Base de Datos I
Curso No: TWB22B

Volumen 4
Fundamentos de MySQL

Unidad 1
Fundamentos de MySQL

Objetivos del Aprendizaje

- Listar las capacidades de MySQL.
- Describir la licencia GPL.
- Describir la historia de MySQL y su evolución.
- Listar las principales características de MySQL.
- Describir el motor de almacenamiento de MySQL.
- Discutir sobre la arquitectura de hilos de MySQL.
- Explicar cómo es implementada la seguridad en MySQL.
- Describir las interfaces proporcionadas en MySQL para los programadores.

Introducción

- MySQL, es un Sistema de Administración de Base de Datos de código abierto, es licenciado bajo la GPL (General Public License) de la GNU.
- Fue creada por la empresa sueca MySQL AB.
- MySQL es el sistema administrador de base de datos más usado en el mundo del software libre, debido a su gran rapidez, confiabilidad y facilidad de uso.
- MySQL es parte de LAMP (Linux, Apache, MySQL, PHP / Perl / Python), fuente de rápido crecimiento de software de código abierto para negocios.

Licencia GPL

- Las licencias que cubren la mayor parte del software son comerciales, es decir el creador de la obra mantiene el código fuente y no es de libre distribución.
- La Licencia Pública General de GNU pretende garantizar la libertad de compartir y modificar software libre para asegurar que el software es libre para todos sus usuarios.
- Cuando se habla de software libre, se refiere a libertad del código fuente, no al precio.

Historia de MySQL

- La necesidad de una base de datos SQL para aplicaciones Web llevaron a desarrollar una base de datos comercial inspirada en proyectos de código abierto (open source), MySQL comenzó a desarrollarse en 1994.
- No se sabe con certeza de donde proviene el nombre MySQL. Probablemente pueda ser de dos fuentes, la compañía MySQL AB los últimos 10 años a colocado como prefijo a los desarrollos realizados la palabra “My”, la otra fuente podría ser el nombre de una de las hijas del co-fundador Michael “Monty” Widenius, esto todavía sigue siendo un misterio.

Hitos importantes en la evolución de MySQL

1995 Liberada primera versión de MySQL 1.0

1996 Liberada la versión MySQL 3.11 sobre Linux y Solaris

1997 Primera Licencia comercial y contrato de soporte

2000 La licencia de MySQL cambia a GPL

Octubre 2004 La versión de producción GA es 4.1

Octubre 2005 La versión de producción GA es 5.0

Características de MySQL

Internas y de portabilidad :

- Escrito en C y C++.
- Está disponible en diferentes plataformas: Linux, Solaris, FreeBSB, Mac OS X, HP-UX, AIX, Windows, etc.
- Disponibilidad de APIs para C, C++, Eiffel, Java, Perl, PHP, Python, Ruby y Tcl.
- Aprovecha la potencia de sistemas multiprocesador, gracias a su implementación multihilo.
- Tablas Hash en memoria, son usadas como tablas temporales.
- El código de MySQL ha sido probado (Tested) con las principales herramientas del mercado.
- El servidor está disponible como un programa separado para ser usado en un ambiente cliente/servidor.

Características de MySQL

Tipos de Columna soportados :

INTEGER de 1, 2, 3, 4, y 8 Bytes
FLOAT
DOUBLE

CHAR
VARCHAR
TEXT
BLOB

DATE, TIME, DATETIME, TIMESTAMP, YEAR

Tipos espaciales OpenGIS

Características de MySQL

Sentencias y Funciones:

- Soporte para las cláusulas GROUP BY y ORDER BY.
- Pueden usarse las funciones: COUNT(), COUNT(DISTINCT ...),AVG(), STD(), SUM(), MAX() y MIN().
- Soporte para LEFT OUTER JOIN y RIGHT OUTER JOIN usando notación SQL estándar.
- Soporte para alias sobre: tablas y columnas usando SQL estándar.
- Las sentencias DELETE, INSERT y UPDATE retornan el número de filas que han sido afectadas.
- Se puede mezclar tablas de diferentes bases de datos en la misma consulta.

Características de MySQL

Seguridad

- Maneja un sistema de privilegios muy seguro, la verificación se hace basado en host.

Escalabilidad y Límites

- Maneja base de datos grandes. Su uso se extiende a más de 50 millones de registros. Se tiene conocimiento de algunos usuarios que usan el servidor MySQL con más de 60,000 tablas y cerca de 5.000.000.000 de filas.
- Hasta 64 índices por tabla son permitidos. Cada índice puede consistir de 1 a 16 columnas.

Conectividad

- Los clientes pueden conectarse al servidor MySQL usando TCP/IP sobre cualquier plataforma.
- El conector/ODBC (MyODBC) provee soporte a programas cliente que usen ODBC (Open Database Connectivity).
- La interfase conector/J provee soporte para programas cliente java que usan JDBC.

¿Porque seleccionar a MySQL?

- **Aplicaciones Web:** Muchas consultas y pocas escrituras, MySQL resuelve rápidamente peticiones.

- **Aplicaciones de Negocios:** Por su estabilidad, bajo costo, rapidez y soporte.

- **Soporte a Código-Abierto:** MySQL es software libre de código abierto, está basado en licencia GPL.

- **Bajo requerimiento:** MySQL no requiere recursos excesivos para ejecutarlo.

- **Disponibilidad para tablas de gran tamaño:**. En algunos sistemas usando MySQL se ha logrado hasta 8 terabytes (TB) por tabla.

- **Estabilidad:** MySQL está escrito en múltiples capas, y diferentes módulos, cada uno de los módulos están en versiones estables de funcionamiento.

