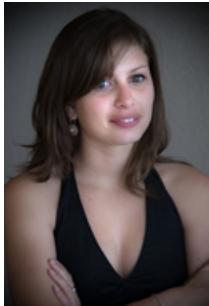


# Diseño físico de bases de datos.

## Caso práctico



Alain Bachellier ([CC BY-NC-SA](#))

En la unidad anterior dejamos a **Vindio y Noiba diseñando un modelo lógico para implantar un sistema de bases de datos en el taller mecánico**. Una vez establecido el modelo lógico, **el siguiente paso consiste en establecer el diseño físico**, compuesto de tablas y relaciones. Pasando de lo abstracto a lo concreto.

Para conseguir esto, la mayoría de los **SGBD** disponen de herramientas gráficas que facilitan el **paso entre el diseño de modelos de datos Entidad/Relación y el diseño de tablas concretas del modelo relacional**.

Veremos que hay muchas similitudes entre ambas representaciones y, si además hemos comprendido el paso del modelo E/R al modelo relacional, no nos costará mucho entender como están estructuradas las tablas y las relaciones entre ellas de nuestra base de datos.

**Vamos pues a aprender a realizar el diseño físico de bases de datos utilizando asistentes, herramientas gráficas y el lenguaje de definición de datos del SGBD.**

En la unidad anterior has visto cómo realizar el diseño conceptual y lógico de una base de datos relacional, esto es:

- ✓ Diseño conceptual mediante diagramas Entidad/Relación.
- ✓ Diseño lógico, transformando el diagrama E/R al modelo o esquema Relacional.
- ✓ Normalización del modelo lógico relacional, eliminando posibles inconsistencias del modelo obtenido anteriormente.

En esta unidad, verás cómo **obtener, a partir del modelo lógico, el diseño físico de una base de datos relacional (BDR)** e implantarla en un sistema gestor de bases de datos relacional (**SGBDR**), utilizando el lenguaje **SQL**, en particular el sublenguaje **DDL** (Lenguaje de Descripción o Definición de Datos, en inglés *Data Definition Language*).

Para ello vamos a necesitar tener instalado un **SGBDR**, de manera que comenzaremos instalando uno y dando los primeros en él.

# 1.- SGBD para implementar bases de datos.

## Caso práctico



Jonny Goldstein (CC BY)

Vindio y Noiba tienen que decidir que **SGBD** incorpora las características específicas que mejor se adaptan al proyecto que están llevando a cabo. En el mercado hay numerosos **SGBD** que incorporan utilidades para crear y manejar las bases de datos. Juan les propone que se documenten bien antes de tomar una decisión.

En la actualidad existen gran cantidad de **SGBD Relacionales o SGBDR**, entre los más extendidos destacan:

- ✓ SQL Server
- ✓ ORACLE
- ✓ DB2 de IBM
- ✓ MySQL
- ✓ MariaDB (fork de MySQL, es decir, una ramificación)
- ✓ PostgreSQL
- ✓ SQLite

Para algunos de ellos existen licencias que permiten el uso gratuito de la aplicación. En particular, MySQL se ha convertido en uno de los **SGBDR** más utilizados en la actualidad, sobre todo en servidores de páginas web, y además dispone de una versión con licencia **GPL General Public License**.

Por esta, y por otras razones que veremos a continuación, nos hemos decantado por el **SGBD MySQL** para usarlo en las prácticas de este módulo.

## Debes conocer

El Sistema Gestor de Bases de Datos MySQL es multiusuario y basado en arquitectura cliente/servidor.

Entre las características de MySQL podemos destacar las siguientes:

- ✓ Rapidez.
- ✓ Robustez.
- ✓ Facilidad de uso.
- ✓ Estabilidad.
- ✓ Ofrece un conjunto de funciones predefinidas muy amplio.

Las versiones iniciales de MySQL trabajaban bajo los sistemas Unix y Solaris. En la actualidad MySQL está disponible para una gran variedad de plataformas o sistemas, entre ellos:

- ✓ UNIX.
- ✓ Linux.
- ✓ OS2.
- ✓ Solaris.
- ✓ SunOS.
- ✓ Windows.
- ✓ macOS

Se estima que hay más de 6 millones de servidores MySQL instalados en el mundo, lo que significa, aproximadamente, el 20% del mercado.

En la versión actual, MySQL 8.0, ofrece nuevas mejoras para la administración del SGBD así como para el desarrollo de aplicaciones web, aplicaciones móviles, y aplicaciones integradas en la nube (Cloud / SaaS / PaaS / DBaaS). Además, permite trabajar tanto con tablas relacionales a través del lenguaje **SQL** como con colecciones de documentos **JSON**, como una base de datos **NoSQL**, lo cual es una opción muy interesante para la futura escalabilidad de la base de datos y las aplicaciones que hagan uso de ella.

Entre sus clientes destacan: PayPal, Alibaba, Cisco, Booking, Netflix, Google, YouTube, o Sony.

MySQL fue inicialmente desarrollado por **MySQL AB**, empresa fundada por David Axmark, Allan Larsson y Michael Widenius. MySQL AB fue adquirida por Sun Microsystems en 2008, y ésta a su vez fue comprada por **Oracle Corporation en 2010**.

**MySQL** se distribuye en varias versiones, una **Community**, distribuida bajo la Licencia pública general de **GNU**, versión 2, y varias versiones **Enterprise**, para las empresas que quieran incorporarlo en productos privativos.

En 2009, algunos desarrolladores (incluidos ciertos desarrolladores originales de MySQL), crearon un fork de MySQL denominado **MariaDB**.

Nosotros trabajaremos con el **SGBD MySQL** de manera que tendremos a **MySQL como servidor**, y como **cliente usaremos Workbench**, una herramienta gráfica para trabajar con MySQL.

## Recomendación

Te puedes descargar el [manual oficial de MySQL 8](#) en formato PDF. Aunque está en inglés, en él podrás encontrar toda la información referente a esta aplicación. Recuerda que es muy importante ser capaz de leer manuales de aplicaciones en inglés, ya que **la mayoría de las aplicaciones ofrecen siempre manuales en inglés**, entre otros idiomas.

## Para saber más

Para tener una visión de todas las posibilidades de MySQL es importante que visites su página oficial y naveges por ella. Desde el siguiente enlace puedes acceder.

[Web Oficial de MySQL](#)

## Autoevaluación

**De la siguiente lista de aplicaciones señala la que consideres que NO corresponde a un SGBD.**

- SQL Server.
- SQLite.
- MySQL Workbench.
- PostgreSQL.

Incorrecto. Es un SGBD propietario de Microsoft.

No es correcto. Es un SGBD relacional muy usado en la programación Móvil, como Android..

Correcto. No se trata de un SGBD, es una herramienta gráfica para modelar, crear y administrar bases de datos.

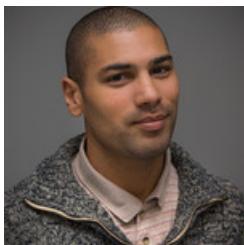
Falso. Es, junto a MySQL, uno de los SGBD más comunes en la red.

## Solución

1. Incorrecto
2. Incorrecto
3. Opción correcta
4. Incorrecto

## 2.- Instalación de MySQL.

### Caso práctico



[Alain Bachellier \(CC BY-NC-SA\)](#)

Vindio y Noiba han estado documentándose sobre diferentes SGBD y han decidido que utilizarán MySQL, puesto que entre otras muchas buenas características, MySQL permite trabajar tanto con tablas relacionales a través del lenguaje SQL como con colecciones de documentos JSON, como una base de datos NoSQL, lo cual es una opción muy interesante para la futura escalabilidad de la base de datos y las aplicaciones que hagan uso de ella. Entre las diferentes versiones que ofrece MySQL van a usar la versión Community Edition, que cuenta con licencia GPL. Ahora será necesario elegir un modo de instalación, y localizar los pasos necesarios que se deben seguir para realizar una correcta instalación.

#### ¿Cómo instalar MySQL?

Para hacer uso de un SGBD MySQL podemos instalarlo en nuestro equipo local o bien podemos usar un servicio de MySQL en la nube (MySQL Cloud Service).

Nosotros vamos a realizar una instalación en modo local, esto es, en nuestro equipo informático.

Para **instalar el sistema de base de datos MySQL** en nuestro equipo tenemos dos opciones:

- ✓ Instalar solo el Servidor de base de datos MySQL y la herramienta o cliente gráfico MySQL Workbench. Para ello, podemos descargar estas aplicaciones desde la página Web del fabricante: [Web oficial de MySQL](#).
- ✓ Instalar una aplicación o paquete de software que nos permita **configurar un sitio Web completo** de una forma muy sencilla. Esos programas se denominan XAMP, sustituyendo la X por W cuando lo instalamos en Windows, o por L cuando el sistema operativo es Linux. Una aplicación XAMP realiza la instalación de una vez el servidor Web Apache, la base de datos MySQL, el lenguaje PHP y una herramienta gráfica para administrar la base de datos vía Web denominada PhpMyAdmin. Existen diferentes aplicaciones de este tipo, algunas de ellas son XXAMP, que es multiplataforma y WampServer para sistemas Windows.

En principio nosotros realizaremos la **instalación de MySQL y Workbench**, y dejaremos para más adelante la instalación de Wampserver.

La versión gratuita de MySQL es MySQL Community Server (GPL), mientras que existe una versión comercial llamada MySQL Enterprise. Las ventajas de esta última respecto a la primera reside en la posibilidad de soporte técnico, actualización automática de versiones, monitorización del sistema para detectar posibles vulnerabilidades del mismo, etc.

Realizaremos la instalación sobre el sistema operativo Windows. Para cualquier otro sistema, la instalación sería muy similar y los pasos a seguir se encuentran en la propia documentación de la página oficial de MySQL.

La última versión recomendada se etiqueta como GA, Current Generally Available Release. En el momento de escribir este documento la versión MySQL Community Edition es la 8.0.27.

En el apartado [Installing MySQL on Microsoft Windows](#) del manual oficial de MySQL, se explica detalladamente el procedimiento a seguir para realizar una correcta instalación de la aplicación.

A continuación vamos a describir, de forma general, los pasos para realizar la instalación de una de las últimas versiones de MySQL, MySQL Community Edition 8:

Accedemos a la página de descarga del [instalador de MySQL Community Server para Windows](#), y descargamos el instalador de la página oficial de MySQL (el fichero mysql-installer-community-8.0.27.1.msi en el caso de la versión 8.0.27). Este instalador incluye la instalación de la mayoría de los productos necesarios para trabajar con MySQL, por ejemplo **Workbench**, **conectores**, **bases de datos de ejemplo**, etc. Además, puede realizar instalaciones en plataformas de 32 y 64 bits.

General Availability (GA) Releases Archives 🔍

## MySQL Community Server 8.0.27

Select Operating System: Microsoft Windows

Looking for previous GA versions?

Recommended Download:

**MySQL Installer for Windows**

All MySQL Products. For All Windows Platforms. In One Package.

Starting with MySQL 5.6 the MySQL Installer package replaces the standalone MSI packages.

Windows (x86, 32 & 64-bit), MySQL Installer MSI

Go to Download Page >

Other Downloads:

Windows (x86, 64-bit), ZIP Archive (mysql-8.0.27-winx64.zip)	8.0.27	209.4M	<a href="#">Download</a>
Windows (x86, 64-bit), ZIP Archive Debug Binaries & Test Suite (mysql-8.0.27-winx64-debug-test.zip)	8.0.27	509.6M	<a href="#">Download</a>

We suggest that you use the [MD5 checksums](#) and [GnuPG signatures](#) to verify the integrity of the packages you download.

 © 2021, Oracle Corporation and/or its affiliates

[Legal Policies](#) | [Your Privacy Rights](#) | [Terms of Use](#) | [Trademark Policy](#) | [Contributor Agreement](#) | [Preferencias sobre cookies](#)

[Oracle MySQL](#). Descarga del archivo de instalación de MySQL (Elaboración propia)

## Paso 2

Antes de poder descargar el instalador se solicita registro, pero podemos saltarlo haciendo clic sobre el enlace de la parte inferior que indica '**No thanks, just start my download!**'.

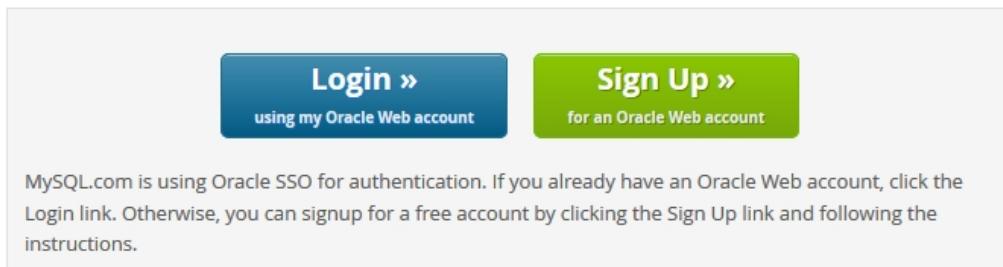
Descargaremos a nuestro equipo local un archivo similar a este: (Fichero **mysql-installer-community-8.0.27.1.msi** en el caso de la versión **8.0.27**.)

## ④ MySQL Community Downloads

[Login Now](#) or [Sign Up](#) for a free account.

An Oracle Web Account provides you with the following advantages:

- Fast access to MySQL software downloads
- Download technical White Papers and Presentations
- Post messages in the MySQL Discussion Forums
- Report and track bugs in the MySQL bug system



[No thanks, just start my download.](#)



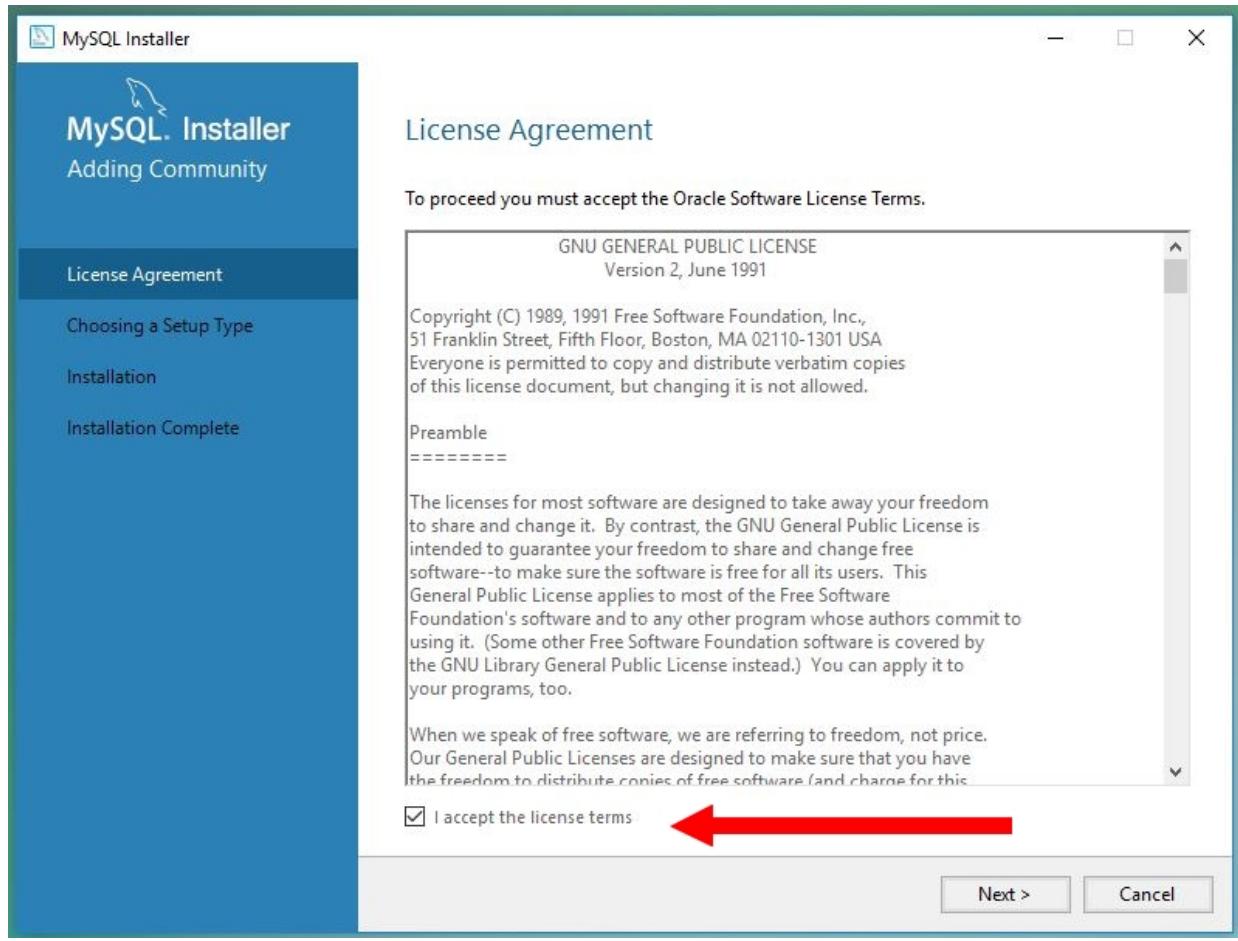
© 2020, Oracle Corporation and/or its affiliates

[Legal Policies](#) | [Your Privacy Rights](#) | [Terms of Use](#) | [Trademark Policy](#) | [Contributor Agreement](#) | [Cookie Preferences](#)

Oracle MySQL (Elaboración propia)

## Paso 3

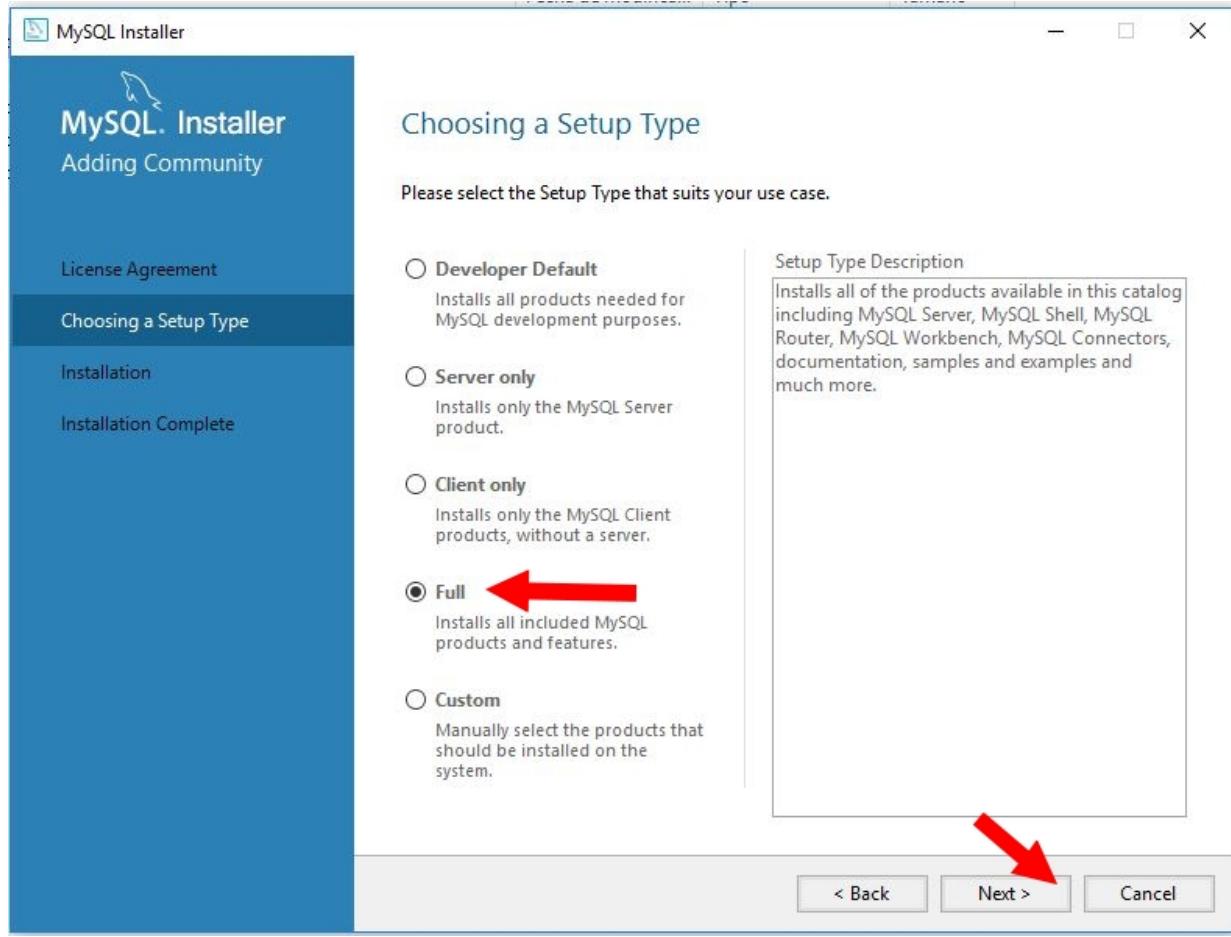
Una vez descargado debes ejecutar el instalador de MySQL. Al ejecutarlo aparece la pantalla en la que debemos **aceptar los términos de la licencia GPL**:



Oracle MySQL (Elaboración propia)

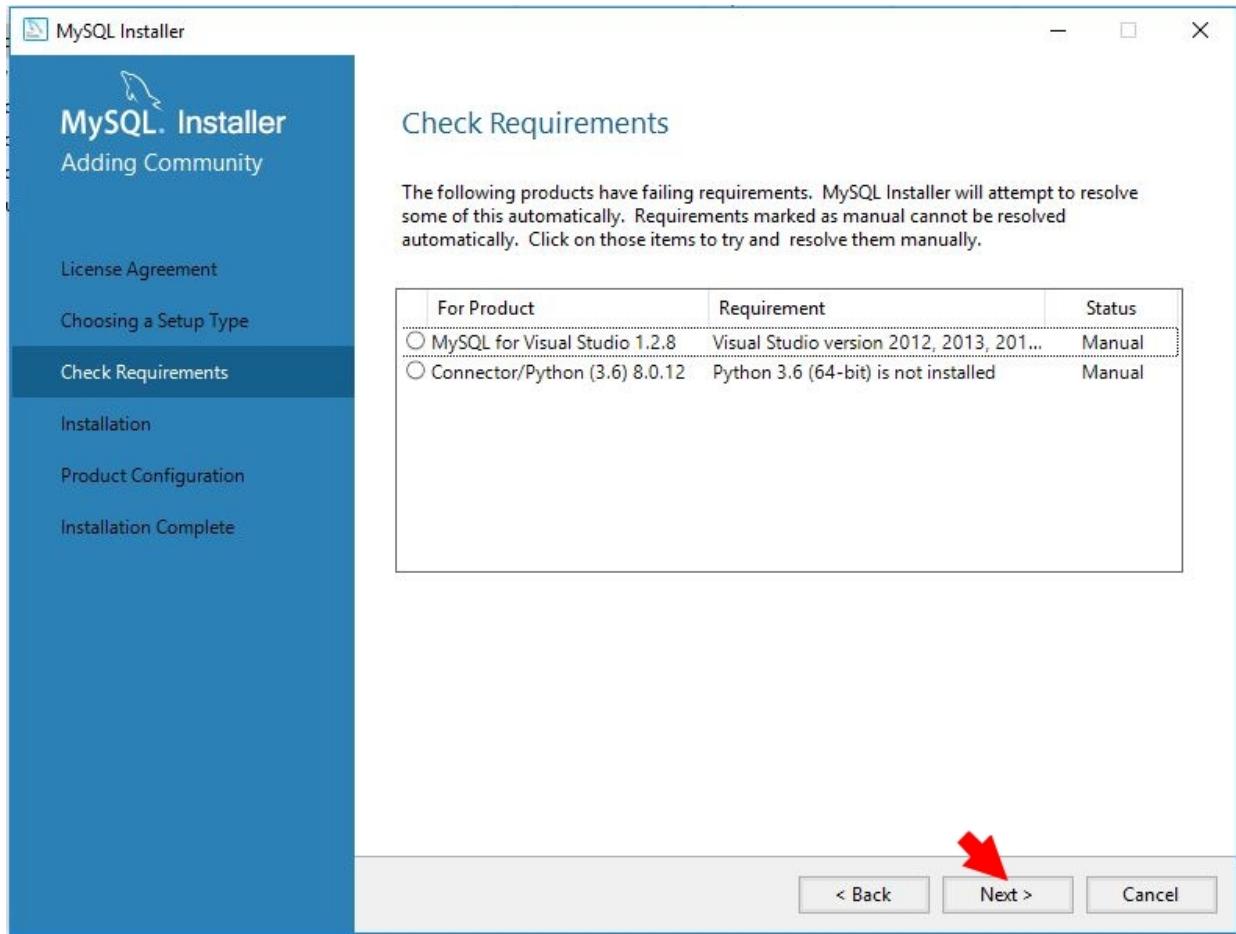
## Paso 4

A continuación, elegimos el **tipo de configuración de la instalación**. Si tenemos suficiente espacio en disco podemos elegir **Full**, completa. Si no queremos instalar todos los elementos elegiremos **Developer Default**, que instala los componentes necesarios para los desarrolladores de software. O bien la opción **Custom**, personalizada, que permite elegir los componentes a instalar.



## Paso 5

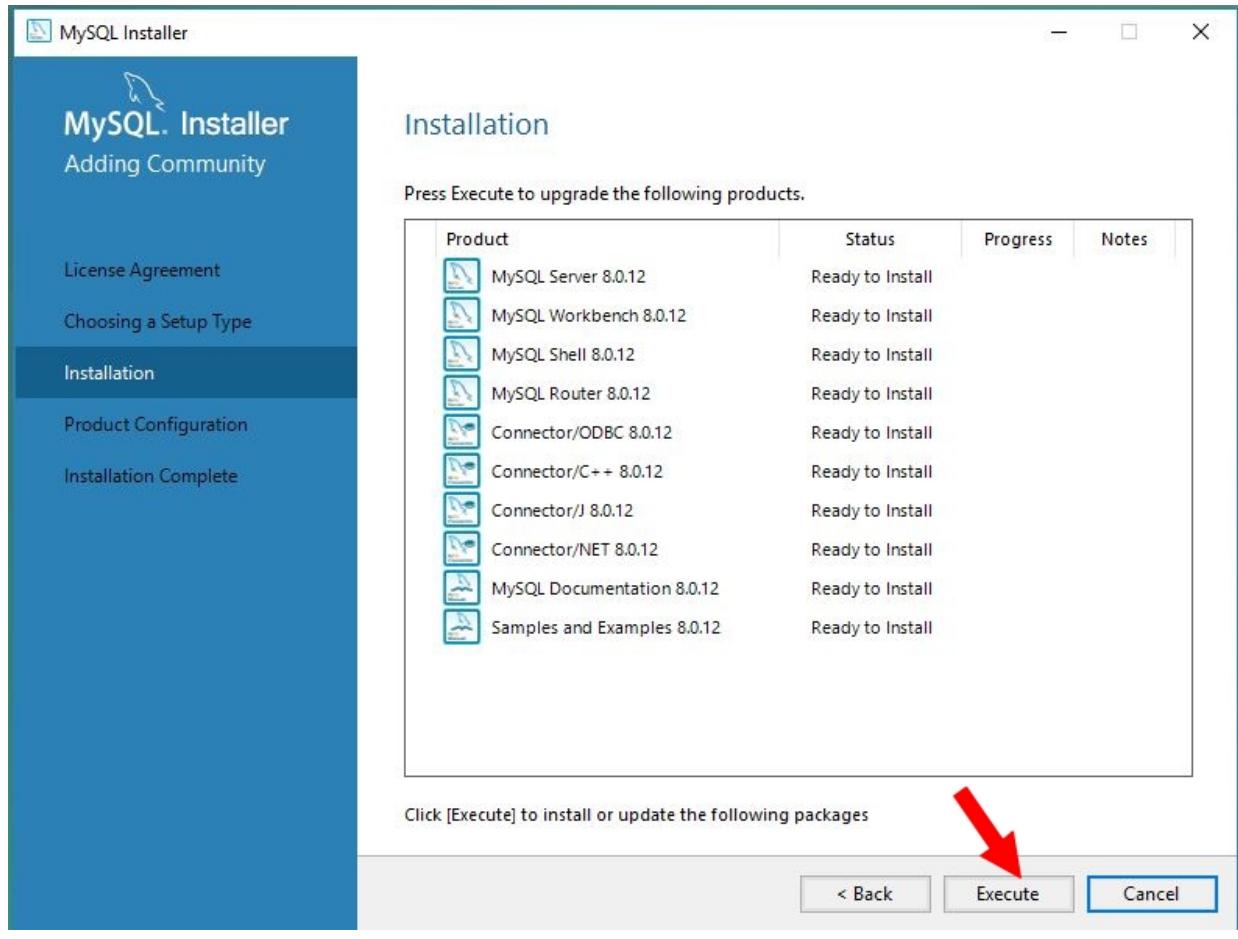
Se muestra una **lista de requerimientos para la instalación**. Algunos de ellos se instalarán automáticamente, otros tendremos que instalarlos de forma manual. En cualquier caso, podemos continuar la instalación sin instalar todos los requerimientos, aunque es posible que en el futuro se soliciten dichos complementos.



