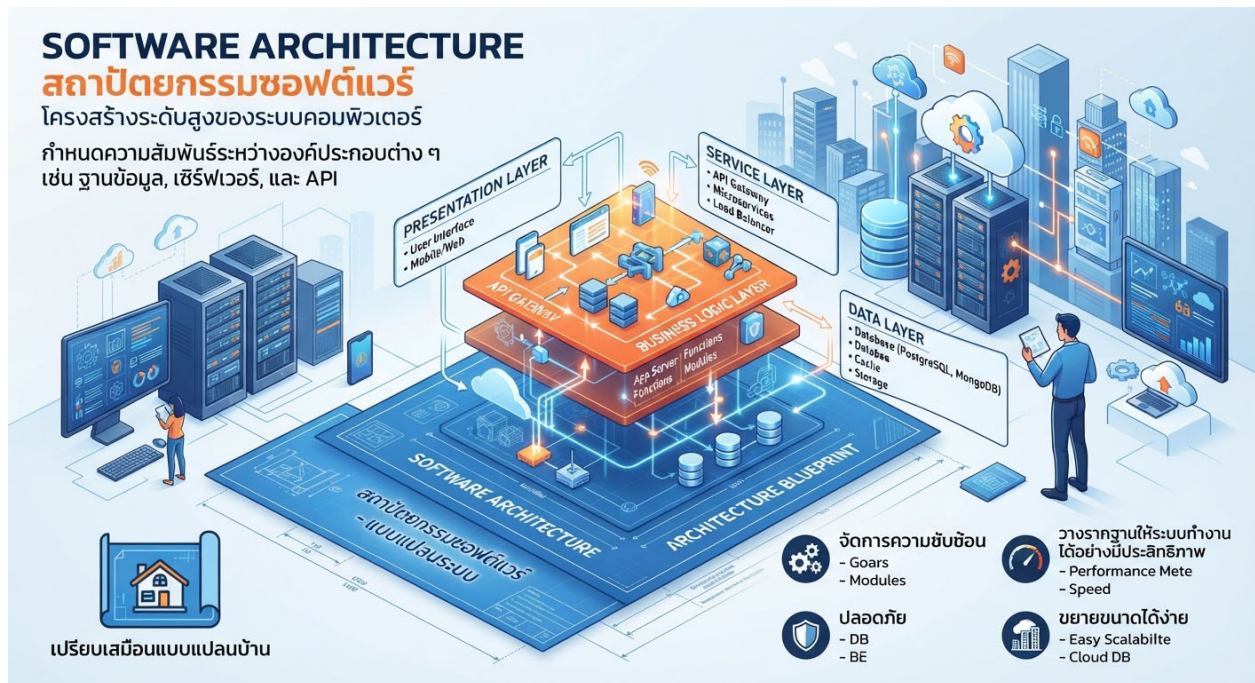


Software Architecture (สถาปัตยกรรมซอฟต์แวร์)



Software Architecture (สถาปัตยกรรมซอฟต์แวร์) คือ โครงสร้างระดับสูงของระบบคอมพิวเตอร์ ซึ่งกำหนดความสัมพันธ์ระหว่างองค์ประกอบต่าง ๆ เช่น ฐานข้อมูล, เซิร์ฟเวอร์, และ API เปรียบเสมือนแบบแปลนบ้าน ที่ช่วยจัดการความซับซ้อน และวางรากฐานให้ระบบทำงานได้อย่างมีประสิทธิภาพ ปลอดภัย และขยายขนาดได้ง่าย

องค์ประกอบสำคัญ

- **Components:** ส่วนประกอบหลักของระบบ เช่น ฐานข้อมูล, แอปพลิเคชัน หรือบริการย่อย (Services)
- **Connectors:** ตัวกลางหรือวิธีการที่ทำให้ส่วนประกอบต่างๆ สื่อสารและส่งข้อมูลถึงกันได้
- **Constraints:** ข้อจำกัดหรือกฎเกณฑ์ต่างๆ ในการออกแบบระบบ

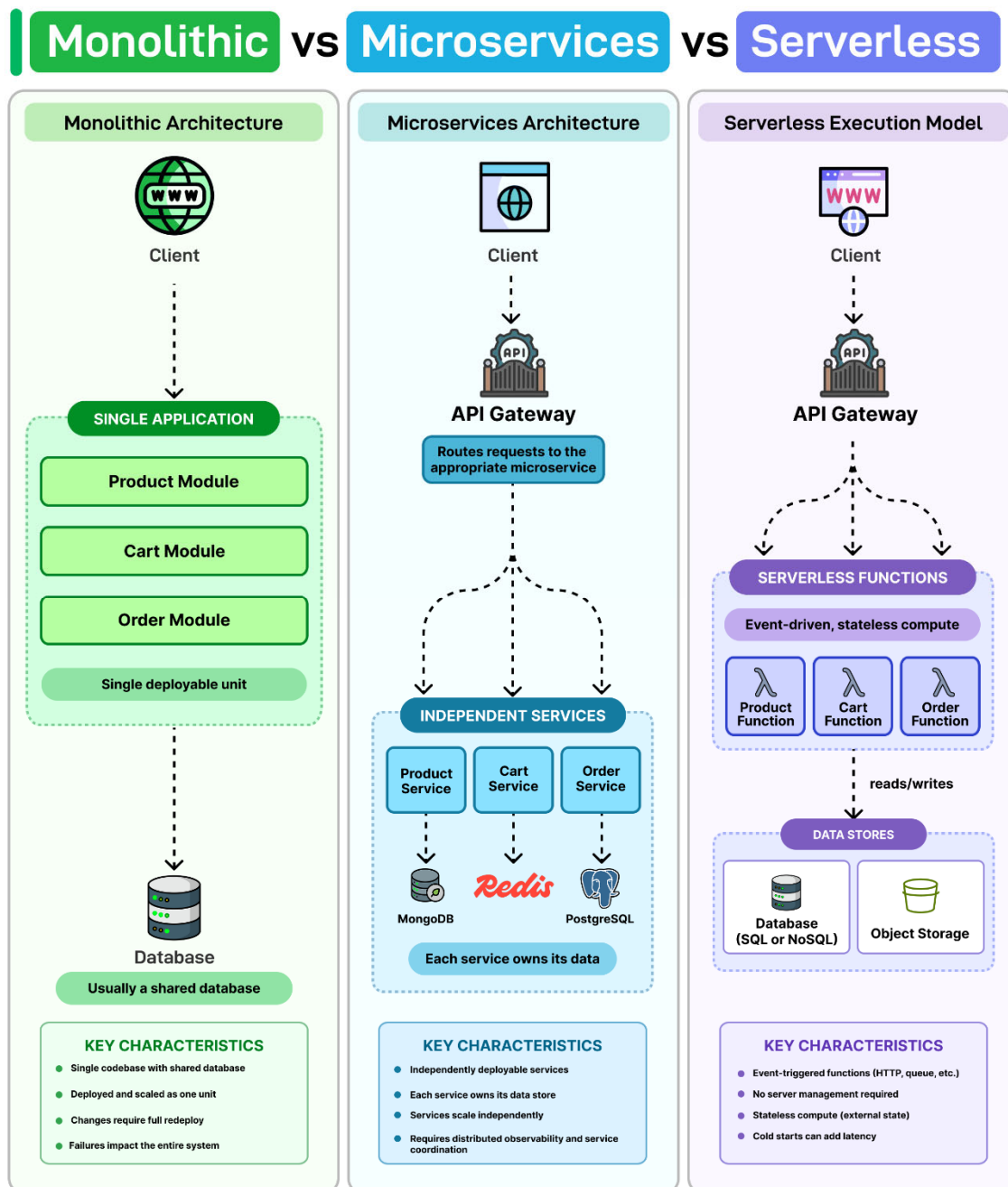
รูปแบบสถาปัตยกรรมที่พบบ่อย (Architectural Styles)

- **Monolith Architecture:** ระบบที่มีโค้ดและฟังก์ชันทั้งหมดรวมกันอยู่ในชุดเดียว เหมาะสำหรับโปรเจกต์ขนาดเล็กหรือเริ่มต้น
- **Microservices Architecture:** การแบ่งระบบใหญ่ออกเป็นบริการย่อย หลายตัวที่ทำงานและแยกฐานข้อมูลออกจากกันอย่างอิสระ

- **Layered (N-Tier) Architecture:** การแบ่งชั้นการทำงานออกจากกัน เช่น Presentation Layer, Business Logic Layer และ Data Access Layer

