Database concepts (CBD)

Lesson plan

Class 1

Database

- Collection of information structured so it can easily be accessed, managed and updated
  Typically = data gathered in records or files (containing information about a specific theme)

- In a *relational database*,
  = information structured into rows, columns and tables (indexed to make it easier to search)

Types of databases

- Databases = invented in the 1960s (great evolution since then)
  Started with hierarchical and network databases

- 1980s = object-oriented databases
  Today = SQL, NoSQL and cloud databases

- Databases = more often classified based on their organizational approach

Relational database

- Invented by *E.F. Codd* (IBM, 1970)
  Relational database = *tabular database*
  Data is structured making it easy to reorganized and accessed it in various ways
  Often using *Structured Query Language* (SQL)

- Made up of a collection of tables
  Tables contain data of predefined categories
  Each table has at least one data category in a column (*field*)
  Each row (*entries*) has a certain data instance for the categories which are defined in the columns

Distributed database

- Portions of the database are stored in multiple physical locations
  AND processing is dispersed or replicated among different points in a network

- Can be homogeneous or heterogeneous:

  - **Homogeneous** = all the physical locations system have the same underlying hardware + run the same applications

  - **Heterogeneous** = the hardware, operating systems or database applications may be different at each of the locations.

Cloud database

- Database that has been optimized or built for a virtual environment
  (hybrid, public or private cloud)

- Provides benefits such as ability to pay for storage capacity + bandwidth based on usage
  AND provide scalability on demand + high availability.

NoSQL database

- Useful for large sets of distributed data

- Especially effective for big data performance
  (where relational databases can't always manage efficiently.)

Object-oriented database

- Object-oriented databases
  = well-suited for items created using object-oriented programming languages

- Structured around objects
  (instead of being structured around actions and around data instead of logic.)

  Example :
  A multimedia record in a relational database can be a definable data object,
  as opposed to an alphanumeric value.

Graph database

- Growing in popularity
  = NoSQL databases using graph theory to store, map and query relationships
- Basically, consists in collections of nodes and edges
- (node represents an entity + edge/line represents connection between nodes
- Often employ SPARQL (declarative programming language and protocol for graph database analytics)
  = Allows to perform all the analytics that SQL can perform
  AND can be used for semantic analysis (the examination of relationships)

MySQL database

- *Structured Query Language*
  = To communicate and interact with a database

- Database = made of tables containing entries (rows) and fields containing values (columns).

Table

- Each row represents an entry
- Each column represent different types of information (field) contained in every entry.

Example :

Name	Age	Phone	email
John Smith	52	514 123-4567	him@fakemail.com
Jane Doe	25	514 123-4567	her@fakemail.com
Ludwig Roth	33	514 123-4567	him@fakemail.com
Susan Cohen	41	514 123-4567	her@fakemail.com

MySQL

- Coding language for :

  - Searching and interacting with a database
  - Modifying table's informations (adding or removing entries, etc)

- Need to work online / MAMP

Primary keys

- Used just as an ID number identifying an entry
  (for situation where infos are similar. e.g.: same name)

Importing a SQL database to MAMP

- Downloading the proper SQL file
- Run MAMP (make sure PHP and MySQL are running - green buttons should show).
- Click *Open WebStart Page* and from the web page menu
- Select TOOLS / PHPADMIN, which will open the PHPADMIN page
- Click the IMPORT tab + select the SQL file you want to import to MAMP
- Select the DB from the left panel (purple background indicates the DB is selected)

STUDENTS INSTALL DATABASE IN MAMP

SQL queries

DEMONSTRATION ON VIDEO
FOLLOWED BY STUDENTS TRYING THEMSELVES

- Click SQL tab of PHPADMIN
  Write code + hit GO button

Selecting a table

- **SELECT * FROM** `*table's name*`

Selecting specific fields from a table

- **SELECT** `*field's name*`**,** `password` **FROM** `*table's name*`

Applying and offset to the results:

- SELECT * FROM `*table's name*` **LIMIT** *5* **OFFSET** *5*

Sorting the results

- SELECT * FROM `*table's name*` **ORDER BY** `*field's name*` **DESC LIMIT 10**

  important:
  LIMIT must be written after ORDER BY or the query would generate an error

- When selecting entries from a table, it is convenient to be able to show the result in a certain order (e.g.: ID number or alphabetical order). To do so, the ORDER BY command is used. The DESC command (standing for "descending") may be used to display the entries from the highest to the smallest id.

Refining queries

- SELECT `*field's name*`, `*field's name*` FROM `*table's name*` **WHERE** `*another field's name*` **<= 10**

- **Example:**
  SELECT `first_name`, `last_name` FROM `users` **WHERE `id` = 4**

  See examples in course notes #1

ASSIGNMENT 01

Have students generate various results using SQL queries.

Class 2

MySQL (SUITE)

DEMONSTRATION ON VIDEO
FOLLOWED BY STUDENTS TRYING THEMSELVES

Refining queries (suite)

- HERE: we are searching the addresses table for entries in the state of California (CAL)
  SELECT * FROM `addresses` **WHERE** `state_province` **=** "CAL"

Refining queries using multiple rules (*AND* / **OR**)

- SELECT * FROM `addresses` WHERE `state_province` **=** "CAL" **AND** `address_id` **= 8**
- SELECT * FROM `addresses` WHERE `state_province` **=** "CAL" **OR** `state_province` **=** "UK"

Refining queries using comparisons

- SELECT * FROM `products` WHERE `product_name` **LIKE** "**%light%**"

- Use of wildcard %
  (%light, light% or %light%)

Refining queries using exclusions

- SELECT * FROM `addresses` WHERE **NOT** `address_line_2` = "South Park"

Working with numbers

COUNT()

- To count the number of entries based on a table's given field
  SELECT **COUNT**(`username`) FROM `users`
  SELECT **COUNT**(`username`) FROM `users` WHERE `user_email` LIKE "%southpark%"

AVG()

- Function calculates the average
  SELECT **AVG**(`product_price`) FROM `products`

SUM()

- Function makes an addition
  SELECT **SUM**(`product_stock`) FROM `products`

Modifying a database in MySQL

DEMONSTRATION ON VIDEO
FOLLOWED BY STUDENTS TRYING THEMSELVES

Adding to a database

Inserting a new row (entry) in a table

- First need to get familiar with the table
  (know how many entries and how many fields there are.,
  if the primary key (ID) is set to *AUTO_INCREMENT*, etc.)

- **INSERT INTO** `*table's name*` (`*table's field 1*`, `*table's field 2*`) VALUES (`*value 1*`, `*value 2*`)

  Example :

  INSERT INTO `users` (`username`, `password`, `user_email`, `first_name`, `last_name`) **VALUES** ("NewBrian", "12345", "nb@fakemail.com", "Brian", "New")

  Nota : no ID because = AUTO_INCREMENT set

Updating an entry

- UPDATE command to select the proper table
- SET command to indicate which field needs to be updated
- Supply the new information to replace current values

- **UPDATE** `users` **SET** `username` **=** "quentin", `user_email` **=** "qw@mail.com" **WHERE** `username` **=** "quentin92"

Deleting an entry

- DELETE command will be used
  BUT comes with great risks = make a backup of the database before

- **DELETE** FROM `users` WHERE `first_name` = "Sibu"

ASSIGNMENT 02

Have students generate various results using refined queries and numbers, and inserting rows to existing tables.

Class 3

MySQL (suite)

DEMONSTRATION ON VIDEO
FOLLOWED BY STUDENTS TRYING THEMSELVES

Creating database elements

Content type

- Each entry is composed of values from different categories (fields)
  Each of these categories store a specific type of data

  EXAMPLE:
  ID will usually use numbers (type = integer)
  Name would be a simple string (type = text)

- IMPORTANT

  "Null" column = constraint (require)
  "YES" = must supply content
  "NO" = empty field accepted

  "Extra"= where you may add *AUTO_INCREMENT*
  (automatic generation of a primary key number)

Creating a new table

- *CREATE* command + *TABLE* + *table's name* + ()
- In the parenthesis = Fields to add + parameters

EXAMPLE:

```
CREATE TABLE `comments` (
        comment_id INT NOT NULL AUTO_INCREMENT,
        comment_title VARCHAR(255) NOT NULL,
        comment_description TEXT(1000) NOT NULL,
        users_id INT NOT NULL,                    // To link to specific user
        product_id INT NOT NULL,                  // To link to users table
        PRIMARY KEY (comment_id)
)
```

Deleting a table

DROP TABLE `*table's name*`

Adding a field to a table

- ALTER + TABLE + table's name + ADD + data type and constraint

ALTER TABLE `*table's name*` **ADD** `new_column` **VARCHAR(255)**;

NB:
CHAR, VARCHAR and TEXT are used for text.
Number in () = maximum number of characters allowed

Modifying a table's field

ALTER TABLE `*table's name*` **MODIFY** `new_column` **TEXT NOT NULL**;

Deleting a table's field

ALTER TABLE `*table's name*` **DROP** `new_column`;

Linking tables

EXAMPLE :
when creating a comments table, we want for the comment to be linked to a specific user. Instead of repeating the information, we can manage to have this information automatically set using what is called a *foreign key*.

Foreign keys

FROM THE DATABASE EXAMPLE

The addresses table = primary key located in **field #1** (*address_id*)
another primary key in **field #2** (*users_id*)
=> These are linking the table to the users field of the users table

SELECT users.username, users.first_name, users.last_name, addresses.state_province **FROM** addresses
JOIN users **ON** addresses.users_id = users_id

Explanation :

SELECT = to select information from the specified fields.
FROM = to indicate from which initial table to search from.
JOIN = to indicate the second table that is linked to the first one.
ON = to tell to retrieve information of entries in which these two fields match.

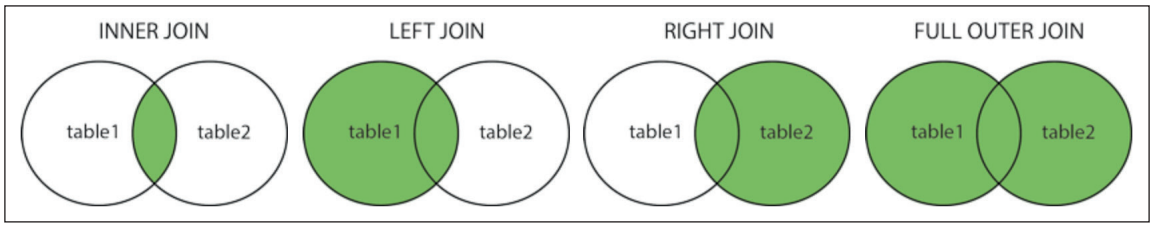
Types of JOIN

(INNER) JOIN
Returns records that have matching values in both tables

LEFT (OUTER) JOIN
Returns all records from the left table, and the matched records from the right table

RIGHT (OUTER) JOIN
Returns all records from the right table, and the matched records from the left table

FULL (OUTER) JOIN
Returns all records when there is a match in either left or right table



ASSIGNMENT 03

Have students create new tables, adding and modifying fields of existing tables, and linking tables together using foreign keys and JOIN command.

Class 4

MySQL (suite)

DEMONSTRATION ON VIDEO
FOLLOWED BY STUDENTS TRYING THEMSELVES

Linking tables (suite)

- Some tables are exclusively composed of IDs
  (only showing the values of other tables primary keys)

  Example : This is the case with the *orders* table of our example database *learning_over_here.sql*.

  ```
  SELECT orders.`order_id`, products.`product_name`, users.`first_name`, addresses.`city`
  FROM orders
  JOIN products ON orders.`product_id` = products.`product_id`
  JOIN users ON orders.`user_id` = `user_id`
  JOIN addresses ON orders.`address_id` = addresses.`address_id`
  ```

  Note :
  You may write the query on multiple lines, like above, to make it easier to read.

Using aliases

- Alias = shortcut that allows you to use an alias instead of the complete table's name

- To do so = specify the alias that will replace the table's name using the command AS

  ```
  SELECT orders.`order_id`, products.`product_name` FROM orders JOIN products ON
  orders.`product_id` = products.`product_id`

  SELECT o.`order_id`, p.`product_name` FROM orders AS o JOIN products AS p ON
  o.`product_id` = p.`product_id`
  ```

Declaring foreign key

- Declaring a foreign key makes the table's foreign key clickable
- Many ways to do so.

Declaring a foreign key when creating a table

```
CREATE TABLES Orders (
        OrderID int NOT NULL,
        ProductID int,
        PRIMARY KEY (OrderID),
        FOREIGN KEY (ProductID) REFERENCES Products(ProductID)
);
```

Declaring a foreign key by modifying a table

```
ALTER TABLE orders
ADD FOREIGN KEY(product_id) REFERENCES products(product_id)
```

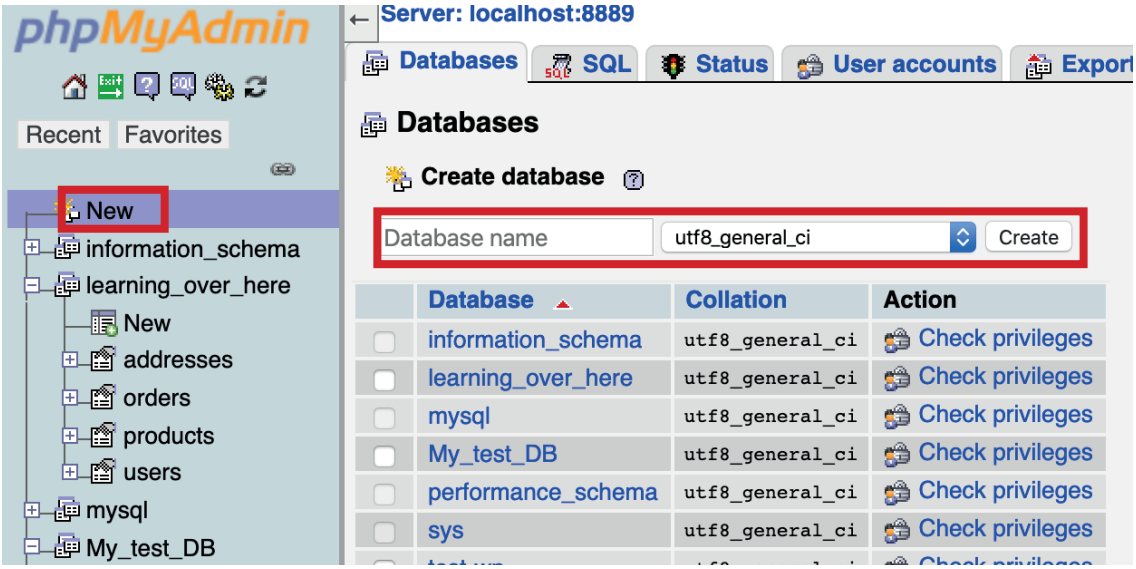
Deleting a foreign key

To delete correctly a foreign key, it won't be possible to simply use the field's name (this would cause an error. In the proper table, click the structure tab and select the relation view and use the name indicated in this panel.

```
ALTER TABLE orders DROP FOREIGN KEY orders_ibfk_1
```

Creating a new database

Creating the new database

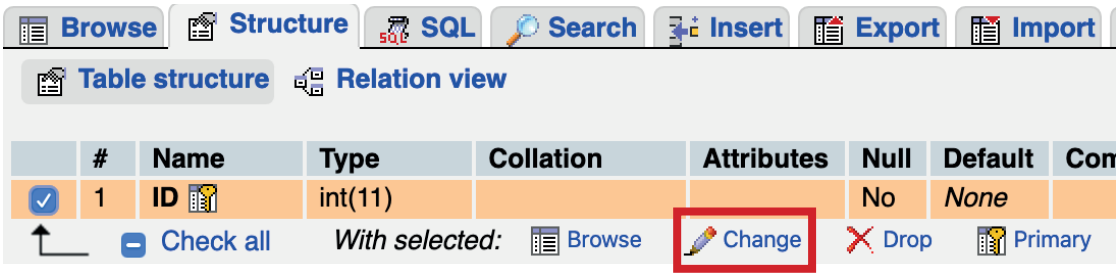


Creating the first table

- Once the database has been created = create its first table
  Write the name of the table + number of columns + Create button

Defining the table's fields

- Define the parameters for each field + save button
- To modify the table = check the proper checkbox + click «Change»



ASSIGNMENT 04

Create a basic database using phpMyAdmin in MAMP.

Class 5

Revision

Workshop

Class 6

Mid-term exam

Class 7
Linking a database to a PHP document

- MyPHPadmin can be use to modify a database and to make different queries
  BUT the real purpose of a database = to be used online + supply data to be displayed on web documents

- Necessary to link the database to a web document SO requests can be made from the web document

- The database must first me connected and the requests allowed THEN requests can be made to finally disconnect the database

Connecting to a MySQLi database in PHP

- MySQLi = improved version of MySQL (declared deprecated in 2011 and removed from PHP 5.5)

- MySQLi and PDO = object oriented that should now be used to prevent your application from *SQL injection*.

```
<?php
$db = new mysqli('localhost', 'root', 'root', 'My_test_DB');          // Server connection info
if($db->connect_errno > 0){
        die('Error : (. $db->connect_errno .') '. $db->connect_error);  // Error message
}
echo 'Connected successfully.';                                       // Success message
$mysqli->close();                                                     // Closing connection
?>
```

```
<?php
DEFINE('DB_USERNAME', 'root');                  // root or the predefined user name
DEFINE('DB_PASSWORD', 'root');                  // root or the predefined password
DEFINE('DB_HOST', 'localhost');                 // localhost when using MAMP
DEFINE('DB_DATABASE', 'My_test_DB');            // Name of the database
$mysqli = new mysqli(DB_HOST, DB_USERNAME, DB_PASSWORD, DB_DATABASE);
if (mysqli_connect_error()) {
        die('Connect Error ('.mysqli_connect_errno().') '.mysqli_connect_error());
}
echo 'Connected successfully.';
$mysqli->close();
?>
```

Explanation :

- *mysqli()* = to open a connection using the database's host, user name, password and named
- Error message = defined in a conditional structure using *mysqli_connect_error()* and *die()*
- *close()* = Closing connection

Retrieving content from a database (object oriented way)

```
<?php
$db = new mysqli('localhost', 'root', 'root', 'My_test_DB');        // Connection stored in variable

if ($db->connect_error) {
        die("Connection failed: " . $conn->connect_error);    // Error message
}

$sql = "SELECT * FROM library";                                     // Prepared statement
$result = $db->query($sql);                                         // Query stored in variable

if ($result->num_rows > 0) {                                        // If there is at least 1 row
        while($row = $result->fetch_assoc()) {                      // Output data of each row
                echo $row["title"]  . "<br>";
        }
} else {
        echo "0 results";                                           // If there is less than 1 row
}

$db->close();                                                       // Closing connection
?>
```

Explanation:

- After connecting to the database + defined the error message:
  Query is set up + stored in the variable ***$sql***
- Query is then run + the result is stored in the variable ***$result***
- ***num_rows()*** = to check if there is at least one row returned
- If more than zero rows = ***fetch_assoc()*** is used to store all the results into an *array* (***$row***)
- while() loop goes through the result set + outputs the data

Retrieving content from a database (procedural way)

```
<?php
...

$result = mysqli_query($db, $sql);

if (mysqli_num_rows($result) > 0) {
        while($row = mysqli_fetch_assoc($result)) {
        echo $row["title"] . "<br>";
        }
} else {
        ...
```

Displaying and ordering the data

```
<?php
$db = new mysqli('localhost', 'root', 'root', 'My_test_DB');
if ($db->connect_error) {
        die("Connection failed: " . $conn->connect_error);
}
$sql = "SELECT title, author, year FROM library ORDER BY year";
$result = mysqli_query($db, $sql);

if (mysqli_num_rows($result) > 0) {
        while($row = mysqli_fetch_assoc($result)) {
        echo "<b>" . $row["title"] . "</b>, " . $row["author"] . ", (" . $row["year"] .
")<br>";
        }
} else {
        echo "0 results";
}
$db->close();
?>
```

Explanation :

- Concatenation for the output + *ORDER BY* command in the query

Connecting and retrieving data from a database in PHP using PDO

- PHP Data Objects = set of PHP extensions making it easier to interact with databases.
- PDO manual: https://www.php.net/manual/fr/book.pdo.php

```
<?php
$username = root;
$password = root;
$db = new PDO("mysql:host=localhost; dbname=My_test_DB", $username, $password);
$table = 'library';
$stmt = $db->query('SELECT * from '.$table);
$db = NULL;

while($rows = $stmt->fetch()){
        echo $rows['title'];
};
?>
```

ASSIGNMENT 05

Example of assignment :
Create a database, have the data displayed on screen and style it efficiently using CSS.

Class 8
Writing to a database using PHP

Example (MySQLi Object-oriented)

```
<?php
// Server info
$servername = "localhost";
$username = "root";
$password = "root";
$dbname = "My_test_DB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection
if ($conn->connect_error) {
        die("Connection failed: " . $conn->connect_error);
}

// Query
$sql = "INSERT INTO library (title, author, year) VALUES ('My new book', 'So Me', '2020')";

// Check data creation
if ($conn->query($sql) === TRUE) {
        echo "New record created successfully";
} else {
        echo "Error: " . $sql . "<br>" . $conn->error;
}

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

Explanation :

- After defining server info + having created a successful connection
- Query is stored in the variable ***$sql***
- A conditional structure checks if the query was correctly executed
- Displays an error or a success message accordingly

Example (MySQLi Procedural)

```
<?php
// Server info
$servername = "localhost";
$username = "root";
$password = "root";
$dbname = "My_test_DB";

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

// Check connection
if (!$conn) {
   die("Connection failed: " . mysqli_connect_error());
}

// Query
$sql = "INSERT INTO library (title, author, year) VALUES ('My new book', 'So Me', '2020')";

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

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

Example (PDO)

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

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 (title, author, year) VALUES ('My Book', 'Me', '2020)";

// 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;
?>
```

Writing in a database using a PHP form

- Useful for = user subscribe to a mailing list / in the context of a CMS.

HTML:

```
<form action="#" method="post">
        <label for="title">Title:</label>
        <input type="text" name="title">

        <label for="author">Author:</label>
        <input type="text" name="author">

        <label for="year">Year:</label>
        <input type="text" name="year">

        <button>Create new entry</button>
</form>
```

PHP:

```
<?php
$db = new mysqli('localhost', 'root', 'root', 'My_test_DB');        // Connect to server

if ($db->connect_error) {                                           // Connection error message
        die("Connection failed: " . $db->connect_error);
}

$sql = "INSERT INTO library (title, author, year)                   // Query
VALUES ('$_POST[title]','$_POST[author]','$_POST[year]')";

if (!mysqli_query($db, $sql)){                                      // Query error message
        die('Error: ' . $db_error());
}

echo "1 record added";                                              // Confirmation message

$db->close();
?>
```

IMPORTANT: Always validate the form to prevent from malicious SQL injection.

Final project

Create a database using a table where users create an account using username and passwords. From the main page, users may create there account and login to the sit that will display a congratulation message (Could be subject of the next class/ workshop).

Class 9

Workshop

Class 10

Revision

Workshop : Group coding/correction of final project

Class 11

Final exam