MÓDULO 1: ORGANIZACIÓN Y ESTRUCTURA DE LA INFORMACIÓN

Tema 3: Interrogación y explotación de datos

Leire Aldaz, Begoña Eguía y Leire Urcola

Índice del tema

INTRODUCCIÓN

El lenguaje de interrogación de datos SQL (*Structured Query Language*) está formado por un conjunto de órdenes que permiten la consulta y la puesta al día de los datos de una base de datos. La puesta al día engloba la inserción de nuevos datos, la modificación y la supresión de datos ya existentes. Sin embargo, nosotros, vamos a centrarnos únicamente en la consulta o selección de los datos.

La consulta de datos en SQL se ejecuta con el verbo SELECT que significa seleccionar o extraer datos.

La sintaxis global de la orden que permite la extracción de datos es la siguiente:

SELECT < lista de atributos > | ***
FROM < tabla/s >
[WHERE < condición >]
[GROUP BY < atributo/s >]
[HAVING < condición >]
[ORDER BY < atributo/s > ASC | DESC] ;

Todas las cláusulas que van entre corchetes son opcionales. Por tanto, las cláusulas SELECT y FROM son las únicas obligatorias y permiten especificar respectivamente la lista de columnas o atributos a extraer y la lista de las tablas a partir de las cuales serán extraídas. El resto, son opcionales y sirven para refinar nuestra selección.

CONSULTA SENCILLA EN SQL

La consulta sencilla se hace mediante las cláusulas obligatorias de la orden precedente.

SELECT < lista de atributos >

FROM < tabla/s >;

La lista de atributos permite especificar las columnas o atributos que se desean visualizar, y éstos deberán estar separados por comas.

Por ejemplo, para visualizar todos los datos de una tabla Clientes, se formulará la sentencia siguiente:

SELECT idcliente, nombre, dirección, ciudad, tel

FROM clientes;

Esta sentencia lista al conjunto de clientes de la tabla Clientes. El orden en el que se presentarán los atributos es el que se especifique en la lista.

Existe una manera de facilitar la formulación de la sentencia, reemplazando la lista de atributos por el carácter genérico "*". Este carácter copia en la lista, los diferentes atributos tomados de la definición de la tabla. Y también toma obligatoriamente el orden especificado en la creación de la misma. Así, la sentencia anterior se formulará como sigue:

SELECT *

FROM clientes;

Hasta ahora hemos visto como listar todos los atributos de una tabla. Sin embargo, se pueden igualmente listar una parte de ellos, escribiendo simplemente en la lista los nombres que deseemos visualizar en un orden específico.

Una consulta sencilla, sin cláusula WHERE permite extraer una parte vertical (proyección) de datos de la tabla. Si lo que se desea extraer es una parte horizontal (restricción) y una parte vertical es necesario entonces realizar una consulta calificada.

CONSULTA CALIFICADA EN SQL

La consulta calificada o cualificada se hace a partir de una consulta simple y una cláusula WHERE. Así, para visualizar únicamente los clientes de Madrid, formularemos la siguiente sentencia:

```
SELECT *
    FROM clientes
    WHERE ciudad = "Madrid";
```

La calificación se hace con la ayuda de la cláusula WHERE seguida de una o varias condiciones. Estas condiciones pueden estar relacionadas entre ellas por los operadores lógicos **AND**, **OR** y **NOT**.

Ejemplos:

1. Listar de la tabla artículos todos aquellos cuyo precio unitario sea superior a 150 y su cantidad de stock inferior o igual a 100.

```
SELECT *
    FROM articulos
    WHERE pu > 150 AND stock<=100;
```

2. Mostrar de la tabla de empleados el dni, nombre, fecha de ingreso y salario de los empleados que trabajen en los departamentos 112 y 122.

```
SELECT dni, nombre, fecha_in, salario
    FROM empleados
    WHERE numdept = 112 OR numdept = 122;
```

3. Presentar de la tabla de Pedidos_Clientes el número del pedido, la identificación del cliente y la fecha aquellos pedidos no registrados en el primer cuatrimestre de 2008.

```
SELECT numpedido, idcliente, fechapedido
    FROM Pedidos_Clientes
    WHERE NOT (fechaped >=  #1/1/2008# AND fechaped <= #30/4/2008# );
```

El predicado BETWEEN v1 AND v2

El predicado BETWEEN ….AND… se utiliza para seleccionar registros que se encuentren entre un intervalo cerrado de valores [v1,v2].

