

Contenido

Introducción	1
Funciones del SGBD. Componentes	2
Tipos de SGBD	3
Arquitectura del SGBD, arquitectura ANSI/SPARC.....	4
SGBD comerciales y libres	5
SGBD monocapa.....	5
SGBD de dos capas y tres capas.	6
Instalación de un SGBD de dos capas.....	7
Instalación de MariaDB	7
Variables.....	9
Variables de sistema	9
Variables de estado	11
Interfaces estándar: conectores ODBC y JDBC.....	12
El Diccionario de Datos.....	12
Ficheros LOG	13
Otros sistemas de almacenamiento.....	17
Copias de seguridad y restauración	18
Documentación.	18
Apéndice. Ejemplos	19

Introducción

En este módulo trabajaremos con el SGBD [MariaDB](#), en primer lugar, porque ya se utilizó en el módulo Gestión de Bases de Datos de este ciclo formativo.

Aunque MySQL es uno de los motores de bases de datos más extendido ha habido una serie de compras entre empresas que ha producido que MySQL acabe siendo propiedad de Oracle, esto ha provocado que MySQL ya no sea completamente Open Source,

sino que se acerca al software comercial, aunque existe una versión Community Edition que sí se puede usar de manera gratuita. Ante esta situación, se creó MariaDB, que es básicamente una variante de MySQL que mantiene toda la filosofía del software libre y nos asegura dos objetivos muy importantes: mantener la compatibilidad con MySQL, de modo que MariaDB pueda usarse como reemplazo de MySQL sin necesidad de ninguna costosa migración, y que la



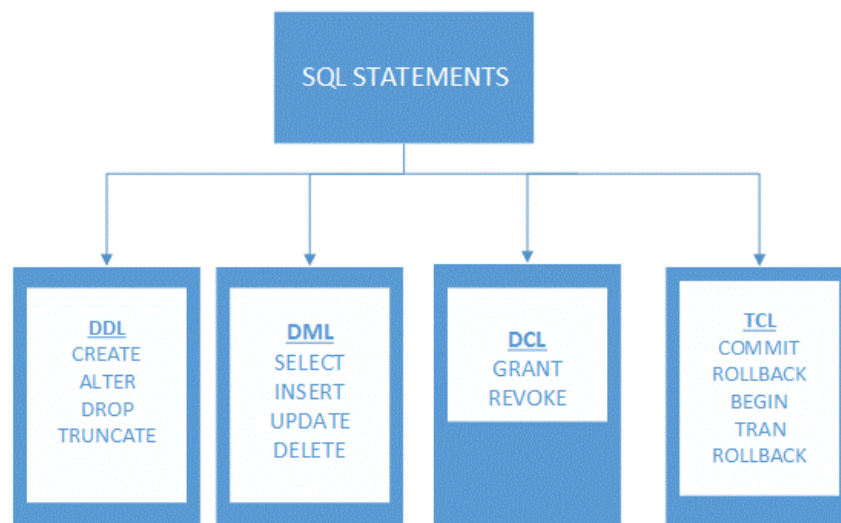
Comunidad pueda continuar trabajando en el sistema gestor de base de datos, sin verse condicionados por las estrategias comerciales.

Funciones del SGBD. Componentes

Un Sistema Gestor de Bases de Datos consiste en una colección de datos relacionados junto con un software que facilita el acceso y control de esos datos.

Entre sus funciones principales destacan:

- Control de redundancia. Se establecen una serie de condiciones para que los datos no estén repetidos de forma descontrolada, de modo que su mantenimiento sea más fácil y favorecer la consistencia de los mismos.
- Restricción de acceso. Se trata de garantizar que sólo las personas autorizadas puedan acceder a los datos que les correspondan.
- Garantizar la integridad. Controla que los datos almacenados sean coherentes con otros datos con los que guarde relación.
- Respaldo y recuperación de datos. Permite realizar copias de seguridad de la base de datos y restablecerlas en caso de necesidad.
- Control de accesos concurrentes. Gestiona que varios usuarios puedan acceder al tiempo a los mismos datos de forma segura.



Los componentes con los que debe contar un SGBD son:

- Gestor de almacenamiento: Es el módulo que proporciona la interfaz entre los datos almacenados y los programas de aplicación y consultas solicitadas al sistema. Consta de:
 - Gestor de transacciones: Asegura que la base de datos sea consistente ante fallos en el sistema.
 - Gestor de archivos: Gestiona la reserva de espacio de disco y las estructuras de datos para representar los datos almacenados en el disco.
 - Gestor de memoria intermedia: Se encarga de traer los datos desde el disco a la memoria principal y decidir qué datos se llevan a la memoria caché.

- Gestor de consultas: Está formado por:
 - Intérprete del DDL: Interpreta el DDL y graba las definiciones en el Diccionario de Datos.
 - Compilador del DML: Traduce las consultas a instrucciones de bajo nivel que puedan ser ejecutadas por el motor de consultas. También optimiza las consultas.
 - Motor de evaluación de consultas: Ejecuta las instrucciones dadas por el Compilador DML.

Tipos de SGBD

Los tipos de SGBD vienen dados en función de características muy diversas y que no guardan una relación directa las unas con las otras.

Entre las distintas categorías posibles destacan las que citamos a continuación:

- Modelo lógico empleado: Define la estructura de datos que se va a implementar. Según este criterio tenemos las siguientes clases:
 - Modelo Jerárquico.
 - Modelo en Red.
 - Modelo Relacional.
 - Modelo Orientado a Objetos.
- Número de usuarios concurrentes: Según la cantidad de usuarios que pueden estar conectados al tiempo, los clasificamos en:
 - Monousuario.
 - Multiusuario.
- Precio de la licencia:
 - Gratuitos.
 - De pago.
- Número de plataformas soportadas: Hace referencia a la cantidad de sistemas operativos en los que se puede instalar.
 - Monoplataforma.
 - Multiplataforma.
- Acceso al código fuente:
 - Libres: Permiten el acceso al código fuente.
 - Privativos: No permiten el acceso al código fuente.
- Número de sitios:
 - Centralizados.
 - Distribuidos: homogéneos y heterogéneos.
- Ámbito de aplicación:
 - Propósito General.
 - Propósito Específico.



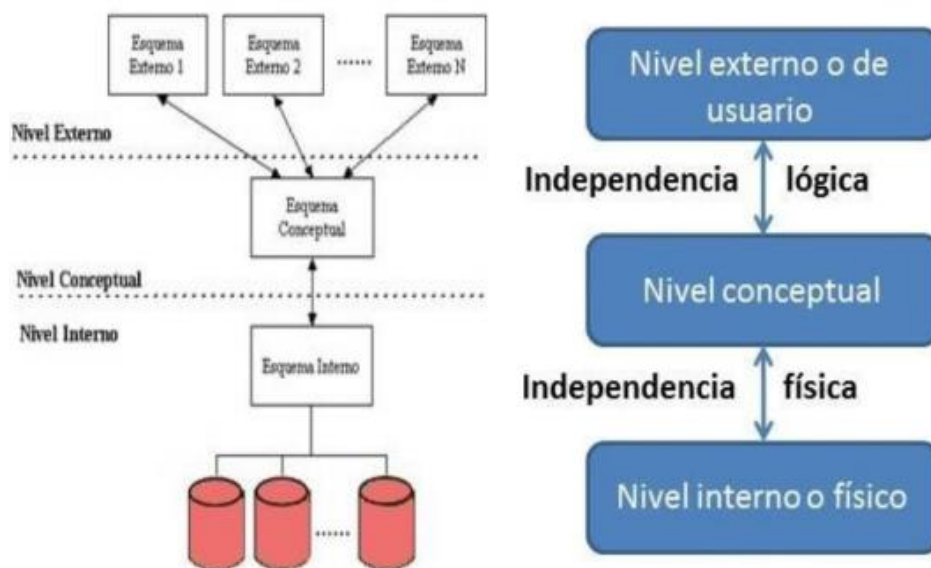
En base a esto, tendremos que, por ejemplo, PostgreSQL es un SGBD de código libre, multiplataforma, multiusuario, gratuito y que se basa en un modelo relacional, aunque también admite un modelo orientado a objetos.

Arquitectura del SGBD, arquitectura ANSI/SPARC

La arquitectura ANSI/X3/SPARC establece tres niveles de abstracción que todo SGBD debe cumplir. Los niveles establecen distintos grados de abstracción sobre el mismo conjunto de datos.

Estos 3 niveles de abstracción, desde el más cercano a la máquina hasta el más cercano al usuario final, son:

- Nivel interno o físico: Es el nivel más bajo de abstracción y en el que existe realmente la base de datos. En él se describen como se almacenan los datos y la estructura de los mismos, para lo que se emplea el Diccionario de Datos.
- Nivel lógico o conceptual: En él se describe cuáles son los datos que se almacenan y qué relaciones se establecen entre ellos, representando de forma completa la base de datos con estructuras simples.
- Nivel externo: Está formado por el conjunto de las vistas de usuario, siendo una vista de usuario la fracción de la base de datos que ve un usuario en concreto. En este nivel es donde trabajan los programas de aplicación de usuario.



En una base de datos determinada habrá un único esquema interno, un único esquema conceptual, pero varios esquemas en el nivel externo.

Es necesario establecer unas reglas para poder pasar del nivel físico al lógico y del lógico al externo. Estas reglas se denominan correspondencias entre niveles y se guardan en el Diccionario de Datos.

Lo que se persigue estableciendo esta estructura es la independencia entre el sistema y los datos, para ocultar al usuario toda la complejidad del sistema y facilitarle la interacción con el mismo.

La independencia de los datos puede darse a dos niveles:

- Independencia física: consiste en modificar el nivel interno sin que se vea afectado el nivel conceptual.
- Independencia lógica: consiste en modificar el nivel lógico sin alterar el nivel físico. La idea de estandarizar el diseño de las redes hizo que la compatibilidad entre todos los elementos aumentara notablemente y con ello la expansión de las redes de comunicación.

SGBD comerciales y libres

A la hora de escoger un SGBD, el primer aspecto a tener en cuenta las particularidades del entorno, por ejemplo si deseo gestionar una aplicación de biblioteca personal, puede ser suficiente usar MS Access como SGBD, e incluso una simple hoja de cálculo puede cubrir suficientemente la necesidad. Por tanto, no hay que escoger un SGBD porque sea el más novedoso o el que más nos guste, sino porque es el que mejor se adapta a nuestras necesidades.

Uno de los factores a considerar es el coste, pero a pesar de ser un factor importante no debería ser tan prioritario, aunque el día a día nos demuestra que esto no es así.

Los sistemas gestores de bases de datos pueden ser libres y no libres, e independientemente de eso, ser gratuitos o no. Por ejemplo, Microsoft SQL Server es un SGBD privativo pero cuenta con alguna versión gratuita como la Compact Edition Básica.

SGBD monocapa



Este tipo de sistema gestor viene pensado para dar servicio a bases de datos pequeñas y con un número de usuarios muy reducido, como puede ser un usuario particular en su domicilio o pequeños negocios, en los que el mantenimiento de un servidor resulta poco práctico, además de caro.

Es en estos casos cuando más se suelen emplear los sistemas gestores que incorporan las suites ofimáticas, como MS Office Access, OpenOffice.org Base o LibreOffice Base; pero mientras que Base viene por defecto dentro del paquete de OpenOffice.org o LibreOffice, no sucede lo mismo con Access que no viene en todas las distribuciones de Microsoft Office.



Estos sistemas pueden ejecutarse desde otra máquina en la que no están instalados accediendo a un recurso compartido en red, aunque lo más habitual es hacerlo en la propia máquina en la que están instalados.

La gran ventaja de estos sistemas es que pueden interactuar con los demás programas del paquete ofimático, lo que permite acceder a los datos almacenados tanto desde el procesador de texto como la hoja de cálculo, pudiéndose además crear macros que permiten automatizar

tareas repetitivas. Además, incorporan asistentes que facilitan la creación de distintos tipos de formularios con facilidad.

Hoy en día emplean motores bastante potentes que no tienen mucho que envidiar a los de SGBD más grandes.

SGBD de dos capas y tres capas.



La situación más habitual cuando accedes a una base de datos es que no estés físicamente sentando ante la máquina que aloja el SGBD, sino que te conectas a él a través de una red informática. De esta manera podemos distinguir entre máquina cliente, donde trabaja

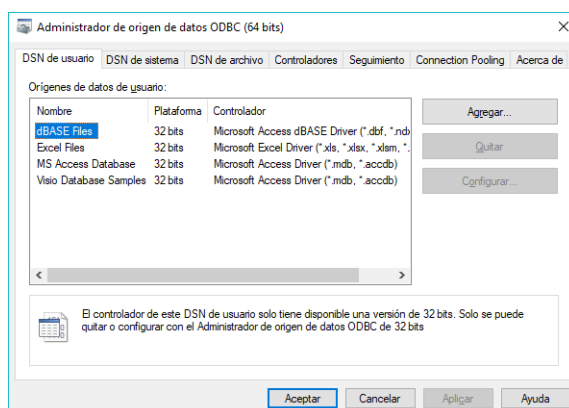
el usuario que se conecta a la base de datos, y máquina servidor, aquella en la que se aloja y ejecuta el sistema de bases de datos.

En una arquitectura de dos capas la aplicación que ejecuta el usuario reside en la máquina cliente y llama a la base de datos alojada en el servidor.

Sin embargo en la arquitectura de tres capas la máquina cliente no hace ninguna llamada a la base de datos. Se ha encapsulado la aplicación en el servidor, creando un servidor de aplicaciones, y se incorpora una interfaz en el cliente que se comunica con dicho servidor siendo este el que accede a la base de datos en el servidor. Esta interfaz en el cliente suele ser una interfaz de formularios.



La lógica de negocio de la aplicación se incorpora al servidor de aplicaciones en la arquitectura de tres capas, en lugar de estar distribuyéndose con la aplicación a todos los clientes en los que se instale esta, como sucede con la arquitectura de dos capas.



Los estándares de conexión ODBC y JDBC se utilizan para la interconexión entre la máquina cliente y la máquina servidor.

Instalación de un SGBD de dos capas

MariaDB es un SGBD de código libre, lo que garantiza que cualquiera puede usar y modificar el código que lo hace funcionar. Esto se debe a que el software está registrado bajo la licencia GPL.

MariaDB funciona bajo una estructura cliente-servidor, consistente en un servidor multihilo y multiplataforma que trabaja con diferentes programas cliente.

A la hora de instalar MariaDB podemos escoger entre la distribución de código fuente o la binaria. Se utiliza la primera cuando en el proceso de instalación queremos incluir o excluir alguna característica determinada, por lo que suele ser más habitual emplear la distribución binaria.

En la web de descarga, <https://downloads.mariadb.org/>, hay diferentes versiones estables (nosotros **usaremos la versión 7.3.9** de 64 bits que la puedes [descargar aquí](#)), código fuente y MariaDB Galera Cluster (mantener réplicas de los datos). Las razones principales para utilizar este SGBD frente a otros son:

- Que es gratuito y multiplataforma.
- Cumple el estándar SQL.
- Es muy fácil encontrar información de apoyo, ya que está basado en MySQL que es un SGBD muy difundido y también el propio MariaDB lo es.

Instalación de MariaDB

Para una instalación en **Ubuntu** utilizamos el comando:

```
sudo apt-get install mariadb-server
```

Para iniciar el servicio:

```
sudo service mariadb start
```

En la instalación inicial no es necesario escribir contraseña (aunque evidentemente es recomendable y en próximos temas lo haremos) para acceder:

```
sudo mysql -u root
```

```
manuel@manuel-VirtualBox:~$ sudo mysql -u root
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 43
Server version: 10.1.34-MariaDB-0ubuntu0.18.04.1 Ubuntu 18.04

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

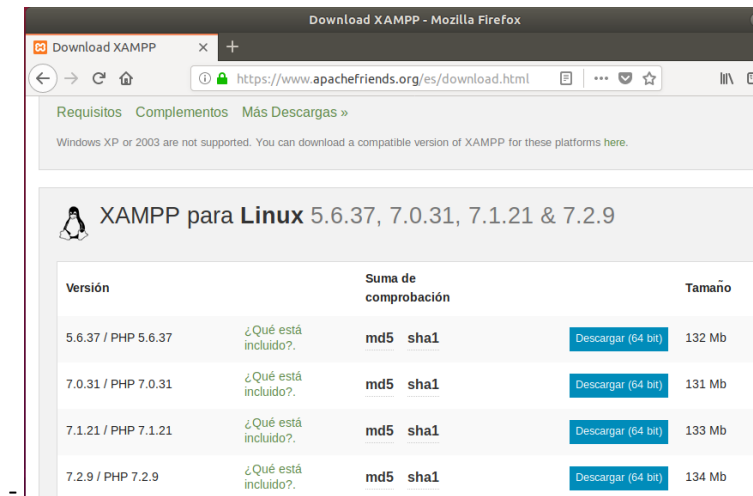
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> use information_schema
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [information_schema]> show tables;
```

También podemos **instalar el paquete XAMPP que incluye MariaDB**, Apache y phpmyadmin, este último es un paquete en PHP que permite trabajar en un entorno web con MariaDB.

En Ubuntu, descargamos el paquete XAMPP de la página de [apachefriends.org](https://www.apachefriends.org)



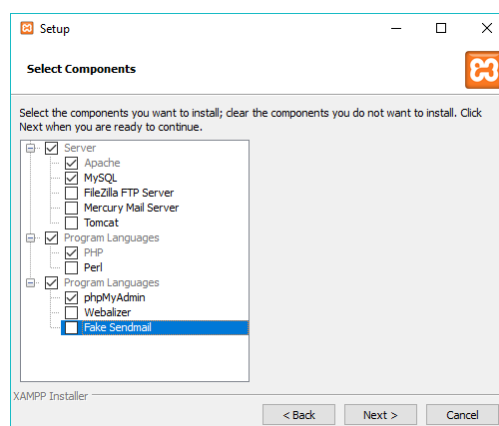
El fichero lo copiamos a nuestro directorio home, damos permisos de ejecución y ejecutamos:

```
manuel@manuel-VirtualBox:~$ ls
Descargas  ejemplos.desktop  Plantillas  xampp-linux-x64-7.2.9-0-installer.run
Documentos Imágenes          Público
Escritorio Música           Videos
manuel@manuel-VirtualBox:~$ chmod +x xampp-linux-x64-7.2.9-0-installer.run
manuel@manuel-VirtualBox:~$ sudo ./xampp-linux-x64-7.2.9-0-installer.run
```

Para acceder por comandos al servidor:

`/opt/lamp/bin/mysql -u root`

En Windows, de forma similar, descargamos el paquete XAMPP y lo ejecutamos ([Descarga aquí la versión 7.3.9](#)). Basta con que instalemos MariaDB, pero para tener un acceso web podemos marcar también Apache, PHP y phpmyadmin.



Para acceder por comandos desde Windows:

- Ejecutamos MariaDB en la ventana de terminal (podemos modificar la ruta de ejecución de comandos a través de **la variable PATH**, puedes consultar el Anexo I de

este tema.). Para ello, desde el directorio `c:\xampp\mysql\bin` tecleamos “`mysql -u root`”. En el caso de tener contraseña el comando será “`mysql -u root -p password`”, dónde sustituimos password por la contraseña.

- Si hemos accedido bien al cliente, en la línea de comandos aparecerá el texto “**MariaDB [(none)]>**” que indicará que el cliente está esperando comandos para ejecutar sobre el servidor.

Para trabajar a través de web, tanto en Ubuntu como en Windows, basta con navegar desde el propio equipo a la dirección “`http://localhost`” o desde otro equipo a [http://ip del servidor](http://ip_del_servidor)



Variables

MariaDB cuenta con distintos tipos de variables que se pueden modificar para ajustar el servidor a nuestras necesidades. Una de las tareas más importantes del administrador es optimizar los valores de dichas variables.

