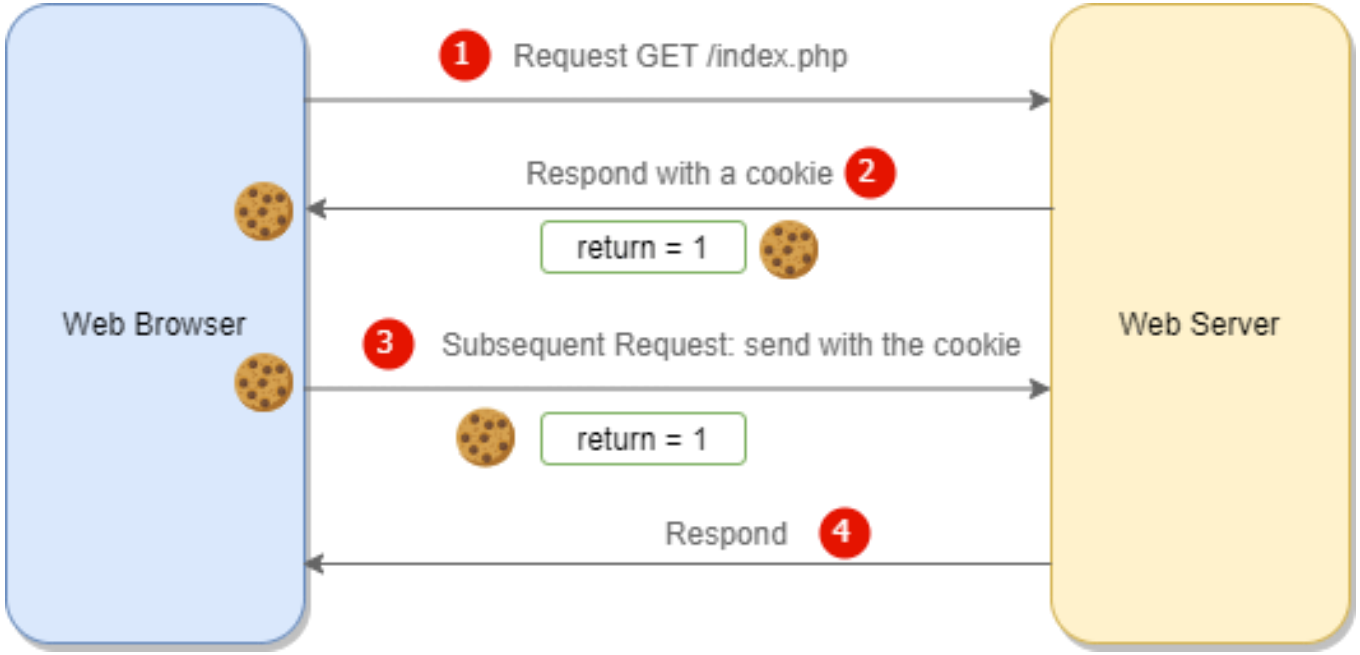

# Backend Development

**PHP Basics**

# Create/Retrieve a Cookie




https://www.w3schools.com/php/php_cookies.asp

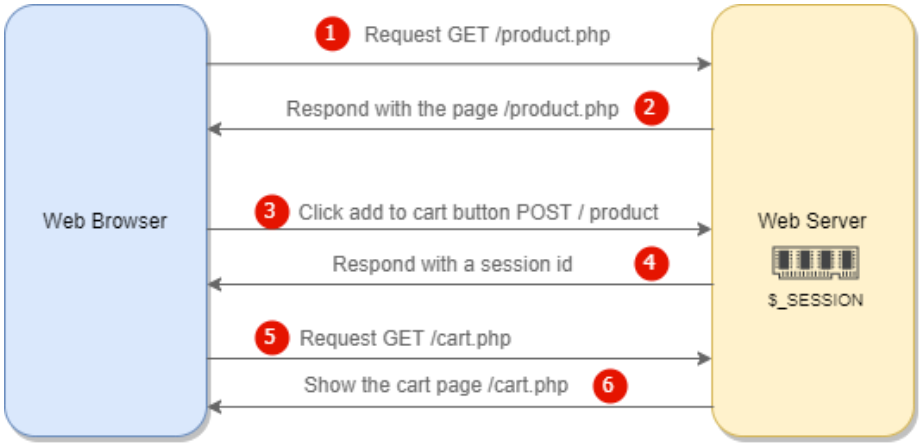
# Create/Retrieve a Cookie

```php
<?php
$cookie_name = "user";
$cookie_value = "John Doe";
setcookie($cookie_name, $cookie_value, time() +
(86400 * 30), "/"); // 86400 = 1 day
?>
<html>
<body>

<?php
if(!isset($_COOKIE[$cookie_name])) {
  echo "Cookie named '" . $cookie_name . "' is not set!";
} else {
  echo "Cookie '" . $cookie_name . "' is set!<br>";
  echo "Value is: " . $_COOKIE[$cookie_name];
}
?>

</body>
</html>
```

https://www.w3schools.com/php/php_cookies.asp

# Sessions

```php
<?php
// Start the session
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
// Set session variables
$_SESSION["favcolor"] = "green";
$_SESSION["favanimal"] = "cat";
echo "Session variables are set.";
?>

</body>
</html>
```




https://www.w3schools.com/php/php_sessions.asp

# What is SQL?

SQL is the standard language for dealing with Relational Databases.

SQL is used to insert, search, update, and delete database records.

# What is RDBMS?

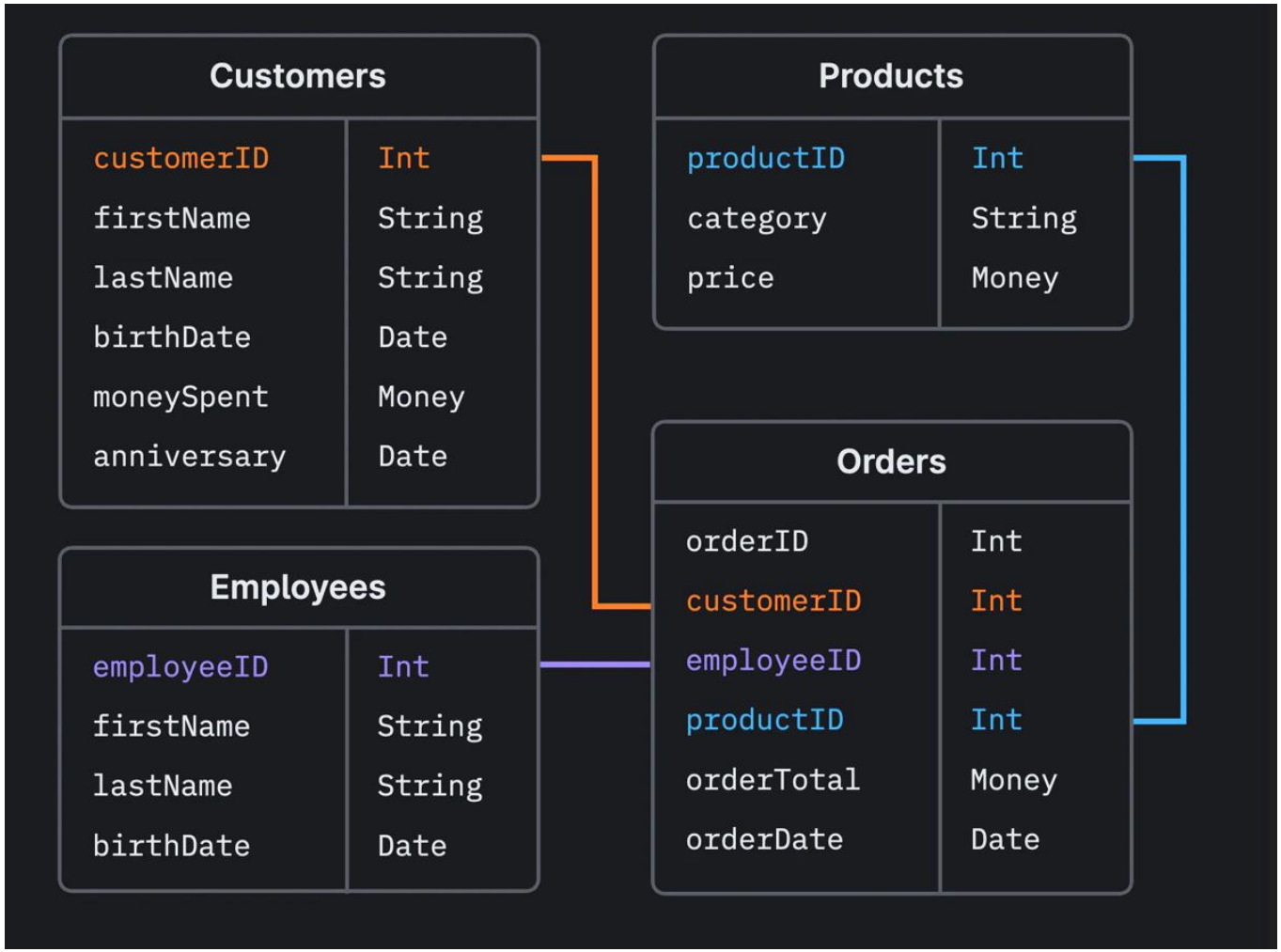
RDBMS stands for Relational Database Management System.

RDBMS is a program used to maintain a relational database.

RDBMS is the basis for all modern database systems such as MySQL, Microsoft SQL Server, Oracle, and Microsoft Access.

RDBMS uses SQL queries to access the data in the database.

# What is RDBMS?



# What is mysql?

- MySQL is a relational database management system
- MySQL is open-source
- MySQL is free
- MySQL is ideal for both small and large applications
- MySQL is very fast, reliable, scalable, and easy to use
- MySQL is cross-platform
- MySQL is compliant with the ANSI SQL standard
- MySQL was first released in 1995
- MySQL is developed, distributed, and supported by Oracle Corporation
- MySQL is named after co-founder Monty Widenius's daughter: My

# phpMyAdmin

phpMyAdmin is a free software tool written in PHP, intended to handle the administration of MySQL over the Web. phpMyAdmin supports a wide range of operations on MySQL and MariaDB. Frequently used operations (managing databases, tables, columns, relations, indexes, users, permissions, etc) can be performed via the user interface, while you still have the ability to directly execute any SQL statement.

# CREATE DATABASE - sql

```
CREATE DATABASE databasename;
```

# DROP DATABASE - sql

```
DROP DATABASE databasename;
```

# CREATE TABLE - sql

```
CREATE TABLE table_name (
    column1 datatype,
    column2 datatype,
    column3 datatype,
   ....
);
```

# Data Types - mysql

https://www.w3schools.com/MySQL/mysql_datatypes.asp

# Constraints

- NOT NULL - Ensures that a column cannot have a NULL value
- UNIQUE - Ensures that all values in a column are different
- PRIMARY KEY - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table
- FOREIGN KEY - Prevents actions that would destroy links between tables
- CHECK - Ensures that the values in a column satisfies a specific condition
- DEFAULT - Sets a default value for a column if no value is specified
- CREATE INDEX - Used to create and retrieve data from the database very quickly

# Create Constraints - sql

```
CREATE TABLE table_name (
    column1 datatype constraint,
    column2 datatype constraint,
    column3 datatype constraint,
    ....
);
```

# AUTO INCREMENT Field

- Auto-increment allows a unique number to be generated automatically when a new record is inserted into a table.
- Often this is the primary key field that we would like to be created automatically every time a new record is inserted.

# Example - sql

```
CREATE TABLE Persons (
    Personid int NOT NULL AUTO_INCREMENT,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    PRIMARY KEY (Personid)
);
```

# dates

- DATE - format YYYY-MM-DD
- DATETIME - format: YYYY-MM-DD HH:MI:SS
- TIMESTAMP - format: YYYY-MM-DD HH:MI:SS
- YEAR - format YYYY or YY

# DROP TABLE - TRUNCATE TABLE - sql

```
DROP TABLE table_name;

TRUNCATE TABLE table_name;
```

# ALTER TABLE - sql

```
// Add Column

ALTER TABLE table_name
ADD column_name datatype;

// Delete Column

ALTER TABLE table_name
DROP COLUMN column_name;
```

# ALTER TABLE - sql

```
// Modify Column

ALTER TABLE table_name
MODIFY COLUMN column_name datatype;
```

# SELECT Statement - sql

```
SELECT column1, column2, ...
FROM table_name;
```

# WHERE Clause - sql

```
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

# ORDER BY Keyword - sql

```
SELECT column1, column2, ...
FROM table_name
ORDER BY column1, column2, ... ASC|DESC;
```

# Views

In SQL, a view is a virtual table based on the result-set of an SQL statement.

A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.

You can add SQL statements and functions to a view and present the data as if the data were coming from one single table.

A view is created with the CREATE VIEW statement.

# Syntax - sql

```
CREATE VIEW view_name AS
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

# Connect to MySQL

- **MySQLi extension** (the "i" stands for improved)
- **PDO (PHP Data Objects)**

# Should I Use MySQLi or PDO?

If you need a short answer, it would be "Whatever you like".

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.

Both support Prepared Statements. Prepared Statements protect from SQL injection, and are very important for web application security.

# Connect to db using pdo

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

try {
  $conn = new PDO("mysql:host=$servername;dbname=myDB",
$username, $password);
  // set the PDO error mode to exception
  $conn->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);
  echo "Connected successfully";
} catch(PDOException $e) {
  echo "Connection failed: " . $e->getMessage();
}
?>
```

# Select data

```php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDBPDO";

try {
  $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username,
$password);
  $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
  $stmt = $conn->prepare("SELECT id, firstname, lastname FROM
MyGuests");
  $stmt->execute();

  // set the resulting array to associative
  //$result = $stmt->setFetchMode(PDO::FETCH_ASSOC);
  foreach($stmt->fetchAll() as $k) {
    echo $k[“firstname”];
  }
} catch(PDOException $e) {
  echo "Error: " . $e->getMessage();
}
$conn = null;
```

# Another example

```
$stmt = $pdo->query("SELECT * FROM users");

while ($row = $stmt->fetch()) {

    echo $row['name']."<br />\n";

}
```

# 3rd example

```php
try {
  $conn = new PDO("mysql:host=$servername;dbname=temp", $username, $password);
  // set the PDO error mode to exception
  $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
  $data = $conn->query("select * from users")->fetchAll();
  foreach ($data as $row) {
    echo $row['firstname'].'<br>';
  }
} catch(PDOException $e) {
  echo "Connection failed: " . $e->getMessage();
}
```

# Query with parameter

```php
try {
  $conn = new PDO("mysql:host=$servername;dbname=temp", $username, $password);
  // set the PDO error mode to exception
  $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
  $data = $conn->prepare("select * from users where user_id=?");
  $id = 1;
  $data->execute([$id]);
  $user = $data->fetch();
  echo $user['firstname'];

} catch(PDOException $e) {
  echo "Connection failed: " . $e->getMessage();
}
```

# Pdo Insert data

```php
try {
  $conn = new PDO("mysql:host=$servername;dbname=$dbname",
$username, $password);
  // set the PDO error mode to exception
  $conn->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);
  $sql = "INSERT INTO MyGuests (firstname, lastname, email)
  VALUES ('John', 'Doe', 'john@example.com')";
  // use exec() because no results are returned
  $conn->exec($sql);
  echo "New record created successfully";
} catch(PDOException $e) {
  echo $sql . "<br>" . $e->getMessage();
}

$conn = null;
```

# HTML Forms

```
<form action="/action_page.php" method="get">
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname"><br>
  <label for="lname">Last name:</label><br>
  <input type="text" id="lname" name="lname">
  <input type="submit" value="Submit">
</form>
```

# Html Form elements

<input>

<label>

<select>

<textarea>

<button>

<fieldset>

<legend>

https://www.w3schools.com/html/html_form_elements.asp

# Upload file by form

Using HTML form and PHP code

# Php post

PHP $_POST is a PHP super global variable which is used to collect form data after submitting an HTML form with method="post". $_POST is also widely used to pass variables.

https://www.w3schools.com/php/php_superglobals_post.asp

# Php get

PHP $_GET is a PHP super global variable which is used to collect form data after submitting an HTML form with method="get".

$_GET can also collect data sent in the URL.

https://www.w3schools.com/php/php_superglobals_get.asp

# PHP_SELF

```
<form method="post" action="<?php echo $_SERVER['PHP_SELF']
;?>">
  Name: <input type="text" name="fname">
  <input type="submit">
</form>

<?php
if ($_SERVER["REQUEST_METHOD"] === "POST") {
  // collect value of input field
  $name = $_POST['fname'];
  if (empty($name)) {
    echo "Name is empty";
  } else {
    echo $name;
  }
}
?>
```

# Pdo insert with Form values

```
$sql = "INSERT INTO users (name, surname, sex) VALUES (?,?,?)";

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

$stmt->execute([$name, $surname, $sex]);
```

# Mysql Collation and character set

A collation in MySQL is a set of rules used to compare the characters in a specific character set. It is a sequence of orders to any particular set. MySQL supports various character sets, and each character set always uses one or more collation, at least one default collation. MySQL does not allow us to have any two character sets use the same collation.

https://www.javatpoint.com/mysql-collation

# Pdo delete row

```
try {
  $conn = new PDO("mysql:host=$servername;dbname=$dbname",
$username, $password);
  // set the PDO error mode to exception
  $conn->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);

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

  // use exec() because no results are returned
  $conn->exec($sql);
  echo "Record deleted successfully";
} catch(PDOException $e) {
  echo $sql . "<br>" . $e->getMessage();
}
```

# Pdo update row

```php
try {
  $conn = new PDO("mysql:host=$servername;dbname=$dbname",
$username, $password);
  // set the PDO error mode to exception
  $conn->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);

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

  // Prepare statement
  $stmt = $conn->prepare($sql);

  // execute the query
  $stmt->execute();

  // echo a message to say the UPDATE succeeded
  echo $stmt->rowCount() . " records UPDATED successfully";
} catch(PDOException $e) {
  echo $sql . "<br>" . $e->getMessage();
}
```

# Relations

One To One

One To Many

Many To Many

## Handling relationships using MYSQL

- **Primary Keys:** is a unique identifier for a row of data.
- Making a column a PRIMARY KEY is essentially equivalent to adding NOT NULL and UNIQUE constraints to that column.
- Example for creating primary key:

```
userId INT AUTO_INCREMENT PRIMARY KEY,
```

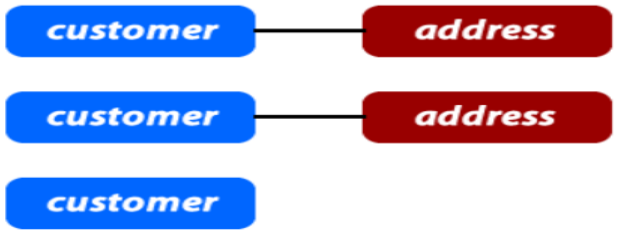
## Handling relationships using MYSQL

- A **Foreign Key** allows us to associate a row in one table to a row in another table. This is done by setting a column in one table as a Foreign Key and having that column reference another table's Primary Key column.
- Example for creating foreign key in mysql:

```
userId INT, CONSTRAINT fk_user FOREIGN KEY (userId)

REFERENCES users(userId)

ON UPDATE CASCADE

ON DELETE CASCADE
```

# 1:1 relationship

- Representing 1:1 relationship




users

(PK)

| id | full_name | enabled | last_login |
|---|---|---|---|
| 1 | John Smith | f | 2017-10-25 10:26:10.015152 |
| 2 | Alice Walker | t | 2017-10-25 10:26:50.295461 |
| 3 | Harry Potter | t | 2017-10-25 10:26:50.295461 |
| 5 | Jane Smith | t | 2017-10-25 10:36:43.324015 |

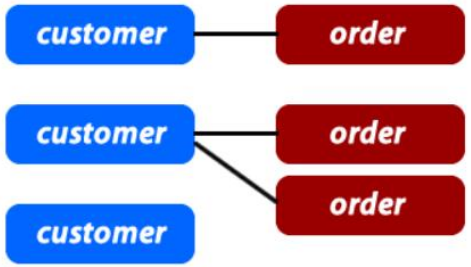
addresses

(PK) (FK)

| user_id | street | city | state |
|---|---|---|---|
| 1 | 1 Market Street | San Francisco | CA |
| 2 | 2 Elm Street | San Francisco | CA |
| 3 | 3 Main Street | Boston | MA |

# 1:M relationship

- A **one-to-many** relationship exists between two entities if an entity instance in one of the tables can be associated with multiple records (entity instances) in the other table. The opposite relationship does not exist; that is, each entity instance in the second table can only be associated with one entity instance in the first table.
- Example: A review belongs to only one book. A book has many reviews.



# 1:M relationship

- Representing 1:M relationship:

books

(PK)

| id | title | author | published_date | isbn |
|---|---|---|---|---|
| 1 | My First SQL book | Mary Parker | 2012-02-22 12:08:17.320053-03 | 981483029127 |
| 2 | My Second SQL book | John Mayer | 1972-07-03 09:22:45.050088-07 | 857300923713 |
| 3 | My Third SQL book | Cary Flint | 2015-10-18 14:05:44.547516-07 | 523120967812 |

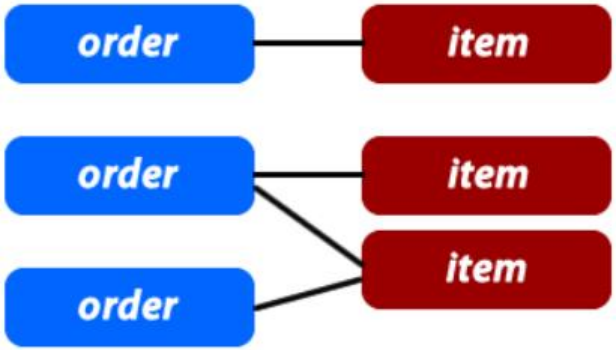
reviews

(PK) (FK)

| id | book_id | reviewer_name | content | rating | published_date |
