

MariaDB Vs. MySQL

Pedro García Sánchez

Helena Hernández Payo

Department of Computer Science and Automation, University of Salamanca

Plaza de la Merced, s/n, 37008, Salamanca, Spain

pedro.gsanchez@usal.es

helenahdez@usal.es

Abstract. Se toma como punto de partida los sistemas gestores de bases de datos (SGBD) MariaDB y MySQL, se hace una comparativa de ambos y se desarrolla su aplicación en la administración de sistemas a través de la creación de un programa que permita a un usuario registrarse en un formulario web y almacenar dicha información en una base de datos, empleando para ello otras herramientas complementarias (PERL, CPAN, phpMyAdmin)

Keywords: MariaDB, MySQL, phpMyAdmin

1. Introducción

En la actualidad la competitividad y rapidez de maniobra para una empresa es imprescindible para su éxito. Cada vez existe una mayor demanda de datos, lo que conlleva una mayor necesidad de gestionarlos. Esta demanda siempre ha estado presente en las empresas, pero se ha disparado debido al incremento de las redes integradas en internet y a la aparición de nuevos dispositivos móviles.

Antes de que existieran aplicaciones informáticas las empresas usaban cajones, carpetas y fichas para poder gestionar todos sus datos, lo cual conllevaba una pérdida de tiempo muy grande. Gracias a la informática, todas las herramientas hacen que se parezcan a esos procesos manuales.

Un sistema de información se basa en bases de datos (BD) y sistemas de bases de datos (SGBD).

Una base de datos es una colección de datos relacionados entre sí con un significado implícito.

Un SGBD es una aplicación que permite a los usuarios definir, crear y mantener una base de datos, además de proporcionar un acceso controlado a la misma.

En este trabajo se hablará del SGBD MySQL en la sección 3 y, a continuación, se incluirá información sobre el también SGBD MariaDB, enfocándolo desde la perspectiva de una comparación con el primer gestor mencionado (todo ello contenido en la sección 4) Finalmente se terminará el informe con un breve ejemplo práctico que emplea varias herramientas para mostrar una posible aplicación de este tipo de sistemas (sección 5) y unas conclusiones (sección 6)

2. Background

Las bases de datos son parte esencial de cualquier sistema informático, puesto que todos los programas necesitan recurrir a diversos datos mientras se ejecutan o generan otros que se han de almacenar de forma fiable, sin contradicciones y a largo plazo. Esto es posible en **bases de datos (BD)** estructuradas y gestionadas por **sistemas de gestión de bases de datos (SGBD)**, aplicaciones de software que interactúan con el usuario o con otros programas para poner a su disposición un segmento de la información guardada en la base de datos.

A día de hoy, la gestión electrónica de datos está dominada por el modelo de base de datos relacional. Una base de datos relacional es una colección de elementos de datos organizados en un conjunto de tablas formalmente descritas desde la que se puede acceder a los datos o volver a montarlos de muchas maneras diferentes sin tener que reorganizar las tablas de la base. Existen numerosos sistemas gestores de bases de datos que permiten trabajar con BD relacionales, pero de entre todos ellos los más utilizados son Db2, Microsoft SQL Server, MySQL, PostgreSQL, Oracle Database y SQLite. En lo que respecta a los SGBD de código abierto, a nivel global sin duda MySQL es el que posee un mayor número de usuarios y por eso es interesante la “reciente” aparición de un fork de este gestor, MariaDB. Este SGBD, también de código abierto, está ganando popularidad muy rápidamente, principalmente por ofrecer una alta compatibilidad con MySQL con nuevas características y, en teoría, mejor rendimiento manteniendo la filosofía de software libre original de MySQL, sin licenciamientos duales. Sin embargo, ¿son realmente tan parecidos estos dos gestores? y, lo más importante, ¿cuál de los dos es mejor?

3. MySQL



3.1. Historia

MySQL es un sistema de bases de datos relacional que fue creado por la empresa sueca MySQL AB, fundada por David Axmark, Allan Larsson y Michael Widenius. El desarrollo original realizado por Axmark y Widenius comenzó en 1994, pero la primera versión de MySQL no apareció hasta el 23 de mayo de 1995. Dicho desarrollo comenzó porque los creadores consideraban mSQL, basado en el lenguaje de bajo nivel ISAM, como una alternativa demasiado lenta y poco flexible. Por ello, crearon una nueva interfaz para SQL manteniendo la API de mSQL, garantizando de esta manera que muchos desarrolladores pudieran utilizar MySQL en vez de éste, que tenía licencia propietaria.

A esta primera versión le siguieron sucesivas mejoras que constituyeron nuevos lanzamientos del sistema de bases de datos hasta que, en 2008, cuando el desarrollo iba por la versión 5.0, MySQL A.B fue adquirida por Sun Microsystems. Dos años más tarde, en 2010, esta empresa fue comprada a su vez por Oracle Corporation (que además ya era dueña desde 2005 de la empresa finlandesa Innobase Oy, desarrolladora del motor InnoDB que utiliza MySQL), actual propietaria del sistema gestor, que ya se encuentra en su versión 8.0.

3.2. Características

Como ya se ha mencionado, MySQL es un sistema de bases de datos relacional que, como la mayoría de gestores relacionales, utiliza el lenguaje SQL (Structured Query Language) creado por IBM en 1981 para la realización de consultas. Para su construcción, los lenguajes empleados fueron C y C++ y dispone de APIs que permiten a aplicaciones escritas en estos lenguajes (así como Eiffel, Java, Perl, PHP, Python, Ruby, y Tcl) acceder a las bases de datos MySQL. Asimismo, el sistema de gestión de bases de datos se destaca por su **amplio soporte de sistemas operativos** (Windows, GNU/Linux, MacOS...) además de por ser la opción

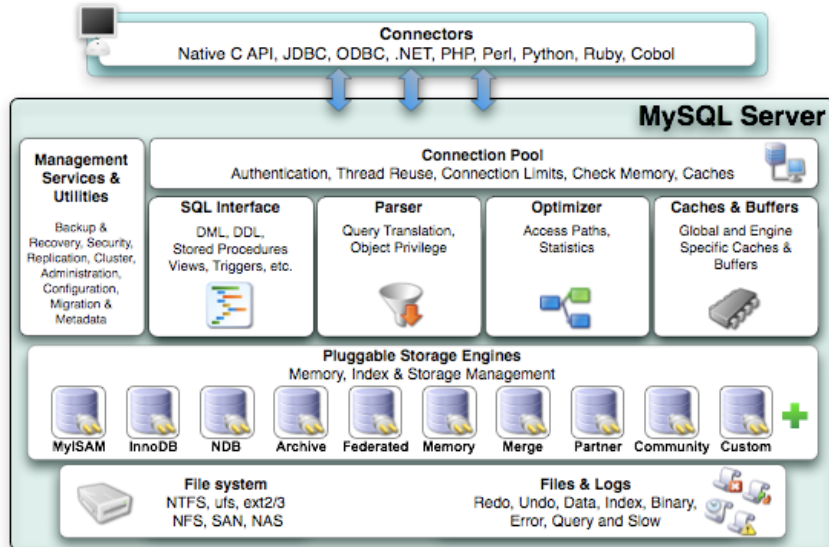
escogida por nombres importantes tales como Facebook, Github, YouTube, Twitter, PayPal, Nokia, Spotify o Netflix, entre otros.

En lo que respecta al tema de licencias, MySQL posee un **licenciamiento dual** ya que existe un tipo de licencia de código abierto (GPL) que es gratuita y ofrece la posibilidad de realizar modificaciones en los archivos fuente y redistribuirlas bajo esta misma licencia, pero también se ofrece una comercial para que las empresas puedan redistribuir el producto bajo sus propios términos, siendo en este caso necesario pagar. El primer tipo viene heredado de los orígenes del proyecto como software libre cuando fue creado por MySQL AB y el segundo apareció tras la compra por parte de Oracle, si bien ambos son gestionados por el mismo equipo. Esto, unido al hecho de que la versión de pago incluye cada vez más características comparada con la versión libre, no se ha visto con buenos ojos por parte de aquellos que buscan que no se pierda la esencia de código abierto del proyecto original.

En temas de estructura, MySQL soporta varios motores de almacenamiento que tratan con distintos tipos de tabla, transaccionales o no. A continuación se detallarán los correspondientes a la versión 8.0 del software, por ser la más actual, si bien versiones anteriores no disponen de algunos de ellos o la configuración por defecto ha variado:

- **InnoDB:** El motor de almacenamiento por defecto. Cumple con las llamadas propiedades ACID y permite realizar operaciones de commit y rollback, disponiendo a su vez de capacidades para recuperarse de fallos de distinta índole, consiguiendo con todo ello una gran protección en los datos de los usuarios. El bloqueo a nivel de fila de InnoDB junto con las lecturas no bloqueantes al estilo de Oracle que garantizan la consistencia, incrementan la concurrencia multi-usuario y el rendimiento. Este motor almacena los datos de los usuarios en índices clúster (tablas agrupadas) para reducir las operaciones de entrada y salida en las consultas más comunes, que se realizan empleando claves primarias. Además, para mantener la integridad de los datos, también se soportan constraints (restricciones) de tipo referencial que emplean claves foráneas.
- **MyISAM:** Estas tablas tienen un footprint pequeño (Espacio que ocupan en disco). Se emplea un bloqueo a nivel de tabla, lo que limita el rendimiento en tareas de lectura/escritura. Esto hace que a menudo se utilice este motor para operaciones de sólo lectura (read-only) o mayormente lectura (read-mostly) en configuraciones web y de almacenamiento de datos. Su característica principal es la gran velocidad que obtiene en las consultas, ya que no necesita hacer comprobaciones de la integridad referencial, ni bloquear las tablas para realizar las operaciones debido a la ausencia de características de atomicidad.

- **Memory**: Anteriormente conocido como **HEAP**, almacena todos los datos en la RAM, para un acceso rápido en entornos que requieren búsquedas veloces sobre datos que no sean de naturaleza crítica. Este motor se usa cada vez menos ya que **InnoDB** proporciona una forma duradera y de propósito general de almacenar la mayor parte o todos los datos en memoria (**buffer pool memory**) y **NDBCLUSTER** posibilita realizar búsquedas rápidas usando el valor de la clave en grandes conjuntos de información.
- **CSV**: Sus tablas son en realidad archivos de texto separados por comas. Este tipo de tablas permiten al usuario importar y volcar datos en formato CSV para poder intercambiarlos con scripts y aplicaciones que leen o escriben en este mismo formato. Debido a que no se encuentran indexadas, lo más habitual es mantener la información en tablas de InnoDB durante la mayor parte del tiempo y solo usar las CSV en la etapa de importación o exportación de datos.
- **Archive**: Se trata de tablas compactas y no indexadas destinadas a almacenar y recuperar grandes cantidades de información de distinto tipo referenciada con poca frecuencia, ya sea histórica, archivada o incluso sobre auditorías de seguridad.
- **Blackhole**: Este motor de almacenamiento acepta, pero no guarda, la información, de modo similar a como trabaja el dispositivo /dev/null en Unix. Las consultas (queries) siempre devuelven el conjunto vacío. Este tipo de tablas pueden ser usadas en configuraciones con replicación donde se mandan sentencias de tipo DML (aquellas que manipulan los datos, es decir, select, insert, update y delete) a servidores esclavos, pero el servidor maestro no conserva una copia de la información.
- **NDB** (también conocido como **NDBCLUSTER**): Este motor está especialmente pensado para aplicaciones que requieren el mayor grado posible de disponibilidad y tiempo de funcionamiento.
- **Merge**: Habilita a un administrador de bases de datos MySQL o a un desarrollador para agrupar una serie de tablas MyISAM idénticas y referenciarlas como un solo objeto. Es de utilidad para entornos **VLDB** (Very Large Databases).
- **Federated**: Ofrece la posibilidad de enlazar servidores MySQL separados para crear una única base de datos a nivel lógico a partir de varios servidores físicos. Muy útil para entornos distribuidos o data marts (almacenes de datos).
- **Example**: Este motor, como su nombre indica, sirve a modo de ejemplo para ilustrar cómo comenzar a escribir nuevos motores de almacenamiento. Primordialmente es de interés para los desarrolladores ya que se trata de un motor “stub” que no hace nada. Se pueden crear tablas bajo este motor, pero no se puede almacenar ni recuperar datos a partir de ellas.



Cabe destacar que, a partir de esta versión, MySQL deja de ofrecer compatibilidad descendente con versiones anteriores, además de incorporar un diccionario de datos transaccionales que almacena información sobre objetos de base de datos. En versiones anteriores de MySQL, los datos del diccionario se almacenaban en archivos de metadatos, tablas no transaccionales y diccionarios de datos específicos del motor de almacenamiento, con lo que este cambio marca una nueva tendencia en el desarrollo y explica las incompatibilidades.

4. MariaDB



MariaDB es un fork (bifurcación) del SGBD MySQL 5.1 con licencia GPL 2 (General Public License) que nace como una alternativa libre a MySQL cuyo propósito principal es proveer capacidades similares a las de MySQL.

Es desarrollado por Michael Widenius (fundador de MySQL AB), la Fundación MariaDB y la comunidad de desarrolladores de software libre.

A finales de febrero de 2008, Sun Microsystems adquirió MySQL AB, de manera que Widenius comenzó a desarrollar MariaDB porque no había certeza de que continuase siendo libre y porque estaba convencido de que el único interés de Oracle era reducir la competencia que MySQL suponía para el mayor proveedor de datos relacionales del mundo.

4.1 Comparativa con MySQL

MariaDB y MySQL están basados en el mismo núcleo de software y cumplen con el modelo de base de datos relacional, con el tiempo MariaDB se ha convertido en un SGDB autónomo y sus principales diferencias respecto a MySQL son las siguientes:

Motores de almacenamiento

Un motor de almacenamiento es un subsistema de almacenamiento que permite al SGDB la creación, lectura, actualización y eliminación de datos en tablas de bases de datos.

MariaDB quiere diferenciarse en el futuro de MySQL a través de la flexibilidad. Es por ello que además de los motores estándar compatibles con MySQL, existe cada vez un número mayor de motores de base de datos alternativos para poder elegir función del objetivo que exista a la hora de crear la base de datos.

En la siguiente tabla se muestra una comparativa de los motores de almacenamiento.

	MySQL 8.0.11	MariaDB 10.3.8
InnoDB/XtraDB	SI	SI
MyISAM	SI	SI
MEMORY	SI	SI
CSV	SI	SI
ARCHIVE	SI	SI
BLACKHOLE	SI	SI
MERGE	SI	SI
COLUMNSTORE	NO	SI
ARIA (Sustitute a MyISAM)	NO	SI
CASSADRA	NO	SI
CONNECT	NO	SI

MROONGA	NO	SI
QQGRAPH	NO	SI
SEQUENCE	NO	SI
SPHINXSE	NO	SI
SPIDER	NO	SI
TOKUDB	NO	SI

Así mismo, en función del objetivo se puede diferenciar:

- **Propósito general:** **XtraDB o InnoDB** son un buen motor de almacenamiento de transacciones en general. **Aria**, la mejora más moderna de MariaDB en MyISAM, tiene una huella pequeña y permite una copia fácil entre los sistemas.
- **Escalado, particionamiento:** cuando se quiere dividir la base de datos en varios servidores u optimizar para escalar. **TokuDB o Spider**.
- **Compresión/Archivo:** para permitir un ahorro del espacio. **TokuDB o archive**.
- **Conexión a otras fuentes de datos:** Cuando quiera usar datos no almacenados en una base de datos MariaDB. **CONNECT o CSV** (preferible CONNECT ya que permite más tipos de datos que CSV que solo permite datos csv), **FederatedX** (para acceder de manera remota a tablas ubicadas en otro servidor) o **CassandraSE**.
- **Búsqueda optimizada:** Buscadores optimizados para búsqueda. **SphinxSE**.
- **Otros motores de almacenamiento especializados:** **SEQUENCE** (creación de secuencias ascendentes o descendientes de número), **BLACKHOLE** (útil para entornos de replicación) o **OQGRAPH** (permite manejar jerarquías).

Mejoras de velocidad

MariaDB incluye mejoras de velocidad en diversos aspectos frente a MySQL como puedan ser:

- Mejoras en el código DBUG para hacer la ejecución más rápida cuando se compila pero no se usa

- Tabla de chequeo de redundancia más rápida. Permite conocer si dos tablas son iguales en menor tiempo.
- El uso del motor **Aria** permite realizar consultas complejas de una manera rápida ya que se usa para tablas temporales internas, las cuales dan mayor velocidad en secciones complejas. **Aria** es más rápida que las tablas temporales cuando se compara con **MyISAM** ya que Aria cachea las filas de datos en memoria y normalmente no escribe filas temporales en el disco.
- Replicación rápida y segura.
- La suite de pruebas está extendida lo que hace que ahora corra más rápido que antes.
- El Sistema para manejar las conexiones se ha mejorado ya que permite tener más de 200.000 conexiones a MariaDB.

Extensiones y nuevas características

MariaDB incorpora nuevas características como:

- Puede manejar 32 segmentos clave por clave sobre los 16 originales.
- Precisión de microsegundos en la lista de procesos
- Columnas virtuales
- Cache de claves segmentadas
- Extensiones de prueba mysqldtest
- Estadísticas extendidas para el usuario

Versiones

Los números de las versiones de MariaDB siguen los de MySQL hasta la versión 5.5. Esto quiere decir que MariaDB ofrece todas las características de MySQL 5.5, pero tras esto los desarrolladores decidieron iniciar una nueva rama iniciando desde el número 10 con el objetivo de dejar claro que MariaDB no es una copia de MySQL y que añade más características.

