ตรวจสอบเส้นทางรถเมล์ในกรุงเทพผ่านมือถือ

BANGKOK BUS ROUTES INFORMATION SEARCHING VIA MOBILE PHONE

โดย

นายนภดล ณ ถลาง

นายภาณุพงษ์ กนกหงษ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมสารสนเทศ คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2549

BANGKOK BUS ROUTES INFORMATION SEARCHING VIA MOBILE PHONE

BY

MR. PANUPONG KANOKHONG

MR. NOPADOL NATALANG

A PROJECT SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENT FOR THE DEGREE OF

BACHELOR IN DEPARTMENT OF INFORMATION ENGINEERING

FACULTY OF ENGINEERING

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2006

หัวข้อปริญญานิพนธ์ ตรวจสอบเส้นทางรถเมล์ในกรุงเทพผ่านมือถือ

BANGKOK BUS ROUTES INFORMATION SEARCHING VIA MOBILE PHONE

ชื่อนักศึกษา	นายนภดล	ณ ถลาง	รหัสประจำตัว 47015839
	นายภาณุพงษ์	กนกหงษ์	รหัสประจำตัว 47015848
อาจารย์ที่ปรึกษา	รศ. นภพินทุ์	อนันตรศิริชัย	
	รศ.ดร. ชวลิต	เบญจางคประเสริฐ	

ระดับการศึกษา ปริญญาตรี วิศวกรรมศาสตร์บัณฑิต
สาขาวิศวกรรมสารสนเทศ

ภาควิชา วิศวกรรมสารสนเทศ

ปีการศึกษา 2549

ปริญญานิพนธ์ฉบับนี้ได้รับการอนุมัติเป็นส่วนหนึ่งของการศึกษาตามหลักสูตร วิศวกรรมศาสตร์บัณฑิต คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

.....

(รศ. นภพินทุ์ อนันตรศิริชัย)
อาจารย์ที่ปรึกษา

.....

(รศ.ดร. ชวลิต เบญจางคประเสริฐ)
อาจารย์ที่ปรึกษาร่วม

หัวข้อปริญญานิพนธ์ ตรวจสอบเส้นทางรถเมล์ในกรุงเทพผ่านมือถือ

ชื่อนักศึกษา นายนภดล ณ ถลาง รหัสประจำตัว 47015839
นายภาณุพงษ์ กนกหงษ์ รหัสประจำตัว 47015848

อาจารย์ที่ปรึกษา รศ. นภพินทุ์ อนันตรศิริชัย
รศ.ดร. ชวิลิต เบญจางคประเสริฐ

ระดับการศึกษา ปริญญาตรี วิศวกรรมศาสตร์บัณฑิต
สาขาวิศวกรรมสารสนเทศ

ภาควิชา วิศวกรรมสารสนเทศ

ปีการศึกษา 2549

บทคัดย่อ

โครงงานประยุกต์การค้นหาเส้นทางการเดินรถเมล์ และการค้นหาข้อมูลรถเมล์ ภายในกรุงเทพ ผ่าน mobile web Application หรือ ใช้ web ผ่านโทรศัพท์มือถือนั้นเอง การเขียนโปรแกรมใช้ ฟังก์ชัน mobile web Application ใน Visual Basic.net 2003 ทำโครงงานนี้เพื่อติดต่อกับฐานข้อมูล

ฟังก์ชันในการทำงานค้นหาเส้นทางการเดินรถเมล์สามารถต่อสายรถเมล์มี 2 ฟังก์ชัน ฟังก์ชันแรกผู้ใช้งานจะต้องระบุสถานที่ต้นทางกับปลายทาง โครงงานนี้จะถูกออกแบบให้สามารถค้นหาเส้นทางต่อรถเมล์มากกว่า 1 ต่อ โดยโปรแกรมจะเลือกแสดงผล เฉพาะเส้นทางที่ต่อรถเมล์น้อยที่สุดส่วนการทำงานค้นหาข้อมูลรถเมล์ ผู้ใช้งานจะต้องระบุชื่อสายรถเมล์กับชนิดของรถเมล์ที่ต้องการ

Thesis Title	BANGKOK BUS ROUTES SEARCHING VIA MOBILE PHONE
Student	Mr. Nopadol Na Tlang ID. 47015839
	Mr. Panupong Kanokhong ID. 47015848
Advisor	Assoc. Prof. Noppin Anuntarasirichai
	Assoc. Prof. Dr. Chawalit Benjangkaprasert
Graduate Level	Bachelor Degree of Information Engineering
Department	Information Engineering
Academic Year	2006

ABSTRACT

Project is the Search Bus Routes and Bus information Application in Bangkok via mobile web Application. Program writing use mobile web Application function of Visual Basic.net 2003 for connected to database.

Function of Bangkok Bus routes information searching via mobile phone has 2 functions. First function user fills the start place in textbox and target place in textbox. This project is designed Search Bus Routes more than one routes. The program will show the shortcut route and user should be select the type of bus.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงได้ เพราะความร่วมมือของสมาชิกภายในกลุ่ม และอาจารย์ทุกท่านที่ได้ให้คำปรึกษา คำแนะนำที่ดี และกำลังใจอีกมากมายจากบิดา มารดา เพื่อนที่คอยถามไถ่ให้ความห่วงใยมาโดยตลอดในยามที่รู้สึกท้อถอย ให้มีกำลังใจเพื่อทำให้ทุกสิ่งลุล่วงไปด้วยดี

สุดท้ายนี้ขอให้สิ่งดี ๆ จงบังเกิดแด่ทุก ๆ ท่านที่กล่าวถึงโดยตลอดไป

คณะผู้จัดทำ

สารบัญ

สารบัญรูปภาพ

หน้า

หน้า

สารบัญตาราง

บทที่ 1
บทนำ

1.1 บทนำ

จากสภาพการจราจรภายในกรุงเทพมหานครปัจจุบันมีความติดขัดมากและเนื่องด้วยราคาน้ำมันสูงขึ้น ประชาชนส่วนใหญ่หันมาใช้บริการรถโดยสารประจำทางกันมากขึ้น เพื่อประหยัดค่าใช้จ่ายในเรื่องของการเดินทางและลดสภาพความแออัดของการจราจร แต่ในทางกลับกันก็มีเรื่องของเวลาในการเดินทางเข้ามาเกี่ยวข้องด้วยเช่นกัน เป็นที่ทราบกันดีว่า การเดินทางภายในกรุงเทพฯนั้น จากที่หนึ่งไปยังอีกที่หนึ่งไม่ใช่เวลาน้อย ๆ เลย บางคนที่เพิ่งจะเข้ามาอยู่ในกรุงเทพฯได้ไม่นานต้องประสบปัญหาเรื่องการเดินทางไปยังสถานที่ต่าง ๆ อยู่บ่อย ๆ จากปัญหาดังกล่าว ทำให้เล็งเห็นว่าควรที่จะพัฒนาโปรแกรมเพื่อช่วยในการตัดสินใจและเลือกเส้นทาง เพื่อเป็นประโยชน์ในการตัดสินใจก่อนออกจากบ้าน

1.2 วัตถุประสงค์

- เพื่อพัฒนาระบบการค้นหาเส้นทางให้มีประสิทธิภาพยิ่งขึ้น
- เพื่อสร้างความสะดวกรวดเร็วในการตัดสินใจในการออกเดินทาง
- เพื่อศึกษาการวิเคราะห์และออกแบบฐานข้อมูลที่มีหลักการถูกต้อง ลดความซ้ำซ้อนของข้อมูลและการจัดเก็บอย่างมีระเบียบ

1.3 ขอบเขตของโครงงาน

- สามารถบอกรายละเอียดบอกเกี่ยวกับเส้นทาง ระยะทางของรถสายนั้นๆได้
- สามารถค้นหาเส้นทางของรถประจำทางได้
- สามารถบอกเส้นทางการเดินทางจากที่หนึ่งไปยังอีกที่ได้

1.4 ผลที่คาดว่าจะได้รับ

- สามารถนำไปใช้ให้เกิดประโยชน์ต่อสังคมได้
- ช่วยในการตัดสินใจเลือกเส้นทางการเดินทางเพื่อให้รวดเร็วต่อสถาณการณ์

1.5 ขั้นตอนการดำเนินงาน

- ศึกษาขอบเขตของระบบที่จะพัฒนา
- วิเคราะห์ระบบงานทั้งหมด
- วิเคราะห์ระบบแล้วนำมาเขียนแผนภาพการไหลของข้อมูล (Data Flow) เพื่อตรวจสอบความเข้าใจของระบบงานระหว่างผู้ใช้และผู้พัฒนาระบบ
- ออกแบบระบบฐานข้อมูลในระดับแนวคิด (Conceptual Model) โดยใช้แบบจำลองทางแนวคิดไนแอม (NIAM) แล้วเขียนเป็นตาราง Data Dictionary
- ออกแบบส่วนของโปรแกรมติดต่อกับผู้ใช้ (Use Interface Design)
- พัฒนาโปรแกรมโดยใช้ Visual Basic.Net เชื่อมต่อกับฐานข้อมูล Access ผ่าน ADO.net
- ตรวจสอบความถูกต้องของโปรแกรม

บทที่ 2
ทฤษฎีหลักการของการออกแบบฐานข้อมูล

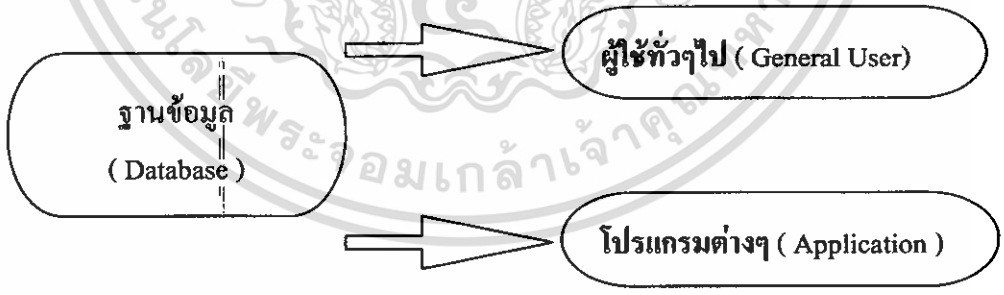
เรามักจะนำคอมพิวเตอร์มาใช้ในการประมวลผลต่างๆ เช่น ระบบการออกใบรายการสินค้า ระบบคลังสินค้า และระบบบันทึกเวลาเข้าออกของพนักงานเป็นต้น เนื่องจากข้อดีของคอมพิวเตอร์ที่สามารถเก็บข้อมูลได้เป็นจำนวนมากและทำงานได้รวดเร็ว ซึ่งในอดีตนั้นต้องใช้กำลังคนจำนวนมาก และยังก่อให้เกิดข้อผิดพลาดได้ง่ายอีกด้วย

สภาวะในปัจจุบัน ระบบธุรกิจที่ต้องอาศัยความรวดเร็วทั้งการตัดสินใจ และขบวนการในการทำงาน การนำคอมพิวเตอร์มาใช้จำเป็นสิ่งที่จำเป็นและหลีกเลี่ยงไม่ได้ถ้าเราต้องการให้ธุรกิจของเราก้าวทันคู่แข่งและประสบความสำเร็จในการดำเนินธุรกิจ

2.1 ระบบฐานข้อมูลและการออกแบบฐานข้อมูลโดยใช้แบบจำลอง NIAM

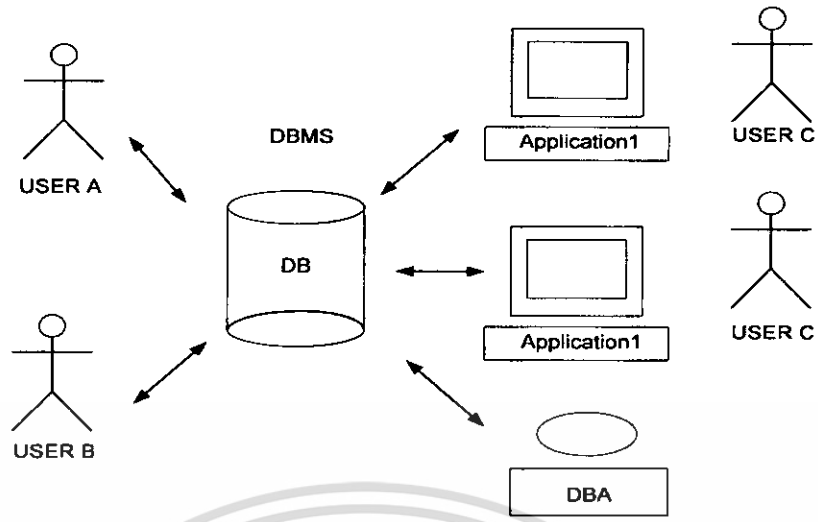
2.1.1 ฐานข้อมูล (Database)

ไฟล์ชนิดหนึ่งที่ใช้เก็บข้อมูล โดยเมื่อต้องการใช้ข้อมูลก็สามารถดึงออกมาใช้ได้ทันที แต่ไฟล์ข้อมูลเหล่านี้ไม่เหมือนกับไฟล์ทั่วๆไป เนื่องจากมีความสามารถในการจัดเก็บข้อมูลที่ดีกว่า รวมทั้งสามารถเรียกใช้งานได้โดยตรงจากผู้ใช้หรือจากโปรแกรมที่โปรแกรมเมอร์ได้สร้างขึ้นฐานข้อมูลนั้น ถือว่าเป็นองค์ประกอบหลักในการใช้งานโปรแกรมแทบทุกประเภทเลยทีเดียว



รูปที่ 2.1 แสดงประโยชน์ของฐานข้อมูลที่เปรียบเสมือนตัวเก็บข้อมูล

ระบบฐานข้อมูล (Database System) จะประกอบไปด้วย 4 ส่วนหลักคือฐานข้อมูล (Database), ซอฟต์แวร์จัดการระบบฐานข้อมูล (DBMS), โปรแกรมใช้งานฐานข้อมูล (Application Program) และ ผู้ใช้งาน (Users) มีความสัมพันธ์ดังรูป



รูปที่ 2.2 แสดงความสัมพันธ์ของฐานข้อมูลและระบบฐานข้อมูล

- Database Management System (DBMS) ทำหน้าที่เป็นตัวกลางระหว่างฐานข้อมูลกับโปรแกรม ที่มาใช้ งานฐานข้อมูลและผู้ใช้งานในการติดต่อไปยังฐานข้อมูลเพื่อทำงานที่ผู้ใช้งานสั่งมาให้สำเร็จ
- Application Programs หมายถึง โปรแกรมหรือแอพลิเคชั่น (Application) ที่พัฒนาขึ้นมาเพื่อใช้ประโยชน์จากข้อมูลที่เก็บไว้ในฐานข้อมูล
- Users หมายถึง ทุกคนที่เกี่ยวข้องกับฐานข้อมูล

โครงสร้างของฐานข้อมูล

- ระดับภายนอก (External View) หมายถึง สิ่งที่ผู้ใช้งานแต่ละคนสามารถมองเห็นในฐานข้อมูลนั้น
- ระดับแนวคิด (Conceptual View) หมายถึง ข้อมูลทั้งหมดที่มีอยู่ในฐานข้อมูลซึ่งจะถูกแสดงตามแบบจำลองข้อมูล
- ระดับภายใน (Inter View) หมายถึง ระดับที่จัดเก็บข้อมูลด้วยโครงสร้างข้อมูลที่เหมาะสมซึ่งมีผลต่อความเร็วและประสิทธิภาพเข้าถึงข้อมูลที่ต้องการ

2.1.2 การออกแบบฐานข้อมูลโดยใช้แบบจำลอง NIAM

NIAM (Nijssen's Information Analysis Methodology) เป็นวิธีการในการออกแบบฐานข้อมูลโดยการแสดงความสัมพันธ์ต่างๆ ของข้อมูล มีพื้นฐานมาจากภาษาธรรมชาติ คือมีรูปแบบประโยคเป็นแบบประธาน กริยา กรรม นอกจากนั้นยังสามารถแปลงโครงสร้างฐานข้อมูลเชิงสัมพันธ์ ดังนั้นจึงสะดวกในการออกแบบฐานข้อมูลของระบบงานที่มีขนาดใหญ่

2.1.2.1 เป้าหมายหลักของการออกแบบระบบฐานข้อมูล

- **ลดความซ้ำซ้อนของข้อมูล** ให้ฐานข้อมูลปราศจากความซ้ำซ้อนหรือมีความซ้ำซ้อนน้อยที่สุดโดยที่ยังคงซ้ำซ้อนอยู่จะต้องถูกควบคุมอย่างเข้มงวดรัดกุมและสามารถตรวจสอบได้ถูกต้อง
- **ควบคุมความถูกต้องของข้อมูล** ข้อมูลที่จัดเก็บในระบบฐานข้อมูลนั้นจะไม่มีประโยชน์เลยหรือมีประโยชน์น้อยลง ถ้าหากข้อมูลที่เก็บรักษาไว้นั้นเป็นข้อมูลที่ไม่น่าเชื่อถือและมีความผิดพลาดสูง

2.1.2.2 ชนิดของความซ้ำซ้อน

- ตารางใด ๆ เก็บข้อมูลของหลายความสัมพันธ์ (FD : Functional Dependency) ไว้ร่วมกันโดยที่ความสัมพันธ์เหล่านั้นมี Determinant ที่แตกต่างกัน
- ตารางใด ๆ เก็บข้อมูลของหลายความสัมพันธ์ (MVD : Multi-Valued Dependency) ไว้ร่วมกัน โดยที่ความสัมพันธ์เหล่านั้นไม่เกี่ยวข้องกันโดยตรง
- เก็บค่าของข้อมูลที่ได้จากการคำนวณจากค่าของข้อมูลอื่นๆที่มีอยู่ในฐานข้อมูลนั้น (หรือค่าข้อมูลที่ได้จากการเชื่อมโยงจากค่าข้อมูล)

2.1.2.3 หลักการของ NIAM

- กำหนดชนิด เอ็นทิตี้ (Entity) และชนิด Label ตามข้อมูลทั้งหมดที่วิเคราะห์ได้จากการวิเคราะห์ปัญหาและการวิเคราะห์ระบบงาน
- เติมสัญลักษณ์แสดง ยูนิคอนสเตรน (Unique Constrain) ตรวจสอบความถูกต้องของชนิดความจริง
- เติมสัญลักษณ์แสดง เล็กซิคอล (Lexical), แมนดาทอรีโรลและสับไทพ (Mandatory Role and Subtype Constrain) ตรวจสอบ ยูนีคไอเดนทิไฟเยอร์ (Unique identifier) ของแต่ละชนิด เอ็นทิตี้
- เติมสัญลักษณ์แสดง อิควอลิตี้คอนเสตรน (Equality Constrain), เอ็กซ์คลูชั่นคอนสเตรน (Exclusion Constrain) และสับเซตคอนสเตรน (Subset Constrain)

- ตรวจสอบความสมบูรณ์ของโครงสร้างทางแนวคิดที่ได้ออกแบบว่าสอดคล้องกับตัวอย่างข้อมูลและไม่มีความซ้ำซ้อนของข้อมูล

2.1.2.4 ส่วนประกอบพื้นฐานของ NIAM

- Entity type หมายถึง เซตของสิ่งเราสนใจ ทั้งที่อยู่ในรูปของสิ่งที่จับต้องได้และจับต้องไม่ได้ เช่นคน , บริษัท
- Attribute หมายถึง ข้อมูลที่แสดงลักษณะของ Entity type เช่นจากตาราง Employee จะพบว่ามี Abttibute ของตารางคือ ID, Name, Surname, Sex, Slary, Dept_ID
- Label type หมายถึง เซตของสิ่งที่ใช้บ่งบอกความแตกต่าง หรือชื่อของแต่ละ entity ที่กำหนด เช่น ชื่อ, นามสกุล
- Role หมายถึง ความสัมพันธ์ที่เกี่ยวข้องกับ entity ที่เชื่อมกันอยู่เช่น พนักงาน ชื่อนี้ทำงานอยู่ที่ บริษัทนี้
- Element fact type หมายถึงเซตความสัมพันธ์ระหว่างสมาชิกของชนิด entity ตั้งแต่ 2 ชนิดขึ้นไปโดยที่ชนิดความจริงที่มี2 ความสัมพันธ์จะเรียกว่า Binary fact type ส่วนชนิดความจริงที่มี 3 ความสัมพันธ์ จะเรียกว่า Ternary fact type สำหรับชนิดความจริงที่มีมากกว่า 3 สัมพันธ์ขึ้นไปจะเรียกว่า N-ary fact type
- Reference type หมายถึง เซตของความสัมพันธ์ระหว่างสมาชิกของชนิด entity กับสมาชิกของ ชนิด label ที่มีอยู่
- Nested fact type หมายถึง ชนิด entity ที่แสดงความสัมพันธ์ในการกำหนดกลุ่มของชนิดความจริงที่มีตั้งแต่ 2 ความสัมพันธ์ขึ้นไป
- Integrity constraints หมายถึงสิ่งที่ใช้แสดงกฎที่ใช้ในการบังคับควบคุมความถูกต้องของข้อมูล

2.1.2.5 ฐานข้อมูลเชิงสัมพันธ์ (Relational Database)

การรวม Entity ที่อยู่ในระบบที่มีความสัมพันธ์ระหว่างกันเข้าไว้ด้วยกันซึ่งชนิดของความสัมพันธ์ก็จะได้กล่าวได้ว่ามีอยู่ 4 ชนิด แต่ใช้จริงๆอยู่ 3 ชนิด

- **ความสัมพันธ์แบบ 1:1 (One-To-One)** เป็นความสัมพันธ์ที่ในหนึ่ง record ของตารางหนึ่งมีความสัมพันธ์กับอีกหนึ่ง record ของอีกตารางหนึ่ง ดังตัวอย่างต่อไปนี้ หมายถึงบริษัททั่วไป แผนกหนึ่งสามารถมีหัวหน้าแผนกได้เพียงคนเดียวเท่านั้น ดังนั้น ความสัมพันธ์ระหว่างตารางแผนกกับตารางพนักงานจึงเป็นแบบ 1:1

Department Table (ตารางแผนก)			หัวหน้าแผนก	Employee Table (ตารางหัวหน้า)			
DeptID	DeptName	MngID		EmpID	Title	Fname	Lname
01	จัดซื้อ	062	→	005	นาย	สุรัตน์	ขยันจริง
02	ขาย	055	→	062	น.ส.	จิติมา	ยิ่งใหญ่
03	บัญชี	088	→	088	นาย	ยิ่งศักดิ์	มักใหญ่

ตารางที่ 2.1 แสดงความสัมพันธ์แบบ One To One

- **ความสัมพันธ์แบบ 1:M (One-To-Many)** เป็นความสัมพันธ์ใน 1 Record ของตารางมีความสัมพันธ์กับอีกหนึ่งหรือหลาย Record ของตารางอื่น ดังตัวอย่างต่อไปนี้ ซึ่งลูกค้าหนึ่งคนสามารถสั่งซื้อสินค้าได้หลายครั้งและใบกำกับสินค้าหนึ่งใบก็สามารถมีลูกค้าได้เพียงคนเดียวเท่านั้น เช่น นายลิท-เติ้ล (ชื่อเล่นว่า น้อยนิด) สั่งซื้อสินค้าจากบริษัททั้งสิ้น 2 ครั้ง ดังนั้นความสัมพันธ์ระหว่างตารางลูกค้ากับใบกำกับสินค้า จึงถือเป็นความสัมพันธ์แบบหนึ่งต่อกลุ่ม หรือ One-To-Many

Customer Table (ตารางลูกค้า)

CustID	Title	Fname	Lname
C01	นาย	การุณ	อุ่นใจ
C02	น.ส.	ลิดเติล	ใจน้อย
C03	น.ส.	เพลินใจ	ไพลิน

ซื้อสินค้า

Invoice Table (ตารางใบกำกับสินค้า)

CustID	InvNo	InvDt
C01	11001	10/12/1999
C02	11002	10/6/2000
C03	11003	30/6/2000
C02	11004	10/7/2000
C01	11005	20/8/2000

ตารางที่ 2.2 แสดงความสัมพันธ์แบบ One To Many

- **ความสัมพันธ์แบบ M:M (Many-To-Many)** เป็นความสัมพันธ์ที่ข้อมูลหนึ่งหรือหลาย Record ในตารางหนึ่งมีความสัมพันธ์กับหนึ่งหรือหลาย Record ในอีกตารางหนึ่ง ดังตัวอย่างต่อไปนี้ ซึ่งจะเห็นได้ว่าลูกค้าคนหนึ่งสามารถซื้อสินค้าได้หลายรายการ และ สินค้าหนึ่งรายการก็สามารถถูกซื้อโดยลูกค้าหลายคนเช่นกัน ซึ่งความสัมพันธ์ลักษณะนี้จะเป็นแบบ Many-To-Many (M:M)

Customer By Product Table (ตารางการซื้อสินค้า)

CustID	Title	Fname	Lname	ProID
C01	นาย	สมบัติ	พัสถาน	P01
C01	นาย	สมบัติ	พัสถาน	P05
C01	นาย	สมบัติ	พัสถาน	P08
C05	น.ส.	ลิตเติล	ใจน้อย	P01
C05	น.ส.	ลิตเติล	ใจน้อย	P05
C05	น.ส.	ลิตเติล	ใจน้อย	P08
C08	น.ส.	บัวบาน	นานนม	P01
C08	น.ส.	บัวบาน	นานนม	P05

ซื้อสินค้า

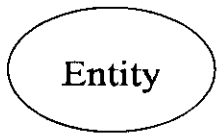
Product Table (ตารางสินค้า)

PrID	PrName	BrName	Price
P01	แอร์	Whirpool	9,500
P05	ตู้เย็น	National	5.000
P08	ตู้เย็น	Hitachi	4,500

ตารางที่ 2.3 แสดงความสัมพันธ์แบบ Many To Many

ความสำพันธ์แบบ M:M ทำให้เกิดปัญหาเรื่องความซ้ำซ้อนของข้อมูลและอาจเกิดปัญหาความผิดพลาดของข้อมูลตามมาได้ เช่น ในกรณีตัวอย่างด้านบนถ้าหากลูกค้าที่ชื่อชื่อ "ลิตเติล ใจน้อย" ได้แต่งงานหรือเปลี่ยนนามสกุลจะต้องแก้ไขข้อมูลถึง 3 Record ซึ่งหากมี การแก้ไขไม่ครบถ้วนก็มีโอกาสผิดพลาดได้สูง

2.1.2.6 สัญลักษณ์ของส่วนประกอบพื้นฐานของแบบจำลอง NIAM

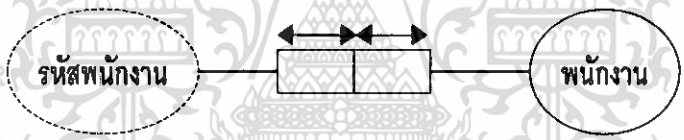


รูปที่ 2.3 แสดงสัญลักษณ์ของ entity

- **อธิบายรูปที่ 2.3 หมายถึง แผนกเป็น entity หนึ่งๆ ซึ่งอาจจะมีส่วนประกอบย่อยๆ อีก**

รูปที่ 2.4 แสดงสัญลักษณ์ label ของรหัสแผนก

- **อธิบายรูปที่ 2.4 หมายถึง รหัสแผนกเป็นส่วนประกอบย่อยของแผนก**

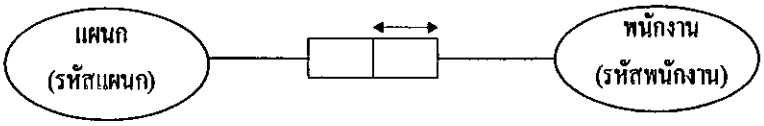


รูปที่ 2.5 แสดงสัญลักษณ์ความสัมพันธ์แบบ one to one

- **อธิบายรูปที่ 2.5 หมายถึงรหัสพนักงานหนึ่งรหัสจะเป็นพนักงานได้หนึ่งคน และพนักงานหนึ่งคนจะมีรหัสพนักงานได้เพียงหนึ่งรหัส**

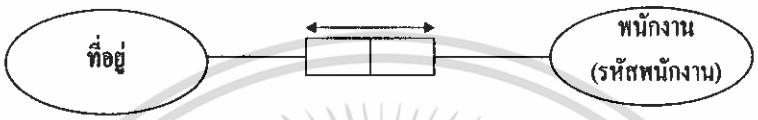
รูปที่ 2.6 แสดงความสัมพันธ์แบบ one to one แบบเจาะจง

- **อธิบายรูปที่ 2.6 หมายถึง เป็นความสัมพันธ์เหมือนรูปที่ 2.4**



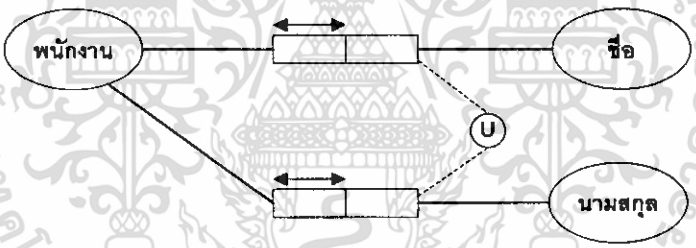
รูปที่ 2.7 แสดงความสัมพันธ์แบบ one to many

- **อธิบายรูปที่ 2.7 หมายถึง แผนกหนึ่งแผนกจะมีพนักงานทำงานได้หลายคน แต่พนักงานหนึ่งคนสามารถทำงานได้เพียงหนึ่งแผนก**



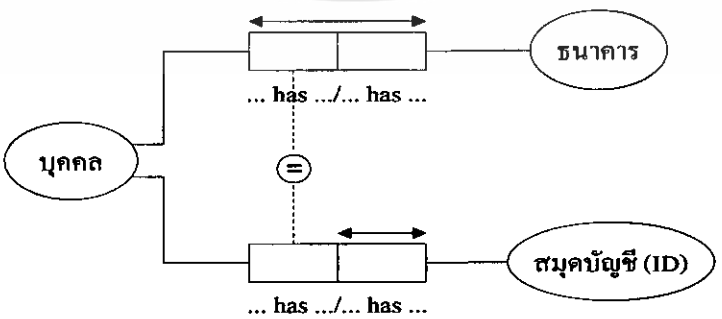
รูปที่ 2.8 แสดงความสัมพันธ์แบบ many to many

- **อธิบายรูปที่ 2.8 หมายถึง ที่อยู่หนึ่งที่อยู่สามารถมีพนักงานอยู่ได้หลายคน และพนักงานหนึ่งคน สามารถมีที่ อยู่ได้หลายที่อยู่**



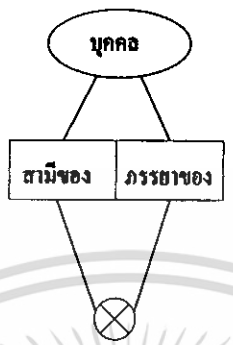
รูปที่ 2.9 แสดงสัญลักษณ์ของการใช้ Inter Fact Type Uniqueness Constraint

- **อธิบายรูปที่ 2.9 หมายถึง พนักงานอาจมีชื่อซ้ำกัน แต่ถ้ารวมนามสกุลด้วยจะไม่ซ้ำกันแน่นอน**



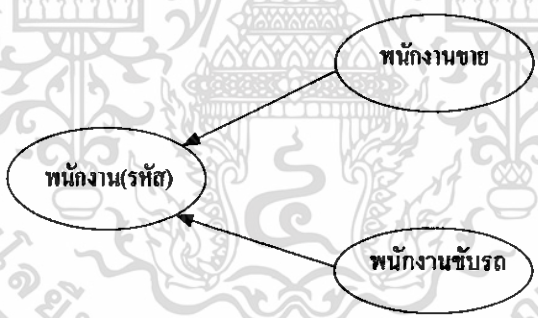
รูปที่ 2.10 แสดงสัญลักษณ์แสดงการใช้ Equality Constraint

- อธิบายรูปที่ 2.10 หมายถึง บุคคลที่มีความสัมพันธ์เป็นลูกค้าของธนาคารใด แล้ว จำเป็นต้องมีสมุดบัญชีของธนาคารนั้นด้วย



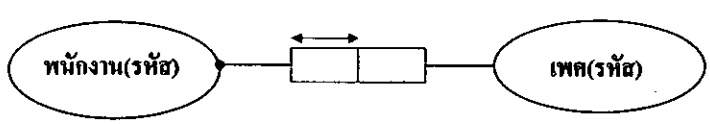
รูปที่ **2.11** แสดงสัญลักษณ์แสดงการใช้ Exclusion Constraint

- อธิบายรูปที่ 2.11 หมายถึงบุคคลใดเป็นภรรยาของอีกบุคคลหนึ่งแล้ว จะไม่เป็นสามีของบุคคลใดๆ และถ้าบุคคลใดเป็นสามีของอีกบุคคลหนึ่งแล้ว จะไม่เป็นภรรยาของบุคคลใดๆ



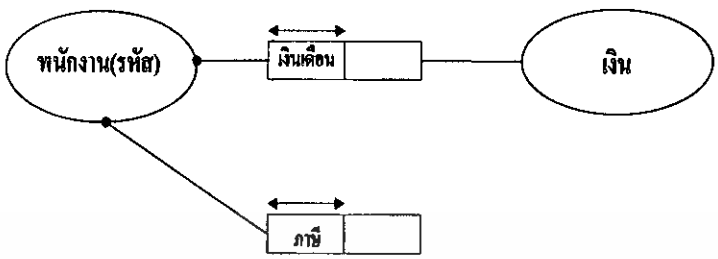
รูปที่ **2.12** แสดงสัญลักษณ์แสดงการใช้ Subtype Constraint

- อธิบายรูปที่ 2.12 หมายถึงในรหัสพนักงานมี 2 subtype แบ่งออกเป็นพนักงานขาย และพนักงานขับรถ



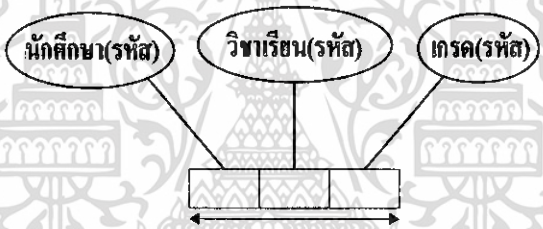
รูปที่ **2.13** แสดงสัญลักษณ์แสดงการใช้ Mandatory Constraint, Lexical Constraint

- อธิบายรูปที่ 2.13 หมายถึง พนักงานทุกคนจำเป็นต้องมีเพศ แต่เพศนั้นไม่จำเป็นต้องเป็นของพนักงานทุกคน



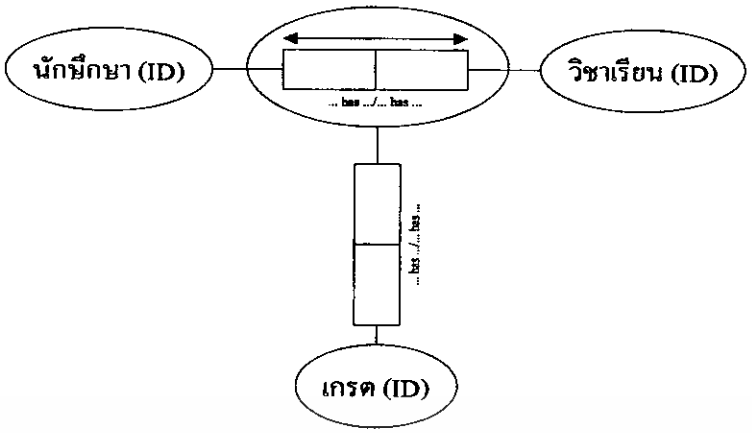
รูปที่ **2.14** แสดงสัญลักษณ์แสดงความสัมพันธ์ที่มี 2 หน้าที่

- อธิบายรูปที่ 2.14 หมายถึงพนักงานหนึ่งคนจะได้รับเงินเดือนเป็นเงินจำนวนหนึ่ง และพนักงานจะต้องเสียภาษีเป็นยอดเงินจำนวนหนึ่ง



รูปที่ **2.15** แสดงสัญลักษณ์แสดงความสัมพันธ์แบบ Ternary Fact Type

- อธิบายรูปที่ 2.15 หมายถึง นักศึกษาหนึ่งคนสามารถมีได้หลายวิชาเรียนมีได้หลายเกรด วิชาเรียนหนึ่งวิชาเรียนสามารถมีนักศึกษาได้หลายคนมีได้หลายเกรด และเกรดหนึ่งเกรดสามารถมีได้หลายวิชาเรียนซึ่งอาจเป็นของนักศึกษาได้หลายคน



รูปที่ 2.16 แสดงสัญลักษณ์แสดงความสัมพันธ์แบบ Nested Fact Type

- อธิบายรูปที่ 2.16 หมายถึง นักศึกษาที่เรียนวิชานั้นๆ จำเป็นต้องมีเกรด แต่ทุกเกรดไม่จำเป็นต้องเป็นของทุกวิชาเรียนที่นักศึกษาเรียน

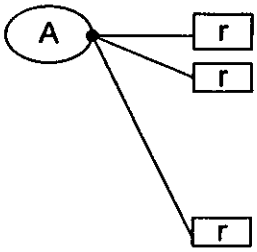
2.1.2.7 ชนิดของกฎบังคับตามความถูกต้องของข้อมูล

- Mandatory role constraints เป็นกฎข้อบังคับความถูกต้องที่ใช้ในการควบคุมเพื่อแสดงให้เห็นถึงการมีอยู่ของข้อมูลว่าต้องมีการบันทึกข้อมูลทุกครั้งที่เกิดมีความสัมพันธ์กันขึ้น สามารถแสดงได้ดังรูป

รูปที่ 2.17 แสดงความสัมพันธ์แบบ Mandatory role constraints

สามารถอธิบายได้ว่าสมาชิกทุกตัวใน entity A จะต้องถูกบันทึกข้อมูลเมื่อมีบทบาท r เกิดขึ้นเนื่องจากมีจุดทึบเชื่อมต่อระหว่าง A กับ r

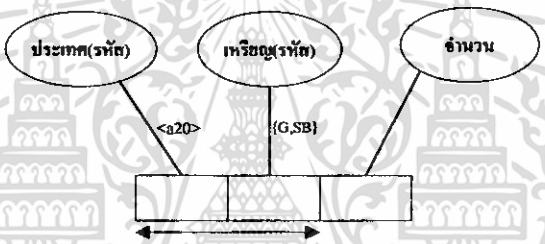
- Inclusion mandatory role constraints เป็นกฎข้อบังคับความถูกต้องที่แสดงให้เห็นถึงทางเลือกของบทบาทในกลุ่มของความสัมพันธ์ ที่มีอยู่ว่าต้องมีการบันทึกข้อมูลอย่างน้อยบทบาทหนึ่งของชนิด entity นั้นดังแสดงในรูป



รูปที่ 2.18 แสดงความสัมพันธ์แบบ Inclusion mandatory role constraints

สามารถอธิบายได้ว่าสมาชิกของชนิด entity A ใดๆต้องมีการบันทึกความสัมพันธ์ที่เกิดขึ้นความสัมพันธ์ใดความสัมพันธ์หนึ่ง

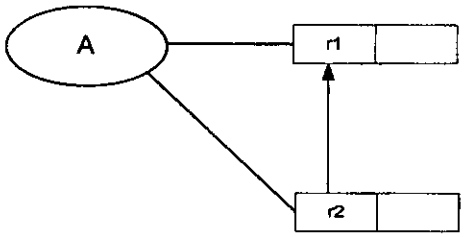
- Entity type constraints (Value constraints) เป็นกฎข้อบังคับความถูกต้องที่ใช้ในการกำหนดค่าของสมาชิกภายในเซ็ตของข้อมูลที่เป็นไปได้ของชนิด label หรือชนิด entity รวมไปถึงการกำหนดชนิดของข้อมูลในเซ็ตด้วยดังแสดงในรูป



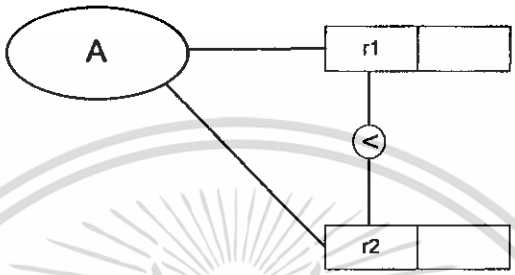
รูปที่ 2.19 แสดงความสัมพันธ์แบบ Entity type constraints

สามารถอธิบายได้ว่า เป็นการระบุชนิดของเหรียญรางวัลในการแข่งขันกีฬาสามารถแยกออกได้เป็นเหรียญทองแดง เหรียญเงิน และเหรียญทอง ระบุถึงจำนวนของเหรียญรางวัลที่ได้ว่าต้องอยู่ในช่วง 1 ถึง 200 เหรียญ รวมทั้งยังสามารถระบุชนิดของข้อมูลได้ด้วยชื่อประเทศถูกกำหนดให้จัดเก็บได้ไม่เกิน 20 ตัวอักษร

- Subset Constraint เป็นกฎข้อบังคับความถูกต้องของข้อมูลที่แสดงความสัมพันธ์ที่เป็นส่วนหนึ่งของความสัมพันธ์ที่มีอยู่แต่จะมีลักษณะความสัมพันธ์ไปในทางเดียวดังแสดงความสัมพันธ์ได้โดยใช้สัญลักษณ์ A $\longrightarrow$ B สามารถแสดงได้ดังรูป



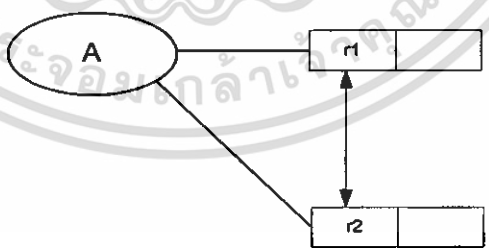
รูปที่ 2.20 แสดงความสัมพันธ์แบบ Subset Constraints (1)



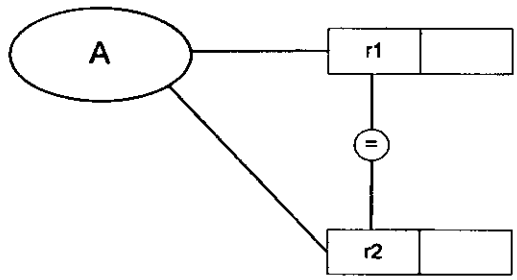
รูปที่ 2.21 แสดงความสัมพันธ์แบบ Subset Constraints (2)

สามารถอธิบายได้ว่า สมาชิกแต่ละตัวของชนิด entity A หากมีการบันทึกความสัมพันธ์ r2 แล้วต้องมีการบันทึกความสัมพันธ์ r1 ด้วยแต่ในทางกลับกัน สมาชิกแต่ละตัวของชนิด entity A หากมีการบันทึกความสัมพันธ์ r1 แล้วไม่จำเป็นต้องมีการบันทึกความสัมพันธ์ r2 ก็ได้

- Equality Constraints เป็นกฎข้อบังคับความถูกต้องที่แสดงให้เห็นว่า ชนิด entity เหล่านั้นจะต้องมีการถูกบันทึกข้อมูลควบคู่กันเสมอไป ใช้สัญลักษณ์แสดงความสัมพันธ์ได้คือ $A \leftrightarrow B$ สามารถแสดงได้ดังรูป



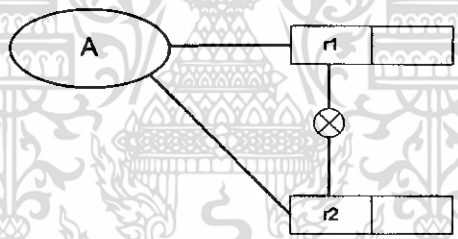
รูปที่ 2.22 แสดงความสัมพันธ์แบบ Equality Constraints (1)



รูปที่ 2.23 แสดงความสัมพันธ์แบบ Equality Constraints (2)

สามารถอธิบายได้ว่าแสดงถึงกฎข้อบังคับความถูกต้องของข้อมูลว่า หากมีการบันทึกข้อมูลความสัมพันธ์ r1 ก็ต้องมีการบันทึกข้อมูลความสัมพันธ์ r2 ของสมาชิกของชนิด entity A ด้วย

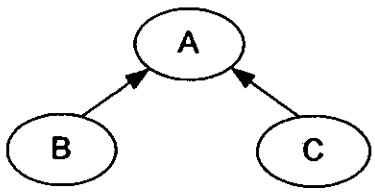
- Exclusion Constraints เป็นกฎข้อบังคับความถูกต้องที่มีลักษณะตรงข้ามกับ Equality constraints คือแสดงความสัมพันธ์ที่ระบุว่า หากมีความสัมพันธ์แบบหนึ่งเกิดขึ้น จะต้องไม่มีความสัมพันธ์อีกแบบหนึ่งเกิดขึ้นโดยเด็ดขาด ดังแสดงในรูป



รูปที่ 2.24 แสดงความสัมพันธ์แบบ Exclusion constraints

สามารถอธิบายได้ว่าแสดงถึงกฎข้อบังคับความถูกต้องของข้อมูลว่า หากมีการบันทึกข้อมูลความสัมพันธ์ r1 ของสมาชิกของชนิด entity A ใด จะต้องไม่มีการบันทึกข้อมูลความสัมพันธ์ r2 ของสมาชิกของชนิด entity A โดยเด็ดขาด

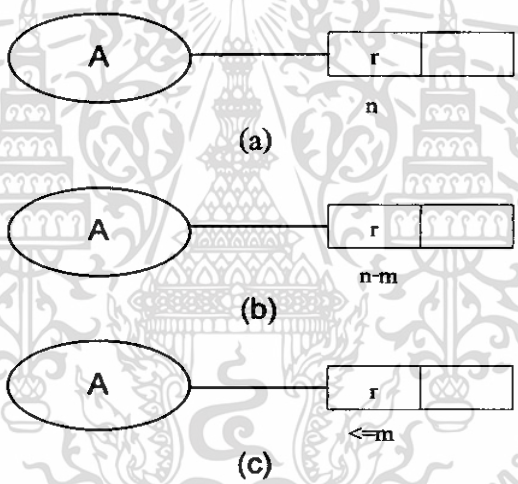
- Subtype constraints เป็นกฎข้อบังคับความถูกต้องที่ระบุถึงการแบ่งกลุ่มของสมาชิกของชนิด entity ที่มีอยู่อย่างชัดเจนซึ่งสมาชิกของชนิด entity ที่เป็น Super Type นั้น จะต้องมีลักษณะและคุณสมบัติที่แตกต่างกันอย่างชัดเจนดังแสดงในรูป



รูปที่ 2.25 แสดงความสัมพันธ์แบบ Subtype constraints

สามารถอธิบายได้ว่า สมาชิกของชนิด entity A โดยจะเรียกว่า Super Type นั้น สามารถแบ่งออกได้เป็น 2 กลุ่มคือ กลุ่มของชนิด entity B และกลุ่มของชนิด entity C ซึ่งเรียกว่า Subtype

- Occurrence frequency constraints เป็นกฎข้อบังคับความถูกต้องของข้อมูลที่ใช้ในการระบุจำนวนครั้งที่สมาชิกของชนิด entity ใดๆจะสามารถแสดงบทบาทใดบทบาทหนึ่งได้ ดังแสดงในรูป



รูปที่ 2.26 แสดงความสัมพันธ์แบบ Occurrence frequency constraints

สามารถอธิบายได้ว่า จากรูปที่ 2.29 (a) เป็นกฎข้อบังคับความถูกต้องของข้อมูล โดยที่แต่ละชนิด entity A จะมีการแสดงบทบาทในคอลัมน์ r เป็นจำนวน n ครั้ง จากรูปที่ 2.29 (b) เป็นกฎข้อบังคับความถูกต้องของข้อมูลโดยที่แต่ละชนิด entity A จะมีการแสดงบทบาทในคอลัมน์ r ได้อย่างน้อยที่สุด n ครั้ง และมากที่สุด m ครั้ง จากรูปที่ 2.30(c) เป็นกฎข้อบังคับความถูกต้องของข้อมูล โดยที่แต่ละชนิด entity A จะมีการแสดงบทบาทในคอลัมน์ r ได้อย่างน้อยที่สุด n ครั้ง

2.1.2.8 คีย์ (Key)

คุณสมบัติหนึ่งที่สำคัญของความสัมพันธ์ก็คือ ความเป็นเอกลักษณ์ (Uniqueness property) สิ่งที่ใช้กำหนดความเป็นเอกลักษณ์ของแถวในความสัมพันธ์ เรียกว่า คีย์ (key)ฐานข้อมูลหนึ่งๆ จะมีข้อมูลอยู่มากมาย ยิ่งฐานข้อมูลมีขนาดใหญ่ขึ้นก็จะมีข้อมูลจำนวนมากขึ้นเป็นเงาตามตัวข้อมูลเหล่านี้อาจมีค่าแตกต่างกัน คล้ายกัน หรือแม้กระทั่งเหมือนกัน ทำให้การแยกแยะโดยอาศัยเพียงตัวข้อมูลอย่างเดียวทำได้อย่างยากลำบาก ดังนั้นจึงมีการกำหนดค่า Keys ประจำข้อมูลเพื่อทำให้การแยกแยะข้อมูลในฐานข้อมูลเป็นไปอย่างถูกต้องคีย์มีหลายประเภท ได้แก่ คีย์หลัก, Secondary key, Foreign key, Candidate key, Super key ดังจะได้กล่าวในรายละเอียดต่อไป

- คีย์หลัก (Primary key) คือ Key หลักที่ใช้ในการอ้างถึง Entity ในฐานข้อมูล การเลือกคีย์หลักสามารถเลือกได้จากRecord ใดๆ ก็ได้ที่ไม่มีโอกาสซ้ำซ้อนกันบนฐานข้อมูลนั้น

เลขประจำตัวประชาชน	ชื่อ	นามสกุล	อายุ
3501552150054	สมชาย	แซ่ตั้ง	25
3210077565107	สมศรี	แซ่อึ้ง	42
4110597520235	สมชาย	แซ่ตั้ง	25
2156800512473	สมปอง	แซ่แต้	16
7812350453784	สมชัย	แซ่เล้ง	50

ตารางที่ 2.4 แสดง ข้อมูลทั่วไปของประชาชน

ตารางที่ 2.4 แสดง Entity ของประชาชนซึ่งประกอบด้วยเลขประจำตัวประชาชน ชื่อ นามสกุล และอายุ โดยจะสามารถเห็นได้ว่า นอกจาก Field เลขประจำตัวประชาชนแล้ว Field อื่นๆ คือชื่อ นามสกุล และอายุ อาจซ้ำกันได้ทั้งนั้น ในกรณีนี้ Field ที่เหมาะสมที่จะเป็นคีย์หลักที่สุดก็คือ Field เลขประจำตัวประชาชนนั่นเอง

คีย์หลักเป็นข้อมูลสำคัญที่จะทำให้การเข้าถึงข้อมูลบนฐานข้อมูลเป็นไปได้อย่างรวดเร็ว ดังนั้นผู้ใช้จึงควรกำหนดคีย์หลักให้ชัดเจนตั้งแต่ขั้นตอนออกแบบฐานข้อมูล หากไม่มีข้อมูลใดเลยในฐานข้อมูลที่เหมาะที่จะเป็นคีย์หลักก็ควรที่จะกำหนด เร็คคอร์ด (Record) ใหม่สำหรับให้เป็นคีย์หลัก

ชื่อ	นามสกุล	อายุ	เพศ
สมชาย	แซ่ตั้ง	25	ชาย
สมศรี	แซ่อึ้ง	42	หญิง
สมชาย	แซ่ตั้ง	25	ชาย
สมปอง	แซ่แต้	16	ชาย
สมชัย	แซ่เล้ง	50	ชาย

ตาราง ที่ 2.5 แสดงข้อมูลทั่วไปของพนักงานบริษัท

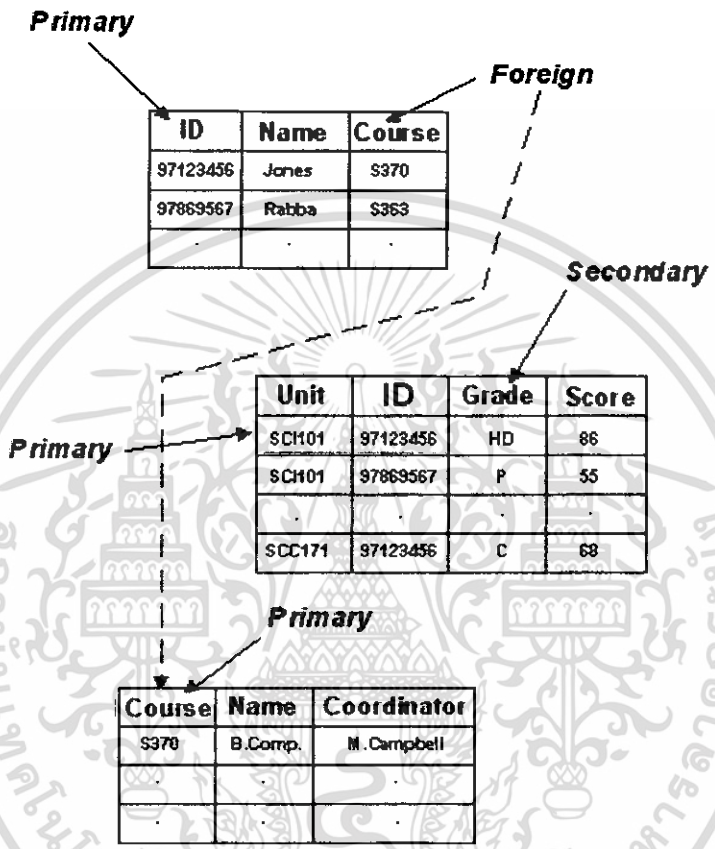
จากตารางที่ 2.5 จะเห็นได้ว่าฐานข้อมูลพนักงานบริษัทนี้ไม่มี ฟิลด์ (Field) ใดเลยที่เหมาะสมที่จะใช้เป็นคีย์หลัก ดังนั้น ผู้ออกแบบฐานข้อมูลจึงควรเพิ่ม ฟิลด์ เฉพาะสำหรับใช้เป็นคีย์หลักของฐานข้อมูลดังรูปที่ 2.32

ID	ชื่อ	นามสกุล	อายุ	เพศ
1	สมชาย	แซ่ตั้ง	25	ชาย
2	สมศรี	แซ่อึ้ง	42	หญิง
3	สมชาย	แซ่ตั้ง	25	ชาย
4	สมปอง	แซ่แต้	16	ชาย
5	สมชัย	แซ่เล้ง	50	ชาย

ตารางที่ 2.6 แสดง ข้อมูลทั่วไปของพนักงานบริษัทหลังจากเพิ่ม คีย์หลัก แล้ว

หลังจากเพิ่ม Field พิเศษคือ ID เข้าไปในตารางที่ 2 เพื่อใช้เป็นคีย์หลัก โดยเฉพาะ จะสามารถทำให้การอ้างถึงข้อมูลในฐานข้อมูลเป็นไปได้สะดวก และมีประสิทธิภาพมากขึ้น

โครงสร้างของคีย์



รูปที่ 2.27 โครงสร้างของคีย์

- คีย์รอง (Secondary Key) คือ คีย์เดี่ยวหรือคีย์ผสม (Single or Composite key) ซึ่งเมื่อใช้ในการค้นหาข้อมูลจากความสัมพันธ์จะได้มากกว่าหนึ่งเรคคอร์ด ต่างจากคีย์หลักที่ทำให้ข้อมูลในตารางไม่ซ้ำกัน ดังนั้นคีย์รองจึงไม่จำเป็นจะต้องเป็นเอกลักษณ์
- คีย์นอก (Foreign key) คือ คีย์เดี่ยวหรือคีย์ผสม ซึ่งปรากฏเป็นคีย์ทั่วไปของความสัมพันธ์หนึ่ง แต่ไปปรากฏเป็นอีกคีย์หลักในอีกความสัมพันธ์หนึ่ง คีย์นอกเป็นอีกคีย์หนึ่งที่มีความสำคัญมากใสฐานข้อมูลเชิงสัมพันธ์ เนื่องจากเป็นตัวที่ใช้สร้างการเชื่อมต่อระหว่างความสัมพันธ์ การเปลี่ยนแปลงค่าของคีย์

นอกจะต้องอาศัยความระมัดระวังเป็นอย่างมากเนื่องจากจะมีผลกระทบโดยตรงต่อข้อมูลในความสัมพันธ์อื่นที่มีการอ้างอิงถึงคีย์นอกตัวนี้ จึงมีกฎและเงื่อนไขที่บังคับใช้เพื่อทำให้ข้อมูลมีความถูกต้องอยู่เสมอ

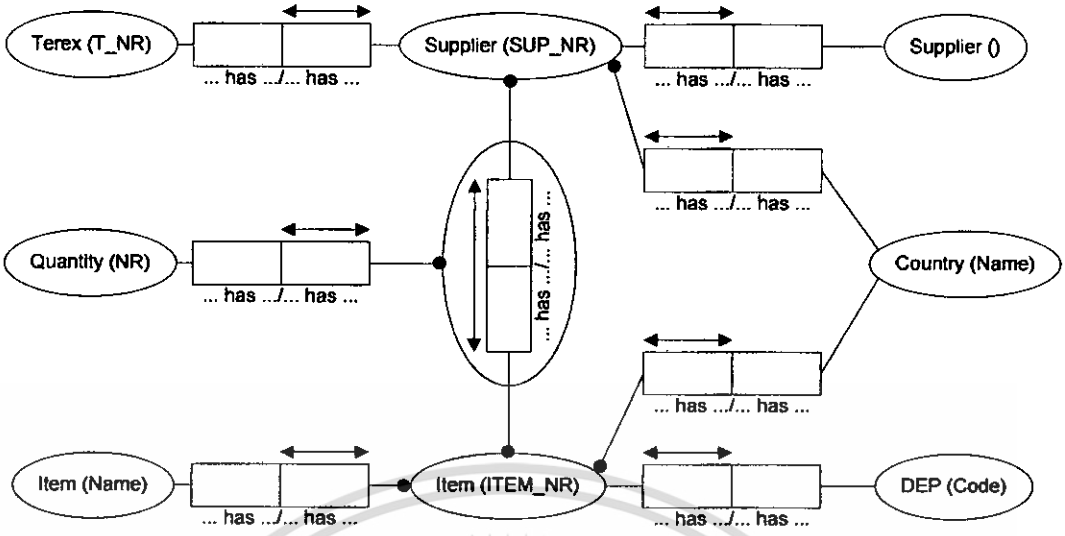
การกำหนดค่าให้กับคีย์นอกของความสัมพันธ์ที่อ้างอิงถึงจะต้องกำหนดค่าของคีย์ให้อยู่ในโดเมนเดียวกันกับความสัมพันธ์ที่คีย์นอกนั้นเป็นคีย์หลัก แต่คีย์นอกนั้นไม่จำเป็นจะต้องเป็นส่วนหนึ่งในคีย์หลักของความสัมพันธ์อื่น

- ซุปเปอร์คีย์ (Superkey) คือกลุ่มของแอททริบิวที่สามารถนำไปใช้ในการค้นหาข้อมูลที่เป็นเอกลักษณ์ได้
- คีย์แข่งขัน (Candidate key) คีย์แข่งขัน ก็คือ ซุปเปอร์คีย์ และไม่มีกลุ่มย่อยของคีย์ใดในคีย์แข่งขันที่จะสามารถเป็นซุปเปอร์คีย์ได้อีก

2.1.3 The Optimal Normal Form algorithm (ONF algorithm)

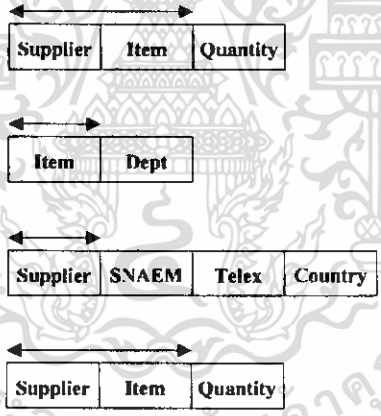
เป็นวิธีการจัดกลุ่มความจริงใน Conceptual Schema ให้เป็น Relational Database Schema โดยมีหลักการดังสรุปดังต่อไปนี้

- สร้าง 1 relation สำหรับชนิดความจริงแบบ binary ซึ่งมีความสัมพันธ์แบบ Many to many โดยที่ Unique Identifier ของชนิด entity ที่เกี่ยวข้องทั้งหมดเป็น Primary key
- สร้าง 1 relation สำหรับชนิดความจริงแบบ n-ary โดยที่ Unique Identifier ของชนิด entity ซึ่งมี Role ถูกบังคับด้าย Uniqueness Constraint เดียวกันเป็น Candidate key
- พิจารณาชนิด entity ที่เกี่ยวข้องกับความจริงแบบ binary ที่มีความสัมพันธ์เป็นแบบ One to one หรือ Many to one โดยที่ role ของชนิด entity เหล่านั้นถูกบังคับด้วย Uniqueness constraint ให้สร้างrelation โดยมี Unique Identifier ของชนิด entity เหล่านั้นเป็น Candidate key



รูปที่ 2.28 แสดงตัวอย่างจำลองข้อมูล (Conceptual Schema)

จาก Conceptual Schema ในรูปที่ 2.28 เมื่อใช้ ONF algorithm จะได้ relation ดังต่อไปนี้



รูปที่ 2.29 แสดง relation ของแบบจำลองรูปที่ 2.28

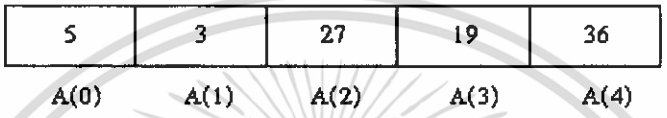
โดยมีเครื่องหมาย ↔ Attribute หรือกลุ่มของ Attribute ที่เป็น Primary key และเครื่องหมาย บน Candidate key ↔

2.2 โครงสร้างข้อมูล (Data Structure)

2.2.1 Arrays and Pointe

2.2.1.1 Array

เป็นโครงสร้างข้อมูลที่ประกอบด้วยกลุ่มของตัวแปรที่เป็นค่าอะไรก็ได้ แต่ต้องมีชนิดเดียวกัน ข้อมูลแต่ละตัวใน Array เรียกว่าสมาชิกของ array การอ้างถึงจะเรียกชื่อตัวแปร array แล้วตามด้วยหมายเลขดัชนี (index) ของสมาชิกที่ต้องการอ้างถึง โดยสมาชิกตัวแรกจะแทนด้วยลำดับหมายเลข 0 สมาชิกตัวที่2 แทนด้วยลำดับหมายเลข 1 ..จนถึงลำดับที่ n



รูปที่ 2.30 รูปแบบการเก็บข้อมูลในแบบ Array

การประกาศค่าตัวแปร array แบบ static คือ การกำหนดจำนวนสมาชิกของ array ตั้งแต่ตอนประกาศตัวแปร array จากในรูปเป็นมีการกำหนดค่าให้กับสมาชิกใน Array A(4) กำหนดดังนี้

A(0) = 5

A(1) = 3

A(2) = 27

A(3) = 19

A(4) = 36

2.2.1.2 Array หลายมิติ

(Multidimontional Array) ประกอบด้วยแถวลำดับที่มี index อ้างอิงหลายตัว ตัวอย่าง Array 2 มิติ (two dimontional Array) อยู่ในรูปของตารางคือมี index หลักและ index แถว อยู่ในรูปชื่อตัวแปรArray (index หลัก, index แถว) A (2, 4)

6 A(0,0)	22 A(1,0)	10 A(2,0)
32 A(0,1)	16 A(1,1)	11 A(2,1)
7 A(0,2)	916 A(1,2)	76 A(12,2)
9 A(0,3)	2 A(1,3)	46 A(2,3)

รูปที่ 2.31 รูปแบบการเก็บข้อมูลในแบบ Multidimontional Arrays

จากรูปตัวแปร Array A มีจำนวนหลัก (Column) ของตาราง 3 หลักมีจำนวนแถว มีจำนวนแถว (Row) 4 หลัก (เนื่องจากในArray สมาชิกแต่ละตัวจะนับเริ่มจาก 0)

A(0,0) คือ สมาชิก Array A หลักที่ 0 แถวที่ 0 มีค่า 6

A(1,0) คือ สมาชิก Array A หลักที่ 1 แถวที่ 0 มีค่า 22

A(2,0) คือ สมาชิก Array A หลักที่ 2 แถวที่ 0 มีค่า 10

A(0,1) คือ สมาชิก Array A หลักที่ 0 แถวที่ 1 มีค่า 32

A(1,3) คือ สมาชิก Array A หลักที่ 1 แถวที่ 3 มีค่า 2 เป็นต้น

2.2.1.3 การคำนวณเนื้อที่สำหรับเก็บ Array

มีค่าพารามิเตอร์ดังนี้

B = ตำแหน่งหน่วยความจำเริ่มต้น (Base Address)

S = ขนาดข้อมูลของสมาชิกแต่ละตัว (Component Size)

d = จำนวนมิติของ Array (dimension)

L = ค่าต่ำสุดของดัชนี (Lower Bound)

U = ค่าสูงสุดของดัชนี (Upper Bound)

สูตร

$$S * (U1-L1+1) * (U2-L2+1) * \ldots.*(Ud-Ld+1) \qquad (2.1)$$

ตัวอย่าง การคำนวณเนื้อที่สำหรับจัดเก็บ Array A(0..4) โดยเนื้อที่ในการจัดเก็บ Integer = 4 ไบต์ (S=4) ดังนั้นการจัดเก็บจะใช้เนื้อที่ 4(4-0+1) = 20 ไบต์

ตำแหน่งในหน่วยความจำ	ดัชนี	ค่าของข้อมูล
100=(100+0*4)	[0]	5
104=(100+1*4)	[1]	3
108=(100+2*4)	[2]	27
112=(100+3*4)	[3]	19
116=(100+4*4)	[4]	36

ตารางที่ 2.7 แสดงค่าการคำนวณเนื้อที่สำหรับจัดเก็บ Array

2.2.1.4 พอยน์เตอร์ Pointer

การจัดสรรเนื้อที่ในหน่วยความจำแบ่งเป็น 2 แบบคือ

1. แบบสแตติก คือ เนื้อที่ในหน่วยความจำถูกจัดสรรหรือจองไว้สำหรับตัวแปร ก่อนการประมวลผล (execute) นั่นคือระหว่างการแปล (compile) ตัวแปรที่ได้ถูกนิยามไว้แล้วจะได้รับการจัดสรรหน่วยความจำทันทีที่ประกาศตัวแปร ถึงแม้ว่าตัวแปรจะยังไม่ได้ถูกใช้ในการเขียนคำสั่ง ซึ่งทำให้เสียเนื้อที่ในหน่วยความจำไปArray มีการจัดสรรเนื้อที่ หน่วยความจำแบบสแตติก เนื้อที่ในหน่วยความจำจะถูกจองไว้เท่ากับขนาดที่กำหนดไว้ใน Array

ตัวอย่างเช่น int array[] = new int[50];

ถึงแม้ตอนรันโปรแกรมกับข้อมูลจริงจะมีข้อมูลสำหรับ Array A เพียง 35 ตัว แต่ก็ต้องเสียเนื้อที่ในหน่วยความจำเท่ากับที่จองไว้ตั้งแต่ตอนประกาศ คือ 50 ตัว ซึ่งทำให้เราสูญเสียเนื้อที่ในหน่วยความจำไปโดยเปล่าประโยชน์

2. แบบไดนามิก คือ ไม่ต้องกำหนดขนาดของข้อมูลก่อนการ compile สามารถเรียกใช้เนื้อที่หน่วยความจำได้ขณะทำงาน เนื่องจากไม่ต้องจองเนื้อที่หน่วย ความจำไว้มากเกินความจำเป็นที่จะต้องใช้เป็นการจัดสรรเนื้อที่หน่วยความจำให้กับตัวแปรระหว่างประมวลผล ซึ่งตัวแปรที่มีคุณสมบัติดังกล่าวคือ ตัวแปรแบบพอยน์เตอร์ (pointer) จากการใช้ Array ซึ่งมีการจองเนื้อที่ในหน่วยความจำ แบบสแตติกนี้เอง ทำให้การจองเนื้อที่ในหน่วยความจำไม่มีประสิทธิภาพ เช่น ต้องประกาศจำนวนเนื้อที่ที่ต้องการจองไว้มาก ถึงแม้ว่าเมื่อใช้งานจริงอาจจะใช้ไม่ถึง หรือว่าหากในการใช้งานจริงใช้เนื้อที่มากกว่าที่จองไว้ก็จะ

ทำให้เกิดการ overflow ได้นอกจากนั้นยังมีปัญหาอันเนื่องมากจากการที่ Array เป็นโครงสร้างแบบเรียงลำดับ (sequential) เช่น กรณีที่มีเรคคอร์ด 100 เรคคอร์ด หากเกิดต้องมีการลบข้อมูลเรคคอร์ดที่ 35 เรคคอร์ดที่อยู่ในลำดับที่ 36 – 100 ก็จะต้องถูกเลื่อนลำดับทั้งหมดภายใน Array คือ จากลำดับที่ 36 ก็มาแทนที่ เรคคอร์ด 35 จากลำดับที่ 37 ก็มาแทนที่ 36 ตามลำดับ และเช่นเดียวกันเมื่อต้องมีการแทรกเรคคอร์ดใหม่เข้าไปตรงกลาง Array เช่น เพิ่มเรคคอร์ดหลังลำดับที่ 65 ดังนั้นเรคคอร์ดลำดับ 66 ถึง 100 ก็ต้องขยับออกไป

2.2.2 Sorting and Searching

ถ้าเราจำเป็นต้องเก็บและค้นหาข้อมูลอยู่เป็นประจำ การเก็บข้อมูลเราก็ต้องจัดเก็บให้เป็นระเบียบ และง่ายในกระบวนการค้นหาข้อมูลเพื่อนำมาใช้ใหม่ เช่นการจัดเรียงหมวดหมู่ของหนังสือในห้องสมุด ต้องมีการจัดการกับรายละเอียดของหนังสือต่างๆ ให้เป็นแฟ้มข้อมูลที่เรียงลำดับตามตัวอักษร เป็นต้นและในการประกอบการต่างๆ ที่เกี่ยวข้องกับการใช้งานด้านคอมพิวเตอร์ ก็มีการจัดเรียงลำดับของข้อมูล โดยวิธีใดวิธีหนึ่งแล้วแต่กำหนด ทำไมเราจึงต้องศึกษาการเรียงลำดับข้อมูล (Sorting) เพื่อช่วยในการออกแบบอัลกอรึทึม เพราะการเรียงลำดับข้อมูล เป็นงานพื้นฐานที่ใช้ในโปรแกรมประยุกต์ต่างๆ ช่วยทำให้การเรียกใช้งานข้อมูลนั้นๆ มีความสะดวก รวดเร็ว ในการค้นหา มากกว่าการเรียกใช้ข้อมูลที่ไม่มีการลำดับ และทำให้เกิดเทคนิคใหม่ๆที่น่าสนใจและสำคัญอันได้แก่ partial order , recursion , merge , lists , การจัด tree ในอาเรย์เป็นต้น อีกทั้ง อัลกอรึทึมที่ใช้เพื่อการเรียงลำดับข้อมูลแต่ละตัวมีข้อดีและข้อเสียแตกต่างกัน ขึ้นอยู่กับจำนวน ชนิดของข้อมูล การกำหนดค่าเริ่มต้น ขนาด และค่าของข้อมูลที่จะทำการเรียงลำดับ สิ่งสำคัญก็คือ เราต้องรู้ว่าเรามีความต้องการอย่างไรเพื่อที่สามารถจะเลือกอัลกอรึทึมที่มีความเหมาะสม และสอดคล้องกับงานของเราได้

การเรียงข้อมูล สามารถแบ่งได้เป็น 2 ประเภทด้วยกันคือ

1. การเรียงข้อมูลแบบภายใน (Internal Sorting) คือ การเรียงลำดับข้อมูลโดยทั้งหมดต้องจัดเก็บอยู่ในหน่วยความจำหลัก (main memory) ที่มีการเข้าถึงข้อมูลได้เร็ว โดยไม่จำเป็นต้องใช้หน่วยความจำสำรอง เช่น ดิสค์ หรือเทปสำหรับการจัดเก็บชั่วคราว ใช้ในกรณีที่ข้อมูลไม่มากเกินกว่าพื้นที่ความจำที่กำหนดให้กับผู้ใช้แต่ละราย

2. การเรียงข้อมูลแบบภายนอก (External Sorting) คือ การเรียงลำดับข้อมูลที่มีขนาดใหญ่เกินกว่าที่จะสามารถเก็บไว้ใน พื้นที่ความจำหลักที่กำหนดให้ได้ในคราวเดียว ดังนั้นข้อมูลส่วนมากต้องเก็บไว้ในไฟล์ข้อมูลที่อยู่บนดิสค์เทป เป็นต้น สำหรับการเรียงข้อมูลแบบภายนอก จะต้องคิดถึงเวลาที่ใช้ในการถ่ายเทข้อมูล จากหน่วยความจำชั่วคราวกับหน่วยความจำหลัก ด้วยเช่นกัน

อัลกอรึทึมสำหรับการเรียงข้อมูล จัดได้ 3 ประเภทใหญ่ๆ ดังนี้

- การเรียงลำดับแบบแลกเปลี่ยน (Exchange Sort)
- การเรียนลำดับแบบแทรก (Insertion Sort)
- การเรียงลำดับแบบเลือก (Selection Sort)

2.2.2.1 Sorting Algorithms

หลักของการเรียงแบบนี้คือ จะเปรียบเทียบและแลกเปลี่ยนข้อมูล 2 ค่าที่อยู่ติดกันในลักษณะที่เรากำหนด เช่น จากน้อยไปมาก หรือจากมากไปน้อย โดยจะทำการเปรียบเทียบข้อมูลทั้งชุดจนกว่าจะมีการเรียงตามลำดับทั้งหมด ขั้นตอนการทำงานของอัลกอริทึม

เริ่มต้น :	5	4	1	2	3
หลังจากผ่านรอบที่ 1 :	1	5	4	2	3
หลังจากผ่านรอบที่ 2 :	1	2	5	4	3
หลังจากผ่านรอบที่ 3 :	1	2	3	5	4
หลังจากผ่านรอบที่ 4 :	1	2	3	4	5

รูปที่ 2.32 การเรียงลำดับแบบ Sorting Algorithms

การเรียงลำดับในลักษณะนี้ เป็นการปรับปรุงมาจากการเรียงลำดับแบบ Bubble เพื่อให้การเรียงลำดับเร็วขึ้น วิธีนี้เหมาะกับการเรียงข้อมูลที่มีจำนวนมาก หรือมีขนาดใหญ่ และเป็นวิธีการเรียงข้อมูลที่ให้ค่าเฉลี่ยของเวลาน้อยที่สุดเท่าที่ค้นพบวิธีหนึ่ง

- การเรียงลำดับแบบ Quick Sort จะเป็นการเปรียบเทียบสมาชิกที่ไม่อยู่ติดกัน โดยกำหนดข้อมูลค่าหนึ่ง เพื่อแบ่งชุดข้อมูลที่ต้องการเรียงลำดับออกเป็น 2 ส่วน จากนั้นก็จะทำการแบ่งย่อยชุดข้อมูล 2 ส่วนนั้นลงไปอีก ทำแบบนี้ไปเรื่อยๆจนข้อมูลแต่ละชุดมีสมาชิกเหลือเพียงตัวเดียวและทำให้ชุดข้อมูลทั้งหมดมีการเรียงลำดับ

รูปที่ 2.33 แสดงการเรียงลำดับแบบ Quick Sort

- Insertion Sort การเรียงลำดับที่ง่ายไม่ซับซ้อน เป็นการนำข้อมูลใหม่เพิ่มเข้าไปในชุดข้อมูลที่มีการเรียงลำดับอยู่แล้ว โดยข้อมูลใหม่ที่นำเข้ามาจะแทรกอยู่ในตำแหน่งทางขวาของชุดข้อมูลเดิมและยังคงทำให้ข้อมูลทั้งหมด มีการเรียงลำดับ วิธีนี้เริ่มต้นโดยการเรียงลำดับข้อมูล 2 ตัวแรกของชุดข้อมูล หลังจากนั้นเพิ่มข้อมูลตัวที่ 3 เข้ามา จะมีการเปรียบเทียบค่ากับข้อมูล 2 ตัวแรก และแทรกอยู่ในตำแหน่งที่เหมาะสม และสำหรับการเพิ่มข้อมูลตัวต่อๆไปก็จะทำเหมือนเดิมจนข้อมูลทุกตัวมีการเรียงลำดับ ขั้นตอนการทำงานของอัลกอริทึม

เริ่มต้น :	5	4	1	2	3
หลังจากผ่านรอบที่ 1 :	4	5	1	2	3
หลังจากผ่านรอบที่ 2 :	1	4	5	2	3
หลังจากผ่านรอบที่ 3 :	1	2	4	5	3
หลังจากผ่านรอบที่ 4 :	1	2	3	4	5

รูปที่ 2.34 แสดงการเรียงลำดับแบบ Insertion Sort

- Heap sort จัดว่าเป็นการเรียงลำดับแบบเลือกที่ใช้หลักการแบบไบนารีทรี สร้างฮีพขึ้นมา กล่าวคือ "ฮีพ" เป็นไบนารีทรีในแบบที่เกือบสมบูรณ์ที่มีคุณสมบัติว่าข้อมูลในโหนดพ่อแม่จะมากกว่าหรือเท่ากับข้อมูลในโหนดลูก

ดังนั้นต้นไม้ฮีพ ขนาด n ใดๆ เป็นต้นไม้ที่ประกอบด้วยข้อมูล h1, h2,…,hn ดังแสดงในภาพข้างล่างนี้ และได้ว่า hi >= h2I and hi >= h2I+1 when I = 1, 2, …,n จะพบว่า h1 เป็นข้อมูลที่มีค่ามากที่สุดในฮีพ

การเรียงลำดับแบบฮีพ มี 2 ขั้นตอน คือ

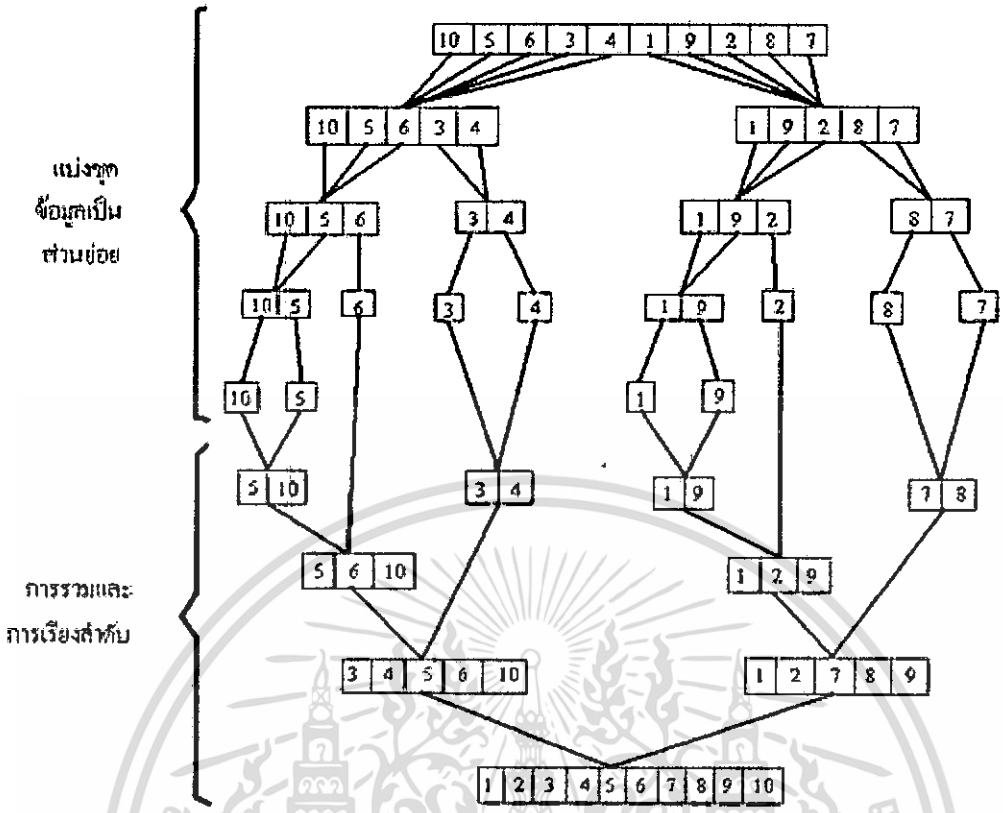
1.การสร้างฮีพ

2.การปรับค่าภายในฮีพ

การสร้างฮีพ เป็นการรับข้อมูลเข้าไปในฮีพ โดยทุกครั้งที่มีการรับข้อมูลเข้ามาให้ทำเสมือนว่าเป็นต้นไม้สมบูรณ์ หรือต้นไม้เกือบสมบูรณ์ และยังรักษาคุณสมบัติของต้นไม้ฮีพไว้ ดังตัวอย่างทางด้านขวามือ

การปรับค่าภายในฮีพ หลังจากสร้างฮีพเสร็จแล้ว จะพบว่าข้อมูลที่ h1 เป็นข้อมูลที่มีค่ามากที่สุดและข้อมูลภายในฮีพยังไม่มีการเรียงลำดับ เมื่อมีการนำข้อมูลที่มีค่ามากที่สุดออกไปเก็บไว้ในแถวลำดับช่องสุดท้าย (หรือในโหนดสุดท้ายของฮีพ) จะต้องมีการปรับค่าภายในฮีพ โดยการนำข้อมูลที่มีค่ามากที่สุดตัวต่อไปมาไว้ที่ h1 และนำข้อมูลไปเก็บไว้ในแถวลำดับก่อนช่องสุดท้าย (หรือในโหนดที่อยู่ก่อนโหนดสุดท้ายของฮีพ) หลังจากนั้นทำเหมือนเดิม จนกระทั่งขนาดของฮีพเป็นศูนย์ และผลลัพธ์ที่ได้คือแถวลำดับช่องแรก (หรือในโหนดแรกของฮีพ) จะเก็บข้อมูลที่มีค่าน้อยที่สุด และแถวลำดับช่องสุดท้าย (หรือในโหนดสุดท้ายของฮีพ) จะเก็บข้อมูลที่มีค่ามากที่สุด

- การเรียงลำดับแบบผสาน (Merge Sort) การเรียงลำดับแบบผสานเป็นลักษณะการเรียงลำดับแบบภายนอก และใช้กับชุดข้อมูลที่มีขนาดใหญ่ โดยมีหลักการทำงานคือ การแบ่งชุดข้อมูลออกเป็นส่วนย่อยๆ 2 ส่วนหรือมากกว่า และทำการเรียงลำดับในแต่ละส่วน หลังจากนั้นนำส่วนที่เรียงลำดับแต่ละส่วนมารวมกันเป็นแถวลำดับที่เรียงเรียบร้อยแล้ว



รูปที่ 2.35 แสดงการเรียงลำดับแบบ Merge Sort

2.2.2.2 Searching

การค้นหาข้อมูลแบบคงที่ (Static Searching)

จากที่เราได้ศึกษากันมาในเรื่องของการเก็บข้อมูลและการจัดเรียงข้อมูล เมื่อเราต้องการนำข้อมูลที่จัดเก็บเหล่านั้นมาใช้จำเป็นที่จะต้องค้นหาข้อมูลเพื่อดึงเอาสิ่งที่ต้องการมานำเสนอ ตัวอย่างเช่นการค้นหาข้อมูลจากสมุดโทรศัพท์โดยวิธีการค้นหาที่ได้นำมาแสดงเป็นการค้นหาแบบ Static search หรือการค้นหาโดยที่ข้อมูลไม่มีการเปลี่ยนแปลงตัวอย่างเช่นการค้นหาข้อมูลใน CD-Rom เป็นต้นโดยมี 2 วิธีที่นำมาแสดงคือ

- **การค้นหาแบบเรียงลำดับ (Sequential Search)**

หากข้อมูลที่ต้องการค้นหาไม่ได้ถูกเรียงลำดับมาก่อน เทคนิคที่ใช้การค้นหาข้อมูลจะมีไม่มากนักซึ่งหนึ่งในวิธีนั้นก็คือการค้นหาแบบเรียงลำดับ ซึ่งเป็นการค้นหาแบบลิเนียร์ โดยการเปรียบเทียบข้อมูลทีละตัวจนกว่าจะเจอข้อมูลที่ต้องการ

ทำให้ Big-O ของวิธีการนี้มีค่าเท่ากับ O(N) โดยมีวิธีที่ใช้ในการค้นหาแบบเรียงลำดับคือ

1. เริ่มต้นกำหนดข้อมูลที่ต้องการจะค้นหา
2. ทำการค้นหาข้อมูลที่ต้องการ โดยเริ่มตั้งแต่ข้อมูลแรกสุด
3. ทำการเปรียบเทียบข้อมูลว่าตรงกับที่ต้องการหรือไม่
4. หากข้อมูลไม่ตรงกับที่ต้องการ ให้เลื่อนไปยังตำแหน่งถัดไปเพื่อเปรียบเทียบ จนกว่าจะเจอข้อมูลที่ต้องการ

จะเห็นว่าในกรณีที่ข้อมูลที่เราต้องการค้นหาอยู่ลำดับท้ายสุด จะทำให้เราต้องเปรียบเทียบกับข้อมูลทุกตัวเพื่อทดสอบทำให้การใช้เวลาในการค้นหานานเมื่อข้อมูลมีจำนวนมาก

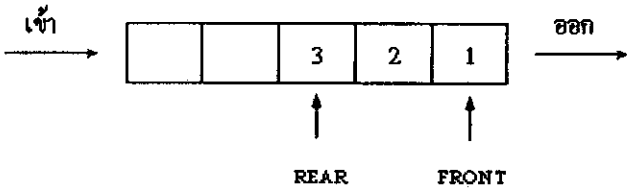
- **การค้นหาแบบไบนารีเสิร์ช (Binary Search)**

เมื่อข้อมูลที่เราต้องการค้นหาถูกเรียงลำดับมาก่อนแล้ว เราสามารถใช้เทคนิคการค้นหาที่มีประสิทธิภาพสูงขึ้นมาได้โดยใช้วิธีการค้นหาแบบไบนารีเสิร์ช โดยการเริ่มต้นค้นหาที่ตรงกลางข้อมูล แทนที่จะเป็นต้น หรือท้ายข้อมูลแบบการค้นหาแบบเรียงลำดับ โดยใช้หลักการของไบนารีเสิร์ชทรี เมื่อเทียบกับข้อมูลในตำแหน่งตรงกลางแล้ว หากข้อมูลที่ต้องการค้นหามีค่าน้อยกว่า จะเริ่มทำการค้นหาข้อมูลอีกครั้งในช่วง low ถึง mid-1 หากข้อมูลที่ต้องการค้นหามีค่ามากกว่าจะทำการเริ่มต้นค้นหาข้อมูลอีกครั้งในช่วง mid-1 ถึง high จากนั้นทำการค้นหาแบบเดิมไปจนกว่าจะเจอกับค่าที่ต้องการ

2.2.3 Queues and Stacks

2.2.3.1 คิว Queues

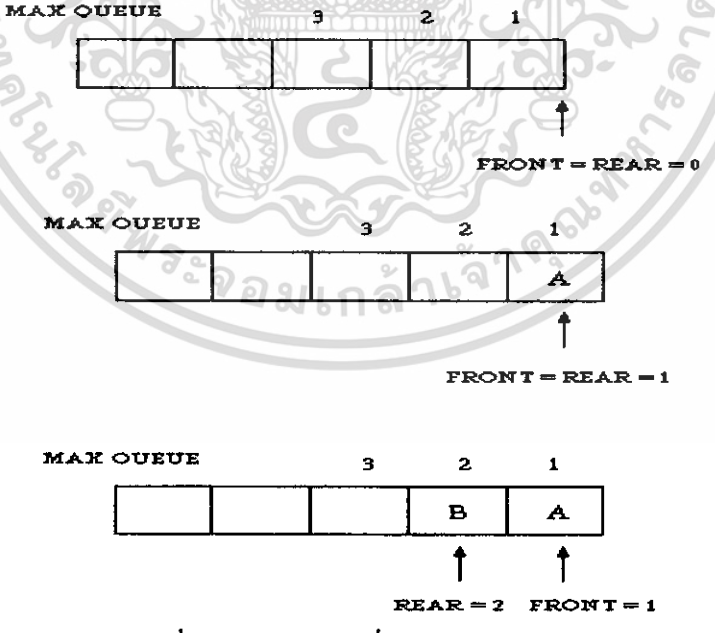
คิวเป็นโครงสร้างข้อมูลแบบลำดับ (Sequential) ลักษณะของคิวเราสามารถพบได้ในชีวิตประจำวัน เช่น การเข้าแถวตามคิวเพื่อรอรับบริการต่างๆ ลำดับการสั่งพิมพ์งาน เป็นต้น ซึ่งจะเห็นได้ว่าลักษณะของการทำงานจะเป็นแบบใครมาเข้าคิวก่อน จะได้รับบริการก่อน เรียกได้ว่าเป็นลักษณะการทำงานแบบ FIFO (First In , First Out) ลักษณะของคิว จะมีปลายสองข้าง ซึ่งข้างหนึ่งจะเป็นช่องทางสำหรับข้อมูลเข้าที่เรียกว่า REAR และอีกข้างหนึ่งซึ่งจะเป็นช่องทางสำหรับข้อมูลออก เรียกว่า FRONT



รูปที่ 2.36 แสดง Queues แบบ FIFO

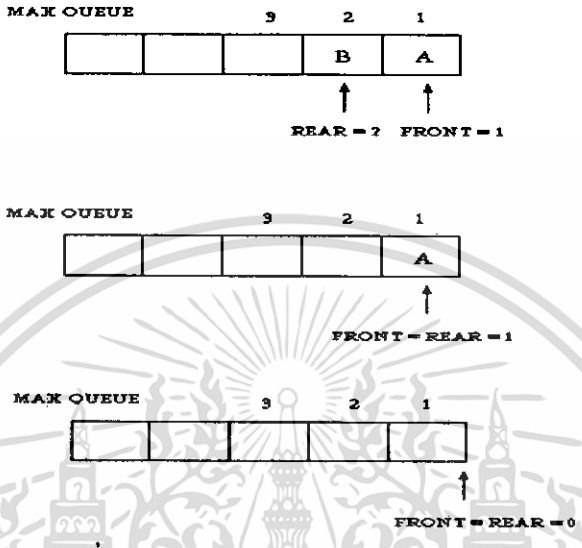
ในการทำงานกับคิวที่ต้องมีการนำข้อมูลเข้าและออกนั้น จะต้องมีการตรวจสอบว่าคิวว่างหรือไม่ เมื่อต้องการนำข้อมูลเข้า เพราะหากคิวเต็มก็จะไม่สามารถทำการนำข้อมูลเข้าได้ เช่นเดียวกัน เมื่อต้องการนำข้อมูลออกก็ต้องตรวจสอบด้วยเช่นกัน ว่าในคิวมีข้อมูลอยู่หรือไม่ หากคิวไม่มีข้อมูลก็จะไม่สามารถนำข้อมูลออกได้เช่นกัน

- การกระทำกับคิว การจะเพิ่มข้อมูลเข้าไปในคิว จะกระทำที่ตำแหน่ง REAR หรือท้ายคิว และก่อนที่จะเพิ่มข้อมูลจะต้องตรวจสอบก่อนว่าคิวเต็มหรือไม่ โดยการเปรียบเทียบค่า REAR ว่า เท่ากับค่า MAX QUEUE หรือไม่ หากว่าค่า REAR = MAX QUEUE แสดงว่าคิวเต็มไม่สามารถเพิ่มข้อมูลได้ แต่หากไม่เท่า แสดงว่าคิวยังมีที่ว่างสามารถเพิ่มข้อมูลได้ เมื่อเพิ่มข้อมูลเข้าไปแล้ว ค่า REAR ก็จะเป็นค่าตำแหน่งท้ายคิวใหม่



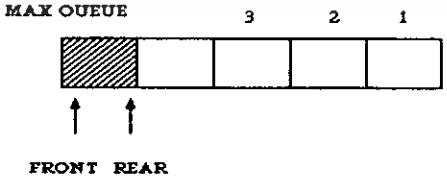
รูปที่ 2.37 แสดงการเพิ่ม Queue ข้อมูล

- การนำข้อมูลออกจากคิว การนำข้อมูลออกจากคิวจะกระทำที่ตำแหน่ง FRONT หรือส่วนที่เป็นหัวของคิว โดยก่อนที่จะนำข้อมูลออกจากคิวจะต้องมีการตรวจสอบก่อนว่ามีข้อมูลอยู่ในคิวหรือไม่ หากไม่มีข้อมูลในคิวหรือว่าคิวว่าง ก็จะไม่สามารถนำข้อมูลออกจากคิวได้



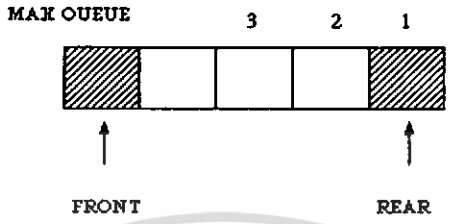
รูปที่ 2.38 แสดงการนำข้อมูลออกจาก Queue

- คิวแบบวงกลม (Circular Queue) คิวแบบวงกลมจะมีลักษณะเหมือนคิวธรรมดา คือ มีตัวชี้ FRONT และ REAR ที่แสดงตำแหน่งหัวคิวและท้ายคิวตามลำดับ โดย FRONT และ REAR จะมีการเลื่อนลำดับทุกครั้งเมื่อมีการนำข้อมูลเข้าและออกจากคิว แต่จะแตกต่างจากคิวธรรมดาก็คือ คิวธรรมดาเมื่อ REAR ชี้ที่ตำแหน่งสุดท้ายของคิว จะทำให้เพิ่มข้อมูลเข้าในคิวอีกเมื่อไม่ได้ เนื่องจาก ค่า REAR=MAX QUEUE ซึ่งแสดงว่าคิวนั้นเต็ม ไม่สามารถเพิ่มข้อมูลเข้าไปได้อีก ทั้งๆ ที่ยังมีเนื้อที่ของคิวเหลืออยู่ก็ตาม ทำให้การใช้เนื้อที่ของคิวไม่มีประสิทธิภาพ



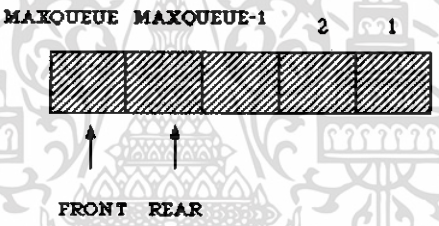
รูปที่ 2.39 แสดงคิวที่ค่า REAR ชี้ที่ตำแหน่งสุดท้ายของคิวทำให้ไม่สามารถนำข้อมูลเข้าได้อีก

สำหรับคิวแบบวงกลม จะมีวิธีจัดการกับปัญหานี้ คือ เมื่อมีข้อมูลเพิ่มเข้ามาในคิว ในลักษณะดังกล่าว คือ ขณะที่ REAR ชี้ตำแหน่งสุดท้ายของคิว ถ้าหากมีการเพิ่มค่าของ REAR REAR จะสามารถวนกลับมาชี้ยังตำแหน่งแรกสุดของคิวได้ ซึ่งจะทำให้คิวมีลักษณะเป็นแบบวงกลม



รูปที่ 2.40 แสดงคิวแบบวงกลมที่ REAR สามารถวนกลับมาชี้ที่ตำแหน่งแรกสุดของคิว

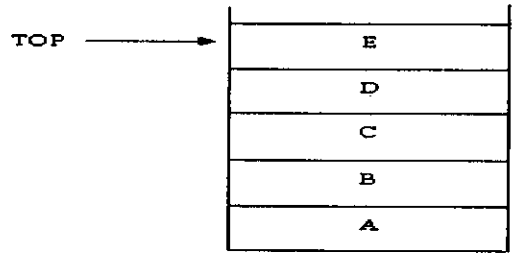
คิวแบบวงกลมจากที่กล่าวมาสามารถเพิ่มข้อมูลได้อีก จนกว่าคิวจะเต็มซึ่งแสดงดังรูป



รูปที่ 2.41 แสดง คิวแบบวงกลมเต็ม Queue

2.2.3.2 สแตค (Stack)

สแตคเป็นโครงสร้างข้อมูลที่มีลักษณะแบบลำดับ (sequential) คือการกระทำกับข้อมูลจะกระทำที่ปลายข้างเดียวกันที่ส่วนปลายสุดของสแตค การกระทำกับข้อมูลของสแตคประกอบไปด้วยการนำเข้าข้อมูลเข้า (PUSH) ที่ส่วนบนสุดของสแตค และการนำข้อมูลออก (POP) ที่ส่วนบนสุดของสแตคเช่นกัน ในการจะ Push ข้อมูลเข้าก็ต้องตรวจสอบด้วยว่าข้อมูลในสแตคเต็มหรือไม่ หากสแตคเต็มก็จะไม่สามารถ Push หรือนำข้อมูลเข้าได้ เช่นเดียวกับการ Pop ข้อมูลออกก็ต้องตรวจสอบด้วยว่ามีข้อมูลอยู่ในสแตคหรือไม่ หากไม่มีข้อมูลอยู่ในสแตคหรือสแตคว่าง (empty stack) ก็ไม่สามารถ pop ได้ การนำข้อมูลเข้า-ออก จากสแตค (push , pop) จะมีลักษณะแบบเข้าหลัง ออกก่อน (LIFO : Last In , First Out) คือ ข้อมูลที่เข้าไปในสแตคลำดับหลังสุด จะถูกนำข้อมูลออกจากสแตคเป็นลำดับแรก ยกตัวอย่างการทำงานแบบ LIFO

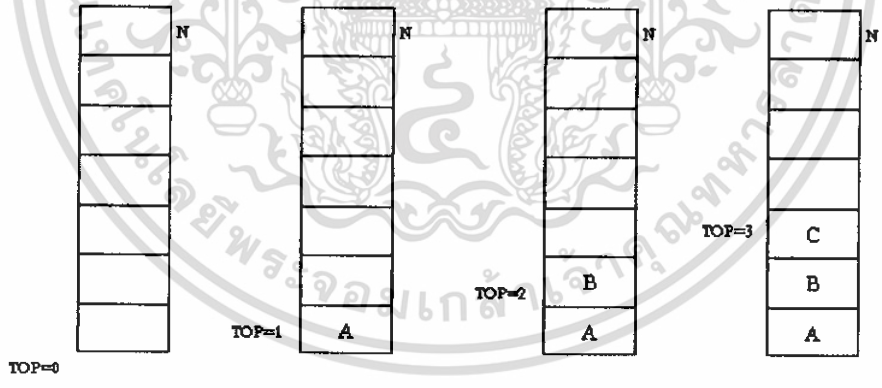


รูปที่ 2.42 แสดงลักษณะของสแตค ที่ประกอบด้วยข้อมูล A , B , C , D และ E มี TOP ที่ชี้ที่สมาชิกตัวบนสุด ของสแตค

Operation ของสแตค

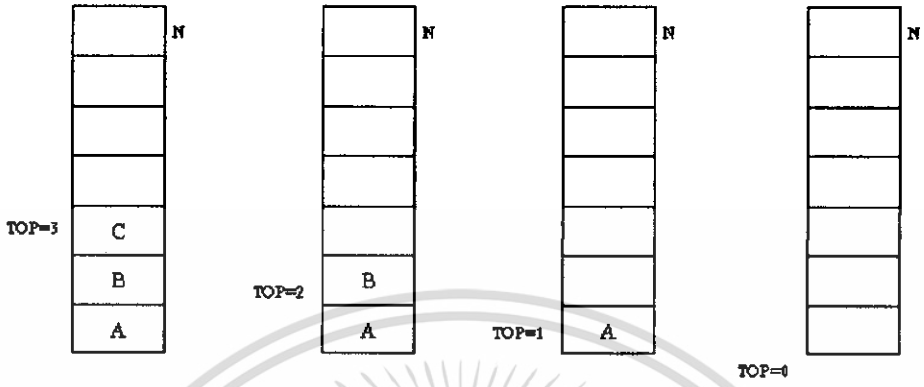
- การเพิ่มข้อมูลลงในสแตค (pushing stack)
- การดึงข้อมูลออกจากสแตค (popping stack)

1. การเพิ่มข้อมูลลงในสแตค การเพิ่มข้อมูลลงในสแตค คือ การนำเข้ามูลเข้าสู่สแตคโดยทับข้อมูลที่อยู่บนสุดของสแตค ข้อมูลจะสามารถนำเข้าได้เรื่อยๆ จนกว่า สแตคจะเต็ม สมมติว่าสแตคจองเนื้อที่ไว้ N ตัว ถ้าหากค่า TOP เป็น 0 แสดงว่า สแตคว่าง หากค่า TOP = N แสดงว่าสแตคเต็มไม่สามารถเพิ่มข้อมูลลงในสแตคได้อีก



รูปที่ 2.43 แสดง การ Push ข้อมูล ABC ลงในสแตคที่มีการจองเนื้อที่ไว้ N ตัว โดยมี TOP ชี้ข้อมูลตัวที่เข้ามาล่าสุด โดยค่าของ TOP จะเพิ่มขึ้นทุกครั้งเมื่อมีข้อมูลเข้ามาในสแตค

2. การดึงข้อมูลออกจากสแตค ก่อนที่จะดึงข้อมูลออกจากสแตคต้องตรวจสอบก่อนว่าสแตคมีข้อมูลอยู่หรือไม่ หรือว่าเป็นสแตคว่าง (Empty Stack)



รูปที่ 2.44 แสดงการเอาข้อมูลออกจาก สแตค

3. การใช้ สแตค เพื่อแปลรูปนิพจน์ทางคณิตศาสตร์

รูปแบบนิพจน์ทางคณิตศาสตร์

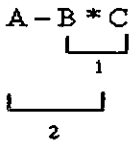
- นิพจน์ Infix คือ นิพจน์ที่เครื่องหมายดำเนินการ (Operator) อยู่ระหว่างตัวดำเนินการ (Operands) เช่น A+B-C
- นิพจน์ Prefix คือ นิพจน์ที่เครื่องหมายดำเนินการ (Operator) อยู่หน้าตัวดำเนินการ (Operands) เช่น +-AB
- นิพจน์ Postfix คือ นิพจน์ที่เครื่องหมายดำเนินการ (Operator) อยู่หลังตัวดำเนินการ (Operands) เช่น AC*+

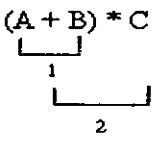
4. ลำดับการทำงานของตัวดำเนินการทางคณิตศาสตร์ (Operator Priority)

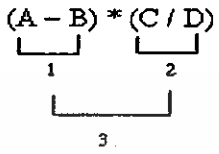
มีการลำดับความสำคัญของตัวดำเนินการจากลำดับสำคัญมากสุดไปน้อยสุด คือ ลำดับที่มีความสำคัญมากที่ต้องทำก่อน ไปจนถึงลำดับที่มีความสำคัญน้อยสุดที่ไว้ทำทีหลัง ดังนี้

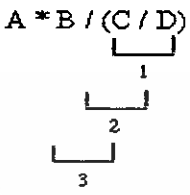
- ทำในเครื่องหมายวงเล็บ
- เครื่องหมายยกกำลัง (^)
- เครื่องหมายคูณ (*) , หาร (/)
- เครื่องหมายบวก (+) , ลบ (-)

ตัวอย่างนิพจน์ทางคณิตศาสตร์และลำดับการคำนวณ









2.2.3.3 อัลกอริทึมการแปลงนิพจน์ Infix เป็น นิพจน์ Postfix

เราสามารถแปลงนิพจน์ Infix ให้เป็น Postfix ได้โดยอาศัยสแตคที่มีคุณสมบัติการเข้าหลังออกก่อนหรือ LIFO โดยมีอัลกอริทึมในการแปลงนิพจน์ ดังนี้

1. ถ้าข้อมูลเข้า (input) เป็นตัวถูกดำเนินการให้นำออกไปเป็นผลลัพธ์ (output)
2. ถ้าข้อมูลเข้าเป็นตัวดำเนินการ (operator) ให้ดำเนินการดังนี้
   - 2.1 ถ้าสแตคว่าง ให้ push operator ลงในสแตค
   - 2.2 ถ้าสแตคไม่ว่าง ให้เปรียบเทียบ operator ที่เข้ามากับ operator ที่อยู่ในตำแหน่ง TOP ของสแตค
     - 2.2.1 ถ้า operator ที่เข้ามามีความสำคัญมากกว่า operator ที่ตำแหน่ง TOP ของสแตคให้ push ลงสแตค
     - 2.2.2 ถ้า operator ที่เข้ามามีความสำคัญน้อยกว่าหรือเท่ากับ operator ที่อยู่ในตำแหน่ง TOP ของสแตค ให้ pop สแตคออกไปเป็นผลลัพธ์ แล้วทำการเปรียบเทียบ operator ที่เข้ามากับ operator ที่ตำแหน่ง TOP ต่อไป จะหยุดจนกว่า operator ที่เข้ามาจะมีความสำคัญมากกว่า operator ที่ตำแหน่ง TOP ของสแตค แล้วจึง push operator ที่เข้ามานั้นลงสแตค
3. ถ้าข้อมูลเข้าเป็นวงเล็บเปิด ให้ push ลงสแตค
4. ถ้าข้อมูลเข้าเป็นวงเล็บปิด ให้ pop ข้อมูลออกจากสแตคไปเป็นผลลัพธ์จนกว่าจะถึงวงเล็บเปิด จากนั้นทิ้งวงเล็บเปิดและปิดทิ้งไป
5. ถ้าข้อมูลเข้าหมด ให้ pop ข้อมูลออกจากสแตคไปเป็นผลลัพธ์จนกว่าสแตคจะว่าง

นิพจน์ A + B * C

นิพจน์ Infix ข้อมูลเข้า (Input)	Stack เก็บตัวดำเนินการ	นิพจน์ Postfix ข้อมูลออก (Output)
A		A
+	+	A
B	+	AB
*	+*	AB
C	+*	ABC
		ABC*+

ตารางที่ 2.8 แสดงตัวอย่างการแปลงนิพจน์ Infix เป็นนิพจน์ Postfix

นิพจน์ (A * B) + (C / D)

นิพจน์ Infix ข้อมูลเข้า (Input)	Stack เก็บตัวดำเนินการ	นิพจน์ Postfix ข้อมูลออก (Output)
(	(	
A	(	A
*	(*	A
B	(*	AB
)		AB*
+	+	AB*
(	+(	AB*
C	+(	AB*C
/	+(/	AB*C
D	+(/	AB*CD
)	+	AB*CD/
		AB*CD/+

ตารางที่ 2.9 แสดงตัวอย่างการแปลงนิพจน์ Infix เป็นนิพจน์ Postfix

นิพจน์ A / B + (C – D)

นิพจน์ Infix ข้อมูลเข้า (Input)	Stack เก็บตัวดำเนินการ	นิพจน์ Postfix ข้อมูลออก (Output)
A		A
/	/	A
B	/	AB
+	+	AB/
(	+(	AB/
C	+(	AB/C
-	+(-	AB/C
D	+(-	AB/CD
)	+	AB/CD-
		AB/CD-+

ตารางที่ 2.10 แสดงตัวอย่างการแปลงนิพจน์ Infix เป็นนิพจน์ Postfix

นิพจน์ (A + B) * (C ^ D – E) * F

นิพจน์ Infix ข้อมูลเข้า (Input)	Stack เก็บตัวดำเนินการ	นิพจน์ Postfix ข้อมูลออก (Output)
(	(	
A	(	A
+	(+	A
B	(+	AB
)		AB+
*	*	AB+
(	*(	AB+
C	*(	AB+C
^	*(^	AB+C
D	*(^	AB+CD
-	*(-	AB+CD^
E	*(-	AB+CD^E
)	*	AB+CD^E-
*	*	AB+CD^E-*
F	*	AB+CD^E-*F
		AB+CD^E-*F*

ตารางที่ 2.11 แสดงตัวอย่างการแปลงนิพจน์ Infix เป็นนิพจน์ Postfix

2.3 Recursion

กลุ่มฟังก์ชั่นหรือเมธอดที่มีการเรียกตัวเองนั้นเราเรียกว่า recursive ซึ่งเป็นเครื่องมือที่มีความสำคัญอย่างมากในการพัฒนาโปรแกรม เพราะช่วยลดการเขียนโค้ดให้มีขนาดสั้นลงได้อีกทั้งยังลดความซับซ้อนและช่วยให้อัลกอริธึมมีประสิทธิภาพมากขึ้น พื้นฐานของ recursion เกิดขึ้นมาจากหลักทางคณิตศาสตร์ mathematical induction ซึ่งสามารถหาข้อมูลเพื่อศึกษาเพิ่มเติมได้ โดยในที่นี้จะยกตัวอย่างพื้นฐานและแนวทางการนำเอาไปประยุกต์ใช้ ตัวอย่างของการนำเอา recursive โดยทั่วไปเช่น

- ไฟล์ในคอมพิวเตอร์ถูกเก็บอยู่ในไดเร็กทอรี่ ผู้ใช้สามารถสร้างไดเร็กทอรี่ย่อยและเก็บไว้และไดเร็กทอรี่ไว้ภายในได้ และเมื่อเราต้องการแสดงข้อมูลที่เก็บอยู่ภายในคอมพิวเตอร์ก็จำเป็นที่จะต้องค้นหาโดยการค้นที่ไดเร็กทอรี่และไดเร็กทอรี่ย่อย ๆ ลงไป ซึ่งเราสามารถนำเอา recursive มาช่วยในการค้นหาได้
- การค้นหาศัพท์ในดิกชั่นนารี ตัวอย่างเช่นเมื่อเราต้องการค้นหาความหมายของศัพท์ที่ต้องการแต่ภายในคำแปลของศัพท์นั้นซึ่งบางทีเราไม่สามารถที่จะเข้าใจได้ทันที อาจจะมีศัพท์ที่ไม่เข้าใจในคำแปลนั้นทำให้ต้องค้นหาคำ ๆ นั้นอีกครั้งหนึ่งซึ่งอาจจะไม่สามารถเข้าใจได้อีก ทำให้ต้องทำซ้ำ ๆ ไปจนกว่าจะได้ความหมายที่ต้องการ ซึ่งเป็นอีกตัวอย่างหนึ่งที่สามารถนำเอา recursive เข้ามาช่วยในการแก้ปัญหานี้
- ภาษาคอมพิวเตอร์มักจะต้องใช้ recursive ตัวอย่างเช่น สมการคำนวณซึ่งมักจะประกอบไปด้วยตัวแปร วงเล็บที่ซ้อนวงเล็บเป็นต้น

Basic Recursion

พื้นฐานที่สำหรับ Recursion ที่ผู้เขียนได้แสดงดังตัวอย่างข้างล่างคือการ หาผลรวมของจำนวน n ซึ่งหากให้ n มีค่าเท่ากับ 5 จะได้ผลลัพธ์เท่ากับ 5+4+3+2+1 หรือเท่ากับ 15 ทั้งนี้ตัวอย่าง ข้างล่างอาจจะไม่ได้แสดงให้เห็นถึงประโยชน์จริง ๆ ของ Recursion เนื่องจากเราสามารถที่จะเขียนโค้ดดังกล่าวให้อยู่ในรูปของ loop แบบปกติได้ แต่โค้ดนี้เป็นโค้ดที่ไม่ซับซ้อนและแสดงให้เห็นถึงการทำงานของ Recursion ได้ง่ายกว่าตัวอย่างอื่นทั่วไป ที่มีการทำงานซับซ้อน

```
1 :     public static int sum( int n ) {
2:                  if( n == 1)
3:                          return 1;
4:                  else
5:                          return s( n – 1 ) + n;  }
```

จากโค้ดข้างบนบรรทัดที่ 2 แสดงให้เห็นว่าหาก เราใส่ค่าให้กับฟังก์ชั่น sum(1) ผลลัพธ์ที่ได้ย่อมเท่ากับ 1

จากโค้ดบรรทัดที่ 5 เนื่องจากเราสามารถเขียนรูปแบบสมการของผลรวม n ให้อยู่ในรูปของ s(n) = s(n-1) + n จึงเป็นที่มีของโค้ดบรรทัดที่ 5 จะเห็นได้ว่าเราแทบจะใช้สมการเดียวกับทางคณิตศาสตร์ที่เราคิดในการสร้างฟังก์ชั่น Recursive ได้ทันที ทำให้สามารถจินตนาการการทำงานของโค้ด Recursive ได้ง่ายยิ่งขึ้น

การทำงานของฟังก์ชั่นเริ่มจาก สมมุติเราเรียกใช้ function sum โดยให้ค่าเท่ากับ sum(4) เพื่อประมวลผลโปรแกรมจะเริ่มส่งค่า 4 เข้าไปประมวลผลยังบรรทัดที่ 2 เมื่อเปรียบเทียบแล้วจะได้ผลเป็น false เพราะ 4 ไม่เท่ากับ 1 จากนั้นจะไปทำงานยังบรรทัดที่ 5 จะได้ผลลัพธ์เป็น s(3) เป็นการเรียกตัวเองให้ทำงานอีกครั้ง ไปทำงานยังบรรทัดที่ 3 และได้ผลเป็น false จึงไปทำงานยังบรรทัดที่ 5 ได้ผลลัพธ์เป็น s(2) และเรียกตัวเองอีกครั้ง เมื่อทำงานยังบรรทัดที่ 3 ก็จะได้ผลเป็น false เช่นเดิม จากนั้นจึงทำงานที่บรรทัดที่ 5 ได้ผลลัพธ์เป็น s(1) เมื่อเรียกตัวเองและทำงานที่บรรทัดที่ 3 จะได้ผลเป็น true ทำให้ return ค่า n กลับมาเป็น 1 ทำให้ s(2) ทำงานได้ผลลัพธ์จากการเรียกตัวเองจาก s(1) บวก 2 ทำให้ได้ผลลัพธ์ s(2) เท่ากับ 3 จากนั้นจะได้ผลเป็น s(2) บวก 3 ได้ผลลัพธ์ s(3) เท่ากับ 6 เมื่อจบที่ s(4) จะได้ผลลัพธ์ของการ return ค่า n เท่ากับ 10

ตัวอย่างที่สอง การแปลงเลขฐานอีกตัวอย่างหนึ่งที่แสดงให้เป็นถึงการทำงานของ recursion ได้เป็นอย่างดี คือโปรแกรมแปลงเลขฐานตัวอย่าง เช่น เราต้องการแปลงเลขฐานสิบไปยังเลขฐานอื่น ๆ เช่นฐานสอง ฐานสิบ หรือฐานสิบหกได้ เริ่มจากเราต้องการแปลงเลข 1369 ให้อยู่ในรูป 1 digit คือแสดงค่าออกมาทีละค่า นั่นคือพิมพ์ค่าออกมาเป็น 1, 3, 6 และ 9 สามารถได้โดยการใช้เทคนิคทำ 10 มาหาร จะทำให้เราได้ค่าของผลลัพธ์ทีละตัว เมื่อถึงตัวสุดท้ายเราไม่สามารถที่จะหารเพื่อให้ได้ค่าออกมาในหลักเดียวได้แต่จะใช้เทคนิคการหารเอาเศษ หรือ mod ก็จะได้ผลลัพธ์ของเลขตัวสุดท้าย

```
1:      private static void printDec (long n)
2:          {
3:              if(n >= 10)
4:                  printDec( n / 10);
5:              System.out.println( n % 10 ) ;
6:          }
```

จากนี้เราจะเพิ่มความสามารถของฟังก์ชั่นขึ้นอีกเพื่อให้สามารถที่จะแปลงเลขฐานสิบไปยังเลขฐานอื่น ๆ ที่ต้องการได้โดยเพียงเปลี่ยนเลข ที่หารไปยังฐานที่เราต้องการจะเปลี่ยนเท่านั้น

```
1:      private static final String DIGIT_TABLE = "0123456789ABCDEF";
2:      //Precondition: n >= 0, base is valid.
3:      private static void printDecimal(long  n, int base)
4:              {
5:
6:                      if(n >= base)
7:                              printDecimal( n / base, base);
8:                      System.out.print( DIGIT_TABLE.charAt((int) (n%base)) ) ;
9:              }
```

หลักการทำงานของโค้ดข้างบนจะเหมือนกับโค้ดก่อนหน้านี้แต่จะเพิ่มความสามารถของ function ให้สามารถที่จะเลือกเลขฐานที่ต้องการแสดงออกมาได้ โดยจะทำการเก็บเลขฐานไว้ใน string ตั้งแต่ 0 ถึง 16 เมื่อเราต้องการใช้เพียงอ้างถึงฐานที่ต้องการ (จำนวนเต็ม) จากนั้นใช้ method ของ class String ที่ชื่อ charAt(int x) ช่วยในการอ้างถึงตำแหน่งของอักษรของตำแหน่ง string นั้น ๆ เมื่อแสดงผลออกมาทางหน้าจอ

แต่ปัญหาหนึ่งของ function นี้คือหากใส่ค่าที่ไม่ถูกต้องให้กับ function จะทำให้เกิด Error ขึ้นเช่นหากให้ค่า n ที่น้อยกว่า 0 หรือให้ค่าเลขฐานที่มากกว่า 16 จะทำเกิดข้อผิดพลาดคือ อ้างอิงถึง array ที่ไม่ได้กำหนดไว้หรือ array index out of bound นั่นเอง ฉะนั้นเราจึงควรมีการตรวจสอบก่อนที่จะมีการให้ค่าแก่ฟังก์ชั่นเพื่อกันข้อผิดพลาด โดยการสร้าง wrapper ขึ้นมาเพื่อห่อหุ้มฟังก์ชั่นของเราอีกทีหนึ่งเรียกว่า Driver

ตัวอย่างการสร้าง wrapper

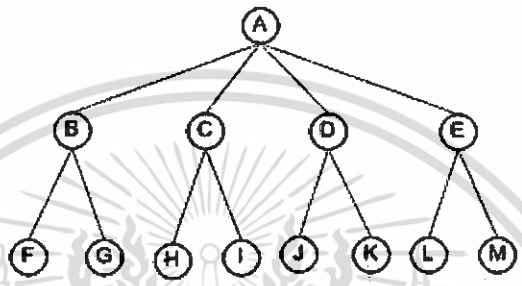
```
private static final String DIGIT_TABLE = "0123456789ABCDEF";
private static int MAX_BASE = DIGIT_TABLE.length();
private static void printDecimal(long  n, int base)
        {
          if(n >= base)
                printDecimal( n / base, base);
```

```
        System.out.print( DIGIT_TABLE.charAt((int) (n%base)) ) ;
        }
    //Driver function of printDecimal
    public static void printInt(long n, int base)
        {
        if(base <= 1 || base > MAX_BASE)
        System.err.println("Cannot print in base "+base);
          else
        {
        if( n < 0 )
        {
        n = -n;
                                System.out.println("-");
        }
        printDecimal(n, base)
    }
```

จากตัวอย่างข้างบนเราได้ทำการสร้างฟังก์ชั่น printInt ขึ้นมาเพื่อเป็น driver คอยทำหน้าที่ตรวจสอบพารามิเตอร์ที่ป้อนเข้ามาเพื่อป้องกันปัญหาที่จะเกิดขึ้นหากใส่ค่าที่ไม่ถูกต้องก่อนจะเรียกไปยังฟังก์ชั่น printDecimal โดยค่า 2 <= base <= MAX_BASE เพื่อแก้ปัญหาการอ้างอิง array เกิดขนาด

2.4 Trees

ทรี หรือโครงสร้างข้อมูลแบบต้นไม้ ประกอบด้วยโหนด (node) ซึ่งเป็นส่วนที่เก็บข้อมูล ในทรีหนึ่งทรีจะประกอบไปด้วยรูทโหนด (root node) เพียงหนึ่งโหนด แล้วรูทโหนดสามารถแตกโหนดออกเป็นโหนดย่อยๆ ได้อีกหลายโหนดเรียกว่าโหนดลูก (Child node) เมื่อมีโหนดลูกแล้ว โหนดลูกก็ยังสามารถแสดงเป็นโหนดพ่อแม่ (Parent Node) โดยการแตกโหนดออกเป็นโหนดย่อยๆได้อีก



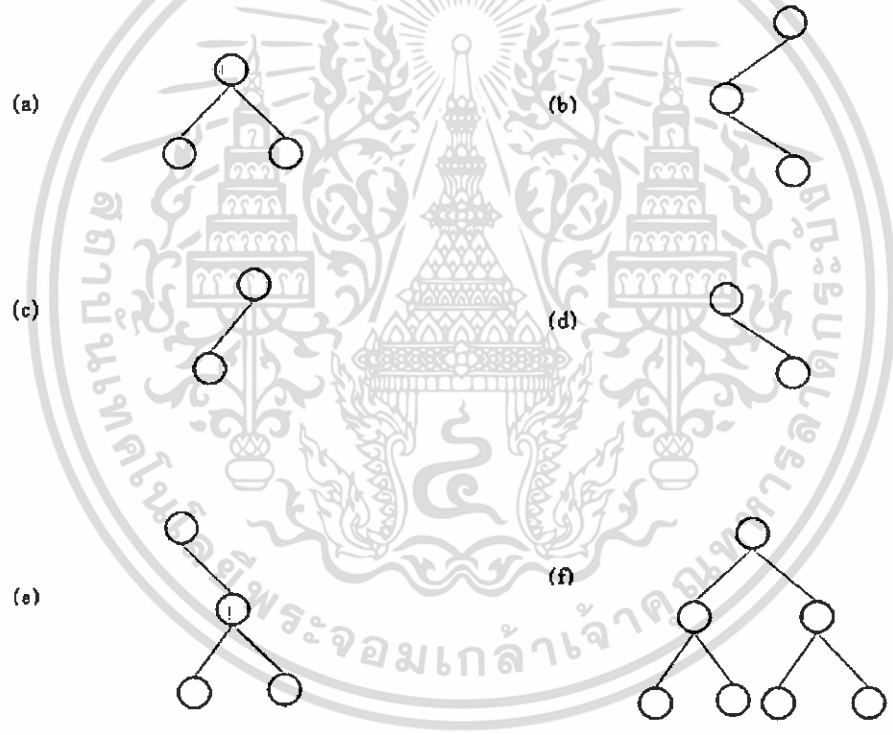
รูปที่ 2.45 แสดงโครงสร้างข้อมูลแบบต้นไม้

- Root Node จากรูป คือ โหนด A
- Child Node หรือ โหนดลูก จากรูป B , C , D และ E เป็น โหนดลูกของ A
- Parent Node หรือโหนดพ่อแม่ โหนด B ที่เป็นโหนดลูกของโหนด A ก็สามารถแตกออกเป็นโหนดย่อยๆ ได้แก่ F และ G ดังนั้น B จึงเป็นโหนดพ่อแม่ของ F และ G ในทำนองเดียวกัน A ก็เป็นโหนด พ่อแม่ของ B , C , D และ E กิ่ง (Branch or Edge) เป็นเส้นที่เชื่อมต่อระหว่างโหนดพ่อแม่กับโหนดลูก
- Brother Node หรือโหนดพี่น้อง คือ โหนดที่มีพ่อแม่เดียวกัน เช่น B , C , D , E เป็นโหนดพี่น้องกันเพราะมีโหนดพ่อแม่เดียวกัน คือ โหนด A และ F และ G เป็นโหนดพี่น้องกันโดยมี B เป็นโหนดพ่อแม่
- Leaf Node โหนดที่ไม่มีโหนดลูกจากรูปโหนดที่ไม่มีโหนดลูกได้แก่ F G H I J K L M
- Branch Node คือ โหนดที่ไม่ใช่ Leaf Node เช่น โหนด B C D E เรียกว่า Branch Node
- Degree คือ จำนวนลูกของโหนด x เช่น degree ของ โหนด A = 4 ได้แก่ B C D E จำนวน degree ของโหนด B = 2 จำนวนdegree ของโหนด F = 0 เนื่องจากโหนด F ไม่มีโหนดลูก
- Direct Descendant Node คือโหนดที่มาทีหลังทันที จากรูป B C D E เป็น direct descendant node ของโหนด A เพราะเป็นโหนดที่มาทีหลังทันที

- Descendant Node คือ โหนดลูกของโหนด x และโหนดที่ทุกโหนดที่แตกจากโหนดลูกของโหนด x ตัวอย่าง descendant ของโหนด A คือ ทุกโหนดที่เหลือในทรี
- Direct Ancestor Node หรือโหนดที่มาก่อนทันที ตัวอย่าง Direct Ancestor ของโหนด H คือ โหนด C , Direct Ancestor ของโหนด C คือ โหนด A
- Level หรือระดับ คือ หมายเลขแสดงระดับของโหนดในทรี ซึ่งรูทโหนดจะมีค่า Level = 0 ส่วนโหนดลูกของรูทโหนดก็จะมีค่า = 1 หากค่าโหนด x อยู่ในระดับ L โหนดลูกของ x ก็จะอยู่ในระดับ L + 1

2.4.1 Binary Tree

มีลักษณะเหมือนกับ Tree ปกติแต่มีคุณสมบัติพิเศษ คือ "แต่ละโหนดจะมีโหนดลูกได้ไม่เกิน 2 โหนด" หรือพูดอีกนัยหนึ่งก็คือ แต่ละโหนดใน binary tree จะมีดีกรีได้ไม่เกิน 2



รูปที่ 2.46 ตัวอย่าง binary tree

2.4.1 Complete Binary Tree

ต้นไม้ไบนารีแบบสมบูรณ์ มีลักษณะคล้ายกับ Binary Tree แต่มีข้อพิเศษ คือ

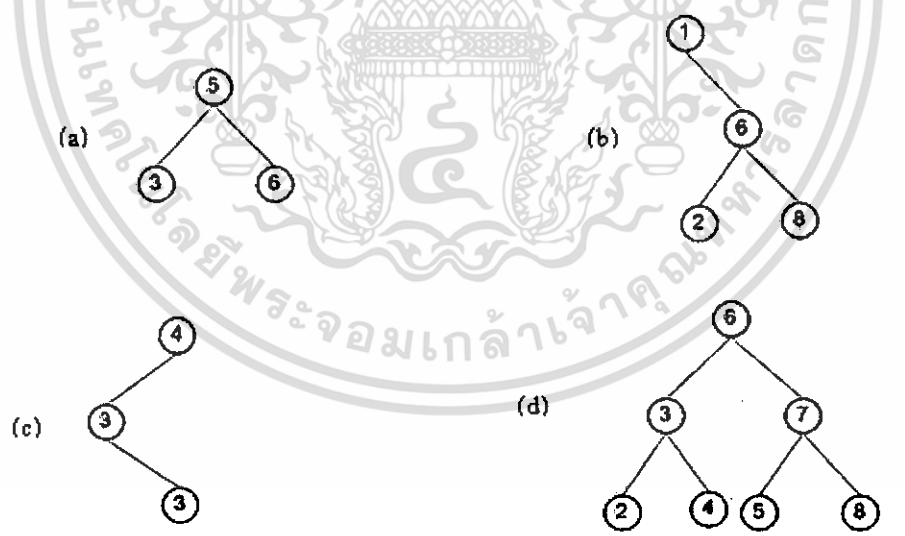
- ทุกโหนดที่ไม่ใช่ Leaf Node จะต้องมีโหนดลูก 2 โหนด
- Leaf Node จะต้องอยู่ในระดับเดียวกัน

รูปที่ 2.47 แสดง Complete Binary Tree

2.4.2 Binary Search Tree

มีลักษณะคล้ายกับ Binary Tree แต่มีลักษณะพิเศษเพิ่มเติม คือ

- ค่าของรูทโหนดมีค่ามากกว่าค่าในต้นไม้ย่อยซ้าย (TL < R)
- ค่าของรูทโหนดมีค่าน้อยกว่าหรือเท่ากับค่าในต้นไม้ย่อยขวา (R <= TR)



รูปที่ 2.48 ตัวอย่าง Binary Search Tree

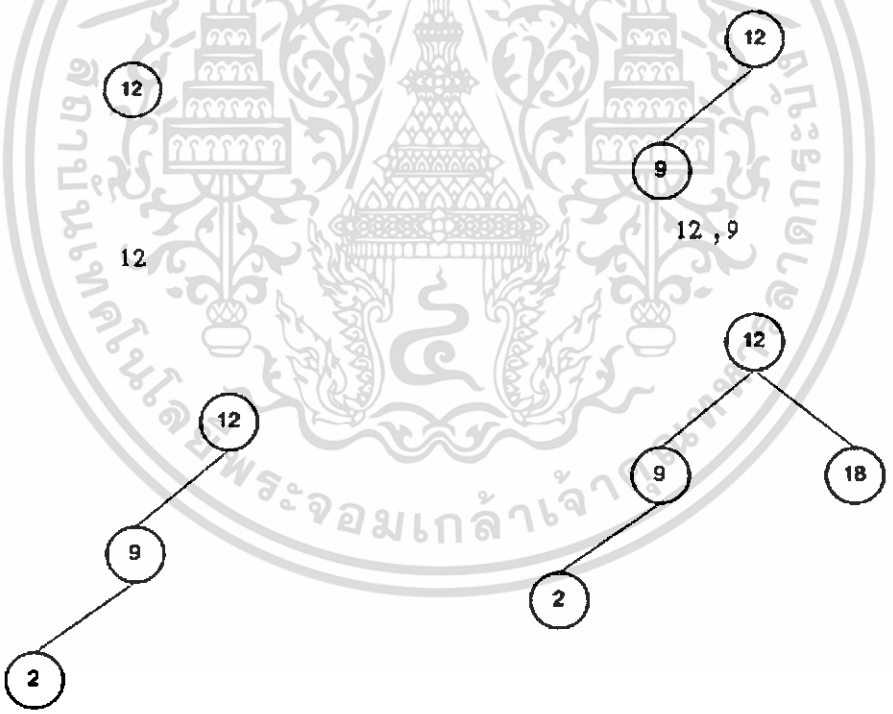
2.4.3.1 การสร้างและเพิ่มข้อมูลใน Binary Search Tree

วิธีการสร้างและเพิ่มข้อมูลเริ่มจาก

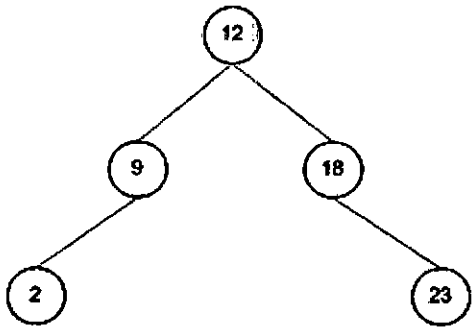
1. ถ้า Binary Search Tree ยังไม่มีข้อมูล ให้โหนดที่เข้ามาใหม่เป็นรูทโหนดของ Binary Search Tree
2. ถ้า Binary Search Tree มีข้อมูลอยู่ ให้เพิ่มโหนดที่เข้ามาดังนี้
   2.1 ถ้าค่าของโหนดใหม่ที่เข้ามา มากกว่า ค่าของรูทโหนด ให้เพิ่มโหนดใหม่ลงในต้นไม้ย่อยด้านขวา
   2.2 ถ้าค่าของโหนดใหม่ที่เข้ามา น้อยกว่า ค่าของรูทโหนด ให้เพิ่มโหนดใหม่ลงในต้นไม้ย่อยด้านซ้าย

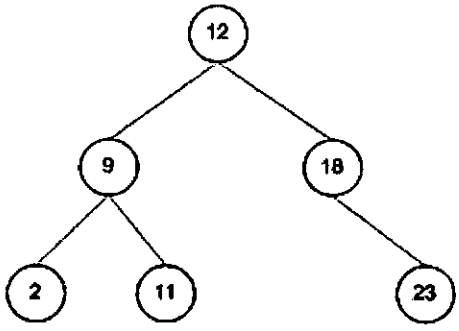
ตัวอย่าง การสร้างและเพิ่มข้อมูล Binary Search Tree

มีค่าข้อมูลดังนี้ 12 , 9 , 2 , 18 , 23 , 11 , 14



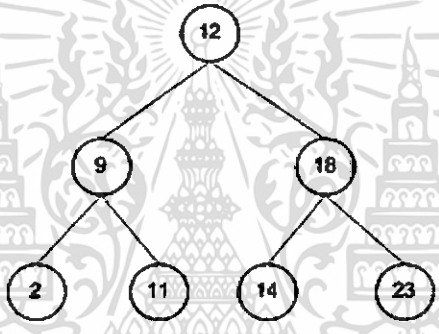
รูปที่ 2.49(ก) รูปการ สร้างและเพิ่มข้อมูล Binary Search Tree





12, 9, 2 , 18 , 23

12 , 9 , 2 , 18 , 23 , 11



12 , 9 , 2 , 18 , 23 , 11 , 14

รูปที่ 2.49(ช) รูปการ สร้างและเพิ่มข้อมูล Binary Search Tree

2.4.3.2 การลบข้อมูลใน Binary Search Tree

กรณีที่1 หากโหนดที่ต้องการลบเป็น Leaf Node สามารถลบได้ทันที

กรณีที่2 ถ้าโหนดที่ต้องการลบมีต้นไม้ย่อยเพียงด้านเดียว เมื่อลบโหนดที่ไม่ต้องการทิ้งแล้ว ให้นำค่ารูทโหนดของต้นไม้ย่อย ไปแทนที่โหนดที่ลบทิ้งไป

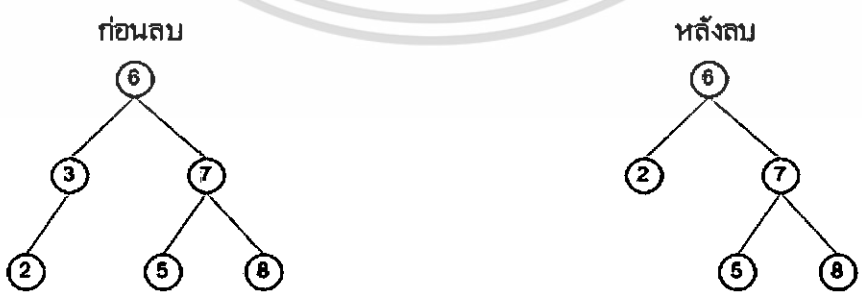
กรณีที่ 3 ถ้าโหนดที่ต้องการลบมีต้นไม้ย่อยทั้งสองด้าน เมื่อลบโหนดที่ไม่ต้องการแล้ว มีวิธีให้เลือกทำอยู่ 3 วิธี คือ

1. นำค่าที่น้อยที่สุดของต้นไม้ย่อยขวาไปแทนที่โหนดที่ลบทิ้งไป
2. นำค่าที่มากที่สุดของต้นไม้ย่อยซ้ายไปแทนที่โหนดที่ลบทิ้งไป
3. นำค่ารูทโหนดของต้นไม้ย่อยขวาไปแทนที่โหนดที่ลบทิ้งไปแล้ว นำค่ารูทโหนดของต้นไม้ย่อยซ้ายไปเป็นโหนดลูกทางซ้ายของโหนดที่มีค่าน้อยที่สุดของต้นไม้ย่อยขวา

กรณีที่ 1 ต้องการลบโหนด 8

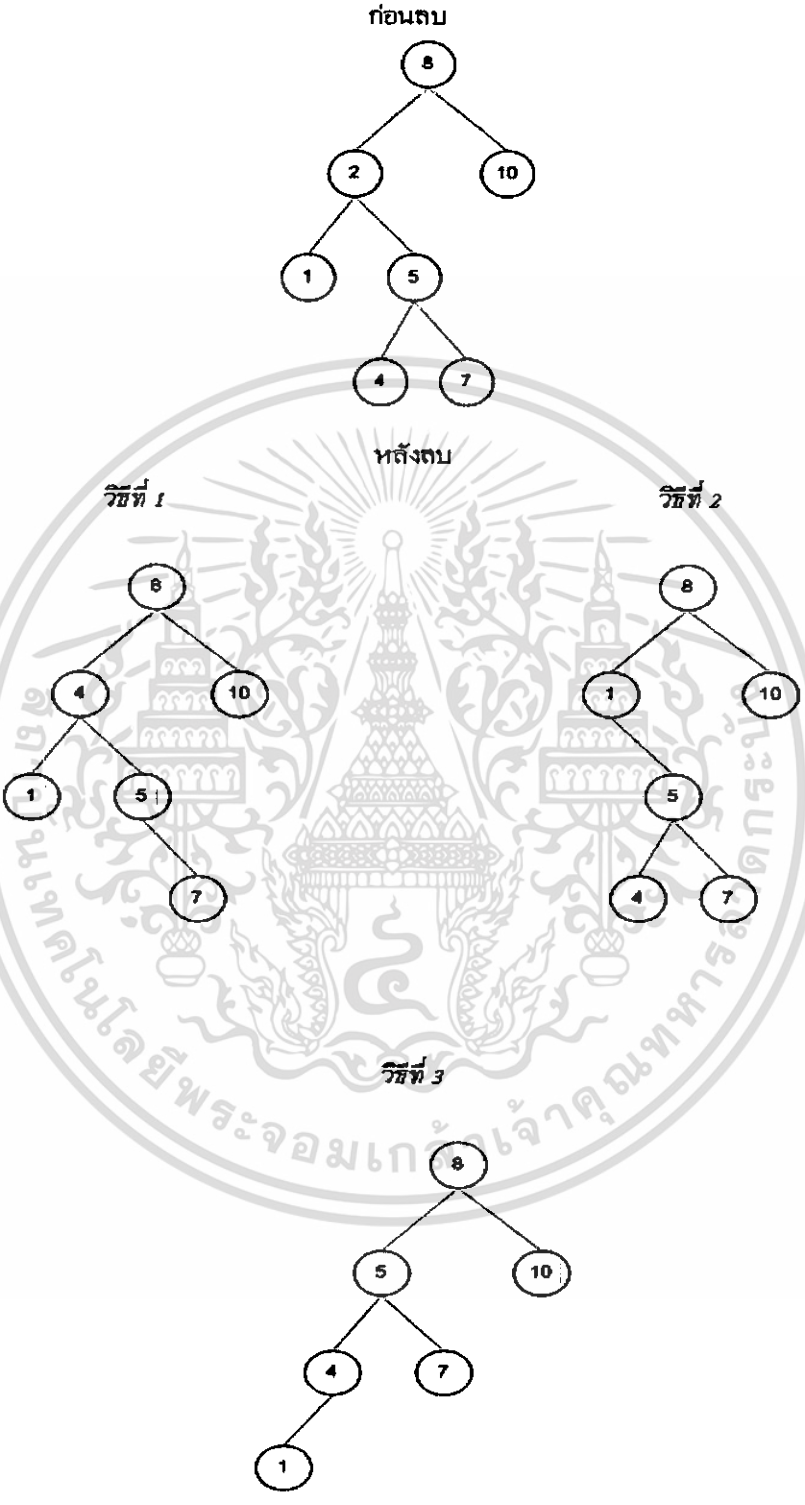
รูปที่ 2.50(ก) การลบโหนด

กรณีที่ 2 ต้องการลบโหนด 3



รูปที่ 2.50(ข) การลบโหนด

กรณีที่3 ต้องการลบโหนด 2



รูปที่ 2.50(ค) การลบโหนด

2.4.3.3 การท่องไปในทรี (Tree Traversal)

สามารถท่องเข้าไปในทรีเพื่อดูข้อมูล ได้ 3 วิธีด้วยกันคือ

1. Preorder
2. Inorder
3. Postorder

ในการท่องเข้าไปในทรีแต่ละแบบจะใช้สัญลักษณ์ดังนี้

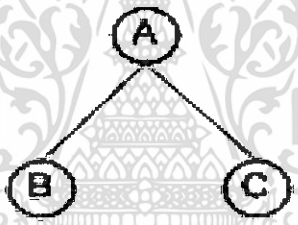
Root = root node

Left = ต้นไม้ย่อยซ้ายของ Root

Right = ต้นไม้ย่อยขวาของ Root

วิธีในการท่องเข้าไปในทรีแต่ละแบบจะมีลักษณะดังนี้

1. Preorder = Root Left Right
2. Inorder = Left Root Right
3. Postorder = Left Right Root



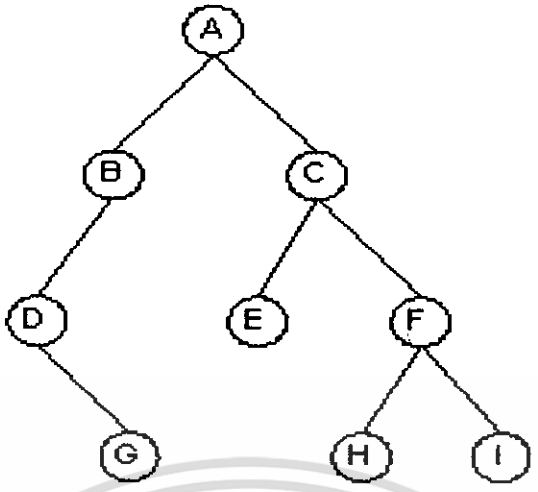
รูปที่ 2.51(ก) ตัวอย่างแสดงการท่องในทรี

เมื่อท่องไปในทรีแบบต่าง ๆ กันจะได้ดังนี้

Preorder จะได้ข้อมูล ABC

Inorder จะได้ข้อมูล BAC

Postorder จะได้ข้อมูล BCA



รูปที่ 2.51(ข) ตัวอย่างแสดงการท่องในทรี

เมื่อท่องไปในทรีแบบ

Preorder จะได้ข้อมูล ABDGCEFHI

Inorder จะได้ข้อมูล DGBAECHFI

Postorder จะได้ข้อมูล GDBEHIFCA

2.6 ระบบฐานข้อมูลโดยใช้ Access 2003

Access เป็นระบบจัดการฐานข้อมูล ตัวหนึ่ง ที่เป็นที่นิยมกันมาก เป็นเพราะ Access โปรแกรมจัดการทางด้านฐานข้อมูลที่เป็นที่นิยมใช้ อีกทั้งยังรองรับการสนับสนุนการใช้งานบนระบบปฏิบัติการมากมาย ไม่ว่าจะเป็น UNIX, OS/2, MAC OS, Windows และอื่นๆ อีกมากมาย และยังสามารถทำงานร่วมกับ Web Development Platform ทั้งหลาย ไม่ว่าจะเป็น C, C++, Java, Perl, PHP, Python, Tel, ASP หรือ ภาษาในสถาปัตยกรรม dotNet ก็ตาม อีกทั้ง Access ยังมีการพัฒนาที่ต่อเนื่อง

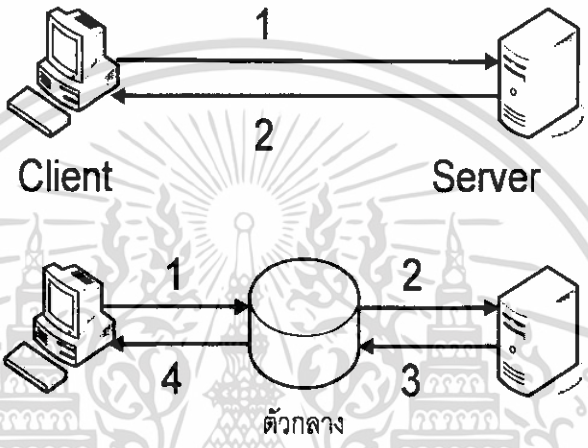
2.6.1 หลักการทำงานในลักษณะ Client to Server

- ที่ฝั่งของ เซริฟเวอร์ (Server) จะมีโปรแกรมหรือระบบสำหรับจัดการฐานข้อมูลทำงานรออยู่ เพื่อเตรียมหรือรอคอยร้องขอการใช้บริการจาก ไคลเอนท์ (Client)
- เมื่อมีการร้องขอการใช้บริการเข้ามา Server จะทำการตรวจสอบตามวิธีการของตนเช่น อาจจะมีการให้ผู้ใช้บริการระบุชื่อและรหัสผ่าน และสำหรับ Access สามารถกำหนดได้ว่าจะอนุญาตหรือปฏิเสธ Client ใดๆ ในระบบที่จะเข้าใช้บริการอีกด้วย
- ถ้าผ่านการตรวจสอบ Server ก็จะอนุมัติ การให้บริการแก่ Client ที่ร้องขอการใช้บริการนั้นๆ ต่อไป และถ้าในกรณีที่ไม่ได้รับการอนุมัติ Server ก็จะส่งข่าวสารความผิดพลาดแจ้งกลับไปที่ Client ที่ร้องขอการใช้บริการนั้น

เครื่องคอมพิวเตอร์ที่ทำหน้าที่เป็น Client หรือ Server อาจจะอยู่บนเครื่องเดียวกันหรือแยกเครื่องกันก็ได้ ทั้งนี้ขึ้นอยู่กับลักษณะการทำงาน หรือการกำหนดของผู้บริหารระบบ ตามปกติถ้าเป็นการทำงานในลักษณะ Web-based , มีการใช้ฐานข้อมูลขนาดไม่ใหญ่นัก ตัว Access Server และ Client มักจะอยู่บนเครื่องเดียวกัน โดยเครื่องคอมพิวเตอร์ดังกล่าวจะต้องมีทรัพยากรเพื่อการทำงานมากพอสมควรแต่สำหรับการทำงานจริง (Real-world Application) ก็มักจะแยก Client และ Server ออกเป็นคนละเครื่องกัน เพราะสามารถรองรับงานได้ดีกว่า มากกว่าดังนั้น ผู้บริหารระบบ หรือผู้กำหนดนโยบายสำหรับการทำงานเครือข่ายจะต้องคำนึงถึงเรื่องที่เกี่ยวข้องเหล่านี้ให้ดี เพื่อที่จะทำให้ระบบมีการทำงานรองรับการให้บริการแก่ผู้ใช้อย่างมีประสิทธิภาพและข้อมูลนี้ความปลอดภัยมากที่สุด

2.6.2 การเชื่อมต่อจาก Client to Server สำหรับ Access

แบบผ่านตัวกลาง แบบที่เป็นที่นิยมใช้งานกันมากที่สุดก็คือ ODBC (Open Database Connectivity) ซึ่งส่วนใหญ่จะใช้กับ Server ที่ใช้ Windows Platform เป็นระบบปฏิบัติการ การทำงานประเภทนี้อาจจะมีการทำงานที่ช้ากว่าแบบ Native เพราะการทำงานในแต่ละครั้งระหว่าง Client และ Server ต้องผ่านตัวกลางก่อน แต่ ODBC ก็ถือว่ามีข้อได้เปรียบในเรื่องฐานผู้ใช้ Windows Platform มากกว่า และด้วย ODBC ทำให้เราสามารถใช้ Client Development Tools ยอดนิยมเช่น Access, VB, ASP เพื่อเชื่อมต่อเข้าหาฐานข้อมูล Access

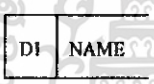
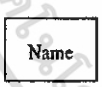


รูปที่ 2.52 การทำงานแบบ Native และแบบผ่านตัวกลาง

2.7 ดาต้าโฟล์วไดอะแกรม (Data Flow Diagram) หรือ DFD

2.7.1 สัญลักษณ์ที่ใช้ในแผนภาพกระแสข้อมูล

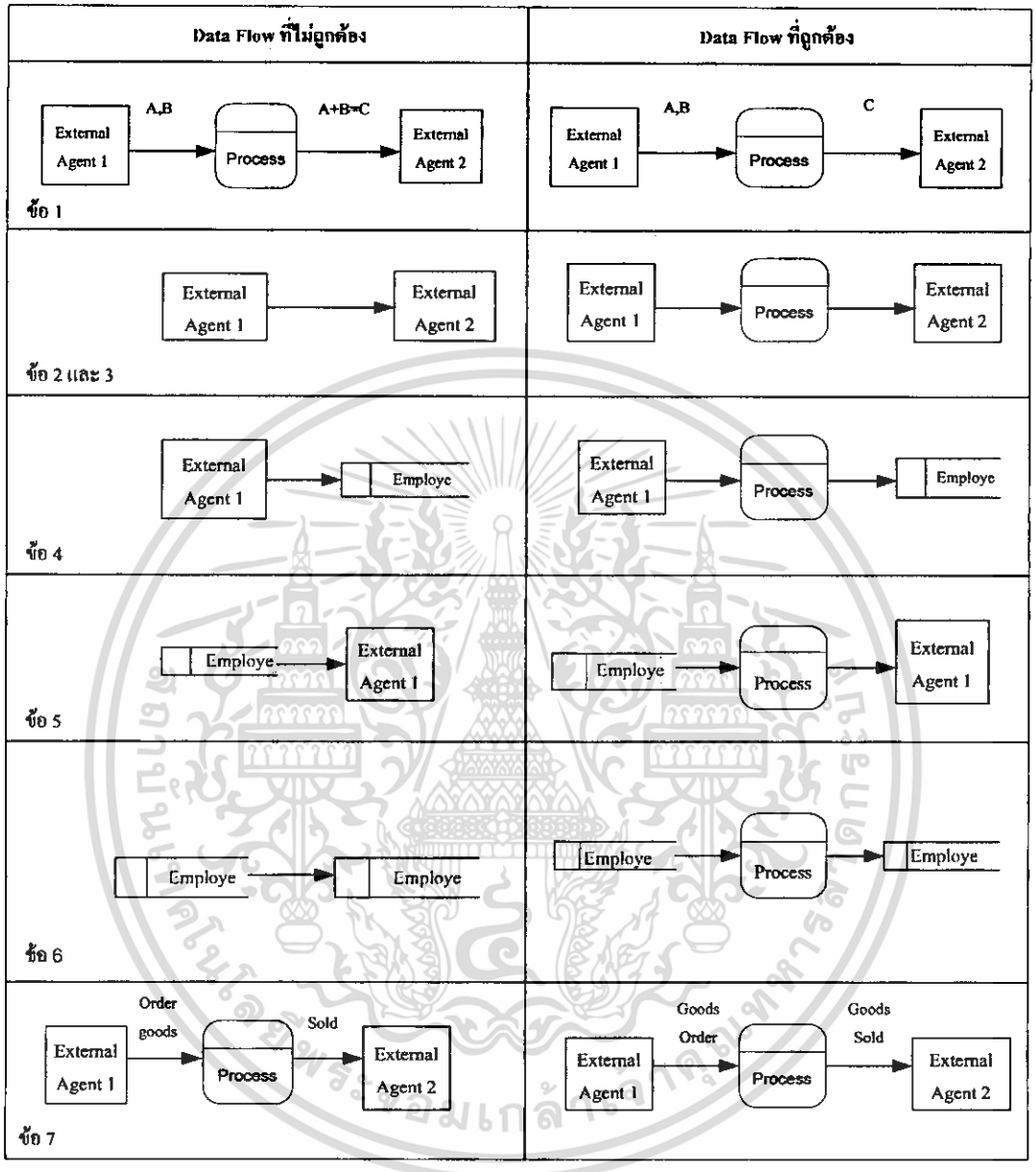
ดาต้าโฟล์วไดอะแกรม หรือเรียกสั้นๆว่า DFD เป็นแบบจำลองที่แสดงถึงขั้นตอนการดำเนินงานทางธุรกิจ และ การเคลื่อนย้ายข้อมูลภายในระบบ โดยมีสัญลักษณ์ที่ใช้ ดังนี้

สัญลักษณ์ที่ใช้	องค์ประกอบสำคัญ	คำอธิบายเพิ่มเติม
	-หมายเลขของกระบวนการ (Process) -ชื่อกระบวนการ (ควรเป็นกริยา)	เป็นสัญลักษณ์แทนกระบวนการทำงานของระบบซึ่งมีการไหลเข้า-ออกของข้อมูลได้หนึ่งทางหรือมากกว่า
	-ชื่อ (ควรเป็นคำนาม) (Data Flow)	เป็นสัญลักษณ์แทนการไหลของข้อมูลตามทิศทางลูกศร
	-หมายเลขของที่เก็บฐานข้อมูล -ชื่อ(ควรเป็นคำนาม) (Data store)	เป็นสัญลักษณ์แทนที่เก็บข้อมูลของระบบซึ่งอาจมีอยู่มากกว่า 1 ตัวก็ได้
	-ชื่อขององค์ประกอบภายนอก (ควรเป็นคำนาม) (External Agent)	เป็นสัญลักษณ์แทนองค์ประกอบภายนอกที่มีส่วนเกี่ยวข้องกับระบบ

ตารางที่ 2.13 แสดงรูปแบบสัญลักษณ์ต่างๆ ที่ใช้ในการเขียน ดาต้าโฟล์วไดอะแกรม

ดาต้าโฟล์วไดอะแกรมจะมีความสำคัญในการออกแบบระบบงานเนื่องจากเป็นแผนภาพที่จะแสดงขั้นตอนต่างๆ ที่กระทำในระบบ รวมทั้งการไหลของข้อมูลต่างๆ ภายในระบบอีกด้วยทำให้สามารถมองเห็นการทำงานทั้งหมดภายในระบบ ส่งผลให้ง่ายต่อการออกแบบในส่วนอื่น

2.7.2 ข้อกำหนดของ Data Flow



ตารางที่ 2.14 แสดงข้อกำหนดของ Data Flow

สัญลักษณ์ต่างๆ ข้างต้น จะถูกนำมาเขียนรวมกันเป็น Data Flow Diagram ของระบบโดยจะแบ่งออกเป็นระดับ

- **เลเวล 1(Level) คือ** คอนแทกซ์ไดอะแกรม (Context Diagram) เป็นแผนภาพที่แสดงการทำงานทุกกระบวนการ (Process) ภายในระบบด้วยกระบวนการเพียงกระบวนการเดียว โดยจะแสดงองค์ประกอบภายนอกที่เกี่ยวข้องกับระบบทั้งหมดว่ามีความเกี่ยวข้องกับระบบอย่างไร
- **เลเวล 0 (Level 0)** ไดอะแกรม เป็นแผนภาพที่แสดงกระบวนการทั้งหมดที่รวมอยู่ในระบบนั้นๆ เป็นการแสดงการย้ายข้อมูลทั้งหมด ว่าเป็นอย่างไรในแต่ละกระบวนการ ซึ่งในส่วนนี้นอกจากจะมีองค์ประกอบภายนอกแล้วยังมีส่วนของที่เก็บข้อมูลเพิ่มเข้ามาด้วย (Data store)เลเวล 1 ไดอะแกรม เป็นแผนภาพที่แสดงกระบวนการทั้งหมดใน ระดับ 0 ซึ่งจะแสดงรายละเอียดย่อยๆ ของขั้นตอนในระดับ 0 ที่ไม่อาจแสดงออกมาได้หมด และแสดงการเคลื่อนย้ายของข้อมูลในในแต่ละขั้นตอนว่าเป็นอย่างไร รวมทั้งรายละเอียดที่มากกว่าในระดับ 0 ซึ่งไม่จำเป็นจะต้องแสดงทุกขั้นตอนในระดับ 0 แต่เป็นการแสดงในขั้นตอนที่มีรายละเอียดย่อยๆลงไปเท่านั้น

2.8 Visual Basic

Visual Basic เป็นโปรแกรมที่ใช้สร้างโปรแกรมประยุกต์ สำหรับระบบปฏิบัติการ Windows visual เป็นส่วนที่หมายถึงเมธอดในการติดต่อแบบ graphical user interface (GUI) ซึ่งการสร้างทาได้โดยการเพิ่มอ็อบเจค ลงบนฟอร์มที่ทำหน้าที่ติดต่อกับผู้ใช้ผ่านจอภาพ

Visual Basic.Net เครื่องมือที่ใช้พัฒนาโปรแกรมแบบ Visual Programming บนระบบปฏิบัติการ Windows ซึ่งได้รับการพัฒนามาจากภาษา BASIC (Beginners All Purpose Symbolic Instruction Code) ซึ่งเป็นภาษาโปรแกรมที่ได้รับความนิยมอย่างแพร่หลาย เนื่องจากภาษา BASIC เป็นภาษาที่เข้าใจได้ง่าย

VB.NET ต่างจาก VB คือการปรับเปลี่ยนภาษาเป็นลักษณะ OOP (Object-Oriented Programming) เต็มตัวเหมือนกับภาษาโปรแกรมสมัยใหม่ เช่น C++, C#, Delphi และ Java เป็นต้น และด้วยความที่ VB.NET อยู่ในตระกูล .NET จึงซึมซับเอาความสามารถอื่นๆใน .NET เข้ามาด้วยกันนอกจากนี้แล้ว VB ยังเป็นภาษาที่ถูกผนวกเข้ากับโปรแกรมอื่นๆของ Microsoft เพื่อนใช้เขียนโปรแกรมลักษณะสคลิปต์ (Script) หรือมาโคร (Macro)

2.8.1 แนวคิดของ Visual Basic

โปรแกรมประยุกต์ Visual Basic เป็นการพัฒนาในสภาพแวดล้อมของ windows ซึ่งแนวคิดพื้นฐานในการทางานของระบบ Windows ที่สำคัญมี 3 ประการ คือ window , events และ ข่าวสาร (message)โปรแกรมประยุกต์ Visual Basic มีการทำงานแบบ Event-Driven ที่เป็นการประมวลผลตามคำสั่งในแต่ละส่วนเพื่อตอบสนองต่อ event ซึ่ง event เหล่านี้สามารถเปลี่ยนโดยการทำงานของผู้ใช้ข่าวสารของระบบหรือโปรแกรมประยุกต์อื่นๆหรือภายในโปรแกรมเดียวกัน ลำดับการทำงานของ eventจะจัดลำดับโดยจากการประมวลคำสั่ง Intrinsic control เป็นตัว control มาตรฐานของ Visual Basic เป็นตัวที่มองเห็นได้ใน Toolbox เมื่ออยู่ใน IDE Window ตัว control Intrinsic control มีข้อได้เปรียบบางประการ คือ

- การสนับสนุน intrinsic control รวมอยู่ใน VBVM60.DLL ไฟล์ Run-time จะกระจายไปยังโปรแกรมประยุกต์ Visual Basic ทุกโปรแกรม มีความหมายว่าโปรแกรมที่ใช้ intrinsic control ไม่จำเป็นต้องเพิ่มไฟล์ OCX ในการ ติดตั้งเพิ่มเติม
- โดยทั่วไปการสร้างและแสดง intrinsic control ทำได้เร็วกว่าตัว control ภายนอก Time เป็นตัว control พิเศษที่ไม่เห็นเมื่อเวลาเรียกใช้ วัตถุประสงค์การใช้คือการสร้าง event ในฟอร์มแม่โดยการเขียนคำสั่งใน procedure ที่เจาะจงสำหรับการทำงานเบื้องหลัง เช่น การตรวจสถานะของอุปกรณ์ต่อพ่วง

2.8.2 การติดต่อกับฐานข้อมูล

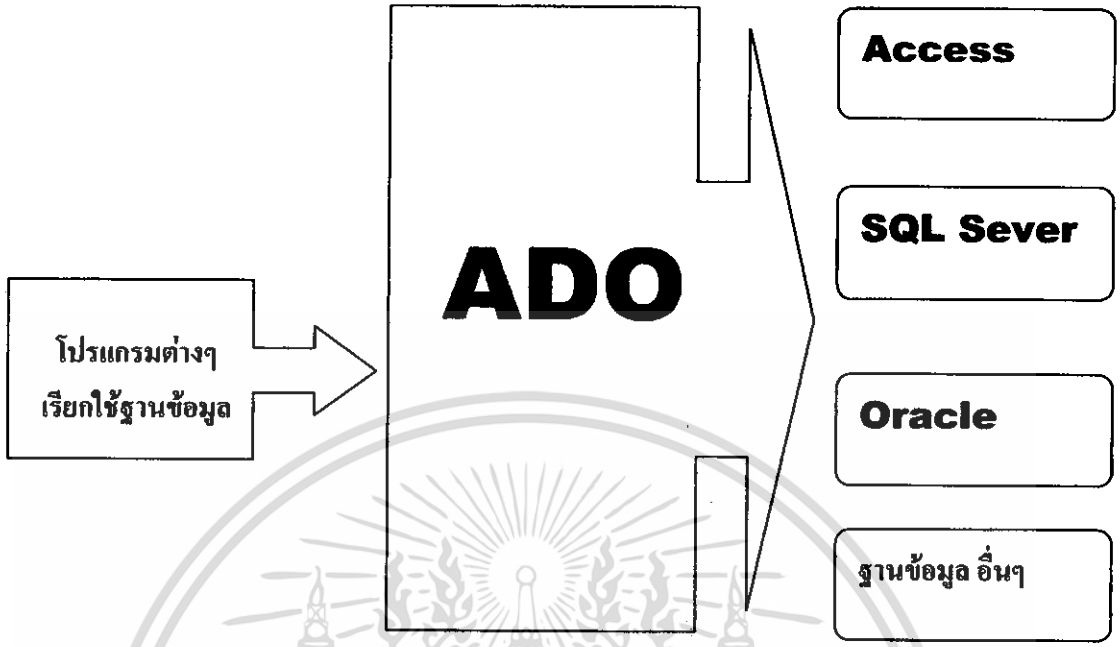
การพัฒนาโปรแกรมประยุกต์ Visual Basic ส่วนใหญ่นำมาใช้กับฐานข้อมูล และการประยุกต์แบบ Client/server ซึ่ง Visual Basic เป็นเครื่องมือที่อำนวยความสะดวกให้กับผู้พัฒนาโปรแกรมการเข้าถึงข้อมูลความสามารถใหม่ที่สัมพันธ์กับฐานข้อมูลของ Visual Basic มีพื้นฐานจาก ActiveX Data Object (ADO) ซึ่งเป็นเทคโนโลยีที่ให้ผู้ใช้เข้าถึงฐานข้อมูลหรือแหล่งข้อมูล เมื่อมีการใช้ OLE DB provider ติดต่อกับแหล่งข้อมูลVisual basic.net มีวิธีการติดต่อกับฐานข้อมูลได้หลายวิธี

- **ODBC** ย่อมาจาก Open Data Connectivity และตั้งค่าการทำงานให้ผู้ใช้ติดต่อกับฐานข้อมูลทั้งภายในพื้นที่และระยะไกล Microsoft เสนอเทคโนโลยีที่เป็นวิธีการเข้าถึงฐานข้อมูลหลายประเภท เช่น dBase, Microsoft FoxPro, Microsoft Access, Microsoft SOL Server, Oracle รวมไฟล์ text แบบ comma-delimited ด้วยการใช้ API ร่วมกัน ส่วนที่โปรแกรมประยุกต์ทำงานกับ DLLเรียกว่า ODBC diver manager ซึ่งจะส่งคำสั่งไปที่ไดร์ฟเวอร์ ODBC ในการระบุฐานข้อมูลที่ผู้ใช้ต้องการ สิ่งท้าทายของ ODBC เป็นการให้ติดต่อร่วมกับฐานข้อมูลประเภทต่างๆ ในทางทฤษฎีผู้เขียนโปรแกรมประยุกต์สามารถเตรียมใช้ ODBC ติดต่อกับฐานข้อมูล Access และเปลี่ยนขนาดไปที่ฐานข้อมูล Sol Server โดยการเปลี่ยนไดร์ฟเวอร์ back-end ของ ODBC และมีคำสั่งไม่มาก การทาสิ่งเหล่านี้ได้เนื่องจากคำสั่งที่ส่งไปยังฐานข้อมูล คือ คำสั่งที่ส่งไปยังฐานข้อมูล คือ คำสั่งมาตรฐาน SQL ภาษา SOL (Structure Query Language) เป็นภาษาโปรแกรมที่ใช้กับฐานข้อมูล ในการปฏิบัติ เลเยอร์ ODBC

สามารถแปลงคำสั่ง SQL ให้เป็นภาษาเฉพาะของฐานข้อมูล ผู้เขียนโปรแกรม ODBC มักจะข้าม engine การแปลของ ODBC และส่งคำสั่งโดยตรงกับฐานข้อมูล ODBC มีประสิทธิภาพเมื่อเปรียบเทียบกับเทคนิคการเข้าถึงข้อมูลความได้เปรียบของ ODBC คือสนับสนุน API ทั้งประเภท16 บิต 32บิต ใน ODBC เวอร์ชั่น 3 เพิ่มเทคนิคการบัฟให้ดีขึ้น เช่น การติดต่อแบบ pool ทำให้โปรแกรมประยุกต์ตอบสนองได้ดีขึ้น Microsoft Transaction Server ใช้การติดต่อแบบ pool เพื่อความเร็วในการเปิดการติดต่อโดย component ของ ActiveX ที่ทำงานอยู่ภายใต้ ODBC

- **DAO** หรือ Data Access Object เป็นสิ่งสำคัญของโปรแกรมประยุกต์ Visual Basic 3 ในพัฒนาการประยุกต์กับฐานข้อมูล DAO เป็นการติดต่อแบบ Object-oriented ไปยัง Microsoft Jet ที่เป็น engine ที่มีความสามารถสูง ผู้พัฒนาโปรแกรมสามารถออกแบบฐานข้อมูล MDB ด้วย Access และใช้ DAO จากโปรแกรมประยุกต์ Visual Basic ในการเปิดฐานข้อมูล เพิ่มและลบเร็คคอร์ด และ จัดการรานเซ็คชัน สิ่งที่ดีสุดของ DAO คือ ไม่จำกัดผู้ใช้กับ Jet database เพราะผู้ใช้สามารถเปิดฐานข้อมูลทุกชนิดที่มีไดร์ฟเวอร์ ODBC ได้โดยตรงหรือผู้ใช้สามารถใช้ Jet attached table ซึ่งเป็น table เสมือนที่ปรากฏตามฐานข้อมูล MDB แต่การดึงและเก็บข้อมูลจริงในแหล่งอื่นของ ODBC
- **RDO** หรือ Remote Data Object เป็นความพยายามครั้งแรกของ Microsoft ในการรวมความง่ายของ DAO กับความสามารถระดับสูงของ Direct ODBC API Programming โดย RDO เป็นแบบจำลองอ็อบเจคที่ไม่ชัดเจน ภายหลัง DAO แต่ใช้การข้าม Jet engine และ DLL ของ DAO และทำงานโดยตรงกับไดรฟเวอร์ ODBC โปรแกรมประยุกต์ที่มาจาก RDO โหลดเฉพาะ DLL จำนวนหนึ่งแทนการใช้ทรัพยากรจำนวนมากของ Jet engine โดยสิ่งที่สำคัญอยู่ที่การออกแบบเฉพาะของ RDO ให้ทำงานกับทรัพยากรของ ODBC ทำให้สามารถทำงานที่ไม่สามารถเข้าถึงได้โดย DAO เทคโนโลยีของ RDO เป็นเทคโนโลยี 32 บิตจึงไม่สามารถใช้ได้จากโปรแกรมประยุกต์ 16 บิต
- **ADO** หรือ Activex Data Object เป็นการติดต่อระดับสูงของ OLE DB มีบทบาทใกล้เคียง RDO ในการทำงานกับ API ของ ODBC ในขณะที่ OLE DB คล้ายกับ API ของ ODBC ที่เป็นการติดต่อระดับล่างที่ไม่สามารถเข้าถึงได้ง่ายจากภาษาระดับสูง เช่น Visual Basic เป็นต้น ADO สร้างบน OLE DB เพื่อให้การทำงานที่ไม่ให้ติดต่อโดยตรง ODBC หรือทำให้ผู้ใช้เขียนคำสั่งที่มีความสามารถ ADO สามารถเปรียบเทียบความสามารถกับ ADO คือทั้งคู่สามารถสร้างคิวรี่แบบ asynchronous และการติดต่อ ADO เพิ่มส่วนใหญ่จำนวนมาก เช่น File-based และ Stand-alone Recordset, hierarchical Recordset และอื่น

ด้วยรูปแบบการติดต่อกับฐานข้อมูลที่เป็นมาตรฐาน โดยติดต่อผ่านออบเจ็คตัวกลางที่เรียกว่า ADO (ActiveX Data Object) ซึ่งการติดต่อแบบนี้ทำให้เราเขียนโปรแกรมในรูปแบบเดียวกันทั้งหมด ไม่ว่าจะใช้ฐานข้อมูลชนิดใดก็ตาม



รูปที่ 2.53 แบบการใช้งานฐานข้อมูลผ่านออบเจ็ค ADO

- ADO.NET นี้จะใช้ DataSet และ DataReader ในการจัดการฐานข้อมูลที่เป็นตาราง หรือ เทเบิล (Table) ปรกติแล้ว ADO ติดต่อกับฐานข้อมูลที่ต้องมีตัวกลาง เช่น OLEDB , ODBC เป็นต้น แต่ใน ASP.NET นี้มีการสร้าง Name Space ไว้ให้เราติดต่อกับฐานข้อมูล SQL Sever โดยไม่ต้องผ่านตัวกลางเหล่านี้ทำให้สามารถใช้ได้อย่างสะดวกและมีประสิทธิภาพยิ่งขึ้น ส่วนการใช้งานผ่านตัวกลางนั้นยังคงสามารถทำได้ ซึ่งเท่ากับว่าเรามีทางเลือก 2 ทาง ในการติดต่อกับฐานข้อมูลมีการส่งข้อมูลในรูปแบบ XML เพื่อใช้แก้ปัญหาที่รับส่งข้อมูลผ่าน Firewall เนื่องจากการส่งข้อมูลรูปแบบ XML จะใช้โปรโตคอล HTTP เช่นเดียวกับข้อมูลแบบ HTML

บทที่ 3

หลักการออกแบบและดำเนินการ

ในโครงงานนี้ได้ทำการศึกษาและรวบรวมข้อมูลเกี่ยวกับ เส้นทาง เดินรถประจำทางภายในกรุงเทพฯ เดิมทีนั้นการเดินทางภายในกรุงเทพฯจากสถานที่หนึ่งไปยังอีกสถานที่หนึ่งไม่ใช่เรื่องง่าย ยิ่งระยะทางยิ่งไกลก็อาจจะทำให้ผู้เดินทางเกิดอาการเบื่อหน่าย และหากผู้เดินทางใช้รถส่วนตัวด้วยแล้วละก็อาจทำให้สิ้นเปลืองพลังงานมากกว่าเดิม

จากปัญหาดังกล่าวมาทั้งหมดนี้ ทำให้ผู้พัฒนาคิดได้ว่า ถ้าเราสามารถจัดเก็บข้อมูลของรถประจำทางไว้ในฐานข้อมูล และดึงฐานข้อมูลออกมาเสนอเป็นรูปแบบของ WAP Card ให้ผู้ใช้บริการรถประจำทางศึกษาเส้นทางและพิจารณาก่อนที่จะออกเดินทาง ก็จะสามารถช่วยลดปัญหาที่เกิดขึ้นได้

