MP_0486. Acceso a datos
UF2. Manejo de conectores a bases de datos relacionales

2.2. Instalación de MySQL y creación de la primera BD

Índice

Objetivos 4
Primeros pasos con MySQL 5
Introducción y conceptos básicos 5
Instalación y configuración de MySQL 5
El lenguaje de definición de datos (DDL) 7
Sublenguaje DDL 7
Tipos de datos numéricos 7
Tipos de datos para fechas y horas 8
Tipos de datos para guardar cadenas de texto 9
Creación, modificación y borrado de tablas 11
Abrir MySQL Workbench y crear el esquema de la base de datos 11
Crear tablas 14
Modificar y borrar tablas 16
ALTER TABLE – ADD COLUM (añadir atributo) 16
ALTER TABLE – ALTER COLUMN (modificar atributo) 17
ALTER TABLE – DROP COLUMN (borrar atributo) 17
Creación, modificación y borrado de índices 18
¿Qué es un índice? 18
Crear índices 19
Modificar índices 20
Borrar índices 20
Restricciones 21
Restricciones de integridad 21
Los tipos de datos 21
Restricción UNIQUE 22
Restricción NOT NULL 22
Restricción PRIMARY KEY 22
Restricción FOREIGN KEY 23
Restricción AUTO_INCREMENT 24
Restricciones ENUM y SET 25

Ejemplo práctico 26
Diseño físico de la BD FERRETERIA 26
Despedida 30
Resumen 30

Objetivos

En esta lección perseguimos los siguientes objetivos:

- Instalar MySQL y establecer la configuración básica.
- Crear el diseño físico de una base de datos, utilizando el lenguaje de definición de datos y el SGBD MySQL.
- Preparar los recursos necesarios para acometer la lección 3 de esta misma unidad.

Primeros pasos con MySQL

Introducción y conceptos básicos

MySQL es el SGBD relacional que más se utiliza actualmente, sobre todo en entornos de desarrollo web.

Los más importantes proyectos de gestores de contenidos, como WordPress, Joomla o Prestashop, utilizan bases de datos MySQL.

MySQL actúa como **servidor de bases de datos a partir de instancias**, también denominadas conexiones MySQL.

Cada instancia o conexión puede alojar varios **esquemas (*schemas*)** y atenderá conexiones de clientes que deseen acceder a dichos esquemas a través del nombre del servidor o dirección IP y un puerto de comunicaciones.

Un esquema es una **estructura que engloba a una base de datos y otros objetos asociados a ella**, como vistas, procedimientos almacenados, lanzadores (*tiggers*), etc.

Instalación y configuración de MySQL

MySQL Workbench es una herramienta visual de diseño que está integrada en el SGBD MySQL, además de una interfaz de usuario para su gestión de forma más rápida y cómoda.

A continuación, aprenderás a descargar, instalar y configurar el servidor de MySQL y sus herramientas asociadas, incluyendo MySQL Workbench.

Existen varias versiones de MySQL. Estos dos vídeos te mostrarán cómo descargar, instalar y configurar la versión *Community*.

El siguiente vídeo muestra los pasos para descargar MySQL desde Internet:

https://vimeo.com/telefonicaed/review/265581955/21bb7d9ae2

Durante la instalación de MySQL pasarás por tres etapas: **instalación de prerrequisitos, instalación de MySQL y configuración de MySQL.**

En la etapa de configuración te pedirá el *password* (contraseña) del usuario *root* (administrador). El usuario *root* se crea por defecto durante la instalación. **Utiliza un *password* del que luego te acuerdes.**

El siguiente vídeo muestra los pasos a seguir para instalar y configurar MySQL:

https://vimeo.com/telefonicaed/review/265581979/cbcec1659b

El lenguaje de definición de datos (DDL)

Sublenguaje DDL

Puesto que el lenguaje de definición de datos (DDL) nos permite, entre otras cosas, realizar el diseño físico de la base de datos por cada tabla o relación que creemos, habrá que determinar sus atributos o columnas.

Para cada atributo será necesario establecer su dominio, es decir, el tipo y rango de valores que puede almacenar.

Pero antes de aprender a crear tablas utilizando el lenguaje DDL es necesario conocer los **tipos de datos que maneja el lenguaje SQL.**

Tipos de datos numéricos

SQL cuenta con gran cantidad de tipos de datos para almacenar números enteros o reales.

TINYINT:
Ocupa 1 byte y su dominio es entre -128 y 127 o entre 0 y 255 (utilizando la restricción UNSIGNED).

SMALLINT:
Ocupa 2 bytes y su dominio es entre -32.768 y 32.767 o entre 0 y 65.535 (utilizando la restricción UNSIGNED).

MEDIUMINT:
Ocupa 3 bytes y su dominio es entre -8.388.608 y 8.388.607 o entre 0 y 16.777.215 (utilizando la restricción UNSIGNED).

INT (INTEGER):
Ocupa 4 bytes y su dominio es entre -2.147.483.648 y 2.147.483.647 o entre 0 y 4.294.967.295 (utilizando la restricción UNSIGNED).

BIGINT:

Ocupa 8 bytes y su dominio es entre -9.223.372.036.854.775.808 y 9.223.372.036.854.775.807 o entre 0 a 18.446.744.073.709.551.615 (utilizando la restricción UNSIGNED).

FLOAT:

Ocupa 4 bytes y almacena un número pequeño en coma flotante de precisión simple. Dominio entre -3.402823466E+38 y -1.175494351E-38, 0, y desde 1.175494351E-38 a 3.402823466E+38 (utilizando la restricción UNSIGNED).

DOUBLE (REAL):

Ocupa 8 bytes. Almacena números de coma flotante con precisión doble. Dominio entre -1.7976931348623157E+308 y -2.2250738585072014E-308, 0, y desde 2.2250738585072014E-308 a 1.7976931348623157E+308 (utilizando la restricción UNSIGNED).

DECIMAL (M, D):

Ocupa M+2 bytes si D>0 y M+1 bytes si D=0. Números en coma flotante almacenados como cadena.

BIT (BOOL, BOOLEAN):

Número entero con valor 0 o 1.

Tipos de datos para fechas y horas

SQL dispone de varios tipos de datos para almacenar fechas, horas o instantes en el tiempo (fecha+hora).

