

Aprendamos MySQL

Introducción a MySQL
Arquitectura de MySQL
Instalación
Primeros pasos

Roberto Cruzado

Upgrade-hub
FullStack Developer
10 Weeks Bootcamp

¿Qué es MySQL 8?

MySQL es un
sistema de gestión
de bases de datos
relacional.

SQL (lenguaje de consultas estructuradas).

SQL (Structured Query Language) es un lenguaje de programación estándar e interactivo para la obtención de información desde una base de datos y para actualizarla

Los comandos de SQL se utilizan tanto para consultas interactivas para obtener información de una base de datos relacional y para la recopilación de datos para los informes.

Las consultas toman la forma de un lenguaje de comandos que permite seleccionar, insertar, actualizar, averiguar la ubicación de los datos, y más.

El gestor de bases de datos más popular

MySQL 8 es la última versión de uno de los sistemas de gestión de bases de datos relacionales más populares del mundo.

Utilizada mayormente por aplicaciones extensamente utilizadas como Wordpress o Joomla

MySQL

<http://www.mysql.com>

PostgreSQL

Similar a MySQL, pero no tan popular
<http://www.postgresql.org>

Microsoft SQL Server

Base de datos comercial que requiere
licencia para su uso

<http://www.microsoft.com/sql-server>

Oracle Database

También es comercial y requiere licencia
para su uso

<http://www.oracle.com>

<http://www.mysql.com>

Punto de partida para la instalación
Extensiva documentación
Community Edition es la versión gratuita

Práctica

Instalación de MySQL en nuestros entornos
<https://dev.mysql.com/downloads/mysql/>

Windows

MacOS

Linux

Yum/APT

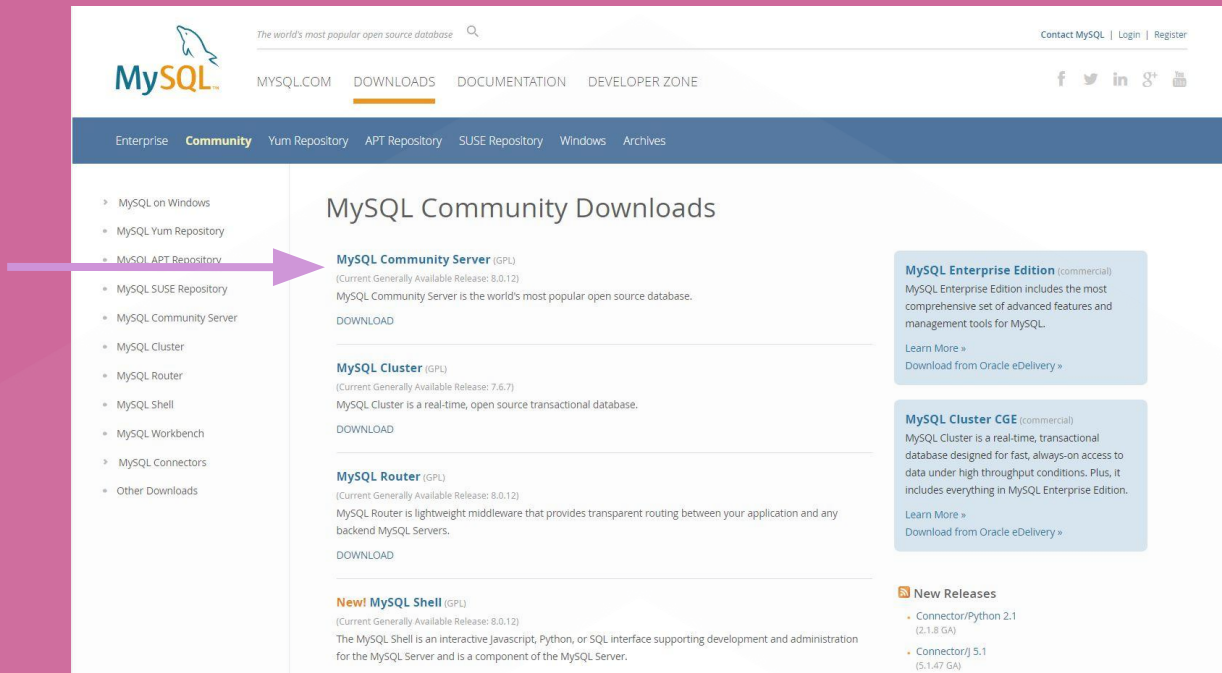
Docker

https://hub.docker.com/_/mysql/

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


¿Qué es MySQL 8?