Oracle MySQL (Elaboración propia)

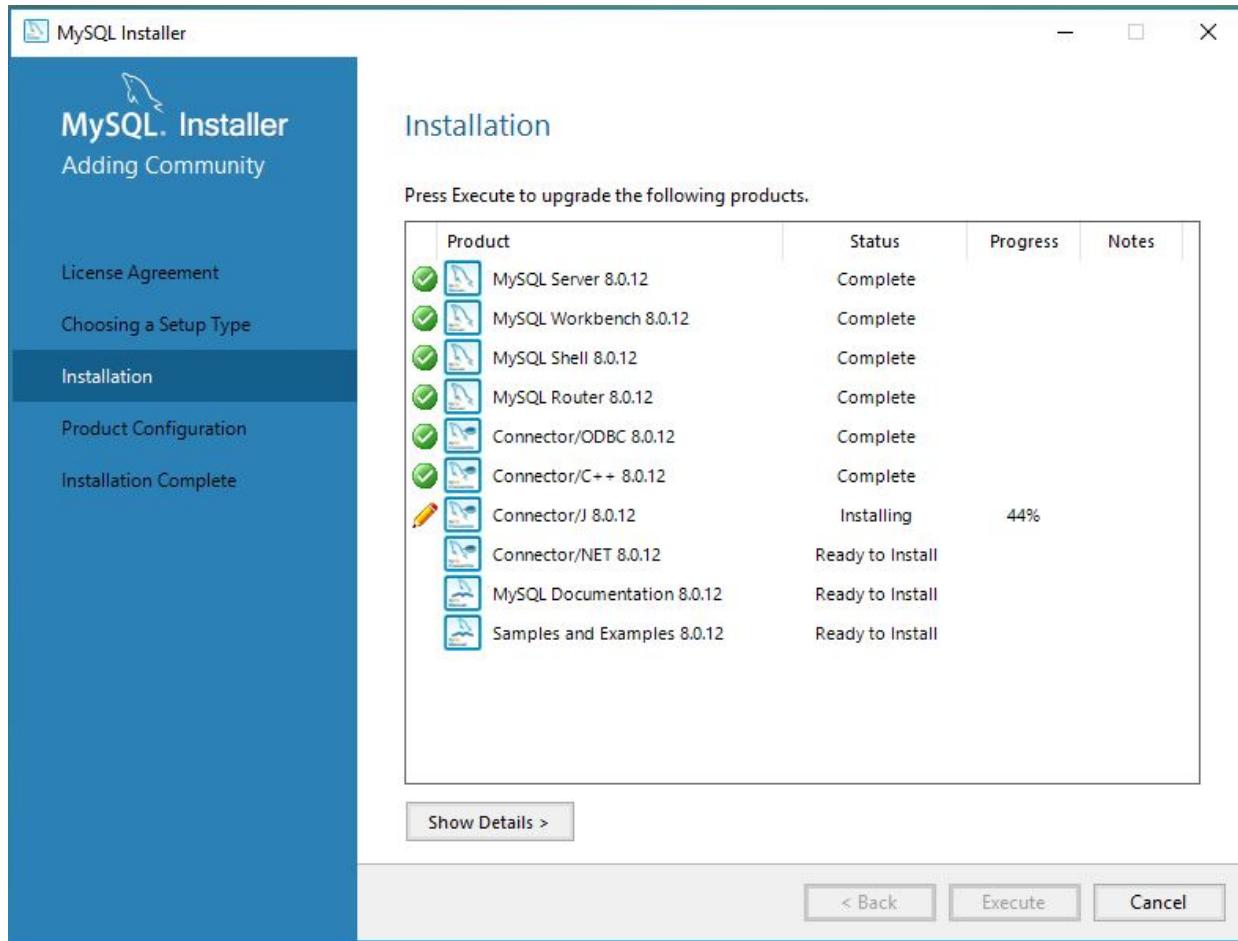
## Paso 6

En la siguiente pantalla, tras pulsar sobre el botón **Execute** comenzará la instalación de los distintos componentes.



## Paso 7

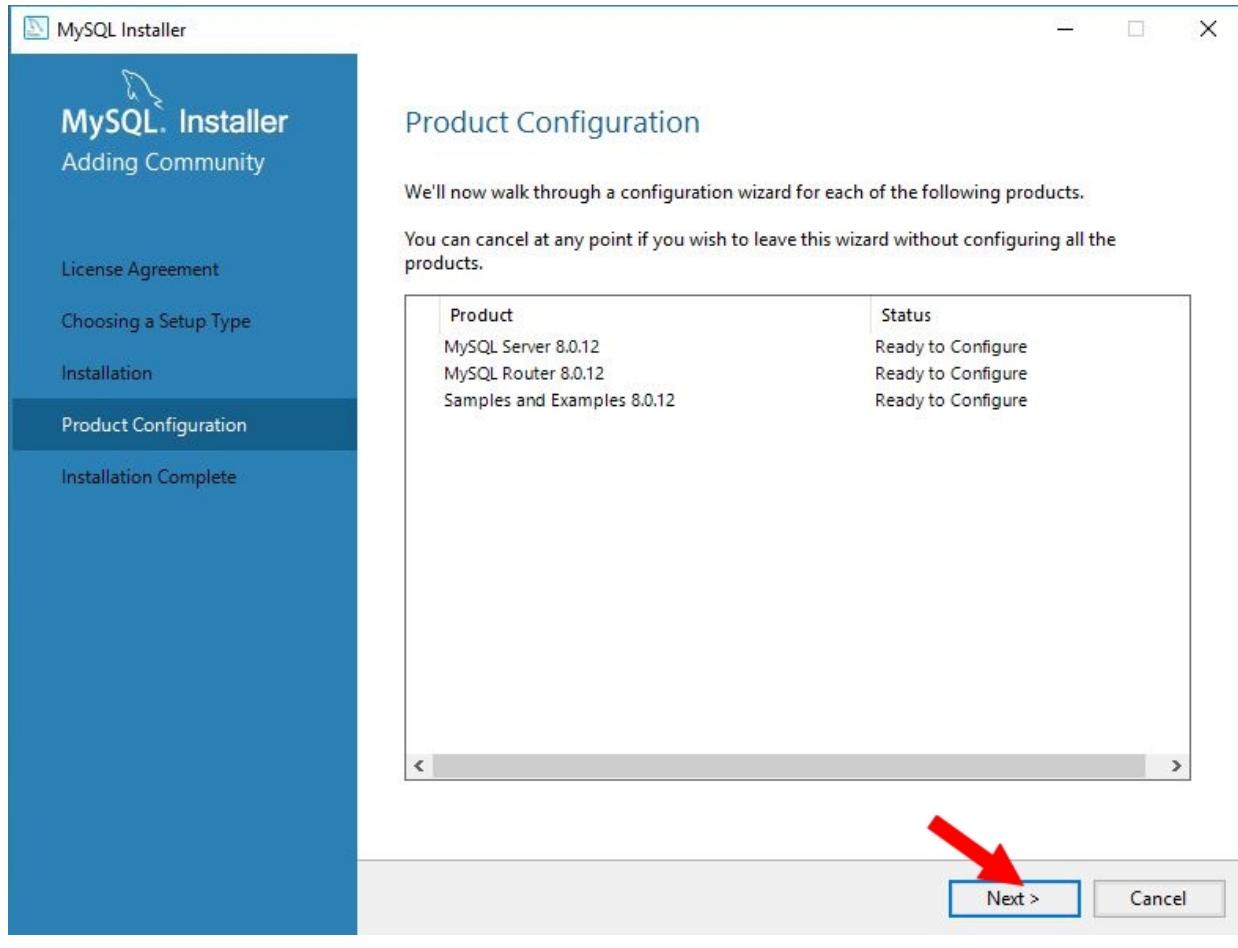
Una vez finalizada la instalación de componentes pulsaremos sobre el botón que permitirá continuar con la instalación.



Oracle MySQL (Elaboración propia)

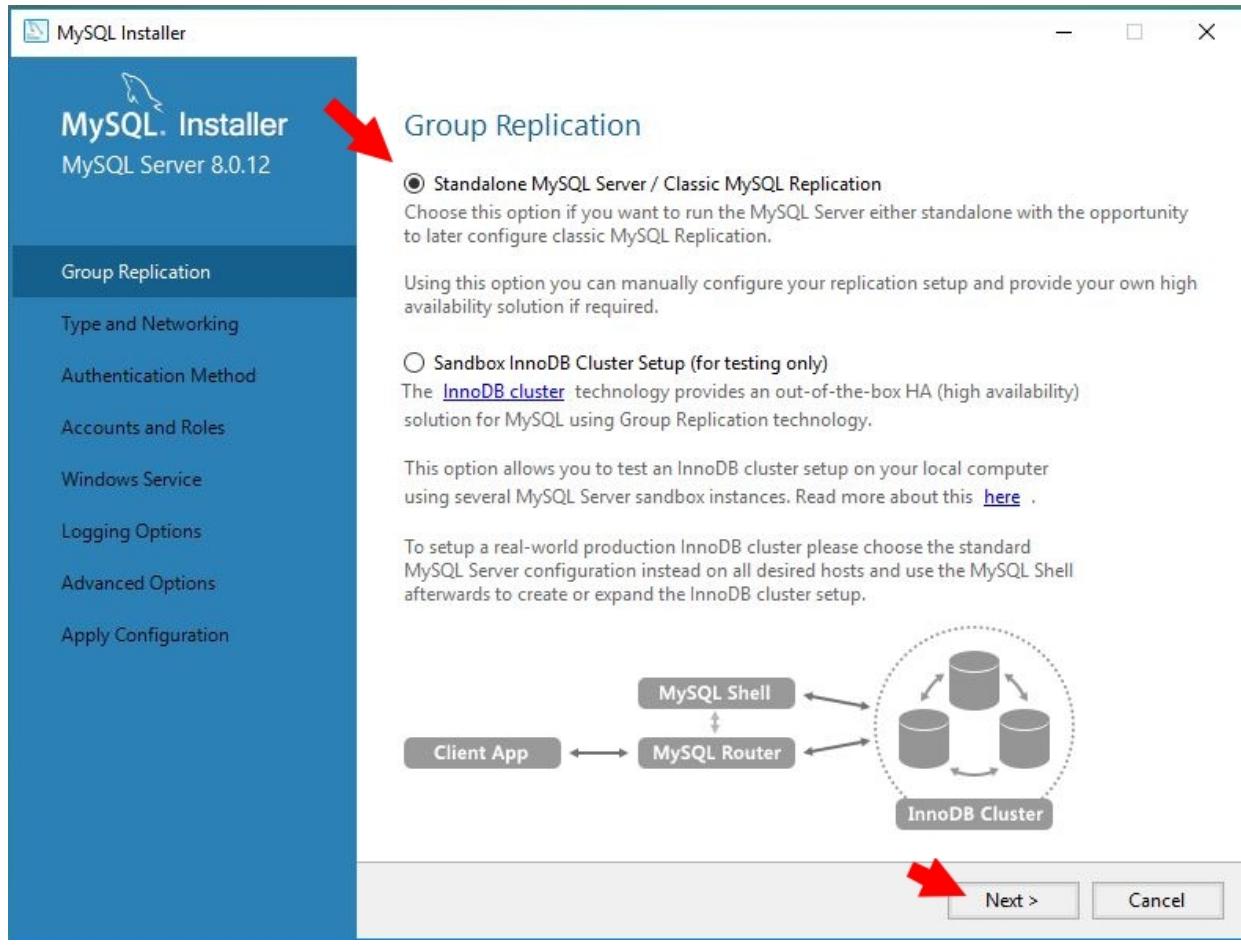
## Paso 8

A continuación se prosigue con la **configuración del servidor** y de otros elementos instalados.



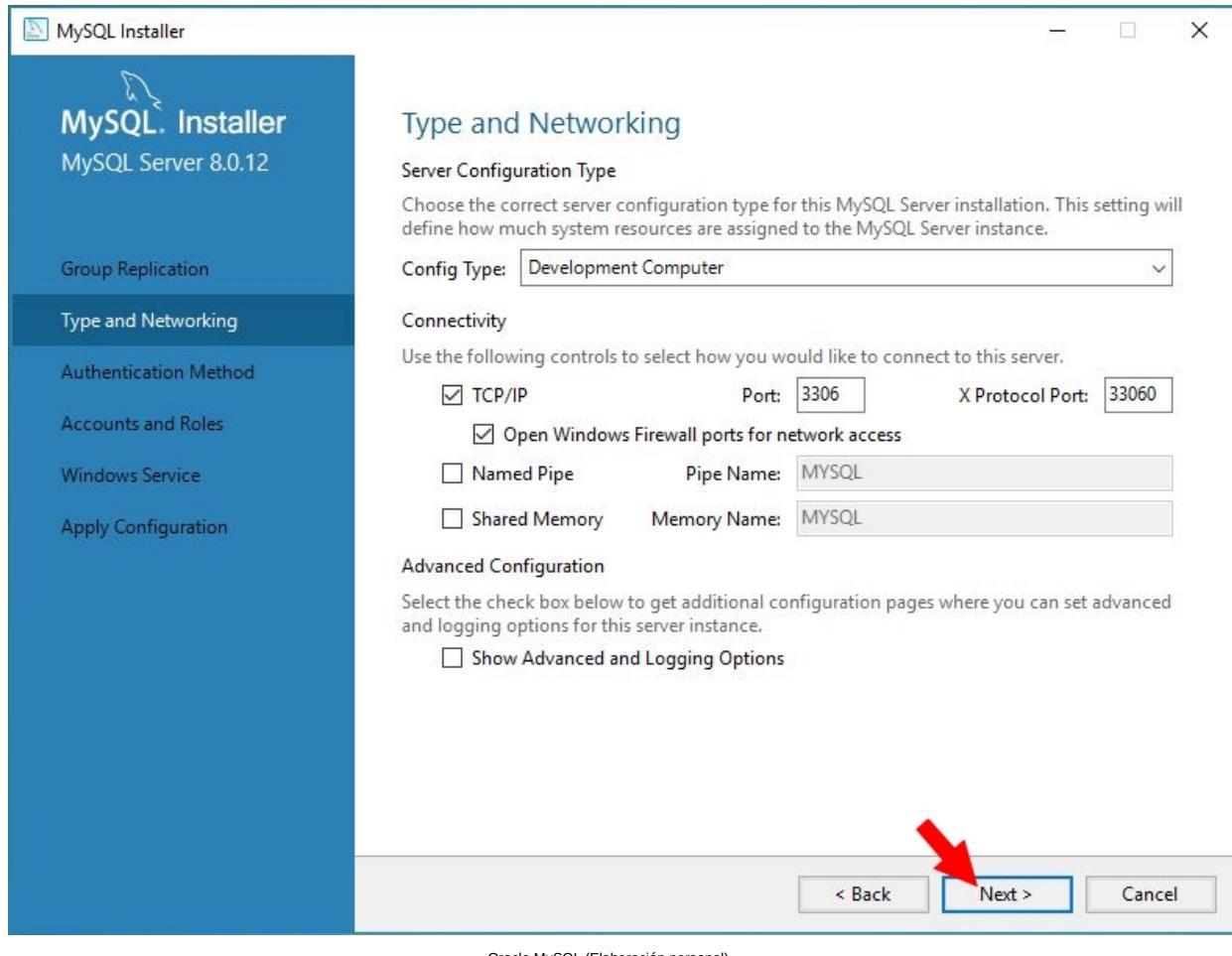
## Paso 9

Indicamos si el servidor formará parte de un clúster, actuará como único servidor, o bien se configurará como servidor de replicación clásico.



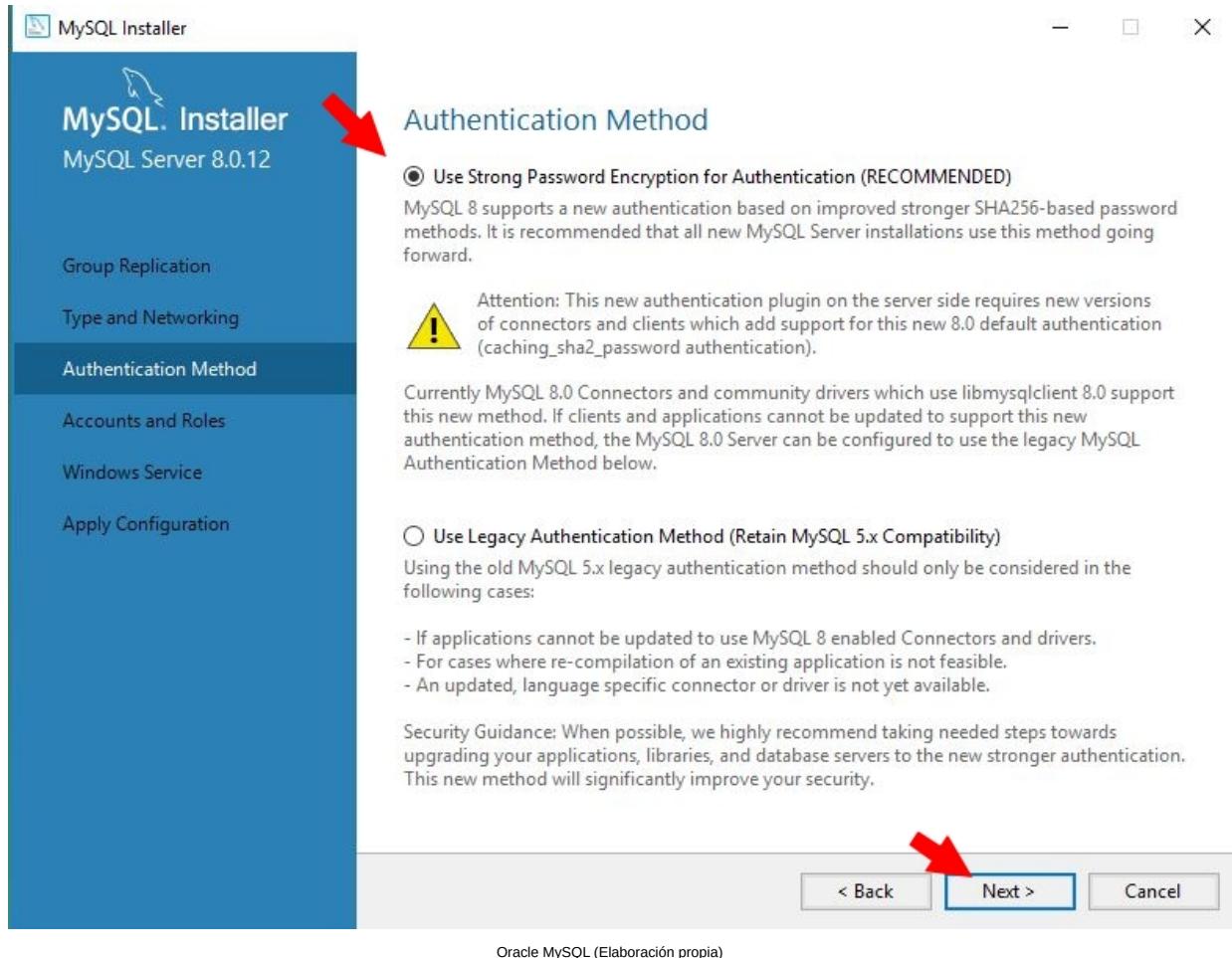
## Paso 10

Se considera que la máquina (ordenador) en la que se instala MySQL se utiliza para desarrollo de aplicaciones (*Development Machine*). El **puerto TCP/IP de escucha para el servicio** será, por defecto, el **3306**.



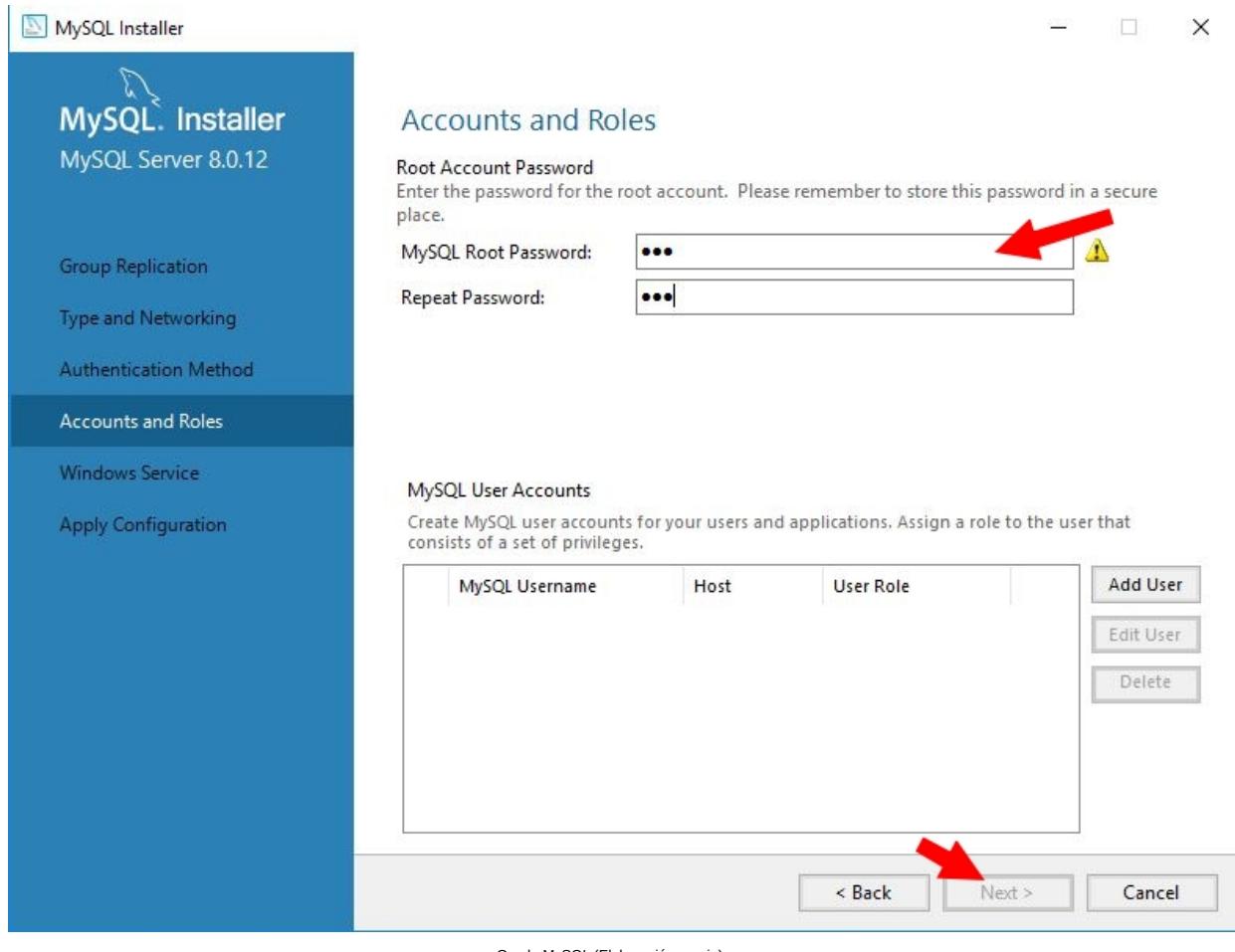
## Paso 11

Elegimos el **método de autenticación**. Elegimos el método de encriptación recomendado.



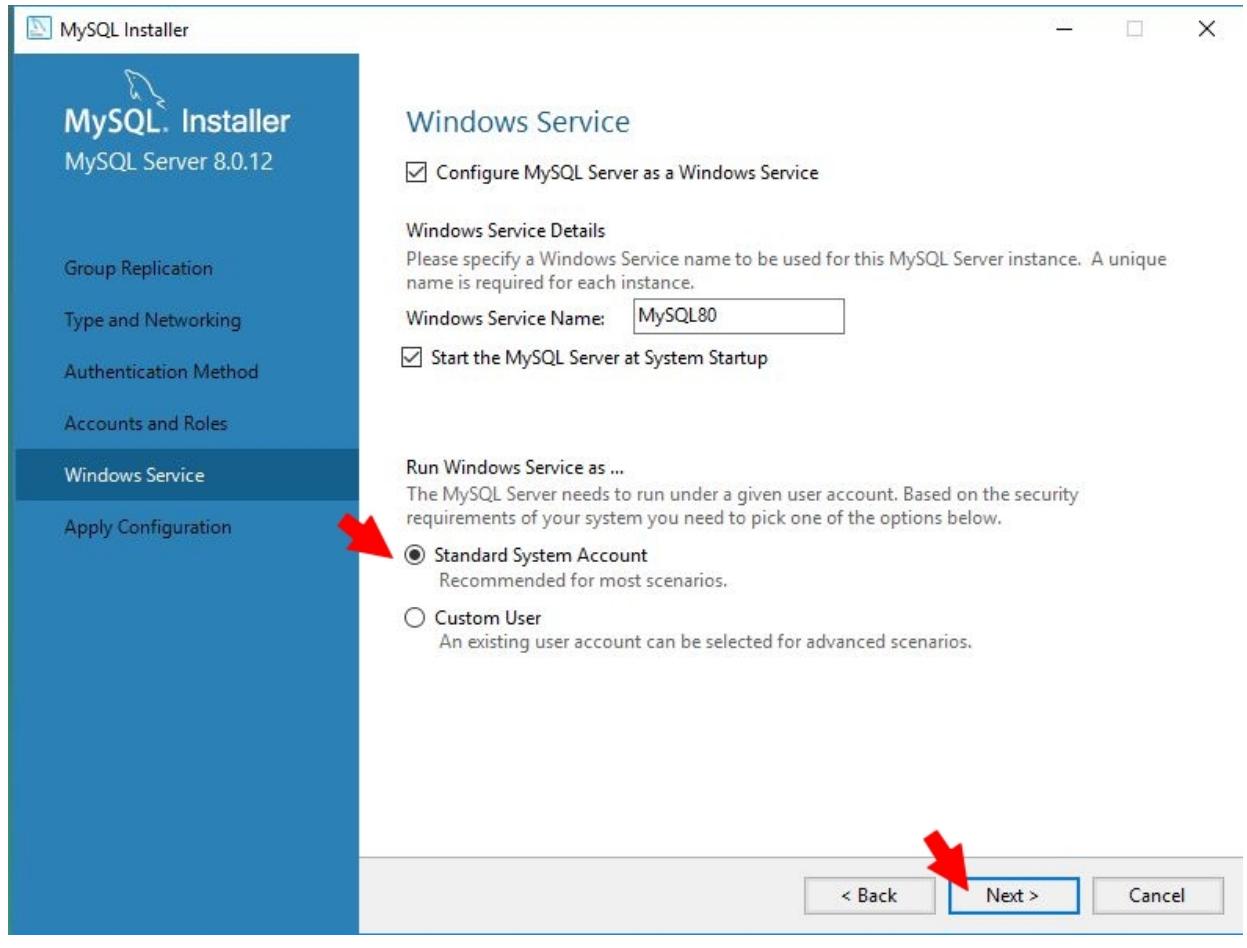
## Paso 12

Establecemos la **contraseña del usuario root**, que es el usuario administrador del sistema.



