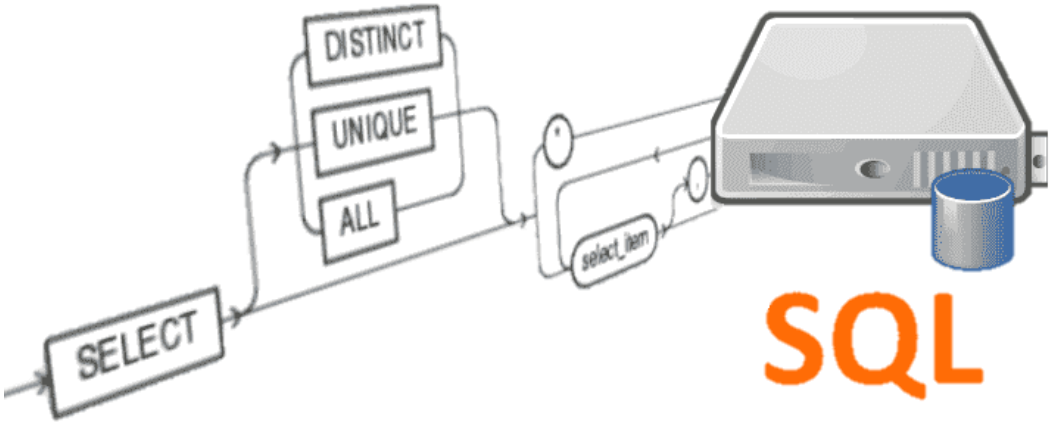
SQL

Structured Query Language

(Lenguaje de Consulta Estructurada)



SubConsultas

Las subconsultas son consultas anidadas dentro de otra consulta principal. Estas subconsultas permiten realizar operaciones avanzadas como filtros, cálculos y comparaciones basadas en otros datos.

Tipos de Subconsultas

Subconsultas Escalares: Devuelven un único valor (una sola fila y columna), que puede utilizarse en expresiones de la consulta externa.

Subconsultas de Fila Única: Devuelven una sola fila de datos, pero pueden tener varias columnas. Se usan cuando se necesita un conjunto específico de valores.

SubConsultas

Subconsultas de Múltiples Filas: Devuelven varias filas de una sola columna y, generalmente, se utilizan con operadores como IN, ANY, o ALL.

Subconsultas Correlacionadas: Se ejecutan una vez por cada fila de la consulta externa, ya que hacen referencia a las columnas de la consulta principal.

Subconsulta Escalar

Encontrar a los empleados cuyo sueldo es igual al sueldo promedio.

SELECT nombre, sueldo
FROM empleados
WHERE sueldo = (**SELECT AVG**(sueldo) **FROM** empleados);

(SELECT AVG(salario**) FROM** empleados**)** es una subconsulta escalar que devuelve el sueldo promedio (**un solo resultado**). La consulta externa selecciona a los empleados cuyo sueldo es mayor que el promedio.

Subconsulta Escalar

Se debe tener en cuenta la cantidad de registros que devuelve la consulta interna.

Mostrar los empleados no trabajan en ventas que trabajan

```
SELECT nombre, sueldo
FROM empleados
WHERE id_dpto = (SELECT id_dpto
                 FROM dpto WHERE dpto <> 'ventas');
```

(SELECT id_dpto FROM dpto WHERE dpto <> 'ventas') devuelve el id_dpto de todos los departamentos diferentes de 'ventas'. La consulta va a dar error si hay mas de un departamento que no sea ventas, debido a que trata de igualar un valor (único) con un conjunto de valores.