Deficiencias de MySQL

Características no presentes en el núcleo de MySQL 4.1.11:

- Procedimientos almacenados
- Disparadores (Triggers):
- Vistas

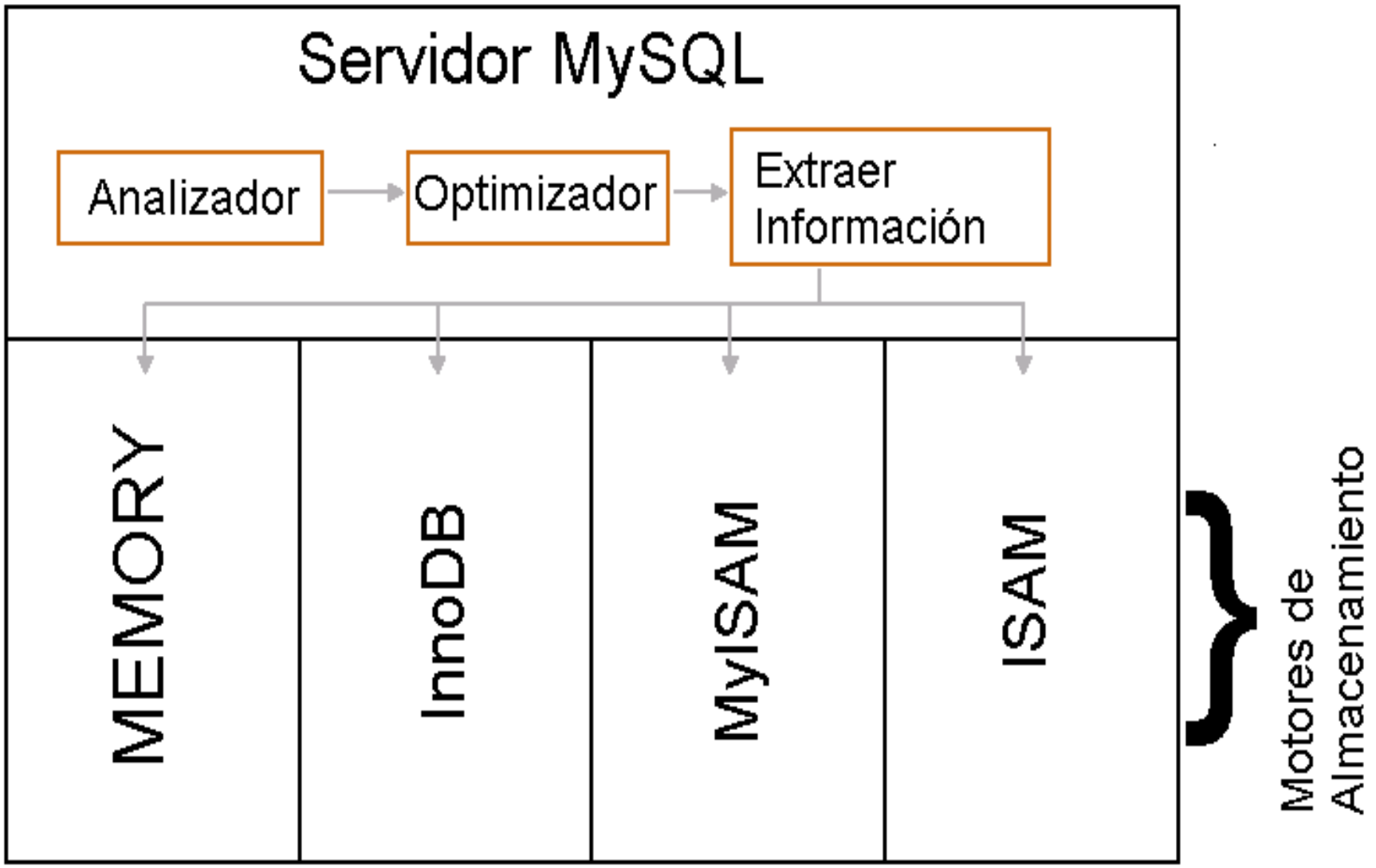
Soporte a estas características ha sido agregado a la serie 5.x, con soporte al estándar ANSI SQL-99 y SQL-2003.

Motor de almacenamiento de MySQL

El motor de almacenamiento (storage engine) es el software que se encarga del manejo de los datos, cómo se organizan y qué relaciones tienen, como se almacenan y de qué forma son accedidos, cómo se gestiona el acceso de distintos usuarios y los bloqueos pertinentes, sus medidas de seguridad y la integridad.

En MySQL es posible seleccionar el tipo de motor de almacenamiento a utilizar, esto se indica en la sentencia de creación de la tabla

Motor de almacenamiento de MySQL



Motor de almacenamiento de MySQL

Motor de almacenamiento ISAM: Motor de almacenamiento original de MySQL, sólo manejaba tablas no-transaccionales, estuvo disponible en la versión MySQL 3.23 y rápidamente fue reemplazado por MyISAM.

Características:

- Registros de longitud fija y variable.
- Sólo pueden ser definidos 16 índices por tabla.
- Claves con longitud máxima de 256 bytes.
- Los datos son almacenados en el formato de la máquina donde esta instalado, más rápido, pero dependiente de la máquina.
- El máximo tamaño de una tabla es 4GB.
- No puede usarse sentencias de respaldo de tablas y restauración de tablas.
- No soporta búsquedas de texto completo y tipos de datos espaciales (OpenGIS).

Motor de almacenamiento de MySQL

Motor de almacenamiento MyISAM: Es el motor de almacenamiento por defecto en MySQL desde la serie 3.23. MyISAM esta basado en el código de ISAM pero tiene otras poderosas propiedades:

- **Características:**
  - Todos los datos son almacenados en formato complemento a dos y el formato de la IEEE de punto flotante.
  - Manejo de tablas no-transaccionales
  - El máximo número de índice por tablas son 64. El máximo número de columnas por índices es 16.
  - MyISAM automáticamente actualiza las columnas definidas como AUTO_INCREMENTO en operaciones NSERT/UPDATE, incrementado la velocidad en al menos 10%.
  - Los índices MyISAM tienen una bandera que indica si la tabla fue cerrada correctamente. Si mysqld es inicializado con la opción de recuperación, el MyISAM automáticamente repara la tabla que no haya sido cerrada correctamente.
  - Usado frecuentemente en aplicaciones Web.

