EJERCICIOS RESUELTOS Y COMENTADOS PARA EL APRENDIZAJE DE LA INSTRUCCIÓN SELECT

1. Conociendo la base de datos.

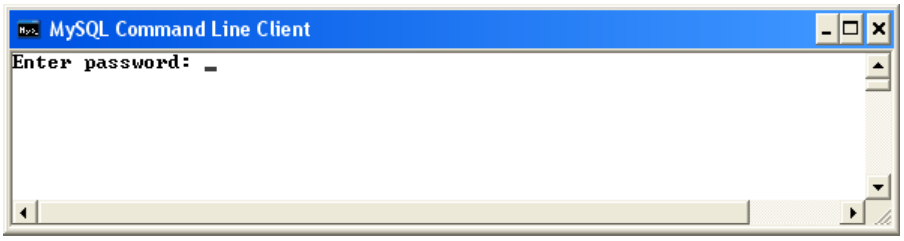
En primer lugar, antes de empezar a estudiar la instrucción SELECT exploraremos la base de datos.

Podemos trabajar de dos formas:

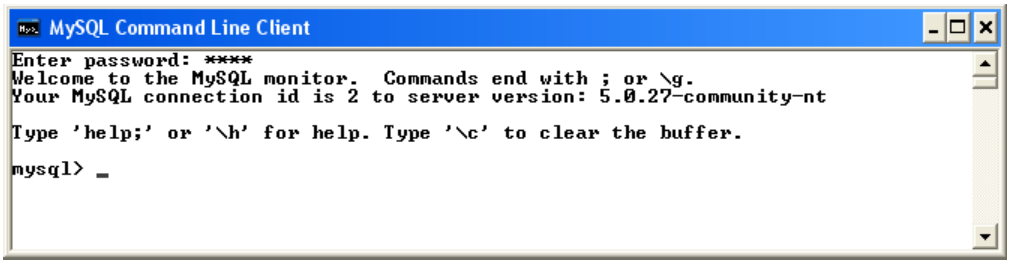
1) Con el cliente mysql a través de la línea de comando.
2) O con la herramienta MySQL Query Browser.

1.1. Conexión con la línea de comando.

Si ejecutamos desde Programa/MySQL/MySQL Server 5.0/MySQL Command Line Client aparecerá

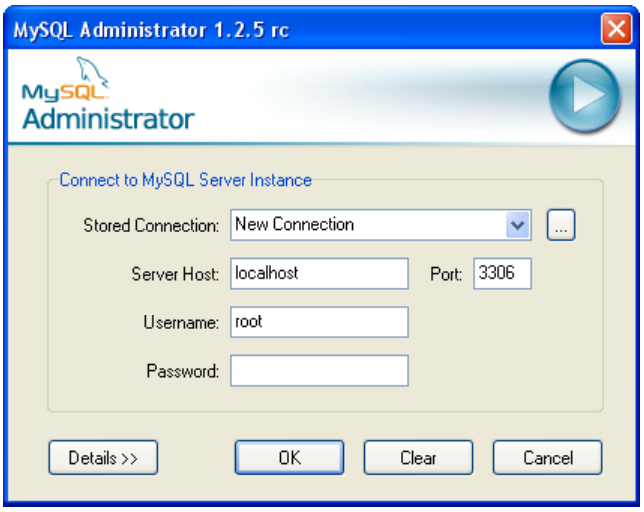


Y después de introducir el password

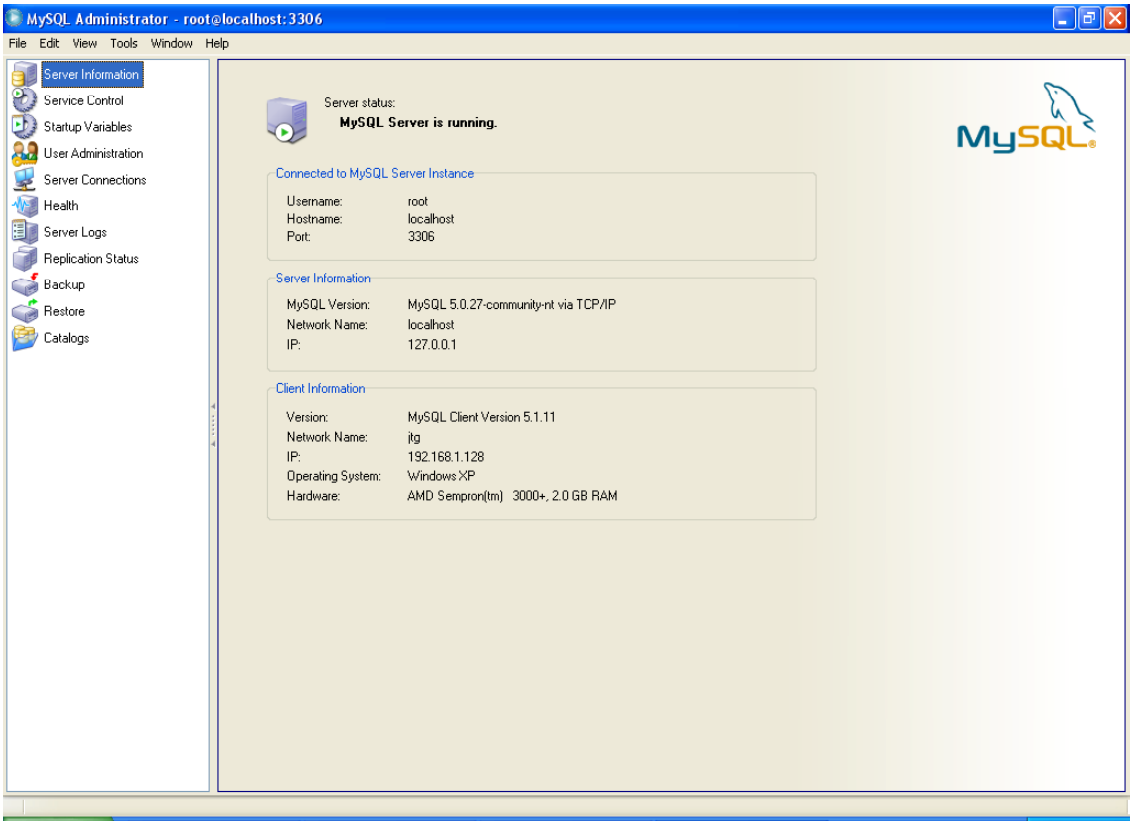


1.2. Conexión con el Administrador.

Si ejecutamos desde Programa/MySQL/MySQL Administrador aparecerá



Si nos identificamos y hacemos OK aparecerá



Desde la opción Tools/MySQL Command Line podemos ir a la línea de comando. También, desde dicha opción, podemos ir a la utilidad MySQL Query Browser, desde donde podemos seguir todo el ejercicio sin mas que ejecutar la sentencias una a una, con la ventaja de disponer de una interfaz más amigable. En estos apuntes utilizaremos la Línea de Comando para no hacer tan extenso los propios apuntes.

1.3. MySQL Migration Toolkit.

1.4. Sentencia USE.

Para acceder a la base de datos utilizamos la sentencia USE.

```
mysql> USE Empresa;
Database changed
```

Recibimos un mensaje "Database changed" (base de datos cambiada). Una base de datos está formada como ya sabemos por tablas.

1.5. Sentencia SHOW TABLES.

Para ver qué tablas tenemos escribimos la sentencia SHOW TABLES.

```
mysql> SHOW TABLES;
+-------------------+
| Tables_in_empresa |
+-------------------+
| articulos         |
| clientes          |
| conformes         |
| empleados         |
| facturas          |
| grupos            |
| lineas_factura    |
| sucursales        |
| ventas            |
+-------------------+
9 rows in set (0.03 sec)
```

Para ver qué contiene la tabla sucursales, o seleccionar todas sus filas escribimos nuestra primera instrucción SELECT :

```
mysql> select * from sucursales;
+----+-------------+
| id | descripcion |
+----+-------------+
|  1 | Centro      |
|  2 | Unión       |
|  3 | Marvín      |
+----+-------------+
3 rows in set (0.00 sec)
```

Esta tabla está formada por dos columnas: Id y Descripcion. Los nombres de columnas tampoco pueden llevar tildes. Debajo de los nombres están los valores. Tenemos una fila, por ejemplo, donde Id vale 2 y Descripcion es "Unión". Para averiguar más sobre esta tabla escribimos:

1.6. Sentencia DESCRIBE.

```
mysql> describe sucursales;
+-------------+------------------+------+-----+---------+----------------+
| Field       | Type             | Null | Key | Default | Extra          |
+-------------+------------------+------+-----+---------+----------------+
| id          | int(10) unsigned | NO   | PRI | NULL    | auto_increment |
| descripcion | varchar(15)      | NO   |     |         |                |
+-------------+------------------+------+-----+---------+----------------+
2 rows in set (0.01 sec)
```

La instrucción "DESCRIBE sucursales" describe la definición de la estructura de latabla.

El varchar(15) que se encuentra en la 3ª fila, 2ºcolumna significa que la columna Descripcion puede tener un número variable de caracteres, hasta 15. Análogamente el int(10) que se encuentra en la 2ª fila, 2ª columna indica que Id es un número entero.

La palabra PRI que se encuentra en la 2ªfila, 4ª columna significa que Id caracteriza de manera única una fila. Es decir, cada fila de sucursales va a tener un número que va a estar en la columna Id. Diferentes filas tendrán diferentes números. De manera que un valor de Id, si está en la tabla, determina una fila. PRI es una abreviatura de PRIMARY KEY , clave primaria. Esta es una expresión usada en computación para indicar una columna que se usa para identificar las filas.

Aún podríamos tener mas información de la tabla si ejecutamos la sentencia:

```
mysql> show create table sucursales;
+------------+----------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
-------------------------+
| Table      | Create Table


                         |
+------------+----------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
-------------------------+
| sucursales | CREATE TABLE `sucursales` (
  `id` int(10) unsigned NOT NULL auto_increment,
  `descripcion` varchar(15) collate latin1_spanish_ci NOT NULL default '',
  PRIMARY KEY  (`id`)
) ENGINE=MyISAM AUTO_INCREMENT=4 DEFAULT CHARSET=latin1 COLLATE=latin1_spanish_ci
|
+------------+----------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
-------------------------+
1 row in set (0.02 sec)
```

Se sugiere al alumnado que explore, usando SELECT * FROM ... y DESCRIBE ..., sobre las diversas tablas de la base de datos empresa.

1.7. Tipos de datos

En una tabla cada columna es de un tipo de datos. Para nuestros fines hay 3 grandes tipos de datos: numéricos, cadenas de caracteres y fechas.

Los datos numéricos pueden ser enteros (int) o decimales (decimal). Los decimales tienen una parte entera y una fraccionaria. Por ejemplo, un decimal puede valer 120.40.

Las cadenas de caracteres (varchar) representan texto, generalmente con un largo variable, hasta un máximo dado. Pueden contener letras, números y otros caracteres como '=','+', etc.

Ya vimos un ejemplo con la columna Descripcion de la tabla sucursales. Las columnas de tipo fecha tienen, en su descripción, la palabra "date". Cualquier columna puede, en principio, tomar el valor NULL, que representa un valor desconocido o inexistente. Sin embargo, cuando se crearon las tablas el administrador de la base de datos normalmente especificó que la mayoría de las columnas no pueden tomar valores NULL. Las que sí pueden aparecen con la palabra YES en la columna NULL de la tabla que se obtiene haciendo "DESCRIBE nombre_de_tabla" . Vemos que por ejemplo en la tabla Sucursales la base de datos no aceptará un valor NULL ni en Id ni en Descripción.

2. Sentencia SELECT

Para extraer información de una base de datos, utilizamos una consulta a través de una instrucción SELECT.

