5. SQL/DQL. Subconsultas

En estos tipos de subconsultas con `SELECT` van a intervenir, como mínimo, dos de estas instrucciones (aunque podríamos seguir anidando hasta conseguir lo que buscamos). Tendremos un `SELECT` principal que de alguna manera estará filtrado por el resultado de la consulta `SELECT` secundaria o subordinada o anidada a la principal.

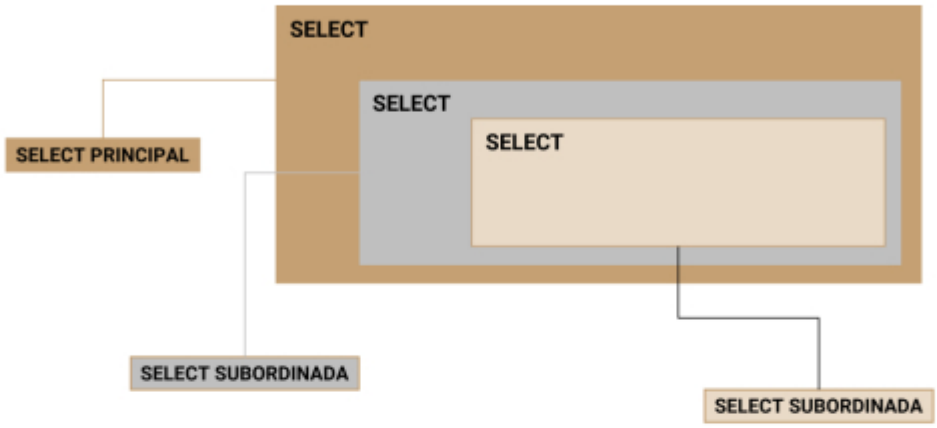

Fig. 6.12. Subconsultas en SQL.

Vamos a ir aprendiendo los tres grupos de subconsultas:

- Las escalonadas, que devolverán un único valor; por ejemplo, la media o el máximo; en fin, suele ser un valor calculado por totales.
- Las de lista, que devolverán una lista o conjunto de resultados; por tanto, es el resultado de cualquier consulta básica con cierto filtro aplicado que devuelve un subconjunto de valores que nos servirán para decidir qué hacer sobre la consulta principal.
- Las correlacionadas son aquellas que tanto la consulta principal como sus subordinadas extraen los valores de esta/s tabla/s, pudiendo ser escalonada o de lista. La única diferencia es que su fuente de datos en ambas es o son la/s misma/s tabla/s.

A. Subconsultas escalonadas

Como ya hemos ido indicando, este tipo de subconsultas persiguen obtener un valor que nos va a permitir utilizar como filtro en la condición `WHERE` y/o `HAVING` de la consulta principal. Por tanto, vamos a tener en cuenta que podemos usar operadores de comparación tradicionales menor <, menor o igual que <=, mayor >, mayor o igual que >=, distinto <> o igual =) porque, como hemos dicho, este tipo de subconsultas devuelve solo un valor.

Vamos a ver un ejemplo de nuestra base de datos MCGRAWHOTELES. Si recordamos, teníamos empleados, pues podemos averiguar primero cuál es el máximo y luego saber quién o quiénes son los que más cobran del hotel. Ejecutaremos la siguiente consulta. Si no os atrevéis a construirla toda directamente, una buena práctica es que escribamos primero la consulta secundaria, verifiquemos que obtenemos el resultado esperado y luego la mentemos entre paréntesis y empezamos a escribir la principal.

```
SELECT * FROM empleado WHERE empleado.sueldoEmp =
            (SELECT MAX(empleado.sueldoEmp) FROM empleado);
```

idEmp	nombreEmp	ApellidoEmp	oficioEmp	sueldoEmp	idHotel
1	Brian	Wilson	Conserge	1000	1
6	Johnny	Cash	Fontanero	1000	1

Fig. 6.13. Resultado de la sentencia `SELECT PRINCIPAL` con subconsultas.

B. Subconsultas de lista

En este tipo de subconsulta, la `SELECT` secundaria nos devuelve más de un valor, o creemos que puedes hacerlo si no hemos probado antes la misma, por eso aquí utilizaremos los operadores de búsqueda literales (`ALL, ANY, EXISTS`) o de comparación literales (`IN`) que vimos. Por tanto, tenemos que acompañarlo con una de ellas para saber qué queremos hacer con el conjunto de datos devuelto por la subconsulta. Vamos a ver algún ejemplo con cada uno de ellos.

B1. Operador ALL

Podemos traducirlo por TODOS los valores de los que nos devuelva la subconsulta. Principalmente, se utiliza con operadores de comparación tradicionales. Vamos a ver un ejemplo: ¿hay un conserje que cobre más que TODOS los directores?

```
SELECT nombreEmp, apellidoEmp FROM empleado WHERE oficioEmp="Conserje"
AND sueldoEmp > ALL (SELECT sueldoEmp FROM empleado WHERE
oficioEmp="Director");
```

B2. Operador ANY

Podemos traducirlo por ALGUNO de los valores de los que nos devuelva la subconsulta. Principalmente se utiliza con operadores de comparación tradicionales. Vamos a ver un ejemplo: ¿hay algún conserje que cobre más que algún director?

```
SELECT nombreEmp, apellidoEmp FROM empleado WHERE oficioEmp="Conserje"
AND sueldoEmp > ANY (SELECT sueldoEmp FROM empleado WHERE
oficioEmp="Director");
```

B3. Operador EXISTS

La cláusula `EXISTS` se utiliza en ciertos tipos de consultas, principalmente:

- Cuando queremos saber si de un conjunto de filas de una tabla todos cumplen una condición que ha sido buscada en otra tabla.
- Cuando queremos saber si de un conjunto de filas de una tabla ninguno cumple una condición que debe ser buscada en otra tabla.
- En consultas complejas que involucran una relación M:M y en las dos tablas originales.

Por ejemplo: si queremos saber en los hoteles en que todas las habitaciones tienen más de una cama. La forma más fácil sería comprobar que no hay ninguno que tenga menos de 2 camas, es decir, no debe haber ninguna habitación en ese hotel que tenga menos de 2 camas.

```
SELECT nombreHotel FROM hotel WHERE NOT EXISTS (SELECT * FROM
habitación WHERE habitacion.idHotel = hotel.idHotel AND camas < 2)
```

B4. Operador IN

Imaginemos que queremos saber el nombre completo de los empleados que trabajan en los hoteles que son de las poblaciones de Carlet o de Sueca. La subconsulta nos devolverá los códigos de los hoteles, y con ellos podremos saber quiénes son esos trabajadores.

```
SELECT nombreEmp, apellidoEmp FROM empleado WHERE idHotel IN (SELECT
idHotel FROM hotel WHERE poblaciónHotel="Carlet" OR
poblaciónHotel="Sueca");
```

C. Subconsultas correlacionadas

Como ya henos indicado y visto en algunos de los ejemplos, este tipo de subconsulta se característica porque todas las `SELECT` implicadas en esta solo obtienen los datos de las mismas fuentes. No son subconsultas cruzadas o entre tablas distintas de las utilizadas en ellas mismas.