Motor de almacenamiento de MySQL

Motor de almacenamiento Memory (HEAP): El motor de almacenamiento MEMORY o HEAP crea tablas que son almacenadas en memoria

Características:

- Las tablas MEMORY permiten hasta 32 índices por tabla, 16 columnas por índice, y un máximo tamaño de clave de 500 bytes.
- Manejo de tablas no-transaccionales.
- Las tablas MEMORY no soportan columnas BLOB y TEXT.
- Las tablas MEMORY no soportan columnas AUTO_INCREMENTO.
- Las tablas MEMORY no pueden ser convertidas a tablas físicas.
- El computador donde esta instalado el servidor MySQL requiere memoria adicional para mantener todas las tablas MEMORY que son usadas en el mismo momento.
- Para liberar memoria usada por una tabla MEMORY, podrían usarse las sentencias DELETE o DROP TABLE.

Motor de almacenamiento de MySQL

Motor de almacenamiento InnoDB : InnoDB provee a MySQL un motor de almacenamiento con soporte a transacciones (propiedades ACID) con capacidades para commit y rollback.

Características:

- Manejo de usuarios concurrentes.
- InnoDB soporta la definición de claves foráneas (FOREIGN KEY).
- En una sentencia SELECT puedan mezclarse diferentes tipos de tablas.
- InnoDB ha sido diseñado para máximo rendimiento cuando se procesan grandes volúmenes de datos.
- Integrada completamente con el servidor MySQL, InnoDB mantiene su propio buffer pool para mantener datos e índices en memoria principal.
- No existe un limite de tamaño predefinido para las tablas InnoDB

Motor de almacenamiento de MySQL

Características InnoDB:

- InnoDB es usado en numerosos lugares a nivel en ambientes de producción
- Mucho más seguro. Si ocurren problemas de hardware o MySQL falla, es posible recuperar la información ya sea por una recuperación automática o desde un archivo log de transacciones de respaldo.
- Es posible combinar muchas sentencias y aceptar todas al mismo tiempo con la sentencia COMMIT.
- Para ignorar los cambios realizados se ejecuta la sentencia ROLLBACK.
- Si una actualización falla todos los cambios son revertidos (en un ambiente no transaccional todos los cambios hechos son permanentes).
- Los motores transaccionales proporcionan un mejor desempeño sobre tablas que tienen muchas actualizaciones concurrentes.

Arquitectura de Hilos de MySQL

- MySQL corre sobre un motor de base de datos multi-hilo.
- Los clientes que se conectan al servidor de base de datos MySQL no necesitan esperar que otro cliente finalice la consulta o proceso que este ejecutando para que sean atendidas sus peticiones.
- Cuando un usuario se conecta al servidor de base de datos MySQL, un nuevo proceso llamado hilo maneja las tareas requerida por esa conexión
- MySQL mantiene activo un hilo administrador que es el encargado de recibir y atender las peticiones de otros hilos en un momento determinado.

Seguridad en MySQL

- El sistema de seguridad en MySQL es referido como el Sistema de Privilegios de Acceso.
- Permite la autenticación de los usuarios del servidor de MySQL y la verificación de las actividades de todos los usuarios sobre el servidor y las bases de datos.
- La seguridad en MySQL es aplicada en dos niveles:
  - Nivel de servidor
  - Nivel de Base de Datos
- MySQL realiza la verificación de privilegios usando unas tablas del sistema llamadas tablas de concesión

Sistema de privilegios de acceso en MySQL

• El mecanismo mediante el cuál se aseguran los datos y la integridad es el sistema de privilegios de MySQL.

• Las tablas de concesión de MySQL son responsables de la autenticación de usuarios y host (otros computadores) que acceden al servidor MySQL

• Las tablas de concesión están localizadas dentro del manejador de base de datos, específicamente en la base de datos mysql.

• MySQL almacena en las tablas de concesión toda información referente a los privilegios de conexión al sistema así como también los privilegios de acceso a las base de datos.

• En el momento de inicio el servidor lee de estas tablas toda la información referente a los privilegios y los carga en memoria principal.

Sistema de privilegios de acceso en MySQL

Tablas de concesión de MySQL:

• **La tabla user** es usada para especificar la información de privilegios de los usuarios que intentan conectarse al servidor MySQL.

• **La tabla db** es usada para especificar los accesos a las base de datos almacenadas en el servidor.

• **La tabla host** es usada en conjunción con la tabla db para permitir que un usuario pueda conectarse a un servidor MySQL desde diferentes máquinas en una red de computadoras.

• **La tabla tables_priv** es usada para especificar los privilegios de los usuarios a nivel de tablas.

• **La tabla columns_priv** es usada para especificar los privilegios de los usuarios a nivel de columnas

Clientes MySQL

- MySQL es inherentemente un sistema administrador de base de datos para redes
- Es posible que un cliente pueda comunicarse con un servidor que esté corriendo localmente o que el servidor esté en un lugar distante.
- Existen muchos programas cliente para MySQL algunos de ellos ofrecen interfaces gráficas (GUI) otros se basan en líneas de comando.
- Clientes:
  - Línea de comandos mysql
  - Cliente MySQL Administrator
  - Cliente MySQL Query Browser

Línea de comandos mysql

- El mysql es un programa cliente de línea de comandos SQL
- Usar el cliente mysql es muy sencillo, solo basta con invocar desde la línea de comandos de Linux

shell>mysql -u [nombre_usuario] -p[password] -h [nombre_host]

Por ejemplo:

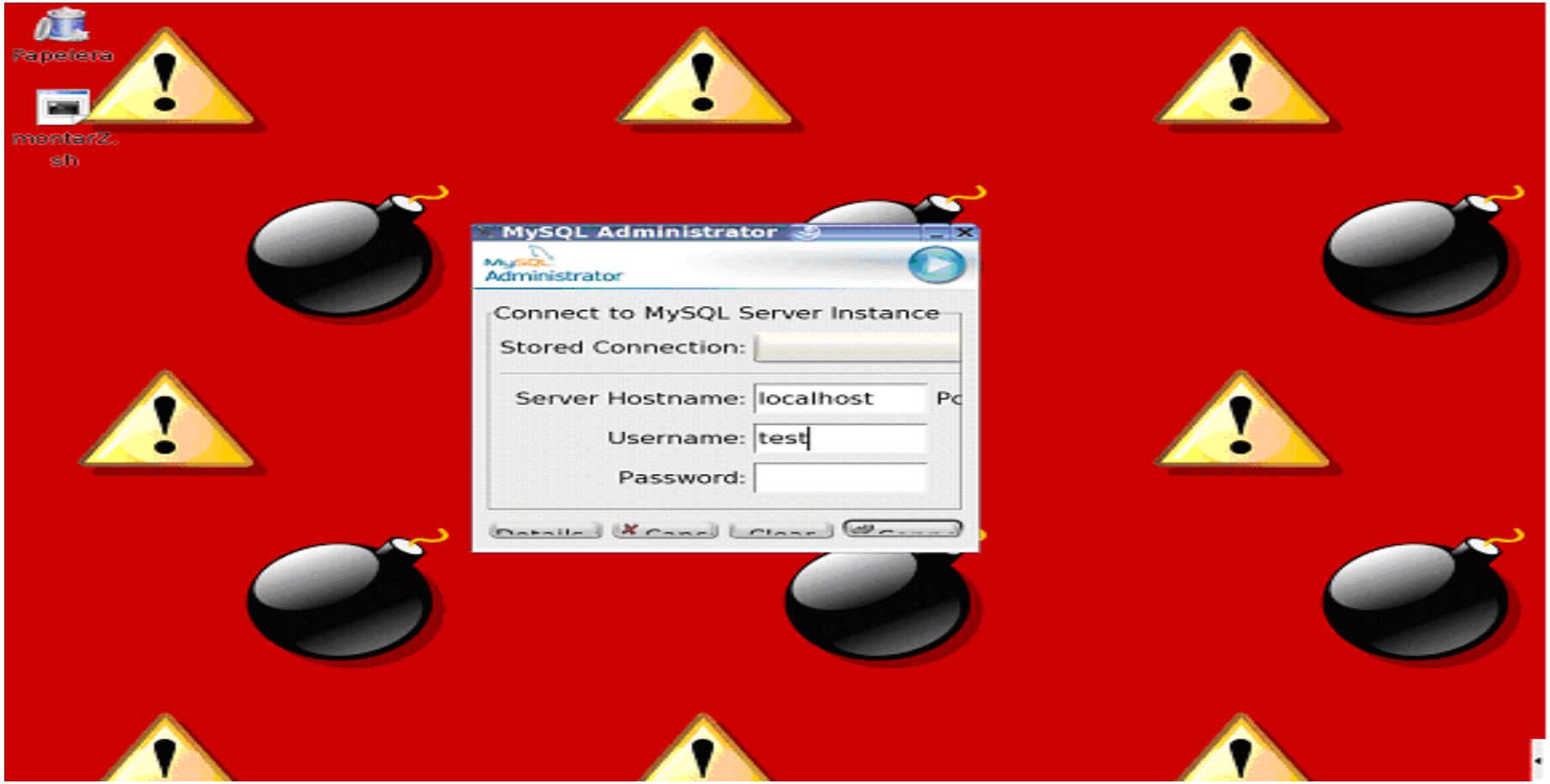
shell>>mysql –u mysqladmin –pmysqladmin test1

Cliente MySQL Administrator

- El administrador MySQL es un programa gráfico que permite a un usuario realizar tareas administrativas.
- Como por ejemplo:
  - Configurar el servidor MySQL,
  - Monitorear el desempeño del servidor
  - Verificar el comportamiento,
  - Iniciar o detener el servidor de base de datos,
  - Administrar usuarios y conexiones,
  - Ejecutar respaldos de la base de datos,
  - Administrar las tablas de concesión

