CENTRE ESPECÍFIC
D´EDUCACIÓ A DISTÀNCIA DE
LA COMUNITAT VALENCIANA

UNIDAD 6

MODELO FÍSICO DQL

BASES DE DATOS 22/23
CFGS DAW

PARTE 2. NIVEL MEDIO

Revisado por:
Sergio Badal, Abelardo Martínez y Pau Miñana
Autores:
Raquel Torres
Paco Aldarias
Fecha: 06/02/23

ÍNDICE DE CONTENIDO

 CONSEJO

El lenguaje SQL **NO es sensible a mayúsculas/minúsculas** pero, como algunos sistemas operativos sí que lo son, te recomendamos que uses seas fiel a la sintaxis usada al crear los metadatos (tablas, atributos, restricciones...).

No obstante, se considera "buena praxis":

=> Usar siempre minúsculas o UpperCamelCase en los metadatos.

=> Usar mayúsculas en los alias de las tablas si las tablas están en minúsculas y viceversa, ya que ayuda a entender mejor las consultas.

=> Los comandos SQL (SELECT, INSERT, FROM, WHERE...) también se recomiendan en mayúsculas, pero son igual de válidos en minúsculas.

UD6.2. MODELO FÍSICO DQL. NIVEL MEDIO

1. AGRUPACIONES

Es común utilizar consultas en las que se desee agrupar los datos a fin de obtener datos calculados a partir agrupaciones de distintos registros.

 CONSEJO

Las agrupaciones, como pasaba con los diagramas E-R, son **bastante** complejas de entender al principio y solo con MUCHA práctica, se consiguen detectar y utilizar con cierta soltura.

Así que no te desanimes si las ves complicadas. Es normal :-)

Con GROUP BY la instrucción SELECT queda de esta forma:

```
SELECT listaDeExpresiones
FROM listaDeTablas
[JOIN tablasRelacionadasYCondicionesDeRelación]
[WHERE condiciones]
[GROUP BY grupos]
[HAVING condicionesDeGrupo]
[ORDER BY columnas];
```

En el apartado GROUP BY, se indican las columnas por las que se agrupa. La función de este apartado es crear un único registro por cada valor distinto en las columnas del grupo.

Dada la siguiente tabla si, por ejemplo, agrupamos en base a las columnas tipo y modelo en una tabla de existencias se creará un único registro por cada tipo y modelo distintos:

SELECT tipo as TI, modelo, n_almacen, cantidad **FROM** existencias	**SELECT** tipo as TI, modelo **FROM** existencias **GROUP BY** TI, modelo;
Si la tabla de existencias sin agrupar es:	La consulta anterior creará esta salida:

TI	MODELO	N_ALMACEN	CANTIDAD
AR	6	1	2500
AR	6	2	5600
AR	6	3	2430
AR	9	1	250
AR	9	2	4000
AR	9	3	678
AR	15	1	5667
AR	20	3	43
BI	10	2	340
BI	10	3	23
BI	38	1	1100
BI	38	2	540
BI	38	3	152

TI	MODELO
AR	6
AR	9
AR	15
AR	20
BI	10
BI	38

Es decir, es un resumen de los datos anteriores. Los datos **n_almacen** y **cantidad** no están disponibles directamente ya que son distintos en los registros del mismo grupo.

1.1 Funciones agregadas con agrupaciones

La funciones agregadas se pueden usar con o sin grupos (GROUP BY). Recordemos cuáles eran:

- **SUM** calcula la suma de los valores de la columna.
- **AVG** calcula la media aritmética de los valores de la columna.
- **COUNT** devuelve el número de elementos que tiene la columna.
- **MAX** devuelve el valor máximo de la columna.
- **MIN** devuelve el valor mínimo de la columna.

 ATENCIÓN

Todos los campos que aparecen en SELECT deben aparecen en el GROUP BY, excepto si forman parte de una o varias funciones de agregado.

CORRECTAS:

SELECT a FROM t GROUP BY a;

SELECT max(a), min(a), count(*) FROM t;

SELECT max(a), min(a), count(*) FROM t GROUP BY a;

SELECT a, max(a), min(a), count(*) FROM t GROUP BY a;

INCORRECTAS:

SELECT a, max(a) FROM t;

SELECT a, b FROM t GROUP BY a;

1.2 Particularidades del COUNT(*)

El count(*) cuenta todos los registros y se comporta de manera diferente si lo usamos con una agrupación o sin ella.

