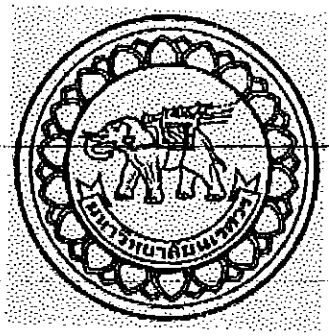


โปรแกรมการจัดการฐานข้อมูลฝ่ายเภสัชกรรม

Pharmacy Database Management Program

นายเกียรติศักดิ์ สิงห์ศิริเจริญกุล รหัส 41360579

นายพรเทพ เกิดจันทึก รหัส 41360397

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร

ปีการศึกษา 2544

ใบรับรองโครงงานวิจัย

หัวข้อโครงงาน	โปรแกรมการจัดการฐานข้อมูลฝ่ายเภสัชกรรม
ผู้ดำเนินโครงงาน	นายเกียรติศักดิ์ สิงห์ศิริเจริญกุล รหัส 41360579
	นายพรเทพ เกิดจันทึก รหัส 41360397
อาจารย์ที่ปรึกษา	อ.ประทีป ตรีรณโอภาส
อาจารย์ที่ปรึกษาร่วม	อ.สิทธิโชค เชาวกุล
สาขา	วิศวกรรมคอมพิวเตอร์
ภาควิชา	วิศวกรรมไฟฟ้าและคอมพิวเตอร์
ปีการศึกษา	2544

.....

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร อนุมัติให้โครงการนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร วิศวกรรมศาสตรบัณฑิต สาขาวิศวกรรมคอมพิวเตอร์

คณะกรรมการสอบโครงการ

.....ประธานกรรมการ
(อาจารย์สิทธิโชค เชาวกุล)

.....กรรมการ
(อาจารย์วัชวีร์ พืชพันธ์)

.....กรรมการ
(อาจารย์ประทีป ตรีรณโอภาส)

.....กรรมการ
(อาจารย์ศิริพร เดชะศิลารักษ์)

หัวข้อโครงงาน	โปรแกรมการจัดการฐานข้อมูลฝ่ายเภสัชกรรม
ผู้ดำเนินโครงงาน	นายเกียรติศักดิ์ สิงห์ศิริเจริญกุล รหัส 41360579
	นายพรเทพ เกิดจันทึก รหัส 41360397
อาจารย์ที่ปรึกษา	อ.ประทีป ตรีรณโอภาส
อาจารย์ที่ปรึกษาร่วม	อ.สิทธิโชค เชาวกุล
สาขา	วิศวกรรมคอมพิวเตอร์
ภาควิชา	วิศวกรรมไฟฟ้าและคอมพิวเตอร์
ปีการศึกษา	2544

บทคัดย่อ

โครงงานนี้เป็นการศึกษาเกี่ยวกับโปรแกรมการจัดการฐานข้อมูลฝ่ายเภสัชกรรมซึ่งเป็นโปรแกรมที่สามารถให้บริการกับฝ่ายเภสัชกรรมของโรงพยาบาล หรือตามร้านขายยาในปัจจุบันได้ ทั้งยังสามารถในการเช็คปริมาณของยา และราคาขาย พร้อมทั้งข้อมูลของบุคคลที่เกี่ยวข้องกับงานในส่วนนี้ได้ ซึ่งใช้ Visual Basic 6.0 ในการทำขึ้น และเขียนคำสั่งโดย SQL และการเก็บข้อมูล โดย Microsoft Access 97 ในการสร้างฐานข้อมูล

ผลที่ได้จากโครงงานนี้ คือ ได้โปรแกรมเกี่ยวกับฝ่ายเภสัชกรรม และสามารถใช้ได้กับร้านขายยา ซึ่งสะดวกและง่ายกว่าระบบที่ใช้ในปัจจุบันที่ยังเก็บข้อมูลบนกระดาษคือสามารถให้เจ้าของกิจการทราบข้อมูลเกี่ยวกับฝ่ายเภสัชกรรมของตนเองได้ตลอดเวลา

Project Title	Pharmacy Database Management Program
Name	Mr.Keittisak Singsiricharoenkul ID.41360579
	Mr.Porntep Gerdchuntug ID.41360397
Project Advisor	Mr.Prateep Treeronopas
Co – Project Advisor	Mr.SittiChok Chaowakul
Major	Computer Engineering
Department	Electrical and Computer Engineering
Academic Year	2001

ABSTRACT

This project is studied about the Pharmacy Database Management Program which is the program that can be serviced pharmacy department of hospital or Drug store. In the person . This program can check the quantity and the price of medicine include a data of person that concern which a job.This use Visual Basic 6.0 to make a program and write a direction by SQL and save a data by Microsoft Excess 97 to build a database .

The result of this project is has a program about pharmacy department and can use Drug store which comfortable and easy than the system that use in the present . This program,a manager can know a data about pharmacy department by themselve every time .

กิตติกรรมประกาศ

ผู้ทำโครงงาน ขอขอบพระคุณ ท่านอาจารย์ประทีป ศรีรณโอภาส และท่านอาจารย์วัชวีร์ พืชพันธ์ที่ให้คำแนะนำเกี่ยวกับการทำโปรแกรมและการค้นหาข้อมูลเพื่อมาทำโครงงานนี้ ตลอดจนได้สละเวลาให้คำแนะนำทั้งภาคทฤษฎีและภาคปฏิบัติ เอาใจใส่ให้ความช่วยเหลืออย่างดียิ่งตลอดเวลาในการทำโครงงานนี้ทั้งอาจารย์ในภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์ทุกท่าน ที่คอยดูแลและให้ความสะดวกในการเล่าเรียน และทำโครงงานนี้

นายเกียรติศักดิ์ สิงห์ศิริเจริญกุล

นายพรเทพ เกิดจันทึก

สารบัญ

หน้า

สารบัญ(ต่อ)

สารบัญตาราง

สารบัญตาราง(ต่อ)

สารบัญรูป

สารบัญรูป(ต่อ)

บทที่ 1

บทนำ

1.1 ที่มาและความสำคัญของโครงงาน

ปัจจุบันนี้ตามฝ่ายเภสัชกรรมของโรงพยาบาลต่างๆ และร้านขายยาทั่วไปยังมีการเก็บข้อมูลของยาต่างๆ และผู้ป่วยที่เข้าซื้อยา พร้อมทั้งคำแนะนำต่างๆ ของเภสัชกร และ การตัดสต็อกยา บนเอกสาร หรือว่าตามการคาดคะเนอยู่ เมื่อมียามากขึ้นและผู้ป่วยที่เข้ามากขึ้นจึงยากที่จะเก็บข้อมูลและคาดคะเนจำนวนของข้อมูลยาต่างๆที่มีในฝ่ายหรือร้านค้าได้ เมื่อมีวิทยาการด้านคอมพิวเตอร์เข้ามาซึ่งสามารถเก็บข้อมูลได้เป็นจำนวนมาก แถมราคาในปัจจุบันก็ถูกลง เราจึงนำเครื่องคอมพิวเตอร์ มาช่วยในการจัดเก็บ ค้นหาข้อมูลต่างๆ เพื่อความสะดวก รวดเร็ว แม่นยำยิ่งขึ้น สามารถจัดเก็บเอกสารที่มากให้เล็กลง และที่สำคัญคือรู้การตัดสต็อกยา ทำให้เรารู้ข้อมูลเกี่ยวกับยาและบุคลากรที่เกี่ยวข้องได้อย่างรวดเร็วและมีประสิทธิภาพ

1.2 วัตถุประสงค์โครงงาน

1.สามารถเก็บและรวบรวมข้อมูลด้านการแพทย์ เพื่อนำไปออกแบบฐานข้อมูลได้

2.สามารถออกแบบฐานข้อมูลเพื่อประโยชน์ทางการแพทย์ได้

3.สามารถศึกษาและเลือกเครื่องมือที่เหมาะสมเพื่อใช้ในการออกแบบโปรแกรมการจัดการฐานข้อมูลได้

4.สามารถออกแบบโปรแกรมการจัดการฐานข้อมูลได้

1.3 ขอบข่ายการทำงาน

1. ศึกษาเกี่ยวกับทฤษฎี และหลักการในสิ่งต่างๆเหล่านี้

1.1 ศึกษาทฤษฎีและหลักการของโปรแกรมการจัดการฐานข้อมูลและ SQL

1.2 ศึกษาหลักการในการในการ เปลี่ยนข้อมูลเพิ่มใหม่ได้อย่างรวดเร็วยิ่งขึ้น

1.3 ศึกษาทฤษฎีและหลักการการเชื่อมโยงฐานข้อมูลในรูปแบบของ VISUAL BASIC

2. รวบรวมข้อมูลทางการแพทย์ที่มีประโยชน์ในการเก็บประวัติคนป่วย ยารักษา โรคและการตัดสต็อกยา

3. ออกแบบโปรแกรมและโครงสร้างฐานข้อมูล

4. บรรจุและจัดการข้อมูลพร้อมกับเชื่อมฐานข้อมูล

5. ทดสอบโปรแกรมและฐานข้อมูล
6. วิเคราะห์การทดสอบพร้อมทั้งสรุปผลเพื่อพัฒนาในระดับสูงขึ้น
7. จัดทำเป็นรูปเล่ม

1.4 ขอบเขตของงานและตัวแปรที่ศึกษา

1. ขอบเขต
   1.1 สามารถรวบรวมประวัติคนป่วย อาการคนป่วยและฐานข้อมูลยาการตัดสต็อกยาได้
   1.2 แพทย์และเภสัชกรสามารถรู้เกี่ยวกับข้อมูล ประวัติ และยา ของคนป่วยได้
   1.3 แพทย์และเภสัชกรสามารถรู้ว่ายาแต่ละชนิดหมดหรือเหลือเท่าไหรได้
2. โปรแกรม
   2.1 Database Management ในแบบ SQL
   2.2 WINDOW 98
   2.3 VISUAL BASIC
   2.4 Microsoft Access 97

1.5 ผลที่คาดว่าจะได้รับ

1. ความเข้าใจในวิธีการออกแบบฐานข้อมูลเพื่อนำไปใช้งานจริงๆ
2. เข้าใจการทำงานของโปรแกรมจัดการระบบฐานข้อมูลทางการแพทย์ คือ
   - SQL,VISUAL BASIC
3. เข้าใจการทำงานของ WINDOW 98 และ Microsoft Access 97
4. ได้โปรแกรมการจัดการฐานข้อมูลฝ่ายเภสัชกรรม

1.6 ขั้นตอนดำเนินงาน

ตารางที่ 1.1 ขั้นตอนการดำเนินโครงงาน

กิจกรรม	เดือน - ปี					
	พ.ย. – ก.พ.	มี.ค.	เม.ย.-พ.ค.	มิ.ย.-ก.ค.	ส.ค.	ก.ย.
ศึกษาโปรแกรม	**********					
รวบรวมข้อมูล	**********					
ออกแบบโปรแกรม	**********					
บรรจุข้อมูล	**********					
ทดสอบโปรแกรม	**********					
วิเคราะห์สรุปผล	**********					
ทำรูปเล่ม	**********					

1.7 งบประมาณ

1. หมึกพิมพ์สี	1200	บาท
2. หนังสือการออกแบบฐานข้อมูล	500	บาท
3. ค่าถ่ายเอกสาร	150	บาท
4. อื่นๆ	150	บาท
รวมทั้งสิ้น	2000	บาท

บทที่2

หลักการและทฤษฎี

การพัฒนาระบบการจัดการฐานข้อมูลข้อมูลในปัจจุบัน เราจำเป็นที่ต้องรู้จักการทำงานของระบบฐานข้อมูล องค์ประกอบ และคำศัพท์ต่างๆที่มีใช้ในการออกแบบระบบฐานข้อมูลเพื่อจะได้ใช้เป็นความรู้ในการออกแบบ และพัฒนาระบบฐานข้อมูลได้ ทั้งยังสามารถมองถึงปัญหาของระบบและแก้ไข เพื่อให้ระบบมีประสิทธิภาพสูงสุดได้ ซึ่งจำแนกเนื้อหาได้ดังนี้

2.1 ระบบฐานข้อมูล [1]

2.1.1 ระบบแฟ้มข้อมูล

ในอดีตระบบการจัดเก็บข้อมูลยังมีการจัดเก็บเป็นแฟ้มเอกสาร เพื่อเป็นหมวดหมู่ ซึ่งอาจจะเป็นข้อมูลที่มีความสำคัญ หรือไม่มีความสำคัญ ปะปนกันไป เช่น การจัดหมวดหมู่ของยา มักจัดตามกลุ่มอาการที่ใช้รักษา และเมื่อหน่วยงาน หรือองค์กรต่างๆ มีขนาดใหญ่ขึ้น เอกสารและข้อมูลทางเภสัชกรรมจึงมีมากขึ้น เมื่อมีความจำเป็นต้องใช้หรือตรวจเช็คข้อมูล จึงยากและลำบากที่จะค้นหา จึงได้มีการนำเครื่องคอมพิวเตอร์เข้ามาช่วยในการทำงานของหน่วยงานหรือองค์กรนั้นๆ ซึ่งรูปแบบของข้อมูลจะอยู่ใน รูปแฟ้มข้อมูล เมื่อแฟ้มข้อมูลรวมกันอยู่มากๆ หรืออาจจะมีความสัมพันธ์กัน เราจึงนำมาไว้ด้วยกัน เรียกว่า " ระบบแฟ้มข้อมูล "

ซึ่งการที่เราจะนำข้อมูลที่อยู่ในระบบแฟ้มข้อมูลมาใช้นั้น เราจะต้องอาศัยผู้ที่มีความรู้เข้าช่วย ซึ่งเราเรียกว่า " โปรแกรมเมอร์ "เพื่ออ่านข้อมูลต่างๆ เข้ามาช่วยในการโปรเซสของข้อมูล ซึ่งใช้ภาษาต่างๆ เช่น ฟอร์แทน ,เบสิค เป็นต้น ซึ่งมีผลตามมาคือ ข้อจำกัดของการเรียกใช้ข้อมูล และความซับซ้อนในการพัฒนาโปรแกรม เช่นเมื่อต้องการเปลี่ยนแปลงแฟ้มข้อมูล โปรแกรมเมอร์ต้องเขียนโปรแกรมขึ้นมาใหม่

2.1.2 ปัญหาของระบบแฟ้มข้อมูล

เมื่อมีการจัดเก็บข้อมูลอยู่อย่างกระจัดกระจาย ของแต่ละหน่วยงาน หรือแต่ละองค์กร ซึ่งอาจจะทำให้เกิดปัญหาของ ข้อมูลซ้ำซ้อน เกิดการขัดแย้งของข้อมูลทำให้ผู้ใช้เข้าใจผิดเมื่อต้องการใช้ข้อมูลได้

2.1.3 ระบบฐานข้อมูล

จากการที่เกิดปัญหาจากหัวข้อที่ผ่านมา เราจึงต้องมีวิธีในการจัดเก็บข้อมูล ซึ่งเรานำมาเก็บในรูปแบบ " ฐานข้อมูล " นั้นเอง ซึ่งมีหลักสำคัญคือ การนำข้อมูลที่มีความสัมพันธ์กันมาไว้ในฐานข้อมูลเดียวกัน เพราะต้องอาศัยข้อมูลของกันและกันในฐานข้อมูลนั้น ๆ

2.1.4 องค์ประกอบของระบบฐานข้อมูล

1.ข้อมูล

2.ฮาร์ดแวร์

3.ซอฟแวร์

4.ผู้ใช้หรือผู้ออกแบบฐานข้อมูล

2.1.5 หน้าที่ของ DBMS

สำหรับรับหน้าที่ของโปรแกรม DBMS มีดังนี้

1. ทำหน้าที่แปลงคำสั่งที่ใช้จัดการกับข้อมูลภายในฐานข้อมูล ให้อยู่ในรูปแบบที่ฐานข้อมูลเข้าใจ
2. ทำหน้าที่ในการนำคำสั่งต่าง ๆ ซึ่งได้รับการแปลแล้ว ไปสั่งให้ฐานข้อมูลทำงาน เช่น การเรียกใช้ข้อมูล (Retrieve) การจัดเก็บข้อมูล (Updata) การลบข้อมูล (Delete) การเพิ่มข้อมูล(Add) เป็นต้น
3. ทำหน้าที่ป้องกันความเสียหายที่จะเกิดกับข้อมูลภายในฐานข้อมูล โดยจะคอยตรวจสอบว่าคำสั่งใดที่สามารถทำงานได้ และคำสั่งใดที่ไม่สามารถทำงานได้
4. ทำหน้าที่รักษาความสัมพันธ์ของข้อมูลภายในฐานข้อมูลให้มีความถูกต้องอยู่เสมอ
5. ทำหน้าที่เก็บรายละเอียดต่าง ๆ ที่เกี่ยวข้องกับข้อมูลภายในฐานข้อมูลไว้ใน Data Dictionary ซึ่งรายละเอียดเหล่านี้จึงมักจะถูกเรียกว่า "ข้อมูลของข้อมูล" (Metadata)

2.1.6 ประโยชน์ของฐานข้อมูล

1. สามารถลดความซ้ำซ้อนของการเก็บข้อมูลได้
2. สามารถหลีกเลี่ยงความขัดแย้งข้อมูลได้
3. ในแต่ละหน่วยงานสามารถใช้ข้อมูลเดียวกันได้
4. สามารถรักษาความถูกต้องให้กับข้อมูลได้
5. สามารถกำหนดระดับความปลอดภัยให้กับข้อมูลได้
6. ทำให้ข้อมูลเป็นอิสระจากโปรแกรมคือเมื่อต้องการแก้ไขข้อมูลจะไม่ส่งผลกระทบต่อตัวโปรแกรมเลย

2.2 สถาปัตยกรรมของระบบฐานข้อมูล [1]

เป็นการอธิบายถึง รูปแบบและโครงสร้างของข้อมูล โดยทั่วไปตามแนวความคิด ไม่ขึ้นกับโครงสร้างของฐานข้อมูลนั้น สถาปัตยกรรมที่นิยมใช้กันคือ

2.2.1 สถาปัตยกรรมANSI/SPARC แบ่งเป็น 3 ระดับคือ

1. ระดับ Internal ซึ่งจะเกี่ยวข้องกับระดับกายภาพ ในการจัดเก็บข้อมูลในฐานข้อมูลมากที่สุด ซึ่งจะเกี่ยวกับการจัดเก็บข้อมูล
2. ระดับ External ซึ่งจะเกี่ยวกับผู้ใช้มากที่สุด จะพูดถึงมุมมองและความคิดของผู้ใช้ในแต่ละคน
3. ระดับ Conceptual จะกล่าวถึง โครงสร้างข้อมูลในระดับแนวคิดแสดงออกมาในรูปแบบ ของภาพของโครงสร้างแทนโครงสร้างทางกายภาพ

2.2.2 Mapping

ในสถาปัตยกรรม ANSI/SPARC นี้มุมมองที่มีต่อข้อมูลในสถาปัตยกรรมในระดับที่สูงกว่า สามารถที่จะถ่ายทอดมุมมองนั้นไปยังสถาปัตยกรรมในระดับที่ต่ำกว่าได้ เช่น การถ่ายทอดมุมมองที่มีต่อข้อมูลจากสถาปัตยกรรมในระดับ Conceptual ไปยังสถาปัตยกรรมในระดับ Internal เพื่อนำโครงสร้างของข้อมูลในระดับ Conceptual Schema ไปแปลงเป็นโครงสร้างของข้อมูลในระดับกายภาพเพื่อใช้ในการจัดเก็บข้อมูล เป็นต้น สำหรับการถ่ายทอดมุมมองจากสถาปัตยกรรมในระดับที่สูงกว่าไปยังระดับที่ต่ำกว่านี้ จะเรียกว่า "การทำ Mapping"ในการทำ Mapping ตามสถาปัตยกรรม ANSI/SPARC สามารถแบ่งออกได้เป็น 2 ลักษณะดังนี้

2.2.3 สถาปัตยกรรมของระบบฐานข้อมูลกับ Database Administrator

ดังที่ได้กล่าวมาแล้วในบทที่ 1 ว่า Database Administrator (DBA) เป็นบุคคลที่มีหน้าที่ในการกำหนดกลยุทธ์และกุศโลบายในการใช้ข้อมูลภายในฐานข้อมูล รวมทั้งจัดหาเทคนิคที่จำเป็นต่อกลยุทธ์และกุศโลบายที่กำหนดขึ้น ดังนั้นผู้ใช้ทำหน้าที่เป็น DBA จึงมีความเกี่ยวข้องกับสถาปัตยกรรมของระบบฐานข้อมูลดังนี้

1. เป็นผู้กำหนดโครงสร้างของข้อมูลในสถาปัตยกรรมในระดับ Conceptual เช่น การกำหนดว่า ข้อมูลใดบ้างที่ควรที่จะจัดเก็บไว้ในฐานข้อมูล และมีโครงสร้างเป็นอย่างไร

2. เป็นผู้กำหนดโครงสร้างของข้อมูลในสถาปัตยกรรมในระดับ Internal ซึ่งได้แก่ โครงสร้างทางกายภาพของข้อมูลที่ใช้ในการจัดเก็บ เช่น ขนาดของแต่ละ Field ประเภทของข้อมูล ฯลฯ เป็นต้น

3. เป็นผู้ที่ทำหน้าที่ตรวจสอบโครงสร้างข้อมูลที่กำหนดขึ้นว่าสามารถรับต่อมุมมองหรือความต้องการในการใช้ข้อมูลของผู้ใช้ในสถาปัตยกรรมในระดับ External หรือไม่

4. เป็นผู้กำหนดการรักษาความปลอดภัย และกฎที่ใช้ในการควบคุมความถูกต้องให้กับข้อมูล ซึ่งเป็น ส่วนหนึ่งของการกำหนดโครงสร้างของข้อมูลในสถาปัตยกรรมในระดับ Conceptual

5. เป็นผู้กำหนดวิธีในการสำรองข้อมูล (Data Backup) และการกู้ข้อมูลที่เสียหายกลับมาใช้งาน (Data Recovery)

6. เป็นผู้ควบคุมให้ระบบฐานข้อมูลมีประสิทธิภาพ และทันสมัยตามความต้องการที่เปลี่ยนไป

2.2.4 สถาปัตยกรรมของระบบฐานข้อมูลกับ Database Management System

Database Management System (DBMS) เป็นส่วนที่มีความสัมพันธ์กับสถาปัตยกรรมของระบบฐานข้อมูลเนื่องจาก DBMS เป็นโปรแกรมที่ทำหน้าที่รับคำสั่งต่าง ๆ ทั้งในกลุ่มคำสั่ง DML และกลุ่มคำสั่ง DDL ที่ผู้ใช้กำหนดในสถาปัตยกรรมในระดับ External ไปกระทำกับข้อมูลในฐานข้อมูล ที่มีโครงสร้างข้อมูลอยู่ในสถาปัตยกรรมในระดับ Internal มาแปลงให้อยู่ในรูปแบบของโครงสร้างข้อมูลในสถาปัตยกรรมในระดับ Conceptual และ External ตามลำดับ เพื่อนำมาแสดงผลต่อผู้ใช้

2.3 แบบจำลองของฐานข้อมูลเชิงสัมพันธ์(Relation Database Model) [1]

เป็นโครงสร้างในแบบ Relational พัฒนาจากแบบจำลองที่กล่าวถึงความสัมพันธ์ระหว่างข้อมูลที่เรียกว่า Relation Model ข้อมูลที่จัดเก็บในแบบนี้ จะเก็บเป็นส่วนย่อยๆที่เรียกว่า Table อยู่ในตารางที่มีความสัมพันธ์ของสดมภ์และแถว โดยข้อมูลแต่ละตารางจะเก็บเป็นเอกเทศ แต่สามารถเอามาสัมพันธ์กันได้ตามแนวความคิด

2.4 โครงสร้างของฐานข้อมูลแบบ Relational [1]

ใน Relation Model ได้นิยามต่างๆ คือ Relation, Attribute, Domain เพื่ออธิบายถึงข้อมูลใน Relation Model นี้

2.4.1 Relation

เป็นการจัดเก็บข้อมูลแบบตารางมีแถวและสดมภ์ ชื่อของแต่ละแถวเรียกว่า "Tuple"ส่วนชื่อของแต่ละสดมภ์เรียกว่า "Attribute" ดังตารางที่ 2.1

ตารางที่ 2.1 ตัวอย่าง Relation

EmpID	Name	Lastname	Sex	Salary	DepID
001	เกียรติศักดิ์	สิงห์ศิริเจริญกุล	M	10,000	02
002	พรเทพ	เกิดจันทึก	M	10,200	03

2.4.1.1 คุณสมบัติของ Relation

คุณสมบัติของ Tuple และ Attribute ของแต่ละ Relation จะประกอบด้วย

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

ตารางที่ 2.2 ตัวอย่าง Relation

POPULAR

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

2. ด้วยเหตุผลเช่นเดียวกับข้อที่ 1 ลำดับที่ของสมาชิกภายในเซตใด ๆ จะไม่มีผลต่อเซตนั้น ดังนั้น ภายใน Relation จึงไม่มีการกำหนดลำดับที่ให้กับแต่ละ Tuple ใน Relation กล่าวคือ จะ

ไม่มีการกล่าวถึงคำว่า Tuple แรก หรือ Tuple สุดท้าย หรือ Tuple ลำดับที่ 5 หรือ Tuple ที่ผ่านมาใน Relation

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

4. ค่าในทุก Attribute ของ Relation จะต้องมีคุณสมบัติ Atomicity ซึ่งเป็นคุณสมบัติที่กำหนดให้ค่าของข้อมูลในแต่ละ Attribute ของ Relation จะต้องมีความหมายใดความหมายหนึ่งเพียงความหมายเดียวไม่ใช่กลุ่มของสิ่งใดสิ่งหนึ่ง หรือกล่าวอีกนัยหนึ่ง ข้อมูลในแต่ละ Attribute ของ Relation จะต้องไม่ใช่ข้อมูลในลักษณะ Repeating Group เช่น กรณีที่พนักงานสามารถสังกัดฝ่ายได้มากกว่า 1 ฝ่าย ข้อมูลใน Attribute "DeptID" ของแต่ละ Tuple ของ Relation "EMPLOYEE" ซึ่งใช้เก็บรหัสของฝ่ายที่พนักงานแต่ละคนสังกัด จะไม่สามารถจัดเก็บทุกรหัสของฝ่ายที่พนักงานคนนั้นสังกัดภายใน Tuple เดียวดังตัวอย่างข้อมูลต่อไปนี้ได้

ตารางที่ 2.3 ตัวอย่าง Relation

EMPLOYEE

EmpID	Name	Surname	Sex	Salary	DeptID
00001	สมบูรณ์	สุขมาก	M	10,000	01, 03
00002	สมเกียรติ	เจริญพร	M	8,000	02
00003	จันจิรา	แจ้งเกิด	F	12,000	03
00004	น้ำฝน	ม่วงทอง	F	9,500	01

จากรูปจะสังเกตเห็นว่า ข้อมูลที่เก็บอยู่ใน Attribute "DeptID" พนักงานที่ชื่อ "สมบูรณ์" มีค่ามากกว่า 1 ค่า ซึ่งจากรูปคือ รหัสฝ่าย "01" และ "03" ดังนั้น relation นี้จึงต้องมีการแยกข้อมูลในลักษณะ Repeating Group นั้นออกมาเป็น Tuple ใหม่ดังตัวอย่างข้อมูลต่อไปนี้

ตารางที่ 2.4 ตัวอย่าง Relation

EMPLOYEE

EmpID	Name	Surname	Sex	Salary	DeptID
00001	สมบูรณ์	สุขมาก	M	10,000	01
00001	สมบูรณ์	สุขมาก	M	10,000	03
00002	สมเกียรติ	เจริญพร	M	8,000	02
00003	จันจิรา	แจ้งเกิด	F	12,000	03
00004	น้ำฝน	ม่วงทอง	F	9,500	01

สำหรับคุณสมบัติของ Relation ในข้อนี้ ได้บังคับให้ทุก Relation มีรูปแบบการจัดเก็บข้อมูลที่เป็นไปตามรูปแบบแรก (First Normal Form) ตามที่กำหนดไว้ในการทำ Nomalization ให้กับ Relation ต่าง ๆ สำหรับรายละเอียดของการทำ Nomalization ให้กับ Relation ต่าง ๆ สำหรับรายละเอียดของการทำ Nomalization ให้กับ Relation จะกล่าวถึงในลำดับต่อไป

5. ชื่อของแต่ละ Attribute ใน Relation เดียวกัน จะต้องมีชื่อที่ไม่ซ้ำกัน

6. ค่าที่ปรากฏในแต่ละ Attribut ใน Relation เดียวกัน จะต้องให้แทนข้อมูลที่มีความหมายเดียวกัน เช่น Attribut "EmpID" ของ Relation "EMPLOYEE" จะต้องใช้เก็บข้อมูลรหัสพนักงานของแต่ละ Tuple เท่านั้น จะไม่สามารถใช้เก็บข้อมูลอื่น ๆ เช่น เพศ ได้ดังตัวอย่างข้อมูลต่อไปนี้

ตารางที่ 2.5 ตัวอย่าง Relation

EMPLOYEE

EmpID	Name	Surname	Sex	Salary	DeptID
00001	สมบูรณ์	สุขมาก	M	10,000	01
M	สมเกียรติ	เจริญพร	00002	8,000	02
00003	จันจิรา	แจ้งเกิด	F	12,000	03
00004	น้ำฝน	ม่วงทอง	F	9,500	01

