



Clase 03. SQL

LENGUAJE SQL

***RECUERDA PONER A GRABAR LA
CLASE***





OBJETIVOS DE LA CLASE

- Reconocer e implementar la sentencia SELECT de SQL
- Identificar los operadores en SQL

CRONOGRAMA DEL CURSO

Clase 2



Introducción a las Bases de datos relacionales



BASES DE DATOS
RELACIONALES, CONCEPTOS



DISEÑO DE UN DIAGRAMA
ENTIDAD-RELACIÓN



TEMÁTICAS POSIBLES PARA
EL PROYECTO FINAL

Clase 3



Lenguaje SQL



SELECT - FROM



PRÁCTICAS CON LA
SECUENCIA WHERE

Clase 4



Sublenguajes SQL



DISEÑO DE UN DIAGRAMA E-R
CON LA TEMÁTICA DE
PROYECTO FINAL ELEGIDA



PRÁCTICA SQL CON LAS
VARIANTES DE LOS
SUBLENGUAJES

MAPA DE CONCEPTOS CLASE 3

¡Para
recordar!





¡PARA RECORDAR!

Guía de instalación de **MySQL**:

- Instalación para [Windows](#)
- Instalación para [MAC](#)

Aprendimos la definición de **datos** y conocimos cómo éstos se organizan en **bases de datos relacionales**, las cuales pueden ser representadas a partir de **diagramas entidad-relación**.

¿Cómo gestionamos “operativamente hablando” la información que se encuentra en la misma?



Aquí es donde SQL se convierte en superhéroe.

Mediante una **sintaxis sencilla** y de fácil aprendizaje,
podremos comunicarnos con nuestra base de datos y operar
sobre ella

En pocas palabras: ¡SQL será un gran amigo!



EL LENGUAJE SQL



¿QUÉ ES?

Es un **lenguaje de consultas estructuradas** que responde a las siglas en inglés ***Structured Query Language***.

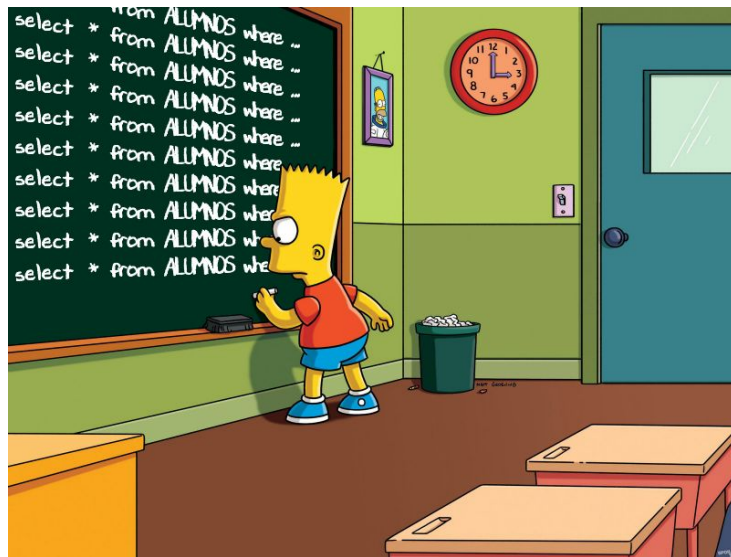
Nos permite acceder y manipular bases de datos.

Es popular por su facilidad de uso y efectividad para convertir grandes volúmenes de datos en información útil.



¿QUÉ PODEMOS HACER?

- **Ejecutar** consultas para recuperar datos.
- **Insertar, modificar y eliminar** registros.
- **Crear** bases de datos, tablas, procedimientos o vistas.
- Establecer **permisos en tablas, procedimientos y vistas**.





EN RESUMEN

Podemos **realizar** sobre la DB **absolutamente todo tipo de operaciones** utilizando sentencias SQL.

Existe una sentencia para todo lo mencionado en la diapositiva anterior y más también. **Lo iremos viendo en detalle** a lo largo del curso.

SENTENCIAS



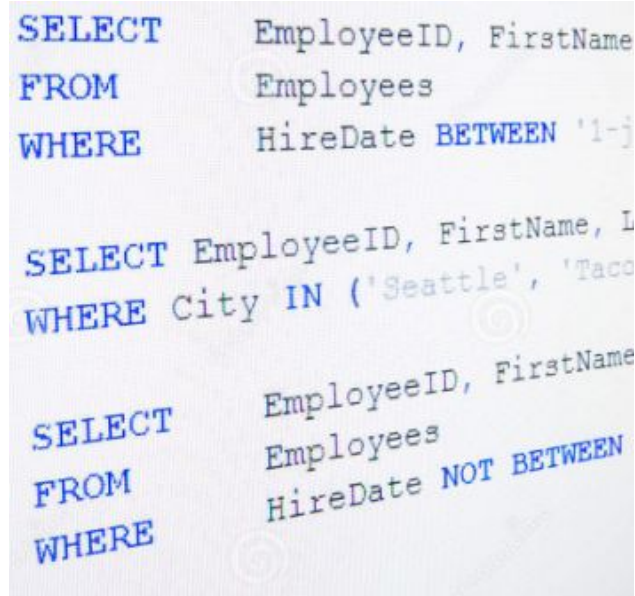
Para concretar la comunicación entre nosotros y las bases de datos, utilizamos **sentencias** SQL.

Y al igual que prácticamente todo lenguaje de programación o ejecución de scripts **su sintaxis es en inglés**, así que, ten presente esto para practicarlo y/o comenzar a dominarlo mejor.

¿QUÉ SON?

Las **sentencias** (también denominadas comandos o cláusulas), son las **palabras reservadas para ejecutar acciones sobre la base de datos.**

Al escribir las instrucciones en un editor, con texto predictivo, veremos que **las sentencias adquieren un color distinto** al resto.



The image shows three SQL queries with keywords highlighted in blue. The first query is: `SELECT EmployeeID, FirstName FROM Employees WHERE HireDate BETWEEN '1-1-2000' AND '1-1-2001'`. The second query is: `SELECT EmployeeID, FirstName, LastName FROM Employees WHERE City IN ('Seattle', 'Tacoma')`. The third query is: `SELECT EmployeeID, FirstName FROM Employees WHERE HireDate NOT BETWEEN '1-1-2000' AND '1-1-2001'`.

OTRA FORMA DE PENSARLO



Mediante las sentencias podemos recuperar y operar sobre ciertos registros, incluso estableciendo condiciones según sus características.

Un buen ejemplo es el juego “*Adivina Quién*”, en el cual vamos “*consultando*” personajes de acuerdo a sus características.

SINTAXIS

CODER HOUSE



La sintaxis SQL

La sintaxis es el **conjunto de reglas que deben seguirse** al escribir el código SQL; para considerarse como **correctas** y así **completar la ejecución** exitosamente.

Sintaxis básica

Las operaciones en SQL siguen una **estructura que describe la operación que deseamos realizar**. Si bien encontraremos consultas muy sencillas y otras más complejas, existen algunos elementos que aparecen con mayor frecuencia:

- 👉 **Acción** (*keyword*): crear, seleccionar, insertar, actualizar, eliminar, etc.
- 👉 **Porción donde operaremos**: puede ser uno o más campos, una o más tablas, o un asterisco (*) para todas.
- 👉 **Tabla/s a la cual/es queremos acceder**: la identificamos escribiendo el/los nombre/s.
- 👉 **Condiciones**: podemos establecer criterios para operar sólo sobre los registros que los cumplan.



VEAMOS ALGUNOS EJEMPLOS

Por ahora pongamos el foco en la estructura; a continuación veremos cada palabra en detalle.

Para consultar determinados campos de una tabla:

```
SELECT id_class, description FROM class;
```

Para consultar todos los campos de una tabla:

```
SELECT * FROM class;
```

ALGUNAS ACLARACIONES SOBRE LA SINTAXIS



Las **sentencias SQL** no son **sensibles** a las **mayúsculas y minúsculas**.

No obstante, es importante **respetarlas** al colocar el **nombre** de un **campo** o **tabla**.

Cada sistema de bases de datos tiene sus particularidades sintácticas.

Sin embargo, si conocemos la base de SQL podremos adaptarnos sin dificultades

Cada consulta finaliza con punto y coma (;)

CONSULTA DE SELECCIÓN

USO DE SELECT - FROM

SELECT *

FROM (tabla)

La sentencia **SELECT**, como lo indica su nombre, **permite seleccionar información** a extraer y gracias a esto visualizar el resultado.

La cláusula **FROM** complementa al **SELECT**. Esta **declara la/s tabla/s desde la/s cual/es se va a extraer la información**.

SELECT *

FROM (tabla)

Ejemplo
en vivo



```
SELECT id_class, description  
FROM class;
```

¿Qué resultado obtendremos de esta consulta?

| CLASS | |
|----------|-------------|
| id_class | description |
| 1 | Action |
| 5 | Thriller |
| 14 | Children |
| 18 | Comedy |
| 19 | Romance |
| 20 | Drama |
| 22 | Documentary |
| 25 | Fantasy |
| 29 | Sci-Fi |
| 31 | Animation |
| 34 | War |

CODER HOUSE

SELECCIÓN DE DETERMINADOS CAMPOS

Ejemplo
en vivo



- 👉 El orden de los campos en el **SELECT** es irrelevante.
- 👉 Podemos definir el que necesitamos en primer lugar, independientemente de la posición donde éste, haya sido definido cuando se creó la tabla.
- 👉 Al visualizar el resultado, el orden de los campos será tal como lo coloquemos en la consulta.

```
SELECT description, id_class FROM class;
```

SELECCIÓN DE TODOS LOS CAMPOS

Ejemplo
en vivo

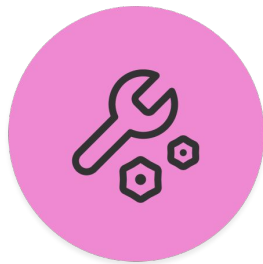


El símbolo *asterisco* (*****) juega el mismo papel que cuando lo usamos para buscar archivos o carpetas. Representa a “*todos*” los que existan.

De la misma forma que en un S.O. buscamos “*Todos los archivos*”, en las consultas hacemos lo propio para traer todos los campos de una tabla.

```
SELECT * FROM system_user;
```

¡VAMOS A PRACTICAR UN POCO!



SELECT - FROM

Determina qué registros traerá cada consulta.

Tiempo estimado: 5 minutos



SELECT - FROM

Pondremos en práctica el uso de **SELECT - FROM**, analizando previamente cuál será el resultado de cada consulta **SELECT**, de acuerdo a los registros que posee la tabla

Trabajaremos con una tabla de la DB [Gamer](#); observa las siguientes consultas y determina qué registros traerá cada una.

Tiempo estimado: 5 minutos



Utiliza el chat para responder.



PONGAMOS EN PRÁCTICA EL USO DE SELECT - FROM

- 1) `SELECT *`
`FROM system_user;`
- 2) `SELECT first_name, last_name`
`FROM system_user;`
- 3) `SELECT first_name, last_name, email`
`FROM system_user;`
- 4) `SELECT id_system_user, first_name, last_name`
`FROM system_user;`

Vamos a utilizar la base de datos que generamos luego de la instalación de MySQL, [Gamer](#)

No te olvides de separar las cláusulas de la consulta

SELECT DISTINCT

SELECT DISTINCT

La cláusula **DISTINCT** funciona en
conjunción con **SELECT**. Permite filtrar de
una consulta aquellos registros repetidos del
resultado de la misma.

| id_system_user | first_name |
|----------------|------------|
| 1 | Tyson |
| 2 | Tam |
| 3 | Rosamund |
| 4 | Bent |
| 5 | Averell |
| 6 | Aurora |
| 7 | Somerset |
| 8 | Victor |
| 9 | Coletta |

¿Podemos considerar los registros de esta tabla como registros repetidos, o no?

IDENTIFICAR EL CAMPO O CAMPOS

Para utilizar **la sentencia SELECT DISTINCT**, debemos incorporar siempre el nombre de al menos un campo de la tabla.

Con ese dato, SQL podrá resolver cómo aplicar correctamente la distinción de los registros recuperados.



SELECT DISTINCT

SELECT DISTINCT

id_system_user, first_name

FROM system_user;

| id_system_user | first_name |
|----------------|------------|
| 1 | Tyson |
| 2 | Tam |
| 3 | Rosamund |
| 4 | Bent |
| 5 | Averell |
| 6 | Aurora |
| 7 | Somerset |
| 8 | Victor |
| 9 | Coletta |

En este caso, **DISTINCT** se aplica sobre los **dos** campos.

De esta forma, **determina que el conjunto de campos de cada registro, es diferente al resto.**



SELECT DISTINCT

```
SELECT DISTINCT first_name  
FROM system_user;
```

| first_name |
|------------|
| Tyson |
| Tam |
| Rosamund |
| Bent |
| Averell |
| Aurora |
| Somerset |
| Victor |
| Coletta |

En este otro caso, **DISTINCT** se aplica sobre al campos **first_name**

Aquí es donde se saca mejor partida con la sentencia DISTINCT



BREAK

¡5/10 MINUTOS Y VOLVEMOS!

OPERADORES SQL

OPERADORES DE COMPARACIÓN

DEFINICIÓN

Los **operadores de comparación** en SQL nos **permiten evaluar una condición y determinar si el resultado es verdadero, falso o desconocido** (TRUE, FALSE o UNKNOWN)

```
first_name = 'Gillie'
```


OPERADORES DE COMPARACIÓN

En SQL se utilizan estos operadores combinados con la **sentencia WHERE**, que veremos a continuación

El **objetivo principal** es **aplicar un filtro sobre los datos almacenados en la tabla**, que cumplan con una cierta condición