Además, el **count(campo)** cuenta todos los **registros donde ese campo no sea nulo**.

Vemos un mismo ejemplo en todas sus combinaciones con una nueva tabla llamada ALUMNADO:

```
mysql> desc alumnado;
+------------------+-------------+------+-----+---------+----------------+
| Field            | Type        | Null | Key | Default | Extra          |
+------------------+-------------+------+-----+---------+----------------+
| ida              | int         | NO   | PRI | NULL    | auto_increment |
| nombre           | varchar(50) | NO   |     | NULL    |                |
| anyo_nacimiento  | int         | YES  |     | NULL    |                |
+------------------+-------------+------+-----+---------+----------------+
```

```
mysql> select * from alumnado order by anyo_nacimiento desc;
+-----+--------+-----------------+
| ida | nombre | anyo_nacimiento |
+-----+--------+-----------------+
|   3 | Maria  |            1980 |
|   6 | Pedro  |            1980 |
|   1 | Juan   |            1978 |
|   4 | Marta  |            1978 |
|   2 | Luis   |            NULL |
|   5 | Elena  |            NULL |
+-----+--------+-----------------+
6 rows in set (0,00 sec)
```

```
mysql> select count(*) from alumnado;
+----------+
| count(*) |
+----------+
|        6 |
+----------+
1 row in set (0,00 sec)

mysql> select count(*) from alumnado group by anyo_nacimiento;
+----------+
| count(*) |
+----------+
|        2 |
|        2 |
|        2 |
+----------+
3 rows in set (0,00 sec)
```

```
mysql> select count(*), anyo_nacimiento from alumnado;
ERROR 1140 (42000): In aggregated query without GROUP BY, expression #2 of SELECT list contains
nonaggregated column 'patients.alumnado.anyo_nacimiento'; this is incompatible with sql_mode=on
y_full_group_by
mysql> select count(*), anyo_nacimiento from alumnado group by anyo_nacimiento;
+----------+-----------------+
| count(*) | anyo_nacimiento |
+----------+-----------------+
|        2 |            1978 |
|        2 |            NULL |
|        2 |            1980 |
+----------+-----------------+
3 rows in set (0,00 sec)

mysql> select count(anyo_nacimiento) from alumnado;
+------------------------+
| count(anyo_nacimiento) |
+------------------------+
|                      4 |
+------------------------+
1 row in set (0,00 sec)

mysql> select count(anyo_nacimiento) from alumnado group by anyo_nacimiento;
+------------------------+
| count(anyo_nacimiento) |
+------------------------+
|                      2 |
|                      0 |
|                      2 |
+------------------------+
3 rows in set (0,00 sec)

mysql> select count(anyo_nacimiento), anyo_nacimiento from alumnado group by anyo_nacimiento;
+------------------------+-----------------+
| count(anyo_nacimiento) | anyo_nacimiento |
+------------------------+-----------------+
|                      2 |            1978 |
|                      0 |            NULL |
|                      2 |            1980 |
+------------------------+-----------------+
3 rows in set (0,00 sec)
```

Veamos algunos ejemplos de uso de agregadas en los que **necesitamos hacer un GROUP BY**.

1.2.1 Consulta 1

Mostrar cuántos empleados hay en cada departamento. Sería recomendable poner un alias para count(*), por ejemplo 'Num_Empleados'.

```
mysql> select count(*), dpto from empleados
    -> group by dpto
    -> order by dpto;
+----------+------+
| count(*) | dpto |
+----------+------+
|        1 | ALM  |
|        1 | CONT |
|        3 | IT   |
+----------+------+
3 rows in set (0.00 sec)
```

 ATENCIÓN

Si no agrupamos por departamento… ¿qué resultado obtendríamos?

Respuesta: ¡UN ERROR o datos INDETERMINADOS!

1.2.2 Consulta 2

Mostrar cuál es la mayor cantidad pedida de un producto en cada uno de los pedidos.

```
mysql> select NumPedido, max(Cantidad) from productospedido
    -> group by NumPedido
    -> order by NumPedido;
+-----------+----------------+
| NumPedido | max(Cantidad)  |
+-----------+----------------+
|         1 |             12 |
|         2 |             15 |
|         3 |             20 |
|         4 |             30 |
|         5 |             18 |
+-----------+----------------+
```

1.2.3 Consulta 3

Mostrar cuál es la mayor cantidad pedida de cada producto.

```
mysql> select RefeProducto, max(Cantidad) from productospedido
    -> group by RefeProducto
    -> order by RefeProducto;
+--------------+----------------+
| RefeProducto | max(Cantidad)  |
+--------------+----------------+
| AFK11        |             30 |
| BB75         |             12 |
| HM12         |             10 |
| NPP10        |             10 |
| P3R20        |             18 |
| PM30         |             20 |
+--------------+----------------+
```

1.3 Agrupamiento con condiciones

Los agrupamientos también nos permiten añadir condiciones de filtrado. Estas condiciones se realizarán con la cláusula HAVING, van detrás de la cláusula GROUP BY y se comprobarán después de haberse realizado el agrupamiento y, por tanto, filtrarán el resultado de éste.

 IMPORTANTE

Las condiciones que podemos incluir en el HAVING son las mismas que hemos utilizado en los filtros de la cláusula WHERE.

La sintaxis completa quedará:

SELECT [**DISTINCT**] select_expr [,select_expr] ...

[**FROM** tabla]

[**WHERE** filtro]

[**GROUP BY** expr [, expr]....]

[**HAVING** filtro_grupos]

[**ORDER BY** {nombre_columna | expr | posición} [**ASC** | **DESC**] , ...]

 CONSEJO

La cosa se complica al introducir el HAVING. Quizás te sirva saber el orden en el que se ejecuta una sentencia SELECT. Es éste:

1º FROM/ JOIN: (lo veremos después)

2º WHERE

3º GROUP BY

4º HAVING

5º SELECT

6º DISTINCT

7º ORDER BY

Más info: https://picodotdev.github.io/blog-bitix/2019/06/orden-de-ejecucion-de-las-clausulas-de-las-sentencias-select-de-sql/

Veamos algunos ejemplos:

1.3.1 Consulta 4

Mostrar el número de empleados de los departamentos que tengan más de un empleado ordenado por el código del departamento. Sería recomendable poner un alias, por ejemplo 'Num_Empleados'.

```
mysql> select count(*),Dpto from empleados
    -> group by Dpto
    -> having count(*) > 1
    -> order by Dpto;
+----------+------+
| count(*) | Dpto |
+----------+------+
|        3 | IT   |
+----------+------+
1 row in set (0.00 sec)
```

1.3.2 Consulta 5

Mostrar los pedidos en los que se ha pedido un total superior a 25 unidades ordenado por el número de pedido de mayor a menor.

```
mysql> select NumPedido, sum(cantidad) from productospedido
    -> group by NumPedido
    -> having sum(cantidad) > 25
    -> order by NumPedido DESC;
+-----------+---------------+
| NumPedido | sum(cantidad) |
+-----------+---------------+
|         5 |            26 |
|         4 |            42 |
|         3 |            40 |
+-----------+---------------+
3 rows in set (0.00 sec)
```

1.3.3 Consulta 6

Mostrar los artículos de los que se han pedido más de 25 unidades ordenados de mayor a menor por el número de unidades pedidas.

```
mysql> select RefeProducto, sum(cantidad) from productospedido
    -> group by RefeProducto
    -> having sum(cantidad) > 25
    -> order by sum(cantidad);
+--------------+---------------+
| RefeProducto | sum(cantidad) |
+--------------+---------------+
| AFK11        |            42 |
| P3R20        |            43 |
+--------------+---------------+
2 rows in set (0.00 sec)
```

2. CONSULTAS MULTITABLA. LA SENTENCIA JOIN

Hasta ahora hemos realizado siempre las consultas sobre una sola tabla. Obviamente, esto ha limitado bastante nuestras posibilidades de crear consultas complejas. Sin embargo, ahora vamos a aprender a realizar consultas multitabla y ello nos permitirá realizar ejercicios más ambiciosos.

Existen varios tipos de consultas que leen de varias tablas de forma simultánea, estos son:

- JOIN cruzado (CROSS) o producto cartesiano
- JOIN interno (INNER JOIN)
- JOIN externo (OUTER JOIN)

2.1 Producto cartesiano (CROSS JOIN)

Este tipo de consulta mostrará como resultado la combinación cada una de las filas de una tabla con todas las de la otra tabla.

En CROSS JOIN nunca se ponen condiciones para filtrar el resultado, solamente se indican los nombres de las tablas.

Se puede realizar de forma explícita o de forma implícita (que es la más habitual).

Forma explicita:

SELECT *

FROM departamentos **CROSS JOIN** empleados;

El resultado que obtenemos lo puedes ver en la siguiente página:

```
mysql> select *
    -> from departamentos cross join empleados;
+---------+----------------+-----------------+-----------+-----------------+---
----------+------------+------+-------+
| CodDpto | Nombre         | Ubicacion       | dni       | nombre          | es
pecialidad | fechaalta  | dpto | codp  |
+---------+----------------+-----------------+-----------+-----------------+---
----------+------------+------+-------+
| ADM     | Administración | Planta quinta U2 | 12345678A | Alberto Gil     | Co
ntable     | 2010-12-10 | CONT | MAD20 |
| ALM     | Almacén        | Planta baja U1   | 12345678A | Alberto Gil     | Co
ntable     | 2010-12-10 | CONT | MAD20 |
| CONT    | Contabilidad   | Planta quinta U1 | 12345678A | Alberto Gil     | Co
ntable     | 2010-12-10 | CONT | MAD20 |
| IT      | Informática    | Planta sótano U3 | 12345678A | Alberto Gil     | Co
ntable     | 2010-12-10 | CONT | MAD20 |
| ADM     | Administración | Planta quinta U2 | 23456789B | Mariano Sanz    | In
formática  | 2011-10-04 | IT   | NULL  |
| ALM     | Almacén        | Planta baja U1   | 23456789B | Mariano Sanz    | In
formática  | 2011-10-04 | IT   | NULL  |
| CONT    | Contabilidad   | Planta quinta U1 | 23456789B | Mariano Sanz    | In
formática  | 2011-10-04 | IT   | NULL  |
| IT      | Informática    | Planta sótano U3 | 23456789B | Mariano Sanz    | In
formática  | 2011-10-04 | IT   | NULL  |
| ADM     | Administración | Planta quinta U2 | 45678901D | Ana Silván      | In
formática  | 2012-11-25 | IT   | MAD20 |
| ALM     | Almacén        | Planta baja U1   | 45678901D | Ana Silván      | In
formática  | 2012-11-25 | IT   | MAD20 |
| CONT    | Contabilidad   | Planta quinta U1 | 45678901D | Ana Silván      | In
formática  | 2012-11-25 | IT   | MAD20 |
| IT      | Informática    | Planta sótano U3 | 45678901D | Ana Silván      | In
formática  | 2012-11-25 | IT   | MAD20 |
| ADM     | Administración | Planta quinta U2 | 67890123F | Roberto Milán   | Lo
gística    | 2010-05-02 | ALM  | NULL  |
| ALM     | Almacén        | Planta baja U1   | 67890123F | Roberto Milán   | Lo
gística    | 2010-05-02 | ALM  | NULL  |
| CONT    | Contabilidad   | Planta quinta U1 | 67890123F | Roberto Milán   | Lo
gística    | 2010-05-02 | ALM  | NULL  |
| IT      | Informática    | Planta sótano U3 | 67890123F | Roberto Milán   | Lo
gística    | 2010-05-02 | ALM  | NULL  |
| ADM     | Administración | Planta quinta U2 | 78901234G | Rafael Colmenar | In
formática  | 2013-06-10 | IT   | TO451 |
| ALM     | Almacén        | Planta baja U1   | 78901234G | Rafael Colmenar | In
formática  | 2013-06-10 | IT   | TO451 |
| CONT    | Contabilidad   | Planta quinta U1 | 78901234G | Rafael Colmenar | In
formática  | 2013-06-10 | IT   | TO451 |
| IT      | Informática    | Planta sótano U3 | 78901234G | Rafael Colmenar | In
formática  | 2013-06-10 | IT   | TO451 |
+---------+----------------+-----------------+-----------+-----------------+---
----------+------------+------+-------+
20 rows in set (0.00 sec)
```

Aunque el resultado es un poco lioso tienes que fijarte en que primero aparecen las cuatro filas de los cuatro departamentos (ADM, ALM, CONT e IT) unidos a Alberto Gil, después otra vez los cuatro departamentos unidos a Mariano Sanz, y así con todos.

Cada fila de una tabla se une a todas las filas de la otra tabla, es decir, **el producto cartesiano (o todos con todos**). Tenemos 4 departamentos y 5 empleados 4 x 5 = 20 filas que tiene el resultado.

Forma implícita

Esta forma es la más común:

SELECT *

FROM departamentos, empleados;

Podemos colocar las tablas separadas por una coma y lo tomará como un CROSS JOIN.

Comprueba que el resultado obtenido con esta forma abreviada es el mismo que el anterior.

2.2 JOIN interno (INNER JOIN)

El INNER JOIN es el JOIN que se emplea por defecto. Consiste en realizar el producto cartesiano de las tablas involucradas y después aplicar un filtro para seleccionar aquellas filas que deseamos mostrar en las consultas.

Es decir, estamos ante un producto cartesiano con filtros. Normalmente los filtros que se utilizan suelen incluir la clave principal de una tabla con la clave foránea en la otra tabla; es decir, los campos que las relacionan.

Este tipo de JOIN también se puede indicar de forma explícita e implícita, veamos ambas:

Forma explicita

SELECT *

FROM departamentos **INNER JOIN** empleados

ON departamentos.coddpto = empleados.dpto;

Estamos seleccionando todos los campos de la tabla Departamentos y de la tabla Empleados donde el código del departamento coincide con el departamento al que pertenece el empleado, es decir estamos mostrando los departamentos con los empleados que lo integran.

Fíjate que hemos colocado el nombre de la tabla, punto, nombre del campo, a la hora de hacer la comparación. Es simplemente para que veas que tomamos un campo de cada una de las tablas, pero en este caso concreto no sería necesario colocar el nombre de la tabla pues los campos tienen nombres diferentes (si hubiesen tenido el mismo nombre sí hubiese sido necesario).

El resultado, aunque un poco lioso como antes, será:

```
mysql> SELECT *
    ->  FROM DEPARTAMENTOS INNER JOIN EMPLEADOS
    ->  ON DEPARTAMENTOS.CODDPTO = EMPLEADOS.DPTO;
+---------+------------+-----------------+-----------+----------------+-----
---------+------------+------+-------+
| CodDpto | Nombre     | Ubicacion       | dni       | nombre         | espe
cialidad | fechaalta  | dpto | codp  |
+---------+------------+-----------------+-----------+----------------+-----
---------+------------+------+-------+
| ALM     | Almacén    | Planta baja U1  | 67890123F | Roberto Milán  | Logi
stica    | 2010-05-02 | ALM  | NULL  |
| CONT    | Contabilidad | Planta quinta U1 | 12345678A | Alberto Gil    | Cont
able     | 2010-12-10 | CONT | MAD20 |
| IT      | Informática  | Planta sótano U3 | 23456789B | Mariano Sanz   | Info
rmática  | 2011-10-04 | IT   | NULL  |
| IT      | Informática  | Planta sótano U3 | 45678901D | Ana Silván     | Info
rmática  | 2012-11-25 | IT   | MAD20 |
| IT      | Informática  | Planta sótano U3 | 78901234G | Rafael Colmenar | Info
rmática  | 2013-06-10 | IT   | TO451 |
+---------+------------+-----------------+-----------+----------------+-----
---------+------------+------+-------+
5 rows in set (0.03 sec)
```

Como puedes observar, ahora aparece cada departamento seguido de los empleados que pertenecen a ese departamento.

Forma implícita

Esta forma es la más común:

SELECT *

FROM departamentos, empleados

WHERE departamentos.coddpto = empleados.dpto;

Al igual que en la explícita, en este caso se podría haber omitido el nombre de las tablas en el filtro al tener los campos un nombre diferente.

Comprueba que el resultado obtenido con esta forma implícita es el mismo que el anterior.

2.3 JOIN externo (OUTER JOIN)

Este tipo de JOIN es diferente al anterior, ya que en el anterior para que se mostrase una fila de cualquier tabla en el resultado debía existir una correspondencia entre ellas. En el JOIN externo la filosofía es distinta: se mostrarán todas las filas de una tabla (tanto si tienen correspondencia como si no) y junto a ella se añadirán las filas correspondientes de la otra tabla.

Existen tres tipos de JOIN EXTERNO:

- **LEFT OUTER JOIN** o **LEFT JOIN** (Join Izquierdo). La tabla de la izquierda muestra todas sus filas y de la tabla de la derecha solamente las que se correspondan con las de la izquierda.
- **RIGHT OUTER JOIN** o **RIGHT JOIN** (Join Derecho). La tabla de la derecha muestra todas sus filas y de la tabla de la izquierda solamente las que se correspondan con la tabla de la derecha.
- **FULL OUTER JOIN** (Join Completo). De la tabla de la izquierda se muestran todas sus filas tengan o no correspondencia con las de la tabla derecha y de la tabla derecha se muestran todas sus filas tengan o no correspondencia con las de la tabla de la izquierda.

El JOIN externo puede realizarse sobre varias tablas a la vez. En los ejemplos siguientes solo lo vamos a ver sobre 2 tablas para mejor comprensión de su funcionamiento. Aquí tenéis dos ejemplos de JOIN externo múltiple:

- https://es.stackoverflow.com/questions/84506/left-join-con-varias-tablas-y-claúsula-where
- https://www.tutorialesprogramacionya.com/mysqlya/temarios/descripcion.php?cod=58&punto=64&inicio=

Veámoslo con un ejemplo, emplearemos de nuevo las tablas empleados y proyectos.

2.4 LEFT OUTER JOIN

> LEFT OUTER JOIN mostrará todas las filas de la tabla de la izquierda y aquellas que se correspondan de la tabla de la derecha.

Veamos el resultado si hacemos una consulta de todos los empleados y los proyectos en los que trabajan.

SELECT empleados.nombre, proyectos.nombre **AS** proyecto

FROM empleados **LEFT OUTER JOIN** proyectos

ON empleados.codp = proyectos.codproy;

```
mysql>  SELECT EMPLEADOS.NOMBRE,PROYECTOS.NOMBRE AS PROYECTO
    ->  FROM EMPLEADOS LEFT OUTER JOIN PROYECTOS
    ->  ON EMPLEADOS.CODP = PROYECTOS.CODPROY;
+-----------------+--------------------------+
| NOMBRE          | PROYECTO                 |
+-----------------+--------------------------+
| Alberto Gil     | Repsol, S.A.             |
| Mariano Sanz    | NULL                     |
| Ana Silván      | Repsol, S.A.             |
| Roberto Milán   | NULL                     |
| Rafael Colmenar | Consejería de Educación  |
+-----------------+--------------------------+
5 rows in set (0.00 sec)
```

Como se puede observar, Mariano Sanz y Roberto Milán no trabajan para ningún proyecto y sin embargo salen en el resultado por ser un LEFT OUTER JOIN. Por otro lado el proyecto del Oceanográfico en el que no trabaja ningún empleado aún, no aparece.

2.5 RIGHT OUTER JOIN

> RIGHT OUTER JOIN mostrará todas las filas de la tabla de la derecha y aquellas que se correspondan de la tabla de la izquierda.

Igual que hemos hecho antes, veamos la relación entre proyectos y empleados pero empleando el RIGHT OUTER JOIN.

SELECT empleados.nombre, proyectos.nombre **AS** proyecto

FROM empleados **RIGHT OUTER JOIN** proyectos

ON empleados.codp = proyectos.codproy;

Cuyo resultado es:

```
mysql>  SELECT EMPLEADOS.NOMBRE,PROYECTOS.NOMBRE AS PROYECTO
    ->  FROM EMPLEADOS RIGHT OUTER JOIN PROYECTOS
    ->  ON EMPLEADOS.CODP = PROYECTOS.CODPROY;
+-----------------+---------------------------+
| NOMBRE          | PROYECTO                  |
+-----------------+---------------------------+
| Alberto Gil     | Repsol, S.A.              |
| Ana Silván      | Repsol, S.A.              |
| Rafael Colmenar | Consejería de Educación   |
| NULL            | Oceanográfico             |
+-----------------+---------------------------+
4 rows in set (0.03 sec)
```

Observa ahora que aparecen todos los proyectos, incluido el del Oceanográfico aunque nadie trabaja en él; y por parte de los empleados solo aparecen los que trabajan en algún proyecto. Mariano Sanz y Roberto Milán -que no trabajan para ningún proyecto- no aparecen.

2.6 JOIN

Vamos a realizar algunas consultas usando JOIN con las tablas departamentos, empleados y proyectos para practicar.

Recuerda que debes leer el enunciado e intentar hacer la consulta sin mirar la solución. Realiza primero todas las consultas.

2.6.1 Consulta 7

Mostrar el nombre de los proyectos y el nombre de los empleados de los proyectos en los que trabajan empleados del departamento de Informática.

```
mysql>  SELECT PROYECTOS.NOMBRE AS PROYECTO, EMPLEADOS.NOMBRE AS EMPLEADO
    ->  FROM PROYECTOS, EMPLEADOS
    ->  WHERE PROYECTOS.CODPROY = EMPLEADOS.CODP AND EMPLEADOS.DPTO = 'IT'
    ->  ORDER BY PROYECTO;
+---------------------------+-----------------+
| PROYECTO                  | EMPLEADO        |
+---------------------------+-----------------+
| Consejería de Educación   | Rafael Colmenar |
| Repsol, S.A.              | Ana Silván      |
+---------------------------+-----------------+
2 rows in set (0.02 sec)
```

Como puedes ver, el colocar el nombre de las tablas hace que nuestra instrucción sea muy larga, por ello se suelen emplear los alias (como ya comentamos en su momento) para reducir su tamaño de la siguiente forma:

```
mysql>  SELECT P.NOMBRE AS PROYECTO, E.NOMBRE AS EMPLEADO
    ->  FROM PROYECTOS P, EMPLEADOS E
    ->  WHERE P.CODPROY = E.CODP AND E.DPTO = 'IT'
    ->  ORDER BY PROYECTO;
+-------------------------+-----------------+
| PROYECTO                | EMPLEADO        |
+-------------------------+-----------------+
| Consejería de Educación | Rafael Colmenar |
| Repsol, S.A.            | Ana Silván      |
+-------------------------+-----------------+
2 rows in set (0.01 sec)
```

Sería equivalente utilizar INNER JOIN:

SELECT P.NOMBRE AS PROYECTO, E.NOMBRE AS EMPLEADO

FROM PROYECTOS P INNER JOIN EMPLEADOS E

ON P.CODPROY = E.CODP AND E.DPTO = 'IT'

ORDER BY PROYECTO;

2.6.2 Consulta 8

Mostrar todos los proyectos con el nombre del departamento al que pertenecen ordenado por el nombre del proyecto.

```
mysql> select p.nombre,d.nombre
    -> from proyectos p left outer join departamentos d
    -> on p.dpto = d.coddpto
    -> order by p.nombre;
+-------------------------+--------------+
| nombre                  | nombre       |
+-------------------------+--------------+
| Consejería de Educación | Informática  |
| Oceanográfico           | NULL         |
| Repsol, S.A.            | Contabilidad |
+-------------------------+--------------+
3 rows in set (0.00 sec)
```

2.6.3 Consulta 9

Mostrar el nombre de los empleados que trabajan en un proyecto, con el nombre del proyecto y el nombre del departamento al que pertenece el empleado, ordenado por el nombre del empleado.

```
mysql>  SELECT E.NOMBRE, P.NOMBRE, D.NOMBRE
    ->  FROM EMPLEADOS E, PROYECTOS P, DEPARTAMENTOS D
    ->  WHERE E.CODP=P.CODPROY AND E.DPTO = D.CODDPTO
    ->  ORDER BY E.NOMBRE;
+-----------------+--------------------------+---------------+
| NOMBRE          | NOMBRE                   | NOMBRE        |
+-----------------+--------------------------+---------------+
| Alberto Gil     | Repsol, S.A.             | Contabilidad  |
| Ana Silván      | Repsol, S.A.             | Informática   |
| Rafael Colmenar | Consejería de Educación  | Informática   |
+-----------------+--------------------------+---------------+
3 rows in set (0.00 sec)
```

2.6.4 Consulta 10

Mostrar el nombre de los empleados y su fecha de alta, de los empleados de la especialidad de informática que trabajan en un proyecto, mostrando también el nombre del proyecto y todo ello ordenado por la fecha de alta del empleado.

```
mysql>  SELECT E.NOMBRE, E.FECHAALTA, P.NOMBRE
    ->  FROM EMPLEADOS E, PROYECTOS P
    ->  WHERE E.ESPECIALIDAD = "Informática" and E.CODP=P.CODPROY
    ->  ORDER BY E.FECHAALTA;
+-----------------+-------------+--------------------------+
| NOMBRE          | FECHAALTA   | NOMBRE                   |
+-----------------+-------------+--------------------------+
| Ana Silván      | 2012-11-25  | Repsol, S.A.             |
| Rafael Colmenar | 2013-06-10  | Consejería de Educación  |
+-----------------+-------------+--------------------------+
2 rows in set (0.00 sec)
```