

TECNICATURA UNIVERSITARIA EN PROGRAMACIÓN

LABORATORIO DE COMPUTACIÓN II

Unidad Temática 2:
Recuperación de Datos

TEÓRICO

Curso: 1 er Año. 2 ndo Cuatrimestre

Índice

Combinación de resultados de consulta. UNION.....	2
Consultas sumarias.....	3
Consultas agrupadas: Cláusula Group By.....	4
Otras formas de composición: Inner, Left y Right Join	6
Bibliografía	9

Recuperación de datos

Combinación de resultados de consulta. UNION

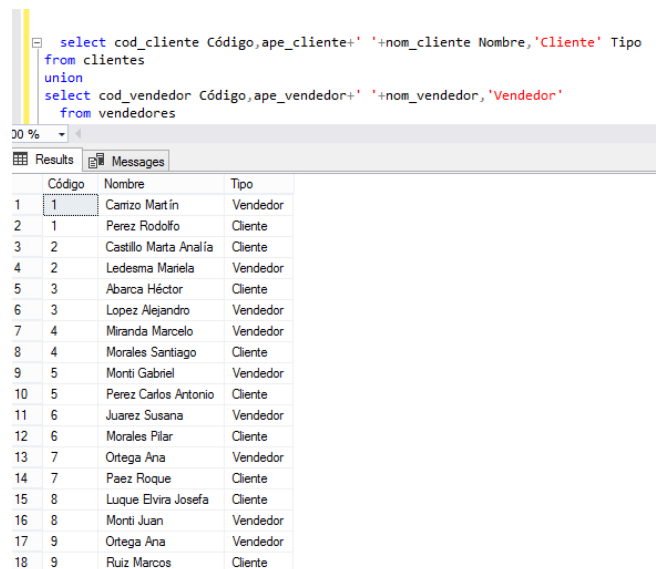
la característica UNION de la sentencia SELECT se utiliza cuando se necesita combinar los resultados de dos o más consultas en una única tabla de resultados totales. Por ejemplo, confeccionar un listado de los clientes y los vendedores indicando a qué grupo pertenece cada uno. Para el listado de clientes utilizaremos la siguiente consulta:

```
select cod_cliente Código, ape_cliente + ' ' + nom_cliente Nombre
from clientes
```

Para la de vendedores, esta otra consulta:

```
select cod_vendedor Código , ape_vendedor + ' ' + nom_vendedor
from vendedores
```

Esto nos estaría dando dos tablas de resultados. Para que ambas consultas aparezcan en una sola tabla de resultado escribiremos lo siguiente, teniendo en cuenta que solo agregaremos alias a la primera consulta y crearemos una columna extra con una constante que indique el origen del registro es decir si es un cliente o un vendedor; a esta nueva columna también le agregaremos un alias.



```
select cod_cliente Código,ape_cliente+' '+nom_cliente Nombre,'Cliente' Tipo
from clientes
union
select cod_vendedor Código,ape_vendedor+' '+nom_vendedor,'Vendedor'
from vendedores
```

	Código	Nombre	Tipo
1	1	Camizo Martín	Vendedor
2	1	Perez Rodolfo	Cliente
3	2	Castillo Marta Analía	Cliente
4	2	Ledesma Mariela	Vendedor
5	3	Abarca Héctor	Cliente
6	3	Lopez Alejandro	Vendedor
7	4	Miranda Marcelo	Vendedor
8	4	Morales Santiago	Cliente
9	5	Monti Gabriel	Vendedor
10	5	Perez Carlos Antonio	Cliente
11	6	Juarez Susana	Vendedor
12	6	Morales Pilar	Cliente
13	7	Ortega Ana	Vendedor
14	7	Paez Roque	Cliente
15	8	Luque Elvira Josefa	Cliente
16	8	Monti Juan	Vendedor
17	9	Ortega Ana	Vendedor
18	9	Ruiz Marcos	Cliente

Restricciones del UNION

Hay varias restricciones sobre las tablas que pueden combinarse con una operación UNION:

- Ambas tablas deben contener el mismo número de columnas.
- El tipo de datos de cada columna en la primera tabla debe ser el mismo que el tipo de datos de la columna correspondiente en la segunda tabla
- Ninguna de las dos tablas pueden estar ordenadas con la cláusula ORDER BY. Se puede ordenar el conjunto combinado que es el resultado de la operación UNION agregando esta cláusula al final de la última consulta SELECT y se debe identificar las columnas por número.

En el ejemplo, cuente la cantidad de columnas que tiene cada una de las consultas: 3 columnas cada una. Ahora observe la primer columna es un Integer en ambas consultas, la segunda es Varchar y la tercera Varchar. Y por último si quisiera ver el listado ordenado de alguna manera, por ejemplo, primero los clientes y luego los vendedores la cláusula order by será la última línea de todas las consultas.

```
select cod_cliente Código,ape_cliente+' '+nom_cliente Nombre,'Cliente' Tipo
from clientes
union
select cod_vendedor Código,ape_vendedor+' '+nom_vendedor,'Vendedor'
from vendedores
order by 3
```

Cada consulta select puede incluir todas las cláusulas ya vistas anteriormente (o las que se verán posteriormente) siempre y cuando se respete las restricciones enunciadas en los párrafos anteriores.

UNION ALL.

La operación UNION podría producir resultados que contuvieran filas duplicadas, pero por omisión se eliminan. Si se desea mostrar la filas duplicadas en una operación UNION, se puede especificar la palabra clave ALL luego de la palabra UNION.

Consultas sumarias

SQL permite sumarizar datos de la base de datos mediante un conjunto de funciones de columnas o funciones de agregado. Estas funciones en SQL aceptan una columna entera de datos como argumentos y produce un único dato que sumaria la columna.

Las funciones de columnas o funciones de agregado son:

SUM(columna) calcula el total de una columna

AVG(columna) calcula el valor promedio de una columna

MIN(columna) encuentra el valor más pequeño en una columna

MAX(columna) encuentra el valor mayor en una columna

COUNT(*) cuenta las filas de resultados de la columna

Se quiere saber el total de la factura Nro. 236, la cantidad de artículos vendidos, cantidad de ventas, el precio máximo y mínimo vendido

```

Select sum(pre_unitario*cantidad) Total, sum(cantidad) 'Cant.de Artículos',
count(*) 'Cant.de Items', max(pre_unitario) 'Mayor precio', min(pre_unitario) 'Menor precio'
from detalle_facturas
where nro_factura = 236

```

	Total	Cant.de Artículos	Cant.de Items	Mayor precio	Menor precio
1	1351.00	52	2	31.00	22.30

Contar registros

Como ya se dijo antes Count(*) cuenta el número de registros que da por resultado una consulta, incluyendo los valores nulos; si lo que se quiere es contar la cantidad de valores reales, no nulos de una columna se puede utilizar Count(columna). Por ejemplo:

```

Select count(*) 'cantidad de clientes',
count([e-mail]) 'cantidad de clientes con e-mail conocido'
from clientes

```

	cantidad de clientes	cantidad de clientes con e-mail conocido
1	12	3

La primer columna daría la cantidad de clientes que se tienen, es decir la cantidad de registros de la tabla clientes la segunda columna devolvería la cantidad de clientes con dirección de correo electrónico, es decir, la cantidad de registros cuyo campo e-mail no contiene valor null.

Count_big

Retorna la cantidad de registros en forma similar a la función Count(*) , la diferencia es que Count_big(*) retorna un valor "bigint" y Count(*), un valor "int".

De forma análoga a Count(columna), Count_big(columna) retorna la cantidad de registros cuyo valor en el columna especificado no es nulo,.

Count(distinct columna)

Todas las opciones de Count anteriores cuenta todos los registros aunque estos tengan valor duplicado, si lo que se desea es que el valor devuelto no cuente valores duplicados se puede combinar esta función con Distinct. Por ejemplo:

```

Select count(cod_cliente) 'cantidad de facturas',
       count(distinct cod_cliente) 'cantidad de clientes'
from facturas

```

	cantidad de facturas	cantidad de clientes
1	540	10

La primer columna devolvería la cantidad total de registros de la tabla factura que tengan un cliente (no contaría registros donde el código de cliente sea null) y en este caso daría el mismo resultado que count(*) o count(nro_factura).

La segunda columna devolvería la cantidad de clientes (distintos) que alguna vez se le ha facturado independientemente de la cantidad de facturas que se le ha registrado a cada uno es decir no cuenta los valores duplicados de cod_cliente.

Promedio

Se necesita conocer el promedio de facturación por factura el año pasado.

Si utilizamos esto: select avg(pre_unitario*cantidad) from detalle_facturas

La función de agregado AVG(columna) suma el valor de la columna y la divide por la cantidad de registros que contenga la consulta, en otras palabras lo que hace es la aplicación de esta operación: SUM(columna)/COUNT(*) con lo que no estaría dando el promedio por factura sino por detalle de factura para dar la solución a lo que pide el punto 4 habría que utilizar en la consulta anterior: SUM(pre_unitario*cantidad)/ COUNT(distinct nro_factura)

Aquí lo resolvemos dando ambos resultados para que se entienda mejor:

```

select avg(pre_unitario*cantidad) 'promedio por detalle de factura',
       sum(pre_unitario*cantidad)/count(distinct d.nro_factura) 'promedio por factura'
from detalle_facturas d, facturas f
where d.nro_factura=f.nro_factura
and year(fecha)=year(getdate())-1

```

promedio por detalle de factura	promedio por factura
538.307821	963.571000

Consultas agrupadas: Cláusula Group By

Las consultas sumarias son como totales finales de un informe; si lo que se necesita es sumarizar los resultados de la consulta a un nivel de subtotal utilizamos la cláusula Group By de la sentencia Select. Es decir, agrupar registros según los valores de una o más columnas y obtener totales de cada grupo.

Una consulta que incluya la cláusula Group By se denomina consulta agrupada, ya que agrupa los datos de las tablas fuente y produce una única fila sumaria por cada grupo de filas. Las columnas indicadas en la cláusula Group By se denominan columnas de agrupación de la consulta, ya que ellas son las que determinan cómo se dividen las filas en grupo.

Múltiples columnas de agrupación.

SQL puede agrupar resultados de consultas en base a contenidos de dos o más columnas. Por ejemplo, calcular el total facturado por cada vendedor y a cada cliente el año pasado ordenado por vendedor primero y luego por cliente:

```

select v.cod_vendedor,ape_vendedor+' '+nom_vendedor'Vendedor', ape_cliente+' '+nom_cliente'Cliente',sum(pre_unitario*cantidad)'Total'
from detalle_facturas d, facturas f,vendedores v, clientes c
where d.nro_factura=f.nro_factura and f.cod_cliente=c.cod_cliente
and f.cod_vendedor=v.cod_vendedor and year(fecha)=year(getdate())-1
group by v.cod_vendedor,ape_vendedor+' '+nom_vendedor,c.cod_cliente,ape_cliente+' '+nom_cliente
order by 2,3

```

	cod_vendedor	Vendedor	Cliente	Total
1	1	Camizo Martín	Abarca Héctor	817.00
2	1	Camizo Martín	Castillo Marta Analía	558.00
3	1	Camizo Martín	Luque Elvira Josefa	150.00
4	1	Camizo Martín	Morales Santiago	3390.00
5	1	Camizo Martín	Paez Roque	200.00
6	1	Camizo Martín	Perez Carlos Antonio	3406.00
7	1	Camizo Martín	Perez Rodolfo	551.50
8	1	Camizo Martín	Ruiz Marcos	800.00
9	2	Ledesma Mariela	Abarca Héctor	996.00
10	2	Ledesma Mariela	Castillo Marta Analía	120.00
11	2	Ledesma Mariela	Luque Elvira Josefa	4759.00
12	2	Ledesma Mariela	Morales Pilar	4665.00
13	2	Ledesma Mariela	Morales Santiago	2692.50
14	2	Ledesma Mariela	Paez Roque	2690.00
15	2	Ledesma Mariela	Perez Carlos Antonio	299.00
16	2	Ledesma Mariela	Perez Rodolfo	1672.00
17	2	Ledesma Mariela	Ruiz Marcos	2574.50
18	3	Lopez Alejandro	Abarca Héctor	1174.00

Esta consulta respondería a la pregunta: ¿Cuánto le vendió cada vendedor a cada cliente el año pasado?

Observe detenidamente, qué es lo que se incluye como columnas de agrupación (en el group by) y qué columnas son las que se utilizaron en la lista de selección.

Restricciones en consultas agrupadas.

Las consultas agrupadas están sujetas a algunas limitaciones bastante estrictas:

- Las columnas de agrupación deben ser columnas efectivas de las tablas designadas en la cláusula from de la consulta.
- No se pueden agrupar las filas basándose en el valor de una expresión calculada.
- Todos los elementos de la lista de selección deben tener un único valor por cada grupo de filas, es decir, pueden ser: una constante, una función de columna o de agregado (que producen un único valor sumando las filas del grupo), una columna de agrupación (que tienen el mismo valor) o una expresión que afecte a combinaciones de los anteriores.

Cláusula Having

Se utiliza para agregar condiciones de búsqueda para grupos es decir para las filas que resultan de la agrupación y cálculo de resultados de las funciones de agregado. Estas condiciones son las mismas explicadas para la cláusula Where.

```

Select nro_factura, sum(pre_unitario*cantidad) Total
from detalle_facturas
group by nro_factura
having sum(pre_unitario*cantidad)>2500

```

	nro_factura	Total
1	45	2565.00
2	167	2630.00
3	223	3012.50
4	287	2630.00
5	311	2630.00
6	403	3470.90
7	406	3058.00
8	414	2947.70
9	420	4170.00
10	422	3040.00
11	461	2510.00
12	465	3154.50
13	466	2662.50
14	470	2966.00
15	475	2878.50
16	513	2857.30
17	519	2533.50
18	522	3040.00
19	532	3040.00
20	537	2734.50

Las restricciones en condiciones de búsqueda de grupos.

La cláusula Having se utiliza para incluir o excluir grupos de filas de los resultados de la consulta, por lo que la condición de búsqueda que especifica debe ser aplicable al grupo en su totalidad en lugar de a filas individuales. Esto significa que un

elemento que aparezca dentro de la condición de búsqueda en el Having pueden ser las mismas que las enumeradas en el group by.

Se sabe que: la cláusula Where se aplica a filas individuales, por lo que las expresiones que contiene deben ser calculables para filas individuales y la cláusula Having se aplica a grupos de filas, por lo que las expresiones que contengan deben ser calculables para un grupo de filas.

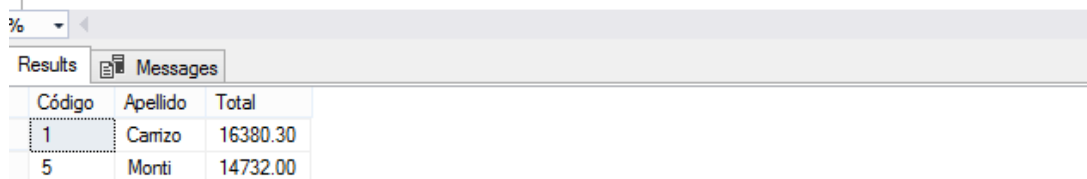
Por lo que la condición de búsqueda sobre la fecha de la factura como no está incluida en la agrupación del group by ni tampoco en una función de agregado deberá ir en el where como también las condiciones para realizar la composición de tablas. La condición referida al importe total de facturación como es una función de agregado no podrá ir en el where sino en el having.

Entonces la consulta quedaría:

```

Select f.cod_vendedor Código, ape_vendedor Apellido, sum(pre_unitario*cantidad) Total
from detalle_facturas d, facturas f,vendedores v
where d.nro_factura = f.nro_factura and f.cod_vendedor=v.cod_vendedor
and year(fecha) = 2017
group by f.cod_vendedor, ape_vendedor
having sum(pre_unitario*cantidad)<17000

```



Código	Apellido	Total
1	Camizo	16380.30
5	Monti	14732.00

Otras formas de composición: Inner, Left y Right Join

Un join es una operación que relaciona dos o más tablas para obtener un resultado que incluya datos (campos y registros) de ambas; las tablas participantes se combinan según los campos comunes a ambas tablas.

Hay tres tipos de combinaciones:

combinaciones internas (inner join o join),

combinaciones externas y

combinaciones cruzadas.

La combinación interna emplea "join", que es la forma abreviada de "inner join". Se emplea para obtener información de dos tablas y combinar dicha información en una salida. La sintaxis básica es la siguiente:

SELECT campos

FROM tabla1

JOIN tabla2

ON condicion de combinacion;

Por ejemplo si se quieren listar los datos de los clientes con sus facturas: (cada factura con su respectivo cliente)

```

select ape_cliente +' '+ nom_cliente CLIENTE, nro_factura FACTURA, fecha FECHA
from facturas f join clientes c on f.cod_cliente=c.cod_cliente

```

En la cláusula From se especifica el nombre de la primera tabla (facturas), luego combinamos join con la segunda tabla (clientes) y luego de on especificamos a través de qué campos se combinan.

Si se quiere realizar una consulta con más de dos tablas, por ejemplo listar los datos de los clientes, con sus facturas y vendedores, sería:

```
select ape_cliente+' '+nom_cliente CLIENTE,nro_factura FACTURA,FORMAT(fecha,'dd/mm/yyyy') FECHA,ape_vendedor+' '+nom_vendedor VENDEDOR
from facturas f join clientes c on f.cod_cliente=c.cod_cliente
join vendedores v on f.cod_vendedor=v.cod_vendedor
```

	CLIENTE	FACTURA	FECHA	VENDEDOR
510	Morales Santiago	510	30/00/2018	Miranda Marcelo
511	Castillo Marta Analía	511	01/00/2018	Miranda Marcelo
512	Morales Santiago	512	01/00/2018	Monti Gabriel
513	Morales Pilar	513	02/00/2018	Monti Gabriel
514	Morales Santiago	514	10/00/2018	Lopez Alejandro
515	Morales Santiago	515	11/00/2018	Miranda Marcelo
516	Perez Rodolfo	516	12/00/2018	Monti Gabriel
517	Castillo Marta Analía	517	14/00/2018	Miranda Marcelo

Usar "join" o "inner join" da el mismo resultado que realizar la composición como una condición de búsqueda en el where, solo se mostrarán los resultados de aquellos registros donde coincidan los valores de sus claves primarias y foráneas respectivas es decir no se mostrarán los registros donde el cliente no tenga facturas o el vendedor no tenga registradas facturas o facturas sin clientes ni facturas sin vendedores.

Combinación externa izquierda: left join

Una combinación interna (join) encuentra registros de la primera tabla que se correspondan con los registros de la segunda, y si un valor de la primera tabla no se encuentra en la segunda tabla, el registro no aparece.

Se emplea una combinación externa izquierda para mostrar todos los registros de la tabla de la izquierda. Si no encuentra coincidencia con la tabla de la derecha, el registro muestra los campos de la segunda tabla seteados a "null". En el siguiente ejemplo solicitamos la descripción del artículo y cantidad vendida en cada factura:

El resultado mostrará las descripciones de los artículos y las cantidades vendidas; los artículos que no han sido vendidos aparecen en el resultado, pero con el valor "null" en los campos "cant" y "nro_factu".

Es importante la posición en que se colocan las tablas en un "left join", la tabla de la izquierda es la que se usa para localizar registros en la tabla de la derecha.

Entonces, un "left join" se usa para hacer coincidir registros en una tabla (izquierda) con otra tabla (derecha); si un valor de la tabla de la izquierda no encuentra coincidencia en la tabla de la derecha, se genera una fila extra (una por cada valor no encontrado) con todos los campos correspondientes a la tabla derecha seteados a "null".

```
select a.cod_articulo,descripcion,nro_factura,cantidad
from articulos a left join detalle_facturas d on a.cod_articulo = d.cod_articulo
```

	cod_articulo	descripcion	nro_factura	cantidad
939	23	Goma para lápiz * 10 u	267	20
940	23	Goma para lápiz * 10 u	276	15
941	24	Goma para lapicera * 10 u	48	25
942	25	Lápices Color largos * 12 FABER	59	2
943	25	Lápices Color largos * 12 FABER	138	7
944	25	Lápices Color largos * 12 FABER	247	1
945	25	Lápices Color largos * 12 FABER	308	1
946	25	Lápices Color largos * 12 FABER	188	1
947	25	Lápices Color largos * 12 FABER	380	1
948	26	Lapicera Bic Azul trazo fino	NULL	NULL
949	27	Cuaderno tapa dura rayado	NULL	NULL
950	28	Lápices Color largos x 12u.	NULL	NULL
951	29	Conjunto geométrico Maped	NULL	NULL

Query executed successfully.

Combinación externa derecha: right join

Vimos que una combinación externa izquierda (left join) encuentra registros de la tabla izquierda que se correspondan con los registros de la tabla derecha y si un valor de la tabla izquierda no se encuentra en la tabla derecha, el registro muestra los campos correspondientes a la tabla de la derecha con "null".

Una combinación externa derecha ("right outer join" o "right join") opera del mismo modo sólo que la tabla derecha es la que localiza los registros en la tabla izquierda. En el siguiente ejemplo solicitamos el nombre del cliente (aunque no haya facturas de ese cliente) y las facturas de dicho cliente:


```
select ape_cliente+' '+nom_cliente CLIENTE,nro_factura FACTURA,FORMAT(fecha,'dd/mm/yyyy') FECHA
from facturas f right join clientes c on f.cod_cliente=c.cod_cliente
```

	CLIENTE	FACTURA	FECHA
530	Perez Rodolfo	530	14/00/2018
531	Ruiz Marcos	531	24/00/2018
532	Ruiz Marcos	532	02/00/2018
533	Ruiz Marcos	533	02/00/2018
534	Ruiz Marcos	534	03/00/2018
535	Ruiz Marcos	535	13/00/2018
536	Ruiz Marcos	536	04/00/2018
537	Ruiz Marcos	537	04/00/2018
538	Ruiz Marcos	538	05/00/2018
539	Ruiz Marcos	539	06/00/2018
540	Ruiz Marcos	540	07/00/2018
541	Gonzalez Adriana	NULL	NULL
542	Perez Ana María	NULL	NULL

Es FUNDAMENTAL tener en cuenta la posición en que se colocan las tablas en los "outer join". En un "left join" la primera tabla (izquierda) es la que busca coincidencias en la segunda tabla (derecha); en el "right join" la segunda tabla (derecha) es la que busca coincidencias en la primera tabla (izquierda).

Combinación externa completa: full join

Una combinación externa completa ("full outer join" o "full join") retorna todos los registros de ambas tablas. Si un registro de una tabla izquierda no encuentra coincidencia en la tabla derecha, las columnas correspondientes a campos de la tabla derecha aparecen seteadas a "null", y si la tabla de la derecha no encuentra correspondencia en la tabla izquierda, los campos de esta última aparecen conteniendo "null". Veamos un ejemplo:

```
select ape_cliente+' '+nom_cliente CLIENTE,nro_factura FACTURA
from facturas f full join clientes c on f.cod_cliente=c.cod_cliente
```

La salida del "full join" precedente muestra todos los registros de ambas tablas, incluyendo los clientes que no tengan facturas y las facturas que no tengan clientes.

Combinaciones cruzadas: cross join

Las combinaciones cruzadas (cross join) muestran todas las combinaciones de todos los registros de las tablas combinadas. Para este tipo de join no se incluye una condición de enlace. Se genera el producto cartesiano en el que el número de filas del resultado es igual al número de registros de la primera tabla multiplicado por el número de registros de la segunda tabla, es decir, si hay 5 registros en una tabla y 6 en la otra, retorna 30 filas.

Combinar varios tipos de join en una misma sentencia.

Es posible realizar varias combinaciones para obtener información de varias tablas. Las tablas deben tener claves externas relacionadas con las tablas a combinar. En consultas en las cuales empleamos varios "join" es importante tener en cuenta el orden de las tablas y los tipos de "join"; recuerde que la tabla resultado del primer join es la que se combina con el segundo join, no la segunda tabla nombrada.

Bibliografía

Groff, J. & Weinberg, P. (1991). Aplique SQL. Madrid. Editorial Mc Graw Hill

Microsoft (2019) Queries. Disponible en: <https://docs.microsoft.com/es-es/sql/t-sql/queries/queries?view=sql-server-2017>

Opel, A. & Sheldon, R. (2010). Fundamentos de SQL. Madrid. Editorial Mc Graw Hill

Tutoriales Ya. (2015) Sql Server Ya. Recuperado de: www.sqlserverya.com.ar