สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

โปรแกรมช่วยวิเคราะห์คะแนน

Grad Point Analysis Aid Progeam

โดย นายชยพล พงษ์ศิริ รหัส 42015552

เลขหมู่.....
เลขทะเบียน 42257
วัน, เดือน, ปี 16 พ.ค. 2545

.b.....
.i.....

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาอุตสาหกรรมศาสตรบัณฑิต

สาขาวิชาเทคโนโลยีอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2544

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

หัวข้อปริญญานิพนธ์ โปรแกรมช่วยวิเคราะห์คะแนน (Grad Point Analysis Aid Program)

โดย นายชยพล พงษ์ศิริ เลขประจำตัว 42015552

อาจารย์ที่ปรึกษา อาจารย์มนชนก ศรีเสือขาม

ภาควิชา เทคนิคอุตสาหกรรม

ปีการศึกษา 2544

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง อนุมัติให้นับปริญญานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาอุตสาหกรรมศาสตร์บัณฑิต

คณะกรรมการการสอบปริญญานิพนธ์

..... ประธานกรรมการ
(.....)

..... กรรมการ
(.....)

..... กรรมการ
(.....)

..... กรรมการ
(.....)

..... กรรมการ
(.....)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โปรแกรมวิเคราะห์คะแนน

Grad Point Analysis Aid Program

โดย นายชยพล พงษ์ศิริ รหัส 42015552

อาจารย์ที่ปรึกษา อาจารย์มนชนก ศรีเสือขาม

ปีการศึกษา 2544

บทคัดย่อ

โครงการนี้ทำขึ้นเพื่อช่วยวิเคราะห์คะแนนของนักศึกษา โดยเขียนจากโปรแกรมภาษา Visual BASIC 6.0 ตัวโครงการนี้ จะมีลักษณะเป็น GUI (Graphic User Interface) สามารถแสดงผลเป็นข้อมูลทางสถิติที่มีความรวดเร็วและแม่นยำสูง เพื่อนำไปวิเคราะห์ หาจุดที่เหมาะสมในการตัดระดับคะแนนนักศึกษา และยังมีการดึงเอาข้อมูลจากฐานข้อมูลมาใช้ เพื่อความสะดวกรวดเร็วมากขึ้น โดยสามารถติดต่อฐานข้อมูลได้ 2 ชนิดคือ Microsoft Access และ Microsoft Excel เพราะว่าทั้ง 2 โปรแกรมเป็นโปรแกรมที่แพร่หลายของผู้ใช้ทั่วไป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Grad Point Analysis Aid Program

BY MR.CHAYAPOL PONGSIRI

ADVISER MISS.MONCHANOK SRISUAKAM

YEAR 2001

ABSTRACT

This project makes up for point analysing of student. By writing from Visual BASIC 6.0 program. This project program has a feature GUI; Graphic User Interface which show about data of stasitic. The output data has a high rapaid and right for brings to analysis and finds appreciate point in get point level of student and should be connect with database system for get data to use in the program which it has a comfortable and quickly too. The program can connect to database 2 types that are Microsoft Access and Microsoft Excel.Because, both applications program are popular with general user.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้ สำเร็จลุล่วงไปด้วยดีทั้งส่วนเอกสารคู่มือ โปรแกรม สถานที่ ก็เลังใจ ทั้งนี้ต้องขอบพระคุณอาจารย์มนชนก ศรีเสือขาม เป็นอย่างสูงที่ให้คำแนะนำและคำปรึกษาไว้ ณ ที่นี้ด้วย และขอบใจเพื่อนๆ ที่สนับสนุนด้วย

ผู้จัดทำ

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญรูป

หน้า

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 1
บทนำ

ในปัจจุบันนั้นคอมพิวเตอร์เข้ามามีบทบาทในชีวิตประจำวัน อาจเป็นเหมือนกิจวัตรประจำวันของบางคนเลยก็ได้ เช่น ตื่นเช้ามาก่อนออกจากบ้านไปทำงานก็ต้องรีบหยิบ "Note Book" (Computer) ติดตัวไปด้วย แทนสมัยก่อนที่หยิบ "Note book" (สมุดโน๊ต) แล้วอาจยังต้องมาเลือกว่าเล่มไหนอีกหรือต้องเอาไปหลายๆ เล่ม หรือในการ "Present" งานแทนที่จะนำแผ่นใสเป็นจำนวนมากพกติดตัว เราก็สามารถ Save เป็น File ใส่แผ่น Disket, ZIP Drive หรือ CD-ROM ได้ ลดภาระในการเดินทางทำให้เกิดความคล่องตัว ข้อมูลจะรั่วไหลหรือสูญหายก็เป็นไปได้ยาก และที่สำคัญที่สุดของประโยชน์ของ Computer คือการประมวลผล (Process) ซึ่งการประมวลผลของคอมพิวเตอร์นั้นมีความแม่นยำและรวดเร็วกว่ามนุษย์หลายเท่านัก เช่นในการคำนวนระยะห่างระหว่างดวงดาวหรือสมการที่ยากซับซ้อนที่มนุษย์ อาจคิดออกในเวลา 20 – 30 ชั่วโมง แต่ในคอมพิวเตอร์ระดับสูงนั้นอาจใช้เวลาเพียงไม่กี่นาทีเท่านั้น และในโลกยุคปัจจุบัน หน่วยประมวลผล (CPU) ก็ยิ่งทวีความเร็วสูงขึ้นมาก และด้วยประโยชน์ที่กล่าวมาแล้วนี้ การเขียนโปรแกมขึ้นเพื่อใช้กับคอมพิวเตอร์นั้นทำให้เราได้ใช้ประโยชน์จาก คอมพิวเตอร์อย่างสูงสุด เช่น โปรแกรมประเภทที่ต้องใช้การวิเคราะห์ทางสถิติเพื่อหาค่าต่างๆ นั้นในกรณีที่ข้อมูลน้อยๆ มันก็อาจจะดูง่ายกว่า แต่ถ้าในกรณีที่ข้อมูลมีจำนวนมากลองคิดดูว่ามันจะวุ่นวายขนาดไหน เช่นหาค่าโดยไม่ใช้โปรแกรมผลรวมครั้งที่ 1 ได้ 121,213 จากข้อมูล 100 ตัว ครั้งที่ 2 ได้ 121,211 เราจะเห็นได้ว่ามีข้อมูลผิดพลาดที่ทำการบวกผิดแต่เราก็จะไม่รู้เลยว่าตัวไหนที่เราบวกผิดทำให้เราต้องบวกอีกครั้ง เพื่อหาข้อสรุปและถ้าครั้งที่ 3 ไม่ตรงกับ 2 ครั้งแรกเราก็คงจะทำอีก ทำให้เสียเวลาและไม่มีความแม่นยำ

ดังนั้นเพื่อความรวดเร็ว แม่นยำ ในการใช้ข้อมูลจำนวนมากเช่นการตัดเกรดนักศึกษาจึงควรใช้โปรแกรมคอมพิวเตอร์ในการวิเคราะห์คะแนน จึงได้เกิดโครงงานนี้ขึ้น ใช้ชื่อว่า โปรแกรมช่วยวิเคราะห์คะแนน (Grad Point Analysis Aid Program) ทำการวิเคราะห์หาค่าต่างๆ เพื่อใช้ประกอบการตัดสินใจของผู้ที่เป็นผู้ตัดเกรดนั่นเอง

1.1 วัตถุประสงค์ของปริญญานิพนธ์

1. เพื่อศึกษาหาลักษณะโปรแกรมที่สมควรใช้ ในการเขียนโปรแกรม
2. เพื่อศึกษาตัวโปรแกรมที่นำมาใช้ในการเขียนโปรแกรม
3. เพื่อศึกษาการทำงาน, ผลลัพธ์ และส่วนประกอบของโปรแกรมที่ทำการเขียนมาแล้ว

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

1.2 เนื้อหาปริญญานิพนธ์

ปริญญานิพนธ์ฉบับนี้โดยรวมมีเนื้อหาแบ่งแยกออกเป็น บทต่างๆ ดังนี้

ในบทที่ 2 กล่าวถึงทฤษฎีในการเลือกโปรแกรมมาใช้งาน และการใช้งานโปรแกรมรวมทั้งข้อดีและประโยชน์ของโปรแกรมที่นำมาใช้

ในบทที่ 3 กล่าวถึงระบบฐานข้อมูล Database Management System ซึ่งเป็นส่วนเพิ่มเติมเข้ามา เพื่อที่จะให้สามารถใช้ข้อมูลที่มีอยู่มาประกอบการตัดสินใจ หรือวิเคราะห์คะแนน รวมทั้งทฤษฎีการนำไปใช้ และทฤษฎีขององค์ประกอบต่างๆ ของโปรแกรมเช่นการสั่งพิมพ์ หรือการทำเป็นโปรแกรม เพื่อนำไปติดตั้งบน PC (Personal Computer) เครื่องอื่นๆ

ในบทที่ 4 กล่าวถึงการออกแบบการทำงานของโปรแกรม,ส่วนติดต่อกับผู้ใช้ (User Interface) และส่วนแสดงผลต่างๆ

ในบทที่ 5 เป็นการสรุปผลการทำงาน, ปัญหา, ข้อแก้ไข, ข้อดีข้อเสีย, โดยรวมของโปรแกรม

1.3 ประโยชน์ที่ได้รับจากปริญญานิพนธ์

1. สามารถนำทฤษฎีทั้งหมดที่มี ไปใช้ในงานจริงได้ Diskette
2. เป็นแบบอย่างของการใช้โปรแกรมในการวิเคราะห์คะแนนเพื่อแก้ไขดัดแปลง, ปรับปรุงหรือใช้ในงานจริงได้
3. เกิดทักษะในการเขียนโปรแกรม รู้แนวทางการเขียนและการออกแบบโปรแกรม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 2
การเขียนโปรแกรมด้วย Visual Basic

2.1 แนวความคิดเบื้องต้นของการเขียนโปรแกรมคอมพิวเตอร์

เหตุผลสำคัญสองประการ ในการศึกษาการทำงาน ของภาษาคอมพิวเตอร์ ในฐานะ ตัวแปรภาษา (Compiler and interpreter)

1. ถ้าหากเรารู้ว่าตัวแปรภาษาคอมพิวเตอร์นั้นทำงานอย่างไรทำให้เราเข้าใจและหาสาเหตุของการเกิดข้อผิดพลาด (Error) ได้รวดเร็วขึ้น และเมื่อเรารู้สาเหตุของข้อผิดพลาดนั้นแล้วก็ทำให้การแก้ไขข้อผิดพลาดดังกล่าวเป็นไปอย่างสะดวกและง่ายดายยิ่งขึ้นเช่นกัน

2. ถ้าหากเราฝึกฝนตัวเรา ให้คิดอย่างตัวแปรภาษาได้ จะทำให้เราเข้าใจ และจดจำกฎเกณฑ์ไวยากรณ์ต่างๆ ของภาษาคอมพิวเตอร์ได้ง่ายขึ้น

ภาษาคอมพิวเตอร์ที่นิยมใช้ในการเขียนโปรแกรมนั้น มีอยู่ด้วยกันหลายภาษา ซึ่งแบ่งออกได้เป็น 2 พวกคือ ภาษาระดับต่ำ (low-level Language) กับภาษาระดับสูง (High-Level Language)

ภาษาระดับต่ำที่สุดก็คือ ภาษาแอสเซมบลี (Assembly) หมายถึง ภาษาที่มีความใกล้ชิดกับระดับฮาร์ดแวร์ของคอมพิวเตอร์มากที่สุด ซึ่งมีไวยากรณ์ และประโยคโปรแกรมของภาษาที่ค่อนข้างยุ่งยากซับซ้อน แต่ก็เป็นภาษาที่มีประสิทธิภาพ ในการประมวลผลสูงที่สุด เพราะใกล้เคียงกับภาษาเครื่อง (Machine Code) มากที่สุด ภาษาที่มีระดับสูงขึ้นมาอีกขั้นหนึ่งและเป็นภาษาที่นิยมใช้ก็คือภาษาซี (C Language), ภาษา C++, ภาษาปาสคาล (Pascal) เป็นต้น ส่วนภาษาระดับสูงก็อย่างเช่น Visual Basic ซึ่งถูกพัฒนาขึ้นจากภาษา C และ BASIC อีกต่อหนึ่ง

อย่างไรก็ตามภาษาคอมพิวเตอร์ทุกภาษานั้นมีรูปแบบการใช้งานและหลักไวยากรณ์ (Syntax) เป็นของตนเอง และจะต้องอาศัยตัวแปรภาษาของภาษานั้นๆ เพื่อที่จะแปลงโค้ดโปรแกรมภาษาของตนเองให้กลายเป็น ภาษาเครื่อง ที่เครื่องคอมพิวเตอร์สามารถนำไปทำงานได้จริงและตัวแปรภาษา ดังกล่าวก็มีอยู่ 2 แบบด้วยกันคือ ตัวแปรภาษาแบบ คอมไพเลอร์ (Compiler) กับแบบ อินเตอร์เพรตเตอร์ (Interpreter)

2.2 หลักการทำงานของอินเตอร์เพรตเตอร์

โปรแกรม DOS ทั่วไปในยุคก่อนนั้นจะมีตัวแปรภาษาเบื้องต้นที่ชื่อว่า Microsoft GWBasic ซึ่งเป็นตัวแปรหนึ่งภาษาหนึ่งของภาษา BASIC ให้เราได้ใช้ในการเขียนภาษาโปรแกรมอย่างง่าย อย่างเช่น Batch File เป็นต้น การเขียนโปรแกรมด้วยตัวแปรภาษาแบบอินเตอร์เพรตเตอร์จะเกี่ยวข้องกับซอฟต์แวร์อยู่ 3 ระดับ ดังรูปที่ 2.1

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กล่าวได้ว่าซอฟต์แวร์ ที่ตั้งอยู่บนฮาร์ดแวร์ ดังรูปที่ 2.1 ทำให้เครื่องคอมพิวเตอร์ เป็นอุปกรณ์ที่มีศักยภาพ ในการทำงานสูง เราสามารถสังเกตได้ว่า เมื่อเราเปิดเครื่องคอมพิวเตอร์ขึ้นมา เพื่อใช้งานซอฟต์แวร์เหล่านี้จะถูกโหลดเข้าสู่ คอมพิวเตอร์ตั้งแต่ข้างล่างสู่ข้างบนตามลำดับ โดยขั้นแรก ระบบ- ปฏิบัติการ (Operating System) จะถูกโหลดเข้าสู่หน่วยความจำก่อนโดยอัตโนมัติ (ในตัวอย่างนี้คือ DOS) เมื่อเราต้องการเขียนโปรแกรมด้วยตัวแปรภาษา Microsoft GWBasic เราก็ต้องโหลด อินเตอร์เพรตเตอร์ (GWBasic.EXE) เข้าสู่หน่วยความจำ และเมื่อ อินเตอร์เพรตเตอร์ถูกโหลดเข้าสู่หน่วยความจำเรียบร้อยแล้ว เราจึงสามารถเริ่มเขียนโปรแกรมของเราได้ (Project.BAS)

