

# Presentación

## SQL DESDE CERO



# Objetivos del curso. ¿Qué aprenderemos?

El objetivo principal del curso es aprender a diseñar y realizar consultas, desde las más elementales a las más complejas, para obtener datos de cualquier base de datos, de forma eficiente y precisa mediante el lenguaje SQL.

- ▶ Dominarás la sintaxis del lenguaje de consultas y como se estructuran los sub-lenguajes que lo forman.
- ▶ Aprenderás como obtener información de cualquier base de datos de una forma precisa y eficiente mediante sentencias SQL.
- ▶ Conocerás como instalar y configurar tu base de datos relacional desde cero

# Temario

- ▶ SECCIÓN 1: Introducción
- ▶ SECCIÓN 2: Preparación del entorno de trabajo
- ▶ SECCIÓN 3: Lenguaje de Manipulación de datos (DML) 1
- ▶ SECCIÓN 4: Lenguaje de Manipulación de datos (DML) 2

## Metodología, prácticas

- ▶ Instalaremos el entorno de trabajo completo: servidor web, BBDD y cliente SQL.
- ▶ Cargaremos la base de datos de ejemplo
- ▶ Realizaremos consultas sobre la base de datos, incrementando la complejidad exponencialmente.

# Sección 1

# INTRODUCCIÓN



A thick, light blue diagonal line runs from the top right corner towards the bottom left, separating the white background on the left from the solid blue background on the right.

1.

**INTRODUCCIÓN A  
LAS BBDD  
RELACIONALES**

# Una Base de Datos (BD) es...

Algunas  
definiciones

“...una colección de ficheros relacionados”

“...un conjunto de diferentes tipos de datos organizados e interrelacionados”

“...un conjunto de datos de diferentes ámbitos, organizado sistemáticamente, es decir, siguiendo ciertas reglas.”



“

Una base de datos relacional (**BDR**) es un tipo de base de datos (**BD**) que cumple con el modelo relacional (el modelo más utilizado actualmente para implementar las BD ya planificadas).

