{Introducción a SQL}

[Clase 1]

Clase 1.

Contenidos principales:

- Introducción: alcances y limitaciones del curso.
- ¿Qué son las bases de datos relacionales?.
- Introducción a MySQL.
- Integrated Development Environment (IDE) Workbench.
- Comandos básicos de MySQL: conectarse a una BD.
- Conocer la base “Sakila”.
- Primeros pasos en SQL: ANSI, DML / DDL.
- Sentencia **SELECT + FROM** y cláusula **WHERE**.
- Funciones sobre la sentencia **SELECT**.
- Ordenamiento de los datos usando **ORDER BY.**
- Sentencia **CREATE TABLE**.

Luego de esta clase serás capaz de:

- Conectarte a una base de datos MySQL y ejecutar comandos básicos
- Iniciar una instancia de base datos usando Workbench.
- Conocer el esquema de datos “Sakila” y las distintas tablas que lo componen.
- Filtrar datos de las distintas tablas.
- Aplicar funciones que modifiquen la forma de ver los datos.
- Calcular cantidades simples, promedios, máximos y mínimos.
- Ordenar los resultados.
- Crear una tabla con resultados obtenidos previamente.

¿Qué son las bases de datos relacionales?

“Una base de datos relacional es una colección de elementos de datos organizados en un conjunto de tablas formalmente descritas desde la que se puede acceder a los datos o volver a montarlos de muchas maneras diferentes sin tener que reorganizar las tablas de la base. La base de datos relacional fue inventada por E.F. Codd en IBM en 1970.”

Sakila rental store.

Sakila es una base de datos creada por MySQL con la finalidad de conocer las distintas funcionalidades y alcances del motor de base de datos.

Simula ser un videoclub de DVD's como los de antes.

Se compone de varias tablas en las cuales podemos encontrar información sobre las películas disponibles, los clientes, los alquileres, el personal de trabajo, etc.

Para más información: https://dev.mysql.com/doc/sakila/en/

Introducción a MySQL

MySQL es un motor de bases de datos relacional actualmente desarrollado por Oracle Corporation. Es uno de los más populares del mundo.

> *"En aplicaciones web hay baja concurrencia en la modificación de datos y en cambio el entorno es intensivo en lectura de datos, lo que hace a MySQL ideal para este tipo de aplicaciones"*

Funciona tanto en Windows como en Linux.

Otras bases de datos del estilo son: Oracle, Microsoft SQL Server, Teradata, SQL Lite, Postgre SQL.

Comandos básicos de MySQL:

- Para ver todos los distintos esquemas de datos:

```
SHOW DATABASES; o SHOW SCHEMAS;
```

- Para seleccionar un esquema de trabajo:

```
USE <nombre del esquema>;
```

- Para ver las tablas que contiene un esquema:

```
SHOW TABLES;
```

- Si queremos saber cómo es la estructura de datos de una tabla específica:

```
DESCRIBE <nombre de la tabla>;
```

Structured Query Language.

SQL significa **lenguaje estructurado de consultas.**

Es un lenguaje que permite ejecutar sentencias de código llamadas **"consultas"** (o "**querys**" en inglés) sobre una base de datos relacional para obtener y manipular datos.

"SQL pasó a ser el estándar del Instituto Nacional Estadounidense de Estándares (ANSI) en 1986 y de la Organización Internacional de Normalización (ISO) en 1987. Desde entonces, el estándar ha sido revisado para incluir más características. A pesar de la existencia de ambos estándares, la mayoría de los códigos SQL no son completamente portables entre sistemas de bases de datos diferentes sin ajustes."

