TEMA 2: REALIZACIÓN DE CONSULTAS BÁSICAS (1ª PARTE)

El lenguaje de manipulación de datos (DML)

OBJETIVOS:

- **Identificar las herramientas y sentencias para realizar consultas.**
- **Realizar consultas simples sobre una tabla**
- **Realizar consultas que generan valores de resumen**
- **Realizar consultas sobre el contenido de varias tablas mediante composiciones internas**
- **Realizar consultas sobre el contenido de varis tablas mediante composiciones externas.**
- **Realizar consultas con subconsultas**
- **Valorar las ventajas e inconvenientes de las distintas opciones válidas para llevar a cabo una consulta determinada**

GESTIÓN DE BASE DE DATOS

IES PUERTO DE LA CRUZ

EL LENGUAJE **DML**

Las sentencias DML del lenguaje SQL son las siguientes:

- La sentencia SELECT, que se utiliza para extraer información de la base de datos, ya sea de una tabla o de varias.
- La sentencia INSERT, cuyo cometido es insertar uno o varios registros en alguna tabla.
- La sentencia DELETE, que borra registros de una tabla.
- La sentencia UPDATE, que modifica registros de una tabla.

Cualquier ejecución de un comando en un SGBD se denomina CONSULTA, término derivado del anglosajón QUERY. Este término debe ser entendido más que como una *consulta* de información, como una *orden*, es decir, las QUERYS o CONSULTAS no son sólo SELECT, sino también cualquier sentencia de tipo UPDATE, INSERT, CREATE, DROP, etc, entendidas todas ellas como peticiones al SGBD para realizar una operación determinada.

La sentencia SELECT

La sentencia SELECT

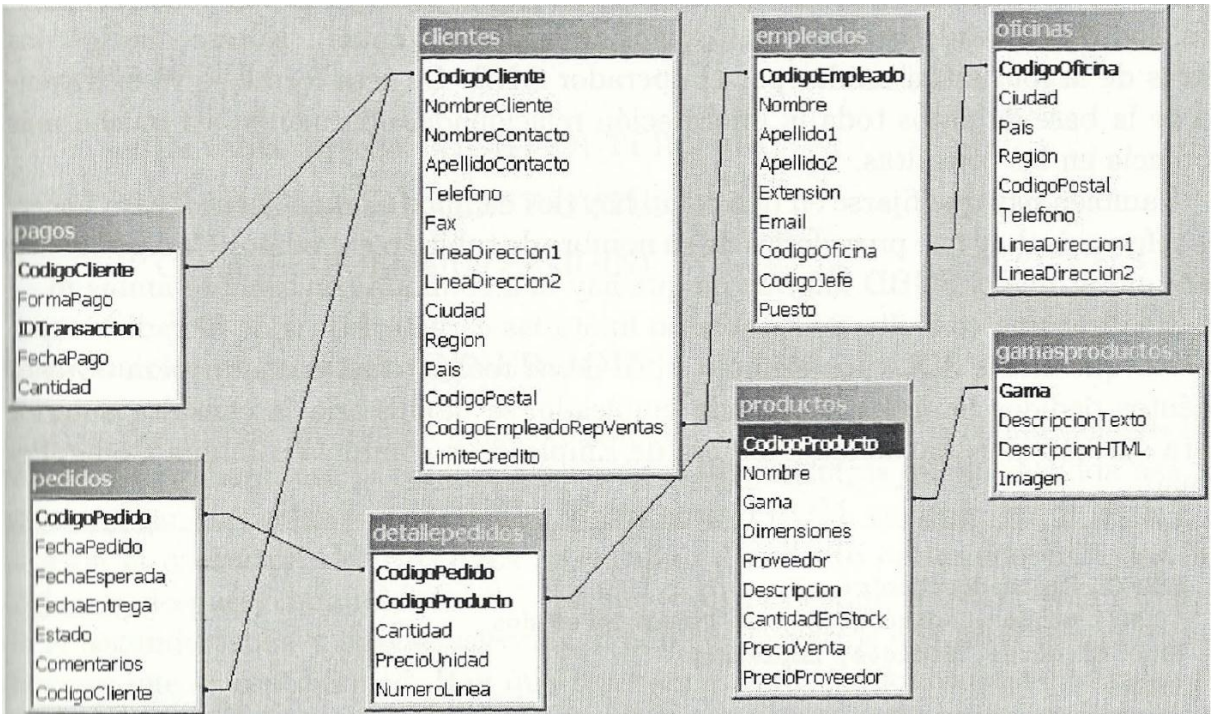
La sentencia SELECT es la sentencia más versátil de todo SQL, y por tanto la más compleja de todas. Como se ha expuesto anteriormente, se utiliza para consultar formación de determinadas tablas. Es posible ejecutar sentencias muy sencillas que muestran todos los registros de una tabla:

```
#esta consulta selecciona todos los campos y muestra todos los
#registros de la tabla clientes
SELECT * FROM clientes;
```

