APUNTES TEMA 6 PARTE 2

-SENTENCIA SELECT

Cláusulas orden de Ejecución

SELECT - Cuarta en Ejecutarse (Elegimos las columnas que quiero que se muestran)

FROM - Primera en Ejecutarse (Elegir dónde están los datos)

WHERE - Segunda en Ejecutarse (Que filas quiero quedarme)

ORDER BY - Tercera en Ejecutarse (Orden en que quiero que se muestran los datos, No disminuye el número de resultados (Filas) que se obtienen)

Cláusulas nuevas

Tercera en ejecutarse - GROUP BY (Hacer grupos para aquellos que coincidan en un campo o conjunto de campos, después de GROUP by ya no hay filas individuales en los resultados, se agrupan en filas los atributo con un valor distinto cada uno)

Cuarta en ejecutarse después del group by - HAVING (Filtra grupos del group by)

Si el enunciado pone ordenar por cada es = Group by

Ejemplo

QUEREMOS SABER CUÁNTOS EMPLEADOS HAY POR CADA DEPARTAMENTO, PERO SOLO DE LOS EMPLEADOS CON HIJOS

```
select select nomde, count(distinct extelem)  as 'Extension distinta', count(*) as 'Numero Empleados', sum(salarem), avg(salarem), max(salarem)
from empleados
where numhiem > 0
group by numde
order by count(*);
```

Se pone cuando hay group by:

select numde, count(distinct extelem) as 'Extension distinta', count(*) as 'Numero Empleados', sum(salarem), avg(salarem), max(salarem)

Cada uno tendra un as 'Nombre'
count(*) (cuenta el número de filas hay en cada grupo, puede ser: *, nomem, numde , etc. Normalmente se usa el *)

count(distinct extelem) con el distinct no se repiten filas

sum(salarem) (Devuelve la suma de las filas indicadas, en este caso el salarem)

avg(salarem) (Devuelve la media del salario de las filas)

min(salarem) (Devuelve el salario minino de las filas)

max(salarem) (Devuelve el salario máximo de cada fila)

El order by no puede tener un atributo porque daría error, se pondría en su caso

order by count(*) ; o otro de los modificadores del select anterior

¿Cuántos empleados tengo?

Con cuantos, se refiere a que hay que usar group by en el enunciado

Tenemos que identificar antes cuántos grupos hay

```
Select count(*)
from empleados;
```

(En este caso solo saldria el total de filas de la tabla empleados)

```
Select count(*)
from empleados;
group by numde;
order by count(*)
```

Importante cuando hay parámetro de entrada es recomendable utilizar dicho parámetro de entrada en el where de la siguiente forma

```
where autor= p_autor;
```

El autor es la fila o atributo y p_autor es el parámetro de entrada

FUNCIONES (FUNCTIONS)

Procedimientos almacenados y función son código precompilado

Las funciones void son procedimientos almacenados vacíos, sirven para devolver un valor.

RECUERDA QUE SI TIENES QUE DEVOVLER MÁS DE UNA COSA SOLO SE PUEDE USAR PROCEDIMIENTOS, NUNCA UNA FUNCION (LAS FUNCIONES SÓLO DEVUELVEN UNA COSA)

RECUERDA, SE DEBE INCLUIR DETERMINISTIC, ENTRE RETURNS TIPODATO Y BEGIN. PORQUE PUEDE DAR ERROR POR VERSION DE MYSQL

Estructura

```
delimiter $$
create function mifuncion(
depto int
(En las funciones solo hay parametros de entrada, no se usa ni in, out, inout.
)

DETERMINISTIC -- Se incluye para que no de error por version MYSQL

returns int (tipo de valor que devolvemos, puede ser decimal o otros tipos)
begin
return (
select count(*)
from empleados
where numde = depto
);
end $$
delimiter ;

Las funciones se llaman de la siguiente forma
select mifuncion();

drop function if exists mifuncion $$ (Elimina la funcion)

select mifuncion(110); (Muestro la funcion, no se usa call)

En caso de que no se use return sería así:

delimiter $$
create function mifuncion(
depto int
)
DETERMINISTIC
returns int
begin
declare totalempleados int default 0; (Se usa la variable declare)
select count(*)
from empleados
where numde = depto
return totalempleados;
end $$
delimiter ;

----------
create procedure ejr(out maxsalario decimal(7,2))

set maxsalrio = (establecemos el valor de la variable de salida)
```

RECUERDA

Cuando haya que devolver en un enunciado, parámetro de salida con procedimientos almacenado y con funciones se utliza return

Siempre hay que respetar los ordenes de los parametros de salida.

