MySQL III

Objectives

- SQL join types
- PHP connect to MySQL
- CRUD operations using PHP/MySQL

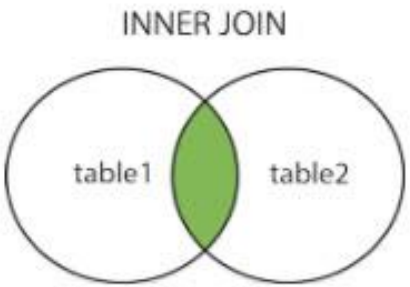
SQL JOIN

- A JOIN clause is used to combine rows from two or more tables, based on a related column between them.
- Different Types of SQL JOINs
  - **(INNER) JOIN**: Returns records that have matching values in both tables
  - **LEFT (OUTER) JOIN**: Return all records from the left table, and the matched records from the right table
  - **RIGHT (OUTER) JOIN**: Return all records from the right table, and the matched records from the left table
  - **FULL (OUTER) JOIN**: Return all records when there is a match in either left or right table

SQL INNER JOIN

- The INNER JOIN keyword selects records that have matching values in both tables.
- Syntax

```
SELECT column_name(s)
FROM table1
INNER JOIN table2 ON table1.column_name = table2.column_name;
```



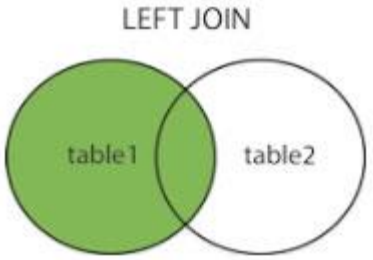
- Example

```
SELECT Orders.OrderID, Customers.CustomerName
FROM Orders
INNER JOIN Customers ON Orders.CustomerID = Customers.CustomerID;
```

SQL LEFT JOIN

- The LEFT JOIN keyword returns all records from the left table (table1), and the matched records from the right table (table2). The result is NULL from the right side, if there is no match.
- Syntax

```
SELECT column_name(s)
FROM table1
LEFT JOIN table2 ON table1.column_name = table2.column_name;
```



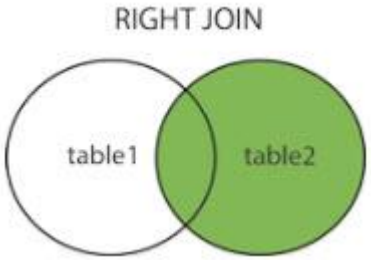
- Example

```
SELECT Customers.CustomerName, Orders.OrderID
FROM Customers
LEFT JOIN Orders ON Customers.CustomerID = Orders.CustomerID
ORDER BY Customers.CustomerName;
```

SQL RIGHT JOIN

- The RIGHT JOIN keyword returns all records from the right table (table2), and the matched records from the left table (table1). The result is NULL from the left side, when there is no match.
- Syntax

```
SELECT column_name(s)
FROM table1
RIGHT JOIN table2 ON table1.column_name = table2.column_name;
```



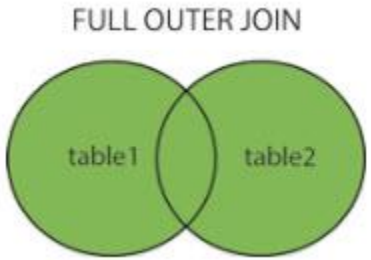
- Example

```
SELECT Orders.OrderID, Employees.LastName, Employees.FirstName
FROM Orders
RIGHT JOIN Employees ON Orders.EmployeeID = Employees.EmployeeID
ORDER BY Orders.OrderID;
```

FULL OUTER JOIN

- The FULL OUTER JOIN keyword return all records when there is a match in either left (table1) or right (table2) table records.
  - **Note:** FULL OUTER JOIN can potentially return very large result-sets!
- Syntax

```
SELECT column_name(s)
FROM table1
FULL OUTER JOIN table2 ON table1.column_name = table2.column_name;
```



- Example

```
SELECT Customers.CustomerName, Orders.OrderID
FROM Customers
FULL OUTER JOIN Orders ON Customers.CustomerID=Orders.CustomerID
ORDER BY Customers.CustomerName;
```

Demo

- Teacher demo join types for students

PHP Connect to MySQL

- PHP 5 and later can work with a MySQL database using:
  - MySQLi extension (the "i" stands for improved)
  - PDO (PHP Data Objects)
- Note: Earlier versions of PHP used the MySQL extension. However, this extension was deprecated in 2012.
- Both MySQLi and PDO have their advantages:
  - PDO will work on 12 different database systems, whereas MySQLi will only work with MySQL databases.
  - So, if you have to switch your project to use another database, PDO makes the process easy. You only have to change the connection string and a few queries. With MySQLi, you will need to rewrite the entire code - queries included.
  - Both are object-oriented, but MySQLi also offers a procedural API.
  - Both support Prepared Statements. Prepared Statements protect from SQL injection, and are very important for web application security.

Open a Connection

- Before we can access data in the MySQL database, we need to be able to connect to the server.
- Example

```php
<?php
$servername = "localhost";
$username = "username";
$password = "password";

// Create connection
$conn = mysqli_connect($servername, $username, $password);

// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}
echo "Connected successfully";
?>
```

Close the Connection

- The connection will be closed automatically when the script ends. To close the connection before, use the following:
  - mysqli_close($conn);

PHP Create Database

- A database consists of one or more tables.
- The CREATE DATABASE statement is used to create a database in MySQL.
- Example

```php
<?php
$servername = "localhost";
$username = "username";
$password = "password";

// Create connection
$conn = mysqli_connect($servername, $username, $password);
// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}

// Create database
$sql = "CREATE DATABASE myDB";
if (mysqli_query($conn, $sql)) {
    echo "Database created successfully";
} else {
    echo "Error creating database: " . mysqli_error($conn);
}

mysqli_close($conn);
?>
```

Create MySQL Tables

- A database table has its own unique name and consists of columns and rows.
- The CREATE TABLE statement is used to create a table in MySQL.
  - The data type specifies what type of data the column can hold.
- After the data type, you can specify other optional attributes for each column:
  - NOT NULL - Each row must contain a value for that column, null values are not allowed
  - DEFAULT value - Set a default value that is added when no other value is passed
  - UNSIGNED - Used for number types, limits the stored data to positive numbers and zero
  - AUTO INCREMENT - MySQL automatically increases the value of the field by 1 each time a new record is added
  - PRIMARY KEY - Used to uniquely identify the rows in a table. The column with PRIMARY KEY setting is often an ID number, and is often used with AUTO_INCREMENT
- Each table should have a primary key column (in this case: the "id" column). Its value must be unique for each record in the table.

Example

- Example creating table:

```
CREATE TABLE MyGuests (
id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
firstname VARCHAR(30) NOT NULL,
lastname VARCHAR(30) NOT NULL,
email VARCHAR(50),
reg_date TIMESTAMP
)
```

Example

```php
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}

// sql to create table
$sql = "CREATE TABLE MyGuests (
id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
firstname VARCHAR(30) NOT NULL,
lastname VARCHAR(30) NOT NULL,
email VARCHAR(50),
reg_date TIMESTAMP
)";

if (mysqli_query($conn, $sql)) {
    echo "Table MyGuests created successfully";
} else {
    echo "Error creating table: " . mysqli_error($conn);
}

mysqli_close($conn);
?>
```

Insert Data Into MySQL

- After a database and a table have been created, we can start adding data in them.
- Here are some syntax rules to follow:
  - The SQL query must be quoted in PHP
  - String values inside the SQL query must be quoted
  - Numeric values must not be quoted
  - The word NULL must not be quoted
- The INSERT INTO statement is used to add new records to a MySQL table:

```
INSERT INTO table_name (column1, column2, column3,...)
VALUES (value1, value2, value3,...)
```

- **Note:** If a column is AUTO_INCREMENT (like the "id" column) or TIMESTAMP (like the "reg_date" column), it is no need to be specified in the SQL query; MySQL will automatically add the value.

Example

```php
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}

$sql = "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('John', 'Doe', 'john@example.com')";

if (mysqli_query($conn, $sql)) {
    echo "New record created successfully";
} else {
    echo "Error: " . $sql . "<br>" . mysqli_error($conn);
}

mysqli_close($conn);
?>
```

Get ID of Last Inserted Record

- If we perform an INSERT or UPDATE on a table with an AUTO_INCREMENT field, we can get the ID of the last inserted/updated record immediately.
- In the table "MyGuests", the "id" column is an AUTO_INCREMENT field:

```
CREATE TABLE MyGuests (
id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
firstname VARCHAR(30) NOT NULL,
lastname VARCHAR(30) NOT NULL,
email VARCHAR(50),
reg_date TIMESTAMP
)
```

Example

```php
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}

$sql = "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('John', 'Doe', 'john@example.com')";

if (mysqli_query($conn, $sql)) {
    $last_id = mysqli_insert_id($conn);
    echo "New record created successfully. Last inserted ID is: " . $last_id;
} else {
    echo "Error: " . $sql . "<br>" . mysqli_error($conn);
}

mysqli_close($conn);
?>
```

Insert Multiple Records

- Multiple SQL statements must be executed with the mysqli_multi_query() function.
- Note: that each SQL statement must be separated by a semicolon.
- Example:

```
$sql = "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('John', 'Doe', 'john@example.com');";
$sql .= "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('Mary', 'Moe', 'mary@example.com');";
$sql .= "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('Julie', 'Dooley', 'julie@example.com')";

if (mysqli_multi_query($conn, $sql)) {
    echo "New records created successfully";
} else {
    echo "Error: " . $sql . "<br>" . mysqli_error($conn);
}
```

Select Data

- The SELECT statement is used to select data from one or more tables

```
SELECT column_name(s) FROM table_name
```

- or we can use the * character to select ALL columns from a table

```
SELECT * FROM table_name
```

Example

```php
// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}

$sql = "SELECT id, firstname, lastname FROM MyGuests";
$result = mysqli_query($conn, $sql);

if (mysqli_num_rows($result) > 0) {
    // output data of each row
    while($row = mysqli_fetch_assoc($result)) {
        echo "id: " . $row["id"]. " - Name: " . $row["firstname"]. " " . $row["lastname"]. "
<br>";
    }
} else {
    echo "0 results";
}
```

Delete Data

- The DELETE statement is used to delete records from a table

```
DELETE FROM table_name
WHERE some_column = some_value
```

- Example

```php
// sql to delete a record
$sql = "DELETE FROM MyGuests WHERE id=3";

if (mysqli_query($conn, $sql)) {
    echo "Record deleted successfully";
} else {
    echo "Error deleting record: " . mysqli_error($conn);
}

mysqli_close($conn);
```

Update Data

- The UPDATE statement is used to update existing records in a table
  - Note: The WHERE clause specifies which record or records that should be updated. If you omit the WHERE clause, all records will be updated !
- Example

```
UPDATE table_name
SET column1=value, column2=value2,...
WHERE some_column=some_value
```

Example

```php
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}

$sql = "UPDATE MyGuests SET lastname='Doe' WHERE id=2";

if (mysqli_query($conn, $sql)) {
    echo "Record updated successfully";
} else {
    echo "Error updating record: " . mysqli_error($conn);
}

mysqli_close($conn);
?>
```

Demo

- Teacher demo code about CRUD operation with MySQL

Summary

- SQL join types
- PHP connect to MySQL
- CRUD operations using PHP/MySQL