รูปที่ 2.1 ระดับชั้นในการทำงานของโปรแกรมที่อาศัยอินเตอร์เพรตเตอร์ในการแปรภาษา

เมื่อเราต้องการใช้งานโปรแกรม Project.BAS เราสามารถใช้คำสั่ง RUN ของอินเตอร์เพรตเตอร์นั้นๆ เพื่อสั่งให้โปรแกรมเริ่มทำงานได้สิ่งที่เราต้องระลึกไว้เสมอคือ ตัวอินเตอร์เพรตเตอร์จะต้องถูกโหลด เข้าสู่หน่วยความจำ ก่อนที่เราจะเรียกโปรแกรมที่เขียนด้วยตัวแปรภาษาชนิดนี้ ขึ้นมาทำงานได้ เหตุผลก็คือว่า อินเตอร์เพรตเตอร์ รับหน้าที่เป็นผู้ที่คอยจัดการ และควบคุมการติดต่อระหว่างโปรแกรมของเรากับระบบปฏิบัติการ สมมติว่าเราเขียนโปรแกรมขึ้นมาโปรแกรมหนึ่ง โดยใช้ชื่อว่า Project.BAS ด้วยตัวแปรภาษา GWBasic และเราจะใช้โค้ดโปรแกรมข้างล่างนี้ เป็นตัวอย่างเพื่อประกอบคำอธิบาย

```
10 FOR J = 1 TO 10
20 PRINT J;
30 NEXT J
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

หลังจากที่เราพิมพ์ข้อความดังกล่าวนี้แล้ว เราก็ใช้คำสั่ง RUN เพื่อให้โปรแกรมทำงาน จากการที่อินเตอร์เพรตเตอร์จะเริ่มทำงานตั้งแต่บรรทัดที่ต่ำที่สุดเสมอ ดังนั้นบรรทัดที่ 10 จะเป็นบรรทัดแรก ในตัวอย่างนี้ สิ่งที่อินเตอร์เพรตเตอร์เห็นเป็นอันดับแรกคือคำว่า FOR (Keyword of GWBasic) นั่นเอง

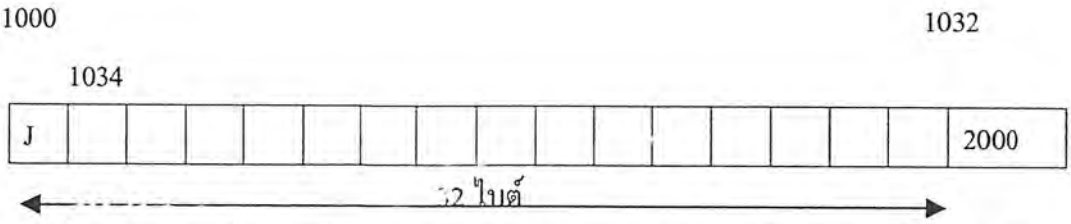
ในตัวอย่างข้างต้น FOR เป็นคีย์เวิร์ด ที่ถูกใช้สำหรับการสร้าง Loop ในขั้นต้น BASIC ย่อมไม่รู้ว่า FOR เป็นคีย์เวิร์ดดังนั้นเมื่อมันเจอคำใดๆ ก็ตาม BASIC ก็จะเริ่มค้นหาในรายชื่อคีย์เวิร์ดที่มีอยู่ว่ามีคำว่า FOR หรือไม่ถ้าหาก อินเตอร์เพรตเตอร์ไม่พบคำว่า FOR จากรายชื่อคีย์เวิร์ด ก็จะฟ้องข้อผิดพลาดขึ้นมาว่า Syntax Error ซึ่งหมายความว่า โค้ดโปรแกรมที่ถูกเขียนขึ้น ไม่เป็นไปตามหลักไวยากรณ์ของภาษานั้นๆ

คีย์เวิร์ดแต่ละตัว ในรายชื่อคีย์เวิร์ด ของตัวแปรภาษาใดๆ จะมีกฎเกณฑ์ และไวยากรณ์ ของตัวมันเอง ตามหลักไวยากรณ์ของคำว่า FOR ใน BASIC จะต้องตามด้วยการกำหนดค่าตัวแปรดังนั้น อินเตอร์เพรตเตอร์ ก็จะมองหาตัวแปรที่ตามหลังคำว่า FOR ซึ่ง อินเตอร์เพรตเตอร์ก็จะพบ ประโยคโปรแกรมว่า J = 1 และพบว่าทุกอย่างไปตามหลักไวยากรณ์ที่ถูกต้อง

จากนั้น อินเตอร์เพรตเตอร์ก็จะเริ่มตรวจสอบต่อไปว่าตัวแปรที่ชื่อ J เคยปรากฏในโปรแกรมแล้วหรือไม่ ก่อนหน้านี้ การตรวจสอบที่ว่านี้ เกิดขึ้นโดย อินเตอร์เพรตเตอร์จะไปค้นหาจาก ตารางสัญลักษณ์ (Symbol Table) ซึ่งหมายถึง หน่วยความจำที่ถูก อินเตอร์เพรตเตอร์หรือคอมไพเลอร์ใช้สำหรับเก็บข้อมูลต่างๆ เกี่ยวกับตัวแปรและข้อมูลอื่นๆ ที่เกี่ยวข้องกับการใช้งานโปรแกรมนั้นๆ

ในกรณีนี้ตารางสัญลักษณ์ยังคงว่างอยู่ เพราะยังไม่เคยมีตัวแปรใดถูกใช้มาก่อนหน้านี้เลย เมื่อ อินเตอร์เพรตเตอร์พบว่า ตัวแปรนี้เป็นตัวแปรใหม่ ที่ถูกใช้เป็นครั้งแรก อินเตอร์เพรตเตอร์ก็จะจัดการเก็บข้อมูลของตัวแปร (J) ลงในตารางสัญลักษณ์ของตน การที่จะทำเช่นนี้ได้ อินเตอร์เพรตเตอร์ก็จะต้องหาตำแหน่งที่อยู่ในหน่วยความจำ เพื่อจะเก็บข้อมูลดังกล่าวนี้เสียก่อน ซึ่งเป็นหน้าที่ของ อินเตอร์เพรตเตอร์เอง ที่จะค้นหาพื้นที่หน่วยความจำ ที่ว่างและจัดการบันทึกข้อมูลเหล่านี้ ไว้ในหน่วยความจำ เราอาจแสดงรูปแบบคร่าวๆ ของการเก็บข้อมูลตัวแปรลงในตารางสัญลักษณ์ได้ดังรูปที่ 2.2

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.2 ภาพจำลองของหน่วยความจำในการสร้างตารางสัญลักษณ์ของอินเตอร์เพรตเตอร์

หากว่าโปรแกรม BASIC ที่เราใช้งานอยู่นี้ อนุญาตให้เราตั้งชื่อตัวแปร ได้สูงสุด 32 ตัวอักษร ดังนั้น เนื้อที่ว่างสำหรับแต่ละตัวแปร ในตารางสัญลักษณ์ จะต้องใหญ่พอสำหรับการจัดเก็บข้อมูล 32 ตัวอักษร จากภาพประกอบเราสมมติว่า ตำแหน่งที่อยู่ในหน่วยความจำของตารางสัญลักษณ์เริ่มต้นขึ้นที่ตำแหน่ง 1000 โดยที่ตัวแปรแต่ละตัวมีความยาวสูงสุด 32 ตัวอักษร (ตำแหน่งหน่วยความจำ 1000 ถึง 1031) และสมมติว่าตัวแปร J ในตัวอย่างนี้ถูกจัดเก็บไว้ที่ตำแหน่ง 2000 และใช้พื้นที่หน่วยความจำเพียง 2 ไบต์ในตารางสัญลักษณ์สำหรับเก็บตำแหน่งที่อยู่ของตัวแปร J ในหน่วยความจำ ดังนั้น ตัวแปรตัวต่อไปที่จะถูกจัดเก็บเข้าสู่ตารางสัญลักษณ์จะต้องเริ่มบันทึกที่ตำแหน่ง 1034

หลังจากที่ อินเตอร์เพรตเตอร์ตรวจสอบไวยากรณ์ภาษา ในบรรทัดแรกจนพบตัวแปร J จากนั้นจัดการหาพื้นที่หน่วยความจำ สำหรับเก็บข้อมูลของตัวแปร J และจัดเก็บลงในตารางสัญลักษณ์เป็นที่เรียบร้อยแล้ว อินเตอร์เพรตเตอร์ ก็จะจัดการ กำหนดค่าตัวแปร ตามที่โค้ดโปรแกรมถูกเขียนขึ้นมาคือ กำหนดค่า 1 ให้กับตัวแปร J

การที่จะกระทำเช่นนั้นได้ อินเตอร์เพรตเตอร์จะเริ่มที่การตรวจสอบตารางสัญลักษณ์ ว่าตำแหน่งที่อยู่ที่ใช้สำหรับเก็บค่าของตัวแปร J นั้นอยู่ที่ไหน โดยหาข้อมูลจากตารางสัญลักษณ์ในส่วนที่เก็บตำแหน่งที่อยู่ของตัวแปร J (ในตัวอย่างนี้คือ 1032 และ 1033) ก็จะพบว่าตำแหน่งที่อยู่ของตัวแปร J อยู่ที่ 2000 ดังนั้น อินเตอร์เพรตเตอร์ก็จะไปยังตำแหน่งที่อยู่ 2000 นี้และจัดการบันทึกค่า 1 เข้าไปเป็นค่าของตัวแปร J

จากนั้น อินเตอร์เพรตเตอร์ก็จะตรวจสอบโค้ดโปรแกรมต่อไปจนพบคำว่า TO ซึ่ง อินเตอร์เพรตเตอร์ก็ทำเช่นเดิมอีกคือค้นหารายชื่อคีย์เวิร์ดว่ามีคำว่า TO หรือไม่ เมื่อพบว่ามีคีย์เวิร์ด TO อยู่ในรายชื่อคีย์เวิร์ด อินเตอร์เพรตเตอร์ก็รู้อีกว่าหลักไวยากรณ์ของ TO นั้นจะต้องตามด้วยค่าๆ หนึ่งที่ใช้สำหรับการวนวงรอบการทำงานซ้ำ (Loop) ในการทำงาน ดังนั้นอินเตอร์เพรตเตอร์ก็จะหาพื้นที่ว่างในหน่วยความจำเพื่อเก็บค่า 10 นี้ไว้สำหรับใช้ในการวนวงรอบการทำงานซ้ำ ถึงตรงนี้การแปรภาษาในบรรทัดแรกของโค้ดตัวอย่างนี้ก็จะเสร็จสิ้น ในบรรทัดต่อมาคือบรรทัดที่ 20 อินเตอร์เพรตเตอร์จะพบคำว่า PRINT และตามด้วยชื่อตัวแปร J ดังจะเห็นได้ว่าค่าของตัวแปร J นั้นถูกใช้โดยคำ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สั่ง PRINT ซึ่งเป็นคำสั่งที่จะนำค่าของตัวแปร J แสดงสู่หน้าจอ เราเรียกฐานะของตัวแปร J ในขณะนี้ว่าเป็นอาร์กิวเมนต์ (Argument) ของ PRINT

อาร์กิวเมนต์ (Argument) หมายถึง หน่วยข้อมูลใดๆ ที่ถูกกำหนดให้ถูกใช้ในการทำงานของคีย์เวิร์ด ซับรูทีน ฟังก์ชัน หรือ โพรซีเยอร์ ต่างๆ

เนื่องจากคำว่า PRINT ก็เป็นคีย์เวิร์ดหนึ่งของภาษา BASIC และเมื่อตรวจสอบตารางสัญลักษณ์ พบว่าค่าของตัวแปร J ถูกเก็บอยู่ในตารางสัญลักษณ์เรียบร้อยแล้ว ดังนั้น อินเตอร์เพรตเตอร์จะรับรู้ว่าประโยคโปรแกรมในบรรทัดที่ 20 นั้นสมบูรณ์ถูกต้อง อินเตอร์เพรตเตอร์ก็จะส่งผ่านการควบคุมไปยังอีกส่วนหนึ่งของ อินเตอร์เพรตเตอร์ที่ทำหน้าที่สร้างโค้ดที่จำเป็นสำหรับการเรียกใช้งานคำสั่ง PRINT

ในบรรทัดที่ 30 อินเตอร์เพรตเตอร์จะพบคำว่า NEXT หลังจากที่ตรวจสอบแล้วก็พบว่ามันเป็นคีย์เวิร์ดหนึ่งของภาษา BASIC จากนั้นก็ตรวจสอบอีกว่ามีตัวแปร J อยู่ในตารางสัญลักษณ์หรือไม่เมื่อทุกอย่างได้รับการยืนยันว่าถูกต้อง ประโยคโปรแกรมในบรรทัดที่ 30 ก็จะสมบูรณ์

จากนั้น อินเตอร์เพรตเตอร์ก็จะตรวจสอบว่าค่าตัวแปร J มีค่ามากกว่า 10 หรือไม่ ถ้าหากค่าของตัวแปร J มีค่าน้อยกว่า 10 อินเตอร์เพรตเตอร์ ก็จะไปยังตำแหน่ง 2000 ซึ่งเป็นตำแหน่งที่อยู่ที่เก็บบันทึกค่าของตัวแปร J ไว้ และจัดการเพิ่มค่าของตัวแปร J ไปเท่ากับ 1 แล้วก็เริ่ม LOOP FOR ใหม่ อีกครั้งด้วยการย้อนกลับไปทำงานในบรรทัดที่ 20 แต่ถ้าค่าของตัวแปร J มากกว่า 10 การทำงานของโปรแกรมก็จะสิ้นสุด

เราจะเห็นได้ว่า อินเตอร์เพรตเตอร์ใช้เวลาเกือบทั้งหมด สำหรับ

1. ตรวจสอบโค้ดโปรแกรมในแต่ละบรรทัด
2. ค้นหาและเปรียบเทียบระหว่างข้อความที่พบในโค้ดโปรแกรมกับรายชื่อคีย์เวิร์ดที่มีอยู่
3. ค้นหาข้อมูลเกี่ยวกับตัวแปรจากตารางสัญลักษณ์

เราสามารถบอกได้ว่าอินเตอร์เพรตเตอร์ใช้เวลาไปกับ 3 ประการข้างต้นนี้ มากกว่าการประมวลผลโค้ดโปรแกรม ที่เราเขียนขึ้นเสียอีก ผลลัพธ์ของมันก็คือ ทำให้การทำงานของโปรแกรมที่เขียนขึ้นด้วนอินเตอร์เพรตเตอร์ ช้ากว่าการทำงานของ โปรแกรมที่เขียนขึ้นสำหรับกรณีที่ใช้คอมไพเลอร์เป็นตัวแปรภาษา

2.3 ข้อแตกต่างระหว่างตัวแปรภาษาแบบ อินเตอร์เพรตเตอร์กับคอมไพเลอร์

ในการใช้งานโปรแกรม ที่ถูกเขียนขึ้นมาด้วย อินเตอร์เพรตเตอร์นั้น โปรแกรม จะอาศัยโค้ดและทรัพยากรอื่นๆ ที่ติดตั้งมาพร้อมกับตัวอินเตอร์เพรตเตอร์ในการทำงานทำให้โปรแกรมที่ถูกเขียน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

แล้วแปรภาษาด้วยอินเตอร์เพรตเตอร์ ต้องมีอินเตอร์เพรตเตอร์โหลดอยู่ ในหน่วยความจำในขณะนั้นด้วย โปรแกรมจึงจะสามารถทำงานได้

แต่สำหรับคอมไพเลอร์ ได้มีการจัดการสร้างภาษาเครื่องต่าง ๆ ที่จำเป็นในการทำงานของโค้ดโปรแกรมแต่ละบรรทัดในไฟล์ออบเจ็กต์โค้ด และลิงเกอร์ก็ได้จัดการเชื่อมโค้ดเหล่านั้นและข้อมูลอื่น ๆ เข้ากับไฟล์ไลบรารีที่โปรแกรมต้องการเป็นที่เรียบร้อย ทำให้ผลลัพธ์ของโปรแกรมที่เขียนแล้วแปรภาษาด้วย คอมไพเลอร์นั้น มีโปรแกรมที่สามารถทำงานได้ด้วยตัวมันเอง (Stand-alone executable program) ผลลัพธ์ที่ได้เป็นไฟล์เอ็กซีคิวต์นี้จะอาศัยข้อมูลและทรัพยากรต่าง ๆ เฉพาะจากระบบปฏิบัติการเพื่อช่วยในการทำงานของโปรแกรมเท่านั้น ดังนั้นจึงไม่จำเป็นที่จะต้องมีโปรแกรมอื่นใด โหลดเข้าสู่หน่วยความจำ นอกจากระบบปฏิบัติการ สำหรับการทำงานของโปรแกรม ที่เราเขียนขึ้นด้วย คอมไพเลอร์

และเนื่องจากว่าไฟล์เอ็กซีคิวต์จากการแปรภาษาด้วย คอมไพเลอร์จะมีเพียงโค้ดโปรแกรมที่จำเป็น สำหรับการทำงานเพียงอย่างเดียว ทำให้ไม่เสียเวลาในการค้นหา และตรวจสอบข้อความต่างๆ ขณะทำงานเหมือน อินเตอร์เพรตเตอร์ ความเร็วในการประมวลผลจึงเรียกได้ว่า มีความรวดเร็วกว่ามาก แต่การเพิ่มขึ้นของความเร็ว ในการประมวลผลนี้ ทำให้ต้องอาศัยเนื้อที่ฮาร์ดดิสก์จำนวนมากในการเก็บข้อมูลต่าง ๆ ที่จำเป็นในการเรียกใช้งานโปรแกรม

ข้อด้อยอีกประการหนึ่งของ คอมไพเลอร์ก็คือ จะเสียเวลาในขั้นตอนการตรวจสอบ และแก้ไขข้อผิดพลาด (Debug) เป็นอันมาก เพราะผู้เขียนโปรแกรมจะรู้ว่า โปรแกรมมีข้อผิดพลาดได้ก็ต่อเมื่อ ทำการ คอมไพเลอร์โปรแกรม และเมื่อพบต้นตอปัญหาแล้วเราก็ต้องทำการแก้ไขโค้ดโปรแกรม จากนั้นก็ต้อง คอมไพเลอร์ซ้ำอีกเที่ยวหนึ่งเพื่อทดสอบการทำงาน และมันจะเป็นเช่นนี้ไปเรื่อย ๆ จนกว่าโปรแกรมจะทำงานได้อย่างสมบูรณ์ ในขณะที่ อินเตอร์เพรตเตอร์สามารถตรวจสอบการทำงานของโปรแกรมได้ตลอดเวลาด้วยคำสั่ง RUN

ภาษาที่ใช้ อินเตอร์เพรตเตอร์เป็นตัวแปลคำสั่งนั้น มักจะมีคำสั่งพื้นฐาน เกี่ยวกับการสั่งให้โปรแกรมหยุดการทำงานชั่วคราว ทำให้เราสามารถใช้ทดสอบการทำงานของโปรแกรม เพื่อตรวจสอบโค้ดโปรแกรมในแต่ละจุดได้ง่าย การตรวจสอบและแก้ไขข้อผิดพลาด รวมทั้งขั้นทดสอบการทำงานของโปรแกรมที่ใช้ อินเตอร์เพรตเตอร์ สามารถทำได้รวดเร็วกว่า คอมไพเลอร์มาก

มีนักเขียนโปรแกรมหลายคนใช้ อินเตอร์เพรตเตอร์ในช่วงการพัฒนาโปรแกรมด้วยเหตุผลที่ว่าสามารถตรวจสอบและแก้ไขข้อผิดพลาดได้ง่ายกว่า เมื่อโค้ดโปรแกรมที่ได้สมบูรณ์แล้ว ก็ค่อยจัดการมันเพื่อให้เป็นไฟล์เอ็กซีคิวต์ ในขณะที่นักเขียนโปรแกรมคนอื่นๆ ก็เลือกใช้ อินเตอร์เพรตเตอร์หรือไม่ก็ คอมไพเลอร์เพียงอย่างเดียว

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ส่วน Project Grad Point Analysis Aid Program ใช้ Visual BASIC 6.0 เป็นภาษาในการเขียน และ Visual BASIC 6.0 ก็เป็นตัวแปรภาษาแบบ อินเตอร์เพรตเตอร์ชนิดหนึ่ง ดังนั้น โปรแกรมของเราที่เขียนและคอมไพล์เป็นไฟล์เอ็กซีคิวต์แล้วก็ยังต้องการไฟล์อีกจำนวนหนึ่งเช่นไฟล์ MSVBVM50.DLL โหลดเข้าสู่หน่วยความจำด้วยในขณะที่เราเรียกโปรแกรมขึ้นมาทำงาน แสดงว่าหากเราต้องการจะเผยแพร่โปรแกรมที่เราเขียนขึ้น ผู้อื่นได้ใช้งาน เราจะต้องรวมเอาไฟล์ต่างๆ ที่จำเป็นไปกับไฟล์ติดตั้งโปรแกรมของเราด้วย และในบางกรณี ก็อาจจะมีไฟล์ชนิดอื่น ๆ อีก เช่น MCI32.OCX ซึ่งเป็นไฟล์ ActiveX Control รวมอยู่ด้วย หากว่าโปรแกรมของเรามีการเรียกใช้งานคอนโทรลชนิดนั้น

2.4 เข้าใจหลักการพื้นฐานของ Visual BASIC

แม้ Visual BASIC จะเป็นตัวแปรภาษาแบบ อินเตอร์เพรตเตอร์ แต่ก็เป็นภาษาคอมพิวเตอร์ที่มีศักยภาพสูง และถูกออกแบบมา สำหรับใช้ในการพัฒนาโปรแกรมประยุกต์ ที่ทำงานบนโปรแกรม Windows โดยเฉพาะ ทั้งรูปแบบการใช้งานก็ง่ายและไม่ซับซ้อนเท่าภาษา C++ ที่เป็น คอมไพเลอร์ของภาษาที่นิยมใช้ออกแบบโปรแกรมบน Windows และ Dos ด้วยเช่นเดียวกัน

คำว่า "Visual" นั้นหมายถึง วิธีการที่ใช้ในการสร้าง ส่วนติดต่อกับผู้ใช้แบบกราฟิก (Graphic User Interface – GUI) ส่วนคำว่า "Basic" นั้นหมายถึงภาษาคอมพิวเตอร์ "BASIC" (Beginners all-purpose Symbolic Instruction Code) ซึ่งเป็นภาษาคอมพิวเตอร์ที่นักเขียนโปรแกรมในอดีตใช้กันอย่างแพร่หลายที่สุด Visual Basic เป็นวิวัฒนาการหนึ่งของภาษา BASIC เดิม โดยมีการเพิ่มเติมประโยคโปรแกรม (Statement) ฟังก์ชัน (Function) และ คีย์เวิร์ด (Keyword) ต่าง ๆ เข้าไปมากมาย รวมทั้งการติดต่อกับ GUI โดยตรง เช่น เมนู ปุ่มสั่งงาน เป็นต้น

ข้อดีประการหนึ่งของ Visual Basic ที่มีเหนือภาษาอื่นๆ สำหรับการพัฒนาโปรแกรม บน Windows ก็คือแทนที่เราจะต้องเขียนโค้ดโปรแกรม หลายๆ บรรทัด เพื่อสร้างส่วนติดต่อผู้ใช้ ซึ่ง Visual Basic เรียกมันว่า คอนโทรล (Control) บนสภาพแวดล้อมของ Windows ขึ้นมาชิ้นหนึ่งด้วยโค้ดโปรแกรมหลายสิบบรรทัด ใน Visual Basic เราเพียงแค่เลือกส่วนติดต่อผู้ใช้ที่เราต้องการจากสิ่งที่ Visual Basic เตรียมไว้ให้แล้ว ด้วยวิธีการลากและวาง (Drag-and-Drop) ลงบนหน้าจอ จากนั้นก็กำหนด คุณลักษณะ (Properties) ของส่วนติดต่อผู้ใช้นั้นๆ ตามต้องการแล้วก็เขียนโค้ดโปรแกรมสำหรับคอนโทรลนั้นๆ เพื่อให้โปรแกรมทำงานตามที่เราต้องการก็เป็นอันเรียบร้อย

เนื่องจากว่า Visual Basic ถูกออกแบบมาสำหรับออกแบบโปรแกรมที่ทำงานบน Windows โดยเฉพาะ ดังนั้นหากเราต้องการจะเข้าใจพื้นฐานการใช้ภาษา Visual Basic เราจึงควรที่จะเข้าใจ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

หลักการทำงานเบื้องต้นของโปรแกรม Windows กับโปรแกรมที่ทำงานบน DOS นั้นมีข้อแตกต่างกันอย่างมีนัยสำคัญ

2.5 วินโดว์ ของโปรแกรม เหตุการณ์ และข้อความของวินโดว์ (Window, Event and Messages)

ระบบปฏิบัติการ Windows เป็นโปรแกรมที่ทำงานในโหมดกราฟิกอยู่ตลอดเวลา ดังนั้นตัวระบบปฏิบัติการ Windows 95/98 เอง รวมทั้ง วินโดว์ (Window) ของโปรแกรมต่างก็เป็นเช่นเดียวกัน คือลักษณะเป็นสี่เหลี่ยม มีขอบของตัวมันเอง และผู้ใช้สามารถเคลื่อนย้าย วินโดว์ไปยังตำแหน่งต่างๆ ของหน้าจอได้ ตัวอย่างของ วินโดว์ได้แก่ หน้าจอของโปรแกรม Window Explorer ใน Windows 95/98 หน้าเอกสารของโปรแกรม Microsoft Word หรือหน้าจอของโปรแกรมเกมส์ Minesweeper ทั้งนี้ คำว่า วินโดว์ในความหมายของตัว ระบบปฏิบัติการเองยังหมายถึง วินโดว์ในลักษณะอื่นที่เราไม่คุ้นเคยอีกด้วย อย่างเช่น ปุ่มสั่งงาน (Command Button), ไอคอน (Icon) ซึ่งต่างก็เป็น วินโดว์อีกรูปแบบหนึ่งในความหมายของระบบปฏิบัติการ Windows เช่นเดียวกัน

ด้วยเหตุที่ระบบปฏิบัติการ Windows ต้องจัดการกับวินโดว์ที่แตกต่างกันอยู่มากมาย โปรแกรมวินโดว์จึงต้องสร้างค่าตัวเลขขึ้นมากลุ่มหนึ่งเพื่อใช้สำหรับการอ้างอิงถึง วินโดว์ แต่ละ วินโดว์โดยชุดหนึ่งคือค่าอินสแตนซ์ของโปรแกรม ซึ่งเป็นค่าเฉพาะ ของแต่ละโปรแกรม ที่กำลังทำงานอยู่ขณะนั้น และอีกชุดหนึ่งเรียกว่า ค่าแฮนเดิลของ วินโดว์ (Windows Handle—hWhd) ซึ่งใช้เป็นค่าที่อ้างถึงแต่ละ วินโดว์ของโปรแกรมโดยระบบปฏิบัติการจะตรวจสอบการทำงานของแต่ละ วินโดว์อยู่ตลอดเวลา เพื่อตรวจจับการทำงานและเหตุการณ์ตอบสนอง (Events) ที่เกิดขึ้นกับ วินโดว์นั้นๆ เหตุการณ์ตอบสนองที่ว่านี้ อาจจะเกิดจากการที่ผู้ใช้ คลิกเมาส์หรือกดคีย์บอร์ด หรือเกิดจากการทำงานของวินโดว์อื่นๆ ก็ได้

ทุกครั้งที่มีเหตุการณ์ตอบสนองเกิดขึ้น ก็จะมี ข้อความของวินโดว์ (Windows's Message) ส่งไปยังตัวระบบปฏิบัติการ ระบบปฏิบัติการก็จะจัดการประมวลผลข้อความนั้นๆ และส่งสัญญาณให้ทุก วินโดว์รู้และแสดงผลลัพธ์ต่างๆ โดยสัมพันธ์กับผลของการประมวลที่ได้

ดังได้เห็นแล้วว่าการทำงานของโปรแกรม Windows นั้นค่อนข้างสลับซับซ้อนและเกี่ยวข้องกับ วินโดว์ เหตุการณ์ตอบสนองต่างๆ และผลของการประมวลผลข้อความของ Windows มากมาย แต่สำหรับการเขียนโปรแกรมด้วย Visual BASIC ผู้เขียนโปรแกรมไม่จำเป็นต้องเกี่ยวข้องกับการจัดการค่าแฮนเดิลต่างๆ เพราะ Visual BASIC จะจัดการงานระดับพื้นฐานของ วินโดว์เช่นนี้ให้เราเอง สิ่งที่เราต้องทำ คงเป็นเพียงการกำหนดเหตุการณ์ตอบสนองอื่นๆ ที่เราต้องการให้โปรแกรมของเราทำงานเมื่อเกิดเหตุการณ์ที่เรากำหนดขึ้น อย่างเช่น การคลิกเมาส์ลงบนปุ่มสั่งงาน หรือการพิมพ์ข้อความลงในช่องข้อความ เป็นต้น เพื่อบรรลุวัตถุประสงค์ของโปรแกรมเท่านั้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.6 การเขียนโปรแกรมแบบตอบสนองตามเหตุการณ์ (Event-Driven Programming)

ในการเขียนโปรแกรมประยุกต์ (Application) ทั่วไปนั้น โปรแกรมจะเริ่มทำงานจาก โค้ดโปรแกรมบรรทัดแรกไปตามลำดับขั้นตอนที่ถูกกำหนดไว้ โดยจะทำการเรียกโพรซีเยอร์ต่าง ๆ ขึ้นมาใช้งานตามแต่สถานการณ์ ส่วน โปรแกรมที่เป็นแบบ Event-Driven อย่างเช่น ตัวระบบปฏิบัติการ Windows 95/98 เอง หรือ โปรแกรมที่เขียนด้วย Visual BASIC นั้นแตกต่างออกไป การทำงานของโค้ดโปรแกรมจะไม่ได้เริ่มต้นที่บรรทัดแรกไปตามลำดับขั้นตอนที่ถูกกำหนดมาแล้วแต่อย่างใด ทว่า โปรแกรมจะทำงานในแต่ละ โพรซีเยอร์เหตุการณ์ตอบสนอง (Event Procedure) เมื่อเกิดเหตุการณ์ตอบสนองนั้นขึ้น เหตุการณ์ตอบสนองดังกล่าว อาจมีสาเหตุมาจากเหตุการณ์ที่ผู้ใช้ เป็นผู้สร้างขึ้น (User Event) ข้อความที่ส่งมาจากระบบ (System Message) เหตุการณ์ตอบสนองจากโปรแกรมอื่นๆ หรือแม้แต่จากเหตุการณ์ตอบสนองที่ตัว โปรแกรมเองเป็นผู้สร้างขึ้น ผลของเหตุการณ์ตอบสนองเหล่านี้จะเป็นตัวกำหนดว่าโค้ด โปรแกรมส่วนใดจะถูกเรียกขึ้นมาทำงาน ดังนั้นลำดับขั้นตอนการทำงานของโค้ด โปรแกรมแต่ละส่วนจะแตกต่างกันไปในแต่ละครั้งที่ โปรแกรมถูกต้องเรียกขึ้นมาใช้งาน

เนื่องจากการทำงานที่ไม่เป็นไปตามลำดับขั้นตอนนี้เอง ทำให้เราต้องระมัดระวังในการเรียกใช้งานแต่ละ โพรซีเยอร์ให้ถูกต้องสมเหตุสมผล เช่น สมมติว่าเราต้องการให้ผู้ใช้ป้อนชื่อของตัวเองลงในช่องแสดงข้อความ (Textbox) แล้วคลิกปุ่มสั่งงานเพื่อป้อนข้อมูล เราจึงควรที่จะเขียน โปรแกรมให้ปุ่มสั่งงานนั้น ไม่สามารถใช้งานได้ (Disable) จนกว่าชื่อของผู้ใช้ จะถูกป้อนลงในช่องข้อความ หรืออาจจะเขียนโปรแกรม ป้องกันเหตุผิดพลาด ที่ผู้ใช้อาจจะเป็นผู้สร้างขึ้น เช่นห้ามมิให้ใส่ตัวหนังสือลงไปในช่องข้อความที่ต้องการตัวเลข เป็นต้น

ในบางกรณีการทำงานของโค้ด โปรแกรม ก็อาจจะเป็นผู้สร้างเหตุการณ์ขึ้นเสียเองก็ได้ ดังนั้นในการออกแบบ โปรแกรมภายใต้หลักการของ Event-Driven จึงต้องคำนึงถึงความสมเหตุสมผลของการเรียกโค้ด โปรแกรมขึ้นมาทำงานตามแต่ละเหตุการณ์ และป้องกันข้อผิดพลาดที่อาจเกิดขึ้นได้อย่างครบถ้วน

2.7 ข้อเด่นของ Visual BASIC

ในการเขียนโปรแกรมคอมพิวเตอร์ ด้วยคอมไพเลอร์ ภาษาใดๆ สามารถแบ่งออกได้เป็น 3 ขั้นคือ การเขียนโค้ดโปรแกรม (Writing Code) การแปลภาษาโปรแกรมให้เป็นภาษาเครื่อง (Compiling) และการทดสอบโปรแกรม (Testing) แต่สำหรับการเขียน โปรแกรมด้วย Visual BASIC ดูเหมือนว่าเราจะไม่สามารถแบ่งแยกขั้นตอนการทำงานออกเป็น 3 ขั้นตอนเช่นนี้ได้อย่างชัดเจนในภาษาอื่นๆ หากว่าเราเขียนประโยค โปรแกรมผิดพลาดข้อผิดพลาดนี้จะถูกค้นพบก็ต่อเมื่อเริ่มทำการ คอมไพล์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

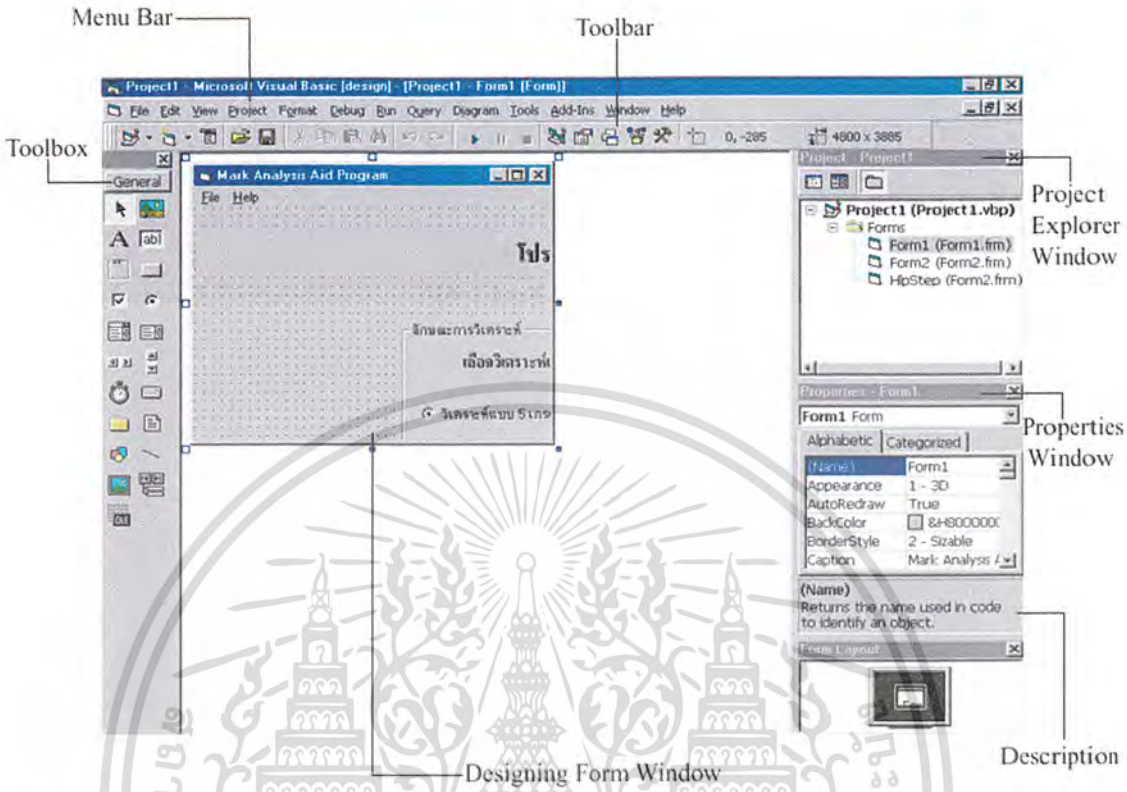
โปรแกรมเท่านั้น เมื่อ คอมไพเลอร์ฟ้องว่ามีที่ใดที่หนึ่งผิดพลาดผู้เขียน โปรแกรมต้องทำการค้นหาข้อผิดพลาดนั้นด้วยตัวเอง ทำการแก้ไขข้อผิดพลาดนั้นด้วยตนเอง จากนั้นก็สั่งให้ คอมไพล์ โปรแกรมอีกครั้งหนึ่ง และทำตามขั้นตอนเช่นนี้ทุกครั้งไป สำหรับแต่ละข้อผิดพลาดที่เกิดขึ้น การค้นหา และแก้ไขข้อผิดพลาด ของโปรแกรมนี้ เป็นขั้นตอนที่กินเวลามากที่สุด เวลาหนึ่ง ในการพัฒนาโปรแกรม

แต่สำหรับตัวแปรภาษา Visual BASIC นั้นจะมีการตรวจสอบโค้ดที่ผู้เขียนทำการเขียนลงไปแต่ละตัวโดยทำการตรวจจับคำผิดพลาด ไวยากรณ์ภาษาที่ผิด และแสดงสีของข้อความฟ้องจุดที่ก่อให้เกิดข้อผิดพลาดนั้นขึ้นมาให้เห็นอย่างชัดเจนในทันทีทันใด ในขณะเดียวกัน Visual BASIC จะทำการตรวจจับข้อผิดพลาดของข้อความที่พิมพ์เข้าไปนั้นนอกจากนี้ Visual BASIC ยังค่อยๆ แปรภาษาโค้ด โปรแกรมที่เราเขียนไปเรื่อยๆ เพื่อว่าเมื่อใดก็ตามที่เราต้องการทดสอบการทำงานของ โปรแกรม ก็จะสามารถกระทำได้อย่างรวดเร็ว โดยเสียเวลาเพียงเล็กน้อย ด้วยคำสั่ง RUN หากว่ายังพบข้อผิดพลาดใดๆ เกิดขึ้น บรรทัดที่เป็นตัวก่อให้เกิดข้อผิดพลาดนั้น ก็จะถูกแสดงขึ้นมา เพื่อให้เราแก้ไขแล้วทำการคอมไพล์ โปรแกรมต่อได้ทันที ทำให้ผู้ใช้สามารถทดสอบการทำงานของโปรแกรมได้ตลอดเวลา ทำการแก้ไขได้อย่างรวดเร็ว และตรงจุด ซึ่งสำหรับนักเขียนโปรแกรมระดับเริ่มต้นแล้ว จุดนี้ถือว่าเป็นประโยชน์อย่างมากเพราะทำให้เรียนรู้ข้อผิดพลาดได้อย่างรวดเร็ว

2.8 สภาพแวดล้อมในการใช้งาน Visual BASIC

สภาพแวดล้อม ในการใช้งานของ Visual BASIC ประกอบไปด้วย คำสั่งและวินโดว์ ที่ใช้ในการทำงานมากมาย เช่น คำสั่งตรวจสอบแก้ไขโค้ด โปรแกรม คำสั่งคอมไฟล์โค้ด โปรแกรมแถบเครื่องมือ สภาพแวดล้อมที่เต็มไปด้วยฟังก์ชัน สำหรับทำงานหลายๆอย่าง ในหนึ่งหน้าจอนี้เรียกว่า IDE (Integrated development Environment) โดยฟังก์ชัน สำหรับทำงานแต่ละตัวนั้น ต่างก็มีหน้าที่ของมันเองตัว โปรแกรมแยกออกจากกันและต่างก็มีส่วนติดต่อผู้ใช้เป็นของตัวเอง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.3 สภาพแวดล้อมในการทำงานแบบ IDE ของ Visual BASIC 6.0

- แถบเมนูคำสั่ง (Menu bar) เป็นแถบที่แสดงปุ่มคำสั่งทั้งหมด ที่สามารถเรียกใช้ได้ ในโปรแกรม Visual BASIC ในแต่ละกลุ่มคำสั่ง จะมีคำสั่งย่อยรวมอยู่ภายใน กลุ่มคำสั่งดังกล่าวนี้จะประกอบด้วย ปุ่มคำสั่งพื้นฐานของ โปรแกรมบน วินโดว์ คือ File, Edit และ Help

- เมนูเสริม (Context Menu) เป็นเมนูที่เรียกใช้คำสั่งที่ใช้บ่อยๆ หรือจำเป็นต่อการใช้งานของ object ใน Visual BASIC ได้อย่างรวดเร็ว เมนูเสริมของ object ใดๆ จะถูกแสดงขึ้นมาก็ต่อเมื่อเราคลิกเมาส์ปุ่มขวาบน object นั้นๆ

- แถบปุ่มคำสั่ง (Toolbar) เป็นแถบแสดงปุ่มต่างๆ เพื่อเข้าถึงคำสั่งต่างๆ ของ โปรแกรม Visual BASIC ได้อย่างรวดเร็ว การคลิกเมาส์ลงบนปุ่มคำสั่งเหล่านี้ให้ผลเช่นเดียวกับการเลือกคำสั่งจากแถบเมนูคำสั่ง เมื่อเราเริ่มต้นใช้ โปรแกรม Visual BASIC นี้เป็นครั้งแรก แถบเครื่องมือนี้จะถูกกำหนดมาให้แสดงให้เราเห็นเป็นเบื้องต้น รวมทั้งแถบเมนูสำหรับการแก้ไขโค้ด โปรแกรม (Editing) การออกแบบฟอร์ม (Form Design) และการแก้ไขข้อผิดพลาด (Debugging) อย่างไรก็ตามแถบปุ่มคำสั่งดังกล่าวนี้สามารถซ่อนไว้ไม่ให้แสดงได้ด้วยคำสั่งย่อยในกลุ่มคำสั่ง View ในแถบเมนูคำสั่ง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

หรือสามารถกำหนดให้มีการ"จัดตำแหน่งอัตโนมัติ" (Dock) หรือกำหนดให้เป็นแบบ " ซ้อนกันได้" (Float) ด้วยคำสั่ง Options ในกลุ่มคำสั่ง Tools

- แถบกล่องเครื่องมือ (Toolbox) แสดงแถบบรรจุคอนโทรล (control) ต่างๆ ที่สามารถนำมาใช้บนฟอร์ม (Form) แถบกล่องเครื่องมือที่เห็นเป็นครั้งแรกเมื่อเริ่มต้นใช้ โปรแกรม Visual BASIC นั้นเป็นแถบแสดงคอนโทรลพื้นฐาน ของ Visual BASIC เราสามารถเพิ่มเติม และกำหนด ชนิดของคอนโทรลที่แสดงอยู่บนแถบกล่องเครื่องมือนี้ได้ด้วยการคลิกเมาส์ปุ่มขวาบนแถบกล่องเครื่องมือเพื่อให้เมนูเสริมปรากฏขึ้นมาและเลือกคำสั่งส่วนประกอบ (Components)

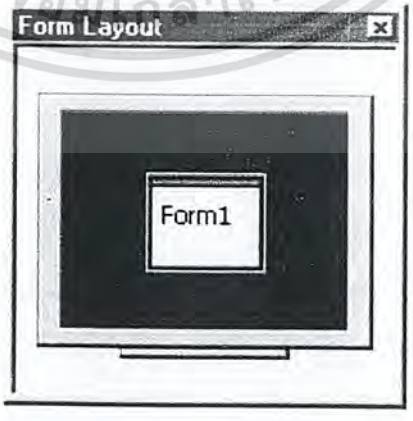
- วินโดว์จัดการโปรเจ็กต์ (Project Explorer Window) แสดงรายชื่อของโมดูลฟอร์มและโมดูลต่างๆ ของโปรเจ็กต์ที่กำลังทำงานอยู่ โปรเจ็กต์ (Project) หมายถึง กลุ่มไฟล์ทุกชนิดของ Visual BASIC ที่เราใช้สำหรับสร้าง โปรแกรมประยุกต์หนึ่งๆ

- วินโดว์แสดงคุณลักษณะ (Properties Window) แสดงค่าคุณลักษณะของฟอร์มหรือคอนโทรลที่กำลังถูกเลือกคุณลักษณะ (Properties) หมายถึงค่าต่างๆที่กำหนดรูปแบบการแสดงออกของออบเจ็กต์นั้นๆ เช่น ขนาด สี เป็นต้น

- วินโดว์สำหรับออกแบบฟอร์มและคอนโทรล (Form Designer) แสดงฟอร์มให้ผู้ใช้เห็นเพื่อใช้สำหรับออกแบบส่วนติดต่อผู้ใช้ (Interface) ของโปรแกรมด้วยการเพิ่มเติมคอนโทรลและกราฟิกต่างๆ ลงบนฟอร์มนั้นๆ

- วินโดว์แสดงโค้ด โปรแกรม (Code Editor Window) ทำหน้าที่เป็นตัวแก้ไขโค้ดโปรแกรมที่เราเขียนขึ้น โดยแต่ละฟอร์มและโมดูลที่เราสร้างขึ้นจะมี วินโดว์แสดงโค้ดโปรแกรมของตัวมันเอง

- วินโดว์แสดงตำแหน่งของฟอร์ม (Form Layout Window) ทำหน้าที่แสดงตำแหน่งของฟอร์ม ที่จะปรากฏขึ้นมาเมื่อมีการเรียกใช้งานผ่านทางหน้าจอจำลองขนาดเล็ก ดังรูปที่ 2.4



รูปที่ 2.4 วินโดว์แสดงตำแหน่งของฟอร์ม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.9 การแสดงหน้าจอหลักและการจัดตำแหน่งวินโดว์

เราสามารถจัดองค์ประกอบ หน้าจอการใช้งานโปรแกรม Visual BASIC ได้ตามความพอใจด้วยการ เลือกการจัดตำแหน่ง (Docking) ของวินโดว์ แต่ละวินโดว์ และการแสดง หน้าจอหลัก (Document Presentation) ว่าจะให้มีลักษณะอย่างไร

การแสดงหน้าจอหลักแบบ SDI และ MDI

การใช้งานภายใต้รูปแบบ SDI (Single Document Interface) หมายถึง ผู้ใช้สามารถเคลื่อนย้าย วินโดว์แต่ละวินโดว์ ไปยังตำแหน่งใดของหน้าจอก็ได้ และวินโดว์นั้นจะไปซ้อนทับวินโดว์อื่น ในขณะที่การใช้งานภายใต้รูปแบบ MDI (Multiple Document Interface) หมายถึง แต่ละวินโดว์จะถูกบรรจุอยู่ในวินโดว์หลัก (Parent Windows) คือโปรแกรม Visual BASIC 6.0 เพียงวินโดว์เดียว

การเปลี่ยนรูปแบบการทำงานระหว่าง 2รูปแบบนี้สามารถเปลี่ยนได้โดยใช้คำสั่งย่อย Options ในกลุ่มคำสั่ง Tools ซึ่งหลังจากเปลี่ยนรูปแบบแล้วจะต้องออกจากโปรแกรมและเริ่มโปรแกรมใหม่อีกครั้ง การเปลี่ยนรูปแบบการแสดงหน้าจอหลักจึงจะมีผล

2.10 การจัดตำแหน่งของวินโดว์โดยอัตโนมัติ

วินโดว์เกือบทั้งหมดใน Visual BASIC สามารถจัดตำแหน่งได้โดยอัตโนมัติ ซึ่งหมายความว่าขอบของวินโดว์หนึ่งๆ จะไปทาบติดกับวินโดว์อื่นๆ พอดี ไม่มีการทับซ้อนกันของวินโดว์ เช่น วินโดว์ของแถบกล่องเครื่องมือ วินโดว์แสดงตำแหน่งของฟอร์ม วินโดว์แสดงคุณลักษณะของออบเจ็กต์

ภายใต้รูปแบบการทำงานแบบ MDI วินโดว์ต่างๆ สามารถจัดให้อยู่ทางด้านไหนของวินโดว์หลักก็ได้ ในขณะที่ภายใต้รูปแบบการทำงานแบบ SDI การจัดตำแหน่งของวินโดว์ต่างๆ สามารถทำได้เฉพาะทางด้านล่าง ของแถบเมนูคำสั่งเท่านั้น ในการเลือกคำสั่ง ว่าต้องการให้มีการจัดตำแหน่งวินโดว์ต่างๆ โดยอัตโนมัติหรือไม่ สามารถเลือกได้จากคำสั่งย่อย Options ในกลุ่มคำสั่ง Tools

2.11 ขั้นตอนการใช้งาน Visual BASIC สร้างโปรแกรมอย่างง่าย

ขั้นตอนการสร้างโปรแกรมด้วย Visual BASIC นั้นแบ่งออกเป็น 3 ขั้นตอนหลักๆ คือ

ขั้นตอนที่ 1 ออกแบบส่วนติดต่อกับผู้ใช้ (Designing User-Interface)

ขั้นตอนที่ 2 การกำหนดค่าคุณลักษณะ (Setting Properties)

ขั้นตอนที่ 3 การเขียนโค้ดโปรแกรม (Writing Code)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.12 หลักการพัฒนา โปรแกรมคอมพิวเตอร์

เราสามารถกล่าวได้ว่าการพัฒนา โปรแกรมคอมพิวเตอร์ทุกชนิด ตั้งแต่โปรแกรมสำหรับงานธุรกิจอย่างเช่น Microsoft Word, Windows 95/98 หรือกระทั่ง โปรแกรมทั้งหลาย ไม่ว่าจะเขียนขึ้นด้วยภาษาคอมพิวเตอร์อะไรก็ตาม จะประกอบไปด้วยขั้นตอนพื้นฐานในการสร้าง 5 ขั้นตอน ดังต่อไปนี้

1. การเริ่มต้นใช้งาน โปรแกรม (Initialization)
2. ข้อมูลอินพุต (Input Data)
3. การประมวลผลข้อมูล (Processing)
4. ข้อมูลเอาต์พุต (Output Data)
5. การจบการทำงานโปรแกรม (Shut Down)

เราจะกล่าวถึงแต่ละขั้นตอนในรายละเอียดดังต่อต่อไปนี้

2.12.1 การเริ่มต้นใช้งาน โปรแกรม

มีโปรแกรมอยู่หลาย โปรแกรมที่ต้องการการตระเตรียมข้อมูลต่างๆ ก่อนที่ โปรแกรมนั้นๆ จะสามารถทำงานได้ ขั้นตอนการกำหนดค่าและทดสอบค่าต่างๆ นี้เองที่เราเรียกว่า ขั้นตอนการเริ่มต้นใช้งาน โปรแกรม (Initialization) ซึ่งหมายถึง การจัดการข้อมูลทั้งหลายที่จำเป็นในการใช้งานโปรแกรม นั้นๆ ก่อนการเริ่มใช้งาน โปรแกรม ยกตัวอย่างเช่น โปรแกรมเกมบน Windows ที่เขียนด้วย DirectX API จะต้องทำการปรับโหมดการแสดงผลของจอภาพก่อนที่จะเริ่มต้น โปรแกรมทุกครั้ง อย่างนี้เป็นต้น

อย่างไรก็ตาม สำหรับ โปรแกรมที่ไม่ซับซ้อนยุ่งยากนัก ก็อาจจะไม่มีการกำหนดค่าใดๆ เลย ก่อนการใช้งาน โปรแกรม ดังนั้น เราสรุปว่าทุก โปรแกรมที่เราจะพัฒนาขึ้นต้องมีการ Initializing ข้อมูลต่างๆ ที่จำเป็นต่อการทำงานของ โปรแกรมก่อนเสมอ

2.12.2 ข้อมูลอินพุต

โปรแกรมคอมพิวเตอร์ทั้งหลายที่เราเห็นนั้น ไม่ได้เป็นอะไรมากไปกว่า โปรแกรมที่ยุ่งอยู่กับการจัดการข้อมูลและดัดแปลงข้อมูล เพราะถ้าหากไม่มีข้อมูลอินพุตป้อนเข้าไปให้ โปรแกรมแล้ว โปรแกรมเกือบทุก โปรแกรมในโลกก็ไม่สามารถทำงานได้ ดังนั้นในส่วนของ ข้อมูลอินพุต (Input Data) จึงเป็นขั้นตอนที่ทำการรวบรวม ตรวจสอบ และคัดเลือกข้อมูลที่จะถูกใช้ใน โปรแกรมของเรา

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

โดยทั่วไปแล้วข้อมูลที่เป็นอินพุตในการประมวลผลมักจะมาจากการป้อนของผู้ใช้งาน หรือการอ่านข้อมูลในไฟล์จากดิสก์ และยังมีรูปแบบของการได้มาซึ่งข้อมูลที่ใช้ในการประมวลผลอื่นๆ อีก เช่น ปากกาแสง (Light Pen) หรือการสัมผัสหน้าจอ (Touching Screen) ซึ่งต้องการอุปกรณ์พิเศษเพื่อการใช้งาน แต่ถึงกระนั้นจุดประสงค์ของการรวบรวมข้อมูลก็ยังคงเป็นเช่นเดิม คือเพื่อนำไปใช้ประมวลผลใน โปรแกรม

2.12.3 การประมวลผลข้อมูล

โปรแกรมเกือบทุก โปรแกรมในโลกถูกออกแบบให้นำข้อมูลที่ป้อนเข้าไป ไปทำการดัดแปลงอย่างใดอย่างหนึ่งจนกลายเป็นข้อมูลใหม่ ซึ่งการดัดแปลงข้อมูลจากรูปแบบหนึ่งไปเป็นข้อมูลอีกรูปแบบหนึ่งนี้เองที่เราเรียกว่า การประมวลผลข้อมูล (Processing) ยกตัวอย่างเช่น เรามีข้อมูลตัวเลขอยู่ 10 ตัวคละเคล้ากัน เราต้องการให้ข้อมูลทั้ง 10 ตัวนี้เรียงจากน้อยไปหามาก การประมวลผลสำหรับงานนี้ก็คือ การเรียงข้อมูล

ประสิทธิภาพ ในการทำงานของโปรแกรมใดๆ ขึ้นอยู่กับรูปแบบ การประมวลผล ของโปรแกรมนั้นๆ ดังเช่นตัวอย่างที่ยกมา มีวิธีเรียงข้อมูลอยู่นับสิบแบบที่ใช้กัน แต่ละวิธีก็มีข้อดีและประสิทธิภาพในการทำงานไม่เท่ากัน อย่างเช่น วิธีการเรียงข้อมูลแบบ Bubble Sort เหมาะสำหรับข้อมูลที่ถูกจัดเรียงเกือบเรียบร้อยแล้ว หากนำมาใช้เรียงข้อมูลที่กระจัดกระจายก็พอใช้ได้ แต่จะช้ากว่ามากเมื่อเทียบกับวิธีอื่นๆ เป็นต้น สิ่งที่เราต้องทำความเข้าใจก็คือ วิธีการเรียงข้อมูลแต่ละแบบที่มีอยู่นั้นมีวิธีการประมวลผลอย่างไร เพื่อเราจะได้เลือกวิธีการ ที่เหมาะสม ไปใช้ได้ถูกต้อง และเกิดประสิทธิภาพในการทำงานสูงสุดกับ โปรแกรมของเรา

เราสามารถสรุปได้ว่า โปรแกรมของเราจะทำงานได้ดีเพียงใดนั้น ขึ้นอยู่กับรูปแบบของการประมวลผลที่เราเขียนขึ้น ขั้นตอนนี้จึงเป็นขั้นตอนสำคัญ ที่ทำให้โปรแกรมหลาย โปรแกรมที่ทำงานคล้ายกันแต่มีประสิทธิภาพแตกต่างกันอย่างเห็นได้ชัด

2.12.4 ข้อมูลเอาต์พุต

หลังจากที่ข้อมูลอินพุตถูกดัดแปลงแล้ว ข้อมูลใหม่ที่ผ่านการประมวลผลแล้วเราเรียกว่า ข้อมูลเอาต์พุต (Output Data) อย่างเช่นในตัวอย่างที่แสดง ข้อมูลที่ผ่านการจัดเรียงจากน้อยไปมากแล้วก็คือผลลัพธ์ของการจัดเรียงข้อมูล และการที่ผู้ใช้จะเห็นผลของการประมวลผลได้ก็โดยผ่านทางหน้าจอมอนิเตอร์ เครื่องพิมพ์ ข้อมูลเป็นไฟล์ในดิสก์ หรือแม้แต่ข้อมูลผ่านทางโมเด็ม

ดังนั้น ไม่ว่าอุปกรณ์ในการแสดงผลลัพธ์จะเป็นอะไร ผลลัพธ์ของขั้นตอนนี้ก็มุ่งสู่จุดหมายเดียวกัน นั่นคือการแสดงข้อมูลผลลัพธ์ของการประมวลผลให้ผู้ใช้ได้รับรู้ในทางใดทางหนึ่ง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

42257

2.12.5 การจบการทำงาน โปรแกรม

เมื่อใดที่เราต้องการเลิกใช้งาน โปรแกรม เราเพียงแค่ออกจาก โปรแกรมนั้น ๆ ฟังดูง่ายมากในสายตาของผู้ใช้งาน แต่ในสายตาของนักเขียน โปรแกรมแล้วเป็นขั้นตอนที่ยุ่งยากกว่านั้นการจบการทำงาน โปรแกรม (Shut Down) นั้นหมายถึง ขั้นตอนที่เกิดขึ้นหลังจากการแสดงผลลัพธ์ของการประมวลผลเสร็จสิ้นแล้ว ยกตัวอย่างเช่น โปรแกรมของเราอาจจะอ่านไฟล์จากดิสก์เพื่อเก็บข้อมูลสำหรับการประมวลผลขณะที่ โปรแกรมกำลังทำงาน เมื่อมีการออกจาก โปรแกรม เราควรที่จะจัดการปิดไฟล์นั้นๆ ด้วย อย่างเช่น โปรแกรมตัวอย่างเกมบางเกมจะมีการใช้พื้นที่ในหน่วยความจำเพื่อเก็บข้อมูลภาพที่จะแสดง และในขั้นตอนของการออกจาก โปรแกรมก็จะมีการคืนหน่วยความจำเหล่านั้นกลับสู่ระบบปฏิบัติการ สำหรับนำไปใช้ในงานอื่นต่อไป

ในการพัฒนา โปรแกรมใดๆ ในวันข้างหน้า ขอให้นักเขียน โปรแกรมทุกคนระลึกถึงขั้นตอนเหล่านี้และใช้เป็นขั้นตอนคร่าวๆ ในการพัฒนา โปรแกรม นักเขียน โปรแกรมระดับเริ่มต้นจำนวนมากมักจะคิดถึงขั้นตอนการประมวลผลเป็นหลัก โดยลืมคำนึงถึงความสัมพันธ์กับขั้นตอนอื่นๆ ทำให้ โปรแกรมเกิดข้อผิดพลาดนับไม่ถ้วน

ดังนั้นจึงขอแนะนำว่าให้คิดถึงการทำงานของ โปรแกรมตั้งแต่ต้นจนจบให้เรียนร้อยเสียก่อนแล้วจึงค่อยลงมือเขียนโค้ด โปรแกรม เพราะว่าจะทำให้เราเขียนโค้ดได้ง่ายมากขึ้น ตัวโปรแกรมเองก็ง่ายต่อการแก้ไขข้อผิดพลาด และการเพิ่มเติมตัว โปรแกรมก็เป็นไปอย่างสะดวกมากขึ้นด้วยเช่นเดียวกัน

2.13 รายละเอียดประกอบการเขียน โปรแกรม

อย่างไรก็ตามขั้นตอน 5 ขั้นตอนที่กล่าวมาข้างต้นนั้น เป็นเพียงเค้าโครงหลักเพื่อให้เรามองเห็นภาพรวมในการสร้าง โปรแกรมเท่านั้น ในแต่ละขั้นตอนยังต้องการ รายละเอียดประกอบการเขียน โปรแกรม (Sideways Refinement) อีกจำนวนหนึ่งก่อนที่เราจะเริ่มลงมือเขียนโค้ด โปรแกรมกันจริงๆ

รายละเอียดที่กล่าวถึงนี้ก็ยังคงตั้งอยู่บน 5 ขั้นตอนดังกล่าว แต่ว่าแบ่งรายละเอียดในแต่ละขั้นตอนออกเป็นส่วนย่อยๆ เพื่อแสดงว่าแต่ละขั้นตอน โปรแกรมต้องทำอะไรบ้าง ยกตัวอย่างเช่น สมมติว่าเรามีข้อมูลยอดขายประจำเดือนอยู่หนึ่งปี และต้องการให้ โปรแกรมของเราจัดการแสดงข้อมูลเหล่านั้นออกมาเป็นรูปแผนภูมิแท่ง เราจะต้องทำการเขียน โปรแกรมดังกล่าวอย่างไร

ก่อนที่เราจะลงมือเขียนโค้ด โปรแกรม ให้บรรยายรายละเอียดของแต่ละขั้นตอนในการสร้าง โปรแกรม รวมทั้งฟังก์ชันที่ต้องการใช้ ดังตัวอย่างตาราง จุดประสงค์ของการสร้างรายละเอียดเช่นนี้ก็เพื่อแสดงขั้นตอนย่อยๆ ในแต่ละขั้นตอนหลักว่าจะต้องมีอะไรบ้าง เพื่อให้เราสามารถมองเห็นเส้นทางและลำดับขั้นตอนในการเขียน โปรแกรมได้ชัดเจนยิ่งขึ้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 2.1 ตัวอย่างรายละเอียดประกอบการเขียน โปรแกรม

ขั้นตอน	งานที่ต้องทำ	ฟังก์ชันที่ใช้ (สมมติ)
การเริ่มต้นใช้งานโปรแกรม	1. เคลียร์หน้าจอ 2. เปิดไฟล์ที่เก็บข้อมูลยอดจำหน่าย	Cls() OpenFile()
ข้อมูลอินพุต	1. ดึงข้อมูลยอดจำหน่ายเฉพาะจากเดือนที่ต้องการนำมาแสดง	Get_MonthSales()
ประมวลผล	1.หาค่าต่ำสุดและสูงสุดของชุดข้อมูล 2. กำหนดสเกลที่จะใช้	FindMinMax() DoScaling()
ข้อมูลเอาต์พุต	1. วาดแกน X และแกน Y 2. แสดงข้อความกำกับแกน 3. วาดแผนภูมิแท่ง 4. แสดงชื่อภาพแผนภูมิแท่ง	DrawAxis() LabelAxis() DrawBars() LabelBars()
การจบการทำงานโปรแกรม	1. ปิดไฟล์ข้อมูล 2. ออกจากโปรแกรม	CloseFile() End

2.14 อัลกอริทึม

ในการที่จะบรรลุซึ่งการกระทำใดๆ นั้นจะต้องมีลำดับความคิดอันนำไปสู่การกระทำและผลลัพธ์ของมัน อัลกอริทึม (Algorithm) ก็หมายถึง ลำดับขั้นตอนในการปฏิบัติงานใดงานหนึ่งให้สำเร็จ ให้ลองนึกถึงตำราปรุงอาหารหรือพิมพ์เขียวในการสร้างตึก สิ่งเหล่านั้นคือ ตัวอย่างที่ดีของความหมายของ คำว่า "อัลกอริทึม" จากตัวอย่างเรื่องการจัดเรียงข้อมูลจากน้อยไปหามากที่ได้กล่าวมา การจัดเรียงข้อมูลแต่ละแบบต่างก็มีอัลกอริทึมที่แตกต่างกัน ตัวอย่างเช่น การจัดเรียงแบบ Bubble Sort ซึ่งมีโค้ด โปรแกรมไม่ยากนัก แต่ไม่ค่อยมีประสิทธิภาพเท่าเทียมกับวิธีอื่นที่มีโค้ด โปรแกรมที่ซับซ้อนกว่าและมีประสิทธิภาพมากกว่า

อัลกอริทึมมีความสัมพันธ์โดยตรงต่อประสิทธิภาพในการทำงานของ โปรแกรม ด้วยผลลัพธ์ของงานอย่างเดียวกัน หากกระทำด้วยอัลกอริทึมที่แตกต่างกัน ก็อาจจะสร้างผลลัพธ์ที่แตกต่างอย่างมีนัยสำคัญได้อย่างไม่น่าเชื่อ โดยการเขียนรายละเอียดประกอบการเขียน โปรแกรม 5 ขั้นตอนดัง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กล่าวเป็นตัวบังคับให้เราต้องคิดถึงอัลกอริทึมในการประมวลผลข้อมูลของ โปรแกรมที่เราต้องการจะสร้างก่อนลงมือจริง

ดังนั้น จงใช้เวลาให้เต็มที่ในการเขียนรายละเอียดประกอบการเขียน โปรแกรมดังกล่าวก่อนลงมือเขียนโค้ด โปรแกรม ซึ่งไม่ว่ามันจะเสียเวลาไปเท่าไร แต่มันก็ยังคุ้มค่าในระยะยาว เมื่อ โปรแกรมของคุณมีขนาดใหญ่ขึ้น หากเป็นไปได้ควรฝึกให้เป็นนิสัยเลยยิ่งดี ดังนั้นจึงขอสรุปว่าแม้รายละเอียดประกอบการเขียน โปรแกรมจะไม่ใช่สิ่งจำเป็นในการพัฒนา โปรแกรมใดๆ แต่ก็เป็นสิ่งที่มีประโยชน์อย่างยิ่ง สำหรับการเขียน โปรแกรม

2.15 การเขียน โปรแกรมด้วยภาษา Visual BASIC

Visual BASIC เป็นภาษาคอมพิวเตอร์ที่เป็น Object-based programming หมายถึง การเขียน โปรแกรมที่มองสิ่งต่างๆ เป็น "ออบเจ็กต์ (วัตถุ)" ความหมายของออบเจ็กต์ ในทางการเขียน โปรแกรมก็ไม่ต่างอะไรกับวัตถุทั่วไปที่เราพบในชีวิตประจำวันมากนัก มีออบเจ็กต์อยู่สัมพันธ์ของแต่ละออบเจ็กต์ใน โปรแกรมของเรานั่นเอง

ถ้าเคยมีประสบการณ์ในการเขียน โปรแกรมด้วยภาษาอื่นๆ มาบ้างแล้ว ไม่ว่าจะเป็น C, C++ Pascal หรืออื่นๆ จะพบว่าเนื้อหานี้ค่อนข้างจะคุ้นเคยอยู่แล้ว เนื่องจากเป็นเรื่องของหลักไวยากรณ์ของภาษาที่มีโครงสร้างไม่ต่างกันมากนักในแต่ละภาษา แต่รูปแบบการออกแบบ โปรแกรมตอบสนองตามเหตุการณ์ (Event-Driven) ของ Visual BASIC เป็นข้อแตกต่างที่สำคัญที่นักเขียน โปรแกรมภาษาอื่นต้องทำความเข้าใจ สำหรับนักเขียน โปรแกรมมือใหม่ยิ่งเป็นเรื่องที่ดีสำหรับการเริ่มต้นการเขียน โปรแกรมภาษาด้วย Visual BASIC เพราะประการหนึ่งนั้น Visual BASIC เป็นภาษาที่มีความซับซ้อนน้อยกว่าภาษาตระกูล C และประการที่สอง Visual BASIC มีศักยภาพสูงเพียงพอต่อการใช้งาน และไม่แน่ว่าในอนาคตอันใกล้อาจมีความเป็นไปได้ว่าจะกลับมาได้รับความนิยมมากอีกครั้งเหมือนเช่นภาษา BASIC ในอดีต

แท้จริงแล้ว "โปรแกรมประยุกต์" (Application) หนึ่งๆ ก็เป็นเพียงที่รวมของคำสั่งทั้งหลายที่สั่งให้คอมพิวเตอร์ทำงานอย่างใดอย่างหนึ่ง โปรแกรมใดจะทำงานอะไรก็ขึ้นอยู่กับผู้เขียน โปรแกรมนั้นๆ เป็นผู้กำหนด โดยคำสั่งเหล่านั้นมีรูปแบบต่างกันไปตามแต่ละภาษาคอมพิวเตอร์แต่ก็เป็นไปเพื่อจุดประสงค์เดียวกันคือ เพื่อสั่งคอมพิวเตอร์ให้ทำงาน

โปรแกรมตั้งแต่แบบที่ง่ายที่สุดเพียงแค่คลิกปุ่มสั่งงานครั้งเดียวเพื่อให้แสดงข้อความ "Hello World" ขึ้นบนหน้าจอที่บรรจุโค้ด โปรแกรมเพียงบรรทัดเดียว จนถึง โปรแกรมช่วยออกแบบทางวิศวกรรมที่มีโค้ด โปรแกรมหลายรอยหน้าก็ถือเป็น โปรแกรมเหมือนกัน โดยเฉพาะอย่างยิ่ง โปรแกรมที่ต้องเกี่ยวข้องกับฟังก์ชันสุ่มค่า (Random Function) เป็นจำนวนมาก ในการเขียน โปรแกรมนั้น ผู้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เขียนจะต้องเขียน โปรแกรมโดยคำนึงถึงผลลัพธ์ที่เป็นไปได้ทั้งหมดเพื่อให้ โปรแกรมสามารถทำงานได้อย่างไม่ผิดพลาด

เนื่องจากว่า Visual BASIC เป็นภาษาคอมพิวเตอร์ที่ทำงานกับออบเจ็กต์ ดังนั้นรูปแบบของโค้ดโปรแกรมก็จะสัมพันธ์กับออบเจ็กต์ต่างๆ ที่เราเห็นบนหน้าจอเป็นหลัก กล่าวคือ ทุกออบเจ็กต์ที่ปรากฏขึ้นบนหน้าจอของ โปรแกรมที่เขียนด้วย Visual BASIC ต่างก็บรรจุข้อมูลเกี่ยวกับคุณลักษณะของตัวเองและโค้ด โปรแกรมจำนวนหนึ่งที่จะกำหนดการใช้งานของมันวินโดว์หนึ่งๆ ใน โปรแกรมที่ออกแบบด้วย Visual BASIC เราจะเรียกว่า โมดูลฟอร์ม (Form Module) คือไฟล์ที่มีนามสกุล *.FRM ในแต่ละฟอร์มจะประกอบไปด้วย ส่วนออกแบบของฟอร์มและคอนโทรลที่เป็นหน้าตา (Presentation) ของฟอร์ม อันปรากฏให้เห็นจริงขณะ โปรแกรมทำงาน และส่วนของโค้ด โปรแกรม (Source Code) อันประกอบด้วย ส่วนประกาศ (General Declaration) และโพรซีเยอร์ย่อยต่างๆ แบ่งเป็น 3 ประเภท คือ โพรซีเยอร์ซับรูทีน โพรซีเยอร์ฟังก์ชัน และโพรซีเยอร์เหตุการณ์ตอบสนอง ในแต่ละโพรซีเยอร์เหตุการณ์ตอบสนองเหล่านั้นจะบรรจุโค้ด โปรแกรมสำหรับทำงานเพื่อตอบสนองเหตุการณ์ในเหตุการณ์หนึ่ง

บนฟอร์มใดๆ จะบรรจุคอนโทรลต่างๆ เอาไว้หรือไม่ก็ได้ แต่โดยทั่วไปแล้วบนฟอร์มจะบรรจุคอนโทรลไว้ด้วยกันหลายชนิด แต่ละคอนโทรลสามารถจะตอบสนองเหตุการณ์ต่างๆ ที่จะเกิดได้ตามโค้ดโปรแกรม ที่ถูกเขียนไว้ในโพรซีเยอร์เหตุการณ์ตอบสนองของคอนโทรลเหล่านั้น

ในบางกรณีที่โค้ด โปรแกรมส่วนหนึ่งไม่ได้เกี่ยวข้องกับฟอร์มใดฟอร์มหนึ่งโดยเฉพาะ และเราต้องการให้ฟอร์มอื่นๆ สามารถเรียกใช้งานได้ด้วย เราก็อาจจะเขียนโค้ด โปรแกรมเหล่านั้นไว้ในโมดูลอื่น อันได้แก่ โมดูลมาตรฐาน (Standard Module (*.BAS)) โพรซีเยอร์ใดก็ตามที่เกี่ยวข้องกับออบเจ็กต์ใดๆ เป็นพิเศษ เพื่อว่าจะได้นำมาบรรจุไว้ในโมดูลมาตรฐานและเรียกใช้ได้จากทุกออบเจ็กต์ แทนการที่เราจะต้องเขียนโค้ด โปรแกรมเหล่านั้นซ้ำแล้วซ้ำอีกสำหรับแต่ละออบเจ็กต์

โมดูลคลาส (Class Module (*.CLS)) เป็นโมดูลอีกชนิดหนึ่งที่มีไว้สำหรับสร้างออบเจ็กต์ที่สามารถถูกเรียกใช้งานได้จากโพรซีเยอร์ต่างๆ ใน โปรแกรมซึ่งหากเปรียบเทียบกับโมดูลมาตรฐานแล้ว โมดูลมาตรฐานจะบรรจุเพียงโค้ด โปรแกรมสำหรับทำงานเท่านั้น ในขณะที่โมดูลคลาสจะบรรจุทั้งโค้ดโปรแกรมและข้อมูลอื่นๆ ที่เกี่ยวข้อง ออบเจ็กต์ดังกล่าวนี้เปรียบเสมือนคอนโทรล คือมีข้อมูลหลายชนิดรวมอยู่ด้วยกันเป็นออบเจ็กต์ แต่ไม่สามารถปรากฏให้เราจับต้องได้เช่นคอนโทรล

เนื่องจากโค้ด โปรแกรมที่บรรจุอยู่ในโมดูลคลาสส่วนมาก จะเป็นโค้ด โปรแกรมที่เกี่ยวข้องกับการสร้างคลาสที่ผู้เขียน โปรแกรมสร้างขึ้นเอง ดังนั้นจึงไม่มีการเอ่ยถึงโมดูลคลาสมากเท่ากับโมดูลฟอร์มและโมดูลมาตรฐาน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.16 เข้าใจการทำงานของ โปรแกรมแบบตอบสนองตามเหตุการณ์ (Event-Driven)

ในเมื่อแต่ละฟอร์ม หรือคอนโทรลสามารถตอบสนองต่อเหตุการณ์ต่างๆ ที่เกิดขึ้นในโปรแกรม ได้เราจะใช้เหตุการณ์เหล่านี้เป็นตัวจุดชนวน (Trigger) ในการทำงานของ โปรแกรม แต่ละฟอร์ม และคอนโทรล Visual BASIC ได้ถูกกำหนดมาก่อนแล้วว่า สามารถจะตอบสนองกับเหตุการณ์แบบใดได้บ้าง โค้ดโปรแกรมที่เราจะเขียนก็จะบรรจุอยู่ในโพรซีเยอร์เหตุการณ์ตอบสนองเหล่านั้นเอง แต่ก็เป็นหน้าที่ของผู้เขียน โปรแกรมเองว่าจะเลือกใช้เหตุการณ์ตอบสนองอะไรของฟอร์มและของคอนโทรลตัวไหน เพื่อจะบรรจุโค้ด โปรแกรมสำหรับทำงานอะไร และอย่างไร แม้ว่าเหตุการณ์ตอบสนองของฟอร์มและคอนโทรลต่างๆ จะแตกต่างกันไปตามแต่ละชนิดของคอนโทรล แต่จะมีเหตุการณ์ตอบสนองพื้นฐานจำนวนหนึ่งที่ฟอร์มและคอนโทรลทุกชนิดมีร่วมกัน

เราสามารถสรุปขั้นตอนในการทำงาน โปรแกรมที่ออกแบบด้วย Visual BASIC ได้ดังนี้

1. โปรแกรมประยุกต์ถูกเรียกขึ้นมาใช้งาน เริ่มด้วยการโหลดตัวเองเข้าสู่หน่วยความจำและแสดงหน้าจอให้ผู้ใช้เห็น
2. ฟอร์มและคอนโทรลต่างๆ จะคอยรับเหตุการณ์ตอบสนองต่างๆ ที่มาจากผู้ใช้ โปรแกรมผ่านทางเมาส์และคีย์บอร์ด หรือมาจากตัวระบบ เช่น เวลา หรือมาจากโค้ด โปรแกรมในตัว โปรแกรมเอง เช่น เหตุการณ์ตอบสนองการโหลด (Load Event) ของฟอร์ม
3. หากว่ามีเหตุการณ์ตอบสนองใดๆ เกิดขึ้น โค้ด โปรแกรมที่บรรจุอยู่ในโพรซีเยอร์เหตุการณ์ตอบสนองดังกล่าวจะถูกเรียกขึ้นมาทำงาน
4. หลังจากที่ทำงานเสร็จ โปรแกรมก็จะคอยเหตุการณ์ตอบสนองอันต่อไป

ดังจะเห็นได้ว่าสิ่งแรกที่เราควรทำในการออกแบบ โปรแกรมด้วย Visual BASIC ก็คือ การออกแบบรูปร่างหน้าตาของ ส่วนติดต่อกับผู้ใช้ (Interface) ของ โปรแกรม อันประกอบด้วยฟอร์มและคอนโทรลทั้งหลาย การจัดวางคอนโทรลต่างๆ บนฟอร์มควรที่จะสัมพันธ์กับการทำงานของโค้ด โปรแกรมที่เราจะเขียนลงไปสำหรับฟอร์มและคอนโทรลเหล่านั้นซึ่งประสิทธิภาพของ โปรแกรมแต่ละ โปรแกรมก็ตัดสินกันตรงความง่ายต่อการใช้งานและการทำงานของโค้ด โปรแกรมนี้เอง การออกแบบความสัมพันธ์ระหว่างส่วนติดต่อผู้ใช้กับโค้ด โปรแกรมที่ดี ยังทำให้ผู้พัฒนา โปรแกรมสามารถปรับปรุง โปรแกรมนั้นๆ ให้ดีขึ้นได้ง่ายกว่าเดิมอีกด้วย

โปรแกรมที่ออกแบบด้วย Visual BASIC ประกอบไปด้วยโมดูลฟอร์มเป็นพื้นฐาน ส่วนโมดูลอื่นๆ นั้นจะมีหรือไม่มีก็ได้ ในแต่ละโมดูลจะประกอบไปด้วยโพรซีเยอร์ย่อยต่างๆ โดยปกติแล้วเมื่อเราเพิ่มโมดูลหรือคอนโทรลใหม่เข้าไปในไฟล์ โปรเจ็กต์ของเรา Visual BASIC จะกำหนดชื่อ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เริ่มต้นให้โดยอัตโนมัติ ดังนั้นเพื่อความสะดวกต่อการสื่อความหมาย เราจึงควรเปลี่ยนค่าคุณลักษณะ Name ของฟอร์มและคอนโทรลเหล่านั้นให้สื่อความหมายมากขึ้น นักเขียน โปรแกรม Visual BASIC โดยทั่วไปนิยมตั้งชื่อฟอร์มและคอนโทรลนำด้วยชื่อย่อของฟอร์มและคอนโทรลแล้วตามด้วยชื่อที่สื่อความหมายถึงฟอร์มและคอนโทรลนั้นๆ

ตารางที่ 2.2 ชื่อย่อที่นิยมใช้กับฟอร์มและคอนโทรล

ฟอร์มและชนิดคอนโทรล	ชื่อย่อที่นิยมใช้	ตัวอย่าง
Form	Frm	FrmMain
Label	Lbl	LblInfo
Textbox	Txt	TxtInput
Command button	Cmd	CmdCancel
Chekbox	Chk	ChkBold
Optionbutton	Opt	OptRed
Combobox	Cbo	CboPayment
Listbox	Lst	LstSurname
Hscrollbar	Hbar	HbarBrightness
Vscrollbar	Vbar	VbarVolume
Timer	Tmr	TmrNextTurn
DriveListbox	Drv	DrvViewer
DirListbox	Dir	DirViewer
FileListbox	Flst	FlstViewer
Picturebox	Pct	PctUser
Image	Omg	ImgLogo

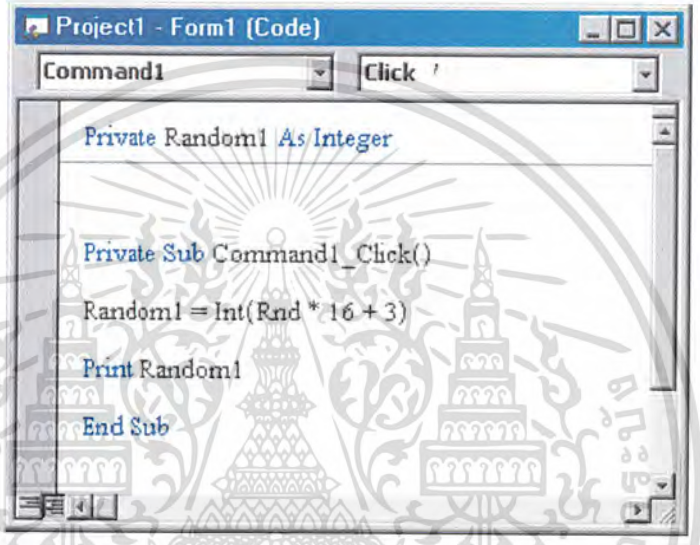
2.17 รูปแบบการเขียนโค้ด โปรแกรมใน Visual BASIC

การเขียนโค้ด โปรแกรมสำหรับ Visual BASIC จะกระทำในวินโดว์แสดงโค้ด โปรแกรม (Code Editor Window) ซึ่งถือเป็น โปรแกรมเขียนข้อความที่มีความสามารถสูงกว่า โปรแกรมเขียนข้อความทั่วไปอยู่หลายด้าน ที่จะช่วยให้การเขียนโค้ด โปรแกรมง่ายมากขึ้น

โปรแกรมของเราจะทำงานตามโค้ด โปรแกรมที่บรรจุอยู่ในโมดูลทั้ง 3 ชนิด ในแต่ละโมดูลก็จะถูกแบ่งเป็นโพรซีเยอร์ย่อยๆ ของออบเจ็กต์ต่างๆ โดยการสร้างหรือไปยังโพรซีเยอร์ของออบเจ็กต์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

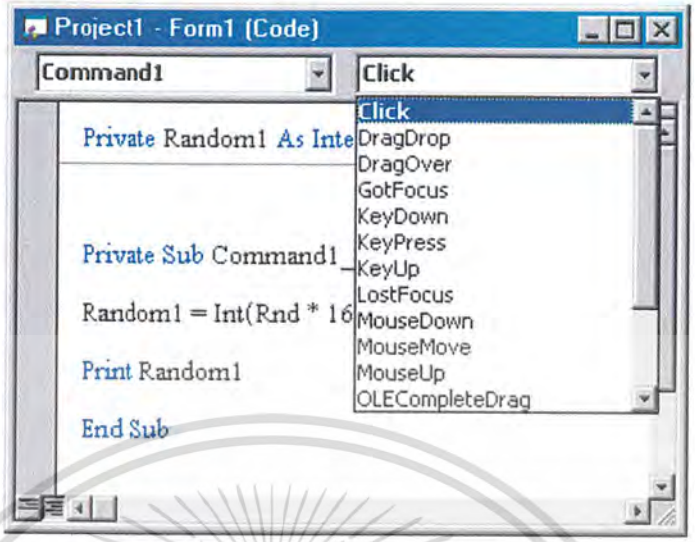
ใดๆ ในแต่ละโมดูลสามารถทำได้ด้วยการคลิกเมาส์ที่ช่องรายชื่อออบเจ็กต์ (Object List Box) ของโมดูลนั้นๆ และเลือกชื่อโพรซีเยอร์ที่ต้องการจากช่องรายชื่อโพรซีเยอร์ (Procedure List Box) ทางด้านขวามือ จะสามารถสังเกตได้ว่าเมื่อเราเลือกที่ส่วนบนสุดของโมดูลฟอร์มทจะมีคำว่า (General) ปรากฏอยู่ในช่องรายชื่อออบเจ็กต์ และมี (Declaration) เป็นโพรซีเยอร์ General ในที่นี้ไม่ใช่ออบเจ็กต์ใดๆ ใน Visual BASIC แต่จะเป็นส่วนที่บอกให้เราทราบว่า โค้ด โปรแกรมในพื้นที่ส่วนนี้



รูปที่ 2.5 วินโดว์แสดงโค้ดโปรแกรม

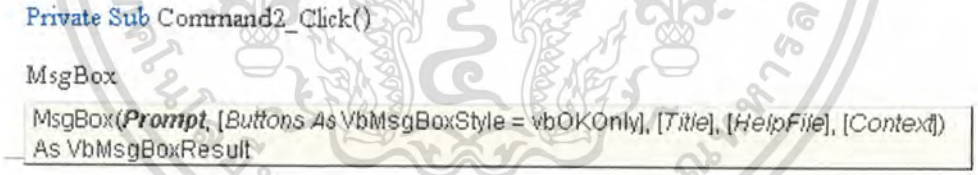
จะบรรจุโค้ดเกี่ยวข้องกับ การประกาศตัวแปร (Variable), ตัวแปรค่าคงที่ (Constants) และโพรซีเยอร์ใน ไดนามิกลิงก์ไลบรารี (DLLs) ต่างๆ ซึ่งเรียกว่า ส่วนประกาศ (General Declaration) และถ้าเราเขียนโค้ด โปรแกรม และชื่อของโพรซีเยอร์เหล่านั้นก็จะถูกเพิ่มเติมเข้าไปในช่องรายชื่อโพรซีเยอร์ เมื่อเราเลือก General เป็นชื่อออบเจ็กต์ รูปแบบการแสดงโค้ด โปรแกรมของวินโดว์ แสดงได้ 2 รูปแบบ คือ แบบแสดงโพรซีเยอร์เพียงโพรซีเยอร์เดียว และแบบแสดงโพรซีเยอร์ที่มีทั้งหมดพร้อมกันโดยมีเส้นคั่นระหว่าง โพรซีเยอร์ ผู้เขียน โปรแกรมสามารถ ใช้ปุ่มตรงมุมด้านล่างซ้ายเพื่อเปลี่ยนไปมาระหว่างการแสดงทั้งสองรูปแบบนี้ได้อย่างรวดเร็ว

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.6 Auto List Members

นอกจาก Visual BASIC จะสามารถแสดงรายชื่อคุณลักษณะและวิธีใช้งานได้อัตโนมัติแล้ว ยังมีความสามารถในการแสดงหลักไวยากรณ์ภาษาของประโยค โปรแกรม และฟังก์ชันต่างๆ (Auto Quick info) ที่ Visual BASIC รู้จักได้อีกด้วย



รูปที่ 2.7 Auto Quick Info

ข้อสำคัญอีกประการหนึ่งที่นักเขียน โปรแกรมทุกคนควรจะฝึกไว้ก็คือ การเติมคำบรรยาย (Comment) ลงไปในโค้ด โปรแกรมที่เขียนขึ้นด้วย คุณประโยชน์สำคัญของคำบรรยายโค้ดเหล่านี้จะปรากฏชัดเจนเมื่อเราต้องกลับมาปรับปรุงแก้ไข โปรแกรมของเราในอนาคต หรือสำหรับผู้เขียนโปรแกรมอื่นในการอ่านให้เข้าใจ เราสามารถเติมคำบรรยายโค้ดลงไปตรงจุดใดของ โปรแกรมก็ได้ด้วยเครื่องหมาย (') เป็นจุดเริ่มต้นแล้วต่อด้วยคำบรรยายของเรา เมื่อใดก็ตามที่ Visual BASIC เห็นเครื่องหมายดังกล่าว Visual BASIC จะไม่เอาข้อความหลังเครื่องหมายมาประมวลผล ดังนั้นคำบรรยายโค้ดจึงนิยมเขียนตรงท้ายประโยค โปรแกรมหรือขึ้นต้นบรรทัดใหม่ต่างหาก ดังตัวอย่างโค้ด โปรแกรมที่มีคำบรรยายโค้ดประกอบดังนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
' This is a comment that was written by programmer.
Text.Text = "Hello"          ' Set Caption of Text1 to "Hello"
```

มีอยู่หลายกรณีที่โค้ด โปรแกรมที่เราเขียนในหนึ่งบรรทัดมีความยาวเกินหนึ่งหน้าจอแสดงผลทำให้เป็นการลำบากเมื่อเราต้องใช้แถบเลื่อนแนวนอนเพื่อจะตรวจสอบข้อมูลที่ล้นจากหน้าจอ ในการเขียนโค้ดโปรแกรมเราสามารถแบ่งให้ประโยค โปรแกรมดังกล่าวไปแสดงในบรรทัดใหม่ได้โดยที่ Visual BASIC ยังมองว่าเป็นประโยคโปรแกรมเดียวกัน ด้วยการใช้ เครื่องหมายบรรทัดต่อเนื่อง (Line-continuation Character) อันประกอบไปด้วยช่องว่าง (Space) และตามด้วยขีดล่าง (_) ดังแสดงในโค้ด ดังนี้

```
If Label1.Visible  = False And Label1.ForeColor = _
VbRed Then Label1.Visible = True
```

2.18 ระบบตัวเลขใน Visual BASIC

ตัวเลขที่ใช้งานใน Visual BASIC เกือบทุกชนิดเป็นเลขฐาน 10 แต่ในบางกรณีอาจจะเป็นเลขฐาน 8 หรือเลขฐาน 16 ซึ่งผู้เขียน โปรแกรมสามารถรับรู้ความแตกต่างได้ โดยเมื่อ Visual BASIC แสดงเลขเป็นฐาน 8 ก็จะมีตัวอักษร &0 ปรากฏนำหน้าเลขฐาน 8 นั้น และตัวอักษร &H จะปรากฏนำหน้าเลขฐาน 16 นั้นๆ ด้วย ต่อไปนี้จะแสดงตัวเลขทั้งสามรูปแบบที่มีค่าเท่ากันในมุมมองของ Visual BASIC

ตารางที่ 2.3 การใช้ระบบตัวเลขใน Visual Basic

เลขฐาน 10	เลขฐาน 8	เลขฐาน16
9	&011	&H9
15	&017	&HF
16	&020	&H10
20	&024	&H14
255	&0377	&HFF

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.19 หลักการตั้งชื่อทั่วไปของ Visual BASIC

ในการใช้งาน Visual BASIC เราจะต้องตั้งชื่อฟอร์มและคอนโทรล ค่าคงที่ตัวแปร ซับรูทีน ฟังก์ชัน โพรซีเยอร์ หรืออื่นๆ อีกมากมาย ชื่อของสิ่งต่างๆ เหล่านี้จะตกอยู่ภายใต้กฎเกณฑ์การตั้งชื่อเดียวกัน คือ

1. จะตั้งขึ้นด้วยตัวอักษรเท่านั้น
2. จะต้องไม่ปรากฏตัวอักษรพิเศษที่ใช้ในการประกาศชนิดของข้อมูลปรากฏอยู่ในชื่อ
3. ความยาวของชื่อจะต้องไม่เกิน 255 ตัวอักษร ในขณะที่ชื่อของฟอร์ม คอนโทรล โมดูลมาตรฐาน และโมดูลคลาสจะต้องไม่เกิน 40 ตัวอักษร
4. จะต้องไม่ซ้ำกับคีย์เวิร์ดทั้งหลายใน Visual BASIC

คีย์เวิร์ด (Keyword)หมายถึง คำที่มีความหมายพิเศษของ Visual BASIC เช่น คำว่า IF, For หรือชื่อของฟังก์ชันที่ Visual BASIC รู้จัก เช่น Len, Abs หรือชื่อของเครื่องหมายต่างๆ เช่น Or, Mod

การตั้งชื่อของฟอร์มและคอนโทรลสามารถมีชื่อซ้ำกับคีย์เวิร์ดต่างๆ ได้ เช่น สามารถตั้งชื่อคอนโทรลว่า For ได้โดยไม่ผิดกฎ เราไม่สามารถอ้างถึงคอนโทรลตัวนี้ได้ในโค้ด โปรแกรม เพราะว่า Visual BASIC จะตีความคำนั้นๆ ว่าเป็นคีย์เวิร์ดของภาษาแทนที่จะเป็นคอนโทรล และนำไปสู่การเกิดข้อผิดพลาดได้ดังโค้ด โปรแกรมตัวอย่าง

```
For.Visible = True              *Program Syntax Error
```

หากจำเป็นจริง ๆ ในการอ้างถึงฟอร์มหรือคอนโทรลที่มีชื่อซ้ำกับคีย์เวิร์ด สามารถกระทำได้ด้วยการกำหนดฟอร์มที่เป็นตัวบรรจุคอนโทรลนั้นก่อนขั้นหนึ่ง หรือใส่เครื่องหมาย “[ ]” รอบชื่อคอนโทรลนั้นๆ

```
“MyForm.For.Visible = True”          ‘Form.Name = For
[For].Visible = True
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 3
ระบบฐานข้อมูล (Database System)

ปัจจุบันมีการใช้คอมพิวเตอร์กันอย่างกว้างขวาง ข้อมูลในด้านต่างๆ ซึ่งในอดีตจัดเก็บอยู่บนกระดาษได้ถูกจัดเก็บไว้ในคอมพิวเตอร์แทน ยิ่งมีแฟ้มข้อมูลจำนวนมากยิ่งทำให้เกิดความซับซ้อนในขบวนการจัดการข้อมูล จึงมีการสร้างระบบจัดเก็บข้อมูลหรือ "ระบบฐานข้อมูล" ขึ้นมา

3.1 ระบบแฟ้มข้อมูล (File System)

ในอดีตองค์กรต่างๆ มักจัดเก็บเอกสารไว้ในแฟ้มเอกสารต่างๆ ซึ่งมีความเกี่ยวข้องกันทางด้านข้อมูลน้อยหรืออาจไม่มีเลย ขึ้นอยู่กับความต้องการในการใช้ข้อมูลนั้นๆ แต่ต่อมาเมื่อองค์กรมีขนาดใหญ่ขึ้นจากเดิมที่สามารถค้นหาเอกสารจากแฟ้มเอกสารเพียงแฟ้มเดียว ก็ต้องค้นหาจากแฟ้มเอกสารจำนวนมากขึ้น ส่งผลให้งานค้นหาเอกสารเป็นงานที่ต้องใช้เวลา และมีความยากลำบากมากขึ้น การจัดเก็บเอกสารในคอมพิวเตอร์ จึงถูกริเริ่มนำมาใช้ในองค์กรแทนการจัดเก็บรูปแบบเดิม แต่การจัดเก็บเอกสารในคอมพิวเตอร์ยุคแรกๆ ยังคงไม่ค่อยมีประสิทธิภาพมากนัก เนื่องจากมีรูปแบบในการจัดเก็บเอกสารคล้ายรูปแบบเดิมอยู่ โดยนำเพียงเอกสารต่างๆในแต่ละแฟ้มเอกสารมาจัดเก็บในรูปของแฟ้มข้อมูลแทน ด้วยเหตุนี้จึงจำเป็นต้องให้ผู้มีความชำนาญ เช่น โปรแกรมเมอร์ หรืออนาไลยเซอร์ เข้ามาช่วยกำหนดโครงสร้างของแฟ้มข้อมูล เพื่อที่จะนำแฟ้มข้อมูลนั้นไปจัดเก็บข้อมูลและนำไปประมวลผลได้ตามความต้องการ

จากบทบาทของคอมพิวเตอร์ที่เข้ามามีอิทธิพลต่อการดำเนินงานภายในองค์กร ได้ส่งผลให้การจัดเก็บข้อมูลในแฟ้มข้อมูลมีการใช้งานแพร่หลายมากยิ่งขึ้น จากเดิมมีเพียง 2 หรือ 3 แฟ้มข้อมูล ได้เพิ่มจำนวนขึ้นเป็น 10 ถึง 20 แฟ้มข้อมูล ดังนั้นจึงจำเป็นต้องมีการเข้ามาควบคุมทางด้านโครงสร้างและการใช้งานแฟ้มข้อมูลต่างๆ ให้มีความเหมาะสมต่อการใช้งานมากขึ้น และรวบรวมแฟ้มข้อมูลเหล่านี้ เข้าเป็นระบบเรียกว่า "ระบบแฟ้มข้อมูล (File System)"

การใช้งานระบบแฟ้มข้อมูล จะต้องอาศัยโปรแกรมเมอร์ พัฒนาโปรแกรมเพื่ออ่านแฟ้มข้อมูลต่างๆขึ้นมาประมวลผล เพื่อให้ผลลัพธ์ตามที่ผู้ใช้ต้องการ ภาษาคอมพิวเตอร์ที่ใช้ในการพัฒนาโปรแกรม โดยทั่วไปจะได้แก่ ภาษาคอมพิวเตอร์ในยุคที่ 3 (Third-generation Language (3GL)) เช่น COBOL, FORTRAN, BASIC ฯลฯ เป็นต้น แต่ภาษาคอมพิวเตอร์เหล่านี้มีข้อจำกัดในการเรียกใช้ข้อมูลจากแฟ้มข้อมูลเนื่องจากภาษาคอมพิวเตอร์เหล่านี้ จะอ้างถึงข้อมูลในแฟ้มข้อมูลตามโครงสร้างทางกายภาพของข้อมูลภายในแฟ้มข้อมูลนั้น เช่นเมื่อต้องการ อ้างถึงข้อมูลในแฟ้มข้อมูลลูกค้าซึ่งมีโครงสร้างดังนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Customer

CUST_NO	CUST_NAME	CUST_ADDRESS	CUST_PHONE
C0001	DEW CORPORATION	166/17 Ladkrabang Bangkok	739-1438
C0002	Dew Industry	123 Raschada Bangkok	738-0292
C0003	TTT Computer	174 Pantip Bankok	277-9297
C0004	CH. (Thai.) Co., Ltd.	143/55 Moo 4 Suratthani	288-619

รูปที่ 3.1 โครงสร้างแฟ้มข้อมูลลูกค้า

ในภาษา COBOL จะต้องมีการนิยาม (Declare) โครงสร้างของ Record เช่น ประเภทของข้อมูล ขนาดของแต่ละ Field ในส่วนของ Data Division ตามโครงสร้างทางกายภาพของ Record ในแฟ้มข้อมูลลูกค้าก่อน จึงจะสามารถอ้างถึง Field ต่างๆ ได้ดังตัวอย่างต่อไปนี้

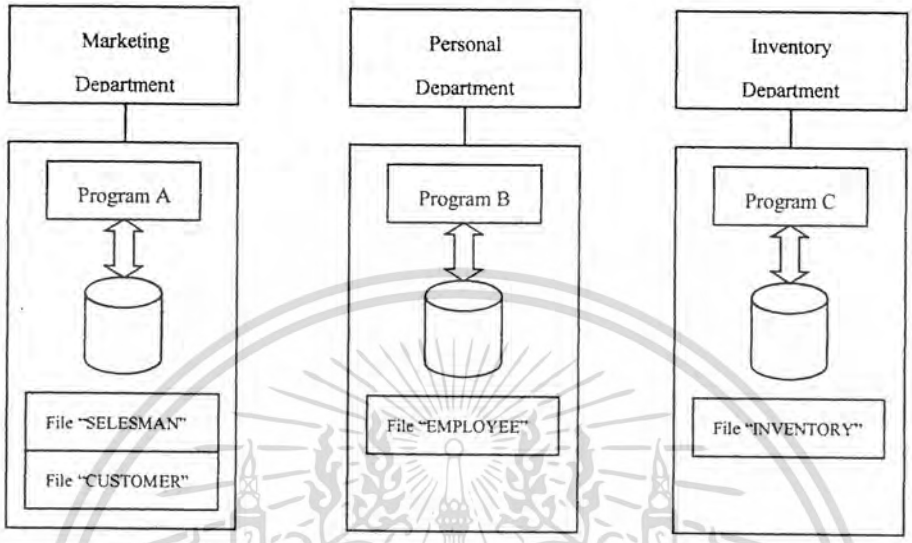
```
01 CUSTOMER
    02 CUST_NO        PIC X(5).
    02 CUST_NAME      PIC X(50).
    02 CUST_ADDRESS   PIC X(50).
    02 CUST_PHONE     PIC X(8).
```

ด้วยเหตุนี้ จึงส่งผลให้การพัฒนา แต่ละโปรแกรมขึ้นใช้งานกับแฟ้มข้อมูลต่างๆ มีความซับซ้อน และต้องใช้เวลาค่อนข้างมาก รวมทั้งทำให้แต่ละโปรแกรมถูกผูกติดอยู่กับข้อมูลต่างๆ ดังนั้น เมื่อมีการเปลี่ยนแปลงโครงสร้างของแฟ้มข้อมูลใดข้อมูลหนึ่ง จึงต้องแก้ไขโปรแกรมต่างๆ ที่มีการเรียกใช้ข้อมูลจากแฟ้มข้อมูลนั้น ตามไปด้วย ส่งผลให้ค่าใช้จ่ายในการบำรุงรักษาค่อนข้างสูง โดยเฉพาะอย่างยิ่งในกรณีที่ต้องมีการแก้ไขโปรแกรมจำนวนมาก

ในยุคเริ่มต้นของการใช้ระบบแฟ้มข้อมูล แต่ละหน่วยงานในองค์กร จะมีการสร้างระบบแฟ้มข้อมูลขึ้นใช้งานภายในหน่วยงานของตนเอง เช่นระบบแฟ้มข้อมูลการขายของฝ่ายการตลาด ระบบแฟ้มข้อมูลพนักงานของฝ่ายการพนักงาน ระบบแฟ้มข้อมูลสินค้าคงคลังของฝ่ายคลังสินค้า ฯลฯ เป็นต้น ซึ่งแต่ละหน่วยงานนั้นจะมีการพัฒนาโปรแกรมที่ใช้ระบบแฟ้มข้อมูลของตนเองขึ้น เพื่อใช้งานภายในหน่วยงานนั้นๆ เช่นฝ่ายการตลาดที่มีการพัฒนาโปรแกรมที่ใช้ข้อมูลจากระบบแฟ้มข้อมูลการขายในการออกใบสั่งซื้อ ใบกำกับสินค้า รายงานแสดงยอดการขายในแต่ละเดือน ฯลฯ หรือฝ่ายคลังสินค้าที่มีการพัฒนาโปรแกรมที่ใช้ข้อมูลจากระบบแฟ้มข้อมูลสินค้าคงคลัง ในการพิมพ์รายงาน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

แสดงยอดคงคลังของสินค้าต่างๆ หรือฝ่ายการพนักงานที่มีการพัฒนาโปรแกรมที่ใช้ข้อมูลจากระบบแฟ้มข้อมูลพนักงานในการคำนวณเงินเดือน คำนวณภาษีเงินได้ ฯลฯ ดังรูป



รูปที่ 3.2 การจัดการข้อมูลของฝ่ายต่างในระบบแฟ้มข้อมูล

3.2 ปัญหาของระบบแฟ้มข้อมูล

ด้วยการจัดเก็บแฟ้มข้อมูลอย่างเป็นเอกเทศ และกระจัดกระจายของระบบแฟ้มข้อมูล เมื่อพิจารณาดูแล้ว จะพบว่าระบบแฟ้มข้อมูลต่างๆ มีความปลอดภัยและความคล่องตัวสูง เนื่องจากมีขนาดเล็กและแยกเก็บภายใน แต่ละหน่วยงานแต่ในขณะเดียวกันได้ก่อให้เกิดปัญหาต่างๆ ขึ้นเช่นเดียวกัน ดังนี้

3.2.1 Data Redundancy

เป็นปัญหาที่เกิดขึ้นเนื่องจากมีการเก็บข้อมูลที่ซ้ำซ้อนกัน กล่าวคือ ข้อมูลชุดเดียวกันถูกจัดเก็บอยู่ใน 2 แฟ้มข้อมูลหรือมากกว่า เช่นข้อมูลของพนักงานที่ชื่อ "ดวง" และ "ใจ" ซึ่งถูกจัดเก็บอยู่ในทั้งแฟ้มข้อมูลพนักงาน ของฝ่ายพนักงาน และข้อมูลพนักงานขายของฝ่ายการตลาด ทั้งๆ ที่ข้อมูลทั้ง 2 เป็นข้อมูลเดียวกัน เนื่องจากพนักงานขายถือเป็นพนักงานคนหนึ่งของบริษัท ซึ่งการซ้ำซ้อนของข้อมูลในลักษณะนี้ จะส่งผลให้เสียพื้นที่ในการจัดเก็บไป

3.2.2 Data Inconsistency

เป็นปัญหาที่มีผลกระทบมาจากการจัดเก็บข้อมูลที่มีความซ้ำซ้อน เนื่องจากการที่มีข้อมูลชุดเดียวกันจัดเก็บอยู่ในหลายแฟ้มข้อมูลอาจทำให้เกิดข้อมูลชุดเดียวกันที่มีค่าต่างกันในต่างแฟ้มข้อมูลได้ ส่งผลทำให้ไม่ทราบว่าข้อมูลใดคือข้อมูลที่ถูกต้อง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.2.3 Data Anomaly

เป็นปัญหาที่มีผลกระทบมาจากการจัดเก็บข้อมูลที่มีความซ้ำซ้อนอีกลักษณะหนึ่ง เนื่องจากการที่มีข้อมูลชุดเดียวกันจัดเก็บอยู่ในหลายแฟ้มข้อมูล อาจส่งผลให้ความสัมพันธ์ระหว่างข้อมูลในแฟ้มข้อมูลต่างๆ สูญเสียไป ในกรณีที่มีการเพิ่ม ลบ หรือเปลี่ยนแปลงค่าของข้อมูลชุดเดียวกันในแฟ้มข้อมูลต่างๆ ที่สัมพันธ์กันไม่ครบถ้วนซึ่งการสูญเสียความสัมพันธ์ระหว่างแฟ้มข้อมูลนี้ เกิดขึ้นได้ 3 ลักษณะดังนี้

1. Modification Anomaly

เป็นการเปลี่ยนแปลงค่าของข้อมูลในแฟ้มข้อมูลต่างๆ ที่สัมพันธ์กันไม่ครบถ้วน

2. Insertion Anomaly

เป็นการกำหนดข้อมูลเพิ่มเติมให้กับแฟ้มข้อมูลต่างๆ ที่สัมพันธ์กันไม่ครบถ้วน

3. Deletion Anomaly

เป็นการลบข้อมูลจากแฟ้มข้อมูลต่างๆ ที่สัมพันธ์กันไม่ครบถ้วน

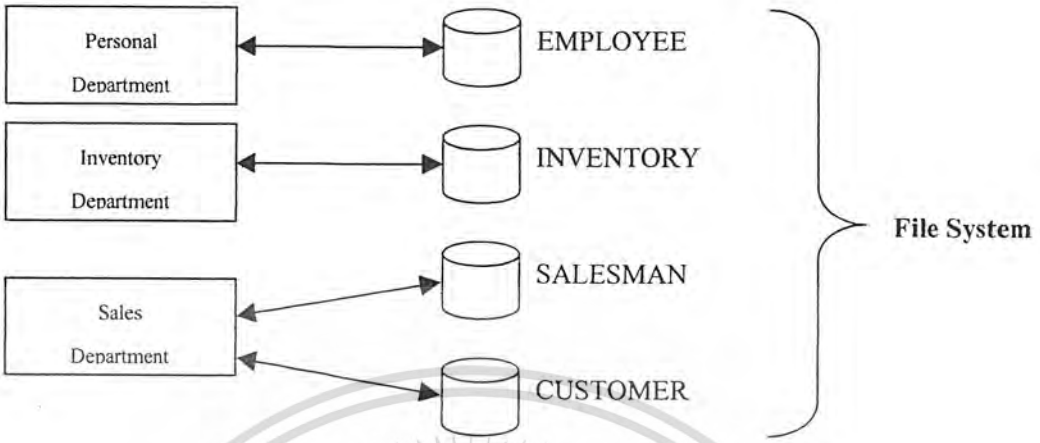
ผลกระทบทั้ง 3 ลักษณะนี้ จะก่อให้เกิด "ข้อมูลที่ไม่มีความสัมพันธ์กับแฟ้มข้อมูลใด เกิดขึ้นในแฟ้มข้อมูลใดแฟ้มข้อมูลหนึ่ง" โดยข้อมูลในลักษณะนี้ จะถือว่าเป็นข้อมูลที่ไม่มีประโยชน์และเป็นการใช้เนื้อที่ในการจัดเก็บอย่างเปล่าประโยชน์

3.3 ระบบฐานข้อมูล (Database Systems) คืออะไร

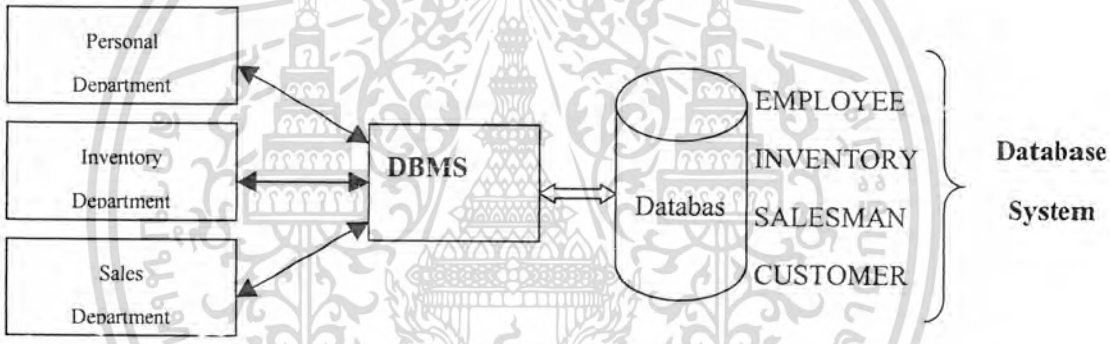
จากปัญหาต่างๆ ที่เกิดในระบบแฟ้มข้อมูล ได้ก่อให้เกิดการจัดเก็บข้อมูลในรูปแบบใหม่ขึ้นที่เรียกว่า "ฐานข้อมูล (Database)" การจัดเก็บข้อมูลในฐานข้อมูลนี้จะแตกต่างกับการจัดเก็บข้อมูลแบบแฟ้มข้อมูล เนื่องจากฐานข้อมูลเป็นการนำเอาข้อมูลต่างๆ ที่มีความสัมพันธ์กัน ซึ่งแต่เดิมจัดเก็บอยู่ในแต่ละแฟ้มข้อมูลมาจัดเก็บไว้ในที่เดียวกันและสามารถแก้ไขปัญหาต่างๆ ที่เกิดขึ้นในระบบแฟ้มข้อมูลได้ ดังรูปที่ 3.3 และ รูปที่ 3.4

ข้อมูลต่างๆ ที่ถูกจัดเก็บเป็นฐานข้อมูล นอกจากจะต้องเป็นข้อมูลที่มีความสัมพันธ์กันแล้ว ยังจะต้องเป็นข้อมูลที่ใช้สนับสนุนการดำเนินงานอย่างน้อยอย่างใดอย่างหนึ่งขององค์กร ดังนั้นจึงอาจกล่าวได้ว่าแต่ละฐานข้อมูลจะเทียบเท่ากับระบบแฟ้มข้อมูล 1 ระบบ และเรียกฐานข้อมูลที่จัดทำขึ้นเพื่อสนับสนุนการดำเนินงานอย่างใดอย่างหนึ่งนั้นว่า "ระบบฐานข้อมูล"

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3.3 ระบบแฟ้มข้อมูล



รูปที่ 3.4 ระบบฐานข้อมูล

3.4 องค์ประกอบของระบบฐานข้อมูล

ระบบฐานข้อมูล โดยทั่วไป จะเกี่ยวข้องกับ 4 ส่วนหลักๆ ดังนี้

3.4.1 ข้อมูล (Data)

ข้อมูลที่จัดเก็บอยู่ในระบบฐานข้อมูล ไม่ว่าจะเป็นเครื่องคอมพิวเตอร์ส่วนบุคคล ไปจนถึงเครื่องคอมพิวเตอร์ขนาดใหญ่ ข้อมูลในแต่ละส่วนจะต้องสามารถนำมาใช้ประกอบกันได้ นอกเหนือจากคุณลักษณะนี้แล้ว ในเครื่องคอมพิวเตอร์ขนาดใหญ่ที่มีผู้ใช้จำนวนมาก ข้อมูลในฐานข้อมูลจะต้องสามารถถูกใช้ร่วมกัน (Data Sharing) จากผู้ใช้หลายๆ คนได้

3.4.2 ฮาร์ดแวร์ (Hardware)

อุปกรณ์ทางคอมพิวเตอร์ที่มีส่วนเกี่ยวข้องกับระบบฐานข้อมูล จะประกอบด้วย 2 ส่วนหลักๆ ดังนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

1. หน่วยความจำสำรอง (Secondary Storage) เนื่องจากเป็นอุปกรณ์ทางคอมพิวเตอร์ที่ใช้จัดเก็บข้อมูล ของฐานข้อมูลดังนั้นสิ่งที่ต้องคำนึงถึงสำหรับอุปกรณ์ในส่วนนี้จึงได้แก่ ความจุของหน่วยความจำสำรองที่นำมาใช้จัดเก็บข้อมูลของฐานข้อมูลนั้น

2. หน่วยประมวลผลและหน่วยความจำหลัก เนื่องจากเป็นอุปกรณ์ที่ต้องทำงานร่วมกัน เพื่อนำข้อมูลจากฐานข้อมูลขึ้นมาประมวลผลตามคำสั่งที่กำหนดดังนั้นสิ่งที่ต้องคำนึงถึงในส่วนนี้จึงได้แก่ความเร็วของหน่วยประมวลผล และขนาดของหน่วยความจำหลักของเครื่องคอม พิวเตอร์ ที่นำมาใช้ประมวลผลร่วมกับฐานข้อมูลนั้น

3.4.3 ซอฟต์แวร์ (Software)

ในการติดต่อข้อมูลภายในฐานข้อมูลของผู้ใช้ จะต้องกระทำผ่านโปรแกรมที่มีชื่อว่า โปรแกรม Database Management System (DBMS) ดังรูป

รูปที่ 3.5 การติดต่อกับระบบฐานข้อมูล

หน้าที่หลักของ DBMS ได้แก่ การทำให้การเรียกใช้ข้อมูลจากฐานข้อมูล เป็นอิสระจากส่วนของ Hardware หรือกล่าวอีกนัยหนึ่ง โปรแกรม DBMS จะมีหน้าที่ในการจัดการและควบคุมความถูกต้อง ความซ้ำซ้อนและความสัมพันธ์ระหว่างข้อมูลต่างๆ ภายในฐานข้อมูลแทนโปรแกรมเมอร์ ส่งผลให้ผู้ใช้สามารถที่จะเรียกใช้ข้อมูลจากฐานข้อมูลได้โดยไม่จำเป็นที่จะต้องทราบถึงโครงสร้างทางกายภาพของข้อมูลในระดับที่ลึกเช่นเดียวกับโปรแกรมเมอร์ เนื่องจาก DBMS นี้ จะมีส่วนของ Query Language ซึ่งเป็นภาษาที่ประกอบด้วยคำสั่งต่างๆ ที่ใช้ในการจัดการและเรียกใช้ข้อมูลจากฐานข้อมูล ซึ่งสามารถนำไปใช้ร่วมกับภาษาคอมพิวเตอร์อื่นๆ เพื่อพัฒนาเป็นโปรแกรมที่ใช้สำหรับเรียกใช้ข้อมูลจากฐานข้อมูลมาประมวลผล

3.4.4 ผู้ใช้ระบบฐานข้อมูล (User)

ผู้ที่เรียกใช้ข้อมูลจากระบบฐานข้อมูลมาใช้งาน สามารถแบ่งออกเป็น 3 กลุ่มได้ดังนี้

1. Application Programmer คือผู้ที่ทำหน้าที่พัฒนาโปรแกรม (Application Program)เพื่อเรียกใช้ข้อมูลจากระบบฐานข้อมูลมาประมวลผลโดยโปรแกรมที่พัฒนาขึ้นส่วนใหญ่ มักจะใช้ร่วมกับคำสั่งในกลุ่ม Data Manipulation Language (DML) ของ Query Language เพื่อเรียกใช้ข้อมูลจากฐานข้อมูล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2. End User คือผู้ที่นำข้อมูลจากฐานข้อมูลไปใช้งาน ซึ่งแบ่งออกเป็น 2 กลุ่มดังนี้

- Naive User ได้แก่ ผู้ใช้ที่เรียกใช้ข้อมูลจากฐานข้อมูลโดยอาศัยโปรแกรมที่พัฒนาขึ้น

- Sophisticated User ได้แก่ ผู้ใช้ที่เรียกใช้ข้อมูลจากระบบฐานข้อมูล ด้วยประโยคคำสั่งของ Query Language ซึ่งโดยทั่งไปผลิตภัณฑ์ทางด้านฐานข้อมูลที่จำหน่ายอยู่ในท้องตลาดจะมีส่วนที่ยอมให้ผู้ใช้ ได้ใช้ประโยคคำสั่งของ Query Language เพื่อเรียกใช้ข้อมูลจากฐานข้อมูลได้โดยตรง สำหรับประโยคคำสั่งเหล่านี้จะถูกส่วน Query Processor ของโปรแกรม DBMS แปลงให้อยู่ในรูปของคำสั่งในกลุ่ม Data Manipulation Language

3. Database Administrator (DBA) ได้แก่ ผู้บริหารที่ทำหน้าที่ควบคุมและตัดสินใจในการกำหนดโครงสร้างของฐานข้อมูล ชนิดของข้อมูล วิธีการจัดเก็บข้อมูล รูปแบบในการเรียกใช้ข้อมูล ความปลอดภัยของข้อมูลและกฎระเบียบที่ใช้ควบคุมความถูกต้องของข้อมูลภายในฐานข้อมูล โดยอาศัยคำสั่งในกลุ่ม Data Definition Language (DDL) ซึ่งเป็นอีกส่วนหนึ่งของ Query Language เป็นตัวกำหนดสำหรับรายละเอียดของคำสั่งในกลุ่ม (DDL)

3.5 Data Independence

ในการเรียกใช้ข้อมูลที่จัดเก็บอยู่ในระบบแฟ้มข้อมูล จะต้องอาศัยโปรแกรมที่เขียนขึ้นเพื่อเรียกใช้ข้อมูลในแฟ้มข้อมูลนั้นโดยเฉพาะ เช่น เมื่อต้องการรายชื่อของพนักงานที่มีเงินเดือนมากกว่า 5,000 บาทต่อเดือน ผู้ใช้จะต้องให้โปรแกรมเมอร์จัดทำโปรแกรม เพื่ออ่านข้อมูลจากแฟ้มข้อมูลพนักงาน และพิมพ์รายงานที่แสดงเฉพาะข้อมูลที่ตรงตามเงื่อนไขที่กำหนด ดังนั้น เมื่อเกิดมีการเปลี่ยนแปลงโครงสร้างทางกายภาพ ของแฟ้มข้อมูลใดแฟ้มข้อมูลหนึ่ง จึงส่งผลให้โปรแกรมต่าง ๆ ที่เรียกใช้ข้อมูลในแฟ้มข้อมูลนั้น ต้องมีการเปลี่ยนแปลงตามไปด้วย เช่น กรณีที่มีการเปลี่ยนแปลงโครงสร้างของ Index File ของแฟ้มข้อมูลพนักงาน จากเดิมซึ่งเรียงลำดับตามรหัสพนักงานมาเป็นเรียงลำดับตามชื่อแทน รายงานที่แสดงรายชื่อพนักงานที่มีเงินเดือนมากกว่า 5,000 บาทต่อเดือนซึ่งแต่เดิมนั้นกำหนดให้เรียงลำดับตามรหัสพนักงาน จึงไม่สามารถพิมพ์ได้ ส่งผลให้ต้องมีการแก้ไขโปรแกรมตามโครงสร้าง Index File ที่เปลี่ยนแปลงไป ซึ่งการที่ข้อมูลและโปรแกรมไม่เป็นอิสระต่อกันนี้ จะเรียกว่า " Data Dependence" หรือความเป็นอิสระของข้อมูล

ภายในระบบฐานข้อมูลไม่สามารถที่จะยอมให้ความไม่เป็นอิสระระหว่างข้อมูล และโปรแกรมเกิดขึ้นได้เนื่องจาก 2 สาเหตุหลัก ๆ ดังนี้

1. เนื่องจากในฐานข้อมูล จะต้องไม่ปรากฏข้อมูลที่ซ้ำซ้อนกันเกิดขึ้น แต่ในแง่ความเป็นจริงแล้ว ข้อมูลที่ผู้ใช้ต้องการ ถึงแม้จะเป็นข้อมูลเดียวกัน ก็อาจต้องการรูปแบบของข้อมูลที่ต่างกันได้ เช่น ผู้ใช้ A ต้องการเงินเดือนในรูปของ Binary ในขณะที่ผู้ใช้ B ต้องการข้อมูลเงินเดือน ในรูปของ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Decimal แทน ซึ่งในกรณีเช่นนี้ จึงจำเป็นที่จะต้องทำให้ข้อมูลและโปรแกรมเป็นอิสระจากกัน เพื่อทำให้รูปแบบของข้อมูล ที่จัดเก็บอยู่ในฐานข้อมูลมีเพียงรูปแบบเดียว แล้วจึงปล่อยให้เป็นหน้าที่ของโปรแกรม DBMS ในการแปลงรูปแบบเป็นไปตามรูปแบบที่ผู้ใช้แต่ละคนต้องการแทน

2. เนื่องจาก DBA มีสิทธิ์ที่จะเปลี่ยนแปลงโครงสร้างของข้อมูลภายในฐานข้อมูลได้ ดังนั้นจึงจำเป็นที่จะต้องทำให้ข้อมูลภายในฐานข้อมูลเป็นอิสระจากโปรแกรมต่าง ๆ ที่เรียกใช้ เพื่อที่จะทำให้ DBA สามารถเปลี่ยนแปลงโครงสร้างของข้อมูลภายในฐานข้อมูลได้ โดยไม่ส่งผลกระทบกับโปรแกรมต่าง ๆ

ด้วยเหตุนี้จึงได้มีการกำหนดให้ข้อมูลภายในฐานข้อมูล จะต้องเป็นอิสระจากตัวโปรแกรมที่เรียกใช้เพื่อที่จะสามารถแก้ไขโครงสร้างทางกายภาพของข้อมูล โดยไม่กระทบต่อโปรแกรมที่เรียกใช้ข้อมูลจากฐานข้อมูลนั้น ซึ่งคุณลักษณะเช่นนี้จะเรียกว่า "Data Independence"

ในการกำหนดให้ข้อมูลเป็นอิสระจากโปรแกรมที่เรียกใช้ จะแบ่งออกเป็น 2 ระดับดังนี้

1. ระดับ Physical เป็นระดับที่โครงสร้างทางกายภาพของข้อมูลเป็นอิสระจากโปรแกรมที่เรียกใช้ เช่นสามารถเปลี่ยนแปลงโครงสร้างของ Index File ได้โดยไม่ต้องแก้ไขโปรแกรมที่เรียกใช้ข้อมูลนั้น

2. ระดับ Logical เป็นระดับที่ความสัมพันธ์ระหว่างข้อมูลในส่วนต่างๆ ภายในฐานข้อมูลเป็นอิสระจากโปรแกรมที่เรียกใช้เช่นสามารถแยกบาง Field ออกไปเป็นแฟ้มข้อมูลใหม่ได้โดยไม่ต้องแก้ไขโปรแกรมที่เรียกใช้ข้อมูลนั้น

เมื่อพิจารณาความเป็นอิสระของข้อมูลที่มีต่อโปรแกรมที่เรียกใช้ทั้ง 2 ระดับจะสังเกตเห็นว่าการกำหนดให้ข้อมูลเป็นอิสระจากโปรแกรมที่เรียกใช้ในระดับ Logical จะกระทำได้ยากกว่าในระดับ Physical เนื่องจากความสัมพันธ์ระหว่างข้อมูลในส่วนต่างๆ ภายในแฟ้มข้อมูลเดียวกัน จะมีความเกี่ยวพันกับโปรแกรมมากกว่าโครงสร้างทางกายภาพของข้อมูล

3.6 Database Management System (DBMS)

เป็นโปรแกรมที่ทำหน้าที่เป็นตัวกลางในการติดต่อระหว่างผู้ใช้กับฐานข้อมูล เพื่อจัดการและควบคุมความถูกต้อง ความซ้ำซ้อน และความสัมพันธ์ระหว่างข้อมูลต่าง ๆ ภายในฐานข้อมูล ซึ่งต่างจากระบบแฟ้มข้อมูลที่หน้าที่เหล่านี้จะเป็นหน้าที่ของโปรแกรมเมอร์ ในการติดต่อกับข้อมูลในฐานข้อมูลไม่ว่าจะด้วยการใช้คำสั่งในกลุ่มคำสั่ง DML หรือ DDL หรือจะด้วยโปรแกรมต่าง ๆ ทุกคำสั่งที่ใช้กระทำกับข้อมูลจะถูกโปรแกรม DBMS นำมาแปล (Compile) เป็นการกระทำ (Operation) ต่าง ๆ ภายใต้คำสั่งนั้น ๆ เพื่อนำไปกระทำกับตัวข้อมูลภายในฐานข้อมูลต่อไป สำหรับส่วนการทำงาน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ต่าง ๆ ภายในโปรแกรม DBMS ที่ทำหน้าที่ในการแปลคำสั่งไปเป็นการกระทำต่าง ๆ ที่จะกระทำกับข้อมูลนั้น ประกอบด้วยส่วนการทำงานต่าง ๆ ดังนี้

1. Database Manager เป็นส่วนที่ทำหน้าที่กำหนดการกระทำต่าง ๆ ให้กับส่วน File Manager เพื่อไปกระทำกับข้อมูลที่เก็บอยู่ในฐานข้อมูล (File Manager เป็นส่วนที่ทำหน้าที่บริหาร และจัดการกับข้อมูลที่เก็บอยู่ในฐานข้อมูลในระดับกายภาพ)

2. Query Processor เป็นส่วนที่ทำหน้าที่แปลงประโยคคำสั่งของ Query Language ให้อยู่ในรูปของคำสั่งที่ Database Manager เข้าใจ

3. Data Manipulation Language Precompiler เป็นส่วนทีทำหน้าที่แปล (Compile) ประโยคคำสั่งของกลุ่มคำสั่ง DML ให้อยู่ในรูปแบบที่ส่วน Application Programs Object Code จะนำไปเข้ารหัสเพื่อส่งต่อไปยังส่วน Database Manager ในการแปลประโยคคำสั่งของกลุ่มคำสั่ง DML ของส่วน Data Manipulation Language Precompiler นี้จะต้องทำงานร่วมกับส่วน Query Processor

4. Data Definition Language Precompiler เป็นส่วนที่ทำหน้าที่แปล (Compile) ประโยคคำสั่งของกลุ่มคำสั่ง DDL ให้อยู่ในรูปแบบของ MetaData ที่เก็บอยู่ในส่วน Data Dictionary ของฐานข้อมูล (MataData ได้แก่ รายละเอียดที่บอกถึงโครงสร้างต่าง ๆ ของข้อมูล)

5. Application Programs Object Code เป็นส่วนที่ทำหน้าที่แปลงคำสั่งต่าง ๆ ของโปรแกรม รวมทั้งคำสั่งในกลุ่มคำสั่ง DML ที่ส่งต่อมาจากส่วน Data Manipulation Language Procompiler ให้อยู่ในรูปของ Object Code ที่จะส่งต่อไปให้ Database Manager เพื่อกระทำกับข้อมูลในฐานข้อมูล

โปรแกรม DBMS นี้ได้ถูกพัฒนาขึ้นมาเพื่อแก้ปัญหาทางด้าน Data Independence ที่ไม่มีในระบบแฟ้มข้อมูล ดังนั้นจึงมีความเป็นอิสระจากทั้งตัว Hardware และตัวข้อมูลภายในฐานข้อมูล กล่าวคือ โปรแกรม DBMS จะมีการทำงานที่ไม่ขึ้นกับรูปแบบ (Platform) ของตัว Hardware ที่นำมาใช้กับระบบฐานข้อมูล รวมทั้งมีรูปแบบใน การอ้างถึงข้อมูลที่ไม่ขึ้นอยู่กับโครงสร้างทางกายภาพของข้อมูล ด้วยการใช้ Query Language ในการติดต่อกับข้อมูลในฐานข้อมูลแทนคำสั่งของภาษาคอมพิวเตอร์ในยุคที่ 3 ส่งผลให้ผู้ใช้สามารถเรียกใช้ข้อมูลจากฐานข้อมูลโดยไม่ จำเป็นต้องทราบถึงประเภทของข้อมูลหรือขนาดของข้อมูลนั้น หรือสามารถกำหนดลำดับที่ของ Field ในการแสดงผลได้โดยไม่ต้องคำนึงถึงลำดับที่จริงของ Field นั้น

3.7 หน้าที่ของ DBMS

สำหรับหน้าที่ของโปรแกรม DBMS มีดังนี้

1. ทำหน้าที่แปลงคำสั่งที่ใช้จัดการกับข้อมูลภายในฐานข้อมูล ให้อยู่ในรูปแบบที่ฐานข้อมูลเข้าใจ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2. ทำหน้าที่ในการนำคำสั่งต่าง ๆ ซึ่งได้รับการแปลแล้ว ไปสั่งให้ฐานข้อมูลทำงานเช่นการเรียกใช้ข้อมูล (Retrieve) การจัดเก็บข้อมูล (Update) การลบข้อมูล (Delete) การเพิ่มข้อมูล (Add) เป็นต้น

3. ทำหน้าที่ป้องกันความเสียหายที่จะเกิดขึ้นกับข้อมูลภายในฐานข้อมูล โดยจะคอยตรวจสอบว่าคำสั่งใดที่สามารถทำงานได้ และคำสั่งใดที่ไม่สามารถทำงานได้

4. ทำหน้าที่รักษาความสัมพันธ์ของข้อมูลภายในฐานข้อมูลให้มีความถูกต้องอยู่เสมอ

5. ทำหน้าที่เก็บรายละเอียดต่าง ๆ ที่เกี่ยวข้องกับข้อมูลภายในฐานข้อมูลไว้ใน Data Dictionary ซึ่งรายละเอียดเหล่านี้จึงมักจะถูกเรียกว่า "ข้อมูลของข้อมูล" (MetaData)

6. ทำหน้าที่ควบคุมให้ฐานข้อมูลทำงานได้อย่างถูกต้องและประสิทธิภาพ

3.8 Data Dictionary และ File Manager

ทุกฐานข้อมูลจะต้องมีส่วนที่ใช้เก็บข้อมูลในลักษณะ MetaData ซึ่งเป็นข้อมูลที่บอกถึงรายละเอียดของ ตัวข้อมูลที่เก็บอยู่ในฐานข้อมูล เช่น โครงสร้างของข้อมูล โครงสร้างของ Table โครงสร้างของ Index กฎที่ใช้ควบคุมความถูกต้องของข้อมูล (Integrity Rule) กฎที่ใช้ในการรักษาความปลอดภัยให้กับข้อมูล (Security Rule) ฯลฯ ข้อมูลเหล่านี้จัดเป็นข้อมูลที่มีความจำเป็นต่อโปรแกรม DBMS ในการตัดสินที่จะดำเนินการใด ๆ กับฐานข้อมูล เช่น ข้อมูลที่เกี่ยวข้องกับกฎที่ใช้ในการรักษาความปลอดภัยให้กับข้อมูล จะถูกนำมาใช้ในการพิจารณาให้สิทธิแก่ผู้ใช้ในการใช้งานฐานข้อมูล เป็นต้น สำหรับส่วนที่ใช้จัดเก็บข้อมูลในลักษณะของ MetaData นี้ได้แก่ Data Dictionary หรือ Catalog

สำหรับ File Manger เป็นส่วนที่ทำหน้าที่บริหารและจัดการกับข้อมูลที่เก็บอยู่ในฐานข้อมูลในระดับกายภาพ (Physical Level)

3.9 ประโยชน์ของฐานข้อมูล

การจัดนำข้อมูลที่มีความสัมพันธ์กันมาใช้ร่วมกันเป็นฐานข้อมูลนั้น จะก่อให้เกิดประโยชน์ดังนี้

1. สามารถลดความซ้ำซ้อนของข้อมูล (Data Redundancy) โดยไม่จำเป็นต้องจัดเก็บข้อมูลที่ซ้ำซ้อนกันไว้ในระบบแฟ้มข้อมูลของแต่ละหน่วยงานเหมือนเช่นเดิม แต่สามารถนำข้อมูลมาใช้ร่วมกันในคุณลักษณะ Integrated แทน

2. สามารถหลีกเลี่ยงความขัดแย้งของข้อมูล (Data Inconsistency) เนื่องจากไม่ต้องจัดเก็บข้อมูลที่ซ้ำซ้อนกันในหลายแฟ้มข้อมูลดังนั้นการแก้ไขข้อมูลในแต่ละชุดจะไม่ก่อให้เกิดค่าที่แตกต่างกันได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3. แต่ละหน่วยงานในองค์กร สามารถใช้ข้อมูลร่วมกันได้

4. สามารถกำหนดให้ข้อมูลมีรูปแบบที่เป็นมาตราฐานเดียวกันได้ เพื่อให้ผู้ใช้ข้อมูลในฐานข้อมูลชุดเดียวกันสามารถเข้าใจและสื่อสารถึงความหมายเดียวกัน

5. สามารถกำหนดระบบความปลอดภัยให้กับข้อมูลได้ โดยกำหนดระดับความสามารถในการเรียกใช้ข้องมูลของผู้ใช้แต่ละคน ให้แตกต่างกันตามความรับผิดชอบ

6. สามารถรักษาความถูกต้องของข้อมูลได้ โดยระบุกฎเกณฑ์ในการควบคุมความผิดพลาดที่อาจเกิดขึ้นจากการป้อนข้อมูลผิด

7. สามารถตอบสนองต่อความต้องการใช้ข้อมูลในหลายรูปแบบ

8. ทำให้ข้อมูล เป็นอิสระจากโปรแกรมที่ใช้งานข้อมูลนั้น (Data Independence) ซึ่งผลส่งให้ผู้พัฒนาโปรแกรมสามารถแก้ไขโครงสร้างของข้อมูล โดยไม่กระทบต่อโปรแกรมที่เรียกใช้งานข้อมูลนั้น เช่น ในกรณีที่ต้องการเปลี่ยนขนาดของ Field สำหรับระบบแฟ้มข้อมูล จะกระทำได้ยาก เนื่องจากต้องเปลี่ยนแปลงตัวโปรแกรมที่อ้างถึง Field นั้นทั้งหมดซึ่งต่างจากการใช้ระบบฐานข้อมูล ที่การอ้างถึงข้อมูลจะไม่ขึ้นกับโครงสร้างทางกายภาพของข้อมูล จึงไม่ส่งผลให้ต้องแก้ไขโปรแกรมที่เรียกใช้ข้อมูลนั้นมากนัก

3.10 ระบบฐานข้อมูลแบบ Relational

โครงสร้างข้อมูลของฐานข้อมูลแบบต่าง ๆ ได้ถูกหลายบริษัท อาทิเช่น IBM, Digital Equipment, Microsoft, Ingress Division of The ASK Group, Computer Associates International Oracle ฯลฯ นำไปใช้เป็นต้นแบบในการออกแบบผลิตภัณฑ์ทางด้านฐานข้อมูล เพื่อออกมาจำหน่ายแข่งขันกันในท้องตลาด แต่ถ้ากล่าวถึงโครงสร้างซึ่งเป็นที่นิยมมากที่สุด คงจะไม่มีนักออกแบบฐานข้อมูลคนใดที่จะไม่กล่าวถึงฐานข้อมูลที่มีโครงสร้างข้อมูลแบบ Relational

รายละเอียดต่าง ๆ ของฐานข้อมูลที่มีโครงสร้างข้อมูลแบบ Relational ได้ถูกนิยามไว้ในแบบจำลองทางด้านฐานข้อมูลที่มีชื่อว่า Relational Model แบบจำลองนี้ถูกคิดค้นโดย Dr. E. F. Codd นักคณิตศาสตร์ในช่วงปี ค.ศ. 1968 ซึ่งขณะนั้นทำหน้าที่เป็นนักวิจัยในบริษัท IBM ด้วยแนวคิดที่ว่าเมื่อนิยามทฤษฎีต่าง ๆ ให้กับโครงสร้างของฐานข้อมูลในรูปแบบของคณิตศาสตร์แล้ว จะสามารถแก้ไขข้อบกพร่องต่าง ๆ ที่เกิดขึ้นกับฐานข้อมูลในแบบเดิมได้

ใน Relational Model ได้นิยามคำว่า "ข้อมูล" แตกต่างจากฐานข้อมูลในแบบอื่น ๆ เพื่อลดความสับสนในด้านความหมายที่ปรากฏอยู่ในฐานข้อมูลประเภทอื่น เช่น คำว่า " Record" ซึ่งสามารถใช้แทนข้อมูลได้หลายความหมาย ขึ้นอยู่กับรูปแบบของการนำไปใช้ กล่าวคือ เมื่อนำไปใช้กับภาษา COBOL ซึ่งเป็นภาษาสำหรับเขียนโปรแกรมคอมพิวเตอร์ภาษาหนึ่ง คำว่า " Record" อาจหมายถึง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ข้อมูล 1 รายการในลักษณะ Repeating Group (Repeating Group หมายถึง Record ที่มีค่าของข้อมูลในบาง Field มากกว่า 1 ค่า) หรือในลักษณะ Flat Record (Flat Record หมายถึง Record ที่มีโครงสร้างเพียงโครงสร้างเดียว) หรือเมื่อนำคำว่า " Record" ไปใช้ในการอธิบายโครงสร้างของข้อมูล 1 รายการ คำว่า " Record" อาจหมายถึง ข้อมูลในระดับแนวความคิด ซึ่งจะมองข้อมูล 1 Record ว่าประกอบด้วย Field อะไรบ้าง หรืออาจหมายถึง ข้อมูลในระดับกายภาพ ซึ่งจะมองข้อมูล 1 Record ว่าประกอบด้วยจำนวนกี่ Byte และแต่ละ Byte มีหน้าที่อะไร เป็นต้น ดังนั้น เพื่อลดความสับสนดังกล่าว Dr. Codd จึงนิยามคำว่า Relation, Tuple, Attribute, Degree, Cardinality และ Domain ขึ้นเพื่ออธิบายถึงคำว่าข้อมูลใน Relational Model แทน

3.11 Relation

Relation มักจะถูกเรียกว่า "Table" เนื่องจาก "Relation" เป็นหน่วยที่ใช้จัดเก็บข้อมูลที่อยู่ในรูปแบบของตารางขนาด 2 มิติ ที่ประกอบด้วยแถว (Row) และสดมภ์ (Column) แถวของ Relation ได้แก่ ข้อมูล 1 รายการซึ่งเทียบเท่ากับ Record ในระบบแฟ้มข้อมูล ส่วนแต่ละสดมภ์ของ Relation ได้แก่ คุณลักษณะต่าง ๆ ของข้อมูลในแต่ละแถว ซึ่งเทียบเท่ากับ Field ของ Record ในระบบแฟ้มข้อมูล สำหรับชื่อของแต่ละแถวของ Relation จะถูกเรียกว่า " Tuple" ส่วนชื่อของแต่ละสดมภ์ของ Relation จะถูกเรียกว่า " Attribute" เช่น Relation ชื่อ " EMPLOYEE" ที่ใช้เก็บประวัติของพนักงานแต่ละคนในบริษัท ที่ประกอบด้วย รหัสพนักงาน ชื่อ นามสกุล เพศ เงินเดือน และ แผนกที่สังกัด

สำหรับข้อแตกต่างระหว่าง Relation กับแฟ้มข้อมูลในระบบแฟ้มข้อมูล มีดังนี้

1. Relation จะเป็นส่วนที่จัดเก็บข้อมูลในระดับแนวความคิด ดังนั้น ผู้ใช้ในทุกระดับไม่ว่าจะเป็น นักวิเคราะห์ระบบ โปรแกรมเมอร์ หรือผู้ใช้ทั่วไป จะมองข้อมูลที่จัดเก็บอยู่ใน Relation นี้ในรูปของตารางเช่นเดียวกัน ทำให้เวลานำข้อมูลจาก Relation ไปใช้งาน หรือจัดเก็บข้อมูลลงใน Relation ผู้ที่นำข้อมูลไปใช้หรือจัดเก็บ ไม่จำเป็นที่จะต้องทราบถึงโครงสร้างทางกายภาพของ Relation นั้น ซึ่งต่างจากแฟ้มข้อมูลในระบบแฟ้มข้อมูล ที่ผู้ใช้ในแต่ละระดับ จะมีมุมมองต่อข้อมูลที่จัดเก็บอยู่ในแฟ้มข้อมูลเดียวกันต่างกัน ดังที่กล่าวมาแล้ว ดังนั้น Relation ใน Relational Model จึงมีความเป็นอิสระจากทั้งส่วนของ Software และ Hardware

2. การจัดเก็บ Relation ในหน่วยความจำสำรองจะอยู่ในที่เดียวกัน ซึ่งต่างจากแฟ้มข้อมูลในระบบแฟ้มข้อมูล ที่แต่ละแฟ้มข้อมูล จะจัดเก็บอยู่อย่างกระจัดกระจาย สำหรับ Relation ที่ข้อมูลมีความสัมพันธ์กัน และนำมาจัดเก็บรวมกันนี้ จะเป็นฐานข้อมูล 1 ฐานข้อมูล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.12 Domain