Subconsulta Múltiples Filas

Mostrar los empleados no trabajan en ventas que trabajan

```
SELECT nombre, sueldo
FROM empleados
WHERE id_dpto IN (SELECT id_dpto
                  FROM dpto WHERE dpto <> 'ventas');
```

(SELECT id_dpto FROM dpto WHERE dpto <> 'ventas') devuelve los departamentos que no son ventas. La consulta externa selecciona a los empleados en esos departamentos, a diferencia de la consulta anterior, esta usa **IN** en lugar de **=**

Subconsulta Correlacionada

Buscar los empleados cuyo sueldo es superior al sueldo promedio en su propio departamento.

```
SELECT nombre, sueldo
FROM empleados e1
WHERE sueldo > (SELECT AVG(sueldo) FROM empleados e2
                WHERE e2.id_depto = e1.id_depto);
```

```
SELECT AVG(salario) FROM empleados e2
WHERE e2.id_depto = e1.id_depto)
```

se ejecuta para cada empleado, cada empleado se compara con el sueldo promedio de su departamento.

Ejemplo – Subconsulta IN/ALL

Encontrar todos los clientes que han realizado al menos un pedido.

```
SELECT nombre
FROM clients
WHERE id_cliente IN (SELECT id_cliente FROM pedidos);
```

Encuentra los empleados con sueldo mayor al sueldo de **cualquier** empleado en el departamento de Ventas.

```
SELECT nombre, salario
FROM empleados
WHERE salario > ANY (SELECT sueldo
                     FROM empleados
                     WHERE departamento = 'Ventas');
```

Ejemplo – Subconsulta ALL

Encontrar los empleados cuyo sueldo sea mayor al sueldo de todos los empleados del departamento de Ventas

```
SELECT nombre, sueldo
FROM empleados
WHERE sueldo > ALL (SELECT sueldo FROM empleados WHERE
                    departamento = 'Ventas');
```

Ejemplo – Subconsulta EXISTS

EXISTS verifica si la subconsulta devuelve al menos una fila. comprueba la existencia de registros en otra tabla que cumplan cierta condición

Encontrar todos los clientes que han realizado algún pedido

SELECT nombre **FROM** clientes c **WHERE EXISTS**
(**SELECT** 1 **FROM** pedidos p **WHERE** p.id_cliente = c.id_cliente)

(SELECT 1 FROM pedidos p WHERE p.id_cliente = c.id_cliente)

busca en pedidos, si se encuentra al menos un pedido que coincida con id_cliente, devuelve TRUE, y el cliente es seleccionado en la consulta principal.

Manipulación de datos - UPDATE

Actualizar datos usando subconsultas. Para actualizar datos en función de valores en otras tablas o basados en alguna consulta compleja, se usa una subconsulta en la cláusula SET o WHERE

Aumentar la categoría de los clientes que tengan al menos un pedido realizado en el año 2023

UPDATE clientes **SET** categoría = categoría + 1
WHERE idcliente **IN** (**SELECT** idcliente **FROM** pedido
WHERE año = 2023)

Este comando actualizará la categoría de los clientes que hicieron pedidos en 2023, sumándole 1 a la categoría actual.

Ejemplo – Update con subconsulta

Actualizar el precio de cada producto para que sea el precio máximo de los productos dentro de su misma categoría

Producto(id_producto, nombre, precio, id_categoría)

```
UPDATE producto
SET precio = (SELECT MAX(precio) FROM productos AS p
              WHERE p.id_categoria = productos.id_categoria)
```

La subconsulta devuelve el precio máximo dentro de cada categoría, actualizando así todos los productos al precio máximo de su categoría.

Ejemplo - Update con subconsulta

Actualizar el sueldo de los empleados que cobran menos del promedio de su departamento a dicho valor.

Empleado(id_empleado, nombre, sueldo, id_depto)

```
UPDATE empleado
SET sueldo =        (SELECT AVG(sueldo)  FROM empleado AS e
                     WHERE e.id_depto = empleado.id_depto)
WHERE sueldo <  (SELECT AVG(sueldo) FROM empleado AS e
                     WHERE e.id_depto = empleado.id_depto)
```

La subconsulta devuelve el promedio del salario dentro del id_depto correspondiente de cada empleado

Manipulación de datos - DELETE

Eliminar datos usando subconsultas. Eliminar a todos los clientes que no tienen ningún pedido asociado

DELETE FROM cliente
WHERE id_cliente **NOT IN** (**SELECT** id_cliente **FROM** pedido)

Elimina todos los clientes cuyo id_cliente no se encuentra en la tabla pedido.

Ejemplo – Delete con Subconsulta

Eliminar clientes sin pedidos recientes (en el último año)

Cliente(id_cliente, nombre)
Pedido(id_pedido, id_cliente, fecha_pedido)

```
DELETE FROM cliente
WHERE id_cliente NOT IN (SELECT id_cliente
                         FROM pedido
                         WHERE fecha_pedido >= '2023-01-01')
```

Elimina los clientes cuyo id_cliente no aparece en la subconsulta (que contiene los clientes que sí tienen pedidos desde el 1 de enero de 2023)

Ejemplo – Delete con Subconsulta

Eliminar productos que nunca fueron comprados

Producto(id_producto, nombre, precio)
Venta(id_venta, id_producto, cantidad)

```
DELETE FROM producto
WHERE id_producto NOT IN (SELECT DISTINCT id_producto
                          FROM venta)
```

Elimina los productos cuyo id_producto no aparece en la tabla ventas. El uso de DISTINCT es opcional, pero ayuda a evitar duplicados en la subconsulta, lo que mejora el rendimiento.

Ejemplo – UNION (U de A.R.)

El operador UNION se usa para combinar el resultado de dos o más consultas. Las filas duplicadas se eliminan por defecto, pero se pueden incluir utilizando UNION ALL.

Listar los clientes y empleados que residen en Madrid

SELECT nombre, ciudad **FROM** clientes **WHERE** ciudad = 'Madrid'
UNION
SELECT nombre, ciudad **FROM** empleados **WHERE** ciudad = 'Madrid'

Ejemplo – INTERSECT (∩ de A.R.)

El operador INTERSECT se usa para devolver las filas que están presentes en ambas consultas. No está disponible en todas las implementaciones de SQL (por ejemplo, MySQL no lo soporta directamente).

Listar los productos que están los inventarios de las tiendas A y B

SELECT producto **FROM** inventario_tienda_a
INTERSECT
SELECT producto **FROM** inventario_tienda_b

En MySQL, ya que no tiene soporte directo para INTERSECT, puedes lograr un resultado equivalente usando una combinación de JOIN o IN en una subconsulta.

Ejemplo – INTERSECT (∩ de A.R.)

Alternativa 1: Usar un INNER JOIN

```
SELECT a.producto
FROM inventario_tienda_a AS a
INNER JOIN inventario_tienda_b AS b ON a.producto = b.producto
```

Alternativa 2: Usar IN con subconsultas

```
SELECT producto
FROM inventario_tienda_a
WHERE producto IN (SELECT producto FROM inventario_tienda_b)
```

Ejemplo – EXCEPT (resta de A.R.)

El operador EXCEPT devuelve las filas que están en la primera consulta pero no en la segunda

SELECT idCliente **FROM** cliente
EXCEPT
SELECT idCliente **FROM** pedidos

Esta consulta devolverá los IDs de los clientes que no han hecho pedidos

Ejemplo – DIVISIÓN (/ de A.R.)

La división se usa cuando quieres encontrar las entidades que están asociadas con todos los elementos de un conjunto. En SQL, se simula usando NOT EXISTS o GROUP BY con HAVING

Queremos encontrar los empleados que han obtenido todas las certificaciones requeridas

Tabla

empleado (id_empleado, nombre, …)
certificacion (id_certificación, certificacion, requeridas)
certificacion_empleado (id_empleado, id_certificacion)

Ejemplo – DIVISIÓN (/ de A.R.)

```
SELECT id_empleado
FROM certificacion_empleado
GROUP BY id_empleado
HAVING COUNT(DISTINCT id_certificacion) =
(SELECT COUNT(DISTINCT id_certificacion) FROM certificacion
                                        WHERE requeridas ='si' );
```

```
SELECT COUNT(DISTINCT id_certificacion) FROM certificacion
                                        WHERE requeridas ='si'
```

Esta subconsulta calcula el número total de certificaciones requeridas

Ejemplo – DIVISIÓN (/ de A.R.)

```
SELECT id_empleado
FROM certificacion_empleado
GROUP BY id_empleado
HAVING COUNT(DISTINCT id_certificacion) =
(SELECT COUNT(DISTINCT id_certificacion) FROM certificacion
                                        WHERE requeridas ='si' );
```

Consulta principal Agrupa los registros de certificacion_empleado por id_empleado. Usa HAVING para seleccionar solo los empleados cuyo número de certificaciones distintas sea igual al número de certificaciones requeridas.

Ejemplo – DIVISIÓN (/ de A.R.)

En vez de comparar cantidades de certificaciones, podemos usar NOT EXISTS para asegurarnos de que no hay ninguna certificación requerida que un empleado no posea.

```
SELECT DISTINCT ce.id_empleado FROM certificacion_empleado ce
WHERE NOT EXISTS (SELECT cr.id_certificacion FROM certificacion cr
                  WHERE cr.requeridas = 'si' AND NOT EXISTS
                  (SELECT 1 FROM certificacion_empleado ce_inner
                  WHERE ce_inner.id_empleado = ce.id_empleado
                  AND ce_inner.id_certificacion = cr.id_certificacion));
```

Ejemplo – DIVISIÓN (/ de A.R.)

Subconsulta más interna (NOT EXISTS):

```
SELECT 1 FROM certificacion_empleado ce_inner
WHERE ce_inner.id_empleado = ce.id_empleado
 AND ce_inner.id_certificacion = cr.id_certificación
```

Esta subconsulta verifica si el empleado actual (ce.id_empleado) tiene una certificación específica (cr.id_certificacion)

Condiciones:

ce_inner.id_empleado = ce.id_empleado: Verifica que estamos evaluando certificaciones del empleado actual.

ce_inner.id_certificacion = cr.id_certificacion: Verifica si el empleado posee la certificación requerida.

Ejemplo – DIVISIÓN (/ de A.R.)

Subconsulta externa (NOT EXISTS)

SELECT cr.id_certificacion **FROM** certificacion cr
WHERE cr.requeridas = 'si' **AND NOT EXISTS** (...)

certificacion (cr): Es la tabla que contiene la lista de las certificaciones, incluyendo las que son requeridas.

WHERE cr.requeridas = 'si': Filtra las certificaciones que son "requeridas"

AND NOT EXISTS (...): Evalúa si una certificación requerida no ha sido obtenida por el empleado actual. Si esta condición es verdadera, significa que el empleado no cumple con al menos una certificación requerida, y por lo tanto, será excluido del resultado final.

Ejemplo – DIVISIÓN (/ de A.R.)

Consulta principal

SELECT DISTINCT ce.id_empleado
FROM certificacion_empleado ce
WHERE NOT EXISTS (...)

certificacion_empleado (ce): Es la tabla que relaciona empleados con certificaciones obtenidas.

WHERE NOT EXISTS (...): Esta condición evalúa para cada empleado si NO existen certificaciones requeridas que no posea. Es decir, la consulta devuelve empleados que cumplan con todas las certificaciones requeridas.

Vistas

Las vistas son consultas guardadas como objetos en la base.

Características de las Vistas

Simplificación de Consultas: oculta la complejidad a los usuarios, proporciona acceso más sencillo a datos específicos.

Seguridad: limita el acceso a ciertas columnas o datos sensibles

Mantenimiento de Integridad: Permite acceder a datos actualizados sin duplicarlos, lo que ayuda a evitar inconsistencias.

CREATE VIEW nombre_vista **AS** (**SELECT** columnas **FROM** tablas
WHERE condiciones);

Ejemplo - Vistas

Vista para Filtrar Datos

Mostrar a los empleados del sector ventas sin que el usuario final tenga acceso a toda la tabla

CREATE VIEW vista_ventas **AS** (**SELECT** id_empleado, nombre, sueldo
FROM empleado
WHERE sector = 'ventas')

Ahora, cuando alguien consulta vista_ventas, solo verá los empleados del sector Ventas

SELECT * **FROM** vista_ventas;

Ejemplo – Vistas

Vista con Datos Agregados.

Mostrar el total de pedidos por cliente en una vista.

CREATE VIEW total_pedidos_cliente **AS**
(SELECT id_cliente, **SUM**(monto) **AS** total_monto
FROM pedido **GROUP BY** id_cliente**)**

Cuando consultas esta vista, se obtiene el monto total de los pedidos para cada cliente

SELECT * **FROM** total_pedidos_cliente;

Ejemplo – Vistas con JOIN

Muestre información los detalles del cliente junto con la de su último pedido.

```
CREATE VIEW vista_clientes_pedidos AS
(SELECT c.id_cliente, c.nombre, p.id_pedido, p.fecha, p.monto
FROM cliente c
INNER JOIN pedido p ON c.id_cliente = p.id_cliente
WHERE p.fecha = (SELECT MAX(fecha) FROM pedido
WHERE pedido.id_cliente = c.id_cliente))
```

```
SELECT * FROM vista_clientes_pedidos WHERE fecha > ‘2024-11-01’;
```

Ejemplo – Vistas Agregación

Muestre la cantidad de días que cada empleado ha trabajado en el último mes.

```
CREATE VIEW vista_asistencias_mensuales AS
(SELECT e.id_empleado, e.nombre, COUNT(a.id_empleado) AS dias
FROM empleados e
LEFT JOIN asistencias a ON e.id_empleado = a.id_empleado
WHERE a.fecha >= DATEADD(MONTH, -1, GETDATE())
GROUP BY e.id_empleado, e.nombre)
```

```
SELECT nombre, dias
FROM vista_asistencias_mensuales
WHERE dias < 20
```

Optimización

Las subconsultas pueden impactar en el rendimiento y a veces es mejor usar INNER JOIN, en ocasiones no mas alternativas que subconsultas. Por ejemplo, se necesita encontrar los empleados que tienen la función de vendedor

SELECT e.id_empleado, p.nombre **FROM** empleado e
INNER JOIN persona p **on** p.id_persona = e.id_empleado **INNER JOIN** funcion f **ON** e.id_funcion = f.id_función **WHERE** funcion **LIKE** 'vendedor'

SELECT e.id_empleado, p.nombre **FROM** empleado e **INNER JOIN** persona p **on** p.id_persona = e.id_empleado
WHERE e.id_función = (**SELECT** id_funcion **FROM** funcion
WHERE funcion **LIKE** 'vendedor')

Optimización

EXPLAIN ANALYZE es un comando en SQL que se utiliza para evaluar el rendimiento de una consulta.

EXPLAIN ANALYZE SELECT e.id_empleado, p.nombre **FROM** empleado e **INNER JOIN** persona p **on** p.id_persona = e.id_empleado **INNER JOIN** funcion f **ON** e.id_funcion = f.id_función **WHERE** funcion **LIKE** 'vendedor’

(actual time=0.259..1.57 rows=300 loops=1)

EXPLAIN ANALYZE SELECT e.id_empleado, p.nombre **FROM** empleado e **INNER JOIN** persona p **on** p.id_persona = e.id_empleado **WHERE** e.id_función = (**SELECT** id_funcion **FROM** funcion **WHERE** funcion **LIKE** 'vendedor')

(actual time=0.169..2.41 rows=300 loops=1)

Muchas Gracias