SQLite PHP: Working with BLOB Data

Summary: in this tutorial, you will learn how to manage BLOB data in SQLite database using PHP PDO.

BLOB stands for a binary large object that is a collection of binary data stored as a value in the database. By using the BLOB, you can store the documents, images, and other multimedia files in the database.

We will create a new table (https://www.sqlitetutorial.net/sqlite-create-table/) named `documents` for the sake of demonstration.

```
CREATE TABLE IF NOT EXISTS documents (
    document_id INTEGER PRIMARY KEY,
    mime_type   TEXT    NOT NULL,
    doc         BLOB
);
```

Writing BLOB into the table

To insert BLOB data into the table, you use the following steps:

1. Connect to the SQLite database (https://www.sqlitetutorial.net/sqlite-php/connect/) by creating an instance of the PDO class.

2. Use `fopen()` function to read the file. The `fopen()` function returns a file pointer.
3. Prepare the INSERT statement (https://www.sqlitetutorial.net/sqlite-php/insert/) for execution by calling the `prepare()` method of the PDO object. The `prepare()` method returns an instance of the `PDOStatement` class.
4. Use the `bindParam()` method of the `PDOStatement` object to bind a parameter to a variable name. For the BLOB data, you bind a parameter to the file pointer.
5. Call the `execute()` method of the PDO statement object.

For example, the following `insertDoc()` method of the `SQLiteBLOB` class inserts a new document into the `documents` table using the above steps:

```
<?php

namespace App;

/**
 * SQLite PHP Blob Demo
 */
class SQLiteBLOB {

    /**
     * PDO object
     * @var \PDO
     */
    private $pdo;

    /**
     * Initialize the object with a specified PDO object
     * @param \PDO $pdo
     */
    public function __construct($pdo) {
        $this->pdo = $pdo;
    }

    /**
```

```
     * Insert blob data into the documents table
     * @param type $pathToFile
     * @return type
     */
    public function insertDoc($mimeType, $pathToFile) {
        if (!file_exists($pathToFile))
            throw new \Exception("File %s not found.");

        $sql = "INSERT INTO documents(mime_type,doc) "
                . "VALUES(:mime_type,:doc)";

        // read data from the file
        $fh = fopen($pathToFile, 'rb');

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

        $stmt->bindParam(':mime_type', $mimeType);
        $stmt->bindParam(':doc', $fh, \PDO::PARAM_LOB);
        $stmt->execute();

        fclose($fh);

        return $this->pdo->lastInsertId();
    }
}
```

The following index.php script inserts two documents: 1 PDF file and 1 picture from the `assets` folder into the `documents` table.

```
<?php

require 'vendor/autoload.php';

use App\SQLiteConnection as SQLiteConnection;
use App\SQLiteBLOB as SQLiteBlob;

$sqlite = new SQLiteBlob((new SQLiteConnection)->connect());
```

```
// insert a PDF file into the documents table
$pathToPDFFile = 'assets/sqlite-sample database-diagram.pdf';
$pdfId = $sqlite->insertDoc('application/pdf', $pathToPDFFile);

// insert a PNG file into the documents table
$pathToPNGFile = 'assets/sqlite-tutorial-logo.png';
$pngId = $sqlite->insertDoc('image/png', $pathToPNGFile);
```

We execute this index.php script file and use the following SELECT (https://www.sqlitetutorial.net/sqlite-java/select/) statement to verify the insert:

```
SELECT id,
       mime_type,
       doc
  FROM documents;
```

document_id	mime_type	doc
1	application/pdf	%PDF-1.4%□□□□4 0 obj<<</Linearized 1/L
2	image/png	□PNG□

Reading BLOB from the table

To read the BLOB from the database, we add a new method named `readDoc()` to the `SQLiteBLOB` class as follows:

```
/**
     * Read document from the documents table
     * @param type $documentId
     * @return type
     */
    public function readDoc($documentId) {
        $sql = "SELECT mime_type, doc "
                . "FROM documents "
                . "WHERE document_id = :document_id";
```

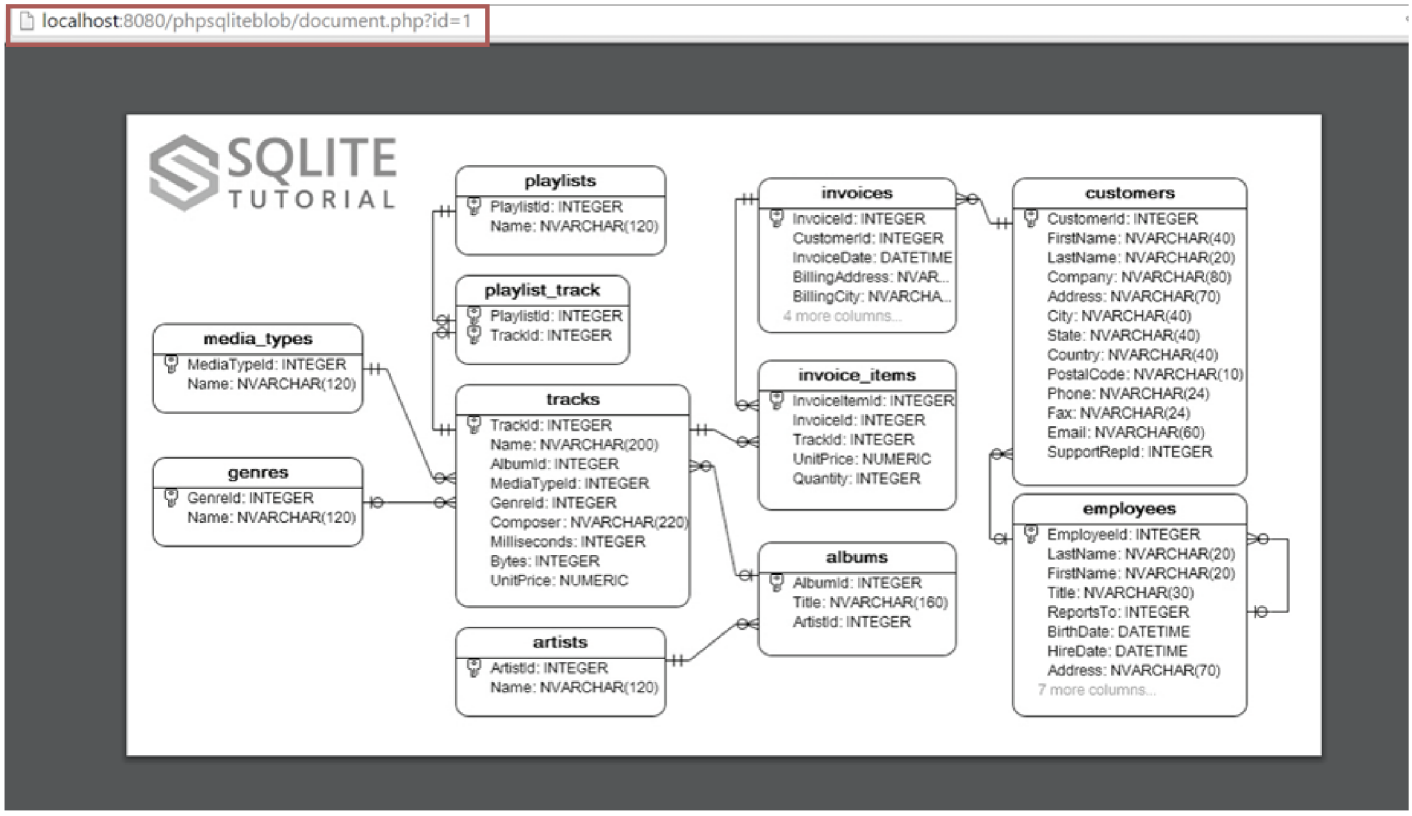
```
        // initialize the params
        $mimeType = null;
        $doc = null;
        //
        $stmt = $this->pdo->prepare($sql);
        if ($stmt->execute([":document_id" => $documentId])) {

            $stmt->bindColumn(1, $mimeType);
            $stmt->bindColumn(2, $doc, \PDO::PARAM_LOB);

            return $stmt->fetch(\PDO::FETCH_BOUND) ?
                    ["document_id" => $documentId,
                     "mime_type" => $mimeType,
                     "doc" => $doc] : null;
        } else {
            return null;
        }
    }
```

The following `document.php` script gets the `document_id` from the query string and calls the `readDoc()` method to render the document.

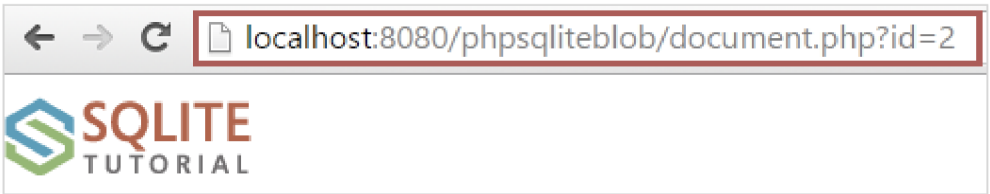
```
<?php

require 'vendor/autoload.php';

use App\SQLiteConnection as SQLiteConnection;
use App\SQLiteBLOB as SQLiteBlob;

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

// get document id from the query string
$documentId = filter_input(INPUT_GET, 'id', FILTER_SANITIZE_NUMBER_INT);

// read documet from the database
$doc = $sqlite->readDoc($documentId);
```

```
if ($doc != null) {
    header("Content-Type:" . $doc['mime_type']);
    echo $doc['doc'];
} else {
    echo 'Error loading document ' . $documentId;
}
```

For example, the following screenshot shows how the document.php script returns the PDF file in the web browser:



To test the document id 2, you change the value in the query string as shown in the screenshot below:



Update BLOB data

The following `updateDoc()` method updates the BLOB data in the `documents` table.

```
    /**
     * Update document
     * @param type $documentId
     * @param type $mimeType
     * @param type $pathToFile
     * @return type
     * @throws \Exception
     */
     public function updateDoc($documentId, $mimeType, $pathToFile) {

         if (!file_exists($pathToFile))
             throw new \Exception("File %s not found.");

         $fh = fopen($pathToFile, 'rb');

         $sql = "UPDATE documents
                 SET mime_type = :mime_type,
                     doc = :doc
                 WHERE document_id = :document_id";

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

         $stmt->bindParam(':mime_type', $mimeType);
         $stmt->bindParam(':data', $fh, \PDO::PARAM_LOB);
         $stmt->bindParam(':document_id', $documentId);

         fclose($fh);

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

In this tutorial, we have shown you how to write, read, and update BLOB data in SQLite database using PHP PDO.