

MySQL Data Modeling Tutorial

Contents

1.	Overview	1
2.	Create a Data Model	2
3.	Add New Tables to the Data Model	3
4.	Add Foreign Key Relationship	4
5.	Saving and Printing the Data Model	6
6.	Foreward Engineering, Reverse Engineering	6

1. Overview

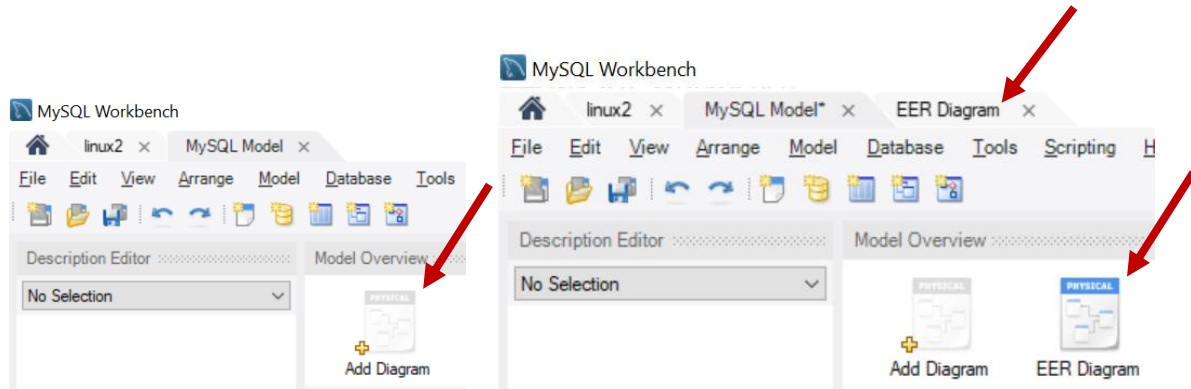
In this tutorial, you will

- create a new data model adding database tables to the model,
- modify the design of the tables,
- add a non-identifying (foreign key) relationship between two tables,
- save the data model (you can open up later from MySQL Workbench),
- print the data model to a PDF
- (optional) generate a SQL script from the data model (script creates tables and relationships if you copy/pasate into a SQL query window).

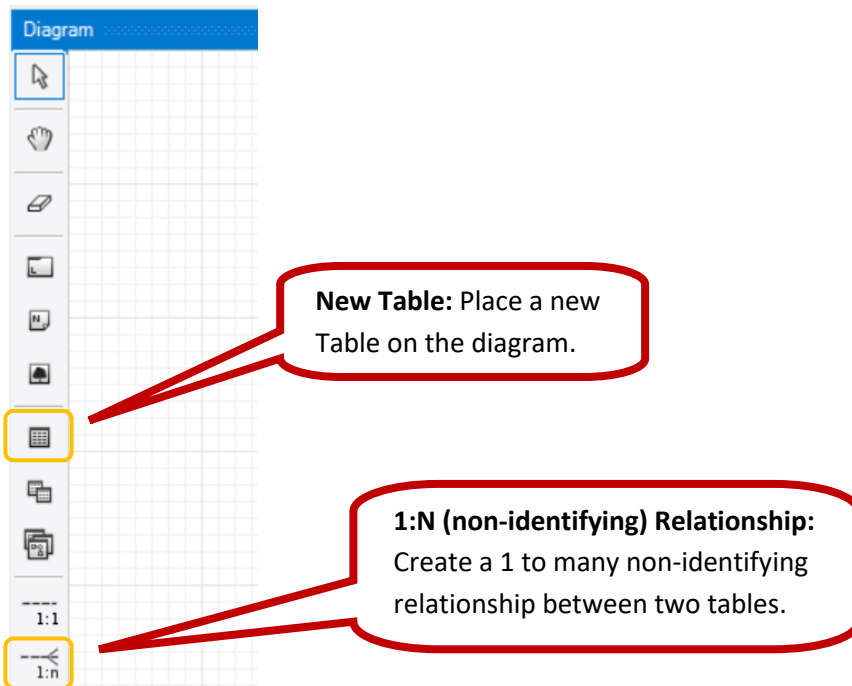
2. Create a Data Model

Create a blank data model from MySQL Workbench by

- Selecting **File – New Model** from the menu.
- Then double click on the “Add Diagram” icon.
- This creates the EER (Enhanced Entity Relationship) Diagram (see below right) then automatically open up that EER Diagram in another tab.

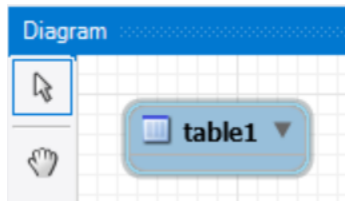


Inside the diagram there are two important icons to use – hover over each to see how the icons are labelled.



3. Add New Tables to the Data Model

- Click on the New Table icon (icon that was shown in the previous diagram), then click on the diagram where you want to place the new table. You should see something like this in your EER Diagram.



- To change the table definition of table1, right click on it and select “Edit Table” (I think they should have named this option “Alter Table”, but...)

The screenshot shows the 'table1 - Table' dialog box. It has a title bar with 'table1 - Table' and a close button. Below the title bar, there is a 'Table Name' field containing 'table1' and a 'Schema' dropdown menu set to 'mydb'. Below these fields is a table with columns for 'Column Name', 'Datatype', and various attributes: PK, NN, UQ, B, UN, ZF, AI, G, and 'Default/Expression'. All attribute checkboxes are currently unchecked.

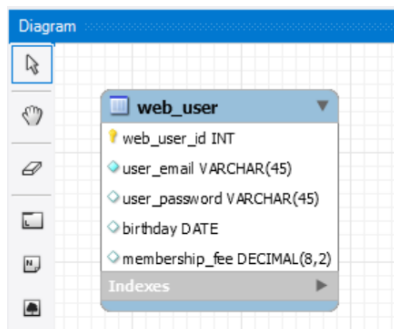
Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Rename the table (to be “web_user”) and add in the following columns and column attributes. Hover over each attribute heading to see what each means (PK is Primary Key, NN is Not Null, UQ is Unique, AI is Auto-Increment).

The screenshot shows the 'web_user - Table' dialog box. It has a title bar with 'web_user - Table' and a close button. Below the title bar, there is a 'Table Name' field containing 'web_user' and a 'Schema' dropdown menu set to 'sallyk'. Below these fields is a table with columns for 'Column Name', 'Datatype', and various attributes: PK, NN, UQ, B, UN, ZF, AI, G, and 'Default/Expression'. The following table represents the data shown in the screenshot:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
web_user_id	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
user_email	VARCHAR(55)	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
user_password	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
birthday	DATE	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
membership_fee	DECIMAL(12,2)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL

When you close out the Alter Table pane, you should see your EER diagram looking like this:



Add a second table to the diagram and then modify its table design as shown (change table name, add column names and specify the column attributes).

web_user - Table

user_role - Table







Table Name:

Schema: **sallyk**



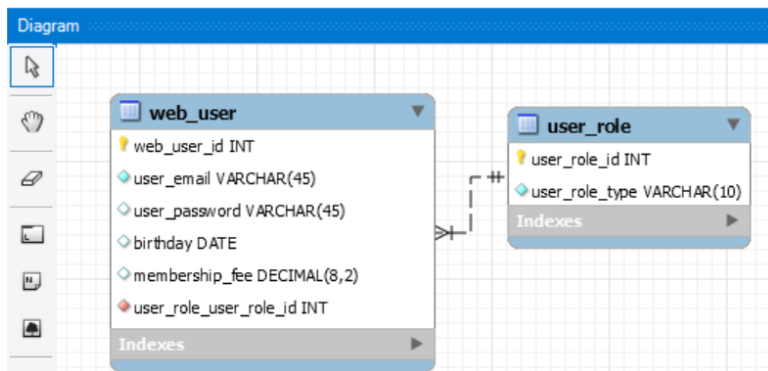
Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
 user_role_id	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="text"/>
 user_role_type	VARCHAR(10)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="text"/>

4. Add Foreign Key Relationship

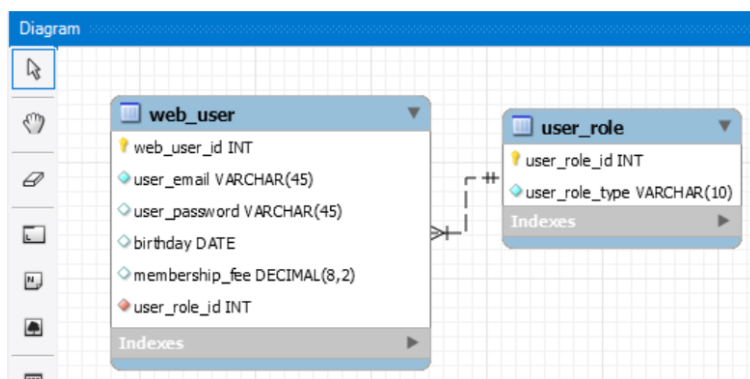
Now we will create the foreign key from web_user to user_role.

