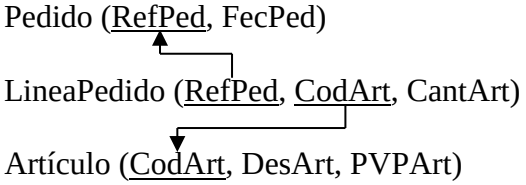
TEMA 7: CONSULTAS SENCILLAS

1. Consultas sencillas sobre una tabla.

A lo largo de este tema trabajaremos con la base de datos Pedidos creada en el anterior tema, y cuyo esquema relacional es el siguiente:



Para realizar la consulta de datos contenidos en tablas de una base de datos relacional se usa la sentencia SELECT. El formato básico de la sentencia SELECT es el siguiente:

```
SELECT expresión1, expresión2,…, expresiónn
FROM tabla1, tabla2, ..., tablan
WHERE criterio de selección
ORDER BY expresión1 [ASC|DESC], expresión2 [ASC|DESC],…, expresiónn [ASC|DESC];
```

Se van a explicar en esta sección cada una de estas cuatro cláusulas de la sentencia SELECT.

1.1. Cláusula FROM.

Después de la cláusula FROM se especificarán separados por comas los nombres de las tablas sobre las que se desea efectuar la consulta. Por ejemplo, si deseamos realizar una consulta sobre las tablas *Pedido* y *LineaPedido*, escribiremos:

```
SELECT expresión1, expresión2,…, expresiónn
FROM Pedido, LineaPedido;
```

Se pueden asignar nuevos nombres o alias a las tablas, los cuales deberán especificarse a continuación del nombre de la tabla tal cual, o separados por la palabra as. Por ejemplo, en las siguientes sentencias SQL asignaríamos el nombre *P* a la tabla *Pedido* y *L* a la tabla *LineaPedido*:

```
SELECT expresión1, expresión2,…, expresiónn
FROM Pedido P, LineaPedido L;

SELECT expresión1, expresión2,…, expresiónn
FROM Pedido as P, LineaPedido as L;
```

A la hora de escribir el nombre de una tabla también se puede indicar la base de datos a la que pertenece mediante la sintaxis *NombreBD.NombreTabla*. Esto se puede hacer siempre que se desee; no obstante, será necesario en caso de que en la sentencia SELECT se

trabaje con tablas de una base de datos distinta de aquella en la que nos encontremos. Así, en la siguiente sentencia SELECT se emplea la sintaxis *NombreBD.NombreTabla*.

```
SELECT expresión1, expresión2,…, expresiónn
FROM Pedidos.Pedido P, Pedidos.LineaPedido L;
```

1.2. Cláusula SELECT.

En la cláusula SELECT se especificarán varias expresiones separadas por comas, que normalmente son atributos de las tablas que se consultan. Por ejemplo, si queremos consultar el código y descripción de los artículos de la tabla *Articulo*, escribiremos:

```
SELECT CodArt, DesArt
FROM Articulo;
```

También se puede escribir después de la cláusula SELECT el símbolo *, indicando que se desea mostrar la totalidad de los atributos de la tabla especificada tras la cláusula FROM. Por ejemplo, para mostrar todos los atributos de la tabla *Artículo*, podríamos usar cualquiera de las dos siguientes sentencias SQL. Como se puede observar, el resultado de la ejecución es el mismo en ambos casos:

```
mysql> select CodArt, DesArt, PVPArt
    -> from Articulo;
+--------+----------------------+--------+
| CodArt | DesArt               | PVPArt |
+--------+----------------------+--------+
| A0012  | Goma de borrar       |   0.15 |
| A0043  | Bolígrafo azul       |   0.78 |
| A0075  | Lápiz 2B             |   0.55 |
| A0078  | Bolígrafo rojo normal |   1.05 |
| A0089  | Sacapuntas           |   0.25 |
+--------+----------------------+--------+
5 rows in set (0.00 sec)

mysql> select * from Articulo;
+--------+----------------------+--------+
| CodArt | DesArt               | PVPArt |
+--------+----------------------+--------+
| A0012  | Goma de borrar       |   0.15 |
| A0043  | Bolígrafo azul       |   0.78 |
| A0075  | Lápiz 2B             |   0.55 |
| A0078  | Bolígrafo rojo normal |   1.05 |
| A0089  | Sacapuntas           |   0.25 |
+--------+----------------------+--------+
5 rows in set (0.00 sec)
```

En la cláusula SELECT también se pueden asignar alias a los atributos si no consideramos conveniente o descriptivo el nombre del atributo. Para asignar un alias a un atributo basta con escribir después del nombre del atributo el texto que queremos que se muestre en lugar de su nombre. Este texto, si se trata de una sola palabra se puede poner tal cual o entre comillas simples (‘) o dobles (“). En el caso de que este texto conste de varias

palabras, es decir, si presenta algún espacio en blanco, será imprescindible ponerlo entre comillas simples o dobles. Por ejemplo, en la siguiente consulta se usan alias para dos atributos de la tabla *Artículo*:

```
mysql > select CodArt "Código del artículo", PVPArt Precio
    -> from Articulo;
+---------------------+--------+
| Código del artículo | Precio |
+---------------------+--------+
| A0012               |   0.15 |
| A0043               |   0.78 |
| A0075               |   0.55 |
| A0078               |   1.05 |
| A0089               |   0.25 |
+---------------------+--------+
5 rows in set (0.00 sec)
```

A veces es conveniente o incluso necesario especificar por cada atributo la tabla a la que pertenece. Será necesario en aquel caso en el que se realice una consulta sobre varias tablas y en ellas haya algún atributo con el mismo nombre. Para ello se empleará la sintaxis *NombreTabla.NombreAtributo* o bien *AliasTabla.NombreAtributo*. Por ejemplo, en la siguiente consulta, aunque usemos esta sintaxis, no sería necesario por tratarse de una consulta sobre una sola tabla.

```
mysql> Select A.CodArt, A.DesArt
    -> From Articulo A;
+--------+----------------------+
| CodArt | DesArt               |
+--------+----------------------+
| A0012  | Goma de borrar       |
| A0043  | Bolígrafo azul       |
| A0075  | Lápiz 2B             |
| A0078  | Bolígrafo rojo normal |
| A0089  | Sacapuntas           |
+--------+----------------------+
5 rows in set (0.00 sec)
```

Es posible indicar por cada atributo, además de la tabla a la que pertenece, la base de datos en la que se encuentra la tabla a la que pertenece el atributo, empleando la sintaxis *NombreBD.NombreTabla.NombreAtributo*. La anterior consulta empleando esta sintaxis se escribiría así:

```
mysql > select Pedidos.Articulo.CodArt, Pedidos.Articulo.DesArt
    -> from Pedidos.Articulo;
+--------+----------------------+
| CodArt | DesArt               |
+--------+----------------------+
| A0012  | Goma de borrar       |
| A0043  | Bolígrafo azul       |
| A0075  | Lápiz 2B             |
| A0078  | Bolígrafo rojo normal |
| A0089  | Sacapuntas           |
+--------+----------------------+
5 rows in set (0.00 sec)
```

Puede ocurrir que a la hora de realizar una consulta no nos interese que aparezcan en el resultado varias filas repetidas. Pues bien, para evitarlo deberemos añadir la palabra DISTINCT delante del nombre del atributo que se repite. Por ejemplo, si deseamos visualizar los códigos de los artículos que han sido solicitados en los pedidos que tenemos en la base de datos, podríamos emplear la instrucción:

```
mysql> select CodArt from LineaPedido;
+--------+
| CodArt |
+--------+
| A0043  |
| A0078  |
| A0043  |
| A0075  |
| A0012  |
| A0043  |
| A0089  |
+--------+
7 rows in set (0.00 sec)
```

Pero, como podemos ver, en el resultado nos aparecen los códigos de varios artículos repetidos; en concreto, el código *A0043* aparece tres veces porque este artículo ha sido solicitado en varios pedidos. Si no deseamos que nos aparezcan estos datos repetidos, tenemos que anteponer al atributo *CodArt* la palabra *distinct*:

```
mysql> select distinct CodArt from LineaPedido;
+--------+
| CodArt |
+--------+
| A0012  |
| A0043  |
| A0075  |
| A0078  |
| A0089  |
+--------+
5 rows in set (0.00 sec)
```

1.3. Cláusula WHERE.

En la cláusula WHERE se especificará la condición que deben cumplir las filas de la tabla que se desean mostrar. En esta condición se pueden emplear distintos tipos de operadores. Estos son:

Operadores de comparación o relacionales.

Estos operadores actúan sobre dos operandos colocados antes y después del operador y nos devuelven un valor 1 (verdadero), 0 (falso) o null (nulo). Estos operadores son los siguientes:

Operador	Significado
<	Menor que
<=	Menor o igual que
>	Mayor
>=	Mayor o igual que
=	Igual a
!= <>	Distinto de

Figura 1: Operadores de comparación.

Por ejemplo, para mostrar la descripción y el precio de los artículos con precio inferior a 0,75 €, escribiremos:

```
mysql> select DesArt, PVPArt
    -> from Articulo
    -> where PVPArt < 0.75;
+----------------+--------+
| DesArt         | PVPArt |
+----------------+--------+
| Goma de borrar |   0.15 |
| Lápiz 2B       |   0.55 |
| Sacapuntas     |   0.25 |
+----------------+--------+
3 rows in set (0.19 sec)
```

Si queremos mostrar los datos de los pedidos realizados el 23 de febrero de 2018, escribiremos:

```
mysql> select *
    -> from Pedido
    -> where FecPed = '2018-02-23';
+--------+------------+
| refped | fecped     |
+--------+------------+
| P0003  | 2018-02-23 |
+--------+------------+
1 row in set (0.00 sec)
```

Operadores aritméticos.

Los operadores aritméticos básicos son:

Operador	Significado
+	Suma
-	Resta
*	Multiplicación
/	División
div	División entera
%	Resto de división entera

Figura 2: Operadores aritméticos.

Estos operadores se pueden emplear tanto en la cláusula SELECT para crear campos calculados, como en la cláusula WHERE.

Operadores lógicos.

Estos operadores actúan sobre valores verdadero, falso o nulo y devuelven también un valor verdadero, falso o nulo. Existen tres operadores lógicos:

- El operador AND o && actúa sobre dos operandos y devuelve verdadero si los operandos sobre los que opera son verdaderos, de acuerdo con la siguiente tabla:

AND	**VERDADERO**	**FALSO**	**NULO**
VERDADERO	VERDADERO	FALSO	NULO
FALSO	FALSO	FALSO	FALSO
NULO	NULO	FALSO	NULO

- El operador OR o || también se aplica sobre dos operandos y devuelve verdadero si uno de los valores sobre los que opera es verdadero, de acuerdo con la siguiente tabla:

OR	**VERDADERO**	**FALSO**	**NULO**
VERDADERO	VERDADERO	VERDADERO	VERDADERO
FALSO	VERDADERO	FALSO	NULO
NULO	VERDADERO	NULO	NULO

- El operador NOT o ! actúa sobre un solo operando y devuelve el valor contrario a aquel sobre el que opera, es decir, NOT VERDADERO = FALSO, NOT FALSO = VERDADERO y NOT NULO = NULO.

Por ejemplo, si deseamos mostrar los datos de los artículos con precio entre 50 céntimos y un euro, emplearemos cualquiera de las dos siguientes consultas:

```
mysql> select * from Articulo
    -> where PVPArt>=0.5 and PVPArt<=1;
+--------+----------------+--------+
| CodArt | DesArt         | PVPArt |
+--------+----------------+--------+
| A0043  | Bolígrafo azul |   0.78 |
| A0075  | Lápiz 2B       |   0.55 |
+--------+----------------+--------+
2 rows in set (0.00 sec)

mysql> select * from Articulo
    -> where PVPArt>=0.5 && PVPArt<=1;
+--------+----------------+--------+
| CodArt | DesArt         | PVPArt |
+--------+----------------+--------+
| A0043  | Bolígrafo azul |   0.78 |
| A0075  | Lápiz 2B       |   0.55 |
+--------+----------------+--------+
2 rows in set (0.00 sec)
```

Operador *like*.

Este operador se utiliza para comparar cadenas de caracteres. Se pueden emplear dos caracteres comodín:

- %, que simboliza cualquier cadena de 0 a n caracteres.

- _, que simboliza cualquier carácter, pero solo uno.

Por ejemplo, si queremos mostrar el código y descripción de los artículos cuya descripción comience por la letra B, pondremos:

```
mysql> select CodArt, DesArt
    -> from Articulo
    -> where DesArt like 'B%';
+--------+----------------------+
| CodArt | DesArt               |
+--------+----------------------+
| A0043  | Bolígrafo azul       |
| A0078  | Bolígrafo rojo normal |
+--------+----------------------+
2 rows in set (0.26 sec)
```

Para mostrar todos los datos de los artículos cuya descripción comience por la letra B y contenga alguna u, pondremos:

```
msyql> select *
    -> from Articulo
    -> where DesArt like 'B%u%';
+--------+---------------+--------+
| CodArt | DesArt        | PVPArt |
+--------+---------------+--------+
| A0043  | Bolígrafo azul |   0.78 |
+--------+---------------+--------+
1 row in set (0.00 sec)
```

Si lo que deseamos es ver todos los datos de los artículos cuyo código contenga el número 4 en la penúltima posición, escribiremos:

```
mysql> select *
    -> from Articulo
    -> where CodArt like '%4_';
+--------+---------------+--------+
| CodArt | DesArt        | PVPArt |
+--------+---------------+--------+
| A0043  | Bolígrafo azul |   0.78 |
+--------+---------------+--------+
1 row in set (0.00 sec)
```

Operador *between*.

Este operador se utiliza para especificar los valores entre los que se desea que se encuentre el valor de un atributo. Se emplea el formato:

$$\texttt{between valor}_1\ \texttt{and valor}_2$$

Por ejemplo, si queremos visualizar las descripciones y precios de todos los artículos con valor entre 0,25 y 0,75 €, escribiremos:

```
mysql> select DesArt, PVPArt
    -> from Articulo
    -> where PVPArt between 0.25 and 0.75;
```

```
+------------+--------+
| DesArt     | PVPArt |
+------------+--------+
| Lápiz 2B   |   0.55 |
| Sacapuntas |   0.25 |
+------------+--------+
2 rows in set (0.03 sec)
```

También se puede preguntar si el valor de un atributo no se encuentra en un cierto intervalo anteponiendo a la palabra *between* el operador lógico *not*. Por ejemplo, para visualizar los datos de los artículos con código no incluido entre el A0050 y A0080, escribiremos la siguiente consulta:

```
msyql> select *
    -> from Articulo
    -> where CodArt not between 'A0050' and 'A0080';
+--------+----------------+--------+
| CodArt | DesArt         | PVPArt |
+--------+----------------+--------+
| A0012  | Goma de borrar |   0.15 |
| A0043  | Bolígrafo azul |   0.78 |
| A0089  | Sacapuntas     |   0.25 |
+--------+----------------+--------+
3 rows in set (0.00 sec)
```

Operador *in*.

Este operador permite consultar si el valor de un atributo se encuentra o no entre una serie de valores especificados a continuación de la palabra *in* entre paréntesis y separados por comas.

Por ejemplo, para mostrar los datos de las líneas de pedido correspondientes a los artículos con código A0043, A0012 y A0075, podríamos usar cualquiera de las dos siguientes consultas:

```
mysql> select *
    -> from LineaPedido
    -> where CodArt in('A0043', 'A0012' , 'A0075');

mysql> select *
    -> from LineaPedido
    -> where CodArt = 'A0043' or CodArt = 'A0012' or CodArt = 'A0075';
+--------+--------+---------+
| RefPed | CodArt | CantArt |
+--------+--------+---------+
| P0001  | A0043  |      10 |
| P0002  | A0043  |       5 |
| P0003  | A0075  |      20 |
| P0004  | A0012  |      15 |
| P0004  | A0043  |       5 |
+--------+--------+---------+
5 rows in set (0.02 sec)
```

Al igual que ocurría con el operador between, también se puede consultar si el valor de un atributo no se encuentra entre los especificados en una lista mediante *not in*. Así, si

queremos mostrar los datos de todas las líneas de pedido excepto aquellas en las que se solicitan 10 o 20 artículos, pondremos:

```
mysql> select *
    -> from LineaPedido
    -> where CantArt not in (10,20);
+--------+--------+---------+
| RefPed | CodArt | CantArt |
+--------+--------+---------+
| P0001  | A0078  |      12 |
| P0002  | A0043  |       5 |
| P0004  | A0012  |      15 |
| P0004  | A0043  |       5 |
| P0004  | A0089  |      50 |
+--------+--------+---------+
5 rows in set (1.01 sec)
```

Operador *is*.

Este operador se utiliza para saber si un atributo toma o no valor nulo dependiendo de si se utiliza *is null* o *is not null* respectivamente. Por ejemplo, si quisiésemos mostrar los datos de los artículos con descripción, escribiríamos la siguiente orden SQL:

```
SELECT *
FROM Articulo
WHERE DesArt is not null;
```

Si lo que queremos es mostrar los datos de todos los artículos que no tienen un precio asignado, usaremos la sentencia:

```
SELECT *
FROM Articulo
WHERE PVPArt is null;
```

También se puede emplear el operador *is* para comprobar si un valor es:

- Verdadero, escribiendo: valor is true.
- Falso, escribiendo: valor is false.
- Desconocido, escribiendo: valor is unknown.

1.4. Cláusula ORDER BY.

La cláusula ORDER BY sirve para especificar el o los campos o expresiones incluidas en la cláusula SELECT por los cuales se desea ordenar el resultado de la consulta. Por defecto la ordenación se realiza en orden ascendente, es decir, para los números y horas de menor a mayor, para los caracteres alfabéticos de la ‘a’ a la ‘z’ y para las fechas de la más antigua a la más reciente.

Después de cada expresión en función de la cual se desea realizar la ordenación se puede especificar:

- ASC: Sirve para indicar que se realice una ordenación ascendente. No es necesario incluir esta cláusula porque es la ordenación que se realiza por defecto.
- DESC: Sirve para indicar que se realice una ordenación descendente.

Por ejemplo, si deseamos mostrar los datos de los artículos de menos de 1 euro del más caro al más barato, emplearemos la orden SQL:

```
mysql> select *
    -> from Articulo
    -> where PVPArt < 1
    -> order by PVPArt desc;
+--------+---------------+--------+
| CodArt | DesArt        | PVPArt |
+--------+---------------+--------+
| A0043  | Bolígrafo azul |   0.78 |
| A0075  | Lápiz 2B       |   0.55 |
| A0089  | Sacapuntas     |   0.25 |
| A0012  | Goma de borrar |   0.15 |
+--------+---------------+--------+
4 rows in set (0.11 sec)
```

Si se incluye más de una expresión en la cláusula ORDER BY, se indica que en caso de que para varias filas del resultado el primer criterio no permita ordenarlas, se emplee (a modo de desempate) el segundo criterio especificado y así sucesivamente. Por ejemplo, si queremos mostrar todos los datos para las líneas de pedido en las que se soliciten menos de 15 unidades, ordenando el resultado en primer lugar por número de unidades (de más a menos) y en segundo lugar por referencia de pedido (de la a a la z), pondremos:

```
mysql> select RefPed, CodArt, CantArt
    -> from LineaPedido
    -> where CantArt < 15
    -> order by CantArt desc, RefPed;
+--------+--------+---------+
| RefPed | CodArt | CantArt |
+--------+--------+---------+
| P0001  | A0078  |      12 |
| P0001  | A0043  |      10 |
| P0002  | A0043  |       5 |
| P0004  | A0043  |       5 |
+--------+--------+---------+
4 rows in set (0.00 sec)
```

También se puede hacer referencia a las expresiones sobre las que se desee realizar la ordenación escribiendo los números que hacen referencia a su posición ordinal en la sentencia SELECT. Por ejemplo, en este caso podríamos poner:

```
mysql> select RefPed, CodArt, CantArt
    -> from LineaPedido
    -> where CantArt < 15
    -> order by 3 desc, 1;
+--------+--------+---------+
| RefPed | CodArt | CantArt |
+--------+--------+---------+
| P0001  | A0078  |      12 |
| P0001  | A0043  |      10 |
| P0002  | A0043  |       5 |
| P0004  | A0043  |       5 |
+--------+--------+---------+
4 rows in set (0.00 sec)
```

2. Consultas sencillas sobre varias tablas.

Las consultas que hemos realizado hasta ahora solo han afectado a una tabla, pero en muchos casos es necesario crear consultas que afecten a varias tablas. Para crear consultas multitabla, hemos de tener en cuenta lo siguiente:

- Se pueden combinar tantas tablas como se desee, las cuales deberán especificarse después de la cláusula FROM.
- En todas las cláusulas se puede hacer referencia a atributos de cualquiera de las tablas incluidas después de la cláusula FROM.
- Si hay atributos con el mismo nombre en varias tablas, para hacer referencia a un atributo con nombre repetido en varias tablas se debe utilizar la sintaxis NombreTabla.NombreColumna.
- El criterio de combinación de tablas se puede especificar mediante la palabra JOIN en la cláusula FROM o bien en la cláusula WHERE. Si no se especifica ningún criterio de combinación de tablas, se hará el producto cartesiano de las tablas especificadas tras la cláusula FROM.

2.1. Combinación de tablas empleado JOIN.

Veamos, para comenzar, la manera de combinar tablas mediante la especificación del criterio de combinación de tablas con la palabra JOIN en la cláusula FROM.

Vamos a crear una consulta en la que mostremos por cada línea de pedido, la referencia del pedido, el código del artículo, su descripción, el número de unidades solicitadas y el precio de cada artículo. Se mostrará el resultado ordenado por referencia del pedido y código de artículo. Observamos que hay atributos tanto de la tabla *LineaPedido* (*RefPed,*

CodArt y *CantArt*) como de la tabla *Articulo* (*DesArt* y *PVPArt*), motivo por el cual en la cláusula FROM de la consulta habremos de especificar estas dos tablas. Asignaremos alias a las tablas con el fin de poder referirnos a ellas con la letra inicial en lugar de tener que escribir el nombre completo de la tabla. En la cláusula SELECT deberán aparecer todos los atributos que queremos mostrar (los cinco indicados), pero para hacer referencia al atributo *CodArt* debemos escribir AliasTabla.NombreAtributo porque este atributo está repetido en las dos tablas. Podríamos pensar que la consulta sería así:

```
mysql> select RefPed, L.CodArt, DesArt, CantArt, PVPArt
    -> from LineaPedido L, Articulo A
    -> order by RefPed, L.CodArt;
+--------+--------+----------------------+---------+--------+
| RefPed | CodArt | DesArt               | CantArt | PVPArt |
+--------+--------+----------------------+---------+--------+
| P0001  | A0043  | Bolígrafo azul       |      10 |   0.78 |
| P0001  | A0043  | Bolígrafo rojo normal |      10 |   1.05 |
| P0001  | A0043  | Goma de borrar       |      10 |   0.15 |
| P0001  | A0043  | Lápiz 2B             |      10 |   0.55 |
| P0001  | A0043  | Sacapuntas           |      10 |   0.25 |
| P0001  | A0078  | Goma de borrar       |      12 |   0.15 |
| P0001  | A0078  | Lápiz 2B             |      12 |   0.55 |
| P0001  | A0078  | Sacapuntas           |      12 |   0.25 |
| P0001  | A0078  | Bolígrafo azul       |      12 |   0.78 |
| P0001  | A0078  | Bolígrafo rojo normal |      12 |   1.05 |
| P0002  | A0043  | Bolígrafo azul       |       5 |   0.78 |
| P0002  | A0043  | Bolígrafo rojo normal |       5 |   1.05 |
| P0002  | A0043  | Goma de borrar       |       5 |   0.15 |
| P0002  | A0043  | Lápiz 2B             |       5 |   0.55 |
| P0002  | A0043  | Sacapuntas           |       5 |   0.25 |
| P0003  | A0075  | Goma de borrar       |      20 |   0.15 |
| P0003  | A0075  | Lápiz 2B             |      20 |   0.55 |
| P0003  | A0075  | Sacapuntas           |      20 |   0.25 |
| P0003  | A0075  | Bolígrafo azul       |      20 |   0.78 |
| P0003  | A0075  | Bolígrafo rojo normal |      20 |   1.05 |
| P0004  | A0012  | Bolígrafo azul       |      15 |   0.78 |
| P0004  | A0012  | Bolígrafo rojo normal |      15 |   1.05 |
| P0004  | A0012  | Goma de borrar       |      15 |   0.15 |
| P0004  | A0012  | Lápiz 2B             |      15 |   0.55 |
| P0004  | A0012  | Sacapuntas           |      15 |   0.25 |
| P0004  | A0043  | Goma de borrar       |       5 |   0.15 |
| P0004  | A0043  | Lápiz 2B             |       5 |   0.55 |
| P0004  | A0043  | Sacapuntas           |       5 |   0.25 |
| P0004  | A0043  | Bolígrafo azul       |       5 |   0.78 |
| P0004  | A0043  | Bolígrafo rojo normal |       5 |   1.05 |
| P0004  | A0089  | Lápiz 2B             |      50 |   0.55 |
| P0004  | A0089  | Sacapuntas           |      50 |   0.25 |
| P0004  | A0089  | Bolígrafo azul       |      50 |   0.78 |
| P0004  | A0089  | Bolígrafo rojo normal |      50 |   1.05 |
| P0004  | A0089  | Goma de borrar       |      50 |   0.15 |
+--------+--------+----------------------+---------+--------+
35 rows in set (0.00 sec)
```

Como podemos observar en el resultado obtenido, se ha realizado el producto cartesiano entre las dos tablas (*LineaPedido* y *Articulo*), es decir, se ha relacionado cada línea de pedido de la tabla *LineaPedido* con cada artículo de la tabla *Articulo*.

Pero esto no es lo que normalmente nos interesa, sino que desearemos que se relacione el artículo solicitado en cada línea de pedido identificado por su código (*CodArt*) con los datos de dicho artículo (*DesArt* y *PVPArt*). Para conseguir esto, hemos de realizar una combinación de las dos tablas (JOIN) escribiendo JOIN entre las dos tablas y después detrás de la palabra ON la condición que vincule ambas tablas a través del atributo común a ambas (la clave ajena), en este caso *CodArt*. Esta condición deberá indicar, por tanto, que el atributo clave ajena *CodArt* de la tabla *LineaPedido* debe coincidir con el valor que tome el atributo clave primaria *CodArt* de la tabla *Articulo*. La consulta por tanto nos quedaría así:

```
mysql> select RefPed, L.CodArt, DesArt, CantArt, PVPArt
    -> from LineaPedido L join Articulo A on L.CodArt = A.CodArt
    -> order by RefPed, L.CodArt;
+--------+--------+----------------------+---------+--------+
| RefPed | CodArt | DesArt               | CantArt | PVPArt |
+--------+--------+----------------------+---------+--------+
| P0001  | A0043  | Bolígrafo azul       |      10 |   0.78 |
| P0001  | A0078  | Bolígrafo rojo normal |      12 |   1.05 |
| P0002  | A0043  | Bolígrafo azul       |       5 |   0.78 |
| P0003  | A0075  | Lápiz 2B             |      20 |   0.55 |
| P0004  | A0012  | Goma de borrar       |      15 |   0.15 |
| P0004  | A0043  | Bolígrafo azul       |       5 |   0.78 |
| P0004  | A0089  | Sacapuntas           |      50 |   0.25 |
+--------+--------+----------------------+---------+--------+
7 rows in set (0.00 sec)
```

En el resultado obtenido se puede observar como se ha relacionado cada línea de pedido con los datos del artículo solicitado en la misma.

En este caso, en la cláusula FROM se han incluido dos tablas, pero no existe ninguna limitación en cuanto al número de tablas. Eso sí, se debe tener en cuenta que deberán combinarse cada dos tablas escribiendo JOIN entre ambas y después de ON la condición de combinación.

Supongamos que queremos mostrar por cada pedido con fecha posterior al 20 de febrero de 2018, su referencia y fecha y además por cada uno de los artículos solicitados en él, su código, descripción, número de unidades solicitadas, importe del artículo e importe de la línea de pedido. Este último dato se calculará multiplicando el número de unidades solicitadas por el importe unitario de cada artículo. Vemos como en esta consulta necesitamos campos de las tres tablas, por lo que deberemos incluir las tres tras la palabra FROM. En este caso, además, vamos a utilizar alias para las tablas. Combinaremos en primer lugar la tabla *Pedido* con *LineaPedido* indicando que el atributo clave ajena de *LineaPedido* (*RefFed*) debe coincidir con el atributo clave primaria de *Pedido* (*RefPed*). El resultado de esta combinación lo combinaremos a su vez con la tabla *Articulo* indicando que el atributo clave ajena de *LineaPedido* (*CodArt*) coincida con el atributo clave primaria de

Articulo (*CodArt*). Para referirnos a los atributos *CodArt* y *RefPed*, al estar repetidos en dos tablas, deberemos poner AliasTabla.NombreAtributo. Por otro lado, en este caso, necesitamos crear un campo calculado como resultado de multiplicar el número de unidades solicitadas de un artículo en una línea de pedido por el precio del artículo. A estos campos es adecuado asignarles un alias. Para evitar que nos salgan muchos decimales en el resultado de esta operación haremos uso de la función *round* que admite dos parámetros: la cantidad que se desea redondear y el nº de decimales que se desea en el resultado (indicaremos que deseamos 2 decimales para que se muestren los céntimos de euro). La consulta nos quedaría como sigue:

```
mysql> select P.RefPed, FecPed, A.CodArt, DesArt, CantArt, PVPArt,
    -> round(CantArt * PVPArt, 2) "Importe línea"
    -> from Pedido P join LineaPedido L on P.RefPed = L.RefPed join Articulo A
    -> on A.CodArt = L.CodArt
    -> where FecPed>'2018/02/20';
+--------+------------+--------+----------------+---------+--------+---------------+
| RefPed | FecPed     | CodArt | DesArt         | CantArt | PVPArt | Importe línea |
+--------+------------+--------+----------------+---------+--------+---------------+
| P0003  | 2018-02-23 | A0075  | Lápiz 2B       |      20 |   0.55 |         11.00 |
| P0004  | 2018-02-25 | A0012  | Goma de borrar |      15 |   0.15 |          2.25 |
| P0004  | 2018-02-25 | A0043  | Bolígrafo azul |       5 |   0.78 |          3.90 |
| P0004  | 2018-02-25 | A0089  | Sacapuntas     |      50 |   0.25 |         12.50 |
+--------+------------+--------+----------------+---------+--------+---------------+
4 rows in set (0.24 sec)
```

Estas consultas multitabla que hemos llevado a cabo hasta el momento han sido composiciones internas, es decir, en ellas solo se han mostrado las filas de las tablas combinadas para las cuales se cumple el criterio de combinación.

Esta manera de combinar tablas es la aconsejada en el último estándar de SQL. Sin embargo, hay otras maneras de combinar tablas, algunas de las cuales se exponen a continuación:

2.2. Combinación de tablas empleando WHERE.

Esta es la forma tradicional de combinar tablas. La sintaxis consiste en escribir después de la palabra FROM separadas por comas los nombres de las tablas que se desean combinar y especificar el criterio de combinación de las mismas en la cláusula WHERE. Este criterio de combinación de tablas se unirá a las demás condiciones que pueda haber en la cláusula WHERE mediante el operador AND. Si se combinan más de dos tablas, deberán incluirse en la cláusula WHERE tantas condiciones unidas por el operador lógico AND como número de tablas menos 1, relacionando cada una de estas condiciones una clave ajena con su correspondiente clave primaria.

Vamos a ver cómo sería la siguiente consulta realizada con anterioridad. Se trata de mostrar por cada línea de pedido, la referencia del pedido, el código y descripción del artículo solicitado, el número de unidades solicitadas y el precio unitario del artículo:

```
mysql> select RefPed, L.CodArt, DesArt, CantArt, PVPArt
    -> from LineaPedido L, Articulo A
    -> where L.CodArt = A.CodArt
    -> order by RefPed, L.CodArt;
+--------+--------+----------------------+---------+--------+
| RefPed | CodArt | DesArt               | CantArt | PVPArt |
+--------+--------+----------------------+---------+--------+
| P0001  | A0043  | Bolígrafo azul       |      10 |   0.78 |
| P0001  | A0078  | Bolígrafo rojo normal |      12 |   1.05 |
| P0002  | A0043  | Bolígrafo azul       |       5 |   0.78 |
| P0003  | A0075  | Lápiz 2B             |      20 |   0.55 |
| P0004  | A0012  | Goma de borrar       |      15 |   0.15 |
| P0004  | A0043  | Bolígrafo azul       |       5 |   0.78 |
| P0004  | A0089  | Sacapuntas           |      50 |   0.25 |
+--------+--------+----------------------+---------+--------+
7 rows in set (0.00 sec).
```

Veamos también cómo sería la segunda consulta del apartado 2.1:

```
mysql> select P.RefPed, FecPed, A.CodArt, DesArt, CantArt, PVPArt,
    -> round(CantArt * PVPArt, 2) "Importe línea"
    -> from Pedido P, LineaPedido L, Articulo A
    -> where P.RefPed = L.RefPed and A.CodArt = L.CodArt and FecPed>'2018/02/20';
+--------+------------+--------+---------------+---------+--------+---------------+
| RefPed | FecPed     | CodArt | DesArt        | CantArt | PVPArt | Importe línea |
+--------+------------+--------+---------------+---------+--------+---------------+
| P0003  | 2018-02-23 | A0075  | Lápiz 2B      |      20 |   0.55 |         11.00 |
| P0004  | 2018-02-25 | A0012  | Goma de borrar |      15 |   0.15 |          2.25 |
| P0004  | 2018-02-25 | A0043  | Bolígrafo azul |       5 |   0.78 |          3.90 |
| P0004  | 2018-02-25 | A0089  | Sacapuntas    |      50 |   0.25 |         12.50 |
+--------+------------+--------+---------------+---------+--------+---------------+
4 rows in set (0.24 sec)
```

2.3. Combinación de tablas empleando JOIN USING.

La sintaxis consiste en escribir las tablas que se combinan en la cláusula FROM unidas por JOIN y a continuación indicar USING y entre paréntesis uno o varios atributos, que son aquellos por los que cuales se vinculan las tablas, es decir, las claves ajenas y correspondientes claves primarias. Debe tenerse en cuenta que estos atributos en las dos tablas deben tener el mismo nombre; en caso contrario, no es posible combinar ambas tablas usando JOIN USING.

