

MySQL Week 2 Exercises

Background

You are in the process of creating an application that will perform CRUD (Create, Read, Update, and Delete) operations on a MySQL database. In the week 1 exercises you wrote code to connect to a MySQL database using the Java Database Connectivity (JDBC) API. In these exercises, you will diagram the database tables using Draw.io to create an Entity-Relationship Diagram. You will then write the SQL statements to create the five project tables. Lastly, you will use a MySQL client (DBeaver) to create the tables.

Objectives

In these exercises, you will:

- Learn how to create an Entity-Relationship Diagram (ERD) in Draw.io with entities and relationship lines.
- Learn about crow's foot notation and how to apply that knowledge in an Entity Relationship Diagram.
- Apply your knowledge of DROP TABLE and CREATE TABLE statements to create tables using a MySQL client (DBeaver),

Important

In the exercises below, you will see this icon: . This means to take a screen shot or snip showing the results of the action or the code in the editor.

Exercises

Use the column definitions in the table below for the Entity-Relationship Diagram (ERD) and the Create Table statements.

Table	Column	Data Type	Nullable	Comment
project	project_id	int	No	primary key
	project_name	varchar(128)	No	
	estimated_hours	decimal(7,2)	Yes	
	actual_hours	decimal(7,2)	Yes	
	difficulty	int	Yes	
	notes	text	Yes	
material	material_id	int	No	primary key
	project_id	int	No	foreign key
	material_name	varchar(128)	No	
	num_required	int	Yes	
	cost	decimal(7,2)	Yes	

step	step_id	int	No	primary key
	project_id	int	No	foreign key
	step_text	text	No	
	step_order	int	No	
category	category_id	int	No	primary key
	category_name	varchar(128)	No	
project_category	project_id	int	No	foreign key, unique key with category_id
	category_id	int	No	foreign key, unique key with project_id

Entity-Relationship Diagram

Documenting a project is an essential skill so that the project will make sense to others who want to know about the project. This includes network diagrams, Entity-Relationship Diagrams, well-commented code and readme files.

In this section, you will create an Entity-Relationship Diagram. This diagram will contain the five table entities and show the relationships between the tables. A good ERD can quickly orient future developers to the data that the application will work with.

Use your knowledge learned in the videos and follow the instructions in this section to create an Entity-Relationship Diagram of the DIY Projects schema.

1. Follow the instructions in the Week 2 Installation Instructions found in the resources packet to either download Draw.io or to use the online tool.
2. In Draw.io, create a new drawing and expand "Entity Relation" in the tool palette on the left.
3. Use Draw.io to create an Entity-Relationship Diagram. Save the file. The file must be uploaded to your GitHub repository for Week 2. Note that it should look similar to the diagram below.




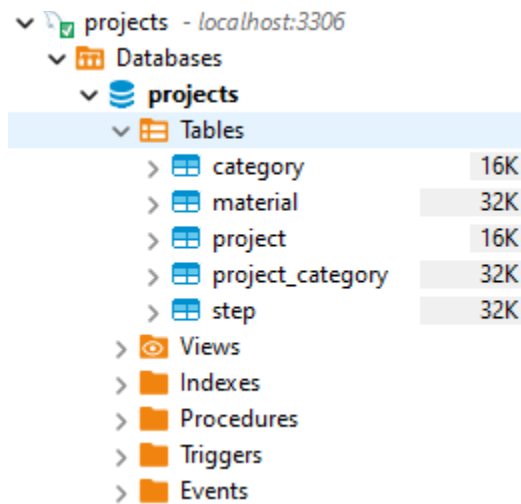
Project Schema

In this section, you will create a new file and write DROP TABLE and CREATE TABLE SQL statements for the five tables in the DIY Projects schema. The goal for this section is to copy the SQL statements from the file, drop them into the DBeaver SQL editor, and have DBeaver send the instructions to the MySQL server so that the five tables are created with no errors.

Creating the tables is the first step to having the application read from and write to them. Next week you will begin that process.

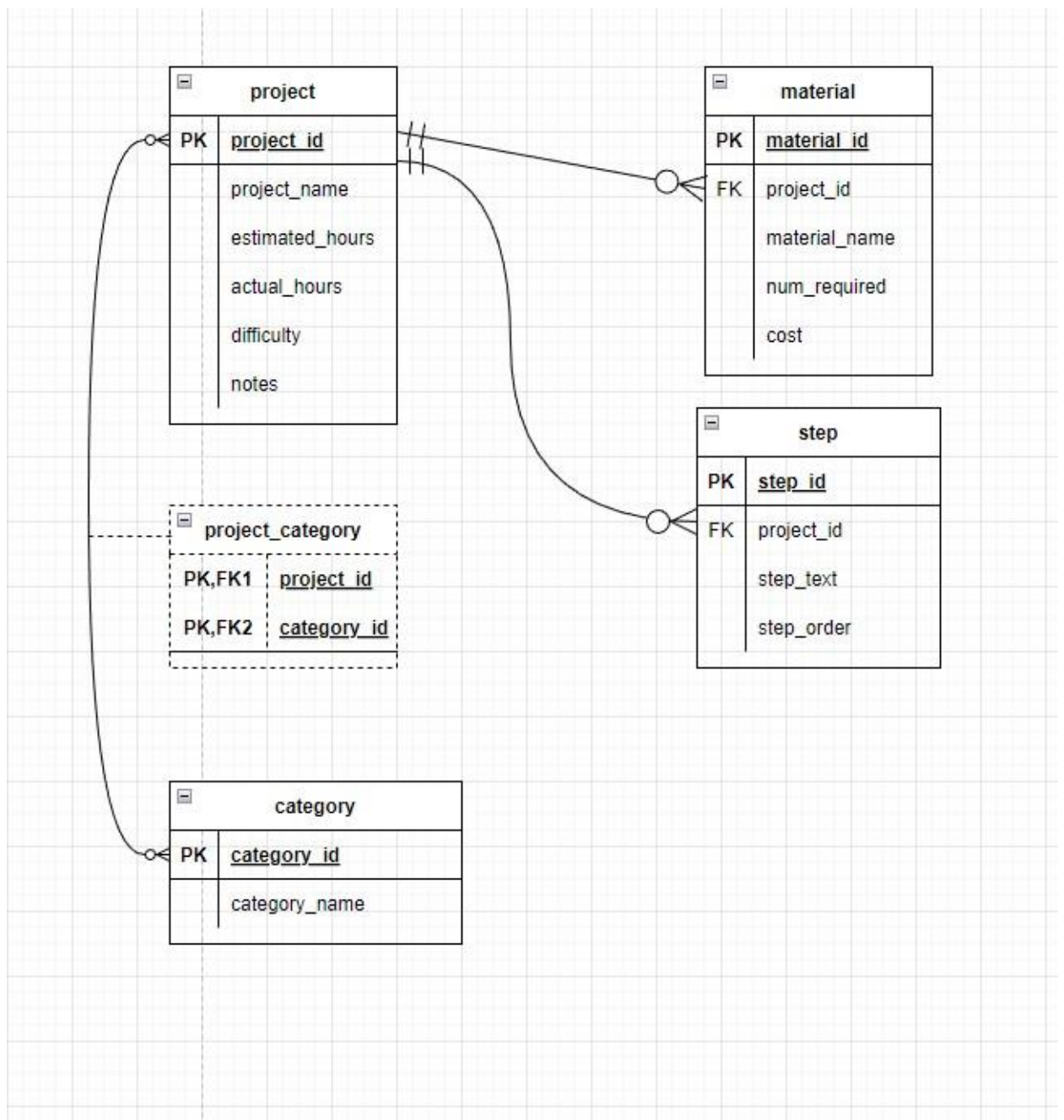
1. In the `mysql-java` project in Eclipse, create a file named `"projects-schema.sql"` in the `src/main/resources` directory.
2. Add DROP TABLE statements at the top of the file to drop the tables in the correct order. There can be some variation but the tables with dependencies (foreign key references to other tables) must be dropped first. The tables should only be dropped if they exist.
3. Write the CREATE TABLE statements in the inverse of the order that they were dropped. Include the following:

- a. Auto-increment the primary key columns.
 - b. Primary key statements.
 - c. Foreign key statements with appropriate ON DELETE CASCADE.
4. Remember to close each DROP TABLE and CREATE TABLE statement with a semicolon.
5. Paste the SQL into the DBeaver SQL editor. Run all DROP TABLE and CREATE TABLE statements. Include a screen shot of the created tables as shown in DBeaver's connection explorer (leftmost panel).  It should look like this:



6. Push your project to GitHub.

▼ projects - localhost:3306	
▼ Databases	
▼ projects	
▼ Tables	
> category	16K
> material	32K
> project	16K
> project_category	32K
> project_categoty	16K
> step	32K
> Views	
> Indexes	
> Procedures	
> Triggers	
> Events	
> Users	
> Administer	
> System Info	



```

1 DROP TABLE IF EXISTS project_category;
2 DROP TABLE IF EXISTS category;
3 DROP TABLE IF EXISTS step;
4 DROP TABLE IF EXISTS material;
5 DROP TABLE IF EXISTS project;
6
7 CREATE TABLE project(
8 project_id INT AUTO_INCREMENT NOT NULL,
9 project_name VARCHAR(128) NOT NULL,
10 estimated_hours DECIMAL(7, 2),
11 actual_hours DECIMAL(7, 2),
12 difficulty INT,
13 notes TEXT,
14 PRIMARY KEY (project_id)
15);
16
17 CREATE TABLE material(
18 material_id INT AUTO_INCREMENT NOT NULL,
19 project_id INT NOT NULL,
20 material_name VARCHAR(128) NOT NULL,
21 num_required INT,
22 cost DECIMAL(7, 2),
23 PRIMARY KEY (material_id),
24 FOREIGN KEY (project_id) REFERENCES project (project_id) ON DELETE CASCADE
25);

```

```

26
27 CREATE TABLE step(
28 step_id INT AUTO_INCREMENT NOT NULL,
29 project_id INT NOT NULL,
30 step_text TEXT NOT NULL,
31 step_order INT NOT NULL,
32 PRIMARY KEY (step_id),
33 FOREIGN KEY (project_id) REFERENCES project (project_id) ON DELETE CASCADE
34);
35
36 CREATE TABLE category(
37 category_id INT AUTO_INCREMENT NOT NULL,
38 category_name VARCHAR(128) NOT NULL,
39 PRIMARY KEY (category_id)
40);
41
42 CREATE TABLE project_category(
43 project_id INT AUTO_INCREMENT NOT NULL,
44 category_id INT NOT NULL,
45 FOREIGN KEY (project_id) REFERENCES project (project_id) ON DELETE CASCADE,
46 FOREIGN KEY (category_id) REFERENCES category (category_id) ON DELETE CASCADE,
47 UNIQUE KEY (project_id, category_id)
48);
49

```