

We have been developing a menu-driven application that demonstrates how to perform CRUD (Create, Read, Update, and Delete) operations on a project database. Thus far, we have learned how to create a connection to a MySQL database and how to insert records into a table. Then, we learned how to query for a list of records and for all details on a single record. In these exercises, we will learn the final two parts of CRUD: Updating and Deleting.

Objectives

In these exercises, you will:

Modify project details using the UPDATE statement.

Delete a project and all child rows using the DELETE statement.

Observe that using ON DELETE CASCADE automatically deletes child rows with a foreign key relationship.

Use the return value from PreparedStatement.executeUpdate() to determine if a row was updated or deleted.

Important

In the exercises below, you will see this icon: .

This means to make sure that you include this functionality in your video showcase.

Instructions

URL to GitHub Repository:<https://github.com/zachuff/mysql-java>

URL to Public Link of your Video:<https://youtu.be/1VRQfLvmx5Y>

Instructions :

1. Follow the Exercises below to complete this assignment.

- In Eclipse, or an IDE of your choice, write the code that accomplishes the objectives listed below. Ensure that the code compiles and runs as directed.
- Create a new repository on GitHub for this week's assignment and push your completed code to this dedicated repo, including your entire Maven Project Directory (e.g., mysql-java) and any .sql files that you create. In addition, screenshot your ERD and push the screenshot to your GitHub repo.
- Include the functionality into your Video when you see:
- Create a video showcasing your work:
- In this video: record and present your project verbally while showing the results of the working project. Don't forget to include the requested functionality, indicated by:
- Easy way to Create a video: Start a meeting in Zoom, share your screen, open Eclipse with the code and your Console window, start recording & record yourself describing and running the program showing the results.
- Your video should be a maximum of 5 minutes.
- Upload your video with a public link.
- Easy way to Create a Public Video Link: Upload your video recording to YouTube with a public link.

2. In addition, please include the following in your Coding Assignment Document:

- The URL for this week's GitHub repository.
- The URL of the public link of your video.

3. Save the Coding Assignment Document as a .pdf and do the following:

- Push the .pdf to the GitHub repo for this week.
 - Upload the .pdf to the LMS in your Coding Assignment Submission.
-

Exercises

In these exercises, you will modify project contents and delete a project. You have already learned how to perform the Create and Read part of CRUD operations. This will complete your CRUD experience by adding Update and Delete.

You should try to follow the instructions as best you can. Suggestions for variable and method names are given – you can take those suggestions or not as you wish. If you deviate from the instructions, try to stick to Java best practices by naming methods and variables for what they do or what they are. If you get stuck, see the Solutions section at the end of this document.

Update project details

In this section, you will update a project row. There is a lot remaining to be done for an industrious student: adding materials, steps, and categories, maintaining categories; modifying materials and steps; changing step order, etc. In this section, you will gain part of that skill set. Follow these steps to update the project details.

Changes to the menu application

In this section, you will make changes to the menu application to allow the user to update project details. You will add a new menu selection and add a method call in the switch statement. Finally, you will create a method to get project detail changes from the user and call the project service to make the modifications.

In this section, you will be working in `ProjectsApp.java`.

Add the line "4) Update project details" to the list of operations.

Add case 4 to the switch statement and call method `updateProjectDetails()`. Let Eclipse create the method for you.

In method `updateProjectDetails()`:

Check to see if `curProject` is null. If so, print a message "\nPlease select a project." and return from the method.

For each field in the `Project` object, print a message along with the current setting in `curProject`. Here is an example:

Create a new `Project` object. If the user input for a value is not null, add the value to the `Project` object. If the value is null, add the value from `curProject`. Repeat for all `Project` variables.

Set the project ID field in the `Project` object to the value in the `curProject` object.

Call `projectService.modifyProjectDetails()`. Pass the `Project` object as a parameter. Let Eclipse create the method for you in `ProjectService.java`.

Reread the current project to pick up the changes by calling `projectService.fetchProjectById()`. Pass the project ID obtained from `curProject`.

Save all files. At this point you should have no compilation errors.

Changes to the project service

In this section you will make changes to the project service. The service is responsible for calling the DAO to update the project details and to return those details to the caller. If the project cannot be found, the service throws an exception. The service method is called by the menu application class, and results are returned to that class.

In this section you will be working in `ProjectService.java`.

In the method `modifyProjectDetails()`,

Call `projectDao.modifyProjectDetails()`. Pass the `Project` object as a parameter. The DAO method returns a boolean that indicates whether the UPDATE operation was successful. Check the return value. If it is false, throw a `DbException` with a message that says the project does not exist.

Let Eclipse create the `modifyProjectDetails()` method for you in `ProjectDao.java`. Save all files.

At this point you should have no compilation errors.

Changes to the project DAO

Now, complete the code in the project DAO to update the project details. The method structure is similar to the `insertProject()` method. You will write the SQL UPDATE statement with the parameter placeholders. Then, obtain a `Connection` and start a transaction. Next, you will obtain a `PreparedStatement` object and set the six parameter values. Finally, you will call `executeUpdate()` on the `PreparedStatement` and commit the transaction.

The difference in this method and the insert method is that you will examine the return value from `executeUpdate()`. The `executeUpdate()` method returns the number of rows affected by the UPDATE operation. Since a single row is being acted on (comparing to the primary key in the WHERE clause guarantees this), the return value should be 1. If it is 0 it means that no rows were acted on and the primary key value (project ID) is not found. So, the method returns true if `executeUpdate()` returns 1 and false if it returns 0.

In this section you will be working in `ProjectDao.java`.

In `modifyProjectDetails()`, write the SQL statement to modify the project details. Do not update the project ID – it should be part of the WHERE clause. Remember to use question marks as parameter placeholders.

Obtain the `Connection` and `PreparedStatement` using the appropriate try-with-resource and catch blocks. Start and rollback a transaction as usual. Throw a `DbException` from each catch block.

Set all parameters on the `PreparedStatement`. Call `executeUpdate()` and check if the return value is 1. Save the result in a variable.

1. Commit the transaction and return the result from `executeUpdate()` as a boolean. At this point there should be no compilation errors.

Test it

First, test the application by updating project details without selecting a project. You should receive an error message. Include in your video a shot of the console showing the selections and error message.

Next, select a project. Then, select "Update project details". Enter new project details and update the project. Include in your video a shot of the console showing the selected project details, the data you input, and the new project details. It should look something like this:

Delete a project

In this section, you will write the code to delete a project. This will require a little preparation. You must verify that ON DELETE CASCADE in the CREATE TABLE statements works to remove child rows (materials, steps, and project_category rows). This means that you will need to make sure the project has child records. Since the application does not currently add the child rows, you will need to add them using a MySQL client like DBeaver or the MySQL CLI. Hint: you may want to test this a couple of times. If you add some insert statements at the end of projects-schema.sql, you can simply load and execute the SQL statements as many times as you want. In the following example, not all CREATE TABLE statements are shown.

Here are the steps for DBeaver:

Right-click on the connection name. Select "SQL Editor" / "Recent SQL script". The editor should open and it should have the name <projects> in the top tab (assuming the connection is named "projects").

Paste the entire contents of projects-schema.sql into the DBeaver editor. Select all the text in the editor. Right-click in the editor. Select "Execute" / "Execute SQL Script"

Changes to the menu application

In this section you will add code to display a new menu operation to the user ("Delete a project"). Then you will add the case statement to the switch. Next, you will write the method that will list the projects to delete, get the project ID from the user, and call the service to delete the project.

In this section you will be working in ProjectsApp.java.

Add a new option: "5) Delete a project" to the list of operations.

Add case 5 to the switch statement. Call the method deleteProject(). Let Eclipse create the method for you.

In method deleteProject():

Call method listProjects().

Ask the user to enter the ID of the project to delete.

Call projectService.deleteProject() and pass the project ID entered by the user.

Print a message stating that the project was deleted. (If it wasn't deleted, an exception is thrown by the service class.)

Add a check to see if the project ID in the current project is the same as the ID entered by the user. If so, set the value of curProject to null.

Have Eclipse create the deleteProject() method in the project service.

Save all files. At this point there should be no compilation errors.

Changes to the project service

The deleteProject() method in the service is very similar to the modifyProjectDetails() method.

You will call the deleteProject() method in the DAO class and check the boolean return value. If the return value is false, a DbException is thrown with a message that the project with the given

ID does not exist. The exception will be picked up by the exception handler in the application menu class.

In this section you will be working in ProjectService.java.

Call deleteProject() in the project DAO. Pass the project ID as a parameter. The method returns a boolean. Test the return value from the method call. If it returns false, throw a DbException with a message stating that the project doesn't exist.

Have Eclipse create the deleteProject() method in the ProjectDao class.

Save all files. At this point there should be no compilation errors.

Changes to the project DAO

The deleteProject() method in the DAO is very similar to the modifyProjectDetails() method. You will first create the SQL DELETE statement. Then, you will obtain the Connection and PreparedStatement, and set the project ID parameter on the PreparedStatement. Then, you will call executeUpdate() and verify that the return value is 1, indicating a successful deletion.

Finally, you will commit the transaction and return success or failure.

In this section you will be working in ProjectDao.java.

In the method deleteProject():

Write the SQL DELETE statement. Remember to use the placeholder for the project ID in the WHERE clause.

Obtain a Connection and a PreparedStatement. Start, commit, and rollback a transaction in the appropriate sections.

Set the project ID parameter on the PreparedStatement.

Return true from the menu if executeUpdate() returns 1.

Test it

In this section, you will perform two tests. The first test will delete a project with an unknown project ID and the second test will actually perform the deletion.

Delete with invalid ID

This tests the delete operation with an invalid project ID.

Run the application.

Select "Delete a project". When you are prompted to enter a project ID to delete, enter an invalid ID.

Include a shot of the console showing that an error was generated, and that the application handled it gracefully. Here is a sample:

Delete a project

In this section you will test that you can do an actual deletion.

Run the application.

Select "Delete a project". When you are prompted to enter a project ID to delete, enter a valid ID.

List the projects to show that the project was deleted with no errors.

Include a shot of the console. Here is a sample:

Verify that materials, steps, and project_category rows were deleted as well. Use DBeaver or the MySQL CLI for this. The child rows should have been deleted due to the ON DELETE CASCADE in the foreign key statements.

Solutions

In these solutions, only the changed parts of the code are shown.

ProjectsApp.java

ProjectService.java

ProjectDao.java