* Índice:

[1. Trabajar con Workbench 6.2 / 8 1](#_Toc53642884)

[**1.1** Conectarse al servidor: 1](#_Toc53642885)

[**1.2** Crear una base de datos en Workbench con el editor gráfico 1](#_Toc53642886)

[1.2.1 Tablas 6](#_Toc53642887)

[1.2.2 Relaciones 8](#_Toc53642888)

[**1.2.2.1** En qué orden hacer click en las tablas para crear la relación 9](#_Toc53642889)

[**1.2.2.2** Identifying Relationsship / Non-identifying relationship 9](#_Toc53642890)

[**1.2.2.3** Mandatory 10](#_Toc53642891)

[1.2.3 Formas de representar gráficamente las cardinalidades: UML, Crow's Foot 10](#_Toc53642892)

[**1.3** Crear un nuevo diagrama importándolo desde un script SQL que contiene los comandos DDL 12](#_Toc53642893)

[**1.4** Ejemplo de script generado por Workbench: 12](#_Toc53642894)

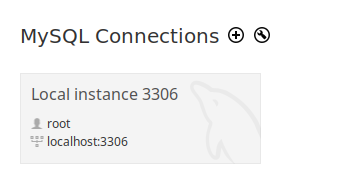
[1.4.1 Significado de algunos comandos que incluye el script 17](#_Toc53642895)

# Trabajar con Workbench 6.2 / 8

## Conectarse al servidor:

Database – Connect to Database

Workbench muestra en la parte inferior conexiones disponibles:



## Crear una base de datos en Workbench con el editor gráfico

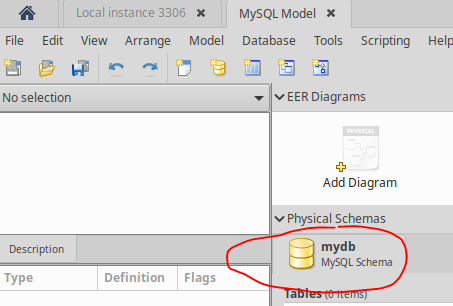
<https://dev.mysql.com/doc/workbench/en/wb-getting-started-tutorial-creating-a-model.html>

1. Crear un nuevo modelo

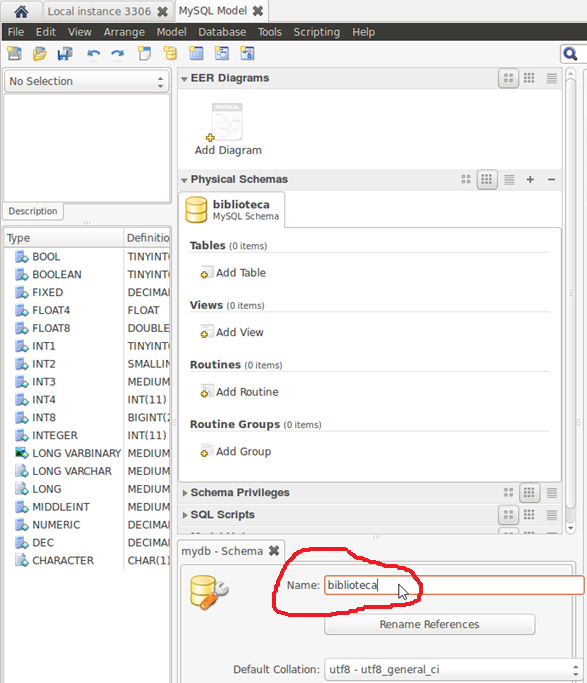
File > New Model

1. Cambiar el nombre del BD que se crea (por defecto se llama mydb):

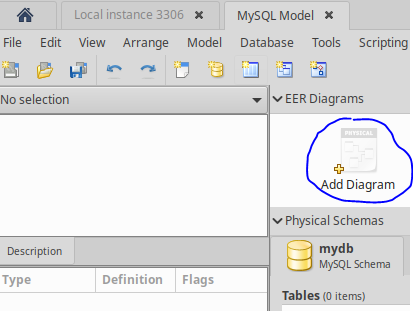
Doble click sobre la pestaña mydb



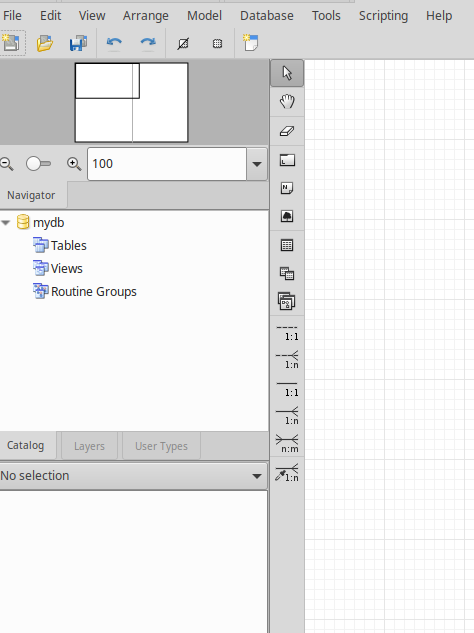
Cambiar el nombre de la BD en el panel inferior:



1. Crear un nuevo diagrama (dibujo) gráfico de la BD, pulsar el icono Add Diagram:



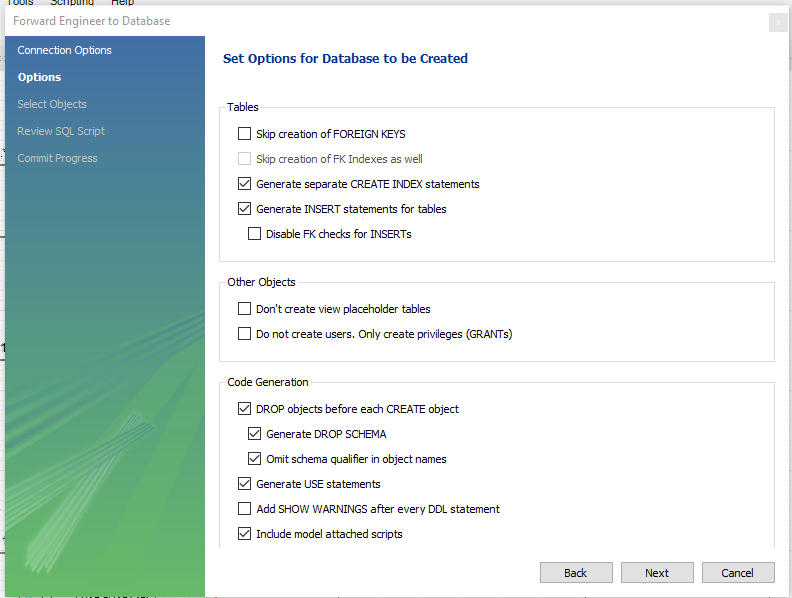
1. Se abre la ventana de diseño gráfico:



1. En esta ventana creamos las tablas y relaciones de nuestra BD. Se crea un esquema similar, pero no igual, al modelo Entidad Relación estándar. En un apartado inferior se indica como crear las tablas y sus relaciones.
2. Una vez creado el diagrama procedemos a crear la Base de datos físicamente (enviándole los comandos al servidor para que cree la BD):

Database - **Forward Engineer**

Lanza un asistente en el que escogemos las opciones deseadas, por ejemplo:



1. El asistente muestra el script que va a generar la BD. Podemos copiar el script al portapapeles para estudiarlo, modificarlo,...
2. Podemos ver un ejemplo de script generado por Workbench en un capítulo que se encuentra más abajo:

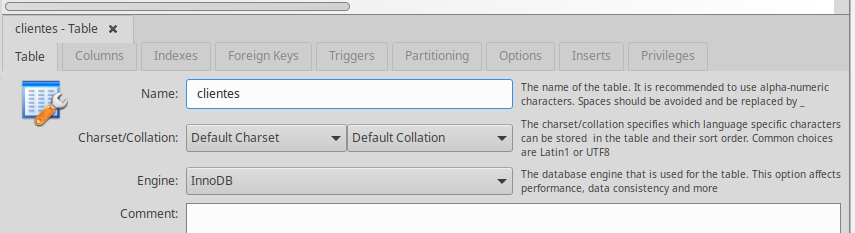
### Tablas

<https://dev.mysql.com/doc/workbench/en/wb-table-editor-columns-tab.html>

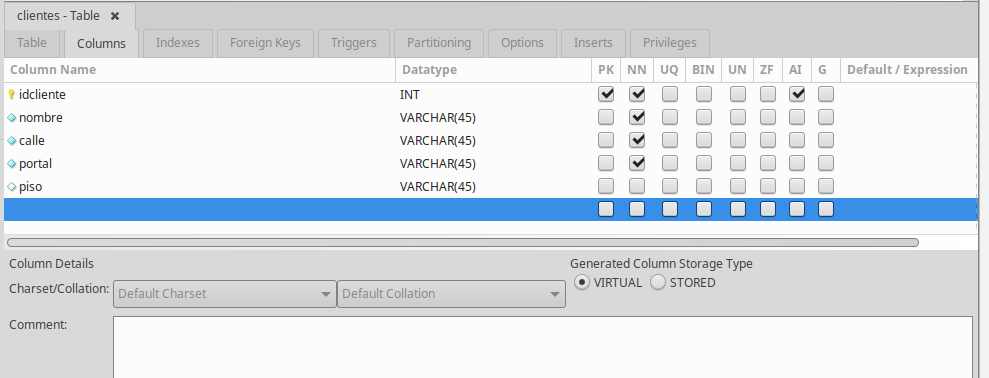
1. zer4Para crear una nueva tabla seguimos los siguientes pasos:
2. Icono para crear una nueva tabla:



1. Pulsar en el lienzo para depositar la tabla.
2. doble clic en la tabla para editar sus propiedades:
3. Cambiar el nombre de la tabla:



1. añadir atributos y definir sus propiedades.

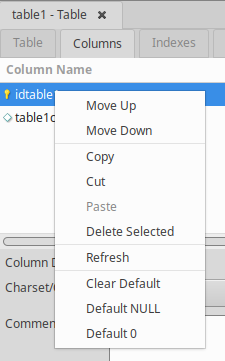


las propiedades disponibles son:



|  |  |  |
| --- | --- | --- |
| PK | PRIMARY KEY | CLAVE PRIMARIA |
| NN | NOT NULL |  |
| UQ | UNIQUE | CLAVE ALTERNATIVA |
| BIN | BINARY |  |
| UN | UNSIGNED | NUMERO POSITIVO |
| ZF | ZEROFILL | RELLENAR CON CEROS A LA IZQUIERDA |
| AI | AUTO\_INCREMENT |  |
| G | Generated Column |  |

1. Pulsando con el botón derecho sobre el nombre de un atributo se muestran propiedades extra:



To change the name, data type, default value, or comment of a column, double-click the value to edit it.

You can also add column comments to the Column Comment field. It is also possible to set the column collation, using the list in the **Column Details** panel.

To the left of the column name is an icon that indicates whether the column is a member of the primary key. If the icon is a small key, that column belongs to the primary key, otherwise the icon is a blue diamond or a white diamond. A blue diamond indicates the column has **NN** set. To add or remove a column from the primary key, double-click the icon. You can also add a primary key by checking the PRIMARY KEY check box in the Column Details section of the table editor.

If you wish to create a composite primary key you can select multiple columns and check the PK check box. However, there is an additional step that is required, you must click the Indexes tab, then in the Index Columns panel you must set the desired order of the primary keys.

### Relaciones

1. Los distintos botones se distinguen por tener línea continua (IDENTIFYING) o línea discontinua (NON IDENTIFYING).

* IDENTIFYING: **la FOREIGN KEY NO formará parte de la clave primaria de la tabla receptora.**
* NON IDENTIFYING: **la FOREIGN KEY SÍ formará parte de la clave primaria de la tabla receptora**

1. Estos son los botones que permiten dibujar las relaciones en Workbench

|  |  |  |
| --- | --- | --- |
|  | **nombre** | **Usarla cuando en el modelo E-R tenemos una relación de este tipo:** |
|  | **1:1** non identifying | **1:1**  **(la FOREIGN KEY NO formará parte de la clave primaria de la tabla receptora)** |
|  | **1:N** non identifying | **1:N**  **(la FOREIGN KEY NO formará parte de la clave primaria de la tabla receptora)** |
|  | **1:1** identifying | **1:1 débil por identificación o por existencia**  **(la FOREIGN KEY SÍ formará parte de la clave primaria de la tabla receptora)** |
|  | **1:N** identifying | **1:N débil por identificación o por existencia.**  **(la FOREIGN KEY SÍ formará parte de la clave primaria de la tabla receptora)** |
|  | **N:M** identifying | **N:M** |
|  | **1:N** identifying | usa columnas ya existentes y no crea nuevas columnas: no copia la PK de la tabla padre como FK en la clave hijo. |

#### En qué orden hacer clic en las tablas para crear la relación

* + **Relaciones 1:N, 1:1 del Workbench**: hacer clic en 1er lugar en la tabla receptora que recibirá la clave ajena y en 2º lugar en la tabla donante (la tabla que contiene la clave primaria que se propagará a la otra tabla receptora). Workbench creará automáticamente en la tabla hija un nuevo tributo (clave ajena) que se relacionará con la clave primaria de la tabla principal. Hay que configurar a mano las opciones ON UPDATE Y ON DELETE.
  + **Relaciones N:M del Workbench**: No importa el orden. Hacer clic primero en una de las tablas y después en la otra, Workbench creará automáticamente una tabla hijo para relacionar ambas tablas padre, se incluirán automáticamente las claves primarias de los padres. Hay que configurar a mano las opciones ON UPDATE Y ON DELETE.

#### Identifying Relationsship / Non-identifying relationship

En Workbench una relación puede ser de tipo:

* **IDENTIFYING** RELATIONSSHIP
* **NO IDENTIFYING** RELATIONSSHIP

1. **IDENTIFYING RELATIONSHIP:**

**La clave primaria de la tabla padre se añade como atributo (clave foránea) a la tabla hijo.**

**Además Workbench modifica la clave primaria de la tabla hijo añadiéndole ese atributo A LA CLAVE PRIMARIA**

**Se usa cuando en el modelor ER tenemos alguna de estas relaciones:**

**N:M**

**1:N débil por identificación o por existencia**

**1:1 débil por identificación o por existencia**

1. **NON-IDENTIFYING RELATIONSHIP**:

La clave primaria de la tabla padre se añade como atributo (clave foránea) a la tabla hijo.

**La clave ajena NO forma parte de la clave primaria de la tabla hijo.**

**Se usa cuando en el modelo ER tenemos alguna de estas relaciones:**

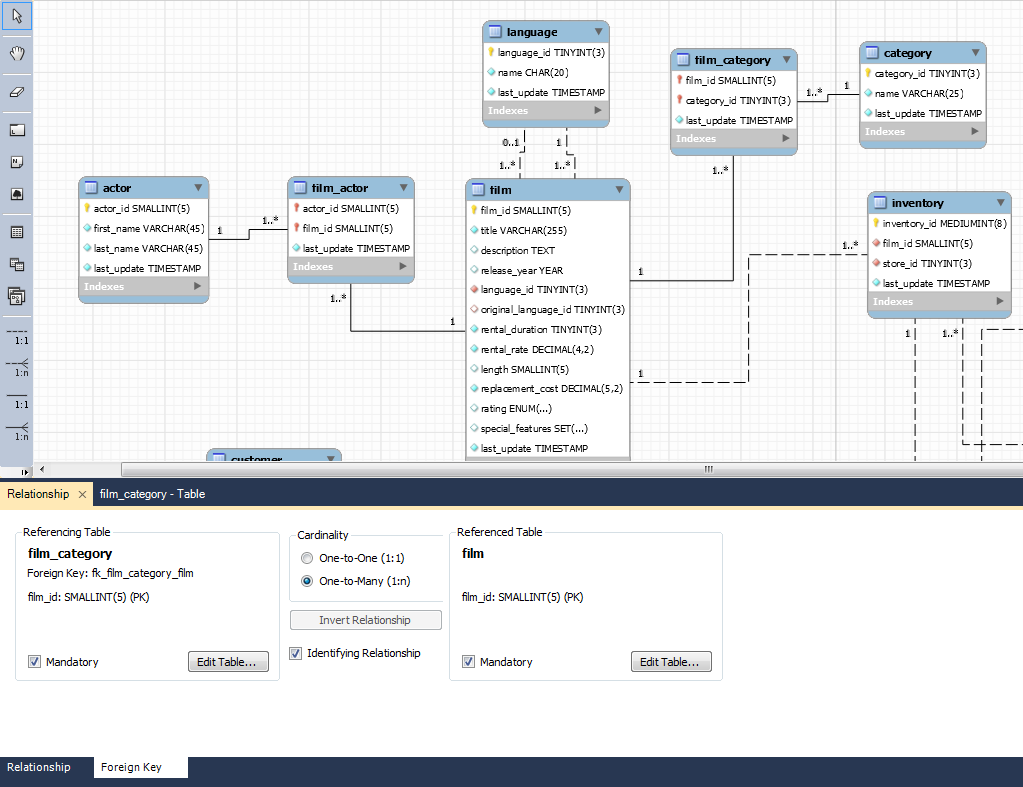
**1:N no débil**

**1:1 no débil**

#### Mandatory

Al editar las propiedades de una relación, haciendo doble click sobre la relación:

* **NO ES MANDATORY** : Cuando la **cardinalidad mínima** de una relación es **0**.
* **SÍ ES MANDATORY**: Cuando la cardinalidad mínima es **1**.

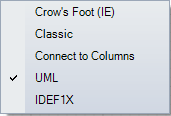


### Formas de representar gráficamente las cardinalidades: UML, Crow's Foot

Existen distintas formas de representar las cardinalidades:

Se cambia en:

Model – Relationship Notation:



Nosotros utilizaremos **UML** por ser el más extendido.

1. Por defecto usa **Crow’s Foot** (Pata de cuervo), antiguamente era el más habitual.

|  |  |
| --- | --- |
| Crow’s Foot (pata de cuervo) | **UML** |
| Classic | Connect to columns |
|  |  |

## Crear un nuevo diagrama importándolo desde un script SQL que contiene los comandos DDL

File – Import – Reverse Engineer MySQL Create Script

La opción de realizar el Reverse Engineer facilita la comprensión de una base de datos que existe físicamente en el servidor. El resultado es un diagrama gráfico con la estructura de la base de datos que nosotros escojamos.

Cuando tenemos un script con el código SQL que crea una base de datos es un poco difícil entender la estructura de la base de datos. Lo que podemos hacer es: utilizar el script para crear físicamente la base de datos en el servidor y posteriormente realizar el Reverse Engineer.

## Ejemplo de script generado por Workbench:

-- MySQL Workbench Forward Engineering

SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0;

SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0;

SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='TRADITIONAL,ALLOW\_INVALID\_DATES';

-- -----------------------------------------------------

-- Schema restaurante

-- -----------------------------------------------------

DROP SCHEMA IF EXISTS `restaurante` ;

-- -----------------------------------------------------

-- Schema restaurante

-- -----------------------------------------------------

CREATE SCHEMA IF NOT EXISTS `restaurante` DEFAULT CHARACTER SET utf8 COLLATE utf8\_general\_ci ;

USE `restaurante` ;

-- -----------------------------------------------------

-- Table `Empleados`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `Empleados` ;

CREATE TABLE IF NOT EXISTS `Empleados` (

`dni` VARCHAR(9) NOT NULL,

`nombre` VARCHAR(45) NOT NULL,

`apelllidos` VARCHAR(45) NOT NULL,

`direccion` VARCHAR(45) NOT NULL,

`ciudad` VARCHAR(45) NOT NULL,

`cp` VARCHAR(5) NOT NULL,

`telefono` VARCHAR(13) NOT NULL,

`fecha\_alta` DATE NOT NULL,

`categoria` VARCHAR(45) NOT NULL,

`salario` DECIMAL(8,2) NOT NULL,

PRIMARY KEY (`dni`))

ENGINE = InnoDB;

CREATE UNIQUE INDEX `dni\_UNIQUE` ON `Empleados` (`dni` ASC);

-- -----------------------------------------------------

-- Table `cocineros`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `cocineros` ;

CREATE TABLE IF NOT EXISTS `cocineros` (

`dni` VARCHAR(9) NOT NULL,

`puesto` VARCHAR(45) NOT NULL,

`especialidad` VARCHAR(45) NOT NULL,

PRIMARY KEY (`dni`),

CONSTRAINT `fk\_cocineros\_Empleados`

FOREIGN KEY (`dni`)

REFERENCES `Empleados` (`dni`)

ON DELETE CASCADE

ON UPDATE CASCADE)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `camareros`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `camareros` ;

CREATE TABLE IF NOT EXISTS `camareros` (

`dni` VARCHAR(9) NOT NULL,

`turno` VARCHAR(45) NOT NULL,

`experiencia` TEXT NULL,

`supervisor` VARCHAR(9) NULL,

PRIMARY KEY (`dni`),

CONSTRAINT `fk\_camareros\_Empleados1`

FOREIGN KEY (`dni`)

REFERENCES `Empleados` (`dni`)

ON DELETE CASCADE

ON UPDATE CASCADE,

CONSTRAINT `fk\_camareros\_camareros1`

FOREIGN KEY (`supervisor`)

REFERENCES `camareros` (`dni`)

ON DELETE SET NULL

ON UPDATE CASCADE)

ENGINE = InnoDB;

CREATE INDEX `fk\_camareros\_camareros1\_idx` ON `camareros` (`supervisor` ASC);

-- -----------------------------------------------------

-- Table `administradores`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `administradores` ;

CREATE TABLE IF NOT EXISTS `administradores` (

`dni` VARCHAR(9) NOT NULL,

`cargo` VARCHAR(45) NOT NULL,

PRIMARY KEY (`dni`),

CONSTRAINT `fk\_administradores\_Empleados1`

FOREIGN KEY (`dni`)

REFERENCES `Empleados` (`dni`)

ON DELETE CASCADE

ON UPDATE CASCADE)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `comedores`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `comedores` ;

CREATE TABLE IF NOT EXISTS `comedores` (

`idcomedor` INT NOT NULL AUTO\_INCREMENT,

`nombre` VARCHAR(45) NOT NULL,

`localizacion` VARCHAR(45) NOT NULL,

`capacidad` INT(11) NOT NULL,

`num\_mesas` INT(11) NOT NULL,

PRIMARY KEY (`idcomedor`))

ENGINE = InnoDB;

CREATE UNIQUE INDEX `nombre\_UNIQUE` ON `comedores` (`nombre` ASC);

-- -----------------------------------------------------

-- Table `mesas`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `mesas` ;

CREATE TABLE IF NOT EXISTS `mesas` (

`idcomedor` INT NOT NULL,

`num\_mesa` INT NOT NULL,

`num\_asientos` INT NOT NULL,

`dni\_camarero\_atiende` VARCHAR(9) NULL,

PRIMARY KEY (`idcomedor`, `num\_mesa`),

CONSTRAINT `fk\_mesas\_comedores1`

FOREIGN KEY (`idcomedor`)

REFERENCES `comedores` (`idcomedor`)

ON DELETE CASCADE

ON UPDATE CASCADE,

CONSTRAINT `fk\_mesas\_camareros1`

FOREIGN KEY (`dni\_camarero\_atiende`)

REFERENCES `camareros` (`dni`)

ON DELETE SET NULL

ON UPDATE CASCADE)

ENGINE = InnoDB;

CREATE INDEX `fk\_mesas\_comedores1\_idx` ON `mesas` (`idcomedor` ASC);

CREATE INDEX `fk\_mesas\_camareros1\_idx` ON `mesas` (`dni\_camarero\_atiende` ASC);

-- -----------------------------------------------------

-- Table `reservas`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `reservas` ;

CREATE TABLE IF NOT EXISTS `reservas` (

`idreserva` INT NOT NULL AUTO\_INCREMENT,

`fecha\_hora` DATETIME NOT NULL,

`comida\_o\_cena` TINYINT(1) NOT NULL,

`nombre\_cliente` VARCHAR(45) NOT NULL,

`fecha\_hora\_reservada` DATETIME NOT NULL,

`num\_personas` SMALLINT NOT NULL,

`otros` VARCHAR(500) NULL,

PRIMARY KEY (`idreserva`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `reservas\_mesas`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `reservas\_mesas` ;

CREATE TABLE IF NOT EXISTS `reservas\_mesas` (

`idreserva` INT NOT NULL,

`idcomedor` INT NOT NULL,

`num\_mesa` INT NOT NULL,

PRIMARY KEY (`idreserva`, `idcomedor`, `num\_mesa`),

CONSTRAINT `fk\_reservas\_has\_mesas\_reservas1`

FOREIGN KEY (`idreserva`)

REFERENCES `reservas` (`idreserva`)

ON DELETE CASCADE

ON UPDATE CASCADE,

CONSTRAINT `fk\_reservas\_has\_mesas\_mesas1`

FOREIGN KEY (`idcomedor` , `num\_mesa`)

REFERENCES `mesas` (`idcomedor` , `num\_mesa`)

ON DELETE NO ACTION

ON UPDATE CASCADE)

ENGINE = InnoDB;

CREATE INDEX `fk\_reservas\_has\_mesas\_mesas1\_idx` ON `reservas\_mesas` (`idcomedor` ASC, `num\_mesa` ASC);

CREATE INDEX `fk\_reservas\_has\_mesas\_reservas1\_idx` ON `reservas\_mesas` (`idreserva` ASC);

-- -----------------------------------------------------

-- Table `platos`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `platos` ;

CREATE TABLE IF NOT EXISTS `platos` (

`idplato` INT NOT NULL AUTO\_INCREMENT,

`nombre` VARCHAR(45) NULL,

`tipo` VARCHAR(45) NOT NULL,

`descripción` TEXT NULL,

`precio` DECIMAL(8,2) NOT NULL,

`platoscol` VARCHAR(45) NOT NULL,

PRIMARY KEY (`idplato`))

ENGINE = InnoDB;

CREATE UNIQUE INDEX `nombre\_UNIQUE` ON `platos` (`nombre` ASC);

-- -----------------------------------------------------

-- Table `platos\_mesas`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `platos\_mesas` ;

CREATE TABLE IF NOT EXISTS `platos\_mesas` (

`idplato` INT NOT NULL,

`idcomedor` INT NOT NULL,

`num\_mesa` INT NOT NULL,

PRIMARY KEY (`idplato`, `idcomedor`, `num\_mesa`),

CONSTRAINT `fk\_platos\_has\_mesas\_platos1`

FOREIGN KEY (`idplato`)

REFERENCES `platos` (`idplato`)

ON DELETE NO ACTION

ON UPDATE CASCADE,

CONSTRAINT `fk\_platos\_has\_mesas\_mesas1`

FOREIGN KEY (`idcomedor` , `num\_mesa`)

REFERENCES `mesas` (`idcomedor` , `num\_mesa`)

ON DELETE NO ACTION

ON UPDATE CASCADE)

ENGINE = InnoDB;

CREATE INDEX `fk\_platos\_has\_mesas\_mesas1\_idx` ON `platos\_mesas` (`idcomedor` ASC, `num\_mesa` ASC);

CREATE INDEX `fk\_platos\_has\_mesas\_platos1\_idx` ON `platos\_mesas` (`idplato` ASC);

-- -----------------------------------------------------

-- Table `proveedores`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `proveedores` ;

CREATE TABLE IF NOT EXISTS `proveedores` (

`idproveedor` INT NOT NULL AUTO\_INCREMENT,

`nombre` VARCHAR(45) NOT NULL,

`direccion` VARCHAR(45) NOT NULL,

`cp` VARCHAR(45) NOT NULL,

`telefono` VARCHAR(13) NOT NULL,

`fax` VARCHAR(13) NULL,

`conctacto` VARCHAR(45) NULL,

PRIMARY KEY (`idproveedor`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `productos`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `productos` ;

CREATE TABLE IF NOT EXISTS `productos` (

`idproducto` INT NOT NULL,

`descripcion` VARCHAR(45) NOT NULL,

`categoria` VARCHAR(45) NOT NULL,

`precio\_ud` DECIMAL(8,2) NOT NULL,

`stock` FLOAT NOT NULL,

`unidad\_base` VARCHAR(45) NOT NULL,

`idproveedor` INT NULL,

PRIMARY KEY (`idproducto`),

CONSTRAINT `fk\_productos\_proveedores1`

FOREIGN KEY (`idproveedor`)

REFERENCES `proveedores` (`idproveedor`)

ON DELETE SET NULL

ON UPDATE CASCADE)

ENGINE = InnoDB;

CREATE INDEX `fk\_productos\_proveedores1\_idx` ON `productos` (`idproveedor` ASC);

-- -----------------------------------------------------

-- Table `platos\_productos`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `platos\_productos` ;

CREATE TABLE IF NOT EXISTS `platos\_productos` (

`idplato` INT NOT NULL,

`idproducto` INT NOT NULL,

PRIMARY KEY (`idplato`, `idproducto`),

CONSTRAINT `fk\_platos\_has\_productos\_platos1`

FOREIGN KEY (`idplato`)

REFERENCES `platos` (`idplato`)

ON DELETE CASCADE

ON UPDATE CASCADE,

CONSTRAINT `fk\_platos\_has\_productos\_productos1`

FOREIGN KEY (`idproducto`)

REFERENCES `productos` (`idproducto`)

ON DELETE NO ACTION

ON UPDATE CASCADE)

ENGINE = InnoDB;

CREATE INDEX `fk\_platos\_has\_productos\_productos1\_idx` ON `platos\_productos` (`idproducto` ASC);

CREATE INDEX `fk\_platos\_has\_productos\_platos1\_idx` ON `platos\_productos` (`idplato` ASC);

-- -----------------------------------------------------

-- Table `facturas`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `facturas` ;

CREATE TABLE IF NOT EXISTS `facturas` (

`idfactura` INT NOT NULL AUTO\_INCREMENT,

`fecha` DATE NOT NULL,

`idreserva` INT NOT NULL,

PRIMARY KEY (`idfactura`),

CONSTRAINT `fk\_facturas\_reservas1`

FOREIGN KEY (`idreserva`)

REFERENCES `reservas` (`idreserva`)

ON DELETE CASCADE

ON UPDATE CASCADE)

ENGINE = InnoDB;

CREATE INDEX `fk\_facturas\_reservas1\_idx` ON `facturas` (`idreserva` ASC);

SET SQL\_MODE=@OLD\_SQL\_MODE;

SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS;

SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS;

### Significado de algunos comandos que incluye el script

<https://dev.mysql.com/doc/workbench/en/workbench-faq.html>

How does MySQL Workbench increase import performance?

When a model is exported (Database, Forward Engineer...), some MySQL server variables are temporarily set to enable faster SQL import by the server. The statements added at the start of the code are:

#Al principio del script generado por Workbench se encuentran estos 3 comandos:

SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0;

SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0;

SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='TRADITIONAL,ALLOW\_INVALID\_DATES';

#cada uno de estos 3 comandos tiene esta estructura:

1) guarda el valor vigente de la variable de estado (@@UNIQUE\_CHECKS) en una variable temporal creada por nosotros (@OLD\_UNIQUE\_CHECKS)

2) cambia el valor de la variable de estado para acelerar el proceso de importación de datos(UNIQUE\_CHECKS=0)

#La función de cada uno de esos 3 comandos es el siguiente:

SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0;: Determines whether InnoDB performs duplicate key checks. Import is much faster for large data sets if this check is not performed. For additional information, see [unique\_checks](https://dev.mysql.com/doc/refman/5.7/en/server-system-variables.html#sysvar_unique_checks).

SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0;: Determines whether the server should check that a referenced table exists when defining a foreign key. Due to potential circular references, this check must be turned off for the duration of the import, to permit defining foreign keys. For additional information, see [foreign\_key\_checks](https://dev.mysql.com/doc/refman/5.7/en/server-system-variables.html#sysvar_foreign_key_checks).

SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='TRADITIONAL';: Sets [SQL\_MODE](https://dev.mysql.com/doc/refman/5.7/en/sql-mode.html) to [TRADITIONAL](https://dev.mysql.com/doc/refman/5.7/en/sql-mode.html#sqlmode_traditional), causing the server to operate in a more restrictive mode, and [ALLOW\_INVALID\_DATES](https://dev.mysql.com/doc/refman/5.7/en/sql-mode.html#sqlmode_allow_invalid_dates), causing dates to not be fully validated.

#Al final del script las variables de estado recuperan el valor que tenían antes de ser modificadas:

SET SQL\_MODE=@OLD\_SQL\_MODE;

SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS;

SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS;