



**Universidad Nacional Autónoma de
México
Facultad de Ingeniería**



Base de datos

Profesor:

Ing. Fernando Arreola Franco.

Grupo: 01

**Alumna: Mondragón Hernández Andrea
Quetzalli**

SUBCONSULTAS

Es una sentencia "select" anidada en otra sentencia "select", "insert", "update" o "delete" (o en otra subconsulta).

Las subconsultas se emplean cuando una consulta es muy compleja, entonces se la divide en varios pasos lógicos y se obtiene el resultado con una única instrucción y cuando la consulta depende de los resultados de otra consulta.

Generalmente, una subconsulta se puede reemplazar por combinaciones y estas últimas son más eficientes.

Características:

- Las subconsultas se deben incluir entre paréntesis.
- Puede haber subconsultas dentro de subconsultas.
- Se pueden emplear subconsultas en lugar de una expresión, siempre que devuelvan un solo valor o una lista de valores y que retornen un conjunto de registros de varios campos en lugar de una tabla o para obtener el mismo resultado que una combinación (join).
- En una subconsulta, especifique sólo una columna o expresión a no ser que esté utilizando IN, ANY, ALL o EXISTS.
- Una subconsulta no puede contener una cláusula BETWEEN ni LIKE.
- Una subconsulta no puede contener una cláusula ORDER BY.
- Una subconsulta de una sentencia UPDATE no puede recuperar datos de la misma tabla en la que deben actualizarse los datos.
- Una subconsulta de una sentencia DELETE no puede recuperar datos de la misma tabla de la que deben suprimirse los datos.

Casos de uso, restricciones y ejemplos:

Select

La subconsulta siempre irá entre paréntesis, sin excepciones.

Las subconsultas se pueden anidar dentro de otras subconsultas.

Sintaxis:

```
SELECT listaExpresiones
FROM tabla
WHERE expresión OPERADOR
      (SELECT listaExpresiones
       FROM tabla);
```

Ejemplo:

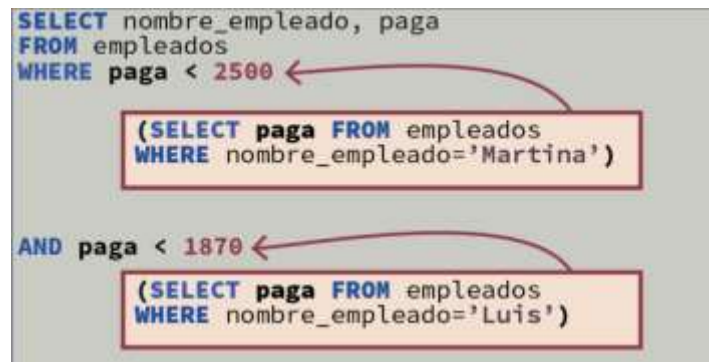
```
SELECT nombre_empleado, paga
FROM empleados
WHERE paga <
      (SELECT paga FROM empleados
       WHERE nombre_empleado='Martina')
;
```

Esa consulta muestra el nombre y paga de los empleados cuya paga es menor que la de la empleada Martina. Para que funcione esta consulta, la subconsulta solo puede devolver un valor (solo puede haber una empleada que se llame Martina).

Se pueden usar subconsultas las veces que haga falta:

```
SELECT nombre_empleado, paga
FROM empleados
WHERE paga <
      (SELECT paga FROM empleados
       WHERE nombre_empleado='Martina')
AND paga >
      (SELECT paga FROM empleado
       WHERE nombre_empleado='Luis');
```

En realidad lo primero que hace la base de datos es calcular el resultado de la subconsulta:



```
SELECT nombre_empleado, paga
FROM empleados
WHERE paga < 2500
      (SELECT paga FROM empleados
       WHERE nombre_empleado='Martina')
AND paga < 1870
      (SELECT paga FROM empleado
       WHERE nombre_empleado='Luis')
```

Una subconsulta que utilice los valores >,<,>=,... tiene que devolver un único valor, de otro modo ocurre un error.

Además, tienen que devolver el mismo tipo y número de datos para relacionar la subconsulta con la consulta que la utiliza (no puede ocurrir que la subconsulta tenga dos columnas y ese resultado se compare usando una sola columna en la consulta general).

From

FROM es utilizada para especificar la tabla de la cual se van a seleccionar los registros

El resultado de una operación de tipo SELECT es una vista (aunque sea temporal, ya que no se almacena de forma permanente). Y las vistas pueden ser utilizadas dentro de otras vistas (al igual que las tablas).

Así una consulta como esta:

```
SELECT tipo,modelo, SUM(cantidad) suma_cantidad
FROM existencias
GROUP BY tipo, modelo
```

Muestra los tipos y modelos de piezas en los almacenes y la suma de cantidades que poseen sumando la de cada almacén. El resultado es una vista de tres columnas. Lo interesante es que puede ser un inicio para una nueva consulta.

Por ejemplo:

```
SELECT tipo, COUNT(modelo), SUM(suma_cantidad)
FROM (
  SELECT tipo,modelo, SUM(cantidad) suma_cantidad
  FROM existencias
  GROUP BY tipo, modelo
)
GROUP BY tipo;
```

Join

```
SELECT tipo,modelo,precio_venta
FROM piezas P1
JOIN (
  SELECT MAX(precio_venta) max_precio_venta
  FROM piezas
) P2 ON P1.precio_venta=P2.max_precio_venta;
```

Así, esta consulta nos muestra el tipo y modelo de las piezas que tiene el precio de venta más alto. No es la única forma de resolver esta consulta, pero nos permite observar la capacidad de usar subconsultas de forma muy avanzada.

A esta técnica se le llama usar vistas en línea (en inglés inline views). Ahora bien, para que eso sea posible las columnas de la subconsulta deben usar alias obligatorios para que no se repita el nombre de la columna y especialmente en las columnas con datos calculados.

Where

WHERE es utilizada para especificar las condiciones que deben reunir los registros que se van a seleccionar

Una cláusula WHERE con las condiciones deseadas más uno de los operadores SQL anteriores (ANY, ALL,SOME,IN,NOT IN, EXISTS o NOT EXISTS) u operadores de comparación como "=", ">" o "<"

La instrucción UPDATE permite modificar filas. Es muy habitual el uso de la cláusula WHERE para indicar las filas que se modificarán. Esta cláusula se puede utilizar con las mismas posibilidades que en el caso del SELECT, por lo que es posible utilizar subconsultas. Por ejemplo:

```
UPDATE empleados
SET sueldo=sueldo*1.10
WHERE id_seccion =(SELECT id_seccion FROM secciones
                    WHERE nom_seccion='Producción');
```

Esta instrucción aumenta un 10% el sueldo de los empleados de la sección llamada Producción.

Es posible utilizar subconsultas correlacionadas.

```
UPDATE empleados e
SET sueldo=sueldo*1.10
WHERE 1 = (SELECT COUNT(*) FROM historial h
           WHERE h.id_empleado=e.id_empleado);
```

Esta consulta aumenta un 10% el sueldo de los empleados si solo han tenido un empleo en su historial.

Having

HAVING Utilizada para expresar la condición que debe satisfacer cada grupo

Una cláusula HAVING con las condiciones deseadas más uno de los operadores SQL anteriores (ANY, ALL, SOME, IN, NOT IN, EXISTS o NOT EXISTS) u operadores de comparación como "=", ">" o "<"

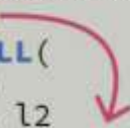
Correlaciones

En las subconsultas a veces se puede desear poder utilizar datos procedentes de la consulta principal. Eso es posible utilizando el alias de la tabla que queremos usar de la consulta principal.

Por ejemplo, supongamos que deseamos obtener de una base de datos geográfica, el nombre y la población de las localidades que sean las más pobladas de su provincia. Es decir, las localidades cuya población es la mayor de su provincia. Para ello necesitamos comparar la población de cada localidad con la de todas las localidades de su provincia. Supongamos que la tabla de las localidades almacena el nombre, población y el número de la provincia a la que pertenecen.

La consulta sería:

```
SELECT l.nombre, poblacion
FROM localidades l
WHERE poblacion>=ALL(
  SELECT poblacion
  FROM localidades l2
  WHERE l2.n_provincia=l.n_provincia
)a
```



En el código anterior se observa que dentro de la subconsulta usamos el alias **l** correspondiente a la tabla de localidades de la consulta principal (por eso se le ha puesto como alias **l2** a la tabla localidades en la subconsulta).

BIBLIOGRAFIAS:

Subconsultas en SQL | 7 Ejemplo para dominar subconsultas. (s. f.). Aprende programación web con tutoriales de calidad | Sr Código Fuente. <https://www.srCodigoFuente.es/subconsultas-en-sql>

IBM Docs. (s. f.). IBM - Deutschland | IBM. <https://www.ibm.com/docs/es/qmf/12.1.0?topic=ddfmtuss-creating-subquery-retrieve-data-from-more-than-one-table>

Tutorial de PostgreSQL - Subconsultas. (s. f.). Tutoriales Programacion Ya. <https://www.tutorialesprogramacionya.com/postgresql/temarios/descripcion.php?inicio=50&cod=216&punto=58>