Por ejemplo, la consulta anterior también se podía haber formulado de la siguiente manera:

```
SELECT numpedido, idcliente, fechapedido
    FROM Pedidos_Clientes
    WHERE fechaped NOT BETWEEN #1/1/2008# AND #30/4/2008# ;
```

Nota:

En la cuadrícula QBE, en la fila de criterios, y en la columna en la que se especifique la fecha del pedido habría que especificarlo de la siguiente manera:

Criterios: **Negado Entre #1/1/2008# Y #30/4/2008#**

EL predicado In(lista de valores)

El predicado IN permite comparar el valor de la expresión situada a la izquierda de la palabra IN con la lista de valores comprendidos entre los paréntesis. La condición de búsqueda es satisfecha cuando la expresión de comparación está comprendida en la lista de valores.

Por ejemplo:

Listar todos los clientes de las ciudades siguientes: “PARIS”, “BARCELONA”, “MADRID”, así como todos los artículos cuyo precio unitario es uno de los siguientes: 150, 200 y 300.

```
SELECT *
        FROM Clientes
        WHERE ciudad In(“PARIS”, “BARCELONA”, “MADRID”);
SELECT *
        FROM articulos
        WHERE pu IN(150, 200, 300) ;
```

Nota: en ***QBE*** en la fila de criterios debajo de la columna ciudad para la tabla clientes y pu para la tabla artículos se especificaría lo siguiente:

Criterios: **In(“PARIS”;”BARCELONA”;”MADRID”)**

Esta dos sentencias pueden formularse utilizando el operador lógico O (OR):

```
SELECT *
    FROM clientes
    WHERE ciudad =" PARIS"
        OR ciudad = "BARCELONA"
        OR ciudad = " MADRID" ;
SELECT *
    FROM articulos
    WHERE pu = 150
        OR pu = 200
        OR pu = 300 ;
```

En ***QBE*** . Criterios: **"PARIS" ó "BARCELONA" ó "MADRID"**

150 ó 200 ó 300

El predicado LIKE

Permite realizar una comparación de semejanza entre el valor de un atributo y el de una cadena de caracteres, para ello se pueden utilizar los caracteres: "*" y "?".

El "*" sustituye a una secuencia de caracteres, mientras que la "?" sustituye a un único carácter.

Por ejemplo, presentar a los empleados cuyo dni comience por 34.

```
SELECT *
    FROM empleados
    WHERE dni LIKE "34*" ;
```

QBE: criterios: **Como "34*"**

FUNCIONES

Dentro de la multitud de funciones que lleva ACCESS incorporadas vamos a manejar las funciones de manejo de fechas y las funciones agregadas.

1. FUNCIONES DE FECHA:

Ahora( ), no lleva argumentos y presenta la fecha y hora actual del sistema.

Fecha( ), no lleva argumentos y presenta la fecha actual o fecha del día del sistema.

Año(arg), devuelve un número - año del argumento - que deberá ser de tipo fecha.

Mes(arg), devuelve un número - mes del argumento - que deberá ser de tipo fecha.

Día(arg), devuelve un número - día del argumento - que deberá ser de tipo fecha.

Operaciones con fechas:

a) A una expresión de fecha se le puede sumar/restar un número determinado de días, dando como resultado una nueva fecha.

<Fecha> + <número de días> => <Nueva fecha>

b) También se pueden restar dos expresiones de fecha, dando como resultado en este caso el número de días que hay entre las dos fechas.

