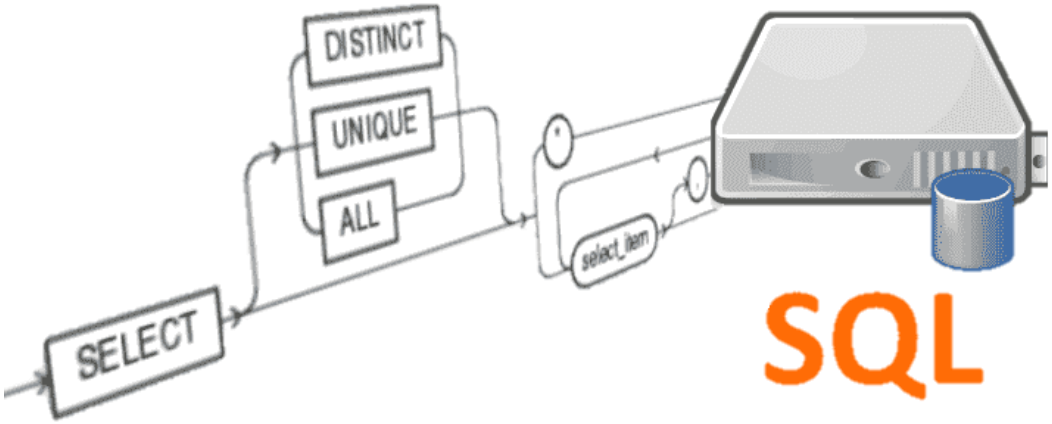
SQL

Structured Query Language

(Lenguaje de Consulta Estructurada)



Sentencia SELECT

SELECT [ALL|DISTINCT]

{ * | *expr_1* [AS *c_alias_1*] [, ... [, *expr_k* [AS *c_alias_k*]]]}

FROM *table_name_1* [*t_alias_1*] [, ... [, *table_name_n* [*t_alias_n*]]]

[WHERE *condition*]

[GROUP BY *name_of_attr_i* [,... [, *name_of_attr_j*]]
[HAVING *condition*]]

[{UNION [ALL] | INTERSECT | EXCEPT} SELECT ...]
[ORDER BY *name_of_attr_i* [ASC|DESC] [, ... [, *name_of_attr_j* [ASC|DESC]]]];

Funciones de agregación

Las funciones de agregación en SQL son funciones que operan sobre un conjunto de valores y devuelven un valor único. Son ampliamente utilizadas para realizar cálculos en bases de datos y extraer información útil de grandes volúmenes de datos.

Las funciones de agregación suelen
utilizarse junto con la cláusula GROUP BY, que permite agrupar los datos antes de aplicar la agregación.

Funciones de agregación

COUNT() : Cuenta el número de filas que cumplen con una condición.

Esto devuelve el total de filas en la tabla empleado.

SELECT COUNT(*) **FROM** empleado

También se puede contar un campo específico o bien filas con una determinada condición

SELECT COUNT(*) **FROM** empleado **WHERE** sueldo > 900000,00

Funciones de agregación

SUM(): Suma todos los valores de una columna numérica.

Esto devuelve la suma de los sueldos de los empleados del sector ventas

SELECT SUM(sueldo) **FROM** empleado **WHERE** idSector = 2

AVG(): Calcula el promedio (media) de una columna numérica.

Esto devuelve el salario promedio de todos los empleados

SELECT AVG(salario) FROM empleado

Funciones de agregación

MIN(): Devuelve el valor mínimo de una columna.

Esto devuelve el salario más bajo de todos los empleados

SELECT MIN(salario) **FROM** empleado.

MAX(): Devuelve el valor máximo de una columna

Esto devuelve el salario más alto de todos los empleados

SELECT MAX(salario) **FROM** empleado

Funciones de agregación

Si queremos conocer el **precio promedio** de todos los productos de la tabla producto, utilizaremos la siguiente consulta:

SELECT AVG(precio**) AS** precio_prom **FROM** producto

Si queremos conocer con cuántos pedidos realizó un determinado cliente utilizaremos la instrucción:

SELECT COUNT(IdCliente**)**
FROM pedido **AS** p **INNER JOIN** cliente **AS** c **USING(**IdCliente**)**
WHERE c.nombre = “Juan Perez”

Funciones de agregación

Si queremos conocer el producto mas caro y mas barato de la tabla producto, utilizaremos la siguiente consulta:

SELECT MAX(precio**) AS** caro**, MIN(**precio**) AS** barato **FROM** producto

Si queremos conocer cuántos Pedidos se hicieron durante el año 2020, utilizaremos la instrucción:

SELECT COUNT(*) FROM pedido **WHERE YEAR**(fechapedido) = 2020

Funciones de agregación

Observemos la siguiente consulta sobre la tabla pedido:

```
SELECT COUNT(*) AS Cant_Pedidos,
        MIN(FechaPedido) AS Desde,
        MAX(FechaPedido) AS Hasta,
        SUM(Monto) AS MontoTotal,
        AVG(Monto) MontoPromedio
FROM pedido AS p
INNER JOIN producto AS pr ON (p.IdProducto= pr.IdProducto)
WHERE pr.nombre = 'Producto 1'
```

Como se puede observar del resultado de la consulta anterior, las funciones de agregación devuelven una sola fila, salvo que vayan unidas a la cláusula GROUP BY, que veremos a continuación

Cláusulas comando SELECT

GROUP BY

- La cláusula **GROUP BY** permite agrupar filas según las columnas que se indiquen como parámetros,
- Permite funciones de agregación para **obtener datos resumidos y agrupados** por las columnas que se necesiten

HAVING

- Permite un nuevo filtrado, **pero sobre las tuplas afectadas por el GROUP BY**, en función de una condición aplicable a cada grupo de filas

Ejemplos (GROUP BY)

Esta consulta devuelve cuántos pedidos realizó el cliente Juan Perez

```
SELECT COUNT(IdCliente)
FROM pedido AS p INNER JOIN cliente AS c USING(IdCliente)
WHERE c.nombre = “Juan Perez”
```

Ahora queremos saber cuantos pedidos hizo cada cliente

```
SELECT c.IdCliente, c.nombre, count(IdCliente) as cantidad
FROM cliente as c
INNER JOIN pedido USING(idCliente)
GROUP BY c.IdCliente, c.nombre
```

Agregación por grupos

SQL permite separar tuplas de una tabla en grupos. En estas condiciones, los operadores agregados ya descriptos pueden aplicarse a los **grupos.**

Esto quiere decir que el valor del operador agregado no se calculan sobre todos los valores de la columna especificada, sino sobre todos los valores de un grupo.

- El **operador agregado se calcula** individualmente **para cada grupo**.
- El agrupamiento de las tuplas se hace utilizando las palabras clave **GROUP BY** seguidas de una lista de atributos que definen los grupos.

Ejemplo (Agregación por grupos)

Averiguar el sueldo promedio por cargo.

SELECT c.nombre, **AVG(**sueldo**)** as **SueldoPromedio**
FROM empleado e **INNER JOIN** cargo c **USING**(IdCargo)
GROUP BY c.nombre

Averiguar cantidad de familiares por empleado.

SELECT e.nombre, **COUNT**(Idfliar) **AS** Cantidad **FROM** empleado e
INNER JOIN Familiares f **USING**(IdEmpleado) **GROUP BY** e.nombre

SELECT e.nombre, **COUNT**(Idfliar) **AS** Cantidad **FROM** empleado e
LEFT JOIN Familiares f **USING**(IdEmpleado) **GROUP BY** e.nombre

Importante

- La cláusula GROUP BY se puede utilizar con más de un campo al mismo tiempo. Si indicamos más de un campo como parámetro nos devolverá la información agrupada por los registros que tengan el mismo valor en los campos indicados.
- **Nota**: Es muy importante tener en cuenta que cuando utilizamos la cláusula GROUP BY, los únicos campos que podemos incluir en el SELECT sin que estén dentro de una función de agregación, son los que vayan especificados en el GROUP BY.

Cláusula HAVING

- La cláusula HAVING trabaja de forma muy parecida a la cláusula WHERE, y se utiliza para considerar sólo aquellos grupos que satisfagan la cualificación dada en la misma.
- Las expresiones permitidas en la cláusula HAVING deben involucrar funcionen agregadas.
- Por otro lado, toda expresión que involucre funciones agregadas debe aparecer en la cláusula HAVING

Ejemplo (Cláusula HAVING)

Averiguar los cargos cuyo sueldo promedio es mayor a $ 30000

SELECT cargo, **AVG**(sueldo) as promedio **FROM** empleado **INNER JOIN** cargo **USING**(idcargo) **GROUP BY** cargo
HAVING AVG(sueldo) >= 30000

Listar los proveedores de Perú que venden menos de 3 artículos

SELECT p.proveedor, **COUNT**(IdProducto) as cantidad
FROM proveedor p **INNER JOIN** producto **USING**(idproveedor)
WHERE p.pais = 'Japón'
GROUP BY p.proveedor
HAVING cantidad < 3

Funciones en MySQL

Las funciones son simplemente son operaciones que devuelven un resultado.

