5.5.60-0ubuntu0.14.04.1 (Ubuntu)

Copyright (c) 2000, 2018, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+-----+
| Database                |
+-----+
| information_schema       |
| Movies                   |
+-----+
2 rows in set (0.08 sec)
```

---

<sup>1</sup> En nuestro caso, 'grup39'.

→ use Movies; (para utilizar la base de datos en cuestión)

```
mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| Movies |
+-----+
2 rows in set (0.08 sec)

mysql> use Movies;
Database changed
mysql>
```

→ show tables; (para ver qué tablas había en la base de datos a exportar –'Movies')

```
mysql> use Movies;
Database changed
mysql> show tables;
+-----+
| Tables_in_Movies |
+-----+
| Actor |
| Actor_movie |
| Director |
| Genre |
| Genre_Movie |
| Movie |
| Person |
+-----+
7 rows in set (0.08 sec)

mysql>
```

→ show columns from <tabla>; (muestra todas las columnas y tipos de cada atributo, en el ejemplo de la tabla 'Movie')

```
mysql> show columns from Movie;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id_movie | int(11) | NO | PRI | NULL | auto_increment |
| title | varchar(511) | YES | | NULL | |
| id_director | int(11) | YES | MUL | NULL | |
| year | int(11) | YES | | NULL | |
| duration | int(11) | YES | | NULL | |
| country | varchar(511) | YES | | NULL | |
| movie_facebook_likes | int(11) | YES | | NULL | |
| imdb_score | double | YES | | NULL | |
| gross | bigint(20) | YES | | NULL | |
| budget | bigint(20) | YES | | NULL | |
+-----+-----+-----+-----+-----+-----+
10 rows in set (0.08 sec)
```



**MODO ALTERNATIVO:** tras conectarnos al servidor remoto del mismo modo, una vez hemos accedido al sistema de input por mysql, utilizar el *information\_schema*:

→ use information\_schema; (para utilizar esta base de datos)

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2387
Server version: 5.5.60-0ubuntu0.14.04.1 (Ubuntu)

Copyright (c) 2000, 2018, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use information_schema;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql>
```

→ show tables; (para ver de qué tablas disponemos para obtener la información que necesitamos):

*Entonces podemos ver la tabla COLUMNS, que puede contener información relevante; además de poder hacer una query sencilla para sacar más información:*

```
Database changed
mysql> show tables;
+-----+
| Tables_in_information_schema |
+-----+
| CHARACTER_SETS               |
| COLLATIONS                   |
| COLLATION_CHARACTER_SET_APPLICABILITY |
| COLUMNS                     |
| COLUMN_PRIVILEGES            |
| ENGINES                      |
| EVENTS                       |
| FILES                        |
| GLOBAL_STATUS                |
| GLOBAL_VARIABLES             |
| KEY_COLUMN_USAGE             |
| PARAMETERS                   |
| PARTITIONS                   |
| PLUGINS                      |
| PROCESSLIST                  |
| PROFILING                    |
| REFERENTIAL_CONSTRAINTS     |
| ROUTINES                     |
| SCHEMATA                     |
| SCHEMA_PRIVILEGES            |
| SESSION_STATUS               |
| SESSION_VARIABLES            |
| STATISTICS                   |
| TABLES                      |
| TABLESPACES                 |
| TABLE_CONSTRAINTS           |
| TABLE_PRIVILEGES            |
| TRIGGERS                     |
| USER_PRIVILEGES              |
| VIEWS                        |
| INNODB_BUFFER_PAGE           |
| INNODB_TRX                   |
| INNODB_BUFFER_POOL_STATS     |
| INNODB_LOCK_WAITS            |
| INNODB_CMPMEM                |
| INNODB_CMP                   |
| INNODB_LOCKS                 |
| INNODB_CMPMEM_RESET         |
| INNODB_CMP_RESET             |
| INNODB_BUFFER_PAGE_LRU      |
+-----+
40 rows in set (0.00 sec)
```

→ select \* from TABLES where TABLE\_SCHEMA not like 'information\_schema' group by TABLE\_NAME;

```
mysql> select * from TABLES where TABLE_SCHEMA not like 'information_schema' group by TABLE_NAME;
```

TABLE_CATALOG	TABLE_SCHEMA	TABLE_NAME	TABLE_TYPE	ENGINE	VERSION	ROW_FORMAT	TABLE_ROWS	AVG_ROW_LENGTH	DATA_LENGTH	MAX_DATA_LENGTH	INDEX_LENGTH	DATA_FREE	AUTO_INCREMENT
def	Movies	Actor	BASE TABLE	InnoDB	10	Compact	6338	28	180224	0	0	4194304	
def	Movies	Actor_movie	BASE TABLE	InnoDB	10	Compact	12721	124	1589248	0	294912	4194304	
def	Movies	Director	BASE TABLE	InnoDB	10	Compact	1776	55	98304	0	0	4194304	
def	Movies	Genre	BASE TABLE	InnoDB	10	Compact	27	606	16384	0	0	4194304	
def	Movies	Genre_movie	BASE TABLE	InnoDB	10	Compact	19725	80	1589248	0	376832	4194304	
def	Movies	Movie	BASE TABLE	InnoDB	10	Compact	5155	95	491520	0	147456	4194304	
def	Movies	Person	BASE TABLE	InnoDB	10	Compact	8767	46	409600	0	0	4194304	

7 rows in set (0.01 sec)

→ También podríamos sacar más información de esa tabla COLUMNS, haciendo referencia a los atributos, como el tipo y los privilegios.

## 5. Proceso de importación y exportación de la base de datos.

A grandes pasos, el proceso de importación y exportación de la base Movies es el siguiente:

- Establecer una conexión con ambas bases de datos (la remota, desde donde importaremos los datos; y la local, donde los exportaremos).
- Al utilizar un sistema que utiliza dos variables (ambas del mismo tipo, un puente SSH), una para la remota y otra para la local, podemos pasar la información de una a otra sin necesidad de almacenar los datos en la aplicación. Dicho esto, abrimos las conexiones.

```
// 1. Inicialitzem les connexions
// 1.1. Crea classe que connecta amb la base remota
SSHConnector remoteCon = new SSHConnector( USR: "grup" + remoteNGroup, pass: "", db: "Movies", port: 3306, dest: "puigpedros");
// 1.2. Crea classe que connecta amb la base local
SSHConnector localCon = new SSHConnector( USR: "root", localPass, db: "Movies", port: 3306, dest: "localhost");

// 2. Connecta amb ambdues classes
connected = remoteCon.connect() && localCon.connect();
```

- [NO NECESARIO] Mostramos por pantalla las tablas que contiene dicha base de datos.

```
public void mostraTaules(SSHConnector remoteCon) {
    ResultSet remoteTables;
    // *. Mostrem les taules a importar:
    System.out.println("\nAquestes són les taules a importar de la base de dades remota:");
    remoteTables = remoteCon.selectQuery("show tables;");
    String result = "";
    try {
        while (remoteTables.next()) {
            result = remoteTables.getString( columnLabel: "Tables_in_movies");
            System.out.println("\t\t\t\t\t" + result);
        }
        System.out.println("");
    } catch (SQLException e) {
        System.out.println("Error de SQL.");
    }
}
```

- Si las conexiones se han establecido correctamente, comenzamos a importar: aunque podríamos ir tabla a tabla importando y exportando reutilizando una misma variable, al no ser un requisito, hicimos una variable de tipo ResultSet para cada tabla, de forma que queda más intuitivo: importamos, con una query, toda la información (*select \* from <tabla>*; en dicha variable ResultSet.

```
// 2. Connecta amb ambdues classes
connected = remoteCon.connect() && localCon.connect();

if (connected) {
    System.out.println("\t-> Hem connectat amb ambdues bases de dades, ara passarem a importar la informació!");
    remoteCon.mostraTaules(remoteCon);

    // 3. Preparem tots els ResultSet's (tot i que podriem optimitzar-ho, així es mes intuítu)
    ResultSet person = remoteCon.selectQuery("select * from Person;");
    ResultSet director = remoteCon.selectQuery("select * from Director;");
    ResultSet actor = remoteCon.selectQuery("select * from Actor;");
    ResultSet actor_movie = remoteCon.selectQuery("select * from Actor_movie;");
    ResultSet movie = remoteCon.selectQuery("select * from Movie;");
    ResultSet genre_movie = remoteCon.selectQuery("select * from Genre_Movie;");
    ResultSet genre = remoteCon.selectQuery("select * from Genre;");
}
```

- Ahora, para cada tabla:
  - o Pasamos la información del ResultSet a la tabla en cuestión con un *insert into <tabla> values (<datos\_del\_resultset>);* [el ejemplo es solo de la tabla **Person** i **Movie**]

```
// 4.1. Person
try {
    System.out.print("\t-> Comencem a exportar a la taula 'Person'...\t");
    while (person.next()) {
        localCon.insertQuery("insert into person values (" +
            person.getInt( columnLabel: "id_person") + ", " +
            "" + person.getString( columnLabel: "name").replace( target: "'", replacement: " ") + ", " +
            person.getInt( columnLabel: "facebook_likes") + ");");
    }
    System.out.println("'Person' exportada amb éxit!");
} catch (SQLException ex) {
    System.out.println("\t-> Problema de SQL!");
}

// 4.5. Movie
try {
    System.out.print("\t-> Comencem a exportar a la taula 'Movie'...\t");
    while (movie.next()) {
        localCon.insertQuery("insert into movie values (" +
            movie.getInt( columnLabel: "id_movie") + ", " +
            "" + movie.getString( columnLabel: "title").replace( target: "'", replacement: " ") + ", " +
            movie.getInt( columnLabel: "id_director") + ", " +
            movie.getInt( columnLabel: "year") + ", " +
            movie.getInt( columnLabel: "duration") + ", " +
            "" + movie.getString( columnLabel: "country").replace( target: "'", replacement: " ") + ", " +
            movie.getInt( columnLabel: "movie_facebook_likes") + ", " +
            movie.getDouble( columnLabel: "imdb_score") + ", " +
            movie.getLong( columnLabel: "gross") + ", " +
            movie.getLong( columnLabel: "budget") + ");");
    }
    System.out.println("'Movie' exportada amb éxit!");
} catch (SQLException ex) {
    System.out.println("\t-> Problema de SQL!");
}
```

- Desconectamos ambas bases de datos.

```
// x. Tanquem les sessions i sortim del sistema:
System.out.println("\t-> Sortim de les sessions obertes de cada connexió...");
remoteCon.disconnect();
System.out.println("\t\t o Hem sortit de la connexió remota!");
localCon.disconnect();
System.out.println("\t\t o Hem sortit de la connexió local!");
System.out.println("\n >> Sortint del sistema... <<");
```

## RESULTAT:

```
Escriu el nombre de grup assignat (només el número!): 
Escriu la contrassenya per connectar amb la base local: 
Connexió a base de dades jdbc:mysql://puigpedros:3306/Movies ... Ok
Connexió a base de dades jdbc:mysql://localhost:3306/Movies ... Ok
-> Hem connectat amb ambdues bases de dades, ara passarem a importar la informació!

Aquestes són les taules a importar de la base de dades remota:
- Actor
- Actor_movie
- Director
- Genre
- Genre_Movie
- Movie
- Person

-> Comencem a exportar a la taula 'Person'... 'Person' exportada amb éxit!
-> Comencem a exportar a la taula 'Director'... 'Director' exportada amb éxit!
-> Comencem a exportar a la taula 'Actor'... 'Actor' exportada amb éxit!
-> Comencem a exportar a la taula 'Actor_Movie'... 'Actor_Movie' exportada amb éxit!
-> Comencem a exportar a la taula 'Movie'... 'Movie' exportada amb éxit!
-> Comencem a exportar a la taula 'Movie_Genre'... 'Movie_Genre' exportada amb éxit!
-> Comencem a exportar a la taula 'Genre'... 'Genre' exportada amb éxit!

-> Sortim de les sessions obertes de cada connexió...
    o Hem sortit de la connexió remota!
    o Hem sortit de la connexió local!

>> Sortint del sistema... <<

Process finished with exit code 0
```

## 6. Explicación detallada de la gestión de usuarios (creación y control de acceso)

Para la gestión de usuario utilizaremos las herramientas que nos ofrece MySQL, tanto de acceso y uso como en creación y asignación de privilegios (pese a que en esta ocasión éstos no tienen tanta influencia).