<https://dev.mysql.com/downloads/>



The screenshot shows the MySQL Community Downloads page. The left sidebar contains a list of download options, with a purple arrow pointing to "MySQL Community Server". The main content area displays details for the MySQL Community Server, including its current release version (8.0.12) and a "DOWNLOAD" link. Other options like MySQL Cluster, MySQL Router, and MySQL Shell are also listed. On the right, there are sections for MySQL Enterprise Edition and MySQL Cluster CGE, both with "DOWNLOAD" links. At the bottom right, there is a "New Releases" section listing Connector/Python 2.1 and Connector/J 5.1.

MySQL Community Downloads

- MySQL on Windows
- MySQL Yum Repository
- MySQL APT Repository
- MySQL SUSE Repository
- MySQL Community Server
- MySQL Cluster
- MySQL Router
- MySQL Shell
- MySQL Workbench
- MySQL Connectors
- Other Downloads

MySQL Community Server (GPL)
(Current Generally Available Release: 8.0.12)
MySQL Community Server is the world's most popular open source database.
[DOWNLOAD](#)

MySQL Cluster (GPL)
(Current Generally Available Release: 7.6.7)
MySQL Cluster is a real-time, open source transactional database.
[DOWNLOAD](#)

MySQL Router (GPL)
(Current Generally Available Release: 8.0.12)
MySQL Router is lightweight middleware that provides transparent routing between your application and any backend MySQL Servers.
[DOWNLOAD](#)

New! MySQL Shell (GPL)
(Current Generally Available Release: 8.0.12)
The MySQL Shell is an interactive Javascript, Python, or SQL interface supporting development and administration for the MySQL Server and is a component of the MySQL Server.

MySQL Enterprise Edition (commercial)
MySQL Enterprise Edition includes the most comprehensive set of advanced features and management tools for MySQL.
[Learn More »](#)
[Download from Oracle eDelivery »](#)

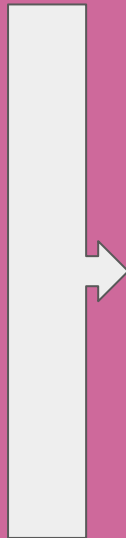
MySQL Cluster CGE (commercial)
MySQL Cluster is a real-time, transactional database designed for fast, always-on access to data under high throughput conditions. Plus, it includes everything in MySQL Enterprise Edition.
[Learn More »](#)
[Download from Oracle eDelivery »](#)

New Releases

- Connector/Python 2.1 (2.1.8 GA)
- Connector/J 5.1 (5.1.47 GA)

Mostrar Bases de Datos y Tablas existentes

- 1.- Buscar el archivo ejecutable, MySQL 8.0 Command Line Client.
- 2.- Ingresar con el password (normalmente: admin).
- 3.- verificar que el prompt es: Mysql>



```
mysql> show databases;  
  
mysql> use NombreBBDD;  
  
mysql> show tables;  
  
mysql> describe NombreTabla;
```

Incluir Base de Datos WORLD

(<https://dev.mysql.com/doc/index-other.html>)

1. **Extraer el archivo de instalación a una ubicación temporal como `C:\temp\o` `/tmp/`. Desempaquetar el archivo, en un sólo documento que se llame: `world.sql`.**

2. **Situarse en la línea de comandos del cliente MySQL:**

```
shell> mysql -u root -p
```

3. **Ejecutaremos el script `world.sql` para crear la estructura de la base de datos, de la siguiente forma:**

```
mysql> SOURCE C:/temp/world.sql;
```

(Reemplazar `C:/temp/` con el path en donde tengamos descargado el fichero `world.sql`)

4. **Confirmar que hemos instalado correctamente esta base de datos de ejemplo con los siguientes comandos...**

Incluir Base de Datos WORLD (comprobación)

```
mysql> USE world;
```

Database changed

```
mysql> SHOW TABLES;
```

```
+-----+  
| Tables_in_world |  
+-----+  
| city             |  
| country          |  
| countrylanguage |  
+-----+
```

```
mysql> SELECT COUNT(*) FROM city;
```

```
+-----+  
| COUNT(*) |  
+-----+  
| 4079     |  
+-----+
```

```
mysql> SELECT COUNT(*) FROM country;
```

```
+-----+  
| COUNT(*) |  
+-----+  
| 239      |  
+-----+
```