Con funciones no se pueden devolver dos valores

El sum, count etc se puede usar en cualquier parte salvo en el from y where.

Las funciones de agregado sum, avg, etc. No se pueden anidar

Las funciones de agregado se suelen usar con un group by, dichas funciones de agregado no van nunca en el where

% o MOD obtener el resto de una division entera

Establece la variable global cadena como hola y le quita los espacios por la izquierda.

```
set @cadena = 'hola';
select concat(ltrim(@cadena), ' y adios');
```

<u>OPERANDO LIKE, LOCATE, SUBSTRING</u>

El operando like busca la palabra sanchez en el campo nompropietarios

```
select *
from propietarios
where nompropietario like '%sanchez%';
```

Busca la localizacion de la palabra sanchez, es decir su posicion contando cada letra

```
select nompropietario, locate('sanchez', nompropietario)
from propietarios;
```

Muestra los nombres que tienen Sanchez segun la posicion que es mayoria a 1. La posicion siempre empieza en 1

```
select nompropietario, locate('sanchez', nompropietario)
from propietarios
where locate('sanchez', nompropietario) > 0;
```

(Busca la posicion donde haya un espacio en blanco)

```
select nompropietario, locate(' ', nompropietario)
from propietarios;

select substring('sanchez', 5, 4);
```

Coge el String que sea sanchez donde la posicion sea 5 y ocupe 4 posiciones

Busca la primera palabra del nombre por la izquierda, utilizando -1 para quitar el espacio en banco

select nompropietario, locate(‘ ’, nompropietario), left(nompropiertario, locate(‘ ’, nompropietario) -1)
from propietarios;

substring(nompropietario, locate(‘’, nompropietario)-1) as nombre,

Lo mismo que left, busca la palabra antes del espacio, por orden empieza por la izquierda. (Busca solo el nombre, hasta la posicion 7)

select left(‘Alberto garzon rodriguez’, 7) as nombre;

Busca el primer apellido por posicion y numero de posiciones

select substring(‘Alberto garzon rodriguez’, 8, 6) as pimerapellido;

El 7 o el 8,6 se pueden reemplazar por un substring de la siguiente forma.

substring(nompropietario, locate(‘ ’, nompropietario) -1)

Lo que busca es el primer espacio

locate(‘ ’, nompropietario) +1)

DATE_ADD y DATE_FORMAT

Averiguar la fecha de salida de los clientes para nuestras reservas. Añade los dias de estancia a la fecha de estancia y despues lo devuelve

select fecinistancia, numdiasestancia, date_add(feciniestancia, interval numdiaesstancia day)
from reservas

Date_format sirve para mostrar la fecha en el formato que deseemos

date_format(date_add(feciniestancia, interval numdiasestancia day),'%e/%m%/%Y')
from reservas

select fecinistancia, numdiasestancia, date_add(feciniestancia, interval numdiaesstancia day),
date_format(date_add(feciniestancia, interval numdiaesstancia day, ‘%Y%’);
from reservas

JOINS

join o innerjoin (join, no tiene orden)

right o left join (Sigue el orden de las tablas, teniendo en cuenta que siempre va primero la tabla con más datos, que sería la tabla de la izquierda)

left join (Añade la tabla de la la izquierda mantenendolo intactos. Refiriendose a los campos situados a la izquierda de la FK)

```
select nomem, nomde
from empleados right join departamentos
on empleados.numde = departamentos.numde;
```

IMPORTANTE

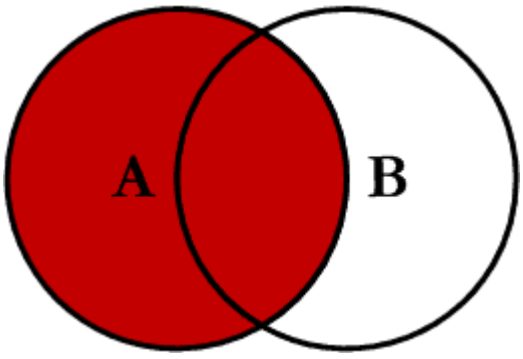
Hay que usar left or right dependiendo de la tabla que queramos mostrar, aquella que tiene mas valores que la otra

Explicación más detallada de los joins

Cláusula LEFT JOIN

A diferencia de un INNER JOIN, donde se busca una intersección respetada por ambas tablas, con LEFT JOIN damos prioridad a la tabla de la izquierda, y buscamos en la tabla derecha.