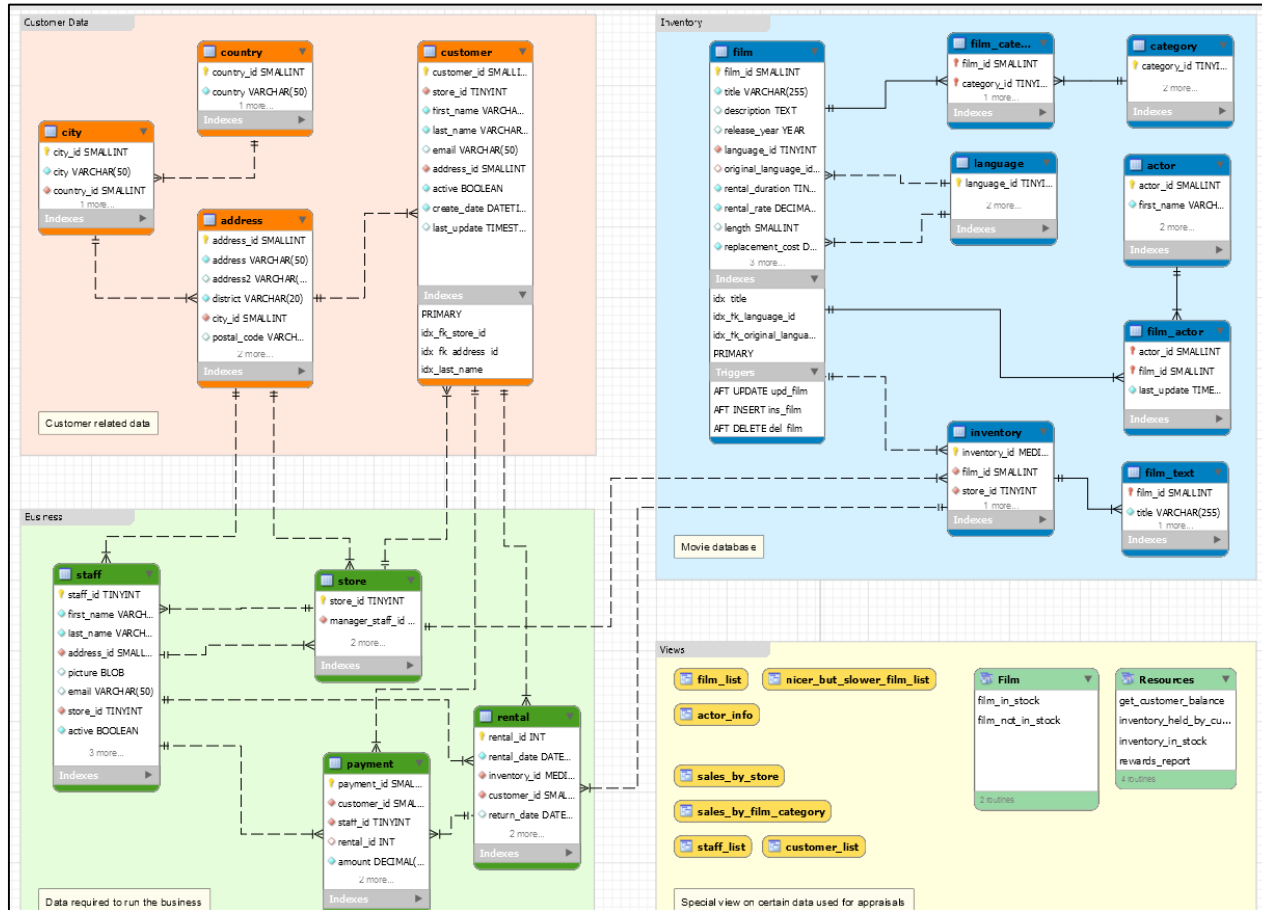


# Principales objetos de una Base de datos:

- ▶ Tablas:
  - ▶ Campos (columnas) – Tipo de datos
  - ▶ Registros (filas)
- ▶ Relaciones
  - ▶ 1 - 1
  - ▶ 1 -  $\infty$
  - ▶  $\infty$  -  $\infty$
- ▶ Índices, claves y restricciones (constraints)
- ▶ Consultas / Vistas
- ▶ Formularios
- ▶ Otros:
  - ▶ Procedimientos / Triggers (disparadores)

# Ejemplo Base de Datos

Esquema  
BBDD **Sakila**



# 2.

**RDBMS (SGBDR)**  
**MÁS COMUNES**

SGBDR  
(RDBMS)

## RDBMS (SGBDR) más comunes

- MS SQL Server
- Oracle

- Access (MS Office)

Propietarias

- PostgreSQL
- MySQL
- MariaDB

- Base (libre / OpenOffice)

Open Source



# 3.

## EL LENGUAJE SQL

SQL: Structured  
Query Language

# SQL Structured Query Language

## (Lenguaje Estructurado de Consultas)

- ▶ Primer lenguaje para implementar el modelo relacional de Codd (1970)
- ▶ Desarrollado por IBM – Primera versión 1974
- ▶ 1986: SQL-86 Pasa a ser un estándar de la ANSI
- ▶ Última revisión SQL: 2016
- ▶ Teóricamente un estándar, pero los fabricantes han introducido funciones y variaciones propias. En la práctica se requieran ciertos ajustes.

# 4.

## SUB-LENGUAJES SQL

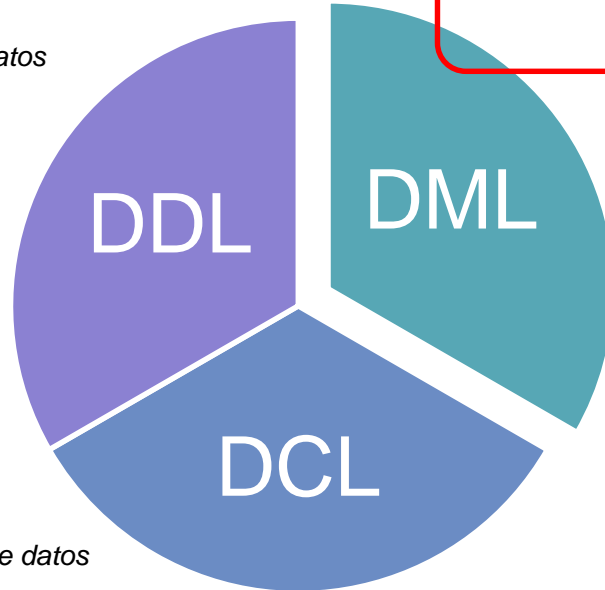
DDL / DML  
DCL (TCL)