Consultas de base de datos

Ambos SGBD buscan una compatibilidad 100% con el lenguaje de base de datos SQL, de manera que las instrucciones SQL son las mismas.

Seguridad

Tanto MySQL como MariaDB ofrecen funciones de codificación para datos inactivos.

MySQL codifica los datos almacenados en la base de datos a nivel de espacio de tabla, pero no permite codificar tablas por separado. En cambio, MariaDB ofrece funciones de codificación de datos en reposo diferenciadas en los siguientes niveles de base de datos:

- Espacios de tabla InnoDB
- Tablas InnoDB
- Tablas Aria
- Archivos temporales
- Archivos de registro binarios

4.2 Compatibilidad con MySQL

Como MariaDB es un descendiente binario en reemplazo de la misma versión de MySQL su objetivo es garantizar la compatibilidad total para facilitar su reemplazo inmediato. Esto quiere decir:

- Todos los nombres de archivos, binaries, rutas, etc... deben ser los mismos.
- Todas las APIs de clientes, protocolos y estructuras son idénticas.
- Todos los conectores MySQL trabajan sin cambios con MariaDB

En muchos casos para cambiar de MySQL a MariaDB serviría simplemente con desinstalar MySQL e instalar MariaDB.

Para reducir los problemas de compatibilidad al mínimo, el equipo de MariaDB sincroniza mensualmente el código de la bifurcación con el Código de MySQL. Sin embargo, esta compatibilidad termina con MySQL 8 ya que deja de ofrecer compatibilidad descendiente con ninguna versión anterior de MySQL ni MariaDB.

4.3 Instalar MariaDB en Debian 9

Debian 9 tiene la característica de que por defecto instala MariaDB aunque se instale MySQL, por lo que aunque aquí se detallan específicamente los pasos para instalar el primero, el resultado final sería el mismo si en vez de mariadb se usara la palabra mysql.

Para poder llevar a cabo nuestra instalación en Debian 9 haremos uso de los repositorios Linux (que actualmente contienen la versión 10.1 del sistema gestor), si bien también se pueden utilizar los repositorios propios de MariaDB para obtener versiones más recientes pero es menos recomendable. En este caso habría que añadir el correspondiente repositorio y clave de éste a nuestro Debian.

```

File Edit View Search Window Help
helen@debian:~$ sudo mysql_secure_installation

NOTE: RUNNING ALL PARTS OF THIS SCRIPT IS RECOMMENDED FOR ALL MariaDB
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.

Set root password? [Y/n] Y
New password:
Re-enter new password:
Password updated successfully!
Reloading privilege tables...
... Success!

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!

```

Mediante el comando **sudo apt install mariadb-server && mariadb-client** se instalará mariadb en la máquina Debian. A continuación, se solicitará una contraseña que es a través de la cual se accede a la base de datos.

Package configuration

Configuring mariadb-server-10.1

While not mandatory, it is highly recommended that you set a password for the MariaDB administrative "root" user.

If this field is left blank, the password will not be changed.

New password for the MariaDB "root" user:

<Ok>

En caso de que durante este proceso no se nos solicite ninguna contraseña para el root (lo que querrá decir que tendremos la que viene por defecto, ninguna) podemos ejecutar el comando **sudo mysql_secure_installation**, que nos preguntará también si deseamos eliminar usuarios anónimos, restringir el acceso del usuario *root* a la máquina local y eliminar las bases de datos de prueba. Es recomendable responder afirmativamente a todas las preguntas para mejorar la seguridad de la instalación.

Tras esto, para poder entrar a la base de datos se pondrá el siguiente comando en la consola: **sudo mariadb -u root -p** y se introducirá la contraseña creada para el usuario root en los pasos anteriores.

```

helena@debian:~$ sudo mariadb -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 18
Server version: 10.1.37-MariaDB-0+deb9u1 Debian 9.6

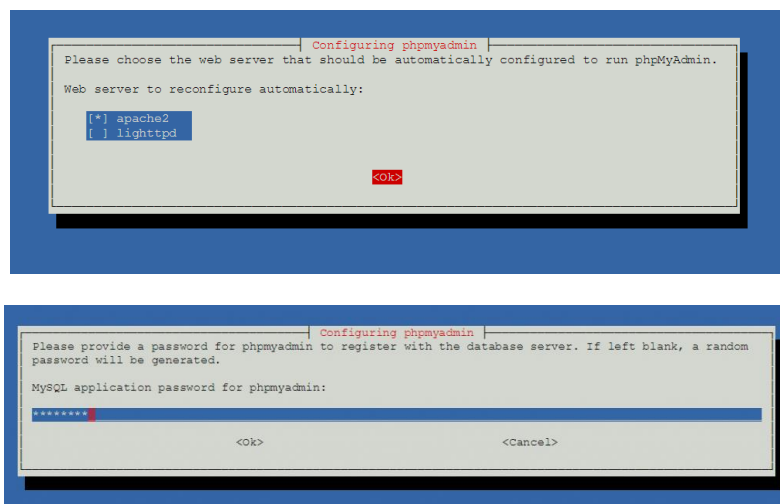
Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

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

MariaDB [(none)]> █

```

Una aplicación muy utilizada en conjunto con estos gestores de bases de datos es phpMyAdmin, que nos permitirá manejar nuestras tablas de un modo más sencillo al proveernos de un entorno gráfico. Para poder instalarla utilizaremos el siguiente comando: **sudo apt install phpmyadmin**. Esta herramienta requiere a su vez que poseamos Apache, de manera que mediante **sudo apt install apache** instalaremos Apache 2 en nuestro ordenador. Nos solicitará una contraseña que es a través de la cual accederemos a las funciones de phpMyAdmin.



Seguidos todos estos pasos ya podemos trabajar con MariaDB, ya sea a través de la terminal utilizando los comandos propios de SQL o empleando el modo gráfico proporcionado por phpMyAdmin. A continuación, se detalla un pequeño ejemplo práctico de como se pueden aplicar estas herramientas que se han instalado para la administración de sistemas.

5. Ejemplo práctico

Partiendo de que se dispone de todas las herramientas detalladas en el punto anterior, seguidamente se procederá a explicar como utilizarlas en conjunto con el

```

File Edit View Search Terminal Help
helena@debian:~$ sudo mariadb -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 72
Server version: 10.1.37-MariaDB-0+deb9u1 Debian 9.6

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 user useradmin@localhost identified by 'rootadmin';
Query OK, 0 rows affected (0.00 sec)

MariaDB [(none)]> create database mariadbtest;
Query OK, 1 row affected (0.00 sec)

MariaDB [(none)]> grant all privileges on mariadbtest.* to 'useradmin'@'localhost';
Query OK, 0 rows affected (0.00 sec)

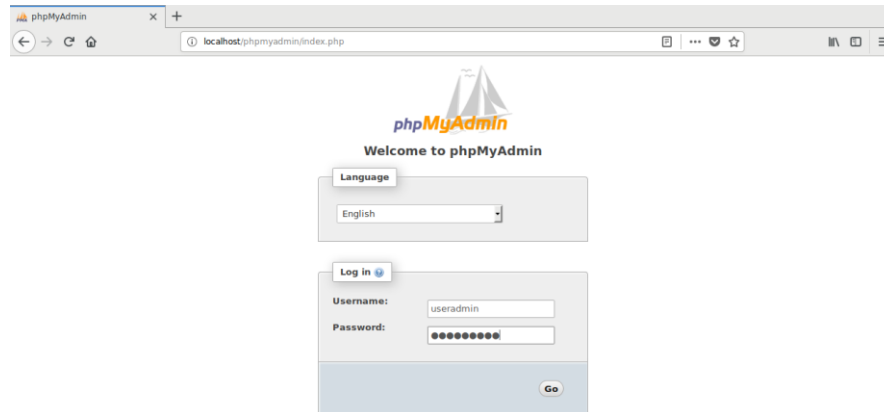
MariaDB [(none)]> █

```

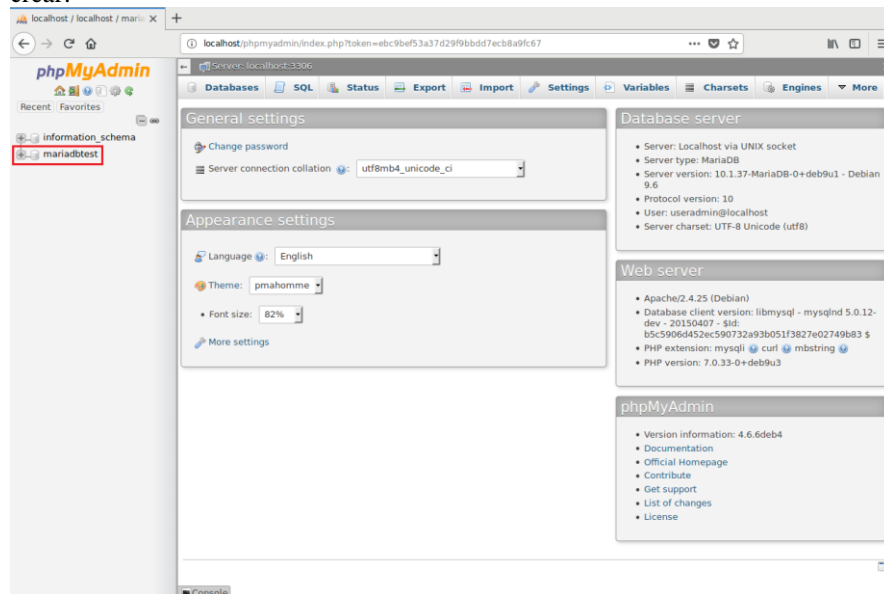
lenguaje de programación Perl para crear un formulario de registro que, una vez completado correctamente, nos añada un nuevo usuario a nuestra base de datos que, a partir de ese momento, pueda acceder al sistema usando las credenciales proporcionadas en dicho registro.

En primer lugar, tenemos que crear una base de datos en la que trabajar, así como una tabla donde almacenar los datos de nuestros usuarios. Para ello, debido a que phpmyadmin no nos deja trabajar con nuestra cuenta root, tenemos que utilizar el usuario “phpmyadmin” (cuya contraseña establecimos durante la instalación) o crear uno nuevo. En cualquiera de los casos, al no ser root, no dispondremos de privilegios, por lo que tenemos que crear una base de datos desde la terminal y concederle todos ellos al usuario que hayamos escogido:

A continuación, abriremos phpmyadmin accediendo a la dirección localhost/phpmyadmin desde nuestro navegador y haremos login con el usuario escogido:



Podemos ver en la parte superior izquierda la base de datos que acabamos de crear:



Si hacemos click en ella, llegaremos a otra pantalla donde se nos indica que esta base de datos no dispone de ninguna tabla, con lo que crearemos una nueva. En nuestro caso la hemos llamado “usuarios” y constará tan solo de 3 columnas (para el nombre de usuario, la contraseña y un correo electrónico):

localhost / localhost / mariadb X

localhost/phpmyadmin/db_structure.php?server=1&db=mariadbtest&token=ebc9bef53a37d29f9bdd7ecb...

Structure SQL Search Query Export Import Operations Routines Events More

No tables found in database.

Create table

Name: usuarios Number of columns: 3

Go

Console

localhost / localhost / mariadb X

localhost/phpmyadmin/db_structure.php?server=1&db=mariadbtest&token=ebc9bef53a37d29f9bdd7ecb...

Table: usuarios

Table name: usuarios Add 1 column(s) Go

Name	Type	Length/Values	Default	Collation	Attributes	Null	Index
username	VARCHAR	10	None				PRIMARY
password	VARCHAR	30	None				---
mail	VARCHAR	100	None				UNIQUE

Table comments: Collation: Storage Engine: InnoDB

PARTITION definition:

Partition by: (Expression or column list)

Partitions: 1

Preview SQL Save

Console

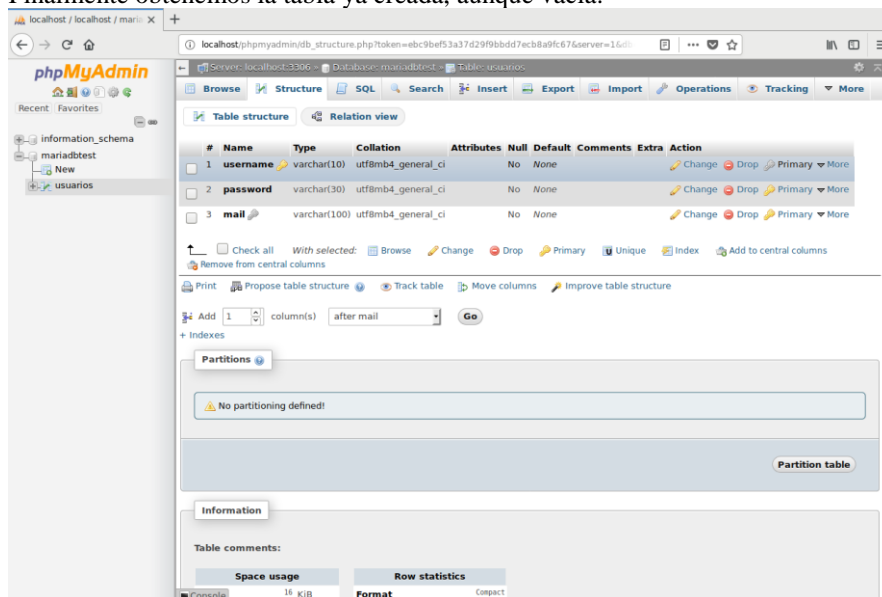
A continuación se detalla el significado de cada una de los campos solicitados:

Opción	Descripción
Name	A cada columna en una tabla de una base de datos se le asigna

	<p>un nombre, el cual puede escogerse libremente, aunque con algunas restricciones. A este respecto, los caracteres del alfabeto latino (mayúsculas o minúsculas, pero sin acentos), los números, el símbolo del dólar y el guion bajo no representan ningún problema. Este se puede usar como alternativa al espacio, que no está permitido (incorrecto: user id; correcto: user_id). Los nombres de las columnas tienen que estar formados por otros símbolos además de números. Asimismo, en el lenguaje para bases de datos SQL se pueden encontrar algunas palabras clave que están reservadas para determinadas tareas. Se puede acceder a una lista de las mismas en la documentación de MySQL. La mayor parte de estas limitaciones pueden eludirse, pero la columna correspondiente siempre debe ir entre comillas simples ('...'). Estas mismas reglas se aplican, entre otros, a los nombres de las tablas en MySQL. Se recomienda optar por nombres de columna elocuentes y que se adapten a los atributos correspondientes.</p>
Type	<p>El tipo de datos pone de relieve cuál es la naturaleza de la información que se guarda en una columna. MySQL y MariaDB te permiten definir datos en forma de números enteros o de números de coma flotante, de hora o fecha, así como de cadenas de texto y datos binarios. Se puede encontrar una descripción de los mismos en la tabla de tipos de datos.</p>
Length/Values	<p>En algunos tipos de datos (por ejemplo, las cadenas de texto) se puede asignar una longitud máxima a los valores de una columna, aunque este ajuste es opcional.</p>
Default	<p>La opción "Default" (por defecto) permite definir un valor estándar para una columna, el cual se inserta automáticamente cuando un conjunto de datos no contiene ningún valor para la columna correspondiente.</p>
Collation	<p>La opción "Collation" define un determinado tipo de caracteres para una columna, el cual puede diferir de los ajustes globales de la base de datos. Se puede modificar, además, la codificación en los diferentes niveles de la tabla para todas las columnas.</p>
Attributes	<p>Algunos tipos de datos se pueden fijar de una manera más precisa a través de atributos opcionales. Así, con los atributos signed y unsigned se puede establecer, por ejemplo, si los números enteros o los de coma flotante asumen valores positivos (unsigned) o también negativos (signed).</p>
Index	<p>A través de la opción "Index" se pueden definir las reglas para la indexación. Si se selecciona el ajuste Index PRIMARY para las columnas, este funciona como clave primaria de la tabla. El ajuste UNIQUE indica que los valores en esta columna solo pueden guardarse una vez, evitando las duplicaciones.</p>
A_I	<p>La abreviatura "A I" hace referencia a AUTO INCREMENT e</p>

	indica al gestor de bases de datos que incremente un valor automáticamente cuando no se indique ninguno a la hora de crear una secuencia de datos. Esta opción es de aplicación cuando se indexan conjuntos de datos.
Comments	El campo “Comments” integra comentarios en las columnas de la tabla.

Hemos establecido como clave primaria el nombre de usuario y se ha marcado el correo electrónico como UNIQUE para evitar duplicaciones en el registro. Finalmente obtenemos la tabla ya creada, aunque vacía:



En esta pantalla podemos hacer diversas modificaciones o consultas en la tabla sin tener que recurrir a comandos SQL (como insertar filas, borrarlas, modificar los atributos...). Sin embargo, como en este ejemplo deseamos que la inserción de los usuarios se haga de manera automática con el registro, no se profundizará en este tema.

Como ya tenemos la tabla donde se va a almacenar la información sobre los usuarios, ahora tan solo nos queda crear un formulario web (usando HTML) donde se llevará a cabo el registro del usuario y un script de Perl que, una vez pulsado en el botón de registro, se ejecute y genere una nueva fila en la tabla con los datos introducidos.

Por no ser el objetivo de la práctica, se considerará que el lector posee conocimientos suficientes de HTML para realizar el formulario detallado anteriormente y tan solo se explicará como programar el script de Perl. Para ello haremos uso del módulo DBI, que nos permite acceder a bases de datos

relacionales, y CGI, para capturar los parámetros introducidos en el registro, los cuales descargaremos mediante CPAN (**install DBD::mysql** e **install CGI**):

```
cpan[1]> install CGI
```

```
cpan[2]> install DBD::MariaDB
Running install for module 'DBD::MariaDB'
Fetching with LWP:
http://www.cpan.org/authors/id/P/PA/PALI/DBD-MariaDB-1.21.tar.gz
Fetching with LWP:
http://www.cpan.org/authors/id/P/PA/PALI/CHECKSUMS
cpan[2]> install DBD::mysql
Running install for module 'DBD::mysql'
Fetching with LWP:
http://www.cpan.org/authors/id/D/DV/DVEEDEN/DBD-mysql-4.050.tar.gz
Fetching with LWP:
http://www.cpan.org/authors/id/D/DV/DVEEDEN/CHECKSUMS
```

El script quedaría de la siguiente forma:

```
#!/usr/bin/perl

use strict;
use warnings;
use CGI qw();
use DBI;

print "Content-type: text/html\n\n";

# Obtencion de los parametros introducidos en el formulario

my $web=CGI->new;
my $LOGIN=$web->param('username');
my $PASSWORD=$web->param('password');
my $EMAIL=$web->param('email');

my $ERROR=0;

#Conexion a la base de datos

my $dbh = DBI -> connect("DBI:mysql:database=mariadbtest; host=localhost; port=3306", "useradmin","rootadmin",
{ 'RaiseError' =>1});

#Comprobacion de existencia de usuario en la base de datos

my $sth = $dbh->prepare("SELECT username, mail from usuarios\n");

$sth -> execute();

while(my $ref = $sth->fetchrow_hashref()){
    if($LOGIN eq $ref->{username}){
        print "<p>ERROR: Nombre de usuario ya existente</p>";
        $ERROR=1;
    }
    if($EMAIL eq $ref->{mail}){
        print "<p>ERROR: Correo ya existente</p>";
        $ERROR=1;
    }
    if($ERROR){
        last;
    }
}

$sth->finish;

#Insertar usuario en la base de datos

$dbh->do("INSERT INTO usuarios (username, password, mail) VALUES ('".$LOGIN."', ' ".$PASSWORD."', ' ".$EMAIL."')");

print "Registro completado con exito\n";
```

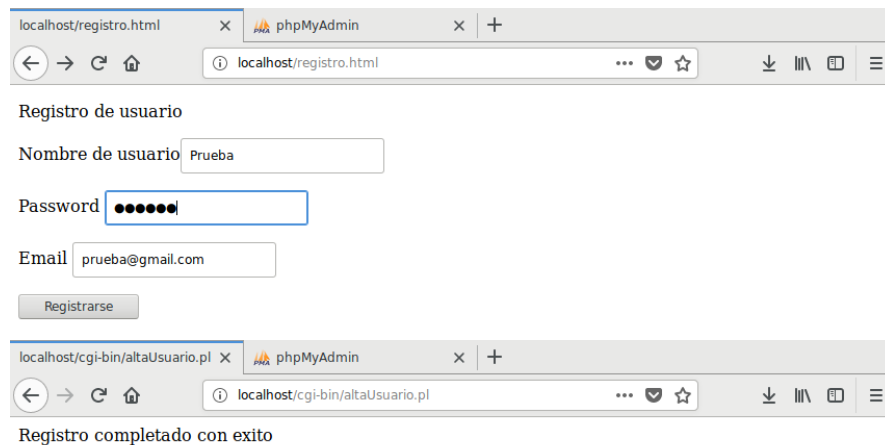
Como se puede apreciar en el código de arriba, lo primero que se hace es obtener los valores de los parámetros de nombre de usuario, contraseña y correo electrónico. Una vez hecho esto, hay que conectarse con la base de datos para poder hacer las operaciones de inserción en nuestra tabla. Esto se lleva a cabo mediante la función “connect”, en la que se deben indicar una serie de datos de acceso siguiendo el siguiente formato:

DBI:mysql:database= “nombre de la base de datos”;host=localname;port=3306, “nombre de usuario de MariaDB”, “contraseña”

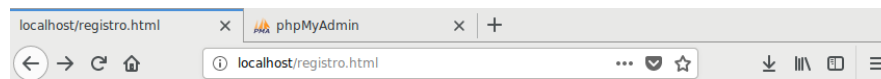
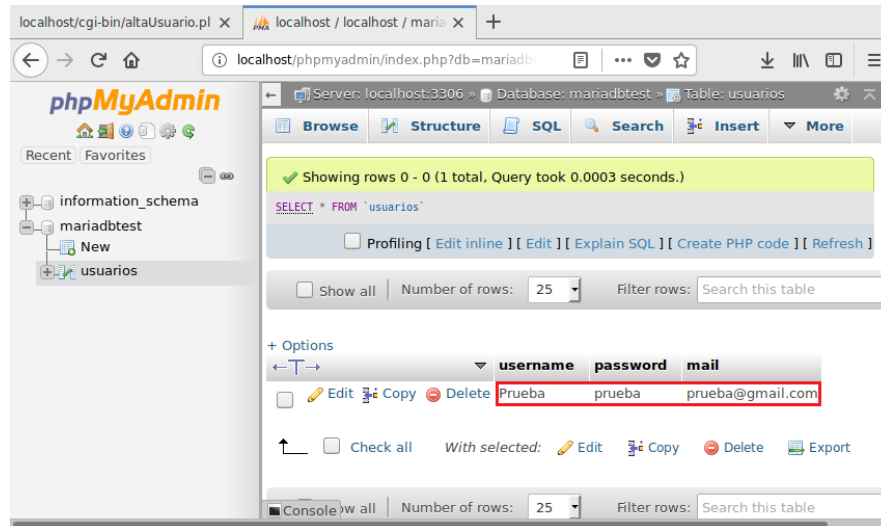
Una vez conectados a la base de datos y previamente a la inserción, preparamos una sentencia SELECT filtrando por el nombre de usuario y el correo y la ejecutamos (comandos “prepare” y “execute”), para poder hacer un recorrido por todos los usuarios en la base de datos y comprobar que ni el username ni el mail están repetidos.

Para terminar, se realiza la sentencia INSERT para añadir una nueva fila a la tabla y el usuario queda registrado en el sistema. En el caso de que los valores de nombre de usuario o correo electrónico estuviesen repetidos, al haberlos declarado como PRIMARY y UNIQUE respectivamente, no se añadirá el usuario a la tabla.

A continuación se muestra el correcto funcionamiento del script ya sea para añadir un usuario a la BD como para detectar que el nombre de usuario o el email están repetidos y no hacerlo:

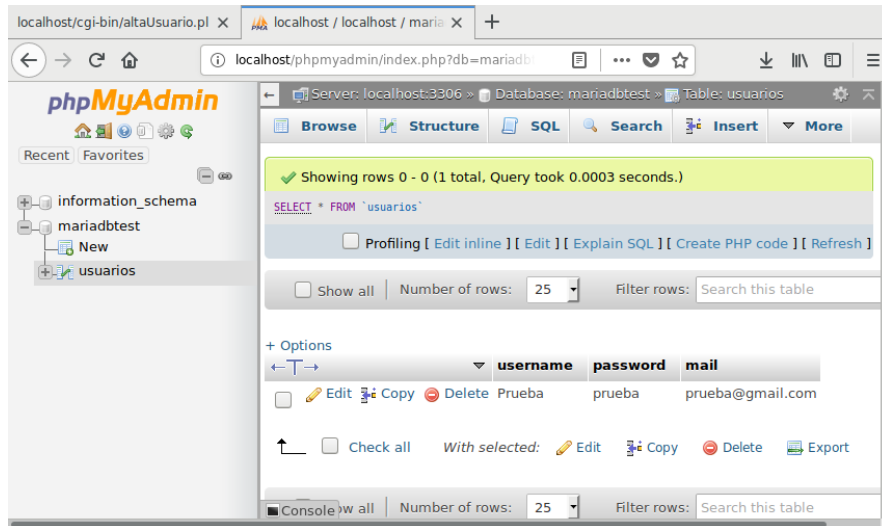


The image contains two screenshots of a web browser window. The top screenshot shows a page titled 'Registro de usuario' with a form containing three input fields: 'Nombre de usuario' (with the value 'Prueba'), 'Password' (with masked characters), and 'Email' (with the value 'prueba@gmail.com'). A 'Registrarse' button is located below the form. The bottom screenshot shows the same browser window after the registration process, displaying the message 'Registro completado con éxito'.



ERROR: Nombre de usuario ya existente

ERROR: Correo ya existente



A partir de aquí es sencillo seguir el mismo proceso para crear un formulario de acceso al sistema (login) que, por medio de una sentencia SELECT recorra la tabla de usuarios y, si encuentra el nombre de usuario, compruebe que la contraseña introducida y la almacenada en la base de datos coinciden:

```
#!/usr/bin/perl

use strict;
use warnings;
use CGI qw();
use DBI;

print "Content-type: text/html\n\n";

# Obtencion de los parametros introducidos en el formulario

my $web=CGI->new;
my $LOGIN=$web->param('username');
my $PASSWORD=$web->param('password');

my $ENCONTRADO=0;

#Conexion a la base de datos

my $dbh = DBI -> connect("DBI:mysql:database=mariadbtest; host=localhost; port=3306", "useradmin","rootadmin",
{'RaiseError'=>1});

#Comprobacion de existencia de usuario en la base de datos y de que la contraseña es correcta

my $sth = $dbh->prepare("SELECT username, password from usuarios\n");

$sth -> execute();

while(my $ref = $sth->fetchrow_hashref()){
    if($LOGIN eq $ref->{username}){
        if($PASSWORD eq $ref->{password}){
            print "<p>Acceso concedido</p>";
        }
        else{
            print "<p>ERROR: La contraseña no coincide</p>";
        }
        $ENCONTRADO=1;
        last;
    }
}

$sth->finish;

if(!$ENCONTRADO){
    print "<p>ERROR: No existe ningún usuario con ese username</p>";
}
```

6. Conclusiones

Hay evidencias de que MariaDB se ha convertido en una alternativa seria a MySQL ya que lo ha sustituido como instalación estándar en diversas distribuciones de Linux (Fedora, Debian, Arch...) y otras aplicaciones conocidas como Google, Mozilla, HP o Wikipedia han apostado por MariaDB.

Aunque actualmente MariaDB y MySQL estén siguiendo caminos parecidos, en un futuro el Desarrollo de ambos proyectos se va a distanciar más ya que están en búsqueda del Desarrollo de características y extensiones exclusivas, como ya se empieza a apreciar con las incompatibilidades surgidas a partir de la versión 8 de MySQL.

7. Referencias

1. <https://www.ionos.es/digitalguide/hosting/cuestiones-tecnicas/mariadb-vs-mysql/>
2. <https://mariadb.com/kb/es/mariadb-spanish/>
3. <https://mariadb.com/kb/en/library/mariadb-vs-mysql-features/>
4. <https://mariadb.com/kb/en/library/mariadb-vs-mysql-compatibility/>
5. <https://mariadb.com/kb/es/basic-sql-statements/>
6. <https://www.overant.com/blog/diferencias-entre-mysql-y-mariadb/>
7. <https://gestionbasesdatos.readthedocs.io/es/latest/Tema1/Teoria.html>
8. <https://dockertips.com/mariadb-engines>
9. <https://es.wikipedia.org/wiki/MariaDB>
10. <https://www.ionos.es/digitalguide/hosting/cuestiones-tecnicas/bases-de-datos-relacionales/>
11. <https://en.wikipedia.org/wiki/MySQL>
12. <https://dev.mysql.com/doc/refman/8.0/en/storage-engines.html>