Para presentar los valores de una columna determinada de una tabla, usamos

SELECT Nombre_de_columna FROM Nombre_de_tabla;

Ejemplo :

```
mysql> SELECT descripcion FROM sucursales;
+-------------+
| descripcion |
+-------------+
| Centro      |
| Unión       |
| Marvín      |
+-------------+
3 rows in set (0.00 sec)
```

Para seleccionar más de una columna,

SELECT Nombre_de_columna_1, Nombre_de_columna_2,... FROM Nombre_de_tabla;

Ejemplo:

```
mysql> SELECT nombre, direccion FROM clientes;
+-------------+---------------------+
| nombre      | direccion           |
+-------------+---------------------+
| Matt Design | NULL                |
| Diana Pérez | Brito del Pino 1120 |
| John Smith  | Zum Felde 2024      |
+-------------+---------------------+
3 rows in set (0.00 sec)
```

Para seleccionar todas las columnas de una tabla,

SELECT * FROM Nombre_de_tabla;

Ejemplo:

```
mysql> SELECT * FROM ventas;
+----+---------+----------+---------+----------+----------+----------+------------+
| id | cliente | vendedor | importe | articulo | cantidad | sucursal | fecha      |
+----+---------+----------+---------+----------+----------+----------+------------+
|  1 |       1 |        1 |  178.50 |        1 |        1 |        1 | 2004-10-01 |
|  2 |       2 |        1 |  195.00 |        2 |        1 |        2 | 2004-10-05 |
|  3 |       3 |        1 |  178.50 |        1 |        1 |     NULL | 2004-10-07 |
|  4 |       1 |        1 |   16.00 |        7 |        2 |        2 | 2004-11-01 |
|  5 |       1 |        2 |  220.00 |        3 |        1 |        1 | 2004-12-02 |
|  6 |       1 |        1 |    5.50 |        5 |        1 |        1 | 2004-10-02 |
|  7 |       1 |        2 |   11.00 |        5 |        2 |        1 | 2004-10-03 |
|  8 |       1 |        2 |    5.50 |        5 |        1 |        1 | 2005-01-01 |
|  9 |       2 |        1 |  200.00 |        2 |        1 |        2 | 2004-10-12 |
+----+---------+----------+---------+----------+----------+----------+------------+
9 rows in set (0.00 sec)
```

Obsérvese que los clientes, vendedores, artículos y sucursales son identificados por su Id. El importe representa el total de la venta, no el precio unitario. También se pueden hacer cálculos con una columna numérica.

Ejemplos:

```
mysql> SELECT importe, 0.10*importe FROM ventas;
+---------+--------------+
| importe | 0.10*importe |
+---------+--------------+
|  178.50 |      17.8500 |
|  195.00 |      19.5000 |
|  178.50 |      17.8500 |
|   16.00 |       1.6000 |
|  220.00 |      22.0000 |
|    5.50 |       0.5500 |
|   11.00 |       1.1000 |
|    5.50 |       0.5500 |
|  200.00 |      20.0000 |
+---------+--------------+
9 rows in set (0.00 sec)
```

Tal como puede verse la expresión 0.10*importe constituye lo que se denomina un campo calculado que se obtiene a partir de un campo/columna de la tabla.

Se pueden agregar a la salida columnas constantes o calculadas, que no tienen o (no tienen) relación con las tablas.

Ejemplo:

```
mysql> SELECT "Descuento: ", 0.05*importe FROM ventas;
+-------------+--------------+
| Descuento:  | 0.05*importe |
+-------------+--------------+
| Descuento:  |       8.9250 |
| Descuento:  |       9.7500 |
| Descuento:  |       8.9250 |
| Descuento:  |       0.8000 |
| Descuento:  |      11.0000 |
| Descuento:  |       0.2750 |
| Descuento:  |       0.5500 |
| Descuento:  |       0.2750 |
| Descuento:  |      10.0000 |
+-------------+--------------+
9 rows in set (0.00 sec)
```

En general es deseable no tener títulos de columna complicados como "0.10*Importe". Para esos casos se usa la instrucción AS (que significa, entre otras cosas, "como").

Ejemplo:

```
mysql> SELECT 0.05*importe AS 'Descuento: ' FROM ventas;
+-------------+
| Descuento:  |
+-------------+
|      8.9250 |
|      9.7500 |
|      8.9250 |
|      0.8000 |
|     11.0000 |
|      0.2750 |
|      0.5500 |
|      0.2750 |
|     10.0000 |
+-------------+
9 rows in set (0.00 sec)
```

Se puede solicitar que no haya repeticiones en las filas seleccionadas. Basta agregar DISTINCT después del SELECT.

Ejemplo:

```
mysql> SELECT DISTINCT cliente FROM ventas;
+---------+
| cliente |
+---------+
|       1 |
|       2 |
|       3 |
+---------+
3 rows in set (0.00 sec)
```

Los resultados de cálculos, si no se toma alguna medida al respecto, a veces salen con demasiados decimales. En esos casos, conviene usar la función ROUND. Esta función redondea los números a la cantidad deseada de decimales después del punto. Por ejemplo, ROUND(2.343,2) produce el resultado 2.34. Análogamente ROUND(2.347,2) produce 2.35, y ROUND(2.245,2) produce 2.34. Para redondear a un entero alcanza con escribir ROUND(número). Por ejemplo, ROUND(2.345) produce 2.

Ejemplo:

```
mysql> SELECT ROUND(importe/3,2) FROM ventas;
+--------------------+
| ROUND(importe/3,2) |
+--------------------+
|              59.50 |
|              65.00 |
|              59.50 |
|               5.33 |
|              73.33 |
|               1.83 |
|               3.67 |
|               1.83 |
|              66.67 |
+--------------------+
9 rows in set (0.02 sec)
```

En el último punto hemos visto por primera vez una función. Una función consta de un nombre, seguido de paréntesis, entre los cuales según los casos puede no haber nada, o haber una o más variables. La función realiza ciertos cálculos con los valores de dichas variables, y genera otra cantidad, llamada el valor de la función. Una consideración práctica es que no se pueden dejar espacios entre el nombre de la función y el primer paréntesis después del mismo. Siempre escribiremos paréntesis después del nombre de una función.

