Unidad 4

El lenguaje SQL. DDL

Gestión de bases de datos

Índice

4.1. SQL. Tipos de lenguajes

4.2. Tipos de datos en SQL

4.3. Instalación de un SGBD
- 4.3.1. MySQL
- 4.3.2. MariaDB

4.4. Lenguaje de definición de datos (DDL)
- 4.4.1. Definición de bases de datos
- 4.4.2. Definición de tablas
- 4.4.3. Definición de índices
- 4.4.4. Definición de tipos de datos

4.5. Resumen de comandos

4.6. Caso práctico

Introducción

Tiempo después de que E.F Codd creara el modelo relacional, dos trabajadores de IBM Research CeNter definieron un cierto lenguaje de programación que nos permitiera interactuar con los datos almacenados en una base de datos. Como el lenguaje era muy similar a la lengua hablada y cotidiana, en inglés, el primer nombre que recibió este lenguaje fue SEQUEL (structured English query language), que significa lenguaje de consulta estructurado en inglés. Este nombre tenía ciertos choques con otro en lo referente a Copyright y decidieron llamarlo por su nombre actual, SQL.

SQL es un lenguaje de alto nivel que coge sus bases de la teoría del álgebra y cálculo relacional y tiene la intención de comunicarse con un sistema gestor de bases de datos.

Este lenguaje nos ayudará a todo tipo de interacción con la base de datos, desde realizar consultas hasta definir la estructura de la base de datos.

Por último, hay que tener en cuenta que el lenguaje SQL es un lenguaje de programación no procedimental que solo sirve para ejecutar acciones en bases de datos y carece de elementos de otros lenguajes de programación.

Al finalizar esta unidad

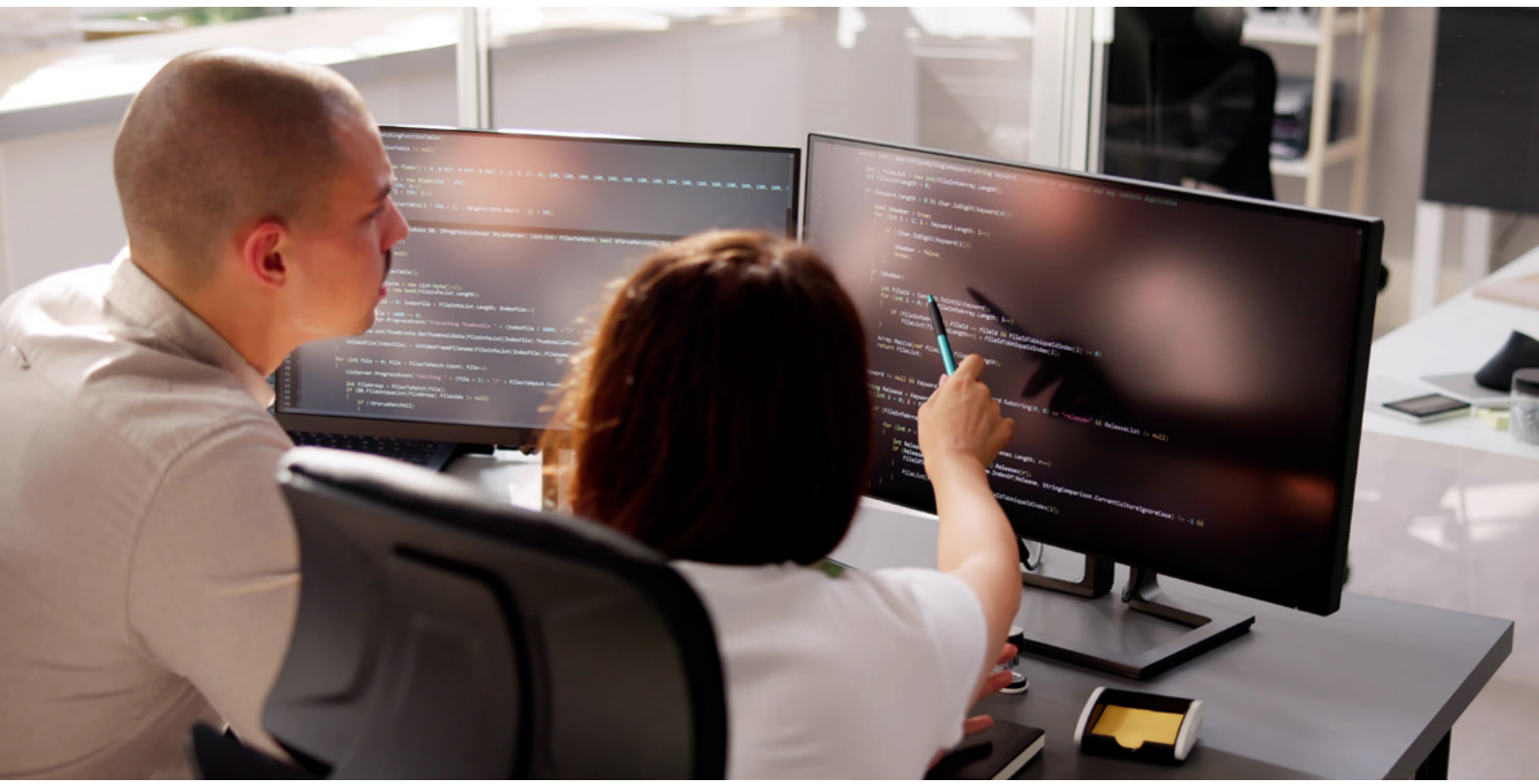
- Conoceremos los condicionantes en que surgió el lenguaje SQL,
- Habremos comprendido el ámbito y estructura del lenguaje SQL.
- Estaremos familiarizados con los tipos de datos del SQL estándar.
- Seremos capaces de crear, modificar y eliminar objetos de la base de datos mediante el DDL.

4.1. SQL. Tipos de lenguajes

Las distintas funciones que se pueden realizar con SQL como lenguaje se pueden dividir en tres sublenguajes:

- **DDL (data definition language).** Lenguaje de definición de datos, nos facilita la creación de la estructura de la base de datos, tablas y la propia base de datos.
- **DML (data manipulation language).** Lenguaje de manipulación de datos, es el lenguaje que nos permite trabajar sobre los datos que contiene la base de datos como la modificación o consulta de los datos.
- **DCL (data control language).** Lenguaje de control de datos, es el lenguaje que usaremos para administrar la seguridad de los datos mediante permisos y usuarios.

Como en el lenguaje SQL no hay un a universalidad debido a que se ha ido implementando de distintos modos a lo largo de los años, nosotros vamos a seguir la sintaxis SQL estándar, que es la recogida en la versión SQL:1999.



4.2. Tipos de datos en SQL

El estándar de SQL nos define los siguientes tipos de datos, que agruparemos por familias:

Cadenas de caracteres

- **CHAR (CHARACTER):** se trata de una cadena de caracteres que tienen una longitud fija y que habrá que especificar. Si no se especifica nada, será de un solo carácter.

  Si tenemos que la longitud del dato que almacenamos es menor de la definida, esto se rellenará con espacio. Por lo que podemos decir que realmente el número es un máximo de caracteres a escribir por el usuario. Un ejemplo sería almacenar un DNI en un campo 'cDNI' que se definiría como *CHAR(9)*.

- **VARCHAR (CHARACTER VARYING):** es una cadena de caracteres en la que su longitud puede variar, pero habrá que definir una longitud máxima.

  La diferencia con la anterior es que, aunque se almacene un dato de una longitud menor a la definida, solo se almacena lo que introducimos y no se rellena de ningún modo. Un ejemplo sería el campo 'cNombre' de tipo *VARCHAR(60)*, donde Juan sería un nombre admitido sin problema.

- **CLOB (CHARACTER LARGE OBJECT):** se trata de una cadena de caracteres que tienen un gran tamaño y por eso se almacenará en un archivo a parte al de la tabla. Algunas veces hay ciertos SGBD que limitan el trabajo con este tipo de datos.

Números exactos

Su uso debe de ser para almacenar código o valores numéricos necesarios para realizar operaciones. El ejemplo más común es el de un número de teléfono, que, aunque sea exacto, debe de almacenarse como una cadena de caracteres porque no se va a operar con él y no sirve de identificador. Los tipos son los siguientes:

- **INT (INTEGER):** definimos en este tipo los números enteros que se pueden almacenar en 4 bytes. El rango comprendido va desde el – 2 147 483 648 al 2 147 483 647.

- **SMALLINT:** comprende los números enteros almacenados en 2 bytes, su rango ahora va desde el – 32 768 al 32 767.

- **NUMERIC o DEC (DECIMAL):** es un número que tienen una parte decimal de tamaño fijo y para eso deberemos de especificar el número de dígitos totales y el de dígitos decimales que se separan por una coma. Un ejemplo sería el campo 'nNotaSelectividad' que se podría definir como NUMERIC(4,2), dos dígitos enteros y dos decimales.

Números aproximados

Nos dan la opción de que el rango donde movernos en valores decimales ser mayor a los datos NUMERIC, pero también menos exactos. Sus tipos son:

- **FLOAT o REAL:** definimos aquí un número real en coma flotante.
- **DOUBLE:** se define un número real en coma flotante, pero con doble precisión.

Fechas, horas e intervalos

- **DATE:** almacenamos la fecha en formato año-mes-día.
- **TIME:** almacenamos la hora en formato hora-minuto-segundo.
- **TIMESTAMP:** se almacena la fecha y la hora en la que se modifique el registro afectado y su formato es año-mes-día-hora-minuto-segundo.
- **INTERVAL:** representa un intervalo a lo largo del tiempo y tenemos dos formatos, años-meses o días-horas-minutos-segundos.

Los SGBD almacenan normalmente los datos de una fecha en concreto como todos los segundos que han transcurrido desde el día 1 de enero de 1970.

Este formato se llama Unix time y es común para otros tipos de software.

Valores lógicos

- **BOOLEAN:** se incluyen aquí os valores "true" y "false" para dictaminar si algo es verdadero o falso y dependiendo de la implementación también se podría almacenar el valor NULL (nulo).

Objetos binarios

- **BLOB (BINARY LARGE OBJECT):** almacenas imágenes, documentos, u otros objetos de carácter binario que normalmente se almacenan en un archivo fuera de la tabla en la que estén referenciados.
- Tipos definidos por el usuario. Los veremos más adelante con detalle.

4.3. Instalación de un SGBD

Para poder trabajar con el lenguaje de bases de datos SQL debemos de tener un Sistema Gestor de Bases de datos instalado.

Existen numerosos sistemas gestores de bases de datos, pero nos vamos a centrar en, *MySQL Server* o *MariaDB* se trata de un *software* libre, gratuito y muy usado en el día a día.

MySQL surgió en 1995 por un programador Michael Widenius para dar solución a un sistema de almacenamiento a sus archivos. En 2008, MySQL fue adquirido por Sun Microsystems, quien en 2010 fue comprado por Oracle Coporation, convirtiéndola en una base de datos comercial.

A raíz de este último hecho, Michael Widenius decidió fundar Maria DB Foundation y crear el SGBD **MariaDB,** un clon de MySQL para garantizar que sea una base de datos open source y gratuita para evitar que la compañía Oracle comercializase completamente MySQL. Por eso, se puede utilizar MySQL o MariaDB ya que son prácticamente el mismo sistema. Sus página oficiales son: https://mysql.com/ y https://mariadb.com/ y disponen de información de los servicios que ofrecen, descargas y manuales de apoyo, por ejemplo para su instalación.

Imagen 1. Logotipos de MySQL y MariaDB

4.3.1. MySQL

En Windows

Previamente necesitamos descargar el ejecutable de la bases de datos, para ello se accederá a la página web oficial, en el apartado descargas e iremos a *MySQL Community (GLP) Downloads* y seleccionaremos la opción *MySQL Installer for Windows*.

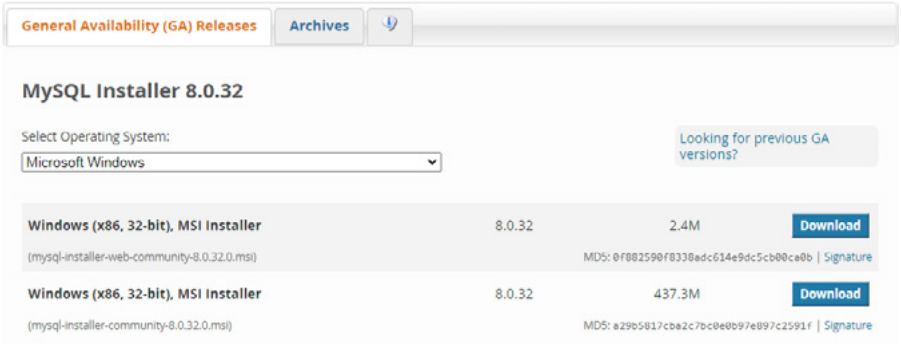


Imagen 2. Descarga de MySQL para Windows

Existen dos instaladores, uno completo para no tener que utilizar internet para la descarga de los ficheros de instalación y otra opción es utilizar el instalador más liviano en tamaño para luego utilizar internet para descargar de los ficheros esenciales.

Proceso:

1. El **asistente de instalación** es bastante intuitivo, ofrece diferentes opciones de instalación, para nuestro caso elegimos la opción *custom* incluyendo el servidor y terminal, las demás funcionalidades no son necesarias.

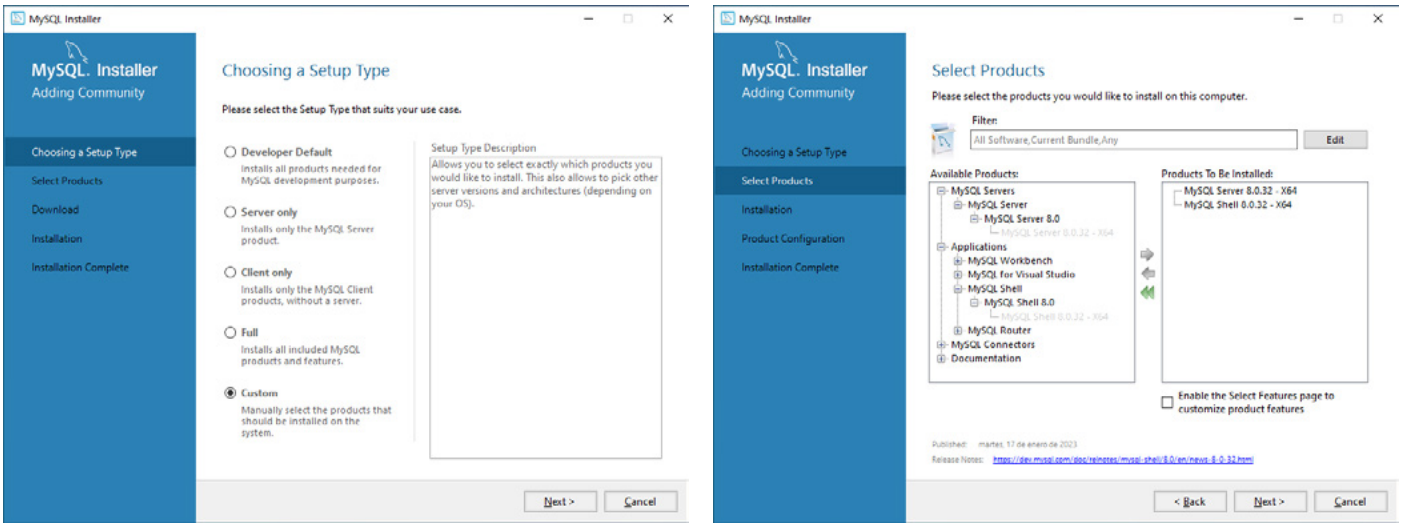


Imagen 3. Primer paso del asistente de instalación de MySQL

2. **Configuración de la conectividad.** Es posible indicar el puerto y otras opciones avanzadas de logging.

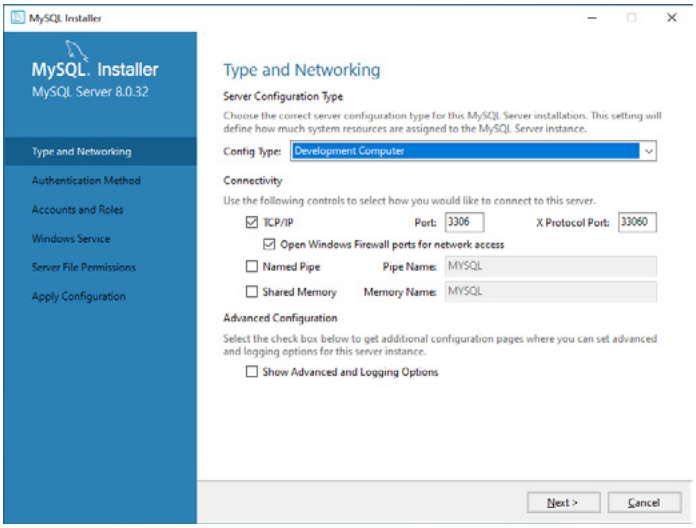


Imagen 4. Configuración de conectividad de MySQL

3. **Configuración de autenticación y usuarios.** Indicar la contraseña que tendrá el usuario principal de administración root.

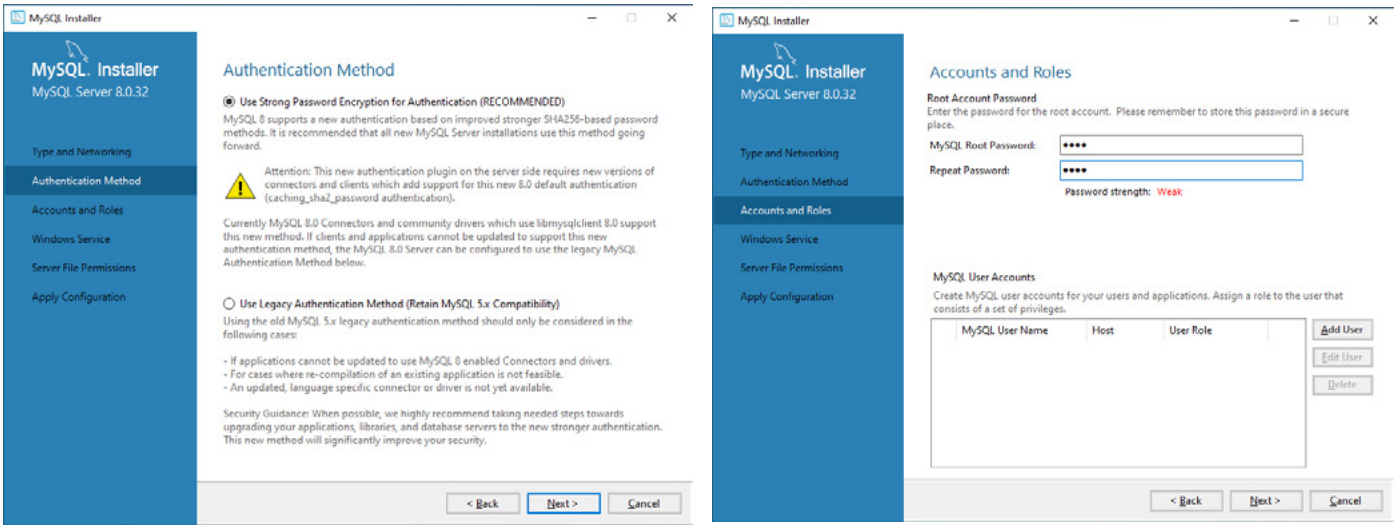


Imagen 5. Configuración de autenticación y usuarios.

4. **Configuración del servicio de Windows.** Indicar el nombre del servicio del servidor y si se comienza automáticamente al arrancar el sistema operativo.

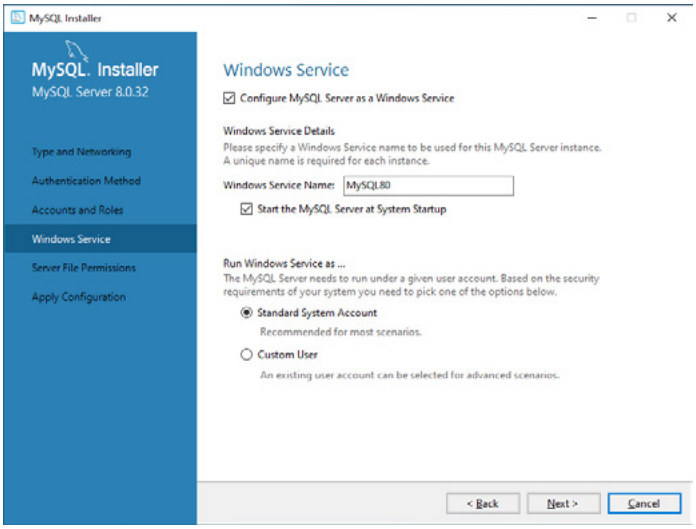
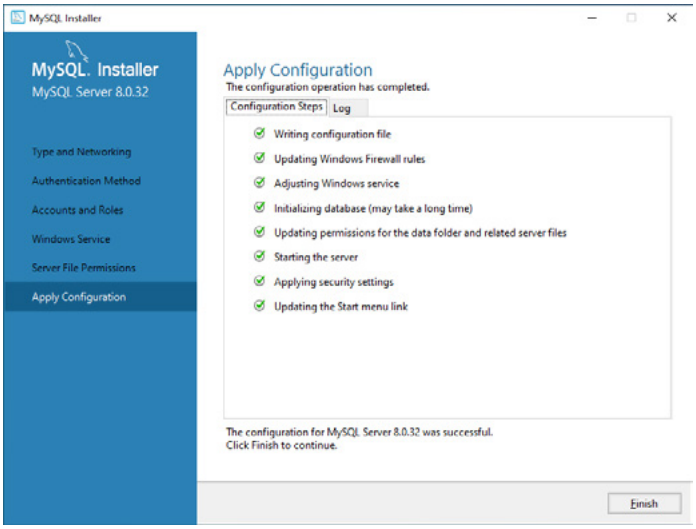


Imagen 6. Configuración del servicio de Windows de MySQL.

5. **Finalizar.** Si todo ha ido correctamente dispondremos del terminal de *MySQL Command Line Client*. nos solicitará la contraseña del usuario root que hemos configurado durante la instalación y una vez logeados podremos ejecutar cualquier sentencia SQL, por ejemplo *select version()* para ver la versión de la base de datos.



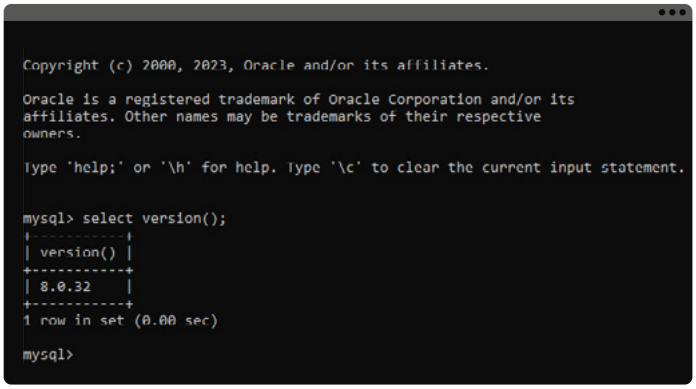


Imagen 7. Finalización de la instalación.

En Linux

A continuación, vamos a ver el proceso de instalación de este SGBD en un sistema operativo Debian:

1. Abrir el terminal del sistema y ejecutar el comando **su root** para loguearnos como superusuario.

2. (Este paso no es necesario en el caso de tener actualizado el sistema operativo.) Previo a la instalación de MySQL hay que añadir un repositorio donde se encuentra diferentes paquetes y librerías esenciales ya que no viene por defecto con Debian. Para realizar esto lanzamos el siguiente comando:

echo "deb http://ftp.debian.org/debian unstable main contrib" > /etc/apt/sources.list.d/debian.list

```
root@debian:/home/user# echo "deb http://ftp.debian.org/debian unstable main con
trib" > /etc/apt/sources.list.d/debian.list
root@debian:/home/user# ls /etc/apt/sources.list.d
debian.list
```

Imagen 8. Añadir el repositorio para instalar MySQL.

3. Una vez que se ha añadido el repositorio, actualizamos los repositorios y actualizamos los paquetes con los comandos **apt update** y **apt upgrade.**