Descripción de las tablas de la base de datos WORLD

```
+-----+
| Tables_in_world |
+-----+
| City             |
| Country          |
| CountryLanguage  |
+-----+
3 rows in set (0.00 sec)
```

```
mysql> describe City;
```

```
+-----+-----+-----+-----+-----+-----+
| Field      | Type      | Null | Key | Default | Extra      |
+-----+-----+-----+-----+-----+-----+
| ID         | int(11)   | NO   | PRI | NULL    | auto_increment |
| Name       | char(35)  | NO   |     |         |              |
| CountryCode | char(3)   | NO   | MUL |         |              |
| District   | char(20)  | NO   |     |         |              |
| Population | int(11)   | NO   |     | 0       |              |
+-----+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)
```

```
mysql> describe Country;
```

```
+-----+-----+-----+-----+-----+-----+
| Field      | Type                                             | Null | Key | Default | Extra      |
+-----+-----+-----+-----+-----+-----+
| Code       | char(3)                                         | NO   | PRI |         |              |
| Name       | char(52)                                       | NO   |     |         |              |
| Continent  | enum('Asia','Europe','North America','Africa','Oceania','Antarctica','South America') | NO   |     |         |              |
| Region     | char(26)                                       | NO   |     |         |              |
| SurfaceArea | float(10,2)                                    | NO   |     | 0.00    |              |
| IndepYear  | smallint(6)                                    | YES  |     | NULL    |              |
| Population | int(11)                                         | NO   |     | 0       |              |
| LifeExpectancy | float(3,1)                                    | YES  |     | NULL    |              |
| GNP        | float(10,2)                                    | YES  |     | NULL    |              |
| GNPOld     | float(10,2)                                    | YES  |     | NULL    |              |
| LocalName  | char(45)                                       | NO   |     |         |              |
| GovernmentForm | char(45)                                       | NO   |     |         |              |
| HeadOfState | char(60)                                       | YES  |     | NULL    |              |
| Capital    | int(11)                                         | YES  |     | NULL    |              |
| Code2      | char(2)                                         | NO   |     |         |              |
+-----+-----+-----+-----+-----+-----+
15 rows in set (0.00 sec)
```

```
mysql> describe CountryLanguage;
```

```
+-----+-----+-----+-----+-----+-----+
| Field      | Type      | Null | Key | Default | Extra      |
+-----+-----+-----+-----+-----+-----+
| CountryCode | char(3)   | NO   | PRI |         |              |
| Language    | char(30)  | NO   | PRI |         |              |
| IsOfficial  | enum('T','F') | NO   |     | F       |              |
| Percentage  | float(4,1) | NO   |     | 0.0     |              |
+-----+-----+-----+-----+-----+-----+
```

PRÁCTICA de EJEMPLO sobre la tabla CITY

1. Ver estructura de la tabla
2. Ver todos los registros de la tabla
3. Ver todos los nombres y distritos de las ciudades
4. Ver el nombre de todas las ciudades que tienen el código ESP
5. Sacar la población menor
6. Averigua la suma de todas los habitantes
7. Saca las ciudades del país USA, que su población sea mayor de 10000

