SUBCONSULTAS MySQL

David Rodera

ÍNDICE

INTRODUCCIÓN

Las subconsultas son **consultas SQL anidadas** que permiten resolver problemas complejos de forma secuencial y legible. Representan otra manera de expresar la misma lógica que los JOINs, con ventajas específicas en claridad.

Beneficios principales:

Legibilidad secuencial: Dividen problemas complejos en pasos lógicos ("primero calcula X, luego filtra por X").

Evitan JOINs complejos: Útiles cuando la lógica es más fácil de entender como pasos que como relaciones.

Expresan existencia/ausencia: EXISTS/NOT EXISTS son más directos que LEFT JOIN + IS NULL para ciertos casos.

Hay distintos tipos de subconsultas: **Escalar en WHERE**, **IN (multivalor)**, **Correlacionada EXISTS**, **Derivada en FROM**, **Escalar en SELECT**. Están se veran más adelante.

PATRONES PARA IDENTIFICARLAS (5 CLAVE)

PALABRAS CLAVE	NECESIDAD LÓGICA	TIPO DE SUBCONSULTA
“Más que la media”, “> promedio”	Calcular valor agregado PRIMERO	Escalar en WHERE
"Pertenece a", "en las de", "que tienen"	Filtrar por pertenencia a conjunto	IN (multivalor)
"Tienen al menos", "sin relación"	Verificar existencia/ausencia	Correlacionada EXISTS
"Más larga/cara de cada categoría/país"	TOP-N o MAX por grupo	Derivada en FROM
"Número total al lado de cada fila"	Valor agregado por fila	Escalar en SELECT

ESCALAR EN WHERE

ESTRUCTURA BÁSICA (PUEDEN HABER VARIACIONES)

```
SELECT
    columna1, columna2
FROM
    tabla1
WHERE
    clausula > ( SELECT
            AVG(columna3)
        FROM
            tabla1);
```

CUANDO USARLA

- "mayor que la media"
- "igual al máximo"
- "menor que el total"

EJEMPLOS

-- Películas más caras que el precio promedio.

```
SELECT
    film_id, title
FROM
    film
WHERE
    replacement_cost > ( SELECT
            AVG(replacement_cost)
        FROM
            film);
```

IN (MULTIVALOR)

ESTRUCTURA BÁSICA (PUEDEN HABER VARIACIONES)

```
SELECT
    columna1, columna2
FROM
    tabla1
WHERE
    columna1 IN ( SELECT
            columna1
        FROM
            tabla1
        WHERE
            clausula = '');
```

CUANDO USARLA

- "Pertenece a las de categoría X"
- "Que han participado en películas Y"
- "Clientes que han alquilado películas Z"

EJEMPLOS

-- Actores que han participado en películas de la categoría 'Action'.

```
SELECT
    actor_id, first_name, last_name
FROM
    actor
WHERE
    actor_id IN ( SELECT
            actor_id
        FROM
            film_actor
                JOIN
            film USING (film_id)
                JOIN
            film_category USING (film_id)
                JOIN
            category USING (category_id)
        WHERE
            name = 'Action');
```

CORRELACIONADA EXISTS

ESTRUCTURA BÁSICA (PUEDEN HABER VARIACIONES)

```
SELECT
    columna1, columna2
FROM
    tabla1
WHERE EXISTS ( SELECT
                1
            FROM
                tabla2
            WHERE
                tabla1.columna1 = tabla2.columna1);
```

CUANDO USARLA

- "Tienen al menos un X"
- "Sin ningún Y relacionado" (NOT EXISTS)
- "Clientes que han alquilado algo"

EJEMPLOS

-- Clientes que tienen al menos un alquiler.

```
SELECT
    customer_id, first_name, last_name
FROM
    customer
WHERE EXISTS ( SELECT
                1
            FROM
                rental
            WHERE
                customer.customer_id = rental.customer_id);
```

DERIVADA EN FROM

ESTRUCTURA BÁSICA (PUEDEN HABER VARIACIONES)

```
SELECT
    columna1, columna2, columna_creada
FROM
    (SELECT
        columna1, MAX(columna3) AS columna_creada
    FROM
        tabla1
    GROUP BY columna1) AS tabla_creada
        JOIN
    tabla1 USING (columna1)
WHERE
    tabla1.columna3 = tabla_creada.columna_creada;
```