DATE:

Ocupa 3 bytes. Fecha en formato año-mes-día, con rango desde '1000-01-01' hasta '9999-12-31'.

DATETIME:

Ocupa 8 bytes. Fecha en formato año-mes-día y hora en formato horas-minutos-segundos. Rango de '1000-01-01 00:00:00' y '9999-12-31 23:59:59'.

TIME:

Ocupa 3 bytes. Para almacenar una hora en formato 'hh:mm:ss'.

YEAR:
Ocupa 1 byte. Almacena un año con 2 o 4 dígitos de longitud. Por defecto son 4 dígitos, para 2 dígitos se expresa YEAR(2). Rango con 4 dígitos de 1901 a 2155. Rango con 2 dígitos 1970 a 2069 (70 a 69).

TIMESTAMP:
Ocupa 4 bytes. Fecha y hora en formato UCT. Rango entre '1970-01-01 00:00:01' y '2038-01-19 03:14:07'.

Tipos de datos para guardar cadenas de texto

SQL dispone de muchísimos tipos de datos para almacenar textos de tamaño fijo o variable.

CHAR(N):
Cadena de longitud fija (N caracteres). Límite 255 caracteres. Si se declara CHAR(25) y se almacena el valor 'PEPE', ocupará 25 caracteres aunque sólo se hayan utilizado 4.

VARCHAR(N):
Cadena de longitud variable (N caracteres). Límite 255 caracteres. Si se declara VARCHAR(25) y se almacena el valor 'PEPE', sólo ocupará 4 caracteres.

TINYTEXT:
Cadena con un máximo de 255 caracteres. No distingue entre mayúsculas y minúsculas a la hora de realizar ordenaciones o búsquedas.

TINYBLOB:
Cadena con un máximo de 255 caracteres. Distingue entre mayúsculas y minúsculas a la hora de realizar ordenaciones o búsquedas.

TEXT:
Cadena con un máximo de 65.535 caracteres. No distingue entre mayúsculas y minúsculas a la hora de realizar ordenaciones o búsquedas.

BLOB:
Cadena con un máximo de 65.535 caracteres. Distingue entre mayúsculas y minúsculas a la hora de realizar ordenaciones o búsquedas.

MEDIUMTEXT:
Cadena con un máximo de 16.777.215 caracteres. No distingue entre mayúsculas y minúsculas a la hora de realizar ordenaciones o búsquedas.

MEDIUMBLOB:
Cadena con un máximo de 16.777.215 caracteres. Distingue entre mayúsculas y minúsculas a la hora de realizar ordenaciones o búsquedas.

LONGTEXT:
Cadena con un máximo de 4.294.967.295 caracteres. No distingue entre mayúsculas y minúsculas a la hora de realizar ordenaciones o búsquedas.

LONGBLOB:
Cadena con un máximo de 4.294.967.295 caracteres. Distingue entre mayúsculas y minúsculas a la hora de realizar ordenaciones o búsquedas.

ENUM:
Campo que sólo puede tomar un valor de una lista específica. Acepta hasta 65.535 valores distintos.

SET:
Campo que sólo puede tomar un valor de una lista específica. Acepta hasta 64 valores distintos.

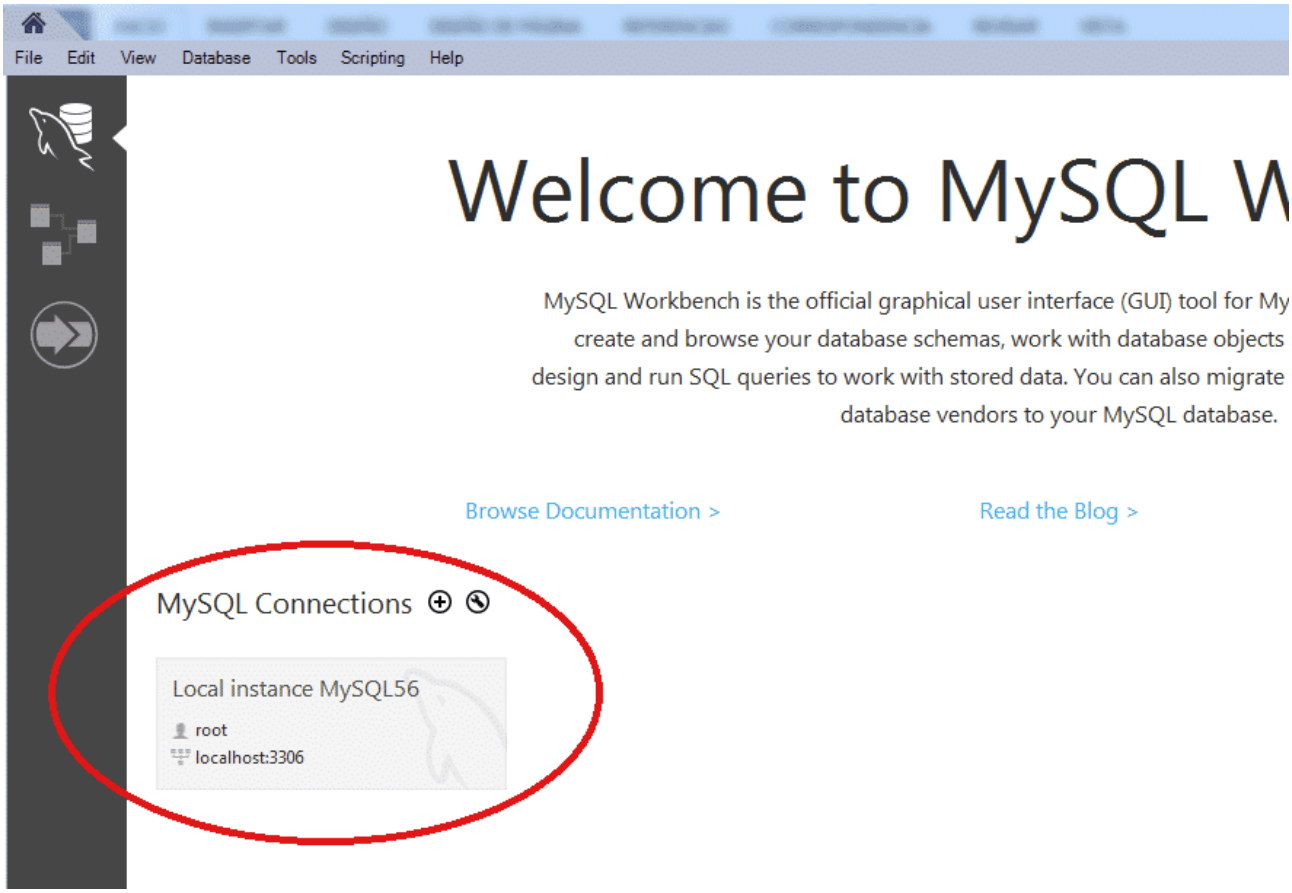
Creación, modificación y borrado de tablas

