



CSCI-3403: Cyber Security Spring 2020

Abigail Fernandes

Department of Computer Science

University of Colorado Boulder

Week 6

- > MySQL

- > SQL Injection

- > Code Walkthrough

What is MySQL

- **Relational SQL** Database Management System
- Data in RDBMS is stored in **tables**
- **Create, Read ,Update, Delete (CRUD)** Operations

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

The diagram illustrates a database table with five columns: ID, NAME, AGE, ADDRESS, and SALARY. The first row is highlighted with a red border, representing a single record. A blue box highlights the first column (ID), representing a single field. A blue arrow points from the text 'Fields' to the first column. Another blue arrow points from the text 'Record/Row' to the first row. A third blue arrow points from the text 'Column' to the first column. The table is enclosed in a dashed green border.

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

Fields

Record/Row

Column

Installation on Linux

```
sudo apt-get install mysql-server
```

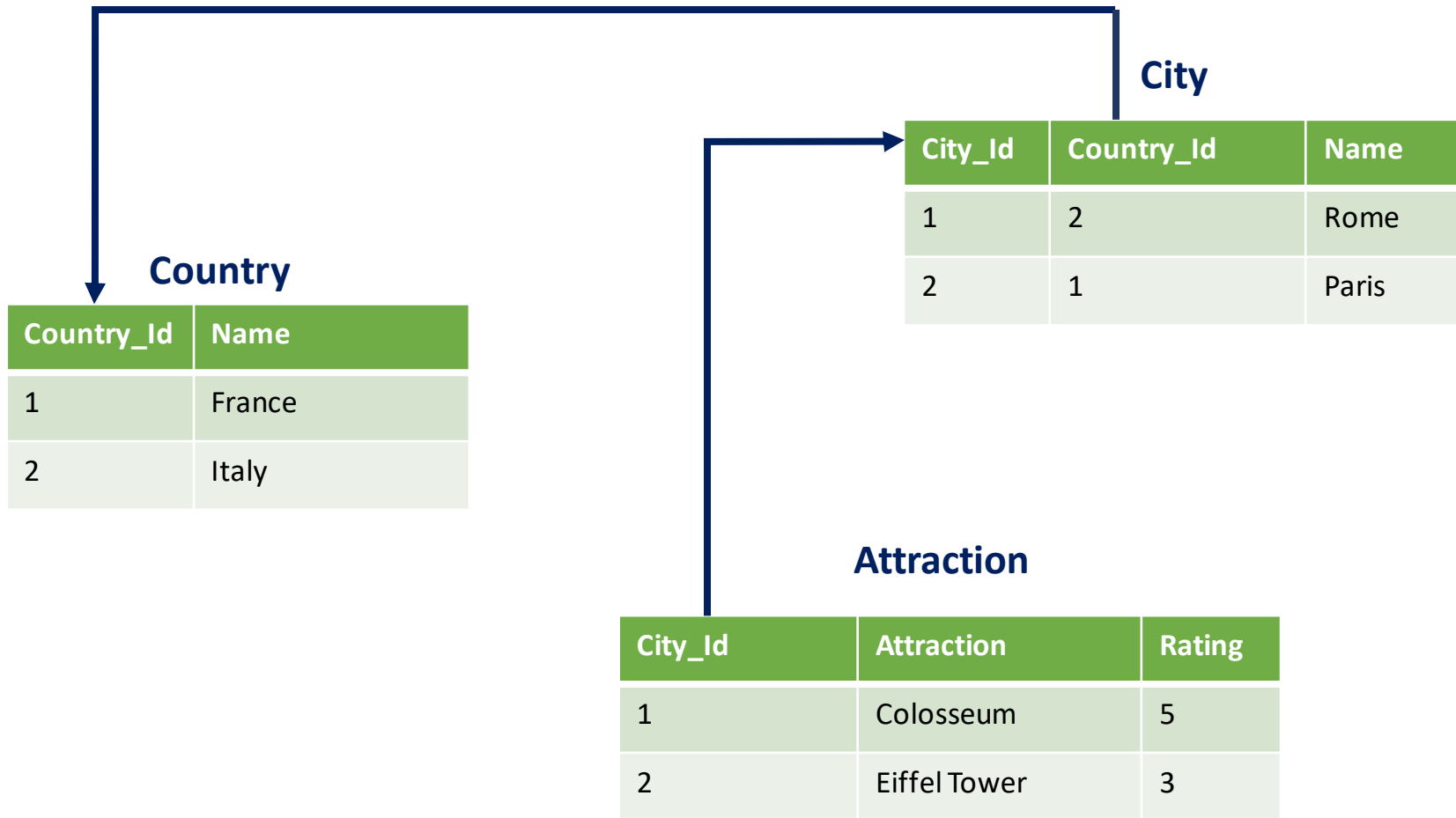
- This works for me on Linux. If you run into installation trouble, you can refer to [this guide](#)
- I skipped adding the MySQL APT Repository since it was already there.
- You can verify that everything is working by running -

```
mysqladmin --version
```



MySQL Queries

Database Schema



Create Table

country_id	country_name
1	Italy
2	Spain
3	France
4	Netherlands
5	Germany

```
CREATE TABLE `3314093_cyber`.`country` (
  country_id INT(6) NOT NULL AUTO_INCREMENT,
  country_name VARCHAR(50) NOT NULL ,
  PRIMARY KEY (`country_id`)|
```

```
CREATE TABLE `3314093_cyber`.`city` (
  `city_id` INT(6) NOT NULL,
  `city_name` VARCHAR(100) NOT NULL,
  `country_id` INT(6) NOT NULL,
  FOREIGN KEY(`country_id`) REFERENCES country(`country_id`),
  PRIMARY KEY(`city_id`)
);|
```

city_id	attraction_name	rating
2	Rialto Bridge	5
2	St. Mark's Square	3
1	Blue Grotto	5
4	Colloseum	4
4	Trevi Fountain	5
6	Sagrada Familia	5
6	Parc Guell	3
6	Camp Nou	5

city_id	country_id	city_name
1	1	Capri
2	1	Venice
3	1	Florence
4	1	Rome
5	1	Amalfi
6	2	Barcelona
7	2	Madrid

```
CREATE TABLE `3314093_cyber`.`attraction` (
  `city_id` INT(6) NOT NULL,
  `attraction_name` VARCHAR(100) NOT NULL,
  `rating` INT NOT NULL,
  FOREIGN KEY(`city_id`) REFERENCES city(`city_id`),
  PRIMARY KEY(`attraction_name`)
);
```


MySQL Queries

Create

```
INSERT INTO city(city_name, country_id) VALUES('Capri', 1)|
```

Read

```
SELECT attraction_name, rating FROM attraction|
```

Update

```
UPDATE attraction|  
SET attraction_name = 'Colosseum',  
WHERE attraction_name = 'Colloseum';
```

DELETE

```
DELETE FROM city  
WHERE city_name = 'Rome'
```

Find all the tourists attractions in a city (JOIN)

```
SELECT city_name, attraction_name  
FROM city INNER JOIN attraction  
WHERE city.city_id = attraction.city_id
```

city_name	attraction_name
Venice	Rialto Bridge
Venice	St. Mark's Square
Capri	Blue Grotto
Rome	Colloiseum
Rome	Trevi Fountain
Barcelona	Sagrada Familia
Barcelona	Parc Guell
Barcelona	Camp Nou

Find how many cities you would visit for every country

```
SELECT country_name, COUNT(*)  
FROM country INNER JOIN city  
WHERE country.country_id = city.country_id  
GROUP BY country.country_name
```

country_name	COUNT(city.city_id)
Italy	5
Spain	2

Week 6

- > MySQL

- > **SQL Injection**

- > Code Walkthrough



SQL Injection

Suppose we issue this query

```
SELECT author, title FROM books  
WHERE publisher='Wiley';
```

What if we issue this query instead

```
SELECT author, title FROM books  
WHERE publisher='O'Wiley';
```

Would this error out?

What if you as an attacker got to control the value of publisher?

SQL Injection

We can insert *Wiley' OR '1' = '1*

The resulting query would look like

```
SELECT author, title FROM books  
WHERE publisher='Wiley' OR '1' = '1';
```

We succeed in the attack and we see all the rows of the table, which was not what the developer intended.

SQL Injection

Single line comments in MySQL start with --

Perform an SQL injection in the previous example by using --

```
SELECT author, title FROM books  
WHERE publisher='Wiley' OR '1' = '1' --';
```

This is particularly useful if you had a complex query like so

```
SELECT author, title FROM books  
WHERE publisher='Wiley' OR '1' = '1' --';
```

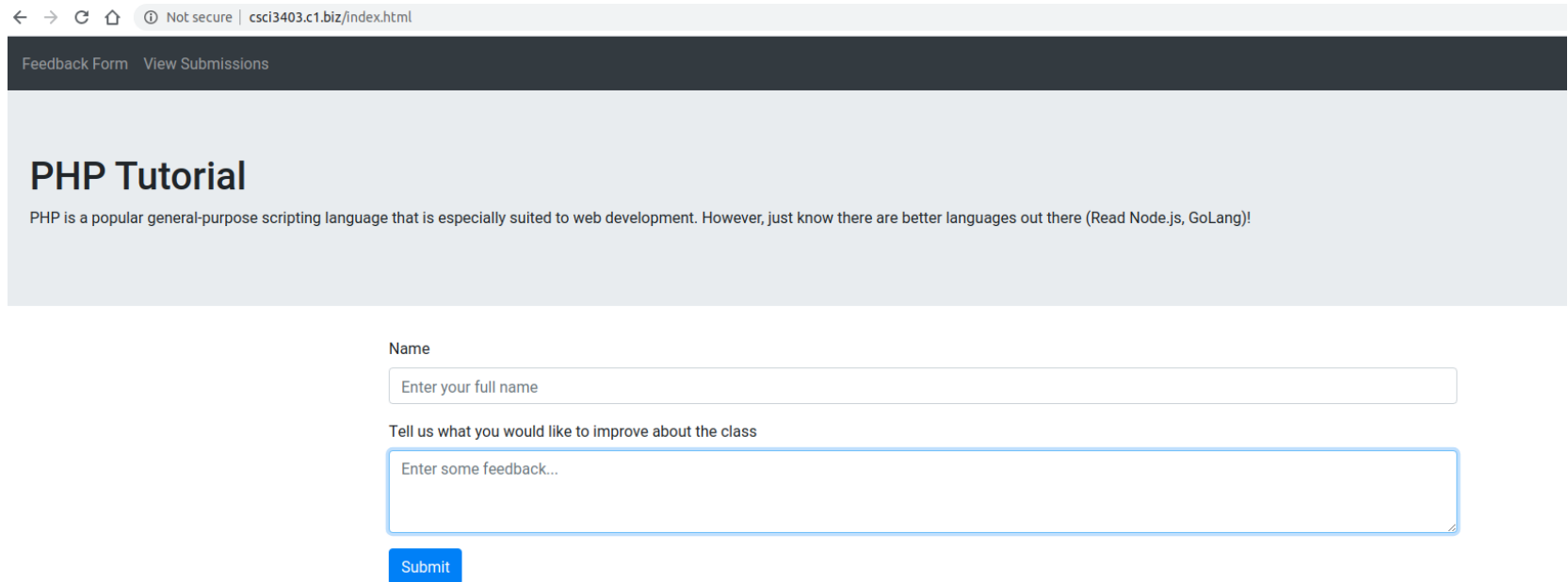

mysqli query vs multi_query

```
$mysqli->multi_query("Select * FROM Users; DROP TABLE  
Users"); // OK
```

```
$mysqli->query("SQL query 1;"); // Not executed
```

```
$mysqli->query(" SQL query 1; SQL query 2"); // Not  
executed
```

Remember the website we built?



The screenshot shows a web browser window with the address bar displaying "Not secure | csci3403.c1.biz/index.html". The page has a dark header with links for "Feedback Form" and "View Submissions". The main content area is light gray and features the title "PHP Tutorial" in bold. Below the title is a paragraph of text: "PHP is a popular general-purpose scripting language that is especially suited to web development. However, just know there are better languages out there (Read Node.js, GoLang)!". Further down, there is a feedback form. It starts with a "Name" label and a text input field containing the placeholder "Enter your full name". Below this is a label "Tell us what you would like to improve about the class" and a larger text area with the placeholder "Enter some feedback...". At the bottom of the form is a blue "Submit" button.

Let's attack it!

Let me show you the query

Here's the query that runs when you enter your name and comment on the site

```
$query = "INSERT INTO comments  
(user, comment) VALUES('".$_POST['name']."', '".$_POST['comment']."')";  
$conn -> multi_query($query)
```

And here is what a harmless normal username and comment would look like

```
"INSERT INTO comments (user, comment) VALUES('Bob', 'Hello');"
```



SQL injection on demo forum

Name

Attacker

Tell us what you would like to improve about the class

Harmless Comment'); DELETE FROM comments WHERE user='Biljith'; --|

Submit

```
"INSERT INTO comments (user, comment)
VALUES('Attacker', 'Harmless Comment');
DELETE FROM comments WHERE user='Bob'; --
' ) "
```



Need more Practice?

- Download DVWA – Damn Vulnerable Web Application
- Install it locally.
- It has a lot of different exploits you can practice. SQL injection is one of them
- Download [here](#)
- Or follow the instructions [here](#) to set it up

Week 6

- > MySQL

- > SQL Injection

- > **Code Walkthrough**

Create Database (Biz.nf)

Create MySQL Database

Database Name	<input type="text" value="3314093_travel"/>	
Database Password	<input type="password" value="....."/>	Generate
	Medium Password.	
Confirm Database Password	<input type="password" value="....."/>	
Database Version	<input type="text" value="5.7"/>	

[Create Database](#)

Database Information

Hosting Tools -> MySQL Databases

Name	User	Host	Port	Quota	Management	Type
3314093_cyber	3314093_cyber	fdb18.biz.nf	3306	Available: 30 MiB Used: 20 KiB	phpMyAdmin 4 See all tools	MySQL

[Information](#) [Password](#) [Management](#) [Delete](#)

Database details for 3314093_cyber

Below you can find detailed information for your database.

Database Host:	fdb18.biz.nf
Database Port:	3306
Database Name:	3314093_cyber
Database User:	3314093_cyber
Database Password:	(The password for 3314093_cyber) change
Database Version:	5.7

Connect to the DB

Database information

```
<?php
```

```
$servername = "fdb18.biz.nf";  
$username = "3314093 cyber";  
$password = [REDACTED];  
$dbname = "3314093_cyber";
```

```
// Create connection  
$conn = new mysqli($servername, $username, $password, $dbname);  
// Check connection  
if ($conn->connect_error) {  
    die("Connection failed: " . $conn->connect_error);  
}
```

Connect to the DB server

PHP works with MySQL DB using

- MySQLi
- PHP Data Objects (PDO)



Reading from DB

```
// Select data with SQLi
$sql = "SELECT * from comments";
$result = mysqli_query($conn, $sql);

if (mysqli_num_rows($result) > 0) {
    // output data of each row
    while($row = mysqli_fetch_assoc($result)) {
        echo "<li class=\"media\">
            <div class=\"media-body\">
                <strong class=\"text-success\">@". $row["user"] . "</strong>
                <p>". $row["comment"] . "
            </p>
            </div>
        </li>";
    }
} else {
    echo "<p>Nothing to see yet!</p>";
}

mysqli_close($conn);

?>
```

@Bob

I like talking to Alice!

@Alice

Eve always keeps interfering!

Recall from SQL Injection!

Here's the query that runs when you enter your name and comment on the site

```
$query = "INSERT INTO comments  
(user, comment) VALUES('".$_POST['name']."', '".$_POST['comment']."')";  
$conn -> multi_query($query)
```

And here is what a harmless normal username and comment would look like

```
"INSERT INTO comments (user, comment) VALUES('Bob', 'Hello');"
```



Prepared Statements

- A *prepared statement* or a *parameterized statement* is used to execute the same statement repeatedly with high efficiency.
- How MySQLi prepared statements work in PHP

1. Prepare an SQL query with empty values as placeholders (with a question mark for each value).
2. Bind variables to the placeholders by stating each variable, along with its type.
3. Execute query.

Inserting into DB

```
// Insert into DB
```

```
$query = "INSERT INTO comments (user, comment)
VALUES(?, ?)";
```

```
$stmt = $conn->prepare($query);
```

```
$stmt->bind_param("ss", $_POST["name"], $_POST["comment"]);
```

```
if ($stmt->execute() === TRUE) {
```

```
    echo "<h3>welcome " . $_POST["name"] . "</h3><br>";
```

```
    echo "<p> Thank you for giving us your feedback! Your feedback has been recorded as
    <em>" . $_POST["comment"] . "</em></p>";
```

```
} else {
```

```
    echo "<p>Error: " . $sql . "<br>" . $conn->error."</p>";
```

```
}
```

```
$conn->close();
```

Prepare the template

Run the code

Attach variables to the dummy values in the prepared template