Funciones integradas

Las funciones integradas son simplemente funciones que ya vienen **implementadas** en el servidor MySQL.

Estas funciones nos permiten realizar diferentes tipos de manipulaciones en los datos.

Las funciones integradas se pueden clasificar básicamente en las siguientes categorías más utilizadas: **numéricas, de cadena, de fecha, etc**

Ejemplos (funciones)

```
SELECT empleado, sueldo, round(sueldo*0.13,2) as Aporte_Jub
FROM empleado
```

```
SELECT  concat(direccion,' - ',ciudad) AS domicilio_completo
FROM   proveedores
```

```
SELECT empleado,
timestampdiff(year, fechanacimiento, curdate()) as edad
FROM Empleado
```

Ejemplos (funciones de Cadena)

Existen varias funciones para trabajar con cadenas de texto. Son útiles para manipular, extraer, formatear o analizar datos en formato de texto.

UPPER() y **LOWER**() Convierte la cadena a mayúsculas o minúsculas

SELECT UPPER(nombre) **AS** nom_may, **LOWER**(apellido) **AS** ape_min
FROM cliente

CONCAT() o **||** : Concatena (une) dos o más cadenas de texto.

SELECT CONCAT(nombre, ' ', apellido) **AS** nombre_completo
FROM cliente

Ejemplos (funciones de Cadena)

LENGTH(): Calcula la longitud (número de caracteres) de una cadena

SELECT nombre, **LENGTH**(nombre) **AS** longitud **FROM** clientes

Nota: **LENGTH**() se usa en MySQL y PostgreSQL, mientras que **LEN**() es común en SQL Server

SUBSTRING(): Extrae una subcadena, dado un índice de inicio y una longitud

SELECT SUBSTRING(nombre, 1, 3) AS iniciales **FROM** clientes
Esto devuelve los primeros tres caracteres de cada nombre

Ejemplos (funciones de Cadena)

TRIM() elimina los espacios en blanco al inicio y al final

LTRIM() elimina los del inicio y **RTRIM**() elimina los del final

SELECT TRIM(nombre) **FROM** cliente

Esto elimina los espacios en blanco al inicio y al final de nombre.

REPLACE(): Reemplaza una cadena específica por otra en una cadena

SELECT REPLACE(direccion, 'Calle', 'Av.') **FROM** cliente

Esto reemplaza la palabra "Calle" por "Av." en la columna direccion.

https://desarrolloweb.com/articulos/funciones-cadena-sql.html

Ejemplos (funciones numéricas)

Mostrar el IVA de los montos de la tabla Pedidos, lo que significa el 21% del mismo.

SELECT monto*21/100 FROM Pedidos (devuelve 6 decimales)

SELECT monto*0.21 FROM Pedidos (devuelve 4 decimales)

SELECT **round**(monto*0.21, 2) FROM Pedidos (para dejar a 2 decimales)

https://www.ediciones-eni.com/open/mediabook.aspx?idR=b6dcd73118868cb874f40dacf3c87acc

Ejemplos (funciones de Fechas)

CURRENT_DATE, **CURRENT_TIME** y **CURRENT_TIMESTAMP**: Devuelven la fecha, la hora o la fecha y hora actual del sistema

```
SELECT CURRENT_DATE as  fecha_actual,
       CURRENT_TIME as hora_actual,
       CURRENT_TIMESTAMP as fecha_y_hora_actual
```

EXTRACT(): Extrae una parte específica (año, mes, día, hora, etc.) de una fecha o fecha y hora

```
SELECT  EXTRACT(YEAR FROM fecha) AS anio,
        EXTRACT(MONTH FROM fecha) AS mes,
        EXTRACT(DAY FROM fecha) AS dia
FROM pedido
```

Ejemplos (funciones de Fechas)

ADDDATE(): Suma un intervalo de tiempo (días, meses, años, etc.) a una fecha dada

SELECT ADDDATE('2024-01-01', **INTERVAL** 10 **DAY**) **AS** fecha_mas_diez

Esto devuelve la fecha 10 días después del 1 de enero de 2024

DATEDIFF(): Calcula la diferencia entre dos fechas, en días

SELECT DATEDIFF('2024-01-15', '2024-01-01') **AS** diferencia_días

Esto devuelve 14, ya que hay 14 días entre el 1 de enero y el 15 de enero de 2024.

Ejemplos (funciones de Fechas)

DATE_FORMAT(): Formatea una fecha en un formato específico

SELECT DATE_FORMAT(fecha, '%d/%m/%Y') AS fecha_formateada
FROM pedidos

Esto muestra la fecha en formato DD/MM/YYYY

DAY(), MONTH(), YEAR() Extrae el día, el mes o el año de una fecha

SELECT DAY(fecha) **AS** dia, **MONTH**(fecha) **AS** mes, **YEAR**(fecha) **AS** anio
FROM pedidos

Esto devuelve el día, mes y año de cada pedido.

Ejemplos (funciones de Fechas)

TIMESTAMPDIFF() calcula la diferencia entre dos fechas en un intervalo específico (años, meses, días, etc.)

TIMESTAMPADD() agrega un intervalo a una fecha

SELECT TIMESTAMPDIFF(YEAR, fecha_nacimiento, **NOW**()) AS edad **FROM** empleados

Esto calcula la edad de los empleados en años.

Ejemplos (funciones de Fechas)

DAYOFWEEK() o **WEEKDAY**(): Devuelve el día de la semana de una fecha (el número de día de la semana, donde el domingo suele ser 1 en **DAYOFWEEK**(), o lunes 0 en **WEEKDAY**())

SELECT DAYOFWEEK(fecha) AS dia_semana **FROM** pedidos

Esto devuelve el número de día de la semana para cada fecha.8.

LAST_DAY(): Devuelve el último día del mes de una fecha dada

SELECT LAST_DAY('2024-01-15') **AS** ultimo_dia_mes

Esto devuelve 2024-01-31, ya que es el último día del mes de enero de 2024.

La función COALESCE

La función **COALESCE** se utiliza para devolver el primer valor no nulo de una lista de expresiones. Permite reemplazar los valores con otro valor alternativo cuando se encuentra un NULL

COALESCE(expresion1, expresion2, ..., expresionN)

SELECT nombre,
COALESCE(telefono_secundario, telefono_principal) **AS** contacto
FROM clientes;

COALESCE devuelve la primera expresión que no sea NULL.

La función COALESCE

Supón que tienes una tabla productos con una columna descuento. Si el valor es NULL, se asume que no tiene descuento (0%).

Puedes usar **COALESCE** para reemplazar el NULL con 0 en los cálculos

```
SELECT nombre, precio,
        COALESCE(descuento, 0) AS descuento,
        precio - (precio * COALESCE(descuento, 0) / 100) AS precio_final
FROM  productos;
```

COALESCE es especialmente útil para simplificar las consultas y mejorar la presentación de los datos, manejando los valores nulos de forma directa y clara.

La función COALESCE

```
SELECT id,
CONCAT(nombre, ' ', apellido) AS nombre_completo,
TIMESTAMPDIFF(YEAR, fecha_contratacion, CURRENT_DATE) AS antig,
TIMESTAMPDIFF(YEAR, fecha_nacimiento, CURRENT_DATE) AS edad,
COALESCE(tel_personal, tel_oficina, 'Sin teléfono') AS telefono_contacto,
salario + COALESCE(bono, 0) AS salario_total,
DATE_FORMAT(fecha_contratacion, '%d/%m/%Y') AS fecha_contratacion
FROM empleados;
```

Condiciones WHERE

- Funciones para cadenas de caracteres
  http://download.nust.na/pub6/mysql/doc/refman/5.0/es/string-functions.html
- Funciones y operadores numéricos
  https://dev.mysql.com/doc/refman/8.0/en/numeric-functions.html
- Funciones Matemáticas
  http://download.nust.na/pub6/mysql/doc/refman/5.0/es/mathematical-functions.html
- Funciones de Fecha y Hora
  http://download.nust.na/pub6/mysql/doc/refman/5.0/es/date-and-time-functions.html

Dónde buscar

- Funciones para cadenas de caracteres
  http://download.nust.na/pub6/mysql/doc/refman/5.0/es/string-functions.html
- Funciones y operadores numéricos
  https://dev.mysql.com/doc/refman/8.0/en/numeric-functions.html
- Funciones Matemáticas
  http://download.nust.na/pub6/mysql/doc/refman/5.0/es/mathematical-functions.html
- Funciones de Fecha y Hora
  http://download.nust.na/pub6/mysql/doc/refman/5.0/es/date-and-time-functions.html

Muchas Gracias