SQLite PHP: Deleting Data

Summary: in this tutorial, we will show you steps of deleting data in an SQLite database using PHP PDO.

Steps for deleting data from PHP PDO

To delete data in a table in the SQLite database from a PHP application, you use these steps:

1. Connect to the SQLite database (https://www.sqlitetutorial.net/sqlite-php/connect/) by creating an instance of the PDO class.
2. Prepare a DELETE statement (https://www.sqlitetutorial.net/sqlite-delete/) for execution.
3. Pass values to the statement using the `bindValue()` method of the `PDOStatement` object.
4. Call the `execute()` method of the `PDOStatement` object to execute the delete statement.
5. Call the `rowCount()` method of the `PDOStatement` to get the number of rows deleted.

SQLite PHP: delete data examples

We will use the `projects` and `tasks` that we created in the previous tutorial (https://www.sqlitetutorial.net/sqlite-php/create-tables/) for the demonstration.

First, we create the `SQLiteDelete` class in the `app` folder.

The following `deleteTask()` method deletes a task by `task_id`. The method returns the number of rows deleted by calling the `rowCount()` method of the `PDOStatement` object.

We use the `bindValue()` method to pass task id value to the `DELETE` statement.

```
/**
 * Delete a task by task id
 * @param int $taskId
 * @return int the number of rows deleted
 */
public function deleteTask($taskId) {
    $sql = 'DELETE FROM tasks '
            . 'WHERE task_id = :task_id';

    $stmt = $this->pdo->prepare($sql);
    $stmt->bindValue(':task_id', $taskId);

    $stmt->execute();

    return $stmt->rowCount();
}
```

The `deleteTaskByProject()` method deletes all tasks associated with a project specified by the project id.

In this method, instead of using the `bindValue()` method to pass values to the `DELETE` statement, we pass the input values as an array to the `execute()` method.

```
/**
 * Delete all tasks associated with a project
 * @param int $projectId
 * @return int the number of rows deleted
 */
public function deleteTaskByProject($projectId) {
    $sql = 'DELETE FROM tasks '
```

```
                . 'WHERE project_id = :project_id';

        $stmt = $this->pdo->prepare($sql);

        $stmt->execute([':project_id' => $projectId]);

        return $stmt->rowCount();
    }
```

The following method deletes a project specified by a project id. It also returns the number of the rows in the projects table deleted.

In addition, due to the foreign key constraint, all the tasks associated with the project are also deleted. The `rowCount()` function does not consider the deleted rows in the tasks table.

```
    /**
     * Delete the project by project id
     * @param int $projectId
     * @return int the number of rows deleted
     */
    public function deleteProject($projectId) {
        $sql = 'DELETE FROM projects '
                . 'WHERE project_id = :project_id';

        $stmt = $this->pdo->prepare($sql);

        $stmt->execute([':project_id' => $projectId]);

        return $stmt->rowCount();
    }
```

Second, we create the `index.php` script to test the `SQLiteDelete` class.

```
<?php

require 'vendor/autoload.php';

```

```
use App\SQLiteConnection as SQLiteConnection;
use App\SQLiteDelete as SQLiteDelete;

$pdo = (new SQLiteConnection())->connect();
$sqlite = new SQLiteDelete($pdo);

// delete task id 1
$taskId = 1;
$rowDeleted = $sqlite->deleteTask($taskId);

echo 'The number of rows deleted: ' . $rowDeleted . '<br>';

// delete task associated with a project id 1
$projectId = 1;
$sqlite->deleteTaskByProject($projectId);

echo 'The number of task in the project #' . $projectId . ' deleted: '

// delete project with id 1 and also its associated tasks
$projectId = 2;
$sqlite->deleteProject(2);
echo 'The number of project deleted: ' . $rowDeleted . '<br>';
```

The following is the output of the script:

```
The number of rows deleted: 1
The number of task in the project #1 deleted: 1
The number of project deleted: 1
```

In this tutorial, we have shown you how to delete data in one or more tables in the SQLite database using PHP PDO.