<Fecha1> - <Fecha2> => <número de días>

Ejemplo:

Listar de la tabla Pedidos_Clientes aquellos pedidos con una antigüedad superior a 30 días.

```
SELECT *
        FROM Pedidos_Clientes
        WHERE Fecha_Pedido < Date( ) – 30 ;
```

O bien: `Date( ) – Fecha_Pedido > 30 ;`

Ejercicio:

Una empresa tiene un plan de jubilación para aquellos empleados con mas de 60 años. Estos empleados percibirán una paga extra de un sueldo completo por cada año que llevan trabajando para la empresa. Presentar los nombres de estos empleados junto a su sueldo extra.

2. FUNCIONES AGREGADAS:

Estas funciones actúan sobre los registros de una tabla.

Suma, realiza el sumatorio de los valores que toma un atributo o una expresión en todos o parte de los registros de una tabla

Promedio, realizar el promedio de los valores que toma un atributo o expresión en todos o parte de los registros de una tabla.

Cuenta, cuenta el número de registros.Puede llevar como argumentos * | [atributo].

Count(*) es considerablemente más rápido que Count(atributo) , y cuenta también campos nulos, mientra que éste último no.

Max, presenta el valor máximo que toma un atributo

Min, presenta el valor mínimo que toma un atributo.

Para utilizar estas funciones en Access, será necesario añadir a la cuadrícula QBE una línea de Totales. Esto se realiza en la ventana de diseño de la consulta pinchando en el icono $\sum$ de la barra de herramientas.

Algunos ejemplos:

(1) Presentar de la tabla de empleados, el valor del sueldo máximo, el mínimo, el total y la media de los sueldos y, el total de empleados tiene la tabla.

QBE:

Atributos:	Máximo: Sueldo	Mínimo: Suedo	Suma: Sueldo	Media: Sueldo	Nº Empleados: **Dni**
Tablas:	Empleados	Empleados	Empleados	Empleados	Empleados
Totales:	**Max**	**Min**	**Suma**	**Promedio**	**Cuenta**
Orden:					
Mostrar	*V*	*V*		*V*	*V*
Criterios:					

En lenguaje SQL:

```
SELECT  Max(Suedo) AS Máximo, Min(Sueldo) AS Mínimo, Sum(Sueldo) AS Suma,
        Avg(Sueldo) AS Media, Count(Dni) AS Nº mpleados
    FROM Empleados ;
```

(2) Presentar en pantalla el número de empleados que ganan más de 1.500 euros.

Atributos:	Nº Empleados: Dni	Suelod
Tablas:	Empleados	Empleados
Totales:	**Cuenta**	
Orden:		
Mostrar	*V*	
Criterios:		> 15000

```
SELECT Count(DNI) AS Nº Empleados
    FROM Empleados
    WHERE sueldo > 1500
```

(3) De la tabla de Artículos presentar en pantalla el valor total del stock de los mismos.

```
SELECT Sum(pu * stock) AS Valor_Stock
        FROM Articulos ;
```

Implementa las siguientes consultas:

(1) Para los departamentos 111 y 112, hallar la media de los años en servicio de sus empleados en el día de hoy, presentando junto a la media calculada, el día de hoy.

(2) Hallar cuántos empleados han ingresado en el año actual.

(3) Para los empleados que han ingresado en la empresa en los últimos cinco años, hallar la edad media en años cumplidos de la edad a la que han ingresado.

La Función: IIF(condición; acción_parte_Verdadera; acción_parte_Falsa)

La función IIF en inglés, o SiInm en castellano, se utiliza para evaluar una condición, y en función de que se cumpla o no se realizará una acción u otra. En un función SiInm se pueden anidar a su vez varias condiciones, es decir varias funciones SiInm. Por ejemplo:

En una campaña de ayuda familiar se ha decidido dar a los empleados una paga extra de 30 euros por hijo, a partir del tercero inclusive. Obtener para estos empleados: nombre y salario total que van a percibir incluyendo esta paga extra, y para el resto presentar únicamente su salario.

SELECT nombre, **IIF(numhijos>2, salario+30*(numhijos-2), salario)** AS salario

FROM empleados;

En QBE:

Atributos:	Nombre	Salario: **SiInm**(numhijos>2;alario+30*(numhijos-2); salario)
Tablas:	Empleados	Empleados

Ejercicio:

Obtener los nombres de los empleados del departamento 112, así como su sueldo total incluyendo la comisión en aquellos que la tengan, y presentar a su lado el literal “con comisión” o “sin comisión” en función de que tengan o no comisión.

AGRUPAMIENTO DE DATOS: GROUP BY

La clausula GROUP BY permite agrupar registros que tengan valores comunes. El atributo o atributos de agrupamiento que se especifiquen junto a esta clausula deben obligatoriamente figurar en la lista de selección de la clausula SELECT.

Utilizando esta cláusula se puede formular por ejemplo una sentencia que liste el número de clientes por ciudades:

SELECT **ciudad**, Count(idcliente) AS NºClientes

FROM clientes

GROUP BY ciudad ;

Atributos:	Ciudad	NºClientes: idcliente
Tablas:	Clientes	clientes
Totales:	**Agrupar por**	**Cuenta**

De la tabla Pedidos_Clientes mostrar cuántos pedidos encarga cada cliente:

SELECT idcliente, Count(NumPedido) AS NºPedidos

FROM Pedidos_Clientes

GROUP BY idcliente;

Atributos:	Idcliente	NºPedidos: NumPedido
Tablas:	Pedidos_Clientes	Pedidos_Clientes
Totales:	**Agrupar por**	**Cuenta**

Hallar el salario medio y la edad media en años para los empleados con igual comisión (es decir, para cada grupo de empleados con igual comisión).

SELECT comisión, Avg(Salario) AS salario_medio,
Avg(Year(Date( )) – Year(Fecha_Naci)) AS edad_media

FROM empleados

GROUP BY comision ;

Atributos:	Comisión	Salario_medio: Salario	Edad-media: Año(Fecha( )) - Año(Fecha_Naci)
Tablas:	Empleados	Empleados	Empleados
Totales:	**Agrupar por**	**Promedio**	**Promedio**

Ejercicio:

De la tabla de empleados, presentar el número de trabajadores, el salario máximo y el salario medio de cada departamento de la empresa.

CONDICIÓN DE AGRUPAMIENTO: HAVING

La clausula *Having* está normalmente relacionada con la cláusula GROUP BY, permitiendo efectuar una restricción sobre un grupo de registros que hayan sido previamente agrupados.

Ejemplo:

Presentar el número de empleados que la empresa ha contratado cada año, mostrando únicamente los contratados a partir de 2000.

SELECT **Year(Fecha_ingreso) AS Año**, Count(Dni) AS NºContratados
FROM empleados
GROUP BY **Year(Fecha_ingreso)**
HAVING Year(Fecha_ingreso) >= 2000 ;

Atributos:	Año: Año(Fecha_ingreso)	NºContratados: Dni
Tablas:	Empleados	Empleados
Totales:	**Agrupar por**	**Cuenta**
Mostrar:	*V*	*V*
Criterios:	>= 2000	

La cláusula HAVING además de utilizarse para especificar condiciones o restricciones sobre el agrupamiento de datos, también se emplea cuando hay que formar condiciones que incluyan funciones agregadas, es decir, condiciones con las funciones: Sum, Avg, Count, etc., tengan o no que ver con el agrupamiento.

Ejemplo:

Presentar de la tabla Pedidos_Clientes aquellos con más de tres pedidos.

```
SELECT idcliente, Count(NumPedido) AS Total_Pedidos
    FROM Pedidos_Clientes
    GROUP BY idcliente
    HAVING Count(NumPedido) > 3 ;
```

Nota: en este caso la condición no tiene que ver con el agrupamiento de datos, pero como se utiliza la función Count es necesario especificarla en la cláusula HAVING, ya que sólo ésta puede realizar las condiciones con las funciones agregadas.

Atributos:	Idcliente	Total_Pedidos: NumPedido
Tablas:	Pedidos_Clientes	Pedidos_Clientes
Totales:	**Agrupar por**	**Cuenta**
Mostrar**:**	*V*	*V*
Criterios:		**> 3**

Ejercicio:

De la tabla empleados, para los departamentos 111, 112 y 122 que tengan empleados de más de diez años de antigüedad y la media de hijos de esos empleados sea superior o igual a 2, presentar su retribución salarial con un incremento de un 15%

ORDENACIÓN DEL RESULTADO DE UNA CONSULTA: ORDER BY

La cláusula ORDER BY permite realizar operaciones de ordenación con los datos extraídos de una SELECT. Se pueden especificar uno o varios criterios de ordenación y en distinto orden, es decir, ascendente (por defecto) y/o descendente. En este último caso es necesario especificar detrás del atributo la opción DESC.

Algunos ejemplos sencillos:

1. Listar todos los clientes por orden de ciudad y nombre de cliente.

   SELECT *

   FROM clientes

   ORDER BY ciudad, nombre ;

2. Presentar el número de empleados que la empresa ha contratado cada año, pero mostrando únicamente los contratados a partir de 2000 en orden descendente.

   SELECT Year(Fecha_ingreso) AS Año, Count(Dni) AS NºContratados

   FROM empleados

   GROUP BY Year(Fecha_ingreso)

   HAVING Year(Fecha_ingreso) >= 2000

   ORDER BY Year(Fecha_ingreso) DESC ;

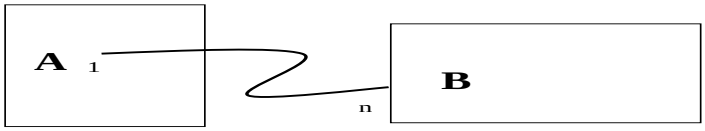
En QBE:

Atributos:	Año: Año(Fecha_ingreso)	NºContratados: Dni
Tablas:	Empleados	Empleados
Totales:	Agrupar por	Cuenta
Orden:	**Descendente**	
Mostrar:	*V*	*V*
Criterios:	>= 2000	

LAS RELACIONES ENTRE TABLAS

Los tipos de relaciones principales que permite un SGBD Relacional son:

Relación de uno a muchos, es el tipo más común de relación entre tablas. En este tipo de relación cada registro de la tabla principal (A) puede tener más de un registro asociado en la tabla subordinada (B), pero a cada registro de la tabla (B) sólo le corresponde un solo registro de la tabla principal (A).



Relación de uno a uno, a cada registro de la tabla (A) le corresponde un único registro de la tabla (B) y viceversa.

Para poder establecer relaciones entre tablas es necesario hacer coincidir la clave principal de la tabla padre con un campo coincidente en la tabla subordinada (clave extranjera). Cuando el tipo de relación es de uno a varios se puede exigir el cumplimiento de la propiedad integridad referencial.

La propiedad integridad referencial debe garantizar lo siguiente:

a) No puede haber registros en la tabla subordinada que no tengan su correspondiente "padre" en la tabla principal.
b) No se pueden eliminar registros de la tabla principal si tiene asociados registros en la tabla subordinada (ya que en este caso se violaría el cumplimiento de esta propiedad. En el caso de desear eliminar registros de la tabla principal y mantener la base de datos en un estado "legal", se puede optar por el borrado en cascada (propagar el borrado), o bien antes de eliminar los registros de la tabla principal se pueden eliminar primero los registros de las tablas subordinadas. Asimismo, se podría optar por anular (poner valor Nulo) el valor de la clave extranjera cuando se elimina el registro al que referencia en la tabla principal.

c) No se pueden cambiar el valor de la clave principal de la entidad padre si tiene registros asociados en una tabla subordinada. En este caso, para no violar el cumplimiento de la propiedad integridad referencial, se tienen las mismas alternativas que en el caso anterior (modificar en cascada, poner a Nulo la clave extranjera o impedir llevar a cabo tal acción).

Algunos ejemplos:

Listar los datos de aquellos clientes que hayan realizado algún pedido.

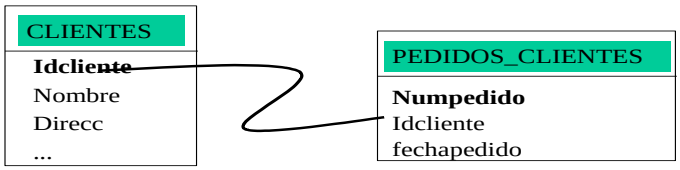
SELECT nombre, direcc, telf, fechapedido

FROM clientes, pedidos_clientes

WHERE **clientes.idcliente = pedidos_clientes.idcliente** ;

El resultado de esta consulta estará formado por las líneas que satisfagan la condición de composición.

En QBE, antes de rellenar la cuadrícula hay que representar gráficamente la relación.

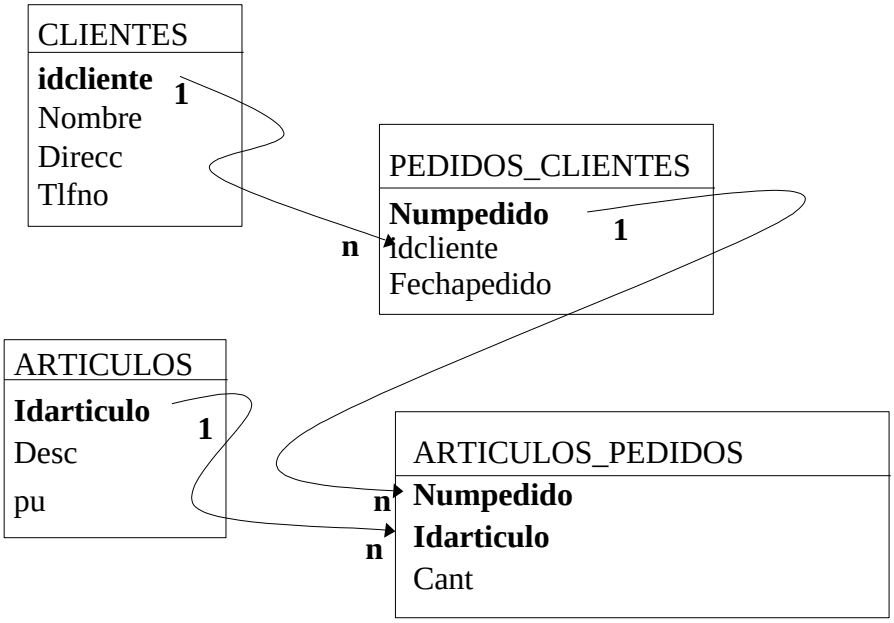


Atributos:	Nombre	Direcc	Tlfno	Fechapedido
Tablas:	Clientes	Clientes	Clientes	Pedidos_clientes
Orden:				
Mostrar:	*V*	*V*	*V*	*V*
Criterios:				
O:				

Listar el importe total de cada pedido.

```
SELECT numpedido, Sum([pu]*[cant]) AS importe
        FROM Articulos_pedidos, articulos
        WHERE Articulos_pedidos.idarticulo = articulos.idarticulo
        GROUP BY numpedido ;
```

El resultado de la consulta lo formarán los registros de ambas tablas que tengan el mismo valor en la identificación del artículo.



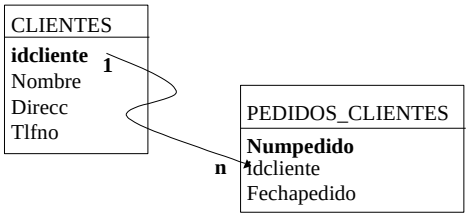
(Aunque en la representación gráfica están plasmadas las relaciones entre las cuatro tablas, para realizar ésta consulta sólo sería necesaria la relación entre "Articulos" y "Articulos_Pedidos").

Atributos	Numpedido	Importe: [pu]*[cant]	
Tablas:	Clientes	Clientes	
Totales	Agrupar por	Suma	
Orden:			
Mostrar:	*V*	*V*	
Criterios:			
O:			

Presentar en pantalla los datos de los clientes de Madrid que hayan realizado algún pedido entre el 12/01/2009 y la fecha de hoy.

```
SELECT nombre, direcc, telfno, fechapedido
    FROM clientes, pedidos_clientes
    WHERE clientes.idcliente = pedidos_clientes.idcliente
        AND fechapedido BETWEEN #12/01/2009# AND Date( )
        AND ciudad = "Madrid" ;
```

En ***QBE***:

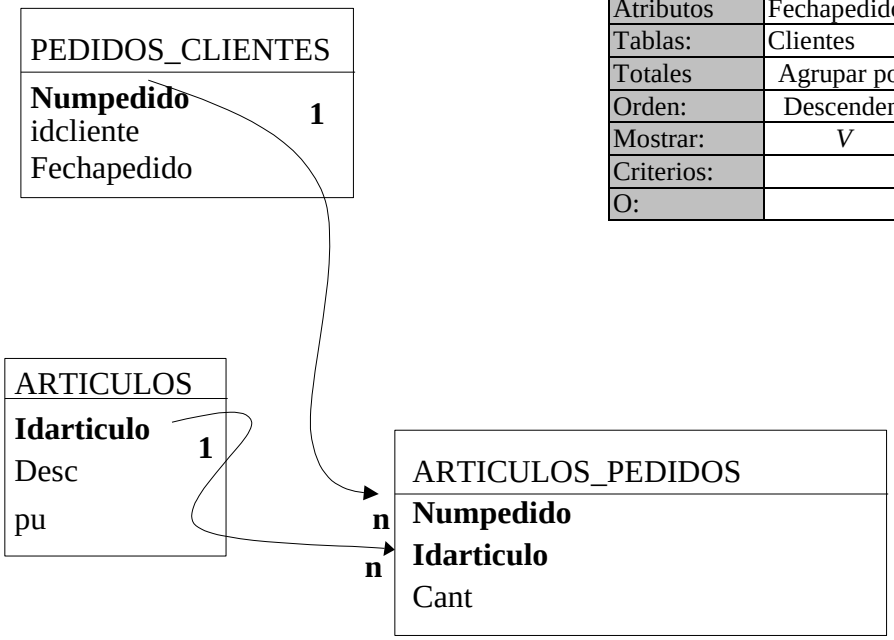


Atributos	Nombre	Direcc	Tlfno	Fechapedido
Tablas:	Clientes	Clientes	Clientes	Clientes
Totales				
Orden:				
Mostrar:	*V*	*V*	*V*	*V*
Criterios:				Entre #12/01/2009# y Fecha( )
O:				

Listar el importe total de pedidos por fecha de pedido, presentando las fechas en orden descendente (es decir, hay que sumar los importes de los pedidos realizados en cada una de las fechas).

```
SELECT fechapedido, Sum([pu]*[cant]) AS importe
    FROM pedidos_clientes, articulos_pedidos, articulos
    WHERE pedidos_clientes.numpedido = articulos_pedidos.numpedido
        AND articulos_pedidos.idarticulo = articulos.idarticulo
    GROUP BY fechapedido
    ORDER BY fechapedido DESC ;
```

En ***QBE*** :



Atributos	Fechapedido	Importe: [pu]*[cant]
Tablas:	Clientes	Clientes
Totales	Agrupar por	Suma
Orden:	Descendente	
Mostrar:	*V*	*V*
Criterios:		
O:		