2.1. Cláusula WHERE.

Esta cláusula sirve para seleccionar filas, dentro de las columnas seleccionadas. WHERE significa "donde". Se pueden seleccionar filas donde una columna tiene un valor determinado.

Ejemplo:

```
mysql> SELECT * FROM clientes WHERE id=1;
+----+-------------+-----------+----------+
| id | nombre      | direccion | vendedor |
+----+-------------+-----------+----------+
|  1 | Matt Design | NULL      |        1 |
+----+-------------+-----------+----------+
1 row in set (0.00 sec)
```

Vemos que de entre todos los clientes hemos obtenido solamente aquél registro para el que el identificador id es igual a uno.

Se puede seleccionar filas donde una columna tiene un valor mayor (o menor) que uno dado.

Ejemplo:

```
mysql> SELECT * FROM clientes WHERE id < 3;
+----+-------------+---------------------+----------+
| id | nombre      | direccion           | vendedor |
+----+-------------+---------------------+----------+
|  1 | Matt Design | NULL                |        1 |
|  2 | Diana Pérez | Brito del Pino 1120 |     NULL |
+----+-------------+---------------------+----------+
2 rows in set (0.00 sec)
```

Las igualdades y desigualdades también se aplican a cadenas de caracteres. Una cadena es menor que otra cuando es previa en orden alfabético. Por ejemplo "azzl" < "baa" . No se distingue entre mayúsculas y minúsculas (aunque, si se lo desea, es posible configurar el servidor MySQL para que sí distinga). Es decir, "A" < "b" < "c" < "D" y "a"="A".

Ejemplo:

```
mysql> SELECT * FROM clientes WHERE nombre < "i";
+----+-------------+---------------------+----------+
| id | nombre      | direccion           | vendedor |
+----+-------------+---------------------+----------+
|  2 | Diana Pérez | Brito del Pino 1120 |     NULL |
+----+-------------+---------------------+----------+
1 row in set (0.00 sec)
```

Este SELECT listará los clientes con nombre empezando en una letra anterior a "i" en el alfabeto.

También se aplican a fechas la igualdad y desigualdad. Para referirse a una fecha, se le escribe como "Año con 4 cifras-Mes-Día". Por ejemplo, "2004-12-20" indica el 20 de Diciembre de 2004.

Ejemplo:

```
mysql> SELECT * FROM ventas WHERE fecha < '2004-12-20';
+----+---------+----------+---------+----------+----------+----------+------------
+
| id | cliente | vendedor | importe | articulo | cantidad | sucursal | fecha
|
+----+---------+----------+---------+----------+----------+----------+------------
+
|  1 |       1 |        1 |  178.50 |        1 |        1 |        1 | 2004-10-01
|
|  2 |       2 |        1 |  195.00 |        2 |        1 |        2 | 2004-10-05
|
|  3 |       3 |        1 |  178.50 |        1 |        1 |     NULL | 2004-10-07
|
|  4 |       1 |        1 |   16.00 |        7 |        2 |        2 | 2004-11-01
|
|  5 |       1 |        2 |  220.00 |        3 |        1 |        1 | 2004-12-02
|
|  6 |       1 |        1 |    5.50 |        5 |        1 |        1 | 2004-10-02
|
|  7 |       1 |        2 |   11.00 |        5 |        2 |        1 | 2004-10-03
|
|  9 |       2 |        1 |  200.00 |        2 |        1 |        2 | 2004-10-12
|
+----+---------+----------+---------+----------+----------+----------+------------
+
8 rows in set (0.00 sec)
```

Para determinar si un dato es NULL se usa la condición 'IS NULL'. Para saber si no es NULL, se usa la condición 'IS NOT NULL'.

Ejemplo:

```
mysql> SELECT * FROM ventas WHERE sucursal IS NULL;
+----+---------+----------+---------+----------+----------+----------+------------
+
| id | cliente | vendedor | importe | articulo | cantidad | sucursal | fecha
|
+----+---------+----------+---------+----------+----------+----------+------------
+
|  3 |       3 |        1 |  178.50 |        1 |        1 |     NULL | 2004-10-07
|
+----+---------+----------+---------+----------+----------+----------+------------
+
1 row in set (0.00 sec)
```

No debe usarse <> NULL. Produce resultados equivocados, en su lugar habría que usar IS NOT NULL.

Llamamos condiciones simples a las siguientes

Expresión	Significado
a = b	a es igual a b
a = b	a es igual a b
a <> b	a es distinto de b
a < b	a es menor que b
a > b	a es mayor que b
a <= b	a es menor o igual a b
a >= b	a es mayor o igual a b
a IS NULL	a es NULL
a IS NOT NULL	a no es NULL

Las cantidades a y b pueden ser números, cadenas de caracteres o fechas, en todos los casos. En una cláusula WHERE se puede usar cualquier condición simple. La cláusula WHERE selecciona aquellas filas en que la condición es verdadera.

Las condiciones simples pueden aparecer combinadas por operadores lógicos. Los operadores lógicos son AND, OR y NOT. Aquí E y F representan condiciones.

Expresión	Significado	Es verdadera cuando ...
X AND Y	X y Y	X es verdadera y Y es verdadera
X OR Y	X o Y o ambos	X es verdadera o Y lo es o ambas
NOT X	No X	X es falsa

NOTA : El operador NOT requiere paréntesis. Es decir se debe escribir WHERE NOT (salario > 50) mientras que es incorrecto WHERE NOT salario > 50. Se entiende que buscamos filas con salarios no mayores que 50.

Ejemplos:

Listar los empleados cuya fecha de ingreso sea anterior al 2004, o cuyo salario sea mayor que 50 (o ambas cosas).

```
mysql> SELECT * FROM empleados WHERE fecha_ingreso < '2004-1-' OR salario > 50;
+----+-----------------+---------------+---------+
| id | nombre          | fecha_ingreso | salario |
+----+-----------------+---------------+---------+
|  1 | Carlos Zaltzman | 1999-12-10    |   10000 |
|  2 | Juan Fernández  | 2000-01-03    |   12000 |
+----+-----------------+---------------+---------+
2 rows in set, 1 warning (0.00 sec)
```

Listar los clientes cuyos nombres empiecen con una letra entre ‘c’ y ‘l’.

```
mysql> SELECT * FROM clientes WHERE "C" <= nombre AND nombre < "M";
+----+-------------+---------------------+----------+
| id | nombre      | direccion           | vendedor |
+----+-------------+---------------------+----------+
|  2 | Diana Pérez | Brito del Pino 1120 |     NULL |
|  3 | John Smith  | Zum Felde 2024      |        1 |
+----+-------------+---------------------+----------+
2 rows in set (0.00 sec)
```

Obsérvese que la condición apropiada para obtener los nombre que empiezan con ‘l’ es nombre <’M’. Si escribiésemos nombre < =‘L’ no obtendríamos un nombre como ‘Luís Zúñiga’ porque ‘L’<’Luís Zúñiga’. Se pueden usar incluso condiciones más complicadas. Por ejemplo, supongamos que ahora queremos listar los empleados cuya fecha de ingreso es anterior al 2004, o cuyo salario sea mayor que 50, pero no ambas cosas a la vez. Entonces, debemos escribir :

```
mysql> SELECT * FROM empleados WHERE (fecha_ingreso < '2004-1-1' OR salario > 50)
    -> AND NOT (fecha_ingreso < '2004-1-1' AND salario > 50);
Empty set (0.00 sec)
```

El resultado es "Empty set", es decir, no hay filas que cumplan la condición. En el caso de condiciones más complicadas se recomienda un amplio uso de paréntesis.

2.2. Cláusula ORDER BY.

La cláusula ORDER BY produce una ordenación de las filas de salida del Query o consulta. Se puede ordenar por una columna seleccionada.

```
mysql> SELECT cliente, articulo FROM ventas ORDER BY cliente;
+---------+----------+
| cliente | articulo |
+---------+----------+
|       1 |        1 |
|       1 |        7 |
|       1 |        3 |
|       1 |        5 |
|       1 |        5 |
|       1 |        5 |
|       2 |        2 |
|       2 |        2 |
|       3 |        1 |
+---------+----------+
9 rows in set (0.02 sec)
```

También se puede ordenar por varias columnas.

```
mysql> SELECT cliente, vendedor, articulo FROM ventas ORDER BY cliente, vendedor;
+---------+----------+----------+
| cliente | vendedor | articulo |
+---------+----------+----------+
|       1 |        1 |        1 |
|       1 |        1 |        7 |
|       1 |        1 |        5 |
|       1 |        2 |        3 |
|       1 |        2 |        5 |
|       1 |        2 |        5 |
|       2 |        1 |        2 |
|       2 |        1 |        2 |
|       3 |        1 |        1 |
+---------+----------+----------+
9 rows in set (0.00 sec)
```

Cuando se ordena por varias columnas, por ejemplo 3, el procedimiento es básicamente el que sigue : Se ordena por la primera columna. Si hay valores repetidos en la primera columna, para cada grupo de valores repetidos se ordenan las filas por el valor de la 2ª columna. Si hay valores repetidos de las dos primeras columnas en conjunto, se ordenan las filas correspondientes por la 3ª columna.

Ejemplo: Supongamos una tabla con las siguientes columnas y valores:

a	b	c	d

125	Diana Pérez	1/1/2004	12313
486	Alejandro Bentancourt	30/12/2005	45646
222	Jorge Rodríguez	2/5/2004	78987
125	Diana Pérez	3/8/2004	654654
222	Adriana Salgado	1/3/2002	12312

Si ordenamos esta tabla por las columnas a, b y c, obtenemos

125	Diana Pérez	1/1/2004	12313
125	Diana Pérez	3/8/2004	654654

222	Adriana Salgado	1/3/2002	12312
222	Jorge Rodríguez	2/5/2004	78987
486	Alejandro Bentancourt	30/12/2005	45646

Por último, se puede ordenar por una cantidad calculada a partir de una o varias columnas.

```
mysql> SELECT articulo, importe, cantidad FROM ventas
    -> ORDER BY articulo, importe/cantidad;
+----------+---------+----------+
| articulo | importe | cantidad |
+----------+---------+----------+
|        1 |  178.50 |        1 |
|        1 |  178.50 |        1 |
|        2 |  195.00 |        1 |
|        2 |  200.00 |        1 |
|        3 |  220.00 |        1 |
|        5 |    5.50 |        1 |
|        5 |   11.00 |        2 |
|        5 |    5.50 |        1 |
|        7 |   16.00 |        2 |
+----------+---------+----------+
9 rows in set (0.00 sec)
```

2.3. Cláusula LIMIT.

La preparación de una consulta complicada implica normalmente un proceso de prueba y error. Aunque no se cometan errores, siempre se empieza escribiendo consultas que sólo realizan una parte de lo que se desea alcanzar. Luego, se van mejorando gradualmente hasta llegar al objetivo buscado.

Cuando se trabaja con tablas auténticas con muchos cientos o miles de filas, puede ser demasiado engorroso ir obteniendo repetidas salidas con cientos o miles de filas. Es obvio que no se pueden observar en la pantalla del cliente mysql. Por otra parte, en su instalación puede haber otros clientes que operen con MySQL.

De todas maneras, interesa una cláusula sencilla que limite el número de filas que produce el SELECT. Esa es la función de LIMIT. Si, por ejemplo, escribimos

```
mysql> SELECT articulo, cliente FROM ventas LIMIT 3;
+----------+---------+
| articulo | cliente |
+----------+---------+
|        1 |       1 |
|        2 |       2 |
|        1 |       3 |
+----------+---------+
3 rows in set (0.00 sec)
```

Vemos 3 filas en la salida, a pesar que hay varias más en Ventas.

El orden de las cláusulas.

Las cláusulas mencionadas, SELECT...FROM.. , WHERE, ORDER BY, y LIMIT deben escribirse, si aparecen, en ese orden. SELECT siempre aparece y va en primer lugar. Las otras 3 son optativas.

2.4. Ejecución de un script en la línea de comando.

Hasta ahora hemos venido escribiendo a continuación del prompt mysql> del cliente mysql.

Este procedimiento presenta inconvenientes cuando tenemos que ejecutar un número importantes de instrucciones o sentencias de forma repetitiva. Un procedimiento sencillo para Windows, es el siguiente:

1. Inicie el programa Bloc de Notas. Suele estar en Inicio, Programas, Accesorios.
2. Escriba en él su consulta. (select * from sucursales;)
3. Guárdelo por ejemplo en C:/MYSQL/MySQL Server5.0/bin/pruebas/pruebas.txt con el nombre pruebas.txt
4. Vaya al cliente mysql y escriba

```
mysql> SOURCE C:/MySQL/MySQL Server 5.0/bin/pruebas/pruebas.txt;
+----+-------------+
| id | descripcion |
+----+-------------+
|  1 | Centro      |
|  2 | Unión       |
|  3 | Marvín      |
+----+-------------+
3 rows in set (0.00 sec)
```

Si la consulta no dio el resultado esperado, ahora no es preciso escribir todo de nuevo. Simplemente corríjalo en el Bloc de Notas. y vuelva a guardarlo. Repita los pasos hasta obtener el resultado buscado. La opción anterior tiene verdaderamente sentido cuando tenemos que ejecutar o probar un grupo importante de sentencias que finalmente conduzcan a un script de mantenimiento o administración de la base de datos diaria, semanal, mensual, etc…

2.5. Problemas resueltos.

1. Obtener todos los datos de la tabla Empleados.

Solución:

```
mysql> SELECT * FROM Empleados;
+----+-----------------+---------------+---------+
| id | nombre          | fecha_ingreso | salario |
+----+-----------------+---------------+---------+
|  1 | Carlos Zaltzman | 1999-12-10    |   10000 |
|  2 | Juan Fernández  | 2000-01-03    |   12000 |
+----+-----------------+---------------+---------+
2 rows in set (0.00 sec)
```

2. Obtener los nombres y las fechas de ingreso de los empleados.

Solución : Formalmente, queremos obtener las columnas Nombre, Fecha_ingreso de todas las filas de la tabla Empleados. La consulta es:

```
mysql> SELECT  Nombre, Fecha_ingreso FROM Empleados;
+-----------------+---------------+
| Nombre          | Fecha_ingreso |
+-----------------+---------------+
| Carlos Zaltzman | 1999-12-10    |
| Juan Fernández  | 2000-01-03    |
+-----------------+---------------+
2 rows in set (0.00 sec)
```

3. Obtener los datos de los empleados que ganan más de 10500,50.

Solución: Queremos obtener todas las columnas de las filas de Empleados para las cuales Salario es mayor que 10500,50.

```
mysql> SELECT * FROM Empleados
    -> WHERE Salario > 10500.50;
+----+----------------+---------------+---------+
| id | nombre         | fecha_ingreso | salario |
+----+----------------+---------------+---------+
|  2 | Juan Fernández | 2000-01-03    |   12000 |
+----+----------------+---------------+---------+
1 row in set (0.00 sec)
```

Obsérvese el uso del punto decimal. Las comas no se pueden utilizar ni en lugar del punto decimal ni para separar los miles.

4. Queremos obtener los datos del funcionario Carlos Zaltzman.

Solución: Se trata de obtener una fila de Empleados, aquella en la cual el nombre del funcionario es Carlos Zaltzman.

```
mysql> SELECT  * FROM  Empleados
    -> WHERE Nombre = 'Carlos Zaltzman';
+----+-----------------+---------------+---------+
| id | nombre          | fecha_ingreso | salario |
+----+-----------------+---------------+---------+
|  1 | Carlos Zaltzman | 1999-12-10    |   10000 |
+----+-----------------+---------------+---------+
1 row in set (0.00 sec)
```

5. ¿Qué empleados ingresaron el 3 de enero del 2000?.

Solución: Se trata de obtener filas de Empleados, aquellas en las cuales la fecha de ingreso sea '2000-1-3'. El formato apropiado para la fecha es Año-Mes-Día. La consulta es:

```
mysql> SELECT  * FROM Empleados
    -> WHERE Fecha_ingreso = '2000-1-3';
+----+----------------+---------------+---------+
| id | nombre         | fecha_ingreso | salario |
+----+----------------+---------------+---------+
|  2 | Juan Fernández | 2000-01-03    |   12000 |
+----+----------------+---------------+---------+
1 row in set (0.00 sec)
```

6. ¿Qué empleados ingresaron en fecha posterior al 1º de enero del 2000?

Solución: Queremos obtener las filas de Empleados con fecha de ingreso mayor que '2000-1-1'. La consulta es:

```
mysql> SELECT  * FROM Empleados
    -> WHERE Fecha_ingreso>'2000-1-1';
+----+----------------+---------------+---------+
| id | nombre         | fecha_ingreso | salario |
+----+----------------+---------------+---------+
|  2 | Juan Fernández | 2000-01-03    |   12000 |
+----+----------------+---------------+---------+
1 row in set (0.00 sec)
```

7. ¿Qué empleados ingresaron el 5 de enero del 2000 o en una fecha posterior?

Solución: La fecha de ingreso debe ser mayor o igual a '2001-1-5'.

```
mysql> SELECT  * FROM Empleados
    -> WHERE Fecha_ingreso>='2000-1-5';
Empty set (0.00 sec)
```

8. Obtener una lista de los nombres de los clientes de los cuales no se tiene dirección.

Solución: Se trata de obtener la columna Nombre de las filas de la tabla Clientes cuya dirección es NULL.

```
mysql> SELECT Nombre FROM Clientes
    -> WHERE Direccion IS NULL;
+-------------+
| Nombre      |
+-------------+
| Matt Design |
+-------------+
1 row in set (0.00 sec)
```

9. Sacar una lista de los nombres y direcciones de los clientes de los cuales sí se tiene la dirección.

Solución: Se trata de obtener la columnas Nombre, Direccion de las filas de la tabla Clientes que tienen direcciones que no sean NULL.

```
mysql> SELECT  Nombre,Direccion FROM Clientes
    -> WHERE Direccion IS NOT NULL;
+-------------+---------------------+
| Nombre      | Direccion           |
+-------------+---------------------+
| Diana Pérez | Brito del Pino 1120 |
| John Smith  | Zum Felde 2024      |
+-------------+---------------------+
2 rows in set (0.00 sec)
```

10. Obtener una lista de los diferentes salarios que se pagan en la empresa.

Solución:

```
mysql> SELECT DISTINCT Salario FROM Empleados;
+---------+
| Salario |
+---------+
|   10000 |
|   12000 |
+---------+
2 rows in set (0.00 sec)
```

11. Obtener una lista ordenada alfabéticamente de los nombres y direcciones de los clientes, ordenados por nombre.

Solución:

```
mysql> SELECT Nombre,Direccion FROM Clientes
    -> ORDER BY Nombre;
+-------------+---------------------+
| Nombre      | Direccion           |
+-------------+---------------------+
| Diana Pérez | Brito del Pino 1120 |
| John Smith  | Zum Felde 2024      |
| Matt Design | NULL                |
+-------------+---------------------+
3 rows in set (0.00 sec)
```

12. Obtener una lista de todos los datos de los empleados ordenados por nombre.

Solución:

```
mysql> SELECT * FROM Empleados
    -> ORDER BY Nombre;
+----+-----------------+---------------+---------+
| id | nombre          | fecha_ingreso | salario |
+----+-----------------+---------------+---------+
|  1 | Carlos Zaltzman | 1999-12-10    |   10000 |
|  2 | Juan Fernández  | 2000-01-03    |   12000 |
+----+-----------------+---------------+---------+
2 rows in set (0.00 sec)
```

13. Obtener una lista de los datos de los empleados ordenados por salario en forma descedente. Los salarios deben ser formateados con comas cada 3 dígitos y dos decimales después del punto y alineados a la derecha, por ejemplo, 1,200,340.50.

Solución:

```
mysql> SELECT Id, Nombre, Fecha_ingreso, LPAD( FORMAT(Salario,2) , 12 , ' ' )
    -> AS Salario FROM Empleados ORDER BY Salario DESC;
+----+-----------------+---------------+-------------+
| Id | Nombre          | Fecha_ingreso | Salario     |
+----+-----------------+---------------+-------------+
|  2 | Juan Fernández  | 2000-01-03    |   12,000.00 |
|  1 | Carlos Zaltzman | 1999-12-10    |   10,000.00 |
+----+-----------------+---------------+-------------+
2 rows in set (0.00 sec)
```

14. Igual al anterior pero se quiere que los números salgan formateados con puntos cada 3 dígitos y coma decimal, en vez de punto. Por ejemplo, 1.200.340,50 .

Solución:

```
mysql> SELECT Id, Nombre, Fecha_ingreso,
    -> REPLACE(
    -> REPLACE(
    -> REPLACE(LPAD(FORMAT(Salario,2),12,' '),
    ->    '.','!'),
    ->    ',','.'),
    ->    '!',',')
    -> AS Salario
    ->     FROM Empleados ORDER BY Salario DESC;
+----+-----------------+---------------+--------------+
| Id | Nombre          | Fecha_ingreso | Salario      |
+----+-----------------+---------------+--------------+
|  2 | Juan Fernández  | 2000-01-03    |    12.000,00 |
|  1 | Carlos Zaltzman | 1999-12-10    |    10.000,00 |
+----+-----------------+---------------+--------------+
2 rows in set (0.00 sec)
```

Si bien es un procedimiento complicado, para usarlo en otro caso sólo hay que copiar la parte que comienza con REPLACE y termina en el paréntesis antes de AS Salario y hacer los siguientes cambios:

1. Cambiar la columna Salario por la que se vaya a usar.
2. Eventualmente cambiar el número de decimales después de la coma, que figura dentro de la función FORMAT, de 2 al valor que se desee.
3. Si es conveniente, variar el ancho de la columna. Este ancho aparece dentro de la función LPAD. En este ejemplo vale 12. El ancho debe ser suficiente para poder escribir todos los valores que aparezcan en la columna, con el número de decimales que se haya solicitado.

3. SELECT (como calculadora)

Se puede usar la cláusula SELECT para hacer cálculos aritméticos.

Ejemplos:

```
mysql> SELECT 1+1;
+-----+
| 1+1 |
+-----+
|   2 |
+-----+
1 row in set (0.00 sec)
```

O este otro:

```
mysql> SELECT 4.5*(3.86+2.34);
+-----------------+
| 4.5*(3.86+2.34) |
+-----------------+
|          27.900 |
+-----------------+
1 row in set (0.00 sec)
```

usado COUNT (DISTINCT ....) para contar cuántos elementos distintos hay en una expresión calculada en base a una columna.

7.2. Búsqueda de texto.

A menudo tenemos columnas que son cadenas de caracteres, y queremos buscar las cadenas que contienen cierta palabra. Esto se realiza a través de un nuevo tipo de condición: Nombre_de_columna LIKE cadena_de_caracteres.

Por ejemplo, Nombre LIKE ‘Carlos’.

Estas condiciones pueden usarse, como todas, en una cláusula WHERE o en una cláusula HAVING. La condición Nombre_de_columna LIKE cadena_de_caracteres (donde cadena_de_caracteres no contiene ni ‘%’ ni ‘_’) es verdadera cuando el valor de Nombre_de_columna coincide con cadena_de_caracteres (salvo que no se distingue entre mayúsculas y minúsculas).

Veamos los valores de la condición A LIKE ’Carlos’ donde A es una columna:

Valor de la columna A	
'Carlos'	Verdadera
'carlos'	Verdadera
'Carlos '	Falsa
'Juan'	Falsa
'Juan Carlos'	Falsa

Un % dentro de cadena_de_caracteres representa cualquier cadena de caracteres, incluso una sin caracteres. Entonces A LIKE ‘Carlos%’ tiene los siguientes valores:

Valor de la columna A	
'Carlos'	Verdadera
'carlos'	Verdadera
'Carlos Zaltzman'	Verdadera
'Carlos '	Verdadera
'Juan'	Falsa
'Juan Carlos'	Falsa

Valores de A LIKE ‘%Carlos’:

Valor de la columna A	
'Carlos'	Verdadera
'carlos'	Verdadera
'Juan carlos'	Verdadera
'Juan Carlos Rodríguez’	Falsa
'Juan Carlos '	Falsa

Valores de A LIKE ‘%Carlos%’:

Valor de la columna A	
'Carlos'	Verdadera
'carlos'	Verdadera
' carlos '	Verdadera
'Juan Carlos Rodríguez’	Verdadera
'Juan'	Falsa

El carácter '_' representa un carácter cualquiera. A diferencia de ‘%’ representa un carácter y uno sólo. Luego dos’_’ seguidos representan dos caracteres cualquiera, etc.

Valores de A LIKE ‘__Carlos’

Valor de la columna A	
'Carlos'	Falsa
'12Carlos'	Verdadera
'xxCarlos'	Verdadera
' Carlos'	Verdadera
'xxcarlos'	Verdadera
'Juan Carlos'	Falsa

NOT LIKE es simplemente la negación de LIKE: es verdadera cuando LIKE es falsa y recíprocamente.

Problemas resueltos

1. Listar el artículo cuyo código es ’mon20'.

Solución:

```
mysql> SELECT * FROM Articulos WHERE Codigo LIKE ‘mon20’;
+----+-------------+--------+--------+-------+
| id | nombre      | precio | codigo | grupo |
+----+-------------+--------+--------+-------+
|  2 | Monitor 20" | 200.00 | mon20  |     1 |
+----+-------------+--------+--------+-------+
1 row in set (0.16 sec)
```

2. Hacer una lista de todos los datos de los artículos en cuyo nombre figura la palabra ‘monitor’.

Solución:

```
mysql> SELECT  * FROM  Articulos WHERE  Nombre LIKE ‘%monitor%’;
+----+-------------+--------+--------+-------+
| id | nombre      | precio | codigo | grupo |
+----+-------------+--------+--------+-------+
|  1 | Monitor 16" | 178.50 | mon16  |     1 |
|  2 | Monitor 20" | 200.00 | mon20  |     1 |
|  3 | Monitor 22" | 220.00 | mon22  |     1 |
+----+-------------+--------+--------+-------+
3 rows in set (0.00 sec)
```

Obsérvese que aunque la palabra monitor esté en la columna Nombre con mayúscula, igualmente aparece en el resultado de la consulta.

3. Listar todos los datos de los artículos cuyo código comience por ‘mon’.

Solución:

```
mysql> SELECT * FROM Articulos WHERE Codigo LIKE ‘mon%’;
+----+-------------+--------+--------+-------+
| id | nombre      | precio | codigo | grupo |
+----+-------------+--------+--------+-------+
|  1 | Monitor 16" | 178.50 | mon16  |     1 |
|  2 | Monitor 20" | 200.00 | mon20  |     1 |
|  3 | Monitor 22" | 220.00 | mon22  |     1 |
+----+-------------+--------+--------+-------+
3 rows in set (0.00 sec)
```

4. Listar los artículos cuyo código termine en ‘20’.

Solución:

```
mysql> SELECT * FROM Articulos WHERE Codigo LIKE '%20';
+----+--------------+--------+--------+-------+
| id | nombre       | precio | codigo | grupo |
+----+--------------+--------+--------+-------+
|  2 | Monitor 20"  | 200.00 | mon20  |     1 |
|  7 | Diskettes 20 |   8.00 | D20    |     2 |
+----+--------------+--------+--------+-------+
2 rows in set (0.00 sec)
```

5. Listar los artículos cuyo código tenga exactamente dos caracteres.

Solución:

```
mysql> SELECT * FROM Articulos WHERE Codigo LIKE "__";
Empty set (0.00 sec)
```

No hay ninguno. Entonces MySQL responde "Empty set", es decir, "Conjunto vacío".

6. Listar todos los artículos en cuyo nombre NO figure la palabra "monitor".

Solución:

```
mysql> SELECT * FROM Articulos WHERE Nombre NOT LIKE "%monitor%";
+----+----------------+--------+--------+-------+
| id | nombre         | precio | codigo | grupo |
+----+----------------+--------+--------+-------+
|  4 | Motherboard FX |  99.50 | mthFX  |     1 |
|  5 | Papel A4-500   |   5.50 | PA4500 |     2 |
|  6 | Diskettes 10   |   4.50 | D10    |     2 |
|  7 | Diskettes 20   |   8.00 | D20    |     2 |
+----+----------------+--------+--------+-------+
4 rows in set (0.00 sec)
```

7.3. Unión de dos tablas de resultados .

UNION ALL indica que se haga la unión de dos resultados, simplemente escribiendo una tabla debajo de la otra. Por ejemplo, si queremos los artículos y cantidades de las ventas a los clientes 1 y 2, basta escribir

```
(SELECT Articulo, Cantidad FROM Ventas
 WHERE Cliente=1)
UNION ALL
(SELECT Articulo, Cantidad FROM Ventas
 WHERE Cliente=2);
```