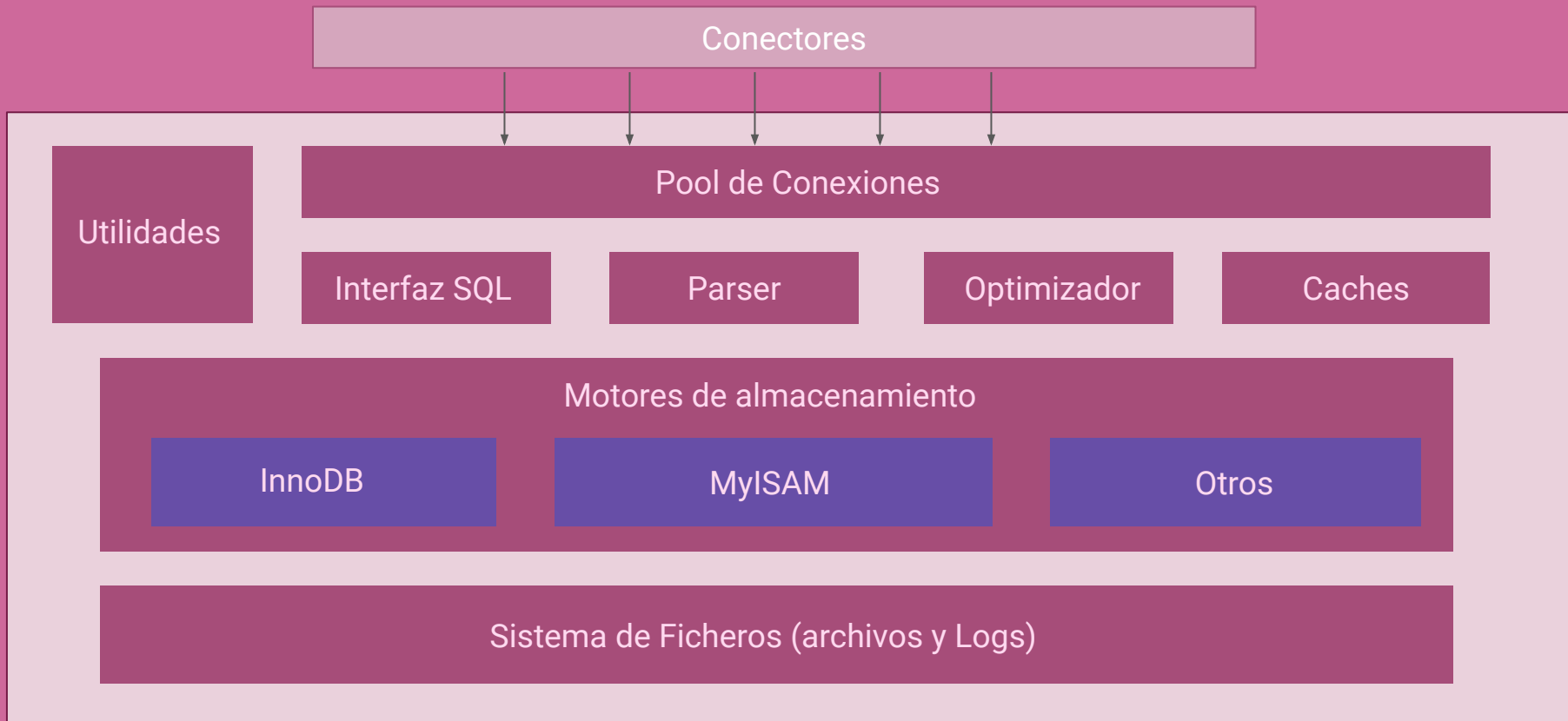
PRÁCTICA: ejemplos sobre la tabla CITY (Solución)

1. Ver estructura de la tabla:
 - `Describe City;`
2. Ver todas las tuplas de la tabla:
 - `Select * from City;`
3. Ver todos los nombres y distritos de las ciudades.
 - `Select name,district from City;`
4. Ver todas las ciudades que tienen el código ESP:
 - `Select name from City Where CountryCode='ESP';`
5. Sacar la población menor:
 - `Select min(population) from City`
6. Obtener el número total de habitantes:
 - `Select sum(population) from city;`
7. Saca las ciudades del país USA, que su población sea mayor de 10000
 - `Select name, population from City where countrycode='USA' and population>10000;`

```
shell> sudo docker pull mysql  
shell> sudo docker run --name some-mysql -e  
MYSQL_ROOT_PASSWORD=my-secret-pw -d mysql:tag
```

```
shell> sudo docker exec -it some-mysql bash
```

```
shell> sudo docker exec -it some-mysql mysql -uroot -p
```

Pool de conexiones: Los usuarios establecen una conexión a la base de datos, que reside en un hilo de procesamiento, esta conexión ejecuta sentencias en nombre del usuario.

Parser: Convierte sentencias SQL en comandos internos

Optimizador: Define cómo se ejecuta la consulta y a qué tablas se accede.

Caches/Buffers: Mantienen los datos en la memoria

InnoDB

Motor de almacenamiento por defecto para MySQL 8 (anteriormente era MyISAM)

Capa de caché: Lugar donde InnoDB almacenará datos en caché para acceder a ellos rápidamente.

Capa de almacenamiento: los datos se guardan en archivos de datos

Capa de transacciones: Se asegura de que todo se escriba de forma segura en un almacenamiento persistente, incluso cuando el servidor de la base de datos se bloquea.