- Click on , the “One-to-Many Non Identifying Relationship” icon , then
- Click on the web_user table (the foreign key or “many” side of the relationship) then click on the user_role table (the table being referenced, the “one” side of the relationship).

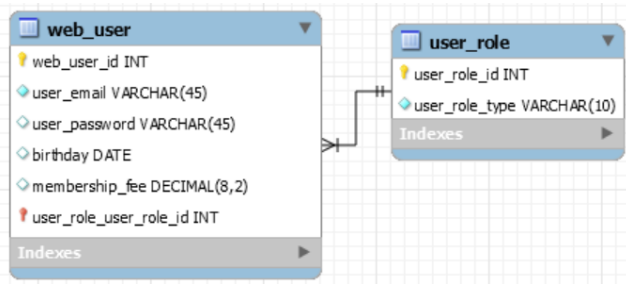
This adds the foreign key field and creates the foreign key constraint (but only in the data model, not in the database). The “crows feet” symbol (attached to the web_user table) indicates the many side of the relationship (many web_users can all point to one user_role).



The field name that MySQL WB assigns to the newly created foreign key is strange. Edit the web_user table and change the name of that foreign key field to a normal name: user_role_id (not user_role_user_role_id):



Note: if your diagram looks like this (web_user.user_role_id showing as key, relationship line solid not dotted), then **you selected the wrong type of relationship** (see next paragraph for explanation). Right click the relationship, delete it and create it again.



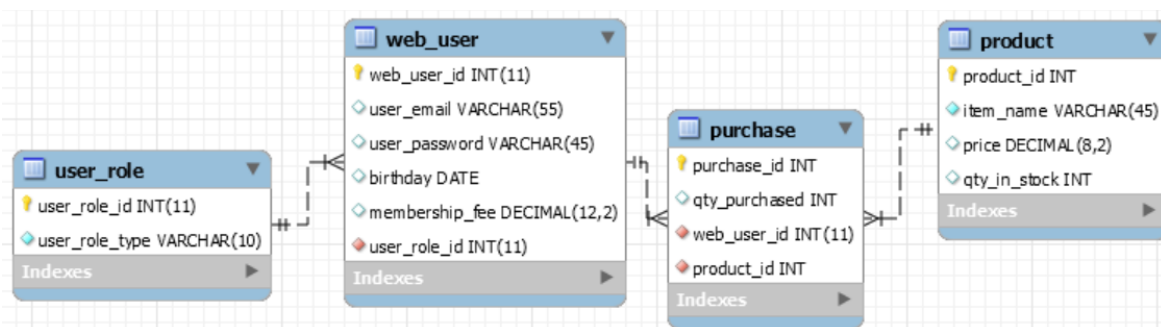
When you create an **identifying relationship**, a foreign key field is added (no problem here) but you are also making that foreign key field part of the primary key. Typically it is not desired to have a compound primary key (less efficient, more trouble to code – always having to specify two fields to uniquely identify a record in a database table).

So, just use the **non-identifying relationship** (where the foreign key field does NOT become part of the primary key).

5. More Tables and Relationships

Add two more tables and two more relationships as described below:

- Add a table to the model, modify its name to “**product**” and enter in the columns similar to what is shown in the data model below. The product table is like the “other” table that you have been asked to come up with.
- Add another table to the model, modifying the name to “**purchase**” and enter all the non FK fields. In other words, do not type in web_user_id and product_id – these will be automatically created when you add the relationships. The purchase table is an example of an associative table (that implements a many to many relationship).
- Add a 1 to many non identifying **relationship from purchase to product** and another such **relationship from purchase to web_user**. Foreign keys enable each purchase to “knows” data about the web_user who made the purchase and about the product that was purchased.



6. Saving and Printing the Data Model

When you have finished, save your data model and print it to a PDF so that (in a later homework) you can publish the model to your web site.

- File – Save Model (saves to a “.mwb” file which you can later open from from MySQL Workbench)
- File – Print to File (PDF)

Note: there is sometimes a bug when you try to open a data model (.mwb file) from MySQL Workbench. If you get this: **Error unserialising GRT data when opening a .mwb file**, close out MySQL Workbench, the open up the .mwb file from the File Manager. Although you get a warning message, the data model does open up properly.

7. Foreward Engineering, Reverse Engineering

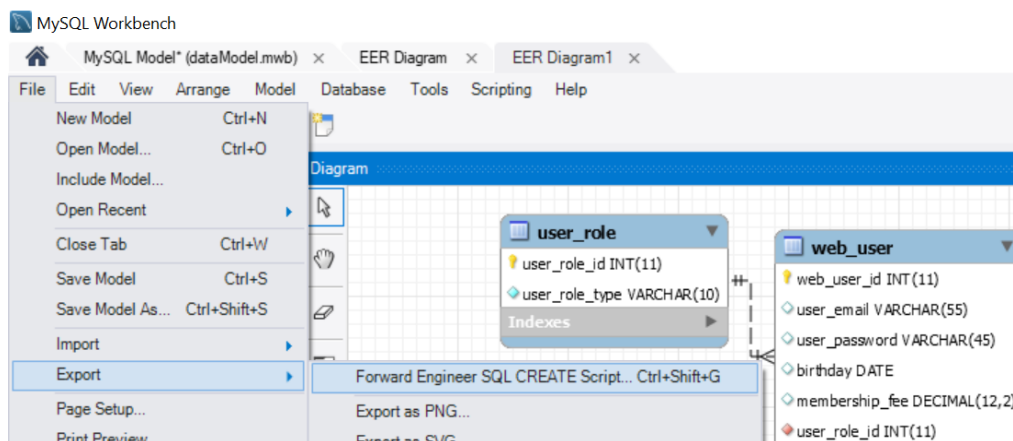
With MySQL Workbench (or most other database management system User Interfaces), we can “**reverse engineer**” a data model which means to create a data model from an existing database schema. This would be very helpful, for example, if you got a job and were asked to write reports on a data model that you knew nothing about.

You can also “**forward engineer**” from a data model to a database schema. This means you could design your tables in the data modelling tool, then ask the tool to create (in the database schema) all of the the tables and relationships that are in the data model. The forward engineer option can also “synchronize changes” meaning that it compares what the schema has against what the data model has and makes whatever adjustments it needs to make both the same (either updating the model or the schema, whichever you prefer).

Due to the HUGE number of student databases at Temple, MySQL Workbench times out whenever it tries to synchronize changes between a database schema and a data model, but please keep in mind that, under normal circumstances, it would be very **easy and valuable** to forward and reverse engineer data models.

Although we cannot synchronize changes between a data model and a database schema, we CAN have MySQL WB generate a SQL CREATE Script for us – which we could copy/paste into a Query window and run that way. Here is how you can do that:

- File – Export – Forward Engineer (generates SQL CREATE Script)



The script generated from the above operation is shown on the next page – when the data model has only web_user and user_role and a single relationship.

NOTE: You could copy the red part of the script below and paste it into a MySQL Workbench query window – to create web_user and user_role (and PKs/FKs), if:

- (1) you have set your DB as the default database (double click on it to bold it) and
- (2) you delete every occurrence of **`mydb`**. from the script ('mydb' is not the right database schema).

-- MySQL Script generated by MySQL Workbench

-- Fri Jan 11 13:50:13 2019

```
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 mydb

```
CREATE SCHEMA IF NOT EXISTS `mydb` DEFAULT CHARACTER SET utf8 ;
USE `mydb` ;
```

-- Table `mydb`.`user_role`

```
CREATE TABLE IF NOT EXISTS `mydb`.`user_role` (
  `user_role_id` INT NOT NULL,
  `user_role_type` VARCHAR(10) NOT NULL,
  PRIMARY KEY (`user_role_id`),
  UNIQUE INDEX `user_role_id_UNIQUE` (`user_role_id` ASC))
ENGINE = InnoDB;
```

-- Table `mydb`.`web_user`

```
CREATE TABLE IF NOT EXISTS `mydb`.`web_user` (
  `web_user_id` INT NOT NULL AUTO_INCREMENT,
  `user_email` VARCHAR(45) NOT NULL,
  `user_password` VARCHAR(45) NULL,
  `birthday` DATE NULL,
  `membership_fee` DECIMAL(8,2) NULL,
  `user_role_user_role_id` INT NOT NULL,
  PRIMARY KEY (`web_user_id`, `user_role_user_role_id`),
  UNIQUE INDEX `user_email_UNIQUE` (`user_email` ASC),
  INDEX `fk_web_user_user_role_idx` (`user_role_user_role_id` ASC),
  CONSTRAINT `fk_web_user_user_role`
    FOREIGN KEY (`user_role_user_role_id`)
      REFERENCES `mydb`.`user_role` (`user_role_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;
```