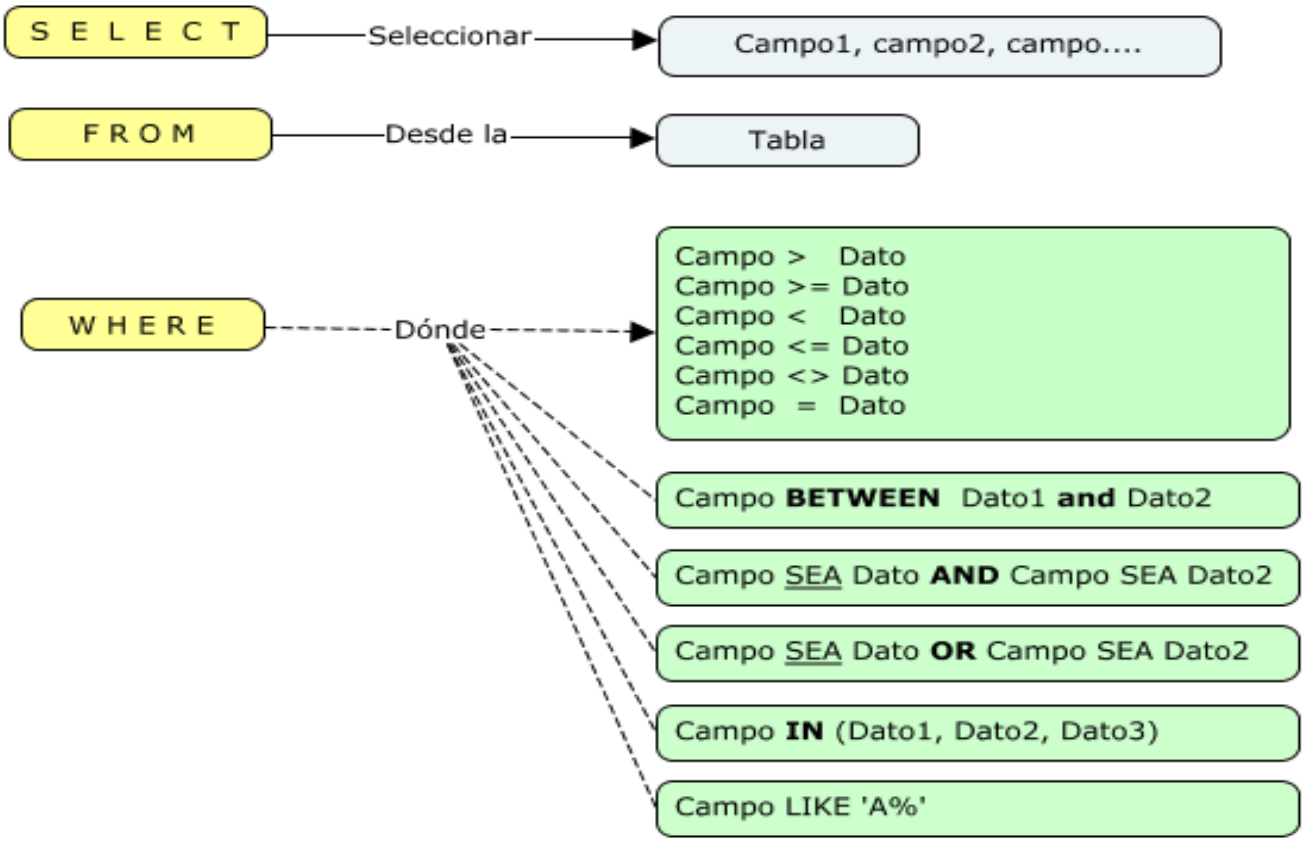
S E L E C T


S E L E C T
Seleccionar
Campo1, campo2, campo....
F R O M
Desde la
Tabla
W H E R E
Dónde
Campo > Dato
Campo >= Dato
Campo < Dato
Campo <= Dato
Campo <> Dato
Campo = Dato
Campo BETWEEN Dato1 and Dato2
Campo SEA Dato AND Campo SEA Dato2
Campo SEA Dato OR Campo SEA Dato2
Campo IN (Dato1, Dato2, Dato3)
Campo LIKE 'A%'

S E L E C T

En general a las operaciones básicas de manipulación de datos que podemos realizar con SQL se les denomina operaciones CRUD (de *Create, Read, Update and Delete*, o sea, Crear, Leer, Actualizar y Borrar, sería CLAB en español, pero no se usa). Lo verás utilizado de esta manera en muchos sitios, así que apréndete ese acrónimo.

Consulta de datos

Nos concentraremos en la "R" de CRUD, es decir, en cómo recuperar la información que nos interesa de dentro de una base de datos, usando para ello el lenguaje de consulta o SQL. Para realizar consultas sobre las tablas de las bases de datos disponemos de la instrucción SELECT. Con ella podemos consultar una o varias tablas. Es sin duda el comando más importante del lenguaje SQL.

Esta instrucción recupera las filas de una base de datos y habilita la selección de varias filas y columnas de tablas que se encuentran contenidas dentro de una base de datos. El proceso más relevante que podemos llevar a cabo en una base de datos es la consulta de los datos. No serviría de nada contar con una base de datos si no pudiéramos consultarla. Esta es una de las instrucciones más utilizadas en el mundo de las bases de datos.

Especificaciones de la sentencia

SELECT: Palabra clave y reservada que indica que la sentencia de SQL que queremos ejecutar es de selección.

FROM: Indica la tabla desde la que queremos recuperar los datos. En el caso de que exista más de una tabla se hace una combinación de tablas usando la instrucción JOIN. En las consultas combinadas es necesario aplicar una condición de combinación a través de una cláusula WHERE.

WHERE: Se utiliza cuando no se desea que se devuelvan todas las filas de una tabla, sino sólo las que cumplen ciertas condiciones.

Condiciones

Son expresiones lógicas por comprobar, para la condición de filtro, que tras su resolución devuelven para cada fila TRUE o FALSE, en función de que se cumplan o no. Se puede utilizar cualquier expresión lógica y en ella utilizar diversos operadores como:

(Mayor)

>= (Mayor o igual)

< (Menor)

<= (Menor o igual)

= (Igual)

o != (Distinto)

<> Distinto

IS [NOT] NULL (para comprobar si el valor de una columna es o no es nula, es decir, si contiene o no contiene algún valor)

Se dice que una columna de una fila es NULL si está completamente vacía. Hay que tener en cuenta que, si se ha introducido cualquier dato, incluso en un campo alfanumérico si se introduce una cadena en blanco o un cero en un campo numérico, deja de ser NULL.

Estructura básica

SELECT → significa Seleccionar, lista de campos

FROM →es Desde, tabla o conjunto de tablas

WHERE → condición o condiciones, para extraer los datos de, o las tablas.

El resultado de una consulta SELECT nos devuelve una tabla lógica. Es decir, los resultados son una relación de datos, que tiene filas/registros, con una serie de campos/columnas. Igual que cualquier tabla de la base de datos. Sin embargo, esta tabla está en memoria mientras la utilicemos, y luego se descarta. Cada vez que ejecutamos la consulta se vuelve a calcular el resultado

Las instrucciones mínimas para crear una consulta SELECT son:

SELECT

FROM

El SELECT posee tres argumentos opcionales.

TOP # devuelve un número de registros específicos.

TOP # PERCENT devuelve un porcentaje especifico del total de registros de la tabla.

DISTINCT muestra registros únicos que se especifican en la línea del SELECT

Ejemplos

Para los siguientes ejemplos se utiliza la siguiente tabla llamada Docentes.

```
SELECT * -- Mostrar todos los datos de la tabla docente.
FROM docentes

SELECT Nombre, cargo, salario -- Mostrar solo las columnas Nombre, cargo y salario..
FROM docentes

SELECT TOP 15 * -- Mostrar los 15 primeros registros.
FROM docentes

SELECT TOP 15 PERCENT * -- Mostrar el 15% de los registros de la tabla (resultado 23)
FROM docentes
```

SELECT DISTINCT ciudad -- Mostar solo la columna ciudad sin repetirlas, registros únicos (resultado 5)

FROM docentes

ALIAS... AS

Permite renombrar columnas si lo utilizamos en la cláusula SELECT, o renombrar tablas si lo utilizamos en la cláusula FROM. Es opcional. Con ello podremos crear diversos alias de columnas y tablas. Enseguida veremos un ejemplo.

SELECT Campo1 **AS** 'nombreColumna', campo2 AS 'nombreColumna'...

FROM Tabla

SELECT Mes **AS** 'Mes Ingreso'

FROM Docente

WHERE

La cláusula WHERE puede comparar valores de columnas, expresiones, funciones, listas de valores, constantes, etc.

La condición **Where** especifica el filtro de las filas devueltas. Se utiliza cuando no se desea que se devuelvan todas las filas de una tabla, sino sólo las que cumplen ciertas condiciones. Lo habitual es utilizar esta cláusula en la mayoría de las consultas.

Para limitar los registros recuperados por una consulta SQL, se usa la cláusula WHERE, justo después de la cláusula FROM, seguida de las condiciones de la comparación.

Ejemplos operadores lógicos

-- Mostrar los datos de los personas que el salario sea mayor o igual a 3 millones

```
SELECT apellido, nombre, cargo, salario

FROM docentes

WHERE salario >=3000000 -- Resultado 86
```

-- Mostrar los datos de los personas que viven en Medellín

```
SELECT apellido, nombre, facultad, salario

FROM docentes
```

WHERE ciudad ='Medellín' -- resultado 77

Ejemplos operadores lógicos AND - OR

AND → debe cumplir con todas las condiciones

OR → debe cumplir al menos una condición

-- Mostrar los datos de los personas que estén en el rango de los 40 y 50 años

SELECT apellido, nombre, facultad, salario

FROM docentes

WHERE años >=40 AND años <=50

-- Mostrar los datos de los personas que viven en los ciudades de Bello o Sabaneta

SELECT apellido, nombre, ciudad, cargo, salario

FROM docentes

WHERE ciudad = 'Bello ´ OR ciudad = 'Sabaneta'

-- Mostrar los datos de los personas que estén en el rango de salarios entre 3 millones y 5 millones

SELECT apellido, nombre, ciudad, facultad, salario

FROM docentes

WHERE → Entre, se utiliza para filtrar un rango de un campo numérico.

salario >= 3000000 AND salario <= 5000000

Ejemplos con BETWEEN:

Se utiliza para un intervalo de valores. Por ejemplo:

-- Mostrar los datos de los personas que estén en el rango de los 40 y 50 años

SELECT apellido, nombre, ciudad, cargo

FROM docentes

WHERE años BETWEEN 40 AND 50

-- Mostrar los datos de los personas que estén su Fecha Ingreso se encuentre entre el 1/1/95 y 31/12/05

SELECT apellido, nombre, facultad, cargo

FROM docentes

WHERE años BETWEEN 40 AND 50

IN( )

Seleccionar varios datos en la misma columna.

para especificar una relación de valores concretos. Por ejemplo:

```
SELECT apellido, nombre, ciudad, salario

FROM docentes

WHERE ciudad IN('bello','medellín')
```

La negación de IN

```
SELECT ciudad, salario

FROM docentes

WHERE ciudad NOT IN('bello','medellín')
```

Like

Se utiliza para comparar una expresión de cadena con un modelo en una expresión SQL.

LIKE: para la comparación de un modelo. Para ello utiliza los caracteres comodín especiales: "%" y "_". Con el primero indicamos que en su lugar puede ir cualquier cadena de caracteres, y con el segundo que puede ir cualquier carácter individual (un solo carácter). Con la combinación de estos caracteres podremos obtener múltiples patrones de búsqueda. Por ejemplo:

El nombre empieza por A: Nombre LIKE 'A%'

El nombre acaba por A: Nombre LIKE '%A'

El nombre contiene la letra A: Nombre LIKE '%A%'

El nombre empieza por A y después contiene un solo carácter cualquiera: Nombre LIKE 'A_'

El nombre empieza una A, después cualquier carácter, luego una E y al final cualquier cadena de caracteres: Nombre LIKE 'A_E%'

```
-- Mostrar los datos de las personas que su apellido comience con la letra A

SELECT apellido, nombre, ciudad, cargo, salario

FROM docentes

WHERE apellido LIKE 'a%'
```

ORDER BY

Define el orden de las filas del conjunto de resultados. Se especifica el campo o campos (separados por comas) por los cuales queremos ordenar los resultados.

ASC es el valor predeterminado, especifica que la columna indicad en la cláusula ORDER BY se ordenará de forma ascendente, o sea, de menor a mayor. Si por el contrario se especifica DESC se ordenará de forma descendente (de mayor a menor).

```
SELECT apellido, nombre, ciudad, cargo, salario

FROM docentes

ORDER BY apellido, Nombre DESC
```

O también se pude:

ORDER BY 2 DESC, 1

Columnas calculadas

Una columna calculada es una columna virtual que no está almacenada físicamente en la tabla, a menos que la columna esté marcada con PERSISTED. Las expresiones de columnas calculadas pueden utilizar datos de otras columnas al calcular un valor para la columna a la que pertenecen.

Limitaciones y restricciones

Una columna calculada no puede utilizarse como definición de restricción DEFAULT o FOREIGN KEY ni como NOT NULL.

Una columna calculada no puede ser el destino de una instrucción INSERT o UPDATE.

Ventajas

Las fórmulas en las columnas calculadas son muy similares a las fórmulas en Excel.

Una columna calculada está basada en datos ya incluidos en una tabla existente, o que se crean mediante una fórmula. Por ejemplo, podría decidir concatenar los valores, realizar la multiplicación, división, suma, resta, etc. extraer las subcadenas o comparar los valores de otros campos.

Los valores de las columnas calculadas se "calculan" cada vez que se consulta la tabla, a no ser que se utilice la cláusula PERSISTED consiguiendo de esta

manera almacenar físicamente el valor de la columna calculada. Así pues, usaremos columnas calculadas sin almacenamiento físico cuando el número de consultas sea muy limitado y utilizaremos PERSISTED cuando la información vaya a ser muy consultada, aunque nos penalice el aumento de espacio físico.

Ejemplo

```
--  Unir las columnas Apellido y Nombre, para que se visualicen en una sola columna.

SELECT CONCAT(Apellidos,' ',Nombres) AS nombre_completo

FROM Alumno

--  Otro método con el mismo resultado

SELECT (apellido + '  ' + Nombre) AS 'Nombre Completo'

FROM docentes

--  Calcular el 4% del salario del docente para pago de EPS

SELECT apellido, nombre, salario, salario * 4/100 AS 'Descuento EPS'

FROM docentes

--  colocar formato de numero a las columnas

SELECT FORMAT(vlrmat, '#,###') AS 'Vlr Matricula' , FORMAT((VlrMat * 10/100),'$, #,###')  AS 'recargo'

FROM Matricula
```

CONSULTAS DE AGRUPACION

GROUP BY

La instrucción GROUP BY se usa a menudo con funciones agregadas (COUNT, MAX, MIN, SUM, AVG) para agrupar el conjunto de resultados por una o más columnas.

Funciones Agregadas que devuelven un único valor

AVG: función utilizada para calcular el promedio de los valores de un campo determinado

COUNT: función utilizada para devolver el número de registros de la selección

SUM: función utilizada para devolver la suma de todos los valores de un campo determinado

MAX: función utilizada para devolver el valor más alto de un campo especificado

MIN: función utilizada para devolver el valor más bajo de un campo especificado

```
SELECT ciudad, COUNT(ciudad) AS cantidad, SUM(salario) AS salario,
MAX(salario) AS 'sal Máximo', MIN(salario) AS 'sal mínimo'

FROM docentes

GROUP BY ciudad
```

```
ORDER BY ciudad


SELECT ciudad, facultad, SUM(salario) as salario
FROM docentes
GROUP BY facultad, ciudad


SELECT ciudad, AVG(salario) as 'promedio'
FROM docentes
GROUP BY ciudad
HAVING SUM(salario) >20000000
```

ORDER BY: Presenta el resultado ordenado por las columnas indicadas. El orden puede expresarse con ASC “orden ascendente” y DESC “orden descendente”. El valor predeterminado es ASC.

HAVING

Especifica una condición que debe cumplirse para que los datos sean devueltos por la consulta. Su funcionamiento es similar al de WHERE pero aplicado al conjunto de resultados devueltos por la consulta. Debe aplicarse siempre junto a GROUP BY y la condición debe estar referida a los campos contenidos en ella.

Con HAVING podemos establecer una condición sobre un grupo de registros, algo muy importante es que HAVING acostumbra a ir acompañado de la cláusula GROUP BY. Esto último es así dado que HAVING opera sobre los grupos que nos "retorna" GROUP BY.

```
-- Agrupar por facultad y mostrar la suma, el promedio, el máximo y el mínimo de la columna salario.

-- Ordenar la columna Total Salario, desde el Mayor al Menor.

SELECT facultad,

    SUM(salario) AS 'Total Salario', FORMAT(AVG(salario),'###,###') AS Promedio,

    MAX(salario) AS 'Salario Max', MIN(salario) AS  'Salario Min'


FROM docentes

GROUP BY facultad

HAVING AVG(salario) >=3000000

ORDER BY 'Total Salario' DESC
```

REFERENCIAS BIBLIOGRAFICAS

https://www.aulaclic.es/sqlserver/t_2_3.htm
http://sql.11sql.com/sql-select.htm
http://deletesql.com/viewtopic.php?f=5&t=5
http://basededatos.umh.es/sql/sql02.htm
http://sql-principiantes.blogspot.com/
https://www.ibm.com/support/knowledgecenter/es/SSEPGG_11.1.0/com.ibm.db2.luw.sql.ref.doc/doc/r0059224.html
https://www.campusmvp.es/recursos/post/Fundamentos-de-SQL-Como-realizar-consultas-simples-con-SELECT.aspx
https://sqltrainning.wordpress.com/2014/12/21/que-es-select/
http://www.mug-it.org.ar/332353-Como-manejar-las-fechas-en-Sql-Server.note.aspx
https://www.lawebdelprogramador.com/cursos/archivos/ManualPracticoSQL.pdf
https://www.manualsqlserver.com/?p=185
http://www.tutorialesprogramacionya.com/sqlserverya/temarios/descripcion.php?cod=33&punto=&inicio=
http://m.sql.11sql.com/sql-funcion-having.htm
http://carmoreno.github.io/sql/2017/02/09/Diferencia-entre-having-y-where/
https://www.1keydata.com/es/sql/sql-having.php
https://es.khanacademy.org/computing/computer-programming/sql