Variables de sistema

Muchas de estas variables pueden modificarse dinámicamente mientras el servidor se está ejecutando. Esto a menudo permite variar la operación del servidor sin tener que detenerlo y reiniciarlo.

El servidor MariaDB mantiene dos clases de variables de sistema: las **variables globales** que afectan a la operación general del servidor y las **variables de sesión** que actúan sobre la operación en conexiones de clientes individuales.

Cuando el servidor arranca, inicializa todas las variables globales a los valores indicados en el fichero de opciones, `my.ini` en Windows (`xampp\mysql\bin`) o `my.cnf` en GNU/Linux (`/opt/lampp/etc`)

Estas variables globales también pueden ser modificadas dinámicamente, una vez iniciado el servidor, mediante la línea de comandos. Para cambiar una variable global debemos tener el privilegio SUPER. Los valores de las variables globales pueden establecerse usando varias sintaxis diferentes:

```
MariaDB [BDenuso] >SET GLOBAL nombre_variable=valor_variable;
```

```
MariaDB [BDenuso] > SET @@global.nombre_variable=valor_variable;
```

El servidor también mantiene un conjunto de variables de sesión para cada cliente que se conecta. **Las variables de sesión de cliente se inicializan en el momento de conectarse, con el valor actual de la correspondiente variable global.** No se requieren privilegios especiales para establecer el valor una variable de sesión, pero un cliente puede modificar solamente sus propias variables, no las de otros clientes.

Para establecer el valor de una variable de sesión, debe emplearse una de las siguientes sintaxis:

```
MariaDB [BDenuso] > SET SESSION nombre_variable = valor_variable;
```

```
MariaDB [BDenuso] > SET @@session.nombre_variable= valor_variable;
```

```
MariaDB [BDenuso] > SET nombre_variable= valor_variable;
```

Un cambio en una variable global es visible para cualquier cliente que acceda a esa variable. Sin embargo, afectará solamente a las variables de sesión de las conexiones que se realicen después del cambio. No afectará a las variables de sesión de los clientes que ya estaban conectados cuando se realizó el cambio. Ni siquiera afectará al cliente que emitió la sentencia SET GLOBAL hasta que no reinicie sesión.

Si al establecer el valor de una variable no se utiliza GLOBAL, por defecto se asume SESSION.

Para recuperar el valor de una variable global debe utilizarse una de las siguientes sentencias:

```
MariaDB [BDenuso] > SELECT @@global.nombre_variable;
```

```
MariaDB [BDenuso] > SHOW GLOBAL VARIABLES like '%nombre_variable%';
```

```
SHOW GLOBAL VARIABLES like '%max%';
```

Para recuperar el valor de una variable de sesión debe utilizarse una de las siguientes sentencias:

```
MariaDB [BDenuso] > SELECT @@nombre_variable;
```

```
MariaDB [BDenuso] > SELECT @@session.nombre_variable;
```

```
MariaDB [BDenuso] > SHOW SESSION VARIABLES like '%nombre_variable%';
```

Cuando se recupera una variable con SELECT @@nombre_var (o sea, no se especifica global., session., o local.), MariaDB devuelve el valor de SESSION si existe y el valor GLOBAL en otro caso.

Igualmente, en el caso de SHOW VARIABLES, si no se especifica GLOBAL, MariaDB devuelve los valores de SESSION.

Nótese que la palabra clave GLOBAL se requiere para establecer el valor de variables globales pero no para recuperar dicho valor. Esto es así para evitar futuros problemas ya que en el caso

de variables que sean tanto de sesión como globales, podría darse el caso de que un usuario con privilegios SUPER podría cambiar accidentalmente el valor de una variable global al querer cambiar solamente la variable de sesión de su propia conexión.

También se pueden consultar las variables consultando las tablas GLOBAL_VARIABLES y SESSION_VARIABLES de la base de datos **information_schema**.

```
C:\xampp\mysql\bin>mysql -u root
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 5
Server version: 10.1.35-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> use information_schema;
Database changed
MariaDB [information_schema]> select variable_name,global_value from system_variables where variable_name like 'aria%';
+-----+-----+-----+
| variable_name | global_value |
+-----+-----+-----+
| ARIA_PAGE_CHECKSUM | ON |
| ARIA_PAGECACHE_DIVISION_LIMIT | 100 |
| ARIA_USED_FOR_TEMP_TABLES | ON |
```

```
SELECT * FROM `GLOBAL_VARIABLES` WHERE `VARIABLE_NAME` like '%max%' order by 1
```

También se pueden consultar estas variables con el comando **mysqladmin**

Variables de estado

Además de las variables de sistema el servidor MariaDB incorpora variables de estado que proveen información sobre sus operaciones.

Se pueden ver estas variables y sus valores con la sentencia SHOW STATUS.

Muchas de estas variables pueden ser inicializadas a 0 mediante la sentencia FLUSH STATUS.

Las variables de estado se pueden consultar usando show status like "patrón";

```
SHOW STATUS like '%conn%';
```

Estas variables se clasifican en los siguientes grupos principales, que es la palabra con la que comienza el nombre de la variable:

- Tipo Com: Son variables que contabilizan el número de veces que se ejecuta un comando.
- Tipo Handler: Estas variables se relacionan con operaciones de lectura y escritura sobre tablas.
- Tipo InnoDB: Facilitan la optimización del motor InnoDB.
- Tipo Key: Cuentan el número de operaciones relacionadas con índices.
- Tipo Qcache: Contabilizan el número de operaciones sobre la caché de consultas (query cache).
- Tipo Ssl: Son variables relacionadas con la criptografía de clave asimétrica (SSL).

- Tipo Threads: Variables relacionadas con los hilos o conexiones creadas en el servidor.

Interfaces estándar: conectores ODBC y JDBC

La norma ODBC es un estándar de acceso a bases de datos con el propósito de proporcionar independencia entre los sistemas de bases de datos y los programas de aplicación. ODBC se encarga de incorporar un interfaz para comunicar ambos lados.

Cada sistema de bases de datos compatible ODBC proporciona una biblioteca que se comunica con el programa cliente. Cuando el programa llama a la API ODBC, la biblioteca se comunica con el servidor para realizar la consulta y obtener los resultados.

La norma JDBC establece una API para que la usen los programas Java y se conecten a bases de datos con total independencia del sistema operativo empleado, que es una de las características del lenguaje de programación Java.

El Diccionario de Datos

Con el fin de que los usuarios conozcan las características de las distintas bases de datos incluidas en el SGBD, es necesario que éste incorpore un sistema que permita explorarlas.

Para ello, existe el **diccionario de datos**, que no es más que una interfaz entre el SGBD y el programador de aplicaciones que incorpora una lista de propiedades de cada base de datos.

```
MariaDB [(none)]> use information_schema;
Database changed
MariaDB [information_schema]> SHOW TABLES;
+-----+
| Tables_in_information_schema |
+-----+
| ALL_PLUGINS                  |
| APPLICABLE_ROLES             |
| CHARACTER_SETS               |
| CHECK_CONSTRAINTS            |
| COLLATIONS                   |
| COLLATION_CHARACTER_SET_APPLICABILITY |
| COLUMNS                     |
| COLUMN_PRIVILEGES             |
| ENABLED_ROLES                 |
| ENGINES                       |
| EVENTS                       |
| FILES                         |
| GLOBAL_STATUS                 |
| GLOBAL_VARIABLES              |
| KEY_CACHES                    |
| KEY_COLUMN_USAGE              |
| OPTIMIZER_TRACE               |
+-----+
```

Un diccionario de datos debe incorporar la siguiente información:

- Las diferentes bases de datos del sistema.
- Para cada base de datos, la lista de tablas que contiene y sus propiedades.
- Para cada tabla, la lista de columnas que contiene con sus propiedades.
- Lista de restricciones de integridad y relaciones que hay entre las tablas.
- Listado de índices de cada tabla.
- Lista de usuarios y sus permisos.
- Listado de procedimientos almacenados, triggers y eventos.
- Cada SGBD tendrá información extra particular, como pueden ser estadísticas, registros de acceso, etc. Depende de lo que cada SGBD implemente.

La manera de acceder a esta información es mediante consultas de SQL tradicionales, por lo que el diccionario de datos es tratado de forma parecida a una base de datos convencional.

Cada SGBD crea una base de datos especial con esta información. El diccionario de datos en MariaDB (como en MySQL) está contenido en la base de datos **information_schema** que se crea en el momento de la instalación.

Dado que en `information_schema` se almacenan todos los datos relacionados con el SGBD y con el fin de evitar que se ejecuten acciones que provoquen que el sistema deje de funcionar por modificar algún elemento clave, en esta base de datos se utilizan vistas de solo lectura en las que no se puede modificar su contenido. La única forma de acceder al contenido de sus tablas es con `SELECT`. No puede insertar, actualizar o borrar su contenido.

Cada usuario MySQL tiene derecho a acceder a estas tablas, pero solo a los registros que se corresponden a los objetos a los que tiene permiso de acceso.

Algunas tablas de la BBDD **information_schema** son:

- **SCHEMATA**. Muestra las bases de datos (o esquemas) del servidor.
- **TABLES**. Proporciona información sobre las tablas de las bases de datos.
- **COLUMNS**. Informa sobre las columnas de las tablas.
- **USER_PRIVILEGES**. Proporciona información sobre permisos globales.
- **SCHEMA_PRIVILEGES**. Proporciona información acerca de los permisos sobre bases de datos.
- **TABLE_PRIVILEGES**. Informa sobre los permisos a nivel de tabla.
- **COLUMN_PRIVILEGES**. Informa de permisos otorgado a nivel de columna.
- **TABLE_CONSTRAINTS**. Describe que tablas tienen restricciones. El valor `CONSTRAINT_TYPE` puede ser `UNIQUE`, `PRIMARY KEY` O `FOREIGN KEY`.
- **KEY_COLUMN_USAGE**. Describe que columnas clave tienen restricciones.

