

NifCli	NomCli	DirCli	SdoCli
15073277J	Olmedo	Obispo 29, Bullas	2500

Aunque union es parte de ANSI/SQL muchos productos comerciales aún no la soportan. Por otro lado minus se ha sustituido en SQL2 por except e intersect se ha incluido como tal.

2.2.4. ORDENACIÓN DE LA PRESENTACIÓN DE TUPLAS

SQL ofrece al usuario cierto control sobre el orden en el que se van a presentar las tuplas en una relación. La cláusula order by hace que las tuplas en el resultado de una consulta se presenten en un orden determinado. Para listar en orden alfabético todos los clientes, escribimos

```
select nifcli, nomcli, dircli, sdocli from cliente
order by nomcli
```

Por omisión, SQL lista los elementos en orden ascendente. Para especificar el tipo de ordenación, podemos indicar desc para orden descendente o asc para orden ascendente. Además, el orden puede realizarse sobre múltiples atributos. Supóngase que queremos listar la relación lin_fra completa en orden descendente de cantidad. Si varias líneas de factura tienen la misma cantidad, los ordenamos en orden ascendente por número de factura. Expresamos esto en SQL como sigue:

```
select numfac, numlin, codart, uniart, implin from lin_fra
order by implin desc, 1 asc
```

obsérvese que es indistinto indicar el atributo de ordenación o su número de orden en la cláusula select.

Para satisfacer una solicitud de order by, SQL debe realizar una ordenación. Puesto que ordenar un gran número de tuplas puede ser costoso, lo más conveniente es ordenar sólo cuando sea necesario. Asimismo, a la hora de efectuar el diseño físico de una BD cuando se prevea utilizar un orden determinado en alguna consulta muy solicitada, lo usual es crear un índice que utilice como clave los atributos del order by.

2.2.5. PRODUCTOS

Para realizar un producto cartesiano en SQL se ha de utilizar la cláusula from indicando a continuación las tablas que se usarán en el producto, indicando o no la cláusula cross join (que es en cierto modo redundante).

Por ejemplo para llevar a cabo la operación de álgebra relacional $r = \text{CLIENTE} \times \text{ARTICULO}$ podemos hacerlo de las dos siguientes formas :

```
select * from cliente, articulo
```

o bien

```
select * from cliente cross join articulo
```

Por otro lado, originalmente SQL no tenía una representación directa de la operación del producto natural. Sin embargo, puesto que el producto natural se define en términos de un producto cartesiano, una selección y una proyección, era relativamente sencillo escribir una expresión en SQL ANSI86 para el producto natural.

Recuérdese la expresión del álgebra relacional que escribimos para obtener los datos de los clientes de cada factura.

$$\sigma_{(\text{Cliente.NifCli}=\text{Factura.NifCli})}(\text{FACTURA} \times \text{CLIENTE})$$

FACTURA \bowtie CLIENTE

En SQL, esto se puede escribir así :

```
select * from factura, cliente
where cliente.nifcli = factura.nifcli
```

el resultado será :

NumFac	FecFac	ImpFac	NifCli	NomCli	DirCli	SdoCli
323/95	31/01/95	14500	21654564T	Lison	Tejera 38, Bolnuevo	163000
342/95	31/01/95	19000	15073277J	Olmedo	Obispo 29, Bullas	2500
355/95	15/02/95	66321	653344L	Buendía	Umbrías 28, La Paca	125466
401/95	28/02/95	11000	653344L	Buendía	Umbrías 28, La Paca	125466

Obsérvese que al igual que el álgebra relacional, SQL usa la notación nombre-relación.nombre-atributo (lo que algunos autores llaman nombres de columnas cualificados), para evitar ambigüedad en los casos en los que un atributo aparece en el esquema de más de una relación.

Ampliamos la consulta anterior y consideremos un caso algo más complicado en el que se requiere también que todos los clientes tengan un saldo determinado: "Encontrar el nombre y la dirección de todos los clientes que tienen una factura de más de 20000 pts." Para escribir esta consulta, necesitaremos declarar dos límites en la cláusula where, conectados por el conector lógico and.

```
select distinct nomcli, dircli from cliente, factura
where cliente.nifcli = factura.nifcli and impfac>20000
```

En este caso la expresión en álgebra relacional sería :

$$\Pi_{(nomcli, dircli)}(\sigma_{((Cliente.NifCli=Factura.NifCli) \wedge (ImpFac>20000))}(FACTURA \times CLIENTE))$$

El estándar SQL2 permite usar la sentencia natural join para expresar consultas como la anterior del siguiente modo:

```
select distinct nomcli, dircli from cliente natural join factura
where impfac > 20000
```

SQL2 permite especificar condiciones en los natural join. Estas condiciones, adicionales a las que impone el propio natural join, se especifican usando la cláusula ON. En los inner join, indicar una cláusula on es exactamente igual que añadirla a una cláusula where, por ejemplo para construir una sentencia equivalente a la anterior usaremos:

```
select distinct nomcli, dircli from cliente natural join factura
on impfac > 20000
```

2.2.6. SUBCONSULTAS

Una subconsulta es una sentencia select que aparece dentro de un predicado y donde el dominio del resultado producido tiene que coincidir con el que se espera en función de los atributos que se incluyen en el predicado y del tipo de predicado en sí. Aunque en el estudio usual del SQL se suelen dejar las subconsultas para el capítulo "temas avanzados", esta es, a nuestro juicio, una actitud injustificada puesto que su estudio no tiene mayor dificultad.

Uno de los usos de las subconsultas es simplificar expresiones complejas de acceso a datos o sencillamente permitir una codificación alternativa de sentencias. Por ejemplo el producto

```
select distinct nomcli, dircli from cliente, factura
where cliente.nifcli = factura.nifcli and impfac > 20000
```

se puede expresar como una subconsulta

```
select distinct nomcli, dircli from cliente where nifcli in
(select nifcli from factura where impfac > 20000)
```

donde también usamos el test de pertenencia a un conjunto in estudiado más adelante. Como vemos, el predicado es nifcli in (subselect), por lo que el subselect ha de producir un resultado con el mismo dominio que nifcli y que se pueda usar en el test de inclusión de conjuntos (es decir el resultado ha de ser un conjunto de valores de dominio igual al de nifcli).

2.2.7. VARIABLES DE TUPLA

SQL toma prestada la notación de variables de tupla del cálculo relacional de tuplas. Una variable de tupla en SQL debe estar asociada con una relación determinada. Las variables de tupla se definen en la cláusula from. Para ilustrarlo, escribiremos la consulta "Encontrar el nif y el nombre de todos los clientes que tienen alguna factura", así:

```
select distinct C.nifcli, nomcli from cliente C, factura F
where C.nifcli = F.nifcli
```

Nótese que una variable de tupla se define en la cláusula from colocándola después del nombre de la relación con la cual está asociada, separada por uno o más espacios.

En consultas que contienen subconsultas, se aplica un ámbito a las variables de tupla. En una subconsulta, está permitido usar sólo variables de tupla definidas en la misma subconsulta o en cualquier consulta que contenga a la subconsulta. Si una variable de tupla está definida tanto localmente en una subconsulta como globalmente en una