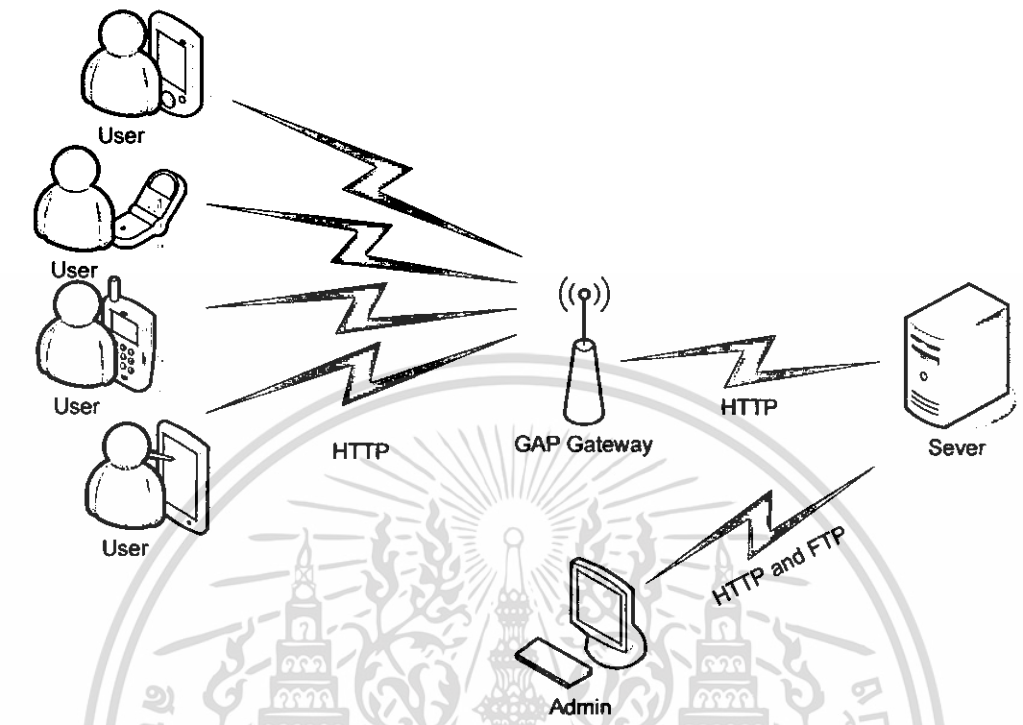
จุดประสงค์การออกแบบโครงงาน

- ออกแบบมาเพื่อใช้งานกับโทรศัพท์มือถือ หรือ PDA และ Pocket PC
- ออกแบบมาเพื่อการค้นหารายละเอียดข้อมูลของสายรถเมล์ที่ต้องการ
- ออกแบบมาเพื่อหาสายรถเมล์ที่ต้องการเพื่อไปยังสถานที่ต่าง

3.1 การออกแบบโครงงาน

- ใช้ ASP.net Mobile Web Application ใน Visual basic.net 2003
- ใช้ Microsoft access 2003 ในการออกแบบฐานข้อมูล และเชื่อมต่อ Visual basic.net 2003 ในการนำฐานข้อมูลไปใช้งาน
- ใช้งานมือถือสามารถเชื่อมต่ออินเตอร์เน็ตได้

3.2 แนวคิดในการวิเคราะห์ออกแบบระบบ



รูปที่ 3.1 บล็อกไดอะแกรมการทำงานของระบบ

การทำงานจะแบ่งออกเป็น 2 ส่วนคือ

- **ส่วนของผู้ดูแลระบบ (Administer)** ทำหน้าที่ในการแก้ไขข้อมูลให้มีความทันสมัย และป้อนข้อมูลสถานที่เพิ่มเติม โดยจะมีโปรแกรมที่ใช้ในการเข้าถึงฐานข้อมูลที่เซิร์ฟเวอร์โดยทำงานในรูปแบบของไคลเอนต์ / เซิร์ฟเวอร์
- **ส่วนผู้ใช้ (User)** จะสามารถใช้บริการได้โดยการสร้างการเชื่อมต่อเข้าสู่เครือข่ายอินเตอร์เน็ตที่ให้บริการโดยผู้ให้บริการเครือข่าย จากนั้นจะเรียกใช้งาน WAP application ตาม URL ที่กำหนด เมื่อผู้ใช้เข้าสู่ WAP จะสามารถเลือกสถานที่ต้นทางและปลายทาง เมื่อได้รับข้อมูลสถานที่ต้นทางและปลายทางแล้ว ระบบจะค้นหาสายรถประจำทางที่ผ่านเส้นทางนั้น ซึ่งระบบสามารถที่จะแสดงได้มากกว่า 1 เส้นทาง เพื่อให้ผู้ใช้งานได้ตัดสินใจเลือกเส้นทางเอง

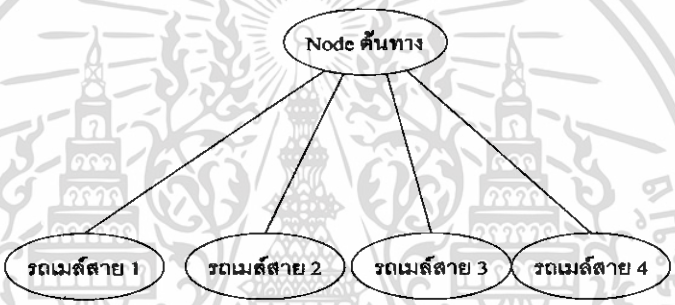
3.3 การออกแบบการทำงานทางฝั่งผู้ให้บริการ (Server Design)

หลักการในการทำงานของโปรแกรมเป็นลักษณะของ Tree

โปรแกรมจะเริ่มทำงานจากรับค่าจาก User คือ ต้นทาง และปลายทาง โปรแกรมจะทำการสร้างทั้ง 2 ค่านี้เป็นโหนด ดังรูป

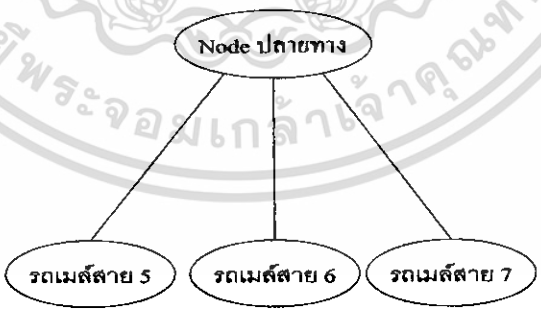
รูปที่ 3.2 ทำการสร้าง Root Node 2 node

โปรแกรมจะค้นหาว่าโหนดสถานที่ต้นทางนั้น มีรถเมล์สายอะไรบ้างที่ผ่านสถานที่นั้น ๆ จากฐานข้อมูล



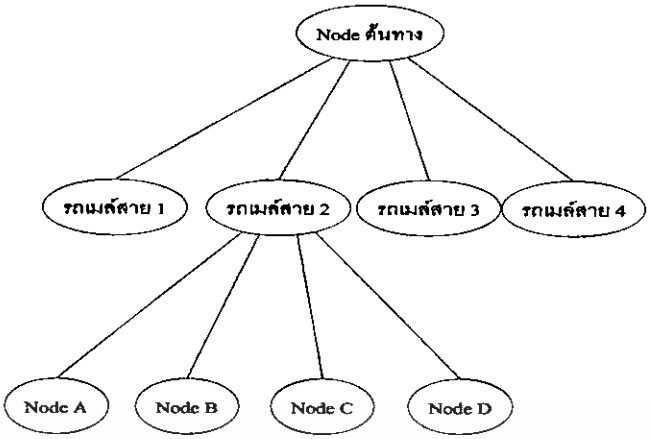
รูปที่ 3.3 แสดง Parent Node ของ Root Node ต้นทาง

ที่โหนดสถานที่ปลายทาง จะค้นหาเช่นเดียวกันกับโหนดสถานที่ต้นทาง



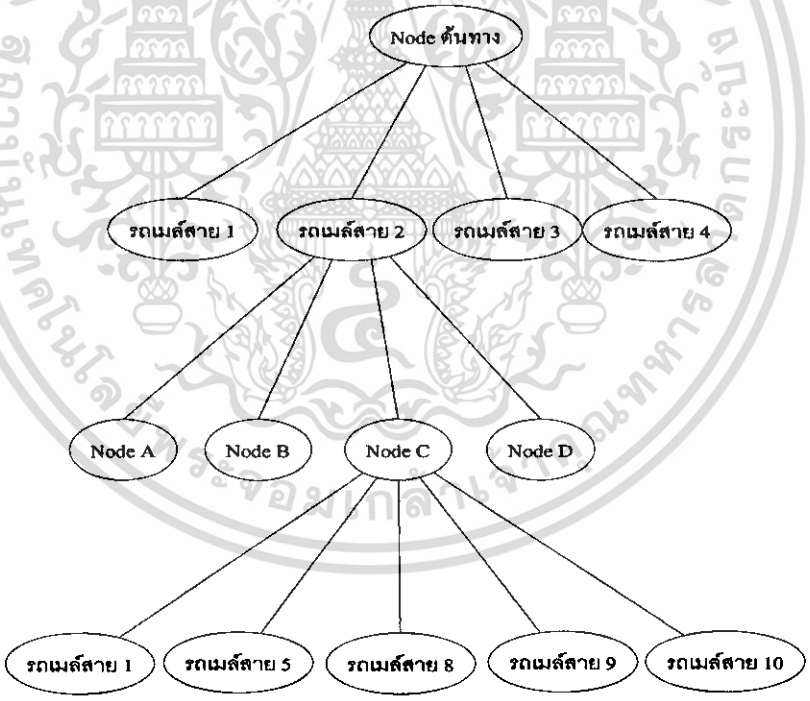
รูปที่ 3.4 แสดง Parent Node ของ Root Node ปลายทาง

หลังจากนั้นโปรแกรมจะเอาข้อมูลสายรถเมล์ทั้ง 2 Root Node มาเปรียบเทียบกันว่าเหมือนกันหรือไม่ ถ้าไม่มีข้อมูลโหนดสายรถเมล์ที่เหมือนกัน จะนำ Root Node ต้นทาง ทำการค้นหาต่อจาก Child Node ต้นทาง ทุก ๆ โหนดว่า Child Node รถเมล์สายนั้นผ่านสถานที่ใดบ้าง



รูปที่ 3.5 แสดงการค้นหา Leaf Node ของ Child Node(สายรถเมล์สาย 2)

โปรแกรมจะค้นหาว่าโหนดสถานที่ต้นทางนั้น มีรถเมล์สายอะไรบ้างที่ผ่านสถานที่นั้น ๆ จากฐานข้อมูลซ้ำอีกจนกว่าจะเจอ Leaf Node สายรถเมล์ที่เท่ากับ leaf Node ของ Root Node ปลายทาง



รูปที่ 3.6 แสดงการค้นหา Leaf Node ของ Child Node(Node C)

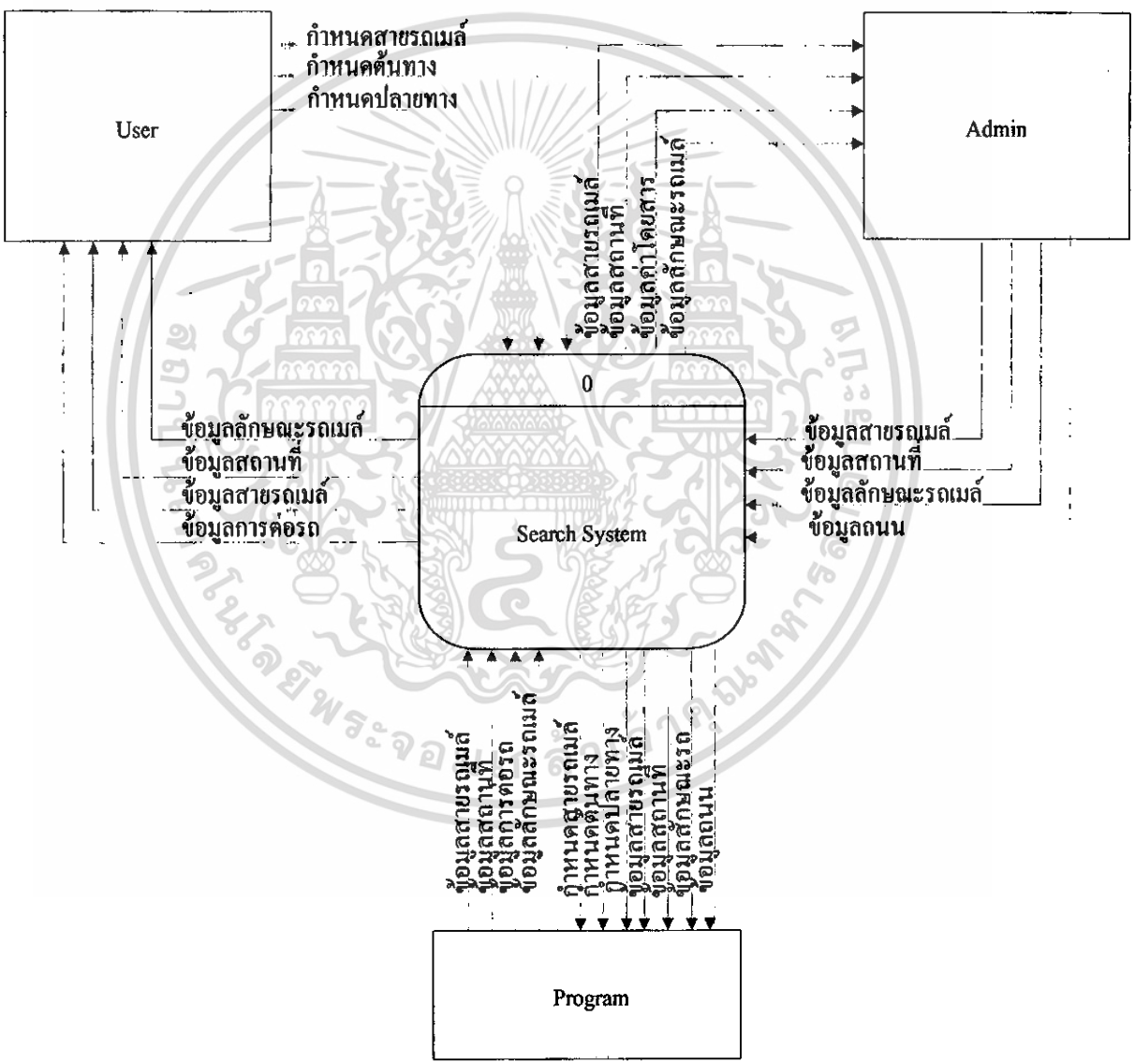
จากตัวอย่างจะทำการดึงข้อมูลจาก array มาเรียงเป็น Queues เก็บเป็น stack เป็นดังนี้ " รถเมล์สาย5 ,Node C , รถเมล์สาย2 , Node ต้นทาง " แล้วจึงเอาข้อมูลออกมาจาก stack มาแสดงได้เป็น " จาก Node ต้นทาง นั่ง รถเมล์สาย2 ลงต่อรถที่ Node C ต่อ รถสาย 5 " เป็นต้น

3.4 การออกแบบส่วนติดต่อกับผู้ใช้งาน (Client Design)

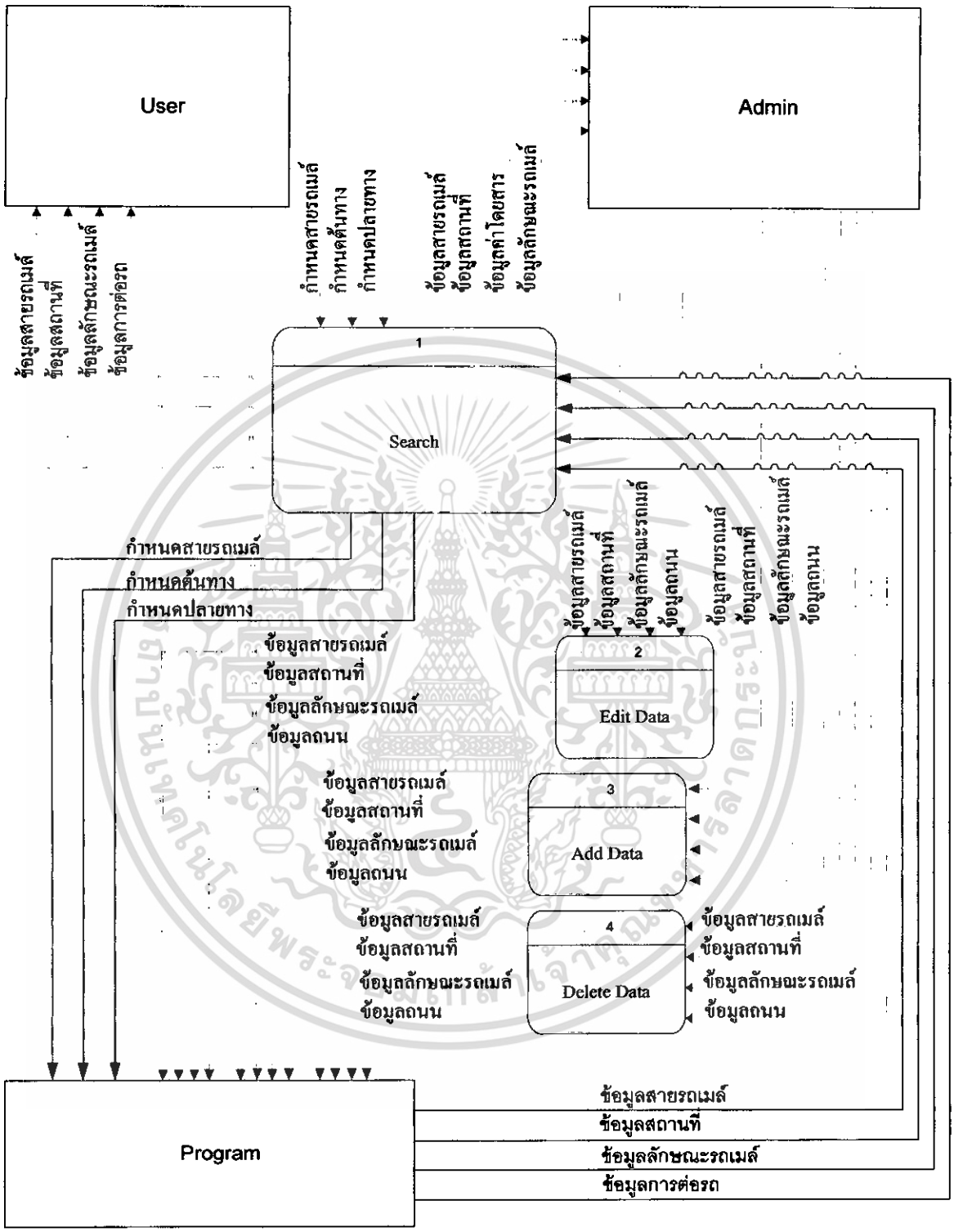
WAP ที่ออกแบบมานี้เน้นการใช้งานทั่วไป และสิ่งที่ต้องคำนึงถึงจึงมีดังต่อไปนี้

- คำนึงถึงการใช้งานที่ง่าย ไม่ซับซ้อน สามารถเข้าใจ และ ปฏิบัติตามขั้นตอนการค้นหาเส้นทางได้อย่างถูกต้อง
- เน้นการแสดงผล หรือ ติดต่อกับผู้ใช้งานผ่านข้อความหรือรูปภาพ หรือ รูปแบบการแสดงผลอื่นๆเพื่อให้ผู้ใช้เข้าใจได้ง่าย

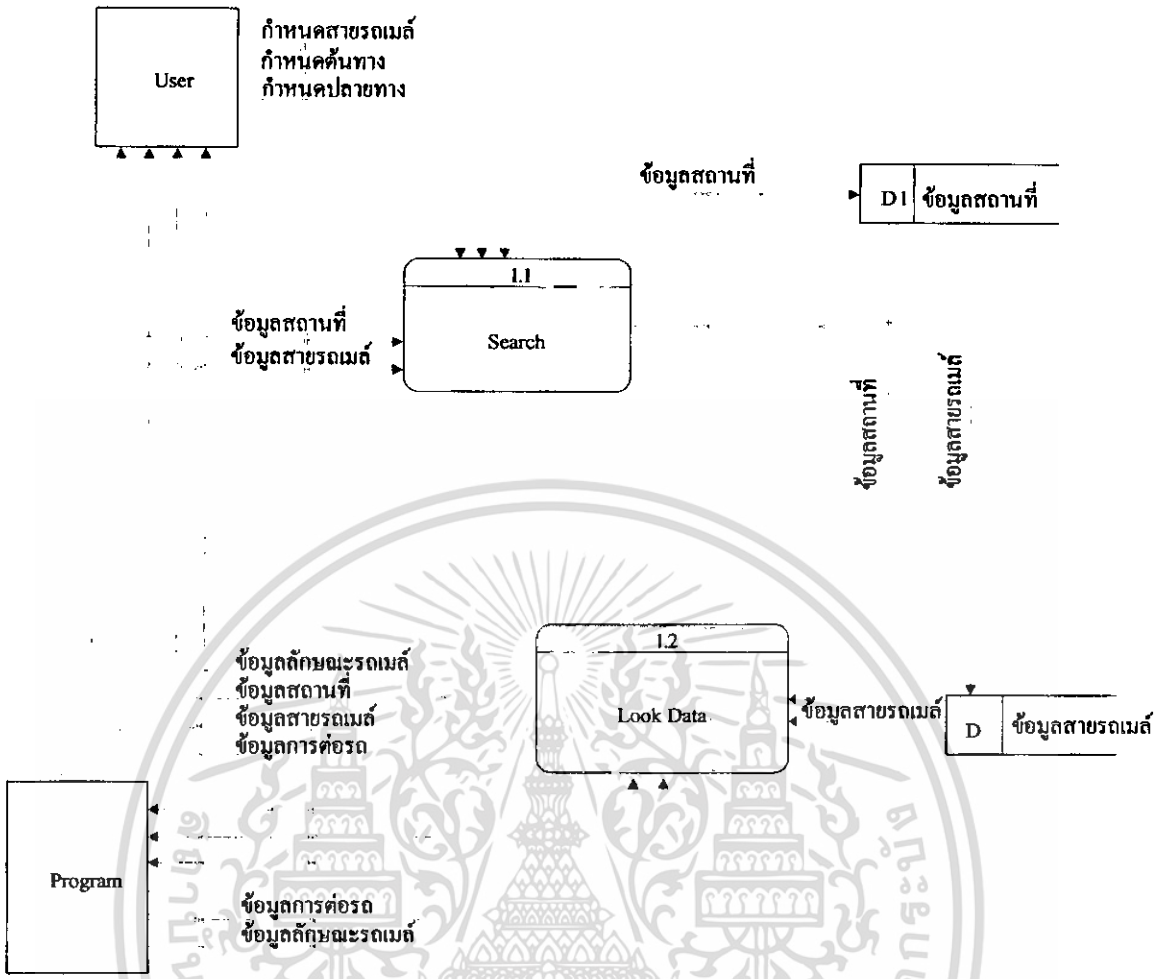
3.4.1 การออกแบบผังการไหลของข้อมูล (Data Flow Diagram)



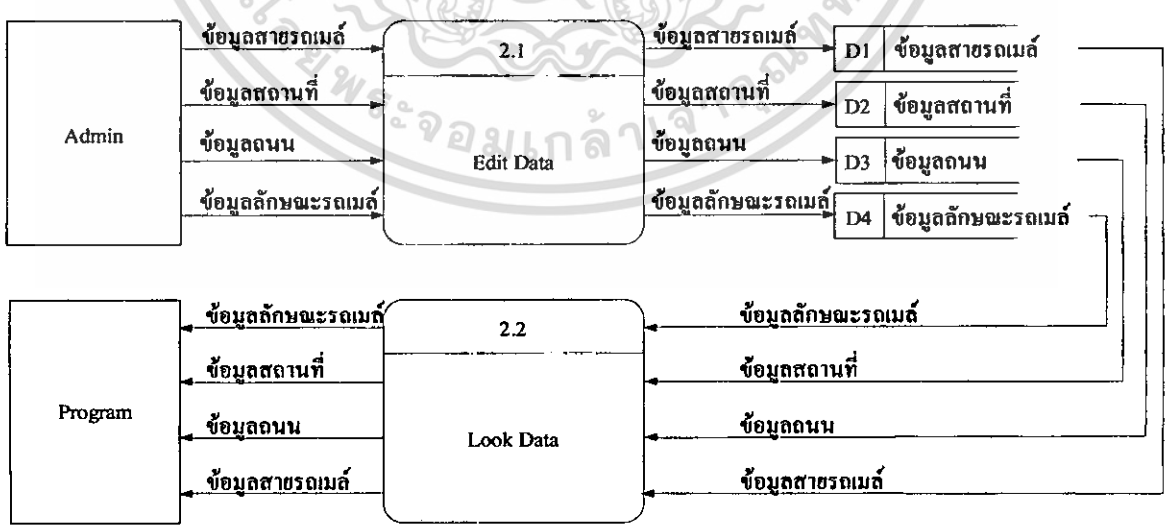
รูปที่ 3.7 แสดงรูปแบบคอนเท็กซ์ไดอะแกรมทั้งระบบ



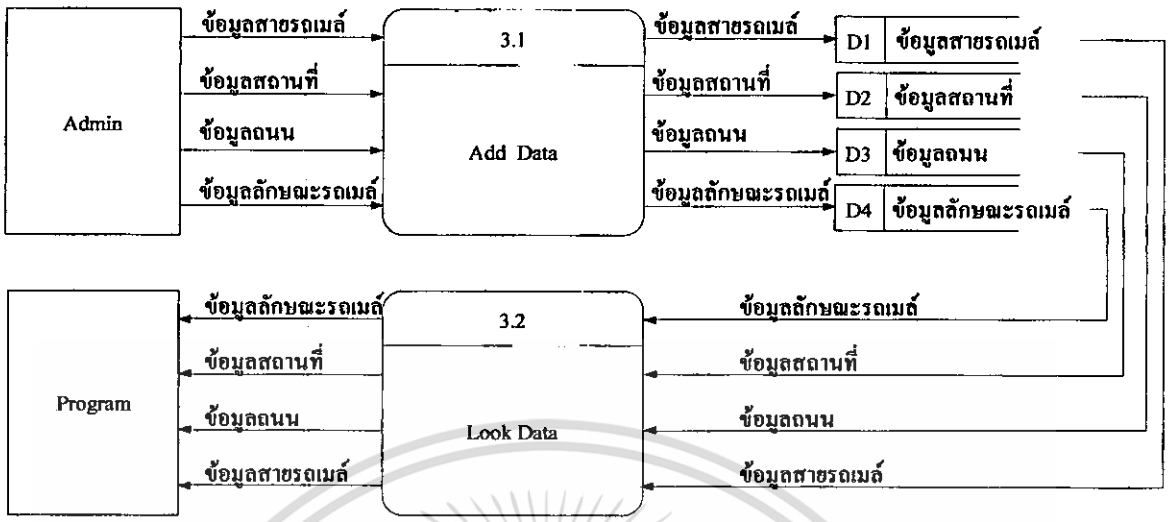
รูปที่ 3.8 แสดงรูปดอนแท็กซ์ไดอะแกรมที่ Level 0



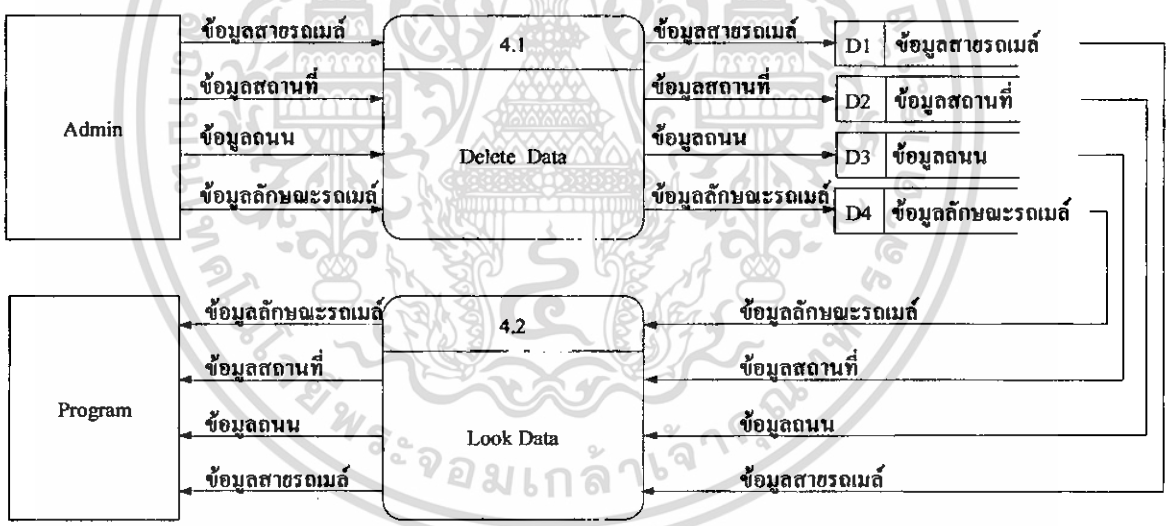
รูปที่ 3.9 แสดงรูปดอนแท็กซ์ไดอะแกรมที่ Level 1



รูปที่ 3.10 แสดงรูปดอนแท็กซ์ไดอะแกรมที่ Level 1 ส่วนของ Edit Data



รูปที่ **3.11** แสดงรูปดอนแท็กซ์ไดอะแกรมที่ Level 1 ส่วนของ Add Data



รูปที่ **3.12** แสดงรูปดอนแท็กซ์ไดอะแกรมที่ Level 1 ส่วนของ Delete Data

3.4.2 การออกแบบ WAP Design

เพื่อความง่ายและเข้าใจได้ไวจึงได้มีการออกแบบการ link ไปยัง page ต่างของ WAP เราสามารถทำการออกแบบเป็น WAP Diagram ได้ดังต่อไปนี้

รูปที่ 3.8 แสดงการเชื่อม link เพื่อไปยัง page ต่าง ๆ ของ WAP

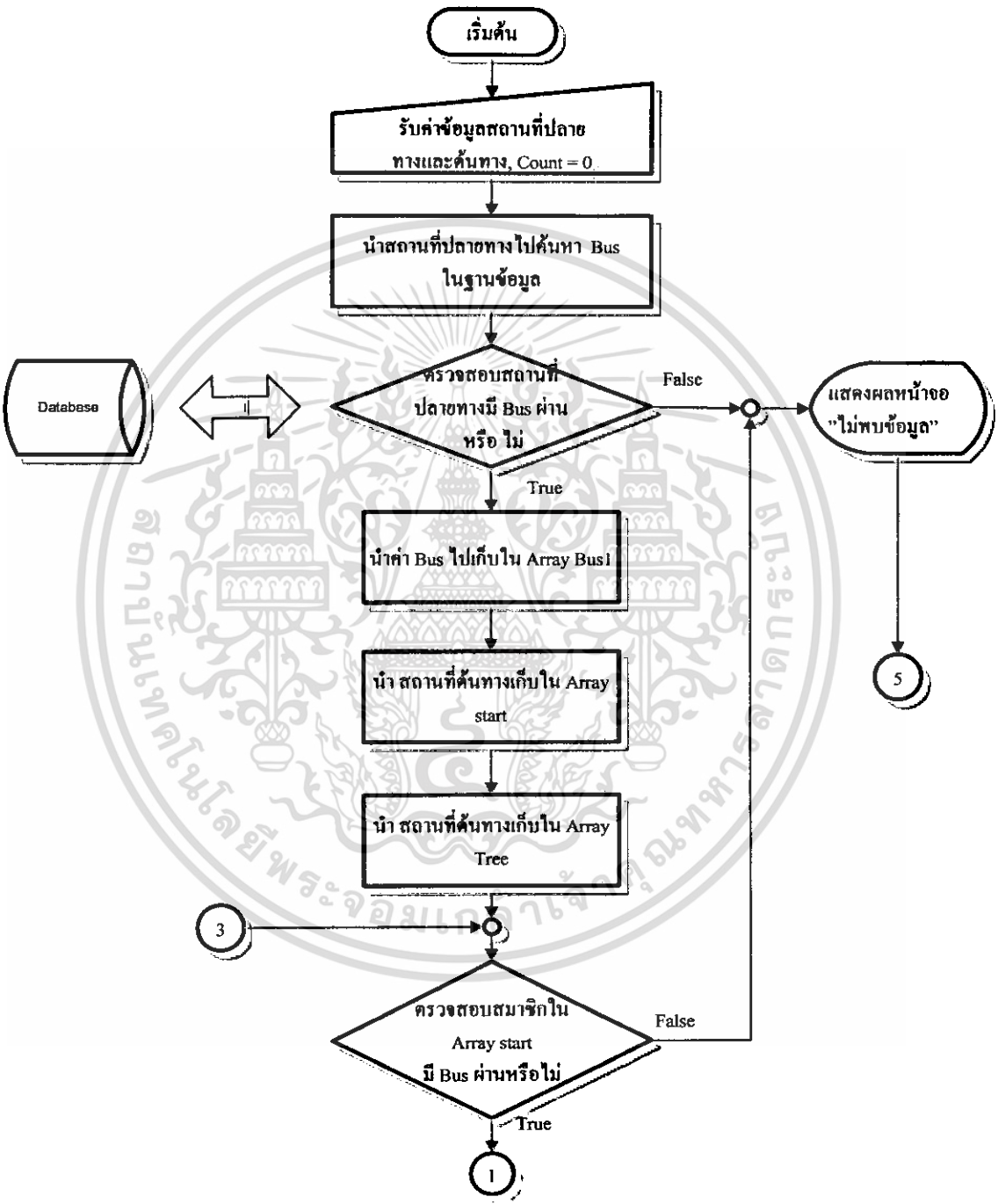
หน้า Main Menu หน้านี้จะมีจุด link ให้เลือกการทำงานอยู่ 2 จุด คือ ค้นหาขอมูล และ คนหาเส้นทางเดินรถเมล์

หน้า ค้นหาข้อมูลรถเมล์ หน้านี้ จะมีการให้กรอกชื่อของรถเมล์ที่ต้องการทราบขอมูล เช่น ที่ช่องสายรถเมล์ กรอก 151 ที่ช่องชนิด เลือก ธรรมดา แล้ว กดปุ่ม search ก็จะ link ต่อไปยังหน้าแสดงข้อมูลของรถเมล์ เพื่อแสดงข้อมูลให้ทราบ เป็นต้น

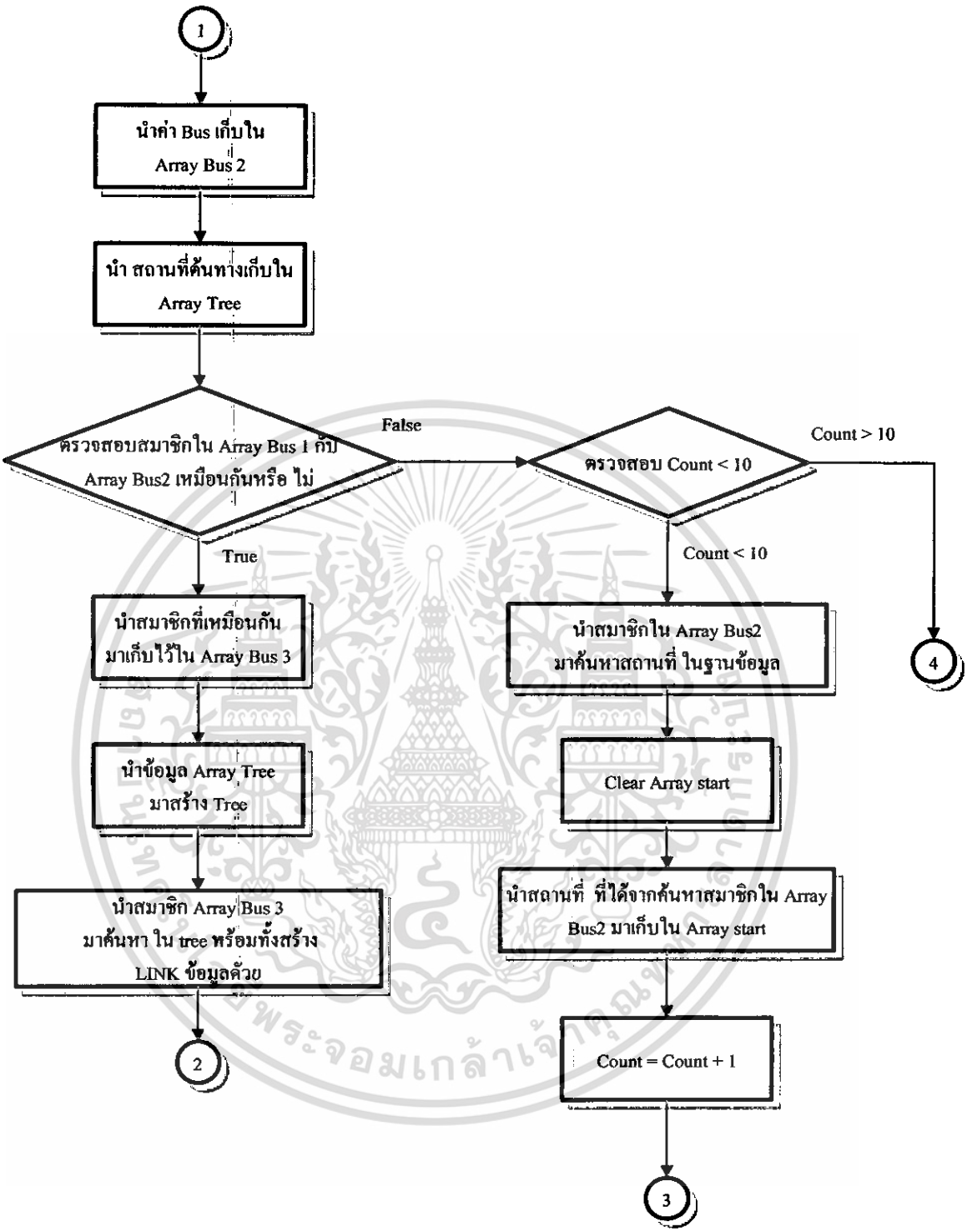
หน้า ค้นหาเส้นทางเดินรถเมล์ หน้านี้ จะมีการให้กรอกสถานที่ต้นทาง และสถานที่ปลายทาง เช่น ที่ช่องสถานที่ต้นทาง กรอก 151 ที่ช่องชนิด เลือก ธรรมดา แล้ว กดปุ่ม search ก็จะ link ต่อไปยังหน้า แสดงข้อมูลของรถเมล์ เพื่อแสดงข้อมูลให้ทราบ เป็นต้น

3.5 แสดงการทำงานของระบบด้วย Flow Chart

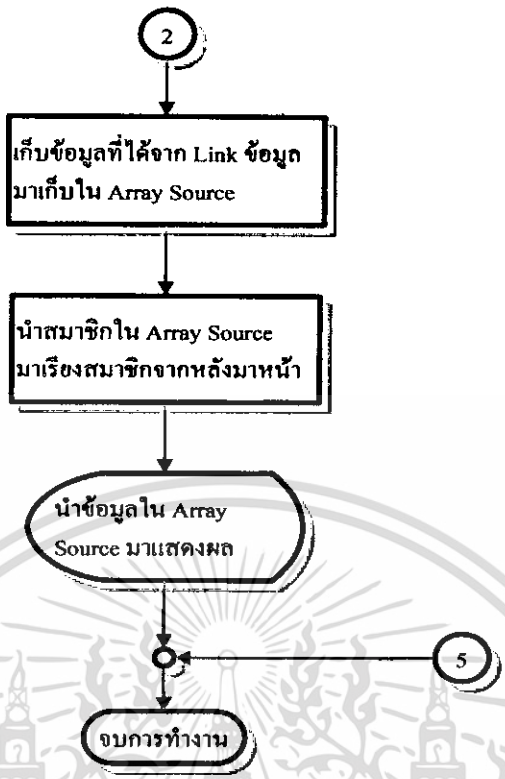
3.5.1 จากแผนผัง Flow Chart ด้านล่างสามารถอธิบายการทำงานของระบบการค้นหาเส้นทางที่เหมาะสมและต่อรถได้ 10 ต่อ พร้อมกับหาเส้นทางรถประจำทางได้



รูปที่ 3.13(ก) การค้นหาข้อมูลรถเมล์และการต่อรถเมล์

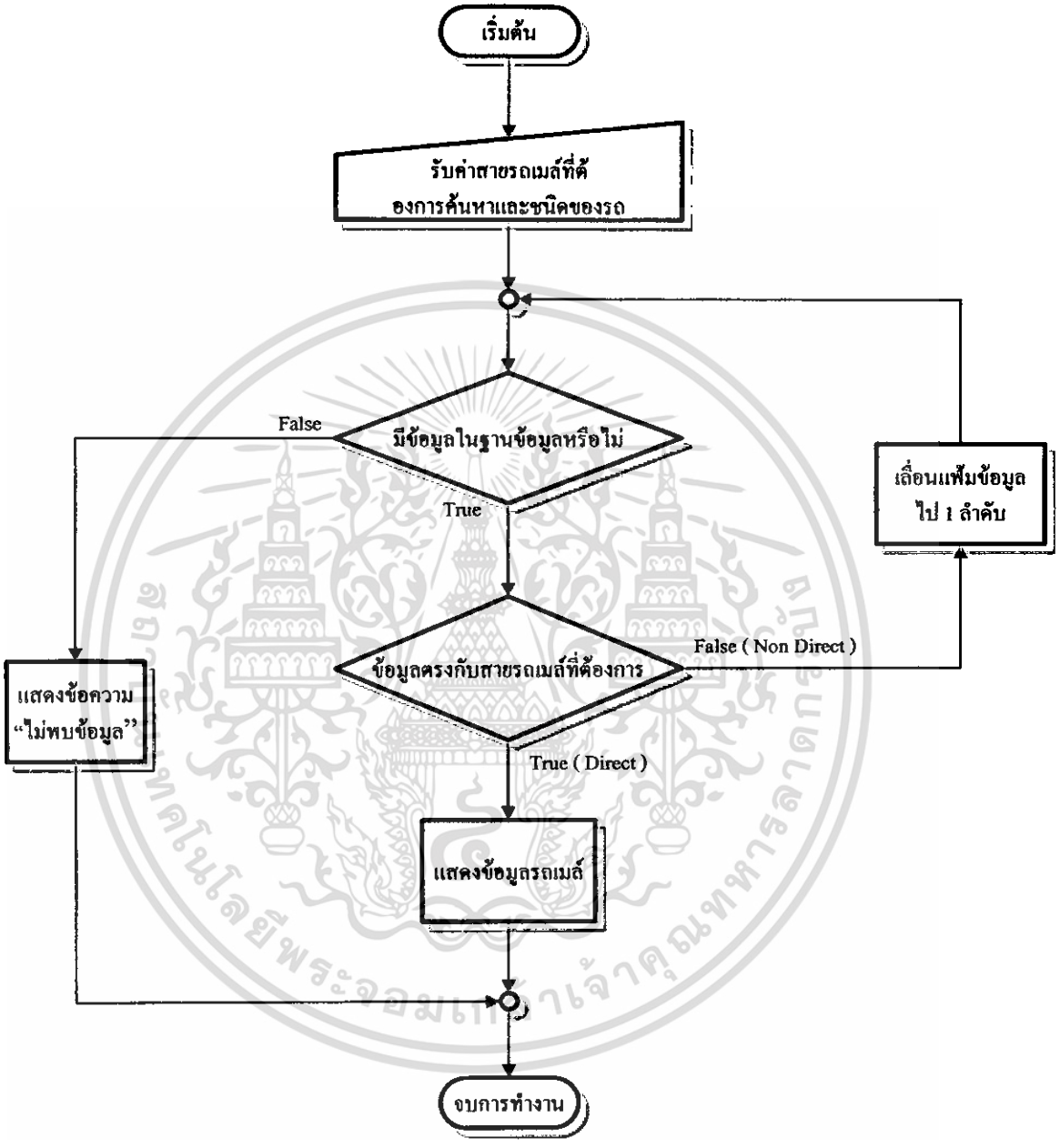


รูปที่ 3.13(ข) การค้นหาข้อมูลรถเมล์และการต่อรถเมล์



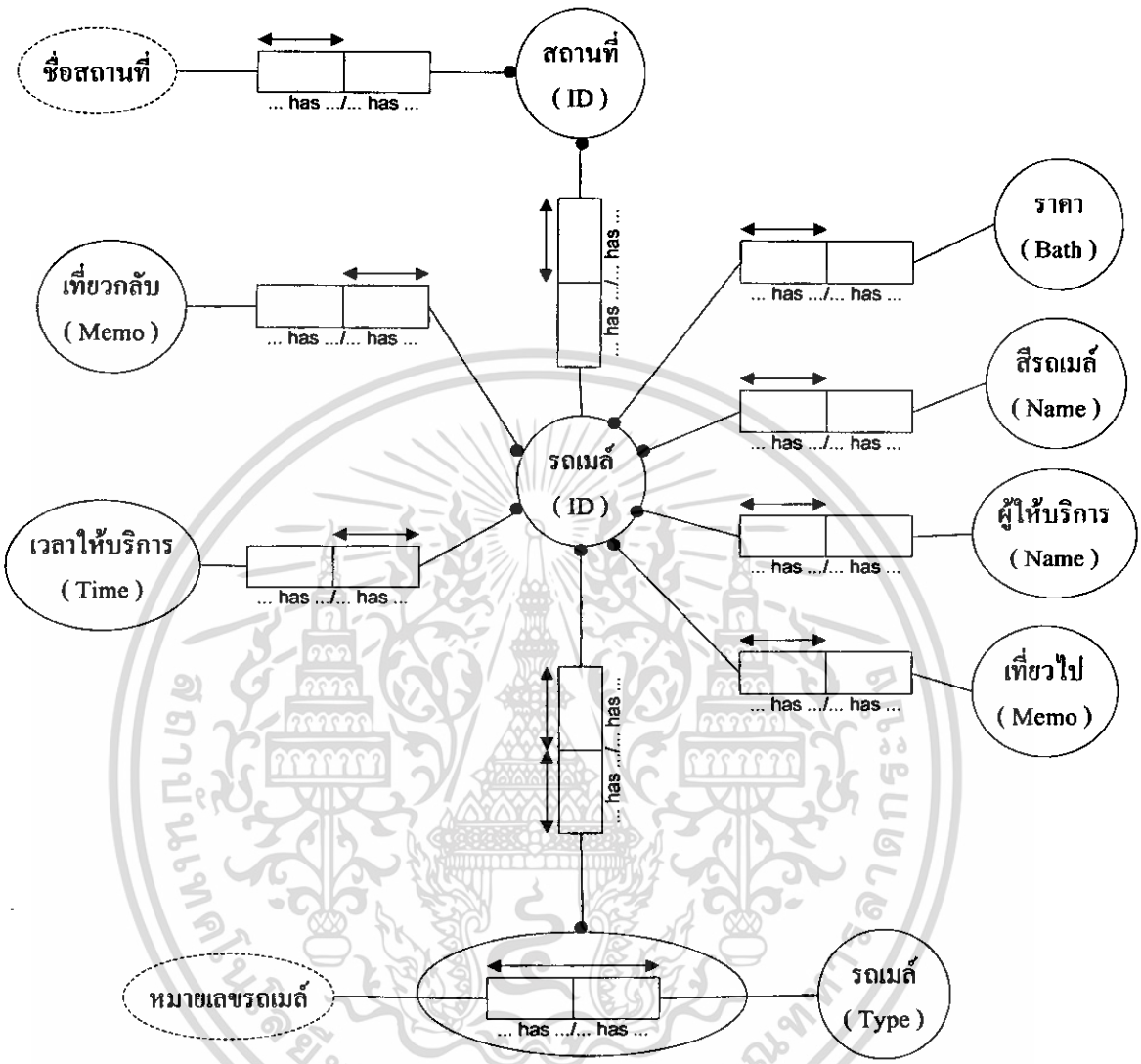
รูปที่ 3.13(ค) การค้นหาข้อมูลรถเมล์และการต่อรถเมล์

3.5.2 จากแผนผัง Flow Chart ด้านล่างสามารถอธิบายการทำงานของระบบการค้นหาข้อมูลรถของประจำทางได้



รูปที่ 3.14 แสดงการทำงานค้นหาข้อมูลรถเมล์

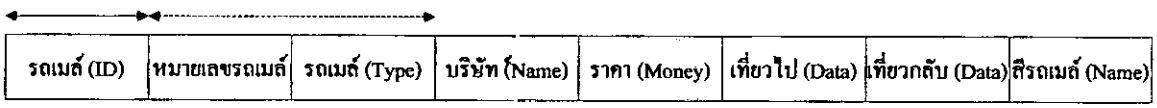
3.6 การออกแบบฐานข้อมูลด้วยไนแอม



รูปที่ 3.15 แผนภาพไนแอมของระบบ

3.7 MAPPING จาก NIAM

เนื่องจากถ้าทำจาก ใช้ Access เท่านั้นจึงจะใช้ภาษาไทยในการ MAP ตารางได้ทางผู้พัฒนาจึงอยากจะเปลี่ยนมาเป็นภาษาอังกฤษเพื่อนที่จะได้เอาไปพัฒนาต่อได้จึงเปรียนการ MAPPING จากรูปแบบของ NIAM ได้ดังนี้



รถเมล์ (ID)	หมายเลขรถเมล์	รถเมล์ (Type)	บริษัท (Name)	ราคา (Money)	เที่ยวไป (Data)	เที่ยวกลับ (Data)	สีรถเมล์ (Name)

ตารางที่ 3.1 แสดงการ map ตาราง Info ในฐานข้อมูล

สถานที่ (ID)	รถเมล์ (ID)	ชื่อสถานที่

ตารางที่ 3.2 แสดงการ map ตาราง Place ในฐานข้อมูล

3.8 DATA DICTIONARY

ในการ Mapping ตารางจะต้องมีการบอกคุณลักษณะหรือรายละเอียดเกี่ยวกับตารางนั้นๆ เพื่อที่ผู้ที่จะพัฒนา ฐานข้อมูลจะได้เข้าใจว่าตารางนั้นๆคืออะไร

ตารางที่ 3.3 แสดง Data Dictionary ตาราง Info ในฐานข้อมูล

Name	Type	Key	Null	Meaning
Bus_ID	VARCHAR(30)	PK	NO	รหัสของรถเมล์
Bus_No	INT(10)		NO	ประเภทของรถเมล์
Bus_Type	VARCHAR(30)		NO	รายละเอียดของรถเมล์
Go	MEMO		NO	รายละเอียดเที่ยวไป
Back	MEMO		NO	รายละเอียดเที่ยวกลับ
Price	INT(10)		NO	ราคาค่าโดยสาร
Service	VARCHAR(30)		NO	บริษัทที่ให้บริการ
Color	VARCHAR(30)		NO	สีของรถเมล์

ตารางที่ 3.4 แสดง Data Dictionary ตาราง Place ในฐานข้อมูล

Name	Type	Key	Null	Meaning
Place_ID	VARCHAR(30)	PK	NO	รหัสของรถเมล์
Bus_ID	VARCHAR(30)		NO	รหัสของสถานที่
Place_Name	VARCHAR(30)		NO	ชื่อของสถานที่

บทที่ 4

ผลการทดลอง

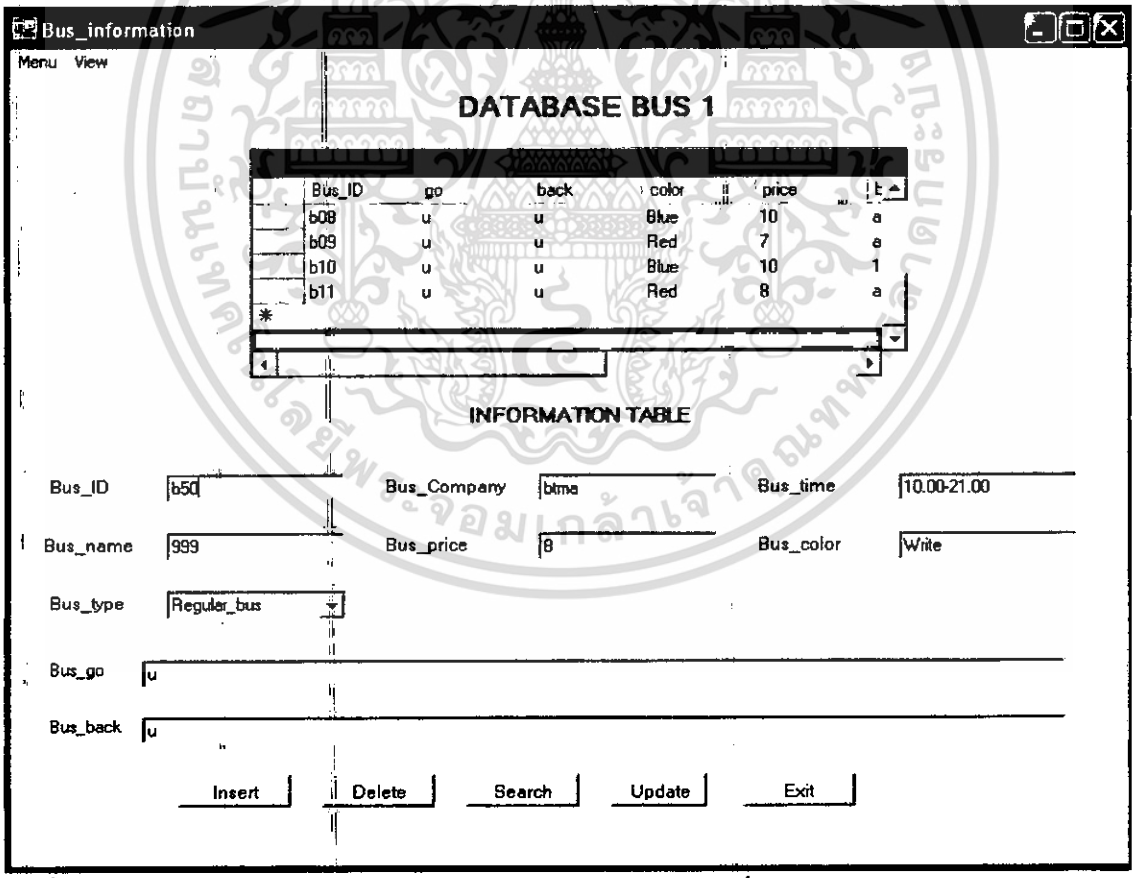
4.1 การจัดการข้อมูลในฐานข้อมูล

เนื่องจากฐานข้อมูลที่ออกแบบไว้มี 2 ตาราง จึงได้มีการออกแบบ From การ Update ข้อมูลเป็น Application ได้ 2 ฟังก์ชันการทำงานดังนี้

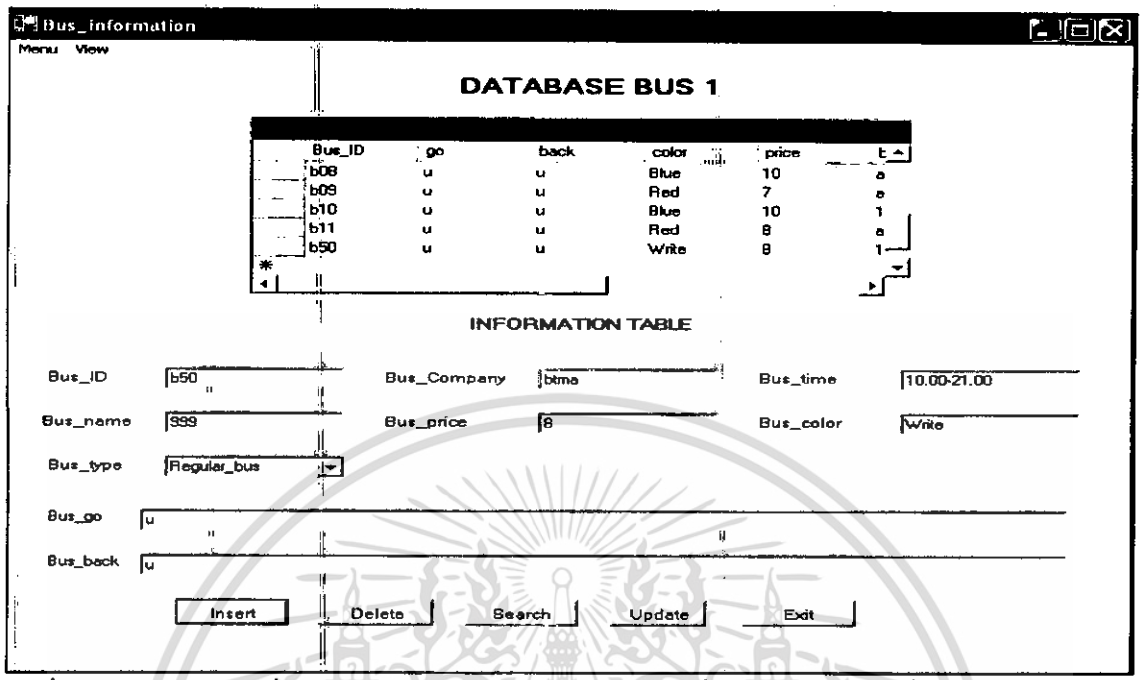
4.1.1 ขั้นตอนการจัดการฐานข้อมูลของผู้ดูแลระบบ

4.1.1.1 การ Insert ข้อมูล

1. **ตาราง Information**

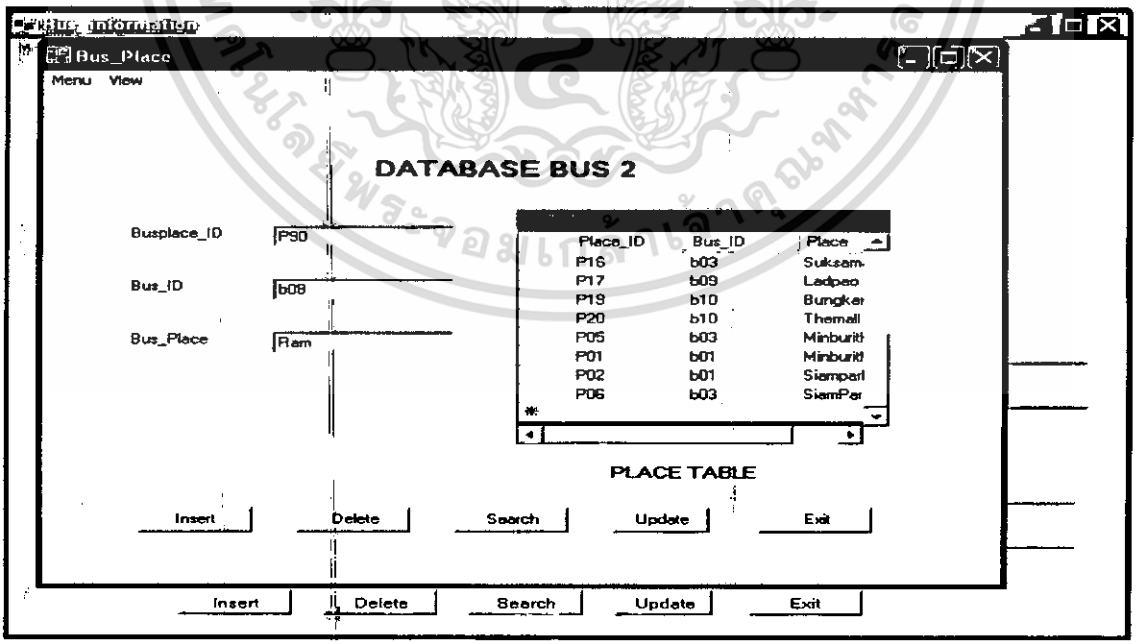


รูปที่ 4.1 จากตาราง Information ทำการกรอกข้อมูลลงในช่องว่าง เมื่อกรอกครบแล้วกดปุ่ม Insert

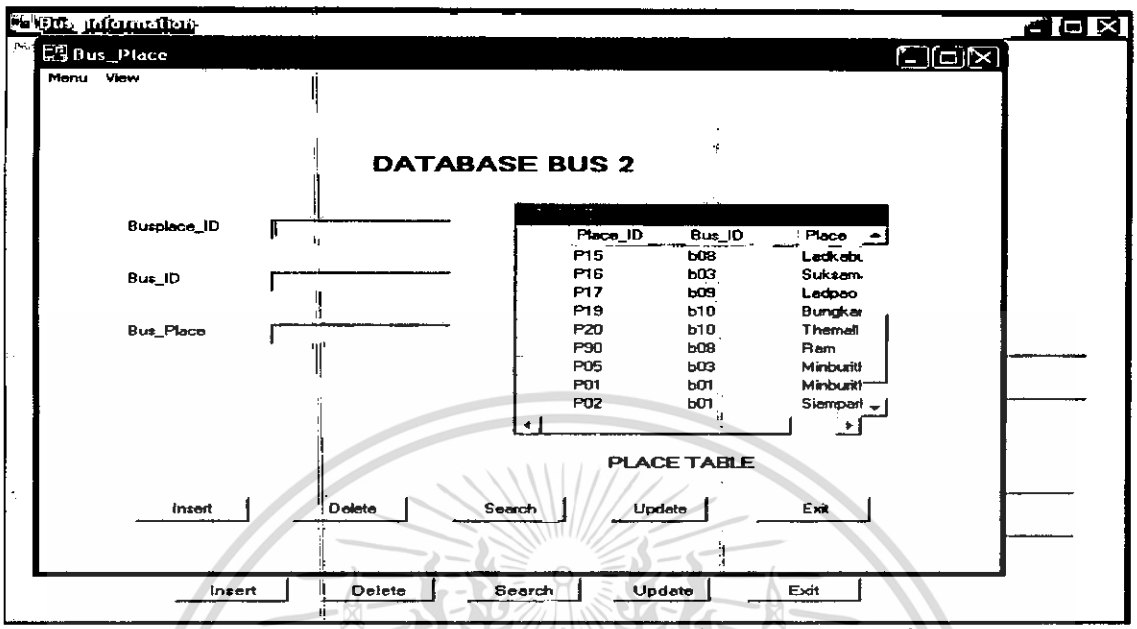


รูปที่ 4.2 แสดงหลังจากที่กดปุ่ม Insert แล้วก็จะเห็นได้ว่าข้อมูลที่ใส่จะไปแสดงอยู่ที่ตารางเป็นอันเสร็จการ Insert ข้อมูล

2. ตาราง Place



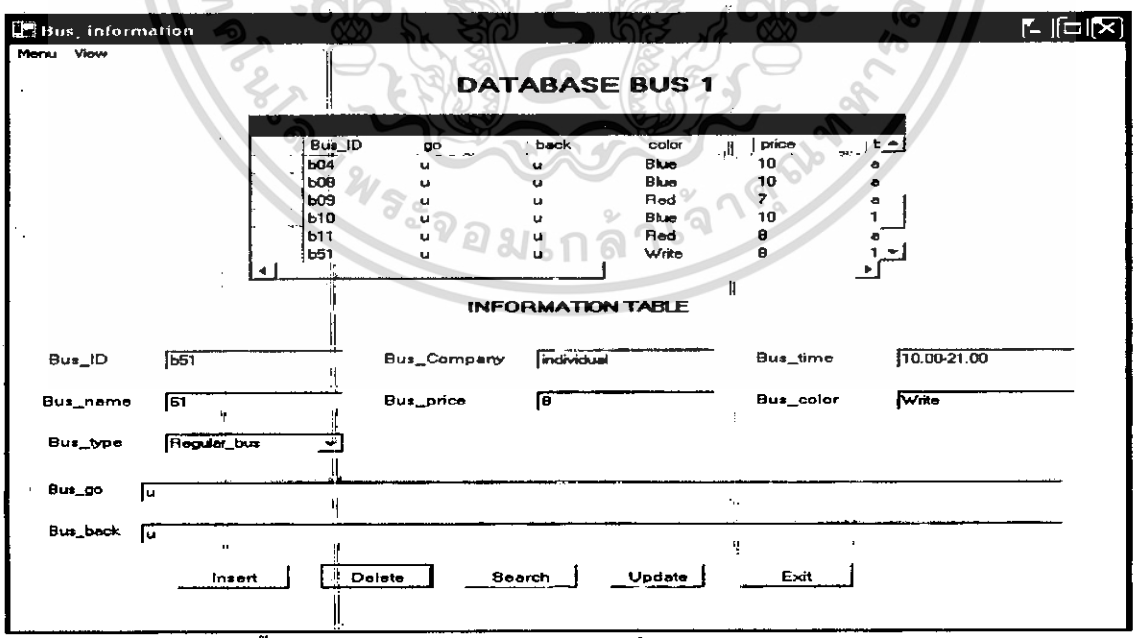
รูปที่ 4.3 จากตาราง Place ทำการกรอกข้อมูลลงในช่องว่าง เมื่อกรอกครบแล้วกดปุ่ม Insert



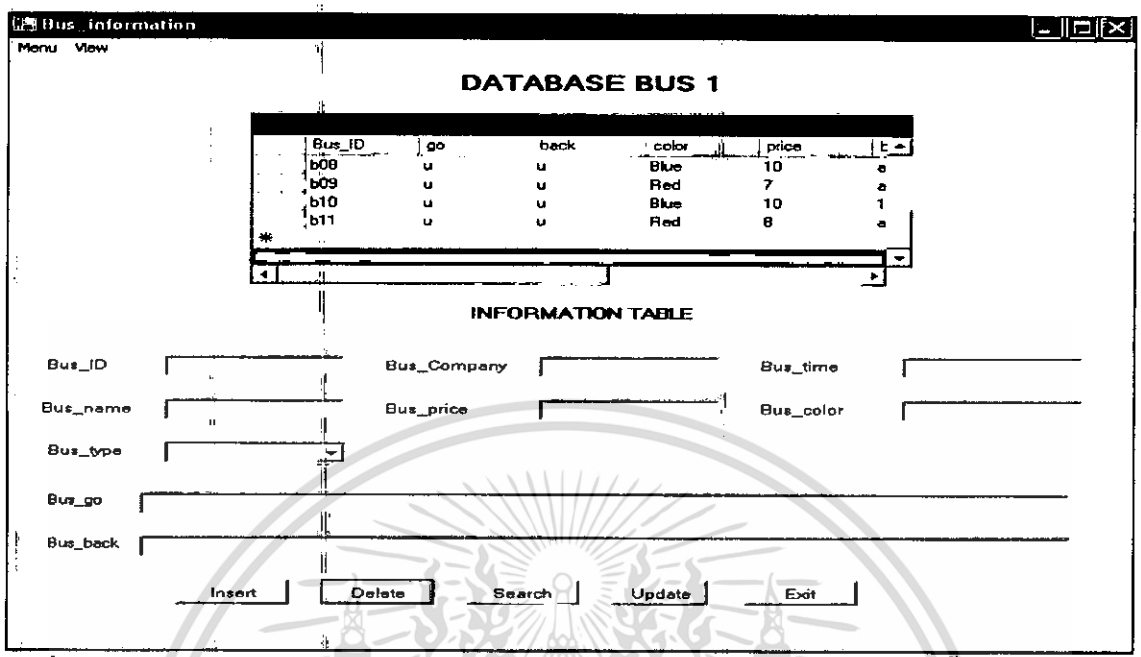
รูปที่ 4.4 แสดงหลังจากที่กดปุ่ม Insert แล้วก็จะเห็นได้ว่าข้อมูลที่ใส่จะไปแสดงอยู่ที่ตารางเป็นอันเสร็จการ Insert ข้อมูล

4.1.1.2 การ Delete ข้อมูล

1. **ตาราง Information**

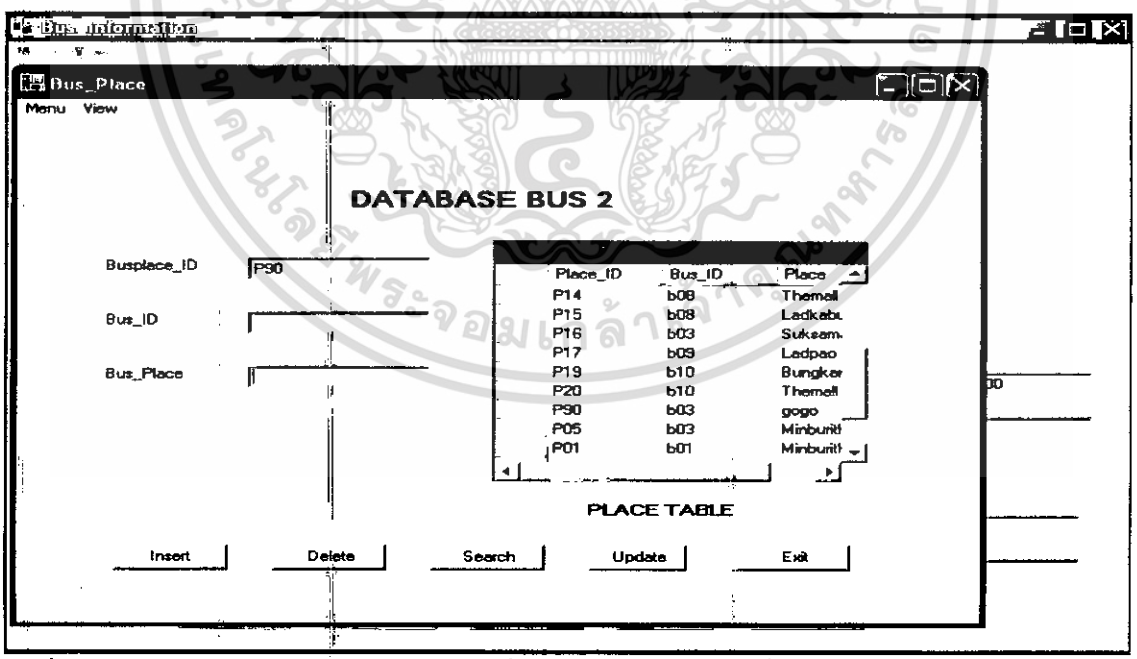


รูปที่ 4.5 ใส่ค่าให้ครบทั้งหมดหรือจะใส่เฉพาะค่ารหัสรถที่ช่อง Bus_ID แล้วกดปุ่ม Delete ก็ได้

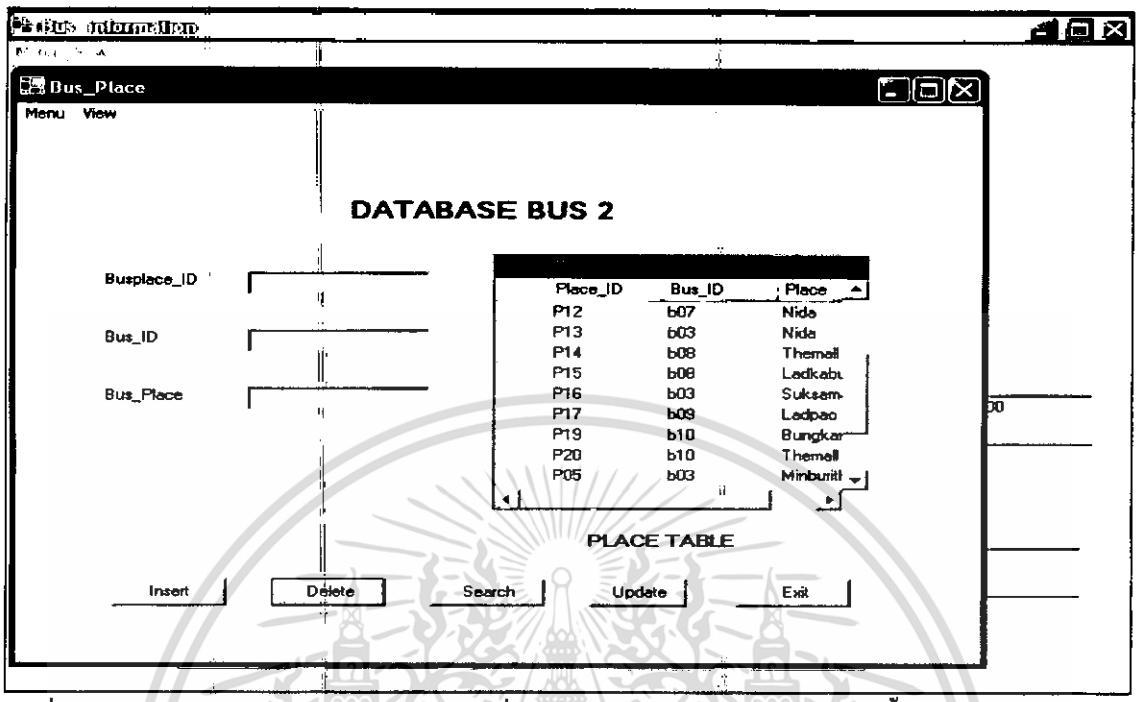


รูปที่ 4.6 หลังจากทำการกดปุ่ม Delete จะเห็นว่าข้อมูลใน Record ดังกล่าวก็จะหายไปทั้ง Reccord เป็นอันเสร็จสิ้นการลบข้อมูล

2. ตาราง Place



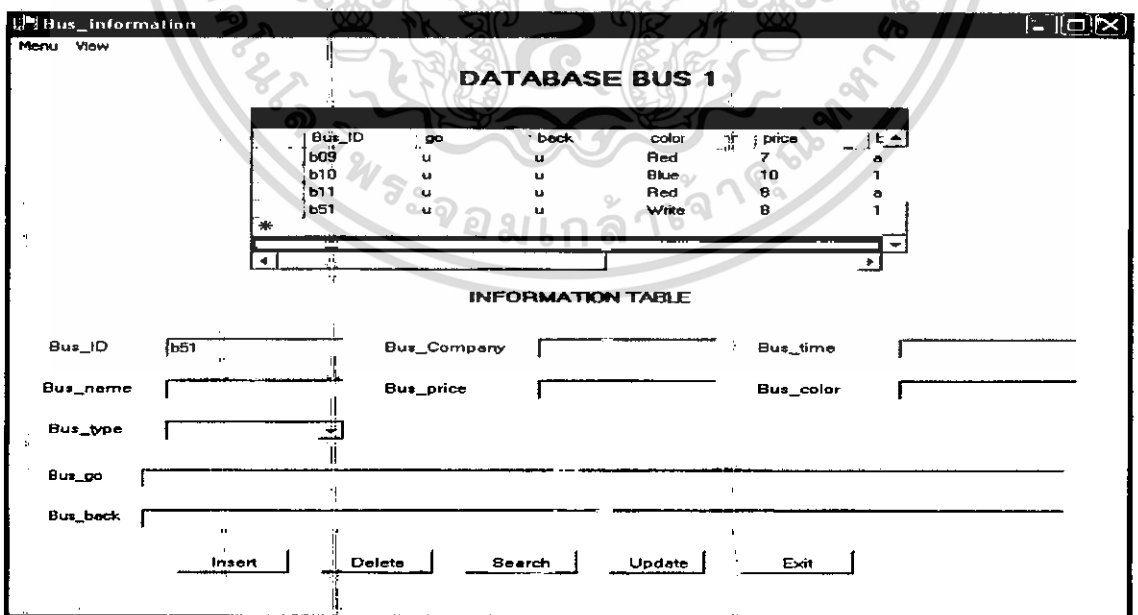
รูปที่ 4.7 ทำการกรอกข้อมูลลงในทุกช่องโดยที่ช่อง Bus_ID จะต้องมีที่ตาราง Infomation จริง แล้วกดปุ่ม Delete หรือจะใส่เฉพาะค่า Busplace_ID ค่าเดียวก็ได้



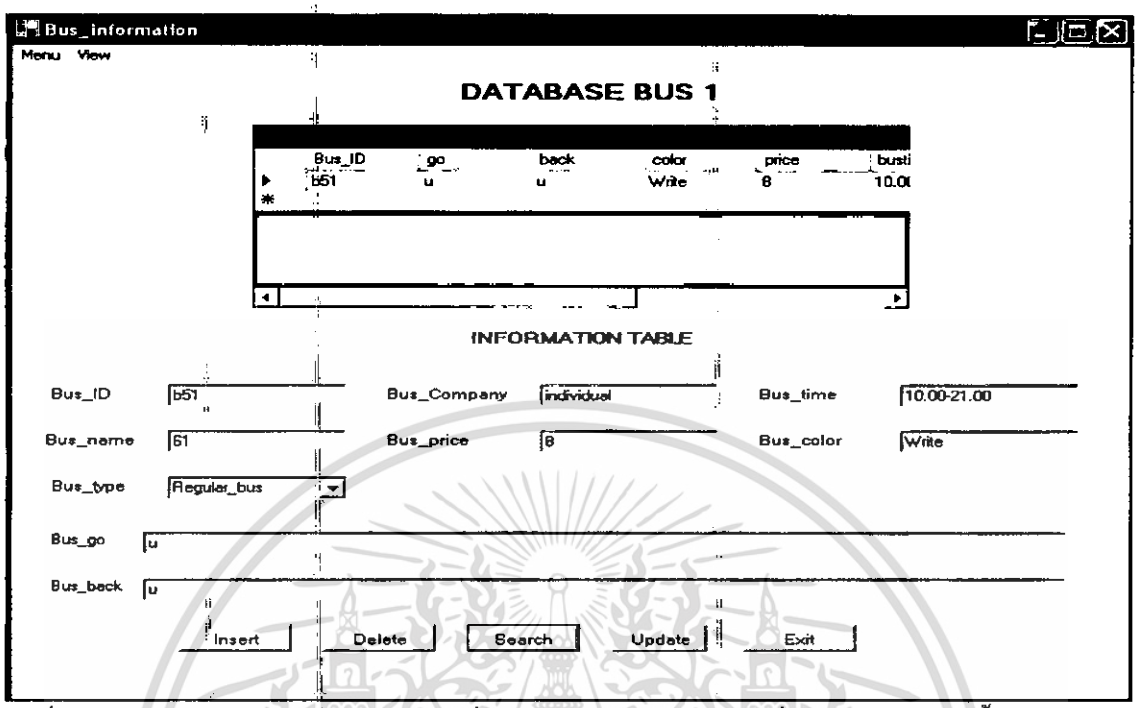
รูปที่ 4.8 หลังจากกดปุ่ม Delete แล้ว Record ที่ Place_ID เป็น P90 ก็จะถูกลบไปทั้ง Record

4.1.1.3 การ Search ข้อมูล

1. ตาราง Information

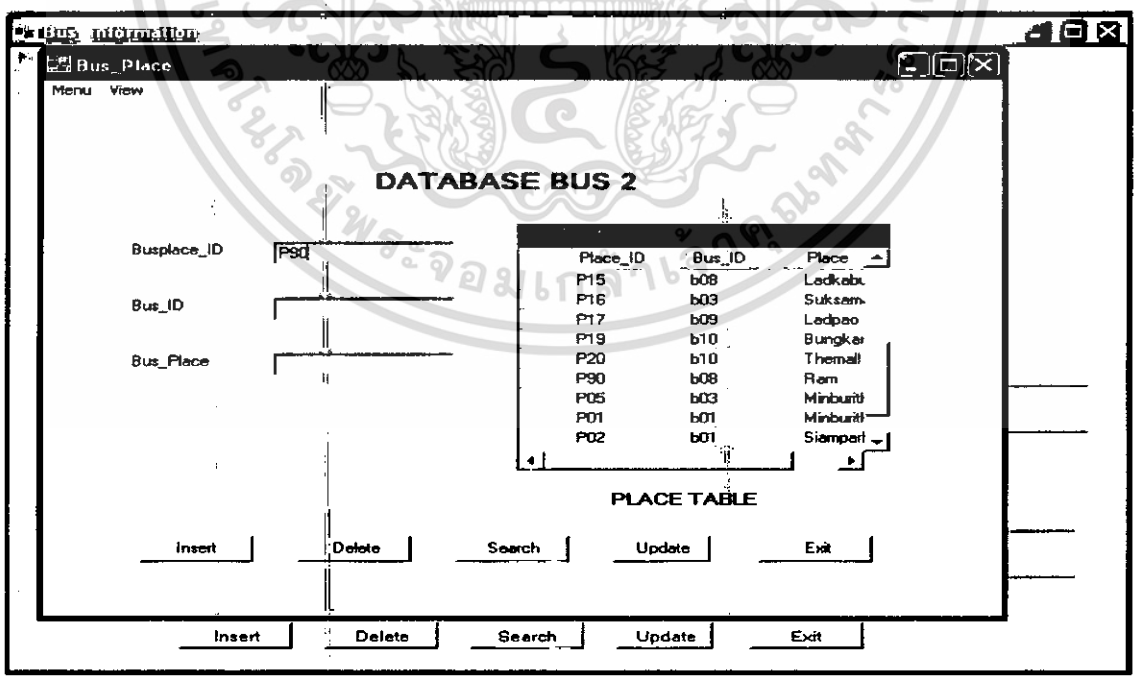


รูปที่ 4.9 การ Search จะใส่เฉพาะค่า Bus_ID หรือจะใส่ทั้งหมดก็ได้แล้วกดปุ่ม Search

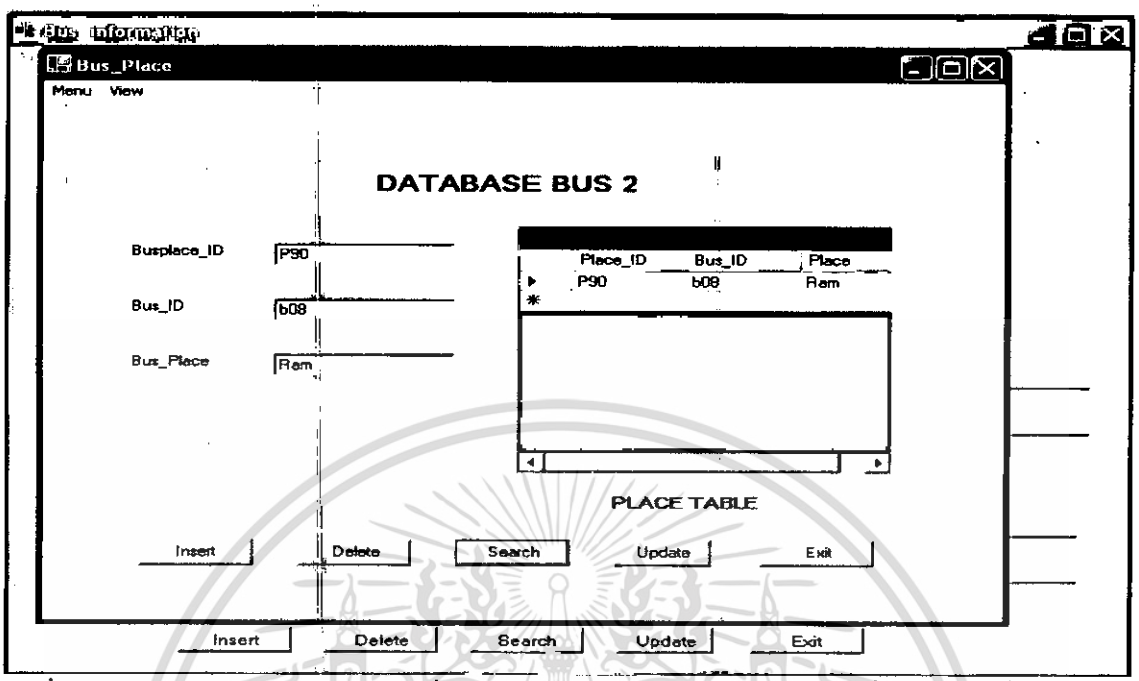


รูปที่ 4.10 หลังจากกดปุ่ม Search Record ที่ Bus_ID เป็น b51จะแสดงที่ช่อง Over View ทั้ง Record

2. ตาราง Place



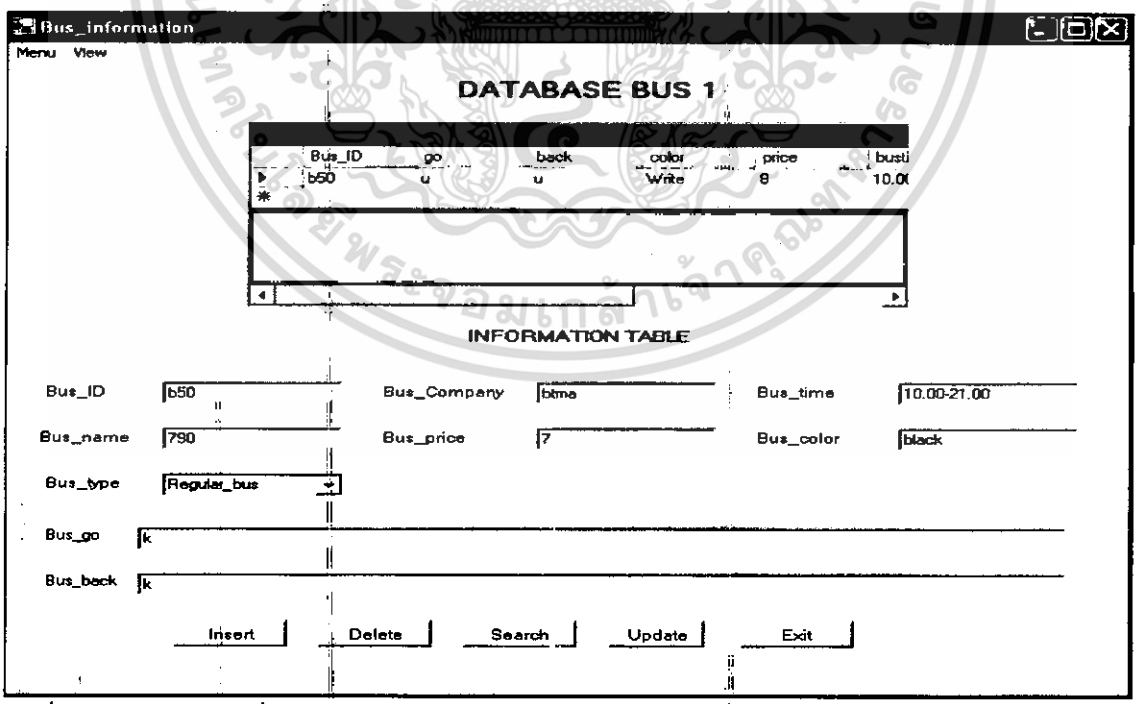
รูปที่ 4.11 ใส่เฉพาะค่า Place_ID ที่ช่อง Busplace_ID หรือจะใส่ทุกค่าก็ได้แล้วกดปุ่ม Search



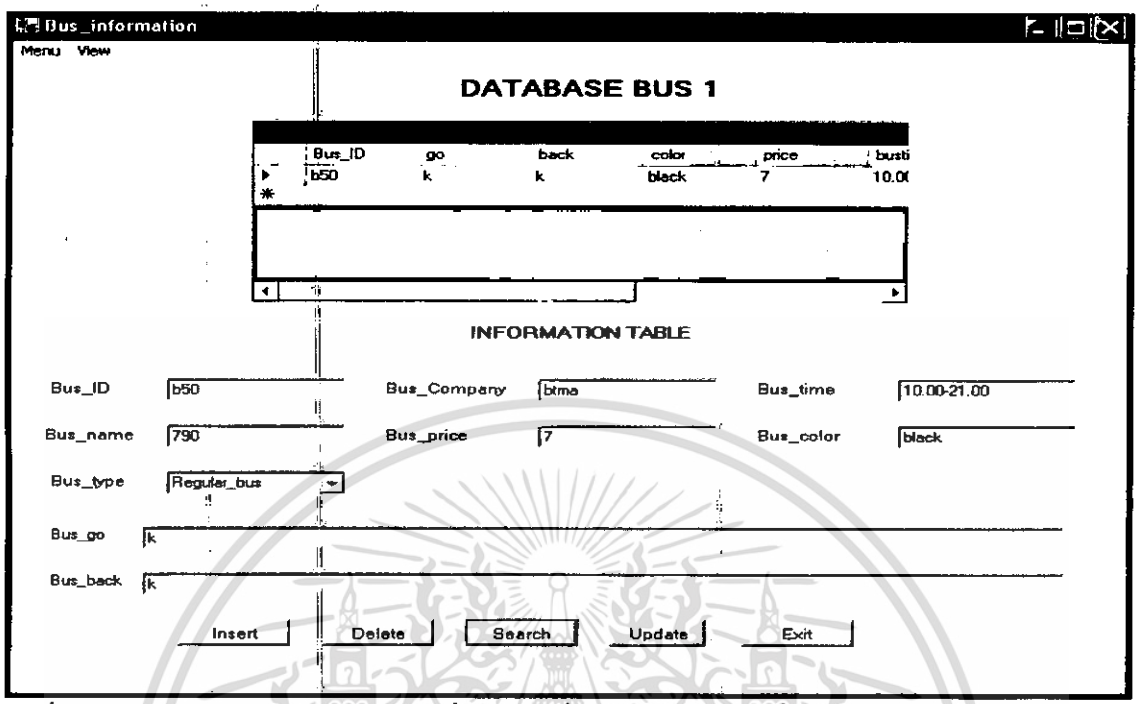
รูปที่ 4.12 หลังจากกดปุ่ม Search แล้ว ที่ช่อง Over View ก็จะถูกแสดง Record ที่ Place_ID ที่ P90

4.1.1.4 การ Update ข้อมูล

1. ตาราง Information

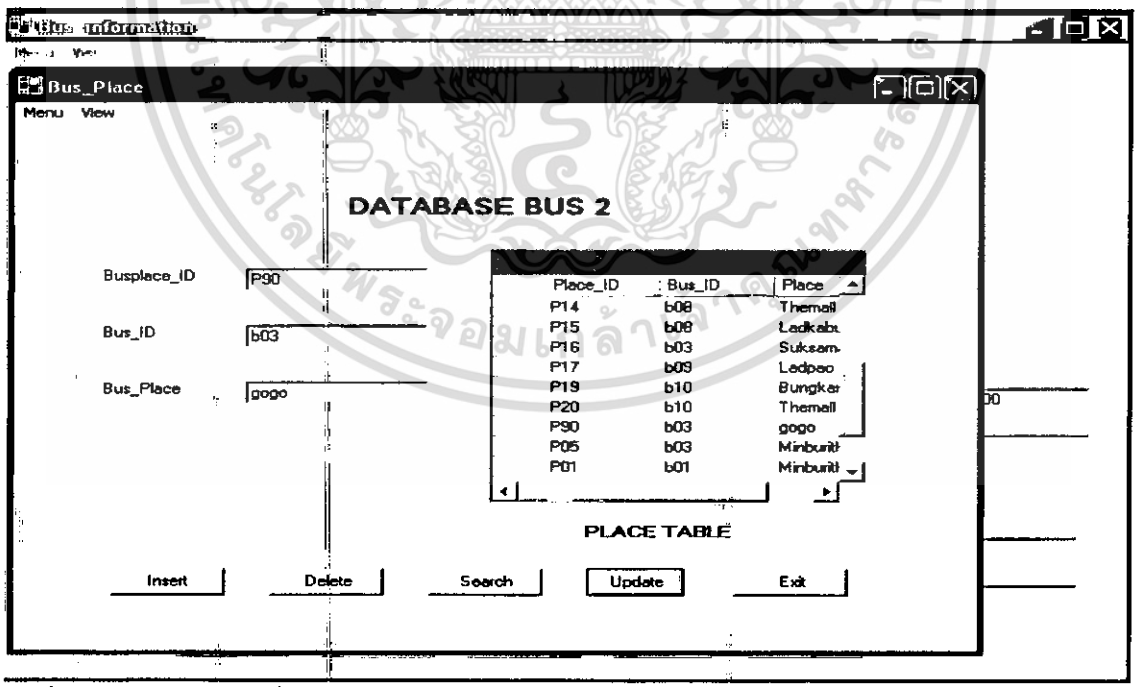


รูปที่ 4.13 ใส่ Bus_ID ที่ต้องการ Update ก่อนแล้วแก้ไข Column ที่ต้องการกดปุ่ม Update

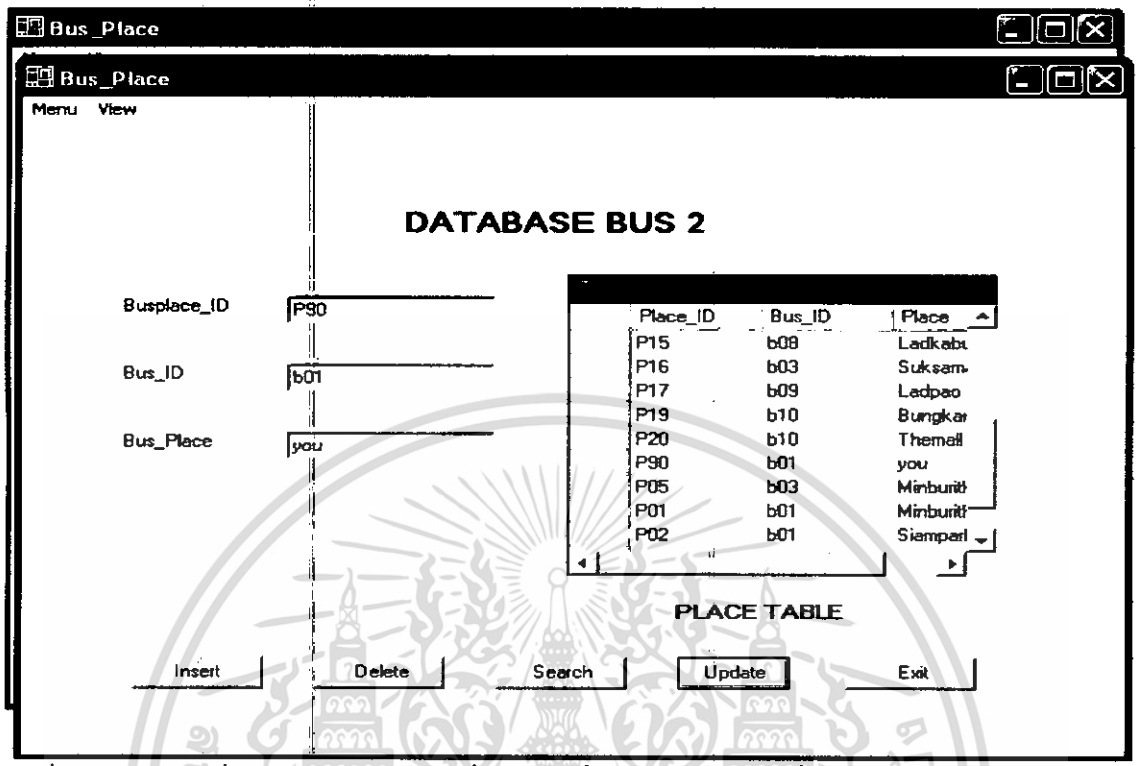


รูปที่ 4.14 หลังจากกดปุ่ม Update ข้อมูลที่ Record ที่ Update ก็จะถูกเปลี่ยน

2. ตาราง Place



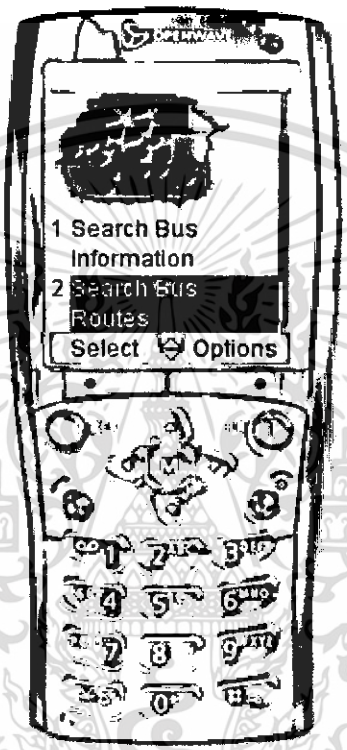
รูปที่ 4.15 ใส่ Place_ID ที่ต้องการ Update ที่ช่อง Busplace_ID ช่องที่เหลือกรอกค่าที่ต้องการ Update แล้วกดปุ่ม Update



รูปที่ 4.16 หลังจากที่กดปุ่ม Update ข้อมูลที่ Record ที่ Update ไปก็จะเปลี่ยน

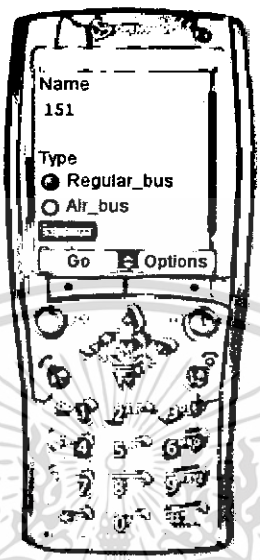
4.2 หน้าต่างแสดงเมนูในการค้นหา

โปรแกรมค้นหาเส้นทางรถประจำทางภายในกรุงเทพมหานคร จะมีการทำงานแยกออกด้วยกัน 2 ส่วน คือ การค้นหาเส้นทางรถเมล์ และ การค้นหาข้อมูลรายละเอียดของรถเมล์ โดยการค้นหาเส้นทางนั้นสามารถเลือกต้นทาง และ ปลายทางที่จะไปได้



รูปที่ 4.17 แสดง Main Menu จะมี 2 ฟังก์ชันการทำงาน

4.2.1 ลิ้งค์ ค้นหาข้อมูลรถเมล์ (Link Search Bus Information)

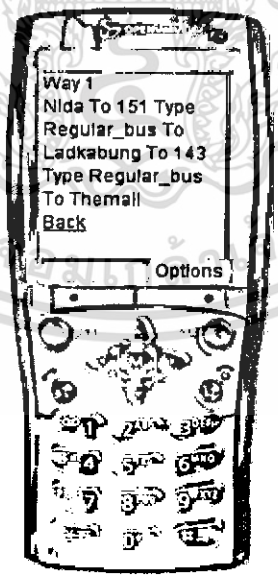


รูปที่ **4.18** เมื่อเลือก link ค้นหาข้อมูลรถเมล์ใส่ชื่อสายรถเมล์ที่ต้องการดูข้อมูล เลือกชนิดของรถ แล้วเลือกที่ Link Search

รูปที่ **4.19** แสดงผลฟังก์ชันค้นหาข้อมูลรถเมล์เมื่อจะต้องการกลับไปที่ Main Menu ให้เลือก Link Back ด้านล่างในหน้านี้

4.2.2 ลิ้งค์ ค้นหาเส้นทางรถเมล์ (Link Search Bus Routes)

รูปที่ 4.20 เมื่อเลือก Link ค้นหาเส้นทางรถเมล์พิมพ์ชื่อสถานที่ต้นทางที่ช่อง START และปลายทางที่ช่อง TARGET



รูปที่ 4.21 แสดงผลของฟังก์ชันค้นหาเส้นทางรถเมล์การต่อรถจากต้นทางจนถึงปลายทางและเมื่อจะต้องการกลับไปที่ Main Menu ให้เลือก Link Back ด้านล่างในหน้านี้

บทที่ 5
สรุปผลการดำเนินงาน

ในบทนี้จะกล่าวถึงประโยชน์ของโครงงาน ปัญหาและอุปสรรคในการดำเนินงาน พร้อมทั้งเสนอวิธีการแก้ไข เพื่อให้เกิดประโยชน์ต่อผู้ที่จะมาศึกษาโครงงานนี้ เพื่อที่จะเป็นแนวทางในการพัฒนาโครงงานอื่นต่อไป

5.1 บทวิจารณ์และสรุปผล

WAP Application นี้ได้ออกแบบ เพื่อที่จะได้ครอบคลุมระบบและผู้ใช้งานเป็นส่วนใหญ่ โดยจะออกแบบให้ใช้งานง่าย ผู้ที่เพิ่งจะเคยเข้าใช้งานเป็นครั้งแรกสามารถทำความเข้าใจได้ไม่ยาก จึงได้ออกแบบทั้ง Database และ WAP Card เป็น 2 ภาษา ทั้งนี้ สำหรับภาษาไทย จึงยังมีแต่ รุ่นที่ใช้ Windows Mobile เท่านั้นที่ใช้เนื่องจาก Font ส่วนใหญ่ได้ออกแบบมาใช้กับ ระบบปฏิบัติการ Windows เท่านั้น ส่วนระบบปฏิบัติการ Symbian ยังต้องรอการพัฒนา Font เพื่อที่จะได้รองรับ Database จาก Windows ได้ส่วนภาษาอังกฤษไม่มีปัญหาแต่อย่างใด

5.2 ประโยชน์ของโครงงาน

เมื่อโครงงานนี้เสร็จสิ้นลงเรียบร้อยแล้วจะได้ประโยชน์จากโครงงานดังนี้

- เพื่อพัฒนาระบบการค้นหาเส้นทางให้มีประสิทธิภาพยิ่งขึ้น
- เพื่อสร้างความสะดวกรวดเร็วในการตัดสินใจในการออกเดินทาง
- เพื่อศึกษาการวิเคราะห์และออกแบบฐานข้อมูลที่มีหลักการถูกต้อง ลดความซ้ำซ้อนของข้อมูลและการจัดเก็บอย่างมีระเบียบ

5.3 ปัญหาและอุปสรรคในการดำเนินงาน

- ใช้เวลาในการศึกษาและออกแบบฐานข้อมูล ตามทฤษฎีฐานข้อมูลเชิงสัมพันธ์
- ใช้เวลาในการเลือกใช้โครงสร้างการเชื่อมต่อของฐานข้อมูลกับ WAP Browser
- ระบบงานข่อนข้างใหญ่และมีความซับซ้อน จึงทำให้ยากแก่การวิเคราะห์และออกแบบระบบ
- ผู้จัดทำขาดประสบการณ์ในการพัฒนาระบบสารสนเทศ อีกทั้งโปรแกรมที่ใช้ในการพัฒนาใช้ระยะเวลาในการศึกษาค่อนข้างมากงานที่ทำจึงยังคงมีบางส่วนที่ยังไม่สมบูรณ์

5.4 แนวทางการพัฒนา

- ควรมีการปรับปรุงทางด้าน Font ที่ไม่รองรับ Symbian เพื่อที่จะรองรับภาษาไทยได้
- ควรมีการปรับปรุงในส่วนของการหาเส้นทาง ให้สามารถทำงานได้ดียิ่งขึ้น
- ควรมีการเพิ่มฟังชั่นการทำงาน เช่น การคำนวณอัตราค่าโดยสาร และ การต่อรถให้มีประสิทธิภาพ
- ควรให้ความสำคัญกับผู้ใช้เป็นหลักโดยให้ผู้ใช้นั้นตรวจสอบและวิจารณ์ระบบที่ได้พัฒนาขึ้นมา เพื่อให้สามารถปรับปรุงและแก้ไขได้ตามความต้องการส่วนใหญ่ของผู้ใช้

บรรณานุกรม

ธาริน สิทธิธรรมชารี. 2545. **สร้างระบบฐานข้อมูลอย่างมืออาชีพ Access 2003.** พิมพ์ครั้งที่1. กรุงเทพฯ : อินโฟเพรส.

น.ต. ไพศาล โมลิสกุลมงคล. 2542. **พัฒนา Web Database ด้วย PHP.** พิมพ์ครั้งที่ 1. กรุงเทพฯ : หจก. ไทยเจริญการพิมพ์.

กิตติ ภักดีวัฒนะกุล. 2542. **คัมภีร์ระบบฐานข้อมูล.** พิมพ์ครั้งที่ 2. กรุงเทพฯ : หจก. ไทยเจริญการพิมพ์

ร.ท.อนุโชต วุฒิพรพงษ์ และร.ต.พันธุ์เทพ แก้วมงคล. 2543. **สร้างWAP ด้วยWML Sclipt.** พิมพ์ครั้งที่ 1. กรุงเทพฯ : อินโฟเพรส.

สุรสิทธิ์ คิวประสิสพศักดิ์ และนันทนี แขวงโสภา. 2545. **Visual Basic.NET ฉบับสมบูรณ์.** กรุงเทพฯ : โปรวิชั่น.