# *Lenguaje SQL- Selección de datos*

Sentencia SELECT.

Extraer datos de una base de datos.

La sintaxis de SELECT es compleja pero una forma general consiste en la siguiente sintaxis:

SELECT [ DISTINCT | ALL | Otras opciones ] *lista_selección*

[INTO OUTFILE *'nombre_fichero' opciones_exportación*]

FROM *tabla_origen* [,*tabla_origen*] ....

[WHERE *condición_de_búsqueda*]

[LIMIT *m,n*]

[GROUP BY *expresión_de_agrupamiento*]

[HAVING *condición_de_búsqueda*]

[ORDER BY *expresión_order_by* [ ASC | DESC]]

# ¿Qué hace SELECT?

La sentencia SELECT recupera filas o columnas (las especificadas en *lista_selección*) de una o varias tablas especificadas en la clausula FROM que satisfagan las condiciones de búsqueda especificadas en WHERE, agrupadas según el criterio de agrupamiento de GROUP BY, cumpliendo las condiciones de búsqueda de HAVING y ordenadas según lo especificado en ORDER BY.

La sentencia SELECT obtiene filas de la base de datos y permite realizar la selección de una o varias filas o columnas de una o varias tablas.

# Sus cláusulas principales son:

SELECT

FROM

WHERE

GROUP

HAVING

ORDER

# CONSULTAMOS

- www.w3scholls.com
- Manual Mysql Completo :

  Capítulo 13

  Apartado 13.2 Sentencias de manipulación de datos

  Subapartado : 13.2.7 Sintaxis SELECT (Pag 685)
- Curso Mysql: Consultas (pag 127)

# Cláusula SELECT

Especifica las columnas que va a devolver la consulta.

SELECT [ DISTINCT | ALL | Otras opciones ] ***lista_selección***

En cuanto a las opciones [ALL | DISTINCT |.....]

ALL especifica que pueden aparecer filas duplicadas en el conjunto de resultados (es el valor predeterminado).

DISTINCT especifica que sólo pueden aparecer filas exclusivas en el conjunto de resultados.

Los valores NULL se consideran iguales a efectos de la palabra clave DISTINCT.

En la sentencia SELECT, a continuación de las opciones, se sitúa la ***lista de selección*** que incluye las columnas que se van a seleccionar para el conjunto de resultados separadas por comas.

SELECT [ DISTINCT | ALL | Otras opciones ] *lista_selección*

- De modo más general, la *lista de selección* puede ser una serie de expresiones separadas por comas donde cada expresión puede ser un nombre de columna, constante, función o una combinación de nombres de columnas, constantes y funciones conectados mediante operadores.

- La *lista de selección* también puede ser la expresión (*), que especifica que se deben devolver todas las columnas de todas las tablas de la cláusula FROM.

- Las columnas se devuelven por tabla, tal como se especifique en la cláusula FROM, en el orden en que se encuentran en la tabla.

SELECT [ DISTINCT | ALL | Otras opciones ] *lista_selección*

- La *lista de selección* también puede ser la expresión *nombre_tabla.**, que limita el alcance de * a la tabla especificada.

- Al especificar las columnas en la *lista de selección* es conveniente situar los nombres de las tablas antes de los de las columnas separadas por un punto (calificar) para evitar realizar una referencia ambigua, como ocurre cuando dos tablas de la cláusula FROM tienen columnas de idéntico nombre.

- Por tanto, la *lista de selección* también puede contener los nombres de las columnas expresadas como *nombre_tabla.nombre_columna*.

SELECT [ DISTINCT | ALL | Otras opciones ] *lista_selección*

- Detrás del nombre de cualquier columna de la *lista de selección* puede situarse un **alias** para las columnas mediante *nombre_columna AS alias_columna*.
- La sintaxis *alias_columna* es un nombre alternativo para reemplazar el nombre de la columna en el conjunto de resultados de la consulta.
- Los alias también se utilizan para especificar nombres para los resultados de expresiones.
- La sintaxis *alias_columna* puede utilizarse en una cláusula ORDER BY, GROUP BY o HAVING, sin embargo, no puede utilizarse en una cláusula WHERE.