O incluso consultas que obtienen información filtrada de múltiples tablas, usando relaciones entre tablas e incluso tablas virtuales creadas a partir de una consulta.

```
#esta consulta obtiene el total de los pedidos
#de los clientes de una tienda
    SELECT NombreCliente,tot.Cantidad
    FROM Clientes,Pedidos,
(SELECT sum(Cantidad*PrecioUnidad) as Cantidad,NumeroPedido
               FROM DetallePedidos GROUP BY NumeroPedido) tot
    WHERE Clientes.NumeroCliente=Pedidos.NumeroCliente
    AND Pedidos.numeroPedido=tot.NumeroPedido ORDER BY Cantidad;
```

Base de datos de ejemplo: jardineria



Consultas básicas

El formato básico para hacer una consulta es el siguiente:

```
SELECT [DISTINCT] select_expr [,select_expr] ... [FROM tabla]

Select_expr: nombre_columna [AS alias] |*|expresión
```

nombre.columna indica un nombre de columna, es decir, se puede seleccionar de una tabla una serie de columnas, o todas si se usa *, o una expresión algebraica compuesta por operadores, operandos y *funciones.*

El parámetro opcional DISTINCT fuerza que sólo se muestren los registros con valores distintos, o, dicho de otro modo, que suprima las repeticiones.

A continuación se muestran algunos ejemplos del uso de la sentencia SELECT

Mostramos todos los datos de todos los clientes

```
mysql> select * from clientes;
```

Mostramos algunos datos de los clientes

```
mysql> select NombreCliente, Telefono from clientes;
+--------------------------------+----------------+
| NombreCliente                  | Telefono       |
+--------------------------------+----------------+
| DGPRODUCTIONS GARDEN           | 5556901745     |
| Gardening Associates           | 5557410345     |
| Gerudo Valley                  | 5552323129     |
| Tendo Garden                   | 55591233210    |
| Lasas S.A.                     | 34916540145    |
| Beragua                        | 654987321      |
| Club Golf Puerta del hierro    | 62456810       |
| Naturagua                      | 689234750      |
| DaraDistribuciones             | 675598001      |
| Madrile??a de riegos           | 655983045      |
| Lasas S.A.                     | 34916540145    |
| Camunas Jardines S.L.          | 34914873241    |
| Dardena S.A.                   | 34912453217    |
| Jardin de Flores               | 654865643      |
| Flores Marivi                  | 666555444      |
| Flowers, S.A                   | 698754159      |
| Naturajardin                   | 612343529      |
| Golf S.A.                      | 916458762      |
| AYMERICH GOLF MANAGEMENT, SL   | 964493072      |
| Aloha                          | 916485852      |
| El Prat                        | 916882323      |
| Sotogrande                     | 915576622      |
| Vivero Humanes                 | 654987690      |
| Fuenla City                    | 675842139      |
| Jardines y Mansiones CACTUS SL | 916877445      |
| Jardiner?¡as Mat?¡as SL        | 916544147      |
| Agrojardin                     | 675432926      |
| Top Campo                      | 685746512      |
| Jardineria Sara                | 675124537      |
| Campohermoso                   | 645925376      |
| france telecom                 | (33)5120578961 |
| Mus?®e du Louvre               | (33)0140205050 |
| Tutifruti S.A                  | 2 9261-2433    |
| FLORES S.L.                    | 654352981      |
| THE MAGIC GARDEN               | 926523468      |
| El Jardin Viviente S.L         | 2 8005-7161    |
+--------------------------------+----------------+
36 rows in set (0.00 sec)
```

Mostramos algunos datos de los clientes

```
mysql> select nombrecliente, concat(nombrecontacto,apellidocontacto) as Contacto
 from clientes;
+--------------------------------+--------------------+
| nombrecliente                  | Contacto           |
+--------------------------------+--------------------+
| DGPRODUCTIONS GARDEN           | Daniel GGoldFish   |
| Gardening Associates           | AnneWright         |
| Gerudo Valley                  | LinkFlaute         |
| Tendo Garden                   | AkaneTendo         |
| Lasas S.A.                     | AntonioLasas       |
| Beragua                        | JoseBermejo        |
| Club Golf Puerta del hierro    | PacoLopez          |
| Naturagua                      | GuillermoRengifo   |
| DaraDistribuciones             | DavidSerrano       |
| Madrile??a de riegos           | JoseTaca??o        |
| Lasas S.A.                     | AntonioLasas       |
| Camunas Jardines S.L.          | PedroCamunas       |
| Dardena S.A.                   | JuanRodriguez      |
| Jardin de Flores               | JavierVillar       |
| Flores Marivi                  | MariaRodriguez     |
| Flowers, S.A                   | BeatrizFernandez   |
| Naturajardin                   | VictoriaCruz       |
| Golf S.A.                      | LuisMartinez       |
| AYMERICH GOLF MANAGEMENT, SL   | MarioSuarez        |
| Aloha                          | CristianRodrigez   |
| El Prat                        | FranciscoCamacho   |
| Sotogrande                     | MariaSantillana    |
| Vivero Humanes                 | FedericoGomez      |
| Fuenla City                    | TonyMu??oz Mena    |
| Jardines y Mansiones CACTUS SL | Eva Mar?¡aS?ínchez |
| Jardiner?¡as Mat?¡as SL        | Mat?¡asSan Mart?¡n |
| Agrojardin                     | BenitoLopez        |
| Top Campo                      | JoseluisSanchez    |
| Jardineria Sara                | SaraMarquez        |
| Campohermoso                   | LuisJimenez        |
| france telecom                 | Fra?â?ºoisToulou   |
| Mus?®e du Louvre               | PierreDelacroux    |
| Tutifruti S.A                  | JacobJones         |
| FLORES S.L.                    | AntonioRomero      |
| THE MAGIC GARDEN               | RichardMcain       |
| El Jardin Viviente S.L         | JustinSmith        |
+--------------------------------+--------------------+
36 rows in set (0.01 sec)
```