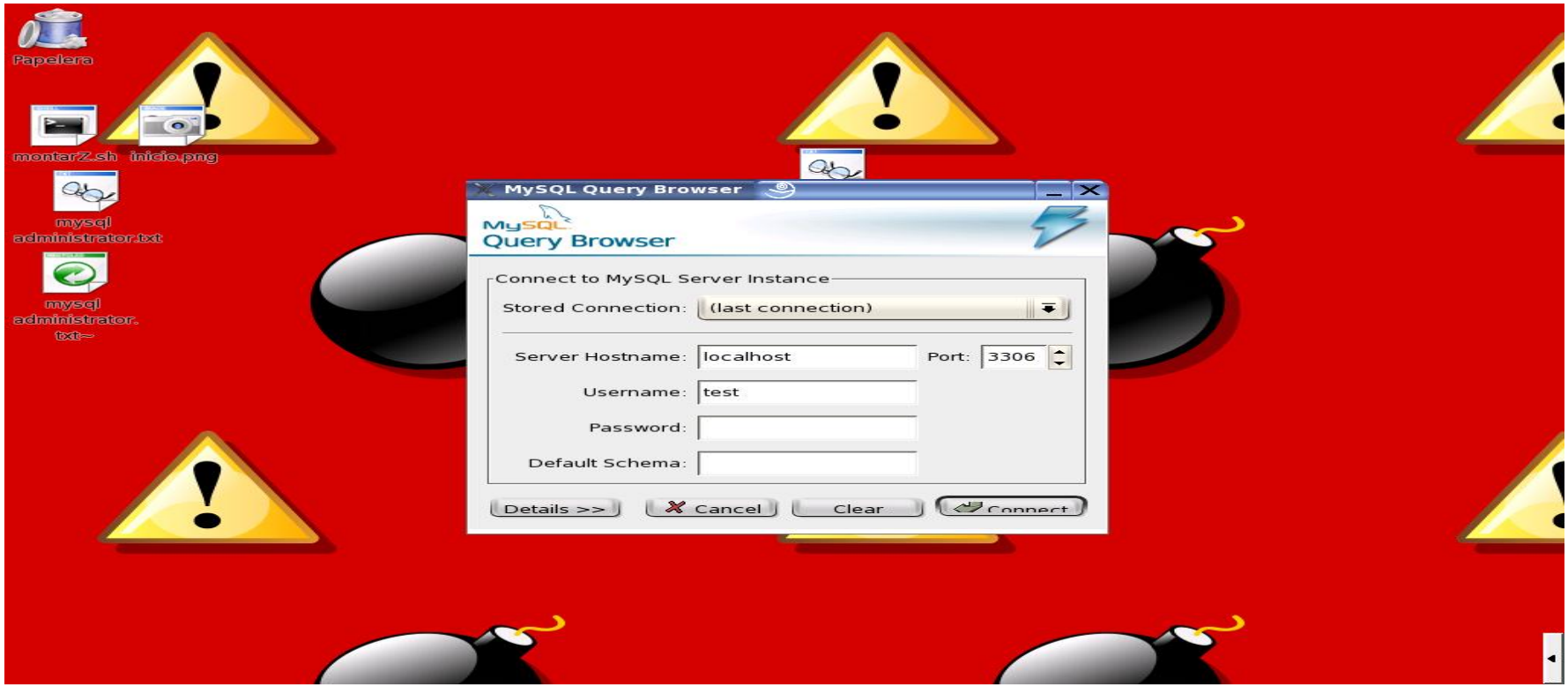
Cliente MySQL Administrator

- Ventana de Inicio, usuario: mysqladmin y password: mysqladmin o usuario: test y password: Test2005



Cliente MySQL Query Browser

- MySQL Query Browser es una utilidad para trabajar con la base de datos MySQL.
- Es un editor de sentencias SQL gráfico.
- Dispone de un editor de tablas y registros



Resumen

- Ahora que ha completado esta unidad, usted debe ser capaz de:
- Explicar la historia y la evolución de MySQL.
- Explicar el concepto de la licencia GPL.
- Listar las fortalezas y debilidades de MySQL.
- Describir el motor de almacenamiento de MySQL.
- Discutir sobre la arquitectura de hilos de MySQL.
- Explicar el sistema de seguridad de MySQL.
- Discutir sobre herramientas de MySQL.

Unidad 2
Ejecución de sentencias SQL con MySQL

Objetivos del Aprendizaje

- Describir el lenguaje SQL
- Describir los tipos de datos soportados por MySQL
- Explicar qué son DDL, DML y DCL
- Describir los privilegios de MySQL
- Escribir una sentencia SELECT simple
- Escribir una sentencia SELECT condicional
- Escribir consultas multitablas
- Generar respaldo de base de datos
- Ejecutar sentencias en el cliente mysql
- Ejecutar sentencias en el cliente MySQL Query Browser

Introducción al SQL

- SQL (Lenguaje de consulta estructurado) es un lenguaje muy sencillo.
- Basado en el habla inglés, orientado principalmente a base de datos relacionales y, sobre todo, al manejo de consultas.
- El lenguaje SQL está compuesto por una serie de sentencias y de cláusulas muy reducidas en número, pero muy potentes en efectividad.
- MySQL es el sistema administrador de base de datos más usado en el mundo del software libre, debido a su gran rapidez, confiabilidad y facilidad de uso.
- En esta unidad, se da una explicación completa del uso de las sentencias SQL sobre el sistemas administrador de base de datos MySQL.

Ejecución de sentencias SQL con el programa mysql

- Para conectarse al servidor, usualmente necesitamos de un nombre de usuario (login) y de una contraseña (password), y si el servidor al que nos deseamos conectar está en una máquina diferente de la nuestra, también necesitamos indicar el nombre o la dirección IP de dicho servidor.

  shell>mysql –u [usuario] –p [Presione la tecla Enter]

  A continuación ingrese el password, y presione [Enter]

  - Usuario:mysqadmin y password:mysqladmin
  - Usuario:test y password:Test2005

```
LIBMP8213:~ # mysql -h localhost -u test -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1 to server version: 4.1.11-standard

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql>
```

Ejecución de sentencias SQL con el programa mysql

- Un comando normalmente consiste de una sentencia SQL seguida por un punto y coma.
- Ejemplos de Comandos MySQL:
  - mysql> SELECT VERSION(), CURRENT_DATE;

```
mysql> SELECT VERSION(), CURRENT_DATE;
+-----------------+--------------+
| VERSION()       | CURRENT_DATE |
+-----------------+--------------+
| 4.1.11-standard | 2005-06-16   |
+-----------------+--------------+
1 row in set (0.02 sec)
```

- Es posible mostrar la información de la base de datos actual usando la sentencia SELECT DATABASE();

Ejecución de sentencias SQL con el programa mysql

- Un comando no necesita ser escrito en una sola línea, mysql determinará en donde finaliza la sentencia cuando encuentre el punto y coma, no cuando encuentre el fin de línea

```
mysql> SELECT USER(),
    -> CURRENT_DATE();
+----------------+----------------+
| USER()         | CURRENT_DATE() |
+----------------+----------------+
| test@localhost | 2005-06-16     |
+----------------+----------------+
1 row in set (0.00 sec)
```

Ejecución de sentencias SQL con el programa mysql

- A continuación se usará la sentencia SHOW DATABASES para listar las bases de datos existentes en el servidor al que estamos conectados, sólo se mostrarán las bases de datos a las cuales se tiene privilegio de acceso.

```
mysql> SHOW DATABASES;
+----------+
| Database |
+----------+
| test     |
+----------+
1 row in set (0.02 sec)
```

- Se uso el usuario test para la conexión

Ejecución de sentencias SQL con el programa mysql

- SHOW TABLES es un comando que permite listar las tablas disponibles en la base de datos activa.
- Para mostrar el motor de almacenamiento usado en las tablas de una base de datos, use:
- SHOW TABLE STATUS FROM <BASE DE DATOS>;

```
mysql> SHOW TABLES;
+-----------------+
| Tables_in_test  |
+-----------------+
| persona         |
+-----------------+
1 row in set (0.00 sec)
```

Tipos de Datos en MySQL

- **Tipo de dato Numérico:**
  - tinyint
  - smallint
  - mediunmint
  - integer, int
  - bigint
  - float
  - xreal, double
  - decimal(m,d), dec(m,d) y numeric(m,d)
- **Tipo de dato Cadena:**
  - char(n)
  - varchar(n)
  - text(n)
  - blob(n)
- **Tipo de dato Fecha:**
  - date
  - datatime
  - timestamp
  - time
  - year

Sentencias DDL

- CREATE DATABASE: Permite crear una base de datos, la sintaxis es la siguiente:
- CREATE DATABASE [IF NOT EXISTS] nombre_base_de_datos;
- Sino se usa IF NOT EXISTS en la sentencia CREATE DATABASE, producirá un mensaje de error si ya existe una base de datos con el mismo nombre.
- DROP DATABASE: Elimina todas las tablas de la base de datos y elimina la base de datos, para ejecutar esta sentencia se debe tener el privilegio DROP sobre la base de datos.

Sentencias DDL

- CREATE TABLE: Esta sentencia es utilizada para crear una tabla y definir las columnas que contiene.

Sintaxis:

```
1.- CREATE TABLE nombre_tabla(
nombre_columna1 tipo [NOT NULL]
[PRIMARY KEY],
nombre_columna2 tipo [NOT NULL],
nombre_columnan tipo [NOT NULL])
[ENGINE | TYPE={INNODB | MYISAM |
HEAP}];
```

```
2.- CREATE TABLE nombre_tabla(
nombre_columna1 tipo [NOT NULL],
nombre_columna2 tipo [NOT NULL],
nombre_columnan tipo [NOT NULL][,
[CONSTRAINT   NOMBRERESTRICCION]
PRIMARY KEY(nombre_columna(s)),
INDEX(nombre_columna(s))])
[ENGINE | TYPE={INNODB | MYISAM |
HEAP}];
```

```
3.- CREATE TABLE nombre_tabla
LIKE nombre_otra_tabla;
```

```
Sintaxis para definir columnas es:
 nombre_col tipo [NOT NULL | NULL] [DEFAULT valor_por_defecto]
 [AUTO_INCREMENT] [[PRIMARY] KEY] [COMMENT 'string']
```

Sentencias DDL

1.- CREATE TABLE cliente
(
id_cliente INT NOT NULL,
nombre VARCHAR(20),
PRIMARY KEY (id_cliente)
) TYPE = INNODB;

2.- CREATE TABLE cliente_copia
LIKE cliente;

3.- CREATE TABLE factura
(
id_factura INT NOT NULL PRIMARY KEY,
id_cliente INT NOT NULL,
monto DOUBLE,
INDEX (id_cliente)
) TYPE = HEAP;

4.- CREATE TABLE PRUEBA
(
ID INT AUTO_INCREMENT PRIMARY KEY,
PREGUNTA VARCHAR(20),
RESPUESTA VARCHAR(20)
)
AUTO_INCREMENT = 10;

Sentencias DDL

Con la sentencia DESCRIBE es posible obtener información sobre la definición de una tabla:

```
DESCRIBE cliente;
```

Usando SHOW COLUMNS FROM cliente puede obtenerse un resultado semejante

Ejemplo:

SHOW COLUMNS FROM cliente;

Sentencias DDL

ÍNDICES: Para definir índices sobre una columna o sobre varias se usan indistintamente las opciones KEY o INDEX.

Ejemplo:

```
CREATE TABLE computador(
id INT,
marca VARCHAR(20),
modelo VARCHAR(20),
INDEX(modelo));
```

```
CREATE TABLE computador(
id INT,
marca VARCHAR(20),
modelo VARCHAR(20),
KEY(modelo));
```

También podemos crear un índice sobre parte de una columna:

```
CREATE TABLE computador(
 id INT,
 marca VARCHAR(20),
 modelo VARCHAR(20),
 INDEX(modelo(4)));
```

Sentencias DDL

- CLAVES FORÁNEAS: Se pueden definir claves foráneas en cualquier tipo de tabla de MySQL, pero únicamente tienen sentido cuando se usan tablas del tipo InnoDB
- Ejemplo:

```
CREATE TABLE cliente
(
   id_cliente INT NOT NULL,
   nombre VARCHAR(30),
   PRIMARY KEY (id_cliente)
) TYPE = INNODB;
```

```
CREATE TABLE factura
(
   id_factura INT NOT NULL,
   id_cliente INT NOT NULL,
   monto   DOUBLE,
   PRIMARY KEY(id_factura),
   FOREIGN    KEY    (id_cliente)
REFERENCES   cliente(id_cliente)
ON DELETE CASCADE
) TYPE = INNODB;
```

Sentencias DDL

- En muchas oportunidades es posible que se requiera cambiar la estructura de las tablas en una base de datos, para hacer esto usamos la sentencia ALTER TABLE
- Ejemplo:

1.- ALTER TABLE cliente ADD COLUMN salario INT;	2.- ALTER TABLE cliente DROP COLUMN salario;
3.- ALTER TABLE cliente ADD COLUMN salario DOUBLE;	4.- ALTER TABLE cliente ADD INDEX salario_ind (salario);

Sentencias DDL

- DROP TABLE: Permite eliminar una o varias tabla(s) de una base de datos

- Sintaxis:

DROP TABLE [IF EXISTS] nombre_tabla[, nombre_tabla2,….] ...

