บทที่ 2
หลักการและทฤษฎี

2.1 ASP คือ อะไร

ASP ย่อมาจาก Active Server Page ซึ่งคิดค้นโดย บริษัทไมโครซอฟต์ ASP เป็นโปรแกรมคอมพิวเตอร์ชนิดที่เป็น "Server side scripting" ซึ่งหมายถึงภาษาทางโปรแกรมที่ทำงานในฝั่งของเครื่องคอมพิวเตอร์ที่ทำหน้าที่เป็น Web Server ที่ให้บริการเอกสารหรือสื่อต่างๆ ในอินเตอร์เน็ต หรือ อินทราเน็ต

2.2 หลักการทำงานของ ASP

ASP จะทำงานบนเซิร์ฟเวอร์ และทำงานร่วมกับโปรแกรม Web Server จะทำหน้าที่ประมวลข้อมูล ที่ได้จากผู้เข้ามาเยี่ยมชม และแสดงผลออกมาทาง Web browser เริ่มจากผู้ใช้ ASP สร้างไฟล์ที่มีนามสกุลเป็น .ASP ขึ้นมาจากนั้น นำไฟล์นั้นไปไว้ในเครื่องคอมพิวเตอร์ ที่ทำหน้าที่เป็น Web Server ที่ติดตั้งโปรแกรม ASP ไว้ และเชื่อมต่ออยู่กับเครือข่ายอินเตอร์เน็ต จากนั้นเมื่อมีผู้ใช้รายใดเรียกใช้ไฟล์นั้นผ่านโปรแกรมบราวเซอร์ (IE ,Netscape...) โปรแกรม ASP ใน Web Server จะเรียกไฟล์นั้นขึ้นมาอ่านแล้วทำตามคำสั่งต่างๆ ที่ผู้สร้างไฟล์นั้นได้กำหนดขึ้น จึงส่งผลที่ได้กลับไปให้ผู้เรียกใช้โดยแสดงผลที่โปรแกรมบราวเซอร์ของผู้เรียก ซึ่งขั้นตอนข้างต้นเป็นหลักการทำงานโดยทั่วไปของ ASP [3]

2.3 ความสามารถและประโยชน์ของ ASP

1. ASP ทำให้เว็บแบบไดนามิก(Dynamic) [6] นั้น คือรูปแบบที่แสดงผลออกมานั้นสามารถเปลี่ยนแปลงได้ตามข้อมูลที่ ASP ได้รับ เช่น ตัวอย่างจากการ Search ข้อมูลในเว็บไซต์ ผลลัพธ์ที่ได้จะเปลี่ยนไปตามที่เรา Search

2. เพิ่มความเร็วในการดูเว็บ เนื่องจากการดูเว็บนั้น เรามักสูญเสียเวลาส่วนใหญ่กับการรอข้อมูลที่มาจากอินเตอร์เน็ต ยิ่งข้อมูลมากขึ้นยิ่งรอนาน ซึ่ง ASP สามารถช่วยในจุดนี้ได้ กล่าวคือ ASP จะทำการคำนวนต่างๆ จะเสร็จและส่งเฉพาะผลลัพธ์ที่เราต้องการเท่านั้น ทำให้ปริมาณการส่งข้อมูลน้อยลง เราก็จะเสียเวลารอ ข้อมูลน้อยลงและสามารถดูเว็บได้เร็วขึ้น

3. เพิ่มความปลอดภัยให้กับระบบ ในการเขียนโปรแกรมต่างๆ บางครั้งเราต้องอ้างถึงไดเร็กทอรีที่เก็บฐานข้อมูล อย่างเช่น เว็บไซต์ Yahoo เป็นต้น ซึ่งการใช้ ASP ไดเร็กทอรี่ต่างๆ จะไม่ถูกแสดงที่ฝั่งผู้ดูเว็บ จะแสดงเฉพาะผลลัพธ์ที่เอามาจากฐานข้อมูลเท่านั้น ทำให้ผู้ดูแลเว็บไม่สามารถรู้ถึงโครงสร้างของเว็บเราได้ง่าย และ ป้องกันผู้ไม่หวังดีมาเจาะระบบของเราด้วย

4. ลดปัญหาความสามารถของเครื่องที่ใช้ดูเว็บ เนื่องจาก ASP จะส่งเฉพาะผลลัพธ์สุดท้ายมาแสดงผลเท่านั้น ดังนั้น ไม่ว่าเครื่องของคุณจะทันสมัยหรือล้าสมัยเพียงใด ก็ไม่ทำให้เวลาที่ใช้เปิดดูเว็บแตกต่างกันมาก เพราะว่าการประมวลผลทั้งหมดเสร็จสิ้นที่ฝั่ง Server แล้ว

2.4 พื้นฐานการเขียน ASP

ประกอบด้วยเรื่องเกี่ยวกับตัวแปร ค่าคงที่ อาร์เรย์ Procedures การรับส่งข้อมูลระหว่างเบราวเซอร์กับเซอร์เวอร์ การทำงานกับ ไฟล์ และฐานข้อมูล ตัวแปรระดับ session ตัวแปรระดับ application การใช้ cookie การใช้ไฟล์ global.asa การใช้ฟังก์ชั่นต่างๆ ที่มีอยู่แล้ว รวมทั้งการควบคุมทิศทางการทำงานของ script

พื้นฐาน VBscript

การเขียน ASP จำเป็นต้องอาศัยภาษา VBscript เข้าช่วยโดยนำมาใช้ในการสร้างหรือประกาศตัวแปร และใช้ควบคุมทิศทางการทำงานของ script เช่น if...then, for...next, while...wend เป็นต้น

ในหัวข้อนี้เราจะได้รู้จักการใช้งานตัวแปลในแบบต่างๆ รวมทั้งการใช้ procedures ซึ่งเป็นการรวมโค้ดที่ใช้งานบ่อยๆ เป็นกลุ่มก้อนเพื่อให้สามารถเรียกใช้งานได้บ่อยๆ เท่าที่ต้องการ

ชนิดของตัวแปลและการใช้งาน

ตัวแปลเป็นสิ่งจำเป็นที่จะต้องมีในการเขียนโปรแกรมทุกภาษา โดยถูกทำมาใช้เก็บค่าต่างๆ เพื่อนำไปคำนวณหาผลลัพธ์ รอแสดงผล หรือรอรับสิ่งที่เราต้องการ เช่น รหัสผ่าน มาจากผู้ใช้

การประกาศตัวแปลขึ้นมาใช้งาน VBScript นั้นไม่จำเป็นต้องกำหนดชนิดของตัวแปลให้วุ่นวาย (ทุกตัวจะมีชนิดเป็น variant) ถึงแม้ว่าในความเป็นจริงแล้วตัวแปลแต่ละตัวจะถูกพิจาณณาได้ด้วยตัวแปลภาษา VBScript ว่าควรเก็บข้อมูลชนิดใด และใช้เนื้อที่ในหน่วยความจำเท่าไรในการเก็บ ซึ่งก็ขึ้นอยู่กับค่าที่เรากำหนดให้ในตัวแปลนั้นๆ

การที่ตัวแปลมีชนิดเป็น variant ทำให้ตัวแปลหนึ่งๆ สามารถเก็บค่าชนิดใดๆ ก็ได้

ตัวแปลภาษา vbscript จะเลือกชนิดตัวแปลที่เหมาะสมเพื่อเก็บข้อมูลที่เรากำหนดให้โดยแบ่งข้อมูลออกเป็นชนิดย่อยๆ ดังนี้

Sub Type	คำอธิบาย	ค่าที่เป็นไปได้
Boolean	เก็บค่าทางตรรกะ	True หรือ False
Single	เก็บเลขทศนิยม	ค่าลบตั้งแต่ -3.402823E38 ถึง -1.401298E-45 ค่าบวกตั้งแต่ 1.401298E-45 ถึง 3.402823
Double	เก็บเลขทศนิยม	ค่าลบตั้งแต่ -1.79769313486232E308 ถึง -4.94065645841247E-324 ถึง ค่าบวกตั้งแต่ 4.940655645841247E-324ถึง 1.79769313486232E308
Byte	เก็บเลขจำนวนเต็มขนาดเล็ก	0 ถึง 255
Integer	เก็บเลขจำนวนเต็มขนาด 16 บิต	-32.768 ถึง 32.767
Long	เก็บเลขจำนวนเต็มขนาด 32 บิต	-2,147,483,648 ถึง 2,147,483,647
String	เก็บข้อความ	ข้อความที่มีความยาวได้มากที่สุดประมาณ 2 ล้านตัวอักษร
Date	เก็บวันที่	วันที่ตั้งแต่ 1 มกราคม ค.ศ. 100 ถึง 31 ธันวาคม ค.ศ. 9999
Time	เก็บเวลา	เวลาในช่วง 1 มกราคม ค.ศ. 100 ถึง 31 ธันวาคม ค.ศ. 9999
Empty	เป็นค่าที่กำหนดให้กับตัวแปลทันทีที่ประกาศตัวแปลด้วย dim	ตัวแปลยังไม่ถูกกำหนดค่า โดยจะมีค่าเป็น 0 ถ้าใช้เก็บตัวเลข และเป็น "" ถ้าใช้เก็บข้อความ
Null	เป็นค่าจากฐานข้อมูล เมื่อยังไม่ได้กำหนดค่าใดๆ ให้กับ ฟิลด์นั้นๆ	ไม่เก็บค่าใดๆ ที่นำไปใช้ได้
Error	เป็นตัวแปลที่เก็บข้อมูลเพื่อช่วยจัดการข้อผิดพลาดที่เกิดขึ้น	หมายเลขข้อผิดพลาด
Object	เป็นออบเจ็ค	ออบเจ็ค

ตารางที่2.1 ชนิดตัวแปล

วิธีประกาศตัวแปลขึ้นมาใช้งาน ทำได้โดยตั้งชื่อตัวแปลตามหลังคำว่า Dim ดังนี้

Dim *ชื่อตัวแปล, ชื่อตัวแปล, ...*

โดยชื่อตัวแปลจะต้องขึ้นต้นด้วยตัวอักษร ไม่ซ้ำกับคำสงวน (Reserve Word) ซึ่งเป็นคำที่ VBscript นำมาใช้แล้ว เช่น Dim และเราควรระจะตั้งชื่อ ให้สื่อความหมายที่สามารถเดาได้ว่าต้องการนำกตัวแปลนั้นไปใช้เก็บค่าใด ตัวอย่างเช่น

Dim Count, Username, CurrentDate

การประกาศค่าคงที่

ค่าคงที่คือ ค่าที่ไม่สามารถเปลี่ยนแปลงได้ โดยเคยมีค่าเป็นเท่าไรก็จะเป็นเท่านั้นตลิดไปในสคริปต์หนึ่งๆ สำหรับวิธีที่ใช้กำหนดค่าคงที่ก็คือ

Const ชื่อค่าคงที่ = ค่าคงที่ที่ต้องการกำหนดให้

วิธีใช้อาร์เรย์

เมื่อต้องการเก็บข้อมูลชนิดเดียวกันที่เกี่ยวข้องเป็นชุดๆ เช่น คะแนนของตัวเลือกแต่ละตัวที่ผู้ใช้โหวตในโพล และเราต้องการทำงานกับข้อมูลทั้งชุดนั้นเหมือนๆ กันในภายหลัง การเก็บข้อมูลดังกล่าวไว้ในอาร์เรย์จะเหมาะสมมาก เนื่องจากเราสามารถเก็บข้อมูลทั้งหมดได้ภายในตัวแปลตัวเดียว (โดยให้เรามองว่าอาร์เรย์ ถูกแบ่งออกเป็นช่องๆ เพื่อเก็บข้อมูลแต่ละตัว) และเราสามารถวนลูป for...next เพื่อทำสิ่งที่ต้องการกับข้อมูลทั้งชุดได้ง่ายๆ โดยใช้ประโยชน์จาก index ของ อาร์เรย์

วิธีประกาศอาเรย์ ทำได้โดยตั้งชื่ออาเรย์ไว้หลังคำว่า Dim แล้วกำหนดจำนวนข้อมูลที่ต้องการไปไว้ภายในวงเล็บซึ่งอยู่ต่อจากชื่ออาเรย์

Dim = *ชื่ออาร์เรย์(อินเด็กซ์)*

โดยถ้าเราต้องการเก็บข้อมูลในอาร์เรย์เป็นจำนวน 4 ตัว ตัวเลขที่เราต้องกำหนดในวงเล็บตอนที่ประกาศตัวแปลคือ 3 เนื่องจากช่องสำหรับเก็บข้อมูลในอาร์เรย์จะเริ่มต้นที่ 0

การสร้าง Procedure ขึ้นมาใช้งานเอง

เราจะสร้าง Procedure ขึ้นมาเวลาที่จำเป็นต้องทำงานอะไรซ้ำๆ กัน ซึ่งในเหตุการณ์อย่างนี้ถ้ามองการเขียน code ขึ้นมาจัดการงานเดิมๆ ใหม่ทุกครั้งก็จะเป็นการเสียเวลา และทำให้สคริปต์

ของเรามีขนาดใหญ่เกินความจำเป็น นอกจากนี้ถ้าเราต้องกลับมาแก้ไขการทำงานของส่วนนี้ในอนาคตเราก็ต้องแก้ไขในทุกๆ ที่ที่ได้เขียน code ไว้ซึ่งทำให้เสียเวลาเป็นอย่างมาก

Procedure มีอยู่ 2 แบบคือ Sub procedure และ function ซึ่งมีจุดเด่นแตกต่างกัน ดังนั้นเราจึงต้องเลือกให้เหมาะกับงานโดย Sub procedure คือการรวม code คำสั่งจำนวนหนึ่งเข้าด้วยกัน เวลาใช้งานให้มองว่าเป็นการนำ code ที่อยู่ใน Sub procedure ไปแปะแทนตำแหน่งที่มีการเรียกใช้ sub procedure โดยเราสามารถเลือกได้ว่าจะส่งค่าเข้าไปใน sub procedure หรือไม่ก็ได้ซึ่งถ้าต้องการส่งค่าให้ sub procedure ก็ให้ทำดังนี้

```
Sub ชื่อ sub procedure(ค่าที่ส่งมาให้)
  ...
End Sub
```

แต่ถ้าไม่ต้องการส่งค่าใดๆ เข้าไปก็ให้ตั้งชื่อ sub procedure หลังคำว่า sub โดยไม่ต้องต่อในตำแหน่งที่ต้องการเรียกใช้ sub procedure ให้เรียกใช้ดังตัวอย่างข้างล่าง ถ้าไม่มีการส่งค่าก็ไม่ต้องต่อท้ายชื่อ sub procedure ด้วยวงเล็บเช่น

```
Call ชื่อ Sub procedure(ที่ส่งมาให้)
```

Function จะต่างจาก sub procedure ตรงที่สามารถส่งกลับค่าไปยังตำแหน่งที่มีการเรียกใช้งานได้ ซึ่งคล้ายกับฟังก์ชั่น VBscript เตรียมไว้ใช้ใช้อยู่แล้ว

การสร้างฟังก์ชั่นขึ้นมาใช้ก็คล้ายกับ Sub procedure โดยที่สามารถเลือกได้ว่าต้องการส่งหรือไม่ส่งค่าเข้าไปใน function ได้เช่นกัน แต่มีส่วนที่เราต้องรู้ไว้คือ การส่งค่ากลับ ซึ่งทำได้ด้วยการกำหนดค่าที่ต้องการส่งค่ากลับ ซึ่งทำได้ด้วยการกำหนดค่าที่ต้องการส่งกลับ ให้กับตัวแปลชื่อเดียวกับฟังก์ชั่นที่บรรทัดสุดท้ายของโค้ดที่อยู่ภายใน function ดังนี้

```
Function ชื่อฟังก์ชั่น (ค่าที่ส่งไปให้)
...
ชื่อฟังก์ชั่น = ค่าที่ต้องการส่งกลับ
End function
```

เมื่อมีการส่งค่ากลับ เราก็ต้องหาตัวแปลมารับค่านั้นๆ ณ จุดที่มีการเรียกใช้ function ด้วย ดังนี้

```
ตัวแปล = ชื่อ function(ค่าที่ส่งไปให้)
```

วิธีรับส่งข้อมูลระหว่างบราวเซอร์ และเซิร์ฟเวอร์

การพัฒนาเว็บแอพพลิเคชั่นขึ้นมา จำเป็นต้องมีการส่งข้อมูลกันระหว่างผู้ใช้ และ Script ของเรา โดยผู้ใช้จะรับและส่งข้อมูลผ่านโปรแกรมประเภท บราวเซอร์ เช่น Internet Exploror หรือ Netscape Navigator ในขณะที่ Script ของเราจะรับและส่งข้อมูลผ่าน เว็บเซอร์เวอร์

ASP ได้จัดเตรียมวิธีง่ายๆ ที่ช่วยให้เราสามารถรับส่งข้อมูลระหว่าง บราวเซอร์ และเซิร์ฟเวอร์ โดยผ่าน object พื้นฐาน 2 ออบเจ็กต์ คือ object Request ซึ่งเป็นข้อมูลประเภท collection เพื่อรับข้อมูลที่ผู้ใช้กรอกมาให้ผ่านฟอร์ม เช่นถ้าเราต้องการสร้างฟอร์มขึ้นมารับชื่อผู้ใช้ด้วยโค้ด HTML ต่อไปนี้

```
<form method=post action=form.asp>
<input type=text name=username>
</form>
```

ใน script form.asp เราก็สามารถรับค่าที่ผู้ใช้กรอกมาในช่องรับข้อมูลประเภท text ซึ่งเราได้ตั้งชื่อในฟอร์ม ไว้ว่า username ได้ โดยนำชื่อ username มาเป็นคีย์เพื่ออ้างอิงค่าที่เก็บใน collection ดังนี้

```
ตัวแปล = Request.Form("username")
```

ข้อมูลที่ส่งต่อท้ายมากับชื่อ URL

นอกจาก script จะได้รับข้อมูลผ่าน Form แล้ว script ยังสามารถรับข้อมูลที่ต่อท้ายมากับชื่อ URL ได้อีกด้วย ซึ่งการส่งข้อมูลในรูปแบบนี้จะเกิดขึ้นเมื่อเราได้กำหนด attribute method ของ <form> เป็น get หรือ มีการส่งข้อมูลต่อท้ายชื่อ URL จาก Script หนึ่ง ไปให้อีก Script หนึ่ง

วิธีรับข้อมูลที่ส่งมาในรูปแบบนี้ทำให้ได้เหมือนการรับข้อมูลจากฟอร์มในแบบแรก แต่เปลี่ยนมาใช้ property QueryString ของ object Request ซึ่งเป็นข้อมูลประเภท Collection เช่นเดียวกับ property Form แทนเช่น

```
ตัวแปล = Request.QueryString("ชื่อ object")
```

Object Response

เป็น Object ที่ใช้ส่งข้อมูลจาก Script ไปยังบราวเซอร์ เพื่อแสดงผลให้ผู้ใช้เห็นซึ่งทำได้ง่าย โดยใช้ method Write ของ Object Response

เราสามารถส่งทั้งข้อมูลที่เรากำหนดในขณะที่เรียกใช้ method write หรือค่าในตัวแปลไปยังเบราวเซอรก็ได้ ดังตัวอย่าง

```
Response.Write "ข้อมูล"
Response.Witte ตัวแปล
```

การใช้ ไฟล์เก็บข้อมูล

ข้อมูลที่เราเก็บไว้ในตัวแปลจะอยู่เมื่อ Script ทำงานอยู่ แต่ถ้าเมื่อ Script ทำงานเสร็จ ข้อมูลเหล่านั้นก็จะหายไป เพราะข้อมูลเหล่านั้นถูกเก็บไว้ในหน่วยความจำ

ถ้าต้องการเก็บข้อมูลไว้ใช้อย่างถาวร ก็สามารถใช้ ไฟล์เก็บข้อมูลไฟล์ ซึ่งเป็นการบันทึกข้อมูลลงฮาร์ดิสก์ของเซิร์ฟเวอร์

การสร้างไฟล์ object FileSystemObject

ASP ได้จัดเตรียม object FileSystemObject ไว้เพื่อให้เราสามารถเข้าถึงระบบไฟล์ ของเว็บเซิร์ฟเวอร์ โดยทำให้เราสามารถจัดการไฟล์ โฟลเดอร์ ไดร์ว รวมทั้งทราบรายละเอียดของไฟล์ โฟล์เดอร์ หรือไดร์วที่ต้องการ

การที่จะเข้าถึงระบบไฟล์ของ เว็บเซิร์ฟเวอร์นั้น ต้องเริ่มที่การสร้างตัวแทนของ object FileSysTemObjcet ขึ้นมาก่อน โดยใช้ method CreateObjcet ของ Objcet Server จากนั้นก็กำหนดตัวแปลขึ้นมาอ้างถึงตัวแทนของ Object ดังกล่าวเพื่อให้สามารถเรียกใช้ method และ property ต่างๆ ของ object นี้ผ่านตัวแปลได้ดังนี้

```
Set ตัวแปล = Server.CreateObject("object ที่ต้องการใช้งาน")
```

หลังจากที่ใช้งาน object FileSystemObject แล้ว สิ่งที่เราควรทำคือ ยกเลิกการอ้างถึงตัวแทนของ object ผ่านตัวแปล เพื่อเป็นการประหยัดทรัยพยากรณ์ของระบบด้วยประโยค นี้

```
Set ตัวแปล = Nothing
```

จะอ่านหรือเขียนไฟล์ได้อย่างไร

เมื่อสร้าง object FileSystemObject ขึ้นมาแล้ว ถ้าเราต้องการจะทำงานกับไฟล์ เราก็ต้องเปิดไฟล์นั้นขึ้นมาด้วย method OpenTextFile ของ object FileSystemObject โดยกำหนดชื่อไฟล์ที่ต้องการให้กับ method ดังกล่าว ซึ่งการใช้ method ของ Object TextStream อ่านหรือเขียนไฟล์อีกทีหนึ่ง

การเปิดไฟล์ขึ้นมาจะเป็นการเปิดเพื่ออ่าน หรือเขียนนั้นขึ้นอยู่กับโหมดที่เรากำหนดให้ method OpenTextFile โดยโหมดที่ใช้จัดการไฟล์ ที่มีค่าเป็นไปได้มีดังนี้

- มีค่าเป็น 1 เมื่อต้องการอ่านไฟล์ (ถ้าไม่กำหนดจะเป็น mode นี้)
- มีค่าเป็น 2 เมื่อต้องการเขียนไฟล์ โดยจะเขียนทับข้อมูลเดิม

- มีค่าเป็น 8 เมื่อต้องการเขียนไฟล์ต่อจากข้อมูลเดิมที่มีอยู่ในไฟล์

หลังจากใช้งานไฟล์เรียบร้อยแล้วเราก็ต้องหิดไฟล์ที่เปิดขึ้นมาด้วย method Close ของ Object TextStream ด้วย โดยมีวิธีใช้เปิดและปิดมีรูปแบบดังนี้

```
Set ตัวแปล1 = Server.CreateObject("Scripting.FileSystemObject")
Set ตัวแปล2 = ตัวแปล1.OpenTextFile("พาธของไฟล์ที่ต้องการ",โหมดที่ใช้จัดการไฟล์)
-
ตัวแปล2.Close
```

Objcet TextStream จะมี method อยู่ 2 method ที่เรามักใช้งานกันอยู่บ่อยๆ นั่นก็คือ method ReadLine อ่านข้อมูลมาทีละบรรทัด และ method WriteLine เขียนข้อมูลทีละบรรทัด

ตัวอย่างการใช้งาน

```
Set objFS=Server.CreateObject("Scriptiong.FileSystemObject")
Set objTF=ObjFS.OpenTextFile("/test.txt")
FileData=objTF.ReadLine
objTF.Close

Set objTF=ObjFS.OpenTextFile("/test.txt",2)
objTF.WriteLine FileData
objTF.Close

SetObjTF=Nothing
SetObjFS=Nothing
```

วิธีเล่นกับไฟล์ หรือ ไดเร็คทอรี่ทั้งหมด

Object Folder ซึ่งมี property 2 ตัวที่น่าสนใจ เมื่อเราต้องการทำงานกับไฟล์หรือ directory ย่อยทั้งหมดภายใน directory ใดๆ คือ Files และ Sub Folder

เริ่มแรกเราต้องสร้างตัวแทน Object Folder ขึ้นมาจาก method GetFolder ของ object FileSystemObject เสียก่อน

ตัวแปล = FileSystemObject.GetFolder("ชื่อ Folder")

หลังจากใช้งาน object Folder ผ่านตัวแปลได้แล้ว เมื่อเราต้องการทำงานกับไฟล์ทั้งหมดใน Directory นี้ เราก็สามารถวน loop For... each เพื่อทำงานกับ object ที่อยู่ใน property files ได้โดย ถ้าตัวแปลที่เรานำมาใช้อ้างถึง object Folder มีชื่อว่า objFolder ก็จะเป็นดังตัวอย่างข้างล่าย

For Each objFile in objFolder.Files

...

Next

แต่เราต้องการจะทำงานกับ directory ย่อยทั้งหมดแทนไฟล์ทั้งหมด ก็สามารถทำได้ด้วยวิธีเดิม แต่เปลี่ยนจากการวน loop เพื่อทำงานกับ object Folder ซึ่งเก็บอยู่ใน property Subfolders แทน

For Each objSubFolder in objFolder.SubFolders

...

Next

เปลี่ยนไปเก็บข้อมูลลงฐานข้อมูล

การเก็บข้อมูลไว้ใช้ถาวรไม่ได้มีแต่การเก็บลงไฟล์เท่านั้น ยังมีทางเลือกอีกอย่างหนึ่งคือการเก็บข้อมูลลงฐานข้อมูลซึ่งดี และเหมาะกับข้อมูลใดๆ ก็ตามที่ถูกแบ่งย่อยลงไปอีก และเราก็มีความจำเป็นที่จะต้องนำข้อมูลย่อยเหล่านั้นมาใช้ อย่างเช่น การเก็บข้อมูลของลูกค้าแต่ละคนสามารถ ถูกแบ่งย่อยออกเป็น ชื่อ สกุล อายุ เพศ ที่อยู่

การใช้งานฐานข้อมูลจาก script ASP
มีขั้นตอนดังนี้
- สร้างการเชื่อมต่อไปยังฐานข้อมูล
- ดึงข้อมูลจากฐานข้อมูลผ่านการเชื่อมต่อในขั้นตอนแรก
- ใช้งานข้อมูลที่ดึงมาจากขั้นที่ 2
- ปิดการเชื่อมต่อที่สร้างขึ้น

ขั้นที่ 1 สร้างการเชื่อมต่อไปยังฐานข้อมูล

การเชื่อมต่อไปยังฐานข้อมูลจะเริ่มที่การสร้างตัวแทนของ object connection ขึ้นมา แล้วกำหนดรายละเอียดของการเชื่อมต่อให้กับ method Open ของ object Connection ว่าต้องการเชื่อมต่อกับฐานข้อมูลประเภทใด ฐานข้อมูลนั้นเก็บอยู่ที่ใด และถ้าหากว่าเราได้ตั้งรหัสผ่านให้กับฐานข้อมูล เราก็ต้องระบุรหัสผ่านดังกล่าวนั้นด้วยเพื่อให้สามารถเชื่อมต่อไปยังฐานข้อมูลได้

ตัวอย่างจะเชื่อมต่อฐานข้อมูล Access ด้วย OLEDB

```
Set Conn=Server.CreateObject("ADODB.Connection")
Conn.Open "Provider=Microsoft.Jet.OLEDB.4.0; Data Source=C:\mydb.mdb"
```

ขั้นที่ 2 ดึงข้อมูลมาจากฐานข้อมูล

หลังจากที่ได้สร้างทางติดต่อกับฐานข้อมูลแล้ว ตอนนี้เราก็พร้อมจะดึงข้อมูลที่ต้องการจากฐานข้อมูลขึ้นมา โดยใช้ภาษา SQL เช่น

```
Select ฟิลด์ From ตาราง Where เงื่อนไข
Select Name From MyUser Where Name = "Maneechote" or Name = "Paang"
Select Name From MyUser Whare Name Like "M%"
Select Age From MyUser Order by Age Desc
```

การดึงข้อมูลจากฐานข้อมูลทำได้โดยการสร้างตัวแทนของ object Recordset ขึ้นมารับข้อมูล ที่ตรงตามที่เรากำหนดด้วยคำสั่ง SQL โดยวิธีที่ง่ายที่สุด คือการใช้ method Execute ของ object Connection ซึ่งจะทำให้ได้ตัวแทนของ object Recordset มาใช้งาน โดยเราจะต้องกำหนดคำสั่ง SQL ให้ method Execute ดังนี้

```
SQL = "Select * From MyUser"
Set rs = Conn.Execute(SQL)
```

การดึงข้อมูลแบบข้างต้นสามารถอ่านข้อมูลได้อย่างเดียว ไม่สามารถเพิ่มข้อมูลลงในฐานข้อมูล หรือเปลี่ยนแปลงแก้ไขข้อมูลได้ วิธีแก้คือการสร้างตัวแทน Recordset ขึ้นมาแล้วเรียกใช้ method Open ของ object Recordset เพื่อดึงข้อมูลสามารถกำหนดคุณสมบัติของ Recordset ได้ดังตัวอย่าง

```
Set rs = Server.CreateObject("ADODB.Recordset")
SQL = "Select * From MyUser"
Rs.Open SQL, Conn, 0, 3, 1
```

สำหรับค่าทั้ง 5 ที่กำหนดให้ method Open ของ Object Recordset ประกอบด้วย
Source คือแหล่งข้อมูลที่ต้องการ ซึ่งอาจเป็นคำสั่ง SQL หรือตารางในฐานข้อมูลก็ได้
ActiveConnection จากตัวอย่างคือ Conn ซึ่งเป็น object Connection ที่ได้ทำการเชื่อมต่อไว้แล้ว
CursorType เป็นการกำหนดประเภทของ Curser ว่าให้เข้าถึงแต่ละ Record ของ Recordset ในทิศทางใดได้บ้าง ค่าปกติคือ 0 ซึ่งเป็นการกำหนดให้เข้าถึงในทิศทางไปข้างหน้าเท่านั้น
LockType เป็นการกำหนดประเภทของการล็อค ค่าปกติคือ 1 หมายถึงเราสามารถอ่านข้อมูลได้อย่างเดียว ไม่สามารถเปลี่ยนแปลงแก้ไขได้

Options คือการบอกประเภท Source โดยเราไม่จำเป็นต้องกำหนดค่านี้ให้กับ method Open ก็ได้ แต่ถ้าเรากำหนดค่านี้ให้ method Open จะทำให้ได้ข้อมูลเร็วขึ้น เพราะระบบไม่ต้องตรวจสอบว่า Source ที่ กำหนดมาอยู่ในรูปแบบใดจากตัวอย่างคือ การกำหนดประเภทของ source เป็น 1 ซึ่งหมายถึงคำสั่ง SQL (ถ้าเราใช้ Source เป็นชื่อของตารางก็ต้องค่าเป็น 512 แทน)

ขั้นที่ 3 ใช้งานข้อมูลที่ดึงขึ้นมา

ข้อมูลที่ได้มาจากฐานข้อมูลนั้น สามารถนำมาใช้ได้โดยอาจอ่านขึ้นมาเพื่อแสดงผลอย่างเดียวเพื่อเพิ่มข้อมูลใหม่ๆ ลงไปเก็บในฐานข้อมูล เพื่อแก้ไขข้อมูลให้ทันสมัย หรือเพื่อลบข้อมูลที่ไม่ต้องการอกจากฐานข้อมูล ก็ได้

การนำข้อมูลใน Recordset ,k.=h

เมื่อได้ข้อมูลที่ต้องการมาอยู่ใน object Recordset แล้วเราก็จะนำข้อมูลดังกล่าวมาใช้โดยอ้างถึงผ่าน Collection "field" ของ object Recordset ในรูปแบบที่กะทัดรัด และเป็นที่นิยมมากที่สุดดังนี้

```
Rs("ชื่อฟิลด์")
```

การอ้างถึงข้อมูลในรูปแบบนี้จะหมายถึงข้อมูลของฟิลด์ที่ต้องการซึ่งอยู่ใน Record ปัจจุบัน หรือก็คือ Record ที่เคอร์เซอร์กำลังชี้อยู่นั่นเอง ซึ่งถ้าเราใช้งานข้อมูลใน Record ปัจจุบันเสร็จแล้ว และต้องการใช้ข้อมูลใน Record ถัดไป เราก็ต้องเลื่อนเคอร์เซอร์ให้เปลี่ยนไปชี้ Record ถัดไปด้วย method MoveNext แล้วนำข้อมูลของ Record ถัดไปมาใช้ด้วยวิธีเดิมคือ Rs("ชื่อฟิลด์")

Rs.MoveNext

นอกจากเราจะสามารถสั่งให้เคอร์เซอร์เลื่อนไปยัง Record ถัดไปแล้ว เราก็ยังสามารถเลื่อน Curser ไปยังไปยัง Record แรก, Record สุดท้าย หรือ Record ก่อนหน้า Record ปัจจุบันด้วย method MoveFirst, MoveLast และ MovePrevious ตามลำดับได้อีกด้วย

ตัวอย่างการใช้ประโยชน์จาก property EOF ในการวน loop แสดงค่าในฟิลด์ Name ทั้งหมดอยู่ใน Recordset เป็นดังนี้

```
While Not rs.EOF
        Response.Write rs("Name")
        Rs.MoveNext
Wend
```

การเพิ่มข้อมูลใหม่ลงฐานข้อมูล

เราจะใช้ เมธอด AddNew ของออบเจ็กต์ Recordset เพื่อสร้าง Record สำหรับเก็บข้อมูลเพิ่มในตาราง (เป็นการเพิ่มแถวขึ้นมาใหม่) จากนั้นจึงใช้ method Update ของ object เดียวกันเพื่อบันทึกการเปลี่ยนแปลงที่เราได้กระทำกับ object Recordset และฐานข้อมูลดังตัวอย่างต่อไปนี้

```
Rs.Addnew
        Rs("Name")="Maneechote"
        Rs("Age")=23
        Rs("Gender")=Female
Rs.Update
```