Si no existe ninguna coincidencia para alguna de las filas de la tabla de la izquierda, de igual forma todos los resultados de la primera tabla se muestran.



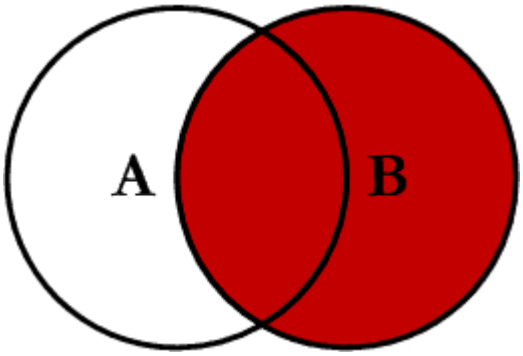
He aquí una consulta de ejemplo:

```
SELECT
  E.Nombre as 'Empleado',
  D.Nombre as 'Departamento'
FROM Empleados E
LEFT JOIN Departamentos D
ON E.DepartamentoId = D.Id
```

- **La tabla Empleados es la primera tabla en aparecer en la consulta (en el FROM), por lo tanto esta es la tabla LEFT (izquierda), y todas sus filas se mostrarán en los resultados.**
- **La tabla Departamentos es la tabla de la derecha (aparece luego del LEFT JOIN). Por lo tanto, si se encuentran coincidencias, se mostrarán los valores correspondientes, pero sino, aparecerá NULL en los resultados.**

Cláusula RIGHT JOIN

En el caso de RIGHT JOIN la situación es muy similar, pero aquí se da prioridad a la tabla de la derecha.



De tal modo que si usamos la siguiente consulta, estaremos mostrando todas las filas de la tabla de la derecha:

```
SELECT
  E.Nombre as 'Empleado',
  D.Nombre as 'Departamento'
FROM Empleados E
RIGHT JOIN Departamentos D
ON E.DepartamentoId = D.Id
```

La tabla de la izquierda es Empleados, mientras que Departamentos es la tabla de la derecha.

La tabla asociada al FROM será siempre la tabla LEFT, y la tabla que viene después del JOIN será la tabla RIGHT.

Entonces el resultado mostrará todos los departamentos al menos 1 vez.

Y si no hay ningún empleado trabajando en un departamento determinado, se mostrará NULL. Pero el departamento aparecerá de igual forma.

<u>RECUERDA</u>

SOLO SE PUEDE PONER UN INTO, NUNCA DOS.

SE PUEDE HACER UN
INTO NOMZONAS, DESCRIPZONAS

NUNCA HAGAS
INTO NOMZONAS, INTO DESCRIPZONAS

Otras Funciones

-Con like se buscan los campos que empiecen por dicha letra, n este caso B

```
SELECT author FROM lyrics WHERE author LIKE 'B%';
```

-Otra forma de usarlo

```
WHERE LEFT (name_field, 1) = 'B';
```

-Obtiene el campo al revés

select REVERSE (proveedores.telefono)

-Obtener el campo numerico elevado al cuadrado o al cubo etc.

select POW(productos.preciounidad, 2)
select POW(productos.preciounidad, 3)

-Obtiene el mes de la fecha actual

SELECT MONTH(CURDATE());

-Compara la fecha de entrega con la fecha actual. Siempre dos fechas

where DATEDIFF(pedidos.fecentrega, CURDATE());

Sustituye en el campo descripcion donde la palabra tarta sera pastel. Reemplaza el string tarta por pastel en el campo descripcion. Para cambiar con un replace, usa el update y set. El select no sirve, solo muestra no cambia o reemplaza

update productos

set descripcion= replace(descripción, 'tarta', 'pastel') as 'Pastel'

-Pone en mayúsculas el campo

UPPER(productos.descripcion)

//Ordena por el tamaño del campo de menor a mayor

order by CHAR_LENGTH(descripcion);

//Obtener el número redondeado a dos decimales

select ROUND(productos.preciounidad, 2)

-Repite un campo dos veces o las que haga falta.

select repeat(productos.descripcion,2)

-Separa nombre primer apellido y segundo apellido

```
SELECT SUBSTRING_INDEX(SUBSTRING_INDEX(nomcontacto, ' ', 1), ' ', -1) AS 'Nombre',
SUBSTRING_INDEX(SUBSTRING_INDEX(nomcontacto, ' ', 2), ' ', -1)  as 'Primer apellido',
SUBSTRING_INDEX(SUBSTRING_INDEX(nomcontacto, ' ', 3), ' ', -1) AS 'Segundo apellido'
from proveedores;
```

//Muestra la fecha sin los zeros en el dia y mes con %e para dia y %c para mes

DATE_FORMAT(empleados.fecnaem, "%e/%c/%Y") as 'Fecha'

%y para años mostrando dos dígitos
%M muestra el mes en palabras
%b muestra el mes abreviado en palabras

-Cambia el idioma para Español a la hora de mostrar fechas

SET lc_time_names = 'es_ES';

- ENCRYPT O PASSWORD sirve para cifrar una contraseña. Es mejor usar password

Las funciones escalares son las funciones que usa mysql.

La funcion pow y power es lo mismo. Puedes usar cualquiera.

<u>DECLARACIONES</u>

```
CREATE FUCTION EJE(EMPLEADO INT)
RETURNS CHAT(8)
```

-Las declaraciones, indican el comportamiento del programa y lo que hace. Va despues del return y antes del begin. Son opcionales, no es obligatorio ponerlas

LANGUAGE SQL (Utiliza lenguaje mysql)

DETERMINISTIC (No va a tener resultados inesperados)

reads sql data (Solo hace lectura, es decir usa solo select)

COMMENT 'Aquí va un comentario, se registra en el sistema'

BEGIN

El substring_index buscaria el último espacion en blanco porque tiene valor negativo. Si es positivo sería el primer espacio

SUBSTRING_INDEX('alberto garcia', ' ', -1)

Ejemplos de DATE_FORMAT y como mostrar distintos formatos de fechas

```
/*
8. Obtener el nombre completo de los empleados y la fecha de nacimiento con los siguientes formatos:
a. "05/03/1978"
b. 5/3/1978
c. 5/3/78
d. 05-03-78
e. 05 Mar 1978
*/

-- a
select concat_ws(' ',empleados.nomem,  empleados.ape1em, empleados.ape2em) as 'Nombre completo', DATE_FORMAT(empleados.fecnaem, "%d/%m/%Y") as 'Fecha'
from empleados;

-- b
select concat_ws(' ',empleados.nomem,  empleados.ape1em, empleados.ape2em) as 'Nombre completo', DATE_FORMAT(empleados.fecnaem, "%e/%c/%Y") as 'Fecha'
from empleados;

-- c
select concat_ws(' ',empleados.nomem,  empleados.ape1em, empleados.ape2em) as 'Nombre completo', DATE_FORMAT(empleados.fecnaem, "%e/%c/%y") as 'Fecha'
from empleados;

-- d
select concat_ws(' ',empleados.nomem,  empleados.ape1em, empleados.ape2em) as 'Nombre completo', DATE_FORMAT(empleados.fecnaem, "%d-%m-%y") as 'Fecha'
from empleados;

-- e
```

select concat_ws(' ',empleados.nomem, empleados.ape1em, empleados.ape2em) as 'Nombre completo', DATE_FORMAT(empleados.fecnaem, "%d %b %Y") as 'Fecha'
from empleados;

EXPRESIONES REGULARES - PATRONES

Por ejemplo like busca por un patrón, la cadena de texto tiene que llevar porcentaje al principio y final.

select *
from propietarios
where nompropietario like '%sanchez%';

Busca a los que no se llamen antonio

where nompropietario not like '%sanchez%';

Que empieze en 29

where codpostal like '29%';

Que finalice con 54

where codpostal like '%54';

Que empiece por un caracter cualquiera y despues una a

where codpostal like '_a%';

Empieza en 29, hay dos caracteres cualquiera y termina en 4

where codpostal like '29__4';

% (cualquier cadena de 0 o más caracteres)
_(cualquier carácter)

EXPRESIONES REGULARES

REGEXP

'^[XY]'

Que empiece por XY, lo que se pone entre corchetes hace referencia a un solo caracter.

