

T16 – DML (II)



Ej. 1.: Ejemplo de consultas en consola.

```
MariaDB [tienda_infor]> -- 5) Obtener el nombre y el precio en pesetas (el precio de la base de
datos se considerará euros; deberá ser multiplicado por 166,386 para hacer la conversión)
MariaDB [tienda_infor]> SELECT nombre, precio * 166.386 AS precio_pesetas FROM articulos;
```

nombre	precio_pesetas
Portátil Lenovo	199663.200
Monitor Samsung	49915.800
Teclado Logitech	8319.300
Ratón Microsoft	4991.580
Impresora HP	33277.200
Disco Duro Externo Western Digital	16638.600
Memoria USB SanDisk	3327.720
Altavoces Creative	13310.880
Webcam Logitech	6655.440
Router WiFi TP-Link	9983.160

```
10 rows in set (0.000 sec)
```

En esta consulta realizamos una multiplicación para hacer una conversión de Euros a Pesetas.

La consulta se realiza sobre la tabla *artículos*, de la cual se obtiene el nombre y el precio a través del comando SELECT.

El nombre no se ve modificado, pero la columna de precio es multiplicada para su conversión a pesetas y se pide que se muestre como *precio_pesetas* utilizando el comando AS.

Ej. 1.: Ejemplo de consultas en consola.

```
MariaDB [tienda_infor]> SELECT a.*, f.nombre AS nombre_fabricante  
-> FROM articulos AS a  
-> INNER JOIN fabricantes AS f ON a.cod_fabricantes = f.cod_fabricantes;
```

cod_articulos	nombre	precio	cod_fabricantes	nombre_fabricante
101	Portátil Lenovo	1200	1	Lenovo
102	Monitor Samsung	300	2	Samsung
103	Teclado Logitech	50	3	Logitech
104	Ratón Microsoft	30	4	Microsoft
105	Impresora HP	200	5	HP
106	Disco Duro Externo Western Digital	100	6	Western Digital
107	Memoria USB SanDisk	20	7	SanDisk
108	Altavoces Creative	80	8	Creative
109	Webcam Logitech	40	9	Logitech
110	Router WiFi TP-Link	60	10	TP-Link

En esta consulta obtendremos datos de la tabla *artículos* y *fabricantes* mediante el comando SELECT e INNER JOIN (realiza una unión interna entre las tablas *artículos* y *fabricantes*).

La consulta realiza abreviaciones a las tablas consultadas:

- artículos → FROM artículos AS a → a
- fabricantes → (FROM) fabricantes AS f → f

Además, también cambia el *nombre* de fabricantes a *nombre_fabricante*.

La unión entre las tablas se realiza mediante INNER JOIN (...) ON a.cod_fabricantes = f.cod_fabricantes: la cláusula ON especifica la condición de unión, que expresa una igualdad de valor entre las columnas seleccionadas de ambas tablas.

Ej. 2.: Ejemplo de consultas en consola.

```
MariaDB [empleados_db]> -- Obtener un listado completo de empleados, incluyendo por cada empleado los datos del empleado y de su departamento
MariaDB [empleados_db]> SELECT e.*, d.nombre AS nombre_departamento, d.presupuesto
-> FROM empleados AS e
-> JOIN departamentos AS d ON e.cod_departamento = d.cod_departamento;
```

DNI	nombre	apellidos	cod_departamento	nombre_departamento	presupuesto
01234567	Carlos	Fernández Pérez	10	Investigación y Desarrollo	180000
12345678	Juan	García Pérez	1	Ventas	100000
23456789	María	López Martínez	2	Marketing	80000
34567890	Antonio	Rodríguez Sánchez	3	Recursos Humanos	120000
45678901	Laura	González Fernández	4	Producción	150000
56789012	David	Martín Hernández	5	Finanzas	90000
67890123	Sofía	Pérez Díaz	6	Tecnología	200000
78901234	Manuel	Ruiz López	7	Logística	110000
89012345	Ana	Sánchez Rodríguez	8	Administración	130000
90123456	Elena	Gómez Martínez	9	Calidad	100000

En esta consulta utilizamos un JOIN como en el ejemplo anterior.

La consulta se realiza sobre la tabla *empleados* (*e*), de la cual se obtienen todas sus columnas (*e.**), la cual se combina con dos columnas de la tabla *departamentos* (*d*): *nombre* y *presupuesto*.

El resultado que obtenemos es una lista de todos los empleados donde, para cada empleado, se muestra su información personal (DNI, nombre, apellidos) junto con el nombre y presupuesto de su departamento.

Ej. 2.: Ejemplo de consultas en consola.

```
MariaDB [empleados_db]> -- Despedir a todos los empleados
MariaDB [empleados_db]> DELETE FROM empleados;
```

En esta consulta eliminamos todos los registros de la tabla *empleados*. Los registros de la tabla departamentos deben permanecer igual, ya que no se han modificado durante esta consulta. Podemos comprobar su eficacia observando las tablas y sus datos guardados en php-localhost:

SELECT * FROM `empleados`

☐ Perfilando [\[Editar en línea \]](#) [\[Editar \]](#) [\[Explicar SQL \]](#) [\[Crear código PHP \]](#) [\[Actualizar \]](#)

DNI

nombre

apellidos

cod_departamento

Operaciones sobre los resultados de la consulta

Crear vista

SELECT * FROM `departamentos`

☐ Perfilando [\[Editar en línea \]](#) [\[Editar \]](#) [\[Explicar SQL \]](#) [\[Crear código PHP \]](#) [\[Actualizar \]](#)

☐ Mostrar todo | Número de filas: 25 | Filtrar filas: Ordenar

Opciones extra

	cod_departamento	nombre	presupuesto
<input type="checkbox"/> Editar Copiar Borrar	1	Ventas	90000
<input type="checkbox"/> Editar Copiar Borrar	2	Marketing	72000
<input type="checkbox"/> Editar Copiar Borrar	3	Recursos Humanos	108000
<input type="checkbox"/> Editar Copiar Borrar	4	Producción	135000
<input type="checkbox"/> Editar Copiar Borrar	5	Finanzas	81000
<input type="checkbox"/> Editar Copiar Borrar	6	Tecnología	180000
<input type="checkbox"/> Editar Copiar Borrar	7	Logística	99000
<input type="checkbox"/> Editar Copiar Borrar	8	Administración	117000
<input type="checkbox"/> Editar Copiar Borrar	9	Calidad	90000
<input type="checkbox"/> Editar Copiar Borrar	10	Investigación y Desarrollo	162000
<input type="checkbox"/> Editar Copiar Borrar	11	Nueva_Calidad	36000

Ej. 3.: Ejemplo de consultas en consola.

```
MariaDB [almacenes_db]> -- 4) Obtener el valor medio de todas las cajas
MariaDB [almacenes_db]> SELECT AVG(valor) AS valor_medio FROM cajas;
+-----+
| valor_medio |
+-----+
|    106.0000 |
+-----+
```

En esta consulta queremos averiguar el valor medio de todas las cajas y para ello, en vez de sumar el valor total de las cajas y dividirlo por el numero de cajas totales vamos a utilizar la función `AVG()` o *average/promedio*.

`AVG()` sólo necesita que especifiquemos sobre qué columna queremos aplicar la fórmula, así que introducimos el nombre de la columna deseada entre los paréntesis: `AVG(valor)`.

El resultado lo mostraremos en una columna llamada `valor_medio` (recordamos el uso de la clausula `AS` para ello).

¡No nos olvidemos de que estos datos los estamos tomando desde la tabla *cajas* usando `FROM`!

Ej. 3.: Ejemplo de consultas en consola.

```
MariaDB [almacenes_db]> -- 8) Obtener el numero de cajas que hay en cada almacén
MariaDB [almacenes_db]> SELECT cod_almacen, COUNT(*) AS num_cajas
    -> FROM cajas
    -> GROUP BY cod_almacen;
```

cod_almacen	num_cajas
1	1
2	1
3	1
4	1
5	1
6	1
7	1
8	1
9	1
10	1

Otra consulta interesante es esta, donde queremos averiguar el número de cajas que hay en cada uno de los almacenes de la tabla *cajas* (FROM).

Para ellos nos fijaremos en el `cod_almacen` (SELECT) y utilizaremos la función `COUNT()`, la cual cuenta las filas de cada grupo, por lo que `COUNT(*)` contará todas las filas, lo que equivale al número de cajas de cada almacén.

Para que la función `COUNT(*)` pueda lograr esto, utilizamos `GROUP BY cod_almacen`, lo que hace que cuando `COUNT(*)` se aplique se utilice la función sobre las filas que tienen el mismo `cod_almacen`, mostrando así las cajas de cada almacén.

Ej. 4.: Ejemplo de consultas en consola.

```
MariaDB [peliculas_salas]> -- 5) Mostrar la información de todas las salas y, si se proyecta alguna película en la sala,
mostrar también la información de la película
MariaDB [peliculas_salas]> SELECT s.*, p.*
-> FROM salas AS s
-> LEFT JOIN peliculas AS p ON s.cod_pelicula = p.cod_pelicula;
```

cod_sala	nombre	cod_pelicula	cod_pelicula	nombre	cal_edad
S01	Sala de Estreno	1	1	Spider-Man: No Way Home	7
S02	Sala de Aventuras	2	2	Dune	12
S03	Sala de Ciencia Ficción	3	3	Eternals	7
S04	Sala de Animación	4	4	Sing 2	0
S05	Sala de Comedia	5	5	La familia Addams 2: La gran excursión	0
S06	Sala de Terror	6	6	Ghostbusters: Afterlife	7
S07	Sala de Suspenso	7	7	Scream	16
S08	Sala de Acción	8	8	Matrix Resurrections	12
S09	Sala de Drama	9	9	El médico africano	12
S10	Sala de Romance	10	10	Licorice Pizza	12

En esta consulta utilizamos LEFT JOIN para combinar las tablas *salas* y *peliculas* según el *cod_película*.

La tabla principal será *salas*, y la secundaria *películas*, ya que es ésta la que esta seguida de LEFT JOIN. Como siempre, hemos abreviado las tablas a *s* y *p*.

En nuestro caso, todas las salas cuentan con una película en emisión, por lo que la información de la película siempre es visible (cada película con su sala correspondiente). Sin embargo, en caso de que una sala estuviese vacía, se mostraría la información de la sala (puesto que hemos usado SELECT s.*) pero no se mostraría ninguna información en las columnas de película (nombre y *cal_edad*, que las hemos seleccionado antes con SELECT p.*).

Ej. 4.: Ejemplo de consultas en consola.

```
MariaDB [peliculas_salas]> -- 7) Mostrar los nombres de las películas que no se proyectan en ninguna sala
MariaDB [peliculas_salas]> SELECT nombre
-> FROM peliculas
-> WHERE cod_pelicula NOT IN (SELECT cod_pelicula FROM salas);
```

En esta consulta queremos saber las películas QUE NO se proyectan; es decir, las que nunca hubiesen salido en la consulta que acabamos de explicar.

Estamos realizando una subconsulta para obtener los cod_película de aquellas películas que no se proyectan.

La condición de búsqueda es WHERE, donde expresamos que cod_película NO DEBE estar (NOT IN) en ninguna sala (SELECT cod_película FROM salas).

La manera de entender esta consulta es:

- obtenemos, desde salas, los cod_película (SELECT cod_película FROM salas)
- con este listado, filtramos aquellos cod_película que no estén en ninguna sala (WHERE cod_película NOT IN)
- por último, desde el listado de los cod_película que hayamos obtenido, creamos la lista de los nombres de la película, cogiendo los datos desde la tabla peliculas (SELECT nombre FROM peliculas).