CENTRE ESPECÍFIC
D´EDUCACIÓ A DISTÀNCIA DE
LA COMUNITAT VALENCIANA

UNIDAD 6

MODELO FÍSICO DQL

BASES DE DATOS 22/23
CFGS DAW

BOLETÍN DQL. NIVELES BÁSICO, MEDIO Y AVANZADO
SOLUCIONES NIVEL MEDIO

Revisado por:
Sergio Badal, Abelardo Martínez y Pau Miñana
Autores:
Raquel Torres
Paco Aldarias
Fecha: 03/02/23

ÍNDICE DE CONTENIDO

1. SOLUCIÓN Consulta 21....3
2. SOLUCIÓN Consulta 22....3
3. SOLUCIÓN Consulta 23....4
4. SOLUCIÓN Consulta 24....5
5. SOLUCIÓN Consulta 25....5
6. SOLUCIÓN Consulta 26....6
7. SOLUCIÓN Consulta 27....7
8. SOLUCIÓN Consulta 28....8
9. SOLUCIÓN Consulta 29.1....9
10. SOLUCIÓN Consulta 29.2....9
11. SOLUCIÓN Consulta 29.3....10
12. SOLUCIÓN Consulta 29.4....10
13. SOLUCIÓN Consulta 29.5 (cont. 16 básico)....11
14. Anexo 1. Funciones agregadas y group by....13

UD06. BOLETÍN MODELO FÍSICO DQL

1. SOLUCIÓN CONSULTA 21

Mostrar el número de clientes que tenemos en cada ciudad en una columna denominada Num_de_Clientes ordenado por el número de clientes de mayor a menor. Recordad que no debemos ordenar nunca por el alias de la columna, sino por la expresión o función agregada.

```
mysql> SELECT COUNT(*) AS Num_de_Clientes, Ciudad
    -> FROM Clientes
    -> GROUP BY Ciudad
    -> ORDER BY COUNT(*) DESC;
+-----------------+--------------------------+
| Num_de_Clientes | Ciudad                   |
+-----------------+--------------------------+
|              11 | Madrid                   |
|               6 | Fuenlabrada              |
|               2 | Miami                    |
|               2 | Barcelona                |
|               2 | Humanes                  |
|               2 | Paris                    |
|               2 | Sydney                   |
|               1 | San Francisco            |
|               1 | New York                 |
|               1 | San Lorenzo del Escorial |
|               1 | Montornes del valles     |
|               1 | Santa cruz de Tenerife   |
|               1 | Canarias                 |
|               1 | Sotogrande               |
|               1 | Getafe                   |
|               1 | London                   |
+-----------------+--------------------------+
16 rows in set (0,00 sec)
```

2. SOLUCIÓN CONSULTA 22

Mostrar el número de clientes que tenemos en cada ciudad de España en una columna denominada Num_de_Clientes, ordenado por la ciudad.

```
mysql> SELECT COUNT(*) AS Num_de_Clientes, Ciudad
    -> FROM Clientes
    -> WHERE Pais = 'España'
    -> GROUP BY Ciudad
    -> ORDER BY Ciudad;
+-----------------+------------------------+
| Num_de_Clientes | Ciudad                 |
+-----------------+------------------------+
|               2 | Barcelona              |
|               1 | Canarias               |
|               4 | Fuenlabrada            |
|               1 | Getafe                 |
|               2 | Humanes                |
|              10 | Madrid                 |
|               1 | Montornes del valles   |
|               1 | Santa cruz de Tenerife |
|               1 | Sotogrande             |
+-----------------+------------------------+
9 rows in set (0,00 sec)
```

3. SOLUCIÓN CONSULTA 23

Mostrar el número de clientes que tenemos en cada ciudad de España con más de un cliente en una columna denominada Num_de_Clientes, ordenado de mayor a menor por el número de clientes.