Mostramos algunos datos del detalle de pedidos con cálculo

```
mysql> select CodigoPedido, CodigoProducto, Cantidad, PrecioUnidad, Cantidad*Pre
cioUnidad AS TotalProducto from detallepedidos LIMIT 10;
+--------------+----------------+----------+--------------+---------------+
| CodigoPedido | CodigoProducto | Cantidad | PrecioUnidad | TotalProducto |
+--------------+----------------+----------+--------------+---------------+
|            1 | FR-67          |       10 |        70.00 |        700.00 |
|            1 | OR-127         |       40 |         4.00 |        160.00 |
|            1 | OR-141         |       25 |         4.00 |        100.00 |
|            1 | OR-241         |       15 |        19.00 |        285.00 |
|            1 | OR-99          |       23 |        14.00 |        322.00 |
|            2 | FR-4           |        3 |        29.00 |         87.00 |
|            2 | FR-40          |        7 |         8.00 |         56.00 |
|            2 | OR-140         |       50 |         4.00 |        200.00 |
|            2 | OR-141         |       20 |         5.00 |        100.00 |
|            2 | OR-159         |       12 |         6.00 |         72.00 |
+--------------+----------------+----------+--------------+---------------+
10 rows in set (0.00 sec)
```

Mostramos determinados datos de empleados con/sin DISTINCT

```
mysql> select puesto from empleados;
+----------------------+
| puesto               |
+----------------------+
| Director General     |
| Subdirector Marketing |
| Subdirector Ventas   |
| Secretaria           |
| Representante Ventas |
| Representante Ventas |
| Director Oficina     |
| Representante Ventas |
| Representante Ventas |
| Representante Ventas |
| Director Oficina     |
| Representante Ventas |
| Representante Ventas |
| Representante Ventas |
| Director Oficina     |
| Representante Ventas |
| Representante Ventas |
| Director Oficina     |
| Representante Ventas |
| Director Oficina     |
| Representante Ventas |
| Representante Ventas |
| Director Oficina     |
| Representante Ventas |
| Representante Ventas |
| Director Oficina     |
| Representante Ventas |
| Representante Ventas |
| Director Oficina     |
| Representante Ventas |
| Representante Ventas |
+----------------------+
31 rows in set (0.00 sec)
```

```
mysql> select DISTINCT puesto from empleados;
+----------------------+
| puesto               |
+----------------------+
| Director General     |
| Subdirector Marketing |
| Subdirector Ventas   |
| Secretaria           |
| Representante Ventas |
| Director Oficina     |
+----------------------+
6 rows in set (0.00 sec)
```

Filtros

Los filtros son condiciones que cualquier gestor de base de datos interpreta para seleccionar registros y mostrarlos como resultado de la consulta. En SQL la palabra clave para realizar filtros es la cláusula *WHERE*.

A continuación se añade a la sintaxis de la cláusula SELECT la sintaxis de los filtros:

```
SELECT [DISTINCT] select_expr [,select_expr] ...
[FROM tabla] [WHERE filtro]
```

filtro es una expresión que indica la condición o condiciones que deben satisfacer los registros para ser seleccionados.

Un ejemplo de funcionamiento sería el siguiente:

```
mysql> select Nombre, Apellido1 from empleados where puesto='Representante Venta
s';
+-----------------+------------+
| Nombre          | Apellido1  |
+-----------------+------------+
| Felipe          | Rosas      |
| Juan Carlos     | Ortiz      |
| Mariano         | L??pez     |
| Lucio           | Campoamor  |
| Hilario         | Rodriguez  |
| Jos?® Manuel    | Martinez   |
| David           | Palma      |
| Oscar           | Palma      |
| Lionel          | Narvaez    |
| Laurent         | Serra      |
| Walter Santiago | Sanchez    |
| Marcus          | Paxton     |
| Lorena          | Paxton     |
| Narumi          | Riko       |
| Takuma          | Nomura     |
| Larry           | Westfalls  |
| John            | Walton     |
| Julian          | Bellinelli |
| Mariko          | Kishi      |
+-----------------+------------+
19 rows in set (0.00 sec)
```

Expresiones para filtros

Los filtros se construyen mediante *expresiones*. Una expresión, es una cornbinación de operadores, operandos y funciones que producen un resultado. Por ejemplo una expresión puede ser:

```
mysql> select (2+3)>(6*2);
+-------------+
| (2+3)>(6*2) |
+-------------+
|           0 |
+-------------+
1 row in set (0.00 sec)
```

da 0. 0=false puesto que es falso que 5>12

Se detalla a continuación los elementos que pueden formar parte de las expresiones:

