



UD3

CONSULTA Y MODIFICACIÓN DE BASES DE DATOS

MP_0484
Bases de Datos

3.4 Consultas
avanzadas

Introducción

En este punto vamos a trabajar con consultas avanzadas. La sentencia SELECT tiene muchas opciones y apenas comenzamos a explorar sus posibilidades. Uno de los objetivos de una BBDD es almacenar datos y ofrecer al usuario acceso a ellos de forma específica, por lo que tener un buen control de los tipos de consultas que se pueden llegar a realizar es parte del éxito de esa BD.

Si bien SELECT será siempre la base para realizar dichas consultas, interpretar lo que se quiere obtener de la BBDD, asociarlo al tipo de consulta que necesitamos y usar el formato adecuado es el éxito de dicha consulta. Practicar es la forma de aprendizaje.

Consultas en varias tablas

En las consultas que hemos visto hasta el momento hemos trabajado con los resultados de una sola tabla, es decir, los datos que se obtenían pertenecían solo a una tabla. SQL nos ofrece la posibilidad de juntar varias tablas en el resultado final de una consulta de manera que la información asociada a través de las claves foráneas pueda mostrarse en una sola tabla. A la hora de trabajar con más de una tabla, tenemos varias situaciones:

- Que necesitemos una subconsulta para comprobar resultados, como parte de una condición. Por ejemplo, mostrar los datos de los empleados cuyo departamento es el que tiene mayor sueldo.
- Que unamos dos o más tablas a través de sus claves foráneas y mostremos datos de más de una de ellas. Por ejemplo, mostrar datos de empleados y del departamento.
- Que realicemos operaciones de álgebra relacional (JOIN) en su variante interna, externa. Trabajaremos con nuevas BBDD para realizar este tipo de consultas.

Subconsultas anidadas

Una subconsulta anidada se define como una sentencia de tipo SELECT-FROM-WHERE que está dentro de otra consulta. Normalmente, las consultas anidadas se utilizan para comprobar pertenencia a conjuntos, así como para cardinalidades y para realizar comparaciones. La principal característica de una subconsulta de SQL es que permite utilizar los resultados de una consulta como parte de otra.

Pertenencia a grupos

La cláusula de pertenencia al conjunto de la subconsulta (IN) es una forma modificada de la pertenencia a conjunto simple. IN compara un único valor de datos con una columna de valores producida por una subconsulta y devuelve un resultado TRUE si el valor coincide con uno de los valores de la columna. Se puede usar también negando la pertenencia (empleando NOT IN).

En el ejemplo propuesto en la figura 1, buscamos mostrar todas las personas (su código de cliente) las cuales compraron el juego después de haber hecho una reserva de dicho juego.

```
SELECT distinct CodCliente
from venta
where CodCliente In (Select CodCliente
                     From reserva)
and CodProducto In (Select CodProducto
                   From reserva);
```

	CodCliente
▶	C01
	C02

Figura 1. Ejemplo de consulta anidada

De la misma forma, podemos mostrar los productos que no se han reservado, pero sí se han comprado (figura 2).

```
Select NombrePro, CodProducto
From producto
Where CodProducto Not In (Select CodProducto
                          from reserva)
and CodProducto In (Select CodProducto
                    from venta);
```

	NombrePro	CodProducto
▶	Call of Duty	P02
*	NULL	NULL

Figura 2. Ejemplo de consulta anidada

Consultas multitable

Cuando necesitamos datos de más de una tabla, debemos tener en cuenta que dichas tablas deben estar relacionadas directamente, a través de las claves foráneas, o indirectamente, a través de otras tablas intermedias con sus claves foráneas. Es muy importante haber realizado un buen diagrama relacional con todas sus relaciones entre tablas bien definidas.

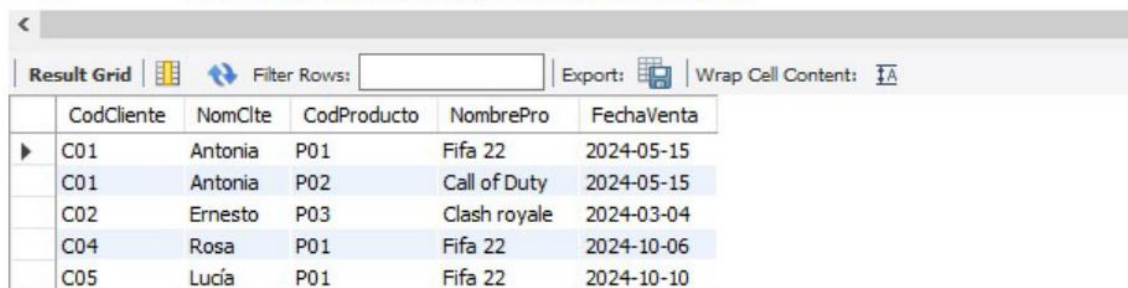
```
SELECT columna1 [, columna2, columna3 ...]
FROM tabla1, tabla2 [, tabla3 ...]
WHERE tabla1.columnaA=tabla2.columnaA
[AND tabla2.columnaB=tabla3.columnaB]
```

Por ejemplo, podemos mostrar información de los clientes, las compras realizadas por los mismos y los productos que han comprado. En este caso se han usado tres tablas: cliente, venta y producto; por ello, ha habido que realizar dos asociaciones de claves foráneas (figura 3).

```

26 • select Cliente.CodCliente, NomClte, venta.CodProducto, NombrePro, FechaVenta
27   From cliente, venta, producto
28   where cliente.CodCliente=venta.CodCliente
29         AND venta.CodProducto=producto.CodProducto;

```



	CodCliente	NomClte	CodProducto	NombrePro	FechaVenta
▶	C01	Antonia	P01	Fifa 22	2024-05-15
	C01	Antonia	P02	Call of Duty	2024-05-15
	C02	Ernesto	P03	Clash royale	2024-03-04
	C04	Rosa	P01	Fifa 22	2024-10-06
	C05	Lucía	P01	Fifa 22	2024-10-10

Figura 3. Consulta multitabla

Tal y como podemos ver en el resultado, es posible mostrar en una consulta resultados que provienen de tablas diferentes. La clave pasa por identificar correctamente las relaciones entre las tablas (como se puede apreciar, en el WHERE se denotan dichas relaciones) y de nuevo, haber trabajado adecuadamente el diagrama relacional.

En las condiciones del WHERE, además de los campos a relacionar, se pueden añadir más condiciones y de esta manera filtrar más el resultado. A continuación, se detallan algunas consideraciones a tener en cuenta al realizar este tipo de consultas:

- Es posible unir tantas tablas como deseemos.
- En la cláusula SELECT se pueden citar columnas de todas las tablas.
- Si hay columnas con el mismo nombre, se deben identificar poniendo nombreTabla.nombreCampo
- Si el nombre de una columna existe solo en una tabla, no será necesario indicar el nombre de la tabla.
- En el WHERE el orden de la asociación de claves foráneas no es representativo, si se omite esto, el resultado será el producto cartesiano (cada fila de una tabla se asocia a una fila de la otra tabla).

Composición de tablas JOIN

Cuando se usan dos tablas relacionadas, el tipo de composición que se crea afecta a las filas resultado, esto es lo que se conoce como JOIN. Hay dos tipos de composiciones/combinaciones:

- Internas: [INNER] JOIN.
- Externas: [OUTER] JOIN.
 - LEFT [OUTER] JOIN.
 - RIGHT [OUTER] JOIN.
 - FULL [OUTER] JOIN.

[INNER] JOIN

Es el tipo más común, devuelve las filas que son comunes en las dos tablas en base a la relación de claves foráneas. Es el tipo de combinación por defecto (figura 4).

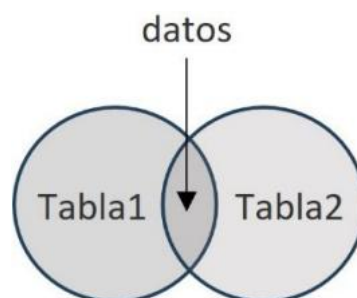


Figura 4. Representación gráfica del INNER JOIN

```
SELECT tabla1.campo1[, tabla.campo1, ...]
```

```
FROM Tabla1 INNER JOIN tabla2
```

```
ON tabla1.campo=tabla2.campo
```

En el ejemplo de la figura 5, podemos ver como recopilamos en una sola consulta los datos de los clientes que han realizado alguna compra y los detalles de dichas compras (en este caso, el producto y la tienda donde se ha realizado la compra).

Esto quiere decir que no se mostrarán los detalles de aquellos clientes que no han realizado ninguna compra. De la misma forma, si alguna compra no estuviese vinculada a ningún cliente (lo cual en este caso es poco probable) tampoco aparecería en los resultados. Es decir, tal y como hemos visto en la representación gráfica anterior, esta consulta sólo devuelve aquellas filas que poseen una relación entre las tablas implicadas.

```

53 • select cliente.CodCliente, NomClte, FechaVenta, CodProducto, tienda
54   from cliente INNER JOIN venta
55   ON cliente.CodCliente=venta.CodCliente;

```

	CodCliente	NomClte	FechaVenta	CodProducto	tienda
▶	C01	Antonia	2024-05-15	P01	MADRID
	C01	Antonia	2024-05-15	P02	MADRID
	C02	Ernesto	2024-03-04	P03	BILBAO
	C05	Lucía	2024-10-10	P01	ONLINE
	C05	Lucía	2024-10-05	P02	ONLINE
	C04	Rosa	2024-10-06	P01	BILBAO

Figura 5. Ejemplo de implementación de INNER JOIN

[OUTER] JOIN

Este tipo de combinación devuelve todas las filas de una de las partes, izquierda o derecha, según se especifique, y solo aquellas que están relacionadas de la otra tabla; los campos que tienen ningún valor los pone a NULL (figura 6).



Figura 6. Representación gráfica de los LEFT y RIGHT JOIN

```

SELECT tabla1.campo1[, tabla.campo1, ...]
FROM Tabla1 LEFT|RIGHT|FULL JOIN tabla2
ON tabla1.campo=tabla2.campo

```

Volviendo al ejemplo presentado en el punto anterior, si nos fijamos en este resultado (figura 7) tenemos siete filas de respuesta, mientras que cuando usamos el INNER JOIN había seis, en este caso aparece el Cliente C003, María, con FechaVenta a NULL, eso quiere decir que María no ha comprado nada todavía.

Es decir, el emplear el LEFT JOIN en nuestra consulta nos va a devolver absolutamente todas las filas de la primera tabla que consignemos (de ahí lo de LEFT, ya que en occidente acostumbramos a leer de izquierda a derecha, por ello la primera tabla a considerar será la de la izquierda) pero sólo aquellas filas de la segunda tabla (la de la derecha) que estén relacionadas.


```

53 • select cliente.CodCliente, NomClte, FechaVenta, venta.CodProducto, tienda
54 from cliente LEFT JOIN venta
55 ON cliente.CodCliente=venta.CodCliente;

```

Result Grid Filter Rows: Export: Wrap Cell Content:					
	CodCliente	NomClte	FechaVenta	CodProducto	tienda
▶	C01	Antonia	2024-05-15	P01	MADRID
	C01	Antonia	2024-05-15	P02	MADRID
	C02	Ernesto	2024-03-04	P03	BILBAO
	C03	María	NULL	NULL	NULL
	C04	Rosa	2024-10-06	P01	BILBAO
	C05	Lucía	2024-10-10	P01	ONLINE
	C05	Lucía	2024-10-05	P02	ONLINE

Figura 7. Ejemplo de uso de LEFT JOIN

Sucede lo opuesto si optamos por el RIGHT JOIN. En el ejemplo anterior hemos usado LEFT JOIN, si hubiéramos cambiado el orden, el resultado sería el mismo que al usar RIGHT JOIN (figura 8). De hecho, el uso del RIGHT JOIN implica quedarnos con todas las filas de la segunda tabla (por el mismo motivo que antes, la que se encuentra a la derecha) y sólo aquellas filas de la primera tabla que están relacionadas (la tabla de la izquierda).

```

53 • select cliente.CodCliente, NomClte, FechaVenta, venta.CodProducto, tienda
54 from venta RIGHT JOIN cliente
55 ON cliente.CodCliente=venta.CodCliente;

```

Result Grid Filter Rows: Export: Wrap Cell Content:					
	CodCliente	NomClte	FechaVenta	CodProducto	tienda
▶	C01	Antonia	2024-05-15	P01	MADRID
	C01	Antonia	2024-05-15	P02	MADRID
	C02	Ernesto	2024-03-04	P03	BILBAO
	C03	María	NULL	NULL	NULL
	C04	Rosa	2024-10-06	P01	BILBAO
	C05	Lucía	2024-10-10	P01	ONLINE
	C05	Lucía	2024-10-05	P02	ONLINE

Figura 8. Ejemplo de uso de RIGHT JOIN

Por último, con el parámetro FULL OUTER JOIN, se devuelven todas las filas de la tabla izquierda y de la derecha; aquellas filas en las que no haya relación se ponen nulos (figura 9).

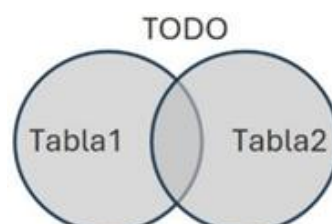


Figura 9. Representación gráfica del FULL OUTER JOIN