การแก้ไขข้อมูลที่เก็บอยู่ในฐานข้อมูล

วิธีแก้ไขเปลี่ยนแปลงข้อมูลทำได้ง่ายๆ โดยกำหนดค่าใหม่ให้ฟิลด์ที่ต้องการแล้วเรียก method Update เพื่อบันทึกการเปลี่ยนแปลงดังนี้

```
Rs("Age")=27
Rs.update
```

การลบข้อมูลในฐานข้อมูล

ด้วย method Delete จะทำให้เราสามารถลบข้อมูลที่ไม่ต้องการได้โดยถ้าเราเรียกใช้ method Delete จะหมายถึง การสั่งให้ลบ Record ปัจจุบันซึ่ง Curser กำลังชี้อยู่เพียง record เดียวดังนี้

```
Rs.Delete
```

แต่ถ้าเราต้องการลบ Record ทั้งหมดภายใน Recordset ขณะนั้น ก็สามารถทำได้โดยการกำหนดค่าคงที่ให้กับ method Delete ดังตัวอย่าง

```
Rs.Delete 3
```

ขั้นที่ 4 ปิดการเชื่อมต่อฐานข้อมูล

ขั้นตอนสุดท้ายคือ การปิดการเชื่อมต่อที่ได้สร้างขึ้นมาโดยการหิดการใช้งาน object connection และ recordset ด้วย method Close และถ้าเราไม่ต้องการนำ object ทั้งสองมาใช้งานอีกครั้ง ก็ควรสั่งให้ลบ object ทั้งสองออกจากหน่วยความจำ ด้วยการกำหนดค่าให้เป็น nothing ดังนี้

```
Rs.Close
Conn.Close


Set rs = Nothing
Set Conn = Nothing
```

2.5 ความรู้ฐานข้อมูลเบื้องต้น [6]

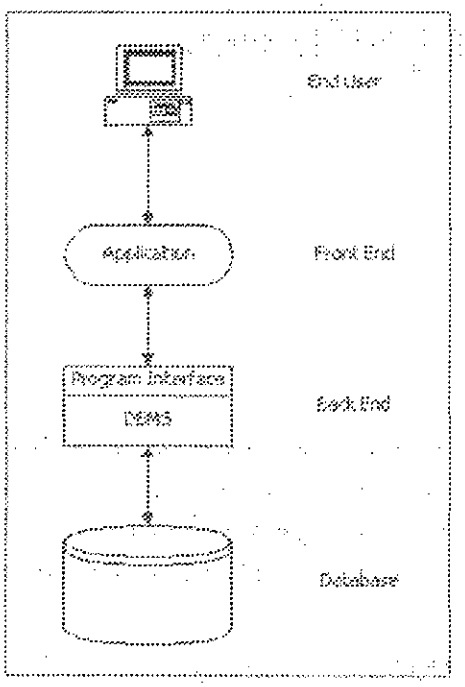
ปัจจุบันการนำระบบการจัดการ เกี่ยวกับฐานข้อมูล(Database System) มาใช้งานเกี่ยวกับการสร้างเว็บไซต์ได้รับความนิยมอย่างมาก โดยเฉพาะอย่างยิ่งในองค์กรที่มีขนาดใหญ่ ทั้งนี้เนื่องจากระบบการจัดการของระบบฐานข้อมูลนั้น มีความสามารถในการควบคุมการทำงานของระบบได้อย่างถูกต้องแม่นยำ และมีความสะดวกรวดเร็ว ทำให้ประสิทธิภาพโดยรวมในการดำเนินงานขององค์กรสูงขึ้นด้วยยั่นเอง

ระบบฐานข้อมูล (Database System)

ระบบฐานข้อมูล คือ การจัดเก็บข้อมูลอย่างมีระบบ ซึ่งผู้ใช้สามารถเรียกใช้ข้อมูลดังกล่าวได้ในลักษณะต่างๆ เช่น การเพิ่มข้อมูล (Add Data), การแทรกข้อมูล (Insert Data), การเรียกใช้ข้อมูล (Retrieve Data), การแก้ไขและการลบข้อมูล (Update & Delete Data) รวมทั้งการเคลื่อนย้ายข้อมูล (Move Data) ด้วย

โครงสร้างของระบบ (Structure of Database)

ระบบฐานข้อมูลในมุมมองของผู้ใช้งาน สามารถแบ่งออกตามลักษณะโครงสร้างของระบบได้เป็นโครงสร้างหลักๆ 2 ส่วนด้วยกัน ได้แก่ ส่วน Front End และ ส่วน Back End



รูปที่2.1 โครงสร้างของระบบฐานข้อมูล

Front End จะอยู่ในลักษณะของโปรแกรมประยุกต์ (Application) ที่อาจจะสร้างจากภาษาต่างๆ เช่น ภาษาระดับสูง, 4GL หรือภาษาอื่นๆ ส่วนนี้โดยปกติจะมีหน้าที่รองรับการทำงานของผู้ใช้ (End User) เพื่อทำหน้าที่ติดต่อกับระบบ

Back End เป็นส่วนที่ทำหน้าที่ในการจัดการกับระบบฐานข้อมูลทั้งหมด ในแง่ของการจัดเก็บและการเรียกใช้ข้อมูลจากแหล่งข้อมูลจริง ซึ่งได้แก่ การปฏิบัติการต่างๆ เกี่ยวกับข้อมูล เช่น การจัดทำ Backup การควบคุมความถูกต้องในการใช้งานข้อมูลพร้อมกัน, การควบคุมความปลอดภัยของระบบ เป็นต้น

องค์ประกอบของระบบฐานข้อมูล

1. Data

เนื่องจากฐานข้อมูลเป็นการจัดรวบรวมข้อมูลให้มีลักษณะเป็นศูนย์กลางอย่างเป็นระบบ ในกรณีที่มีผู้ใช้งานร่วมกันหลายคน (Multi – User) ข้อมูลจะต้องสามารถเรียกใช้งานร่วมกันได้ ซึ่งในทางปฏิบัติผู้ใช้จะมองภาพของข้อมูลที่แตกต่างกันไปตามระดับการออกแบบระบบ

2. Hardware

Hardware ที่เกี่ยวข้องกับระบบจะพิจารณาถึงส่วนประกอบที่สำคัญ 2 ประการ ซึ่งได้แก่

1. สื่อในการเก็บข้อมูล คือ การเก็บข้อมูลด้วย Magnetic Disk รวมไปถึงการติดต่อระหว่างอุปกรณ์ที่เกี่ยวข้อง เช่น I/O Device ต่างๆ

2. ความเร็วในการทำงานของโปรเซสเซอร์และเมมโมรี ซึ่งขึ้นอยู่กับขนาดของข้อมูลในระบบ และจำนวนของผู้ใช้งานเป็นตัวกำหนด

3. User

บุคลากรที่เกี่ยวข้องกับระบบฐานข้อมูลมีดังนี้

3.1 Programmer ทำหน้าที่เขียนโปรแกรมประยุกต์ใช้งาน เพื่อการจัดเก็บและการเรียกใช้งาน ซึ่งจะเป็นไปตามความต้องการของผู้ใช้

3.2 End User เป็นผู้ใช้ข้อมูลจากระบบโดยปกติจะทำงานใน 3 ลักษณะ คือ การอ่านค่า (Read Only) การเพิ่มหรือลบข้อมูล (Add/Delete) และการแก้ไขข้อมูล (Modify Data)

3.3 DAB (Database Administrator) เป็นผู้ควบคุมและบริหารงานของระบบทั้งหมด นั่นคือเป็นผู้ตัดสินใจว่าข้อมูลใดที่จะรวบรวมเข้าสู่ระบบ รวมไปถึงเป็นผู้กำหนดกฎเกณฑ์ที่ใช้ภายในระบบ เช่น วิธีการจัดเก็บข้อมูล เป็นต้น

4. Software

เป็นสื่อกลางระหว่างผู้ใช้งานและข้อมูลที่ถูกจัดเก็บ Software ในส่วนนี้จะเรียกว่า Database Management System (DBMS) นั่นคือความต้องการใช้ข้อมูลจากผู้ใช้จะถูกจัดการโดย DBMS เพื่อที่จะทำงานในลักษณะต่างๆ ไม่ว่าจะเป็นการเรียกใช้ข้อมูล การจัดทำรายงาน และการเปลี่ยนแปลงหรือแก้ไขต่างๆ

2.6 ข้อดีและข้อเสียของการประมวลผลด้วยระบบฐานข้อมูล

ข้อดีของการประมวลผลด้วยระบบฐานข้อมูล

1. ลดความซ้ำซ้อนของข้อมูล (Minimal Data Redundancy)

การจัดเก็บข้อมูลในลักษณะเป็นแฟ้มข้อมูล อาจทำให้ข้อมูลประเภทเดียวกันถูกเก็บไว้หลายๆ แห่งทำให้เกิดความซ้ำซ้อนของข้อมูลขึ้นได้ ดังนั้นการนำข้อมูลรวมมาเก็บไว้ในระบบฐานข้อมูลจะช่วยลดปัญหาความซ้ำซ้อนของข้อมูลได้

2. หลีกเลี่ยงความขัดแย้งของข้อมูลได้ (Consistency of Data)

การจัดเก็บข้อมูลในลักษณะเป็นแฟ้มข้อมูล โดยที่ข้อมูลเป็นเรื่องเดียวกันซึ่งอาจมีอยู่ในหลายแฟ้ม ซึ่งก่อให้เกิดความขัดแย้งของข้อมูลขึ้นได้ ทั้งนี้อาจเนื่องมาจากการแก้ไขข้อมูลที่แฟ้มงานแห่งหนึ่ง แต่ไม่ได้แก้ไขข้อมูลเรื่องเดียวกันที่อยู่ในแฟ้มงานอื่นๆ

3. จัดทำความผิดพลาดในการป้อนข้อมูลให้น้อยที่สุด (Data Integrity)

บางครั้งความผิดพลาดของข้อมูล อาจเกิดจากการป้อนข้อมูลที่ไม่ถูกต้องเข้าสู่ระบบ ดังนั้นการจัดการฐานข้อมูล จึงจำเป็นต้องกำหนดกฎเกณฑ์ในการรับข้อมูลจากการป้อนของผู้ใช้ เพื่อรักษาความถูกต้องของข้อมูลให้ดีที่สุดเท่าที่จะทำได้

4. สามารถใช้ข้อมูลร่วมกันได้ (Sharing of Data)

เนื่องจากระบบฐานข้อมูลเป็นการจัดเก็บข้อมูลไว้ในที่เดียวกัน เมื่อผู้ใช้ต้องการเรียกใช้ข้อมูลจากแฟ้มที่แตกต่างกัน ก็สามารถทำได้โดยง่าย

5. สามารถกำหนดความเป็นมาตรฐานเดียวกันได้ (Enforcement of Standard)

การจัดเก็บข้อมูลไว้ด้วยกันจะสามารถกำหนด และควบคุมความมีมาตรฐานของข้อมูลให้เป็นไปในทิศทางเดียวกันได้ ดังนั้นจึงทำให้ระบบเกิดความเชื่อถือมากยิ่งขึ้น

6. สามารถกำหนดระบบความปลอดภัยของข้อมูลได้ (Security and privacy control)

เนื่องจากระบบจะทำการกำหนดระดับของผู้ใช้แต่ระคน ตามลำดับความสำคัญของผู้ใช้ ดังนั้นจึงสามารถที่จะควบคุมและดูแลความปลอดภัยของข้อมูลภายในระบบได้ดียิ่งขึ้น

7. ข้อมูลมีความเป็นอิสระ (Data Independence)

ระบบฐานข้อมูลจะทำหน้าที่เป็นตัวเชื่อมโยงกับโปรแกรมประยุกต์ ที่ทำงานกับข้อมูลโดยตรง การแก้ไขข้อมูล เช่น ต้องการเปลี่ยนรหัสไปรษณีย์จากเลข 4 หลักเป็นเลข 5 หลัก ก็จะทำการแก้ไขข้อมูลที่เป็นรหัสไปรษณีย์เฉพาะโปรแกรมที่เรียกใช้รหัสไปรษณีย์เท่านั้น ส่วนโปรแกรมอื่นจะเป็นอิสระต่อการเปลี่ยนแปลง

ข้อเสียของการประมวลผลด้วยระบบฐานข้อมูล

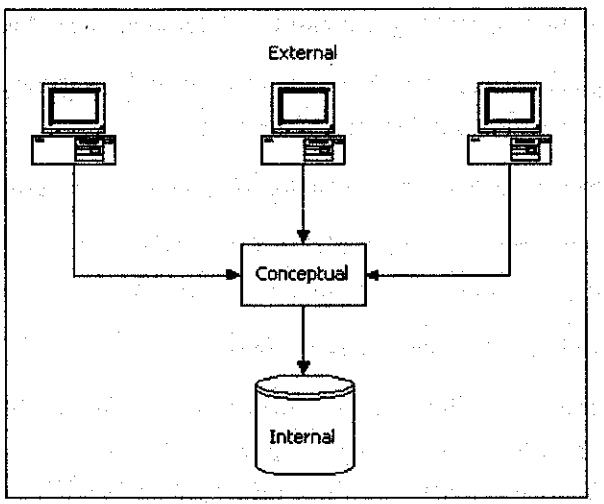
1. ขั้นตอนการออกแบบการดำเนินการ และการบำรุงรักษามีต้นทุนสูง เนื่องจากระบบต้องใช้ผู้เชี่ยวชาญเฉพาะในการออกแบบระบบ

2. ระบบที่มีความซับซ้อนจำเป็นต้องมีผู้ดูแลระบบที่มีความสามารถ เพื่อรองรับสถานการณ์ที่ผิดพลาดที่อาจเกิดขึ้นได้

3. การเสี่ยงต่อการหยุดชะงักของระบบ เนื่องจากข้อมูลอาจถูกจัดเก็บแบบรวมศูนย์ (Centralized Database System) ความล้มเหลวของการทำงานบางส่วน อาจทำให้ระบบฐานข้อมูลโดยรวมหยุดชะงักการทำงานได้

สถาปัตยกรรมระบบฐานข้อมูล (Database System Architecture)

ระบบฐานข้อมูลออกแบบมาเพื่อรองรับโครงสร้างข้อมูลที่มีผู้ใช้หลายคน ดังนั้นจึงต้องมีการแบ่งระดับของข้อมูลออกเป็นหลายระดับ ทั้งนี้เพื่อให้เหมาะสมกับความต้องการของผู้ใช้แต่ละคน เช่น ผู้บริหาร, ผู้ดูแลระบบ, ผู้ใช้งานทั่วไป เป็นต้น การแบ่งระดับดังกล่าวนี้เราเรียกรวมว่าสถาปัตยกรรมของระบบฐานข้อมูล ซึ่งจะอาศัยลักษณะในการมองภาพรวมของระบบ เพื่อจำแนกความแตกต่างออกได้เป็น 3 ระดับดังนี้



รูปที่2.2 สถาปัตยกรรมของระบบฐานข้อมูล

1. Internal Level จะเป็นระดับที่ใช้ในการเก็บข้อมูลจริง ซึ่งจะได้แก่ ส่วนที่ทำหน้าที่ในการจัดการเก็บข้อมูลของระบบ ซึ่งจะครอบคลุมไปถึงการกำหนดชนิดของข้อมูลที่เหมาะสม ตามโครงสร้างที่กำหนด นอกจากนี้ยังรวมไปถึงการจัดการเกี่ยวกับวิธีการในการเข้าถึงข้อมูลแบบต่างๆ ด้วย

2. Conceptual Level เป็นการมองภาพรวมที่เกี่ยวข้องกับข้อมูลทั้งหมด ที่ปรากฏอยู่ในฐานข้อมูลของระบบ ในเชิงของการออกแบบระบบฐานข้อมูล

3. External Level เป็นระดับของข้อมูลที่สนองตอบต่อการใช้งานของผู้ใช้แต่ละคน ซึ่งจะมีผลต่อภาพของข้อมูลที่แตกต่างกัน ดังนั้นมุมมองและวิธีการเข้าหาข้อมูล ของผู้ใช้แต่ละคนจะแตกต่างกันไปด้วย โดยทั่วไปจะเป็นเพียงการใช้ข้อมูลกับฐานข้อมูลเป็นบางส่วน แล้วแต่ผู้ออกแบบระบบจะเป็นตัวกำหนด

2.7 นิยามพื้นฐานในระบบฐานข้อมูล

Data คือ ข้อเท็จจริง (Fact) ที่เกี่ยวข้องกับบุคคล สถานที่ เหตุการณ์ หรือสิ่งของต่างๆ ซึ่งสามารถนับจำนวนได้

Information คือ ข้อมูลที่ถูกจัดรวบรวมให้อยู่ในรูป ที่สามารถจะนำไปประกอบการตัดสินใจอย่างใดอย่างหนึ่งได้

Entity คือ สิ่งใดสิ่งหนึ่งซึ่งใช้สำหรับแสดงความสัมพันธ์ต่อกันระหว่างข้อมูลในระบบ ได้แก่ ชื่อของบุคคล สถานที่ สิ่งของ หรือการกระทำ เช่น Employee, Student เป็นต้น

Attribute คือ รายละเอียดของข้อมูลใน Entity หนึ่ง ที่ใช้แสดงลักษณะ และคุณสมบัติของ Entity ที่ถูกอ้างอิง เช่น Attribute ของ Student จะได้แก่ รหัสประจำตัว, ชั้นหรือแผนกที่สังกัด เป็นต้น

Entity Set คือ Entity หลายๆ ตัวที่มีค่า Attribute เหมือนกัน และสามารถนำมารวมกันในรูปของ Table เพื่อความสะดวกในการเข้าถึงข้อมูลกลุ่มดังกล่าว

Field เกิดจากการรวมตัวของข้อมูลที่เล็กที่สุดภายในคอมพิวเตอร์ที่เรียกกันว่า Bit นำมาประกอบกันจะได้ข้อมูลที่เรียกว่า Byte หรือ Character หากนำ Character มาประกบกันเป็นกลุ่มก็จะได้ข้อมูลที่ขยายตัวเป็นรูปแบบใหม่ๆ ที่เรียกว่า Field อาจกล่าวได้ว่า ส่วนของ Field ก็จะได้แก่ Attribute นั่นเอง

Record เกิดจากการรวมตัวของ Field หรือ Attribute ที่แสดงคุณสมบัติของ Entity ตัวใดตัวหนึ่ง

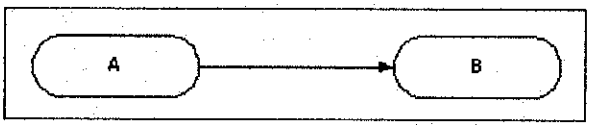
File คือกลุ่มของ Record ชนิดเดียวกัน ที่ถูกนำมารวมกันเป็นหมวดหมู่ ข้อมูลที่อยู่ภายในไฟล์ จะสามารถมองได้เป็นอะเรย์ 2 มิติ นั้นคือในรูปของแถวซึ่งแสดงถึงจำนวน Record และคอลัมน์ซึ่งแทนค่าของ Attribute แต่ละตัวนั่นเอง

Association คือ สัญลักษณ์แสดงความสัมพันธ์ (Relationships) ระหว่าง Entity ซึ่งจะเกิดขึ้นได้กับ Entity ตั้งแต่สองตัวขึ้นไป

รูปแบบความสัมพันธ์ระหว่าง Entity (Type of Entity Association)

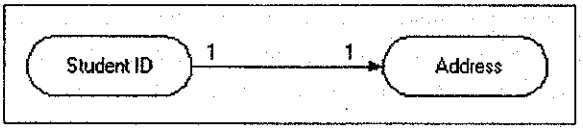
ค่าของ Entity แต่ละตัวจะถูกเก็บอยู่ในรูปของ File ในขณะเดียวกันค่าของ Attribute ก็จะได้แก่ ค่าของ Field นั่นเอง ส่วนความสัมพันธ์ระหว่าง Entity ก็จะได้แก่ความสัมพันธ์ระหว่าง File ที่ถูกนำเสนอในรูปของการกำหนดค่าของ Field ใน File หนึ่ง เพื่อแสดงความสัมพันธ์ไปยังอีก File หนึ่งนั่นเอง

ความสัมพันธ์ระหว่าง Entity สามารถเขียนแทนได้ด้วย สัญลักษณ์หัวลูกศร ซึ่งสามารถแบ่งชนิดของความสัมพันธ์ได้เป็น 3 ลักษณะ ดังต่อไปนี้



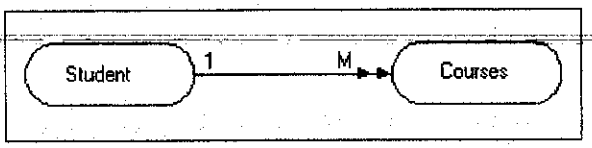
รูปที่2.3 ความสัมพันธ์ระหว่าง Entity

1. ความสัมพันธ์แบบหนึ่งต่อหนึ่ง (One to Relationships) หมายถึง ในช่วงระยะเวลาที่กำหนดค่าของ Entity A จะมีความสัมพันธ์กับค่าของ Entity B เพียงค่าเดียวเท่านั้น นั่นคือ หากทราบค่าของ Entity A ก็สามารถที่จะหาค่าของ Entity B ได้ด้วย เช่น ในกรณีของเลขประจำตัวนักศึกษาจาก Entity A จะอ้างถึงค่าของ Entity B ได้เพียงค่าเดียวเท่านั้น คือ ที่อยู่ของนักศึกษาคนนั้นๆ



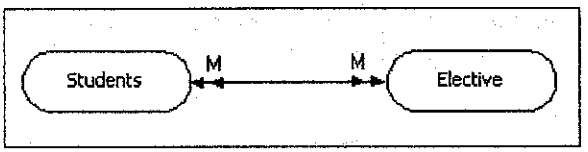
รูปที่2.4 แบบหนึ่งต่อหนึ่ง

2. ความสัมพันธ์แบบหนึ่งต่อกลุ่ม (One to Many Relationships) หมายถึง ในช่วงระยะเวลาปกติกำหนดค่าของ Entity A จะมีความสัมพันธ์กับค่าของ Entity B ได้มากกว่า 1 ค่า เช่น นักศึกษา 1 คน สามารถเลือกเรียนวิชาใดหลายวิชา เป็นต้น ความสัมพันธ์ในลักษณะนี้จะเกิดขึ้นเป็นส่วนใหญ่ในระบบ



รูปที่2.5 แบบหนึ่งต่อกลุ่ม

3. ความสัมพันธ์แบบกลุ่มต่อกลุ่ม (Many to Many Relationships) หมายถึงในช่วงระยะเวลาที่กำหนด ทั้งค่าของ Entity A จะมีความสัมพันธ์กับค่าของ Entity B ได้มากกว่าหนึ่งค่า เช่น ในกรณีของวิชาเรียนเลือกเสรี ที่มีนักศึกษาจากหลายแผนก ลงทะเบียนเรียนรวมกันได้ เป็นต้น



รูปที่2.6 แบบกลุ่มต่อกลุ่ม

2.8 วิธีการค้นหาข้อมูล

Search Table [4]

ในทางคณิตศาสตร์แนวคิดในเรื่องการทำงานของเซทจะเป็นสิ่งสำคัญมาก มีการศึกษาต่อมาจนกระทั่งเป็นรูปแบบที่ชัดเจน ที่เรียกว่า Set Theory ในส่วนของการโปรแกรมคอมพิวเตอร์การใช้วิธีการเกี่ยวกับเซทไม่ได้รับความนิยมเท่าใดนัก แต่อย่างไรก็ตามเราได้นำความรู้เรื่องเซทมาประยุกต์ในการสร้างโครงสร้างข้อมูลแบบใหม่ที่เรียกว่า Search table ซึ่งได้รับความนิยมมาก และมีรายละเอียดดังนี้

1. การกำหนดค่า จะเรียกว่า 2 Tuple โดยแทนด้วยสัญลักษณ์ (Ki, Vi) โดยที่ K จะถูกเรียกว่า Key Field และ V จะถูกเรียกว่า Value Field ซึ่งสามารถเขียนโดยใช้ลักษณะการเขียนแบบเดียวกับเซทได้ดังนี้

$$S = \{(KO,VO), (K1,V1),\ldots,(Kn,Vn)\}$$

2. การทำงานของ Search table จะได้แก่ การสร้างตาราง, การเพิ่มข้อมูลเข้าสู่ตาราง, การลบข้อมูลออกจากตาราง และการกำหนดตำแหน่งของตาราง เป็นต้น

การทำงานของ Search table ซึ่งปกติการเปลี่ยนแปลงค่าของ Value Field จะอาศัยการดำเนินการผ่านค่า Key Field ซึ่งจะเป็นค่าที่ถูกกำหนดให้มีเพียงหนึ่งเดียว (Unique) เสมอ โดยไม่จำเป็นต้องระบุถึงตำแหน่งของค่านั้นๆ เนื่องจาก Key Field จะมีได้เพียงค่าเดียว ดังนั้นการทำงานของSearch Table จะไม่ยอมให้มีค่าที่ซ้ำกันภายในตารางได้

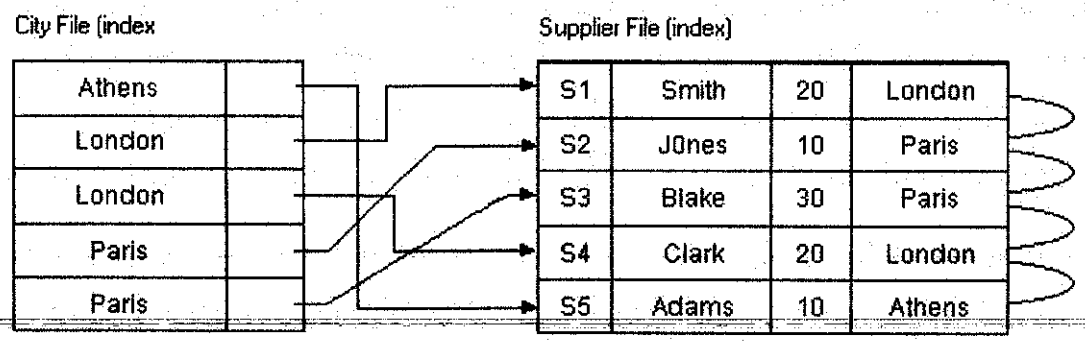
Index

Index เป็น Search Table วิธีหนึ่งที่ช่วยในการเพิ่มความเร็วในการค้นหาข้อมูลนั่นเอง การทำงานของ Index จะมีลักษณะเป็นการทำงานที่เหมือนกับการค้นหาหนังสือในห้องสมุด โดยดัชนีการค้นหา เช่น บัตร ผู้แต่ง เป็นต้น

โครงสร้างการทำงานแบบ Index จะเกี่ยวข้องกับคีย์ที่ต้องการค้นหา เช่น หากไฟล์มีการจัดเรียงลำดับเรคคอร์ดไว้แล้ว และทำการกำหนด Index สำหรับการค้นหาไว้หลายๆ ตัว Index ที่มีค่า Search Key ระบุอยู่ตามลำดับเรคคอร์ดจะถูกเรียกว่า Primary Index ในขณะที่ตัวอื่นๆ จะถูกเรียกว่า Secondary Index ปกติแล้วค่า Search Key ของ Primary Index จะถูกเรียกว่า Primary Key นั่นเอง

โครงสร้างการทำงานแบบใช้ Index สามารถแบ่งออกได้เป็น 2 ชนิด คือ

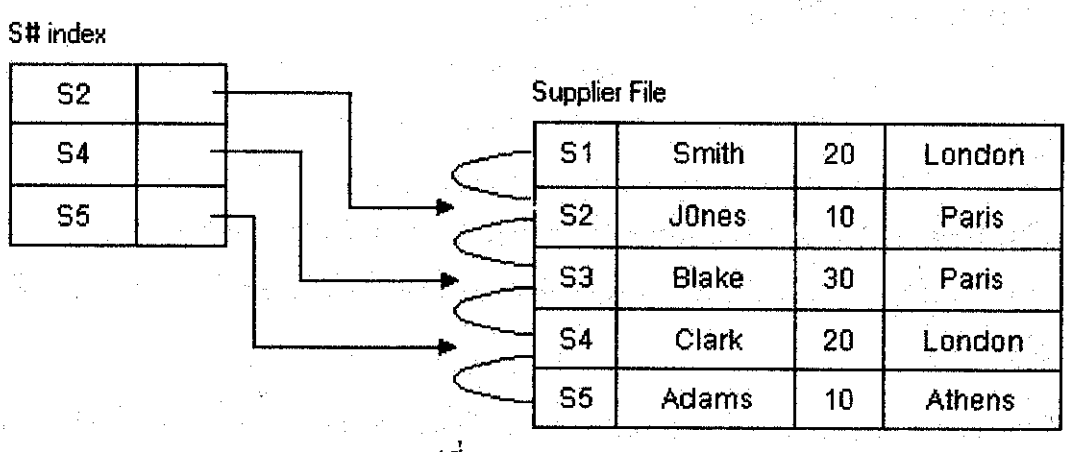
1. **Dense index** ได้แก่การค้นหาข้อมูล ซึ่งการกำหนดค่าตารางดัชนีจะเลือกใช้ค่าที่มีปรากฏอยู่ในทุกเรคคอร์ด ซึ่งการค้นหาใช้ค่าของดัชนี ซึ่งมี Pointer ชี้ไปยังเรคคอร์ดที่ถูกระบุ



รูปที่ 2.7 Dense index

จากภาพจะเห็นได้ว่า CITY จะถูกกำหนดให้เป็น Index Record การค้นหาข้อมูลจะทำได้จากการเปรียบเทียบค่าที่เท่ากัน (จากตาราง Supplier ที่มีค่า CITY เหมือนกัน) จากนั้นจะตามค่า Pointer ไปยังเรคคอร์ดที่ถูกระบุ เพื่อทำการโปรเซสต่อไป

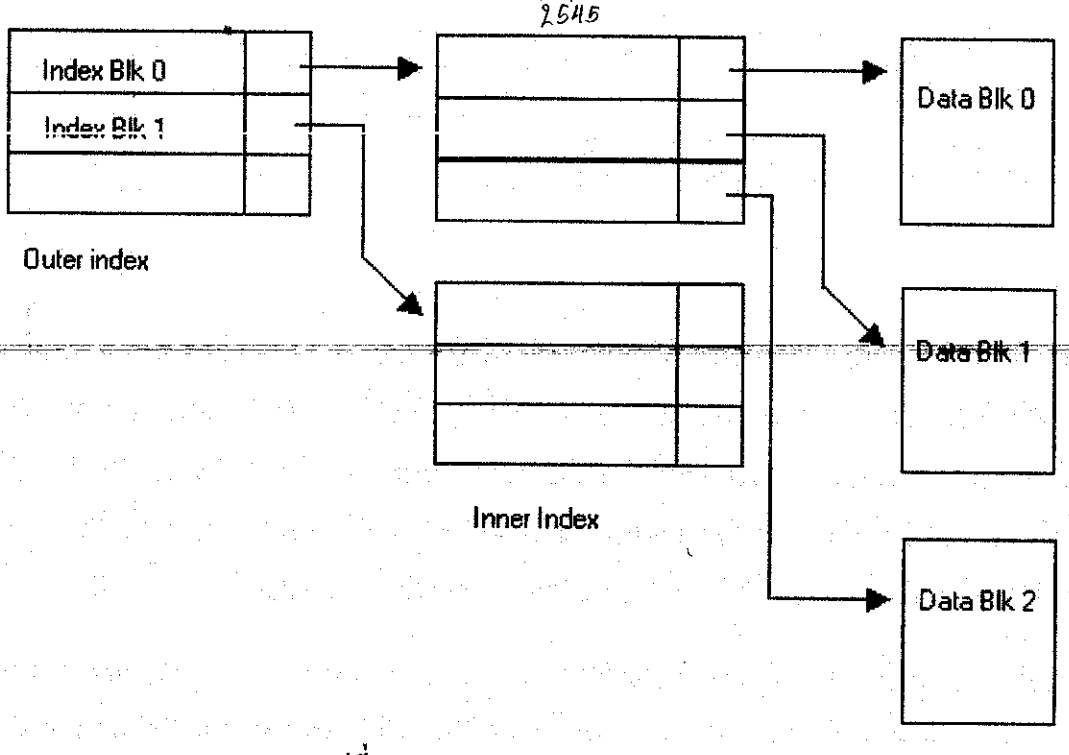
2. Sparse Index การค้นหาด้วยวิธี Index จะถูกกำหนดขึ้นจากค่าที่ปรากฏอยู่เพียงบางเรคคอร์ดเท่านั้น การค้นหาจะดำเนินการโดยการเริ่มต้นจากตำแหน่งของ Index ที่มีอยู่ จากนั้นจึงเลื่อนตำแหน่งการชี้ไปยังเรคคอร์ดถัดไป จนกระทั่งพบกับเรคคอร์ดที่ต้องการ



รูปที่ 2.8 Sparse index

เช่น ต้องการหาค่าของ S3 ก็จะเริ่มค้นหาจาก Index Record ที่ S2 ก่อน จากนั้นจะไปยังตำแหน่งที่ Pointer ชี้อยู่ และเริ่มต้นค้นหาในตำแหน่งถัดไปนั่นเอง

การค้นหาโดยอาศัยดัชนี ไม่ว่าด้วยวิธี Dense หรือ Search มีข้อเสีย เช่นในกรณีที่เรคคอร์ดมีจำนวนมาก จำนวนของ Index ที่ต้องการใช้ก็จะมีจำนวนมากตามไปด้วย ดังนั้นการแก้ไขอาจกระทำได้โดยการเพิ่มจำนวนของ Index ให้เป็นสองระดับ นั่นคือ Outer Index และ Inner Index โดยจัดโครงสร้างแบบ Sparse Index การค้นหาเรคคอร์ดที่ต้องการจะเริ่มจากการใช้ Binary Search เพื่อหาค่าจากเรคคอร์ดที่ถูก Pointer ของ Inner Index ซึ่งอยู่นั่นเอง



รูปที่ 2.9 Outer Index และ Inner Index

เนื่องจากการทำงานโดยใช้ Index เหมาะสมที่จะทำงานกับไฟล์ ที่มีการจัดเรียงลำดับเรคคอร์ดไว้แล้ว ดังนั้นการ Update ข้อมูลกระทำได้เพียงสองแบบ คือ การลบและการเพิ่มข้อมูล