
Unidad Didáctica 9. Subconsultas

Apuntes de BD para DAW, DAM y ASIR

José Juan Sánchez Hernández

Curso 2023/2024

Índice

- 1 Subconsultas 1**
 - 1.1 Tipos de subconsultas 1
 - 1.2 Subconsultas en la cláusula WHERE 1
 - 1.3 Subconsultas en la cláusula HAVING 1
 - 1.4 Subconsultas en la cláusula FROM 2
 - 1.5 Subconsultas en la cláusula SELECT 2

- 2 Operadores que podemos usar en las subconsultas 3**
 - 2.1 Operadores básicos de comparación 3
 - 2.2 Subconsultas con ALL y ANY 4
 - 2.3 Subconsultas con IN y NOT IN 4
 - 2.4 Subconsultas con EXISTS y NOT EXISTS 5

- 3 Errores comunes 6**
 - 3.1 Número incorrecto de filas en la subconsulta 6

- 4 Referencias 7**

- 5 Licencia 8**

Índice de figuras

Índice de cuadros

1 Subconsultas

Una subconsulta es una consulta anidada dentro de otra consulta.

Debe tener en cuenta que no existe una única solución para resolver una consulta en SQL. En esta unidad vamos a estudiar cómo podemos resolver haciendo uso de subconsultas, algunas de las consultas que hemos resuelto en las unidades anteriores.

1.1 Tipos de subconsultas

El estándar SQL define tres tipos de subconsultas:

- **Subconsultas de fila.** Son aquellas que devuelven más de una columna pero una única fila.
- **Subconsultas de tabla.** Son aquellas que devuelve una o varias columnas y cero o varias filas.
- **Subconsultas escalares.** Son aquellas que devuelven una columna y una fila.

1.2 Subconsultas en la cláusula WHERE

Por ejemplo, suponga que queremos conocer el nombre del producto que tiene el mayor precio. En este caso podríamos realizar una primera consulta para buscar cuál es el valor del precio máximo y otra segunda consulta para buscar el nombre del producto cuyo precio coincide con el valor del precio máximo. La consulta sería la siguiente:

```
1 SELECT nombre
2 FROM producto
3 WHERE precio = (SELECT MAX(precio) FROM producto)
```

En este caso sólo hay un nivel de anidamiento entre consultas pero pueden existir varios niveles de anidamiento.

1.3 Subconsultas en la cláusula HAVING

Ejemplo: Devuelve un listado con todos los nombres de los fabricantes que tienen el mismo número de productos que el fabricante *Asus*.

```
1 SELECT fabricante.nombre, COUNT(producto.codigo)
2 FROM fabricante INNER JOIN producto
```

```

3 ON fabricante.codigo = producto.codigo_fabricante
4 GROUP BY fabricante.codigo
5 HAVING COUNT(producto.codigo) >= (
6     SELECT COUNT(producto.codigo)
7     FROM fabricante INNER JOIN producto
8     ON fabricante.codigo = producto.codigo_fabricante
9     WHERE fabricante.nombre = 'Asus');

```

1.4 Subconsultas en la cláusula FROM

Ejemplo: Devuelve una listado de todos los productos que tienen un precio mayor o igual al precio medio de todos los productos de su mismo fabricante.

```

1 SELECT *
2 FROM producto INNER JOIN (
3     SELECT codigo_fabricante, AVG(precio) AS media
4     FROM producto
5     GROUP BY codigo_fabricante) AS t
6 ON producto.codigo_fabricante = t.codigo_fabricante
7 WHERE producto.precio >= t.media;

```

1.5 Subconsultas en la cláusula SELECT

Las subconsultas que pueden aparecer en la cláusula `SELECT` tienen que ser subconsultas de tipo **escalar**, que devuelven una única fila y columna.

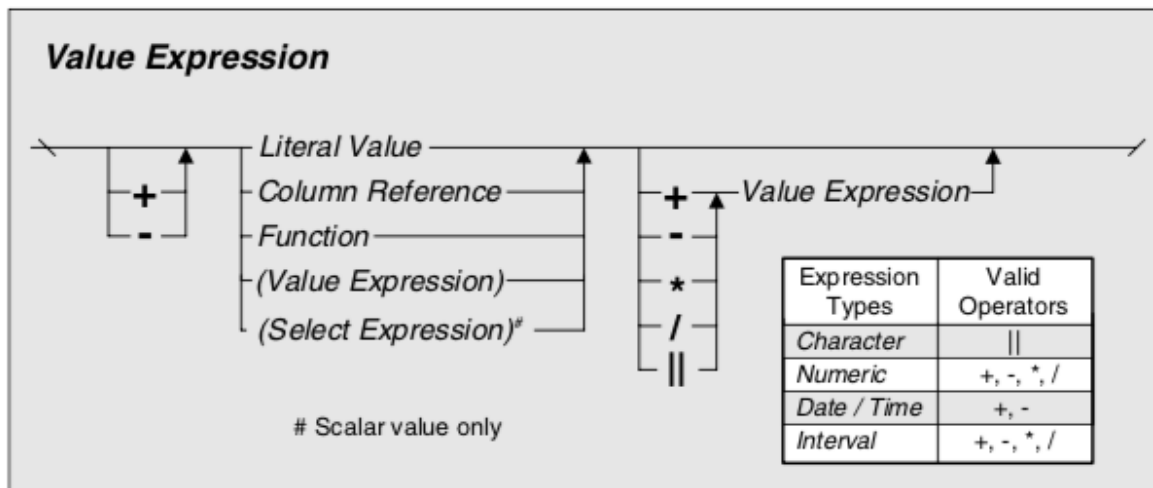


Imagen: Imagen extraída del libro «SQL Queries for Mere Mortals» de John L. Viescas.

2 Operadores que podemos usar en las subconsultas

Los operadores que podemos usar en las subconsultas son los siguientes:

- Operadores básicos de comparación (>, >=, <, <=, !=, <>, =).
- Predicados **ALL** y **ANY**.
- Predicado **IN** y **NOT IN**.
- Predicado **EXISTS** y **NOT EXISTS**.

2.1 Operadores básicos de comparación

Los operadores básicos de comparación (>, >=, <, <=, !=, <>, =) se pueden usar cuando queremos comparar una expresión con el valor que devuelve una subconsulta.

Los operadores básicos de comparación los vamos a utilizar para realizar comparaciones con **subconsultas que devuelven un único valor, es decir, una columna y una fila**.

Ejemplo: Devuelve todos los productos de la base de datos que tienen un precio mayor o igual al producto más caro del fabricante *Asus*.

```
1 SELECT *
2 FROM producto
3 WHERE precio >= (SELECT MAX(precio)
4     FROM fabricante INNER JOIN producto
5     ON fabricante.codigo = producto.codigo_fabricante
6     WHERE fabricante.nombre = 'Asus');
```

La consulta anterior también se puede escribir con subconsultas sin hacer uso de **INNER JOIN**.

```
1 SELECT *
2 FROM producto
3 WHERE precio = (
4     SELECT MAX(precio)
5     FROM producto
6     WHERE codigo_fabricante = (
7         SELECT codigo
8         FROM fabricante
9         WHERE nombre = 'Asus'));
```

2.2 Subconsultas con ALL y ANY

ALL y **ANY** se utilizan con los operadores de comparación (>, >=, <, <=, !=, <>, =) y nos permiten comparar una expresión con el conjunto de valores que devuelve una subconsulta.

ALL y **ANY** los vamos a utilizar para realizar comparaciones con **subconsultas que pueden devolver varios valores, es decir, una columna y varias filas**.

Ejemplo: Podemos escribir la consulta que devuelve todos los productos de la base de datos que tienen un precio mayor o igual al producto más caro del fabricante **Asus**, haciendo uso de **ALL**. Por lo tanto estas dos consultas darían el mismo resultado.

```
1 SELECT *
2 FROM fabricante INNER JOIN producto
3 ON fabricante.codigo = producto.codigo_fabricante
4 WHERE precio >= (SELECT MAX(precio)
5     FROM fabricante INNER JOIN producto
6     ON fabricante.codigo = producto.codigo_fabricante
7     WHERE fabricante.nombre = 'Asus');
```

```
1 SELECT *
2 FROM fabricante INNER JOIN producto
3 ON fabricante.codigo = producto.codigo_fabricante
4 WHERE precio >= ALL (SELECT precio
5     FROM fabricante INNER JOIN producto
6     ON fabricante.codigo = producto.codigo_fabricante
7     WHERE fabricante.nombre = 'Asus');
```

La palabra reservada **SOME** es un alias de **ANY**. Por lo tanto, las siguientes consultas devolverían el mismo resultado:

```
1 SELECT s1 FROM t1 WHERE s1 <> ANY (SELECT s1 FROM t2);
2
3 SELECT s1 FROM t1 WHERE s1 <> SOME (SELECT s1 FROM t2);
```

2.3 Subconsultas con IN y NOT IN

IN y **NOT IN** nos permiten comprobar si un valor está o no incluido en un conjunto de valores, que puede ser el conjunto de valores que devuelve una subconsulta.

IN y **NOT IN** los vamos a utilizar para realizar comparaciones con **subconsultas que pueden devolver varios valores, es decir, una columna y varias filas**.

Ejemplo: Devuelve un listado de los clientes que no han realizado ningún pedido.

```
1 SELECT *
2 FROM cliente
3 WHERE id NOT IN (SELECT id_cliente FROM pedido);
```

Cuando estamos trabajando con subconsultas, **IN** y **ANY** realizan la misma función. Por lo tanto, las siguientes consultas devolverían el mismo resultado:


```
1 SELECT s1 FROM t1 WHERE s1 = ANY (SELECT s1 FROM t2);
2
3 SELECT s1 FROM t1 WHERE s1 IN (SELECT s1 FROM t2);
```

Ocurre lo mismo con `NOT IN` y `<> ALL`. Por lo tanto, las siguientes consultas devolverían el mismo resultado:

```
1 SELECT s1 FROM t1 WHERE s1 <> ALL (SELECT s1 FROM t2);
2
3 SELECT s1 FROM t1 WHERE s1 NOT IN (SELECT s1 FROM t2);
```

Importante:

Tenga en cuenta que cuando hay un valor `NULL` en el resultado de la consulta interna, la consulta externa no devuelve ningún valor.

Ejemplo: Devuelve un listado con el nombre de los departamentos que no tienen empleados asociados.

```
1 SELECT nombre
2 FROM departamento
3 WHERE codigo NOT IN (
4     SELECT codigo_departamento
5     FROM empleado);
```

La consulta interna `SELECT codigo_departamento FROM empleado`, devuelve algunas filas con valores `NULL` y por lo tanto la consulta externa no devuelve ningún valor.

La forma de solucionarlo sería quitando los valores `NULL` de la consulta interna:

```
1 SELECT nombre
2 FROM departamento
3 WHERE codigo NOT IN (
4     SELECT codigo_departamento
5     FROM empleado
6     WHERE codigo_departamento IS NOT NULL);
```

2.4 Subconsultas con EXISTS y NOT EXISTS

Ejemplo: Devuelve un listado de los clientes que no han realizado ningún pedido.

```
1 SELECT *
2 FROM cliente
3 WHERE NOT EXISTS (SELECT id_cliente FROM pedido WHERE cliente.id = pedido.
4     id_cliente);
```

3 Errores comunes

3.1 Número incorrecto de filas en la subconsulta

```
1 ERROR 1242 (ER_SUBSELECT_NO_1_ROW)
2 SQLSTATE = 21000
3 Message = "Subquery returns more than 1 row"
```

```
1 SELECT * FROM t1 WHERE column1 = (SELECT column1 FROM t2);
```

La consulta anterior sólo se podrá ejecutar cuando la consulta interna `SELECT column1 FROM t2` devuelva una única fila.

4 Referencias

- **SQL: The Complete Reference** de James R. Groff y otros.
- **SQL Queries for Mere Mortal** de John L. Viescas.
- [Subquery Syntax](#). Documentación oficial de MySQL.
- [Optimizing Subqueries](#). Documentación oficial de MySQL.

5 Licencia

Esta página forma parte del curso Bases de Datos de José Juan Sánchez Hernández y su contenido se distribuye bajo una licencia Creative Commons Reconocimiento-NoComercial-CompartirIgual 4.0 Internacional.