## Cláusula FROM

**E**specifica las tablas de donde se van a obtener las filas.

SELECT [ DISTINCT | ALL | Otras opciones ] *lista_selección*

La cláusula FROM es necesaria excepto cuando la *lista de selección* sólo contiene constantes, variables, expresiones aritméticas (no nombre de columnas) y expresiones del tipo *.

Su sintaxis básica es la siguiente:

[FROM *lista_tablas*]

Lo habitual es que *lista_tablas* sean varios nombres de tablas separados por comas, en cuyo caso la SELECT devuelve todas las combinaciones posibles de filas de dichas tablas (producto cartesiano).

# [FROM *lista_tablas*]

- Los elementos de *lista_tablas* pueden estar separados por cláusulas de unión ( en vez de por comas).

- Estas cláusulas pueden ser **JOIN, CROSS JOIN, INNER JOIN, STRAIGHT_JOIN y LEFT_JOIN**.

- El argumento JOIN indica que las tablas especificadas en FROM deben combinarse.

- También pueden aparecer mezcladas comas y las cláusulas unión.

# [FROM *lista_tablas*]

Algunas sintaxis particulares más detalladas para *lista_tablas* son:

*nombre_tabla* [CROOS] JOIN *nombre_tabla*

*nombre_tabla* INNER JOIN *nombre_tabla*

*nombre_tabla* STRAIGHT_ JOIN *nombre_tabla*

*nombre_tabla* LEFT [OUTER] JOIN *nombre_tabla* ON *expresión_condicional*

*nombre_tabla* LEFT [OUTER] JOIN *nombre_tabla* USING (*lista_columnas*)

*nombre_tabla* NATURAL LEFT [OUTER] JOIN *nombre_tabla*

LAS VEREMOS MÁS ADELANTE

# Cláusula WHERE

Especifica una condición de búsqueda para restringir las filas que se van a devolver.

Su sintaxis es la siguiente:

[WHERE (*condición_búsqueda*)]

El argumento (*condición_búsqueda*) limita las filas devueltas en el conjunto de resultados mediante el uso de predicados.

No hay límite en el número de predicados que se pueden incluir en una condición de búsqueda.

El conjunto de resultados también puede estar limitado por HAVING y LIMIT.

# Cláusula GROUP BY

Especifica las agrupaciones que se van a realizar en las filas de salida y, en caso de incluir funciones agregados como **COUNT, MAX**, etc... en la cláusula SELECT (*lista_selección*), calcula el valor resumen de cada grupo.

SELECT [ DISTINCT | ALL | Otras opciones ] ***lista_selección***

[GROUP BY *expresión_de_agrupamiento*]

Cuando se especifica GROUP BY , cada columna que no esté en una expresión de agregado de la **lista de selección** se debe incluir en la lista de GROUP BY o la expresión GROUP BY debe coincidir exactamente con la expresión de la lista de selección.

# Cláusula GROUP BY

Su sintaxis es:

[GROUP BY *expresión_de_agrupamiento*]

La *expresión agrupamiento* especifica la expresión en la que se realiza el agrupamiento o columna de agrupamiento y puede ser una columna o una expresión diferente a una de agregado que hace referencia a una columna.

# Cláusula GROUP BY

SELECT [ DISTINCT | ALL | Otras opciones ] ***lista_selección***

[GROUP BY *expresión_de_agrupamiento*]

No se puede utilizar un alias de columna en la **lista de selección** para especificar una columna de agrupamiento.

Si no se especifica la cláusula ORDER BY, los grupos devueltos con la cláusula GROUP BY no estarán en un orden particular.

**Se recomienda que siempre se utilice la cláusula ORDER BY con GROUP BY para especificar un orden determinado de los datos.**

# Cláusula HAVING

Especifica una condición de búsqueda de un grupo o agregado y normalmente se utiliza con la cláusula GROUP BY.

Su sintaxis es la siguiente:

[HAVING *condición_de_búsqueda*]

El argumento *condición_de_búsqueda* especifica la condición de búsqueda del grupo o agregado que se debe cumplir.

# Cláusula HAVING

HAVING especifica una expresión secundaria que se **usa para limitar filas** después de que éstas hayan satisfecho las condiciones citadas en la cláusula WHERE.