- Operandos: Los operandos pueden ser constantes, por ejemplo el número entero 3, el número real 2.3, la cadena de caracteres 'España' o la fecha '2010-01-02'; también pueden ser variables, por ejemplo el campo *edad* o el campo *Puesto;* y pueden ser también otras expresiones.
- Operadores aritméticos: +, -, *, /, %. El operador + y el operador - se utilizan para sumar o restar dos operandos (binario) o para poner el signo positivo o negativo a un operando (unario). El operador * es la multiplicación de dos operandos y el operador / es para dividir. El operador % o resto de la división entera a%b devuelve el resto de dividir a entre b.

> Todas los operandos numéricos ya sean reales o enteros van sin comilla simple, y cualquier otra cosa que no sea número, por ejemplo, cadenas de caracteres o fechas, van entre comillas simples.

- Operadores relacionales: >, <, <>, >=, <=, =. Los operadores relacionales sirven para comparar dos operandos. Así, es posible preguntar si un campo es mayor que un valor, o si un valor es distinto de otro. Estos operadores devuelven un número entero, de tal manera que si el resultado de la expresión es *cierto* el resultado será 1, y si el resultado es *falso* el resultado será 0. Por ejemplo, la expresión a > b devuelve 1 si a es estrictamente mayor que b y 0 en caso contrario. La expresión d<>e devuelve 1 si d y e son valores distintos.

- Operadores lógicos: AND, OR, NOT. Los operadores lógicos toman como operandos valores lógicos esto es, cierto o falso, en caso de SQL, **1** o **0**. Los operadores lógicos se comportan según las siguientes tablas de verdad:

Operando 1	Operando 2	Op1 AND Op2	Op1 OR Op2	NOT Op1
falso	falso	falso	falso	cierto
falso	cierto	falso	cierto	cierto
cierto	falso	falso	cierto	falso
cierto	cierto	cierto	cierto	falso

Cuadro 1: Tabla de verdad de los operadores lógicos.

Por otro lado, se necesita un tratamiento de los valores nulos; hay que incluir como un posible operando el valor nulo:

Operando 1	Operando 2	Op1 AND Op2	Op1 OR Op2	NOT Op1
falso	falso	falso	falso	cierto
falso	cierto	falso	cierto	cierto
cierto	falso	falso	cierto	falso
cierto	cierto	cierto	cierto	falso
nulo	X	nulo	nulo	nulo
X	nulo	nulo	nulo	no X

Cuadro 2: Tabla de verdad de los operadores lógicos con valores nulos.

- Paréntesis: (). Los operadores tienen una prioridad, por ejemplo, en la expresión 3+4*2, la multiplicación se aplica antes que la suma, se dice que el operador * tiene más prioridad que el operador +. Para alterar esta prioridad, se puede usar el operador paréntesis, cuyo cometido es

precisamente dar máxima prioridad a una parte de una expresión. Así, (3+4)*2, no es lo mismo que 3+4*2.

- Funciones: date_add, concat, left, right .... Cada SGBD incorpora su propio repertorio de funciones que en pocas ocasiones coincide con el de otros SGBD.

En la tabla que se muestra a continuación aparecen los resultados que provoca la ejecución de algunos de los operadores descritos:

Operación	Resultado
7+2*3	13
(7-2)*3	15
7>2	1
9<2	0
7>2 AND 4<3	0
7>2 OR 4<3	1
(10>=10 AND 0<=1)+2	3

Cuadro 3: Tabla resumen de los operadores usados en expresiones.

Construcción de filtros

A continuación, se muestran ejemplos de construcción de filtros para una base de datos, todos ellos compatibles para Oracle y MySQL:

```
mysql> select NombreCliente, Pais from clientes where pais='Spain';
+----------------------+-------+
| NombreCliente        | Pais  |
+----------------------+-------+
| Lasas S.A.           | Spain |
| Lasas S.A.           | Spain |
| Camunas Jardines S.L. | Spain |
| Dardena S.A.         | Spain |
+----------------------+-------+
4 rows in set (0.00 sec)
```

```
mysql> select NombreCliente, Ciudad, Pais
    -> from clientes
    -> where pais='Spain' and
    -> ciudad='Madrid'
    -> ;
+---------------+--------+-------+
| NombreCliente | Ciudad | Pais  |
+---------------+--------+-------+
| Dardena S.A.  | Madrid | Spain |
+---------------+--------+-------+
1 row in set (0.00 sec)
```

```
mysql> select NombreCliente, Ciudad, Pais
    -> from clientes
    -> where pais='Spain' and
    -> ciudad='Madrid' or
    -> ciudad='Fuenlabrada';
+---------------+-------------+---------+
| NombreCliente | Ciudad      | Pais    |
+---------------+-------------+---------+
| Lasas S.A.    | Fuenlabrada | Spain   |
| Lasas S.A.    | Fuenlabrada | Spain   |
| Dardena S.A.  | Madrid      | Spain   |
| Flores Marivi | Fuenlabrada | Espa??a |
| Fuenla City   | Fuenlabrada | Espa??a |
| Campohermoso  | Fuenlabrada | Espa??a |
+---------------+-------------+---------+
6 rows in set (0.00 sec)
```

```
mysql> select NombreCliente, Ciudad, Pais
    -> from clientes
    -> where pais='Spain' and
    -> (ciudad='Madrid' or
    -> ciudad='Fuenlabrada');
+---------------+-------------+-------+
| NombreCliente | Ciudad      | Pais  |
+---------------+-------------+-------+
| Lasas S.A.    | Fuenlabrada | Spain |
| Lasas S.A.    | Fuenlabrada | Spain |
| Dardena S.A.  | Madrid      | Spain |
+---------------+-------------+-------+
3 rows in set (0.00 sec)
```