Ej con numeros, tambien serviria con letras
'^[1-6]' uno al 6
'^[16]' sea 1 o 6
'^[aeiou]'

where nomcontacto regexp '^[XY]';

‘^[‘^XY]’; (Siginica que no empiece por, como si fuera un not like. El gorro siempre dentro del corchete para esta condicion

el ^ solo hace referncia a un carácter o []

Si queremos que termine por, se usa $

where nomcontacto regexp ‘[XY]$’;

Para que busque la cadena de texto atocha

where nomcontacto regexp ‘atocha’;

Que empieza por una vocal y sigue con la letra l o la n

where nomcontacto regexp ‘^[aeiou][ln]’;

Que empiece por vocal y que se repite dos veces, es decir que tenga dos vocales seguidas

where nomcontacto regexp ‘^[aeiou]{2}’;

Que empiece por e, despues 0, 1 o más de una b
where nomcontacto regexp ‘^eb*’;

Que empiece por e, despues 1 o más de una b y despues una a
where nomcontacto regexp ‘^eb+a’;

Cualquier carácter se utiliza un . Un punto en vez del guión bajo

Que empiece e incluya ambos caracteres es decir que la palabra tenga au. () incluyen todos los caracteres

where nomcontacto regexp ‘^(au)’;

Que empieze o por a o por e
where nomcontacto regexp ‘^a|^e’;

Sería lo mismo
where nomcontacto regexp ‘^[ae]’;

not regexp (Sirve para negacion, como un not like)

FUNCION REGEXP_LIKE

Es una funcion más compleja que REGEXP

EJ: busca por el campo nomem, donde empiece por e seguido de b, que aparezca 0 o solo una vez (Para se utiliza la ?) y seguido de a

where regexp_like(nomem, '^eb?a');

'^.' busca cualquier dato que empiece por culaquier carácter

Que empiece por 2 seguido de 9, el 0 aparece repetido dos veces y un número distinto de 0

where tal rlike '290{2} [^0]';

Que empiece por 2 seguido de 9 y 0 y seguido de la siguiente condicion Que tenga un 1 o un 2 seguido de un 0 o tenga un 0 o un 1 seguido de un número del 1 al 9

where tal rlike '290([12]0 | [01][1-9]);

Carácter de escape \\, si se utiliza de esta forma\. es para que no interprete como cualquier caracer sino como como carácter de escape (Es decir que lo trata como un . no como cualquier carácter).

Contiene @ seguido de una letra de la a la z que aparece 0 una sola vez seguido del caracter de escape (un punto) y que finaliza con una de las condiciones del paréntesis

'@[a-z]*\\.(com|net|es|eu)$;

**Para que aparezca un símbolo tipo $, . etc utiliza carácter de escape **

IMPORTANTE

Distinto en un enunciado es igual a distinct

VISTAS Y SUBSELECT

SUBSELECT

```
-- BUSCA EMPLEADOS QUE COMPARTEN DEPTO CON EL EMPLEADO 120

SET @DEPTO = (SELECT NUMDE FROM EMPLEADOS WHERE NUMEM = 120); -- SUBSELECT, UNA QUERY DENTRO DE OTRA EXPRESION

SELECT NUMEM, NOMEM
FROM EMPLEADOS
WHERE NUMDE = @DEPTO;
```

TAMBIÉN SE PUEDE HACER ASÍ

```
SELECT NUEM, NOMEM
FROM EMPLEADOS
WHERE NUMDE = (SELECT NUMDE FROM EMPLEADOS WHERE NUMEM = 120);
```

NO SE PUEDE COMPARAR EL NUMDE CON DOS COLUMNAS O VALORES, O DOS FILAS.

Los subselect siempre van entre paréntesis

create temporary table if no exists (tabla temporal que solo se mantiene durante la sesion)

(Podemos comparar distintas bases de datos y acceder a sus tablas y filas con el punto)

```
update empleados
set extelem = (select empresaclase.empleados.extelem from empresaclase.empleados
where empresaclse.empleados.numem = empresaclase2021.empleados.numem);
```

Solo se utilizan para saber qué datos tenemos en nuestra base de datos

<u>VISTAS</u>

Es un objeto parecido a una tabla, sirven para oultar informacion para determinados usuarios.

Las vistas nos muestran los datos de una tabla

Ejemplo para crear una vista

```
create view empleadosPorCentro
(numeroEm, nombreEm, nombreCen) (Si no ponemos nombre, es decir obviamos el paréntesis y
sus contenidos, tomará por defecto los campos del select)

as
-- Irian las sentencias a continuacion

select numem, nomeme, nomce
from empleados join departmentos on empleados.numde = departemtnos.numde
join centros on departamentos.numce = centros.numce;

Podemos hacer un select como si se tratara de una tabla de datos
select * from empleadosPorCentro
```

Actualizamos la tabla y modificamos el campo donde su valor numerico es 120

```
update empleados
set nombre = ‘’
where número = 120;
```

UNIÓN

Para unir varias filas con UNION (MISMO NÚMERO DE COLUMNAS Y DOMINIOS COMPATIBLES)

EL JOIN SOLO SIRVE PARA UNIR COLUMNAS

```
select concat_ws(‘’, nomcli,ape1cli,ape2cli),tlf_contacto
from clinetes
UNION
select nompropietario, tlf_contacto
from propietarios;
```

Utilizando una vista

```
CREATE VIEW LISTA USUARIOS
(nomuser, tlfuser)
as
select concat_ws(‘’, nomcli,ape1cli,ape2cli), tlf_contacto
from clinentes
UNION
select nompropietario, tlf_contacto
from propietarios;
```

Que contenga un 9 en el telefono de usuario

```
SELECT * FROM LISTAUSUARIOS
where tlfuser like ‘%9%’;
```

Operadores cuantificados some, any, all, in, exists, NOT.

Estos operadores van en el where

El operador IN, se utiliza sin operador logicos es decir sin > =, etc

SOME = ALGUNO
ANY = CUALQUIERA

SOME Y ANY SON EQUIVALENTES

ALL = TODOS

IN = SE ENCUENTRE DENTRO DEL CONJUNTO DE RESULTADOS (QUE COINCIDA CON ALGUNO)

EXISTS = EXISTE

EL NOT SE PUEDE USAR CON TODOS
NOT = NO SE ENCUENTRA

SINTAXIS

EXPRESION [NOT] =|<>|<|<=|>|>= SOME, ANY, ALL (SUBQUERY)

EJ

EXPRESION [NOT] IN (SUBQUERY)

Where numde = All (Select numde from empleados where…)

numde >= ALL (SUBQUERY)

salarem*0,5 = some (subquery)

avg(salarem) in (subquery)

SINTAXIS IN

EXPRESION [NOT] IN (SUBQUERY)

EQUIVALENCIAS EN EL USO DE OPERADORES CUANTIFICADOS

EXPRESION1 = SOME (SUBQUERY)
EXPRESION1 = ANY (SUBQUERY)
EXPRESION1 IN (SUBQUERY)

SON IGUALES

SON TAMBIÉN EQUIVALENTES

EXPRESION1 <>ALL (SUBQUERY)
EXPRESION1 NOT IN (SUBQUERY)

EJEMPLOS

```
* busca empleados con el mismo nombre que alguno del departamento 122*/
select empleados.nomem, empleados.numde
from empleados
where nomem = SOME(select empleados.nomem from empleados where empleados.numde = 122);

/* busca empleados cuyo salario sea superior a todos los del xdepto 122 */
select empleados.salarem
from empleados
where empleados.salarem > ALL(select empleados.numde from empleados where
empleados.numde = 122);

/* busca empleados cuyo extension telefonica sea diferente a los del dpto 122 */
```

select empleados.extelem
from empleados
where empleados.extelem not in (select empleados.extelem from empleados where empleados.numde = 122);

HAVING

CLAUSULA HAVING. SENTENCIA SELECT (REPASO)

RECUERDA: SENTENCIA SELECT ==> (FORMADA POR CLÁUSULAS)

ORDEN DE ESCRITURA DE LAS CLÁUSULAS (SINTAXIS):

SELECT ==> que quiero ver

FROM ==> donde está

WHERE ==> qué filas me interesan (filtro sobre filas)

GROUP BY ==> hace grupos sobre ciertos datos para trabajar con ellos

HAVING ==> que grupos me interesan (filtro sobre grupos)

ORDER BY ==> Orden en el que quiero mostrar las columnas de claúsula SELECT

ORDEN DE EJECUCIÓN DE LAS CLÁUSULAS (INTERPRETACIÓN)

1. FROM ==> El MOTOR Busca los datos que necesitamos
2. WHERE ==> El MOTOR filtra las filas que queremos:
   Comprueba las filas (fila a fila) cuyo resultado es verdadero para los predicados de where.
   DESCARTA LAS QUE DEVUELVEN FALSO O DESCONOCIDO
3. GROUP BY ==> Con las filas que han quedado tras where, el MOTOR hace los grupos.

OJO! A partir de aquí no hay filas individuales

4. HAVING ==> El MOTOR filtra los grupos que queremos:
   Comprueba los grupos (grupo a grupo) cuyo resultado es verdadero para los predicados de where. DESCARTA LOS QUE DEVUELVEN FALSO O DESCONOCIDO.
5. ORDER BY ==> El motor ordena los datos según nos interesa
6. SELECT ==> El motor prepara las columnas o expresiones que queremos obtener

---

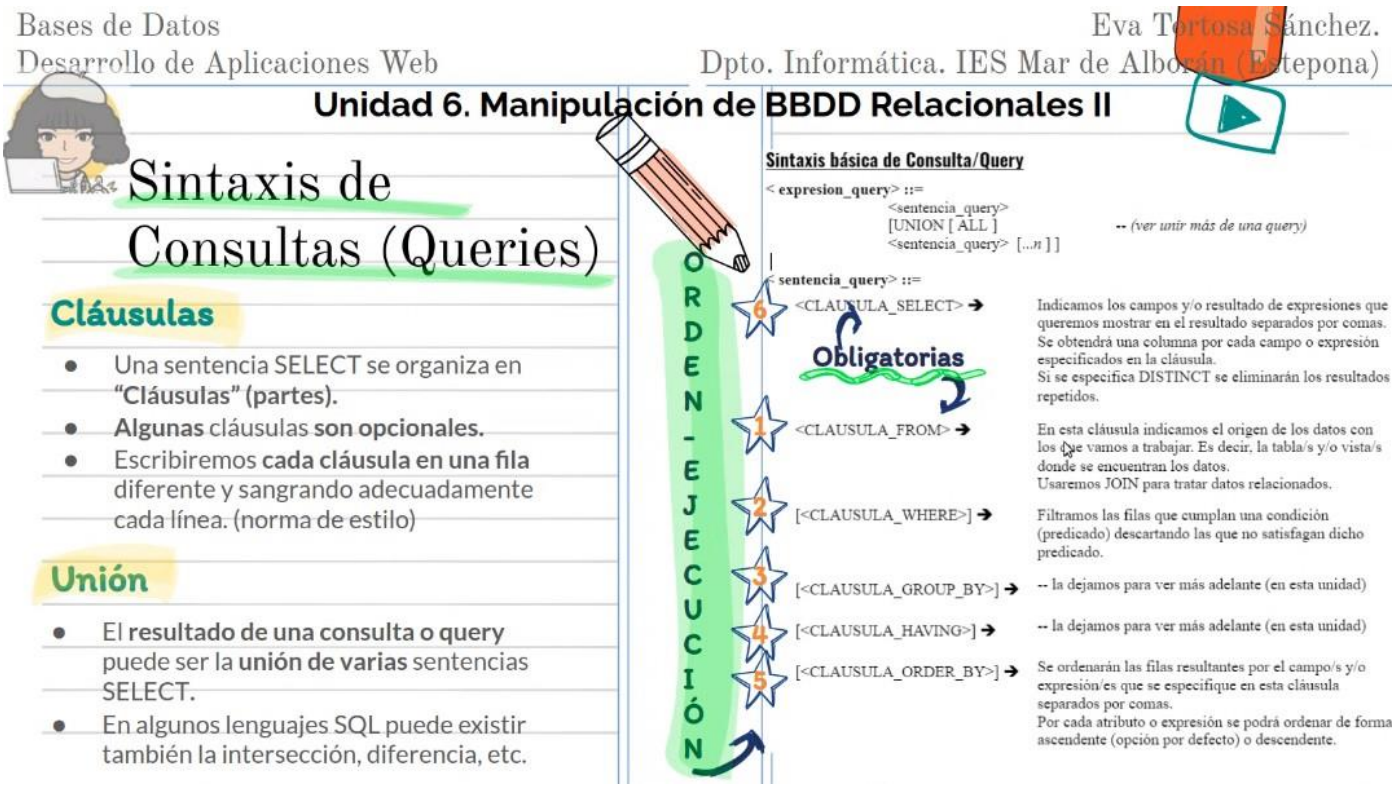
A partir del having se pueden usar funciones de agregado, del group by para arriba no se pueden usar. Por lo tanto se puede usar un count(*) tanto en el having como order by

Sirve para filtrar grupos, se usa siempre que haya group by

Filtra la media de los salarios del grupo que son mayores al salario medio. Usando un subselect despues del having

select numde, count(distinct extelem), avg(salarem)
from emepleados
group by numde
having avg(salarem) > (select avg(salarem) from empleados);

ORDEN DE EJECUCION COMPLETO



EXISTS

Es un operador que sirve para saber si existe la condicion del where si es veradero mostrará 1, si es nulo será falso y mostrará 0

Es como un booleano binario

1 = verdadero
0 = falso o nulo

Busca aquellos centros que contenga atocha utilizando exists para saber si existe o no

```
select nomce
from centros
where exists (select * from centros where dirce like '%viento%');
```

UNION ALL

Con la union all se repiten los datos, si están en ambas aparecerán repetidas. Con union mormal solo aparece uno

```
-- EJERCICIO PROPUESTO:

-- PARA LA BD DE PROMOCIONES:
/*
```

```
QUEREMOS TENER PREPARADO SIEMPRE (VISTA) UN LISTADO CON LOS PRECIOS A DÍA DE
HOY (CUANDO SE CONSULTE) DE LOS ARTÍCULOS
NECESITO: refart, nomarticulo preciobase, pecioHoy, codcat

*/
create view catalogoprecios

(referencia, descripcion, preciobase, precioHoy, categoria)

as


select refart, nomart, preciobase, precioventa, codcaat

from articulos

where refarticulo not in

      (select catalogospromos.refarticulo

      from catalogospromos join promociones on catalogospromos.codpromo =
promociones.codpromo
      where curdate() between promociones.fecinipromo

                        and date add (promociones.fecinipromo,
 interval promociones.duracion day)

      )


union all /* se repiten */
-- union  /* no se repiten */


select articulos.desarticulo, catalogospromos.precioventa

from catalogospromos join promociones on catalogospromos.codpromo =
promociones.codpromo
      where curdate() between promociones.fecinipromo

                        and date add (promociones.fecinipromo,
 interval promociones.duracion day)
```

DIFERENCIAS

UNION La instrucción UNION se utiliza para seleccionar los datos relacionados entre dos tablas. Las columnas tienen que ser del mismo tipo de datos. Solo se devuelven los valores distintos

UNION ALL La instrucción UNION ALL es similar a UNION con la excepción que se seleccionan todos los valores.

Por lo tanto la diferencia principal entre UNION y UNION ALL es que UNION ALL no eliminará de la consulta los datos duplicados.

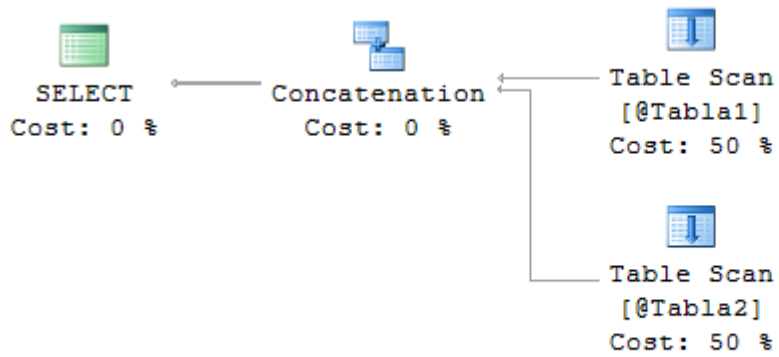
Con respecto al rendimiento UNION realiza un Select DISTINCT sobre los resultados por lo que UNION ALL devuelve los resultados más rápidamente (Comparen los planes de ejecución que agregué al final)

A continuación algunos ejemplos:

DECLARE @Tabla1 TABLE (ColDetalle VARCHAR(10)) INSERT INTO @Tabla1 SELECT 'A' UNION ALL SELECT 'B' UNION ALL SELECT 'C' UNION ALL SELECT 'D' UNION ALL SELECT 'E' DECLARE @Tabla2 TABLE (ColDetalle VARCHAR(10)) INSERT INTO @Tabla2 SELECT 'A' UNION ALL SELECT 'B' UNION ALL SELECT 'C'
Pruebas:

/* UNION ALL */ SELECT * FROM @Tabla1 UNION ALL SELECT * FROM @Tabla2 Order by 1
Resultado y plan de ejecución:

	ColDetalle
1	A
2	A
3	B
4	B
5	C
6	C
7	D
8	E



/* UNION */ SELECT * FROM @Tabla1 UNION SELECT * FROM @Tabla2 Order by 1
Resultado y plan de ejecución:

	ColDetalle
1	A
2	B
3	C
4	D
5	E