LISTA DE OPERADORES DE COMPARACIÓN

| | | | | | |
|---------|------------------------|---------------|------------------------|----------------|---------------------------|
| = | <i>igual a</i> | IS [NOT] NULL | <i>no es nulo</i> | BETWEEN | <i>entre</i> |
| < | <i>menor a</i> | NOT | <i>NOT lógico</i> | [NOT] BETWEEN | <i>no esta entre</i> |
| > | <i>mayor a</i> | LIKE | <i>es como</i> | IN | <i>en (lista)</i> |
| <= | <i>menor o igual a</i> | [NOT] LIKE | <i>no es como</i> | [NOT] IN | <i>no esta en (lista)</i> |
| => | <i>mayor o igual a</i> | IS [NOT] TRUE | <i>no es verdadero</i> | IS [NOT] FALSE | <i>no es falso</i> |
| != ó <> | <i>distinto de</i> | AND | <i>AND lógico</i> | OR | <i>OR lógico</i> |

En la próxima clase, sacaremos el máximo provecho de estos operadores; usándolos tanto de forma individual como combinados.

SENTENCIA WHERE

APLICAR CONDICIONALES

DEFINICIÓN

La sentencia **WHERE** permite **agregar condiciones** para **filtrar** los resultados.

Obtendremos únicamente los registros que cumplan con dichas condiciones

SELECT *

FROM (tabla)

WHERE (condición)



USO DE WHERE JUNTO AL OPERADOR DE COMPARACIÓN IGUAL (=)

| SYSTEM_USER | | |
|----------------|------------|------------|
| id_system_user | first_name | last_name |
| 56 | Gillie | O' Finan |
| 71 | Reinaldos | La Grange |
| 171 | Gillie | Wauchope |
| 717 | Gillie | Philippson |
| 826 | Reinaldos | Bowshire |
| 858 | Reinaldos | Cowton |

| SYSTEM_USER | | |
|----------------|------------|-----------|
| id_system_user | first_name | last_name |
| 56 | Gillie | O' Finan |

```
SELECT id_system_user, first_name, last_name
FROM system_user
WHERE id_system_user = 56;
```

IMPORTANTE

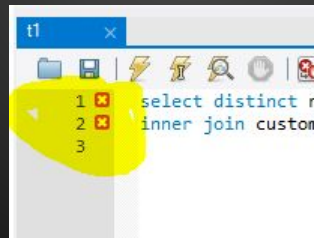
Las **sentencias** tienen un **orden** para su correcto funcionamiento.

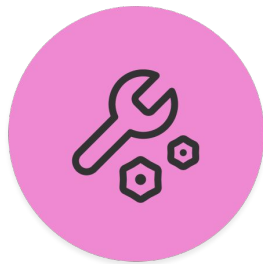
Cuando tenemos una **sintaxis** incorrecta, el **SGBD** nos presenta el error y en la mayoría de los casos lo hace de forma explícita.

1º **SELECT** (campos...)

2º **FROM** (tabla)

3º **WHERE** (condición/es)





PRÁCTICAS CON LA SENTENCIA WHERE

Analicemos qué trae cada una de las siguientes sentencias

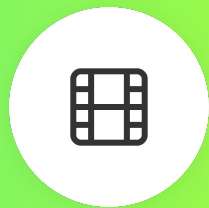
Tiempo estimado: 10 minutos



PRÁCTICAS CON LA SENTENCIA WHERE

Aplicaremos las consultas con la sentencia **WHERE** de la siguiente diapositiva; a la tabla **SYSTEM_USER** y discutiremos por Chat el resultado de cada una de las consultas:

- 1) **SELECT * FROM system_user WHERE first_name = 'Gillie';**
- 2) **SELECT first_name, last_name FROM system_user WHERE id_user_type = 334;**
- 3) **SELECT first_name, last_name FROM system_user WHERE id_system_user = 56;**
- 4) **SELECT * FROM system_user WHERE first_name = 'Reinaldos';**



***¿QUIERES SABER MÁS? TE DEJAMOS
MATERIAL AMPLIADO DE LA CLASE***



- Artículo de opinión: [7 razones para aprender SQL](#) | CampusMPV.es
- Artículo: [5 Bases de datos para la empresa](#) | Francisco Palazón
- Practicar SQL: [w3schools](#)

¿PREGUNTAS?



LES PUEDO HACER
UNA CONSULTA?



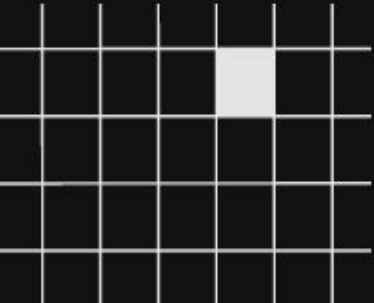
SELECT * FROM





¡MUCHAS GRACIAS!

Resumen de lo visto en clase hoy:

- El lenguaje SQL
 - SELECT, DISTINCT
 - Operadores de comparación
 - Sentencia WHERE
- 



OPINA Y VALORA ESTA CLASE

#DEMOCRATIZANDO LA EDUCACIÓN

CODER HOUSE