Ver la diferencia de resultados de las dos últimas dependiendo del uso de paréntesis.

Filtros con operador de pertenencia a conjuntos

Además de los operadores presentados anteriormente (aritméticos, lógicos, etc.) se puede hacer uso del operador de pertenencia a conjuntos IN, cuya sintaxis es la siguiente:

```
nombre_columna IN (Valuel, Value2, ... )
```

Este operador permite comprobar si una columna tiene un valor igual que cualquier de los que están incluidos dentro del paréntesis, así por ejemplo, si se desea seleccionar los clientes del país Spain y de las ciudades Madrid o Fuenlabrada, se codificaría así:

```
mysql> select NombreCliente, Ciudad, Pais
    -> from clientes
    -> where pais='Spain' and
    -> ciudad IN ('Madrid','Fuenlabrada');
+---------------+-------------+-------+
| NombreCliente | Ciudad      | Pais  |
+---------------+-------------+-------+
| Lasas S.A.    | Fuenlabrada | Spain |
| Lasas S.A.    | Fuenlabrada | Spain |
| Dardena S.A.  | Madrid      | Spain |
+---------------+-------------+-------+
3 rows in set (0.00 sec)
```

Filtros con operador de rango

El operador de rango BETWEEN permite seleccionar los registros que estén incluidos en un rango. Su sintaxis es:

nombre_columna BETWEEN Value1 AND Value2

Por ejemplo, para seleccionar los pagos a clientes por cantidades entre 2000 y 3000 se codificaría la siguiente query:

```
mysql> select CodigoCliente, FechaPago, Cantidad
    -> from pagos
    -> where Cantidad BETWEEN 2000 AND 3000;
+---------------+------------+----------+
| CodigoCliente | FechaPago  | Cantidad |
+---------------+------------+----------+
|             1 | 2008-11-10 |  2000.00 |
|             1 | 2008-12-10 |  2000.00 |
|             7 | 2009-01-13 |  2390.00 |
|            13 | 2008-08-04 |  2246.00 |
|            15 | 2009-01-15 |  2081.00 |
+---------------+------------+----------+
5 rows in set (0.00 sec)
```

Que sería equivalente a:

```
mysql> select CodigoCliente, FechaPago, Cantidad
    -> from pagos
    -> where Cantidad>=2000 AND
    -> Cantidad<=3000;
+---------------+------------+----------+
| CodigoCliente | FechaPago  | Cantidad |
+---------------+------------+----------+
|             1 | 2008-11-10 |  2000.00 |
|             1 | 2008-12-10 |  2000.00 |
|             7 | 2009-01-13 |  2390.00 |
|            13 | 2008-08-04 |  2246.00 |
|            15 | 2009-01-15 |  2081.00 |
+---------------+------------+----------+
5 rows in set (0.00 sec)
```

Filtros con test de valor nulo

Los operadores IS e IS NOT permiten verificar si un campo es o no es nulo respectivamente. De esta manera, es posible comprobar, por ejemplo, los empleados que no tienen jefe y los que lo tienen:

```
mysql> select Nombre, Apellido1, Apellido2
    -> from empleados
    -> where CodigoJefe IS NULL;
+--------+-----------+-----------+
| Nombre | Apellido1 | Apellido2 |
+--------+-----------+-----------+
| Marcos | Maga??a   | Perez     |
+--------+-----------+-----------+
1 row in set (0.00 sec)
```

```
mysql> select Nombre, Apellido1, Apellido2
    -> from empleados
    -> where CodigoJefe IS NOT NULL;
+-----------------+------------+-----------+
| Nombre          | Apellido1  | Apellido2 |
+-----------------+------------+-----------+
| Ruben           | L??pez     | Martinez  |
| Alberto         | Soria      | Carrasco  |
| Maria           | Sol?¡s     | Jerez     |
| Felipe          | Rosas      | Marquez   |
| Juan Carlos     | Ortiz      | Serrano   |
| Carlos          | Soria      | Jimenez   |
| Mariano         | L??pez     | Murcia    |
| Lucio           | Campoamor  | Mart?¡n   |
| Hilario         | Rodriguez  | Huertas   |
| Emmanuel        | Maga??a    | Perez     |
| Jos?© Manuel    | Martinez   | De la Osa |
| David           | Palma      | Aceituno  |
| Oscar           | Palma      | Aceituno  |
| Francois        | Fignon     |           |
| Lionel          | Narvaez    |           |
| Laurent         | Serra      |           |
| Michael         | Bolton     |           |
| Walter Santiago | Sanchez    | Lopez     |
| Hilary          | Washington |           |
| Marcus          | Paxton     |           |
| Lorena          | Paxton     |           |
| Nei             | Nishikori  |           |
| Narumi          | Riko       |           |
| Takuma          | Nomura     |           |
| Amy             | Johnson    |           |
| Larry           | Westfalls  |           |
| John            | Walton     |           |
| Kevin           | Fallmer    |           |
| Julian          | Bellinelli |           |
| Mariko          | Kishi      |           |
+-----------------+------------+-----------+
30 rows in set (0.00 sec)
```

Filtros con test de patrón

Los filtros con test patrón seleccionan los registros que cumplan una serie de características. Se pueden usar los caracteres comodines % y _ para buscar una cadena de caracteres. Por ejemplo, seleccionar de la tabla de clientes aquellos que su código postal empiece por 28:

```
mysql> select NombreCliente, CodigoPostal
    -> from clientes
    -> where CodigoPostal Like '28%';
+---------------------------+--------------+
| NombreCliente             | CodigoPostal |
+---------------------------+--------------+
| Lasas S.A.                | 28945        |
| Beragua                   | 28942        |
| Club Golf Puerta del hierro | 28930      |
| Naturagua                 | 28947        |
| DaraDistribuciones        | 28946        |
| Madrile??a de riegos      | 28943        |
| Lasas S.A.                | 28945        |
| Camunas Jardines S.L.     | 28145        |
| Dardena S.A.              | 28003        |
| Jardin de Flores          | 28950        |
| Flores Marivi             | 28945        |
| Naturajardin              | 28011        |
| Vivero Humanes            | 28970        |
| Fuenla City               | 28574        |
| Agrojardin                | 28904        |
| Top Campo                 | 28574        |
| Campohermoso              | 28945        |
+---------------------------+--------------+
17 rows in set (0.00 sec)
```

Clientes cuyo nombre contenga el texto o la palabra ‘jardin’:

```
mysql> select NombreCliente
    -> from clientes
    -> where NombreCliente Like '%jardin%';
+-------------------------------+
| NombreCliente                 |
+-------------------------------+
| Camunas Jardines S.L.         |
| Jardin de Flores              |
| Naturajardin                  |
| Jardines y Mansiones CACTUS SL |
| Jardiner?¡as Mat?¡as SL       |
| Agrojardin                    |
| Jardineria Sara               |
| El Jardin Viviente S.L        |
+-------------------------------+
8 rows in set (0.00 sec)
```

El carácter comodín % busca coincidencias de cualquier número de caracteres, incluso cero caracteres. El carácter comodín _ busca coincidencias de exactamente un carácter.

Para ilustrar el funcionamiento del carácter _ , se consultan aquellos clientes que contengan la palabra “Jardin” en 4 posición.

```
mysql> select NombreCliente
    -> from clientes
    -> where NombreCliente Like '___Jardin%';
+------------------------+
| NombreCliente          |
+------------------------+
| El Jardin Viviente S.L |
+------------------------+
1 row in set (0.00 sec)
```

Filtros por límite de número de registros

Este tipo de filtros no es estándar y su funcionamiento varía con el SGBD. Consiste en limitar el número de registros devuelto por una consulta. En MySQL, sintaxis es:

```
[LIMIT [desplazamiento,] nfilas]
```

nfilas especifica el número de filas a devolver y *desplazamiento* especifica a partir de qué fila se empieza a contar (desplazamiento).

```
mysql> select CodigoPedido, FechaPedido
    -> from pedidos
    -> where FechaEntrega IS NULL
    -> LIMIT 5;
+--------------+-------------+
| CodigoPedido | FechaPedido |
+--------------+-------------+
|            3 | 2008-06-20  |
|            4 | 2009-01-20  |
|           10 | 2009-01-15  |
|           11 | 2009-01-20  |
|           12 | 2009-01-22  |
+--------------+-------------+
5 rows in set (0.00 sec)
```

```
mysql> select CodigoPedido, FechaPedido
    -> from pedidos
    -> where FechaEntrega IS NULL
    -> LIMIT 5,5;
+--------------+-------------+
| CodigoPedido | FechaPedido |
+--------------+-------------+
|           14 | 2009-01-02  |
|           19 | 2009-01-18  |
|           20 | 2009-01-20  |
|           23 | 2008-12-30  |
|           25 | 2009-02-02  |
+--------------+-------------+
5 rows in set (0.00 sec)
```

Oracle limita el número de filas apoyándose en una pseudo columna, de nombre rownum:

```
--Saca los 25 primeros jugadores
SELECT * FROM jugadores WHERE rownum <= 25;
```

Ordenación

Para mostrar ordenados un conjunto de registros se utiliza la cláusula *ORDER BY* de la sentencia SELECT.

SELECT [DISTINCT] select_expr [,select_expr] ...
[FROM tabla]
[WHERE filtro]
[ORDER BY {nombre_columna | expr | posición} [ASC | DESC] , ...]

Esta cláusula permite ordenar el conjunto de resultados de forma ascendente (ASC) o descendente (DESC) por una o varias columnas. Si no se indica ASC o DESC por defecto es ASC. La columna por la que se quiere ordenar se puede expresar por el nombre de la columna, una expresión o bien la posición numérica del campo que se quiere ordenar. Por ejemplo:

```
mysql> select CodigoOficina, Ciudad, Pais from oficinas order by Ciudad;
+---------------+----------------------+------------+
| CodigoOficina | Ciudad               | Pais       |
+---------------+----------------------+------------+
| BCN-ES        | Barcelona            | Espa??a    |
| BOS-USA       | Boston               | EEUU       |
| LON-UK        | Londres              | Inglaterra |
| MAD-ES        | Madrid               | Espa??a    |
| PAR-FR        | Paris                | Francia    |
| SFC-USA       | San Francisco        | EEUU       |
| SYD-AU        | Sydney               | Australia  |
| TAL-ES        | Talavera de la Reina | Espa??a    |
| TOK-JP        | Tokyo                | Jap??n     |
+---------------+----------------------+------------+
9 rows in set (0.00 sec)
```

```
mysql> select CodigoOficina, Pais, Ciudad
    -> from oficinas
    -> order by Pais ASC, Ciudad DESC;
+---------------+------------+----------------------+
| CodigoOficina | Pais       | Ciudad               |
+---------------+------------+----------------------+
| SYD-AU        | Australia  | Sydney               |
| SFC-USA       | EEUU       | San Francisco        |
| BOS-USA       | EEUU       | Boston               |
| TAL-ES        | Espa??a    | Talavera de la Reina |
| MAD-ES        | Espa??a    | Madrid               |
| BCN-ES        | Espa??a    | Barcelona            |
| PAR-FR        | Francia    | Paris                |
| LON-UK        | Inglaterra | Londres              |
| TOK-JP        | Jap??n     | Tokyo                |
+---------------+------------+----------------------+
9 rows in set (0.00 sec)
```

Consultas de resumen

En SQL se pueden generar consultas más complejas que resuman cierta información, extrayendo información calculada de varios conjuntos de registros. Un ejemplo de consulta resumen seria la siguiente:

```
mysql> select count(*) from empleados;
+----------+
| count(*) |
+----------+
|       31 |
+----------+
1 row in set (0.00 sec)
```

Esta consulta devuelve el número de registros de la tabla empleados, es decir, se genera un resumen de la información contenida en la tabla empleados. La expresión count(*) es una función que toma como entrada los registros de la tabla consultada y cuenta cuántos registros hay. El resultado de la función count es un único valor (1 fila, 1 columna) con el número 31 (número de registros de la tabla empleados).

Para poder generar información resumida hay que hacer uso de las *funciones de columna*. Estas funciones de columna convierten un conjunto de registros en una información simple cuyo resultado es un cálculo. A continuación, se expone una lista de las funciones de columna disponibles en SQL:

SUM (Expresión)	Suma los valores indicados en el argumento
AVG (Expresión)	Calcula la media de los valores
MIN (Expresión)	Calcula el mínimo
MAX (Expresión)	Calcula el máximo
COUNT (nbColumna)	Cuenta el número de valores de una columna (excepto los nulos)
COUNT (*)	Cuenta el número de valores de una fila (Incluyendo los nulos).

A modo de ejemplo, se muestran algunas consultas resúmenes:

```
mysql> select max(Cantidad) from pagos;
+---------------+
| max(Cantidad) |
+---------------+
|      23794.00 |
+---------------+
1 row in set (0.00 sec)
```

```
mysql> select min(Cantidad) from pagos;
+---------------+
| min(Cantidad) |
+---------------+
|        232.00 |
+---------------+
1 row in set (0.00 sec)
```

```
mysql> select avg(Cantidad) from pagos;
+---------------+
| avg(Cantidad) |
+---------------+
|   7613.076923 |
+---------------+
1 row in set (0.01 sec)
```

```
mysql> select count(*) from empleados;
+----------+
| count(*) |
+----------+
|       31 |
+----------+
1 row in set (0.00 sec)

mysql> select count(CodigoJefe) from empleados;
+-------------------+
| count(CodigoJefe) |
+-------------------+
|                30 |
+-------------------+
1 row in set (0.00 sec)
```

Con las consultas de resumen se pueden realizar *agrupaciones* de registros. Se denomina agrupación de registros a un conjunto de registros que cumplen que tienen una o varias columnas con el mismo valor. Por ejemplo, en la tabla empleados:

```
mysql> select CodigoEmpleado, Puesto from empleados;
+----------------+-----------------------+
| CodigoEmpleado | Puesto                |
+----------------+-----------------------+
|              1 | Director General      |
|              2 | Subdirector Marketing |
|              3 | Subdirector Ventas    |
|              4 | Secretaria            |
|              5 | Representante Ventas  |
|              6 | Representante Ventas  |
|              7 | Director Oficina      |
|              8 | Representante Ventas  |
|              9 | Representante Ventas  |
|             10 | Representante Ventas  |
|             11 | Director Oficina      |
|             12 | Representante Ventas  |
|             13 | Representante Ventas  |
|             14 | Representante Ventas  |
|             15 | Director Oficina      |
|             16 | Representante Ventas  |
|             17 | Representante Ventas  |
|             18 | Director Oficina      |
|             19 | Representante Ventas  |
|             20 | Director Oficina      |
|             21 | Representante Ventas  |
|             22 | Representante Ventas  |
|             23 | Director Oficina      |
|             24 | Representante Ventas  |
|             25 | Representante Ventas  |
|             26 | Director Oficina      |
|             27 | Representante Ventas  |
|             28 | Representante Ventas  |
|             29 | Director Oficina      |
|             30 | Representante Ventas  |
|             31 | Representante Ventas  |
+----------------+-----------------------+
31 rows in set (0.00 sec)
```

En esta consulta hay ocho registros cuyo Puesto='Director Oficina'. Se puede agrupar estos registros formando un único grupo, de tal manera que cada grupo contiene a los empleados que desempeñan el mismo puesto. A este grupo de registros se le puede aplicar una función de columna para realizar determinados cálculos, por ejemplo, contarlos:

```
mysql> select Puesto, count(*) AS empleados
    -> from empleados
    -> group by Puesto;
+----------------------+-----------+
| Puesto               | empleados |
+----------------------+-----------+
| Director General     |         1 |
| Director Oficina     |         8 |
| Representante Ventas |        19 |
| Secretaria           |         1 |
| Subdirector Marketing |         1 |
| Subdirector Ventas   |         1 |
+----------------------+-----------+
6 rows in set (0.00 sec)
```

En este caso, si se agrupa (GROUP BY) por el campo puesto, salen 6 grupos. La función de columna, cuando se agrupa por un campo, actúa para cada grupo. En este caso, para cada grupo se ha contado el número de registros que tiene.

La sintaxis para la sentencia SELECT con GROUP BY queda como sigue:

SELECT [DISTINCT] select_expr [,select_expr] ...
[FROM tabla]
[WHERE filtro]
[GROUP BY expr [expr]....]
[ORDER BY {nombre_columna | expr | posición} [ASC | DESC] , ...]

Se observa que GROUP BY va justo antes de la cláusula ORDER BY. A continuación, a modo de ejemplo, se muestran algunas consultas con grupos y funciones de columna.

Máxima cantidad pagada por cada cliente

```
mysql> select CodigoCliente, max(Cantidad)
    -> from pagos
    -> group by CodigoCliente;
+---------------+---------------+
| CodigoCliente | max(Cantidad) |
+---------------+---------------+
|             1 |       2000.00 |
|             3 |       5000.00 |
|             4 |      20000.00 |
|             5 |      23794.00 |
|             7 |       2390.00 |
|             9 |        929.00 |
|            13 |       2246.00 |
|            14 |       4160.00 |
|            15 |      10000.00 |
|            16 |       4399.00 |
|            19 |        232.00 |
|            23 |        272.00 |
|            26 |      18846.00 |
|            27 |      10972.00 |
|            28 |       8489.00 |
|            30 |       7863.00 |
|            35 |       3321.00 |
|            38 |       1171.00 |
+---------------+---------------+
18 rows in set (0.00 sec)
```

Cantidades totales pagadas por cada cliente

```
mysql> select CodigoCliente, sum(Cantidad)
    -> from pagos
    -> group by CodigoCliente;
+---------------+---------------+
| CodigoCliente | sum(Cantidad) |
+---------------+---------------+
|             1 |       4000.00 |
|             3 |      10926.00 |
|             4 |      81849.00 |
|             5 |      23794.00 |
|             7 |       2390.00 |
|             9 |        929.00 |
|            13 |       2246.00 |
|            14 |       4160.00 |
|            15 |      12081.00 |
|            16 |       4399.00 |
|            19 |        232.00 |
|            23 |        272.00 |
|            26 |      18846.00 |
|            27 |      10972.00 |
|            28 |       8489.00 |
|            30 |       7863.00 |
|            35 |       3321.00 |
|            38 |       1171.00 |
+---------------+---------------+
18 rows in set (0.00 sec)
```

¿Cuántos empleados representantes de ventas hay en cada oficina?

```
mysql> select CodigoOficina, COUNT(*)
    -> from empleados
    -> WHERE Puesto='Representante Ventas'
    -> group by CodigoOficina;
+---------------+----------+
| CodigoOficina | COUNT(*) |
+---------------+----------+
| BCN-ES        |        3 |
| BOS-USA       |        2 |
| LON-UK        |        2 |
| MAD-ES        |        3 |
| PAR-FR        |        2 |
| SFC-USA       |        1 |
| SYD-AU        |        2 |
| TAL-ES        |        2 |
| TOK-JP        |        2 |
+---------------+----------+
9 rows in set (0.00 sec)
```

IMPORTANTE: Se observa que para cada agrupación, se ha seleccionado también el nombre de la columna por la cual se agrupa.