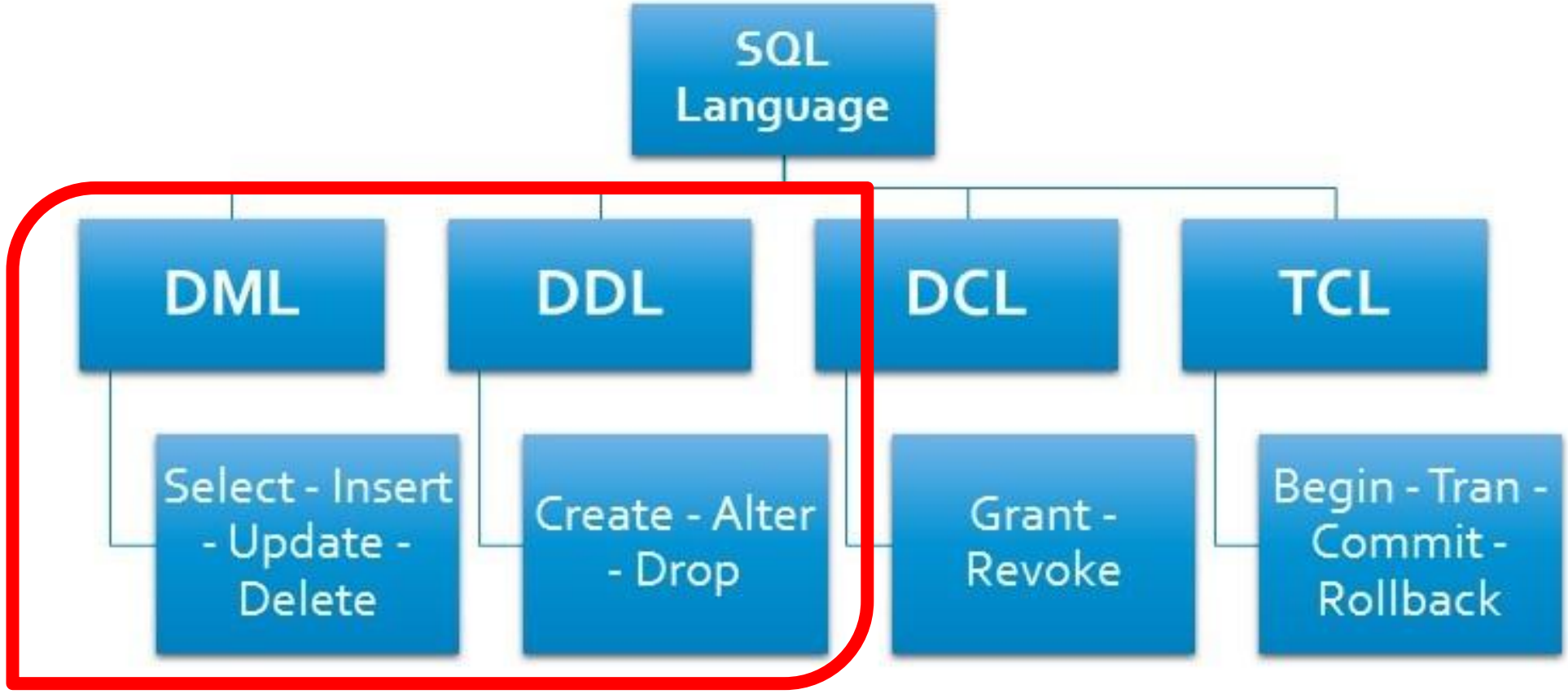
Las principales sentencias se dividen en dos grandes grupos: **DDL (Data Definition Language) y DML (Data Manipulation Language).**

DDL: es el lenguage de **definición** de datos. Permite crear tablas, modificarlas, borrarlas. También permite crear otro tipo de objetos como: vistas, disparadores, índices, etc.

DML: es el lenguage de **manipulación** de datos. Sirve para hacer consultas a la base de datos, insertar, modificar o borrar registros de una tabla.

Es un lenguaje "case insensitive", lo que significa que importa si escribimos en minúscula o mayúscula las consultas que ejecutemos. No obstante sí podría importar para los datos almacenados.

Tipos de operaciones en SQL:



Sentencia **SELECT + FROM**

Permite seleccionar las columnas de una tabla específica de nuestra BD. Es decir, muestra todos los registros de la tabla, pero solo las columnas seleccionadas.

Sintaxis:

```
SELECT <columnas separadas por coma>
FROM <nombre de la tabla>;
```

Si queremos obtener todas las columnas de una tabla usamos un asterisco "*" en lugar de poner los nombres de cada columna.

```
SELECT *
FROM <nombre de la tabla>;
```

CLIENTES					
ID	NOMBRE	APELLIDO	EDAD	SEXO	TELEFONO
10	JUAN	RODRIGUEZ	25	M	48259944
12	MARIA	JUAREZ	32	F	48222147
35	ANA	FERRARI	18	F	49621475
47	MARCOS	PALMA	10	M	4565745
61	AUGUSTO	FLORENTIN	41	M	4787314

ID	EDAD	SEXO
10	25	M
12	32	F
35	18	F
47	10	M
61	41	M

```
SELECT id,edad,sexo
FROM clientes;
```

Filtrando resultados con la cláusula **WHERE**:

Si queremos filtrar de una tabla los registros que cumplan con una determinada condición debemos usar la cláusula **WHERE** dentro de una sentencia SQL.

Para eso debemos saber cómo expresar una **condición** que sea **verdadera** o **falsa**. Por ejemplo: quiero quedarme con todos los registros de una tabla que cumplan que el campo EDAD sea mayor a 18 años. Los registros que cumplan esta condición (ie. registros con edades mayores a 18 años) se mostrarán dentro de los resultados.

Sintaxis:

```
SELECT <columnas seperadas por coma>
FROM   <nombre de la tabla>
WHERE  <condición verdadera o falsa>;
```

STOCK_CONCESIONARIA				
MARCA	MODELO	MOTOR	COLOR	PRECIO
VOLKSWAGEN	GOL	1,4	ROJO	155.000
PEUGEOT	207	1,4	BLANCO	162.000
PEUGEOT	208	1,8	GRIS	221.000
FIAT	PUNTO	1,6	GRIS	175.000
FORD	FOCUS	2	AZUL	296.000
VOLKSWAGEN	VENTO	2	NEGRO	312.000
FORD	FIESTA	1,4	GRIS	175.000
VOLKSWAGEN	AMAROK	3	BLANCO	788.000

```
SELECT *
FROM stock_concesionaria
WHERE color='GRIS';
```

MARCA	MODELO	MOTOR	COLOR	PRECIO
PEUGEOT	208	1,8	GRIS	221.000
FIAT	PUNTO	1,6	GRIS	175.000
FORD	FIESTA	1,4	GRIS	175.000

```
SELECT *
FROM stock_concesionaria
WHERE precio>300000;
```

MARCA	MODELO	MOTOR	COLOR	PRECIO
VOLKSWAGEN	VENTO	2	NEGRO	312.000
VOLKSWAGEN	AMAROK	3	BLANCO	788.000

STOCK_CONCESIONARIA				
MARCA	MODELO	MOTOR	COLOR	PRECIO
VOLKSWAGEN	GOL	1,4	ROJO	155.000
PEUGEOT	207	1,4	BLANCO	162.000
PEUGEOT	208	1,8	GRIS	221.000
FIAT	PUNTO	1,6	GRIS	175.000
FORD	FOCUS	2	AZUL	296.000
VOLKSWAGEN	VENTO	2	NEGRO	312.000
FORD	FIESTA	1,4	GRIS	175.000
VOLKSWAGEN	AMAROK	3	BLANCO	788.000

MARCA	MODELO	COLOR
FORD	FOCUS	AZUL
PEUGEOT	208	GRIS
FORD	FIESTA	GRIS
PEUGEOT	207	BLANCO

```
SELECT marca,modelo,color
 FROM stock_concesionaria
WHERE precio BETWEEN 160000 AND 300000
AND marca IN ('FORD','PEUGEOT');
```

Tipos de condiciones para usar en la cláusula **WHERE**:

1) Comparativas:

>	mayor
>=	mayor o igual
<	menor
<=	mejor o igual
=	igual

```
SELECT rental_id,amount
FROM payment
WHERE customer_id=599;
```

```
SELECT title,description
FROM nicer_but_slower_film_list
WHERE price<=3;
```

```
SELECT *
FROM nicer_but_slower_film_list
WHERE category ='Family';
```

2) Rangos, inclusión, exclusión y patrones de búsqueda.