# Sub lenguajes SQL

DDL  
DML  
DCL

Data Definition Language  
*Definición de datos*

- CREATE
- ALTER
- DROP



Data Manipulation Language  
*Manipulación de datos*

- SELECT
- INSERT
- UPDATE
- DELETE

Data Control Language  
*Control de datos*

- GRANT
- REVOKE

TCL: Transaction Control Language

- BEGIN
- COMMIT
- ROLLBACK



## Sección 2

# PREPARACIÓN DEL ENTORNO DE TRABAJO



# 1.

## DESCRIPCIÓN ENTORNO DE TRABAJO

XAMPP

“

**XAMPP** es una distribución de Apache completamente gratuita y fácil de instalar que contiene:

**MariaDB, PHP y Perl.**



# ELEMENTOS ENTORNO XAMPP

- ▶ Multi sistema operativo: X
  - ▶ Servidor HTTP Apache
  - ▶ Motor de BBDD MariaDB (anteriormente MySQL)
  - ▶ Lenguaje PHP + Lenguaje Pearl
  - ▶ Otros: FTP Filezilla, Tomcat...y phpMyAdmin
- 
- Alternativa: WAMPP (Exclusivo Windows)



# CLIENTE SQL HeidiSQL



- ▶ Hereda de **MySQL-Front**
- ▶ Conecta con **MS SQL server, My SQL, MariaDB, PotgreeSQL**, etc...
- ▶ **Open Source**
- ▶ Existe la versión **Portable**

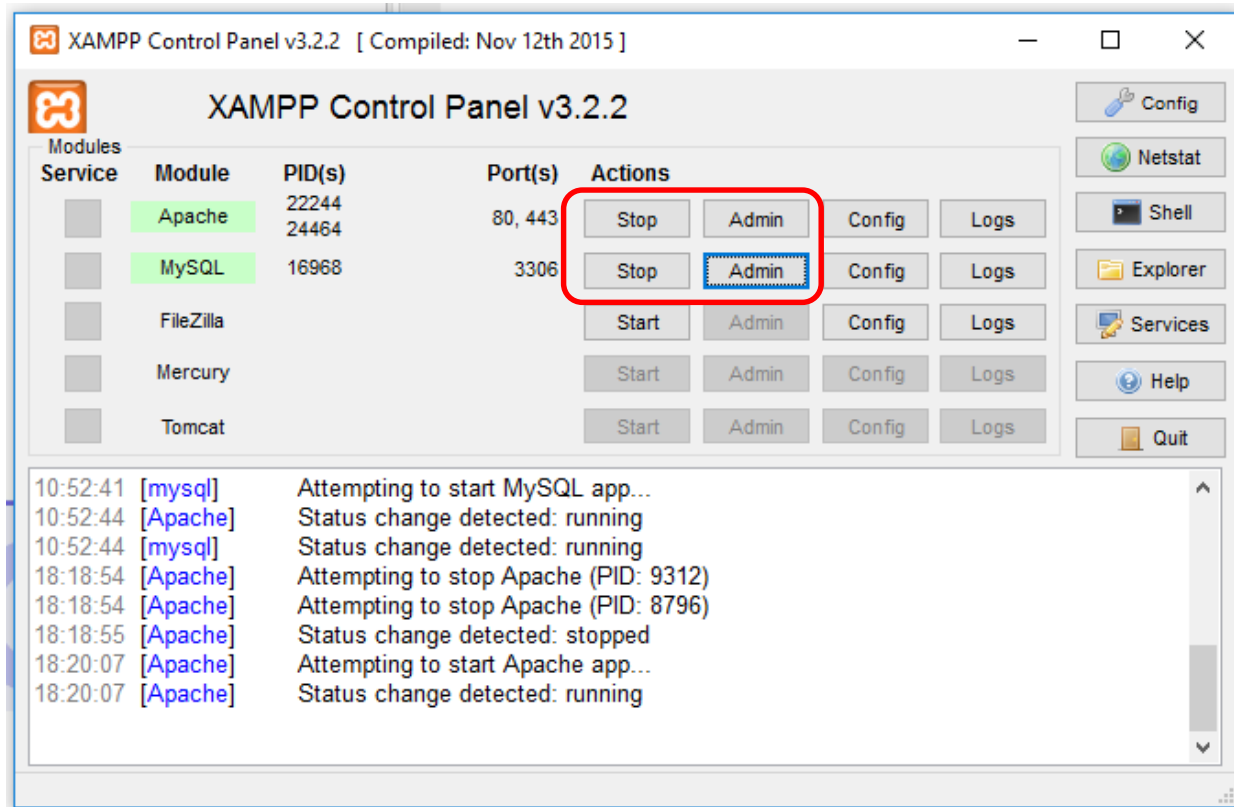
<https://www.heidisql.com/>

# 2.

## INSTALACIÓN ENTORNO DE TRABAJO

XAMPP  
HeidiSQL

# Panel control XAMPP



XAMPP Control Panel v3.2.2 [ Compiled: Nov 12th 2015 ]

**XAMPP Control Panel v3.2.2**

Modules

Service	Module	PID(s)	Port(s)	Actions
<input type="checkbox"/>	Apache	22244 24464	80, 443	Stop Admin Config Logs
<input type="checkbox"/>	MySQL	16968	3306	Stop Admin Config Logs
<input type="checkbox"/>	FileZilla			Start Admin Config Logs
<input type="checkbox"/>	Mercury			Start Admin Config Logs
<input type="checkbox"/>	Tomcat			Start Admin Config Logs

Config Netstat Shell Explorer Services Help Quit

10:52:41 [mysql] Attempting to start MySQL app...  
10:52:44 [Apache] Status change detected: running  
10:52:44 [mysql] Status change detected: running  
18:18:54 [Apache] Attempting to stop Apache (PID: 9312)  
18:18:54 [Apache] Attempting to stop Apache (PID: 8796)  
18:18:55 [Apache] Status change detected: stopped  
18:20:07 [Apache] Attempting to start Apache app...  
18:20:07 [Apache] Status change detected: running

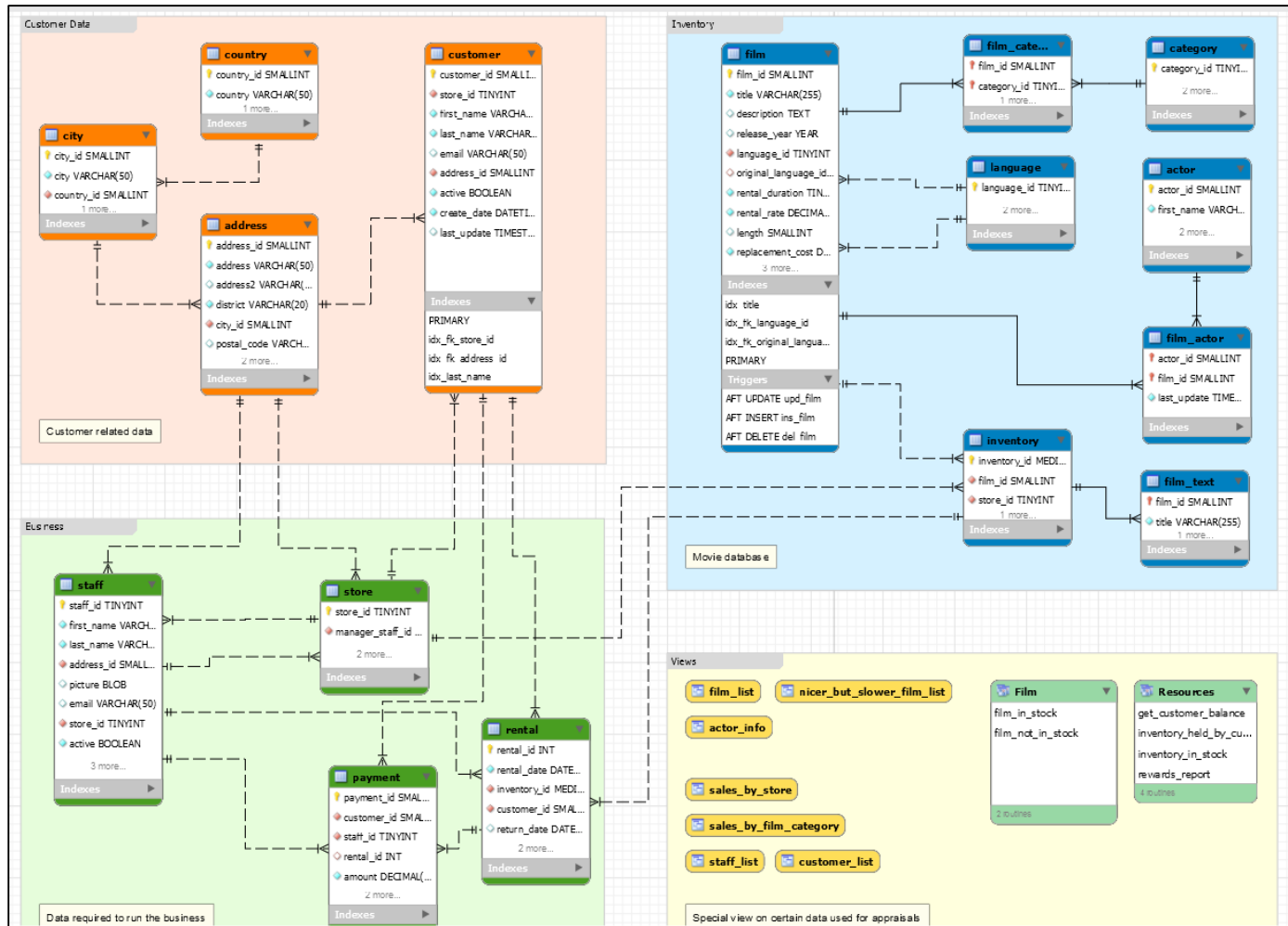
# 3.

## INSTALACIÓN BBDD EJEMPLO

Sakila DB



# ESQUEMA BBDD SAKILA



# DESCARGAR E INSTALAR LA BBDD SAKILA

- ▶ MySQL doc: <https://dev.mysql.com/doc/index-other.html>
  - ▶ Sakila: [downloads.mysql.com/docs/sakila-db.zip](https://dev.mysql.com/downloads/mysql/docs/sakila-db.zip)
  - ▶ Como instalar:  
<https://dev.mysql.com/doc/sakila/en/sakila-installation.html>

# Resumen instalación:

To install the Sakila sample database, follow these steps:

1. Extract the installation archive to a temporary location such as `C:\temp\` or `/tmp/`. When you unpack the archive, it creates a directory named `sakila-db` that contains the `sakila-schema.sql` and `sakila-data.sql` files.

2. Connect to the MySQL server using the **mysql** command-line client with the following command:

```
1 shell> mysql -u root -p
```

Enter your password when prompted. A non-root account can be used as long as the account has privileges to create new databases.

3. Execute the `sakila-schema.sql` script to create the database structure by using the following command:

```
1 mysql> SOURCE C:/temp/sakila-db/sakila-schema.sql;
```

Replace `C:/temp/sakila-db` with the path to the `sakila-schema.sql` file on your system.

## Note

On Windows, use slashes, rather than backslashes, when executing the `SOURCE` command.

4. Execute the `sakila-data.sql` script to populate the database structure with the following command:

```
1 mysql> SOURCE C:/temp/sakila-db/sakila-data.sql;
```

Replace `C:/temp/sakila-db` with the path to the `sakila-data.sql` file on your system.



En nuestro caso:  
`C:\xampp\mysql\bin`

# Sección 3

## LENGUAJE DE MANIPULACIÓN DE DATOS DML (1)



1.

# SINTAXIS BÁSICA SQL

Comando  
SELECT

# PRINCIPIOS SINTAXIS SQL

- ▶ Estructurado (**S**tructured...)
- ▶ Normalizado (*casi* estándar)
  - ▶ Comandos (SELECT, INSERT...)
  - ▶ Cláusulas (FROM, WHERE...)
  - ▶ Operadores (DISTINCT, LIKE...)

# SINTAXIS SQL

- ▶ No sensitivo a mayúsculas (aunque hay convenio):
  - ▶ **SELECT = select = Select, etc...**
- ▶ Usar **;** para terminar cada consulta
- ▶ Se puede estructurar libremente (filas)

**SELECT \* FROM actor;**

**SELECT \*  
FROM actor  
;**

- ▶ Comentarios (puede variar en cada SGBDR):  
**/\* esto son comentarios, bla, bla \*/**

