

TEMA 11. LAS SUBCONSULTAS

Una **subconsulta** es una sentencia **SELECT** que aparece dentro de otra sentencia **SELECT** que llamaremos **consulta principal**.

Se pueden encontrar subconsultas **en la lista de selección, en la cláusula WHERE o en la cláusula HAVING** de la consulta principal.

Uno de los usos típicos es su utilización para filtrar datos de otra consulta.

Una subconsulta tiene la misma sintaxis que una sentencia **SELECT** normal exceptuando que aparece **encerrada entre paréntesis**, no puede contener la cláusula **ORDER BY**, ni puede ser la **UNION** de varias sentencias **SELECT**, además tiene algunas restricciones en cuanto a número de columnas según el lugar donde aparece en la consulta principal. Estas restricciones las iremos describiendo en cada caso.

Cuando se ejecuta una consulta que contiene una subconsulta, **la subconsulta se ejecuta por cada fila de la consulta principal**.

Se aconseja no utilizar campos calculados en las subconsultas, ralentizan la consulta.

Las consultas que utilizan subconsultas suelen ser **más fáciles de interpretar por el usuario**.

Referencias externas


A menudo, es necesario, dentro del cuerpo de una subconsulta, hacer referencia al valor de una columna en la fila actual de la consulta principal, ese nombre de columna se denomina **referencia externa**.

Una **referencia externa** es un nombre de columna que estando en la subconsulta, no se refiere a ninguna columna de las tablas designadas en la **FROM** de la subconsulta sino a una **columna de las tablas designadas en la FROM de la consulta principal**. Como la subconsulta se ejecuta por cada fila de la consulta principal, el valor de la referencia externa irá cambiando.

Ejemplo: Obtener el código, nombre y la fecha de su primer pedido de cada uno de los empleados.

```
SELECT CodEmpleado, Nombre, (SELECT MIN(FechaPedido) FROM Pedidos
AS P WHERE CodEmpleado = CodRepresentante)
FROM Empleados AS E;
```

```
/*Referencias Externas|*/
SELECT CodEmpleado, Nombre, (
    SELECT Padmin(FechaPedido)
    FROM Pedidos AS Ped
    WHERE Emp.CodEmpleado = Ped.CodRepresentante)
FROM Empleados AS Emp
```



En este ejemplo la consulta principal es **SELECT... FROM empleados**.

La subconsulta es **(SELECT MIN(fechapedido) FROM pedidos WHERE CodEmpleado = CodRepresentante)**.

En esta subconsulta tenemos una referencia externa (*CodEmpleado*) es un campo de la tabla Empleados (origen de la consulta principal).

¿Qué pasa cuando se ejecuta la consulta principal?

- se coge el primer empleado y se calcula la subconsulta sustituyendo *CodEmpleado* por el valor que tiene en el primer empleado. La subconsulta obtiene la fecha más antigua en los pedidos del CodRepresentante = 101,
- se coge el segundo empleado y se calcula la subconsulta con *CodEmpleado* = 102 (*CodEmpleado* del segundo empleado), y así sucesivamente hasta llegar al último empleado.
- Al final obtenemos una lista con el número, nombre y fecha del primer pedido de cada empleado.

```
SELECT CodEmpleado, Nombre, (SELECT MIN(FechaPedido) FROM Pedidos AS P )  
FROM Empleados AS E;
```

Si quitamos la cláusula **WHERE** de la subconsulta obtenemos la fecha del primer pedido de todos los pedidos no del empleado correspondiente.

Si queremos que se muestren únicamente los empleados que tienen pedidos podemos añadir otra subconsulta en la cláusula **WHERE**

```
SELECT CodEmpleado, Nombre,  
(SELECT MIN(FechaPedido) FROM Pedidos AS P WHERE CodEmpleado =  
CodRepresentante )  
FROM Empleados AS E  
WHERE CodEmpleado IN (SELECT codRepresentante FROM Pedidos);
```

Anidar subconsultas

Las subconsultas pueden **anidarse** de forma que **una subconsulta aparezca en la cláusula WHERE** (por ejemplo) **de otra subconsulta** que a su vez forma parte de otra consulta principal. En la práctica, una consulta consume mucho más tiempo y memoria cuando se incrementa el número de niveles de anidamiento. La consulta resulta también más difícil de leer, comprender y mantener cuando contiene más de uno o dos niveles de subconsultas.

Ejemplo: Obtener el nombre del representante de Alberto Juanes Álvarez

```
SELECT CodEmpleado, Nombre  
FROM Empleados  
WHERE CodEmpleado IN  
(SELECT CodRepresentante FROM Pedidos WHERE CodCliente = (SELECT  
CodCliente FROM Clientes WHERE Nombre = 'Alberto Juanes Álvarez'));
```

En este ejemplo, por cada línea de pedido se calcula la subconsulta de clientes, y esto se repite por cada empleado, en el caso de tener 10 filas de empleados y 200 filas de pedidos (tablas realmente pequeñas), la subconsulta más interna se ejecutaría 2000 veces (10 x 200).

Esta consulta también se podría escribir de la siguiente forma:

```
SELECT DISTINCT CodEmpleado, e.Nombre
FROM Empleados e JOIN Pedidos ON codRepresentante = codEmpleado
JOIN Clientes c USING (codCliente)
WHERE c.Nombre = 'Alberto Juanes Álvarez';
```

Tabular Explain							
id	select_type	table	partitions	type	possible_keys	key	key... ref
1	PRIMARY	Empleados		ALL	PRIMARY		
1	PRIMARY	<subquery2>		eq_ref	<auto_distinct_key>	<auto_distinct_key>	20 aaempresasabc01.Empleados.codEm...
2	MATERIALIZED	Pedidos		ref	FKPedClientes,FKPedEmpleados	FKPedClientes	20 const
3	SUBQUERY	Clientes		ALL			

Subconsulta en la lista de selección

Este método permite utilizar una subconsulta como un campo, es este caso necesitaremos hacer uso de referencias externas para relacionar la subconsulta con el registro.

Cuando la subconsulta aparece **en la lista de selección** de la consulta principal, la subconsulta, **no puede devolver varias filas ni varias columnas**, de lo contrario se da un mensaje de error, es decir, **el valor devuelto por la subconsulta debe ser un único valor.**

Muchos SQLs no permiten que una subconsulta aparezca en la lista de selección de la consulta principal pero eso no es ningún problema ya que normalmente se puede obtener lo mismo utilizando como origen de datos las dos tablas.

Obtener la fecha del primer pedido de cada representante

```
SELECT CodEmpleado, Nombre, (SELECT MIN(FechaPedido) FROM Pedidos
WHERE codEmpleado = codRepresentante)
FROM Empleados;
```

```
SELECT CodEmpleado, Nombre, (SELECT MIN(FechaPedido))
FROM Empleados LEFT JOIN Pedidos ON Empleados.CodEmpleado =
Pedidos.CodRepresentante
GROUP BY CodEmpleado, Nombre;
```

El ejemplo anterior se puede obtener de la siguiente forma:

```
SELECT CodEmpleado, Nombre, MIN(FechaPedido)
FROM Empleados LEFT JOIN Pedidos ON Empleados.CodEmpleado =
Pedidos.CodRepresentante
GROUP BY CodEmpleado, Nombre
```

En la cláusula FROM

En la cláusula FROM se puede encontrar una sentencia SELECT encerrada entre paréntesis pero **más que subconsulta sería una consulta** ya que no se ejecuta para cada fila de la tabla origen sino que se ejecuta una sola vez al principio, su resultado se combina con las filas de la otra tabla para formar las filas origen de la SELECT primera y no admite referencias externas.

En este caso mediante la subconsulta estamos generando una tabla que nos servirá como tabla de búsqueda.

En la cláusula FROM vimos que se podía poner un nombre de tabla o un nombre de consulta, pues en vez de poner un nombre de consulta se puede poner directamente la sentencia SELECT correspondiente a esa consulta encerrada entre paréntesis.

Ejemplo:

```
SELECT subConsulta.CodEmpleado, subConsulta.nombre FROM
  (SELECT CodEmpleado, nombre, sueldo , sueldo + IFNULL(comision,
    0) FROM Empleados as emp WHERE categoria = 'Representante') AS
  SubConsulta
WHERE subConsulta.sueldo > 3000;
```

Si utilizamos una subconsulta en la cláusula FROM hay que ponerle un alias

Subconsulta en las cláusulas WHERE y HAVING

Se suele utilizar subconsultas en las cláusulas WHERE o HAVING cuando los datos que queremos visualizar están en una tabla pero para seleccionar las filas de esa tabla necesitamos un dato que está en otra tabla.

Mediante subconsultas en la cláusula WHERE podemos extraer información sobre tablas externas, para restringir la consulta principal a un atributo dado por la subconsulta.

Ejemplo: Listar los nombres de los clientes que tiene asignado el representante García Gómez, Luis Antonio (suponiendo que no puede haber representantes con el mismo nombre).

/* 1. En primer lugar obtenemos el CodEmpleado de García Gómez, Luis Antonio */

```
SELECT *
FROM Empleados
WHERE nombre = 'García Gómez, Luis Antonio';
```

	codEmpleado	nombre	fecNacimiento	oficina	categoria	fecContrato	codJefe	sueldo	comision	retencionesIRPF	retencionesSS	objetivo
▶	106	García Gómez, Luis Antonio	1959-03-25	11	Director General	1989-03-25	NULL	5600	NULL	0.12	0.07	NULL
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Ahora podemos obtener los clientes de dicho representante 106.

/* 2. Tras obtener el código de empleado que es 106, lo sustituimos en la búsqueda */

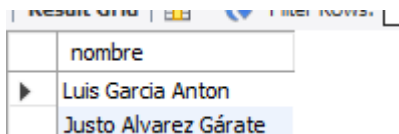
```
SELECT *
FROM Clientes
WHERE codRepresentante = '106';
```

Result Grid Filter Rows: Edit: Export/Imp				
	codCliente	nombre	codRepresentante	limiteCredito
▶	2101	Luis Garcia Anton	106	4000
	2118	Justo Alvarez Gárate	106	2000
*	NULL	NULL	NULL	NULL

Juntando las anteriores consultas

/* 3. Nos queda juntar los resultados anteriores en este caso se trata de una subconsulta en la CLÁUSULA WHERE */

```
SELECT nombre
FROM Clientes
WHERE codRepresentante =
      (SELECT codEmpleado
       FROM Empleados
       WHERE nombre = 'García Gómez, Luis Antonio');
```



nombre
Luis Garcia Anton
Justo Alvarez Gárate

Ejemplo: listamos el número y nombre de los empleados cuya fecha de contrato sea igual a la primera fecha de todos los pedidos de la empresa.

```
SELECT CodEmpleado, Nombre
FROM Empleados
WHERE Contrato = (SELECT MIN(FechaPedido) FROM Pedidos);
```

En una cláusula **WHERE** / **HAVING** tenemos siempre una condición y la subconsulta actúa de operando dentro de esa condición.

En el ejemplo anterior se compara contrato con el resultado de la subconsulta. Hasta ahora las condiciones estudiadas tenían como operandos valores simples (el valor contenido en una columna de una fila de la tabla, el resultado de una operación aritmética) ahora la subconsulta puede devolver una columna entera por lo que es necesario definir otro tipo de **condiciones especiales** para cuando se utilizan con subconsultas.

Esquema de subconsultas en la cláusula WHERE

- 1º) Restringimos los resultados que deseamos que aparezcan.
- 2º) Seleccionamos los datos que mostraremos para dicha selección.
- 3º) encadenamos los dos resultados.

Ejercicio: Listar los clientes cuyo agente asociado tiene como categoría Representante.

```
SELECT nombre
FROM Clientes
WHERE codRepresentante IN
      (SELECT codEmpleado
       FROM Empleados
       WHERE categoria = 'Representante');
```

Condiciones de selección con subconsultas

Las **condiciones de selección** son las condiciones que pueden aparecer en la cláusula **WHERE** o **HAVING**. La mayoría se han visto en el tema 2 pero ahora incluiremos las condiciones que utilizan una subconsulta como operando.

Las subconsultas pueden devolver diferentes cantidades de información:

- Una subconsulta escalar devuelve un valor simple.
- Una subconsulta de columna devuelve una única columna de uno o más valores.
- Una subconsulta de fila devuelve una fila única de uno o más valores (columnas).
- Una subconsulta de tabla devuelve una tabla de una o más filas de una o más columnas.

En MySQL tenemos cuatro nuevas condiciones que se utilizarán según cual sea el valor devuelto por la subconsulta:

- el **test de comparación con subconsulta**
- el **test de comparación cuantificada**
- el **test de pertenencia a un conjunto**
- el **test de existencia**

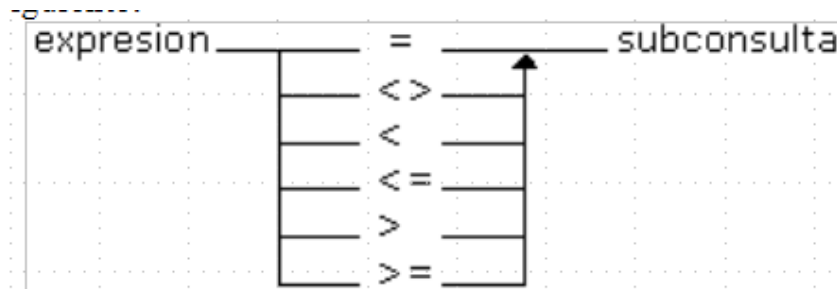
En todos los tests estudiados a continuación *expresión* puede ser cualquier nombre de columna de la consulta principal o una expresión válida.

El test de comparación con subconsulta.

Es el **equivalente al test de comparación simple**. Se utiliza para comparar un valor de la fila que se está examinado con un único valor producido por la subconsulta. La subconsulta debe devolver una **única columna**, sino se produce un error. (**subconsulta escalar**).

Si la subconsulta no produce **ninguna fila** o devuelve el valor **nulo**, el test devuelve el **valor nulo**, si la subconsulta produce **varias filas**, SQL devuelve una **condición de error**.

La sintaxis es la siguiente:



Lista las oficinas cuyo objetivo sea superior al 75% de la suma de los salarios de sus empleados.

```
SELECT CodOficina, Nombre
FROM Oficinas
WHERE Objetivo > (SELECT SUM(Sueldo)*0.75
FROM Empleados WHERE Empleados.Oficina = Oficinas.CodOficina);
```

Ejemplo: Seleccionar que empleado fue contratado primero.

```
SELECT * FROM Empleados  
Contrato = MIN(Contrato);
```

Da un **error**: Uso invalido de función de grupo. Porque la cláusula WHERE indica que registros debemos seleccionar, pero el valor MIN(fecha_nacimiento) no se conoce hasta después de que los registros hayan sido seleccionados.

Con una subconsulta:

```
SELECT * FROM Empleados WHERE  
fecContrato = (SELECT MIN(fecContrato) FROM Empleados);
```

Ejemplo: Seleccionar los pedidos superiores a la media de los pedidos del representante 106.

```
SELECT CodPedido, FechaPedido, SUM(cantidad * precioVenta),  
CodRepresentante FROM Pedidos JOIN LineasPedido USING(CodPedido)  
WHERE CodRepresentante = 106  
GROUP BY CodPedido  
HAVING SUM(cantidad * precioVenta) >  
(SELECT AVG(cantidad * precioVenta)  
FROM LineasPedido WHERE CodRepresentante = 106  
GROUP BY codPedido);
```

Si una consulta devuelve una fila, se puede utilizar un constructor de fila para comparar un conjunto de valores con el resultado de la subconsulta.

Ejemplo: Queremos conocer los empleados que tengan la misma categoría y edad que Viguer Ojanguren, Antonio.

```
SELECT Nombre, Categoria, FecNacimiento FROM Empleados  
WHERE (Categoria, FecNacimiento)#constructor de fila  
= (SELECT Categoria, FecNacimiento FROM Empleados  
WHERE Nombre = 'Viguer Ojanguren, Antonio');
```

También se puede utilizar ROW como constructor de fila.

```
SELECT Nombre, Categoria, FecNacimiento FROM Empleados  
WHERE (Categoria, FecNacimiento)#constructor de fila  
= (SELECT Categoria, FecNacimiento FROM Empleados  
WHERE Nombre = 'Viguer Ojanguren, Antonio');
```

En este caso la subconsulta devuelve una única columna y una única fila (es una consulta de resumen sin **GROUP BY**)

El test de comparación cuantificada.

Este test es una extensión del test de comparación y del test de conjunto. **Compara el valor** de la expresión **con cada uno de los valores** producidos por la subconsulta. La subconsulta debe devolver una **única columna** sino se produce un error. (subconsulta de fila).

Tenemos el test **ANY** o **SOME** (*algún, alguno* en inglés) y el test **ALL** (*todos* en inglés). ANY y SOME son sinónimos.

- El test **ANY**.

La subconsulta debe devolver una única columna sino se produce un error. Se evalúa la comparación con cada valor devuelto por la subconsulta.

Si **alguna de las comparaciones individuales produce el resultado verdadero**, el test **ANY** devuelve el resultado verdadero.

Si la subconsulta no devuelve **ningún valor**, el test **ANY** devuelve **falso**.

Si el test de comparación es **falso** para **todos los valores** de la columna, **ANY** devuelve **falso**.

Si el test de comparación **no es verdadero** para ningún valor de la columna, **y es nulo** para al menos alguno de los valores, **ANY** devuelve **nulo**.

Ejemplo: Lista las oficinas cuyo objetivo sea superior a alguna de las sumas de los sueldos de sus empleados agrupados por oficina.

```
SELECT CodOficina, Ciudad
FROM Oficinas
WHERE Objetivo > ANY
(SELECT SUM(Sueldo) FROM Empleados GROUP BY Oficina);
```

En este caso la subconsulta devuelve una única columna con las sumas de las cuotas de los empleados de cada oficina.

- El test **ALL**.

La subconsulta debe devolver una única columna sino se produce un error. Se evalúa la comparación con cada valor devuelto por la subconsulta. Si **todas** las comparaciones individuales, producen un resultado **verdadero**, el test devuelve el valor **verdadero**.

Si la subconsulta no devuelve **ningún valor** el test **ALL** devuelve el valor **verdadero**.

Si el test de comparación es **falso** para algún valor de la columna, el resultado es **falso**.

Si el test de comparación **no es falso** para ningún valor de la columna, pero es **nulo** para alguno de esos valores, el test **ALL** devuelve valor **nulo**.

Ejemplo: En este caso se listan las oficinas cuyo objetivo sea superior al 25% de todas las sumas de los sueldos

```
SELECT CodOficina, Nombre
FROM Oficinas
WHERE Objetivo > ALL
(SELECT SUM(sueldo)*0.25 FROM Empleados GROUP BY Oficina);
```

- Test de pertenencia a conjunto (IN, NOT IN).

Examina si el **valor** de la expresión es uno de los valores **incluidos en la lista de valores producida por la subconsulta**.

La subconsulta debe generar una única columna y las filas que sean. Si la subconsulta no produce ninguna fila, el test da falso.

Tiene la siguiente sintaxis:

```
expresion — IN — subconsulta
```


Ejemplo: Con la subconsulta se obtiene la lista de los números de oficina de Euzkadi este y la consulta principal obtiene los empleados cuyo número de oficina sea uno de los números de oficina de Euzkadi

```
SELECT CodEmpleado, Nombre, Oficina
FROM Empleados
WHERE Oficina IN
```

```
(SELECT Oficina FROM Oficinas WHERE Region = 'Euzkadi');
```

Por lo tanto lista los empleados de las oficinas de Euzkadi.

El test de existencia **EXISTS, NOT EXISTS**.

Examina si la subconsulta produce alguna fila de resultados.

Si la subconsulta contiene filas, el test adopta el valor **verdadero**, si la subconsulta no contiene ninguna fila, el test toma el valor falso, nunca puede tomar el valor nulo.

Con este test la subconsulta **puede tener varias columnas**, ya que el test se fija no en los valores devueltos sino en si hay o no fila en la tabla resultado de la subconsulta.

Cuando se utiliza el test de existencia en la mayoría de los casos habrá que utilizar una referencia externa. Si no se utiliza una referencia externa la subconsulta devuelta siempre será la misma para todas las filas de la consulta principal y en este caso se seleccionan todas las filas de la consulta principal (si la subconsulta genera filas) o ninguna (si la subconsulta no devuelve ninguna fila)

Ejemplo: Con la subconsulta se obtiene la lista de los números de oficina de Euzkadi este y la consulta principal obtiene los empleados cuyo número de oficina sea uno de los números de oficina de Euzkadi

```
SELECT CodEmpleado, Nombre, Oficina
FROM Empleados
WHERE EXISTS (SELECT * FROM Oficinas WHERE Region = 'Euzkadi' AND
Empleados.Oficina = Oficinas.CodOficina);
```

Este ejemplo obtiene lo mismo que el ejemplo del test **IN**.

Observa que delante de **EXISTS** no va ningún nombre de columna.

En la subconsulta se pueden poner las columnas que queramos en la lista de selección (hemos utilizado el *). Hemos añadido una condición adicional al **WHERE**, la de la referencia externa para que la oficina que se compare sea la oficina del empleado.

NOTA. Cuando se trabaja con tablas muy voluminosas el test **EXISTS** suele dar mejor rendimiento que el test **IN**

Subconsultas correlacionadas

Las subconsultas pueden ser correlacionadas o sin correlacionar:

- Una subconsulta sin correlacionar no tiene referencias a los valores de la consulta externa. Y puede ejecutarse por sí misma como una sentencia independiente.

Ejemplo: Seleccionar los clientes que tienen algún pedido.

```
SELECT * FROM Clientes
```

WHERE CodCliente IN (SELECT CodCliente FROM Pedidos);

- Una subconsulta correlacionada contiene referencias a los valores de la consulta externa, es decir, depende de ella. Por ello la subconsulta no se puede ejecutar independientemente.

**SELECT i2 FROM t2 WHERE i2 IN
(SELECT i1 FROM t1 WHERE i1 = i2);**

Las subconsultas correlacionadas se utilizan como subconsultas EXISTS y NOT EXISTS que son útiles para encontrar registros en una tabla que coinciden o no coinciden con los registros de otra.

Las subconsultas correlacionadas trabajan pasando valores desde la consulta externa a la subconsulta para ver si coinciden con las condiciones especificadas en la subconsulta. Por ello es necesario calificar los datos.

Ejemplo: Seleccionar los productos que se han vendido por lo vemos una vez

**SELECT IdFabricante, IdProducto, Descripcion FROM Productos
WHERE EXISTS (SELECT * FROM LineasPedido WHERE IdFabricante =
Fabricante AND IdProducto = Producto);**

EXISTS identifica coincidencias entre tablas, es decir, los valores que están presentes en ambas tablas.

NOT EXISTS identifica no coincidencias, valores de una tabla que no están presentes en ambas tablas.

Ejemplo: Seleccionar los clientes que no tienen ningún pedido.

**SELECT CodCliente, Nombre FROM Clientes
WHERE NOT EXISTS (SELECT * FROM Pedidos WHERE
Clientes.CodCliente = Pedidos.CodCliente);**

Subconsultas como Uniones

Dos consultas están correlacionadas si la ejecución de una depende de la otra consulta.

Las subconsultas pueden afectar al rendimiento de una base de datos, es por ello que debe de considerarse la posibilidad de reformularla mediante **UNIONES (joins)**.

Ejemplos:

- Reescribir subconsultas que seleccionan valores coincidentes.

Ejemplo: Seleccionar los empleados de Galicia. Con subconsultas

**SELECT * FROM Empleados
WHERE Oficina IN
(SELECT CodOficina FROM Oficinas WHERE
Region = 'Galicia');**

La misma consulta mediante una unión sencilla

**SELECT Em.* FROM Empleados AS Em, Oficinas AS Of
WHERE Em.Oficina = Of.CodOficina AND Region = 'Galicia';**

La misma consulta mediante INNER JOIN

```
SELECT Em.* FROM Empleados AS Em INNER JOIN Oficinas AS Of ON  
Em.Oficina = Of.CodOficina WHERE Region = 'Galicia';
```

Ejemplo: Seleccionar los pedidos de los clientes cuyo límite de crédito sea superior a 3000 €. Con subconsultas

```
SELECT * FROM Pedidos  
WHERE CodCliente IN  
(SELECT CodCliente FROM Clientes WHERE LimiteCredito > 3000);
```

La misma consulta mediante una unión sencilla

```
SELECT * FROM Pedidos P, Clientes C  
WHERE P.CodCliente = C.CodCliente AND LimiteCredito > 3000;
```

La misma consulta mediante INNER JOIN

```
SELECT * FROM Pedidos INNER JOIN Clientes USING(CodCliente)  
WHERE LimiteCredito > 3000;
```

- Reescribir **subconsultas que seleccionan valores no coincidentes**.

Son consultas que buscan valores de una tabla que no están presentes en la otra tabla. (No están da una pista sobre el tipo de consulta).

Ejemplo: Seleccionar los productos que no se hayan vendido. Con subconsultas

```
SELECT * FROM Productos  
WHERE NOT EXISTS (SELECT * FROM LineasPedido WHERE IdFabricante =  
Fabricante AND Idproducto = Producto);
```

Estas consultas se pueden reescribir con LEFT JOIN o RIGHT JOIN.

```
SELECT * FROM Productos LEFT JOIN LineasPedido ON IdFabricante =  
Fabricante AND Idproducto = Producto WHERE Codpedido IS NULL;
```