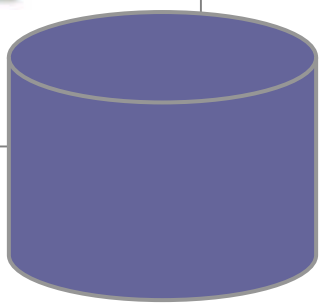
DATABASE PROGRAMMING

DB 102

```
Sub sysINI()

    GenTmpDb

    '-------- Read Application Define Caption
    '
    '
    '
    Dim rst As Recordset
    Dim i
    Set rst = DBEngine(0)(0).OpenRecordset("Select * from sysConfig where CONFIG like 'CALL*'")
    i = 0
    Do Until rst.EOF

        If MID(rst("CONFIG"), 5) <> rst("CONFIGVALUE") Then
            i = i + 1
            ReDim Preserve WordFr
            ReDim Preserve WordTo
            WordFrom(i - 1) = MID
            WordTo(i - 1) = rst("
        End If

        rst.MoveNext
    Loop

    WordCount = i
```



โดย : อุทัย เกียรติวิกรัย

จัดพิมพ์ครั้งที่ 2

CONTENTS

คำนำ

หนังสือเล่มนี้เขียนจากประสพการณ์การเป็น Programmer บนระบบงานฐานข้อมูลของผู้เขียนเอง โดยใช้ VISUAL BASIC และ VISUAL BASIC FOR APPLICATION เป็นเครื่องมือในการพัฒนา หลายส่วนที่ได้แสดงในตำราเล่มนี้ อ้างอิงจาก HELP ของตัว Microsoft Access ซึ่งเป็นภาษาอักฤษและได้นำมาอธิบายเพิ่มเติม

ก่อนใช้งานหนังสือเล่มนี้ ผู้อ่านควรศึกษา Microsoft Access ในการใช้งาน Table, Form, Report รวมถึงการศึกษาความเข้าใจขององค์ประกอบฐานข้อมูลมาก่อนแล้ว

ผู้เขียนหวังเป็นอย่างยิ่งว่าหนังสือเล่มนี้ จะช่วยให้ผู้อ่านที่ต้องการเป็น Programmer สำหรับงาน Database ได้เข้าใจ Microsoft Access ในแง่ของการพัฒนาระบบงาน Database และช่วยให้ผู้อ่านเข้าใจขบวนการพัฒนาระบบงานด้านฐานข้อมูลได้ดีขึ้น

หนังสือเล่นนี้ถูกจัดทำครั้งแรก เพื่อใช้ในการพัฒนาบุคลากรด้านสารสนเทศ ของ มหาวิทยาลัยเกษตรศาสตร์ มหาวิทยาลัยวลัยลักษณ์ มหาวิทยาลัยมหาสารคาม มหาวิทยาลัยนเรศวร ภายใต้ชื่อ UNIT GROUP ภายหลังผู้เขียนได้ใช้หนังสือนี้ประกอบในการอบรมวิชาด้านฐานข้อมูลบ่อยครั้ง จึงได้รวบรวมและ จัดทำเป็นสื่ออิเลคโทรนิคส์อีกครั้ง เพื่อให้สามารถสำเนา และ จัดพิมพ์ใหม่ได้สะดวกขึ้น

อุทัย เกียรติวิกรัย
10 มิถุนายน 2547
email: himmidi@yahoo.com
download ได้ที่ : www.singingweb.com

C H A P T E R 1

การพัฒนาซอฟต์แวร์ด้วย ACCESS

1

นอกจากการใช้งานเป็นเครื่องมือทางด้านฐานข้อมูลแล้ว ACCESS สามารถนำมาพัฒนาซอฟต์แวร์ หรือระบบงานได้ การพัฒนาซอฟต์แวร์ด้วย ACCESS มีประเด็นสำคัญหลายๆประการที่ผู้พัฒนาซอฟต์แวร์ ซึ่งอาจจะมาจากการเป็นผู้ใช้งาน ACCESS มาก่อน หรือจะเป็น PROGRAMMER มาก่อนแล้วก็ตาม ต้องทำความเข้าใจขบวนการ และแนวคิดในการพัฒนาซอฟต์แวร์ด้วย Access เสียก่อน

เนื้อหาของบทนี้ จะเป็นการนำเสนอภาพรวมของการนำ ACCESS มาใช้ในการพัฒนาระบบงาน และให้มีความเข้าใจเบื้องต้นกับการพัฒนาซอฟต์แวร์สำหรับใช้งานจริง

CONTENT

- พื้นฐานการพัฒนา SOFTWARE
- ENVIRONMENT ของระบบงานทางด้านฐานข้อมูล
- การ ATTACH TABLE
- การแยกฐานข้อมูลออกจากโปรแกรม
- การพัฒนาระบบด้วย ACCESS
- การกำหนด STARTUP ของ ACCESS
- ขั้นตอนพัฒนาระบบด้วย ACCESS

พื้นฐานการพัฒนาซอฟต์แวร์

ขณะที่เราใช้ ACCESS ในการจัดเก็บข้อมูล ออกรายงาน ทำ QUERY หรือแม้กระทั่งการทำ FORM, REPORT ยังจัดว่าเรายังคงใช้ ACCESS ในลักษณะเป็นเครื่องมือทางด้านฐานข้อมูล แต่ในการพัฒนาซอฟต์แวร์ที่ใช้งานจริง เราคงไม่ได้ให้ผู้ใช้เป็นคนออกรายงานด้วยการสร้างรายงานเอง แต่เรา ในฐานะ PROGRAMMER เป็นผู้สร้างสิ่งเหล่านี้ไว้ และเราเป็นผู้ทำหน้าจอติดต่อกับ ผู้ใช้งาน ในลักษณะ MENU เพื่อนำมาเรียกใช้ REPORT FORM ที่เราสร้างไว้อีกครั้งหนึ่ง ขั้นตอนนี้เรามักเรียกกันว่าการทำ USER INTERFACE

นอกจากการทำ USER INTERFACE ในการเรียกใช้รายงาน ข้อมูลที่มีแล้ว เราอาจต้องคำนึงถึงการรักษาความปลอดภัย ในการใช้งานรายงาน หรือหน้าจอบันทึกบางตัว ผู้ใช้คนใดใช้ได้ ใช้ไม่ได้ ในบางหน้าจอบันทึกข้อมูล เราอาจต้องทำการตรวจสอบการบันทึก หรือการทำรายการพิเศษๆ ที่ซับซ้อนมากยิ่งขึ้น นอกจากการบันทึกเข้าไปใน TABLE ตรงๆ

ACCESS เป็น SOFTWARE ที่ไม่ได้พัฒนามาเพื่อการใช้งานเป็นเครื่องมือทางด้านฐานข้อมูลแต่เพียงอย่างเดียว แต่ยังถูกพัฒนามาเพื่อเป็นเครื่องมือในการพัฒนาซอฟต์แวร์ หรือระบบงานด้วย การใช้งาน ACCESS ในแง่มุมนี้จะทำให้สิ่งที่เราพัฒนาบน ACCESS กลายเป็น SOFTWARE ที่ทำงานบนฐานข้อมูลที่คล่องตัวสูงตัวหนึ่ง

เราทำอะไรกับ ACCESS ได้บ้าง

ในการใช้ ACCESS พัฒนาซอฟต์แวร์ เราจะพบว่าสิ่งที่เราเข้าไปเกี่ยวข้องในอนาคตอันใกล้นี้ ก็คือ OBJECT , PROPERTIES, EVENT ฟังชื่อแล้วอาจจะคุ้นหูมาบ้าง ตัวอย่างที่จะพูดถึงขณะนี้จะทำให้เข้าใจกลไกของทั้ง 3 ส่วนที่จะเข้ามาเกี่ยวข้องกับการพัฒนาซอฟต์แวร์ของเราในอนาคต

ควบคุมองค์ประกอบต่างๆของ ACCESS (OBJECT) : ใน ACCESS เรามี FORM, REPORT, QUERY, TABLE เราสามารถเรียกองค์ประกอบเหล่านี้ว่าเป็น OBJECT ในการพัฒนาระบบงาน จะมีการเรียก FORM, REPORT จากจุดต่างๆ ในระบบงานได้เสมอ การเรียก OBJECT เหล่านี้จะไม่ได้เรียกจากหน้าจอ DATABASE WINDOWS ของ ACCESS เอง แต่จะเรียกจาก FORM ที่เราสร้างขึ้นเป็นส่วนใหญ่ การเรียกใช้ดังกล่าว จะเกี่ยวข้องกับการใช้คำสั่งในการเรียก OBJECT ให้ขึ้นมาทำงาน เช่น

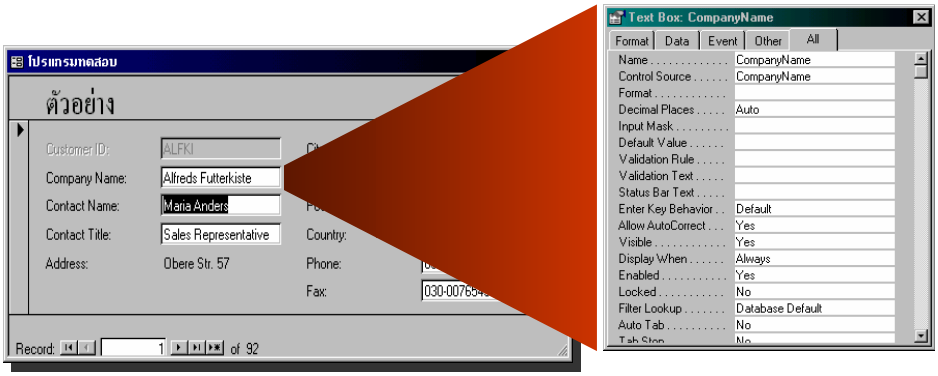
- หน้าจอบันทึกรายชื่อหนังสือ มีปุ่มที่ใช้ในการเรียกหน้าจอสำหรับการค้นหารายชื่อหนังสือ (FORM เรียก FORM)
- หน้าจอการออกใบเสร็จเมื่อบันทึกข้อมูลใบเสร็จแล้ว ต้องการพิมพ์ใบเสร็จที่หน้านั้นเลย จำเป็นต้องเรียกรายงานที่เป็นใบเสร็จออกทางเครื่องพิมพ์ (FORM เรียก REPORT)

นอกจากการเรียก OBJECT ที่เป็น FORM, REPORT, QUERY แล้ว เรายังมีการสั่งให้ CONTROL ใน FORM ซึ่งเราจัดเป็น OBJECT ประเภทหนึ่ง ให้ทำการต่างๆ กัน

การพัฒนาระบบของเราจะเกี่ยวข้องกับ OBJECT เป็นอย่างมาก เราต้องทราบว่า ณ ขณะหนึ่งๆ เรากำลังทำการกับ OBJECT อะไร OBJECT นั้นเป็นของใคร เช่น LISTBOX OBJECT ที่อยู่ใต้ FORM OBJECT ชื่อ prgSTUDENT เป็นต้น

กำหนด PROPERTIES ของ OBJECT: OBJECT ทุกตัวจะมีค่าที่ใช้ในการกำหนดคุณลักษณะของมัน ที่เรียกว่า PROPERTIES ตรงนี้เราคงคุ้นเคยกันอยู่แล้วในการใช้งาน ACCESS ทั่วไป เราสามารถกำหนด PROPERTIES ของ OBJECT ได้เอง ขณะที่ FORM หรือ REPORT ทำงานอยู่ ซึ่งโดยปกติ ขณะที่เราเป็น User จะทำไม่ได้ การกำหนดค่า Properties นี้จะทำผ่านการเขียนโปรแกรมด้วยภาษา Visual Basic แทรกเข้าไปใน FORM หรือ REPORT นั้นๆ ตัวอย่างเช่นการกำหนดสี ของ CONTROL ตามมูลค่าการสั่งซื้อ ถ้าการสั่งซื้อมีค่ามากกว่า 1000 บาท ให้สีของตัวเลขเป็นสีแดง เป็นต้น

```
if  ME.ctlAmount > 1000 then
    Me.CtrlAmount.Forecolor = rgb(255,0,0)
    Else
    Me.CtrlAmount.ForeColor = rgb(0,0,0)
end if
```

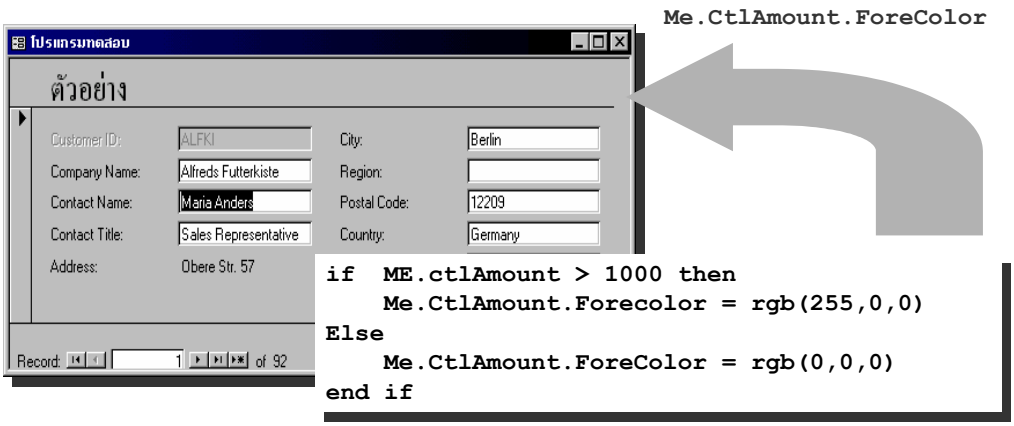


การกำหนด Properties ของ OBJECT จะทำให้ OBJECT แต่ละตัว ประพฤติตัว ตามความต้องการของเรา ดังนั้นเราจะต้องทราบว่า

- มี Properties ใดบ้างในแต่ละ Object
- การกำหนด Properties ทำอย่างไร
- เมื่อใดกำหนดได้ เมื่อใดกำหนดไม่ได้

กำหนดการประมวนผลพิเศษบน EVENT ต่างๆของ FORM, REPORT : เพราะในขณะที่เราทำการบันทึกข้อมูล ย้าย FIELD ที่กรอกข้อมูล CLICK ปุ่มต่างๆ หรือ ทำการส่วนต่างๆบนจอ จะเกิด สิ่งที่เรียกว่า EVENT ที่เราสามารถ กำหนดให้ เหตุการณ์ หรือ EVENT แต่ละ EVENT บนแต่ละ OBJECT ทำการคำนวณ หรือประมวนผลอะไรที่พิเศษๆ ไปกว่าที่ ACCESS มีได้ เช่น เมื่อคุณกรอกจำนวนเงินที่เกินกว่า 1000 บาท ให้มี ข้อความเตือน (MSGBOX) ก่อนทำการบันทึก เป็นต้น

การทำการดังกล่าว เราจะต้องเขียนโปรแกรม แทรกเข้าไปใน EVENT ต่างๆ ด้วยภาษา Visual Basic เช่นกัน ไม่ใช่ทุก Object ที่มี Event และ Event ที่มีในแต่ละ Object ก็จะไม่เหมือนกัน เราจะต้องเรียนรู้ว่า



- OBJECT แต่ละตัว มี EVENT อะไรบ้าง
- อะไรเกิดก่อนหลังบ้าง

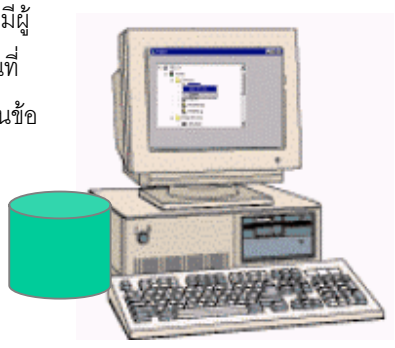
การเชื่อมต่อกับโปรแกรมภายนอก เช่นเราต้องการ COPY FILE MDB ที่ทำการบันทึกลง DISK แต่ไฟล์ใหญ่เกินไป ต้องใช้โปรแกรมย่อไฟล์ ซึ่งอาจเป็น WINZIP เราจะต้องโปรแกรม ACCESS ให้ไปเรียกใช้โปรแกรม WINZIP เพื่อมาทำการย่อขนาดไฟล์ให้เราเป็นต้น การเรียกใช้โปรแกรมภายในของ Access ทำได้ตั้งแต่การเรียกโปรแกรมตรงๆ การเรียก DLL หรือกระทั่ง การทำงานร่วมกับ ActiveX control เป็นต้น

ENVIRONMENT ของระบบงาน

สำหรับระบบงานด้านฐานข้อมูล ลักษณะสภาพแวดล้อมของ HARDWARE SOFTWARE ที่ทำงานร่วมกับโปรแกรมของเราจะพบได้ในหลายๆลักษณะ โอกาศที่เป็นได้คือ

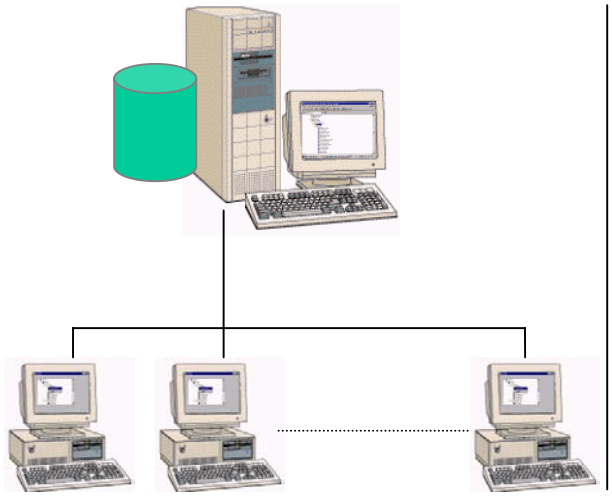
STANDALONE APPLICATION

เป็นสภาพที่เป็นพื้นฐานที่สุด คือ 1 เครื่องทำงานบนฐานข้อมูลของตนเอง มีผู้ใช้ที่เวลาหนึ่งๆ เพียง 1 คนเท่านั้น การพัฒนาระบบในลักษณะนี้มีประเด็นที่ต้องคำนึงถึงน้อยที่สุด แต่มีข้อจำกัดการใช้งานค่อนข้างต่ำ ไฟล์ที่เป็นฐานข้อมูลจะถูกเก็บบนเครื่องของผู้ใช้ มักใช้กับระบบงานที่เล็ก หรือต้องการให้มีผู้ใช้คนเดียว เช่นระบบงานเงินเดือนพนักงาน เป็นต้น ความเร็วของระบบจะขึ้นกับเครื่องที่ใช้เป็นสำคัญ



MULTIUSER : FILE SHARING

ในสภาวะนี้จะมีผู้ใช้ในระบบหลายคนที่เข้ามาทำงานพร้อมๆกันกับฐานข้อมูลของเรา สิ่งที่เราบันทึก จะถูกเห็นได้โดยผู้ใช้คนอื่นๆ พร้อมๆ กันกับที่เราเห็นข้อมูลที่ผู้ใช้คนอื่นบันทึก ได้ประโยชน์มากในการที่สามารถเห็นข้อมูลถึงกัน



MULTIUSER: FILE SHARING
เก็บ FILE ที่เป็นฐานข้อมูลไว้บน NETWORK แล้วให้ทุกเครื่องเข้าไปอ่าน

ได้ บันทึกข้อมูลพร้อมๆกันหลายคนได้ แต่ไม่สามารถรับ LOAD งานได้มาก มีประเด็นที่ต้องคำนึงถึงมากเช่น

การเขียนข้อมูลพร้อมกัน ที่ ROW เดียวกัน TABLE เดียวกัน จะมีผู้ที่เขียนเข้าไปได้เพียงผู้เดียว เนื่องจากผู้เขียนคนที่ 2 จะเขียนทับข้อมูลของผู้เขียนคนแรก

(ประเด็นนี้ ACCESS จะไม่ยอมเขียนข้อมูลของคนที่ 2 เข้าไปถ้าเราใช้ FORM ในการบันทึก เว้นแต่การสั่งเขียนโดยคำสั่ง SQL)

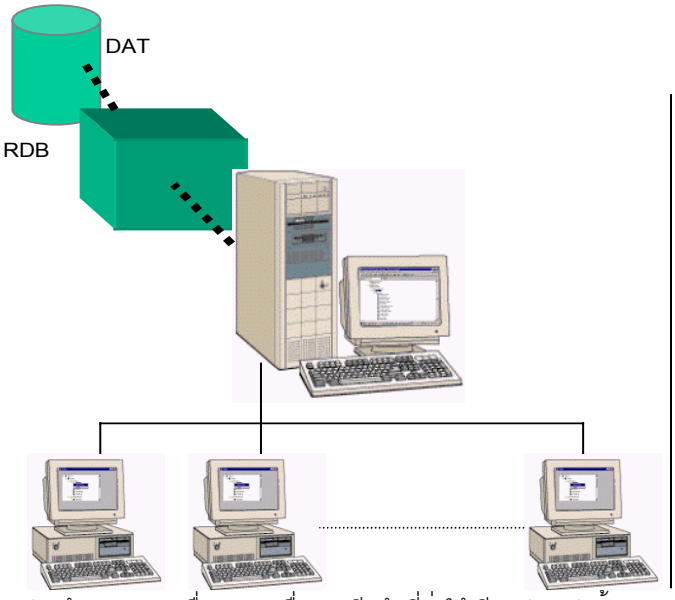
ข้อมูลที่แสดงบนจอ อาจไม่ใช่ข้อมูล ณ ขณะนั้น ๆจริง ต้องทำการ REFRESH หน้าจอ เพื่อเรียกข้อมูลล่าสุดมาแสดง เนื่องจากอาจมีผู้ใช้คนอื่น ทำการปรับเปลี่ยนข้อมูลหน้าจอไปเป็นอย่างอื่นแล้ว หลังจากเราอ่านข้อมูลครั้งสุดท้าย

การ LOCK RECORD การเขียนข้อมูลทุกครั้ง ACCESS หรือระบบฐานข้อมูลอื่นๆ ก็จะทำการ LOCK ฐานข้อมูล ก่อนทำการเขียนทุกครั้ง ระหว่างที่ LOCK ผู้อื่นจะทำการ อ่าน หรือ เขียน ROW นั้นไม่ได้ หากเครื่องที่เขียนนั้นหยุดทำงานขณะนั้นพอดี หรือ มีการวิ่งเข้าไป เขียนข้อมูลพร้อมกันพอดี อาจเกิดสิ่งที่เรียกว่า DEAD-LOCK เหมือนการ HANG ของ PC แต่เป็นการ HANG ของตัวฐานข้อมูล ดังนั้นการอ่านเขียนข้อมูลที่มีการ อ่านเขียน จากผู้ใช้หลายๆคน จะต้องทำให้สั้นที่สุด และต้องให้ความระมัดระวังเป็นพิเศษ

เขียนอ่านโดยเครื่องลูก เนื่องจากข้อมูลจะถูกเก็บเป็น FILE ไว้ที่ SERVER และทำการอ่านเขียนโดยเครื่องของผู้ใช้ หรือเครื่องลูก หากเครื่องลูกมีข้อบกพร่อง เช่น มี VIRUS มีความไม่เสถียรของ ระบบเครือข่าย อาจทำให้การเขียนข้อมูลผิดพลาด ทำให้ข้อมูลเสียหายได้ จะต้องเลือกเครื่องที่มีประสิทธิภาพตรงกับระบบงาน และมีการความเร็วในการทำงานที่ใกล้เคียงกัน

MULTIUSER : CLIENT-SERVER

เป็นสภาวะที่สามารถรับ LOAD งานได้ดี และสามารถแก้ปัญหาในแบบ FILE SHARING ได้ แต่มีราคาต้นทุนที่สูงกว่า โดยในสภาวะแบบนี้ จะมี DATABASE SERVER เป็น SOFTWARE ที่ทำงานบนตัว SERVER คอยทำหน้าที่



MULTIUSER: CLIENT-SERVER
ใช้ระบบจัดการฐานข้อมูลเป็นตัวช่วยจัดการ

เขียนอ่านข้อมูลแทน เครื่องลูก เครื่องลูกมีหน้าที่สั่งให้เขียนอ่านเท่านั้น หากเครื่องลูกหยุดทำการ หรือมีปัญหา ตัว DATABASE SERVER จะทำการตัดตัวลูกออกจากระบบแทน การเขียนอ่านพร้อมกัน ก็จะจัดการโดย DATABASE SERVER เพื่อเลือกตามลำดับการเขียนอ่านให้ ประเด็นสำคัญในการใช้สภาวะแวดล้อมดังกล่าวคือ

การเชื่อมต่อเข้ากับ DATABASE SERVER จะต้องมีการใช้ DRIVER ในการติดต่อ เช่นถ้าเราใช้ ACCESS ในการเชื่อมต่อกับ INFORMIX เราจะต้องมี DRIVER ในการเชื่อมต่อกับ INFORMIX ผ่าน ODBC เป็นต้น

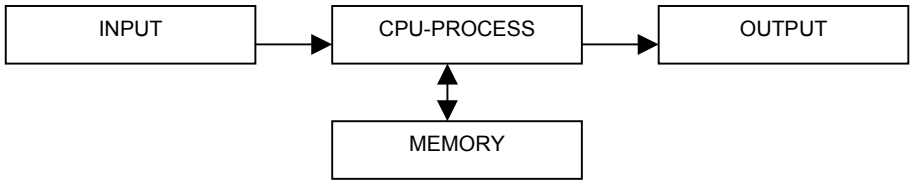
ความซับซ้อนในการจัดการ เนื่องจากฐานข้อมูลมิได้เป็นไฟล์ปกติเหมือน FILE ACCESS แบบเดิม การดูแลจัดการ จะซับซ้อนมากกว่า และเป็นเรื่องเฉพาะของผู้ผลิตแต่ละราย

ความเร็วช้าในการใช้งาน จะไม่ขึ้นอยู่กับการทำงานของระบบงานของเราแต่เพียงอย่างเดียว หากจะขึ้นกับ ตัว DATABASE SERVER , DRIVER ที่ใช้เชื่อมต่อ หากเราเขียนโปรแกรมไม่เหมาะสม อาจทำงานช้ากว่าระบบที่เป็น FILE SHARING

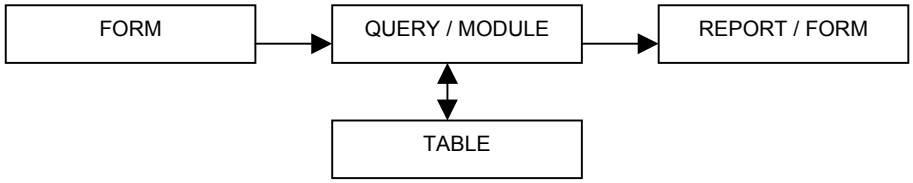
การพัฒนาระบบด้วย ACCESS

ดังที่ได้กล่าวไปแล้ว เรื่องการพัฒนาระบบบน ACCESS จะเกี่ยวข้องกับการเขียนโปรแกรมที่กระทำกับ OBJECT เป็นส่วนใหญ่ โดยหลักการของระบบคอมพิวเตอร์ในทุกๆเรื่อง จะมีการทำงานอยู่ในรูปของการ รับข้อมูล (INPUT) ประมวลผล (PROCESS) , จัดเก็บ (MEMORY) , และ แสดงผล (OUPUT) หากเปรียบเทียบกับการใช้ ACCESS แล้วจะได้ตามตาราง

Computer System Model



Access / Database Application



เปรียบเทียบ ACCESS กับ COMPUTER

COMPUTER	ACCESS	NOTE
INPUT	FORM	เราใช้เป็นจุดรับข้อมูล และรับคำสั่งจากผู้ใช้โปรแกรมของเรา นำคำสั่งไปประมวลผล หรือนำข้อมูลไปเก็บ
PROCESS	FORM / MODULE / QUERY	เป็นขั้นตอนของการเขียนโปรแกรม และ ประมวลผลคำนวณ แล้วตอบตอบโต้กับผู้ใช้ เช่นกด ปุ่มหมายเลข 1 ให้ออกรายงานสรุปเป็นต้น
MEMORY	TABLE	เป็นศูนย์กลางข้อมูลของระบบ จะเห็นว่าทุกส่วนจะกลับมาอ่านเขียนข้อมูลที่จุดนี้ ดังนั้น การออกแบบฐานข้อมูลที่ดี การกำหนด TABLE , COLUMN เงื่อนไขต่างๆ รวมทั้งการกำหนดความสัมพันธ์ จะเป็นจุดเริ่มต้นที่สำคัญของระบบงานทั้งหมด

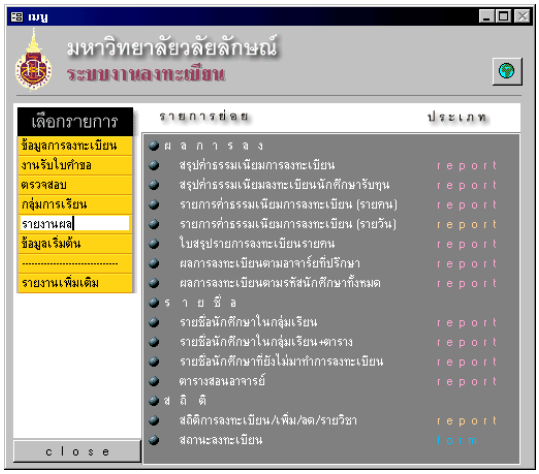
COMPUTER	ACCESS	NOTE
OUTPUT	FORM / REPORT	เป็นการนำข้อมูลที่จัดเก็บไว้มาใช้ประโยชน์ ด้วยรูปแบบการนำเสนอแบบต่างๆ การนำเสนอ จะเรียกเอาข้อมูลโดยตรง หรือจาก QUERY รวมกับการใส่สูตรต่างๆ ไว้ในรายงาน แล้ว ให้ FORM เป็นผู้เรียก

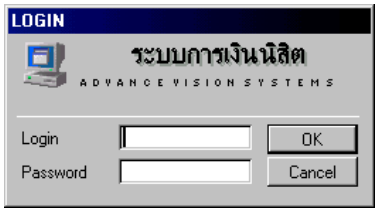
ดังนั้นการพัฒนาระบบของเรา ก็จะขาดในส่วนของ MODULE ที่จะเข้าเข้ามาเป็นตัวเสริมการประมวลผล และควบคุมส่วนต่างๆ ซึ่งเราสามารถใช้ MACRO ได้ด้วยเช่นกัน แต่ตำราเล่มนี้ จะกล่าวถึงเฉพะการพัฒนาโดยใช้ MODULE ซึ่งใช้ภาษา VBA (VISUAL BASIC FOR APPLICATION) ซึ่งสามารถใช้งานกับการพัฒนาซอฟต์แวร์ด้วย VISUAL BASIC หรือ การพัฒนาอื่นๆภายใต้ภาษา BASIC เนื้อหาของการเขียนโปรแกรมด้วยภาษานี้จะอยู่ในบทถัดไป ***VBA หรือ VB for ACCESS จะเป็นส่วนสำคัญที่จะกำหนดความสำเร็จในการใช้ ACCESS เป็น TOOL ในการพัฒนาซอฟต์แวร์***

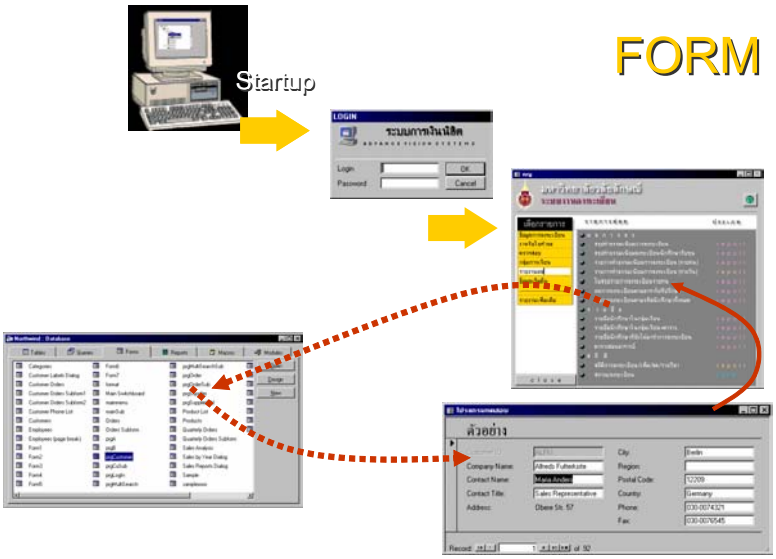
FORM นั้นสำคัญไฉน

การใช้ ACCESS ในแง่ของ TOOL ทางด้าน DATABASE ดูเหมือนว่า FORM ส่วนที่ทำการรับข้อมูลเข้า TABLE แต่สำหรับการพัฒนาซอฟต์แวร์ด้วย ACCESS แล้ว FORM เป็นส่วนที่สำคัญที่สุดในการพัฒนาระบบ ตัวอย่างที่กล่าวถึงนี้เป็นเพียงบางส่วนที่เราใช้งาน FORM เราจะกล่าวถึงโดยละเอียดเมื่อเราเข้าสู่เนื้อหาของ FORM-REPORT ในบทหลังจากนี้

- เราใช้ FORM เป็นหน้าจอที่รับคำสั่งจาก USER (นอกจากข้อมูล) เพื่อที่จะไปเรียก FORM, QUERY , หริอ REPORT อื่นๆ ในลำดับต่อไป หรือจะเรียกว่าเป็น MENU หรือ USER INTERFACE ก็ได้
- FORM เป็นตัวกำหนดเงื่อนไขการออกรายงาน เช่นถ้าเรามีรายงานรายชื่อหนังสือ แต่เราต้องการค้นหาหนังสือที่ขึ้นต้นด้วย “COMPUTER” เราจะไม่ทำ REPORT อีกหนึ่งตัวเพื่อให้ผู้ใช้กำหนด CRITERIA ในการออกรายงาน แต่เราจะใช้ REPORT เดิม แต่กำหนดเงื่อนไขจาก FORM ในขณะเรียกรายงานให้กำหนด Criteria ให้โดยอัตโนมัติ หรือแม้แต่การกำหนด PREVIEW หรือการพิมพ์ทันทีเป็นต้น







- เราใช้ FORM เป็นตัวรับ LOGIN และ PASSWORD ในการเข้าสู่ระบบ เพื่อเราจะได้ตรวจสอบกลไกเงื่อนไขการเข้ามาใช้ระบบ
- เราใช้ FORM ในการทำ SCREEN SAVER หรือคอยตรวจสอบสอบการทำงานบางอย่าง เช่นคอยบันทึกข้อมูลตลอดเวลาลงใน DATABASE เป็นต้น (TIMER)

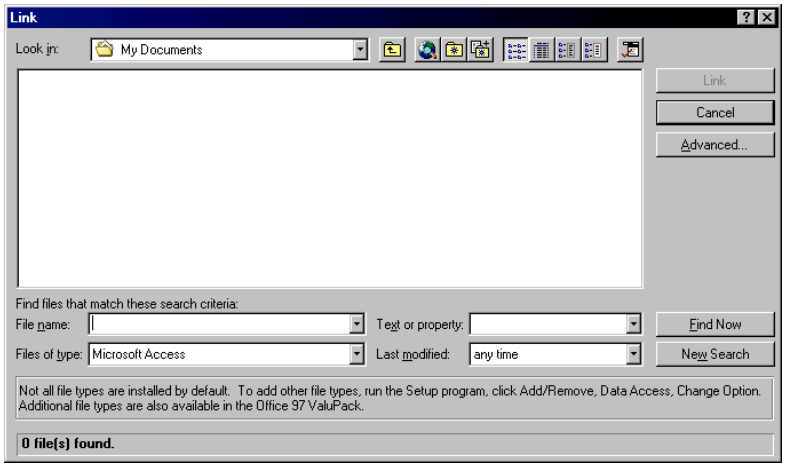
การ ATTACH TABLE

การ ATTACH TABLE เป็นการสั่งให้ ACCESS เข้าไปอ่านข้อมูลใน TABLE จากไฟล์อื่นๆ ที่ไม่ได้อยู่ใน TABLE ของ MDB ที่ทำงานอยู่ การ ATTCH TABLE นอกจากจะใช้ในการอ่าน TABLE ที่อยู่ใน FILE MDB ด้วยกันแล้ว ยังใช้ในการอ่านไฟล์ข้อมูลอื่นๆ ที่ไม่ใช้ MDB ด้วย อาทิ การเชื่อมต่อกับไฟล์ dBF สำหรับระบบที่เขียนด้วย FOXPRO หรือ dBASE หรือการเชื่อมต่อกับ DATABASE SERVER ทำให้เราสามารถพัฒนาระบบที่เชื่อมโยงฐานข้อมูลหลายแบบได้ด้วย Access

การ ATTACH TABLE (MDB)

1. **เลือกคำสั่ง** LINK TABLE **จาก** MENU:File -> Get External data
2. **ระบุประเภทไฟล์ที่** File of type

3. **ระบุ ตำแหน่งที่เก็บไฟล์**
4. **ระบุชื่อ** TABLE

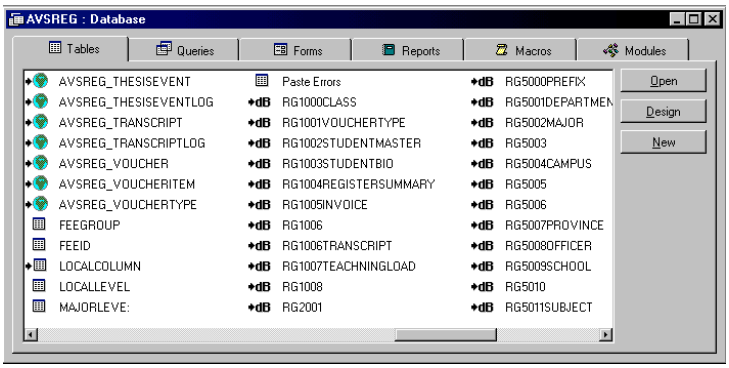


NOTE:
กรณีที่เป็นไฟล์ ที่เป็น ODBC **จะมีขั้นตอนอื่นเพิ่มเติม เช่นการ** LOGIN **เข้า** DATABASE SERVER **สำหรับ** dBF **ไฟล์ จะมีการถามถึง** Index **ไฟล์ที่จะใช้ร่วมกัน**

ลักษณะของ ATTACH TABLE

ประเด็นที่สำคัญของ TABLE ที่เป็น ATTACH TABLE (ไม่ใช่ INTERNAL TABLE) คือ

1. TABLE ที่ ATTACH จะแบ่งเป็น 2 จำพวก คือผ่าน ODBC และไม่ผ่าน ODBC , TABLE ที่ผ่าน ODBC จะแสดงด้วยรูปลูกโลก สำหรับ TABLE อื่นๆ จะแสดงเครื่องหมายตามประเภทของไฟล์ปลายทาง เช่น TABLE ที่เป็น dBASE จะแสดงด้วย dB เป็นต้น



2. TABLE ที่ ATTACH จะไม่สามารถแก้ไขโครงสร้างของ TABLE ได้ จะต้องเข้าไปแก้ที่ไฟล์นั้นๆโดยตรง กรณีที่ข้อมูลที่ ATTACH ไป ไม่เป็น MDB เราจะต้องใช้เครื่องมือของ DATABASE นั้นในการแก้ไข เช่น

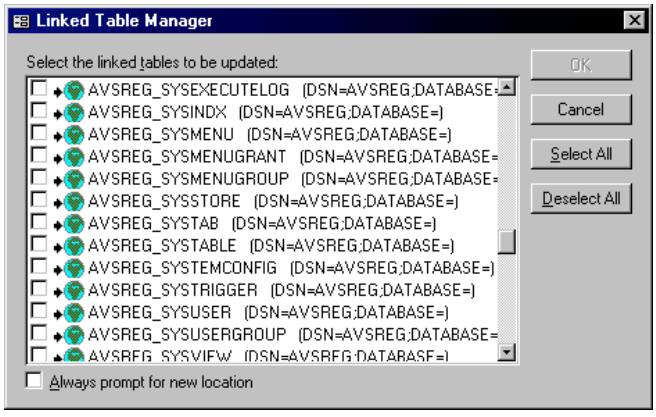
dbf ไฟล์ ก็ต้องใช้โปรแกรม dBASE เป็นตัวแก้ไข TABLE ที่ผ่าน ODBC ต้องตรวจสอบสอบว่า ODBC นั้นเชื่อมไปหา DABASE SERVER อะไร ก็ต้องไปแก้ที่ตัวนั้น เป็นต้น

3. ACCESS จะเก็บเฉพาะโครงสร้าง TABLE, INDEX ไว้ที่ตัว ACCESS เอง การเข้าไปใช้ข้อมูล ACCESS จะอ่านค่าที่เคยเก็บไว้ไปอ่านข้อมูล รวมทั้งตำแหน่งที่เก็บไฟล์ ชื่อไฟล์ ถ้าไฟล์ดังกล่าว เปลี่ยนชื่อไป หรือเปลี่ยน Directory ACCESS จะหาไม่พบ และจะขึ้น Error ดังนั้นตำแหน่ง File จะต้องไว้ที่เดิม และเป็นที่เดิม มิฉนั้น จะต้องทำการ ATTACH ใหม่หรือใช้การ REFRESH TABLE
4. เราสามารถออก REPORT หรือทำ FORM, QUERY กับ TABLE เหล่านี้ เสมือนเป็น TABLE ในระบบได้เหมือน TABLE ที่เป็น INTERNAL
5. TABLE ที่ ATTACH ไว้ ไม่จำเป็นต้องมีชื่อเหมือนกับชื่อ TABLE ที่เป็นฉบับจริงๆ เพราะ ACCESS จะเก็บข้อมูลการ LINK ไปยัง TABLE ดังกล่าวไว้อีกที่หนึ่ง ชื่อ TABLE อาจเป็นชื่ออื่นได้

ตัวอย่างเช่น
เรามี MDB ไฟล์ 2 ไฟล์ที่เป็นข้อมูลรายชื่อหนังสือ ของร้านหนังสือ 2 ร้าน ชื่อ TABLE BOOK เมื่อเรา ATTACH จาก 2 TABLE นี้แล้ว เราอาจตั้งชื่อ TABLE ที่ ATTACH แล้วเป็น BOOKBRACH1 กับ BOOKBRACH2 เป็นต้น

การ REFRESH LINK TABLE

ในบางครั้ง เราต้องการทำการ REFRESH TABLE ที่เคย LINK ไปแล้วใหม่ เราอาจใช้ การ REFRESH TABLE แทนการ ATTACH ใหม่ เนื่องจาก เราอาจมีการบันทึก คำอธิบาย หรือมีการเปลี่ยนชื่อ TABLE ที่ทำการ Link ไปแล้ว หากเราลบการ LINK ของ TABLE ทิ้ง และทำการ Attach ใหม่ เราจะต้องทำขั้นตอนเดิมซ้ำ การ REFRESH TABLE จะทำการปรับปรุง ตำแหน่งที่เก็บไฟล์ ชื่อไฟล์ ขนาด COLUMN ชื่อ COLUMN INDEX, KEY ของเดิมที่เก็บไว้ใน ACCESS ให้ตรงกับ TABLE ปัจจุบัน

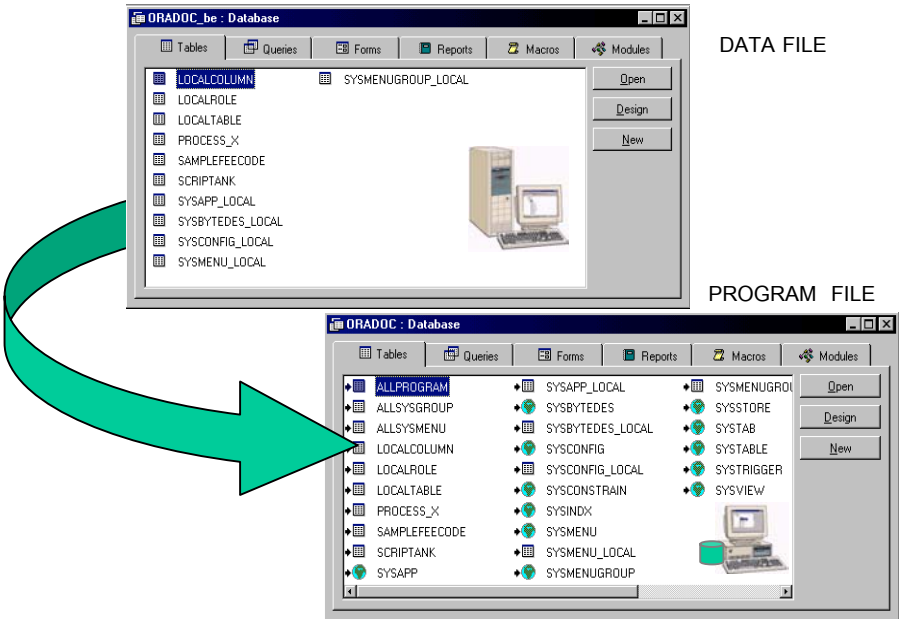


การ REFRESH ATTACH TABLE

1. เลือก Linked Table Manager จาก Menu : Tool, Add-Ins

2. Click เลือกชื่อ Table ที่ต้องการ Refresh
3. Click OK , หาก ACCESS หาตำแหน่งชื่อไฟล์เดิมไม่พบ ACCESS จะถามชื่อไฟล์จากเรา

การแยก TABLE จากโปรแกรม



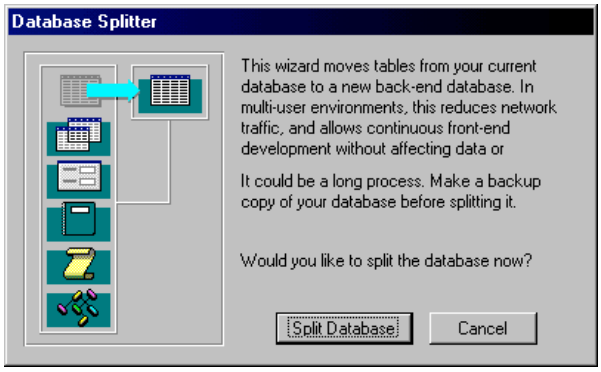
ดังที่เราทราบแล้วว่า FORM, REPORT, QUERY, MODULE เป็นส่วนที่จัดว่าเป็น INPUT, OUTPUT, PROCESS แต่ TABLE เป็นส่วนที่เป็นเนื้อข้อมูลจริงๆ ถ้าเรานำระบบไปติดตั้งให้กับผู้ใช้ ซึ่งเป็น FILE ที่มีทั้งโปรแกรม และข้อมูลอยู่ด้วยกัน สำหรับการลงโปรแกรมครั้งแรกคงไม่มีปัญหาอะไร แต่หากว่าคุณมีการแก้ไขโปรแกรม เช่นแก้ไข FORM บาง FORM เพิ่ม REPORT สัก 2, 3 ตัว ถ้าเรานำ ไฟล์ที่แก้ไขแล้วไปทับ ข้อมูลที่ผู้ใช้ใช้อยู่ซึ่งเคยมีการบันทึกเข้าไประหว่างที่เราทำการแก้ไขโปรแกรม ก็จะหายหมด ทางที่ดีเราควรแยก 2 องค์ประกอบนี้ออกจากกัน การติดตั้งการแก้ไขก็จะง่ายขึ้น นอกจากประเด็นดังกล่าวแล้วเราจะพบว่า

ขนาดของไฟล์ ส่วนที่มีขนาดใหญ่คือข้อมูล และโตขึ้นตามการเก็บข้อมูลของผู้ใช้ เมื่อเราแยกข้อมูล และโปรแกรมจากกันแล้วขนาดไฟล์ที่เป็นโปรแกรมก็จะเล็กลง และมีขนาดคงที่ จัดการได้ง่ายขึ้น

การทำการ BACKUP ข้อมูลที่แยกเป็นไฟล์ ๆ สามารถทำการจัดเก็บสำรองได้ง่าย เนื่องจากไฟล์ข้อมูลได้ถูกแยกออกจากกัน การ COPY, BACKUP ไฟล์ในแต่ละวันก็เพียงแต่ COPY ไฟล์ข้อมูลนี้ไปไว้ในที่พักสำรอง และหากต้องการนำข้อมูลย้อนหลังมาใช้งาน ก็เพียงแต่ COPY กลับมาทับที่ไฟล์เดิมเท่านั้น

การแยกโปรแกรมจาก TABLE โดย DATABASE SPLITER

1. เรียก MDB ที่ต้องการ
2. เลือก DATABASE SPLITER ที่ MENU: Tool, Add-Ins
3. CLICK OK
4. โปรแกรมจะถามชื่อไฟล์ที่จะใช้ไว้เก็บ TABLE อย่างเดียว ระบุชื่อ และตำแหน่งที่เก็บ DIRECTORY

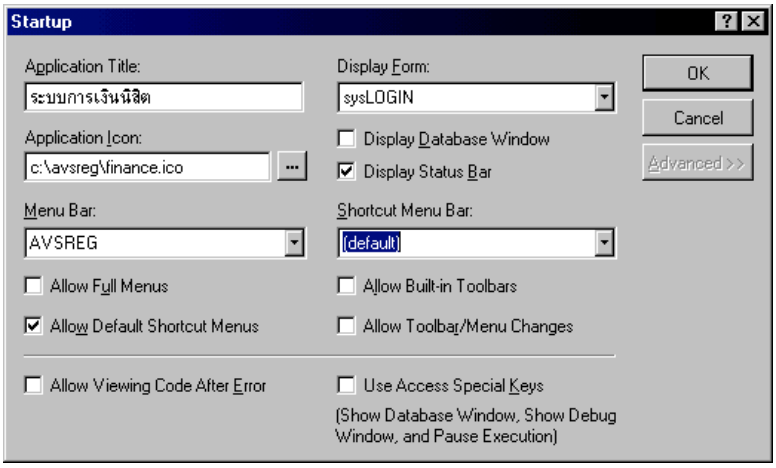


การแยกโปรแกรมจาก TABLE ด้วยตัวเอง

1. COPY ไฟล์ MDB ที่ต้องการแยกไปไว้อีกชื่อหนึ่ง ไว้ใน Directory ที่ต้องการเป็นทีเก็บ DATA
2. เปิดไฟล์ที่เป็นต้นฉบับบเดิม แล้วลบ TABLE ทุก TABLE ทิ้ง
3. ATTACH TABLE ไปยังไฟล์ที่ COPY ไว้ในหัวข้อ 1
4. เปิดไฟล์ข้อมูลที่ COPY ไว้ แล้วลบ FORM, REPORT, QUERY, MODULE, MACRO ในไฟล์ที่ COPY ไว้ทิ้งทั้งหมด

การกำหนด STARTUP ของ ACCESS

เราสามารถกำหนดให้ ACCESS ทำการเปิด FORM ใด FORM หนึ่งขณะเริ่มต้นเรียกไฟล์ MDB ที่เป็นโปรแกรมของเราได้ทันที มักกำหนดให้ FORM ที่เป็นการ LOGIN หรือ MAIN MENU เป็น FORM ที่ STARTUP เพื่อนำผู้ใช้เข้าสู่ระบบงานตามขั้นตอน นอกจากนี้เรายังกำหนดข้อกำหนดในการเปิดไฟล์ได้อีกหลายส่วนดังตารางแสดงไว้ด้านล่าง



การกำหนด STARTUP

1. เลือก STARTUP จาก MENU:TOOL
2. กำหนดชื่อ FORM ลงในช่อง Display Form
3. กำหนดชื่อ Application ลงใน Application Title ชื่อนี้จะแสดงที่ Task Bar และที่ Windows ของ Access แทนคำว่า Microsoft Access
4. กำหนดไฟล์ที่เป็น ICON หรือ รูปที่แสดงบน Task Bar ของ Windows แทนรูปกุญแจ

สำหรับข้อกำหนดอื่นที่ระบุได้คือ

หัวข้อ	คำอธิบาย	ควร
Menu Bar	ระบุชื่อ Menu Bar ที่ต้องการใช้แทน Menu Bar ปกติของ Access	สร้าง / กำหนด
Shortcut Menu Bar	ระบุชื่อ Shortcut Menu Bar ที่ต้องการใช้แทน Menu Bar ปกติของ Access	Default
Allow Full Menus	เลือกให้ User สามารถใช้ Menu ทุกตัวของ Access สำหรับการพัฒนาโปรแกรมแล้ว เราไม่ควรเลือก แล้วกำหนด Menu Bar ของเราเอง	ไม่ CLICK

หัวข้อ	คำอธิบาย	ควร
Allow Built-in Toolbars	อนุญาติให้ User ใช้ Toolbar ในโปรแกรมหรือไม่ (ไม่ควร)	ไม่ CLICK
Allow Default Shortcut Menus	อนุญาติให้ User ใช้ Shortcut Menu ที่ Access ทำไว้หรือไม่	Click
Allow Toolbar/Menu Change	อนุญาติให้ User ทำการเปลี่ยนแปลงแก้ไข Menu หรือ Toolbar หรือไม่ (ไม่ควร, ให้เป็นไปตามที่กำหนดไว้ใน Menu Bar, Shortcut Menu bar)	ไม่ CLICK
Display Database Windows	อนุญาติให้สามารถแสดง Database Windows ได้หรือไม่	ไม่ CLICK
Display Status bar	ให้มีการแสดง Status bar หรือไม่	CLICK
Allow Viewing Code After Error	ให้ Access แสดงข้อความ Error จาก VB Code เมื่อเกิด Error หรือไม่	ไม่ CLICK
Use Access Special key	ให้สามารถใช้ปุ่มพิเศษที่ใช้ในการหยุดการคำนวณ คำสั่ง F11 เพื่อดู Database หรือไม่	ไม่ CLICK

NOTE:
หากต้องเข้าโปรแกรมโดยยกเลิกเงื่อนไขที่กำหนด ให้กดปุ่ม SHIFT ที่ Keyboard ค้างไว้ ขณะเรียกไฟล์ MDB ที่กำลังทำการ, Access จะข้ามเงื่อนไขที่กำหนดใน Startup

ขั้นตอนการพัฒนาระบบด้วย ACCESS

ขั้นตอนการออกแบบระบบที่พัฒนาด้วย ACCESS หรือจะเป็นระบบที่ใช้ TOOL อื่นๆทาง DATABASE มีแนวทางพื้นฐานคือ

- **ขั้นตอน**

1	ออกแบบ TABLE ให้ดี	จุดเริ่มต้นของความสำเร็จคือการออกแบบฐานข้อมูลที่ดี สร้าง RELATIONSHIP กำหนดค่า DEFAULT ที่เหมาะสม ทำบนกระดาษก่อนก็ได้ แล้วจึงนำไปสร้าง TABLE จริง อย่าลืมทดลองกรอกข้อมูลลงไปใน TABLE ตรงๆ ดูว่ามีจุดใดไม่เหมาะสม ติดขัดอะไรบ้าง แล้วแก้ไขให้ดีที่สุดก่อนข้ามไปขั้นตอนต่อไป
2	ออกแบบ FORM	ทำ FORM ที่จะใช้ในการกรอกข้อมูลทุก TABLE พยายาม ตรวจสอบ ว่ามี TABLE ใดยังไม่มี FORM ใดเลยที่ไปทำการเขียนข้อมูล
3	ออกแบบ REPORT	สร้างรายงานจากฐานข้อมูลที่มี อาจทำไปพร้อมๆ กับการทำ FORM ที่เกี่ยวข้องกับข้อมูลนั้นๆ
4	เพิ่มความคล่องตัวของ FORM	เริ่มทำ FUNCTION ที่เข้ามาช่วยให้ FORM หรือ รายงานของเราทำงานได้ดีขึ้น เช่น มีปุ่มค้นหา ปุ่มออกรายงานจาก FORM ตอนนี้เราต้องใช้ ความรู้ของการเขียนโปรแกรมด้วย VB
5	รวบรวมหน้าจอด้วย MENU	ทำ MENU ที่จะเชื่อมไปหา FORM หรือ REPORT ต่างๆที่ เราได้สร้างไว้เพื่อให้ผู้ใช้สะดวกในการใช้งาน
6	กำหนด STARTUP FORM	กำหนด STARTUP FORM ที่จะเป็นหน้าแรกของระบบงานของเรา อาจเป็นหน้าจอ FORM ที่เป็น MENU หรือหน้าจออื่นๆ
7	แยก โปรแกรมกับข้อมูล	แยก TABLE ออกจากระบบที่เราพัฒนา กำหนดตำแหน่งที่เก็บ FILE ที่เป็นข้อมูล โปแกรม เพื่อให้การ ATTACH TABLE ไปถึง
8	ทำ MDE	COMPILE โปรแกรมให้เป็น MDE (MDB –> MDE) เพื่อไม่ให้ผู้ใช้เห็น SOUREC CODE โปรแกรมของเรา รวมทั้งการแก้ไข FORM, REPORT

C H A P T E R 2

VISUAL BASIC สำหรับ ACCESS

ในบทนี้จะเป็นเนื้อหาที่เกี่ยวกับภาษา BASIC ตระกูล VISUAL BASIC ที่ใช้งานกับ ACCESS เพื่อการเขียนโปรแกรมใช้งานในตัว ACCESS เอง เนื้อหาส่วนนี้จะเป็นองค์ประกอบที่สำคัญที่สุดในการที่จะพัฒนาระบบงานด้วย Access ให้สัมฤทธิ์ผล และเป็นแนวทางในการเรียนรู้ภาษา Basic ของค่าย Microsoft ทั้งที่อยู่ใน VISUAL BASIC, Active Server Page หรือ Product อื่นๆในอนาคต

ในเนื้อหาส่วนนี้จะค่อนข้างยากสำหรับผู้ที่ไม่มีพื้นฐานทางด้านคอมพิวเตอร์มาก่อน ในที่นี้จะกล่าวถึงแนวความคิดพื้นฐานทางด้าน Programming เสริม สำหรับผู้ที่ไม่เคยมีประสพการณ์ด้าน Programming มาก่อนด้วย

CONTENT

- FIRSTAPP : ID = ID +1
- รู้จักเครื่องมือในการเขียน
- VISUAL BASIC เบื้องต้น
- การควบคุมการประมวลผล
- การเปรียบเทียบ เงื่อนไข
- การกำหนดตัวแปร
- Function ใน Access
- การสร้าง FUNCTION, SUBROUTINE
- การควบคุม ERROR

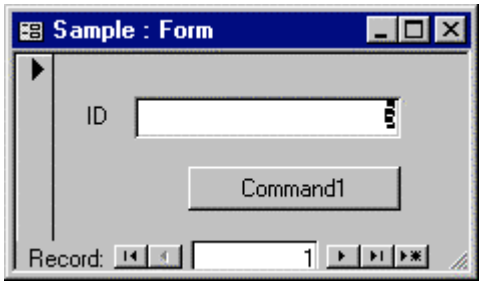
FIRSTAPP: ID = ID + 1

VISUAL BASIC สำหรับ ACCESS เป็น ภาษา BASIC แบบเดียวกับที่ใช้ใน VISUAL BASIC มีข้อแตกต่างบางประการที่ไม่เหมือนกับตัว VISUAL BASIC เอง การเรียนรู้ภาษา VISUAL BASIC จะประกอบด้วยการเรียนรู้ 2 ส่วนคือ การเรียนรู้ภาษา VISUAL BASIC เอง และเรียนรู้ OBJECT ใน ACCESS กล่าวได้ว่าการ PROGRAMMING ใน ACCESS ก็คือการนำเอาภาษา VISUAL BASICไปควบคุม OBJECT ต่างๆ ของ ACCESS นั่นเอง

เพื่อให้เข้าใจถึงองค์ประกอบทั้ง 2 ส่วน เริ่มต้นด้วยการทำ PROGRAM ที่ทำการเพิ่มค่าใน TABLE เมื่อมีการกดปุ่ม

- FIRST APP

1. สร้าง TABLE 1 TABLE ชื่อ INDEXCOUNT มี FIELD 1 FIELD ชื่อ ID เป็น Long Integer
2. เพิ่มข้อมูลเข้าไปใน Table นี้ 1 Record โดยมีค่า ID = 1
3. สร้าง FORM ใหม่ 1 FORM
4. เลือก Record Source ให้เป็น INDEXCOUNT
5. ลาก FIELD ID มาลงที่ FORM
6. สร้างปุ่ม 1 ปุ่ม (CANCEL WIZARD ถ้ามี)
7. Click ด้วย Mouse ปุ่มขวา แล้วเลือก Build Event ตามด้วย Code Builder จะปรากฏหน้าจอตามภาพ



8. ป้อนคำสั่งเข้าไประหว่างบรรทัด Procedure และ End Sub

```
ID = ID + 1
```

9. กลับไปหา FORM เดิม แล้วเรียก FORM ให้ทำการ (VIEW FORM ใน Run-time Mode)
10. CLICK ที่ปุ่ม จะปรากฏว่าตัวเลขของ ID เพิ่มขึ้นที่ละหนึ่ง

ส่วนที่เราบันทึกเข้าไปคือ **การเขียนโปรแกรม** โดยสั่งให้ OBJECT ที่เป็น TEXTBOX CONTROL มีค่าเท่ากับค่าเดิมบวกอีกหนึ่ง เครื่องหมายเท่ากับในที่นี้ไม่ได้แปลว่า “เท่ากับ” แต่หมายถึงการกำหนดค่าลงไปใน ตัวแปร ,OBJECT หรือ อะไรก็ตามที่อยู่ด้านซ้ายมือ ด้วยค่าที่อยู่ทางด้านขวามือ คำสั่งนี้ เป็นคำสั่งในการกำหนดค่า เราจะต้องทราบว่าคำสั่งในภาษา Basic มีอะไรบ้าง แล้วจึงนำคำสั่งนั้นๆ เป็นเครื่องมือในการติดต่อกับ OBJECT บน Access นั่นเอง

คำสั่งนี้จะทำการเมื่อไร ?

ขณะที่เราเปิด Form ใน Mode ปกติ เราเรียกว่าว่า Run-Time Mode ส่วนในขณะที่ออกแบบเราเรียกว่า Design Mode เมื่อ Form อยู่ใน Run-Time Mode , ACCESS จะคอยตรวจสอบการทำการใดๆบน Form ของเรา แล้วจะทำการเรียกคำสั่งที่เราเพิ่งเขียนเข้าไปนี้ เมื่อมีการ Click ที่ปุ่ม Command1

ACCESS ทราบได้อย่างไรว่าจำต้องทำคำสั่งใด ที่ขณะใด ? ให้ลองสังเกตุดูว่า คำสั่งที่เรา กรอกอยู่ระหว่าง Private Sub Command1_Click และ End Sub ตรงนี้เองเราเรียกมันว่า Sub Routine หรือโปรแกรมย่อย สำหรับโปรแกรมย่อยนี้มีว่า `Command1_Click` ก็หมายถึง โปรแกรมที่ทำงานเมื่อ OBJECT ชือ Command1 มีการ Click นั่นเอง การ Click เราเรียกว่า Event ถ้ามีหลายๆ Object ที่มีการฝัง โปรแกรมไว้ Access ก็จะดูชื่อ Object ตามด้วยชื่อ Event เพื่อเรียกคำสั่งที่เราแทรกไว้ออกมาทำการนั่นเอง

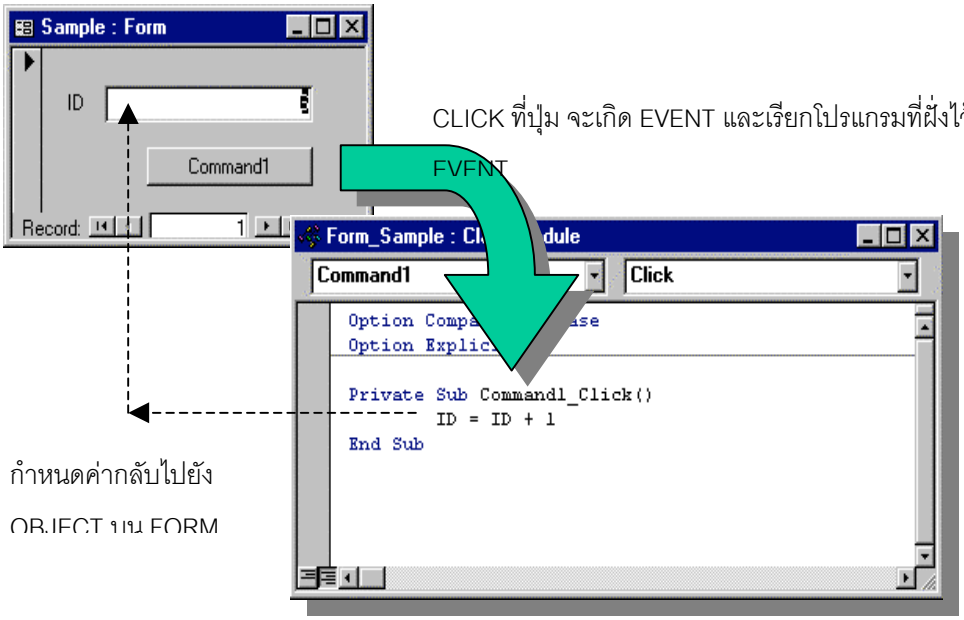
เราทดลองขั้นตอนต่อไป โดยการสร้างปุ่มอีกหนึ่งปุ่ม ใช้ขั้นตอน 6,7,8 แต่ในขั้นตอนที่ 8 ให้กรอกว่า ID = ID – 1 เมื่อเรา Click ปุ่มนี้ ค่าของ ID ก็จะลดลงที่ละหนึ่ง ตอนนี้คำสั่งทั้งหมดจะเป็นดังนี้

```
Option Compare Database
Option Explicit
Private Sub Command1_Click()
        ID = ID + 1
End Sub
Private Sub Command2_Click()
        ID = ID - 1
End Sub
```

วิธีการเรียกดู Code ดังกล่าว ทำได้โดยการเลือก Object แล้วเลือก Build Event แบบที่เคยทำ หรือ จะใช้คำสั่ง Code ใน Menu View หรือเรียกจาก Tool Bar โดยตรงก็ได้ 

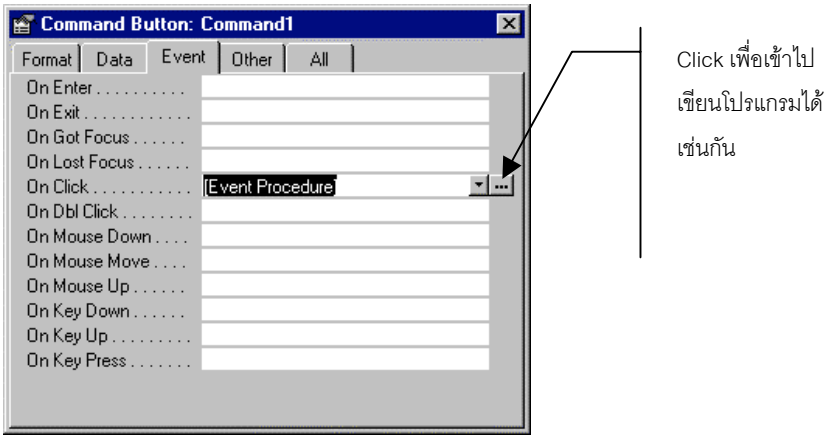
Event Sub Rountine

คงจะทราบแล้วว่า Access จะเข้ามาเรียกโปรแกรมที่เราเขียนไว้ โดยโปรแกรมที่เราเขียนไว้ จะถูกจัดแยกเป็นสัดส่วน ตาม EVENT ที่เราต้องการให้โปรแกรมนั้นๆทำการ การจัดโปรแกรมดังกล่าวก็คือการจัดโปรแกรมให้เป็น Sub Routine เพื่อให้ Event ไปเรียกใช้ โดยมีข้อกำหนดว่า ใช้ชื่อ Object ตามด้วย Event เราเรียก Sub ประเภทนี้ว่า Event Sub Routine



Control นี้มี Event อะไรบ้าง

เราคงอยากทราบแล้วว่า Control แต่ละตัวมี Event อะไรบ้าง ? วิธีที่ง่ายที่สุด คือดูที่ Properties ของ Control นั้นๆ จะมีรายการ Event แทรกอยู่ด้านล่างสุด หรือเรียก Tab กลุ่ม Event เลยก็ได้ ถ้า Event ใดมีการเขียนโปรแกรมแทรกแล้ว จะปรากฏคำว่า [Event Procedure]

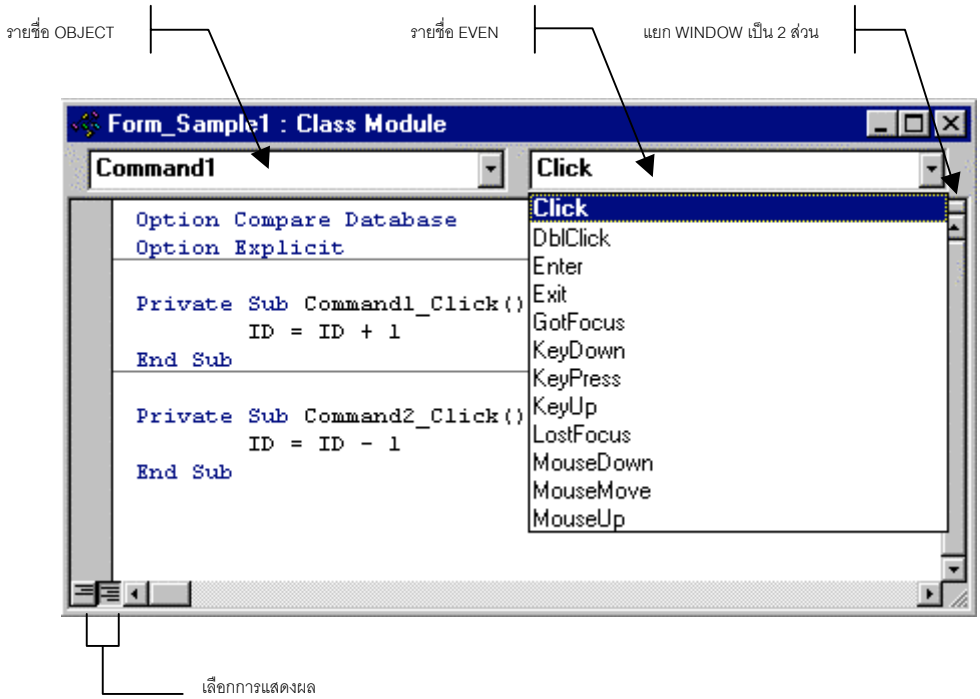


อีกวิธีหนึ่งคือ เข้าที่หน้าที่เขียนโปรแกรม (CLASS MODULE) เลือก Control จาก Combo Box ตัวแรก จากนั้น Combo Box ด้านซ้ายจะแสดงรายการ Event ที่มีของ Control ตัวนั้น Event ตัวใดมีการเขียนโปรแกรมลงไปแล้ว ตัวอักษรที่แสดง จะเป็นตัวเข้มที่ชื่อ Event

รู้จักเครื่องมือในการเขียน

เรามาทำความรู้จักกับเครื่องมือในการเขียนโปรแกรมของเรากันก่อนที่จะเริ่มเขียนโปรแกรม

CODE WINDOW

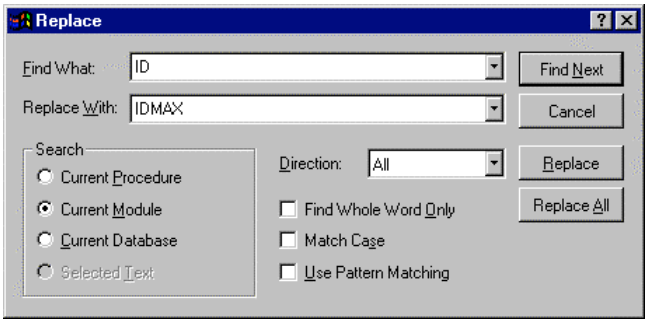


เป็นหน้าจอที่เราใช้ในการเขียนโปรแกรม จะมี COMBO BOX ปรากฏอยู่ 2 ส่วนคือส่วนที่แสดงรายชื่อ Control กับส่วนที่ใช้แสดงรายชื่อ Event ของ Control ขณะที่เรา Click เลือก Control หรือ Event , Cursor จะย้ายไปที่ Sub Routine ที่กำลังชี้อยู่ทันที

ส่วนของการเลือกการแสดงผล จะใช้ Click เลือกเพื่อแสดงเฉพาะ Sub Routine ที่กำลังทำการ หรือแสดงทั้งหมดอย่างที่แสดงในภาพ

เราสามารถแยกหน้าจอที่ทำการบันทึกออกเป็น 2 ส่วนได้โดยการ กดแล้วลาก ตำแหน่งที่แสดงในภาพ (DRAG) หน้าจอจะแบ่งเป็น 2 ส่วนเพื่อให้การเขียนโปรแกรมสะดวกขึ้น

การใช้ FIND- REPLACE



บ่อยครั้ง ที่การเขียนโปรแกรม มีการเปลี่ยนชื่อตัว Object เช่น Field ID ในตัวอย่างของเรา หากมีการแก้ไขเป็นชื่อ IDMAX เราจะต้องแก้ไขทุกส่วนใน Program ที่เคยมีการใช้คำว่า ID เป็น IDMAX เพราะไม่มี ID อีกต่อไปแล้ว วิธีที่ง่ายที่สุดคือการใช้ Find / Replace เลือกคำสั่ง Replace จาก Menu Edit เราสามารถกำหนด Scope ในการทำการได้ทั้งใน Sub Routine นี้เท่านั้น (Current Procedure), ทั้ง Form (Current Module), หรือทั้ง Database (Current Database)

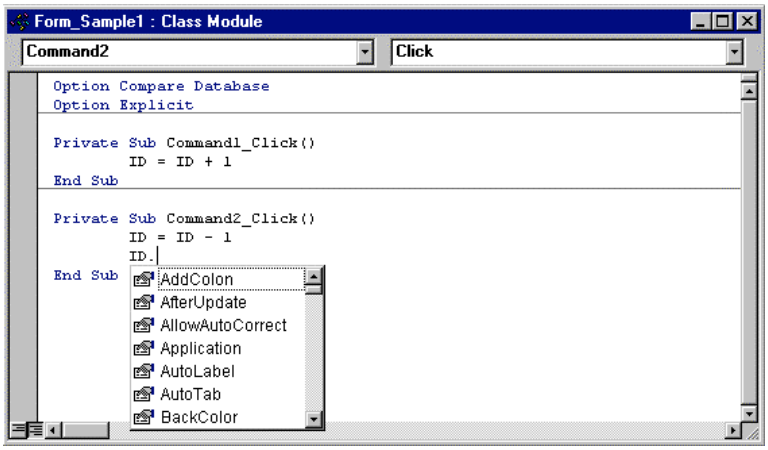
หากต้องการค้นหาแต่ไม่เปลี่ยนค่า ให้ใช้คำสั่ง Find

การ COPY, CUT, PASTE

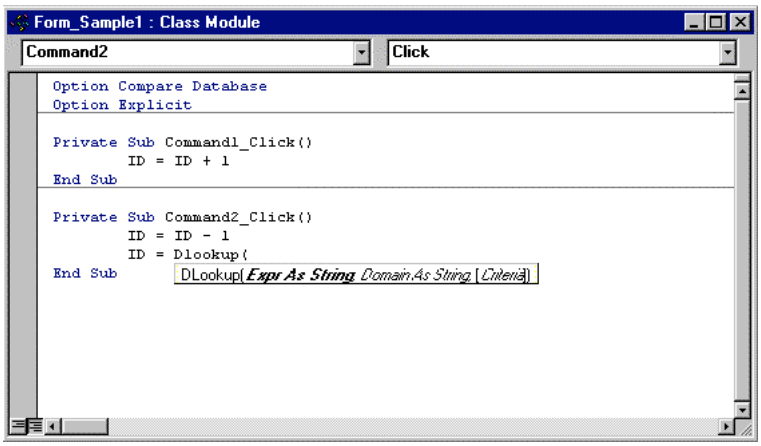
การเขียนโปรแกรมคงหนีไม่พ้นการ COPY โปรแกรมจากจุดหนึ่งไปยังจุดหนึ่ง พยายามใช้การ Copy Cut Paste ให้ชิน โดยเฉพาะการใช้ Hot Key (Ctrl+C= COPY, Ctrl + V = PASTE, Ctrl+X = Cut)

การใช้ Autometic Code

เมื่อเราเริ่มเขียนโปรแกรมคำสั่งแรก จะพบว่า VB สามารถแนะนำ คำสั่งที่จะใช้เขียนถัดไปได้ โดยแสดงเป็น List Box ของ Syntax ที่จะเป็นไปได้ให้เราดู ซึ่ง VB ทำได้ 2 อย่างคือ



- List สิ่งที่เกี่ยวกับ Object นั้นหลังจากที่เราบันทึกชื่อ Object นั้นเสร็จแล้ว เช่นเมื่อเราพิมพ์คำว่า ID. จะปรากฏรายการ Properties, Method, event ทั้งหมดของ Object ออกมา เราเพียงแต่เลื่อน Key ลูกศรลงมายังตัวที่ต้องการ แล้ว เคาะ SpaceBar ก็จะได้ชื่อรายการดังกล่าว
- กรณีที่เป็น Function หรือ Statement (คำสั่ง) VB จะบอกให้ทราบว่า มีค่าตัวแปรอะไรบ้างที่เราต้องส่งผ่าน เช่น การใช้คำสั่ง Dlookup เป็นต้น การใช้สิ่งที่ VB บอกมา ให้สังเกตุว่า ถ้าตัวแปรที่คลุมด้วยเครื่องหมายก้ามปู จะหมายถึงตัวแปรที่เป็น Option ใส่ก็ได้ไม่ใส่ก็ได้



การเรียก Help

เราสามารถขอความช่วยเหลือสำหรับคำสั่งต่างๆได้โดยการกด F1 ถ้าเราสงสัยคำสั่งใด ให้พิมพ์คำสั่งนั้นลงไป แล้วใช้ Mouse เลือกคลุมคำสั่งนั้น แล้วกด F1 VB จะอธิบายให้ทราบโดยละเอียดถึงคำสั่งดังกล่าว

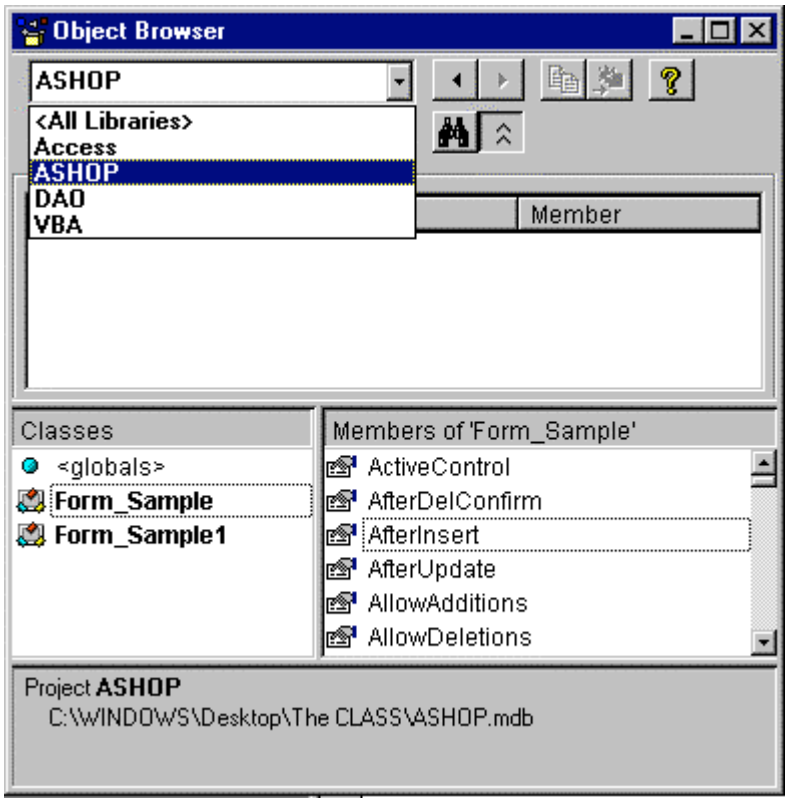
การใช้ OBJECT BROWSER

มีคำสั่งมากมาย และ OBJECT มากมายที่ Programmer ไม่สามารถจำได้หมดทุกตัว VB จะมี เครื่องมือในการค้นหา Object และ องค์ประกอบของมัน ซึ่งอยู่ใน Menu View ชื่อ Object Browser

Object Browser จะใช้เลือกดูว่า มี Object อะไรอยู่บ้างในระบบ Object Browser จะจัดแบ่งกลุ่มของ Object เป็น 4 หมวดคือ

- **หมวด Form Report** หรือ Object ที่เปิดอยู่ในปัจจุบัน การ Object ที่แสดงก็จะเป็นรายการ Object ที่อยู่ภายใต้ กลุ่ม Object นี้อีกครั้งหนึ่ง
- **Access** เป็นกลุ่ม Object ที่อยู่ภายใต้ตัว Access เอง

- VBA เป็นกลุ่ม คำสั่ง Visual Basic พื้นฐาน
- DAO เป็นกลุ่ม Object ที่เกี่ยวข้องกับการจัดการบน Database เราจะได้กล่าวถึงในบทถัดไป



VISUAL BASIC เบื้องต้น

เราเริ่มต้นรู้จักภาษา VB สำหรับ ACCESS แล้ว คราวนี้เราจะมาลงในรายละเอียดของตัวภาษากัน

MODULE

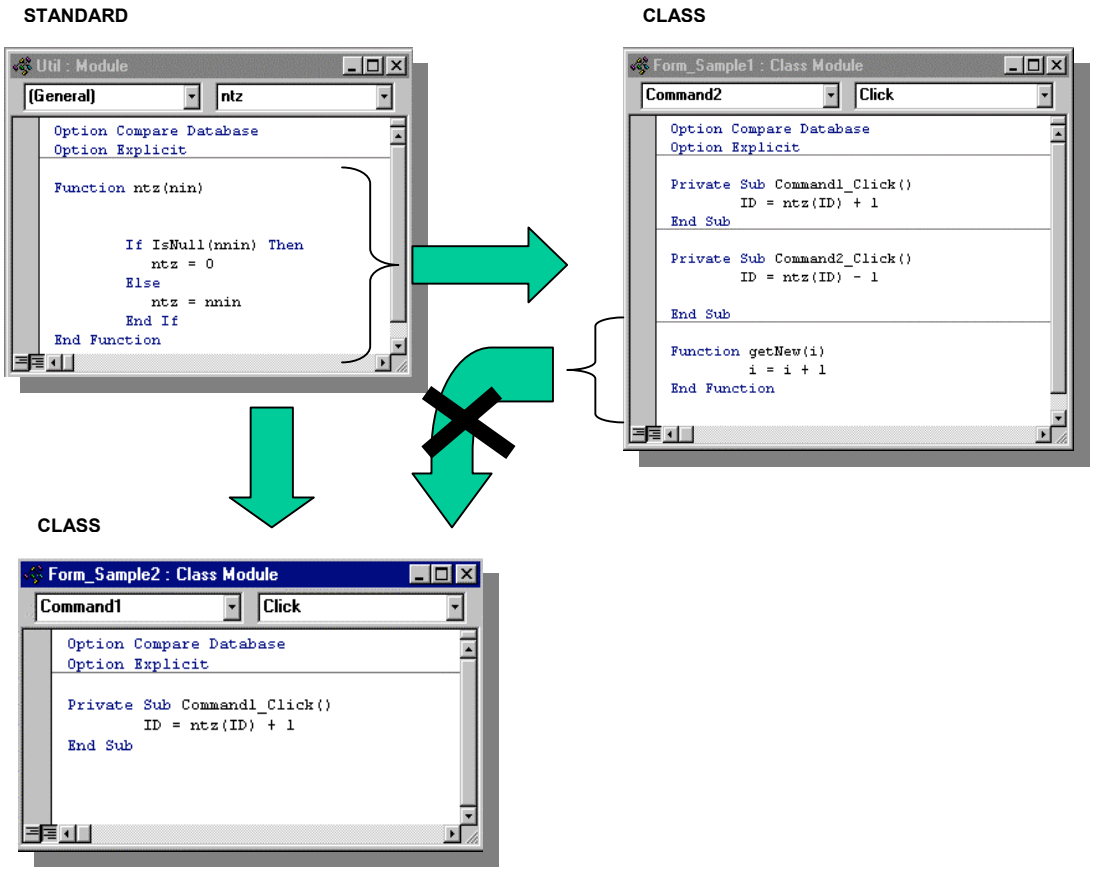
เวลาที่เราเขียน CODE ลงใน ACCESS ทุกครั้ง เรากำลังทำงานอยู่กับ Module , Module เป็นที่เก็บโปรแกรม หรือ Source Code ของโปรแกรม ถูกจัดวางแยกไว้เป็นส่วนๆ สิ่งที่ประกอบอยู่ใน Module คือ

- **ส่วน Declare** สำหรับประกาศตัวแปร และกำหนด Option พิเศษของ Module นั้นๆ จะอยู่ส่วนบนสุดของ Module
- **ส่วน Event โปรแกรม** หรือเรียกว่า Event Procedure เป็นส่วนที่เก็บโปรแกรมที่ทำงานเมื่อมี Event ต่างๆ บน Form หรือ Report ทำงาน โปรแกรม หรือ Procedure เหล่านี้ จะถูกฝังไว้ใน Form หรือ Report ใช้ชื่อตาม Object นั้นตามด้วย "_" และชื่อของ Event เช่น Form_Click เป็นต้น
- **ส่วน โปรแกรมทั่วไป** เป็นโปรแกรมที่ฝังไว้ใน Module โดยที่ไม่อิงกับ Event ของ Object ปกติ เรามีไว้เพื่อให้โปรแกรมที่อยู่ใน Event มาเรียกใช้อีกครั้งหนึ่ง มี 2 ประเภทคือ Sub กับ Function

เราสามารถจัดแบ่ง Module ได้เป็น 2 แบบคือ

- **Standard Module** เป็น Module ที่ไม่มีโปรแกรมประเภท Event Procedure เก็บอยู่ ใช้เก็บเฉพาะส่วน Declaration และ โปรแกรมทั่วไปเท่านั้น เราสร้าง Standard Module โดยการเรียกคำสั่ง New ใน Tab Module ของ Database Windows, Sub หรือฟังก์ชั่นที่มีใน Standard Module สามารถเรียกใช้จาก Module อื่นๆ หรือเรียกจาก Expression ใน Report , Form หรือแม้กระทั่ง Query ก็ได้
- **Class Module** เป็น Module ที่เก็บโปรแกรมสำหรับ กลุ่ม Object หนึ่งๆ เช่น Form หรือ Report ซึ่งเป็นกลุ่ม Object เป็นต้น โปรแกรมที่เราเขียนในตัวอย่างแรก จัดว่าเป็น Class Module ชนิดหนึ่ง Class Module ที่สร้างขึ้นจาก Form และ Report จะผูกติดกับ Form และ Report นั้นๆไปตลอด ไม่สามารถเรียกใช้งานข้ามกันได้

  สำหรับ Class Module อีกประเภทหนึ่ง จะเป็น Class Module ที่ไม่มี Object Form หรือ Report มาเกี่ยวข้อง จะสร้างจากคำสั่ง Class Module ใน Menu Insert Class Module ประเภทนี้จะคล้ายกับ Standard Module มากกว่า แต่การเรียกใช้ Sub Rountine หรือ ค่าตัวแปร จะใช้คำสั่งที่เป็นลักษณะ Object Oriented เราต้องสร้าง Properties, Method ของ Class ขึ้นเอง เราจะใช้ Class Module ประเภทนี้น้อย Feature นี้ถูกสร้างมาให้ Developer ที่คุ้นเคยกับการพัฒนาโปรแกรมแบบ Object Oriented Programming สำหรับผู้เขียนคิดว่า ในช่วงเริ่มต้นของการเรียนรู้ อาจทำให้สับสนได้ และ Feature ดังกล่าวไม่มีใน Version เก่าของ Access ในหนังสือเล่มนี้ จะข้ามส่วนดังกล่าวไป



ในทางปฏิบัติ เราจะทำงานกับ Class Module ที่ฝั่งบน Form Report เป็นส่วนใหญ่ และ ทำงานกับ Standard Module สำหรับโปรแกรมที่ต้องมีการใช้ร่วมกันบ่อย ยกตัวอย่างเช่น โปรแกรม แปลงตัวเลข เป็น ภาษาไทย (100 เป็น “หนึ่งร้อยบาท”) เราจะไม่ฝั่งโปรแกรมนี้ไว้ที่ Form หรือ Report แต่จะเก็บไว้ที่ Standard Module แทน ซึ่งทุกๆ Module สามารถเรียกใช้ โปรแกรมที่อยู่ใน Module นี้ได้

การสร้าง Standard Module

1. เรียก Database Windows (กด F11)
2. เลือก Tab Module
3. กด New
4. Save และตั้งชื่อที่ต้องการ

สร้าง Sub Routine แรก

สำหรับ Sub ที่เป็น Event Procedure ใช้วิธีทำจาก Form หรือ Report แต่กรณีที่เป็น Sub ที่เราสร้างขึ้นใช้เอง ไม่ว่าจะสร้างไว้ที่ Standard Module หรือที่ Class Module (ใต้ Form หรือ Report) ให้ทำดังนี้

การสร้าง Sub

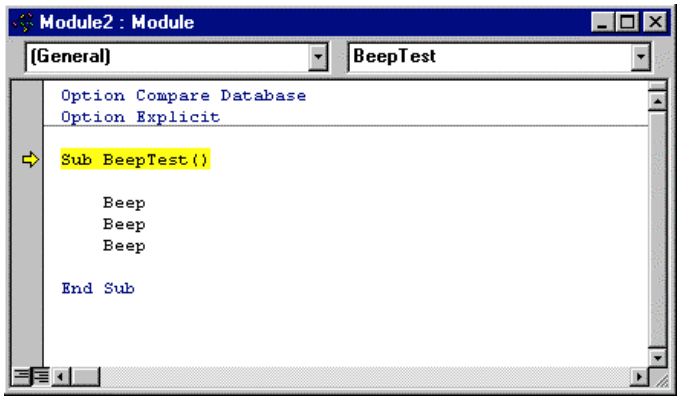
1. หาตำแหน่งว่างใน Code Windows ที่ไม่อยู่ระหว่างกลางของ Sub กับ End sub คู่ใดคู่หนึ่ง
2. พิมพ์คำว่า Sub แล้วตามด้วยชื่อ ฟังก์ชั่น แล้วเคาะปุ่ม Enter, Access จะใส่ End Sub ให้เอง

ทดลองสร้าง Sub สำหรับการส่งเสียง Beep 3 ครั้งติดกัน ตามโปรแกรมตัวอย่าง โดยการสร้าง Standard Module ขึ้นมาใหม่

```
Sub BeepTest
    Beep
    Beep
    Beep
End Sub
```

การ RUN PROGRAM

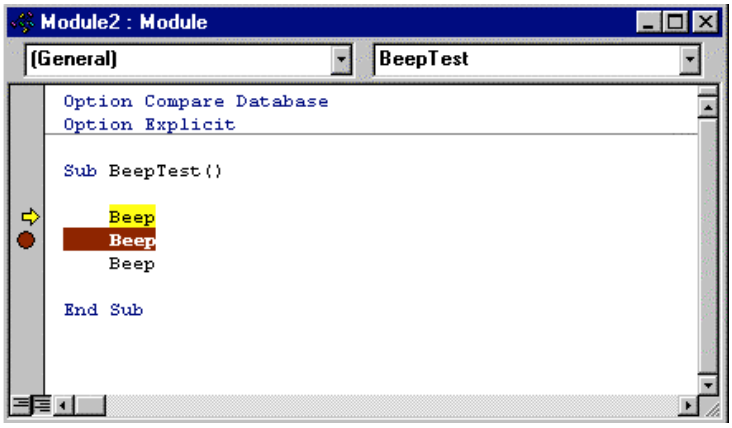
การทดสอบโปรแกรมทำโดยการกดปุ่ม F5 หรือเรียกคำสั่ง Run จาก Menu ปรากฏว่าเสียงที่เกิดขึ้น เหมือนเป็นเสียงเดียว ทั้งที่คำสั่ง Beep เป็นการสั่งให้ Computer สร้างเสียง Beep แล้วเราได้สั่งถึง 3 ครั้ง สาเหตุเป็นเพราะ Computer ทำงานเร็วมาก ในแต่ละคำสั่งที่เราบันทึกเข้าไป Computer จะทำงานด้วยความเร็วตาม Mhz ของเครื่อง (1 Mhz = 1 ล้านคำสั่งต่อวินาที)



การ STEP โปรแกรม

การที่เราจะเห็นขั้นตอนการทำงานของมัน ให้ลองใช้ปุ่ม F8 จะพบว่า จะปรากฏแถบสีเหลืองบน Sub Beep Test แล้วลองกด F8 ซ้ำไปเรื่อยๆ จะพบว่า แถบจะเลื่อนไปทีละคำสั่ง ทำให้เราได้ยินเสียง Beep ถนัดขึ้น คำสั่ง F8 เป็นคำสั่ง STEP เป็นการบอกให้ Access ทำงานที่ละคำสั่ง และหยุดรอ ซึ่งต่างกับ F5 ที่ทำงานทั้งหมดจนจบ

เราใช้ประโยชน์จากคำสั่ง F8 ค่อนข้างมาก เพื่อไว้ตรวจสอบโปรแกรมของเรา เราสามารถย้ายตำแหน่งที่เป็นคำสั่งปัจจุบันได้โดยการ ย้ายลูกศร ที่อยู่บนแถบโดยการ กดแล้วลาก ไปยังบันทัดที่ต้องการ



การ SET BREAK POINT

เราสามารถกำหนดให้ Access หยุดการทำงาน เมื่อถึงบรรทัดใดบรรทัดหนึ่งได้ โดยกำหนด Break Point (กดปุ่ม F9)เมื่อ Access ทำงานโดยคำสั่ง Run (F5) Access จะหยุดที่บรรทัดนั้น รอให้เราทำการต่อ ไม่ว่าจะ Step ด้วย F8 หรือ Run ต่อ ด้วย F5 ก็ตาม

เราสามารถกำหนด Break Point ได้หลายจุดในโปรแกรม การยกเลิก Break Point เพียงแต่ไปที่บรรทัด ที่มี Break Point แล้วกด F9 ซ้ำ หรือใช้คำสั่ง Clear All Break Point (Ctrl + Shift + F9) ใน Menu Debug ก็ได้

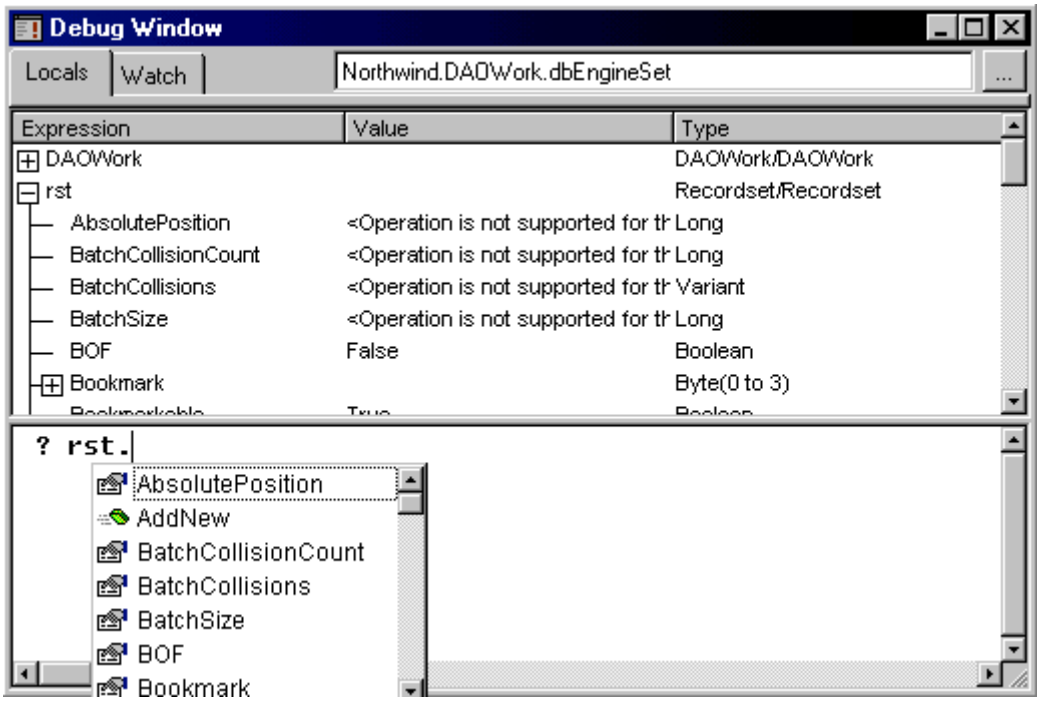
การใช้ Debug Windows

Debug Windows เป็นหน้าต่างๆ อิสระ เสมือนกระดาษทด ที่เราใช้ทดสอบคำสั่ง VB บน Debug Windows จะประกอบด้วย 2 ส่วนคือ ส่วนที่แสดงค่า Object หรือตัวแปรขณะนั้น เราสามารถ Browse ดูค่าต่างๆ ตามลำดับชั้นของ Object กรณีค่าเหล่านั้นฝังอยู่ใน Object

ส่วนที่ 2 เป็น ที่เราเอาไว้ทดลองคำสั่ง เช่น อยากทราบว่าคำสั่ง Beep ทำงานงานอย่างไร ก็ทดลองบันทึกเข้าไปที่ส่วนที่ 2 นี้

เรายังใช้ Debug ในการตรวจสอบค่า ณ ขณะที่เรา Break โปรแกรมที่บรรทัดใดบรรทัดหนึ่งได้ เช่น อยากทราบว่าตัวแปรปัจจุบันมีค่าเท่าไร เราจะพิมพ์ เข้าไปว่า ? แล้วตามด้วยชื่อตัวแปร เราสามารถทดสอบ Function ต่างๆ ณ ที่ Debug Windows ได้ด้วยเช่นกัน

การใช้ Debug Windows จะช่วยให้เราสามารถตรวจสอบความผิดพลาดของโปรแกรมได้ดีขึ้น การเรียก DebugWindows ให้กด Ctrl+G หรือกดปุ่ม  จะได้ Windows ตามตัวอย่าง



ในขณะที่เราอยู่ในโปรแกรม เราสามารถบังคับให้มีการทดลองพิมพ์ค่าที่เราต้องการตรวจสอบออกมาทาง Debug Windows ขณะนั้นได้ โดยใช้คำสั่ง Debug.Print ทดลองใส่คำสั่งนี้ลงในตัวอย่าง Beep

```
Sub BeepTest
    Beep
    Debug.print "Beep 1 Process"
    Beep
    Debug.print "Beep 2 Process"
    Beep
    Debug.print "Beep 3 Process"
End Sub

Debug Windows
--------------
Beep 1 Process
Beep 2 Process
```

```
Beep 3 Process
```

เมื่อเราเรียก Debug Windows ขึ้นมาดู จะพบว่ามีข้อความ Beep n Processs 3 ตัวตามลำดับ ข้อความบนนี้เปรียบเสมือนกระดาษทด เราจะลบทิ้ง ก็ไม่ส่งผลกับโปรแกรมแต่อย่างไร

การเรียก Sub จาก Sub อื่นๆ

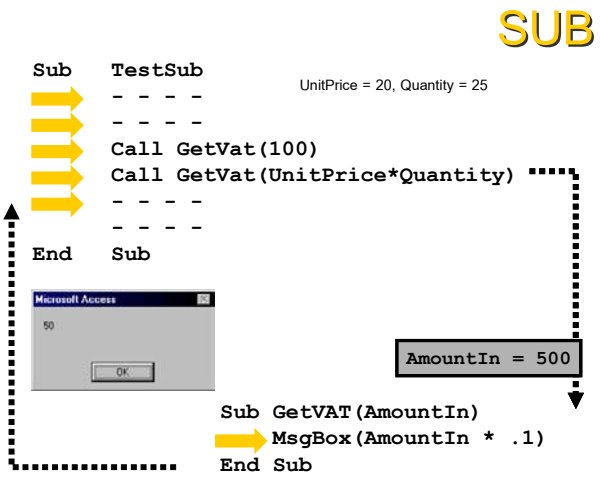
การให้โปรแกรม ทั้งที่อยู่ใน Event ของ Form Report หรือ ใน Sub Rountine ด้วยกัน เรียก Sub ตัวอื่น ก็ทำได้โดยการใช้คำสั่ง Call เช่น การเรียก Sub BeepTest นี้ จะใช้ คำสั่ง Call BeepTest ทดลองสร้าง Sub ตามตัวอย่างอีก 1 Sub ใน Module เดิม

```
Sub CallTest ()
    Call BeepTest
End Sub
```

Sub ที่มีการผ่านค่าตัวแปร

โปรแกรมที่ทำการเรียก Sub สามารถผ่านค่าที่ฝากมาให้ตัว Sub ที่คำนวนได้โดยตัว Sub จะต้องสร้างให้มีการรับตัวแปร และเมื่อมีการเรียกตัวแปร ก็ให้มีการเรียกโดยผ่านตัวแปรเช่นกัน ตัวอย่าง

```
Sub GetVat(PriceIn)
    Msgbox(PriceIn * .10 )
End Sub
Sub TestVat()
    Call GetVAT  ( 100 )
End Sub
```



ในตัวอย่าง เราได้รู้จักคำสั่ง Msgbox ซึ่งใช้ในการแสดงข้อความขึ้นมาบนจอ ตัวอย่างนี้ นำค่าผ่านเข้ามาทาง GetVat ที่ตัวแปร PriceIn ขณะที่ โปรแกรม TestVat เรียกโปรแกรม GetVat โปรแกรมจะแสดงค่า 10 เมื่อโปรแกรมทำงาน (เราต้อง Run โปรแกรมขณะที่ Cursor อยู่ที่ TestVat เนื่องจากโปรแกรมจะทำงานจากจุดนี้)

เราสามารถผ่านค่าตัวแปรได้หลายๆตัว โดยขณะสร้าง Sub ให้ใส่ชื่อตัวแปรที่ต้องการ แล้วขั้นด้วย ,

```
SUB [PROCEDURE NAME] ( v1, v2,…,vn )
```

ก่อนที่จะผ่านไปยังคำสั่งต่อไป ทดลองคำสั่ง InputBox โดยแทน คำสั่ง Call GetVat ด้วย

```
Call getVat(Inputbox("ENTER PRICE")
```

คำสั่ง INPUTBOX ทำงานกลับกับ คำสั่ง MSGBOX ใช้การรับค่าจากโปรแกรม เมื่อได้ค่าแล้ว ค่านั้นก็ถูกส่งผ่านตัวแปร PriceIn ใน GetVat ตัว GetVat ก็จะทำการคำนวนต่อไป

การสร้าง Function

Function เป็นอีกรูปแบบหนึ่งของ Procedure เราเคยใช้ Functionที่มีใน Acces เองมาบ้างแล้วเช่น IIF เป็นต้น เราสามารถสร้าง Function ของเรา ไว้ใช้งานเองได้ Function มีการผ่านตัวแปร หรือไม่ผ่านก็ได้ การเขียน Function ก็ทำใน Module เช่นกัน

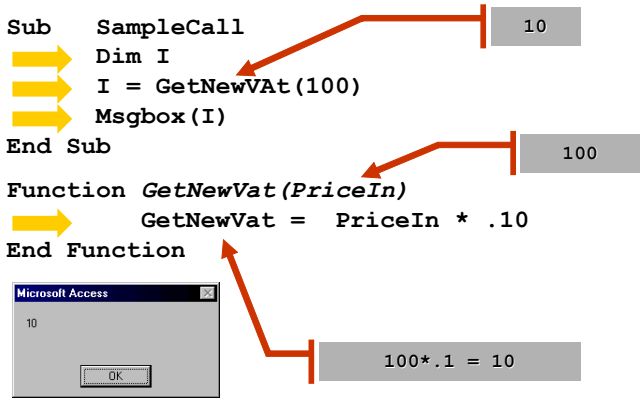
```
Function    GetNewVat(PriceIn)

    GetNewVat =  PriceIn * .10

End Function
```

ตัวอย่างนี้เราใช้โปรแกรมเดิมที่คำนวณ VAT จาก Sub คราวนี้เราใช้ Function แทน แต่ผลการคำนวณ เราสามารถผ่านไปยังผู้เรียกได้ ถ้าเราใช้ FUNCTION การผ่านค่าคืนให้ผู้เรียกก็เพียง แต่กำหนดค่าลงในชื่อฟังก์ชั่นเลย ตามตัวอย่าง

การเรียก Function จากโปรแกรมอื่นๆ จะต่างกับการเรียก Sub เราจะต้องเอาตัวแปรมารับ หรือส่งต่อให้ Object หรือผ่านค่าให้กับฟังก์ชั่นอื่นๆ ตัวอย่างเช่น

```
Sub SampleCall

    Dim I
    I = getNewVAt(100)
    Msgbox(GetNewVat(100))


End Sub
```

(บรรทัดที่เขียนว่า Dim I เป็นการประกาศตัวแปร I เพื่อรับค่าที่ได้จากการคำนวณ VAT)



การกำหนดค่า

การกำหนดค่านั้น เราจะใช้เครื่องหมาย = ในการกำหนด เราสามาถกำหนดค่าให้กับตัวแปร กำหนดให้กับ ค่าใน Control กำหนด Properties แบบที่เราใช้ในการกำหนดค่าลงใน Text Control ในตัวอย่างแรก

ด้านขวาของการกำหนดเราสามารถคำนวณด้วยสูตรคำนวณ หรือเรียกใช้ Function อื่นๆ ซ้อนเข้าไปได้เช่น

```
ID = 1
ID = (( 2 * 3 ) - 10 ) /2
ID = getNewVat(10000)
ID = ( getNewVat(10000) * 10 ) + 100
ID = ID + 1
```

เราสามารถใช้ Function Expression ในการกำหนดค่าแบบเดียวกับที่เคยใช้ใน Form, Report, Query ได้เช่นกัน

เรื่องของตัวแปร

ตัวแปรเป็นที่เก็บค่าที่ประมวนผล ระหว่างการประมวลผลในทุกภาษาคอมพิวเตอร์ การใช้ตัวแปรใน Access ควรจะต้องมีการประกาศค่าเสียก่อน โดยปกติสำหรับ Access 97 บนตอนต้นของ Module จะมีคำสั่ง Option Explicit ซึ่งเป็น Option ที่กำหนดให้มีการประกาศค่าตัวแปรก่อนทุกครั้ง ก่อนใช้งาน การประกาศตัวแปรใน Class Module ใช้คำสั่ง Dim ถ้าเราไม่ประกาศตัวแปร จะเกิด ERROR

วิธีประกาศ	คำอธิบาย
Dim I	ไม่กำหนดว่าเป็นตัวแปรประเภทอะไร Access ใช้เป็น Variant
Dim I as Integer	กำหนดให้เป็น Integer

ในเบื้องต้น ให้กำหนดตัวแปรโดยใช้คำสั่ง Dim ซึ่งเราจะกล่าวถึงตัวแปรโดยละเอียดอีกครั้งในหัวข้อตัวแปร

การควบคุมการประมวลผล

ตามปกติ โปรแกรมคอมพิวเตอร์จะทำงานตามลำดับคำสั่งที่เรากำหนด เช่นการสั่งคำสั่ง BEEP 3 ครั้งในตัวอย่างแรก เราสามารถสั่งให้โปรแกรมทำคำสั่งในแบบที่ไม่เป็นลำดับได้ ตรงนี้ก็คือการควบคุมลำดับการประมวลผลนั่นเอง

หัวใจของการเขียนโปรแกรม คือการกำหนด ควบคุม ขั้นตอนการประมวลผล ภาษา BASIC มี ชุดคำสั่งเหล่านี้ให้เราสามารถทำการควบคุมการประมวลผลของคอมพิวเตอร์ได้เป็นอย่างดี คอมพิวเตอร์สามารถทำงานที่ซับซ้อนได้เป็นอย่างดี เนื่องมาจากความสามารถของ Programmer ที่จะกำหนด ควบคุม การประมวลผลของโปรแกรมให้ทำงานที่สลับซับซ้อน ตามเงื่อนไขการประมวลที่กำหนด การเขียนโปรแกรมจึงเป็นการบังคับให้ Computer ประมวลผลตามที่ Programmer กำหนดนั่นเอง

หากแบ่งคำสั่งที่ควบคุมการประมวลผลแล้ว อาจจัดได้ 2 กลุ่มคือ

- **กลุ่มที่บังคับให้ประมวลผลอย่างมีเงื่อนไข (IF)**

  ใช้ในการตรวจสอบค่า แล้วทำคำสั่งตามเงื่อนไขที่กำหนด

  - IF…THEN
  - IF… THEN… ELSE
  - IF… THEN… ELSEIF
  - SELECT… CASE

- **กลุ่มที่บังคับให้ทำซ้ำ ๆ (LOOP)**

  ใช้ในการสั่งให้ทำคำสั่งซ้ำจนกว่าจะถึงเงื่อนไขที่กำหนด

  - FOR…NEXT
  - DO … LOOP
  - WHILE … Wend

กลุ่มที่บังคับให้ประมวลผลอย่างมีเงื่อนไข (IF)

IF … THEN

คำสั่งนี้เป็นคำสั่งพื้นฐานมี SYNTAX คือ

SYNTAX

```
IF  CONDITION THEN  COMMAND
```

หรือ

```
IF  CONDITION THEN
    COMMAND 1
    COMMAND 2
    ….
    ….
    COMMAND n

END IF
```

คำสั่งนี้ จะใช้ตรวจสอบเงื่อนไข (CONDITION) ถ้าเป็นจริงก็จะทำงานในชุดคำสั่ง ที่ตามหลังคำสั่ง THEN ถ้าไม่เป็นจริงก็จะข้ามคำสั่งที่อยู่หลัง THEN ไป คำสั่งทั้ง 2 แบบต่างกันที่แบบที่ 2 จะมีคำสั่งหลัง THEN มากกว่า 1 คำสั่ง จะต้องขึ้นบรรทัดใหม่ แล้วปิดชุดคำสั่งด้วย END IF เพื่อให้ตัว VB ทราบว่าขอบเขตของคำสั่งจบที่ใด ถ้าเราไม่กำหนด จะเกิด Error

เงื่อนไขที่ระบุเรามักจะใช้ในการตรวจสอบค่าตัวแปร, ค่าใน PROPERTIES การที่เงื่อนไขเป็นจริง ก็หมายถึงเงื่อนไขนั้นให้ค่าเป็น TRUE ถ้าไม่จริงก็จะเป็น FALSE

เราลองย้อนกลับไปที่โปรแกรมเดิม แล้วใส่คำสั่งลงใน COMMAND1_CLICK เป็น

```
Private Sub Command1_Click()
        If ID < 10 then
           ID = ID + 1
        End if
        IF ID >= 10 then
           Msgbox("Maximum 10 only ")
        End if
End Sub
```

โปรแกรมนี้จะทำงานแบบมีเงื่อนไขทันที นั่นคือถ้า ID < 10 ถึงจะทำการบวก ถ้า ID >=10 จะขึ้นข้อความเตือน การกำหนดเงื่อนไข ต้องอาศัยความคุ้นเคยทางด้านตรรกศาสตร์ ในการกำหนด เราสามารถใช้ AND , OR , NOT ในการกำหนดการร่วมเงื่อนไขกับเงื่อนไข แต่ละเงื่อนไขจะต้องให้ค่าที่เป็น จริง หรือเท็จ ทดลองดูตัวอย่างการกำหนดเงื่อนไขดังต่อไปนี้ จะช่วยให้เข้ามากขึ้น

โปรแกรม	คำอธิบาย
`IF TRUE THEN` `IF NOT ( FALSE ) THEN`	ให้ค่าที่เป็นจริงเสมอ , ทำคำสั่งถัดไปเสมอ
`IF FALSE THEN` `IF NOT ( TRUE ) THEN`	ให้ค่าที่เป็นเท็จเสมอม ข้ามคำสั่งหลัง Then ไป
`IF 1 = 1 THEN`	ให้ค่าที่เป็นจริงเสมอ, ทำคำสั่งถัดไปเสมอ
`IF 1 = 2 THEN`	ให้ค่าที่เป็นเท็จเสมอ ข้ามคำสั่งหลัง Then ไป
`IF I = 1 THEN`	ถ้า I = 1 ทำคำสั่งถัดไป
`IF I + 10 > J THEN` `IF NOT(I + 10 <= J ) THEN`	ถ้า I + 10 แล้วมากกว่า J ก็จะทำคำสั่งถัดไป
`IF (I +10 > J) and ( X = 2 ) THEN` `IF not ((I +10 <= J) or ( X <> 2 ))THEN`	เมื่อ I +10 > J **และ** X = 2 การใช้ AND จะทำให้เราเชื่อมต่อเงื่อนไข หลายๆ เงื่อนไขได้

`IF (I +10 > J) or ( X = 2 ) THEN`
`IF not ((I +10 <= J) and ( X <> 2 ))`
`THEN`
	เมื่อ I +10 > J **หรือ** X = 2

`IF ((I +10 > J) or ( X = 2 )) and (L >` `10) THEN`	เป็นจริงเมื่อเงื่อนไขใดเงื่อนไขหนึ่งในคู่ OR เป็นความจริง และ L จะต้องมากกว่า 10 ด้วย

เงื่อนไขที่ระบุเพื่อการเปรียบเทียบ ทั้ง 2 ด้านที่ทำการเปรียบเทียบ ควรจะเป็นตัวแปรชนิดเดียวกัน หากเป็นตัวแปรที่ไม่ตรงกัน VB อาจให้ผลผิดพลาดได้ ตัวอย่างเช่น

```
Dim x

x = "11"
If x > 10 Then
   MsgBox ("TIME")
End If
```

ควรใช้คำสั่ง Val เพื่อแปลง X ซึ่งเก็บค่าที่เป็น String อยู่ให้เป็นตัวเลข เราจะกล่าวถึง Function ในการแปลงตัวแปรอีกครั้งในเรืองการใช้ตัวแปร

```
Dim x

x = "11"
If val(x) > 10 Then
   MsgBox ("TIME")

End If
```

การใช้ IF สามารถใส่คำสั่ง IF หลังคำสั่ง IF เป็นชั้นๆ ต่อเนื่องไปได้ ซึ่งจะได้ความหมายเดียวกับการใช้ AND ถ้าคำสั่ง IF ที่ตามหลังไม่คำสั่งอื่น เช่น

```
IF I > 10 > J THEN
```

```
    IF X = 2 THEN
    …….
    END IF
END IF
```

เท่ากับคำสั่ง

```
IF (I + 10 > J) and (X = 2) Then
    ……..

END IF
```

IF….. THEN ….. ELSE

คำสั่ง IF Then Else จะเหมือนกับคำสั่ง IF เพียงแต่ เมื่อเงื่อนที่กำหนดไม่เป็นจริง จะทำคำสั่งหลัง Else แทน VB จะเลือกทำคำสั่งหลัง Then จนถึง Else ถ้าเงื่อนไขเป็นจริง และทำคำสั่งหลัง Else จนถึง End If เมื่อเงื่อนไขเป็นเท็จ มีรูปโครงคำสั่งคือ

SYNTAX

```
IF  CONDITION THEN
    TRUE COMMAND 1
    TRUE COMMAND 2
    ….
    ….
    TRUE COMMAND n
ELSE
    FALSE COMMAND 1
    FALSE COMMAND 2
    ….
    ….
    FALSE COMMAND n

END IF
```

การกำหนดเงื่อนไข จะทำเช่นเดียวกับการกำหนดเงื่อนของ IF ปกติทั่วไป เราสามารถอธิบายคำสั่ง IF THEN ELSE ด้วย IF 2 ชุดดังแสดง

```
IF  CONDITION1 THEN
    TRUE COMMAND 1
    TRUE COMMAND 2
    ….
    ….
    TRUE COMMAND n
END IF
IF NOT ( CONDITION1 ) THEN
    FALSE COMMAND 1
    FALSE COMMAND 2
    ….
    ….
    FALSE COMMAND n

END IF
```

IF… THEN … ELSEIF …. THEN

ชุดคำสั่งนี้เป็นการกำหนดเงื่อนไขการตรวจสอบเป็นลำดับเงื่อนไข คล้าย IF THEN ELSE หลายๆ ลำดับ มีโครงสร้างในการเขียนคือ

SYNTAX

```
IF CONDITION1 THEN
    COMMAND11
    COMMAND12
    ……
    COMMAND1n
ELSEIF CONDITION2 THEN
       COMMAND21
       COMMAND22
            ……
       COMMAND2n
ELSEIF CONDITION3 THEN
       COMMAND31
       COMMAND32
       ……
       COMMAND3n
END IF
```

ซึ่งจะมีความหมายเดียวกับการใช้ IF หลายตัวคือ

```
IF CONDITION1 THEN
    COMMAND11
    COMMAND12
    ……
    COMMAND1N
END IF
IF NOT(CONDITION1) AND CONDITION2 THEN
    COMMAND21
    COMMAND22
    ……
    COMMAND2N
END IF
IF NOT(CONDITION1) AND NOT(CONDITION2) AND CONDITION3 THEN
    COMMAND31
    COMMAND32
    ……
    COMMAND3N
END IF
```

เราใช้ IF แบบนี้กับการตรวจสอบเงื่อน แบบ TREE ซึ่งเหมือนกับการใช้คำสั่ง CASE ดังจะกล่าวถึงต่อไป ข้อดีของคำสั่ง ชุด IF ELSEIF นี้ คือเราสามารถตรวจสอบเงื่อนไขหลายเงื่อนไข โดยจบชุดเงื่อนไข ด้วย END IF เพียงตัวเดียว และการเขียนเงื่อนไข จะซับซ้อนน้อยกว่า เนื่องจากไม่ต้องกำหนดเงื่อนไข AND ไปเรื่อย ในตัวอย่าง Code ที่แสดงไว้

SELECT…. CASE

การใช้ Select Case เป็นการควบคุมการประมวลผล แบบมีหลายๆ เงื่อนไข โดยใช้การตรวจสอบค่า แล้วแยกไปทำการประมวลผลในชุดคำสั่งที่กำหนด ตามค่าที่ตั้งไว้ มีโครงสร้างภาษาคือ

SYNTAX

```
Select Case i
Case Val1
        COMMAND11
        COMMAND12
        ….
        COMMAND1N
Case Val2
        COMMAND21
        COMMAND22
        ….
        COMMAND2N
Case Val3
        COMMAND31
        COMMAND32
        ….
        COMMAND3N

Case ValM
        COMMANDM1
        COMMANDM2
        ….
        COMMANDMN
End Select
```

เรามักใช้ Select Case ในการแปลงค่า หรือเลือกประมวลผล เช่น เราต้องการตรวจสอบประเภทของผู้ใช้ แล้วเลือกทำคำสั่งที่เหมาะสม สำหรับ User แต่ละประเภท

```
sub CheckUSer(UserType)

    Select Case UserType
    Case "S"
            'Command for Supervisor User
            '
            '
            Msgbox("You are Supervisor User")

    Case "N"
            'Command for Normal User
            '
            '
            Msgbox("You are Normal User")

    Case "A"
            'Command for Admin User
            '
            '
            Msgbox("You are Admin User")

    End Select
End Sub
```

การเลือกใช้ IF หรือ SELECT CASE

เราเลือกที่จะใช้ IF หรือ SELECT CASE โดยวิเคราะห์การประมวลผลแบบทางใดทางหนึ่ง หรือ เลือกได้หลายทาง ถ้าเป็นทางใดทางหนึ่งเราจะใช้ IF แต่ถ้าต้องเลือกทางเลือกหลายๆทาง เราจะใช้ SELECT CASE แทน

การทำงานกับฐานข้อมูลโดยการเขียน Code Form เราจะได้ใช้ IF, SELECT CASE เพื่อการตรวจสอบเงื่อนไขต่างๆ แล้วบังคับให้ Access ประมวลผลแตกต่างกันไป หรือ การห้ามให้ User ทำ หรือ ไม่ทำเงื่อนไขบางอย่าง

กลุ่มที่บังคับให้ทำซ้ำๆ (LOOP)

FOR…NEXT

คำสั่ง FOR NEXT เป็นการกำหนดให้ทำการประมวลผลซ้ำๆ ตามจำนวนครั้งที่กำหนด ในแต่ละการทำซ้ำ ซึ่งเราเรียกว่า LOOP จะทำการนับค่าที่กำหนดไว้ในตัวแปรไปเรื่อย มีโครงสร้างคำสั่งคือ

SYNTAX

```
For  <var> = <start number> to <end number> step <step>
    COMMAND1
    COMMAND2
    …….
    COMMANDn
Next <var>
```

คำสั่งที่อยู่บรรทัดหลัง FOR จนถึง NEXT จะถูกประมวลผลไปเรื่อย จนกว่า ค่า <var> จะมีค่ามากกว่า <end number> โดยในแต่ละ Loop ค่า <var> จะเปลี่ยนแปลงอัตโนมัติ คือ <var> = <var> + <step> ถ้าเรา ไม่กำหนด Step, VB จะใช้ค่า 1 แทน ทดลองคำสั่งชุดนี้ เพื่อให้เข้าใจมากขึ้น

```
For I = 1 to 10 Step 2
    Msgbox("Hello "  & I )
Next I
```

คำสั่งจะทำการ LOOP 5 ครั้ง ในแต่ละ LOOP ค่า I จะเพิ่มขึ้นที่ละ 2 นับจาก 1,3,5,7,9 ถ้าเราย้อนไปดูตัวอย่างคำสั่ง BEEP สามครั้งที่เราเคยทดลอง จะพบว่าสามารถใช้คำสั่ง FOR แทนได้คือ

```
For I = 1 to 3
    BEEP
Next I
```

บ่อยครั้งที่เรามักใช้ For Loop ซ้อนกับ For Loop หลายตัว เพื่อทำงานหลายๆชั้น จำนวน Loop ก็จะทวีคูณทันที ตัวอย่างด้านล่างนี้เป็นการทดลองใช้ Loop คำนวนเพื่อหาสูตรคูณ 1 ถึง 10

```
Sub TestLoop()

    Dim i, j
```

```
    For i = 1 To 10

        For j = 1 To 10
            MsgBox ("Result of : " & i & " X " & j & " = " & i * j)
        Next j

    Next i

End Sub
```

ในระหว่างที่อยู่ใน For Loop เราสามารถบังคับให้ออกจาก For Loop ได้ โดยการใช้คำสั่ง Exit For โดยเราจะใช้ควบคู่ไปกับคำสั่ง IF เสมอ เช่นตัวอย่างด้านบน ถ้าเราให้โปรแกรมหยุดทำงานที่ สูตร 5 คูณ 5 โปรแกรมจะเป็น

```
Sub TestLoop()

    Dim i, j
    For i = 1 To 10

        For j = 1 To 10
            MsgBox ("Result of : " & i & " X " & j & " = " & i * j)
            If I = 5 and J = 5 then
               Exit For
            End if
        Next j

    Next i

End Sub
```

DO…LOOP

คำสั่ง Do Loop มีการทำงานคล้ายกับ For แต่เงื่อนไขที่กำหนดว่า Loop จะทำการจนถึงเมื่อไร จะกำหนดด้วยเงื่อนไข แล้วใช้ Until หรือ While ในการควบคุม มีรูปแบบคำสั่งคือ

```
Do
    Command1
    Command2
    --------
    Commandn
Loop

Do Until Condition
    Command1
    Command2
    --------
    Commandn
Loop

Do While Condition
    Command1
    Command2
    --------
    Commandn

Loop

Do
    Command1
    Command2
    --------
    Commandn
```

```
Loop Until Condition

Do
    Command1
    Command2
    --------
    Commandn

Loop While Condition
```

- **กำหนดเงื่อนไขหลัง DO หรือ Loop** ถ้า While หรือ Until อยู่หลัง DO โปรแกรมจะตรวจสอบเงื่อนไข ก่อนเริ่มทำชุดคำสั่ง ส่วน กรณีที่อยู่หลัง Loop โปรแกรมจะทำการจนเสร็จสิ้นชุดคำสั่ง แล้วจึงตรวจเงื่อนไข การเลือกตำแหน่งที่กำหนด ก็จะขึ้นกับการใช้งานแต่ละครั้ง
- **ใช้ While หรือ Until** ความจริง While หรือ Until มีความหมายที่ตรงกันข้ามกันอยู่ กล่าวคือ ถ้า Condition หลัง While เป็นจริงคำสั่งใน Loop จะถูกประมวลผลไปเรื่อยๆ จน เงื่อนไขที่กำหนด **เป็นเท็จ** ส่วน Until จะเป็นการบอกให้ Do Loop ประมวลผลไปเรื่อยๆ จนกว่า เงื่อนไขหลัง Until **จะเป็นจริง**
- **ถ้าเราไม่กำหนด Until หรือ While เลย** Do Loop จะทำงานไปเรื่อยๆ จนกว่าเราจะสั่งให้ หยุด ด้วยคำสั่ง Exit Do เรามักใช้คู่กับ IF หรือ Case เสมอ จะพบว่า Do While True , Do Until False จะมีความหมายเหมือนกับ Do Loop แบบไม่กำหนดเงื่อนไขนั่นเอง

ใช้ Do..Loop หรือ For

เรามักใช้ Do While ในการทำการซ้ำๆ จนกว่าจะถึงเงื่อนที่เรากำหนด โดยที่เราไม่ทราบจำนวน Loop ที่จะเกิดขึ้นแน่นอน จำนวน Loop ที่โปรแกรมจะทำ จะขึ้นอยู่กับชุดคำสั่งที่กำลังประมวลผล ส่วน For เราจะใช้กับ Loop ที่เรารู้จำนวน Loop แน่นอน

เวลาที่เราใช้ Do Loop เรามักนับจำนวน Loop ในการประมวลผลด้วยเสมอ เพื่อให้รู้ว่าเราทำไปกี่ Loop แล้ว แบบที่เราสามารถอ่านค่าได้โดยตรงจากการใช้ For Next วิธีการนับจำนวน Loop เราจะใช้ตัวแปรมานับ ด้วยคำสั่ง I = I + 1 เมื่อสิ้นสุด Loop ตัวแปรนั้นก็จะเก็บค่าจำนวน Loop ที่ทำพอดี

```
I = 0
Do While <Condition>

    Command1
    Command2
    --------
    Commandn

    I = I +1
Loop
```

While..Wend

คำสั่งนี้ จะคล้ายกับกับคำสั่ง Do While ทุกประการ มี รูปแบบโครงสร้างคือ

```
While <Condition>
    Command1
```

```
    Command2
    --------
    Commandn
Wend
```

การเปรียบเทียบ เงื่อนไข

เรามารู้จักเครื่องหมายที่ใช้ในการกำหนดเงื่อนไขต่างๆ ที่มีใน Access เพื่อใช้ประโยชน์ Access ได้มากขึ้น