4. Ahora hay que añadir el repositorio que contiene la lista de paquetes de MySQL. Primero descargaremos el repositorio a nuestro local utilizaremos el comando:

wget https://dev.mysql.com/get/mysql-apt-config_0.8.24-1_all.deb

Ya solo queda añadirlo con:

sudo dpkg -i mysql-apt-config_0.8.24-1_all.deb

IMPORTANTE

Es importante mencionar que la versión puede variar. Es recomendable ir a la página oficial para consultar la última versión.

```
root@debian:/home/user# wget https://dev.mysql.com/get/mysql-apt-config_0.8.24-1_all.deb
--2023-02-13 10:31:42--  https://dev.mysql.com/get/mysql-apt-config_0.8.24-1_all.deb
Resolviendo dev.mysql.com (dev.mysql.com)... 23.223.86.162, 2a02:26f0:e0:594::2e31, 2a02:26f0:e0:5bd::2e31
Conectando con dev.mysql.com (dev.mysql.com)[23.223.86.162]:443... conectado.
Petición HTTP enviada, esperando respuesta... 302 Moved Temporarily
Localización: https://repo.mysql.com//mysql-apt-config_0.8.24-1_all.deb [siguiendo]
--2023-02-13 10:31:42--  https://repo.mysql.com//mysql-apt-config_0.8.24-1_all.deb
Resolviendo repo.mysql.com (repo.mysql.com)... 184.24.68.238
Conectando con repo.mysql.com (repo.mysql.com)[184.24.68.238]:443... conectado.
Petición HTTP enviada, esperando respuesta... 200 OK
Longitud: 18048 (18K) [application/x-debian-package]
Grabando a: «mysql-apt-config_0.8.24-1_all.deb»

mysql-apt-config_0. 100%[==================>]  17,62K  --.-KB/s    en 0,002s

2023-02-13 10:31:42 (7,45 MB/s) - «mysql-apt-config_0.8.24-1_all.deb» guardado [18048/18048]
```

```
root@debian:/home/user# sudo dpkg -i mysql-apt-config_0.8.24-1_all.deb
Seleccionando el paquete mysql-apt-config previamente no seleccionado.
(Leyendo la base de datos ... 144624 ficheros o directorios instalados actualmente.)
Preparando para desempaquetar mysql-apt-config_0.8.24-1_all.deb ...
Desempaquetando mysql-apt-config (0.8.24-1) ...
Configurando mysql-apt-config (0.8.24-1) ...
Warning: apt-key should not be used in scripts (called from postinst maintainerscript of the package mysql-apt-config)
Warning: apt-key is deprecated. Manage keyring files in trusted.gpg.d instead (see apt-key(8)).
OK
```

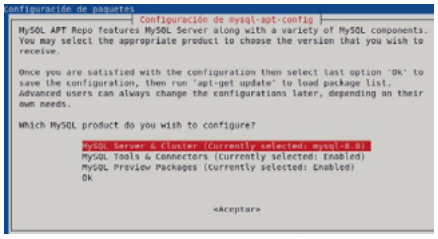


Imagen 9. Procedimiento para añadir repositorio de MySQL

5. Volvemos a actualizar los paquetes **sudo apt-get update**
6. Procedemos a instalar MySQL.

```
sudo apt-get install mysql-server
```

» Durante el proceso de instalación nos solicitara la contraseña del usuario root y su confirmación.

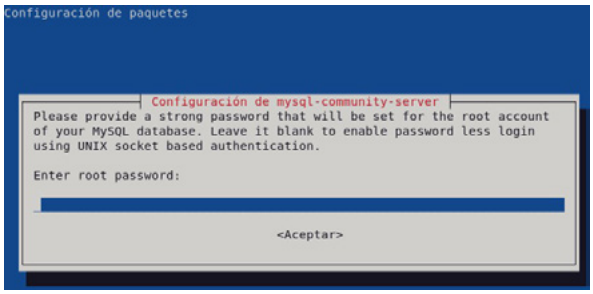


Imagen 10. Solicitud de contraseña del usuario root.

» Solicitará la forma de autenticación. Se dejará la que viene por defecto

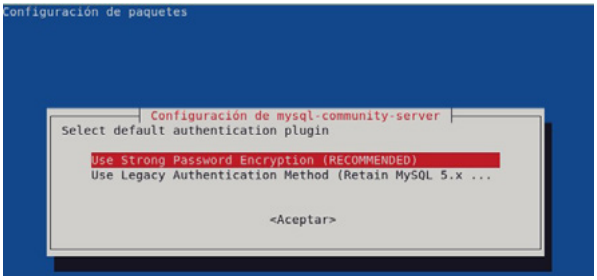


Imagen 11. Forma de autenticación.

» Finalmente habrá que reiniciar.

7. Verificamos que se ha instalado correctamente con el comando, **systemctl status mysql**

```
root@debian:/home/user# systemctl status mysql
● mysql.service - MySQL Community Server
     Loaded: loaded (/lib/systemd/system/mysql.service; enabled; vendor preset:
     Active: active (running) since Thu 2023-02-16 10:25:23 CET; 24s ago
       Docs: man:mysqld(8)
             http://dev.mysql.com/doc/refman/en/using-systemd.html
   Main PID: 7398 (mysqld)
     Status: "Server is operational"
      Tasks: 39 (limit: 2278)
     Memory: 365.9M
        CPU: 640ms
     CGroup: /system.slice/mysql.service
             └─7398 /usr/sbin/mysqld

feb 16 10:25:22 debian systemd[1]: Starting MySQL Community Server...
feb 16 10:25:23 debian systemd[1]: Started MySQL Community Server.
```

Imagen 12. Verificación del servidor de base de datos

8. Conectamos con la base de datos con, **mysql -p**

```
root@debian:/home/user# mysql -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.0.32 MySQL Community Server - GPL

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

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

mysql> select version();
+-----------+
| version() |
+-----------+
| 8.0.32    |
+-----------+
1 row in set (0,00 sec)
```

Imagen 13. Conexión con el servidor de base de datos MySQL

4.3.2. MariaDB

En Windows

Previamente necesitamos descargar el ejecutable de la base de datos, para ello se accederá a la página web oficial, en el apartado descargas y seleccionaremos la versión y el sistema operativo Windows.

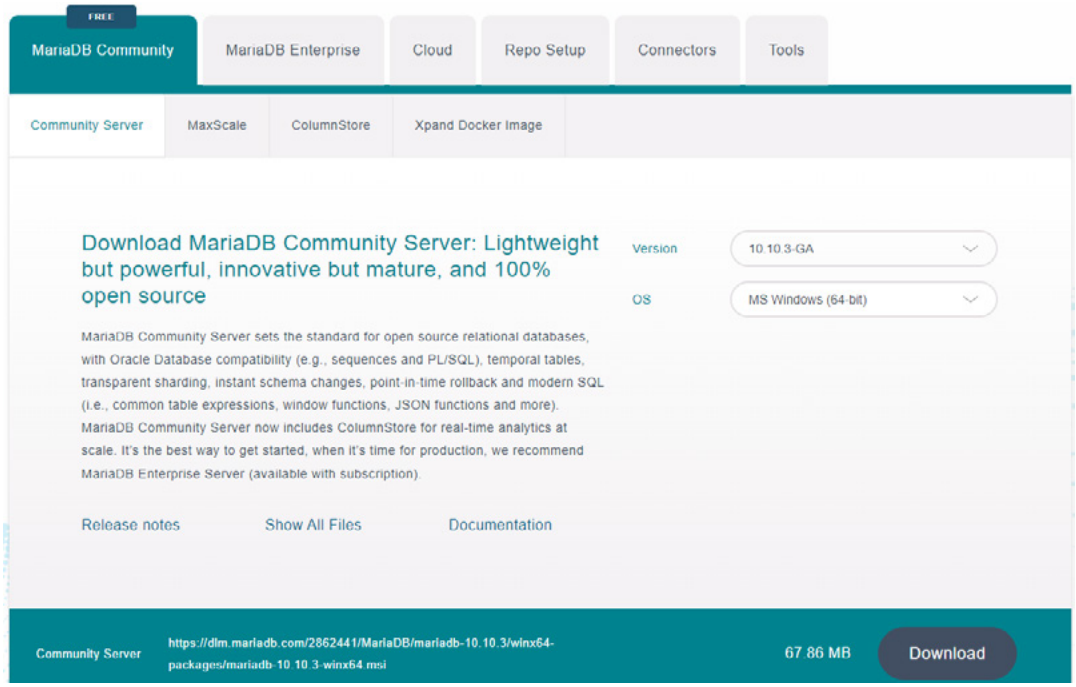


Imagen 14. Descarga de MariaDB para Windows

Proceso:

1. **Asistente de instalación.** La primera opción es elegir las funcionalidades que ofrece MariaDB, en este caso, vamos a dejar por defecto todas las opciones seleccionadas. La más interesantes es HeidiSQL un cliente gráfico que nos facilita poder conectarnos al servidor y trabajar con él.

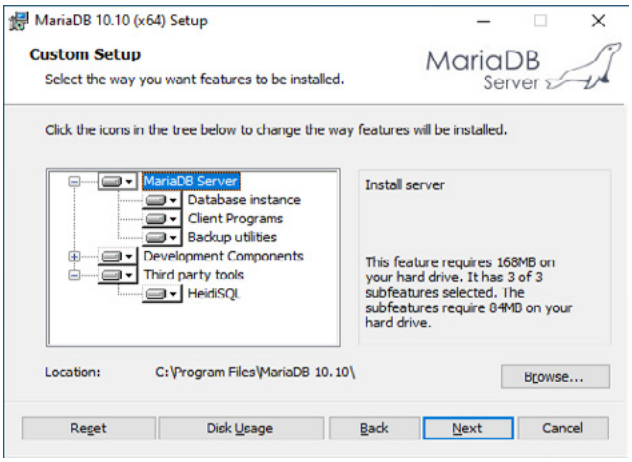


Imagen 15. Primer paso del asistente de instalación de MariaDB

2. **Configuración de autenticación del usuario root y ruta de datos.** Indicar la contraseña que tendrá el usuario principal de administración root, la codificación de los caracteres y la ruta de los datos del servidor.

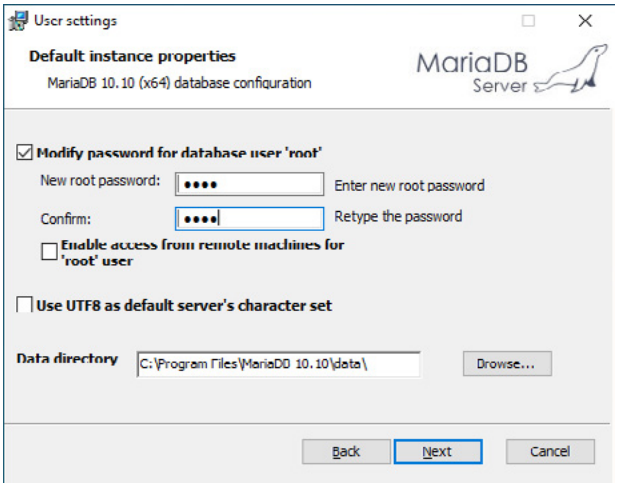


Imagen 16. Configuración de autenticación y ruta de datos en MariaDB.

3. **Configuración del servicio de Windows y conectividad.** Indicar el nombre del servicio del servidor, el puerto y el tamaño del buffer de la base de datos.

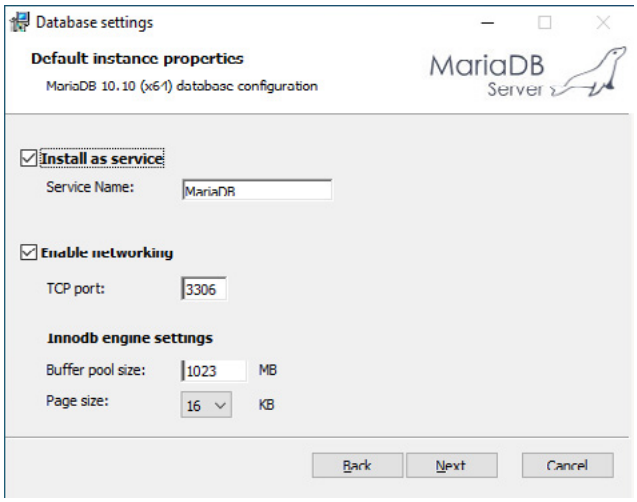


Imagen 17. Configuración del servicio de Windows y conectividad en MariaDB.

4. **Finalizar.** Si todo ha ido correctamente dispondremos del terminal de MySQL Client para MariaDB. Nos solicitará la contraseña del usuario root que hemos configurado durante la instalación y una vez logeados podremos ejecutar cualquier sentencia SQL, por ejemplo *select version()* para ver la versión de la base de datos.

Imagen 18. Finalización de la instalación de MariaDB.

En Linux (Debian)

Es recomendable tener actualizado todos los paquetes y librerías del sistema antes de proceder con la instalación de MariaDB.

Proceso:

1. Abrir el terminal del sistema y ejecutar el comando **su root** para loguearnos como superusuario.
2. Actualizar la lista del repositorio **apt-get update**
3. Instalar el servidor de la base de datos, ejecutando el comando **sudo apt-get install mariadb-server**

Es posible que requiera algunos paquetes adicionales procedentes de la ISO o CD de instalación de Debian.

```
root@debian:/home/user# sudo apt-get install mariadb-server
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes adicionales:
  galera-4 gawk libaio1 libcgi-fast-perl libcgi-pm-perl
  libconfig-inifiles-perl libdbd-mysql-perl libdbi-perl libfcgi-bin
  libfcgi-perl libfcgi0ldbl libhtml-template-perl libmariadb3 libsigsegv2
  libterm-readkey-perl mariadb-client-10.5 mariadb-client-core-10.5
  mariadb-common mariadb-server-10.5 mariadb-server-core-10.5 mysql-common
  rsync socat
Paquetes sugeridos:
  gawk-doc libmldbm-perl libnet-daemon-perl libsql-statement-perl
  libipc-sharedcache-perl mailx mariadb-test netcat-openbsd openssh-server
Se instalarán los siguientes paquetes NUEVOS:
  galera-4 gawk libaio1 libcgi-fast-perl libcgi-pm-perl
  libconfig-inifiles-perl libdbd-mysql-perl libdbi-perl libfcgi-bin
  libfcgi-perl libfcgi0ldbl libhtml-template-perl libmariadb3 libsigsegv2
  libterm-readkey-perl mariadb-client-10.5 mariadb-client-core-10.5
  mariadb-common mariadb-server mariadb-server-10.5 mariadb-server-core-10.5
  mysql-common rsync socat
0 actualizados, 24 nuevos se instalarán, 0 para eliminar y 0 no actualizados.
Se necesita descargar 0 B/17,2 MB de archivos.
Se utilizarán 158 MB de espacio de disco adicional después de esta operación.
```

Imagen 19. Ejecución del comando de instalación de MariaDB.

4. Verificar que se ha instalado correctamente con el comando, **mariadb --version**

```
root@debian:/home/user# mariadb --version
mariadb  Ver 15.1 Distrib 10.5.18-MariaDB, for debian-linux-gnu (x86_64) using
EditLine wrapper
```

Imagen 20. Comprobar la instalación y versión de MariaDB

5. Configurar la seguridad de acceso a la base de datos con el comando **sudo mysql_secure_installation.**

   Al ejecutar el comando nos saltará un asistente de texto para especificarle diferentes configuraciones:

   - Nos preguntara la contraseña del usuario root. Indicar una contraseña y apuntarla.
   - Si queremos cambiar el acceso a la base de datos con el sistema de autenticación de unix. Indicamos N
   - Si queremos cambiar la contraseña del usuario root. Indicamos N
   - Si borramos los usuarios Anonymous. Indicamos Y
   - Si deshabilitamos el login remoto. Indicamos Y
   - Si borramos la base de datos test y su acceso. Indicamos N

```
root@debian:/home/user# 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
haven't set the root password yet, you should just press enter here.

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

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

You already have your root account protected, so you can safely answer 'n'.

Switch to unix_socket authentication [Y/n] n
 ... skipping.

You already have your root account protected, 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!
```

Imagen 21. Configurar aspectos de seguridad de acceso a MariaDB

6. Acceder a la base de datos con el comando **mysql -u root -p .** Nos solicitara la contraseña del usuario root que hemos indicado en el paso anterior.

```
root@debian:/home/user# mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 36
Server version: 10.5.18-MariaDB-0+deb11u1 Debian 11

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)]> select version();
+-------------------------+
| version()               |
+-------------------------+
| 10.5.18-MariaDB-0+deb11u1 |
+-------------------------+
1 row in set (0,000 sec)
```

Imagen 22. Acceder a la base de datos MariaDB

4.4. Lenguaje de definición de datos (DDL)

Como se ha comentado anteriormente, la función del lenguaje *DDL* es la de crear, modificar, y borrar objetos presentes en una base de datos y también la propia base de datos.

4.4.1. Definición de bases de datos

Para crear una base de datos

```
CREATE DATABASE nombre_bd;
```

En el siguiente ejemplo creamos una base de datos que se va a llamar *bduniversae.*

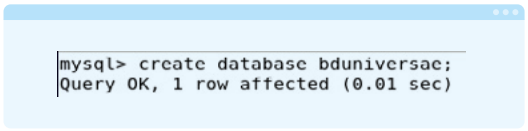


Imagen 23. Ejemplo de creación de base de datos.

Ahora lanzamos el siguiente comando para ver si la base de datos se ha añadido al sistema:

```
SHOW databases;
```

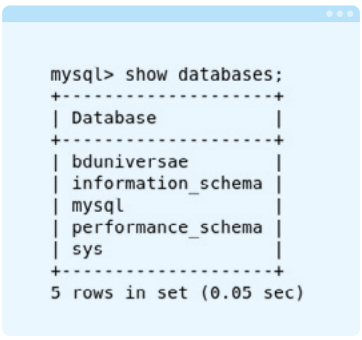


Imagen 24. Bases de datos del sistema.

Si nos fijamos, MySQL trae por defecto cuatro bases de datos para la gestión del SGBD, entonces con la que hemos creado serían 5 las bases de datos que tenemos.

Para borrar una base de datos

```
DROP DATABASE nombre_bd;
```

Siguiendo nuestro ejemplo, borramos la base de datos que hemos creado anteriormente.

```
mysql> drop database bduniversae;
Query OK, 0 rows affected (0.01 sec)
```

Imagen 25. Ejemplo de borrado de una base de datos.

Ahora lanzamos de nuevo el comando para ver las bases de datos que tenemos en el SGBD y comprobamos que ha desaparecido.

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
4 rows in set (0.01 sec)
```

Imagen 26. Bases de datos del sistema.

4.4.2. Definición de tablas

Para crear una tabla

```
CREATE TABLE nombre_tabla (
    campo1 tipo[(longitud)] [NOT NULL] [UNIQUE]
    [PRIMARY KEY] [CHECK condición] [DEFAULT
    valor][, [
    …
    campoN … ]
    [,  PRIMARY KEY (campos)]
    [, FOREIGN KEY (campos) REFERENCES tabla
    (campos)
    [{ON UPDATE [NO ACTION|SET DEFAULT|SET
    NULL|CASCADE]
    [ON DELETE [NO ACTION|SET DEFAULT|SET
    NULL|CASCADE]]
    }|
    {ON DELETE [NO ACTION|SET DEFAULT|SET
    NULL|CASCADE]
    [ON UPDATE [NO ACTION|SET DEFAULT|SET
    NULL|CASCADE]]
    }]]
    …
);
```

Para poder crear una tabla, primero tendremos que solicitar usar una base de datos en concreto, con el comando:

```
use nombre_bd;
```

```
mysql> use bdempresa;
Database changed
```

Imagen 27. Usar una base de datos.

Vamos a crear ahora una de las tablas que se crearon en ejemplos de la unidad anterior:

```
mysql> create table TProveedor (
    -> nCodProveedorID NUMERIC(5,0) NOT NULL UNIQUE PRIMARY KEY,
    -> cNombreProveedor VARCHAR(50) NOT NULL,
    -> cWeb VARCHAR(50),
    -> nTelefono NUMERIC(9,0) NOT NULL UNIQUE,
    -> cDireccion VARCHAR(50)
    -> );
Query OK, 0 rows affected (0.07 sec)
```

Imagen 28. Ejemplo de creación de una tabla.

Como se puede ver, los campos 'cNombreProveedor' y ' nTelefono' no pueden estar nulos, y además, el campo para guardar los datos del teléfono también debe ser único, pues dos trabajadores no pueden tener el mismo número de teléfono.

```
mysql> create table TProvision (
    -> nCodProveedorID NUMERIC(5,0) NOT NULL UNIQUE,
    -> nProductoID NUMERIC(7,0) NOT NULL UNIQUE,
    -> CONSTRAINT PK_Provision PRIMARY KEY(nCodProveedorID,nProductoID),
    -> FOREIGN KEY (nCodProveedorID) REFERENCES TProveedor (nCodProveedorID)
    -> ON UPDATE CASCADE
    -> ON DELETE CASCADE
    -> );
Query OK, 0 rows affected (0.03 sec)
```

Imagen 29. Creación de tabla con clave foránea.

Ahora hemos creado una tabla adicional, la tabla 'TProvision', y como podemos ver, hemos indicado que hay una doble clave primaria y que además uno de los campos que referencia a la clave primaria hace referencia a la clave primaria de 'TProveedor'. Por último, hemos añadido 'ON UPDATE CASCADE ON DELETE CASCADE', que quiere decir que, si se actualiza la clave principal o se elimina un registro, se actualice dicha información en la clave foránea.

- **Para modificar una tabla,** Para realizar modificaciones en tablas usamos la sentencia **ALTER TABLE** y con esta podemos:

  - Añadir un campo:

```
ALTER TABLE nombre_tabla
ADD campo tipo[(longitud)] [NOT NULL] [UNIQUE]
[PRIMARY KEY]
[CHECK condición] [DEFAULT valor];
```

Vamos a añadir a la tabla 'TProveedor' un campo 'cCorreo' donde escribir la dirección de correo electrónico.

```
mysql> ALTER TABLE TProveedor
    -> ADD cCorreo VARCHAR(256);
Query OK, 0 rows affected (0.05 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Imagen 30. Añadir un campo a una tabla.

  - Modificar un campo:

```
ALTER TABLE nombre_tabla
MODIFY COLUMN campo [tipo(longitud)] | [DROP
DEFAULT];
```

Antes habíamos creado el campo 'nTelefono' como numérico, pero realmente si vemos la explicación que dimos anteriormente de los tipos de datos, sería una cadena de caracteres, puesto que no se van a realizar operaciones con los números de teléfono, vamos a cambiarlo.

```
mysql> ALTER TABLE TProveedor
    -> Modify Column nTelefono VARCHAR(9);
Query OK, 0 rows affected (0.07 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Imagen 31. Modificar un campo de una tabla.

  - Borrar un campo:

```
ALTER TABLE nombre_tabla
DROP campo;
```

Borramos el campo anteriormente creado.

```
mysql> ALTER TABLE TProveedor
    -> DROP cCorreo;
Query OK, 0 rows affected (0.05 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Imagen 32. Borrar el campo de una tabla.

IMPORTANTE

DROP DEFAULT se usa para borrar el dato que se haya puesto por defecto, si es que lo hay.

Para eliminar una tabla

```
DROP TABLE nombre_tabla;
```

Vamos a eliminar ahora la última tabla creada, 'TProvision'.

```
mysql> drop table TProvision
    -> ;
Query OK, 0 rows affected (0.02 sec)
```

Imagen 33. Ejemplo de borrado de una tabla.

Para comprobar el estado de una tabla y que campos tiene junto con sus características, se usa el comando:

```
DESC nombre_tabla;
```

Describimos a continuación la tabla que nos queda, que es 'TProveedor'.

```
mysql> desc TProveedor;
+------------------+--------------+------+-----+---------+-------+
| Field            | Type         | Null | Key | Default | Extra |
+------------------+--------------+------+-----+---------+-------+
| nCodProveedorID  | decimal(5,0) | NO   | PRI | NULL    |       |
| cNombreProveedor | varchar(50)  | NO   |     | NULL    |       |
| cWeb             | varchar(50)  | YES  |     | NULL    |       |
| nTelefono        | varchar(9)   | YES  | UNI | NULL    |       |
| cDireccion       | varchar(50)  | YES  |     | NULL    |       |
+------------------+--------------+------+-----+---------+-------+
5 rows in set (0.02 sec)
```

Imagen 34. Ejemplo de descripción de una tabla.

4.4.3. Definición de índices

Los índices se usan para marcar ciertos campos con la intención de asignarles una importancia o relevancia para luego poder trabajar sobre ellos con mayor facilidad.

A continuación, se muestra una sintaxis algo genérica sobre cómo se trabaja con los índices:

Para crear un índice

```
CREATE [UNIQUE] INDEX nombre_indice
ON nombre_tabla
(campo1 [ASC|DESC][, campo2 [ASC|DESC][, …, campoN [ASC|DESC]]);
```

Si especificamos *ASC* o *DESC* en alguno de los campos estamos haciendo referencia a que el orden del índice será ascendente o descendente, por defecto será descendente.

Vamos a crear un índice para agrupar el nombre y el teléfono de un proveedor.

```
mysql> CREATE INDEX iProveedor_Nombre_Telefono
    -> ON TProveedor
    -> (cNombreProveedor, nTelefono);
Query OK, 0 rows affected (0.06 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Imagen 35. Ejemplo de creación de índice.

Para borrar un índice

```
DROP INDEX nombre_indice
ON nombre_tabla;
```

Borramos el índice anteriormente creado.

```
mysql> DROP INDEX iProveedor_Nombre_Telefono on TProveedor;
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Imagen 36. Ejemplo de borrado de índice.

En *MySQL* también se puede usar la modificación de tablas para borrar índices como en el ejemplo siguiente:

```
mysql> alter table TProveedor
    -> drop index iProveedor_Nombre_Telefono;
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Imagen 37. Otra forma de borrar índices.

4.4.4. Definición de tipos de datos

En algunas bases de datos el usuario puede crear tipos como quiera para reutilizarlo a la hora de crear o modificar campos para las tablas. Estos tipos nuevos realmente son alias de otros tipos.

La sentencia para crear tipos es:

```
CREATE TYPE nombre_tipo AS tipo[(longitud)];
```

En MySQL, por ejemplo, no funciona la creación de tipos.

4.5. Resumen de comandos

Comando	Función
show databases;	Visualiza las bases de datos actualmente activas
use *database;*	Nos permite utilizar una base de datos
show tables;	Muestra las tablas de la base de datos actual
desc *table*;	Nos permite ver la estructura completa de una tabla
create database *database*;	Crea una base de datos
drop database *database*;	Borra la base de datos seleccionada
create table *table*;	Crea una tabla
alter table *table*	
+ **change** *campo*;	+ Modifica un campo
+ **rename** *nombre*;	+ Cambia el nombre a la tabla
+ **drop** *columna*;	+ Borra una columna
+ **add** *columba*;	+ Añade una columna
+ **type=innoDB;**	+ Se convierte la tabla a tipo innoDB
+ **add index** *indice (campos)*;	+ Crea índices
+ **add unique index** *indice (campos)*;	+ Crea un índice único
+ **add primary key** *(campos)*;	+ Se crean las claves primarias de la tabla
+ **drop index** *indice*;	+ Se borra un índice
+ **drop primary key;**	+ Borra la clave primaria
+ **modify..**	+ Para cambiar el tipo de una columna sin renombrarla
drop table *table*;	Borra una tabla
create index *indice* **on** *table (campos)*;	crea índices
drop index *indice* **on** *table*;	Borra un índice
create index *indice* **on** *table (campos)*;	Crea índices
drop index *indice* **on** *table*;	Borra un índice
select user();	Muestra el usuario actual
select version();	Muestra la versión actual de mysql
select now(); / select sysdate();	Muestra la fecha y hora actual
select database();	Muestra la base de datos actual
select curdate(); / select current_date;	Muestra la fecha actual
truncate table table;	Borra la tabla y la vuelve a crear vacía
create type *nombre*;	Crea un tipo de dato según el tipo de base de dato

4.6. Caso práctico

Se solicita realizar una base de datos del mundo de Pokemons. Nos piden crear la estructura de la base de datos según la siguiente descripción:

"Es necesario registrar los entrenadores, identificando cada uno de ellos con un código numérico, y deben disponer de la siguiente información, nombre, sexo, dinero, fecha en que empieza la aventura y el tiempo jugado que registrara el número de minutos.

Cada entrenador solo puede poseer una Pokédex que registrará el número total de los Pokemons vistos y capturados. Las Pokédex se identifican por un código numérico.

Se debe de disponer de todos los Pokemos existentes. Por cada uno de ellos se identificará por un código numérico, junto con su nombre que debe ser único, sexo, peso, altura y si es legendario o no. Un Pokémon deben poder conocer a que Pokémon evoluciona, siempre y cuando exista tal evolución.

Una Pokédex debe registrar todos los Pokemons que un entrenador ha visto o capturado, conociendo la fecha en que se produce el evento, la experiencia y el nivel si el Pokémon es capturado, en el caso que solo se haya visto, no será necesario indicar la experiencia y nivel.

Además de los Pokemons, se debe de poder conocer los tipos que pertenece cada uno y cada tipo de Pokémon debe disponer de los tipos de ataque que pueden hacer, cada nombre de ataque debe ser único.

En los mapas de juego existen regiones identificadas por un código numérico, nombre, coordenadax, coordenaday y extensión. Una región puede disponer de varias ciudades y una ciudad solo puede estar en una región, además las regiones deben de poder identificar el tipo de área que son, por ejemplo, Desierto, Campo, Bosque, Tundra, etc..

Los Pokemos pueden pertenecer a más de una región y una región puede contener diferentes Pokemons. Para concluir, se debe de poder conocer la fecha en el que un entrenador visita una región y el tiempo que ha estado."

Adicionalmente nos indican que se van a realizar muchas operaciones sobre los campos nombre de un entrenador y el nombre de una región. Determina que elemento de estructura es necesario para optimizarlo y muestra el comando necesario.

Diagrama E/R

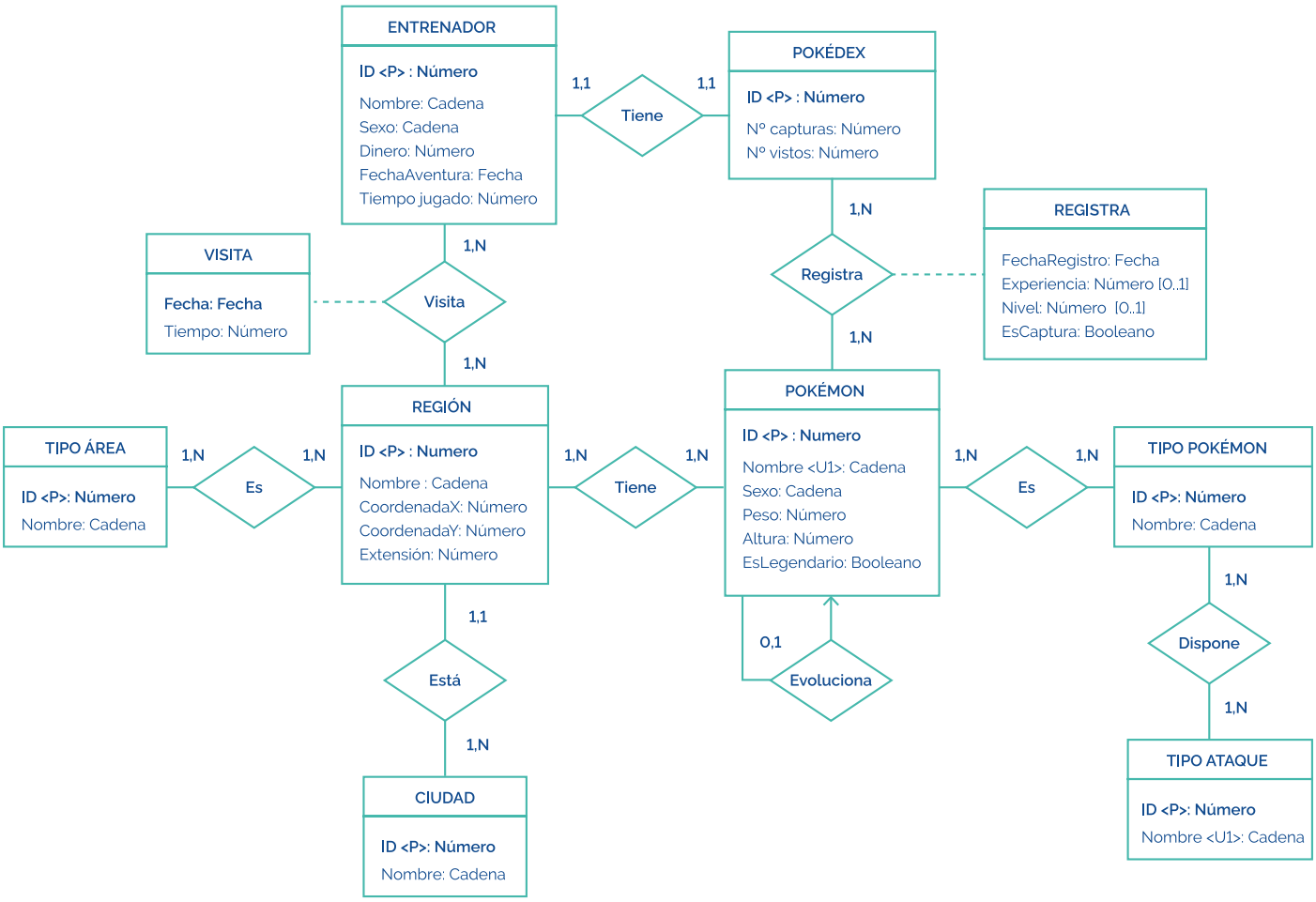


Imagen 38. Diagrama E/R de la base de datos del mundo de Pokemons

Modelo físico

POKEDEXS (Código, **NumCapturas, NumVistos**)

ENTRENADORES (Código, **Nombre, Sexo, Dinero, FechaAventura, TiempoJugado, CódigoPokedex**)

- {CódigoPokedex} es clave foránea de POKEDEXS

POKEMONS (Código, Nombre, **Sexo, Peso, Altura, Legendario,** Evolución)

- {Nombre} es clave alternativa
- {Evolución} es clave foránea de POKEMONS

REGISTROS (CódigoPokedex, CódigoPokemon, FechaRegistro, Experiencia, Nivel, **Captura)**

- {CódigoPokedex} es clave foránea de POKEDEXS
- {CódigoPokemon} es clave foránea de POKEMONS

TIPOSATAQUE (Código Nombre)

- {Nombre} es clave alternativa

TIPOSPOKEMON (Código, **Nombre**)

ATAQUESPORTIPO (CódigoTipo, CódigoTipoAtaque)

- {CódigoTipo} es clave foránea de TIPOSPOKEMON
- {CódigoTipoAtaque} es clave foránea de TIPOSATAQU

TIPOS (CódigoPokemon, CódigoTipo)

- {CódigoPokemon} es clave foránea de POKEMONS
- {CódigoTipo} es clave foránea de TIPOSPOKEMON

REGIONES (Código, **Nombre, CoordenadaX, CoordenadaY, Extensión**)

CIUDADES (Código, **Nombre, CódigoRegión**)

- {CódigoRegión} es clave foránea de REGIONES

TIPOSAREA (Código, **Nombre**)

VISITAS (CódigoRegión, CódigoEntrenador, Fecha, **Tiempo**)

- {CódigoRegión} es clave foránea de REGIONES
- {CódigoEntrenador} es clave foránea de ENTRENADORES

AREASPORREGION (CódigoRegión, CódigoÁrea)

- {CódigoRegión} es clave foránea de REGIONES
- {CódigoÁrea} es clave foránea de TIPOSAREAS

POKEMONSPORREGION (CódigoRegión, CódigoPokemon)

- {CódigoRegión} es clave foránea de REGIONES
- {CódigoPokemon} es clave foránea de POKEMONS

Solución:

1. El primer paso para crear la estructura es creará la base de datos

```
CREATE DATABASE pokemonDB;
```

2. Especificamos el uso de la base de datos creada.

```
USE pokemonDB;
```

3. Creamos las tablas. El orden es muy importante, primero creamos las tablas con menor dependencia en las relaciones, ya que si ponemos primero tablas con claves foráneas, nos dará un error que no existe la tabla a la que hace referencia.

```
CREATE OR REPLACE TABLE POKEDEXS (
   codigo INT AUTO_INCREMENT,
   capturas INT NOT NULL,
   vistos INT NOT NULL,
   CONSTRAINT PK_POKEDEXS PRIMARY KEY (codigo)
);
CREATE OR REPLACE TABLE ENTRENADORES (
   codigo INT AUTO_INCREMENT,
   nombre VARCHAR(15) NOT NULL,
   sexo VARCHAR(10) NOT NULL,
   dinero FLOAT NOT NULL,
   fechaaventura DATE NOT NULL,
   tiempojugado INT NOT NULL,
   codigoPokedex INT NOT NULL,
   CONSTRAINT PK_ENTRENADORES PRIMARY KEY
   (codigo),
   CONSTRAINT FK_POKEDEX FOREIGN KEY (codigoP-
   okedex)
      REFERENCES POKEDEXS(codigo)
);
CREATE OR REPLACE TABLE POKEMONS (
   codigo INT AUTO_INCREMENT,
   nombre VARCHAR(15) NOT NULL,
   peso FLOAT NOT NULL,
   altura FLOAT NOT NULL,
   legendario BOOLEAN NOT NULL,
   evolucion INT,
   CONSTRAINT PK_POKEMONS PRIMARY KEY (codigo),
   CONSTRAINT U_NOMBRE UNIQUE (nombre),
   CONSTRAINT FK_EVOLUCION FOREIGN KEY (evolucion)
      REFERENCES POKEMONS(codigo)
);
```

```
CREATE OR REPLACE TABLE REGISTROS (
   codigoPokedex INT NOT NULL,
   codigoPokemon INT NOT NULL,
   fechaRegistro DATETIME NOT NULL,
   experiencia FLOAT,
   nivel INT,
   captura BOOLEAN NOT NULL,
   CONSTRAINT PK_REGISTROS PRIMARY KEY (codigoP-
   okedex, codigoPokemon, fechaRegistro)
);
CREATE OR REPLACE TABLE TIPOS_ATAQUE (
   codigo INT AUTO_INCREMENT,
   nombre VARCHAR(10) NOT NULL,
   CONSTRAINT PK_TIPOATAQUES PRIMARY KEY (codigo),
   CONSTRAINT U_TIPO UNIQUE (nombre)
);
CREATE OR REPLACE TABLE TIPOS_POKEMON (
   codigo INT AUTO_INCREMENT,
   nombre VARCHAR(10) NOT NULL,
   CONSTRAINT PK_TIPOSPOKEMON PRIMARY KEY (codigo)
);
CREATE OR REPLACE TABLE ATAQUES_POR_TIPO (
   codigoTipo INT NOT NULL,
   codigoAtaque INT NOT NULL,
   CONSTRAINT  PK_ATAQUESPORTIPO  PRIMARY  KEY
   (codigoTipo, codigoAtaque),
   CONSTRAINT   FK_TIPOPOKEMON   FOREIGN   KEY
   (codigoTipo)
      REFERENCES TIPOS_POKEMON(codigo),
   CONSTRAINT   FK_TIPOATAQUE   FOREIGN   KEY
   (codigoAtaque)
      REFERENCES TIPOS_ATAQUE(codigo)
);
CREATE OR REPLACE TABLE TIPOS (
   codigoPokemon INT NOT NULL,
   codigoTipo INT NOT NULL,
   CONSTRAINT   PK_POKEMONTIPOS   PRIMARY   KEY
   (codigoPokemon, codigoTipo),
   CONSTRAINT FK_POKEMON FOREIGN KEY (codigoP-
   okemon)
      REFERENCES POKEMONS(codigo),
   CONSTRAINT FK_TIPO FOREIGN KEY (codigoTipo)
      REFERENCES TIPOS_POKEMON(codigo)
);
```

```
CREATE OR REPLACE TABLE REGIONES (
   codigo INT AUTO_INCREMENT,
   nombre VARCHAR(15) NOT NULL,
   coordenadaX FLOAT NOT NULL,
   coordenadaY FLOAT NOT NULL,
   extension REAL NOT NULL,
   CONSTRAINT PK_REGIONES PRIMARY KEY (codigo)
);
CREATE OR REPLACE TABLE CIUDADES (
   codigo INT AUTO_INCREMENT,
   nombre VARCHAR(15) NOT NULL,
   codigoRegion INT NOT NULL,
   CONSTRAINT PK_CIUDADES PRIMARY KEY (codigo),
   CONSTRAINT FK_REGION FOREIGN KEY (codigoRe-
   gion)
      REFERENCES REGIONES(codigo)
);
CREATE OR REPLACE TABLE TIPOS_AREA (
   codigo INT AUTO_INCREMENT,
   nombre VARCHAR(15) NOT NULL,
   CONSTRAINT PK_TIPO_AREAS PRIMARY KEY (codigo)
);
CREATE OR REPLACE TABLE VISITAS (
   codigoRegion INT NOT NULL,
   codigoEntrenador INT NOT NULL,
   fecha DATETIME NOT NULL,
   tiempo INT NOT NULL,
   CONSTRAINT PK_VISITAS PRIMARY KEY (codigoRe-
   gion, codigoEntrenador, fecha),
   CONSTRAINT FK_REGIONVISITA FOREIGN KEY
   (codigoRegion)
      REFERENCES REGIONES(codigo),
   CONSTRAINT FK_ENTRENADORVISITA FOREIGN KEY
   (codigoEntrenador)
      REFERENCES ENTRENADORES(codigo)
);
```

```
CREATE OR REPLACE TABLE AREAS_POR_REGION (
   codigoRegion INT NOT NULL,
   codigoArea INT NOT NULL,
   CONSTRAINT PK_AREASPORREGION PRIMARY KEY
   (codigoRegion, codigoArea),
   CONSTRAINT FK_REGIONAREA FOREIGN KEY (codig-
   oRegion)
      REFERENCES REGIONES(codigo),
   CONSTRAINT FK_AREASREGION FOREIGN KEY
   (codigoArea)
      REFERENCES TIPOS_AREA(codigo)
);
CREATE OR REPLACE TABLE POKEMONS_POR_REGION (
   codigoRegion INT NOT NULL,
   codigoPokemon INT NOT NULL,
   CONSTRAINT PK_POKEMONSPORREGION PRIMARY KEY
   (codigoRegion, codigoPokemon),
   CONSTRAINT FK_REGIONPORPOKEMON FOREIGN KEY
   (codigoRegion)
      REFERENCES REGIONES(codigo),
   CONSTRAINT FK_POKEMONPORREGION FOREIGN KEY
   (codigoPokemon)
      REFERENCES POKEMONS(codigo)
);
```

4. Para finalizar, los campos nombre de la tabla entrenador y nombre de la tabla región van a recibir bastantes operaciones sobre sus datos, en esta situación es recomendable crear un índice para cada campo para optimizar las operaciones de consulta.

```
CREATE INDEX IDX_NOMBRE_ENTRENADORES
   ON ENTRENADORES (nombre DESC);
CREATE INDEX IDX_NOMBRE_REGIONES
   ON REGIONES (nombre DESC);
```

www.universae.com