La siguiente prueba es muy interesante:

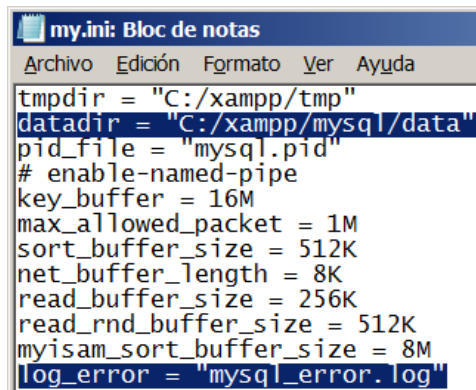
Crear tabla en base de datos test y buscar en el Diccionario de Datos qué ha guardado en *tables*, *columns*, *table_constraints*,

Ficheros LOG

Para poder observar el funcionamiento del SGBD y depurar posibles problemas, MariaDB proporciona ficheros de log, o ficheros de registros. Esto es muy común en los servicios que no disponen de interfaz gráfica. Su método de comunicación con el DBA es a través de estos ficheros, que reportan cada evento que ocurre en el SGBD; de esta manera el DBA puede averiguar si el rendimiento del gestor es bueno, o si hay problemas en algunas partes de la base de datos, si los usuarios están actuando correctamente, si hay problemas de seguridad, etc.

Los ficheros de log más interesantes en MariaDB son los siguientes:

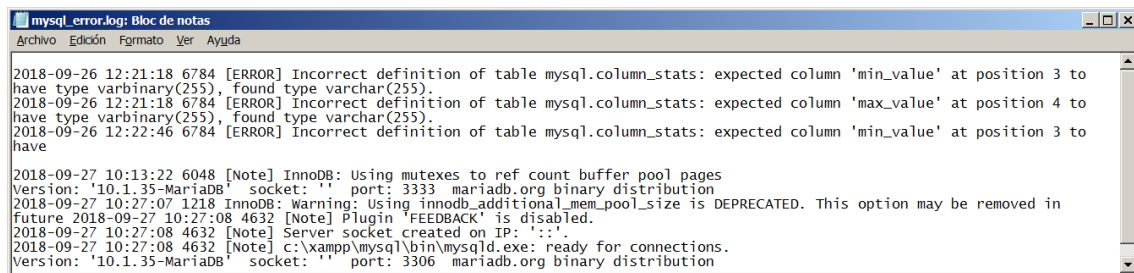
- **LOG DE ERRORES**: Muestra información sobre los sucesos anormales que se han producido a nivel de sistema en el servidor de la base de datos, por ejemplo, que al arrancar el servidor no encuentre el fichero de configuración. El nombre de este fichero es el que aparece en la variable de sistema `log_error`, la ubicación aparece en la variable `datadir`. Si ocurriera algún problema, el DBA debe, como primer paso, consultar este log para averiguar qué ha ocurrido. El log de errores proporciona tres tipos de mensajes:
 - ✓ **Note**: Simple información de un evento sucedido, por ejemplo, que se arranca/apaga un módulo o información relevante sobre el procesador.
 - ✓ **Warning**: Son avisos, deben servir como mensajes de cautela para el DBA. Por ejemplo, que algún parámetro está obsoleto o depreciado.
 - ✓ **Error**: Errores críticos que impiden el funcionamiento normal del sistema.



```

my.ini: Bloc de notas
Archivo  Edición  Formato  Ver  Ayuda
tmpdir = "C:/xampp/tmp"
datadir = "C:/xampp/mysql/data"
pid_file = "mysql.pid"
# enable-named-pipe
key_buffer = 16M
max_allowed_packet = 1M
sort_buffer_size = 512K
net_buffer_length = 8K
read_buffer_size = 256K
read_rnd_buffer_size = 512K
myisam_sort_buffer_size = 8M
log_error = "mysql_error.log"

```



```

mysql_error.log: Bloc de notas
Archivo  Edición  Formato  Ver  Ayuda
2018-09-26 12:21:18 6784 [ERROR] Incorrect definition of table mysql.column_stats: expected column 'min_value' at position 3 to
have type varbinary(255), found type varchar(255).
2018-09-26 12:21:18 6784 [ERROR] Incorrect definition of table mysql.column_stats: expected column 'max_value' at position 4 to
have type varbinary(255), found type varchar(255).
2018-09-26 12:22:46 6784 [ERROR] Incorrect definition of table mysql.column_stats: expected column 'min_value' at position 3 to
have
2018-09-27 10:13:22 6048 [Note] InnoDB: Using mutexes to ref count buffer pool pages
Version: '10.1.35-MariaDB' socket: '' port: 3333 mariadb.org binary distribution
2018-09-27 10:27:07 1218 InnoDB: Warning: Using innodb_additional_mem_pool_size is DEPRECATED. This option may be removed in
future
2018-09-27 10:27:08 4632 [Note] Plugin 'FEEDBACK' is disabled.
2018-09-27 10:27:08 4632 [Note] Server socket created on IP: '::'.
2018-09-27 10:27:08 4632 [Note] c:\xampp\mysql\bin\mysqld.exe: ready for connections.
Version: '10.1.35-MariaDB' socket: '' port: 3306 mariadb.org binary distribution

```

- **LOG GENERAL DE CONSULTAS:** Almacena todas las consultas solicitadas a la base de datos. Se registran todas las operaciones SQL que han realizado todos los usuarios de la base de datos. De esta manera, el DBA puede indagar sobre las consultas que se han realizado antes de que se produjera cierta situación indeseable. Por defecto, MariaDB no tiene el log general de consultas activado, ya que activarlo repercute en el rendimiento del servidor de un modo muy negativo. Las **variables de sistema** asociadas al log general de consultas son:
 - ✓ **general_log:** El registro general de consultas está activado cuando esta variable toma el valor 1.
 - ✓ **log_output:** Admite tres valores, FILE, TABLE o NONE. Con el valor FILE, el SGBD escribe el log general de consultas directamente en un fichero. Con el valor TABLE, el SGBD almacena las consultas ejecutadas en una tabla. Si está configurado para usar tablas, se puede consultar el log con la siguiente consulta: `select * from mysql.general_log`. Con el valor NONE el SGBD no almacena ninguna consulta aunque el registro de consultas esté activado.
 - ✓ **general_log_file:** Si log_output tiene el valor FILE, la ubicación del fichero se toma del parámetro general_log_file.

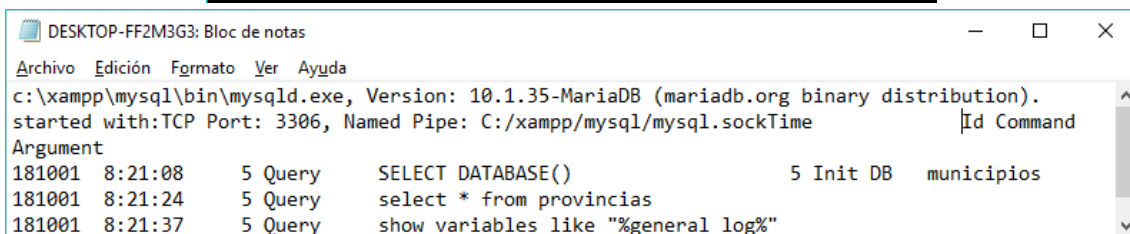
```

MariaDB [(none)]> show variables like "%general_log%";
+-----+-----+
| Variable_name | Value |
+-----+-----+
| general_log   | OFF   |
| general_log_file | DESKTOP-FF2M3G3.log |
+-----+-----+
2 rows in set (0.00 sec)

MariaDB [(none)]> show variables like "%log_output%";
+-----+-----+
| Variable_name | Value |
+-----+-----+
| log_output    | FILE  |
+-----+-----+
1 row in set (0.00 sec)

MariaDB [(none)]> set global general_log=1;
Query OK, 0 rows affected (0.01 sec)

```



- **LOG DE CONSULTAS LENTAS:** Almacena las consultas que pueden ser consideradas como pesadas en tiempo de ejecución. Este tipo de registro tampoco está activado y se puede configurar mediante las siguientes variables de sistema de MariaDB:
 - ✓ **slow_query_log:** El log de consultas lentas está activado cuando esta variable toma el valor 1.
 - ✓ **slow_query_log_file:** Contiene la ubicación del fichero que almacena las consultas lentas.
 - ✓ **long_query_time:** Una consulta se considera lenta cuando supera el tiempo especificado en segundos en esta variable. Es muy fácil generar una consulta lenta gracias a la función sleep (segundos), que retarda el tiempo de ejecución de cualquier consulta. Ejemplo: select sleep (4), user from mysql.user;

```

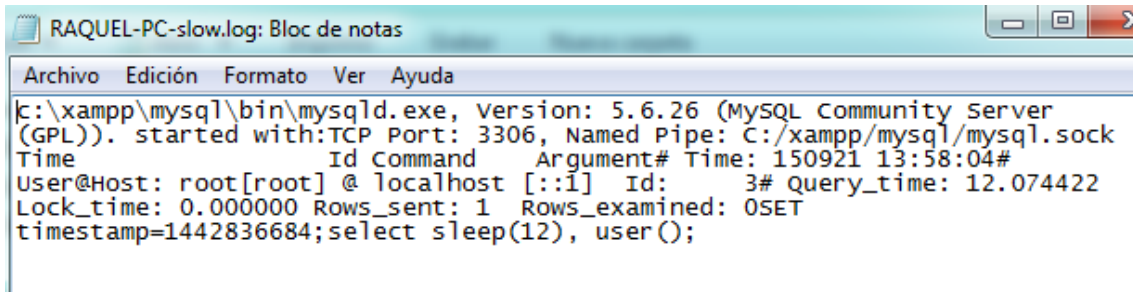
mysql> set global slow_query_log=1;
Query OK, 0 rows affected (0.06 sec)

mysql> show variables like "%slow_query%";
+-----+-----+
| Variable_name | Value |
+-----+-----+
| slow_query_log | ON    |
| slow_query_log_file | C:\xampp\mysql\data\RAQUEL-PC-slow.log |
+-----+-----+
2 rows in set (0.00 sec)

mysql> show variables like "%long_query_time%";
+-----+-----+
| Variable_name | Value |
+-----+-----+
| long_query_time | 10.000000 |
+-----+-----+
1 row in set (0.00 sec)

```

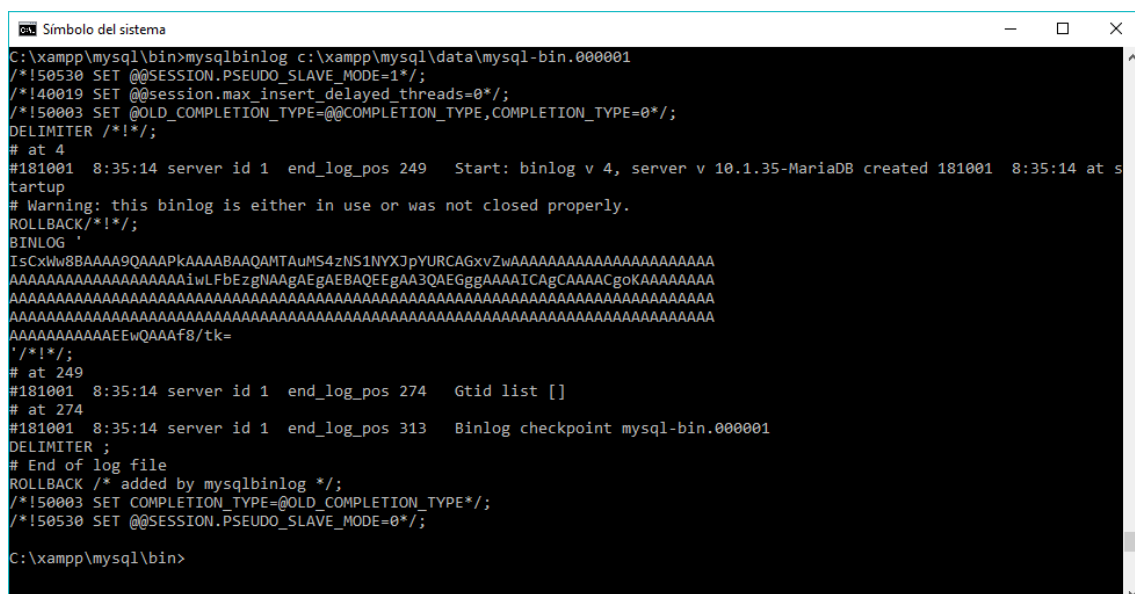
```
mysql> select sleep(12), user();
+-----+-----+
| sleep(12) | user() |
+-----+-----+
|          | root@localhost |
+-----+-----+
1 row in set (12.07 sec)
```



```
RAQUEL-PC-slow.log: Bloc de notas
Archivo Edición Formato Ver Ayuda
c:\xampp\mysql\bin\mysqld.exe, Version: 5.6.26 (MySQL Community Server
(GPL)). started with:TCP Port: 3306, Named Pipe: C:/xampp/mysql/mysql.sock
Time Id Command Argument# Time: 150921 13:58:04#
User@Host: root[root] @ localhost [::1] Id: 3# Query_time: 12.074422
Lock_time: 0.000000 Rows_sent: 1 Rows_examined: 0SET
timestamp=1442836684;select sleep(12), user();
```

- **LOG BINARIO:** Almacena cualquier evento que pueda producir una modificación de los datos, y sirve, por ejemplo, para replicar bases de datos o para restaurar una copia de seguridad del sistema en cualquier punto anterior. No almacena consultas de datos, solamente aquellas operaciones que cambien o puedan cambiar los datos. Por ejemplo, en el log binario se almacenaría una operación delete aunque el resultado fuera 0 filas afectadas. Se llama binario porque la información no se almacena en texto plano sino en formato binario, por lo que para examinar su contenido se debe utilizar una herramienta llamada **mysqlbinlog** o el comando **show binlog events**. Hay que tener en cuenta que activar este registro perjudica ligeramente el rendimiento del sistema, aunque no tanto como el log de consultas. Para configurar el log binario hay que incluir la opción **--log-bin[=nombre_fichero]** en el fichero **my.ini** de Windows (**my.cnf** en GNU/Linux). MariaDB generará ficheros de un tamaño igual al valor de la variable **max_binlog_size** y agregará una extensión numérica al nombre del registro binario para numerar los distintos ficheros binarios.

```
# Replication Master Server (default)
# binary logging is required for replication
# log-bin deactivated by default since XAMPP 1.4.11
log-bin=mysql-bin
max_binlog_size = 1G
```



```
Símbolo del sistema
C:\xampp\mysql\bin>mysqlbinlog c:\xampp\mysql\data\mysql-bin.000001
/*!50530 SET @@SESSION.PSEUDO_SLAVE_MODE=1*/;
/*!40019 SET @@session.max_insert_delayed_threads=0*/;
/*!50003 SET @OLD_COMPLETION_TYPE=@@COMPLETION_TYPE,COMPLETION_TYPE=0*/;
DELIMITER /*!*/;
# at 4
#181001 8:35:14 server id 1 end_log_pos 249 Start: binlog v 4, server v 10.1.35-MariaDB created 181001 8:35:14 at s
tartup
# Warning: this binlog is either in use or was not closed properly.
ROLLBACK/*!*/;
BINLOG '
IsCxWw8BAAAA9QAAAPKAAAABAAQANTAuMS4zNS1NYXJpYURCAGxvZwAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAiWLFbEzgNAAgAEgAEBAQEgAA3QAEggAAAAICAgCAAAACgoKAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAEEwQAAf8/tk=
'/*!*/;
# at 249
#181001 8:35:14 server id 1 end_log_pos 274 Gtid list []
# at 274
#181001 8:35:14 server id 1 end_log_pos 313 Binlog checkpoint mysql-bin.000001
DELIMITER ;
# End of log file
ROLLBACK /* added by mysqlbinlog */;
/*!50003 SET COMPLETION_TYPE=@OLD_COMPLETION_TYPE*/;
/*!50530 SET @@SESSION.PSEUDO_SLAVE_MODE=0*/;
C:\xampp\mysql\bin>
```


Es importante comprender que el **log binario no restaura datos, sino restaura acciones**. Por ejemplo (teniendo el log binario activo):

1.- Si yo cargo un script de una BD, después hago una serie de INSERT INTO y tras ello hago un DROP de esa BD; yo puedo recuperar la BD con el script original y después ejecutar esos INSERT que estarán grabados en el log binario, por lo que la BD quedará en una situación consistente y correcta.

2.- Si yo cargo un script de una BD, después hago un borrado de datos con DELETE de forma accidental, después hago una serie de INSERT. Con el log binario no podemos recuperar esos datos borrados con DELETE, ya que lo que almacena el log binario son las acciones ejecutadas sobre la BD, no los datos borrados en sí.

Es necesario limpiar estos archivos cada cierto tiempo, antes de hacer copias de seguridad, para que no ocupen excesivo espacio en el servidor. Para realizar esta tarea se emplea la sentencia: **flush logs** o los comandos mysqladmin flush-logs o mysqladmin refresh.

También es interesante almacenar los logs, en otro soporte antes de refrescarlos, porque son una fuente de información sobre el funcionamiento de nuestro servidor, lo que resulta muy útil para optimizar su ejecución.

Otros sistemas de almacenamiento.

Muchas aplicaciones utilizan datos en estructuras XML, los cuales hay que almacenar. HAY PRINCIPALMENTE dos tipos de sistemas de almacenamiento para datos XML:

- Sistemas de almacenamiento basados en documentos: Tienen una estructura irregular y utilizan tipos de datos relativamente simples que dan gran importancia al orden.
- Sistemas de almacenamiento estructurados: Utilizan tipos de datos relativamente complejos y dando poca importancia al orden frente a las relaciones entre los datos.

Los sistemas basados en documentos suelen utilizar principalmente dos estructuras:

- Bases de datos XML: Son bases de datos que usan XML como modelo de datos básico. También es posible construir una capa XML sobre una base de datos relacional.
- Almacenamiento en ficheros planos: Se debe a que XML es un formato de archivo. A pesar de los inconvenientes de los sistemas de archivos para guardar datos, XML proporciona una gran variedad de herramientas que hacen sencillo el acceso y consulta a los datos.

Respecto a los sistemas estructurados, resulta una gran ventaja almacenar datos XML porque hay muchas aplicaciones que utilizan BDR que pueden acceder a estos datos fácilmente.

Convertir datos XML a una estructura relacional es algo muy sencillo, si los datos se han generado desde una estructura relacional y se usó XML como método de intercambio.

Las alternativas que ofrece este tipo de almacenamiento son:

- Almacenamiento como cadena: se trata de almacenar cada elemento hijo del elemento de mayor nivel como una cadena en una tupla separada de la base de datos. Este esquema es fácil de usar, pero tiene el inconveniente de que el sistema de base de datos no conoce el esquema de los elementos almacenados, ya que no aparecen dentro de la estructura lógica, dejando como única opción crear elementos extra para poder realizar la indexación.
- Representación en árbol: se almacenan los datos XML en una estructura de árbol mediante una pareja de relaciones del tipo:
 - nodo(id, tipo, etiqueta, valor).
 - hijo(id_padre, id_hijo,)

De esta manera toda la información se puede representar de forma relacional, y convertir fácilmente las consultas XML a consultas relacionales; con el inconveniente de que cada elemento se divide en muchos trozos.

- Asignación a relaciones: los elementos XML cuyo esquema es conocido se asocian a relaciones, creando una relación para cada uno de ellos y los atributos XML son los atributos de la relación; mientras que para los no conocidos se utiliza una de las estructuras anteriormente citadas creando una estructura híbrida

Copias de seguridad y restauración

Una de las tareas principales de un administrador de BBDD es planificar copias de seguridad y en su caso ejecutar la restauración de estas en caso borrado accidental.