CUANDO USARLA

- "**Más [valor] de cada** [grupo/categoría]"
- "TOP-N por categoría/país"
- "Primero agrupa X, luego filtra los grupos"

EJEMPLOS

-- Película más cara de cada idioma.

```
SELECT
    language_id, name, film_id, title, max_precio
FROM
    (SELECT
        language_id, MAX(replacement_cost) AS max_precio
    FROM
        film
    GROUP BY language_id) AS maximos
        JOIN
    film USING (language_id)
        JOIN
    language USING (language_id);
```

ESCALAR EN SELECT

ESTRUCTURA BÁSICA (PUEDEN HABER VARIACIONES)

```
SELECT
    columna1,
    (SELECT
            COUNT(*)
        FROM
            tabla2
                JOIN
            tabla3 USING (columna2)
        WHERE
                        tabla2.columna  =  tabla1.columna1)  AS
columna_creada
FROM
    tabla1;
```

CUANDO USARLA

- "Número total de X al lado de cada Y"
- "Mostrar cada película + sus alquileres"
- "Cliente + total gastado"

EJEMPLOS

-- Cada película junto con su número total de alquileres.

```
SELECT
    film_id,
    title,
    (SELECT
            COUNT(*)
        FROM
            rental
                JOIN
            inventory USING (inventory_id)
        WHERE
                        inventory.film_id  =  film.film_id)  AS
total_alquileres
FROM
    film;
```

WITH

La cláusula **WITH** crea tablas temporales dentro de una consulta única, mejorando la legibilidad y evitando subconsultas repetidas.

ESTRUCTURA BÁSICA (PUEDEN HABER VARIACIONES)

```
WITH nombre_tabla AS (
    SELECT
        columna1, columna2
    FROM
        tabla1
    GROUP BY columna1
)
SELECT ... FROM nombre_tabla;
```

PARA PRACTICAR

1. Película(s) más larga(s) por categoría
2. Número de películas sin stock disponible en ninguna tienda
3. Recaudación mensual por categoría en 2005
4. Clientes con alquileres pero sin pagos registrados
5. Cliente(s) que más ha(n) gastado en cada país
6. Categorías con ingresos superiores a la media global

SOLUCIONES

1.

```
SELECT
    category_id, name, film_id, title, length
FROM
    (SELECT
        category_id, MAX(length) AS max_length
    FROM
        film
    JOIN film_category using (film_id)
    GROUP BY category_id) AS maximum
        JOIN
    film_category USING (category_id)
        JOIN
    film USING (film_id)
        JOIN
    category USING (category_id)
WHERE
    film.length = maximum.max_length;
```

2.

```
SELECT
    COUNT(*) AS num_unavailable_films
FROM
    film
WHERE
    NOT EXISTS ( SELECT
            1
        FROM
            inventory
        WHERE
            inventory.film_id = film.film_id);
```

3.

```
SELECT
    MONTHNAME(payment_date) AS month, SUM(amount) AS amount
FROM
    payment
WHERE
    YEAR(payment_date) = 2005
GROUP BY
    month;
```

4.

```
SELECT
    customer_id, first_name, last_name
FROM
    customer
WHERE EXISTS ( SELECT
                1
            FROM
                rental
            WHERE
                customer.customer_id = rental.customer_id)
AND NOT EXISTS ( SELECT
                1
            FROM
                payment
            WHERE
                customer.customer_id = payment.customer_id);
```

5.

```
SELECT
     country_id, country, customer_id, first_name, last_name,
amount
FROM
    (SELECT
        customer_id, MAX(amount) AS max_amount
    FROM
        payment
    GROUP BY customer_id) AS maximum
        JOIN
    payment USING (customer_id)
        JOIN
    customer USING (customer_id)
        JOIN
    address USING (address_id)
        JOIN
    city USING (city_id)
        JOIN
    country USING (country_id)
WHERE
    payment.amount = maximum.max_amount;
```

6.

```
WITH category_amount AS (
SELECT
    category_id, name, SUM(amount) as total_amount
FROM
    category
        JOIN
    film_category USING (category_id)
        JOIN
    film USING (film_id)
        JOIN
    inventory USING (film_id)
        JOIN
    rental USING (inventory_id)
        JOIN
    payment USING (rental_id)
GROUP BY category_id, name)
SELECT
    category_id, name, total_amount
FROM
    category_amount
WHERE total_amount > ( SELECT
            AVG(total_amount)
         FROM
            category_amount)
```