BETWEEN A **AND** B	registros comprendidos entre los valores A y B
IN (A,B,C)	registros que sean iguales a A, B o C
NOT IN (A,B,C)	registros que sean distintos a A, B o C
LIKE (‘expresión’)	para comparar patrones o cadenas de texto

```
select rental_id,return_date from rental
where return_date between '2005-05-26' and
'2005-05-27';
```

```
select *
from sales_by_film_category
where category in ('Drama','Comedy');
```

```
select *
from sales_by_film_category
where category not in ('Drama','Comedy');
```

```
select customer_id,first_name,last_name
from customer
where first_name like '%John%';
```

¿Qué pasa si quiero filtrar varias condiciones al mismo tiempo?

3) Operaciones lógicas:

AND	conjunción	**select** film_id,title, rental_rate,replacement_cost **from** film **where** rental_rate>2 **AND** replacement_cost **between** 20 **and** 30;
OR	disyunción	**select** film_id,title, rental_rate,replacement_cost **from** film **where** rental_rate>2 **OR** replacement_cost **between** 20 **and** 30;
NOT	negación	**select** film_id,title, rental_rate,replacement_cost **from** film **where NOT** rental_rate>2;
XOR	disyunción exclusiva	**select** film_id,title, rental_rate,replacement_cost **from** film **where** rental_rate>2 **XOR** replacement_cost **between** 20 **and** 30;

Algunas funciones especiales:

DISTINCT	Permite obtener los valores distintos de una o más columnas.
LOWER / UPPER	Convierte una columna en minúsculas/mayúsculas.
SUBSTR	Sirve para extraer una parte de una cadena de texto.
LENGTH	Calcula el largo de los valores de una columna.
DATE_FORMAT / STR_TO_DATE	Para trabajar con fechas.
COUNT / SUM	Cuenta filas / Suma los valores de una columna.
AVG	Promedio
MAX / MIN	Máximo y mínimo.
SYSDATE	Fecha actual.
+, - ,*, /	Suma, resta, multiplicación, división,
POWER	Potencias.
SQRT	Raíz cuadrada.
CONCAT	Concatena cadenas de texto.

Ordenamiento.

Para ordenar los resultados debemos utilizar la cláusula **ORDER BY** al final de la consulta, seguida de la/s columna/s que queramos ordenar.

También podemos especificar si el ordenamiento es ascendente (**ASC**)o descendente (**DESC**). Por default el orden es ascendente, por lo que no hace falta especificarlo.

Sintaxis:

```
SELECT <columnas a seleccionar>
FROM nombre_de_tabla
ORDER BY <columnas para ordenar>
```

Ejemplos:

```
select * from film ORDER BY title;
select title,rental_duration,rental_rate from film
ORDER BY rental_rate,rental_duration DESC;
```

Otra forma de ordenar los resultados de una colsulta es especificando la posición de las columnas:

```
select title,price,length from nicer_but_slower_film_list ORDER BY 2,3 DESC;
```

Creación de tablas de resultados.

Para crear una tabla se utiliza la sentencia **CREATE TABLE** seguida del nombre que queramos darle y de una sentencia que calcule los resultados a guardar en ella.

Sintaxis:

```
CREATE TABLE <nombre_de_tabla> AS
    SELECT . . . ;
```

Ejemplo:

```
create table ventas_mayores_a_3 as
    select *
    from payment a
    where amount>=3;
```

También podemos crear una tabla sin registros, es decir solo con la estructura de datos y luego cargarle información, pero eso lo veremos más adelante.

Realizar la ejercitación que se encuentra en el archivo: