

TECNICATURA UNIVERSITARIA EN PROGRAMACIÓN

LABORATORIO DE COMPUTACIÓN II

Unidad Temática 3: Subconsultas

TEÓRICO

Curso: 1 er Año. 2 ndo Cuatrimestre

Índice

| | |
|--|---|
| Subconsultas | 2 |
| Subconsultas en la cláusula WHERE | 2 |
| Subconsultas en la cláusula HAVING | 4 |
| Otras Subconsultas | 4 |

Subconsultas

Una *subconsulta* es una consulta que puede aparecer dentro de la cláusula WHERE o HAVING de otra sentencia SQL. La subconsulta debe ir encerrada entre paréntesis, y tiene el formato de una sentencia SELECT, pero existen diferencias entre una subconsulta y una sentencia SELECT real. Una subconsulta:

- debe producir una **única** columna de datos como resultados. Esto significa que una subconsulta siempre tiene un único elemento de selección en su cláusula SELECT;
- no puede ser UNION de varias sentencias SELECT diferentes.
- La cláusula ORDER BY no puede ser especificada en una subconsulta.
- Los nombres de columna que aparecen en una subconsulta pueden referirse a columnas de tablas en la consulta principal, esto se denomina **referencias externas**.

Subconsultas en la cláusula WHERE

Las subconsultas suelen ser utilizadas en la cláusula WHERE de una sentencia SQL, en este caso funciona como parte del proceso de selección de filas.

Las condiciones de búsqueda en subconsultas son las que se describen:

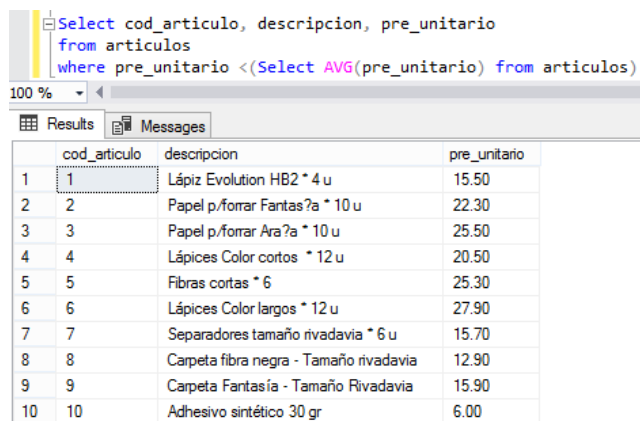
Test de comparación subconsulta: =, <, >, <=, >=. Es una forma modificada del test de comparación simple. Compara el valor de una expresión con el valor único producido por una subconsulta, y devuelve un resultado TRUE si la comparación es cierta.

Este es por ejemplo el caso del punto 1 del Problema N° 9 donde se necesitaba el listado de artículos cuyos precios son menores al promedio:

```

Select cod_articulo, descripcion, pre_unitario
from articulos
where pre_unitario < (Select AVG(pre_unitario) from articulos)

```



| cod_articulo | descripcion | pre_unitario |
|--------------|--|--------------|
| 1 | Lápiz Evolution HB2 * 4 u | 15.50 |
| 2 | Papel p/fornar Fantasia * 10 u | 22.30 |
| 3 | Papel p/fornar Ara?a * 10 u | 25.50 |
| 4 | Lápices Color cortos * 12 u | 20.50 |
| 5 | Fibras cortas * 6 | 25.30 |
| 6 | Lápices Color largos * 12 u | 27.90 |
| 7 | Separadores tamaño rivadavia * 6 u | 15.70 |
| 8 | Carpeta fibra negra - Tamaño rivadavia | 12.90 |
| 9 | Carpeta Fantasia - Tamaño Rivadavia | 15.90 |
| 10 | Adhesivo sintético 30 gr | 6.00 |

La subconsulta especificada en este test debe producir una *única fila* de resultados.

Test de pertenencia al conjunto: IN. Es una forma modificada del test de pertenencia a conjunto simple; compara un único valor de datos con una columna de valores producida por una subconsulta y devuelve un resultado TRUE si el valor coincide con uno de los valores de la columna.

Listar los datos de los clientes que compraron este año:

```

select * from clientes
where cod_cliente in (select cod_cliente
                      from facturas
                      where year(fecha) = year(getdate()))

```

Donde la consulta principal listaría todos los datos de los clientes cuyo cod_cliente esté dentro del conjunto de cod_cliente que surjan de la subconsulta. La subconsulta generará un listado de cod_cliente desde la tabla facturas cuyo año de la fecha sea

igual al año de la fecha actual. La subconsulta no muestra nada solo es utilizada para verificar la condición de búsqueda del where.

Se quieren listar los clientes que no vinieron el año pasado

```
select cod_cliente Codigo, ape_cliente+' '+nom_cliente Nombre
from clientes
where cod_cliente not in (select cod_cliente
                        from facturas
                        where year(fecha) = year(getdate()-1)
                        )
```

| | Codigo | Nombre |
|---|--------|------------------|
| 1 | 11 | Perez Ana María |
| 2 | 12 | Gonzalez Adriana |

En este caso la subconsulta listaría todos los códigos de clientes que compraron el año pasado. En el where de la consulta principal se va a analizar fila por fila si el código de cliente que se quiere listar no pertenece al conjunto de valores devueltos por la subconsulta.

Test de existencia: EXISTS. Comprueba si una subconsulta produce alguna fila de resultados; no hay test de comparación simple que se asemeje al test de existencia, solamente se utiliza con subconsultas.

El ejemplo anterior (lista los datos de los clientes que compraron este año) con este test sería:

```
select cod_cliente Codigo, ape_cliente+' '+nom_cliente Nombre
from clientes c
where exists (select cod_cliente
             from facturas f
             where c.cod_cliente = f.cod_cliente
             and year(fecha) = year(getdate())
             )
```

En este caso las subconsulta devolvería filas de resultado si el cliente que se quiere listar en la consulta principal (recuerde que se realiza el análisis fila a fila) posee algún registro en la tabla facturas correspondiente a la fecha con año igual al año de la fecha actual. Observe el **c.cod_cliente** de la subconsulta es una *referencia externa* dado que se trata de un campo utilizado en la subconsulta y que pertenece a la tabla de la consulta principal

Para listar los datos de los clientes que no compraron el año pasado utilizando este test sería:

```
select cod_cliente Codigo, ape_cliente+' '+nom_cliente Nombre
from clientes c
where not exists (select cod_cliente
                 from facturas f
                 where c.cod_cliente = f.cod_cliente --referencia externa
                 and year(fecha) = year(getdate()-1)
                 )
```

En este otro ejemplo, la subconsulta devolvería filas de resultado si el cliente que se quiere listar en la consulta principal posee algún registro en la tabla facturas correspondiente a la fecha con año igual al año anterior a la fecha actual. Si no existen filas de resultados el test NOT EXISTS devuelve verdadero y el registro se muestra en la tabla final.

Test cuantificados: ANY; ALL. Estos test extienden el de pertenencia al conjunto IN, a otros operadores de comparación como =, <, <=, >, >= y <>.

El test ANY: se utiliza con uno de estos seis operadores de comparación. Para efectuar el test, SQL utiliza el operador de comparación especificado para comparar el valor del test con *cada* valor de datos en la columna, uno cada vez. Si *alguna* de las comparaciones individuales producen un resultado TRUE, el test ANY devuelve un resultado TRUE.

Por ejemplo si se quiere listar los clientes que alguna vez compraron un producto menor a \$ 10.-

```
select cod_cliente Codigo, ape_cliente+' '+nom_cliente Nombre
from clientes c
where 10 > any (select pre_unitario
                from facturas f join detalle_facturas d on f.nro_factura=d.nro_factura
                where c.cod_cliente=f.cod_cliente --referencia externa
               )
```

| | Codigo | Nombre |
|---|--------|-----------------------|
| 1 | 1 | Perez Rodolfo |
| 2 | 2 | Castillo Marta Analía |
| 3 | 3 | Abarca Héctor |
| 4 | 4 | Morales Santiago |
| 5 | 5 | Perez Carlos Antonio |
| 6 | 6 | Morales Pilar |
| 7 | 7 | Paez Roque |

La subconsulta de este ejemplo lista todos los precios unitarios de los artículos vendidos al cliente que la consulta principal quiere listar; si alguno de éstos precios es menor a 10 entonces el registro de ese cliente forma parte de las filas de resultado.

El test ALL se utiliza también con uno de los seis operadores de comparación. Para efectuar el test, SQL utiliza el operador de comparación especificado para comparar el valor de test con todos y *cada* uno de los valores de datos de la columna. Si **todas** las comparaciones individuales producen un resultado TRUE, el test ALL devuelve un resultado TRUE.

Como el punto 8: se quiere listar los clientes que siempre fueron atendidos por el vendedor 3:

```
select cod_cliente Codigo, ape_cliente+' '+nom_cliente Nombre
from clientes c
where 3 = all (select cod_vendedor
               from facturas f
               where c.cod_cliente=f.cod_cliente --referencia externa
              )
```

| | Codigo | Nombre |
|---|--------|------------------|
| 1 | 10 | Moriel Roberto |
| 2 | 11 | Perez Ana María |
| 3 | 12 | Gonzalez Adriana |

En este caso, la subconsulta emite un listado de códigos de vendedores cuyas ventas hayan sido realizadas al cliente que se quiere listar en la consulta principal (referencia externa). El where verificará si todos los códigos de vendedores listados en la subconsulta son iguales a 3.

Subconsultas en la cláusula HAVING

También pueden utilizarse subconsultas en la cláusula HAVING y funciona como parte de la selección de un grupo de filas efectuada por ésta cláusula.

En el caso del problema deberíamos listar los vendedores con sus ventas totales agrupado por vendedor y como condición es que su venta total (en el having) sea superior al resultado de una subconsulta que calcule el promedio general de ventas.

```
select v.cod_vendedor CODIGO,ape_vendedor+' '+nom_vendedor VENDEDOR, sum(cantidad*pre_unitario) IMPORTE
from facturas f join detalle_facturas d on f.nro_factura = d.nro_factura
join vendedores v on f.cod_vendedor = v.cod_vendedor
group by v.cod_vendedor, ape_vendedor+' '+nom_vendedor
having avg(cantidad*pre_unitario) < (select avg(cantidad*pre_unitario)
                                   from detalle_facturas
                                   )
```

| | CODIGO | VENDEDOR | IMPORTE |
|---|--------|-----------------|----------|
| 1 | 3 | Lopez Alejandro | 76706.10 |
| 2 | 1 | Camizo Martin | 65345.75 |
| 3 | 2 | Ledesma Mariela | 76622.90 |

Otras Subconsultas

En apartados anteriores se explicó el tema de subconsultas en el where y en el having, pero también pueden:

- haber subconsultas dentro de subconsultas, se admiten hasta 32 niveles de anidación.
- en lugar de una expresión, siempre que devuelvan un solo valor o una lista de valores.
- como columna en la lista de selección
- que retornen un conjunto de registros de varios campos en lugar de una tabla (en el from) o para obtener el mismo resultado que una combinación (join).

Como ejemplo de una subconsulta como parte de una expresión, se quiere listar el precio de los artículos y la diferencia de éste con el precio del artículo más caro:

```
select descripcion, precio_unit,
       (select max(pre_unitario)from articulos)- pre_unitario 'diferencia'
from articulos
```

Otro caso sería cuando se usa una subconsulta en una columna por ejemplo: listar el precio actual de los artículos y el precio histórico vendido más barato:

```
select descripcion, pre_unitario 'precio actual',
       (select min(pre_unitario) from detalle_facturas d
        where d.cod_articulo=a.cod_articulo) 'precio historico'
from articulos a
```

Observe que en ambos ejemplos las subconsultas devuelven un solo registro en caso contrario daría error.

Por último, se quiere emitir un listado de las facturas del año en curso detallando número de factura, cliente, fecha y total de la misma

```
select nro_factura, fecha, apellido+' '+nombre 'Cliente',
       (select sum(d.precio_unit*cantidad)
        from detalle_facturas d
        where f.nro_factura=d.nro_factura) 'total'
from facturas f join clientes c on c.cod_cliente=f.cod_cliente
where year(getdate())= year(fecha)
```

Se pueden emplear subconsultas que retornen un conjunto de registros de varios campos en lugar de una tabla. Se la denomina tabla derivada y se coloca en la cláusula "from" para que la use un "select" externo. La tabla derivada debe ir entre paréntesis y tener un alias para poder referenciarla.

Supongamos el ejemplo anterior, donde se quiere emitir un listado de las facturas del año en curso detallando número de factura, cliente, fecha y total de la misma se podría resolver de la siguiente manera:

```
select f.nro_factura, fecha, apellido+' '+nombre 'Cliente', f2.total
from facturas f join clientes c on c.cod_cliente=f.cod_cliente
join (select nro_factura, sum(precio_unit*cantidad) 'total' from detalle_facturas
group by nro_factura) as f2 on f2.nro_factura=f.nro_factura
where year(fecha)= year(getdate())
```

Subconsultas en el UPDATE y DELETE

En la cláusula Where de las sentencias Update y Delete pueden ir todas las condiciones de búsquedas que ya hemos visto en el Where de la sentencia Select, incluso subconsultas.

Por ejemplo si se quiere aumentar un 5% los precios de los artículos cuyos precios son inferiores al promedio general.

```
Update articulos
Set pre_unitario= pre_unitario*1.05
where pre_unitario<(select avg(pre_unitario)
                    from articulos)
```

Otro ejemplo, se quiere eliminar los clientes que no vinieron nunca.

```
delete clientes
where cod_client not in(select cod_cliente
                        from facturas)
```

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