Las copias de seguridad se pueden hacer de muchas formas: copiando los archivos del sistema de ficheros que almacena BBDD con aplicaciones de terceros, exportando la BD a scripts SQL, exportando la BD a formatos binarios, etc.

En MariaDB podemos usar el comando `mysqldump` para hacer el volcado de datos en formato SQL (aunque también puede exportar a CSV u otros formatos) y usaremos el cliente `mysql` para la carga de datos.

Además, como tendremos instalado PHPMysqlAdmin, desde este entorno podremos hacer la misma tarea que con `mysqldump` pero en entorno Web.

Documentación.

De todo servidor se debe guardar una información detallada y actualizada de los componentes que lo forman y de los servicios que proporciona.

En el caso de un servidor de bases de datos, además de guardar los datos correspondientes al hardware de la máquina: capacidad de disco, memoria principal y extendida, placa base, etc. y al software del mismo: sistema operativo y programas en ejecución; se debe guardar también información del SGBD instalado y de las tareas de mantenimiento que se realizan sobre él, como actualizaciones de versiones.

En esta documentación se deben reflejar todos los elementos instalados, reflejando claramente sus versiones y las correspondientes actualizaciones de los programas que se hayan instalado.

Entre la documentación a incluir debe figurar la información sobre las bases de datos que estén instaladas en el servidor. La información sobre las bases de datos debe cubrir el gráfico del modelo Entidad-Relación, el esquema relacional y las características de los campos de datos: tipos de datos, restricciones y relaciones entre ellos; sin olvidarnos de otros objetos de la base de datos (funciones, disparadores, etc.) o de los usuarios de la misma.

Hay herramientas que ayudan a realizar la documentación de la base de datos en su origen y sacar la estructura y otra información de la base de datos.

Nota: Ver en phpmyadmin el diseñador y transformación de entrada.

Para el mantenimiento de parte de esta información, que ayuda a determinar el estado de funcionamiento del servidor, resulta de gran utilidad la obtenida a partir de los ficheros log del servidor de bases de datos.

Apéndice. Ejemplos

Ejemplo 1: Importar la base de datos municipios a MariaDB.

Podrás encontrar el fichero de la base de datos en los recursos de la web de @vanza.

Paso 1: crear la base de datos, para ello abrimos una conexión:

```
mysql -u root
```

```
MariaDB [none]> create database municipios;
```

Paso 2: importar la base de datos desde el símbolo del sistema:

```
mysql -u root -p municipios < ruta\municipios.sql
```

Ejemplo 2: Exportar la base de datos municipios.

Para la exportación de datos disponemos de la utilidad mysqldump que se ejecuta desde el símbolo del sistema:

```
mysqldump -u root municipios > archivocopia.sql
```

Ejemplo 3: Realizar consultas a la base de datos municipios

Mostrar todas las provincias en la tabla de provincias:

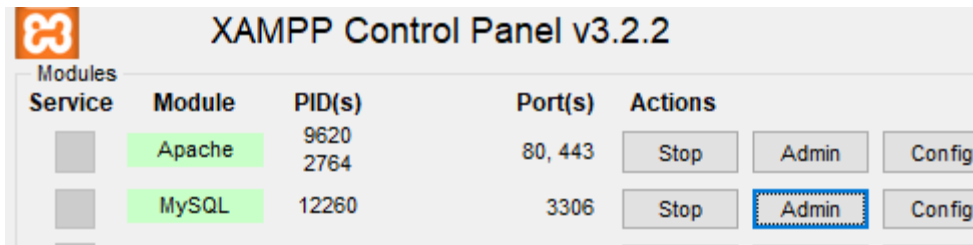
```
MariaDB [none]> use municipios;
```

```
MariaDB [municipios]> select * from provincias;
```

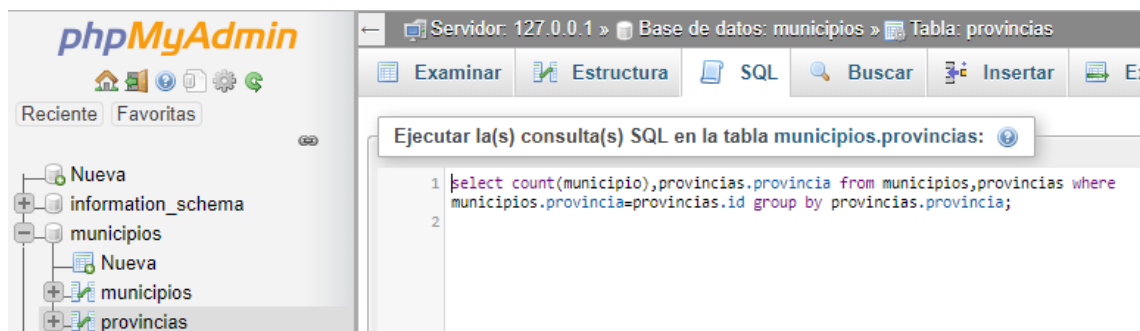
Contar los municipios de cada provincia, mostrando provincia y total

MariaDB [municipios]> **select count(municipio), provincias.provincia from municipios,provincias where municipios.provincia=provincias.id group by provincias.provincia;**

Realizar esta misma consulta desde phpmyadmin: La forma más sencilla de acceder a phpmyadmin es hacer clic en el botón “Admin” del panel de control de xampp, o teclear <http://ipserveridor/phpmyadmin> en cualquier navegador



Una vez en el navegador seleccionamos la base de datos municipios y en la pestaña SQL ejecutamos la sentencia pulsando sobre el botón “continuar”



Ejemplo 4: Mostrar las tablas de la base de datos municipios usando el diccionario de datos.

Abrimos una conexión, seleccionamos la base de datos information_schema y ejecutamos el select:

mysql -u root

MariaDB [none]> **use information_schema;**

MariaDB [information_schema]> **select table_name, table_rows,table_type,engine from tables where table_schema="municipios";**

```
MariaDB [information_schema]> select table_name,table_rows,table_type,engine from tables where table_schema="municipios";
+-----+-----+-----+-----+
| table_name | table_rows | table_type | engine |
+-----+-----+-----+-----+
| municipios |      8325 | BASE TABLE | InnoDB |
| provincias |        52 | BASE TABLE | InnoDB |
+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

Ejemplo 5: Activar las estadísticas, abrir dos sesiones y mostrar usuarios conectados:

mysql -u root (abrimos una conexión)

MariaDB [none]> **show global variables like 'userstat';**

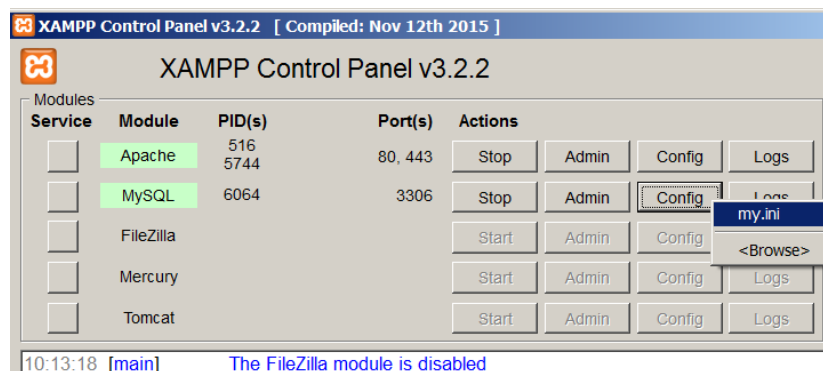
MariaDB [none]> **set global userstat=ON;**

Una vez activada la variable, el servidor comenzará la recopilación de información estadística. Si abrimos 2 conexiones desde dos ventanas del sistema (mysql -u root) y mostramos las estadísticas veremos estas 2 conexiones abiertas

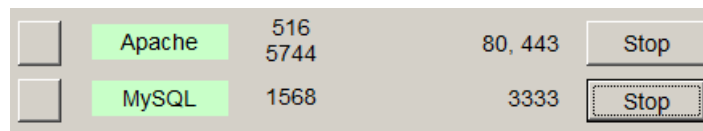
MariaDB [none]> **show user_statistics;**

Ejemplo 6: Cambiar el puerto de escucha de MariaDB a 3333

El puerto de escucha es un parámetro que se indica en el fichero de configuración my.ini (para Windows) o my.cnf (para Linux). Para cambiarlo en Windows (si tenemos instalado xampp):

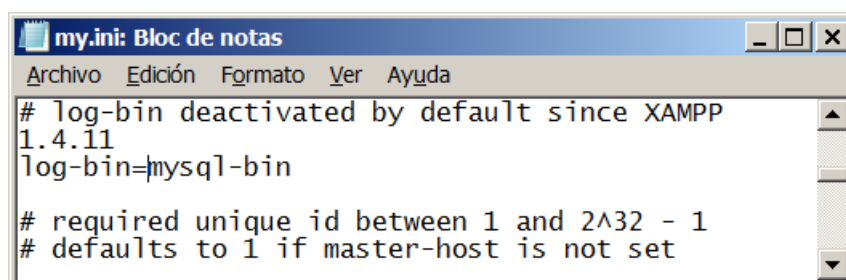


En el fichero my.ini o my.cnf buscamos la línea: port=3306 y la cambiamos por port=3333, guardamos la configuración, paramos y arrancamos de nuevo MariaDB.



Ejemplo 7: Activar el registro binario, sobre la base de datos original municipios haremos unos cambios. Supongamos una caída del servidor. Restauramos los datos originales y realizamos todas las acciones para restaurar la base de datos al estado exacto antes de la caída.

Paso 1: Activar el registro binario descomentando la línea log-bin:



Paso 2: Arrancar el servidor MariaDB y comprobamos que realmente está activado:

MariaDB [none]> **show global variables like '%log_bin%';**

Paso 3: Sobre los 10 primeros registros alfabéticos realizamos cambios:



Se produce la caída. Vamos a restaurar la base de datos de municipios inicial y después ejecutar los cambios para llegar al estado con estos últimos cambios mostrados:

MariaDB [none]> **drop database municipios;** (simulamos caída)

MariaDB [none]> **create database municipios;**

MariaDB [none]> **exit**

C:\xampp\mysql\bin> **mysql -u root -p municipios < ruta\municipios.sql**

C:\xampp\mysql\bin> **mysql -u root**

MariaDB [none]> **show binlog events;**

Si tenemos varios ficheros binlog, es decir, muchas manipulaciones de datos, podemos exportar su contenido a ficheros de texto con:

C:\xampp\mysql\bin> **mysqlbinlog c:\xampp\mysql\data\fichero.00000x > datos.txt**

Buscamos dentro de este fichero los "update" que queremos restaurar, y nos quedamos con la posición "at":

Mostramos a partir de la posición encontrada, esto nos permitirá ver "hasta donde" deseamos restaurar:

MariaDB [none]> **show binlog events in 'registro-binario.000002' from 446590 limit 30;**

