สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

ระบบควบคุมการใช้งานห้องปฏิบัติการไมโครคอมพิวเตอร์เพื่อการบริหาร

The Microcomputer Laboratory Controlling System For Managing Purpose

T046144

โดย

นายบรรพต เทียนรุ่งศรี

นายสามารถ สงวนทรัพย์

อาจารย์ที่ปรึกษา

รศ. สมศักดิ์ มิตะถา

อ. อัครเดช วัชระภูพงษ์

เลขหมู่.....
เลขทะเบียน 46144
วัน, เดือน, ปี 20 มี.ค. 2546

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

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2544

b11287834

ปริญญานิพนธ์ปีการศึกษา 2544

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ระบบควบคุมการใช้งานห้องปฏิบัติการไมโครคอมพิวเตอร์เพื่อการบริหาร

The Microcomputer Laboratory Controlling System For Managing Purpos

ผู้จัดทำ

1. นายบรรพต เทียนรุ่งศรี 42015308
2. นายสามารถ สงวนทรัพย์ 42015324

_____ อาจารย์ที่ปรึกษา

(รศ. สมศักดิ์ มิตะถา)

_____ อาจารย์ที่ปรึกษา

(อ. อัครเดช วัชระภูพงษ์)

ระบบควบคุมการใช้งานห้องปฏิบัติการไมโครคอมพิวเตอร์เพื่อการบริหาร

นายบรรพต เทียนรุ่งศรี 42015308
นายสามารถ สงวนทรัพย์ 42015324
รศ. สมศักดิ์ มิตะถา อาจารย์ที่ปรึกษา
อ. อัครเดช วัชระภูพงษ์ อาจารย์ที่ปรึกษา
ปีการศึกษา 2544

บทคัดย่อ

โครงงานระบบควบคุมการใช้งานห้องปฏิบัติการไมโครคอมพิวเตอร์เพื่อการบริหาร ได้จัดทำขึ้นเพื่อการบริหารการใช้งานห้องปฏิบัติการไมโครคอมพิวเตอร์โดยจะอนุญาติให้ผู้มีสิทธิ์ในการใช้ห้องเท่านั้นที่สามารถเข้าใช้ห้องได้และผู้ที่มีสิทธิ์เข้าใช้ห้องเท่านั้นที่สามารถล็อกอินสู่ระบบได้ และจะเก็บข้อมูลการใช้งานของผู้ใช้เพื่อเอาไปวิเคราะห์เพื่อประโยชน์ในการบริหาร ระบบจะแบ่งออกเป็น 2 ส่วน คือส่วนที่เป็นฮาร์ดแวร์ทำหน้าที่ในการรับข้อมูลจากผู้ใช้ที่ต้องการเข้าใช้ห้องและควบคุมการเปิดปิดประตู และส่วนที่เป็นซอฟต์แวร์ทำหน้าที่ในการตรวจสอบว่าผู้ที่ขอเข้าใช้ห้องมีสิทธิ์ในการเข้าใช้ห้องหรือไม่ และควบคุมดูแลการล็อกอินของผู้ใช้ตลอดจนเก็บข้อมูลการใช้งานเพื่อนำมาวิเคราะห์เพื่อประโยชน์ในการบริหาร โดยฮาร์ดแวร์ของระบบที่จัดทำขึ้นสามารถรองรับได้ 64 ประตูโดยจะมีอุปกรณ์ที่ทำหน้าที่ควบคุมการรับและส่งข้อมูลอยู่ 2 ชุดเพื่อป้องกันข้อมูลสูญหายในกรณีที่ผู้ใช้รูดบัตรพร้อมกัน โดยชุดหนึ่งจะติดตั้งในแต่ละชั้นเพื่อควบคุมการรับและส่งข้อมูลในระดับชั้น อีกชุดหนึ่งจะติดตั้งอยู่ที่เซิร์ฟเวอร์เพื่อรับและส่งข้อมูลของทั้งตึกที่ส่งมาจากแต่ละชั้น

การทดสอบการทำงานแบ่งออกเป็นสองส่วน คือ ทางด้านฮาร์ดแวร์และทางด้านซอฟต์แวร์โดยทางด้านฮาร์ดแวร์ทำการทดสอบการทำงานของอุปกรณ์รับข้อมูลว่าสามารถรับข้อมูลได้ถูกต้องตามที่ต้องการหรือไม่ ทั้งตัวที่อยู่ตามชั้นและส่วนที่อยู่ที่เซิร์ฟเวอร์ ส่วนทางด้านซอฟต์แวร์ก็จะทำการตรวจสอบว่าหลังจากผู้ใช้รูดบัตรเพื่อขอเข้าใช้งานห้องปฏิบัติการระบบสามารถตรวจสอบว่าผู้ใช้มีสิทธิ์เข้าใช้ห้องได้หรือไม่ สามารถอีนาเบิ้ลหรือดิสเอเบิ้ลสิทธิ์ของผู้ใช้ในการล็อกอินเข้าสู่ระบบได้หรือไม่ ตรวจสอบว่าสามารถเก็บข้อมูลของผู้ใช้งานได้อย่างถูกต้องหรือไม่ ตรวจสอบว่าระบบสามารถนำข้อมูลที่เก็บเข้ามาไปวิเคราะห์และรายงานให้กับผู้บริหารได้อย่างถูกต้องหรือไม่

ผลที่ได้จากโครงงานนี้คือจะทำให้เราได้ทราบถึงวิธีการใช้งานของไมโครคอนโทรเลอร์ การเขียนโปรแกรมเพื่อติดต่อกับฐานข้อมูลและการดึงข้อมูลจากฐานข้อมูลมาทำการวิเคราะห์ และการเขียนโปรแกรมในการจัดการสิทธิ์ของผู้ใช้

The Microcomputer Laboratory Controlling System For Managing Purpose

	Banphot	Tienrungsri	
	Samart	Sangunsup	
Assoc.Prof.	Somsak	Mittatha	Advisor
	Akkradach	Watcharapupong	Advisor

ABSTRACT

This project presents The Microcomputer Laboratory Controlling System for Managing Purpose. The main goal of this project is to manage the microcomputer laboratory by allowing permitted users to access the laboratory, login to the system and collecting user's information for analyzing. The proposed system consists of 2 parts, hardware part and software part. The hardware part, implemented based on microcontroller MCS51, has the functions to collect user's information and controlling the doors, this hardware can support up to 64 doors simultaneously. While the software part of this system, developed with Visual Basic, receives the data from the hardware part and check whether it is authorized user or not. If that user has the right, the software part will control the hardware part to open the door. This system has 2 sets of data transmission's controller, used for prevent data losing when more than one user use ID card at the same time, one installed at each floor and the other set up at server room.

By working on this project, we received the knowledge of working with microcontroller, database programming, database accessing & analyzing and user's management programming.

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้คงไม่อาจเสร็จไปได้ด้วยดี หากไม่ได้รับความช่วยเหลือและร่วมมือจากหลายๆฝ่ายด้วยกัน บุคคลแรกที่ต้องกล่าวถึงเพราะเป็นส่วนสำคัญที่ทำให้วิทยานิพนธ์ฉบับนี้เสร็จลงได้ด้วยดีก็คือ อ. สมศักดิ์ มิตะถา และอีกท่านก็คือ อ. อัครเดช วัชระภูพงษ์ อาจารย์ที่ปรึกษาที่ให้ความเอาใจใส่แนะนำ และให้ความช่วยเหลือเสมอมา ซึ่งต้องขอขอบพระคุณเป็นอย่างมาก และ พี่ต๋อง พี่นเรศ พี่อ๊อด พี่เดอ พี่วุฒิ พี่จุ๋ม และพี่ๆทุกคนที่อยู่ในห้อง ปฏิบัติการฮาร์ดแวร์ ที่ให้ความช่วยเหลือในทุกด้านตั้งแต่ทำงานและความเป็นอยู่ในทุกๆด้าน ขอบคุณ มาม่า เวลาหิว ขอบคุณเพื่อน 3P น้องๆห้องฮาร์ดแวร์ ที่ชว:และเล่นเกมส์ยั่วเวลาทำโปรเจ็กต์ ขอบคุณ น้องส้มสำหรับ ปริ้นเตอร์ ขอบคุณ เพื่อนแด็ก สำหรับสแกนและที่ขาดไม่ได้ขอบคุณห้องปฏิบัติการฮาร์ดแวร์ที่ให้สถานที่ทำโปรเจ็ค

และที่ต้องขอขอบพระคุณบุคคลสำคัญที่สุดที่ทำให้ข้าพเจ้ามีวันนี้ ก็คือ บิดา มารดา อันเป็นที่เคารพรักยิ่งซึ่งได้อบรมเลี้ยงดูข้าพเจ้ามาเป็นอย่างดีพร้อมทั้งให้โอกาสทางการศึกษาอย่างเต็มที่และยังให้กำลังใจเสมอมาในทุกๆด้านอย่างหาที่เปรียบมิได้ข้าพเจ้าจะขอระลึกถึงพระคุณอย่างสุดประมาณและขอกราบขอบพระคุณมา ณ ที่นี้ด้วย

บรรพต เทียนรุ่งศรี

สามารถ สงวนทรัพย์

สารบัญ

	หน้าที่
บทคัดย่อภาษาไทย	I
บทคัดย่อภาษาอังกฤษ	II
กิตติกรรมประกาศ	III
สารบัญ	IV
สารบัญตาราง	XI
สารบัญภาพ	IX
บทที่ 1 บทนำ	1
1.1 ความสำคัญและที่มา	1
1.2 วัตถุประสงค์	1
1.3 ขอบเขตของโครงงาน	1
1.4 วิธีการดำเนินโครงงาน	2
บทที่ 2 ทฤษฎีพื้นฐาน	3
2.1 โครงสร้างของ MCS-51	3
2.1.1 คุณสมบัติของ ไมโครคอนโทรลเลอร์ MCS-51	3
2.1.2 โครงสร้างของไมโครคอนโทรลเลอร์ ตระกูล MCS – 51	3
2.1.3 โครงสร้างภายในของ 8051	5
2.1.4 Port ต่างๆ ภายใน 8051	6
2.1.5 ผังเวลาของ CPU (CPU Timing)	9
2.1.6 การแบ่งประเภทของหน่วยความจำ	11
2.1.7 พื้นที่หน่วยความจำที่เข้าถึงข้อมูลทางอ้อมเท่านั้น	12
2.1.8 พื้นที่หน่วยความจำที่เข้าถึงข้อมูลโดยตรงและทางอ้อม	12
2.2 พอร์ตสื่อสารข้อมูลแบบอนุกรมใน MCS-51	13
2.2.1 อัตราเร็วในการรับส่งข้อมูล	14
2.2.2 อธิบายการทำงานพอร์ตสื่อสารข้อมูลแบบอนุกรม (โหมด 1 เพิ่มเติม)	14
2.3 แอ็คทีฟ ไดเร็คทรอรี่เซอร์วิส อินเตอร์เฟส (Active Directory Service Interfaces)	15
2.4 อีนาเบิ้ลติ้ง เอ็นที ยูสเซอร์ (Enabling NT users)	15
2.5 แอ็คเคาน์ดิสเอเบิ้ล (Account Disable)	15
2.6 อีเวนท์ ล็อกกิ้ง (Event Logging)	15
2.6.อะเบาท์อีเวนท์ ล็อกกิ้ง (About Event Logging)	16
2.6.2 อีเวนท์ไทป์ (Event Type)	16
2.6.3 อีเวนท์ล็อกเร็คคอร์ด (EVENTLOGRECORD)	17

สารบัญ (ต่อ)

	หน้าที่
บทที่ 3 พื้นฐานการเขียนโปรแกรมด้านฐานข้อมูล	19
3.1 รูปแบบการติดต่อกับฐานข้อมูลด้วย Visual Basic	20
3.1.1 ทำความรู้จักกับ Relational Database	22
3.1.2 Normalization คืออะไร	23
3.1.3 แนวทางในการปรับปรุงให้ตารางอยู่ในระดับ 3NF (BCNF)	24
3.2 การสร้างฐานข้อมูลใน Access 2000	28
3.2.1 ชนิดของข้อมูลในฐานข้อมูล Access 2000	30
3.3.2 แนวความคิดในการกำหนด Primary Key	32
3.3 การพัฒนาแอพพลิเคชั่น ด้านฐานข้อมูลภายใต้สภาพแวดล้อม Data Project	33
3.3.1 รู้จักกับ VBIDE ชนิด Data Project	34
3.3.2 การตรวจสอบเวอร์ชันของ ADO ที่ใช้งาน	35
3.3.3 เทคนิคการแสดงผลภาษาไทยใน Editor ของ VB	36
บทที่ 4 การใช้งานภาษา SQL	38
4.1 โครงสร้างของภาษา SQL	38
4.1.1 ตัวดำเนินการ (Operator)	39
4.1.2 กลุ่มฟังก์ชั่น Aggregate	39
4.2 ลักษณะการใช้งานของกลุ่มคำสั่ง DML	40
4.2.1 คำสั่ง DELETE	40
4.2.2 คำสั่ง INSERT	41
4.2.3 คำสั่ง SELECT	44
4.2.4 การใช้งานคำสั่ง SELECT แบบมีเงื่อนไข	45
4.2.5 การใช้งานคำสั่ง SELECT ร่วมกับคำสั่ง LIKE	52
4.2.6 คำสั่ง อัพเดท	53
4.3 การเชื่อมโยงตาราง (Joins Tables)	54
4.3.1 การเชื่อมโยงตารางโดยใช้คำสั่ง SELECT	55
4.3.2 การเชื่อมโยงตารางโดยใช้คำสั่ง INNER JOIN	56
4.3.3 การเชื่อมโยงตารางโดยใช้คำสั่ง LEFT JOIN	57
4.3.4 การเชื่อมโยงตารางโดยใช้คำสั่ง RIGHT JOIN	58
4.4 ตัวอย่างการใช้งานภาษา SQL : โปรเจ็กต์ 4-1	59
4.4.1 ตัวอย่างการใช้งานภาษา SQL : โปรเจ็กต์ 4-2	64
4.4.2 อธิบายการทำงานของโปรเจกต์ที่ 4-2	68

สารบัญ (ต่อ)

	หน้าที่
บทที่ 5 การออกแบบ	73
บทที่ 6 DataDic ของระบบ	81
6.1 โพรเซส 1.0 รับข้อมูลและควบคุมประตู	81
6.2 โพรเซส 2.0 เซิร์ฟเวอร์	81
6.3 โพรเซส 1.1 ตรวจสอบตำแหน่งประตู	82
6.4 โพรเซส 1.2 ตรวจสอบตำแหน่งชั้น	82
6.5 โพรเซส 1.1.1 ข้อมูลประตูที่ 1-8 ชั้นที่ 1	83
6.6 โพรเซส 1.1.2 สแกนตรวจสอบข้อมูลประตูชั้นที่ 1	83
6.7 โพรเซส 1.1.3 ใส่รหัสประตูที่ 1-8 ชั้นที่1	83
6.8 โพรเซส 1.1.4 ส่งข้อมูลชั้นที่ 1	83
6.9 โพรเซส 1.1.5 สั่งเปิดประตูชั้นที่ 1	84
6.10 โพรเซส 1.2.1 บับเฟอร์ ข้อมูลชั้นที่ 1	84
6.11โพรเซส 1.2.2 ตรวจสอบชั้นที่เปิดประตู	84
6.12 โพรเซส 1.2.3 ใส่รหัสชั้นที่1-8	84
6.13 โพรเซส 1.2.4 สแกนตรวจสอบข้อมูลในชั้น	85
6.14 โพรเซส 1.2.5 ส่งข้อมูลผู้ใช้	85
6.15 โพรเซส 2.1 ควบคุมการเข้าใช้ห้อง	85
6.16 โพรเซส 2.2 ควบคุมการล็อกอิน	86
6.17 โพรเซส 2.3 รายงานผล	86
6.18 โพรเซส 2.1.1 ตรวจสอบสิทธิ์เข้าใช้ห้อง	87
6.19 โพรเซส 2.1.2 สั่งเปิดประตู	87
6.20 โพรเซส 2.1.3 อัพเดท ข้อมูลการเข้าใช้ห้อง	87
6.21 โพรเซส 2.1.4 อัพเดท ข้อมูลผู้มีสิทธิ์เข้าใช้ห้องปฏิบัติการ	87
6.22 โพรเซส 2.2.1 ตรวจสอบการล็อกอิน ของ User	88
6.23 โพรเซส 2.2.2 เก็บข้อมูลการล็อกอิน ลงฐานข้อมูล	88
6.24 โพรเซส 2.2.3 อัพเดท ฐานข้อมูลผู้ที่ได้รับอนุญาตให้ล็อกอิน	88
6.25 โพรเซส 2.2.4 อีนาเบิ้ลแอ็คเคาน์	88
6.26 โพรเซส 2.2.5 ดิสเอเบิ้ลแอ็คเคาน์	89
6.27 โพรเซส 2.2.6 Delete ข้อมูลผู้มีสิทธิ์ล็อกอิน	89
6.28 ดาต้าโฟว์ รหัสนักศึกษา	89
6.29 ดาต้าโฟว์ ข้อมูลร้องขอเข้าใช้ห้อง	90

สารบัญ (ต่อ)

	หน้าที่
6.30 คาต้าโฟว์ ผลการตรวจสอบสิทธิ์การเข้าใช้ห้อง	90
6.31 คาต้าโฟว์ ข้อมูลการเปิดประตู	90
6.32 คาต้าโฟว์ คำสั่ง อัพเดท ข้อมูลผู้มีสิทธ์เข้าใช้ห้องปฏิบัติการ	91
6.33 คาต้าโฟว์ ข้อมูล อัพเดท ผู้มีสิทธิ์เข้าใช้ห้อง	91
6.34 คาต้าโฟว์ ข้อมูลผู้มีสิทธิ์เข้าใช้ห้อง	91
6.35 คาต้าโฟว์ ล็อกไฟล์เร็คคอร์ด	92
6.36 คาต้าโฟว์ ข้อมูล อัพเดท การล็อกอิน	92
6.37 คาต้าโฟว์ ข้อมูลการล็อกอิน	93
6.38 คาต้าโฟว์ ข้อมูล อัพเดท ผู้ที่ได้รับอนุญาตให้ล็อกอิน	93
6.39 คาต้าโฟว์ ข้อมูลผู้ที่ได้รับอนุญาตให้ล็อกอิน	93
6.40 คาต้าโฟว์ คำสั่ง อีนาเบิ้ลแอ็คเคาน์	94
6.41 คาต้าโฟว์ คำสั่ง ดิสเอเบิ้ลแอ็คเคาน์	94
6.42 คาต้าโฟว์ คำร้องขอข้อมูลการเข้าใช้ห้อง	94
6.43 คาต้าโฟว์ รายงานข้อมูลการใช้ห้อง	95
6.44 คาต้าโฟว์ สถานะข้อมูลประตู	95
6.45 คาต้าโฟว์ คำร้องขอข้อมูล	95
6.46 คาต้าโฟว์ ข้อมูลประตู	96
6.47 คาต้าโฟว์ ข้อมูล ผู้ใช้ ภายในชั้น	96
6.48 คาต้าโฟว์ ข้อมูลตำแหน่งกลอน	96
6.49 คาต้าโฟว์ คำสั่งเปิดประตู	96
6.50 คาต้าโฟว์ สถานะข้อมูลภายในชั้น	97
6.51 คาต้าโฟว์ ข้อมูลผู้ใช้	97
6.52 คาต้าโฟว์ ข้อมูลกรเปิดประตู	97
6.53 คาต้าโฟว์ ล็อกอิน	98
6.54 คาต้าโฟว์ ฐานข้อมูลการเข้าใช้ห้อง	98
6.55 คาต้าโฟว์ ฐานข้อมูลผู้ที่ได้รับอนุญาตให้ล็อกอิน	98
6.56 คาต้าโฟว์ ฐานข้อมูลผู้มีสิทธิ์เข้าใช้ห้องปฏิบัติการ	98
6.57 คาต้าโฟว์ ฐานข้อมูลการล็อกอิน	99

สารบัญ (ต่อ)

	หน้าที่
บทที่ 7 การทดสอบ	100
7.1 วัตถุประสงค์	100
7.2 การทดสอบอุปกรณ์รับข้อมูล	100
7.3 การทดสอบทางด้านซอฟต์แวร์	100
7.4 ขั้นตอนการทดสอบ	100
7.5 ผลการทดสอบ	101
7.6 แนวทางการพัฒนาต่อ	101
7.6.1 ข้อจำกัดของต้นแบบ	102
7.6.2 การพัฒนาทางด้านฮาร์ดแวร์	102
7.6.3 การพัฒนาทางด้านซอฟต์แวร์	102

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

สารบัญภาพ

	หน้าที่
รูปที่ 2.1 8051 บล็อกไดอะแกรมของ MCS 8051	5
รูปที่ 2.2 แสดงโครงสร้าง port 0 (bit)	6
รูปที่ 2.3 แสดงโครงสร้าง port 1 (bit)	6
รูปที่ 2.4 แสดงโครงสร้าง port 2 (bit)	7
รูปที่ 2.5 แสดงโครงสร้าง port 2 (bit)	7
รูปที่ 2.6 แสดงการทำงานของคำสั่งต่างๆ	9
รูปที่ 2.7 เป็นผังเวลาของสัญญาณซึ่งเกี่ยวข้องกับ Fetch	10
รูปที่ 2.8 แสดงโครงสร้าง port 2 (bit)	10
รูปที่ 2.9 ผังหน่วยความจำสำหรับโปรแกรมสำหรับเบอร์ 8051	11
รูปที่ 2.10 ผังหน่วยความจำสำหรับเก็บโปรแกรมสำหรับเบอร์ 8052	11
รูปที่ 3.1 โมเดลของสถาปัตยกรรม UDA ที่เกี่ยวข้องกับ Visualbasic	19
รูปที่ 3.2 แสดงกลุ่มของฟิลด์	22
รูปที่ 3.3 แสดงฐานข้อมูลที่ไม่มีการปรังปรุง	23
รูปที่ 3.4 แสดงฟิลด์ที่เหลือในตาราง	25
รูปที่ 3.5 แสดงฟิลด์ TranID ใช้สร้างความสัมพันธ์ระหว่างสองตาราง	26
รูปที่ 3.6 ใช้ฟิลด์รหัสวิชาเป็นตัวเชื่อมสองตาราง	26
รูปที่ 3.7 ตารางที่ได้แก้ไขปัญหาในแต่ละขั้นตอน	27
รูปที่ 3.8 ไดอะล็อกบ็อกซ์ เริ่มต้นใช้งาน Access 2000	28
รูปที่ 3.9 ไดอะล็อกบ็อกซ์ สำหรับตาราง	28
รูปที่ 3.10 แสดงการกำหนดชื่อในแต่ละฟิลด์	29
รูปที่ 3.11 VBIDE ชนิด มาตรฐาน EXE	33
รูปที่ 3.12 VBIDE ชนิด DATA Project	33
รูปที่ 3.13 แถบเครื่องมือ VBIDE ทั้ง 2 ชนิด	34
รูปที่ 3.14 Project Explorer ของ VBIDE ชนิด Data Project	34
รูปที่ 3.15 การเลือกใช้ ADO 2.5	35
รูปที่ 3.16 แสดงการใช้คำสั่ง	35
รูปที่ 3.17 แสดงบ็อกซ์ Option	36
รูปที่ 3.18 กำหนดให้ editor ใช้ ฟอนต์ MS Sans Serif ขนาด 10	36
รูปที่ 3.19 รูปแบบของ ฟอนต์ MS Sans Serif ขนาด 10	37
รูปที่ 4.1 ฟอร์มที่ใช้รันคำสั่ง	59

สารบัญภาพ (ต่อ)

หน้าที่

รูปที่ 4.2 สร้างชุดคำสั่ง 60
รูปที่ 4.3 ผลการรันโปรเจ็กต์ 63
รูปที่ 4.4 ตัวอย่างการทำคิวรี 63
รูปที่ 4.5 แบบฟอร์มการค้นหาข้อมูล 64
รูปที่ 4.6 ผลการรันโปรเจ็กต์ 68
รูปที่ 4.7 ทดลองใช้ตัวดำเนินการค้นหาข้อมูล 71
รูปที่ 4.8 กรณีพบเร็คคอร์ดตามเงื่อนไขที่ระบุไว้ 71
รูปที่ 4.9 กรณีไม่พบเร็คคอร์ดตามเงื่อนไขที่ระบุไว้ 72
รูปที่ 5.1 ภาพรวมของระบบ 74
รูปที่ 5.2 ส่วนรับข้อมูลและควบคุมประตู 75
รูปที่ 5.3 ส่วนตรวจสอบตำแหน่งประตู 76
รูปที่ 5.4 ส่วนตรวจสอบตำแหน่งชั้น 77
รูปที่ 5.5 ส่วนของเซิร์ฟเวอร์ 78
รูปที่ 5.6 ส่วนควบคุมการเข้าใช้ห้อง 79
รูปที่ 5.7 ส่วนควบคุมการล็อกอิน 80

สารบัญตาราง

หน้า

ตารางที่ 4-1 ประกอบการอธิบาย 43
ตารางที่ 4-2 ชื่อว่า tblEmployee ประกอบคำอธิบาย 45
ตารางที่ 4-3 student รายชื่อนักศึกษา 54
ตารางที่ 4-4 major รายชื่อคณะที่เปิดสอน 54

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มา

ในปัจจุบันห้องปฏิบัติการไมโครคอมพิวเตอร์ของทางภาควิชาฯนั้นไม่สามารถตรวจสอบหรือรู้ได้เลยว่ามีผู้ใดเข้ามาใช้บริการห้องเวลาใดบ้าง ดังนั้นทางผู้บริหารจึงไม่สามารถวิเคราะห์สิ่งต่างๆได้ อีกทั้งยังไม่มีระบบใดที่จะเข้ามาช่วยผู้ดูแลระบบเลยทั้งทางด้านซอฟต์แวร์หรือฮาร์ดแวร์ จึงเล็งเห็นประโยชน์และความสำคัญของโครงงานนี้

ระบบที่ทำจะเข้ามาช่วยควบคุมการใช้งานห้องไมโครคอมพิวเตอร์ให้มีประสิทธิภาพดียิ่งขึ้นทั้งทางด้านการเก็บข้อมูลของผู้ใช้เพื่อนำไปวิเคราะห์ข้อมูลทางด้านบริหาร และยังช่วยกำหนดสิทธิการเข้ามาใช้บริการห้องปฏิบัติการไมโครคอมพิวเตอร์และยังสามารถตรวจสอบข้อมูลทางด้านประสิทธิภาพข้อเครื่องคอมพิวเตอร์ได้อีกด้วย

1.2 วัตถุประสงค์

1.2.1 เพื่อศึกษาข้อมูลเกี่ยวกับการกำหนดสิทธิ์เพื่อวิเคราะห์และเลือกทฤษฎีรวมไปถึงการนำเอามาประยุกต์ใช้กับโครงงานนี้

1.2.2 เพื่อช่วยควบคุมการใช้งานห้องไมโครคอมพิวเตอร์ให้มีประสิทธิภาพดียิ่งขึ้น

1.2.3 เพื่อช่วยกำหนดสิทธิ์ให้กับบุคคลที่มีสิทธิ์ใช้ห้องปฏิบัติการไมโครคอมพิวเตอร์เท่านั้น

1.2.4 เพื่อเก็บข้อมูลต่างๆของผู้ใช้ที่เป็นประโยชน์และนำข้อมูลไปใช้ทางด้านบริหาร

1.2.5 เพื่อเก็บข้อมูลต่างๆเกี่ยวกับเครื่องเพื่อนำมาวิเคราะห์เกี่ยวกับประสิทธิภาพของเครื่อง

1.3 ขอบเขตของโครงงาน

ระบบช่วยควบคุมห้องปฏิบัติการไมโครคอมพิวเตอร์สามารถแบ่งหน้าที่หลักๆดังนี้

1.3.1 กำหนดสิทธิผู้ที่เข้ามาใช้ห้องปฏิบัติการ

1.3.2 อนุญาตให้ผู้มีสิทธิ์เท่านั้นที่สามารถล็อกอินเข้าสู่ระบบได้

1.3.3 เก็บข้อมูลต่างๆที่เป็นประโยชน์เช่นใครเข้ามาใช้เครื่องบ้างเวลาใดเพื่อวิเคราะห์ทำสถิติ ฯลฯ

1.4 วิธีการดำเนินโครงงาน

1.4.1 ศึกษาค้นคว้าหาข้อมูลและนำเอาระบบเก่ามาประยุกต์และพัฒนาให้มีประสิทธิภาพดียิ่งขึ้น

1.4.2 ศึกษาทฤษฎีพื้นฐานเพื่อนำมาประยุกต์ใช้ในโครงงานนี้

1.4.3 สร้างวงจรและเขียนโปรแกรมควบคุมการปิดเปิดประตู และเช็ค สิทธิ์

1.4.4 ทดสอบการทำงานของวงจรและโปรแกรม

1.4.5 เขียนโปรแกรมเก็บข้อมูลของผู้ใช้ เพื่อนำไปวิเคราะห์ทางด้านบริหาร

1.4.6 สร้างวงจร และ เขียนโปรแกรมเพื่อช่วยลดภาระหน้าที่ของผู้ดูแลระบบในการดูแลห้อง

1.4.7 ทดสอบและทำการติดตั้งระบบทั้งหมด

1.4.8 จัดทำเอกสารของโครงงานนี้

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

บทที่ 2
ทฤษฎีพื้นฐาน

2.1 โครงสร้างของ MCS-51

บทนำ

ไมโครคอนโทรลเลอร์ตระกูล MCS -51 เป็น ไมโครคอนโทรลเลอร์ แบบชิพเดี่ยวที่ผลิตโดยบริษัทอินเทลมีอยู่ด้วยกันหลายเบอร์ด้วยกันซึ่งแต่ละเบอร์นั้นมีคุณสมบัติที่แตกต่างกันตามแต่จะเลือกใช้งานให้เหมาะสมโดยทั่วไปแล้วคุณสมบัติที่แตกต่างกันในแต่ละเบอร์ก็คือ จำนวนหน่วยความจำรอมและแรม, บิตอินพุทเอาท์พุท

2.1.1 คุณสมบัติของไมโครคอนโทรเลอร์ MCS-51

- ต้องการแหล่งจ่ายไฟ +5 โวลต์ ชุดเดียว
- มีหน่วยความจำโปรแกรม (Program Memory) ขนาด 4 กิโลไบต์สำหรับเบอร์ 8051 และ8031 สำหรับเบอร์ 8052 มีหน่วยความจำถึง 8 กิโลไบต์
- มีหน่วยความจำสำหรับเก็บข้อมูล (Data Memory) ขนาด 128 ไบต์ สำหรับเบอร์ 8052 ขึ้นไปมีถึง 256 ไบต์
- หน่วยวามจำสำหรับเก็บโปรแกรมและข้อมูลแยกจากกันอย่างละ 64 กิโลไบต์
- มีไทม์เมอร์ เคาน์เตอร์ ขนาด 16 บิต 2 ชุด (สำหรับ 8052 มี 3 ชุด) ทำงานได้ 4 โหมด
- รับอินเตอร์รัพท์ได้ 6 แหล่ง 5 เวคเตอร์ สำหรับเบอร์ 8052 ขึ้นไปมี 8 แหล่ง 6 เวคเตอร์
- มีพอร์ทรับส่งข้อมูลอนุกรม (UART) 2 พอร์ทแบบ Full Duplex เลือกรูปแบบได้ 4 โหมด
- มีคำสั่งในการทำ AND,OR หรือ Complement ได้ทั้งแบบ 8 บิตและ 1 บิต

2.1.2 โครงสร้างของไมโครคอนโทรลเลอร์ตระกูล MCS – 51

เราจะกล่าวถึงรายละเอียดคร่าวๆของไมโครคอนโทรลเลอร์ตระกูล MCS – 51 โดยมีจุดประสงค์เพื่อให้ผู้อ่านทำความเข้าใจและเห็นภาพกว้างๆ ของไมโครคอนโทรลเลอร์ตระกูลนี้เพื่อเป็นพื้นฐานในการศึกษารายละเอียดต่อไป ไมโครคอนโทรลเลอร์ตระกูล MCS -51 มีสมาชิกในตระกูลหลายเบอร์ด้วยกันแต่ละเบอร์จะมีคุณสมบัติพิเศษบางอย่างแตกต่างกัน เช่น มีหน่วยความจำภายในสำหรับเก็บโปรแกรมและข้อมูลภายในชิพเพิ่มขึ้น มีวงจรเปลี่ยนค่าสัญญาณอนาล็อกเป็นดิจิตอลในตัว สามารถรับสัญญาณอินเตอร์รัปต์ได้หลายชนิด ทำกระบวนการ DMA (Direct Memory Access) ได้ในตัว มีรีจิสเตอร์สำหรับใช้เป็นไทม์เมอร์หรือเคาน์เตอร์เพิ่มขึ้น คุณสมบัติพิเศษที่แตกต่างกันของไมโครคอนโทรลเลอร์แต่ละเบอร์ในตระกูล นี้

ไมโครคอนโทรลเลอร์เบอร์ที่นับได้ว่าเป็นเบอร์พื้นฐานสำหรับตระกูล MCS -51 นี้ได้แก่เบอร์ 8051, 8031 , 8751 โดยเบอร์ 8051 จัดเป็นสมาชิกตัวแรกในตระกูลซึ่งมีหน่วยความจำสำหรับเก็บโปรแกรมภายในชิพเป็น ROM ขนาด 4 กิโลไบต์ และหน่วยความจำ สำหรับเก็บข้อมูลทั่วไปภายใน MCS -51 (RAM) เองจำนวน 128 ไบต์ มีพอร์ตขนาด 8 บิต 4 พอร์ต มีรีจิสเตอร์สำหรับใช้เป็นไทม์เมอร์หรือเคาน์เตอร์ขนาด 16 บิต รวม 2 ตัว รับสัญญาณอินเตอร์รัปต์จากภายนอกได้ 2 ชนิดสามารถรับส่งข้อมูลแบบอนุกรมผ่านทางพอร์ตสื่อสารข้อมูลแบบอนุกรม มีวงจรออสซิลเลเตอร์เพื่อ สร้างสัญญาณนาฬิกาควบคุมการทำงานในตัว ส่วนเบอร์ 8751 จะมีคุณสมบัติเหมือนเบอร์ 8051 ทุกอย่างต่างกันเพียงชนิดของหน่วยความจำสำหรับเก็บโปรแกรมภายในชิพของเบอร์ 8751 จะเป็นอีพรอมแทนที่จะเป็นรอมส่วนเบอร์ 8031 จะเหมือนกับเบอร์ 8051 ต่างเพียงในเบอร์ 8031 ไม่มีหน่วยความจำสำหรับเก็บโปรแกรมภายในชิพเท่านั้น

ไมโครคอนโทรลเลอร์ตระกูล MCS -51 ทุกเบอร์ใช้แรงดันไฟเพียง 5 โวลต์ในการทำงานส่วนกระแสไฟฟ้าที่ใช้จะแตกต่างกันไปตามชนิดของเทคโนโลยีที่ใช้ในการผลิต เบอร์ของไมโครคอนโทรลเลอร์ตระกูลนี้ที่มีตัวอักษร C อยู่ตรงกลางเบอร์ เช่น 80C31, 80C51 จะเป็นเบอร์ของชิพที่ผลิตโดยอาศัยเทคโนโลยี CHMOS ซึ่งใช้พลังงานในการทำงานน้อยกว่าและสามารถควบคุมการใช้พลังงานของตัวชิพได้จากโปรแกรมเพื่อการประหยัดพลังงานในระบบ

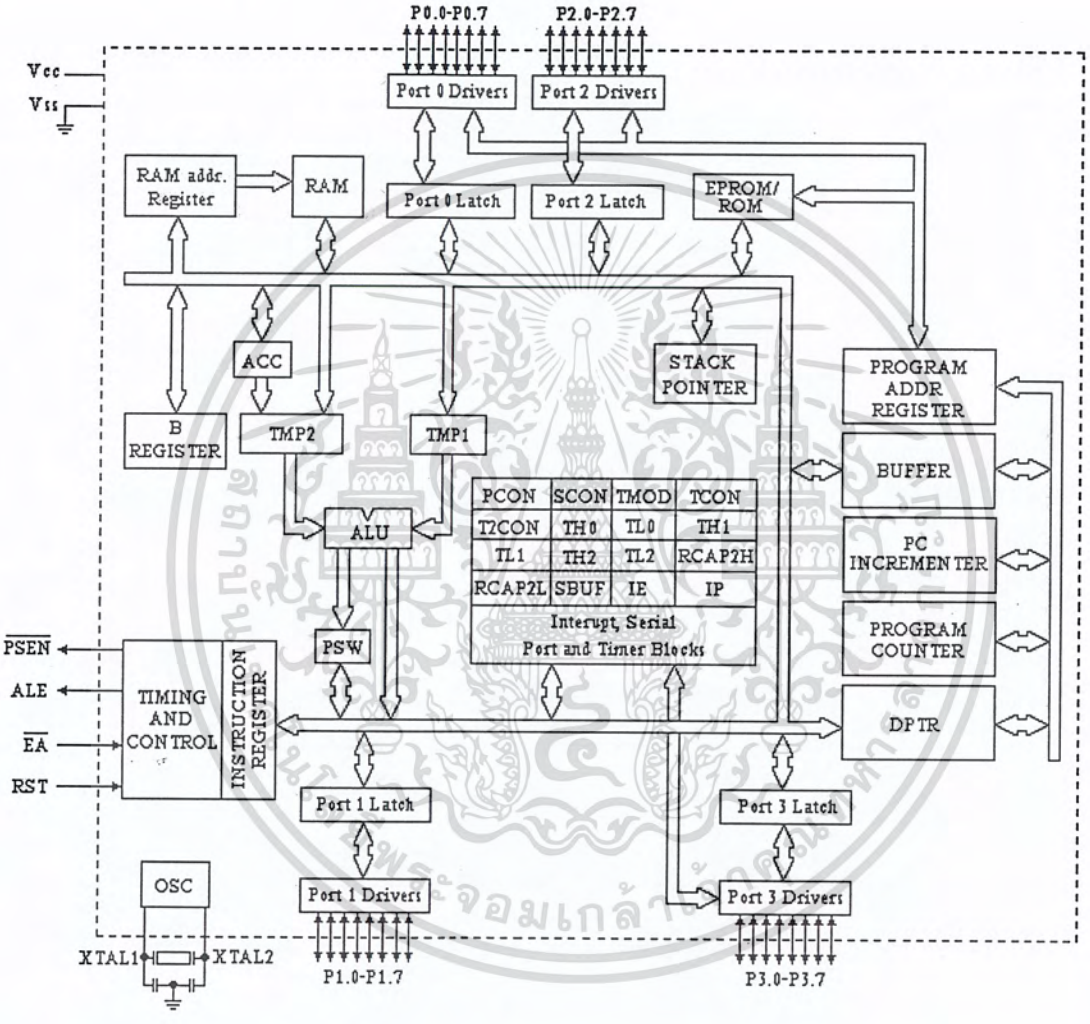
MCS -51 เป็นตระกูลของไมโครคอนโทรลเลอร์ที่ถูกพัฒนาขึ้นมาจากตระกูล MCS -48 ดังนั้นจึงมีความสามารถเหนือกว่าหลายอย่าง เช่น ความเร็วในการประมวลผลของ MCS -51 สามารถใช้ความถี่ได้ถึง 12 เมกะเฮิรตซ์ หรือบางเบอร์ในตระกูล สามารถใช้ได้ถึง 16 เมกะเฮิรตซ์ ทำให้ช่วงระยะเวลาในการทำงานแต่ละคำสั่งน้อยมาก

หน้าที่การใช้งานแต่ละขาของชิพไมโครคอนโทรลเลอร์ตระกูล MCS -51 มีดังนี้

- ขา Vss (ขา 20) สำหรับ ต่อลง GND
- ขา Vcc (ขา 40) สำหรับ ต่อลงแหล่งจ่ายกระแสตรงขนาด 5V (DC)
- ขา พอร์ต 0 (ขา32 - 39) มี 8 ขาใช้เป็นขาสำหรับพอร์ต 0 บิตขนาด8 บิต (P0.0 – P0.7) แบบ Open Drain Bidirectional พอร์ตนี้สามารถใช้งานเป็นอินพุทและเอาท์พุทพอร์ตทั่วไปได้โดยหากใช้งานเป็นอินพุทพอร์ต ต้องโหลดค่า 1 ไปยังแต่ละบิตของพอร์ตนี้เพื่อบังคับให้ขาอยู่ในสถานะถูกปล่อยลอย (มีสถานะ High Impedance) นอกจากใช้งานเป็น อินพุทและเอาท์พุทพอร์ตแล้ว พอร์ต 0 ยังใช้ในการติดต่อหน่วยความจำสำหรับเก็บโปรแกรมและข้อมูลภายนอกชิพด้วย โดยส่งค่า แอดเดรสบัสไบต์ต่ำ (A0 – A7) และมัลติเพลกซ์กับการรับข้อมูล (D0 – D7) จากหน่วยความจำภายนอกในระหว่างการเขียนหรืออ่านข้อมูลโดยมีวงจรพูลอัพภายใน
- ขาพอร์ต 1 (ขา 1 – 8) มี 8 ขา ใช้เป็นขาสำหรับพอร์ต 1 (P1.0 – P1.7) สามารถใช้งานเป็นอินพุทและเอาท์พุทพอร์ต

2.1.3 โครงสร้างภายในของ 8051

ไมโครคอนโทรเลอร์ MCS-51 ใช้เทคโนโลยีในการผลิตเป็นแบบ NMOS และ CMOS เบอร์ 8032 และ 8052 จะมีรอมแบบพื้นฐาน อยู่ภายในจึงสะดวกสำหรับโปรแกรมเมอร์ที่จะเขียนโปรแกรมด้วยภาษาเบสิก โครงสร้างภายในสำหรับเบอร์ 8051 เป็นดังรูปโครงสร้างภายในของ 8051

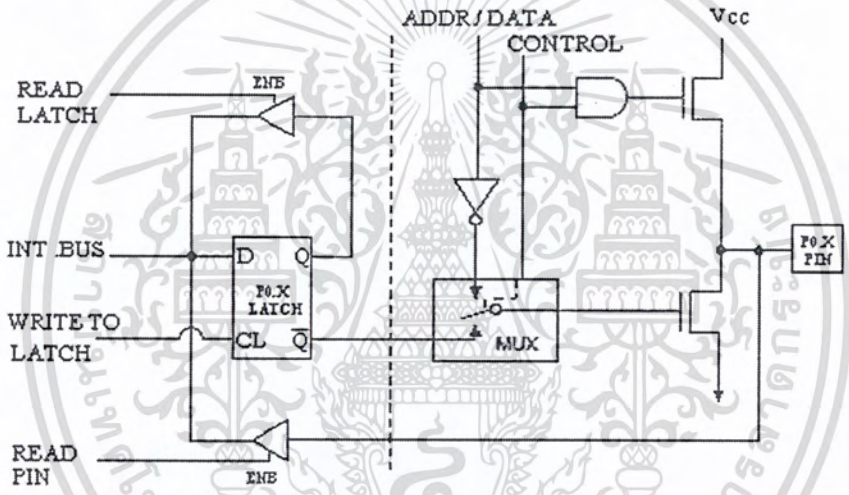


รูปที่ 2.1 8051 บล็อกไดอะแกรมของ MCS 8051

2.1.4 Port ต่างๆ ภายใน 8051

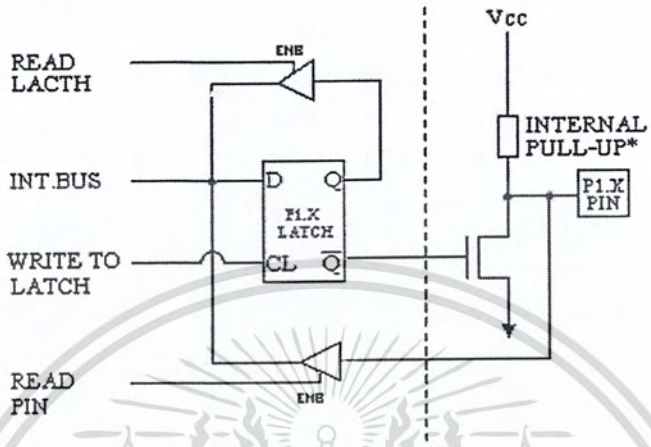
พอร์ทของ 8051 เป็น ไมโครคอนโทรเลอร์ ขนาด 40 ขา ซึ่งมีขาต่าง ๆดังนี้

- Vcc (ขา 40) ต่อกับ +5V
- Vss (ขา 20)เป็นขา GND
- พอร์ท 0 (ขา 32-39) มีทั้งหมด 8 บิต คือ P0.7- P0.0 ใช้งานได้ 2 หน้าที่คือ แอดเดรสบัส และ ดาต้าบัส เมื่อต้องการให้ทำงานเป็นอินพุตพอร์ทต้องส่งลอจิก “1” ไปยังพอร์ทนี้ จะมีผลให้ Q bar ของ D-FF เป็น “0” ทำให้ FETตัวล่างมีสถานะเป็น OFF สัญญาณที่ใช้อ่านอินพุทพอร์ทแลทซ์ โดยส่งสัญญาณ READ LATCH ไปกระตุ้นที่ Tri State Buffer ตัวบนและการอ่านพอร์ท(pin) จะใช้สัญญาณ Read(pin)



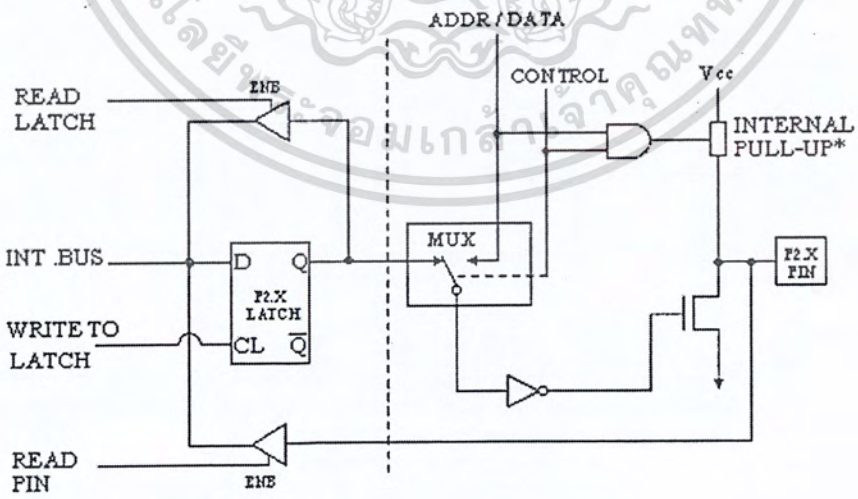
รูปที่ 2.2 แสดงโครงสร้าง port 0 (bit)

- พอร์ท 1 (ขา 1-8) มีทั้งหมด 8 บิตคือ (P1.0- P1.7) มีโครงสร้างคล้าย พอร์ท 0 แต่จะใช้ความต้านทานภายในพูลอัพแทน (Internal Pull up Register) มีโครงสร้างดังรูป



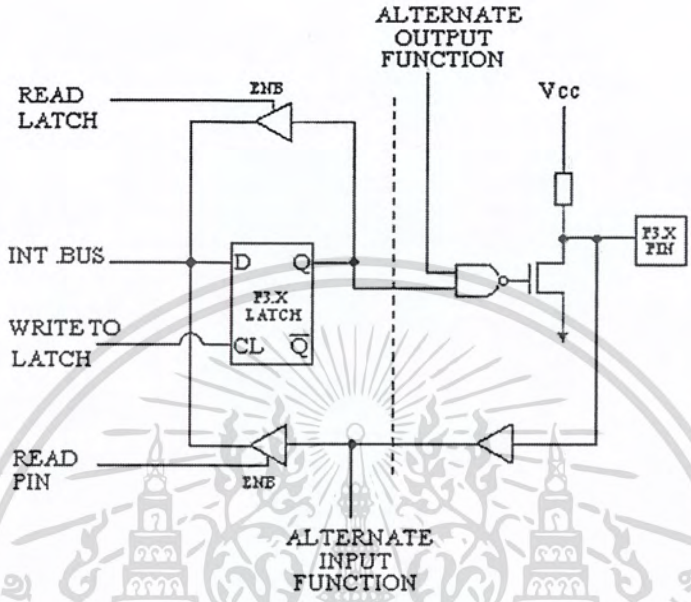
รูปที่ 2.3 แสดงโครงสร้าง port 1 (bit)

- พอร์ท 2 (ขา 21-28) มีทั้งหมด 8 บิต คือขา (P2.7- P2.0)มีโครงสร้างคล้าย พอร์ท 0 โดยมี FETตัวล่างตัวเดียวส่วนด้านบนใช้ความต้านทานพูลอัพแทน(Internal Pull up) พอร์ทนี้ทำงาน 2 หน้าที่ คือสามารถใช้เป็น แอดเดรสบัสขนาด 8 บิต และเป็น ไอโอพอร์ทใช้งานทั่วไปเมื่อจะใช้งานเป็นอินพุทพอร์ทต้องส่งลอจิก “1” มาที่พอร์ทนี้ก่อนเพื่อบังคับให้ FETอยู่ในสภาวะ off ดังรูป



รูปที่ 2.4 แสดงโครงสร้าง port 2 (bit)

- พอร์ท 3 (ขา 10-17) มีทั้งหมด 8 บิต คือ ขา P3.7-P3.0 มีโครงสร้างคล้าย พอร์ท 1 ทำงานได้ 2 หน้าที่ คือ เป็นไอโอพอร์ท ถ้าจะโปรแกรมให้เป็น อินพุทพอร์ทต้องส่ง ลอจิก "1" มาที่พอร์ทนี้ก่อน



รูปที่ 2.5 แสดงโครงสร้าง port 3(bit)

อีกหน้าที่หนึ่งก็คือใช้ส่งสัญญาณควบคุมออกมาและรับสัญญาณเข้าไปสัญญาณต่าง ๆ ดังนี้

P3.0/RXD (Serial Input Port) เป็นขาที่ใช้รับข้อมูลแบบอนุกรม (UART)

P3.1/TXD (Serial Output Port) เป็นขาที่ใช้ส่งข้อมูลแบบอนุกรม (UART)

P3.2/INT0 (External Interrupt 0) ใช้รับสัญญาณการขัดจังหวะจากภายนอกเบอร์ 0

P3.3/INT1 (External Interrupt 1) ใช้รับสัญญาณการขัดจังหวะจากภายนอกเบอร์ 1

P3.4/T0 (Counter 0 External Input) ขารับสัญญาณพัลซ์อินพุทเข้าไปยังวงจรเคาท์เตอร์0

P3.5/T1 (Counter 1 External Input) ขารับสัญญาณพัลซ์อินพุทเข้าไปยังวงจรเคาท์เตอร์1

P3.6/WR (External Data Memory Write Strobe) ขาสัญญาณควบคุมการเขียนข้อมูลลงหน่วยความจำข้อมูลภายนอก

P3.7/RD (External Data Memory Read Strobe) ขาสัญญาณควบคุมการอ่านข้อมูลจากหน่วยความจำข้อมูลภายนอก

ALE (ขา 30) เป็นขาส่ง Strobe สำหรับใช้ในการแล็ซแอดเดรสไบต์ ต่ำ (A7-A0) ที่ส่งออกมาจาก port 0 สัญญาณนี้จะ active ทุก ๆ 2 ครั้งในแมชชีนไซเกิล

PSEN (ขา 29) เป็นขา Strobe สำหรับใช้อ่านข้อมูลจากโปรแกรมเมมโมรี่ภายนอกสัญญาณนี้จะส่งออกมา 2 ครั้งในแต่ละแมชชีนไซเกิลแต่ถ้าเป็นการอ่านโปรแกรมเมมโมรี่ภายในจะไม่มีสัญญาณออกที่ขานี้

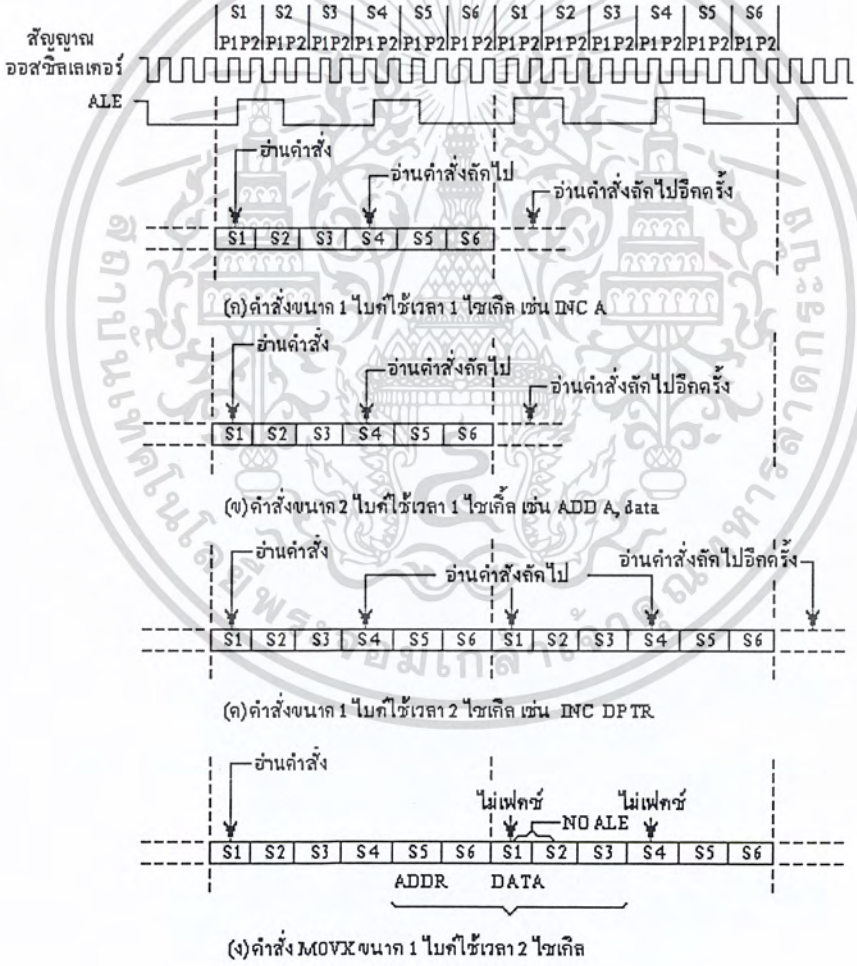
EA (ขา 31) ใช้เลือกหน่วยความจำโปรแกรมภายนอก ป้อน " 0 " จะอ่านโปรแกรมจากภายนอกชิพ ป้อน "1" จะอ่านโปรแกรมจากภายในชิพ

RST (ขา 9) ขารีเซ็ทจะรีเซ็ทได้ก็ต่อเมื่อป้อนลอจิก " 1 " เข้าที่ขานี้นานอย่างน้อย 2 แมชชีนไซเกิล

XTAL1 (ขา 19) ใช้ต่อคริสตอลภายนอกโดยเป็นอินพุทเข้าสู่วงจรออสซิสเลเตอร์ภายใน

XTAL2 (ขา 18) ใช้ต่อคริสตอลภายนอกโดยเป็นเอาท์พุทของวงจรออสซิสเลเตอร์ภายใน

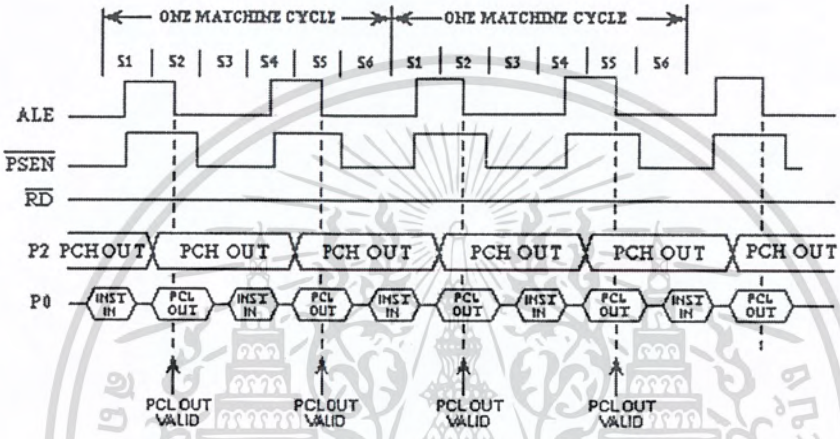
2.1.5 ผังเวลาของ CPU (CPU Timing)



รูปที่ 2.6 แสดงการทำงานของคำสั่งต่างๆ

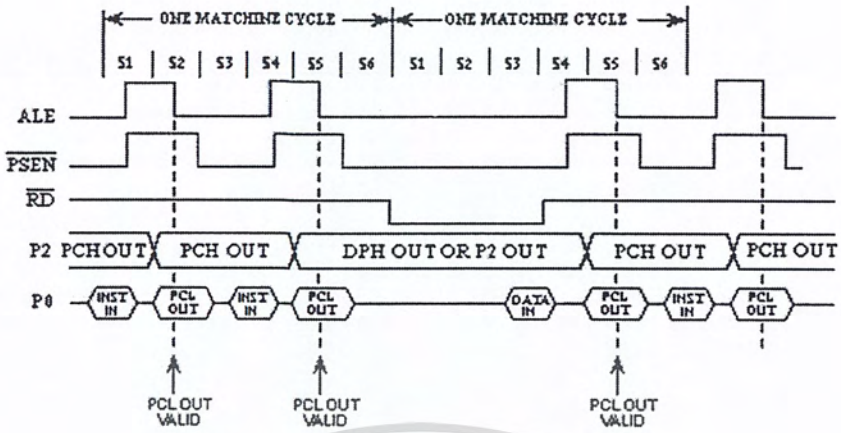
การทำงานใน 1 คำสั่งต่ำสุดจะกินเวลาเพียง 1 ไมโครวินาที เช่นคำสั่ง INC A ซึ่งเป็นคำสั่ง 1 ไบต์ 1 แมชชีนไซเกิลซึ่งจะใช้สัญญาณนาฬิกาไปเท่ากับ 12 ลูก โดยสัญญาณนาฬิกาลูกที่ 1 และ 2 จะอยู่ในช่วง S1 P1 และ S1 P2 และ สัญญาณนาฬิกาลูกที่ 12 ก็จะอยู่ในช่วง S6P2 นั่งเอง (ปรกติแล้ว CPU จะทำงานด้วยความเร็วเท่ากับ 12 MHz ดังนั้นสัญญาณนาฬิกา 12 ลูกจะกินเวลาเท่ากับ 1 ไมโครวินาที)

*** คำว่า 1 แมชชีนไซเกิล คือช่วงการทำงานตั้งแต่ S1 จนถึง S6 ***



รูปที่ 2.7 เป็นผังเวลาของสัญญาณซึ่งเกี่ยวข้องกับ Fetch

รูปที่ 2.7 เป็นผังเวลาของสัญญาณซึ่งเกี่ยวข้องกับ Fetch เมื่อส่วนของหน่วยความจำโปรแกรมอยู่ภายนอก ดังนั้น สัญญาณที่จำนำไปใช้อ่านออปโค้ด จากหน่วยความจำโปรแกรมก็คือ PSEN ซึ่งจะ active 2 ครั้งใน 1 แมชชีนไซเกิล ดังนั้น สัญญาณที่ใช้อ่านข้อมูลจาก หน่วยความจำโปรแกรม จะใช้สัญญาณ PSEN

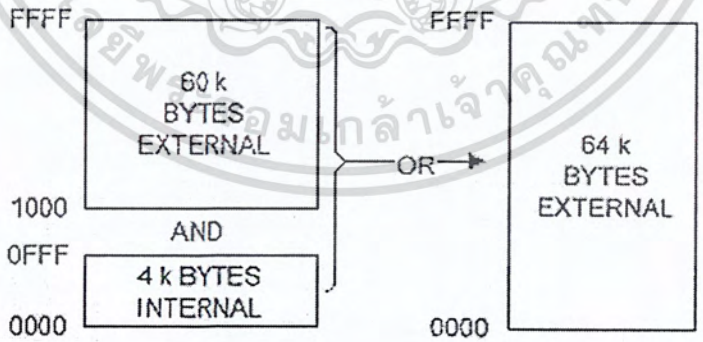


รูปที่ 2.8 แสดงโครงสร้าง port 2 (bit)

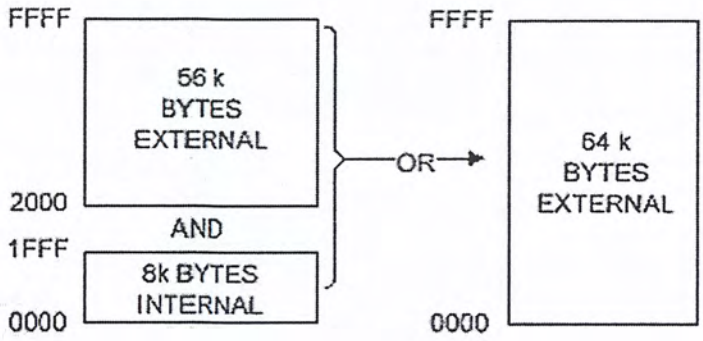
2.1.6 การแบ่งประเภทของหน่วยความจำ

หน่วยความจำที่ใช้กับ MCS –51 มีอยู่ด้วยกัน 2 ชนิด คือ

- หน่วยความจำสำหรับเก็บโปรแกรม (Program Memory)
- หน่วยความจำสำหรับเก็บข้อมูล (Data Memory)
- หน่วยความจำสำหรับเก็บโปรแกรม เป็นหน่วยความจำที่ใช้เก็บโปรแกรมสั่งงานบรรจุอยู่ในชิพ 8051 ส่วนที่เป็น หน่วยความจำโปรแกรม ก็คือ ROM ขนาด 4 กิโลไบต์นั่นเอง แต่ถ้าเป็นเบอร์ 8052 จะมี ROM ขนาด 8 กิโลไบต์ ดังแสดงในรูป



รูปที่ 2.9 ผังหน่วยความจำสำหรับโปรแกรมสำหรับเบอร์ 8051



รูปที่ 2.10 ผังหน่วยความจำสำหรับเก็บโปรแกรมสำหรับเบอร์ 8052

2.1.7 พื้นที่หน่วยความจำที่เข้าถึงข้อมูลทางอ้อมเท่านั้น (Indirect Addreses Area)

พื้นที่หน่วยความจำบริเวณ (80h – FFh) เป็นพื้นที่ซ้อนกันอยู่อย่างละ 128 ไบต์ โดยส่วนแรกจะเป็น SFR address และ Indirect Address Area ดังนั้นถ้าจะติดต่อกับ SFR จะต้องใช้คำสั่งแบบเข้าถึงข้อมูลโดยตรงเท่านั้น (Direct Address Area) ส่วนพื้นที่อีกส่วนหนึ่งจะเข้าถึงข้อมูลแบบทางอ้อมเท่านั้น (Indirect Address Area) ส่วนตำแหน่ง (00h – 7Fh) จะเข้าถึงข้อมูลได้ทั้ง 2 แบบ

2.1.8 พื้นที่หน่วยความจำที่เข้าถึงข้อมูลโดยตรงและทางอ้อม (Direct and Indirect Address Area)

- Register Bank (Register Banks 0-3) ตั้งแต่ตำแหน่ง (00-Fh) จะเป็นส่วนของรีจิสเตอร์แบงค์ (0-3) โดยแบ่งเป็นแบงค์ แบงค์ละ 8 ไบต์รวมแล้วได้ 32 ไบต์ (แต่ละแบงค์นั้นจะมีรีจิสเตอร์ R0,R1,R2,R3,R4,R5,R6,R7) ถ้า CPU ทำงานอยู่ที่แบงค์ 3 เมื่อถูก reset ก็จะกลับมาทำงานที่แบงค์ 0 เสมอ และ SP จะมาเริ่มต้นที่ตำแหน่ง 07h ทันที
- บริเวณหน่วยความจำที่ใช้คำสั่งอ่านเขียนทีละกับบิตได้ (Bit Addressable Area) พื้นที่ตั้งแต่ address (20h-7Fh) จำนวน 16 ไบต์หรือแบ่งเป็นบิตจะได้เท่ากับ 128 บิตซึ่งจะมีตำแหน่งบิตดังนี้ 00,01,02,03,04,05,06,07 จนถึง 7Fh
- บริเวณหน่วยความจำที่ใช้งานทั่วไป (Scratch Pad Area) พื้นที่ตั้งแต่ (30h-7Fh) จะเขียนข้อมูลได้ทีละไบต์เท่านั้น ไม่สามารถใช้คำสั่งเกี่ยวกับบิตได้ถ้าย้ายเนื้อที่แสตคมาบริเวณนี้

2.2 พอร์ตสื่อสารข้อมูลแบบอนุกรมใน MCS-51

การใช้งานสื่อสารข้อมูลอนุกรมใน MCS-51 มีความสะดวกและคล่องตัวสูงทั้งนี้เนื่องจากผู้ใช้สามารถกำหนดการทำงานที่แตกต่างกันได้ถึง 4 ประเภทโดยสามารถกำหนดได้จากค่าของบิตในรีจิสเตอร์ใช้งานเฉพาะ SCON การใช้งานที่แตกต่างกัน 4 ประเภทนี้ มีจุดประสงค์ เพื่อความคล่องตัวในการรับหรือส่งข้อมูลแบบอนุกรมแต่ระประเภทดังนี้

- โหมด 0 การทำงานของพอร์ตสื่อสารข้อมูลแบบอนุกรมในโหมด 0 ขา RXD จะสำหรับรับข้อมูลและส่งข้อมูล ส่วนขา TXD มีไว้เพื่อใช้สร้างสัญญาณ shift clock เพื่อกำหนดจังหวะในการรับและส่งข้อมูลในโหมดนี้ การรับส่งข้อมูลจะเป็นแบบ 8 บิต โดยเริ่มนับและส่งบิตต่ำสุดก่อน อัตราการรับส่งข้อมูลในการทำงานโหมด 0 ถูกกำหนดไว้ที่ 1/12 ของความถี่ออสซิลเลเตอร์ที่ใช้ การทำงานของพอร์ตสื่อสารอนุกรมในโหมด 0 จะไม่มีบิตเริ่มต้นของข้อมูลและบิตสิ้นสุดของข้อมูล เพราะจังหวะในการรับและส่งข้อมูลถูกกำหนดจากสัญญาณ shift clock

- โหมด1 ทำงานแบบที่สองหรือการทำงานโหมด 1 นี้ มีการรับส่งข้อมูลครั้งละ 10 บิต ข้อมูลจะถูกส่งออกไปภายนอกผ่านทางขา TDX และ การรับส่งข้อมูลเข้ามาทางขา RXD ข้อมูลทั้ง 10 บิต ประกอบไปด้วยบิตเริ่มต้นของข้อมูล 1 บิต (มีค่าเป็น 0 เสมอ) บิตข้อมูล 8 บิต (รับและส่งบิตต่ำสุดก่อน) และบิตสิ้นสุดข้อมูลอีก 1 บิต (มีค่าเป็น 1 เสมอ) ในขณะทำการรับส่งข้อมูล ค่าในบิตสิ้นสุดของข้อมูลที่ได้รับจะได้ไปอยู่ในบิต RB8 ของ รีจิสเตอร์ใช้งานเฉพาะ SCON อัตราเร็วในการรับหรือส่งข้อมูลของพอร์ตสื่อสารข้อมูลแบบอนุกรมในโหมดนี้สามารถเปลี่ยนแปลงได้

- โหมด 2 การทำงานแบบที่สามหรือโหมด 2 จะมีการรับและส่งข้อมูลครั้งละ 11 บิต ข้อมูลจะถูกส่งออกภายนอกผ่านทางขา TXD และรับเข้ามาผ่านทางขา RXD ข้อมูลที่รับและส่งทั้ง 11 บิตจะประกอบไปด้วยบิตเริ่มต้นของข้อมูล 1 บิต บิตข้อมูล 8 บิต ตามด้วยบิตที่ 9 ซึ่งเป็นบิตที่สามารถกำหนดให้มีค่าเป็นศูนย์หรือหนึ่งก็ได้ และบิตสุดท้ายคือบิตสิ้นสุดข้อมูล ดังนั้นจึงมีจำนวนบิตที่รับส่งทั้งหมด 11 บิต

- โหมด 3 การทำงานของพอร์ตสื่อสารข้อมูลแบบอนุกรมแบบสุดท้าย คือการทำงานในโหมด 3 ในการทำงานโหมดนี้ข้อมูลจำนวน 11 บิตจะถูกส่งออกทางขา TXD และรับข้อมูลผ่านทางขา RXD ข้อมูลทั้ง 11 บิตประกอบไปด้วยบิตเริ่มต้นของข้อมูล 1 บิต บิตข้อมูล 8 บิตตามด้วยบิตที่ 9 ซึ่งเป็นบิตที่สามารถกำหนดค่าได้เหมือนในโหมดที่ 2 และบิตสุดท้ายคือบิตสิ้นสุดข้อมูล อัตราเร็วในการรับหรือส่งข้อมูลสามารถเปลี่ยนแปลงได้ ซึ่งในโหมดนี้สามารถกำหนดอัตราเร็วในการรับหรือส่งข้อมูลได้ตามความต้องการของผู้ใช้งาน

2.2.1 อัตราเร็วในการรับส่งข้อมูล

Baud rate หมายความถึงอัตราเร็วในการรับหรือส่งข้อมูล โดยใน MCS-51 ค่าอัตราเร็วในการรับส่งข้อมูลจะมีค่าเท่าใดขึ้นอยู่กับการทำงานในแต่ละโหมดของ port สื่อสารข้อมูลแบบอนุกรมดังนี้

$$\text{Baud rate โหมด 0} = \frac{\text{ความถี่ออสซิสเลเตอร์ที่ใช้}}{12}$$

หากใช้คริสตอลความถี่ 12 MHz ค่า baud rate ของ port สื่อสารข้อมูลแบบอนุกรมโหมด 0 จะมีค่าสูงถึง 1 MHz.ในโหมดที่2 ค่า baud rate ขึ้นอยู่กับค่าของ บิต SMOD ที่อยู่ในรีจิสเตอร์ใช้งานเฉพาะ PCON โดยบิต SMOD = 0 ค่า baud rate

จะเป็น 1/64 ของความถี่ออสซิลเลเตอร์ที่ใช้ บิต SMOD = 1 ค่า baud rate จะเป็น 1/32 ของความถี่ออสซิลเลเตอร์ที่ใช้หลังจากการ reset MCS-51 ค่าในบิต SMOD จะเป็น 0 เสมอและเราสามารถเขียนสูตรสำหรับคำนวณค่า baud rate ได้ดังสมการนี้

$$\text{Baud rate โหมด 2} = \left[\frac{2^{(SMOD)} \times (\text{ความถี่ออสซิสเลเตอร์ที่ใช้})}{64} \right]$$

หากใช้คริสตอลความถี่ 12 MHz baud rate สูงสุดในการทำงานโหมดนี้คือ 375 K

2.2.2 อธิบายการทำงานพอร์ตสื่อสารข้อมูลแบบอนุกรม (โหมด 1 เพิ่มเติม)

การทำงานของพอร์ตสื่อสารข้อมูลแบบอนุกรมโหมด 1 ข้อมูล 10 บิต ถูกส่งออกไปภายนอกผ่านขา TXD และรับเข้ามาจากภายนอกผ่านทางขา RXD ข้อมูลที่รับหรือส่งจะประกอบไปด้วย บิตเริ่มต้นของข้อมูล 1 บิต (0) บิตข้อมูล 8 บิตที่เริ่มรับหรือส่งด้วยบิตต่ำสุดก่อน (LSB first) ตามด้วยบิตสิ้นสุดของข้อมูลอีก 1 บิต (1) ดังแสดงในรูปที่

ในการรับข้อมูล บิตสิ้นสุดของข้อมูลจะถูกนำไปไว้ในบิต RB 8 ของรีจิสเตอร์ใช้งานเฉพาะ SCON ค่า baud rate ถูกกำหนดโดยอัตราการเกิด overflow ของไทม์เมอร์ 1 (timer 1 overflow rate) ส่วนใน 8052 ซึ่งมีไทม์เมอร์ 2 อย่างใดอย่างหนึ่งเพิ่มขึ้นมา ค่า baud rate สามารถถูกกำหนดโดยอัตราการเกิด overflow ของไทม์เมอร์ 1 หรือ ไทม์เมอร์ 2 อย่างใดอย่างหนึ่งหรือสองอย่าง (ใช้ตัวหนึ่งสำหรับการกำหนด + ในการส่งข้อมูลและอีกตัวหนึ่งสำหรับการรับข้อมูล) โดยใช้บิต RCLK และบิต TCLK เป็นตัวเลือก

2.3 Active Directory Service Interfaces

Active Directory ™ Service Interfaces (ADSI) เป็นรูปแบบความสามารถ directory services ในระบบเครือข่ายที่แตกต่างกัน ซึ่งจัดเตรียมการจัดสรรสิ่งแวดล้อมในการคำนวณ รวมถึงจัดเตรียมการติดต่อสื่อสารกับ directory service เพื่อการบริหารทรัพยากรในระบบโดยที่ผู้บริหารและผู้พัฒนาระบบสามารถใช้บริการของ ADSI ในการให้รายละเอียดและบริหารทรัพยากรใน directory service

directory services (ADSI) ทำให้การทำงานที่เกี่ยวกับการบริหารจัดการได้ง่ายขึ้น, เช่นการเพิ่ม users การบริหาร Printer และกำหนดขอบเขตของทรัพยากรรวมถึงจัดสรรสิ่งแวดล้อมในการคำนวณ

2.4 Enabling NT users

ทางที่ง่ายที่สุดในการ disable หรือ enable user account คือการทำผ่าน ADSI (ซึ่งมีอยู่แล้วใน windows 2000 และใน windows NT ก็สามารถที่จะ download software มาลงได้) ซึ่งโค้ด ข้างล่างจะแสดงถึงการ disable หรือ enable user's account โดยผ่านทาง Flags parameter โดยการset flag UF_ ACCOUNTDISABLE ของ user flag

2.5 Account Disable

เราสามารถ disable user account โดยการ set Account Disable property ให้เป็น True ในการ interface แบบ ADIS User

Example Code [Visual Basic]

```
Set usr = GetObject ("LDAP://  CN = Jsmith, OU=sales, DC = ArcadiaBay,DC Com")
usr . AccountDisabled = TRUE 'disabled the account
urs . SetInfo
```

2.6 Event Logging

Application หลายอันจะเก็บ record error log ใน format ที่แตกต่างกันโดยคุณไม่สามารถรวบรวมข้อมูลให้อยู่ในรูป complete report ได้ดังนั้นจึงต้องทำการเช็คจากหลายๆแหล่งข้อมูลโดย Event Logging จะจัดการข้อมูลให้อยู่ในรูปแบบ มาตรฐานที่เป็นกลางทำให้เป็น 1 เดียว Event Viewer ก็จะช่วยให้เห็น Log โดยโปรแกรมจะสามารถให้เราตรวจสอบ Log ได้

2.6.1 About Event Logging

เมื่อเกิด error ขึ้นก็จะทำการตรวจสอบว่าอุปกรณ์ใดที่ทำให้เกิด error และพยายามที่จะ recorver ข้อมูลที่หายไปและทำการป้องกันไม่เกิดขึ้นอีก มันจะเป็นการดีมากที่ OS , Application , System Service จะเก็บ Event ที่สำคัญเอาไว้ เช่น Low Memory Condition ผู้ควบคุมระบบจะสามารถระบุได้ว่าอะไรที่ทำให้เกิด error โดยดู Event Log ตามลำดับและผู้ควบคุมระบบยังสามารถระบุได้ว่าที่ใดจะเกิดความเสียหาย

2.6.2 Event Type

Event Type มีด้วยกันทั้งหมด 5 แบบของ Event ที่สามารถ Log ได้ทุกๆ Event มีลักษณะของการจัดการข้อมูลเหมือนกันเหมือนเป็น Option ว่าสามารถรวม Event Specific Data ได้ Application จะระบุว่าอยู่ใน Type ไหนใน 5 แบบโดยEvent เป็นได้แค่ชนิดเดียวเพราะฉะนั้น Event Viewer จะจัดจำพวกของ Type และจัดว่าควรอยู่ตำแหน่งใด

- Information จะระบุข้อมูลที่ไม่ได้เกิดขึ้นบ่อยหรือข้อมูลที่สำคัญ เช่น SQL Server Load เสร็จ SQL Server Start แต่จะไม่เหมาะกับโปรแกรมจิปาถะ เช่น Excell
- Warning จะระบุเหตุการณ์ที่ไม่สำคัญมากในทันทีทันใดโดยจะทำการเตือนไว้ก่อนการใช้ทรัพยากรเป็นตัวอย่างที่ดีอันหนึ่ง เช่น ดิสก์เหลือน้อยแล้วก็จะทำการโชว์
- Error จะทำการระบุปัญหาที่สำคัญที่ผู้ใช้ควรจะรู้และการสูญหายของข้อมูล เช่น ถ้าผู้ใช้ทำการสั่ง reboot เครื่องแล้วเครื่องไม่ boot จะทำการโชว์ Event Error
- Success audit จะระบุการตรวจสอบที่ประสบผลสำเร็จเป็น Event ที่ปลอดภัยจะโชว์เมื่อตรวจสอบสำเร็จก็จะทำการโชว์
- Failure audit จะระบุการทำงานที่ไม่ประสบผลสำเร็จ เช่น พยายามเปิด file ข้อมูลแล้วไม่สามารถเปิดได้ จะทำการโชว์

สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

2.6.3 EVENTLOGRECORD

```
typedef struct _EVENTLOGRECORD {
        DWORD  Length;
        DWORD  Reserved;
        DWORD  RecordNumber;
        DWORD  TimeGenerated;
        DWORD  TimeWritten;
        DWORD  EventID;
        WORD   EventType;
        WORD   NumStrings;
        WORD   EventCategory;
        WORD   ReservedFlags;
        DWORD  ClosingRecordNumber;
        DWORD  StringOffset;
        DWORD  UserSidLength;
        DWORD  UserSidOffset;
        DWORD  DataLength;
        DWORD  DataOffset;
        //
        // Followed by:
        //
        // TCHAR SourceName[]
        // TCHAR Computername[]
        // SID   UserSid
        // TCHAR Strings[]
        // BYTE  Data[]
        // CHAR  Pad[]
        // DWORD Length;
        //
} EVENTLOGRECORD, *PEVENTLOGRECORD;
```

46144

Example Code [C++]

```
IADsUser *pUser;
HRESULT hr = S_OK;
LPWSTR adsPath;
adsPath=L"WinNT://Fabrikam/JSmith";
hr = ADsGetObject(adsPath,IID_IADsUser,(void**)&pUser);
hr = pUser->put_AccountDisabled(true);
hr = pUser->SetInfo();
```

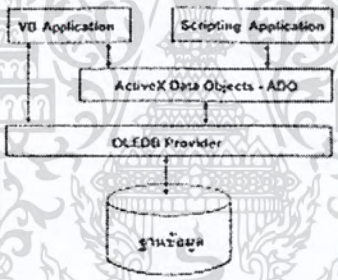
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

บทที่ 3
พื้นฐานการเขียนโปรแกรมด้านฐานข้อมูล

บทนำ

การเขียนโปรแกรมด้านฐานข้อมูลด้วย visual Basic เป็นอีกหัวข้อหนึ่งที่น่าศึกษาเป็นอย่างยิ่ง เนื่องจากว่า ในปัจจุบัน ไมโครซอฟท์ ได้คิดค้นเทคโนโลยีในการเข้าถึงข้อมูลในฐานข้อมูลรูปแบบใหม่ ซึ่งมีชื่อเรียกว่า Universal Data Access เรียกย่อๆ ว่า UDA โดยมีแนวความคิดว่า ต้องการเป็นเทคโนโลยีที่จะสามารถเข้าถึงข้อมูลในฐานข้อมูลจาก BDBMS(Relational Database Management System) ได้หลายๆ ชนิด เช่น MS SQL Server, Oracale, MS Access เป็นต้นรวมถึงข้อมูลที่ถูกจัดเก็บในฐานข้อมูลตามมาตรฐานอื่นๆ อีก เช่น มาตรฐาน ISAM (Index Swquential Access methods) และยังได้ขยายแนวความคิดให้สถาปัตยกรรม UDA นี้ ครอบคลุมถึงการเข้าถึงข้อมูลในฐานข้อมูลผ่านทางอินเตอร์เน็ตอีกด้วย

การเขียนโปรแกรมด้านฐานข้อมูลโดยใช้เทคโนโลยี UDA จะอาศัย OLE Provider เป็นตัวจัดการในการเข้าถึงข้อมูลในฐานข้อมูลทั้งหมด โดยจะใช้เทคโนโลยี ActiveX Data Object หรือเรียกย่อๆ ว่า ADO ทำหน้าที่เป็นตัวกลางระหว่าง OLE Provider กับ Visual Basic ดังรูปที่ 3-1



รูป 3.1 โมเดลของสถาปัตยกรรม UDA ที่เกี่ยวข้องกับ Visualbasic

จากรูปที่ 3-1 จะเห็นว่า ADO จะเป็นตัวกลางที่จะไปสั่งให้ OLE Provider เข้าถึงข้อมูลในฐานข้อมูล ซึ่งจะถูกเรียกโดยรวมว่า Microsoft Data Access Component (MDAC) ในส่วนของ Scripting Applicatoin เป็นการเข้าถึงข้อมูลในฐานข้อมูลผ่านทางอินเตอร์เน็ต โดยใช้ภาษาสคริปต์เป็นตัวเรียกใช้งาน คือ Active Server Pages (ASP)

สำหรับในการเขียนโปรแกรมด้านฐานข้อมูลด้วย Visual Basic โดยอาศัยเทคโนโลยี ADO ผู้เขียนจะแบ่งออกเป็น 2 ส่วนใหญ่ๆ คือ

- การใช้งานกลุ่มคอนโทรลด้านฐานข้อมูล (Bound Controls and ActiveX Bound Control) ที่ใช้ OLEDB Provider เป็นตัวเข้าถึงข้อมูลในฐานข้อมูล โดยที่คอนโทรลในกลุ่มนี้ จะมีคำว่า OLEDB ต่อท้าย
- การเรียกใช้งานกลุ่มของ ActiveX DATA OBJECT โดยตรง

3.1 รูปแบบการติดต่อกับฐานข้อมูลด้วย Visual Basic

ในการติดต่อกับฐานข้อมูล โดยปกติแล้ว VB จะเชื่อมโยงผ่านทาง Database Engine ที่เรียกว่า JET Engine จึงอาจกล่าวได้อีกนัยหนึ่งว่า JET Engine คือ ไดรเวอร์ชนิดหนึ่ง ซึ่งทำหน้าที่เป็นตัวเชื่อมโยงให้ VB สามารถติดต่อกับฐานข้อมูลได้ทุกชนิดเช่นกัน โดยอาศัยเทคโนโลยีหลายๆ อย่าง

สำหรับฐานข้อมูลที่ผู้เขียนจะนำเสนอ จะใช้ฐานข้อมูลของ MS Access 2000 Thai Edition เป็นหลัก ดังนั้นถ้าคุณสามารถใช้งาน MS Access 2000 ได้บ้างก็จะทำให้การสร้างแอพพลิเคชั่นด้านฐานข้อมูลด้วย Visual Basic รวดเร็วขึ้น โดยที่ผู้เขียนจะกล่าวถึงวิธีการใช้งาน MS Access 2000 เท่าที่เกี่ยวข้อง

การติดต่อฐานข้อมูลใน Access Basic จะแยกออกเป็น 4 ประเภทใหญ่ๆ คือ

แบบที่ 1 ติดต่อโดยอาศัยคอนโทรลด้านฐานข้อมูล

มีศัพท์เรียกคอนโทรลกลุ่มนี้โดยเฉพาะว่า Bound Controls ส่วนใหญ่แล้วก็คือ กลุ่มของคอนโทรลมาตรฐานที่ใช้กันโดยทั่วไปนั่นเอง เช่น คอนโทรล TextBox, Picture Box, Image, ListBox,Combobox เป็นตัวเชื่อมโยงระหว่างฐานข้อมูลกับกลุ่ม Bound Controls

คุณสามารถตรวจสอบได้ว่า คอนโทรลตัวบ้างถูกจัดอยู่ในกลุ่มของ Bound Controls โดยการตรวจสอบว่าคอนโทรลตัวนั้น มีคุณสมบัติที่ขึ้นต้นด้วยคำว่า Data… เช่น DataField, Dataformat, DataSource หรือไม่ ถ้ามีหมายถึง คอนโทรลดังกล่าว ถูกจัดอยู่ในกลุ่มของ Bound Controls ด้วยเช่นกัน

ยังมีคอนโทรลอีกกลุ่มที่เรียกว่า Activex Bound Control หมายถึง กลุ่มของคอนโทรลที่มีคุณสมบัติที่ขึ้นต้นด้วยคำว่า Data… เช่นกัน แต่มีข้อแตกต่างจากคอนโทรลในกลุ่ม Bound Controls ก็คือ กลุ่ม ActiveX Bound Visual Basic โดยที่คุณต้องเพิ่มเติมคอนโทรลกลุ่มนี้เข้ามาในแถบเครื่องมือเอง ดังนั้นจึงเรียกคอนโทรลกลุ่มนี้ว่า ActiveX Bound Controls เช่น คอนโทรล Dblist หรือ คอนโทรล DB Combo ฯลฯ เป็นต้น

แบบที่ 2 ติดต่อโดยใช้ออบเจ็กต์ DATA Access Object (DAO)

ถือว่าเป็นวิธีที่ล้าสมัยแล้ว โดยมีแนวคิดในการติดต่อหรือถึงข้อมูลในฐานข้อมูลผ่านทางองค์ประกอบต่างๆ ในฐานข้อมูล เช่น ฟิลด์ (Field) , เร็คคอร์ด (Record), ความสัมพันธ์ระหว่างตาราง (Relation) เป็นต้น โดยจะแทนแต่ละองค์ประกอบเหล่านั้นด้วยออบเจ็กต์ (Object) และควบคุมออบเจ็กต์ต่างๆ เหล่านี้ด้วยการเขียนโค้ด

แม้จะทำงานได้ดีกว่า อิสระกว่า แต่มีความยุ่งยากในการเขียนโปรแกรมด้วยเช่นกัน อีกทั้งเป็นเทคโนโลยีที่เก่าแล้วคือ เน้นเฉพาะระบบฐานข้อมูลที่เป็นตาราง (โดยเฉพาะ ACCESS รุ่นเก่าๆ) แต่ว่าการเก็บข้อมูลในปัจจุบัน ถูกจัดเก็บอยู่ในสภาพแวดล้อมแตกต่างกันมากมาย เช่น รูปภาพ (Image) , ข้อความ (Text) และรูปแบบอื่นๆ อีกมากมายทำให้ต้องสร้างออบเจ็กต์ใหม่ๆ ขึ้นมาเรื่อยๆ แต่นั่นไม่ใช่สิ่งที่ทำได้ง่าย และกลายเป็นข้อจำกัดที่สำคัญของ DAO

ตัวอย่างของออบเจ็กต์ในกลุ่มนี้ เช่น ออบเจ็กต์ RecordSet, ออบเจ็กต์ TableDef, คอลเล็กชัน Fields เป็นต้น

แบบที่ 3 ติดต่อผ่าน ODBC โดยตรง (ODBC Direct)

เป็นการติดต่อกับฐานข้อมูลแบบ 32 บิต ที่สนับสนุนมาตรฐาน ODBC (Open Database Connectivity) ที่ JET Engine(กลไกการติดต่อบานข้อมูลของไมโครซอฟท์ ซึ่งเน้นที่ Access) ไม่สามารถจัดการได้ เช่น ฐานข้อมูลของ Oracle, ฐานข้อมูลของ Microsoft SQL Server เป็นต้น ซึ่งเป็นการติดต่อเฉพาะฐานข้อมูลที่มีการเก็บข้อมูลในรูปแบบตารางเท่านั้น

ไมโครซอฟท์เองได้สร้างออบเจ็กต์ขึ้นมาอีกชุดหนึ่งที่ชื่อว่า Remote Data Object หรือเรียกสั้นๆ ว่า RDO เพื่อใช้สำหรับติดต่อกับฐานข้อมูลแบบเครือข่าย (เช่น Oracle, SQL Server, DB2, ฯลฯ) ซึ่งเป็นไปตามสถาปัตยกรรม n-Tier (n เทียร์) เช่น Client/Server (2-Tier ทูเทียร์) หรือ Middle Tier (3-Tear ทรีเทียร์) โดยอาศัยมาตรฐาน ODBC ในการเชื่อมโยงที่เกี่ยวกับ Visual Basic คือ

คอนโทรล Remote Data (Remote Data Control – RDC)

ออบเจ็กต์ Remote Data (Remote Data Object – RDO)

แบบที่ 4 – เข้าถึงข้อมูลโดยอาศัยเทคโนโลยี OLEDB

เป็นรูปแบบการติดต่อกับฐานข้อมูลผ่านทางกลุ่มออบเจ็กต์ในโมเดล ADO ซึ่งใช้ OLEDB Provider เป็นกลไกในการเข้าถึงข้อมูลในฐานข้อมูลแทน JET Engine โดยเปลี่ยนจากมุมมองการติดต่อฐานข้อมูลแบบเดิมๆ ที่ต้องกำหนดชนิดของฐานข้อมูลมาเป็นมองที่รูปแบบของการเชื่อมต่อ (Connectivity) เข้ากับฐานข้อมูล

OLEDB เป็นเทคโนโลยีล่าสุดที่ไมโครซอฟต์วางตำแหน่งให้เป็นยุคต่อไป ในการเข้าถึงข้อมูลในฐานข้อมูลที่มีรูปแบบการเก็บข้อมูลสารพัดรูปแบบ ซึ่งไม่ได้จำกัดตัวเองเฉพาะตาราง โดย OLEDB เป็นส่วนหนึ่งของสถาปัตยกรรม Universal Data Access (UDA) ของไมโครซอฟต์

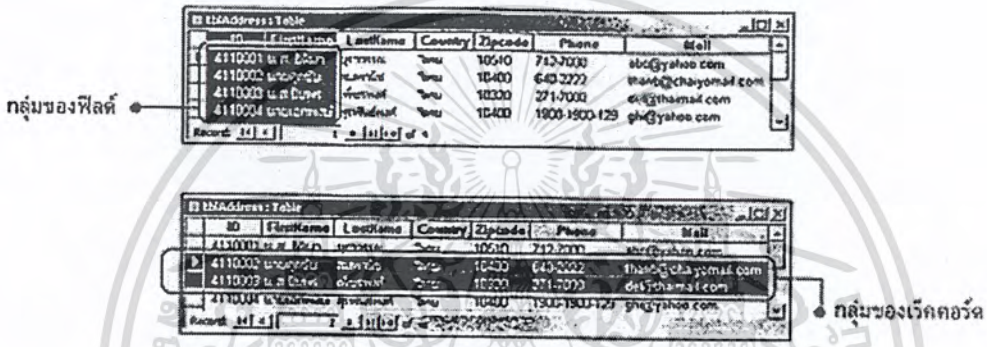
ใน Visual Basic สามารถใช้ OLEDB ได้ 2 แบบ คือ โดยการใช้

- คอนโทรล ADO Data (ADO Data Control) รวมกับกลุ่มของ ActiveX Bound Contols ซึ่งสังเกตจากชื่อของคอนโทรลที่มีคำว่า OLEDB ต่อท้าย เช่น คอนโทรล DataGrid, DataList, Hierachical FlexGrid เป็นต้น
- ชุดออบเจ็กต์ ActiveX Data (ActiveX Data Object – ADO) โดยการเขียนโปรแกรมควบคุม

3.1.1 ทำความรู้จักกับ Relational Database

Relational Database หมายถึง กลุ่มของข้อมูลที่จัดเก็บอยู่ในฐานข้อมูล โดยอาศัยรูปแบบของตาราง (Table) เป็นตัวสร้างความสัมพันธ์ระหว่างข้อมูล โดยที่ถ้ามองข้อมูลในลักษณะแนวนอน (Row) จะถูกเรียกว่า เร็คคอร์ด (Record) แต่ถ้ามองในลักษณะแนวตั้ง (Column) จะถูกเรียกว่า ฟิลด์ (Field) หรือ แอ็ตทริบิวต์ (Attribute)

ตารางดังกล่าว จะมีความสัมพันธ์กัน (Relative) หรือเกี่ยวข้องกัน ในลักษณะที่ข้อมูลที่อยู่ในเร็คคอร์ดของตารางหนึ่งมีความสัมพันธ์ หรือเชื่อมโยงกับ เร็คคอร์ดของอีกตารางหนึ่ง ส่งผลให้ตารางทั้ง 2 ตารางมีความสัมพันธ์กัน และอยู่ร่วมกันเป็นฐานข้อมูลเชิงสัมพันธ์ (Relational Database) โดยมี RDBMS ทำหน้าที่จัดการข้อมูลที่อยู่ในฐานข้อมูลให้มีประสิทธิภาพ



รูปที่ 3.2 แสดงกลุ่มของฟิลด์

ชนิดของความสัมพันธ์ระหว่างตาราง

ความสัมพันธ์ของข้อมูลระหว่างตาราง สามารถแยกออกได้ 3 ชนิด คือ

- One-to-One หมายถึง ข้อมูล 1 เร็คคอร์ดที่อยู่ในตารางหนึ่ง มีความสัมพันธ์หรือเชื่อมโยงกับอีกตารางหนึ่งโดยที่สามารถอ้างอิงได้เพียง 1 เร็คคอร์ดเท่านั้น เช่นนักศึกษา 1 คนมีรหัสประจำตัวได้ 1 หมายเลขในทางกลับกันรหัสประจำตัว 1 หมายเลข สามารถอ้างถึงนักศึกษาได้เพียง 1 คนเท่านั้น
- One-to-Many หมายถึง ข้อมูล 1 เร็คคอร์ดที่อยู่ในตารางหนึ่ง มีความสัมพันธ์หรือเชื่อมโยงกับอีกตารางหนึ่งโดยทีสามารถอ้างาได้มากกว่า 1 เร็คคอร์ด เช่น มหาวิทยาลัยมีนักศึกษาได้หลายคน ในทางกลับกันมีนักศึกษาหลายคนศึกษาอยู่ในมหาวิทยาลัยเดียวกัน
- Many-to-Many หมายถึง ข้อมูลหลายเร็คคอร์ดที่อยู่ในตารางหนึ่ง มีความสัมพันธ์หรือเชื่อมโยงกับอีกตารางหนึ่ง โดยที่สามารถอ้างอิงได้มากกว่า 1 เร็คคอร์ด เช่น นักศึกษา 1 คนลงทะเบียนได้หลายวิชา และแต่ละวิชามีนักศึกษาลงทะเบียนได้หลายคนเช่นกัน

3.1.2 Normalization คืออะไร

Normalization คือ การ จัดระเบียบ , จัดรูปแบบ, จัดโครงสร้าง, ลดความซ้ำซ้อนของข้อมูลที่คุณจัดเก็บในฐานข้อมูล ขั้นตอนหรือทฤษฎีในการทำ Normalization เพื่อให้คุณสามารถออกแบบฐานข้อมูลแล้วนำไปใช้ในการเขียนโปรแกรมด้านฐานข้อมูลได้

การทำ Normalization นั้นมี ความสำคัญ เพราะถ้าคุณออกแบบฐานข้อมูลไม่ดีพอ จะส่งผลให้คุณเขียนโค้ดไม่ได้ หรือไม่ดีเท่าที่ควร รวมถึงความถูกต้องของข้อมูล (Data Integrity) ในฐานข้อมูลด้วย

การทำ Normalization กับฐานข้อมูล เป็นการแบ่งตารางออกเป็นตารางย่อยๆ ตารางที่แบ่งย่อยออกมา จะถูกเรียกว่า Normal Form มีทั้งสิ้น 5 ระดับคือ First Normal (1NF) ไปจนถึง 5 NF ก็แล้วแต่ว่าฐานข้อมูลของคุณมีขอบเขตการใช้งานและขนาดของฐานข้อมูลมากเท่าใด แต่โดยปกติแล้ว จะทำถึงในระดับ 3NF เท่านั้น ส่วนระดับ 4NF และ 5NF จะใช้กับฐานข้อมูลที่มีขนาดใหญ่มากๆ ในระดับองค์กร ที่มีผู้ใช้ฐานข้อมูลมากในเวลาเดียวกัน และเป็นฐานข้อมูลที่มีโครงสร้างการจัดเก็บข้อมูลที่ซับซ้อน

การทำให้ตารางอยู่ใน Normal Form ระดับ 3 NF ถือได้ว่า เป็นระดับที่เหมาะสมในการใช้งาน คุณจะต้องจัดตารางให้อยู่ในระดับ 1NF, 2NF และ 3NF ตามลำดับ หมายความว่า ถ้าตารางของคุณอยู่ในระดับ 2NF ด้วยเช่นกัน

แต่ยังมีการทำ Normalization อีกวิธีหนึ่งซึ่งเรียกว่า BCNF (Boyce-Codd Normal Form) อาจจะเรียกวิธีนี้ว่า วิธีลัดในการทำให้ ตารางอยู่ในระดับ 3NF ก็ได้ โดยที่คุณไม่จำเป็นต้องทำตามลำดับ 1NF ไปจนถึง 3NF ผู้เขียนเห็นว่า ถ้าคุณใช้วิธีการทำ Normalization แบบปกติได้แล้ว การออกแบบฐานข้อมูลของคุณในครั้งต่อๆ ไป คุณจะสามารถลดกระบนการแก้ไขตารางของคุณได้เองโดยอัตโนมัติ กล่าวคือคุณกำลังจะใช้ BCNF ไปโดยปริยาย

การแบ่งตารางย่อยในการทำ Normalization

การแบ่งตารางย่อยออกมา มีจุดประสงค์ก็คือ

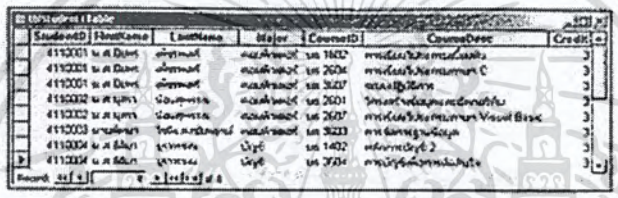
- ควบคุมความถูกต้องของข้อมูล ป้องกันไม่ให้ข้อมูลขัดแย้งกันเอง เช่นตารางข้อมูลลูกค้ากบตารางขายสินค้า สมมติว่า ลูกค้าเปลี่ยนชื่อ-นามสกุล ส่งผลให้ข้อมูลในตารางข้อมูลลูกค้ากับตารางขายสินค้า ต้องเปลี่ยนไปด้วยทั้ง 2 ตาราง ไม่ใช้ข้อมูลที่อยู่ในตารางข้อมูลลูกค้าเปลี่ยน แล้วข้อมูลในตารางขายสินค้าไม่เปลี่ยนทำให้คุณไม่รู้ว่าลูกค้าคนดังกล่าวคือคนๆ เดียวกันนั่นเอง ก็จะทำให้ข้อมูลในฐานข้อมูลของคุณผิดพลาด ซึ่งจะเป็นข้อมูลที่ไม่ถูกต้อง
- ควบคุมและจัดการข้อมูลได้ง่ายกล่าวคือ คุณสามารถที่จะเพิ่มเติม แก้ไขข้อมูลในฐานข้อมูลของคุณ โดยที่มีขั้นตอนเท่าที่จำเป็นเท่านั้น เช่นสมมติว่าบริษัทต้องการเพิ่มเติมรายการสินค้าที่ต้องการขาย ก็ควรที่จะเกี่ยวข้องกับตารางสินค้าเท่านั้น ไม่ควรไปเกี่ยวข้องกับตารางอื่นๆ ในฐานข้อมูล ไม่ใช่ว่าต้องการเพิ่มรายการสินค้ารายการเดียว ต้องไปแก้ไข 5 ตาราง 1 ตาราง ซึ่งไม่ควรเกิดขึ้นเป็นอย่างยิ่ง หรือว่าพนักงานย้ายแผนก ย้ายสังกัดคนเดียว ต้องไปแก้ไขตารางมากมาย เป็นต้น

แต่ถ้าคุณแบ่งตารางออกมามากเกินความจำเป็น มันจะส่งผลให้การเข้าถึงข้อมูลในฐานข้อมูลช้า ดังนั้น

การทำ Normalization จึงควรที่จะอยู่ในระดับ 3NF ซึ่งเป็นระดับที่ยอมรับกันโดยทั่วไป

3.1.3 แนวทางในการปรับปรุงให้ตารางอยู่ในระดับ 3NF (BCNF)

แนวทางในการปรับปรุงให้ตารางของคุณอยู่ในรูป 3 NF โดยวิธี BCNF ผู้เขียนเห็นว่า น่าจะเป็นแนวทางที่ช่วยให้คุณออกแบบฐานข้อมูลของคุณได้เป็นอย่างดี คุณสามารถนำแนวทางนี้ไปใช้งานกับฐานข้อมูลประเภทอื่นๆ ได้เช่นกัน เพราวามาจากทฤษฎีของการทำ Normalization แบบปกติและทฤษฎีของการทำ BCNF ซึ่งจะมีขอบเขตที่ทำให้ตารางอยู่ในระดับ 3NF โดยที่ผู้เขียนจะยกตัวอย่างฐานข้อมูลสมมติขึ้นมาประกอบการอธิบาย เพื่อให้คุณเห็นภาพได้อย่างชัดเจน และให้คุณทำความเข้าใจกับเหตุผลของแต่ละขั้นตอนเป็นสำคัญที่สุด เพราะว่าถ้าคุณนำไปใช้กับฐานข้อมูลประเภทอื่นๆ อาจจะไม่เรียงตามลำดับที่ผู้เขียนนำเสนอ แต่ให้คุณนำเหตุผลที่ได้ไปใช้ในกาปรับปรุงฐานข้อมูลของคุณ ดังนี้



รูปที่ 3.3 แสดงฐานข้อมูลที่ไม่มีการปรังปรุง

จากรูปที่ 3-2 เป็นตารางที่สามารถเก็บข้อมูลทุกอย่างตามที่ต้องการได้ทั้งหมด ไม่ว่าจะเป็นรหัสนักศึกษา ชื่อ-สกุล, สาขาที่เรียน, รหัสวิชา, ชื่อวิชา และจำนวนหน่วยกิจ เป็นตารางที่อยู่ในระดับ 1NF เพราะว่าในทุกๆ ฟิลด์จะเก็บเพียงค่าเดียว ซึ่งเรียกว่า No Repeating Group แต่ว่าตารางดังกล่าวมีข้อเสียดังนี้

1. สมมติ ว่า น.ส.ปิยพร ต้องการลงทะเบียนเรียนเพิ่ม คุณต้องเพิ่มเร็คคอร์ดเข้าไปเท่ากับจำนวนวิชาที่
   น.ส.ปิยพร ลงทะเบียน ถ้ามีนักศึกษาต้องการทำเช่นเดียวกับ น.ส.ปิยพร 100 คน ส่งผลให้คุณต้องเพิ่มเร็คคอร์ดเข้าไปโดยไม่จำเป็น และจะเป็นตารางที่เก็บข้อมูลซ้ำซ้อนมากมาย
2. ถ้า น.ส. ปิยพร ต้องการเปลี่ยนชื่อ หรือเปลี่ยนนามสกุล หรือเปลี่ยนสาขาที่เรียน หรือเปลี่ยนวิชาลงทะเบียนคุณจะต้องแก้ไขทุกๆ เร็คคอร์ดที่เป็นของ น.ศ.ปิยพร
3. ถ้าถามว่า มีวิชาอะไรบ้างที่เปิดสอน ตารางนี้ไม่สามารถบอกคุณได้
4. ถ้าคุณต้องการเปิดสอนวิชาใหม่เพิ่ม คุณไม่สามารถทำได้ เพราะว่าคุณไม่รู้ด้วยซ้ำว่า มีวิชาอะไรบ้างที่มหาวิทยาลัยนี้เปิดสอนอยู่

5. ถ้ามีนักศึกษามาลงทะเบียนเพิ่มขึ้นอีก 50 คน แต่ละคนลงทะเบียนคนละ 7 วิชา จะเกิดความซ้ำซ้อนเพิ่มขึ้นอีก
6. มหาวิทยาลัยนี้เปิดสอนกี่วิชา ตารางนี้ไม่สามารถบอกคุณได้

ขั้นตอนในการปรับปรุงให้ตารางอยู่ในระดับ 3NF (BCNF)

วิธีการแก้ไขให้ตารางนี้ สามารถตอบคำถามข้างต้นนี้ได้คือ

ขั้นแรก : สิ่งแรกที่คุณเห็นอย่างชัดเจนคือ ความซ้ำซ้อนของเร็คคอร์ดจะเพิ่มขึ้นตามจำนวนที่มีการลงทะเบียนเพิ่มเข้ามา ไม่ว่าจะเกิดจากนักศึกษาเดิม หรือนักศึกษาที่จะเข้ามาใหม่ ดังนั้นคุณจึงต้องแยกฟิลด์ CourseID, CourseDesc และ Credit ออกมา เหตุผลคือ

- เป็นสาเหตุของความซ้ำซ้อนที่เกิดขึ้น
- เมื่อคุณต้องการเปิดสอนวิชาใหม่ คุณต้องเพิ่มรหัสนักศึกษาด้วย ซึ่งไม่เกี่ยวข้องกันแต่อย่างใดสำหรับ ฟิลด์ Major คุณอาจจะแยกออกมาหรือไม่ก็ได้ โดยขึ้นอยู่กับว่า
- ต่อไปในอนาคตมีโอกาสที่จะเปิดสาขาขึ้นมาใหม่หรือไม่
- ต้องการดูว่ามหาวิทยาลัยนี้เปิดสอนกี่สาขา
- ถ้านักศึกษาคนใดต้องการย้ายสาขา คุณต้องการแก้ไขในตาราง Student เพียงตารางเดียว

ถ้าคุณต้องการทั้ง 3 อย่างให้แยกฟิลด์ Major ออกมาด้วย ในกรณีที่ผู้เขียนจะแยกฟิลด์Major ออกมา และให้คุณเพิ่มฟิลด์ MajorID เข้าไปในตาราง Student เหตุผลก็คือ เพื่อใช้สำหรับระบุว่า นักศึกษานี้เรียนอยู่สาขาใด การทำในขั้นตอนนี้ เป็นการปรับปรุงให้ตาราง Student เหลือแต่ฟิลด์ที่เกี่ยวข้องกับข้อมูลนักศึกษาเท่านั้นคือ StudentID, FirstName และ Last Name ส่วนฟิลด์ Major ID จะใช้สำหรับอ้างอิงไปยังตาราง Major เพื่อใช้ระบุว่า นักศึกษาเรียนสาขาใด

ให้คุณยึดถือว่า เมื่อคุณแบ่งตารางออกมาแล้ว ไม่ว่าจะเป็นขั้นตอนใดก็ตาม จะต้องสามารถแสดงข้อมูลได้เหมือนเดิม โดยการร้างความสัมพันธ์ระหว่างตารางเดิมกับตารางใหม่ที่แบ่งออกมา

ตาราง Student

StudentID	FirstName	LastName	MajorID
4110001	น.ส.ปิยพร	เพ็ชรพงศ์	01
4110002	น.ส.นุสรา	น้อยสุพรรณ	01
4110003	นายพัทยา	โชติพงศ์ขัพพฤกษ์	01
4110004	น.ส.ธัญมา	ยุราการรณ	02
0			

ตาราง MajorID

MajorID	MajorDesc
01	คอมพิวเตอร์
02	บัญชี
03	การตลาด
04	การเงินและการธนาคาร
05	บริหารทั่วไป

ตาราง TranID

StudentID	CourseID	CourseDesc	Credit
4110001	บธ 1602	การเขียนโปรแกรมเบื้องต้น	3
4110001	บธ 2604	การเขียนโปรแกรมภาษา C	3
4110001	บธ 3607	ระบบปฏิบัติการ	3
4110002	บธ 2601	โครงสร้างข้อมูลและอัลกอริทึม	3
4110002	บธ 2607	การเขียนโปรแกรมภาษา Visual Basic	3
4110003	บธ 3603	การจัดการฐานข้อมูล	3
4110004	บธ 1402	หลักการบัญชี 2	3
4110004	บธ 3604	การบัญชีเพื่อการตัดสินใจ	3
0			0

รูปที่ 3.4 แสดงฟิลด์ที่เหลือในตาราง

ขั้นที่สอง : ในขั้นตอนนี้คุณต้องแก้ไขให้ตาราง TranID แสดงข้อมูลว่า นักศึกษาคนใดลงทะเบียนวิชาใดบ้างซึ่งเป็นข้อมูลของตารางเดิมที่มีอยู่ก่อนการแก้ไข ส่วนในตาราง Student และตาราง major ใช้ได้แล้ว มีเหตุผลคือ

- คุณทราบแล้วว่ามหาวิทยาลัยนี้เปิดสอนกี่สาขา โดยดูจากตาราง Major และถ้าภายหลังคุณต้องการเปิดสาขาที่สอนใหม่ คุณก็มาเพิ่มที่ตาราง Major เพียงตารางเดียว
- ถ้ามีนักศึกษาใหม่เพิ่มเข้ามา คุณก็มาเพิ่มชื่อนักศึกษาที่ตาราง Student เพียงตารางเดียว
- ถ้ามีนักศึกษาต้องการเปลี่ยนชื่อ-นามสกุล เปลี่ยนสาขาวิชา คุณก็สามารถแก้ไขที่ตาราง Student เพียงตารางเดียวเช่นกัน

หลังจากที่แก้ไขแล้ว คุณยังคงทราบเหมือนเดิมว่านักศึกษาคนไหนเรียนอยู่สาขาใด โดยระบุค่า Major ID ให้ตรงกัน รวมทั้งเป็นการแก้ไขความซ้ำซ้อนของตาราง Student เดิมด้วยเช่นกัน

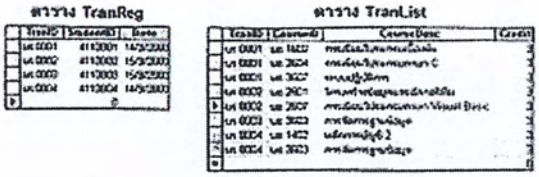
ส่วนตาง TranID จะมีข้อเสียดังนี้

- ข้อมูลที่อยู่ในแต่ละ เร็คคอร์ด จะมีค่าซ้ำซ้อนเท่ากับจำนวนนักศึกษาลงทะเบียน ซึ่งเหมือนกับตารางเดิมในตอนแรก และถ้าในภายหลังนักศึกษาต้องการลงทะเบียนเพิ่มทุกคน ก็จะเกิดความ
  ซ้ำซ้อนมากขึ้นเข้าไปอีก
- ถ้าคุณต้องเปิดสอนวิชาใหม่ คุณไม่สามารถเพิ่มได้เพราะว่า เมื่อคุณเพิ่มเข้ามา 10 วิชา ก็จะไปเพิ่มรหัสนักศึกษา 10 หมายเลข ซึ่งไม่ได้เกี่ยวข้องกันแต่อย่างใด ดังนั้นฟิลด์ StudentID กับฟิลด์ CourseID ไม่ควรอยู่ด้วยกัน
- สมมติว่า น.ส.ปิยพร ลาออก คุณก็ต้องลบรหัส 4110001 ออกที่ตาราง Student และตาราง TranID ส่งผลให้วิชาที่ น.ส.ปิยพร ลงทะเบียนหายไปด้วย ซึ่งไม่ได้เกี่ยวข้องกันแต่อย่างใด เพราะว่านักศึกษาลาออกแต่มหาวิทยาลัยก็ยังเปิดสอนวิชานั้นอยู่ ดังนั้นคุณต้องแยกรหัสวิชากับชื่อวิชาออกมาอีก 1 ตาราง
- ถ้าคุณต้องการตรวจสอบว่า นักศึกษาคนใดลงทะเบียนวันไหนกับใบลงทะเบียนหมายเลขใด คุณจะไม่สามารถตรวจสอบได้เลย ข้อสังเกตอันนี้เป็นการเพิ่มเติมความสมบูรณ์ของข้อมูล

จากข้อเสียของตาราง TranID จะเอามาเป็นโจทย์ในการแก้ไข โดยเพิ่มฟิลด์ TranID เพื่อใช้สำหรับเก็บหมายเลขใบลงทะเบียนของนักศึกษาแต่ละคน โดยมีฟิลด์วันที่เพิ่มเข้ามาเพื่อความสมบูรณ์ของข้อมูล

ขั้นที่สาม ขั้นต่อไปพิจารณาว่า ในใบลงทะเบียนควรจะมีอะไรบ้างซึ่งก็น่าจะมีหมายเลขทะเบียน, รหัสนักศึกษา, ชื่อนักศึกษา, รหัสวิชา, ชื่อวิชา และหน่วยกิตของแต่ละวิชา

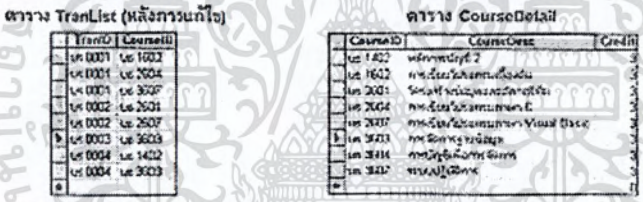
แต่คุณทราบมาแล้วว่า รหัสวิชากับชื่อวิชาจะอยู่รวมกับรหัสนักศึกษาไม่ได้ จากเหตุผลดังกล่าว คุณจึงจำเป็นต้องใช้หมายเลขใบลงทะเบียน (ฟิลด์ TranID) ระบุในตารางแรกว่า นักศึกษารหัสใดลงทะเบียน ส่วนในตารางที่ 2 ระบุว่า ลงทะเบียนรหัสวิชาใด ชื่อวิชาอะไร และจำนวนหน่วยกิตดังนี้



รูปที่ 3.5 แสดงฟิลด์ TranID ใช้สร้างความสัมพันธ์ระหว่างสองตาราง

เหตุผลที่ใช้หมายเลขใบลงทะเบียน (ฟิลด์ TranID) เพื่อบอกว่า นักศึกษารหัสนี้ลงทะเบียนกับใบลงทะเบียนหมายเลขใด และนักศึกษาคนเดียวกันนี้ลงทะเบียนวิชาอะไรบาง ตอนนี้แก้ปัญหาให้ฟิลด์รหัสนักศึกษากับรหัสวิชา + ชื่อวิชา แยกกันได้แล้ว ในขั้นนี้ตาราง TranReg ใช้ได้แล้ว

ส่วนในตาราง TranList จะมีปัญหาใหม่คือ เมื่อคุณเพิ่มวิชาใหม่เข้ามา แล้วมันจะไปเพิ่มหมายเลขใบลงทะเบียนซึ่งไม่เกี่ยวข้องกันแต่อย่างใด ก็ใช้วิธีแก้ไขที่เหมือนกับกรณีเปิดสอนวิชาใหม่แล้วไปเพิ่มรหัสนักศึกษาให้แยกออกเป็นตาราง TranList และ CourseDetail จะได้ผลดังนี้

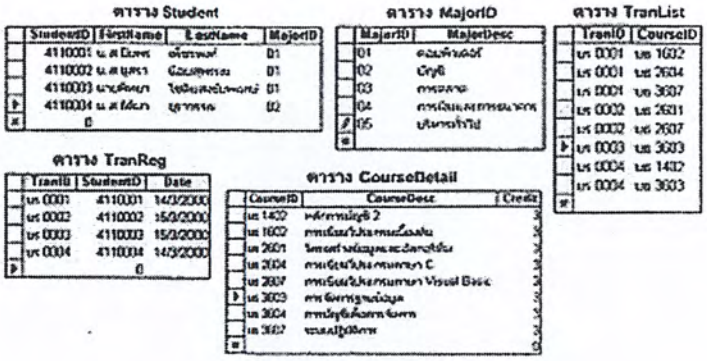


รูปที่ 3.6 ใช้ฟิลด์รหัสวิชาเป็นตัวเชื่อมสองตาราง

ผลจากการแก้ไขตาราง TranList จะทำให้

- เมื่อยกเลิกใบลงทะเบียน วิชายังคงเปิดสอนอยู่ แต่หมายเลขใบลงทะเบียนหายไป โดยการลบในตาราง TranReg และตาราง Translist
- สามารถเพิ่มวิชาใหม่เข้าไปได้ที่ตาราง Course Detail เพียงตารางเดียว
- ถ้ามีนักศึกษายกเลิกการลงทะเบียนวิชาใดวิชาหนึ่ง แก้ไขที่ตาราง Tranlist ตารางเดียว

จากขั้นตอนการแก้ไขที่ผ่านมา จะได้ตารางทั้งหมด 5 ตารางคือ



รูปที่ 3.7 ตารางที่ได้แก้ไขปัญหาในแต่ละขั้นตอน

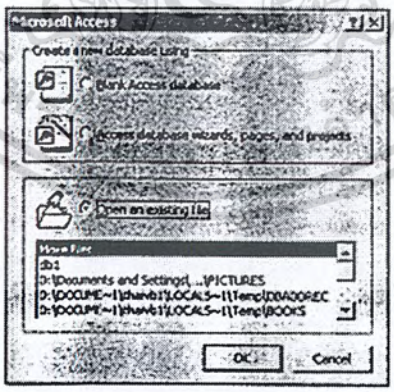
สำหรับคุณผู้อ่านที่ต้องการศึกษาเทคนิคการเพิ่มฟิลด์ต่างๆ เข้ามาแก้ปัญหาที่เกิดขึ้นระหว่างการทำ Normalization คุณสามารถศึกษาได้จากตำราด้านการออกแบบแบบฐานข้อมูล หรือตำราที่เกี่ยวกับการทำ Normalization โดยเฉพาะ จะมีรายละเอียดที่สมบูรณ์ และอธิบายเทคนิคต่างๆ ได้ดีกว่าผู้เขียน

ในหัวข้อนี้ผู้เขียนหวังว่า จะช่วยเป็นแนวทางให้คุณออกแบบฐานข้อมูลของคุณได้บ้าง ซึ่งจะทำให้คุณเขียนโค้ด จัดการข้อมูลในฐานข้อมูลได้ง่ายขึ้น

3.2 การสร้างฐานข้อมูลใน Access 2000

หัวข้อนี้เป็นวิธีการสร้างฐานข้อมูลด้วย Access 2000 Thai Edition ผู้เขียนจะใช้เป็นฐานข้อมูลหลัก สำหรับการเขียนโปรแกรมด้านฐานข้อมูล มีขั้นตอนดังนี้

1. ที่ DeskTop ของ Window 2000 ให้เลือก Start>Programs>Microsoft Access ก็จะได้พบกับไดอะล็อกบ็อกซ์ ดังรูป

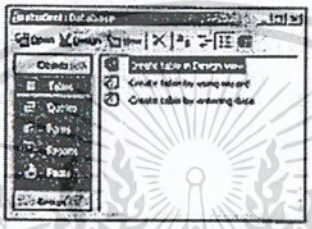


รูปที่ 3.8 ไดอะล็อกบ็อกซ์ เริ่มต้นใช้งาน Access 2000

2. รายละเอียดในไดอะล็อกบ็อกซ์ มีดังนี้
    - ถ้าต้องการสร้างฐานข้อมูลใหม่ให้เลือก Blank Access Database
    - ถ้าต้องการใช้งาน Wizard ช่วยสร้างฐานข้อมูลให้เลือก Access Database wizards, Pages and projects
    - ถ้าต้องการเปิดฐานข้อมูลเดิมที่มีอยู่แล้วให้เลือก Open an existing File

ในกรณีที่ต้องการสร้างฐานข้อมูลใหม่ เลือกคำสั่ง Blank Access database คลิก OK จากนั้นให้เลือก Folder ที่คุณต้องการเก็บฐานข้อมูลที่จะสร้างขึ้นมา ผู้เขียนจะเก็บไว้ที่โฟลเดอร์ MyDocuments จากนั้นคลิก Create

3. หลักจากนั้นจะเข้าสู่ไดอะล็อกบอกซ์ที่ใช้สร้างตารางในฐานข้อมูล ดังรูปที่ 3.9

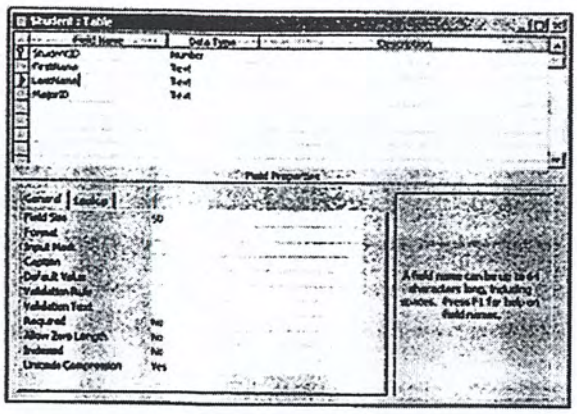


รูปที่ 3.9 ไดอะล็อกบ็อกซ์ สำหรับตาราง

แต่ละหัวข้อมีรายละเอียดดังนี้

- Create Table in Design View หมายถึง เราจะกำหนดรายละเอียดต่างๆ ในตารางเอง เช่น ชื่อ, ฟิลด์, ชนิดของข้อมูลในแต่ละฟิลด์ รวมถึงข้อกำหนดอื่นๆ ของแต่ละฟิลด์
- Create table by using wizard หมายถึง ต้องการสร้างตาราง โดยใช้ Wizard ช่วยในการสร้างตาราง
- Create table by entering data หมายถึง ต้องการสร้างตาราง โดยการใส่ข้อมูลไปด้วย และแก้ไขฟิลด์ไปด้วยในกรณีนี้ ต้องการกำหนดรายละเอียดตารางเอง ให้ดับเบิ้ลคลิกเลือกคำสั่ง Create table in Design view

4. ให้กำหนดชื่อฟิลด์ ในช่อง Field Name กำหนดชนิดของข้อมูลในช่อง Data Type



รูปที่ 3.10 แสดงการกำหนดชื่อในแต่ละฟิลด์

ถ้าต้องการกำหนดเงื่อนไขให้กับแต่ละฟิลด์ มีรายละเอียดดังนี้

- Required หมายถึง กำหนดว่าให้ผู้ใช้ต้องใส่ข้อมูลให้ฟิลด์นั้นด้วยหรือไม่
- Allow Zero Length หมายถึง ยอมให้ฟิลด์นั้นเป็นฟิลด์ว่างได้หรือไม่
- Indexed หมายถึง ต้องการสร้างดัชนี (Index) ให้กับตาราง

คลิกเลือกไอคอน Key เพื่อสร้าง Primary Key

5. ให้บันทึก และตั้งชื่อตารางให้ดับเบิ้ลคลิกที่ตารางเพื่อเพิ่ม หรือแก้ไขข้อมูลในตาราง

3.2.1 ชนิดของข้อมูลในฐานข้อมูล Access 2000

สำหรับข้อมูลชนิดต่างๆ ที่สามารถเก็บในฐานข้อมูลของ Access 2000 มีดังนี้

ชนิดของข้อมูล	พื้นที่ที่ใช้เก็บ	ขอบเขตข้อมูล
Auto Number	ขึ้นอยู่กับข้อมูล	เป็นตัวเลขที่เพิ่มขึ้นโดยอัตโนมัติ เมื่อมีการเพิ่มเร็คคอร์ดเข้ามาจะมี 2 ลักษณะคือ • Increment หมายถึง เพิ่มค่าขึ้นโดยอัตโนมัติครั้งละ 1 ตัว เลขที่จะไม่ซ้ำกัน (Default) • Random หมายถึง สุ่มตัวเลขขึ้นมา ค่าตัวเลขที่ได้จะซ้ำหรือไม่ ขึ้นอยู่กับการกำหนด Indexed
True/False	2 Bytes	เก็บค่าทางตรรกะ เช่น ใช่-ไม่ใช่, จริง-เท็จ เป็นต้น Yes/No, On/Off
Byte	1 Byte	เก็บค่าตัวเลขจำนวนตั้งแต่ 0-255 เป็นรหัสแอสกี ASCII
Currency	8 Byte	ใช้เก็บค่าตัวเลขจำนวนจริง มีค่าระหว่าง –

		922,337, 203,685,477.5808 ถึง 922,337 ,203,685,477.5807 ใช้สำหรับเก็บตัวเลขทางการเงินโดยเฉพาะ เพราะมีความละเอียดสูง มีทศนิยม 4 ตำแหน่ง
Hyperlink	ขึ้นอยู่กับข้อมูลที่เก็บ	ใช้เก็บ URL
Integer	2 Bytes	เก็บค่าตัวเลขจำนวนเต็มที่มีค่าระหว่าง –2,147,483,648 ถึง2,147,483,647
Long	4 Bytes	ใช้เก็บตัวเลขจำนวนเต็มที่มีค่าระหว่าง –2, 147, 483, 648 ถึง 2,147,483,647
Memo	ขึ้นอยู่กับข้อมูลที่เก็บ	เก็บข้อมูลชนิดตัวอักษรเช่นเดียวกับแบบ Text แต่สามารถเก็บตัวอักษรได้ 1.2 GB
ชนิดของข้อมูล	พื้นที่ที่ใช้เก็บ	ขอบเขตข้อมูล
Double	8 byte	ใช้เก็บตัวเลขจำนวนจริง แยกเป็น 2 กรณีคือ ค่าบวกอยู่ระหว่าง 4.9465645841247e-324 ถึง 1.79769313486232e38 ค่าลบอยู่ระหว่าง –1.79769313486232E308 ถึง –4.94065645841247E-324
Text	1 ตัว/1 ไบต์ (ASCII)	ใช้เก็บตัวอักษร, ข้อความ และตัวเลขไม่เกิน 255 ตัวอักษร
Date/Time	8 byte	ใช้สำหรับเก็บวันเดือนปี หรือเวลา
Decimal	12 Bytes	กรณีตัวเลขจำนวนเต็มมีค่าตั้งแต่ –79,228,162,14,264,337,593,543,950,335 ถึง +79,228,162,514,264,337,593,543,950,335 กรณีเลขทศนิยม มีค่าตั้งแต่ –7.9228162514264337593543950335 ถึง +7.9228162614264337593543950335 ค่าที่น้อยที่สุดไม่เท่ากับศูนย์ที่สามารถเก็บได้คือ 0.0000000000000000000000000000000001

OLE Object	ขึ้นอยู่กับออบเจ็กต์	ใช้สำหรับเก็บข้อมูลที่นอกเหนือจากตัวอักษร เช่น ไฟล์ภาพ, ที่เก็บเสียง หรือสื่อมัลติมีเดียอื่นๆที่ OLE สนับสนุน โดยมีขนาดไม่เกิน 1.2 GB

3.3.2 แนวความคิดในการกำหนด Primary Key

ในระหว่างการออกแบบฐานข้อมูลสิ่งหนึ่งที่คุณต้องคำนึงถึงก็คือ การกำหนด Primary Key ให้กับตารางต่างๆ ในฐานข้อมูล เพื่อกำหนดให้ฟิลด์ใดฟิลด์หนึ่งเป็นฟิลด์ที่ใช้บอกความแตกต่างระหว่างเร็คคอร์ด ซึ่งถือได้ว่า คุณจะต้องให้ความสำคัญเป็นอย่างยิ่ง หัวข้อนี้เป็นแนวความคิดกว้างๆ สำหรับการกำหนด Primary Key ให้คุณพิจารณาตารางข้างล่างต่อไปนี้ เพื่อประกอบการอธิบาย

ID	FirstName	LastName	Salary	Address
001	นายอดิเทพ	ตรังคานนท์	10000	ตรัง
002	นายศุภชัย	สมพานิช	10000	กรุงเทพ
003	น.ส.ชลธิชา	สีดา	12000	กรุงเทพ
004	น.ส.ปิยพร	เพ็ชรพงศ์	15000	กรุงเทพ

จากตาราง ผู้เขียนสมมติให้เป็นตารางในฐานข้อมูลใดๆ โดยมีทั้งสิ้น 4 ฟิลด์คือ ID, FirstName, LastName และ ADDRESS การกำหนดให้ฟิลด์ใดฟิลด์หนึ่งใน 4 ฟิลด์นี้เป็น Primary Key สิ่งที่คุณจะต้องระวังไว้ให้ดี ในกรณีที่ข้อมูลในฟิลด์นั้นๆ มีโอกาสซ้ำกันได้หรือไม่ หรือว่ามีโอกาสไม่มีค่า (Null) ได้หรือไม่ เพราะว่าฟิลด์ทำหน้าที่เป็น Primary Key จะต้องมีคุณสมบัติอ่างน้อยที่สุด 2 ข้อคือ

1. จะต้องมีค่าไม่ซ้ำกันหรือไม่เหมือนกัน (Unique Value) เสมอ ไม่ว่าตารางนั้นจะมีกี่เร็คคอร์ดก็ตาม
2. จะต้องมีค่าใดค่าหนึ่งอยู่เสมอ ห้ามเป็นค่าว่าง (Null) หรือไม่มีค่า

ยกตัวอย่างเช่น

กรณีแรก : สมมติว่า กำหนดให้ฟิลด์ที่ซึ่งว่า FirstName เป็น Primary Key จะเห็นได้ว่า โอกาสที่ตารางนี้ จะเก็บชื่อพนักงานที่มีชื่อเหมือนกันมีโอกาสเป็นไปได้ ดังนั้นกรณีนี้ตกไป

กรณีที่ 2 : สมมติให้ฟิลด์ ที่ชื่อว่า LastName เป็น Primary Key สำหรับกรณีนี้ ที่จะเห็นได้ชัดที่สุดก็คือ ถ้าพนักงานเป็นญาติ, เป็นพี่น้อง หรือเป็นสามีภรรยากัน แน่นนอนว่าจะต้องมีนามสกุลเหมือนกัน ดังนั้นกรณีที่ 2 ใช้ไม่ได้เช่นกัน

กรณีที่ 3 : สมมติให้ฟิลด์ ที่ชื่อว่า Salary เป็นPrimary Key กรณีนี้มีโอกาสที่เป็นไปได้สูง ที่พนักงานระดับ ปฏิบัติการในบริษัทใหญ่ๆ หรือโรงงานอุตสาหกรรม ซึ่งมีการจ้างแรงงานเป็นรายวัน จะมีค่าแรงหรือเงินเดือนเท่ากันดังนั้นกรณีนี้ใช้ไม่ได้

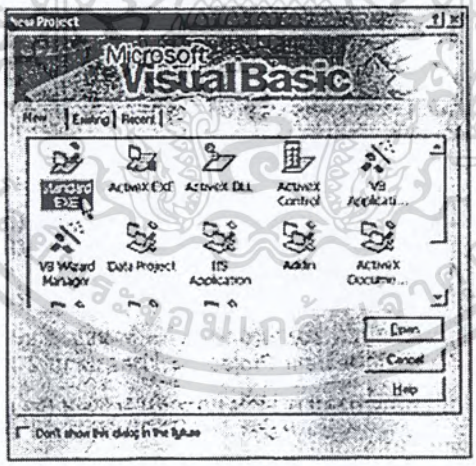
กรณีที่ 4 : สมมติให้ฟิลด์ ที่ชื่อว่า Address เป็น Primary key ในคามเป็นจริงแล้วจะมีการเก็บบ้านเลขที่, ถนน, ตรอก, ซอย, อำเภอ, จังหวัด ฯลฯ โอกาสที่จะมีที่อยู่ที่เดียวกันนั้นยังคงมี กล่าวคือในกรณีที่พนักงานพักอยู่ด้วยกันจึงทำให้กรณีนี้ตกไป

กรณีที่ 5 : สมมติให้ฟิลด์ ที่ชื่อว่า ID เป็น Primary Key กรณีนี้เราจะใช้เป็น Primary Key ได้เป็นอย่างดีเพราะว่า เรามักที่จะออกแบบให้ฟิลด์ลักษณะนี้ เป็น Primary Key ซึ่งจะมีการเพิ่มค่าตัวเลขขึ้นมาโดยอัตโนมัติ ขณะที่มีการเพิ่มข้อมูลเข้ามา ทำให้ค่าตัวเลขที่อยู่ในฟิลด์นี้ไม่มีโอกาสที่จะซ้ำกันได้เลยและจะมีค่าเท่ากันเสมอ

ในกรณีที่คุณพบปัญหาว่า เมื่อคุณตรวจสอบทุกๆ ฟิลด์แล้ว ยังไม่สามารถกำหนด Primary Key ได้ คุณอาจใช้ฟิลด์มากกว่า 1 ฟิลด์ เป็น Primary Key ก็ได้ ซึ่งก็จะทำให้โอกาสที่แต่ละเร็คคอร์ดจะมีค่าซ้ำกันลดน้อยลง

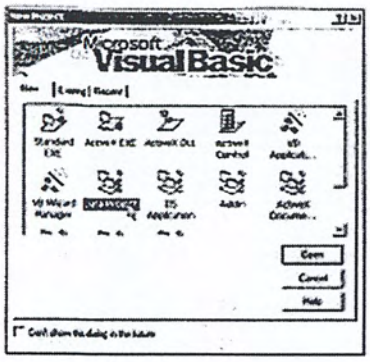
3.3 การพัฒนาแอพพลิเคชั่น ด้านฐานข้อมูลภายใต้สภาพแวดล้อม Data Project

โดยปกติแล้ว เราจะใช้ VBIDE ชนิด Standard EXE เป็นสภาพแวดล้อมในการทำงาน ดังรูปที่ 3.11



รูปที่ 3.11 VBIDE ชนิด มาตรฐาน EXE

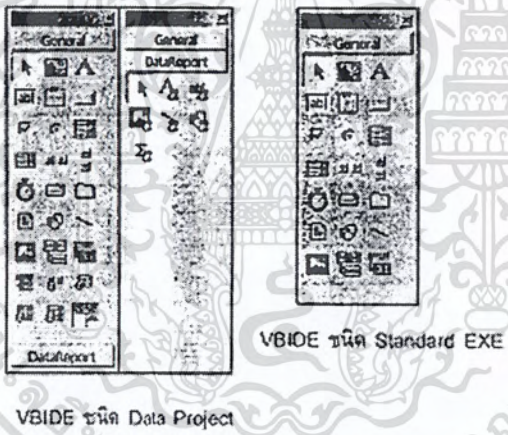
แต่ในกรณีที่คุณต้องพัฒนาแอพพลิเคชันด้านฐานข้อมูลโดยอาศัยเทคโนโลยี OLEDB แล้ว ผู้เขียนแนะนำให้คุณเลือก VBIDE ชนิด Data Project ซึ่งจะเป็นสภาพแวดล้อมที่เหมาะสมที่สุดในการใช้งาน ADO



รูปที่ 3.12 VBIDE ชนิด DATA Project

3.3.1 รู้จักกับ VBIDE ชนิด Data Project

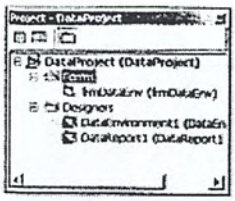
ใน VBIDE ชนิด Data Project ที่แถบเครื่องมือ จะมีกลุ่มคอนโทรล ActiveX Bound Controls ที่ใช้เทคโนโลยี OLEDB ถูกเพิ่มเข้ามาโดยอัตโนมัติ เมื่อเทียบ กับ VBIDE ชนิด Standard EXE



รูปที่ 3.13 แถบเครื่องมือ VBIDE ทั้ง 2 ชนิด

จากรูปที่ 3.13 แถบเครื่องมือของ VBIDE ชนิด Data Project จะประกอบด้วย 2 แท็บ คือ

- แท็บ General ประกอบไปด้วย กลุ่มคอนโทรลมาตรฐาน และกลุ่ม ActiveX Bound Controls ที่ใช้ OLEDB เป็นกลไกในการเข้าถึงข้อมูลในฐานข้อมูล
- แท็บ DataReport ประกอบไปด้วย กลุ่มคอนโทรลที่ใช้สำหรับทำแบบฟอร์มรายงานสิ่งที่แตกต่าง อีกอย่างหนึ่งคือ ในหน้าต่าง Project Explorer จะเพิ่มเครื่องมือขึ้นมาอีก 2 อย่างคือ

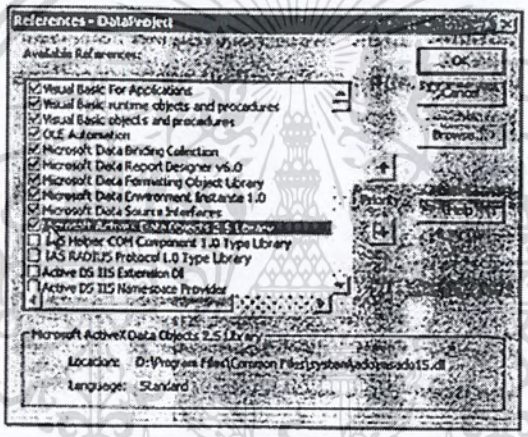


รูปที่ 3.14 Project Explorer ของ VBIDE ชนิด Data Project

- DataEnvironment เป็นเครื่องมือที่อำนวยความสะดวกให้คุณใช้ ADO ได้ง่ายขึ้น
- DataReport ใช้สำหรับสร้างแบบฟอร์มรายงาน

3.3.2 การตรวจสอบเวอร์ชันของ ADO ที่ใช้งาน

โดยปกติแล้ว เมื่อคุณต้องการการใช้งานออบเจ็กต์ ADO ใน VBIDE แบบ Standard Exe คุณต้องเพิ่มเข้ามาโดยการเลือกเมนู Project>Reference… แล้วคลิกเลือกตัวเลือก Microsoft ActiveX Data Object Libaray 2.0 แล้วเปลี่ยนเป็น Microsoft ActiveX Data Object Library 2.5 แทน ดังรูปที่ 3.15

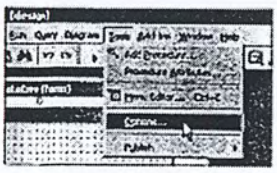


รูปที่ 3.15 การเลือกใช้ ADO 2.5

3.3.3 เทคนิคการแสดงผลภาษาไทยใน Editor ของ VB

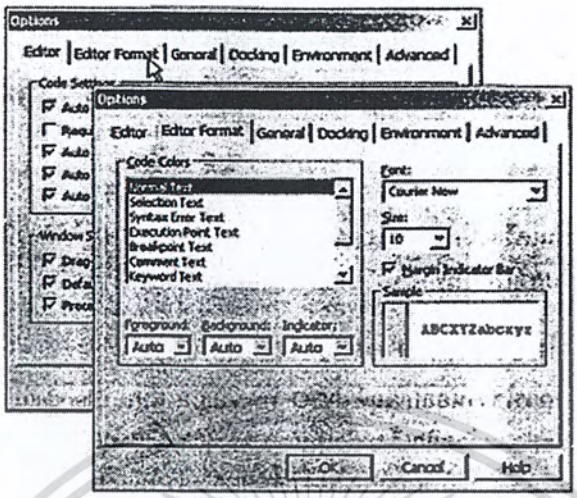
โดยปกติแล้ว Editor ของ VB จะไม่สามารถแสดงผลภาษาไทยได้ ทำให้เราไม่สามารเขียนหมายเหตุ หรือข้อความภาษาไทยได้ Editor จะแสดงเป็นตัวอักษรที่อ่านไม่รู้เรื่อง แต่เราสามารถกำหนดให้ Editor แสดงผลภาษาไทยได้ มีขั้นตอนดังนี้

1. ให้คุณเลือกคำสั่ง Tools>Options… ดังรูปที่3.16



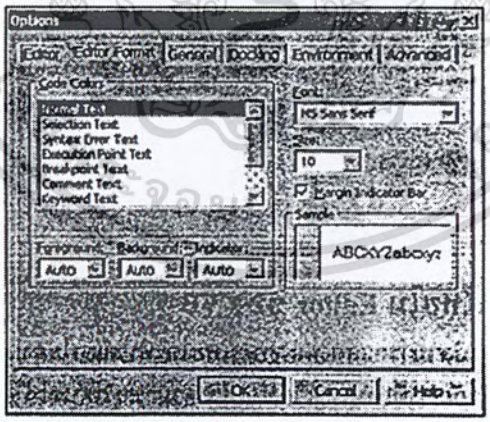
รูปที่ 3.16 แสดงการใช้คำสั่ง

2. คุณจะได้พบกับไดอะล็อกบ็อกซ์ Options ให้คลิกเลือกที่แท็บ Editor Format ดังรูปที่ 3.9

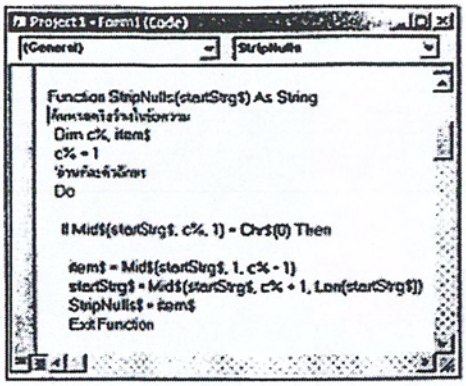


รูปที่ 3.17 แสดงบ็อกซ์ Option

ในรายการ Normal Text ให้คุณเปลี่ยนชื่อฟอนต์ในช่อง Font เป็น MS Sans Serif ขนาด 10 หรือฟอนต์ที่ลงท้ายด้วย UPC ขนาด 14 ก็ได้ ผู้เขียนแนะนำให้เลือก MS Sans Serif ขนาด 10 จะมีลักษณะใกล้เคียงกับแบบปกติดังรูปที่ 3.18 จากนั้นคลิกที่ OK



รูปที่ 3.18 กำหนดให้ editor ใช้ ฟอนต์ MS Sans Serif ขนาด 10



รูปที่ 3.19 รูปแบบของ ฟอนต์ MS Sans Serif ขนาด 10

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

บทที่ 4
การใช้งานภาษา SQL

บทนำ

SQL ย่อมาจากคำว่า Structure Query Language หมายถึง ภาษามาตรฐานกลางที่ใช้สำหรับจัดการข้อมูลในฐานข้อมูล โดยเฉพาะอย่างยิ่งฐานข้อมูลประเภท RDBMS (Relation Database Management System) จะรู้จักภาษา SQL เป็นอย่างดี เราจะใช้ SQL เพื่อจัดการกับข้อมูลในฐานข้อมูลได้หลายอย่าง เช่น การแสดงข้อมูลจากฐานข้อมูลแบบมีเงื่อนไข, การเพิ่ม, การลบ และการนำข้อมูลจากหลายๆตารางมาแสดงร่วมกันได้ เป็นต้น เป็นภาษาที่ใช้จัดการข้อมูลในฐานข้อมูลได้มีประสิทธิภาพมากที่สุด

เราจะใช้ภาษา SQL เพื่อทำคิวรี (Query) ข้อมูลที่อยู่ในตาราง ในวัตถุประสงค์ที่ต่างๆกัน เช่น อาจจะต้องการข้อมูลที่มาจากตารางเดียว หรือหลายตารางมาแสดงด้วยกันในเวลาเดียวกัน ดังนั้นการทำคิวรีจึงเป็นการสร้างตารางเสมือนขึ้นมา ประกอบด้วยข้อมูลที่มาจากตารางเดียวหรือหลายตารางก็ได้ เป็นตารางที่ไม่มีอยู่จริงในฐานข้อมูล เป็นมุมมองของข้อมูลในฐานข้อมูลตามที่คุณต้องการ

4.1 โครงสร้างของภาษา SQL

ภาษา SQL ประกอบไปด้วย 3 ส่วนใหญ่ๆ คือ

1. Data Definition Language (DDL) เป็นกลุ่มคำสั่งในภาษา SQL ที่ใช้สำหรับจัดการโครงสร้างของฐานข้อมูล เช่น การสร้างฐานข้อมูล, ปรับปรุงโครงสร้างของฐานข้อมูล เป็นต้น ตัวอย่างการใช้งานกลุ่มคำสั่ง DDL นี้ก็คือ การสร้างฐานข้อมูลด้วย MS SQL Server 7.0 ก็จะมีการใช้งานคำสั่งในกลุ่ม DDL เป็นหลัก
2. Data Manipulation Language (DML) เป็นกลุ่มคำสั่งในภาษา SQL ที่ใช้สำหรับจัดการข้อมูลในฐานข้อมูล เช่น การแสดงข้อมูลแบบมีเงื่อนไข, การลบข้อมูล, การเพิ่มข้อมูล และการแสดงข้อมูลที่มาจากตารางหลายตาราง เป็นต้น
3. กลุ่มฟังก์ชั่น Aggregate Function เป็นฟังก์ชั่นพิเศษของภาษา SQL ที่ทำหน้าที่เฉพาะอย่าง เช่น หาผลรวมเร็คคอร์ด, ค่าสูงสุด, ค่าต่ำสุด เป็นต้น เป็นกลุ่มฟังก์ชั่นที่มีประโยชน์มาก เพราะจะช่วยลดภาระให้คุณไม่ต้องเขียนโค้ดจัดการเอง

ในบทนี้ ผู้เขียนจะแบ่งออกเป็น 2 ส่วนใหญ่ๆ คือ ในส่วนแรกจะเป็นการศึกษาทฤษฎี และไวยากรณ์ในการใช้งานคำสั่งของ SQL ก่อน ในส่วนที่ 2 จะเป็นการนำ SQL ไปใช้งานร่วมกับ Visual Basic ในการจัดการข้อมูลในด้านต่างๆ

สำหรับการใช้งานภาษา SQL ร่วมกับ Visual Basic เพื่อจัดการข้อมูลในฐานข้อมูล ผู้เขียนจะใช้งานกลุ่มคำสั่ง DML เป็นหลัก โดยที่ในบทนี้จะมีขอบเขตเนื้อหา คือ อธิบายการใช้งานกลุ่มคำสั่ง DML ร่วมกับกลุ่มฟังก์ชั่น Aggregate และกำหนดเงื่อนไขโดยใช้ตัวดำเนินการด้านต่างๆ

คำสั่งในกลุ่ม DML จะมีคำสั่งพื้นฐานอยู่ 4 คำสั่ง คือ

- DELETE เป็นคำสั่งที่ใช้สำหรับลบข้อมูลหรือลบเร็คคอร์ดใดๆในตาราง
- INSERT เป็นคำสั่งที่ใช้สำหรับเพิ่มข้อมูลหรือเพิ่มเร็คคอร์ดใดๆเข้าไปในตาราง
- SELECT เป็นคำสั่งที่ใช้สำหรับเลือกข้อมูลหรือแสดงเร็คคอร์ดใดๆที่ต้องการจากตาราง อาจจะมาจากตารางเดียว หรือหลายตารางก็ได้
- UPDATE ใช้สำหรับการแก้ไขข้อมูลหรือแก้ไขเร็คคอร์ดใดๆในตาราง

4.1.1 ตัวดำเนินการ (Operator)

ตัวดำเนินการเปรียบเทียบที่น่าสนใจ ได้แก่

ตัวดำเนินการ	ความหมาย
=	เท่ากับ (Equal)
<>	ไม่เท่ากับ (Not Equal)
<	น้อยกว่า (Less Than)
>	มากกว่า (Greater Than)
<=	น้อยกว่าหรือเท่ากับ (Less Than or Equal To)
>=	มากกว่าหรือเท่ากับ (Greater Than or Equal To)
Like	เป็นการเปรียบเทียบโดยใช้ตัวอักษรพิเศษ

ตัวดำเนินการด้านตรรกะ (**Logical Operator**)

ตัวดำเนินการด้านตรรกะที่นิยมใช้จะมีอยู่ 3 ชนิด คือ And, Or, และ Not

4.1.2 กลุ่มฟังก์ชั่น Aggregate

กลุ่มฟังก์ชั่น Aggregate เป็น ฟังก์ชั่นที่ใช้คำนวณทางคณิตศาสตร์ โดยช่วยให้การนำเสนอผลการค้นหาเป็นไปอย่างมีประสิทธิภาพ ซึ่งมีฟังก์ชั่นพื้นฐานต่อไปนี้

ชื่อฟังก์ชัน	หน้าที่
AVG ( )	หาค่าเฉลี่ยของฟิลด์ทั้งหมด จากเร็คคอร์ดทั้งหมด
COUNT ( )	นับจำนวนเร็คคอร์ด
FIRST ( )	หาค่าแรกในฟิลด์
LAST ( )	หาค่าสุดท้ายในฟิลด์
MAX ( )	หาค่ามากที่สุด หรือ ค่าสูงสุด
MIN ( )	หาค่าน้อยที่สุด หรือค่าต่ำสุด
SUM ( )	หาผลรวมทั้งหมดของฟิลด์

4.2 ลักษณะการใช้งานของกลุ่มคำสั่ง DML

รายละเอียดของการใช้งานกลุ่มคำสั่ง DML ทั้ง 4 คำสั่ง มีดังนี้

4.2.1 คำสั่ง DELETE

เป็นคำสั่งที่ใช้สำหรับลบข้อมูล หรือเร็คคอร์ดใดๆออกจากตาราง มีรูปแบบการใช้งาน 2 ลักษณะใหญ่ คือ

```
รูปแบบที่ 1
DELETE FROM ชื่อตาราง WHERE เงื่อนไข
```

หรือ

```
รูปแบบที่ 2
DELETE* FROM ชื่อตาราง
```

ชื่อตารางในที่นี้ หมายถึง ชื่อของตารางที่ต้องการลบ ส่วนเงื่อนไขจะหมายถึง เงื่อนไขในการลบข้อมูล หรือลบเร็คคอร์ดในตารางนั้นๆ

สำหรับเครื่องหมาย * หมายถึง ข้อมูลใดๆ หรือข้อมูลทุกเร็คคอร์ด

NOTE ในการใช้งานคำสั่ง DELETE ในแบบที่ 2 ขอให้คุณใช้อย่างระมัดระวัง เพราะเป็นการลบข้อมูลทั้งหมดในตาราง

เพื่อความเข้าใจ ลองดูตัวอย่างต่อไปนี้

```
DELETE * FROM tblSell
```

เป็นการลบเร็คคอร์ดทั้งหมดที่อยู่ในตารางที่ชื่อว่า tblSell

```
DELETE  FROM tblCustomer WHERE OrderValue < 10000
```

เป็นการลบเร็คคอร์ดในตารางที่ชื่อว่า tblCustomer โดยมีเงื่อนไขว่า จะลบเฉพาะเร็คคอร์ดที่ค่าในฟิลด์มีชื่อว่า OrderValue มีค่าน้อยกว่า 10,000 เท่านั้น ออกจากตาราง ดังนั้น ถ้าในเร็คคอร์ดมีค่าในฟิลด์ OrderValue มากกว่า 10,000 ก็จะไม่ถูกลบออกไป

เรายังสามารถใช้ตัวดำเนินการด้านตรรกะ เช่น AND หรือ OR มาใช้ระบุเป็นเงื่อนไขได้อีกด้วย เช่น

```
DELETE  FROM tblStudent WHERE Gpa < 1.50 OR Point < 10
```

เป็นการลบเร็คคอร์ดจากตารางที่ชื่อว่า tblStudent โดยมีเงื่อนไขว่า จะลบเฉพาะชื่อนักศึกษาที่มีผลการเรียนต่ำกว่า 1.5 (ค่าของฟิลด์ที่ชื่อว่า Gpa น้อยกว่า 1.50) หรือมีคะแนนความประพฤติต่ำกว่า 10 คะแนน (ค่าของฟิลด์ที่ชื่อว่า Point น้อยกว่า 10)

4.2.2 คำสั่ง INSERT

เป็นคำสั่งที่ใช้สำหรับเพิ่มข้อมูล หรือเพิ่มเร็คคอร์ดเข้าไปในตาราง ในกรณีที่ฟิลด์เป็นข้อมูลชนิด Text คุณต้องใช้เครื่องหมาย ' กำกับฟิลด์นั้นด้วย มีรูปแบบการใช้งาน 2 ลักษณะ ดังนี้

```
รูปแบบที่ 1     INSERT INTO tablename (field 1, field 2 , field 3…. ) VALUES (value 1,
                value 2,value 3…)
รูปแบบที่ 2     INSERT INTO tablename2
        SELECT * FROM tablename 1 WHERE criteria
```

กรณีนี้ ผู้เขียนสมมติให้ field 2 เป็นข้อมูลชนิด Text

ตัวแปร tablename หมายถึง ชื่อตารางที่ต้องการเพิ่มเร็คคอร์ดเข้าไป

ตัวแปร tablename 1 หมายถึง เลือกข้อมูลจากตารางที่ชื่อว่า tablename 1 ตามเงื่อนไขที่ระบุไว้ในตัวแปร criteria แล้วนำมาเพิ่มที่ตาราง tablename 2

ตัวแปร field 1-fieldn หมายถึง ชื่อของฟิลด์ต่างๆที่อยู่ในตารางที่คุณต้องการเพิ่มข้อมูล

ตัวแปร value1-valuen หมายถึง ค่าของฟิลด์ที่จะเพิ่มเข้าไป โดยที่คุณต้องระบุค่าให้ตรงกับฟิลด์ด้วย

ตัวแปร criteria หมายถึง เงื่อนไขในการดึงข้อมูลจากตาราง tablename 1

เพื่อความเข้าใจ เรามาลองศึกษาตัวอย่างต่อไปนี้

```
INSERT INTO tblCustomer (ID,FirstName,LastName,Salary)
        VALUES (0005,'ปิยพร',' เพ็ชรพงศ์ ',10000
```

จากตัวอย่างเป็นการเพิ่มข้อมูลเข้าไปในตาราง ชื่อว่า tblCustomer ประกอบไปด้วย ฟิลด์ ID, Firstname, Lastname, Salary โดยที่ให้ใส่ค่า 0005 ลงในฟิลด์ ID, ปิยพร ลงในฟิลด์ Firstname, เพ็ชรพงศ์ ลงในฟิลด์ Lastname ฟิลด์ทั้ง 2 เป็นข้อมูลชนิด Text จึงต้องใช้เครื่องหมาย ' กำกับไว้ และใส่ค่า 10000 ลงในฟิลด์ Salary

```
INSERT INTO tblTop (ID,FirstName,LastName,Gpa)
        SELECT * FROM tblStudent WHERE Gpa >= 3.50
```

ต้องการดึงข้อมูลรายชื่อนักศึกษาจากตารางที่ชื่อว่า tblStudent เฉพาะนักศึกษาที่มีผลการเรียนมากกว่าหรือเท่ากับ 3.50 (ค่าของฟิลด์ Gpa มากกว่าหรือเท่ากับ 3.50) เพิ่มเข้าไปในตารางนักศึกษาเรียนดี (ตารางชื่อว่า tblTop) โดยให้ใส่ค่าตามลำดับฟิลด์ ID, Firstname, Lastname, และ Gpa

ข้อควรระวังในการใช้งานคำสั่ง INSERT

- กรณีที่ 1 ใส่ค่าฟิลด์ที่เป็น Primary Key ซ้ำกับค่าเดิมที่มีอยู่แล้ว หรือฟิลด์ที่เป็น Primary Key คุณไม่ได้กำหนดค่า หรือใส่ค่าให้ (มีค่าเท่ากับ Null)
- กรณีที่ 2 ค่าที่ใส่เข้าไปซ้ำกับเร็คคอร์ดที่มีอยู่แล้วในฐานข้อมูล

ทั้ง 2 กรณี ส่งผลให้คำสั่ง INSERT ไม่มีการเพิ่มเร็คคอร์ดนั้นๆเข้าไปในตาราง เหตุผลให้ดูในตารางที่ 4-1 ประกอบการอธิบาย

ID	FirstName	LastName	Sex	Salary
0001	เอกพงษ์	สุรพันธ์พงษ์	M	10000
0002	อดิเทพ	ตรังคานนท์	M	12000
0003	สุรัตน์	จงประสบโชค	F	15000
0004	ศุภชัย	สมพานิช	M	12000
0005	ธิติมา	ยุราวรรณ	F	20000

จากตาราง tblTest กำหนดให้ฟิลด์ที่ชื่อว่า ID เป็น Primary Key สมมติว่า มีการเพิ่มข้อมูลเข้ามาอีก 1 เร็คคอร์ด ด้วยคำสั่ง

```
INSERT INTO tblTest (ID,FirstName,LastName,Sex,Salary) VALUES (0005,'ชลธิชา ','สีดา ','F ',10000)
```

จะเห็นได้ว่า ค่าในฟิลด์ ID (ซึ่งเป็น Primary Key ของตารางนี้) ซ้ำกับค่าเดิมที่มีอยู่แล้ว ตรงกับกรณีที่ 1 ทำให้ไม่มีการเพิ่มเร็คคอร์ดนี้แต่อย่างใด เพราะว่าฟิลด์ที่เป็น Primary Key จะต้องมีค่าไม่ซ้ำกัน

ต่อมาสมมติ ใช้คำสั่ง INSERT อีกครั้งดังนี้

```
INSERT INTO tblTest (ID,FirstName,LastName,Sex,Salary) VALUES (0003,'สุรัตน์ ','จงประสบโชค ','F ',15000)
```

จะเห็นได้ว่า ฟิลด์ที่ต้องการเพิ่มเข้าไปมีอยู่แล้วในตาราง ตรงกับกรณีที่ 2 ดังนั้นคำสั่งนี้จึงไม่มีการเพิ่มเร็คคอร์ดแต่อย่างใด

4.2.3 คำสั่ง SELECT

ใช้สำหรับเลือกหรือดึงข้อมูล (Retrieve Data) ที่เราต้องการจากตารางที่ระบุไว้ เป็นคำสั่งที่มีความยืดหยุ่นสูงมาก เพราะว่าเงื่อนไขในการนำข้อมูลออกมาจากตารางมีมากมายหลายลักษณะ แต่มีรูปแบบการใช้งานพื้นฐานมีอยู่ 2 ลักษณะคือ

```
SELETE * FROM ชื่อตาราง
```

หรือ

```
SELETE ฟิลด์ที่ 1,ฟิลด์ 2,....,ฟิลด์ที่ n FROM ชื่อตาราง WHERE เงื่อนไข
```

โดยชื่อตาราง จะหมายถึง ชื่อตารางที่ต้องการดึงข้อมูล

ส่วนตัวแปร ฟิลด์ที่ 1,ฟิลด์ที่ 2,..., ฟิลด์ที่ n จะหมายถึง ชื่อฟิลด์ที่ต้องการดึงข้อมูล ถ้ามีมากกว่า 1 ฟิลด์จะใช้เครื่องหมาย, คั่นระหว่างฟิลด์ สำหรับเงื่อนไข หมายถึง เงื่อนไขในการดึงข้อมูล อาจเป็นเงื่อนไขทางคณิตศาสตร์ หรือเป็นคำสั่ง SELECT ซ้อนอยู่ข้างในก็ได้

เพื่อความเข้าใจ ลองพิจารณาจากตัวอย่างต่อไปนี้

```
SELETE * FROM tblCustomer
```

เป็นการเลือกข้อมูลทุกเร็คคอร์ดจากตารางที่ชื่อว่า tblCustomer

```
SELETE ID,First,LastName,Salary FROM tblCustomer
```

เป็นการเลือกข้อมูลจากฟิลด์ที่เราต้องการ ใช้ในกรณีที่คุณไม่ต้องการแสดงข้อมูลทุกฟิลด์

```
SELETE * FROM tblStudent WHERE Gpa >= 3.50
```

เป็นการดึงเร็คคอร์ดเฉพาะที่มีค่าในฟิลด์ Gpa มีค่ามากกว่า 3.50 เท่านั้น

4.2.4 การใช้งานคำสั่ง SELECT แบบมีเงื่อนไข

เป็นการใช้งานคำสั่ง SELECT ร่วมกับตัวดำเนินการอื่นๆ เพื่อเป็นเงื่อนไขในการแสดงข้อมูล รวมถึงการใช้งานร่วมกับกลุ่มฟังก์ชั่น Aggregate ด้วย การใช้งานคำสั่ง SELECT แบบมีเงื่อนไข ให้ดูตารางที่ 4-2 ชื่อว่า tblEmployee ประกอบคำอธิบาย

Code	FirstName	LastName	Salary	Dept	DateStart
001	น.ส. ปิยพร	เพ็ชรพงศ์	20000	Manager	12/04/2000
002	น.ส.สุรัตน์	จงประสบโชค	14000	Teacher	10/03/1999
003	น.ส.ธิติมา	ยุราวรรณ	17500	Admin	16/12/1996
004	น.ส.ชลธิชา	สีดา	16000	ADBA	24/06/1999
005	นายอดิเทพ	ตรังคานนท์	12000	Staff	15/04/1999
006	นายพิทยา	โชติแสงชัยพฤกษ์	15000	Manager	20/07/1999
007	นายณัฐพล	กิจประชา	16000	Admin	18/05/1996
008	นายเอกพงษ์	สุรพันธ์พงษ์	13000	Staff	18/02/2000
009	น.ส.สาวินี	ไชยทอง	15000	DBA	07/07/1998
010	น.ส.นุสรา	น้อยสุพรรณ	15000	Teacher	27/09/1998

ตัวอย่างแรก ถ้าต้องการดูเฉพาะรายชื่อที่อยู่ในตำแหน่ง DBA เท่านั้น

```
SELECT FirstName, LastName, Salary
FROM tblEmployee
WHERE Dept = "DBA";
```

ผลที่ได้ คือ

FirstName	LastName	Salary
น.ส.ชลธิชา	สีดา	16000
น.ส.สาวินี	ไชยทอง	15000

ถ้าต้องการดูเฉพาะที่มีเงินเดือนน้อยกว่า 15000 และอยู่ในตำแหน่ง Teacher

```
SELECT FirstName, LastName, Salary, DateStart
FROM tblEmployee
WHERE Salary < 15000 AND Dept = 'Teacher';
```

ผลที่ได้ คือ

FirstName	LastName	Salary	DateStart
น.ส.สุรัตน์	จงประสบโชค	14000	10/03/1999

ตัวอย่างต่อไป เราจะมาสร้างเงื่อนไขการค้นหาที่ซับซ้อนมากยิ่งขึ้น เช่น ต้องการหาฟิลด์ Salary >= 15000 เป็นอันดับแรกก่อน จากนั้นจึงนำเงื่อนไขที่ระบุไว้ไปจำกัดรายการเร็คคอร์ดที่ได้ ให้เหลือแต่เร็คคอร์ดที่ตรงกับเงื่อนไข ดังนี้

```
SELECT Code, FirstName, LastName, Salary
FROM tblEmployee
WHERE Salary >= 15000 AND Dept = 'Teacher' OR Dept = 'DBA';
```

ผลที่ได้ คือ

Code	FirstName	LastName	Salary
004	น.ส.ชลธิชา	สีดา	16000
009	น.ส.สาวินี	ไชยทอง	15000
010	น.ส.นุสรา	น้อยสุพรรณ	15000

กรณีที่ต้องการรายชื่อพนักงาน โดยใช้ตำแหน่งเป็นเงื่อนไข อาจใช้คำสั่ง IN และ BETWEEN ก็ได้ เช่น ต้องการแสดง รายชื่อพนักงานที่อยู่ในตำแหน่ง Manage และ Admin ทั้งหมด

```
SELECT FirstName, LastName, Salary, Dept
FROM tblEmployee
WHERE Dept IN ('Manager', 'Admin');
```

ผลที่ได้ คือ

FirstName	LastName	Salary	Dept
น.ส. ปิยพร	เพ็ชรพงศ์	20000	Manager
น.ส. ปิยพร	เพ็ชรพงศ์	20000	Manager
นายพิทยา	โชติแสงชัยพฤกษ์	15000	Manager
นายณัฐพล	กิจประชา	16000	Admin

หรือใช้คำสั่ง BETWEEN เพื่อจำกัดขอบเขตก็ได้ เช่น ต้องการรายชื่อเฉพาะที่มีเงินเดือนอยู่ระหว่าง 13,000-15,000 จะพบว่า เงินเดือนที่เท่ากับ 13,000 หรือ 15,000 จะรวมเข้ามาด้วย

```
SELECT FirstName, LastName, Dept, Salary
FROM tblEmployee
WHERE Salary BETWEEN 13000 AND 15000
```

ต้องการรายชื่อเฉพาะที่มีเงินเดือนอยู่ระหว่าง 13,000 – 15,000 จะพบว่า เงินเดือนที่เท่ากับ 13,000 หรือ 15,000 จะรวมเข้ามาด้วย

FirstName	LastName	Dept	Salary
น.ส.สุรัตน์	จงประสบโชค	Teacher	14000
นายอดิเทพ	ตรังคานนท์	Staff	12000
นายพิทยา	โชติแสงชัยพฤกษ์	Manager	15000
นายเอกพงษ์	สุรพันธ์พงษ์	Staff	13000
น.ส.สาวินี	ไชยทอง	DBA	15000
น.ส.นุสรา	น้อยสุพรรณ	Teacher	15000

ในทางกลับกัน ถ้าไม่ต้องการให้อยู่ในช่วงที่กำหนดไว้ ให้ช่วยตัวดำเนินการ NOT เข้าช่วย

```
SELECT FirstName, LastName, Dept, Salary
FROM tblEmployee
WHERE Salary NOT BETWEEN 13000 AND 15000;
```

ผลที่ได้ก็คือ เร็คคอร์ดที่อยู่ในช่วงที่กำหนดไว้จะถูกตัดออกทั้งหมด

FirstName	LastName	Dept	Salary
น.ส. ปิยพร	เพ็ชรพงศ์	Manager	20000
น.ส.ธิติมา	ยุราวรรณ	Admin	17500
น.ส.ชลธิชา	สีดา	ADBA	16000
นายณัฐพล	กิจประชา	Admin	16000

กรณีที่ต้องการจำกัดจำนวนเร็คคอร์ดที่แสดงออกมา ให้ใช้คำสั่ง Top

```
SELECT Top 5 * FROM tblEmployee;
```

เป็นการเลือกทุกฟิลด์ จากตาราง tblEmployee จำกัดให้แสดง 5 เร็คคอร์ดแรกเท่านั้นผลที่ได้คือ

Code	FirstName	LastName	Salary	Dept	DateStart
001	น.ส. ปิยพร	เพ็ชรพงศ์	20000	Manager	12/4/2000
002	น.ส.สุรัตน์	จงประสบโชค	14000	Teacher	10/3/1999
003	น.ส.ธิติมา	ยุราวรรณ	17500	Admin	16/12/1996
004	น.ส.ชลธิชา	สีดา	16000	ADBA	24/6/1999
005	นายอดิเทพ	ตรังคานนท์	12000	Staff	15/4/1999

ถ้าต้องการเรียงลำดับข้อมูลที่จะแสดงออกมาด้วย ให้ใช้คำสั่ง ORDER BY ร่วมกับชื่อฟิลด์ที่คุณต้องการใช้เป็นเงื่อนไขในการเรียงลำดับ

ถ้าคุณไม่ระบุ จะเป็นการเรียงลำดับจากค่าน้อยไปหาค่ามาก หรือถ้าเป็นข้อความก็จะเรียงตามลำดับตัวอักษร แต่ถ้าในกรณีที่ต้องการเรียงลำดับจากค่ามากไปหาค่าน้อย ให้ใช้คำสั่ง DESC กำกับไว้ด้วย

```
SELECT * FROM tblEmployee Order By DateStart DESC
```

เป็นการเรียงลำดับจากค่าน้อยไปหาค่ามาก โดยใช้ฟิลด์ DataStart เป็นเงื่อนไข ผลที่ได้ คือ

Code	FirstName	LastName	Salary	Dept	DateStart
007	นายณัฐพล	กิจประชา	16000	Admin	18/05/1996
003	น.ส.ธิติมา	ยุราวรรณ	17500	Admin	16/12/1996
009	น.ส.สาวินี	ไชยทอง	15000	DBA	07/07/1998
010	น.ส.นุสรา	น้อยสุพรรณ	15000	Teacher	27/09/1998
002	น.ส.สุรัตน์	จงประสบโชค	14000	Teacher	10/03/1999
005	นายอดิเทพ	ตรังคานนท์	12000	Staff	15/04/1999
004	น.ส.ชลธิชา	สีดา	16000	ADBA	24/06/1999
006	นายพิทยา	โชติแสงชัยพฤกษ์	15000	Manager	20/07/1999
008	นายเอกพงษ์	สุรพันธ์พงษ์	13000	Staff	18/02/2000
001	น.ส. ปิยพร	เพ็ชรพงศ์	20000	Manager	12/04/2000

กรณีที่ต้องการเรียงจากค่ามากไปหาค่าน้อย

```
SELECT * FROM tblEmployee Order By DateStart DESC;
```

กรณีที่ข้อมูลแสดงออกมาซ้ำกันหลายๆเร็คคอร์ด ถ้าคุณต้องการแสดงเพียงเร็คคอร์ดเดียว ให้ใช้คำสั่ง DISTINCT เช่น

```
SELECT  Salary FROM tblEmployee;
```

เป็นการเลือกเฉพาะฟิลด์เงินเดือน ผลที่ได้ คือ

Salary
20000
14000
17500
16000
12000
15000
16000
13000
15000
15000

จากผลที่ได้จะพบว่า มีค่าที่ซ้ำกันอยู่ คือ 15000 และ 16000 แต่ถ้าคุณใช้คำสั่ง DISTINCT

```
SELECT DISTINCT Salary FROM tblEmployee;
```

ผลที่ได้ คือ

Salary
20000
14000
17500
16000
12000
15000
13000

กรณีที่ใช้ร่วมกับกลุ่มฟังก์ชั่น Aggregate เช่น ฟังก์ชั่น MAX ()

```
SELECT MAX(Salary) FROM tblEmployee;
```

เป็นการหาฟิลด์ Salary ที่มีค่าสูงสุด ผลที่ได้คือ

Salary
20000

ส่วนฟังก์ชั่น Aggregate อื่นๆจะมีลักษณะการใช้งานเช่นเดียวกับฟังก์ชั่น MAX () ผลที่ได้จะเป็นไปตามหน้าที่ของแต่ละฟังก์ชั่น

4.2.5 การใช้งานคำสั่ง SELECT ร่วมกับคำสั่ง LIKE

คำสั่ง LIKE เป็นคำสั่งที่ใช้สำหรับกำหนดเงื่อนไขที่เกี่ยวข้องกับข้อมูลชนิดข้อความ (Text หรือ String) ปกติแล้วจะใช้ร่วมกับตัวอักษรพิเศษ ที่เรียกว่า Wild Card Characters เพื่อระบุเป็นเงื่อนไขให้กับคำสั่ง LIKE ในการจำกัดข้อมูลที่เป็นข้อความ

ตารางแสดงตัวอักษรพิเศษ

ลักษณะอักษร	ความหมาย
*	ใช้แทนตัวอักษรใดๆ โดยไม่มีการจำกัดจำนวนตัวอักษร เช่น ca* หมายถึง ข้อความใดก็ตามที่มีตัวอักษร ca จะถือว่าใช่ทั้งหมด เช่น cat, catch, canada, cash, case เป็นต้น จะเห็นได้ว่า ไม่จำกัดจำนวนตัวอักษรใน 1 คำ แต่ขอให้มีตัวอักษร ca เท่านั้น
?	ใช้แทนตัวอักษรใดๆ เช่นกัน แต่จำกัดตัวอักษร เช่น ca?? หมายถึง cash, case, card เป็นต้น
%	มีความหมายเช่นเดียวกับ
#	ใช้กับข้อมูลชนิดตัวเลข แทน 1 ตัวอักษร เช่น 12# หายถึง 123, 124, 125 เป็นต้น
[ ]	ใช้กำหนดขอบเขตของข้อความ เช่น [a-d] หมายถึง ตัวอักษร a ถึง d เท่านั้น
[!]	ความหมายตรงข้ามกับ [ ] เช่น [!a-d] หมายถึง ตัวอักษร e ถึง z

ตัวอย่างการใช้งานคำสั่ง LIKE ร่วมกับ Wild Card Characters เช่น ต้องการแสดงรายละเอียดของพนักงานที่ชื่อของเขามีตัวอักษร ส. อยู่ภายใน

```
SELECT * FROM tblEmployee
WHERE FirstName LIKE "%ส%"
```

ผลที่ได้ คือ

Code	FirstName	LastName	Salary	Dept	DateStart
002	น.ส.สุรัตน์	จงประสบโชค	14000	Teacher	10/03/1999
009	น.ส.สาวินี	ไชยทอง	15000	DBA	07/07/1996
010	น.ส.นุสรา	น้อยสุพรรณ	15000	Teacher	27/09/1998

ในทางกลับกัน ถ้าไม่ต้องการชื่อที่มี ส อยู่ด้วย ให้ใช้คำสั่งนี้

```
SELECT * FROM tblEmployee
WHERE FirstName LIKE "%ส%"
```

4.2.6 คำสั่ง UPDATE

เป็นคำสั่งที่ใช้สำหรับแก้ไข หรือเปลี่ยนแปลงข้อมูลในเร็คคอร์ดที่มีอยู่ในตาราง มีรูปแบบการใช้งาน ดังนี้

```
UPDATE ชื่อตาราง SET ชื่อฟิลด์ = ค่าที่กำหนด WHERE เงื่อนไข
```

จากรูปแบบคำสั่งนั้น เราจะระบุชื่อตารางที่ต้องการแก้ไขไว้ในชื่อตาราง และกำหนดฟิลด์ที่ต้องการแก้ไขไว้ในชื่อฟิลด์ ซึ่งค่าที่ถูกแก้ไขจะกำหนดไว้ในค่าที่กำหนด โดยจะมีเงื่อนไขในการแก้ไขไว้ในเงื่อนไข

เพื่อความเข้าใจลองศึกษาจากตัวอย่างต่อไปนี้

```
UPDATE tblStudent SET Point = 50
```

จากตัวอย่างเป็นการแก้ไขฟิลด์ที่ชื่อว่า Point เฉพาะที่มีค่าน้อยกว่า 10 ให้มีค่าเท่ากับ 50ในตาราง tblStudent

```
UPDATE tblStudent SET Point = 50 WHERE Point < 10
```

4.3 การเชื่อมโยงตาราง (Joins Tables)

การเชื่อมโยงตาราง เป็นการนำข้อมูลจากตารางตั้งแต่ 2 ตารางขึ้นไป มาแสดงร่วมกันในเวลาเดียวกัน ให้คุณดูตารางที่ 4-3 และ 4-4 ประกอบคำอธิบาย

ตารางที่ 4-3 student รายชื่อนักศึกษา

StudentID	FirstName	LastName	MajorID
410001	น.ส. ปิยพร	เพ็ชรพงศ์	01
410002	น.ส.ธิติมา	ยุราวรรณ	01
410003	นายณัฐพล	กิจประชา	02
410004	น.ส.กานดา	แสงสว่าง	03
410005	นายพิทยา	โชติแสงชัยพฤกษ์	01
410006	น.ส.สุรัตน์	จงประสบโชค	04
410007	น.ส.ชลธิชา	สีดา	03

ตารางที่ 4-4 major รายชื่อคณะที่เปิดสอน

MajorID	MajorDesc
01	คอมพิวเตอร์
02	การตลาด
03	การเงินและธนาคาร
04	การบัญชี
05	วิศวคอมพิวเตอร์

การเชื่อมโยงตารางตั้งแต่ 2 ตารางขึ้นไป จะเรียกว่า Join Tables มีอยู่ 4 ลักษณะ คือ

1. ใช้คำสั่ง SELECT
2. ใช้คำสั่ง INNER JOIN
3. ใช้คำสั่ง LEFT JOIN
4. ใช้คำสั่ง RIGHT JOIN

NOTE ยังมีการ Join Table อีกลักษณะหนึ่งที่ว่า Outer Join หรือเรียกอีกอย่างว่า Full Join แต่ผู้เขียนจะไม่ถือว่า เป็นอีก 1 แบบ เพราะว่า เป็นการใช้คำสั่ง Left และ Right อีกลักษณะหนึ่ง

4.3.1 การเชื่อมโยงตารางโดยใช้คำสั่ง SELECT

วิธีการนี้เป็นการเชื่อมโยงตาราง โดยระบุชื่อฟิลด์ที่มีความสัมพันธ์กัน เช่น

```
SELECT StudentID, FirstName, LastName, MajorDesc
FROM student, major
WHERE student.MajorID = major.MajorID
```

เป็นการเชื่อมโยงตาราง student เข้ากับตาราง major การเชื่อมโยงตารางแบบนี้มีเงื่อนไขเล็กน้อย คือ ฟิลด์ที่นำมาเชื่อมโยงกัน จะต้องเป็นฟิลด์ที่มีชนิดของข้อมูลเหมือนกัน แต่ชื่อฟิลด์ไม่จำเป็นต้องเหมือนกัน กรณีนี้ใช้ฟิลด์ MajorID ของตาราง student เชื่อมโยงกับฟิลด์ MajorID ของตาราง major ผลที่ๆได้ คือ

StudentID	FirstName	LastName	MajorDesc
410001	น.ส. ปิยพร	เพ็ชรพงศ์	คอมพิวเตอร์
410002	น.ส.ธิติมา	ยุราวรรณ	คอมพิวเตอร์
410003	นายณัฐพล	กิจประชา	การตลาด
410004	น.ส.กานดา	แสงสว่าง	การเงินและธนาคาร
410005	นายพิทยา	โชติแสงชัยพฤกษ์	คอมพิวเตอร์
410006	น.ส.สุรัตน์	จงประสบโชค	การบัญชี
410007	น.ส.ชลธิชา	สีดา	การเงินและธนาคาร

4.3.2 การเชื่อมโยงตารางโดยใช้คำสั่ง INNER JOIN

วิธีการนี้เป็นการเชื่อมโยงข้อมูลจาก 2 ตารางเข้าด้วยกัน โดยระบุชื่อฟิลด์ที่มีความสัมพันธ์กัน หลังคำสั่ง ON เป็นเงื่อนไขการ JOIN ตาราง โดยที่ฟิลด์ดังกล่าว จะต้องเป็นข้อมูลชนิดเดียวกัน มีลักษณะเช่นเดียวกับการใช้คำสั่ง SELECT มีรูปแบบการใช้งานดังนี้

```
SELECT fieldname
FROM tablename1
INNER JOIN tablename2
ON tablename1.fieldname1 OPERATION tablename2.fieldname2
```

ตัวแปร fieldname หมายถึง ชื่อฟิลด์ที่อยู่ในตารางที่ 1 หรือ 2

ตัวแปร tablename1 หมายถึง ชื่อตารางที่ 1

ตัวแปร tablename2 หมายถึง ชื่อตารางที่ 2

ตัวแปร fieldname 1 หมายถึง ชื่อฟิลด์ที่อยู่ในตารางที่ 1

ตัวแปร fieldname 2 หมายถึง ชื่อฟิลด์ที่อยู่ในตารางที่ 2

ตัวแปร operation หมายถึง เงื่อนไขหรือตัวดำเนินการต่างๆ ที่ใช้ในการเชื่อมโยง

เพื่อความเข้าใจในการเชื่อมโยงตารางด้วยคำสั่ง INNER JOIN เราลองพิจารณาจากตัวอย่างต่อไปนี้

```
SELECT StudentID, FirstName, LastName, MajorDesc
FROM student
INNER JOIN major
ON student. MaiorID = maior.MaiorID
```

ผลของคำสั่งนี้ จะเหมือนกับกรณีใช้คำสั่ง SELECT ของหัวข้อที่ผ่านมา

4.3.3 การเชื่อมโยงตารางโดยใช้คำสั่ง LEFT JOIN

วิธีนี้เป็นการเชื่อมโยง 2 ตารางเข้าด้วยกัน โดยกำหนดให้ตารางแรก (ตารางทางซ้ายของคำสั่ง LEFT JOIN) เป็นหลัก แล้วนำตารางที่ 2 เข้ามาเชื่อมโยง ตามเงื่อนไขที่ระบุไว้ มีรู)แบบการใช้งานดังนี้

```
SELECT fieldname
FROM tablename1 LEFT JOIN tablename2
ON tablename1.fieldname1 OPERATION tablename2.fieldname2
```

ตัวแปร fieldname หมายถึง ชื่อฟิลด์ที่อยู่ในตารางที่ 1 หรือ 2

ตัวแปร tablename1 หมายถึง ชื่อตารางที่ 1

ตัวแปร tablename2 หมายถึง ชื่อตารางที่ 2

ตัวแปร fieldname 1 หมายถึง ชื่อฟิลด์ที่อยู่ในตารางที่ 1

ตัวแปร fieldname 2 หมายถึง ชื่อฟิลด์ที่อยู่ในตารางที่ 2

ตัวแปร operation หมายถึง เงื่อนไขหรือตัวดำเนินการต่างๆ ที่ใช้ในการเชื่อมโยง

เพื่อความเข้าใจในการเชื่อมโยงตารางด้วยคำสั่ง LEFT JOIN เราลองพิจารณาจากตัวอย่างต่อไปนี้

```
SELECT StudentID, FirstName, LastName, MajorDesc
FROM student LEFT JOIN major
ON student.MajorID = major.MajorID
```

จากตัวอย่างเป็นการกำหนดตาราง student เป็นหลัก เพราะว่าอยู่ทางซ้ายของคำสั่ง LEFT JOIN จากนั้นจะนำเร็คคอร์ดจากตาราง major ที่จะนำมาแสดง จะต้องมีรหัสคณะ (MajorID) ตรงกับตาราง student ผลที่ได้ คือ

StudentID	FirstName	LastName	MajorDesc
410001	น.ส. ปิยพร	เพ็ชรพงศ์	คอมพิวเตอร์
410002	น.ส.ธิติมา	ยุราวรรณ	คอมพิวเตอร์
410003	นายณัฐพล	กิจประชา	การตลาด
410004	น.ส.กานดา	แสงสว่าง	การเงินและธนาคาร
410005	นายพิทยา	โชติแสงชัยพฤกษ์	คอมพิวเตอร์
410006	น.ส.สุรัตน์	จงประสบโชค	การบัญชี
410007	น.ส.ชลธิชา	สีดา	การเงินและธนาคาร

NOTE ในกรณีที่ชื่อฟิลด์ในตารางมีช่องว่าง เช่น First name, Last name ให้คุณใส่เครื่องหมาย [] ครอบชื่อฟิลด์นั้นๆด้วย เมื่อคุณนำชื่อฟิลด์ดังกล่าวมาใช้ใน SQL เช่น Select [First Name] , [Last Name] From Student เป็นต้น

4.3.4 การเชื่อมโยงตารางโดยใช้คำสั่ง RIGHT JOIN

วิธีการนี้เป็นการเชื่อมโยง 2 ตารางเข้าด้วยกัน แต่กำหนดให้ตารางที่อยู่ทางด้านขวาของคำสั่ง Right Join เป็นหลัก แล้วนำเร็คคอร์ดของตารางที่อยู่ทางซ้ายของคำสั่ง RIGHT JOIN เข้ามาร่วมแสดงเฉพาะเร็คคอร์ดที่ตรงตามเงื่อนไขที่ได้ระบุไว้ มีรูปแบบการใช้งานดังนี้

```
SELECT fieldname
FROM tablename1 RIGHT JOIN tablename2
ON tablename1.fieldname1 OPERATION tablename2.fieldname2
```

ตัวแปร fieldname หมายถึง ชื่อฟิลด์ที่อยู่ในตารางที่ 1 หรือ 2

ตัวแปร tablename1 หมายถึง ชื่อตารางที่ 1

ตัวแปร tablename2 หมายถึง ชื่อตารางที่ 2

ตัวแปร fieldname 1 หมายถึง ชื่อฟิลด์ที่อยู่ในตารางที่ 1

ตัวแปร fieldname 2 หมายถึง ชื่อฟิลด์ที่อยู่ในตารางที่ 2

ตัวแปร operation หมายถึง เงื่อนไขหรือตัวดำเนินการต่างๆ ที่ใช้ในการเชื่อมโยง

เพื่อความเข้าใจในการเชื่อมโยงตารางด้วยคำสั่ง RIGHT JOIN เราลองพิจารณาจากตัวอย่างต่อไปนี้

```
Select StudentID, FirstName, LastName, MajorDesc
From student RIGHT JOIN major
On student.MajorID = major.MajorID
```

จากตัวอย่างเป็นการกำหนดให้ตาราง major เป็นหลัก แล้วนำเร็คคอร์ดที่อยู่ในตาราง student มาแสดงร่วมกัน โดยมีเงื่อนไขว่า เอาเฉพาะเร็คคอร์ดที่มีรหัสคณะ (ฟิลด์ MajorID) ตรงกับที่อยู่ในตาราง major เท่านั้น ผลที่ได้ คือ

StudentID	FirstName	LastName	MajorDesc
410001	น.ส. ปิยพร	เพ็ชรพงศ์	คอมพิวเตอร์
410002	น.ส.ธิติมา	ยุราวรรณ	คอมพิวเตอร์
410005	นายพิทยา	โชติแสงชัยพฤกษ์	คอมพิวเตอร์
410003	นายณัฐพล	กิจประชา	การตลาด
410004	น.ส.กานดา	แสงสว่าง	การเงินและธนาคาร
410007	น.ส.ชลธิชา	สีดา	การเงินและธนาคาร
410006	น.ส.สุรัตน์	จงประสบโชค	การบัญชี
			วิศวคอมพิวเตอร์

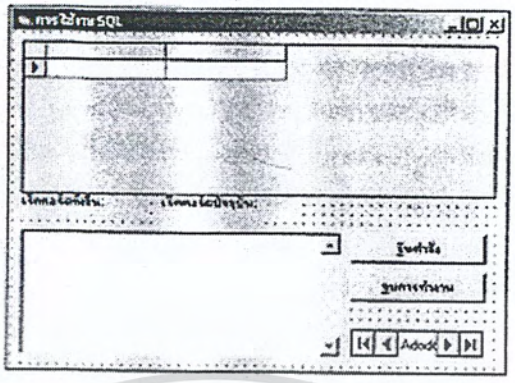
จากผลที่ได้สามารถอธิบายได้ดังนี้ เร็คคอร์ดแรกของตาราง major ระบุไว้ว่าเป็นรหัส 01 หมายถึง คณะคอมพิวเตอร์ ดังนั้น จึงแสดงรายชื่อนักศึกษาคณะคอมพิวเตอร์ทั้งหมดขึ้นมาก่อน

ในตาราง major เร็คคอร์ดต่อมาระบุว่า 02 หมายถึง คณะการตลาด ก็คือ แสดงรายชื่อนักศึกษาคณะการตลาดทั้งหมดเป็นลำดับถัดมา จนกระทั่งถึงคณะสุดท้าย คือ 05 วิศวคอมพิวเตอร์ ที่ยังไม่มีนักศึกษาคนใดเลือกนั่นเอง

4.4 ตัวอย่างการใช้งานภาษา SQL : โปรเจ็กต์ 4-1

ตัวอย่างโปรเจ็กต์ที่ 4-1 การใช้งานภาษา SQL ใช้สำหรับทดลองภาษา SQL ได้เป็นอย่างดี คุณสามารถนำไปใช้กับฐานข้อมูลอื่นก็ได้

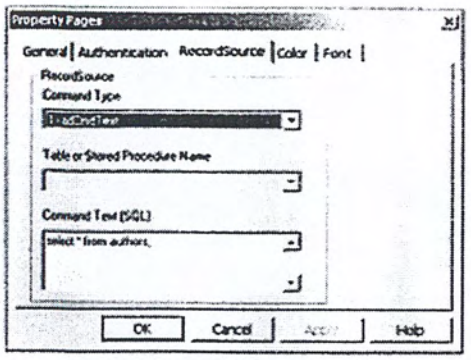
1. สำหรับตัวอย่างนี้ผู้เขียนทดลองใช้กับฐานข้อมูล Biblio.mdb ให้คุณออกแบบฟอร์มดังรูปที่ 4-1



รูปที่ 4.1 ฟอร์มที่ใช้รันคำสั่ง

2. ให้คุณกำหนดคุณสมบัติของคอนโทรลต่างๆดังนี้

คอนโทรล	คุณสมบัติ	ค่าที่กำหนด
Adodc1		เชื่อมต่อกับฐานข้อมูล Biblio.mdb กับตาราง Titles ให้สร้างชุดคำสั่ง ดังรูป 4-2
DataGrid1	DataSource	Adodc1
Label1	Caption	เร็คคอร์ดทั้งสิ้น
Label2	Caption	
Label3	Caption	เร็คคอร์ดปัจจุบัน
Label4	Caption	
TextBox1	MultiLine	TRUE
command1	Name	cmdRun
	Caption	&รันคำสั่ง
command2	Name	cmdEnd
	Caption	&จบการทำงาน



รูปที่ 4.2 สร้างชุดคำสั่ง

3. จากนั้นเขียนโค้ดต่อไปนี้

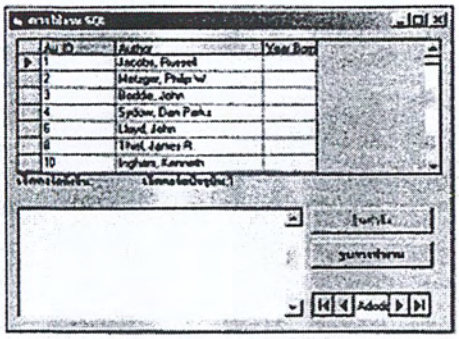
โค้ดโปรเจ็กต์ที่ 4 – 1 การใช้งานคำสั่ง SQL

```
Private Sub Form_Load()
'ตรวจสอบก่อนว่า ในคอนโทรล TextBox ว่างหรือไม่
If Text1.Text = "" Then          'ถ้าว่าง กำหนดให้ปุ่ม RUN ใช้งานไม่ได้ชั่วคราว
        CmdRun.Enabled = False
End If
End Sub
'ขณะที่พิมพ์ข้อความในคอนโทรล TextBox
Private Sub Text1_KeyPress(KeyAscii As Integer)
If Text1.Text<> "" Then          ' ถ้าคอนโทรล TextBox มีข้อความแล้ว
        CmdRun.Enabled = True 'กำหนดให้ปุ่ม Run ใช้งานได้
End If
End Sub
Private Sub Adodc1_MoveComplete (ByVal adReason As ADODB.EventReasonEnum, ByVal pError As
ADODB.Error, adStatus As ADODB.EventStatusEnum, ByVal pRecordset As ADODB.Recordset)
'แสดงเร็คคอร์ดปัจจุบันที่คอนโทรล Label4
        Label4.Caption = Adodc1.Recordset.AbsolutePosition
End Sub
```

โค้ดโปรเจ็กต์ที่ 4 – 1 การใช้งานคำสั่ง SQL ต่อ

```
Private Sub cmdEnd_Click ()
        End                         'จบการทำงาน
End Sub

Private Sub cmdRun_Click ()
On Error GoTo SQLError          'ถ้ามีข้อผิดพลาดให้ไปที่ SQLError
กำหนดให้คอนโทรล Adodc1 ติดต่อกับฐานข้อมูล โดยใช้คำสั่ง SQL จากคอนโทรล TextBox1
With Adodc1
        .RecordSource = Text1.Text
        .Refresh
End With
'ถ้าพบเร็คคอร์ดแล้ว
If Adodc1.RecordSource <> "" Then
        If (Adodc1.Recordset.RecordCount >= 1) Then     ' ถ้ามีจำนวนเร็คคอร์ดมากกว่า 1 เร็คคอร์ด
                With Adodc1.Recordset                   ' หาจำนวนเร็คคอร์ด
                        .MoveLast
                        .MoveFirst
                        Label2.Caption = .RecordCount   ' แสดงจำนวนเร็คคอร์ดทั้งหมดที่ Label2
                End With
        End If
End if
Exit Sub
SQLErro;
        MsgBox Err.Description                          ' แสดงสาเหตุที่ผิดพลาด
        Exit Sub
End Sub
```



รูปที่ 4.3 ผลการรันโปรเจ็กต์

5. จากรูปที่ 4-3 จะพบว่า เมื่อยังไม่ได้ใส่ชุดคำสั่ง SQL ปุ่ม รันคำสั่ง ก็จะยังใช้ไม่ได้ คุณสามารถทดลองภาษา SQL ทำคิวรีโดยใช้เงื่อนไขต่างๆก็ได้เช่นกัน

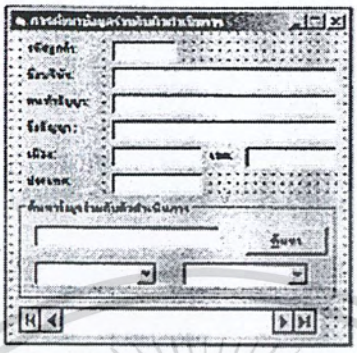
รูปที่ 4.4 ตัวอย่างการทำคิวรี

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

4.4.1 ตัวอย่างการใช้งานภาษา SQL : โปรเจ็กต์ 4-2

ตัวอย่างโปรเจ็กต์ที่ 4-2 การค้นหาข้อมูลร่วมกับตัวดำเนินการ ผู้เขียนจะใช้ร่วมกับฐานข้อมูล nwid.mdb ซึ่งเป็นฐานข้อมูลตัวอย่างอีกตัวหนึ่ง ที่มากับการติดตั้ง Visual Basic 6.0

1. ให้คุณออกแบบฟอร์มดังรูปที่ 4-5



รูปที่ 4.5 แบบฟอร์มการค้นหาข้อมูล

2. ให้คุณกำหนดคุณสมบัติของคอนโทรลต่างๆดังนี้

คอนโทรล	คุณสมบัติ	ค่าที่กำหนด
Label1	Caption	รหัสลูกค้า
Label2	Caption	ชื่อบริษัท
Label3	Caption	คนทำสัญญา
Label4	Caption	ชื่อสัญญา
Label5	Caption	เมือง
Label6	Caption	เขต
Label7	Caption	ประเทศ
TextBox1	DataSource	Adodc1
	DataFied	CustomerID
TextBox2	DataSource	Adodc1
	DataFied	CompaneName
TextBox3	DataSource	Adodc1
	DataFied	ContactName
TextBox4	DataSource	Adodc1
	DataFied	ContactTitle

	DataFied	ContactTitle
TextBox5	DataSource	Adodc1
	DataFied	City
TextBox6	DataSource	Adodc1
	DataFied	Region
TextBox7	DataSource	Adodc1
	DataFied	Country
Command1	Name	cmdSearch
	Caption	&ค้นหา
Frame1	Caption	ค้นหาข้อมูลร่วมกับตัวดำเนินการ

3. จากนั้นให้คุณเขียนโค้ดต่อไปนี้

โค้ดโปรเจ็กต์ที่ 4 – 2 การค้นหาข้อมูลร่วมกับตัวดำเนินการ

```
Private Sub Form_Load ()
'ติดต่อกับฐานข้อมูล
        Adodc1.ConnectionString =_
                Provider=Microsoft.Jet.OLEDB.4.0;Data Source = D:\Program Files\Microsoft Visual
Studio\VB98\nwind.MDB;Persist Security Info=False"
'เพิ่มชื่อฟิลด์ไปใน ComboBox1
With Combo1
        .AddItem "CustomerID"
        .AddItem "CompanyName"
        .AddItem "ContactName"
        .AddItem "ContactTitle"
        .AddItem "City"
        .AddItem "Region"
        .AddItem "Country"
        .ListIndex = 0
```

โค้ดโปรเจ็กต์ที่ 4 – 2 การค้นหาข้อมูลร่วมกับตัวดำเนินการ (ต่อ)

```
        End With
        'เพิ่มตัวดำเนินการเข้าไปใน ComboBox2
With Combo2
        .AddItem "Like"
        .AddItem "<"
        .AddItem ">"
        .AddItem ">="
        .AddItem "<="
        .AddItem "<>"
        .AddItem "="
        .ListIndex = 0
End With
End Sub
Private Sub cmdSearch_Click ()
On Error GoTo ErrorTrap                    ' ถ้าผิดพลาดให้ไปที่ ErrorTrap
Dim sqlTmp As String                       ' ตัวแปรเก็บชุดคำสั่ง SQL
' เก็บเงื่อนไขที่ผู้ใช้เลือกใน ComboBox ทั้ง 2 ตัว
sqlTmp = Combo1 & " " & Combo2 & " '" & Text8.Text & "'"
' ติดต่อกับตาราง Customers โดยใช้ชุดคำสั่ง SQL ร่วมกับเงื่อนไขที่ตัวแปร sqlTmp เก็บไว้
With Adodc1
        .RecordSourct = "Select * From Customer Where" & sqlTmp
        .Refresh
        .Recordset.MoveLast
        .Recordset.MoveFirst
End With
```

โค้ดโปรเจ็กต์ที่ 4 – 2 การค้นหาข้อมูลร่วมกับตัวดำเนินการ (ต่อ)

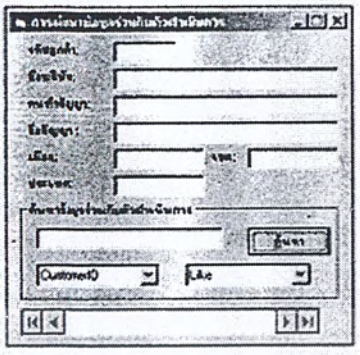
```
MsgBox "พบ" & Adodc1.Recordset.RecordCount & "เร็คคอร์ด?", vbOKOnly, "ผลการค้นหา"
Exit Sub

ErrorTrap;
        Adodc1.Caption = 'ไม่พบเร็คคอร์ดตามที่คุณต้องการ"
End Sub
'แสดงเร็คคอร์ดปัจจุบันและเร็คคอร์ดทั้งหมด
Private Sub Adodc1_MoveComplete(ByVal adReason As ADODB.EventReasonEnum, ByVal pError As

ADODB.Error, adStatus As ADODB.EventStatusEnum, ByVal pRecordset As ADODB.Recordset)
Dim CurrentRecord As Integer
Dim AllRecord As Integer

        With Adodc1.Recordset
                CurrentRecord = .AbsolutePosition
                IIRecord = .RecordCount
        End With
Adodc1.Caption = 'เร็คคอร์ดที่" & CurrentRecord & "/" & AllRecord
End Sub
'ถ้าผู้ใช้กดปุ่ม Enter แล้วให้เรียกโพรซีเจอร์ cmdSearch_Click ( ) ทำงาน
Private Sub Text8_KeyPress (KeyAscii As Integer)
If KeyAscii = 13 Then
        Call cmdSearch_Click
End if
End Sub
```

4. จากนั้นให้คุณรันโปรเจ็กต์ ดังรูปที่ 4-6



รูปที่ 4.6 ผลการรันโปรเจ็กต์

5. ให้คุณทดลองหาข้อมูล โดยเลือกชื่อฟิลด์ที่ต้องการเปรียบเทียบในคอนโทรล ComboBox1 และ เลือกตัวดำเนินการในคอนโทรล ComboBox2 จากนั้นคลิกปุ่ม ค้นหา หรือกดปุ่ม <Enter> ได้เลย

4.4.2 อธิบายการทำงานของโปรเจกต์ที่ 4-2

เพื่อความเข้าใจยิ่งขึ้น ผู้เขียนขออธิบายการทำงานของแอพพลิเคชันในโปรเจกต์ 4-2 ดังนี้

1. ในโพรซีเจอร์ Form Load () เริ่มติดต่อกับฐานข้อมูลโดยใช้คุณสมบัติ ConnectionString ของคอนโทรล ADO Data

```
Adodc1.ConnectionString =_
        Provider=Microsoft.Jet.OLEDB.4.0;Data Source = D:\Program Files\Microsoft Visual
Studio\VB98\nwind.MDB;Persist Security Info=False"
```

2. เนื่องจากว่าจะใช้ชื่อฟิลด์ในตาราง Customers เป็นเงื่อนไขในการค้นหา ดังนั้นจึงต้องเพิ่มชื่อฟิลด์ของตาราง Customers ไว้ในคอนโทรล ComboBox1 และกำหนดให้รายการ CustomerID ปรากฏขึ้นมาเป็นรายการแรกด้วยคุณสมบัติ Listindex=e ของคอนโทรล ComboBox

```
With Combo1
        .AddItem "CustomerID"
        .AddItem "CompanyName"
        .AddItem "ContactName"
        .AddItem "ContactTitle"
        .AddItem "City"
        .AddItem "Region"
        .AddItem "Country"
        .ListIndex = 0
End With
```

3. ส่วนรายการในคอนโทรล ComboBox2 จะเก็บตัวดำเนินการต่างๆและกำหนดให้รายการ Like ปรากฏขึ้นมาเป็นลำดับแรกด้วยคุณสมบัติ Listindex=0 ด้วยเช่นกัน

```
With Combo2
        .AddItem "Like"
        .AddItem "<"
        .AddItem ">"
        .AddItem ">="
        .AddItem "<="
        .AddItem "<>"
        .AddItem "="
        .ListIndex = 0
End With
```

4. ส่วนในโพรซีเจอร์ cmdSearch_Click () เริ่มต้นจะต้องสร้างเงื่อนไขของชุดคำสั่ง SQL ก่อน โดยขึ้นอยู่กับว่า ผู้ใช้จะเลือกชื่อฟิลด์ใดในคอนโทรล ComboBox1 เป็นฟิลด์เปรียบเทียบ และเลือกตัวดำเนินการใดในคอนโทรล ComboBox2 โดยอาศัยวิธีการต่อ String ให้คุณเว้นวรรคและใช้เครื่องหมายให้ถูกต้องด้วย

```
SqlTmp = Combo1 & " " & Combo2 & " '" & Text8.Text & "'"
```

5. จากนั้นจึงกำหนดให้คอนโทรล Adodc1 ดึงข้อมูลจากตาราง Customers ตามเงื่อนไขที่ตัวแปร sqlTmp เก็บไว้ คุณควรกำหนดให้คอนโทรล Adodc1 ทำการ Refresh ข้อมูลทุกครั้งด้วย ต่อมาจะสั่งให้เคอร์เซอร์วิ่งมาที่เร็คคอร์ดสุดท้าย แล้วกลับมาที่เร็คคอร์ดสุดท้าย แล้วกลับมาที่เร็คคอร์ดแรกเพื่อหาจำนวนเร็คคอร์ดที่พบตามเงื่อนไข

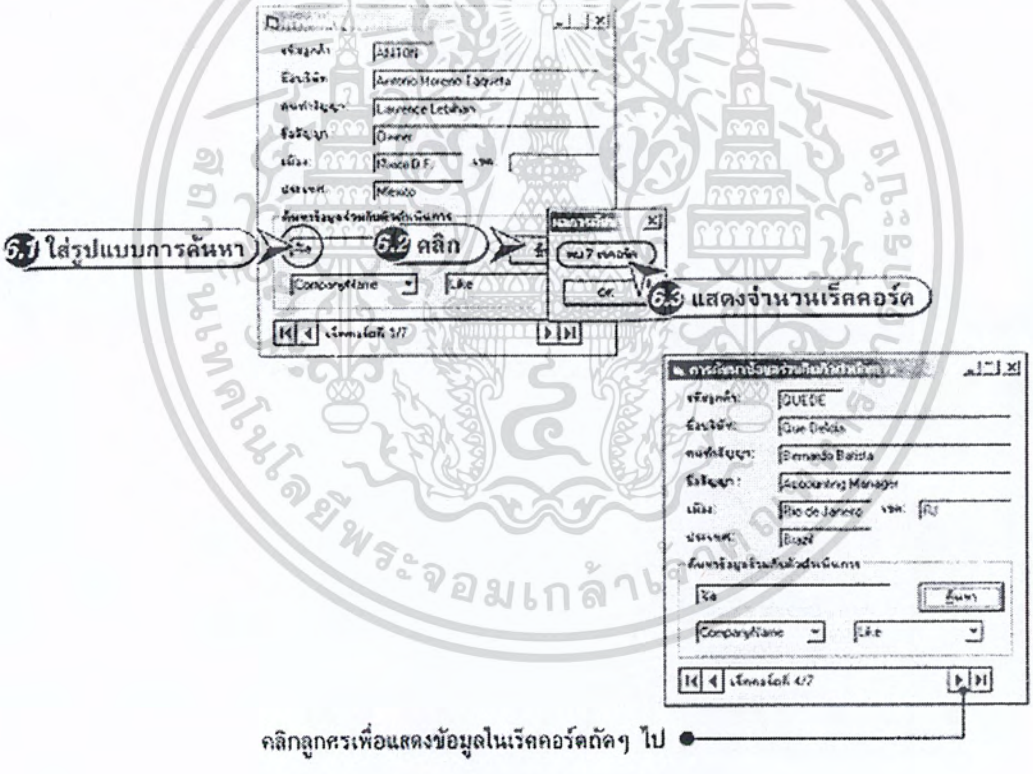
```
With Adodc1
        .RecordSourct = "Select * From Customer Where" & sqlTmp
        .Refresh
        .Recordset.MoveLast
        .Recordset.MoveFirst
End With
```

6. ขั้นตอนท้ายสุด จะแสดงจำนวนเร็คคอร์ดที่พบ

```
MsgBox "พบ" & Adodc1.Recordset.RecordCount & "เร็คคอร์ด?", vbOKOnly, "ผลการค้นหา"
```

รูปที่ 4.7 ทดลองใช้ตัวดำเนินการค้นหาข้อมูล

จากรูปที่ 4-7 ผู้เขียนทดสอบโดยการเลือกฟิลด์ Company Name และเลือกตัวดำเนินการ Like สำหรับค้นหาข้อมูล โดยระบุ %a เพื่อต้องการหาชื่อที่มีตัวอักษร a ผลการค้นหาดังรูปที่ 4-8



รูปที่ 4.8 กรณีพบเร็คคอร์ดตามเงื่อนไขที่ระบุไว้

รูปที่ 4.9 กรณีไม่พบเร็คคอร์ดตามเงื่อนไขที่ระบุไว้

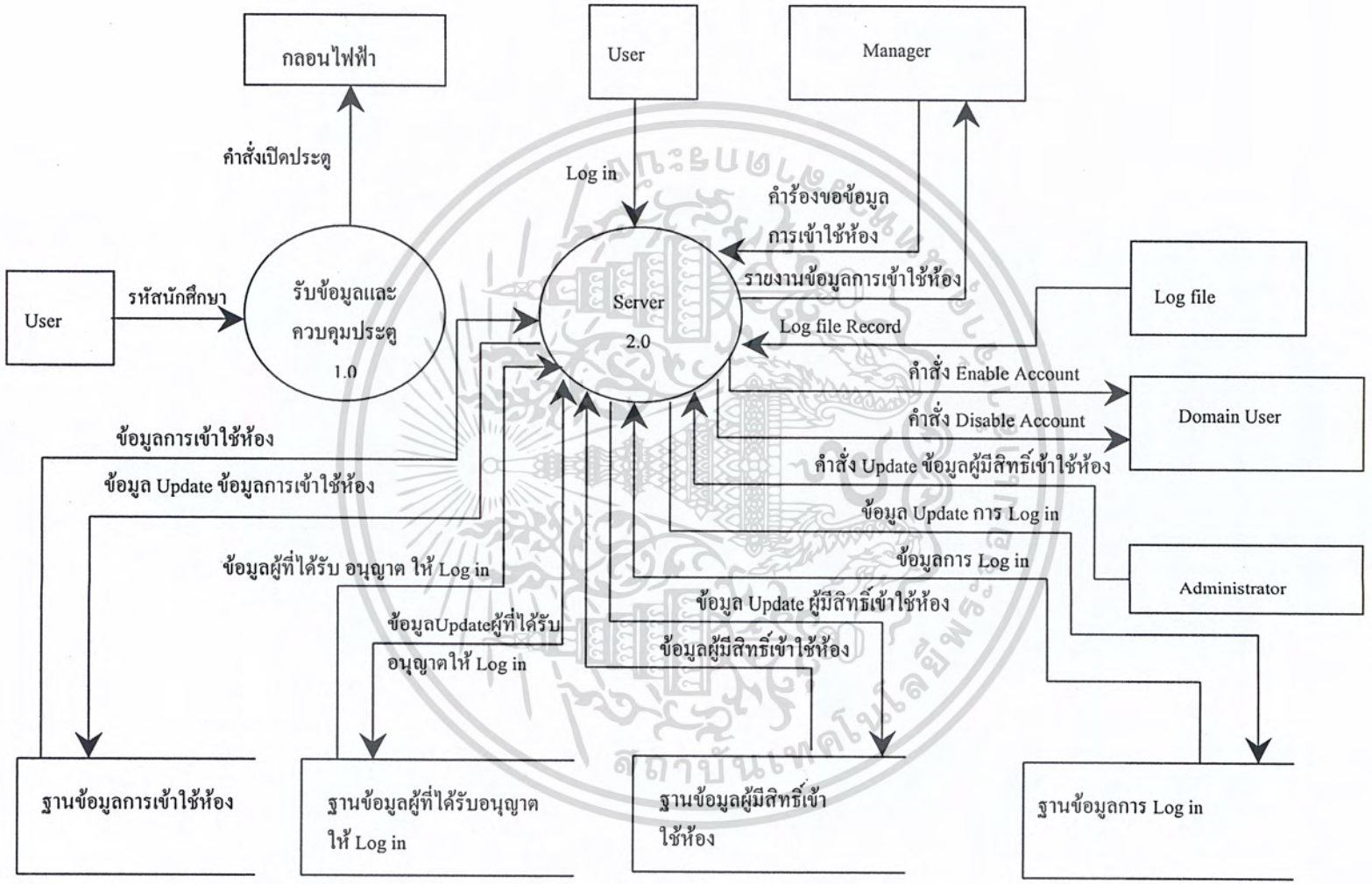
จากรูปที่ 4-9 เป็นกรณีที่ไม่มีเร็คคอร์ดใดตรงตามเงื่อนไข จะแสดงข้อความที่ Adodc 1

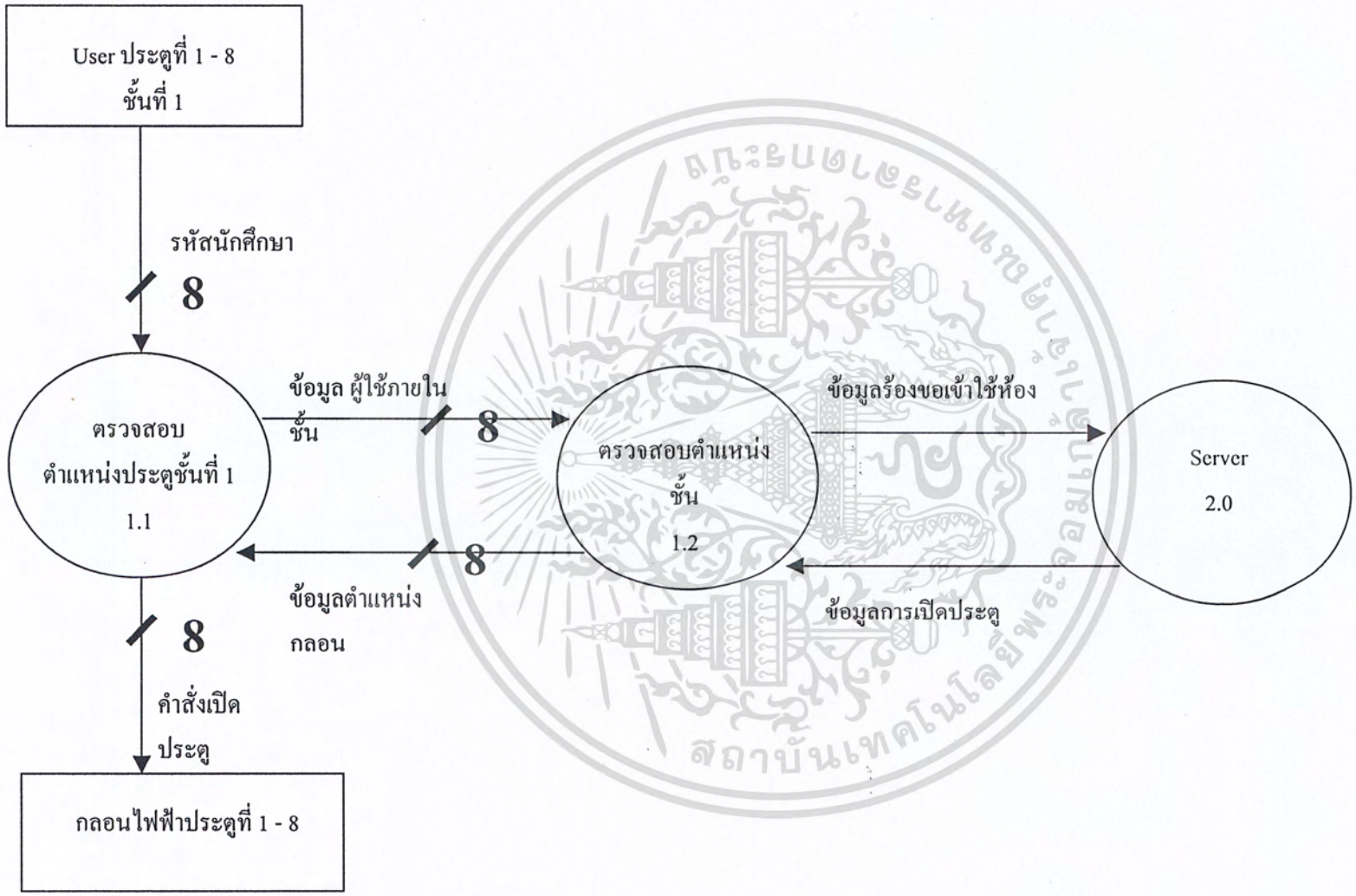
สรุปท้ายบท

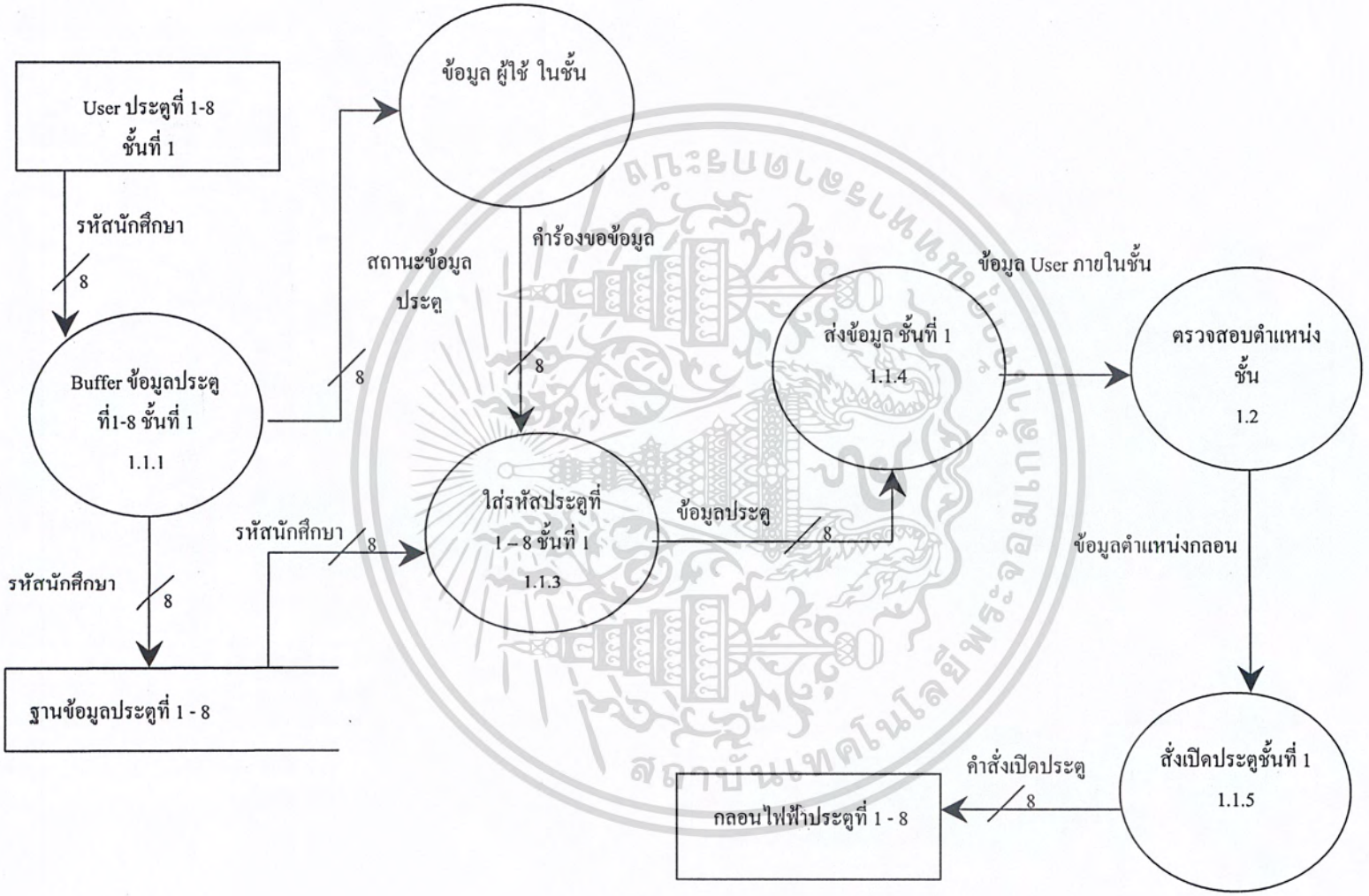
ในบทนี้เป็นการศึกษาทฤษฎีการใช้งานภาษา SQL ร่วมกับดำเนินการด้านต่างๆเพื่อนำไปใช้ในการทำคิวรีข้อมูล ซึ่งถือได้ว่าเป็นหัวข้อพื้นฐานที่คุณต้องนำไปใช้ในการเขียนโปรแกรมด้านฐานข้อมูล

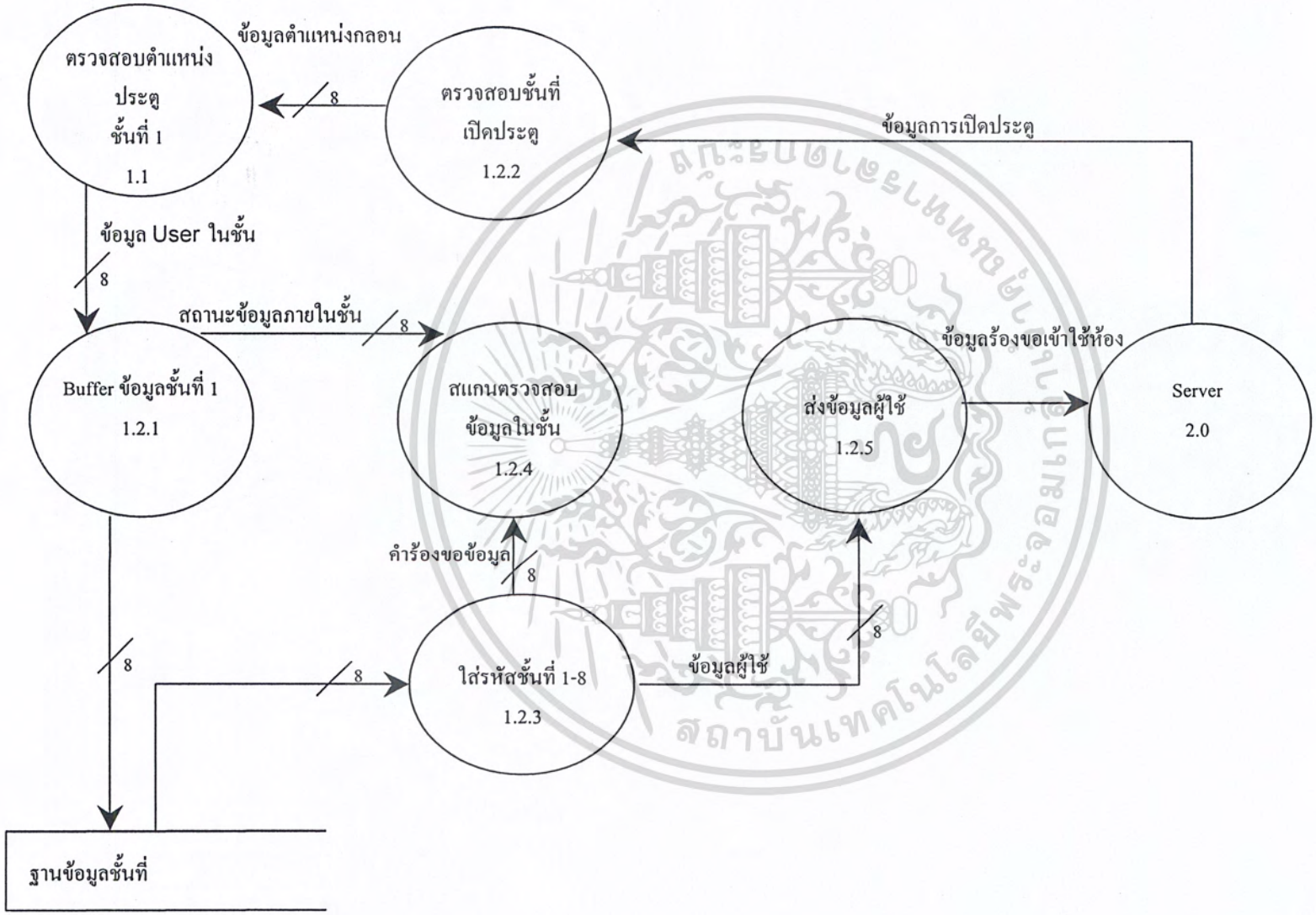
เนื่องจากว่า ภาษา SQL มีรายละเอียดค่อนข้างมาก รวมถึงมีชุดคำสั่งอื่นๆอีกพอสมควรที่ผู้เขียนมิได้กล่าวถึง ถ้าคุณต้องการศึกษาภาษา SQL เพิ่มเติม คุณสามารถศึกษาได้จากตำราที่กล่าวถึงเรื่องนี้โดยเฉพาะ ซึ่งจะอธิบายทฤษฎีและหลักการใช้งานได้ละเอียดสมบูรณ์มากกว่าผู้เขียน

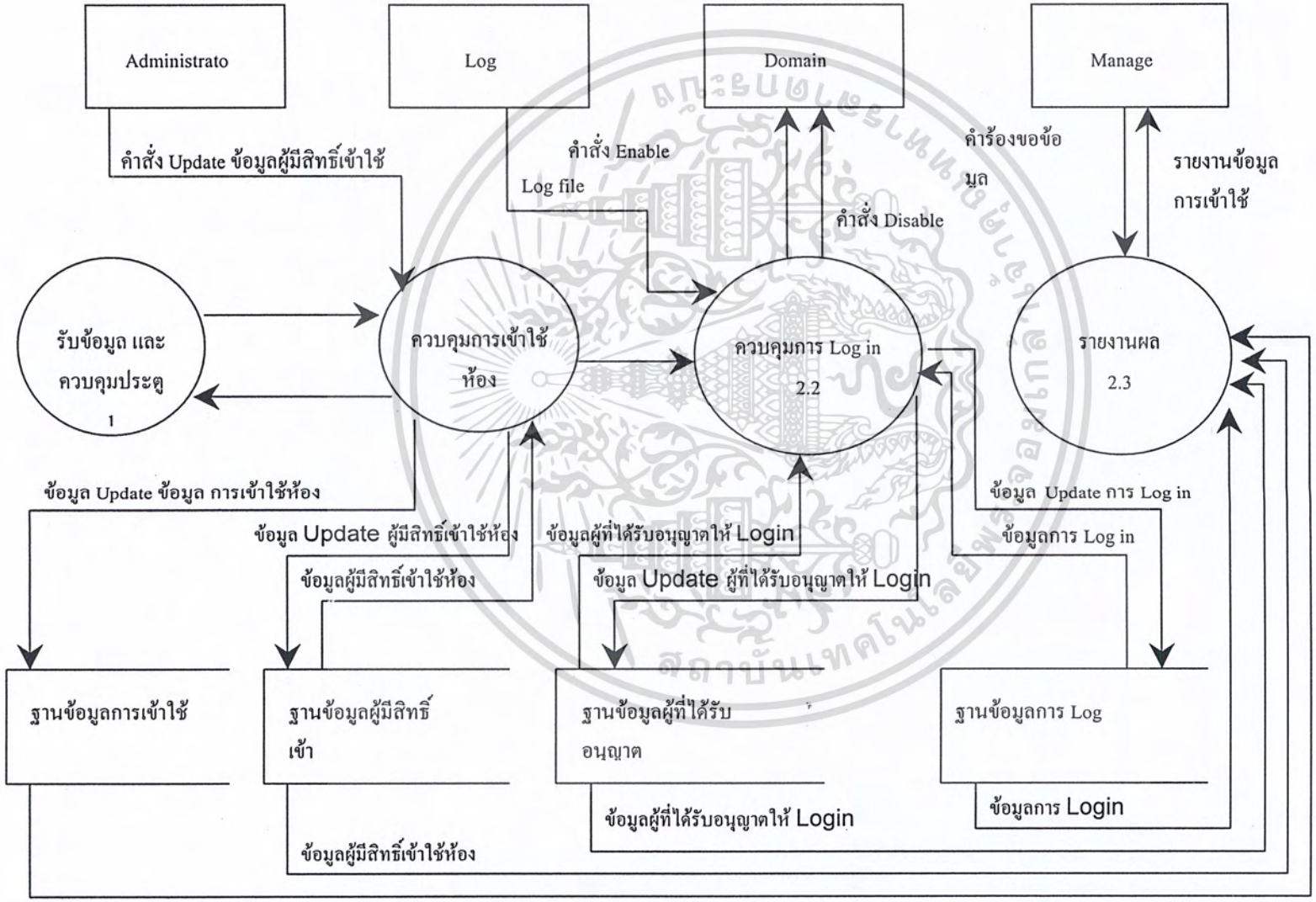
บทที่ 5 การออกแบบ

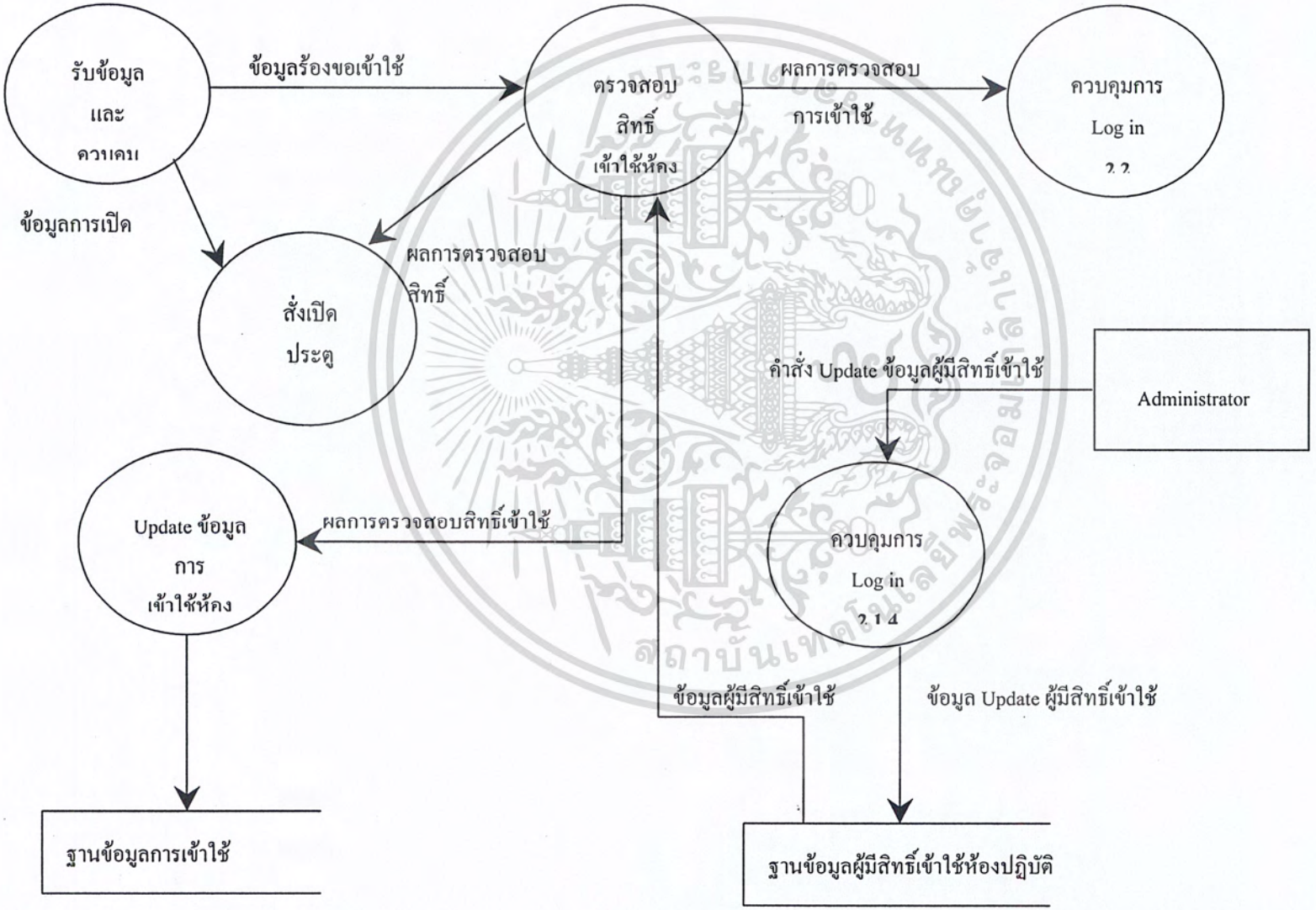

กลอนไฟฟ้า
User
Manager
คำสั่งเปิดประตู
Log in
คำร้องขอข้อมูล
การเข้าใช้ห้อง
รายงานข้อมูลการเข้าใช้ห้อง
User
รหัสนักศึกษา
รับข้อมูลและ
ควบคุมประตู
1.0
Server
2.0
Log file
Log file Record
คำสั่ง Enable Account
คำสั่ง Disable Account
Domain User
ข้อมูลการเข้าใช้ห้อง
ข้อมูล Update ข้อมูลการเข้าใช้ห้อง
คำสั่ง Update ข้อมูลผู้มีสิทธิ์เข้าใช้ห้อง
ข้อมูล Update การ Log in
ข้อมูลการ Log in
Administrator
ข้อมูลผู้ที่ได้รับ อนุญาต ให้ Log in
ข้อมูล Update ผู้มีสิทธิ์เข้าใช้ห้อง
ข้อมูลUpdateผู้ที่ได้รับ
อนุญาตให้ Log in
ข้อมูลผู้มีสิทธิ์เข้าใช้ห้อง
ฐานข้อมูลการเข้าใช้ห้อง
ฐานข้อมูลผู้ที่ได้รับอนุญาต
ให้ Log in
ฐานข้อมูลผู้มีสิทธิ์เข้า
ใช้ห้อง
ฐานข้อมูลการ Log in

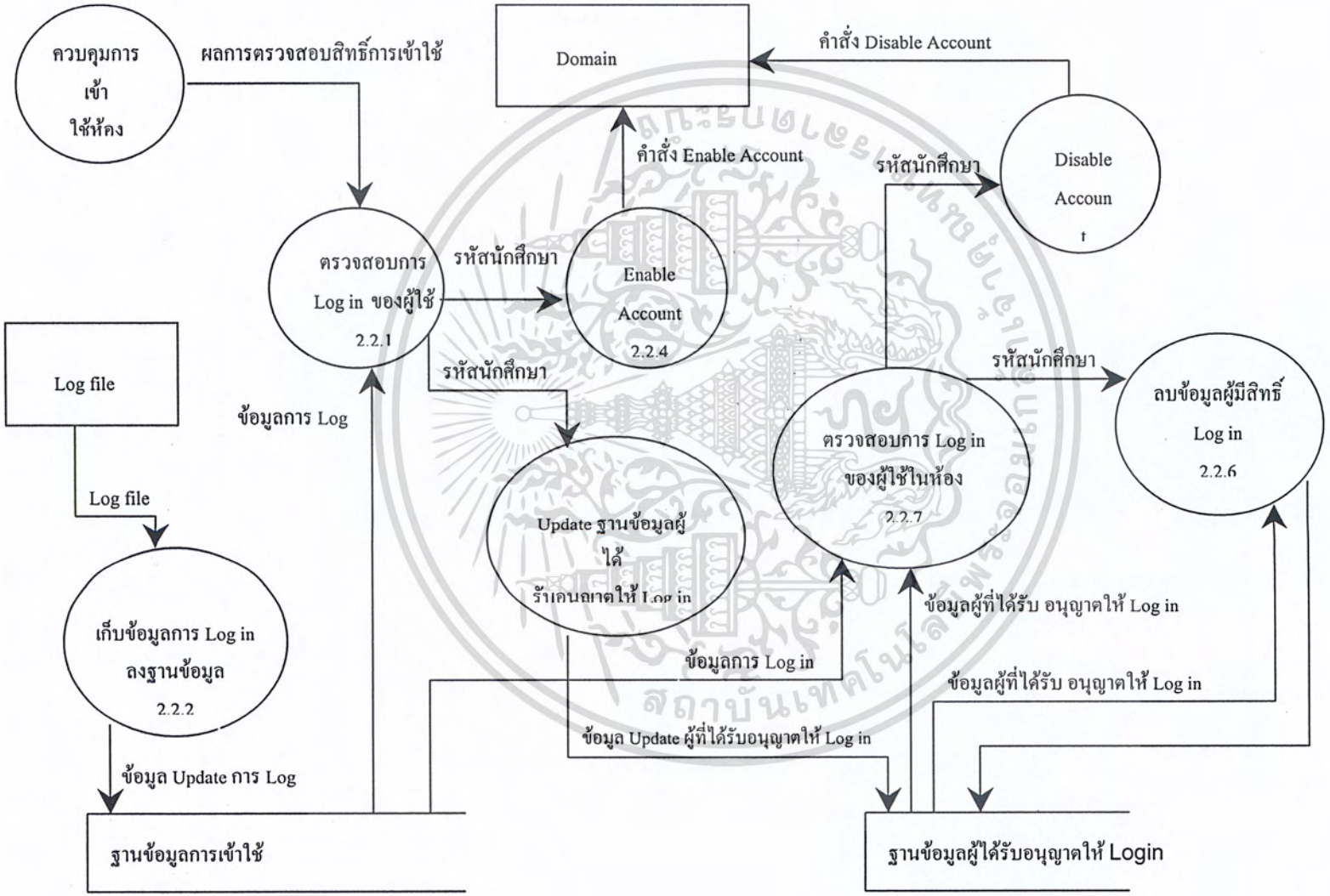

User ประตูที่ 1 - 8
ชั้นที่ 1
รหัสนักศึกษา
8
ตรวจสอบ
ตำแหน่งประตูชั้นที่ 1
1.1
ข้อมูล ผู้ใช้ภายใน
ชั้น
8
ตรวจสอบตำแหน่ง
ชั้น
1.2
ข้อมูลร้องขอเข้าใช้ห้อง
Server
2.0
8
ข้อมูลตำแหน่ง
กลอน
ข้อมูลการเปิดประตู
8
คำสั่งเปิด
ประตู
กลอนไฟฟ้าประตูที่ 1 - 8


User ประตูที่ 1-8
ชั้นที่ 1
รหัสนักศึกษา
8
Buffer ข้อมูลประตู
ที่1-8 ชั้นที่ 1
1.1.1
รหัสนักศึกษา
8
ฐานข้อมูลประตูที่ 1 - 8
ข้อมูล ผู้ใช้ ในชั้น
สถานะข้อมูล
ประตู
8
คำร้องขอข้อมูล
8
รหัสนักศึกษา
8
ใส่รหัสประตูที่
1 – 8 ชั้นที่ 1
1.1.3
ข้อมูลประตู
8
ส่งข้อมูล ชั้นที่ 1
1.1.4
ข้อมูล User ภายในชั้น
ตรวจสอบตำแหน่ง
ชั้น
1.2
ข้อมูลตำแหน่งกลอน
สั่งเปิดประตูชั้นที่ 1
1.1.5
คำสั่งเปิดประตู
8
กลอนไฟฟ้าประตูที่ 1 - 8


ตรวจสอบตำแหน่ง
ประตู
ชั้นที่ 1
1.1
ข้อมูลตำแหน่งกลอน
8
ตรวจสอบชั้นที่
เปิดประตู
1.2.2
ข้อมูลการเปิดประตู
ข้อมูล User ในชั้น
8
สถานะข้อมูลภายในชั้น
8
Buffer ข้อมูลชั้นที่ 1
1.2.1
สแกนตรวจสอบ
ข้อมูลในชั้น
1.2.4
ข้อมูลร้องขอเข้าใช้ห้อง
ส่งข้อมูลผู้ใช้
1.2.5
Server
2.0
คำร้องขอข้อมูล
8
8
8
8
ใส่รหัสชั้นที่ 1-8
1.2.3
ข้อมูลผู้ใช้
ฐานข้อมูลชั้นที่


Administrato
Log
Domain
Manage
คำสั่ง Update ข้อมูลผู้มีสิทธิ์เข้าใช้
คำสั่ง Enable
Log file
คำสั่ง Disable
คำร้องขอข้อมูล
รายงานข้อมูลการเข้าใช้
รับข้อมูล และควบคุมประตู 1
ควบคุมการเข้าใช้ห้อง
ควบคุมการ Log in 2.2
รายงานผล 2.3
ข้อมูล Update ข้อมูล การเข้าใช้ห้อง
ข้อมูล Update ผู้มีสิทธิ์เข้าใช้ห้อง
ข้อมูลผู้มีสิทธิ์เข้าใช้ห้อง
ข้อมูลผู้ที่ได้รับอนุญาตให้ Login
ข้อมูล Update ผู้ที่ได้รับอนุญาตให้ Login
ข้อมูล Update การ Log in
ข้อมูลการ Log in
ฐานข้อมูลการเข้าใช้
ฐานข้อมูลผู้มีสิทธิ์เข้า
ฐานข้อมูลผู้ที่ได้รับอนุญาต
ฐานข้อมูลการ Log
ข้อมูลผู้ที่ได้รับอนุญาตให้ Login
ข้อมูลการ Login
ข้อมูลผู้มีสิทธิ์เข้าใช้ห้อง


รับข้อมูล
และ
ควบคุม
ข้อมูลร้องขอเข้าใช้
ตรวจสอบ
สิทธิ์
เข้าใช้ห้อง
ผลการตรวจสอบ
การเข้าใช้
ควบคุมการ
Log in
2.2
ข้อมูลการเปิด
สั่งเปิด
ประตู
ผลการตรวจสอบ
สิทธิ์
คำสั่ง Update ข้อมูลผู้มีสิทธิ์เข้าใช้
Administrator
Update ข้อมูล
การ
เข้าใช้ห้อง
ผลการตรวจสอบสิทธิ์เข้าใช้
ควบคุมการ
Log in
2.1.4
ข้อมูลผู้มีสิทธิ์เข้าใช้
ข้อมูล Update ผู้มีสิทธิ์เข้าใช้
ฐานข้อมูลการเข้าใช้
ฐานข้อมูลผู้มีสิทธิ์เข้าใช้ห้องปฏิบัติ


ควบคุมการ
เข้า
ใช้ห้อง
ผลการตรวจสอบสิทธิ์การเข้าใช้
Domain
คำสั่ง Disable Account
คำสั่ง Enable Account
รหัสนักศึกษา
Disable
Accoun
t
ตรวจสอบการ
Log in ของผู้ใช้
2.2.1
รหัสนักศึกษา
Enable
Account
2.2.4
Log file
รหัสนักศึกษา
รหัสนักศึกษา
ลบข้อมูลผู้มีสิทธิ์
Log in
2.2.6
ข้อมูลการ Log
ตรวจสอบการ Log in
ของผู้ใช้ในห้อง
2.2.7
Log file
Update ฐานข้อมูลผู้
ได้
รับอนุญาตให้ Log in
ข้อมูลผู้ที่ได้รับ อนุญาตให้ Log in
เก็บข้อมูลการ Log in
ลงฐานข้อมูล
2.2.2
ข้อมูลการ Log in
ข้อมูลผู้ที่ได้รับ อนุญาตให้ Log in
ข้อมูล Update ผู้ที่ได้รับอนุญาตให้ Log in
ข้อมูล Update การ Log
ฐานข้อมูลการเข้าใช้
ฐานข้อมูลผู้ได้รับอนุญาตให้ Login

บทที่ 6

Datadic ของระบบ

PROCESS NAME:	1.0 รับข้อมูลและควบคุมประตู
DESCRIPTION:	ทำหน้าที่รับรหัสนักศึกษาจากเครื่องสแกนบาร์โค้ทมาใส่ข้อมูลของตำแหน่งประตูและตำแหน่งชั้นเพื่อส่งข้อมูลต่อไปให้ เซิร์ฟเวอร์ ทำการเช็คว่าผู้ใช้มีสิทธิ์เข้าใช้ห้องหรือไม่และจะทำการรับคำสั่งเปิดประตูจากเซิร์ฟเวอร์มาเปิดประตูในกรณีที่ผู้ใช้มีสิทธิ์เข้าใช้ห้อง
INBOUD DATA FLOWS:	- รหัสนักศึกษา - ข้อมูลการเปิดประตู
OUTBOUND DATA FLOWS:	- ข้อมูลร้องขอเข้าใช้ห้อง

PROCESS NAME:	2.0 SERVERS
DESCRIPTION:	- ทำหน้าที่ตรวจสอบว่า ผู้ใช้ ที่ทำการรูดบัตรมีสิทธิ์เข้าใช้ห้องปฏิบัติการหรือไม่ถ้ามีสิทธิ์ก็จะส่งข้อข้อมูลการเปิดประตูไปให้ Process ส่วนรับ ข้อมูลและควบคุม ประตูทำการเปิดประตู - ทำหน้าที่ดูแลการ ล็อกอิน ของ ผู้ใช้ ว่ามีสิทธิ์ ล็อกอิน หรือไม่และถ้ามีสิทธิ์ ผู้ใช้ นั้นทำการ ล็อกอิน Logout ช่วงเวลาใด ล็อกอิน มาจากเครื่องไหน - ทำหน้าที่รายงานข้อมูลการใช้ห้องปฏิบัติการของ ผู้ใช้ กับผู้บริหาร
INBOUD DATA FLOWS:	- ข้อมูลร้องขอเข้าใช้ห้อง - คำร้องขอข้อมูลการใช้ห้อง - คำสั่ง Update ข้อมูลผู้มีสิทธิ์เข้าใช้ห้องปฏิบัติการ - ข้อมูลการ ล็อกอิน - ข้อมูลผู้มีสิทธิ์เข้าใช้ห้อง - ล็อกอิน - Logfiles Record - ข้อมูลผู้ที่ได้รับอนุญาตให้ ล็อกอิน - ข้อมูลการเข้าใช้ห้อง
OUTBOUND DATA FLOWS:	- ข้อมูลการเปิดประตู

	- รายงานข้อมูลการใช้ห้อง - คำส่ง Enable Account - คำส่ง Disable Account - ข้อมูล Update การ ล็อกอิน - ข้อมูล Update ผู้มีสิทธิ์เข้าใช้ห้อง - ข้อมูล Update ผู้ที่ได้รับอนุญาตให้ ล็อกอิน - ข้อมูล Update การเข้าใช้ห้อง

PROCESS NAME:	**1.1 ตรวจสอบตำแหน่งประตู**
DESCRIPTION:	ทำหน้าที่ตรวจสอบว่า ผู้ใช้ ได้ทำการรูดบัตรจากประตูไหนแล้วทำการใส่ข้อมูลของตำแหน่งประตูเพิ่มลงไปกับรหัสนักศึกษา และทำหน้าที่รับข้อมูลตำแหน่งกลอนไฟฟ้ามาทำการเปิดประตู
INBOUD DATA FLOWS:	- รหัสนักศึกษา - ข้อมูลตำแหน่งกลอน
OUTBOUND DATA FLOWS:	- ข้อมูลผู้ใช้ภายในชั้น - คำสั่งเปิดประตู

PROCESS NAME:	**1.2 ตรวจสอบตำแหน่งชั้น**
DESCRIPTION:	ทำหน้าที่รับข้อมูลผู้ที่ทำการรูดบัตรในแต่ละชั้นเข้ามาตรวจสอบว่าข้อมูลนั้นมาจากชั้นไหนและทำการเพิ่มข้อมูลตำแหน่งชั้นลงไปกับข้อมูล อีกทั้งยังทำหน้าที่รับข้อมูลการเปิดประตู ตรวจสอบว่าคำสั่งนั้นต้องการเปิดประตูที่ชั้นไหนและทำการส่งข้อมูลตำแหน่งกลอนไฟฟ้าออกไปเพื่อทำการเปิดประตู
INBOUD DATA FLOWS:	- ข้อมูลผู้ใช้ภายในชั้น - ข้อมูลการเปิดประตู
OUTBOUND DATA FLOWS:	- ข้อมูลร้องขอเข้าใช้ห้อง - ข้อมูลตำแหน่งกลอน

PROCESS NAME:	1.1.1 ข้อมูลประตูที่ 1-8 ชั้นที่ 1
DESCRIPTION:	- ทำหน้าที่รับข้อมูลรหัสนักศึกษามาจาก ผู้ใช้ และทำการนำมาเก็บลงใน Buffer ไว้ ก่อนเพื่อรอการส่งต่อ เพื่อป้องกันปัญหาการรูดบัตรพร้อมกันหลายๆประตูแล้วเครื่องทำการรับข้อมูลไม่ทัน - ทำหน้าที่ส่งสถานะข้อมูลว่าขณะนี้มีข้อมูลอยู่ใน Buffer อยู่หรือไม่
INBOUD DATA FLOWS:	- รหัสนักศึกษา
OUTBOUND DATA FLOWS:	- รหัสนักศึกษา - สถานะข้อมูลประตู

PROCESS NAME:	1.1.2 สแกนตรวจสอบข้อมูลประตูชั้นที่ 1
DESCRIPTION:	ทำหน้าที่ตรวจสอบข้อมูลประตูต่างๆว่ามีข้อมูลอยู่หรือไม่ ถ้ามีข้อมูลอยู่ Process ก็จะทำการส่งคำร้องขอข้อมูลออกไป และจะทำการหน่วงเวลาอยู่ช่วงเวลาหนึ่งจนกว่าจะส่งข้อมูลเสร็จ แล้วถึงจะเริ่มสแกนประตูต่อไป
INBOUD DATA FLOWS:	- สถานะข้อมูลประตู
OUTBOUND DATA FLOWS:	- คำร้องขอข้อมูล

PROCESS NAME:	1.1.3 ใส่รหัสประตูที่ 1-8 ชั้นที่1
DESCRIPTION:	ทำหน้าที่รับข้อมูลรหัสนักศึกษาเข้ามาแล้วทำการใส่รหัสประตูเพิ่มเข้าไปแล้วทำการส่งข้อมูลที่ได้ต่อไป
INBOUD DATA FLOWS:	- รหัสนักศึกษา - คำร้องขอข้อมูล
OUTBOUND DATA FLOWS:	- ข้อมูลประตู

PROCESS NAME:	1.1.4 ส่งข้อมูลชั้นที่ 1
DESCRIPTION:	ทำหน้าที่รับข้อมูลของแต่ละประตูแล้วทำการส่งต่อออกไปทีละประตู
INBOUD DATA FLOWS:	- ข้อมูลประตู
OUTBOUND DATA FLOWS:	- ข้อมูล ผู้ใช้ ภายในชั้น

PROCESS NAME:	1.1.5 สั่งเปิดประตูชั้นที่ 1
DESCRIPTION:	ทำหน้าที่รับข้อมูลตำแหน่งกลอนไฟฟ้ามาตรวจสอบว่าอยู่ประตูที่เท่าไหร่แล้วทำการส่งเปิดประตู
INBOUD DATA FLOWS:	- ข้อมูลตำแหน่งกลอน
OUTBOUND DATA FLOWS:	- คำสั่งเปิดประตู

PROCESS NAME:	1.2.1 Buffer ข้อมูลชั้นที่ 1
DESCRIPTION:	- ทำหน้าที่รับข้อมูลในแต่ละชั้นมา และทำการนำมาเก็บลงใน Buffer ไว้ก่อนเพื่อรอการส่งต่อ - ทำหน้าที่ส่งสถานะข้อมูลภายในชั้นว่าขณะนี้มีข้อมูลอยู่ใน Buffer อยู่หรือไม่
INBOUD DATA FLOWS:	- ข้อมูลในชั้น
OUTBOUND DATA FLOWS:	- ข้อมูลในชั้น - สถานะข้อมูลภายในชั้น

PROCESS NAME:	1.2.2 ตรวจสอบชั้นที่เปิดประตู
DESCRIPTION:	ทำหน้าที่รับข้อมูลการเปิดประตูเข้ามาตรวจสอบว่าต้องการเปิดประตูชั้นไหน จากนั้น Process ก็จะส่งข้อมูล ตำแหน่งกลอนไฟฟ้าไปตามชั้นที่ระบุ
INBOUD DATA FLOWS:	- ข้อมูลการเปิดประตู
OUTBOUND DATA FLOWS:	- ข้อมูลตำแหน่งกลอน

PROCESS NAME:	1.2.3 ใส่รหัสชั้นที่1-8
DESCRIPTION:	ทำหน้าที่รับข้อมูลในชั้นเข้ามาใส่รหัสชั้นเพิ่มเข้าไปแล้วทำการส่งต่อ
INBOUD DATA FLOWS:	- ข้อมูลในชั้น - คำร้องขอข้อมูล
OUTBOUND DATA FLOWS:	- ข้อมูลผู้ใช้

PROCESS NAME:	1.2.4 สแกนตรวจสอบข้อมูลในชั้น
DESCRIPTION:	ทำหน้าที่สแกนข้อมูลในแต่ละชั้นว่ามีข้อมูลอยู่หรือไม่ถ้ามีข้อมูลอยู่ Process จะทำการส่งคำร้องขอข้อมูลออกไป
INBOUD DATA FLOWS:	- สถานะข้อมูลภายในชั้น
OUTBOUND DATA FLOWS:	- คำร้องขอข้อมูล

PROCESS NAME:	1.2.5 ส่งข้อมูลผู้ใช้
DESCRIPTION:	ทำหน้าที่รับข้อมูลผู้ขอเข้าใช้ห้องปฏิบัติการของแต่ละชั้นมาทำการส่งต่อ โดยส่งทีละชั้น
INBOUD DATA FLOWS:	- ข้อมูลผู้ใช้
OUTBOUND DATA FLOWS:	- ข้อมูลร้องขอข้าใช้ห้อง

PROCESS NAME:	2.1 ควบคุมการเข้าใช้ห้อง
DESCRIPTION:	ทำหน้าที่ควบคุมการเข้าใช้ห้องปฏิบัติการของ ผู้ใช้ ว่ามีสิทธิ์เข้าใช้ห้องหรือไม่ โดยรับข้อมูลการร้องขอเข้าใช้ห้อง จาก Process รับข้อมูลและควบคุมประตู มาทำการตรวจสอบสิทธิ์ ถ้า ผู้ใช้ มีสิทธิ์เข้าใช้ห้องก็จะทำการส่งข้อมูลการเปิดประตูไปให้ Process รับข้อมูลและควบคุมประตู ทำการเปิดประตู จากนั้น Process จะส่งผลการตรวจสอบไปให้ Process ควบคุมการ ล็อกอิน เพื่อทำการ Enable การ ล็อกอิน ให้ ผู้ใช้ ต่อไป อีกทั้งยังทำหน้าที่ Update ข้อมูลของผู้ที่มีสิทธิ์เข้าใช้ห้องปฏิบัติการ
INBOUD DATA FLOWS:	- ข้อมูลร้องขอเข้าใช้ห้อง - คำสั่ง Update ข้อมูลผู้มีสิทธ์เข้าใช้ห้องปฏิบัติการ - ข้อมูลผู้มีสิทธิ์เข้าใช้ห้อง
OUTBOUND DATA FLOWS:	- ข้อมูลการเปิดประตู - ผลการตรวจสอบสิทธิ์การเข้าใช้ห้อง - ข้อมูล Update ผู้มีสิทธิ์เข้าใช้ห้อง - ข้อมูล Update การเข้าใช้ห้อง

PROCESS NAME:	2.2 ควบคุมการ ล็อกอิน
DESCRIPTION:	ทำหน้าที่รับข้อมูลผลการตรวจสอบสิทธิ์การเข้าใช้ห้องของ ผู้ใช้ มาตรวจสอบว่า ผู้ใช้ คนนั้น ล็อกอิน อยู่หรือเปล่าถ้า ล็อกอิน อยู่(ในกรณีที่ ผู้ใช้ มีความจำเป็นต้องทิ้งเครื่องที่ใช้งานไปชั่วขณะเช่น ไปเข้าห้องน้ำ) ก็ไม่ต้องสนใจแต่ถ้า ผู้ใช้ ไม่ได้ทำการ ล็อกอิน อยู่ Process ก็จะทำการ Enable Account ให้กับ ผู้ใช้ อีกทั้งในช่วงเวลาหนึ่งๆระบบยังทำหน้าที่ดึงข้อมูลการ ล็อกอิน มาจาก Logfile ลงในฐานข้อมูล เพื่อทำการตรวจสอบว่า ผู้ใช้ ยังใช้เครื่องอยู่หรือเปล่าถ้า ผู้ใช้ ไม่ได้ใช้เครื่องแล้วระบบก็จะทำการ Disable Account ของ ผู้ใช้ และตรวจสอบว่า ผู้ใช้ Account ที่ทำการ Enable Account ไปแล้วนั้นได้ทำการ ล็อกอิน ไปแล้วหรือยังถ้ายังระบบจะทำการ Disable Account เพื่อป้องกันการรูดบัตรทิ้งไว้ให้ Account Enable แล้วให้เพื่อนเข้าห้องมา ล็อกอิน ภายหลัง
INBOUD DATA FLOWS:	- Logfile - ข้อมูลการ ล็อกอิน - ข้อมูลผู้ที่ได้รับอนุญาตให้ ล็อกอิน - ผลการตรวจสอบสิทธิ์เข้าใช้ห้องปฏิบัติการ
OUTBOUND DATA FLOWS:	- คำสั่ง Enable Account - คำสั่ง Disable Account - ข้อมูล Update การ ล็อกอิน - ข้อมูล Update ผู้ที่ได้รับอนุญาตให้ ล็อกอิน

PROCESS NAME:	2.3 รายงานผล
DESCRIPTION:	ทำหน้าที่รับคำร้องขอข้อมูลการใช้ห้องจากผู้บริหารมาทำการดึงข้อมูลจากฐานข้อมูลต่างๆมาออกรายงานให้กับผู้บริหาร
INBOUD DATA FLOWS:	- คำร้องขอข้อมูลการใช้ห้อง - ข้อมูลการ ล็อกอิน - ผู้มีสิทธิ์เข้าใช้ห้องปฏิบัติการ - ข้อมูลผู้ที่ได้รับอนุญาตให้ ล็อกอิน - ข้อมูลการเข้าใช้ห้อง
OUTBOUND DATA FLOWS:	- รายงานข้อมูลการใช้ห้อง

PROCESS NAME:	2.1.1 ตรวจสอบสิทธิ์เข้าใช้ห้อง
DESCRIPTION:	ทำหน้าที่รับข้อมูลร้องขอเข้าใช้ห้องจาก Process รับข้อมูลและควบคุมประตู มาทำการตรวจสอบว่า ผู้ใช้ มีสิทธิ์เข้าใช้ห้องปฏิบัติการหรือไม่
INBOUD DATA FLOWS:	- ข้อมูลร้องขอเข้าใช้ห้อง - ข้อมูลผู้มีสิทธิ์เข้าใช้ห้อง
OUTBOUND DATA FLOWS:	- ผลการตรวจสอบสิทธิ์การเข้าใช้ห้อง

PROCESS NAME:	2.1.2 สั่งเปิดประตู
DESCRIPTION:	ทำหน้าที่รับผลการตรวจสอบสิทธิ์การเข้าใช้ห้องของ ผู้ใช้ มาทำการสั่งเปิดประตูให้ ผู้ใช้ เข้าใช้ห้องปฏิบัติการ
INBOUD DATA FLOWS:	- ผลการตรวจสอบสิทธิ์การเข้าใช้ห้อง
OUTBOUND DATA FLOWS:	- ข้อมูลการเปิดประตู

PROCESS NAME:	2.1.3 Update ข้อมูลการเข้าใช้ห้อง
DESCRIPTION:	ทำหน้าที่รับผลการตรวจสอบสิทธิ์การเข้าใช้ห้องมาทำการเก็บข้อมูลการเข้าใช้งานห้องปฏิบัติการลงฐานข้อมูล
INBOUD DATA FLOWS:	- ผลการตรวจสอบสิทธิ์การเข้าใช้ห้อง
OUTBOUND DATA FLOWS:	- ข้อมูล Update การเข้าใช้ห้อง

PROCESS NAME:	2.1.4 Update ข้อมูลผู้มีสิทธิ์เข้าใช้ห้องปฏิบัติการ
DESCRIPTION:	ทำหน้าที่ Update ข้อมูลผู้มีสิทธิ์เข้าใช้ห้องปฏิบัติการ
INBOUD DATA FLOWS:	- คำสั่ง Update ข้อมูลผู้มีสิทธิ์เข้าใช้ห้องปฏิบัติการ
OUTBOUND DATA FLOWS:	- ข้อมูลผู้มีสิทธิ์เข้าใช้ห้อง

PROCESS NAME:	2.2.1 ตรวจสอบการ Login ของ User
DESCRIPTION:	ทำการตรวจสอบว่า ผู้ใช้ ที่ได้รับอนุญาตให้เข้าใช้ห้องปฏิบัติการนั้นได้ทำการ ล็อกอิน อยู่แล้วหรือไม่ ถ้า ล็อกอิน อยู่(ในกรณีที่ ผู้ใช้ มีความจำเป็นต้องทิ้งเครื่องที่ใช้งานไปชั่วขณะเช่น ไปเข้าห้องน้ำ) ก็ไม่ต้องสนใจ แต่ถ้า ผู้ใช้ ไม่ได้ทำการ ล็อกอิน อยู่ Process ก็จะทำการส่งรหัสนักศึกษาไปให้ Process enable account และ Process Update ฐานข้อมูลผู้ที่ได้รับอนุญาตให้ ล็อกอิน ทำการ Enable Account และ Update ฐานข้อมูลผู้ที่ได้รับอนุญาตให้ ล็อกอิน ตามลำดับ
INBOUD DATA FLOWS:	- ผลการตรวจสอบสิทธิ์การเข้าใช้ห้อง - ข้อมูลการ ล็อกอิน
OUTBOUND DATA FLOWS:	- รหัสนักศึกษา

:

PROCESS NAME:	2.2.2 เก็บข้อมูลการ Login ลงฐานข้อมูล
DESCRIPTION:	ทำหน้าที่ดึง Logfileเข้ามาเก็บข้อมูลการล็อกอิน ลงฐานข้อมูลการล็อกอิน
INBOUD DATA FLOWS:	- Logfile
OUTBOUND DATA FLOWS:	- ข้อมูล Update การ ล็อกอิน

PROCESS NAME:	2.2.3 Update ฐานข้อมูลผู้ที่ได้รับอนุญาตให้ ล็อกอิน
DESCRIPTION:	ทำหน้าที่รับรหัสนักศึกษาที่ผ่านการตรวจสอบสิทธิ์มาทำการ Update ฐานข้อมูลผู้ที่ได้รับอนุญาตให้ ล็อกอิน
INBOUD DATA FLOWS:	- รหัสนักศึกษา
OUTBOUND DATA FLOWS:	- ข้อมูล Update ผู้ที่ได้รับอนุญาตให้ ล็อกอิน

PROCESS NAME:	2.2.4 Enable Account
DESCRIPTION:	ทำหน้าที่รับรหัสนักศึกษามาทำการ Enable Account
INBOUD DATA FLOWS:	- รหัสนักศึกษา
OUTBOUND DATA FLOWS:	- คำสั่ง Enable Account

PROCESS NAME:	2.2.5 Disable Account
DESCRIPTION:	ทำหน้าที่รับรหัสนักศึกษามาทำการ Disable Account
INBOUD DATA FLOWS:	- รหัสนักศึกษา
OUTBOUND DATA FLOWS:	- คำสั่ง Disable Account

:

PROCESS NAME:	2.2.6 Delete ข้อมูลผู้มีสิทธิ์ Login
DESCRIPTION:	ทำหน้าที่รับรหัสนักศึกษามาทำการลบข้อมูลผู้มีสิทธิ์ ล็อกอิน
INBOUD DATA FLOWS:	- รหัสนักศึกษา - ข้อมูลผู้ที่ได้รับอนุญาตให้ ล็อกอิน
OUTBOUND DATA FLOWS:	- ข้อมูล Update ผู้ที่ได้รับอนุญาตให้ ล็อกอิน

DATA FLOW NAME:	รหัสนักศึกษา
DESCRIPTION:	รหัสประจำตัวนักศึกษาที่ทางสถาบันเป็นผู้ออกให้
FROM PROCESS:	2.2.1 ตรวจสอบการ ล็อกอิน ของ User
TO PROCESS:	2.2.4 Enable Account
FROM PROCESS:	2.2.7 ตรวจสอบการ ล็อกอิน ของ User ในห้องปฏิบัติการ
TO PROCESS:	2.2.5 Disable Account
FROM PROCESS:	2.2.7 ตรวจสอบการ ล็อกอิน ของ User ในห้องปฏิบัติการ
TO PROCESS:	2.2.6 Delete ข้อมูลผู้มีสิทธิ์ ล็อกอิน
DATA STRUCTURES:	

DATA FLOW NAME:	**ข้อมูลร้องขอเข้าใช้ห้อง**
DESCRIPTION:	เป็นข้อมูลที่ระบุว่า ผู้ใช้ ทำการรูดบัตรจากประตูไหนชั้นไหน
FROM PROCESS:	1.0 ข้อมูลร้องขอเข้าใช้ห้อง
TO PROCESS:	2.1.1 ตรวจสอบสิทธิ์เข้าใช้ห้อง
DATA STRUCTURES:	ข้อมูลร้องขอเข้าใช้ห้อง - รหัสนักศึกษา - ตำแหน่งประตู - ตำแหน่งชั้น

DATA FLOW NAME:	ผลการตรวจสอบสิทธิ์การเข้าใช้ห้อง
DESCRIPTION:	เป็นข้อมูลที่บอกว่า ผู้ใช้ ทีระบุมีสิทธิ์เข้าใช้ห้องหรือหรือไม่
FROM PROCESS:	2.1.1 ตรวจสอบสิทธิ์เข้าใช้ห้อง
TO PROCESS:	2.2 ควบคุมการ Login
FROM PROCESS:	2.1.1 ตรวจสอบสิทธิ์เข้าใช้ห้อง
TO PROCESS:	2.1.3 Update ข้อมูลการเข้าใช้ห้อง
FROM PROCESS:	2.1.1 ตรวจสอบสิทธิ์เข้าใช้ห้อง
TO PROCESS:	2.1.2 สั่งเปิดประตู
DATA STRUCTURES:	ผลการตรวจสอบสิทธิ์การเข้าใช้ห้อง - รหัสนักศึกษา - STATUS

DATA FLOW NAME:	ข้อมูลการเปิดประตู
DESCRIPTION:	เป็นข้อมูลที่บอกว่าต้องการสั่งเปิดประตูใด
FROM PROCESS:	2.1.2 สั่งเปิดประตู
TO PROCESS:	1.0 รับข้อมูลและควบคุมประตู
DATA STRUCTURES:	ข้อมูลการเปิดประตู - ตำแน่งประตู - ตำแหน่งชั้น

DATA FLOW NAME:	**คำสั่ง Update ข้อมูลผู้มีสิทธิ์เข้าใช้ห้องปฏิบัติการ**
DESCRIPTION:	เป็นคำสั่งจัดการกับฐานข้อมูลผู้มีสิทธิ์เข้าใช้ห้องปฏิบัติการ
FROM PROCESS:	Administrator
TO PROCESS:	Update ข้อมูลผู้มีสิทธิ์เข้าใช้ห้องปฏิบัติการ
DATA STRUCTURES:	คำสั่ง Update ข้อมูลผู้มีสิทธิ์เข้าใช้ห้องปฏิบัติการ - Database Operation - รหัสนักศึกษา

DATA FLOW NAME:	**ข้อมูล Update ผู้มีสิทธิ์เข้าใช้ห้อง**
DESCRIPTION:	เป็นข้อมูล ผู้มีสิทธิ์เข้าใช้ห้องที่ต้องการ Update
FROM PROCESS:	2.1.4 Update ข้อมูลผู้มีสิทธิ์เข้าใช้ห้องปฏิบัติการ
TO DATA STORE:	ฐานข้อมูลผู้มีสิทธิ์เข้าใช้ห้องปฏิบัติการ
DATA STRUCTURES:	ข้อมูล Update ผู้มีสิทธิ์เข้าใช้ห้อง - รหัสนักศึกษา - STATUS

DATA FLOW NAME:	**ข้อมูลผู้มีสิทธิ์เข้าใช้ห้อง**
DESCRIPTION:	เป็นข้อมูลผู้ที่มีสิทธิ์เข้าใช้ห้องปฏิบัติการ
FROM DATA STORE:	ฐานข้อมูลผู้มีสิทธิ์เข้าใช้ห้องปฏิบัติการ
TO PROCESS:	2.1.1 ตรวจสอบสิทธิ์เข้าใช้ห้อง
DATA STRUCTURES:	ข้อมูลผู้มีสิทธิ์เข้าใช้ห้อง - รหัสนักศึกษา - STATUS

DATA FLOW NAME:	**Logfile Record**
DESCRIPTION:	เป็น Logfile Security ของ เซิร์ฟเวอร์
FROM PROCESS:	Logfile
TO PROCESS:	2.2.2 เก็บข้อมูลการ Login ลงฐานข้อมูล
DATA STRUCTURES:	Logfile Record - Length - Reserved - RecordNumber - TimeGenerated - TimeWritten - EventID - EventType - NumStrings - EventCategory - ReservedFlags - ClosingRecordNumber - StringOffset - UserSidLength - UserSidOffset - DataLength - DataOffset

DATA FLOW NAME:	**ข้อมูล Update การ Login**
DESCRIPTION:	เป็นข้อมูลการ ล็อกอิน ของ ผู้ใช้ ที่จะ Update ลงฐานข้อมูล
FROM PROCESS:	2.2.2 เก็บข้อมูลการ Login ลงฐานข้อมูล
TO DATA STORE:	ฐานข้อมูลการ ล็อกอิน
DATA STRUCTURES:	ข้อมูล Update การ ล็อกอิน - รหัสนักศึกษา - Login time - Logout time - ล็อกอิน จากเครื่อง

DATA FLOW NAME:	ข้อมูลการ Login
DESCRIPTION:	เป็นข้อมูลการ ล็อกอิน ที่บอกว่า ผู้ใช้ ล็อกอิน Logout เวลาใดจากเครื่องไหน
FROM DATA STORE:	ฐานข้อมูลการ ล็อกอิน
TO PROCESS:	2.2.1 ตรวจสอบการ Login ของ User
DATA STRUCTURES:	ข้อมูลการ ล็อกอิน - รหัสนักศึกษา - Login time - Logout time - Login จากเครื่อง

DATA FLOW NAME:	ข้อมูล Update ผู้ที่ได้รับอนุญาตให้ Login
DESCRIPTION:	เป็นข้อมูลของผู้ที่ได้รับอนุญาตให้ ล็อกอิน ที่จะทำการ Update ลงฐานข้อมูล
FROM PROCESS:	2.2.3 Update ฐานข้อมูลผู้ที่ได้รับอนุญาตให้ Login 2.2.6 Delete ข้อมูลผู้มีสิทธิ์ Login
TO DATA STORE:	ฐานข้อมูลผู้ที่ได้รับอนุญาตให้ ล็อกอิน
DATA STRUCTURES:	ข้อมูล Update ผู้ที่ได้รับอนุญาตให้ ล็อกอิน - รหัสนักศึกษา - Enable Login Time

DATA FLOW NAME:	ข้อมูลผู้ที่ได้รับอนุญาตให้ Login
DESCRIPTION:	เป็นข้อมูลของผู้ที่ได้รับอนุญาตให้ ล็อกอิน
FROM DATA STORE:	ฐานข้อมูลผู้ที่ได้รับอนุญาตให้ ล็อกอิน
TO PROCESS:	2.2.7 ตรวจสอบการ Login ของ User ในห้องปฏิบัติการ 2.2.6 Delete ข้อมูลผู้มีสิทธิ์ Login
DATA STRUCTURES:	ข้อมูล Update ผู้ที่ได้รับอนุญาตให้ ล็อกอิน - รหัสนักศึกษา - Enable Login Time

DATA FLOW NAME:	**คำสั่ง Enable account**
DESCRIPTION:	
FROM PROCESS:	2.2.4 Enable Account
TO DESTINATION:	DOMAIN USER
DATA STRUCTURES:	คำสั่ง Enable account - รหัสนักศึกษา - Diaable account status

DATA FLOW NAME:	**คำสั่ง Disable account**
DESCRIPTION:	
FROM PROCESS:	2.2.5 Disable Account
TO DESTINATION:	Domain User
DATA STRUCTURES:	คำสั่ง Enable account - รหัสนักศึกษา - Disable account status

DATA FLOW NAME:	**คำร้องขอข้อมูลการเข้าใช้ห้อง**
DESCRIPTION:	เป็นข้อมูลที่ผู้บริหารให้กับระบบว่า ผู้บริหารต้องการข้อมูลอะไร
FROM SOURCE:	Manager
TO PROCESS:	2.3 รายงานผล
DATA STRUCTURES:	คำร้องขอข้อมูลการเข้าใช้ห้อง - รหัสนักศึกษา - ชื่อเครื่อง - ช่วงวันที่ - ช่วงเวลา

DATA FLOW NAME:	**รายงานข้อมูลการใช้ห้อง**
DESCRIPTION:	เป็นข้อมูลการใช้ห้องที่รายงานให้กับผู้บริหาร
FROM PROCESS:	2.3 รายงานผล
TO DESTINATION:	Manager
DATA STRUCTURES:	รายงานข้อมูลการใช้ห้อง - รหัสนักศึกษา - ชื่อเครื่อง - เวลา ล็อกอิน - เวลา Logout - เวลาเข้าใช้ห้อง

DATA FLOW NAME:	**สถานะข้อมูลประตู**
DESCRIPTION:	เป็นข้อมูลที่บอกว่าตอนนี้มีข้อมูลอยู่ใน Bufer หรือไม่
FROM PROCESS:	1.1.1 Buffer ข้อมูลประตูที่ 1-8 ชั้นที่ 1
TO DESTINATION:	1.1.2 สแกนตรวจสอบข้อมูลประตูชั้นที่1
DATA STRUCTURES:	Status

DATA FLOW NAME:	**คำร้องขอข้อมูล**
DESCRIPTION:	ทำหน้าที่บอกว่าตอนนี้ต้องการให้ Process ที่รับสัญญาณนี้ไปทำการส่งข้อมูล
FROM PROCESS:	1.1.2 สแกนตรวจสอบข้อมูลประตูชั้นที่ 1
TO DESTINATION:	1.1.3 ใส่รหัสประตูที่ 1-8 ชั้นที่ 1
DATA STRUCTURES:	Status

DATA FLOW NAME:	ข้อมูลประตู
DESCRIPTION:	ข้อมูลรหัสนักศึกษาที่ผ่านการใส่รหัสประตูเพิ่มเข้าไปแล้ว
FROM PROCESS:	1.1.3 ใส่รหัสประตูที่ 1-8
TO DESTINATION:	1.1.4 ส่งข้อมูลชั้นที่ 1
DATA STRUCTURES:	ข้อมูลประตู - รหัสนักศึกษา - ตำแหน่งประตู

DATA FLOW NAME:	ข้อมูล ผู้ใช้ ภายในชั้น
DESCRIPTION:	ข้อมูลรหัสนักศึกษาที่ผ่านการใส่รหัสประตูเพิ่มเข้าไปแล้ว
FROM PROCESS:	1.1.3 ใส่รหัสประตูที่ 1-8
TO DESTINATION:	1.1.4 ส่งข้อมูลชั้นที่ 1
DATA STRUCTURES:	ข้อมูลประตู - รหัสนักศึกษา - ตำแหน่งประตู

DATA FLOW NAME:	ข้อมูลตำแหน่งกลอน
DESCRIPTION:	เป็นข้อมูลที่บอกว่าจะให้ไปเปิดกลอนไฟฟ้าของประตูไหน
FROM PROCESS:	1.2 ตรวจสอบตำแหน่งชั้น
TO DESTINATION:	1.1.5 สั่งเปิดประตูชั้นที่ 1
DATA STRUCTURES:	ตำแหน่งประตู

DATA FLOW NAME:	คำสั่งเปิดประตู
DESCRIPTION:	เป็นข้อมูลที่ส่งออกไปเปิดประตู
FROM PROCESS:	1.1.5 สั่งเปิดประตูชั้นที่ 1
TO DATA STORE:	กลอนไฟฟ้าประตูที่ 1-8
DATA STRUCTURES:	Status

DATA FLOW NAME:	สถานะข้อมูลภายในชั้น
DESCRIPTION:	เป็นข้อมูลที่บอกว่าตอนนี้มีข้อมูลอยู่ใน Bufer หรือไม่
FROM PROCESS:	1.2.1 Buffer ข้อมูลชั้นที่ 1
TO DESTINATION:	1.2.4 สแกนตรวจสอบข้อมูลในชั้น
DATA STRUCTURES:	Status

DATA FLOW NAME:	ข้อมูลผู้ใช้
DESCRIPTION:	เป็นข้อมูลที่บอกว่า ผู้ใช้ นั้นทำการรูดบัตรจากประตูไหนชั้นไหน
FROM PROCESS:	1.2.3 ใส่รหัสชั้นที่ 1-8
TO PROCESS:	1.2.5 ส่งข้อมูลผู้ใช้
DATA STRUCTURES:	ข้อมูลผู้ใช้ - รหัสนักศึกษา - ตำแหน่งประตู - ตำแหน่งชั้น

DATA FLOW NAME:	ข้อมูลกรเปิดประตู
DESCRIPTION:	เป็นข้อมูลที่ใช้ในการการเปิดประตูให้กับ ผู้ใช้ ที่ได้รับการอนุญาตให้เข้าใช้ห้องปฏิบัติการ
FROM PROCESS:	2.0 Server
TO PROCESS:	1.2.2 ตรวจสอบชั้นที่เปิดประตู
DATA STRUCTURES:	ข้อมูลการเปิดประตู - Status - ตำแหน่งชั้น - ตำแหน่งประตู

DATA FLOW NAME:	Login
DESCRIPTION:	เป็นข้อมูลที่ ผู้ใช้ ส่งไปให้ เซิร์ฟเวอร์ เพื่อทำการ ล็อกอิน
FROM SOURCE:	ผู้ใช้
TO PROCESS:	2.0 Server
DATA STRUCTURES:	ล็อกอิน - รหัสนักศึกษา - รหัสผ่าน

DATA STORE:	ฐานข้อมูลการเข้าใช้ห้อง
DESCRIPTION:	เป็นฐานข้อมูลที่เก็บข้อมูลการเข้ามาใช้งานห้องปฏิบัติการของ ผู้ใช้
DATA STRUCTURES:	ฐานข้อมูลการเข้าใช้ห้อง - รหัส - เวลาเข้า - วันที่เข้า

DATA STORE:	ฐานข้อมูลผู้ที่ได้รับอนุญาตให้ Login
DESCRIPTION:	เป็นฐานข้อมูลที่เก็บข้อมูลผู้ที่มีสิทธิ์ ล็อกอิน เข้าระบบ
DATA STRUCTURES:	ฐานข้อมูลผู้ที่ได้รับอนุญาตให้ ล็อกอิน - รหัสนักศึกษา - เวลาที่ได้รับการอนุมัติให้ ล็อกอิน - วันที่ได้รับการอนุมัติให้ ล็อกอิน

DATA STORE:	ฐานข้อมูลผู้มีสิทธิ์เข้าใช้ห้องปฏิบัติการ
DESCRIPTION:	เป็นฐานข้อมูลที่เก็บข้อมูลผู้ที่มีสิทธิ์เข้าใช้ห้องปฏิบัติการคอมพิวเตอร์
DATA STRUCTURES:	ฐานข้อมูลผู้มีสิทธิ์เข้าใช้ห้องปฏิบัติการ - รหัสนักศึกษา - Status

DATA STORE:	**ฐานข้อมูลการ Login**
DESCRIPTION:	เป็นฐานข้อมูลที่เก็บข้อมูลที่เก็บข้อมูลการ ล็อกอิน เข้าระบบของ ผู้ใช้
DATA STRUCTURES:	ฐานข้อมูลการ ล็อกอิน - รหัสนักศึกษา - เครื่องที่ ผู้ใช้ ใช้ ล็อกอิน - Login time - Logout time

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

บทที่ 7
การทดสอบ

7.1 วัตถุประสงค์

ในการทดสอบโครงงานระบบควบคุมการใช้งานห้องปฏิบัติการไมโครคอมพิวเตอร์เพื่อการบริหาร จะแบ่งการทดสอบออกเป็นสองส่วนคือ การทดสอบอุปกรณ์รับข้อมูล และการทดสอบทางด้านซอฟต์แวร์โดยแต่ละแบบมีวัตถุประสงค์ดังนี้

7.2 การทดสอบอุปกรณ์รับข้อมูล

เพื่อทดสอบการทำงานของอุปกรณ์รับข้อมูลว่าสามารถรับข้อมูลได้ถูกต้องตามที่ต้องการหรือไม่ ทั้งตัวที่อยู่ตามชั้นและส่วนที่อยู่ที่ เซิร์ฟเวอร์

7.3 การทดสอบทางด้านซอฟต์แวร์

1. ทดสอบความสามารถในการทำการตรวจสอบสิทธ์ในการเข้าใช้ห้องของ ผู้ใช้ว่าสามารถทำได้ถูกต้องตามที่ต้องการหรือไม่
2. ทดสอบความสามารถในการทำการกำหนดสิทธิ์การล็อกอินของผู้ใช้ว่าทำได้ถูกต้องตามที่ต้องการหรือไม่
3. ทดสอบความสามารถในการออกรายงานข้อมูลการใช้งานของห้องปฏิบัติการให้กับผู้บริหารได้ถูกต้องตามที่ต้องการหรือไม่

7.4 ขั้นตอนการทดสอบ

ในการทดสอบนั้นจะต้องเริ่มจากการทดสอบอุปกรณ์รับข้อมูลก่อนว่าสามารถทำงานได้อย่างถูกต้องจากนั้นก็นำอุปกรณ์รับข้อมูลมาเป็น input ให้กับการทดสอบทางด้านซอฟต์แวร์ต่อไป โดยลำดับการทดลองมีดังต่อไปนี้

1. ทดสอบรูดบัตรในแต่ละประตูแล้วดูว่าอุปกรณ์รับข้อมูลที่อยู่ประจำชั้นนั้นสามารถรับและส่งข้อมูลออกมาได้อย่างถูกต้องหรือไม่
2. ทดสอบรูดบัตรพร้อมกันหลายๆประตูดูว่าอุปกรณ์สามารถรับข้อมูลและส่งข้อมูลออกมาถูกต้องหรือไม่
3. นำอุปกรณ์รับข้อมูลที่อยู่ประจำชั้นมาประกอบร่วมกับอุปกรณ์รับข้อมูลที่อยู่ที่ เซิร์ฟเวอร์
4. ทำการทดสอบรูดบัตรในแต่ละประตูทุกชั้นดูว่าอุปกรณ์สามารถรับข้อมูลและส่งข้อมูลได้อย่างถูกต้องหรือไม่
5. ทำการทดสอบรูดบัตรพร้อมกันหลายๆชั้นพร้อมกันว่าอุปกรณ์สามารถรับและส่งข้อมูลได้อย่างถูกต้องหรือไม่
6. ทำการทดสอบส่งข้อมูลไปให้อุปกรณ์รับข้อมูลให้ทำการเปิดประตูว่าสามารถเปิดประตูที่ระบุได้อย่างถูกต้องตามต้องการหรือไม่
7. นำอุปกรณ์รับข้อมูลมาต่อร่วมกับ เซิร์ฟเวอร์ ของระบบเพื่อทำการทดสอบการทำงานทางด้านซอฟต์แวร์ต่อไป

8. ทดสอบตามขั้นตอนที่ 1-7 ใหม่เพื่อทดสอบว่าระบบสามารถตรวจสอบสิทธิ์การเข้าใช้งานห้องปฏิบัติ การและสั่งเปิดประตูที่ ผู้ใช้ รูดบัตรขอเข้าใช้งานห้องปฏิบัติการได้อย่างถูกต้องหรือไม่
9. ทดสอบว่าหลังจากที่ระบบตรวจสอบสิทธิ์การเข้าใช้งานห้องปฏิบัติการแล้วระบบสามารถ อีนาเบิ้ลหรือ ดิสเอเบิ้ลแอ็คเคาต์ได้อย่างถูกต้องหรือไม่
10. ทดสอบล็อกอินเข้าระบบหลังจากที่ได้รับอนุญาตให้เข้าใช้งานห้องปฏิบัติการว่าสามารถ ล็อกอินได้หรือไม่
11. ทดสอบหลังจากที่ ล็อกอิน เข้าระบบแล้ว ทิ้งเครื่องทิ้งไว้ชั่วระยะเวลาหนึ่งที่ระบบยังไม่ตัดการ ล็อกอิน แล้วทำการรูดบัตรซ้ำห้องเดิมเพื่อจำลองสถานะ การณ์ที่ ผู้ใช้ ไปเข้าห้องน้ำว่าระบบยังทำงานได้อย่าง ถูกต้องหรือไม่
12. หลังจากที่ผู้ใช้ ล็อกเอาท์ ออกจากระบบแล้วระบบทำการ ดิสเอเบิ้ลแอ็คเคาต์ได้อย่างถูกต้องหรือ ไม่
13. ทดสอบเรียกดูข้อมูลการ ล็อกอิน ที่ผ่านมาของระบบว่าออกรายงานได้อย่างถูกต้องตามที่ต้องการหรือ ไม่

7.5 ผลการทดสอบ

1. ผลการทดสอบรูดบัตรในแต่ละประตูแล้วพบว่าอุปกรณ์รับข้อมูลที่อยู่ประจำชั้นนั้นสามารถรับและส่งข้อมูลออกมาได้อย่างถูกต้องหรือไม่
2. ผลการทดสอบรูดบัตรพร้อมกันหลายๆประตูพบว่าอุปกรณ์สามารถรับข้อมูลและส่งข้อมูลออกมาถูกต้อง
3. ผลการทดสอบรูดบัตรในแต่ละประตูทุกชั้นพบว่าอุปกรณ์สามารถรับข้อมูลและส่งข้อมูลได้อย่างถูกต้อง
4. ผลการทดสอบรูดบัตรพร้อมกันหลายๆชั้นพร้อมกันพบว่าอุปกรณ์สามารถรับและส่งข้อมูลได้อย่างถูกต้อง
5. ผลการทดสอบส่งข้อมูลไปให้อุปกรณ์รับข้อมูลทำการเปิดประตูพบว่าสามารถเปิดประตูที่ระบุได้อย่างถูกต้อง
6. ผลการทดสอบหลังจากที่ระบบตรวจสอบสิทธิ์การเข้าใช้งานห้องปฏิบัติการแล้วระบบสามารถ อีนาเบิ้ลหรือ ดิสเอเบิ้ลแอ็คเคาต์ได้อย่างถูกต้อง
7. ผลการทดสอบ ล็อกอิน เข้าระบบหลังจากที่ได้รับอนุญาตให้เข้าใช้ห้องปฏิบัติพบว่าสามารถ ล็อกอิน ได้
8. ผลการทดสอบหลังจากที่ ล็อกอิน เข้าระบบแล้ว ทิ้งเครื่องทิ้งไว้ชั่วระยะเวลาหนึ่งที่ระบบยังไม่ตัดการ ล็อกอิน แล้วทำการรูดบัตรซ้ำห้องเดิมเพื่อจำลองสถานะ การณ์ที่ ผู้ใช้ ไปเข้าห้องน้ำพบว่าระบบยังทำงานได้อย่างถูกต้อง
9. ผลการทดสอบหลังจากล็อกเอาท์ ออกจากระบบแล้ว ระบบทำการ ดิสเอเบิ้ลแอ็คเคาต์ได้อย่างถูกต้อง
10. ผลการทดสอบเรียกดูข้อมูลการ ล็อกอิน ที่ผ่านมาของระบบพบว่าออกรายงานได้อย่างถูกต้องตามที่ต้องการ

แนวทางการพัฒนาต่อ

หลังจากที่ได้ศึกษาค้นคว้าและสร้างระบบควบคุมการใช้งานห้องปฏิบัติการไมโครคอมพิวเตอร์เพื่อการบริหารขึ้นมาสามารถสรุปถึงแนวทางการพัฒนาออกมาดังนี้

7.6 ข้อจำกัดของต้นแบบ

1. ระบบสามารถควบคุมได้แค่เพียงชั้นละ 8 ประตู และสามารถควบคุมได้ 8 ชั้น
2. ระบบใช้การส่งข้อมูลผ่านทาง Serial Port ทำให้ระยะทางการส่งข้อมูลมีจำกัด
3. ระบบยังต้องส่งข้อมูลมาที่ เซิร์ฟเวอร์ ทำการตรวจสอบว่า ผู้ใช้ มีสิทธิ์เข้าใช้ห้องปฏิบัติการหรือไม่ทำให้การทำงานช้า
4. ฐานข้อมูลของระบบยังใช้ฐานข้อมูลของ ไมโครซอฟต์ แอ็คเซ็ซอยู่ทำให้ถ้าฐานข้อมูลมีขนาดใหญ่มากๆ การทำงานของระบบจะทำงานได้ช้า

7.7 การพัฒนาทางด้านฮาร์ดแวร์

1. ถ้าทำการเปลี่ยนไมโครคอนโทรเลอร์ให้มีพอร์ท อินพุทและเอาท์พุท มากขึ้นก็จะสามารถ รองรับประตูได้มากขึ้น
2. ถ้าระบบทำการแปลงการส่งข้อมูลจาก RS232 มาเป็น RS 485 ก็จะทำให้ระบบสามารถรับส่งข้อมูลได้ไกลขึ้น
3. ถ้าให้ชุดอุปกรณ์รับข้อมูลเก็บข้อมูลผู้มีสิทธิ์เข้าใช้ห้องปฏิบัติการที่ตัวเครื่องเลย เมื่อเวลา อัพเดท ข้อมูลค่อยส่งข้อมูลจาก เซิร์ฟเวอร์ ไป อัพเดท ที่อุปกรณ์รับข้อมูลทำให้การตรวจสอบสิทธิ์การเข้าใช้งานห้องปฏิบัติการทำงานได้เร็วขึ้นและโอกาศที่ข้อมูลจะสูญหายระหว่างทางก็จะหายไป

7.8 การพัฒนาทางด้านซอฟต์แวร์

จากที่ใช้ระบบฐานข้อมูลของ ไมโครซอฟต์ แอ็คเซ็ซก็หันมาทำ คาต้า เบท เซิร์ฟเวอร์ เพื่อให้ระบบจัดการกับฐานข้อมูล ได้อย่างมีประสิทธิ์มากขึ้นทำให้ระบบสามารถทำงานได้ไวมากยิ่งขึ้น

บรรณานุกรม

หนังสืออ้างอิง

1. ศุภชัย สมพานิช : "Database programming ด้วย Visual Basic ฉบับมืออาชีพ" สำนักพิมพ์ อินโฟเพรส 2543 พิมพ์ครั้งที่ 2
2. ปรเมษฐ์ ประนยานันทน์ : "คู่มือและการประยุกต์ใช้งานไมโครคอนโทรลเลอร์ MCS-51" บริษัท ซีเอ็ดยูเคชั่น จำกัด (หมาชน) 2536

โฮมเพจอ้างอิง

1. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/debug/eventlog_622b.asp
2. http://www.thaiio.com/programming/vbserial.htm

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง