Apuntes accesibles de subconsultas

Sitio: FRANCISCO DE GOYA
Curso: Bases de datos ASIR1, DAM1 y DAW1
Libro: Apuntes accesibles de subconsultas

Imprimido por: Alberto Bollo Rodríguez
Día: martes, 6 de febrero de 2024, 11:48

Tabla de contenidos

1. Subconsultas

Las subconsultas en SQL son consultas que internamente usan otras consultas. Habrá una jerarquía entre ellas de forma que una consulta exterior usará una consulta que alojará en una de sus cláusulas:

- From: cuando se usa el resultado de una consulta a modo de tabla. Esto forma una tabla temporal que necesitará un alias.
- Where: cuando se usa como una condición. En este caso hay dos posibilidades:
  - Comparación con un solo valor. La consulta ha de devolver un único valor siempre (una fila con una sola columna), con los comparadores =, !=, <, >, >=, <=, like, etc.
  - Comparación con una lista. Si la consulta puede devolver ninguna, una o más filas con una sola columna. Se puede usar con el comparador IN.
- Con operadores. De la misma forma que se puede comparar con una selección, en los campos seleccionados tras la cláusula SELECT se puede incluir una operación, por ejemplo una resta, entre un campo y el resultado de un select. En este caso, la subconsulta, también debería devolver un único valor.
- Modificando valores: devuelven un valor, ya sea numérico, de texto u otro tipo, que sirve para operar con otros valores fijos o de una columna. Los operadores pueden ser +, -, *, /, %, like, not like, etc.

En todo caso, las subconsultas son una herramienta para realizar selecciones más complejas que en ocasiones son imposibles con una sola consulta. De hecho pueden usarse una o varias subconsultas, con niveles jerárquicos cualesquiera. Por ejemplo un select con una subconsulta que dentro tiene otra subconsulta, o una subconsulta en el from y otras dos en el where.

También pueden afectar a tablas diferentes, esto es, que la consulta principal o externa selecciona sobre una tabla y una o todas las subconsultas usan otras tablas. En este caso no se consideran consultas multitabla dado que cada select opera sobre una única tabla en la cláusula FROM.

1.1. From. Tablas temporales.

Hay algunas consultas que es necesario realizarlas paso a paso, obteniendo primero unos datos que, por ejemplo si están agrupados, impide realizar otras operaciones, o cualquier otra operación que lo dificulte. Otras veces por simple lógica.

En todo caso el modo de proceder debería ser realizar **primero las subconsultas más internas** y posteriormente ir reailzando las consultas que usan a las primeras.

Toda subconsulta de este tipo debe llevar un **alias** como si del nombre de la tabla se tratara.

Realicemos dos ejemplos, uno en la base de datos bebés, y otro en miscelánea:

Ejemplo 1

Buscamos el número de veces que cada letra es inicial de un nacimiento:

```
select count(*),left(nombre,1) from nacimientos group by left(nombre,1);
```

count(*)	left(nombre,1)
4	A
1	B
...	...
1	S

Como podemos ver, hay 11 filas, con 11 letras y cada una tiene el número de nacimientos cuya inicial del nombre coincide.

Es recomendable que se renombren los campos de la consulta de la siguiente forma:

```
select count(*) as num,left(nombre,1) as inicial from nacimientos group by left(nombre,1);
```

Si se ejecuta, se verá que no hay ningún cambio por asignar alias a los campos. Esto será muy útil para acceder a estos desde una consulta superior, esto es, que utilice este resultado como si fuese una tabla.

num	inicial
4	A
1	B
...	...
1	S

Por ello ahora vamos a buscar las iniciales cuyo número sea mayor que 2. Si consideramos que el resultado anterior es una tabla, se accedería de la misma forma, usando los campos:

```
select num,inicial from temporal1 where num > 2;
```

Pero la tabla **temporal1** no existe. Entonces lo que debemos hacer es:

1. Coger toda la consulta que genera el resultado, rodearla de paréntesis y añadir un alias después del paréntesis:
   ```
   ( select count(*) as num,left(nombre,1) as inicial from nacimientos group by left(nombre,1)) as temporal1
   ```
2. Sustituir **temporal1** por lo que hemos obtenido, en la consulta que queríamos hacer y no dejaba porque temporal1 no existía.
   ```
   select num,inicial from ( select count(*) as num,left(nombre,1) as inicial from nacimientos group by left(nombre,1) ) as temporal1 where num > 2;
   ```

El resultado final que nos aporta la consulta con subconsulta es:

count(*)	left(nombre,1)
4	A
3	M
6	R

De esta forma hemos creado una consulta exterior (select num, inicial from ... where num > 2) que usa en el from un resultado de un select, en vez de una tabla. Si se es observador se verá que este ejemplo es equivalente a usar un having en una sola consulta, pero es muy ilustrativo sobre cómo hacer una subconsulta sencilla y adicionalmente, sobre cómo se puede obtener el mismo resultado por dos medios diferentes.

Ejemplo 2

Queremos saber el precio total máximo que ha pagado un cliente por una sola factura, en la tabla facturas de miscelánea. Para ello será necesario primero operar precio_producto por unidades.

```
select num,cliente, unidades * precio_unidad as total_prod from facturas;
```

num	cliente	total_prod
12	Compraqui	3.6
12	Compraqui	1.8
...	...	...
17	Super Paquito	1.36

En total 114 filas. No necesitamos el nombre del producto, ni la fecha. Mantenemos el nombre del cliente porque es lo que querremos mostrar al final y num porque identifica cada factura.

El siguiente paso sería sumarlo todo.

```
select num,cliente,sum(unidades*precio_unidad) as suma from facturas group by num,cliente
```

num	cliente	total_prod
1	Super Paquito	202.22
2	Super Paquito	390.10
...	...	...
17	Super Paquito	288.32

Total 17 filas. En este caso no ha hecho falta realizar una subconsulta dado que operamos y agrupamos. Pero tras esto ya no podríamos operar mucho más. Necesitamos usar el resultado como si fuese una tabla y realizar otra función de agrupación.

```
select cliente,max(suma) as mx from (select num,cliente,sum(unidades*precio_unidad) as suma from facturas group by num,cliente) as sel1  group by cliente
```

cliente	suma
Ahorra Muchísimo	363.74
Baratolandia	1880.80
...	...
Super Paquito	390.1

Lo que se ha hecho es sobre la tabla anterior calcular el máximo valor de la suma, pero agrupado por cliente. De esta forma obtenemos la mayor factura que ha pagado cada cliente.

De hecho posteriormente se puede realizar una consulta aún más compleja inicialmente, pero muy fácil de realizar teniendo ya hechas las anteriores: sumar todos los totales de la factura más altas de cada cliente. Esto no se puede hacer con un simple max(sum(suma)) ... sino que hay que usar otra consulta que envuelva la anterior.

```
select sum(mx) as suma_maximos from (select cliente,max(suma) as mx from (select num,cliente,sum(unidades*precio_unidad) as suma from facturas group by num,cliente) as sel1  group by cliente) as sel2;
```

suma_maximos
4503.15

1.2. Subconsultas con operadores y un resultado.

Hay algunas consultas que requieren **un valor, y solo un valor,** que se calcula en otra consulta y se usa como condición. Por ello, hay que calcular primero ese valor y luego utilizarlo en otra consulta. Y la consulta que devuelve el valor **ha de devolver una fila y una columna obligatoriamente**. No sirven varias columnas ni varias filas.

Por otro lado, los operadores de comparación de la condición han de ser aquellos que comparan solo dos valores, los operadores lógicos: >, <, >=, <=, =, !=, like, not like...

Vamos a realizar el siguiente ejemplo:

Ejemplo

Usaremos la base de datos miscelánea y la tabla facturas. En este caso queremos saber qué productos se han comprado que cuesten más del doble que la media de los productos que aparecen en la tabla.

Lo primero sería averiguar cuál es el doble de la media:

```
select avg(precio_unidad)*2 as media from facturas
```

media
3.553158

Para acabar pronto, podríamos usar directamente el número (mal) en otra consulta:

```
select producto, precio_unidad from facturas where precio_unidad > 3.553158
```

producto	precio_unidad
Queso fresco	6.02
Flan	4.5
Queso fresco	6.02
Flan	4.5
...	...
Queso fresco	6.02

El resultado parece correcto, y temporalmente lo es. Pero en cuanto se hiciesen inserts, updates o deletes ya no lo sería. Siempre se necesitan querys que obtengan resultados siempre correctos, y no en un momento concreto. Por ello la solución es muy sencilla: cambiar ese 3.553158 por la primera consulta.

Por ello, una vez tenemos el resultado, rodearemos la consulta que lo produce con un paréntesis. A diferencia de las subconsultas del apartado anterior, en este caso no es necesario un alias.

```
select producto, precio_unidad from facturas where precio_unidad > (select avg(precio_unidad)*2 as media from facturas)
```

Esta consulta siempre será correcta, pase lo que pase con los datos de la tabla. Y es que una vez obtenida la media, usar el resultado escrito o copiar la consulta, tienen la misma complejidad. Por ello siempre será correcto e igual de fácil hacerlo correctamente con una subconsulta.

1.3. Subconsultas con operadores y varios resultados.

Otras subconsultas obtienen varias filas de una columna en un solo select. La consulta superior usará esa lista, pero para ello solo es posible usar ciertos operadores: **like** y **not like**.

La subconsulta, en esta ocasión, puede devolver ninguna fila, una o varias. Pero sigue siendo obligatorio que sea **una columna**. También es importante resaltar que no es necesario poner un alias a la subconsulta.

Operador IN

Queremos saber los nombres de los autores de los libros cuyas editoriales tengan entre 3 y 4 libros. En esta tabla se sabe simplemente porque hay varios libros que coinciden en la editorial.

Lo primero que habrá que averiguar es, ¿cuántos libros tiene cada editorial?

```
select count(*), editorial from libros group by editorial
```

count(*)	editorial
3	COCINA LEIDA
5	LIBRO AMIGO
4	PLANETA AGUSTIN

Y así, lo siguiente será descartar aquellas que no estén entre 3 y 4, usando un having, y eliminar la cuenta, ya que una vez usada en el filtro, no será necesaria esa información:

```
select editorial from libros group by editorial having count(*) between 3 and 4;
```

editorial
COCINA LEIDA
PLANETA AGUSTIN

Esta consulta será la lista que necesitaremos para la consulta superior. La consulta superior en sí es muy sencilla y de forma manual quedaría así:

```
select nom_autor from libros where editorial in ("COCINA LEIDA", "PLANETA AGUSTIN")
```

Pero al igual que en apartados anteriores, esta consulta solo es válida en el instante que se realiza la subconsulta. Si posteriormente los datos cambian mediante update, insert o delete, ya no sería válida. Por ello sustituiremos la lista entre paréntesis por la consulta que la genera:

```
select nom_autor from libros where editorial in (select editorial from libros group by editorial having count(*) between 3 and 4);
```

nom_autor
CARMEN
RAUL
ROSA
FERNANDO
ROSALTA
ALBERTO
SANDRA

Operadores ANY / SOME y ALL

A diferencia del operador IN, ANY y SOME necesitan un operador posterior, ya sea >, =, <, etc. Valga el siguiente ejemplo, donde seleccionamos nombre y especie de las mascotas que no sean perros, pero hayan nacido un mismo año que cualquier perro:

```
select nombre, especie from mascotas where year(nacimiento) = ANY (select year(nacimiento) from mascotas where especie = "Perro") and especie not like "Perro";
```

1.4. Subconsultas operando en los campos y condiciones

Al igual que es posible operar con una subconsulta en una condición del where, mediante operadores lógicos, también lo es operando sobre campos mediante otros operadores:

- Operadores aritméticos en campos del select: podríamos cambiar un valor numérico, por ejemplo sumando el resultado de una subconsulta:

```
select a + (select sum(b) from t2) from t1
```

- Operadores aritméticos en campos del where: a diferencia de las subconsultas del epígrafe 1.2, en este caso cambian el valor a comparar:

```
select a from t1 where b = c + (select max(x) from t2)
```

- Operadores de cadena en el where:

```
select a from t1 where b = like concat("%",(select x from t2 where id = 6),"%")
```

Partiendo de aquí, puede haber tantos tipos de subconsultas como se pueda imaginar. Véanse algunos ejemplos:

Ejemplo 1

Se usará la base de datos miscelánea. Se quiere saber el nombre de los autores que tienen la misma inicial en su nombre que la inicial del autor cuyas páginas del libro son 446. Notar que es un ejemplo, pues si hubiese dos libros con el mismo número de páginas devolvería dos filas, por lo tanto la consulta superior daría un error.

Se comienza averiguando la inicial del autor que ha escrito un libro con 446 páginas.

```
select left(nom_autor,1) from libros where num_paginas = 446
```

left(nom_autor,1)
R

Posteriormente podríamos incluir directamente en la consulta superior la subconsulta con el operador like:

```
select nom_autor from libros where nom_autor like (select left(nom_autor,1) from libros where num_paginas = 446)
```

En cuanto a la construcción de una subconsulta es correcto. Pero esto no funciona, dado que necesitamos colocar un "%" después de la subconsulta:

```
select nom_autor from libros where nom_autor like concat((select left(nom_autor,1) from libros where num_paginas = 446),"%");
```

Por lo que finalmente tenemos el resultado:

nombre
RAUL
ROSA
ROSALIA

Ejemplo 2

Otra subconsulta sería en miscelánea sumar una décima parte de la media de las páginas a todos los libros, mostrando el título y el número de páginas. Comenzaríamos por encontrar la décima parte de esa media.

```
select avg(num_paginas)/10 from libros
```

avg(num_paginas)
34.31666667

Una vez tenemos el valor, se podría poner directamente en la consulta superior, pero como en anteriores ocasiones, solo serviría en ese instante y hasta que cambien los datos.

```
select titulo, num_paginas + 34.31666667 from libros;
```

Por ello, donde se colocaría el número, colocamos la consulta entre paréntesis

```
select titulo, num_paginas + (select avg(num_paginas)/10 from libros) from libros;
```

titulo	num_paginas + ...
COMO IR DE VACACIONES SIN DEJARSE EL SUELDO	322.31666667
EL PERIODICO	2008-03-17
...	...
ANA EN CASA	499.31666667

1.5. Subconsultas con listas agrupadas (error full-group-by)

En ocasiones es posible encontrar que no se puede usar una subconsulta que devuelve una lista de datos, obtenidos mediante una función de agrupación, como filtro en otra consulta.

Por ejemplo, en la base de datos starwars:

```
select year(Lanzamiento) as anio from Novela where Editorial not like "%Disney%" group by anio having count(*) > 1;
```

Devuelve:

anio
2015
2018

Pero si la usamos como subconsulta:

```
select * from Novela where year(Lanzamiento) in (select year(Lanzamiento) as anio from Novela where Editorial not like "%Disney%" group by anio having count(*) > 1)
```

Da error. En este caso se ha de crear una tabla temporal con la consulta de agrupación, esto es crear una subconsulta que sea usada en el from:

```
select anio from (select year(Lanzamiento) as anio from Novela where Editorial not like "%Disney%" group by anio having count(ISBN_N) > 1) as s1
```

Que da el mismo resultado. Esta sí se puede usar como subconsulta con el propósito inicial:

```
select * from Novela where year(Lanzamiento) in (select anio from (select year(Lanzamiento) as anio from Novela where Editorial not like "%Disney%" group by anio having count(ISBN_N) > 1) as s1);
```

Cuyo resultado es:

ISBN_N	Titulo	Editorial	Ultima_edicion	Lanzamiento	Descatalogada	SW_Era	Id_a
978-1-101-88495-9	Discipulo Oscuro	Planeta Comic	2016-03-01	2015-07-07	no	Rise of the Empire	8
978-1-368-01630-8	Most Wanted	Disney Lucasfilm Press	NULL	2018-05-14	no	Rise of the Empire	11
978-8-416-40170-3	Consecuencias	Paneta Comic	NULL	2015-11-24	no	Rebellion	12
978-84-9146-889-9	Star Wars La última orden	Planeta Comic	NULL	2018-06-05	no	Rebellion	5
978-84-9173-009-5	Star Wars Ahsoka	Planeta Comic	NULL	2018-10-02	no	Rinse of the Empire	4

1.6. Subconsultas utilizando campos de la consulta superior

Una subconsulta tiene acceso a los valores de la consulta superior que la envuelve. Veamos el siguiente ejemplo:

```
select num, cliente as c2, precio_unidad* unidades -
      (select count(*)*0.1 from facturas  where cliente = c2 group by cliente)
as precio_final from facturas ;
```

Si nos fijamos en el código resaltado en amarillo, la columna c2 se define en la consulta principal, y es utilizada en la subconsulta.

El sentido de esta consulta es el siguiente: quiero descontar a cada línea de factura el equivalente a 10 céntimos por cada producto que haya pedido. Para ello calculamos el descuento con esta consulta:

```
select cliente, count(*)*0.1 from facturas  group by cliente
```

Lo que devuelve el siguiente resultado:

cliente	Descuento
Compraqui	1.0
Super Paquito	3.5
Ahorra Muchisimo	1.4
Baratolandia	2.8
El Barato	1.1
El Campo	1.1
Blanca Lecheria	0.5

Esto ha calculado el descuento que debe tener cada cliente. Ahora bien, si queremos que a cada fila de la tabla facturas se le aplique el descuento concreto a cada cliente, se debe filtrar con un where, pero de una información que está en la consulta superior.

Por ello, a dicha subconsulta se le añade el where y se quita el nombre del cliente del resultado. Esta subconsulta no puede ejecutarse independientemente:

```
select cliente count(*)*0.1 from facturas where cliente = c2 group by cliente
```

Esta consulta por si sola da error, al no existir el campo c2. Pero incrustada como subconsulta en la otra, c2 sí existe, por lo que funciona.

2. Errores típicos

Listado de algunos de los errores típicos, sus explicaciones y posibles soluciones.