1. ภาพเปรียบเทียบโครงสร้าง Monolith vs Microservices vs Serverless

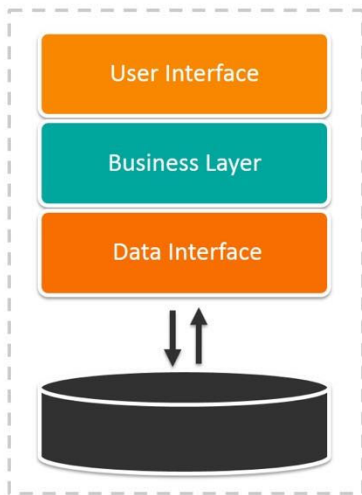
ภาพเปรียบเทียบสไตล์สถาปัตยกรรมหลักแบบคู่ขนาน ช่วยให้นักศึกษาเห็นว่า Monolith จะมั่วรวมทุกอย่างไว้ที่ก้อนเดียว ส่วน Microservices จะแยกเป็นอิสระต่อกัน และ Serverless จะจัดเป็นฟังก์ชันตามการใช้งาน



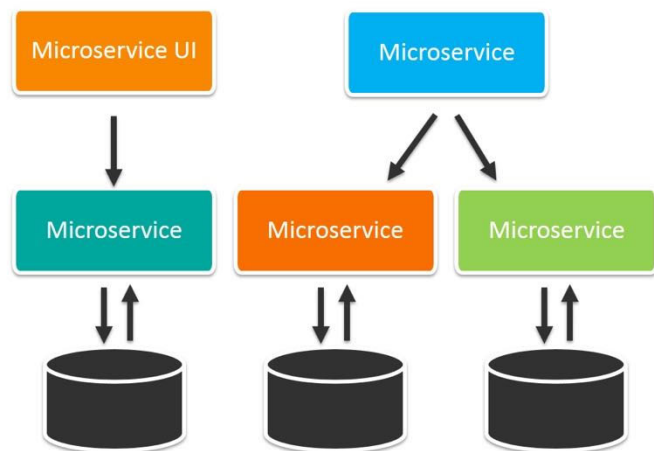
2. แผนผังจำลองการทำงานภายในระบบ (Architecture Diagrams) ไดอะแกรมแสดงการสื่อสารของข้อมูล

ไดอะแกรมแสดงการสื่อสารของข้อมูล ภาพกลุ่มนี้จะช่วยอธิบายโครงสร้างภายใน (เช่น หน้าจอ UI, ส่วนประมวลผล Business Logic และฐานข้อมูล Database) ทำให้นักศึกษาเข้าใจคำว่า "Components" และการต่อสาย "Connectors" ภายในระบบได้ง่ายขึ้น

Monolithic Architecture



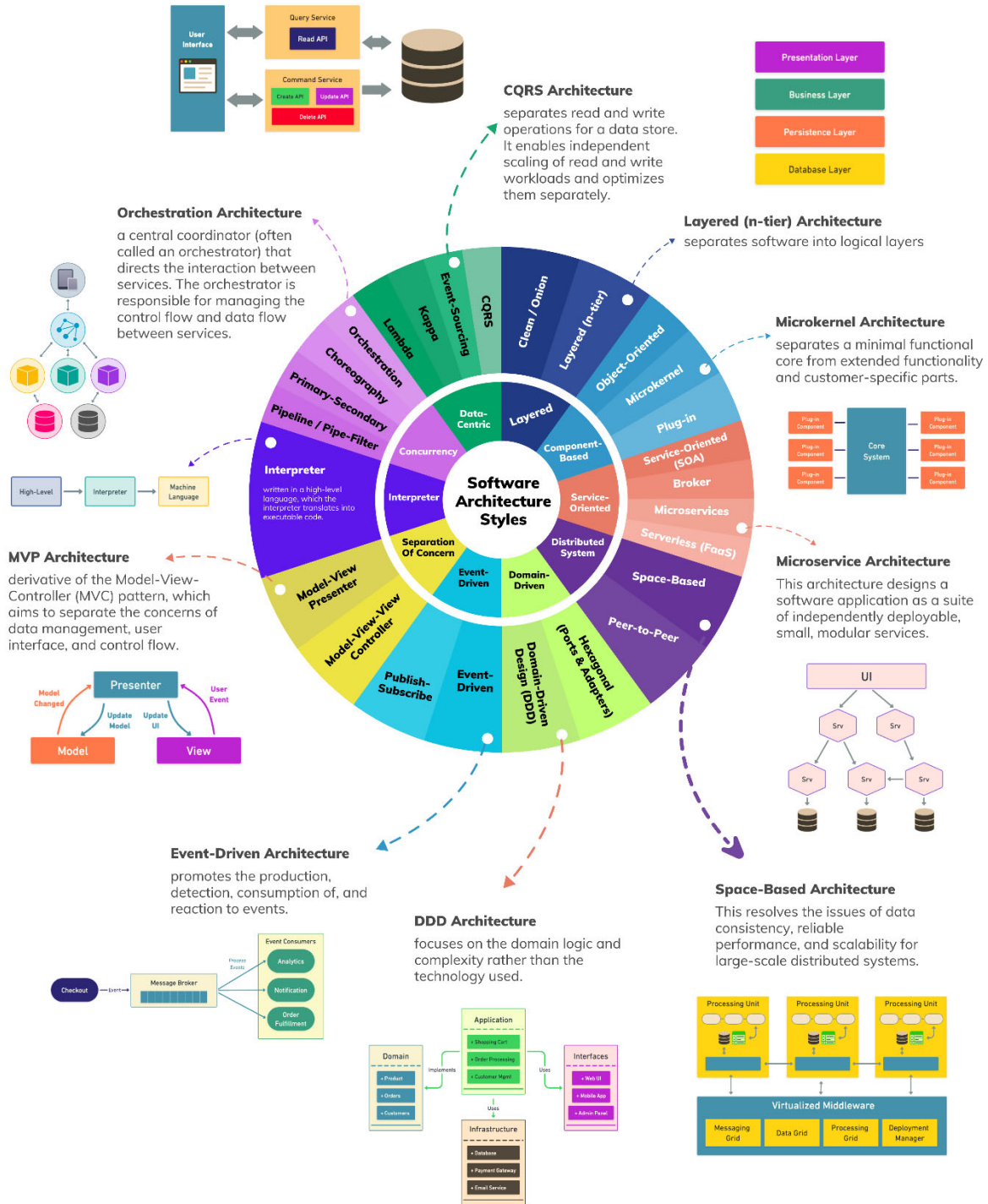
Microservices Architecture



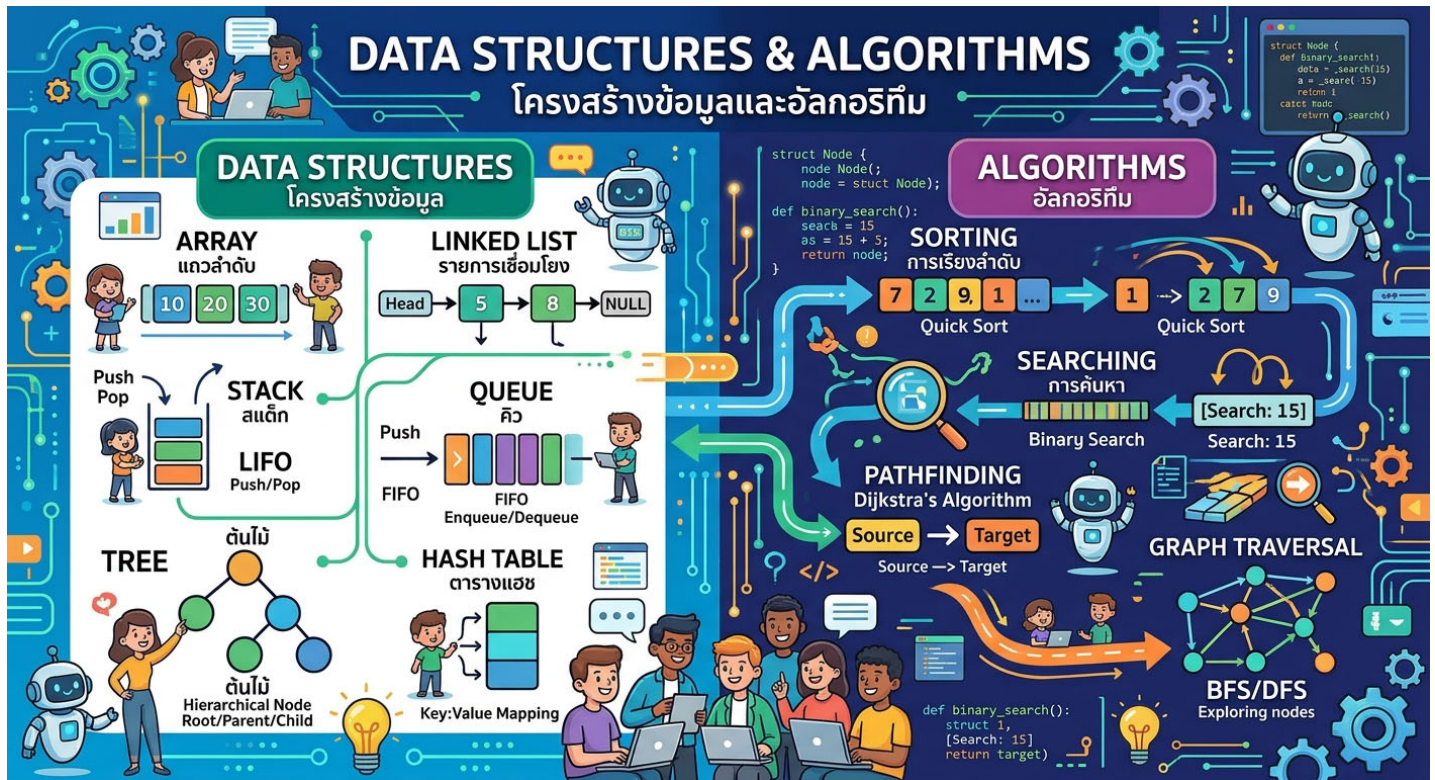
3. ภาพสรุปภาพรวมในรูปแบบ Infographic

แผนผังวงล้อและภาพรวมสรุปรูปแบบ (Software Architecture Patterns) เหมาะสำหรับการเปิดเป็นสไลด์นำเข้าบทเรียน เพื่อให้นักศึกษาเห็นแผนผัง Roadmap หรือสไตล์การออกแบบซอฟต์แวร์ที่หลากหลายในอุตสาหกรรมปัจจุบัน

Software Architecture Styles



Data Structures (โครงสร้างข้อมูล)



Data Structures (โครงสร้างข้อมูล) และ Algorithms (อัลกอริทึม) คือรากฐานสำคัญของการเขียนโปรแกรม และการพัฒนาซอฟต์แวร์ โดยสรุปความหมายและการทำงานได้

1. Data Structures (โครงสร้างข้อมูล)

- **ความหมาย:** คือวิธีหรือรูปแบบในการจัดระเบียบและจัดเก็บข้อมูลไว้ในหน่วยความจำของคอมพิวเตอร์ เพื่อให้สามารถเรียกใช้ ค้นหา และแก้ไขข้อมูลได้อย่างมีประสิทธิภาพ
- **ประเภทหลัก ๆ:**
 - **แบบเชิงเส้น (Linear):** ข้อมูลเรียงต่อกันไปทีละลำดับ เช่น Array, Stack (เข้าที่หลังออกก่อน), Queue (เข้าก่อนออกก่อน), Linked-List
 - **แบบไม่เชิงเส้น (Non-Linear):** ข้อมูลมีความสัมพันธ์กันแบบเครือข่ายหรือลำดับขั้น เช่น Tree (ต้นไม้) และ Graph (กราฟ)

ตัวอย่างการเขียน Array และ Linked List ในภาษา PHP เพื่อให้เห็นภาพการทำงานและโครงสร้างการเขียนโค้ดที่ชัดเจนครับ

1. การเขียน Array ใน PHP

ในภาษา PHP ตัว Array จะมีความยืดหยุ่นสูงมาก (เป็นแบบ Dynamic Array ที่ขยายขนาดอัตโนมัติ) และสามารถทำเป็นทั้งแบบมีดัชนีเป็นตัวเลข (Indexed Array) หรือเป็นข้อความคีย์ (Associative Array) ก็ได้

ตัวอย่างโค้ด:

php

```
<?php
// 1. สร้าง Array แบบธรรมดา (Indexed Array) เก็บข้อมูลคะแนนสอบ
$scores =;

// 2. การเข้าถึงข้อมูล (Access) ด้วย Index (เริ่มจาก 0)
echo "คะแนนในช่องแรก: " . $scores[0] . "\n"; // ผลลัพธ์: 85

// 3. การแก้ไขข้อมูล (Update)
$scores[2] = 80; // เปลี่ยนจาก 78 เป็น 80

// 4. การเพิ่มข้อมูลต่อท้าย (Append)
$scores[] = 88;

// 5. แสดงผล Array ทั้งหมดด้วย Loop
echo "ข้อมูลใน Array: ";
foreach ($scores as $score) {
    echo $score . " ";
}
// ผลลัพธ์รวม: 85 92 80 90 88
?>
```

Use code with caution.



2. การเขียน Linked List ใน PHP

เนื่องจาก PHP ไม่มีตัวชี้หน่วยความจำ (Pointer) แบบภาษา C แต่เราสามารถใช้ Object References (การอ้างอิงออบเจกต์) เพื่อสร้าง Node ที่เชื่อมโยงกันได้เหมือนกันครับ

ตัวอย่างโค้ด:

php

```
<?php
// 1. นิยามโครงสร้างของ Node
class Node {
    public $data; // สำหรับเก็บข้อมูล
    public $next; // สำหรับเก็บตัวชี้ไปยัง Node ถัดไป (Object ถัดไป)

    public function __construct($data) {
        $this->data = $data;
        $this->next = null; // ดอนแรกตั้งให้เป็นว่างเปล่าก่อน
    }
}

// 2. นิยามโครงสร้างของ Linked List
class LinkedList {
    public $head = null; // จุดเริ่มต้นของ List

    // ฟังก์ชันสำหรับเพิ่มข้อมูลต่อท้าย List
    public function append($data) {
        $newNode = new Node($data);

        // ถ้า List ยังว่างอยู่ ให้ Node ใหม่เป็นจุดเริ่มต้น (Head)
        if ($this->head === null) {
            $this->head = $newNode;
            return;
        }

        // ถ้ามีข้อมูลอยู่แล้ว ให้เดินตามหา Node สุดท้าย
        $last = $this->head;
        while ($last->next !== null) {
            $last = $last->next;
        }

        // เชื่อมโยง Node สุดท้ายเดิม เข้ากับ Node ใหม่
        $last->next = $newNode;
    }
}
```

```

// ฟังก์ชันสำหรับพิมพ์ข้อมูลทั้งหมดออกมาดู
public function printList() {
    $current = $this->head;
    while ($current != null) {
        echo $current->data . " -> ";
        $current = $current->next; // เลื่อนไปยัง Node ถัดไป
    }
    echo "NULL\n";
}
}

// --- วิธินำไปใช้งาน ---
$myList = new LinkedList();
$myList->append(10);
$myList->append(20);
$myList->append(30);

// แสดงผลลัพธ์
$myList->printList();
// ผลลัพธ์: 10 -> 20 -> 30 -> NULL
?>

```

Use code with caution.



3. การเขียน Stack ใน PHP

PHP มีคลาสมาตรฐานชื่อ SplStack สำหรับจัดการข้อมูลแบบ Stack โดยเฉพาะ

php

```
<?php
// สร้าง Stack
$stack = new SplStack();

// 1. Push (เพิ่มข้อมูลเข้าสู่ Stack)
$stack->push("หนังสือเล่มที่ 1");
$stack->push("หนังสือเล่มที่ 2");
$stack->push("หนังสือเล่มที่ 3");

// ดูข้อมูลตัวบนสุด (Peek)
echo "ตัวบนสุดตอนนี้: " . $stack->top() . "\n"; // ผลลัพธ์: หนังสือเล่มที่ 3

// 2. Pop (นำข้อมูลตัวบนสุดออกจาก Stack)
$removedItem = $stack->pop();
echo "ข้อมูลที่ถูกนำออก: " . $removedItem . "\n"; // ผลลัพธ์: หนังสือเล่มที่ 3

// ดูข้อมูลตัวบนสุดใหม่
echo "ตัวบนสุดหลังจาก Pop: " . $stack->top() . "\n"; // ผลลัพธ์: หนังสือเล่มที่ 2

// ตรวจสอบว่า Stack ว่างหรือไม่
if ($stack->isEmpty()) {
    echo "Stack ว่างเปล่า";
} else {
    echo "ยังมีข้อมูลอยู่ใน Stack";
}
```

Use code with caution.



4. Queue (คิว) ในภาษา PHP

แบบพื้นฐาน (FIFO - First In, First Out เข้าก่อนออกก่อน) โดยใช้ Class ในการจัดการโครงสร้างข้อมูล

php

```
<?php
class SimpleQueue {
    private array $queue = [];

    // เพิ่มข้อมูลเข้าคิว (Enqueue)
    public function enqueue($item): void {
        array_push($this->queue, $item);
    }

    // นำข้อมูลออกจากคิว (Dequeue)
    public function dequeue() {
        if ($this->isEmpty()) {
            return null;
        }
        return array_shift($this->queue);
    }

    // ดูข้อมูลตัวแรกสุดในคิวโดยไม่ลบออก (Peek)
    public function peek() {
        if ($this->isEmpty()) {
            return null;
        }
        return $this->queue[0];
    }

    // ตรวจสอบว่าคิวว่างหรือไม่
    public function isEmpty(): bool {
        return empty($this->queue);
    }

    // นับจำนวนข้อมูลทั้งหมดในคิว
    public function size(): int {
        return count($this->queue);
    }

    // แสดงข้อมูลทั้งหมดในคิว
    public function getAll(): array {
        return $this->queue;
    }
}
```

```
// --- วิธีการใช้งาน (Usage) ---
$q = new SimpleQueue();

// เพิ่มข้อมูลเข้าคิว
$q->enqueue("ลูกค้ารายที่ 1");
$q->enqueue("ลูกค้ารายที่ 2");
$q->enqueue("ลูกค้ารายที่ 3");

echo "คิวปัจจุบัน: " . implode(", ", $q->getAll()) . "\n";
// ผลลัพธ์: ลูกค้ารายที่ 1, ลูกค้ารายที่ 2, ลูกค้ารายที่ 3

// ดึงข้อมูลออกจากคิว (ตัวแรกที่เข้าไปจะถูกดึงออกก่อน)
echo "กำลังให้บริการ: " . $q->dequeue() . "\n";
// ผลลัพธ์: ลูกค้ารายที่ 1

echo "คิวที่เหลือ: " . implode(", ", $q->getAll()) . "\n";
// ผลลัพธ์: ลูกค้ารายที่ 2, ลูกค้ารายที่ 3
```

Use code with caution.



2. Algorithms (อัลกอริทึม / ขั้นตอนวิธี)

- **ความหมาย:** คือลำดับขั้นตอนหรือกระบวนการทำงานแบบที่ละขั้นเพื่อแก้ปัญหาใดปัญหาหนึ่งให้สำเร็จอย่างถูกต้อง เช่น การค้นหาข้อมูล การเรียงลำดับตัวเลข หรือการคำนวณเส้นทาง
- **การวัดประสิทธิภาพ:** ประเมินจาก **Time Complexity** (ใช้เวลาประมวลผลเร็วแค่ไหน) และ **Space Complexity** (ใช้พื้นที่หน่วยความจำเปลืองแค่ไหน)

ตัวอย่างการเขียน **Algorithms (อัลกอริทึม)** โดยแบ่งออกเป็น 2 ประเภทหลักคือ การค้นหาข้อมูล (Searching) และ การเรียงลำดับข้อมูล (Sorting) ครับ

Linear Search Algorithm (การค้นหาแบบเส้นตรง)

เป็นอัลกอริทึมการค้นหที่ง่ายที่สุด โดยจะวิ่งไล่ตรวจเช็คข้อมูลใน Array ทีละตัวตั้งแต่ตัวแรกไปจนถึงตัวสุดท้ายจนกว่าจะเจอตัวที่ต้องการ

php

```
<?php
function linearSearch($arr, $target) {
    // วนลูปเช็คข้อมูลที่ละช่อง
    for ($i = 0; $i < count($arr); $i++) {
        if ($arr[$i] === $target) {
            return $i; // เจอแล้ว! ส่งกลับตำแหน่ง (Index) ที่เจอ
        }
    }
    return -1; // หาไม่เจอ ส่งกลับ -1
}

// --- วิธีใช้งาน ---
$data = [];
$target = 42;

$result = linearSearch($data, $target);

if ($result != -1) {
    echo "เจอข้อมูล {$target} อยู่ในตำแหน่งที่: " . $result;
} else {
    echo "ไม่พบข้อมูลที่ต้องการค้นหา";
}
// ผลลัพธ์: เจอข้อมูล 42 อยู่ในตำแหน่งที่: 3
?>
```

Use code with caution.



Bubble Sort Algorithm (การเรียงลำดับแบบฟองสบู่)

เป็นอัลกอริทึมการเรียงลำดับพื้นฐาน โดยมันจะเปรียบเทียบข้อมูลที่อยู่ติดกันคู่กันไปเรื่อยๆ หากตัวหน้ามากกว่าตัวหลังก็จับสลับที่กัน (Swap) ข้อมูลตัวที่ใหญ่ที่สุดจะค่อยๆ ขยับไปอยู่ท้ายสุดเหมือนฟองสบู่ที่ลอยขึ้นมา

php

```
<?php
function bubbleSort($arr) {
    $n = count($arr);

    // วนลูปตามจำนวนสมาชิก
    for ($i = 0; $i < $n; $i++) {
        // วนลูปเปรียบเทียบทีละคู่
        for ($j = 0; $j < $n - $i - 1; $j++) {
            // ถ้าตัวหน้ามากกว่าตัวหลัง ให้สลับที่กัน
            if ($arr[$j] > $arr[$j + 1]) {
                $temp = $arr[$j];
                $arr[$j] = $arr[$j + 1];
                $arr[$j + 1] = $temp;
            }
        }
    }
    return $arr;
}

// --- วิธีใช้งาน ---
$unsortedData =;
echo "ก่อนเรียงลำดับ: " . implode(", ", $unsortedData) . "\n";

$sortedData = bubbleSort($unsortedData);
echo "หลังเรียงลำดับ: " . implode(", ", $sortedData);
// ผลลัพธ์: หลังเรียงลำดับ: 11, 12, 22, 25, 34, 64, 90
?>
```

Use code with caution.



Database (ฐานข้อมูล)



Database หรือ ฐานข้อมูล คือ ระบบจัดเก็บข้อมูลทางอิเล็กทรอนิกส์ที่รวบรวมข้อมูลซึ่งมีความสัมพันธ์กันไว้ด้วยกันอย่างเป็นระบบ โดยมีซอฟต์แวร์ที่เรียกว่า ระบบจัดการฐานข้อมูล (DBMS: Database Management System) ทำหน้าที่เป็นตัวกลางในการควบคุม จัดเก็บ ค้นหา แก้ไข และดูแลรักษาความปลอดภัยของข้อมูล

หน้าที่และความสำคัญ

จัดการข้อมูลเป็นระบบ: ช่วยลดความซ้ำซ้อนของข้อมูล และประหยัดพื้นที่จัดเก็บ

ค้นหาและเรียกใช้รวดเร็ว: รองรับการดึงข้อมูลปริมาณมากได้ทันทีตลอด 24 ชั่วโมง

รองรับผู้ใช้งานพร้อมกัน: อนุญาตให้หลายคนหรือหลายแผนกเข้ามาใช้งานข้อมูลชุดเดียวกันได้พร้อมกัน

รักษาความปลอดภัย: ควบคุมสิทธิ์การเข้าถึง และมีการเข้ารหัสข้อมูลเพื่อป้องกันการรั่วไหล

ประเภทหลักของ Database

Relational Database (ฐานข้อมูลเชิงสัมพันธ์): เก็บข้อมูลในรูปแบบตาราง (แถวและคอลัมน์) เชื่อมโยงกันด้วยความสัมพันธ์ ใช้ภาษา SQL ในการจัดการ เช่น MySQL, PostgreSQL, Oracle

NoSQL Database: ออกแบบมาเพื่อรองรับข้อมูลขนาดใหญ่ หรือข้อมูลที่ไม่มีโครงสร้างแน่นอน (Unstructured Data) มีความยืดหยุ่นสูง เช่น MongoDB

Cloud Database: ฐานข้อมูลที่ทำงานและจัดเก็บอยู่บนระบบคลาวด์ เพิ่มความสะดวกในการขยายขนาดระบบ เช่น Amazon RDS, Google Cloud SQL

In-Memory Database: จัดเก็บข้อมูลไว้ในหน่วยความจำ RAM เพื่อความเร็วสูงสุดในการประมวลผล

ตัวอย่างคำสั่งพื้นฐาน SQL

1. DDL (Data Definition Language) - คำสั่งจัดการโครงสร้างฐานข้อมูล

สร้างฐานข้อมูล (Create Database)

```
CREATE DATABASE MyDatabase
```

สร้างตาราง (Create Table)

```
CREATE TABLE Employees (  
    EmployeeID INT IDENTITY(1,1) PRIMARY KEY,  
    FirstName NVARCHAR(50) NOT NULL,  
    LastName NVARCHAR(50) NOT NULL,  
    Salary DECIMAL(18, 2) DEFAULT 0.00,  
    BrithDate DATE  
);
```

ลบตาราง (Drop Table)

```
DROP TABLE Employees;
```

2. DML (Data Manipulation Language) – คำสั่งจัดการข้อมูลในตาราง

เพิ่มข้อมูล (INSERT)

```
INSERT INTO Employees (FirstName, LastName, Salary, BirthDate) VALUES ('สมชาย',  
'ใจดี', 25000.00, '2015-01-15');
```

เรียกดูข้อมูล (SELECT)

ดึงข้อมูลทั้งหมด

```
SELECT * FROM Employees;
```

ดึงข้อมูลแบบมีเงื่อนไขและเรียงลำดับ

```
SELECT FirstName, LastName, Salary FROM Employees WHERE Salary >= 20000  
ORDER BY Salary DESC;
```

แก้ไขข้อมูล (UPDATE)

```
UPDATE Employees SET Salary = 28000.00 WHERE EmployeeID = 1;
```

ลบข้อมูล (DELETE)

```
DELETE FROM Employees WHERE EmployeeID = 1;
```

3. คำสั่งพิเศษที่พบบ่อยใน SQL Server

การดึงข้อมูลจำนวนจำกัด (TOP)

```
SELECT TOP 5 FirstName, LastName FROM Employees ORDER BY BirthDate ASC;
```

การรวมข้อมูลจาก 2 ตาราง (INNER JOIN)

```
SELECT E.FirstName, B.DepartmentName FROM Employees A INNER JOIN  
Departments B ON A.DepartmentID = B.DepartmentID
```