```

Selecciónar Símbolo del sistema - mysql -u root

10 rows in set (0.00 sec)
MariaDB [(none)]> show binlog events in 'registro-binario.000002' from 446590 limit 30;

+-----+-----+-----+-----+-----+
| Log_name | Pos | Event_type | Server_id | End_log_pos | Info |
+-----+-----+-----+-----+-----+
| registro-binario.000002 | 446590 | Xid | 1 | 446617 | COMMIT /* xid=6670 */ |
| registro-binario.000002 | 446617 | Gtid | 1 | 446655 | BEGIN GTID 0-1-853 |
| registro-binario.000002 | 446655 | Query | 1 | 446804 | use 'municipios'; UPDATE 'municipios' SET 'municipio' = 'Ababuj123' WHERE 'municipios'. 'id' = 6525 |
| registro-binario.000002 | 446804 | Xid | 1 | 446831 | COMMIT /* xid=6738 */ |
| registro-binario.000002 | 446831 | Gtid | 1 | 446889 | BEGIN GTID 0-1-854 |
| registro-binario.000002 | 446889 | Query | 1 | 447111 | REPLACE INTO 'phmyadmin`.`pma_table_miprefs' (username, db_name, table_name, prefs) VALUES ('root', 'municipios', 'municipios', '{\' |
| registro-binario.000002 | 447111 | Xid | 1 | 447138 | COMMIT /* xid=6744 */ |
| registro-binario.000002 | 447138 | Gtid | 1 | 447176 | BEGIN GTID 0-1-855 |
| registro-binario.000002 | 447176 | Query | 1 | 447326 | use 'municipios'; UPDATE 'municipios' SET 'municipio' = 'Abadin123' WHERE 'municipios'. 'id' = 4208 |
| registro-binario.000002 | 447326 | Xid | 1 | 447353 | COMMIT /* xid=6812 */ |
| registro-binario.000002 | 447353 | Gtid | 1 | 447391 | BEGIN GTID 0-1-856 |
| registro-binario.000002 | 447391 | Query | 1 | 447633 | REPLACE INTO 'phmyadmin`.`pma_table_miprefs' (username, db_name, table_name, prefs) VALUES ('root', 'municipios', 'municipios', '{\' |
| registro-binario.000002 | 447633 | Xid | 1 | 447660 | COMMIT /* xid=6818 */ |
| registro-binario.000002 | 447660 | Gtid | 1 | 447698 | BEGIN GTID 0-1-857 |
| registro-binario.000002 | 447698 | Query | 1 | 447941 | REPLACE INTO 'phmyadmin`.`pma_table_miprefs' (username, db_name, table_name, prefs) VALUES ('root', 'municipios', 'municipios', '{\' |
| registro-binario.000002 | 447941 | Xid | 1 | 447968 | COMMIT /* xid=6950 */ |
| registro-binario.000002 | 447968 | Gtid | 1 | 448006 | BEGIN GTID 0-1-858 |
| registro-binario.000002 | 448006 | Query | 1 | 448248 | REPLACE INTO 'phmyadmin`.`pma_table_miprefs' (username, db_name, table_name, prefs) VALUES ('root', 'municipios', 'municipios', '{\' |
| registro-binario.000002 | 448248 | Xid | 1 | 448275 | COMMIT /* xid=6986 */ |
| registro-binario.000002 | 448275 | Gtid | 1 | 448313 | BEGIN GTID 0-1-859 |
| registro-binario.000002 | 448313 | Query | 1 | 448463 | use 'municipios'; UPDATE 'municipios' SET 'municipio' = 'Abades_333' WHERE 'municipios'. 'id' = 5845 |
| registro-binario.000002 | 448463 | Xid | 1 | 448490 | COMMIT /* xid=6954 */ |
| registro-binario.000002 | 448490 | Gtid | 1 | 448528 | BEGIN GTID 0-1-860 |
| registro-binario.000002 | 448528 | Query | 1 | 448770 | REPLACE INTO 'phmyadmin`.`pma_table_miprefs' (username, db_name, table_name, prefs) VALUES ('root', 'municipios', 'municipios', '{\' |
| registro-binario.000002 | 448770 | Xid | 1 | 448797 | COMMIT /* xid=6960 */ |
| registro-binario.000002 | 448797 | Gtid | 1 | 448835 | GTID 0-1-861 |
| registro-binario.000002 | 448835 | Query | 1 | 448928 | drop database municipios |
| registro-binario.000002 | 448928 | Gtid | 1 | 448966 | GTID 0-1-862 |
| registro-binario.000002 | 448966 | Query | 1 | 449061 | create database municipios |
| registro-binario.000002 | 449061 | Gtid | 1 | 449099 | GTID 0-1-863 |

30 rows in set (0.00 sec)
MariaDB [(none)]>

```

En nuestro caso tenemos que restaurar desde la posición 446655 hasta la posición 448463, para ello ejecutamos el comando:

C:\xampp\mysql\bin>mysqlbinlog c:\xampp\mysql\data\registro-binario.000002 --start-position=446655 --stop-position=448463 | mysql -u root

```

Símbolo del sistema - mysql -u root
MariaDB [municipios]> select * from municipios order by municipio limit 10;

+-----+-----+-----+
| provincia | municipio | id |
+-----+-----+-----+
| 44 | Ababuj | 6525 |
| 40 | Abades | 5845 |
| 10 | Abadia | 1542 |
| 27 | Abadin | 4208 |
| 48 | Abadiño | 7456 |
| 31 | Abóigar | 4599 |
| 9 | Abajas | 1171 |
| 26 | Abalos | 4034 |
| 20 | Abaltzisketa | 3126 |
| 19 | Abónades | 2838 |

10 rows in set (0.00 sec)

MariaDB [municipios]> exit
Bye

C:\xampp\mysql\bin>
'm' no se reconoce como un comando interno o externo,
programa o archivo por lotes ejecutable.

C:\xampp\mysql\bin>mysqlbinlog c:\xampp\mysql\data\registro-binario.000002 --start-position=446655 --stop-position=448463 | mysql -u root

C:\xampp\mysql\bin>mysql -u root
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MariaDB connection id is 1151
Server version: 10.1.35-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> use municipios;
Database changed
MariaDB [municipios]> select * from municipios order by municipio limit 10;

+-----+-----+-----+
| provincia | municipio | id |
+-----+-----+-----+
| 44 | Ababuj123 | 6525 |
| 40 | Abades | 5845 |
| 10 | Abadia | 1542 |
| 27 | Abadin123 | 4208 |
| 48 | Abadiño | 7456 |
| 31 | Abóigar | 4599 |
| 9 | Abajas | 1171 |
| 26 | Abalos | 4034 |
| 20 | Abaltzisketa | 3126 |
| 19 | Abónades | 2838 |

10 rows in set (0.00 sec)

```

Ejemplo 8: Comparar el uso de motor InnoDB y CSV para almacenar tablas

Crear una tabla en phpMyAdmin con motor de almacenamiento InnoDB y otra con motor CSV.

Seleccionar desde las columnas centrales

Seleccionar desde las columnas centrales

Seleccionar desde las columnas centrales

Comentarios de la tabla: Cotejamiento: Motor de almacenamiento: InnoDB

definición de la PARTICIÓN:

Añadir un par de tuplas a cada tabla creada en en punto anterior.

Acceder al **directorio donde MariaDB está almacenando físicamente la BBDD en forma de ficheros** (el directorio de almacenamiento está indicado en my.cnf o my.ini).

Abrir los ficheros de almacenamiento de las tablas creadas y observar qué y cómo han almacenado los datos.

Ejemplo 9. Mostrar los datos en filas en lugar de columnas.

Si alguna vez tenéis que recoger datos de tablas con muchas columnas y no se ve bien en el terminal, el comando SQL puede terminar en **\G** en lugar de **;** para que os muestre la información en horizontal en lugar de vertical.

```

MariaDB [empresasex]> SELECT * FROM listadoempresas LIMIT 1;
+-----+-----+-----+-----+-----+-----+-----+-----+
| CIF   | NOMBRE                                     | FORMA_JURIDICA |
|-----+-----+-----+-----+-----+-----+-----+
| B10225662 | CENTRO PARA EL DESARROLLO DE EDUCACION EUROPEA | Sociedad limitada | | | |
| 1998 | 8520 | 1 | 0 | MALPARTIDA DE CACERES | Caceres |
| CARRETERA NACIONAL 521 | NULL | NULL | Extremadura |
| ESPANA | 10910 | EXTREMADURA |
+-----+-----+-----+-----+-----+-----+
1 row in set (0.000 sec)

```



```

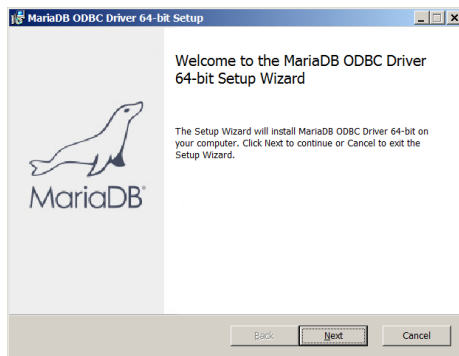
MariaDB [empresasex1]> SELECT * FROM listadoempresas LIMIT 1\G
***** 1. row *****
      CIF: B10225662
    NOMBRE: CENTRO PARA EL DESARROLLO DE EDUCACION EUROPEA
  FORMA_JURIDICA: Sociedad limitada
    FECHA_CONST: 1998
      CNAE: 8520
      N_EMP: 1
    IMP_VENTAS: 0
    MUNICIPIO: MALPARTIDA DE CACERES
    PROUINCIA: Cádiz
    DIRECCION: CARRETERA NACIONAL 521
    EXPORTADORA:
      WEB: NULL
    TELEFONO: NULL
  COMUNIDAD_AUTONOMA: Extremadura
      PAIS: ESPANA
    CODIGO_POSTAL: 10910
    euroregion: EXTREMADURA
1 row in set (0.001 sec)

MariaDB [empresasex1]>

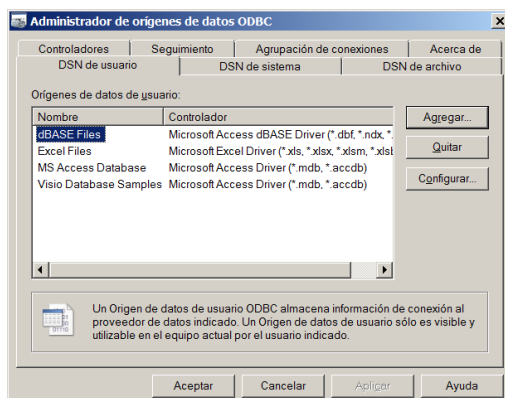
```

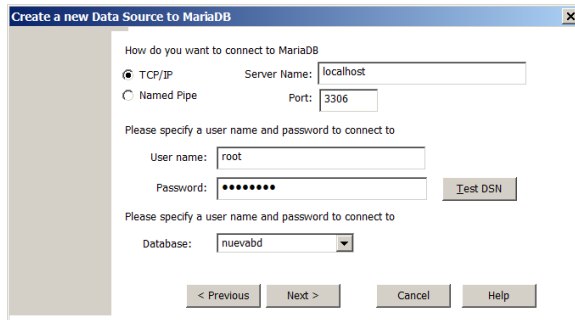
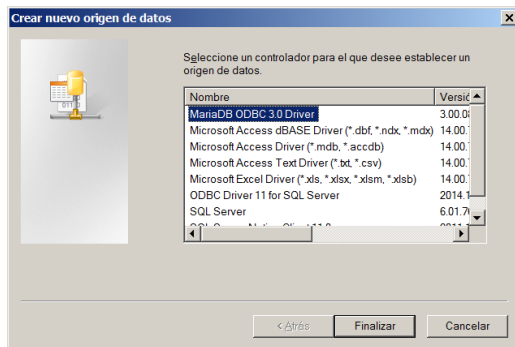
Ejemplo 10: Conectar y exportar base de datos Base a MariaDb

Paso 1: descargar Maria ODBC

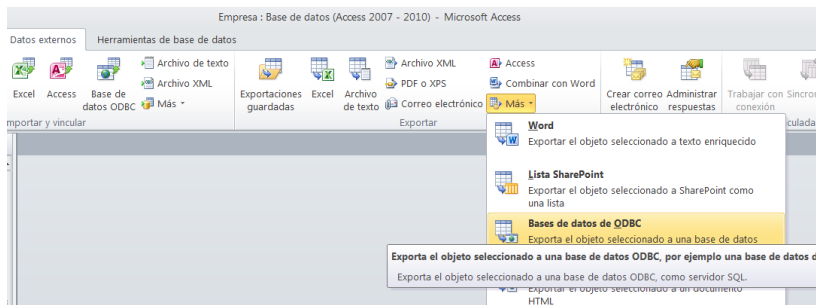


Crear un acceso ODBC a MariaDB





Desde Access exportamos una a una todas las tablas



Ejemplo 11. Instalación segura de MariaDB en entornos Linux

En sistemas Linux, existe la posibilidad de ejecutar un script que realiza una serie de configuraciones para hacer que la instalación de MariaDB quede bien segura y protegida. Con este script podemos poner contraseña al root (por defecto viene sin contraseña), eliminar usuarios anónimos, etc.

El script se ejecuta **con mariadb-secure-installation o mysql-secure-installation**.

```
SERVERS IN PRODUCTION USE! PLEASE READ EACH STEP CAREFULLY!

In order to log into MariaDB to secure it, we'll need the current
password for the root user. If you've just installed MariaDB, and
you haven't set the root password yet, the password will be blank,
so you should just press enter here.

Enter current password for root (enter for none):
OK, successfully used password, moving on...

Setting the root password ensures that nobody can log into the MariaDB
root user without the proper authorisation.

You already have a root password set, so you can safely answer 'n'.

Change the root password? [Y/n] n
... skipping.

By default, a MariaDB installation has an anonymous user, allowing anyone
to log into MariaDB without having to have a user account created for
them. This is intended only for testing, and to make the installation
go a bit smoother. You should remove them before moving into a
production environment.

Remove anonymous users? [Y/n] y
... Success!

Normally, root should only be allowed to connect from 'localhost'. This
ensures that someone cannot guess at the root password from the network.

Disallow root login remotely? [Y/n] y
... Success!

By default, MariaDB comes with a database named 'test' that anyone can
access. This is also intended only for testing, and should be removed
before moving into a production environment.

Remove test database and access to it? [Y/n] y
- Dropping test database...
... Success!
- Removing privileges on test database...
... Success!

Reloading the privilege tables will ensure that all changes made so far
will take effect immediately.

Reload privilege tables now? [Y/n] y
... Success!

Cleaning up...

All done! If you've completed all of the above steps, your MariaDB
installation should now be secure.

Thanks for using MariaDB!
[root@TecMint ~]#
```

Más información en: <https://computingforgeeks.com/how-to-install-mariadb-on-debian/>

Ejemplo 12. Copias de BBDD con mysqldump

Si ejecutamos **mysqldump --help** podemos comprobar la gran cantidad de opciones que contiene, aunque podemos hacer un uso básico de él como se muestra a continuación.

- En la siguiente captura hacemos copia de la BD prueba y dentro del cliente MariaDB la eliminamos. Usamos el redireccionador > para hacer la exportación.

```
c:\xampp\mysql\bin>mysqldump -u root --add-drop-database --databases prueba > copia.sql

c:\xampp\mysql\bin>mysql -u root
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 303
Server version: 10.4.6-MariaDB mariadb.org binary distribution
Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
MariaDB [(none)]> DROP DATABASE prueba;
Query OK, 2 rows affected (0.029 sec)
```

- En la siguiente volvemos a recuperar la BD prueba con el cliente mysql usando el script copia.sql usando el redireccionador <.

```
c:\xampp\mysql\bin>mysql -u root < copia.sql
c:\xampp\mysql\bin>
```

Para más información sobre mysqldump puede consultar:

<https://www.daniloaz.com/es/como-hacer-backups-de-mysql-mariadb-con-el-comando-mysqldump/>

<https://rm-rf.es/backups-mysql-con-mysqldump/>

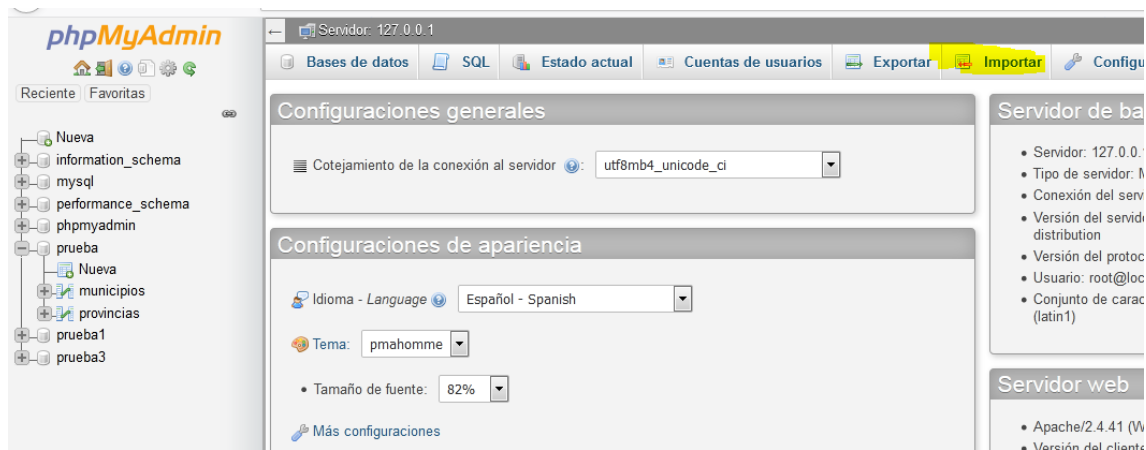
Ejemplo 13. Copias de BBDD con PHPMyAdmin

Usamos importar/exportar para hacer las copias de seguridad y las restauraciones.

Para hacer copias usamos exportar.



Para recuperar o crear bases de datos usamos importar (puede ser necesario **crear la BBDD antes si el script no tiene CREATE DATABASE** en el inicio de este).



Ejemplo 14. Carga de BD usando source en el cliente

Otra forma de cargar datos es usando source (por ejemplo, si el script no tiene el CREATE DATABASE).

```
c:\xampp\mysql\bin>mysql -u root
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MariaDB connection id is 328
Server version: 10.4.6-MariaDB mariadb.org binary distribution
Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
MariaDB [(none)]> CREATE DATABASE prueba2;
Query OK, 1 row affected (0.001 sec)
MariaDB [(none)]> USE prueba2;
Database changed
MariaDB [prueba2]> source c:\datos\copia.sql
```