CONSULTAS DEL LENGUAJE SQL

Seleccionar registros de una tabla

La sentencia **SELECT** permite realizar operaciones de selección, ordenación, agrupación y filtrado de registros, es decir:

- Se utiliza para seleccionar información almacenada en una o varias tablas.
- Para seleccionar todas las columnas se utiliza el * (asterisco)
- La cláusula WHERE se utiliza para establecer un criterio de búsqueda o filtro

El siguiente ejemplo selecciona de la tabla **Articulos** los valores de todas las columnas de todos los registros.

Sintaxis

```
SELECT * FROM Articulos;
```

El siguiente ejemplo selecciona de la tabla **Articulos** los valores de la columna **Nombre** de todos los registros.

Sintaxis

```
SELECT Nombre FROM Articulos;
```

El siguiente ejemplo selecciona de la tabla **Artículos** todos los valores de la columna **Nombre** y **Precio** de todos los registros.

Sintaxis

```
SELECT Nombre, Precio FROM Articulos;
```

El siguiente ejemplo selecciona de la tabla **Articulos** los valores de todas las columnas de todos los registros.

Sintaxis

```
SELECT * FROM Articulos;
```

El siguiente ejemplo selecciona de la tabla **Articulos** los valores de todas las columnas y agrega una nueva **'Precio con Aumento'** incrementando en un 25% el valor de la columna Precio.

Sintaxis

```
SELECT *, Precio * 1.25 as 'Precio con Aumento' FROM Articulos;
```

El siguiente ejemplo selecciona de la tabla **Articulos** los valores de todas las columnas y agrega una nueva **'Origen'** con el valor constante 'China'.

Sintaxis

```
SELECT *, 'China' as Origen FROM Articulos;
```

Ordenamiento de Datos ORDER BY

Tiene como finalidad ordenar los resultados de las consultas por columnas en vez del campo índice(primary key) por defecto.

- El orden puede ser ascendente (por defecto) o descendente.
- Se puede ordenar por una columna o un conjunto de ellas.

El siguiente ejemplo ordena el resultado por el atributo apellido por defecto ascendente.

Sintaxis

```
SELECT nombre, apellido
FROM clientes
ORDER BY apellido; -- ASC
```

El siguiente ejemplo ordena el resultado por el atributo apellido por defecto ascendente y Nombre descendente.

Sintaxis

```
SELECT nombre, apellido
FROM clientes
ORDER BY apellido ASC, nombre DESC;
```

Limitar la Cantidad de Registros LIMIT

Su funcionalidad es la de limitar el número de filas (registros/resultados) devueltas en las consultas SELECT. También establece el número máximo de registros a eliminar con DELETE.

El siguiente ejemplo limita a dos el número total de registros.

Sintaxis

```
SELECT nombre, apellido
FROM clientes
LIMIT 2;
```

OFFSET

Opera en combinación con la cláusula LIMIT y permite posicionarse en el registro indicado, luego LIMIT seleccionará la cantidad de registros establecidos en la sentencia.

El siguiente ejemplo limita a dos el número total de registros a partir del 4 registro.

Sintaxis

```
SELECT nombre, apellido
FROM clientes
LIMIT 2 OFFSET 4;
```

CONCEPTOS AVANZADOS

SQL ofrece distintos tipos de funciones disponibles para ser utilizadas, las cuales tienen como finalidad simplificar la presentación y la obtención de los resultados de las consultas. Recuerda que no debe haber espacios entre un nombre de función y los paréntesis porque MySQL puede confundir una llamada a una función con una referencia a una tabla o campo que tenga el mismo nombre de una función.

FUNCIONES - Algunos ejemplos

El siguiente ejemplo utiliza una función para concatenar cadenas de caracteres (valores de campo o valores constantes de texto) en una sola columna.

Sintaxis

```
SELECT CONCAT('Sr/a. ', nombre, ' ', apellido) as 'Nombre completo' FROM clientes;
```

El siguiente ejemplo utiliza una función para concatenar cadenas de caracteres (ws "with separator"). El primer argumento debe especificar el carácter que se utilizará para separar las cadenas a concatenar.

Sintaxis

```
SELECT CONCAT_WS(';', nombre, apellido, cuit, direccion) as 'Datos completos' FROM clientes;
```

El siguiente ejemplo utiliza una función que convierte a mayúsculas el valor del campo **nombre**, asimismo le asigna el alias **Nombre** sin utilizar la palabra **as**, ya que el uso de la misma es opcional

Sintaxis

```
SELECT UPPER(nombre) Nombre FROM clientes;
```

El siguiente ejemplo utiliza una función que convierte a minúsculas el valor del campo **apellido y, al no asignarse un alias a la columna resultante, ésta tendrá como nombre la función o fórmula utilizada.**

Sintaxis

```
SELECT LOWER(apellido) FROM clientes;
```

El siguiente ejemplo utiliza una función que devuelve la primera letra del campo **nombre**, en este ejemplo se utiliza para concatenar con un '.'. El segundo argumento de la función LEFT debe ser un número entero que indica la cantidad de caracteres a extraer.

Sintaxis

```
SELECT CONCAT(LEFT(nombre, 1), '.') As Inicial_nombre FROM clientes;
```

El siguiente ejemplo utiliza una función que devuelve la última letra del campo **cuit**, en este ejemplo se utiliza para obtener el último dígito verificador del cuit. El segundo argumento de la función RIGHT debe ser un número entero que indica la cantidad de caracteres a extraer.

Sintaxis

```
SELECT RIGHT(cuit, 1) as 'Dígito verificador' FROM clientes;
```

El siguiente ejemplo utiliza una función para extraer del campo **cuit**, la cadena de caracteres que identifica el número de documento. Utiliza 3 argumentos: el primer argumento de la función debe ser el campo o cadena de caracteres, el segundo un número entero que indica desde qué posición de la cadena se extraerá el contenido, y el tercero la cantidad de caracteres a extraer.

Sintaxis

```
SELECT SUBSTRING(cuit, 4, 8) as 'DNI' FROM clientes;
```

El siguiente ejemplo utiliza una función para redondear el valor del campo **precio dividido por 3 con 2 decimales**. El segundo argumento de la función debe ser un número entero que indica la cantidad de decimales. **Si el segundo argumento es 0 es resultado será un número entero.**

Sintaxis

```
SELECT ROUND(precio/3, 2) FROM articulos;
```

El siguiente ejemplo utiliza una función que devuelve el año de un campo de tipo fecha.

Sintaxis

```
SELECT YEAR(fecha) as 'Año' FROM facturas;
```

El siguiente ejemplo utiliza una función que devuelve el mes de un campo de tipo fecha.

Sintaxis

```
SELECT MONTH(fecha) as 'Mes' FROM facturas;
```

El siguiente ejemplo utiliza una función que devuelve el día de un campo de tipo fecha.

Sintaxis

```
SELECT DAY(fecha) as 'Día' FROM facturas;
```

El siguiente ejemplo utiliza una función que devuelve la hora de un campo de tipo fecha/hora (Datetime).

Sintaxis

```
SELECT HOUR(fecha) as 'HORA' FROM facturas;
```

El siguiente ejemplo utiliza una función que devuelve la fecha actual.

Sintaxis

```
SELECT CURDATE() as 'FECHA ACTUAL';
```

El siguiente ejemplo utiliza una función que devuelve la hora actual.

Sintaxis

```
SELECT CURTIME() as 'HORA ACTUAL';
```

El siguiente ejemplo utiliza una función que devuelve la diferencia de días entre 2 fechas.

Sintaxis

```
SELECT DATEDIFF('2020-06-30','2020/01/01') as 'Días transcurridos';
```

El siguiente ejemplo utiliza una función que devuelve el nombre del día de la semana de la fecha actual.

Sintaxis

```
SELECT DAYNAME(CURDATE()) as 'Nombre del día';
```

El siguiente ejemplo utiliza una función que devuelve el índice o número del día de la semana de la fecha actual.

Sintaxis

```
SELECT DAYOFWEEK(CURDATE()) as 'Nombre del día de la semana';
```

El siguiente ejemplo utiliza una función que devuelve el índice o número del día del año de la fecha actual.

Sintaxis

```
SELECT DAYOFYEAR(CURDATE()) as 'Día del año';
```

CASE

Permite asignar un valor a una columna tomando como referencia otro valor de columna de la tabla. Por cada valor o grupo de valores hay un **when** y un **then**; si encuentra un valor coincidente en algún **when** ejecuta el **then** correspondiente a ese **when**, si no encuentra ninguna coincidencia, se ejecuta el **else**. Finalmente se coloca **end** para indicar que el **case** ha finalizado.

El siguiente ejemplo asigna un posible valor (CARO / BARATO / EQUILIBRADO) a la columna Categoría.

Sintaxis

```
SELECT nombre, precio,
CASE
WHEN precio < 20 THEN 'BARATO'
WHEN precio BETWEEN 20 AND 40 THEN 'EQUILIBRADO'
ELSE 'CARO' END as Categoría
FROM articulos;
```