Según los distintos casos y, en orden lógico de uso, esto sería lo que sucedería:

**PRIMER PASO:** El usuario abre el sistema y la interfaz de login se muestra sin ningún cambio respecto a la ofrecida.

**SEGUNDO PASO:** El programa intenta conectarse al servidor local con los datos introducidos en los campos de la interfaz, y si *connect()* devuelve un false quiere decir que no ha logrado conectarse. Por otro lado, si éste devuelve un true significa que la conexión se ha establecido satisfactoriamente.

```
private void ferLogin() {
    try {
        this.localCon = new SSHConnector(usuari.getName(), usuari.getPass(), db: "Movies", port: 3306, dest: "localhost");
        if (localCon.connect()) {
            finestraPrincipal.swapToSearchPanel(); //canvia al panell de cerca
            finestraPrincipal.addUser(getLoginFoto("joel.lopez.2015"), finestraPrincipal.getLogin());
            this.actualitzaDBMovies(); //pop up i carrega de DB Movies desde el servidor remot
        }
        else JOptionPane.showMessageDialog(finestraPrincipal, message: "No s'ha pogut fer login!", title: "ERROR", JOptionPane.ERROR_MESSAGE);
    } catch (Exception e) {
        System.out.println("Error al fer el LOGIN!");
    }
}
```

Tras introducir los datos, o sin haberlos introducido, estas serían las opciones.

- El nombre de usuario y su contraseña no existe o no se ha encontrado: se muestra un Pop-up con dicha información.
- El login se ha realizado correctamente: se muestra un Pop-up con dicha información y la interfaz cambia a la de búsqueda.
- El usuario quiere registrarse: la interfaz cambia por la de registrar nuevo usuario y la gestión cambia ligeramente.

**REGISTRANDO UN NUEVO USUARIO:** El programa establece una conexión automática como

```
private void ferRegistre() {
    try {
        // per fer registre s'ha de connectar amb l'admin i crear l'usuari (REUTILITZEM LA CLASSE SSHConnector)
        this.localCon = new SSHConnector(usr: "admin", pass: "adminpass", db: "Movies", port: 3306, dest: "localhost");
        if (localCon.connect()) {
            boolean afegit = localCon.insertQuery("call afegixUsuari('"+usuari.getName()+"', '"+usuari.getPass()+"');");
            localCon.disconnect();
            if (afegit) {
                JOptionPane.showMessageDialog(finestraPrincipal, message: "Usuari creat amb èxit!");
                finestraPrincipal.swapToLoginPanel();
            }
            else {
                JOptionPane.showMessageDialog(finestraPrincipal, message: "L'usuari ja existeix! Escull un altre nom!", title: "ERROR", JOptionPane.ERROR_MESSAGE);
            }
            //amb això basta, perquè si l'usuari no existeix el procedire ja l'haurà registrat!
        }
        else JOptionPane.showMessageDialog(finestraPrincipal, message: "No es pot establir comunicació amb la base de dades!", title: "ERROR", JOptionPane.ERROR_MESSAGE);
    } catch (Exception e) {
        System.out.println("Error al fer el REGISTRE!");
    }
}
```

administrador de la base de datos, ya que si no podemos añadir un usuario. Una vez se ha conectado como administrador a la base local (en caso contrario aparecerá un Pop-up informando de un error de conexión con ésta), llama al procedimiento *afegixUsuari()* (con los parámetros con el valor de los campos nombre y contraseña introducidos en la interfaz) como si fuera una declaración de inserción de datos, debido a que necesitamos un valor de retorno que nos informe de si ha conseguido crear el usuario o ha surgido alguna complicación: el booleano “afegit” (que será false cuando la llamada *executeCommand* devuelva una excepción -haya habido algún problema).

→ Procedimiento **afegeixUsuari(name, pass);**

```
delimiter $$
drop procedure if exists afegeixUsuari$$
create procedure afegeixUsuari(in name varchar(255), in pass varchar(255))
begin
    set @aux = concat('create user \'', name, '\' identified by \'', pass, '\';');
    set @grants = concat('grant all on *.* to \'', name, '\' identified by \'', pass, '\';');
    -- select @aux;
    -- select @grants;
    prepare stmt1 from @aux;
    execute stmt1;
    deallocate prepare stmt1;
    prepare stmt2 from @grants;
    execute stmt2;
    deallocate prepare stmt2;
end $$
delimiter ;
```

Este procedimiento es una simple llamada de creación de usuario y asignación de todos privilegios menos el de asignar privilegios (pese a que en el programa bastaría con hacer queries, de este modo no da problema y es más sencillo). Esta operación puede resultar en:

- Registro satisfactorio: la interfaz vuelve a la de hacer login (no hace login automáticamente), tras mostrar un Pop-up informando del éxito del registro.
- Registro desfavorable: muestra un Pop-up informando del problema y permite volver a introducir los datos, no cambia de interfaz.

## 7. Explicación detallada de como se ha realizado la lógica del buscador

La implementación del buscador ha sido parecida a la del registro, llamando a un procedimiento habitual, sólo que esta vez, al necesitar una tabla de resultados, hemos “disfrazado” la llamada al *stored procedure* de query, de modo que pudiéramos tratar los resultados con un ResultSet para irlos agregando a la tabla fila a fila sin complicaciones. Procedimiento de java:

```
private void ferCerca() {  
    // agafem els valors de la vista  
    String title = finestraPrincipal.getJtfMovTitle();  
    String genre = finestraPrincipal.getJtfGenre();  
    String actor = finestraPrincipal.getJtfActor();  
    String director = finestraPrincipal.getJtfDirector();  
    String country = finestraPrincipal.getJtfCountry();  
    String order_by_what = finestraPrincipal.getOrderWhat();  
    String order_by_how = finestraPrincipal.getOrderHow();  
  
    switch (order_by_what) {  
        case "Movie title":  
            order_by_what = "title";  
            break;  
        case "Genre":  
            order_by_what = "description";  
            break;  
        case "Director":  
            order_by_what = "director";  
            break;  
        case "Country":  
            order_by_what = "country";  
            break;  
        case "IMDB score":  
            order_by_what = "imdb_score";  
            break;  
    }  
  
    switch (order_by_how) { // això és perquè a l'hora d'omplir la taula, els orders by s'inverteixen.  
        case "ASC":  
            order_by_how = "DESC";  
            break;  
        case "DESC":  
            order_by_how = "ASC";  
            break;  
    }  
}
```

Al llenar las filas de la tabla de la interfaz, los datos recibidos por el *procedure ferCerca()*, se añadían de manera que los resultados se veían ordenados de forma contraria a cómo el usuario lo había seleccionado, de modo que invertimos el *order\_by\_how* antes de llamar al *procedure* para que se insertaran bien en nuestra tabla.

A la hora de hacer la llamada al *stored procedure*, los parámetros que enviamos son todos los de la interfaz introducidos por el usuario y comprobamos mediante if's cada campo que el usuario haya dejado posiblemente vacío.

```

try {
    JOptionPane.showMessageDialog(finestraPrincipal, (message: "Realitzant cerca, pot trigar uns minuts (~5)...");
    System.out.println("call fesCercaUltimate('" + title + "', '" + genre + "', '" + actor + "', '" + director + "', '" + country + "', '" + order_by_what + "', '" + order_by_how + "')");
    ResultSet dades = localCon.selectQuery("call fesCercaUltimate('" + title + "', '" + genre + "', '" + actor + "', '" + director + "', '" + country + "', '" + order_by_what + "', '" + order_by_how + "')");
    int i = 0;
    while (dades.next()) {
        finestraPrincipal.addResultsRow(new String[]{dades.getString(columnLabel: "title"),
            dades.getString(columnLabel: "genre"),
            dades.getString(columnLabel: "director"),
            dades.getString(columnLabel: "country"),
            dades.getDouble(columnLabel: "imdb_score")+""});
    }
    JOptionPane.showMessageDialog(finestraPrincipal, (message: "Ha terminat la cerca!");
} catch (Exception e) {
    System.out.println("Error de SQL.");
    JOptionPane.showMessageDialog(finestraPrincipal, (message: "La query no ha retornat cap resultat!", title: "WARNING", JOptionPane.WARNING_MESSAGE);
}
]

```

Esta es la llamada al procedimiento y la inserción en la tabla de resultados.

### .sql de creación de los procedimientos de gestión:

```
use new_movies2; -- taula local
```

```

delimiter $$
drop procedure if exists afegeixUsuari$$
create procedure afegeixUsuari(in name varchar(255), in pass varchar(255))
begin
    set @aux = concat('create user \'', name, '\' identified by \'', pass,
    '\';');
    set @grants = concat('grant all on *.* to \'', name, '\' identified by \'',
    pass, '\';');
    -- select @aux;
    -- select @grants;
    prepare stmt1 from @aux;
    execute stmt1;
    deallocate prepare stmt1;
    prepare stmt2 from @grants;
    execute stmt2;
    deallocate prepare stmt2;
end $$
delimiter ;

```

```

delimiter $$
drop procedure if exists fesCercaUltimate$$
create procedure fesCercaUltimate(in title varchar(255), in genre varchar(255),
in actor varchar(255), in director varchar(255), in country varchar(255), in
ob_what varchar(255), in ob_how varchar(255))
begin
    declare cerca text;
    set cerca = concat('(select m.title as title, g.description as genre,
p.name as director, m.country as country, m.imdb_score as imdb_score from movie
as m, genre_movie as gm, genre as g, person as p, director as d, person as a,
actor_movie as am where m.id_movie = gm.id_movie and gm.id_genre = g.id_genre
and a.id_person = am.id_actor and am.id_movie = m.id_movie and p.id_person =
d.id_director and d.id_director = m.id_director)');
    if genre not like '' then
        set cerca = concat(cerca, ' and genre like \'%', genre, '%\');
    end if;
    if title not like '' then
        set cerca = concat(cerca, ' and title like \'%', title, '%\');
    end if;
    if actor not like '' then
        set cerca = concat(cerca, ' and actor like \'%', actor, '%\');
    end if;
end $$

```



```

        if director not like '' then
            set cerca = concat(cerca, ' and director like \'%',
director,'%\');
        end if;
        if country not like '' then
            set cerca = concat(cerca, ' and country like \'%', country,'%\');
        end if;

        set cerca = concat(cerca, ') union (select m.title as title, \'\'', \'\'',
m.country as country, m.imdb_score as imdb_score from movie as m where
(m.id_director not in (select d1.id_director from director as d1) or
(m.id_movie not in(select am1.id_movie from actor_movie as am1) or m.id_movie
not in (select gm1.id_movie from genre_movie as gm1)))');
        if title not like '' then
            set cerca = concat(cerca, ' and title like \'%', title,'%\');
        end if;
        if country not like '' then
            set cerca = concat(cerca, ' and country like \'%', country,'%\');
        end if;
        set cerca = concat(cerca,') order by title asc limit 10;');

set @aux = cerca;
prepare stmt1 from @aux;
execute stmt1;
deallocate prepare stmt1;

end $$
delimiter ;

```

El procedimiento de búsqueda realiza una query que trata las relaciones con las tablas que tienen todos los datos, y un **unión para que las relaciones de la tabla no exploten si alguno de los campos no existe (como ocurrió en la entrevista con la película ‘Base de dades’, cuyos actores (inexistentes) no se encontraban en la tabla de actores y, por tanto, no podía relacionar las tablas).**

*Seguro que hay una manera mucho más óptima de realizar la búsqueda especialmente, pero ya comprobamos el funcionamiento en la entrevista.*

## 8. Modificaciones en la interfaz proporcionada

La interfaz que se nos ha proporcionado no ha sufrido ninguna modificación visual.

## 9. Dedicación

Aproximadamente hemos dedicado toda la semana antes de la entrega, que llega a ser unas 15h, esto para realizar la primera fase. Una vez entregada la primera fase, tardamos aproximadamente unas 8 horas en realizar la fase 2 de primeras y unas 3 más para arreglar errores y terminar la entrevista.

## 10. Conclusiones

Las conclusiones que hemos podido llegar con esta práctica 2 es que ya sabemos cómo conectarnos desde una interfaz en java a un servidor donde luego, en este mismo servidor, se encuentra la base de datos en la cual estamos interesados. Ahora tenemos muchos más conocimientos en relación con el lenguaje de Java y las comandas que se utilizan para conectarse y obtener información de las tablas de la base de datos, además de entender cómo podemos utilizar las comandas que conocemos, tanto de bases de datos como las de la consola de comandos del sistema para exportar e importar información.

Utilizar la interfaz proporcionada ha sido un poco incómodo al principio, pero mucho más intuitiva a medida que avanzábamos en la entrega, ya que teníamos más claro cómo utilizar todas las herramientas que nos proporcionaban (especialmente la tabla de resultados y el método de añadir una fila), y ha sido una experiencia interesante.

También una buena manera de reaprovechar la fase 1. De no ser por la base de datos remota caída, evidentemente.

## 11. Bibliografía

<https://es.wikipedia.org/wiki/Wikipedia:Portada>

<https://www.techopedia.com/definition/24436/online-transaction-processing-oltp>

<https://www.powerdata.es/data-warehouse>