


# Create MySQL Database - MySQL Workbench Tutorial

 [clevertechie.com/php/98/create-mysql-database-mysql-workbench-tutorial](https://clevertechie.com/php/98/create-mysql-database-mysql-workbench-tutorial)

PHP | 19 Feb, 2017 | Clever Techie

Google +3 13 0

## Create MySQL Database - MySQL Workbench Tutorial

In this lesson we're going to be using MySQL Workbench to model and create a MySQL database with many-to-many relational tables.

```
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';
```

```
CREATE SCHEMA IF NOT EXISTS `movies` DEFAULT CHARACTER SET utf8 COLLATE  
utf8_general_ci ;
```

```
USE `movies` ;
```

```
CREATE TABLE IF NOT EXISTS `movies`.`movies` (
```

```
`id` INT NOT NULL AUTO_INCREMENT COMMENT "
```

```
`title` VARCHAR(45) NOT NULL COMMENT "
```

```
`year` INT NOT NULL COMMENT "
```

```
`image_url` VARCHAR(255) NOT NULL COMMENT "
```

```
`certificate` VARCHAR(45) NULL COMMENT "
```

```
`runtime` INT NULL COMMENT "
```

```
`imdb_rating` FLOAT NULL COMMENT "
```

```
`description` TEXT NULL COMMENT "
```

```
`metascore` INT NULL COMMENT "
```

```
`votes` INT NULL COMMENT "
```

```
`gross` INT NULL COMMENT "
```

```
PRIMARY KEY (`id`) COMMENT "
```

```
UNIQUE INDEX `title_UNIQUE` (`title` ASC) COMMENT ")
```

ENGINE = InnoDB;

```
CREATE TABLE IF NOT EXISTS `movies`.`directors` (  
  `id` INT NOT NULL AUTO_INCREMENT COMMENT "  
  `name` VARCHAR(45) NOT NULL COMMENT "  
  `about` TEXT NULL COMMENT "  
  PRIMARY KEY (`id`) COMMENT "  
  UNIQUE INDEX `name_UNIQUE` (`name` ASC) COMMENT "  
ENGINE = InnoDB;
```

```
CREATE TABLE IF NOT EXISTS `movies`.`stars` (  
  `id` INT NOT NULL AUTO_INCREMENT COMMENT "  
  `name` VARCHAR(45) NOT NULL COMMENT "  
  `about` TEXT NULL COMMENT "  
  PRIMARY KEY (`id`) COMMENT "  
  UNIQUE INDEX `name_UNIQUE` (`name` ASC) COMMENT "  
ENGINE = InnoDB;
```

```
CREATE TABLE IF NOT EXISTS `movies`.`genres` (  
  `id` INT NOT NULL AUTO_INCREMENT COMMENT "  
  `name` VARCHAR(45) NOT NULL COMMENT "  
  PRIMARY KEY (`id`) COMMENT "  
  UNIQUE INDEX `name_UNIQUE` (`name` ASC) COMMENT "  
ENGINE = InnoDB;
```

```
CREATE TABLE IF NOT EXISTS `movies`.`movies_directors` (  
  `movies_id` INT NOT NULL COMMENT "  
  `directors_id` INT NOT NULL COMMENT "  
  PRIMARY KEY (`movies_id`, `directors_id`) COMMENT "  
  INDEX `fk_movies_has_directors_directors1_idx` (`directors_id` ASC) COMMENT "  
  INDEX `fk_movies_has_directors_movies_idx` (`movies_id` ASC) COMMENT "  
  CONSTRAINT `fk_movies_has_directors_movies`
```

```

FOREIGN KEY (`movies_id`)
REFERENCES `movies`.`movies` (`id`)
ON DELETE NO ACTION
ON UPDATE NO ACTION,
CONSTRAINT `fk_movies_has_directors_directors1`
FOREIGN KEY (`directors_id`)
REFERENCES `movies`.`directors` (`id`)
ON DELETE NO ACTION
ON UPDATE NO ACTION)
ENGINE = InnoDB;

CREATE TABLE IF NOT EXISTS `movies`.`movies_stars` (
`movies_id` INT NOT NULL COMMENT "",
`stars_id` INT NOT NULL COMMENT "",
PRIMARY KEY (`movies_id`, `stars_id`) COMMENT "",
INDEX `fk_movies_has_stars_stars1_idx` (`stars_id` ASC) COMMENT "",
INDEX `fk_movies_has_stars_movies1_idx` (`movies_id` ASC) COMMENT "",
CONSTRAINT `fk_movies_has_stars_movies1`
FOREIGN KEY (`movies_id`)
REFERENCES `movies`.`movies` (`id`)
ON DELETE NO ACTION
ON UPDATE NO ACTION,
CONSTRAINT `fk_movies_has_stars_stars1`
FOREIGN KEY (`stars_id`)
REFERENCES `movies`.`stars` (`id`)
ON DELETE NO ACTION
ON UPDATE NO ACTION)
ENGINE = InnoDB;

CREATE TABLE IF NOT EXISTS `movies`.`movies_genres` (

```

```

`movies_id` INT NOT NULL COMMENT ",
`genres_id` INT NOT NULL COMMENT ",
PRIMARY KEY (`movies_id`, `genres_id`) COMMENT ",
INDEX `fk_movies_has_genres_genres1_idx` (`genres_id` ASC) COMMENT ",
INDEX `fk_movies_has_genres_movies1_idx` (`movies_id` ASC) COMMENT ",
CONSTRAINT `fk_movies_has_genres_movies1`
FOREIGN KEY (`movies_id`)
REFERENCES `movies`.`movies` (`id`)
ON DELETE NO ACTION
ON UPDATE NO ACTION,
CONSTRAINT `fk_movies_has_genres_genres1`
FOREIGN KEY (`genres_id`)
REFERENCES `movies`.`genres` (`id`)
ON DELETE NO ACTION
ON UPDATE NO ACTION)
ENGINE = InnoDB;
SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;

```

Take a look at the following MySQL database model. what we're gonna be doing is creating a database and tables which will hold all the information from from the IMDB (internet movie database) website. The movies table is going to be the main table which will store all information about the movies. The directors, stars, and genres tables will hold data about those guys.

We're also going to create three joining tables which will be our many-to-many relationships between the main movies table and directors, stars and genres. The reason we're doing this, is because movies can have many stars, directors and genres, and vice versa.

In the article on [web scraping imdb.com using PHP](#) we have created a regular expression script which scrapes internet movie database ([imdb.com](#)), go to the link to download complete source code for the script. The script is one big function and it parses out the movies and returns all the data in a nicely formatted array.

Let's look at an example of how this function can be used. Four parameters must be passed to the `scrape_imdb()` function. The starting year of the movie, the ending year (when the movies were released) followed by the page start and finally page end as integer values.

Note: before you guys start using the script, be advised that it's kind of a hacking thing that we're doing by scraping all the movie data from [imdb.com](http://imdb.com) website and if you overdo it, they might not like it and they might ban your IP so when you're just testing out the script be careful with how many pages you specify. Start with a very small range (a couple of pages).

Here is what happens when we execute the script. It will parse out all the HTML on the movie pages and then print out this big array with all the data in it. If you right click on the page where the script was executed and choose "View Source", you'll see how the array is organized nicely with all the movie information.

Let's go ahead and create the database model which will hold all the data now using MySQL Workbench (a free program that you can download [here](#)). When you first open the Workbench, create a new connection by clicking on the + icon.

I'm going to name my connection localhost and put "root" for the username. Then press the storage vault button and enter your password. You can then click on "test the connection" and you should get the message: "successfully made the MySQL connection." At this point, click OK, and then double click on this newly created connection to connect to your MySQL database server.

Now go to File->New Model and double click on Add Diagram. This is the part of the program where we're going to be designing our database model. You will see a toolbar on the left side along with all the tools available which we can use to model the database. Go ahead and double-click on the New Table icon which will place a new table in the diagram. When the table shows up on the diagram, double-click on it, the new box will show up at the bottom of the screen where we can rename our table as well as specify all the column names and their appropriate data types.

Name this table that we have just created "movies", and then for the very first column, enter "id" which will be the unique identifier for this table. This id will also act as primary key which is going to auto increment so make sure the primary key and auto-increment check boxes are both checked as well as not null (NN). Now let's go ahead and enter the rest of the column names.

Look at this screen shot again and enter all the column names along with the data types shown. Make the title field unique, because we only want to have unique movies in our table by checking the "UN" box while the title column is selected. Also, check not null (NN) boxes by title, year and image\_url since we're always going to be inserting those, so they can't be empty. The rest of the fields can be null, and in fact they should be, because a lot of our movies will be missing the remaining data.

At this point the movies table is complete, go ahead and save your work.

Now, using the same steps, you can go ahead and add directors, stars and genres tables on your own. Make sure you specify id's for all three tables, make them primary keys, auto-increment them and make sure they're not null. And now have all the data tables which will store all the movie information from the data array.

Now we're ready to specify what the relationship is going to be between these tables because, as mentioned in the beginning of this article, many movies can have many directors, stars and genres and vice versa, which means it's a many-to-many relationship.

To create many-to-many relationship in Workbench, and this is what makes it really awesome, on the toolbar, click on the icon as shown in the screen shot below, then click on the movies table once and finally click on the target table starting with "directors". You will notice that Workbench has created a new table called "movies\_has\_directors", this is the table which will keep track of what movies are related to what directors and vice versa. I prefer to name a table like this "movies\_directors", so I've renamed mine by you can leave yours like it is if you'd like. Now follow the same steps to create many-to-many relationships between movies<->stars and movies<->genres.

The last step is to specify physical database name which MySQL workbench will create all of our tables in. You should see the "Catalog Tree" window in your Workbench, and you should also see a database which is named "mydb" under it, go ahead and rename it to "movies". Also, create "movies" database on your localhost by either logging in to phpMyAdmin or via MySQL command prompt.

At this point we have successfully modeled and created the MySQL database diagram, now let's physically create it on the server. Make sure you save the diagram by going to File->Save Model. Now go ahead and click on the "Database" in the menu and then choose "Forward Engineer" which is just a fancy way of saying "create this database now on our localhost".

A new window will pop up, and then it will give you a bunch of options, just keep clicking next all the way through until the end, until you get to "forward engineering progress" window. If you get all the messages with check boxes by them, congratulations! You have successfully "forward engineered a MySQL database with three many-to-many relational tables"! (how about that one to impress your next date?)

PS: You should login to phpMyAdmin to make sure the tables have been created in the movies database. Post your comments below!