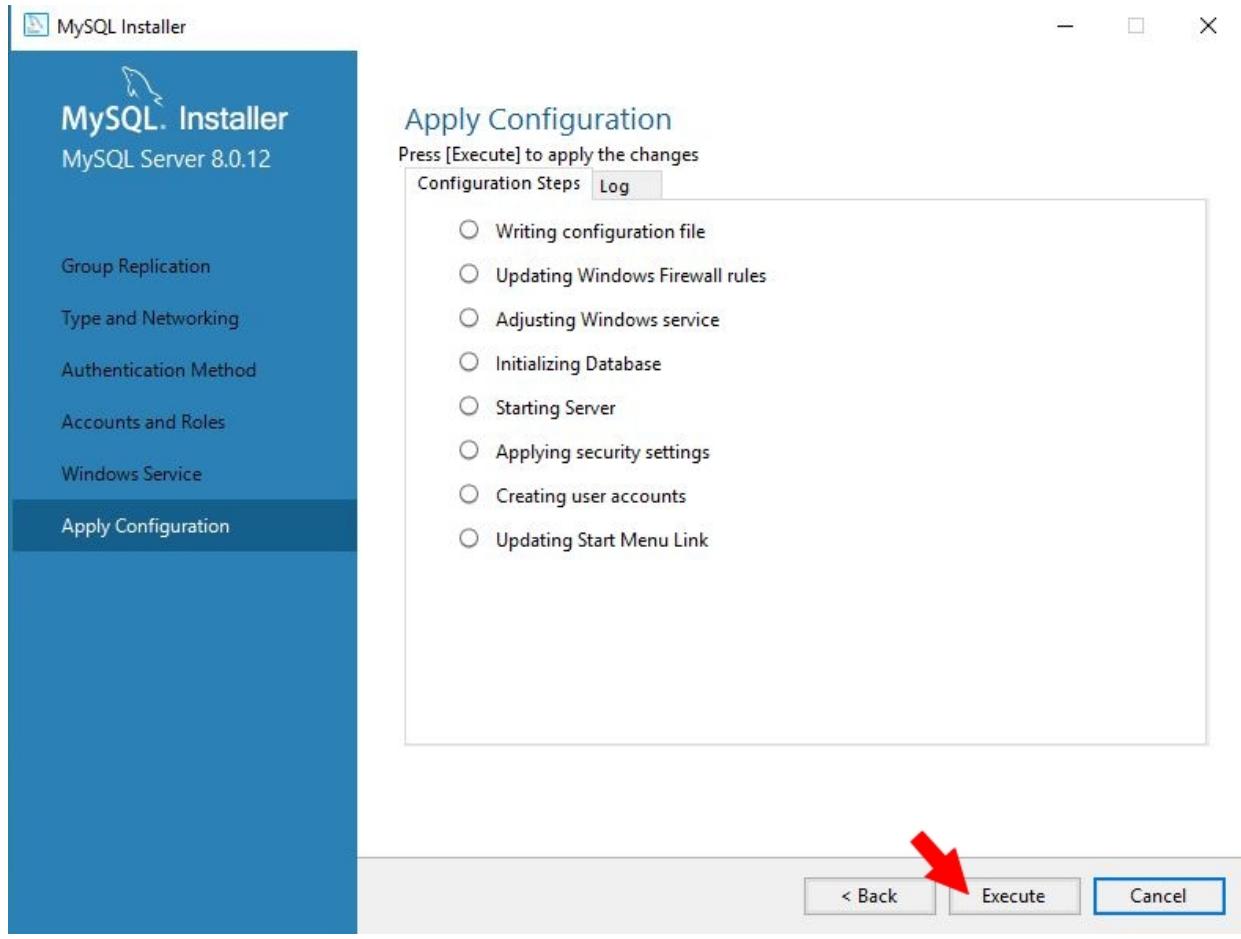
## Paso 13

Lo habitual es **configurar MySQL como un servicio más de Windows**. Se iniciará automáticamente en el inicio del sistema operativo.



## Paso 14

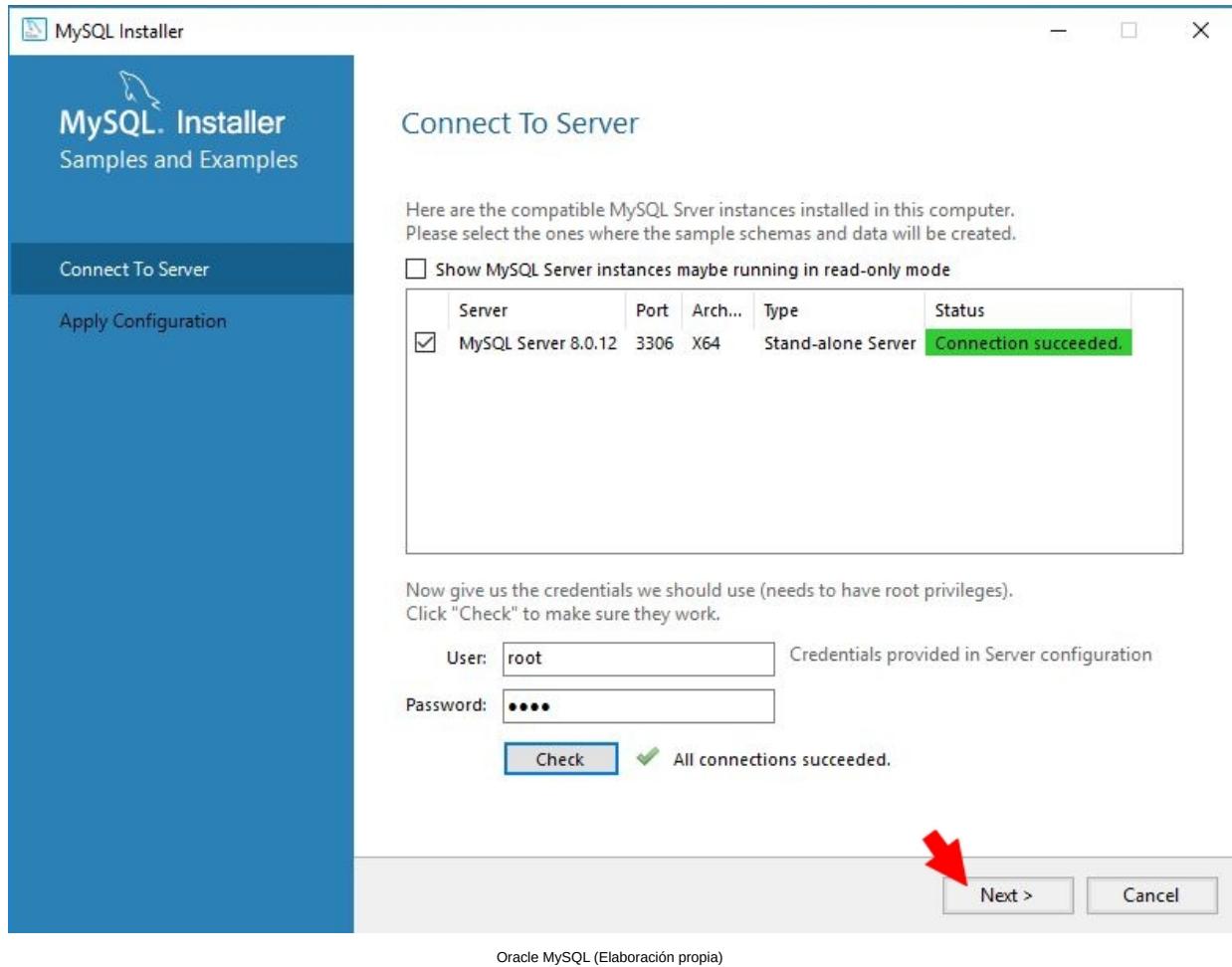
A continuación se **aplica la configuración** elegida.



Oracle MySQL (Elaboración propia)

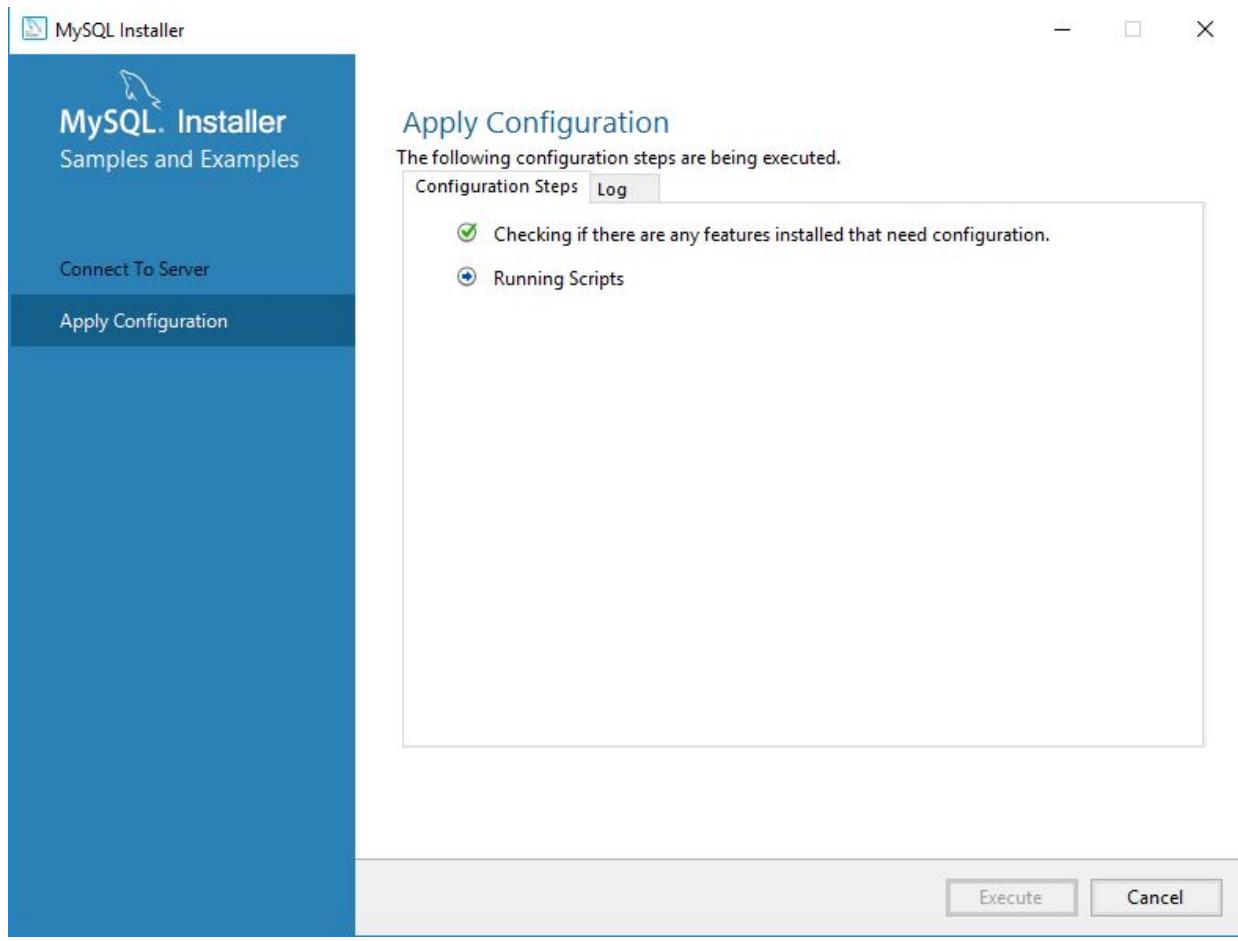
## Paso 15

Podemos probar la conexión de prueba de la conexión con el servidor, pulsando sobre **Check** y si no hay problemas, debe aparecer la confirmación exitosa. A continuación pulsamos el botón **Next**.



## Paso 16

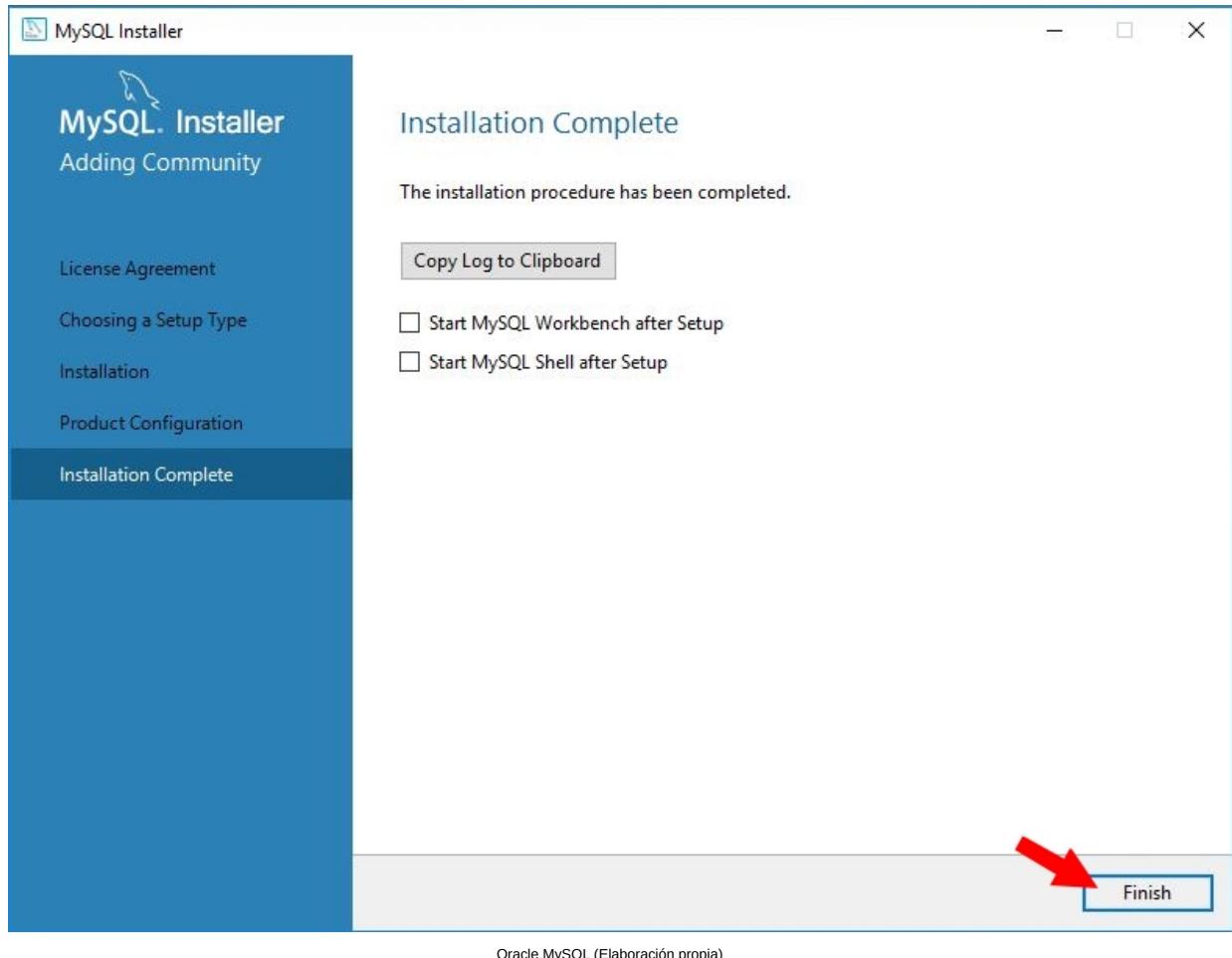
Dependiendo de la instalación, es posible que el instalador realice **configuraciones adicionales**.



Oracle MySQL (Elaboración propia)

## Paso 17

Y ya tenemos la **instalación completada**. Pulsamos sobre el botón **Finish** y ya tendremos el servidor MySQL instalado.



Oracle MySQL (Elaboración propia)

## Recomendación

Ahora debes realizar la instalación del servidor MySQL 8 en tu equipo personal siguiendo los pasos indicados arriba. Recuerda que también puedes recurrir al manual oficial de MySQL.

Si necesitas repasar los pasos de instalación, puedes consultar además los siguientes videotutoriales de instalación de MySQL.

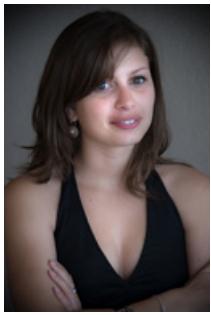
[Videotutorial de Instalacion de MySQL de forma personalizada o custom](#)

[Descargar MySQL de 64 bits en WINDOWS 10 2021. \(MAB55\)](#)

[Instalar MySQL 8.0.22 server en Windows 10. \(Solvetic\)](#)

## 3.- Primeros pasos con MySQL.

### Caso práctico



**Noiba** y **Vindio** han finalizado con la instalación de MySQL Server, pero ahora necesitan aprender los mecanismos de interacción con el servidor. Lo primero que deben hacer es localizar las aplicaciones que actúan de cliente del servidor, para poder conectar con la parte servidor de MySQL.

Noiba sabe que es posible acceder al servidor desde un **cliente de línea de comandos** o bien a través de un cliente gráfico, como **Workbench**, **PhpMyAdmin**, **Navicat**, **HeidiSQL**, etc.

Alain Bachelier ([CC BY-NC-SA](#))

En los siguientes apartados vamos a detallar los primeros pasos que debemos dar con MySQL.

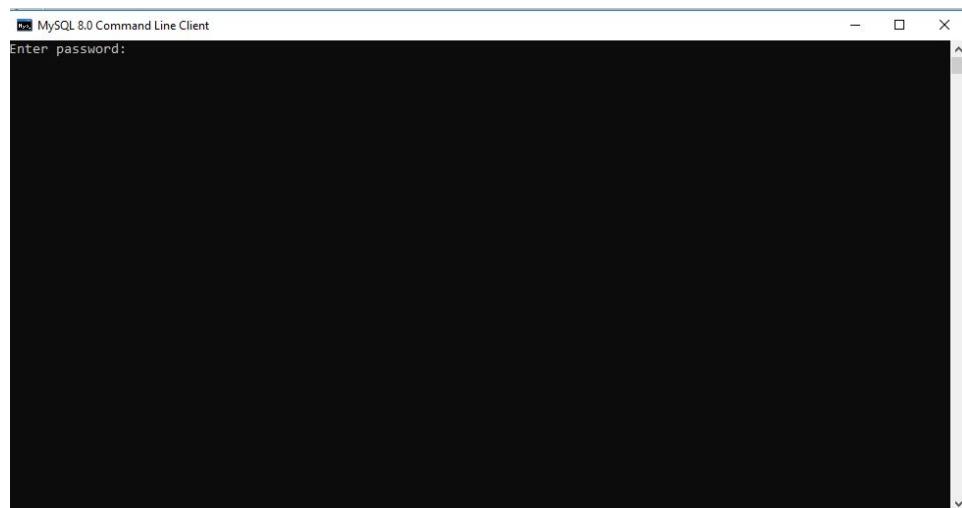
¡Vayamos a ello!

### 3.1.- El cliente de línea de comandos y primeros comandos.



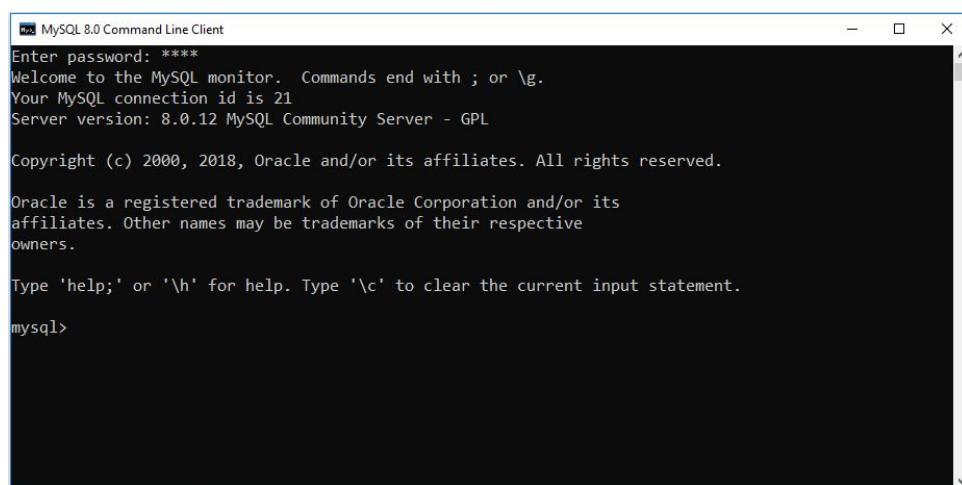
El programa **MySQL Command Line Client**, que aparece en el grupo de programas MySQL, permitirá acceder al servidor como usuario root (administrador) desde la línea de comandos. La siguiente imagen muestra como se solicita la contraseña para el administrador o usuario `<strong>root</strong>`.

[Everaldo Coelho and YellowIcon \(GNU/GPL\)](#)



Oracle MySQL (Elaboración propia)

Después de introducir la contraseña del usuario **root** aparecerá el **prompt** de **MySQL**, que es `mysql>` como se puede ver en la siguiente imagen, lo que indica que el servidor MySQL está listo para recibir consultas:



Oracle MySQL (Elaboración propia)

También es posible ejecutar este programa desde el **símbolo del sistema operativo** o **shell del sistema operativo**. Si el usuario invitado está activado bastará con escribir el comando `<strong>mysql </strong>` en el prompt del sistema operativo. En otro caso habrá que especificar un nombre de usuario y una contraseña. Más adelante veremos cómo hacerlo. Si la ruta en la que está almacenado el ejecutable (`mysql.exe`) no está incluida en la variable del sistema PATH, entonces tendremos que situarnos en el directorio en el que está almacenado el programa. Posiblemente ese directorio sea `C:\Program Files\MySQL\MySQL Server 8.0\bin`

## Debes conocer

En informática, el **shell** o **intérprete de órdenes** o **intérprete de comandos** es el programa informático que provee una interfaz de usuario para acceder a los servicios del sistema operativo.

Dependiendo del tipo de interfaz que empleen, los shells pueden ser:

- ✓ De líneas texto (**CLI**, Command-Line Interface, interfaz de línea de comandos),
- ✓ Gráficos (**GUI**, Graphical User Interface, interfaz gráfica de usuario).

Los shells están diseñados para facilitar la forma en que se invocan o ejecutan los distintos programas disponibles en la computadora.

En algunas ocasiones usaremos el **cliente de línea de comandos** (programa `mysql.exe`) para conectarnos al servidor MySQL.

Vamos a ver, a continuación, cómo realizar la conexión y desconexión del servidor en modo comando, así como algunos ejemplos de comandos básicos y sintaxis.

Conexión al servidor MySQL con el comando o cliente en modo texto `mysql.exe`

Desconexión con el comando `exit` o `quit`

Ejecutando algunas consultas y comandos. Sintaxis

Mostrar las bases de datos del servidor: `SHOW DATABASES;`

Poner en uso una base de datos: `USE nombrebasedatos;`

Ver las tablas de una base de datos: `SHOW TABLES;`

Para ver la estructura o descripción de una tabla: `DESCRIBE nombreTabla;`

## 3.2.- Directorios de MySQL.

Tras la instalación de MySQL se crean una serie de directorios (carpetas) en el sistema de archivos que debes conocer. Independientemente del sistema operativo en el que hayas instalado MySQL, la ruta física de esos directorios se guarda en las variables denominadas `datadir` y `basedir`:

- ✓ <span lang="en"><strong>`datadir`</strong></span>. es el directorio de los datos, apunta al directorio físico `\data` estructurado en subcarpetas o subdirectorios.

En el directorio físico \<strong></strong><span lang="en"><strong>`data`</strong></span> se creará un subdirectorio para cada base de datos creada con MySQL.

Tras la instalación se crean automáticamente algunas bases de datos, por ejemplo la **base de datos del sistema mysql**. Se trata de una base de datos de uso propio del servidor; en ella se guardan los usuarios, passwords, permisos, etc. Otra **base de datos** importante es `information_schema`, en ella se almacenan metadatos.

- ✓ <strong><span lang="en">`basedir`</strong></span>: es el directorio base y apunta al directorio físico de la instalación de MySQL. El directorio físico, estructurado en subdirectorios contiene, entre otros:

El directorio<strong> `\bin`</strong>: en este subdirectorio se almacenan los ejecutables o comando de línea asociados a MySQL. Entre ellos destacamos:

<strong>`mysqld`</strong>: es el programa servidor de mysql.

<strong>`mysql`</strong>: es el programa cliente de mysql.

<strong>`mysqladmin`</strong>: aplicación usada para realizar diferentes tareas administrativas sin necesidad de conectarnos al servidor.

<strong>`mysqldump`</strong>, <strong>`mysqlimport`</strong>, <strong>`mysqlhotcopy`</strong>,...: son aplicaciones usadas con diferentes propósitos como realización de copias de seguridad, etc.

El directorio **include**: contiene las distintas librerías de funciones utilizadas por el programa MySQL. Todas están escritas en lenguaje C.

En Windows <span lang="en"><strong>`basedir`</strong></span> apunta, por defecto, a la siguiente ruta C:\Archivos de programa\MySQL\MySQL Server x.x. y <span lang="en"><strong>`datadir`</strong></span> a esta otra ruta C:\ProgramData\MySQL\MySQL Server x.x\data.

Un archivo muy importante es el archivo de configuración de MySQL que en Windows se llama **my.ini**, y en Linux <strong>`my.cnf`</strong>, este archivo se encuentra en una de esas carpetas.

¿Cómo podemos saber la ruta física de esos directorios para cualquier sistema operativo?

Esas rutas físicas las podemos obtener siempre, para cualquier sistema operativo en el que se haya instalado MySQL, ejecutando los siguientes comandos desde el shell de mysql:

- ✓ `SHOW VARIABLES LIKE 'basedir';`
- ✓ `SHOW VARIABLES LIKE 'datadir';`

En el caso de la instalación en un sistema Windows:

```
mysql> show variables like 'basedir';
+-----+-----+
| Variable_name | Value           |
+-----+-----+
| basedir      | C:\Program Files\MySQL\MySQL Server 8.0\ |
+-----+-----+
1 row in set (0.01 sec)

mysql> show variables like 'datadir';
+-----+-----+
| Variable_name | Value           |
+-----+-----+
| datadir      | C:\ProgramData\MySQL\MySQL Server 8.0\Data\ |
+-----+-----+
1 row in set (0.00 sec)

mysql>
```

Oracle MySQL (Elaboración propia)



Everaldo Coelho and  
YellowIcon (GNU/GPL)

# Autoevaluación

Indica si la siguiente afirmación es verdadera o falsa.

Tras la instalación de MySQL se creará un subdirectorio denominado de forma genérica **datadir**, apunta al directorio físico ..\data en el que se creará un subdirectorio para cada base de datos creada con MySQL.

- Verdadero  Falso

**Verdadero**

Recuerda que para obtener la ruta física de ese directorio se puede ejecutar la siguiente orden:  
mysql> SHOW VARIABLES LIKE 'datadir';

## Recomendación

Localiza y accede al contenido de los directorios que se han creado al instalar el servidor MySQL. Intenta localizar en tu sistema el fichero de configuración de MySQL.

### 3.3.- El cliente gráfico Workbench y primeras sentencias.

Podemos conectarnos a MySQL mediante el cliente gráfico Workbench. Esta herramienta permite además realizar el modelado o diseño de una base de datos de forma gráfica.

Realmente **Workbench nos va a permitir realizar diferentes tareas**:

- ✓ Crear el modelo o diseño de una base de datos de forma gráfica. (No es imprescindible la conexión al servidor MySQL)
- ✓ Lanzar sentencias SQL contra el servidor de bases de datos.
- ✓ Trabajar en modo gráfico con las bases de datos almacenadas en el servidor MySQL.
- ✓ Realizar tareas administrativas en el servidor de bases de datos.



A continuación vemos como realizar una **conexión al servidor de bases de datos MySQL con el cliente gráfico Workbench** y algunas de las **primeras sentencias** que debes conocer.

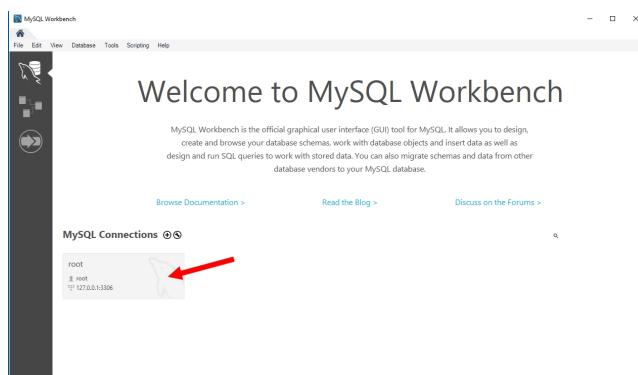
◀ 1 2 3 4 5 6 7 8 ►

## Conexión al servidor con Workbench

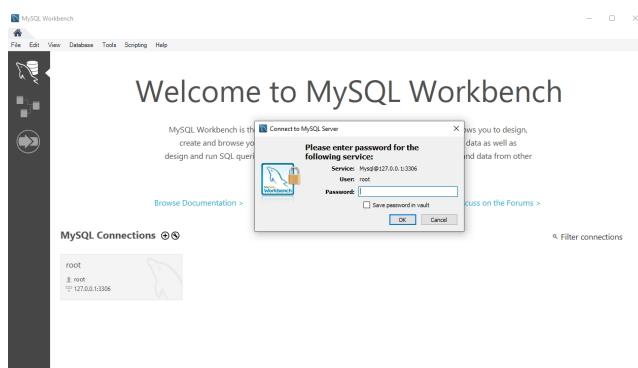
Si ejecutas Workbench, en la primera pantalla se mostrará una **conexión para el usuario root** que viene predefinida por defecto. Esa conexión lleva configurados los siguientes parámetros:

- ✓ usuario a conectarse,
- ✓ protocolo de red,
- ✓ dirección del servidor de bases de datos y
- ✓ puerto de escucha del servidor.

Pulsando con **doble clic sobre ella** aparece una ventana emergente que pedirá la contraseña del usuario y tras introducirla y pulsar al botón ok se conectarán al servidor de bases de datos, si este está iniciado. (Por defecto al instalar MySQL el servidor queda iniciado).



MySQL Workbench (Elaboración propia)



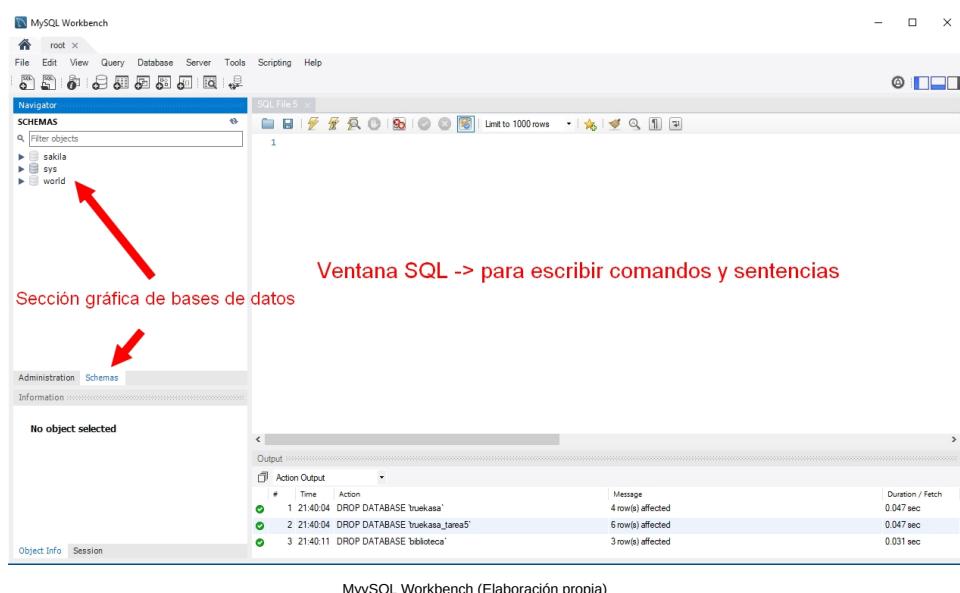
## Secciones en Workbench

Puedes ver las diferentes secciones en que se estructura Workbench. Las más importantes, de momento, son:

Ventana SQL. La usaremos para lanzar las sentencias o comandos contra el servidor de bases de datos, al igual que si lo hiciéramos desde la línea de comandos. Situada en la parte central.

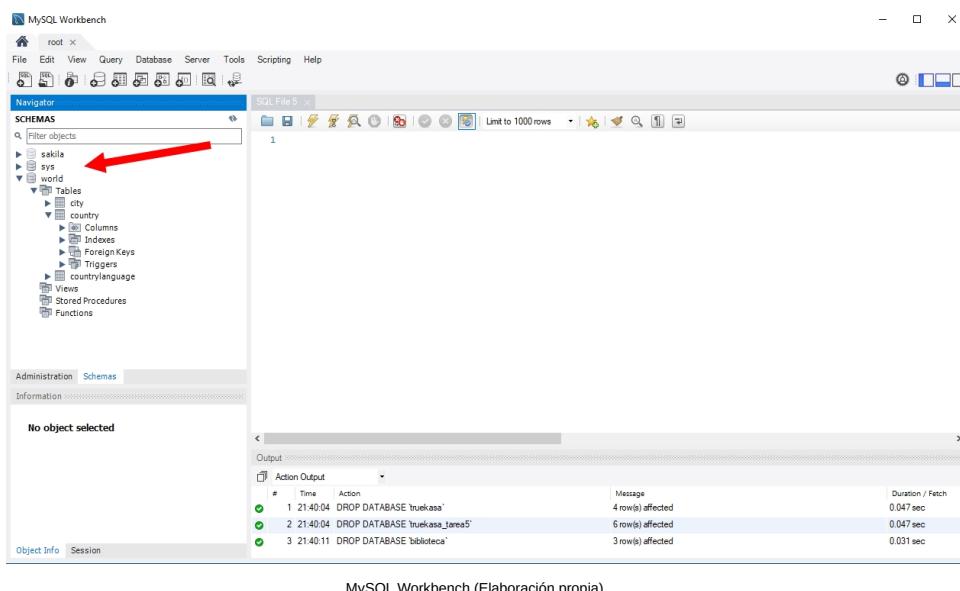
Sección que muestra de forma gráfica las bases de datos instaladas en el servidor. Situada en la parte izquierda.

Este cliente gráfico solo muestra una de las bases de datos del sistema, la denominada SYS. Las otras bases de datos que aparecen al lado de SYS, son las bases de datos de ejemplo que se añadieron durante la instalación de MySQL y son, WORLD y SAKILA.



## Desplegando elementos de una base de datos en modo gráfico

Te puedes situar sobre una base de datos y pulsar en la viñeta triangular para ir desplegando sus diferentes componentes, por ejemplo la imagen muestra desplegada la base de datos WORLD, que está formada por las tablas city, country y language.



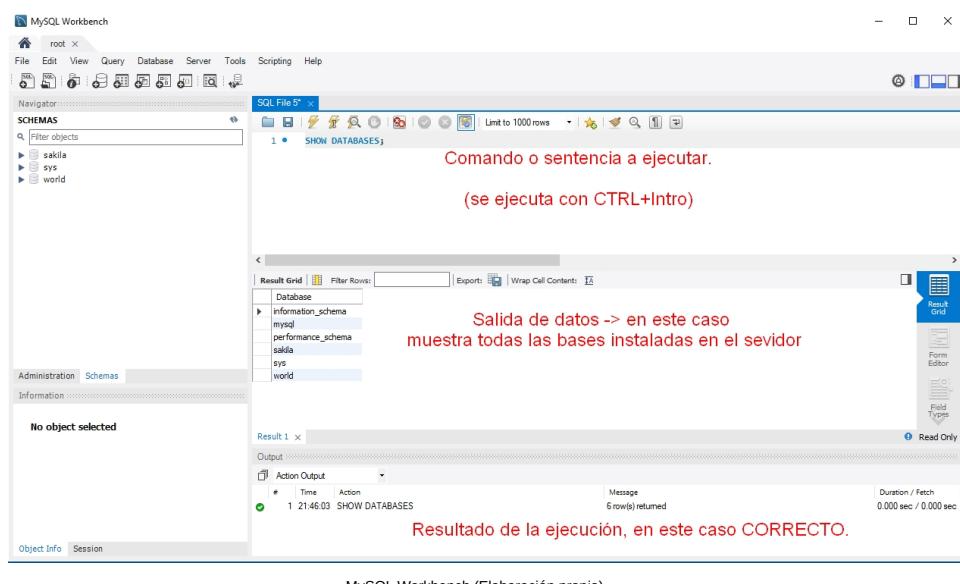
MySQL Workbench (Elaboración propia)

## Comando SHOW DATABASES;

En la ventana SQL podemos lanzar comandos o sentencias SQL contra el sevidor MySQL.

- ✓ Cada sentencia debe terminar en ";" (punto y coma).
- ✓ Para ejecutar una sentenica lo mejor es usar las teclas de atajo (CTRL+Intro) tras situar el cursor al final de la sentencia.
- ✓ Los comandos o sentencias SQL se pueden escribir en mayúsculas o minúsculas.

El resultado de ejecutar SHOW DATABASES; es que se muestran todas las bases de datos instaladas en el servidor.



MySQL Workbench (Elaboración propia)

**Poner en uso una base de datos (USE nombreBaseDatos;) y mostrar su tablas (SHOW**

## TABLES;).

Para trabajar con una determinada base de datos hay que activarla o ponerla en uso. Eso se hace con el comando USE nombreBaseDatos;

En este ejemplo, se ha puesto en uso la base de datos SAKILA y después se han mostrado sus tablas ejecutando los comandos:

- ✓ USE sakila;
- ✓ SHOW TABLES;

Observa que en este cliente gráfico, cuando una **base de datos está en uso** aparece **resaltada en negrita**. Eso también se puede conseguir en modo gráfico, haciendo doble clic sobre la base de datos que queramos activar.



MySQL Workbench (Elaboración propia)

## Consultando los datos de una tabla (SELECT \* FROM nombreTabla).

Podemos lanzar cualquier sentencia SQL, como una consulta mediante el comando SELECT para ver el contenido de una tabla, por ejemplo, con la sentencia SELECT \* FROM country; se listarán todas las filas de la tabla country .

The screenshot shows the MySQL Workbench interface. In the top navigation bar, 'root' is selected. The 'Navigator' pane on the left shows the 'SCHEMAS' section with 'sakila' selected, containing 'Tables', 'Views', 'Stored Procedures', and 'Functions'. The 'Information' pane below it says 'No object selected'. The main workspace displays the results of a query:

```

1 • SHOW DATABASES;
2 • USE sakila;
3 • SHOW TABLES;
4 • SELECT * FROM country; Consultando los datos de la tabla country

```

The 'Result Grid' shows the following data:

country_id	country	last_update
1	Afghanistan	2006-02-15 04:44:00
2	Angola	2006-02-15 04:44:00
3	American Samoa	2006-02-15 04:44:00
4	Angola	2006-02-15 04:44:00
5	Anguilla	2006-02-15 04:44:00
6	Argentina	2006-02-15 04:44:00
7	Armenia	2006-02-15 04:44:00
8	Australia	2006-02-15 04:44:00
9	Austria	2006-02-15 04:44:00
10	Azerbaijan	2006-02-15 04:44:00
11	Bahrain	2006-02-15 04:44:00
12	Bangladesh	2006-02-15 04:44:00
13	Belarus	2006-02-15 04:44:00
14	Bolivia	2006-02-15 04:44:00
15	Bulgaria	2006-02-15 04:44:00
16	Burundi	2006-02-15 04:44:00
17	Búlgaria	2006-02-15 04:44:00
18	Cambodia	2006-02-15 04:44:00
19	Camerún	2006-02-15 04:44:00
20	Canada	2006-02-15 04:44:00
21	Chad	2006-02-15 04:44:00

The 'Output' pane at the bottom shows the execution log:

#	Time	Action	Message	Duration / Fetch
2	21:46:32	USE sakila	0 row(s) affected	0.000 sec
3	21:46:49	SHOW TABLES	23 row(s) returned	0.000 sec / 0.000 sec
4	21:47:39	SELECT * FROM country LIMIT 0, 1000	109 row(s) returned	0.016 sec / 0.000 sec

MySQL Workbench (Elaboración propia)

## Mostrar la estructura o descripción de una tabla (DESCRIBE nombreTabla;)

Otro comando muy útil desde el principio, es el que describe o muestra la estructura de una tabla, esto es, las columnas que tiene, el tipo de datos de cada columna y sus restricciones. Para ello si queremos ver la descripción o estructura de la tabla country daríamos la siguiente orden:

- ✓ DESCRIBE country;
- ✓ o también se puede poner, de forma abreviada, DESC country;

The screenshot shows the MySQL Workbench interface. In the top navigation bar, 'root' is selected. The 'Navigator' pane on the left shows the 'SCHEMAS' section with 'sakila' selected, containing 'Tables', 'Views', 'Stored Procedures', and 'Functions'. The 'Information' pane below it says 'No object selected'. The main workspace displays the results of a query:

```

1 • show databases;
2 • use sakila;
3 • show tables;
4 • select * from country;
5 • describe country; Mostrando la estructura de la tabla country

```

The 'Result Grid' shows the following table structure:

Field	Type	Null	Key	Default	Extra
country_id	smallint(5) unsigned	NO	PRI	NULL	auto_increment
country	varchar(50)	NO		NULL	
last_update	timestamp	NO		CURRENT_TIMESTAMP	on update CURRENT_TIMESTAMP

The 'Output' pane at the bottom shows the execution log:

#	Time	Action	Message	Duration / Fetch
4	17:27:05	select * from country LIMIT 0, 5000	109 row(s) returned	0.016 sec / 0.000 sec
5	17:27:26	describe country	3 row(s) returned	0.000 sec / 0.000 sec

MySQL Workbench (Elaboración propia)

## Desconexión del servidor de bases de datos

Para cerrar la sesión o realizar la desconexión del servidor de bases de datos, basta con cerrar la pestaña de la conexión o bien salir del cliente Workbench.



MySQL Workbench (Elaboración propia)

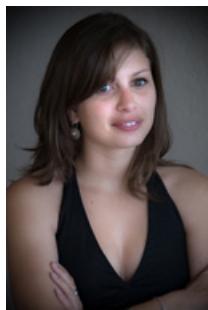
## Recomendación

Realiza una conexión con Workbench (como usuario root ) a tu servidor de bases de datos MySQL y realiza las operaciones vistas anteriormente:

- 1.- Conéctate al servidor MySQL como usuario root.
- 2.- Ejecuta el comando para ver todas las bases de datos instaladas en tu servidor.
- 3.- Pon en uso una de las bases de datos que tengas.
- 4.- Muestra las tablas de esa base de datos.
- 5.- Describe la estructura de una de sus tablas.
- 6.- Consulta todos los datos de una tabla.
- 7.- Desconecta del servidor o cierra sesión.

## 4.- Estructuras físicas de almacenamiento.

### Caso práctico



Alain Bachellier (CC BY-NC-SA)

Noiba debe decidir entre los distintos motores de almacenamiento que utiliza un SGBDD. Dependiendo de los motores de almacenamiento soportados veremos los tipos de tablas que se pueden crear y sus limitaciones. Noiba sabe que la elección del tipo de tabla puede afectar mucho al rendimiento de la base de datos del taller. Por ejemplo, cuando debe primar la seguridad en las operaciones de modificación sobre el contenido de las tablas, se debe optar por las tablas de transacción segura (**InnoDB**). Si se desea que prime la velocidad en el acceso a los datos, por ejemplo cuando se accede a ellos a través de Internet, debe optarse por tablas **MyISAM**. Veamos que significa todo esto.

MySQL soporta varios motores de almacenamiento. Algunos motores de almacenamiento tratan con tablas transaccionales y otros no. Por tanto definiremos previamente el concepto de tabla transaccional o tabla de transacción segura.

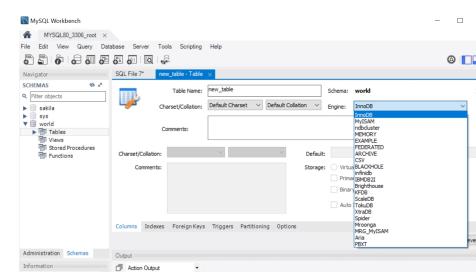
Una transacción en un Sistema de Gestión de Bases de Datos (SGBD), es un conjunto de órdenes que se ejecutan formando una unidad de trabajo, es decir, en forma indivisible o atómica. Un SGBD se dice transaccional si es capaz de mantener la integridad de los datos, haciendo que estas transacciones no puedan finalizar en un estado intermedio. Cuando por alguna causa el sistema debe cancelar la transacción, empieza a deshacer las órdenes ejecutadas hasta dejar la base de datos en su estado inicial (llamado punto de integridad), como si la orden de la transacción nunca se hubiese realizado.

Para esto, el lenguaje de consulta de datos SQL (Structured Query Language), provee los mecanismos para especificar que un conjunto de acciones deben constituir una transacción.

Una tabla se dice transaccional si es capaz de soportar transacciones.

A la hora de elegir el motor de almacenamiento tendremos que saber que algunos de ellos serán de uso obligatorio si queremos tener ciertas opciones disponibles. Por ejemplo, el soporte para claves ajenas o foráneas sólo está disponible para el motor **InnoDB**. Los motores de almacenamiento más utilizados son:

- ✓ **BerkeleyDB o BDB:** tablas de transacción segura con bloqueo de página.
- ✓ **HEAP o MEMORY:** tablas almacenadas en memoria.
- ✓ **InnoDB:** tablas de transacción segura con bloqueo de fila y claves foráneas.
- ✓ **MERGE o MRG\_MyISAM:** una colección de tablas MyISAM usadas como una única tabla.
- ✓ **MyISAM:** el nuevo motor binario de almacenamiento portable que reemplaza a ISAM.



Workbench (Elaboración propia)

Para todos los tipos de tablas, excepto las HEAP, se crean varios archivos que contienen información sobre su estructura y sobre su contenido. Parte de estos archivos quedan almacenados en la carpeta o directorio de la base de datos correspondiente.

Como MyISAM, los motores de almacenamiento MEMORY y MERGE tratan tablas no transaccionales. Los motores de almacenamiento que soportan transacciones son InnoDB y BDB. Generalmente usaremos tablas **MyISAM** o tablas **InnoDB**. A veces, cuando se requiera una gran optimización, crearemos tablas temporales en memoria (MEMORY).

## Autoevaluación

Selecciona la opción correcta si necesitamos trabajar con tablas que soporten transacciones y claves ajenas.

- BDB soporta tanto transacciones como claves ajenas.
- InnoDB soporta tanto transacciones como claves ajenas.
- Podemos elegir cualquier tipo de tabla.
- Las opciones a y b son correctas.

Incorrecto. El motor BDB no soporta claves ajenas.

Cierto, este motor tiene soporte para claves ajenas y proporciona tablas transaccionales.

Falso. Únicamente InnoDB soporta claves ajenas.

Falso. Debes leer de nuevo este apartado antes de continuar.

## Solución

1. Incorrecto
2. Opción correcta
3. Incorrecto
4. Incorrecto

## 4.1.- Tablas no transaccionales.

Comenzamos viendo los motores mas importantes para el tipo de tablas no transaccionales.

### Motor de almacenamiento MyISAM.

Este motor trata tablas no transaccionales. Son tablas de acceso secuencial indexado, los índices indican la posición relativa en el fichero de datos; por tanto son tablas portables de un equipo a otro. Se consideran óptimas cuando las tablas se usan preferentemente para consultas ya que proporcionan almacenamiento y recuperación de datos rápida, sin embargo para realizar inserciones necesitan más recursos. Se soporta en todas las configuraciones MySQL, y es el motor de almacenamiento por defecto.

Cada tabla se almacena en disco en 3 ficheros que se denominan igual que la tabla pero con las extensiones.



#### MySAM

Extensión del fichero	Contenido del fichero
frm	almacena la definición de la tabla
MYD	almacena los datos
MYI	almacena el índice

### Motor de almacenamiento MERGE.

Es una colección de tablas MyISAM similares que pueden usarse como una sola. Todas las tablas tienen que tener los mismos tipos de columnas y de índices y estar en el mismo orden. Esto se aplica porque cuando una tabla de tipo MyISAM llega a ser demasiado grande, es aconsejable crear otra tabla MyISAM con la misma estructura, de forma que parte de los datos estén en una tabla y parte en otra porque las operaciones de consulta y modificación se hacen muy lentas cuando las tablas son muy grandes. Pero si partimos la tabla en dos, las consultas tendríamos que realizarlas en dos tablas, con lo cual tampoco sería lo adecuado. Para evitar esto se crea una tabla de tipo MERGE que es el resultado de la unión de las dos tablas. Estas tablas no contienen los datos, sino que es un símbolo de que cuando se haga una consulta sobre la tabla MERGE, la consulta se hace sobre las tablas que componen la tabla MERGE. Sin embargo nunca se podrán realizar inserciones sobre la tabla MERGE, porque el servidor desconocería en cuál de las dos tablas debería insertarse las filas.

Cuando se crea una tabla de este tipo MySQL crea dos ficheros en disco con nombres que comienzan con el nombre de la tabla y las siguientes extensiones:

#### MERGE

Extensión del fichero	Contenido del fichero
frm	almacena la definición de la tabla
MRG	contiene los nombres de las tablas que deben usarse como una

Las tablas no tienen que estar en la misma base de datos que la tabla MERGE misma.

### Motor de almacenamiento MEMORY.

El motor de almacenamiento MEMORY crea tablas con contenidos que se almacenan en memoria. Anteriormente se conocían como HEAP. Estas tablas usan índices hash por defecto, lo que las hace muy rápidas, y muy útiles para crear tablas temporales. Sin embargo, cuando se apaga el servidor, todos los datos almacenados en las tablas se pierden. Sirven para copiar en ellas los datos de una tabla en disco y que los procesos se hagan más rápido al trabajar en la memoria. Para utilizar estas tablas es necesario realizar dos procesos: copiar los datos de la tabla del disco a la tabla MEMORY y después copiar los datos de nuevo a la tabla del disco cuando se han hecho las modificaciones y antes de cerrar la sesión.

Cada tabla está asociada con un fichero de disco. El nombre de fichero comienza con el nombre de la tabla y tiene una extensión de .frm

## MEMORY

Extensión del fichero	Contenido del fichero
frm	almacena la definición de la tabla

## 4.2.- Tablas transaccionales.

Uno de los motores de almacenamiento más utilizados y que soporta tablas transaccionales es el motor **InnoDB**.

### Motor de almacenamiento InnoDB.

Este motor de almacenamiento **proporciona tablas transaccionales**.

InnoDB también se incluye por defecto en todas las distribuciones binarias de MySQL 8.0. En otras se puede activar o desactivar.

Se utiliza en grandes bases de datos que necesitan alto rendimiento puesto que además de la capacidad de recuperación de fallos, InnoDB gestiona múltiples usuarios simultáneamente. Como característica destacada, soporta también restricciones de clave ajena (FOREIGN KEY) de las cuales hablaremos en esta unidad. Las tablas InnoDB pueden ser de cualquier tamaño.

InnoDB almacena tablas e índices en un espacio de tablas que puede consistir en varios ficheros. Por eso no se puede trasladar un fichero InnoDB trasladando el archivo <b>.frm</b>.

Cuando se crea una tabla InnoDB se crea un archivo de formato de tabla, con extensión **.frm**. De forma predeterminada MySQL crea un archivo denominado <b>ibdata1</b> en la carpeta DATA. A este tipo de archivo se le denomina tablespace o espacio de tablas y en él se almacena toda la información relativa a índices y contenidos de todas las tablas de tipo InnoDB. Inicialmente este tablespace se crea con un tamaño de 10 MB, pero se incrementa automáticamente a medida el contenido de las tablas InnoDB lo requiera. También se puede modificar el espacio de tablas por defecto para todas las tablas.

Carpetas	Nombre	Tamaño
AppServ	test	1 KB
Apache2.2	clientes	10.240 KB
moodledata	ib_logfile0	10.240 KB
MySQL	ib_logfile1	10.240 KB
bin	ibdata1	10.240 KB
data	infoalisa11.pid	1 KB

Oracle MySQL (Elaboración propia)

### InnoDB

Extensión del fichero	Contenido del fichero
frm	Ibdata1
almacena la definición y todas las entradas del diccionario de datos	Información relativa a índices y contenidos de todas las tablas InnoDB

### Motor de almacenamiento BDB.

El motor de almacenamiento **BDB** proporciona también tablas transaccionales. Este tipo de tablas no se pueden mover de un directorio a otro. En comparación con las tablas MyISAM la búsqueda secuencial es más lenta y también se trata de tablas más grandes porque suele haber huecos donde pueden insertarse registros. BDB se incluye en la distribución binaria MySQL-Max en aquellos sistemas operativos que la soportan.

En BDB cada tabla se guarda en dos ficheros: un fichero con extensión <b>.frm</b> y otro con extensión <b>.db</b>.

### BDB

Extensión del fichero	Contenido del fichero
.frm	almacena la definición de la tabla
.db	Contiene los datos de la tabla e índices

## Autoevaluación

Responde si es verdadera o falsa la siguiente afirmación:

Si necesitamos crear una tabla que contenga todos los servicios que ofrecemos en nuestro taller mecánico a través de nuestra página web la opción más apropiada sería una tabla tipo MyISAM.

Verdadero  Falso

**Falso**

Si necesitamos crear una tabla que contenga todos los servicios que ofrecemos en nuestro taller a través de nuestra página Web la opción más apropiada sería una tabla tipo InnoDB

## Para saber más

En este apartado se han resumido las características de los principales motores de almacenamiento de MySQL. Si necesitas ampliar esta información puedes consultar el manual correspondiente.

[Motores de almacenamiento de MySQL](#)

## 5.- Tipos de datos.

### Caso práctico

Para definir la estructura de la base de datos **Noiba debe empezar por crear las tablas y las relaciones entre ellas**, tal y como ha recogido en el esquema. Dentro de cada tabla deberá definir una a una las columnas correspondientes a los atributos que necesita almacenar, y en este punto tendrá que detenerse a pensar qué tipo de datos requiere cada uno de esos atributos.

**Vindio** recomienda a **Noiba** que comience por **conocer previamente los tipos de datos** de que dispone el gestor y después aplicar un criterio profesional, ya que de ello van a depender los procedimientos que luego se lleven a cabo con esos datos.

Por ejemplo, si **Noiba** necesita un campo para guardar un código compuesto por números, por ejemplo el DNI, quizás sea más interesante utilizar un tipo de datos de caracteres en lugar de numérico, ya que no necesitaremos realizar procedimientos matemáticos con ese atributo y en cambio los datos de caracteres son más rápidos de procesar.



Alain Bachellier (CC BY-NC-SA)

Cada tabla de una base de datos se crea con una o más columnas. En la sentencia de creación de tablas (CREATE TABLE) es necesario especificar el tipo de datos que cada columna puede contener. Para elegir el tipo de columna habrá que tener en cuenta:

- ✓ Qué clase de valores vamos a almacenar.
- ✓ Cuanto espacio ocupan esos valores.
- ✓ Si son de ..... longitud fija o de ..... longitud variable.
- ✓ Si permite o no valores NULL.

Cada base de datos introduce tipos de valores de columna que no necesariamente están presentes en otras. Sin embargo, existe un conjunto de tipos que están representados en la totalidad de estas bases.

### Tipos de datos

Tipo de valor	Contenido
Alfanuméricicos	Contienen caracteres. Presentan una longitud limitada (255 caracteres).
Numéricos	Existen de varios tipos, principalmente, enteros (sin decimales) y reales (con decimales).
Booleanos	Poseen dos formas: verdadero y falso (Sí o No).
Fecha y hora	Almacenan fechas facilitando posteriormente su explotación. Almacenar fechas de esta forma posibilita ordenar los registros por fechas o calcular los días entre una fecha y otra.
Memos	Son campos alfanuméricicos de longitud ilimitada. Presentan el inconveniente de no poder ser indexados (veremos más adelante lo que esto quiere decir).
Autoincrementales	Son campos numéricos enteros que incrementan en una unidad su valor para cada registro incorporado. Su utilidad resulta más que evidente: Servir de identificador ya que resultan exclusivos de un registro.

### Autoevaluación

**De los siguientes atributos ¿Con cuál de ellos podrías utilizar un tipo autoincrementable?**

- El CIF de un cliente.
- El número de matrícula de un alumno (siempre que fuese codificado exclusivamente con dígitos).
- El número de unidades compradas en una factura.
- La edad de un empleado.

Falso. No se trata de un valor consecutivo, además incorpora caracteres alfabéticos.

Correcto. Podría guardarse como un número consecutivo. Cada vez que se añada un nuevo alumno a la base de datos el sistema le asignará el número de matrícula resultante de sumar 1 al número de matrícula del alumno anterior.

Falso. No puede ser autoincrementable, se necesita almacenar un número concreto.

Incorrecto. Este dato se incrementa anualmente, no con cada nuevo registro.

## Solución

1. Incorrecto
2. Opción correcta
3. Incorrecto
4. Incorrecto

## Para saber más

Si necesitas más información sobre los tipos de datos, para columnas, puedes acceder al manual de MySQL en español

[Tipos de columnas en MySQL](#)

## 5.1.- Tipos de datos de cadena.

Los tipos que se utilizan en MySQL para almacenar datos alfanuméricos son los siguientes:

### Tipos de datos cadena

Tipo	Descripción	Rango	Almacenamiento
<code>&lt;b&gt;CHAR(N)&lt;/b&gt;</code>	Cadena de longitud fija. Se rellena con espacios.	De 1 a 255 caracteres	N bytes
<code>&lt;b&gt;VARCHAR(N)&lt;/b&gt;</code>	Cadena de longitud variable.	De 1 a 255 caracteres	L+1 bytes
<code>&lt;b&gt;TINYBLOB&lt;br /&gt;BLOB&lt;br /&gt;MEDIUMBLOB&lt;br /&gt;LONGBLOB&lt;/b&gt;</code>	Pequeño valor BLOB BLOB Normal BLOB Medio BLOB Grande	L < 2 <sup>8</sup> L < 2 <sup>16</sup> L < 2 <sup>24</sup> L < 2 <sup>32</sup>	L+1 bytes L+2 bytes L+3 bytes L+4 bytes
<code>&lt;b&gt;TINYTEXT&lt;br /&gt;TEXT&lt;br /&gt;MEDIUMTEXT&lt;br /&gt;LONGTEXT&lt;/b&gt;</code>	Pequeño valor TEXT TEXT Normal TEXT Medio TEXT TEXT Grande	L < 2 <sup>8</sup> L < 2 <sup>16</sup> L < 2 <sup>24</sup> L < 2 <sup>32</sup>	L+1 bytes L+2 bytes L+3 bytes L+4 bytes
<code>&lt;b&gt;ENUM("valor1", "valor2",...)&lt;/b&gt;</code>	Se le asigna un valor del conjunto enunciado	Número máximo de valores: 65.535	1 ó 2 bytes
<code>&lt;b&gt;SET("valor1", "valor2",...)&lt;/b&gt;</code>	Se le asigna 0 o más valores del conjunto	Número máximo de valores: 64	1,2,3,4, u 8 bytes

L: longitud que ocupa la cadena.

N: número máximo de caracteres que puede tener la cadena.

A continuación se especifica algo más sobre estos tipos:

#### ✓ `<b>CHAR Y VARCHAR</b>`

Se diferencian en que `<b><i>CHAR</i></b>` es de longitud fija y `<b><i>VARCHAR</i></b>` de longitud variable. No pueden mezclarse en la misma tabla columnas de longitud fija y variable, por tanto `<b>CHAR</b>` no puede mezclarse con columnas `<b><i>VARCHAR</i></b>`, `<b><i>BLOB</i></b>` o `<b><i>TEXT</i></b>`. En este caso se convierten las columnas `<b><i>CHAR</i></b>` a `<b><i>VARCHAR</i></b>`.

Se pueden mezclar columnas `<b><i>CHAR</i></b>` menores de cuatro caracteres y `<b><i>VARCHAR</i></b>`, sin que se cambien todas a `<b><i>VARCHAR</i></b>`. Si todas las columnas son cortas se convierten todas automáticamente a `<b>CHAR</b>` para optimizar el almacenamiento.

Cuando los valores no varían mucho de longitud es mejor `<b><i>CHAR</i></b>` porque se procesan de forma más eficiente.

#### ✓ `<b>TEXT</b>`

Permite definir cadenas de longitud variable. Entre paréntesis se pone la máxima longitud que puede tener la cadena. `<b><i>TINYTEXT</i></b>` se comporta igual que `<b><i>VARCHAR</i></b>`, pero el acceso a `<b>VARCHAR</b>` es más rápido, por eso es más aconsejable.

#### ✓ `BLOB`

Se trata de un objeto binario largo que puede albergar todo lo que se quiera. Se utilizan para almacenar documentos de procesamiento de textos, imágenes y sonidos, etc.

#### ✓ `<b>ENUM y SET</b>`

Son tipos de cadena en las que los valores se deben elegir de un conjunto fijo de cadenas. Se diferencian en que los valores **<code><i>ENUM</i></code>** deben elegir un miembro del conjunto de valores, mientras que los valores **<code><i>SET</i></code>** pueden estar formados por alguno o todos los miembros del conjunto.

## Atributos para columnas de tipo cadena:

- ✓ **<code>BINARY</code>**

Se puede especificar para los tipos **<code><i>CHAR</i></code>** y **<code><i>VARCHAR</i></code>**, sirve para que los valores de las columnas sean tratados como cadenas binarias.

- ✓ **<code>NULL / NOT NULL</code>**

Se puede utilizar en cualquier tipo de cadena. Por defecto es **<code><i>NULL</i></code>**. El valor **NOT NULL** es diferente de la cadena vacía. Para evitar una cadena vacía es necesario preverlo en las aplicaciones.

- ✓ **<code>DEFAULT</code>**

Sirve para especificar un valor predeterminado o valor por defecto, excepto en **<code><i>BLOB</i></code>** y **<code><i>TEXT</i></code>**.

### Para saber más

En el siguiente enlace puedes consultar todos los tipos de dato cadena (*string* en inglés) en la web oficial de MySQL

[Tipos de dato cadena en MySQL](#)

## 5.2.- Tipos de datos numéricos.



En la siguiente tabla puedes ver los diferentes tipos de datos numéricos en MySQL:

Everaldo Coelho and  
YellowIcon (GNU/GPL)

**Tipos de datos numéricos**

Tipo de datos	Descripción	Rango de valores		Almacenamiento requerido
		Con signo	Sin signo	
TINYINT[(N)]	Entero muy pequeño	-128 a 127	0 a 255	1 byte
SMALLINT[(N)]	Entero pequeño	-32.768 a 32.767	0 a 65.535	2 bytes
MEDIUMINT[(N)]	Entero mediano	-8.388.608 a 8.388.607	0 a 16.777.215	3 bytes
INT[(N)] INTEGER[(N)]	Entero estándar	-2.147.683.648 a 2.147.483.647	0 a 4.294.967.295	4 bytes
BIGINT[(N)]	Entero largo	-9.223.372.036.854.775.808 a 9.223.372.036.854.775.807	0 a 18.446.744.073.709.551.615	8 bytes
FLOAT[(N,D)]	Número pequeño de simple precisión de coma flotante	Valor mínimo distinto de 0  ±1,175.494.351E-38s	Valor máximo distinto de 0  ±3,402.823.466E+38s	4 bytes
DOUBLE[(N,D)] REAL[(N,D)]	Número grande de doble de precisión de coma flotante	Valor mínimo distinto de 0  ±2,2.250.738.585.072.014 E-308	Valor máximo distinto de 0  ±1,7.976.931.348.623.157 E+308	8 bytes
DECIMAL[(N,[D])] NUMERIC[(N,[D])] DEC[(N,[D])]	Número de coma flotante, representado como una cadena	Varios, el rango depende de N y D		N+2 bytes

N : Tamaño máximo de pantalla (precisión)

D : Número de decimales (escala)

### Atributos para columnas de tipo numérico:

- ✓ <b>ZEROFILL</b>

Hace que los valores mostrados en pantalla se rellenen con ceros hasta ocupar el ancho asignado a dicho valor. Cuando los valores son más grandes que el ancho de pantalla se muestran en su totalidad, no se recortan.

- ✓ <b>NULL / NOT NULL</b>

Si no se especifica, el valor predeterminado es <b><i>NULL</i></b>.

- ✓ <b>DEFAULT</b>

Sirve para especificar el valor predeterminado. Si no se especifica, se elige un valor automáticamente. Para los tipos de columna numérica es <b><i>NULL</i></b>, si no puede contener este valor, el valor por defecto sería 0.

## Atributos para columnas de tipo numérico entero:

- ✓ <b>AUTO\_INCREMENT</b>

Para generar identificadores únicos o valores en serie. Normalmente comienzan por 1 y aumentan en 1 por cada fila. Se aplica sólo a columnas de tipo entero.

- ✓ <b>UNSIGNED</b>

Impide almacenar valores negativos.

### Para saber más

En el siguiente enlace puedes consultar más sobre los tipos de datos numéricos en MySQL

[Tipos de dato numérico en MySQL](#)

## 5.3.- Tipos de datos de fecha/hora.

[Everaldo Coelho and YellowIcon \(GNU/GPL\)](#)

En la siguiente tabla puedes ver los diferentes tipos de datos en MySQL para almacenar valores de fecha y de hora:

**Tipos de datos fecha/hora**

Tipo	Descripción	Rango	Almacenamiento
DATE	Representa una fecha en formato 'AAAA-MM-DD'	"1000-01-01" a "9999-12-31"	3 bytes
TIME	Representa una hora en formato: 'HH:MM:SS'	"-838:59:59" a "838:59:59"	3 bytes
DATETIME	Representa una combinación de fecha y hora en formato: 'AAAA-MM-DD HH:MM:SS'	"1000-01-01 00:00:00" a "9999-12-31 23:59:59"	8 bytes
TIMESTAMP	Representa una combinación de fecha y hora en formato: 'AAAAMMDD HHMMSS'	19700101000000 a cualquier fecha del año 2037	4 bytes
YEAR[(2 4)]	Representa un año en formato: AAAA	1901 a 2155	1 byte

**< b > TIME Y DATETIME**

La diferencia entre **< b > < i > TIME < /i > < /b >** y **< b > < i > DATETIME < /i > < /b >** es que en **< b > < i > DATETIME < /i > < /b >** la hora es una hora del día y **< b > < i > TIME < /i > < /b >** representa el tiempo transcurrido. Es necesario especificar los segundos. Las 12:30 son 00:12:30 en lugar de las 12:30:00

**< b > TIMESTAMP**

Representan valores en formato "AAAAMMDDhhmmss". Tiene la particularidad de grabar un registro que se crea o se modifica.

**< b > YEAR**

Se utiliza para almacenar el año en lugar de una fecha completa.

### Atributos para columnas de tipo fecha y hora:

No existen atributos específicos. Pueden especificarse: **< b > < i > NULL, NOT NULL, DEFAULT < /i > < /b >**.

### Formatos para columnas de tipo fecha y hora:

En la siguiente tabla puedes ver diferentes formatos para estos tipos de dato.

Formatos	
Tipo:	Formatos permitidos:

DATETIME, TIMESTAMP	“AAAA-MM-DD hh:mm:ss”
	“AA-MM-DD hh:mm:ss”
	“AAAAMMDDhhmmss”
	“AAMMDDhhmmss”
	AAAAMMDDhhmmss
	AAMMDDhhmmss
DATE	“AAAA-MM-DD”
	“AA-MM-DD”
	“AAAAMMDD”
	“AAMMDD”
	AAAAMMDD
	AAMMDD
TIMETIME	“hh:mm:ss”
	“hhmmss”
	hhmmss
YEAR	“AAAA”
	“AA”
	AAAA
	AA

## Para saber más

En el siguiente enlace puedes ampliar la información sobre el tipo de dato fecha y hora

[Datos de tipo fecha y hora en MySQL](#)

## 6.- Introducción: el lenguaje SQL.

### Caso práctico



Jonny Goldstein (CC BY)

Noiba se plantea empezar a implementar la base de datos en el SGBD elegido buscando la utilidad gráfica que permita crear las tablas y sus relaciones, pero pronto descubre que generalmente es más operativo hacerlo a través de instrucciones. Juan le recuerda que detrás de esas herramientas se encuentra un lenguaje que permite, con instrucciones muy sencillas, **crear, consultar, manipular y administrar** las bases de datos y que es soportado por todos los SGBD relacionales. Se trata del **lenguaje SQL**, y conocerlo nos va a resultar muy práctico para comunicarnos con la base de datos. Aunque en un principio estaba pensado como lenguaje **para consultar los datos** hoy en día incorpora también instrucciones **para definir las estructuras** que necesitamos.

Para entender mejor lo que es el **lenguaje de definición de datos** vamos a explicar primero lo que es el **SQL** de una forma global y luego nos centraremos en el **Lenguaje de Definición de Datos (LDD)** (o **DDL**, *Data Definition Language* en inglés) como una parte del lenguaje **SQL**.

**SQL** (*Structured Query Language*) o Lenguaje de Consultas Estructurado, es un lenguaje que se compone de una serie de comandos que permiten a los usuarios crear las bases de datos, las estructuras de tablas y relaciones, consultar y manipular los datos almacenados y administrar esos datos.

### Tipos de sentencias SQL

Este lenguaje tiene sentencias que se utilizan en tareas muy variadas. Dependiendo de esas tareas podemos clasificar las sentencias **SQL** en cuatro grupos:

#### Sentencias SQL

Grupo de sentencias SQL	Descripción	Sentencias o comandos
<b>LDD o DDL</b> (Lenguaje de Definición de Datos)	Incluye sentencias para gestionar las estructuras.	CREATE ALTER DROP
<b>LMD o DML</b> (Lenguaje de Manipulación de Datos)	Incluye sentencias para gestionar los datos.	SELECT INSERT UPDATE DELETE
<b>LCD o DCL</b> (Lenguaje de Control de Datos)	Incluye sentencias para gestionar la seguridad y los permisos.	GRANT REVOKE
<b>LCT o TCL</b> (Lenguaje de control de Transacciones)	Incluye sentencias para controlar y gestionar transacciones.	START TRANSACTION COMMIT ROLLBACK SAVEPOINT

## Normativas y versiones.

No existe una única versión del lenguaje SQL. Muchos fabricantes de software han desarrollado extensiones del conjunto de comandos básico o variaciones concretas que llamaremos dialectos SQL, pero existen un conjunto de normas básicas que todos deben cumplir, así que la mayoría de ellos son compatibles con la normativa actual.

Las normativas pretenden evitar que cada fabricante saque al mercado sus propias versiones. Estas estandarizaciones están a cargo de la organización ISO (*International Standardization Organization*). Algunas normativas son las siguientes:

- ✓ SQL-86: contiene la funcionalidad mínima para que un lenguaje se considere SQL.
- ✓ SQL-89: añade instrucciones para gestionar las claves ajena (reglas de integridad referencial).
- ✓ SQL-92: contiene una gran cantidad de variaciones sobre el original.
- ✓ SQL:1999: se añaden extensiones hacia la programación orientada objetos.
- ✓ SQL:2003: Se han introducido características de SQL/ XML.
- ✓ SQL-2005: Define las maneras en las cuales SQL se puede utilizar conjuntamente con XML. Define maneras de importar y guardar datos XML en una base de datos SQL, manipulándolos dentro de la base de datos y publicando el XML y los datos SQL convencionales en forma XML.
- ✓ SQL-2008: Permite el uso de la cláusula ORDER BY fuera de las definiciones de los cursos. Incluye los disparadores del tipo INSTEAD OF. Añade la sentencia TRUNCATE.
- ✓ SQL-2011: Datos temporales (PERIOD FOR). Mejoras en las funciones de ventana y de la cláusula FETCH.
- ✓ SQL-2016: Permite búsqueda de patrones, funciones de tabla polimórficas y compatibilidad con los ficheros JSON.

## Autoevaluación

De las siguientes alternativas selecciona la respuesta correcta:

- La normativa SQL:2003 introduce características de HTML.
- El lenguaje SQL es una parte del LDD (lenguaje de definición de datos).
- El lenguaje SQL permite crear, consultar, modificar y controlar la base de datos.
- El lenguaje SQL es único y común a todos los fabricantes de Software.

Falso. Esta normativa añade un nuevo apartado relativo a SQL/XML.

No es así, el LDD es una parte del lenguaje SQL.

Correcto. El lenguaje SQL incorpora sentencias de definición, manipulación y control de datos.

No existe una única versión de SQL, los fabricantes tienen sus propias versiones basadas en un mismo estándar.

## Solución

1. Incorrecto
2. Incorrecto
3. Opción correcta
4. Incorrecto

## 6.1.- Sintaxis para escribir instrucciones en SQL.

En este apartado nos referiremos a los símbolos que se utilizan para describir cada una de las instrucciones que se pueden utilizar en MySQL.

La sintaxis usa los siguientes símbolos:

### Símbolos de notación

Everaldo Coelho (YellowIcon)  
(GNU/GPL)

Símbolo	Descripción
Mayúsculas	Una palabra en mayúsculas refleja una palabra reservada de <u>SQL</u> y por tanto hay que escribirla tal y como aparece en la sintaxis.
Minúsculas	Una palabra en minúsculas es algo variable y hay que sustituirla por un dato concreto.
[ ]	Los <b>corchetes</b> se utilizan para reflejar que las alternativas que contienen son opcionales. Por tanto toda palabra no encerrada entre corchetes debe ser incluida obligatoriamente en la sentencia.
	La <b>barra vertical</b> sirve para separar opciones alternativas en una lista. Solo se puede usar una de ellas.
[   ]	Cuando una lista separada con <b>barras</b> se encierra entre <b>corchetes</b> indica que se puede elegir entre estos valores opcionalmente.
{ }	Si una lista se encierra entre <b>llaves</b> , igual que pasa con los corchetes, indica que sus valores se pueden elegir, pero es necesario elegir uno de ellos.
...	Varias palabras separadas por comas y finalizadas con <b>puntos suspensivos</b> significan que se puede usar un número variable de esos datos y siempre separados con comas.

### Reflexiona

Cuando hablamos de símbolos para describir las instrucciones no significa que esos símbolos formen parte de esas instrucciones ala hora de escribirlas. Por ejemplo, los **corchetes** se utilizan para reflejar que algo es opcional, pero los corchetes en sí no se escriben como parte de una sentencia SQL. Si los incluyes te mostrará error de sintaxis. Lo mismo ocurre con las **llaves**. Sin embargo los **paréntesis** forman parte de la propia sentencia SQL y sí hay que incluirlos.

En el caso de las **mayúsculas** las utilizamos para reflejar que esa palabra es una palabra reservada de SQL y hay que escribirla así, pero podríamos hacerlo en minúsculas y no nos devolvería error. Lo mismo ocurre con las **minúsculas**, se utilizan para nombres de columnas, de tablas, o de otros objetos. El lenguaje SQL no es "case sensitive" (no distingue entre mayúsculas y minúsculas), pero por convenio y claridad se recomienda seguir la regla de **mayúsculas para palabras reservadas y minúsculas para objetos de la base de datos**.

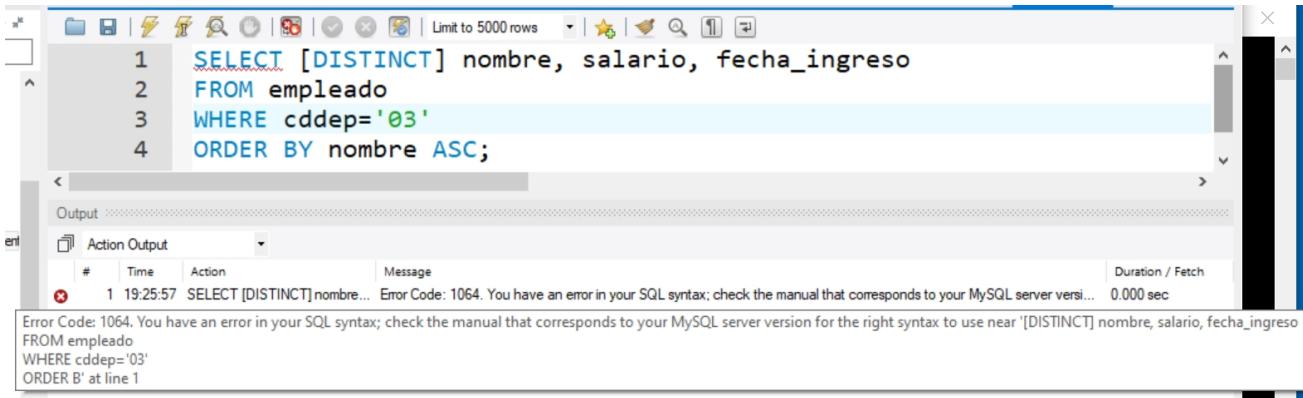
Vamos a poner como **ejemplo** la sentencia SELECT que trataremos a fondo en la siguiente unidad:

```
SELECT [ALL | DISTINCT] columna1 [columna2, columna3,.....] | * FROM tabla1 [tabla2, tabla3, ....]  
[WHERE condición ] [ORDER BY expr1 [DESC | ASC] [, expr2 [DESC | ASC] ....]
```

- ✓ Es obligatorio escribir:  
**<b><i>SELECT</i></b>** el nombre de una columna o \* (que representa todas las columnas).  
**<b><i>FROM</i></b>** nombre de una tabla.
- ✓ Son opcionales:
  - <b><i>ALL</i></b>** o **DISTINCT**: son cláusulas opcionales y si se usan, sólo puede usarse una, nunca las dos a la vez.
  - <b><i>WHERE</i></b>** condición.
  - ORDER BY expr**: A su vez tiene cláusulas opcionales incompatibles **<b><i>ASC</i></b>** y **<b><i>DESC</i></b>**.
- ✓ Tras **<b><i>SELECT</i></b>** se pueden escribir los nombres de varias columnas (separadas por comas).

## EJEMPLO.

En el ejemplo de la siguiente imagen hemos olvidado quitar los corchetes en la cláusula **<b>DISTINCT</b>** y MySQL nos muestra el error de sintaxis.



The screenshot shows a MySQL Workbench interface. In the SQL editor pane, the following query is displayed:

```

1 SELECT [DISTINCT] nombre, salario, fecha_ingreso
2 FROM empleado
3 WHERE cddep='03'
4 ORDER BY nombre ASC;

```

In the 'Output' pane, there is one entry:

#	Time	Action	Message	Duration / Fetch
1	19:25:57	SELECT [DISTINCT] nombre...	Error Code: 1064. You have an error in your SQL syntax; check the manual that corresponds to your MySQL server versi...	0.000 sec

Below the table, the full error message is shown:

Error Code: 1064. You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '[DISTINCT] nombre, salario, fecha\_ingreso' FROM empleado WHERE cddep='03' ORDER B' at line 1

Workbench (Elaboración propia)

## 6.2.- Elementos de las sentencias SQL.

Casi todas las sentencias SQL tienen una forma básica, empiezan por un verbo que es una palabra reservada que describe lo que hace la sentencia (ejemplo SELECT), a continuación le siguen una o más cláusulas que concretan mas detalles acerca de la sentencia y que también empiezan con una palabra clave, (por ejemplo FROM o WHERE). Algunas de estas cláusulas son opcionales y otras obligatorias, como vimos en el apartado anterior.

[Everaldo Coelho and YellowIcon \(GNU/GPL\)](#)

Libre Office Writer (Elaboración propia)

Hemos visto los tipos de datos que maneja MySQL y sus características, ahora veremos cómo se representan valores correspondientes a los **principales tipos de datos**:

- Un valor de tipo **carácter o cadena** de caracteres se representa **entre comillas simples o dobles**. En las comparaciones entre valores cadena de caracteres no se distingue entre mayúsculas y minúsculas.  
“SANTANDER”  
“Y”  
‘n’  
‘123’
- **Los enteros pueden representarse con signo.** Son representaciones enteras válidas:  
+78  
78  
-78
- **Los reales pueden representarse con signo.** Como separador entre parte entera y fraccionaria se usa el punto. Puede usarse la representación exponencial. Son representaciones reales válidas:  
56  
-89.008  
3.1089E+24
- **Las fechas se representan entre comillas** y en formato numérico usando como separador el guión. Por ejemplo, el 17 de enero de 2011 se representa:  
“2011-1-17”
- **Las horas** (datos de tipos TIME) se representan **entre comillas** y usando como separador el carácter dos puntos. Por ejemplo, para representar las 4 y 20 de la tarde:  
“16:20:00”
- Los datos de **tipo lógico o booleano** sólo admiten dos valores: **true y false**.

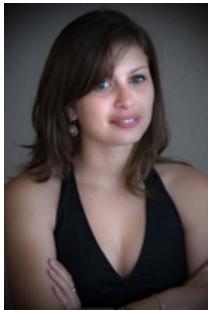
### Debes conocer

Tanto para esta unidad como para las unidades siguientes es importante que conozcas una serie de **reglas sobre cómo se escriben sentencias en SQL con relación a: valores literales, valores NULL, identificadores, palabras reservadas, comentarios y operadores**. Conocer estas reglas te evitará cometer muchos errores a la hora de escribir las instrucciones en MySQL.

[Reglas para escribir sentencias en SQL](#) (pdf - 0,37 MB)

## 7.- El lenguaje de definición de datos: LDD.

### Caso práctico



Parece que **Noiba ya tiene claro lo que necesita saber para enfrentarse al lenguaje SQL**. En este apartado presentaremos un subconjunto de ese lenguaje: el **lenguaje de definición de datos (LDD)**. En los siguientes apartados aprenderemos con ella a utilizar los comandos que nos permitirán crear las tablas dentro de una base de datos. Empezaremos por ejemplos sencillos y poco a poco iremos ampliando el modelo.

[Alain Bachellier \(CC BY-NC-SA\)](#)

La primera fase del trabajo con cualquier base de datos comienza con sentencias **DDL**, puesto que antes de poder almacenar y recuperar información debemos definir las estructuras donde almacenar la información. Las estructuras básicas con las que trabaja **SQL** son las tablas.

Conocer el Lenguaje de Definición de Datos (**DDL**) es imprescindible para crear, modificar y eliminar objetos de la base de datos (es decir, los metadatos). En el mercado hay suficientes aplicaciones y asistentes que nos facilitan esta labor, a través de una interfaz visual que nos oculta el lenguaje **SQL** y en los cuales nos limitamos a poner nombres a los campos, elegir el tipo de datos y activar una serie de propiedades.

Por ejemplo, con Workbench de MySQL podemos:

- ✓ Realizar el diseño de la base de datos, mediante su herramienta Data Modeling e implantar mediante ingeniería directa la base datos en el Servidor (si lo tenemos en marcha) o bien generar el script **SQL** (con las sentencias **SQL** para la creación de la base de datos y sus tablas) y cargarlo posteriormente en el servidor.
- ✓ Trabajar en modo gráfico con esta herramienta, a modo de formularios, y crear la base de datos, sus tablas, etc.

Es cierto que estas herramientas nos facilitan el trabajo, pero resulta imprescindible comprender y conocer en profundidad el lenguaje, ya que nos veremos en muchas situaciones donde necesitaremos crear un objeto, modificarlo o eliminarlo sin depender de esas herramientas visuales.

¿De qué objetos estamos hablando? Éstos podrán ser tablas, vistas, índices u otros objetos relacionados con la definición de la base de datos. ¿Y quién puede crear y manipularlos? En principio el usuario propietario (el que los creó) y los administradores de la base de datos. Más adelante veremos que podemos modificar los privilegios de los objetos para permitir el acceso a otros usuarios.

El lenguaje de definición de datos es la parte del **SQL** que más varía de un **SGBD** a otro. Esto se debe a que cada sistema organiza internamente los datos de una forma particular y la función del **DDL** está estrechamente vinculada a la organización interna de los datos.

El lenguaje de definición de datos es el que se encarga de la creación y modificación de la estructura de los objetos de la base de datos: creación de bases de datos, creación de tablas, modificación de la estructura de tablas, eliminación de tablas, etc.

MySQL dispone de varias sentencias **DDL**. Las operaciones básicas se realizan con las sentencias: **CREATE**, **ALTER** y **DROP**.

Veamos estas sentencias con un poco más de detalle.

**CREATE**: Crea un objeto de la base de datos. Estos objetos pueden ser tablas, vistas, índices, triggers, funciones, procedimientos, etc., o la propia base de datos. Las opciones más comunes son:

- ✓ <b>CREATE DATABASE</b>
- ✓ <b>CREATE INDEX</b>
- ✓ <b>CREATE TABLE</b>
- ✓ <b>CREATE VIEW</b>

**ALTER:** Permite modificar la estructura de un objeto. Podemos añadir o borrar campos en una tabla, modificar el tipo de un campo, añadir o quitar índices, etc. La sintaxis más frecuente es:

- ✓ <b>ALTER TABLE</b>

**DROP:** Elimina un objeto de la base de datos. Puede eliminar una tabla, una vista, un índice, trigger, función, etc., cualquier objeto que soporte la base de datos, y también la base de datos. Presenta varias opciones:

- ✓ <b>DROP DATABASE</b>
- ✓ <b>DROP INDEX</b>
- ✓ <b>DROP TABLE</b>
- ✓ <b>DROP VIEW</b>

## Autoevaluación

Relaciona cada uno de los conceptos con la instrucción SQL correspondiente:

### Ejercicio de relacionar

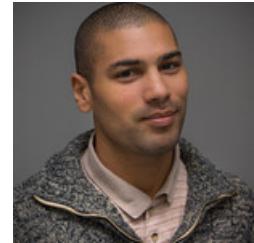
Concepto	Relacionar	Instrucción
Añadir la columna Población en la tabla MOROSOS	0	1 CREATE DATABASE
Crear la tabla MOROSOS en la base de datos ALQUILERES	0	2 ALTER TABLE
Crear la base de datos ALQUILERES	0	3 DROP DATABASE
Borrar la base de datos ALQUILERES	0	4 CREATE TABLE

El lenguaje SQL es un lenguaje muy intuitivo.

## 8.- Creación, modificación y eliminación de bases de datos.

### Caso práctico

Siguiendo los pasos de **Noiba** y **Vindio** hemos aprendido que la base de datos es una estructura compartida e integrada que aloja un conjunto de datos y metadatos para el usuario final y que para un SGBD la estructura de la base de datos es un conjunto de archivos físicos guardados en disco.



Alain Bachellier (CC BY-NC-SA)

**Noiba** y **Vindio** se disponen ahora a dar el siguiente paso que consistirá en hacer dos cosas:

- 1.- **Crear una estructura de base de datos que contendrá todas las tablas.**
- 2.- **Crear las tablas que guardarán los datos.**

En este apartado aprenderemos con ellos a crear la estructura de la base de datos. Dejaremos la creación de las tablas para los apartados siguientes.

Básicamente, la creación de la base de datos consiste en crear las tablas que la componen.

Crear una base de datos implica indicar los archivos y ubicaciones que se van a utilizar además de otras indicaciones técnicas y administrativas. Es obvio que todo esto sólo lo puede realizar si se tiene privilegio de Administrador.

Con el estándar de SQL la instrucción a usar sería CREATE DATABASE, pero cada SGBD tiene su propio procedimiento para crear las bases de datos.

En el caso de MySQL tenemos 3 sentencias a nivel de base de datos:

- CREATE DATABASE: para crear bases de datos.
- USE DATABASE: para activar una base de datos o ponerla en uso.
- DROP DATABASE: para eliminar una base de datos.

Veamos estas sentencias con más detalle a continuación.

#### ✓ CREATE DATABASE

Cuando se crea una base de datos nueva el gestor crea automáticamente las tablas que guardarán los metadatos. Para crear una base de datos nueva escribiremos:

```
CREATE {DATABASE | SCHEMA} [IF NOT EXISTS] NombreBaseDatos;
```

- ✓ Obligatoriamente se debe poner bien SCHEMA o bien DATABASE, en MySQL es indiferente, las dos formas pueden usarse indistintamente.
- ✓ Opcionalmente se puede incluir IF NOT EXISTS. Si lo incluimos, al crear la base de datos de nombre NombredemiBaseDatos, si no existe una base de datos con ese nombre, la crea, si ya existe, no intentará crearla ni dará error.

Automáticamente el servidor crea una carpeta vacía dentro de la carpeta DATA con el mismo nombre de la base de datos.

Por ejemplo, para crear una base de datos de nombre talleres\_faber, la sentencia SQL sería:

```
CREATE DATABASE talleres_faber;
```

#### ✓ USE DATABASE

Con la sentencia `CREATE DATABASE talleres_faber;` hemos creado una base de datos denominada `talleres_faber`, pero eso no significa que esa sea la base de datos activa. Para convertirla en la base de datos activa tendremos que escribir:

```
USE NombreBaseDeDatos;
```

A partir de este momento cualquier comando que se ejecute se referirá a la base de datos que acabamos de activar.

Por ejemplo, para activar la base de datos de nombre `talleres_faber`, la sentencia SQL sería:

```
USE talleres_faber;
```

✓ <span class="destacado\_inline">DROP DATABASE</span>

Se utiliza para eliminar una base de datos. Elimina la base de datos y todos los objetos que contenga (tablas,índices, vistas, etc.)

```
DROP DATABASE NombreBaseDeDatos;
```

Por ejemplo, si queremos eliminar la base de datos `talleres_faber`, la sentencia sería:

```
DROP DATABASE talleres_faber;
```

## Reflexiona

Workbench (Elaboración propia)

MySQL (Elaboración propia)

Tal y como viste en el apartado de Primeros pasos con MySQL, bien desde el cliente de la línea de comandos o bien desde el cliente gráfico Workbench, te puedes conectar a MySQL con el usuario creado en la instalación y su contraseña, y acceder a la ventana para introducir el código SQL como se ve en las imágenes de arriba.

## Autoevaluación

**Indica si la siguiente afirmación es verdadera o falsa.**

Al crear una base de datos mediante la sentencia:

CREATE DATABASE talleres\_faber;  
de manera automática la base de datos talleres\_faber se activa y queda en uso.

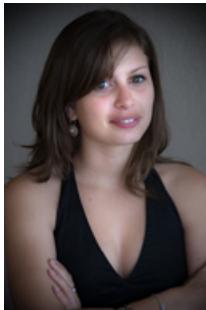
- Verdadero  Falso

**Falso**

La única forma de activar o poner en uso una base de datos es mediante el comando USE nombreBaseDatos;

## 9.- Creación, modificación y eliminación de tablas.

### Caso práctico



Alain Bachellier (CC BY-NC-SA)

Noiba ya tiene creada la base de datos **TalleresFaber**, pero hasta el momento no contiene ninguna tabla. La mayoría de los SGBD poseen potentes editores para crear de forma rápida y sencilla cualquier tipo de tabla; sin embargo **Noiba decide que es mejor guardar los comandos SQL en un guión** (fichero *script* en inglés) para poder realizar el mantenimiento y actualización de la base de datos de forma fácil. Por esto, y porque muchas veces podemos ahorrarnos quebraderos de cabeza cuando no conocemos bien el editor, aprenderemos con ella a crear, modificar y borrar tablas a partir de sentencias **SQL**.

Una vez creada nuestra base de datos "talleres\_faber" podemos comprobar que existe una carpeta con ese nombre dentro de la carpeta <i><b>data</b></i>, en la siguiente ruta:

C:\ProgramData\MySQL\MySQL Server 8.0\Datatalleres\_faber

Aquí es donde MySQL va a guardar nuestras bases de datos.

Una vez creada la base de datos, el siguiente paso sería crear las tablas de esa base de datos.

¿Qué necesitamos para poder guardar los datos? Lo primero será definir los objetos donde vamos a agrupar esos datos. Los objetos básicos con los que trabaja SQL son las tablas, que como ya sabemos es un conjunto de filas y columnas cuya intersección se llama celda. Es ahí donde se almacenarán los elementos de información, los datos que queremos recoger.

Antes de crear la tabla es conveniente planificar algunos detalles:

- ✓ **Qué nombre** le vamos a dar a la tabla.
- ✓ **Qué nombre** le vamos a dar a cada una de las **columnas**.
- ✓ **Qué tipo y tamaño de datos** vamos a almacenar en cada columna.
- ✓ **Qué restricciones** tenemos sobre los datos.
- ✓ Alguna otra **información adicional** que necesitemos.

Y debemos tener en cuenta otras **reglas** que se deben cumplir **para los nombres de las tablas**:

- ✓ No podemos tener nombres de tablas duplicados en una misma base de datos (usuario).
- ✓ Deben comenzar por un carácter alfabético.
- ✓ Su longitud máxima es de 30 caracteres.
- ✓ Solo se permiten letras del alfabeto inglés, dígitos o el signo de guión bajo.
- ✓ No puede coincidir con las palabras reservadas de **SQL** (por ejemplo, no podemos llamar a una tabla <span lang="en">SELECT</span>).
- ✓ No se distingue entre mayúsculas y minúsculas.

Como te hemos comentado antes, podrías crear la base de datos y sus tablas utilizando una herramienta gráfica. Aunque aprender a utilizar las herramientas gráficas proporcionadas por el SGBD no es el objetivo de este módulo, ya que lo que pretendemos es conocer el lenguaje **SQL**, en muchos casos estas herramientas nos pueden ahorrar trabajo. Por tanto veremos cómo se accede a la creación de tablas con Workbench:

- 1.- Nos conectamos al servidor MySQL mediante el cliente Workbench, con el usuario y contraseña que creamos en la instalación.
- 2.- Seleccionamos nuestra base de datos en el menú de la izquierda de la pantalla.
- 3.- La ponemos en uso haciendo doble clic sobre ella.
- 4.- Aparece un contenedor para tablas. Hacemos clic derecho y seleccionamos 'crear tabla'.
- 5.- En la ventana siguiente definiremos una a una las características de cada columna.

Observa la siguiente imagen:

Workbench (Elaboración propia)

Como ves, podemos crear de forma gráfica las tablas de la base de datos, pero nuestro objetivo es aprender el lenguaje SQL.

### **Sentencias SQL para la creación, modificación y eliminación de tablas**

Las principales sentencias SQL para manipular tablas en una base de datos son las siguientes:

- ✓ <b>CREATE TABLE</b>
- ✓ <b>ALTER TABLE</b>
- ✓ <b>DROP TABLE</b>

Veremos cada una de ellas en los siguientes apartados.

## 9.1.- Creación de tablas.

---

La sentencia SQL para crear tablas es CREATE TABLE.

Veamos a continuación muchas de las posibilidades que nos ofrece esta sentencia.

<span class="destacado\_inline">CREATE TABLE</span>

**Esta es la sentencia que sirve para crear tablas.** La sintaxis de esta sentencia es muy compleja porque existen muchas opciones diferentes a la hora de crear una tabla. Básicamente para crear una tabla debemos especificar:

Everaldo Coelho and  
YellowIcon (GNU/GPL)

- ✓ El nombre que le queremos asignar a la tabla.
- ✓ Los nombres de las columnas (atributos o campos) y todas las opciones que admite cada columna.
- ✓ Otras características a nivel de tabla: como si alguno de esos campos van a ser índices y de qué tipo, etc.

Si nos centramos en MySQL, la sintaxis para crear una tabla sería la siguiente:

```
CREATE TABLE [IF NOT EXISTS] nombretabla(  
Columna1 TipoDatos [restricciones de columna],  
Columna2 TipoDatos [restricciones de columna],  
.....
```

```
[restricciones de tabla]  
[ {ENGINE | TYPE} = Tipo_Tabla];
```

donde:

- ✓ nombretabla: nombre de la tabla (identificador válido según las reglas soportadas por el gestor de bases de datos).
- ✓ IF NOT EXISTS: la tabla se creará si no existe ya una con ese nombre.
- ✓ ColumnaN: nombre de la columna (identificador válido según las reglas soportadas por el gestor de bases de datos).
- ✓ TipoDatos: Tipo de dato para la columna (por ejemplo: INT, VARCHAR, CHAR, DATE, etc.)
- ✓ ENGINE: indica el tipo de almacenamiento para la tabla mediante Tipo\_Tabla. En la misma BD puede haber tablas con diferente tipo de almacenamiento, como por ejemplo son las tablas MyISAM y las tablas InnoDB. Si no se indica, asume el tipo por defecto. (Los tipos de almacenamiento de tablas permitidos en MySQL los podemos ver con la sentencia: SHOW ENGINES;)

Por ejemplo, vamos a crear una tabla de CLIENTES con las opciones básicas:

```
CREATE TABLE Clientes(  
CodCliente INTEGER NOT NULL PRIMARY KEY,  
DNI CHAR(10),  
Apellidos CHAR(25),  
Nombre CHAR(25),  
Direccion CHAR(50),  
Telefono CHAR(25)  
);
```

¿Qué son las restricciones?

Una **restricción** es una condición que una o varias columnas deben cumplir obligatoriamente. (En el siguiente apartado las veremos en detalle)

- ✓ Restricciones de columna: Afectan solo a esa columna (clave primaria, no nulo, etc.)
- ✓ Restricciones de tabla: Se indican después de especificar todas las columnas, se les puede asignar un nombre y pueden afectar a varias columnas.

Para poder estudiar todas las opciones relativas a estas restricciones vamos a dividirlas en partes:

- ✓ **Restricciones de tipo 1: restricciones o definiciones a nivel de columna.**

```
CREATE TABLE (
    Columna1 tipo [NOT NULL | NULL] [DEFAULT ValorPorDefecto] [AUTO_INCREMENT]
    [UNIQUE [KEY] | [PRIMARY] KEY] [REFERENCES NombreTabla [(NombreColumnaIndice)]] [ON
    DELETE RESTRICT | CASCADE | SET NULL | NO ACTION ] [ON UPDATE RESTRICT | CASCADE |
    SET NULL | NO ACTION] ]
    | CHECK (expr),
```

```
    Columna2 tipo [DefiniciónColumna 2] ,
    ColumnaN tipo [DefiniciónColumna N] ,  
.....  
.....
```

#### ✓ Restricciones de tipo 2: otras definiciones o restricciones.

```
.....  
.....  
| [CONSTRAINT [simbolo]] PRIMARY KEY (NombreColumnasIndice,...)  
| KEY [NombreIndice] (NombreColumnasIndice,...)  
| INDEX [NombreIndice] (NombreColumnasIndice,...)  
| [CONSTRAINT [simbolo]] UNIQUE [INDEX] [NombreIndice] [TipoIndice] (NombreColumnasIndice,...)  
| [FULLTEXT | SPATIAL] [INDEX] [NombreIndice] (NombreColumnasIndice,...)  
| CONSTRAINT [simbolo]] FOREIGN KEY [NombreIndice] (NombreColumnasIndice,...)  
| REFERENCES NombreTabla(NombreColumnasIndice,...)  
| [ON DELETE RESTRICT | CASCADE | SET NULL | NO ACTION]  
| [ON UPDATE RESTRICT | CASCADE | SET NULL | NO ACTION]  
| CHECK (expr)  
) <code>[ENGINE [=] NombreDeMotor]
```

## Para saber más

En el siguiente enlace de la web oficial de MySQL puedes ampliar información y ver el formato completo de la sentencia CREATE TABLE , así como ejemplos de creación de tablas.

[Crear tablas en MySQL](#)

## 9.1.1.- Restricciones a nivel de columna.

**Restricciones de columna o de tipo 1.** A la hora de definir columnas tenemos más opciones además del nombre y el tipo de datos, como: valor por defecto, si puede contener o no valores nulos, crear la clave primaria, índices, etc. A esas condiciones se les denomina restricciones. Si una misma columna tiene varias restricciones, éstas se separan con espacios (nunca con comas). La coma se utilizará al finalizar la definición de una columna y pasar a la siguiente.

La sintaxis para definir columnas es la siguiente:

Everaldo Coelho and  
YellowIcon (GNU/GPL)

```
NombreColumna tipo [NOT NULL | NULL] [DEFAULT ValorPorDefecto] [AUTO_INCREMENT]
[UNIQUE [KEY] | [PRIMARY] KEY] [REFERENCES NombreTabla [(NombreColumnaIndice)]] [ON
DELETE opcion] [ON UPDATE opcion] ] | CHECK (expr)
```

Vamos a ver el significado de cada una de esas restricciones:

- ✓ **NULL | NOT NULL:** Sirve para definir si la columna podrá contener o no valores nulos. La opción por defecto es NULL. En el caso de que sean columnas que forme parte de la clave primaria no podrán contener valores nulos, pero hay otros casos en los que esta opción nos podría interesar.
- ✓ <b>DEFAULT:</b> Sirve para definir el valor que tendrá por defecto una columna. Este se asignará de forma automática cuando no se especifique otro valor al añadir filas. Cuando una columna puede contener nulos y no se especifica valor por defecto este será NULL.
- ✓ **AUTO\_INCREMENT:** Es una columna autoincrementada que tiene que ser de tipo entero. Si en una columna definida como auto\_increment no introducimos ningún valor o introducimos un valor nulo, se añade una unidad al valor anterior. Se usa sobre todo para crear una clave primaria artificial.
- ✓ <b>PRIMARY KEY</b>: Solo puede existir una clave primaria en cada tabla. Para definirla podemos usar la palabra KEY o PRIMARY KEY. La clave primaria nunca puede tener valores NULL. Si no lo especificamos MySQL lo hace de forma automática pero es aconsejable acostumbrarse a definirlo.
- ✓ **INDEX:** Indica que la columna es un índice y por tanto su contenido se almacena en una tabla de índices que agiliza las operaciones de búsqueda sobre las tablas.
- ✓ **UNIQUE:** Indica que la columna es un índice que no admite repeticiones en los datos que se introduzcan en esa columna. Por **ejemplo**: en la tabla Clientes, se ha elegido el CodCliente como PRIMARY KEY, pero el atributo DNI de igual forma no puede repetirse. Para recoger esta restricción lo podemos definir como UNIQUE.
- ✓ **REFERENCES:** Se usa para especificar que esa columna es clave ajena que referencia a una columna que es clave primaria en otra tabla. A continuación se puede definir lo que se hará en nuestra tabla en caso de que se borre o modifique la clave primaria de la tabla relacionada. Esto se explicará más despacio en el siguiente apartado.
- ✓ **CHECK:** Indica que los valores introducidos en la columna deben cumplir una determinada condición. Como verás más adelante, no tiene efecto en versiones de MySQL anteriores a 8.0.16.

Como **ejemplo** veamos como crear la tabla CLIENTES con algunas restricciones:

```
CREATE TABLE Clientes (
CodCliente INT(4) NOT NULL AUTO_INCREMENT PRIMARY KEY,
DNI VARCHAR(9) UNIQUE NOT NULL,
Apellidos VARCHAR(50),
Nombre VARCHAR(25),
Direccion VARCHAR(50),
Poblacion VARCHAR(25) DEFAULT 'Almería',
Telefono VARCHAR(9)
);
```

Aunque puede hacerse, las restricciones de tipo PRIMARY KEY, REFERENCES, INDEX, UNIQUE no suelen definirse como restricciones a nivel de columna sino como otro tipo de restricciones que veremos a continuación.

## 9.1.2.- Definición de otras restricciones (I).

Además de la definición de las columnas que acabamos de ver, podemos añadir otras definiciones o restricciones que hemos denominado **restricciones de tabla o de tipo 2**.

Cuando definimos restricciones a nivel de columna no podemos, por ejemplo, definir una clave compuesta ya que no podríamos asignar la PRIMARY KEY a dos columnas porque se interpretaría como dos claves principales. De la forma que veremos a continuación podemos definir restricciones sobre cualquier columna individual o sobre conjuntos de columnas que se hayan definido en las líneas anteriores. Al ser una sintaxis más general, utilizaremos este tipo de restricciones en muchos casos.

Everaldo Coelho and YellowIcon (GNU/GPL)

La sintaxis a seguir es la siguiente:

```
| [CONSTRAINT [simbolo]] PRIMARY KEY (NombreColumnasIndice,...)
| KEY [NombreIndice ] (NombreColumnasIndice,...)
| INDEX [NombreIndice] (NombreColumnasIndice,...)
| [CONSTRAINT [simbolo]] UNIQUE [INDEX] [NombreIndice] [TipoIndice] (NombreColumnasIndice,...)
| [FULLTEXT | SPATIAL] [INDEX] [NombreIndice] (NombreColumnasIndice,...)
| CONSTRAINT [simbolo]] FOREIGN KEY [NombreIndice] (NombreColumnasIndice,...)
| REFERENCES NombreTabla(NombreColumnasIndice,..)
| [ON DELETE RESTRICT | CASCADE | SET NULL | NO ACTION] |
| [ON UPDATE RESTRICT | CASCADE | SET NULL | NO ACTION]
| CHECK (expr)
```

El significado de cada opción es el siguiente:

- ✓ **PRIMARY KEY:** En el apartado anterior vimos como crear una clave primaria **de una sola columna**. Tanto si la clave está formada por una sola columna como, obligatoriamente, **cuando esté formada por más de una columna** (claves compuestas) otra alternativa es especificar la clave principal después de definir las columnas.
- ✓ **KEY e INDEX:** Además de la clave primaria tenemos otro tipo de índice que permite definir índices sobre una columna, sobre varias o sobre partes de columnas. Un índice se debe definir para aquellas columnas que se usen frecuentemente. Las opciones **KEY** o **INDEX** se usan indistintamente. El nombre del índice es opcional.
- ✓ **UNIQUE:** Permite definir índices con valores únicos. Lo importante de este índice es que no admite añadir filas con valores repetidas (excepto NULL, que sí se puede repetir). Se puede parecer a una PRIMARY KEY pero tiene dos diferencias fundamentales:
  - ✓ Sí admite valores NULL.
  - ✓ Puede haber más de un índice UNIQUE en una tabla.

Puede definirse sobre una columna, sobre varias o sobre partes de columnas. Se definen sobre todo sobre columnas que representan claves alternativas.

- ✓ **FULLTEXT:** Indica que una o más columnas forman un índice de tipo búsqueda de texto. Sirve para ser utilizado con funciones que permiten buscar palabras dentro de columnas de texto de tipo **CHAR**, **VARCHAR** y la familia de tipos **<b><i>TEXT</i></b>**. Se comporta igual que un índice **<b><i>INDEX</i></b>** o **<b><i>UNIQUE</i></b>** pero solo puede aplicarse a esos tipos de columna. Estos índices sólo pueden usarse con tablas MyISAM. Las búsquedas full-text se realizan con las funciones **<b><i>MATCH(</i></b>** y **<b><i>AGAINST(</i></b>**.
- ✓ **CHECK:** Establece una condición que se debe cumplir para los valores insertados en la tabla. Esta restricción se puede poner al crear las tablas, pero no tiene efecto, es decir, el servidor no valida los datos afectados por esta restricción. Hasta MySQL 8.0.16 se admitía **<b><i>CHECK</i></b>** por compatibilidad con otros sistemas.

### Reflexiona

Los índices sirven para optimizar las consultas y las búsquedas de datos. Si recordamos como funciona un fichero con índices, entenderemos que su uso hace que la localización de las filas con determinados valores de columna o seguir un determinado orden a la hora de buscar es mucho más rápido. La alternativa es hacer búsquedas secuenciales, que en tablas grandes requieren mucho tiempo.

## Ejercicio resuelto

Crea la siguiente tabla definiendo las restricciones a continuación de la definición de las columnas:

<b>CLIENTES (CodCliente, DNI, Apellidos, Nombre, Direccion, Poblacion, Telefono)</b>

Restricciones:

- ✓ CodCliente: está formado por números enteros de 4 dígitos autoincrementados y es la clave principal.
- ✓ DNI: contiene 9 caracteres (incluyendo la letra). No se puede repetir ni ser nulo.
- ✓ Población: Por defecto será "Huelva".
- ✓ La tabla será de tipo InnoDB.

[Mostrar retroalimentación](#)

```
CREATE TABLE Clientes (
    CodCliente INT(4) NOT NULL AUTO_INCREMENT,
    DNI VARCHAR(9) NOT NULL,
    Apellidos VARCHAR(50),
    Nombre VARCHAR(25),
    Direccion VARCHAR(50),
    Poblacion VARCHAR(25) DEFAULT 'Huelva',
    Telefono VARCHAR(9),
    PRIMARY KEY (CodCliente),
    UNIQUE KEY (DNI)
) ENGINE=InnoDB;
```

## 9.1.3.- Definición de otras restricciones (II).

<br />**FOREIGN KEY:** Este tipo de restricción indica que las columnas indicadas entre paréntesis forman una clave ajena relativa a las columnas de otra tabla que se indica tras la palabra <b><i>REFERENCES</i></b>.

La sintaxis es la siguiente:

[Everaldo Coelho and YellowIcon \(GNU/GPL\)](#)

```
[CONSTRAINT [símbolo]] FOREIGN KEY [NombreClaveAjena] (NombreColumnasIndice...)
REFERENCES NombreTabla(NombreColumnasIndice, ...) [ON DELETE opción] | [ON UPDATE opción]
```

Los nombres de la restricción <b><i>símbolo</i></b> y de la clave ajena <b><i>NombreClaveAjena</i></b> son opcionales que se suelen dar para poder utilizarlos en instrucciones de modificación o eliminación de esa restricción.

El nombre de la tabla <b><i>NombreTabla</i></b> es la tabla a la que hace referencia la clave ajena y las opciones <b><i>ON DELETE</i></b> y <b><i>ON UPDATE</i></b> expresan las opciones de borrado y de eliminación sobre las filas de la tabla principal relacionada, es decir qué ocurre en la tabla que estamos creando si se intenta modificar o eliminar filas en la tabla principal que están relacionadas con estas.

Las opciones opcionales pueden ser:

- ✓ **CASCADE:** Borrado o modificación en cascada.
- ✓ **RESTRICT:** Borrado o modificación restringido. Significa que no se puede realizar la acción si hay filas relacionadas en la tabla secundaria.
- ✓ **SET NULL:** Borrado o modificación con puesta a nulos.
- ✓ **NO ACTION:** Borrado o modificación sin acción. Esta es la opción por defecto.

¿Pero qué significa todo eso realmente?

Al relacionar tablas mediante una columna necesitamos que el dato o valor de la columna que es clave ajena en una tabla (que llamaremos secundaria) previamente haya sido incluido en su tabla de procedencia (tabla primaria) donde es clave primaria. Recuerda que a eso se le llama **Integridad Referencial**.

Con el propósito de mantener la integridad referencial tenemos las siguientes opciones:

- ✓ **ON DELETE CASCADE:** te permitirá borrar todos los registros cuya clave ajena sea igual a la clave del registro borrado.
- ✓ **ON DELETE SET NULL:** colocará el valor NULL en todas las claves ajenas relacionadas con la borrada.
- ✓ **ON DELETE RESTRICT:** (igual que NO ACTON en MySQL) no te permitirá borrar el registro principal, si hay registros asociados con ese valor en su clave ajena.

Otras cláusulas, para mantener la Integridad Referencial, que van tras **REFERENCE** y permiten cambiar de forma automática el valor de las claves ajena en función de la modificación hecha en el valor de la clave principal son:

- ✓ **ON UPDATE CASCADE:** te permitirá modificar el valor de la clave ajena de todos los registros cuya clave ajena sea igual a la clave del registro modificado.
- ✓ **ON UPDATE SET NULL:** colocará el valor NULL en todas las claves ajena relacionadas con la modificada.
- ✓ **ON UPDATE RESTRICT:** (igual que NO ACTON en MySQL) no te permitirá modificar la clave primaria el registro principal, si hay registros asociados con ese valor en su clave ajena.

Por defecto, en **MySQL**, si no se indica nada, las opciones son:

**ON DELETE RESTRICT**

**ON UPDATE RESTRICT**

**Debes conocer**

Como habrás deducido el tipo de tablas más conveniente para nuestras bases de datos será el InnoDB, ya que, además de transacciones, nos permite establecer restricciones de clave ajena.

En las últimas versiones de MySQL el tipo predeterminado para las tablas es InnoDB.

## Ejercicio resuelto

Escribe la sentencia SQL que permita crear el siguiente esquema relacional. Elige el tipo de datos que consideres más adecuado. Incluye la actualización en cascada en las claves ajenas.

Libre Office base (Elaboración propia)

[Mostrar retroalimentación](#)

```
CREATE TABLE REPARACIONES (
    IdReparacion INTEGER NOT NULL AUTO_INCREMENT,
    Matricula VARCHAR(8) NOT NULL,
    FechaEntrada DATE,
    Avería VARCHAR(50),
    FechaSalida DATE,
    HorasEmpleadas DECIMAL(4,2),
    Reparado TINYINT(1),
    Observaciones VARCHAR(100),
    PRIMARY KEY(IdReparacion)
) ENGINE=InnoDB;

CREATE TABLE COMPONENTES (
    IdRComponente VARCHAR(5) NOT NULL,
    Descripcion VARCHAR(50),
    UnidadBase VARCHAR(25),
    Stock TINYINT,
    PrecioReferencia DECIMAL(6,2),
    PRIMARY KEY(IdComponente)
) ENGINE=InnoDB;

CREATE TABLE Incluyen (
    IdComponente VARCHAR(5) NOT NULL,
    IdReparacion INTEGER NOT NULL,
    Unidades TINYINT,
    PRIMARY KEY (IdComponente, IdReparacion),
    FOREIGN KEY fk_1 (IdComponente) REFERENCES COMPONENTES(IdComponente) ON UPDATE CASCADE,
    FOREIGN KEY fk_2 (IdReparacion) REFERENCES REPARACIONES(IdReparacion) ON UPDATE CASCADE
) ENGINE=InnoDB;
```

# Reflexiona

¿Te has encontrado algún problema en el ejercicio anterior?

Imagina que decides crear la tabla Incluyen antes que la tabla REPARACIONES, entonces cuando intentas definir la segunda clave ajena de la tabla Incluyen te estás refiriendo a la tabla REPARACIONES que aún no has creado, con lo cual MySQL no puede definir esta restricción. Está claro que esto se puede solucionar fácilmente: creando primero las tablas que no lleven claves ajenas y dejando ésta para el final; ahora bien ¿Te imaginas hacer esto en un esquema de, por ejemplo, 10 tablas relacionadas como la base de datos de TalleresFaber?

[Mostrar retroalimentación](#)

A pesar de que podemos definir todo tipo de restricciones con esta sentencia, la forma habitual de crear nuestras bases de datos será la siguiente:

- ✓ Primero crearemos todas las tablas con las restricciones necesarias, excepto las claves ajenas(**CREATE TABLE**).
- ✓ Por último añadiremos las restricciones de clave ajena, como modificaciones de las tablas ya creadas(**ALTER TABLE**).

¿No lo entiendes? No te preocupes, cuando tratemos la sentencia ALTER TABLE lo entenderás.

## Ejercicio resuelto

Partiendo del modelo relacional correspondiente a un ejercicio resuelto en la unidad 2, escribe las instrucciones necesarias para crear las tablas.

AULA ( **N\_Aula**, Piso, Pasillo, N\_Plazas, Proyector, Pizarra, Clase )

COMÚN ( **N\_Aula(fk)** )

S. ORDENADORES ( **N\_Aula(fk)**, N\_ordenadores, Scaneres, Impresoras, Varios)

LABORATORIO ( **N\_Aula(fk)**, Tipo, Equipamiento)

**Nota:** Las columnas subrayadas forman parte de las claves primarias, las columnas que aparecen con (fk) forman parte de las claves ajenas.

[Mostrar retroalimentación](#)

```
CREATE TABLE AULA (
    N_Aula TINYINT NOT NULL,
    Piso VARCHAR(2),
    Pasillo VARCHAR(10),
    N_Plazas TINYINT,
    Proyector TINYINT(1),
    Pizarra TINYINT(1),
    Clase VARCHAR(10),
    PRIMARY KEY(N_Aula)
) ENGINE=InnoDB;
```

```
CREATE TABLE COMUN (
    N_Aula TINYINT NOT NULL,
    PRIMARY KEY(N_Aula),
    FOREIGN KEY(N_Aula) REFERENCES AULA(N_Aula)
```

```
) ENGINE=InnoDB;

CREATE TABLE S_ORDENADORES (
N_Aula TINYINT NOT NULL,
N_ordenadores TINYINT,
Scaneres TINYINT,
Impresoras TINYINT,
Varios VARCHAR(100),
PRIMARY KEY(N_Aula),
FOREIGN KEY(N_Aula) REFERENES AULA(N_Aula)
) ENGINE=InnoDB;

CREATE TABLE LABORATORIO (
N_Aula TINYINT NOT NULL,
Tipo VARCHAR(25),
Equipamiento VARCHAR(50),
PRIMARY KEY(N_Aula),
FOREIGN KEY(N_Aula) REFERENCES AULA(N_Aula)
) ENGINE=InnoDB;
```

## 9.2.- Modificación de tablas.

Te puedes estar preguntando... ¿Qué ocurriría si una vez creadas las tablas, hay que añadir alguna columna que no se tuvo en cuenta inicialmente? ¿Y si se desea añadir una nueva restricción o, por el contrario, eliminarla? Está claro que el esquema de una base de datos puede sufrir algunas modificaciones a lo largo del tiempo. Esto se hace con la sentencia ALTER TABLE.

Vamos a ver cómo se usa.

Everaldo Coelho and  
YellowIcon (GNU/GPL)

**ALTER TABLE:** Permite modificar la estructura de una tabla: añadir una nueva columna, cambiar el tipo de una columna, establecer una clave primaria, eliminar una columna, establecer una clave ajena, etc. La sintaxis general de esta sentencia admite muchas opciones, las más importantes son:

```
ALTER TABLE NombreTabla
| ADD [COLUMN] DefiniciónColumna [FIRST | AFTER NombreColumna]
| ADD INDEX [NombreIndice] [TipoIndice] (NombreColumnasIndice,...)
| ADD [CONSTRAINT [Simbolo] ] PRIMARY KEY [TipoIndice] (NombreColumnasIndice,...)
| ADD [CONSTRAINT [Simbolo] ] UNIQUE [NombreIndice] [TipoIndice] (NombreColumnasIndice,...)
| ADD [FULLTEXT | SPATIAL] [NombreIndice] (NombreColumnasIndice,...)
| ADD [CONSTRAINT [Simbolo] ] FOREIGN KEY [NombreIndice] (NombreColumnasIndice,...)
| REFERENCES NombreTabla(NombreColumnasIndice,...)
| DROP [COLUMN] NombreColumna
| DROP PRIMARY KEY
| DROP INDEX NombreIndice
| DROP FOREIGN KEY Simbolo
| ALTER [COLUMN] NombreColumna {SET DEFAULT texto | DROP DEFAULT}
| CHANGE [COLUMN] NombreColumna NuevaDefinicionColumna [FIRST | AFTER NombreColumna]
| MODIFY [COLUMN] NuevaDefinicionColumna [FIRST | AFTER NombreColumna]
| RENAME NuevoNombreTabla
```

Muchas de las opciones que admite ALTER TABLE las hemos visto al estudiar CREATE TABLE. Veamos ahora las que son nuevas:

- ✓ **CHANGE:** Permite modificar el nombre, el tipo y las restricciones de una columna mediante lo indicado en NuevaDefinicionColumna. Si no se quiere cambiar el nombre habrá que repetir dos veces el mismo (el antiguo y el nuevo nombre).
- ✓ **MODIFY:** Permite modificar el tipo y las restricciones de la columna indicada en DefinicionColumna por lo indicado en esta definición. Se diferencia de **<b><i>CHANGE</i></b>** en que con **<b><i>MODIFY</i></b>** no se puede cambiar el nombre, solo la definición.
- ✓ **RENAME:** Permite renombrar la tabla.

### Ejercicio resuelto

Realiza las siguientes modificaciones en la tabla Clientes:

- 1.- Añade dos nuevas columnas:
  - 1.1.- Una para contener el código postal : entero, tamaño 5, sin signo y rellenando con 0. Que no admita nulos.
  - 1.2.- Otra columna para recoger el tipo de cliente: si es un particular o una empresa (solo dos valores posibles: "particular" o "empresa").
- 2.- Añade un índice por apellidos. (Ten en cuenta que puede haber clientes que sean hermanos).
- 3.- Elimina el índice anterior.
- 4.- Modifica la columna *Direccion* con los siguientes datos: se llamará *<i>Domicilio</i>* y será texto de tamaño 75.
- 5.- Elimina el valor por defecto que tiene la columna Ciudad.
- 6.- Borra las columnas *<i>Telefono</i>* y *<i>Ciudad</i>*.

[Mostrar retroalimentación](#)

```
ALTER TABLE Clientes
ADD COLUMN Cp INT(5) UNSIGNED ZEROFILL NOT NULL,
ADD COLUMN Tipo ENUM(`Particular`, `Empresa`);

ALTER TABLE Clientes
ADD INDEX key_1 (Apellidos);

ALTER TABLE Clientes
DROP INDEX key_1;

ALTER TABLE Clientes
CHANGE Direccion Domicilio VARCHAR(75);

ALTER TABLE Clientes

ALTER Ciudad DROP DEFAULT;

ALTER TABLE Clientes
DROP COLUMN Telefono,
DROP COLUMN Ciudad;
```

## Ejercicio resuelto

Tomando el ejercicio práctico del apartado anterior REPARACIONES-Incluyen-COMPONENTES crea las tablas en dos fases:

- 1.- Define las tablas con CREATE TABLE, a excepción de las claves ajenas.
- 2.- Con ALTER TABLE añade las restricciones de tipo clave ajena.

[Mostrar retroalimentación](#)

```
CREATE TABLE REPARACIONES (
IdReparacion INTEGER NOT NULL AUTO_INCREMENT,
Matricula VARCHAR(8) NOT NULL,
FechaEntrada DATE,
Avería VARCHAR(50),
FechaSalida DATE,
HorasEmpleadas DECIMAL(4,2),
Reparado TINYINT(1),
Observaciones VARCHAR(100),
PRIMARY KEY(IdReparacion)
) ENGINE=InnoDB;

CREATE TABLE Incluyen (
IdComponente VARCHAR(5) NOT NULL,
IdReparacion INTEGER NOT NULL,
Unidades TINYINT,
PRIMARY KEY (IdComponente, IdReparacion)
) ENGINE=InnoDB;

CREATE TABLE COMPONENTES (
IdComponente VARCHAR(5) NOT NULL,
```

```
Descripcion VARCHAR(50),  
UnidadBase VARCHAR(25),  
Stock SMALLINT,  
PrecioReferencia DECIMAL(6,2),  
PRIMARY KEY(IdComponente)  
) ENGINE=InnoDB;  
  
ALTER TABLE Incluyen  
ADD CONSTRAINT fk_1 FOREIGN KEY(IdComponente) REFERENCES COMPONENTES(IdComponente)  
ON UPDATE CASCADE,  
ADD CONSTRAINT fk_2 FOREIGN KEY(IdReparacion) REFERENCES REPARACIONES(IdReparacion)  
ON UPDATE CASCADE;
```

## Para saber más

Es interesante que veas más ejemplos de uso del comando ALTER en MySQL.

[Ejemplos de ALTER TABLE](#)

## 9.3.- Eliminación de tablas.

Imagina que en la fase de creación de las tablas has creado por error una tabla que no necesitas. ¿Cómo la puedes eliminar? Con la sentencia DROP TABLE.

<span class="destacado\_inline">DROP TABLE</span>

Esta sentencia elimina una o más tablas de una base de datos. La sintaxis es:

Everaldo Coelho and  
YellowIcon (GNU/GPL)

DROP TABLE [IF EXISTS] NombreTabla1, [NombreTabla2,...]

- ✓ IF EXISTS: La cláusula IF EXISTS se utiliza para que, en caso de que la tabla no exista, la sentencia no devuelva un código de sentencia con resultado erróneo, (que en un procedimiento o función puede ser muy importante, como veremos más adelante).

Cuando se realiza un DROP TABLE la tabla no es recuperable, por eso tendremos que utilizar esta instrucción con cuidado.

### Ejercicio resuelto

Borra las tablas REPARACIONES, INCLUYEN y COMPONENTES en una sola sentencia.

[Mostrar retroalimentación](#)

Si escribes:

DROP TABLE REPARACIONES, INCLUYEN, COMPONENTES;

MySQL te muestra un error porque la tabla INCLUYEN tiene claves ajenas. No puedes borrar la tabla REPARACIONES antes que la tabla INCLUYEN porque se perdería la integridad referencial y lo mismo ocurre con COMPONENTES. No olvides que la tabla INCLUYEN relaciona una reparación con los componentes que se han sustituido, si borras los datos de la reparación, o de los componentes, los datos de INCLUYEN asociados no tendrían sentido. Se borra en primer lugar la tabla con claves ajenas, o se eliminan las claves ajenas con ALTER TABLE.

La solución correcta sería:

DROP TABLE INCLUYEN, REPARACIONES, COMPONENTES;

### Autoevaluación

Selecciona las repuestas que consideres correctas con relación a la sentencia DROP TABLE:

- Al borrar una tabla se eliminan todas las filas que contiene, pero se mantiene la estructura de la tabla.

- Si borramos todas las tablas de una base de datos, se elimina también la base de datos.

- Esta sentencia es equivalente a ALTER TABLE ...DROP COLUMN..

- Al borrar una tabla se eliminan todas las filas que contiene, junto con la estructura de la misma.

- Si borramos todas las tablas de una base de datos, la base de datos permanece vacía.

[Mostrar retroalimentación](#)

## Solución

1. Incorrecto
2. Incorrecto
3. Incorrecto
4. Correcto
5. Correcto

## 9.4.- Ejemplos y casos resueltos.

---

Vamos a ver como ejemplo, la creación del **script SQL** que dará lugar a la **base de datos Talleres Faber**. En la unidad anterior hemos visto los pasos para diseñar un **modelo E/R** adecuado y lo hemos trasladado al **modelo relacional**. En esta unidad hemos **seleccionado un SGBD** que nos permita crear, manipular y administrar nuestras bases de datos y, por último, hemos conocido la parte del **lenguaje SQL** que nos va a permitir transformar el modelo relacional obtenido en objetos de nuestra base de datos definiendo sus propiedades.

Aplicando todo lo que has aprendido a nuestro caso práctico habremos obtenido un modelo relacional similar al que ves a continuación:

Everaldo Coelho and  
YellowIcon (GNUGPL)

CLIENTES (<b><u>CodCliente</u></b>, DNI, Apellidos, Nombre, Dirección, Teléfono)  
FACTURAS (<b><u>IdFactura</u></b>, FechaFactura, <b><i>CodCliente(fk)</i></b>, <b><i>IdReparación(fk)</i></b>)  
VEHICULOS (<b><u>Matrícula</u></b>, Marca, Modelo, Color, FechaMatriculación, <b><i>CodCliente(fk)</i></b>)  
REPARACIONES (<b><u>IdReparación</u></b>, Matrícula, FechaEntrada, Km, Avería, FechaSalida, Reparado, Observaciones)  
Intervienen (<b><u><i>CodEmpleado(fk), IdReparación(fk),</i></u></b> Horas)  
EMPLEADOS (<b><u>CodEmpleado</u></b>, DNI, Nombre, Apellidos, Dirección, Teléfono, CP, FechaAlta, Categoría)  
Incluyen (<b><u><i>IdRecambio(fk), IdReparación(fk),</i></u></b> Unidades)  
RECAMBIOS (<b><u>IdRecambio</u></b>, Descripción, UnidadBase, Stock, PrecioReferencia)  
Realizan (<b><u><i>IdReparación(fk), Referencia(fk),</i></u></b> Horas)  
ACTUACIONES (<b><i>Referencia</i></b>, Descripción, TiempoEstimado, Importe)

**Nota:** Las columnas subrayadas forman parte de las claves primarias, las columnas que aparecen con (fk) forman parte de las claves ajenas.

A continuación necesitamos abrir el editor **SQL** de nuestro **SGBD** para escribir las instrucciones que creen esas tablas en un **script**. En la siguiente imagen verás el editor de **SQL** en MySQL Workbench:

Workbench (Elaboración propia)

Una vez escritas todas las instrucciones SQL guarda el script en un fichero de extensión sql que podrás editar con el bloc de notas.

Para ver el script que genera la base de datos TalleresFaber pulsa en el siguiente enlace:

## Debes conocer

En este **videotutorial** verás como realizar el **diseño físico de una base de datos relacional y su implantación en el SGBD MySQL**. Veremos como ejemplo práctico la implantación física de la base de datos **Veterinaria**, cuyo diseño conceptual y diseño lógico se realizó en la unidad anterior.

Los puntos que vamos a tratar son los siguientes:

Revisión de algunos aspectos teóricos a tener en cuenta del **modelo relacional y del lenguaje SQL**.

Caso práctico de **implantación física de la base de datos Veterinaria y posterior modificación en la estructura** de algunas tablas con ALTER TABLE. Utilizaremos el cliente gráfico MySQL Workbench para redactar las sentencias SQL.

También vas a ver, cómo **crear un usuario de base de datos** con los permisos necesarios para gestionar esa base de datos

### Modelo relacional y creación de la base de datos veterinaria

<https://www.youtube.com/embed/vd3ljH7-7Ns>

Isabel Cruz [Descripción textual alternativa del video](#) (Dominio público)

Para practicar con lo explicado en el videotutorial, puedes descargar el enunciado de Veterinaria así como sus modelos conceptual y lógico, y los scripts SQL de creación del usuario de base de datos, la base de datos y las sentencias ALTER TABLE de modificación de tablas.

[Enunciado de Veterinaria y modelo ERE y Relacional](#) (pdf - 197722 B)

[Script SQL para crear al usuario veterinaria](#) (sql - 138 B)

[Script SQL de creación de la base de datos Veterinaria](#) (zip - 733 B)

[Script SQL para modificar tablas con ALTER TABLE](#) (sql - 463 B)

## Otros ejemplos resueltos: implantación de Películas, Videoteca y Campeonato

En los siguientes apartados vas a ver otros ejemplos de implantación de bases de datos en el SGBD MySQL, cada una de ellas con particularidades diferentes, con las que puedes practicar todo lo aprendido hasta ahora. En concreto verás el diseño físico de las bases de datos cuyo diseño conceptual y lógico realizamos en la unidad anterior: Peliculas, Videoteca y Campeonato.

En la implantación de cada una de estas nuevas bases de datos verás nuevas particularidades, como por ejemplo:

- ✓ El tipo de dato ENUM() y la restricción AUTO\_INCREMENT.
- ✓ Implementación de **claves primarias y foráneas compuestas**.
- ✓ Personalización del **nombre de CONSTRAINT** o restricciones.
- ✓ Creación de tablas sin restricción de clave foránea y su posterior **implementación de claves foráneas con ALTER TABLE**.
- ✓ Creación de vistas o VIEW para resolver consultas, aunque su estudio lo tienes detallado en un punto posterior de esta unidad.
- ✓ Uso de Workbench en modo gráfico para creación y modificación de tablas.
- ✓ Obtención del **modelo** de una base de datos implantada en el servidor mediante **ingeniería inversa**.

### Ejemplo 1. Implantación física de la base de datos PELÍCULAS.

## Implantación de BD Películas

En el siguiente videotutorial puedes ver cómo se realiza la implantación física de la base de datos Películas en MySQL, que posteriormente viene detallada en las otras secciones de este apartado.

- ✓ Verás el uso de AUTO\_INCREMENT.

### Creación de la base de películas y uso de auto\_increment

[https://www.youtube.com/embed/ob2bxkX\\_VIs](https://www.youtube.com/embed/ob2bxkX_VIs)

Isabel Cruz [Descripción textual alternativa del video](#) (Dominio público)

## Enunciado

Las especificaciones y requerimientos de la base de datos Películas son los siguientes.

- 1.- Una película se caracteriza por: código, título, año de estreno y género.
- 2.- Una película pertenece obligatoriamente a un género y solo a uno. Pero de un determinado género puede haber varias películas o ninguna.
- 3.- El género de una película se caracteriza por un código, nombre y descripción.
- 4.- En una película participan uno o varios actores y cada actor puede participar en una o varias películas.
- 5.- De los actores interesa su código, DNI, nombre, fecha de nacimiento, edad y varios email.
- 6.- Hay que considerar dos posibles tipos de actores, aunque puede haber otros tipos:

- ✓ Actores oscarizados: de los que hay que almacenar además, el total de oscars y la fecha del primer oscar.
- ✓ Actores productores: de los que hay que almacenar además, el numero de películas producidas, aportación mínima, y primer año de producción.  
Un actor oscarizado también puede ser productor.

## Diagrama ERE y Modelo Relacional

Como resultado del diseño CONCEPTUAL obtuvimos el siguiente **diagrama Entidad/Relación Extendido**:

Software DIA (Elaboración propia)

Como resultado del diseño LÓGICO obtuvimos el siguiente **modelo relacional**:

Libre Office (Elaboración propia)

## Restricciones para diseño físico

Para realizar el diseño físico y su implantación en MySQL nos dan las siguientes especificaciones y restricciones. Si en alguna tabla no se especifica de forma explícita alguna restricción y/o tipo de dato de ciertas columnas, es porque se debe deducir del propio modelo relacional y/o valor que va a almacenar.

Tabla **GENERO**.

- ✓ El código es alfanumérico de tamaño fijo y de 4 caracteres. El nombre como máximo de 20 caracteres. La descripción puede ocupar mas de 300 caracteres.

Tabla **PELICULA**.

- ✓ El código es un valor numérico que se incrementa de forma automática. El título tiene como máximo 30 caracteres y es obligatorio (no puede quedar vacío).

Tabla **ACTOR**.

- ✓ El código es alfanumérico de tamaño fijo de 5 caracteres. EL `<abbr title="Documento Nacional de Identidad">DNI</abbr>` es alfanumérico de tamaño fijo de 10 caracteres. (es clave alternativa).
- ✓ El nombre es de tamaño variable, como máximo de 20 caracteres.

Tabla **EMAIL**.

- ✓ El <span lang="en">email </span>es alfanumérico de tamaño variable, máximo 40 caracteres.

#### Tabla OSCARIZADO.

- ✓ El total oscar es un entero pequeño.

#### Tabla PRODUCTOR.

- ✓ El número de películas es una entero. El importe mínimo puede tener parte decimal.

#### RESTRICCIONES DE CLAVE AJENA para borrados y modificaciones.

- ✓ Al modificar la clave primaria de cualquier tabla principal se deben modificar de igual forma las claves ajenas o foráneas relacionadas.
- ✓ No se puede eliminar un género si tiene películas relacionadas.
- ✓ No se puede eliminar una película en la que participen actores.
- ✓ No se puede eliminar un actor que participe en películas.
- ✓ Si se elimina un actor, se deben eliminar todos sus email relacionados y sus particularidades como oscarizado y/o productor.

## Creación de base de datos y tablas en SQL

Redactamos las sentencias SQL para crear la base de datos y cada tabla desde la línea de comandos o bien en la ventana SQL de la herramienta o cliente Workbench.

Es IMPORTANTE el **orden de creación de tablas**. Podemos seguir dos criterios de creación:

- 1.- Crear primero las tablas que no tienen claves foráneas (no referencian a otras tablas), pues no se puede crear una tabla T1 que referencia a la tabla T2, si no se ha creado antes la tabla T2.
- 2.- Crear todas las tablas y después con ALTER añadir la restricción de claves ajenas o foráneas.

- ✓ En este caso vamos a seguir la estrategia 1.- Crear primero las tablas que no tienen claves foráneas y después ir creando las tablas que referencian con sus claves foráneas a tablas que ya hemos creado previamente.

En el siguiente enlace tienes el script SQL de creación de la base de datos películas.

[Script SQL de creación de la base de datos Peliculas](#) (zip - 740 B)

## Ejemplo 2. Implantación física de la base de datos VIDEOTECA.

[Implantación de BD Videoteca](#)    [Enunciado](#)    [Diagrama ERE y Modelo Relacional](#)

[Restricciones para diseño físico](#)    [Creación de base de datos y tablas en SQL.](#)

## Implantación de BD Videoteca

En el siguiente videotutorial puedes ver cómo se realiza la implantación física de la base de datos VIDEOTECA en MySQL, que posteriormente viene detallada en las otras secciones.

En esta ocasión:

- ✓ Le vamos a dar un nombre personalizado a las restricciones o CONSTRAINT a nivel de tabla.
- ✓ Realizamos ciertas modificaciones sobre algunas tablas con ALTER TABLE y añadiendo índices con INDEX.
- ✓ Crearemos un par de vistas, VIEW, para resolver determinadas consultas.

### Creación de la base de datos Videoteca, constraint personalizadas y VIEW

<https://www.youtube.com/embed/YCKINlzzPjM>

## Enunciado

Las especificaciones y requerimientos de la base de datos Videoteca son los siguientes

- 1.- Una película se caracteriza por un código, título, año de estreno y género.
- 2.- Una película pertenece obligatoriamente a un género y solo a uno. Pero de un determinado género puede haber varias películas o ninguna.
- 3.- El género de una película se caracteriza por un código, nombre (que es único) y su descripción.
- 4.- Todas las películas tienen una o varias copias y pueden ser alquiladas. Una copia es de una y solo una película. Si se descataloga la película no tiene sentido mantener la copia.
- 5.- Las copias de cada película se identifican con un número sucesivo (1, 2, 3, 4 ...) y de ellas se guarda la fecha de compra y estado de conservación.
- 6.- Las copias de la películas se alquilan a clientes. De los clientes interesa su DNI, nombre, fecha\_nacimiento, edad y varios email.
- 7.- Los clientes puedes ser de dos tipos:
  - 7.1.- Socios: de ellos se guarda además un código, fecha de alta, descuento y cuota.
  - 7.2.- Eventuales: de estos clientes se guarda la fecha de su primer alquiler.
- 8.- Tanto los clientes socios como los eventuales han alquilado al menos una copia, pudiendo alquilar varias, y cada copia puede haber sido alquilada varias veces o ninguna.
- 9.- Al alquilar una copia interesa saber la fecha de alquiler y precio de alquiler.
- 10.- Los clientes eventuales pueden haber sido invitados por un único cliente eventual, y un cliente eventual puede invitar a varios cliente eventuales o a ninguno.

## Diagrama ERE y Modelo Relacional

Como resultado del diseño CONCEPTUAL obtuvimos el siguiente **diagrama Entidad/Relación Extendido**:

Software DIA (Elaboración propia)

Como resultado del diseño LÓGICO obtuvimos el siguiente **modelo relacional**:

## Restricciones para diseño físico

Para realizar el diseño físico y su implantación en MySQL nos dan las siguientes especificaciones y restricciones. Si en alguna tabla no se especifica de forma explícita alguna restricción y/o tipo de dato de ciertas columnas, es porque se debe deducir del propio modelo relacional y/o valor que va a almacenar.

### Tabla GENERO.

- ✓ El código es alfanumérico de tamaño fijo y de 4 caracteres. El nombre como máximo de 20 caracteres, y no se puede repetir. La descripción puede ocupar mas de 300 caracteres.

### Tabla PELICULA.

- ✓ El código es alfanumérico de tamaño fijo y de 5 caracteres. El título tiene como máximo 30 caracteres y es obligatorio (no puede quedar vacío).

### Tabla COPIA.

- ✓ El num\_copia es un entero. El estado solo puede tomar los siguientes valores ('malo',' regular', 'bueno','excelente').

### Tabla CLIENTE.

- ✓ El DNI es alfanumérico de tamaño fijo. El nombre como máximo de 60 caracteres.

### Tabla EMAIL.

- ✓ El <span lang="en">email </span>es alfanumérico de tamaño variable, máximo 40 caracteres.

### Tabla se\_alquila.

- ✓ El importe puede tener parte decimal, no puede ser nulo y por defecto tiene el valor 2,5.

### Tabla SOCIO.

- ✓ El código es alfanumérico de tamaño fijo 5 caracteres, obligatorio y no se puede repetir. El descuento puede tener parte decimal.
- ✓ La cuota es un valor entero, obligatoria y por defecto tiene el valor 10.

### RESTRICCIONES DE CLAVE AJENA para borrados y modificaciones.

- ✓ Al modificar la clave primaria de cualquier tabla principal se deben modificar de igual forma las claves ajenas o foráneas relacionadas.
- ✓ No se puede eliminar un género si tiene películas relacionadas.
- ✓ Si se elimina una película se eliminan sus copias relacionadas.
- ✓ No se puede eliminar una copia que esté alquilada.
- ✓ No se puede eliminar un cliente con copias alquiladas.

- ✓ Si se elimina un cliente, se deben eliminar todos sus email relacionados y sus particularidades como socio y/o eventual.
- ✓ Si se elimina un eventual que es el que invitó, los invitados por él quedan sin valor en la columna dni\_invita.

## Creación de base de datos y tablas en SQL.

Redactamos las sentencias SQL para crear la base de datos y cada tabla desde la línea de comandos o bien en la ventana SQL de la herramienta o cliente Workbench.

Es IMPORTANTE el **orden de creación de tablas**. Podemos seguir dos criterios de creación:

- 1.- Crear primero las tablas que no tienen claves foráneas (no referencian a otras tablas), pues no se puede crear una tabla T1 que referencia a la tabla T2, si no se ha creado antes la tabla T2.
- 2.- Crear todas las tablas y después con ALTER añadir la restricción de claves ajenas o foráneas.

- ✓ En este caso vamos a seguir la **estrategia 1**.- Crear primero las tablas que no tienen claves foráneas y después ir creando las tablas que referencian con sus claves foráneas a tablas que ya hemos creado previamente.
- ✓ Además vamos a dar nombre personalizado a las restricciones o CONSTRAINT.

[Script SQL de creación de la base de datos videoteca](#) (zip - 771 B)

### Ejemplo 3. Implantación física de la base de datos CAMPEONATO.

[Implantación BD Campeonato](#)    [Enunciado](#)    [Diagrama ERE y Modelo Relacional](#)

[Restricciones para diseño físico](#)    [Creación de base de datos y tablas en SQL](#)

## Implantación BD Campeonato

En el siguiente videotutorial puedes ver cómo se realiza la implantación física de la base de datos CAMPEONATO en MySQL, que posteriormente viene detallada en las otras secciones.

- ✓ En este caso vamos a crear todas las tablas sin ninguna restricción de clave foránea y después con ALTER TABLE **añadiremos esas restricciones de FOREIGN KEY**.
- ✓ Veremos como **crear y modificar tablas en modo gráfico**.
- ✓ Una vez implantada la base de datos CAMPEONATO **obtendremos su modelo mediante ingeniería inversa**.

### Creación de la base de datos campeonato, añadir después claves foráneas y obtener el modelo con ingeniería inversa

[https://www.youtube.com/embed/h\\_z3ZVGzF9g](https://www.youtube.com/embed/h_z3ZVGzF9g)

Isabel Cruz [Descripción textual alternativa del video](#) (Dominio público)

## Enunciado

La Asociación Andaluza de Videojuegos o e-Sports necesita gestionar diferentes campeonatos de juegos online mediante una base de datos que almacene información sobre los concursantes, sus equipos y los diferentes juegos en los que participan los concursantes.

Se debe realizar el diseño de una base de datos relacional a partir de las siguientes especificaciones o requisitos que debe cumplir:

- 1.- De los concursantes se desea almacenar un código, su nombre, sexo, fecha de inscripción y cuota de inscripción.
- 2.- Cada concursante pertenece obligatoriamente a un equipo y solo a uno. Pero en un equipo puede haber ninguno o varios concursantes.
- 3.- Cada equipo se caracteriza por un código, nombre, comunidad y año de fundación.
- 4.- Los equipos, no todos, pueden organizar varios juegos. Cada juego es organizado por uno y solo un equipo.
- 5.- Los juegos se caracterizan por su código, nombre, nivel de dificultad y número de *megusta*.
- 6.- Los concursantes pueden participar en ninguno o varios juegos, independientemente del equipo al que pertenezcan.
- 7.- En un juego pueden participar ninguno o varios concursantes.
- 8.- Cuando un concursante participa en un juego, se anota la fecha de comienzo en la que inició ese juego y se guardan también los puntos que va acumulando en ese juego.
- 9.- Los concursantes pueden designar a otro concursante como su ídolo, de manera que un concursante puede ser ídolo de ninguno o varios concursantes, pero como ídolo solo puede tener a un único concursante o a ninguno.

## Diagrama ERE y Modelo Relacional

Como resultado del diseño CONCEPTUAL obtuvimos el siguiente **diagrama Entidad/Relación**:

Software DIA (Elaboración propia)

Como resultado del diseño LÓGICO obtuvimos el siguiente **modelo relacional**:

Donde en el modelo textual se ha indicado la restricción de cada clave ajena o foránea en cuanto a los borrados y las modificaciones:

- ✓ BR: borrado restrictivo, BN: borrado con puesta a nulos, BC: borrado en cascada.
- ✓ MC: modificación en cascada.

## Restricciones para diseño físico

Para realizar el diseño físico y su implantación en MySQL nos dan las siguientes especificaciones y restricciones. Si en alguna tabla no se especifica de forma explícita alguna restricción y/o tipo de dato de ciertas columnas, es porque se debe deducir del propio modelo relacional y/o valor que va a almacenar.

### Tabla CONCURSANTE.

- ✓ El código es alfanumérico de tamaño fijo y de 3 caracteres. El nombre como máximo de 30 caracteres y obligatorio. La cuota de inscripción puede llevar parte decimal.
- ✓ El sexo solo puede tomar los valores 'H' o 'M' y es obligatorio.

### Tabla EQUIPO.

- ✓ El código es un valor alfanumérico de 2 caracteres. El nombre es obligatorio y de una máxima de 50 caracteres alfanuméricos. La comunidad como máximo de 20 caracteres alfanuméricos.

### Tabla JUEGO.

- ✓ El código es alfanumérico de tamaño fijo de 3 caracteres.
- ✓ El nombre es de tamaño variable, como máximo de 40 caracteres, obligatorio y no se puede repetir.
- ✓ La dificultad es obligatoria y solo puede tomar los siguientes valores: 'alta', 'media', 'baja'.
- ✓ La columna megusta es de tipo entero y no puede ser negativa.

### Tabla participa.

- ✓ Los puntos es una columna numérica, no puede ser negativa, es obligatoria y con valor por defecto 0.

### RESTRICCIONES DE CLAVE AJENA para borrados y modificaciones.

- ✓ Al modificar la clave primaria de cualquier tabla principal se deben modificar de igual forma las claves ajena o foráneas relacionadas.
- ✓ No se puede eliminar un equipo si tiene concursantes relacionados.
- ✓ No se puede eliminar un equipo si tiene juegos relacionados.
- ✓ Si se elimina un concursante se eliminan las filas relacionadas en la tabla participa.
- ✓ Si se elimina un concursante que es ídolo, los concursantes que lo tenían como ídolo quedan sin ídolo.
- ✓ Si se elimina un juego, se eliminan las filas relacionadas en la tabla participa.

## Creación de base de datos y tablas en SQL

Redactamos las sentencias SQL para crear la base de datos y cada tabla desde la línea de comandos o bien en la ventana SQL de la herramienta o cliente Workbench.

Es IMPORTANTE el **orden de creación de tablas**. Podemos seguir dos criterios de creación:

- 1.- Crear primero las tablas que no tienen claves foráneas (no referencian a otras tablas), pues no se puede crear una tabla T1 que referencia a la tabla T2, si no se ha creado antes la tabla T2.
- 2.- Crear todas las tablas y después con ALTER TABLE añadir la restricción de claves ajenas o foráneas.

- ✓ En este caso vamos a seguir **la estrategia 2.**- Crear todas las tablas, en cualquier orden, y después con ALTER TABLE añadir la restricción de clave ajena o foránea a las tablas que corresponda.

En el siguiente enlace puedes descargar el script SQL de creación de la base de datos campeonato.

[Script SQL de creación de la base de datos campeonato](#) (zip - 847 B)

# 10.- Creación, modificación y eliminación de índices.

## Caso práctico

Con las sentencias vistas hasta ahora **Noiba** y **Vindio** ya conocen todas las **instrucciones necesarias para añadir, modificar o eliminar todo tipo de índices**. Sin embargo, **Juan** les aconseja que consulten la documentación del lenguaje **SQL**, y revisen si es necesario añadir índices a alguna de las tablas con el propósito de optimizar las consultas.

Considera pues que es importante conocer la sentencia que permite crear índices y también aprender algo más sobre la utilidad de los índices en una base de datos.



[Jonny Goldstein \(CC BY\)](#)

Un índice permite consultar las filas de una tabla de una forma más rápida.

Sabemos que los índices ayudan a la localización más rápida de la información contenida en las tablas. Pero, ¿dónde se almacenan? ¿debemos tener muchos índices en las tablas?

Vamos a verlo:

- ✓ **Almacenamiento:** La información sobre los índices se almacena en una tabla de índices de tamaño mucho más reducido que la tabla de datos, ordenados de forma ascendente junto con información relativa a donde están almacenados los datos correspondientes a cada valor del índice.
- ✓ **Acceso:** El usuario no puede acceder a la tabla de índices aunque siempre que realiza una operación condicionada a los valores de un índice, el servidor MySQL busca primero en la tabla de índices para, a través de la información almacenada en el índice, acceder de forma directa a los datos correspondientes de la tabla de datos.
- ✓ **Ventajas y desventajas:** Si bien un índice permite consultar de forma más rápida la tabla, las operaciones de añadir filas, modificar filas o eliminar filas consumen más tiempo. Además, los datos de una tabla en la que se definen índices ocupan más espacio que el dedicado exclusivamente a esos datos, puesto que necesita un espacio adicional para almacenar una tabla de índices. Por ello, no es aconsejable definir índices de manera indiscriminada. Lo que se debe hacer es definir índices sobre columnas a partir de las cuales se vayan a hacer consultas frecuentes.

### Creación de índices

Veamos ahora la sentencia **SQL** para crear índices, que es **CREATE INDEX**.

#### CREATE INDEX:

**Permite añadir índices a tablas existentes.** Un índice, al ser creado, puede recibir un nombre. Si no se especifica ese nombre, el índice recibe el nombre de la primera columna que forma ese índice. Un índice se puede definir sobre una columna o sobre un grupo de columnas. Para los índices formados por columnas de tipo CHAR o VARCHAR se puede especificar que sólo los primeros caracteres de esas columnas formen el índice con un número entre paréntesis (longitud).

La sintaxis es:

```
CREATE [UNIQUE | FULLTEXT] INDEX NombreIndice ON NombreTabla(NombreColumna[(longitud)]  
[ASC|DESC],....);
```

Donde las opciones son:

- ✓ **UNIQUE | FULLTEXT** : Si no se especifica UNIQUE o FULLTEXT el índice admite valores duplicados. En caso de que sea UNIQUE los valores del índice no podrán repetirse. El índice de tipo FULLTEXT sirve para buscar palabras dentro del texto contenido en las columnas correspondientes. Un índice FULLTEXT sólo se admite sobre columnas CHAR, VARCHAR o cualquiera de los tipos TEXT.
- ✓ **ASC | DESC**: Los índices pueden ordenar los datos de forma ascendente (ASC) o descendente (DESC).

**Ejemplo:** Crear un índice para los 10 primeros caracteres de la columna Apellidos de la tabla Clientes.

```
CREATE UNIQUE INDEX Un_1 ON Clientes (Apellidos(10));
```

Además de con la sentencia CREATE INDEX, en MySQL también podemos crear índices de la siguientes maneras:

- ✓ Al mismo tiempo que creamos la tabla con el uso de la opción INDEX.

```
CREATE TABLE CLIENTES (
CodCliente VARCHAR(5) NOT NULL,
DNI VARCHAR(10) NOT NULL,
Apellidos VARCHAR(50),
Nombre VARCHAR(25),
Direccion VARCHAR(50),
Telefono VARCHAR(9),
PRIMARY KEY (CodCliente),
INDEX ind_ape (Apellidos)
)ENGINE=InnoDB;
```

- ✓ Con la sentencia ALTER TABLE si es que la tabla ya existe, mediante ADD INDEX.

```
ALTER TABLE Clientes
ADD INDEX ind_ape (Apellidos);
```

### Eliminación y modificación de índices

Para eliminar los índices se utiliza la sentencia DROP INDEX.

**DROP INDEX:** Elimina un índice de una tabla,

La sintaxis es:

```
DROP INDEX NombreIndice ON NombreTabla;
```

Por ejemplo, para eliminar el índice creado sobre la columna Apellidos de la tabla Clientes, sería:

```
DROP INDEX ind_ape ON Clientes;
```

En MySQL se crean **de forma implícita** índices sobre las columnas con las restricciones de PRIMARY KEY, FOREIGN KEY o UNIQUE.

## Autoevaluación

Relaciona las instrucciones que tengan la misma función con respecto a los índices:

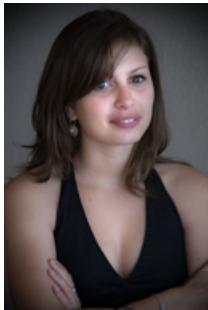
## Ejercicio de relacionar

Instrucciones	Relacionar	Indices
ALTER TABLE.....ADD UNIQUE...	0	1. CREATE INDEX...
ALTER TABLE... DROP INDEX.....	0	2. CREATE UNIQUE INDEX
ALTER TABLE.....ADD INDEX....	0	3. DROP INDEX.....

Se pueden crear y borrar índices de varias formas.

# 11.- Creación, modificación y eliminación de vistas.

## Caso práctico



Alain Bachellier (CC BY-NC-SA)

Aunque no se trata de instrucciones propias del LDD, **Noiba** se plantea que quizás le sea útil en su trabajo con la base de datos TalleresFaber estudiar ahora el **funcionamiento de las vistas**. Se trata de consultas almacenadas que trataremos igual que si fueran tablas. La ventaja que ofrecen las vistas para **Noiba** es que se podrá **mostrar los datos que el usuario pueda manejar como si fuera una tabla** y ocultarle el resto. Esto puede ser importante para que no todos los empleados del taller tengan acceso a toda la información almacenada.

Una vista es sinónimo de una consulta almacenada en MySQL como una tabla virtual, que permite acceder a los datos que se obtienen como resultado de una consulta determinada. Una vez creada una vista, tiene un comportamiento similar al de una tabla, ya que se pueden realizar acciones como consultar datos sobre la vista. El usuario maneja aparentemente datos de la vista, pero realmente está manejando datos de las tablas de cuya consulta se ha obtenido la vista.

Los datos de una vista, como se ha dicho, no existen realmente. Cuando se hace una consulta sobre una vista, el servidor de base de datos realiza la consulta SELECT que crea la vista primero y, sobre los datos obtenidos (la vista) obtiene el resultado de la consulta que se ha hecho.

Las vistas se usan para poder evitar que determinados usuarios accedan a ciertas tablas, dándoles acceso solamente a una parte de esas tablas a través de las vistas.

Las sentencias SQL para manejar vistas son:

- ✓ CREATE VIEW
- ✓ ALTER VIEW
- ✓ DROP VIEW

Vamos a verlas.

### Crear vistas

<b>CREATE VIEW</b>

Crea una vista a partir de una consulta que se hace a una o más tablas existentes. La sintaxis es:

```
CREATE [OR REPLACE] VIEW NombreVista [(NombreColumna1, NombreColumna2,...)] AS SELECT
```

.....

Analizaremos ahora la primera parte de esta sentencia, ya que la segunda parte incluye una consulta con la sentencia SELECT, que veremos en la siguiente unidad:

- ✓ Los nombres que le demos a las columnas: NombreColumna1, NombreColumna2, ..., no tienen por qué coincidir con los nombres que tenían en las tablas originales, pero sí tiene que haber el mismo número de columnas devueltas en la sentencia <b>SELECT</b> que en la vista. Si no se especifica ninguna columna, las columnas creadas para la vista son las que se obtienen en la consulta.

- ✓ Las vistas no pueden contener subconsultas ni variables de usuario o del sistema.
- ✓ La vista no puede tener el mismo nombre que una tabla existente.
- ✓ Si una vista está basada en una sola tabla, se pueden añadir, modificar o eliminar los datos de la tabla de la que se obtiene la vista añadiendo, modificando o eliminando filas en la vista. La vista realmente no se actualiza porque es virtual.

## Modificar Vistas

ALTER VIEW

Modifica la definición de una vista existente. Es una sentencia similar a CREATE VIEW.

```
ALTER [OR REPLACE] VIEW NombreVista [(NombreColumna1, NombreColumna2,...)] AS SELECT .....
```

## Eliminar Vistas

<b>DROP VIEW</b>

Elimina una o más vistas de la base de datos.

```
DROP VIEW [IF EXISTS] NombreVista1 [, NombreVista2],...
```

## Ejercicio resuelto

Crear una vista llamada **Reparados** que muestre el nombre y apellidos de los clientes, marca y modelo de los automóviles, y las averías de los vehículos que han sido reparados.

[Mostrar retroalimentación](#)

```
CREATE VIEW Reparados (Nombre, Apellidos, Marca, Modelo, Averia) AS
SELECT Nombre, Apellidos, Marca, Modelo, Averia
FROM CLIENTES, REPARACIONES, VEHICULOS
WHERE CLIENTES.CodCliente= VEHICULOS.CodCliente
AND VEHICULOS.Matricula=REPARACIONES.Matricula AND reparado;
```

## 12.- Otras sentencias útiles.

### Caso práctico



Con **Vindio y Noiba** hemos recorrido todas las **sentencias que podemos necesitar para realizar el diseño físico** de una base de datos. Pero aún nos queda conocer una serie de sentencias que no corresponden al **LLD** pero que resultan muy útiles a la hora de trabajar con **SQL**. Muchas veces resulta más fácil recurrir a ellas que consultar la información desde las opciones que ofrece la herramienta gráfica del **SGBD**. Veremos a continuación cuáles son esas sentencias y cómo se utilizan.

[Alain Bachelier \(CC BY-NC-SA\)](#)

Para mostrar los recursos disponibles en una base de datos podemos utilizar la sentencia SHOW.

SHOW es una sentencia que permite mostrar cierta información de la base de datos.

Algunos de los casos y usos de SHOW son los siguientes:

- ✓ **SHOW DATABASES:** Muestra todas las bases de datos.
- ✓ **SHOW TABLES:** Muestra todas las tablas de la base de datos activa.

**DESCRIBE, SHOW COLUMNS, SHOW FIELDS:** Muestra las columnas de una tabla. De varias formas:

- ✓ <b>DESCRIBE</b>NombreTabla;
- ✓ <b>SHOW COLUMNS FROM</b> NombreTabla;
- ✓ <b>SHOW FIELDS FROM</b> NombreTabla;

**SHOW CREATE:** Muestra la sentencia SQL con la que se crea un objeto. Presenta varias opciones:

- ✓ <b>SHOW CREATE DATABASE;</b>
- ✓ <b>SHOW CREATE TABLE</b> NombreTabla;
- ✓ <b>SHOW CREATE VIEW</b> NombreVista;

<b>SHOW INDEX:</b> Muestra los índices de una tabla.

- ✓ <b>SHOW INDEX FROM</b> NombreTabla;

<b>SHOW GLOBAL VARIABLES:</b> Muestra las variables globales.

**SHOW SESSION VARIABLES:** Muestra las variables de sesión.

### Ejercicio resuelto

Realiza las siguientes consultas en SQL:

1. Muestra todas las bases de datos de Mysql.
2. Selecciona la base de datos TalleresFaber como base de datos activa.
3. Muestra las tablas de la base de datos.
4. Muestra las columnas de la tabla Clientes.
5. Muestra los índices de la tabla Clientes.

[Mostrar retroalimentación](#)

```
-- 1.-  
SHOW DATABASES;  
-- 2.-  
USE TalleresFaber;  
-- 3.-  
SHOW TABLES;  
-- 4.- Serían válidas cualquiera de las tres siguientes  
SHOW COLUMNS FROM Clientes;  
DESCRIBE Clientes;  
SHOW FIELDS FROM Clientes;  
-- 5.-  
SHOW INDEX FROM Clientes;
```

# 13.- Documentación del diseño: el diccionario de datos.

## Caso práctico

Finalmente **Noiba** y **Vindio** ya tienen el diseño físico de la base de datos que recoge la actividad del taller mecánico. Antes de dar por concluido el trabajo de diseño e implantación, **Juan** les indica que deben confeccionar el **diccionario de datos**. En él se hace una **descripción detallada de todos los elementos que intervienen en el sistema** y nos servirá para tener una visión global de la base de datos, sin ambigüedad. Esta información puede ser útil tanto para **Noiba**, como para otras personas que se incorporen al proyecto.

Afortunadamente la mayoría de los **SGBD** incorporan utilidades o herramientas que confeccionan este diccionario automáticamente.



[Jonny Goldstein \(CC BY\)](#)

En **MySQL** esta información se puede obtener desde Workbench y nos muestra en un informe la descripción de cada objeto de la base de datos y sus propiedades. Este informe está formado por las tablas, vistas y sus descripciones, permitiendo saber:

- ✓ Estructura lógica y física de la base de datos
- ✓ Los usuarios de la base de datos.
- ✓ Restricciones de integridad sobre las tablas de la base de datos.
- ✓ Espacio asociado a cada objeto en la base de datos.

## Debes conocer

Consulta la siguiente página y siguiendo los pasos indicados allí, aprende a generar el diccionario de datos en MySQL con la herramienta Workbench.

[Obtener el diccionario de datos en MySQL con Workbench](#)

En **MySQL** la información relativa a los metadatos de la base de datos se encuentra en **<INFORMATION\_SCHEMA>**. Se trata de una **base de datos que almacena información acerca de todas las otras bases de datos** que tengamos en nuestro servidor. Cada usuario tiene derecho a acceder a estas tablas, pero sólo a los registros que se refieren a los objetos a los que tiene permiso de acceso. Se puede acceder a su contenido con **<SELECT>** pero no se puede insertar, actualizar o borrar.

## Para saber más

Si quieres conocer más sobre la información contenida en las tablas de **INFORMATION\_SCHEMA** consulta el siguiente enlace:

[Información sobre las tablas INFORMATION\\_SCHEMA en MySQL](#)

# Condiciones y términos de uso de los materiales

Materiales desarrollados inicialmente por el Ministerio de Educación, Cultura y Deporte y actualizados por el profesorado de la Junta de Andalucía bajo licencia Creative Commons BY-NC-SA.

Antes de cualquier uso leer detenidamente el siguiente [Aviso legal](#)

## Historial de actualizaciones

Versión: 02.00.01	Fecha de actualización: 11/11/21	
Actualización de materiales y correcciones menores.		
Versión: 02.00.00	Fecha de actualización: 24/05/21	Autoría: Isabel Cruz Granados
<b>Ubicación:</b> Apartado 9.4 <b>Mejora (tipo 3):</b> En el apartado 9.4 se puede cambiar el nombre del epígrafe para que sea: 9.4.- Casos resueltos, e incluir los mismos ejemplos que se han visto en los tutoriales propuestos de la unidad 2 (punto 4.2 y 6.) finalizando así las fases del diseño de una base de datos con la implantación física de los ejemplos desarrollados anteriormente. <b>Ubicación:</b> Ningún cambio <b>Mejora (Mapa conceptual):</b> Ningún cambio <b>Ubicación:</b> Apartado 9.4 <b>Mejora (Orientaciones del alumnado):</b> Cambio del título		
Versión: 01.00.00	Fecha de actualización: 23/07/20	
Versión inicial de los materiales.		