2.4.1.2 ประเภทของ Relation

Relation สามารถแบ่งออกเป็นประเภทต่าง ๆ ได้ดังนี้

1. Named Relation

เป็น Relation ที่สร้างขึ้นด้วยคำสั่ง SQL ซึ่งอาจเป็น Relation จริงในฐานข้อมูล หรือเป็น เพียง Relation ที่สร้างขึ้นด้วยคำสั่งของ Query Language

2. Base Relation

เป็น Named Relation ในส่วนที่เป็น Relation จริงในฐานข้อมูล ซึ่งให้เก็บข้อมูลใน หน่วยความจำสำรองดังนั้นจึงเป็น Relation จริงที่เกิดขึ้นจากการออกแบบฐานข้อมูล

3. Derived Relation

เป็น Named Relation ในส่วนของ Relation ซึ่งได้มาจากการใช้เงื่อนไขประกอบกับ คำสั่งของ Query Language กับ Base Relation เช่น การนำเอา Relation "EMPLOYEE" ซึ่งเป็น Base Relation และมีข้อมูลดังตัวอย่างข้อมูลต่อไปนี้

ตารางที่ 2.6 ตัวอย่าง Relation

EMPLOYEE

EmpID	Name	Surname	Sex	Salary	DeptID
00001	สมบูรณ์	สุขมาก	M	10,000	01
00002	สมเกียรติ	เจริญพร	M	8,000	02
00003	จันจิรา	แจ้งเกิด	F	12,000	03

มาใช้เงื่อนไข "เฉพาะพนักงานชาย" ผลที่ได้ ได้แก่ Derived Relation ที่มีชื่อว่า "MALE" ที่มีข้อมูลเฉพาะพนักงานที่มีเพศชายดังตัวอย่างข้อมูลต่อไปนี้

ตารางที่ 2.7 ตัวอย่าง Relation

EMPLOYEE

EmpID	Name	Surname	Sex	Salary	DeptID
00001	สมบูรณ์	สุขมาก	M	10,000	01
00002	สมเกียรติ	เจริญพร	M	8,000	02

4. Expressible Relation

เป็น Relation ที่ได้มาจากการกระทำกับ Named Relation ด้วยเงื่อนไขทางด้านความสัมพันธ์ของข้อมูลระหว่าง Named Relation ที่ต้องการ ดังนั้นเมื่อนำเอาทุก Expressible Relation มาประกอบกัน ผลลัพธ์ที่ได้ จึงได้แก่ทุก ๆ Base Relation และ Derived Relation ที่ Expressible Relation นั้นใช้สร้างขึ้น

5. View

เป็น Derived Relation ประเภทหนึ่ง แต่จะเป็น Relation เสมือน (Virtual Relation) ที่ถูกสร้างขึ้นไว้ในฐานข้อมูล

6. Snapshot

เป็น Derived Relation ประเภทหนึ่งเช่นเดียวกับ View แต่จะต่างกันที่ Snapshot เป็น Relation ในฐานข้อมูล ที่สามารถอ่านข้อมูลได้เพียงอย่างเดียว และสามารถกำหนดเวลาในการปรับปรุงค่าของข้อมูลใน Snapahot ได้ เช่น ทุกวัน ทุกสัปดาห์ เป็นต้น

7. Query Result

เป็น Relation ชั่วคราว ซึ่งเกิดจากการใช้ประโยคคำสั่งของ Query Language กับ Relation ในฐานข้อมูลข้อมูลใน Relation ประเภทนี้ จะเกิดขึ้นก็ต่อเมื่อมีการเรียกใช้เท่านั้น และจะหายไปเมื่อเลิกใช้งาน

8. Immediate Result

เป็น Relation ชั่วคราวที่เกิดขึ้นในขณะทำการประมวลผลประโยคคำสั่งของ Query Language ที่มีความซับซ้อน เช่น คำสั่งต่อไปนี้

```
((S JOIN SP) WHERE P# = 'P2') [S#, CITY]
```

ในการประมวลผล จะเริ่มจากคำสั่งในวงเล็บก่อน ซึ่งได้แก่ คำสั่ง "S" และ "SP" ก่อนที่จะนำไปใช้ในขั้นตอนต่อไปซึ่ง Relation ที่เกิดขึ้นชั่วคราวนี้ จะถูกเรียกว่า "Intermediate Result"

9. Stored Relation

เป็น Expressible Relation ที่สามารถจัดเก็บค่าของข้อมูลได้ ดังนั้น ในบางครั้งจึงอาจกล่าวว่า Relation ประเภทนี้เป็น Base Relation

2.4.2 Key

Key คือ Attribute หรือ ชุดของ Attribute ที่ทำให้ข้อมูลในแต่ละ Tuple ใน Relation ไม่ซ้ำกัน เช่น Attribute EmpID ในตารางในข้อ 4.1 Key แบ่งเป็นดังนี้

1. Primary Key เป็น Key ที่ใช้ตรวจสอบการซ้ำกันของข้อมูล ระหว่างข้อมูลที่ป้อนหรือข้อมูลใหม่ ให้กับ Relation เช่น Attribute EmpID

ตารางที่ 2.8 ตัวอย่าง Primary Key

EmpID	Name	Lastname	Sex	Salary	DepID
001	เกียรติศักดิ์	สิงห์ศิริเจริญกุล	M	10,000	02
002	พรเทพ	เกิดจันทึก	M	10,200	03

2. Foreign Key จะเป็น Attribute ใด Attribute หนึ่ง ใน Relation ที่ใช้อ้างอิงไปยัง Attribute ที่ทำหน้าที่เป็น Primary Key ของอีกตารางหนึ่ง ใน " DepID "จะเป็น Foreign Key

ตารางที่ 2.9 ตัวอย่าง Foreign Key

EmpID	Name	Lastname	Sex	Salary	DepID
001	เกียรติศักดิ์	สิงห์ศิริเจริญกุล	M	10,000	02
002	พรเทพ	เกิดจันทึก	M	10,200	03

DeptID	Depname
02	การเงิน
03	พัสดุ

2.4.3 NULL

ข้อมูลที่จัดเก็บอยู่ในฐานข้อมูล บ่อยครั้งที่ปรากฏข้อมูลที่ถูกจัดเก็บไม่ครบถ้วน ซึ่งอาจเกิดจากการกรอกข้อมูลไม่ครบถ้วน หรือการจัดเก็บข้อมูลที่ไม่ดีพอ เช่น ข้อมูลวันเดือนปีเกิดของประชากร ซึ่งบางคนไม่สามารถระบุได้หรืออาจระบุได้เพียงปี เป็นต้น แต่เนื่องจากทุก Attribute ในฐานข้อมูล

แบบ Relational จะต้องมีค่า ดังนั้นเพื่อแก้ปัญหาดังกล่าว จึงได้มีการกำหนดค่าสำหรับข้อมูลที่ไม่สามารถระบุค่าได้ขึ้นมา ซึ่งเรียกว่า Null ดังรูป

ตารางที่2.10 ตัวอย่าง Candinate Key

Name	Sex	Birth_Date
สมบูรณ์ สุดขยัน	ชาย	1/8/2507
หฤทัย แสนประเสริฐ	หญิง	11/5/2511
แดง ดำดี	ชาย	Null
ปานทิพย์ สุดซึ้ง	หญิง	2499

Null จะมีค่าที่แตกต่างจากช่องว่างหรือจำนวนศูนย์ เนื่องจาก Null จะไม่ใช่ค่าที่ปรากฏอยู่จริงในโลกของความเป็นจริง รวมทั้งเป็นค่าที่แทนข้อมูลที่ไม่สามารถระบุค่าได้ ส่วนช่องว่างหรือจำนวนศูนย์นั้น เป็นค่าที่ปรากฏอยู่จริง จึงถือว่าเป็นค่าของข้อมูลที่ระบุค่าได้

2.5 การออกแบบฐานข้อมูล [1]

ฐานข้อมูลเป็นส่วนที่สำคัญสำหรับการออกแบบระบบสารสนเทศ ที่ใช้คอมพิวเตอร์ในการประมวลผล เนื่องจากเป็นส่วนที่ใช้ในการจัดเก็บข้อมูลต่างๆ ที่ใช้เป็นอินพุทของระบบสารสนเทศ จึงให้ความสำคัญกับการออกแบบฐานข้อมูลเช่นเดียวกับการออกแบบส่วนของการประมวลผล

2.5.1 วงจรชีวิตของการพัฒนาระบบงานสารสนเทศ(System Development Life Cycle)

ในการพัฒนาระบบงานสารสนเทศ โดยทั่วไป จะดำเนินตามขั้นตอนต่าง ๆที่กำหนดไว้ใน System Development Life Cycle (SDLC) แต่เนื่องจาก SDLC มีอยู่ด้วยกันหลายวิธี (Methodology) ดังนั้น จำนวนและรายละเอียดของขั้นจอนต่าง ๆ จึงแตกต่างกันไปตาม Methodology ของ SDLC ที่นักพัฒนาระบบงานสารสนเทศเลือกใช้แต่อย่างไรก็ตาม ขั้นตอนต่าง ๆ ของแต่ละ Methodology ก็ไม่ได้แตกต่างกันอย่างสิ้นเชิง เนื่องจาก Methodology ของ SDLC ส่วนใหญ่ จะยึดแนวทางในการแก้ปัญหาของ Federick Taylor ที่เรียกว่า Scientific Management เป็นหลัก แต่เนื่องจากในหนังสือเล่มนี้จะกล่าวถึงการออกแบบฐานข้อมูล ดังนั้น ขั้นตอนต่าง ๆ ในการพัฒนาระบบงานสารสนเทศของ SDLC ที่กล่าวถึงในที่นี่ จึงเป็นขั้นตอนหลัก ๆ ที่พบอยู่ใน Methodology ต่าง ๆ ของ SDLC ซึ่งประกอบด้วยขั้นตอนต่าง ๆ ดังนี้

2.5.1.1 Feasibility Study เป็นขั้นตอนที่เกี่ยวข้องกับการประเมินต้นทุนของทางเลือกต่าง ๆ ของการพัฒนาระบบงานสารสนเทศ เพื่อพิจารณาเลือกทางในการพัฒนาระบบงานสารสนเทศที่มีความคุ้มค่ามากที่สุด

2.5.1.2 Requirement Collection and Analysis เป็นขั้นตอนในการจัดเก็บรวบรวมความต้องการต่าง ๆ จากผู้ใช้ (User's Requirement) มาวิเคราะห์เพื่อจำแนกถึงปัญหา และความต้องการออกเป็นกลุ่ม เพื่อใช้กำหนดขอบเขต ให้กับระบบงานสารสนเทศที่จะพัฒนาขึ้น

2.5.1.3 Design เป็นขั้นตอนที่นำเอาปัญหา และความต้องการด้านต่าง ๆ ที่จำแนกไว้ในขั้นตอนที่ 2 มาใช้ในการออกแบบระบบงานสารสนเทศ

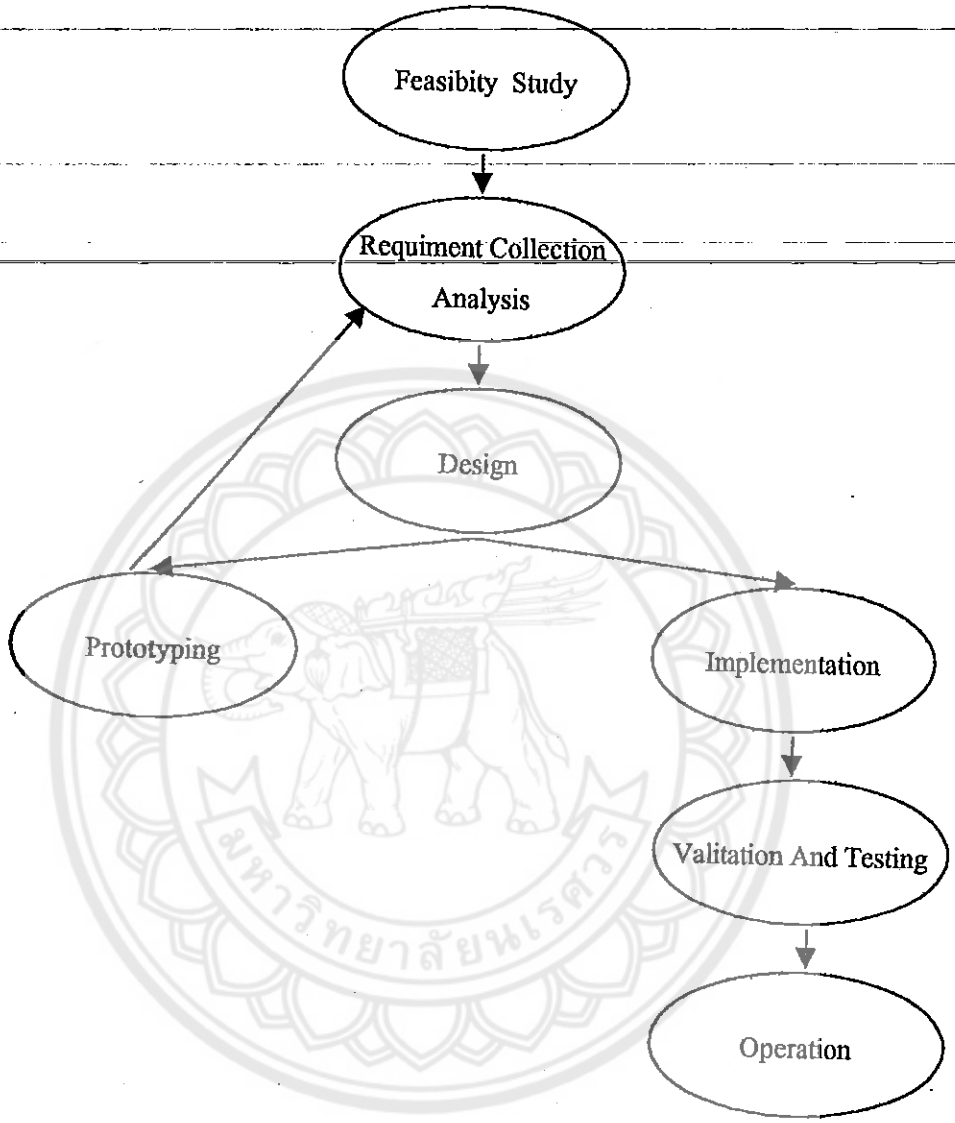
2.5.1.4 Prototyping เป็นขั้นตอนที่นำเอาส่วนต่าง ๆ ที่ได้ออกแบบไว้ในขั้นตอนที่ 3 มาพัฒนาต้นแบบของระบบงาน (Protoype) เพื่อนำไปทดลองใช้หาข้อผิดพลาดของระบบงานก่อนนำไปใช้งานจริง ในกรณีที่มีข้อผิดพลาดเกิดขึ้นรายละเอียดของข้อผิดพลาดต่าง ๆ จะถูกนำไปเป็นข้อมูลสำหรับขั้นตอนที่ 2 ได้ใหม่

2.5.1.5 Implementation เป็นขั้นตอนที่นำเอาระบบงานสารสนเทศที่พัฒนาเสร็จเรียบร้อยไปทดลองใช้งาน

2.5.1.6 Validation and Testing เป็นขั้นตอนของการตรวจสอบความถูกต้องของระบบงานสารสนเทศที่พัฒนาขึ้น

2.5.1.7 Operation เป็นขั้นตอนสุดท้าย ซึ่งแน่ใจแล้วว่า ระบบงานสารสนเทศที่พัฒนาขึ้นสามารถทำงานได้อย่างถูกต้องจึงเริ่มนำข้อมูลต่าง ๆ มาใช้งานจริง

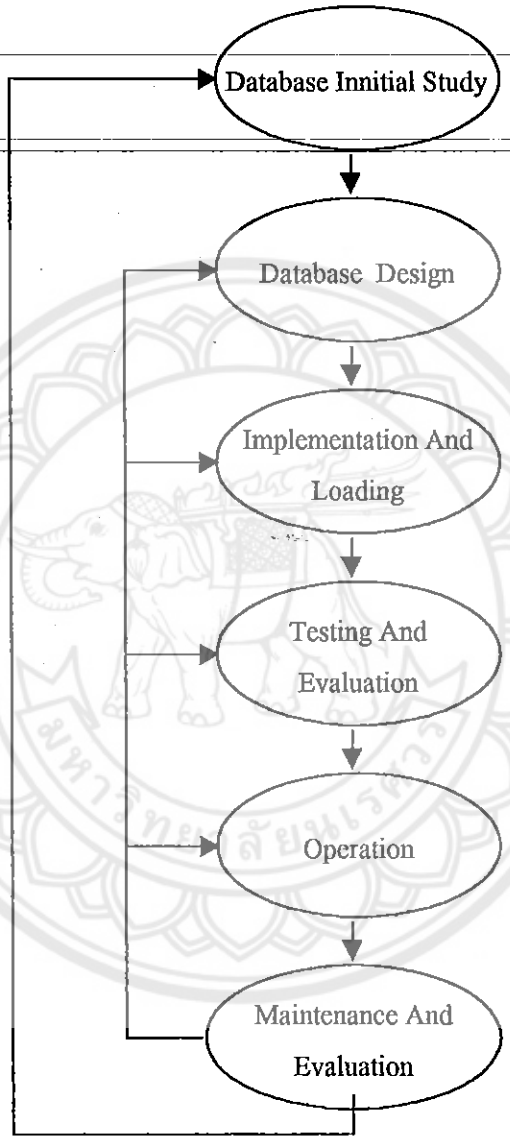
สำหรับทั้ง 7 ขั้นตอนนี้สามารถแสดงด้วยแผนภาพดังนี้



รูปที่ 2.1 วงจรชีวิตการพัฒนาระบบสารสนเทศ

2.5.2 วงจรชีวิตของการพัฒนาระบบฐานข้อมูล (Database Life Cycle)

วงจรชีวิตการพัฒนาระบบฐานข้อมูล เป็นขั้นที่กำหนดขึ้นเพื่อใช้เป็นแนวทางในการพัฒนาฐานข้อมูลขึ้นใช้งาน ประกอบด้วยขั้นตอนต่างๆ ดังนี้



รูปที่ 2.2 ขั้นตอนการออกแบบฐานข้อมูล

รายละเอียดที่ได้จากแต่ละขั้นตอนของการพัฒนาระบบฐานข้อมูลจะสามารถสะท้อนกลับไปขั้นตอนการทำงานก่อนหน้า ซึ่ง จะช่วยปรับปรุง และแก้ไขข้อผิดพลาดในการออกแบบฐานข้อมูล ในแต่ละขั้นตอนที่ผ่านมาได้เป็นอย่างดี

2.5.3 ขั้นตอนการออกแบบฐานข้อมูล

การออกแบบฐานข้อมูลแบ่งออกได้เป็น 3 ขั้นตอนดังนี้

2.5.3.1 การออกแบบฐานข้อมูลในระดับ Conceptual การออกแบบฐานข้อมูลในระดับนี้ จะเป็นการกำหนดโครงร่างเริ่มต้น ที่มีจุดมุ่งหมายเพื่ออธิบายโครงสร้างหลัก ๆ ของข้อมูลภายในระบบฐานข้อมูลโดยไม่คำนึงถึงฐานข้อมูลที่นำมาใช้ การอกแบบใน ระดับนี้มีความสำคัญมาก เนื่องจากโครงสร้างที่ได้จากการออกแบบในขั้นตอนนี้จะถูกนำไปใช้ในขั้นตอนต่อไปนี้ โครง่ร่างหรือที่เรียกว่า Schema ที่ได้จากการออกแบบในขั้นตอนนี้เรียกว่า Conceptual Schema

2.5.3.2 การออกแบบในระดับ Logical การออกแบบในระดับนี้จะเป็นระดับที่ต่อเนื่องมาจากระดับ Conceptual กล่าวคือ การออกแบบในระดับนี้จะอาศัยโครงร่างที่จากการออกแบบในระดับ Conceptual มาปรับปรุงให้มีโครงสร้างที่เป็นไปตามโครงสร้างข้อมูลที่จะนำมาใช้งานโดยยังไม่คำนึงถึงผลิตภัณฑ์ทางด้านฐานข้อมูลที่จะนำมาใช้งาน การออกแบบในขั้นตอนนี้ต้องปรับปรุงโครงสร้างบางอยางใน Conceptual Schema ให้สอดคล้องกับฐานข้อมูลที่จะนำมาใช้งาน การออกแบบในขั้นตอนนี้จึงต้องมีการตรวจสอบความถูกต้องของโครง่ร่างที่ออกแบบขึ้นกับส่วนประมวลผลต่าง ๆ ที่ออกแบบไว้รวมทั้งจะต้องแปลงโครงร่างต่าง ๆ ให้อยู่ในรูป Relation

2.5.3.3 การออกแบบฐานข้อมูลในระดับ Physical การออกแบบในระดับนี้ จะเป็นขั้นตอนสุดท้ายของการออกแบบฐานข้อมูลในขั้นตอนนี้ จะเป็นการปรับปรุงโครงสร้างของโครงร่างที่ออกแบบเช่นเดียวกัน แต่การปรับปรุงโครงสร้างของการออกแบบฐานข้อมูลในขั้นตอนนี้ จะเป็นการนำเอาโครงร่างที่ได้จากการออกแบบในระดับ Logical มาปรับปรุงโครงสร้างให้เป็นไปตามโครงสร้างของผลิตภัณฑ์ทางด้านฐานข้อมูลที่จะนำมาใช้งาน ผลลัพธ์ที่ได้จากการออกแบบในระดับนี้คือโครงสร้างของระบบฐานข้อมูล ที่สามารถนำไปใช้ในการสร้างตัวฐานข้อมูลจริง

2.6 Data Flow Diagram (DFD) [5]

กรรมวิธีการวิเคราะห์ระบบอย่างมีโครงสร้างนั้น วิธีหนึ่งนิยมในทางปฏิบัติคือ การมองภาพรวมในรูปแบบการไหลของข้อมูล (Data Flow) โดยที่วิธีนี้จะช่วยให้นักวิเคราะห์สามารถแบ่งระบบเป็นระบบย่อยได้ง่ายขึ้นและสามารถตรวจสอบได้สะดวก

การนำเสนอระบบแบบการไหลของข้อมูลนั้นจะใช้สัญลักษณ์แทนการบรรยายการทำงานของระบบ ซึ่งลักษณะที่จะใช้จะเป็นรูปวงกลม สี่เหลี่ยมจัตุรัส สี่เหลี่ยามผืนผ้าปลายเปิด เส้นโค้ง ลูกศร โดยนำสัญลักษณ์เหล่านี้มาเชื่อมต่อ การแสดงการต่อเนื่องของข้อมูลและการประมวลผล

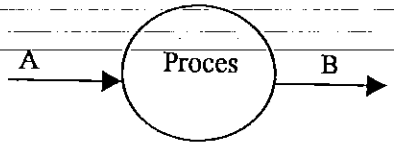
2.6.1 สัญลักษณ์ Data Flow Diagram (DFD)

ในแผนภาพของ DFD จะประกอบด้วยสัญลักษณ์ต่าง ๆ ดังนี้

1. ลูกศร ใช้แทนการไหลของข้อมูลพร้อมกับชื่อของข้อมูลนั้น ๆ จะต้องกำกับไว้ด้วย

รูปที่ 2.3 การแทนกระแสข้อมูลเป็นลูกศร

2. รูปวงกลม ใช้แทนกริยาการกระทำต่อข้อมูลที่ไหลเข้ามา โดยไม่คำนึงถึงว่าจะเป็นการกระทำโดยคนหรือคอมพิวเตอร์ก็ตาม จะได้มาซึ่งผลลัพธ์ที่จะไหลออกจากวงกลมภายในวงกลมจะระบุคำสั้น ๆ ที่จะใช้แทนการกระทำต่อข้อมูล

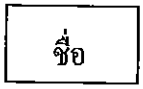


A = ข้อมูลเข้า

B = ข้อมูลออก

รูปที่ 2.4 การแทนกระแสข้อมูลเป็นลูกศร

3. รูปสี่เหลี่ยม ใช้แทนนามที่อยู่ภายนอกระบบซึ่งเป็นการกำเนิดของข้อมูลหรือสิ้นสุดของข้อมูล โดยมีชื่ออยู่ภายในสี่เหลี่ยม



รูปที่ 2.5 การแทนนามที่อยู่นอกระบบ

4. รูปสี่เหลี่ยมผืนผ้าปลายเปิด เป็นตัวแทนของแหล่งเก็บข้อมูลหรือ แฟ้มข้อมูลเสมือน เป็นตัวพักหรือช่วงขาดของการไหลของข้อมูลเพื่อนำไปเก็บเท่านั้น การกำหนดชื่อของแหล่งเก็บข้อมูลต้องอยู่ในสี่เหลี่ยม

รูปที่ 2.6 การแทนแหล่งเก็บข้อมูล

5. สัญลักษณ์เพิ่มเติม จะใช้เติมลงในสัญลักษณ์ที่กล่าวมาข้างต้นเพื่อแสดงความเป็นสิ่งเดียวกัน แต่จะถูกกล่าวหลาย ๆ ครั้งในแผนภาพ

รูปที่ 2.7 การแทนสัญลักษณ์เพิ่มเติม

2.6.2 ลำดับชั้นใน Data Flow Diagram

ในการเขียน DFD นักวิเคราะห์ระบบจะต้องมองระบบจากภาพรวมก่อนจากนั้นมองลึกเข้าสู่รายละเอียดข้างในของระบบยิ่งมองลึกมากเท่าใด ก็ยิ่งเห็นรายละเอียดของระบบย่อยได้มากขึ้นเท่านั้น

DFD ระดับที่ 0

ให้คือว่าระบบทั้งระบบเป็น PROCESS หรือวงกลมหนึ่งวง มีลูกศรแทน INPUT และ OUTPUT ตามที่จำเป็น

DFD ระดับที่ 1

ให้แตกวงกลมที่ลำดับ 0 ออกเป็นวงกลมย่อย 2-5 วงตามความเหมาะสม

DFD ระดับที่ 2

ให้แตกวงกลมที่ลำดับ 1 ออกเป็นวงกลมย่อยลงไปอีกเท่าที่จะทำได้

DFD ระดับที่ 3

- ถ้าจำเป็นก็ต้องตรวจดูว่า วงกลมใดในภาพลำดับที่ 2 ยังมีความซับซ้อนที่จำเป็นต้องแตกย่อยก็ต้องแตกย่อย ก็ต้องสร้างแผนภาพประกอบด้วยวงกลมย่อยแทนวงกลมนั้นให้ได้รายละเอียดสุดท้าย

2.6.3 ทำไมจึงต้องใช้ Data Flow Diagram

เหตุผลที่ต้องใช้ DFD เป็นแผลภาพของระบบก็เพราะ

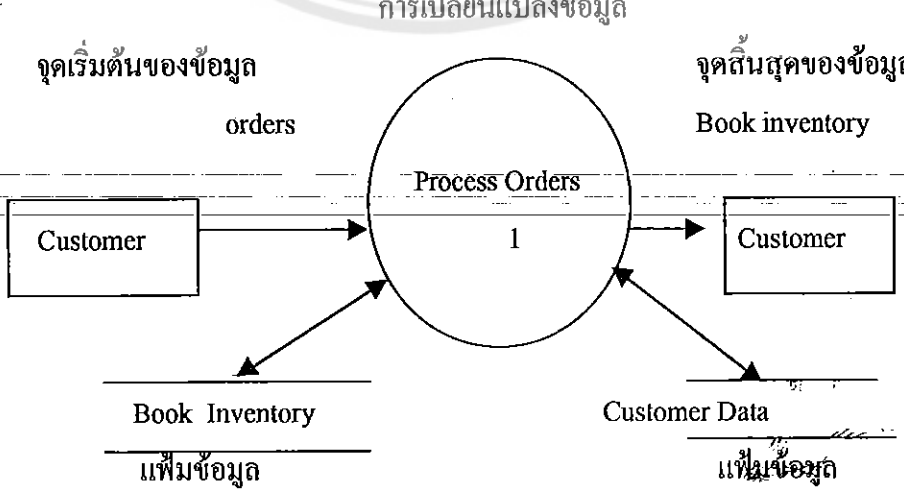
2.6.3.1 DFD ช่วยให้นักวิเคราะห์ระบบสามารถ

- สรุปข้อมูลที่เกี่ยวกับระบบ
- เข้าใจถึงปัญหาสำคัญของระบบและระบุส่วนของการทำงานที่ซ้ำซ้อน
- เข้าใจถึงความสำพันธ์ระหว่างส่วนต่าง ๆ ของระบบแบะการประกอบกันเป็นระบบ
- พัฒนาระบบได้อย่างมีประสิทธิภาพ

2.6.3.2 DFD เป็นเอกสารร่วมที่ช่วยให้นักวิเคราะห์ระบบและผู้ใช้สามารถข้าใจระบบและตรวจสอบความถูกต้องได้สองฝ่าย

2.6.3.3 ในการตรวจสอบเรื่องเวลาที่ใช้ในแต่ละขบวนการนั้น นักวิเคราะห์ สามารถใช้ DFD เป็นเครื่องมือที่ช่วยให้ทราบถึงขอบเขตในการพัฒนารูปแบบของระบบว่ามีทางที่จะเป็นไปได้อย่างไรบ้าง

2.6.4 ตัวอย่างการใช้ Data Flow Diagram



รูปที่ 2.8 การปฏิบัติงานของบริษัท หนังสือไทย

เป็นแผนภาพแสดงการสั่งซื้อหนังสือจากร้านหนังสือไทยลูกค้าในช่องสี่เหลี่ยมซ้ายมือสั่งสินค้ามายังวงกลม สิ่งที่จะต้องทำก็คือ ตัดสต๊อกของหนังสือที่ถูกสั่งออกไปพร้อมกับใบเก็บเงินไปยังลูกค้าด้านขวามือ โดยรูปแบบของสี่เหลี่ยมจัตุรัส ของลูกค้าทั้งด้านซ้ายและขวามีลักษณะเหมือนกัน จะหมายถึงสิ่งเดียวกัน

2.7 Entity-Relationship Model [1,4]

การออกแบบฐานข้อมูลใช้งานในระบบงานสารสนเทศใด ๆ ต้องอาศัยแบบจำลองของข้อมูลเพื่อนำเสนอรายละเอียดต่าง ๆ ที่เกี่ยวข้องกับข้อมูลในฐานข้อมูลที่ออกแบบ แบบจำลองของข้อมูลที่นิยมใช้ ได้แก่ Entity-Relationship Model

2.7.1 Semantic Model

แบบจำลองของข้อมูลในยุคแรก ๆ มีข้อจำกัดในการนำเสนอรายละเอียดที่เกี่ยวข้องกับข้อมูลในฐานข้อมูล กล่าวคือ จะมีการนำเสนอเฉพาะรายละเอียดทางด้านโครงสร้างของฐานข้อมูลเช่น คุณสมบัติ Atomicity ของข้อมูล กฎระเบียบต่าง ๆ ที่ใช้สำหรับควบคุมความถูกต้องของข้อมูล ฯลฯ เป็นต้น แต่ยังไม่มีการนำเสนอรายละเอียดทางด้านความหมาย (Semantic) ของข้อมูลภายในฐานข้อมูลนั้น ๆ เช่น จำนวน และน้ำหนักของสินค้าว่ามีความสัมพันธ์กันอย่างไร หรือ Domain ของข้อมูลต่าง ๆ สามารถมีค่าเป็นอะไรได้บ้าง หรือข้อมูลใดที่ทำหน้าที่เป็น Candidate Key หรือ Foreign Key ฯลฯ เป็นต้น ดังนั้น จึงมีการคิดค้นแบบจำลองของข้อมูลใหม่ขึ้นที่เรียกว่า "Semantic Model"

2.7.2 คำศัพท์ที่ใช้ใน Semantic Model

ใน Semantic Model ได้มีการนิยามคำขึ้นแทนข้อมูลในความหมายตาง ๆ ที่เรียกว่า Concept ดังนี้

2.7.2.1 Entity

Entity เปรียบเหมือนกับคำนามที่สามารถระบุได้ในความเป็นจริง อาจเป็นสิ่งที่จับต้องได้ เช่น นายไมเคิล หนังชื่อที่ชื่อ "ระบบฐานข้อมูล" หรือเป็นสิ่งที่อยู่ในรูปนามธรรมที่ไม่สามารถจับต้องได้ เช่น จำนวนวันลาพักร้อนของพนักงาน ซึ่งเมื่อนำแต่ละ Entity มารวมกันภายใต้คุณลักษณะใดลักษณะหนึ่งที่เหมือนกันแล้ว Entity เหล่านั้น จะถูกเรียกว่า "Entity Set" ดังตัวอย่างดังนี้

ตารางที่ 2.11 ตัวอย่าง Entity

Entity ของนาย ก.

นาย ก.	ไทย

Entity ของนาย ข.

นาย ข.	ไทย

Entity Set "ประชากรสัญชาติไทย"

นาย ก.	ไทย
นาย ข.	ไทย

2.7.2.2 Property หรือ Attribute

Property หรือ Attribute คือ ข้อมูลที่แสดงลักษณะและคุณสมบัติของ Entity เช่นProperty ของ Entity Set ที่ชื่อ "รถยนต์" ที่ประกอบด้วย หมายเลขทะเบียนรถยนต์ หมายเลขตัวถัง หมายเลขเครื่องยนต์ ยี่ห้อ รุ่น สี ดังรูป

ตารางที่ 2.12 ตัวอย่าง Property

Property

ENTITY	ทะเบียน	หมายเลขตัวถัง	หมายเลขเครื่องยนต์	ยี่ห้อ	รุ่น	สี
→	4ษ-2100	2AZ0012523	EE859222	TOYOTA	CORONA	ฟ้า
→	1ศ-1700	1 WE9944560	AS985112	HONDA	CIVIC	ม่วง
→	8ฮ-2545	6AA4456455	EZ123444	NISSAN	SUNNY	แดง

2.7.2.3 Identity

แต่ละ Entity ภายใต้ Entity Set เดียวกัน ถึงแม้ว่าจะต้องมี Property ที่เหมือนกันแต่อย่างไรก็ตาม จะต้องมี Property ใน Property หนึ่ง ซึ่งเป็นเอกลักษณ์เฉพาะของ Entity นั้นเช่น

Property "หมายเลขประชาชน" ของแต่ละ Entity ใน Entity Set "ประชาชน" ซึ่งจะไม่มีหมายเขใดที่ซ้ำกัน ดังตารางข้างล่าง

ตารางที่ 2.13 ตัวอย่าง Identity

Identity

หมายเลขบัตรประชาชน	ชื่อ	นามสกุล	เพศ	สัญชาติ	วันเดือนปีเกิด
123456789	แดง	สด	ชาย	ไทย	12/12/22
987654321	ดำ	ดี	หญิง	ไทย	11/11/22

สำหรับ Property ที่สามารถนำมากำหนดเป็นเอกลักษณ์เฉพาะให้กับแต่ละ Entity นี้จะเรียกว่า "Identity"

2.7.2.4 Relationship

ได้แก่ Entity Set ที่สร้างขึ้นจาก 2 Entity Set เดิมหรือมากกว่า เพื่อใช้แสดงความสัมพันธ์ระหว่างแต่ละ Entity ใน Entity Set เดิมเหล่านั้น ในการสร้าง Relationship อาจสร้างด้ายการนำเอาแต่ละ Entity ใน Entity Set เดิมเหล่านั้นมาเชื่อมโยงข้อมูลกันภายใต้ค่าของ Property ที่เหมือนกัน ซึ่งการสร้างความสัมพันธ์ในลักษณะนี้ Property ของ Relationship จะเกิดจากการนำเอา Property ของแต่ละ Entity Set มารวมกัน เช่น Relationship ที่ชื่อ "สังกัดคณะ" ซึ่งเกิดจากการที่ Entity Set "นักศึกษา" และ "คณะ" มี Property "รหัสคณะ" ที่เหมือนกัน ดังรูป

ตารางที่ 2.14 ตัวอย่าง Relationship

Entity Set "นักศึกษา"

รหัสนักศึกษา	ชื่อ - สกุล	เพศ	รหัสคณะ
380012	เอก โท	ชาย	02
381202	ตรี จัตวา	หญิง	01
380052	เอก ตรี	ชาย	03

Entity Set " คณะ"

รหัสคณะ	คณะ
01	วิศวกรรมศาสตร์
02	บริหารธุรกิจ

Relationship "สังกัดคณะ"

รหัสนักศึกษา	ชื่อ – สกุล	เพศ	รหัสคณะ	คณะ
380012	เอก โท	ชาย	02	บริหารธุรกิจ
381202	ตรี จัตวา	หญิง	01	วิศวกรรมศาสตร์
380052	เอก ตรี	ชาย	03	วิทยาศาสตร์

2.7.3 Entity-Relationship Model

Semantic Model ที่นิยมใช้มากที่สุด ได้แก่ Entity-Relationship Model หรือที่นิยมเรียกกันสั้นว่า "E-R Model" E-R Model นับเป็นแบบจำลองที่ครอบคลุมนิยามต่าง ๆ ที่กำหนดไว้ใน Semantic Model เนื่องจากมีรูปแบบที่ใช้แทนทุก ๆ แนวความคิดที่กำหนดไว้ Semantic Model ซึ่งได้แก่ Entity, Property, Relation และ Subtypeสำหรับแผนภาพที่สร้างขึ้นโดยใช้รูปภาพต่าง ๆ ภายใน E-R Model เพื่อแสดงความเป็นจริงต่างๆ ของข้อมูลในฐานข้อมูล จะเรียกว่า Entity-Relationship Diagram หรือที่นิยมเรียกกันสั้น ๆ ว่า แผนภาพ E-R (E-R Diagram)

2.7.3.1 Entity

ได้แก่ Entity Set ต่าง ๆ ที่นิยามไว้ใน Semantic Model แต่ใน E-R Model จะเรียก Entity Set ว่า Entity แทน สำหรับ Entity ใน E-R Model จะแบ่งออกเป็น 3 ประเภทดังนี้

1. **Regular Entity** หรือบางครั้งเรียกว่า Strong Entity ได้แก่ Entity ส่วนใหญ่ที่ปรากฏอยู่ในระบบฐานข้อมูล ซึ่งสมาชิกภายใน Entity ประเภทนี้ สามารถมีคุณสมบัติ Identity ได้ด้วยตัวมันเอง เช่น Entity "EMPLOYEE" ซึ่งสมาชิกภายใน Entity นี้ได้แก่ ข้อมูลของพนักงานแต่ละคนในบริษัท ซึ่งประกอบด้วย Property ต่าง ๆ ดังนี้

ตารางที่ 2.15 ตัวอย่าง Regular Entity

Entity "EMPLOYEE"

EmpID	NAME	SEX	SALARY
00001	สมชาย นิลกลัด	M	8500
00002	สมถวิล กลั่นเจริญ	F	9000
00003	เจริญ ก้าวหน้า	M	12000

จากตัวอย่างข้อมูลของ Entity "EMPLOYEE" จะสังเกตเห็นว่า สมาชิก ซึ่งได้แก่ ข้อมูลในแต่ละแถวของตาราง สามารถมีคุณสมบัติ Identity ได้โดยอาศัยค่าของ Property "EmpID" เนื่องจากพนักงานแต่ละคน จะมีหมายเลขประจำตัวพนักงานที่ไม่ซ้ำกัน สำหรับรูปภาพที่ใช้แทน Entity ประเภทนี้ ได้แก่ รูปสี่เหลี่ยมผืนผ้า โดยมีชื่อของ Entity นั้นอยู่ภายใน ดังรูป

รูปที่ 2.9 Regular Entity

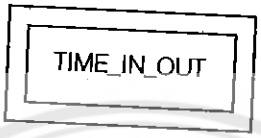
2. **Weak Entity** เป็น Entity ที่มีลักษณะตรงข้ามกับ Regular Entity กล่าวคือ สมาชิกของ Entity ประเภทนี้จะสามารถมีคุณสมบัติ Identity ได้ จะต้องอาศัย Property ใด Property หนึ่งของ Regular Entity มาประกอบกับ Property ของตัวมันเอง เช่น Entity "TIME_IN_OUT" ซึ่งสมาชิกของ Entity นี้ได้แก่ เวลาเข้าออกของพนักงานแต่ละคนในแต่ละวัน ซึ่งประกอบไปด้วย Property ต่าง ๆ ดังนี้

ตารางที่ 2.16 ตัวอย่าง Weak Entity

Entity "TIME_IN_OUT"

EmpID	Date	Time_In	Time_Out
00001	15/9/41	7.30	17.13
00002	15/9/41	8.00	18.00
00003	15/9/41	7.45	17.49
00001	16/9/41	8.00	18.00
00002	16/9/41	7.45	17.49
00003	16/9/41	7.30	17.13

จากตัวอย่างข้อมูลของ Entity "TIME_IN_OUT" จะสังเกตเห็นว่า แต่ละสมาชิกของ Entity "TIME_IN_OUT" จะสามารถมีคุณสมบัติ Identity ได้ จะต้องอาศัยค่าของ Property "EmpID" ซึ่งเป็น Property ของ Entity "TIME_IN_OUT" มาประกอบกับ Property " Date " ซึ่งเป็น Property ของ ตนเอง จึงจะทำให้สมาชิกของแต่ละ Entity "TIME_IN_OUT" มีคุณสมบัติ Identity สำหรับรูปภาพที่ใช้แทน Entity ประเภทนี้ ได้แก่ รูปสี่เหลี่ยมผืนผ้าสองรูปซ้อนกัน โดยมีชื่อของ Entity นั้นอยู่ใน ดังรูป

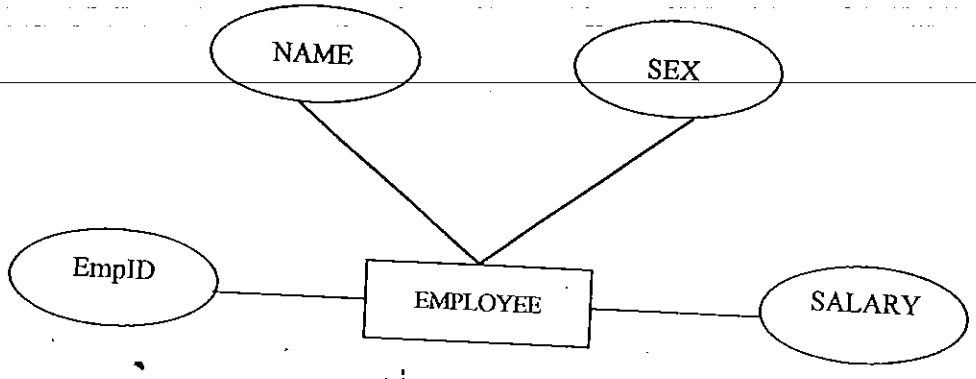


รูปที่ 2.10 Weak Entity

2.7.3.2 Property

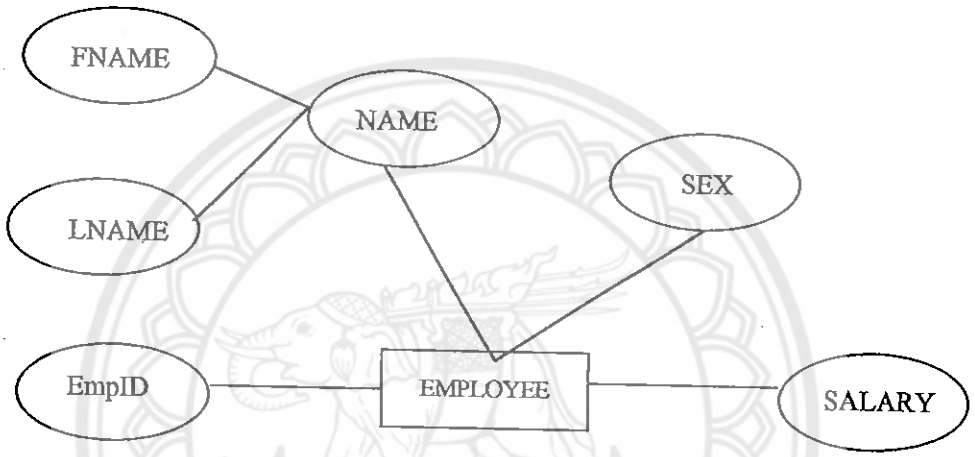
ได้แก่ Property ต่างๆ ของ Entity หรือ Relationship ที่นิยามไว้ใน Semantic Model เช่น Property "EmpID", "NAME", "SEX" และ "SALARY" ของ Entity "EMPLOYEE" เป็นต้นสำหรับ Property ใน E-R Model จะสารมารถแบ่งย่อยได้ดังนี้

I) Simple Property ได้แก่ Property ที่ค่าภายใน Property นั้นไม่สามารถแบ่งย่อยได้อีก ซึ่งได้แก่ Property ของ Entity โดยทั่วไป เช่น Property "SEX" ของ Entity "EMPLOYEE" ที่ใช้เก็บเพศของพนักงานแต่ละคน หรือ Property "SALARY" ของ Entity "EMPLOYEE" ที่ใช้เก็บเงินเดือนของพนักงานแต่ละคน ซึ่งค่าทั้ง 2 Property นี้ ไม่สามารถแบ่งออกเป็นค่าย่อยได้อีกสำหรับรูปภาพที่ใช้แทน Property ประเภทนี้ ได้แก่ วงรีที่มีเส้นเชื่อมต่อไปยัง Entity ที่เป็นเจ้าของ Property นั้น โดยมีชื่อของ Property นั้นอยู่ภายใน เช่น Property ต่างๆ ของ Entity "EMPLOYEE" ดังรูป



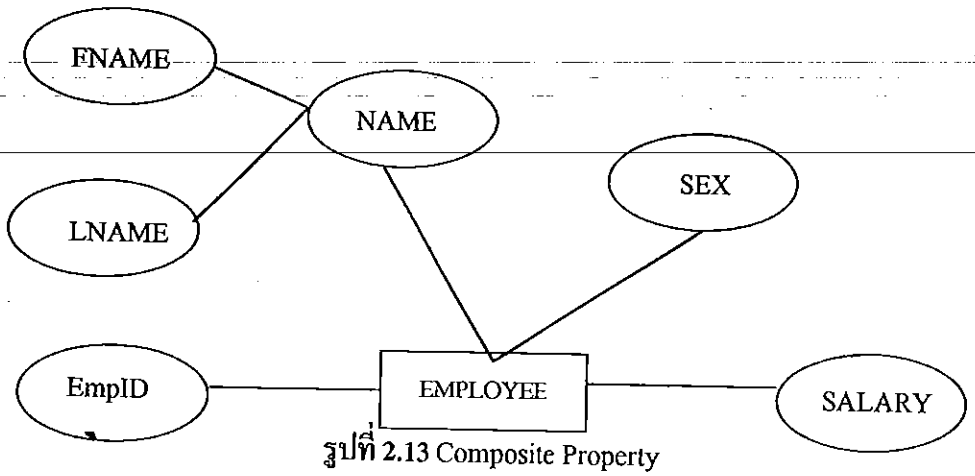
รูปที่ 2.11 Weak Entity

2) Composite Property เป็น Property ที่มีลักษณะตรงข้ามกับ Simple Property กล่าวคือ จะเป็น Property ที่ค่าภายใน Property นั้น ยังสามารถแยกเป็น Property ย่อยได้อีก เช่น Property "NAME" ของ Entity "EMPLOYEE" ที่สามารถแบ่งย่อยออกได้เป็น Property "FNAME" ที่ใช้เก็บชื่อและ Property "SNAME" ที่ใช้เก็บนามสกุล เป็นต้น สำหรับรูปภาพที่ใช้แทน Property ประเภทนี้จะใช้วงรีเดียวกับ Simple Property แต่จะเป็นวงรีที่เชื่อมกับวงรีของ Simple Property ที่เป็นเจ้าของ Composite Property นั้น เช่น Property "FNAME" และ "SNAME ของ Entity "EMPLOYEE" ดังรูป



รูปที่ 2.12 Composite Property

3) Key เป็น Property หรือกลุ่มของ Property ที่มีค่าในแต่ละสมาชิกของ Entity ไม่ซ้ำกันซึ่งถูกนำมาใช้กำหนดคุณสมบัติ Identity ให้กับ Entity เช่น Property "EmpID" ของ Entity "EMPLOYEE" ซึ่งใช้แทนรหัสประจำตัวพนักงาน สำหรับรูปภาพที่ใช้แทน Key ของ Entity จำใช้รูปวงรีเช่นเดียวกับ Property แต่จะมีเส้นขีดอยู่ให้ Property ที่เป็น Key ดังรูป



รูปที่ 2.13 Composite Property

4) Single-valued Property เป็น Property ที่มีค่าของข้อมูลภายใต้ค่าของ Property ใด Property เพียงค่าเดียว เช่น Property "SALARY" ที่ใช้เก็บเงินเดือนของพนักงาน ซึ่งพนักงานแต่ละคนสามารถมีเงินเดือน ได้เพียงค่าเดียว ดังตัวอย่างข้อมูลต่อไปนี้

ตารางที่ 2.17 ตัวอย่าง Single-valued Property

Entity **"EMPLOYEE"**

EmpID	NAME	SEX	SALARY
0001	สมชาย นิลกลัด	M	8500
0002	สมถวิล กลั่นเจริญ	F	9000
0003	เจริญ ก้าวหน้า	M	12000
0004	ชุติมา สกุลดี	F	10000
0005	นิวัติ เหล่าสุวรรณ	M	25000

สำหรับรูปภาพที่ใช้แทน Property ประเภทนี้ จะใช้รูปภาพเช่นเดียวกับ Simple Property

5) Multi-valued Property เป็น Property ที่มีลักษณะตรงข้ามกับ Property แบบ Single-valued กล่าวคือ เป็น Property ที่มีค่าของข้อมูลได้หลายค่าภายใต้ค่าของ Property ใด Property หนึ่งเช่น Property "DEGREE" ที่ใช้ระบุระดับการศึกษาของพนักงานแต่ละคน ซึ่งพนักงานแต่ละคน จะมีระดับการศึกษาได้หลายระดับ ดังตัวอย่างข้อมูลต่อไปนี้

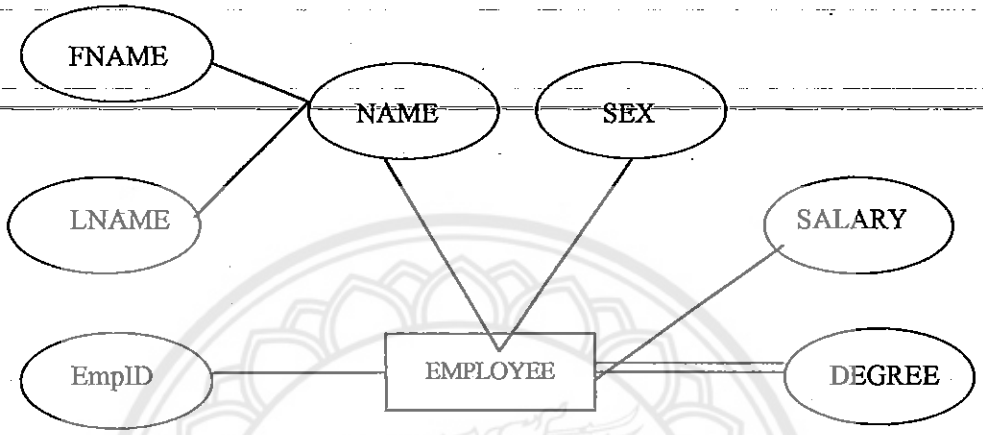
ตารางที่ 2.18 ตัวอย่าง Multi-valued Property

Entity **"EMPLOYEE"**

EmpID	NAME	SEX	SALARY	DEGREE	SCHOOL/UNIVERSITY
0001	สมชาย นิลกลัด	M	8500	0	ร.ร ขอนแก่นวิทยา
0002	สมถวิล กลั่นเจริญ	F	9000	0	ร.ร อ่างทองวิทยา
0003	เจริญ ก้าวหน้า	M	12000	0	ร.ร วัดสุทธิวราราม
				1	ม. กรุงเทพ
0004	ชุติมา สกุลดี	F	10000	0	ร.ร นรีเวชวิทยา
0005	นิวัติ เหล่าสุวรรณ	M	25000	0	ร.ร กรุงเทพคริสเตียน

จากตัวอย่างข้อมูล จะสังเกตเห็นว่า Property "DEGREE" ภายใต้พนักงานแต่ละคนสามารถมีค่าของข้อมูลได้มากกว่า 1 ค่า เช่น Property "DEGREE" ของพนักงานที่ชื่อ "เจริญ ก้าวหน้า" จะมีอยู่

ด้วยกัน 2 ค่าคือ 0 และ 1 เนื่องจากพนักงานคนนี้ จบการศึกษาระดับปริญญาตรี จึงมีข้อมูลของทั้งระดับมัธยมศึกษา และปริญญาตรี หรือ Property "DEGREE" ของพนักงานที่ชื่อ "นิวัติ เหล่าสุวรรณ" จะมีอยู่ด้วยกัน 3 ค่า คือ 0, 1 และ 2 เนื่องจากพนักงานคนนี้จบการศึกษาระดับปริญญาโท เป็นต้น Entity หรือ Relation จะใช้เส้น 2 เส้นแทน ดังรูป



รูปที่ 2.14 Multi-valued Property

2.7.3.3 Relationship

ได้แก่ Relationship ที่นิยามไว้ใน Semantic Model สำหรับรูปภาพที่ใช้แทน Relationship ใน E-R Model จะได้แก่ รูปสี่เหลี่ยมข้าวหลามตัด ที่มีชื่อของ Relationship นั้นอยู่ภายใน สำหรับรูปภาพของ Relationship นี้ไม่สามารถปรากฏอยู่เดี่ยวๆ ได้ แต่จะต้องปรากฏอยู่คู่กับ Entity เสมอ

Relationship โดยทั่วไป จะกำหนดขึ้นจาก Entity ที่มี Property ร่วมกันเช่น Entity "EMPLOYEE" และ "Department" ซึ่งสามารถแสดงด้วยตัวอย่างข้อมูลได้ดังนี้

ตารางที่ 2.19 ตัวอย่าง Relationship

Entity **"EMPLOYEE"**

EmpID	NAME	SEX	SALARY	DEP_ID
0001	สมชาย นิลกลัด	M	8500	01
0002	สมถวิล กลั่นเจริญ	F	9000	03
0003	เจริญ ก้าวหน้า	M	12000	01
0004	ชุติมา สกุลดี	F	10000	02
0005	นิวัติ เหล่าสุวรรณ	M	25000	02

Entity **"Department"**

DEP_ID	DEP_NAME
01	ธุรการ
02	บุคคล
03	การเงิน

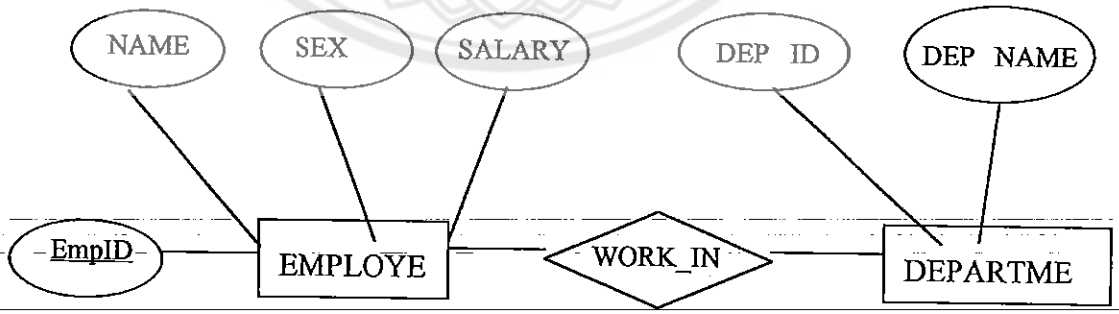
จากตัวอย่างข้อมูล จะสังเกตเห็นว่า ทั้ง 2 Entity มี Property ร่วมกันคือ Property "DEP_ID" นี้ได้ ซึ่งในที่นี้ได้แก่ Relationship เพื่อแสดงความสัมพันธ์ ซึ่งแสดงถึงฝ่ายที่พนักงานแต่ละคนสังกัด ดังนี้

ตาราง 2.19 (ต่อ) ตัวอย่าง Relationship

Relation **"WORK_IN"**

EmpID	NAME	SEX	SALARY	DEP_ID	DEP_NAME
0001	สมชาย นิลกลัด	M	8500	01	ธุรการ
0002	สมถวิล กลั่นเจริญ	F	9000	03	การเงิน
0003	เจริญ ก้าวหน้า	M	12000	01	ธุรการ
0004	ชุติมา สกุลดี	F	10000	02	บุคคล
0005	นิวัติ เหล่าสุวรรณ	M	25000	02	บุคคล

และสามารถแสดงด้วยรูปภาพได้ดังนี้



รูปที่ 2.15 Relationship

2.7.3.4 Cardinality Ratio

สมาชิกใน Entity ที่เกี่ยวข้องกับ Relation จะถูกเรียกว่า Participant ซึ่งจำนวนของ Participant นี้จะถูกเรียกว่า Degree ของ Relationship นั้น และจะถูกนำไปใช้กำหนดประเภทของ Relationship ที่เรียกว่า "Cardinality Ratio"

1. One-to-One Relationship เป็น Relationship ที่แต่ละ Participant ของ Entity หนึ่งจะมีความสัมพันธ์กับอีก Participant ของอีก Entity หนึ่งเพียง Participant เดียว ดังตัวอย่างข้อมูลต่อไปนี้

ตารางที่ 2.20 ตัวอย่าง Relationship แบบ One-to-One

Entity "CUSTOMER"

NAME	ADDRESS	ACCT_NO
แพง พลเมืองดี	111 บางพลัด กทม.	111111111
จิราพร สมตน	222 บางซื่อ กทม.	222222222
สุภาพร อุดมศิลป์	333 ปทุมวัน กทม.	333333333
กิตติ มั่นคง	444 บางบอน กทม.	444444444

Entity "ACCOUNT"

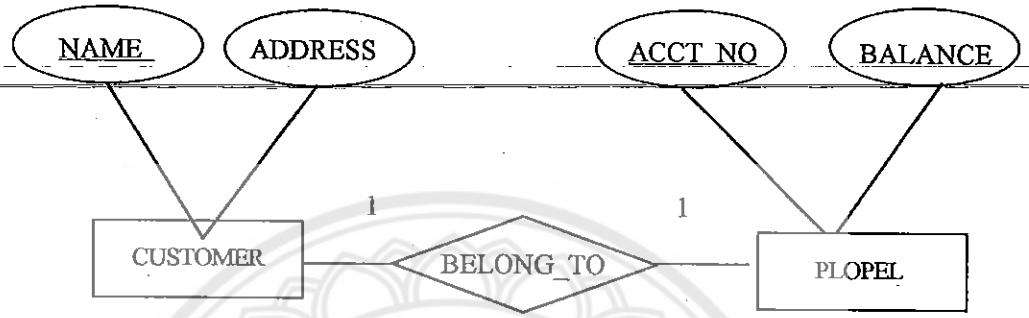
ACCT_NO	BALANCE
111111111	54000
222222222	12000
333333333	14000
444444444	100000

Relationship "BELONG_TO"

NAME	ADDRESS	ACCT_NO	ACCT_NO	BALANCE
แพง พลเมืองดี	111 บางพลัด กทม.	111111111	111111111	54000
จิราพร สมตน	222 บางซื่อ กทม.	222222222	222222222	12000
สุภาพร อุดมศิลป์	333 ปทุมวัน กทม.	333333333	333333333	14000
กิตติ มั่นคง	444 บางบอน กทม.	444444444	444444444	100000

จากตัวอย่างข้อมูลของ Relationship "BELONG_TO" จะสังเกตเห็นว่า ลูกค้า ซึ่งได้แก่ แต่ละรายการใน Entity "CUSTOMER" จะมีความสัมพันธ์กับรายการใน Entity "ACCOUNT" ได้เพียงรายการเดียว และเพียงรายการเดียว และในมุมกลับกัน แต่ละรายการใน Entity "ACCOUNT"

สำหรับรูปภาพที่ใช้แทนความสัมพันธ์นี้ จะใช้รูปภาพรายการเดียวกับ Relationship โดยทั่วไป ดังนั้น รูปภาพของความสัมพันธ์นี้ จึงมีลักษณะดังรูป



รูปที่ 2.16 Relationship แบบ One-to-One

2) One-to-Many Relationship เป็น Relationship ที่แต่ละ Participant ของ Entity หนึ่ง มีความสัมพันธ์กับอีก participant ของอีก Entity หนึ่งมากกว่า 1 Participant เช่น กรณีลูกค้าสามารถมีเงินฝากได้มากกว่า 1 บัญชี และแต่ละบัญชีเงินฝากจะต้องมีเจ้าของบัญชีเพียงคนเดียว ดังตัวอย่างข้อมูลต่อไปนี้

ตารางที่ 2.21 ตัวอย่าง Relationship แบบ One-to-Many

Entity "CUSTOMER"

NAME	ADDRESS	ACCT_NO
แพง พลเมืองดี	111 บางพลัด กทม.	111111111
แพง พลเมืองดี	111 บางพลัด กทม.	111111112
จิราพร สมตน	222 บางซื่อ กทม.	222222222
สุภาพร อุดมศิลป์	333 ปทุมวัน กทม.	333333333
กิตติ มั่นคง	444 บางบอน กทม.	444444444

Entity "ACCOUNT"

ACCT_NO	BALANCE
111111111	54000
222222222	12000
333333333	14000
444444444	100000
111111112	4000

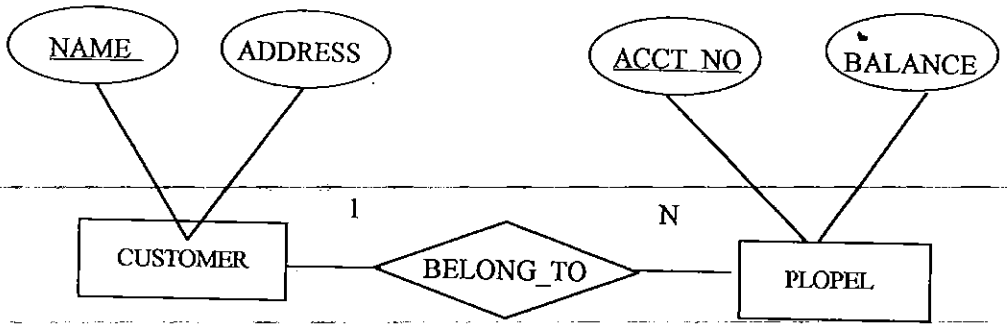
ตารางที่ 2.21 (ต่อ) ตัวอย่าง Relationship แบบ One-to-Many

Relationship "BELONG_TO"

NAME	ADDRESS	ACCT_NO	BALANCE
แพง พลเมืองดี	111 บางพลัด กทม.	111111111	54000
		111111112	4000
จิราพร สมตน	222 บางซื่อ กทม.	222222222	12000
สุภาพร อุดมศิลป์	333 ปทุมวัน กทม.	333333333	14000
กิตติ มั่นคง	444 บางบอน กทม.	444444444	100000

จากตัวอย่างข้อมูลของ Relationship "BELONG_TO" จะสังเกตเห็นว่า ลูกค้าที่ชื่อ "แพง พลเมืองดี" เป็นเจ้าของบัญชีเงินฝาก 2 บัญชี คือ บัญชีเงินฝากเลขที่ "11111111" และ "111111112" แต่ในมุมกลับกัน แต่ละบัญชี จะมีเจ้าของได้เพียงคนเดียว

สำหรับสัญลักษณ์ที่ใช้กับ Relationship ประเภทนี้ ได้แก่ ตัวเลข 1 และอักษร M โดยตัวเลข 1 จะถูกกำหนดไว้ทางด้านของ Entity ที่มีจำนวน participant ที่เกี่ยวข้องกับ Relationship เพียง Participant เดียว ส่วนตัวอักษร M จะถูกกำหนดไว้ทางด้านของ Entity ที่มีจำนวน Participant ที่เกี่ยวข้องกับ Relationship มากกว่า 1 Participant ซึ่งจากรูป ด้าน Entity "CUSTOMER" เป็น Entity ที่มีจำนวน Participant ที่เกี่ยวข้องกับ Relationship เพียง Participant เดียว ส่วนด้าน entity "ACCOUNT" จะเป็น Entity ที่มีจำนวน Participant ที่เกี่ยวข้องกับ Relationship มากกว่า 1 Participant ดังนั้น จึงปรากฏตัวเลข 1 ไว้ทางด้าน Entity "CUSTOMER" และตัวอักษร M ทางด้าน Entity "ACCOUNT" ดังรูป



รูปที่ 2.17 Relationship แบบ One-to-Many

3. Many-to-Many Relationship เป็น Relationship ที่ Participant มากกว่า 1 Participant ของ Entity หนึ่ง มีความสัมพันธ์กันอีก Participant ของอีก Entity หนึ่งมากกว่า 1 Participant เช่น กรณีลูกค้าสามารถมีเงินฝากได้มากกว่า 1 บัญชี และแต่ละบัญชีเงินฝากสามารถมีเจ้าของบัญชีได้มากกว่า 1คน ดังตัวอย่างข้อมูลต่อไปนี้

ตารางที่ 2.22 ตัวอย่าง Relationship แบบ Many-to-many

Entity "CUSTOMER"

NAME	ADDRESS	ACCT_NO
แพง พลเมืองดี	111 บางพลัด กทม.	111111111
แพง พลเมืองดี	111 บางพลัด กทม.	111111112
จิราพร สมตน	222 บางซื่อ กทม.	222222222
สุภาพร อุดมศิลป์	333 ปทุมวัน กทม.	333333333
เอกสิทธิ์ ทับทอง	444 คลองตัน กทม.	111111112

Entity "ACCOUNT"

ACCT_NO	BALANCE
111111111	54000
222222222	12000
333333333	14000
444444444	100000
111111112	4000

Relationship **"BELONG_TO"**

NAME	ADDRESS	ACCT_NO	BALANCE
แพง พลเมืองดี	111 บางพลัด กทม.	111111111	54000
		111111112	4000
จิราพร สมตน	222 บางซื่อ กทม.	222222222	12000
สุภาพร อุดมศิลป์	333 ปทุมวัน กทม.	333333333	14000
กิตติ มั่นคง	444 บางบอน กทม.	444444444	100000
เอกสิทธิ์ ทับทอง	444 คลองตัน กทม.	111111112	4000

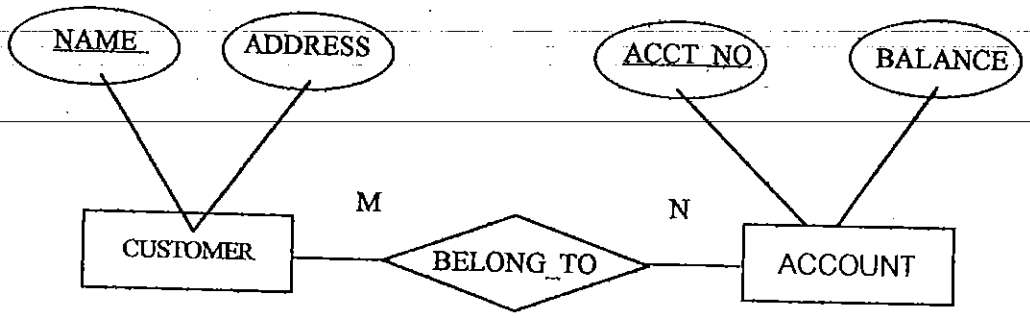
และในมุกมกลับกัน เห็นว่า บัญชีเลขที่ "111111112" มีเจ้าของบัญชี 2 คน คือ **"แพง พลเมืองดี"** และ "เอกสิทธิ์ ทับทอง" ซึ่งแสดงด้วย Relationship "BELONG_TO" ได้เช่นเดียวกัน ดังนี้

ตาราง 2.22(ต่อ) ตัวอย่าง Relationship แบบ Many-to-Many

Relationship **"BELONG_TO"**

ACCT_NO	NAME	ADDRESS	BALANCE
111111112	แพง พลเมืองดี	111 บางพลัด กทม.	4000
	เอกสิทธิ์ ทับทอง	444 คลองตัน กทม.	4000
111111111	แพง พลเมืองดี	111 บางพลัด กทม.	54000
222222222	จิราพร สมตน	222 ปทุมวัน กทม.	12000
333333333	สุภาพร อุดมศิลป์	333 ปทุมวัน กทม.	14000
444444444	กิตติ มั่นคง	444 บางบอน กทม.	1000000

ความสัมพันธ์ระหว่างลูกค้าและบัญชีเงินฝาก จึงเป็น Many-to-Many Relationship สำหรับสัญลักษณ์ที่ใช้กับ Relationship ประเภทนี้ได้แก่ ตัวอักษร M กำหนดไว้ทั้ง 2 ด้านของ Entity ดังรูป

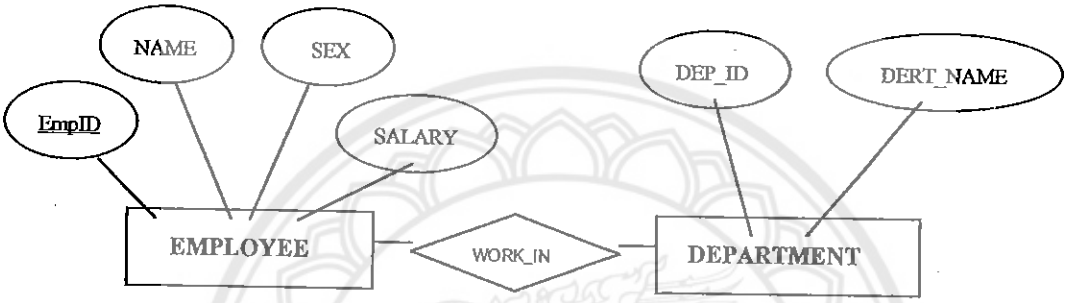


รูปที่ 2.18 Relationship แบบ Many-to-Many

2.7.3.5 ประเภทของ Relationship

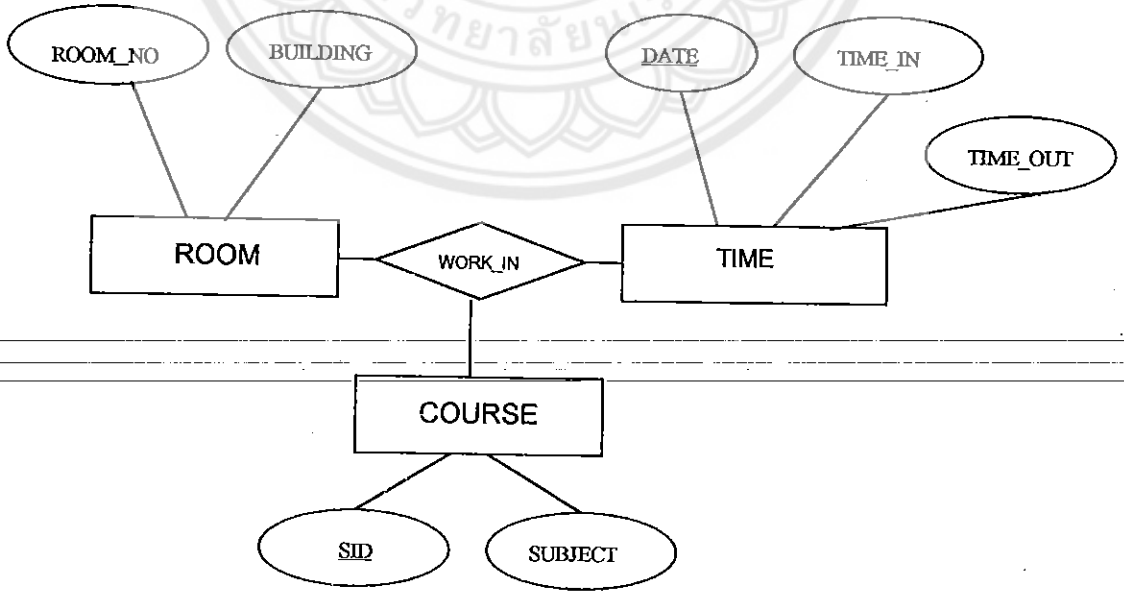
นอกเหนือจากการใช้จำนวนของ Participant ในการจัดประเภทของ Relationship แล้วยังสามารถใช้จำนวนของ Entity ที่มีความสัมพันธ์กับแต่ละ Relationship มากำหนดประเภทของ Relationship ได้ดังนี้

1. Binary Relationship เป็น Relationship ที่พบมากที่สุดในแผนภาพ E_R โดยเป็น Relationship ที่เกิดขึ้นระหว่าง 2 Entity ใด ๆ เช่น Relationship "WORK_IN" ดังรูป



รูปที่ 2.19 Relationship แบบ Binary Relationship

2. N-aray Relationship เป็น Relationship เกิดขึ้นระหว่าง Entity มากกว่า 2 Entity ขึ้นไป เช่น Relationship "SCHEDULE" ซึ่งใช้แสดงตารางเรียนวิชาต่าง ๆ ดังรูป



รูปที่ 2.20 Relationship แบบ N-aray Relationship

ตารางที่ 2.23 Relationship แบบ N-aray Relationship

Entity "ROOM"

ROOM_NO	BUILDING
KMB101	KMB
LB201	KLB
LTB201	LTB

Entity "TIME"

TID	DATE	TIME
T1	จ.อ	8.00-11.00
T2	จ.อ	13.00-16.00
T3	พ.พฤ	8.00-11.00
T4	พ.พฤ	13.00-16.00
T5	ศ.ส	8.00-11.00
T6	ศ.ส	13.00-16.00

Entity "COURSE"

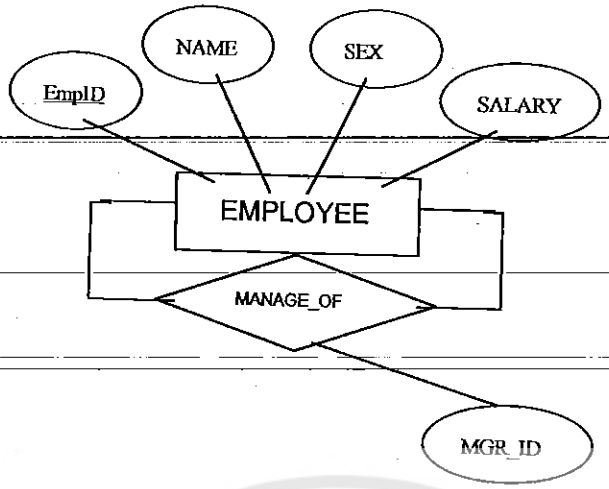
SID	SUBJECT
EN101	อังกฤษ 1
MA111	คณิตศาสตร์ 1
PH111	ฟิสิกส์ 1
CH111	เคมี 1
CH112	เคมี 2
EN102	อังกฤษ 2

Relationship "SCHEDULE"

SID	TID	ROOM_NO
EN101	T1	KMB101
MA111	T2	KMB101
PH111	T3	LTB201
CH111	T2	KLB201
CH112	T3	KLB201
EN102	T1	LTB201

จากตัวอย่างข้อมูล จะสังเกตเห็นว่า Relationship "SCHEDULE" กำหนดขึ้นจาก Property "ROOM_NO" ของ Entity "ROOM" และ PROPERTY "DATA" ของ Entity "TIME" และ Property "SID" ของ Entity "COURSE" มารวมกัน ดังนั้น Relationship "SCHEDULE" นี้จึงมีความสัมพันธ์กับ 3 Entity ดังกล่าว

3. Recursive Relationship เป็น Rlationship ที่เกิดขึ้นกับ Entity เดียว ในกรณีที่ Property ของ Entity นั้น สามารถสร้างความสัมพันธ์กับอีก Property หนึ่งภายใน Entity เดียวกัน เช่น Relationship "MANAGE_OF" ซึ่งใช้แสดงชื่อหัวหน้างานแต่ละคน ดังรูป



รูปที่ 2.21 Relationship แบบ Recursive Relationship

ซึ่งสามารถแสดงด้วยตัวอย่างข้อมูลได้ดังนี้

ตารางที่ 2.24 Relationship แบบ Recursive Relationship

Entity "EMPLOYEE"

EmpID	NAME	SEX	SALARY	MGR_ID
00001	สมชาย นิลกลัด	M	8,500	00003
00002	สมถวิล กลั่นเจริญ	F	9,000	00003
00003	เจริญ ก้าวหน้า	M	12,000	-
00004	ชุติมา สกุลดี	F	10,000	00003
00005	นิวัติ เหล่าสุวรรณ	M	25,000	00003

Relationship "MANAGE_OF"

EmpID	NAME	SEX	SALARY	MGR_ID	MGR_NAME
00001	สมชาย นิลกลัด	M	8,500	00003	เจริญ ก้าวหน้า

2.7.3.6 คุณสมบัติของแผนภาพ E-R ที่ดี

เนื่องจากแผนภาพ E-R ถูกใช้เป็นเครื่องมือในการนำเสนอความเป็นจริงเกี่ยวข้องกับข้อมูล ดังนั้นจึงควรที่จะต้องมีคุณสมบัติดังนี้

1. Expressiveness แผนภาพ E-R ที่ดี จะต้องสามารถอธิบายโครงสร้างของข้อมูลได้เป็นอย่างดีและครบถ้วน
2. Simplicity แผนภาพ E-R ที่ดี จะต้องมีรูปแบบที่ง่ายต่อการเข้าใจ
3. Minimality แผนภาพ E-R ที่ดี จะต้องมีความชัดเจน และไม่สามารถตีความเป็นอื่น
4. Formality แผนภาพ E-R ที่ดี จะต้องไม่ซ้ำซ้อน และมีรูปแบบที่เป็นมาตราฐาน

2.7.3.7 การค้นหา Entity

สิ่งที่จำเป็นสำหรับการวาดแผนภาพ E-R ได้แก่ การกำหนด Entity, Property, Relationship และ Cardinality ที่ใช้ข้อมูลต่างๆ ภายในฐานข้อมูลที่ต้องการออกแบบนั้น ซึ่งมีขั้นตอนที่ค่อนข้างซับซ้อน แต่อย่างไรก็ตามผู้ออกแบบสามารถที่จะกำหนด Entity ขึ้นภายในระบบได้อย่างคราวๆ โดยวิธีการค้นหา Entity ด้วยวิธีของ Data Perspective ซึ่งกำหนดไว้ว่า Entity จะถูกกำหนดขึ้นจากสิ่งต่างๆ ที่ปรากฏอยู่ในความต้องการของผู้ใช้ดังต่อไปนี้

1. ขั้นตอนการทำงาน เอกสาร และรายงานต่างๆ ที่ปรากฏอยู่ในระบบ
2. อุปกรณ์ต่างๆ ที่ระบบใช้การติดต่อ
3. ระบบงานที่เกี่ยวข้องในการส่งผ่านข้อมูลระหว่างระบบกัน
4. เหตุการณ์ที่เกิดขึ้นในระบบ และระบบจะต้องมีหารบันทึกไว้เป็นหลักฐาน
5. บุคคลหรือสิ่งที่เกี่ยวข้องกับระบบงาน
6. สถานที่ที่เกี่ยวข้องกับระบบงาน
7. หน่วยงานต่างๆ ในองค์กรที่เกี่ยวข้องกับระบบงาน

2.8 การทำงาน Normalization [1,2]

การออกแบบฐานข้อมูลด้วย E-R Model มีจุดมุ่งหมายเพื่อนำเสนอข้อเท็จจริงต่างๆที่เกี่ยวข้องกับข้อมูล โดยไม่ได้คำนึงถึงว่า ฐานข้อมูลที่ออกมานั้น จะมีปัญหาทางด้านความซับซ้อนของข้อมูล ความถูกต้องของข้อมูล ความผิดพลาดในการเพิ่ม ลบ และแก้ไข หรือไม่ ดังนั้น จึงต้องมีวิธีการตรวจสอบ และแก้ไขปัญหาต่างๆ เหล่านี้ วิธีการดังกล่าว "การทำ Normalization"

2.8.1 Normalization

เป็นวิธีการที่ใช้ในการตรวจสอบ และแก้ไขปัญหาทางด้านความซับซ้อนของข้อมูล โดยดำเนินการให้ข้อมูลในแต่ละ relation อยู่ในรูปที่เป็นหน่วยเล็กที่สุดที่ไม่สามารถแตกออกเป็นหน่วยย่อยๆ ได้อีก โดยยังคงความสัมพันธ์ระหว่างข้อมูลใน Relation ต่างๆ ไว้ตามหลักการที่กำหนด ไว้ใน Relation Model

การทำ Normalization นี้เป็นการดำเนินงานอย่างเป็นระดับ ที่กำหนดไว้ด้วยกัน เป็นขั้นตอน ตามปัญหาที่เกิดขึ้นในตอนขั้นๆ ซึ่งแต่ละขั้นตอนจะมีชื่อตามโครงสร้างข้อมูลที่กำหนดไว้ ดังนี้

1. ขั้นตอนการทำ First Normal Form (1NF)
2. ขั้นตอนการทำ Second Normal Form (2NF)
3. ขั้นตอนการทำ Third Normal Form (3NF)
4. ขั้นตอนการทำ Boyce-Codd Normal Form (BCNF)
5. ขั้นตอนการทำ Fourth Normal Form (4NF)

ในแต่ละขั้นตอนของการทำ Normalization จะมีการระบุรูปแบบของโครงสร้างของข้อมูลที่ควรจะเป็น เรียกว่า Normal Form ไว้ ซึ่งโครงสร้างเหล่านี้จะสามารถแก้ไขปัญหาที่เกิดขึ้นในโครงสร้างที่ระบุนี้ จะสามารถแก้ไขปัญหาที่เกิดขึ้นในโครงสร้างข้อมูลของขั้นตอนก่อนหน้าได้หรือกล่าวอีกนัยหนึ่ง การทำ Normalization ในขั้นตอน จะต้องอาศัยผลที่ได้จากการทำ Normalization ในขั้นตอนก่อนหน้า มาปรับปรุงเพื่อให้มีโครงสร้างเป็นไปตามโครงสร้างที่กำหนดไว้ในขั้นตอนนั้นๆ แต่อย่างไรก็ตาม ในการทำ Normalization ไม่จำเป็นต้องเริ่มจากขั้นตอนการทำ First Normal Form เสมอไป กล่าวคือ การทำ First Normal Form และสิ้นสุดในขั้นตอนการทำ Fourth Normal Form เสมอไป

2.8.2 First Normal Form (1NF)

เป็นขั้นตอนสำหรับปรับโครงสร้างของข้อมูลของ Relation เพื่อให้ทุก Attribute ของ Relation มีคุณสมบัติ Atomicity กล่าวคือ โครงสร้างข้อมูลของ Relation ในแบบ 1NF นี้ จะต้องประกอบด้วย Attribute ที่ไม่อยู่ในรูป Repeating Group เช่น ตัวอย่างข้อมูลต่อไปนี้

ตารางที่ 2.25 ตัวอย่าง First Normal Form

ORDER

CUST_NO	CUST_NAME	CITY	ZONE_SALE	ORDER_CONTENT	
					ORDER_QTY
C001	นารี เกิดสว่าง	อยุธยา	001	P001	24
				P003	30
				P004	50
C002	สลักจิต สว่างภพ	ศรีสะเกษ	002	P001	29
				P002	40
				P004	30

จากตารางจะสังเกตเห็นว่า (Attribute "CUST_NO") 1 คนสามารถมีรายการสินค้าที่สั่งซื้อ (Attribute "CUST_NAME") ได้มากกว่า 1 รายการ ดังนั้นจึงกล่าวได้ว่า Attribute "CUST_NO" นี้ มีความสัมพันธ์กับ Attribute "ORDER_CONTENT" ในแบบ Repeating Group ส่งผลให้ Relation นี้มีโครงสร้างที่ไม่สอดคล้องกับ 1NF ดังนั้นจึงต้องทำการ Normalization โดยการแปลง Attribute ที่อยู่ในรูป Repeating Group ให้มีคุณสมบัติ Atomicity พร้อมกับกำหนดให้ Attribute ดังกล่าวเป็น Relation Key ของ Relation ดังนั้น จากตัวอย่างข้างต้นจึงถูกแปลงให้อยู่ในรูปดังนี้

ตารางที่ 2.25 (ต่อ)ตัวอย่าง First Normal Form

ORDER

CUST_NO	CUST_NAME	CITY	ZONE_SALE	PRODUCT_ID	ORDER_QTY
C001	นารี เกิดสว่าง	อยุธยา	001	P001	24
C001	นารี เกิดสว่าง	อยุธยา	001	P003	30
C001	นารี เกิดสว่าง	อยุธยา	004	P004	50
C002	สลักจิต สว่างภพ	ศรีสะเกษ	001	P001	29

จะสังเกตเห็นว่า แต่ละค่าที่เป็น Repeating Group ของ Attribute "PRODUCT_ID" และ "ORDER_QTY" จะถูกแยกออกมา Tuple ใหม่ พร้อมกำหนดให้ Attribute "PRODUCT_ID" ทำหน้าที่ Relation Key เป็นร่วมกับ Attribute "CUST_NO"

อย่างไรก็ตาม Relation ที่อยู่ในรูป 1NF ถึงแม้จะทำให้ทุก Attribute มีคุณสมบัติ Atomicity แต่กลับเกิดปัญหาความซ้ำซ้อนกันของข้อมูล (Redundacy) ขึ้นใน Attribute "CUST_NO", "CUST_NAME", "CITY" และ "ZONE_SALE" และก่อให้เกิดปัญหาทางด้าน Anomaly ตามมาดังนี้

1. Insert Anomaly

เมื่อพิจารณาจากตัวอย่าง จะสังเกตเห็นว่าการเพิ่มข้อมูลลูกค้า จะทำได้ก็ต่อเมื่อลูกค้านั้นมีรายการสั่งซื้อสินค้าแล้วเท่านั้น หมายถึงว่า Relation นี้จะไม่สามารถจัดเก็บข้อมูลของลูกค้าที่ยังไม่มีการสั่งซื้อสินค้า ได้ เช่น เมื่อต้องการเพิ่มข้อมูลลูกค้ารหัส "C006" โดยที่ยังไม่มีการกำหนดการสั่งซื้อสินค้าใน Attribute "PRODUCT_ID" จะไม่สามารถกระทำได้ เนื่องจากการ Attribute "PRODUCT_ID" นี้ถูกกำหนดให้เป็น Relation Key จึงไม่สอดคล้องตามกฎของ Entity Integrity Rule ในส่วนที่ว่า Attribute หรือกลุ่ม Attribute ที่เป็น Relation Key จะมีค่าเป็น Null ไม่ได้

2. Delete Anomaly

เมื่อพิจารณาจากตัวอย่าง จะสังเกตเห็นว่า การลบข้อมูลรายการสั่งซื้อ (Attribute "PRODUCT_ID" และ "ORDER_QTY") บาง Tuple ใน Relation นี้จะทำให้ข้อมูลลูกค้าบางคนสูญหายไป เช่น เมื่อลบข้อมูลรายการสั่งซื้อของสินค้ารหัส "P005" ของลูกค้ารหัส "C003" จะทำให้ข้อมูลลูกค้ารหัส "C003" ถูกลบตามไปด้วย

3. Update Anomaly

เมื่อพิจารณาจากตัวอย่าง จะสังเกตเห็นว่า การปรับปรุงข้อมูลใน Tuple ที่มีค่าของข้อมูลซ้ำซ้อนกันไม่ครบถ้วน อาจก่อให้เกิดความขัดแย้งของข้อมูลที่ซ้ำซ้อนกันนั้นได้ เช่น การเปลี่ยนชื่อลูกค้าจาก " สลักจิต สว่างภพ" เป็น "รินลณี สว่างภพ" ของลูกค้ารหัส "C002" ซึ่งถ้าแก้ไขไม่ครบแล้ว จะทำให้ ลูกค้ารหัส "C002" มีชื่อทั้ง "สลักจิต สว่างภพ" และ "รินลณี สว่างภพ"

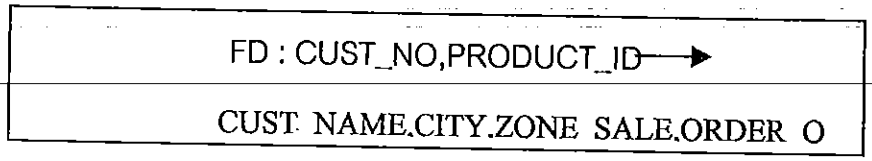
2.8.3 Second Normal Form (2NF)

ในการทำ Normalization ในขั้นตอน Second Normal Form จำเป็นต้องรู้จักถึง Prime Attribute และ Nonprime Attribute เนื่องจาก Attribute ทั้ง 2 ประเภทนี้ จะมีความสำคัญต่อการทำ Normalization แบบ Second Normal Form

Prime Attribute ได้แก่ ทุก Attribute ที่ทำหน้าที่เป็น Relation Key ของ Relation ส่วน Nonprime Attribute ได้แก่ Attribute ที่ไม่ได้เป็นส่วนหนึ่งของ Relation Key

1. ต้องมีโครงสร้างเป็นไปตามโครงสร้างของ 1NF
2. ทุก Nonprime Attribute จะต้องไม่ขึ้นกับ Relation Key ที่อยู่ในรูป Subset

ตัวอย่างเช่น Relation "ORDER1" ในตัวอย่าง ที่ผ่านมา ซึ่งเป็นRelation ที่มีคุณสมบัติของ 1NF จะสังเกตเห็นว่า Attribute (Cust No, Product ID) เป็น Attribute ที่ทำให้ข้อมูลในแต่ละ Tuple มีค่าไม่ซ้ำกัน ดังนั้น Attribute ทั้ง 2 จึงทำหน้าที่เป็น Relation keys ซึ่งสามารถเขียนได้ด้วย Functional Dependency ได้ดังนี้



เมื่อพิจารณาค่าของ Attribute "CUST_NO", "CUST_NAME", "CITY", "ZONE_SALE" จะสังเกตว่า Tuple ที่ประกอบขึ้นจาก Attribute เหล่านี้ จะมีข้อมูลที่ซ้ำกันเป็นชุดๆ

และมีเพียง Attribute "ORDER_QTY" เท่านั้น ที่มีค่าเปลี่ยนตามค่าของ Relation Key ดังนั้นจึงสามารถเขียนด้วย Functional Dependency ได้ดังนี้

d1 : CUST_NO,PRODUCT_ID ⟶ ORDER_QTY

d2 : CUST_NO ⟶ CUST_NAME,CITY,ZONE_SALE

ใน d2 จะสังเกตเห็นว่า Attribute "CUST_NAME", "CITY", และ "ZONE_SALE" เป็น Nonprime Attribute ของ Relation ที่ไม่ได้ขึ้นอยู่กับเฉพาะ Relation "ORDER1" จึงไม่มีคุณสมบัติเป็นไปตามคุณสมบัติของ 2NF จึงต้องแตก Relation "ORDER1" ออกเป็น 2 Relation ตาม d1 และ d2 ดังนี้

ตารางที่ 2.26 ตัวอย่าง Second Normal Form

CustOrder

CUST_NO	PRODUCT_ID	ORDER_QTY
C001	P001	24
C001	P003	30
C001	P004	50
C002	P001	29

Cust

CUST_NO	CUST_NAME	CITY	ZONE_SALE
C001	นารี เกิดสว่าง	อยุธยา	001
C002	สลักจิต สว่างภพ	ศรีสะเกษ	002
C003	สุทิศา เจ็กสกุล	เชียงใหม่	004
C004	ฟ้า เพิ่มพร	ศรีสะเกษ	002
C009	ต้นสาย ต้นเจริญ	เชียงใหม่	004

สำหรับโครงสร้างของ Relation "CustOrder" และ "Cust" นี้ จะสังเกตเห็นว่าสามารถแก้ปัญหา Anomaly ที่เกิดขึ้นใน Relation "ORDER1" ได้ดังนี้

- Relation "cust" สามารถเพิ่มข้อมูลลูกค้าได้โดยไม่ต้องมีการสั่งซื้อสินค้าที่เกิดขึ้น เนื่องจากข้อมูลการสั่งซื้อสินค้าจะถูกแยกจัดเก็บใน Relation "CustOder"จึงสามารถแก้ปัญหา Insert Anomaly ที่เกิดขึ้นได้
- สามารถลบข้อมูลรายการสั่งซื้อสินค้ารหัส "P005" ของลูกค้ารหัส "C003" ใน Relation "CustOder" ได้โดยไม่ส่งผลต่อข้อมูลของลูกค้ารหัส "C003" เนื่องจากข้อมูลลูกค้าถูกแยกจัดเก็บใน Relation "Cust" จึงสามารถแก้ปัญหา Delete Anomaly ที่เกิดขึ้นได้
- เนื่องจากข้อมูลของลูกค้าที่ซ้ำซ้อนจะถูกแยกมาจัดเก็บใน Relation "Cust" ดังนั้น การเปลี่ยนแปลงรายละเอียดข้อมูลลูกค้าจึงกระทำกับRelation "Cust" เพียง Relationเดียว จึงก่อให้เกิดปัญหา Update Anomaly

2.8.4 Third Normal Form (3NF)

สำหรับ Relation ที่จะมีโครงสร้างในแบบ 3NF จะต้องมีคุณสมบัติดังนี้

1. ต้องมีคุณสมบัติของ 2NF
2. ต้องไม่มี Functional Dependencyเกิดขึ้นระหว่าง Nonprime Attribute ด้วยกันเอง ที่เรียกว่า "Transitive Dependency"จาก Relation "Cust" ในหัวข้อที่ผ่านมา ถึงแม้ว่าจะมีโครงสร้างเป็นไปตามคุณสมบัติของ 2NF แต่จะสังเกตเห็นว่า ค่าของ Atteribute "CITY" และ "ZONE_SALE" ถ้าปรากฏข้อมูลที่มีค่าซ้ำซ้อนกันอยู่เป็นคู่ ๆ หรือกล่าวอีกนัยหนึ่งทั้ง 2 Attribute สามารถที่จะระบุค่าระหว่างกันได้ กล่าว คือ เมื่อระบุค่าให้กับ Attribute "ZONE_SALE" จะสามารถทราบถึงชื่อเมืองใน Attribute"CITY" ได้ ซึ่งความสัมพันธ์ในลักษณะนี้ จะเรียกว่า Transitive Dependency ดังนั้น Relation นี้จึงขาดคุณสมบัติของ 3NF และยังก่อให้เกิดปัญหาความผิดพลาดทางด้าน Anomaly ดังนี้

1. Insert Anomaly

ใน Relation "Cust" เมื่อต้องการเพิ่มข้อมูลให้กับ Attribute "CITY" และ "ZONE_SALE" ซึ่งมีความสัมพันธ์ในแบบ Transitive Dependency จะไม่สามารถกระทำได้เนื่องจากข้อมูลใน 2 Attribute นี้ ไม่ใช่ Relation Key ดังนั้น จึงต้องเพิ่มข้อมูลนี้ของลูกค้ารายใหม่ให้กับ Attribute "CUST_NO" และ "CUST_NAME" ตามไปด้วย

2. Update Anomaly

ใน Relation "Cust" จะสังเกตเห็นว่า เมื่อมีการแก้ไขข้อมูลใน Attribute "ZONE_SALE" จาก "004" ไปเป็น "005" จะต้องทำการแก้ไขข้อมูลในทุก ๆ Tuple ที่มีค่าของ Attribute "ZONE_SALE" เท่ากับ "004" ให้ครบถ้วน เนื่องจากเมื่อทำการแก้ไขข้อมูลไม่ครบถ้วนจะก่อให้เกิดข้อมูลใน Relation ที่มีความขัดแย้งกันได้

3. Delete Anomaly

ใน Relation "Cust" จะสังเกตเห็นว่า ถ้ามีการลบข้อมูลของ Tuple ที่จัดเก็บข้อมูลในกลุ่ม Transitive Dependency ที่ปรากฏอยู่เพียงชุดเดียวใน Relation จะส่งผลให้ข้อมูลในกลุ่ม Transitive Dependency นั้นสูญหายไปจาก Relation ได้เช่น เมื่อทำการลบข้อมูลใน Tuple ของลูกค้ารหัส "C001" นอกจากจะทำให้ข้อมูลของลูกค้ารหัส "C001" หายไปแล้ว ยังส่งผลให้ข้อมูลของเขตการขายที่อยุธยาสูญหายไปด้วย

จากปัญหา Anomaly ที่เกิดขึ้นจาก Transitive Dependency เหล่านี้ จึงต้องทำการแยก Nonprime Attribute ที่ก่อให้เกิด Transitive Dependency ของ Relation "Cust" ออกมาเป็น Relation ใหม่ ซึ่งจากตัวอย่าง ได้แก่ Attribute "CITY" และ "ZONE_SALE" ดังนี้

ตารางที่ 2.27 ตัวอย่าง Third Normal Form

Cust2

CUST_NO	CUST_NAME	CITY
C001	นารี เกิดสว่าง	อยุธยา
C002	สลักจิต สว่างภพ	ศรีสะเกษ
CUST_NO	CUST_NAME	CITY
C003	สุทิศา เจ็กสกุล	เชียงใหม่
C004	ฟ้า เพิ่มพร	ศรีสะเกษ

CityZone

CITY	ZONE_SALE
อยุธยา	001
ศรีสะเกษ	002
เชียงใหม่	004

ซึ่งสามารถเขียนด้วย Functional Dependency ได้ดังนี้

d1 : CUST_NO → CUST_NAME,CITY

d2 : CITY → ZONE_SALE

ข้อสังเกต ในการแยก Nonprime Attribute ที่ก่อให้เกิด Transitive Dependency ออกมาเป็น Relation ใหม่ มีหลักการอยู่ 2 ข้อดังนี้

1. แยก Nonprime Attribute ที่ก่อให้เกิด Transitive Dependency ออกเป็น Relation ใหม่
2. กำหนดให้ Nonprime Attribute ที่เป็นตัวระบุค่า ให้เป็น Relation Key ของ Relationใหม่

2.8.5 Fourth Normal Form (4NF)

สำหรับ Relation ที่จะมีโครงสร้างแบบ 4NF จะต้องมีคุณสมบัติดังนี้

1. ต้องมีคุณสมบัติของ BCNF
2. ต้องไม่ปรากฏความสัมพันธ์ระหว่าง Attribute ในแบบ Multi-value Dependency เช่น ตัวอย่างข้อมูลของ Relation "EMPOLYEE_SKILL" ซึ่งใช้จัดเก็บข้อมูลเกี่ยว กับความสามารถของพนักงาน ทางด้านการใช้คอมพิวเตอร์ (Attribute "COMPUTER_SKILL")และทางด้านภาษาต่างประเทศ (Attribute "LANGUAGE_SKILL") ดังนี้

ตารางที่ 2.28 ตัวอย่าง Fourth Normal Form

EMPLOYEE_SKILL

EMPLOYEE#	COMPUTER_SKILL	LANGUAGE_SKILL
1267	Word Processing	ฝรั่งเศส
1267	Word Processing	เยอรมัน
1345	Spreadsheets	สเปน
1267	Spreadsheets	ฝรั่งเศส
1267	Spreadsheets	เยอรมัน
1345	COBOL	สเปน
1193	Word Processing	ฝรั่งเศส

จากตัวอย่างข้อมูล จะสังเกตเห็นว่า Relation นี้ มีคุณสมบัติเป็น BCNF แต่ยังไม่เป็น 4NF เนื่องจาก ปรากฏโครงสร้างข้อมูลในแบบ Multi-value Dependency กล่าวคือ เมื่อระบุค่าของ Attribute "EMPLOYEE#" ซึ่งทำหน้าที่เป็น Determinant จะสามารถแสดงค่าของ Attribute"COMPUTER_SKILL" และ "LANGUAGE_SKILL" ที่ทำหน้าที่เป็น Dependency ได้มากกว่า 1 ค่า

ดังนั้น จึงต้องแบ่ง Relation นี้ออก เป็น Relation ใหม่ตามโครงสร้างข้อมูลแบบ Multi-value Dependency ที่ปรากฏอยู่ ซึ่งได้แก่ Relation "COMPUTER_SKILL" และ "LANGUAGE_SKILL" ดังนี้

ตารางที่ 2.28 (ต่อ) ตัวอย่าง Fourth Normal Form

COMPUTER_SKILL

EMPLOYEE#	COMPUTER_SKILL
1267	Word Processing
1345	Spreadsheets
1267	Spreadsheets
1345	COBOL
1193	Word Processing

LANGUAGE_SKILL

EMPLOYEE#	LANGUAGE_SKILL
1267	ฝรั่งเศส
1267	เยอรมัน

2.8.7 สรุป

โครงสร้างของฐานข้อมูลที่ออกแบบขึ้น ควรที่จะนำมาปรับปรุงโดยใช้วิธีของการทำ Normalization เพื่อปรับเปลี่ยนโครงสร้างของ Relation ต่าง ๆ ที่ได้ออกแบบไว้ ให้มีโครงสร้างที่เป็นไปตามคุณสมบัติของ Relation ที่กำหนดไว้ใน Relation Model รวมทั้งมีโครงสร้างที่เหมาะสมต่อการนำไปใช้งานในการทำ Normalization จะมีอยู่ด้วยกัน 5 ขั้นตอน แต่ส่วนใหญ่ในทางปฏิบัติ แล้วการทำ Normalization จะกระทำถึงขั้นตอน Boyce-Codd Normal Form (BCNF) เท่านั้นก็ เพียงพอแล้ว

บทที่ 3

การออกแบบโปรแกรมฐานข้อมูลฝ่ายเภสัชกรรม

3.1 ขั้นตอนการดำเนินงาน

1. ศึกษาทฤษฎีและหลักการพื้นฐาน ในการทำ Database การเขียน ER การใช้ Visual Basic 6.0 และการเขียนคำสั่งเพื่อการทำงาน SQL

2. รวบรวมข้อมูลเกี่ยวกับยาต่างๆ และข้อมูลที่เกี่ยวข้อง เช่น การแบ่งประเภทของยา การจำแนกยา และลักษณะอาการที่ควรใช้กับยาแต่ละประเภท เป็นต้น

3. สำรวจและรวบรวมความต้องการของผู้ใช้ (Requirement) ของผู้ใช้ ซึ่งมีดังนี้

3.1 ความต้องการข้อมูลของ สต็อกยา ได้แก่

3.1.1 ความต้องการด้านการซื้อยา ประเภทของยา วันที่ลงทะเบียน วันที่หมดอายุ ราคาซื้อ ราคาขาย ชนิดยา เป็นต้น

3.1.2 ความต้องการด้านการตัดสต็อกยา คือถ้ายาต่ำกว่าค่าขั้นต่ำที่มีในสต็อกหรือว่าหมดแล้ว จะฟ้องการสั่งเพิ่ม Order ได้ไหม และจะลบยาตัวเดิมได้ยังไง ถ้ายานั้นไม่ต้องการหรือว่าไม่ใช้แล้ว

3.1.3 การกำหนดหมายการรับยาครั้งต่อไป

3.2 ความต้องการด้านบุคลากรที่เกี่ยวข้องกับ ฝ่ายเภสัชกรรม

3.2.1 ต้องการประวัติของบุคลากร เช่น เภสัชกร หรือ แพทย์ที่สั่งยาเป็นต้น

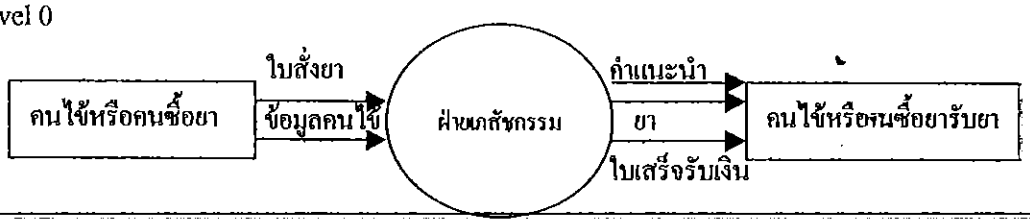
3.2.2 ความต้องการด้านข้อมูลของลูกค้า ในกรณีร้านขายยาและส่วนลด ค่าใช้จ่ายของลูกค้า

3.3 สามารถเรียกดูข้อมูลยาทั้งหมดที่มีในสต็อก ทั้งใหม่และเก่าได้ หรือค้นหาชื่อยาและ ประเภทของยาได้

3.4 เรียกดูข้อมูลของบุคลาทั้งหมดได้ หรือจะหาเฉพาะบุคคลได้

4. นำข้อมูลที่ได้ทั้งหมดมา รวบรวม และ วิเคราะห์ เพื่อดู ส่วนข้อมูลเข้า และ ส่วนแสดงผลของการทำโปรแกรม และการเขียน DFD Level 0 ของ DFD ได้

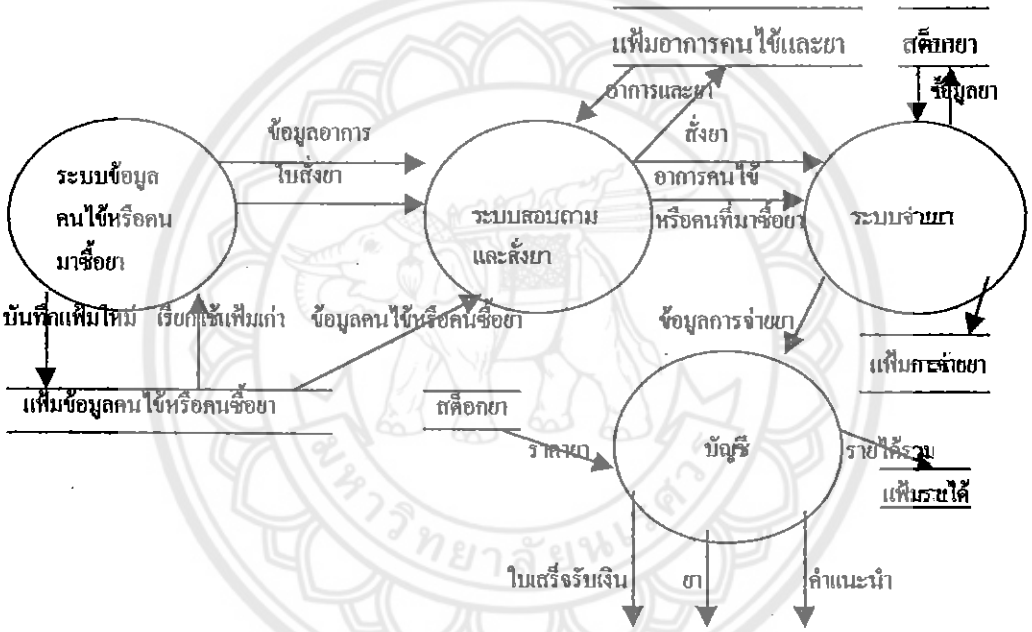
DFD Level 0



รูปที่ 3.1 DFD Level 0

เราจะเห็นได้ว่าขั้นตอนที่ 4 จะมองภาพรวมของทั้งระบบ ได้อย่างคร่าวๆ คือ ได้ Input และ Output นั้นเอง

Process 0 Level 1

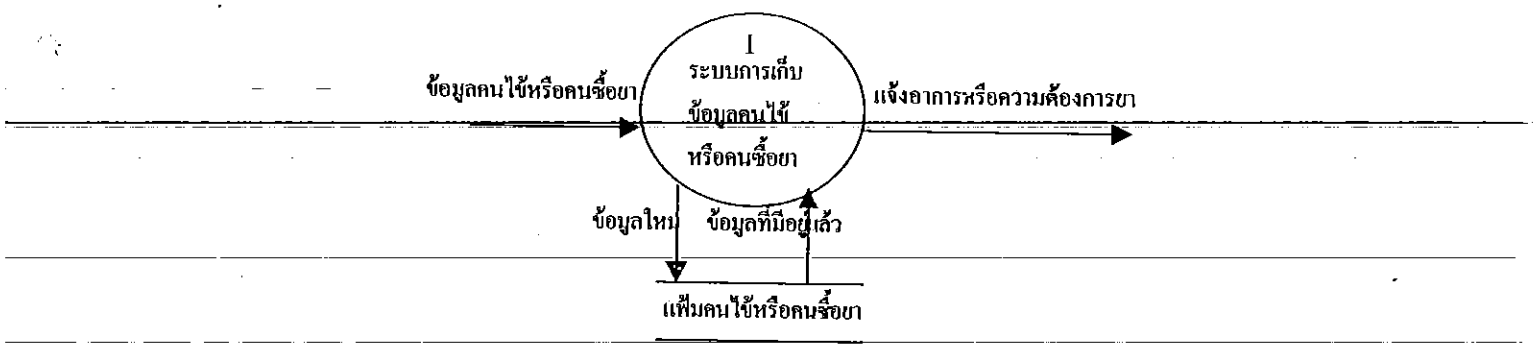


รูปที่ 3.2 แผนภาพ DFD ของกระบวนการ 0 ระดับ 1

จากแผนภาพ การไหล ของ Level 1 นี้ เราจะแสดงรายละเอียด ของกระบวนที่ 0 ใน Level 0 ออกเป็น 4 ส่วนประกอบด้วย

1. ระบบข้อมูลคนไข้หรือคนซื้อยา
2. ระบบสอบถามและสั่งยา
3. ระบบจ่ายยา
4. ระบบบัญชี

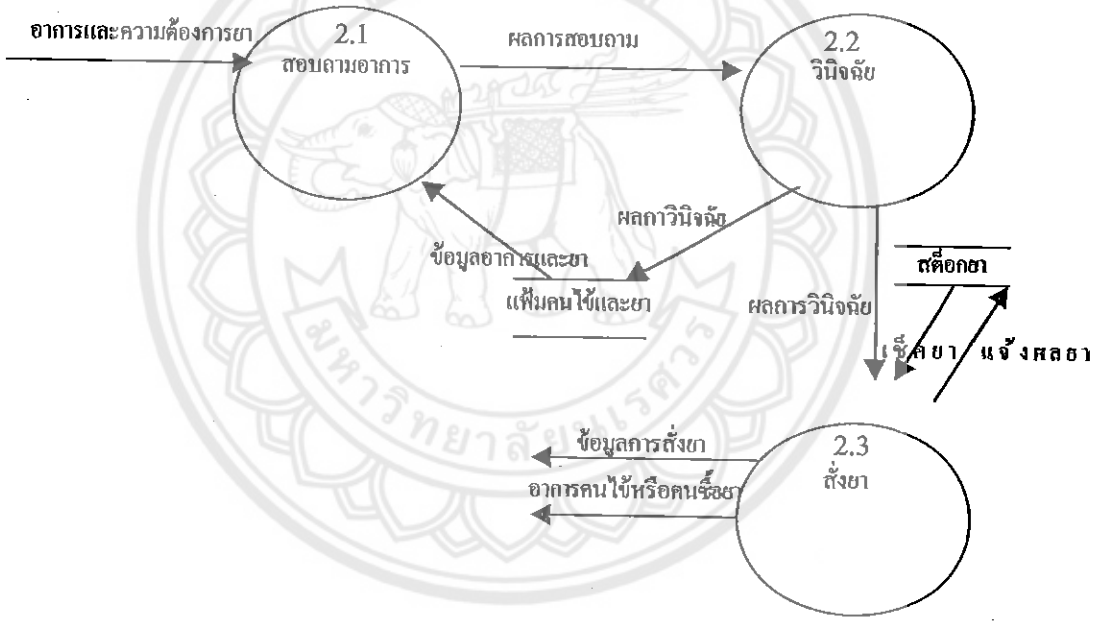
Process 1 Level 1



Process1.1 : ระบบการจัดเก็บข้อมูลคนไข้ หรือคนมาซื้อยา

รูปที่ 3.3 แผนภาพ DFD ของกระบวนการที่ 1 ระดับที่ 1

Process 2 Level 1



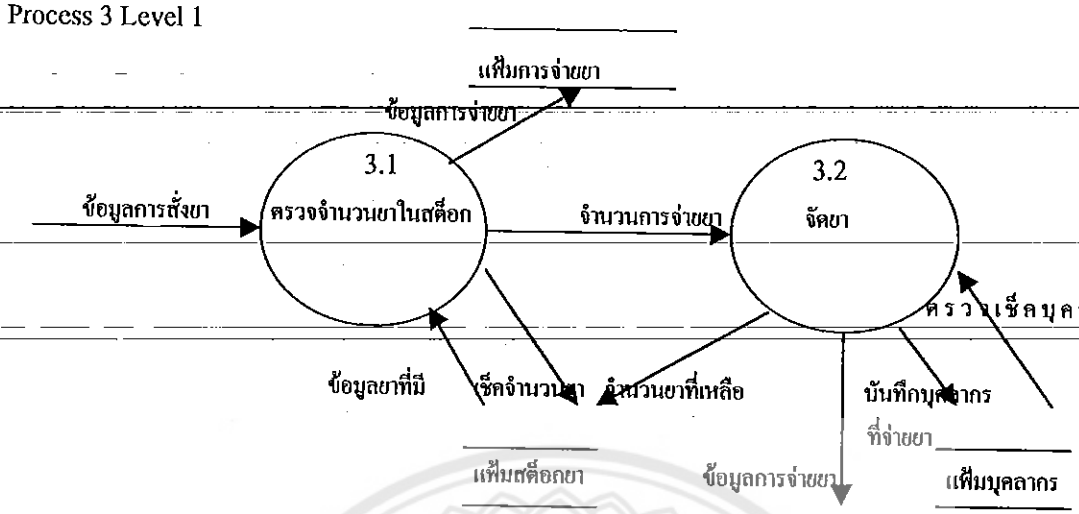
Process2.1 : สอบถามอาการ

Process2.2 : วินิจฉัยหรือดูจากแฟ้มคนไข้

Process2.3 : สั่งยา

รูปที่ 3.4 แผนภาพ DFD ของกระบวนการที่ 2 ระดับที่ 1

Process 3 Level 1

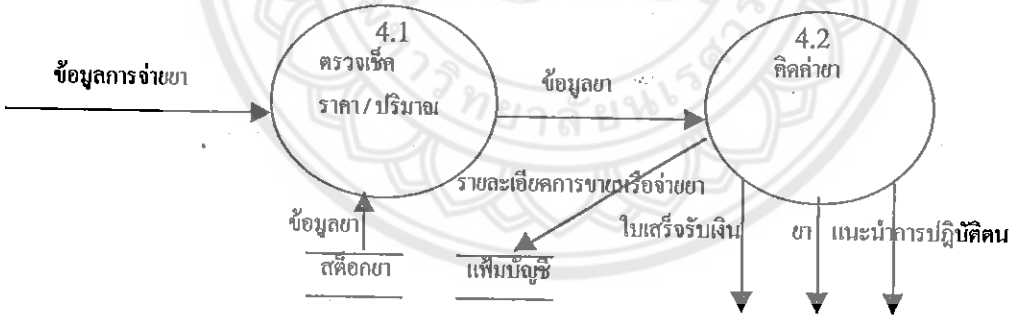


Process3.1 : ตรวจจำนวนยาในสต็อก

Process3.2 : จัดยาตามสั่ง

รูปที่ 3.5 แผนภาพ DFD ของกระบวนการที่ 3 ระดับที่ 1

Process 4 Level 1

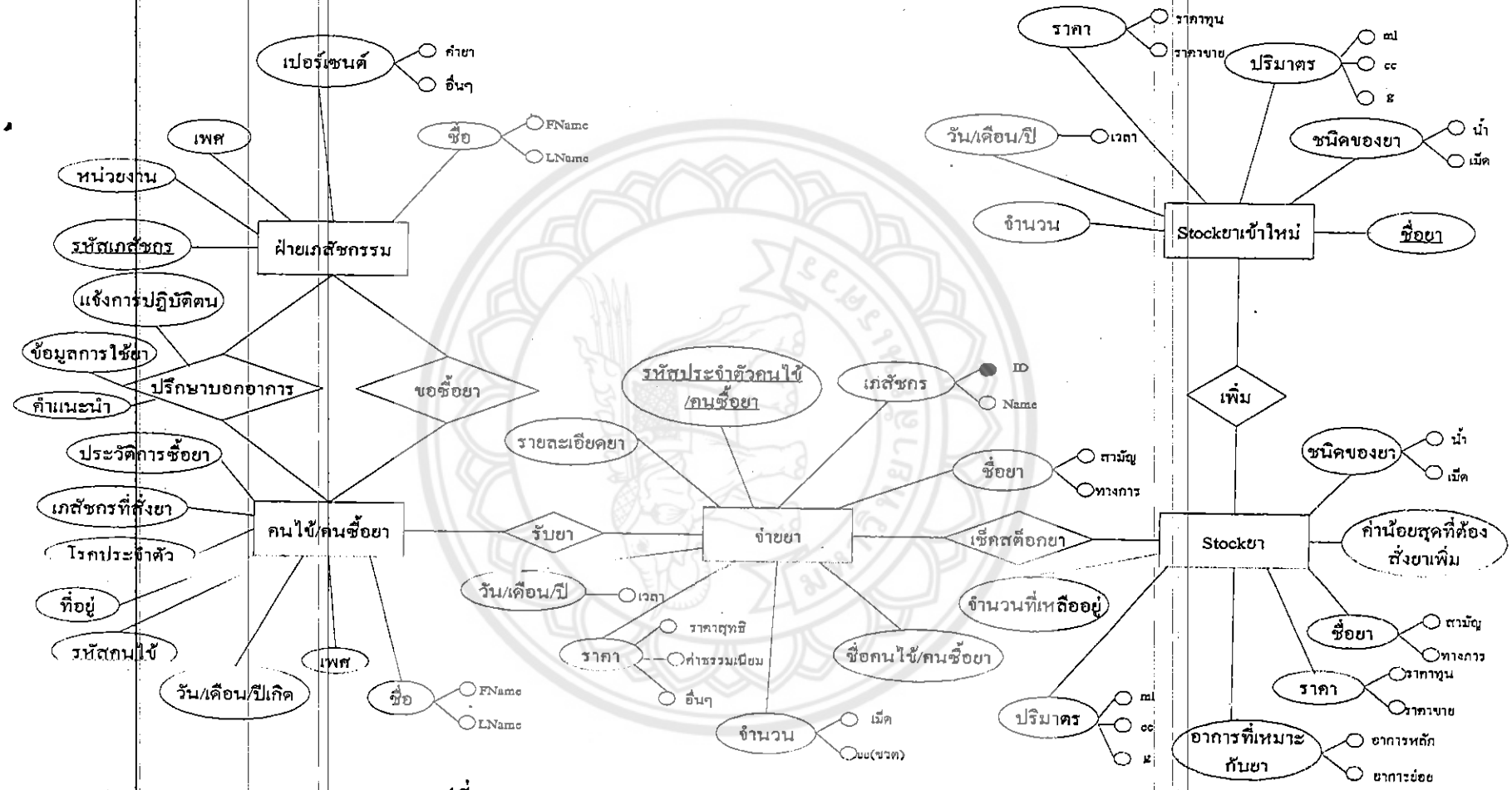


Process4.1 : ตรวจเช็คปริมาณยา / ราคา

Process4.2 : คิดค่ายารวม พร้อมออกใบเสร็จ และให้คำแนะนำการใช้ยา

รูปที่ 3.6 แผนภาพ DFD ของกระบวนการที่ 4 ระดับที่ 1

5. ออกแบบ ER Diagram ของโปรแกรมการจัดการฐานข้อมูลฝ่ายเภสัชกรรม



รูปที่ 3.7 ER Diagram โปรแกรมการจัดการฐานข้อมูลฝ่ายเภสัชกรรม

6. ออกแบบ Database ให้ครอบคลุม Requiment และง่าย สะดวกที่สุด ซึ่ง Database จะแบ่งเป็นส่วนหลัก ๆ ดังนี้

ตารางที่ 3.1 ฐานข้อมูลโปรแกรมการจัดการฐานข้อมูลฝ่ายเภสัชกรรม

6.1 ฐานข้อมูลประวัติคนไข้หรือคนที่มาซื้อยา

รหัส	ชื่อ	เพศ	ศาสนา	วันเกิด	อายุ	ที่อยู่	อำเภอ	จังหวัด	รหัสไปษณีย์	โทรศัพท์
แพ้ยา		โรคประจำตัว		ผู้จ่ายยา						

6.2 ฐานข้อมูลกำหนดนัดรับยาครั้งต่อไป

วันที่นัด	รหัส	ชื่อ	รหัสเภสัชกร	ชื่อคนจ่าย	วันที่นัด	หมายเหตุ

6.3 ฐานข้อมูลอาการ

วันที่	ชื่อ	เวลา	เภสัชกร	น้ำหนัก	ส่วนสูง	ความดันโลหิต	อุณหภูมิ	อาการ	อื่นๆ

6.3 ฐานข้อมูลเภสัชกร

รหัส	ชื่อ	เปอร์เซ็นต์ค่ายา	ค่าใบประกอบวิชาชีพ	ระดับเงินเดือน	อื่นๆ

6.4 ฐานข้อมูลสต็อกยาปัจจุบัน

ชื่อยา	ชนิด	ประเภท	กลุ่มอาการหลัก	กลุ่มอาการย่อย	วันที่	ยาน้อยสุด	ราคาทุน
ราคาขาย	ยอดคงเหลือ						

6.5 ฐานข้อมูลสต็อกยาใหม่

ชื่อยา	วันที่	จำนวน	ชนิด	ราคาทุน	ราคาขาย

6.6 ฐานข้อมูลการจ่ายยา

วันที่	ชื่อคนไข้	ชื่อยา	ชนิด	จำนวนต่อครั้ง	จำนวนยาทั้งหมด	ค่ายารวม	ค่าห้อง
คำแนะนำการใช้ยา		ชื่อเภสัชกร		อื่นๆ			

6.7 ฐานข้อมูลญาติคนไข้

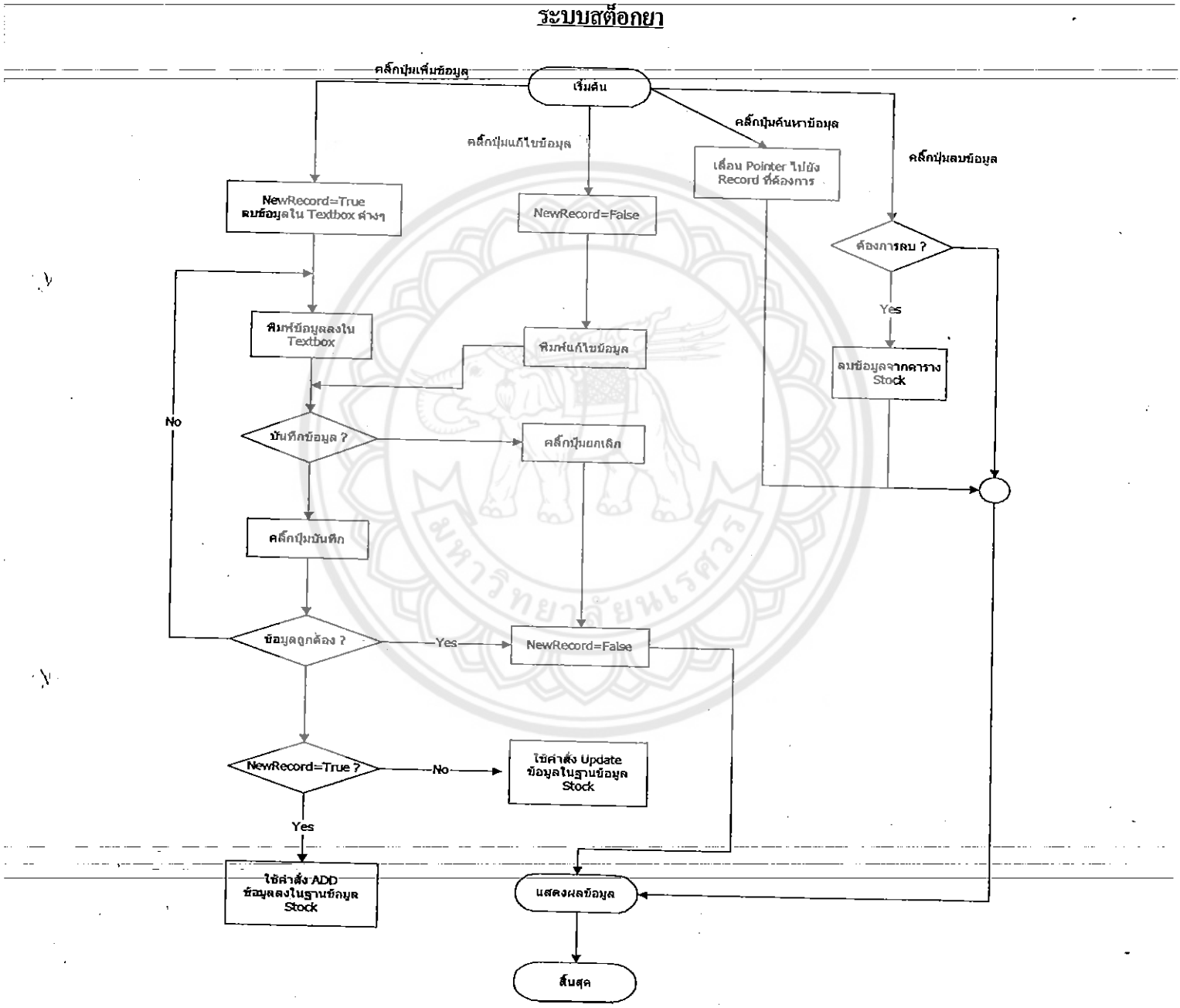
ชื่อ	ที่อยู่	เบอร์โทรศัพท์	ความสัมพันธ์

บทที่ 4

การพัฒนาโปรแกรม

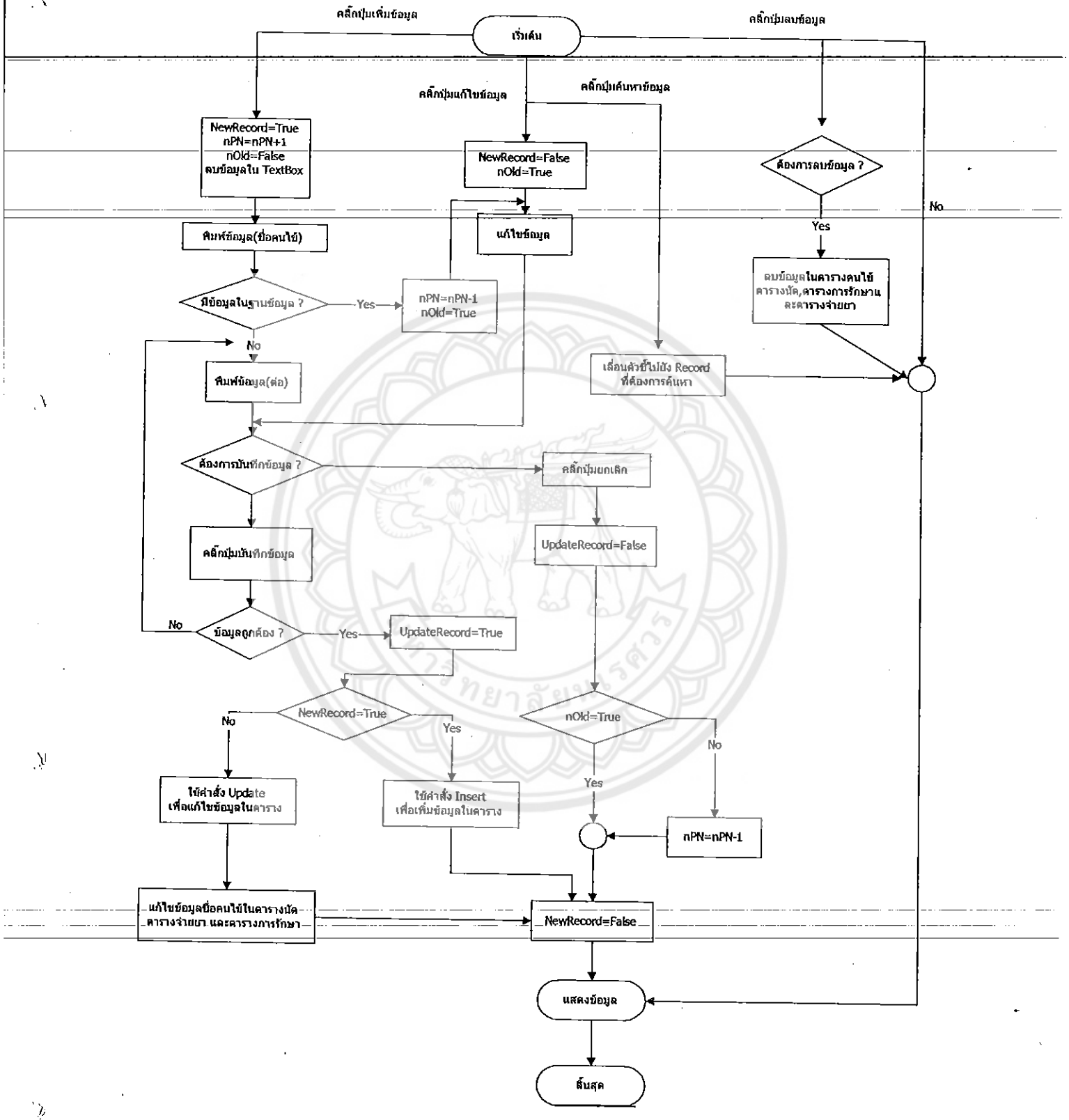
โปรแกรมการจัดการฐานข้อมูลฝ่ายเภสัชกรรม ถูกเขียนโดยใช้โปรแกรม Microsoft Visual Basic 6.0 เป็นส่วนหน้าจอติดต่อกับผู้ใช้ และ Microsoft Access 97 เป็นส่วนฐานข้อมูล มีแนวคิดดังนี้

4.1 แผนภาพการทำงานของโปรแกรม

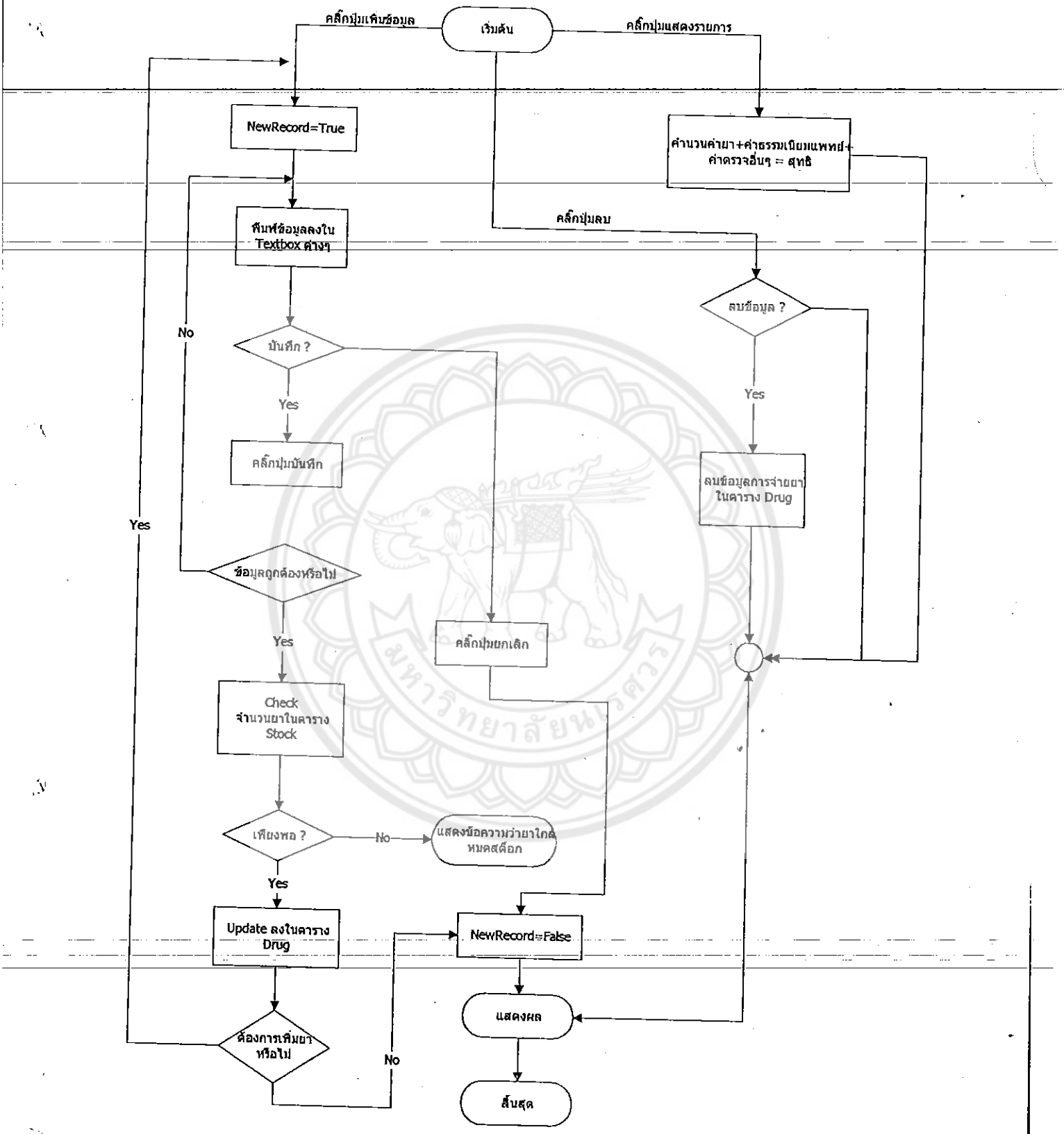


รูปที่ 4.1 โฟลชาร์ท โปรแกรมระบบสต๊อกยา

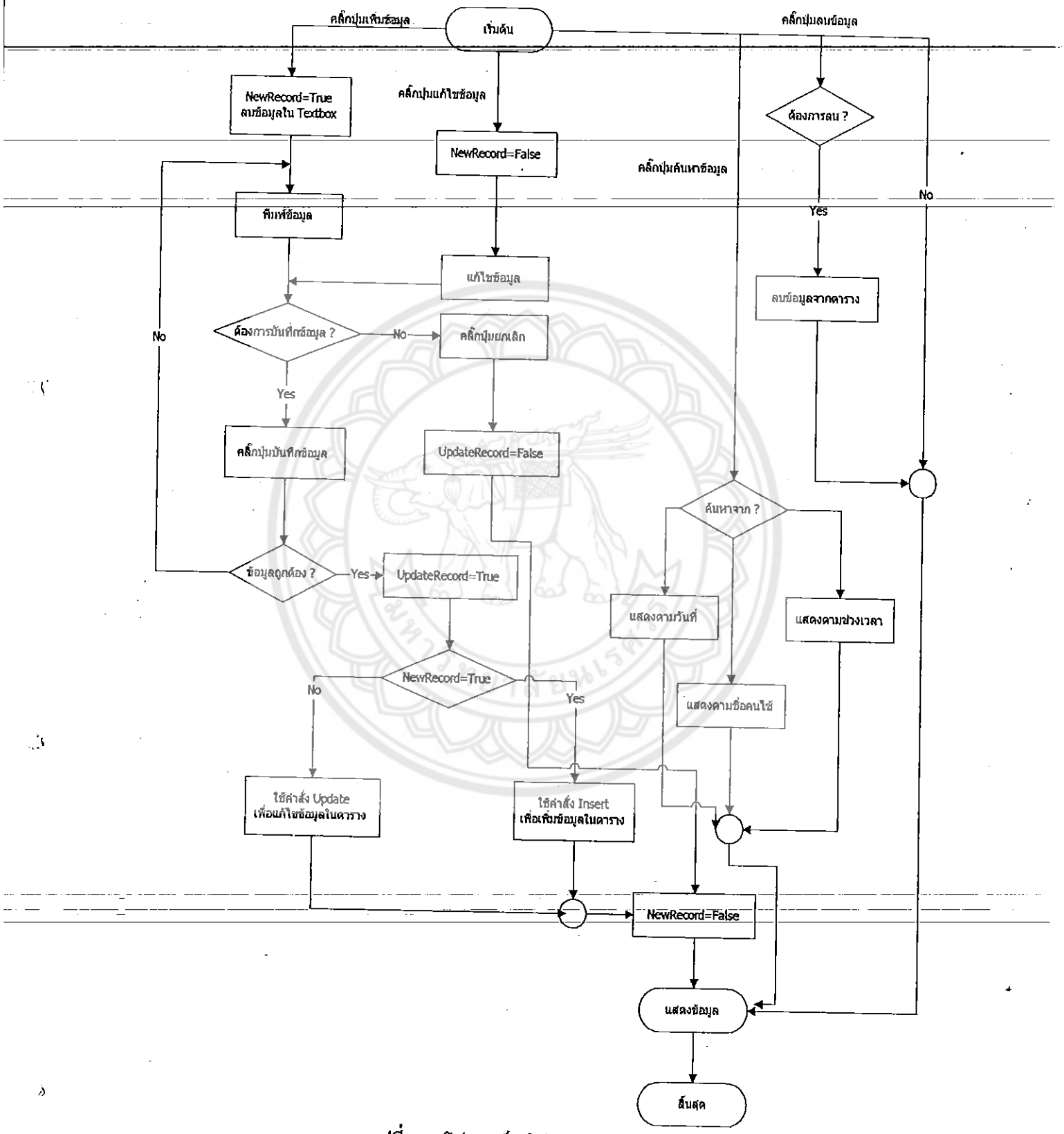
ระบบฐานข้อมูลประวัติคนไข้



รูปที่ 4.2 โฟลชาร์ทโปรแกรมระบบฐานข้อมูลประวัติคนไข้

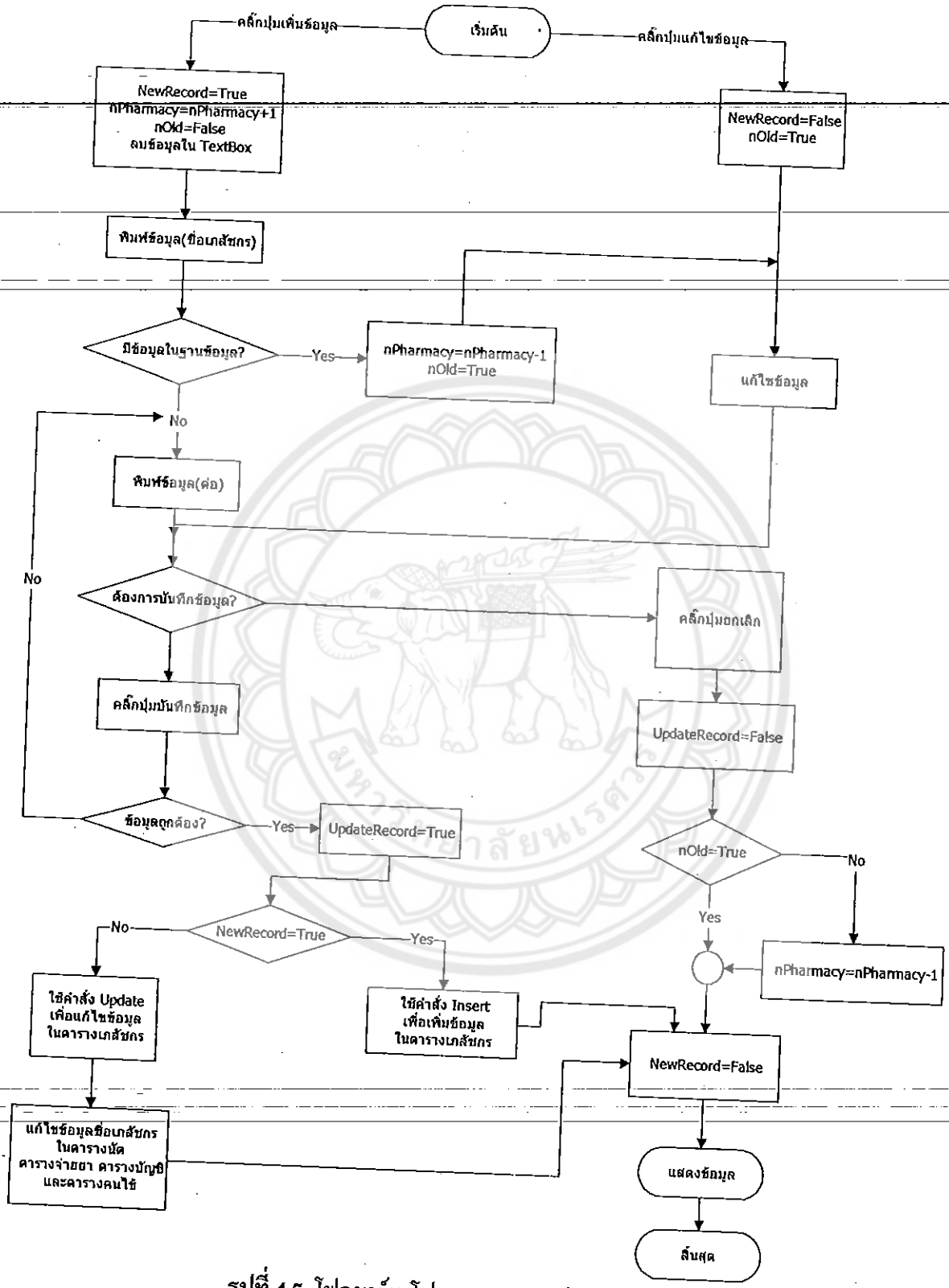


รูปที่ 4.3 โฟลชาร์ท โปรแกรมระบบการจ่ายยา



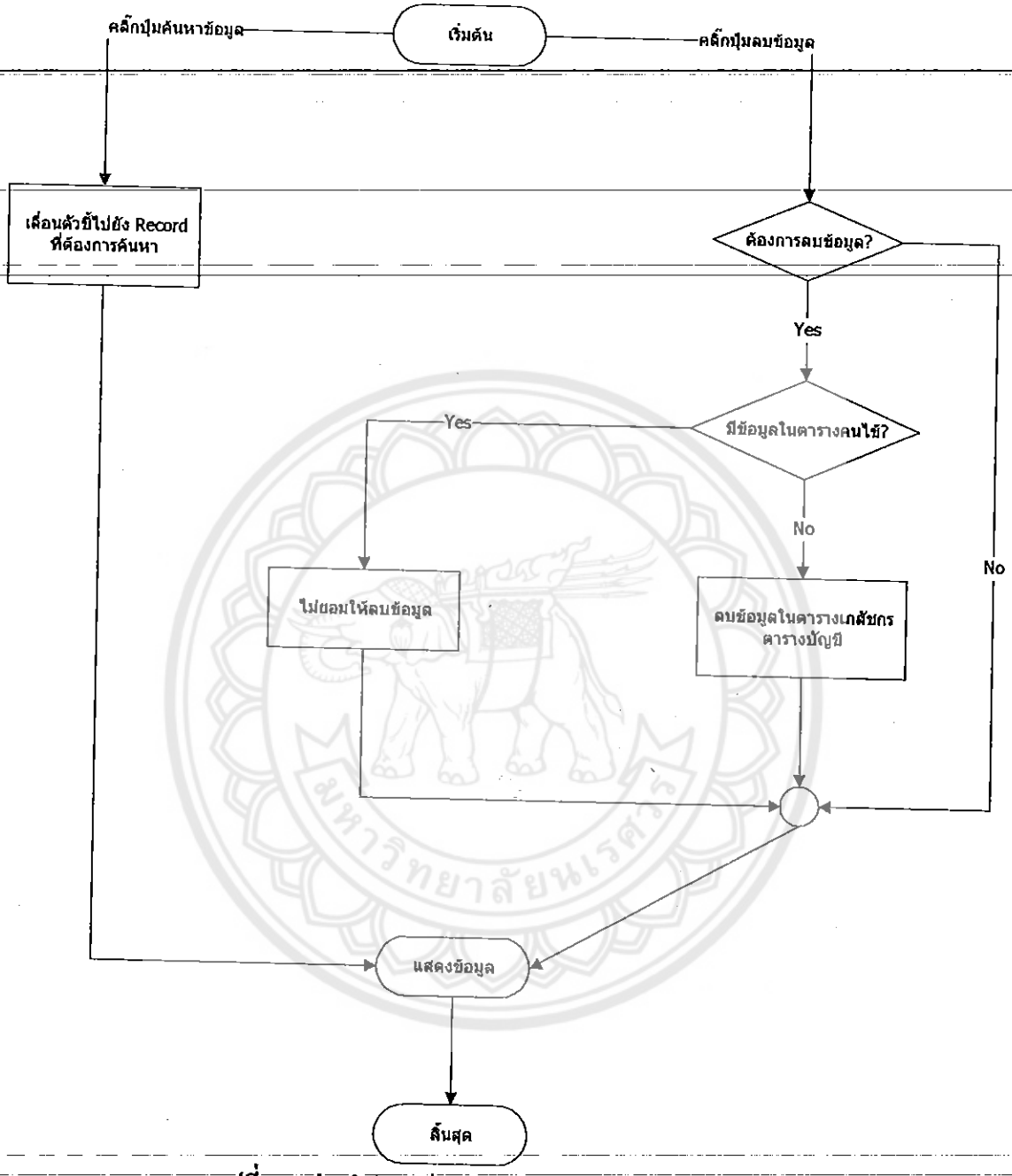
รูปที่ 4.4 โฟลชาร์ท โปรแกรมระบบการนัดหมาย

ระบบประวัติเภสัชกร



รูปที่ 4.5 โฟลชาร์ท โปรแกรมระบบประวัติเภสัชกร

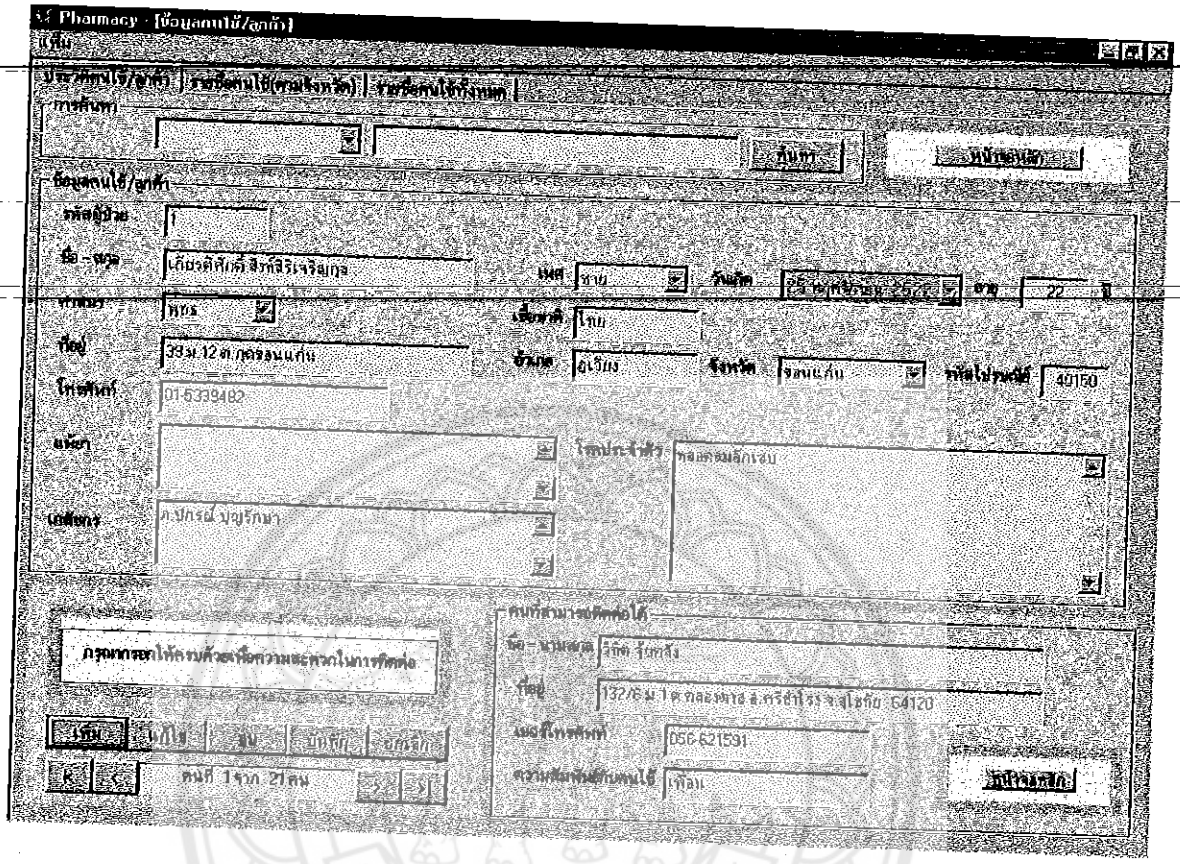
ระบบประวัติเภสัชกร (ต่อ)



รูปที่ 4.5(ต่อ) โฟลชาร์ท โปรแกรมระบบประวัติเภสัชกร

4.2 รูปแบบของโปรแกรมระบบการจัดการฐานข้อมูลฝ่ายเภสัชกรรม

ระบบประวัติคนไข้



รูปที่ 4.6 ลักษณะของโปรแกรมจัดการฐานข้อมูลประวัติลูกค้าและคนไข้

การจัดเก็บข้อมูลในฟอร์มคนไข้ สิ่งที่จัดเก็บประกอบไปด้วยข้อมูลพื้นฐานของคนไข้ และข้อมูลของญาติ เพื่อที่จะทำให้เราทราบได้ว่าคนไข้มีภูมิลำเนาที่ไหน และสามารถติดต่อได้อย่างไรเมื่อต้องการแจ้งผลการตรวจ รายละเอียดการทำงานของฟอร์มคนไข้มีดังนี้

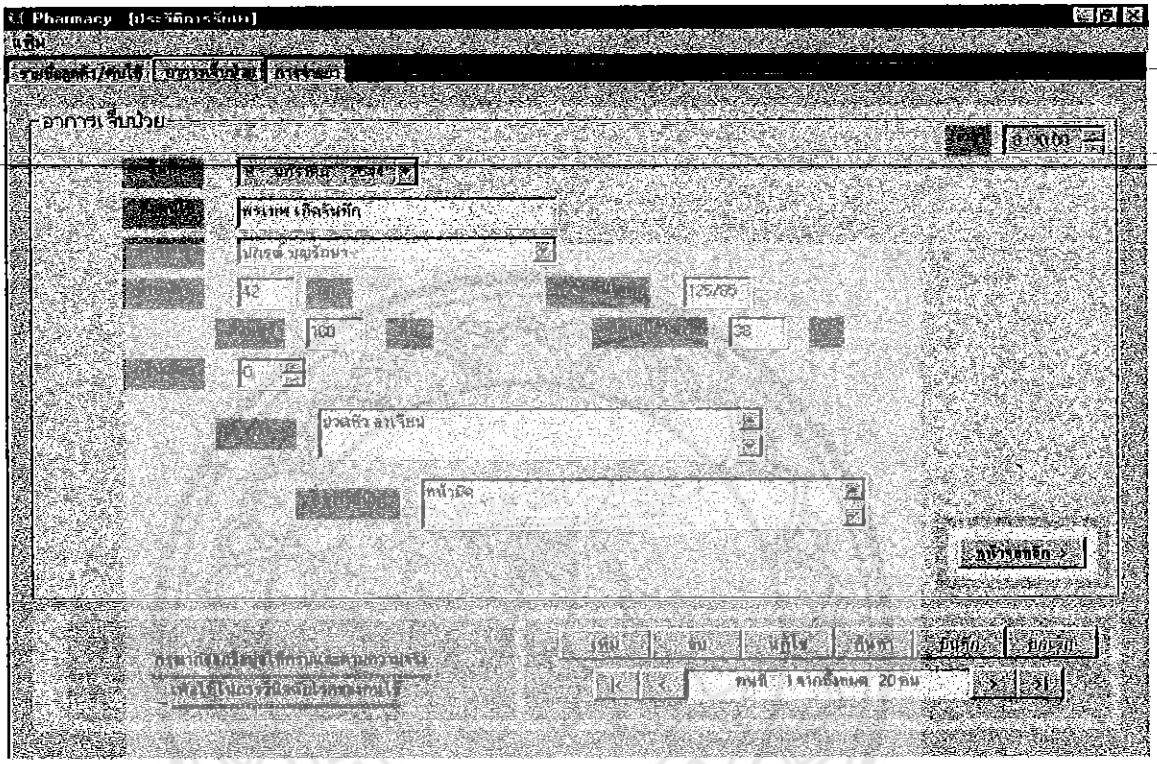
- การเพิ่มข้อมูล หลังจากกดปุ่มเพิ่มข้อมูล ช่องที่แสดงผลข้อมูล (Text Box) จะกลายเป็นช่องว่าง เพื่อที่จะคอยรับค่าข้อมูลที่ป้อนเข้าไป และปุ่มคอนโทรลต่างๆ ก็จะถูกทำให้ไม่ได้ยกเว้นปุ่มบันทึกข้อมูลและปุ่มยกเลิกการแก้ไข หลังจากกรอกข้อมูลเรียบร้อยแล้ว ถ้าต้องการบันทึกข้อมูลก็ทำการกดปุ่มบันทึกข้อมูล ถ้าหากไม่ต้องการบันทึกข้อมูลที่กรอกลงไปก็ให้กดปุ่มยกเลิกการแก้ไข

การแก้ไขข้อมูล ปกติแล้วที่ช่องแสดงข้อความจะไม่สามารถกรอก หรือแก้ไขข้อมูลได้ ดังนั้นเมื่อเราต้องการแก้ไขข้อมูล ต้องทำการกดปุ่มแก้ไขข้อมูลจึงจะสามารถแก้ไขข้อมูลได้

- การลบข้อมูล เมื่อต้องการลบข้อมูลเราต้องทำการเลื่อนพอยน์เตอร์ไปยังเรคคอร์ดที่ต้องการลบที่ (โดยการค้นหาจากช่องค้นหาข้อมูล) หลังจากนั้นก็ทำการกดปุ่มลบข้อมูล โปรแกรมก็จะทำการลบข้อมูลออกจากฐานข้อมูล

- สำหรับการค้นหาข้อมูล เราสามารถค้นหาข้อมูลได้จากชื่อของคนไข้ หรือค้นหาจากรหัสคนไข้ก็ได้

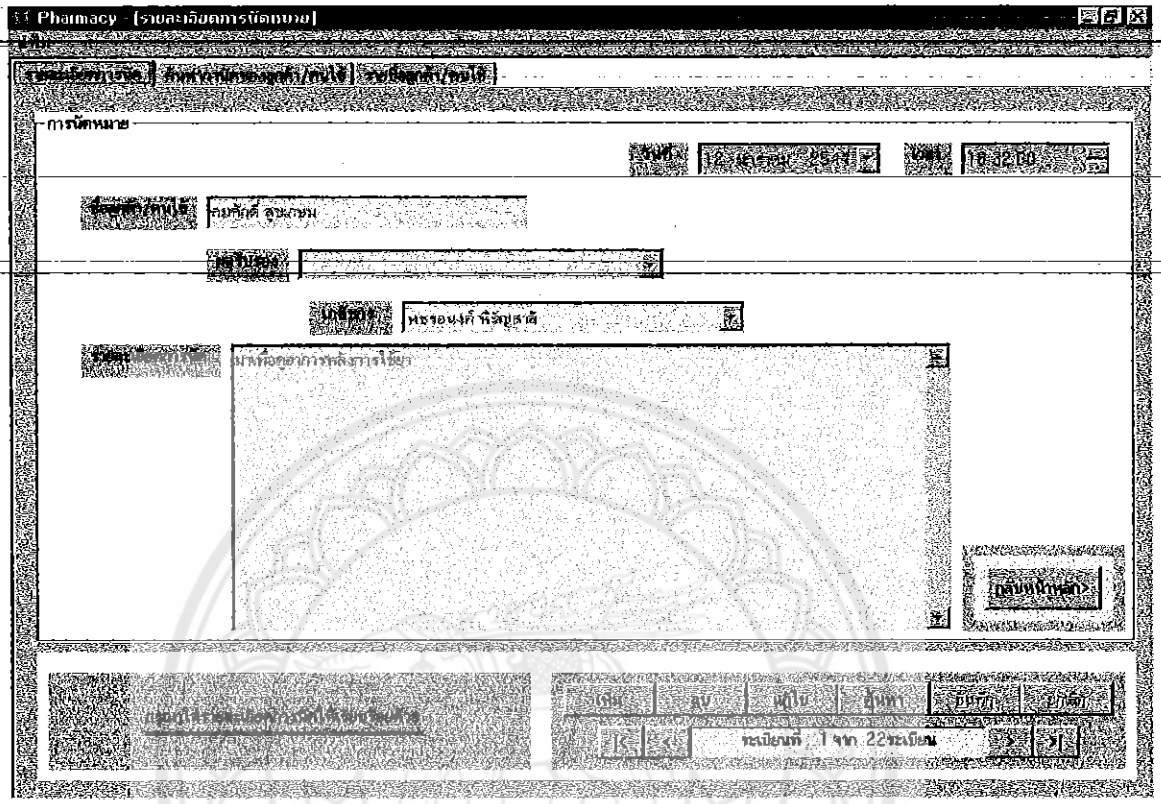
ระบบบันทึกประวัติการรักษา



รูปที่ 4.7 ลักษณะของโปรแกรมจัดการฐานข้อมูลประวัติการรักษา

ลักษณะการทำงานก็คล้ายกันกับฟอร์มคนไข้ แต่การค้นหาข้อมูล เราสามารถดับเบิ้ลคลิ๊กที่ช่องข้อมูลคนไข้ทั้งหมด โดยเราได้ทำการแสดงผลตามตัวอักษรชื่อคนไข้ ซึ่งทำให้สะดวกในการค้นหาข้อมูลอาการเจ็บป่วยครั้งก่อนมาวิเคราะห์อาการเจ็บป่วยครั้งล่าสุดรวมทั้งการจ่ายยาให้กับคนไข้เพื่อเป็นแนวทางในการตรวจในตรวจที่ถูกต้อง

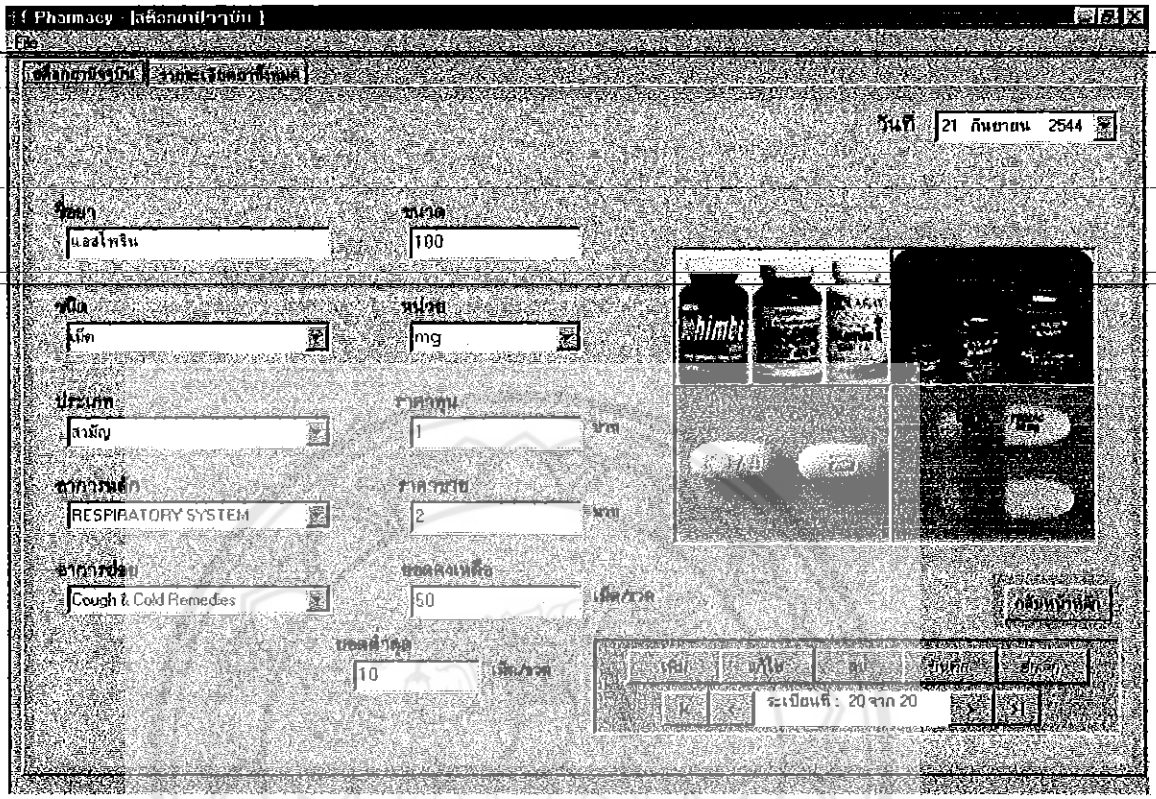
ระบบนัดหมาย



รูปที่ 4.8 ลักษณะของโปรแกรมจัดการฐานข้อมูลนัดหมาย

ฟอร์มนัดผู้ป่วย จัดทำขึ้นเพื่อที่จะสะดวกในการค้นหาเรียกดูข้อมูลว่า วันใด เวลาใดนัดผู้ป่วยมาทำอะไร และใครเป็นผู้นัด ทำให้การติดต่อหรือการทำงานของระบบคลินิกเพื่อติดตามผลของอาการหลังการให้ยา การทำงานของฟอร์มนี้เหมือนกับฟอร์มประวัติการรักษา โดยการค้นหาสามารถค้นหาโดยเลือกช่วงเวลาวัน วันที่นัด หรือจากการกรอกชื่อของคนไข้ได้

ระบบสต็อกยา



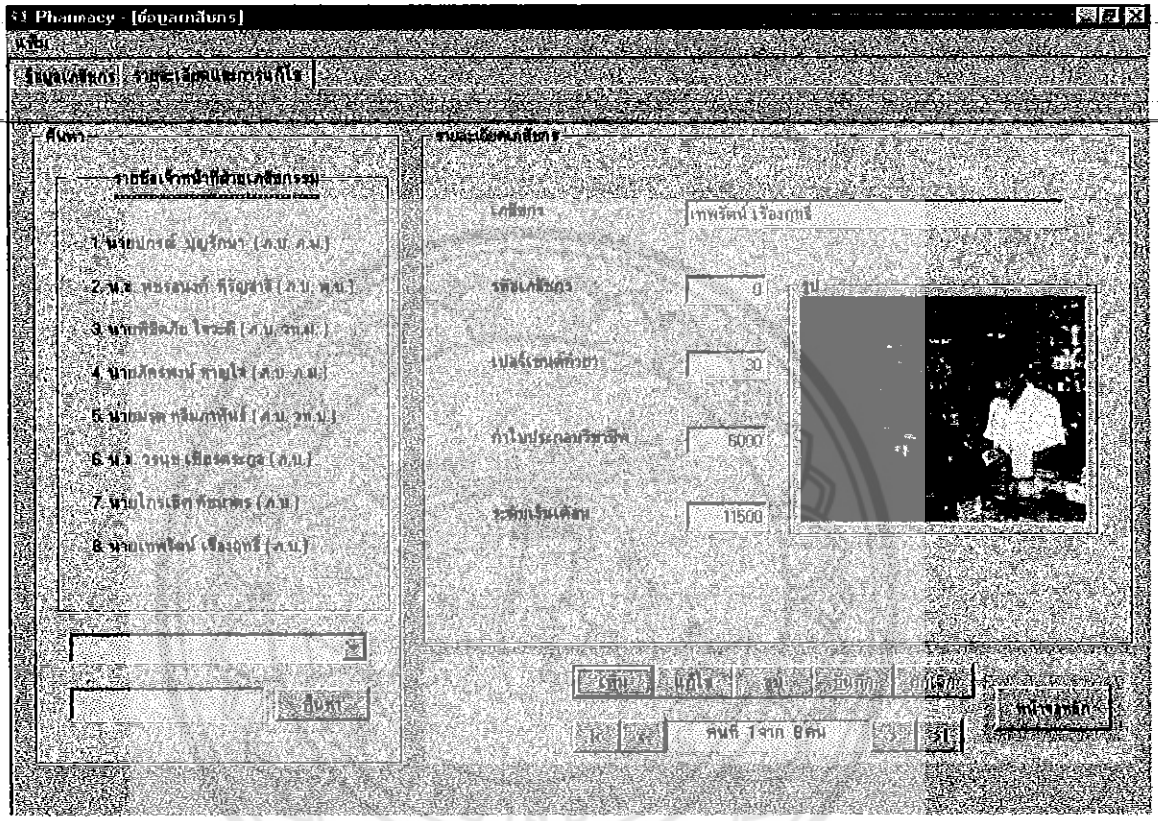
รูปที่ 4.9 ลักษณะของโปรแกรมจัดการฐานข้อมูลสต็อกยา

ฐานข้อมูลระบบสต็อกยาจะเก็บข้อมูลต่างๆ คือ วันที่ ชื่อยา ประเภท กลุ่มอาการ กลุ่มอาการย่อย ราคาทุน ราคาขาย จำนวนคงเหลือ และจำนวนน้อยสุดที่จะทำการสั่งยา การจัดเก็บข้อมูลลงในฐานข้อมูลสต็อกยานี้ทำได้ โดยให้ผู้ใช้กรอกข้อมูลลงใน TextBox หรือ ComboBox ที่ระบุชื่อของข้อมูลอยู่ข้างหน้า เมื่อต้องการเพิ่มข้อมูลลงไปในระบบสต็อกยาให้ผู้ใช้คลิกปุ่ม "เพิ่ม" แล้ว TextBox และ ComboBox ต่างๆ จะว่างพร้อมที่จะพิมพ์ข้อมูลลงไปได้ผู้ใช้ทำการพิมพ์ข้อมูลต่างๆ ของแต่ละหัวข้อได้เมื่อกรอกข้อมูลครบเรียบร้อยแล้วจึงทำการคลิกปุ่ม "บันทึก" จะเป็นการนำเอาข้อมูลที่ผู้ใช้กรอกไปเก็บลงในฐานข้อมูล ก่อนที่ผู้ใช้จะคลิกปุ่ม "บันทึก" ถ้าผู้ใช้ต้องการยกเลิกการเพิ่มข้อมูลก็ให้คลิกปุ่ม "ยกเลิก" ได้ จะปรากฏ MessageBox ถ้าผู้ใช้ว่าต้องการที่จะยกเลิกหรือไม่ เมื่อผู้ใช้ต้องการแก้ไขข้อมูลที่มีอยู่ก่อนแล้วให้ผู้ใช้คลิกปุ่ม "แก้ไข" TextBox และ ComboBox ต่างๆ จะอยู่ในสถานะพร้อมที่แก้ไขข้อมูล เมื่อแก้ไขข้อมูลเรียบร้อยแล้วจึงทำการคลิกปุ่ม "บันทึก" ผู้ใช้ต้องการค้นหาข้อมูลให้ผู้ใช้คลิกปุ่ม "ค้นหา" แล้วจึงทำการเลือกหัวข้อที่ต้องการค้นหา แล้วพิมพ์สิ่งที่ต้องการค้นหาลงไป เมื่อผู้ใช้คลิกปุ่ม "บันทึก" โปรแกรมจะมีการตรวจสอบ

รูปแบบข้อมูลไม่ถูกต้องจะมี MessageBox แสดงต่อผู้ใช้ว่าที่ใดที่ผู้ใช้กรอกข้อมูลผิดพลาด เพื่อความถูกต้องของข้อมูลในฐานข้อมูล

ระบบข้อมูลเภสัชกร

ลักษณะของหน้าจอของระบบฐานข้อมูลเภสัชกรมีลักษณะดังนี้

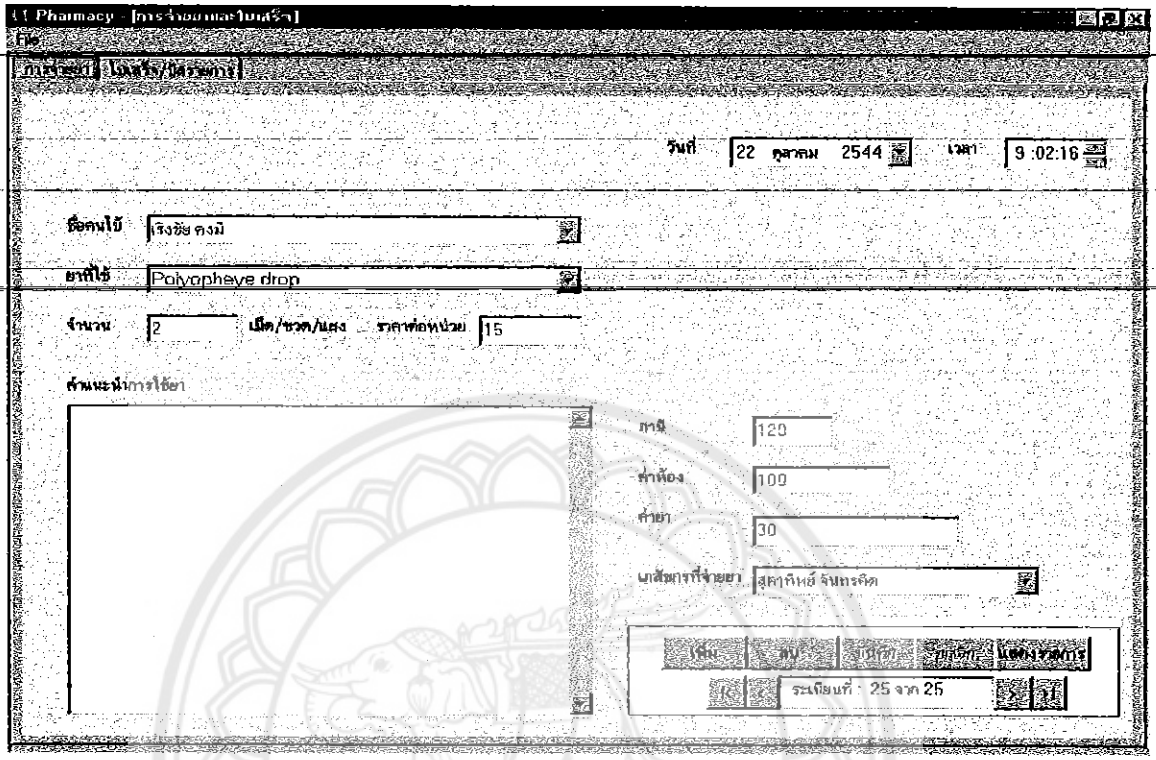


รูปที่ 4.10 ลักษณะของโปรแกรมจัดการฐานข้อมูลเภสัชกร

ระบบฐานข้อมูลเภสัชกรจะมีการจัดเก็บข้อมูล รหัสเภสัชกรชื่อ-นามสกุลของเภสัชกร การจัดเก็บข้อมูลลงในฐานข้อมูลจะคล้ายกับระบบสต็อกยา คือ เมื่อต้องการเพิ่มข้อมูลของเภสัชกร ผู้ใช้ก็คลิกปุ่ม “เพิ่ม” และต้องการแก้ไขข้อมูล ผู้ใช้จึงทำการคลิกปุ่ม “แก้ไข” แล้วจึงทำการแก้ไขข้อมูล ถ้าต้องการยกเลิกการเพิ่มหรือแก้ไข้ผู้ใช้สามารถคลิกปุ่ม “ยกเลิก” เมื่อทำการเพิ่มหรือแก้ไขข้อมูลเรียบร้อยแล้วจึงทำการคลิกปุ่ม “บันทึก” เมื่อต้องการลบข้อมูลผู้ใช้ก็ทำการคลิกปุ่ม “ลบ” ผู้ใช้ต้องทำการค้นหาข้อมูลให้ผู้ใช้คลิกปุ่ม “ค้นหา” แล้วจึงทำการเลือกหัวข้อที่ต้องการค้นหา แล้วพิมพ์สิ่งที่ต้องการค้นหาลงไป เมื่อผู้ใช้คลิกปุ่ม “บันทึก” โปรแกรมจะมีการตรวจสอบรูปแบบของข้อมูลใน TextBox เมื่อมีส่วนใดที่รูปแบบของข้อมูลไม่ถูกต้อง จะมี MessageBox แสดงต่อผู้ใช้ว่าที่ใดที่ผู้ใช้กรอกข้อมูลผิดพลาด เพื่อความถูกต้องของข้อมูลในฐานข้อมูล

ระบบฐานข้อมูลการจ่ายยา

รูปแบบของหน้าจอของระบบฐานข้อมูลการจ่ายยามีดังนี้



รูปที่ 4.11 ลักษณะของโปรแกรมจัดการฐานข้อมูลประวัติการจ่ายยา

ฐานข้อมูลการจ่ายยาจะเก็บประวัติการใช้ยาของคนไข้ จะเก็บข้อมูล วันที่จ่ายยา เวลา ชื่อคนไข้ รายละเอียดการใช้ยา ชื่อเภสัชกร จำนวน ราคาต่อหน่อย ค่าธรรมเนียมเภสัชกร ค่าตรวจรักษาอื่นๆ ยาที่ใช้เมื่อต้องการเพิ่มข้อมูลของเภสัชกร ผู้ใช้ก็คลิกปุ่ม "เพิ่ม" และถ้าคนไข้ใช้ยามากกว่า 1 ราย การเมื่อต้องการเพิ่มรายการยาให้กับคนไข้คนปัจจุบันที่กำลังทำการแก้ไข ให้ผู้ใช้คลิกปุ่ม "เพิ่มรายการยา" โปรแกรมจะให้ผู้ใช้เพิ่มรายการเฉพาะ รายละเอียดการใช้ยา ชื่อยา และจำนวน และ ราคาต่อหน่วย และต้องการแก้ไขข้อมูลผู้ใช้จึงทำการคลิกปุ่ม "แก้ไข" แล้วจึงทำการแก้ไขข้อมูล ถ้าต้องการยกเลิกการเพิ่มหรือแก้ไขผู้ใช้สามารถคลิกปุ่ม "ยกเลิก" เมื่อทำการเพิ่มหรือแก้ไขข้อมูลเรียบร้อยแล้วจึงคลิกปุ่ม "บันทึก" เมื่อต้องการลบข้อมูลผู้ใช้ก็ทำการคลิกปุ่ม "ลบ" ผู้ใช้ต้องการค้นหาข้อมูลให้ผู้ใช้คลิกปุ่ม "ค้นหา" แล้วจึงทำการเลือกหัวข้อที่ต้องการค้นหา แล้วพิมพ์สิ่งที่ต้องการค้นหาลงไป เมื่อผู้ใช้คลิกปุ่ม "บันทึก" โปรแกรมจะมีการตรวจสอบรูปแบบของข้อมูลใน TextBox และ ComboBox ต่างๆ ถ้ามีข้อมูลส่วนใดที่รูปแบบของข้อมูลไม่ถูกต้อง จะมี MessageBox แสดงต่อผู้ใช้ว่าที่ใดที่ผู้ใช้กรอกข้อมูลผิดพลาด เพื่อความถูกต้องของข้อมูลในฐานข้อมูล

บทที่5

บทสรุป

5.1 สรุปผล

1. ฐานข้อมูลประวัติคนไข้จะเก็บข้อมูลเกี่ยวกับประวัติพื้นฐานของคนไข้ รวมถึงข้อมูลของญาติของคนไข้คนนั้น ในส่วนของโปรแกริมผู้ใช้สามารถเพิ่มข้อมูล ลบข้อมูล แก้ไขข้อมูลประวัติคนไข้ได้ และสามารถค้นหาข้อมูล ตามชื่อ รหัส จังหวัด ของคนไข้ได้

2. ฐานข้อมูลประวัติการรักษาจะเก็บข้อมูลอาการของคนไข้ และการจ่ายยาของเภสัชกรไว้ เพื่อใช้เรียกดูประกอบการตัดสินใจของเภสัชกรในการจ่ายยาครั้งต่อไปเมื่อคนไข้มาพบเภสัชกรอีกครั้ง ในส่วนของโปรแกรมผู้ใช้สามารถเพิ่มข้อมูล ลบข้อมูล แก้ไขข้อมูลประวัติการรับยาของคนไข้ได้ และสามารถค้นหาข้อมูลตามชื่อคนไข้ วันที่ที่มารักษาได้

3. ฐานข้อมูลการนัดหมาย จะเก็บข้อมูลการนัดในแต่ละครั้งเมื่อคนไข้ได้รับการนัดจากเภสัชกรที่จ่ายยาให้เพื่อใช้เรียกดูว่ามีการนัดเมื่อไร รายละเอียดการนัด ในส่วนของโปรแกรมผู้ใช้สามารถเพิ่มข้อมูล ลบข้อมูล แก้ไขข้อมูลการนัดได้ และสามารถค้นหาข้อมูลตามชื่อคนไข้ วันที่ที่นัดได้

4. ฐานข้อมูลสต็อกยา จะเก็บข้อมูลยาทั้งหมดที่มีอยู่ สามารถเตือนได้ว่ายาใกล้หมดสต็อกแล้วหรือยัง ในส่วนของโปรแกรมผู้ใช้สามารถเพิ่มข้อมูล ลบข้อมูล แก้ไขข้อมูลสต็อกยาได้และสามารถค้นหาข้อมูลตามชื่อยา ประเภท ชนิดของยาได้

5.2 ประเมินผลและข้อเสนอแนะ

จากผลของโครงงานเมื่อเปรียบเทียบกับวัตถุประสงค์ที่ตั้งไว้ได้ผลว่า

1.จากการดำเนินงานที่ผ่านมาสามารถออกแบบฐานข้อมูลได้ตามโครงสร้างฐานข้อมูลในบทที่3

2.เครื่องมือที่ใช้ในการพัฒนาโปรแกรมระบบการจัดการฐานข้อมูลฝ่ายเภสัชกรรม ได้แก่ โปรแกรม Microsoft Visual Basic 6.0 ในการเขียนโปรแกรม และใช้ Microsoft Access 97 เป็นฐานข้อมูลของระบบ เนื่องจากโปรแกรม Microsoft Access 97เป็นฐานข้อมูลที่ใช้กันอย่างแพร่หลายและเหมาะสมกับระบบภาษาไทยในระบบปฏิบัติการ Microsoft Windows 95,98 และโปรแกรม Microsoft Visual Basic 6.0 เป็นเครื่องมือที่ใช้พัฒนาโปรแกรมในด้านฐานข้อมูลที่ง่ายต่อการเข้าใจ และมีเครื่องมือที่เกี่ยวกับฐานข้อมูลให้ใช้ได้เหมาะสมกับการพัฒนาระบบฐานข้อมูลแบบ Stand Alone

3.โปรแกรมระบบการจัดการฐานข้อมูลฝ่ายเภสัชกรรม ที่ได้พัฒนาขึ้นประกอบด้วยส่วนของโปรแกรมที่จัดการฐานข้อมูลต่างๆดังนี้

1.1 ฐานข้อมูลประวัติคนไข้

1.2 ฐานข้อมูลประวัติการรักษา

1.3 ฐานข้อมูลการนัดหมาย

1.4 ฐานข้อมูลการจ่ายยา

1.5 ฐานข้อมูลระบบสต็อกยา

5.3 ปัญหา ข้อเสนอแนะ และแนวทางการแก้ไข

1. โครงงานวิศวกรรม "โปรแกรมระบบการจัดการฐานข้อมูลฝ่ายเภสัชกรรม" อยู่บนพื้นฐานของการออกแบบฐานข้อมูลฝ่ายเภสัชกรรมจริง โครงสร้างของฐานข้อมูลที่ออกแบบมานั้นอาจจะไม่สมบูรณ์ตามทฤษฎี เพราะมีข้อจำกัดหลายๆ เช่น การคิดค่ายาจะยังสามารถคิดได้ทีละตัวยา และยังไม่ได้สร้างการหมดอายุของยาเป็นต้น

2. โปรแกรมที่ได้สามารถทำงานได้ตามจุดประสงค์ที่ตั้งไว้ แต่ว่าในส่วนติดต่อกับผู้ใช้ การใช้งานอาจจะใช้งานได้ไม่สะดวกมากนัก

เอกสารอ้างอิง

[1] กิตติ ภักดีวัฒนะกุล, จำลอง ครูอุตสาหะ. **คัมภีร์ระบบฐานข้อมูล.พิมพ์ครั้งที่2** กรุงเทพฯ: ไทยเจริญการพิมพ์, หจก. 2542.

[2] กิตติ ภักดีวัฒนะกุล, จำลอง ครูอุตสาหะ. **Visual Basic 6 ฉบับโปรแกรมเมอร.** พิมพ์ครั้งที่1 กรุงเทพฯ: ไทยเจริญการพิมพ์, หจก. 2542.

[3] กิตติ ภักดีวัฒนะกุล, จำลอง ครูอุตสาหะ. **Visual Basic 6 ฉบับฐานข้อมูล.** พิมพ์ครั้งที่1 กรุงเทพฯ ไทยเจริญการพิมพ์, หจก. 2542.

[4] ดวงแก้ว สวามิภักดิ์. ระบบฐานข้อมูล. กรุงเทพฯ : ซีเอ็ดยูเคชั่น.จำกัด,บริษัท. 2534.

[5] อำไพ พรประเสริฐสกุล. การวิเคราะห์และออกแบบระบบ. พิมพ์ครั้งที่3 กรุงเทพฯ:ศูนย์เทคโนโลยีอิเล็กทรอนิกส์และคอมพิวเตอร์แห่งชาติ.

[6] ธาริน สิทธิธรรมชารี, สุรสิทธิ์ คิวประสพศักดิ์. **Advanced Microsoft Visual Basic 6.0.** พิมพ์ครั้งที่3 กรุงเทพฯ:ส.เอเซียเพรส(1989)จำกัด,บริษัท.

[7] ฉัททวุฒิ พืชผล, พิชิต สันติกุลานนท์. คู่มือเรียน **Visual Basic.**พิมพ์ครั้งที่1 กรุงเทพฯ:เอช เอ็นกรุ๊ป จำกัด,บริษัท. 2542.

ภาคผนวก ก

รูปแบบการติดต่อกับฐานข้อมูลด้วย Visual Basic

ในการติดต่อกับฐานข้อมูล โดยปกติแล้ว VB จะเชื่อมโยงผ่านทาง Database Engine ที่เรียกว่า JET Engine จึงอาจกล่าวได้อีกนัยหนึ่งว่า JET Engine คือ ไดรเวอร์ชนิดหนึ่ง ซึ่งทำหน้าที่เป็นตัวเชื่อมโยงให้ VB สามารถติดต่อกับฐานข้อมูลได้นั่นเอง โดยที่ฐานข้อมูลหลัก (Default) ที่ VB รู้จักเป็นอย่างดีก็คือ MS Access แต่ Visual Basic สามารถติดต่อกับฐานข้อมูลได้ทุกชนิดเช่นกัน โดยอาศัยเทคโนโลยีหลายๆอย่าง

สำหรับฐานข้อมูลที่จะนำเสนอ จะใช้ฐานข้อมูลของ MS Access 2000 Thai Edition เป็นหลัก ดังนั้นถ้าคุณสามารถใช้งาน MS Access 2000 ได้บ้าง ก็จะทำให้การสร้างแอพพลิเคชันด้านฐานข้อมูลด้วย Visual Basicรวดเร็วขึ้น โดยที่จะกล่าวถึงวิธีการใช้งาน MS Access 2000 เท่าที่เกี่ยวข้อง

การติดต่อฐานข้อมูลใน Visual Basic จะแยกออกเป็น 4 ประเภทใหญ่ๆ คือ

1. แบบที่1-ติดต่อโดยอาศัยคอนโทรลด้านฐานข้อมูล

มีศัพท์เรียกคอนโทรลกลุ่มนี้โดยเฉพาะว่า Bound Controls ส่วนใหญ่แล้วก็คือ กลุ่มของคอนโทรลมาตรฐานที่ใช้กันโดยทั่วไปนั่นเอง เช่น คอนโทรล TextBox, PictureBox, Image, ListBox, ComboBox เป็นต้น โดยใช้คอนโทรล Data (Data Control) เป็นตัวเชื่อมโยงระหว่างฐานข้อมูลกับกลุ่ม Bound Controls

คุณสามารถตรวจสอบได้ว่า คอนโทรลตัวใดบ้างถูกจัดอยู่ในกลุ่มของ Bound Controlsโดยการตรวจสอบว่าคอนโทรลตัวนั้น มีคุณสมบัติที่ขึ้นต้นด้วยคำว่าData...เช่น DataField, Dateformat, DataSource หรือไม่ ถ้ามีหมายถึง คอนโทรลตัวดังกล่าว ถูกจัดอยู่ในกลุ่มของ Bound Controls ด้วยเช่นกัน

ยังมีคอนโทรลอีกกลุ่มหนึ่งที่เรียกว่า ActiveX Bound Controls หมายถึง กลุ่มของคอนโทรลที่มีคุณสมบัติที่ขึ้นต้นด้วยคำว่า Data... เช่นกัน แต่มีข้อแตกต่างจากคอนโทรลในกลุ่ม Bound Controls ก็คือกลุ่ม ActiveX Bound Controls อาจจะมาจากผู้ผลิตรายอื่นๆ(Thirds Party) ซึ่งเป็นคอนโทรลที่ไม่ได้อยู่ในแถบเครื่องมือมาตรฐานของ Visual Basic โดยที่คุณต้องเพิ่มเติมคอนโทรลกลุ่มนี้เข้ามาในแถบเครื่องมือเอง ดังนั้นจึงเรียกคอนโทรลกลุ่มนี้ว่าActiveX Bound Controls เช่น คอนโทรล DBCombo ฯลฯ เป็นต้น

2. แบบที่ 2 –ติดต่อโดยใช้ออบเจ็กต์ Data Access Object (DAO)

ถือว่าเป็นวิธีที่ล้าสมัยแล้ว โดยมีแนวคิดในการติดต่อหรือเข้าถึงข้อมูลในฐานข้อมูลผ่านทางองค์ประกอบต่างๆในฐานข้อมูล เช่น ฟิลด์ (Field), เร็คคอร์ด (Record), ความสัมพันธ์ระหว่างตาราง (Relation) เป็นต้น โดยจะแทนแต่ละองค์ประกอบเหล่านั้นด้วยออบเจ็กต์ (Object) และควบคุมออบเจ็กต์ต่างๆเหล่านี้ด้วยการเขียนโค้ด แม้จะทำงานได้ดีกว่า อิสระกว่า แต่มีความยุ่งยากในการเขียนโปรแกรมด้วยเช่นกัน อีกทั้งเป็นเทคโนโลยีที่เก่าแล้วคือ เน้นเฉพาะระบบฐานข้อมูลที่เป็นตาราง (โดยเฉพาะ Access รุ่นเก่าๆ) แต่ว่า

การเก็บข้อมูลในปัจจุบัน ถูกจัดเก็บอยู่ในสภาพแวดล้อมแตกต่างกันมากมาย เช่น รูปภาพ(Image),ข้อความ (Text)และรูปแบบอื่นๆอีกมากมาย ทำให้ต้องสร้างออบเจ็กต์ใหม่ๆขึ้นมาเรื่อยๆ แต่นั่นไม่ใช่สิ่งที่ทำได้ง่าย และกลายเป็นข้อจำกัดที่สำคัญของ DAO ตัวอย่างของออบเจ็กต์ในกลุ่มนี้ เช่น ออบเจ็กต์ RecordSet, ออบเจ็กต์ TableDef, คอลเล็กชัน Fields เป็นต้น

3. แบบที่ 3 – ติดต่อผ่าน ODBC โดยตรง

เป็นการติดต่อกับฐานข้อมูลแบบ32บิตที่สนับสนุนมาตรฐาน ODBC(Open DataBase Connectivity) ที่ JET Engine (กลไกการติดต่อฐานข้อมูลแบบไมโครซอฟท์ ซึ่งเน้นที่ Access) ไม่สามารถจัดการได้ เช่น ฐานข้อมูลของ Oracle ,ฐานข้อมูลของ Microsoft SQL Server เป็นต้น ซึ่งเป็นการติดต่อเฉพาะฐานข้อมูลที่มีการเก็บข้อมูลในรูปแบบตารางเท่านั้น

ไมโครซอฟท์เองได้สร้างออบเจ็กต์ขึ้นมาอีกชุดหนึ่งที่ชื่อว่า Remote Data Object หรือเรียกสั้นๆว่า RDO เพื่อใช้สำหรับติดต่อกับฐานข้อมูลแบบเครือข่าย (เช่น Oracle, SQL Server, DB2, ฯลฯ) ซึ่งเป็นไปตามสถาปัตยกรรม n-Tier เช่น Client/Server (2-tier) หรือ Middle Tier (3-Tier) โดยอาศัยมาตรฐาน ODBC ในการเชื่อมโยงที่เกี่ยวข้องกับ Visual Basic คือ

- คอนโทรล Remote Data (Remote Data Control - RDC)
- ออบเจ็กต์ Remote Data (Remote Data Object - RDO)

4. แบบที่4 – เข้าถึงข้อมูลโดยอาศัยเทคโนโลยี OLEDB

เป็นรูปการติดต่อกับฐานข้อมูลผ่านทางกลุ่มออบเจ็กต์ในโมเดล ADO ซึ่งใช้ OLEDB Provider เป็นกลไกในการเข้าถึงข้อมูลในฐานข้อมูลแทน JET Engine โดยเปลี่ยนจากมุมมองการติดต่อฐานข้อมูลแบบเดิมๆที่ต้องกำหนดชนิดของฐานข้อมูลมาเป็นมองที่รูปแบบของการเชื่อมต่อ (Connection) เข้ากับฐานข้อมูล OLEDB เป็นเทคโนโลยีล่าสุดที่ไมโครซอฟท์วางตำแหน่งให้เป็นยุคต่อไป ในการเข้าถึงข้อมูลในฐานข้อมูลที่มีรูปแบบการเก็บข้อมูลสารพัดรูปแบบ ซึ่งไม่ได้จำกัดตัวเองเฉพาะตารางโดย OLEDB เป็นส่วนหนึ่งของสถาปัตยกรรม Universal Data Access (UDA) ของไมโครซอฟท์

ใน Visual Basic สามารถใช้ OLEDB ได้2แบบคือ โดยการใช้

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

ภาคผนวก ข

การเขียนโปรแกรมกับฐานข้อมูลด้วย Data Control

1. Data Control ใน Visual Basic 6

Visual Basic จะอาศัย Control ชื่อ "Data" ในการทำงานร่วมกับ Database โดยที่ Data Control จะใช้อ้างถึง 1 Table หรือมากกว่าในฐานข้อมูล ข้อมูลที่ถูกอ่านจาก Table มาเก็บไว้ใน Data Control เรียกว่า "Recordset" ดังนั้นจึงอาจกล่าวได้ว่า Recordset ได้แก่ กลุ่มของ Record ใน Table ซึ่งอาจมาจาก 1 หรือมากกว่า 1 Table ที่ถูกอ่านเข้ามาเก็บไว้ในตัว Data Control ตามเงื่อนไขที่กำหนด ในการอ้างถึง Recordset ใน Data Control ให้เขียนอยู่ในรูปแบบนี้

Datacontrol.Recordset.*methodproperty*

โดยที่ Data Control หมายถึง ชื่อของ Data Control

methodproperty หมายถึง Method หรือ property ของ Data Control

ในการทำงานเกี่ยวกับ Data Control จะต้องอาศัย Property ต่างๆดังนี้

1. Connect ใช้กับกำหนดประเภทของฐานข้อมูล เช่น Base,Foxpro,Paradox และ Textfile ทั่วๆไป
2. DatabaseName ใช้สำหรับกำหนด Path และชื่อของ Database
3. RecordType ใช้สำหรับกำหนดประเภทของ Recordset ประกอบด้วย
   - 3.1 Table เป็น Recordset ซึ่งกระทำกับ Table เพียง Table เดียว
   - 3.2 Dynaset เป็น Recordset ซึ่งกระทำกับ Table ตั้งแต่ 1 Table ขึ้นไป สามารถแก้ไขข้อมูลได้
   - 3.3 Snapshot เป็น Recordset ซึ่งกระทำกับ Table ตั้งแต่ 1 Table ขึ้นไป แต่ไม่สามารถแก้ไขข้อมูลได้ จะใช้แสดงผลข้อมูลอย่างเดียว
4. RecordSource ใช้สำหรับกำหนดชื่อของ Table
5. BOFAction ใช้กำหนดการทำงานให้กับ Data Control เมื่อเกิดสถานะ Begin of File
   - เป็น "0-MoveFirst" เพื่อเลื่อนพอยเตอร์ไป Record แรก

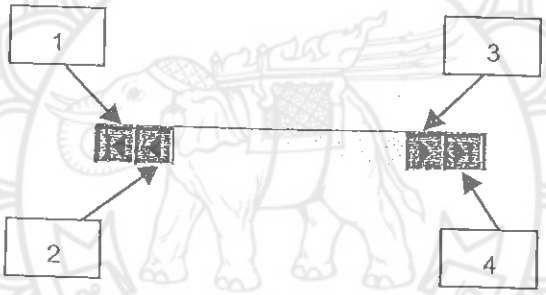
- เป็น "1-BOF" เพื่อกำหนดค่าของ property "BOF" เป็น True ทำให้ Data-Control ใช้ Method MoveLast ไม่ได้

6. EOFAction ใช้กำหนดการทำงานให้กับ DataControl เมื่อเกิดสถานะ "End of File"

- เป็น "0-MoveLast" เพื่อเลื่อนพอยเตอร์ไป Record สุดท้าย
- เป็น "1-EOF" เพื่อกำหนดค่าของ property "EOF" เป็น True ทำให้ Data Control ใช้ Method MoveNext ไม่ได้
- เป็น "2-AddNew" เพื่อเพิ่ม Record ให้กับ Recordset โดยอัตโนมัติ

7. ReadOnly เป็น Property ที่มีลักษณะข้อมูลแบบตรรกะ ใช้กำหนดให้ Data Control อ่านได้อย่างเดียว เมื่อกำหนดให้มีค่าเป็น True และจะสามารถแก้ไขได้เมื่อกำหนดให้มีค่าเป็น False

Data Control นี้จะอยู่ในรูปแบบที่ประกอบด้วยปุ่มลูกศรจำนวน 4 ปุ่ม ดังรูป ทั้ง 4 ปุ่มนี้จะใช้เลื่อน Pointer ของ Record ดังนี้



- ปุ่มหมายเลข1 ใช้เลื่อน Pointer ไปยัง Record แรกใน Recordset
- ปุ่มหมายเลข2 ใช้เลื่อน Pointer ไปยัง Record ก่อนหน้า Record ปัจจุบัน ใน Recordset
- ปุ่มหมายเลข3 ใช้เลื่อน Pointer ไปยัง Record หลัง Record ปัจจุบัน ใน Recordset
- ปุ่มหมายเลข4 ใช้เลื่อน Pointer ไปยัง Record สุดท้ายในRecordset

2. Method ที่ใช้จัดการข้อมูลของ Data Control

2.1 การเพิ่มข้อมูล

ในการเพิ่มข้อมูลไปยัง Table ต่างๆในฐานข้อมูลจะอาศัย Method "AddNew" เพื่อสร้าง Record ว่างต่อเพิ่มเข้าไปในRecordset

```
Recordset.AddNew
```

โดยที่ Recordset หมายถึง ชื่อ Object ที่เป็นเจ้าของ Recordset ในกรณีที่ใช้ Data Control ใช้กำหนดในรูปแบบ datacontrol.Recordset โดย Data Control ได้แก่ ชื่อ Data Control นั้น

2.2 การแก้ไขข้อมูล

Method ที่ใช้ในการบันทึกข้อมูลที่มีการแก้ไข ได้แก่ Method "Update" โดยจะบันทึกข้อมูลที่ปรากฏอยู่ใน Object ที่เป็น Bound Control กลับลงไปยัง Record ปัจจุบันใน Recordset รูปแบบคำสั่งเป็นดังนี้

```
Recordset.Update
```

โดยที่ Recordset หมายถึง ชื่อ Object ที่เป็นเจ้าของ Recordset ในกรณีที่ใช้ Data Control ใช้กำหนดในรูปแบบ datacontrol.Recordset โดย datacontrol ได้แก่ ชื่อ Data Control นั้น

2.3 การลบข้อมูล

Method ที่ใช้สำหรับลบข้อมูล Record ปัจจุบันออกจาก Recordset ได้แก่ Method "Delete"

```
Recordset.Delete
```

โดยที่ Recordset หมายถึง ชื่อ Object ที่เป็นเจ้าของ Recordset ในกรณีที่ใช้ Data Control ใช้กำหนดในรูปแบบ datacontrol.Recordset โดย data control ได้แก่ ชื่อ data control นั้น

2.4 การค้นหาข้อมูล

Method ที่ใช้ในการค้นหาข้อมูลใน Recordset จะประกอบไปด้วย

1. FindFirst ที่ใช้ในการค้นหาข้อมูลใน Record โดยทิศทางการหาจะเริ่มจาก Record แรกไปยัง Record สุดท้ายใน Recordset
2. FindLast ที่ใช้ในการค้นหาข้อมูลใน Record โดยทิศทางการหาจะเริ่มจาก Record สุดท้ายไปยัง Record แรกใน Recordset
3. FindNext ที่ใช้ในการค้นหาข้อมูลใน Record โดยทิศทางการหาจะเริ่มจาก Record ปัจจุบันไปยัง Record สุดท้ายใน Recordset

4. FindPrevious ที่ใช้ในการค้นหาข้อมูลใน Record โดยทิศทางการหาจะเริ่มจาก Record ปัจจุบันไปยัง Record แรกใน Recordset

```
Recordset.{FindFirst/FindLast/FindNext/FindPrevious}criteria
```

โดยที่ Recordset หมายถึง ชื่อ Object ที่เป็นเจ้าของ Recordset ในกรณีที่ใช้ Data Control ใช้กำหนดในรูปแบบ datacontrol.Recordset โดย Data Control ได้แก่ ชื่อ Data Control นั้น

Criteria หมายถึง ประโยคเงื่อนไขที่ใช้ในการค้นหา

ภาคผนวก ค

การเขียนโปรแกรมกับฐานข้อมูลด้วย Data Access Object (DAO)

การเรียกใช้ข้อมูลด้วย Data Control ถึงแม้ว่าจะค่อนข้างสะดวก แต่อย่างไรก็ตามยังมีข้อจำกัด ดังนั้นจึงมีวิธีที่ได้รับความนิยมมากกว่า Data Control ได้แก่ การใช้ Objectที่ใช้ในการเข้าถึงข้อมูล ที่มีชื่อว่า "Data Access Object" ซึ่งเรียกย่อๆว่า "DAO"

การเข้าถึงข้อมูลด้วย DAO ผู้เขียนจะต้องเขียนโปรแกรมขึ้นควบคุมเองทั้งหมด อีกทั้งยังไม่สามารถใช้งานร่วมกับ Data Bound Control ได้โดยตรง แต่การเข้าถึงข้อมูลด้วย DAO จะมีความยืดหยุ่นและคล่องตัวสูง

การกำหนด Library สำหรับ DAO

ก่อนที่เราจะสามารถนำDAO มาใช้งานได้ ต้องมีการกำหนด Library ก่อนโดยทำได้ดังนี้

Project ⟶ References ⟶ Microsoft DAO 3.51 Object Library

Workspace

Object "Workspace" มักถูกใช้ในการเปิดไฟล์ฐานข้อมูลของ DAO กำหนดขึ้นเพื่อแทนพื้นที่การใช้งานของผู้ใช้แต่ละคน เนื่องจากโดยทั่วไปไฟล์ฐานข้อมูลจะมีการกำหนดผู้ใช้ที่มีสิทธิ์การใช้งานไว้ดังนั้นเมื่อเปิดไฟล์ฐานข้อมูลมาใช้งาน จึงต้องกำหนดพื้นที่การใช้งานให้กับผู้ใช้แต่ละคน

การสร้าง Object "Workspace" มีขั้นตอนดังนี้

1. ประกาศตัวแปรประเภท Object เพื่อแทน Object "Workspace" ด้วยคำสั่งดังนี้

Dim *workspace* AS Workspace

โดยที่ *workspace* หมายถึง ชื่อของตัวแปรประเภท ที่ใช้แทน Workspace

2. สร้าง Workspace ด้วยคำสั่งดังนี้

Set *workspace* = Create Workspace (*name,user,password,type*)

โดยที่ *workspace* หมายถึง ชื่อของตัวแปรประเภท Object ที่กำหนดในข้อ 1

name หมายถึง ชื่อที่กำหนดแทน Workspace ที่สร้างขึ้น

user หมายถึง ชื่อของผู้ใช้ที่เป็นเจ้าของ Workspace ที่สร้างขึ้น

password หมายถึง รหัสผ่านที่กำหนดให้กับ Workspace ที่สร้างขึ้น โดยมีความยาวได้ 14 ตัวอักษร

type หมายถึง ค่าคงที่ที่ใช้ระบุประเภทของ Workspace ที่สร้างขึ้น ได้แก่

การเปิด Recordset ด้วย DAO

เมื่อต้องการเปิด Recordset ด้วย DAO ให้ทำตามขั้นตอนดังนี้

1. ประกาศตัวแปรประเภท Object เพื่อแทน Recordset ด้วยคำสั่งดังนี้

Dim *recordset* **AS Recordset**

โดยที่ *recordset* หมายถึง ชื่อของตัวแปรประเภท Object ที่ใช้แทน Recordset

2. สร้างRecordset ด้วยคำสั่งดังนี้

Set *recordset* = *database*.**Opendatabase** (*source*[,*type*[,*options*[,*lockedits*]]])

โดยที่ *recordset* หมายถึง ชื่อของตัวแปรประเภท Object ที่กำหนดในข้อ 1

database หมายถึง ชื่อของตัวแปรประเภท Object ที่ใช้แทนไฟล์ฐานข้อมูลที่ต้องการนำมาสร้าง Recordset

source หมายถึง ชื่อ Table , Query หรือประโยคคำสั่ง SQL ที่ใช้สร้าง Recordset

type หมายถึง ค่าคงที่ที่ใช้ระบุประเภทของ Recordset ซึ่งจะระบุหรือไม่ก็ได้

options หมายถึง ค่าคงที่ที่ใช้ระบุรูปแบบในการ Lock ข้อมูลในระดับ Recordset ซึ่งจะระบุหรือไม่ก็ได้

lockedits หมายถึง ค่าคงที่ที่ใช้ระบุรูปแบบในการ Lock ข้อมูลในระดับ Record ของ Recordset ซึ่งจะระบุหรือไม่ก็ได้

สำหรับค่าคงที่ที่มักกำหนดให้กับส่วน type มีดังนี้

ค่าคงที่	ประเภทของ Recordset
DbOpenTable	(ค่า Default) Recordset แบบ Table
DbOpenDynaset	Recordset แบบ Dynaset
DbOpenSnapshot	Recordset แบบ Snapshot

การปิด Recordset

เมื่อต้องการปิด Recordset ใด ให้ยกเลิกตัวแปรประเภท Object ที่แทนแต่ละ Recordset นั้นด้วยคำสั่งดังนี้

recordset.**Close** หรือ **Set** *recordset* = **Nothing**

โดยที่ *recordset* หมายถึง ชื่อของตัวแปรประเภท Object ที่ใช้แทน Recordset

การทำ Binding ระหว่าง Bound Control กับ Recordset ของ DAO

การอ้างถึง Field ใน Recordset ของ DAO นี้ สามารถทำได้ 3 วิธีคือ

1. อ้างผ่าน Property "Value" ด้วยรูปแบบดังนี้

recordset!field.**Value**

2. อ้างผ่านเครื่องหมาย " ! " ด้วยรูปแบบดังนี้

recordset!field

ในกรณีที่ชื่อของ field มีการเว้นวรรค ให้ใช้เครื่องหมาย "[]" ประกอบดังนี้

recordset![*field*]

เช่น Field "Year Born" ซึ่งจะสังเกตเห็นว่ามีการเว้นวรรคระหว่างคำว่า "Year" และ "Born"

ดังนั้นในกรณีนี้จึงต้องกำหนดอยู่ในเครื่องหมาย "[ ]"

3. อ้างผ่าน Collection "Field" ด้วยรูปแบบดังนี้

recordset.**Field**(index).**Value**

วิธีนี้จะต้องทราบถึงลำดับที่ของ Field ที่ต้องการภายใน Recordset

โดยที่ *recordset* หมายถึง ชื่อของตัวแปรประเภท Object ที่ใช้แทน Recordset

field หมายถึง ชื่อของ Field ที่ต้องการอ้างถึง

index หมายถึง ลำดับที่หรือชื่อของ Field ภายใน Recordset ที่ ต้องการอ้าง โดยลำดับที่ของ Field จะเริ่มต้นจาก 0 จนกระทั่งถึงลำดับที่ของ Field สุดท้ายใน Recordset ลบด้วย 1

ประวัติผู้ทำโครงงาน

ชื่อ	นายเกียรติศักดิ์ สิงห์ศิริเจริญกุล
วัน เดือน ปีเกิด	25 พฤศจิกายน 2522
สถานที่เกิด	จังหวัด ขอนแก่น
ประวัติการศึกษา	มัธยมศึกษาตอนต้น โรงเรียนภูเวียงวิทยาคม จังหวัด ขอนแก่น พ.ศ.2536 มัธยมศึกษาตอนปลาย โรงเรียนขอนแก่นวิทยายน จังหวัด ขอนแก่น พ.ศ.2539

ชื่อ	นายพรเทพ เกิดจันทึก
วัน เดือน ปีเกิด	24 เมษายน 2524
สถานที่เกิด	จังหวัด นครราชสีมา
ประวัติการศึกษา	มัธยมศึกษาตอนต้นโรงเรียนมัธยมวัดเบญจมบพิตร จังหวัด กรุงเทพมหานคร พ.ศ.2537 มัธยมศึกษาตอนปลายโรงเรียนมัธยมวัดเบญจมบพิตร จังหวัด กรุงเทพมหานคร พ.ศ.2540