MEJORES PRÁCTICAS AL CREAR UNA TABLA

Cada tabla debe tener una clave primaria (PRIMARY_KEY)

Elige siempre el tipo de dato correcto para cada columna

Elige el tipo de dato más pequeño que se ajuste a tus datos

Crea un índice para cada columna que vaya a ser una condición de un WHERE

Comandos

- Lenguaje de Definición
- Lenguaje de Manipulación

El Lenguaje SQL LDD: Lenguaje de Definición

Sus acciones buscan definir la semántica del esquema relacional: qué relaciones hay, y cómo son, cuáles son sus dominios, cuáles las asociaciones, restricciones, etc.

- Tres acciones básicas: creación, supresión, alteración
- Tres instrucciones básicas: CREATE, DROP, ALTER
- Aplicables a una amplia gama de elementos: tablas, vistas, índices....

El Lenguaje SQL LMD: Lenguaje de Manipulación

Sus instrucciones constan de: LOCALIZACIÓN + ACCIÓN

Dos tipos de instrucciones: actualización y recuperación:

- Tres acciones de actualización: inserción, borrado, modificación
 - Tres instrucciones: **INSERT, DELETE, UPDATE**
- Acción de recuperación: selección
 - Una Instrucción: **SELECT**

Lenguaje de Definición

- **CREATE**
- **DROP**
- **ALTER**

CREATE (databases)

- Una base de datos almacena información mediante un conjunto de tablas, cada una de ellas con una estructura, a priori, diferente..

```
mysql> CREATE DATABASE escuela;
```

La llamada a la variable STATUS, nos indicará, entre otras cosas, la base de datos sobre la que estamos trabajando.

```
mysql> status;
```

CREATE (table)

- Al crear una tabla debemos resolver qué campos (columnas) tendrá y qué tipo de datos almacenarán cada uno de ellos, es decir, su estructura.
- La tabla debe ser definida con un nombre que la identifique y con el cual accederemos a ella.
- Cada campo (columna) debe tener un nombre. El nombre del campo hace referencia a la información que almacenará.
- Cada campo (columna) también debe definir el tipo de dato que almacenará.
- Si intentamos crear una tabla con un nombre ya existente -->ERROR

CREATE (table)

```
mysql> CREATE TABLE profesores (  
→ CodProfe VARCHAR(10),  
→ nombre VARCHAR (30),  
→ apellidos VARCHAR (50),  
→ FechaEntrada DATE,  
→ departamento VARCHAR (20),  
→ asignatura VARCHAR(20)  
→ );
```

TIPOS DE DATOS BÁSICOS

Antes de crear una tabla debemos pensar en sus campos y optar por el tipo de dato adecuado para cada uno de ellos.

- **VARCHAR:** se usa para almacenar cadenas de caracteres.
Se coloca entre comillas (simples): 'Hola'.
65535 caracteres
`varchar(30)` → almacenará una cadena de hasta 30 caracteres.
- **INTEGER o INT:** se usa para guardar valores numéricos enteros, de -2000000000 a 2000000000 aprox.
- **FLOAT:** se usa para almacenar valores numéricos decimales.
Se utiliza como separador el punto (.).

OTROS TIPOS DE DATOS

- Fechas y Horas: DATE (fecha), DATETIME (fecha y hora), TIME (hora).
- CHAR (x): define una cadena de longitud fija, su rango es de 1 a 255 caracteres.
- TEXT: bloques de caracteres de mucha longitud para almacenar descripciones, normalmente.
- "NULL". El valor 'null' significa “valor desconocido” o "dato inexistente". No es lo mismo que 0 o una cadena vacía.
- SET('value1','value2',...). Un campo que puede contener ninguno, uno ó varios valores de una lista. La lista puede tener un máximo de 64 valores

OTROS TIPOS DE DATOS

- INTEGER tiene subtipos:
 - ❖ MEDIUMINT : va de -8000000 a 8000000 aprox. Sin signo va de 0 a 16000000 aprox.
 - ❖ SMALLINT : va de -30000 a 30000 aprox., sin signo, de 0 a 60000 aprox.
 - ❖ TINYINT : define un valor entero pequeño, cuyo rango es de -128 a 127 . El tipo sin signo va de 0 a 255.
 - ❖ BOOL o BOOLEAN : sinónimos de `tinyint(1)`. Un valor cero se considera falso, los valores distintos de cero, verdadero.
 - ❖ BIGINT : es un entero largo. Va de -9000000000000000000 a 9000000000000000000 aprox. Sin signo es de 0 a 10000000000000000000 .

OTROS TIPOS DE DATOS

- DATE tiene subtipos:
 - ❖ DATE: representa una fecha con formato "YYYY-MM-DD". El rango va de "1000-01-01" a "9999-12-31".
 - ❖ DATETIME: almacena fecha y hora, su formato es "YYYY-MM-DD HH:MM:SS.ssssss". El rango es de "1000-01-01 00:00:00" a "9999-12-31 23:59:59.999999".
 - ❖ TIME: una hora. Su formato es "HH:MM:SS". El rango va de "-838:59:59" a "838:59:59".
 - ❖ TIMESTAMP: Marca temporal en formato 'YYYY-MM-DD HH:MM:SS'.
 - ❖ YEAR(2) y YEAR(4): un año. Su formato es "YYYY" o "YY". Permite valores desde 1901 a 2155 (en formato de 4 dígitos) y desde 1970 a 2069 (en formato de 2 dígitos).

OTROS TIPOS DE DATOS

- FLOAT (t,d): número de coma flotante. Su rango es de $-3.4e+38$ a $-1.1e-38$ (9 cifras).
- DECIMAL o NUMERIC (t,d): el primer argumento indica el total de dígitos y el segundo, la cantidad de decimales.
 - ❖ Si queremos almacenar valores entre 0.00 y 99.99 debemos definir el campo como tipo "decimal (4,2)".
 - ❖ Para los tipos "float" y "decimal" se utiliza el punto como separador de decimales.
- Todos los tipos enteros pueden tener el **atributo "unsigned"**, esto permite sólo valores positivos y duplica el rango (de 0 a 4000000000).
- Los tipos de coma flotante también aceptan el atributo "unsigned", pero el valor del límite superior del rango no se modifica.

Ejemplos de distintos datos numéricos

- Facturación de una PYME.
- Edad de unos participantes.
- Kilómetros de distancia entre planetas.
- Cantidades de un producto.
- Cuenta total de una compra.
- Usuarios de media en un acceso a evento.
- ...

Ejemplo a ver:

```
create table visitante(  
  codvisit varchar(10),  
  codTienda varchar(12),  
  nombre varchar(30),  
  edad tinyint unsigned,  
  sexo char(1),  
  domicilio varchar(30),  
  ciudad varchar(20),  
  telefono varchar(11),  
  montocompra float unsigned,  
  primary key (codvisit),  
  foreign key (codTienda) references tienda (codigo)  
);
```

MySQL no es
sensible a
mayúsculas

Modificaciones para CREATE

- Un campo de tipo entero puede tener otro atributo extra 'auto_increment'. Los valores de un campo 'auto_increment', se inician en 1 y se incrementan en 1 automáticamente.
- Se utiliza generalmente en campos correspondientes a códigos de identificación para generar valores únicos para cada nuevo registro que se inserta.
- Sólo puede haber un campo "auto_increment" y debe ser clave primaria (o estar indexado)

AUTO_INCREMENT

- Un campo de tipo entero puede tener otro atributo extra 'auto_increment'. Los valores de un campo 'auto_increment', se inician en 1 y se incrementan en 1 automáticamente.
- Se utiliza generalmente en campos correspondientes a códigos de identificación para generar valores únicos para cada nuevo registro que se inserta.
- Sólo puede haber un campo "auto_increment" y debe ser clave primaria (o estar indexado)

AUTO_INCREMENT

- Para definir un campo autoincrementable colocamos "auto_increment" después de la definición del campo al crear la tabla.
- Para establecer que un campo autoincrementa sus valores automáticamente, éste debe ser entero (integer) y debe ser clave primaria o estar indexado:

```
○ create table libros(  
  codigo int auto_increment,  
  titulo varchar(50),  
  autor varchar(50),  
  editorial varchar(25),  
  primary key (codigo)  
);
```

- Cuando un campo tiene el atributo "auto_increment" no es necesario ingresar valor para él, porque se inserta automáticamente tomando el último valor como referencia, o 1 si es el primero

AUTO_INCREMENT

- Un campo "auto_increment" funciona correctamente sólo cuando contiene únicamente valores positivos.
- Está permitido ingresar el valor correspondiente al campo "auto_increment", por ejemplo:

```
insert into libros (codigo,titulo,autor,editorial)  
values(6,'Martin Fierro','Jose Hernandez','Paidos');
```
- Pero debemos tener cuidado con la inserción de un dato en campos "auto_increment". Debemos tener en cuenta que:
 - si el valor está repetido aparecerá un mensaje de error y el registro no se ingresará.
 - si el valor dado saltea la secuencia, lo toma igualmente y en las siguientes inserciones, continuará la secuencia tomando el valor más alto.
 - si el valor ingresado es 0, no lo toma y guarda el registro continuando la secuencia.
 - si el valor ingresado es negativo (y el campo no está definido para aceptar sólo valores positivos), lo ingresa.--> ERROR, debemos definir : `codigo int unsigned auto_increment`

PRIMARY KEY

- La clave PRIMARY **KEY**, es una columna o varias columnas, que sirven para señalar cuál es la clave primaria de la tabla actual.
- La columna o columnas señaladas no podrán contener valores nulos o repetidos. Será la forma de identificar unívocamente a un registro en particular de es tabla.

```
CREATE TABLE usuario(  
  user_id INT AUTO_INCREMENT PRIMARY KEY,  
  username VARCHAR(40),  
  password VARCHAR(255),  
  email VARCHAR(255) );
```

```
CREATE TABLE departamentos (  
  dep int NOT NULL,  
  departamento varchar(255),  
  PRIMARY KEY (dep) );
```

- Puede definirse como AUTO_INCREMENT, y la clave será creada automáticamente cada vez que se ingrese un registro.
- Definida en el propio campo o al final.

PRIMARY KEY

La clave primaria, **PRIMARY KEY**, identifica de manera única cada fila de una tabla.

La columna definida como clave primaria (**PRIMARY KEY**) debe ser UNIQUE (valor único) y NOT NULL (no puede contener valores nulos).

Cada tabla sólo puede tener una clave primaria (**PRIMARY KEY**).

- Podemos tener varias columnas que forman parte de la clave principal:

```
CREATE TABLE Persona (  
  ID int NOT NULL,  
  Apellido varchar(50) NOT NULL,  
  Nombre varchar(25),  
  Age int,  
  PRIMARY KEY (ID,Apellido)  
);
```

```
CREATE TABLE Persona (  
  ID int NOT NULL,  
  Apellido varchar(50) NOT NULL,  
  Nombre varchar(25),  
  Age int,  
  CONSTRAINT PK_Persona PRIMARY KEY (ID,Apellido)  
);
```

Aquí sólo existe una PK, pero su valor se compone del contenido de dos columnas

PRIMARY KEY

- Para crear una nueva restricción de clave primaria en una columna (ID), cuando la tabla ya está creada:

```
ALTER TABLE Persona ADD PRIMARY KEY (ID);
```

```
ALTER TABLE Persona ADD CONSTRAINT PK_Persona PRIMARY KEY (ID,Apellido);
```

- Eliminación de clave primaria

```
ALTER TABLE Persona DROP PRIMARY KEY;
```

FOREIGN KEY

- La clave externa o **FOREIGN KEY**, es una columna o varias columnas, que sirven para señalar cuál es la clave primaria de otra tabla.
- La columna o columnas señaladas como **FOREIGN KEY**, sólo podrán tener valores que ya existan en la clave primaria PRIMARY KEY de la otra tabla.

```
CREATE TABLE departamentos (  
  dep int NOT NULL,  
  departamento varchar(255),  
  PRIMARY KEY (dep)  
)
```

```
CREATE TABLE empleado(  
  per int NOT NULL,  
  nombre varchar(255),  
  apellido1 varchar(255),  
  depto int NOT NULL,  
  PRIMARY KEY (per),  
  FOREIGN KEY (depto) REFERENCES  
  departamentos(dep)  
)
```

FOREIGN KEY - ejemplo -

- Clave primaria de una tabla que se obtiene de una relación triple (una 1:1 y dos 1:N) en un DER, y está formada por dos claves ajenas:

```
CREATE TABLE user_roles(  
  user_id INT NOT NULL,  
  role_id INT NOT NULL,  
  PRIMARY KEY(user_id,role_id),  
  FOREIGN KEY(user_id) REFERENCES users(user_id),  
  FOREIGN KEY(role_id) REFERENCES roles(role_id)  
);
```

FOREIGN KEY

- Varias FK en una tabla:
 - `FOREIGN KEY (nomCampo2) REFERENCES nomTabla (nomCampo1),
CONSTRAINT fk2 FOREIGN KEY (nomCampo2) REFERENCES nomTabla (nomCampo1)`
- Modificación:
 - `ALTER TABLE empleado ADD FOREIGN KEY (dep) REFERENCES departamentos(dep);`

```
mysql> ALTER TABLE pedido ADD FOREIGN KEY (codCli) REFERENCES cliente(codcliente);
```