เป็นการนิยามขอบเขตของค่าที่เป็นไปได้ให้กับข้อมูลในแต่ละ Attribute เพื่อป้องกันไม่ให้เกิดการป้อนข้อมูลที่เกินขอบเขตที่กำหนด เช่น การกำหนดให้ค่าของเงินเดือนของพนักงาน จะต้องมีค่ามากกว่า 0 เพื่อป้องกันไม่ให้ผู้ใช้ ป้อนจำนวนเงินที่เป็น 0 หรือมีค่าติดลบ ให้กับข้อมูลในส่วนเงินเดือนของพนักงาน หรือ การกำหนดให้เพศของพนักงานแต่ละคนจะต้องมีค่าเป็น ชาย (M) หรือ หญิง (F) เท่านั้น

การกำหนด Domain ให้กับข้อมูล จะมีข้อกำหนดต่าง ๆ ดังนี้

1. ค่าที่นิยมให้กับข้อมูลจะต้องมีค่าเป็น Scalar กล่าวคือ จะต้องเป็นค่าที่มีความหมายในหน่วยที่เล็กที่สุด ที่ไม่ปรากฏโครงสร้าง ที่สามารถแยกย่อยออกเป็นโครงสร้างย่อย ๆ ได้อีก เช่น ที่อยู่ซึ่งสามารถแยกย่อยออกเป็นบ้านเลขที่ ถนน อำเภอ จังหวัด ฯลฯ จึงไม่ใช่ข้อมูลที่มีค่าเป็น Scalar และจึงไม่สามารถนำมากำหนดเป็น Domain ให้กับข้อมูลในฐานข้อมูลได้ ซึ่งแตกต่างจากจังหวัดที่ไม่สามารถแยกย่อยออกเป็นข้อมูลส่วนย่อย ๆ ได้อีก จึงสามารถนำมากำหนดเป็น Domain ให้กับข้อมูลในฐานข้อมูลได้ สำหรับข้อมูลที่มีค่าเป็น Scalar นี้จะเรียกข้อมูลนั้นว่า มีคุณลักษณะของ Atomicity

2. ข้อมูลที่สามารถนำมากำหนด Domain ได้ จะต้องเป็นข้อมูลที่เป็นอิสระจากข้อมูลอื่นเช่น รหัสพนักงานของพนักงานแต่ละคน ชื่อของจังหวัดตามรหัสจังหวัด จำนวนสินค้าในรายการสั่งซื้อ ฯลฯ ส่วนรหัสของฝ่ายที่พนักงานแต่ละคนสังกัด ซึ่งจะต้องมีค่าเป็นไปตามรหัสของฝ่ายที่กำหนดใน Relation " DEPARTMENT"

3. ข้อมูลที่สามารถนำมากำหนด Domain ได้ จะต้องเป็นข้อมูลประเภทเดียวกัน เช่น เพศที่ Domain ของข้อมูลต้องเป็นเพศชายหรือหญิง แต่ถ้ากำหนดให้ค่าของเพศชาย สามารถเป็นได้ทั้ง "0" หรือ "True" และค่าของเพศหญิง สามารถเป็นได้ทั้ง "1" หรือ "False" แล้ว ข้อมูลเพศนี้ จะไม่สามารถนำมากำหนด Domain ให้กับข้อมูลได้ เนื่องจากประเภทของข้อมูลสามารถเป็นได้ทั้งตัวเลขและข้อมูลตรรกะ

4. Domain ที่กำหนดให้กับ Attribute ที่จะต้องถูก Attribute อื่นอ้างถึง สามารถถ่ายทอด Domain ของตนให้กับ Attribute ในอีก Relation หนึ่งที่อ้างถึงไปด้วย เช่น Attribute " DeptID" ของ Relation " DEPARTMENT" ที่จะถูก Attribute ชื่อเดียวกันใน "EMPLOYEE" อ้างถึง สามารถที่จะถ่ายทอด Domain ของตนเองให้กับ Attribute ชื่อเดียวกันใน Relation "EMPLOYEE" ไปด้วย

5. ค่าของ Domain ที่กำหนดให้กับข้อมูล ไม่จำเป็นที่จะต้องปรากฏอยู่ในข้อมูลนั้น ๆ เช่น กรณีที่กำหนดให้ Domain ของเงินเดือนพนักงาน จะต้องมีค่ามากกว่า 0 แต่ก็มิได้หมายความว่าทุกจำนวนที่มีค่ามากกว่า 0 จะต้องปรากฏเป็นข้อมูลใน Attribute ดังกล่าว

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ประโยชน์ของการกำหนด Domain ให้กับข้อมูล นอกเหนือจากจะเป็นการกำหนดค่าที่เป็นไปได้ ที่ผู้ใช้สามารถกำหนดให้กับข้อมูลในส่วนนั้น ๆ เพื่อป้องกันไม่ให้ผู้ใช้ป้อนข้อมูลเกินขอบเขตที่กำหนดไว้แล้ว ยังสามารถสร้างความเชื่อมั่นในการนำข้อมูลที่สัมพันธ์กันมาเปรียบเทียบกันได้อีกด้วย เช่น ในกรณีที่ต้องการทราบถึงฝ่ายที่พนักงานแต่ละคนสังกัด ซึ่งจะต้องอาศัยการนำเอาค่าของ Attribute "DeptID" ใน Relation "EMPLOYEE" ไปเปรียบเทียบกับค่าของ Attribute ชื่อเดียวกันใน Relation "DEPARTMENT" ซึ่งการเปรียบเทียบในลักษณะนี้ จะเป็นการเปรียบเทียบในแง่ของความหมายของข้อมูลมากกว่าเป็นเพียงการเปรียบเทียบการเท่ากันของข้อมูล โดยทั่วไปเนื่องจาก Domain ของข้อมูล จะทำให้เชื่อมั่นได้ว่า ทุกค่าใน Attribute "DeptID" ของ Relation "EMPLOYEE" จะต้องปรากฏอยู่ใน Tuple หนึ่งของ Relation " DEPARTMENT"

3.13 คุณสมบัติของ Relation

คุณสมบัติของ Tuple และ Attribute ของแต่ละ Relation จะประกอบด้วย

1. เนื่องจาก Relation ใน Relational Model อยู่ในรูปแบบของเซตทางคณิตศาสตร์ ที่ภายในเซตจะต้องประกอบด้วยสมาชิกที่มีค่าไม่ซ้ำกัน ดังนั้น ภายใน Relation ใด ๆ จึงต้องมี Attribute ใด Attribute หนึ่ง ที่ทำให้แต่ละ Tuple ใน Relation มีข้อมูลที่ไม่ซ้ำกัน เช่น Relation " POPULAR" ที่ใช้เก็บข้อมูลของประชากรในประเทศไทยซึ่งถึงแม้ว่าจะมีบุคคลที่มีชื่อและนามสกุลที่ซ้ำกัน เช่น ประชากรที่ชื่อ " นายสมบูรณ์ สุขมาก" แต่ข้อมูลในแต่ละ Tuple ของ Relation "POPULAR" ก็จะไม่ปรากฏข้อมูลที่ซ้ำกัน เนื่องจากค่าของข้อมูลใน Attribute "ID" ซึ่งใช้เก็บหมายเลขบัตรประจำตัวประชาชน มีข้อมูลที่ไม่ซ้ำกันดังตัวอย่างต่อไปนี้

ID	Name	Surname	Sex
1230000100	สมบูรณ์	สุขมาก	M
1230000101	สมเกียรติ	เจริญพร	M
1230000102	สมบูรณ์	สุขมาก	M
1230000103	น้ำฝน	ม่วงทอง	F

รูปที่ 3.6 Attribute "ID" ที่ไม่ซ้ำกัน

2. ด้วยเหตุผลเช่นเดียวกับข้อที่ 1 ลำดับที่ของสมาชิกภายในเซตใด ๆ จะไม่มีผลต่อเซตนั้น ดังนั้น ภายใน Relation จึงไม่มีการกำหนดลำดับที่ให้กับแต่ละ Tuple ใน Relation กล่าวคือ จะไม่มีการกล่าวถึงคำว่า Tuple แรก หรือ Tuple สุดท้าย หรือ Tuple ลำดับที่ 5 หรือ Tuple ถัดไป หรือ Tuple ที่ผ่านมาใน Relation

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3. ภายใน Relation จะไม่มีการกำหนดลำดับที่ให้กับแต่ละ Attribute เนื่องจากในการอ้างถึง Attribute ใน Relation จะใช้ชื่อของ Attribute นั้นในการอ้างถึง ดังนั้น จึงไม่มีการกล่าวถึงคำว่า Attribute แรก หรือ Attribute สุดท้ายหรือ Attribute ลำดับที่ 5 หรือ Attribute ที่ผ่านมา หรือ Attribute ถัดไปใน Relation เช่นเดียวกับ Tuple

4. ค่าในทุก Attribute ของ Relation จะต้องมีคุณสมบัติ Atomicity ซึ่งเป็นคุณสมบัติที่กำหนดให้ค่าของข้อมูลในแต่ละ Attribute ของ Relation จะต้องมีความหมายใดความหมายหนึ่งเพียงความหมายเดียว ไม่ใช่กลุ่มของสิ่งใดสิ่งหนึ่ง หรือกล่าวอีกนัยหนึ่ง ข้อมูลในแต่ละ Attribute ของ Relation จะต้องไม่ใช่ข้อมูลในลักษณะ Repeating Group เช่น กรณีที่พนักงานสามารถสังกัดฝ่ายได้มากกว่า 1 ฝ่าย ข้อมูลใน Attribute "DeptID" ของแต่ละ Tuple ของ Relation "EMPLOYEE" ซึ่งเก็บรหัสของฝ่ายที่พนักงานแต่ละคนสังกัด จะไม่สามารถจัดเก็บทุกรหัสของฝ่ายที่พนักงานคนนั้นสังกัดภายใน Tuple เดียว สำหรับคุณสมบัติของ Relation ในข้อนี้ ได้บังคับให้ทุก Relation มีรูปแบบการจัดเก็บข้อมูลที่เป็นไปตามรูปแบบแรก (First Normal Form) ตามที่กำหนดไว้ในการทำ Normalization ให้กับ Relation ต่าง ๆ

5. ชื่อของแต่ละ Attribute ใน Relation เดียวกัน จะต้องมีชื่อที่ไม่ซ้ำกัน

6. ค่าที่ปรากฏในแต่ละ Attribute ใน Relation เดียวกัน จะต้องใช้แทนข้อมูลที่มีความหมายเดียวกัน เช่น Attribute "EmpID" ของ Relation "EMPLOYEE" จะต้องใช้เก็บข้อมูลรหัสพนักงานของแต่ละ Tuple เท่านั้น จะไม่สามารถใช้เก็บข้อมูลอื่น ๆ เช่น เพศ ได้

3.14 Degree

ได้แก่ จำนวนของ Attribute หรือกล่าวอีกนัยหนึ่ง คือ จำนวน Domain ในแต่ละ Relation ยกตัวอย่างเช่น Relation "EMPLOYEE" ซึ่งมีข้อมูลดังนี้

EMPLOYEE

EmpID	Name	Surname	DeptID
00001	สมบูรณ์	สุขมาก	01
00002	สมเกียรติ	เจริญพร	02
00003	จันจิรา	แจ้งเกิด	03
00004	น้ำฝน	ม่วงทอง	01

รูปที่ 3.7 ลักษณะ Degree = 4

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Degree ของ Relation "EMPLOYEE" นี้ จะมีค่าเท่ากับ 4 เนื่องจากมีอยู่ 4 Attribute คำว่า Degreeนี้ ในบางครั้งอาจเรียกเป็น "Arity" แทน ดังนั้น Relation ที่มี Degree เท่ากับ 1 จึงอาจถูกเรียกว่า " Unary" ส่วน Relation ที่มี Degree เท่ากับ 2 จึงอาจถูกเรียกว่า " Binary" ที่มี Degree เท่ากับ 3 จึงอาจถูกเรียกว่า "Ternary" จนกระทั่ง Relation ที่มี Degree เท่ากับ n จึงอาจถูกเรียกว่า "n – array"

3.15 Key

เนื่องจากข้อมูลที่เก็บอยู่ใน Relation ที่ต่างกัน จะสามารถนำมาอ้างถึงกันได้นั้น ข้อมูลใน Relation ที่ถูกอ้างถึงจะต้องประกอบด้วง Tuple ที่มีเอกลักษณ์เฉพาะตัว หรือกล่าวอีกนัยหนึ่งจะต้องเป็น Tuple ที่มีค่าของ Attribute หรือชุดของ Attribute ที่ทำให้ข้อมูลในแต่ละ Tuple ของ Relation นั้นมีค่าที่ไม่ซ้ำกัน เพื่อที่จะทำให้การอ้างถึงข้อมูลนั้น สามารถบ่งบอกได้ว่าข้อมูลใดสัมพันธ์กับข้อมูลใดโดยไม่เกิดการสับสน เช่น กรณีที่ต้องการทราบที่อยู่ของลูกค้าที่ชื่อ "สมชาย" เพื่อนำมาพิมพ์ลงในใบกำกับสินค้า แต่ปรากฏว่าใน Relation" CUSTOMER" ปรากฏ Tuple ของลูกค้าที่ชื่อสมชายอยู่ 3 คน ดังนั้น จึงไม่สามารถระบุได้ว่า Tuple ใดเป็น Tuple ของลูกค้าที่ต้องการ แต่ถ้าเปลี่ยนมาระบุรหัสของลูกค้าแทน โดยระบุเป็นรหัส "C01" แล้ว ก็จะส่งผลให้เหลือเพียงข้อมูลของ "นายสมชาย เข็มเงิน" เพียงคนเดียว ซึ่งสามารถนำไปหาที่อยู่ที่ต้องการได้ สำหรับคุณสมบัติที่ทำให้ข้อมูลมีเอกลักษณ์เฉพาะตัวนี้ได้แก่ Key

Key ได้แก่ Attribute หรือชุดของ Attribute ที่ทำให้ข้อมูลแต่ละ Tuple ใน Relation มีค่าของข้อมูลที่ไม่ซ้ำกัน (Unique) เช่น Attribute "CustID" ของ Relation " CUSTOMER" ซึ่งใช้แทนรหัสลูกค้าแต่ละคนจะมีรหัสประจำตัวที่ไม่ซ้ำกัน การใช้ key หับ Relation นี้ จัดเป็นพื้นฐานในการกำหนด General Integrity Rule ขึ้นเพื่อควบคุมให้การอ้างถึงระหว่างข้อมูลใน Relation ต่าง ๆ มีความถูกต้องและเชื่อมั่นได้ สำหรับ key ที่ใช้ใน Relational Model ได้แก่ Candidate Key, Primary Key, Alternate Key และ Foreign Key

3.16 Candidate Key

Candidate Key คือ Key ขนาดเล็กที่สุดที่ทำให้ข้อมูลในแต่ละ Tuple ของ Relation มีค่าของข้อมูลที่ไม่ซ้ำกัน ดังนั้นจึงกล่าวได้ว่า Candidate Key เป็นส่วนพื้นฐานที่จำเป็นต่อ General Integrity Rule เนื่องจาก Candidate Key จัดเป็นเครื่องมือที่สามารถระบุตำแหน่งของแต่ละ Tuple ใน Relation ได้ จึงเป็นหลักประกันได้ว่า เมื่อมีการกระทำที่ใช้ค่าของ Candidate Key เพื่อเรียกใช้ข้อมูลใน Relation แล้ว ผลลัพธ์ที่ได้ จะปรากฏข้อมูลออกมาเพียง Tuple เดียว ดังนั้นจึง สามารถเชื่อมั่นได้ว่า เมื่อนำ Relation ที่มี Candidate Key มาสัมพันธ์กัน จะไม่เกิดการสับสนในกรณีที่ Tuple ใน Relation หนึ่ง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ไม่ทราบว่าจะจับคู่กับ Tuple ใดในอีก Relation หนึ่งที่มีความเกี่ยวข้องกันอันเนื่องมาจากมีมากกว่า 1 Tuple ที่มีข้อมูลที่เหมือนกัน เช่น Relation " DEPARTMENT" ที่ไม่มีการกำหนด Candidate Key

สำหรับคุณสมบัติของ Candidate Key ประกอบไปด้วย

1. โดยปกติ เมื่อกล่าวในแง่ทฤษฎีของ Relation แล้ว แต่ละ Relation ควรที่จะมี Candidate Key เพียง Candidate Key เดียว แต่อย่างไรก็ตาม ในแง่ความเป็นจริงแล้ว Relation สามารถมี Candidate Key ได้ มากกว่า 1 Candidate Key ถ้า Relation นั้นปรากฏ Attribute ที่สามารถทำให้แต่ละ Tuple ใน Relation มีค่าของข้อมูลที่ไม่ซ้ำกันได้ เช่น Relation "EMPLOYEE" สามารถมี Candidate Key ทั้งที่กำหนดขึ้นจาก Attribute "EmpID" ซึ่งเป็นรหัสพนักงาน และที่กำหนดขึ้นจาก Attribute "Name" ซึ่งเป็นชื่อของพนักงาน เนื่องจากทั้ง 2 Attribute ไม่ปรากฏค่าใน Tuple ใดที่ซ้ำกัน

2. Candidate Key สามารถกำหนดขึ้นจาก Attribute เพียง Attribute เดียวหรือมากกว่าขึ้นอยู่กับว่า Attribute ที่นำมากำหนดเป็น Candidate Key นั้น สามารถทำให้แต่ละ Tuple ใน Relation มีค่าของ ข้อมูลที่ไม่ซ้ำกันได้หรือไม่ เช่น กรณีของ Attribute " EmpID" สามารถนำมากำหนดเป็น Candidate Key เพียง Attribute เดียวได้ เนื่องจากพนักงานแต่ละคนจะมีรหัสพนักงานที่ไม่ซ้ำกัน แต่ในกรณีที่ใช้ Attribute " Name" มากำหนดเป็น Candidate Key จะไม่สามารถกระทำได้ ถ้าพนักงานในบริษัทนั้นมีชื่อที่ซ้ำกัน

3. ค่าของ Candidate Key จะต้องไม่สามารถแยกออกเป็น Key ย่อย ๆ ได้อีก หรือกล่าวอีกนัยหนึ่ง Candidate Key จะต้องเป็น Key ที่มีขนาดเล็กที่สุด ที่ทำให้ข้อมูลในแต่ละ Tuple ของ Relation มีค่าของข้อมูลที่ไม่ซ้ำกัน เช่น Key ที่ประกอบขึ้นจาก Attribute "Name" และ "EmpID" จะไม่สามารถนำมาใช้เป็น Candidate Key ได้ เนื่องจากเฉพาะค่าของ Attribute "EmpID" ก็สามารถทำให้แต่ละTuple ใน Relation มีค่าของข้อมูลที่ไม่ซ้ำกันอยู่แล้ว ดังนั้น Candidate Key ที่กำหนดขึ้นจาก Attribute "Name" และ "EmpID" จึงไม่ใช่ Key ที่มีขนาดเล็กที่สุด เป็นต้น

4. การกำหนด Candidate Key ให้กับ Relation จะไม่เกี่ยวข้องกับการกำหนด Index เนื่องจากโครงสร้างของ Index จะเป็น Key ทางกายภาพมากกว่า

3.17 Primary Key และ Alternate Key

ดังที่กล่าวมาแล้วว่า แต่ละ Relation ในแง่ความเป็นจริง สามารถมี Candidate Key ได้หลาย Candidate Key ด้วยกัน ดังนั้น จึงต้องมีการเลือก Candidate Key ที่เหมาะสมมาทำหน้าที่เป็น Primary Key ส่วน Candidate Key ที่เหลือ จะทำหน้าที่เป็น Alternate Key แทน สำหรับ Primary Key แล้วนับเป็น Key ที่สำคัญสำหรับแต่ละ Relation เนื่องจาก Primary Key จะถูกใช้เป็น Key หลัก สำหรับตรวจสอบการซ้ำกันของข้อมูลระหว่างที่ทำการป้อนข้อมูลหรือกำหนดข้อมูลใหม่ให้กับ Relation

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Primary Key เป็น Key ที่ใช้กำหนดให้กับ Base Relation แต่ก็ไม่ได้หมายความว่า ทุก Base Relation จะต้องมี Primary Key เนื่องจากบาง Relation ก็ไม่ต้องการอ้างถึงจาก Relation อื่น

ในการเลือก Candidate Key มาเป็น Primary Key นั้น ถ้า Relation นั้นมี Candidate Key เพียง Key เดียว ก็สามารถนำเอา Candidate Key นั้นมากำหนดเป็น Primary Key ได้ทันที แต่ถ้า Relation นั้นมี Candidate Key มากกว่า 1 Candidate Key แล้ว ให้เลือกเอา Candidate Key ที่มีขนาดเล็กที่สุด และถูกเรียกใช้โดยการกระทำ (Operation) ต่าง ๆ มากที่สุดมาเป็น Primary Key เช่น Relation "EMPLOYEE" ที่มี 2 Candidate Key ซึ่งได้แก่ Candidate Key "EmpID" และ Candidate Key "Name +Surname" ซึ่งเมื่อพิจารณาดู จะพบว่า Candidate Key "EmpID" มีขนาดที่เล็กกว่า Candidate Key "Name + Surname" รวมทั้งเป็น Candidate Key ที่ถูกเรียกใช้โดยการกระทำต่าง ๆ มากกว่า (ส่วนใหญ่ เวลาป้อนข้อมูลในโปรแกรมต่าง ๆ มักจะอ้างถึงรหัสพนักงานเป็นหลัก) ดังนั้นจึงเลือก Candidate Key "EmpID" มาทำหน้าที่เป็น Primary Key เป็นต้น

3.18 Foreign Key

จุดมุ่งหมายหนึ่งของ Relational Model ได้แก่ ความต้องการให้ได้ฐานข้อมูลที่ปลอดจากการมีข้อมูลที่ซ้ำซ้อนกัน ดังนั้น ในแนวความคิดของ Relational Model ข้อมูลจึงถูกแยกเก็บอยู่ใน Relation ต่าง ๆ ที่มีโครงสร้างในระดับที่ย่อยที่สุด แล้วหันมาให้ความสนใจกับความสัมพันธ์ระหว่างข้อมูลในแต่ละ Relation แทน ซึ่งการที่จะทำให้ข้อมูลในแต่ละ Relation คงความสัมพันธ์อยู่ได้นั้น จึงต้องอาศัย Key ที่เรียกว่า Foreign Key

Foreign Key ได้แก่ Attribute ใด Attribute หนึ่งใน Relation ที่ใช้อ้างอิงไปยัง Attribute ที่ทำหน้าที่เป็น Candidate Key ของอีก Relation หนึ่งที่มีความสัมพันธ์กัน เช่น ค่าของข้อมูลใน Attribute "DeptID" ของ Relation "EMPLOYEE" ที่ทำหน้าที่เป็น Foreign Key เพื่ออ้างอิงไปยัง Attribute "DeptID" ที่ทำหน้าที่เป็น Candidate Key ของ Relation "DEPARTMENT"

สำหรับคุณสมบัติของ Foreign Key จะประกอบด้วย

1. Foreign Key สามารถกำหนดขึ้นจาก Attribute เพียง Attribute เดียวหรือมากกว่า เช่นเดียวกับ Primary Key สำหรับ Attribute ที่ทำหน้าที่เป็น Foreign Key มักเขียนอยู่ในเครื่องหมาย "{}" เช่น {DeptID}

2. ค่าของ Foreign Key ใน Relation หนึ่ง จะต้องปรากฏอยู่ใน Candidate Key ของ Relation ที่สัมพันธ์กันนั้น แต่ในแง่กลับกัน ทุกค่าของ Candidate Key ใน Relation หนึ่ง ไม่จำเป็นที่จะต้องปรากฏอยู่ใน Foreign Key ของอีก Relation หนึ่งที่สัมพันธ์กัน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3. จำนวนของ Attribute ที่จะมาประกอบเป็น Foreign Key จะขึ้นอยู่กับจำนวนของAttribute ที่กำหนดเป็น Candidate Key ของ Relation ที่มีความสัมพันธ์กับ Relation นั้น ที่ถูกนำมาใช้เปรียบเทียบกับค่าของ Foreign Key

4. Domain ของ Attribute ที่กำหนดเป็น Foreign Key จะต้องมีค่าเช่นเดียวกับ Domain ของ Attribute ที่เป็นCandidate Key ของ Relation ที่สัมพันธ์กัน

5. Attribute ที่ทำหน้าที่เป็น Foreign Key ไม่จำเป็นที่จะต้อง Candidate Key ของ Relation นั้น เช่น Attribute "DeptID" ของ Relation "EMPLOYEE" ซึ่งไม่ได้เป็น Candidate Key ของ Relation แต่สามารถทำหน้าที่เป็น Foreign Key

6. Relation ที่เป็นเจ้าของ Foreign Key จะถูกเรียกว่า "Referencing Relation" ส่วน Relation ที่เป็นเจ้าของ Candidate Key ที่สัมพันธ์กับ Foreign Key นั้น จะถูกเรียกว่า "Referenced Relation" หรือ "Target Relation"

7. Relation ใด ๆ สามารถเป็นได้ทั้ง Referenced Relation และ Referencing Relation เช่น Relation "DEPARTMENT" ซึ่งทำหน้าที่เป็น "Referencing Relation" ในกรณีที่มีความสัมพันธ์กับ Relation "EMPLOYEE" และในขณะเดียวกัน ก็สามารถทำหน้าที่เป็น Referenced Relation ในกรณีที่มีความสัมพันธ์กับ Relation "DIVISION"

8. Relation ใด ๆ สามารถมีความสัมพันธ์ด้วย Foreign Key กับตัวมันเองได้ เช่น Attribute "MGR_EMP" ของ Relation "EMPLOYEE" สามารถมีความสัมพันธ์กับ Attribute "EmpID" ภายใน Relation เดียวกันในลักษณะ foreign Key ได้ เพื่อแสดงถึงชื่อผู้จัดการของพนักงานแต่ละคน

3.19 Relationship

Relation ในฐานข้อมูลเดียวกัน โดยทั่วไป Tuple ภายในแต่ละ Relation มักจะต้องมีความสัมพันธ์กับ Tuple ของ Relation ใด Relation หนึ่งในฐานข้อมูลเดียวกันนั้นผ่านทาง Foreign Key เสมอ เช่น Relation "EMPLOYEE" และ "Department"

จำนวนของ Tuple ใน Relation หนึ่งที่มีความสัมพันธ์กับ Tuple ในอีก Relation หนึ่ง สามารถนำมาใช้กำหนดประเภทของความสัมพันธ์ที่เกิดขึ้นระหว่างข้อมูลใน Relation ที่มีความสัมพันธ์กันนั้นได้ดังนี้

1. One-to-One Relationship

เป็น Relationship ที่แต่ละ Participant ของ Entity หนึ่ง จะมีความสัมพันธ์กับอีก Participant ของอีก Entity หนึ่งเพียง Participant เดียว เช่น กรณีลูกค้าสามารถมีบัญชีเงินฝากได้เพียงบัญชีเดียว และแต่ละบัญชีเงินฝากจะมี เจ้าของบัญชีได้พียงคนเดียว

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2. One-to-Many Relationship

เป็น Relationship ที่แต่ละ Participant ของ Entity หนึ่งมีความสัมพันธ์กับ Participant ของอีก Entity หนึ่งมากกว่า 1 Participant เช่น กรณีลูกค้าสามารถมีบัญชีเงินฝากได้มากกว่า 1 บัญชี และแต่ละบัญชีเงินฝากจะต้องมีเจ้าของบัญชีเพียงคนเดียว

3. Many-to-Many Relationship

เป็น Relationship ที่ participant มากกว่า 1 Participant ของ Entity หนึ่ง มีความสัมพันธ์กับ Participant ของอีก Entity หนึ่งมากกว่า 1 Participant เช่น กรณีลูกค้าสามารถมีบัญชีเงินฝากได้มากกว่า 1 บัญชี และแต่ละบัญชีเงินฝากสามารถมีเจ้าของบัญชีได้มากกว่า 1 คน

3.20 ภาษาทางด้านฐานข้อมูล (Query Language)

ผลิตภัณฑ์ทางด้านฐานข้อมูลที่มีโครงสร้างข้อมูลในแบบ Relational ซึ่งจำหน่ายอยู่ในท้องตลาด สิ่งที่จำเป็นสิ่งหนึ่งสำหรับผลิตภัณฑ์เหล่านี้ ได้แก่ การมีภาษาทางด้านฐานข้อมูล หรือเรียกว่า Query Language เช่น ภาษา Structured Query Language (SQL) ภาษา Query – by – Example (QBE) และภาษา Quel ฯลฯ เป็นต้น ภาษาเหล่านี้ได้ถูกพัฒนาขึ้นจากแนวคิดที่ต่างกัน เช่น ภาษา QBE ซึ่งถูกพัฒนาขึ้นจากแนวคิดของ Relational Calculus ส่วนภาษา Quel ถูกพัฒนาขึ้นจากแนวคิดของ Tuple Relational Calculus ในขณะที่ภาษา SQL ถูกพัฒนาขึ้นจากแนวคิดของ Relational Calculus และ Relational Algebra เป็นหลัก แต่อย่างไรก็ตาม ภาษาที่ได้รับความนิยมมากที่สุดได้แก่ ภาษา Structured Query Language

Structured Query Language เป็นภาษาทางด้านฐานข้อมูล ที่นิยมใช้มากที่สุดภาษาหนึ่ง โดยมักถูกเรียกย่อ ๆ ว่า "SQL" ในการอ่าน สามารถอ่านได้ทั้ง " S – Q –L" และ "Sequel" แต่โดยทั่วไปจะนิยมอ่านว่า "S-Q-L" มากกว่า SQL เริ่มต้นพัฒนาครั้งแรกโดย San Jose Research Laboratory (ปัจจุบันเปลี่ยนชื่อเป็น Almaden Research Center) ของบริษัท IBM โดยมีชื่อแรกเริ่มว่า "Sequel" ซึ่งเป็นงานวิจัยในโครงการ R ในต้นทศวรรษ 1970 ที่ต่อมาได้เปลี่ยนชื่อมาเป็น "SQL" และได้ถูกนำมาใช้เป็นต้นแบบของภาษา SQL ของผลิตภัณฑ์ทางด้านฐานข้อมูลจำนวนมาก แต่อย่างไรก็ตาม ภาษา SQL ของแต่ละผลิตภัณฑ์ยังคงมีข้อแตกต่างกันในรายละเอียดทางด้านการใช้งาน ดังนั้นในปี ค.ศ. 1986 ทางด้าน American National Standards Institute (ANSI) จึงได้กำหนดมาตรฐานของ SQL ขึ้น รวมทั้งบริษัท IBM ที่ได้กำหนดมาตรฐานของตัวเองขึ้นมาเช่นเดียวกัน โดยมีชื่อว่า Systems Application Architecture Database Interface (SAA-SQL) ซึ่งต่อมา ทั้ง 2 มาตรฐานนี้ได้เป็นมาตรฐานในการผลิตภาษา SQL ของแต่ละบริษัท

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

คำสั่งต่าง ๆ ของภาษา SQL สามารถแบ่งตามลักษณะการใช้งาน ออกได้เป็น 3 กลุ่ม ดังนี้

1. กลุ่มคำสั่ง Data Definition Language (DDL) เป็นกลุ่มคำสั่งที่ใช้สำหรับสร้างฐานข้อมูล หรือใช้กำหนดโครงสร้างให้กับ Relation ภายในฐานข้อมูล เช่น การเพิ่ม เปลี่ยนแปลง ลบ Attribute ของ Relation ฯลฯ เป็นต้น

2. กลุ่มคำสั่ง Data Manipulation Language (DML) เป็นกลุ่มคำสั่งที่พัฒนาขึ้นตามแนวคิดของ Relational Algebra และ Record Relational Calculus โดยประกอบด้วยคำสั่งที่ใช้สำหรับเพิ่ม ลบ หรือเปลี่ยนแปลงข้อมูลในฐานข้อมูล

3. กลุ่มคำสั่ง Data Query Language เป็นกลุ่มคำสั่ง DML ประเภทหนึ่ง ที่ใช้ในการเลือกข้อมูลจาก Relation ขึ้นมาแสดงผลตามรูปแบบที่ต้องการ

3.21 ฐานข้อมูลแบบ Relational กับ Microsoft Access

ใน Microsoft Access คำว่า Relation จะถูกเรียกว่า Table แทน เนื่องจาก Relation ของ Microsoft Access ถึงแม้ว่าจะพัฒนาขึ้นจากแนวความคิดของ Relation ใน Relational Model ก็ตาม แต่เพื่อความสะดวกต่อการใช้งาน ทาง Microsoft Access จึงได้มีการเปลี่ยนแปลง และเพิ่มคุณสมบัติบางประการเข้าไป ส่งผลให้ Relation ของ Microsoft Access มีคุณสมบัติที่ไม่ตรงแนวคิดของRelation ใน Relational Model ดังนั้นคำว่า "Relation" จึงถูกเปลี่ยนเป็น "Table"

เมื่อพิจารณาโดยทั่วไป ทั้ง 2 คำนี้ สามารถนำมาใช้แทนกันได้ แต่ในแง่ความเป็นจริงแล้ว ทั้ง 2 คำกลับมีความหมายที่แตกต่างกัน กล่าวคือ คำว่า Relation เป็นคำที่ใช้กล่าวถึงหน่วยที่ใช้จัดเก็บข้อมูลของงานข้อมูลตามแนวความคิดที่กำหนดไว้ใน Relational Model มากกว่าคำว่าTable หรือกล่าวอีกนัยหนึ่ง คำว่า Relation จะเป็นคำที่ใช้กล่าวถึงหน่วยที่ใช้จัดเก็บข้อมูลในรูปของแนวความคิดมากกว่า Table เนื่องจากผลิตภัณฑ์ทางด้านฐานข้อมูลต่าง ๆ เมื่อนำเอาความคิดของ Relation ไปใช้ จะมีการปรับปรุงบางคุณสมบัติของ Relation ให้มีรูปแบบที่สามารถนำไปใช้จริง และเหมาะสมกับฟังก์ชันการทำงานต่าง ๆ ของตัวผลิตภัณฑ์นั้น จึงส่งผลให้โครงสร้างของ Table มีลักษณะคาบเกี่ยวระหว่างโครงสร้างของแฟ้มข้อมูล และโครงสร้างของ Relation

สำหรับคุณสมบัติของ Table ที่แตกต่างจาก Relation มีดังนี้

1. Table สามารถมีข้อมูลในแถวใดแถวหนึ่งซ้ำกันได้ ซึ่งต่างจาก Relation ที่จะไม่ปรากฏ Tuple ที่มีข้อมูลที่ซ้ำกัน สำหรับข้อยืนยันของคุณสมบัติในข้อนี้ ได้แก่ ผลิตภัณฑ์ทางด้านฐานข้อมูลที่มีโครงสร้างข้อมูลแบบ Relational โดยทั่วไป จะมีการใช้คำสั่ง DISTINCT ในคำสั่ง SQL เพื่อเลือกเฉพาะ Tuple ที่มีข้อมูลที่ไม่ซ้ำกับ Tuple อื่น สำหรับรายละเอียดของ SQL จะกล่าวถึงในลำดับต่อไป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2. ลำดับที่ของแถวใน Table จะถูกกำหนดลำดับที่ตายตัว ดังนั้น ใน Table จึงมีการกำหนด Pointer เพื่อชี้ยังตำแหน่งของแถวปัจจุบัน ส่งผลให้ผู้ใช้สามารถเลื่อน Pointer ไปยังแถวแรก หรือแถวสุดท้าย หรือแถวก่อนหน้าหรือแถวถัดไปใน Table ได้ รวมทั้งทำให้การเรียกชื่อของแต่ละแถวใน Table ถูกเปลี่ยนจากคำว่า Tuple ไปเป็น Record แทน

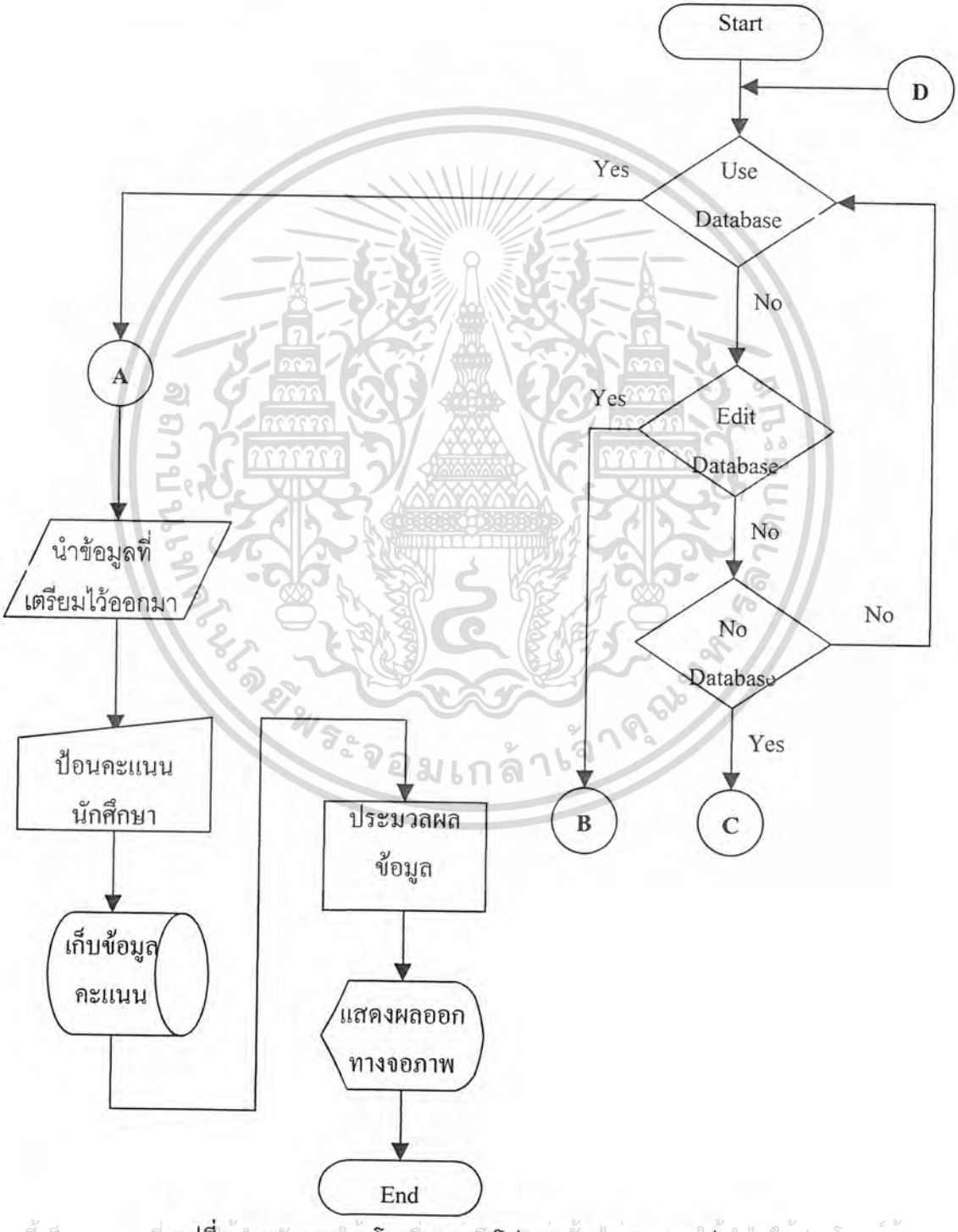
3. ลำดับที่ของสดมภ์ใน Table จะถูกกำหนดลำดับที่ตายตัว เช่นเดียวกับแถว ดังนั้น จึงสามารถอ้างถึงข้อมูลในสดมภ์ต่าง ๆ ด้วยลำดับที่ของสดมภ์นั้นได้ รวมทั้งทำให้การเรียกชื่อของแต่ละสดมภ์ใน Table จะถูกเปลี่ยนจากคำว่า Attribute ไปเป็น Field แทน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 4

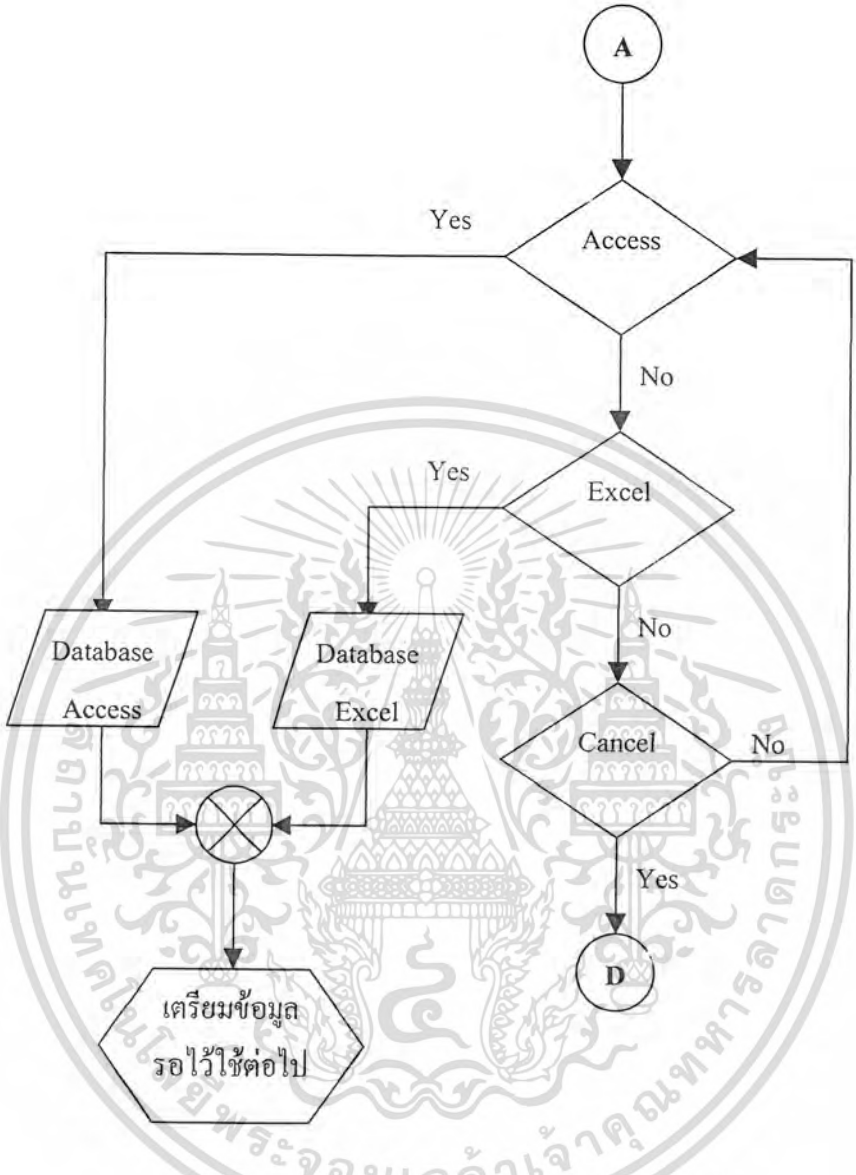
ออกแบบ (Design)

4.1 Flow Chart การทำงานของโปรแกรม Grad Point Analysis Aid Program



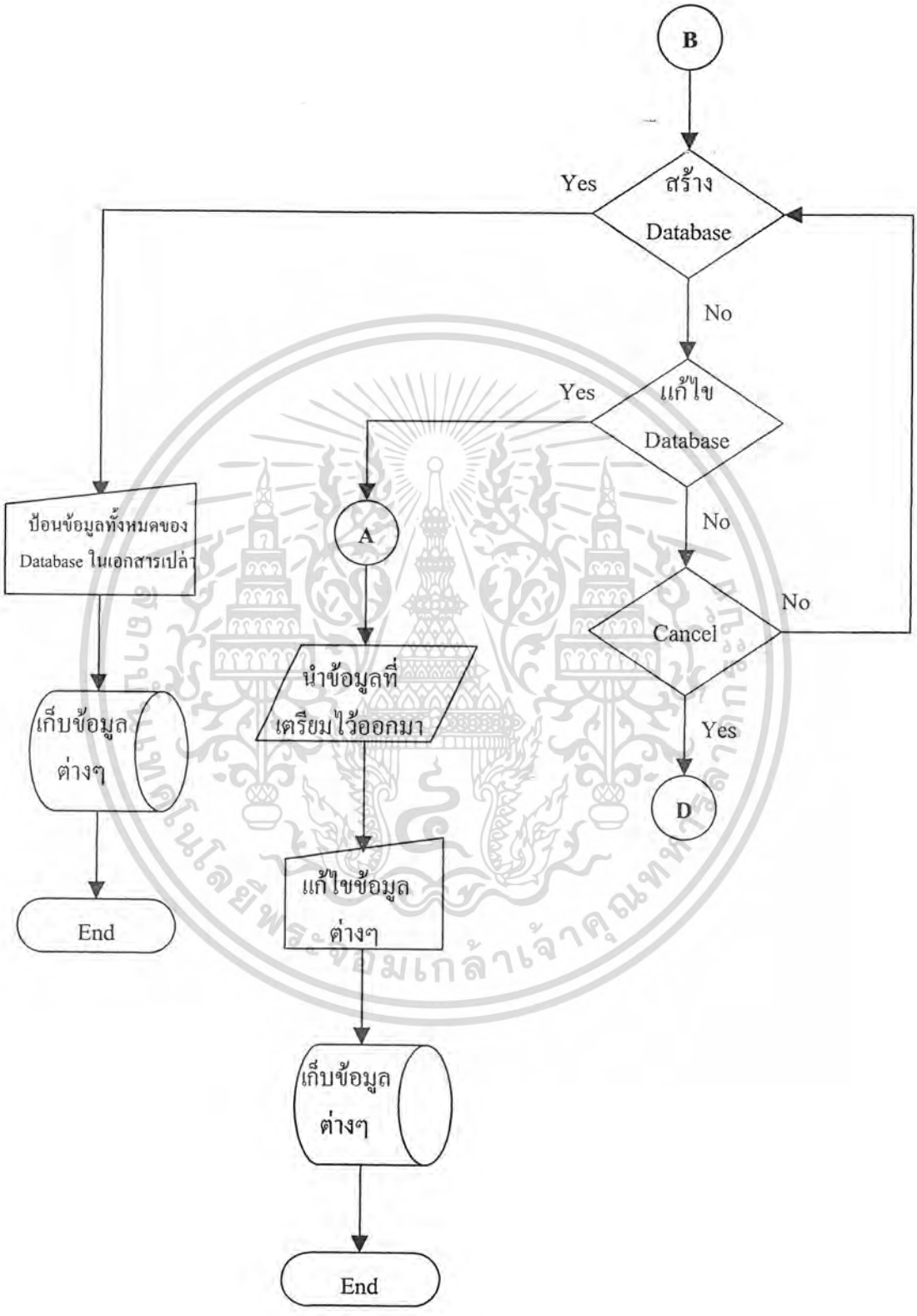
รูปที่ 4.1 Flow Chart โดยรวมของโปรแกรมช่วยวิเคราะห์คะแนน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4.2 Flow Chart โปรแกรมย่อยเรียกใช้ไฟล์ฐานข้อมูล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



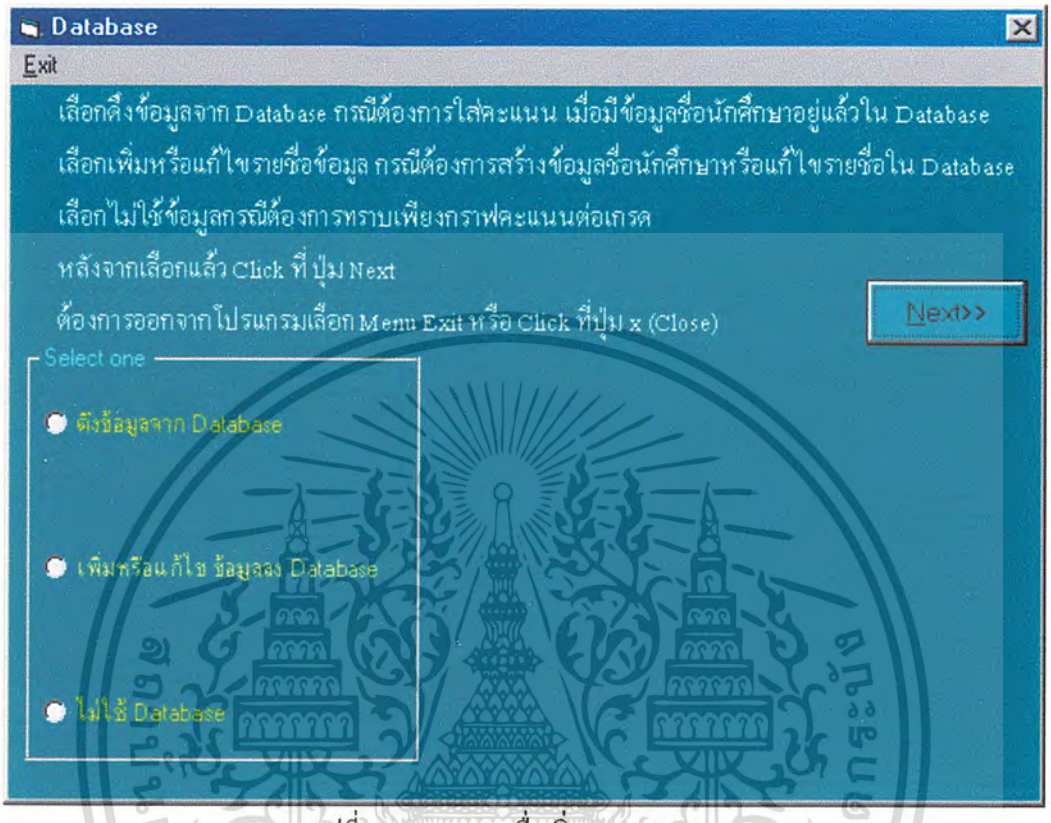
รูปที่ 4.3 โปรแกรมส่วนเรียกไฟล์ฐานข้อมูลมาเพื่อแก้ไข, เพิ่ม, ลบ ค่าของ Record

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รูปที่ 4.4 โปรแกรมย่อยกรณีไม่ใช้ฐานข้อมูล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.2 ลักษณะส่วนติดต่อผู้ใช้ของโปรแกรม (Graphic User Interface)

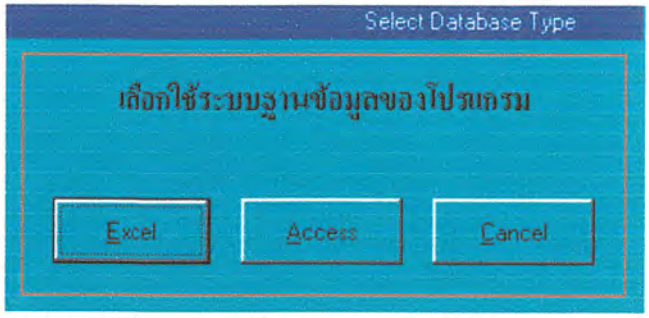


รูปที่ 4.5 Graphic เมื่อเริ่ม Run Program

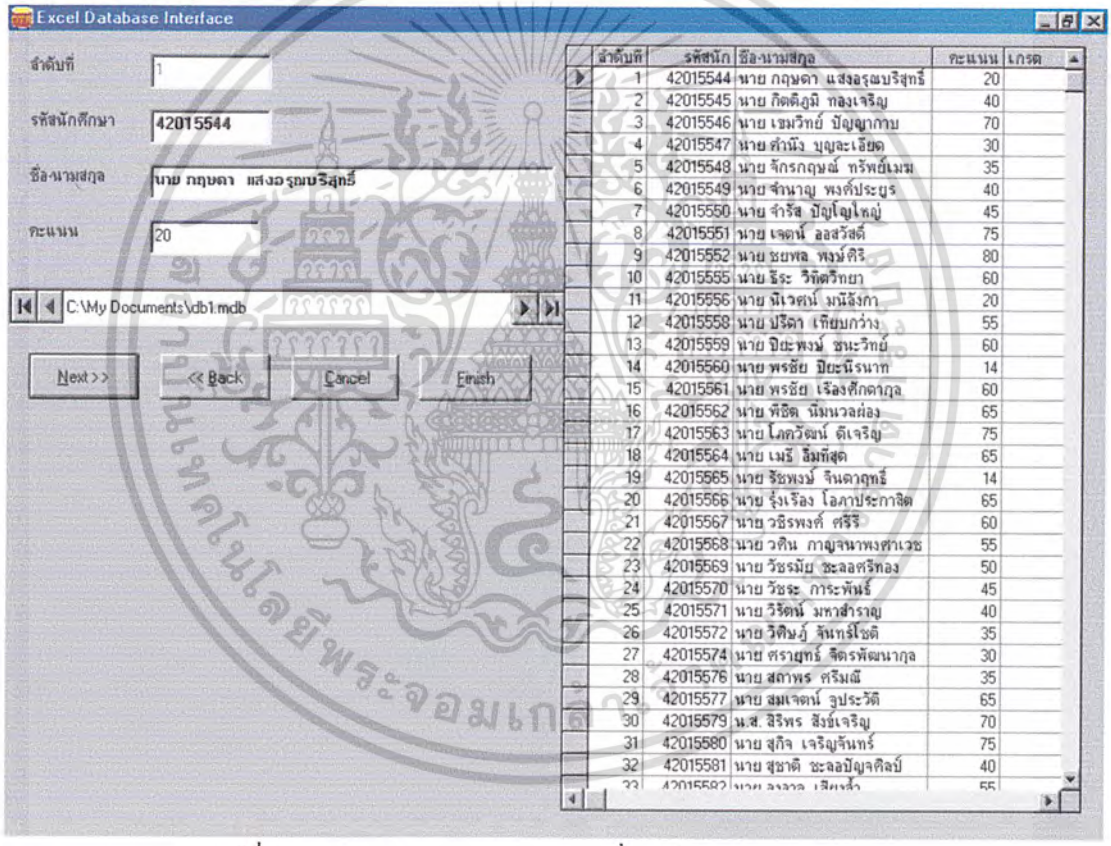
รูปที่ 4.6 Graphic เมื่อมีการเลือกส่วนต่าง ๆ ของหน้าจอหลัก

โดยแต่ละส่วนได้มีโปรแกรมย่อยรองรับการทำงานของแต่ละ Option การใช้งานโดยเลือก Option 1 ตัวเลือกโดยแต่ละตัวเลือกมีคำอธิบายไว้แล้วภายในโปรแกรม หลังจากทำการ Click เลือก Optionที่ต้องการแล้ว Click ปุ่ม Next เพื่อทำการเตรียมสภาวะแวดล้อมเพื่อใช้งานต่อไป ถ้าต้องการออกจากโปรแกรม ใช้ Menu Exit หรือ Click ปุ่มกากบาทขวาบนของ Form

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



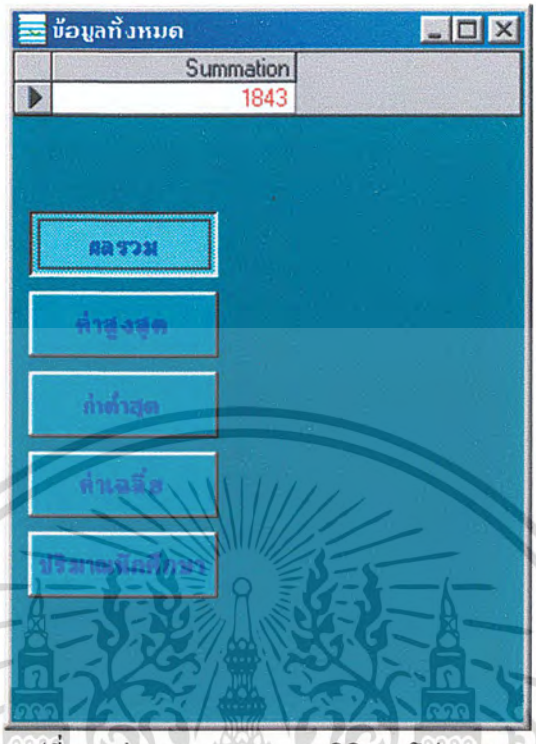
รูปที่ 4.7 Graphic ส่วนเลือกใช้ระบบฐานข้อมูลโปรแกรม (โปรแกรมย่อย A ใน Flow Chart)



รูปที่ 4.8 File db1.mdb ของ Access ที่ถูกเรียกออกมาแสดงผล

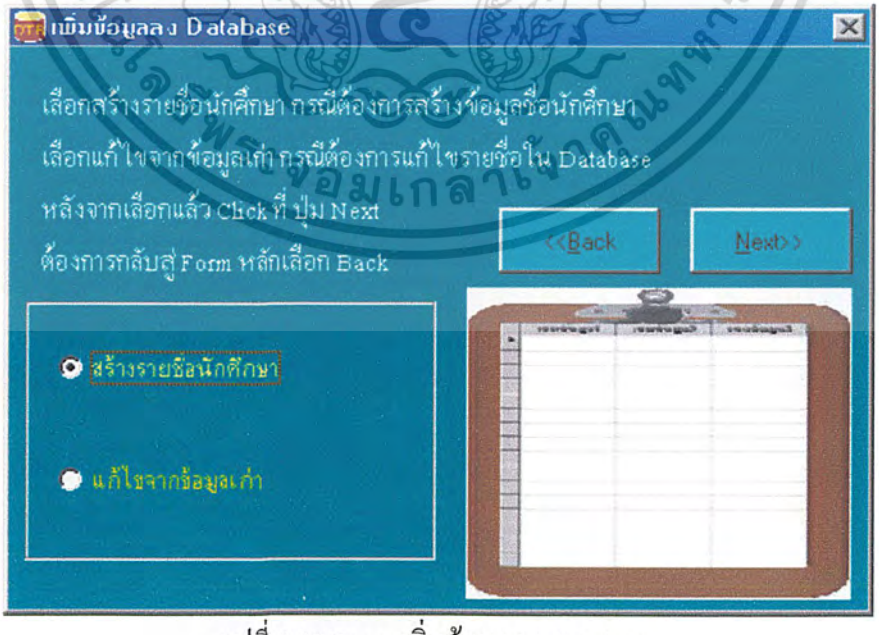
เมื่อเลือกไฟล์จากฐานข้อมูลมาแล้วจะแสดงผลพร้อมให้กรอกคะแนน กรณีเลือกหน้าจอหลักเป็นแบบดึงข้อมูลจาก Database จะไม่สามารถแก้ไขส่วนอื่น เช่น รหัสนักศึกษา, ชื่อนามสกุล ได้ ใช้ได้เฉพาะการใส่หรือแก้ไขคะแนนเท่านั้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



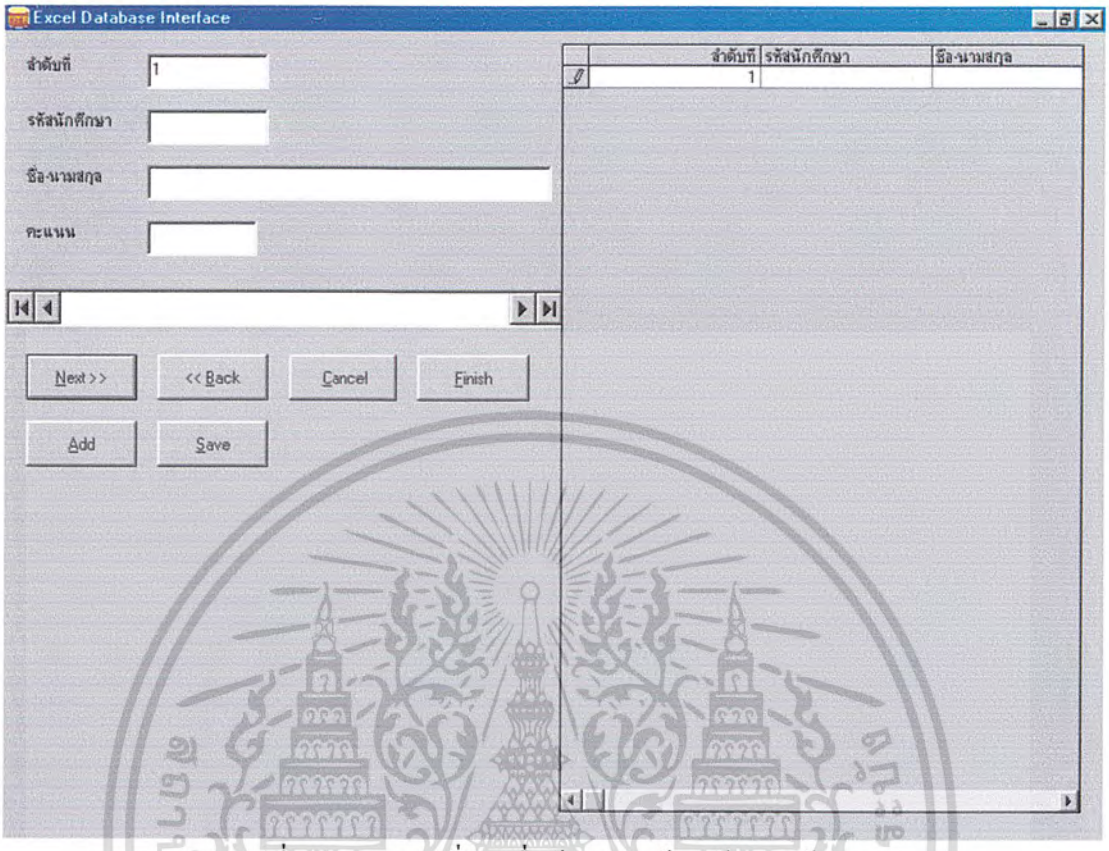
รูปที่ 4.9 ส่วนแสดงผลทางสถิติของโปรแกรม

จากรูปที่ 4.8 เมื่อป้อนคะแนนจนครบแล้ว Click ที่ปุ่ม Finish จะมีส่วนแสดงผลออกมาทางจอภาพดังรูปที่ 4.9

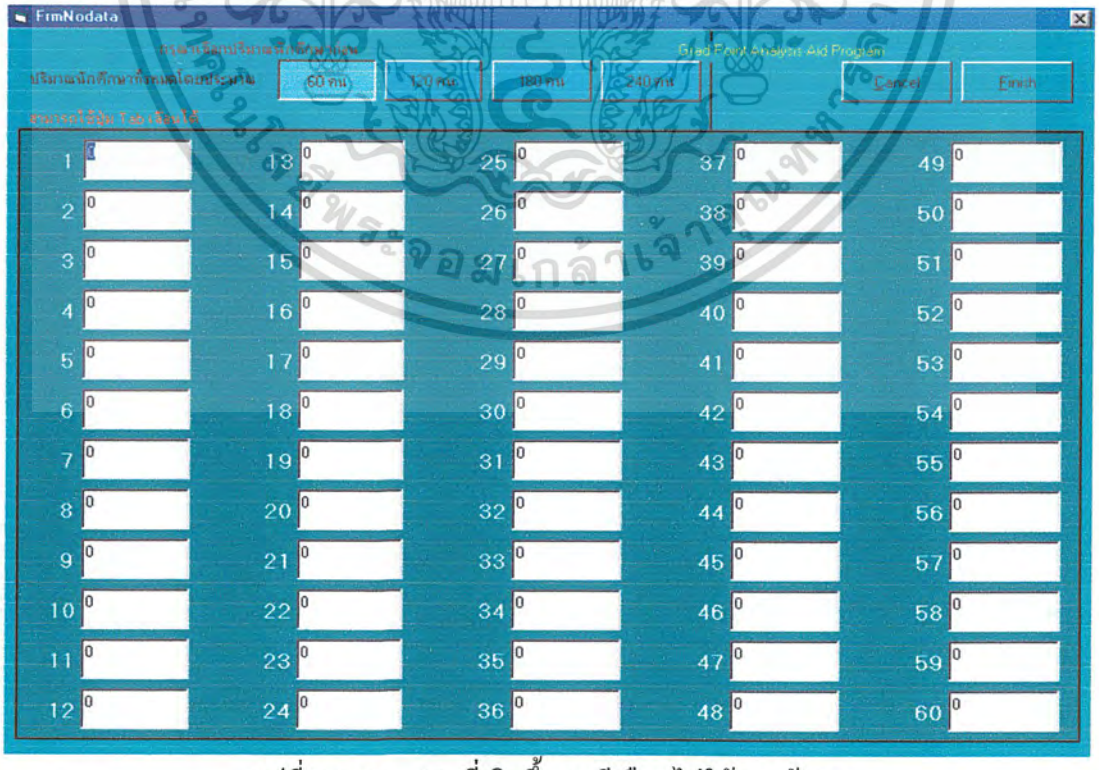


รูปที่ 4.10 Form เพิ่มข้อมูลลง Database

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4.11 Graphic ที่เกิดเมื่อต้องการสร้างแฟ้มฐานข้อมูล



รูปที่ 4.12 Graphic ที่เกิดขึ้นกรณีเลือกไม่ใช้ฐานข้อมูล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากรูปที่ 4.10 มาจากการเลือกเพิ่มข้อมูลลง Database ของ Form หลัก โดยสามารถเลือกได้ 2 แบบคือสร้างรายชื่อนักศึกษา และแก้ไขข้อมูลเก่า โดยที่ถ้าเลือกสร้างรายชื่อนักศึกษาจะปรากฏ Form ตามรูปที่ 4.11 โดยทำการใส่รายชื่อหรือรหัสนักศึกษา หรือค่าต่างๆ แล้ว Add ลง Database แต่ถ้าเลือกแก้ไขข้อมูลเก่า จะปรากฏ รูปที่ 4.7 เพื่อเลือก File ที่จะทำการแก้ไข

4.3 Source Code Program Grad Point Analysis Aid Program

4.3.1 FrmStart

```
Public Datain As Boolean, Dataout As Boolean, Nodata As Boolean
Public Access As Boolean, Excel As Boolean

Private Sub CmdNext_Click()
        Dim Key As Integer
            If OptDataout = True Then
                FrmSDTB.Show            'เลือก Option แล้วเปิด Form ต่อไป
            End If
            If OptDatain = True Then FrmDatain.Show
            If OptNodata = True Then FrmNodata.Show
            If Datain = False And Dataout = False And Nodata = False Then
                MsgBox "กรุณาเลือกตัวเลือกก่อน", vbCritical, "Please select one Option"
            End If
End Sub
Private Sub Form_Load()
                FrmGraph.Show
Label1.Font.Size = 14
Label1.Caption = "เลือกดึงข้อมูลจาก Database กรณีต้องการใส่คะแนน เมื่อมีข้อมูลชื่อนักศึกษา _
อยู่แล้วใน Database" & vbCrLf & "เลือกเพิ่มหรือแก้ไขรายชื่อข้อมูล กรณีต้องการสร้างข้อมูลชื่อ _
นักศึกษาหรือแก้ไขรายชื่อใน Database " & vbCrLf & "เลือกไม่ใช้ข้อมูลกรณีต้องการทราบเพียง _
กราฟคะแนนต่อเกรด" & vbCrLf & "หลังจากเลือกแล้ว Click ที่ ปุ่ม Next" & vbCrLf & "ต้องการ_
ออกจากโปรแกรมเลือก Menu Exit หรือ Click ที่ปุ่ม x (Close)"
End Sub
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
Private Sub Form_Unload(Cancel As Integer)
        Dim KeyPress As Integer
                intkeypress = MsgBox("ต้องการออกจากโปรแกรม ? ", vbOKCancel, "Exit")
        Select Case intkeypress
                Case vbOK
                        End
                Case vbCancel
                        Cancel = True
        End Select
End Sub
Private Sub mnuExit_Click()
                Unload FrmStart
End Sub

Private Sub OptDatain_Click()
                Datain = True
                Dataout = False
                Nodata = False
                ImgDatain.Visible = True
                ImgDataout.Visible = False
                ImgNodata.Visible = False
End Sub
Private Sub OptDataout_Click()
                Datain = False
                Dataout = True
                Nodata = False
                ImgDataout.Visible = True
                ImgDatain.Visible = False
                ImgNodata.Visible = False
End Sub
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
Private Sub OptNodata_Click()
        Datain = False
        Dataout = False
        Nodata = True
        ImgNodata.Visible = True
        ImgDataout.Visible = False
        ImgDatain.Visible = False
End Sub
```

4.3.2 FrmDatabase

```
Sub Excel()
        Data1.Connect = "Excel 8.0;"
        Data1.DatabaseName = FrmSDTB.CmdlFileName
        Data1.RecordSource = "Sheet1$"
        Data1.Caption = FrmSDTB.CmdlFileName
    End Sub
Sub Access()
        Data1.Connect = "Access"
        Data1.DatabaseName = FrmSDTB.CmdlFileName
        Data1.RecordSource = "Table1"
        Data1.Caption = FrmSDTB.CmdlFileName
End Sub
Private Sub CmdAdd_Click()
        Data1.Recordset.AddNew
End Sub

Private Sub CmdCancel_Click()
        FrmStart.Show
        Unload Me
End Sub
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
Private Sub CmdFinish_Click()
        FrmStiti.Show
End Sub

Private Sub CmdSave_Click()
    Dim strFilename As String
    CommonDialog1.Filter = "Database Excel (*.xls) |*.xls| Database Access (*.mdb) |*.mdb"
        CommonDialog1.ShowSave
        'Open strFilename For Output As #1
        'Print #1, strFilename
        Close #1
End Sub

Private Sub Command1_Click()   'Next Buttom
        Data1.Recordset.MoveNext
        If Data1.Recordset.EOF = True Then Data1.Recordset.MoveLast
        If FrmSDTB.Access = True Or FrmSDTB.Excel = True Then
        Text4.Text = ""
        Text4.SetFocus
        End If
End Sub

Private Sub Command2_Click()   'Back Buttom
        Data1.Recordset.MovePrevious
        If Data1.Recordset.BOF = True Then Data1.Recordset.MoveFirst
End Sub
Private Sub Form_Activate()
        Data1.Recordset.MoveLast      'กำหนดค่าให้กับ Property "RecordCount"
        Data1.Recordset.MoveFirst
End Sub
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
Private Sub Form_Load()
        If FrmStart.Dataout = True Then
            If FrmSDTB.Excel = True Then  Excel
            If FrmSDTB.Access = True Then Access
            CmdAdd.Visible = False
            CmdSave.Value = False
        End If
        If FrmStart.Datain = True Then
            If FrmDatain.BlnAdd = True Then Excel
            Data1.DatabaseName = "C:\My Documents\book1.xls"
            Data1.Refresh
            Text2.Locked = False
            Text3.Locked = False
            Text1.Enabled = True
            Text1.Locked = False
        End If
        If FrmDatain.BlnEdit = True Then
            If FrmSDTB.Excel = True Then Excel
            If FrmSDTB.Access = True Then Access
        End If
        CmdAdd.Visible = True
        CmdSave.Visible = True
End If

End Sub
Private Sub Text4_LostFocus()
        If FrmStart.OptDataout = True Then
            If Text4.Text = "" Then MsgBox "กรุณาใส่คะแนน", vbOKOnly, "Warning"
        End If
End Sub
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.3.3 FrmSDTB

```
Public CmdlFileName As String, Access As Boolean, Excel As Boolean
Private Sub CmdAccess_Click()
        Access = True
        Cmdl1.Filter = "Database Access (*.mdb) |*.mdb|"
        Cmdl1.ShowOpen
        CmdlFileName = Cmdl1.FileName
        If CmdlFileName = "" Then FrmSDTB.Show
        frmDatabase.Show
        Unload FrmSDTB
End If
End Sub

Private Sub CmdExcel_Click()
        Excel = True
        Cmdl1.Filter = "Database Excel (*.xls) |*.xls|"
        Cmdl1.ShowOpen
        CmdlFileName = Cmdl1.FileName
        If CmdlFileName = "" Then
        FrmSDTB.Show
        frmDatabase.Show
        Unload FrmSDTB
        End If
End Sub
Private Sub Command1_Click()
        FrmStart.Show
End Sub
Private Sub Form_Load()
        Label2.Left = FrmSDTB.Width - 1700
End Sub
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
Private Sub Timer1_Timer()
        Timer2.Enabled = True
        Shape1.Visible = True
End Sub
Private Sub Timer2_Timer()
        Shape1.Visible = False
        Timer2.Enabled = False
End Sub
Private Sub Timer3_Timer()
        Label2.Left = Label2.Left - 20
        If Label2.Left < -1500 Then
            Label2.Left = FrmSDTB.Width
        End If
End Sub
```

4.3.4 FrmStiti

```
Dim SQLcmd As String
Private Sub Form_Deactivate()
        Unload Me
End Sub
Private Sub Form_Load()
        Data1.DatabaseName = FrmSDTB.CmdlFileName
End Sub

Private Sub OpMax_Click()
        SQLcmd = "SELECT MAX(คะแนน) AS Maximum FROM Table1"
        Data1.RecordSource = SQLcmd
        Data1.Refresh
End Sub
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
Private Sub OptAvg_Click()
        SQLcmd = "SELECT AVG(คะแนน) AS Average FROM Table1"
        Data1.RecordSource = SQLcmd
        Data1.Refresh
End Sub

Private Sub OptCount_Click()
        SQLcmd = "SELECT Count(คะแนน) AS Count FROM Table1"
        Data1.RecordSource = SQLcmd
        Data1.Refresh
End Sub

Private Sub OptMin_Click()
        SQLcmd = "SELECT MIN(คะแนน) AS Minimum FROM Table1"
        Data1.RecordSource = SQLcmd
        Data1.Refresh
End Sub

Private Sub OptSum_Click()
        SQLcmd = "SELECT SUM(คะแนน) AS Summation FROM Table1"
        Data1.RecordSource = SQLcmd
        Data1.Refresh
End Sub
```

4.3.5 FrmDatain

```
Public BlnAdd As Boolean, BlnEdit As Boolean
Private Sub CmdBack_Click()
        FrmStart.Show
        FrmDatain.Hide
End Sub
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
Private Sub CmdNext_Click()
        If BlnAdd = False And BlnEdit = False Then
            MsgBox "กรุณาเลือกตัวเลือกก่อน", vbCritical, "Please select one"
        Else
        FrmDatain.Hide
        End If
        If BlnAdd = True Then frmDatabase.Show
        If BlnEdit = True Then FrmSDTB.Show
End Sub

Private Sub Form_Load()
Label1.Caption = "เลือกสร้างรายชื่อนักศึกษา กรณีต้องการสร้างข้อมูลชื่อนักศึกษา" & vbCrLf & _
"เลือกแก้ไขจากข้อมูลเก่า กรณีต้องการแก้ไขรายชื่อใน Database " & vbCrLf & _
"หลังจากเลือกแล้ว Click ที่ ปุ่ม Next" & vbCrLf & "ต้องการกลับสู่ Form หลักเลือก Back"
End Sub

Private Sub OptAdd_Click()
        BlnAdd = True
        Image1.Visible = True
        Image2.Visible = False
End Sub

Private Sub OptEdit_Click()
        BlnEdit = True
        Image2.Visible = True
        Image1.Visible = False
End Sub
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 5
สรุปและวิจารณ์

5.1 ข้อดีของโปรแกรม

Grad Point Analysis Aid Program นี้สามารถวิเคราะห์คะแนนของนักศึกษาได้ทั้งแบบที่มีหรือ ไม่มี Database ตัวโปรแกรมมีความแม่นยำเชื่อถือได้ สะดวกรวดเร็ว ใช้งานง่ายมีลักษณะการทำงานเป็น Step ไม่ซับซ้อนโดยออกแบบเพื่อผู้ใช้งาน (User) ทั่วไปเป็นหลัก

มีลักษณะการติดต่อกับผู้ใช้แบบ GUI (Graphic User Interface) ส่วน Database ที่ใช้ได้เป็นของ Program Microsoft Access และ Program Microsoft Excel ซึ่งเป็นโปรแกรมที่มีผู้ใช้งานกันอย่างแพร่หลาย

5.2 ข้อเสียของโปรแกรม

Grad Point Analysis Aid Program นี้ยังไม่สามารถที่จะติดต่อกับฐานข้อมูลได้ทุกชนิด ดังที่ได้กล่าวไปแล้วคือสามารถติดต่อได้เพียง Program Microsoft Access และ Program Microsoft Excel เท่านั้นซึ่งอาจยังคงไม่เพียงพอสำหรับผู้ใช้ Database ตัวอื่น เช่น DBASE, FoxPro, Lotus, Paradox ฯลฯ

5.3 สิ่งที่ต้องปรับปรุงเพิ่มเติม

โปรแกรมนี้ถ้าต้องการให้มีประสิทธิภาพ 100 % ควรที่จะมีการวิเคราะห์คำนวณหาค่าที่เหมาะสมในการตัดเกรดนักศึกษา มาเพื่อทำการเขียนลงในโปรแกรมเพื่อการแสดงกราฟ แต่เนื่องจากยังไม่มีการคำนวณค่าที่แน่นอนของโปรแกรม ตัว Project นี้จึงยังไม่มีการแสดงกราฟแต่อย่างใด หากมีสูตรหรือข้อมูลที่มีการวิจัยเรื่องนี้โดยตรงการเขียนโปรแกรมเพื่อแสดงกราฟก็ไม่ใช่เรื่องยากแต่อย่างใด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้