A modo de ejemplo, vamos a ver cómo sería la siguiente consulta realizada con anterioridad. Se trata de mostrar por cada línea de pedido, la referencia del pedido, el código y descripción del artículo solicitado, el número de unidades solicitadas y el precio unitario del artículo. Pues bien, pondremos en la cláusula FROM las dos tablas involucradas (*LineaPedido* y *Articulo*) unidas con JOIN. Ambas tablas están vinculadas a través del atributo *CodArt*, que

es clave ajena en *LineaPedido* y clave primaria en *Articulo*. Recordemos que un requisito para poder usar JOIN USING es que el atributo que vincula las tablas tenga igual nombre en las dos tablas, condición que se cumple en este caso. Pues bien, pondremos USING y después entre paréntesis el nombre de este atributo. De esta manera no precisaremos de cláusula WHERE. La consulta quedará como sigue:

```
mysql> select RefPed, L.CodArt, DesArt, CantArt, PVPArt
    -> from LineaPedido L join Articulo A using (CodArt)
    -> order by RefPed, L.CodArt;
+--------+--------+----------------------+---------+--------+
| RefPed | CodArt | DesArt               | CantArt | PVPArt |
+--------+--------+----------------------+---------+--------+
| P0001  | A0043  | Bolígrafo azul       |      10 |   0.78 |
| P0001  | A0078  | Bolígrafo rojo normal |      12 |   1.05 |
| P0002  | A0043  | Bolígrafo azul       |       5 |   0.78 |
| P0003  | A0075  | Lápiz 2B             |      20 |   0.55 |
| P0004  | A0012  | Goma de borrar       |      15 |   0.15 |
| P0004  | A0043  | Bolígrafo azul       |       5 |   0.78 |
| P0004  | A0089  | Sacapuntas           |      50 |   0.25 |
+--------+--------+----------------------+---------+--------+
7 rows in set (0.19 sec)
```

La segunda consulta del apartado 2.1 también se podría llevar a cabo empleando este método de combinación de tablas porque los atributos que combinan cada par de tablas tienen el mismo nombre. Veamos también cómo quedaría esta consulta:

```
mysql> select P.RefPed, FecPed, A.CodArt, DesArt, CantArt, PVPArt,
    -> round(CantArt * PVPArt,2) "Importe línea"
    -> from Pedido P join LineaPedido L using(RefPed) join Articulo A using (CodArt)
    -> where FecPed>'2018/02/20';
+--------+------------+--------+---------------+---------+--------+---------------+
| RefPed | FecPed     | CodArt | DesArt        | CantArt | PVPArt | Importe línea |
+--------+------------+--------+---------------+---------+--------+---------------+
| P0003  | 2018-02-23 | A0075  | Lápiz 2B      |      20 |   0.55 |         11.00 |
| P0004  | 2018-02-25 | A0012  | Goma de borrar |      15 |   0.15 |          2.25 |
| P0004  | 2018-02-25 | A0043  | Bolígrafo azul |       5 |   0.78 |          3.90 |
| P0004  | 2018-02-25 | A0089  | Sacapuntas    |      50 |   0.25 |         12.50 |
+--------+------------+--------+---------------+---------+--------+---------------+
4 rows in set (0.24 sec)
```

2.4. Combinación de tablas empleando NATURAL JOIN.

Esta tercera opción de JOIN, al igual que ocurría con la anterior, requiere que los atributos que vinculan las tablas (clave ajena y correspondiente clave primaria) tengan el mismo nombre y, además, que no haya aparte de la clave ajena y correspondiente clave primaria, ningún otro atributo con igual nombre en las tablas que se vinculan. En este caso, solo hay que unir las tablas con NATURAL JOIN y el SGBD entiende que debe combinarlas a través del/de los atributo/s con idéntico nombre en las dos tablas, no siendo necesario especificar para nada los nombres de estos atributos. En el caso que venimos haciendo, por

tanto, no sería necesario especificar que el atributo que vincula las dos tablas es *CodArt*. La consulta nos quedaría como sigue:

```
mysql> select RefPed, L.CodArt, DesArt, CantArt, PVPArt
    -> from LineaPedido L natural join Articulo A
    -> order by RefPed, L.CodArt;
+--------+--------+----------------------+---------+--------+
| RefPed | CodArt | DesArt               | CantArt | PVPArt |
+--------+--------+----------------------+---------+--------+
| P0001  | A0043  | Bolígrafo azul       |      10 |   0.78 |
| P0001  | A0078  | Bolígrafo rojo normal |      12 |   1.05 |
| P0002  | A0043  | Bolígrafo azul       |       5 |   0.78 |
| P0003  | A0075  | Lápiz 2B             |      20 |   0.55 |
| P0004  | A0012  | Goma de borrar       |      15 |   0.15 |
| P0004  | A0043  | Bolígrafo azul       |       5 |   0.78 |
| P0004  | A0089  | Sacapuntas           |      50 |   0.25 |
+--------+--------+----------------------+---------+--------+
7 rows in set (0.00 sec)
```

La segunda consulta del apartado 2.1 también se podría llevar a cabo empleando este método de combinación de tablas porque los atributos que combinan cada par de tablas tienen el mismo nombre y además no hay ningún otro atributo con nombre repetido en las tablas que se combinan. Veamos también cómo quedaría esta consulta:

```
mysql> select P.RefPed, FecPed, A.CodArt, DesArt, CantArt, PVPArt,
    -> round(CantArt * PVPArt,2) "Importe línea"
    -> from Pedido P natural join LineaPedido L natural join Articulo A
    -> where FecPed > '2018-02-20';
+--------+------------+--------+---------------+---------+--------+---------------+
| RefPed | FecPed     | CodArt | DesArt        | CantArt | PVPArt | Importe línea |
+--------+------------+--------+---------------+---------+--------+---------------+
| P0003  | 2018-02-23 | A0075  | Lápiz 2B      |      20 |   0.55 |         11.00 |
| P0004  | 2018-02-25 | A0012  | Goma de borrar |      15 |   0.15 |          2.25 |
| P0004  | 2018-02-25 | A0043  | Bolígrafo azul |       5 |   0.78 |          3.90 |
| P0004  | 2018-02-25 | A0089  | Sacapuntas    |      50 |   0.25 |         12.50 |
+--------+------------+--------+---------------+---------+--------+---------------+
4 rows in set (0.00 sec)
```

2.5. Combinación de tablas empleando LEFT o RIGHT OUTER JOIN.

Las consultas multitabla que hemos llevado a cabo hasta el momento han sido composiciones internas, es decir, en ellas solo se han mostrado las filas de las tablas combinadas para las cuales se cumple el criterio de combinación.

Las composiciones externas son combinaciones entre dos o más tablas en las cuales aparecen en el resultado filas de una tabla aunque no exista correspondencia con filas de la otra tabla con la que se combina. Para explicar esto, en primer lugar vamos a añadir a la tabla *Pedido* dos nuevos pedidos con los siguientes datos, para los cuales no vamos a crear líneas de pedido:

RefPed	FecPed
P0005	2019/02/28
P0006	2019/02/03

```
mysql> insert into pedido values ('P0005', '2019/02/28');
Query OK, 1 row affected (0.11 sec)

mysql> insert into pedido values ('P0006', '2019/02/03');
Query OK, 1 row affected (0.04 sec)
```

Vamos a realizar una consulta entre las tablas *Pedido* y *LineaPedido* que nos muestre por cada pedido su referencia y fecha y por cada uno de los artículos solicitados en él, su código y el número de unidades pedidas. Esta consulta mediante una combinación interna nos quedaría como sigue:

```
mysql> select P.RefPed, FecPed, CodArt, CantArt
    -> from Pedido P join LineaPedido L on P.RefPed = L.RefPed;
+--------+------------+--------+---------+
| RefPed | FecPed     | CodArt | CantArt |
+--------+------------+--------+---------+
| P0001  | 2018-02-16 | A0043  |      10 |
| P0001  | 2018-02-16 | A0078  |      12 |
| P0002  | 2018-02-18 | A0043  |       5 |
| P0003  | 2018-02-23 | A0075  |      20 |
| P0004  | 2018-02-25 | A0012  |      15 |
| P0004  | 2018-02-25 | A0043  |       5 |
| P0004  | 2018-02-25 | A0089  |      50 |
+--------+------------+--------+---------+
7 rows in set (0.00 sec)
```

Como se puede observar, en el resultado no aparecen los pedidos nuevos añadidos (los de referencia *P0005* y *P0006*) porque no hay filas para estos pedidos en la tabla *LineaPedido*. Si deseamos realizar una combinación externa consistente en este caso en que se muestren todas las filas de la tabla *Pedido* aunque no tengan correspondencia con filas de la tabla *LineaPedido*, debemos usar una LEFT OUTER JOIN o simplemente una LEFT JOIN en lugar de una JOIN, indicando que para la tabla de la izquierda (*Pedido*) queremos que se muestren todos sus datos aunque no haya filas correspondientes en la tabla de la derecha (*LineaPedido*).

```
mysql> select P.RefPed, FecPed, CodArt, CantArt
    -> from Pedido P left outer join LineaPedido L on P.RefPed = L.RefPed;
+--------+------------+--------+---------+
| RefPed | FecPed     | CodArt | CantArt |
+--------+------------+--------+---------+
| P0001  | 2018-02-16 | A0043  |      10 |
| P0001  | 2018-02-16 | A0078  |      12 |
| P0002  | 2018-02-18 | A0043  |       5 |
| P0003  | 2018-02-23 | A0075  |      20 |
| P0004  | 2018-02-25 | A0012  |      15 |
| P0004  | 2018-02-25 | A0043  |       5 |
| P0004  | 2018-02-25 | A0089  |      50 |
| P0005  | 2019-02-28 | NULL   |    NULL |
| P0006  | 2019-02-03 | NULL   |    NULL |
+--------+------------+--------+---------+
9 rows in set (0.00 sec)
```

El mismo resultado habríamos obtenido si hubiésemos especificado en la cláusula FROM primero la tabla *LineaPedido* y después *Pedido* y hubiésemos escrito una RIGHT OUTER JOIN o una RIGHT JOIN en lugar de una LEFT OUTER JOIN, pues en este caso estaríamos indicando a MySQL que queremos que se muestren los datos de la tabla de la derecha aunque no haya filas combinadas en la tabla de la izquierda:

```
mysql> select P.RefPed, FecPed, CodArt, CantArt
    -> from LineaPedido L right join Pedido P on P.RefPed = L.RefPed;
+--------+------------+--------+---------+
| RefPed | FecPed     | CodArt | CantArt |
+--------+------------+--------+---------+
| P0001  | 2018-02-16 | A0043  |      10 |
| P0001  | 2018-02-16 | A0078  |      12 |
| P0002  | 2018-02-18 | A0043  |       5 |
| P0003  | 2018-02-23 | A0075  |      20 |
| P0004  | 2018-02-25 | A0012  |      15 |
| P0004  | 2018-02-25 | A0043  |       5 |
| P0004  | 2018-02-25 | A0089  |      50 |
| P0005  | 2019-02-28 | NULL   |    NULL |
| P0006  | 2019-02-03 | NULL   |    NULL |
+--------+------------+--------+---------+
9 rows in set (0.00 sec)
```