การเชื่อมต่อ STRING

เวลาที่เราต้องการเชื่อมต่อ ข้อมูลที่เป็น String เข้าด้วยกัน เราจะใช้เครื่องหมาย **&** หรือเรียกว่าการ Concatenation ข้อมูลที่เป็น String จะมีการปิดหัว และหางของ String ด้วยเครื่องหมาย “ เพื่อให้ Access รู้ว่า String เริ่มจากที่ใด และไปจบที่ใด ทดลอง Run โปรแกรมการเชื่อมต่อ String

```
Sub test()

    Dim i, strx
    strx = "SAMPLE STRING:"
    For i = 1 To 12
        strx = strx & i & ","

    Next i
    MsgBox (strx)
End Sub
```

การเปรียบเทียบ String

เราสามารถเปรียบเทียบ String หรือข้อมูลที่เป็นตัวอักษรด้วยเครื่องหมายดังต่อไปนี้

=, IS

เปรียบเทียบทั้ง 2 ด้านของเครื่องหมาย โดยค่าทั้ง 2 ด้านจะต้องเหมือนกัน จึงจะให้ค่าเป็นจริง

```
if StrA =  “ACCESS” then
   Msgbox(“Yes , Access”)
End If
if StrA IS  “ACCESS” then
   Msgbox(“Yes , Access”)
End If
```

ประโยคดังกล่าว จะเหมือนกับชุดคำสั่งนี้ ที่ใช้การต่อ STRING แทน

```
StrX = “ACC”
if StrA =  Strx & “ESS” then
   Msgbox(“Yes , Access”)
```

```
End If
```

เนื่องจาก Strx & “ESS” แล้วได้เท่ากับ ACCESS พอดี

Like

เราใช้ประโยชน์จากเครื่องหมาย Like เป็นอย่างมากในการค้นหา เราจะใช้ *, ?, # ช่วยในการเปรียบเทียบ ด้านขวาของ Like เราจะกำหนดเป็น Pattern ส่วนด้านซ้ายจะเป็น String ที่เราต้องการเปรียบเทียบ เช่น

```
If Strx Like "A*" then
```

ประโยคนี้ให้ค่าเป็นจริงเมื่อ Strx เป็น String ที่ขึ้นต้นด้วย A การกำหนดตัวอักษรพิเศษนี้ เรียกว่า WildCard มีรูปแบบคือ

เครื่องหมาย	การใช้	คำอธิบาย
*	A*	ตัวอักษรทุกตัวที่ขึ้นต้นด้วย A จะเป็น ACCESS, ANT ,.. ได้ทั้งนั้น
?	B?LL	ลำดับตัวอักษรที่เป็น ? เป็นตัวอะไรก็ได้ ขอให้ขึ้นต้นด้วย B และลงท้ายด้วย LL ดังนั้น BALL, BILL, BELL จะให้ค่าที่เป็นจริงทั้งหมด
#	A#A	ตรงที่เป็นเครื่องหมาย # จะต้องเป็นตัวเลข และเริ่มต้นด้วย A จบ ด้วย A
[]	A[abc]C	การระบุ [ ] เพื่อให้เลือกตัวอักษรที่อยู่ในปีกกา หมายความว่าตัวอักษรที่ 2 ต้องเป็น a, b, c ดังนั้นจะได้ว่า AaC, AbC, AcC จะให้ค่าที่เป็นจริงทั้งสิ้น
!	A[!abc]C	อันนี้จะตรงกันข้ามกับ A[abc]C กล่าวคือ ค่าที่จะเป็นจริงก็ต่อเมื่อ ตัวอักษรที่ 2 ไม่ใช่ a, b, c
-	A[a-f]C	เราใช้ – ช่วยในการกำหนดตัวอักษรเป็นช่วง เพราะการกำหนดเป็นตัวๆ คงไม่สะดวกนัก เราจึงใช้ – เพื่อบอกขอบเขตตัวอักษรเริ่ม และ จบ เช่น A[a-f]C = A[abcdef]C ถ้าเป็น A[!a-c]C ก็เท่ากับ A[!abc]C

การเปรียบเทียบตัวเลข

การเปรียบเทียบแบบตัวเลข คงหนีไม่พ้นการใช้มากกว่า น้อยกว่าดังตาราง

เครื่องหมาย	ความหมาย	ตัวอย่าง
=	เท่ากับ	`IF I = 10 Then`
>	มากกว่า	`IF I > 10 Then`
<	น้อยกว่า	`IF  I < 10 Then`
>=	มากกว่าหรือเท่ากับ	`IF I >= Then`
<=	น้อยกว่าหรือเท่ากับ	`IF I <= Then`
<>	ไม่เท่ากับ	`IF I <> Then`

การกำหนดตัวแปร

ตัวแปรเป็นเสมือนที่พักการคำนวณของคอมพิวเตอร์ การเลือกใช้ และกำหนดตัวแปรเป็นสิ่งสำคัญมาก ตัวแปรที่สำคัญของ Access ที่เราควรรู้จักได้แสดงตามตาราง ตัวแปรแต่ละประเภทจะมีขนาดของหน่วยความจำที่ใช้ ความสามารถในการเก็บ ค่าสูงสุด ต่ำสุด แตกต่างกันไป

ตัวแปร	เครื่องหมาย	ขนาด	ขอบเขตของค่าที่เก็บ
Byte	None	1 byte	0 ถึง 255
Boolean	None	2 bytes	True หรือ False
Integer	%	2 bytes	–32,768 ถึง 32,767
Long	&	4 bytes	–2,147,483,648 ถึง 2,147,483,647
Single	!	4 bytes	–3.402823E38 ถึง –1.401298E–45 สำรับตัวเลขลบ 1.401298E–45 ถึง 3.402823E38 สำหรับตัวเลขบวก
Double	#	8 bytes	4.94065645841247E–324 ถึง 1.79769313486232E308 สำหรับตัวเลขบวก –4.94065645841247E–324 ถึง -1.79769313486232E308 สำหรับตัวเลขลบ
Currency	@	8 bytes	-922,337,203,685,477.5808 ถึง 922,337,203,685,477.5807
Date	None	8 bytes	January 1, 100 to December 31, 9999 เก็บทั้งวัน และเวลา
Object	None	4 bytes	ตามแบบ Object ที่กำหนด
String ที่ไม่กำหนดขนาด	$	10 bytes + กับความยาวของ String จริงที่เก็บ	ขนาดตั้งแต่ 0 ถึง 2,000 ล้าน
String ที่กำหนดขนาด	None	ความยาวตามขนาดที่กำหนด	1 ถึง 65,400
Variant ที่เป็นตัวเลข	None	16 bytes	ขนาดสูงสุดเท่ากับ Double
Variant ที่เป็นตัวอักษร	None	22 bytes + ขนาดของ String	ขนาดสูงสุดเท่ากับ String ที่ไม่กำหนดขนาด

วิธีการประกาศตัวแปรเพื่อใช้ในระบบ จะใช้คำสั่ง Dim ตามด้วยชื่อ และประเภท โดยจะขั้นด้วย AS

```
Dim strX as String*10
Dim strX as String
Dim DateX  as Date
Dim intCount as Integer
Dim NonSet
```

ถ้าเราไม่กำหนด ประเภทตัวแปร VB จะถือว่าตัวแปรนั้นเป็น Variant ซึ่งมีขนาดหน่วยความจำที่ใช้สูงสุด หากรู้ว่าตัวแปรที่กำลังจะใช้เป็นประเภทใดแน่นอน ก็ควรกำหนดประเภทให้ชัดเจน การกำหนดตัวแปรควรประกาศไว้ส่วนบนสุดของ โปรแกรม

SCOPE การเข้าถึงตัวแปร

เราสามารถกำหนดตัวแปรได้หลายตำแหน่ง แต่ละตำแหน่งจะใช้ไม่เหมือนกัน ดังอธิบายในตาราง

ประเภท	คำอธิบาย	ตัวอย่าง

กำหนดตัวแปรใต้ SUB	ตัวแปรที่กำหนด จะมองเห็นเฉพาะโปรแกรม หรือ คำสั่งใต้ Sub นั้นเท่านั้น	`Sub TestMe()`
		`  Dim I as Integer`
		`  ……`
		`  …..`
		`End Sub`

กำหนดตัวแปรไว้ที่ส่วน Declarations ของ Class Module , Standard Module	ตัวแปรที่กำหนดในขอบเขตนี้ จะใช้ได้กับ Sub หรือ Function ทุกตัว ที่อยู่ใน Module เดียวกัน การกำหนดค่า อ่านค่า ก็สามารถผ่านค่าข้ามกันได้เฉพาะ Sub หรือ Function ใน Module เดียวกัน	`Dim I as Integer`
		`Sub TestME1()`
		`  I = I +1`
		`End Sub`
		`Sub TestME2()`
		`  Msgbox(I)`
		`End Sub`

กำหนดเป็น Global แทนคำว่า Dim โดยไว้ที่ Standard Module	ตัวแปรดังกล่าว เมื่อกำหนดไว้ที่ Standard Module ด้วยคำสั่ง Global แทน Dim จะทำให้ตัวแปรนั้น มองเห็นโดยทุก Sub หรือ Function และในทุกๆ Module	`Global I as Integer`
		`Sub TextMe1()`
		`  I= I +1`
		`End Sub`
		`Sub TestMe2()`
		`  I = I +1`
		`End Sub`

การใช้ Global แทน Dim จะทำได้กับเฉพาะใน Standard Module เท่านั้น ไม่สามารถกำหนดไว้ใน Class Module หรือ Form Module ได้

การกำหนด Constant

เราสามารถกำหนดค่าคงที่ สำหรับการใช้งานในระบบได้ เพื่อสะดวกในการอ้างถึง ตัวอย่างเช่นกำหนดค่าคงที่ VAT เป็น 10 ไว้ที่ Constant กำหนดชื่อบริษัทเป็นค่า Costant วิธีการกำหนดค่า Constant ทำโดยกำหนด

```
Const VAT = 10
```

หรือ

```
Global Const  gblVAT = 10
```

ในเรื่องของ Scope ของ Const จะเหมือนกับ Dim ทุกประการ กล่าวคือ

- ถ้ากำหนดไว้ใต้ Sub จะเห็นเฉพาะ Sub
- ถ้ากำหนดไว้ที่ Declaration จะเห็นใน Module นั้น
- ถ้าใช้ Global นำหน้า และกำหนดไว้ที่ Standard Module ค่า Constant นี้ จะใช้งานได้ทั้งระบบ

การตั้งชื่อตัวแปรซ้อน

ถ้าบังเอิญเราตั้งชื่อตัวแปรซ้ำกัน ในแต่ละ Scope คือตั้งไว้ใน Sub / Function และใน Declaration และเป็น Global , VB จะใช้ค่าตัวที่อยู่ใน Scope ที่ใกล้ตัวกับมันมากที่สุด กล่าวคือ จะเลือกตัวแปรที่ประกาศไว้ใน Sub หรือ Function ของตัวมันเองก่อน ถัดมาเป็นตัวที่อยู่ใน Module เดียวกัน แล้วจึงเป็นตัวที่เป็น Global ตัวอย่างเช่น

```
Option Explicit
Dim I as Integer

Sub RunMe
    I = 10
    Call Test
End Sub

Sub Test
    Dim I
    I = I + 12
    Msgbox(I)
End Sub
```

ทดลอง Run Program ที่ RunME จะพบว่า ค่า I ที่แสดงผ่าน Msgbox เท่ากับ 12 เพราะ Runme ไปเรียก Test ที่มีการประกาศตัวแปร I ไว้ ภายใต้ Rountine นี้ จะไม่มีการใช้ I ที่ประกาศไว้ด้านบนสุด เมื่อ I = I + 12 จึงได้ค่าเป็น 12 เนื่องจากยังไม่เห็นค่า I ที่เป็นส่วน Declaration.

ทดลองลบคำสั่ง Dim I ใน Test แล้ว Run Program อีกครั้งจาก RunMe จะพบว่า ค่าที่ได้จะเป็น 22 เพราะเป็นการอ้างอิงถึงค่า I ตัวเดียวกันแล้ว

ควรตั้งตัวแปรไว้ที่ใด

การเลือกที่ประกาศตัวแปร ไม่ส่งผลกับการเขียนโปรแกรม แต่ จะส่งผลต่อความเร็วในการทำงานของ VB และ Memory ที่มีในระบบ ถ้าเราประกาศตัวแปรไว้ใน Sub /Function ตัวแปรนั้นจะถูกทำลาย เมื่อ Sub/Function โปรแกรมออกจาก Sub/Function นั้น Memory ที่พักสำรองสำหรับตัวแปรนั้น ก็จะถูก Free

เช่นเดียวกับการประกาศไว้ที่ Class Module โดยเฉพาะที่ Form ตรง Declaration เมื่อ Form ถูกเรียก VB จะต้องใช้หน่วยความจำเพื่อไว้พักการคำนวณ ตราบใดที่ Form นี้ยังเปิดอยู่ และหากเราประกาศเป็น Global ไว้ แน่นอนว่า หน่วยความจำนั้นจะถูกใช้ จนกว่าเราจะปิด Access นั่นเอง

การเลือกตำแหน่งที่จะประกาศตัวแปรจึงสำคัญในแง่ของ Performance ของระบบ ต้องคำนึงถึงอายุการใช้งานของตัวแปรนั้น ว่าต้องใช้ทั้งระบบหรือไม่ ตัวแปรบางอย่างจำเป็นต้องเป็น Global เช่น ชื่อ Login เพราะเราต้อง Track ชื่อตัวแปรตลอดเวลา ส่วนตัวแปรที่ไว้พักการคำนวณ สำหรับ For Next, Do Loop ควรประกาศไว้ใช้งานใน Sub นั้นก็พอ เป็นต้น

ตัวแปรแบบ Array

เป็นตัวแปรที่มีลักษณะเป็น Set หรือ กลุ่มตัวแปร เวลาที่อ้างค่า เราจะต้องบอกลำดับที่ Index ที่ต้องการอ่านค่า ส่วนประเภทของค่าตัวแปร จะขึ้นอยู่กับที่เราประกาศ แบบตัวแปรปกติ เช่น

```
Dim dayThai(7) as String
```

ประกาศตัวแปรแบบ Array 7 ตัว เป็นแบบ String เวลาที่เราอ่านค่า หรือกำหนดค่า จะใช้ ค่า 0 ถึง 6 หรือจำนวนตัวแปรทั้งหมด ลบ 1 เพราะตัวแปรตัวแรก มีดัชนีเป็น 0 ไล่ไปจนถึงตัวที่ n-1

```
dayThai(0) = “อาทิตย์”
dayThai(1) = “จันทร์”
----------------
```

เรามักอ่านค่าโดยการแทนค่าตัวแปรลงในลำดับที่ Array ที่ต้องการ เช่น

```
I =1
Msgbox(dayThai(I))
```

Function ใน Access

ใน ACCESS มี Function อยู่มากมายให้เลือกใช้ เราจะใช้ Function ในการแปลงค่า ทำการบางอย่าง หรืออ่านค่าบางอย่างในระบบอยู่บ่อยครั้ง เราจะกล่าวถึงเฉพาะ Function ที่มีการใช้งานอยู่บ่อยๆ

กลุ่ม Convert ข้อมูล

เราต้องแปลงข้อมูลจากข้อมูลประเภทหนึ่ง ไปเป็นข้อมูลในอีกประเภทหนึ่ง หรือคือการเปลี่ยน Data Type นั่นเอง เช่นการเปลี่ยน ตัวแปรที่เป็นตัวเลข เป็น String จาก String เป็น วันที่ ทั้งเพื่อการคำนวณ และ การเปรียบเทียบ เป็นต้น

Function การแปลงข้อมูลทั่วไป

Function	คำอธิบาย	ตัวอย่าง

Cbyte	แปลงจากตัวแปรประเภทอื่นให้เป็น Byte โดยการปัดเศษ	`I = "1.6"`
		`Msgbox(Cbyte(I))`
		`=2`

CDbl	แปลงจากตัวแปรประเภทอื่นให้เป็น Double	`I = "1.6"`
		`Msgbox(CDbl(I))`
		`=1.6`

Cint	แปลงจากตัวแปรประเภทอื่นให้เป็น Integer โดยการปัดเศษ	`I = "1.6"`
		`Msgbox(CInt(I))`
		`= 2`

CLng	แปลงจากตัวแปรประเภทอื่นให้เป็น Long โดยการปัดเศษ	`I = "1.6"`
		`Msgbox(CLng(I))`
		`=2`

Cbool	แปลงจากตัวแปรประเภทอื่นให้เป็น Bool	`I = "1.6"`
		`Msgbox(Cbool(I))`
		`= True`

Cdate	แปลงจากตัวแปรประเภทอื่นให้เป็น Date	`I = "05/05/1969"` `If I = Date() then`

Int	แปลงจากตัวแปรประเภทอื่นให้เป็น Integer **โดยไม่ปัดเศษ แต่ถ้าตัวแปรนั้นมีค่าเป็น ลบ จะทำการ ปัดเศษ**	`I = 1.6`
		`Msgbox(int(I))`
		`Msgbox(int(-I))`

Fix	แปลงจากตัวแปรประเภทอื่นให้เป็น Integer **โดยไม่ปัดเศษ ต่างกับ Int ตรงที่จะไม่ปัดเศษ แม้เป็นค่าลบ**	`I = 1.6`
		`Msgbox(Fix(I))`
		`Msgbox(Fix(-I))`

VAL	แปลงตัวอักษรให้เป็นค่าตัวเลข ถ้ามีตัวอักษรปนอยู่กับตัวเลข Val อาจให้ค่าเป็น 0 ถ้าตัวอักษรนำหน้าตัวเลข	`Val("asd12") = 0`
		`Val("12asd12") = 12`
		`Val("12.23") = 12.23`

DateSerial	ใช้ในการแปลง ค่าปีเดือนวัน ให้เป็นตัวแปรแบบวันที่ โดยมีการกำหนดค่าให้ Function นี้ 3 ค่า คือ ปี, เดือน , วัน
	`DateSerial(Year,Month,Day)`
		`YX = 1999`
		`MX = 12`
		`DX = 10`
		`Msgbox(DateSerial(YX,MX,DX))`

Function	คำอธิบาย	ตัวอย่าง
DateValue	แปลง String ที่มีการจัดเรียงแบบวันที่ ให้เป็นตัวแปรวันที่ `DateValue(String)`	`msgbox(DateValue("12/10/1999"))`

TimeSerial	ใช้ในการแปลงค่าชั่วโมง นาที วินาที เป็นตัวแปรวันที่ เฉพาะตรงส่วนของเวลา
	`TimeSerial(Hour,Minute,Sec)`
		`HX = 12`
		`MX = 10`
		`SX = 05`
		`Msgbox(timeSerial(HX,MX,SX)`

TimeValue	แปลง String ที่มีการจัดเรียงแบบเวลา ให้เป็นตัวแปรแบบวันที่ ในส่วนของเวลา `TimeValue(String)`	`Msgbox(TimeValue("12:23"))`

- FORMAT

เราใช้ Format ในการแปลงข้อมูลเช่นกัน แต่เป็นการแปลงจากตัวแปรอื่นๆ ให้เป็นตัวอักษร โดยเราสามารถกำหนดรูปแบบตัวอักษรในการแสดงได้ เหมือนที่เรากำหนดไว้ใน Format Properties ของ Form หรือ Report

Syntax

```
Format(Variable,Picture)
```

การกำหนด Picture หรือรูปแบบการแสดงผลของ String ที่ Convert แล้ว จะใช้ตัวอักษรพิเศษในการบังคับ ตัวอย่างเช่น Format(Now(),"hh:nn dd/mm/bbbb") จะได้เป็น 15:01 05/05/2542 (สมมุติว่าขณะนี้เป็นวันเวลาดังกล่าว) ตัวอักษรพิเศษที่ใช้กำหนดคือ

กลุ่มวันเวลา

เครื่องหมาย	คำอธิบาย
: (colon)	ใช้เป็นตัวขั้นเวลา
/	ใช้เป็นตัวขั้นวันที่
C	บังคับให้ใช้ตามแบบของ System Date ใน Windows
D	บังคับให้แสดงตัวเลขวันที่ 1 – 31 ไม่ต้องใส่ 0 นำหน้ากรณีที่เป็นวันที่น้อยกว่า 10
Dd	บังคับให้แสดงวันที่ 01-31 ใส่ 0 นำหน้าถ้าวันที่น้อยกว่า 10
Ddd	ให้ค่าวันในสัปดาห์ เป็นตัวย่อ (Mon, Tue,..)
Dddd	ให้ค่าวันในสัปดาห์เป็นตัวเต็ม (Monday, Tuesday ,..)
Ddddd	บังคับให้ใช้ Short Date ที่กำหนดไว้ใน Windows
Dddddd	บังคับให้ใช้ Long Date ที่กำหนดไว้ใน Windows
W	ให้ค่า 1- 7 ของวันในสัปดาห์

เครื่องหมาย	คำอธิบาย
Ww	ให้ค่าลำดับสัปดาห์ที่ ใน 1 ปี (1 to 53).
M	บังคับให้แสดงเดือน 1-12 ไม่ใส่ 0 นำหน้าถ้าไม่ถึงเดือน 10
Mm	บังคับให้แสดงเดือน 01-12 ใส่ 0 นำหน้าถ้าไม่ถึงเดือน 10
Mmm	ชื่อเดือนเป็นตัวย่อ 3 ตัวอักษร (Jan, Feb,..)
Mmmm	ชื่อเดือนเป็นตัวเต็ม (January, February,…)
Q	แสดงลำดับที่ไตรมาศของปี
Y	Number of the day of the year (1 to 366).
Yy	แสดง 2 หลักสุดท้ายของปี
Yyyy	แสดงปีเป็น 4 หลักเต็ม
H	แสดงชั่วโมง ไม่มี 0 นำหน้าถ้าน้อยกว่า 10
Hh	แสดงชั่วโมง มี 0 นำหน้าถ้าน้อยกว่า 10
N	แสดงนาที ไม่มี 0 นำหน้าถ้าน้อยกว่า 10
Nn	แสดงนาที มี 0 นำหน้าถ้าน้อยกว่า 10
S	แสดงวินาที ไม่มี 0 นำหน้าถ้าน้อยกว่า 10
SS	แสดงวินาที มี 0 นำหน้าถ้าน้อยกว่า 10
Ttttt	กำหนดให้แสดงตาม Long Time ของ Windows
AM/PM	แสดงค่า AM/PM เป็นตัวใหญ่
Am/pm	แสดงค่า am/pm เป็นตัวเล็ก
A/P	แสดงค่า A/P เป็นตัวใหญ่
a/p	แสดงค่า a/p เป็นตัวเล็ก
AMPM	แสดง AMPM ตามที่กำหนดใน Windows
ว	วันที่เป็นเลขไทย ไม่มี 0 นำหน้า ถ้าน้อยกว่า 10
วว	วันที่เป็นเลขไทย มี 0 นำหน้า ถ้าน้อยกว่า 10
ววว	วันที่ไทยในสัปดาห์ เป็นตัวย่อ
วววว	วันที่ไทยในสัปดาห์ เป็นตัวเต็ม
ววววว	วันที่ไทยตาม Short Date
วววววว	วันที่ไทยตาม Long Date
ด	เดือนเป็นเลขไทย ไม่มี 0 นำหน้าถ้าน้อยกว่า 10
ดด	เดือนเป็นเลขไทย มี 0 นำหน้าถ้าน้อยกว่า 10
ดดด	ชื่อเดือนไทยเป็นตัวย่อ
ดดดด	ชื่อเดือนไทยเป็นตัวเต็ม

เครื่องหมาย	คำอธิบาย
ปป	ปีพุทธศักราช เป็นเลขไทย 2 หลัก
ปปปป	ปีพุทธศักราช เป็นเลขไทย 4 หลัก
BB	ปีพุทธศักราช เป็นเลขอาราบิค 2 หลัก
BBBB	ปีพุทธศักราช เป็นเลขอาราบิค 4 หลัก

กลุ่มตัวเลข

กลุ่มตัวเลขสามารถกำหนดได้ซับซ้อนกว่าปกติ สามารถกำหนดได้เป็น 4 แบบ ตามค่าของตัวแปร การกำหนด Format ของทั้ง 4 แบบ จะขั้นด้วย ; (Semi-Colon) เช่น Format(X,"0.00;(0.00);-;?) ทดลอง Run Program ต่อไปนี้ แล้ว Step ดูทีละขั้นจะเข้าใจมากขึ้น

```
Sub TestFormat()

    Dim i

    Do Until i = 2

       MsgBox (Format(i, "0.00;(0.00);-;?"))

       If IsEmpty(i) Then
           i = -2
       End If

       i = i + 1
    Loop

End Sub
```

การกำหนดแบบการแสดงผลทั้ง 4 ตามลำดับคือ

- แบบ 1 สำหรับผลเมื่อค่าเป็นบวก
- แบบ 2 สำหรับผลเมื่อค่าเป็นลบ
- แบบ 3 สำหรับค่าเป็น 0
- แบบ 4 สำหรับค่าเป็น Null หรือ Empty

(Funtion isEmpty ใช้สำหรับตรวจค่าว่า ตัวแปรนั้น เคยกำหนดถูกกำหนดค่าหรือยัง)

สำหรับเครื่องหมายที่ใช้กำหนดรูปแบบการแสดงบ่อยๆ ในทั้ง 4 แบบคือ

เครื่องหมาย	คำอธิบาย
. (period)	ระบุตำแหน่งจุด ทศนิยม
, (comma)	ระบุ Commar กั้นหลัก 1000
0	บังคับให้แสดงตัวเลข ถ้าไม่มีค่าก็จะแสดง 0 เช่น Format(1,"000") จะได้เป็น 001
#	ไม่บังคับการแสดงตัวเลข ถ้าค่านำหน้า หรือทศนิยมไม่มี ก็จะไม่แสดง 0 นำหน้า
$	แสดงเครื่องหมาย $

%	บังคับให้ หาร 100 และมีการแสดงเครื่องหมาย %

กลุ่ม Function String

ใช้ในการจัดการกับตัวแปรที่เป็น String เช่นการตัดคำ Scan หาคำเป็นต้น

Function	คำอธิบาย	ตัวอย่าง

MID	ใช้ในการตัดคำบน String ณ ตำแหน่งที่กำหนด โดยระบุจำนวนที่ตัวอักษรที่ต้องการ
	`Mid(String, StartAt, Length)`
	ถ้าไม่มีการระบุความยาวตัวอักษรที่ต้องการ จะหมายถึงเอาตัวอักษรตั้งแต่ StartAt จนถึงตัวสุดท้าย
		`Dim Strx`
		`Dim StrTmp`
		`StrX = "STU10046AB"`
		`StrTmp = Mid(Strx,4,5)`
		`Msgbox(StrTmp)`
		`StrTmp = Mid(Strx,4)`
		`Msgbox(StrTmp)`

Left	ตัดตัวอักษรจากซ้าย ตามจำนวนตัวที่กำหนด มีรูปแบบคือ
	`Left(String,Length)`
	คล้ายกับคำสั่ง Mid ที่เริ่มจากตัวแรกเสมอ
		`Dim Strx`
		`Dim StrTmp`
		`StrX = "STU10046AB"`
		`StrTmp = Left(Strx,3)`
		`Msgbox(StrTmp)`

Right	ตัดตัวอักษรจากขวา ตามจำนวนตัวที่กำหนด มีรูปแบบคือ
	`Right(String,Length)`
		`Dim Strx`
		`Dim StrTmp`
		`StrX = "STU10046AB"`
		`StrTmp = Right(Strx,2)`
		`Msgbox(StrTmp)`

Instr	ใช้ในการ Scan หาตำแหน่งของตัวอักษรที่ต้องการค้นหาใน String เช่นเราต้องการทราบว่า มี "." อยู่ในตัวแปร String ที่ตำแหน่งใด กำหนด 3 Input คือ ตำแหน่งเริ่มต้น , String ที่ถูกค้น, คำที่ต้องการค้น
	`Instr(Start,StrSearch,StrX)`
		`Dim StrTmp, Sname`
		`StrTmp = "COMMAND.COM"`
		`Sname = Mid(StrTmp, 1, InStr(1, StrTmp, ".") - 1)`
		
		`MsgBox (Sname)`

Trim	ตัด Space หน้าหลังของตัวอักษร
	`Trim(String)`
		`Dim Strx`
		`Strx ="  HELLO  "`
		`Msgbox("-" & Strx & "-")`
		`Strx = Trim(Strx)`
		`Msgbox("-" & Strx & "-")`

Ltrim, RTrim	ตัด Space หน้าหน้าสำหรับ Ltrim และ Space หลังสำหรับ Rtrim	`Dim Strx`
		`Strx ="  HELLO  "`
		`Msgbox("-" & RTrim(Strx) & "-")`
		`Msgbox("-" & LTrim(Strx) & "-")`

Function	คำอธิบาย	ตัวอย่าง
	`Rtrim(String)` `Ltrim(String)`	
Len	Return ค่าความยาวของ String `Len(String)`	`Msgbox(Len("Hello"))` `Msgbox(Len("Hello-")`
Lcase	ปรับตัวอักษรให้เป็นตัวเล็กทั้งหมด `Lcase(String)`	`Msgbox(Lcase("Hello"))`
Ucase	ปรับตัวอักษรให้เป็นตัวใหญ่ทั้งหมด `Ucase(String)`	`Msgbox(Ucase("Hello"))`

กลุ่ม Function วันที่ / เวลา

ใช้ในการจัดการเกี่ยวกับวันที่เวลา การประมวลผลเกี่ยวกับเวลา ซึ่ง Function เหล่านี้ จะช่วยหักลบ ปี ที่มี 27, 28 วันให้อัตโนมัติ เราใช้ในเชิงธุรกิจมากเช่น กำหนดวัน Due ชำระเงิน หรือกำหนดงวดบัญชีเป็นต้น

Function	คำอธิบาย	ตัวอย่าง
Now	ให้ค่าวันที่ เวลา ปัจจุบัน ไม่ต้องกำหนดค่า input	`Msgbox(Now)`
Time	ให้ค่าเวลาปัจจุบัน ไม่ต้องกำหนดค่า input	`Msgbox(Time)`
Date	ให้ค่าวันที่ปัจจุบัน ไม่ต้องกำหนดค่า input	`Msgbox(Date)`

Year, Month, Day, Hour, Minute, Second, Weekday	ให้ค่า ปี เดือน วัน ชั่วโมง นาที วินาที และวันที่ในสัปดาห์ กำหนดค่าเป็นตัวแปรที่เป็นวันที่	`Year(Now)`
		`Month(Now)`
		`Day(Now)`
		`Hour(now)`
		`Minute(Now)`
		`Second(Now)`
		`WeekDay(Now)`

DateAdd	ใช้ในการบวกค่าวันเวลา มี 3 Input คือ กำหนดหน่วยที่ต้องการบวก, จำนวนที่ต้องการบวก, ค่า หรือ ตัวแปร Date ที่ทำการ
	`DateAdd(Unit, Number,Date)`
	กรณีที่ต้องการลบ เราจะใช้ Number เป็นค่าติดลบ Unit ที่ใช้ ให้ดูตาราง Unit ที่แสดงถัดไป
		บวกวันปัจจุบันไปอีก 30 วัน
		`DateAdd("d",30,Date())`
		ย้อนวันไป 2 สัปดาห์ที่ผ่านมา
		`DateAdd("ww",-2,Date())`
		เวลาอีก 30 นาทีข้างหน้า
		`DateAdd("n",30,Time())`

Function	คำอธิบาย	ตัวอย่าง

DateDiff	ใช้ในการหาค่าความแตกต่างของช่วงเวลาที่กำหนด โดยกำหนดหน่วยที่ต้องการทราบกับตัว วันเวลา 2 ตัวที่ต้องการหาความแตกต่าง
	`DateDiff(Unit,DateTime1,DateTime2)`
		หาจำนวนชั่วโมงที่ผ่านไปนับจากเช้าวันนี้
		`DateDiff("H",Date(),Now())`
		จำนวนวันที่เหลือก่อนถึงปี 2000
		`DateDiff("D", Date, DateValue ("1/1/2000"))`

DatePart	คล้ายกับ Format บน Date ใช้ในการหาค่าวันเวลา ตามหน่วยที่เรากำหนด โดย Return ค่าเป็น Variant แบบ Integer (Format ให้ค่าเป็น String) `DatePart(Unit,DateTime)`	`Msgbox(DatePart("D",Date()))` `Msgbox(DatePart("H",Date()))`

ตารางแสดงการกำหนดหน่วยของวันเวลา

หน่วย	คำอธิบาย
yyyy	ปี
q	ไตรมาส
m	เดือน
y	วันในหนึ่งปี
d	จำนวนวัน
w	วันที่ของสัปดาห์ (WeekDay)
ww	สัปดาห์ที่
h	ชั่วโมง
n	นาที
s	วินาที

กลุ่ม Function ดึงข้อมูล

กลุ่ม Function นี้ใช้ในการเปิดฐานข้อมูล แล้วดึงค่าตามที่กำหนด เรียกอีกอย่างว่า Domain Aggregate Function กลุ่มนี้ Function นี้ จะคล้ายกับการใช้ Query แต่ทำเป็น Function เพื่อใช้ในการดึงค่า และ Return ค่า แบบ Function เช่นการดึงชื่อลูกค้าจากฐานข้อมูล โดยค้นหารหัสเป็นต้น การใช้ Function นี้ค่อนข่างยาก และต้องเข้าใจการต่อ String เป็นอย่างดี

โดย Default จะประกอบด้วย 3 Input คือ

1. ส่วนแรก (Field / Expression) เป็นค่าที่ต้องการอ่านซึ่งอาจเป็นชื่อ Field หรือ Expression ที่เราเคยทำใน Query
2. ส่วนที่ 2 (Domain- Table / Query) เป็นชื่อ Table หรือ Query ที่ต้องการเปิดข้อมูล
3. ส่วนที่ 3 (Criteria) เป็น Option ใช้ระบุ Criteria ที่ต้องการค้นหา การระบุในส่วนนี้ จะบอกเป็นเงื่อนไขเหมือนที่เรากำหนด IF

วิธีกำหนดค่า

การกำหนดค่าลงในทั้ง 3 ส่วน จะต้องระบุเป็น String ตัวอย่างเช่นการอ่านชื่อสินค้า (ProductName) จาก Table Products จากรหัสของสินค้า (ProductID)

```
Dim tmpStr as String
TmpStr = Dlookup("ProductName","Products","ProductID = 19")
Msgbox(TmpStr)
```

จะเห็นว่าทั้ง 3 Input จะต้องมีเครื่องหมาย " คลุม เพื่อให้ VB ทราบว่าส่วนที่อยู่ระหว่าง " เป็นค่าที่ต้องการส่งผ่าน ถ้าเราใส่ว่า

```
Dlookup(ProductName,"Products","ProductID = 19")
```

VB จะคิดว่า ProductName เป็นตัวแปร ทำให้เกิด Error เพราะ VB ยังไม่รู้จักตัวแปรชื่อ ProductName ทดลองดูคำสั่งต่อไปนี้ให้ความหมายเดียวกัน แต่ผ่านค่า String ที่ต้องการผ่านตัวแปร แล้วแทนตัวแปรลงใน Function อีกต่อหนึ่ง

```
Dim tmpStr As String
Dim strScope As String
Dim strField As String
Dim strWhere As String

strScope = "ProductName"
strField = "Products"
strWhere = "ProductID = 19"

tmpStr = DLookup(strScope, strField, strWhere)
MsgBox (tmpStr)
```

การ Where

ตัวอย่างด้านบนเป็นการ Where โดยใช้ ProductID ซึ่งเป็นตัวเลข ถ้ากรณีที่ Field ที่กำลัง Where เป็น String จะต้องกำหนดขอบเขต String ด้วย ' แทนการใช้ " แบบปกติ ตัวอย่างเช่น

```
Dim tmpStr As String
tmpStr = DCount("Country", "Suppliers", "Country = 'USA'")
MsgBox (tmpStr)
```

ถ้าเราใช้ " แทน ' VB จะคิดว่า Input ที่ 3 จบที่เครื่องหมาย = พอดี แล้วส่วนที่เหลือหลังจากนั้นจะเป็นอะไร ? ผลก็คือ เราจะได้รับ Error จาก VB

เราสามารถกำหนดเงื่อนไขเพิ่มเติมบนส่วน Where ได้เหมือนการกำหนดเงื่อนไขทั่วไป เช่นการใช้ AND, OR, NOT สามารถใช้การเปรียบเทียบได้ทุกแบบ เช่น >=, >,< ,<=, Like เป็นต้น

```
Dim tmpStr As String
tmpStr = DCount("Country", "Suppliers", "Country = 'USA' and SupplierID > 10")
MsgBox (tmpStr)
```

ถ้าเราไม่กำหนดเงื่อนไขที่ 3 ก็เท่ากับไม่มีการ Where ใดใด Scope ของข้อมูลก็จะเป็นทั้ง Table หรือทั้ง query กรณีคำสั่ง Dlookup จะไม่สามารถบอกได้ว่า ค่าที่ได้ จะเป็นค่าใด หากมีข้อมูลมากกว่า 1 Record ในการ Lookup ครั้งนั้น

การกำหนด Field ที่ต้องการ

เราสามารถกำหนดค่าที่ต้องการในเงื่อนไขที่ 1 มากกว่า การกำหนดแค่ Field เพียง Field เดียว ตัวอย่างเช่น การอ่านชื่อประเทศ และ เมืองพร้อมกัน

```
Dim tmpStr As String
tmpStr = DLookup("City & ' - ' & Country", "Suppliers", "SupplierID = 12")
MsgBox (tmpStr)
```

สามารถกำหนดเป็นสูตร บวก ลบ คูณ หาร ได้ตามปกติ แต่เรามักใช้ในการอ่านค่า Field มากกว่า 1 Field พร้อมๆ กัน อย่างกรณีที่ยกตัวอย่างเป็นต้น เพราะเราสามารถอ่านค่า แล้วนำมาคำนวณภายหลังจากคำสั่ง พวกนี้ได้ การอ่านค่า Field หลาย Field พร้อมกัน แทนที่จะใช้ Dlookup หลายครั้ง จะช่วยให้ระบบทำงานเร็วขึ้น เพราะการใช้ Function กลุ่มนี้แต่ละครั้ง VB จะต้องเปิดฐานข้อมูล และค้นหาข้อมูลให้เราใหม่ทุกครั้ง

Function	คำอธิบาย	ตัวอย่าง
Dlookup	สำหรับการดึงค่าโดยตรง	`Dlookup("ProductName","Products", "ProductID = 19")`
Dsum	หาค่ารวมของ Field ที่กำหนด	`Dsum("UnitInStock","Products","ProductID >10"`
Dcount	นับจำนวนของ Field ที่กำหนด (ไม่รวมค่า Null)	`DCount("UnitInStock","Products","ProductID >10"`
Davg	หาค่าเฉลี่ยของ Field ที่กำหนด	`Davg("UnitPrice","Products","ProductID >10"`
Dmin	หาค่าต่ำสุด	`Dmin("UnitInStock","Products","ProductID >10"`
Dmax	หาค่าสูงสุด	`Dmax("UnitInStock","Products","ProductID >10"`

ในการใช้งานกลุ่มฟังก์ชั่นเหล่านี้ จะมีการกำหนด INPUT ที่เป็น String ที่ซับซ้อน มากกว่าการผ่านค่า String ปกติ ดังตัวอย่างที่แสดง

```
Dim tmpStr As String
Dim strX as String
Strx = "USA"
tmpStr = DCount("Country", "Suppliers", "Country = '" & Strx & "'")
MsgBox (tmpStr)
tmpStr = DCount("Country", "Suppliers", "Country Like  '" & mid(Strx,1,1)  & "*'")
MsgBox (tmpStr)
```

จะพบว่าการแทนค่าตัวแปร Strx และเชื่อม String ด้วย & เพื่อให้ชุด String ที่ 3 มีค่าเท่ากับ "Country ='USA'" แต่ค่า USA อยู่ในตัวแปร จึงต้องตัด String เป็นชิ้นๆ และประกอบเข้ากับตัวแปร StrX ส่วนคำสั่งที่ 2 เป็นการ Where แบบใช้ Like มีการใช้คำสั่ง Mid มาช่วยในการตัดคำ เพื่อต่อกับ * ในชุด String ถัดมาได้

กลุ่ม Function ตรวจสอบ

บ่อยครั้งที่เราต้องตรวจสอบค่าของตัวแปร ก่อนทำการประมวลผลคำสั่งบางอย่าง เพื่อให้แน่ใจว่า ค่าที่เก็บไว้ในตัวแปรนั้น สามารถใช้ในการประมวลผลต่อไปได้ คำสั่งกลุ่มนี้มักใช้ร่วมกับ If อยู่เสมอ

Function	คำอธิบาย	ตัวอย่าง

IsNull	ตรวจสอบตัวแปรว่ามีค่าเป็น Null หรือไม่
	`ISnull(Variable)`
	เรามักใช้กับการตรวจสอบค่าที่อ่านจาก Database ซึ่งจะกล่าวถึงในลำดับต่อไป
		ตัวอย่างการทำ Function ในการตรวจสอบค่า Null ของตัวแปร ถ้าเป็น Null จะ Return ค่า 0 กลับไปให้
		`Function ntz(nIn)`
		`If isnull(nnin) then`
		`Ntz = 0`
		`Else`
		`Ntz = nin`
		`End if`
		`End Function`

IsEmpty	ตรวจสอบว่ามีการกำหนดค่าลงในตัวแปรที่เป็น Variant แล้วหรือยัง ถ้ายัง จะมีค่าเป็น Empty
	`IsEmpty(Variable)`
		ตัวอย่างการทำ Function ในการตรวจสอบค่า Empty ของตัวแปร ถ้าเป็น Empty จะ Return ค่า 0 กลับไปให้
		`Function etz(nIn)`
		`If isEmpty(nnin) then`
		`Etz = 0`
		`Else`
		`Etz = nin`
		`End if`
		`End Function`

IsNumeric	ตรวจสอบค่าที่เก็บอยู่ในตัวแปรว่าเป็นตัวเลขหรือไม่ เรามักใช้ตรวจสอบก่อนทำการคำนวณ
	`IsNumeric(Variable)`
		`Function nAdd(nIn)`
		`If not isnumeric(nnin) then`
		`Nadd = 0`
		`Else`
		`Nadd = nIn * 10`
		`End if`
		`End Function`

Function	คำอธิบาย	ตัวอย่าง

IsDate	ตรวจสอบค่าที่เก็บอยู่ในตัวแปรว่าเป็นค่าวันที่ หรือสามารถแปลงเป็นวันที่ได้หรือไม่ เรามักใช้ตรวจสอบก่อนทำการคำนวณในส่วนของวันที่
	`IsDate(Variable)`
		`Function nDate(nIn)`
		` If not isDate(nnin) then`
		`  Ndate = Null`
		` Else`
		`  Ndate = DateAdd("d",10,nin)`
		` End if`
		`End Function`

กลุ่ม Function อื่นๆ

Function	คำอธิบาย	ตัวอย่าง
Asc	Return ค่ารหัส AscII ของตัวอักษร `ASC(Character)`	`Msgbox(Asc("A"))`

Abs	ให้ค่า Absolute , คือค่าที่ไม่มีบวกลบของตัวเลข
	`ABS(Character)`
		`Dim I as Integer`
		`I = -10`
		`Msgbox(Abs(I))`
		`I = 10`
		`Msgbox(Abs(I))`

Chr	ให้ค่าตัวอักษรตามรหัส AscII ที่กำหนด `Chr(AsciiNumber)`	`Msgbox(Chr(56))`
InputBox	ใช้ในการอ่านค่าจาก User โดยจะขึ้นกล่องเพื่อรับข้อความ สามารถกำหนด ค่า Default, TitleBar ได้ `InputBox(Prompt,Title,Default)`	`Inputbox("Enter Year","Access",1999)`
IIF	เหมือนคำสั่ง IF แต่เป็นบรรทัดเดียว ใช้ในการเลือกค่าตามเงื่อนไข มี 3 INPUT ได้แก่ เงื่อนไข, ค่าเมื่อเงื่อนไขเป็นจริง, ค่าเมื่อเงื่อนไขเป็นเท็จ `IIF(Condition, TrueValue, FalseValue)` เราใช้คำสั่งนี้บ่อยๆ ใน การทำ Expression บน Control ของ Form หรือ Report	`OTRate = IIF(Hour(Time())> 17,1.5,1)`
CHOOSE	เหมือนคำสั่ง SELECT CASE แต่เป็น	`Dim I as Integer`

Function	คำอธิบาย	ตัวอย่าง

	บรรทัดเดียว ใช้ในการเลือกค่าตาม ดัชนี (Index) มีจำนวน Input = 1 + N ตัว โดย N เป็นจำนวนเงื่อนไข Input ตัวแรกเป็นค่า Index ที่ต้องการตรวจสอบมีค่าตั้งแต่ 1 ถึง N ส่วน Input ที่เหลือเป็นค่าที่สำหรับเลือกตามลำดับ Index
	`CHOOSE(Index, Value1,Value2,… ValueN)`
	เราใช้คำสั่งนี้บ่อยๆ ใน การทำ Expression บน Control ของ Form หรือ Report
		`Dim tmpStr`
		`For I = 1 to 4`
		`If I <4 and I > 0 then`
		`TmpStr = CHOOSE (I,"St","Nd","Rd")`
		`Else`
		`Tmpstr = "Th"`
		`End If`
		`Msgbox(I & TmpStr )`
		`Next I`

SWITCH	เหมือนกับคำสั่ง SELECT CASE แต่เป็นบรรทัดเดียว ใช้เลือกค่าเป็นคู่ ตัวแรกเป็นค่าที่ตรวจสอบ ตัวที่สองเป็นผลที่ได้ สลับไปเรื่อยๆ เป็นคู่ `SWITCH(Ex1, Value1,Ex2, Value2,.., Exn, Valuen)`	`I = I mod 2` `Msgbox(Switch(0,"Even",1,"Odd")`

การสร้าง Function และ Subroutine

เราสามารถสร้าง Function หรือ Sub ของเราขึ้นใช้งานเองได้ โดยการสร้างไว้ที่ส่วน Declaration ของ Class Module หรือที่ Standard Module ความแตกต่างของ 2 ที่นี้คือ การมองเห็นของ Function หรือ Sub ที่เราสร้างขึ้นในทั้งโปรแกรม หรือ มองเห็นเฉพาะใน Module ที่เก็บ Function หรือ Sub นั้นเท่านั้น

การสร้าง Function

Function สามารถ Return ค่าที่ต้องการ กลับไปยังโปรแกรมที่เรียกได้ การ Return ค่าก็เพียงแต่ ใส่ค่าที่ต้องการ Return ลงในชื่อ Function นั้น มีรูปแบบคือ

SYNTAX

```
Function FunctionName(Var1 as VarType, Var2, Var3,… VarN) as VarReturn
    -----
End Function
```

VarType เป็นการระบุประเภทตัวแปรที่เป็น Input ของตัวแปรแต่ละตัว ส่วน VarReturn เป็นการระบุตัวแปรของค่าที่จะ Return เราอาจไม่กำหนด VarType หรือ VarReturn ก็ได้ ตัวอย่าง Function การตรวจสอบค่าตัวเลข ถ้าเป็น Null จะให้ค่าเป็น 0 ถ้าไม่เป็น Null จะให้ค่าเดิม

```
Function ntz(din )
    If isnull(din) then
            ntz = 0
    Else
            ntz = din
    End if
End Function
```

จากตัวอย่าง จะเห็นการ Return ค่ากลับมายังผู้เรียก โดยการแทนค่าลงบน ntz แบบมีเงื่อนไข การเรียกใช้ Function ก็ทำเหมือนฟังก์ชั่นของ Access ทั่วไป เช่น

```
Dim I as Integer
Dim  CountX as Integer
I = 99
CountX = DLookup("OrderID","Orders","CustomerID ='" & I & "'")
Countx = ntz(Countx)
Msgbox(CountX)
```

จากตัวอย่างด้านบนจะพบว่า ถ้าเราไม่เรียก ntz มาตรวจสอบค่า ก่อนการแสดงผลด้วย msgbox จะเกิด Error ทันที เนื่องจากค่าที่ dLookup ได้ มีค่าเป็น Null

เราสามารถสั่งให้ VB ออกจากฟังก์ชั่นใดฟังก์ชั่นหนึ่งทันที โดยการใช้คำสั่ง Exit Function ดังตัวอย่าง

```
Function getVAT(AMount)

        If IsNull(AMount) Then
            getVAT = 0
            Exit Function

        End If

        getVAT = AMount * 7 / 107

End Function
```

การสร้าง Sub

คล้ายกับการสร้าง Function แต่ Sub จะใช้สำหรับการทำงานบางอย่าง แต่ไม่ Return ค่าแบบ Function สามารถรับค่าตัวแปรได้แบบเดียวกับ Function

SYNTAX

```
Sub SubName(Var1 as VarType, Var2, Var3,… VarN)
    -----


End Sub
```

เวลาที่เรียกใช้ Sub ทำโดยการเรียกชื่อ Sub ดังกล่าว ตามด้วยวงเล็บเปิด ตัวแปร ตามจำนวนที่กำหนดไว้ใน Sub และวงเล็บปิด เหมือนกับ Function แต่ไม่ต้องมีตัวแปรมารับค่า นิยมใช้คำสั่ง Call นำหน้าเพื่อทราบว่าเป็น Sub พิเศษ

```
Sub TestBeep()

    Call BeepX(3)

End Sub

Sub BeepX(X)

    Dim i, j
    For i = 1 To X
        Beep
        j = Timer
        Do Until Timer - j > 1
        Loop

    Next i

End Sub
```

ตัวอย่างข้างต้น เป็นการเรียก Sub ที่ใช้ในการสร้างเสียง Beep ตามจำนวนครั้งที่กำหนด ส่วน TestBeep เป็นตัวทดสอบการเรียก Function ใน BeepX จะมีจำนวน Loop เท่ากับจำนวนที่สั่งผ่าน X มีการใช้ค่า Timer ซึ่งจะ Return ค่าเวลา หน่วยเป็นวินาที นับจากเที่ยงคืน เราสั่งให้ Loop จนกว่า ค่า Timer มีค่ามากกว่า ค่าที่เราอ่านมาพักไว้ก่อนหน้า เป็นระยะเวลา 1 วินาที จึงออกจาก Do Loop ก็จะได้ช่วงเวลา 1 เว้นระหว่าง Beep เท่ากับ 1 วินาที พอดี

กรณีที่ต้องการให้ VB ออกจาก Sub ทันที เราจะใช้คำสั่ง Exit Sub เราสามารถใช้ Exit Sub ใน Sub ที่เป็น Event ได้ด้วยเช่นกัน

การควบคุม Error

ในระหว่างที่ทำการประมวลผลคำสั่ง อาจเกิด Error ซึ่ง Error ดังกล่าว อาจเป็นผลมาจากค่าของข้อมูล ณ ขณะนั้น หรือความผิดพลาดจากการป้อนข้อมูลของ User การจัดการกับ Error ใน VB สามารถควบคุมได้ โดยสั่งให้ VB กระโดดไปที่ตำแหน่งใดตำแหน่งหนึ่ง แล้วทำการประมวลผลพิเศษ ตามปัญหานั้นๆ ใช้ คำสั่ง On Error ซึ่งมีหลายแบบคือ

ON ERROR GOTO *labelX*

บอกให้ VB กระโดดไปที่ตำแหน่งที่กำหนด เรากำหนดตำแหน่ง ในโปรแกรมของเราโดยใช้ชื่อตำแหน่งตามด้วย Colon เช่น

```
Sub getTest()
    Dim X
```

```
    X = GetOrderCount(99)
    If X <> -1 Then
        MsgBox (X)
    End If

End Sub

Function GetOrderCount(CusID)
    Dim X As Integer

    On Error GoTo Err_trap
    X = DLookup("OrderID", "Orders", "CustomerID ='" & CusID & "'")
    GetOrderCount = X

resume_trap:
    Exit Function
Err_trap:
    MsgBox ("Found Error")
    GetOrderCount = -1
    Resume resume_trap
End Function
```

ทดลอง Run โปรแกรม getTest จะพบว่า มี Error ที่ X = … ซึ่งได้ค่าเป็น Null ไม่สามารถใส่ค่าลง X ที่เป็น Integer ได้ แต่โปรแกรมจะกระโดดไปที่ Err_trap จากนั้น แสดง Msgbox แล้ว Resume ซึ่งใช้ในการสั่งให้ VB กลับไปทำงานตามเดิม แต่ให้ไปเริ่มที่ใดใดที่หนึ่ง ในที่นี้ ให้ไปเริ่มที่ resume_trap ซึ่งจะพบคำสั่ง Exit Function ก็จะออกจากโปรแกรม ทดลอง Step โปรแกรมโดยใช้ F8 แล้วลองใช้ค่ารหัส Customer ที่มีข้อมูลใน Table Order แทนลงใน X = GetOrderCount(…) เพื่อดูกรณีที่ไม่มี Error เปรียบเทียบด้วย

การที่เราสามารถกำหนดให้ VB ทำการพิเศษๆ เมื่อพบ Error ที่ตำแหน่งท้ายโปรแกรม เราจะต้องสั่งคำสั่ง Resume เพื่อให้ VB กลับไปทำงานตามเดิม โดยระบุที่ไปเสมอ และก่อนถึงตำแหน่ง Resume เราจะใส่ Exit sub หรือ Exit Function เสมอ เพื่อกันไม่ให้ VB ทำคำสั่งต่อเนื่อง ไปยังส่วนที่ตรวจ Error ในกรณีที่ ไม่พบ Error เกิดขึ้นใน Routine

เราสามารถตรวจสอบรหัสการเกิด Error ได้โดยการ ตรวจสอบค่า **err** ซึ่งจะเป็นค่าตัวเลข และ **Error** ซึ่งเป็นค่า String ของ Error นั้น ทดลองเปลี่ยนคำสั่ง msgbox(“Found Error”) เป็น

```
Msgbox("Found Error Code-" & Err & " :  " & Error)
```

จะพบว่าเราจะได้ค่า Error ของ VB ปกติ ถ้าไม่มี Error ค่า err จะเท่ากับ 0

ON ERROR RESUME NEXT

คำสั่งนี้ ระบุให้ VB ทำคำสั่งต่อไปทันที เมื่อเกิด Error เราใช้ ในการ Trap Error เองในกรณีที่เรา ทราบว่าจะเกิด Error ที่คำสั่งส่วนใดส่วนหนึ่ง แล้วต้องการจัดการกับ Error นั่นๆเอง เช่น

```
Function GetOrderCount2(CusID)
    Dim X As Integer

    On Error Resume Next
    X = DLookup("OrderID", "Orders", "CustomerID ='" & CusID & "'")
    If Err Then
```

```
        MsgBox ("Found Zero Order")
        GetOrderCount = -1
        Else
        GetOrderCount2 = X
    End If
End Function
```

ON ERROR GOTO 0

เป็นค่า Default ของการ Trap Error กล่าวคือ เมื่อเกิด Error ในระหว่างการประมวลผล VB จะแสดงข้อความ Error เอง ถ้าเราไม่กำหนด On Error ใดใดก่อนหน้า VB จะประพฤติตัวแบบนี้ เราใช้คำสั่งนี้ เพื่อ Reset ค่า On Error ที่เคยตั้งไว้ก่อนหน้า ให้กลับมาเป็นค่า Defualt เช่น

```
Function GetOrderCount2(CusID)
    Dim X As Integer
    Dim ErrTmp


    On Error Resume Next
    X = DLookup("OrderID", "Orders", "CustomerID ='" & CusID & "'")
    ErrTmp = Err
    On Error Goto 0
    If ErrTmp Then
        MsgBox ("Found Zero Order")
        GetOrderCount = -1
        Else
        GetOrderCount2 = X
    End If
End Function
```

จากตัวอย่าง เป็นการ Trap Error โดยใช้ On Error Resume Next แล้ว เก็บค่า Err ไว้ในตัวแปร ErrTmp จาก Reset ตัว On Error ให้เป็นปกติ

ขอบเขตของ On Error

ของเขตของ On Error จะอยู่ภายใต้ Sub หรือ Function เท่านั้น จะไม่กระทบกับ Sub หรือ ฟังก์ชั่นอื่นๆ แต่ถ้ามีการเรียก Sub หรือ Function (B) จาก Sub หรือ Function อื่น (A) และตัวที่เรียก(A)มีการ ประกาศ On Error Resume อะไรไว้ ส่วนตัวที่ถูกเรียก(B) ไม่มีการประกาศ On Error ไว้ หากเกิด Error ที่ตัวที่ถูกเรียก (B) โปรแกรม จะกระโดดไปที่ On Error ของตัวที่เรียก (A)ทันที ดูตัวอย่างต่อไปนี้

```
Sub getTest()
    On Error GoTo Err_trap

    Dim X

    X = GetOrderCount(99)
    If X <> -1 Then
        MsgBox (X)
    End If

resume_trap:
    Exit Sub
Err_trap:
```

```
    MsgBox ("Found Error")
    Resume resume_trap


End Sub

Function GetOrderCount(CusID)
    Dim X As Integer

    X = DLookup("OrderID", "Orders", "CustomerID ='" & CusID & "'")
    GetOrderCount = X


End Function
```

จากตัวอย่างข้างบนจะพบว่า มี Error ที่ GetOrderCount ตรงบรรทัด X= โปรแกรมจะกลับไปที่ Err_trap ทันที เพราะตัวที่เรียก มี On Error ไว้

C H A P T E R 3

FORM / REPORT MEMBER

3

FORM เป็นส่วนสำคัญของ ACCESS ที่จะใช้ในการสร้าง User Interface กับผู้ใช้โปรแกรมของเรา จะเป็นตัวที่นำผู้ใช้โปรแกรมของเราเข้าไปสู่ขั้นตอนต่างๆของการใช้โปรแกรมซึ่งจะส่งผลต่อการ Insert Update Delete ข้อมูลของเรา รวมไปถึงการเรียกใช้ Report เรียกใช้ โปรแกรมย่อยๆ เรียก Query ขึ้นมาทำงาน เราจะต้องรู้จักองค์ประกอบของ Form เพื่อใช้ VB เข้าไปควบคุมให้ทำงานตอบโต้กับผู้ใช้ และกลับเข้าไปปรับปรุงข้อมูลของเรา

ในบทนี้จะเป็นเรื่องของการนำ VB มาสอดแทรกเข้าไปใน Form ซึ่งจะเกี่ยวข้องกับ Member หรือองค์ประกอบของ Form ได้แก่ Event Method Properties

CONTENT

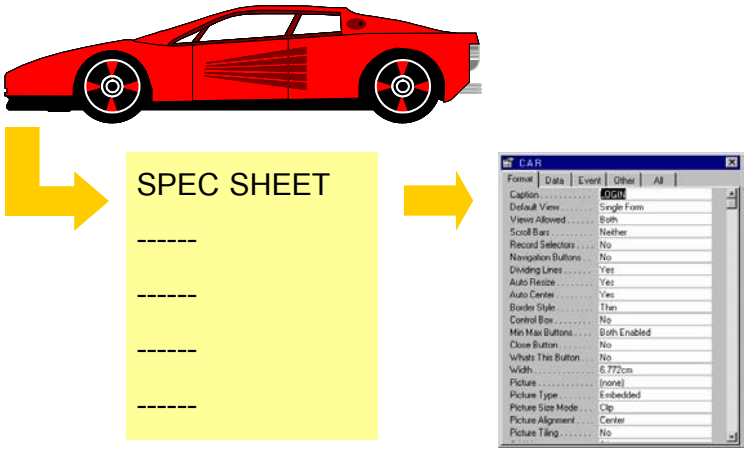
- รู้จัก Object
- การอ้างอิง Object
- กำหนด PROPERTIES
- FORM EVENT
- FORM METHOD
- FORM TIMER
- CONTROL EVENT
- SUBFORM
- REPORT EVENT

รู้จัก OBJECT

FORM เป็น OBJECT ประเภทหนึ่งของ Access ทุกๆ Object ที่มีใน Access จะประกอบด้วยองค์ประกอบหลักๆ 3 ส่วนคือ

- Properties
- Method
- Event

เพื่อให้เข้าใจง่ายขึ้นขอยกตัวอย่าง Object เป็นรถยนต์ เพื่ออเปรียบเทียบ



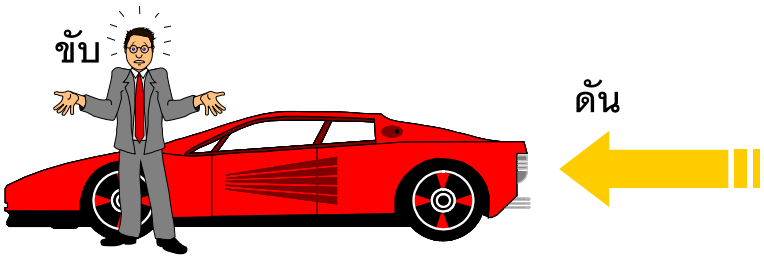
Properties

รถทุกคันมีสี มีขนาด CC มีระดับความเร็วที่วิ่งได้ สิ่งเหล่านี้ จัดว่าเป็น Properties ของ รถยนต์ Properties บางอย่างเราสามารถเปลี่ยนแปลงได้ เช่น ขนาดล้อยาง หรือสีของรถ แต่เลือกได้เฉพาะในตอนที่เราเลือกซื้อ (Design Time) Properties บางอย่างเปลี่ยนแปลงไม่ได้ เช่น อายุการใช้งาน ระยะไมล์ที่วิ่งไป เป็นสิ่งที่สะท้อนตัวของ Object เอง แต่เรารู้ว่าค่าของมันเป็นเท่าไร หรือ อย่างไรแล้ว เรียกว่าอ่านมาดูได้อย่างเดียว (ReadOnly Properties)

OBJECT ต่างๆ บน Access ก็เช่นเดียวกัน มี Properties ที่เราสามารถอ่านเขียนได้ Properties บางอย่าง เราอ่านได้อย่างเดียว บาง Properties เปลี่ยนแปลงได้ บาง Properties เปลี่ยนแปลงได้ ณ ขณะ Design เท่านั้น เหมือนกับขณะที่เราเลือกซื้อรถยนต์

ทุกครั้งที่เราเขียนโปรแกรมกับส่วนต่างๆ บน Access เราต้องตั้งคำถามว่ามีอะไรบ้างที่เราสามารถเปลี่ยนแปลงแก้ไขได้ และเปลี่ยนแปลงแก้ไขได้ ณ เวลาใดใด มี 2 ช่วงคือ DesignTime กับ Runtime ยกตัวอย่างเช่น List Box เราจะใช้ Font อะไร ใช้ อะไรเป็น RowSource อะไรเป็น RecordSource เป็นต้น

Method



กิจกรรมที่สามารถกระทำลงบน OBJECT ได้

- **METHOD ที่ให้ USER INTERFACE ทำการ**
- **METHOD ที่สั่งจากโปรแกรมได้**

เราทำอะไรกับรถยนต์ได้บ้าง ? เราสตาร์ทเครื่อง เราขับ เราเข็น เราเติมน้ำมัน เราเติมลมยาง กริยาทุกชนิดที่เราสามารถกระทำกับตัวรถยนต์ได้ จะถือว่าเป็น Method ของ Object นั้น การสั่งให้เกิด Method ต่างๆบน Object นั้นจะมีผลแตกต่างกันไป ขึ้นอยู่กับว่า Object ขณะนั้นเป็นอย่างไร เช่น ขณะนั้นรถยนต์ยางแบน การเข็น ก็ต้องเข็นยาก เป็นต้น เรียกว่าขณะนั้นรถยนต์มี Properties ของยาง เป็น แบน Method ที่เข้าไปเกี่ยวข้องกับการเคลื่อนไหวก็จะสัมพันธ์กับ Properties นี้

Object ของ Access มี Method เช่นกัน แต่ก็ไม่ทุกตัวที่มี Method ตัวอย่างเช่น Form มี Method Requery ซึ่งเมื่อสั่งให้ Requery ก็ขึ้นอยู่กับว่า Form มี Propertie ของ RowSource เป็นอย่างไร Method ที่ใช้บ่อยบน Form มีไม่มากเท่ากับ Properties การทำการบางอย่างบน Object ไม่มี Method รองรับ กล่าวคือเราสั่งให้ Object ทำการอย่างใดอย่างหนึ่งได้ เช่น การ Click บน Object เราสั่งให้ Object ถูก Click เองไม่ได้ เพราะการ Click ต้องทำโดย User ไม่สามารถใช้ โปรแกรมไปสั่งให้มันถูก Click ได้

Event

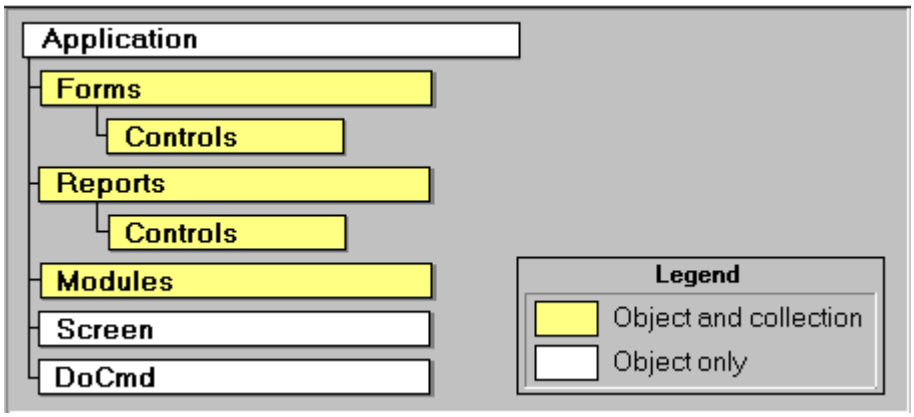
Object ทุกตัวมีการตอบโต้มากมายจากการที่เราไปกระทำอะไรกับมัน หรือแม้กระทั้งกับสิ่งมันเกิดขึ้นเอง ตัวอย่าง Event หรือเหตุการณ์เช่น พอเราสตาร์ทเครื่อง ก็จะเกิดเสียง Start แล้วระบบไฟก็จะติดติด ขณะเบรค รถยนต์ก็จะหยุด เหยียบคันเร่ง รถก็จะวิ่ง หรือขณะที่รถวิ่งไปเรื่อยก็จะมีเหตุการณ์ที่รถเตือนว่าน้ำมันใกล้หมดซึ่งเป็นสิ่งเกิดขึ้นเองจากรถ

Object บน Access จะมี Event ต่างๆ มากมาย มักตอบโต้กับ Method ที่เข้าไปทำการกับมัน หรือเมื่อเมื่อมันถูกกระทำโดยบุคคลที่ 3 ในที่นี้ก็คือ User นั้นเอง ตัวอย่างเช่น List Box เมื่อเรา Click ตัว List Box จะเกิด Event บน List Box นับตั้งแต่ GotFocus, Enter, Click, BeforeUpdate, AfterUpdate

Event ต่างๆที่เกิดขึ้น จะเปิดช่องให้เราเข้าไปเขียนโปรแกรมเข้าไปแทรกกับมัน สิ่งที่แทรกก็คือ VB CODE ที่เราเรียนรู้ไปแล้ว และนำไปกระทำกับ Method, Properties ของ Control ตัวอื่นๆ หรือตัวมันเองก็ตาม

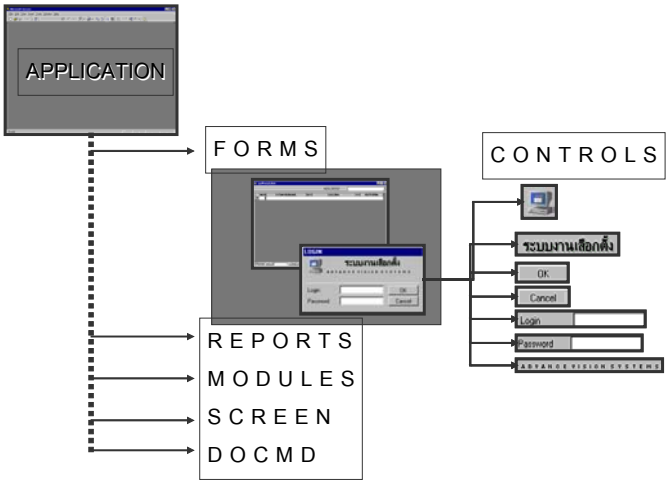
การเรียนรู้ และเข้าใจ Event ของ Object สำคัญๆ ตั้งแต่ Control แต่ละตัว จนไปถึงกับตัว Form เอง นับเป็นสิ่งสำคัญมาก รวมถึงการรู้ลำดับขั้นการเกิด Event แต่ละตัว ของในแต่ละ Control ตรงนี้จะทำให้เราเข้าใจ และสามารถสอดแทรก Code ได้ดีขึ้น

Container คืออะไร



แต่ละ Object อาจมี Object ย่อยภายใต้ตัวของมันอีกมากมาย เช่น FORM มี Control ใต้ตัวของมัน เราจะจัดว่า Control พวกนั้นเป็น Control ย่อยๆของ Form อีกต่อหนึ่ง เหมือน รถยนต์ ที่มีองค์ประกอบหลายๆ องค์ประกอบที่รวมเป็นรถทั้งคัน เช่นเบาะนั่ง ก็เป็นอีก 1 Object ที่อยู่ใต้ Object รถ มีสี ปรับที่นั่งได้ เป็นต้น Container ที่ใหญ่ที่สุดของ Access ก็คือ Application ใต้ Application เป็น Forms, Reports,.. ตามภาพที่แสดง

ใน Access จะแบ่ง ชุด Object เป็น 2 ส่วน คือส่วน Application และ DBEngine ซึ่งเป็นส่วนของฐานข้อมูลทั้งหมด ในบทที่ 5 เราจะอธิบายในส่วนที่เกี่ยวกับ Database Object อีกครั้งหนึ่ง



การอ้างอิง Object

Object แต่ละ Object สามารถอ้างอิงถึงกันโดยใช้หลักการอ้างอิงแบบ Tree กล่าวคือ ตั้งสมมุติฐานว่า ทุกวลี หรือคำสั่งที่อ้างอิงถึง Object ไม่ว่าจะเป็นส่วน Properties หรือ Method จะต้องบอกถึงตัว Container เป็นลำดับชั้นลงมา แต่ ถ้าไม่กำหนดอะไรเลย จะถือว่าใช้ Container ปัจจุบัน

ทดลองบันทึก Code

4. ทดลองสร้าง Form หนึ่ง Form แล้ว Save ใช้ชื่อ Sample
5. ใส่ Properties ของ Form ที่ caption เป็น “SAMPLE”
6. สร้าง Code เข้าไปในส่วนของ Detail_Click

```
Option Compare Database
Option Explicit
Private Sub Detail_Click()
        StrX = Application.Forms("Sample").Caption
        MsgBox ("Full Name:" & StrX)
        StrX = Forms("Sample").Caption
        MsgBox ("Form Prefix:" & StrX)
        StrX = Caption
        MsgBox ("None Prefix:" & StrX)
        StrX = Me.Caption
        MsgBox ("Use Me Prefix:" & StrX)

End Sub
```

จากตัวอย่าง จะเห็นว่าเป็นการอ่านค่า Properties Caption ของ Form มาแสดงผล ผ่าน MsgBox การอ้างอิงชื่อทำได้หลายแบบ

- บอกแบบเต็มยศ นับตั้งแต่ Application, Forms, ถ้าเป็นกรณีอ้างอิงตัว Control ก็จะต้องเพิ่มชื่อ Control เข้าไปอีกชั้นหนึ่ง เช่น (Application.Forms(“Sample”).Text0 เป็นต้น)
- บอกโดยละ Application เมื่อเราละ Application , VB ก็จะถือเอาตัว Container ที่คลุมตัว Object ณ ขณะนั้น ในที่นี่ไม่บอก ก็หมายถึง ตัว Access Application นั่นเอง
- บอกเฉพาะ Properties ซึ่งการบอกแบบนี้ VB จะดูว่าขณะนี้ เราอยู่ใน Object อะไร ก็ใช้ Caption ของ Object นั้น
- บอกโดยใช้ Me นำหน้า เป็นการระบุ ตัว Container ปัจจุบัน ซึ่งในที่นี้ ก็คือ Form

ในการเขียนโปรแกรม เราควร

- ใช้ ME….. ให้มากที่สุด เพราะ การอ้างด้วย Forms(“ชื่อ Form”) จะสร้างปัญหาให้เรา เมื่อมีการเปลี่ยนชื่อ Form ทำให้เราต้องแก้ไข Code ทั้งหมด แต่ การใช้ Me ไม่จำเป็นต้องแก้ไขใดใด เพราะเป็นการอ้างถึง Form ปัจจุบันอยู่แล้ว

- ไม่ต้องใช้ Application นำหน้า เพื่อความรวดเร็ว ยกเว้นว่าเราต้องอ้างอิงข้าม Application เท่านั้น
- ใช้การอ้างอิง Forms("ชื่อ Form") เมื่อต้องการอ้างอิงกับ Object ที่อยู่คนละ Form เท่านั้น ถ้าอยู่ใน Form เดียวกัน ให้ Me นำหน้า จะดีที่สุด

การอ้างอิง Control บน Form

การอ้างอิง Object บน Forms ใน Form คนละ Form กับปัจจุบัน เราจะอ้างอิง โดยการบอกชื่อ Form ตามด้วยชื่อ Control ตามตัวอย่าง การอ่าน ค่า Control Text0 บน Form Sample จาก Form อีก Form หนึ่ง ตัวอย่างการอ้างอิงได้แก่

วิธีอ้างอิง	คำอธิบาย
`Forms("Sample").Text0`	แบบนี้เป็นแบบที่นิยมใช้มกาที่สุด กล่าวคือ ระบุชื่อ Form ลงในวงเล็บ และมี เครื่องหมาย " คลุม ตามด้วย . และชื่อ Control
`Forms(0).Text0`	แบบนี้ใช้ดัชนีที่เป็นตัวเลขระบุ Form แต่เป็นลำดับที่ของ Form ที่เปิดอยู่ ไม่นิยมใช้ เพราะไม่ทราบแน่ว่า Form ที่กำลังอ้างอิงถูกเปิดเป็นลำดับที่เท่าไร
`Forms!Sample.Text0`	ใช้เครื่องหมาย ! เพื่อให้ทราบว่าตัวที่ตามหลังเป็นชื่อ Form สะดวก และสั้น แต่ไม่สามารถใช้งานได้ ในกรณีที่ชื่อ Form มี ช่องว่างเช่น Form ชื่อ Sample 1 เป็นต้น
`Forms![Sample].Text0`	เป็นแบบที่ใช้มาแก้ไข กรณีที่ 3 สามารถกำหนดชื่อ Form ที่มีช่องว่างได้ โดยใช้เครื่องหมายปีกกา คลุม

สาเหตุที่นิยมใช้ในแบบแรกมากที่สุด เนื่องจากค่าที่ผ่านในลงวงเล็บ สามารถใช้ตัวแปร หรือทำ String มาต่อกันเพื่อให้ได้ชื่อ Form ก็ได้ ตัวอย่างเช่น

```
Dim Fname as String
Fname = "Sample"
Msgbox(Forms(Fname).Text0)
For I = 1 to 10
    Msgbox(Forms("FORMTEST" & I).Caption))
Next I
```

ในตัวอย่างด้านบน สมมุติว่า เรามี Forms ชื่อ FORMTEST1 ถึง FORMTEST10 และต้องการอ่านชื่อ Form ทีละตัวออกมา เราสามารถใช้ตัวแปร มาควบคุมได้ เพราะ FORMTEST & I ก็คือ FORMTEST1 ถึง FORMTEST10 นั่นเอง ซึ่งถูกแทนค่าลงไปในวงเล็บที่ละขั้นตามลำดับ

กำหนด PROPERTIES

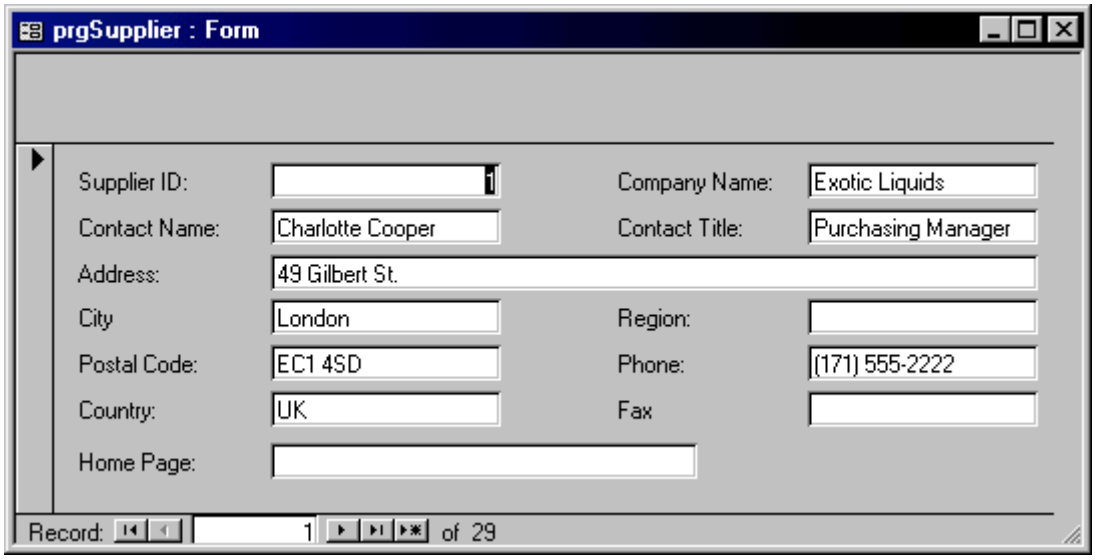
การเขียนค่าลง Form Properties ทำเหมือนการ Assign ค่าตัวแปรตามปกติ ดังตัวอย่างเช่น

```
Me.Caption = "Open Since:" & Format(Now,"dd/mm/yyyy hh:nn")
```

ทดลองใส่ Code ดังกล่าวลงใน Load Event ของ Form จะเห็นว่า การ Set ค่าสามารถใช้ Function การต่อ String เหมือนกับที่เราเรียนรู้ใน VB ทุกประการ กรณีที่เป็น String เราก็จะต้องระบุ โดยมีเครื่องหมาย " คลุม สำหรับการระบุ Propeties ของ Control บน Form ก็ทำคล้ายกันแต่ต้องระบุชื่อ Control ก่อน เช่น

```
Me.Text0.Caption = "Open Since:" & Format(Now,"dd/mm/yyyy hh:nn")
```

เรามักไม่ได้ Set Properties ของ Form มากนัก แต่มักเป็นการ Set ค่าของ Control บน Form มากกว่า



ก่อนที่เราจะข้ามไปส่วนต่อไปซึ่งเป็นการเรียนรู้การควบคุม Properties ของ Form มาทดลองสร้าง Form เพื่อใช้สำหรับเรียนรู้ในส่วนต่อไปกันก่อน

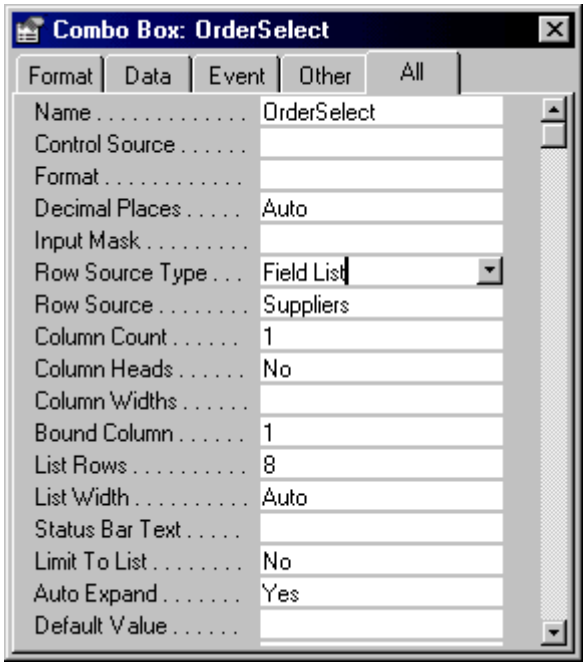
FORM สำหรับทดสอบ

1. ใช้ FILE northwind เป็นตัวอย่าง
2. สร้าง Form เปล่า 1 Form โดยใช้ Table Supplier เป็น RecordSource
3. วาง Field ลงบน FORM ตามตัวอย่าง

ORDERBY PROPERTIES

เราสามารถอำนวยความสะดวกให้กับ User โดยการให้ User สามารถเลือกการ Sort ของข้อมูลในแบบที่ต้องการ โดย สั่งที่ Propertiies ของ Form ให้มีการ OrderBy ตามกำหนด

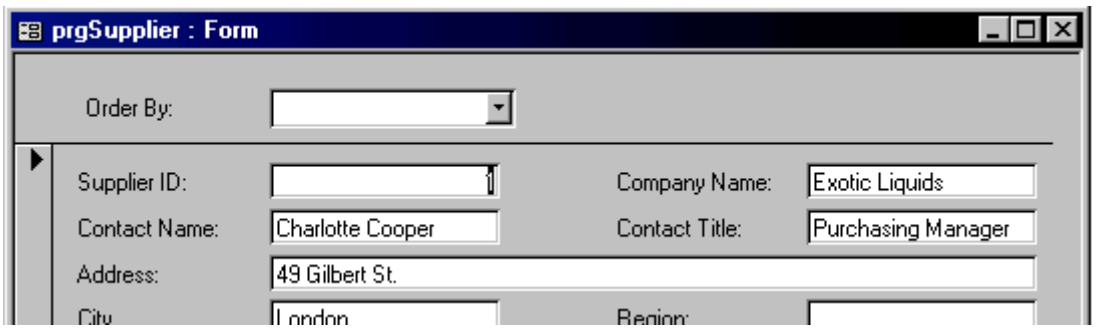
สร้าง COMBO สำหรับ OrderBy



1. สร้าง Combo Box ใหม่บน Form Header
2. Cancel Wizard (ถ้ามี)
3. กำหนด Properties ของ ComboBox: Name = OrderSelect, Row Source Type = Field List, Row Source = Suppliers ตามภาพ
4. ใส่ Code ลงใน OrderSelect_AfterUpdate ตามตัวอย่าง

```
Private Sub
OrderSelect_AfterUpdate()
            Me.OrderBy =
Me.OrderSelect
End Sub
```

5. ทดลอง Run Form และลองเปลี่ยน ลำดับการ Order ใน List Box จะพบว่า Form จะเปลี่ยนการเรียงหรือ Order By ตามที่เรากำหนด



FILTER PROPERTIES

เช่นเดียวกับกาใช้ OrderBy เราสามารถกำหนด Filter ของข้อมูลที่แสดงบน Form ได้โดยการ ระบุลงใน Filter Properties แล้วสั่งให้ทำการ Filter ทันที โดยการกำหนดให้ FilterOn = True ทดลองตามตัวอย่าง

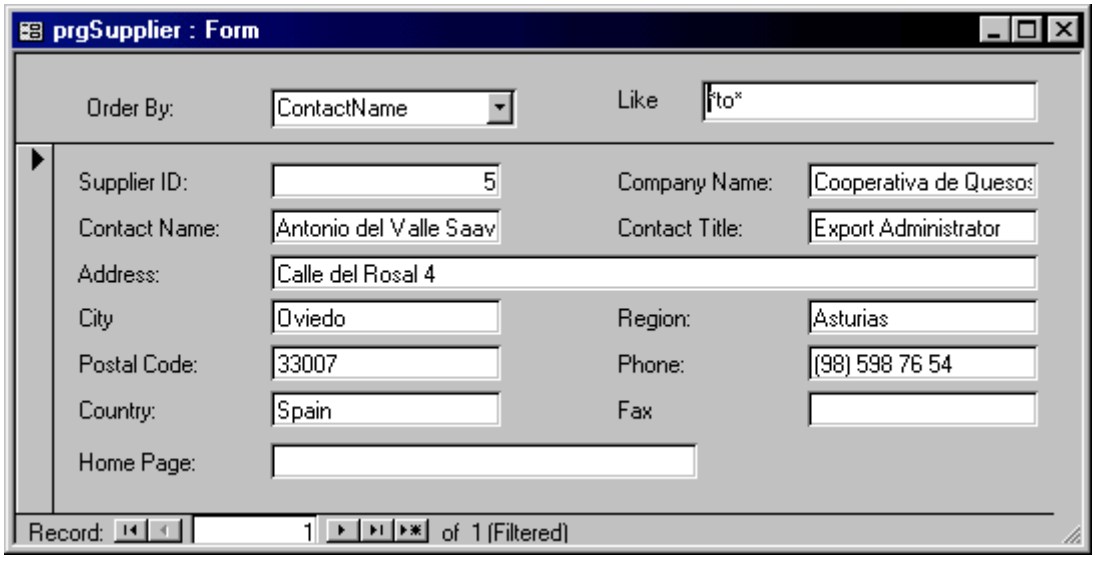
การใช้ Filter

1. สร้าง TextBox ชื่อ SearchMe ลงบน Form Header
2. บรรจุ Code ลงใน AfterUpdate ของ SearchMe ตามตัวอย่าง

```
Private Sub SearchMe_AfterUpdate()
            Me.Filter = Me.OrderSelect & " Like '" & Me.SearchMe & "'"
            Me.FilterOn = True

End Sub
```

3. ทดลอง Run Form
4. เลือกค่า OrderBy Field ซึ่งในตัวอย่างใช้เป็น Field ที่ใช้ในการ Where ด้วย
5. ระบุเงื่อนไขในการค้นหาเป็น WildCard

ตัวอย่างนี้มีการต่อ String เพื่อใช้ในการ Where เนื่องจาก Field ที่ต้องการ Where เป็น String จึงต้องสร้าง String ที่มี เครื่องหมาย ' คลุมหัวท้าย และมี " คลุมชุด String ทั้งหมด เพราะถ้าเราใช้ " ทั้งหมด Access คงเข้าใจผิดว่า " คู่แรกจบลงที่ไหน



สมมุติว่าเราต้องการค้นหา ContactName ที่มีคำว่า TO เราจะต้องสร้าง String ที่ระบุลงใน Filter เป็น "ContactName Like '*TO*'" ค่า * TO * ได้มาจากตัวแปร Me.SearchMe และค่า Field ที่ Search ได้จาก Me.OrderSelect ทำให้เราต้องตัด String เป็นท่อนๆ ดังตัวอย่าง ซึ่งจะเห็นว่าเป็น 4 ท่อนประกอบกัน

RECORDSOURCE PROPERTIES

วิธีที่สะดวกอีกวิธีหนึ่ง ในการจัดการรูปแบบการแสดงผล คือการเปลี่ยน RecordSource ของ Form โดยตรง แต่การเปลี่ยน RecordSource ของ Form จำเป็นต้องมีการใช้ SQLCOMMAND หรือใช้ Query ที่เราสร้างเอาไว้ก่อนหน้า สำหรับการ Set ในบทนี้ เราจะกล่าวถึงเฉพาะการเปลียน Record Source ไปเป็น Query ที่สร้างไว้ เราสามารถเปลี่ยน Recordsource โดยการแทนค่าเหมือน Properties ทั่วไป คือ

```
Me.RecordSource = "SupplierHaveProduct"
```

- **ทดลองเปลี่ยน** Record Source

1. สร้าง Query ชื่อ SupplierHaveProduct โดยเชื่อม Products กับ Suppliers และทำ GroupBy บน Suppliers ทุก Field (Copy Sql Command จากที่แสดงนี้ ไปลงที่ หน้า SQL ก็ได้)

```
SELECT Suppliers.SupplierID, Suppliers.CompanyName, Suppliers.ContactName,
Suppliers.ContactTitle, Suppliers.Address, Suppliers.City,
Suppliers.Region, Suppliers.PostalCode, Suppliers.Country,
Suppliers.Phone, Suppliers.Fax, First(Suppliers.HomePage) AS
FirstOfHomePage
FROM Suppliers INNER JOIN Products ON Suppliers.SupplierID =
Products.SupplierID
GROUP BY Suppliers.SupplierID, Suppliers.CompanyName,
Suppliers.ContactName, Suppliers.ContactTitle, Suppliers.Address,
Suppliers.City, Suppliers.Region, Suppliers.PostalCode, Suppliers.Country,
Suppliers.Phone, Suppliers.Fax;
```

2. สร้าง ปุ่มที่สั่งให้มีการเปลียน RecordSource แล้วใส่ Code ตามตัวอย่าง

```
Me.RecordSource = "SupplierHaveProduct"
```

3. สร้างปุ่มอีกปุ่มหนึ่ง เพื่อให้ ผู้ใช้สามารถกลับมายัง สถานะเดิม คือ Supplier ทุกคน โดยใส่ Code ว่า

```
Me.RecordSource = "Suppliers"
```

FORM EVENT

Event บน Form เป็น Event ที่สำคัญสำหรับการเขียนโปรแกรมบน Access เพราะมี Event ที่เกี่ยวข้องการการ Update ข้อมูลเข้ามาเกี่ยวข้องด้วย เราจะกล่าวถึง Event ที่สำคัญบน Form ใน Process ต่างๆ ที่อาจเกิดขึ้นได้เกิดบน Forms ดังนี้

EVENT **เมื่อเปิด** Form

เมื่อเราเรียก Form ขึ้นมาด้วยคำสั่ง Open จะเกิด Event ตามลำดับ บน Form คือ

Open -> Load -> Resize -> Activate -> GotFocus -> Current

- **OPEN EVENT**

เกิดเมื่อมีการ Open Form เราสามารถบังคับให้ยกเลิกการ Open Form ได้โดยการ Set ค่าลงในตัวแปร Cancel ปกติค่าตัวแปร Cancel จะเป็น False ถ้าเราไม่ใส่ Code อะไรไว้ Form ก็จะเปิดตามปกติ แต่ถ้าเราสั่งให้ Cancel = True Form นี้จะถูกยกเลิกการ Open, Event ตามหลังจากนี้ ก็จะไม่เกิดขึ้น เรามักใช้ Event ตัวนี้ ตรวจสอบสิทธิในการ Open Form เช่นตัวอย่างต่อไปนี้

```
Private Sub Form_Open(Cancel As Integer)
        If InputBox("Enter Magiccode") <> "6358" Then
           MsgBox ("Sorry! Magiccode was wrong")
           Cancel = True
        End If
End Sub
```

ตัวอย่างนี้ ถ้าใส่รหัสผ่าน ไม่ถูกต้อง ก็จะไม่สามารถเข้าสู่ระบบได้ Form ตัวนี้ก็จะปิดตัวเองทันที

- **LOAD EVENT**

เกิดเมื่อเริ่มทำการ Open เหมาะสำหรับการ แทรก Code ที่ใช้สำหรับตรวจสอบค่าต่างๆของระบบ เช่น User คนนี้ทำอะไรได้บ้างบนระบบ แล้วทำการจัด Form ให้ตรงกับ User นั้นๆ หรือ การแสดงวันเวลาที่ เปิด Form เป็นต้น Event นี้จะเกิดขึ้นครั้งเดียวในการ Open Form จนกว่าจะมีการปิด แล้วเปิดใหม่

- **RESIZE EVENT**

เกิดเมื่อมีการปรับเปลี่ยนขนาดของ Form ไม่ได้เกิดเฉพาะในขณะ Open Form เท่านั้น เมื่อ Form Open แล้ว แล้วมีการปรับขนาด Form ภายหลัง ก็จะเกิด Event นี้เช่นกัน

- **ACTIVATE EVENT**

เกิดเมื่อมีเรียก Form ขึ้นมาใช้งานทุกครั้งจะเกิด Event นี้ ถ้า Form ถูกซ่อนอยู่โดยยังไม่ถูกปิด และถูกเรียกกลับมาอีกครั้ง จะเกิด Event นี้เช่นกัน เวลาที่เรา Switch ระหว่าง Form ใดใด Form หนึ่งจะเกิด Event นี้ เมื่อ Form นั้นถูกเรียกขึ้นมา

- **GOTFOCUS EVENT**

เกิดเมื่อมี User เรียก Form นี้ขึ้นมา ซึ่งหมายถึงการ Focus ของ User จะกลับมาที่ Form นี้ แต่ Event นี้จะเกิด ก็ต่อเมื่อ บน Form นั้นไม่ มี Control อื่นๆเลย การ Focus ของระบบ จึงต้องไปตกกับ Form ทั้ง Form แต่ถ้ามี Control อยู่บน Form แล้ว การ GotFocus จะไปเกิดที่ Event ของ Control แทน ดังนั้นการฝัง Code ที่ให้ทำงานกับ Form สำหรับกรณีที่มีการเรียก Form ทุกครั้ง ควรใส่ไว้ใน ACTIVATE EVENT

▪ **CURRENT EVENT**

EVENT นี้ จะเกี่ยวข้องกับฐานข้อมูล กล่าวคือ เมื่อ Form ได้ทำการ Open สมบูรณ์แล้ว Form จะไปเรียกฐานข้อมูลออกมา เมื่อ Form แสดงข้อมูลลงบนจอ Record ใดใดก็ตาม จะเกิด Event นี้ ถ้า User ย้ายข้อมูลไปทีละ Record ทุกครั้งที่ย้ายไป จะเกิด Event นี้ เราใช้ Event นี้บ่อยมาก ในการตรวจสอบข้อมูลในแต่ละ Record

ทดลองดูตัวอย่างการ ตรวจสอบ เบอร์ FAX ของ Supplier เมื่อไรก็ตามที่ User เลื่อน Record และพบ Supplier ที่ไม่มีเบอร์ FAX จะมี Msgbox เตือน ทดลอง แทรก Code ชุดนี้ลงใน Current Event ของ Form ที่มี RecordSource เป็น Supplier

```
Private Sub Form_Current()
        If IsNull(Me.Fax) Then
           MsgBox ("Please ask FAX number for " & Me.CompanyName)
        End If
End Sub
```

EVENT เมื่อปิด FORM

Event ที่เกิดขึ้นขณะปิด Form จะสวนทางกับการ Open Form ถ้าขณะที่ปิด Form ข้อมูลยังไม่ถูก Save, ACCESS จะพยามยาม Save ข้อมูลก่อน ซึ่งอาจมี Event ของการ Update ข้อมูลก่อนหน้า ชุด Event นี้ สำหรับชุด ที่เกิดขึ้นขณะ ปิด Form ตามปกติคือ

Unload -> Deactivate -> Close

▪ **Unload Event**

เป็นการ Unload Object ออกจากระบบ มีค่าที่สามารถบังคับให้การ Unload นี้ดำเนินต่อไปได้หรือไม่ โดยการ Set ค่า Cancel หรือไม่ Cancel คล้ายกับการ Open ถ้า Cancel = True จะไม่สามารถปิด Form ได้ เรามักใช้ Event นี้ควบคุม User ให้บันทึก หรือทำการข้อมูลให้เสร็จสิ้นก่อนออกจาหน้าจอ

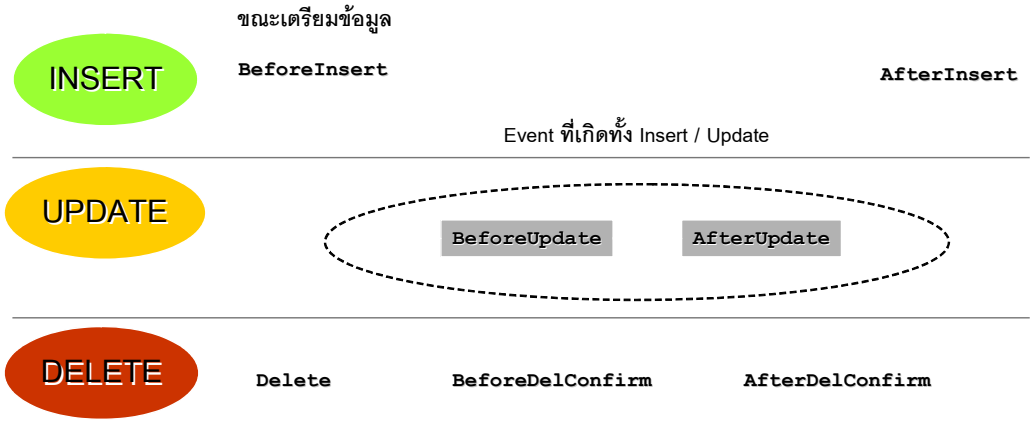
▪ **Deactivate Event**

เป็น Event ที่ตรงกันข้ามกับ Activate Event Event นี้ มิได้เกิดขึ้นเฉพาะเมื่อเราปิด Form เท่านั้น ขณะที่ เรา Switch Form ไปมา ก็จะเกิด Event นี้เช่นกัน โดยเกิดในขณะที่เราออกจาก Form ใด Form หนึ่ง

▪ **Close Event**

เกิดขึ้นเมื่อ Form ถูกปิดแล้วจริง ๆ เราใช้ทำการใดใด ที่คล้ายกับการสรุป เช่น บันทึกวันเวลาที่ User หยุดใช้ งาน Form ตัวนี้เป็นต้น

EVENT เมื่อมีการแก้ไขข้อมูล



Event ชุดนี้ จะเกิดขึ้นเมื่อมีการแก้ไขข้อมูล และข้อมูลนั้นกำลังจะเข้าไป Write เข้าไปใน Table ลำดับการเกิด Event อง Form คือ

BeforeUpdate -> AfterUpdate

- **BeforeUpdate Event**

เกิดขึ้นก่อนข้อมูลกำลังจะถูก Update เราสามารถตรวจสอบเงื่อนไขบางอย่าง ก่อนที่จะปล่อยให้เกิดการ Update จริงในข้อมูลได้ โดยการใส่ Code เข้าไปควบคุม และในขณะเดียวกัน เราสามารถยกเลิก การ Update ของ User ได้โดยการใช้คำสั่ง Cancel = True ทดลองดูตัวอย่างการบังคับให้ User สามารถ Update ข้อมูลได้ เฉพาะช่วงเวลากำหนด

```
Private Sub Form_BeforeUpdate(Cancel As Integer)
            If Time < TimeValue("08:00") Or Time > TimeValue("17:00") Then
               MsgBox ("Please update data in officehour")
               Cancel = True
            End If
End Sub
```

- **AfterUpdate Event**

Event นี้เกิดขึ้น เมื่อมีการ Update ข้อมูลแล้ว เรามักใช้ Event นี้ ในการคำนวณค่าสรุป หรือ เก็บ Log การบันทึกข้อมูลของ User เป็นต้น

```
Private Sub Form_AfterUpdate()
        MsgBox ("You Just Update A Data Record !")
End Sub
```

EVENT เมื่อมีการลบข้อมูล

ขณะเกิดการ Delete ข้อมูล เราพบว่า Access จะมีข้อความเตือนเราก่อนการลบ ทุก Step ที่เกิดขึ้น จะเป็นไปตามลำดับ Event ดังที่แสดง

Delete-> BeforeDelConfirm -> AfterDelConfirm

- **Delete Event**

เกิดขึ้นทันที ที่ User พยายามจะลบข้อมูล Event นี้เกิดก่อนที่จะมีการ Confirm โดย User เราสามารถ Cancel การลบ ได้ตั้งแต่จุดนี้ โดยให้ค่า Cancel = True

- **BeforeDelConfirm Event**

Event นี้เกิดขึ้น ก่อนที่จะมี ข้อความที่ Access ถามเพื่อการ Confirm การลบ เราสามารถกำหนดให้ Confirm การลบเองได้เลย โดยการผ่านค่าสู่ตัวแปร Response ใน Sub นี้ และ Sub นี้มีตัวแปร Cancel ให้ยกเลิกการลบได้ด้วยเช่นกัน การผ่านค่า Response มี 2 ค่าคือ

Constant	คำอธิบาย
AcDataErrContinue	ไม่ต้องแสดงข้อความเตือนของ Access ให้ Confirm การลบเลย
AcDataErrDisplay	แสดงข้อความเตือนการลบ ซึ่งเป็นค่า Default ของ Access หากไม่มีการแก้ไขอะไรก็จะเป็นเงื่อนนี้

ตัวอย่างเช่น เราไม่ต้องการให้ Access ถามข้อความเตือนในช่วงเวลา 08:00 – 17:00 เราจะระบุ Code ตามตัวอย่าง

```
Private Sub Form_BeforeDelConfirm(Cancel As Integer, Response As Integer)

            If Time > TimeValue("08:00") And Time < TimeValue("17:00") Then
               Response = acDataErrContinue
            End If

End Sub
```

- **AfterDelConfirm Event**

เกิดขึ้นหลังจากผ่านขั้นตอนทั้ง 2 แล้ว ใช้สำหรับทำการสรุปข้อมูล เช่นการตัด Stock หรือ เก็บ Log ผู้ที่ลบข้อมูล เป็นต้น เราสามารถตรวจสอบได้ว่า ผลการลบที่ผ่านมาเป็นอย่างไร โดยดูที่ตัวแปร Status ซึ่งจะมีค่าดังต่อไปนี้

Constant	คำอธิบาย
AcDeleteOK	การลบข้อมูลสำเร็จ
AcDeleteCancel	การลบข้อมูลถูกยกเลิก
AcDeleteUserCancel	การลบข้อมูลถูกยกเลิกโดย User

ตัวอย่างด้านล่างเป็นการทดสอบการ Delete ข้อมูล และแสดงข้อความตอบโต้กับผู้ใช้

```
Private Sub Form_AfterDelConfirm(Status As Integer)

    Select Case Status

    Case acDeleteOK         'Indicates the deletion was successful.
         MsgBox ("Record Deleted")
    Case acDeleteCancel     'Indicates the deletion was canceled in Visual Basic.
         MsgBox ("Record Deletion Canceled")

    Case acDeleteUserCancel  'Indicates the deletion was canceled by the user.
         MsgBox ("Record Deletion Canceled by User")
    End Select

End Sub
```

EVENT เมื่อมีการเพิ่มข้อมูล

ขณะที่เราทำการเพิ่มข้อมูลลงใน ACCESS จะเกิดชุด Event ขึ้น ซึ่งมี Event ที่ซ้ำกับการ Update ข้อมูลคือ BeforeUpdate และ AfterUpdate มีลำดับขั้นคือ

BeforeInsert -> BeforeUpdate -> AfterUpdate-> AfterInsert

ขณะที่อยู่ใน BeforeUpdate เราสามารถตรวจสอบได้ว่า ขณะนั้น เป็น Event ที่เกิดจากการแก้ไขข้อมูลเก่า หรือเกิดจากการเพิ่มข้อมูลใหม่ โดยอ่านค่า Properties NewRecord ของ Form เช่น

```
Private Sub Form_BeforeUpdate(Cancel As Integer)

        If Me.NewRecord Then
           MsgBox ("-- Inserted")

        Else
           MsgBox ("-- Updated")
        End If
End Sub
```

- **BeforeInsert Event**

Event นี้เกิดขึ้นขณะที่ User เริ่มบันทึก Stroke แรก ลงในการ Add ข้อมูล เรามักใช้สำหรับการอ่านค่าพิเศษ ที่ต้อง Write ไปในฐานข้อมูลทุกครั้งที่มีการ Add ข้อมูล คล้ายกับการทำ Default แต่มีเงื่อนไขที่ซับซ้อนกว่า การใช้ Default ปกติ Event นี้สามารถยกเลิกการ Add ข้อมูลได้โดยการใช้ Cancel = True เช่นกัน

- **AfterInsert Event**

เกิดเมื่อขั้นตอนการ Insert เสร็จสิ้นแล้ว เรามักใช้ในการทำ Summary หรือ สร้าง Log File เก็บประวัติการเพิ่มข้อมูล เป็นต้น

EVENT เมื่อมี Error

- **Error Event**

เมื่อเกิด Error บน Form จะเกิด Event ตัวนี้ ให้เราสามารถ Trap การ Error ได้ และทำการตอบโต้ Error กับ Access ได้เองผ่านตัวแปร Response โดยค่า Error จะส่งเป็น รหัสผ่านตัวแปรชื่อ DataErr

Constant	คำอธิบาย
AcDataErrContinue	ข้ามข้อความ Error ของ Access แล้วทำงานต่อไป แต่ Error ยังคงค่าอยู่ เพียงแต่ปิดข้อความ Error ของ Access เท่านั้น เราใช้สำหรับการจัดการข้อความ Error ด้วยโปรแกรมของเราเอง
AcDataErrDisplay	ค่า Default ของ Access คือ แสดงข้อความ Error ของ Access

```
Private Sub Form_Error(DataErr As Integer, Response As Integer)
    Const conDuplicateKey = 3022
    Dim strMsg As String

    If DataErr = conDuplicateKey Then
            Response = acDataErrContinue
            strMsg = "รหัสรายชื่อพนักงานจะต้องไม่ซ้ำกับรหัสเดิม"
            MsgBox strMsg
    End If
End Sub
```

ตัวอย่างข้างต้นเป็นการตรวจสอบ Error โดยบรรจุลงใน Form ที่ทำการบันทึกรหัสพนักงาน หากมีการใส่รหัสซ้ำก็จะแสดง Error เป็นข้อความภาษาไทยที่สัมพันธ์กับ Table ทดลองใช้ตัวอย่าง Code ชุดนี้กับหน้า Form บันทึกข้อมูลของ Table อื่นๆได้เช่นกัน

FORM METHOD

Form มี Method ให้เราควบคุม ซึ่งจะเกี่ยวข้องกับการแสดงผลเป็นส่วนใหญ่

Method ที่สำคัญ	คำอธิบาย	ตัวอย่าง
REPAINT	บังคับให้ปรับการแสดงผลที่ยังวาดไม่เสร็จ ให้ทำการวาดทันที ไม่เกี่ยวข้องใดใดกับข้อมูลเลย (Graphic Display Only)	`Me.Repaint`
RECAL	สั่งให้คำนวณค่าของ Control ที่เป็นสูตรทันที (Cal Control Only)	`Me.Recal`

Method ที่สำคัญ	คำอธิบาย	ตัวอย่าง
REFRESH	สั่งให้ Form Refresh ค่าบนจอโดยอ่านค่า Record ปัจจุบันที่แสดงอยู่ จากฐานข้อมูล รวมถึงบังคับให้ Form ทำการ Save ค่าในขณะนั้นทันที (Current Record Refresh)	`Me.Refresh`
REQUERY	สั่งให้ Form ทำการอ่านค่าจากฐานข้อมูลใหม่ทั้งหมด ต่างกับ REFRESH ที่ REQUERY จะทำการทั้งหมด ทุก Record เรามักใช้ กับ Form ที่เป็น DataSheet เพราะมีการแสดงข้อมูลมากกว่า 1 Record และ เมื่อมีการแก้ไขข้อมูลลบนฐานข้อมูล หรือแก้ไข RecordSource แล้วต้องการให้ แสดงค่าใหม่ ที่เก็บใน Database ทันที (All Record Refresh)	`Me.Requery`
SETFOCUS	บังคับให้ Form ถูก Focus	`Forms("Form1").Setfocus`

FORM TIMER

มี Event ที่พิเศษ สำหรับ Form คือ Timer Event ซึ่งจะทำงานร่วมกับ TimerInterval Properties เป็น Event ที่เกิดขึ้นเอง ในทุกระยะเวลาที่กำหนดใน TimerInterval มีหน่วยเป็น 1/1000 ของวินาที เราใช้ประโยชน์จาก Event นี้ในการทำงานพิเศษบางอย่างเช่น Screen Saver เมื่อไม่มีการทำการใดใดบน Form ให้ปิด Form หรือ อ่านค่าจากฐานข้อมูลตลอดเวลา เช่น สร้าง Form ที่คอย Monitor ยอดขายสินค้าเป็นต้น ทดลองทำตามตัวอย่าง

Monitor จำนวน Customer

1. สร้าง Form ใหม่ขึ้นหนึ่ง Form
2. สร้าง Control ชื่อ Text0
3. กำหนด TimerInterval ใน Properties ของ Form เป็น 2000 (2 วินาที)
4. สร้าง Code ลงใน Form_Timer

```
Private Sub Form_Timer()
        Me.Text0 = DCount("CustomerID", "Customers")

End Sub
```

5. ทดลอง Run Form
6. ทดลอง Add ข้อมูลลงในฐานข้อมูล Customers

ตัวอย่างนี้ แสดงให้เห็นว่า ในการทำงานแบบ Multiuser เราสามารถกำหนดให้ Form เข้าไปอ่านค่าใน ฐานข้อมูลขึ้นมาแสดงได้ตลอดเวลา เราสามารถใช้วิธีการดังกล่าวในการ Monitor ราคาหุ้นสูงสุด ต่ำสุด จำนวนที่นั่งที่เหลือในหนังแต่ละรอบ

จากตัวอย่างข้างต้น การ Set ค่า Timer ยิ่งน้อย จะทำให้ Access ต้องวิ่งเข้าไปอ่านข้อมูลบ่อยขึ้น ซึ่งทำให้กิน Load ของระบบงาน เพราะในขณะนั้น User คนอื่นอาจกำลัง ทำการเขียนอ่าน Table ที่เราเข้าไปอ่านอยู่เช่นกัน ควรระมัดระวังในการใช้ กลุ่มคำสั่งที่ต้องอ่านเขียนข้อมูล ใน Timer เป็นอย่างดี

CONTROL EVENT

Control ที่วางบน Form ก็มี Event เช่นเดียวกับ Form แต่ลักษณะ Event ก็แตกต่างกันไปตามประเภทของ Control เราแบ่งกลุ่ม Event ของ Control เป็น 2 กลุ่มใหญ่คือ

กลุ่ม Event ที่สัมพันธ์กับข้อมูล

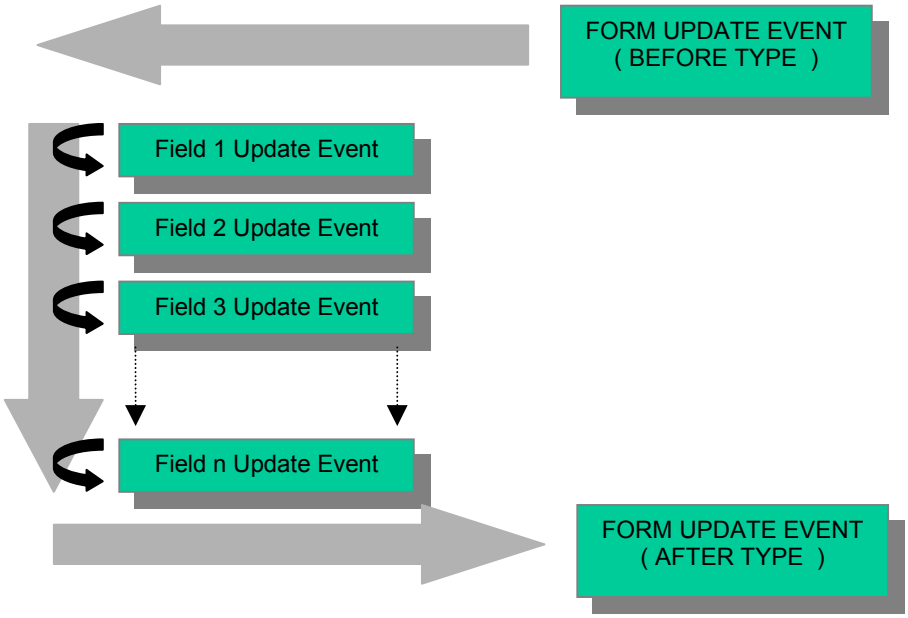
เป็น Event ที่มีในทุกๆ Control ที่ สามารถ Bound เข้ากับ Field ของ Database ได้ มี 2 Event คือ BeforeUpdate กับ AfterUpdate , Event พวกนี้เกิดระหว่างที่เริ่มมีการ Update ข้อมูลบน Form จนถึงจบขั้นตอนการ Update ข้อมูลของ Form นั้น

เราใช้ Event เหล่านี้ในการตรวจสอบข้อมูล เป็นราย Field ก่อนให้มีการ Update จริงเข้าไปใน Field นั้นๆ การ Update Field จะยังไม่เข้าไปในฐานข้อมูลจริงๆ แต่จะเก็บในหน่วยความจำสำรองของตัว Access เองก่อน จนกว่าจะทำการบันทึกจนครบทุก Field หรือ มีการ Update ทั้ง Record และ ในขณะที่กำลังจะทำการ Update ทั้ง Record ลงใน Database จริง ก็จะเกิด Event ในส่วนของ Form พวก BeforeUpdate อีกเช่นกัน

หากดูลำดับขั้นตามภาพจะพบว่า

- เมื่อเริ่มทำการบันทึกข้อมูล จะเป็น Event ของ Form พวกที่เกิดก่อนมีการ Update Record เช่น BeforeInsert
- จากนั้นก็เข้าสู่ขบวนการของ Event ในส่วนของ Control ทุก Control ก็จะมี BeforeUpdate, AfterUpdate เป็นคู่ๆ ตาม Control แต่ละตัว หากทำการที่ Control ใด ก็จะเกิด Event บน Control นั้น
- เมื่อทุก Field ที่ Bound กับฐานข้อมูลถูกบันทึก หรือ Form ถูกบังคับให้ทำการ Update , Event ของ Form ชุดที่ทำการก่อนการ Update ก็จะขึ้นมาทำงาน

เราอาจไม่ใช้ Event เหล่านี้เลย ถ้าเราไปรอการตรวจสอบข้อมูลที่ BeforeUpdate ของ Form ก็ได้ แต่ในทางปฏิบัติมีการใช้ Event ที่แทรกเพื่อทำการบน Control แต่ละตัว เพื่อให้ ทำงานได้คล่องตัวขึ้น



- **BeforeUpdate Event**

เป็น Event ที่เกิดขึ้นก่อนการเก็บค่าที่บันทึกลงไปใน Field สามารถ Cancel การบันทึกได้ โดยใช้ Cancel = True เรามักตรวจสอบค่าของ Field ก่อนที่จะผ่านให้ค่าลงไปใน Field จริงๆ ด้วย Event นี้

ในระหว่างที่กำลัง Update นี้ เราไม่สามารถทำการแก้ไข Field ที่เกิด Event นี้ได้ ตัวอย่างเช่น Code ชุดนี้ จะเกิด Error เมื่อเริ่มทำงาน เพราะพยายามแก้ไข Field Text0 บน Event BeforeUpdate ของ Text0 เอง

```
Private Sub Text0_BeforeUpdate(Cancel As Integer)
            Text0 = Text0 & "I"
End Sub
```

- **AfterUpdate Event**

เป็น Event ที่เกิดหลังจากที่ได้ทำการ Update ข้อมูลลงไปใน Field แล้ว เราสามารถทำการแก้ไข ค่าใน Field นี้ที่ Event นี้ได้ แต่การแก้ไขนี้ จะไม่เกิด Event กลับซ้อนไปที่ BeforeUpdate อีกครั้ง เราจะใช้ทั้ง BeforeUpdate และ AfterUpdate คู่กัน เพื่อตรวจสอบ และแก้ไขข้อมูล ที่เป็นระดับ Field

แก้ปัญหาวันที่ของ ACCESS ด้วย AFTERUPDATE

Access 97 Version ภาษาไทย มี Bug เรื่องวันที่ คือเมื่อกรอก วันเดือนปี ขณะที่ Regional Setting เป็น Thai อยู่ Access จะทำการสลับ วันกับเดือน กลับไปกลับมา เข้าใจว่า ทีมทำ Localize คงไม่ได้ตรวจสอบ Feature นี้ให้ดี เพราะ เวอร์ชั่นภาษาอังกฤษ ไม่มีปัญหานี้ และ ใน Version ก่อนๆ ที่เป็นภาษาไทย ก็ไม่เคยมีปัญหานี้เช่นกัน

เราแก้ไขปัญหานี้โดยการเขียน Function ในการแปลงวันเดือนปี อีกครั้งให้ถูกต้อง และ ซ่อนการแปลงดังกล่าว ไว้ใน AfterUpdate ของ Field ที่เป็น วันที่

```
Private Sub DATEFROM_AfterUpdate()
        Me.DATEFROM = DateFix(Me.DATEFROM)
End Sub

Function DateFix(dateIn)

    '-- For fixing ACCESS date and Y2K problem
    '
    '
    If IsNull(dateIn) Then
       DateFix = dateIn
       Exit Function
    End If
    Dim yx, dx

    dx = DateValue(Format(dateIn, "dd/mm/yyyy"))
    yx = Year(dateIn)
    If yx < 1970 Then
       dx = DateSerial(yx + 57, Month(dx), Day(dx)) + TimeSerial(Hour(dateIn),
Minute(dateIn), Second(dateIn))
       DateFix = dx
    Else
       dx = DateSerial(yx, Month(dx), Day(dx)) + TimeSerial(Hour(dateIn), Minute
(dateIn), Second(dateIn))
       DateFix = dx

    End If


End Function
```

Function นี้ไม่สงวนลิขสิทธิ์ ซึ่งนอกจากจะแก้ปัญหาของตัว Access เองแล้ว ยังช่วยแปลงปี พศ ไทย ให้เป็น ปี คศ ที่ถูกต้อง เพราะถ้าเรากรอก 10/10/42 Access อาจคิดว่าเป็นปี 1942 ได้ โปรแกรมจะตรวจสอบปี หากปีน้อยกว่า 1970 ผู้ใช้น่าจะกรอกผิด จะบวก 57 ปีเข้าไป แต่ถ้าไม่ใช้ ก็ แกปัญหา Access ตามปกติ โดยใช้ Function DateSerial แล้ว ถอด String จากค่าวันที่ที่ใส่มา เป็น วัน เดือน ปี ตามลำดับ

ทำไม ไม่ใช้ Function Year() ใน DateSerial ? ทำไมต้องมี ตัวแปร Yx มารับก่อน ?
เพราะ วิธีนี้ ทำให้เราเรียกฟังก์ชั่น Year เพียงครั้งเดียว ทุกครั้งที่เรียก Function จะกิน Resource ของเครื่องเพิ่มขึ้น หากต้องใช้ Function เดิม ที่อ่านค่าเดิม ควรเรียกครั้งเดียว แล้วเก็บค่าไว้ในตัวแปร จากนั้นใช้ค่าที่อยู่ในตัวแปรนั้นแทน

กลุ่ม Event ที่เฉพาะกับ Control แต่ละประเภท

กลุ่มนี้ จะเจาะจงกับลักษณะ Control แต่ละตัว เช่น ปุ่ม Command ก็จะมี Event Click เป็นต้น Event พวกนี้จะเกิดขึ้นทั้งในระหว่างที่เราทำการบันทึกข้อมูล ร่วมกับ Event ที่สัมพันธ์กับ ข้อมูล หรือเกิดขึ้นอิสระ โดยไม่ต้องมีการเริ่มต้น Update ข้อมูลบน Form ก็ได้ เช่น MouseMove แต่ Event ส่วนใหญ่มักทำงานควบคู่กันไปกับการ Update ข้อมูลเสมอ เพราะ Action ที่ไปกระทบกับ Control ก็เพื่อการ Update ข้อมูล แบ่งกลุ่ม Event ได้เป็น

กลุ่ม Mouse

เป็น Event ที่เกี่ยวกับกับ Mouse แบ่งเป็น 5 ตัว ซึ่งมี Event อยู่ 5 ตัวคือ Click, DoubleClick, MouseMove, MouseDown, MouseUp แต่ละตัวเกิดขึ้นได้ใน 3 แบบ คือ

- การย้าย Mouse ไปบน Control โดยไม่ Click จะเกิด Event **MouseMove** Event นี้จะเกิดซ้ำในระหว่างที่มีการเลื่อน Mouse
- การ Click ทุกครั้งที่มีการ Click จะเกิด Event **MouseMove,MouseDown, MouseUp, Click,MouseMove** MouseMove ตัวแรก เกิดเมื่อมีการย้าย Mouse มาบน Control เกิด MouseDown เมื่อเริ่มกดลง และเกิด MouseUp เมื่อยกนิ้วขึ้น จากนั้นเกิด Event Click ตามด้วย Event MouseMove อีกครั้งหนึ่ง หาก Mouse ไม่เลื่อนไปไหน MouseMove ตัวแรก จะไม่เกิด Event ขึ้น
- การ DoubleClick คล้ายกับการ Click แต่จะมี Event DoubleClick ตามหลังชุด Event ของ Click อีกครั้ง **MouseMove,MouseDown, MouseUp, Click, MouseMove, DoubleClick**

▪ **MouseMove,MouseDown, MouseUp**

Event ทั้ง 3 ตัวมีค่า Parameter ที่ตามหลังมาด้วย ได้แก่

- Button - ปุ่มของ Mouse ที่กด
- Shift – ปุ่มบน Keyboard ที่มีการกดขณะที่มี Event ทั้ง 3 เราสามารถใช้ปปุ่มบน Keyboard ควบคู่ไปกับการ Click ของ Mouse ได้ แบบที่ Application ทั่วไปมักทำ เช่น กด Ctrl แล้ว Click Mouse เป็นต้น
- X ตำแหน่ง X บนจอ
- Y ตำแหน่ง Y บนจอ

Button – ปุ่มที่กด	Constant	ค่า
ปุ่มซ้าย	AcLeftButton	1
ปุ่มขวา	AcRightButton	2
ปุ่มกลาง	AcMiddleButton	4

Keyboard ที่กด	Constant	ค่า
ปุ่ม Shift	AcShiftMask	1
ปุ่ม Ctrl	AcCtrlMask	2

Keyboard ที่กด	Constant	ค่า
ปุ่ม Alt	AcAltMask	4

ปุ่มที่กด สามารถกดได้พร้อมกันได้ที่ละปุ่ม หรือพร้อมกันก็ได้ จึงทำให้ค่าที่ส่งกลับมา ที่ Button หรือ Shift จึงเป็นดังตาราง สมมุติว่าเรากดปุ่มซ้าย พร้อมกับปุ่มขวา ค่าที่ได้คือ 1+2 = 3 วิธีการตรวสอบการปุ่ม จะใช้ AND ซึ่งเป็น Opertor คนละตัวกับ AND ที่ใช้เชื่อมเงื่อนไข ทดลองดูตามการตรวจสอบตามตัวอย่าง

```
Private Sub Text0_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As
Single)
        Dim tmpstr

        If (Button And acLeftButton ) > 0 Then
           tmpstr = "Left,"

        End If

        If (Button And acRightButton) > 0 Then
           tmpstr = tmpstr & "Right,"

        End If

        If (Button And acMiddleButton) > 0 Then
           tmpstr = tmpstr & "Middle,"

        End If
        MsgBox ("Click on:" & tmpstr)
End Sub
```

กลุ่ม Keyboard

เป็น Event ที่เกิดเมื่อมีการกด Keyboard แบ่งเป็น 3 ลำดับคือ KeyDown -> KeyPress- > KeyUp ในทุกๆ Stroke ที่มีการบันทึกลงบน Control จะเกิด Event ดังกล่าวตามลำดับ คือ กด Keydown, เกิด Keypress ตรงนี้เราได้ตัวอักษร จากนั้น KeyUp เมื่อยกนิ้วออกจากแป้น Keyboard

กรณีที่การกดนั้น ไม่มีตัวอักษร เช่น F1-F12 Event KeyPress จะไม่เกิดขึ้น , Control ที่เราเกี่ยวข้องด้วยมักเป็น TextBox Control, ปุ่ม Command หรือ Control ที่มีการใช้ Keyboard สั่งการได้ และเช่นเดียวกับ Mouse เราสามารถอ่าน Parameter ในการกด Keyboard ได้ใน Event เช่นกัน

- **KeyDown, KeyUp**

ค่าที่ส่งผ่านมากับ Event คู่นี้คือ KeyCode กับ Shift สำหรับ Shift เป็นตัวบอกการกดปุ่ม Ctrl, Alt, Shift ค่าของ มันจะเหมือนกับที่แสดงในตาราง Shift ของ Mouse

ส่วน Keycode เป็นรหัส Keyboard ที่กด ต้องทำความเข้าใจเสียก่อนว่าทุก Key บนแป้น Keyboad มีรหัส Keycode แต่ ไม่ใช่ทุก Key ที่กดแล้วจะเป็นตัวอักษรบนจอ เช่นเวลาเรากด F1 – F12 เป็นต้น เราใช้ Keydown

และ KeyUp เป็นตัวตรวจสอบ Key พิเศษเหล่านี้ วิธีที่จะดูว่า KeyCode ใดเป็นของปุ่มใด ให้ลองสร้าง Code ซ่อนไว้ใน Event Keydown แล้วสั่งให้ ส่งค่าผ่าน Msgbox ให้เราทราบ

```
Private Sub Text2_KeyDown(KeyCode As Integer, Shift As Integer)
        MsgBox (KeyCode)
End Sub
```

- **KeyPress**

สิ่งที่แตกต่างกับ KeyDown, KeyUp คือ Keypress ไม่มีการส่ง KeyCode แต่จะส่ง KeyAscii เฉพาะเมื่อการกดบนแป้นนั้นมี รหัส Ascii หรือตัวอักษรที่ตอบสนองกับการกดนั้น รหัส KeyAscII เป็นรหัสตัวอักษร (ไม่ใช่รัหส KeyBoard) เช่นเดียวกับ KeyDown เราทดสอบรหัส KeyAscII ได้ด้วย Event เช่นกัน

```
Private Sub Text2_KeyPress(KeyAscii As Integer)
        MsgBox (KeyAscii & " : Charater is " & Chr(KeyAscii))

End Sub
```

กลุ่ม Focus

ณ ขณะหนึ่ง Access จะมี Form ที่กำลังทำงานอยู่ และ ในแต่ละ Form นั้นก็จะมี Control ที่ Form Form นั้นอยู่ในตำแหน่ง Focus ซึ่งเทียบได้กับตำแหน่ง Cursor ในโปรแกรม ที่เป็น Dos Mode เมื่อมีการเคลื่อนไหวระหว่าง Control จะเกิด Event ที่บอกให้เราทราบว่า Focus ได้เปลี่ยนตำแหน่งจาก จุดหนึ่งไปอีกจุดหนึ่ง มีรายละเอียดคือ

- **GOTFOCUS EVENT**

เกิดเมื่อมี Control ตัวใดตัวหนึ่งมี Cursor ไปกระพริบที่ตัวมัน การ GotFocus นี้จะเกิดทุกครั้งที่ Control ได้รับ Focus ไม่ว่าเป็นการ Switch ระหว่าง Form ก็ตาม ปกติ การย้ายจาก Form หนึ่งไปอีก Form หนึ่ง จะเกิด Form_GotFocus แต่พอ เมื่อมี Control บน Form, Focus Event นี้ จะย้ายไปที่ Control แทน

- **ENTER EVENT**

ENTER คล้ายกับ GotFocus แต่มีความแตกต่างที่ Event นี้จะเกิดเมื่อมีการย้ายจาก Control หนึ่ง มายัง Control หนึ่งเท่านั้น เมื่อ Focus บน Form Form หนึ่ง ยังอยู่ที่ Control เดิม แล้วมีการย้ายไปมาระหว่าง Form การย้ายกลับมายัง Form เดิม จะไม่เกิด Enter Event แต่จะเกิด GOTFOCUS เราควรจำว่า Enter Event เกิดเมื่อมีการย้ายระหว่าง Control ใน Form เดียวกัน ส่วน GotFocus เกิดขึ้นเสมอ เมื่อมีการ Focus บน Control ในขณะใดใดก็ตาม

- **LOSTFOCUS EVENT**

เช่นเดียวกับ GotFocus , Event ตัวนี้ทำงาน เมื่อ Focus ย้ายออกจาก Control และเกิดขึ้นทุกครั้งที่มีการย้ายออกจาก Control เช่นเราอยู่ที่ Control เดิม แต่ย้ายไป Form อื่น ก็ LostFocus เช่นกัน แม้ว่า Focus สำหรับ Form นั้นยังเป็น Control ตัวเดิม

- **EXIT EVENT**

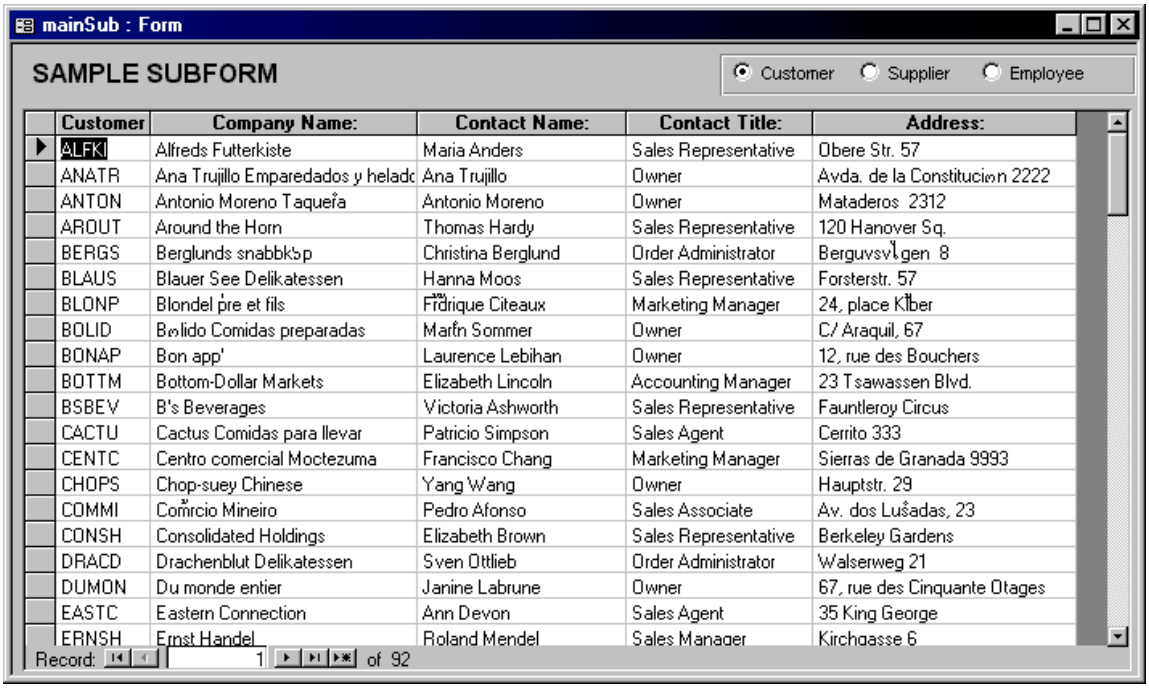
คล้ายกับ Enter Event แต่เกิดเมื่อมีการย้ายจาก Control ตัวหนึ่งไปอีกตัวหนึ่งเท่านั้น ไม่เกิด Event นี้ ถ้ามีการย้าย Form โดยที่ Focus บน Form นั้นๆ ยังเป็นตัวเดิม

SUBFORM

Subform เป็น Control แบบหนึ่ง แต่เป็นการนำ Form อีก Form หนึ่งมาเป็น Form ลูกที่เชื่อมเข้าหากัน เราใช้ประโยชน์จากการทำ Subform ในการแสดงข้อที่สัมพันธ์กับ Form แรก สิ่งที่เราเข้าไปเกี่ยวข้องกับ Subform บ่อยๆ ได้แก่

การเปลี่ยน SourceObject

เราบังคับ ให้ Subform เปลี่ยนตัว Subform ที่ใช้ เราใช้บ่อยในการทำ Form กลาง ที่ใช้ควบคุมการบันทึกข้อมูลพื้นฐาน ตัวอย่างเช่นตาราง Customer, Supplier, Employee เราทำ Form ที่เป็น Form มาตราฐานสำหรับการบันทึกและให้ User ทำการเลือกที่จะบันทึก ฐานข้อมูลตัวใดตัวหนึ่งก็ได้ ทดลองทำตามตัวอย่าง



Customer	Company Name:	Contact Name:	Contact Title:	Address:
ALFKI	Alfreds Futterkiste	Maria Anders	Sales Representative	Obere Str. 57
ANATR	Ana Trujillo Emparedados y helado	Ana Trujillo	Owner	Avda. de la Constitucion 2222
ANTON	Antonio Moreno Taquería	Antonio Moreno	Owner	Mataderos 2312
AROUT	Around the Horn	Thomas Hardy	Sales Representative	120 Hanover Sq.
BERGS	Berglunds snabbköp	Christina Berglund	Order Administrator	Berguvsvägen 8
BLAUS	Blauer See Delikatessen	Hanna Moos	Sales Representative	Forsterstr. 57
BLONP	Blondel père et fils	Frédrique Citeaux	Marketing Manager	24, place Kléber
BOLID	Bólido Comidas preparadas	Martín Sommer	Owner	C/ Araquil, 67
BONAP	Bon app'	Laurence Lebihan	Owner	12, rue des Bouchers
BOTTM	Bottom-Dollar Markets	Elizabeth Lincoln	Accounting Manager	23 Tsawassen Blvd.
BSBEV	B's Beverages	Victoria Ashworth	Sales Representative	Fauntleroy Circus
CACTU	Cactus Comidas para llevar	Patricio Simpson	Sales Agent	Cerrito 333
CENTC	Centro comercial Moctezuma	Francisco Chang	Marketing Manager	Sierras de Granada 9993
CHOPS	Chop-suey Chinese	Yang Wang	Owner	Hauptstr. 29
COMMI	Comércio Mineiro	Pedro Afonso	Sales Associate	Av. dos Lusíadas, 23
CONSH	Consolidated Holdings	Elizabeth Brown	Sales Representative	Berkeley Gardens
DRACD	Drachenblut Delikatessen	Sven Ottlieb	Order Administrator	Walserweg 21
DUMON	Du monde entier	Janine Labrune	Owner	67, rue des Cinquante Otages
EASTC	Eastern Connection	Ann Devon	Sales Agent	35 King George
ERNSH	Ernst Handel	Roland Mendel	Sales Manager	Kirchgasse 6

ตัวอย่างการควบคุม Subform

1. ทำ Form ใหม่ ให้ชื่อว่า Sub1
2. ใช้ RecordSource เป็น Customers
3. กำหนด DefaultView กับ ViewAllowed ให้เป็น Datasheet
4. เลือก Field ที่ต้องการนำมาแสดง
5. Save Form
6. ทำ Form Sub2, Sub3 โดยใช้ RecordSource เป็น Suppliers, Employees แล้วทำตามขั้นตอนเดิม
7. สร้าง Form แม่ ตั้งชื่อเป็นอะไรก็ได้
8. ลาก Sub1 มาเป็น Subform
9. สร้าง OptionGroup โดยใส่ Option 3 ตัวคือ Customers, Suppliers, Employees ตามลำดับ ให้ชื่อ Option Group นี้ว่า SelectX
10. ใส่ Code ตามตัวอย่าง ลง SelectX

```
Private Sub SelectX_BeforeUpdate(Cancel As Integer)
        Me.SubX.SourceObject = "sub" & SelectX
End Sub
```

11. ทดลอง Run Program

ตัวอย่างข้างต้น เป็นการเปลี่ยน SourceObject จาก Form หนึ่งไปอีก Form หนึ่ง โดยเปลี่ยนแค่ชื่อ วิธีนี้ เราจะได้ Form ที่สามารถทำบันทึก ข้อมูลได้ทั้ง 3 ตาราง จาก Form เดียว ทำให้ User ใช้งานง่ายขึ้น โดยใช้คำสั่งเพียงแค่ Me.SubX.SourceObject = “Sub” & SelectX เท่านั้น

เราควร Set ให้ Form แม่ มี RecordSelector , Recordnavigator เป็น No และ ปิด ScrollBar ด้วย เพื่อให้เป็นหน้าจอบันทึกที่คลุม SubForm อีกต่อหนึ่ง ผู้ใช้จะไม่สับสนกับแถบ Selector และ Navigator

การเปลี่ยน RecordSource

เราเปลี่ยน RecordSource ของ Subform ได้เช่นกัน อันนี้จะคล้ายกับการทำ Filter แต่ สามารถกำหนดเป็นภาษา Sql ได้โดยตรง เราจะคุยถึงเรื่องนี้อีกครั้งในเรื่องของ SQL

Subform Method / Event

เราใช้ Method และ Event เดียวกับ Form ตามปกติ แต่เวลาสั่งการ เราจะใช้ ชื่อ Subform นำหน้าแทน

Parent Properties

ในขณะที่อยู่ใน Subform เราสามารถอ้างอิง ตัว Form ที่เป็น Form แม่ โดยใช้ Properties ชื่อ Parent ซึ่งดีกว่าการอ้างชื่อ Form เต็มของ Form แม่ เราใช้ Paraent นำหน้า แล้วตามด้วย ชื่อ Control ตามปกติ เช่น การ Set ค่า Text0 บน Form แม่ ขณะที่ Code อยู่ใน Subform คือ

```
Me.Parent.Text0 = "Hello"
```

REPORT EVENT

เราเข้าไปเกี่ยวข้องกับ Object บน Report น้อยกว่า Form มาก อย่างไรก็ดี เราควรรู้จักกับ Event ของ Report ที่สามารถช่วยให้เราสามารถ ออก Report ที่ซับซ้อนได้มากขึ้น

เรามี Event บนตัว Report เท่านั้น ไม่มี Event บน Control ที่อยู่บน Report เนื่องจาก เราไม่มีการ บันทึกข้อมูลใน Report จึงไม่มี Event ของ Control บน Report เหมือนกับ Form

เราแบ่ง Event ได้เป็น กลุ่มใหญ่ ๆ คือ

- Event ของตัว Report
- Event เมื่อมีการประมวผล หรือพิมพ์รายการในส่วน Detail
- Event เมื่อมีการทำ Page Header/Footer เมื่อมีการกำหนดให้มี Page Header /Footer
- Event เมื่อมีการทำ Report Header/Footer เมื่อมีการกำหนดให้มี Report Header /Footer
- Event เมื่อมีการทำ Group Header / Footer สำหรับกรณี Report ที่มี Group Header / Footer โดยแยกเป็น Event ของ Group แต่ละตัว

Event ของ Report

คล้ายกับชุด Event ของ Form มี Open, Close, Activate, Deactivate, Error เช่นกัน มีกลไกการทำงานแบบเดียวกับ Form ทุกประการ มี Event ที่แตกต่างคือ

- **NODATA EVENT**

Event ตัวนี้เกิดขึ้นเมื่อ Report ตัวนั้น ไม่มีข้อมูลที่จะนำมาแสดง หรือ RecordCourse ที่กำหนดไว้ ให้จำนวน Record เป็น 0 เราสามารถ Cancel การ Open Report ตัวนี้ได้โดยการระบุ Cancel = True , Event ตัวนี้มีประโยชน์มาก เพราะ ช่วยให้การพิมพ์ Report ที่ไม่มีข้อมูล ไม่ให้แสดงข้อความว่า Error บนหน้ากระดาษ โดยการยกเลิก Report เสียก่อน

```
Private Sub Report_NoData(Cancel As Integer)
       MsgBox ("ไม่พบข้อมูลในรายงาน")
       Cancel = True
```

```
End Sub
```

Event ของ Detail

ทุกทุก บรรทัดของข้อมูลที่มีการแสดงบน Detail มี Event กลุ่มนี้เกิดขึ้นเสมอ เราสามารถทำการพิเศษๆ บน Event เหล่านี้ได้ ส่วนใหญ่เราใช้ Event Format ส่วน อีก 2 ตัวเราใช้น้อยดังต่อไปนี้

- **FORMAT EVENT**

เกิดขึ้นเมื่อ Access ดึงข้อมูลมาวางบน Report และเริ่มจัดรูปแบบเพื่อเตรียมพิมพ์ ใน EVENT นี้เราสามารถอ่านค่าของ Control มาทำการคำนวณพิเศษๆ ได้ ในทุกๆ บรรทัดของข้อมูลที่มีการอ่านมาแสดงผล

สมมุติว่าเราไม่ใส่ pages (จำนวนหน้า) ลงใน Control , Access จะไม่อ่านข้อมูลจนหมด เพื่อทดสอบว่า รายงานของเราต้องใช้กี่หน้า แต่ถ้าเรากำหนด ก็จะเกิด Event นี้เท่ากับจำนวน Record ที่มีทันทีเพื่อ Access จะได้ทราบว่าต้องใช้กระดาษกี่หน้า และหากมีการสั่งพิมพ์อีกครั้งหนึ่ง Event เหล่านี้จะเกิดซ้ำ ดังนั้นการทำการบน Event จะต้องคำนึงถึงการกลับมาเรียกซ้ำได้ ของ Event นี้

นอกจากนี้ การตั้งให้ Object CanGrow, CanShrink ก็เป็นส่วนที่ทำให้ มีการทำการ Format ซ้ำได้ เพราะ Access ต้องทดสอบจำนวนหน้าโดยการวาง Control ลงบนหน้ากระดาษ หากล้นหน้า ก็ต้องขึ้นหน้าใหม่ แล้วทดลองวาง Control ใหม่ ตรงนี้จะเกิด Event ซ้าซ้อน ซึ่งต่างจาก Case แรก เพราะ Event ที่ซ้ำจะเกิดติดๆกันในระหว่าง 1 Control เราสามารถตรวจสอบการเรียกซ้ำ จากค่า FormatCount ถ้า FormatCount > 1 แสดงว่าเป็นการเกิด Event Format ซ้ำซ้อนเป็นต้น

ในแต่ละ Detail เราสามารถสั่งให้มีการยกเลิกรายการบน Report ได้โดยใช้คำสั่ง Cancel = True เช่นกัน วิธีนี้ทำให้รายการที่ถูก Cancel หายไปจาก รายงาน ตัวอย่าง Code ชุดนี้เป็นการบังคับให้แสดงเฉพาะรายชื่อ Supplier ที่มีรหัสเป็นเลขคู่ ทดลองสร้าง รายงานที่มี Supplier เป็น Recordsource และบรรจุ Code ชุดนี้เข้าไป

```
Private Sub Detail_Format(Cancel As Integer, FormatCount As Integer)
        If (Me.SupplierID And 1) > 0 Then
           Cancel = True
        End If

End Sub
```

- **RETREAT EVENT**

Event นี้ไม่เหมือนกับ Format เป็น Event ที่อาจเกิดมากกว่า 1 ครั้งได้เหมือนกัน Event นี้คล้ายกับการทำ Format ซ้ำ แต่เกิดขึ้นระหว่าง Section ของ Report เช่นระหว่างที่ทำการ Format Detail อยู่ แล้วต้องขึ้นหน้าใหม่ เพราะหน้ากระดาษไม่พอ แต่มีการกำหนดให้ Group Header ขึ้นทุกหน้า จึงเกิดการ Format ส่วน Group header ตรงนี้จะเกิด Event Retreat บน GroupHeader แล้วเมื่อทำการ Format เสร็จ ก็จะกลับมาที่ Detail อีกครั้ง แล้วเริ่มทำ

การพิมพ์ข้อมูลในส่วน Detail ต่อไป ตรงนี้จะเกิด Event Retreat บน Detail อีกครั้งหนึ่ง ในขณะกลับมาจาก GroupHeader, Event นี้คล้ายกับการ GotFocus ของ Form นั่นเอง ไม่ค่อยได้มีโอกาสได้ใช้เท่าไร

- **PRINT EVENT**

เมื่อทำการพิมพ์ข้อมูลลงบน Report เกิดหลังจากทำการ Format เสร็จสมบูรณ์แล้ว ดังนั้นเราจะตรวจสอบที่ Format ก่อน แล้วหากเราไม่ทำการ Cancel , ก็จะเกิด Event นี้ขึ้น เราอาจทำการตรวจสอบอีกครั้งที่ Event นี้ก็ได้ แต่ในทางปฏิบัติ ก็ใช้ Format Event แทนกันได้ ใน Event นี้ก็มี Format Count ให้ตรวจสอบการเรียก Event นี้ซ้ำเช่นกัน และสามารถ Cancel ได้เหมือน Format ทุกประการ

EVENT ของ Page / Report / Group

เหมือนกับ Event ของ Detail แต่จังหวะในการเกิด Event ไม่เหมือนกัน แตกต่างกันไปตามประเภท โดยแบ่งเป็น Header กับ Footer

EVENT	คำอธิบาย
ReportHeader_Format	เมื่อมีการเริ่มทำการ Format หัว Report เกิดขึ้นหนึ่งครั้งเท่านั้น เรามักใช้ในการ Initial ค่าตัวแปรเริ่มต้น หากต้องมีการทำการคำนวณพิเศษๆ ใต้ Detial
ReportFooter_Format	เมื่อมีการเริ่มทำการ Format ท้าย Report ใช้สำหรับการคำนวณปิดท้าย
PageHeader_Format	เมื่อมีการเริ่มทำการ Format หัวกระดาษ
PageFooter_Format	เมื่อมีการเริ่มทำการ Format ท้ายกระดาษ
GroupnHeader_Format	เมื่อมีการเริ่มทำการ Format Group ที่ n ที่หัวกระดาษ สำหรับ Set ค่าตัวแปรในแต่ละหมวด
GroupnFooter_Format	เมื่อมีการเริ่มทำการ Format Group ที่ n ที่ท้ายกระดาษ ใช้สำหรับการคำนวณปิดท้ายของแต่ละ Group

CHAPTER 4

คำสั่งพิเศษ

จากที่ได้เรียนรู้ VB และ กลไกของ Object ไปแล้ว เนื้อหาดังกล่าวจะเป็นกลางกับ Application ตระกูล Microsoft Office ซึ่งเราไปทำความเข้าเพิ่มเติมกับ VBA ตัวอื่นๆ เช่น Excel, Word ได้

สำหรับในบทนี้เราจะได้ทำความรู้จักกับคำสั่งพิเศษๆ เพิ่มเติม ทั้งที่เฉพาะเจาะจงกับตัว Access เอง ซึ่งใช้ในการเรียก Feature พิเศษๆ ของ Access มาทำงาน และ Feature สำคัญในการอ่าน File จาก Text ไฟล์เป็นต้น

CONTENT

- การใช้ Docmd
- การใช้ SysCmd
- การทำงานกับ File
- การเรียกโปรแกรมอื่นๆ
- การอ่านเขียน File

การใช้ Docmd

Docmd เป็นคำสั่งที่พิเศษที่ใช้การเรียก Feature พิเศษๆ ของ Access ขอวมาใช้งาน ซึ่ง Feature เหล่านี้ ตัว Object ใน VBA ไม่รองรับโดยตรง เช่นคำสั่งในการเปิด Form หรือ Report เป็นต้น

รูปแบบคำสั่ง Docmd จะเสมือนเป็น Opject ตัวหนึ่ง แล้วใช้ Method ที่ต้องการให้ตัว Docmd ทำงาน เช่น ต้องการให้ Open Form ชื่อ prgSupplier จะใช้คำสั่งว่า

```
Docmd.OpenForm "prgSupplier"
```

แต่ละคำสั่งใน Docmd จะมี parameter ที่ต้องส่งผ่านให้ เช่นกร OpenForm ก็ต้องบอกว่า จะให้ OpenForm ชื่ออะไรเป็นต้น ในบทนี้เราตั้งเป้าว่าจะทำ Menu ที่ใช้ในการเรียก Form หรือ Report จาก โดยใช้คำสั่ง Docmd ในการจัดการทั้งหมด

Docmd.OpenForm

เป็นคำสั่งที่ใช้ในการ OpenForm มีรูปแบบคำสั่งคือ

```
DoCmd.OpenForm formname[, view][, filtername][, wherecondition][, datamode][, windowmode][, openargs]
```

ตัว Parameter ที่อยู่ใน [ ] ถือเป็น Option ไม่ใส่ก็ได้ สำหรับ Parameter ที่ส่งผ่านมากับ Openform มีดังนี้.

ค่าตัวแปร	คำอธิบาย
Formname	ระบุชื่อ Form ที่ต้องการเปิด เป็น String
View	ระบุ Mode ที่ใช้ในการเปิด Form ใช้ค่า Constant เหล่านี้ได้เลย **AcDesign** : เปิดในแบบ Design Mode **AcFormDS** : เปิดในแบบ DataSheet **AcNormal** (default) : เปิดในแบบปกติ คือ SingleForm หรือ Continue ตามที่กำหนดไว้ในตัว Form **AcPreview**: เปิดในแบบ Preview เพื่อรอพิมพ์
Filtername	ระบุ Filter ที่ใช้ในการเปิด Form ถ้าไม่ระบุ ถือว่าไม่มี Filter
Wherecondition	ระบุ Where ที่ใช้การเปด ตรงนี้ต้องระบุเป็น String โดยบอกเงื่อนไขที่ต้องการ Where เช่น “SupplierID > 10 “ เป็นต้น `Docmd.OpenForm prgSupplier ,,, "SupplierID > 10 "` ในตัวอย่างนี้ เราไม่ระบุค่า parameter ของ View และ FilterName จึงให้

ค่าตัวแปร	คำอธิบาย
	ถือเป็นค่า Default
Datamode	บอก Mode ของการใช้ฐานข้อมูล ว่าการเปิด Form ดังกล่าวต้องการให้ทำการเปลี่ยนแก้ไขข้อมูลได้หรือไม่ **AcFormAdd** : สั่งให้เปิด Form แบบให้เพิ่มข้อมูลได้เท่านั้น **AcFormEdit**: สั่งให้เปิด Form มาเพื่อการ Edit เท่านั้น เพิ่มลดไม่ได้ **AcFormPropertySettings** (default) : ให้ใช้ค่าที่ Set ไว้ใน Form ตรงนี้เป็นค่า Default **AcFormReadOnly**; สั่งให้ Form ดังกล่าวสามารถดูได้อย่างเดียว แก้ไขไม่ได้
Windowmode	บอกสถานะของ Windows ที่เปิด ว่าต้องการให้เปิดในสถานะใดเช่น **AcDialog**: เปิดเป็น Dialog ไม่สามารถไป Click ที่อื่นได้ จนกว่าจะปิด Form นี้ **AcHidden**: เปิดแบบ ซ่อน กล่าวคือ เปิดแล้ว แต่ Form.Visible = False **AcIcon**: เปิดเป็น Icon หรือการ Minimize ทิ้งไว้อยู่ที่ด้านล่าง **AcWindowNormal** (default) : เปิดเป็นหน้าจอ Windows ปกติ

Openargs	เป็นค่าที่สามารถส่งผ่านไปยัง Form ได้ ขณะที่ Form ทำการ Open จะสามารถ อ่านค่า OpenArgs นี้ออกมาตรวจสอบได่ เช่นการผ่านระดับ User ไปยัง Form ลูก หรือ ระบุรหัส Supplier ต้องการดูในการเปิด Form ดังกล่าว
	`Sub OpenToCallahan()`
	`  DoCmd.OpenForm "Employees", acNormal, , , acReadOnly, , "Callahan"`
	`End Sub`
	ที่ Form ที่ถูก Open สามารถอ่านค่า OpenArgs จาก Properties ของ Form เอง
	`Sub Form_Open(Cancel As Integer)`

ค่าตัวแปร	คำอธิบาย

```
        Dim strEmployeeName As String
        StrEmployeeName = Forms!Employees.OpenArgs
        If Len(strEmployeeName) > 0 Then
            DoCmd.GoToControl "LastName"
           DoCmd.FindRecord strEmployeeName, , True,
, True, , True
        End If
End Sub
```

ทดลองสร้าง ระบบจัดการ Menu เพื่อใช้ในการเรียก Form ขึ้นตามตัวอย่างดังนี้

- **ทำระบบควบคุม Menu**

1. สร้าง Table ที่เก็บรายชื่อ Form ให้ชื่อว่า MainMenu ที่มีในระบบ มี Field ดังนี้

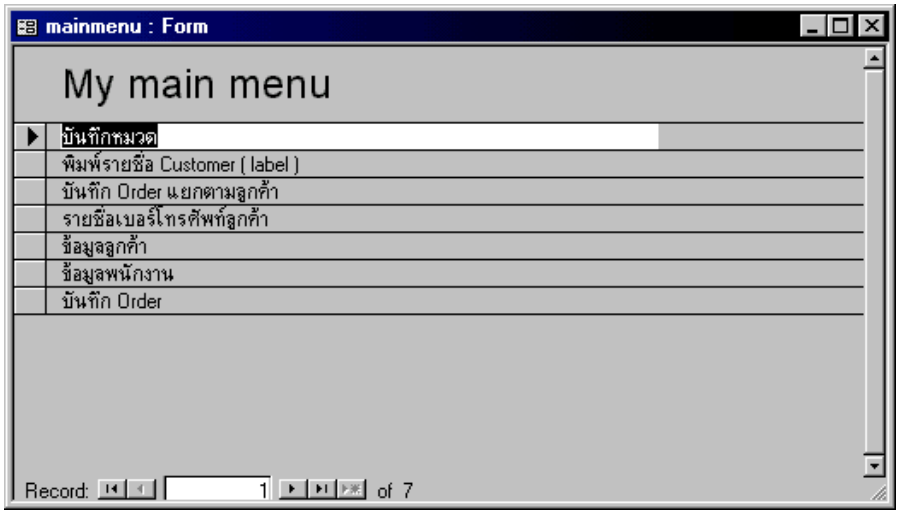
Primary	FieldName	FieldType	Size
Yes	MenuID	AutoNumber	
	MenuSequence	Number	Byte
	MenuName	Text	100
	MenuCall	Text	100

2. บันทึกชื่อ Form ที่มีทั้งหมดลงใน Table ที่ Field MenuCall และตั้งชื่อที่ User ใช้เรียกที่ MenuName ตรงใส่ชื่อไทยก็ได้ จากนั้นกำหนดลำดับที่ที่ MenuSequence ให้ทดลองทำการบันทึกรายชื่อ Form สัก 10 Form

mainmenu : Table

MENUID	MENUSEQUENCE	MENUNAME	MENUCALL
2	0	บันทึกหมวด	Categories
3	1	พิมพ์รายชื่อ Customer (label)	Customer labels Dialog
4	2	บันทึก Order แยกตามลูกค้า	Customer Orders
5	3	รายชื่อเบอร์โทรศัพท์ลูกค้า	Customer Phone List
6	4	ข้อมูลลูกค้า	Customers
7	5	ข้อมูลพนักงาน	Employees
8	6	บันทึก Order	Orders
utoNumber)	0		

3. สร้าง Form เปล่า 1 Form ชื่อ Mainmenu
4. Set Properties ให้ AutoCenter = True, AutoResize = True, DefaultView = Continuous Forms, Allow Edit = Yes, Allow Deletion = No, Allow Addition = No, RecordSource = Mainmenu, Order By = MenuSequence, RecordsetType = SnapShot
5. ลาก Field MenuName มาวางที่ส่วน Detail ให้ได้ Form ดังภาพ



6. แทรก Code ลงใน MenuName_Click ตามตัวอย่าง

```
Private Sub MENUNAME_Click()
        DoCmd.OpenForm Me.MENUCALL
End Sub
```

7. ทดลอง Run Program และทดลอง Click ที่ ชื่อ Menu โปรแกรมจะทำการเรียก Form ที่เรากำหนดไว้ขึ้นมาทำงาน

จากตัวอย่างข้างต้น จะเห็นว่าเราใช้ คำสั่ง Docmd ในการเปิด Form ที่เราต้องการ เราฝั่งชื่อ Form ทิ้งไว้ใน Table แล้วตัว Form MainMenu จะทำการเรียก Form ให้ เมื่อผู้ใช้ทำการ Click Form ดังกล่าว คำสั่งที่ใช้ในการ Open Form เป็นคำสั่ง OpenForm แบบง่าย โดยไม่ระบุ ค่า Parameter โดยส่วนใหญ่เรา ก็จะไม่ระบุ parameter

Docmd.OpenReport

เป็นคำสั่งที่ใช้ในการเปิด Report ที่ต้องการออกมาใช้งาน มีรูปแบบคำสั่งคือ

```
DoCmd.OpenReport reportname[, view][, filtername][, wherecondition]
```

ค่าตัวแปร	คำอธิบาย
Reportname	ระบุชื่อรายงานที่ต้องการเปิด

ค่าตัวแปร	คำอธิบาย
View	ระบุ Mode ที่ทำการเปิดรายงาน เช่น Preview / Print **AcViewDesign** : เปิดในแบบ Design Mode **AcViewNormal** (default): เปิดในแบบปกติ คือพิมพ์ออกทางเครื่องพิมพ์ **AcViewPreview**: เปิดรายงานในแบบ Preview
Filtername	ชื่อ Fileter ที่ต้องการระบุ
Wherecondition	ระบุเงื่อนไข Where เพิ่มเติมสำหรับการเปิดรายงาน เช่น “SupplierID > 10 “ เป็นต้น `Docmd.OpenReport repSupplier ,,acViewPreview, “SupplierID > 10 “` ในตัวอย่างนี้ เราไม่ระบุค่า parameter ของ View และ FilterName จึงให้ถือเป็นค่า Default

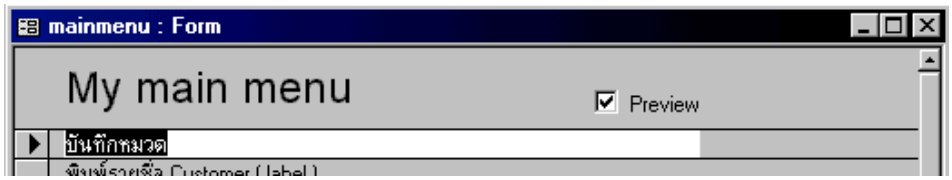
เราใช้คำสั่ง OpenReport ผนวกเข้าไปในระบบ Menu ที่เราทำ ได้เช่นกัน แต่เราต้องระบุเงื่อนไขเพิ่มเติมในการเปิดรายงาน โดยการเพิ่ม Field เข้าไปใน Table Mainmenu อีกครั้งหนึ่ง ให้ชื่อ Field ว่า MenuType เป็น Text 1 ตัวอักษร ทดลองทำตามตัวอย่าง

ปรับระบบควบคุม Menu

1. แก้ไข Table Mainmenu โดยเพิ่ม Field MenuType

Primary	FieldName	FieldType	Size
Yes	MenuID	AutoNumber	
	MenuSequence	Number	Byte
	MenuName	Text	100
	MenuCall	Text	100
	MenuType	**Text**	**1**

2. ใส่ค่า F ลงใน Mainmenu.MenuType สำหรับรายการเก่า เพื่อให้ทราบว่าเป็นรายการประเภท Form
3. ใส่ชื่อ Report สัก 2 3 ตัว โดยให้มี MenuType เป็น R
4. ทำการแก้ไข Form Mainmenu โดยเพิ่ม Field ชื่อ chkPreview เป็น CheckBox ลงบน Form Header



5. แก้ไข โปรแกรมที่การ Click บน Menuname เพิ่มเติมตามตัวอย่าง

```
Private Sub MENUNAME_Click()
        Select Case Me.MENUTYPE
        Case "F"
            DoCmd.OpenForm Me.MENUCALL
        Case "R"
            If Me.ChkPreview Then
               DoCmd.OpenReport Me.MENUCALL, acViewPreview
            Else
               DoCmd.OpenReport Me.MENUCALL, acViewNormal
            End If
        End Select
End Sub
```

Docmd.OpenQuery

คล้ายกับคำสั่งที่แล้วมา เป็นคำสั่งที่ใช้ในการเปิด Query ถ้า Query ประเภท Select Query ก็จะแสดงผล Query ตามปกติ หาก Query นั้นเป็น Update Query จะทำการ Update ข้อมูล ตามที่กำหนดไว้ใน Query

DoCmd.OpenQuery queryname[, view][, datamode]

ค่าตัวแปร	คำอธิบาย	
Queryname	ชื่อ Query	
view	ระบุแบบในการเปิด Query	
	AcViewDesign	เปิด Query ในแบบ Design Mode
	AcViewNormal	(default) เปิด Query ในแบบปกติ คือการ Run Query นั่นเอง
	AcViewPreview	เปิดQuery ในแบบ Preview พร้อมพิมพ์

ค่าตัวแปร	คำอธิบาย	
datamode	ระบุเงื่อนไขในการอนุญาติให้ปรับปรุงข้อมูล กรณีที่เป็น Select Query ที่ทำการ Update ได้	
	AcAdd	สามารถเพิ่มข้อมูล
	AcEdit	(default) สามารถแก้ไขได้
	AcReadOnly	ให้อ่านได้อย่างเดียว

การสั่งให้ OpenQuery ทำงานคล้ายกับ Openform ทุกประการ กรณีที่เป็น Action Query จะมีข้อความเตือนก่อนการ ปรับปรุงเสมอ เราสามารถปิดคำเตือนของ Access ด้วยคำสั่ง SetWarnings

Docmd.SetWarnings

ใช้ในการปิดข้อความเตือน ขณะทำการปรับปรุงข้อมูลโดยใช้ ActionQuery มี parameter เป็น True กับ False

```
Docmd.Setwarnnings True
Docmd.Setwarnnings False
```

ตัวอย่างเช่นการเรียก Query ที่ทำการ Update ข้อมูลชื่อ ActionUpdateMenu

```
Docmd.setWarnings False
Docmd.OpenQuery "ActionUpdateMenu"
Docmd.setWarnings True
```

Docmd.Close

ใช้ในการปิด Windows ใด Windows หนึ่งที่ทำการเปิดอยู่ หากไม่ระบุอะไรค่าตามหลังเลย จะหมายถึงการปิด Form หรือ Report ปัจจุบัน

```
DoCmd.Close [objecttype, objectname], [save]
```

ค่าตัวแปร	คำอธิบาย
objecttype	ระบุประเภทของ Object ที่ต้องการปิด **AcDefault** (default) : คือไม่ระบุ **AcForm** ระบุการปิด Windows ของ Form **AcMacro** ระบุการปิด Windows ของ Macro **AcModule** ระบุการปิด Windows ของ Module **AcQuery** ระบุการปิด Windows ของ Query **AcReport** ระบุการปิด Windows ของ Report **AcTable** ระบุการปิด Windows ของ Table
save	หากมีการแก้ไขบน Object ตัวนั้น และมีการปิด Object Access จะทำการถามถึงการ Save เราระบุวิธีการถามเพื่อ Save ได้ดังนี้ **AcSaveNo** : ไม่มีการ Save ใดใด แม้ไม่มีการแก้ไขก็ตาม **AcSavePrompt** (default) : ให้มี ข้อความถาม User ก่อนปิด หากมีการแก้ไข **AcSaveYes** : สั่ง Save ทันที

ปกติ เราจะใพียงแค Docmd.Close เพียงอย่างเดียว เพื่อทำการปิด Form ที่เปิดทิ้งไว้

Docmd.Quit

เป็นคำสั่งที่ใช้ในการปิดโปรแกรม Access มี Option ที่ตามมาคือ ค่า Parameter ว่าต้องการให้ Access ทำการ Save งานที่ค้างอยู่หรือไม่

```
Docmd.Quit [Option]
```

ค่าตัวแปร	คำอธิบาย
options	ระบุวิธี **AcQuitPrompt** : ออกโดยถามการ Confirm จากผู้ใช้ว่าก่อนออกจาก Access อีกครั้งหนึ่ง **AcQuitSaveAll** (default) : Save งานที่ค้างอยู่ แล้วออกจาก Access **AcQuitSaveNone**: ออกจาก Access โดยไม่ทำการ Save

Docmd.GotoRecord

ใช้สำหรับเลื่อน Record ของ Form, Query หรือ table มีรูปแบบคือ

```
DoCmd.GoToRecord [objecttype, objectname][, record][, offset]
```

ค่าตัวแปร	คำอธิบาย
objecttype	ระบุประเภท Object ที่กำลังจะทำการเลื่อน Record **AcActiveDataObject** (default) : ใช้ Form, Query หรือ Table ตัวที่ Active อยู่ปัจจุบัน **AcDataForm**: ระบุ Form **AcDataQuery**: ระบุ Query **AcDataTable**: ระบุ table
objectname	ระบุชื่อ Object ที่กำลังสั่งให้เลื่อน Record

ค่าตัวแปร	คำอธิบาย	
record	ระบุวิธีการเลื่อน Record มีค่าคือ	
	AcFirst:	ไปยัง Record แรก
	AcGoTo :	ไปยัง Record ลำดับที่ ที่กำหนด
	AcLast :	ไปยัง Record สุดท้าย
	AcNewRec:	ไปยังท้ายสุดของ table เพื่อทำการเพิ่มข้อมูล
	AcNext	(default): ไปยัง Record ต่อไป
	AcPrevious:	ไปยัง Record ก่อนหน้า
Offset	ระบุระยะ Record ที่ต้องการเลื่อนไป ใช้กับ AcGoto, AcNext, AcPrevious	

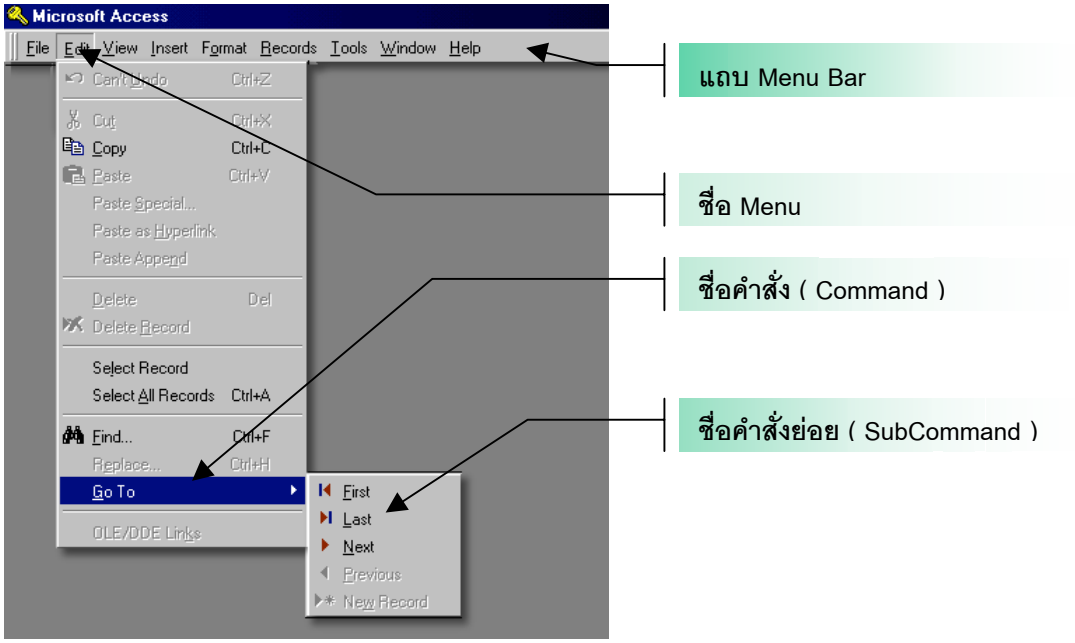
Docmd.Maximize, Docmd.minimize

ใช้ในการ Maximize หรือทำให้เต็มจอ Minimize หรือทำให้ย่อ Windows สั่งโดยตรง โดยไม่ต้องระบุอะไรตามหลัง คำสั่งนี้จะกระทบกับ Current Form เท่านั้น

DoMenuItem

DomenuItem เป็น Docmd ที่ใช้ในการเรียกคำสั่งที่ซ่อนอยู่ใน Menu ออกมาทำงาน มีรูปแบบคำสั่งคือ

```
DoCmd.DoMenuItem menubar, menuname, command[, subcommand][, version]
```



ค่าตัวแปร	คำอธิบาย
Menubar	ระบุชื่อแถบ menuBar ที่ต้องการเรียกคำสั่ง มีค่า acFormBar ที่สามารถใช้เป็น Constant ได้ แต่สำหรับแถบ Menu อื่น จะต้องใช้เลขที่ตามลำดับ Menu Bar ต้องทดลองใส่เลข โดยเริ่มจาก 0 เป็นต้นไป ตรงนี้จะเป็นเลขที่แถบ MenuBar ที่มีในระบบ ปกติจะใช้เพียง acFormBar
menuname	ชื่อของ Menu ที่ต้องการสั่ง ระบุเป็น Constant ได้ดังนี้ **acFile** เลือก File Menu **acEditMenu** เลือก Edit Menu **acRecordsMenu** เลือก Record Menu สำหรับ Menu อื่น เราระบุเป็นตัวเลขลำดับที่ Menu ที่มีใน แถบ Menu Bar นั้นๆ

ค่าตัวแปร	คำอธิบาย	
command	ระบุคำสั่งที่ปรากฏใน Menu นั้น เราระบุเป็นค่าคงที่ได้ดังนี้	
	AcNew :	เลือกคำสั่ง New
	AcSaveForm :	เลือกคำสั่ง SaveForm
	AcSaveFormAs :	เลือกคำสั่ง SaveFormAs
	AcSaveRecord:	เลือกคำสั่ง Save record
	AcUndo :	เลือกคำสั่ง Undo
	AcCut :	เลือกคำสั่ง Cut
	AcCopy :	เลือกคำสั่ง Copy
	AcPaste :	เลือกคำสั่ง Paste
	AcDelete :	เลือกคำสั่ง Delete
	AcSelectRecord :	เลือกคำสั่ง SelectRecord
	AcSelectAllRecords :	เลือกคำสั่ง เลือกทุก Record
	AcObject :	เลือกคำสั่ง Object
	AcRefresh :	เลือกคำสั่ง Refresh
	สำหรับคำสั่งที่นอกเหนือไปจากนี้ เราจะใช้ เลขที่แทน ซึ่งเลขที่จะระบุเป็นลำดับที่รายการคำสั่งใน Menu นั้น	
subcommand	ระบุลำดับที่คำสั่งย่อยที่ต้องการ บอกเป็นเลขที่โดยนับจาก 0 เป็นต้นไป	
Version	ระบุ Version ของ Access เนื่องจาก Menu ในแต่ละ Version อาจไม่เหมือนกัน หากเราไม่ระบุ จะใช้เป็น Version 1.0 ดังนั้น หากต้องการระบุ Menu ที่พิเศษเฉพาะ Version 97 ต้องระบุค่าคงที่ตามหลังด้วย	
	acMenuVer70 : สำหรับ Access 95	
	acMenuVer20 : สำหรับ Access 2.0	
	acMenuVer1X : สำหรับ Access 1.X	

หากดูตามภาพที่แสดงองค์ประกอบของ Menu ถ้าเราต้องการคำสั่ง **GotoLast** เราจะใช้คำสั่งว่า

```
DoCmd.DoMenuItem 0, 1, 12, 1, acMenuVer70
```

สร้าง FORM LOGIN

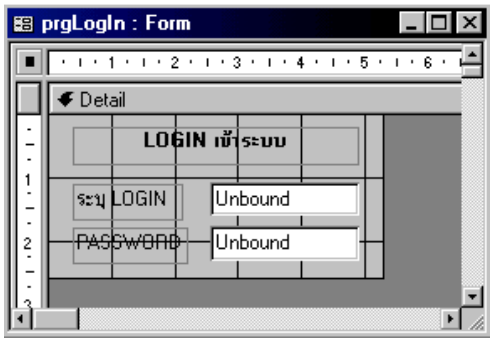
เราจะใช้เนื้อหาที่กล่าวมาถึง ณ ขณะนี้ ทำการสร้าง FORM LOGIN และ เมื่อ LOGIN ผ่านแล้ว จึงไปเรียก FORM MAINMENU ของเราอีกต่อหนึ่ง สามารถใช้ตัวอย่างนี้ในการสร้าง Form Login ที่ซับซ้อนได้ในอนาคต

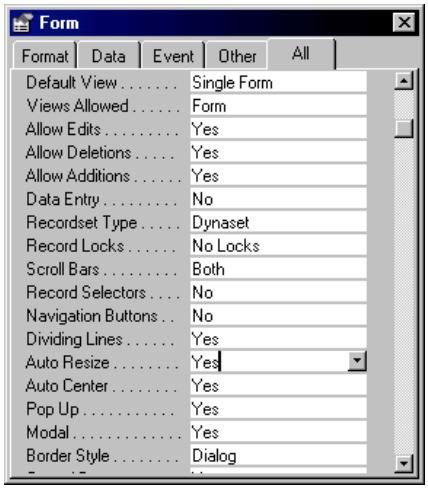
ขั้นตอนการสร้าง

1. สร้าง TABLE ชื่อ LOGINNAME มี 2 FIELD ชื่อ LoginName กับ Password ขนาด 16 ตัวอักษร แล้วทดลองใส่ชื่อ กับ Password ของ User เข้าไป สัก 2-3 ชื่อ

Primary	FieldName	FieldType	Size
Yes	LoginName	Text	16
	Password	Text	16

2. สร้าง FORM ใหม่ 1 FORM ให้ชื่อว่า prgLogin





3. สร้าง Text Control 2 ตัว ชื่อ fldLoginName และ fldPassword ตัว Password ใช้ InputMark เป็น Password
4. กำหนด Properties ของ Form ให้มี AUTO RESIZE = TRUE, AUTO CENTER = TRUE, RecordSelector = FALSE, NAVIGATORBOTTON
5. บรรจุ CODE ชุดนี้ลงใน BeforeUpdate ของ fldPassword แล้วทดลอง Run Form ดู

```
Private Sub fldPassword_BeforeUpdate(Cancel As Integer)
        Dim sysPassWord

        If Not IsNull(Me.fldLoginName) Then
           sysPassWord = DLookup("PassWord", "LoginName", "LoginName ='" &
Me.fldLoginName & "'")
           If IsNull(sysPassWord) Then
```

```
            MsgBox ("ไม่พบรายชื่อในทะเบียน, หรือยังไม่กำหนด PASSWORD")
         Else
            If sysPassWord <> fldPassword Then
               MsgBox ("Password ผิด")
            Else
               DoCmd.Close
               DoCmd.OpenForm "MainMenu"

            End If
         End If


      End If
End Sub
```

จากโปรแกรมที่แสดงไว้ insull ตัวแรก กันกรณีที่ User ไม่ใส่ชื่อ จากนั้น Dlookup Password ของ User มาไว้ที่ sysPassword ถ้าไม่เป็น Null แสดงว่าการ Where ได้ผล มี User คนนี้จริง แต่ถ้าไม่ใช่ เป็นไปได้ว่า ค่าใน Field Password เป็น Null จริงๆ

จากนั้นทำการตรวจ Password กับ Password ที่ User กรอกเข้ามา หากถูกต้องก็ ปิดตัวเองด้วย DoCmd.Close แล้ว เรียก Form MainMenu ออกมา

ถ้าจะให้ดีควร SET STARTUP FORM ของ ไฟล์ MDB เป็น Form prgLogin นี้เสียเลย เพื่อตรวจสอบ Login ของ User ก่อนเข้าระบบ ดูวิธี Set ที่บทที่ 1 ได้

การใช้ SysCmd

เราใช้ SysCmd ในการบังคับให้ Access ทำการพิเศษที่เกี่ยวกับตัวระบบ เช่นการแสดง StatusBar Meter เป็นแถบวิ่งที่เราเห็น ในขณะการทำ Update Query เป็นต้น ลักษณะคำสั่งเป็น Fucntion โดยจะ Return กลับมายังผู้เรียก หากคำสั่งที่ให้ทำไม่มี Error แต่ถ้ามี Error จะเกิด Runtime-Error เอง

เราสามารถแบ่งคำสั่ง SysCmd ได้เป็น 3 ประเภทคือ

- SysCmd แบบให้มีการ Set ค่าบน StatusBar

- SysCmd แบบที่ใช้ในการอ่านของระบบ
- SysCmd แบบที่ใช้ในการอ่านสถานะของ Object ในระบบ

Syscmd สำหรับ Statusbar

ใช้ในการกำหนด และ ยกเลิก StatusBar ทั้งในรูปแบบ Meter และ แบข้อความทั่วไป มีรูปแบบคำสั่งคือ

```
VarX = SysCmd (Action[,Arg1][,Arg2])
```

ให้ประกาศตัวแปร VarX เป็น Variant หรือไม่กำหนดตัวแปร เพื่อมารอรับค่าจาก Syscmd ปกติ จะ Return ค่า Null ถ้าไม่มี Error จะเกิด Runtime-Error

รหัสคำสั่ง	คำอธิบาย

AcSysCmdInitMeter	ทำการ Set ค่าเริ่มต้นของ Meter Bar โดยจะต้องระบุ ข้อความลงใน Text และ ค่าสูงสุดลงใน หรือค่าเต็มที่ 100 % ลงใน Parameterทั้ง 2 ตัว เช่น
	`Dim X, dtaCount`
	`DtaCount = Dcount("SupplierID","Suppliers")`
	`X = SysCmd(acSysCmdInitMeter,"Start Process",dtaCount)`

AcSysCmdUpdateMeter	ทำการ Update ค่า Meter ให้มีค่าตามที่กำหนด เราจะต้องระบุค่า Paramter แรกเป็นค่าที่ได้ เช่น `X = SysCmd(acSysCmdInitMeter,I)`
AcSysCmdRemoveMeter	ยกเลิก Meter Bar ออกจากแถบ StatusBar เราต้องเอา Meter Bar ออกจากระบบเองทุกครั้งเมื่อเราใช้เสร็จ ไม่ระบุค่า Paramter
AcSysCmdSetStatus	ใช้ในการกำหนดข้อความลงใน StatusBar ต้องระบุค่า Paramter 1 ตัว เป็นข้อความที่ต้องการแสดง
AcSysCmdClearStatus	ยกเลิก Statusbar ที่สั่งแสดงไว้ก่อนหน้า ไม่มีค่า Paramter

SysCmd สำหรับอ่านค่าของระบบ

ใช้ในการอ่านค่าที่เกี่ยวข้องกับระบบ ของ Acces เอง วิธีการเรียกใช้ จะเหมือนกับการเรียก Function ทั่วไป โดยระบุค่า Paramter ตามตาราง

```
VarX = SysCmd (Action)
```

รหัสคำสั่ง	คำอธิบาย
AcSysCmdRuntime	รหัสนี้ใช้ตรวจสอบว่า ณ ขณะนี้ Access ทำงานอยู่บน Access แบบ

รหัสคำสั่ง	คำอธิบาย
	Runtime หรือ แบบ Access Version ปกติ Version Runtime จะไม่สามารถทำการออกแบบ หรือแก้ไขได้ acSysCmdRuntime จะให้ค่าจริง ถ้าเป็น Version Runtime
AcSysCmdAccessVer	ใช้รหัสนี้ในการอ่าน Version ของ Access
AcSysCmdIniFile	อ่านค่า INI ไฟล์ที่ใช้กับ Access ปัจจุบัน
AcSysCmdAccessDir	Returns the name of the directory where Msaccess.exe is located.
AcSysCmdProfile	Returns the /profile setting specified by the user when starting Microsoft Access from the command line.
AcSysCmdGetWorkgroupFile	Returns the path to the workgroup file (System.mdw).

ทดลอง Run Code ชุดนี้ จะเห็นค่าที่เก็บไว้จากฟังก์ชั่นชุดนี้

```
Sub testThis()

    Dim tmpx
    tmpx = SysCmd(acSysCmdIniFile)
    MsgBox ("Access Ini file:" & tmpx)

    tmpx = SysCmd(acSysCmdRuntime)
    If tmpx Then
        MsgBox ("This is Runtime version")
        Else
        MsgBox ("This is not Runtime version")
    End If

    tmpx = SysCmd(acSysCmdAccessVer)
    MsgBox ("Access Version:" & tmpx)

    tmpx = SysCmd(acSysCmdProfile)
    MsgBox ("Access Profile:" & tmpx)

    tmpx = SysCmd(acSysCmdGetWorkgroupFile)
    MsgBox ("Access WorkGroupFile:" & tmpx)

End Sub
```

Sycmd สำหรับอ่านสถานะของ Object

เราใช้ในการตรวจสอบว่ามี Object ตัวใดเปิดอยู่บ้าง เช่นอยากทราบว่า Form ชื่อ MainMenu เปิดอยู่หรือไม่ มีรูปแบบคำสั่งคือ

```
VarX = SysCmd (acSysCmdGetObjectState,ObjectType,ObjectName)
```

เราจะต้องระบุประเภท Object ที่ต้องการอ่านที่ ObjectType ส่วน acSysCmdGetObjectState เป็นค่า Constant สำหรับ ObjectName เป็น String ที่เป็นชื่อ Object ที่ต้องการตรวจสอบค่าที่คืนให้กับ Function นี้จะเป็นตามตารางถัดไป หากเป็นค่าศูนย์ แสดงว่า Object ไม่ได้ถูก Open

ค่าที่ระบุใน ObjectType	**คำอธิบาย**
AcTable	ระบุการอ่านสถานะของ Table
AcQuery	ระบุการอ่านสถานะของ Query
AcForm	ระบุการอ่านสถานะของ Form
AcReport	ระบุการอ่านสถานะของ Report
AcMacro	ระบุการอ่านสถานะของ Macro
AcModule	ระบุการอ่านสถานะของ Module

ค่าคงที่ที่ Return ให้	**สถานะของ Object ที่ถาม**
0	กำลัง Close ปิด ไม่ถูก Open
AcObjStateOpen	กำลัง Open อยู่
AcObjStateNew	อยู่ใน Mode New ยังไม่ทำการ Save
AcObjStateDirty	ถูกแก้ไขแล้ว แต่ยังไม่ save

```
Sub testObj()
    Dim x, tmpstr, fName
    fName = "MainMenu"
    x = SysCmd(acSysCmdGetObjectState, acForm, fName)
    Select Case x
    Case 0
        tmpstr = "Not Open"
    Case acObjStateOpen
        tmpstr = "Open"
    Case acObjStateNew
        tmpstr = "New"
    Case acObjStateDirty
        tmpstr = "Open /Dirty"
    End Select
    MsgBox (fName & " is " & tmpstr)
End Sub
```

สามารถใช้ Code ชุดนี้ตรวจสอบคำสั่งชุดนี้ได้ แต่พบว่า acObjStateDirty ไม่ได้ค่าที่ถูกต้อง ควรเป็น 3 ขณะที่คุณอ่านหนังสือเล่มนี้ และพบว่า acObjStateDirty = 4 แสดงว่า Access Version ที่ใช้ยังมี Bug ต้องตรวจสอบค่าด้วย 3 แทน บรรทัด Case acObjStateDirty ต้องใช้ Case 3 แทน

การทำงานกับ File

ชุดคำสั่งต่อไปนี้ เป็นสั่งที่เป็นมาตราฐานของ VB เป็นคำสั่งที่ใช้ในการจัดการ ไฟล์ งานที่เกี่ยวข้องกับ Database มีโอกาศที่จะต้องไปทำการกับ Text ไฟล์บ่อยครั้ง เราจะมองข้ามคำสั่งชุดนี้ไปไม่ได้เลยที่เดียว

คำสั่งเกียวกับ Directory

- **ChDir**

เป็นคำสั่งสำหรับเปลี่ยน Directory ของ Drive แต่ละ ถ้าไม่ระบุชื่อ Drive จะถือว่าเป็น Drive ปัจจุบัน มีรูปแบบคำสั่งคือ

ChDir *Path*

Path เป็นชื่อ Directory ที่ต้องการเปลี่ยนไป ระบุเป็นตัวแปร String เช่น

```
Chdir "C:\Windows"
```

- **ChDrive**

เป็นคำสั่งสำหรับเปลี่ยน Drive ของเครื่อง มีรูปแบบคำสั่งคือ

ChDrive *Drive*

Drive เป็นชื่อ Drive ที่ต้องการเปลี่ยนไป ระบุเป็นตัวแปร String เช่น

```
ChDrive "C:"
```

- **CurDir**

เป็นคำสั่งสำหรับอ่านค่า Directory ปัจจุบัน Return ค่าเป็น String มีรูปแบบคำสั่งคือ

= **CurDir**

ตัวอย่างเช่น

```
Dim strDir

ChDrive "C:\WINDOWS"
StrDir = CurDir
Msgbox ( StrDir )
```

- **MkDir**

เป็นคำสั่งสำหรับสร้าง Directory บน Drive ที่ต้องการ มีรูปแบบคำสั่งคือ

MkDir *Path*

Path เป็นชื่อ Directory ที่ต้องการสร้าง ถ้าไม่ระบุ Drive จะถือว่าเป็น Drive ปัจจุบันDrive ระบุเป็นตัวแปร String ตัวอย่างต่อไปนี้เป็นการสร้าง TmpDirectory สำหรับเก็บ Backup ข้อมูล 7 Directory

```
Sub GenDir()
```

```
    Dim i As Integer
    Dim tmpDir
    tmpDir = "C:\dbBackup"
    MkDir tmpDir
    For i = 1 To 7
        MkDir tmpDir & "\" & i
    Next i
End Sub
```

ถ้ามี Directory แล้วทำการ สร้างเพิ่ม จะเกิด Error รหัส 75 เราสามารถป้องกันโดยใช้ OneError Resume Next และตรวจรหัส Err หลังจาก MkDir เช่น

```
Sub GenTmpCal()

    On Error Resume Next
    MkDir "C:\dbTmp"
    If Err Then
        If Err = 75 Then
            MsgBox ("Directory C:\dbTmp already exist.")
        Else
            MsgBox (Error)

        End If
    End If

End Sub
```

- **RmDir**

เป็นคำสั่งสำหรับลบ Directory บน Drive ที่ต้องการ มีรูปแบบคำสั่งคือ

RmDir *Path*

Path เป็นชื่อ Directory ที่ต้องการสร้าง ถ้าไม่ระบุ Drive จะถือว่าเป็น Drive ปัจจุบันDrive ระบุเป็นตัวแปร String ตัวอย่างต่อไปนี้เป็นการลบ dbBackUp Directory ที่เราสร้างไว้ในตัวอย่าง MkDir

```
Sub ClearDir()

    Dim i As Integer
    Dim tmpDir
    tmpDir = "C:\dbBackup"

    For i = 1 To 7
        RmDir tmpDir & "\" & i
    Next i
    RmDir tmpDir

End Sub
```

คำสั่งเกี่ยวกับ File

- **Dir**

เป็นคำสั่งสำหรับค้นหา File หรือ List ไฟล์ เหมือนกับ Dir บน Dos มีรูปแบบคำสั่งคือ

```
=Dir[(pathname[, attributes])]
```

เราสั่งครั้งแรก จะต้องระบุ File ที่ต้องการค้นหา ระบุเป็น String ที่ pathname แบบเดียวกับที่ใช้บน Dos เช่น “C:*,MDB” เป็นต้น ส่วน attributes เป็นค่า Paramter ที่ใช้ระบุประเภทของ File ที่ต้องการค้นหา ตามตาราง ค่าที่ Return มาจะเป็นชื่อ File

เมื่อทำสั่งครั้งแรกแล้ว คำสั่ง Dir ครั้งต่อๆไป ไม่ต้องมีการใส่ค่า pathname อีก Dir จะใช้ Pathname เดิม แล้วค้นหารายการต่อไป

ค่าคงที่ของ Attribute	ค่าจริง	คำอธิบาย
vbNormal	0	ไม่ระบุการค้นหาพิเศษ จะ Scan เฉพาะไฟล์ปกติ
VbHidden	2	ระบุให้รวม ไฟล์ที่เป็นสถานะ Hidden ด้วย
VbSystem	4	ระบุให้รวม System ไฟล์ เพื่อการค้นหาด้วย
VbVolume	8	สำหรับต้องการอ่านชื่อ Volume label ของ Disk
VbDirectory	16	สำหรับค้นหาเฉพาะ Directory

ค่า Paramter ที่กำหนด สามารถกำหนดพร้อมกัน หลายๆค่าได้ เนื่องจากเป็นเลขฐาน 2 สมมุติว่าต้องการค้นหาไฟล์ ทั้งที่เป็น Hidden และะ System File จะใช vbNormal + vbHidden + vbSystem

ตัวอย่างการอ่านชื่อไฟล์ใน C:\

```
Sub GetCFile()
    Dim tmpStr, i

    i = 0
    tmpStr = Dir("C:\*.*")
    Do Until tmpStr = ""
       i = i + 1
       Debug.Print tmpStr
       tmpStr = Dir
    Loop

    MsgBox ("Total: " & i & " files")

End Sub
```

ตัวอย่างข้างต้น หากต้องการให้มีการ Scan ทั้ง Hidden และ System ไฟล์ จะใช้

```
tmpStr = Dir("C:\*.*",vbNormal + vbHidden + vbSystem)
```

ตัวอย่างการอ่านชื่อ Volume จะต้องระบุชื่อ Dir ตามตัวอย่าง

```
MsgBox (Dir("C:", vbVolume))
```

- **GetAttr**

เป็นคำสั่งสำหรับตรวจสอบคุณสมบัติของ File ว่าเป็น File, Dirrectory หรือถ้าเป็นไฟล์แล้ว เป็น System File, Hidden File เป็นต้น

```
=GetAttr(PathName)
```

ระบุชื่อ File ลงที่ pathName ค่าที่ส่งคืนมาเป็นเลขฐาน 2 เพราะไฟล์ 1 ไฟล์อาจเป็นSystemFile และ Read Only ก็ได้ วิธีตรวจสอบจะต้องใช้การ And เรานำโปรแกรมเดิมมาเพิ่มเติมส่วนของการตรวจสอบประเภทไฟล์ อย่าลืมเปิด Debug Windows เพื่อดูผล

```
Sub GetCFile()
    Dim tmpStr, i, strType, tmpType

    i = 0
    tmpStr = Dir("C:\*.*", vbNormal + vbHidden + vbSystem + vbDirectory)
    Do Until tmpStr = ""
       i = i + 1
       tmpType = GetAttr("C:\" & tmpStr)
       strType = ""
       If (tmpType And vbNormal) > 0 Then strType = strType & "Normal,"
       If (tmpType And vbReadOnly) > 0 Then strType = strType & "ReadOnly,"
       If (tmpType And vbHidden) > 0 Then strType = strType & "Hidden,"
       If (tmpType And vbSystem) > 0 Then strType = strType & "System,"
       If (tmpType And vbDirectory) > 0 Then strType = strType & "Directory,"
       If (tmpType And vbArchive) > 0 Then strType = strType & "vbArchive,"

       Debug.Print tmpStr, strType
       tmpStr = Dir
    Loop

 End Sub
```

ค่าคงที่ที่ Return	ค่าจริง	คำอธิบาย
VbNormal	0	เป็นไฟล์ Normal
VbReadOnly	1	เป็นไฟล์ประเภท Read-only
VbHidden	2	เป็นไฟล์ประเภท Hidden
VbSystem	4	เป็นไฟล์ประเภท System
VbDirectory	16	เป็น Directory หรือ folder
VbArchive	32	เป็นไฟล์ประเภท Archive คือถูกแก้ไขแล้ว จากการ Backup ล่าสุด

- **SetAttr**

คำสั่งนี้ จะตรงกันข้ามกับ GetAttr เป็นการ Set ค่า File Attribute ตามที่ต้องการ เป็น Sub วิธีเรียก ต้องรียกโดย Call หรือใช้ชื่อ Sub โดยไม่ต้องมีค่ามารับ มี 2 Paramter ระบุชื่อ File ที่ต้องการ Set ที่ pathName และ Attribute ที่ต้องการลงใน attributes ซึ่งมีค่าตามตาราง ใช้การบวก เพื่อให้ File มี Attribute ที่เป็นไปได้หลายกรณี

SetAttr *pathname, attributes*

สำหรับค่าคงที่ของ attributes จะเหมือนกับ GetAttri โดยที่ไม่มี vbDirectory

- **FileDatetime**

ใช้สำหรับตรวจสอบวันที่ของ File Return ค่าเป็นวันที่

```
=FileDateTime(PathName)
```

ระบุชื่อ File ลงที่ pathName ตัวอย่างการอื่น วันที่ของ C:\COMMAND.COM

```
Dim tmpDate
TmpDate =FileDateTime("C:\Command.Com")
Msgbox(TmpDate)
```

- **FileLen**

ใช้สำหรับตรวจสอบขนาดของ File ค่าที่ Return เป็น Long Integer หน่วนเป็น Byte

```
=FileLen(PathName)
```

ระบุชื่อ File ลงที่ pathName ตัวอย่างการอื่น วันที่ของ C:\COMMAND.COM

```
Dim tmpSize
TmpSize =FileLen("C:\Command.Com")
Msgbox(TmpSize)
```

- **FileCopy**

เป็นคำสั่งสำหรับ Copy File เหมือนคำสั่ง Copy บน Dos ไม่สามารถใช้ WildCard ได้ มี 2 Input คือ File ต้นทาง และ File ปลายทาง

Filecopy *SrcPathName , DestPathName*

ตัวอย่างการ Copy ไฟล์จาก C:\ ไปไว้ที่ tmpBackup

```
Sub CopyAuto()
    Dim tmpStr
    On Error Resume Next
    MkDir "C:\tmpBackup"
    On Error GoTo 0

    tmpStr = Dir("C:\*.*")
    Do Until tmpStr = ""
       FileCopy "C:\" & tmpStr, "C:\tmpBackup\" & tmpStr
       tmpStr = Dir
    Loop

End Sub
```

- **Kill**

เป็นคำสั่งสำหรับลบ File สามารถระบุเป็น wild card ได้ ควรระมัดระวังในการใช้งาน เพราะหากสั่งไม่ดี อาจลบทั้ง Directory ได้

Kill *PathName (Wild Card Allow)*

ตัวอย่างการลบ File ที่ได้ทำไว้ในตัวอย่างไฟล์ Copy

```
Sub KillAuto()
    Dim tmpStr

    tmpStr = Dir("C:\tmpBackup\*.*")
```

```
    Do Until tmpStr = ""
       Kill "C:\tmpBackup\" & tmpStr
       tmpStr = Dir
    Loop

End Sub
```

การเรียกโปรแกรมอื่น ๆ

บ่อยครั้งที่ เราต้องบังคับโปรแกรมของเราให้ออกไปเรียกโปรแกรมอื่นๆ ขึ้นมาทำงานต่อเนื่องจากโปรแกรมที่เราทำงานอยู่ เช่น ทำการ Copy ข้อมูลแล้ว ก็จะต้องทำการ Zip ไฟล์ และ Copy ไปลงที่ Drive A เป็นต้น ตรงนี้เราอาจต้องเรียก Bat ไฟล์ที่ทำไว้

- **Shell**

ใช้สำหรับเรียกโปรแกรมอื่นให้มาทำงาน มีรูปแบบคำสั่งคือ

```
Shell(pathname[,windowstyle])
```

PathName เป็นชื่อโปรแกรมที่ต้องการเรียก ส่วน WindowStyle เป็นการระบุสถานะ Windows ของ Application ที่ถูกเรียก

ค่าคงที่	ค่าจริง	ตำอธิบาย
VbHide	0	เรียก Application แต่ให้อยู่ใน Hide Mode ไม่ถูก Focus
VbNormalFocus	1	ทำงานโดยถูก Focus จากระบบ และอยู่ใน Mode Windows ปกติ
VbMinimizedFocus	2	อยู่ในสถานะ Minimize (Icon) แต่ได้รับ Focus
VbMaximizedFocus	3	อยู่ในสถานะ Maximize และได้รับ Focus
VbNormalNoFocus	4	เปิด Windows ขึ้นมาทำงาน โดยใช้สถานะ Windows ตามปกติ แต่ไม่ต้อง Focus
VbMinimizedNoFocus	6	อยู่ในสถานะ Minimize (Icon) และไม่ได้รับ Focus

การเรียกด้วยคำสั่ง Shell ติดกัน Access จะประมวณผลต่อเนื่อง ไม่เหมือนกับ BAT ไฟล์บน Dos ที่ประมวลผลที่ละคำสั่ง หากเรากำลังใช้ Shell ไปเรียก Application ที่ทำงานต่อเนื่องกัน อาจเกิดปัญหาได้ เพราะลำดับที่ทำก่อหน้า ตัวอย่างเช่น การทำ Zip ไฟล์ต่อไปนี้

```
Sub TestBatX()
    Dim x
    x = Shell("pkunzip a:test.zip c:\tmp", VbNormalFocus)
    x = Shell("command.com /crename C:\tmp\test.mdb test2.mdb", VbNormalFocus)
End Sub
```

ชุดโปรแกรมนี้ ไม่สามารถทำการ Rename ไฟล์ได้ เนื่องจาก เมื่อ Access เรียก pkunzip แล้ว จะไป บรรทัดต่อไปทันที คือการ Rename ซึ่ง โปรแกรม Unzip ยังอาจทำงานไม่เสร็จ

วิธีแก้ปัญหากรณีดังกล่าว ควรทำ Bat ไฟล์ที่เก็บคำสั่งทั้ง 2 ไว้ แล้วเรียก Bat ไฟล์นี้ เพียงครั้งเดียว

```
Sample.BAT
-----------------
pkunzip a:test.zip c:\tmp
rename C:\tmp\test.mdb test2.mdb
-----------------
Sub TestBatX()
    Dim x
    x = Shell("sample.bat", VbNormalFocus)
End Sub
```

- **AppActivate**

ใช้สำหรับเรียกโปรแกรมที่ Run แล้ว ซึ่งจะสังเกตุจาก TaskBar ให้ขึ้นมาเป็น Application ที่รับ Focus แทน เราต้องมีการเรียก Applicationดังกล่าว ด้วย Shell ก่อนเสมอ เพราะถ้าเราเรียก AppActivate โดยไม่มี Application มาก่อน อาจเกิด Error ได้

```
AppActivat title[,wait])
```

Title เป็นชื่อโปรแกรมที่ต้องการเรียก เป็นชื่อที่ปรากฏบน TaskBar ส่วน wait เป็นการระบุให้ Application ที่เรียกได้รับ Focus ทันทีWindowStyle เป็นการระบุสถานะการอ เป็น True หรือ False เป็นการบอกให้ Application ที่ถูกเรียกได้รับ Focus ทันที (False เป็นค่า Default หากไม่ระบุ) หรือ รอให้ Windows ของผู้เรียก ได้รับ Focus ก่อน แล้วจึง ให้ Windows ของ Application ได้รับ Focus (True)

```
Sub TestAppActivate()
    Dim x
    On Error Resume Next
    AppActivate "Calculator"
    If Err Then
        x = Shell("Calc.exe")
        AppActivate "Calculator"
    End If
End Sub
```

ตัวอย่างข้างต้น จะทำการทดสอบการ Activate โดยไม่ระบุการ Shell ก่อน เผื่อว่า Calculator ได้รับการ Run แล้ว หาก พบว่า Error จะทำการเรียก Shell ก่อน แล้วเรียก AppActivate อีกครั้ง

การอ่านเขียน File

ไฟล์ข้อมูลที่มีการใช้กับ Database มีการใช้ Text ไฟล์ หรือ Structure ไฟล์ในการเก็บ เพื่อการโอนย้ายข้ามระบบ การโอนไฟล์ข้ามระบบโดยใช้ Text ไฟล์ เกิดขึ้นบ่อยๆ เช่น การเชื่อมอ่านข้อมูลรายชื่อผู้โอนเงินเข้าบัญชี จาก Text

ไฟล์ ที่ธนาคารส่งมาให้ หรือการที่เราต้องสร้าง Text ไฟล์ ให้กับ ระบบ Computer พวก Main Frame หรือ กระทั่งการใช้ Access ไปสร้าง Wev Page ซึ่งจัดเป็น Text ไฟล์ประเภท หนึ่ง

โดยปกติ Utilities ที่มา Access ก็เพียงพอในการ Convert Text ไฟล์ ที่เป็นปกติ แต่ในสถานะการณ์จริง เราพบว่าเราต้องใช้การเปิด File ขึ้นมาจัดการเองบ่อยครั้ง เราจึงควรทำเข้าใจการสร้าง อ่านเขียนไฟล์ให้เข้าใจ เพื่อประโยชน์กับงาน Programming ในอนาคต

สำหรับในบทนี้ อาจยังไม่พบประโยชน์โดยตรงของการอ่านเขียนไฟล์ด้วยตัวเองนัก จนกว่าจะผ่านบทที่ 5 แล้ว

การเปิด ปิดไฟล์

ปกติ เราจะเปิดปิดไฟล์ จะต้องมีการเปิดสิ่งที่เรียก Handle เพื่อให้ Operting System ทราบว่า มีโปรแกรมของเราซึ่งกำลังใช้ไฟล์อยู่ เราจะต้องขอเลขที่ Handle นี้จาก Operating System ก่อนทำการเปิดไฟล์ เราใช้คำสั่ง FreeFile ในการอ่านค่าดังกล่าว

เปิดแล้วต้องปิด

เมื่อเราเปิดไฟล์แล้ว จะต้องทำการปิดไฟล์ที่เปิดค้างไว้ มิฉะนั้นไฟล์นั้นจะอยู่ในสถานะที่ไม่สมบูรณ์ จนกว่าเราจะปิด Application นั้นทั้งตัว ในที่นี้ก็คือ Access นั่นเอง เราใช้คำสั่ง Close ตามด้วย FileNumber หรือ Handle ที่เปิดไว้

แบบของไฟล์ที่เปิด

ไฟล์ที่เราเปิด แบ่งได้เป็น กลุ่มใหญ่คือ

- ลักษณะเป็น Text File สามารถระบุ Mode ได้เป็น
  1. **Append**: เปิดมาเพื่อเพิ่มรายการ จากเดิมที่ทำไปแล้ว,
  2. **Input**: เปิดเพื่ออ่าน,
  3. **Output**: สร้างไฟล์ใหม่ ถ้ามีแล้วให้ทับไฟล์เดิม

  ใช้ไฟล์ที่จัดเรียงโดยแยกเป็น บรรทัดด้วย Enter และเป็น Ascii ไฟล์ คือ อ่านได้ปกติ
- ลักษณะ **Binary** File คืออ่านมาเป็นตัวๆ ค่า อาจเป็นค่าที่อ่านไม่ได้เป็นตัวอักษร หรือ คือไม่เป็น รหัส Ascii อ่านตามลำดับที่เก็บไว้ใน File จริง
- ลักษณะเป็น **Random** เหมือน Binary แต่เวลาอ่านเขียนจะทำเป็น Record โดยระบุขนาดของ Record แต่ละตัว แบบ Binary คือแบบ Record ที่มีขนาด 1 Byte นั่นเอง

ทำงานกับ Text File

ตัวอย่างการเปิด Text ไฟล์ AUTOEXEC.BAT

```
Sub ReadAutoExec()

    Dim Filex, tmpStr
    Filex = FreeFile

    Open "C:\AUTOEXEC.BAT" For Input As Filex
    Do Until EOF(Filex)
        Line Input #Filex, tmpStr
        Debug.Print (tmpStr)
            '--- another process
            '
            '
    Loop
    Close Filex
End Sub
```

ขั้นตอนตามปกติในการเปิดไฟล์คือ

1. อ่านค่า FreeFile มาเก็บไว้ในตัวแปร เพื่อให้เราทราบว่า ตอนนี้เราได้ลำดับที่ File ที่เท่าไร
2. ใช้คำสั่ง Open ในการเปิด ระบุชื่อ File ตามด้วย Mode ที่การเปิดในที่นี้ คือการ Input และระบุลำดับที่ File
3. ทำการ Do loop โดยใช้ ฟังก์ชั่น EOF ซึ่งจะต้องระบุเลขที่ลำดับไฟล์ที่ขอจากระบบ ในที่ก็คือค่าที่เก็บไว้ใน Filex
4. ใช้คำสั่ง Line Input ในการอ่านไฟล์เข้ามาพักไว้ในตัวแปร ที่ละบรรทัด
5. นำตัวแปร ซึ่งขณะที่เก็บข้อมูลในแต่ละบรรทัดที่อ่านได้ไว้ไปทำการต่อ
6. ปิดไฟล์ เมื่อทำงานเรียบร้อย

สำหรับขั้นตอนการสร้างไฟล์ใหม่ ก็ใช้วิธีการเดียวกัน เพียงแต่เปลี่ยน Mode การ Open จาก Input เป็น Output และ ใช้คำสั่ง Print แทนคำสั่ง Write ทดลองดูตัวอย่างต่อไปนี้ เป็นการ อ่าน File Autoexec.bat มาสร้างเป็น WebPage

```
Sub GenAutoExecHtm()

    Dim Filex, FileHtm, tmpStr, i
    Filex = FreeFile

    Open "C:\AUTOEXEC.BAT" For Input As Filex
    FileHtm = FreeFile
    Open "C:\AUTOEXEC.HTM" For Output As FileHtm

    Print #FileHtm, "<HTML>"
    Print #FileHtm, "<BODY BGCOLOR = WHITE>"
    Print #FileHtm, "<Font Size = +3>Your Autoexec.bat is</font><br>"
    Print #FileHtm, "<TABLE>"
    i = 1
    Do Until EOF(Filex)
        Line Input #Filex, tmpStr
        Print #FileHtm, "<TR><TD>" & i & "): </TD><TD> " & tmpStr & "</TD></TR>"
        i = i + 1
    Loop
    Print #FileHtm, "</TABLE>"
    Print #FileHtm, "</BODY>"
    Print #FileHtm, "</HTML>"
```

```
    Close Filex
    Close FileHtm

End Sub
```

ข้อสังเกตุ

- เราใช้คำสั่ง Print ในการสร้าง Html ทีละบรรทัด
- ในแต่ละบรรทัด ถือเป็น Row ของ Table ใช้ TR คุมการขึ้นบรรทัดหใม่ ในขณะที่ที่เราเริ่มอ่านข้อมูลมาจาก Autoexec.bat ทีละตัว
- ขณะทำการ Print เราต่อ String เพื่อให้ได้ Text ตามที่ต้องการ
- เราขอ FreeFile ติดกัน 2 คำสั่งไม่ได้ เพราะเราจะได้เลขเดียวกัน เนื่องจากเรายังไม่ทำการ Open ไฟล์ จึงต้องทำการ ขอ FreeFile แล้วทำการ Open สลับกันไป

ในบทถัดไป เราจะใช้ กลไกดังกล่าวในการสร้าง Web Page จาก Database โดยการ สร้างเป็น Static Page

เมื่อไรใช้ Append

Append ต่างกับ Output ตรงที่ Append จะไม่ลบไฟล์เดิม เมื่อสั่ง Print จะทำการ Write ข้อมูลต่อจากไฟล์เดิม สมมุติว่าเราลืมใส่ข้อมูลบางส่วน ในตัวอย่างข้างต้น เราอาจใช้ คำสั่ง Append ตามตัวอย่าง

```
Sub ExtendAutoHtm()
    Dim FileHtm
    FileHtm = FreeFile
    Open "C:\AUTOEXEC.HTM" For Append As FileHtm

    Print #FileHtm, "<P>"
    Print #FileHtm, "This page is generated by Uthai Kiattivikrai<BR>"
    Print #FileHtm, "Using VBA<BR>"
    Close FileHtm

End Sub
```

ทำงานกับ Binary File

Binary ไฟล์ เป็นไฟล์ที่อาจเป็นรูปภาพ jpg ไฟล์ หรือ ไฟล์พิเศษอื่นๆ ที่ไม่สามารถพิมพ์ออกมาดูได้ ขณะที่ทำการอ่านไฟล์เราจะต้องรู้ตำแหน่งที่ต้องการอ่านไฟล์เสียก่อน จึงจะทำการอ่านได้ การอ่านไฟล์แบบนี้ จะนับทุก byte ที่เก็บอยู่ในไฟล์ รวมทั้ง CR/LF ที่ใช้ขึ้นบรรทัดใหม่

ตัวอย่างต่อไปนี้เป็นการตรวจสาอบว่า File ที่ระบุ เป็น File BMP หรือไม่ โดย ที่ File BMP จะขึ้นต้น “BM” ที่หัวไฟล์ นั้นคือ ตำแหน่ง 1,2 เราทำตัวแปรขนาด 2 Byte เป็น String เข้าไปรับค่า และระบุตำแหน่งที่ต้องการที่ Byte แรกทันที หากค่าที่อ่านได้เป็น Bmp แสดงว่าเป็น File BMP ถึงแม้จะ Rename นามสกุล ก็ยังสามารถตรวจสอบได้

```
Sub getBmpInfo(bmpFile)
```

```
    Dim Filex
    Dim tmpByte As String * 2
    Dim i
    Filex = FreeFile

    Open bmpFile For Binary As Filex
    i = 1

    Get Filex, i, tmpByte

    Close Filex
    If tmpByte = "BM" Then
       MsgBox ("This is BMP file")
    Else
       MsgBox ("This is Not BMP file")
    End If
End Sub

Sub testBmp()
    getBmpInfo "C:\windows\circles.bmp"
End Sub
```

สำหรับการเขียน Binary File ก็จะใช้คำสั่ง Put แทนคำสั่ง Get ขอให้ระมัดระวังในการ Put File ที่สำคัญๆ เพราะหากเขียนผิด ระบบงานที่ใช้ File ดังกล่าวอาจทำงานผิดพลาดทันที

การทำงานกับ Record File

หรือเรียกอีกแบบว่า Structure ไฟล์ กล่าวคือเก็บไฟล์เป็น Blockๆ แต่ละ Block จะมีขนาดที่เท่ากัน ขณะอ่านกลับก็จะต้องกำหนดขนาดในการอ่านแต่ละครั้ง เราจะอ้างอิง Record แต่ละ Record เป็นลำดับที่ ทดลองสร้างไฟล์ที่เป็น Record ที่เก็บ Record ละ 12 Byte จำนวน 1000 Record ตามตัวอย่าง

```
Sub GenStructurFile()

    Dim Filex
    Dim tmpByte As String * 12
    Dim i
    Filex = FreeFile

    Open "c:\tmptest.txt" For Random As Filex Len = 12
    i = 1

    For i = 1 To 1000
        tmpByte = Format(i, "hh:nn:ss") & Format(i, "0000")
        Put Filex, i, tmpByte
    Next i

    Close Filex

End Sub
```

หากทดลองเปิด File ดังกล่าวด้วย Notepad จะพบว่า ข้อมูลจะต่อกันไปเรื่อยๆ โดยไม่มีการขึ้นบรรทัดใหม่ การอ่านไฟล์ดังกล่าวจึงต้องใช้ Structure ไฟล์ในการอ่าน ซึ่งเพียงแต่ใช้คำสั่ง Get แทนคำสั่ง Put

```
Sub ReadStructurFile()

    Dim Filex
    Dim tmpByte As String * 12
    Dim i
    Filex = FreeFile
```

```
    Open "c:\tmptest.txt" For Random As Filex Len = 12


    For i = 1 To 1000

        Get Filex, i, tmpByte
        Debug.Print tmpByte
    Next i

    Close Filex

End Sub
```

ทดลอง Code ดังกล่าว และอ่านค่าจาก Debug Windows จะพบค่าที่อ่านได้เรียงต่อกันไปอย่างถูกต้อง เราใช้การอ่าน Structure ไฟล์ บ่อยมาก สำหรับการ Port ไฟล์ข้ามระบบ ในกรณีที่ไฟล์ดังกล่าว ไม่มีการขั้นด้วยการขึ้นบรรทัดใหม่

C H A P T E R 5

DATA ACCESS OBJECT

5

เนื้อหาในบทนี้ เป็นการใช้ VB เข้าไปควบคุมฐานข้อมูลของ ACCESS ซึ่งเป็น Object กลุ่ม Database ที่แยกออกจากกลุ่ม Object ที่เราได้เรียนรู้มา

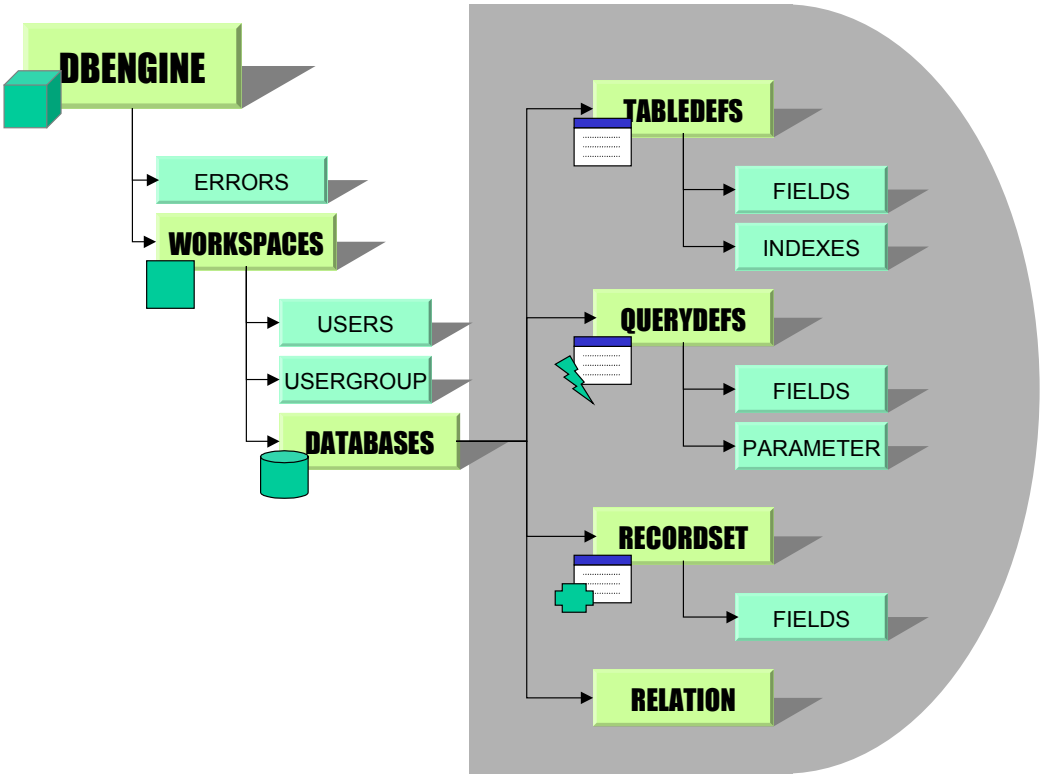
การใช้ DAO (Database Object) ในการพัฒนาโปรแกรม จะทำให้เราสามารถทำระบบงานที่มีความซับซ้อนกว่าปกติที่ Access มีให้ เราสามารถใช้คำสั่งในชุดนี้ ในการสร้างฐานข้อมูลใหม่ สร้าง Table Index การ Scan ข้อมูลที่ละ Record เป็นต้น

ความเข้าใจที่เกิดขึ้นในบทนี้ จะเป็นฐานที่ดีในการใช้งานกับระบบฐานข้อมูลอื่นๆ ในอนาคต และการใช้งาน DAO ของ Visual Basic Verison ปกติด้วย

CONTENT

- Jet Database Engine
- การอ้างอิง databases(0)
- การทำงานกับ Database Object
- การทำงานกับ Record Set
- การทำงานกับ TableDefs

Jet Database Engine



DBENGINE

Access มีกลไกสำคัญคือ Jet Database Engine เป็นตัวจัดการระบบฐานข้อมูลที่ใช้ใน Access รวมทั้งที่ใช้ Visual Basic Version ปกติ Engine ดังกล่าว เสมือนเป็นชุดโปรแกรมที่ทำงานร่วมกับ Access คอยดูแลการเชื่อมต่อข้อมูลกับ Access ให้เรา

ภายใน Dbengine เป็น ชุดโปรแกรม ที่มี Object ย่อยๆ ตามลำดับขั้นที่แสดงในภาพ ทันทีที่เราเรียก Access เราจะเปิด Dbengine 1 Engine ทันที่ Engine จะถูกใช้งานภายใต้ Access สมมุติว่าเราเปิดโปรแกรมสักตัวหนึ่งที่เขียนด้วย VB Version ปกติ ขึ้นมาใช้งาน และใน Application ทำการเชื่อมต่อ Database ตรงนี้จะมีอีกหนึ่ง Dbengine ที่รับใช้โปรแกรมตัวนี้ เพราะอย่างที่กล่าวไว้ว่า Dbengine ก็เป็นส่วนหนึ่งของ Database บน Visual Basic Version ปกติด้วย **ดังนั้น 1 Dbengine จะรับใช้ภายใต้ 1 Application เท่านั้น**

WorkSpaces Object

ภายใต้ 1 Dbengine เรามีอิสระเต็มที่ ที่จะเปิดฐานข้อมูล ได้ทีละหลายตัว ภายใต้พื้นที่ทำงานของเรา 1 พื้นที่ตรงนี้เราเรียกว่า WorkSpaces หากจะเปรียบเทียบ Dbengine เป็นห้องทำงาน Workspaces ก็จะเป็นโต๊ะทำงาน ในหนึ่งห้อง เรามี WorkSpaces หลายๆจุดได้เหมือนเรามีโต๊ะทำงานหลายตัวในห้องทำงานหนึ่ง

ขณะที่เราเปิด 1 Workspaces เราจะต้องมี User ที่ทำงานบน WorkSpaces นั้นเหมือนพนักงานที่ทำงานบนโต๊ะทำงาน ตอนที่เราเข้า Access มาก็เช่นกัน เรากำลังอยู่ที่ Workspaces หมายเลข 0 นั่นเอง โดยมี User เป็น Admin ซึ่งเป็นค่า Default

ปกติเราจะใช้ WorkSpaces เดียวตลอดการใช้งาน เว้นแต่เมื่อมีการเชื่อมโยงกับฐานข้อมูลที่ซับซ้อนมากขึ้น และต้องการพื้นที่ทำงานใหม่เท่านั้น

Database Object

งานที่อยู่บนโต๊ะทำงานของ WorkSpaces ก็คือการเข้าไปยุ่งเกียวกับ Database นั่นเอง ภายใต้ WorkSpaces ก็จะมี Databases เป็นลูกอยู่ภายใต้ WorkSpaces อีกต่อหนึ่ง ลองนึกภาพในขณะที่เรานั่งทำงานที่โต๊ะทำงาน เรามี แฟ้มเอกสารมากมาย เมื่อใดก็ตามที่เราเปิดแฟ้ม 1 แฟ้มขึ้นมาดูเอกสารภายใน ก็เหมือนกับเรากำลังเปิด 1 Database 1 ไฟล์มาใช้งานบนบน WorkSpaces นั่นเอง

เราจะเห็นความซับซ้อนดังกล่าว เนื่องจากความเป็นจริงที่เราต้องทำงานบน Database ที่ละหลายๆไฟล์ หรือหลายตัวพร้อมๆ กันเสมอ ตัวอย่างเช่น เรากำลัง Backup ข้อมูลจาก File Mdb หนึ่งไปอีก Mdb หนึ่งเป็นต้น

การ ATTACH TABLE ? : กรณีที่เรา Attach Table ไว้บน Database , Access จะติดต่อกับ Table ที่ Attach ไว้บนตัวของมัน โดยถือเสมือนหนึ่งว่า เป็น Table ในระบบของ WorkSpaces นั้น ถึงแม้จะเป็น Table ที่มาจากคนละไฟล์ก็ตาม นั่นหมายความว่า หากมีการเปิดไฟล์ที่เป็น Attach Table ตรงนี้จะไม่เกิด Databases Object ใหม่ ยังคงเป็น Database Object เดิม แต่ถ้าเราเรียการ Export /Import ตรงนี้ที่เกิด DatabaseObject ใหม่เกิดขึ้นทันที สังเกตุให้ดี จะพบว่าเวลาที่เราทำการ Export / Import Table เราจะต้องระบุชื่อไฟล์ ตรงนี้เองก็เหมือนการเปิด Database Object ใหม่นั่นเอง

ACTION QUERY: เวลาที่เราสั่งให้มีการทำ Action Query ตรงนี้เราทำงานบน Database Object เป็น Method หนึ่งบน Database เหมือนเราเอาไฟล์เอกสาร 3 ไฟล์ มาเย็บติดกัน แล้วส่งไปถ่ายเอกสาร ที่เราอยู่บน Database Object เพราะ เราสามารถทำงานกับหลาย Table พร้อมๆกันได้ เราสั่งคำสั่งผ่านไปกับ Database Object ตัวเดียวให้ไปทำงานกับ Table เหล่านั้น แล้ว DbEngine ก็ไปรวมรวมองค์ประกอบที่สั่งใน Action Query แล้วไปประมวลผลนั่นเอง เราจะใช้ Action Query ที่สั่งผ่าน Database Object อีกครั้งในบท SQL

Connection คืออะไร

ทุกครั้งที่เรามีการเปิด Table ที่อยู่ใน Database หรือเข้าไปเกี่ยวข้องในแง่มุมใดๆ ก็ตาม กับตัว Database ซึ่งรวมถึงแบบที่ใช้ใน Action Query ก็เช่นกัน ณ ขณะนั้นเรากำลังสร้างที่เรียกว่า Connection คือการเปิดช่องสัญญาณในการคุยกับ Database ตัวนั้นอยู่เหมือนการเปิดไฟล์ (use) บน Dbf

1 Database เปิดหลาย Connection : 1 Database อาจเปิด Connection เข้าไปคุยกับไฟล์ Database อื่นพร้อมๆกันได้หลาย Connection ได้ 1 Connection จะเป็นการต่อกับ 1 Database ไฟล์ ถ้า Connection นั้นเป็น Query แบบ join Table แล้วทุก Table อยู่ใน File Mdb เดียวกัน เราใช้เพียง 1 Connection

แต่ถ้ามาจากหลายไฟล์ เช่นกรณี Attach Table , Database Object ของเราจะต้องวิ่งเข้าไปคุยกับที่ละ Database File ที่ละตัว ทุกครั้งที่มีการเปิดเข้าไปคุย ก็จะเกิด Connection เป็นรายตัวกับแต่ละ Database File **การควบคุม Conection ให้มีจำนวนน้อย จะช่วยให้ระบบทำงานได้เร็วขึ้น**

TableDefs/ QueryDefs/ Relation

คราวนี้เราเปิดฐานข้อมูลออกมา จาก Database เราก็ต้องไปพบอีกว่า ในหนึ่ง File MDB เรายังมี Table ได้มากกว่า 1 Table มี Query ได้มากกว่า 1 ตัว แต่ละ Table ก็มีความสัมพันธ์ถึงกันอีกด้วย ทั้ง 3 Object นี้จะฝังอยู่ใน Database , Object เหล่านี้เป็นเพียง Defination หรือเก็บข้อกำหนดของ Table/Query/Relation ไว้ว่ามีลักษณะอย่างไร มีกี่ Field เป็นต้น เทียบได้กับกับการ Design Table ที่เราจะเห็นแต่โครงสร้างของข้อมูล

เราต้องทำงานกับ Object เหล่านี้ หากเราต้องการสร้าง หรือ แก้ไขฐานข้อมูลในแต่ละไฟล์ Database เหล่านี้ เราต้องตั้งคำถามกับตัวเองว่า ขณะนี้เราอยู่ โต๊ะทำงานตัวใด(WorkSpaces) ที่แฟ้มอะไร (Databases) ก่อนที่เราจะไปทำการแก้ไขข้อมูล แน่นอนว่าเราต้องเปิดแฟ้มก่อนการใช้งาน เราจะพบต่อไปว่า การทำงานใดใดก็ตามบน Database จะต้องระบุเจาะจง Database ก่อนเสมอ

ตอนที่เราทำการ Attach Table ครั้งแรก, Access จะทำการเปิด Database Object ใหม่ แล้วแสดง TabaleDefs ของ Database Object ตัวใหม่ที่เปิด จากนั้นให้เราเลือก พอเลือกเสร็จ Access จะสร้าง TableDefs ใหม่อีกตัวบน Database ตัวที่เรากำลังใช้งานอยู่ จากนั้นทำการคัดลอกเอาค่าต่างๆของ Table ที่ทำการ Attach จากต้นตอ Database เดิมนั้นมาเก็บไว้บน TableDefs แล้วปิด Database Object ตัวที่เปิดใหม่เสีย ตรงนี้แสดงให้เห็นว่า Access จะมี Properties ของ TableDefs ที่ทำให้รู้ว่า TableDefs ตัวนี้ ต่ออยู่กับ Database ข้างนอกที่ไหนอย่างไร

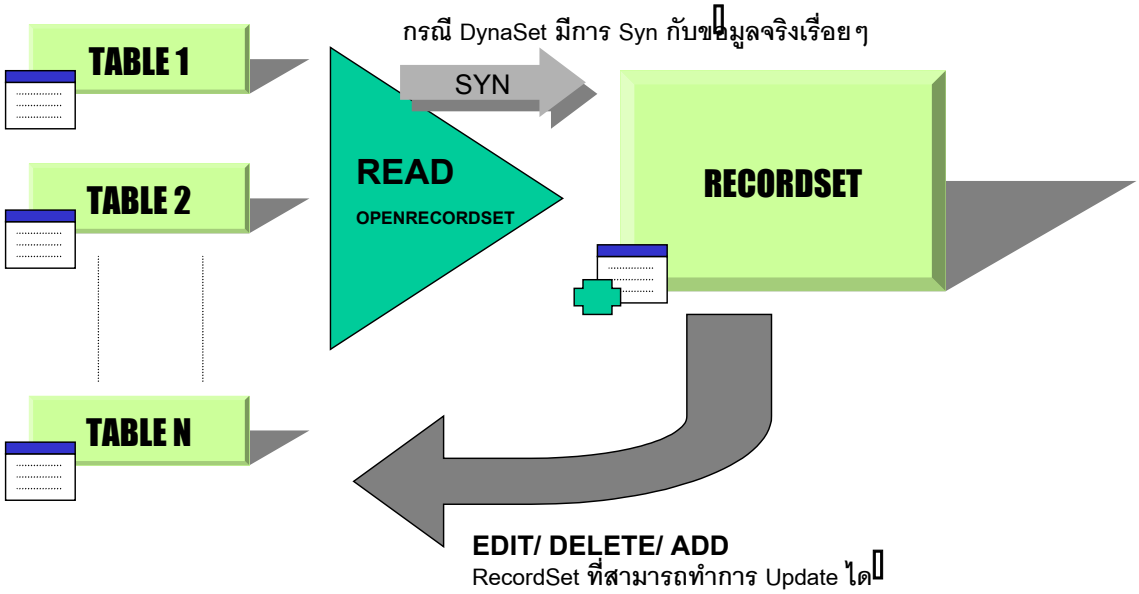
RecordSet

เมื่อเข้าใจกลไกต่างๆแล้ว คราวนี้เราพอเปิด Table เปิด Form ที่ Link กับ Table เราเห็นเนื้อข้อมูลเป็น Record ตรงนี้ เรากำลังเปิดสิ่งที่เรียกว่า Connection เหมือนการเรียก Action Quey แต่เราต้องเปิดค้างไว้ เพื่อใช้งาน ถ้าเป็น Action Query จะเปิดไว้ พอทำเสร็จก็ปิด

แต่ Connection นี้เป็นแบบที่เป็น Cursor คือ วิ่งไปวิ่งมาบน Record แต่ละ Record ได้ กลุ่ม Record นี้เราเรียกว่า RecordSet หรือกลุ่มข้อมูลที่เราเปิดขึ้นมา RecordSet มี Field เป็นองค์ประกอบย่อยๆ อยู่ภายใน RecordSet จะเปรียบเสมือนถังข้อมูลที่เราคัดลอกมากจาก Table อีกต่อหนึ่ง หากเป็น เป็น RecordSet ที่เกิดจาก หลายๆ Table หรือเป็น Query ก็อาจมีข้อมูลจากหลาย Table ได้ด้วย ไม่จำกัดว่าจะต้องมาจาก Table เดียวกันเสมอ

หากเปรียบเทียบแฟ้มข้อมูลเป็น Database ด้านในมี เอกสารหลายชุด (Table) แต่ละชุดมีข้อความเต็มไปหมด (Record) ถ้าเรานำเอกสารไปถ่ายเอกสาร แล้วนำมาอ้างอิง สิ่งที่เราคัดไปถ่ายเอกสารนี้ก็คือ RecordSet นั้นเอง เราอาจตัดข้อความจากหลายชุดเอกสาร มาถ่ายเอกสารพร้อมกัน ก็เหมือนกับกับเราทำ Query ที่มาจากหลาย Table มาเป็น RecordSet ซึ่งเป็นไปได้ในทางปฏิบัติ

RecordSet เป็น Set ของ Record เวลาที่ใช้เราทำงานกับ RecordSet จะมีตำแหน่ง Cursor ปัจจุบันที่เราอยู่ เหมือนเราเปิด Table DataSheet แล้วเราอยู่ที่ Record ใดใด Record หนึ่งเป็นต้น บนหน้าจอจะมีการแสดงตำแหน่งดังกล่าวไว้



RecordSet ใช้เสร็จแล้วต้องปิด เพราะถึงแม้เราจะคัดข้อมูลมาแล้ว แต่ Dbengine จะถือชุดข้อมูลนี้ไว้ และคอยดูแลตำแหน่งของ Record ที่เราอยู่ เป็นภาระงานของ Dbengine เช่นกัน ยิ่งหากเป็น RecordSet ที่เป็น

Dynaset (แบบหนึ่งของ RecordSet) และขณะที่เปิด RecordSet แล้ว มี User คนอื่น Update ข้อมูลในชุดที่เรา Copy มา สิ่งที่อยู่ใน RecordSet ของเราก็จะเปลี่ยนแปลงตามไปด้วย, ตรงนี้ Dbengine ต้องคอยตรวจสอบว่าใครถือ Copy ข้อมูลตัวที่ถูก Update อยู่บ้างแล้วไปแก้ไขชุด Copy ของ Recordset แต่ละคน ซึ่งเป็นภาระงานที่หนักมาก

ในขณะเดียวกัน เราก็สามารถ Update Recordset ได้ จาก RecordSet ที่เราเปิดอยู่ การ Update RecordSet เป็นการเข้าไปแก้ไขข้อมูลต้นทางที่เราคัดลอกมาจาก Table ต้นทางนั่นเอง ตรงนี้ก็เป็นหน้าที่ของ DbEngine อีกเช่นกัน

การอ้างอิง Databases(0)

อย่างที่เราทราบว่า DataBases(0) เป็น Database ตัวที่เราทำงานอยู่ด้วยเป็นส่วนใหญ่ การอ้างอิง Databases(0) นี้ ทำได้หลายวิธี DataBase จะอยู่ใต้ Dbengine และ WorkSpaces ตามลำดับ การอ้างอิง Databases(0) ทำได้หลายวิธี เช่น

```
Sub dbEngineSet()

    Dim rst As Recordset

    Set rst = dbEngine.Workspaces(0).Databases(0).OpenRecordset("Suppliers")
    Set rst = Workspaces(0).Databases(0).OpenRecordset("Suppliers")
    Set rst = dbEngine(0)(0).OpenRecordset("Suppliers")
    Set rst = CurrentDb.OpenRecordset("Suppliers")

End Sub
```

ตัวอย่างข้างต้น เป็นการ Set DB แล้วทำการเปิด RecordSet ซึ่งทำได้หลายวิธี ดังต่อไปนี้

- แบบแรกใช้ dbEngine.WorkSpaces(0).Databases(0) วิธีการอ้างอิงดังกล่าว จะต้องมี dbEngine และ WorkSpaces(0) นำหน้า เราใช้การอ้างอิงแบบนี้ ในกรณีอื่นมากกว่า เพราะเขียนค่อนข้างยาว กรณีที่ต้องใช้Dbengine นำหน้าได้แก่การสร้าง Database ใหม่เป็นต้น แต่เป็นแบบที่มาตราฐานที่สุด
- แบบที่ 2 ละคำว่า Dbengine ไว้ เหมือนแบบแรกทุกประการ
- สำหรับแบบที่ 3 ใช้ dbEngine(0)(0) เป็นแบบที่ดีที่สุด ซึ่งหมายถึง dbEngine.Workspaces(0).Databases(0) นั่นเอง แต่เขียนสั้นกว่า **ควรใช้แบบนี้**
- แบบที่ 4 นี้เขียนง่าย แต่ไม่ดี เนื่องจาก Access จะเปิด WorkSpaces ใหม่ ทำให้เปลืองทรัพยากรของระบบ

การทำงานกับ WorkSpace Object

เรามี Databases(0) สำหรับใช้งาน ซึ่งเป็น Database หลัก คราวนี้ หากเราต้องเปิด Mdb ตัวอื่น แล้วเข้าไปเอาฐานข้อมูลมาใช้ โดยไม่มีการ Attach ล่วงหน้า เราจะทำอย่างไร ? เราต้องสั่งตัว WorkSpace ให้ทำการเปิด Database ขึ้นมาใช้งาน

บน Object ทั่วไปจะมี Method ในการใช้งาน สำหรับ Object นั้นๆ ตัว WorkSpace Object ก็เช่นกัน มี Method ที่ใช้งานเช่นกัน โดยมี Method ที่ใช้บ่อยๆ คือ

- **OpenDatabase**

เป็นการเปิดไฟล์ Database ตามปกติ ในหนังสือนี้ จะไม่กล่าวถึงการ OpenDatabase ที่ซับซ้อนกว่าการ OpenDatabase ตามปกติ คำสั่งที่ใช้ในการ OpenDatabase คือ

```
Set database = workspace.OpenDatabase (name)
```

การ OpenDatabase จริงๆจะมีเงื่อนไขที่ระบุได้สลับซับซ้อนมากกว่านี้ การ OpenDatabase ตามปกติ ใช้เพียงชื่อไฟล์ก็เพียงพอ

```
Sub dbOpenTest()
    Dim db As Database

    MsgBox ("Before Open :" & Workspaces(0).Databases.Count)
    Set db = Workspaces(0).OpenDatabase("C:\AVSREG\ENROLL.MDB")
    MsgBox ("After Open:" & Workspaces(0).Databases.Count)
    Set db = Nothing
    MsgBox ("After Close / Set Nothing : " & Workspaces(0).Databases.Count)

End Sub
```

ตัวอย่างข้างต้น แสดงให้เห็นว่า เมื่อมี OpenDatabase แล้ว จำนวน Database ที่เปิดใน WorkSpaces จะมากขึ้นทันที และเมื่อเราทำการ OpenDatabase ทุกครั้ง เมื่อใช้เสร็จ ควรใช้คำสั่ง Set Db = Nothing ทุกครั้งเพื่อปล่อยทรัพยากรของระบบให้ว่าง หลังจากที่เราปล่อย Database แล้ว จำนวน Database ใน WorkSpaces ก็จะลดลงทันที

- **CreateDatabase**

เป็นการสั่งให้มีการสร้าง Database ไฟล์ใหม่ โดยเราจะต้องระบุชื่อ ไฟล์ที่ต้องการสร้าง และภาษาของ Database ที่ต้องการสร้างเป็นอย่างน้อย มีรูปแบบคำสั่งคือ

```
Set database = workspace.CreateDatabase (name, locale, options)
```

ค่าที่ระบุ	คำอธิบาย
Name	เป็นชื่อไฟล์ของ Database ที่ต้องการสร้าง
Local	เป็นรหัสที่บอกภาษาของตัว Database ให้เปิด Help ของค่ารหัสนี้เอง แต่สำหรับ

ค่าที่ระบุ	คำอธิบาย
	Database ไทย ใช้ DbLangThai ส่วน อังกฤษ ใช้ DbLangGeneral
Option	เป็นเงื่อนไขในการสร้าง Database ตามตารางถัดไป สามารถระบุ Version พร้อมกับการ Encrypt Database ได้ โดยใช้ค่า Constant ของ Version บวกกับค่า Constant ของการ Encrypt

ค่าคงที่ของ Option	คำอธิบาย
DbEncrypt	ทำ Database แบบ Encrypt
DbVersion10	ใช้ FileFormat ของ Version 1.0
DbVersion11	ใช้ FileFormat ของ Version 1.1
DbVersion20	ใช้ FileFormat ของ Version 2.0
DbVersion30	ใช้ FileFormat ของ Version 3.0 ใช้กับ 3.5 ได้ ตัวนี้เป็นค่า Default

```
Sub dbCreateSample()
    Dim db As Database
    Dim i As Integer
    Dim tmpName

    On Error Resume Next
    MkDir "C:\tmpDb"
    On Error GoTo 0

    For i = 1 To 10
        tmpName = "C:\tmpdb\dbtest" & i & ".mdb"
        If Dir(tmpName) <> "" Then
           Kill "C:\tmpdb\dbtest" & i & ".mdb"
        End If
        Set db = dbEngine.Workspaces(0).CreateDatabase(tmpName, dbLangThai)

    Next I

End Sub
```

ตัวอย่างข้างต้นเป็นการสร้าง Database 10 Files ลงที่ Directory C:\TmpDb โดยทำการตรวจสอบว่ามี File เก่าอยู่เดิมหรือไม่ ก่อนทำการสร้าง ถ้ามี จะลบทิ้ง แล้วทำการสร้างใหม่

การสร้าง Database ทุกครั้ง DbEngine จะเปิด Database ตัวนั้นไปในในตัว ***เราไม่ต้องสั่ง OpenDatabase ซ้ำหลังจากทำการ Open เพราะ DbEngine จะทำการ Open แล้วเก็บการ OpenDatabase ไว้ในตัวแปรที่มารอรับขณะ CreateDatabase ให้เลย*** ในที่นี้ ก็คือตัวแปรนั่นเอง db

การทำงานกับ RecordSet

เราเปิด RecordSet ถัดจาก WorkSpaces และ Database เป็นการเรียก Set ของข้อมูลที่ต้องการมาใช้งานเฉพาะอย่าง เราใช้ RecordSet สำหรับการจัดการฐานข้อมูลที่ซับซ้อน ที่ไม่สามารถใช้ Query ทำการแทนได้ หรือการเปิดฐานข้อมูล เพื่ออ่านค่า หลายอย่างพร้อมกัน การเปิด RecordSet ใช้คำสั่งดังนี้

การเปิด RecordSet

ใช้คำสั่ง OpenRecordSet มีรูปแบบคำสั่งคือ

```
Set recordset = object.OpenRecordset (source[, type][, options][, lockedits])
```

ตัวแปรที่ระบุ	คำอธิบาย
Recordset	ตัวแปรที่นำมารับ RecordSet ที่เปิดได้
Object	Object ที่ให้ RecordSet นอกจากจะเป็น Database แล้ว ยังใช้กับ Object ที่เป็น Connection ได้ด้วย ในหนังสือนี้จะกล่าวเฉพาะการเปิด RecordSet จาก Database เท่านั้น
Source	ระบุแหล่งข้อมูลที่นำมาเป็นข้อมูลใน RecordSet อาจเป็นชื่อ Table, Query หรือ SQL Command ก็ได้
Type	ระบุวิธีการเปิด RecordSet ตามตาราง
Options	ระบุ Option พิเศษที่ทำการเปิดฐานข้อมูล
Lockedits	ระบุก็ได้ ไม่ระบุก็ได้ เป็นการบอกเงื่อนไขในการ Lock ข้อมูลขณะทำการแก้ไขข้อมูล กรณีอยู่ในสภาพที่มี User หลายคน เงื่อนไขนี้มีความหมายมากสำหรับการทำงานบน Multiuser

Type วิธีการเปิด	คำอธิบาย
DbOpenTable	เปิดแบบ Table กรณีที่ระบุใน Source เป็น Table แบบนี้คล้ายกับ Dynaset แต่มีข้อแตกต่างที่เป็นการเปิดกับ Table เดียว หากเราไม่ระบุอะไรที่ Type, Access จะพยายามเปิดด้วย Type นี้ก่อน
DbOpenDynamic	เปิด dynamic ทำการ Update ตามข้อมูลจริงตลอดเวลา ใช้งานกับ ODBC เท่านั้น

Type วิธีการเปิด	คำอธิบาย
DbOpenDynaset	เปิดข้อมูลแบบ dynamic และทำการ Update ตามข้อมูลจริงตลอดเวลา คล้ายแบบ Table แต่สามารถเปิด หลายๆ Table ได้ หาก Access พบว่าไม่สามารถเปิดแบบ dbOpenTable ได้ และไม่มีการกำหนดเงื่อนไขอื่นๆ Access จะเปิดในแบบ Type นี้
DbOpenSnapshot	ทำสำเนาข้อมูลไว้ หากมีการแก้ไขจาก Table ต้นฉบับ จะไม่มีการ Update ตามการเปลี่ยนแปลง แบบนี้เป็นแบบที่เร็วมาก เราใช้ในการอ่านข้อมูลจาก Table โดยการเปิด RecordSet แบบนี้
DbOpenForwardOnly	สำหรับ RecordSet ที่อ่านไปข้างหน้าอย่างดียว ไม่สามารถสั่งให้ RecordSet เลื่อนตำแหน่ง Cursor ถ่อยหลังได้ แบบนี้ทำงานเร็วเช่นกัน

Option ในการเปิด	Description
dbAppendOnly	อนุญาติให้ Recordset ที่เปิดสามารถเพิ่มข้อมูลเข้าไปได้ แต่ไม่สามารถแก้ไข หรือลบข้อมูลได้ เราใช้วิธีการเปิดแบบนี้สำหรับ Recordset ที่เปิดเพื่อทำการ Add เท่านั้น ซึ่งจะเร็วกว่าเปิด Table ปกติ แล้วมาทำการ Add
DbSQLPassThrough	เป็นการสั่งให้ส่งคำสั่ง SQL ที่ทำเป็นตัวสร้าง Recordset เข้าสู่ตัว ODBC ที่ Database Engine ต่ออยู่โดยตรง แบบนี้ทำให้ให้ประมวลผลเร็วขึ้น สำหรับกรณีที่ใช้ ODBC Database
DbSeeChanges	บังคับให้ Dbengine เกิด Error หาก Recordset ที่เราเปิดอยู่มีการอ่านโดย User คนอื่นๆ ในขณะที่เราทำการเขียนข้อมูลเข้าไปใน Table ที่เป็นส่วนหนึ่งใน RecordSet ที่เรากำลังเปิด การใช้ Option นี้จะมีประโยชน์ในการใช้กับระบบฐานข้อมูลที่มีการเขียนอ่านข้อมูล พร้อมกันเสมอ ๆ.
dbDenyWrite	เปิดแบบ Lock การเขียนคือ ระหว่างที่เปิด RecordSet นี้ ไม่อนุญาติให้ User อื่นที่กำลังเปิด Record ที่อยู่ใน Recordset นี้ ทำการเขียนข้อมูลได้ จนกว่าจะมีการปิด RecordSet นี้เสียก่อน ใช้สำหรับการ Lock เลขที่นั่ง Lock เลขใบเสร็จเป็นต้น
dbDenyRead	เปิดแบบ Lock การอ่าน คือ ระหว่างที่เปิด RecordSet นี้ ไม่อนุญาติให้ User อื่นที่กำลังเปิด Record ที่อยู่ใน Recordset นี้ ทำการอ่านข้อมูลชุดนี้ จนกว่าจะปิด RecordSet นี้ ใช้สำหรับการ Lock เลขที่นั่ง Lock เลขใบเสร็จเป็นต้น

Option ในการเปิด	Description
dbForwardOnly	สำหรับ RecordSet ที่อ่านไปข้างหน้าอย่างดียว ไม่สามารถสั่งให้ RecordSet เลือนตำแหน่ง Cursor ถ่อยหลังได้ แบบนี้เร็วที่สุด ตรงนี้คล้ายกับที่กำหนดใน Type แต่เราควรใช้วิธีกำหนดใน Type มากกว่า เพราะการกำหนดใน Option นี้เป็นของ Access Version เก่า ใน Version ยังคงมีไว้ เพื่อให้ Code เก่า Compat กันเท่านั้น
dbReadOnly	เปิดข้อมูลแบบอ่านอย่างเดียว เขียน แก้ไขไม่ได้
DbRunAsync	ทำงานแบบ asynchronous กับ ODBC ใช้กับ ODBCDirect workspaces เท่านั้น
dbExecDirect	ใช้งานสำหรับ ODBC ทำการเปิด RecordSet โดยไม่ทำการใช้คำสั่ง SQLPrepare ก่อนทำการประมวลผล ใช้วิธีส่งคำสั่งเข้าไปประมวลผลทันที (ใช้ SQLExecDirect)
dbInconsistent	อนุญาติให้มีการ Update Recordset ที่ Field ข้างใดข้างหนึ่งของการ Join Table สำหรับ กรณีที่มีการเปิด Table มากกว่า 1 Table และทำการ Join กันอยู่ โดย Update ที่ Field ที่ Join ด้านใด ด้านหนึ่ง โดยไม่ต้อง Update ทั้ง 2 ข้าง หากการ Update นั้นไม่ผิด Referential
dbConsistent	ตรงกันข้ามกับการเปิดแบบ dbInconsistent กล่าวคือ ต้อง Update Field ที่ Join กันทั้ง 2 ด้านให้ครบ และเหมือนกัน

LockEdits วิธี Lock Record ขณะเขียน	คำอธิบาย
DbReadOnly	เปิดแบบอ่านอย่างดียว เขียนไม่ได้ ตรงนี้จะซ้ำซ้อนกับ Type, Option สามารถใช้จาก Type หรือ Option ก็ได้
dbPessimistic	ทำการ Lock Record ทันทีที่มีการสั่งคำสั่ง EDIT (**EDIT METHOD**)
dbOptimistic	ทำการ Lock Record หลังจากที่สั่ง Update แล้ว (**UPDATE METHOD**)

จะเห็นว่า Type, Option, Lock ในการเปิด Recordset ค่อนข้างซับซ้อน แต่การใช้งานจริง จะใช้อยู่ไม่กี่ตัว เช่น

วัตถุประสงค์ในการเปิด	คำสั่ง
เปิดแบบเพื่ออ่านอย่างเดียว ใช้	`set rst = dbengine(0)(0).openrecordset("Suppliers",dbOpenSnapShot)`
เปิดเพื่อ Scan Record (เดินหน้าอย่างเดียว)	`set rst = dbengine(0)(0).openrecordset("Suppliers",dbOpenSnapShot, dbForwardOnly)`
เปิดเพื่อทำการ Update ข้อมูล	`set rst = dbengine(0)(0).openrecordset("Suppliers")` เพราะไม่ระบุอะไร จะเป็น dbOpenDynaset หรือ dbOpenTable อยู่แล้ว

หากเราต้องใช้ Option พิเศษในการเปิด จึงค่อยระบุ Option เหล่านั้นเข้าไป หากต้องการเปิด Recordset จาก Table มากกว่า 1 Table สามารถใช้ Query ที่ทำมาจาก หลายๆ Table Join กัน หรือ จะใส่เป็น SQL COMMAND เลยก็ได้ อันนี้จะกล่าวถึงอีครั้ง ในบทของ SQL

การอ่านค่าจาก RecordSet

เราอ่านค่าจาก Recordset โดยใช้การอ่านค่าแบบตัวแปร Array ธรรมดา สามารถอ้างอิงได้ 2 วิธี คือ

วิธีอ้างอิง	คำอธิบาย
rst(IndexNumber)	ใช้รหัสลำดับที่ Field ที่เลือกมาจากฐานข้อมูลมาแสดง เช่น rst(0),rst(1) เป็นต้น ตัวอย่าง Recordset จาก Table Suppliers Filed แรกคือรหัสดัชนี 0 หรือ SupplierID นั่นเอง
rst(FieldName)	ระบุชื่อ Field โดย FieldName เป็นตัวแปรแบบ String

```
Sub RecordSetSup()
    Dim rst As Recordset
    Set rst =
dbEngine(0)(0).OpenRecordset("Suppliers",dbOpenSnapShot,dbForwardOnly)

    Do Until rst.EOF
        Debug.Print rst(0), rst("SupplierID")
        rst.MoveNext

    Loop
    Set rst = Nothing

End Sub
```

ตัวอย่างด้านบนเป็นการ Print ค่าจาก Recordset ที่ Field 0 หรือ Field ชื่อ SupplierID โดยการใช้การอ้างอิงทั้ง 2 แบบ สำหรับ RecordSet.MoveNext เป็นคำสั่งในการบังคับให้ RecordSet เคลื่อนไปลำดับที่ Record ต่อไป คำสั่ง Rst.Eof เป็นการตรวจสอบ Properties EOF ของ Rst ว่าเลื่อนไปจนหมด Record สุดท้ายหรือยัง

เราใช้ Recordset ในการหยิบค่าจาก Table แทนการใช้คำสั่ง Dlookup โดยเฉพาะกรณีที่เราต้องการอ่านค่าจาก Table หลายครั้ง เช่นการอ่าน ชื่อสินค้า และ ราคา หากเราใช้ Dlookup 2 ครั้งก็จะได้ค่าเหมือนกันแต่จะช้ากว่า ซึ่งความเป็นจริง คำสั่ง Dlookup ก็เป็นการเปิด Recordset เหมือนกับที่เราทำทุกประการ ดังนั้นเราจึงควรใช้ RecordSet มากกว่าในการเปิดค่าดังกล่าวมากกว่า ตัวอย่างเช่น

```
Sub GetProductPrice()
    Dim rst As Recordset
    Dim x
    x = InputBox("Enter ProductID")

    Set rst = dbEngine(0)(0).OpenRecordset("Select ProductName, UnitPrice from
Products where ProductID = " & x, dbOpenSnapshot)
    If rst.RecordCount > 0 Then
       MsgBox (rst(0) & " - " & rst(1))
    Else
       MsgBox ("ID not found")
    End If
    Set rst = Nothing
End Sub
```

ตัวอย่างข้างต้น อาจดูซับซ้อนขึ้นอีกนิด ที่การใช้ SQL String แทนการใช้ Table เป็น Source Table โดยมีการ Where รหัส ProductID ด้วย เมื่อได้ผลจากการเปิด Recordset แล้ว ก็นำค่าที่ได้จาก Rst(0) คือชื่อสินค้า และ Rst (1) คือราคามาแสดง ในทางปฏิบัติเราอาจนำไปเก็บไว้ในตัวแปรเพื่อใช้ในการคำนวณเป็นต้น เมื่อใดก็ตามที่มีการอ่านค่าจาก Table มากกว่า 1 Field เรานิยมใช้วิธีการดังกล่าวแทนคำสั่ง Dlookup

การเลื่อนตำแหน่ง Record

คำสั่งในการเลื่อน Record จะทำให้ตำแหน่ง Current Record ของ RecordSet เลื่อนไป และค่าที่อ่านออกมาจะเป็นไปตามค่าของข้อมูล ณ ขณะที่อยู่ที่ Record นั้นๆ ประกอบด้วยคำสั่ง 4 ตัวดังตาราง วิธีการสั่ง จะคล้ายกับการใช้ Method ตามปกติ

คำสั่ง	วิธีการเลื่อน
MoveNext	สั่งให้ไป Record ถัดไป
MovePrevious	สั่งให้ไป Record ก่อนหน้า
MoveFirst	สั่งให้ไป Record แรก
MoveLast	สั่งให้ไป Record สุดท้าย

ตัวอย่างที่อยู่ในเรื่องการอ่านค่าจาก RecordSet ได้แสดงไว้แล้ว สำหรับ MoveLast เรายังใช้ในการบังคับ ให้ Access อ่านค่าจำนวน Record ทั้งหมดออกมาด้วย

RECORDSET PROPERTIES ที่สำคัญ

เราใช้ Recordset Properties หลายตัวในการตรวจสอบสถานะของ Recordset ขณะนั้น ได้แก่

- **Recordcount**

ใช้อ่านจำนวน Record ที่มีใน Recordset ทันทีที่เราเปิด Recordset หากพบว่ามีข้อมูลอย่างน้อย 1 Record ค่า RecordCount จะเป็น 1 และค่อยๆ เพิ่มค่าไปเรื่อยๆ จนเป็นจำนวน Record ทั้งหมด ดังนั้น **เมื่อเราเปิด RecordSet ครั้งแรก เราจะอ่านค่า RecordCount ที่ยังไม่ถูกต้อง** จนกว่า Access จะตรวจจำนวน Record ทั้งหมดได้ หรือใช้ วิธีสั่ง MoveLast ทันที เพื่อบังคับให้ Access เลื่อนไปยัง Record สุดท้ายทันที ณขณะนั้น Access จะได้จำนวน RecordCount แต่หาก Table นั้นมีจำนวน Record มากก็อาจใช้เวลานาน

อย่างไรก็ตาม เรายังใช้ RecordCount ในการตรวจสอบว่า การ OpenRecordSet ขอแรา ได้ Record ออกมาหรือไม่ โดยตรวจว่า RecordCount > 0 หรือไม่ ตามตัวอย่าง เพราะถ้ามากว่าศูนย์ แสดงว่าผลของการ OpenRecordSet ได้ Record แน่นอน เพียงแต่ยังไม่ทราบว่าเป็นเท่าไร

```
Sub RecordSetSup()
    Dim rst As Recordset
    Set rst = dbEngine(0)(0).OpenRecordset("Suppliers", dbOpenSnapshot)
    If rst.RecordCount = 0 Then
       MsgBox ("No Record Found")
       Set rst = Nothing
       Exit Sub
    End If

    rst.MoveLast
    rst.MoveFirst

    MsgBox ("Total Record is: " & rst.RecordCount)

    Do Until rst.EOF
        Debug.Print rst.AbsolutePosition, rst(0), rst(1)
       rst.MoveNext

    Loop
    rst.MovePrevious
    MsgBox ("Last Record is :" & rst(0) & " - " & rst(1))

    Set rst = Nothing

End Sub
```

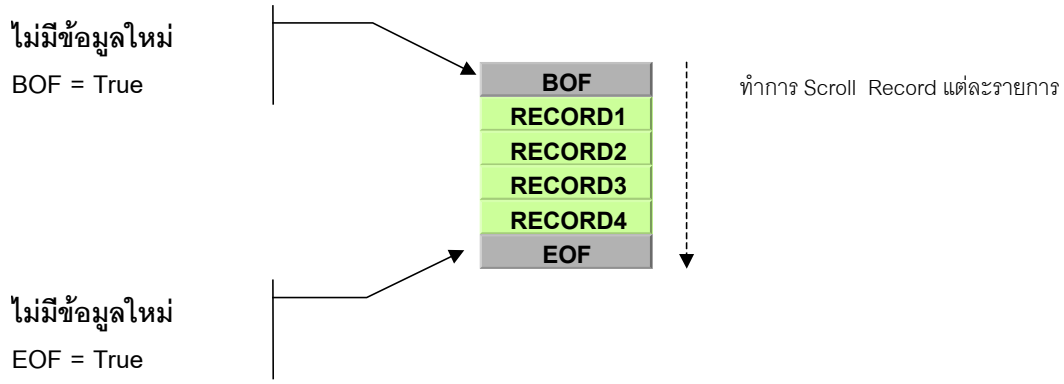
ทดลองดู Code ตามตัวอย่างจะพบว่า

- มีการตรวจสอบ RecordCount หลังจาก Open หาก RecordCount = 0 ก็จะเตือนด้วย MsgBox
- สั่งให้มีการ MoveLast และ MoveFirst เพื่อต้องการทราบจำนวน Record ทั้งหมด

- เมื่อทำการ Scroll ไปทีละ Record แล้ว จึงสั่งให้ MovePrevious เพื่อเอา Record สุดท้าย เพราะขณะ EOF จะไม่มี Record

- **AbsolutePosition**

ใช้ในการอ่านค่าตำแหน่งของ RecordSet ปัจจุบัน โดยนับจาก 0 จนถึง RecordCount –1



- **EOF**

ใช้ในการตรวจสอบว่า ขณะนี้ Recordset มาอยู่ที่ตำแหน่งท้าย File หรือยัง Return ค่า True หรือ False เราใช้ Properties นี้ในการป้องกันการอ่านข้อมูลจาก RecordSet ในขณะที่อยู่ท้ายสุด ซึ่งตำแหน่งดังกล่าว หากมีการอ่านข้อมูล จะเกิด Error เพราะไม่มีข้อมูลให้อ่าน

- **BOF**

ใช้ในการตรวจสอบว่าขณะนี้ Recordset มาอยู่ที่ Record ส่วนบนสุดของ RecordSet หรือไม่ คล้ายกับ EOF แต่อยู่คนละข้าง ปกติ Properties นี้จะเป็น False เว้นแต่มีการ Scroll RecordSet กลับหลังจนเลย Record แรกไป Return ค่า True หรือ False เช่นกัน

การ Add ข้อมูล

การ Add ข้อมูลลงใน Table โดยใช้ Recordset สามารถทำได้ โดยการใช้คำสั่ง ADDNEW แล้วปิดท้าย ด้วยคำสั่ง Update การระบุค่าที่ต้องการ Add ก็เพียงแต่ ใส่ค่าลงใน Field ตามตัวอย่าง

```
rst.AddNew
rst("SupplierName") =  "Mr. Bean"
rst.Update
```

สำหรับ Field ที่เป็น Counter หรือ Field ที่มีค่า Default อยู่แล้ว ตัว Dbengine จะจัดการใส่ค่าให้เราเอง เช่นการใส่เลขลำดับถัดไป หรือใส่ค่า Default ให้ หากเราเว้น Field พวกนั้นไว้ ทดลองดูการ Add ข้อมูลลงใน Table Products จำนวน 100 Record โดยการเปิด RecordSet สำหรับ การ Add โดยเฉพาะ

```
Sub AddTimeProduct()

    Dim rst As Recordset
    Dim i As Integer
    Set rst = dbEngine(0)(0).OpenRecordset("Products", dbOpenDynaset,
dbAppendOnly)
    For i = 0 To 100
        rst.AddNew

            rst("ProductName") = "Add Number: " & i & " " & Format(Now, "dd/mm/bb
hh:nn:ss")
        rst.Update


    Next i
    Set rst = Nothing

End Sub
```

คำสั่ง AddNew ไม่จำเป็นต้องใช้กับการเปิด Recordset ใน Option แบบ dbAppenOnly เท่านั้น อาจใช้กับการ Open ใน Option อื่นก็ได้ ขอให้การ Open นั้นไม่เป็นการ Open ที่เป็น ReadOnly ก็พอ แต่การระบุ dbAppendOnly จะทำให้ Recordset ที่เราเลือกขึ้นมา มีจำนวน RecordCount เป็น 0 และรอการ Add ทันที ตรงนี้จะลดภาระงานของ dbEngine ที่ต้องไป Scroll Record เพื่อให้ทราบจำนวน Record ทั้งหมด แล้วให้เราไป Add ที่ Record สุดท้าย ซึ่งจะเสียเวลากว่ามาก

การลบข้อมูล

การลบข้อมูลใน RecordSet ใช้คำสั่ง Delete ที่ขณะที่ ตำแหน่งของ Record อยู่ที่ Record ที่ต้องการลบ ทดลองดูตัวอย่างที่ใช้ในการลบข้อมูลที่ได้ทำการ Add ไว้ในตัวอย่างการ Add ข้อมูลข้างต้น โปรแกรมจะตรวจสอบที่ชื่อ ProductName หากมี คำว่า “Add Number “ ก็จะทำการลบ Record นั้นเสีย

```
Sub DeleteProduct()
    Dim rst As Recordset
    Dim i As Integer
    Set rst = dbEngine(0)(0).OpenRecordset("Products", dbOpenDynaset)
    Do Until rst.EOF

        If rst("ProductName") Like "Add Number:*" Then

            rst.Delete

        End If
        rst.MoveNext

    Loop
    Set rst = Nothing

End Sub
```

การแก้ไขข้อมูล

การแก้ไขข้อมูลบน RecordSet ใช้คำสั่ง EDIT แล้วตามด้วยคำสั่ง Update ในหว่าง EDIT และ Update เราจะกำหนดค่าที่ต้องการเปลี่ยนแปลงลงที่ RecordSet ณ Field ที่ต้องการแก้ไข หากเราสั่งเปลี่ยนแปลงค่าที่ RecordSet โดยที่ไม่อยู่ในระหว่างคำสั่ง EDIT และ UPDATE จะเกิด Error ขึ้น

```
Sub ProductPriceAdjust()
    Dim rst As Recordset
    Dim i As Integer
    Dim ratio
    Set rst = dbEngine(0)(0).OpenRecordset("Products", dbOpenDynaset)
    Do Until rst.EOF

        Select Case rst("UnitPrice")
        Case Is <= 5
             ratio = 0.05
        Case Is <= 10
             ratio = 0.06
        Case Is <= 20
             ratio = 0.08
        Case Is <= 50
             ratio = 0.1
        Case Is <= 100
             ratio = 0.15
        Case Is > 100
             ratio = 0.2
        End Select

        rst.Edit
            rst("UnitPrice") = rst("UnitPrice") * (1 + ratio)
        rst.Update

        rst.MoveNext

    Loop
    Set rst = Nothing
End Sub
```

ตัวอย่างโปรแกรมข้างต้นเป็นการ Update ราคาใน Table Product ซึ่งมีกลไกการ Update ตามขนาดราคาที่มีอยู่เดิม โดยเก็บค่าที่ต้องการขึ้นราคาไว้ใน Ratio แล้วนำไปเปลี่ยนราคา ระหว่างที่คำสั่ง EDIT และ UPDATE

การใช้คำสั่ง Edit เพื่อให้เราสามารถทำการ Update ข้อมูลที่ซับซ้อนกว่าการใช้ Query ปกติได้ การได้มีโอกาศให้เราทดสอบค่าในแต่ละ Record ในแต่ละ Field ก่อนทำการ Update ทำให้เราสามารถตัดสินใจเลือกเงื่อนไขที่เหมาะสมในการ Update ข้อมูล ในแต่ละ Record ได้ซับซ้อนกว่า การใช้ Update Query ธรรมดา

การกำหนดวิธีการ Lock ในการ OpenRecordSet จะส่งผลต่อการ Lock record ในการ Edit / Update เนื่องจากกลไกการ Update ข้อมูล จะเริ่มจาก Edit และ Update การเลือกที่จะเริ่ม Lock ที่คำสั่ง Edit (dbPessimistic) หรือ Lock เมื่อจะเข้าไป Update (dbOptimistic) จะทำให้ระยะเวลาในการ Lock Database แตกต่างกัน เพราะในขณะที่เราเริ่ม EDIT อาจมีคนอื่น Edit Record เดียวกัน พร้อมกับเรา พร้อมทั้งทำการ Update ก่อนที่เรา Update หากเราต้องอ่านข้อมูลใน Record นั้นมาตรวจสอบ อาจได้ค่าผิดพลาดได้

การ Copy RecordSet จาก Form

ตามที่เราทราบว่า Form ก็ทำหน้าที่เปิด RecordSet อยู่เช่นกัน เราสามารถ Copy RecordSet จาก Form ได้โดยการใช้คำสั่ง RecordSetClone จาก Object Form ตัวอย่างต่อไปนี้ ใช้การอ้างอิงด้วย Me

```
Private Sub Form_Load()
            Dim rst As Recordset
            Set rst = Me.RecordsetClone
            rst.MoveLast
            MsgBox ("This form have : " & rst.RecordCount & " record.")
             Set rst = Nothing


End Sub
```

เราสามารถอ้างอิง Me.RecordsetClone มาพักไว้ที่ Recordset ของเราเมื่อไรก็ได้ เมื่อเราได้ RecordSet แล้ว เราสามารถทำการบน RecordSet ตัวได้ได้อิสระ เช่นเดียวกับ RecordSet ทั่วไปทุกประการ

การสร้าง WEB PAGE จาก RECORDSET

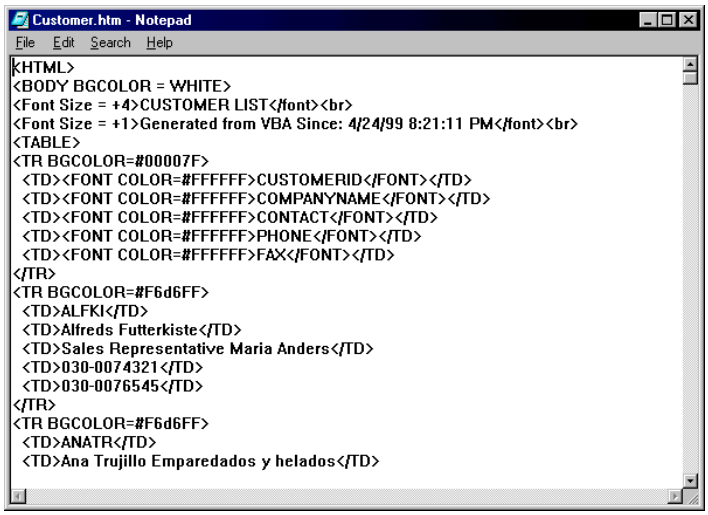
ตัวอย่างนี้เป็นการแสดงให้เห็นการใช้ RecordSet ในการจัดการที่ซับซ้อน เช่นการ ใช้ RecordSet พิมพ์ Text ไฟล์ที่มีโครงสร้างแปลก เป็นต้น ในส่วนนี้เราจะทดลองใช้ RecordSet ทำการสร้าง WEB page โดยใช้ Table Customer เป็นแหล่งข้อมูล

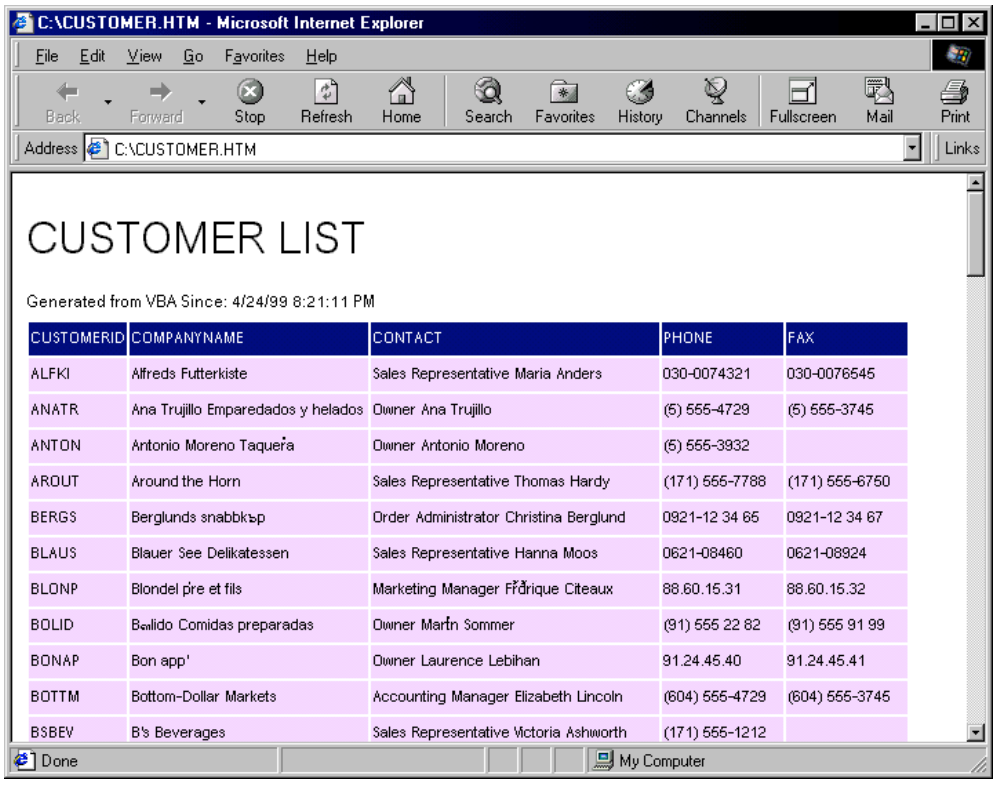
```
Sub WebGen()

    Dim filex
    Dim rst As Recordset

    Set rst = dbEngine(0)(0).OpenRecordset("Customers")

    filex = FreeFile

    Open "C:\CUSTOMER.HTM" For Output As filex


    Print #filex, "<HTML>"
    Print #filex, "<BODY BGCOLOR = WHITE>"
    Print #filex, "<Font Size = +4>CUSTOMER LIST</font><br>"
    Print #filex, "<Font Size = +1>Generate from VBA Since: " & Now() & "</font>
<br>"
    Print #filex, "<TABLE>"

    Print #filex, "<TR BGCOLOR=#00007F>"
    Print #filex, "  <TD><FONT COLOR=#FFFFFF>CUSTOMERID</FONT></TD>"
    Print #filex, "  <TD><FONT COLOR=#FFFFFF>COMPANYNAME</FONT></TD>"
    Print #filex, "  <TD><FONT COLOR=#FFFFFF>CONTACT</FONT></TD>"

    Print #filex, "  <TD><FONT COLOR=#FFFFFF>PHONE</FONT></TD>"
    Print #filex, "  <TD><FONT COLOR=#FFFFFF>FAX</FONT></TD>"
    Print #filex, "</TR>"


    Do Until rst.EOF

        Print #filex, "<TR BGCOLOR=#F6d6FF>"
        Print #filex, "  <TD>" & rst("CustomerID") & "</TD>"
        Print #filex, "  <TD>" & rst("CompanyName") & "</TD>"
```

```
        Print #filex, "  <TD>" & rst("ContactTitle") & " " & rst("ContactName") &
"</TD>"

        Print #filex, "  <TD>" & rst("Phone") & "</TD>"
        Print #filex, "  <TD>" & rst("Fax") & "</TD>"
        Print #filex, "</TR>"
        rst.MoveNext

    Loop
    Print #filex, "</TABLE>"
    Print #filex, "</HTML>"

    Set rst = Nothing
    Close


End Sub
```

หากดูโปรแกรมดังกล่าว จะพบว่าไม่มีอะไรที่ซับซ้อน เพียงแต่ใช้การ LOOP ของ RECORDSET ในการพิมพ์ข้อมูลใน TABLE ออกมาเป็นไฟล์ HTML เท่านั้น ความยุ่งยากจะอยู่ที่การจัด TAG HTML ให้ลงตัวมากกว่า

ไฟล์ที่ Gen ออกมา จะเป็น HTML ไฟล์ที่เป็นไปตาม Loop ที่เราสั่งพิมพ์ตามลำดับ และปิดท้ายด้วย TAG ตามปกติ





การทำงานกับ TableDefs

TableDefs เสมือนเป็นถังที่เก็บข้อมูลของ Table ในระบบทั้งหมด Object TableDefs นี้ หากเราใช้อ้างอิงโดยใส่ค่าลงในวงเล็บเหมือนกับตัวแปร Array จะหมายถึงการอ่านอ่านค่า TableDefs ของ Table ใด Table หนึ่งใน Database เลขที่นั้น จะเป็นลำดับที่ของ Table ใน Database ขณะเดียวกัน เราก็สามารถระบุเป็นชื่อ ตรงนี้จะหมายถึง Table นั้นตรงๆ เช่นตัวอย่างต่อไปนี้

```
Sub readOrderCount()

        Dim x
        x = dbEngine(0)(0).tableDefs("Orders").RecordCount
        MsgBox ("Total " & x & " order(s) now.")

End Sub
```

เราสามารถอ่านค่า Properties ที่เก็บไว้ใน TableDefs ได้ อย่างเช่นในตัวอย่างข้างต้นเป็นการอ่านค่า Properties RecordCount ของ Table ชื่อ Orders , ตัว TableDefs มี Properties อยู่หลายตัว ตัวที่น่าสนใจได้แก่

- **Count**

เป็นจำนวน Table ที่มีใน Database เป็น Properties โดยตรงของ TableDefs ไม่ต้องระบุว่าเป็น Table ใด Table หนึ่ง

```
Sub readTableCount()

        Dim x
        x = dbEngine(0)(0).tableDefs.Count
        MsgBox ("Total " & x & " Table.")

End Sub
```

- **Name**

เป็น Properties ของราย Table ใช่สำหรับอ่านชื่อ Table ที่อยู่ใน TableDefs ทั้งหมด

```
Sub TableDefsWork()
    Dim i

    i = dbEngine(0)(0).tableDefs.Count
    For i = 0 To i - 1
        Debug.Print dbEngine(0)(0).tableDefs(i).Name

    Next i
End Sub
```

หากสำรวจจำนวน Table กับรายชื่อ Table ที่ Print ออกมาที่ Debug Windows จะพบว่า มี Table ที่เกินมาจากที่เราเห็นใน Table ทั้งหมด เนื่องจาก Access เองก็ใช้ Table ของมันเอง ในการเก็บข้อมูลเกี่ยวกับ Object ต่างๆในระบบด้วยเช่นกัน แต่ Table พวกนี้ จะไม่เห็นโดย Database Windows ธรรมดา เราจะทราบได้อย่างไรว่า Table ไหน เป็น Table ของ ระบบ ? เราใช้ Properties Attribute

- **Attributes**

ใช้ในการตรจสอบว่า Table ดังกล่าวเป็น System Table หรือไม่ และเป็น Table ประเภทไหน ถ้า Attributes เป็น 0 แสดงว่าเป็น Table ทั่วไป

```
Sub TableDefsWork()
    Dim i

    i = dbEngine(0)(0).tableDefs.Count
    For i = 0 To i - 1
        If dbEngine(0)(0).tableDefs(i).Attributes = 0 Then
            Debug.Print dbEngine(0)(0).tableDefs(i).Name
        End If
    Next i
End Sub
```

- **RecordCount**

ไว้ใช้ในการอ่านจำนวน Record ที่มีใน Table ใช้กับ Database ที่ MDB เท่านั้น

- **DateCreated**

วันเวลาที่สร้าง Table

- **LastUpdated**

วันเวลาที่ทำการ Update ล่าสุด

การสร้าง Table

เราใช้ Method บน TableDef ในการสร้าง Table ใหม่โดยใช้โปรแกรมเข้าไปสร้าง Table ขึ้นเอง เราต้องใช้การสร้าง Table ด้วย VB บ่อยครั้ง โดยเฉพาะการสร้าง Table สำรองเป็นต้น

การสร้าง Table จะใช้การ Append ซึ่งเป็น Method หนึ่ง ทำโดยการ Append เข้าไปใน TableDefs โดย Table ที่สร้างขึ้นต้องมีอย่างน้อย 1 Field

เริ่มต้นจากการสร้าง Object ที่เป็น TableDef สำหรับเก็บ TableDef ที่กำลังสร้างขึ้น ระว่างที่ทำการสร้าง Table มีการ Add Field เข้าไปใน Table ทีละ Field หรือ จะทำการ AddIndex ไปพร้อมๆกันก็ได้

Field หรือ Index ที่ทำการ Add จะเข้าไปใน TableDef สำรอง จนกว่าเราจะใช้คำสั่ง Append เพื่อดึง TableDef ที่เราทำพักไว้ เข้าสู่ TableDefs รวมในตัวใหญ่อีกต่อหนึ่ง

ทดลองดูการสร้าง Table ชื่อ MySample โดยมี Field Test1 แบบ Text ขนาด 10 byte

```
Sub CreateTableTest()

    Dim tbx As TableDef

    Set tbx = dbEngine(0)(0).CreateTableDef("MySample")
    tbx.Fields.Append tbx.CreateField("Test1", dbText, 10)
    '—add another field
    '
    '
    dbEngine(0)(0).tableDefs.Append tbx


End Sub
```

ระหว่างที่เป็น tbx นั้น ยังไม่ปรากฏ Table ดังกล่าวใน List Table แต่อย่างใด จนกว่าจะมีการ Add เข้าที่ คำสั่ง Append ท้ายสุด

ในการ Add Field เข้าใน Table ในช่วงการสร้าง Table เราใช้คำสั่ง CreateField โดยระบุชื่อ Field ระบุแบบของ Field ซึ่งเป็นค่า Constant ตามตาราง และ ขนาดของ Field สำหรับ Field ที่ต้องระบุขนาด หาก Field นั้น ไม่จำเป็นต้องระบุขนาด ให้เว้นว่าง เช่นกรณี Add Field ที่เป็นตัวเลข Byte

```
tbx.Fields.Append tbx.CreateField("Test2", dbByte)
```

Constant	Description
dbBigInt	Big Integer
dbBinary	Binary
dbBoolean	Boolean
dbByte	Byte
dbChar	Char
dbCurrency	Currency
dbDate	Date/Time
dbDecimal	Decimal
dbDouble	Double
dbFloat	Float
dbGUID	GUID
dbInteger	Integer

```
dbLong                          Long
dbLongBinary                    Long Binary (OLE Object)
dbMemo                          Memo
dbNumeric                       Numeric
dbSingle                        Single
dbText                          Text
dbTime                          Time
dbTimeStamp                     Time Stamp
dbVarBinary                     VarBinary
```

การลบ Table

การลบ Table จะใช้ Method Delete บน TableDefs แล้วตามด้วยชื่อ Table ที่ต้องการลบ เช่น

```
dbEngine(0)(0).tableDefs.Delete "MySample"
```

การเพิ่ม Field ลงใน Table

เมื่อเราใช้ Table ไปถึง ณ จุดๆหนึ่ง เราอาจต้องการเพิ่ม Field เข้าไปใน Table แต่ต้องการเพิ่มด้วยโปรแกรม เช่น โปรแกรมที่เรา Install ให้ลูกค้าไปแล้ว แล้วต้องการ Upgrade Version แล้วต้องการเพิ่ม Field เป็นต้น

การ Add Field ทำเหมือนกับการ Add Field ลงใน Table ขณะที่สร้าง Table ใหม่ แต่ตัว TableDef ที่ใช้ในการอ้างอิง จะไม่ได้มากจากการ CreateTableDef แต่เป็นการเปิด TableDef ธรรมดา และยังคงใช้คำสั่ง Append Field เข้าไปใน Object Fields ของ Table เช่นเดิม

```
Sub AddField()

    Dim tbx As TableDef
    Dim i As Byte

    Set tbx = dbEngine(0)(0).tableDefs("MySample")
    For i = 1 To 7
        tbx.Fields.Append tbx.CreateField("TmpField" & i, dbByte)
    Next i

End Sub
```

ตัวอย่างข้างต้นเป็นการ AddField เขาไปใน Table MySample จำนวน 7 Field มีชื่อ tmpField1-7 ตามลำดับ โดยเป็น Field ประเภท dbByte

การลบ Field จาก Table

การลบ Field จะคล้ายกับการลบ Table คือใช้ Method Delete แล้วตามด้วยชื่อ Table ตามตัวอย่างข้างล่าง ซึ่งใช้ในการลบ Field ที่เราเพิ่งสร้างจาก ตัวอย่างของการสร้าง Field ก่อนหน้า

```
Sub DeleteField()
    Dim tbx As TableDef
    Dim i As Byte

    Set tbx = dbEngine(0)(0).tableDefs("MySample")
    For i = 1 To 7

        tbx.Fields.Delete "tmpField" & i
```

```
    Next i

End Sub
```

การอ่านชื่อ Field จาก Table

เราสามารถอ่านชื่อ Field ของ Table จากการอ่านค่า Properties ของ Field ซึ่งเป็น Object ใต้ Table ออกมาได้เช่นกัน โดยอ้างอิง Properties Name ของ Field ตามตัวอย่าง

```
Sub ReadTableField()

    Dim tbx As TableDef
    Dim tblRun As Integer
    Dim FldRun As Integer
    Dim rst As Recordset

    Set tbx = dbEngine(0)(0).CreateTableDef("TableInfo")

    tbx.Fields.Append tbx.CreateField("TableName", dbText, 100)
    tbx.Fields.Append tbx.CreateField("ColumnName", dbText, 100)
    dbEngine(0)(0).tableDefs.Append tbx

    Set rst = dbEngine(0)(0).OpenRecordset("TableInfo", dbOpenDynaset,
dbAppendOnly)


    For tblRun = 0 To dbEngine(0)(0).tableDefs.Count - 1
        For FldRun = 0 To dbEngine(0)(0).tableDefs(tblRun).Fields.Count - 1
            rst.AddNew

            rst("TableName") = dbEngine(0)(0).tableDefs(tblRun).Name
            rst("ColumnName") = dbEngine(0)(0).tableDefs(tblRun).Fields
(FldRun).Name

            rst.Update


        Next FldRun

    Next tblRun
    Set Rst = Nothing


End Sub
```

ตัวอย่างข้างต้นเป็นการ อ่านชื่อ Field และชื่อ Table จากฐานข้อมูล แล้วไปสร้างเป็น Table ใหม่ ชื่อ TableInfo เพื่อเก็บรายชื่อ Field และ Table ในระบบ

กำหนด tblRun และ fldRun เป็นตัวแปร ในการอ้างอิง Table, Field ในแต่ละตัว และทำการอ้างอิงชื่อ Name ไปทีละ Field ซึ่งเราสามารถอ่านค่าอื่นจาก Field ได้อีกเช่น ประเภทของ Field ขนาดของField เป็นต้น ขอให้ดู Help ของ Field Properties อีกครั้งหนึ่ง โดยส่วนใหญ่ เราใช้ไม่บ่อยนัก

การเปลี่ยนชื่อ Field

เราสามารถเปลี่ยน ชื่อ Field ได้โดยตรง โดยการแทนที่ Properties Name ตามปกติ ตามตัวอย่าง

```
Sub ChangeFieldName()
    Dim tbx As TableDef

    Set tbx = dbEngine(0)(0).tableDefs("TableInfo")
    tbx.Fields("TableName").Name = "TblName"


End Sub
```

การปรับเปลี่ยนชนิด Field และ ขนาด

เราไม่สามารถปรับเปลี่ยน Filed ทั้งขนาด และ ประเภทของ Field ได้โดยตรง จะต้องใช้วิธีสร้าง Field ใหม่ แล้วทำการ Copy ข้อมูลจาก Field เก่ามาไว้ที่ Field ใหม่ แล้วทำการลบ Field เก่าทิ้ง

ขณะที่เราใช้ Access และทำการปรับขนาด Field หรือ เปลี่ยนประเภทของ Field ก็ดี Access จะเป็นคนทำกระบวนการดังกล่าวให้เราแทน การเขียนโปรแกรมเข้าไปเปลี่ยน Field เองค่อนข้างยุ่งยาก ทดลองดูตามตัวอย่างต่อไปนี้ ซึ่งเป็นการปรับขนาด Field ColumnName จาก 100 เป็น 90

```
Sub ChangeFiledSize()

    Dim tbx As TableDef
    Dim rst As Recordset

    Set tbx = dbEngine(0)(0).tableDefs("TableInfo")
    tbx.Fields("ColumnName").Name = "WillDelete"
    tbx.Fields.Append tbx.CreateField("ColumnName", dbText, 90)

    Set rst = dbEngine(0)(0).OpenRecordset("TableInfo")

    Do Until rst.EOF
        rst.Edit
            rst("ColumnName") = rst("WillDelete")
        rst.Update

        rst.MoveNext
    Loop

    Set rst = Nothing

    tbx.Fields.Delete "WillDelete"

End Sub
```

เริ่มต้นจากการ Rename ชื่อ Field เดิมเสียก่อน แล้วทำการ สร้าง Field ใหม่ที่ต้องการ จากนั้นใช้ RecordSet ในการอ่านข้อมูลเพื่อโอนถ่ายข้อมูลข้าม Field ระหว่าง ColumnName กับ WillDelete เมื่อเสร็จสิ้น ก็ทำการลบ Field เก่าทิ้ง (WillDelete)

การโอนถ่ายข้อมูลควรใช้ SQL COMMAND มากกว่า ซึ่ง จะได้พบในบทถัดไป โดยเราสามารถแทน ชุดคำสั่ง ตั้งแต่การเปิด RecordSet เป็นคำสั่ง Update Query เพียงตัวเดียวตามตัวอย่างนี้

```
Sub ChangeFiledSize2()

    Dim tbx As TableDef


    Set tbx = dbEngine(0)(0).tableDefs("TableInfo")
    tbx.Fields("ColumnName").Name = "WillDelete"
    tbx.Fields.Append tbx.CreateField("ColumnName", dbText, 91)
    dbEngine(0)(0).Execute "Update TableInfo set ColumnName = WillDelete"
    tbx.Fields.Delete "WillDelete"

End Sub
```

การจัดทำ Index

การ Add Index เข้าใน Table ใช้กลไกการ Append เช่นกัน แต่เนื่องจาก 1 Index อาจมีได้ หลาย Field ที่มาทำ Index ร่วมกัน จึงจำเป็นต้องทำการเตรียม Index ก่อนนำ Index เข้าไปใน Table

เราควรสร้างตัวแปร Index ที่เก็บ Field ที่ประกอบเป็น Index ก่อนนำเข้าไปใน Table ตัวอย่างการทำ Index บน Table Products ที่ Field ProductID และ UnitPrice แสดงไว้ตามตัวอย่างคือ

```
Sub AddIndex()
    Dim tbx As TableDef
    Dim idx As Index

    Set tbx = dbEngine(0)(0).tableDefs("Products")
    Set idx = tbx.CreateIndex("SampleIDX")
    idx.Fields.Append idx.CreateField("ProductID")
    idx.Fields.Append idx.CreateField("UnitPrice")
    tbx.Indexes.Append idx


End Sub
```

กรณีที่ต้องการสร้าง PrimaryKey เราจะ Set ที่ Properties ของ Index ตัวนั้น ทดลองดูตัวอย่างการ สร้าง Table MySample พร้อมกับการสร้าง PrimaryKey

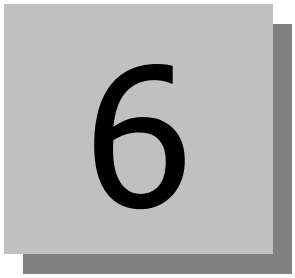
```
Sub CreateTableTest()

    Dim tbx As TableDef
    Dim idx As index


    Set tbx = dbEngine(0)(0).CreateTableDef("MySample")
    tbx.Fields.Append tbx.CreateField("ID", dbLong)
    tbx.Fields.Append tbx.CreateField("Test1", dbText)
    dbEngine(0)(0).tableDefs.Append tbx

    Set idx = tbx.CreateIndex("PK1")
    idx.Fields.Append idx.CreateField("ID")
    idx.Primary = True

    tbx.Indexes.Append idx


End Sub
```

การลบ Index

เราลบ Index ที่สร้างขึ้น โดยใช้คำสั่ง Delete แล้วตามด้วยชื่อ Index โดยกระทำกับ Object ที่เป็น Index สมมุติว่าเราต้องการลบ Index ที่เราทดลองสร้างไว้ใน Table Product ในตัวอย่างก่อนหน้า

```
Sub DropIndex()
    dbEngine(0)(0).tableDefs("Products").Indexes.Delete "SampleIDX"
End Sub
```

CHAPTER 6
SQL COMMAND



ในบทนี้เราจะกล่าวถึง ภาษา SQL ซึ่งเป็นภาษามาตราฐานในการสั่งงานระบบฐานข้อมูล ตัวภาษา SQL บน Access เป็น ภาษา SQL ที่สามารถใช้เรียนรู้ เพื่อปูพื้นฐานไปสู่การใช้ภาษา SQL ใน Database Server ตัวใหญ่ต่อไปในอนาคต

เราใช้ SQL กับ Access ในการทำงานร่วมกับ RecordSet การทำการประมวลผลข้อมูล ด้วยภาษา SQL ในลักษณะของ Action Query และเรายังสามารถ ใช้ภาษา SQL ในการสร้าง Table แก้ไข Table ได้อีกเช่นกัน การเรียนรู้ในบทนี้จะต้องนำเอาการใช้ Database Object ในบทที่ผ่านมาใช้ประกอบการเรียนรู้ด้วย

CONTENT

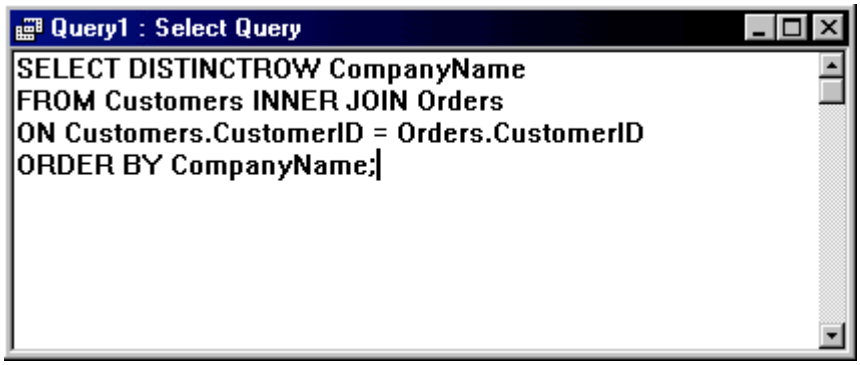
- การใช้ และทดสอบ SQL บน ACCESS
- ลักษณะของ ACCESS SQL
- SELECT QUERY
- AGGREGATE QUERY
- ACTION QUERY
- UNION QUERY
- การ SQL บน DatabaseObject

การใช้ และทดสอบ SQL บน ACCESS

ก่อนที่เราจะเริ่มเรียนรู้ตัวภาษา SQL เรามาดูเครื่องมือที่จะใช้ในการทดสอบ ภาษา SQL ที่เราจะใช้เรียนรู้ในบทนี้เสียก่อน

การใช้ QUERY WINDOWS

ใช้ QUERY ที่มีในแบบ DBGRID แต่ ย้ายไปใช้มี่หน้า SQL แทน เมื่อทำการใส่คำสั่ง SQL แล้วก็กด RUN ที่ปุ่ม ! ดังกล่าว Query ที่เราทำการบันทึกไว้ ก็จะทำงานทันที



เรายังสามารถ Switch ไปมาระหว่าง SQL และ Design View ได้ ในกรณีที่เป็น SQL SELECT แบบธรรมดา ซึ่งสามารถแสดงด้วยหน้า Design Query ของ Access และในขณะเดียวกัน เรายังกลับมายังหน้า Design View ในการทดลอง Query บางอย่าง เพื่อทดสอบ และดูค่า SQL ที่แทนความหมายของการ Design Query ของเรา ตรงนี้จะช่วยให้เราเข้าใจตัวภาษา SQL ได้ดีขึ้น

ลักษณะของ ACCESS SQL

SQL ของ ACCESS เป็น SQL ที่จัดเป็นมาตราฐาน ANSI-89 Level 1 ภาษา SQL ที่ใช้บน ACCESS กับ ANSI มีข้อแตกต่างกัน ในบางประการ ได้แก่

ข้อแตกต่างที่สำคัญ

- Access SQL มี Reserve Word ที่ไม่ตรงกับใน ANSI SQL ซึ่งรวมไปถึง Reserve Word ของขนาด Field ที่ใช้ในการออกแบบ Table ขอให้ตรวจดู Reserve Word ทั้ง 2 ได้จาก Help
- การใช้ Between...And ที่ ACCESS SQL อนุญาติให้ค่าแรก มากกว่าค่า 2 ได้ โดยที่ ANSI ไม่อนุญาติ
- การใช้ WILDCARD ใน LIKE ของ 2 ตัว ไม่เหมือนกัน

การแทนที่การค้นหา	ACCESS SQL	ANSI SQL
ตัวอักษรอิสระ 1 ตัว	?	_ (underscore)
ตัวอักษรอิสระกี่ตัวก็ได้	*	%

- ACCESS SQL อนุญาติให้มีการใช้ SQL ที่ไม้เหมาะสมได้โดยไม่เกิด Error กล่าว คือไม่ Strict เหมือน ANSI SQL
- มีการใช้ Expression หรือ Function ที่มีใน Access มาประกอบเป็น SQL ได้

สิ่งที่ ACCESS SQL มีมากกว่า

- คำสั่ง TRANSFORM ซึ่งสามารถใช้ในการสร้าง CROSSTAB Query
- มี Aggregate Function ที่ ANSI ไม่มีคือ StDev StDevP,Var, VarP สำหรับใช้ในการหาค่าเบี่ยงเบนมาตราฐาน และค่าความแปรปรวน
- สามารถกำหนด PARAMETERS กรณีที่ใช้กับ PARAMETERS QUERY

สิ่งที่ ACCESS SQL ทำไม่ได้เมื่อเทียบกับ ANSI SQL

- ไม่สามารถใช้คำสั่งเกี่ยวกับระบบความปลอดภัยได้ เช่น COMMIT, GRANT, LOCK.
- ไม่สามารถทำ DISTINCT บน aggregate function ได้ เช่นการใช้ SUM(DISTINCT columnname).
- ไม่สามารถใช้ LIMIT TO nn ROWS เพื่อระบุจำนวน RecordSet ขณะ Select ได้ จะต้องใช้ Where เท่านั้น หรือกำหนด SCOPE ของ QUERY แทน

SELECT QUERY

Query ที่มีการใช้มากที่สุดคือ QUERY แบบ SELECT ใช้ในการเรียกข้อมูลจาก Table ที่กำหนด สามารถระบุ Field ที่ต้องการเรียกจากแต่ละ Table ได้โดยไม่จำเป็นต้องเรียกทุก Filed เราสามารถกำหนดเงื่อนไขการ WHERE และ ORDER BY สำหรับการ SORT ไปพร้อมกันในคำสั่ง SELECT ด้วย มีรูปแบบคำสั่งคือ

```
SELECT [predicate] { * | table.* | [table.]field1 [AS alias1] [, [table.]field2
[AS alias2] [, ...]]}
FROM tableexpression  [, ...] [IN externaldatabase]
[WHERE... ]
[ORDER BY... ]
```

องค์ประกอบในคำสั่ง SELECT ประกอบด้วยส่วนต่างๆดังนี้

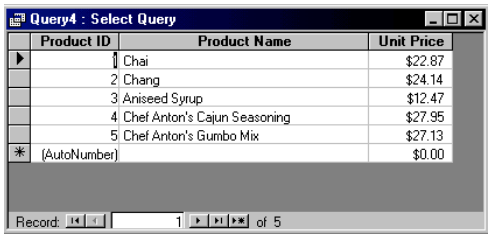
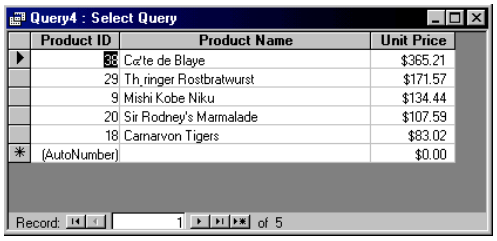
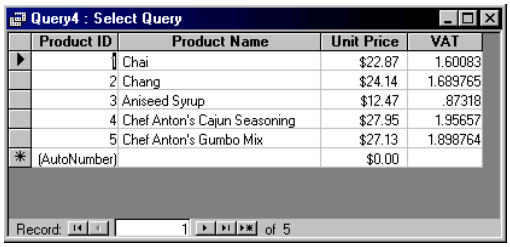
ส่วนประกอบ	คำอธิบาย	
predicate	เป็นส่วนที่กำหนดจำนวนข้อมูลที่จะเป็นผลของการ SELECT สามารถแบ่งได้เป็น	
	ALL	เป็นค่า Default หากเราไม่ระบุอะไรเลย หมายถึงหยิบทุก Record ที่ทำการ Select ได้ออกมาแสดงทั้งหมด
	DISTINCT	บังคับให้แสดงข้อมูลโดยให้ตัด Record ที่มีการซ้ำกันในทุก Column ออกไป เพื่อให้ได้ ข้อมูลทุก Row ที่ไม่ซ้ำกัน
	DISTINCTROW	คล้ายกับ DISTINT แต่การตรวจสอบการซ้ำ จะดูที่ว่า Primary Key ของ Table นั้นซ้ำหรือไม่ หรืออาจเรียกว่าเป็นการตัด Record ที่ซ้ำกันในส่วนของ Record ไม่ได้ดูที่ Column ซ้ำแล้วตัดออกเท่านั้น ต้องดูว่า Primary Key ซ้ำด้วยหรือไม่ ตรงนี้จะเห็นผลเมื่อเป็นการ Select แบบ Join Table ที่มีการใช้หลาย Table พร้อมกัน
	TOP	เป็นการระบุเป็นจำนวน Record ที่ต้องการ SQL จะเลือกขอมูลตามจำนวนที่กำหนด เช่น ระบุ TOP 5 จะได้จำนวนข้อมูลจำนวน 5 Record เป็นต้น

*	เราใช้ * ในกรณีที่เราต้องการทุก Field หาก เราไม่ระบุชื่อ Table นำหน้า * จะหมายถึงทุกๆ Filed ที่ทำการ Select หากมีการใช้ Table มากกว่า 1 Table ใน From จะได้ทุก Field จากทุก Table แต่ถ้าเราระบุ ชื่อ Table ก่อนจะบังคับให้เลือกทุก Field เฉพาะของ Table นั้น
	`Select *`
	`from Orders inner join [Order Details] on Orders.OrderID = [Order Details].OrderID`
	ตรงนี้จะได้ทุก Field จากทั้ง 2 Table
	`Select Orders.CustomerID, [Order Details].*`
	`from Orders inner join [Order Details] on Orders.OrderID = [Order Details].OrderID`
	แต่สำหรับ Query นี้ จะหยิบเฉพาะ OrderDetails ทุก Field แต่ Orders หยิบเฉพาะ Customer	

table	ระบุชื่อ Table ที่เป็นเจ้าของ Field ตัวที่เราต้องการ Select แล้วตามด้วย " . " สำหรับกรณีที่ในการ Select ข้อมูลมากจากข้อมูลหลายๆ Table และในหลายๆ Table นั้นมีชื่อ Field ที่ซ้ำกันอยู่ เราต้องระบุชื่อ Table นำหน้าเสมอ เพราะ SQL คงไม่ทราบว่าจะเลือกชื่อ Field นั้นจาก Table ใด	

ส่วนประกอบ	คำอธิบาย
	หากชื่อ Table นั้นมี Space อยู่ภายใน จะต้องใช้เครื่องหมาย [ ] คลุมชื่อ Table หัวท้าย เพื่อให้ SQL ทราบว่าชื่อ Table ยังไม่จบในระหว่างที่พบ Space ที่ตามมาในระหว่างชื่อ Table หากเราระบุ Alias ของชื่อ Table ใหม่แล้ว การอ้างอิงอิงชื่อ Table สำหรับ Field จะต้องใช้ตาม Alias ใหม่ที่กำหนด
field1, field2	ระบุชื่อ Field ที่ต้องการ อาจระบุเป็น Expression ได้ แต่ควรใช้ Build-in Function ที่ไม่เป็น Domain Aggregate Function (เช่นพวก Dlookup) หากชื่อ Field นั้นมี Space อยู่ภายใน จะต้องใช้เครื่องหมาย [ ] คลุมชื่อ Table หัวท้าย เพื่อให้ SQL ทราบว่าชื่อ Field ยังไม่จบในระหว่างที่พบ Space ที่ตามมาในระหว่างชื่อ Field
alias1, alias2	เราใช้ในการแทนชื่อ Field ที่เรา Select มา เป็นการกำหนดชื่อ Field ใหม่หลังจากที่ Select แล้ว เช่น `Select CustomerID as CusID from CustomerS`
Tableexpression	เป็นส่วนที่อยู่หลัง FROM เป็นตัวกำหนด Table หรือ Query ที่นำมาใช้เป็นแหล่งข้อมูลในการ Select ระบุเป็น Table แล้วขั้นด้วย Comma หรือระบุเป็น Join Table ก็ได้ ให้ดูส่วนถัดไป
Externaldatabase	เป็นการระบุชื่อไฟล์ กรณีที่การ Select นี้มากจาก Table ที่อยู่คนละไฟล์กับที่เราทำงานอยู่ เช่น `SELECT * FROM TmpDB IN "C:\TMPDBPATH\TMP.MDB"`
Where	ระบุเงื่อนไขการหยิบข้อมูล ให้ดูเนื้อหาถัดไป
Order By	ระบุการเรียงตัวของข้อมูลที่ Select ได้ ระบุเป็นชื่อ Field มี 2 เงื่อนไขคือ ASC สั่งให้ SORT จากน้อยไปหามาก หากเราไม่ระบุอะไรในการ Sort โดยระบุชื่อ Field ที่ Sort เท่านั้น ค่า ASC นี้จะเป็นค่า Default DESC สั่งให้ Sort จากมากไปหาน้อย กรณีที่ต้องการ Sort มากกว่า 1 Field ให้ใช้ Comma ขั้นแต่ Filed ที่ Sort เช่น `Select * from Products Order By UnitPrice Desc, ProductID Desc`

ทดลองดูตัวอย่างการ SELECT ข้อมูลต่อไปนี้

วิธีการ	การ SELECT
SELECT ข้อมูลโดยกำหนด TOP	`select top 5 ProductId, ProductName, UnitPrice from products` ได้ผลเป็นข้อมูล Product 5 Record 
SELECT ข้อมูลโดยกำหนด TOP แล้วระบุการ SORT	`select top 5 ProductId, ProductName, UnitPrice from products order by UnitPrice desc` ได้ผลเป็นข้อมูล Product ที่มีราคาสูงสุด 5 Record แรก เรามักใช้ Top คู่กับการ Sort ในการค้นหาข้อมูลตามค่าสูงสุด ต่ำสุด ตามจำนวนที่กำหนด 
SELECT โดยใช้ Expression เข้าไปร่วมด้วย	`select top 5 ProductId, ProductName, UnitPrice, UnitPrice *.07 as VAT from products` ได้ผลเป็นข้อมูล Product ที่มีราคาสูงสุด 5 Record พร้อมกับคำนวณ VAT 7 % และตั้งชื่อ Field เป็น VAT 

Query4 : Select Query

Product ID	Product Name	Unit Price
1	Chai	$22.87
2	Chang	$24.14
3	Aniseed Syrup	$12.47
4	Chef Anton's Cajun Seasoning	$27.95
5	Chef Anton's Gumbo Mix	$27.13
(AutoNumber)		$0.00

Record: 1 of 5

Query4 : Select Query

Product ID	Product Name	Unit Price
38	Cœte de Blaye	$365.21
29	Th,ringer Rostbratwurst	$171.57
9	Mishi Kobe Niku	$134.44
20	Sir Rodney's Marmalade	$107.59
18	Carnarvon Tigers	$83.02
(AutoNumber)		$0.00

Record: 1 of 5

Query4 : Select Query

Product ID	Product Name	Unit Price	VAT
1	Chai	$22.87	1.60083
2	Chang	$24.14	1.689765
3	Aniseed Syrup	$12.47	.87318
4	Chef Anton's Cajun Seasoning	$27.95	1.95657
5	Chef Anton's Gumbo Mix	$27.13	1.898764
(AutoNumber)		$0.00	

Record: 1 of 5

การระบุ Alias ของ TABLE

เราสามารถกำหนด Alias ของชื่อ Table ได้เพื่อให้การเขียน SQL สะดวกขึ้น ได้ง่าย **โดยระบุชื่อ Alias ตามหลังชื่อ Table โดยไม่ต้องมีคำว่า As** เหมือนกับการกำหนดใน Alias ใน Field ตัวอย่างการใช้ Alias ของ TABLE

```
Select P.* from ProductName P
```

กำหนด P เป็น Alias ของ Table Product จากนั้นในการอ้างอิง Field เราใช้ P.xxxxxx แทนการใช้ชื่อ Table เต็ม

```
select D.*, O.CustomerID
from   Orders O inner join [Order Details] D
          on O.OrderID = D.OrderID
```

ตัวอย่างข้างต้นใช้ D แทน OrderDetails และ O แทน Orders เพื่อให้การบันทึก SQL ง่ายขึ้น

การใช้ INNER JOIN

สำหรับการเชื่อมต่อ TABLE หลายๆ TABLE เข้าด้วยกัน เราใช้คำสั่ง INNER JOIN ในการเชื่อมโยงแต่ละคู่ของ Table โดยมีความหมายว่าการเชื่อมโยงจากทั้ง 2 Table จะต้องมี Field ที่ตรงกันเท่านั้น

การ Join กันของ Table 1 คู่จะได้ผลเสมือนเป็น Table อีก Table หนึ่ง จากเราสามารถ Join คู่ Table ที่ Join แล้วต่อไปยังตัว Table ใหม่ หรือ คู่ Table ใหม่อีกก็ได้

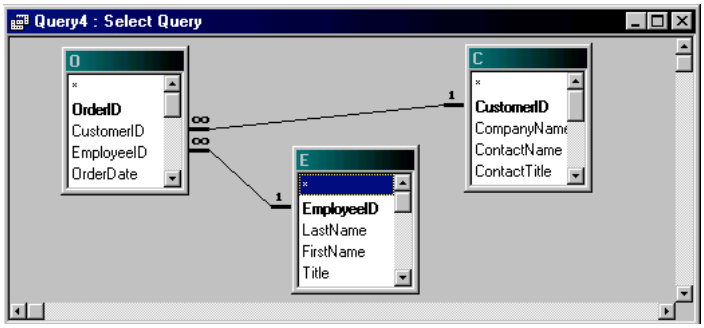
ตัวอย่างการ Join Orders กับ Customers

```
select     C.ContactName, O.OrderID
from       Orders O inner join Customers C on O.CustomerID = C.CustomerID
```

จากตัวอย่างด้านบน หากเราต้องการ Join Table Employees เข้าไปร่วมด้วยกับชุด Join ดังกล่าว ให้เราคิดว่า ชุด Join ดังกล่าวเป็น 1 Table แล้วใส่วงเล็บคลุมไว้ จากนั้นใช้การ Join ตามปกติ จะได้เป็น

```
select     C.ContactName,
           O.OrderID,
           E.FirstName & '  '  & E.LastName as EmpName
from       (Orders O  inner join Customers C on O.CustomerID = C.CustomerID)
                      inner join Employees E on E.EmployeeID = O.EmployeeID
```

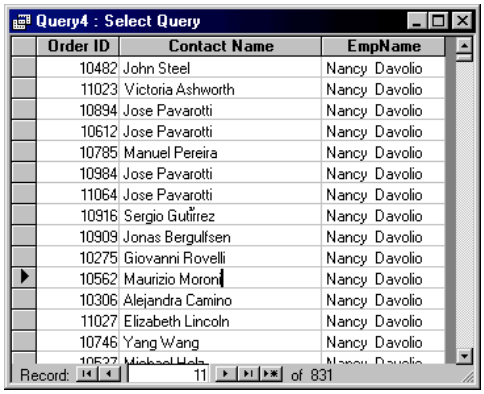
การกำหนด Inner join สามารถดูผลการเชื่อมเส้น Relation ที่ Join ได้จากปุ่ม Design โดยทดลองเรียกดูได้ ผู้เขียนแนะนำให้ใช้หน้า Design ของ QUERY ช่วยในการทำความเข้าใจการใช้ Join Query เพื่อลดความซับซ้อนใน



การเรียนรู้การเชื่อมโยงการ Join ด้วย SQL COMMAND

การทำงานของ Join Table

การ Join Table จะเกิดจากการนำ Table ที่ต้องการ Join มาสร้างเป็นตารางข้อมูลรวมของ Table ที่มา Join กัน แล้วเลือกเฉพาะชุดข้อมูลที่เงื่อนไขตรงกับที่ระบุใน ON



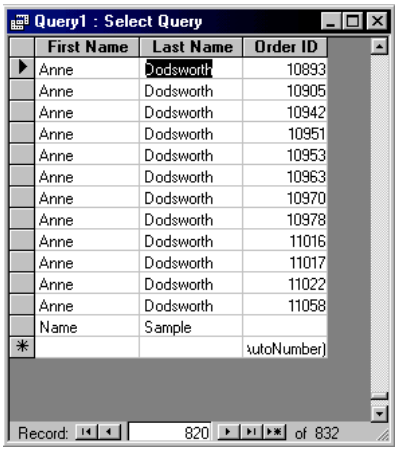
ถ้า Table Order มี Record อยู่จำนวน 1000 Record และ Customer มีอยู่ 100 Record หากเราเอา 2 Table นี้มา Join กัน จะเกิดผลที่เป็นไปได้ทันที 1000 X 100 จากนั้น SQL จะลดจำนวนโดยการใช้การ Where ตามเงื่อนที่ระบุใน ON

หากเราเลือก Table 2 Table มาไว้ใน From โดยไม่กำหนดการ Join เลย จะได้ผลจำนวน Record เป็นผลคูณดังกล่าว เช่น การสั่ง SQL COMMAND ต่อไปนี้

```
select * from  orders,customers
```

การใช้ OUTER JOIN

เป็นการ Join ของ Table ที่ระบุให้มีการเลือกข้อมูลจากด้านใดด้านหนึ่งทั้งหมด ถึงแม้ว่าเงื่อนไขการตรวจสอบ ระหว่าง Field ที่ระบุบน ON อาจไม่พบข้อมูลที่ตรงกันก็ได้



เช่นเราเชื่อม Table Employees กับ Orders เข้าด้วยกัน เราต้องการ Order ของทุกๆ Employees ถึงแม้ไม่มี Orders ใน Employees นั้นๆ ก็ยังคงให้แสดงชื่อ Employees นั้นด้วยเป็นเป็นต้น

เราจะต้องบอกให้ SQL ทราบว่าข้อมูลจาก Table ใดให้ถูกเลือกทั้งหมด โดยการการใช้คำสั่ง Left Join หรือ Right Join

ข้อกำหนดในการระบุคือ ถ้าเราระบุ Left Join แสดงว่า Table ที่อยู่ด้านซ้าย หรือ Table แรก ให้เรียกมาทั้งหมด ถ้าระบุ Right Join ให้เอา Table ที่อยู่ด้านขวามาทั้งหมด ตัวอย่างการเชื่อม Employees ด้วย Left Join ได้แสดงไว้แล้ว อย่าลืมทดลองกรอกชื่อ พนักงานใหม่ลงใน Table Employees เพื่อทำการทดสอบ

```
SELECT      E.FirstName,
            E.LastName,
            O.OrderID
From        Employees E LEFT JOIN  Orders O on E.EmployeeID = O.EmployeeID
```

การใช้ OUTER JOIN หลายชั้นๆ อาจทำให้ SQL ทำงานไม่ได้ เนื่องจากการ Conflict ของการ Join ที่ต้องการทั้งซ้ายและ ขวา ขอให้ระมัดระวังในการใช้งาน

ส่วนค่าที่ได้ในกรณีที่ อีกด้านหนึ่งไม่พบ Record เช่นกรณีที่ไม่พบ OrderID ของ Employees ค่าที่ว่างไว้ จะมีค่าเป็น NULL สามารถใช้ Function ISNULL ในการตรวจสอบค่าได้ หรือ Check ที่ค่าดังกล่าวว่าเป็น Null หรือไม่

ตัวอย่างการตรวจสอบค่า OrderID เพื่อดูว่า มี Record ใดทำการ Join แล้วไม่พบข้อมูล เช่น

```
SELECT      E.FirstName,
            E.LastName,
            O.OrderID
From        Employees E LEFT JOIN  Orders O on E.EmployeeID = O.EmployeeID
WHERE       ORDERID IS NULL
```

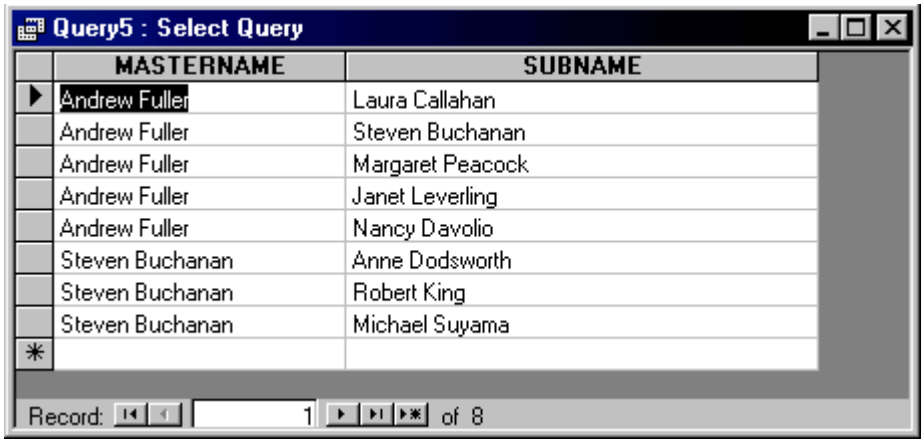
ตัวอย่างข้างต้น จะให้ผลเป็นรายชื่อพนักงานที่ไม่มี Order ปรากฏใน Table Orders

การใช้ SELF JOIN

เราใช้ SELF JOIN เมื่อมีการเชื่อมกันเองระหว่าง Table เดียวกัน เช่น TABLE EMPLOYEE มี Field ที่เป็น รหัสพนักงานอีก 1 Field หมายถึงรหัสพนักงานที่เป็น SUPERVISOR ของพนักงานคนนั้น (ชื่อ Field ReportsTo) จะพบว่า Employees เชื่อมกับ Employees กันเอง ที่ Field EmployeeID กับ ReportsTo

วิธีกำหนดลงใน Join ก็ทำเหมือนการทำการ Join ปกติ ใช้ได้ทั้ง Inner Join และ Outer Join แต่เราจะใช้ Alias ในการกำหนดชื่อ Table เพื่อแยกความแตกต่างของ 2 Table ที่เป็นตัวเดียวกัน ให้มีความแตกต่างจากกัน

```
SELECT      M.FirstName  & ' '  &  M.LastName as MASTERNAME,
            S.FirstName  & ' ' &   S.LastName as SUBNAME
FROM        Employees M inner join Employees  S on M.EmployeeID = S.ReportsTo
Order By    M.EmployeeID
```



จากตัวอย่างข้างต้น เราแทน Alias ของผู้บังคับบัญชา ด้วย M และผู้ใต้บังคับบัญชาด้วย S ซึ่งทั้ง 2 Table เชื่อมกันด้วย M.EmployeeID = S.ReportsTo จากนั้นเราสั่งให้มีการเรียงตามรหัสของผู้บังคับบัญชา

ในทางปฏิบัติ เราอาจใช้ Table ตัวเดียว หลายๆ ครั้งใน SQL COMMAND เดียวได้ ไม่จำกัดที่ 2 Table และต้องเชื่อมถึงกันบน Table เดียวกันเท่านั้น

การใช้ WHERE

การใช้ WHERE เพื่อระบุข้อมูลที่ต้องการค้นหา จะทำหลังชุดคำสั่ง FROM เป็นการระบุเงื่อนไขข้อมูลที่นำมาใช้ในการ WHERE

การ Where จะต้องอ้างอิง FILED ที่มีอยู่ใน TABLE ซึ่งได้ระบุไว้ใน FROM ไม่จำกัดว่าจะต้องเป็น FIELD ที่ SELECT ไว้เท่านั้น ข้อให้มี Field ดังกล่าวใน Table ที่ FROM ก็พอ

กรณีที่มีการใช้ Alias ของ Table แล้วจะต้องอ้างอิงชื่อ Alias ใหม่ของ Table แทนการอ้างอิงด้วยชื่อ Table เดิม ในกรณีที่จำเป็นต้องอ้างอิงชื่อ Field โดยมีการใช้ชื่อ Table นำหน้า

การ WHERE สามารถใช้การ AND , OR ในการเชื่อมต่อเงื่อนไขให้ซับซ้อนได้ ควรใช้วงเล็บปิดทุกชุดเงื่อนไขเพื่อเข้าใจง่าย

```
SELECT      PRODUCTS.UnitPrice, PRODUCTS.ProductName, PRODUCTS.SupplierID
FROM        Suppliers INNER JOIN PRODUCTS ON Suppliers.SupplierID =
PRODUCTS.SupplierID
WHERE (((PRODUCTS.UnitPrice)>20) AND ((Suppliers.CompanyName)="Tokyo Traders"));
```

AGGREGATE QUERY

AGGREGATE QUERY เป็น SELECT Query อีกชนิดหนึ่งที่ทำการสะสมค่าระหว่างที่ทำการ SELECT การสะสมค่าดังกล่าวสามารถเลือกได้ว่าต้องการสะสมโดยมีเงื่อนไขอย่างไร เราเรียกว่า FUNCTION ในการ AGGREGATE QUERY แบบนี้จะไม่สามารถทำการ Update ได้ เราใช้ในการ Select ขึ้นมาดูเพียงอย่างเดียว การที่เราใช้ Dsum, Dmax, Dcount, Dmin ก็เป็นการใช้ AGGREGATE QUERY โดยทางอ้อม เพราะ Access จะเป็นผู้เรียก Query ประเภทนี้แทนเรา

การกำหนด GROUP BY

เราจะใช้ Group By ไว้กำหนดชุดกลุ่มข้อมูลที่ต้องการทำการสะสม FIELD ที่กำหนดใน GROUP BY จะไม่ถูกทำการสะสมค่า คำสั่ง GROUP BY จะตามหลังคำสั่ง SELECT เดิม

เมื่อเราทำการระบุ Group By จะส่งผลต่อการสะสมข้อมูลของ Field ที่สั่งให้มีการสะสมข้อมูล โดย SQL จะทำการสะสมตามกลุ่ม Group By ที่ซ้ำๆกันเท่านั้น เช่น นำ Table Orders มานับ จำนวน Order หากเราระบุ GroupBy เป็น EmployeeID การนับจำนวนดังกล่าว ก็จะเป็นการนับในระหว่าง 1 EmployeeID ที่ซ้ำๆกัน เมื่อ Record ที่ทำการสะสม พบ EmployeeID รหัสใหม่ การนับนั้นก็เริ่มต้นนับใหม่ และเก็บค่าที่นับเดิมไว้ในบรรทัดของ EmployeeID เก่า

หากเราระบุการ Group By หลาย Field การนับ ก็จะนับเมื่อทุก Field ที่อยู่ใน Group By ตรงกัน หากขึ้นค่า Field ใหม่ ที่ Field ใด Field หนึ่ง Group By ก็ทำการนับใหม่ทันที

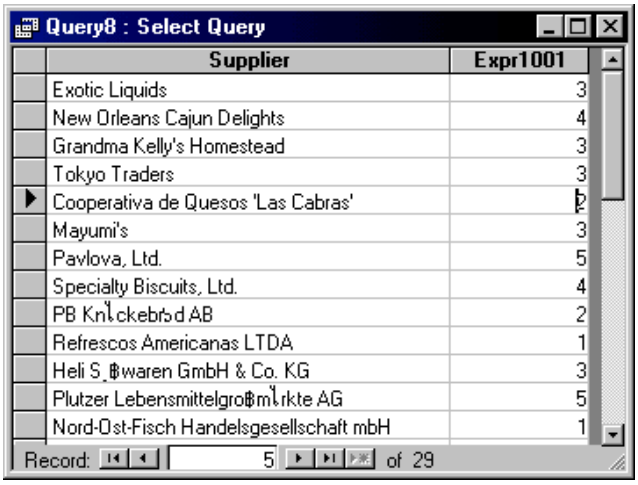
ถ้าเราเอา Primary Key ของ Table ที่ย่อยที่สุดใน Query มาทำ Group By ก็จะไม่เกิดประโยชน์อะไร เนื่องจากการ Group By Field ที่เป็น PrimaryKey ก็ย่อมไม่มีการซ้ำกันของข้อมูลอยู่แล้ว ก็จะได้ผลเหมือนการ Select ข้อมูลธรรมดา

ฟังก์ชั่นที่มีใช้ในการสะสมข้อมูลคือ

FUNCTION	คำอธิบาย

SUM	ใช้ในการรวมค่า Field ที่ต้องการ เราสามารถบรรจุ Function ลงในการ SUM ได้ เช่น
	
	`SELECT Sum(Products.UnitsInStock) AS [All Stock Unit],`
	`Sum(IIf([UnitPrice]>=20 And [UnitPrice]<100,`
	`[UnitsInStock],0)) AS [StockFor 20$-100$],`
	`Sum(IIf([UnitPrice]>100,[UnitsInStock],0)) AS [StockFor 100$ Up]`
	
	`FROM Products;`
	
	ตัวอย่างข้างต้นเป็นการนับ STOCK ทั้งหมด และนับ STOCK แยกตามลำดับราคา โดยใช้ IIF หากราคาอยู่ในช่วงที่กำหนด จะให้ค่า UnitsInStock ถ้าไม่ใช่ จะให้ค่า 0
	
	 

COUNT	ใช้สำหรับนับ Record ที่มีใน Database เราอาจใช้ Count(*) แทนการ Count(ชื่อ Field ได้) โดยการนับ Record จะแจกแจงตามจำนวนที่นับได้
	
	`SELECT COUNT(*)`
	`FROM PRODUCTS`
	
	แบบนี้ เท่ากับการนับจำนวน Record ของทั้ง Table
	
	`SELECT SUPPLIERID, COUNT(*)`
	`FROM PRODUCTS`
	`GROUP BY SUPPLIERID`
	
	แบบนี้มีการสั่ง Group By และใช้ SuplierID เป็นตัวแจกแจง ทำให้การนับ ของ Count(*)

FUNCTION	คำอธิบาย
	เป็นการนับจำนวน Record แยกตามตัว SupplierID 
MIN	ใช้หาค่าต่ำสุด เช่น Min(UnitPrice) `Select Min(UnitPrice) From Products`
MAX	ใช้หาค่าสูงสุด เช่น Max(UnitPrice) `Select Max(UnitPrice) From Products`
AVG	ใช้หาค่าเฉลี่ยของข้อมูล หากข้อมูลที่นำมาเฉลี่ยเป็น Null จะไม่มีการเฉลี่ยค่าบน Record นั้นเหมือนกับว่าข้าม Record นั้นไป `Select AVG(UnitPrice) From Products`
STDEV	ใช้หาค่าเบี่ยงเบนมาตราฐานของข้อมูลที่ทำการสะสมใน QUERY STDEV เป็นค่าเบี่ยงเบนมาตราฐานของกลุ่มข้อมูลตัวอย่าง มีสูตรคือ $\sqrt{\frac{n\sum x^2-\left(\sum x\right)^2}{n(n-1)}}$ `Select STDEV(UnitPrice) From Products`
STDEVP	คล้ายกับ STDEV แต่เป็นสูตรของค่าเบี่ยงเบนมาตราฐานของประชากรทั้งหมด หมายถึงข้อมูลที่เรานำมาใช้ในสูตรนี้เป็นข้อมูลจริงทั้งหมด มิได้เกิดจากการสุ่มตัวอย่าง $\sqrt{\frac{n\sum x^2-\left(\sum x\right)^2}{n^2}}$

FUNCTION	คำอธิบาย
	`Select STDEVP(UnitPrice) From Products`
VAR	เป็นค่าความแปรปรวนทางสถิติของชุดข้อมูลที่นำมาสะสมใน Aggregate Query เป็นสูตรของชุดข้อมูลที่เป็นกลุ่มตัวอย่างประชากร $\frac{n\sum x^2 - (\sum x)^2}{n(n-1)}$ `Select VAR(UnitPrice) From Products`
VARP	เป็นค่าความแปรปรวนทางสถิติของชุดข้อมูลที่นำมาสะสมใน Aggregate Query เป็นสูตรของชุดข้อมูลที่กลุ่มประชากรจริง ที่ไม่ได้มาจากการสุ่มตัวอย่าง $\frac{n\sum x^2 - (\sum x)^2}{n^2}$ `Select VARP(UnitPrice) From Products`

ตัวอย่างการใช้ Group By Query ที่ใช้ในการหาค่าสถิตการสั่งซื้อสินค่าของลูกค้าแต่ละคน ซึ่งต้องใช้การ Join Table จาก Orders และ OrderDetails

```
SELECT      Orders.CustomerID,
            Sum([UnitPrice]*[Quantity]) AS OrderAmount,
            Avg([UnitPrice]*[Quantity]) AS AVGAmount,
            Min([UnitPrice]*[Quantity]) AS MinAmount,
            Max([UnitPrice]*[Quantity]) AS MaxAmount,
            StDev([UnitPrice]*[Quantity]) AS STDAmount
FROM        Orders INNER JOIN [Order Details]
            ON Orders.OrderID = [Order Details].OrderID
GROUP BY    Orders.CustomerID;
```

customerorderStat : Select Query

Customer	OrderAmount	AVGAmount	MinAmount	MaxAmount	STDAmount
Alfreds Futterkiste	$4,596.20	$383.02	$24.00	$878.00	303.56925774
Ana Trujillo Emparedados y helados	$1,402.95	$140.30	$28.80	$348.00	135.80336776
Antonio Moreno Taquería	$7,515.35	$442.08	$20.00	$1,050.00	337.58734104
Around the Horn	$13,806.50	$431.45	$0.00	$4,050.00	705.03056671
Berglunds snabbköp	$26,968.15	$518.62	$27.00	$3,952.50	692.22326218
Blauer See Delikatessen	$3,239.80	$231.41	$30.00	$714.00	212.65451163
Blondel père et fils	$19,088.00	$734.15	$48.00	$3,465.00	734.23845839
Bólido Comidas preparadas	$5,297.80	$882.97	$249.60	$2,475.80	927.42943164
Bon app'	$23,850.95	$542.07	$37.50	$1,500.00	405.78843338
Bottom-Dollar Markets	$22,607.70	$645.93	$37.50	$2,958.00	583.95947026
B's Beverages	$6,089.90	$276.81	$34.80	$1,380.00	309.29313618
Cactus Comidas para llevar	$1,814.80	$164.98	$12.50	$364.80	126.21946615
Centro comercial Moctezuma	$100.80	$50.40	$20.80	$80.00	41.860721446
Chop-suey Chinese	$12,261.10	$645.32	$36.00	$1,590.00	477.14181048
Comércio Mineiro	$3,810.75	$381.08	$48.00	$1,485.00	460.56842974
Consolidated Holdings	$1,719.10	$245.59	$4.80	$640.50	198.59921474
Drachenblut Delikatessen	$3,763.21	$376.32	$42.00	$1,650.00	488.27055224
Du monde entier	$1,615.90	$179.54	$54.00	$585.00	173.81547048

Record: 1 of 89

การใช้ Order By หลัง Group By

เราสามารถสั่ง Order By หลัง Group By เพื่อบังคับให้มีการ Sort ตามผลที่ต้องการ บ่อยครั้งที่เราต้องการ Sort ตามมูลค่าสะสมที่เราทำไว้ เช่น บังคับให้ Sort ตามค่าเฉลี่ยในการสั่งซื้อเป็นต้น การระบุการ Order By บน field ที่เกิดจากการสะสมค่า จะต้องระบุเป็นสูตรเช่นกัน

นอกจากการกำหนด Order By แล้ว เรายังกำหนดสามารถใช้ predict ในการกำหนด TOP เพื่อเลือกเฉพาะ 5 RECORD แรกของผลการใช้ Group By Query ได้ ตามตัวอย่าง

```
SELECT      TOP 5
            Orders.CustomerID,
            Sum([UnitPrice]*[Quantity]) AS OrderAmount,
            Avg([UnitPrice]*[Quantity]) AS AVGAmount,
            Min([UnitPrice]*[Quantity]) AS MinAmount,
            Max([UnitPrice]*[Quantity]) AS MaxAmount,
            StDev([UnitPrice]*[Quantity]) AS STDAmount
FROM        Orders INNER JOIN [Order Details]
            ON Orders.OrderID = [Order Details].OrderID
GROUP BY    Orders.CustomerID
ORDER BY    Avg([UnitPrice]*[Quantity]) DESC;
```

customerorderStat : Select Query

Customer	OrderAmount	AVGAmount	MinAmount	MaxAmount	STDAmount
QUICK-Stop	$117,483.39	$1,366.09	$40.00	$15,810.00	1970.4118098
Simons bistro	$18,138.45	$1,209.23	$16.00	$10,540.00	2620.7416492
Piccolo und mehr	$26,259.95	$1,141.74	$100.00	$10,540.00	2114.5210881
Ernst Handel	$113,236.68	$1,110.16	$40.00	$6,360.00	1222.7678385
Hanari Carnes	$34,101.15	$1,065.66	$77.00	$15,810.00	2735.7610892

Record: 1 of 5

การใช้ HAVING ในการระบุเงื่อนไข

HAVING เหมือนกับ WHERE แต่เราใช้ WHERE ระบุเงื่อนไขของชุดข้อมูลที่จะเข้ามาทำ QUERY ส่วน HAVING ใช้สำหรับกำหนดเงื่อนไขกับผลที่ได้ทำการสะสมยอดแล้ว เช่น สั่งให้ทำการ SUM แล้วต้องการเฉพาะที่ผลการ SUM ที่มากกว่าเท่านั้น เท่านี้เป็นต้น

การกำหนด HAVING จะกำหนดหลัง GROUP BY และ ก่อน ORDER BY เช่นตัวอย่างข้างต้น หากเราต้องการข้อมูลที่มีค่าเฉลี่ยในการสั่งซื้อสูงกว่า 1000 จะได้ SQL เป็น

```
SELECT      Orders.CustomerID,
            Sum([UnitPrice]*[Quantity]) AS OrderAmount,
            Avg([UnitPrice]*[Quantity]) AS AVGAmount,
            Min([UnitPrice]*[Quantity]) AS MinAmount,
            Max([UnitPrice]*[Quantity]) AS MaxAmount,
            StDev([UnitPrice]*[Quantity]) AS STDAmount
FROM        Orders INNER JOIN [Order Details]
            ON Orders.OrderID = [Order Details].OrderID
GROUP BY    Orders.CustomerID
HAVING      Avg([UnitPrice]*[Quantity])  > 1000
ORDER BY    Avg([UnitPrice]*[Quantity]) DESC;
```

customerorderStat : Select Query

Customer	OrderAmount	AVGAmount	MinAmount	MaxAmount	STDAmount
QUICK-Stop	$117,483.39	$1,366.09	$40.00	$15,810.00	1970.4118098
Simons bistro	$18,138.45	$1,209.23	$16.00	$10,540.00	2620.7416492
Piccolo und mehr	$26,259.95	$1,141.74	$100.00	$10,540.00	2114.5210881
Ernst Handel	$113,236.68	$1,110.16	$40.00	$6,360.00	1222.7678385
Hanari Carnes	$34,101.15	$1,065.66	$77.00	$15,810.00	2735.7610892
Hungry Owl All-Night Grocers	$57,317.39	$1,042.13	$22.40	$9,903.20	1697.0509871
Mre Paillarde	$32,203.90	$1,006.37	$112.00	$10,329.20	1833.2612140

Record: 1 of 7

ACTION QUERY

เป็น QUERY ชนิดที่ไม่ RETURN ค่าเป็น RECORDSET แต่ใช้ในการจัดการกับข้อมูลบางอย่าง ได้แก่การ UPDATE, INSERT , DELETE เป็นต้น

UPDATE QUERY

ใช้ในการปรับปรุงข้อมูลใน TABLE สามารถกำหนดขอบเขตจำนวน Record ที่จะทำการ Update ได้โดยการระบุคำสั่ Where

เราใช้ Update ในการ Update ข้อมูลทีละหลายๆ Record พร้อมกัน โดยมีรูปแบบคำสั่งคือ

```
UPDATE      table
SET         field = ValueExpression [,field2 = ValueExpression2][,..]
WHERE       criteria
```

องค์ประกอบของคำสั่ง Update จะประกอบด้วย 3 ส่วนคือ

ส่วนที่	คำอธิบาย
table	ชื่อ Table ที่ต้องการ Update สามารถใช้ Join Table ได้ แต่การ Join บางประเภท อาจทำให้ไม่สามารถ Update ข้อมูลได้ ควรใช้การ Update ข้อมูลลงบน Table เดียวจะปลอดภัยกว่า
SET ค่า	เป็นชุดรายการ Field ที่ต้องการ Update อยู่หลังจาก SET ระบุเป็นราย Field ขั้นด้วย Comma การระบุ จะด้วยระบุชื่อ Field ที่ต้องการ Update , เครื่องหมายเท่ากับ และค่าที่ต้องการ Update สามารถระบุเป็น Expression รวมทั้งการใช้ สูตรในการ Update ได้เช่นกัน
เงื่อนไข	ระบุเงื่อนไข Where อยู่หลัง Where ใช้แบบเดียวกับที่ทำในการ SELECT ข้อมูล

ตัวอย่างการ Update ข้อมูล Table Employees โดยเปลี่ยนรหัสผู้บังคับบัญชาของกลุ่มคนที่เคยอยู่ใต้รหัสพนักงานรหัส 2 ไปเป็นรหัส 5

```
UPDATE Employees SET ReportsTo = 5 WHERE ReportsTo = 2;
```

ตัวอย่างนี้ ทำการเพิ่มราคาสินค้าใน Table Products โดยเพิ่มราคา 10 % ในสินค้าของ SupplierID = 8 และเป็นสินค้าที่ยังไม่ DisContinue

```
UPDATE Products SET UnitPrice = UnitPrice * 1.1
WHERE SupplierID = 8 AND Discontinued = No;
```

ตัวอย่างการใช้ JOIN QUERY เพื่อทำการ Update ข้อมูล โดยทำการลดราคาสินค้าเป็นมูลค่า 5 % จากราคาเดิมในสินค้าที่มาจาก Tokyo Traders ใช้การ Join Products กับ Suppliers เพื่อจะได้ Where ชื่อ Supplier ได้

```
UPDATE      Suppliers INNER JOIN Products
            ON Suppliers.SupplierID = Products.SupplierID
SET         UnitPrice = UnitPrice * .95
WHERE       CompanyName = 'Tokyo Traders' AND Discontinued = No;
```

หากเปรียบเทียบกับการใช้ RecordSet แล้ว การ Update ด้วย SQL COMMAND ทำได้เร็วกว่า เพราะทำทีละหลายๆ Record ได้ แต่ใช้กับการ Update ที่ไม่ซับซ้อนเท่านั้น เพราะหากเงื่อนไขการ Update ที่ซับซ้อน การใช้ RecordSet จะทำได้ดีกว่า

INSERT QUERY

ใช้ในการเพิ่ม RECORD เข้าไปใน TABLE สามารถ Insert ทีละหลาย Record หรือทีละ Record ก็ได้ มีรูปแบบคำสั่ง 2 แบบดังนี้

แบบที่ละหลาย Record ใช้วิธี Select ข้อมูลจาก Table อื่นเพื่อ Insert เข้าไปใน Table

```
INSERT INTO target [IN externaldatabase] [(field1[, field2[, ...]])]
SELECT     [source.]field1[, field2[, ...]
FROM       tableexpression
WHERE      condition
```

แบบที่ละ Record เราใช้

```
INSERT INTO target [(field1[, field2[, ...]])]
VALUES (value1[, value2[, ...])
```

การ INSERT แบบหลาย Record ช่วยให้สามารถ INSERT พร้อมกันได้หลาย RECORD โดยการกำหนดเงื่อนไขลงที่การ SELECT ซึ่งการ SELECT นี้ เหมือนกับการ SELECT ทั่วไป เพียงแต่ให้รายการ Column ในผลการ SELECT ตรงกันกับที่แสดงไว้ใน FIELD LIST และมีประเภท Field แบบเดียวกัน ก็ใช้ได้

ส่วนที่	คำอธิบาย
Target	ระบุชื่อ TABLE ที่ต้องการ ADD
Externaldatabase	กรณีที่เป็นการ Insert เข้าไปใน Table ที่อยู่คนละไฟล์กับ Database ปัจจุบัน ให้ระบุชื่อไฟล์ที่นี่ หากไม่ระบุ ถือเป็นการ Insert ในไฟล์เดียวกัน โดยใช้ IN นำหน้าชื่อไฟล์
FIELD	ระบุชื่อ FIELD ใน Table ที่กำลังเพิ่มข้อมูลเข้าไป ว่าจะแทนค่า Field ใดบ้าง จำนวน Field ที่แสดง จะต้องสอดคล้องกับที่ระบุใน แหล่งข้อมูล
แหล่งข้อมูล	ระบุเป็น SELECT QUERY หากใช้ข้อมูลที่มีอยู่เป็นข้อมูลที่จะ Insert เข้าไปในฐาน การระบุ SELECT QUERY สามารถใช้ Join Table, Group Query ระบุเงื่อนไข Where หรือ Having เพื่อกำหนดข้อมูลที่จะเข้าไปใน Table ได้เช่นกัน หากระบุด้วย VALUES จะเป็นการระบุค่าที่จะ Insert ลงไปใน Table แบบนี้เป็นการ Insert แบบทีละ 1 Record

ในการทำงานกับ RecordSet เราอาจไม่ใช้คำสั่ง AddNew - Update เลยก็ได้ เพราะเราสามารถใช้คำสั่ง Insert Query แบบ Values แทนกันได้ แต่การกำหนด SQL STRING จะดูยุ่งยากกว่า แต่ให้ผลที่เร็วกว่าการ Add ผ่าน Recordset มาก

ตัวอย่างการใช้ Insert Query ด้วย VALUES (อย่าลืมว่าการกำหนด String ต้องคลุมด้วย ' หัวท้ายเสมอ)

```
INSERT INTO Employees (FirstName,LastName, Title)
VALUES     ('Uthai', 'Kiattivikrai', 'Trainee');
```

ตัวอย่างการ Add ข้อมูลเข้า Table Customers โดยเลือกจากรหัสพนักงาน โดยถือว่าพนักงานทุกคนสามารถสั่งซื้อสินค้าได้ จึงทำการ Copy รายชื่อพนักงานมาเป็นรายชื่อลูกค้าด้วย

```
INSERT INTO Customers ( CustomerID, ContactName, CompanyName )
SELECT      Employees.EmployeeID,
            [FirstName] & " " & [LastName] AS Expr1,
            "Internal Customer" AS Expr2
FROM        Employees;
```

DELETE QUERY

เป็น QUERY ที่ใช้ในการลบข้อมูล ใน TABLE มีรูปแบบคำสั่งคือ

```
DELETE      [table.*]
FROM        table
WHERE       criteria
```

ระบุชื่อ Table ที่ต้องการลบที่ชื่อ Table และเงื่อนไข Where ที่ต้องการลบ ส่วน [table.*] ระบุ หรือไม่ระบุก็ได้ สมมุติว่าเราต้องการลบข้อมูลใน Table Customers จากที่ได้ Add เข้าในตัวอย่างกก่อนหน้า เราใช้คำสั่งดังนี้

```
DELETE      FROM Customers
WHERE       CustomerID <"A";
```

SELECT … INTO QUERY

ใช้ในการสร้าง Table ใหม่จากการ SELECT ข้อมูล ใช้เหมือนคำสั่ง SELECT ทั่วไป แต่กำหนด INTO เพื่อสั่งให้นำผลที่ทำการ SELECT แล้ว ไปสร้างเป็น TABLE ใหม่ มีรูปแบบคำสั่งคือ

```
SELECT      field1[, field2[, ...]]
INTO        newtable [IN externaldatabase]
FROM        source
[WHERE      condition]
[GROUP BY   grouplist]
[HAVING     havinglist]
```

ส่วนที่แทรกมาจากเดิมคือ บรรทัดที่ 2 นอกนั้นเหมือนการใช้ SELECT ปกติ

ส่วนเพิ่ม	คำอธิบาย
newtable	ชื่อ Table ใหม่ที่ต้องการสร้าง
externaldatabase	กรณีที่ต้องการสร้าง Table ไว้ในไฟล์ฐานข้อมูลที่อยู่คนละไฟล์กับ ไฟล์ปัจจุบัน ให้ระบุ IN ตามด้วยชื่อไฟล์ Database ที่ต้องการส่ง Table ใหม่เข้าไปอยู่

ตัวอย่างนี้เป็นการนำ Query เดิมที่ใช้ Query ค่าสถิติในการสั่งซื้อ มาสร้างเป็น Table เก็บไว้ในชื่อ Table ORDERSTAT

```
SELECT      Orders.CustomerID,
            Sum([UnitPrice]*[Quantity]) AS OrderAmount,
            Avg([UnitPrice]*[Quantity]) AS AVGAmount,
            Min([UnitPrice]*[Quantity]) AS MinAmount,
            Max([UnitPrice]*[Quantity]) AS MaxAmount,
            StDev([UnitPrice]*[Quantity]) AS STDAmount
INTO        ORDERSTAT
FROM        Orders INNER JOIN [Order Details]
```

```
            ON Orders.OrderID = [Order Details].OrderID
GROUP BY    Orders.CustomerID
```

UNION QUERY

การใช้ UNION QUERY เหมือนกับการใช้ QUERY ที่เป็น SELECT QUERY หลายๆ ตัวมาบวกต่อกันไปเรื่อย

มีรูปแบบคือ

```
[TABLE] query1 UNION [ALL] [TABLE] query2 [UNION [ALL] [TABLE] queryn [ ... ]]
```

ระบุ ALL

มีเงื่อนไขในการระบุหลัง UNION คือ ต้องการ ALL หรือไม่ ALL

ถ้าระบุ ALL เราจะได้ทุก Record แม้ว่าแหล่งข้อมูลที่มา UNION จะให้ข้อมูลที่ซ้ำกัน ถ้าไม่ระบุ ALL โดยว่างไว้ จะถือว่าให้ SQL ตัด Record ที่ซ้ำออกไป

การ UNION

แหล่งข้อมูลที่มานำมา UNION กัน จะต้องมีจำนวน FIELD เท่ากัน และ ประเภทของ FILED ในแบบเดียวกัน จึงจะ UNION ได้

การ Union สามารถ UNION ต่อกันไปได้เรื่อยๆ การระบุแหล่งข้อมูลที่มาทำการ UNION สามารถทำได้ 2 แบบ คือระบุชื่อ TABLE หรือใช้ SELECT QUERY ตามปกติ เช่น ตัวอย่างนี้ สมมุติว่ามี TABLE ชื่อ New Accounts ซึ่งมีโครงแบบเดียวกับ Customer แล้ว

```
TABLE       [New Accounts]
UNION ALL
SELECT      *
FROM        Customers
WHERE       OrderAmount > 1000;
```

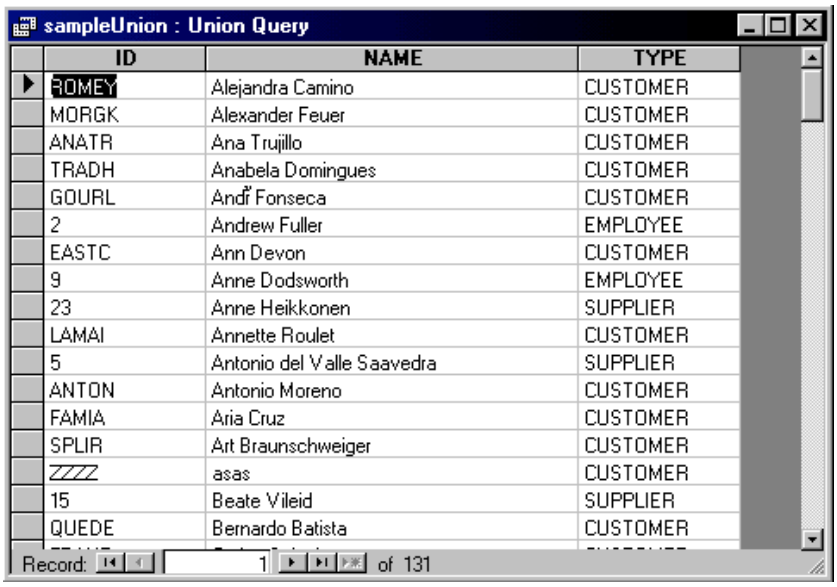
ระบุ ORDER BY

เราสามารถระบุ ORDER BY ลงบนส่วนท้ายของการ UNION เพื่อให้ผลของการ UNION จัดเรียงข้อมูลตามที่เราต้องการ ทดลองนำ EMPLOYEES, SUPPLIERS, CUSTOMERS มา UNION รวมกัน

```
SELECT      CUSTOMERID AS ID ,
            CONTACTNAME AS NAME ,
            'CUSTOMER' AS TYPE
FROM        CUSTOMERS
UNION
SELECT      SUPPLIERID AS ID ,
            CONTACTNAME AS NAME,
            'SUPPLIER' AS TYPE
FROM        SUPPLIERS
```

```
UNION
SELECT      EMPLOYEEID AS  ID,
            FIRSTNAME & ' ' & LASTNAME AS NAME,
            'EMPLOYEE' AS TYPE
FROM        EMPLOYEES

ORDER BY NAME
```

ตัวอย่างนี้นอกจากจะทำการ UNION ด้วย FIELD แล้วยังทำ FILED ด้วย TEXT ชื่อ TYPE ที่บอกประเภทของ TABLE เป็น CUSTOMER, SUPPLIER, EMPLOYEE เพื่อใช้ในการ UNION เมื่อเราสั่ง SORT จะได้มี FIELD ดังกล่าวบอกเราว่า RECORD นี้มาจาก TABLE อะไร เราอาจได้ใช้กลไกดังกล่าวต่อไปในอนาคต เช่นถ้าเราใช้ QUERY นี้เป็น COMBO BOX เราจะทราบว่า บุคคลที่ USER เลือกเป็นกลุ่มบุคคลใด ต้องอ้างอิงที่ TABLE ใด เป็นต้น



การใช้ SQL ผ่าน DatabaseObject

ในส่วนนี้เราจะเข้ามาทำความเข้าใจการใช้ SQL บน DATABASE OBJECT เพื่อให้การทำงานของ DATABASE OBJECT สามารถทำงานได้ซับซ้อนขึ้น

การต่อ STRING ให้เป็น SQL

การส่ง SQL เข้า DATABASE OBJECT จะต้องสั่งเป็น STRING ดังนั้นเราจะต้องสร้าง STRING ตามเงื่อนไข SQL ที่กำหนดก่อนที่จะส่งเข้าไปที่ DATABASE OBJECT

สมมุติว่าเราต้อง ทำ RECORDSET ที่เก็บ รหัส CUSTOMER ที่มียอดสั่งซื้อมากกว่าระดับที่กำหนด เราต้องทำดังนี้

```
Sub RecordReturn()

  Dim rst As Recordset
  Dim strSQL As String
  Dim setAmount
  strSQL = " SELECT CUSTOMERID, AVG(UNITPRICE * QUANTITY) as AVGAmount FROM " & _
           " ORDERS O INNER JOIN [ORDER DETAILS] D ON  O.OrderID = D.OrderID " & _
           " GROUP  BY CUSTOMERID " & _
           " HAVING AVG(UNITPRICE * QUANTITY) >  "


  setAmount = InputBox("Select price")
  strSQL = strSQL & setAmount

  Set rst = dbEngine(0)(0).OpenRecordset(strSQL)

  Do Until rst.EOF

     Debug.Print rst(0), rst(1)
     rst.MoveNext

  Loop
  Set Rst = Nothing

End Sub
```

สำหรับ & _ ใช้ในการต่อบรรทัด ใช้เมื่อเราเขียนคำสั่งใน VB แล้วไม่จบในหนึ่งบรรทัด VB จะถือบรรทัดที่ตอมาเป็นคำสั่งในบรรทัดเดียวกัน

ตัวอย่างนี้เป็นการต่อ STRING เพื่อให้ได้เป็น SQL ที่ต้องการ ก่อนส่งเข้าไปในการ OpenRecordSet มีการอ่านค่าจาก INPUTBOX เพื่อรับค่าที่ต้องการต่ำสุดของยอดสั่งซื้อเฉลี่ย แล้วนำไปต่อกับ SqlStr

เราอาจไม่ใช้ตัวแปรมารับค่า SQL ก่อนก็ได้ โดยใส่ในวงเล็บของการ OpenRecordSet ทันที แต่การใช้ String ในการสร้าง SQL ก่อนจะทำให้ โปรแกรมอ่านเข้าใจง่ายขึ้น ตัวอย่างการใส่คำสั่ง SQL เข้าไปใน OpenRecordSet เลย เช่น

```
Set rst = dbEngine(0)(0).OpenRecordset("Select ProductID, UnitPrice from
Products")
```

การต่อรวมตัวแปร String

ปัญหาในการต่อ SQL String มักเป็นเรื่องของการต่อค่าตัวแปรเข้าไปใน String สมมุติว่าเราต้องการ Where ProductName ด้วยการ Like ตามค่าตัวแปรที่ถูกกำหนดโดย User การนำตัวแปรดังกล่าวไปแทรกในชุด SQL จะต้องแทรกระหว่างเครื่องหมาย Quat String (‘) ทำให้ต้องระมัดระวังในการตัด String

```
Sub StringTEst()

    Dim SearchX
    Dim strSQL As String
    Dim rst As Recordset

    SearchX = InputBox("Enter any named, can use wildcard ?")
    strSQL = "Select ProductID, ProductName from Products " & _
             " Where Productname like '" & SearchX & "'"
```

```
    MsgBox ("Your SQL is " & vbCrLf & StrSQL)
    Set rst = dbEngine(0)(0).OpenRecordset(strSQL)
    If rst.RecordCount > 0 Then
       rst.MoveLast
       MsgBox ("Found : " & rst.RecordCount & " Record(s)")
    End If
    Set rst = Nothing

End Sub
```

ตัวอย่างนี้แทรกตัวแปร String ชื่อ SearchX เข้าไปใน Where Clause เพื่อสร้างเป็นชุด SQL ในการค้นหา ส่วนท้ายของ SearchX ยังคงต้องต่อด้วย & “’” เพื่อให้ชุด SQL ปิดสมบูรณ์ด้วยเครื่องหมาย ‘ ทั้ง 2 ด้าน

คำสั่ง Msgbox จะช่วยให้เราเห็นกลไกการต่อ SQL STRING ได้ดีขึ้น

การต่อรวมกับตัวแปรวันที่

การใช้เงื่อนไขกับตัวแปรวันที่ หรือการทำการด้วยตัวแปรวันที่ ควรจะ Convert ค่า วันที่ให้เป็น MM/DD/YYYY โดยมี # ปิดหัวท้าย เช่น

```
Sub DateTest()
    Dim DateX
    Dim strSQL As String
    Dim rst As Recordset

    DateX = InputBox("Date to where ?")
    strSQL = "Select OrderID from Orders" & _
             " Where OrderDate < #" & Format(DateX, "MM/DD/YYYY") & "#"
    MsgBox ("Your SQL is " & vbCrLf & StrSQL)
    Set rst = dbEngine(0)(0).OpenRecordset(strSQL)
    If rst.RecordCount > 0 Then
       rst.MoveLast
       MsgBox ("Found : " & rst.RecordCount & " Record(s)")
    End If
    Set rst = Nothing
End Sub
```

การใช้ ACTION QUERY

เราใช้ ACTION QUERY กับ Object ระดับ Database ได้เลย โดยใช้คำสั่ง EXECUTE

```
Sub RunTest()

    Dim StrSQL As String
    Dim numX
    numX = InputBox("Enter percent to up ?")
    StrSQL = "Update Products set UnitPrice = UnitPrice * (1+" & numX / 100 & ") "
    MsgBox ("Your SQL is " & vbCrLf & StrSQL)
    dbEngine(0)(0).Execute StrSQL

End Sub
```

LAB FOR DB102 : DATABSE PROGRAMMING

บทที่ 1 การพัฒนา SOFTWARE ด้วย ACCESS

- DATABASE APPLICATION ENVORONMENT
- การจัดการ FIRST SCREEN ด้วย FORM แล้ว GUIDE ไปจนจบขบวนการ

LAB1:การ ATTACH TABLE

1. ทำเครื่องครูให้เป็น SERVER
2. ใช้ NORTHWIND ไฟล์ เป็น SHARE FILE ไว้ที่เครื่องครู
3. ให้นักเรียน ATTACH ไฟล์มายังเครื่องครู

- อธิบายกลไกการทำงานบนระบบเครือข่ายแบบ FILE SHARING
- ทดลองให้มีการ UPDATE พร้อมกันจากหลายๆ Terminal
- การใช้ Rerefsh (F9)

LAB2:การ ATTACH TABLE กับ dBF

1. ทำเครื่องครูให้เป็น SERVER
2. ใช้ DBF FILE ตัวอย่างเป็น SHARE FILE ไว้ที่เครื่องครู
3. ให้นักเรียน ATTACH ไฟล์มายังเครื่องครู

LAB3:การแยก DATABASE กับ โปรแกรม (WIZARD)

LAB4:การแยก DATABASE กับ โปรแกรม (MANUAL)

1. COPY ไฟล์ MDB ไปเป็น PRG.MDB กับ DATA.MDB
2. ลบ TABLE ใน PRG.MDB
3. ลบทุกย่างยกเว้น TABLE ใน DATA.MDB
4. ทำ ATTACH TABLE

LAB5:การตั้งค่า STARTUP PARAMETER

1. สร้าง MDB ที่ต้องการ
2. สร้าง FORM 1 FORM
3. กำหนด STARTUP ของ DATABASE ให้เป็น FORM ดังกล่าว
4. ปิดโปรแกรม

LAB6:การทำ STARTUP SCREEN / ICON

(ไม่มีในตำรา ต้องเตรียมตัวอย่าง ICON ไฟล์มาใช้ในการทำ LAB)

1. ทำ FILE BMP มีชื่อเดียวกับตัว MDB สำหรับ STARTUP แล้วไว้ที่ Directory เดียวกัน
2. ทำ FILE ICO มีชื่อเดียวกับตัว MDB สำหรับ ICON แล้วไว้ที่ Directory เดียวกัน
3. ทดลองเปิดไฟล์ MDB

บทที่ 2 VISUAL BASIC สำหรับ ACCESS

LAB1:การทำ FIRST APP

Lสร้าง FIRST APP ตามขั้นตอนในหนังสือ

- พยายามให้ผู้เรียนเข้าใช้ EVENT_SUB ของ ACCESS

LAB2:การใช้ CODE WINDOWS

- ให้ผู้เรียนคุ้นเคยกับเครื่องมือในการเขียน

1. สร้าง FORM
2. กด Mouse เข้าสู่หน้าจอ CODE ของ FORM
3. กำหนดการแสดงผลของ CODE

LAB3:การ RUN และใช้ BREAK POINT

- ให้ผู้เรียนเข้าใจถึงการทำงานของโปรแกรมทั่วไป ควร STEP ให้ผู้เรียนเห็น เวลาที่โปรแกรม Execute

1. ใช้ตัวอย่าง BEEP โปรแกรม
2. ทดลองใช้ BREAK POINT

LAB4:การใช้ DEBUG WINDOWS

- ให้ผู้เรียน ใช้ DEBUG WINDOWS เหมือนกระดาษทด

LAB5: SUB

- ใช้ตัวอย่างการเรียก SUB ตามหนังสือ

LAB6:FUNCTION

- ใช้ตัวอย่างการเรียก FUNCTION ตามหนังสือ

LAB7:การทำ STANDARD MODULE

- ให้ผู้เรียนเข้าใจถึงการสร้าง SUB ไว้ที่ STANDARD MODULE เพื่อการใช้งานร่วมกันทั้งโปรแกรม

1. ทดลองสร้าง SUB ไว้ใน STANDARD MODULE แล้ว CALL จาก FIRST APP
2. ทดลองสร้าง SUB ไว้ใน EVENT MODULE MODULE แล้ว CALL จาก FIRST APP อีกที

LAB8:การใช้ IF

- ขอให้ใช้ตัวอย่างที่มีในหนังสือ
- ทดลองทำโปรแกรมการตรวจสอบรหัสผ่าน โดยใช้ INPUTBOX แล้วถาม PASSWORD ที่ต้องการ

LAB9:การใช้ CASE

- ทำต่อจากคำสั่ง IF แต่ให้ตรวจสอบว่า PASSWORD ที่ใส่มี 7 ตัวตามวัน เทียบกับ IF ที่ตรวจสอบได้ตัวเดียว

```
Sub Casetest()
    Dim passme
    Select Case WeekDay(Date)
    Case 1
        passme = "123"
    Case 2
        passme = "456"
    Case 3
        passme = "4561"
    Case 4
        passme = "4563"
    Case 5
        passme = "236"
    Case 6
        passme = "4526"
    Case 7
        passme = "236"
    End Select
    If InputBox("Enter password") = passme Then
       MsgBox ("OK")
    Else
       MsgBox ("FAIL")
    End If

End Sub
```

LAB10:การใช้ FOR/NEXT/LOOP

- ใช้ตัวอย่างตามหนังสือ

LAB11:การต่อ STRING

- ให้ตัวอย่างการต่อ STRING กับผู้เรียน
- ต้องการ STRING ในแบบ

  SELECT * FROM [TABLE} where [TABLE]ID ='[VALUE]'

  โดย [TABLE], [VALUE] ให้กรอกโดย USER เช่น

  SELECT * FROM CUSTOMER where CUSTOMERID ='345'

```
Sub TestString()
    Dim x, y, z
    x = InputBox("Enter Table")
    y = InputBox("Enter Where")
    z = "SELECT * FROM " & x & " where " & x & "ID ='" & y & "'"
    MsgBox (z)
End Sub
```

LAB12:การเปรียบเทียบ STRING

- ตามตัวอย่างในหนังสือ

LAB13:การประกาศตัวแปร

- ทดลองทำตัวอย่างที่มีการประกาศตัวแปรไว้ใน STANDARD MODULE แล้วเรียกใช้

- ให้ข้อดีข้อเสียการประกาศตัวแปร (มีในหนังสือ)

LAB14:การ CONVERT ตัวแปร

- ทดลองทำตามในตัวอย่าง
- เพิ่มคำสั่ง ROUND ใช้ในการปัดเศษ

LAB15:ทำตัวเลข RANDOM (ใช้ INT)

- Rountine นี้ให้ผู้เรียนทดลองใช้ INT ในการตัดเฉพาะตัวเลข INTEGER

```
Sub Randomme()
    Dim x, i, tmpstr
    tmpstr = ""
    For i = 1 To 100
        Randomize
        x = Int(10 * Rnd)
        tmpstr = tmpstr & "-" & x
        If i Mod 10 = 0 Then
           tmpstr = tmpstr & vbCrLf
        End If
    Next i
    MsgBox (tmpstr)

End Sub
```

LAB16:ใช้คำสั่ง FORMAT

- ตามหนังสือ

LAB17:การใช้ STRING FUNCTION

- ตามหนังสือ

LAB18:ตัดตัวเลข

- สร้างโปรแกรมที่ตัดตัวเลขเป็นตัว เพื่อให้เห็นการใช้ MID

```
Sub MidMe()
    Dim x, i, fname
    x = InputBox("Enter number")
    For i = 1 To Len(x)
        fname = "IMG" & Mid(x, i, 1) & ".JPG"
         MsgBox (fname)
    Next i
End Sub
```

LAB19:การควบคุม ERROR

บทที่ 3 FORM / REPORT MEMBER

LAB1:อ่าน CONTROL NAME

1. ทดลองสร้าง Form หนึ่ง Form แล้ว Save ใช้ชื่อ CTLTest
2. ใส่ Control ประเภท Text Box ลงใน FORM 3 ตัว ตั้งชื่อเป็น TEXT1, TEXT2, TEXT3
3. สร้าง Code เข้าไปในส่วนของ Detail_Click

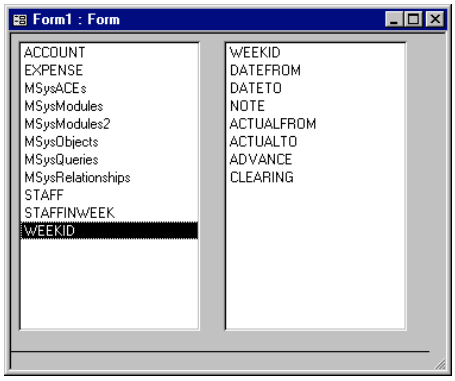
```
Dim i
For i = 0 To Me.Controls.Count - 1

    MsgBox (Me.Controls(i).Name)

Next I
```

ทดลอง RUN FORM แล้ว CLICK ที่ DETAIL

LAB2:Assign ค่าลงบน Properties

ทดลองสร้าง FORM ในการอ่านรายชื่อ TABLE ใน DATABASE มาแสดงใน LISTBOX แล้วแสดงชื่อ Field ทั้งหมดใน TABLE นั้นๆ

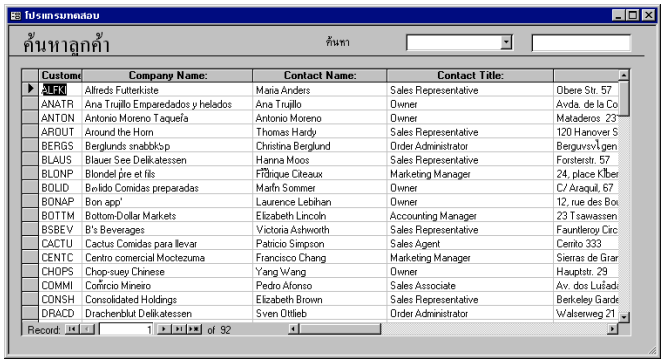
1. สร้าง FORM ใหม่ 1 FORM
2. สร้าง CONTROL ประเภท LISTBOX ใช้ชื่อว่า TABLELIST กำหนด ROWSOURCETYPE เป็น VALUELIST
3. สร้าง CONTROL ประเภท LISTBOX ใช้ชื่อว่า FIELDLIST กำหนด ROWSOURCETYPE เป็น FIELDLIST
4. สร้าง CODE ลงใน FORM_LOAD และ TABLELIST_BEFOREUPDATE ตามตัวอย่าง




```
Private Sub Form_Load()
        Dim i, strx

        For i = 0 To DBEngine(0)(0).TableDefs.Count - 1
            strx = strx & DBEngine(0)(0).TableDefs(i).Name & ";"


        Next i
        Me.TABLELIST.RowSource = strx

End Sub

Private Sub TABLELIST_BeforeUpdate(Cancel As Integer)
        Me.COLUMNLIST.RowSource = Me.TABLELIST
End Sub
```

LAB3:Assign ค่าลงบน Properties (2)

ทดลองสร้าง FORM สำหรับค้นหา CUSTOMER แบบ DATASHEET คล้ายกับในหนังสือ แต่ค้นหาใน Subform



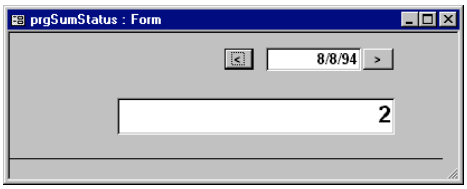
1. สร้าง SUB FORM ที่ โดยมี RECORDSOURCE เป็น CUSTOMER
2. กำหนดให้ Deafault View และ View Allow เป็น Data Sheet
3. เลือก Field ที่ต้องการนำมาแสดง
4. Save Form โดยกำหนดชื่อเป็น LAB3_3Sub

5. สร้างเปล่า สำหรับเป็นเป็น Form แม่
6. ลาก SubForm จากที่ทำไว้ (LAB3_3Sub)
7. สร้าง Combo Box ให้ชื่อว่า SearchSelect กำหนดให้ RowSourceType เป็น FieldList และ RowSource เป็น Customer
8. สร้าง Text Box แล้วกำหนด Code ดังนี้

```
Private Sub txtSearch_BeforeUpdate(Cancel As Integer)
        Me.LAB3_3Sub.Form.Filter = Me.SearchSelect & " Like '" &
Me.txtSearch & "'"
        Me.prgCusSub.Form.FilterOn = True
End Sub
```

LAB4:หน้าจอแสดงยอดขายประจำวัน

ทำหน้าจอแสดงจำนวน Order ประจำวัน



1. สร้าง FORM มี Text Box 1 ตัวตั้งชื่อเป็น DateSearch กำหนด Default เป็น Date()
2. สร้างปุ่ม cmdDec กับปุ่ม cmdInc
3. สร้าง TextBox หนึ่งตัว ตั้งชื่อเป็น SumOrder บรรจุสูตรลงใน Properties ขอว Control Source ดังนี้
   `=DCount("*","Orders","OrderDate = #" & [DateSearch] & "#")`
4. บระะจุ Code ลงใน ปุ่ม

```
Private Sub cmdDec_Click()
        DateSearch = DateSearch - 1

End Sub
```

```
Private Sub cmdInc_Click()
        DateSearch = DateSearch + 1

End Sub
```

LAB5:เพิ่ม Timer ลงใน Form

ทำต่อจาก Lab 5 เพื่อให้ อ่านข้อมูลทุก 1 วินาที โดยใช้ Method Requery

1. กำหนด TimerInternal ของ Form เป็น 1000
2. ใส่ Code ลงใน Timer Event ของ Form

```
Private Sub Form_Timer()
        SumOrder.Requery
End Sub
```

3. เปิด Form ที่ทำเสร็จ แล้วทดลองบันทึกข้อมูลลงใน Order เพิ่ม โดยกำหนดวันให้ตรงกับที่กำหนดบนจอ จะพบว่าจำนวน Order จะเพิ่มตาม

LAB6:ทำ Form บันทึกข้อมูล ที่เลือก Table และ ค้นหาได้

ทำตามขั้นตอนในการสร้าง Form ที่เปลี่ยน Source Object ตามตัวอย่างในหนังสือ

1. เพิ่ม Combo Box สำหรับเลือก Filed ที่ต้องการค้น และสร้าง Text Box แบบเดียวกับการทำหน้าค้นหา Customer / Supplier ทั่วไป
2. ปรับ Code ที่อยู่ใน SelectX ให้กำหนด RowSource ของ ComboBox ชื่อ OrderSelect

```
Option Compare Database
Option Explicit

Private Sub SearchMe_BeforeUpdate(Cancel As Integer)
      Me.SubX.Form.Filter = Me.OrderSelect & " Like '" & Me.SearchMe & "'"
      Me.SubX.Form.FilterOn = True

End Sub

Private Sub SelectX_BeforeUpdate(Cancel As Integer)
        Dim tmpStr
        Me.SubX.SourceObject = "sub" & SelectX
        Select Case SelectX
        Case 1
            tmpStr = "Customers"
        Case 2
            tmpStr = "Suppliers"
        Case 3
            tmpStr = "Employees"
        End Select
        Me.OrderSelect.RowSource = tmpStr
        Me.OrderSelect.Requery

End Sub
```

บทที่ 4 คำสั่งพิเศษ

LAB1:แก้ไขระบบ MENU ให้มี Menu Group

1. สร้าง TABLE อีก 1 TABLE ชื่อ menuGroup

Primary	FieldName	FieldType	Size
Yes	GroupID	Number	Byte
	GroupName	Number	Byte

2. เพิ่ม Field GroupID ลงใน Table menuMain จากตัวอย่างในหนังสือ

Primary	FieldName	FieldType	Size
Yes	MenuID	AutoNumber	
	MenuSequence	Number	Byte
	MenuName	Text	100
	MenuCall	Text	100
	GroupID	***Number***	***Byte***

3. บันทึกข้อมูลตัวอย่างลงใน Table menuGroup

GROUPID	GROUPNAME
1	บันทึกข้อมูล
2	บันทึกหมวด

4. เปิด Table menuMain มาใส่ข้อมูล GROUPID โดย Menu ที่เป็น Form เป็น GroupID = 1 และ Menu ที่เป็น Report เป็น GroupID = 2
5. นำ Form Mainmenu มาแก้ไขโดยการสร้าง ComboBox โดยใช้ RowSource จาก Table menuGroup ให้เลือก Bound Column ที่ Field GROUPID (Column 1) ตั้งชื่อ Combo นี้ว่า GroupID
6. ใส่ Code ลงที่ ComBo ดังนี้

```
Private Sub GROUPID_AfterUpdate()
        Me.Filter = "GroupID = " & Me.GROUPID
         Me.FilterOn = True
End Sub
```

LAB2:สร้างปุ่ม QUIT ให้ MAIN MENU

1. สร้าง COMMAND บน FORM mainmenu
2. ใส่ CODE ลงบน COMMAND

```
Private Sub Command7_Click()
       DoCmd.Quit acQuitSaveNone
End Sub
```

LAB3:ทำโปรแกรมเรียก BAT ไฟล์ที่ตรวจสภาวะการทำการ

ในการที่ SHELL APPLICATION ออกนอก ACCESS จะไม่ทราบว่า ไฟล์ที่ SHELL ออกไปทำการเสร็จหรือยัง ตัวอย่างการ COPY ไฟล์จากแผ่น และต้องการทราบว่า การ COPY สมบูรณ์แล้ว จึงจะทำงานต่อได้ ให้ทำตาม LAB นี้ ต้องการ COPY ไฟล์ TMPDB จาก A: มา C:\TMPTEST

1. สร้าง FORM สำหรับทำการ COPY
2. สร้าง COMMAND สำหรับบรรจุ CODE
3. บันทึก CODE

```
Private Sub Command25_Click()
        Dim FileNum, x

    '1:Create Directory
        '

        On Error Resume Next
        MkDir "C:\TMPTEST"
        On Error GoTo 0

    '2:GENERATE BAT FILE
    '
       FileNum = FreeFile
        Open "C:\TMPTEST\DUMMY.BAT" For Output As FileNum
        Print #FileNum, "Copy A:\tmpdb.mdb C:\TMPTEST"
        Print #FileNum, "Dir C:\TMPTEST\*.* > C:\TMPTEST\finish.txt"
        Close

    '3:EXECUTE BAT FILE WHICH WE JUST CREATED
        '

        x = Shell("C:\TMPTEST\DUMMY.BAT", vbNormalFocus)

    '4:WAIT TILL the finish.txt Created
        '
        Do Until Dir("C:\TMPTEST\finish.txt") <> ""
        Loop

    '5:Whether File Existing
        '

        If Dir("C:\TMPTEST\tmpdb.mdb") <> "" Then
           MsgBox ("File Copy OK")
           Else
           MsgBox ("Fail to copy file")
        End If

End Sub
```

โปรแกรมประกอบด้วย 5 Part คือ

1. สร้าง Directory ด้วยคำสั่ง MkDir
2. สร้าง BAT ไฟล์ที่ต้องการ ด้วยการ Create File
3. เรียก BAT ไฟล์ตัวที่สร้างขึ้น
4. รอจนกว่า จะพบไฟล์ finish.txt ที่จะถูกสร้างหลังการ COPY สมบูรณ์แล้ว จึงออกจาก LOP
5. ตรวจสอบว่าไฟล์ที่ COPY มามีหรือไม่

ตรวจสอบที่ Directory C:\TMPTEST จะพบไฟล์ที่สร้างไว้

ลอง APPLY ให้โปรแกรมดังกล่าวสามารถระบุ Directory ที่ต้องการ Copy ไป Drive ที่เก็บข้อมูล และ ชื่อ File ที่ต้องการ Copy จาก Form เพื่อให้ User เลือกทำการบันทึกเองได้

บทที่ 5 DATA ACCESS OBJECT

LAB1:สร้างไฟล์ DATABASE ตามวัน

ความต้องการในการ สร้าง BACKUP ไฟล์ตามวันที่กำหนด เป็นเงื่อนไขในการใช้งานระบบฐานข้อมูล LAB นี้เป็นการสร้างไฟล์ MDB เพื่อการ BACKUP

1. สร้าง FORM เปล่า
2. สร้าง TEXTBOX วันที่ เพื่อให้ USER กำหนดได้เอง ใช้ชื่อว่า txtDATE กำหนด Default เป็น วันที่ปัจจุบัน
3. สร้าง TEXTBOX ที่เก็บ Directory สำหรับ BACKUP ใช้ชื่อว่า txtPath กำหนด Default เป็น C:\tmpBackup
4. สร้าง ปุ่มสำหรับทำการ
5. บรรจุ CODE ตามตัวอย่าง

```
Private Sub cmdBackup_Click()

        Dim db As Database
        Dim dbName

        On Error Resume Next
        MkDir Me.txtPath

        dbName = txtPath & "\" & Format(Me.txtDate, "yyyymmdd") & ".MDB"
        Kill dbName
        On Error GoTo 0


        Set db = dbEngine(0).CreateDatabase(txtPath & "\" & Format(Me.txtDate,
"yyyymmdd"), dbLangGeneral)


End Sub
```

LAB2:สร้าง TABLE อัตโนมัติ

STEP ต่อไปเป็นการสร้าง TABLE ตามที่เราต้องการ จาก dataBase ที่เราทำไว้แล้ว

1. ใช้ FORM เดิมใน LAB 1
2. บรรจุ CODE ต่อจากการ CREATE DATABASE โดย LAB นี้จะทำการสร้าง เฉพาะ TABLE CUSTOMERS

```
Private Sub cmdBackup_Click()

        Dim db As Database
        Dim dbName

        On Error Resume Next
        MkDir Me.txtPath

        dbName = txtPath & "\" & Format(Me.txtDate, "yyyymmdd") & ".MDB"
        Kill dbName
        On Error GoTo 0
```

```
        Set db = dbEngine(0).CreateDatabase(txtPath & "\" & Format(Me.txtDate,
"yyyymmdd"), dbLangGeneral)

    'FOR GENERATE TABLE STRUCTURE
    '
        Dim tblDes As TableDef
        Dim tblSrc As TableDef
        Dim i

        Set tblSrc = dbEngine(0)(0).tableDefs("Customers")
        Set tblDes = db.CreateTableDef("Customers")
        For i = 0 To tblSrc.Fields.Count - 1
            tblDes.Fields.Append tblDes.CreateField(tblSrc.Fields(i).NAME,
tblSrc(i).Type, tblSrc(i).Size)


        Next i
        db.tableDefs.Append tblDes


End Sub
```

LAB3:สร้าง TABLE และข้อมูล โดยใช้คำสั่ง DOCMD

ใช้ขบวนการเดิม แต่การสร้างฐานข้อมูล ใช้คำสั่ง DOCMD แทน

1. COPY FORM เดิมของ LAB 1
2. บรรจุ CODE ต่อไปนี้แทน

```
Private Sub cmdBackup_Click()

        Dim db As Database
        Dim dbName

        On Error Resume Next
        MkDir Me.txtPath

        dbName = txtPath & "\" & Format(Me.txtDate, "yyyymmdd") & ".MDB"
        Kill dbName
        On Error GoTo 0

        Set db = dbEngine(0).CreateDatabase(txtPath & "\" & Format(Me.txtDate,
"yyyymmdd"), dbLangGeneral)

        DoCmd.CopyObject dbName, , acTable, "Customers"

End Sub
```

LAB4:สร้าง WEB PAGE จาก TABLE ที่กำหนด

สร้างโปรแกรมเสริมจากการสร้าง WEBPAGE เดิม โดยให้สามารถสร้าง WEB PAGE TABLE ใดก็ได้

1. สร้าง FORM หนึ่ง FORM
2. สร้าง LISTBOX สำหรับ แสดงรายการ TABLE ในระบบ ให้ชื่อว่า TABLELIST
3. บรรจุโปรแกรมในการดึงรายชื่อ TABLE ในระบบ

```
Private Sub Form_Load()
        Dim i, strx

        For i = 0 To DBEngine(0)(0).TableDefs.Count - 1
            strx = strx & DBEngine(0)(0).TableDefs(i).Name & ";"
```

```
            Next i
            Me.TABLELIST.RowSource = strx

    End Sub
```

4. สร้าง SUB ROUNTINE ในการสร้าง WEBPAGE ชื่อ tblTOWEB มี Parameter เป็นชื่อ TABLE และชื่อ FILE HTML ตามตัวอย่าง

```
Sub tblToWeb(TblName, FileName)

    Dim rst As Recordset
    Dim i, filex

    Set rst = dbEngine(0)(0).OpenRecordset(TblName)
    filex = FreeFile

    Open FileName For Output As filex
    Print #filex, "<HTML>"
    Print #filex, "<BODY BGCOLOR = WHITE>"
    Print #filex, "<Font Size = +4>" & TblName & " LIST</font><br>"
    Print #filex, "<Font Size = +1>Generated from VBA Since: " & Now() & "
</font><br>"
    Print #filex, "<TABLE>"
    Print #filex, "<TR BGCOLOR=#00007F>"

    ' --- Prepare Header
    '
    '

    For i = 0 To rst.Fields.Count - 1
        Print #filex, "  <TD><FONT COLOR=#FFFFFF>" & rst.Fields(i).NAME & "
</FONT></TD>"

    Next i

    Print #filex, "</TR>"


    Do Until rst.EOF
        Print #filex, "<TR BGCOLOR=#F6d6FF>"
        For i = 0 To rst.Fields.Count - 1
            Print #filex, "  <TD><FONT COLOR=#FFFFFF>" & rst.Fields(i).Value &
"</FONT></TD>"
        Next i
        Print #filex, "</TR>"
        rst.MoveNext
    Loop
    Print #filex, "</TABLE>"
    Print #filex, "</HTML>"

    Set rst = Nothing
    Close


End Sub
```

5. สร้าง COMMAND สำหรับทำการสร้าง WEBPAGE แล้วบรรจุ CODE ตามตัวอย่าง

```
    Private Sub Command34_Click()

            If IsNull(Me.TABLELIST) Then
               MsgBox "SELECT TABLE NAME FIRST !", vbCritical
               Exit Sub
            End If
            Call tblToWeb(Me.TABLELIST, "C:\tmp.htm")
    End Sub
```

บทที่ 6 SQL

LAB1:สร้าง Function DXXX เอง

Function Dsum, Dlookup ,.. เป็น Function ที่ภายในใช้ SQL Command ในการประมวลผล ทดลองทำ Function ดังกล่าวดังนี้ มีลักษณะคล้ายกับ Function DLOOKUP

1. สร้าง FORM เปล่า
2. สร้าง TEXTBOX 5 ตัว ให้ชื่อว่า txtVALUE, txtSOURCE, txtWHERE,txtSQL,txtRESULT
3. สร้าง Sub Routine ชื่อ DXXX มี Parameter dValue, dSource, dWhere

```
Function DXXX(dValue, dSource, Optional dWhere)

    On Error GoTo err_dxxx
    Dim strx
    Dim rst As Recordset
    DXXX = Null
    strx = "Select " & dValue & " From " & dSource
    If Not IsNull(dWhere) Then
       strx = strx & " WHERE " & dWhere
    End If
    Me.txtSQL = strx
    Set rst = dbEngine(0)(0).OpenRecordset(strx, dbOpenSnapshot, dbForwardOnly)
    If rst.RecordCount > 0 Then
       DXXX = rst(0)
    End If

resume_dxxx:
    Exit Function
err_dxxx:
    DXXX = Error

    Resume resume_dxxx

End Function
```

4. บรรจุ CODE ลงใน TEXTBOX แต่ละตัว ตามตัวอย่าง

```
    Private Sub txtSource_AfterUpdate()
            Me.txtResult = DXXX(Me.txtValue, Me.txtSource, Me.txtWhere)
    End Sub
    Private Sub txtValue_AfterUpdate()
            Me.txtResult = DXXX(Me.txtValue, Me.txtSource, Me.txtWhere)
    End Sub
    Private Sub txtWhere_AfterUpdate()
            Me.txtResult = DXXX(Me.txtValue, Me.txtSource, Me.txtWhere)
    End Sub
```

LAB2:ทำ Function Update Record ตามเงื่อนไข

ทดลองทำ FUNCTION ที่สามารถ UPDATE TABLE ที่ต้องการ โดยระบุ ชื่อ TABLE , UPDATE, และเงื่อนไข

1. สร้าง FORM เปล่า
2. สร้าง TEXTBOX 5 ตัว ให้ชื่อว่า txtTable, txtUpdate, txtWHERE,txtSQL,txtRESULT
3. สร้าง Sub Routine ชื่อ DXXX มี Parameter dValue, dSource, dWhere

```
Sub dUpdate(dTable, dUpdate, Optional dWhere)

    On Error GoTo err_dUpdate
    Dim strx
    Dim rst As Recordset
    Me.txtResult = ""

    strx = "Update " & dTable & " Set " & dUpdate
    If Not IsNull(dWhere) Then
       strx = strx & " WHERE " & dWhere
    End If
    Me.txtSQL = strx
    dbEngine(0)(0).Execute strx

resume_dUpdate:

    Exit Sub
err_dUpdate:
    Me.txtResult = Error

    Resume resume_dUpdate

End Sub
```

4. บรรจุ CODE ลงใน TEXTBOX แต่ละตัว ตามตัวอย่าง

```
Private Sub txtTABLE_AfterUpdate()
        Call dUpdate(Me.txtTABLE, Me.txtUPDATE, Me.txtWhere)
End Sub
Private Sub txtUPDATE_AfterUpdate()
        Call dUpdate(Me.txtTABLE, Me.txtUPDATE, Me.txtWhere)
End Sub
Private Sub txtWhere_AfterUpdate()
        Call dUpdate(Me.txtTABLE, Me.txtUPDATE, Me.txtWhere)
End Sub
```

ทดลองทำ Ddelete เพื่อใช้ในการลบข้อมูลอัตโนมัติ แบบเดียวกับการทำ dUPDATE

LAB3: BACKUP ข้อมูลตามวันอย่างมีเงื่อนไข

ตัวอย่างนี้เราจะใช้ SQL COMMAND ในการแยกข้อมูลจาก TABLE ORDERS เป็นไฟล์ๆ แต่ละไฟล์ MDB จะมีข้อมูลของ ORDERS ตามเดือนนั้นๆ ชื่อไฟล์ MDB ที่ทำ BACKUP จะเป็น FORMAT YYYYMM ตาม ปีเดือนของข้อมูล มี STEP ในการเขียนโปรแกรมดังนี้

- สร้าง DIRECTORY ที่เก็บ MDB
- สร้าง RECORDSET ที่เก็บปี และเดือนของข้อมูลที่มีใน TABLE ORDERS ทั้งหมด
- LOOP เข้าใน RECORDSET แต่ละ RECORD
- สร้าง MDB ตาม YYYYMM
- ใช้คำสั่ง SELECT INTO เพื่อสร้างข้อมูลลงในปลายทาง

1. สร้าง FORM เปล่า และ ปุ่ม Command
2. ใส่ Code ลงใน Command_Click
3. บันทึก Code ของโปรแกรม GenBackup

```
Private Sub Command25_Click()
```

```
        Call GenBackup
End Sub


Sub GenBackup()

    On Error Resume Next
    MkDir "C:\tmpBackup"
    On Error GoTo 0

    Dim rst As Recordset
    Dim i, x, tmpName
    Dim db As Database

    Set rst = dbEngine(0)(0).OpenRecordset("select Format(orderDate,'YYYYMM')
from Orders Group By Format(orderDate,'YYYYMM')")
    rst.MoveLast
    x = SysCmd(acSysCmdInitMeter, "PROCESS", rst.RecordCount)
    rst.MoveFirst
    i = 0

    Do Until rst.EOF

       If rst(0) <> "" Then
            On Error Resume Next
            tmpName = "C:\tmpBackup\" & rst(0) & ".MDB"
            Kill tmpName
            On Error GoTo 0
            Set db = dbEngine(0).CreateDatabase(tmpName, dbLangGeneral)
            dbEngine(0)(0).Execute "Select * INTO Orders IN '" & tmpName & "' "
& _
                                   " From  Orders where
Format(OrderDate,'YYYYMM') ='" & rst(0) & "'"

       End If
       i = i + 1
       x = SysCmd(acSysCmdUpdateMeter, i)
       rst.MoveNext

    Loop
    x = SysCmd(acSysCmdRemoveMeter)



End Sub
```