- Eliminación:
 - `ALTER TABLE personas DROP FOREIGN KEY dep;`

DROP

- Comando utilizado para eliminar una tabla.

```
mysql> DROP TABLE profesor;
```

- Si quisiéramos borrar una tabla que ya estuviera borrada o directamente una que no existiera en nuestra bbdd → ERROR.

```
mysql> DROP TABLE IF EXISTS profesor;
```

DROP vs TRUNCATE

- Comando utilizado para eliminar una tabla PRESERVANDO su estructura.

```
mysql> TRUNCATE TABLE profesor;
```

- La diferencia con "drop table" es que esta sentencia borra la tabla, "truncate table" simplemente la vacía.
- La diferencia con "delete" es la velocidad, es más rápido "truncate table" que "delete" (se nota cuando la cantidad de registros es muy grande) ya que éste borra los registros uno a uno.

PRÁCTICA GUIADA

- Crear una base de datos librería
- Crear tablas para libros, compradores, editoriales (proveedores), pedidos, facturas.
- Elegir los tipos de datos adecuados a cada campo.
- Identificar las “posibles” relaciones entre tablas mediante un DER.
- Visualice las tablas creadas.
- Visualice la estructura de cada una de las tablas y verifique que de verdad era lo que quería indicar, en caso contrario, borre la tabla, y vuelva a crearla.

ALTER

- Modificar la estructura de una tabla existente:
 - agregar nuevos campos,
 - eliminar campos existentes,
 - modificar el tipo de dato de un campo,
 - (agregar o quitar modificadores como "null", "unsigned", "auto_increment"),
 - cambiar el nombre de un campo,
 - agregar o eliminar la clave primaria,
 - agregar y eliminar índices,
 - renombrar una tabla.

ALTER

- Modificar la estructura de una tabla existente:

```
ALTER TABLE [table] ([actions]);
```

actions:

```
ADD COLUMN [columndef] (AFTER) [othercolumn] (FIRST)
```

```
DROP COLUMN [column]
```

```
MODIFY COLUMN [column] [columnnewdef]
```

```
CHANGE COLUMN [column] [columnnewdef]
```

```
RENAME TO [newtablename]
```

ALTER

```
ALTER TABLE libro ADD cantidad smallint unsigned not null;
```

```
ALTER TABLE libro ADD cantidad2 tinyint unsigned after autor;
```

```
ALTER TABLE libro DROP editorial;
```

Si una tabla tiene sólo un campo, éste no puede ser borrado.

```
ALTER TABLE libro DROP editorial, drop cantidad;
```

Si eliminamos un campo clave, la clave también se elimina

```
ALTER TABLE libro MODIFY cantidad smallint unsigned;
```

```
ALTER TABLE libro MODIFY titulo varchar(40) not null;
```

ALTER

```
ALTER TABLE libro CHANGE precio coste decimal (5,2);  
ALTER TABLE libro CHANGE titulo nombre varchar(40) not null;
```

```
ALTER TABLE usuario RENAME contacto;  
RENAME TABLE usuario TO contacto;
```

ALTER

Eliminación de clave antigua e imposición de nueva.

```
ALTER TABLE libro DROP PRIMARY KEY;
```

```
ALTER TABLE libro MODIFY Codtitulo VARCHAR(9) PRIMARY KEY;
```

O más elegantemente:

```
ALTER TABLE libro ADD PRIMARY KEY (CodTitulo);
```

Reordenación de campos:

```
ALTER TABLE libros MODIFY COLUMN numpags int AFTER editorial
```

PRÁCTICA EN PAREJAS II

- Crear una base de datos Club_Deportivo
- Identificar (discutir y justificar), mínimo cuatro entidades principales, como necesidad de información importante..
- Definir la estructura de cada una de las tablas.
- Argumentar los tipos de datos a utilizar.
- Identificar las “posibles” relaciones entre tablas mediante un DER.
- Visualice las tablas creadas.
- Visualice la estructura de cada una de las tablas y verifique que de verdad era lo que quería indicar, en caso contrario, borre la tabla, y vuelva a crearla.
- Realice modificaciones, borrados, ... y justifique los cambios