Las filas que no satisfagan la condición HAVING serán rechazadas en el resultado de la consulta.

# Cláusula HAVING

HAVING suele usarse para expresiones que implican resúmenes de funciones que no pueden ser examinadas por la cláusula WHERE.

De todas formas, si una condición de búsqueda es legal tanto en la cláusula WHERE como en la cláusula HAVING, es preferible situarla en la primera.

# Cláusula ORDER BY

Con esta cláusula se especifica el orden del conjunto de resultados.

La cláusula ORDER BY no es válida en vistas, funciones en línea, tablas derivadas ni subconsultas, salvo que se especifique también TOP.

Su sintaxis es la siguiente:

**[ORDER BY { *expresion_order_by* [ASC | DESC]}  [,...N]]**

El argumento ***expresion_order_by*** especifica la columna según la que se realiza la ordenación.

# Cláusula ORDER BY

Se puede especificar una columna de orden como un nombre o alias de columna ( que puede estar calificado con el nombre de una tabla), una expresión .

[ORDER BY { ***expresion_order_by*** [ASC | DESC]} [,...N]]

También se pueden especificar varias columnas en orden.

La secuencia de columnas de orden en la cláusula ORDER BY define la estructura del conjunto ordenado de resultados.

# Cláusula ORDER BY

La cláusula ORDER BY puede incluir elementos que no aparecen en la lista de selección.

SELECT [ DISTINCT | ALL | Otras opciones ] *lista_selección*

[ORDER BY { *expresion_order_by* [ASC | DESC]} [,...N]]

Sin embargo , si se especifica SELECT DISTINCT, o si la instrucción SELECT contiene un operador UNION, las columnas ordenadas deben aparecer en la lista de selección.

Además, cuando la instrucción SELECT contiene un operador UNION, los nombres o los alias de las columnas deben ser los especificados en la primera lista de selección

# Cláusula ORDER BY

[ORDER BY { *expresion_order_by* [**ASC** | **DESC**]} [,...N]]

El argumento **ASC** indica que los valores de la columna especificada se deben ordenar ascendentemente (orden por defecto), desde el valor más bajo al más alto.

El argumento **DESC** indica que los valores de la columna especificada se debe ordenar de manera descendente, desde el valor más alto al valor más bajo.

Los valores NULL se tratan como los valores más bajos posibles.

No hay límite para el número de elementos en una cláusula ORDER BY.

# Cláusula LIMIT

La clausula LIMIT se utiliza para seleccionar una sección de filas del conjunto de resultados.

Su sintaxis admite las dos opciones siguientes:

[LIMIT n] devuelve las primeras **n** filas de la selección

[LIMIT **m, n**] devuelve **n** filas a partir de la fila **m**

Para LIMIT las filas se numeran comenzando por la 0, no por la 1.

# Esquema resumen de la sentencia SELECT

SELECT [ opción]{ * | tabla.* |[tabla.]campo1 [AS alias1] [, [tabla.]campo2 [AS alias2] [, ...]]}

FROM expresiónTabla [,...]

[WHERE ...]

[GROUP BY ....]

[HAVING ...]

[ORDER BY ...]

**Si se incluye el mismo nombre de campo en más de una tabla en la cláusula FROM, se escribe delante el nombre de la tabla y el operador . (punto).**

# La sintaxis para la cláusula SELECT consta de estos argumentos:

| **Argumento** | **Descripción** |
|---|---|
| opción | Un predicado lógico de tipo : ALL, DISTINCT, ... puede utilizar el predicado para limitar el número de registros devueltos (todos, los que son distintos en los campos seleccionados, los que son distintos en todos los campos o los comprendidos en un intervalo). Si no se especifica ningún predicado, el valor predeterminado es ALL. |
| * | Especifica que se seleccionan todos los campos de la tabla o tablas especificadas |
| tabla | El nombre de la tabla que contiene los campos de la que se seleccionan los registros. |
| campo1,campo2 | Los nombre de los campos (columnas) que contienen los datos que se desean recuperar. Si incluye más de un campo, éstos se recuperan en el orden enumerado. |
| alias1, alias2 | Los nombre que se va utilizar como encabezado de columnas en vez de los nombres de columnas originales en la tabla. |
| expresiónTabla | El nombre de la tabla o tablas que contienen los datos que se desean recuperar |

# Observaciones relativas a la sentencia SELECT

El orden de las cláusulas en la sentencia SELECT es importante.

Se puede omitir cualquiera de las cláusulas opcionales, pero, cuando se utilizan, deben aparecer en el orden apropiado.

SELECT

FROM

WHERE

GROUP

HAVING

ORDER

# El orden de procesamiento de una instrucción SELECT que contiene cláusulas WHERE, GROUP BY, HAVING

1. La cláusula WHERE excluye las filas que no cumplen su condición de búsqueda.
2. Las cláusulas GROUP BY recopila las filas seleccionadas en un grupo para cada valor exclusivo de la cláusula GROUP BY.
3. Las funciones de agregados especificadas en la lista de selección calculan los valores de resumen de cada grupo.
4. La cláusula HAVING excluye también las filas que no cumplen su condición de búsqueda

# Algunos ejemplos

**Forma incondicional**
**Limitar las columnas: proyección**
**Alias**
**Mostrar filas repetidas**
**Limitar las filas: selección**
**Agrupar filas**
**Ordenar resultados**
**Limitar el número de filas de salida**

**Forma incondicional**

La forma más sencilla consiste en pedir todas las columnas de la/s tabla/s y no especificar condiciones.

```
SELECT * FROM gente;
```

**Limitar las columnas**

```
SELECT nombre FROM gente;

SELECT clave,poblacion FROM ciudad5;

SELECT SIN(3.1416/2), 3+5, 7*4;

SELECT nombre,fecha,DATEDIFF(CURRENT_DATE(),fecha)/365
    FROM gente;
```

## Alias

```
SELECT nombre, fecha, DATEDIFF(CURRENT_DATE(),fecha)/365 AS edad
    FROM gente;
```

## Mostrar filas repetidas

```
SELECT DISTINCT fecha FROM gente;
```

## Limitar las filas: selección

```
SELECT * FROM gente WHERE nombre="Mengano“;
SELECT * FROM gente WHERE fecha>="1986-01-01“;
SELECT * FROM gente WHERE fecha>="1986-01-01"
    AND fecha < "2000-01-01";
```

# Agrupar filas

SELECT fecha FROM gente GROUP BY fecha;

```
mysql> SELECT fecha, COUNT(*) AS cuenta FROM gente GROUP BY fecha;
```

Esta sentencia muestra todas las fechas diferentes y el número de filas para cada fecha

```
+-------------+---------+
| fecha       | cuenta  |
+-------------+---------+
| 1978-06-15  |       2 |
| 1985-04-12  |       2 |
| 1993-02-10  |       1 |
| 2001-12-02  |       1 |
+-------------+---------+
4 rows in set (0.00 sec)
```

**Agrupar filas – con funciones de grupo**

```
SELECT student_name, AVG(test_score)
FROM student
GROUP BY student_name;
```

Dos Tablas ( **student**, **course**)

```
SELECT student.student_name,COUNT(*)
FROM student,course
WHERE student.student_id=course.student_id
GROUP BY student_name;
```

## Algunas Funciones

```
SELECT COUNT(DISTINCT results) FROM student;
```

```
SELECT student_name,MIN(test_score),MAX(test_score)
FROM student
GROUP BY student_name;
```

```
SELECT MAX(nombre) FROM gente
```

tabla: muestras

Columnas: **ciudad** fecha temperatura

(,'Madrid' '2005-03-17', 23),
('París', '2005-03-17', 16),
('Berlín', '2005-03-17', 15),
('Madrid', '2005-03-18', 25),
('Madrid', '2005-03-19', 24),
('Berlín', '2005-03-19', 18);

```
+--------+------------------+
| ciudad | MAX(temperatura) |
+--------+------------------+
| Berlín | 18 |
| Madrid | 25 |
+--------+------------------+
```

```
SELECT ciudad, MAX(temperatura)
 FROM muestras
 GROUP BY ciudad
HAVING MAX(temperatura)>16;
```

**Ordenar resultados**

SELECT * FROM gente ORDER BY fecha;

**Limitar el número de filas de salida**

SELECT * FROM gente LIMIT 3;