|---|---|---|---|---|---|
| 1 | 1 | 'John Smith' | 'My first review' | 4 | 2017-12-10 05:50:11.127281-02 |
| 2 | 2 | 'John Smith' | 'My second review' | 5 | 2017-10-13 15:05:12.673382-05 |
| 3 | 2 | 'Alice Walker' | 'Another review' | 1 | 2017-10-22 23:47:10.407569-07 |

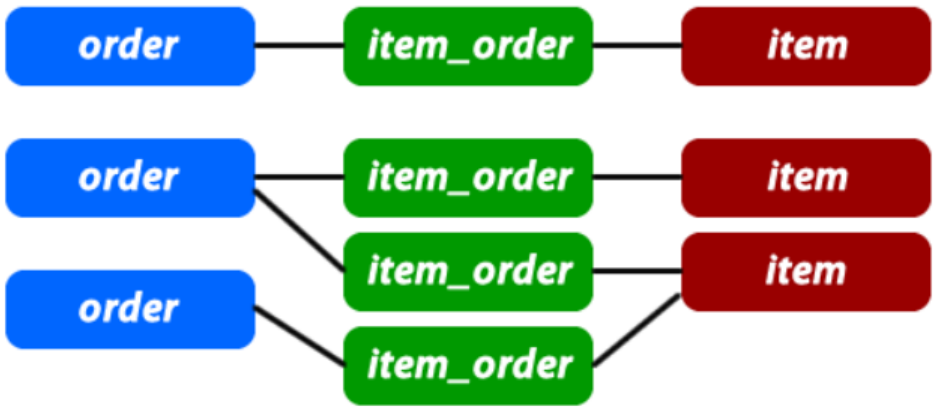
# M:N relationship

- Representing N:M relationship:



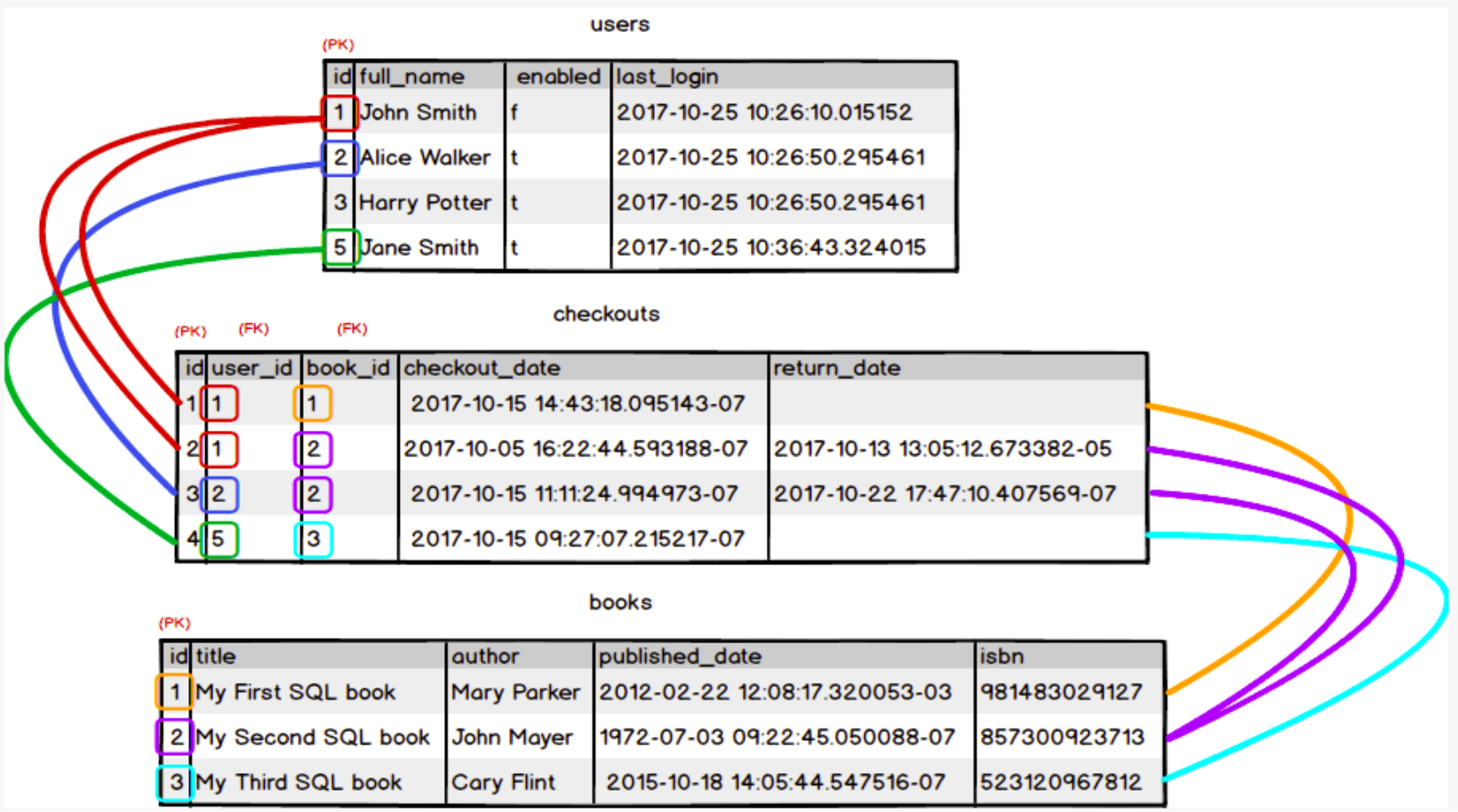
# M:N relationship

- Representing N:M relationship:



# M:N relationship

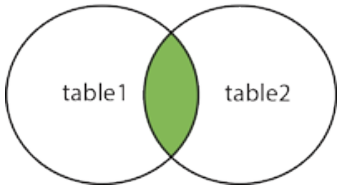
- Representing N:M relationship:



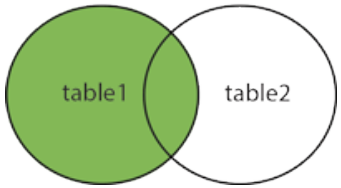
# Join

https://www.w3schools.com/MySQL/mysql_join.asp

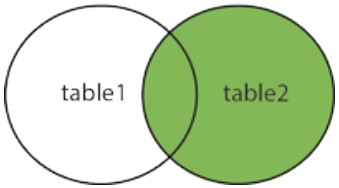
INNER JOIN




LEFT JOIN




RIGHT JOIN



# Mysql storage engines

InnoDB

MyISAM

https://hevodata.com/learn/myisam-vs-innodb/

# Group by

SELECT *column_name(s)*
FROM *table_name*
WHERE *condition*
GROUP BY *column_name(s)*
ORDER BY *column_name(s);*

# Having

SELECT *column_name(s)*
FROM *table_name*
WHERE *condition*
GROUP BY *column_name(s)*
HAVING *condition*
ORDER BY *column_name(s);*

# Difference between Where and Having

https://www.geeksforgeeks.org/difference-between-where-and-having-clause-in-sql/

# More about sql

https://www.w3schools.com/sql/default.asp

KOICA
Korea International Cooperation Agency

السويد
Sverige

ECTD
Egyptian Council
For Training & Development

التوظيف و العمل اللائق للمرأة

منظمة العمل الدولية

هيئة الأمم المتحدة للمرأة

المجلس القومي للمرأة
THE NATIONAL COUNCIL FOR WOMEN

# Thank you

php