Database & Web Dev

Database provides a way to implement one important software design principle

> One should separate that which varies from that which stays the same

MySQL

Relational database is composed of one or more tables

A table is 2D container containing Records and Fields

Records: rows

Fields: columns

Special column called primary key is used to identify each record

- primary key can be auto-incremented

Efficiency and Speed of MySQL databases

- columns in a database are associated with data structure called index
  - Created automatically for primary key
  - tree structure are often used
- database enforce rules about what kind of data can be stored
  - Types
    - BIT — a single bit
    - BLOB — Binary Large Object, used to store an image
    - CHAR(n) — fixed number of characters
    - DATE — a date, time and datetime
    - FLOAT — decimal number
    - DOUBLE
    - DECIMAL
    - INT — integer
    - SMALLINT
    - VARCHAR(n) — variable number of characters
    - TEXT — a string with maximum length 65,535 characters
- Foreign key can be used to relate different table on separate topics
  - Tables linked this way are in a relationship

SQL (Structured Query Language)

A language that allows us to create, access, update, and delete data in database

Create & Use Database

```
CREATE DATABASE databaseName;

USE databaseName;
```

Create Table in Database

```
CREATE TABLE table_name (
    columnName1 datatype,
    columnName2 datatype,
    columnName3 datatype,
    PRIMARY KEY (columnName1)
);
```

`SELECT` allows us to get data from table

```
SELECT ISBN10, Title FROM Books
```

- `SELECT` — SQL keyword that indicates the type of query
- `ISBN10, Title` — fields to retrieve
- `FROM` — SQL keyword for specifying the table
- `Books` — Table to retrieve from

`ORDER BY field_to_sort_by`

`WHERE expression take form field operator value`

`WHERE` category

`INSERT INTO`
primary key fields are often set to AUTO_INCREMENT — It will be unique

`UPDATE` — is used to modify existing records in table

- important to specify which table

```
UPDATE ArtWorks
SET Title='Night Watch', YearOfWork=1642, ArtistID=105
```

```
WHERE ArtWorkID=54
```

- AUTO_INCREMENT cannot have values updated

`DELETE`

```
DELETE FROM table_name WHERE condition;
```

Member Group By

```
SELECT Nationality, Count(ArtistID) AS NumArtists
FROM Artists
GROUP BY Nationality
```

MySQL extension — was the original extension to php for MySQL and has been replaced
mysqli extension — provides both a procedural & oop approach this extension supports latest features of MySQL
PHP data objects (PDOs) — provides abstraction layer with appropriate drivers

- unable to use latest MySQL Features
- not just MySQL databases

Accessing MySQL in PHP

1. Connect to database
2. Handle connection erros
3. Execute SQL query
4. Process results
5. Free resource and close connection

```
define('DBHOST', 'localhost');
define('DBNAME', 'databasename');
define('DBUSER', 'username');
define('DBPASS', 'mypassword');

$connectionString = 'mysql:host='.DBHOST.';bname='.DBNAME;

$pdo = new PDO($connectionString, DBUSER, DBPASS);
```

Use Try Catch

```
try {
    $pdo = new PDO($connectionString, DBUSER, DBPASS);
    $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    // other database things here
} catch(PDOException $e)
    die($e->getMessage());
```

Executing Query

```
$sql = "SELECT * FROM Categories ORDER BY CategoryName";
$result = $pdo->query($sql);
```

Processing the Results

```
while($row = $result->fetch()) {
    echo $row['ID']."-".$rowp['CategoryName'];
    echo "<br/>";
}
```

Close the Connection

```
$pdo = null;
```

Working with Parameters

```
// get input from user
$newCatName = $_POST['catName'];

$sql = "UPDATE Categories SET CategoryName = 'Web' where
    CategoryName = '{$newCatName}'";
$count = $pdo->exec($sql); // exec instead of query
echo "<p>Updated ".$count." rows</p>";
```

There's a problem with this because...

Prepared statements are used to improve the performance of multiple execution of queries, this can avoid SQL injection attacks

```
$sql = "INSERT INTO Books (ISBN10, Title, CopyrightYear, ImprintId, ProductionStatusId, TrimSize, Description) VALUES (:isbn, :title, :year, :imprint, :status, :size, :desc)";
$statement = $pdo->prepare($sql);
$stement->bindValue(':isbn', $_POST['isbn']);
$statement->bindValue(':title',$_POST['title']);
$statement->bindValue(':year', $_POST['CopyRightYear']);
$statement->bindValue(':imprint', $_POST['ImprintId']);
```

```
$statement->bindValue(':status', $_POST['ProductionStatusId']);
$statement->bindValue(':size', $_POST['TrimSize']);
$statement->bindValue(':desc', $_POST['Description']);
$statement->execute();
```