Analicemos el enunciado:

- Mostrar el número de clientes → count(*)
- En cada ciudad → agrupado por ciudad
- De España → donde el país es España
- Con más de 1 clientes → número de clientes > 1
- Ordenado de mayor a menor → por número de clientes descendente

Ahora creamos la instrucción SQL que tiene en cuenta todo esto.

```
mysql> SELECT COUNT(*) AS Num_de_Clientes, Ciudad
    -> FROM Clientes
    -> WHERE Pais = 'España'
    -> GROUP BY Ciudad
    -> HAVING COUNT(*) > 1
    -> ORDER BY COUNT(*) DESC;
+-----------------+-------------+
| Num_de_Clientes | Ciudad      |
+-----------------+-------------+
|              10 | Madrid      |
|               4 | Fuenlabrada |
|               2 | Barcelona   |
|               2 | Humanes     |
+-----------------+-------------+
4 rows in set (0,01 sec)
```

4. SOLUCIÓN CONSULTA 24

Mostrar cuál es el beneficio máximo que se puede obtener con la venta de un producto de los que tenemos en Stock en cada una de las gamas que tenemos. Ordena el resultado por el beneficio de mayor a menor.

```
mysql> SELECT MAX(PrecioVenta-PrecioProveedor) AS Beneficio, Gama
    -> FROM Productos
    -> WHERE CantidadEnStock > 0
    -> GROUP BY Gama
    -> ORDER BY MAX(PrecioVenta-PrecioProveedor) DESC;
+-----------+--------------+
| Beneficio | Gama         |
+-----------+--------------+
|     93.00 | Ornamentales |
|     20.00 | Frutales     |
|      3.00 | Herramientas |
|      1.00 | Aromáticas   |
+-----------+--------------+
4 rows in set (0,01 sec)
```

5. SOLUCIÓN CONSULTA 25

Obtener cuántos pedidos ha realizado cada cliente, ordenado por el número de pedidos, de mayor a menor número de pedidos.

```
mysql> SELECT CodigoCliente, COUNT(*) AS Total_Num_Pedidos
    -> FROM Pedidos
    -> GROUP BY CodigoCliente
    -> ORDER BY COUNT(*) DESC;
+---------------+-------------------+
| CodigoCliente | Total_Num_Pedidos |
+---------------+-------------------+
|             1 |                11 |
|            30 |                10 |
|            16 |                10 |
|             3 |                 9 |
|             7 |                 5 |
|             9 |                 5 |
|            13 |                 5 |
|            14 |                 5 |
|            15 |                 5 |
|             5 |                 5 |
|            19 |                 5 |
|            23 |                 5 |
|            26 |                 5 |
|            27 |                 5 |
|            28 |                 5 |
|             4 |                 5 |
|            35 |                 5 |
|            36 |                 5 |
|            38 |                 5 |
+---------------+-------------------+
19 rows in set (0,00 sec)
```

6. SOLUCIÓN CONSULTA 26

Mostrar cuántos pedidos ha rechazado cada uno de nuestros clientes, ordenado descendentemente por el número de rechazo.

```
mysql> SELECT CodigoCliente, COUNT(*) AS Total_Rechazados
    -> FROM Pedidos
    -> WHERE Estado = 'Rechazado'
    -> GROUP BY CodigoCliente
    -> ORDER BY COUNT(*) DESC;
+---------------+------------------+
| CodigoCliente | Total_Rechazados |
+---------------+------------------+
|             1 |                2 |
|             3 |                2 |
|             4 |                2 |
|             5 |                2 |
|            13 |                2 |
|            16 |                2 |
|            30 |                2 |
|            38 |                2 |
|             7 |                1 |
|             9 |                1 |
|            14 |                1 |
|            15 |                1 |
|            19 |                1 |
|            23 |                1 |
|            28 |                1 |
|            36 |                1 |
+---------------+------------------+
16 rows in set (0,00 sec)
```

7. SOLUCIÓN CONSULTA 27

Mostrar el importe total del pedido número 10.

```
mysql> SELECT SUM(Cantidad*PrecioUnidad) AS Total_Pedido_10
    -> FROM DetallePedidos
    -> WHERE CodigoPedido = 10;
+-----------------+
| Total_Pedido_10 |
+-----------------+
|         2920.00 |
+-----------------+
1 row in set (0,00 sec)
```

8. SOLUCIÓN CONSULTA 28

Obtener la máxima cantidad de un producto solicitada en un pedido siempre que ésta sea mayor o igual a 100. Mostrar el resultado ordenado por la Cantidad pedida.

```
mysql> SELECT CodigoProducto, MAX(Cantidad) AS Maximo_Cant_Pedida
    -> FROM DetallePedidos
    -> GROUP BY CodigoProducto
    -> HAVING MAX(Cantidad) >= 100
    -> ORDER BY MAX(Cantidad);
+---------------+--------------------+
| CodigoProducto | Maximo_Cant_Pedida |
+---------------+--------------------+
| AR-002        |                110 |
| OR-157        |                113 |
| FR-29         |                120 |
| FR-48         |                120 |
| 30310         |                143 |
| OR-177        |                150 |
| OR-247        |                150 |
| AR-006        |                180 |
| FR-57         |                203 |
| OR-214        |                212 |
| AR-009        |                290 |
| FR-17         |                423 |
| AR-008        |                450 |
+---------------+--------------------+
13 rows in set (0,00 sec)
```

En la cláusula de ordenación ORDER BY, también podemos hacer referencia al número de orden de la columna en el SELECT. De este modo, esta sentencia sería equivalente:

```
SELECT CodigoProducto, MAX(Cantidad) AS Maximo_Cant_Pedida
FROM DetallePedidos
GROUP BY CodigoProducto
HAVING MAX(Cantidad) >= 100
ORDER BY 2;
```

9. SOLUCIÓN CONSULTA 29.1

Mostrar el código del producto y el importe total pedido de cada producto cuyo importe total esté situado entre los 800 y los 1000 euros ordenado por el total obtenido.

```
mysql> SELECT CodigoProducto, SUM(Cantidad*PrecioUnidad) AS Total_Producto
    -> FROM DetallePedidos
    -> GROUP BY CodigoProducto
    -> HAVING SUM(Cantidad*PrecioUnidad) BETWEEN 800 AND 1000
    -> ORDER BY SUM(Cantidad*PrecioUnidad);
+----------------+----------------+
| CodigoProducto | Total_Producto |
+----------------+----------------+
| OR-225         |         840.00 |
| FR-17          |         846.00 |
| OR-208         |         884.00 |
| FR-79          |         946.00 |
| OR-218         |         950.00 |
| OR-237         |         950.00 |
| FR-29          |         960.00 |
| OR-217         |         975.00 |
| FR-82          |         980.00 |
| AR-009         |         986.00 |
| 22225          |         996.00 |
+----------------+----------------+
11 rows in set (0,01 sec)
```

10. SOLUCIÓN CONSULTA 29.2

Mostrar el código del producto y el importe total pedido de cada producto, de los productos con un precio mayor o igual a 50 euros y menor o igual a 100 y cuyo importe total esté situado entre los 800 y los 1000 euros, ordenado por el código del producto.

```
mysql> SELECT CodigoProducto, SUM(Cantidad*PrecioUnidad) AS Total_Producto
    -> FROM DetallePedidos
    -> WHERE PrecioUnidad BETWEEN 50 AND 100
    -> GROUP BY CodigoProducto
    -> HAVING SUM(Cantidad*PrecioUnidad) BETWEEN 800 AND 1000
    -> ORDER BY CodigoProducto;
+----------------+----------------+
| CodigoProducto | Total_Producto |
+----------------+----------------+
| FR-82          |         980.00 |
| OR-217         |         975.00 |
+----------------+----------------+
2 rows in set (0,00 sec)
```

11. SOLUCIÓN CONSULTA 29.3

Mostrar el código del cliente, su nombre y los números de los pedidos que han realizado los clientes del representante cuyo nombre es Emmanuel.

```
mysql> SELECT C.CodigoCliente, C.NombreCliente, P.CodigoPedido
    -> FROM Empleados E, Clientes C, Pedidos P
    -> WHERE E.CodigoEmpleado = C.CodigoEmpleadoRepVentas
    ->     AND C.CodigoCliente = P.CodigoCliente
    ->     AND E.Nombre = 'Emmanuel'
    -> ORDER BY C.CodigoCliente;
+---------------+---------------+--------------+
| CodigoCliente | NombreCliente | CodigoPedido |
+---------------+---------------+--------------+
|             7 | Beragua       |           13 |
|             7 | Beragua       |           14 |
|             7 | Beragua       |           15 |
|             7 | Beragua       |           16 |
|             7 | Beragua       |           17 |
|             9 | Naturagua     |           18 |
|             9 | Naturagua     |           19 |
|             9 | Naturagua     |           20 |
|             9 | Naturagua     |           21 |
|             9 | Naturagua     |           22 |
+---------------+---------------+--------------+
10 rows in set (0,00 sec)
```

12. SOLUCIÓN CONSULTA 29.4

Mostrar el nombre de los empleados y el número de pedidos realizados por todos sus clientes ordenado de menor a mayor por el número de pedidos.

```
mysql> SELECT E.Nombre, COUNT(P.CodigoPedido) AS Total_Pedidos_Clientes
    -> FROM Empleados E, Clientes C, Pedidos P
    -> WHERE C.CodigoCliente = P.CodigoCliente
    ->     AND E.CodigoEmpleado = C.CodigoEmpleadoRepVentas
    -> GROUP BY E.Nombre
    -> ORDER BY COUNT(P.CodigoPedido);
+-----------------+------------------------+
| Nombre          | Total_Pedidos_Clientes |
+-----------------+------------------------+
| Michael         |                      5 |
| Mariano         |                     10 |
| Lucio           |                     10 |
| Emmanuel        |                     10 |
| José Manuel     |                     10 |
| Lorena          |                     10 |
| Julian          |                     10 |
| Mariko          |                     10 |
| Felipe          |                     20 |
| Walter Santiago |                     20 |
+-----------------+------------------------+
10 rows in set (0,00 sec)
```

13. SOLUCIÓN CONSULTA 29.5 (CONT. 16 BÁSICO)

Mostrar cuál es el beneficio máximo (en una columna denominada Beneficio) que se puede obtener con la venta de un producto de los que tenemos en Stock (si no tiene stock no cuenta). Necesitamos saber también a qué producto pertenece ese beneficio.

a) PRIMERA OPCIÓN

La primera opción que nos viene a la cabeza es ésta, consistente en pedir el nombre y un valor agregado **sin usar GROUP BY**, que nos da error (ver anexo 1 al final de este documento):

```
mysql> SELECT Nombre, MAX(PrecioVenta-PrecioProveedor) AS Beneficio
    -> FROM Productos
    -> WHERE CantidadEnStock > 0;
ERROR 1140 (42000): In aggregated query without GROUP BY, expression #1 of SELECT list contains nonaggregated column 'jardineria.Productos.Nombre'; this is incompatible with sql_mode=only_full_group_by
```

Ahora bien, **en versiones anteriores de MySQL**, esta sentencia nos devolvía un resultado:

```
mysql> select nombre, max(precioventa-precioproveedor) as Beneficio from productos
    -> where cantidadEnStock > 0;
+----------------------+-----------+
| nombre               | Beneficio |
+----------------------+-----------+
| Sierra de Poda 400MM |     93.00 |
+----------------------+-----------+
1 row in set (0.00 sec)
```

Esta opción nos da un resultado IMPREDECIBLE (como explican en este enlace), ya que nos está devolviendo el máximo y un valor arbitrario para "nombre" puesto que la "Sierra de Poda" no es el producto que tiene ese beneficio como podemos ver si hacemos esta otra consulta:

```
mysql> SELECT Nombre, PrecioVenta, PrecioProveedor, (PrecioVenta-PrecioProveedor) AS Beneficio
    -> FROM Productos
    -> WHERE CantidadEnStock > 0
    -> ORDER BY (PrecioVenta-PrecioProveedor) DESC;
+-----------------------------------------------------------+-------------+-----------------+-----------+
| Nombre                                                    | PrecioVenta | PrecioProveedor | Beneficio |
+-----------------------------------------------------------+-------------+-----------------+-----------+
| Trachycarpus Fortunei                                     |      462.00 |          369.00 |     93.00 |
| Bismarckia Nobilis                                        |      266.00 |          212.00 |     54.00 |
```

Pocos SGBD nos permiten usar agregados y campos en el select sin requerir un GROUP BY. En la actualidad MySQL -por defecto- no permite hacerlo, aunque se puede desactivar esta opción y forzar a que se pueda usar (ver anexo 1 al final de este documento). No obstante, **se desaconseja este modo de operar, ya que se devuelven resultados impredecibles**.

b) SEGUNDA OPCIÓN

```
mysql> SELECT Nombre, MAX(PrecioVenta-PrecioProveedor) AS Beneficio
    -> FROM Productos
    -> WHERE CantidadEnStock > 0
    -> GROUP BY Nombre
    -> ORDER BY MAX(PrecioVenta-PrecioProveedor) DESC LIMIT 1;
+----------------------+-----------+
| Nombre               | Beneficio |
+----------------------+-----------+
| Trachycarpus Fortunei |     93.00 |
+----------------------+-----------+
1 row in set (0,00 sec)
```

La opción correcta sería la siguiente, consistente en aplicar un **GROUP BY** y la cláusula **LIMIT 1**, que ofrece solo el primer resultado de todos los posibles.

14. ANEXO 1. FUNCIONES AGREGADAS Y GROUP BY

En SQL estándar podemos usar funciones agregadas (MAX,AVG, etc.) sin tener que poner GROUP BY, pero no todos los SGBD las aceptan.

Con MySQL podemos usar funciones agregadas sin el group by pero nos puede devolver resultados impredecibles (como explican en este enlace).

Si aun así quieres ejecutar una consulta de este tipo, sigue estos pasos:

```
mysql> SELECT Nombre, MAX(PrecioVenta-PrecioProveedor) AS Beneficio
    -> FROM Productos
    -> WHERE CantidadEnStock > 0;
ERROR 1140 (42000): In aggregated query without GROUP BY, expression #1 of SELECT list contains nonaggregated column 'jardineria.Productos.Nombre'; this is incompatible with sql_mode=only_full_group_by
```

Podemos ver que está a on la variable ONLY_FULL_GROUP_BY de MySQL con:

select @@sql_mode;

```
mysql> select @@sql_mode;
+-----------------------------------------------------------------------------------------------------------------------------------------------+
| @@sql_mode                                                                                                                                    |
+-----------------------------------------------------------------------------------------------------------------------------------------------+
| ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION |
+-----------------------------------------------------------------------------------------------------------------------------------------------+
1 row in set (0.00 sec)
```

Vamos a desactivar ONLY_FULL_GROUP_BY con:

SET sql_mode=(SELECT REPLACE(@@sql_mode, 'ONLY_FULL_GROUP_BY', ''));

```
mysql> SET sql_mode=(SELECT REPLACE(@@sql_mode, 'ONLY_FULL_GROUP_BY', ''));
Query OK, 0 rows affected (0.00 sec)
```

Vemos que ya no aparece 'ONLY_FULL_GROUP_BY

```
mysql> select @@sql_mode;
+-----------------------------------------------------------------------------------------------------------------------+
| @@sql_mode                                                                                                            |
+-----------------------------------------------------------------------------------------------------------------------+
| STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION |
+-----------------------------------------------------------------------------------------------------------------------+
1 row in set (0.00 sec)
```

Ahora no da error (AUNQUE EL RESULTADO NO ES CORRECTO):

```
mysql> select nombre, max(precioventa-precioproveedor) as Beneficio from productos
    -> where cantidadEnStock > 0;
+----------------------+-----------+
| nombre               | Beneficio |
+----------------------+-----------+
| Sierra de Poda 400MM |     93.00 |
+----------------------+-----------+
1 row in set (0.00 sec)
```

Más info en este enlace