Relationship between tables in Relational database tables:

* One-to-many relationships:
* A subject has many pages
* And each page belongs to subject

Best Practice in MySql

* Always create indexing for foreign keys it’s a good practice like INDEX (subject\_id).
* Always define a query in a separate variable. It is really handy because if we want to see the query or debug the query then we can see the query easily like echo the query.
* Always use single quotes in MySQL not double quotes.

SQL Language

DML Syntax:

* SELECT
* INSERT
* UPDATE
* DELETE

DDL Syntax:

* CREATE
* ALTER
* DROP

Database Server:

* A database server is an essential part of modern web applications. It stores and processes the content.
* Database servers may have many databases.
* Each database may contain many tables.

SQL Injection:

* In SQL syntax, text provided by the user should be always enclosed in single quotes: ‘-‘.
* The single quote character is part of the SQL language, not part of user’s input!
* Escaping strings for mysql

--Backslash before single-quote

Example --$menu\_name = “Today\’s widget Trivia”;

* For avoiding sql injection in mysql use mysqli\_real\_escape\_string($db,$string) designed for string
* Type casting for int or float.
* Mysqli\_real\_escape\_string() -- escapes all strings.
* What kind of things should be escaping those are following:
* We should don’t trust anything, anything means those are come from user whether from url string, form value, cookie value, even our database value which may have originated from somewhere else or some user also we also need to escape

Prepared Statements:

* Prepared statements are very useful against SQL injections.
* A prepared statement is a feature used to execute the same (or similar) SQL statements repeatedly with high efficiency.
* Prepared Statements Example: insert into subjects (menu\_name, position, visible) values (?,?,?)
* Advantages of prepare statement:
* Prepare statement once, reuse many times
* Faster
* Separate query for dynamic data
* Prevents SQL injection

Example:

$query = “SELECT id, first\_name, last\_name “;

$query .= “FROM users “;

$query .= “WHERE username = ? AND password = ?”;

$stmt = mysqli\_prepare($connection, $query);

Mysqli\_stmt\_bind\_param($stmt, ‘ss’, $username, $password);

Mysqli\_stmt\_execute($stmt);

Mysqli\_stmt\_bind\_result($stmt, $id, Sfirst\_name, $last\_name);

Mysqli\_stmt\_fetch($stmt);

Mysqli\_stmt\_close($stmt);

The ‘ss’ argument lists the types of data that the parameters are. The s character tells mysql that the parameter is a string.

Note: If we want to insert any data from external sources (like user input), it is very important that the data is sanitized and validated.

The argument may be one of four types:

* I – integer
* D - double
* S – string
* B – BLOB

We must have one of these for each parameter.

MySQL Database Connection Related Functions:

* Mysqli\_connect() - is used to connect the script with the database.
* Mysqli\_connect\_errno() - find the error number.
* Mysqli\_connect\_error() - find the error.
* Mysqli\_close() - close the connection with the database.

Retrieving Data from MySQL:

* Mysqli\_query() - which is used for our query. The mysqli\_query() function performs a query against the database.
* Mysqli\_fetch\_row() - if we need result back then we use this function.
* Mysqli\_free\_result() - if we no longer need of the recent data set then we tell php ok to flush the memory.

Some useful functions of MySql

* Mysqli\_error() - Returns the last error description for the most recent function call.
* Mysqli\_insert\_id() - Returns the auto-generated id used in the last query
* Mysqli\_num\_rows() - function returns the number of rows in a result set.
* Mysqli\_affected\_rows() - The mysqli\_affected\_rows() function returns the number of affected rows in the previous SELECT, INSERT, UPDATE, REPLACE, or DELETE query. An integer > 0 indicates the number of rows affected. 0 indicates that no records where affected. -1 indicates that the query returned an error
* Mysqli\_real\_escape\_string($db, $string) 🡪Escapes special characters in a string for use in an SQL statement. Always need to use this function to construct a string in MySQL.

PHP Database Interaction follows Five Steps

* Create a database connection
* Perform database query (If we use select then it might return data to us)
* Use returned data(if any)
* Release returned data
* Close database connection

Steps 1 and 5 happens once only php scripts but steps 2 to 4 may be repeat many times

$result = mysqli\_query($connection, $query)

Here $result is a resource; resource is a special kind of thing or object which is collection of database rows

Data fetch from Database

* Mysqli\_fetch\_row
* Results are in a standard array
* Keys are integers
* Mysqli\_fetch\_assoc
* Results are in an associative array
* Keys are column names
* Mysqli\_fetch\_array
* Results in either or both(standard or associative) types of arrays
* Results are configurable like as follows
* MYSQL\_NUM, MYSQL\_ASSOC, MYSQL\_BOTH
* Mysqli\_fetch\_object
* The mysqli\_fetch\_object() function returns the current row of a result set, as an object.
* Fieldnames returned from this function are case-sensitive.

The most useful and convenient and faster is mysqli\_fetch\_assoc among 4 of this.

Traditional Convention for Database:

1. MySQL commands or keywords in upper case while database names or columns or fields names in lower case.
2. Each table contains one type of information. Examples - products, customers, orders etc.
3. Avoid login as a root user instead login as a user for better security.
4. Primary key table in plural while foreign key in singular form like subjects (pk tbl name) and subject\_id (is foreign key).

CRUD in MySQL:

* In MySQL select or read statement is used for reading data from database.
* In MySQL insert statement is used for creating record in database.

Some Useful MySql Command:

1. Retrieving data from the database:

The basic form of a select statement is:

SELECT [options] items

[INTO file\_details]

FROM tables

[ WHERE conditions ]

[ GROUP BY group\_type ]

[ HAVING where\_definition ]

[ ORDER BY order\_type ]

[LIMIT limit\_criteria ]

[PROCEDURE proc\_name(arguments)]

[lock\_options];

1. SHOW DATABASES; //return the list of the databases.
2. USE db\_name; //for using selected databases.
3. DROP DATABASE db\_name; //for deleting a database
4. GRANT ALL PRIVILEGES ON db\_name.\*

TO ‘username’@’localhost’

IDENTIFIED BY ‘password’; //here .\* means all database tables of the database.

1. SHOW GRANTS FOR ‘username’@’localhost’;
2. SHOW TABLES; //shows all the tables.
3. SHOW COLUMNS FROM table\_name; //show all the columns from the table.
4. SHOW FIELDS FROM table\_name; // show all the columns from the table with the descriptions.
5. Create table syntax in SQL:

CREATE TABLE admins(

id INT(11) NOT NULL AUTO\_INCREMENT,

username VARCHAR(50) NOT NULL,

hashed\_password VARCHAR(60) NOT NULL,

PRIMARY KEY (id)

);

1. Create TABLE Subjects(

Id

Menu\_name – which is the shown in the public side?

Position – identify the position of the subjects.

Visible – identify the subject is shown or not.

);

Database APIs in PHP:

* Mysql and mysqli is pre-configured for MySQL and only used in Mysql
* PDO supports other database as well.
* PDO works with any database like postgres or others.
* PDO is only object oriented.
* Mysqli works on both like procedural and object oriented.
* Die() works as like exits or break.