Abrir MySQL Workbench y crear el esquema de la base de datos

Antes de ponernos a crear tablas, es necesario comenzar por crear un esquema y ponerlo en uso.

Para comenzar abre MySQL Workbench.

La pantalla será similar a la mostrada en la imagen.

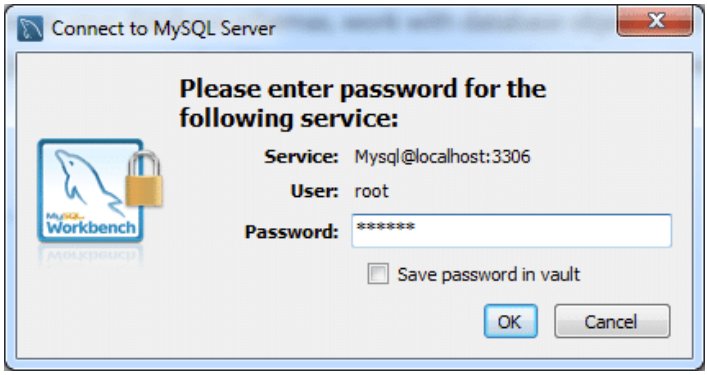


En la zona que se muestra en la imagen con un óvalo rojo, es donde aparece la instancia o *Connection* de MySQL. Podrían existir varias.

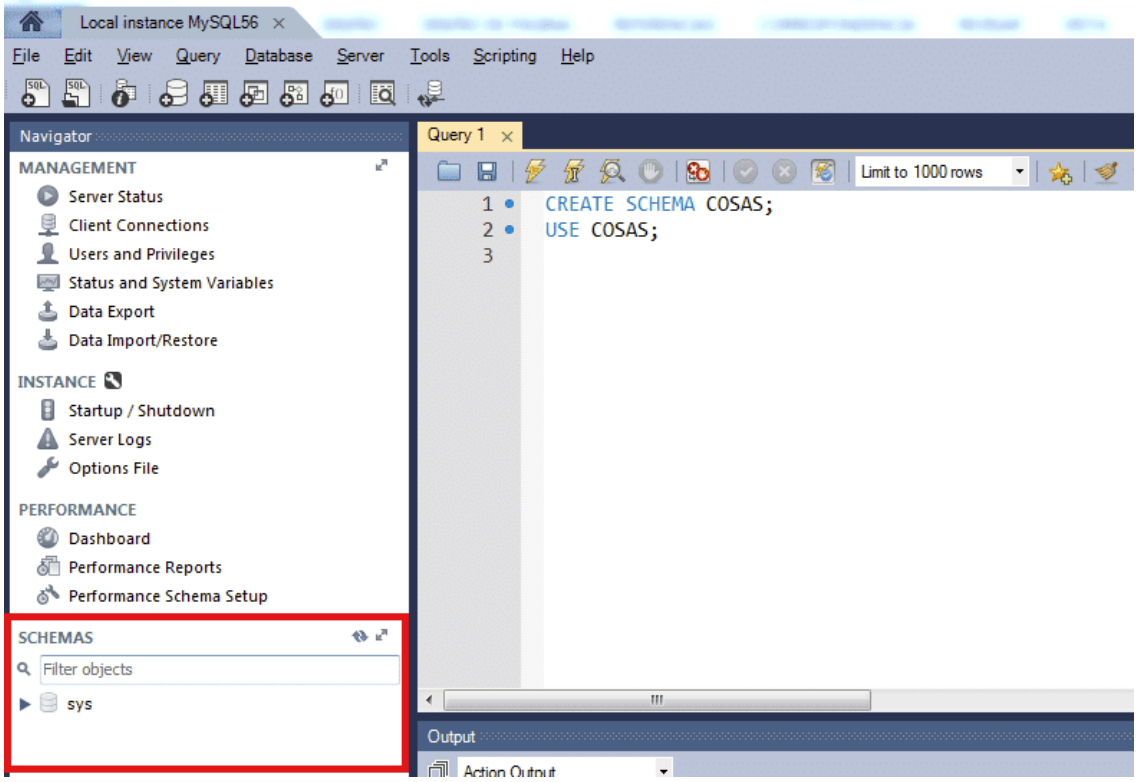
Observa que a la derecha del texto "MySQL Connections" aparece un icono con un signo + que te permitirá crear una nueva instancia si lo deseas, y otro icono con una llave inglesa que te permitirá modificar la configuración de la instancia si es necesario.

A continuación, aparecen el nombre de la instancia, el usuario autorizado, el servidor y el puerto. Para comenzar a trabajar **debes hacer clic sobre el nombre de la instancia. En la imagen aparece como "Local instance MySQL56".**

A continuación, **te pedirá la contraseña para el usuario *root*** que especificaste durante la instalación.



Pulsa OK y llegarás al entorno de trabajo de MySQL Workbench, donde podrás comenzar en breve a escribir tus consultas SQL.



El panel de la izquierda te brinda las distintas herramientas disponibles en MySQL Workbench.

Resaltado en rojo está el área donde se van mostrando los distintos esquemas. Por el momento sólo se ve el esquema *sys* (base de datos del sistema).

El área de la derecha es el editor para escribir SQL; para comenzar puedes escribir lo siguiente:

```
CREATE SCHEMA COSAS;
USE COSAS;
```

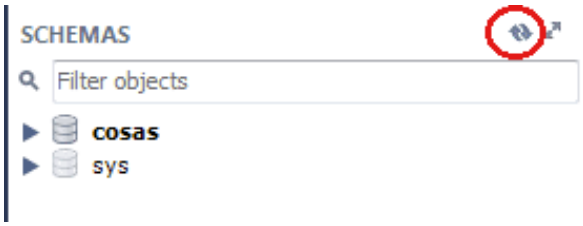
El carácter de **punto y coma representa el final de línea** o sentencia. Es necesario cuando ejecutamos varias sentencias para identificar cuando termina una y comienza la otra.

Luego, **ejecuta** ambas sentencias **haciendo clic en el icono** que está representado por un **rayo amarillo.**

Por el momento la idea no es que crees una base de datos normalizada completa, sino que comiences a familiarizarte con el entorno de MySQL Workbench y los tipos de datos. Por esa razón, hemos llamado al esquema *Cosas*, ya que será un almacén de tablas de prueba y aprendizaje.

Al ejecutar estas sentencias, primero has creado un esquema llamado *Cosas* y luego lo has puesto en uso, lo que significa que cuando comiences a crear tablas e índices se crearán en este esquema y no en otro.

El área de esquemas te mostrará el nuevo esquema una vez que hagas clic en el botón "Actualizar", que está resaltado con un óvalo rojo en la siguiente imagen:



El esquema que se muestra en negrita es el esquema en uso. Debes tener esto siempre en cuenta para no crear por error objetos en el esquema incorrecto.

Crear tablas

Para crear nuevas relaciones o tablas se utiliza la sentencia *CREATE TABLE.*

Esta sentencia tiene la siguiente **estructura**:

```
CREATE TABLE nombre_tabla (atributo1 tipo, atributo2 tipo,... atributoN tipo)
```

Como ejemplo, vamos a crear una tabla llamada *Amigos*, que permitirá almacenar el nombre, dirección y teléfono de cada uno de nuestros amigos e identificar mediante una etiqueta dónde lo conocimos (en la escuela, en el trabajo, en el gimnasio o en otro lugar).

```
CREATE TABLE AMIGOS (
        Nombre CHAR(25),
        Direccion VARCHAR(100),
        Tlf VARCHAR(15),
        Etiqueta SET('Escuela','Trabajo','Gimnasio','Otros')
);
```

Para comprobar que las restricciones de dominio asignadas en la tabla *Amigos* están funcionando, puedes insertar una nueva fila con una sentencia INSERT como esta:

```
INSERT INTO AMIGOS VALUES ('Carlos', 'C/ Oca, 25', '616616616','Barrio');
```

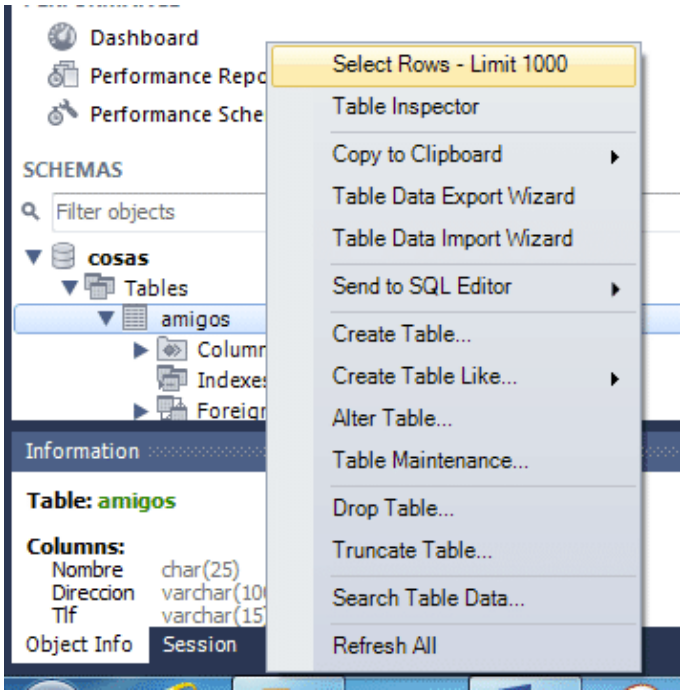
Comprobarás que da error y no añade la nueva fila porque el valor 'Barrio' está fuera del conjunto de valores del dominio ('Escuela', 'Trabajo', 'Gimnasio', 'Otros').

Prueba ahora a cambiar el valor por uno de los que forman parte del dominio, como en el siguiente ejemplo:

```
INSERT INTO AMIGOS VALUES ('Carlos', 'C/ Oca, 25', '616616616','Trabajo');
```

Ahora sí que ha funcionado.

Para comprobar que el nuevo registro ha sido insertado correctamente puedes hacer doble clic sobre el nombre del esquema *Cosas*, luego en *Tables* y luego en *Amigos*. Si no aparece, tendrás que volver a usar el icono actualizar.



Si haces un clic derecho sobre la tabla *Amigos* aparece un menú contextual como el que ves en la imagen.

La opción "Select Rows ..." te mostrará los registros o filas.

ACTIVIDADES

A continuación, te proponemos dos ejercicios en los que tendrás que crear una tabla. Para cada tabla deberás consultar la lista de tipos de datos para escoger el tipo idóneo en cada caso.

Después de crear la tabla, inserta un registro como muestra.

1. Tabla *Alumno* con los siguientes campos: *Nombre* (cadena de longitud variable con un ancho máximo de 25), *Edad* (valor numérico entero entre 0 y 255), *Tlf* (cadena de longitud fija con un ancho de 12), *Mensualidad* (número real de 4 bytes que no admita negativos), *NumeroMatricula* (número entero positivo con valor máximo 65535) y *fechaMatricula* (fecha de 3 bytes de ocupación con máximo valor '9999-12-31').

   Solución:

```
CREATE TABLE ALUMNO (
        Nombre VARCHAR(25),
        Edad TINYINT UNSIGNED,
        Tlf CHAR(12),
        Mensualidad FLOAT UNSIGNED,
        FechaMatricula DATE
);
INSERT INTO ALUMNO VALUES (
```

```
        'Perico de los Palotes',
        25,
        '913670488',
        53.50,
        3524,
        '2017-12-11'
);
```

2. Tabla *Candidato* con los siguientes campos: *Nombre* (cadena de longitud variable con un ancho máximo de 25), *FechaNacimiento* (fecha de 3 bytes de ocupación con máximo valor '9999-12-31'), *Tlf* (cadena de lontigud fija con un ancho de 12), *FechaHoraPresentacion* (fecha y hora de 4 bytes) y *Curriculum* (cadena máximo 16.777.215 caracteres que no distinga entre mayúsculas y minúsculas).

   Solución:

```
CREATE TABLE Candidato (
        Nombre VARCHAR(25),
        FechaNacimiento DATE,
        Tlf CHAR(12),
        FechaHoraPresentacion TIMESTAMP,
        Curriculum MEDIUMTEXT
);
INSERT INTO Candidato VALUES (
        'Pedro Delta',
        '1981-12-03',
        '913676767',
        '2017-12-15',
        'Aquí el curriculum de Pedro Delta'
);
```

Modificar y borrar tablas

Para eliminar o modificar la estructura de una tabla debes utilizar la sentencia ALTER.

ALTER TABLE – ADD COLUM (añadir atributo)

Añadir nueva columna en una tabla existente.

Formato

```
ALTER TABLE nombreTabla
ADD COLUMN atributo tipo
```

Ejemplo

```
ALTER TABLE Candidato ADD COLUMN Ciudad VARCHAR(50);
```

Has añadido a la tabla *Candidato* el atributo o columna *Ciudad*. Para las filas que ya estuvieran guardadas en la tabla establecerá la *Ciudad* con el valor NULL. El nuevo atributo quedará en la última columna.

Si queremos situar el nuevo atributo en una columna concreta, podemos utilizar las palabras reservadas BEFORE o FIRST delante del nombre de una columna.

```
ALTER TABLE ALUMNO ADD COLUMN Ciudad VARCHAR(50)
AFTER Nombre;
```

ALTER TABLE – ALTER COLUMN (modificar atributo)

Modificar la definición de una columna existente.

En el siguiente ejemplo, modificamos la definición del atributo *Ciudad* dejándolo en 40 caracteres.

```
ALTER TABLE ALUMNO MODIFY COLUMN Ciudad VARCHAR(40);
```

ALTER TABLE – DROP COLUMN (borrar atributo)

Eliminar una columna.

En el siguiente ejemplo, eliminamos el campo *Ciudad* de la tabla *Alumno*.

```
ALTER TABLE ALUMNO DROP COLUMN Ciudad;
```

Creación, modificación y borrado de índices

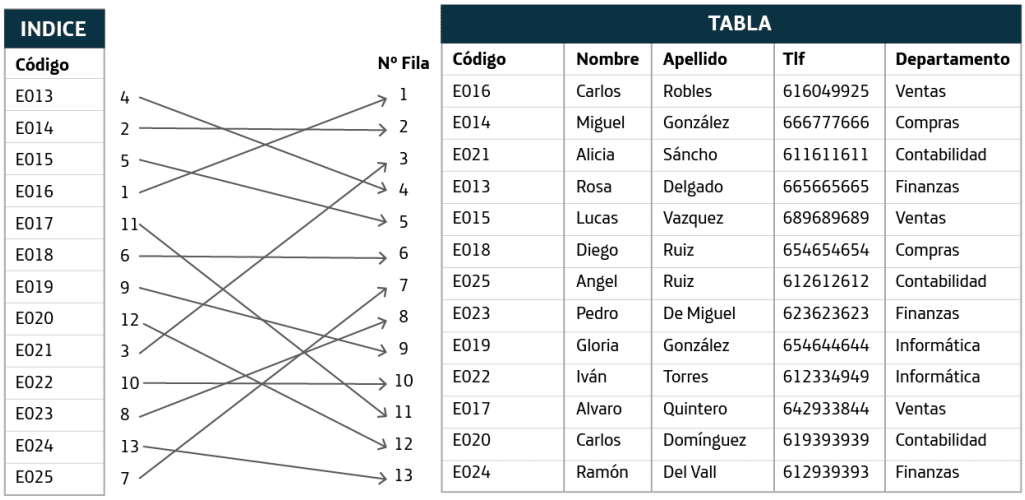
¿Qué es un índice?

Los índices de las tablas son estructuras que permiten agilizar las consultas.

Para que comprendas mejor su funcionalidad puedes compararlo con el índice de cualquier libro.

Se tarda más en buscar un contenido pasando página tras página, que consultando primero en el índice y luego accediendo directamente a la página.

Cada índice es como una nueva tabla formada por una clave, que estará formada por un solo atributo o la unión de varios atributos de la tabla original.



Funcionamiento de los índices.

Cuando realizamos la consulta de un empleado a partir del código, el SGBD buscará en el índice y accederá directamente a la posición de la tabla a la que apunta. Esta operación es transparente para el usuario. Nosotros realizamos la consulta de la misma manera, tanto si hay índice como si no lo hay.

Crear índices

Para crear un índice utilizamos la sentencia CREATE INDEX.

Esta sentencia tiene la siguiente estructura:

```
CREATE INDEX nombre_indice
ON nombre_tabla (column1, column2, ...);
```

Ejemplo

Creamos una tabla llamada *Articulo* y después un índice para la tabla *Articulo* cuya clave es el atributo *Codigo*.

```
CREATE TABLE Articulo (
        Codigo CHAR(4),
        Descripcion VARCHAR(100),
        Precio FLOAT,
        Stock INT
);

CREATE INDEX idx_Codigo
ON Articulo(Codigo);
```

La creación del índice *idx_codigo* agiliza las búsquedas de artículos a partir del código, pero no evita que puedan duplicarse códigos. Para crear un índice que, además, añada la restricción UNIQUE, la sentencia SQL sería así:

```
CREATE UNIQUE INDEX idx_Codigo
ON Articulo(Codigo);
```

CREATE INDEX no se utiliza para crear la clave primaria. Para eso se emplea la cláusula PRIMARY KEY en el momento en que se crea o modifica la tabla.

A continuación, veremos tres alternativas para crear la tabla *Articulo* y establecer el atributo *Codigo* como clave primaria, aunque el resultado será exactamente igual en los tres casos.

1. **Como una restricción aplicada al atributo *Codigo***

```
CREATE TABLE Articulo (
        Codigo CHAR(4) PRIMARY KEY,
        Descripcion VARCHAR(100),
        Precio FLOAT,
        Stock INT
);
```

2. **Como una cláusula que recibe como argumento al atributo de clave primaria**

```
CREATE TABLE Articulo (
        Codigo CHAR(4),
        Descripcion VARCHAR(100),
        Precio FLOAT,
        Stock INT,
        PRIMARY KEY(Codigo)
);
```

3. **A *posteriori* con la sentencia ALTER**

```
CREATE TABLE Articulo (
Codigo CHAR(4),
        Descripcion VARCHAR(100),
        Precio FLOAT,
        Stock INT
);

ALTER TABLE Articulo
MODIFY COLUMN Codigo CHAR(4) PRIMARY KEY;
```

Modificar índices

Ahora aprenderemos que es posible cambiar el nombre de un índice.

Para ello, aplicaremos el siguiente formato de la sentencia ALTER.

```
ALTER TABLE nombre_tabla
RENAME INDEX nombre_indice TO nombre_indice_nuevo;
```

Ejemplo

```
ALTER TABLE Articulo
RENAME INDEX idx_Codigo TO index_Codigo;
```

Borrar índices

Para eliminar un índice se utiliza la sentencia ALTER con la cláusula DROP aplicando el siguiente formato:

```
ALTER TABLE nombre_tabla
DROP INDEX nombre_indice
```

Ejemplo

```
ALTER TABLE Articulo
DROP INDEX idx_Codigo;
```

Si lo que deseamos es borrar la clave primaria:

```
ALTER TABLE Articulo
DROP PRIMARY KEY;
```

Restricciones

Restricciones de integridad

Las restricciones de integridad de una base de datos definen cómo se encargará el DBMS de exigir ciertas reglas.

Estas reglas establecen los **valores permitidos en determinados atributos o campos** para exigir la integridad y coherencia de la base de datos en su conjunto.

Los tipos de datos

El mero hecho de crear una tabla y asignar un tipo de dato a cada atributo lleva consigo una serie de restricciones de integridad, inherentes al modelo relacional.

Observa de nuevo la sentencia para la creación de la tabla *Articulo*:

```
CREATE TABLE Articulo (
        Codigo CHAR(4),
        Descripcion VARCHAR(100),
        Precio FLOAT,
        Stock INT
);
```

El motor de MySQL garantizará que se cumplan las siguientes restricciones:

- No se podrán introducir más de 4 caracteres en el código de producto.
- No se podrán introducir más de 100 caracteres en la descripción del producto.
- El precio deberá ser un valor numérico, no admitirá letras.
- El stock deberá ser un número entero.

Todas estas son **restricciones de dominio**, ya que determinan el conjunto de valores válidos para un atributo.

Restricción UNIQUE

La restricción UNIQUE no permiteque el atributo al que se aplica tenga valores repetidos.

```
CREATE TABLE AMIGOS (
        Nombre CHAR(25) UNIQUE,
        Direccion VARCHAR(100),
        Tlf VARCHAR(15),
        Etiqueta SET('Escuela','Trabajo','Gimnasio','Otros')
);
```

¡Ojo! No se permitirá introducir dos filas en la tabla *Amigos* con el mismo valor del atributo *Nombre*.

Restricción NOT NULL

La restricción NOT NULL no permiteque el atributo al que se aplica se deje vacío.

```
CREATE TABLE AMIGOS (
        Nombre CHAR(25) NOT NULL,
        Direccion VARCHAR(100),
        Tlf VARCHAR(15),
        Etiqueta SET('Escuela','Trabajo','Gimnasio','Otros')
);
```

No se admiten valores nulos en el atributo *Nombre*. Eso significa que tendremos que asignar un valor obligatoriamente.

Hay que tener en cuenta que un campo de texto sin rellenar tendrá por defecto valor NULL.

Restricción PRIMARY KEY

El hecho de definir un atributo como PRIMARY KEY (clave primaria) lleva consigo una importante restricción.

```
CREATE TABLE Articulo (
        Codigo CHAR(4) PRIMARY KEY,
        Descripcion VARCHAR(100),
        Precio FLOAT,
        Stock INT
);
```

Establecer el campo *Codigo* como PRIMARY KEY garantiza:

- **Que no habrá dos productos con el mismo código**, ya que lleva implícita la restricción UNIQUE.
- **Que no se podrá rellenar una fila dejando el código de producto vacío**, ya que lleva implícita la restricción NOT NULL (no admite valores nulos).

Restricción FOREIGN KEY

La restricción FOREIGN KEY permite garantizar la integridad referencial cuando existe una asociación entre dos relaciones o tablas.

Veamos un ejemplo.

Aquí tenemos una tabla *Articulo* que contiene los artículos de un almacén, y otra relación o tabla *VentaContado* que registra las veces que se vende uno de los artículos del almacén.

```
CREATE TABLE Articulo (
        Codigo CHAR(4) PRIMARY KEY,
        Descripcion VARCHAR(100),
        Precio FLOAT,
        Stock INT
);

CREATE TABLE VentaContado (
        NumeroVenta INT AUTO_INCREMENT PRIMARY KEY,
        Codigo_Articulo CHAR(4),
        Unidades INT,
        Precio FLOAT,
        FOREIGN KEY (Codigo_Articulo) REFERENCES Articulo(Codigo)
)
```

En el anterior ejemplo, *CodigoArticulo* es la clave extranjera o ajena que establece la asociación con la tabla *Articulo*.

Sin embargo, en el ejemplo siguiente, la restricción FOREIGN KEY está garantizando que no se añada ninguna *VentaContado* con un código de artículo que no esté previamente registrado en la tabla *Articulo*.

Otra alternativa es crear la tabla *VentaContado* sin establecer la clave ajena y hacerlo después con ayuda de la sentencia ALTER TABLE de la siguiente manera:

```
CREATE TABLE VentaContado (
        NumeroVenta INT AUTO_INCREMENT PRIMARY KEY,
        Codigo_Articulo CHAR(4),
        Unidades INT,
        Precio FLOAT
);
```

```
ALTER TABLE VentaContado
ADD FOREIGN KEY (Codigo_Articulo) REFERENCES Articulo(Codigo);
```

IMPORTANTE: Cada clave ajena es un índice que tendrá un nombre que lo identifique. Sin embargo, en ninguno de los ejemplos anteriores hemos decidido qué nombre deseamos que tenga dicho índice. El motor del SGBD de MySQL ha elegido el nombre por nosotros.

En la sentencia ALTER TABLE es posible indicar el nombre que deseamos para el índice, de la siguiente manera:

```
ALTER TABLE VentaContado
ADD CONSTRAINT FK_ArticuloVenta
FOREIGN KEY (Codigo_Articulo) REFERENCES Articulo(Codigo);
```

FK_ArticuloVenta es ahora el nombre del índice de clave ajena. Conocer el nombre nos permitirá poder eliminar el índice en caso necesario, de la siguiente manera:

```
ALTER TABLE VentaContado
DROP FOREIGN KEY FK_ArticuloVenta;
```

Restricción AUTO_INCREMENT

La cláusula AUTO_INCREMENT permite que un atributo de tipo numérico entero adquiera un valor automático, que va incrementándose (contador).

Por defecto, el primer valor será 1 e irá incrementándose, sumando 1 en cada registro.

```
CREATE TABLE Empleado (
    ID int AUTO_INCREMENT PRIMARY KEY,
    Nombre VARCHAR(50),
    Edad int,
    CHECK (Edad>=18)
);
```

MySQL obliga a que sólo exista un atributo AUTO_INCREMENT en cada tabla y, además, debe ser clave.

Si queremos que el valor inicial sea distinto de 1 podemos utilizar la sentencia ALTER de la siguiente manera:

```
ALTER TABLE Empleado AUTO_INCREMENT=100;
```

AUTO_INCREMENT se considera una restricción porque de manera implícita **hace que el atributo cumpla con las restricciones NOT NULL y UNIQUE**.

Restricciones ENUM y SET

Las restricciones ENUM y SET permiten limitar la entrada de datos en un atributo a un conjunto de valores previamente establecido.

```
CREATE TABLE AMIGOS (
        Nombre CHAR(25),
        Direccion VARCHAR(100),
        Tlf VARCHAR(15),
        Etiqueta SET('Escuela','Trabajo','Gimnasio','Otros')
);
```

El campo *Etiqueta* sólo podrá contener los valores 'Escuela', 'Trabajo', 'Gimnasio' u 'Otros'. ENUM y SET se utilizan exactamente igual, pero presentan las siguientes diferencias:

ENUM

Campo que sólo puede tomar un valor de una lista específica. Acepta hasta 65.535 valores distintos.

SET:

Campo que sólo puede tomar un valor de una lista específica. Acepta hasta 64 valores distintos.

Ejemplo práctico

Diseño físico de la BD FERRETERIA

En este apartado vamos a crear la estructura de la base de datos FERRETERIA, que nos servirá como base para la parte práctica de la siguiente lección.

Y la crearemos conforme al siguiente diagrama de Modelo-Entidad-Relación.

Un Modelo-Entidad-Relación refleja las tablas (también llamadas entidades) que intervienen en una base de datos y las asociaciones existentes entre dichas tablas.

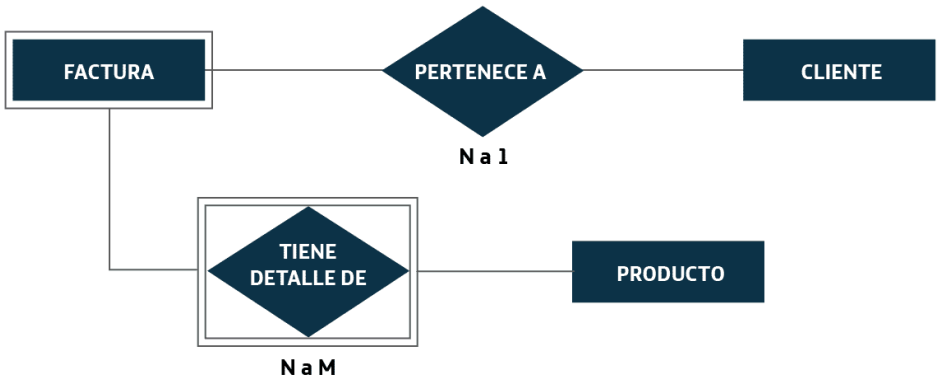


Diagrama de un Modelo Entidad-Relación.

Por el diagrama sabemos que existe una tabla llamada CLIENTE y una tabla llamada FACTURA.

También sabemos que cada factura pertenece a un cliente, es decir, existe una asociación entre ambas tablas. El N a 1 significa que un cliente puede tener N facturas, pero cada factura pertenece a un solo cliente.

También podemos comprobar en el diagrama que existe una tabla PRODUCTO que representa el almacén y que se relaciona con la factura, porque "una factura tiene detalle de producto".

Esta asociación es de N a M, porque cada factura incluye varios detalles de productos y cada producto está incluido en varias facturas. Esta asociación quedará finalmente reflejada físicamente en una tabla llamada DETALLE.

Ahora podemos realizar el diseño físico de la base de datos FERRETERIA con ayuda del lenguaje SQL.

1. Ejecuta este código en MySql para crear la estructura de la BD FERRETERIA.

```
CREATE SCHEMA FERRETERIA;
USE FERRETERIA;

CREATE TABLE CLIENTE (
        NIF CHAR(9) PRIMARY KEY,
        NOMBRE VARCHAR(25) NOT NULL,
        DOMICILIO VARCHAR(100),
        TLF VARCHAR(25),
        CIUDAD VARCHAR(50)
);

CREATE TABLE PRODUCTO (
        CODIGO CHAR(4) PRIMARY KEY,
        DESCRIPCION VARCHAR(100) NOT NULL,
        PRECIO FLOAT,
        STOCK FLOAT,
        MINIMO FLOAT,
        CHECK (PRECIO > 0)
);

CREATE TABLE FACTURA (
        NUMERO INT PRIMARY KEY,
        FECHA DATE,
        PAGADO BOOL,
        NIF CHAR(9),
        FOREIGN KEY (NIF) REFERENCES CLIENTE(NIF)
);

CREATE TABLE DETALLE (
        NUMERO INT,
        CODIGO CHAR(4),
        UNIDADES INT,
        PRECIO FLOAT,
        FOREIGN KEY (NUMERO) REFERENCES FACTURA(NUMERO),
        FOREIGN KEY (CODIGO) REFERENCES PRODUCTO(CODIGO),
        PRIMARY KEY (NUMERO, CODIGO)
)
```

2. Ejecuta esté código en MySql para añadir filas o registros a la tabla CLIENTE.

```
INSERT INTO CLIENTE VALUES
('43434343A', 'DELGADO PEREZ MARISA', 'C/ MIRAMAR, 84 3°A', '925-200-967',
'TOLEDO');

INSERT INTO CLIENTE VALUES
('51592939K', 'LOPEZ VAL SOLEDAD', 'C/ PEZ, 54 4°C', '915-829-394', 'MADRID');

INSERT INTO CLIENTE VALUES
('51639989K', 'DELGADO ROBLES MIGUEL', 'C/ OCA, 54 5°C', '913-859-293',
'MADRID');

INSERT INTO CLIENTE VALUES
```

```
('51664372R', 'GUTIERREZ PEREZ ROSA', 'C/ CASTILLA, 4 4°A', '919-592-932',
'MADRID');
```

3. Ejecuta esté código en MySql para añadir filas o registros a la tabla PRODUCTO.

```
INSERT INTO PRODUCTO VALUES
('CAJ1', 'CAJA DE HERRAMIENTAS DE PLASTICO', 8.50, 4.00, 3);

INSERT INTO PRODUCTO VALUES
('CAJ2', 'CAJA DE HERRAMIENTAS DE METAL', 12.30, 3.00, 2);

INSERT INTO PRODUCTO VALUES
('MAR1', 'MARTILLO PEQUEÑO', 3.50, 5, 10);

INSERT INTO PRODUCTO VALUES
('MAR2', 'MARTILLO GRANDE', 6.50, 12, 10);

INSERT INTO PRODUCTO VALUES
('TOR7', 'CAJA DE 100 TORNILLOS DEL 7', 0.80, 20, 100);

INSERT INTO PRODUCTO VALUES
('TOR9', 'CAJA DE 100 TORNILLOS DEL 9', 0.80, 25, 100);

INSERT INTO PRODUCTO VALUES
('TUE7', 'CAJA DE 100 TUERCAS DEL 7', 0.50, 40, 100);

INSERT INTO PRODUCTO VALUES
('TUE9', 'CAJA DE 100 TUERCAS DEL 9', 0.50, 54, 100);
```

4. Ejecuta esté código en MySql para añadir filas o registros a la tabla FACTURA.

Observa cómo los valores introducidos para el campo NIF tienen correspondencia con los valores introducidos en la tabla CLIENTE. De esta manera, se establece la asociación de cada factura con el cliente al que pertenece.

```
INSERT INTO FACTURA VALUES
(5440, '2017-09-05', true, '43434343A');

INSERT INTO FACTURA VALUES
(5441, '2017-09-05', true, '51639989K');

INSERT INTO FACTURA VALUES
(5442, '2017-09-06', false, '43434343A');

INSERT INTO FACTURA VALUES
(5443, '2017-10-10', true, '51639989K');

INSERT INTO FACTURA VALUES
(5444, '2017-10-13', true, '51664372R');

INSERT INTO FACTURA VALUES
(5445, '2017-10-14', false, '43434343A');
```

5. Ejecuta este código en MySql para añadir filas o registros a la tabla DETALLE.

Observa que los valores introducidos para los campos NUMERO y CODIGO tienen correspondencia con los valores introducidos en las tablas FACTURA y PRODUCTO. De esta manera, se establece la asociación de cada línea de detalle con la factura y el artículo al que pertenece.

```
INSERT INTO DETALLE VALUES (5440, 'CAJ2', 2, 12.3);
INSERT INTO DETALLE VALUES (5440, 'MAR1', 1, 3.50);
INSERT INTO DETALLE VALUES (5440, 'TOR7', 2, 0.80);
INSERT INTO DETALLE VALUES (5440, 'TUE7', 2, 0.50);
INSERT INTO DETALLE VALUES (5441, 'CAJ1', 1, 8.50);
INSERT INTO DETALLE VALUES (5442, 'CAJ1', 1, 8.50);
INSERT INTO DETALLE VALUES (5442, 'MAR1', 2, 3.50);
INSERT INTO DETALLE VALUES (5443, 'TOR7', 1, 0.80);
INSERT INTO DETALLE VALUES (5443, 'TUE7', 1, 0.50);
INSERT INTO DETALLE VALUES (5444, 'MAR2', 1, 12.0);
INSERT INTO DETALLE VALUES (5445, 'TOR7', 5, 0.80);
INSERT INTO DETALLE VALUES (5445, 'TOR9', 5, 0.80);
INSERT INTO DETALLE VALUES (5445, 'TUE7', 5, 0.50);
INSERT INTO DETALLE VALUES (5445, 'TUE9', 5, 0.50);
```

Ya tienes el terreno preparado para acometer con éxito la siguiente lección, donde podrás manipular la base de datos FERRETERIA desde un programa Java.

Despedida

Resumen

Has terminado la lección, repasemos los puntos más importantes que hemos tratado.

- Antes de crear la primera base de datos necesitamos instalar un **DBMS**.
- **MySQL Workbench** es una **herramienta visual de diseño integrada en el SGBD MySQL**, además de una interfaz de usuario para su gestión de forma más rápida y cómoda.
- El **lenguaje de definición de datos (DDL)** forma parte del estándar SQL y consta de tres sentencias que permiten la creación, eliminación y modificación de tablas, índices y otros objetos de la base de datos.
- Sentencias DDL: **CREATE**, **ALTER** y **DROP**.
- El **lenguaje de definición de datos** es vital para la creación del diseño físico de una base de datos, y requiere del conocimiento de los distintos tipos de datos a la hora de definir los atributos de las distintas tablas.
- Existen muchos **tipos de datos en el estándar SQL** y se clasifican en tres categorías:
  - Tipos de datos **numéricos**.
  - Tipos de datos **para almacenar fechas y horas.**
  - Tipos de datos **para almacenar texto.**