- Ejemplo

DROP TABLE personal, asegurados;

Sentencias DML

- INSERT: Sentencia usada para insertar datos en una tabla
- **Sintaxis:**

INSERT INTO
nombre_tabla[(nombre_columna1,nombre_columna2,..,..)]VALUES(valor1,valor2,..);

- **Usando INSERT con SELECT:**

INSERT INTO
nombre_tabla1 (col1,col2,...) SELECT col1,col2,.. FROM nombre_tabla2;

- **Ejemplos:**

INSERT INTO cliente values (1,'Juancho',54121);
INSERT INTO cliente values (2, 'Luis',45687.25),(3,'Pedro',5487.54),
(4,'Pedro',54877.59),(5,'Pedro',54875.25),(6,'Pablo',125478.25),
(7,'Nelson',12657.36),(8,'Nelson',54875.25) ,(9,'Jesus',5647.25),
(10,'Jesus',5487), (11,'Miguel',45877) ,(12,'Miguelina',45877),
(13,'Pedrito',458377);
INSERT INTO factura values(1,1,5878787.26);

Sentencias DML

- CARGA de DATOS DESDE UN ARCHIVO: Para cargar datos en una tabla desde un archivo externo se usa la sentencia LOAD DATA
- Ejemplo:

```
mysql> LOAD DATA LOCAL INFILE "personal.txt“ INTO TABLE personal;
```

- La sentencia LOAD DATA nos permite especificar cuál es el separador de columnas, y el separador de registros, por defecto el “tabulador” es el separador de columnas (campos), y el “salto de línea” es el separador de registros
- En el ejemplo anterior la estructura de la tabla personal debe estar definida en la base de datos.

Sentencias DML

- UPDATE: La sentencia UPDATE es usada para actualizar valores de columnas de una tabla.

- **Sintaxis:**

UPDATE nombre_tabla

SET nombrecol1=valor, [nombrecol2=valor, ...]

[WHERE Condición]

[LIMIT #]

- **Ejemplo:**

UPDATE cliente

SET nombre='Luis'

Where id_cliente=1;

Sentencias DML

- DELETE: La sentencia DELETE es usada para eliminar registros de una tabla.

Sintaxis:

1.- DELETE FROM nombre_tabla [WHERE Condición] [ORDER BY ....] [LIMIT #];	2.- DELETE <nombretabla1>, <nombretabla2>..... FROM nombretabla1,nombretabla2... [WHERE Condición];

Ejemplos:

1.- DELETE FROM cliente WHERE id_cliente=5;
2.- DELETE FROM cliente ORDER by nombre LIMIT 1;
3.- DELETE cliente,factura FROM cliente,factura where cliente.id_cliente = factura.id_cliente;

Sentencias DQL

- SELECT: La sentencia SELECT se usa para recuperar filas seleccionadas de una o más tablas.

- **Sintaxis:**

  SELECT [ALL | DISTINCT | DISTINCTROW]
  nombrecolumna1,nombrecolumna2,....
  FROM tabla1[,tabla2...]
  [WHERE condiciones]
  [GROUP BY (nombre_col)]
  [HAVING condiciones]
  [ORDER BY nombre_col] [ASC | DESC]
  [LIMIT contador]

- **Ejemplo:**

  SELECT * FROM cliente;

Sentencias DQL

- DISTINCT y DISTINCTROW: Son usadas para restringir la aparición de filas repetidas.
  - **Ejemplo.**
    - **SELECT [DISTINCT | DISTINCTROW] nombre FROM cliente;**
- WHERE: Generalmente nos interesará saber qué filas se ajustan a determinados parámetros. Esto se realiza con la cláusula WHERE del SELECT
  - **Ejemplo:**
    - **SELECT * FROM cliente WHERE nombre='Luis';**
- ALIAS: El alias se usa como un nombre de columna en expresiones, también es posible definir alias a las tablas de la cláusula FROM.
  - **Ejemplo:**
    - **SELECT CONCAT(nombre,', ',salario) AS Nombre_Salario**
      FROM cliente ORDER BY Nombre_Salario;

Sentencias DQL

- GROUP BY: Es posible agrupar filas en la salida de una sentencia SELECT usando la cláusula GROUP BY.
  - **Ejemplo:**
    - **SELECT id_cliente, nombre FROM cliente GROUP BY nombre;**

- HAVING: La cláusula HAVING permite hacer selecciones sobre datos agrupados.
  - **Ejemplo:**
    - **SELECT nombre, count(nombre) FROM cliente GROUP BY nombre HAVING COUNT(nombre) >= 2;**

- ORDER BY: Es usada para dar ordenamiento a los datos recuperados en una sentencia SELECT.
  - **Ejemplo:**
    - **SELECT id_cliente,nombre FROM cliente ORDER BY id_cliente, nombre;**

Sentencias DQL

- LIMIT: La cláusula LIMIT permite limitar el número de filas recuperadas por la sentencia SELECT.
  - **Ejemplo:**
    - **SELECT * FROM cliente LIMIT 3;**

- Operador IN: El operador IN se usa para realizar comparaciones con una lista de valores.
  - **Ejemplo:**
    - **SELECT * FROM cliente WHERE nombre = 'Jesus' OR nombre= 'Pedro' OR nombre='Nelson';**
    - **Esta sentencia puede ser reescrita como:**
    - **SELECT * FROM cliente WHERE nombre IN ('Jesus','Pedro','Nelson');**

Sentencias DQL

- Operador BETWEEN: El operador BETWEEN se usa para comprobar si cierto valor está dentro de un rango dado.
  - **Ejemplo:**
    - **SELECT id_cliente,nombre FROM cliente WHERE salario >= 1000 AND salario <= 50000;**
    - **La consulta anterior se puede escribir también usando el operador BETWEEN.**
    - **SELECT id_cliente,nombre FROM cliente WHERE salario BETWEEN 1000 AND 50000;**
- Operador IS: Los operadores IS NULL e IS NOT NULL son utilizados para verificar si una expresión determinada es o no nula.
  - **Ejemplo:**
    - **SELECT * FROM cliente WHERE nombre IS NOT NULL;**

Sentencias DQL

- Operador LIKE: Es usado para hacer comparaciones entre cadenas y patrones. El resultado es verdadero (1) si la cadena se ajusta al patrón, y falso (0) en caso contrario.
- Los patrones son cadenas de caracteres en las que se pueden encontrar en cualquier posición los caracteres especiales '%' y '_'.
- '%' Es usado para hacer coincidir cualquier número de caracteres, incluso ninguno.
- '_' Es usado para hacer coincidir un único carácter.

  - **Ejemplo:**
    - **SELECT * FROM cliente WHERE nombre LIKE 'Miguel%';**

Consultas Multitablas

- Los datos están distribuidos en diferentes tablas debido a la normalización
- Enlaza tablas de datos por medio de un valor de atributo común en ambas tablas
- Puede ser aplicada a dos o más tablas para poder recuperar registros de múltiples tablas
- Tipos de JOIN:

  - Cartesian JOIN (Producto Cartesiano)
  - INNER JOIN
  - Right Outer
  - Left Outer
  - Full Outer (No esta implementado en MySQL)
  - Self

Consultas Multitablas

- PRODUCTO CARTESIANO (CARTESIAN JOIN): Hace corresponder todas las filas de la primera tabla con todas las filas de la segunda tabla y presenta una combinación de todos los registros de ambas tablas.

  - **Ejemplo**
    - **SELECT * FROM personal, division;**

Consultas Multitablas

- INNER JOIN: Las composiciones internas (INNER JOIN) se definen a partir de un producto cartesiano, eliminado las filas que no cumplen la condición de composición.
  - **Ejemplos**
    - **SELECT * FROM personal JOIN division ON (personal.no_division=division.no_division);**
    - **SELECT * FROM personal INNER JOIN division ON (personal.no_division=division.no_division);**
    - **SELECT * FROM personal JOIN division USING(no_division);**

Consultas Multitablas

- RIGHT OUTER JOIN: La sentencia Right Outer JOIN hace corresponder los registros de la tabla del lado derecho con los registros de la tabla del lado izquierdo basándose en la igualdad de valores que se especifica en la condición JOIN.

  - **Ejemplos**
    - **SELECT * FROM personal RIGHT OUTER JOIN division ON (personal.no_division=division.no_division);**
    - **Puede ser escrito como:**
    - **SELECT * FROM personal RIGHT OUTER JOIN division USING(no_division);**

Consultas Multitablas

- LEFT OUTER JOIN: La sentencia Left Outer JOIN hace corresponder los registros de la tabla del lado izquierdo con los registros de la tabla del lado derecho basándose en la igualdad de valores que es especificada en la condición JOIN.

  - **Ejemplo**
    - **SELECT * FROM personal LEFT OUTER JOIN division ON (personal.no_division=division.no_division);**
    - **Puede ser escrito como:**
    - **SELECT * FROM personal LEFT OUTER JOIN division USING (no_division);**

Consultas Multitablas

- SELF JOIN: Es posible realizar un JOIN en una misma tabla si una tabla tiene dos atributos que comparten el mismo valor.

  - **Ejemplos**
  - **SELECT S.nombre, M.nombre FROM personal S, personal M WHERE S.no_grnt = M.no_personal;**

SubConsultas

- Subconsulta Correlacionada: Cuando la subconsulta hija hace referencia a una o más columnas de la subconsulta padre.
  - **Ejemplo**
    - **SELECT division.nombre_division FROM division**

      WHERE EXISTS (

      SELECT no_division FROM personal WHERE personal.no_division = division.no_division);

- Subconsulta No Correlacionada: No existe referencia de las columnas entonces se dice que es no-correlacionada.
  - **Ejemplo**
    - **SELECT * FROM personal WHERE no_division =**

      (SELECT no_division FROM division WHERE nombre_division='HARDWARE');

Usuarios y Privilegios

- MySQL implementa 5 niveles de privilegios:
  - **Globales**: Se aplican al conjunto de todas las bases de datos en un servidor. Es el nivel más alto de privilegio.
  - **De base de datos**: Se aplican a bases de datos individuales, y a todos los objetos que contiene cada base de datos.
  - **De tabla**: Se aplican a tablas individuales, y a todas las columnas de esas tablas.
  - **De columna**: Se aplican a una columna en una tabla.
  - **De rutina**: Se aplican a los procedimientos almacenados (MySQL 5.x).

Usuarios y Privilegios

- SENTENCIA GRANT: Usando GRANT podemos crear un usuario y al mismo tiempo concederle privilegios.

- La sintaxis es:

- GRANT tipo_priv [(columnas)] [, tipo_priv [(columnas)]] ...
  ON {nombre_tabla | * | *.* | nombre_bd.*}
  TO usuario [IDENTIFIED BY [PASSWORD] 'password']
  [,usuario [IDENTIFIED BY [PASSWORD] 'password']] ...
  [WITH GRANT OPTION]

Usuarios y Privilegios

- SENTENCIA REVOKE: Para revocar privilegios se usa la sentencia REVOKE, la sintaxis es similar a la sentencia GRANT:

- REVOKE tipo_priv [(columnas)] [, tipo_priv [(columnas)]] ...
  ON {nombre_tabla | * | *.* | nombre_bd.*}
  FROM user [, user] ....

- SENTENCIA SHOW GRANT: Podemos ver qué privilegios se han concedido a un usuario mediante la sentencia SHOW GRANTS. La salida de esta sentencia es una lista de sentencias GRANT que se deben ejecutar para conceder los privilegios que tiene el usuario.
- Por ejemplo:
- Para mostrar los privilegios del usuario actual ejecutamos SHOW GRANTS:

Respaldo y Restauración

- El programa cliente **mysqldump** es usado para respaldar una base de datos o colecciones de base de datos.
- La sintaxis es la siguiente:

1. Ejecutar desde la línea de comandos de Linux:
2. mysqldump nombre_bd –u nombre_usuario –p > nombre_archivo_salida
3. Después presione [ENTER] y coloque su contraseña, en el archivo estarán las sentencias DDL y DML.

– **Ejemplo:**

- **shell>mysqldump test –u test –p >respaldo.txt**

Resumen

- Describir el lenguaje SQL
- Describir los tipos de datos soportados por MySQL
- Explicar qué son DDL, DML y DCL
- Describir los privilegios de MySQL
- Escribir una sentencia SELECT simple
- Escribir una sentencia SELECT condicional
- Escribir consultas multitablas
- Generar respaldo de base de datos
- Ejecutar sentencias en el cliente mysql