# Consulta Básica en SQL

```
SELECT *  
FROM actor  
;
```

## Selección de columnas:

```
SELECT last_name, first_name  
FROM actor  
;
```



## Registros únicos: **DISTINCT**

```
SELECT DISTINCT last_name  
FROM actor  
;
```

## Aquí, afectaría en algo?

```
SELECT DISTINCT *  
FROM actor  
;
```

## Alias de columna: **AS**

```
SELECT title AS 'film name', description AS sinopsis  
FROM film  
;
```

# 2.

## RESTRICCIÓN Y ORDENACIÓN

Cláusulas  
WHERE  
ORDER BY

## Restricción (*filtrar*): Cláusula **WHERE**

```
SELECT last_name, first_name  
FROM actor  
WHERE last_name = 'ALLEN'  
;
```

## Operandos de comparación:

= / > / < / >= / <= / <> (!= en algunos DBMS)

SELECT \*

FROM film

**WHERE** length >= 120;

## Ordenar: Cláusula **ORDER BY + [ASC] / DESC**

```
SELECT *  
FROM film  
ORDER BY title DESC  
;
```

## Seleccionar, filtrar y ordenar: **WHERE + ORDER BY**

```
SELECT last_name, first_name  
FROM actor  
WHERE last_name = 'ALLEN'  
ORDER BY title DESC  
;
```

# 3.

## OPERADORES LÓGICOS

Uso de:  
AND, OR, NOT  
LIKE, IN, BETWEEN



## Condiciones múltiples: **AND / OR**

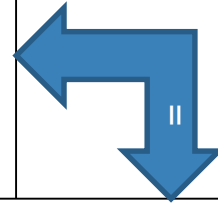
```
SELECT title, rental_rate  
FROM film  
WHERE length < 90 AND rental_rate <=3  
;
```

## Filtro complejo: **AND / OR**

```
SELECT *  
FROM customer  
WHERE (first_name = 'JOHN' OR last_name = 'SMITH')  
AND active = 1  
;
```

## Negación, uso de **NOT**

```
SELECT last_name, first_name  
FROM actor  
WHERE NOT last_name = 'ALLEN'  
;
```



```
SELECT last_name, first_name  
FROM actor  
WHERE last_name <> 'ALLEN'  
;
```

## Operador **LIKE** + *wildcards* (%)

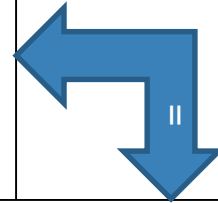
```
SELECT *  
FROM film  
WHERE title LIKE '%LOVE%'  
;
```

<https://dev.mysql.com/doc/refman/5.7/en/pattern-matching.html>

<https://mariadb.com/kb/en/library/like/>

## Operador **IN** vs igualdad (=)

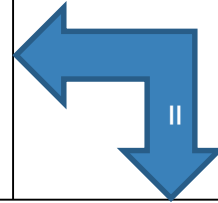
```
SELECT *  
FROM actor  
WHERE last_name IN  
( 'ALLEN', 'MCCONAUUGHY', 'PALTROW' )  
;
```



```
SELECT *  
FROM actor  
WHERE last_name = 'ALLEN'  
OR last_name = 'MCCONAUUGHY'  
OR last_name = 'PALTROW'
```

## Operator **BETWEEN** vs $\geq x$ **AND** $\leq y$

```
SELECT *  
FROM film  
WHERE rental_rate BETWEEN 0.5 AND 2.5  
;
```



```
SELECT *  
FROM film  
WHERE rental_rate  $\geq$  0.5  
AND rental_rate  $\leq$  2.5  
;
```

Ejemplo consulta completa:  
Seleccionar columnas / Alias  
filtrar (combinado)  
ordenar (múltiple):

```
SELECT title AS 'film name', rental_rate AS price  
FROM film  
WHERE (length BETWEEN 90 AND 120)  
AND (rental_rate <=3 OR replacement_cost <15)  
ORDER BY price DESC, title  
;
```

# Sección 4

## LENGUAJE DE MANIPULACIÓN DE DATOS DML (2)





1.

## CONSULTAS DE UNIÓN

Comando  
UNION

## Comando **UNION [ALL]**:

```
SELECT last_name, first_name  
FROM customer
```

**UNION**

```
SELECT last_name, first_name  
FROM actor
```



2.

**SUBCONSULTAS**

SUBQUERIES

## Ejemplo subconsultas

# SUBQUERIES

```
SELECT lastName, firstName  
FROM employees  
WHERE officeCode IN (SELECT officeCode  
                     FROM offices  
                     WHERE country = 'USA')  
;
```

# 3.

## **FUNCIONES ESCALARES**

CONCAT, LEFT, LCASE, TRIM...

## Ejemplo función **LCASE**

```
SELECT LCASE(title)
FROM film
;
```

<https://mariadb.com/kb/en/library/built-in-functions/>

# 4.

## AGRUPAMIENTO

### 4.1 FUNCIONES

### AGREGADO

COUNT, MAX, MIN, SUM, AVG

## Ejemplo función **COUNT()**

```
SELECT count(*) AS 'Clientes Activos'  
FROM customer  
WHERE active = 1  
;
```

<https://mariadb.com/kb/en/library/aggregate-functions/>



## Ejemplo función **MAX()**

```
SELECT max(rental_rate) AS 'Precio máximo'  
FROM film  
;
```

# 4.

## AGRUPAMIENTO

### 4.2 GROUP BY / HAVING

GROUP BY / HAVING

## Ejemplos Agrupación: **GROUP BY**

```
SELECT count(film_id), rental_rate  
FROM film  
GROUP BY rental_rate  
;
```

```
SELECT count(film_id), rental_rate  
FROM film  
WHERE replacement_cost <= 20  
GROUP BY rental_rate  
ORDER BY rental_rate DESC  
;
```

# IMPORTANTE

## GROUP BY

Si requerimos de agrupación, no podemos exigir detalle y viceversa

## Agrupación: **GROUP BY / HAVING**

```
SELECT count(film_id), replacement_cost
FROM film
WHERE replacement_cost <= 20
GROUP BY replacement_cost
HAVING count(film_id) >= 50
ORDER BY replacement_cost
;
```

# IMPORTANTE

## HAVING

Hay quien llama a HAVING 'El Where dentro del Where' ;)

# 5.

## CONSULTAS SOBRE MÁS DE UNA TABLA

[INNER] JOIN  
LEFT / RIGHT / FULL [OUTER] JOIN

# TIPOS DE RELACIÓN





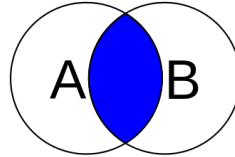
## Consultas multi-tabla mediante **WHERE (sin Join)** Alias de tabla: **[AS]**

```
SELECT first_name, last_name, address  
FROM customer c, address a  
WHERE c.address_id = a.address_id  
;
```

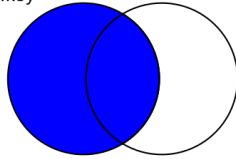
# TIPOS DE JOIN

## SQL JOINS

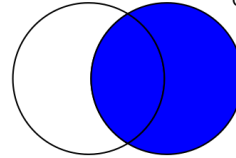
SELECT <fields>  
FROM TableA A  
**INNER JOIN** TableB B  
ON A.key = B.key



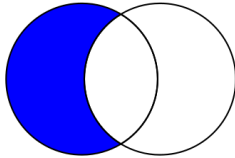
SELECT <fields>  
FROM TableA A  
**LEFT JOIN** TableB B  
ON A.key = B.key



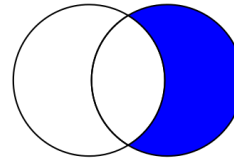
SELECT <fields>  
FROM TableA A  
**RIGHT JOIN** TableB B  
ON A.key = B.key



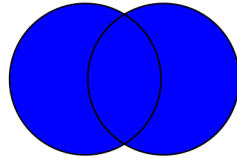
SELECT <fields>  
FROM TableA A  
**LEFT JOIN** TableB B  
ON A.key = B.key  
WHERE B.key IS NULL



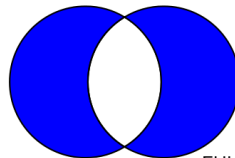
SELECT <fields>  
FROM TableA A  
**RIGHT JOIN** TableB B  
ON A.key = B.key  
WHERE A.key IS NULL



SELECT <fields>  
FROM TableA A  
**FULL OUTER JOIN** TableB B  
ON A.key = B.key



SELECT <fields>  
FROM TableA A  
**FULL OUTER JOIN** TableB B  
ON A.key = B.key  
WHERE A.key IS NULL  
OR B.key IS NULL



## Ejemplo consulta multi-tabla: [INNER] JOIN

```
SELECT first_name, last_name, address  
FROM customer c JOIN address a  
ON c.address_id = a.address_id  
;
```