การพัฒนาระบบสารสนเทศบนระบบฐานข้อมูลเชิงวัตถุ และระบบฐานข้อมูลเชิงสัมพันธ์

THE DEVELOPMENT OF AN INFORMATION SYSTEM ON AN OBJECT ORIENTED DATABASE SYSTEM AND A RELATIONAL DATABASE SYSTEM.

นายสุชาติ วิชาช่วย

MR. SUCHAT WICHACHUAI

วิทยานิพนธ์เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต

สาขาวิชาวิศวกรรมไฟฟ้า

บัณฑิตวิทยาลัย

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ. 2541

ISBN 974-622-315-1

THE DEVELOPMENT OF AN INFORMATION SYSTEM ON AN OBJECT ORIENTED DATABASE SYSTEM AND A RELATIONAL DATABASE SYSTEM.

SUCHAT WICHACHUAI

A THESIS SUYBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE

MASTER OF ENGINEERING IN ELECTRICAL ENGINEERING

KING MONGKUT'S INSTUTITE OF TECHNOLOGY LADKRABANG

1998.

ISBN 974-622-315-1

หัวข้อวิทยานิพนธ์	การพัฒนาระบบสารสนเทศบนระบบฐานข้อมูลเชิงวัตถุและระบบฐานข้อมูลเชิงสัมพันธ์
นักศึกษา	นายสุชาติ วิชาช่วย
อาจารย์ผู้ควบคุมวิทยานิพนธ์	รศ.ดร.ศุภมิตร จิตตะยโศธร
ระดับการศึกษา	วิศวกรรมศาสตรมหาบัณฑิต สาขาวิชาวิศวกรรมไฟฟ้า สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
พ.ศ.	2541

บทคัดย่อ

การพัฒนาเทคโนโลยีทางด้านระบบสารสนเทศ และการพัฒนาระบบงานฐานข้อมูลทั่วๆไปในปัจจุบัน สามารถทำการพัฒนาได้หลายภาพแบบ ที่นิยมกันมากคือการพัฒนาระบบงานฐานข้อมูลเชิงสัมพันธ์ (Relational Database System) ซึ่งเทคโนโลยีทางด้านนี้มีการพัฒนาก้าวหน้าตลอดเวลา แต่ในปัจจุบันมีการพัฒนาระบบฐานข้อมูลชนิดต่างๆด้วยเทคโนโลยีที่สูงขึ้น ระบบฐานข้อมูลแบบหนึ่งที่มีการพัฒนาอย่างโดดเด่น มีการยอมรับในคุณภาพและความสามารถ ตลอดจนมีความสดวกในการใช้งานคือ ระบบฐานข้อมูลเชิงออบเจ็กต์สัมพันธ์ (Object Relational Database System)

ระบบฐานข้อมูลแบบเชิงออบเจ็กต์สัมพันธ์ เป็นระบบฐานข้อมูลที่มีการพัฒนาให้สนับสนุนกับข้อมูลชนิดใหม่ๆเช่น ภาพ, เสียง, วีดีโอ, ไฟล์ และ ลายเซ็น ฯลฯ หรืองานกราฟฟิกอื่นๆ ซึ่งข้อมูลดังกล่าวเป็นข้อมูลที่อยู่ในภาพแบบของมัลติมีเดีย(Multimedia) ทำให้การจัดเก็บข้อมูลเป็นไปด้วยความยุ่งยากมีความซับซ้อนและไม่สามารถที่จะจัดเก็บด้วยระบบฐานข้อมูลแบบเชิงสัมพันธ์ได้

วิทยานิพนธ์นี้มีการนำเสนอการออกแบบระบบสารสนเทศด้วยวิธีการเชิงออบเจ็กต์ พร้อมทั้งทำการทดลองพัฒนาระบบงานจริง โดยใช้เครื่องมือฐานข้อมูลเชิงสัมพันธ์ และทดลองพัฒนาด้วยเครื่องมือเชิงออบเจ็กต์สัมพันธ์อีกครั้งหนึ่งเพื่อเป็นการเปรียบเทียบ ผลการทดลองปรากฏว่าการออกแบบด้วยวิธีเชิงออบเจ็กต์ สามารถนำมาพัฒนาต้นแบบโดยใช้เครื่องมือฐานข้อมูลเชิงสัมพันธ์และเครื่องมือเชิงออบเจ็กต์สัมพันธ์ได้เป็นอย่างดี

Thesis Title	The development of an information system on an object oriented database system and a relational database system.
Student	Mr. Suchat wichachuai
Thesis Advisor	Assoc.Prof.Dr. Suphamit Chittayasothorn
Level of Study	Master of Engineering in Electrical Engineering King Monkut's Institute of Technology Language
Year	1998

Abstract

The development of information system and database systems can be achieved in many ways. The most popular approach is the use of relational database management systems and relational database design methodogies. However,is recent years object - oriented database and methodologies emerged as a new alternative.

Object oriented databases will let the programe and users see logical objects rather than tables. This is more coneehience than tables for many applications. Object - oriented databases are suitable for complex data and systems which reguire graphical presentations.

This thesis presents the development of information system using an object - oriented database design methodology. The design is implement by using a conventional relational database product and a more recent client/server object-relational system.

กิตติกรรมประกาศ

การจัดทำวิทยานิพนธ์นี้ สำเร็จสมบูรณ์ได้ด้วยความอนุเคราะห์และความกรุณาให้คำแนะนำของ รศ.ดร.ศุภมิตร จิตตะยโศธร อาจารย์ที่ปรึกษา ที่กรุณาให้คำปรึกษา ให้กำลังใจและชี้แนวทางในวิธีการซึ่งเป็นประโยชน์อย่างมาก ในการดำเนินการวิจัย ข้าพเจ้าขอกราบขอบพระคุณเป็นอย่างสูง

ขอขอบพระคุณ ผศ.ดร.บุญวัฒน์ อัตชู ที่กรุณาให้คำแนะนำและคำปรึกษาที่เป็นประโยชน์ในการทำวิทยานิพนธ์นี้

ขอขอบพระคุณ รศ.ดร.รัตติกร วรากูลศิริพันธุ์ ที่กรุณาให้คำแนะนำและคำปรึกษาที่เป็นประโยชน์ในการทำวิทยานิพนธ์นี้

ขอขอบพระคุณ ผศ.ดร.เอื้อน ปิ่นเงิน ที่กรุณาให้คำแนะนำและคำปรึกษาที่เป็นประโยชน์ในการทำวิทยานิพนธ์นี้

ขอขอบพระคุณ ผศ.ดร.บุญธีร์ เครือตราชู ที่กรุณาให้คำแนะนำและคำปรึกษาที่เป็นประโยชน์ในการทำวิทยานิพนธ์นี้

ขอขอบพระคุณผู้จัดการบริษัทอินฟอร์มิก จำกัด คุณศุภประเสริฐ วงศ์สุวรรณ คุณบัณฑิต สื่อวิโรจนกุล และคุณไชโย ตั้งวงค์ศานต์ ที่ได้ให้คำแนะนำระบบฐานข้อมูลอินฟอร์มิก

ขอขอบคุณ คุณ ธนา หงษ์สุวรรณ คุณ ธวัชชัย สุทธิทศธรรม
คุณ นพดล มณีรัตน์ คุณ พิทักษ์ ธรรมวารินทร์
คุณ บัณฑิต พัสยา คุณ ธีรณัติ เริ่มคิดการ
คุณ นวพร วรรณวิมลศรี คุณ สุมณฑา หลิมศิริ
คุณ เทวิน ธนะวงษ์ คุณ รุ่งมณี หงษ์สุวรรณ
คุณ นันทา ต่ำแดง คุณ สรวิศ วิชาช่วย
คุณ เสฏฐวุฒิ วิชาช่วย คุณ สุกฎตยช วิชาช่วย
คุณ สุภาพรรณ วิชาช่วย คุณ พิมล วิชาช่วย
คุณ จันทิมา แสงเลิศอุทัย ที่คอยให้กำลังใจ และสนับสนุนการทำวิจัยนี้

ขอขอบคุณ คุณเดชา ปาลวัฒน์ ที่ได้กรุณาให้คอมพิวเตอร์นำมาทำงานวิจัยในครั้งนี้

รวมทั้งขอกราบขอบพระคุณ คุณพ่อสุดใจ วิชาช่วย และ คุณแม่สนั่น วิชาช่วย ที่ได้กรุณาอบรมและส่งเสริมให้ได้รับความรู้ ความก้าวหน้าตลอดมา

สุชาติ วิชาช่วย

สารบัญ

หน้า

สารบัญ(ต่อ)

หน้า

สารบัญ(ต่อ)

หน้า

สารบัญ(ต่อ)

หน้า

สารบัญ(ต่อ)

หน้า

สารบัญ(ต่อ)

หน้า

สารบัญภาพ

สารบัญภาพ(ต่อ)

สารบัญภาพ(ต่อ)

สารบัญภาพ(ต่อ)

สารบัญภาพ(ต่อ)

บทที่ 1
บทนำ

1.1 ความรู้ทั่วไปเกี่ยวกับระบบสารสนเทศ

การพัฒนาระบบงานสารสนเทศโดยส่วนใหญ่มีการใช้งานกันอย่างกว้างขวาง ทั้งในภาพแบบของบริษัท องค์กร ส่วนราชการ และส่วนบุคคล ซึ่งส่วนใหญ่พัฒนาระบบงานด้วยระบบฐานข้อมูลเชิงสัมพันธ์(Relational Database System) การพัฒนาระบบฐานข้อมูลเชิงสัมพันธ์มีการนำไปใช้งานได้หลายรูปแบบ ซอฟท์แวร์ที่ใช้ในการพัฒนาระบบงานฐานข้อมูลเชิงสัมพันธ์นั้น มีหลายชนิดเช่น dBase, Foxpro, Ingress, Gupta, Informix และ DB2 โดยการพัฒนาระบบงานฐานข้อมูลเชิงสัมพันธ์นี้ ได้รับการยอมรับในมาตรฐานการใช้งานสูง แต่การพัฒนาระบบงานฐานข้อมูลเชิงสัมพันธ์มีลักษณะของข้อมูลเป็นแบบตาราง ทำให้การมองข้อมูลและการจัดการกับข้อมูลมีขีดจำกัด ไม่สามารถพัฒนาระบบงานฐานข้อมูลให้เหมาะสมกับลักษณะงานบางอย่างได้ และระบบงานบางลักษณะไม่สามารถทำการพัฒนาด้วยระบบฐานข้อมูลเชิงสัมพันธ์ได้ ในปัจจุบันได้มีการค้นพบความก้าวหน้าทางด้านเทคโนโลยีของระบบสารสนเทศเพิ่มขึ้น เทคโนโลยีแบบหนึ่งที่มีการพัฒนามาอย่างต่อเนื่องคือ การพัฒนาระบบงานฐานข้อมูลเชิงออบเจ็กต์(Object Oriented Database System) โดยการพัฒนาระบบงานฐานข้อมูลเชิงออบเจ็กต์นี้ มีการศึกษาข้อมูลต่างๆเป็นแบบออบเจ็กต์อันหนึ่งแทนแบบตาราง ทำให้เราสามารถที่จะเข้าถึงและจัดการกับข้อมูลได้โดยตรง การพัฒนาระบบฐานข้อมูลเชิงออบเจ็กต์นี้ มีการพัฒนาเทคโนโลยีทางด้านการติดต่อกับผู้ใช้(User Interface) รวมทั้งมีการใช้ชนิดของข้อมูลที่ซับซ้อน การพัฒนาระบบงานโดยการใช้เทคโนโลยีใหม่นี้จึงทำได้สะดวกและรวดเร็ว ซอฟแวร์ระบบงานฐานข้อมูลเชิงออบเจ็กต์นี้ก็มีหลายอย่าง เช่น Imformix, Delphi, Gupta, Ingres, Postgres ฯลฯ ในวิทยานิพนธ์ฉบับนี้ ใช้ระบบฐานข้อมูลอินฟอร์มิก(Informix) เวอร์ชัน 9.14 เป็นตัวพัฒนาระบบฐานข้อมูลเชิงออบเจ็กต์สัมพันธ์ ระบบฐานข้อมูลอินฟอร์มิกเป็นระบบจัดการฐานข้อมูล

ปัจจุบันการใช้แนวความคิดแบบออบเจ็กต์โอเรียนเต็ด ช่วยทำให้การออกแบบระบบงานมีความยืดหยุ่นต่อการแก้ไขปัญหา และยังเหมาะสมกับระบบปัจจุบันที่ใช้การเชื่อมต่อเป็นแบบยูสเซอร์อินเตอร์เฟส(UGI :Graphic User Interface) โดยเน้นแนวความคิดของออบเจ็กต์ด้วย

ในการพัฒนาซอฟต์แวร์ทางด้านฐานข้อมูลที่มีอยู่ในปัจจุบันนั้น เป็นการเขียนโปรแกรมแบบโครงสร้างและใช้ฐานข้อมูลเชิงสัมพันธ์(Relational Database)เป็นระบบเก็บข้อมูล ซึ่งฐานข้อมูลเชิงสัมพันธ์มีขีดจำกัดอยู่หลายประการคือ

●ขีดจำกัดในการดูแลซอฟต์แวร์เนื่องจากในการสร้างซอฟต์แวร์แบบโปรแกรมโครงสร้างมีความซับซ้อนและมีปัญหาในการแก้ไขโปรแกรม เนื่องจากโปรแกรมเมอร์แต่ละคนมีวิธีการเขียน

โปรแกรมบนมาตรฐานที่ต่างกัน ทำให้การดูแลรักษาระบบซอฟต์แวร์เกิดความยุ่งยาก และมีค่าใช้จ่ายในการดูแลระบบงานสูงตามไปด้วย

●ฐานข้อมูลเชิงสัมพันธ์ เป็นฐานข้อมูลที่ได้รับการยอมรับและใช้งานกันอย่างกว้างขวาง แต่ระบบฐานข้อมูลชนิดนี้ มีจุดด้อยหลายประการ เช่นไม่สามารถรองรับข้อมูลที่ซับซ้อนได้(Complex Data)

●การพัฒนาซอฟต์แวร์ที่ทำงานบนฐานข้อมูลชนิดสัมพันธ์ ผู้พัฒนาจำเป็นต้องควบคุมความถูกต้องของฐานข้อมูลเอง ซึ่งหากขาดตกบกพร่องก็อาจทำให้ซอฟต์แวร์ที่พัฒนาขึ้นนี้เกิดปัญหาได้ โดยความถูกต้องที่ต้องควบคุมนี้ ผู้พัฒนาต้องป้อนเข้าไปในตัวโปรแกรม เป็นการยุ่งยากหากมีการนำโปรแกรมนี้มาแก้ไขในภายหลัง

●เทคโนโลยีทางด้านมัลติมิเดีย ทางวีดีโอ, อิมเมจ, เสียง, ข้อมูลเท็กซ์, กราฟิก ได้เจริญก้าวหน้าไปอย่างมากมาย ระบบฐานข้อมูลเชิงสัมพันธ์ไม่สามารถที่จะรองรับการพัฒนาในระบบเหล่านี้ได้ จึงต้องใช้ฐานข้อมูลชนิดฐานข้อมูลเชิงออบเจ็กต์สัมพันธ์

จากขีดจำกัดหลายประการที่ได้กล่าวมาแล้วนั้น ทำให้มีการนำเอาหลักการของการสร้างซอฟต์แวร์เชิงออบเจ็กต์(Object Oriented System Design) และฐานข้อมูลเชิงออบเจ็กต์สัมพันธ์ (Object-Relational Database)มาใช้งาน เพื่อพัฒนาระบบซอฟต์แวร์ที่ทำงานบนระบบฐานข้อมูลให้มีประสิทธิภาพสูงที่สุด ในวิทยานิพนธ์นี้เป็นการนำหลักการการออกแบบซอฟต์แวร์เชิงออบเจ็กต์มาใช้กับระบบงานคลังสินค้า และใช้ระบบฐานข้อมูลของอินฟอร์มิกซึ่งเป็นฐานข้อมูลเชิงออบเจ็กต์สัมพันธ์ โดยระบบงานนี้เป็นระบบงานแรกในประเทศไทยที่ได้มีการออกแบบและพัฒนาขึ้นโดยวิธีการดังกล่าว

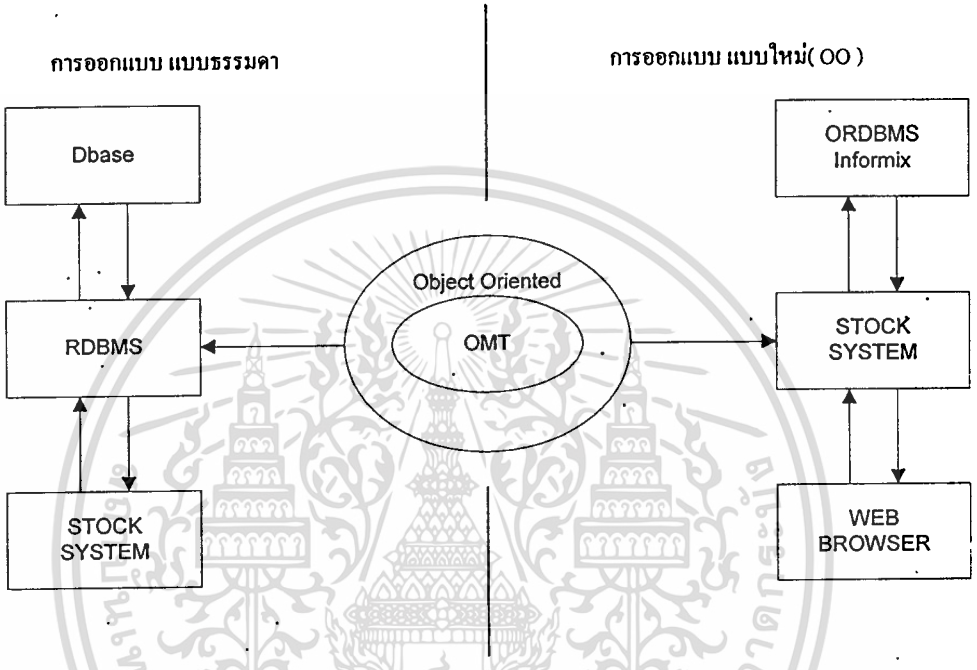
1.2 วัตถุประสงค์

1. ศึกษาการออกแบบระบบซอฟต์แวร์ด้วยแนวคิดทางออปเจ็กต์ หลายวิธี
2. เปรียบเทียบข้อดีและข้อเสียของวิธีการออกแบบหลายๆวิธีที่ได้ศึกษามา
3. เลือกกรรมวิธีที่ได้ศึกษามา 1 วิธี เพื่อสร้างซอฟว์แวร์จริง โดยใช้ร่วมงานกับฐานข้อมูลเชิงสัมพันธ์ (RDBMS) และฐานข้อมูลเชิงออบเจ็กต์สัมพันธ์ (ORDBMS)
4. ศึกษาทางการทำงานแบบออปเจ็กต์ทั้งกระบวนการ
   - Object Orented Design
   - Object Oriented Programming
   - Object Relational Database
5. เปรียบเทียบข้อดีและข้อเสียของการใช้งานโปรแกรมประยุกต์ที่ใช้งานร่วมกับฐานข้อมูลเชิงสัมพันธ์ และโปรแกรมประยุกต์ที่ใช้งานร่วมกับฐานข้อมูลเชิงออบเจ็กต์

1.3 องค์ประกอบของการทำวิจัย

1. ออกแบบโปรแกรมประยุกต์โดยแนวคิดทางออปเจ็กต์
2. ออกแบบฐานข้อมูลโดยใช้แนวคิดทางออปเจ็กต์ร่วมกับฐานข้อมูลเชิงออบเจ็กต์สัมพันธ์
3. สร้างโปรแกรมประยุกต์ โดยใช้แนวคิดทางออปเจ็กต์

ภาพที่ 1.1



แสดงสถาปัตยกรรมของระบบ

1.4 ระบบงานที่ใช้ในการออกแบบ

ระบบงานที่ใช้เป็นระบบฐานข้อมูลแบบไคลเอนต์/เซิร์ฟเวอร์มีภาพแบบที่ง่าย ระบบฐานข้อมูลแบบไคลเอนต์/เซิร์ฟ(C/S: Client Server) แบ่งการจัดการฐานข้อมูลเป็นสองระบบ คือ

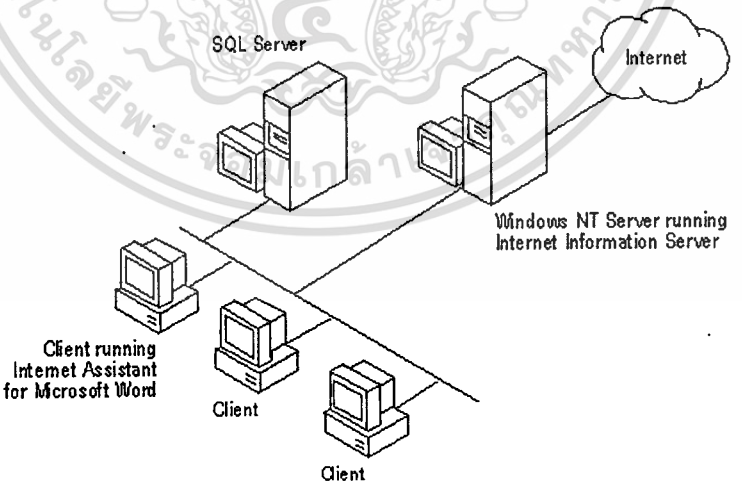
●ไคลเอนต์พีซี ที่จะเรียกใช้ระบบฐานข้อมูล

●ดาต้าเบสเซิร์ฟเวอร์(Database Server) ที่จะเรียกใช้ระบบ DBMS ทั้งหมดหรือบางส่วน ไฟล์เซิร์ฟเวอร์บนเครือข่ายยังเตรียมรีซอร์สที่ใช้ร่วมกันเช่น เนื้อที่ดิสก์สำหรับแอปพลิเคชันและเครื่องพิมพ์ ดาต้าเบสเซิร์ฟเวอร์สามารถทำงานบนเครื่องเดียวกับไฟล์เซิร์ฟเวอร์ หรือบนเครื่องคอมพิวเตอร์ของตนเองแอปพลิเคชันฐานข้อมูลบนไคลเอนต์พีซีถูกเรียกว่าระบบฟรอนต์เอนด์(Front-End System)ทำหน้าที่จัดการกับหน้าจอ และการจัดการอินพุต/เอาต์พุตของผู้ใช้ ส่วนแบ็คเอนด์ (Backend) บนดาต้าเบสเซิร์ฟเวอร์ทำหน้าที่บนข้อมูลและการเข้าถึงดิสก์ ตัวอย่างเช่นผู้ใช้ที่อยู่บนฟรอนต์เอนด์สร้างคำร้องขอ(Query) เพื่อขอข้อมูลจากดาต้าเบสเซิร์ฟเวอร์ และแอปพลิเคชันฟ

รอนต์เอนด์ส่งคำร้องขอนี้ผ่านทางเน็ตเวิร์กไปยังเซิร์ฟเวอร์ ดาต้าเบสเซริฟเวอร์จะดำเนินการค้นหาอย่างแท้จริงและส่งเฉพาะข้อมูลที่เป็นคำตอบของคำร้องขอของผู้ใช้กลับไปเท่านั้น ดังภาพที่ 1.1 และประโยชน์โดยตรงของระบบ C/S ที่ชัดเจนก็คือ การแบ่งการจัดการออกเป็นสองระบบเป็นการช่วยลดจำนวนจองการจราจรของข้อมูลบนสายเคเบิลเน็ตเวิร์ก การพัฒนาระบบงานโดยใช้ ระบบของ C/S

งานวิจัยนี้ได้นำเสนอการออกแบบระบบงานโดยใช้หลักการเชิงออบเจ็กต์ ซึ่งแตกต่างไปจากการออกแบบระบบงานแบบเดิม(Conventional) ที่ใช้วิธีการเก็บรวบรวมข้อมูลต่างๆเป็นแฟ้มข้อมูล(File)หรือฐานข้อมูล(Database) แยกกันกับโปรแกรมระบบงาน ในขณะที่หลักการเชิงออบเจ็กต์จะกำหนดสิ่งต่างๆ ที่อยู่ภายในขอบเขตของระบบงานเป็นออบเจ็กต์(Object)ซึ่งในออบเจ็กต์จะประกอบด้วยข้อมูล(Data)และเมทธอด(Method) รวมกันอยู่ภายในออบเจ็กต์(Encapsulation) เมทธอดเปรียบเทียบได้ทำนองเดียวกันกับโพรซีเยอร์(Procedure) ในโปรแกรมระบบงานแบบเดิมนั่นเอง การเรียกใช้เมทธอดหรือการสั่งให้เกิดการกระทำตามที่กำหนดไว้ในเมทธอดคือการส่งเมสเสจ(Message) ไปยังออบเจ็กต์นั้นๆ ดังนั้นหากมีการเปลี่ยนแปลงวิธีการกระทำ(Function) ใดๆ ของออบเจ็กต์จะไม่มีผลกระทบกับเมสเสจที่ส่งไปยังออบเจ็กต์นั้นและจะไม่มีผลกระทบกับออบเจ็กต์อื่นๆด้วย ซึ่งนับว่าดีอย่างมากกับระบบงานเพราะโดยปกติแล้วพบว่าระบบงานมักจะมีการเปลี่ยนแปลงวิธีการปฏิบัติอยู่เสมอๆ โดยเมทธอด

ภาพที่ 1.2



ระบบไคลเอนต์/เซิร์ฟเวอร์

สามารถใช้อธิบายถึงพฤติการณ์ของออบเจ็กต์แต่ละออบเจ็กต์ได้ ออบเจ็กต์ที่มีพฤติการณ์แบบเดียวกันสามารถจัดรวมกลุ่มเป็นกลุ่มเดียวกันได้เรียกว่าคลาส(Class) ในระหว่างคลาสสามารถที่จะถ่ายทอดคุณสมบัติ(Inheritance) จากคลาสหนึ่งไปยังอีกคลาสหนึ่งได้ โดยเรียกคลาสที่ถ่ายทอดคุณ

สมบัติว่าซุปเปอร์คลาส(Superclass) และเรียกคลาสที่ได้รับการถ่ายทอดคุณสมบัติว่าซับคลาส (Subclass) ซึ่งทำให้การบำรุงรักษาระบบงานทำได้ง่าย เริ่มต้นงานวิจัยด้วยการศึกษาหลักการเชิงออบเจ็กต์ต่อจากนั้นทำการวิเคราะห์เชิงออบเจ็กต์(Object Oriented Analysis) โดยใช้วิธีออบเจ็กต์โมเดลลิ่งเทคนิค(Object Modeling Techique)หรือโอเอ็มที(OMT)ในระบบงานสินค้าคงคลัง หลังจากนั้นสร้างแบบจำลองเพื่อใช้อธิบายระบบงานทั้งระบบงานซึ่งมีอยู่ด้วยกัน 3 แบบจำลอง คือหนึ่งออบเจ็กต์โมเดล(Object Model) ใช้อธิบายรายละเอียดเกี่ยวกับออบเจ็กต์ว่ามีข้อมูลและเมทธอดอะไรบ้าง และแสดงความสัมพันธ์ระหว่างออบเจ็กต์แต่ละออบเจ็กต์ภายในระบบงาน แบบจำลองที่สองคือ ไดนามิกโมเดล(Dynamic Model) ซึ่งใช้สเต็ดไดอะแกรม(State Diagram)อธิบายพฤติกรรมของออบเจ็กต์ที่กระทำโต้ตอบกันภายในระบบ ไดนามิกโมเดลนี้ไม่ค่อยมีความจำเป็นมากนักสำหรับระบบงานสินค้าคงคลัง เพราะว่าเป็นระบบงานเกี่ยวกับการเก็บข้อมูลเสียเป็นส่วนใหญ่ แบบจำลองสุดท้าย คือฟังก์ชั่นนัลโมเดล(Functional Model) ซึ่งใช้ดาต้าโฟลไดอะแกรม(Data Flow Diagram) อธิบายถึงวิธีการคำนวนของเมทธอดภายในออบเจ็กต์แต่ละออบเจ็กต์ หลังจากสร้างแบบจำลองเสร็จแล้วอันดับต่อไปก็คือการพัฒนาระบบ งาน(Imalementation)บนวิชวลเบสิค ระบบปฏิบัติการที่ใช้ในที่นี้ใช้ระบบปฏิบัติการ WINDOWS NT 4.0 และใช้ระบบฐานข้อมูลของอินฟอร์มิกซึ่งเป็นระบบฐานข้อมูลแบบเชิงออบเจ็กต์สัมพันธ์(ORDBMS)

1.5 สาเหตุที่เลือกใช้ออบเจ็กต์โมเดลลิ่งเทคนิค

วิธีการเชิงออบเจ็กต์(Object-Oriented Methodology)มีอยู่หลายวิธีด้วยกันเช่นออบเจ็กโอเรียนเต็ดอะนาไลซีส(Object-Oriented System Analysis) ซึ่งสร้างขึ้นโดยเชลเออร์และเมลเลอร์ (Shlaer and Millor) [Shlaer 1988] [1] โดยใช้อีอาร์ไดอะแกรม(ER-diagram) อธิบายโครงสร้างของข้อมูล ต่อมาในปี 1990 Coad-Yourdon ได้สร้างออบเจ็กต์โอเรียนเต็ดอะนาไลซีส(Object-Oriented Analysis) หรือโอโอเอ (OOA) [1] ซึ่งคล้ายๆ กับออบเจ็กต์โอเรียนเต็ดซิสเต็มอะนาไลซิส(Object Oreinted System Analysis) หรือโอเอสเอ(OSA) ซึ่งพัฒนามาจากออบเจ็กต์รีเลชันชิพโมเดล(Object Relationship Model) หรือโออาร์เอ็ม(ORM) [2] โออาร์เอ็มเป็นผลงานวิจัยของเคร์ก(Kurtz) [Kurtz 1988] โอโอเอแตกต่างกับโอเอสเอ คือ โอเอสเอนั้นสะท้อนถึงภาพแบบของการเขียนโปรแกรมเชิงออบเจ็กต์(Object-Oriented Pregramming) ในปี 1991 เจมส์แรมบอก(James Rambaugh) สร้างออบเจ็กต์โมเดลลิ่งเทคนิค(Object Modeling Technique-OMT) ซึ่งแอ็กซ์แทนเดดอีอาร์ไดอะแกรม (Extended ER diagram) อธิบายโครงสร้างของออบเจ็กต์ ใช้สเตทไดอะแกรม(State Diagram) อธิบายถึงพฤติกรรมโต้ตอบระหว่างออบเจ็กต์ และใช้ดาต้าโฟลไดอะแกรมอธิบายวิธีการคำนวณของเมทธอดในออบเจ็กต์ เราจะสังเกตุเห็นได้ว่าการวิเคราะห์เชิงออบเจ็กต์แต่ละวิธีจะคล้ายๆกัน ดังนั้นผู้ทำวิจัยจึงเลือกใช้วิธีออบเจ็กต์โมเดลลิ่งเทคนิคในงานวิจัยนี้ การออกแบบระบบงานโดยใช้หลักการเชิงออบเจ็กต์เท่าที่ผ่านมามักจะอยู่ในภาพของปัญหาสั้นๆ หรือเป็นเพียงบางส่วนเท่านั้น ดัง

นั้นผู้วิจัยจึงคิดที่จะทดลองออกระบบงานสินค้าคงคลังแบบครบวงจร(Real Lift System)โดยใช้ออบเจ็กต์โมเดลลิ่งเทคนิค และพัฒนาระบบงานบนวิชวลเบสิคเพราะว่าวิชวลเบสิคเป็นซอฟต์แวร์เชิงออบเจ็กต์

1.6 ส่วนประกอบของวิทยานิพนธ์

วิทยานิพนธ์ประกอบด้วยเนื้อหาจำนวน 9 บท และภาคผนวก 1 บท เริ่มจากบทที่ 1 จะกล่าวถึงความเป็นมาของเทคโนโลยีทางด้านระบบสารสนเทศและที่มาของวิทยานิพนธ์ รวมทั้งประโยชน์ที่ได้รับเมื่อวิทยาพนธ์นี้เสร็จสิ้นลง รวมทั้งส่วนประกอบของวิทยานิพนธ์ บทที่ 2 จะกล่าวถึงระบบของฐานข้อมูลเชิงออบเจ็กต์ในรายละเอียดต่างๆ ทั้งในส่วนของความรู้ทั่วไปของออบเจ็กต์โอเรียนเต็ดและอื่นๆ ในบทที่ 3 ได้กล่าวถึงวิธีการออกแบบ แบบออบเจ็กต์โอเรียนเต็ด บทที่ 4 กล่าวถึงระบบฐานข้อมูลเชิงสัมพันธ์ ในบทที่ 5 จะกล่าวถึงระบบฐานข้อมูลเชิงออบเจ็กต์สัมพันธ์และภาษา SQL3 และในบทที่ 6 เป็นระบบฐานข้อมูลของอินฟอร์มิก ซึ่งเป็นระบบฐานข้อมูลแบบออบเจ็กต์ทั้งในส่วนของแบล็คเอ็น(Backend)และฟรอนท์เอ็น(Frontend)บทที่ 7 เป็นการสื่อสารข้อมูลทางอินเตอร์เน็ต บทที่ 8 เป็นกรณีศึกษาการออกแบบระบบฐานข้อมูลเชิงออบเจ็กต์สัมพันธ์ในระบบงานคลังสินค้าในบทที่ 9 ทำการสรุปหาผลของความแตกต่างในการออกแบบ และการวิจารณ์ผลการทดลอง รวมทั้งสรุปปัญหาที่เกิดขึ้น พร้อมข้อเสนอแนะและมีรายละเอียดเกี่ยวกับเอกสารอ้างอิงที่ใช้ในการทำวิทยานิพนธ์นี้.

บทที่ 2
แนวคิดเชิงออบเจ็กต์

แนวความคิดที่กำลังได้รับความนิยมในวงการคอมพิวเตอร์ในปัจจุบันคือ แนวความคิดของออบเจ็กต์โอเรียนเต็ด(OO : Object Oriented)[1],[2] มีความสามารถที่ดีเหมาะสมในการแก้ปัญหา และมีประโยชน์ต่อวงการสารสนเทศในปัจจุบัน อย่างไรก็ตามการนำแนวความคิดนี้มาใช้ให้เกิดประโยชน์อย่างมีประสิทธิภาพนั้น จำเป็นต้องมีความเข้าใจถึงแนวความคิดดังกล่าวเป็นอย่างดี ด้วยเช่นกันออบเจ็กต์โอเรียนเต็ดไม่ได้ถูกจำกัดอยู่เฉพาะกับการโปรแกรมมิ่งเพียงอย่างเดียว แต่ยังรวมไปถึงปรัชญาทั้งหมดเช่น การวิเคราะห์ระบบ(SA : System Analysis) การออกแบบระบบ(SD : System Design) การออกแบบฐานข้อมูล(Database Design) และเรื่องอื่นๆที่เกี่ยวข้องในสาขาเดียวกันนี้ด้วย

2.1 การพัฒนาของออบเจ็กต์โอเรียนเต็ด

การเกิดขึ้นของออบเจ็กต์โอเรียนเต็ดเป็นการสะท้อนประวัติศาสตร์ทั้งหมด ของการคำนวน (Computing) โดยงานทางการคำนวนในยุคแรกๆ จะเกี่ยวข้องกับการโปรแกรมมิ่ง(Programming) ตราบจนในภายหลังที่เริ่มมีความสนใจเกี่ยวกับ การออกแบบและการวิเคราะห์ระบบขึ้นมาเป็นเรื่องที่แยกออกไป ในทำนองเดียวกันเราสามารถกล่าวได้ว่า เป็นเพราะการโปรแกรมแบบออบเจ็กต์โอเรียนเต็ดนั่นเองที่ดึงดูดความสนใจในระยะแรก ต่อมาในภายหลังการออกแบบและการวิเคราะห์โดยวิธีออบเจ็กต์โอเรียนเต็ด จึงเกิดมาเป็นที่สนใจของผู้คนในวงการคอมพิวเตอร์ โดยที่ออบเจ็กต์โอเรียนเต็ดเริ่มต้นมีการพัฒนามาจากภาษาซิมูล่า(Simula) ในนอร์เวย์ ตอนปีคศ. 1967 ภาษาซิมูล่า นี้เป็นภาษาที่ใช้ในการเลียนแบบเหตุการณ์(Discrete Event Simulanon) แล้วได้ถูกพัฒนาต่อมาเป็นอีกภาษาหนึ่งที่ได้ใช้แนวความคิดทางออบเจ็กต์โอเรียนเต็ดอย่างแท้จริง ซึ่งภาษานั้นก็คือภาษาสมอลทอล์ก(Smalltalk)ที่เกิดในช่วงทศวรรษที่ 1970

แต่รูปแบบของซิมมูเลชันโมเดลลิ่ง(Simulation Modelling) เป็นปัญหาอย่างมาก สำหรับโปรแกรมเมอร์ที่ใช้ภาษาแบบคอนเว็นชันนอล(Conventional) หรือภาษาในยุคที่สาม(Third-Generation Language) เพราะโปรแกรมเมอร์จะต้องเข้าใจถึงฟังก์ชันนอลโฟลว์(Functional Flow) ในการควบคุมอย่างลึกซึ้ง แต่เป็นสิ่งที่ง่ายสำหรับภาษาที่เป็นแบบคอนโทรลโฟลว์(Control Flow) โดยใช้อธิบายในภาพแบบของออบเจ็กต์ที่ซับซ้อน สามารถเปลี่ยนสถานะและมีอิทธิพลต่อเหตุการณ์ จากช่วงหนึ่งไปถึงอีกช่วงหนึ่ง ในภาษาแบบออบเจ็กต์โอเรียนเต็ดฟังก์ชันนอลโฟลว์ว่าจะถูกแทนด้วยการส่งผ่านเมสเสจ(Massage)ระหว่างออบเจ็กต์ต่างๆ ที่เป็นสาเหตุให้เกิดการเปลี่ยนแปลงสถานะของออบเจ็กต์ ดังนั้นการโปรแกรมแบบออบเจ็กต์โอเรียนเต็ดจึงเป็นวิธีที่ดูเป็นธรรมชาติ เพราะโครงสร้างของโปรแกรมจะสะท้อนถึงปัญหาที่ต้องการแก้โดยตรง

ภาพที่ 2.1

แสดงตัวอย่างออบเจ็กต์

ในระยะต่อมาถึงช่วงทศวรษที่ 1980 มีผู้ให้ความสนใจเกี่ยวกับคอมพิวเตอร์ ในเรื่องของยูสเซอร์อินเตอร์เฟส โดยมีผู้นำทางการค้าที่เป็นที่รู้จักมากที่สุดได้แก่ซีรอกส์(Xerox) ได้นำระบบยูสเซอร์อินเตอร์เฟสแบบใหม่มาใช้ นั่นคือ WIMP(WIMP: Windows Icons Mice and Pointer) และนำแนวความคิดในภาษาสมอลทอล์ก(Smalltalk) มาใช้ในการพัฒนาระบบดังกล่าวโดยมีการอ้างอิงถึงภาพแบบการติดต่อกับผู้ใช้โดยการใช้ภาพกราฟฟิก หรืออีกนัยหนึ่งก็คือการโปรแกรมมิ่งแบบออบเจ็กต์โอเรียนเต็ดนี้จะสนับสนุนการพัฒนาระบบยูสเซอร์อินเตอร์เฟส ในทางกลับกันรูปแบบของภาษาแบบออบเจ็กต์โอเรียนเต็ดก็ได้รับอิทธิพลมาจาก WIMP ด้วยเช่นกัน สิ่งที่เป็นสาเหตุที่ทำให้ออบเจ็กต์โอเรียนเต็ดโปรแกรมมิ่งประสบความสำเร็จเป็นอย่างมากก็คือ ความซับซ้อนของกราฟฟิกยูสเซอร์อินเตอร์เฟส และความสามารถที่จะนำกลับมาใช้ได้อีก(Reuseability) ของโปรแกรมแบบออบเจ็กต์โอเรียนเต็ด ระบบการติดต่อแบบนี้จึงสามารถสร้างขึ้นมาใช้งานในระดับกว้างขวางได้ เมื่อออบเจ็กต์โอเรียนเต็ดโปรแกรมมิ่งเริ่มต้นที่จะเติบโตเต็มที่ ความสนใจก็ได้เลื่อนไปอยู่ที่วิธีในการออกแบบ โดยใช้แนวความคิดของออบเจ็กต์โอเรียนเต็ด รวมทั้งการวิเคราะห์แบบออบเจ็กต์โอเรียนเต็ด และการกำหนดรายละเอียด(Specification) ข้อดีของการนำกลับมาใช้ได้อีก(Reuseablity) และการที่สามารถขยายได้(Extensibility) ก็ได้ถูกนำมาประยุกต์ใช้กับการออกแบบโดยการกำหนดรายละเอียดเช่นเดียวกับการโปรแกรม แต่ยังมีการถกเถียงกันอยู่ในสาขานี้ด้วย เป็นต้นว่าการออกแบบออบเจ็กต์โอเรียนเต็ดนี้ ต้องนำมาสร้างหรืออิมพลีเมนต์(Implement) ด้วยภาษาแบบออบเจ็กต์โอเรียนเต็ดด้วยหรือไม่

สำหรับในด้านระบบฐานข้อมูล ภายหลังจากที่รีเลชันนอลดาต้าเบส ได้รับบการยอมรับแล้ว ผู้ผลิตระบบฐานข้อมูลรายใหญ่ต่างๆ ได้มีการเพิ่มเติมรีเลชันนอลดาต้าเบสในส่วนที่เรียกว่า โพสรีเลชันนอล(Post-Relational) ออกมาอีกหลายแบบ โดยแต่ละแบบต่างก็มีฟิลด์(Field) ที่เป็นต้น

กำเนิดแตกต่างกันออกไปเช่น ระบบผู้เชี่ยวชาญ ฟังก์ชันนอลโปรแกรมมิ่ง(Functional Programming) และปัจจุบัน ได้มีการสร้างออบเจ็กต์โอเรียนเต็ดโปรแกรมมิ่งสำหรับออบเจ็กต์โอเรียนเต็ดออกมาใช้ในทางธุรกิจเมื่อไม่นานนี้

2.2 แนวความคิดของออบเจ็กต์โอเรียนเต็ด

แนวความคิดของออบเจ็กต์โอเรียนเต็ดเป็นแนวความคิดที่มีคุณสมบัติแตกต่างจากแนวความคิดเดิมโดยเฉพาะแนวความคิดในการพัฒนาโปรแกรมแบบเดิมที่เป็นโปรแกรมโครงสร้าง (Structured Programming) ส่วนที่เป็นโปรแกรมและข้อมูลจะแยกจากกัน(Separates Data and Program) ในขณะที่แนวความคิดของออบเจ็กต์โอเรียนเต็ดจะมีออบเจ็กต์(Object)เป็นที่รวมของข้อมูลและวิธีการ(Object Combines Data and Methods) โดยมีคลาส(Class) เป็นตัวกำหนดคุณสมบัติของออบเจ็กต์ ซึ่งเป็นเครื่องมือสำคัญในการพัฒนาต้นแบบ(Prototyping Development) รวมถึงสามารถนำออบเจ็กต์กลับมาใช้ใหม่ได้อีก

2.3 คุณสมบัติของออบเจ็กต์โอเรียนเต็ด

กำหนดรายละเอียดขั้นตอนการประมวลผลว่าควรจะมีขั้นตอนอะไรบ้าง เช่น

- มีการกำหนดข้อมูลที่ใช้
- มีการปรับปรุงหรือเปลี่ยนแปลงข้อมูลอย่างไร
- มีการคำนวนหรือมีฟังก์ชันที่เกี่ยวข้องอะไรบ้าง
- มีการแสดงผลของข้อมูลจะแสดงอย่างไร

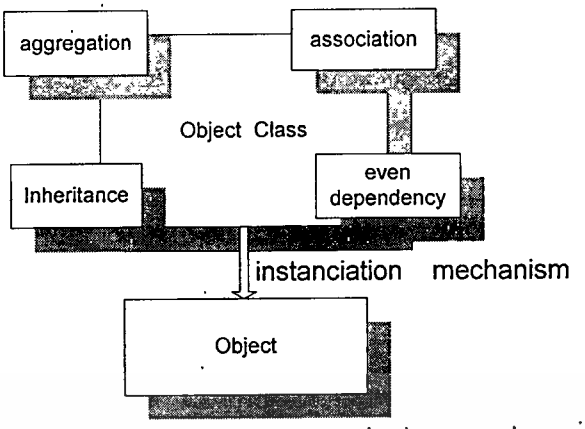
ขั้นตอนการทำงานที่เชื่อมต่อกับผู้ใช้ในการแสดงผลออกมานี้ เรียกว่าดาต้าแอปสแตรกชัน (Data Abstraction)เพื่อใช้ในการกำหนดคำนิยามของคลาส ลักษณะการกำหนดรายละเอียดเหล่านี้จะเป็นลักษณะจากล่างขึ้นบน(Bottom - Up Design)

1. ออบเจ็กต์(Object) ถูกกำหนดว่ามีสมาชิกหรือรายละเอียดของข้อมูล(Attributes) และฟังก์ชันหรือขั้นตอนที่เกี่ยวข้องอะไรบ้าง โดยกำหนดคำนิยามในคลาส

2. การนิยามคลาส คลาสต่างๆสามารถนำกลับมาใช้ใหม่ได้ถ้าต้องการ(Reuseable) ซึ่งเป็นวิธีการสืบทอดคลาส(Class Inheritance) จากโปรแกรมเดิมมายังโปรแกรมใหม่ เพื่อทำการปรับปรุงเพิ่มเติมแทนโดยไม่ต้องต้องเสียเวลาเขียนทั้งโปรแกรม

3. โครงสร้างโปรแกรมแบบออบเจ็กต์โอเรียนเต็ดขึ้นอยู่กับข่าวสาร(Message) ที่ส่งไปยังออบเจ็กต์เพื่อให้ทำงานตามขั้นตอนที่ระบุไว้

ภาพที่ 2.2



แสดงภาพแนวความคิดของออบเจ็กต์โอเรียนเต็ด

2.4 การสร้างโมเดลของฐานข้อมูลออบเจ็กต์

การสร้างโมเดลของฐานข้อมูลแบบออบเจ็กต์ มีวัตถุประสงค์เพื่อช่วยให้การออกแบบฐานข้อมูล สามารถนำไปใช้ในการสร้างฐานข้อมูลได้ โดยมีหลักการคล้ายคลึงกับการสร้างโมเดลอีอาร์ (E-R Model) มีเพียงข้อแตกต่างบางประการ โดยขั้นตอนในการสร้างโมเดล สามารถสรุปได้ดังนี้

- กำหนดว่ามีออบเจ็กต์อะไร(Object Name) และออบเจ็กต์นั้นๆมีรายละเอียดอะไรบ้าง (Attributes) รวมถึงวิธีการ(Methods)
- กำหนดความสัมพันธ์ระหว่างออบเจ็กต์ว่ามีความสัมพันธ์อะไรบ้าง
- การเชื่อมโยงในการส่งข่าวสารว่ามีอะไรบ้าง

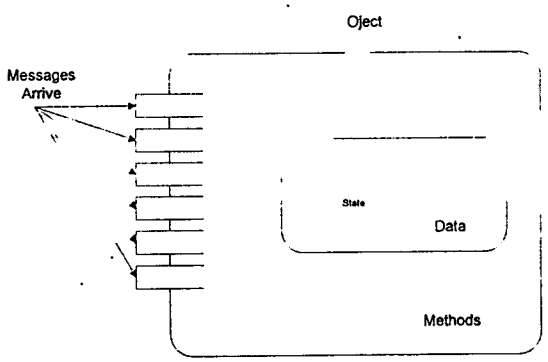
2.5 วิธีของออบเจ็กต์โอเรียนเต็ด

ระเบียบวิธีทางออบเจ็กต์โอเรียนเต็ดนี้หมายถึงหลายๆสิ่ง เช่นเดียวกับคำว่าออบเจ็กต์โอเรียนเต็ด แต่ความหมายเฉพาะที่เราให้ความสนใจอยู่ได้แก่ในเรื่องของ

- ออบเจ็กต์โอเรียนเต็ดโปรแกรมมิ่ง
- การออกแบบ แบบออบเจ็กต์โอเรียนเต็ด
- การวิเคราะห์ แบบออบเจ็กต์โอเรียนเต็ด
- ฐานข้อมูล แบบออบเจ็กต์โอเรียนเต็ด

หรือว่าอีกนัยหนึ่งก็คือทุกๆเรื่องที่เกี่ยวข้องกับการพัฒนาระบบที่ใช้หลักการออบเจ็กต์โอเรียนเต็ดนั่นเอง ในหัวข้อต่อไปจะแนะนำให้ทราบถึงคำเฉพาะที่ใช้อยู่ในระเบียบวิธีทางออบเจ็กต์โอเรียนเต็ด คำเฉพาะเหล่านี้มีการนำมาใช้ในความหมายที่แตกต่างกันสำหรับผู้เขียนแต่ละคน เหตุที่เป็นเช่นนี้ก็เนื่องจากว่าเรื่องของออบเจ็กต์โอเรียนเต็ดนี้เป็นเรื่องที่เกิดขึ้นมาใหม่นั่นเอง

ภาพที่ 2.3



แสดงโครงสร้างความสัมพันธ์ระหว่างออบเจ็กต์คาต้าเมสเสจและเมทธอด.

จากภาพที่ 2.3 ออบเจ็กต์โอเรียนเต็ดคือปริมาณหนึ่งในระบบที่ประกอบขึ้นด้วย 2 ส่วนคือ ข้อมูล และโค๊ดโปรแกรม ส่วนข้อมูลใช้เก็บสถานะของตัวมันเองเรียกว่า คาต้า(Data) และส่วนโค๊ดโปรแกรม ใช้ในการตอบสนองออบเจ็กต์ตัวอื่นในระบบเดียวกันเรียกว่าเมทธอด(Method) ออบเจ็กต์ใดๆ ในระบบมีการสื่อสารกับออบเจ็กต์อื่นๆเพื่อให้บรรลุความต้องการของตนเอง การสื่อสารนี้เป็นลักษณะ " การร้องขอและการตอบสนอง "เมื่อออบเจกต์หนึ่งขอความช่วยเหลือจากอีกออบเจ็กต์หนึ่งเราเรียกว่าวิธีการแบบนี้ว่า การส่งเมสเสจ(Message)

2.6 ลักษณะพื้นฐานของออบเจ็กต์

แนวความคิดแบบออบเจ็กต์โอเรียนเต็ดในส่วนของการออกแบบ, การโปรแกรม และการออกแบบฐานข้อมูล ถูกกำหนดให้อยู่ภายใต้คุณสมบัติสำคัญอยู่ 2 ประการคือ

- แอปสแตรกชัน(Abstraction)
- อินเฮียริแทนส์(Inheritance)

โดยแอบสแตรกชันและอินเฮียริแทนส์ต่างก็มีแนวความคิดที่สำคัญอื่นๆซ่อนไว้อยู่ภายใน

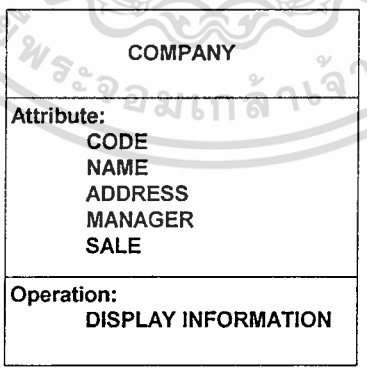
2.6.1 แอปสแตรกชัน(Abstraction)หมายถึง การแทนคุณสมบัติที่สำคัญๆ ของบางสิ่งบางอย่างโดยไม่นำรายละเอียดที่ไม่จำเป็นมา ในทางโปรแกรมมิ่งคำนี้จึงหมายถึงว่าออบเจ็กต์ควรจะถูกแทนด้วยการรวมข้อมูล(Data)และโพรเซส(Process)ที่เกี่ยวข้องเข้าไว้ด้วยกัน เพื่อเป็นตัวแทนของออบเจ็กต์นั้น ออบเจ็กต์จะถูกกำหนดโดยกลุ่มของแอททริบิวต์(Attributes)ที่ใช้รวมกัน ใช้แทนออบเจ็กต์นั้น แอททริบิวต์ดังกล่าวนี้อาจมีชื่อเรียกว่าอินสแตนท์วาไรเอเบิล(Instance variable) หรือคลาสวาไรเอเบิล(Class variable) นอกจากลุ่มของแอททริบิวต์แล้ว ออบเจ็กต์สามารถจะกำหนดไปพร้อมๆ กันกับกลุ่มของเมทธอด(Method) ที่ได้รับอนุญาต และสามารถปฏิบัติการกับแอททริบิวต์ของออบเจ็กต์ได้ โดยปกติแล้วแอททริบิวต์ของออบเจ็กต์จะไม่สามารถเข้าถึงได้จากภายนอก แต่เข้าถึงแอททริบิวต์เหล่านั้นได้โดยผ่านทางเมทธอดของออบเจ็กต์นั้นเพียงอย่างเดียว

แนวความคิดที่มีการผูกฟังก์ชันและข้อมูลเข้าด้วยกันนี้ เป็นความสัมพันธ์อย่างใกล้ชิดกับแนวความคิดในเรื่องชนิดของข้อมูล ในภาษาโปรแกรมมิ่งทั่วไป เช่นในขณะที่ 3 เป็นอินสแตนท์ของข้อมูลชนิดจำนวนเต็ม และจำนวนเต็มถูกกำหนดให้มีแอททริบิวต์หนึ่งตัวคือ ค่าของมันเอง แต่อนุญาตให้มีเมทธอด(Operation) ทางคณิตศาสตร์ได้หลายตัว ตัวเลขจำนวนจริงที่ถูกกำหนดให้มีสิ่งต่างๆ ในลักษณะเดียวกัน แต่การอิมพลีเม้นต์(Implementation) ของการคูณสำหรับตัวเลขแบบโฟลตติ้งพอยต์(Floation point) จะแตกต่างกับการคูณตัวเลขจำนวนเต็มธรรมดา ในลักษณะนี้เราจึงสามารถจัดการกับออบเจ็กต์ที่มีความซับซ้อนได้

2.6.2 ออบเจ็กต์(Objects) เป็นรูปแบบพื้นฐานของข้อมูลเชิงออบเจ็กต์ซึ่งออบเจ็กต์หมายถึงสิ่งใดๆ ที่มีอยู่จริงและสามารถจับต้องได้หรือสิ่งใดๆ ที่สามารถแสดงลักษณะใดๆ ได้ โดยที่ลักษณะเหล่านั้นเป็นคุณสมบัติเฉพาะของแต่ละออบเจ็กต์ที่สามารถแปรเปลี่ยนไปมาได้ตลอดการคงอยู่ของออบเจ็กต์เหล่านั้น หรือออบเจ็กต์คือตัวแปรของคลาส โดยมีรายละเอียดข้อมูล(Attributes) และขั้นตอนการทำงานหรือฟังก์ชันไว้ในออบเจ็กต์นั้น(Message)

ตัวอย่างเช่นจากรีเลชัน Company สมมุติให้เป็นรีเลชันผู้ผลิตซึ่งข้อมูลในรีเลชันผู้ผลิตเป็นออบเจ็กต์ที่ประกอบด้วยรายละเอียด รหัสผู้ผลิต(CODE) ชื่อ(NAME)และจังหวัด(PROVINCE) รวมถึงวิธีการที่ทำงานของออบเจ็กต์เป็นต้น ออบเจ็กต์จะถูกใช้งานโดยการส่งข่าวสารไปที่ออบเจ็กต์ โดยข่าวสารเหล่านี้ประกอบด้วยพารามิเตอร์ที่ระบุไว้ในคลาสนั่นเอง วิธีการทำงานที่ระบุให้กับผู้ผลิตเป็นการแสดงรายละเอียดของผู้ผลิต การจัดกลุ่มผู้ผลิตที่อยู่จังหวัดเดียวกัน

ภาพที่ 2.4

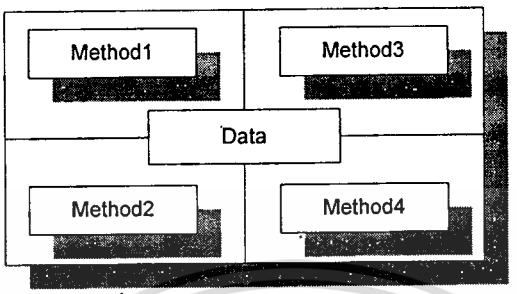


แสดงรีเลชัน Company

2.6.3 เอ็นแคปซูเลชัน(Encapsulation) เป็นรายละเอียดของข้อมูล และวิธีการของออบเจ็กต์ ที่ระบุให้ทราบถึงการที่จะได้มาซึ่งผลลัพท์ โดยในระบบมีตัวประสานของออบเจ็กต์เป็นตัวจัดการให้ว่าต้องทำอะไรบ้าง ซึ่งการรวมกันของโครงสร้างระหว่างข้อมูลกับฟังก์ชันที่เกี่ยวข้องจะทำให้ข้อมูลมีความมั่นคงขึ้น มีผลทำให้ไม่มีการเข้าถึงข้อมูลโดยตรงอีกต่อไป นั่นคือเป็นการทำ

การซ่อนข้อมูลที่เรียกว่าดาต้าไฮดิ้ง(Data Hiding) ซึ่งข้อดีของการทำเอ็นแคปซูเลชันคือทำให้เราสามารถแบ่งระดับของการเข้าถึงข้อมูลได้ด้วยตัวเมทธอดของตัวออบเจ็กต์เอง

ภาพที่ 2.5



แสดงโครงสร้าง เอ็นแคปซูเลชัน

คุณสมบัติของ เอ็นแคปซูเลชันคือ

1. กำหนดขอบเขตที่ชัดเจนให้กับออบเจ็กต์
2. กำหนดส่วนของการอินเตอร์เฟส ซึ่งหมายความว่าออบเจ็กต์นั้นจะติดต่อกับออบเจ็กต์ อื่นอย่างไร
3. ส่วนอิมพลีเม้นต์ไม่สามารถเข้าถึงข้อมูลนอกจากขอบเขตที่กำหนด

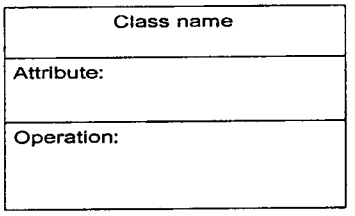
ตัวอย่างเช่นออบเจ็กต์เกี่ยวกับการจัดส่งสินค้าคือ ORDER ประกอบด้วยรายละเอียด รหัสผู้ผลิต(CODE) รหัสสินค้า(ST_CODE) และจำนวนสินค้าที่จัดส่ง(SE_NUMBER) โดยได้กำหนดวิธีการไว้ว่า ให้สรุปยอดรวมการจัดส่งของผู้ผลิตแต่ละราย หากมีการจัดส่งของให้กับผู้ผลิตรายใดเพิ่มเติมในออบเจ็กต์ของ ORDER จะมีการส่งข่าวสารสรุปยอดรวมการส่งของออกมาโดยอัตโนมัติ ซึ่งวิธีการนี้ ผู้ที่ต้องการแสดงผลลัพท์ไม่จำเป็นต้องระบุวิธีการแสดงผลอีก

2.6.4 คลาส(CLASS) คือกลุ่มของออบเจ็กต์ที่แบ่งตามลักษณะเฉพาะและการใช้งาน หรือออบเจ็กต์ที่มีคุณสมบัติพื้นฐานเหมือนกัน ซึ่งข้อมูลเชิงออบเจ็กต์ที่เก็บอยู่ภายในคลาสจะเรียกว่าอินสแตนท์ (Instance)ของคลาส โดยที่อินสแตนท์จะเป็นส่วนขยายของคลาสและถูกเก็บอยู่ในส่วนของฐานข้อมูล ดังนั้นข้อมูลต่างๆที่เป็นตัวกำหนดคลาสจะถูกเข้าถึงและถ่ายทอดผ่านทางออบเจ็กต์ของอินสแตนท์ และเมทธอดของคลาสโดยคลาสจะห่อหุ้มคุณลักษณะทั้งหมดของออบเจ็กต์ต่างๆ และกำหนดชนิดของข้อมูล ที่บรรจุอยู่ในออบเจ็กต์พร้อมกับกำหนดเมทธอดต่างๆ สำหรับการเข้าถึงข้อมูลไว้ด้วย

คลาสต์ในความหมายของออบเจ็กต์โอเรียนเต็ดโปรแกรมมิ่งจะประกอบไปด้วย กลุ่มของออบเจ็กต์ที่ใช้แอททริบิวต์ และเมทธอดพื้นฐานร่วมกันโดยออบเจ็กต์มีมุมมองทั้งจากภายในและภายนอกคือมุมมองภายในจะบอกถึงสถานะการอิมพลีเม้นต์ของออบเจ็กต์ ส่วนมุมมองภายนอกจะ

แสดงชื่อของเมทธอดและชนิดของพารามิเตอร์(Parameter) ซึ่งโครงสร้างของคลาสแสดงดังภาพที่ 2.6

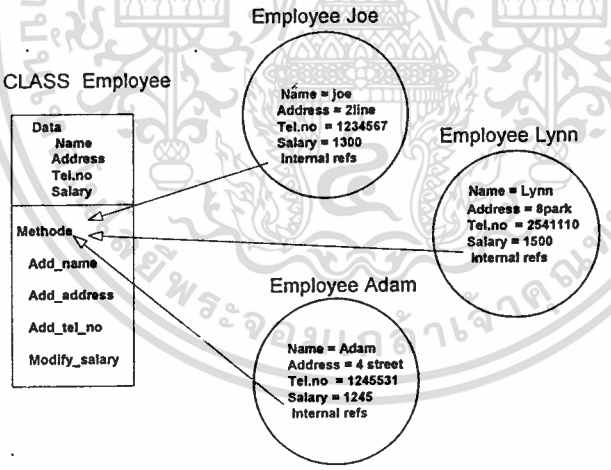
ภาพที่ 2.6



แสดงโครงสร้างของ คลาส

ภาพต่อไปนี้เป็นสัญลักษณ์ที่ใช้ในการเขียนอธิบายออบเจ็กต์หนึ่งๆ

ภาพที่ 2.7



แสดงความสัมพันธ์ระหว่าง คลาส กับออบเจ็กต์

จากแนวความคิดในการปกปิดข้อมูลไม่ให้สามารถติดต่อโดยตรงจากภายนอกของออบเจ็กต์ การติดต่อระหว่างออบเจ็กต์จึงเป็นการใช้แมสเสจติดต่อกัน ระหว่างออบเจ็กต์แมสเสจหนึ่งจะประกอบด้วยแอดเดรส(Address)ของออบเจ็กต์ที่จะส่งเมสเสจนั้นไปถึง และคำสั่งที่ประกอบด้วยชื่อของเมทธอด และตามด้วยพารามิเตอร์ถ้ามี ถ้าออบเจ็กต์ที่รับแมสเสจนั้นมีเมทธอดที่ต้องทำการตอบข้อมูลบางตัวก็สามารถส่งกลับไปให้ออบเจ็กต์ที่ส่งแมสเสจนั้น

ออบเจ็กต์ต่างๆจะกำหนดขึ้นในคลาส โดยผู้ใช้จะระบุถึงโครงสร้างข้อมูล และวิธีการจัดการข้อมูลข้อออบเจ็กต์ต่างๆ ในคลาสหรือการกำหนดฟังชันต่างๆ ที่กระทำกับข้อมูลนั่นเอง ดังนั้นคลาสมีความสำคัญสำหรับแนวความคิดนี้ ที่บอกถึงข้อมูลและฟังก์ชันที่จัดการกับออบเจ็กต์ ตัวอย่างเช่น คำสั่งใน C++ การนิยามรายละเอียดการถูกใช้งานของสมาชิกในคลาส สามารถทำได้ 2 ภาพแบบ คือ

แบบ Private หมายถึงข้อมูลและฟังก์ชันที่ระบุไว้ในส่วนนี้จะถูกใช้งานโดยฟังก์ชันที่ระบุไว้ใน Public ที่อยู่ภายในคลาสเดียวกันเท่านั้นเป็นการปกป้องไม่ให้โปรแกรมอื่นๆที่อยู่ภายนอกคลาส มาใช้ข้อมูลได้คุณสมบัตินี้เรียกว่าการซ่อนข้อมูล(Data Hiding)

แบบ Public หมายถึงข้อมูลและฟังก์ชันต่างๆ ในส่วนของ Public จะสามารถใช้โดยคลาสที่อยู่ภายนอกได้

ภาพที่ 2.8

```
CLASS COMPANY
{
      private :
                  char  code(6);
                  char  name(30);
                  char  provice(12);
      public :
                  void  show_data(void);
                  void  update(void);
};
```

แสดงการนิยามคลาส COMPANY

จากภาพ 2.8 เป็นการนิยามคลาสของ Company โดยแบ่งออกเป็นส่วนของPrivate และ Public โดยใช้ภาษา C++ ออบเจ็กต์ที่ถูกระบุเป็น Private จะถูกใช้งานตามฟังก์ชันที่ระบุใน Public เช่น show_data หรือ update เท่านั้น เป็นการป้องกันไม่ให้โปรแกรมอื่นๆ เข้ามาใช้

อย่างไรก็ตาม การเข้าถึงหรือการเรียกใช้ข้อมูลในส่วนที่เป็น Private จากภายนอก อาจจะทำได้โดยการกำหนดให้ฟังก์ชันที่อยู่นอกคลาสเป็นฟังก์ชันเพื่อน เพื่อเป็นการยืดหยุ่นให้สามารถใช้งานในส่วนของ Private ได้ ตัวอย่างคำสั่งใน C++ จะใช้คำสั่ง Friend กำหนดฟังก์ชันเพื่อนขึ้นมาการนิยามคลาสในส่วนของ Public ดังภาพที่ 2.9

ภาพที่ 2.9

```
CLASS COMPANY
{
        private :
                char  code(6);
                char  name(30);
                char  provice(12);
        public :
                void  show_data(void);
                void  update(void);
                friend int total_qty(const company
                                &d1, constt order
&d2);
};
```

```
CLASS ORDER
{
        private :
                char  code(6);
                char  st_code(12);
                char  se_invoice(12);
                int   se_number;
                int   se_price;
        public :
                void  show_data(void);
                void  update(void);
                friend int total_qty(const company
                                &d1, constt order
&d2);
};
```

แสดงการนิยามคลาสสองคลาสคือ Company และ Order

จากภาพที่ 2.9 จะเห็นว่ามีการนิยามคลาสสองคลาสคือ COMPANY และ ORDER โดยมีการระบุฟังก์ชันเพื่อนในตอนท้ายของการนิยามคลาสว่า TOTAL โดย SE_NUMBER เป็นฟังก์ชันเพื่อนเพื่อเป็นการบอกล่าวล่วงหน้าว่าฟังก์ชันเพื่อนในคลาส COMPANY จะมีนิยามคลาส ORDER ตามมา ซึ่งตัวแปรโปรแกรมจะทราบเมื่อมีการอ้างอิงถึงข้อมูลในคลาส ORDER

2.6.5 แอททริบิวต์(Attributes) คือค่าของข้อมูลที่เป็นส่วนบ่งบอกรายละเอียดของออบเจ็กต์ที่อยู่ในคลาส ซึ่งแอททริบิวต์จะแบ่งออกเป็น 2 ประเภท คือ

■ Class Attributes คือแอทริบิวต์ของคลาสนั้นๆ ที่ถูกเข้าถึงโดยตัวคลาสเอง Instance และ Subclass ของตัวคลาส

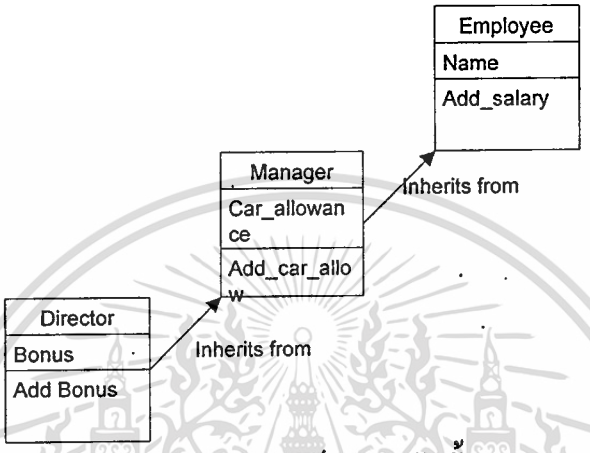
■ Instance Attribute คือแอททริบิวต์ที่เข้าถึงโดยข้อมูลของคลาสที่กำหนดเท่านั้น ซึ่งจะคล้ายคลึงกับแอททริบิวต์ในโครงสร้างอื่นๆ โดยที่แอททริบิวต์ของข้อมูลนั้นจะมีค่าตายตัวของมันเอง(Defaults Instance Attributes) หรืออาจจะมีค่าสากลที่ประยุกต์ใช้กับแต่ละ ข้อมูลของคลาสได้(Share Instance Attributes)

2.6.6 พฤติกรรมของคลาส(Behavior) คือการที่ออบเจ็กต์สามารถทำสิ่งต่างๆให้กับตัวมันเองหรือมีสิ่งต่างๆทำให้มัน โดยจะพิจารณาที่ข้อมูลของคลาสนั้นดำเนินการเปลี่ยนแปลงสถานะภายในของมันหรือเมื่ออินสแตนท์ถูกขอร้องให้ทำบางสิ่งบางอย่างจากคลาสหรือจากออบเจ็กต์อื่น

2.6.7 ความสัมพันธ์ระหว่างคลาส(Relationship) คือสิ่งที่บ่งบอกถึงความสัมพันธ์ระหว่างคลาสกับคลาส**ความสัมพันธ์ระหว่างคลาส**สามารถแบ่งออกได้เป็น 2 ชนิดคือ ความสัมพันธ์เชิงลำดับขั้น (Hierarchical)และความสัมพันธ์แบบไม่มีกฎเกณฑ์(Non -Hierarchical)

- Hierarchical คือความสัมพันธ์ที่เชื่อมระหว่างคลาสแบบเชิงลำดับขั้น

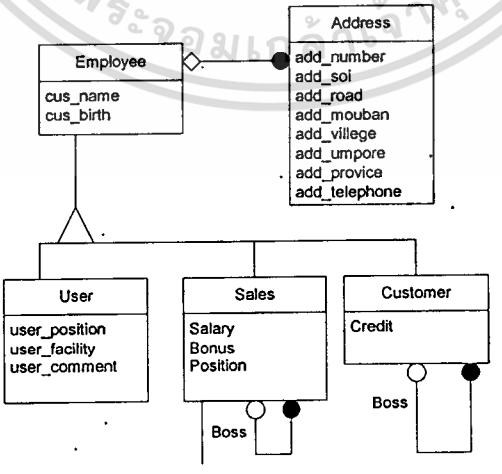
ภาพที่ 2.10



แสดงความสัมพันธ์แบบลำดับชั้น

■ Non-Hierarchical คือความสัมพันธ์ในภาพแบบที่ไม่สามารถเสนอออกมาในรูปของความสัมพันธ์เชิงลำดับขั้นได้ แต่จะแสดงได้อย่างชัดเจนในรูปแบบของ Object Relationship Mapping นั้นคือความสัมพันธ์ในภาพแบบของ 1-1, 1-M, M-1 และ M-M

ภาพที่ 2.11

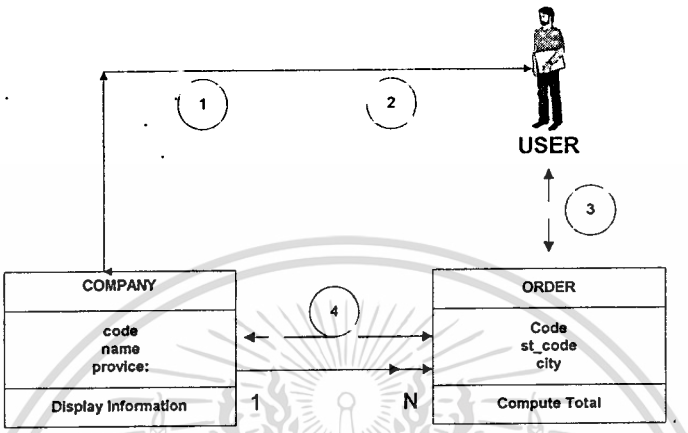


แสดงความสัมพันธ์ในภาพแบบของ 1-1, 1-M, M-1 และ M-M

2.6.8 เมทธอด(Methods)ถูกกำหนดขึ้นมาเพื่อใช้กับคลาส ให้มีหน้าที่ในการจัดการตามหน้าที่หรือตามที่คลาสระบุขึ้นมา

2.6.9 การส่งข่าวสาร(Messages) ออบเจ็กต์ต่างๆ ในคลาสจะถูกสั่งให้ทำงานเมื่อมีการส่งข่าวสารจากออบเจ็กต์หนึ่งไปยังอีกออบเจ็กต์หนึ่ง โดยตัวประสานหรือตัวเชื่อมข่าวสาร(Message Connection)จะแสดงเส้นทางการติดต่อสื่อสารระหว่างคลาส

ภาพที่ 2.12



แสดงวิธีการส่งข่าวสารระหว่างออบเจ็กต์.

จากภาพ 2.13 แสดงการส่งข่าวสารระหว่าง COMPANY และ ORDER โดยให้แสดงข้อมูลของผู้ผลิตแล้วทำการเพิ่มข้อมูลผู้ผลิต และให้เพิ่มข้อมูลการส่งสินค้าของผู้ผลิต แล้วทำการคำนวนจำนวนรวมของการจัดส่งสินค้าผู้ผลิตแต่ละคน

การส่งข่าวสาร(Message)

1. Company. Display Information
2. Company. Occur. Add
3. Order. Occur. Add
4. Order. Compute Total

2.6.10 โพลิมอร์ฟิซึม(Polymorphism) คือการที่คลาสหนึ่งๆ สามารถแปรเปลี่ยนไปได้หลายภาพแบบ ขึ้นอยู่กับสภาพแวดล้อมหรือสถานการในขณะนั้น ความสามารถอีกอย่างหนึ่งของโพลิมอร์ฟิซึมก็คือสามารถกำหนดการดำเนินการใหม่ให้กับตัวดำเนินการหรือแม้แต่ฟังก์ชันได้ เรียกว่าการทำโอเวอร์โหลดดิ้ง(Overloading) คือการเรียกใช้ฟังก์ชันหรือตัวดำเนินการเดิมให้ไปทำงานอีกอย่างหนึ่งได้ คุณลักษณะเด่นอีกอย่างหนึ่งของแนวความคิดออบเจ็กต์โอเรียนเต็ด คือความสามารถที่มีฟังก์ชันหรือโอเปอเรเตอร์ต่างกันใช้ชื่อเดียวกันได้ โดยโปรแกรมออบเจ็กต์โอเรียนเต็ดจะแยกความแตกต่างของการใช้ชื่อที่เหมือนกันได้ด้วยจำนวนและชนิดของพารามิเตอร์

ตัวอย่างเช่นฟังก์ชันการพิมพ์ที่ใช้ใน C++ สมมุติให้เรียกชื่อฟังก์ชัน PRINT เหมือนกัน ดังนี้

แบบที่ 1 : void print(char, width)

แบบที่ 2 : void print(int, width)

เมื่อมีการใช้ฟังก์ชัน PRINT( ) เข้ามา ตัวแปรโปรแกรมจะทำหน้าที่เรียกฟังก์ชัน PRINT ที่อยู่ในภาพแบบตามชนิดของพารามิเตอร์ที่ส่งเข้ามา เช่น

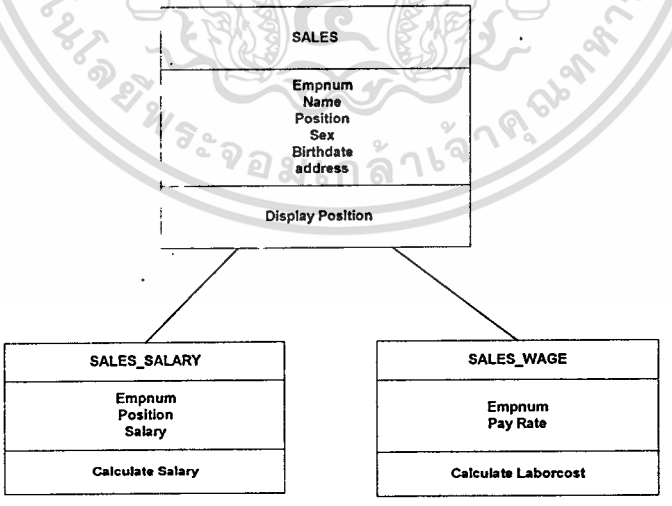
ตัวอย่างแรก : PRINT("OBJECT ORIENTED", 20) จะนำแบบที่ 1 จากข้างบนมาใช้

ตัวอย่างที่สอง : PRINT(200000, 6) จะนำ แบบที่2 จากข้างบนมาใช้

คำว่า โพลิมอร์ฟีซึม และโอเปอเรเตอร์โอเวอร์โหลดดิ้ง(Operator Overloading) คำทั้งสองนี้หมายถึงความสามารถที่จะใช้เครื่องหมายสัญลักษณ์เดียวกัน โดยมีความหมายที่แตกต่างกันได้ เช่นเมื่อส่งเมสเสจ "บวก 5" ไปยังตัวเลขจำนวนเต็มและเลขจำนวนจริงออบเจ็กต์ของตัวเลขแต่ละแบบทั้งสองก็จะทำการกระตุ้นโปรแกรมในการบวกที่แตกต่างกัน ทั้งในแบบของจำนวนเต็มและจำนวนจริง แต่มันเป็นการง่ายที่จะใช้เครื่องหมาย + แทนความหมายทั้งสองรวมทั้งยังทำให้ภาษาโปรแกรมง่ายต่อการเรียนรู้และใช้งานอีกด้วย

2.6.11.การสืบทอดคลาส(Inheritance) จุดเด่นของแนวความคิดออบเจ็กต์โอเรียนเต็ด คือ การสืบทอดคลาสโดยการทำการแปลงคุณสมบัติคลาสที่เคยใช้อยู่เดิม(Base Class) มาเป็นคลาสใหม่ที่สืบทอดมาจะรับคุณสมบัติของรายละเอียดและวิธีการต่างๆ จากคลาสเก่า(Dervived Class) การสืบทอดคลาสจะช่วยให้การพัฒนาโปรแกรมใหม่ทำได้เร็วยิ่งขึ้น

ภาพที่ 2.13



แสดงภาพแบบของการสืบทอดของคลาส Sales

ตัวอย่างจากภาพ 2.11 ในคลาส Sales ประกอบด้วยรายละเอียด รหัสพนักงาน(Empnum) ชื่อ(Name) ตำแหน่ง(Position) เพศ(Sex) วันเดือนปีเกิด(Birthdate) และที่อยู่(Address) และมี

คลาสย่อย ชื่อ EMP_SALARY และ EMP_WAGE โดยที่คลาสย่อย EMP_SALARY สืบทอดรายละเอียดรหัสพนักงาน และตำแหน่งจากคลาส Employee รวมถึงวิธีการมาด้วย คือ ให้แสดงรหัสและตำแหน่งของพนักงาน(Display Position) และในขณะเดียวกัน ก็อาจจะมีรายละเอียดของข้อมูลและวิธีการในคลาสย่อยอีก คือเงินเดือนและการคำนวนเงินเดือนสุทธิหลังภาษีตามลำดับ ซึ่งการสืบทอดรายละเอียดและวิธีการในลักษณะนี้จะทำให้การพัฒนาชุดคำสั่งงานทำให้รวดเร็วมากขึ้น

อินเฮียริแทนส์(Inheritance) คือการสืบทอด หรือกระบวนการในการจัดการคลาส เป็นการสร้างส่วนของคลาสใหม่ขึ้นมาโดยมีพื้นฐานมาจากคลาสเดิม แต่จะมีข้อมูลหรือมีฟังชันค์ ที่พิเศษ เป็นของตัวเองเพิ่มขึ้นมาจากคลาสเดิม ซึ่งก่อให้เกิดการแตกสายพันธ์ของออบเจ็กต์ เป็นออบเจ็กต์ใหม่ที่มีคุณสมบัติของออบเจ็กต์เดิมซ่อนอยู่

■ คลาส และ อินสแตนท์ การขยายความของคลาสจะนำเสนอโดยกลุ่มของอินสแตนท์ ที่มีคุณสมบัติเดียวกัน นั่นคือ ความสัมพันธ์ของคลาสแม่(Super class) และคลาสลูก(Subclass) จะถูกนำเสนอออกมา โดยผ่านทางอินสแตนท์

■ แอททริบิวต์ซึ่งคลาสลูกจะสืบทอดแอททริบิวต์ต่างๆ จากคลาสแม่โดยตรงนั่นคือถ้ามีการเปลี่ยนแปลงใดๆ ที่คลาสแม่แล้วคลาสลูกจะได้รับการเปลี่ยนแปลงโดยตรง การสืบทอดแอททริบิวต์ทั้งชื่อ ข้อจำกัด และความสัมพันธ์ต่างๆ ภายในแต่ละแอททริบิวต์ ซึ่งข้อจำกัดที่คลาสลูกได้รับนั้นอาจจะเป็นเพียงส่วนหนึ่งของคลาสแม่ หรือได้รับการถ่ายทอดมาทั้งหมด ซึ่งจะรวมทั้งการกำหนดค่า Default Value ด้วย

■ Relationship การสืบทอดคุณสมบัติความสัมพันธ์จะรวมในส่วนของชื่อ ชนิดของความสัมพันธ์ ข้อจำกัดต่างๆ และความสัมพันธ์ที่เกี่ยวข้องกับคลาสอื่นๆ

■ โอเปอเรชันจะถูกสืบทอดโดยคลาสลูก ซึ่งจะเกี่ยวข้องกับ

1. คลาสลูกสามารถเพิ่มโอเปอเรชันเข้าไปได้โดยอัตโนมัติ
2. โอเปอเรชันของคลาสลูกสามารถเป็นส่วนขยายของคลาสแม่ได้

อินเฮียริแทนส์ เป็นคุณสมบัติของออบเจ็กต์โอเรียนเต็ด ซึ่งการอินเฮียริแทนส์ก็คือวิธีการในการจัดการกับความสัมพันธ์ทางโครงสร้างและความหมายระหว่างออบเจ็กต์กับคลาส และตัดความซ้ำซ้อนของการที่ต้องการเก็บข้อมูล หรือโพรซีเดอร์มากเกินกว่าที่จำเป็น

2.6.11.1 การใช้งานของ อินเฮียริแทนส์ ที่ทำให้เกิดการสับสนมีอยู่ 6 อย่างคือ

●อินเฮียริแทนส์และซับไทป์(Subtyping) การใช้งานในภาพแบบของออบเจ็กต์โอเรียนเต็ด ทั้งอินเฮียริแทนส์และซับไทป์บางครั้งจะมีการใช้สลับปรับเปลี่ยนกันไปมาทำให้เกิดความสับสน แต่ทั้งอินเฮียริแทนส์และซับไทป์ก็มีความแตกต่างกันทั้งในด้านโครงสร้างและแนวความคิด

●ความสัมพันธ์ของตัวแปรและวิธีการจัดการบางครั้งการใช้งานของตัวแปร มีการยินยอมให้มีการใช้ตัวแปรได้โดยตรง แต่บางครั้งการใช้งานของตัวแปรก็มีการแยกการใช้งานระหว่างตัวแปร private และตัวแปรแบบ public

●อินเฮียริแทนส์และเอ็นแคปซูเลชัน. ในความเป็นจริงถ้าในตัวแปรของ คลาสแม่สามารถที่จะถูกจัดการได้โดยตรงมันจะส่งผลให้เกิดการขัดแย้งกันระหว่างอินเฮียริแทนส์ และเอ็นแคปซูเลชัน

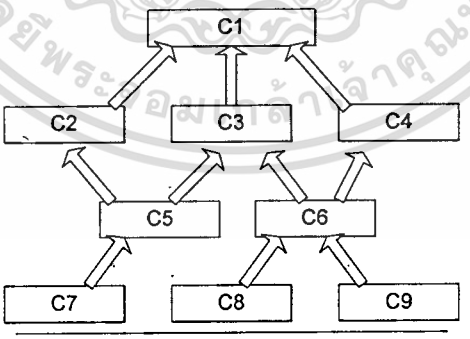
●การกำหนดอินเฮียริแทนส์ของคลาสสามารถกำหนดเพิ่มเติมให้มีการขยายทั้งในส่วนของตัวแปรและคุณสมบัติของคลาส ซึ่งจากการที่สามารถขยายได้อาจมีผลทำให้เกิดความขัดแย้งกัน

●ออบเจ็กต์อินเฮียริแทนส์ในภาษาของออบเจ็กต์ฮโอเรียนเต็ดมีความสามารถที่จะสร้างหรือกำหนด Class Inheritance

●มัลติเพิ่ลอินเฮียริแทนส์ ในบางครั้งเพื่อความเหมาะสมเรามีความต้องการที่จะให้มีการอินเฮียริต(Inherit)มากกว่าหนึ่งคลาส ซึ่งในกรณีนี้เราเรียกว่ามัลติเพิ่ลอินเฮียริแทนส์และเมื่อคลาสมีอินเฮียริตมากว่าหนึ่ง จะเกิดการขัดแย้งขึ้นมาทั้งในด้านเมทธอดและตัวแปร

2.6.11.2 อินเฮียริแทนส์และซับไทป์คือเซ็ตของออบเจ็กต์และเซ็ตของเมทธอดของออบเจ็กต์

ภาพที่ 2.14

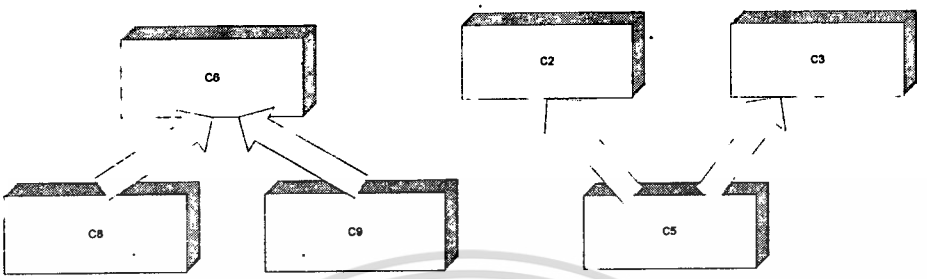


แสดง Class Hiearchy ด้วย อินเฮียริแทนส์ แบบ Single และ Multiple

ออบเจ็กต์หนึ่งๆ จะทำการสืบทอดคุณสมบัติหรืออินเฮียริทจากคลาสของมัน และเนื่องจากคลาสก็เป็นออบเจ็กต์หนึ่งก็สามารถสืบทอดคุณสมบัติจากคลาสอื่นได้ด้วย ซึ่งคลาสนี้จะเป็นซุปเปอร์คลาสของคลาสที่ทำการอินเฮียริทคุณสมบัติมาจากการที่มีการสืบทอดคุณสมบัติต่อๆ กันนี้จึงทำให้เกิดอินเฮียริแทนส์เน็ตเวอร์ก(Inheritance Network)ขึ้น ออบเจ็กต์หนึ่งๆ สามารถอินเฮียริทมา

จากคลาสมากว่าหนึ่งคลาสได้ในเวลาเดียวกัน ซึ่งเราจะเรียกความสามารถในการอินเฮียริทแบบนี้ว่า มัลติเพิลอินเฮียริแทนส์(Multiple inheritance)

ภาพที่ 2.15

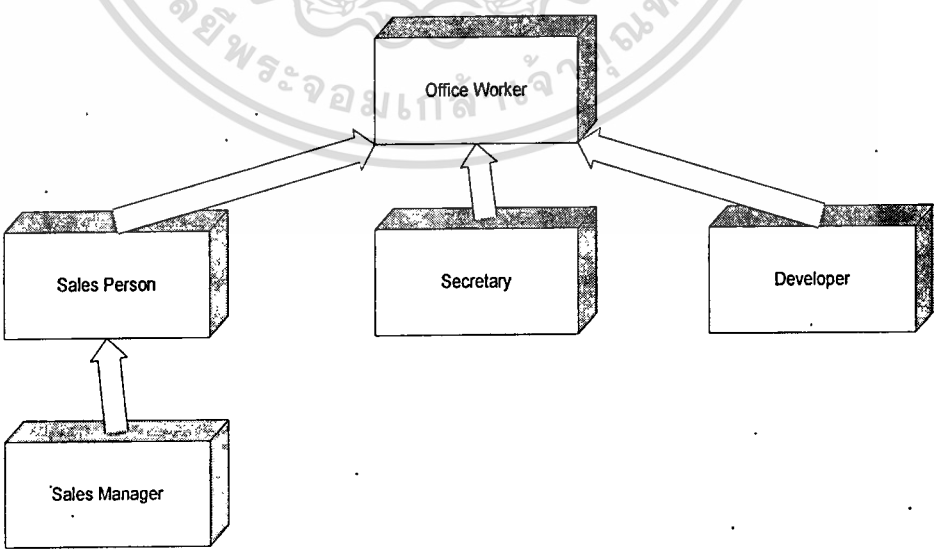


(A) Single Inheritance (B) Multiple Inheritance.

แสดง คลาสแบบลำดับชั้นด้วยอินเฮียริแทนส์แบบเดี่ยวและรวม

แต่มัลติเพิลอินเฮียริแทนส์ก์มีปัญหาที่อาจเกิดขึ้นได้เช่นกัน ปัญหาของมัลติเพิลอินเฮียริแทนส์จะเกิดขึ้นเมื่อคุณสมบัติที่ออบเจ็กต์ทำการอินเฮียริทมาจากหลายๆ คลาสนั้นต่างขัดแย้งกัน ในทางปฏิบัติก็สามารถแก้ปัญหาเหล่านี้ได้ เช่น กำหนดดีฟอสท์(Default) ของคุณสมบัติเหล่านั้น เป็นต้น ภาพต่อไปนี้จะแสดงตัวอย่างของอินเฮียริแทนส์เน็ตเวอร์คของลูกจ้างของบริษัท

ภาพที่ 2..16



แสดงอินเฮียริแทนส์ของ Office workers

บทที่ 3

การออกแบบเชิงวัตถุ

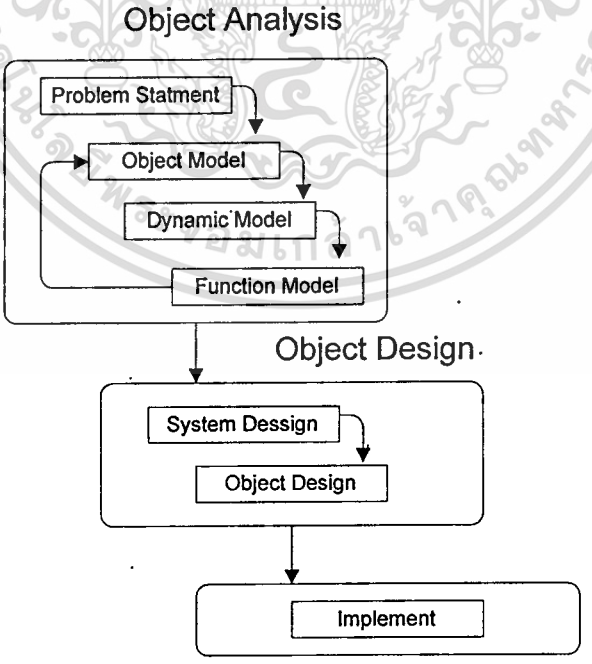
การออกแบบออบเจ็กต์โอเรียนเต็ดเป็นแนวความคิดใหม่ที่นำมาใช้เพื่อวิเคราะห์การแก้ปัญหา โดยการสร้างแบบจำลองอิงกับปัญหาในรูปแบบที่เป็นรูปธรรม แต่วิธีการออกแบบออบเจ็กต์โอเรียนเต็ดมีหลายวิธี วิธีที่นิยมใช้กันมากแบบหนึ่งคือออบเจ็กต์โมเดลลิ่งเทคนิค

3.1 ออบเจ็กต์โมเดลลิ่งเทคนิค(OMT : Object Modeling Technique)

ออบเจ็กต์โมเดลลิ่งเทคนิค(OMT)[1],[2]เป็นวิธีการออกแบบออบเจ็กต์โอเรียนเต็ด ที่สามารถแบ่งออกเป็นส่วนสำคัญได้ 3 ส่วนคือ

1. การวิเคราะห์ออบเจ็กต์(Object Anaylysis.)
2. การออกแบบออบเจ็กต์(Object Design.)
3. การประยุกต์การใช้งาน(Implement)

ภาพที่ 3.1



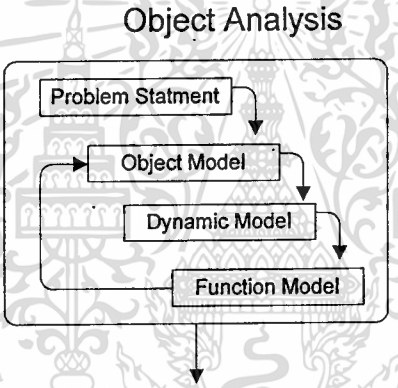
แสดงส่วนประกอบของการทำงานแบบ OMT

การออกแบบเพื่อแก้ปัญหาโดยเริ่มจากการวิเคราะห์(Analysis),ออกแบบ(Design)และประยุกต์พัฒนาโปรแกรม(Implementation) ขั้นตอนของ OMT เริ่มต้นจากการออกแบบและสร้างแบบจำลองของปัญหา แล้วเพิ่มรายละเอียดเข้าไปในแบบจำลอง ในขั้นตอนของการวิเคราะห์แบบจำลองของ OMT

3.2 การวิเคราะห์ออบเจ็กต์

ขั้นตอนการวิเคราะห์เริ่มจากการเก็บรวบรวมข้อกำหนดต่าง และลักษณะของปัญหาจากระบบงาน แล้วนำมาทำการสร้างแบบจำลองทั้งสามดังนี้

ภาพที่ 3.2

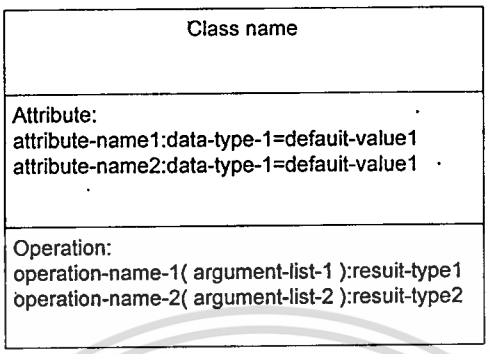


แสดงภาพแบบของการวิเคราะห์ออบเจ็กต์

3.3 แบบจำลองออบเจ็กต์(Object Modeling)

เป็นแบบจำลองที่แสดงถึงตัวออบเจ็กต์และความสัมพันธ์ระหว่างออบเจ็กต์ รวมทั้งแอททริบิวต์และเมทธอดของออบเจ็กต์คลาสด้วย แบบจำลองออบเจ็กต์ถือว่าเป็นแบบจำลองที่มีความสำคัญมากที่สุดในแบบจำลองทั้งสามแบบที่จะกล่าวต่อไป เพราะว่าเป็นแบบจำลองที่แสดงถึงโครงสร้างของระบบโดยใช้รูปภาพและสัญลักษณ์ต่างๆ สื่อความหมายเช่นทำให้ทั้งผู้ออกแบบระบบและผู้ใช้ระบบสามารถทำความเข้าใจกับตัวระบบได้อย่างเป็นรูปธรรม

ภาพที่ 3.3



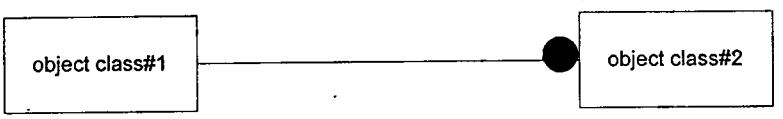
แสดงสัญลักษณ์ที่ใช้แทนออบเจ็กต์คลาส

สัญลักษณ์ที่ใช้แทนออบเจ็กต์คลาส จะประกอบด้วยส่วนหลักๆ สามส่วนด้วยกัน ส่วนแรกใช้แสดงชื่อคลาส ส่วนที่สองแสดงรายการของแอททริบิวต์ของออบเจ็กต์คลาส ในแต่ละแอททริบิวต์ที่แสดงในส่วนที่สองจะมีรายละเอียดเพิ่มเติมอีกเล็กน้อย เช่นชนิดของข้อมูลและค่าเริ่มต้น ในส่วนที่สามหรือส่วนของเมทธอดก็จะประกอบด้วยรายละเอียดปลีกย่อยเช่น รายการและชนิดของผลลัพธ์ อย่างไรก็ตามเมื่อถึงเวลาในการเขียนแบบจำลองแล้ว ชื่อของแอททริบิวต์และเมทธอดบางตัวอาจละไว้ได้ตามสมควร ความสัมพันธ์ระหว่างออบเจ็กต์สามารถกำหนดได้ด้วยสัญลักษณ์ด้านล่างซึ่งมีทั้งแบบ หนึ่งต่อหนึ่ง(One -to -One)และแบบหลายต่อหลาย(Many-to-Many) โดยทั่วไปแล้วความสัมพันธ์ที่ปรากฏในไดอะแกรม(Diagram)จะได้มาจากข้อกำหนดของปัญหา(Problem Statement) ที่ได้ตระเตรียมไว้

ภาพที่ 3.4

ความสัมพันธ์แบบ1:1 (One to One)

ความสัมพันธ์แบบ1:M (One to Many)



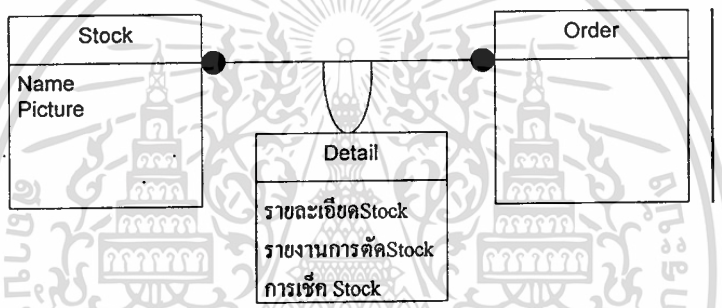
ความสัมพันธ์แบบM:M (Many to Many)

ภาพที่ 3.4 (ต่อ)

แสดงความสัมพันธ์แบบต่างๆ ใช้ในออบเจ็กต์คลาส

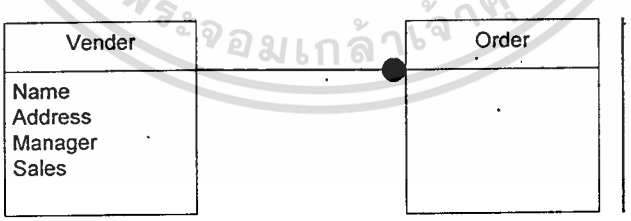
การเชื่อมโยงหรือลิ้งค์(link)ที่มีแอททริบิวต์มากกว่าหนึ่งแอททริบิวต์ สามารถแสดงความสัมพันธ์ระหว่างคลาส ในรูปแบบต่างๆได้คือ

ภาพที่ 3.5



แสดงความสัมพันธ์ที่เหมาะสมระหว่างคลาส

ภาพที่ 3.6



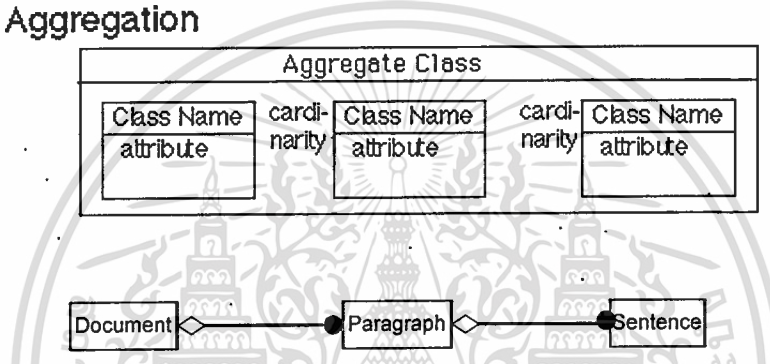
แสดงความสัมพันธ์ที่ไม่เหมาะสมระหว่างคลาส

การใส่แอททริบิวต์ให้กับลิ้งค์หรือแอซโซสิเอชัน ทำให้เราสามารถเปลี่ยนความสัมพันธ์ระหว่างออบเจ็กต์คลาสได้โดยไม่กระทบกับออบเจ็กต์คลาสของเดิม และนอกจากนี้แล้วยังทำให้การจัดเก็บข้อมูลในภาพของฐานข้อมูลลดความซ้ำซ้อนลงอีกด้วย ในบางครั้งเราสามารถที่จะกำหนดความสัมพันธ์ให้เป็นคลาสก็ได้

3.3.1 โรล(Role Nmae)หมายถึงชื่อที่กำหนดให้กับจุดต่างๆ ของการเชื่อมต่อกันของคลาส โดยปกติจะอยู่ติดกับตัวคลาส

3.3.2. แอกกริเกชัน(Aggregation)หมายถึงการแสดงความสัมพันธ์ระหว่างคลาสในภาพแบบพิเศษ ซึ่งบ่งบอกถึงการเป็นส่วนประกอบของคลาสที่นำมาเชื่อมต่อกัน ซึ่งภาพแบบของแอกกริเกรชันแสดงดังภาพข้างล่าง

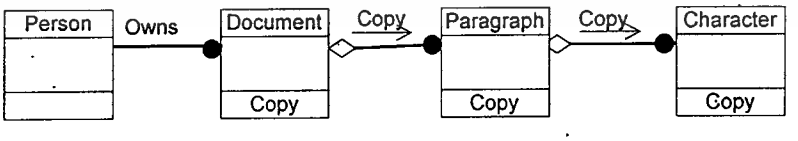
ภาพที่ 3.7



แสดงความสัมพันธ์ระหว่างคลาสในภาพแบบพิเศษ

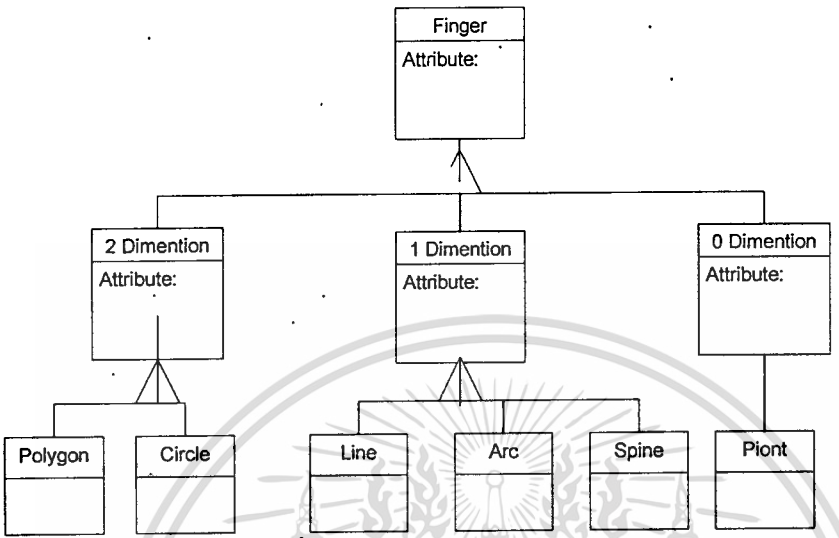
3.3.3 โพรพาเกรชัน(Propagation)หรือเรียกอีกอย่างหนึ่งว่า ทริกเกอร์(Trigger)ใช้ในการกระตุ้นจากเหตุการณ์หนึ่งไปยังอีกเหตุการณ์หนึ่ง

ภาพที่ 3.8



แสดง รูปแบบของทริกเกอร์(Trigger)

ภาพที่ 3.9



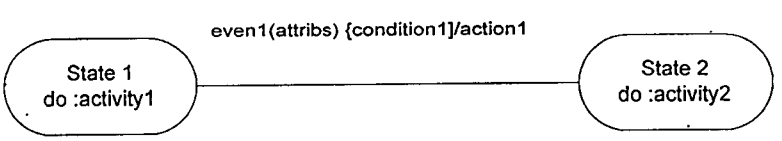
แสดงการเชื่อมต่อกันของคลาส Multilevel aggregation

3.4 แบบจำลองไดนามิค(Dynamic Modeling)

ลำพังแต่ออบเจ็กต์โมเดลเราไม่สามารถที่จะอธิบายความหมายของระบบทั้งหมดได้ เพราะออบเจ็กต์โมเดลอธิบายแต่ความสัมพันธ์และโครงสร้างของออบเจ็กต์เท่านั้น แต่แบบจำลองไดนามิคทำหน้าที่อธิบายถึงลำดับการเปลี่ยนแปลงโอเปอร์เรชันของออบเจ็กต์ ที่ถูกกระตุ้นโดยเหตุการณ์(Events) โดยไม่สนใจว่าโอเปอร์เรชันนั้นจะทำอะไรหรือทำขึ้นมาอย่างไร

หลักการสำคัญของแบบจำลองไดนามิคคือสถานะ(States) ที่แสดงค่าของออบเจ็กต์ ณ จุดหนึ่งและอีเว้นต์(Events)เป็นตัวกระตุ้นให้เกิดการเปลี่ยนแปลงสถานะ รวมกันเป็นสเต็ทไดอะแกรม (State Diagram)สัญลักษณ์ที่ใช้แสดงในสเต็ทไดอะแกรม มีภาพแบบดังต่อไปนี้

ภาพที่ 3.10



แสดงส่วนประกอบของ สเต็ทไดอะแกรม

ส่วนประกอบของสเต็ทไดอะแกรม ประกอบด้วยตัวสเต็ท ซึ่งชื่อของสเต็ท จะถูกเขียนด้วยตัวพิมพ์หนา ส่วนชื่อของเหตุการณ์จะอยู่บนเส้นตรงระหว่างสเต็ท และอาจตามด้วยแอททริบิวต์เงื่อนไข และแอคชัน(Actions) โดยในตัวของสเต็ท จะมีกิจกรรม(Activity) ที่ต้องทำอยู่ภายใน

อย่างที่ได้ทราบกันแล้วว่าแบบจำลองไดนามิคทำหน้าที่แสดงจังหวะและเวลาการเกิดขึ้นของเหตุการณ์ต่างๆ ในระบบ โดยเฉพาะกับระบบที่มีการโต้ตอบกับผู้ใช้เป็นหลักนั้น แบบจำลองไดนามิคจะมีความสำคัญมากเพราะจะใช้เป็นตัวแสดงขั้นตอนการโต้ตอบระหว่างระบบกับผู้ใช้

ขั้นตอนในการสร้างแบบจำลองไดนามิค เริ่มจากการเตรียมโครงร่างของเหตุการณ์ (Scenario)ขึ้นมาก่อน โครงร่างที่จัดเตรียมขึ้นมาไม่จำเป็นต้องครอบคลุมเหตุการณ์ทั้งหมดของระบบ แต่อย่างน้อยจะต้องครอบคลุมถึงเหตุการณ์การโต้ตอบของระบบในขั้นพื้นฐาน เมื่อได้จัดเตรียมโครงร่างของระบบเสร็จเรียบร้อยแล้ว หน้าที่ที่จะต้องทำต่อไปก็คือ ทำการแยกแยะและกำหนดเหตุการณ์เสียก่อน จากนั้นก็กำหนดเหตุการณ์เหล่านั้นให้กับออบเจ็กต์ที่เหมาะสม แล้วจึงมาจัดเรียงลำดับเหตุการณ์ให้เป็นสเต็ทไดอะแกรม ส่วนในขั้นสุดท้ายจะทำการเปรียบเทียบสเต็ทไดอะแกรมกับออบเจ็กต์อื่นๆ เพื่อตรวจสอบการส่ง เหตุการณ์ระหว่างออบเจ็กต์ให้มีความถูกต้อง

3.4.1 การเตรียมโครงร่างเหตุการณ์(Preparing Scenario)ทำได้โดยเขียนขั้นตอนการโต้ตอบระหว่างระบบและผู้ใช้เพื่อเป็นภาพแบบและแนวทางของระบบ โครงร่างที่เขียนขึ้นจะแสดงถึงการโต้ตอบหลักๆ ของระบบภาพแบบของหน้าจอและการแลกเปลี่ยนข้อมูลระหว่างระบบกับผู้ใช้เป็นต้น ข้อที่ควรคำนึงถึงของการเขียนโครงร่างเหตุการณ์ก็คือต้องเขียนขั้นตอนหลักๆ ลงไปในโครงร่างนั้นๆ เหตุการณ์ในระบบจะเกิดขึ้นได้ก็ต่อเมื่อมีการแลกเปลี่ยนข้อมูลระหว่างภายในหรือภายนอกออบเจ็กต์ โดยค่าของข้อมูลที่แลกเปลี่ยนกันนั้นเราเรียกว่า พารามิเตอร์(Parameter) ของเหตุการณ์

การเขียนโครงร่างของเหตุการณ์ที่ดีนั้นจะต้องมีทั้งในกรณีที่เหตุการณ์เกิดขึ้นอย่างปกติและโครงร่างของเหตุการณ์ที่เกิดขึ้นในกรณีพิเศษ เช่น การเกิดข้อผิดพลาดของระบบ การกระทำใดก็ตามที่เกิดจากออบเจ็กต์ที่ทำให้เกิดการเคลื่อนย้ายหรือส่งต่อข้อมูลจะถือว่าสิ่งนั้นเป็นเหตุการณ์

3.4.2 ภาพแบบการติดต่อ(Interface Format)โดยส่วนใหญ่แล้วการติดต่อจะแบ่งออกเป็นสองส่วน

ก) Application logic

ข) User interface การเขียนแบบจำลองไดนามิคจะทำให้เราสามารถกำหนดตรรกที่ใช้ในการควบคุมการติดต่อ(control logic) ของระบบได้

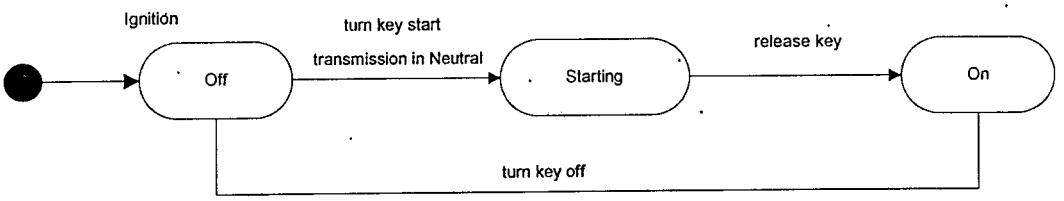
3.4.3 การเลือกกำหนดเหตุการณ์(Identifying Events) เหตุการณ์ที่กล่าวถึงนี้หมายถึง สัญญาณ(Signals),อินพุท(input) และการกระทำโดยหรือจากผู้ใช้ระบบ หรือจากสิ่งอื่นภายนอกระบบ การคำนวณใดๆ ก็ตามที่เกิดขึ้นในระบบไม่ถือว่าเป็นเหตุการณ์ เว้นแต่ว่าจะเป็นจุดที่ทำให้เกิดการเปลี่ยนแปลงการตัดสินใจเกี่ยวกับสิ่งอื่นๆ การกระทำใดๆก็ตามที่เกิดจากออบเจ็กต์แล้วมีผลทำให้เกิดการเคลื่อนย้ายหรือส่งต่อข้อมูลจะถือว่าสิ่งนั้นเป็นเหตุการณ์ เมื่อสามารถกำหนดเหตุการณ์ของระบบออกมาได้แล้ว เราก็จะมาทำการกำหนดเหตุการณ์เหล่านี้ให้กับออบเจ็กต์คลาส เหตุการณ์หนึ่ง ๆ อาจเป็นได้ทั้งอินพุทและเอ้าพุทให้กับออบเจ็กต์คลาสและในบางครั้งออบเจ็กต์คลาสสามารถที่จะส่งเหตุการณ์ย้อนกลับไปให้ตัวเองได้

โครงร่างของเหตุการณ์ที่เกิดขึ้นจะถูกนำมาแสดงในภาพของอีเว้นต์เทรค(event trace)ซึ่งเป็นการเขียนลำดับของเหตุการณ์ที่เกิดขึ้นระหว่างออบเจ็กต์ที่แตกต่างกัน โดยออบเจ็กต์แต่ละตัวจะถูกเขียนแยกไว้เป็นคอลัมน์ๆ จากการเขียนอิเว้น์เทรดจะทำให้เห็นลักษณะและผลกระทบของเหตุการณ์ที่เกิดขึ้นกับออบเจ็กต์ แผนผังที่จะขาดไม่ได้เลยของการทำแบบจำลองไดนามิคก็คืออีเว้นต์โฟล์ไดอะแกรม ที่มีหน้าที่แสดงเหตุการณ์ที่เกิดขึ้นระหว่างออบเจ็กต์คลาส โดยไม่ต้องกล่าวถึงลำดับเหตุการณ์ที่เกิดขึ้น ซึ่งทั้งเหตุการณ์และเส้นทาง(paths) ต่าง ๆ ที่เกิดขึ้นใน อิเว้นทโฟล์ไดอะแกรมจะเป็นตัวแสดงการควบคุมการไหลของระบบ(control flow)ที่เป็นไปได้

3.4.4 การสร้าง(State Diagram) จากอีเว้นต์เทรคไดอะแกรม(event trace diagram) โดยการเลือกไดอะแกรมสำหรับออบเจ็กต์คลาสที่ต้องการเขียนสเต็ทไดอะแกรมขึ้นมา แล้วนำเหตุการณ์จากไดอะแกรมในคอลัมน์เดียวกันมาเขียนลงในสเต็ทไดอะแกรม โดยช่วงเวลาระหว่างเหตุการณ์สองเหตุการณ์เราจะกำหนดให้เป็นสเต็ท และถ้าสเต็ทนั้นสามารถกำหนดชื่อได้เราก็จะกำหนดชื่อให้ แต่ถ้าไม่สามารถหาชื่อที่มีความหมายให้ได้ก็ไม่จำเป็นต้องกำหนดก็ได้ หลังจากการเขียนสเต็ท และเหตุการณ์เรียบร้อยแล้ว ก็ให้ทำการหาลูพ(loop)ที่เกิดขึ้นในไดอะแกรม นั่นคือให้สเต็ทแรกและสเต็ทสุดท้ายเป็นสเต็ทเดียวกัน สเต็ทไดอะแกรมที่ดีควรจะต้องแสดงรายละเอียดของการเกิดข้อผิดพลาดของระบบด้วย

3.4.5 ตรวจสอบความถูกต้อง การตรวจสอบความถูกต้องและความสมบูรณ์ของสเต็ทไดอะแกรมของออบเจ็กต์คลาสแต่ละตัว โดยที่ทุกๆเหตุการณ์จะต้องมีทั้งผู้ส่งและผู้รับ ในสเต็ทใด ๆ ถ้าไม่มีสเต็ทนำหน้าหรือตามหลังให้ทำการตรวจสอบสเต็ทนั้นให้รอบคอบว่า เป็นสเต็ทเริ่มต้นหรือสเต็ทสุดท้าย นอกจากนี้ยังต้องทดลองไล่ตามสเต็ทและเหตุการณ์ ในแผนผังเพื่อเปรียบเทียบว่าสอดคล้องกับโครงร่างของเหตุการณ์ที่ได้เขียนไว้ก่อนแล้วหรือไม่

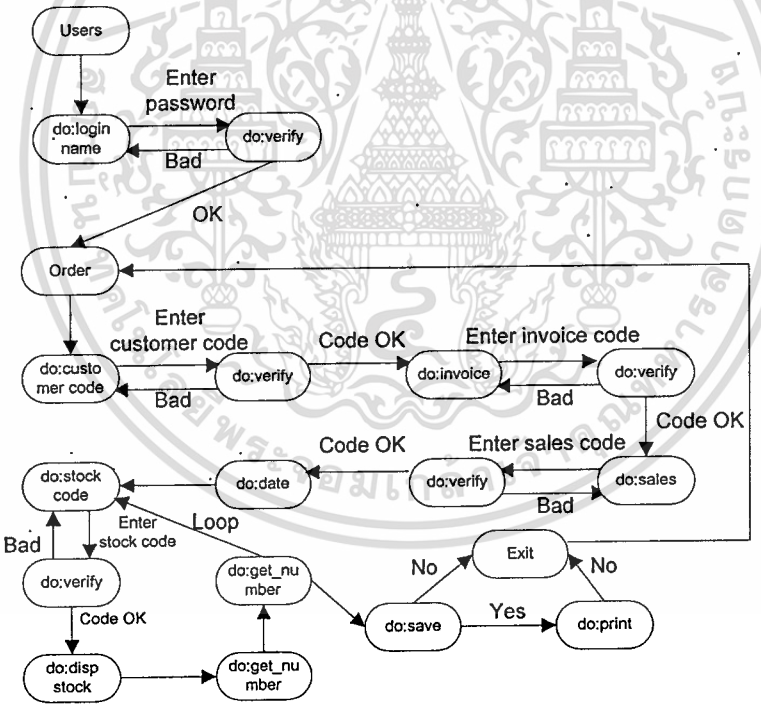
ภาพที่ 3.11



แสดง สเต็ทไดอะแกรม

แสดงความสัมพันธ์ระหว่างคลาสและสเต็ทไดอะแกรม วิธีการสร้างสเต็ทไดอะแกรม เราจะกำหนดให้ คลาส หนึ่งคลาสจะมีสเต็ทไดอะแกรม ประจำตัวหนึ่งอัน

ภาพที่ 3.12



Stock state diagrame

แสดงสเต็ทไดอะแกรมของระบบงานสต็อค

3.5 แบบจำลองฟังก์ชัน(Functional Modeling)

เป็นแบบจำลองที่ใช้แสดงวิธีการคำนวณค่าต่างๆในระบบว่ามีวิธีการคำนวณอย่างไร โดยไม่คำนึงถึงลำดับขั้นตอน การตัดสินใจหรือโครงสร้างของออบเจ็กต์ แบบจำลองที่เขียนขึ้นนี้ก็คือ ดาต้าโฟล์ไดอะแกรม(Data Flow Diagram) ซึ่งเรานำมาอธิบายความสัมพันธ์ของข้อมูลและฟังก์ชัน รวมถึงการประมวลผลต่างๆ(Process)ที่ปรากฏในไดอะแกรม เพื่อให้สอดคล้องกับกิจกรรมและการกระทำในสเต็ทไดอะแกรมที่เขียนขึ้น ส่วนข้อมูลที่ไหลผ่านในไดอะแกรม ก็จะสอดคล้องกับออบเจ็กต์หรือแอททริบิวต์ในออบเจ็กต์ไดอะแกรม การเขียนแบบจำลองฟังก์ชันที่เหมาะสมที่สุดนั้นควรจะเขียนหลังจากการเขียนแบบจำลองออบเจ็กต์และแบบจำลองไดนามิคแล้ว

3.5.1 การกำหนดค่าอินพุทและเอ้าพุท เริ่มต้นโดยการเขียนรายการอินพุทและเอ้าพุทที่จะเกิดขึ้น โดยนำมาจากข้อมูลระหว่างระบบและสิ่งต่างๆ ภายนอกระบบ ค่าข้อมูลเข้าและข้อมูลออกเหล่านี้จะถูกใช้เป็นพารามิเตอร์ของเหตุการณ์ที่เกิดขึ้น ค่าของอินพุทใด ๆ ถ้ามีผลกระทบต่อการไหลของข้อมูลเพียงอย่างเดียวเราจะไม่นำมาคิดเป็นอินพุท

3.5.2 การสร้างดาต้าโฟล์ไดอะแกรม โดยปกติแล้วดาต้าโฟล์ไดอะแกรมจะถูกสร้างขึ้นเป็นชั้นๆ ในชั้นแรกอาจประกอบไปด้วยโปรเซส(Process) เพียงโปรเซสเดียว ในแต่ละระดับของ ดาต้าโฟล์ไดอะแกรม เราจะทำการออกแบบย้อนกลับโดยเริ่มจากการดูเอ้าพุทก่อน แล้วค่อยมากำหนดฟังก์ชันที่ทำหน้าที่ให้เอ้าพุทนั้นออกมา และจากโปรเซสใหญ่ๆเราก็จะแตกออกเป็นโปรเซสย่อยลงไปจนกระทั่งสามารถทำการประยุกต์(Implement) ได้จากโปรเซสย่อย ๆ นั้น

3.5.3 อธิบายฟังก์ชันดาต้าโฟล์ไดอะแกรมที่ผ่านการตรวจสอบจนสมบูรณ์แล้ว ให้เขียนคำอธิบายให้กับฟังก์ชันแต่ละตัวซึ่งคำอธิบายอาจเป็นคำพูดธรรมดา สมการคณิตศาสตร์ ตารางการตัดสินใจ หรือ ในภาพแบบอื่นที่เหมาะสม โดยอธิบายความเหมาะสมของอินพุทและเอ้าพุท

3.5.4 การกำหนดข้อบังคับระหว่างออบเจ็กต์ เป็นการกำหนดข้อบังคับของข้อมูลซึ่งแบ่งได้เป็นสองชนิด

- preconditon เป็นการกำหนดข้อบังคับให้กับข้อมูลเข้า(Input) ให้กับฟังก์ชัน
- Postcondition เป็นการกำหนดข้อบังคับให้กับข้อมูลออก(Output)

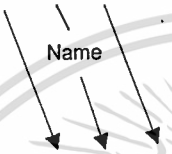
การกำหนดข้อบังคับนี้จะต้องทำควบคู่กันไประหว่างไดนามิคโมเดลและ ฟังก์ชันเนลโมเดล

3.5.5 การกำหนดขอบเขตของการอ๊อพติไมซ์(Optimize) การกำหนดขอบเขตของการอ๊อพติไมซ์ ในบางครั้งอาจเกิดความขัดแย้งขึ้นระหว่างจุดสองจุดหรือมากกว่า เพราะฉะนั้นก่อนทำการตัดสินใจก็ขอให้ทำการศึกษาของผลกระทบที่จะตามมา ดูผลได้ผลเสีย ดูลำดับความสำคัญเสียก่อนแล้วจึงตัดสินใจเลือกกระทำการ อ๊อพติไมซ์ ในส่วนที่ต้องการ

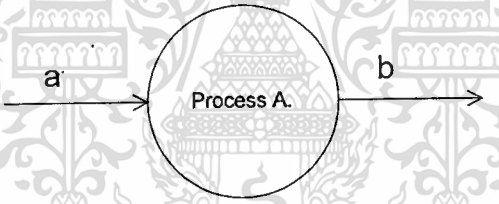
3.6 สัญญาลักษณ์ของดาต้าโฟล์ไดอะแกรม(Data Flow Diagram :DFD)

การออกแบบทางเทคนิคของดาต้าโฟล์นั้นจะต้องแปลงความคิดออกมาในภาพแบบของดาต้าโฟล์ไดอะแกรมซึ่งเป็นภาพแสดงการไหลและการเปลี่ยนแปลงของข้อมูลจากส่วนเริ่มต้นไปจนถึงผลลัพธ์ตามที่ต้องการ ซึ่งในแผนภาพของดาต้าโฟล์ไดอะแกรมจะประกอบไปด้วยสัญญาลักษณ์ต่างๆ ดังต่อไปนี้

3.6.1 **ลูกศร**ใช้แทนการๆหลของข้อมูลพร้อมกับชื่อของข้อมูลนั้นๆจะต้องกำกับไว้ในลักษณ์ดังนี้.



3.6.2.**วงกลม**ใช้แสดงกริยาการกระทำต่อข้อมูลที่ไหลเข้ามา โดยไม่คำนึงว่าจะเป็นการกระทำด้วยคนหรือไม่ก็ตาม และการได้มาซึ่งผลลัพธ์ที่จะไหลออกจากวงกลม ภายในวงกลมจะระบุคำสั้นๆ ที่ใช้แทนการกระทำต่อข้อมูลเพื่อให้ทราบว่า' ทำ - อะไร '

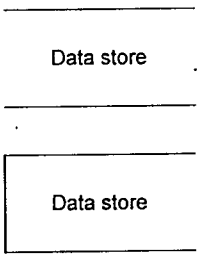


a = ข้อมูลเข้า

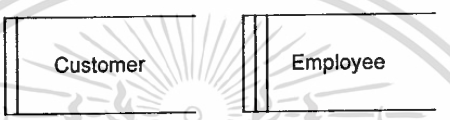
b = ข้อมูลออก

3.6.3 **ภาพสี่เหลี่ยม**.ใช้แทนนามที่อยู่ภายนอกระบบ ซึ่งเป็นกำเหนิดของข้อมูล โดยมีชื่ออยู่ภายในสี่เหลี่ยม ซึ่งชื่อนั้นสามารถซ้ำได้ถ้าจำเป็น หรือเมื่อต้องหารหมายถึงสิ่งเดียวกัน

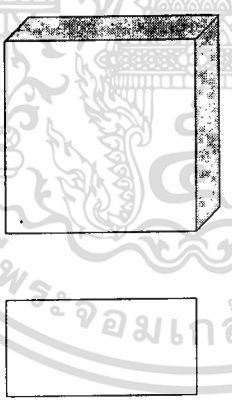
3.6.4 **สี่เหลี่ยมผืนผ้าปลายเปิด** เป็นตัวแทนของแหล่งเก็บข้อมูลหรือแฟ้มข้อมูล เสมือนเป็นตัวพักหรือช่วงขาดของการไหลของข้อมูลเพื่อนำไปเก็บเท่านั้น การกำหนดชื่อของแหล่งเก็บข้อมูลก็เช่นกันต้องอยู่ภายในสี่เหลี่ยม และชื่อการก็ควรเป็นชื่อที่ผู้ใช้คุ้นเคยด้วย การกำหนดชื่อของลูกศรที่แสดงการไหลเข้าและการไหลออกก็ควรให้ชัดเจน เพราะข้อมูลอาจถูกเปลี่ยนแปลงหรือไม่ก็ได้



3.6.5.**สัญญาลักษณ์เพิ่มเติม** จะใช้เติมลงในสัญญาลักษณ์ที่กล่าวมาข้างต้นเพื่อแสดงความเป็นสิ่งเดียวกัน แต่จะถูกกล่าวหลายๆครั้งในแผนภาพเช่น



และเครื่องหมายการแสดงการตัดกันของการไหลของข้อมูลและยังมีสัญญาลักษณ์อื่นๆ



ลำดับขั้นใน ดาต้าโฟล์ไดอะแกรม

ในการเขียนดาต้าโฟล์ไดอะแกรม นักวิเคราะห์ระบบจะต้องมองระบบจากภาพรวมก่อนจากนั้นมองลึกเข้าไปข้างในของระบบเสมือนกระบวนการท๊อป-ดาวน์(Top-Down)คือการมองจากภาพรวมก่อน เราจะเห็นขอบเขตของระบบและจุดใหญ่ๆภายในระบบ เมื่อมองลึกลงไปก็จะมองเห็นรายละเอียดของจุดใหญ่ๆนั้น ยิ่งมองลึกลงไปหรือมองใกล้มากขึ้น ก็จะยิ่งเห็นรายละเอียดระบบย่อยมากขึ้น ซึ่งขั้นตอนของดาต้าโฟล์ไดอะแกรมมีดังนี้

สร้างภาพระดับที่ 0

ให้คิดว่าระบบทั้งระบบเป็นโปรเซสหรือวงกลมวงหนึ่ง มีลูกศรแทนอินพุทและเอ้าพุทตามที่จำเป็นภาพนี้เป็นคอนเท็กไดอะแกรม(Context Diagram)ของระบบ

สร้างภาพระดับที่ 1

ให้แตกวงกลมที่ระดับ 0 เป็นวงกลมย่อย 2 - 5 วงแล้วแต่ความเหมาะสม

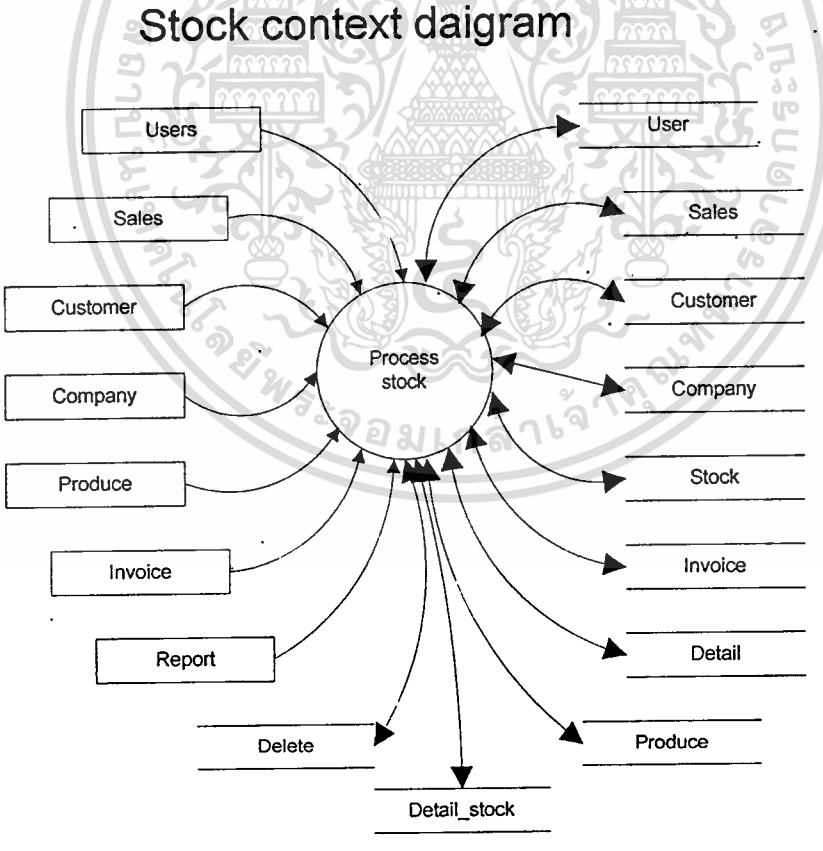
สร้างภาพระดับที่ 2

ให้แตกวงกลมที่ระดับ 1 เป็นวงกลมย่อยลงไปอีกเท่าที่จะทำได้

สร้างภาพระดับที่ 3

ถ้าจำเป็นก็ต้องตรวจดูว่า วงกลมใดที่อยู่ในระดับ 2 ยังมีความซับซ้อนที่จำเป็นต้องแตกย่อย ก็ต้องสร้างแผนภาพประกอบด้วยวงกลมย่อยแทนวงกลมนั้นให้ได้รายละเอียดสุดท้าย.

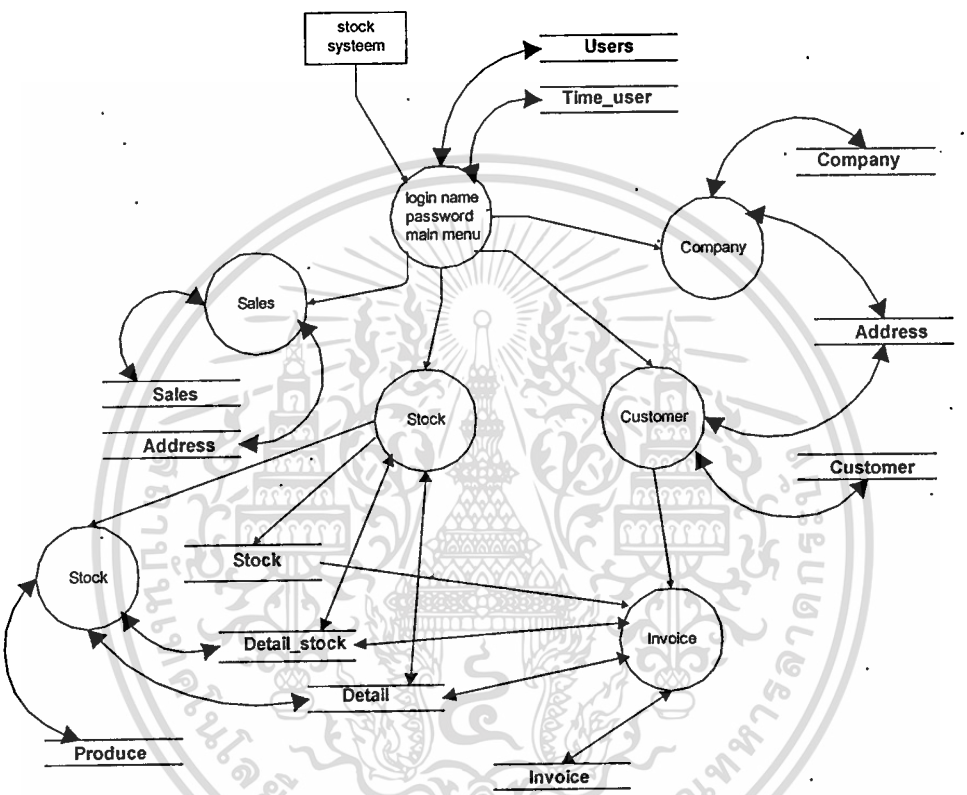
ภาพที่ 3.13



DFD #00 Stock diagrame

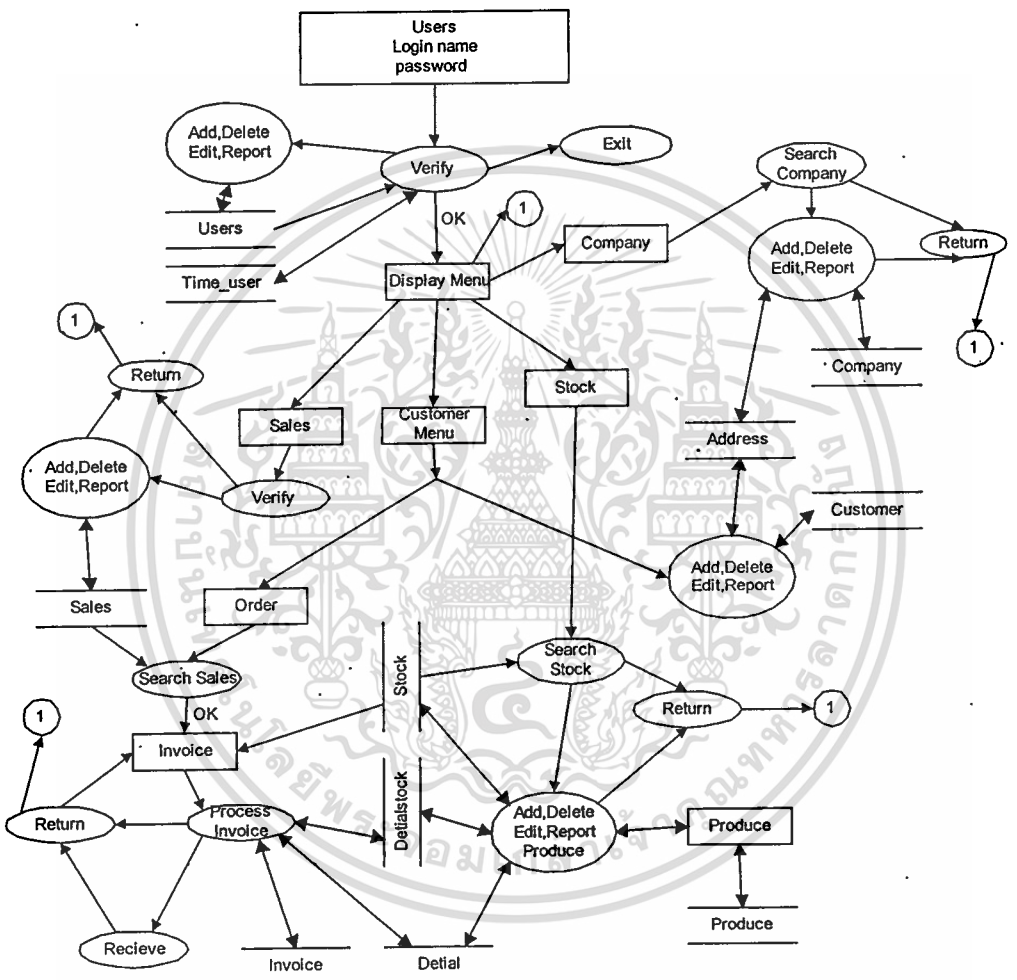
แสดง DFD -ขั้นตอนที่ 0

ภาพที่ 3.14



แสดง DFD -ขั้นตอนที่ 1

ภาพที่ 3.15

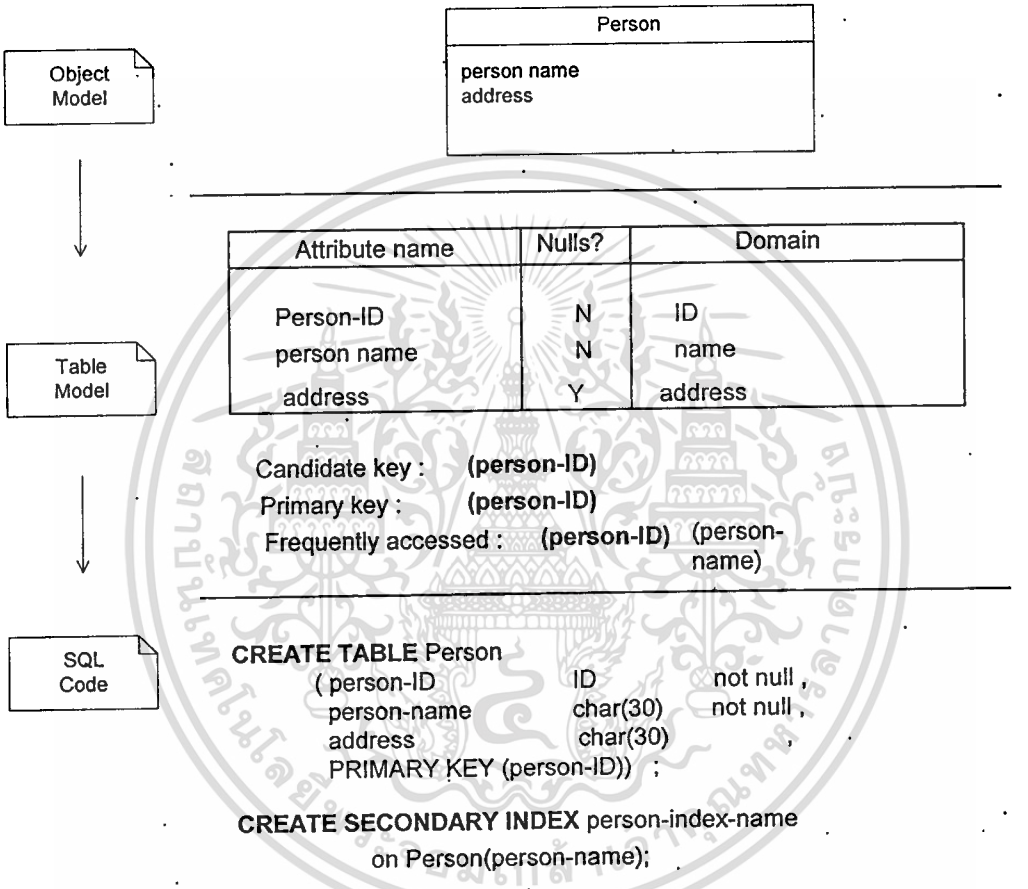


DFD #02 Stock diagrame

แสดง DFD -ขั้นตอนที่ 2

3.7 การแปลงออบเจ็กต์คลาสไปเป็นตาราง (Mapping Object Classes to Tables)[1]

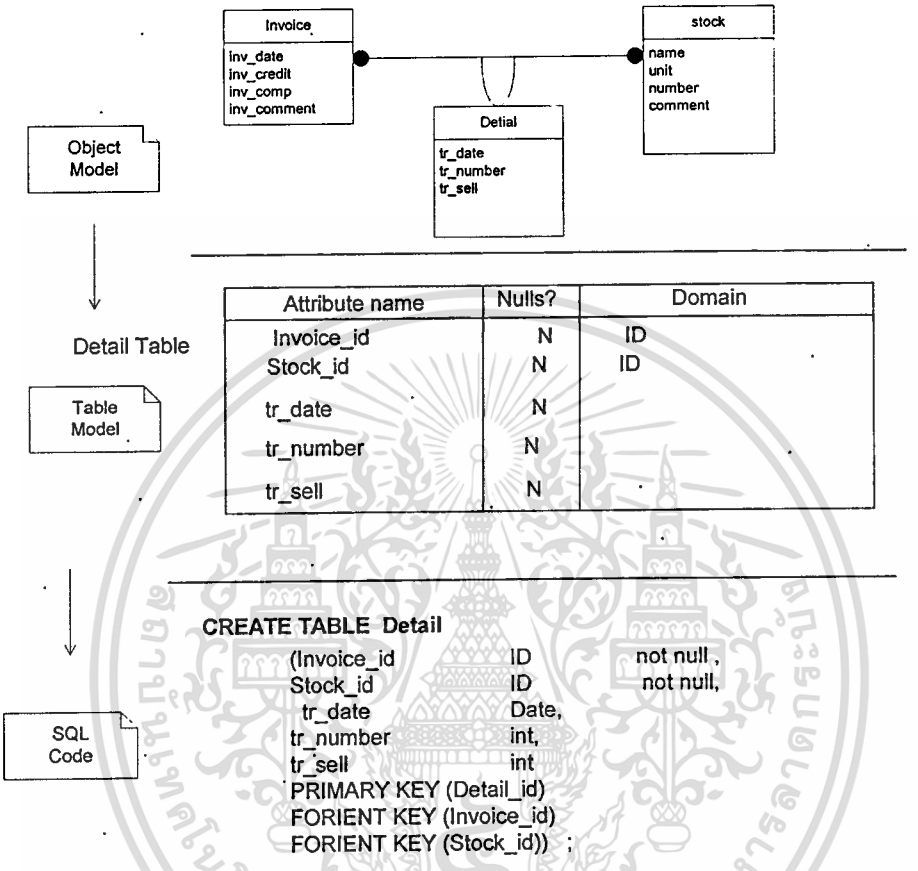
ภาพที่ 3.16



แสดงการแปลงจาก คลาสมาเป็นตาราง

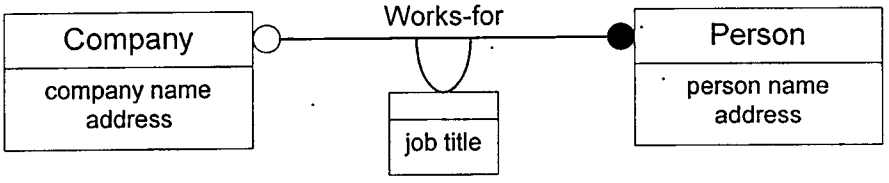
จากคลาสของ Person มี แอททริบิวต์ 2 ตัวคือ person name และ address เมื่อแปลงมาเป็นตารางแล้วจะได้ แอททริบิวต์เพิ่มอีก 1 ตัวคือ person-ID และเป็นคีย์หลัก(Primary Key)ด้วยและมีคีย์รองคือ Person-name

ภาพที่ 3.17



แสดงการแปลงคลาสแบบ Association มาเป็นตารางแบบ Many To Many

ภาพที่ 3.18



Works-for table

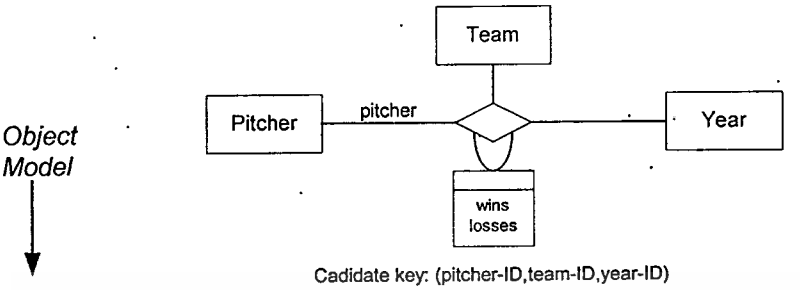
Attribute name	Nulls?	Domain
company-ID	N	ID
person-ID	N	ID
job-title	Y	title

Candidate key: (person-ID)
Primary key: (person-ID)
Frequently accessed: (company-ID) (person)

แสดงการแปลงคลาสแบบ Association มาเป็นตารางแบบ One To Many

```
CREATE TABLE Person (
Com_id          int,
Person_id       int,
Name            char(40),
Address         char(60) );
CREATE TABLE Work_for (
Com_id          int,
Person_id       int,
Job_title       char(50));
CREATE TABLE Computer(
Com_id          int,
Name            char(30),
Address         char(50));
```

ภาพที่ 3.19



Person table

Team table

Year table

Table Model

Ternary table

Attribute name	Nulls?	Domain
pitcher-ID	N	ID
team-ID	N	ID
year-ID	N	ID
wins	Y	games
losses	Y	games

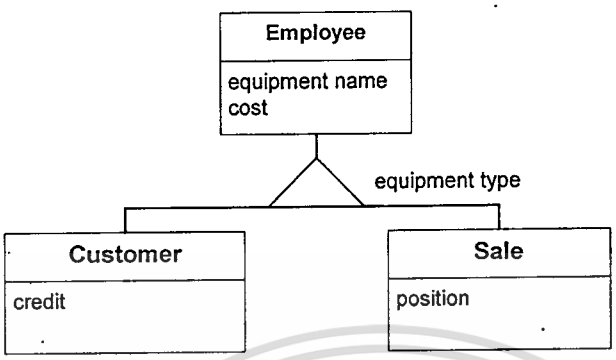
Candidate key: (pitcher-ID,team-ID,year-ID)
Primary key: (pitcher-ID,team-ID,year-ID)
Frequently accessed: (pitcher-ID) (team-ID) (year-ID)

SQL Code

```
CREATE TABLE  Person
              ( Person-ID      ID      not null  ,
                and other attribute ...  ,
              PRIMARY KEY (person-ID));
              also create table for Team and Year

              CREATE TABLE  Peson-Team-Year-ternary
                    ( pitcher-ID      ID      not null  ,
                      team-ID         ID      not null  ,
                      year-ID         ID      not null  ,
                      wins            integer
                      losses          integer
PRIMARY KEY  (pitcher-ID,  team-ID,  year-ID),
FOREIGN KEY  (pitcher-ID)  REFERENCES  Person,
FOREIGN KEY  (team-ID)     REFERENCES  Team,
FOREIGN KEY  (year-ID)     REFERENCES  Year),

also create indexes ...
```

แสดงแปลงคลาสแบบ การTernary Association มาเป็นตาราง

ภาพที่ 3.20

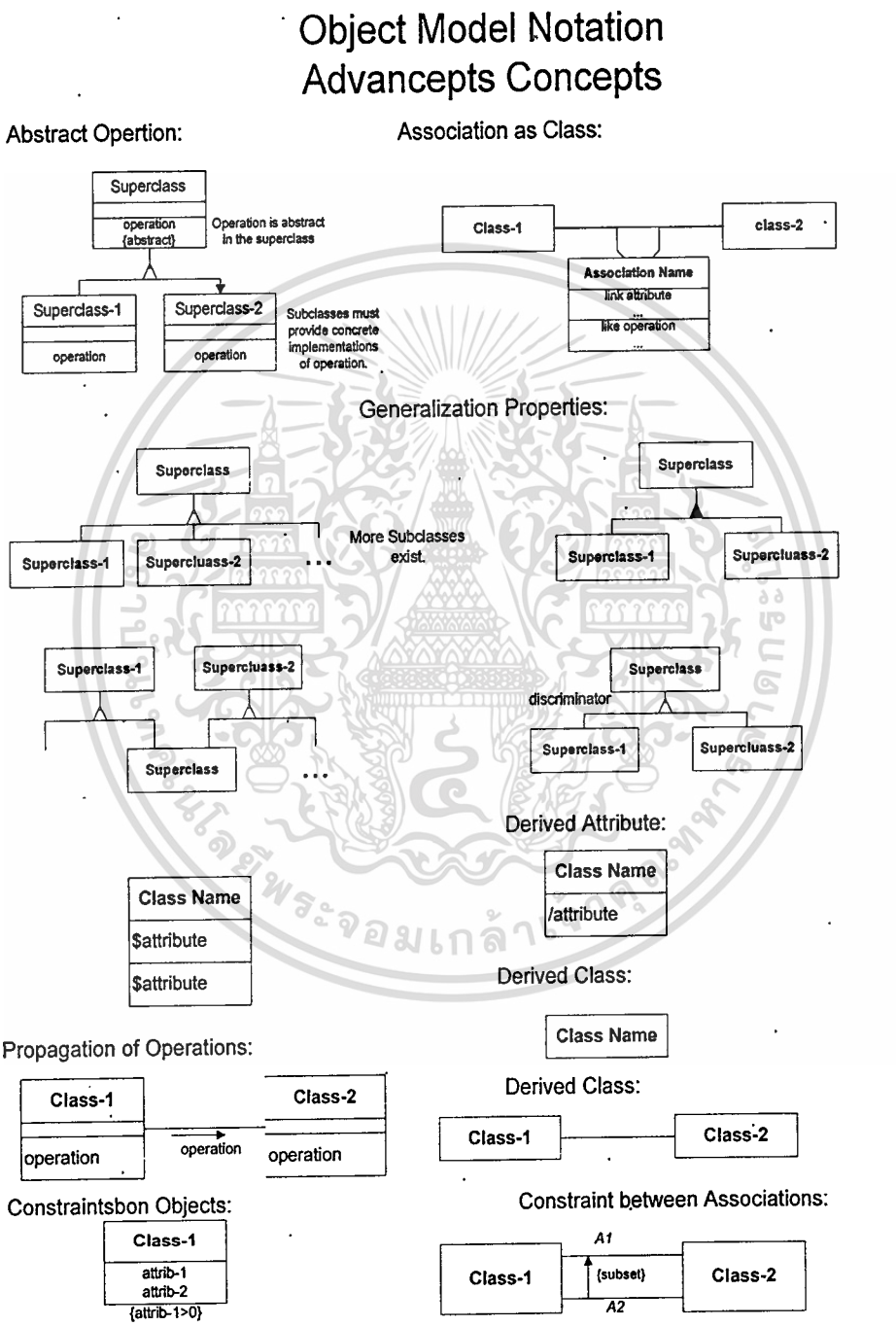


แสดงการแปลงคลาสแบบ Generallization มาเป็นตาราง

```
CREATE  TABLE  Employee (
    Emp_id          int,
    Name            char(40),
    Cost            int
);
CREATE TABLE Customer (
    Emp_id          int,
    Name            char(40),
    Cost            int,
    Credit          int
);
CREATE TABLE Sales (
    Emp_id          int,
    Name            char(40),
    Cost            int,
    Position int
);
```

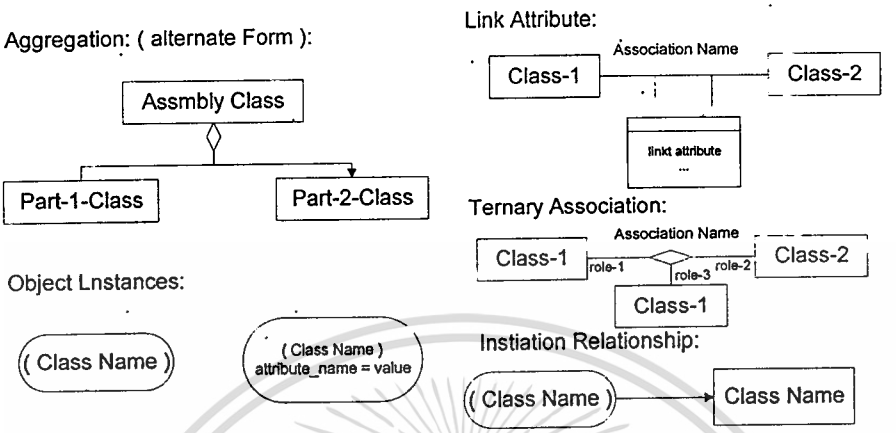
3.8 สัญลักษณ์ที่ใช้ในการออกแบบ Object model

ภาพที่ 3.21



แสดงภาพสัญลักษณ์ของออบเจ็กต์โมเดล

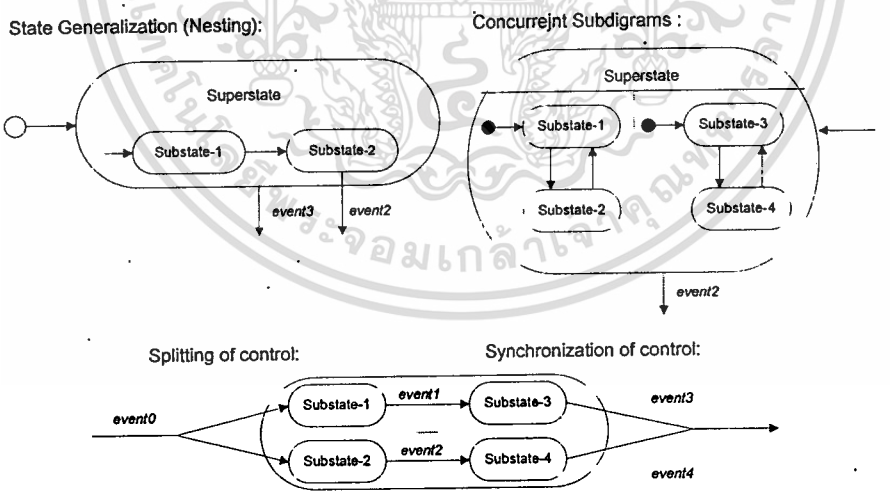
ภาพที่ 3.21 (ต่อ)



แสดงภาพสัญลักษณ์ของออบเจ็กต์โมเดล

3.9 แสดงสัญลักษณ์ที่ใช้ใน State Diagram

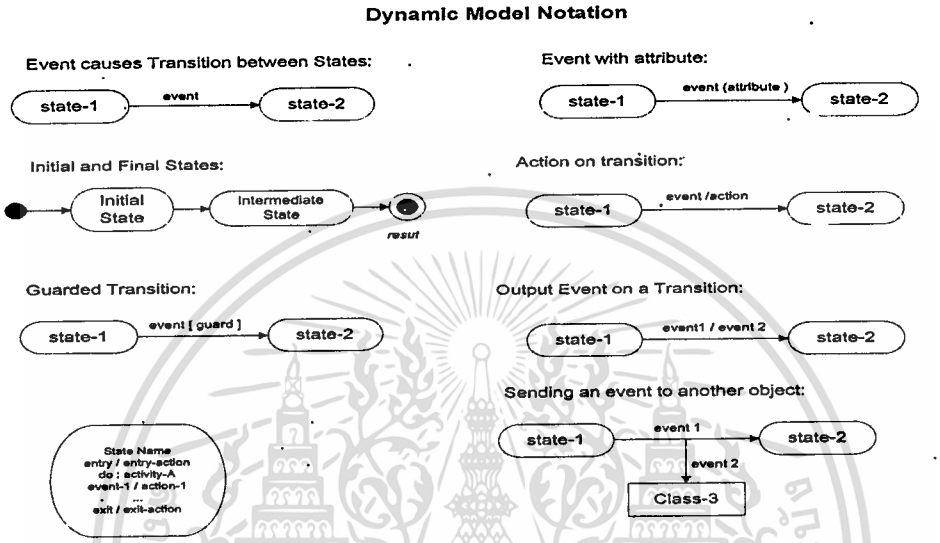
ภาพที่ 3.22



แสดงภาพสัญลักษณ์ของ State Diagram

3.10 สัญลักษณ์ที่ใช้ในการออกแบบ Dynamic Model

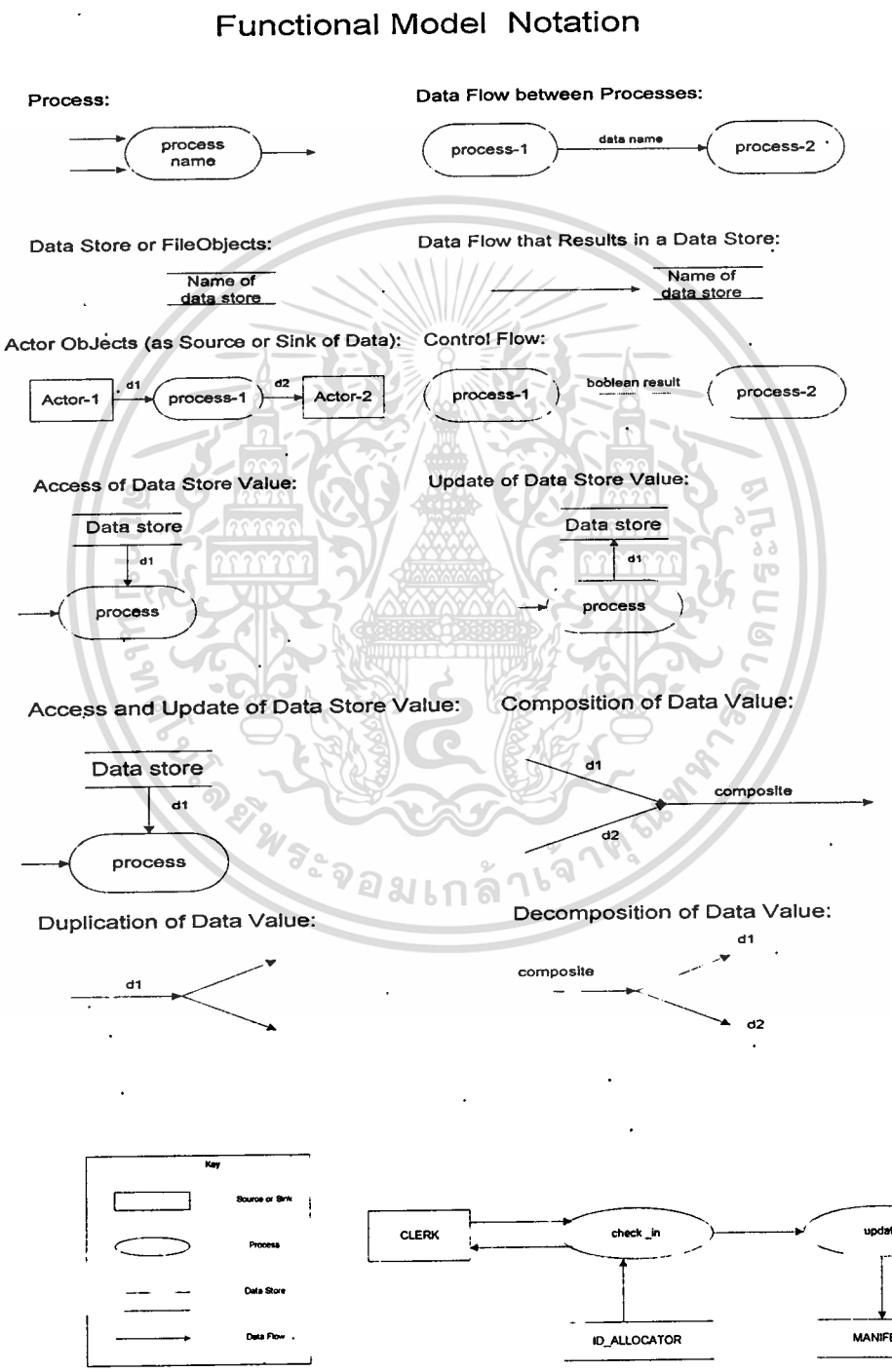
ภาพที่ 3.23



แสดงภาพสัญลักษณ์ของ Dynamic Model

3.10 สัญลักษณ์ที่ใช้ในการออกแบบ Function Model

ภาพที่ 3.24



แสดงภาพสัญลักษณ์ของ Function Model

3.11 การเปรียบเทียบระหว่างโอเอ็มทีกับระเบียบวิธีการแบบอื่นๆ

การสรุประเบียบวิธีการทางวิศวกรรมซอฟต์แวร์แบบอื่นๆ และเปรียบเทียบกับวิธีโอเอ็มที โดยที่เราจะเน้นสตรักเจอร์ดแอนนาไลซิส/สตรักเจอร์ดดีไซน์ (SA/SD, Structured Analysis / Structured Design), แจ็กสันสตรักเจอร์ดดีไซน์ (JSD, Jackson Structured Design) เราจะพิจารณาถึงจุดอ่อนจุดแข็งของวิธีการเหล่านี้

เป้าหมายหลักของเราในบทนี้ก็คือ เราจะพยายามชี้ให้เป็นความแตกต่างระหว่างโอเอ็มทีกับวิธีการอื่นๆ ซึ่งจะส่งผลให้เราสามารถเข้าใจโอเอ็มทีได้ดียิ่งขึ้น การกล่าวถึงระเบียบวิธีการแบบอื่นๆ นั้น เราจะกล่าวอย่างสั้นและสรุปเพื่อที่จะพอให้เห็นเป็นแนวทาง แต่ถ้าผู้อ่านมีความสนใจในรายละเอียดของระเบียบวิธีการแบบใดเป็นพิเศษ ก็สามารถค้นคว้าต่อได้จากบัญชีรายชื่อหนังสือหรือเอกสารอ้างอิงที่จะมีกำกับไว้ให้ภายในเอกสารอ้างอิง

3.11.1 สตรักเจอร์ดแอนนาไลซิส/สตรักเจอร์ดดีไซน์ (SA/SD)

ในปัจจุบันนี้ ระเบียบวิธีการในทางวิศวกรรมซอปต์แวร์จะอยู่บนพื้นฐานของด้าต้าโฟลว์ไดอะแกรม เป็นส่วนใหญ่ ในทางปฏิบัติแล้วก็จะมีดาต้าโฟลว์ไดอะแกรมอยู่หลากหลายแบบที่มีการนำมาใช้กัน เราจะพูดถึงสตรักเจอร์ดแอนนาไลซัส / สตรักเจอร์ดดีไซน์ในแง่ที่มันเป็นตัวแทนของระเบียบวิธีการที่ใช้ดาต้าโฟล์วไดอะแกรมแบบอื่นๆ

ทั้งระเบียบวิธีการแบบโอเอ็มทีและเอสเอ / เอสดี (SA/SD) ต่างก็รวมส่วนประกอบในการโมเดลลิ่งระบบที่คล้ายคลึงกัน วิธีการทั้งสองสนับสนุนการมองระบบในสามมุมมองเช่นกัน คือ ออพเจ็กต์, ไดนามิกส์ และฟังก์ชันนอลโมเดล สิ่งที่ต่างกันระหว่างวิธีการทั้งสองนี้ก็คือ โอเอ็มที่จะเน้นหนักที่ออพเจ็กต์โมเดล ซึ่งออพเจ็กต์จะเป็นตัวแทนสิ่งต่างๆ ในโลกแห่งความเป็นจริง ออพเจ็กต์และความสัมพันธ์ระหว่างกันจะถูกใช้เป็นฐานในการทำความเข้าใตพฤติกรรมทางไดนามิกส์ และฟังก์ชันนอลของระบบ ถ้าเปรียบเทียบกับเอสเอ / เอสดีแล้ว เอสเอ / เอสดีจะเน้นที่การแบ่งแยกย่อย (Decomposition) ระบบตามหน้าที่หรือฟังก์ชันนอลของระบบ ระบบจะถูกมองเสมือนเป็นสิ่งที่ใช้ฟังก์ชันแก่ผู้ใช้มากกว่าจะเป็นออพเจ็กต์

3.11.1.1 สรุปของวิธีแบบเอสเอ / เอสดี

เอสเอ / เอสดี มีสัญลักษณ์อยู่มากมายที่ใช้ในการอธิบายซอฟต์แวร์ให้เป็นระบบระเบียบในระหว่างขั้นตอนการวิเคราะห์ ดาต้าโฟล์วไดอะแกรม, โพรเซสสเปกซิฟิเคชัน (process specification) ดาต้าดิคชันนารี่ (data dictionary), สเตททรานชิชันไดอะแกรม (state transition diagram), และเอ็นติตี้รีเลชันชิพ ไดอะแกรมจะถูกใช้เพื่ออธิบายระบบที่ทำการวิเคราะห์ให้อยู่ในภาพแบบนามธรรม (logical) ในขั้นตอนการออกแบบ รายละเอียดของระบบจะถูกเพิ่มเติมลงไปในโมเดล

ต่างๆ ที่สร้างขั้นตอนการวิเคราะห์ และดาต้าโฟลว์ไดอะแกรมก็จะถูกแปลงไปเป็นสตรักเจอร์ชาร์ท (structure chart) และก็จะถูกแปลงไปเป็นโค๊ด (code) ของภาษาที่ใช้ในการโปรแกรม

ดาต้าโฟลว์ไดอะแกรมจะเป็นการจำลองของแปลงของข้อมูลเมื่อมันไหลผ่านระบบ และเอสเอ / เอสดีเน้นตรงเรื่องของดาต้าโฟลว์นี้มาก ดาต้าโฟลว์ไดอะแกรมจะประกอบด้วยโพรเซส, ดาต้าโฟลว์, แอ็กเตอร์ (actord) และดาต้าสตอร์ (data store) เริ่มต้นจากดาต้าโฟลว์ไดอะแกรมระดับบนสุด วิธีการแบบเอสเอ/เอสดี จะทำการรวนแตกโพรเซสที่มีความซับซ้อนลงไปเป็นซับไดอะแกรม (subdiagrams) จนกระทั่งโพรเซสที่ถูกแยกย่อนแล้วนั้นง่ายพอที่จะทำการสร้างได้ เมื่อโพรเซสสเปกซิฟิเคชันที่ทำขึ้นจะถูกเขียนสำหรับแต่ละโพรเซสในระดับต่ำสุดนั้นๆ โพรเซสสเปกซิฟิเคชันที่ทำขึ้นนี้อาจถูกอธิบายโดยใช้ดีซิชันเทเบิล (decision table), ซูโดโค๊ด (psoudocode) หรือเทคนิควิธีการอื่นก็ได้

ดาต้าดิกชันนารี่จะเป็นสิ่งที่ใช้เก็บรายละเอียดที่ไม่ได้ใส่ไว้ในดาต้าโฟลว์ไดอะแกรมดาต้าชันนารี่จะเป็บความหมายหรือคำอธิบายของดาต้าโฟลว์ ดาต้าสตอร์ และความหมายของชื่อกับคำสำคัญอื่นๆ

สเตททรานชิชันไดอะแกรมจะแสดงพฤติกรรมที่ขึ้นกับเวลาของระบบหรือโปรแกรมซึ่งจะคล้ายคลึงกับสเตทโมเดลของวิธีโอเอ็มที สเตทไดอะแกรมส่วนใหญ่จะอธิบายโพรเซสการควบคุมหรือไทม์มิ่ง (timimg) ของการปฏิบัติการฟังก์ชันและการเข้าถึงข้อมูลที่ถูกกระตุ้นโดยเหตุการณ์หรืออีเวนท์หนึ่งๆ

เอ็นติตี้รีเลชันชิพไดอะแกรมจะเน้นถึงความสัมพันธ์กันระหว่างดาต้าสตอลต่างๆ ที่เป็นอยู่ในโพรเซสสเปกซิฟิเคชัน แต่ละดาต้าอีลิเม้นท์(data element)ในอีอาร์ไดอะแกรมจะสัมพันธ์กับดาต้าสตอลอันหนึ่งบนดาต้าโฟลว์ไดอะแกรม

เครื่องมือต่างๆ ที่กล่าวมาแล้วนี้ ถูกใช้ในระหว่างกระบวนการวิเคราะห์ระบบหรือสตรักเจอร์ดแอนนาไลซิสสตรักเจอร์ดดีไซน์จะตามมาหลังจากสตรักเจอร์ดแอนนาไลซิส สตรักเจอร์ดดีไซน์นี้จะจัดการกับระบบหรือโปรแกรมในรายละเอียดที่ต่ำลงไปอีกจากขั้นตอนการวิเคราะห์ เป็นต้นว่า ในระหว่างการทำสตรักเจอร์ดดีไซน์, โพเซสบนดาต้าโผลว์ไดอะแกรมจะถูกจับรวบกันเป็นกลุ่มของงาน (tasks) และถูกกำหนดไปยังโพรเซสของระบบปฏิบัติการและซีพียู (CPU) ดาต้าโฟลว์ไดอะแกรมจะถูกแปลงไปเป็นฟังก์ชันในภาษาที่ใช้ในการโปรแกรมและสตรักเจอร์ดชาร์ทจะถูกสร้างเพื่อแสดงให้เห็นถึงลำดับการเรียกใช้ในภาพแบบโพรซีเดอร์คอลล์ทรี (procedure call tree)

3.11.1.2 การเปรียบเทียบกับโอเอ็มที

เอสเอ/เอสดี กับโอเอ็มที่มีสิ่งที่คล้ายกันอยู่หลายอย่าง ระเบียบวิธีการทั้งสองใช้โครงสร้างการจำลองระบบที่คล้ายคลึงกัน อีกทั้งยังให้มุมมองระบบทั้งสามมุมมองเหมือนกันด้วย ความแตกต่างระหว่างเอสเอ/เอสดี กับ โอเอ็มทีนั้น โดยหลักๆ ก็จะเป็นในเรื่องของสไตล์และการเน้นหนักในวิธีเอส/เอดี นี้นการเน้นหนักจะตกอยู่ที่ฟังก์ชันนอลดมเดล รองลงมาก็จะเป็นไดนามิกส์โมเดล และออพเจ็กต์โมเดลเป็นอันดับสุดท้าน เมื่อเทียบกับโอเอ็มทีแล้ว โอเอ็มทีจะเน้นที่ตัวออพเจ็กต์โมเดลมากที่สุด ไดมามิกส์และฟังก์ชันนอลดมเดลน้อยรองกันลงมาตามลำดับ

เอสเอ/เอสดี จะจัดกระบบในภาพแบบโพรซีเดอร์ประกอบเข้าด้วยกัน ในขณะที่โอเอ็มทีจะจัดระบบอยู่ในภาพแบบของวัตถุในโลกแห่งความเป็นจริง หรือออพเจ็กต์ในจิตนาการที่อยู่ในความคิดเกี่ยวกับโลกแห่งความเป็นจริงของผู้ใช้ เนื่องจากการเปลี่ยนรีควายเม้นต์มักจะเป็นการเปลี่ยนแปลงทางฟังก์ชันของระบบมากกว่าในทางวัตถุหรือออพเจ็กต์ของระบบ ดังนั้นการเปลี่ยนแปลงจึงมักจะมีผลกระทบอย่างมากในการออกแบบที่อยู่บนพื้นฐานของโพรซีเดอร์ เมื่อเปรียบเทียบแล้วการเปลี่ยนแปลงทางฟังก์ชันถูกทำให้เหมาะสมในการออกแบบแบบออพเจ็กต์โอเรียนเต็ด โดยการเพิ่มเติมหรือเปลี่ยนแปลงโอเปอร์เรชัน ปล่อยให้โครงสร้างพื้นฐานของออพเจ็กต์ไม่ถูกเปลี่ยนแปลงเอสเอ/เอสดี จึงเหมาะสมสำหรับปัญหาที่ฟังก์ชันมีความสำคัญมากกว่าและซับซ้อนกว่าตัวข้อมูล ที่เป็นเช่นนี้ก็เนื่องจากว่าเอสเอ/เอสดี ได้ถึงคิดขึ้นบนสมมุติฐานของปัญหาประเภทดังกล่าว

เอสเอ/เอสดีดีไซด์ (ผลกาลออกแบบโดยวิธีเอสเอ/เอสดี) จะมีขอบเขตของระบบที่ถูกนิยามไว้อย่างชัดเจน คือแบ่งระหว่างซอฟต์แวร์โพรซิเดอร์ที่ต้องติดต่อกับโลกแห่งความเป็นจริง โครงสร้างของเอสเอ/เอสดี ดีไซด์จะถูกสร้างขึ้นเป็นส่วนๆ จากขอบเขตของระบบ ดังนั้นจึงเป็นการยากที่จะขยายออพเจ็กต์โอเรียนเต็ดดีไซน์ให้เป็นไปตามขอบเขตใหม่ มันจะเป็นการง่ายกว่าจะขยายออพเจ็กต์โอเรียนเต็ดดีไซน์ โดยอาจเพิ่มออพเจ็กต์และรีเลชันชิพใกล้ๆ กับขอบเขตเพื่อที่จะแสดงออพเจ็กต์ที่มีอยู่ก่อนหน้านี้แต่อยู่ภายนอกของขอบเขตเดิม ออพเจ็กต์โอเรียนเต็ดดีไซน์จึงเหมาะสมกว่าเมื่อมีเหตุการณ์ที่ต้องเปลี่ยนแปลงและสามารถขยายขอบเขตระบบได้ง่ายกว่าด้วย

การเปรียบกันอย่างตรงไปตรงมาระหว่างออพเจ็กต์ในออพเจ็กต์โอเรียนเต็ดดีไซน์และออพเจ็กต์ในขอบเขตของปัญหาจริง ทำให้ระบบที่อธิบายนี้สามารถทำความเข้าใจได้ง่ายจากโมเดลที่ใช้แสดงระบบนั้น สิ่งนี้ทำให้ดำไซน์สามารถเข้าใจได้ดีกว่าและเป็นธรรมชาติมากกว่าและทำให้การตรวจสอบระหว่างรีควายเมนท์และซอฟต์แวร์โค้ดเป็นไปได้อย่างสะดวก นอกจากนั้นการเปรียบเทียบกับปัญหาจริงยังทำให้คนที่ไม่ได้ร่วมอยู่ในทีมที่วิเคราะห์ออกแบบสามารถทำความเข้าใจกับโมเดลของระบบนั้นได้ง่ายขึ้นด้วย

ในเอสเอ/เอสดี การแยกย่อยโพรเซสหนึ่งๆ ออกเป็นซับโพรเซสเป็นอะไรที่ตามแต่ผู้ออกแบบจะนึกคิดเอาเอง ผู้ออกแบบคนละคนกันสามารถแยกย่อยโพรเซสให้ออกมาในแบบที่ไม่เหมือนกันได้ ในออพเจ็กต์โอเรียนเต็ดดีไซน์ การแยกย่อยจะอยู่บนพื้นฐานของออพเจ็กต์หรือวัตถุในสภาวะแวดล้อมของปัญหา ดังนั้นผู้พัฒนาที่ต่างคนหรือต่างชุดกันก็มีแนวโน้มที่จะกำหนดออพเจ็กต์ได้เหมือนๆ กัน สิ่งนี้ทำให้เกิดการที่สามารถนำเอาส่วนประกอบจากระบบหนึ่ง นำกลับมาใช้ได้อีกในระบบหนึ่ง

วิธีการแบบออพเจ็กต์โอเรียนเต็ดจะสามารถรวมฐานข้อมูลกับโปรแกรมมิ่งโค็ดได้ดีกว่าโดยเป็นแบบที่เหมือนๆ กันออพเจ็กต์สามารถโมเดลได้ทั้งฐานข้อมูลและโครงสร้างการโปรแกรม การวิจัยเกี่ยวกับออพเจ็กต์โอเรียนเต็ดดาต้าเบสในอนาคต จะยิ่งช่วยให้สิ่งที่กล่าวถึงนี้เป็นไปได้ดี และชัดเจนยิ่งขึ้น เมื่อเปรียบเทียบกับวิธีการดีไซด์แบบโพรซิเดอรร์รอลแล้ว วิธีการแบบโพรซิเดอร์รอลจะอุ้ยอ้ายมากกว่าในการจัดการกับฐานข้อมูล มันเป็นการยากที่จะรวมโปรแกรมมิ่งโค็ดที่ถูกจัด (organize) เพื่อฟังก์ชันกับฐานข้อมูลที่ถูกจัดเพื่อตัวข้อมูล

มีอยู่หลายเหตุผลที่ว่าทำไมวิธีการแบบดาต้าโฟลว์จึงเป็นที่ยิยมใช้กันอย่างแพร่หลาย อย่างแรกก็คือโปรแกรมเมอร์มีแนวโน้มที่จะคิดในแบบที่เป็นฟังก์ชันมากกว่าดังนั้นระเบียบวิธีที่มีพื้นฐานอยู่บนฟังก์ชันหรือโพรเซสอย่างดาต้าโฟลว์ไดอะแกรมจึงเรียนรู้ได้ง่ายกว่า อีกเหตุผลหนึ่งก็เป็นไปตามประวัติศาสตร์ที่เอสเอ/เอสดี เป็นระเบียบวิธีที่เป็นทางการ (formal) วิธีแรกที่ถูกคิดขึ้นมาเพื่อการพัฒนาซอฟต์แวร์หรือระบบ ดังนั้นจึงมีผู้ใช้วิธีการนี้อยู่เดิมแล้วมาก แต่ในอนาคตเมื่อวิธีการแบบออพเจ็กต์โอเรียนเต็ดถูกพัฒนาให้ดีมากขึ้น เราเชื่อว่าเทคโนโลยีแบบออพเจ็กต์โอเรียนเต็ดจะถูกนำมาใช้ในการวิเคราะห์ออกแบบและสร้างระบบอย่างแพร่หลายมากขึ้น

3.11.2. แจ็กสันสตรักเจอร์ดดีวีลอพเมนต์ (เจเอสดี)

แจ็กสันสตรักเจอร์ดดีวีลอพเมนต์ หรือเจเอสดีนี้ก็เป็นอีกระเบียบวิธีหนึ่งที่ถูกพัฒนาขึ้นเพื่อนำมาใช้ในการวิเคราะห์และออกแบบระบบ ซึ่งวิธีเจเอสดีนี้มีลักษณะแตกต่างออกไปจากเอกเอ/เอสดี และโอเอ็มที่มากพอสมควร ระเบียบวิธีแบบเจเอสดีถูกพัฒนาโดย Michel Jackson และได้รับความนิยามมากในประเทศแถบยุโรป เจเอสดีจะไม่แยกขั้นตอนระหว่างการวิเคราะห์และออกแบบระบบ แต่ได้รวมทั้งสองขั้นตอนเข้าไว้ด้วยกัน แล้วกำหนดให้เรียกขั้นตอนที่รวมกันนี้ว่าขั้นตอนการกำหนดรายละเอียดของระบบ (Specification) เจเอสดีแบ่งการพัฒนาระบบออกเป็น 2 ขั้นตอน คือ การกำหนดรายละเอียดของระบบ และจากนั้นก็เป็นการสร้างระบบ ในขั้นตอนแรก เจเอสดีจะวิเคราะห์ว่าระบบนี้ "คืออะไร (What)" แล้วตอนหลังจึงพิจารณาว่าจะแก้ไขปัญหาได้ "อย่างไร (How)" เจเอสดีถูกพัฒนาขึ้นโดยเฉพาะสำหรับแอพพลิเคชันที่ไทม์มิ่งมีความสำคัญ

เจเอสดีใช้ดมเดลที่เป็นแผนภาพเหมือนกันกับเอสเอ/เอสดี, โอเอ็มที และวิธีอื่นใช้แต่เราขอที่จะไม่แสดงแผนภาพของเจเอสดีไว้ ณ ที่นี้ ทั้งนี้เนื่องจากแผนภาพไม่ได้เป็นจุดที่ทำให้เราเห็นถึงข้อดีของเจเอสดีแต่อย่างใด ในความเป็นจริงแล้วเราคิดว่าเจเอสดี มีลักษณะที่เป็นกราฟฟิกหรือแผนภาพน้อยกว่าเอสเอ/เอสดี หรือ โอเอ็มทีเสียด้วยซ้ำไป

3.11.2.1 สรุปวิธีการแบบเจเอสดี

วิธีการแบบเจเอสดีเริ่มต้นด้วยการพิจารณาโลกแห่งความเป็นจริง แม้ว่าจุดมุ่งหมายของระบบจะเพื่อการใช้ฟังก์ชันนอลลิตี้ (Functionality) หรือสิ่งที่ให้การทำงานก็ตามแต่วิธีการของแจ็กสันคิดว่าในลำดับแรก ผู้ที่จะวิเคราะห์และออกแบบจะต้องทำการพิจารณาว่าฟังก์ชันนอลลิตี้ดังกล่าวนั้นจะเหมาะสมกับโลกแห่งความเป็นจริงอย่างไร เจเอสดีโมเดลจะอธิบายโลกแห่งความเป็นจริงในภาพแบบของเอ็นติตี้ (Enitiy), แอ็คชัน (Action) และลำดับของแอ็คชันเหล่านั้น โดยปกติแล้วเอ็นติตี้จะปรากฏในลักษณะที่เป็นคำนามในคำบรรยายความต้องการของระบบและแอ็คชันจะปรากฏในภาพของคำกริยา การพัฒนาซอฟต์แวร์ด้วยเจเอสดีจะประกอบไปด้วยขั้นตอน 6 ขั้นตอน ที่เรียงลำดับกันดังนี้ ขั้นตอนเอ็นติตี้แอ็คชัน (Entity action step), ขั้นตอนเอ็นติตี้สตรักเจอร์ (Entity structure step), ขั้นตอนอินนิเทียลโมเดล (Initial model step), ขั้นตอนฟังก์ชัน (Function step), ขั้นตอนซิสเต็มไทม์มิ่ง (System timing step) และขั้นตอนในการสร้าง (Implementation step)

ระหว่างขั้นตอนเอ็นติตี้แอคชัน ผู้พัฒนาจะเขียนรายการของเอ็นติตี้และแอ็คชันสำหรับส่วนของโลกแห่งความเป็นจริงโดยใช้จุดมุ่งหมายรวมของระบบทั้งหมดเป็นแนวทางในการเลือกเอ็นติตี้และแอ็คชัน อินพุตสำหรับขั้นตอนเอ็นติตี้แอ็คชัน คือคำบรรยายความต้องการของระบบและเอาร์พุตจะเป็นรายการของเอ็นติตี้และแอ็คชัน

ในบทความของแจ็กสัน [Jackson-83] ได้แสดงตัวอย่างไว้อย่างตัวอย่างซึ่งตัวอย่างหนึ่งก็เป็นการออกแบบระบบควรคุมลิฟท์ เราจะอ้างอิงกับตัวอย่างระบบลิฟท์ของแจ็กสันในบทนี้ด้วย ระบบควรคุมลิฟท์นี้จะควรคุมลิฟท์ 2 ตัวที่คอยให้บริการชั้น 6 ชั้น ในลิฟท์แต่ละตัวจะมีปุ่มอยู่ 6 ปุ่มซึ่งแต่ละปุ่มจะสำหรับชั้นแต่ละชั้นที่แต่ละชั้นจะมีปุ่มให้กดขึ้นและลงในบริเวณที่คอยลิฟท์ด้วย แจ็กสันกำหนดเอ็นติตี้ขึ้นมาสองเอ็นติตี้สำหรับตัวอย่างการควบคุมลิฟท์นี้ - - ปุ่ม และลิฟท์ เขายังกำหนดแอ็คชันขึ้นอีกสามอัน คือ กดปุ่ม, ลิฟท์ถึงชั้นที่ N และลิฟท์ออกจากชั้นที่ N

แอ็คชันจะเป็นสิ่งเกิดขึ้นในโลกแห่งความเป็นจริง และไม่ได้เป็นสิ่งที่มนุษย์สร้างขึ้นเอง (Artifact) แอ็คชันจะเกิดขึ้น ณ จุดหนึ่งของเวลาและเป็นอะตอมมิก (Atomic) ซึ่งก็คือไม่สามารถแบ่งแยกแอ็คชันต่อไปได้ ขั้นตอนเอ็นติตี้สตรักเจอร์จะทำการจัดลำดับของแอ็คชันบางส่วนโดยใช้เวลาเป็นตัวจัดลำดับ ระบบควบคุมลิฟท์เป็นตัวอย่างที่แสดงให้เห็นความสำคัญของการจัดลำดับแอ็คชัน มันยอมรับได้สำหรับลิฟท์ที่จะมาถึงชั้น 3 แล้วออกจากชั้นที่ 3, มาถึงชั้นที่ 2 แล้วออก

จากชั้นที่ 2 และอื่นๆ แต่มันจะเป็นไปไม่ได้ที่จะเกิดแอ็คชันแบบที่ลิฟท์มาถึงชั้นที่ N สองครั้งติดต่อกัน ขั้นตอนอินนิเทียลโมเดลจะกล่าวถึงการเชื่อมต่อระหว่างโลกแห่งความเป็นจริง และแบบจำลองว่าเชื่อมต่อกันอย่างไร เจเอสดีสนับสนุนการเชื่อมต่อกันทั้งแบบสเตทเวคเตอร์ (State-Vector Connection) และดาต้าสตรีม (Data Stream Connection) ซึ่งจะได้อธิบายในย่อหน้าต่อไป

ระบบการควบคุมลิฟท์แสดงให้เห็นการเชื่อมต่อกันแบบสเตทเวคเตอร์ อะไรจะเกิดขึ้นเมื่อมีบางคนกดปุ่มขึ้น 5 ครั้งติดต่อกัน? ผู้ใช้ลิฟท์ไม่ได้ต้องการให้ระบบควบคุมลิฟท์จำการกดแต่ละครั้งแล้วส่งลิฟท์มา 5 ครั้งตามการกดนั้น แต่แทนที่จะเป็นเช่นนั้น การกดปุ่มขึ้นจะทำการตั้งแฟล็กการขึ้น (Up-flag) ให้เป็นจริง (True) การกดปุ่มขึ้นหายๆ ครั้งจะไม่มีผลกระทบอื่นใด ระบบคอมพิวเตอร์ขอโมเดลเจเอสดีจะไม่รับรู้ถึงจำนวนการกดปุ่ม และจะติดต่อกับโลกแห่งความเป็นจริงโดยผ่านแฟล็กการขึ้นเท่านั้น แจ็กสันเรียกแฟล็กการขึ้นนี้ว่าเป็นการติดต่อแบบสเตทเวคเตอร์

พริ้นท์บัฟเฟอร์ของคอมพิวเตอร์ (Computer Print Buffer) จะแสดงให้เป็นการเชื่อมต่อแบบดาต้าสตรีมผู้ใช้คอมพิวเตอร์ไม่ต้องการให้เกิดการสูญหายของข้อมูลขึ้นถ้าคอมพิวเตอร์สามารถส่งข้อมูลได้เร็วกว่าที่เครื่องพิมพ์จะสามารถพิมพ์ได้ พริ้นบัฟเฟอร์จะเป็นสิ่งที่กั้นระหว่างเครื่องคอมพิวเตอร์กับเครื่องพิมพ์และทำให้การประมวลผลของซีพียูกับการพิมพ์สามารถซ้อนเหลื่อมกันได้ พริ้นบัฟเฟอร์จะมีขนาดที่จำกัดและเมื่อบัฟเฟอร์เต็ม คอมพิวเตอร์จะต้องรอก่อนที่จะส่งข้อมูลต่อไปได้อีก การติดต่อแบบดาต้าสตรีมของเจเอสดีก็จะเป็นในลักษณะเดียวกันกับบัฟเฟอร์ที่มีขนาดจำกัดขั้นตอนอินนิเทียบของเจเอสดีจะไม่ได้กล่าวถึงข้อจำกัดทางกายภาพของบัฟเฟอร์เอง

ขั้นตอนฟังก์ชันจะใช้ซูโดโค๊ด (Pseudocode) ในการอธิบายผลลัพท์หรือเอาร์พุตของแอ็คชันในตอนท้ายของขั้นตอนนี้ผู้พัฒนาจะมีสเปคซิฟิเคชันที่สมบูรณ์ของระบบ ในตัวอย่างระบบลิฟท์ การเปิดปิดที่แสดงว่าลิฟท์ได้มาถึงแต่ละชั้นจะเป็นฟังก์ชันที่จะต้องกำหนด

ขั้นตอนซิสเต็มไทม์มิ่งจะพิจารณาว่าจะอนุญาติให้โมเดลห่อหุ้มโลกแห่งความเป็นจริงได้มากน้อยเพียงใด สำหรับส่วนใหญ่แล้วผลลัพท์ของขั้นตอนไทม์มิ่งจะเป็นกลุ่มของบันทึกที่ไม่เป็นทางการ (Informal note) ในเรื่องเกี่ยวกับข้อจำกัดทางประสิทธิภาพระบบ (Performance constraints) เป็นต้นว่าระบบควบคุมลิฟท์ต้องตรวจจับว่าเมื่อใดปุ่มถูกกดลงและปล่อยขึ้น ระยะเวลายาวนานเท่าใดที่ผู้ใช้กดปุ่มให้หน้าสัมผัสต่อกัน มันจะน่ารำคาญถ้าดกปุ่มไปแล้วระบบกลับไม่มีการตอบสนองแต่อย่างใดดังนั้นเราจะต้องกำหนดเรื่องพวกนี้ด้วยระยะเวลาที่กำหนดไว้สั้นจะหมายความว่าระบบควบคุมมักจะตรวจพบเซอร์วิสรีเควส (Service request) อย่างไรก็ตามถ้าการกดปุ่มถูกตรวจจับโดยใช้วิธีการแบบพูลลิ่ง(Polling scheme) ค่าที่ต่ำจะหมายถึงว่า ต้องใช้ทรัพยากรของเครื่องคอมพิวเตอร์มากขึ้นในการตรวจสอบ ผู้ออกแบบจะต้องหาจุดสมดุลย์ของประสิทธิภาพ (Performance trade-offs) อย่างชัดเจนในระหว่างขั้นตอนซิสเต็มไทมมิ่งนี้

ขั้นตอนการสร้าง หรืออิมพลีเมนต์เทชันจะมุ่งความสนใจในปัญหาของการจัดกระบวนการ (Process scheduling) และ การกำหนดตัวประมวลผลสำหรับแต่ละกระบวนการ (Allocates processors to processes) จำนวนของโพรเซสอาจจะไม่เท่ากันกับจำนวนของตัวประมวลผลก็ได้ โมเดลในการควบคุมลิฟท์ของแจ็กสันจะประกอบไปด้วยกระบวนการ 50 กระบวนการ ผู้พัฒนาจะต้องตัดสินใจว่าจะให้แต่ละกระบวนการถูกทำโดยแต่ละซีพียูซึ่งจะต้องมีทั้งหมดถึง 50 ตัว หรือจะใช้ตัวประมวลผลร่วมกันระหว่างกระบวนการหลายกระบวนการ หลังจากผ่านขั้นตอนทั้งหกของเจเอสดีจะมาถึงการเขียนโค๊ดและการออกแบบฐานข้อมูล

3.11.2.2 การเปรียบเทียบกับโอเอ็มที

ผู้เขียนบางคนกล่าวถึงเจเอสดีว่าเป็นออพเจ็กต์โอเรียนเต็ดซึ่งเราไม่เห็นด้วย แม้ว่าเจเอสดีจะเริ่มต้นด้วยการพิจารณาโลกแห่งความเป็นจริงซึ่งในแง่นี้เป็นเหมือนแนวความคิดแบบออพเจ็กต์โอเรียนเต็ด แต่อย่างไรก็ตามแจ็กสันกำหนดเอ็นติตี้ (ออบเจ็กต์) และแสดงโครงสร้างของมันน้อยเกินไปแต่ละตัวอย่างจากสามตัวอย่างที่แจ็กสันนำเสนอ มีเอ็นติตี้เพียงสองหรือสามเอ็นติตี้เท่านั้น เราเชื่อว่าโมเดลแบบออพเจ็กต์โอเรียนเต็ดจะต้องมีการประกอบกันของโครงสร้างข้อมูลและความสัมพันธ์ระหว่างกันอย่างมาก (Rich mixture of data structure and relationships)

เราพบว่าวิธีการของแจ็กสันมีความซับซ้อนมากเกินไปและเป็นการยากที่จะสามารถเข้าใจทั้งหมด พวกเราคิดว่าเจเอสดีมีลักษณะที่คลุมเครือมากกว่าวิธีการแบบคาต้าโฟลว์และออพเจ็กต์โอเรียนเต็ด สาเหตุหนึ่งที่ทำให้เจเอสดีมีความซับซ้อนก็เนื่องจากว่าเจเอสดีต้องพึ่งการใช้ซูโดโค๊ดมากนั่นเอง ทั้งที่โมเดลที่เป็นแผนภาพจะสามารถทำให้เข้าใจได้ง่ายมากกว่า นอกจากนี้เจเอสดียังซับซ้อนเพราะว่ามันถูกออกแบบมาเพื่อจัดการกับปัญหาแบบเรียลไทม์ (Real Time) ซึ่งเป็นปัญหาที่ยากนั่นเอง สำหรับปัญหาประเภทดังกล่าว เจเอสดีอาจจะทำให้ได้ผลการออกแบบที่ดีกว่าและคุ้มค่ากับความซับซ้อนของโมเดลที่เพิ่มขึ้น อย่างไรก็ตามความซับซ้อนของเจเอสดีไม่มีความจำเป็นหรือมากเกินไปสำหรับปัญหาทั่วๆ ไป หรือปัญหาที่ง่ายกว่าปัญหาแบบเรียลไทม์

วิธีการของแจ็กสัน จะเน้นที่แอ็คชันมากกว่าที่แอททริบิวส์อย่างที่วิธีของโอเอ็มทีที่แอ็คชันของเจเอสดีบางอันดูเหมือนแอ็คโซซิเอชันของโอเอ็มที ตัวอย่างเช่น พนักงานจัดผลิตภัณฑ์ตามคำสั่งซื้อ (A cleak allocates product to an order) ตามวิธีโอเอ็มทีจะเรียก “จัด” (Allocate) ว่าเป็นแอ็คโซซิเอชัน แต่แจ็กสันเรียกมันว่าเป็นแอ็คชัน แจ็กสันคิดว่าแอททริบิวส์น่าสับสนและพยายามหลีกเลี่ยงมัน แอ็คชันมีบทบาทอย่างมากในการโมเดลลิ่งแบบเจเอสดีจนมันไม่เน้นถึงแอททริบิวส์ซึ่งก็เหมือนกันกับที่แอททริบิวส์มีบทบาทมากกว่าในออพเจ็กต์โมเดลของโอเอ็มที

เจเอสดีจะเป็นระเบียบวิธีการที่มีประโยชน์สำหรับแอพพลิเคชันประเภทต่อไปนี้

- คอนเคอร์เรนท์ซอฟแวร์ (Concurrent software) ซึ่งโพรเซสจะต้องชิงโครไนซ์ (Synchronize) ระหว่างกัน
- เรียลไทม์ซอฟแวร์ (Real time software) เจเอสดีโมเดลลิ่งจะบอกรายละเอียดได้อย่างดี และมุ่งความสนใจกับเวลาได้ดี
- ไมโครโค๊ด (Microcode) เจเอสดีจะมีรายละเอียดครอบคลุม ไม่มีการตั้งสมมุติฐานเกี่ยวกับการมีระบบปฏิบัติการและพิจารณาถึงการคอนเคอร์เรนท์กระบวนการและไทม์มิ่ง
- การโปรแกรมคอมพิวเตอร์แบบขนาน (Programming parallel computer) ภาพแบบการมีหลายกระบวนการของเจเอสดีจะช่วยในการออกแบบแอพพลิเคชันแบบนี้มาก

เจเอสดีจะไม่เหมาะสำหรับแอพพลิเคชันต่อไปนี้

- แอพพลิเคชันที่ต้องมีการวิเคราะห์ระดับสูง (High level analysis) เนื่องจากเจเอสดีไม่สนับสนุนการทำความเข้าใจในปัญหาอย่างกว้าง ครอบคลุมทั้งหมด เจเอสดีจะเป็นวิธีที่ไม่มีประสิทธิภาพดีพอในการทำแอปสแตรกชันและการทำซิมพลิฟิเคชัน (Abstraction and simplification) เจเอสดีจัดการรายละเอียดอย่างพิถีพิถันแต่ไม่ได้ช่วยให้ผู้พัฒนาเข้าใจถึงแก่นแท้ของปัญหา
- ฐานข้อมูล การออกแบบฐานข้อมูลจะเป็นเรื่องที่ยากถ้าใช้วิธีการของแจ็กสัน โมเดลแบบเจเอสดีเน้นที่แอ็คชันกว่าที่เอ็นติตี้และแอททริบิวท์มากจนเกินไป จึงเป็นเหตุให้มันเป็นวิธีการที่แย่ในการออกแบบฐานข้อมูล
- ซอฟต์แวร์ทั่วไปที่วิ่งอยู่ภายใต้ระบบปฏิบัติการตัวหนึ่งๆ เนื่องจากเจเอสดีจะมองระบบในแบบที่เป็นกระบวนการนับร้อยพัน ซึ่งทำให้เกิดความสับสนและบางส่วนก็ไม่จำเป็น

3.11.3 สรุป

วิธีการทางวิศวกรรมซอฟต์แวร์ที่เป็นที่นิยมหลายวิธีต่างก็มีพื้นฐานมาจากแนวคิดเกี่ยวกับการไหลของข้อมูลหรือดาต้าโฟลว์ และระเบียบวิธีการแบบเอสเอ/เอสดี ก็เป็นตัวแทนของวิธีการที่ใช้แนวความคิดของดาต้าโฟลว์ เอสเอ/เอสดีเริ่มต้นด้วยกระบวนการหรือฟังก์ชันเพียงอันเดียวที่แสดงถึงเป้าหมายของระบบทั้งหมดที่ต้องการ หลังจากนั้นเอสเอ/เอสดีก็จะเริ่มวนซ้ำเพื่อแบ่งกระบวนการที่ซับซ้อนนั้นออก จนกระทั่งได้ฟังก์ชันที่มีขนาดเล็กและสามารถนำมาสร้างได้อย่างมีประสิทธิภาพ

เอสเอ/เอสดี กับโอเอ็มทีมีอะไรหลายอย่างที่เหมือนกัน ซึ่งทั้งสองระเบียบวิธีการสนับสนุนมุมมองที่สมมาตรทั้งสามด้านของระบบ - - ออพเจ็กต์, ไดนามิกส์และฟังก์ชันนอลโมเดล ความแตกต่างของมันอยู่ที่ว่าเอสเอ/เอสดี เน้นที่ฟังก์ชันนอลโมเดล ในขณะที่โอเอ็มทีเน้นที่ออพเจ็กโมเดล พวกเราเชื่อว่าสำหรับปัญหาส่วนใหญ่แล้ววิธีการแบบออพเจ็กต์โอเรียนเต็ดจะเหมาะสมกว่าวิธีการ

แบบดาต้าโฟลว์ ผลการออกแบบด้วยวิธีออดเจ็กต์โอเรียนเต็ดจะสามารถเพิ่มขยายและสามารถตรวจสอบได้ง่ายกว่าและยังสามารถรวมฐานข้อมูลกับโค๊ดที่โปรแกรมได้ดีกว่าด้วย

เจเอสดีโมเดลเริ่มต้นด้วยการพิจารณาโลกแห่งความเป็นจริง จากนั้นผู้พัฒนาก็ทำการเขียนเอ็นติตี้และแอ็คชันที่สำคัยในโลกแห่งความเป็นจริงโดยใช้มุมมองจากตัวแอพพลิเคชัน ขั้นตองต่อๆไปที่เหลือของเจเอสดีจะเป็นการเพิ่มเติมรายละเอียดของซูโดโค๊ดเพื่อที่จะสามารถกำหนดพฤติกรรมแต่ละระเบียบวิธีการก็มีประเภทของงานที่มันสามารถแสดงเป็นโมเดลได้ดีไม่เหมือนกัน เจเอสดีจะเป็นวิธีการที่ดีมากสำหรับแอพพลิเคชันที่เป็นเรียลไทม์หรือเกี่ยวกับไมโครโค๊ด แต่เจเอสดีจะไม่หมาะสำหรับการวิเคราะห์ในระดับสูงและกี่ยวกับฐานข้อมูล

บทที่ 4

ระบบฐานข้อมูลเชิงสัมพันธ์

4.1. ความรู้พื้นฐานของระบบฐานข้อมูล[7],[8]

การประมวลผลในระบบแฟ้มข้อมูล ได้แบ่งหน่วยของข้อมูลไว้หลายระดับ ดังนี้

บิท(Bit) หมายถึงหน่วยของข้อมูลที่มีขนาดเล็กที่สุด

ไบท์(byte) หมายถึง หน่วยของข้อมูลที่เกิดจากการนำบิทมารวมกันเป็นตัวอักษร

ฟิลด์(Field) หมายถึง หน่วยของข้อมูลที่ประกอบด้วยหลายๆ ตัวอักษร เพื่อแทนความหมายของสิ่งหนึ่ง เช่น รหัสพนักงาน ชื่อ เป็นต้น

เรคคอร์ด(Record) หมายถึงหน่วยของข้อมูลที่เกิดจากการนำเอาฟิลด์หลายๆ ฟิลด์ มารวมกันเพื่อแสดงรายละเอียด ข้อมูลในเรื่องใดเรื่องหนึ่งเช่นเรคคอร์ดหนึ่งๆ ของพนักงานประกอบด้วยฟิลด์ต่างๆ เช่นรหัสพนักงาน ชื่อ แผนก เงินเดือน เป็นต้น

แฟ้มข้อมูล(File)หมายถึงหน่วยของข้อมูลที่เกิดขึ้นจากการนำเรคคอร์ดหลายๆ เรคคอร์ดมารวมกัน

สำหรับในระบบฐานข้อมูล คำศัพท์พื้นฐานที่เกี่ยวข้องมีดังต่อไปนี้

เอ็นทิตี้(Entity)หมายถึงชื่อของสิ่งใดสิ่งหนึ่งอาจเกี่ยวกับคนสถานที่ สิ่งของ การกระทำซึ่งต้องการจัดเก็บข้อมูลไว้ เอนทิตี้พนักงาน สินค้า ลูกค้า การสั่งซื้อ เป็นต้นบางเอนทิตี้ในฐานข้อมูลจะไม่มีความหมายหากไม่มีเอนทิตี้อื่นในฐานข้อมูลเข้ามาเกี่ยวข้อง เอนทิตี้ประเภทนี้เรียกว่าเอนทิตี้ชนิดอ่อนแอ(Weak Entity) ตัวอย่างเช่นเอนทิตี้ประวัติครอบครัวของพนักงานเป็นเอนทิตี้ชนิดอ่อนแอเพราะถ้าปราศจากเอนทิตี้พนักงานแล้ว เอนทิตี้นี้จะไม่มีความหมายเพราะไม่รู้ว่าเป็นประวัติของพนักงานคนใด

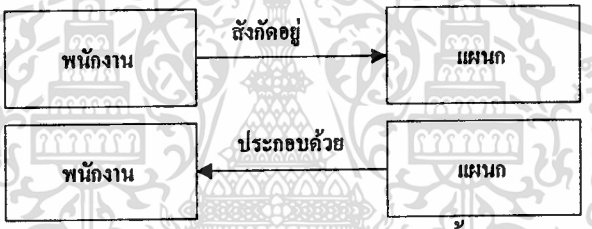
แอททริบิวต์(Attribute)หมายถึงรายละเอียดของข้อมูลในเอนทิตี้หนึ่งๆ เช่นเอนทิตี้พนักงาน ประกอบด้วยแอทริบิวต์รหัสพนักงาน ชื่อ เงินเดือน หรือเอนทิตี้ลูกค้า ประกอบด้วยแอททริบิวต์รหัสลูกค้า ชื่อ ที่อยู่ หมายเลขโทรศัพท์ หรือแอนทิตี้แผนก ประกอบด้วยแอททริบิวต์รหัสแผนกชื่อ เป็นต้น แอททริบิวต์บางแอททริบิวต์ประกอบด้วยข้อมูลหลายส่วนมารวมกันซึ่งอาจแยกเป็นชื่อแอททริบิวต์ย่อยได้อีก แอททริบิวต์ที่มีคุณสมบัติอย่างนี้เรียกว่าแอททริบิวต์ผสม(Composite Attribute)ตัวอย่างเช่น แอททริบิวต์ที่อยู่ เป็นแอททริบิวต์ผสมที่ประกอบด้วยข้อมูล บ้านเลขที่ ถนน ซอย อำเภอ จังหวัด และรหัสไปรษณีย์ ซึ่งสามารถแยกออกเป็นแอททริบิวต์ย่อยได้อีก(ถ้าต้องการ) เช่นแยกเป็นแอททริบิวต์ที่อยู่ ซึ่งประกอบด้วยบ้านเลขที่ ถนน ซอย อำเภอ และแอทริบิวต์ที่อยู่ประกอบด้วยจังหวัด และรหัสไปรษณีย์ นอกจากนี้ แอททริบิวต์บางแอททริบิวต์อาจจะไม่มีค่าของตัวเอง แต่สามารถหาค่าได้จากแอททริบิวต์อื่นๆ เช่น แอททริบิวต์อายุ สามารถคำนวณได้จากแอทท

ริบิวต์วันเกิด เป็นต้น แอททริบิวต์ที่มีคุณสมบัติดย่างนี้เรียกว่า แอททริบิวต์ที่ถูกแปลค่ามา(Derrived Attribute)

ความสัมพันธ์(Relationship)หมายถึง คำกริยาที่แสดงความสัมพันธ์ระหว่างสองเอนทิตี้ เช่น เอนทิตี้พนักงาน และเอนทิตี้แผนก มีความสัมพันธ์ในด้าน "ทำงานสังกัดอยู่" นั่นคือพนักงานแต่ละคนสังกัดอยู่ในแผนกใด แผนกหนึ่ง เป็นต้น

ความสัมพันธ์ระหว่างเอนทิตี้ ในการระบุชื่อความสัมพันธ์ระหว่างเอนทิตี้ จะพิจารณาโดยกำหนดทิศทางของความสัมพันธ์จากเอนทิตี้หนึ่งไปยังอีกเอนทิตี้หนึ่งว่ามีความสัมพันธ์กันที่เรียกว่าอะไร เช่น ความสัมพันธ์ จากเอนทิตี้พนักงานไปยังเอนทิตี้แผนกเป็นความสัมพันธ์ที่เรียกว่า "สังกัดอยู่" นั่นคือ พนักงานแต่ละคนจะสังกัดอยู่ในแผนก ในทางตรงกันข้าม อาจจะระบุทิศทางของความสัมพันธ์ว่า ความสัมพันธ์จากเอนทิตี้แผนกไปยังเอนทิตี้พนักงาน เป็นความสัมพันธ์ที่เรียกว่า "ประกอบด้วย" นั่นคือแต่ละแผนกประกอบด้วยพนักงาน เป็นต้นดังภาพ 4.1

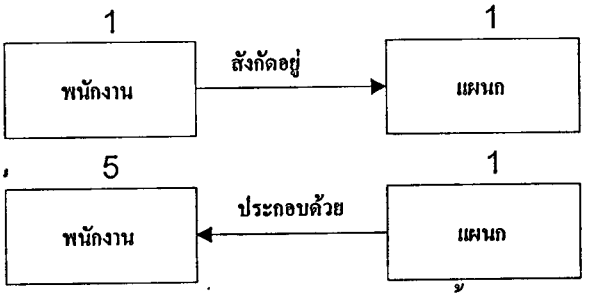
ภาพที่ 4.1



แสดงความสัมพันธ์ระหว่างเอนทิตี้

นอกจากคำนึงถึงความสัมพันธ์ ยังต้องพิจารณาถึงจำนวนข้อมูลที่เกิดขึ้นระหว่างความสัมพันธ์ของสองเอนทิตี้ว่ามีเท่าไร(Cardinality Ratio)ตัวอย่างเช่น ความสัมพันธ์ของข้อมูลจากเอนทิตี้พนักงานไปยังเอนทิตี้แผนก เป็นอัตราส่วน 1:1 นั่นคือพนักงานแต่ละคนสังกัดอยู่เพียงแผนกเดียว หรือความสัมพันธ์ของข้อมูลจากเอนทิตี้แผนกไปนังเอนทิตี้พนักงานเป็นอัตราส่วน 1:5 นั่นคือแต่ละแผนกประกอบด้วยพนักงาน 5 คน เป็นต้น ดังภาพ 4.2

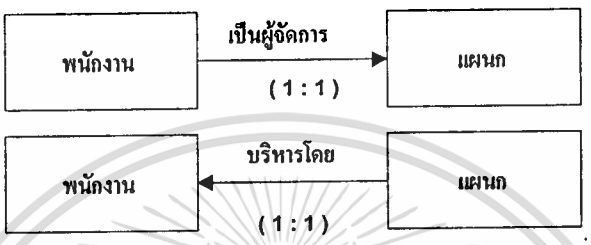
ภาพที่ 4.2



แสดงความสัมพันธ์ระหว่างเอนทิตี้

1. ความสัมพันธ์แบบหนึ่งต่อหนึ่ง(One to One Relationship)เป็นการแสดงความสัมพันธ์ของข้อมูลของเอนทิตี้หนึ่งว่ามีความสัมพันธ์กับข้อมูลอย่างมากหนึ่งข้อมูลกับอีกเอนทิตี้หนึ่งในลักษณะที่เป็นหนึ่งต่อหนึ่ง ตัวอย่างเช่น จากภาพ 4.3 พนักงาน อย่างมากหนึ่งคนเท่านั้นที่จะเป็นผู้จัดการแผนก ในขณะเดียวกัน แต่ละแผนกจะมีผู้จัดการเพียงหนึ่งคนเท่านั้น ความสัมพันธ์ระหว่างพนักงานกับแผนกในลักษณะที่เป็นแบบหนึ่งต่อหนึ่ง

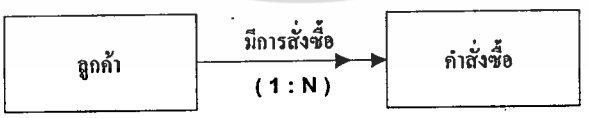
ภาพที่ 4.3



แสดงความสัมพันธ์แบบ หนึ่งต่อหนึ่ง

2. ความสัมพันธ์แบบหนึ่งต่อกลุ่ม(One to Many Relationship)เป็นการแสดงความสัมพันธ์ของข้อมูลของเอนทิตี้หนึ่งว่ามีความสัมพันธ์กับข้อมูลหลายข้อมูลกับอีกเอนทิตี้หนึ่ง ตัวอย่างเช่น จากภาพ 4.4 ความสัมพันธ์ของลูกค้าไปยังคำสั่งซื้อ เป็นความสัมพันธ์แบบหนึ่งต่อกลุ่ม **(One to Many)**นั่นคือ ลูกค้าแต่ละคนสามารถสั่งซื้อได้หลายคำสั่งซื้อ ในทางตรงกันข้าว ความสัมพันธ์ของคำสั่งซื้อไปสู่ลูกค้า จะเป็นลักษณะหนึ่งต่อหนึ่ง เพราะว่าหนึ่งคำสั่งซื้อเกิดจากคำสั่งซื้อของลูกค้าเพียงคนเดียว ดังนั้นความสัมพันธ์ระหว่างเอนทิตี้ลูกค้าคำและคำสั่งซื้อจึงเป็นหนึ่งต่อกลุ่ม (1:N)

ภาพที่ 4.4



แสดงความสัมพันธ์แบบ หนึ่งต่อกลุ่ม

3. ความสัมพันธ์แบบกลุ่มต่อกลุ่ม(Many to Many Relationship) เป็นการแสดงความสัมพันธ์ของข้อมูลของสองเอนทิตี้ในลักษณะแบบกลุ่มต่อกลุ่ม ตัวอย่างเช่น จากภาพ 4.5 ในเอนทิตี้การสั่งซื้อสินค้าแต่ละครั้ง สามารถสั่งซื้อสินค้าได้มากกว่าหนึ่งชนิด ความสัมพันธ์ของคำสั่งซื้อไปยังเอนทิตี้สินค้าเป็นแบบหนึ่งต่อกลุ่ม(1 : N) และสินค้าแต่ละชนิดสามารถถูกสั่งซื้อจากคำสั่งซื้อ

ของลูกค้าหลายคน ซึ่งเป็นความสัมพันธ์ของเอนทิตี้ที่สินค้าไปยังเอนทิตี้คำสั่งซื้อแบบหนึ่งต่อกลุ่ม (1 : M) ดังนั้น ความสัมพันธ์ของเอนทิตี้ทั้งสองจึงเป็นแบบกลุ่มต่อกลุ่ม(M : N)

ความสัมพันธ์ระหว่างข้อมูลของสองเอนทิตี้เป็นแบบกลุ่มต่อกลุ่ม เป็นเรื่องที่ค่อนข้างจะยุ่งยากในการออกแบบฐานข้อมูล เช่น อาจจะมีปัญหาในด้านของการปรับปรุงแก้ไขข้อมูล โดยทั่วไปจะสร้างเอนทิตี้ใหม่ขึ้นมา(Associative Entity)เพื่อเป็นเอ็นทิตี้ที่เชื่อมความสัมพันธ์กับสองเอนทิตี้เดิม โดยมีวัตถุประสงค์เพื่อปรับความสัมพันธ์ให้อยู่ในภาพของหนึ่งต่อกลุ่ม(1 : M) ตัวอย่างเช่น

สินค้าเป็นส่วนหนึ่งของหลายรายการที่สั่งซื้อ

คำสั่งซื้อประกอบด้วย หลายรายการที่สั่งซื้อ

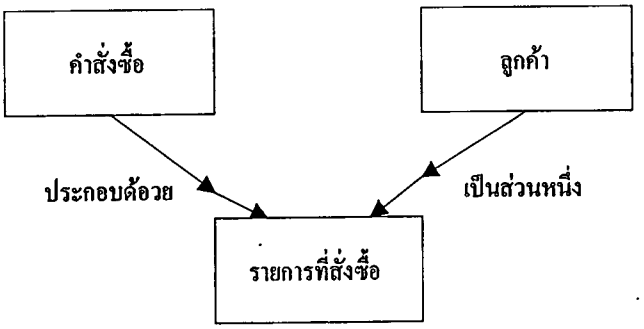
การกำหนดเอนทิตี้ใหม่ชื่อ “รายการที่สั่งซื้อ” เป็นเอนทิตี้ใหม่ที่มีความสัมพันธ์กับเอนทิตี้สินค้าและคำสั่งซื้อแบบหนึ่งต่อกลุ่ม ดังภาพ 4.6

ภาพที่ 4.5

แสดงความสัมพันธ์แบบ กลุ่มต่อกลุ่ม

การกำหนดความสัมพันธ์ของเอนทิตี้จะเป็นภาพลักษณะใด ขึ้นอยู่กับการออกแบบฐานข้อมูลและสมมุติฐานที่นำมาใช้ ซึ่งเป็นข้อมูลสมมุติฐานตามที่เกิดขึ้นจริงของระบบฐานข้อมูลนั้นๆ เช่น พนักงานหนึ่งคนจะสังกัดเพียงแผนกเดียว เป็นต้น นอกจากนี้ ในการระบุความสัมพันธ์ของข้อมูลระหว่างเอนทิตี้มี วัตถุประสงค์หนึ่งเพื่อใช้ในการออกแบบฐานข้อมูลโดยการกำหนดชื่อแอททริบิวต์ที่ใช้ในการเชื่อโยงข้อมูลต่อกันความสัมพันธ์ระหว่างเอนทิตี้ที่กล่าวมาข้างต้นเป็นความสัมพันธ์ระหว่างสองเอนทิตี้(Binary Relationship) เช่น เอนทิตี้พนักงานกับเอนทิตี้แผนก นอกจากนี้อาจจะมีความสัมพันธ์ของเอนทิตี้ในลักษณะอื่นๆ เช่น ความสัมพันธ์ระหว่างเอนทิตี้มากกว่าสองเอนทิตี้(Ternary Relationship) ความสัมพันธ์ระหว่าง Supertype และ Subtype หรือความสัมพันธ์กับเอนทิตี้ของตัวเอง(Recursive หรือ Self Relationship หรือ Unary Relationship

ภาพที่ 4.6



แสดงความสัมพันธ์ระหว่างเอนทิตี้มากกว่าสองเอนทิตี้

ความสัมพันธ์ระหว่าง Supertype และ Subtype ตามที่ได้กล่าวมาแล้วว่า **Subtype** เป็นเซ็ทย่อยของเอนทิตี้หนึ่งๆ หรือที่เรียกว่า **Supertype** โดยที่ **Subtype** ประกอบด้วยแอททริบิวต์ทุกแอททริบิวต์ที่มีอยู่ใน **Supertype** นอกจากนี้ ยังประกอบด้วยแอททริบิวต์เฉพาะเพิ่มเติม ตัวอย่างเช่น เอนทิตี้พนักงานเป็น **Supertype** ที่ประกอบด้วย พนักงาน 2 ประเภท คือ **Subtype** พนักงานที่มีเงินเดือนประจำ(EMP_SALARY) และพนักงานที่คิดค่าแรงต่อชั่วโมง(EMP_WAGE)ความสัมพันธ์ระหว่าง **Supertype** และ **Subtype** เป็นความสัมพันธ์ที่บอกถึงสถานภาพของพนักงาน โดยเป็นความสัมพันธ์แบบหนึ่งต่อหนึ่ง ดังภาพ 4.7

ภาพที่ 4.7

แสดงความสัมพันธ์ระหว่างเอนทิตี้มากกว่าสองเอนทิตี้

ในบางครั้ง ข้อมูลใน **Supertype** จะถูกจัดกลุ่ม(Category) ให้มีความสัมพันธ์กับ **Subtype** ใด **Subtype** หนึ่งเท่านั้น ลักษณะเช่นนี้เรียกว่า **Subtype** ที่เป็นอิสระจากกัน(Mutually Exclusive Subtype)ตัวอย่างเช่น พนักงานแต่ละคนจะมีสถานภาพเป็นพนักงานที่มีเงินเดือนประจำ หรือพนักงานที่คิดค่าแรงชั่วโมงเพียงสถานภาพเดียว จะมีสองสถานภาพในเวลาเดียวกันไม่ได้

4.2 แนวความคิดเกี่ยวกับการออกแบบฐานข้อมูล

เมื่อพูดถึงการออกแบบฐานข้อมูลจะหมายรวมถึง การออกแบบฐานข้อมูลในระดับแนวความคิด(Conceptral Level) และการออกแบบฐานข้อมูลในระดับภายในหรือเชิงกายภาพ(Internal หรือ Physical Level) การออกแบบฐานข้อมูลในระดับแนวความคิดเป็นการออกแบบเค้าโครงร่างของรีเลชันว่ารีเลชันๆประกอบด้วยแอททริบิวต์ต่างๆ รวมถึงการเรียกใช้ข้อมูลด้วย ส่วนการออกแบบฐานข้อมูลในระดับภายใน เป็นการออกแบบที่เน้นในเรื่องของการจัดเก็บข้อมูลว่าควรจะมีการจัดเก็บอย่างไร

การออกแบบเค้าร่างของข้อมูลเป็นการกำหนด รีเลชันต่างๆ รวมถึง แอททริบิวต์ในแต่ละรีเลชันเพื่อให้ได้ข้อมูลตามที่ผู้ใช้ต้องการ แต่การกำหนดหรือออกแบบอย่างไรจึงจะได้เค้าร่างของข้อมูลที่ดี กล่าวอีกนัยหนึ่งคือเค้าร่างของรีเลชันประกอบด้วย แอททริบิวต์ที่เหมาะสมหรือมากเกินไปหรือไม่ข้อมูลที่เก็บในรีเลชันนั้นๆ มีความซ้ำซ้อนเกิดขึ้นหรือไม่เป็นต้น ดังนั้นในการออกแบบเค้าร่างของรีเลชันที่ดีควรจะเป็นการออกแบบที่สามารถลดปัญหาต่างๆ ที่จะเกิดขึ้นกับฐานข้อมูลให้มากที่สุด เช่นปัญหาข้อมูลซ้ำซ้อน ปัญหาข้อมูลไม่ถูกต้องปัญหาการรักษาความปลอดภัยของข้อมูล

ก่อนที่จะกล่าวถึงขั้นตอนการออกแบบฐานข้อมูล จะกล่าวถึงแนวความคิดที่สำคัญที่ใช้เป็นแนวทางในการออกแบบฐานข้อมูลในระดับแนวความคิด แนวความคิดดังกล่าวประกอบด้วยความสัมพันธ์ระหว่างค่าของแอททริบิวต์ในแต่ละรีเลชัน(Functional Dependency)และการทำรีเลชันให้อยู่ในภาพแบบบรรทัดฐานต่างๆ(Normalization)

4.3 ความสัมพันธ์ระหว่างค่าของแอททริบิวต์ในแต่ละรีเลชัน(Dependency)

เนื่องจากค่าของแอททริบิวต์ในแต่ละรีเลชัน อาจจะมีความสัมพันธ์กันในลักษณะที่เมื่อทราบค่าของแอททริบิวต์หนึ่งๆ จะสามารถทราบถึงค่าของแอททริบิวต์อื่นๆ ของตารางหนึ่งๆ ในรีเลชันได้ ลักษณะของความสัมพันธ์ระหว่างค่าของแอททริบิวต์ในแต่ละรีเลชัน

4.3.1 ความสัมพันธ์ระหว่างค่าของแอททริบิวต์แบบฟังก์ชัน(Functional Dependency)

ความสัมพันธ์ระหว่างค่าของแอททริบิวต์แบบฟังก์ชัน คือ การที่แอททริบิวต์หนึ่งหรืออาจมากกว่าหนึ่งแอททริบิวต์ประกอบกันสามารถระบุค่าของแอททริบิวต์อื่นๆ ในตารางหนึ่งได้ชัดเจน เมื่อพูดถึงความสัมพันธ์ในการระบุค่าแอททริบิวต์จะเกี่ยวข้องกับคีย์หลัก(คีย์หลัก) ทั้งนี้เพราะว่าคุณสมบัติของคีย์หลักจะเป็นแอททริบิวต์ที่มีค่าของการเป็นเอกลักษณ์(Unique) ที่สามารถระบุค่าของแอททริบิวต์อื่นๆในตารางหนึ่งๆ ได้

เพื่อให้เข้าใจถึงความสัมพันธ์ระหว่างค่าของแอททริบิวต์แบบฟังก์ชันได้ดียิ่งขึ้น จะใช้รีเลชัน Employee เป็นตัวอย่างในการอธิบาย รีเลชันนี้ประกอบด้วยรหัสสมาชิก(CUS_CODE) ชื่อสมาชิก (NAME) จังหวัดที่สมาชิกอยู่ (PROVICE) ดังนี้

ภาพที่ 4.8

CUS_CODE	NAME	PROVICE
100001	SOMMAI THONGSAI	LOBBURI
200011	BENJA TUMMADEE	NONTABURI
300021	PANYA SAIPUNTA	NONTABURI
400031	NANTA TUMDANG	SARABURI
500040	NOPDOL MANEERUT	BANGKOK
600051	BANDIT TOYAYAG	BANGKOK

แสดงข้อมูลของตาราง Employee.

จากภาพ 4.8 จะเห็นว่า เมื่อทราบค่าแอททริบิวต์ของรหัสสมาชิก จะสามารถทราบค่าของแอททริบิวต์ตัวอื่นๆ ได้อย่างชัดเจน นั่นหมายความว่าแอททริบิวต์ของรหัสสมาชิก จะมีความสัมพันธ์ในการระบุค่าของแอททริบิวต์แบบฟังก์ชันกับแอททริบิวต์อื่นๆ

ในทางตรงกันข้ามหากทราบค่าของแอททริบิวต์จังหวัดที่สมาชิกอาศัยอยู่ จะสามารถทราบค่าของรหัสของสมาชิก หรือชื่อของสมาชิกได้ไม่ชัดเจน ตัวอย่างเช่นหากทราบค่าของจังหวัด BANGKOK ค่านี้ไม่สามารถระบุได้ชัดเจนว่าเป็นของรหัสสมาชิก 500040 หรือ 600051 ทั้งนี้เพราะแอททริบิวต์ชื่อจังหวัดสมาชิกมีชื่อซ้ำกัน หรือสามารถกล่าวอีกนัยหนึ่งได้ว่าค่าของแอททริบิวต์ต่างๆในรีเลชัน Employee ไม่สามารถระบุค่าของแอททริบิวต์ จังหวัดที่สมาชิกอยู่ได้ชัดเจนนั่นเอง ดังนั้นถ้าแอททริบิวต์หนึ่งมีความสัมพันธ์ระหว่างค่าของแอททริบิวต์อื่นๆ แสดงว่าแอททริบิวต์นั้นเป็นตัวระบุค่า(Determinant) ของแอททริบิวต์อื่นๆในฟังก์ชันก์

4.3.2. ความสัมพันธ์ระหว่างค่าของแอททริบิวต์แบบบางส่วน(Partial Dependency)

ความสัมพันธ์แบบนี้จะเกิดขึ้นได้เมื่อรีเลชันหนึ่งๆมีคีย์หลักเป็นคีย์ผสม (Composite Key) นั่นคือคีย์หลักของรีเลชันนั้นๆประกอบด้วย แอททริบิวต์หลายแอททริบิวต์ความสัมพันธ์ระหว่างค่าของแอททริบิวต์แบบบางส่วนเกิดขึ้นเมื่อ แอททริบิวต์บางส่วนของคีย์หลักสามารถระบุค่าของแอททริบิวต์อื่นๆ ที่ไม่ใช่คีย์หลักของรีเลชันได้(Non-Key Attribute) ตัวอย่างเช่น

ภาพที่ 4.9

CUS_CODE	ST_CODE	SE_NUMBER
100001	10100110	250
100001	10100111	100
100001	10200101	270
100001	10300101	10
200001	10100110	320
200001	10100111	30
200001	10200101	250
300021	10100110	100
300021	10100111	270
400031	10300101	10
500041	20100110	320
600051	20500601	30

แสดงความสัมพันธ์ของ คีย์ต่างๆในตาราง Transac.

จากภาพ 4.9 รีเลชันของตาราง Transac จะมีแอททริบิวต์ CUS_CODE และ ST_CODE เป็นคีย์หลักและประกอบด้วย แอททริบิวต์อื่นๆ ดังต่อไปนี้

CUS_CODE หมายถึง รหัสสมาชิก

ST_CODE หมายถึง รหัสสินค้า

SE_NUMBER หมายถึง จำนวนสินค้า

จะเห็นว่ารีเลชันของตาราง Transac นั้นมีแอททริบิวต์ CUS_CODE และ ST_CODE เป็นคีย์หลักที่สามารถระบุค่าของจำนวนสินค้าที่ถูกจัดซื้อ(SE_NUMBER)ขณะเดียวกันมีความสัมพันธ์ระหว่างค่าของแอททริบิวต์แบบบางส่วนเกิดขึ้นโดยแอททริบิวต์ ST_CODE ซึ่งเป็นส่วนประกอบตัวหนึ่งของคีย์หลัก(CUS_CODE)หรือ(ST_CODE)สามารถระบุค่าของจำนวนสินค้าได้

ภาพที่ 4.10

CUS_CODE	ST_CODE	ST_NAME	SE_NUMBER
100001	10100110	MS_DIS VERSION 6.22 THAI	250
100001	10100111	ANTI VIRUS CARD MODEK 2000I	100
100001	10200101	FINGERTRIP	270
100001	10300101	VIDEO NATIONAL MODEL G-20	10
200001	10100110	MS-DOS VERSION 6.22 THAI	320
200001	10100111	ANTI VIRUS CARD MODEK 200I	30
200001	10200101	FIGERTRIP	250
300021	10100110	MA-DOS VERSION 6.22 THAI	100
300021	10100111	ANTI VIRUS CARD MODEK 2000I	270
400031	10300101	FIGERTRIP	10
500041	20100110	CISKETE 3 1/2 ICH	320
600051	20500601	MAG MONITOR 14" MODEL M1450	30

แสดงความสัมพันธ์ของ คีย์ต่างๆในตาราง Transac.

จากภาพ 4.10 รีเลชันของตาราง Transac มีแอททริบิวต์ CUS_CODE และ ST_CODE เป็นคีย์หลักและประกอบด้วย แอททริบิวต์อื่นๆ ดังต่อไปนี้

CUS_CODE หมายถึง รหัสสมาชิก

ST_CODE หมายถึง รหัสสินค้า

ST_NAME หมายถึง ชื่อสินค้า

SE_NUMBER หมายถึง จำนวนสินค้า

รีเลชันของตาราง Transac นั้นมีแอททริบิวต์ CUS_CODE และ ST_CODE เป็นคีย์หลักที่สามารถระบุค่าของชื่อสินค้า(ST_NAME) และจำนวนสินค้าที่ถูกจัดส่ง(SE_NUMBER) ขณะเดียวกัน มีความสัมพันธ์ระหว่างค่าของแอททริบิวต์แบบบางส่วนเกิดขึ้น โดยแอททริบิวต์ ST_CODE ซึ่งเป็นส่วนประกอบตัวหนึ่งของคีย์หลัก(CUS_CODE) หรือ(ST_CODE) สามารถระบุค่าของชื่อสินค้าได้(ST_NAME) ความสัมพันธ์ลักษณะนี้ก่อให้เกิดปัญหาในเรื่องของความซ้ำซ้อน และการปรับปรุงข้อมูล

ดังนั้นความสัมพันธ์ลักษณะนี้จะไม่เกิดขึ้นกับรีเลชันที่มีแอททริบิวต์เดียวเป็นคีย์หลัก แต่เป็นกับคีย์ผสม ซึ่งแอททริบิวต์ตัวใดตัวหนึ่งที่ประกอบเป็นคีย์หลักสามารถระบุค่าของแอททริบิวต์อื่นๆที่ไม่ใช่คีย์หลักได้ เหมือนดังตัวอย่างนี้

4.3.3. ความสัมพันธ์ระหว่างค่าของแอททริบิวต์แบบทรานซีทีฟ (Transitive Dependency)

แอททริบิวต์ที่มีคุณสมบัติเป็นคีย์หลัก จะสามารถระบุค่าของทุกแอททริบิวต์ในแต่ละตารางได้ อย่างไรก็ตามในบางรีเลชัน(ที่ออกแบบไม่เหมาะสม) อาจจะมีกรณีแอททริบิวต์อื่น(NonKey Attribute) ที่สามารถระบุค่าของแอททริบิวต์อื่นๆในตารางได้ ลักษณะของความสัมพันธ์ในการระบุค่าของแอททริบิวต์แบบนี้ เรียกว่าความสัมพันธ์ระหว่างค่าของแอททริบิวต์แบบทรานซิทีฟ(Transitive Dependency)

ภาพที่ 4.11

CUS_CODE	NAME	PROVICE	CPOST
100001	SOMMAI THONGSAI	LOBBURI	15000
200011	BENJA TUMMADEE	NONTABURI	11120
300021	PANYA SAIPUNTA	NONTABURI	10100
400031	NANTA TUMDUNG	SARABURI	15135
500041	NOPADOL MANEERUT	BANGKOK	10230
600051	BANDIT TOYAYAG	BANGKOK	10230

แสดงรีเลชันของตาราง Employee

ตัวอย่างเช่น จากภาพ 4.11 เป็น รีเลชันของตาราง Employee ประกอบด้วย แอททริบิวต์ ดังต่อไปนี้

CUS_CODE	หมายถึง	รหัสสมาชิก
NAME	หมายถึง	ชื่อสมาชิก
PROVICE	หมายถึง	ชื่อจังหวัดที่สมาชิกอยู่
CPOST	หมายถึง	ระหัสไปรษณีย์

รีเลชันนี้มีรหัสสมาชิกเป็นคีย์หลัก ที่สามารถระบุค่าของแอททริบิวต์อื่นๆ ในแต่ละตารางได้ นอกจากนี้ยังพบว่าแอททริบิวต์ PROVICE ซึ่งไม่ได้เป็นคีย์หลักสามารถกำหนดค่าของแอททริบิวต์ CPOST ว่าถูกจัดอันดับไว้ในอันดับใดบ้าง

4.3.4.ความสัมพันธ์ระหว่างค่าของแอททริบิวต์แบบหลายค่า(Multivalued dependency)

จากความสัมพันธ์ระหว่างค่าของแอททริบิวต์แบบฟังก์ชันที่กล่าวมาข้างต้น เป็นลักษณะที่แอททริบิวต์หนึ่งมีคุณสมบัติในการระบุค่าของแอททริบิวต์อื่นๆ ในแต่ละตารางได้เพียงหนึ่งค่า ซึ่งคุณสมบัติของแอททริบิวต์ที่สามารถระบุค่าของแอททริบิวต์อื่นๆได้ คือแอททริบิวต์ที่เป็นคีย์หลักหรือคีย์คู่แข่ง อย่างไรก็ตามในบางรีเลชันอาจจะมีกรณีของความสัมพันธ์ระหว่างค่าของแอททริบิวต์แบบหลายค่าเกิดขึ้นได้ โดยความสัมพันธ์นี้จะเกิดขึ้นกับรีเลชันที่ประกอบด้วย แอททริบิวต์อย่างน้อย 3 แอททริบิวต์ และเป็นรีเลชันที่แอททริบิวต์หนึ่งสามารถระบุค่าของแอททริบิวต์อื่นๆในรีเลชันได้มากกว่าหนึ่งค่า กรณีเช่นนี้เรียกว่ารีเลชันนั้นๆ มีความสัมพันธ์ในการระบุค่าของแอททริบิวต์แบบหลายค่า(Multivalued Dependency)

4.4 การทำรีเลชันให้อยู่ในภาพแบบบรรทัดฐาน(Normalization)

แนวความคิดในการทำรีเลชันให้อยู่ในรูปแบบบรรทัดฐาน(Normalization Process)ถูกคิดค้นโดย อี.เอฟ.คอดด์(E.F.CODD) เป็นกระบวนการที่นำเค้าโครงร่างของรีเลชันมาทำให้อยู่ในภาพแบบที่เป็นบรรทัดฐาน(Normal Form) เพื่อให้แน่ใจว่าการออกแบบเค้าโครงร่างของรีเลชันเป็นการออกแบบที่เหมาะสม

วัตถุประสงค์ของการทำให้เป็นบรรทัดฐานมีดังนี้คือ

1. เพื่อลดเนื้อที่ในการจัดเก็บข้อมูล การทำให้เป็นบรรทัดฐานเป็นการลดความซ้ำซ้อนของข้อมูลในรีเลชัน ซึ่งเป็นการลดเนื้อที่ในการจัดเก็บข้อมูลได้
2. เพื่อลดปัญหาที่ข้อมูลไม่ถูกต้อง(Inconsistency)เนื่องจากข้อมูลในรีเลชันหนึ่งจะมีข้อมูลไม่ซ้ำกัน เมื่อมีการปรับปรุงข้อมูลก็จะปรับปรุงตารางนั้นๆครั้งเดียว ไม่ต้องปรับปรุงหลายแห่งทำให้โอกาสที่จะเกิดความผิดพลาดในการปรับปรุงข้อมูลที่จะไม่ครบถ้วนทุกตารางก็จะไม่เกิดขึ้น

3. เป็นการลดปัญหาที่เกิดจากการเพิ่ม ปรับปรุงและลบข้อมูล(Insert, Update and Delete Anomalies) ช่วยแก้ปัญหาที่อาจจะเกิดขึ้นจากการปรับปรุงข้อมูลไม่ครบ หรือข้อมูลหายไปจากฐานข้อมูลหรือการเพิ่มข้อมูล

4.5 การจัดระบบข้อมูลด้วยวิธีนอร์มอลไลเซชัน(Normalization)

วิธีการรูปแบบบรรทัดฐานเป็นเทคนิคที่มีประโยชน์มากในการสร้างระบบฐานข้อมูล แต่เทคนิคนี้ก็ยังมีข้อด้อยอยู่ โดยเฉพาะในด้านการจัดการระบบควบคุมอินพุทและเอ้าพุท(Input / Output) หากมีการวางแผนงานไม่ดีพอในกรณีที่ต้องเรียกใช้ข้อมูลหลายชุด ที่จัดเก็บแยกกันทางกายภาพ(แยกกันอยู่คนละไฟล์) การจัดการด้านอินพุทและเอ้าพุท (คือการอ่านเขียนข้อมูล) ก็จะเกิดขึ้นบ่อยครั้ง ซึ่งจะมีผลกระทบต่อการทำงานของระบบ นั่นคือระบบจะเสียเวลาทำงานนานขึ้น ผู้ออกแบบจะต้องหาความสมดุลย์ที่เหมาะสมระหว่างการทำนอร์มอลไลเซชันของข้อมูลกับการทำงานของระบบ โดยส่วนรวมด้วย

4.5.1. ระบบการจัดการข้อมูลภายในฐานข้อมูล

วิธีการต่างๆตั้งแต่กระบวนการสร้างไฟล์ฐานข้อมูลเป็นคลังเก็บข้อมูล(Data storage files) จนถึงการจัดการเชื่อมโยงความสัมพันธ์ระหว่างไฟล์ในส่วนที่ผ่านๆมาทั้งหมด อาจสรุปเป็นขั้นตอนการออกแบบจัดโครงสร้างข้อมูลให้เป็นระบบได้ดังนี้

1.1. ระบุตาราง (Tables) ต่างๆ ในระบบฐานข้อมูล

1.2. จัดวางภาพแบบโครงสร้างของฟิลด์ใน ตาราง

1.3. กำจัดกลุ่มของฟิลด์ที่ซ้ำๆ กัน

1.4. ระบุ ฟิลด์หลัก(คีย์หลัก) ใน ตาราง

1.5. ระบุ ฟิลด์รอง(Secondary Key) ใน ตาราง

1.6. ระบุ ฟิลด์ ที่ใช้เชื่อมโยง กับตาราง อื่นเข้าด้วยกัน (Foreign Key)

1.7. ระบุความสัมพันธ์ระหว่าง ตาราง

ความจริงแล้วเทคนิคการนอร์มอลไลเซชันนี้จะนำมาใช้ในการออกแบบระบบโครงสร้างข้อมูล ของระบบจัดการฐานข้อมูลที่ต้องการ ตั้งแต่ขั้นตอนเริ่มวางแผนออกแบบครั้งแรกก็ได้ ไม่จำเป็นต้องนำมาใช้เพื่อแก้ไขปัญหาที่เกิดขึ้นในภายหลังอย่างเดียว แต่อย่างไรก็ตามในการออกแบบครั้งแรกสุด ทีมผู้ออกแบบหรือผู้ใช้ระบบอาจจะยังมอง ไม่เห็นปัญหาต่างๆที่จะเกิดขึ้นภายในระบบ ดังนั้นเทคนิคนี้จะช่วยให้มองเห็นปัญหาต่างๆดังกล่าวในภายหลังได้ชัดเจนขึ้น

ก่อนที่จะกล่าวถึงวิธีการทำนอร์มอลไลเซชันเพื่อจัดภาพแบบและโครงสร้างของข้อมูลภายในเรคอร์ดของแต่ละตารางมีนิยามคำศัพท์พื้นฐานต่างๆ ที่เกี่ยวข้องและควรทราบดังนี้

1.1 ตาราง/ไฟล์(T/F : Tables/files) เป็นหน่วยที่ใช้ในการจัดเก็บชุดข้อมูลให้เป็นระบบ

ในการกำหนดสร้าง T/F ผู้ออกแบบระบบจะต้องคำนึงถึงหน่วย(Entity) ที่จะใช้ในการจัดเก็บข้อมูลหน่วยที่จะใช้มีหลายภาพแบบ อาจจะเป็นตัวพนักงาน แบบฟอร์มสมัครงาน ใบสั่งซื้อสินค้า ใบลงทะเบียนของสมาชิกเป็นต้น การกำหนดตารางตามหน่วยที่จะใช้จัดเก็บข้อมูล อาจจะไม่ใช่เรื่องง่ายเสมอไป เพราะบางทีจาก หนึ่ง หน่วยจัดเก็บข้อมูลอาจจะแตกแยกออกเป็นหลายตาราง หรือเป็นตารางเดียวก็ได้ ทั้งนี้ขึ้นอยู่กับว่าผู้ออกแบบระบบ มองเห็นลักษณะการจัดกลุ่มของข้อมูลในภาพของฟิลด์ ที่มีความสัมพันธ์กันขึ้นมาเป็นภาพแบบของเรคอร์ดเช่นไร กระบวนการจัดภาพแบบของฟิลด์ขึ้นมาเป็นเรคอร์ด นี้เป็นขั้นตองหนึ่งของวิธีการนอร์มอลไลเซชัน

1.2 ข้อมูล/ฟิลด์(Data items/fields)เป็นหน่วยของข้อมูลในเรคอร์ดของหน่วยที่จัดเก็บข้อมูลหลังจากที่กำหนดหน่วยแล้ว ผู้ออกแบบระบบจะต้องกำหนดว่าแต่ละเรคร์ดจะประกอบด้วยหัวข้อข้อมูล(DI : Data Items)ใดบ้างและมีค่าเป็นแบบใด แต่ละฟิลด์ควรจะกว้างยาวเท่าใดจึงจะเหมาะสม นอกจากนี้ยังต้องคำนึงถึงความจำเป็นในด้านการเรียกใช้ค่าของฟิลด์ในภายหลังด้วย หลักการ สำคัญๆ ของการกำหนดฟิลด์ได้แก่

1.2.1. แบ่งฟิลด์ใหญ่ออกเป็นฟิลด์ย่อยๆ เช่นฟิลด์ชื่อสมาชิก(NAME)ควรแบ่งเป็น 2 ฟิลด์(First_name, Last_name) หรือฟิลด์ที่อยู่อาจแบ่งเป็น 3 ฟิลด์(ADDRESS, PROVICE, CPOST) เป็นต้น

1.2.2. ไม่ควรกำหนดให้มีฟิลด์ของข้อมูลที่ได้จากการคำนวณอีกทอดหนึ่ง เช่น ในกรณีที่ 2 ฟิลด์คือ Quantity กับ Unit_price แล้ว ไม่จำเป็นจะต้องมีฟิลด์ Amount ที่ได้จากการนำเอาฟิลด์ Quantity คูณกับฟิลด์ Unit_price เพราะเมื่อต้องการเอาท์พุทเป็น Amount ก็จะสามารถเขียนโปรแกรมคำนวณค่า 2 ฟิลด์ออกเอาต์พุตตามที่ต้องการได้ ไม่ต้องเก็บค่าที่คูณกันนี้เป็นฟิลด์ในเรคอร์ด เพราะไม่มีความจำเป็น

1.2.3. ควรใช้วันที่ในกรณีที่ต้องการเก็บข้อมูลเป็นช่วงเวลา เช่น ไม่ควรมีฟิลด์อายุ (Age) เพราะอายุจะเปลี่ยนทุกปี ดังนั้นจะต้องเปลี่ยนแก้ไขเรคคอร์ดกันทุกปี วิธีที่ถูกต้องคือมีฟิลด์วันเดือนปีเกิดแทนที่ เมื่อใดต้องการเอาต์พุตเป็นอายุจึงทำการคำนวณอายุ ออกมา

1.2.4. ขนาดของฟิลด์ไม่ควรกว้างเกินความจำเป็น ชื่อของฟิลด์ควรจะสื่อความหมาย และถ้าฟิลด์ใดที่จะไม่ใช้ในการคำนวณควรจัดเก็บเป็นตัวอักษร เช่น CPOST(รหัส ไปรษณีย์) เป็นต้น

1.3 ความซ้ำซ้อนของข้อมูล(RG : Repeating Groups)เป็นลักษณะของการมี DI ที่มีลักษณะเหมือนกันและซ้ำๆกันอยู่ภายในเรคอร์ดถ้าระบบการจัด DI ในเรคอร์ด เป็นเช่นนี้จะถือว่าเป็น การวางภาพแบบระบบข้อมูลที่ไม่ดี ตัวอย่างเช่นในเรคอร์ดของสมาชิกจะมีฟิลด์ชื่อของสินค้าที่ได้มีการซื้อสินค้า รายการฟิลด์ชื่อสินค้าที่ได้ซื้อนี้จะถือว่าเป็นความซ้ำซ้อนของข้อมูลเพราะจะยืดยาวมีหลายฟิลด์ซ้ำๆกัน สมาชิกบางคนอาจซื้อสินค้า 5-10 สินค้า หรืออาจมากกว่า จะต้องมีฟิลด์ ชื่อสินค้า 5 - 10 ฟิลด์หรือมากกว่านั้นตามจำนวนของสินค้าที่สมาชิกได้ซื้อ ดังนั้นการกำหนดโครง

สร้างของเรคอร์ดจะต้องกำหนด จำนวนฟิลด์เอาไว้มากๆ อาจจะเป็น10 ฟิลด์ ,20 ฟิลด์ หรือ 100 ฟิลด์ก็ได้ ซึ่งเกินความจำเป็นกว่าที่จะมีค่าใส่จริง และเป็นการสิ้นเปลืองโดยใช่เหตุ

ภาพที่ 4.12

CUS_CODE	NAME	ST_CODE	ST_NAME	ST_NUMBER	...../.....	ST_CODE	ST_NAME	ST_NUMBER

แสดงรีเลชันของตาราง ที่เกิดการซ้ำซ้อนของข้อมูล.

วิธีการแก้ไขความซ้ำซ้อนของข้อมูลทำได้ง่าย คือจะต้องมีการสร้างตารางใหม่ ขึ้นมารองรับฟิลด์เหล่านี้ เช่น ในที่นี้จะต้องสร้างๆไฟล์ Stock ขึ้นมาเก็บชื่อสินค้าโดยใช้สินค้าเป็นหน่วยของเรคคอร์ดของไฟล์นั้นๆ

ภาพที่ 4.13

CUS_CODE	NAME	ADDRESS	PROVICE	CPOST

EMPLOYEE

แสดงรีเลชันของตารางใหม่ที่มีการแก้ไขการซ้ำซ้อนของข้อมูล.

ภาพที่ 4.14

ST_CODE	ST_NAME	ST_NUMBER	ST_UNIT

STOCK

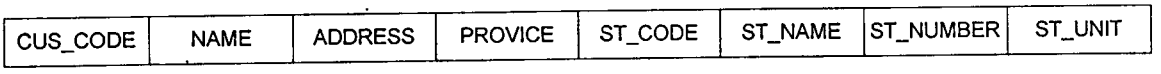
แสดงรีเลชันของตารางใหม่ที่มีการแก้ไขการซ้ำซ้อนของข้อมูล.

1.4 คีย์หลัก(PK :คีย์หลัก) คือฟิลด์ที่มีลักษณะเด่นเฉพาะตัว(Unique) ที่ถูกใช้เป็นหลักในการจัดเรคอร์ดนั่นคือฟิลด์อื่นๆ ทั้งหมดที่อยู่ในเรคอร์ดจะมีความสัมพันธ์กับคีย์หลักฟิลด์นี้ ตัวอย่างของคีย์หลักของ Employee Table คือ CUS_CODE หรือของ Stock Table คือ ST_CODE เพราะฟิลด์อื่นๆ ทุกฟิลด์จะมีความสัมพันธ์ในลักษณะที่เป็นองค์ประกอบส่วนหนึ่งที่เกี่ยวข้องกับ คีย์หลักดังกล่าว

ตัวอย่างต่อไปนี้แสดงลักษณะของฟิลด์ย่อยๆ ในตารางของ Transac ที่ฟิลด์ NAME, ADDRESS, PROVICE ควรขึ้นอยู่กับคีย์หลักของ Employee Table ดังภาพที่ 4.16 และข้อมูลที่เกี่ยวข้องกับสินค้าทั้ง 4 ฟิลด์จะมีความสัมพันธ์โดยตรงกับ ST_CODE ในตาราง Stock ดังภาพที่ 4.16 แต่ข้อมูลของสินค้าไม่ได้มีความสัมพันธ์โดยตรงกับสมาชิก จะมีเพียง ST_CODE เท่านั้นที่มีสัมพันธ์กับตัวเจ้าของในกรณีนี้ฟิลดิ์ NAME, ADRESS, และ PROVICE ควรอยู่ในตาราง Employee เป็น

ฟิลด์ที่ไม่ควรอยู่ใน Transac และฟิลด์ ST_NAME, ST_UNIT ก็ไม่ควรจะอยู่ในตาราง Transac แต่ควรจะอยู่ในตาราง Stock แทน.

ภาพที่ 4.15



TRANSAC

แสดงรีเลชันของตาราง Transac

โดยสรุปลักษณะของคีย์หลักที่ดีคือเป็นคีย์ที่แสดงเอกลักษณ์ของฟิลด์นั้น(Uniqueness) และต้องเป็นคีย์ที่มีความหมายต่อการใช้งานเช่น แผนกบุคคลากรจะจัดเรคอร์ดตามรหัสประจำตัวสมาชิกซึ่งเป็นเลขเฉพาะตัวและมีความหมายคือเป็นเลขที่ๆใช้อ้างถึงหรืออ้างอิงถึงสมาชิก นอกจากนี้ควรจะเป็น คีย์ ที่สั้นและมีขนาดเหมาะสม สามารถนำไปใช้เชื่อมโยง ความสัมพันธะต่างๆ ได้ง่าย

ภาพที่ 4.16

EMPLOYEE

แสดงรีเลชันของตาราง Employee ที่มีคีย์หลักคือ CUS_CODE

1.5 ความสัมพันธ์ระหว่างตาราง(Table Relationships)ในการออกแบบระบบข้อมูล ไม่ว่าจะเป็นระบบของฐานข้อมูลขนาดเล็กหรือขนาดใหญ่ จะต้องมีการกำหนดโครงสร้างของตารางต่างๆ ที่เกี่ยวข้องกันหลายตารางและผลลัพธ์ที่ได้จากการเชื่อมโยงความสัมพันธ์ดังกล่าว ซึ่งได้นำมากล่าวแล้วมีลักษณะการเชื่อมโยงความสัมพันธ์อยู่ 2 ภาพแบบคือ

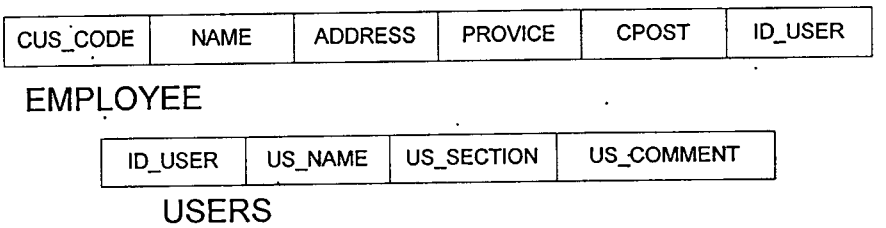
1.5.1 One - to - Many

1.5.2 Many - to - Many

ผลลัพธ์ที่ได้จากการเชื่อมโยงดังกล่าว จะเป็นตัวกำหนดภาพ แบบของเอาต์พุตที่ผู้ใช้ต้องการ

1.6 คีย์สำรอง(Foreign Key)คือคีย์ที่ใช้ในการเชื่อมโยงความสัมพันธ์ของตารางต่างๆเข้าด้วยกัน จะกระทำได้โดยมี Common Key Field ที่จะเป็นฟิลด์ร่วมกันระหว่างตารางต่างๆ

ภาพที่ 4.17

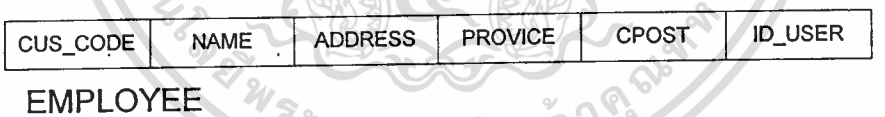


แสดงรีเลชันของตาราง Employee ที่มีคีย์หลักคือ CUS_CODE

ในกรณีของตาราง Employee และตาราง User ซึ่งมีคีย์หลักแตกต่างกันคือตาราง Employee ใช้ CUS_CODE และตาราง User ใช้ ID_USER เป็นฟิลด์หลัก ดังนั้นการที่ จะเชื่อมโยง 2ตาราง นี้เข้าด้วยกันได้ จะต้องกำหนดให้มี Common Key ในตารางหนึ่ง ในที่นี้ฟิลด์ที่เหมาะสมที่จะใช้เชื่อมโยง 2 ตารางเข้าด้วยกัน คือ CUS_CODE แต่ในตาราง Employee ฟิลด์ ID_USER นี้ไม่ใช่คีย์หลักของไฟล์ดังนั้น ฟิลด์ ID_USER ในไฟล์ตาราง Employee จะถูกเรียกว่าเป็นคีย์สำรองแต่คีย์ ID_USER จะเป็นคีย์หลักของตาราง USER.

1.7 คีย์รอง(Secondary Key) หลังจากที่ระบบข้อมูลที่ใช้เป็นคีย์หลักถูกจัดสร้างเรียบร้อยแล้ว ผู้ออกแบบระบบจะต้องคำนึงถึงวิธีการที่จะทำให้การเรียกใช้ข้อมูล(Access) เป็นไปได้สะดวกและรวดเร็วขึ้น เทคนิคหนึ่งที่ใช้กันคือการกำหนดคีย์รองซึ่งเป็นฟิลด์ที่จะทำให้การชี้ตำแหน่ง(Locate) ข้อมูลสะดวกขึ้น

ภาพที่ 4.18



แสดงรีเลชันของตาราง Employeeที่มีคีย์รองคือ NAME

ตัวอย่างเช่น ในตาราง Employee การจดจำระหัสสมาชิก(CUS_CODE) อาจจะเป็นเรื่องยุ่งยากกว่าการจำชื่อสมาชิก แม้ว่า CUS_CODE จะดีกว่าในด้านการเป็นคีย์หลักเพราะสั้นขนาดกระชับและจัดเป็นระบบเรียงลำดับในตารางได้ดีกว่า ดังนั้นผู้ออกแบบระบบจึงอาจกำหนดฟิลด์ NAME ซึ่งเป็นฟิลด์ในตาราง Employee ขึ้นมาเป็นคีย์รองเพื่อใช้ในการทำดัชนี(Index) และเพื่อที่ผู้ใช้ทั่วไปจะได้ค้นหาเรคอร์ดต่างๆตามชื่อสมาชิกได้

4.5.2 จุดมุ่งหมายของวิธีนอร์มอลไลเซชัน

วิธีนอร์มอลไลเซชันเป็นกระบวนการจัดการออกแบบโครงสร้างข้อมูลทีละขั้นตอน ทั้งนี้เพื่อกำจัดปัญหาที่อาจจะเกิดขึ้นในขณะที่ผู้ใช้สั่งแก้ไข(Update) หรือเรียกใช้งาน(Access)ข้อมูลที่อยู่ในเรคอร์ดต่างๆ ในไฟล์ในฐานะของผู้ออกแบบระบบที่ต้องกำหนดโครงสร้างของข้อมูล(คือ

กำหนดฟิลด์ต่างๆ ในเรคอร์ด) ผู้ออกแบบต้องกำหนดวิธีการ ที่จะใช้ดำเนินการเพื่อหลีกเลี่ยงการเกิดปัญหาในการใช้ และจัดการกับข้อมูล

เมื่อนำระบบ มาติดตั้งใช้งานจริง ลักษณะความผิดปกติของการจัดข้อมูล(Data Anomalies) อาจ แบ่งแยกได้เป็น 3 ลักษณะ คือ

1. Insertion anomaly ภาพแบบการจัดข้อมูลที่ดีจะต้องไม่ทำให้การมีอยู่ของหน่วยใดหน่วยหนึ่ง มีอิทธิพลต่อการมีอยู่ของอีกหน่วยหนึ่ง ตัวอย่างเช่นผู้ออกแบบระบบเห็นว่าไม่จำเป็นต้องมีตารางเกี่ยวกับสินค้า(Stock Table) แยกเป็นไฟล์ต่างหากจึงได้จัดเอา 2 ฟิลด์ คือชื่อสินค้า(ST_NAME) และจำนวน(SE_NUMBER) ไปเก็บรวมไว้ในไฟล์เกี่ยวกับรายการซื้อสินค้าของสมาชิกคือตาราง Transac ดังนั่นเมื่อมีสินค้าใหม่ และต้องการเก็บข้อมูลเกี่ยวกับสินค้าใหม่ใส่ไว้ในฐานข้อมูล ก็จะต้องไปเพิ่มเรคอร์ดในตาราง Transac แต่การจะเพิ่มเรคอร์ดของ ตาราง Transac จะกระทำได้ก็ต่อเมื่อมีสมาชิก ได้มาทำการซื้อสินค้าในสินค้าตัวใหม่ด้วย ซึ่งตาราง Transac จะ มีข้อมูลที่เป็นฟิลด์ต่างๆ เกี่ยวกับการเคลื่อนไหวของสินค้า ดังนั้นผู้ใช้ระบบนี้ก็จะเพิ่มเรคอร์ดสินค้าใหม่ไม่ได้ จนกว่าจะมีการซื้อสินค้าในสินค้าตัวใหม่นั้นเข้ามา ซึ่งลักษณะการต้องรีรอข้อมูล เพราะข้อมูลชุดหนึ่ง(ของ สินค้าตัวใหม่) ไปขึ้นอยู่กับข้อมูลอีกชุดหนึ่ง(การซื้อสินค้าในสินค้าตัวใหม่) นี้เรียกว่าเป็นลักษณะไม่ถูกต้องในการแทรกเพิ่มข้อมูล(Insertion Anomaly)

2.Deletion Anomaly เป็นภาพแบบการจัดการข้อมูลที่ตรงข้ามกับแบบแรก จะเกิดขึ้นเมื่อ ผู้ใช้ระบบสั่งลบเรคอร์ดหนึ่งแล้ว อาจจะมีผลกระทบไปลบอีกเรคอร์ดหนึ่งโดยที่ไม่ได้ตั้งใจจะลบไปด้วย

ภาพที่ 4.19

CUS_CODE	NAME	ADDRESS	PROVICE	ST_CODE	ST_NAME	ST_NUMBER	ST_UNIT

TRANSAC

แสดงรีเลชันของตาราง Transac

ในตัวอย่างเดิมถ้าผู้ใช้ระบบสั่งลบเรคอร์ดของสมาชิกออกไป ก็อาจจะเป็นการลบข้อมูลที่เกี่ยวกับสินค้าที่อยู่ในเรคอร์ดนั้นออกไปจากฐานข้อมูลด้วย ทั้งนี้เพราะรายละเอียดเกี่ยวกับข้อมูลของสินค้านั้น(Transaction Description) ถูกจัดให้เก็บรวมอยู่กับฟิลด์ต่างๆ ของการซื้อสินค้าในตารางของ Transac

3. Update anomaly ถ้าจะมีการเปลี่ยนแปลงค่าของฟิลด์ใดๆในฐานข้อมูล ก็ควรจะมีการเปลี่ยนแปลงของค่านั้นเฉพาะในที่แห่งเดียว ในระบบฐานข้อมูลทั้งหมด ถ้าค่าเดียวจะต้องถูกเปลี่ยนในหลายๆ ที่ แสดงว่าเกิดลักษณะ Update Anomaly ลักษณะนี้จะเป็น ลักษณะของการมีฟิลด์ของข้อมูลซ้ำๆ กัน(Data Redundancy) ตัวอย่างเช่นในตาราง Transac เดิมถ้ามี 25 รายการที่มีการ

ซื้อสินค้าในสินค้าตัวใหม่เข้ามา ฉะนั้นในตาราง Transac ก็จะมีอยู่ทั้งหมด 25 เรคอร์ดที่เป็นรายละเอียดเกี่ยวกับสินค้าเหล่านั้น แต่ต่อมาผู้ใช้ระบบพบว่ารายการสินค้านั้นระบุผิดมาตลอดเช่น " MS-DOS VERSION 6.20 THAI " ทั้งๆ ที่จ ควรจะเป็น " MS-DOS VERSION 6.22 THAI " ดังนั้นเมื่อจะแก้รายชื่อสินค้านั้น ก็จะต้องสั่งแก้ทั้งหมดใน 25 เรคอร์ด แทนที่จะสั่งแก้ที่เดียวทีเดียว

ลักษณะการซ้ำซ้อนของข้อมูล(Data Redundancy) นี้ไม่ควรจะจัดให้มีในภาพ แบบโครงสร้างใดๆ ตั้งแต่ในช่วงแรกของการออกแบบระบบแล้ว แต่อย่างไรก็ตาม จุดที่จะยอมให้ความซ้ำซ้อนของฟิลด์ได้ก็คือ ฟิลด์ที่ทำหน้าที่เป็นคีย์นอก(Foreign Key) ที่จะใช้เชื่อมโยงความสัมพันธ์ระหว่างข้อมูลของตารางต่างๆได้ การป้องกันและควบคุมความซ้ำซ้อน จึงเป็นปัจจัยที่มีสำคัญต่อการออกแบบระบบฐานข้อมูล จากลักษณะความผิดพลาดของการวางโครงสร้างระบบข้อมูล(Data Anomalies) ดังกล่าวได้ทำให้เกิดปัญหาเกี่ยวกับการจัดการข้อมูล ไม่ว่าจะเป็นการแทรก เพิ่มข้อมูล การสั่งลบข้อมูล หรือการแก้ไขข้อมูลในฐานข้อมูล ดังนั้นเทคนิคการวิธีนอร์มอลไลเซชันจึงถูกพัฒนาขึ้นมาเพื่อพยายามแก้ไขปัญหาดังกล่าว และกำหนดรูปแบบโครงสร้างของข้อมูลที่ถูกต้องและเหมาะสมออกมาใช้งาน

4.5.3 ลำดับขั้นตอนของวิธีนอร์มอลไลเซชัน

กระบวนการวิธีนอร์มอลไลเซชัน เริ่มต้นด้วยการพิจารณาจากมุมมองของผู้ใช้ระบบ (serview) ได้แก่สิ่งที่ผู้ใช้มองเห็นหรือสิ่งที่เป็นเอาต์พุตที่ต้องการจากระบบ หลังจากนั้นจะเป็นการพิจารณาฟิลด์ต่างๆของเรคอร์ด โดยที่แต่ละหน่วยจะถูกวิเคราะห์ไปตามลำดับขั้นตอน ผลลัพธ์ที่ได้จากการวิเคราะห์ในขั้นแรกจะเรียกว่า First Normal Form, ขั้นที่ 2 เรียกว่า Second Normal Form เช่นนี้ไปเรื่อยๆตามลำดับ ซึ่งอาจต้องวิเคราะห์สืบเนื่องไปจนถึง Fifth Normal Form แต่อย่างไรก็ตามในบางระบบอาจจะวิเคราะห์เพียงแค่ 2 ถึง 3 ขั้นเท่านั้น ขึ้นอยู่กับความซับซ้อนของโครงสร้างของข้อมูลที่มีอยู่ในระบบที่จะใช้งาน

ขั้นที่1: พิจารณาสิ่งที่ผู้ใช้มองเห็น

ขั้นตอนแรกสุดก่อนที่จะวิเคราะห์ระบบการจัดข้อมูล ผู้ออกแบบระบบจะต้องพิจารณาภาพแบบของเอาต์พุต ที่ต้องการว่าเป็นอย่างไร จะมีเรื่องใดบ้างเข้ามาเกี่ยวข้อง มีอะไรที่จะเป็นอินพุตเข้ามา และจะดำเนินการอย่างไรจึงจะได้เอาต์พุตที่ต้องการ

ตัวอย่างต่อไปนี้เป็นการวิเคราะห์การซื้อสินค้าของสมาชิก ในแต่ละครั้งสมาชิกแต่คนจะต้องมีอินพุตต่างๆ เข้ามาหลาย ฟิลด์ด้วยกันและเป็นที่มีข้อมูลซ้ำๆ กัน(Repeating Group)จากการพิจารณาใบรายการซื้อสินค้าดังกล่าว ผู้ออกแบบระบบอาจกำหนด ฟิลด์ ต่างๆที่ต้องการลงในตาราง Transac.ดังในภาพ 4.20

ภาพที่ 4.20

CUS_CODE	NAME	PROVICE	ST_CODE	SE_NUMBER
100001	SOMMAI THONGSAI	LOPBURI	10100110	250
			10100111	100
			10200101	270
			10300101	10
200011	BENJA TUMMADEE	NONTABURI	10100110	320
			10110111	30
			10200101	250
300021	PANYA SAIPUNTA	NONTABURI	10100110	100
			10100111	270
400031	NANTA TUMDANG	SARABURI	10300101	10
500041	NOPDOL MANEERUT	BANGKOK	20100110	320
600051	BANDIT TOYAYAG	BANGKOK	20500601	30

แสดงตารางฐานข้อมูลที่ไม่อยู่ในรูปบรรทัดฐานหรือเรียกอีกอย่างหนึ่งว่าไม่เป็นรีเลชัน

จากภาพ 4.20 เป็นตารางที่ประกอบด้วย แอททริบิวต์รหัสของสมาชิก ที่มีข้อมูลของรหัสสินค้า(ST_CODE) หลายค่าวิธีแก้ปัญหานี้คือการใส่ข้อมูลของสมาชิก ลงไปในทุกรหัสสินค้า (ST_CODE) ดังภาพ 4.20 โดยผลจากการใส่ข้อมูลดังกล่าวนี้จะทำให้รหัสสมาชิกไม่ใช่คีย์หลัก (คีย์หลัก)อีกต่อไปอย่างไรก็ตามเมื่อมีการกำหนดให้ คีย์หลักเป็นรหัสสมาชิกและรหัสสินค้า (CUS_CODE และ ST_CODE)จะทำให้ข้อมูลในรีเลชันนี้อยู่ในภาพแบบบรรทัดฐานขั้นที่หนึ่งแล้วก็ตาม ความผิดพลาดบางอย่างที่อาจจะเกิดขึ้นกับข้อมูลในรีเลชันนี้ก็ยังมีอยู่ดังนี้คือ

1.1. ความผิดพลาดที่เกิดจากการเพิ่มข้อมูล(Insert Anomaly)จากรีเลชันนี้ จะเห็นว่าการที่จะเพิ่มข้อมูลของสมาชิกจะทำได้ต่อเมื่อ สมาชิกรายนั้นได้มีการซื้อสินค้าเข้ามา หากไม่มีการซื้อสินค้าก็จะไม่สามารถเพิ่มสมาชิกเข้ามาได้ และถ้าหากมีการเพิ่มสมาชิกใหม่เข้ามาโดยไม่ได้มีการซื้อสินค้าก็จะมีปัญหาคือ การกำหนดแอททริบิวต์ที่เป็นคีย์หลักของรีเลชันนี้จะเป็นการกำหนดที่ยังไม่เหมาะสม จากกฎความบูรณภาพของเอนทิตี้(The Entity Integrity Rule)ที่กล่าวไว้ว่า แอททริบิวต์ที่เป็นส่วนของคีย์หลักจะไม่มีค่าไม่ได้ แต่กรณีนี้รหัสสินค้า(ST_CODE).จะเป็นค่าว่างและมีเพียงข้อมูลของสมาชิก เพียงอย่างเดียว

1.2. ความผิดพลาดที่เกิดจากการลบข้อมูล(Delete Anomaly) การลบข้อมูลในตารางทิ้งไปดังในภาพ 4.21 จะเป็นการลบทั้งข้อมูลสมาชิกและข้อมูลการซื้อสินค้าในฐานข้อมูลออกไป ปัญหาของรีเลชันนี้ คือรีเลชันนี้ประกอบด้วยแอททริบิวต์ที่เกิดจากเงื่อนไขในการใช้งาน กล่าวอีกนัยหนึ่งคือ มีแอททริบิวต์ที่ต้องการใช้งานคู่กันของคีย์หลัก นั่นคือถ้าต้องการลบสมาชิก ข้อมูลสินค้าก็จะถูกลบออกไปด้วย

1.3. ความผิดพลาดที่เกิดจากการปรับปรุงข้อมูล(Update Anomaly) การปรับปรุงข้อมูลของรีเลชันจะทำให้เกิดความยุ่งยากและเสียเวลา รวมถึงอาจก่อให้เกิดความผิดพลาดที่เกิดจาก

การที่ข้อมูลไม่เหมือนกัน ตัวอย่างเช่นการปรับปรุงข้อมูลของสมาชิก 400031 โดยต้องการเปลี่ยนชื่อจังหวัดจาก SINGBURI เป็น BANGKOK ซึ่งการปรับเปลี่ยนข้อมูลนี้จะต้องค้นหารหัสสมาชิก 400031 ทุกตารางมาปรับปรุง ทำให้เสียเวลาและอาจมีโอกาสเกิดความผิดพลาดที่สมาชิกรหัส 400031 บางตารางไม่ได้ถูกเปลี่ยนเป็นชื่อจังหวัดใหม่

ปัญหาที่เกิดขึ้นของรีเลชัน ในตัวอย่างที่กล่าวมาข้างต้น สามารถแก้ไขโดยแตกรีเลชัน(Decomposition)นี้เป็นสองรีเลชันคือรีเลชัน Employee และรีเลชัน Transac ดังนี้

ภาพที่ 4.21

CUS_CODE	NAME	PROVICE	CPOST
100001	SOMMAI THONGSAI	LOBBURI	15000
200011	BENJA TUMMADEE	NONTABURI	11120
300021	PANYA SAIPUNTA	NONTABURI	10100
400031	NANTA TUMDUNG	SARABURI	15135
500041	NOPADOL MANEERUT	BANGKOK	10230
600051	BANDIT TOYAYAG	BANGKOK	10230

CUS_CODE	ST_CODE	SE_NUMBER
100001	10100110	250
100001	10100111	100
100001	10200101	270
100001	10300101	10
200001	10100110	320
200001	10100111	30
200001	10200101	250
300021	10100110	100
300021	10100111	270
400031	10300101	10
500041	20100110	320
600051	20500601	30

แสดงตารางฐานข้อมูลที่อยู่ในภาพบรรทัดฐาน

การแตกรีเลชันดังกล่าวนี้จะแก้ปัญหาความผิดพลาดทั้งสามประเภทที่กล่าวมาข้างต้นได้เช่นการเพิ่มข้อมูลของสมาชิกได้โดยที่สมาชิกนั้นยังไม่มีการซื้อสินค้าหรือการลบข้อมูลการซื้อสินค้าก็ไม่ทำให้ข้อมูลของสมาชิกหายไปจากฐานข้อมุล หรือการปรับปรุงข้อมูลของสมาชิกก็ทำเพียงครั้งเดียวเพราะไม่มีความซ้ำซ้อนของข้อมูล

ขั้นที่ 2: เปลี่ยนสิ่งที่ผู้ใช้มองเห็นให้เป็น First Normal Form (1NF)

จากการใช้ 1NF ผลที่ได้คือหน่วยของข้อมูล(Entity) ที่ไม่มีฟิลด์ซ้ำๆ กันในตัวอย่างข้างต้น จะเห็นได้ว่าเรคอร์ดของตารางดังภาพที่ 4.13. ยังไม่อยู่ในรูปแบบของ 1NF เพราะยังคงมีฟิลด์ที่มีลักษณะซ้ำๆ กันอยู่คือรายการของการซื้อสินค้าของสมาชิกคนหนึ่งคนสามารถลงได้หลายสินค้า

ขั้นตอนโดยสรุป สำหรับจัดให้หน่วยข้อมูลอยู่ในภาพของ 1NF มีดังนี้

1. จัด Data items(ฟิลด์) ซึ่งซ้ำซ้อนกันออกมาเป็นตารางใหม่
2. ดึงเอาคีย์หลักของตารางเดิมมาใส่ไว้ในตารางใหม่ด้วย
3. กำหนดคีย์หลักของตารางใหม่ ที่จะใช้เป็นหลักในการจัด เรคอร์ดของตารางใหม่

จากตัวอย่างรีเลชัน Employee และ Transac รีเลชันทั้งสองอยู่ในภาพแบบของบรรทัดฐานขั้นที่ 2(2NF)แล้ว ในรีเลชันของ Employee มีรหัสของสมาชิกเป็นหลัก เช่น เมื่อทราบค่ารหัสสมาชิกก็จะทราบชื่อ และจังหวัด ของสมาชิก คือชื่อ SOMMAI อยู่ที่จังหวัด LOPBURI หรือในรีเลชัน Transac ค่าผลการเรียนของเกรดที่ได้จะถูกระบุโดยรหัสสมาชิกและรหัสสินค้าดังนั้นค่าของแอททริบิวต์อื่นๆ ที่ไม่ได้เป็นคีย์หลักของรีเลชัน Employee และ Transac สามารถระบุโดยค่าของแอททริบิวต์ที่เป็นคีย์หลัก

แต่ถ้าหากสมมติตัวอย่างที่ 2 Example2 ถูกออกแบบโดยประกอบด้วยแอททริบิวต์ดังนี้

ภาพที่ 4.22

CUS_CODE	ST_CODE	ST_NAME	SE_NUMBER
100001	10100110	MS_DIS VERSION 6.22 THAI	250
100001	10100111	ANTI VIRUS CARD MODEK 2000I	100
100001	10200101	FINGERTRIP	270
100001	10300101	VIDEO NATIONAL MODEL G-20	10
200001	10100110	MS-DOS VERSION 6.22 THAI	320
200001	10100111	ANTI VIRUS CARD MODEK 200I	30
200001	10200101	FIGERTRIP	250
300021	10100110	MA-DOS VERSION 6.22 THAI	100
300021	10100111	ANTI VIRUS CARD MODEK 2000I	270
400031	10300101	FIGERTRIP	10
500041	20100110	CISKETE 3 1/2 ICH	320
600051	20500601	MAG MONITOR 14" MODEL M1450	30

แสดงตารางฐานข้อมูลที่อยู่ในภาพบรรทัดฐาน

จากภาพ 2NF(A) รีเลชัน Exomple2 มีแอททริบิวต์รหัสสมาชิก(CUS_CODE) และรหัสสินค้า(ST_CODE) ประกอบกันเป็นคีย์หลักที่สามารถระบุค่าของชื่อสินค้าและจำนวนของสินค้า(SE_NUMBER)และยังเกิดความสัมพันธ์ที่แอททริบิวต์ชื่อสินค้า(ST_NAME)ที่ไม่ได้เป็นคีย์

หลัก สามารถระบุค่าโดยคีย์หลักที่เป็นคีย์ผสม(CUS_CODE และ ST_CODE) และขณะเดียวกันยังสามารถระบุค่าโดยแอททริบิวต์ที่เป็นส่วนหนึ่งคือรหัสสินค้า(ST_CODE)

ในตัวอย่างการซื้อสินค้า เราสามารถแยกตารางเดิมออกเป็น 2 ตาราง โดยดึงเอาฟิลด์ซ้ำๆ กันออกมาเป็นตารางใหม่ชื่อ Transac ซึ่งจะมีคีย์หลักของตารางใหม่คือ CUS_CODE และ ST_CODE

เพราะฉนั้นรีเลชัน Exomple2 ไม่ได้อยู่ในภาพแบบบรรทัดฐานขั้นที่ 2 และต้องทำการแตกรีเลชันเพื่อลดปัญหาความซ้ำซ้อนของข้อมูลดังต่อเป็นดังนี้คือ

ภาพที่ 4.23

CUS_CODE	ST_CODE	SE_NUMBER
100001	10100110	250
100001	10100111	100
100001	10200101	270
100001	10300101	10
200001	10100110	320
200001	10100111	30
200001	10200101	250
300021	10100110	100
300021	10100111	270
400031	10300101	10
500041	20100110	320
600051	20500601	30

ST_CODE	ST_NAME
10100110	MS_DOS VERSION 6.22 THAI
10100111	ANTI VIRUS CARD MODEL 2000I
10200101	FINGERTIP
10300101	VIDEO NATIONAL MODEL G-20
20100110	DISKETE 3 1/2 INCH
20500601	MAG MONITOR 14" MODEL M1450

แสดงตารางฐานข้อมูล

โดยรีเลชัน Transac มี CUS_CODE และ ST_CODE เป็นคีย์หลักส่วนรีเลชัน Stock มีคีย์ ST_CODE เป็นคีย์หลัก

ขั้นที่ 3: เปลี่ยนเป็น Second Normal Form: (2NF)

2NF คือหน่วยข้อมูลที่ทุกฟิลด์ขึ้นอยู่กับคีย์หลัก ผู้ออกแบบระบบจะต้องสำรวจแต่ละฟิลด์ดูว่ามีฟิลด์ใดบ้างที่เป็นฟิลด์ขึ้นตรงต่อคีย์หลักของเรคอร์ดนั้น(Key Dependency) และฟิลด์ใดบ้างที่ไม่เกี่ยวข้องโดยตรง ขั้นตอนการดำ เนินการในขั้นที่สอง อาจสรุปได้ดังนี้

1. ถ้าฟิลด์ใดมีส่วนเกี่ยวข้องกับคีย์หลักเดิมเป็นบางส่วน ไม่มีความเกี่ยวข้องกันโดยตรงกับส่วนอื่นๆ ทั้งหมดให้แยกออกฟิลด์นั้นไปไว้ในตารางใหม่พร้อมกับส่วนของคีย์หลักนั้นด้วย

2. ถ้ามีฟิลด์อื่นๆ ที่มีส่วนเกี่ยวข้องกับส่วนของคีย์หลักเดิมนั้นเช่นกันให้นำออกไปไว้ในตารางใหม่ด้วย

3. จัดให้ส่วนของคีย์หลัก ที่ยกมาจากตารางเดิมเป็นคีย์หลัก หลักของตารางใหม่

4. ตรวจสอบอีกทีให้แน่ใจว่าแต่ละฟิลด์ ขึ้นโดยตรงต่อคีย์หลัก ของมัน

ในตัวอย่าง First Normal Form ที่ผ่านมาจะเห็นได้ว่าในตาราง Transac มี SE_NUMBER เป็นฟิลด์จำนวนสินค้าซึ่งจะขึ้นโดยตรงต่อคีย์หลัก(คือ CUS_CODE และ ST_CODE) เพราะ

จำนวนสินค้าที่สั่งซื้อจะต้องเกี่ยวข้องกับรหัสชื่อสินค้าในตาราง Transac จึงไม่ได้ขึ้นอยู่กับ ST_CODE อย่างเดียว เมื่อพิจารณาดูฟิลด์ถัดมาคือ ST_NAME จะเห็นได้ชัดว่ารายละเอียดของสินค้ามีส่วนเกี่ยวพันกับ ST_CODE แต่จะเป็นอิสระจาก CUS_CODE นั่นคือ ST_NAME ขึ้นอยู่กับ ST_CODE เพียง Key เดียว จึงควรแยก ST_NAME มาอยู่ในตารางใหม่(Stock)พร้อมกับดึงเอา ST_COADE มาเป็นส่วนหนึ่งของตารางใหม่ด้วย

สำหรับ ST_SELL ซึ่งน่าจะขึ้นโดยตรงต่อ ST_CODE ก็ควรจะแยกมาอยู่ตารางใหม่(Stock Table) ด้วย เพราะเมื่อเราขึ้นราคาสินค้าต่อหน่วยของสินค้าใดๆเราก็จะต้องเช็คกับ ST_CODE ว่าจะขึ้นราคาสินค้ารหัสราคาสินค้าหมายเลขใด แต่อย่างไรก็ตามสำหรับใบสั่งซื้อสินค้าในตาราง Transac ที่ออกไปก่อนหน้าที่จะขึ้นราคานั้น ก็จะต้องมีราคาต่อหน่วยเดิมติดค้างอยู่ด้วยในกรณีนี้ราคาสินค้าต่อหน่วยจะขึ้นอยู่กับทั้ง CUS_CODE และ ST_CODE ที่ปรากฏในใบสั่งซื้อสินค้าในตาราง Transac ดังนั้นราคาสินค้า ต่อหน่วย(ST_SELL) จึงเป็นฟิลด์ที่ต้องอยู่ใน 2 ตารางคือทั้งในตาราง Transac และ ในตาราง Stock.

ขั้นที่ 4: ตรวจสอบลักษณะ Data Amaties ของ Second Normal Form

เมื่อพิจารณาดูฟิลด์ต่างๆ ในตาราง Transac จะพบว่าทุกฟิลด์ขึ้นตรงต่อคีย์หลักคือ CUS_CODE และ ST_CODE นั่นคือใบสั่งซื้อสินค้าในตาราง Transac 1 ใบ จะเป็นตัวกำหนดชื่อลูกค้า 1 คน พร้อมกับเลขที่ลูกค้า ตำบลที่อยู่ เบอร์โทรศัพท์ ของลูกค้าคนนั้น แต่อย่างไรก็ตาม ตำบลที่อยู่ เบอร์โทรศัพท์ของลูกค้าก็จะมีส่วนสัมพันธ์ ต่อหมายเลขประจำตัวต่อลูกค้าอยู่ด้วย ลักษณะเช่นนี้เรียกว่า เป็นความเกี่ยวพันเชิง ซ้อน(Multiple Dpendencies) วิธีการในขั้นตอนที่ 5 คือการจัดทำ 3NF เพื่อจะแก้ไขปัญหานี้

ก่อนที่จะจัดภาพแบบ 3NF ผู้ออกแบบระบบจะพิจารณาลักษณะที่เป็นปัญหาของข้อมูลใน 2NF ตัวอย่างเช่น ถ้าทางบริษัทได้พบกับลูกค้าคนใหม่ที่สนใจสินค้าของบริษัทแต่ยังไม่มีใบสั่งสินค้าเข้ามาทางบริษัทจะยังใส่ เรคอร์ดของลูกค้าคนใหม่ไม่ได้จนกว่าจะมีใบสั่งสินค้าเข้ามา ทั้งนี้เพราะข้อมูลรายละเอียดเกี่ยวกับลูกค้าไปผูกโยงอยู่กับ CUS_CODE และ ST_CODE หรืออีกกรณีหนึ่งเช่น ถ้าบริษัทต้องการจะออก Report/label เกี่ยวกับลูกค้าและที่อยู่ของลูกค้า โปรแกรมในการออกรายงานก็จะต้องมองหารายชื่อของลูกค้าในตาราง Transac แต่เนื่องจากลูกค้าบางคนอาจมีใบสั่งซื้อสินค้าเข้ามาหลายใบ(นั่นคือมีหลายเรคอร์ดของลูกค้าคนนั้น) โปรมแกรม ก็จะต้องกำหนดเงื่อนไขที่จะตัดชื่อลูกค้าที่ซ้ำๆ กันออกไปไม่เช่นนั้นก็จะต้องพิมพ์ ชื่อและที่อยู่ลูกค้าซ้ำๆ กันออกมาใน Report/label ซึ่งลักษณะดังกล่าวนี้เป็นลักษณะ ของ Insert Delete Anomalies.ของโครงสร้างระบบข้อมูลที่เป็น 2NF แล้ว

ขั้นที่ 5: เปลี่ยนเป็น Third Normal Form (3NF)

ดังได้กล่าวในขั้นที่ 4 แล้วว่าอาจจะเกิดกรณีความเกี่ยวพันของข้อมูลในเชิง ของ Transitive dependencies คือบางฟิลด์มีค่าเกี่ยวพันขึ้นอยู่กับฟิลด์อื่นนอกเหนือไปจากคีย์หลัก(ขึ้นอยู่กับทั้งฟิลด์อื่นๆ และคีย์หลักพร้อมๆกัน) ดังนั้น การจัดโครงสร้างข้อมูลให้เป็น 3NF คือการทำให้ฟิลด์ที่ไม่ได้เป็นส่วนหนึ่งโดยตรงสมบูรณ์ของคีย์หลักเป็นอิสระทำให้ไม่เกิดลักษณะของTransitive Dependencies ในขั้นตอนนี้จะต้อง ดำเนินการดังนี้

1. ระบุฟิลด์ที่มีลักษณะเป็น Transitive dependencies
2. ย้ายฟิลด์ที่มีลักษณะดังกล่าวไปไว้ในตารางใหม่
3. ระบุคีย์หลัก ของตารางใหม่
4. จัดให้คีย์หลักของตารางใหม่เป็น Foreign Key ของหน่วยข้อมูลเติมในขั้นตอนนี้ควรจะไม่มีลักษณะของ Transitive dependencies เหลืออยู่ในตารางเดิม

ในตัวอย่างของตาราง INVOICE จะมี NAME, ADDRESS, PROVICE, CPOST และ TELEPHONE ที่เป็นฟิลด์ที่ขึ้นอยู่กับทั้ง CUS_CODE และ ST_CODE แต่ก็มีลักษณะเป็น Transitive dependencies ดังนั้น เรา จึงควรแยกฟิลด์เหล่านี้ออกเป็นตารางใหม่(ชื่อ Employee)โดยที่ฟิลด์ที่จะยังคงอยู่ในตาราง Transac ในฐานะเป็น Foreign Key คือ CUS_CODE เท่านั้น และในตารางใหม่นี้จะใช้ CUS_CODE เป็นคีย์หลัก ด้วย

ย้อนกลับมาพิจารณาตาราง Transac ก็จะพบปัญหาเช่นเดียวกัน คือตารางยังไม่อยู่ในภาพของ Third Normal Form เพราะว่า SE_TOTAL เป็นฟิลด์ ที่ขึ้นอยู่กับ Transac NUMBER(Primary Key) ก็จริงแต่ก็ขึ้นอยู่กับ SE_NUMBER และ SE_SELL(SE_TOTAL = SE_NUMBER X SE_SELL) อีกด้วย ดังนั้นจึงถือ ได้ว่า SE_TOTAL เป็นฟิลด์ที่มีค่าซ้ำซ้อน(redundancy) เกินจำเป็น สามารถ จะตัดทิ้งไปได้ เมื่อใดก็ตามที่ต้องการค่า SE_TOTAL ในโปรแกรม ก็เพียงใช้ค่าของ อีก 2 ฟิลด์คูณกันได้ ผลลัพธ์ที่ได้สุดท้ายเป็น 5 ตาราง คือ Transac, INVOICE, Stock และ Employee ในตัวอย่างนี้ไม่ได้ดำเนินการจัดระบบข้อมูลวิธีนอร์มอลไลเซชันต่อเป็น Forth หรือ Data Anomalites อื่นๆ เช่น Update anomalies หรือลักษณะที่มีคีย์หลักซ้อนกันมากๆ ก็จะต้องดำเนินการทำ Normalization ต่อไปอีกหลายขั้นตอนจนกว่าจะได้ฟอร์มที่สมบูรณ์เป็นอิสระต่อกัน

ขั้นที่ 6: สำรวจความสัมพันธ์กับส่วนอื่นๆ ของระบบฐานข้อมูลทั้งระบบ

1. ดำเนินการวิธีนอร์มอลไลเซชันกับทุกตารางในส่วนอื่นๆของระบบ โดยใช้กระบวนการขั้นที่ 1 ถึง 5 เช่นเดียวกับที่ได้ทำมาแล้วในตัวอย่างที่ผ่านมาเป็นการวิธีนอร์มอลไลเซชันเฉพาะส่วนของระบบออกใบสั่งซื้อสินค้าของตาราง Transac ให้ลูกค้าความจริงแล้วอาจมีเอาต์พุตเกี่ยวข้างอื่นๆที่ต้องดำเนินการเป็นขั้นตอนตามลำดับเช่นเดียวกันเช่นระบบ Inventory - Control เพื่อทำรายงานรายการสำรวจจำนวนสินค้าในคลัง(On-Hand) และจำนวนที่ต้อง Recorder เข้ามาระบบ Customer Account เพื่อจัดการแบบฟอร์มบัญชีลูกค้าและ credit application ของลูกค้า

ระบบลูกหนี้ ซึ่งจัดการคำนวณยอดดุลค้างชำระ

ระบบสำรวจยอดขายของพนักงานขาย

คำแนะนำกว้างๆ ในการจัดการคือควรจัดดำเนินงานวิธีนอร์มอลไลเซชันทีละระบบ หรือหากทำงานกันเป็นทีม อาจแบ่งแยกกันไปวิเคราะห์เป็นทีมย่อยทีมละระบบ

2. พิจารณาทบทวนความก้าวหน้าของการจัดการระบบข้อมูลเป็นครั้งคราว ควร จัดให้มีการทบทวนพบปะเพื่อตกลงกันในเรื่องภาพแบบและโครงสร้างของฟิลด์เป็นครั้งคราว ผู้เข้าร่วมประชุมควรจะประกอบด้วยทีมผู้ออกและทีมผู้ใช้ระบบ ทั้งนี้เพื่อที่จะทำให้ได้ เอาต์พุตตรงตามความต้องการ การประชุมร่วมกันจะทำให้ได้ภาพแบบของระบบตามต้องการ ได้เร็วขึ้น ไม่เสียเวลาแก้ไขในภายหลัง

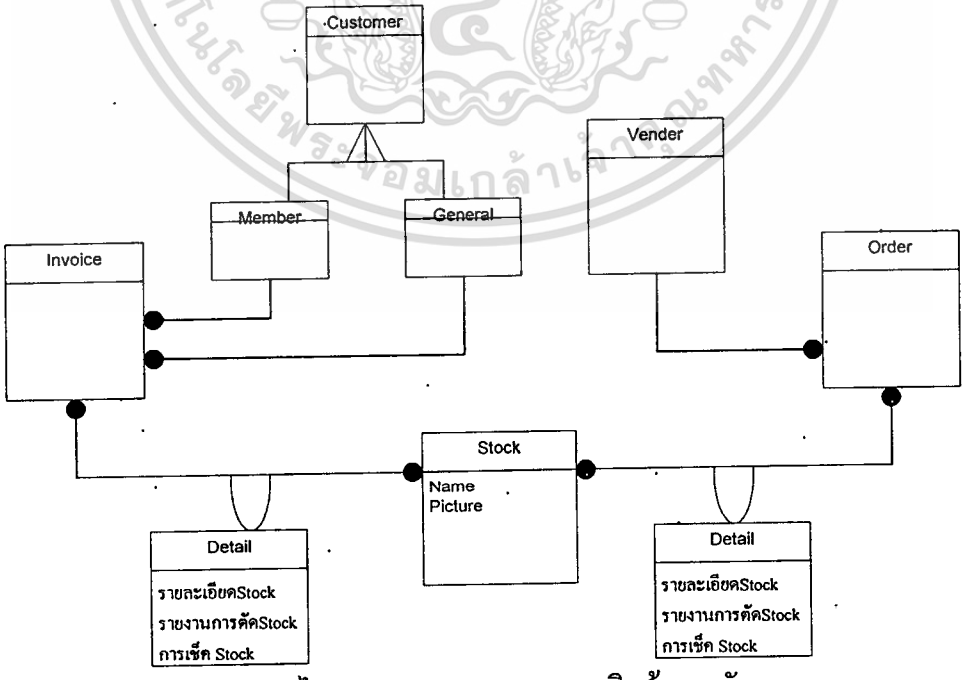
3. เชื่อมโยงส่วนต่างๆ เข้าด้วยกัน(view integration) เป็นระบบจัด การฐานข้อมูลที่ต้องการให้เป็นขั้นตอนสุดท้ายสิ้นสุดกระบวนการวิธีนอร์มอลไลเซชันทั้ง ระบบ

4.6 การออกแบบระบบงานด้วยวิธีการของ OMT

4.6.1 การออกแบบด้วย Object Modeling

ในการออกแบบระบบงานขั้นแรกเรานำออบเจ็กต์มาออกแบบระบบงานตามวัตถุประสงค์ของเราซึ่งจะได้ระบบงานดังรูป

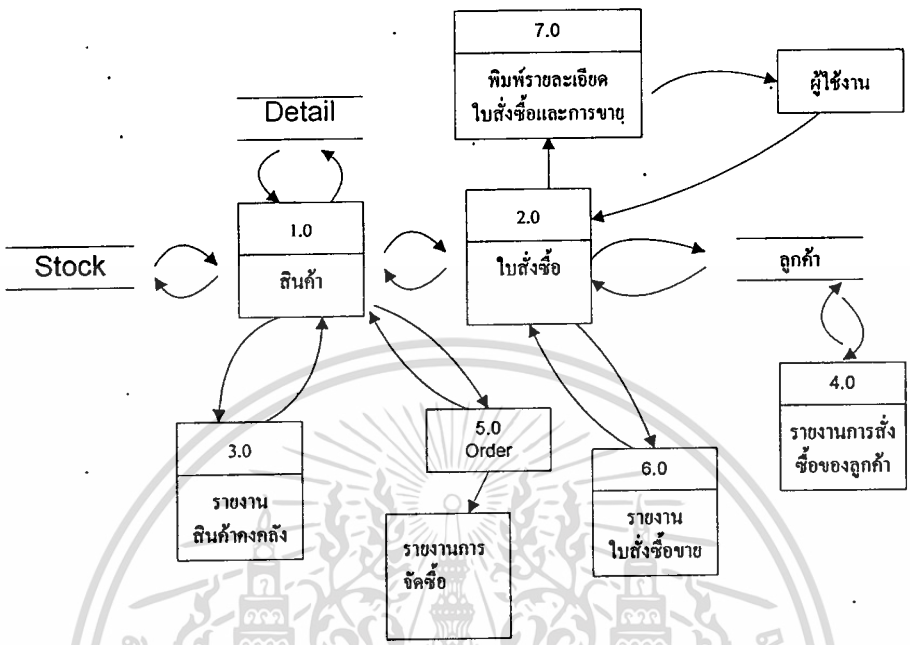
ภาพที่ 4.24



แสดงไดอะแกรมของคลาสระบบสินค้าคงคลัง

4.6.2 การออกแบบด้วย Data Flow Diagram เป็นการออกแบบขบวนการต่างๆของระบบ

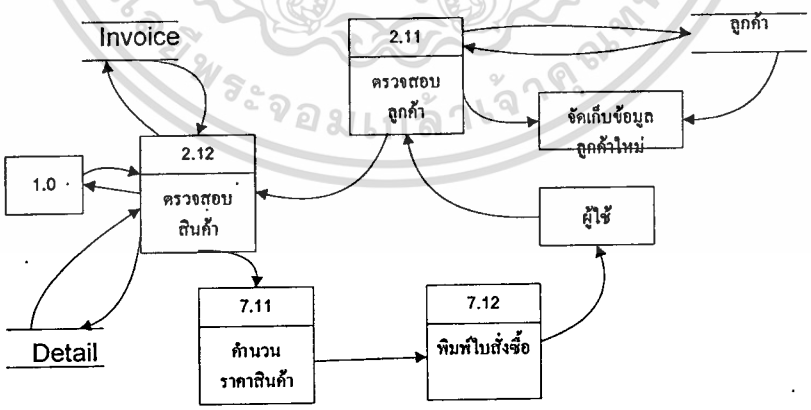
ภาพที่ 4.25



DFD # 0 Stock Diagram

ดาต้าโฟล์ไดอะแกรม 0.0

ภาพที่ 4.26

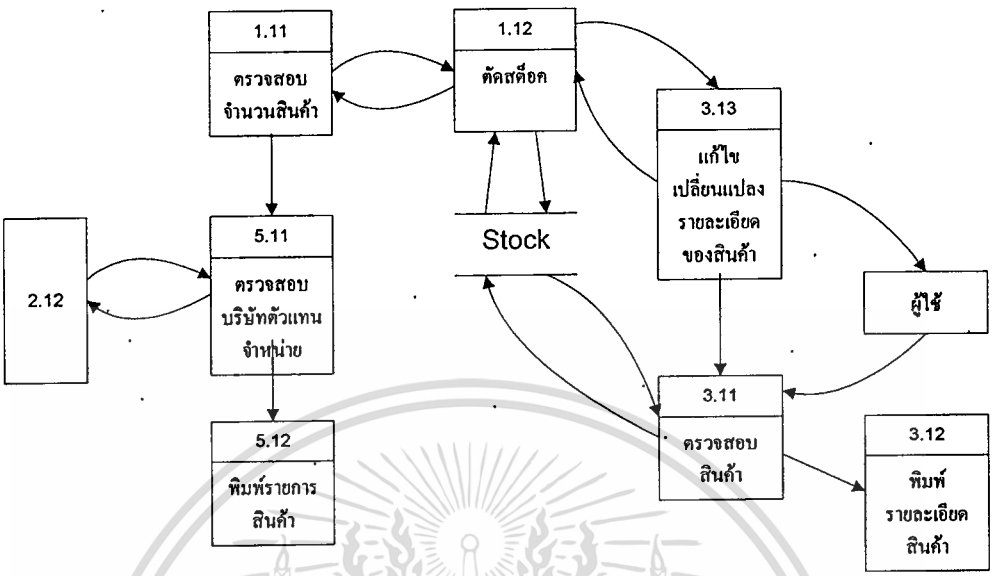


DFD #01 Stock Diagram

แสดงดาต้าโฟล์ไดอะแกรม 01

การออกแบบในขั้นตอนนี้จะเป็นการออกแบบโปรเซสเพื่อให้ได้รายละเอียดให้มากที่สุด โดยการมองในลักษณะของ Top – Down จนลงไปในรายละเอียดไปเป็นตามลำดับขั้น

ภาพที่ 4.27



DFD # 02 Stock Diagram

แสดงดาต้าโฟล์ไดอะแกรม 02

4.6.2.1 ระบบงานคลังสินค้า

ระบบงานคลังสินค้าเป็นระบบงานที่ควบคุมการเบิกจ่ายสินค้าทั้งหมดในระบบ นอกจากนี้ยังมีระบบการสั่งซื้อเพื่อควบคุมการสั่งซื้อสินค้าเมื่อสินค้าหมดอีกด้วย โดยหลักการของระบบคลังสินค้ามีดังนี้

ระบบจะเริ่มต้นจาก ลูกค้าสั่งซื้อสินค้าผ่านระบบ Invoice ระบบ Invoice นี้จะมีการตรวจสอบลูกค้าดังกล่าว หากเป็นลูกค้าใหม่ก็จะเก็บรายละเอียดของลูกค้าใหม่ลงในฐานข้อมูลด้วย โดยลูกค้าจะต้องนำใบสั่งสินค้ามากรอกเข้าระบบ ผู้ที่กรอกใบสั่งสินค้าเข้าระบบจะเป็นผู้ใช้งานระบบคลังสินค้านี้ ในขณะที่ผู้ใช้งานระบบคลังสินค้าป้อนรายการในใบสั่งสินค้า ระบบตรวจสอบสินค้าแต่ละตัวที่สั่งซื้อหากสินค้าใดหมดหรือมีไม่พอสำหรับการสั่งซื้อ ก็จะมีการแจ้งให้ทราบและจะบันทึกสินค้าที่ต้องสั่งซื้อส่งไปยังโปรเซสการสั่งซื้อ เมื่อป้อนรายการสั่งซื้อเสร็จเรียบร้อยแล้ว ระบบจะคำนวณเงินทั้งหมดที่ผู้สั่งซื้อจะต้องชำระ นอกจากนี้แล้วยังต้องพิมพ์ใบเสร็จและรายการที่สั่งซื้อให้กับผู้สั่งซื้ออีกด้วย

ในระบบคลังสินค้านี้ยังมีการนำเสนอรายงานต่าง ๆ อาทิ รายงานสินค้าคงคลัง รายงานการจัดซื้อ รายงานการสั่งซื้อสินค้าของลูกค้า เป็นต้น ซึ่งจะช่วยให้ระบบคลังสินค้านี้มีสมบูรณ์มากยิ่งขึ้น สำหรับรายละเอียดในการทำงานแต่ละขั้นตอนมีดังนี้

4.6.2.2 โพรเซส 1.0 สินค้า

โพรเซสสินค้านี้เป็นโพรเซสในการควบคุมความถูกต้องของสินค้า กล่าวคือจะมีการตัดสต๊อกสินค้า และจัดซื้อสินค้าที่หมดเข้ามาใหม่ นอกจากนี้โพรเซสยังควบคุมการพิมพ์รายงานต่างๆ ไม่ว่าจะเป็นรายงานสินค้าคงคลัง รายงานการจัดซื้อสินค้าที่หมด ซึ่งจะช่วยให้ผู้ใช้งานระบบสามารถตรวจสอบสินค้าได้โดยตรง ในโพรเซสนี้จะเป็นเหมือนหัวใจสำคัญของระบบทีเดียว เพราะจะมีการควบคุมสินค้าทุกชิ้น

4.6.2.3 โพรเซส 2.0 ใบสั่งซื้อ

โพรเซสนี้เป็นโพรเซสแรกที่ผู้ใช้ระบบจะต้องเรียกใช้ กล่าวคือ เมื่อได้รับใบสั่งสินค้ามาแล้ว จะต้องเข้ามากรอกใบสั่งสินค้านี้โดยตรงเพื่อออกใบเสร็จให้กับลูกค้า โพรเซสนี้จะเชื่อมต่อโดยตรงกับโพรเซสสินค้า โพรเซสใบสั่งซื้อนี้จะมีการตรวจสอบราคาของสินค้าที่สั่ง รวมถึงตรวจสอบจำนวนของสินค้าที่สั่งด้วย หากสินค้าไม่พอจ่ายก็จะให้โพรเซสสินค้าจัดการเตรียมซื้อสินค้าเข้ามาจำหน่ายต่อไป ในโปรเซสใบสั่งซื้อนี้จะมีการตรวจสอบรายละเอียดลูกค้าด้วยดังที่กล่าวมาแล้ว เมื่อกรอกใบสั่งซื้อเสร็จเรียบร้อยแล้ว โพรเซสนี้จะสั่งให้มีการตัดสต๊อกโดยส่งผ่านไปยังโพรเซสสินค้าเป็นโพรเซสที่ควบคุมการตัดสต๊อก นอกจากนี้โพรเซสใบสั่งสินค้าจะต้องมีการพิมพ์รายการสั่งสินค้าประจำเดือนและประจำสัปดาห์อีกด้วย

4.6.2.4 โพรเซส 3.0 รายงานสินค้าคงคลัง

โพรเซสนี้เป็นการตรวจสอบสินค้า โดยจะตรวจสอบจำนวนสินค้าที่มีอยู่ รวมถึงรายละเอียดต่างๆ ของสินค้าที่ต้องการตรวจสอบไม่ว่าจะเป็นสี ขนาด น้ำหนัก เป็นต้น ระบบตรวจสอบสินค้านี้ผู้ใช้สามารถเรียกใช้ได้โดยตรงหรือจะเรียกใช้ผ่านโพรเซสสินค้าก็ได้ เมื่อสั่งให้มีการตรวจสอบรายละเอียดของสินค้าใดๆ ผู้ใช้จะต้องกรอกรหัสของสินค้าให้ถูกต้อง เมื่อป้อนรหัสสินค้าอย่างถูกต้องเรียบร้อยแล้ว รายละเอียดต่างๆ ของสินค้าที่ต้องการจะปรากฏขึ้นมา

4.6.2.5 โพรเซส 4.0 รายงานการสั่งซื้อของลูกค้า

โพรเซสนี้เป็นโพรเซสสำหรับควบคุมการออกรายงานการสั่งซื้อสินค้าของลูกค้าแต่ละคน นั่นคือ ผู้ใช้งานระบบคลังสินค้านี้สามารถพิมพ์รายงานย้อนหลังได้ว่าลูกค้าคนใดซื้อสินค้าใดไปเป็นจำนวนเท่าใด หรือกล่าวอีกนัยหนึ่งก็คือ พิมพ์ประวัติการซื้อสินค้าของลูกค้านั่นเอง ในโพรเซสนี้ต้องป้อนรหัสลูกค้าลงไปในระบบอย่างถูกต้อง จากนั้นโพรเซสจะทำการค้นหารายการซื้อสินค้าของลูกค้าแล้วนำมาพิมพ์รายงานต่อไป

4.6.2.6 โพรเซส 5.0 การสั่งซื้อสินค้าที่ขาดเพิ่ม

โพรเซสการสั่งซื้อสินค้าที่ขาดเพิ่มนี้ เป็นโพรเซสที่ช่วยให้ระบบคลังสินค้านี้มีความสมบูรณ์มากขึ้น โดยที่หากสินค้าใดหมดหรือไม่พอจำหน่ายจะมีการเตือนไปยังผู้ใช้งานระบบนี้ จากนั้นจะมีการพิมพ์รายงานสินค้าที่ต้องสั่งซื้อเพิ่มเข้ามาเพื่อจำหน่าย ดังนั้นโพรเซสนี้จะช่วยให้การ

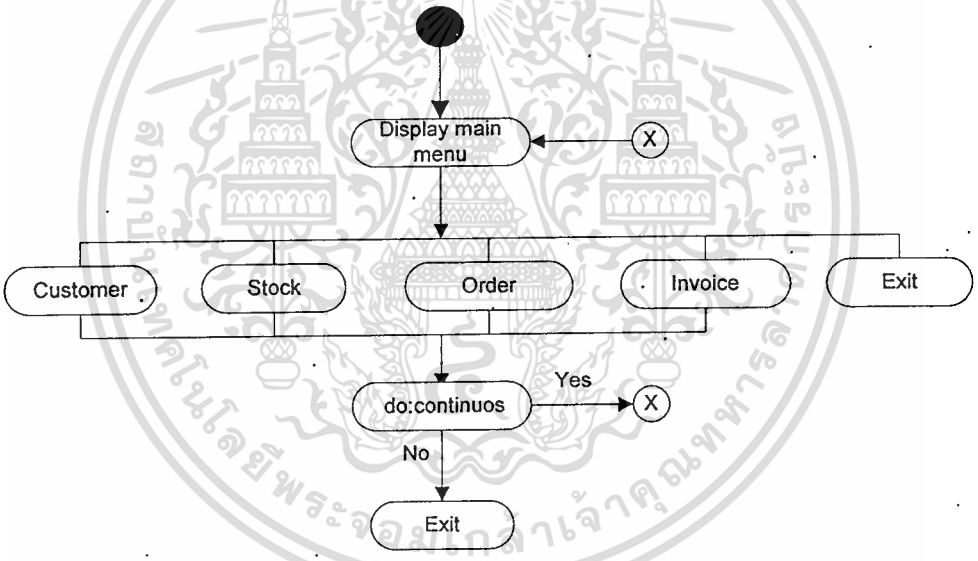
ตรวจคลังสินค้าให้ได้ผลมากขึ้นอีกด้วย โดยที่ในโพรเซสที่ 1.0 ก็มีการตรวจสอบรายละเอียดสินค้าแต่ละตัวอยู่แล้ว แต่ในโพรเซสนี้จะช่วยให้ทราบถึงสินค้าที่จะต้องสั่งซื้อทุกรายการ

4.6.2.7 โพรเซส 6.0 รายละเอียดใบสั่งซื้อและการขาย

เมื่อลูกค้าต้องการสั่งซื้อสินค้าก็สามารถกรอกรายการสินค้าได้ จากนั้นจึงได้นำใบรายการสินค้าที่ลูกค้ากรอกนี้มาป้อนเข้าเครื่องคอมพิวเตอร์เพื่อตัดสต๊อก และคำนวณยอดเงิน เมื่อเสร็จกระบวณการตัดสต๊อกและคำนวณเงินแล้ว ก็จะพิมพ์ใบเสร็จให้กับลูกค้า ในโปรเซสนี้เองเป็นการพิมพ์ใบเสร็จให้ลูกค้า หรือกล่าวอีกนัยหนึ่งก็คือพิมพ์รายละเอียดใบสั่งซื้อและการขายนั่นเอง

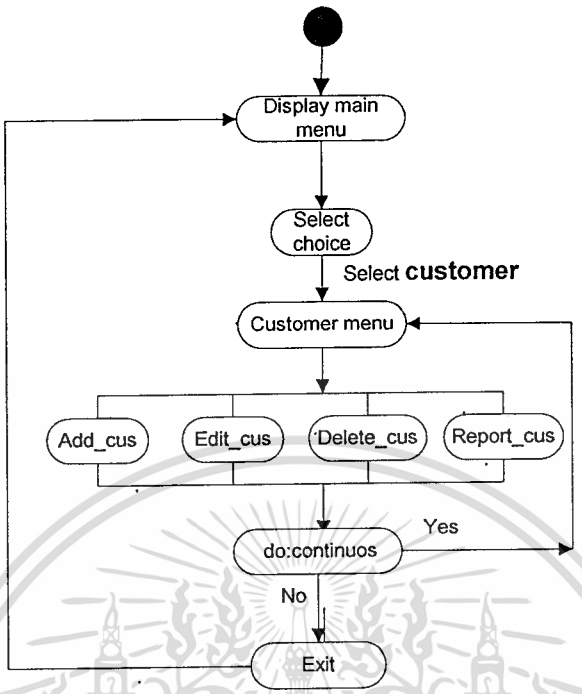
4.6.3 การออกแบบด้วย เสตจไดอะแกรม เป็นการออกแบบการทำงานตามสถานะต่างๆของแต่ละโปรเซส

ภาพที่ 4.28



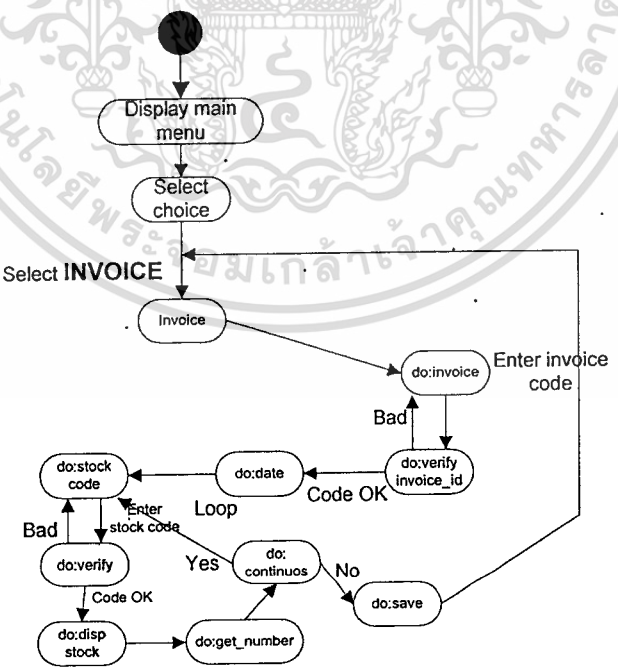
แสดงสเต็ทไดอะแกรมของระบบงานสต๊อค

ภาพที่ 4.29



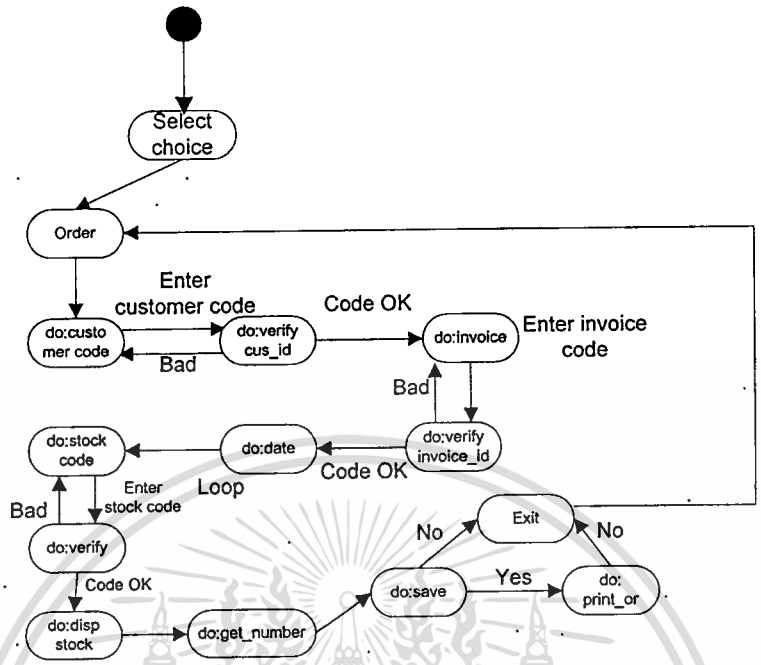
แสดงสเต็ทไดอะแกรมของคลาส Customer

ภาพที่ 4.30



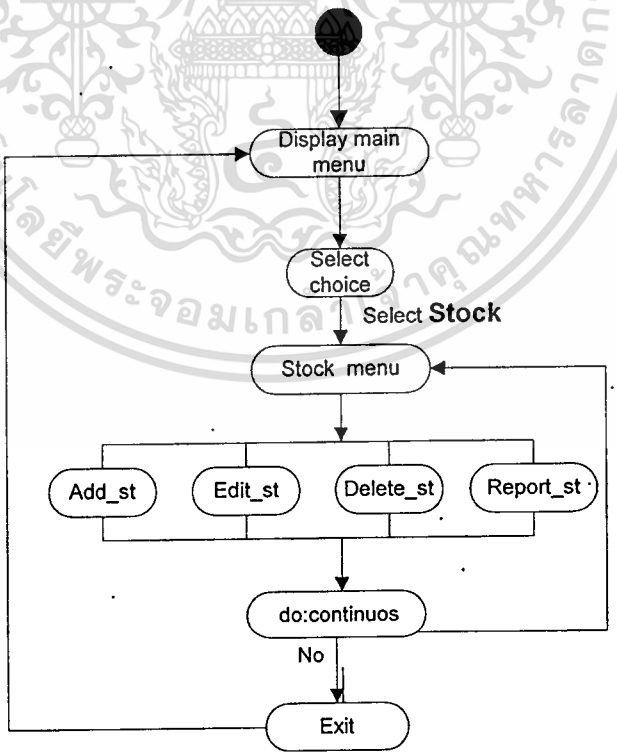
แสดงสเต็ทไดอะแกรมของคลาส Invoice

ภาพที่ 4.31



แสดงสเตจไดอะแกรมของคลาส Order

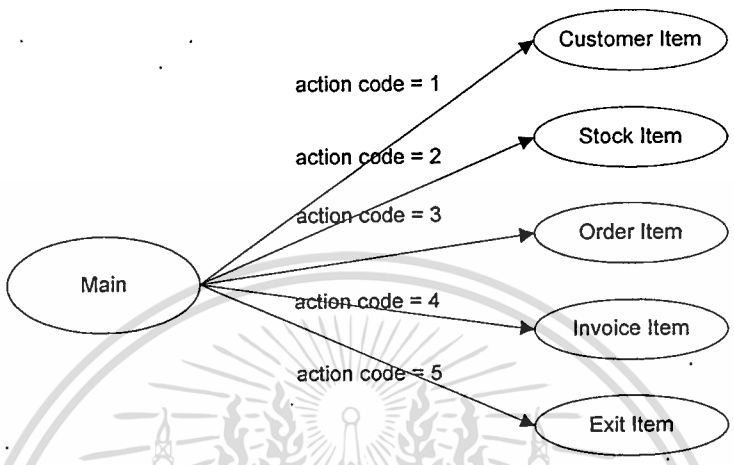
ภาพที่ 4.32



แสดงสเตจไดอะแกรมของคลาส Stock

4.6.3 การออกแบบ Function model เป็นการออกแบบฟังค์ชั่นหรือหน้าที่ ที่ใช้งานในคลาสต่างๆ

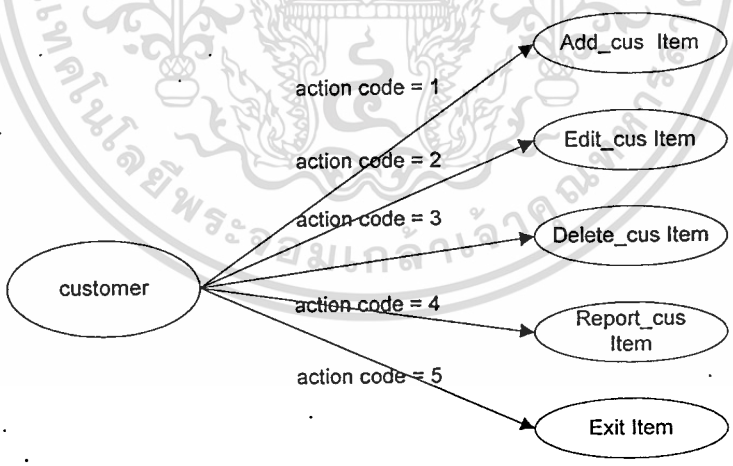
ภาพที่ 4.33



Functional model on main menu

แสดงฟังก์ชันโมเดลของคลาส Shop

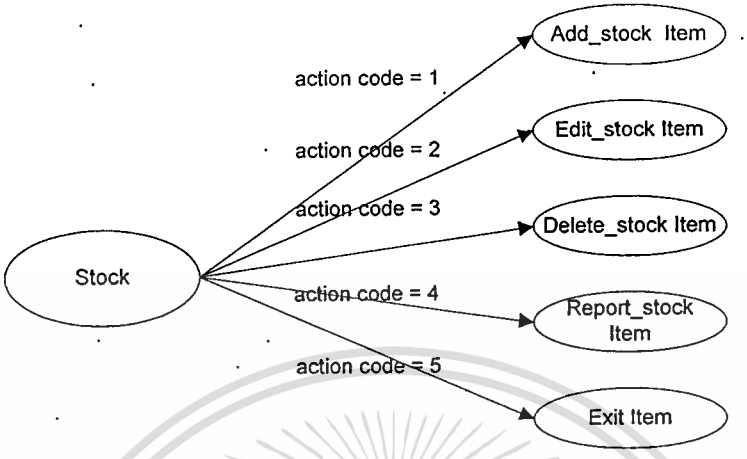
ภาพที่ 4.34



Functional model on customer menu

แสดงฟังก์ชันโมเดลของคลาส Customer

ภาพที่ 4.35



Functional model on Stock menu

แสดงฟังก์ชันโมเดลของคลาส Stock

4.7 การแปลงข้อมูลจากคลาสมาเป็นตาราง[1]

การออกแบบเมื่อเสร็จเรียบร้อยแล้ว เนื่องจากเราใช้กับฐานข้อมูลแบบเชิงออบเจ็กต์ จึงจำเป็นต้องทำการแปลงคลาสให้อยู่ในรูปของตารางได้ดังนี้

```
CREATE ROW TYPE Address_emp
        (       number char( 10 ),
                soi     char( 20 ),
                road    char( 20 ),
                squad   char( 20 ),
                villege char( 20 ),
                aumpore         char( 20 ),
                province        char( 20 ),
                zip             integer,
                telephone       char( 11 ) );

CREATE TABLE Customer
        (       cus_id          char(9) คีย์หลัก ,
                firstname       char( 25 ),
                lastname        char( 25 ),
```

```
        picture         opaque,
        describe        char( 30 ),
        emp_addr        address_emp,
        emp_birdthday   date );

CREATE ROW TYPE Official
    (   position char( 30 ),
        base_salary     numeric( 12,2 ) ,
        ) UNDER address_emp;

CREATE ROWE TYPE Member (
        credit  numeric( 12,2 ),
        debit   numeric( 12,2 ) CHECK (debit <= credit )
        ) UNDER address_emp;
CREATE TABLE Invoice
(   invoice_id      char( 10 )คีย์หลัก,
    cus_id          char(9),
    date            date,
    credit          integer,
    comment                 char( 5 ),
    FOREIGN KEY (cus_id ) REFERENCES customer);

CREATE TABLE Stock
(   Stock_id        integer  PRIMARY  KEY,
    name    char( 30 ),
    label   char( 15 ),
    limit   integer,
    unit    char( 15 ),
    comment         char( 15 ),
    invoice_id      char( 10 ),
    FOREIGN KEY (invoice_id )REFERENCES invoice);
```

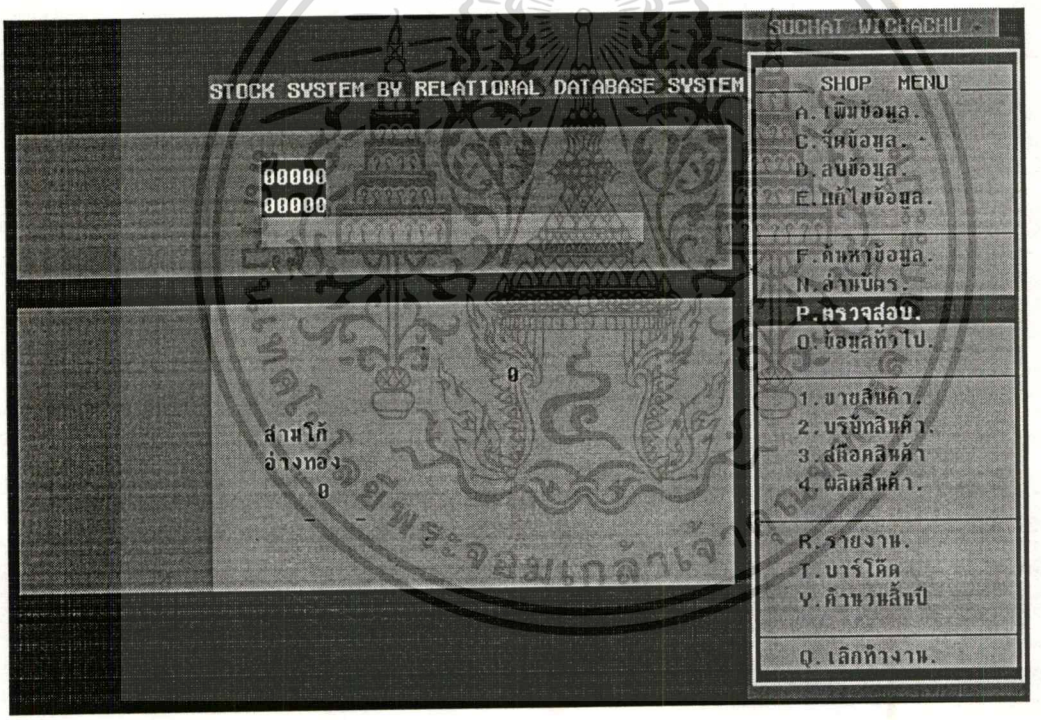
```
CREATE TABLE List_Stock
        (list_id  char( 10 ) PRIMARY  KEY
        number          numeric( 12,2 ),
        buy             integer,
        sell            integer,
        date            date,
        comment                 char( 5 ),
        Stock_id        integer ,
        FOREIGN KEY (Stock_id )REFERENCES Stock );

CREATE TABLE  Detail
(       inv_id          integer  not null,
        number          numeric( 12,2 ),
        buy             integer,
        sell            integer,
        date            date,
        inv_code        char( 10 ),
        Stock_id                integer ,
        FOREIGN KEY (Stock_id )REFERENCES Stock,
        invoice_id      char( 10 ),
        FOREIGN KEY (invoice_id )REFERENCES invoice );
```

4.8 การออกแบบระบบฐานข้อมูลเชิงสัมพันธ์ในระบบงานคลังสินค้า

ระบบฐานข้อมูลของดีเบสโฟร์ (dBASE IV) เป็นโปรมแกรมสำเร็จภาพทางด้านการจัดการฐานข้อมูลเชิงสัมพันธ์ (Relational Database Management System) ที่ผลิตโดยบริษัท Ashtom-Tate มีการพัฒนาขีดความสามารถเพิ่มขึ้นอย่างมากมาย ผู้ที่ยังไม่เคยใช้โปรแกรมสำเร็จภาพทางด้านนี้มาก่อนก็สามารถเริ่มต้นใช้ได้โดยง่าย ส่วนผู้ที่เคยใช้ dBASE II, dBASE III หรือ dBASE IV ได้นำเอา QBE และ SQL ซึ่งเป็น Query Language ของบริษัทไอบีเอ็มเข้ามารวมอยู่ด้วย จึงสามารถใช้งานได้ในระดับมาตรฐานกับเครื่องคอมพิวเตอร์ขนาดใหญ่ ในการจัดการฐานข้อมูลที่ใช้กันอยู่นั้นมีสองแนวทาง แนวทางหนึ่งคือการใช้โปรแกรมภาษาจัดการฐานข้อมูล เช่น การใช้ภาษาโคบอล ซี ปาสคาล หรือภาษาที่สูงขึ้นมา เช่น ดีเบส SQL เป็นต้น การใช้ภาษามาตรฐานที่ใช้กับคอมพิวเตอร์จัดการข้อมูลนั้น ผู้เขียนโปรแกรมต้องจัดการลึกลงไปถึงระดับเรคอร์ด ส่วน SQL สามารถจัดการข้อมูลได้ในระดับเซต แต่ทั้งหมดก็ยังต้องเขียนโปรแกรม การจัดการฐานข้อมูลอีกแนวทางหนึ่งคือการดำเนินโดยไม่ต้องเขียนโปรแกรมวิธีการนี้ผู้จัดการข้อมูลจะสร้างระบบงานโดยไม่ต้องเขียนโปรแกรมเองเลยแม้แต่น้อย แต่สามารถพัฒนาระบบงานขึ้นมาเองได้โดยง่าย ดีเบสโฟร์เป็นซอฟต์แวร์ที่น่าทึ่งตัวหนึ่งที่พยายามเก็บวิธีการจัดการฐานข้อมูลแบบรีเลชั่นแนลไว้หมด โดยคงภาพแบบเดิมคือเขียนเป็นโปรแกรม (.PRG) และทำงานตามโปรแกรมเพื่อว่าผู้คุ้นเคยกับดีเบสทรีจะใช้ได้ทันที ดีเบสโปร์ยังมีส่วนของการจัดการฐานข้อมูลด้วยคำสั่ง SQL ทำได้ดีเบสโฟร์เทียบได้กับซอฟต์แวร์จัดการฐานข้อมูลแบบรีเลชั่นแนลที่กำลังโด่งดังได้ แต่ที่สำคัญอีกส่วนหนึ่งคือดีเบสโปร์มีการจัดการฐานข้อมูลโดยไม่ต้องเขียนโปรแกรมเลยแม้แต่น้อย ดีเบสโฟร์มีตัวสร้างโปรแกรมอัตโนมัติที่จะเขียนโปรแกรมให้เราตามข้อกำหนด ดีเบสโฟร์จัดการฐานข้อมูลรีเลชั่นแนลแบบนี้ด้วยการผ่านคอนโทรลเซนเตอร์ หนังสือดีเบสโฟร์เล่มนี้นำเอาคอนโทรลเซนเตอร์มาเสนอรายละเอียดเพื่อให้เห็นวิธีการจัดการฐานข้อมูลแบบรีเลชั่นแนล โดยไม่ต้องเขียนโปรแกรมเลยแม้แต่น้อย และท่านจะแปลกใจมากเมื่อเห็นดีเบสโฟร์สามารถเขียนโปรแกรม .PRG ให้ท่านได้เป็นพัน ๆ บรรทัด โดยที่ท่านไม่ต้องมีความรู้ในเรื่องการเขียนโปรแกรม คอนโทรลเซนเตอร์ของดีเบสโฟร์สามารถจัดการฐานข้อมูลในภาพแบบการสร้าง QBE หรือ Query by example ซึ่งเป็นหลักการของ DBMS ทั่วไปมีการสร้างฟอร์มด้วยการเขียนจอภาพตามที่ต้องการอย่างง่าย สามารถเขียนรายงานในภาพแบบต่าง ๆ และยังทำลาเบลสำหรับจ่าหน้าซองได้ดี ส่วนที่สำคัญของคอนโทรลเซนเตอร์ส่วนหนึ่งคือส่วนที่เรียกว่า โปรแกรมเยเนอเรเตอร์ คือสร้างโปรแกรมให้เองอย่างอัตโนมัติ โปรแกรมเหล่านี้เราสามารถพิมพ์ออกมาดูได้หรือนำเอาไปคอมพายล์ด้วยคลิปเปอร์หรือดีเบสคอมพายเลอร์เพื่อใช้งานต่อไปได้ ผู้เขียนเห็นว่าหลักการจัดการฐานข้อมูลแบบรีเลชั่นแนลของดีเบสโฟร์ด้วยคอนโทรลเซนเตอร์นี้น่าใช้ เพราะสามารถสร้างระบบงานได้ในเวลาที่สั้น เราสามารถลดเวลาการพัฒนางานได้หลายสิบเท่าตัวทีเดียว การสร้างเมนูและเชื่อมโยงกับโมเดลต่าง ๆ ทำได้ในเวลาไม่กี่นาที แต่หากต้องเขียนโปรแกรมเองอาจต้องใช้เวลาเป็นวันเพื่อสร้างเมนูเหล่านี้ หลักการของการจัด

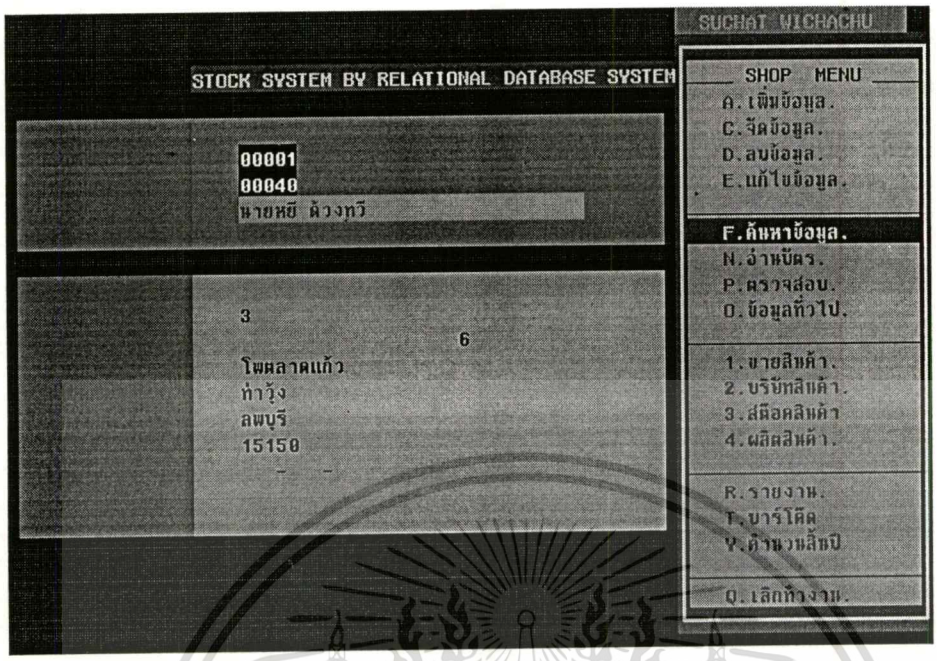
การฐานข้อมูลแบบรีเลชั่นแนลของดีเบสโฟร์ ก็เป็นแบบอย่างที่น่าศึกษา เพราะจะเป็นแนวทางในการเข้าใจระบบจัดการฐานข้อมูลได้ดี สำหรับในรายละเอียดของคอนโทรลเซนเตอร์มีอยู่มาก ผู้เขียนได้เขียนขึ้นมาโดยนำเอาหลักในส่วนที่สำคัญให้ผู้อ่านสามารถทดลองทำตามได้ เพื่อจะได้เข้าใจในการใช้คอนโทรลเซนเตอร์ และสามารถขยายความต่อไปเพื่อให้ได้ระบบงานตามที่ตนเองปรารถนา ผู้เขียนหวังว่าคอนโทรลเซนเตอร์ของดีเบสโฟร์ในเล่มนี้จะเป็นประโยชน์กับผู้สนใจ ผู้เขียนขอเน้นว่าการใช้งานดีเบสโปร์ด้วยคอน โทรลเซนเตอร์นี้ไม่จำเป็ต้องมีความรู้ในเรื่องโปรแกรมมิ่งเลยแม้แต่น้อย แค่ใช้เป็นพิมพ์ได้ก็สามารถเล่นกับคอนโทรลเซนเตอร์ของดีเบสโฟร์ได้ เพราะทุกขั้นตอนจะผ่านเมนูหมดสิ้น หากมีโอกาสลองใช้ดูจะพบว่าคอนโทรลเซนเตอร์สามารถจัดการฐานข้อมูลให้เราได้เกินกว่าที่เราคาดคิดไว้มาก

ภาพที่ 4.36



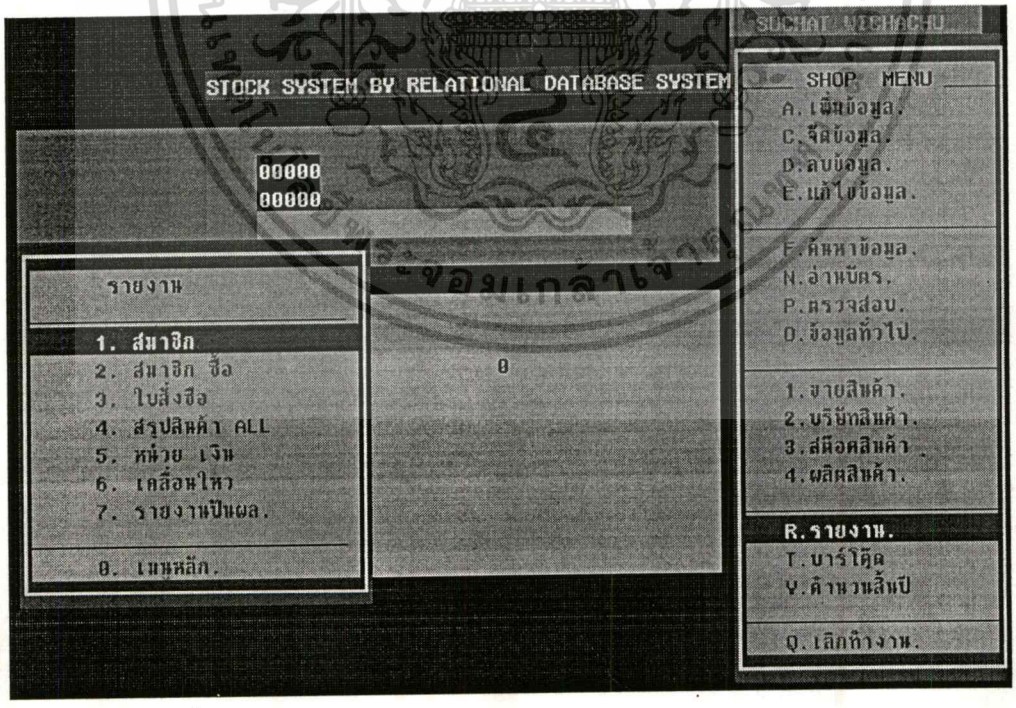
แสดงการทำงานของระบบสต๊อค

ภาพที่ 4.37



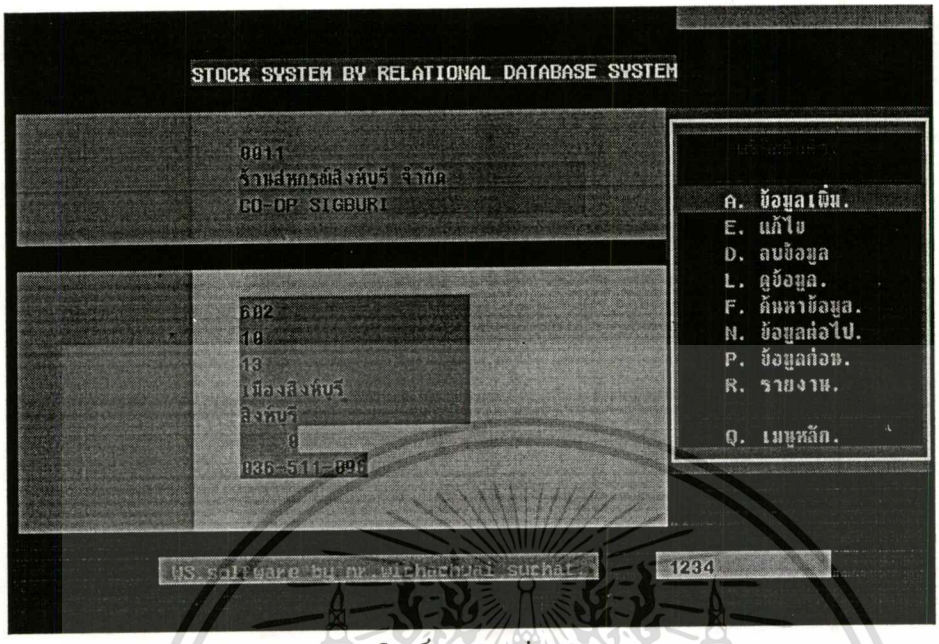
แสดงการทำงานของระบบสต๊อค

ภาพที่ 4.38



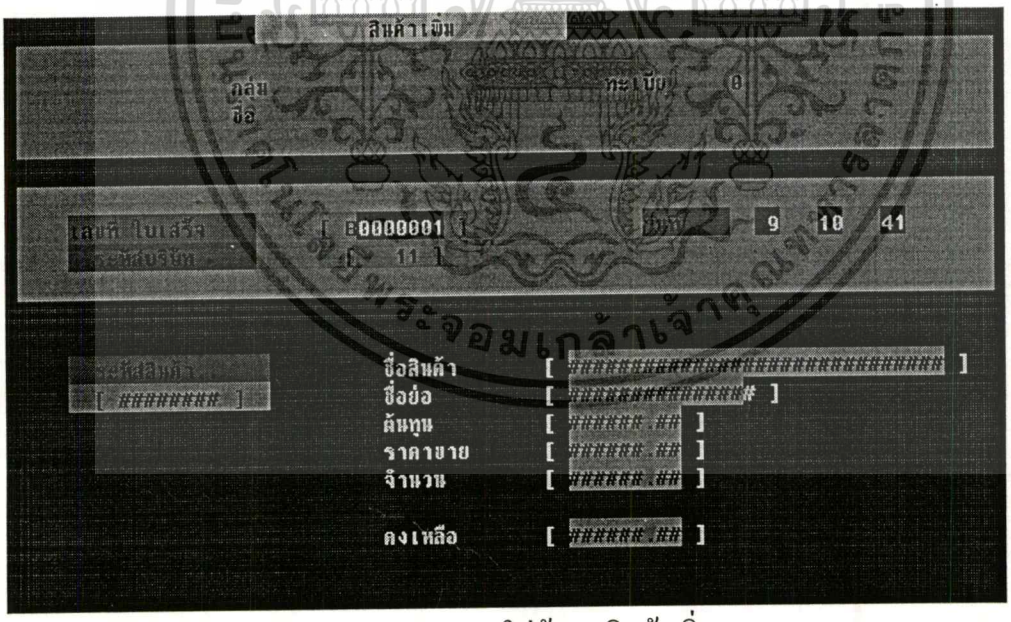
แสดงการทำงานของระบบสต๊อคในการใส่ข้อมูลบริษัท

ภาพที่ 4.39



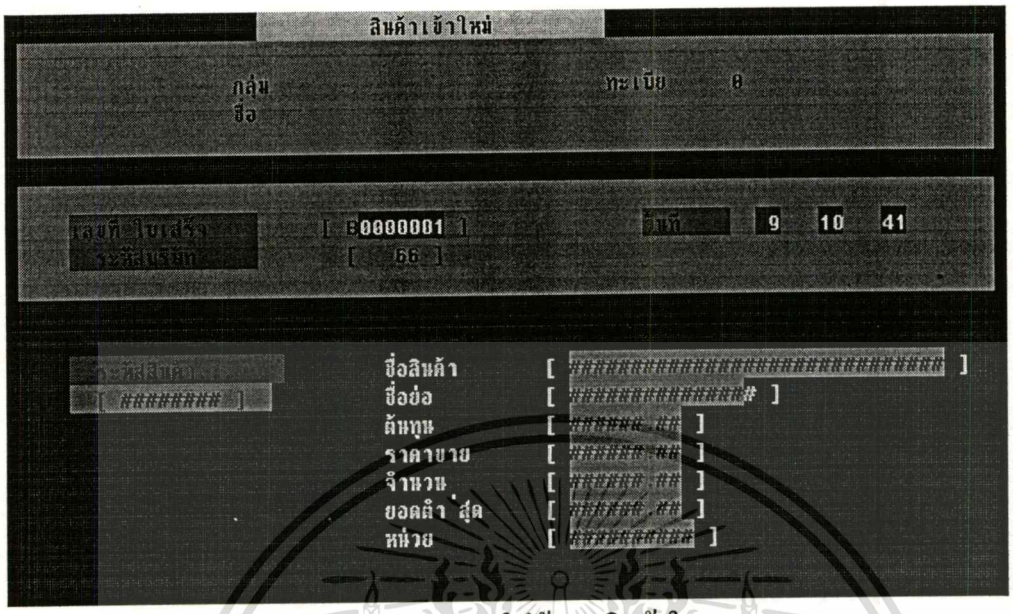
แสดงการทำงานของการพิมพ์รายงานต่างๆ

ภาพที่ 4.40



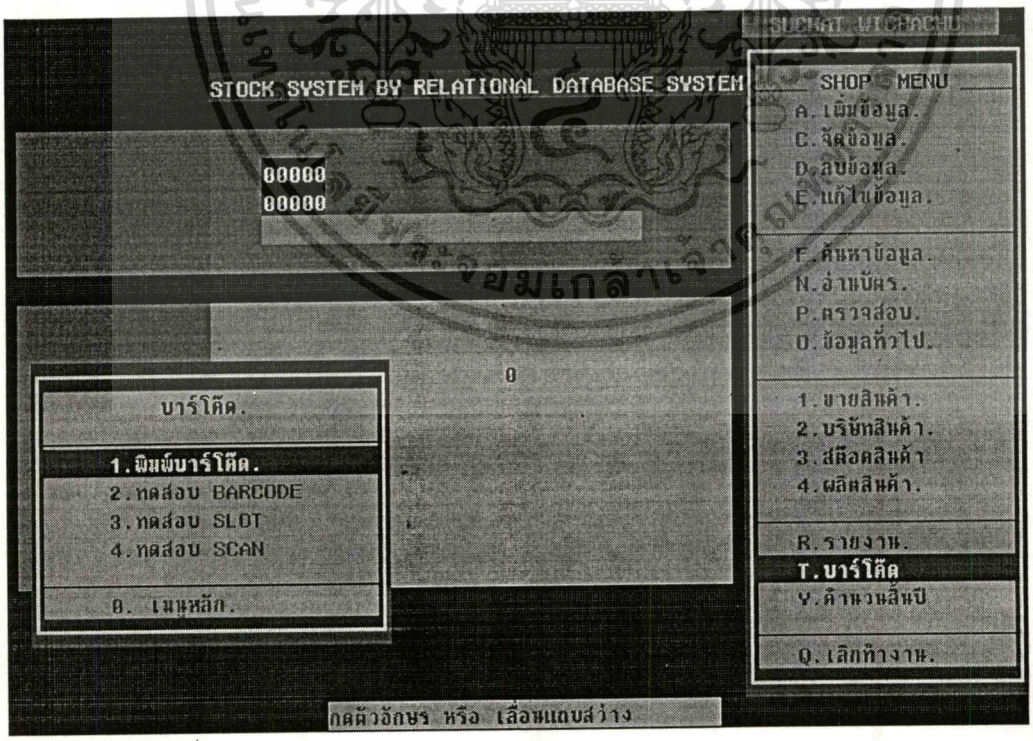
แสดงการทำงานการใส่ข้อมูลสินค้าเพิ่ม

ภาพที่ 4.41



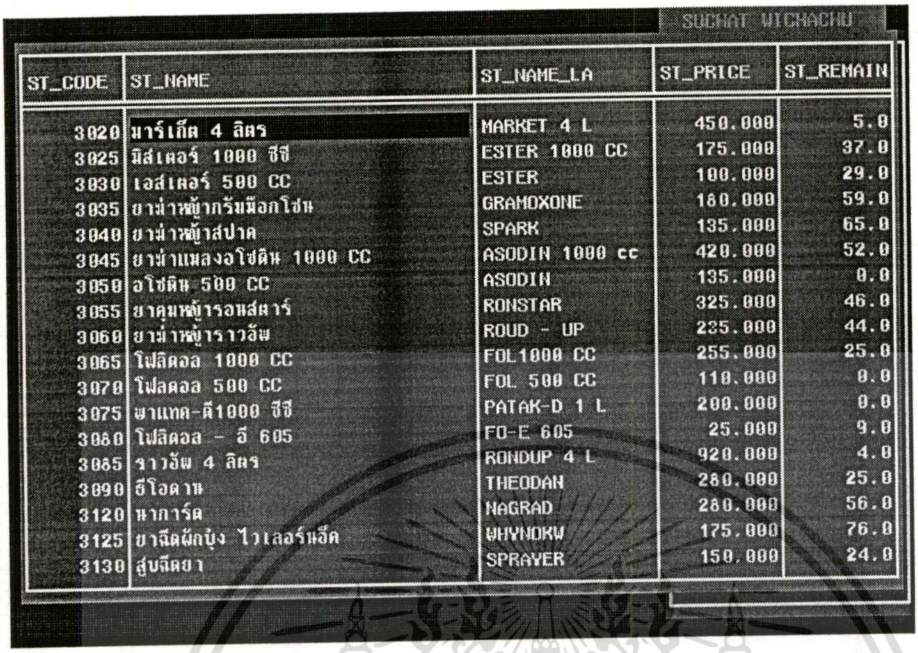
แสดงการทำงานการใส่ข้อมูลสินค้าใหม่

ภาพที่ 4.42



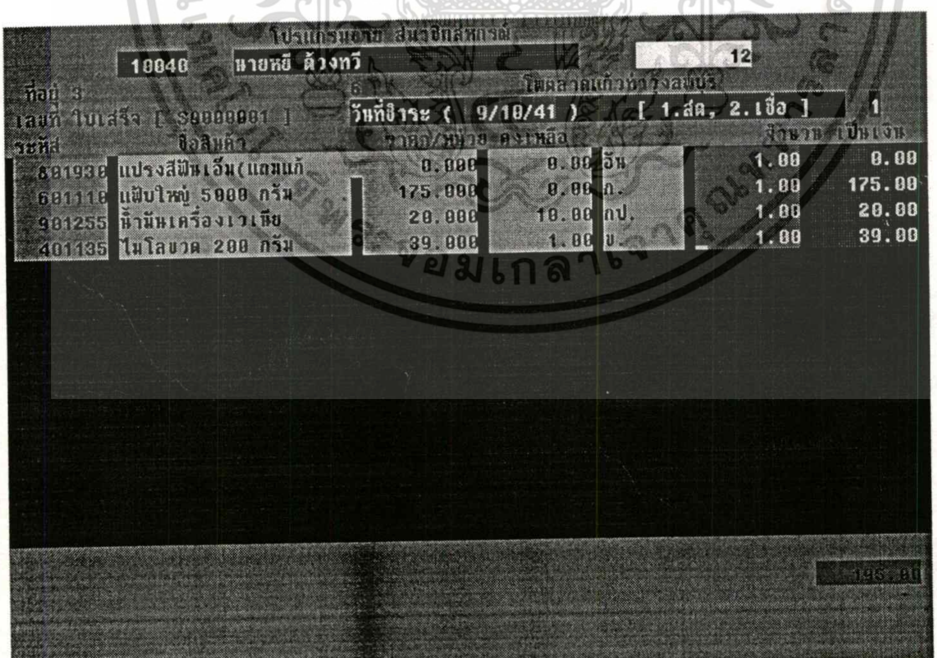
แสดงการทำงานเมนูการพิมพ์บาร์โค๊ด

ภาพที่ 4.43



แสดงการทำงานดูข้อมูลสินค้าในสต๊อค

ภาพที่ 4.44



แสดงการทำงานโปรแกรมการขายสินค้า

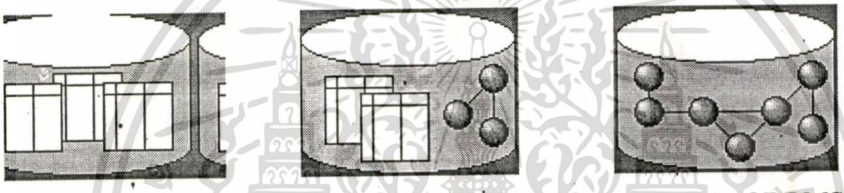
บทที่ 5

ระบบฐานข้อมูลเชิงวัตถุสัมพันธ์และภาษา SQL3

5.1 ระบบฐานข้อมูลแบบออบเจ็กต์โอเรียนเต็ด (ODBMS : Object-Oriented Database System)

ระบบฐานข้อมูลแบบออบเจ็กต์โอเรียนเต็ด[8],[9]คือ การนำวิธีการจัดการข้อมูลเชิงวัตถุมาพิจารณาประกอบร่วมกับความสามารถต่างๆ ของฐานข้อมูลเพื่อนำไปประยุกต์ใช้เป็นระบบจัดการฐานข้อมูลในภาพแบบของวัตถุ ระบบฐานข้อมูลเชิงวัตถุนี้เป็นภาพแบบที่มีความเป็นอิสระต่อกันในการทำงานของวัตถุ จึงทำให้ระบบฐานข้อมูลแบบออบเจ็กต์โอเรียนเต็ดมีการพัฒนาและเทคโนโลยีที่สูงขึ้นเรื่อยๆซึ่งระบบฐานข้อมูลที่มีการพัฒนาและนำมาใช้งานแสดงดังภาพที่ 5.1

ภาพที่ 5.1



ก. แบบ RDBMS ข. แบบ ORDBMS ค. แบบ ODBMS

แสดงลักษณะของฐานข้อมูล RDBMS , ORDBMS, ODBMS

5.2. ระบบฐานข้อมูลเชิงวัตถุสัมพันธ์ (ORDB: Object Relational Database System)

ระบบฐานข้อมูลเชิงวัตถุสัมพันธ์ เป็นระบบฐานข้อมูลที่มีการมองข้อมูลและการจัดเก็บข้อมูลเป็นแบบตารางโดยมีการจัดการฐานข้อมูลในภาพแบบฐานข้อมูลเชิงสัมพันธ์ แต่ในส่วนการติดต่อกับผู้ใช้งานได้นำระบบของออบเจ็กต์ออร์เรียนเต็ดมาใช้งาน ซึ่งระบบออบเจ็กต์ออร์เรียนเต็ดนี้มีการสนับสนุนการพัฒนาและมีการดูแลระบบฐานข้อมูลขนาดใหญ่ๆได้ดีกว่า นอกจากนี้ระบบฐานข้อมูลเชิงวัตถุสัมพันธ์ยังสนับสนุนฟังก์ชันพื้นฐานคือ

1. Persistance of objects
2. Data and transaction integrity
3. Access or query languages
4. Security
5. Management of very large amounts of data
6 Sharing and concurrent multi-user acess
7 Recovery

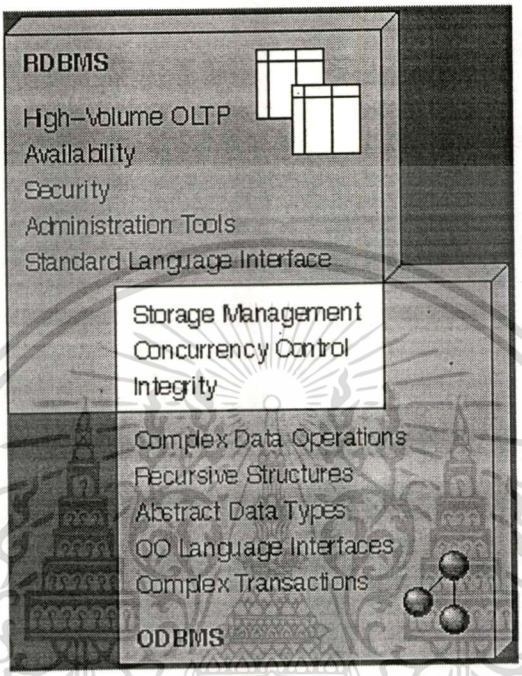
และมีการเพิ่มเติมขยายความสามารถบางส่วนของข้อมูลที่เกี่ยวกับออบเจ็กต์ ขึ้นมาคือ

- Inheritance
- Encapsulation
- Polymorphism

- Object identity

โดยมีรูปแบบและคุณสมบัติพื้นฐานของฐานข้อมูลแบบ RDBMS และฐานข้อมูลแบบ ORDBMS คือ

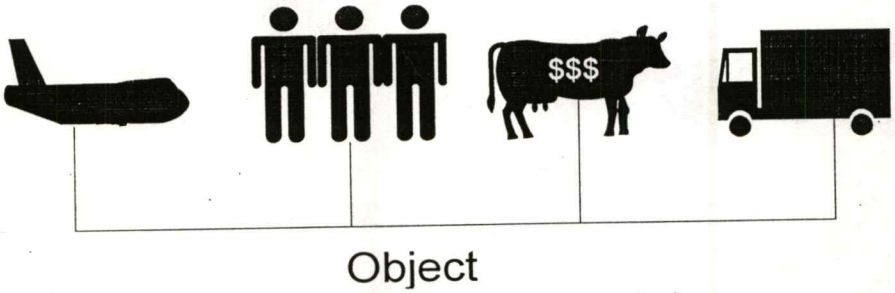
ภาพที่ 5.2



แสดงภาพแบบของข้อมูลแบบ RDBMS และ ODBMS

5.2.1 ออบเจ็กต์ (Objects) เป็นรูปแบบพื้นฐานของข้อมูลเชิงวัตถุซึ่งวัตถุหมายถึงสิ่งใดๆ ที่มีอยู่จริงและสามารถจับต้องได้หรือสิ่งใดๆ ที่สามารถแสดงลักษณะใดๆ ได้ โดยที่ลักษณะเหล่านั้นเป็นคุณสมบัติเฉพาะของแต่ละวัตถุที่สามารถแปรเปลี่ยนไปมาได้ตลอดการคงอยู่ของวัตถุเหล่านั้น

ภาพที่ 5.3



แสดงภาพแบบของข้อมูลแบบออบเจ็กต์

ตามหลักความคิดพื้นฐานทางด้านฐานข้อมูล ออบเจ็กต์จะแสดงคุณสมบัติของการซ่อนข้อมูลคือการกำหนดชื่อและเมทธอดเป็นต้น ออบเจ็กต์แต่ละออบเจ็กต์สามารถติดต่อสื่อสารซึ่งกันและกัน โดยผ่านวิธีการที่เรียกว่า "การส่งผ่านข้อความ"(Message Passing) ที่เป็นตัวกระตุ้นการตอบสนองโดยผ่านโอเปอเรชันที่ซ่อนอยู่ในตัวของออบเจ็กต์ที่เป็นผู้รับ นั่นคือข้อมูลที่จัดเก็บอยู่ภายในออบเจ็กต์ผู้รับจะถูกเข้าถึง และดำเนินการโดยผ่านเมทธอดของออบเจ็กต์นั้นเอง

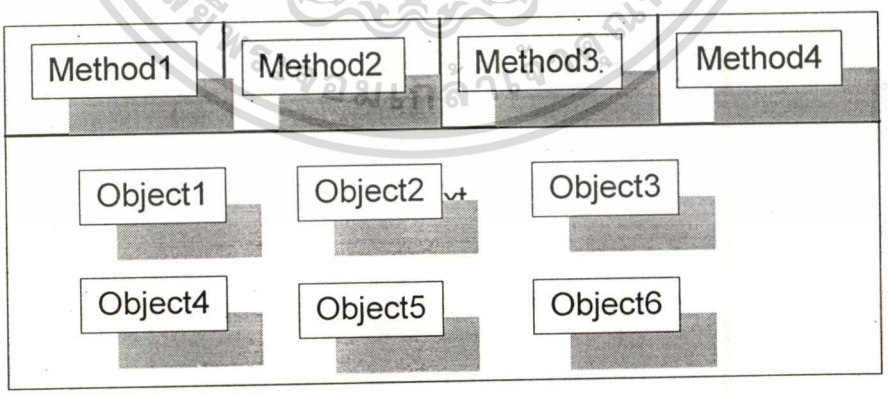
ออบเจ็กต์จะประกอบด้วยส่วนของการเชื่อมต่อและส่วนของการพัฒนา(Implementation) ซึ่งการพัฒนาจะประกอบด้วยส่วนของโครงสร้างข้อมูล(Data Structure) และเมทธอด ซึ่งโครงสร้างข้อมูลจะถูกซ่อนจากภายนอก โดยได้รับการเข้าถึงและจัดการโดยตัวเอง สำหรับเมทธอดนั้นเมื่อมีการเปลี่ยนแปลงจะปราศจากผลกระทบใดๆ กับส่วนที่เชื่อมต่อกัน

5.2.2 แอททริบิวต์ (Attributes) คือโครงสร้างที่เป็นรายละเอียดของออบเจ็กต์ โดยการตั้งชื่อของ จะพยายามให้สื่อความหมายของแอททริบิวต์

5.2.3 คลาส(Class) คือกลุ่มของออบเจ็กต์โดยแบ่งตามลักษณะเฉพาะและการใช้งานหรือออบเจ็กต์ที่มีคุณสมบัติพื้นฐานเหมือนกัน ซึ่งข้อมูลของออบเจ็กต์ที่เก็บอยู่ภายในคลาส จะเรียกว่าอินสแตนท์ของคลาส โดยที่อินสแตนท์จะเป็นส่วนขยายของคลาส และถูกเก็บอยู่ในส่วนของฐานข้อมูล ดังนั้นข้อมูลต่างๆที่เป็นตัวกำหนดคลาสจะถูกเข้าถึง และถ่ายทอดผ่านทางวัตถุอินสแตนท์และ เมทธอดของคลาส

- Instance method คือ เมทธอดที่ถูกประยุกต์ใช้งานกับข้อมูลแต่ละตัวในคลาสที่กำหนด
- Class method คือ วิธีที่ถูกประยุกต์ใช้กับส่วนของคลาสทั้งหมด

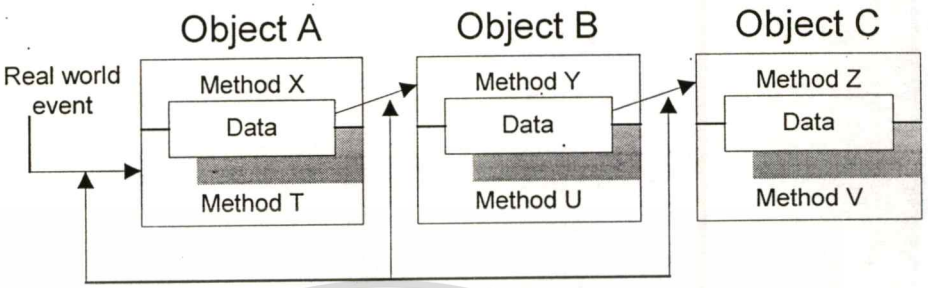
ภาพที่ 5.4



แสดงออบเจ็กต์ที่ใช้ **เม็ททอด** ร่วมกันของคลาส

5.2.4 เมสเสจและเมทธอด (Messages and Methods) เป็นข่าวสารที่ใช้ติดต่อกันระหว่างออบเจ็กต์และวิธีการการใช้ข้อมูล

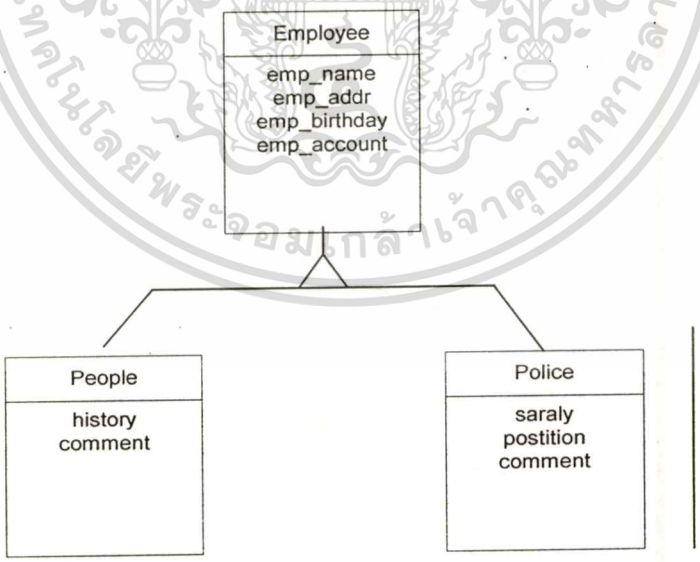
ภาพที่ 5.5



แสดงการส่ง เมสเสจระหว่าง ออบเจ็กต์

5.2.5 **การสืบทอด(Inheritance)คือ** การสืบทอดหรือกระบวนการในการจัดการคลาส เป็นการสร้างส่วนของคลาสใหม่ขึ้นมาโดยมีพื้นฐานมาจากคลาสเดิม แต่จะมีข้อมูลหรือฟังก์ชันที่พิเศษเป็นของตัวเองเพิ่มขึ้นมาจากคลาสเดิม ทำให้เกิดการแตกสายพันธ์ของออบเจ็กต์ เป็นออบเจ็กต์ใหม่ที่มีคุณสมบัติของออบเจ็กต์เดิมซ่อนอยู่ไม่มากก็น้อย

ภาพที่ 5.6



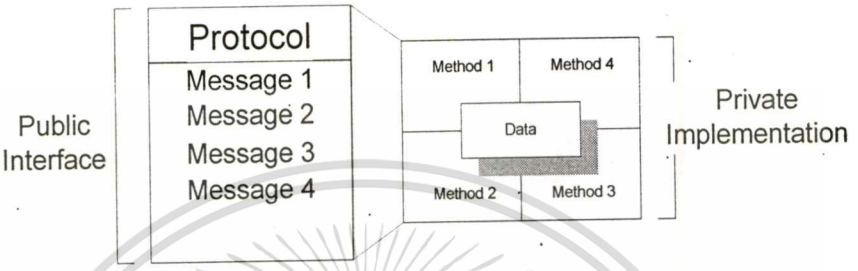
แสดงการสืบทอดความสัมพันธ์ของคลาสแม่กับคลาสลูก

■ การสืบทอดคุณสมบัติของคลาสการขยายความของคลาสจะนำเสนอโดยกลุ่มของ อินเฮียริแทนซ์ที่มีคุณสมบัติเดียวกัน นั่นคือความสัมพันธ์ของคลาสแม่(Superclass)และคลาสลูก (Subclass) จะถูกนำเสนอออกมา โดยผ่านทางอินเฮียริแทนซ์

■ แอททริบิวต์ของคลาสลูกจะสืบทอดแอททริบิวต์ต่างๆ จากคลาสแม่โดยตรงนั่นคือถ้ามีการเปลี่ยนแปลงใดๆ ที่คลาสแม่แล้วคลาสลูกจะได้รับการเปลี่ยนแปลงนั้นโดยตรง

5.2.6 โปรโตคอล (Protocol) คือภาพแบบการสื่อสารข้อมูลระหว่างออบเจ็กต์ โดยลักษณะของข้อมูลภายในออบเจ็กต์ สามารถได้แบ่งแยกเป็น 2 ส่วนคือ Public และ Private

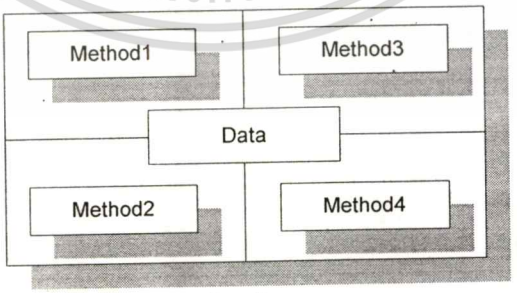
ภาพที่ 5.7



แสดงการสื่อสารข้อมูลระหว่างออบเจ็กต์

5.2.7 เอ็นแคปซูเลชัน (Encapsulation) คือการรวมกันทางโครงสร้างของข้อมูลกับฟังก์ชันที่เกี่ยวข้อง เกิดเป็นวัตถุใหม่ที่สามารถทำการซ่อนข้อมูลจากภายนอกได้ ผลจากการนำโครงสร้าง และฟังก์ชันที่ใช้งานข้อมูลนั้นมารวมกันสร้างความสัมพันธ์ให้กันและกัน จะทำให้ข้อมูลมีความมั่นคงขึ้น ผลก็คือไม่มีการเข้าถึงข้อมูลโดยตรงอีกต่อไป นั่นคือเป็นการทำ Data hiding ข้อดีของการทำ เอ็นแคปซูเลชันคือ เราสามารถแบ่งระดับของการเข้าถึงข้อมูลได้ด้วยตัวเมทธอดของออบเจ็กต์เอง

ภาพที่ 5.8



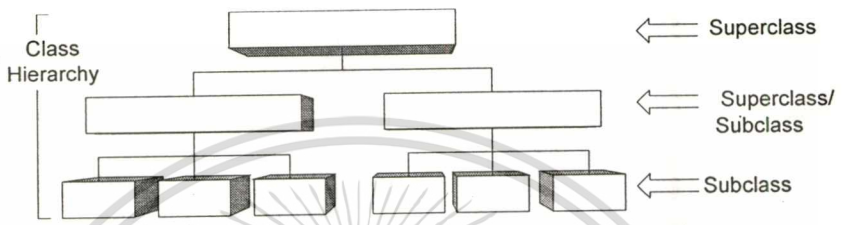
แสดง Object ประกอบด้วยข้อมูลและ Methods

5.2.8 โพลีมอร์ฟิซึม (Polymorphism)คือการที่คลาสหนึ่งๆ สามารถทำงานได้หลายภาพแบบ ขึ้นกับสภาพแวดล้อมหรือสถานะการในขณะนั้น โพลีมอร์ฟิซึมสามารถที่จะกำหนดการดำเนินการใหม่ให้กับตัวดำเนินการหรือแม้แต่ฟังก์ชันได้เรียกว่าการทำโอเวอร์โหลดดิ่'(Overloading) คือการเรียกใช้ฟังก์ชันหรือตัวดำเนินการเดิมให้ไปทำงานอีกอย่างหนึ่ง

5.2.9 รีเลชันชิพ(Relationship) คือความสัมพันธ์ระหว่างคลาสที่มีกับคลาสต่างๆ ซึ่งรีเลชันชิพสามารถแบ่งออกได้เป็น 2 ชนิดหลักคือความสัมพันธ์เชิงลำดับขั้น(Hierarchical Relationship) และความสัมพันธ์แบบไม่มีกฎเกณฑ์(Non - Hierarchical Relationship)

■ Hierarchical Relationship คือความสัมพันธ์ที่เชื่อมระหว่างคลาส 2 คลาส ในภาพแบบเชิงลำดับขั้น

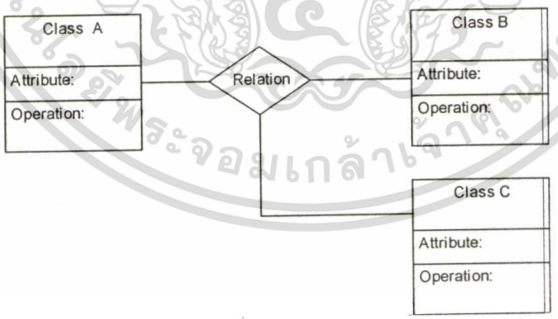
ภาพที่ 5.9



แสดงความสัมพันธ์ของคลาสแบบเชิงลำดับขั้น

■ Non-Hierarchical Relationship คือความสัมพันธ์ที่ไม่สามารถเสนอออกมา ในภาพของความสัมพันธ์เชิงลำดับขึ้นได้ แต่จะแสดงได้อย่างชัดเจนในรูปแบบของความสัมพันธ์ระหว่างออบเจ็กต์ คือความสัมพันธ์ในภาพแบบของ 1-1, 1-M, M-1 และ M-M

ภาพที่ 5.10



ความสัมพันธ์ระหว่างวัตถุของคลาสแบบไม่เป็นลำดับขั้น

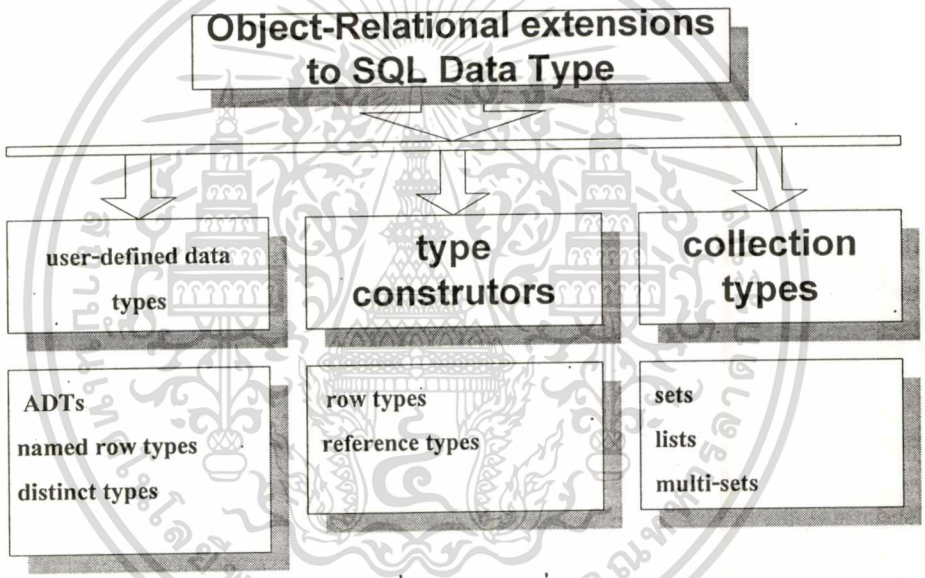
5.3 การพัฒนา ภาษา SQL3

ในการประชุมเกี่ยวกับมาตรฐานของ ANSI(X3H2)และ ISO(ISO/IEC JTC1/SC21/WG3) SQL ได้มีการกำหนดมาตรฐานของ SQL เพิ่มเติม โดยให้มีการสนับสนุนการจัดการฐานข้อมูลเกี่ยวกับออบเจ็กต์โอเรียนเต็ด ซึ่งมาตรฐานของภาษา SQL ในปัจจุบันคือ SQL3 ยังคงสนับสนุนคุณสมบัติของ SQL2 โดยมีการขยายเพิ่มในส่วนของออบเจ็กต์เข้ามาและขยายโครงสร้างชนิดของข้อ

มูลให้มีชนิดใหม่และมีความสามารถสูงขึ้น รวมทั้งเพิ่มเติมขยายโครงสร้างของตารางด้วยแต่ ทำให้ SQL3 มีการพัฒนาและมีความสามารถสูงขึ้น ซึ่งรายละเอียดของโครงสร้างของข้อมูลที่มีการขยายเพิ่มเติมของ SQL3 คือ

- **Large Objects** (LOBs) มีการยอมให้จัดเก็บข้อมูลชนิดใหม่และมีขนาดใหญ่ได้
- **User Defined Types** (UDTs) สามารถสร้างชนิดของข้อมูลขึ้นใหม่ได้
- **User Defined Functions** (UDFs) สามารถสร้างฟังก์ชันใหม่ได้
- **SQL extensions** มีการขยายความสามารถของ SQL ให้มีความสามารถเพิ่มขึ้น
- **Triggers / Constraints** มีการตรวจสอบข้อมูลและความถูกต้องของข้อมูล

ภาพที่ 5.11



แสดงโครงสร้างของข้อมูลที่มีการขยายเพิ่มเติมใน SQL3

5.3.1 เมทธอดหรือโอเปอร์เรชัน (Operations) หมายถึงการจัดการกับข้อมูล เป็นส่วนที่ถูกใช้อ้างถึงในการดึงข้อมูลเช่น (SELECT, INSERT, UPDATE, DELETE) รวมทั้งครอบคลุมถึงการกำหนดนิยามฟังก์ชันคุณสมบัติและรูทีน (Routines) ของแอปสแตรกดาต้าไทพ์(ADTs : Abstract Data Type) ซึ่งทั้งสองอย่างนี้เป็นสิ่งที่จำเป็นสำหรับ ADTs คือใช้ในการกำหนดฟังก์ชันสำหรับการกำหนดไทพ์ ซึ่งการกำหนดฟังก์ชันจะเป็นตัวจัดการกับข้อมูลบน ADTs และมีการส่งกลับเพียงหนึ่งค่าตามที่มีการกำหนดในขนิดของข้อมูล ฟังก์ชันทั้งสองจะเป็นจะเป็นฟังก์ชันของ SQL ที่สมบูรณ์ในการกำหนดภาพแบบตามคำนิยามโครงร่างของ SQL หรือ ฟังก์ชันภายนอกที่ถูกกำหนดโดยภาษาโปรแกรมมาตรฐาน

ทริกเกอร์ (Trigger)เป็นการกระตุ้นของเหตุการณ์หนึ่งที่มีผลต่ออีกเหตุการณ์หนึ่งเมื่อเหตุการณ์ที่กระตุ้นนั้นเป็นจริง เช่น

```
CREATE TRIGGER update_balance
   BEFORE INSERT ON account_history          /* event */
   REFERENCING NEW AS ta
   FOR EACH ROW
   WHEN( ta.TA_type = 'W' )                  /* condition */
   UPDATE accounts                           /* action */
     SET balance = balance - ta.amount
     WHERE account_# = ta.amount_#
```

ทริกเกอร์สามารถถูกนำมาใช้ได้ตามความต้องการเช่น กำหนดความถูกต้องของข้อมูลทางด้านอินพุท, การอ่านข้อมูลจากตารางอื่นที่มีลักษณะของข้อมูลที่แตกต่างกัน

5.3.2 รีเลชันชิพ (Relationships) หมายถึงความสัมพันธ์ของตารางโดยทั่วๆไปสามารถถูกใช้ในภาพแบบของ n-ary เหมือน SQL2 ซึ่งการอ้างถึงและการจัดการเกี่ยวกับความถูกต้องสามารถที่จะถูกกำหนดบนตารางได้ โดยมีส่วนที่เพิ่มเติมเข้ามาคือ ADTs ซึ่งเป็นภาพแบบของความสัมพันธ์ใน SQL3 ตัวอย่างเช่น

```
CREATE TYPE Person
        ( name VARCHAR,
        birthdate  DATE,
        works_for company
        ……..);
```

ในตัวอย่างนี้ค่าของแอททิบิวต์ works_for จะเป็น OID ของบริษัทซึ่งจะแสดงความสัมพันธ์ระหว่างข้อมูลของพนักงานและข้อมูลของบริษัท(ในหนึ่งบริษัทสามารถที่จะมีข้อมูลของพนักงานหลายคน)ซึ่งการอ้างถึงข้อมูลของออบเจ็กต์ใน SQL3 สามารถที่จะกำหนดในภาพแบบของ MULTISET(..),LIST(..),และ SET(..)

5.3.3 แอททิบิวต์ (Attributes) คือรายละเอียดของข้อมูลที่อยู่ในคลาสที่เป็นตัวบ่งบอกให้เราทราบว่าคลาสนั้นมีรายละเอียดอะไรบ้าง เช่น

```
CREATE TABLE employee
        ( name CHAR(40),
        address  ROW (street CHAR(30),
                city CHAR(20),
                zip ROW(original CHAR(5),
                        plus4 CHAR(4))));
```

INSERT INTO employees

VALUES('John Doe',('2225 Coral Drive',;San Joes',('95124','2347')))

5.3.4 โพลีมอร์ฟิซึม(Polymorpism)หมายถึงโปรแกรมที่มีชื่อเหมือนกันแต่มีความแตกต่างกันทางด้านการทำงานของโปรแกรม หรือการทำงานของโปรแกรมในภาพแบบที่มีการเรียกใช้งานบ่อยๆเช่นการยินยอมให้มีการกำหนดซับไทพ์ขึ้นมาใหม่ โดยมีเมทธอดที่มีความสัมพันธ์กับซุปเปอร์ไทพ์ แต่เราจะมีการกำหนดพารามิเตอร์ที่ใช้ในแต่ละความหมายให้มีความแตกต่างกัน

5.3.5 อินแคปซูเลชัน (Encapsulation) ในแต่ละส่วนของ ADTs จะมีระดับของ เอ็นแคปซูเลชัน หลายระดับเช่น PUBLIC, PRIVATE หรือ PROTECTED. องค์ประกอบของ PUBLIC จะอยู่ในภาพของการเชื่อมต่อของ ADTs ซึ่งในส่วนนี้ผู้ใช้สามารถที่จะมองเห็นได้ ทุกคน ในส่วนของ PRIVATE นั้น จะอยู่ในส่วนของอินแคปซูเลชันทั้งหมด และจะยอมให้มีการกำหนดการใช้ในส่วนของ ADTs เท่านั้น ในส่วนของ PROTECTED เป็นรายละเอียดของ ADTs ซึ่งจะยอมให้มีการมองเห็นเฉพาะภายใน ADTs เอง และภายในจะมีการกำหนด ซับไทพ์ของ ADTs ซึ่ง SQL3 จะมีส่วนที่ใช้ในการสนับสนุนอินแคปซูเลชันสำหรับตาราง การขยายตารางหรือการมองตารางจะมีการกำหนดให้ใช้อินแคปซูเลชัน

5.3.6 แอกรีแก็ต(Aggregates)ของSQL3จะมีภาพแบบของ ROWs(ROW TYPE)เป็นรายละเอียดของโครงสร้างของข้อมูล ข้อมูลแบบโรไทพ์สามารถจะถูกกำหนดให้อยู่ในตารางได้ และกำหนดชนิดของข้อมูลได้หลายอย่างเช่นในภาพแบบของ SET (<type>), MULTISET(<type >),LIST(<type>) และใช้ชนิดของข้อมูลได้หลายอย่างเช่น SET(INTEGER), SET(LIST (INTEGER))

5.3.7 ไทพ์และรีเฟอร์เร็นซ์ (Type and References) SQL3 จะสนับสนุนชนิดของ ADT และโรไทพ์(ROWs : Row Type)ซึ่งชนิดของโรไทพ์ก็เหมือนกับตาราง แต่บางครั้งก็จะมีภาพแบบคล้ายกับออบเจ็กต์และSQL3 ในปัจจุบันถูกกำหนดให้เป็นพื้นฐานของการใช้งานระบบฐานข้อมูลที่มีส่วนจำเป็นในการสนับสนุนออบเจ็กต์โอเรียนเต็ด ซึ่งการสนับสนุนของ SQL3 มีแนวความคิดที่สำคัญคือ

5.3.7.1 โรว์ชนิดที่มีชื่อ(ROWs : Named Row Types)เป็น ROWs ที่มีการกำหนดชื่อให้กับ ROWs ซึ่งในระบบฐานข้อมูลปัจจุบัน ROWs บนตารางเป็นสิ่งหนึ่งที่มีความสำคัญมากของออบเจ็กต์ เมื่อมีการออกแบบระบบฐานข้อมูลจึงมีการนำ ROWs มาใช้เป็นพื้นฐานซึ่งมีการออกแบบที่เป็นแบบมาตรฐานคือ

1. การวิเคราะห์เอ็นติตี้ที่เหมือนกัน
2. การออกแบบให้อยู่ในภาพโครงสร้างของ ER ไดอะแกรม
3. เปลี่ยนโครงสร้างของ ER ไปเป็นโครงสร้างของรีเลชันแนล

ดังนั้นการพัฒนาออบเจ็กต์ในเชิงธุรกิจจะจบลงด้วยการออกแบบใช้ROWsในตารางตัวอย่างเช่น ไทพ์และรีเฟอร์เร็นซ์ (type and references:the big picture)

ในโครงสร้างของไทพ์ที่มีการขยายเพิ่มเติมเข้ามาเพื่อสนับสนุนการใช้งานของภาษา SQL3 จะประกอบไปด้วย

1. Named Row Types
2. Row Type
3. Reference Type

5.3.7.2 โรว์ชนิดที่ไม่มีชื่อ(UnNamed Row Types or Row Types)ในฐานข้อมูล ของ SQL3 โรว์(ROWs)ในตารางเป็นสิ่งที่มีความสำคัญมาก ที่กำหนดให้อยู่ในรูปแบบของออบเจ็กต์ได้ซึ่งมีรายละเอียดในการออกแบบดังนี้

1. วิเคราะห์ระบบงานที่ทำหาเอ็นตีตี้ที่เหมาะสมกับลักษณ์ของงานแล้วนำมาเชื่อมต่อกันโดยใส่ความสัมพันธ์ระหว่างเอ็นติตี้เข้าไปด้วย
2. ออกแบบการขั้นต่อมาโดยการใช้ ER ไดอะแกรม
3. เปลี่ยนรูปบบของ ER ไปเป็น ตาราง

ซึ่งจากการทำงานของ ROWs ที่ได้มาสามารถที่จะมองให้อยู่ในรูปภาพของออบเจ็กต์ ซึ่งการกำหนดบทบาทและหน้าที่ของ ROWs ทำได้โดยการใส่คอนเทรนส์และทริกเกอร์ให้กับ ROWs นั้น ผลของการออกแบบโดยใช้ ROWs ที่เกิดขึ้นในตารางเช่น

```
CREATE TABLE Customer (
        ssno        INT,
        name        VARCHAR(50),
        address     VARCHAR(200),
        PRIMARY KEY ssno );

CREATE TABLE Account (
        acctno      INT,
        custno      INT,
        type        CHAR(1),
        opened      DATE,
        rate        DOUBLE PRECISION,
        PRIMARY KEY acctno,
        FOREIGN KEY ( custno ) REFERENCE Customer );
```

จะเห็นได้ว่า Rows เป็นพื้นฐานของออบเจ็กต์ และมีความสำคัญมากในการออกแบบฐานข้อมูลในปัจจุบัน ซึ่ง Rows นั้นสามารถที่จะจัดการรวมทั้งใช้คำสั่งของ SQL3 ได้ด้วย

```
CREATE ROW TYPE  Address_t (
        street      VARCHAR(120),
        city        VARCHAR(68),
        state       CHAR(12),
        zip         CHAR(10), );

CREATE ROW TYPE Customer_t(
        Customer_id     REF     (Customer_t),
        ssno        INT,
        name        VARCHAR(50),
        address     address_t );

CREATE TABLE Customer OF Customer_t(
        PRIMARY         KEY     ssno,
        VALUES FOR Customer_id          ARE  SYSTEM GENERATED  );

CREATE ROW TYPE Account_t (
        acctno      INT,
        cust        REF(Customer_t),
        type        CHAR(1),
        opened      DATE,
        balance     DOUBLE  PRECISTION );

CREATE TABLE Account OF Account_t (
        PRIMARY         KEY     acctno,
        SCOPE   FOR     cust    IS Customer );
```

5.3.7.3 การอ้างอิงข้อมูล (Reference Types)เป็นสิ่งที่มีความสำคัญมากอย่างหนึ่งของระบบฐานข้อมูลในปัจจุบัน ในฐานข้อมูลแบบรีเลชันจะมีการอ้างอิงข้อมูลไปยังเทเบิ้ลต่างๆ โดยผ่านทาง Foreign key และ Primary key แต่ในกรณีของระบบฐานข้อมูลเชิงวัตถุสัมพันธ์ จะพิจารณาใน

ภาพแบบการอ้างอิงทั้งของ Row และ Object ในกรณีของ Row เราจะมี Reference Types เป็นสิ่งที่สนับสนุนในการอ้างถึง Row ต่างๆโดยจะไม่ใช้ Foreign key เหมือนในแบบรีเลชัน ทำให้การออกแบบทำได้สะดวกรวดเร็วและมีความยุ่งยากน้อย ตัวอย่างของ Reference type คือ

```
SELECT a.cust->name
    FROM account a
    WHILE a.cust->address..city = "Hollywood"
    AND a.balance > 100000;

CREATE FUNCTION debit (REF(account_t), DOUBLE PRECISION )
        RETURN DOUBLE PREC ISION ,,, ;
```

5.3.8. แอปสแตรกดาต้าไทพ์(ADTs and References)ADTs มีส่วนที่สำคัญมากที่ทำให้มีการยอมรับในคุณสมบัติของ SQL3 ซึ่ง ADTs เป็นพื้นฐานของการพัฒนาในชนิดของคอลัมน์ชนิดใหม่ ซึ่งเป็นข้อมูลชนิดใหม่ เช่น TEXT , IMAGE, SPATIAL TYPE และอื่นๆ ซึ่งมีการพัฒนาให้มีการใช้มัลติมีเดีย (Multimedia) ในระบบฐานข้อมูล และ ADTs เป็นสิ่งที่ได้รับการพัฒนาไปในแนวทางนี้ ตัวอย่างของ ADTs คือ

```
CREATE OBJECT TYPE Person_type
        ( name VARCHAR      NOT NULL ,
          sex CONSTANT      CHAR(1),
          age UPDATETABLE   VIRTUAL
              GET   WITH   age   SET WITH SET_age,
        PRIVATE
              biethdate DATE CHECK ( birthdate <> DATE '1992-01-01' ),
        PUBLIC
              EQUALS        DEFUALT,
              LESS          THAN          NONE,
        ACTOR FUNCTION age ( : P person_type )RETURNS REAL,
          RETURN < code to calculate the age >
        END FUNCTION,
```

```
        ACTOR FUNCTION set_age ( :P person_type ) RETURNS person_type
                < code to update the biethdate >
                RETURN :P
        END FUNCTION,
        DESTRUCTOR FUNCTION remove_person ( : P person_type )
                RETURNS person_type
        < various cleanup action >
        DESTROY  :P;
        RETURN   :P;
        END FUNCTION;
        );
```

ส่วนตัวอย่างของ ROW TYPE มีดังนี้

```
CREATE DOMAIN us_address ROW
        ( street CHAR(30),
        city CHAR(20),
        zip  ROW (original CHAR(20),plus4 CHAR(4))  );
CREATE TABLE employees
        ( last_name  CHAR(20),
        first_name  CHAR(20),
        age INTEGER,
        address us_address  );
```

เบื้องหลังการเพิ่มเติมในส่วนของออบเจ็กต์ นั่นคือเป็นการเพิ่มเติมโครงสร้างของไทพ์ที่ถูกกำหนดโดย SQL3 ซึ่งผู้ใช้สามารถนำมาใช้ได้ ไทพ์ชนิดนี้อาจจะถูกนำมาใช้ในโครงสร้างของไทพ์ในลักษณะเดียวกัน ตัวอย่างเช่น Columns ในตารางของรีเลชันอาจจะกำหนดค่าที่ใช้ให้เหมือนกับของ User Defined Types

ภาพแบบของ User Defined Types ใน SQL3 เป็นรูปแบบที่ง่าย

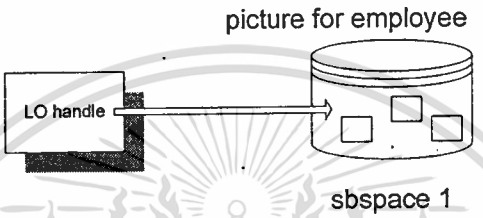
```
CREATE DISTINCT TYPE us_dolla as decimal(9,2)
CREATE DISTINCT TYPE cannadian_dolla as decimal(9,2)
```

5.3.9 การเก็บข้อมูลขนาดใหญ่ (Large BLOBs and CLOBs) หมายถึงชนิดของข้อมูลที่ใช้ในการเก็บข้อมูลขนาดใหญ่ เช่น ภาพ , เสียง, หนัง ฯลฯ ซึ่งแบ่งออกเป็น 2 ชนิด คือ

BLOB(Binary Large Object) จะใช้สำหรับการเก็บชนิดของข้อมูลเป็นแบบไบนารี่หรือข้อมูลแบบกราฟฟิก

CLOB(Character Large Object) จะใช้สำหรับการเก็บชนิดของข้อมูลเป็นแบบแคแรคเตอร์หรือข้อมูลที่มีขนาดใหญ่

ภาพที่ 5.12



แสดงการเก็บข้อมูลแบบ BLOB ที่เป็นภาพลงในฐานข้อมูล

ตัวอย่างของการใช้ข้อมูล BLOB และ CLOB คือ

```
CREATE TABLE EMPLOYEE (
    ID    INTEGER,
    NAME        VARCHAR(30),
    SALARY      US_DALLAR,
    ..... .
    RESUME      CLOB(75K),
    SIGNATURE BLOB(1M),
    PICTURE     BLOB(12M) );
```

ในการจัดการข้อมูลชนิด LOBs จะมีเมทธอดที่ใช้ในการจัดการข้อมูลคือ

Greate Than and Less Than operation

Primary, unique, and foreign keys

GROUP BY and ORDER BY

UNION operator

และมีเมทธอดในการสนับสนุนข้อมูลชนิด LOBs คือ

Retrive value (or Partial value)

Replace value

LIKE predicate

Concatenation

SUBSTRING, POSITION, and LENGTH function

UNION ALL operator

5.3.10 ข้อมูลชนิดพิเศษ(Distinct Types) คือรูปแบบพื้นฐานของผู้ใช้ที่สามารถกำหนดชนิดของข้อมูล(User Defined Types)ขึ้นมาใช้ใหม่ได้ตามความต้องการ

```
CREATE DISTINCT TYPE US_DOLLAR AS  DECIMAL(9,2)

CREATE DISTINCT TYPE CDN_DOLLAR AS DECIMAL (9,2)
CREATE DISTINCT TYPE US_DOLLAR AS DECIMAL (9,2)

SELECT CUSTNO, CONTRACT_NO
        FROM CDN_SALES CDN, US_SALES USA
        WHERE CDN.CONTRACT_NO = USA.CONTACT_NO
        and CDN.TOTAL > USA.TOTAL FAILS!!! cannot compare CND and US

SELECT CUSTNO, CONTRACT_NO
        FROM CDN_SALES CDN, US_SALES USA
        WHERE CDN.CONTRACT_NO = USA.CONTACT_NO
        and CDN.TOTAL > CDN_DOLLAR (USA.TOTAL)

CREATE TABLE CDN_SALES
        (CUSTNO INTEGER,
        CONTRACT_NO INTEGER,
        TOTAL CDN_DOLLAR)
```

```
CREATE TABLE USA_SALES
        (CUSTNO INTEGER,
        CONTRACT_NO INTEGER,
        TOTAL USA_DOLLAR)
```

5.3.11 Abstract Data Types หมายถึงชนิดของข้อมูลที่ผู้ใช้สามารถกำหนดขึ้นมาตามความเหมาะสม พร้อมทั้งกำหนดคุณสมบัติตามที่ต้องการและเป็นเอ็นแคปซูเลชันนอกจากนี้ยังสนับสนุนข้อมูลชนิดซับซ้อน

```
CREATE TYPE note
        (sender CHAR (20),
        receiver CHAR (20),
        s_date DATE,
        contents BLOB (1M),
FUNCTION subject (note) RETURNS VARCHAR (256))

CREATE TYPE Address
        ( street char (30),
        city char (20),
        state char (2),
        zip integer);

CREATE DISTINCT TYPE bitmap AS BLOB;

CREATE TYPE Real_estate
        ( rooms INTEGER,
        size DECIMAL(8,2),
        location address,
        text_description VARCHAR(1024),
        front_view_image bitmap,
        document doc);
```

```
CREATE TPYE note (
    sender      CHAR (20),
    receive     CHAR (20),
    s_date      DATE,
    contents    BLOB  (1M)
    FUNCTION        subject (note)  RETURNS  VARCHAR(256) );
```

นอกจากนี้ ADTs ยังสนับสนุนโครงสร้างข้อมูลแบบข้อมูลชนิดซับซ้อน

```
CREATE TYPE  Address (
    street      CHAR  (30),
    city            CHAR  (20),
    state       CHAR  (2),
    zip         integer );
CREATE  DISTINCT  TYPE  bitmap  AS  BLOB;
CREATE  TYPE  real_estate (
        roomINTEGER,
        size  DECIMAL(8,2),
        location      address,
        text_description      VARCHAR  (1024),
        front_view_image      bitmap,
        document      doc );
```

และ ADTs จะเป็นเอ็นแคปซูเลชันที่จะให้คุณลักษณะของรายละเอียดของข้อมูลและวิธีการของออบเจ็กต์ที่สมบูรณ์นั่นคือ

ADTs สามารถเข้าถึงได้โดยฟังก์ชัน

ADTs จะเชื่อมต่อโดยผ่านฟังก์ชัน

การจัดการข้อมูลใน ADTs นั้นสามารถกระทำได้โดยผ่านทางฟังก์ชันที่มีการกำหนดอยู่ใน ADTs เช่น

```
INSERT INTO real_estate_info
        VALUES ( US_DALLAR (300000),
                'John Doe',
```

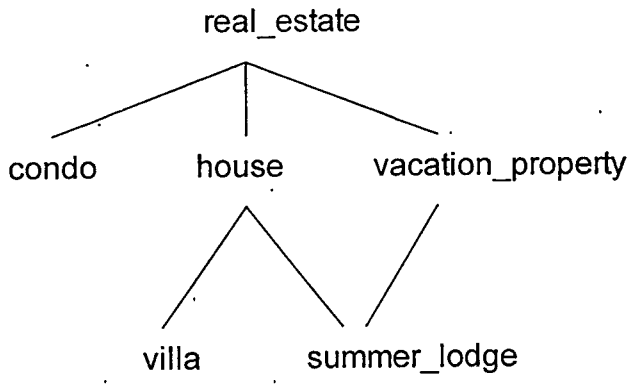
```
real_estate (4, 2000, address ( '2225  Coral Drive',
                                'San  Jose',
                                'CA',
                                95125 )) );
```

```
UPDATE real_estate_info
    SET price  =  US_DALLAR (0.9 x amount (price))
    WHERE  property..location..city = 'San Jose';
```

```
SELECT D_mark (price), owner, property..address
    FROM real_estate_info
    WHERE size (property) > 2000
    AND size (property)..room > 5
    AND contains (property..text_description, 'excellent school distinct')
    AND contains (property..front_view_image, 'tree');
```

ADTs สามารถที่จะสร้าง Subtype ได้ตั้งแต่หนึ่งหรือมากกว่า ADTs ซึ่งจะสืบทอดโครงสร้างและคุณสมบัติมาจาก Supertypes นอกจากนั้นยังสนับสนุน Multiple Inheritance เช่น

```
CREATE TYPE  real_estate…
CREATE TYPE  condo UNDER real_estate…
CREATE TYPE  house  UNDER real_estate…
CREATE TYPE  villa  UNDER house
CREATE TYPE  vacation_property UNDER real_estate…
CREATE TYPE summer_lodge UNDER vacation_property, house…
```



ข้อมูลของ ADTs จะสืบทอดแอททริบิวต์ มาจาก Sypertypes ได้ทั้งหมด รวมทั้งสามารถที่จะออกแบบแอททริบิวต์และฟังก์ชันเพิ่มเติมขึ้นมาใหม่ได้

```
CREATE TYPE Real_estate (
    room            INTEGER,
    size            DECIMAL(8,2),
    location        address,
    text_description    VARCHAR (1024),
    front_view_image    bitmap,
    document    doc );

CREATE TYPE House UNDER Real_estate
    (stories            INTEGER,
    total_area    DECIMAL(10,2));
```

ซึ่งข้อมูลของ House จะมี แอททริบิวต์ ของ room, size, location, address, text_description, front_view_image, document, stories และ total_data

Functions บน แอปสแต็รก ดาต้าไทพ์ สามารถที่จะสร้างได้หลายภาพแบบ สามารถสร้าง function ที่มีชื่อเหมือนกันได้ ตัวอย่างคือ

features (real_estate) → VARCHAR(512)

features (real_estate,VARCHAR(512)) → VARCHAR(512)

features (house) → VARCHAR(384)

โดยเราสามารถที่จะสร้าง Function และ Procedures บน ADTs และกำหนด ADTs ได้ทั้งภายในและภายนอก

การสร้าง Function ภายใน ADTs สามารถทำได้คือ

```
CREATE TYPE t_employee (
    PUBLIC   name          char (20),
             B_address     t_address,
             manager       t_employee,
             hiredate      DATE,
    PUBLIC   base_salary   decimal(7,2),
    PUBLIC   commission    decimal(7,2),
    PUBLIC   FUNCTION  salary (p t_employee)  RETURNS DECIMAL
             <code to decimal salary> );
```

การสร้าง Function ภายนอก ADTs สามารถทำได้คือ

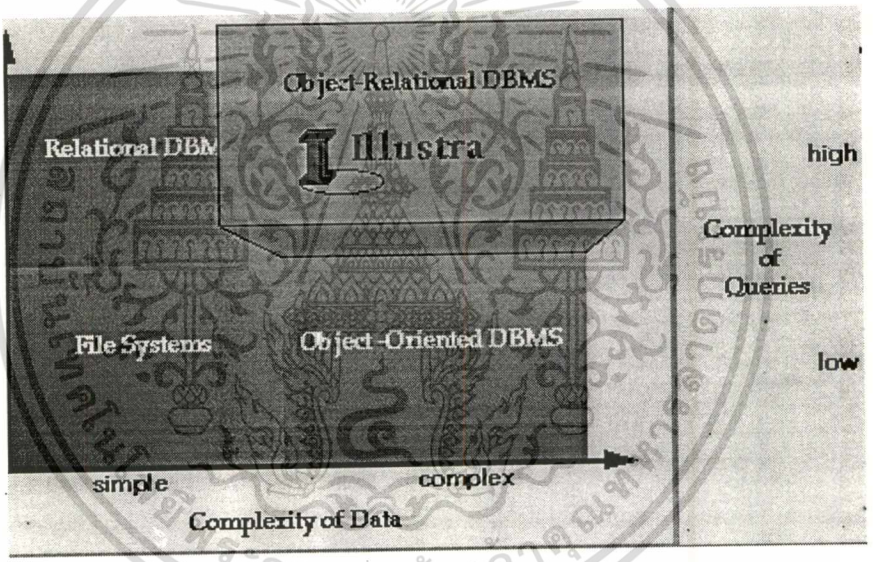
```
CREATE TYPE t_employee(
    PUBLIC   name          char (20),
             b_address             t_address,
             manager       t_employee,
             hiredate      DATE,
    PUBLIC   base_salary   decimal(7,2),
    PUBLIC   commission    decimal(7,2) );

CREATE FUNCTION  salary (p t_employee)  RETURNS DECIMAL
         <code to decimal salary> );
```

บทที่ 6
ระบบฐานข้อมูลของอินฟอร์มิก

6.1 บทนำ

ระบบฐานข้อมูลอินฟอร์มิกหรือเรียกว่า **อินฟอร์มิก ยูนิเวอร์แซล เซิร์บเวอร์**(IUS :Informix Universal Server)[6],[9] เป็นระบบฐานข้อมูลแบบ ORDB (Object Relational Database System) ที่เป็นระบบฐานข้อมูลเชิงวัตถุสัมพันธ์ มีคุณสมบัติตามมาตรฐานของ SQL3 โดยมาตรฐาน SQL3 นั้นเป็นมาตรฐานใหม่ที่ได้รับการยอมรับในระบบฐานข้อมูลและมีการนำมาพัฒนาใช้งานอย่างแพร่หลายในปัจจุบัน ได้อย่างมีประสิทธิภาพ

ภาพที่ 6.1



แสดงระบบฐานข้อมูลของ อินฟอร์มิก ที่เป็นแบบ ORDB

6.2 ลักษณะทั่วๆไปของฐานข้อมูลอินฟอร์มิก ยูนิเวอร์แซล เซิร์บเวอร์

ระบบฐานข้อมูลของ IUS เป็นระบบจัดการฐานข้อมูลที่ผสมผสานความสามารถที่ดีที่สุดให้มารวมกันอยู่ในระบบเดียวของสถาปัตยกรรมแบบ Object Relational Database Management Systems (ORDBMS) โดยการออกแบบระบบจัดการฐานข้อมูลของ IUS นี้ ได้ออกแบบเพื่อช่วยให้ระบบฐานข้อมูลเดิมเปลี่ยนไปเป็นระบบฐานข้อมูลแบบใหม่ ที่มีประสิทธิภาพสูงสุดอย่างแท้จริง

ระบบจัดการฐานข้อมูลนี้ได้รวมเอาระบบจัดการฐานข้อมูลที่มีประสิทธิภาพและสามารถขยายระบบได้อย่างต่อเนื่อง(Scalability)ของสถาปัตยกรรมของอินฟอร์มิกในแบบDynamic Scalable Architecture (DSA) ซึ่งเป็นระบบจัดการฐานข้อมูลแบบขนานที่มีประสิทธิภาพสูงที่มีใช้

ในปัจจุบัน ผนวกเข้ากับการจัดการข้อมูล ที่ขยายขีดความสามารถในการจัดการข้อมูลทุกแบบเข้ารวมกันเป็นระบบจัดการฐานข้อมูลแบบใหม่ เป็นนวัตกรรมทางเทคโนโลยีคลื่นลูกใหม่ของวงการระบบจัดการฐานข้อมูลที่มีชื่อว่า IUS

การออกแบบดังที่กล่าวข้างต้น ทำให้ IUS จัดเป็นระบบจัดการฐานข้อมูลที่เหมาะสมสำหรับองค์กรต่างๆ ที่ต้องการระบบจัดการฐานข้อมูลที่มีความเชื่อถือได้มีประสิทธิภาพสูง สามารถขยายระบบให้มีประสิทธิภาพมากขึ้นตามการขยายงานที่เติบโตขึ้นและจัดการฐานข้อมูลได้ทุกรูปแบบภายใต้ระบบจัดการฐานข้อมูลเดียว ซึ่งจะตอบสนองต่อจินตนาการที่กว้างไกลในการใช้ข้อมูลทุกรูปแบบในการบริหารอย่างแท้จริง ในการขยายขีดความสามารถในการจัดการการค้นหาข้อมูลที่ซับซ้อน และข้อมูลในทุกรูปแบบไม่ว่าจะเป็นภาพ เสียงเว็บ หรือเอกสารอิเล็กทรอนิกส์หรือข้อมูลแบบ 2D 3D ได้นั้น บริษัทอินฟอร์มิกได้สร้างเทคโนโลยีที่ทำงานร่วมกับ IUS ที่มีประสิทธิภาพที่มีชื่อเรียกว่า Datablade ที่จะขยายขีดความสามารถในการจัดการข้อมูลนั้นเอง

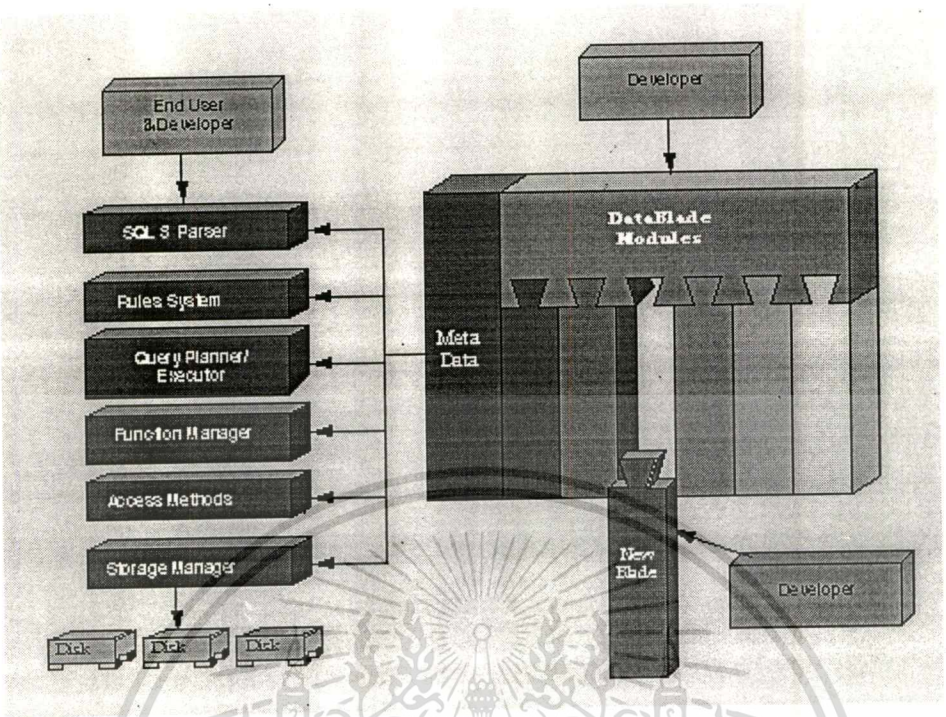
Datablade นั้นเปรียบเสมือนมีดโกนที่คมกริบที่จะเฉือนเข้าไปในเนื้อข้อมูลขององค์กรเพื่อที่จะดึงเอาข้อมูลในรูปแบบลักษณะที่ต้องการขึ้นมาได้ และสามารถจะใช้ Datablade สำหรับใช้กับข้อมูลแบบ Text Video Audio และพัฒนาร่วมกับบริษัทคู่ค้าในการสร้าง Datablade สำหรับการค้นหาแบบ 2D 3D หรือแม้กระทั่งการค้นหาหน้าของคนที่เหมือนกับหน้าของคนที่เรามีอยู่ในระบบฐานข้อมูลหรือ Datablade ที่ใช้ในการหาข้อมูลทางการเงินย้อนหลังตามเวลา (Timeseries)

ในปัจจุบันนี้มีผู้เข้าร่วมพัฒนา Datablade ตามคุณลักษณะต่างๆ ให้ผู้ใช้จะได้ประโยชน์จากการพัฒนา Datablade นี้เป็นอย่างมาก อีกทั้งระบบจัดการฐานข้อมูลแบบ IUS ยังออกแบบให้ผู้ใช้ใดๆ สามารถที่จะพัฒนา Datablade เพื่อใช้งานค้นหาข้อมูลเฉพาะของท่านเองได้อีกด้วยโดยการพัฒนานี้จะไม่มีผลต่อโครงสร้างหลักของระบบจัดการฐานข้อมูล อีกทั้งผู้ใช้ที่ต้องการค้นหาข้อมูลแบบที่ต้องการ ก็เปรียบเสมือนการเปลี่ยนมีดโกนที่ใช้งานเฉพาะงานนั้นๆ นั้นเอง และเรียกมาใช้งานได้อีกตามต้องการ

6.3 โครงสร้างระบบฐานข้อมูลของอินฟอร์มิก

โครงสร้างระบบฐานข้อมูลของอินฟอร์มิก จะมีโครงสร้างดังภาพที่ 6.2 ที่จะแสดงถึงโครงสร้างของฐานข้อมูลเป็นแบบ PLUG-INS คือ ฟังก์ชันหรือโมเดลต่างสามารถที่จะเพิ่มเติมหรือมีการขยายเพิ่มขีดความสามารถในเทคโนโลยีใหม่ได้ตลอดเวลาโดยไม่กระทบกับโมเดลอื่นๆ

ภาพที่ 6.2



แสดงถึงโครงสร้างของฐานข้อมูลเป็นแบบ PLUG-INS

6.4 จุดเด่นของโครงสร้างสถาปัตยกรรมของ IUS

6.4.1 การออกแบบมีประสิทธิภาพสูง(Performance) การออกแบบ อินฟอร์มิก-Universal Server ใช้โครงสร้างเดียวกับระบบจัดการฐานข้อมูลแบบขนานแบบ Dynamic Scalable Architecture ที่มีชื่อเสียงของอินฟอร์มิก ซึ่งออกแบบให้สามารถใช้ได้ทั้งระบบคอมพิวเตอร์แบบโปรเซสเซอร์เดียวและแบบหลายโปรเซสเซอร์ โดยมีประสิทธิภาพที่สามารถขยายได้หรือที่เรียกว่า Scalable และยังเป็นระบบที่สามารถใช้ประสิทธิภาพของฮาร์ดแวร์ได้อย่างมีประสิทธิภาพ ไม่ว่าจะเป็นการทำงานสอบถามข้อมูลแบบขนาน(Parallel Database Query) การจัดการแบบขนานต่างๆ อีกทั้งการออกแบบให้สามารถจัดการกับข้อมูลในทุกรูปแบบ นอกเหนือจากข้อมูลแบบตัวเลข,ตัวอักษร ซึ่งทำให้ IUS สามารถใช้งานกับข้อมูลหลายแบบมีประสิทธิภาพสูงและสามารถขยายได้

นอกเหนือจากนั้น IUS ยังสามารถปรับแต่งให้การทำงานของระบบ ที่บางครั้งอาจเป็น Process ที่ซับซ้อนให้สามารถจัดการกับข้อมูลที่เกี่ยวข้องอย่างใกล้ชิด โดยอย่างในส่วนของ Server ซึ่งเป็นการเพิ่มประสิทธิภาพการทำงานไปในตัวอีกด้วย และการออกแบบก็จะทำให้ง่ายต่อการพัฒนาและการบำรุงรักษา ซึ่งจะทำให้โปรแกรมเมอร์พัฒนาโปรแกรมได้ง่าย โดยที่ไม่จำเป็นต้องค้นหาข้อมูลซับซ้อนเอง และแน่นอนที่สุดก็เป็นการย่อระยะเวลาในการพัฒนาโปรแกรมที่ใช้งานอย่างมีประสิทธิภาพ อีกทั้งยังนำมาใช้ได้อีกตามที่ต้องการอีกด้วย

6.4.2 การออกแบบให้สามารถขยายได้ (Extensible) ในระบบจัดการฐานข้อมูลโดยทั่วไปจะจัดการกับข้อมูลที่เป็นตัวเลข ตัวอักษร แต่ IUS ออกแบบให้สามารถใช้งานอย่างไร้ขีดจำกัด โดยสามารถที่จะใช้งานกับข้อมูล จะเป็นประเภทใด เช่น ภาพวิดีโอ เสียง แกนเวลา สองมิติ หรือสามมิติ ภาพ ข้อความ และข้อมูลที่ใช้กับเว็บ

6.4.3 ออกแบบให้เชื่อมต่อกันได้อย่างดี (Integrated) การออกแบบที่รวบรวมของข้อดีของระบบจัดการฐานข้อมูลแบบขนานที่มีชื่อเสียงของอินฟอร์มิก คือ โครงสร้างแบบ Dynamic scalable Architecture และโครงสร้างแบบออบเจ็กต์โอเรียนเต็ดที่ทำให้ระบบสามารถขยายขีดความสามารถและนำกลับมาใช้ใหม่ได้อย่างมีประสิทธิภาพนี้ นำทั้ง 2 โครงสร้างมารวมเข้าด้วยอย่างกลมกลืน ซึ่งทั้ง 2 โครงสร้างนี้เป็นที่ยอมรับในวงการระบบจัดการฐานข้อมูล การรวมกันนี้ทำให้เป็นระบบจัดการฐานข้อมูลที่ทั้งมีประสิทธิภาพ ขยายได้(Scalable) และเพิ่มความสามารถของระบบ (Extensible) ได้เป็นอย่างดี และที่สำคัญคือในโครงสร้างใหม่นี้จะเป็นเพียงหนึ่งเดียวของ Source Code เท่านั้น การที่สามารถอยู่ในระบบเดียวกันนี้ทำให้ได้ประสิทธิภาพสูงสุด และลดความซับซ้อนของระบบจัดการฐานข้อมูลและระบบงานได้อีกด้วย ในขณะที่คงไว้ซึ่งข้อดีในแง่ของประสิทธิภาพความเชื่อถือได้อย่างสูง และความปลอดภัยซึ่งทำให้เชื่อถือได้ว่า จะเป็นระบบที่เป็นพื้นฐานของธุรกิจได้อย่างดี ประโยชน์ที่ได้รับจากการรวมกันเป็นหนึ่งเดียวที่ทำให้ IUS มีประสิทธิภาพน่าเชื่อถือและคุ้มค่าต่อการลงทุน จากการที่เป็นระบบเดียวที่การพัฒนาและการปรับปรุงระบบจะมีการดำเนินการอย่างต่อเนื่องอีกด้วย

6.4.4ออกแบบที่นวัตกรรมใหม่ (Innovation) การออกแบบ IUS ที่คำนึงถึงความเป็นระบบเปิด(Open System)ระบบที่ขยายได้อย่างไร้ขีดจำกัด (Unlimit Extensibility) และเพิ่มประสิทธิผลให้กับโปรแกรมเมอร์ ทำให้สามารถที่จะสร้างระบบงานที่เหมาะสมกับธุรกิจต่างๆ เป็นอย่างดี ซึ่งทำให้การออกแบบอิงกับระบบที่เป็นสากล(Standard)ต่างๆ อาทิมาตรฐาน SQL-3, การเชื่อมต่อกับ Netscape หรือ Microsoft Browser การเชื่อมต่อกับ Conmnectivity ที่มีจำหน่ายในท้องตลาดหรือภาษาที่ใช้เป็นต้น ที่ช่วยให้การใช้งานรวมกับเทคโนโลยี่ใหม่ๆ ไม่ว่าจะเป็น Web Technology, Data Warehousing หรือการใช้งานในระบบขนาดเล็กจนถึงขนาดใหญ่ได้อย่างมีประสิทธิภาพ

6.5 การสอบถามข้อมูลของ อินฟอร์มิก Universal Server

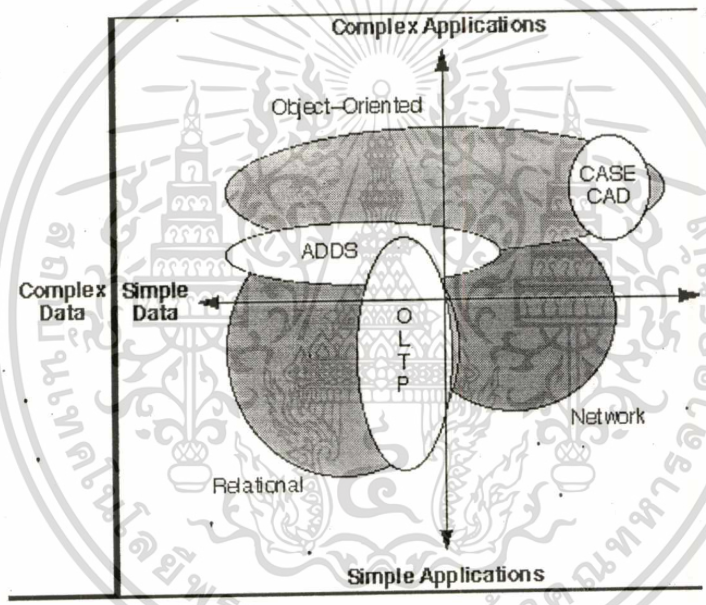
ระบบฐานข้อมูลของอินฟอร์มิก มีข้อมูลที่มีโครงสร้างเป็นแบบ Complex Data ซึ่งเราสามารถที่จะ Query Data ในภาพแบบของ Complex Data ได้ รวมทั้งสร้างข้อมูลที่เป็น Type ชนิดใหม่ได้

ภาพที่ 6.3

Query	Relational DBMS	Object-Relational DBMS
No Query	File System	Object-Oriented DBMS
	Simple Data	Complex Data

แสดงการ Query Data ของระบบฐานข้อมูลแบบต่างๆ

ภาพที่ 6.4



แสดงการพัฒนาการใช้งานบนระบบฐานข้อมูลแบบต่างๆ

6.6 การสนับสนุนมาตรฐานของ SQL3

ลักษณะการทำงานของ SQL3 จะแตกต่างไปจากการใช้งานในส่วนของ SQL92 ตรงที่จะสนับสนุนการทำงานในส่วนของ Objict-Oriented หลายส่วน เช่น Inheritance,Time Interval,Create Function,CREATE Type,Create Operators เป็นต้น

6.6.1 การใช้ข้อมูลขนาดใหญ่(SLO: SLO)

ในการกำหนดคอลัมน์(Column)ในตารางหรือการกำหนดแอททริบิวต์ใน Opaque Type ให้มีชนิดเป็น BLOB หรือ CLOB นั้นเราจะต้องมีการจองเนื้อที่สำหรับเก็บข้อมูลเหล่านี้ที่เรียกว่า

sbspace ซึ่งในการใช้งานจริงตัว Lo_handle จะชี้มาที่ sbspace นี้ ใน SQL Statement ไม่สามารถที่จะใช้ BLOB และ CLOB ใน

- Arithmetic หรือใน Boolean expression
- GROUP BY หรือ ORDER BY clause
- UNIQUE test
- Index ที่เป็นแบบ B-tree (อย่างไรก็ตามใน DataBlade สามารถที่จะสร้าง index บน CLOB Column ได้)

ใน SELECT Statement เราสามารถที่จะ

- ระบุ Null Values เป็น Default ตอนสร้างตารางโดยใช้ Default Null Clause
- ไม่อนุญาตให้เป็น Null โดยใช้ NOT Null constraint ตอนสร้างตาราง
- ใช้ IS [NOT] Null

สมมุติว่าสร้างตาราง Inmate และ fbi_list ดังนี้

```
CREATE TABLE Inmate
(
        id_num  INT,
        picture  BLOB,
        felony  CLOB
);
CREATE TABLE fbi_list
(
        id          INTEGER,
        mugshot  BLOB
) PUT mugshot IN (sbspace1);
```

คำสั่งต่อไปนี้แสดงการ INSERT ข้อมูลโดยใช้ FileToBLOB() และ FileToCLOB() function เพื่อทำแทรกข้อมูลในตาราง Inmate

```
INSERT INTO Inmate
VALUES (437,FILETOBLOB('Datafile',client),FILETOCLOB('tmp/text','server'))
```

Argument แรกของ FileToBLOB() และ FileToCLOB() เป็นการระบุ path ของ source file ที่จะทำการ copy ไปที่ BLOB หรือ CLOB ส่วน argument ที่สองเป็นการระบุว่า source file อยู่ที่ Server หรือ Client

คำสั่งต่อไปเป็นการแสดงการ UPDATE โดยใช้ LOCOPY() function ทำการ copy BLOB จาก column mugshot ในตาราง fbi_list มาที่ column picture ของตาราง Inmate

```
UPDATE Inmate(picture)
        SET picture=LOCOPY(mugshot,'fbi_list','mugshot')
        WHERE Inmate.id_num = 437 and fbi_list.id = 669;
```

Argument แรกของ function LOCOPY() ระบุถึง Column ที่จะทำการ Export SLO ส่วน Argument ที่สองระบุถึงตารางและ Argument ที่สามระบุ Column ที่จะใช้ Storage Characteristics ในการสร้าง Large Object ใหม่

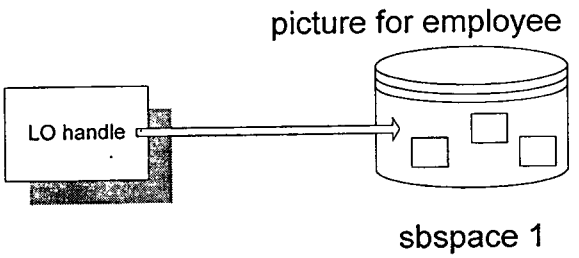
ส่วนคำสั่งต่อไปจะแสดงถึง Select statement ที่ใช้ function LOTOFILE ที่ทำการ copy ข้อมูลจาก Column felony ไปไว้ที่ fenol_322.txt ที่อยู่ที่ฝั่ง Client

```
SELECT id_num,LOTOFILE(felony,'fenol_322.txt','client)
FROM Inmate
WHERE id=322
```

6.6.2 การอ้างถึง SLO

ในการอ้างถึง SLO ไม่ว่าจะเป็นใน Column ที่มีการกำหนดชนิดข้อมูลเป็นแบบ BLOB หรือ CLOB หรือจะเป็นการอ้างใน Attribute ที่อยู่ Opaque Data Type ก็ตามจะอ้างโดยผ่านส่วนที่เรียกว่า LO-Handle ซึ่งเป็นข้อมูลแบบ Opaque Type ชนิดหนึ่งที่จะถูกเก็บใน Column หรือใน Attribute ที่มีชนิดเป็น SLO โดยที่ LO-Handle จะชี้ไปที่ Sbspace ซึ่งเป็นส่วนที่ทำการจัดเก็บ BLOB หรือ CLOB จริงๆ

ภาพที่ 6.5



แสดงการเก็บภาพลงในฐานข้อมูล

sbspace เป็น logical storage ซึ่งภายในจะประกอบด้วย 2 ส่วน

- metaData area เป็นส่วนที่เก็บข้อมูลเกี่ยว SLO ซึ่งประกอบด้วย
  - internal information ที่ใช้สำหรับ SLO optimizer ในการจัดการกับข้อมูล
  - storage characteristic ของ SLO
  - status information ของ SLO
  - user Data area เป็นส่วนที่เก็บข้อมูลของ SLO

ข้อมูลเกี่ยวกับ SLO

จะเก็บไว้ในส่วนของ metaData area ซึ่งประกอบด้วย

- storage characteristic
- status information

storage characteristic

จะประกอบด้วยข้อมูล

- Disk-storage information เป็นข้อมูลที่สำหรับจัดการกับข้อมูลบน disk เช่น allocation extent information,sizing information,location information
- Attribute information เป็นการกำหนดว่าต้องการให้มี option อะไรบ้างการเข้าถึงข้อมูล ซึ่งประกอบด้วย loggig indicator,last-access-time

ซึ่งในการระบุ storage characteristic ทำได้หลายระดับโดยสามารถกระทำได้ตอน

- สร้าง sbspace โดยใช้ onspace utility
- สร้างตารางโดยใช้ keyword PUT clause ใน CREATE TABLE statement
- สร้าง SLO โดยใช้ DataBlade API function

status information

เป็นข้อมูลที่เกี่ยวกับสถานะต่างๆของ SLO ที่จัดเก็บไว้ในส่วนของ meta Data area ของ sbspace

6.6.3 การสร้างและการเข้าถึง SLO ผ่านทาง DataBlade API

ใน อินฟอร์มิก ได้มี Interface ต่างๆ ที่ใช้กับ SLO โดยประกอบด้วย Data Structure ที่ใช้เก็บข้อมูลต่างๆ เกี่ยวกับ SLO ดังนี้

SLO Data sturcture	Data Type	Description
L O specitication	MI_LO_SPEC	เก็บ storage characteristics
LO handle	MI_LO_HAND LE	ใช้ระบุถึงตำแหน่ง

LO file descriptor	MI_LO_FD	ระบุถึงการ open SLO
LO status	MI_LO_STAT	เก็บ status ต่างๆ ของ SLO

หมายเหตุ : function และ Data Structure จะอยู่ใน milo.h ซึ่งในการใช้งานจะต้อง Include ไว้ด้วย

Operation ที่กระทำกับ SLO ประกอบด้วย

- Create; Access, Obtain status, Delete

6.6.4 แสดงคำสั่ง SQL ในการ register >

```
begin work;
create opaque type Mo_point(
        internallength = 8,
        alignment = 4 );
grant usage on type Mo_point to public;

create function Mo_pointIn (lvarchar)
       returns Mo_point
       external name "$InformixDIR/extend/MyObject.1.0/MyObject.bld(Mo_pointInput)"
       language c;

grant execute on function Mo_pointIn (lvarchar) to public;

create implicit cast  (
         lvarchar as Mo_point
         with  Mo_pointIn  );

create function Mo_pointOut (Mo_point)
       returns lvarchar
       external name "$InformixDIR/extend/MyObject.1.0/MyObject.bld(Mo_pointOutput)"
       language c;

grant execute on function Mo_pointOut (Mo_point) to public;
```

```
create cast (
        Mo_point as lvarchar
        with  Mo_pointOut  );

create function Mo_pointSend (Mo_point)
        returns sendrecv
        external name "$InformixDIR/extend/MyObject.1.0/MyObject.bld(Mo_pointSend)"
        language c;

grant execute on function Mo_pointSend (Mo_point) to public;
create cast (
        Mo_point as sendrecv
        with  Mo_pointSend  );

create function Mo_pointRecv (sendrecv)
        returns Mo_point
        external name "$InformixDIR/extend/MyObject.1.0/MyObject.bld(Mo_pointReceive)"
        language c;

grant execute on function Mo_pointRecv (sendrecv) to public;

create implicit cast (
        sendrecv as Mo_point
        with  Mo_pointRecv );

create function Compare (Mo_point,Mo_point)
        returns integer
        with ( not variant  )
        external name "$InformixDIR/extend/MyObject.1.0/MyObject.bld(Mo_pointCompare)"
        language c;

grant execute on function Compare (Mo_point,Mo_point) to public;
```

```
create function Equal (Mo_point,Mo_point)
        returns boolean
        with  ( not variant  )
        external name "$InformixDIR/extend/MyObject.1.0/MyObject.bld(Mo_pointEqual)"
        language c;

grant execute on function Equal (Mo_point,Mo_point) to public;

create function NotEqual (Mo_point,Mo_point)
        returns boolean
        with ( not variant )
        external name "$InformixDIR/extend/MyObject.1.0/MyObject.bld(Mo_pointNotEqual)"
        language c;

grant execute on function NotEqual (Mo_point,Mo_point) to public;

create opaque type Mo_circle(
          internallength = 12,
          alignment = 4  );

grant usage on type Mo_circle to public;

create function Mo_circleIn (lvarchar)
        returns Mo_circle
        external name "$InformixDIR/extend/MyObject.1.0/MyObject.bld(Mo_circleInput)"
        language c;

grant execute on function Mo_circleIn (lvarchar) to public;

create implicit cast (
          lvarchar as Mo_circle
```

```
        with  Mo_circleIn  );

create function Mo_circleOut (Mo_circle)
        returns lvarchar
        external name "$InformixDIR/extend/MyObject.1.0/MyObject.bld(Mo_circleOutput)"
        language c;
grant execute on function Mo_circleOut (Mo_circle) to public;

create cast  (
        Mo_circle as lvarchar
        with  Mo_circleOut  );

create function Mo_circleSend (Mo_circle)
        returns sendrecv
        external name "$InformixDIR/extend/MyObject.1.0/MyObject.bld(Mo_circleSend)"
        language c;

grant execute on function Mo_circleSend (Mo_circle) to public;

create cast  (
        Mo_circle as sendrecv
        with  Mo_circleSend );

create function Mo_circleRecv (sendrecv)
        returns Mo_circle
        external name "$InformixDIR/extend/MyObject.1.0/MyObject.bld(Mo_circleReceive)"
        language c;

grant execute on function Mo_circleRecv (sendrecv) to public;

create implicit cast (
        sendrecv as Mo_circle
```

```
        with  Mo_circleRecv );

create function Compare (Mo_circle,Mo_circle)
        returns integer
        with ( not variant )
        external name "$InformixDIR/extend/MyObject.1.0/MyObject.bld(Mo_circleCompare)"
        language c;

grant execute on function Compare (Mo_circle,Mo_circle) to public;

create function Equal (Mo_circle,Mo_circle)
        returns boolean
        with ( not variant  )
        external name "$InformixDIR/extend/MyObject.1.0/MyObject.bld(Mo_circleEqual)"
        language c;

grant execute on function Equal (Mo_circle,Mo_circle) to public;

create function NotEqual (Mo_circle,Mo_circle)
        returns boolean
        with (  not variant )
        external name "$InformixDIR/extend/MyObject.1.0/MyObject.bld(Mo_circleNotEqual)"
        language c;

grant execute on function NotEqual (Mo_circle,Mo_circle) to public;

create function Mo_radius (Mo_circle)
        returns integer
        with ( not variant )
        external name "$InformixDIR/extend/MyObject.1.0/MyObject.bld(Mo_radius)"
        language c;
```

```
grant execute on function Mo_radius (Mo_circle) to public;

create function Mo_area (circle Mo_circle)
        returns double precision
        DEFINE pi FLOAT;
        LET pi=3.1415926;
        RETURN (pi*POW(Mo_radius(circle),2));
end function;
grant execute on function Mo_area (Mo_circle) to public;
commit work;
```

6.7 ความถูกต้องของข้อมูล(Data Integrity with Constrains)

การเก็บข้อมูลลงในระบบฐานข้อมูล ในกรณีที่มีข้อมูลจำนวนมากอาจจะมีข้อมูลทั้งภาพและเสียง การเก็บข้อมูลจึงจำเป็นจะต้องคำนึงถึง ความถูกต้องของข้อมูลรวมทั้งกฎข้อบังคับต่างๆ ด้วย ซึ่งจะแบ่งออกเป็น 3 แบบ คือ

- Semantic Integrety
- Entity
- Referential

6.7.1 ชนิดของข้อมูล(Data Type) ดาต้าไทพ์จะถูกกำหนดในภาพแบบของคอลัมน์เมื่อมีการสร้างหรือปรับปรุงในส่วนของตาราง ตัวอย่างเช่น คอลัมน์ชื่อ Balance สามารถจะถูกสร้างด้วยข้อมูลชนิด Small Integer หรือ float เพราะว่า ค่าของ Balance เป็นค่าของตัวเลข ถ้ามีการแก้ไขหรือมีการปรับปรุงข้อมูลโดยการที่พยายามจะใส่ ข้อมูลที่เป็น character เข้าไปใน คอลัมน์ balance ซึ่งจะทำให้เกิดข้อผิดพลาด เนื่องจากข้อมูลเป็นคนละชนิดกัน ซึ่งชนิดของข้อมูลที่ใช้ในตารางมีดังนี้

Data type	Data value
BYTE	Byte Data
CHAR or CHARACTER	Determine Size String
DATE	Configurable Date layouts
DATETIME	Configurable Date Time layouts
DEC,DECMAL,NUMERIC	Number configurated to a specific precision
FLOAT or DOUBLE PRECISION	Number preset to double-precision
INT,INTEGER	Number from - 2,147,483,647 to 2,147,483,647
INTERVAL	Configuarable time span layout

MONEY	Configurable currency layout
NCHAR	Mixed mode(Letters, numbers,and symbols),determine size string
NVARCHAR	Mixed mode(Letters, numbers,and symbols),varying size string
REAL or SMALLFlOAT	Single precision numbers
SERIAL	Sequential integers
SMALLINT	Whole numbers from -32767 to 32767
TEXT	Varying size text streams
VARCHARor CHARACTER VARYING	Varying size string

ชนิดของข้อมูลจะถูกกำหนดในขั้นตอนของการสร้างตารางโดยใช้คำสั่งของ SQL คือ CREATE TABLE ซึ่งสามารถที่จะกำหนดชนิดของข้อมูลได้ดังนี้

```
CREATE TABLE Customer (
        Customer_name       CHAR(20),
        Customer_id         SERIAL
        street              VARCHAR(30,20)
        city                CHAR(20)
        state               CHAR( 5)
        zip                 CHAR(10)
        last_update         DATE
        balance             MONEY(5,2)
        total_orders        INT     );
```

การเปลี่ยนแปลงชนิดของข้อมูลสามารถทำได้โดยการใช้คำสั่ง SQL คือ ALTER TABLE ระหว่าง การใช้คำสั่ง ALTER ตัวระบบของฐานข้อมูลจะทำการ COPY ข้อมูลออกจากตารางแล้วจึงทำการเปลี่ยนแปลงชนิดของข้อมูล เมื่อเปลี่ยนแปลงชนิดของข้อมูลแล้วตัวระบบฐานข้อมูลก็จะแทนค่าที่ COPY ออกมาเข้าไปในตารางตามเดิม การใช้คำสั่ง ALTER เพื่อปรับปรุงชนิดของข้อมูล เราสามารถที่จะปรับเปลี่ยนได้ทีละหลาย คอลัมน์เช่น

```
ALTER TABLE Customer MODIFY
        city VARCHAR(20,10);

ALTER TABLE Customer MODIFY
```

```
( city VARCHAR(20,10),
total_orders SMALLINT );
```

นอกจากนั้นเรายังสามารถใช้คำสั่ง ALTER ในการเพิ่มคอลัมน์ใหม่ในตารางได้

```
ALTER TABLE Customer ADD
        phone CHAR(10) BEFORE last_update;

ALTER TABLE Customer ADD
        ( area_code CHAR(3),
          line CHAR(7) );
```

6.7.2 การกำหนดค่าเริ่มต้น(Defualt Value)ในการกำหนดค่าเริ่มต้นเพื่อให้เป็นไปตามกฎ ถ้าในระหว่างการ INSERT ข้อมูลถ้าไม่มีการใส่ข้อมูลในคอลัมน์ ค่าของข้อมูลในคอลัมน์จะถูกกำหนดโดยค่าที่กำหนดในเบื้องต้น

Data type defualt literals

Data Type	literal	Examples
INT,SMALLINT,DEC,MONEY	Integer	1,258,999
FLOAT,SMALLFLOAT,DEC,MONEY	Decimal	1.1,2.58,0.9
FLOAT,SMALLFLOAT,DEC,MONEY NVCHAR,VARCHAR,DATE	Character	"Y","Joe","1-1-90")
INTERVAL	Interval	(2 11)DAY TO DAY
DATETIME	Date and time	96-04-19 11:30

6.7.3 Data type defualt fucntions

Data Type	Function	Purpose
CHAR,NCHAR DESERVERNAME NVARCHAR,VARCHAR or SITENAME		Provide Database server name
CHAR,VARCHAR	USER	Provides the user ID
DATE	TODAY	Providesthe current calenda date in mm-dd-yy format
DATETIME	CURRENT	Provides the current calenda date

and current timein mm-dd-yy

hh:mm::ss format

กำหนดค่าสามารถที่กำหนดค่าเริ่มต้นเอาไว้ในขั้นตอนของการใช้คำสั่งเช่น CREATE TABLE หรือใน ALTER TABLE โดยการกำหนดค่าเหล่านั้นไว้ในแต่ละคอลัมน์เช่น

```
CREATE TABLE Customer (
        Customer_name       CHAR(20)        NOT NULL,.
        Customer_id         SERIAL,
        street              VARCHAR(30,20),
        city                CHAR(20),
        state               CHAR( 5)        DEFUALT "AT",
        zip                 CHAR(10)
        last_update         DATE            DEFUALT TODAY,
        balance             MONEY(5,2)      DEFUALT 0,
        total_orders        INT             DEFUALT 0     );

ALTER TABLE Customer MODIFY
        ( city DEFAULT "Sterling",
          zip DEFAULT "20640"  );
INSERT INTO Customer ( Customer_name ) VALUES ("Joes Pizza");
```

หลังจากที่ได้มีการ INSERT ข้อมูลแล้วข้อมูลที่ได้จากการที่ได้ใส่ข้อมูลเข้าไปและข้อมูลที่ได้มีการ DEFAULT ในตอนนี้คือ

Customer_name	Joes Pizza
Customer_id	1
street	NULL
city	Sterling
state	VA
zip	20640
last_update	4-19-96
balance	0.00
total_orders	0

6.7.4 การเช็คความถูกต้อง (Check Contraints) เมื่อต้องการที่จะใส่ข้อมูลในรูปแบบที่กำหนด เช่น กำหนดขอบเขตของข้อมูล 2 - 15 หรือการกำหนดข้อมูลเช่นให้ใส่ข้อมูลแค่ "M","F" เมื่อเรา INSERT ข้อมูลเข้ามาในตารางจะมีการตรวจสอบข้อมูลเหล่านั้นก่อนว่าเป็นไปตามเงื่อนไขที่มีการกำหนดไว้ก่อนหรือไม่ซึ่งจะใช้การตรวจให้เป็นไปตามข้อกำหนดของข้อมูล การตรวจสอบมีดังนี้คือ

- Coloumn-level Check Constraints
- Table-level Check Constraints

6.7.4.1 การเช็คความถูกต้องระดับคอลัมน์(Coloumn-level Check Constraints) การตรวจสอบในระดับคอลัมน์จะมีการใช้คำสั่ง CHECK .ในคำสั่งของ SQL ซึ่งคำสั่ง CHECK นี้จะต้องอยู่ในภาพแบบของเงื่อนไขเช่น

```
CREATE TABLE Customer (
        Customer_name       CHAR(20)      NOT NULL,
        Customer_id         SERIAL,
        street              VARCHAR(30,20),
        city                CHAR(20),
        state               CHAR( 5)      DEFUALT "VA",
CHECK ( state IN ( "VA","MD","DC")),
        zip                 CHAR(10)
        last_update         DATE          DEFUALT TODAY,
        balance             MONEY(5,2)    DEFUALT 0,
CHECK ( balance BETWEEN 0 and 999 ),
        total_orders        INT           DEFUALT 0
CHECK ( total_orders >= 0 ) );
```

ค่าใดๆ ที่ถูก INSERT หรือ UPDATE ภายในคอลัมน์นี้ (state)จะต้องมีการตรวจเช็คความถูกต้องของข้อมูลที่ใส่เข้าไปหากผิดผลาดนอกจากนี้จะเกิดข้อผิดพลาด ก็จะไม่สามารถที่จะ INSERT หรือ UPDATE ได้

ซึ่งถ้าหากว่าเรามีการ เพิ่มหรือเปลี่ยนแปลงเงื่อนไข ข้อมูลที่อยู่ใน คอลัมน์ก็จะต้องเป็นไปตามเงื่อนไขใหม่ด้วย

```
ALTER TABLE Customer MODIFY
        total_orders CHECK ( toatl_orders >= 1 );
```

6.7.4.2.การเช็คความถูกต้องระดับตาราง(Table-level Check Constraints)เป็นการเช็คตรวจสอบในระดับของตารางนั่นคือในกรณีที่มีการเช็คตรวจสอบในแต่ละคอลัมน์แล้ว เมื่อเป็นไปตามเงื่อนไขของทุกอลัมน์แต่ผิดเงื่อนไขของตารางก็จะทำให้ข้อมูลนั้นเกิดการผิดพลาดและไม่สามารถที่จะใส่ข้อมูลหรือแก้ไขข้อมูลหรือ ปรับปรุงข้อมูลได้ เช่น

```
CREATE TABLE Customer (
        Customer_name       CHAR(20)        NOT NULL,
        Customer_id         SERIAL,
        street              VARCHAR(30,20),
        city                CHAR(20),
        state               CHAR( 5)        DEFUALT "VA",
CHECK ( state IN ( "VA","MD","DC")),
        zip                 CHAR(10)
        last_update         DATE            DEFUALT TODAY,
        cur_balance         MONEY(5,2)      DEFUALT 0,
CHECK ( cur_balance BETWEEN 0 and 999 ),
        prev_balance        MONEY(5,2)      DEFUALT 0,
CHECK ( prev_balance BETWEEN 0 and 999 ),
        last_balance        MONEY(5,2)      DEFUALT 0,
CHECK ( last_balance BETWEEN 0 and 999 ),
        total_orders        INT             DEFUALT 0
CHECK ( total )orders >= 0 ),
CHECK ( prev_balance - last_payment = cur_balance ) );
```

ซึ่งการเช็คตรวจสอบข้อมูล CHECK (prev_balance - last_payment = cur_balance) นี้จะทำตามเงื่อนไขของข้อมูลหลังจากที่มีการตรวจเช็คระดับคอลัมน์แล้ว และถ้าหากมีการใช้คำสั่ง ALTER TABLE เพื่อปรับปรุงเงื่อนไขหรือปรับปรุงตาราง หลังจากนั้นก็จะมีการตรวจสอบเงื่อนไขด้วย

```
ALTER TABLE Customer ADD CONSTRAINT
        CHECK ( prev_balance - last_balance = cur_balance );
```

6.7.5 Entity Integrity เอ็นติตี้ หมายถึง ชื่อของสิ่งใดสิ่งหนึ่ง อาจเกี่ยวกับ คน สถานที่ สิ่งของ การกระทำซึ่งต้องการจัดเก็บข้อมูลไว้ ซึ่งโดยปกติแล้วจะเป็นที่รวมของคอลัมน์และจะถูกใช้

ในการอ้างอิงถึงคอลัมน์อื่นๆ ในการหาคอลัมน์ที่สำคัญจะมีพิจารณาการหาข้อมูลจากคอลัมน์ที่เป็น primary key และ primary key จะต้องมีเอกลักษณ์ คือมีเพียงหนึ่งเดียว (Unique) ในแต่ละคอลัมน์ในตารางเช่น

```
CREATE TABLE Advisors (
        ssn CHAR ( 9 ) UNIQUE,
        name CHAR ( 20 ) );

ALTER TABLE Advisors MODIFY
        ssn UNIQUE;
```

ในกรณีที่เราทำการปรับปรุงตารางโดยใส่ UNIQUE เข้าไปที่คอลัมน์ที่มีการใช้ UNIQUE อยู่แล้วการปรับปรุงนั้นจะใช้ไม่ได้

นอกจากนี้การใช้ Primary Key ในการอ้างถึงข้อมูลตามที่เราต้องการได้ ซึ่งการใช้ Primary Key จำเป็นต้องใช้กับคอลัมน์ที่เป็น Unique เท่านั้นเช่น

```
CREATE TABLE Advisors (
        ssn CHAR ( 9 ) UNIQUE,
        name CHAR ( 20 ),
        PRIMARY KEY ( ssn ) );

ALTER TABLE Advisors ADD CONSTRAINT
        PRIMARY KEY ( name );

CREATE TABLE Classes (
        course_number INT ( 5 ),
        daytaught DATETIME ( HOUR ),
        teacher CHAR ( 9 ),
        PRIMARY KEY ( course_number ,daytaught,timetaught ) );

ALTER TABLE Classes ADD CONSTRAINT
        PRIMARY KEY ( course_number ,daytaught );
```

6.7.6 การอ้างถึงข้อมูล(Referential Constraints)เป็นการอ้างถึงข้อมูลจากตารางหนึ่งไปยังอีกตารางหนึ่ง

```
CREATE TABLE Mutual_funds
(       fund_id   INT,
        fund_name CHAR(12),
        numofowners INT,
        value INT,
        PRIMARY KEY (fund_id) );

CREATE TABLE Mutual_funds
(       fund_id         INT     PRIMARY KEY,
        fund_name       CHAR(12),
        numofowners     INT,
        value           INT );

ALTER TABLE Mutual_funds
        fond_id PRIMARY KEY;

CREATE TABLE Fund_owners
        ( owner_id      INT ,
          owner_name    CHAR(15),
          fund_id       INT,
          numoshares    INT,
          FORIENT KEY       ( fund_id )
                REFERENT mutual_funds );

CREATE TABLE Fund_owners
        (owner_id       INT ,
         owner_name     CHA R(15),
         fund_id        INT     FORIENT KEY
                REFERENT mutual_funds,
         numoshares     INT );
```

```
ALTER TABLE Fund_owners MODIFY
        ( fund_id       INT     FORIENT KEY
                REFERENT mutual_funds );

CREATE TABLE Mutual_funds
(       fund_id         INT     PRIMARY KEY,
        CONSTAINTS pk_fund_id
        fund_name       CHAR(12),
        numofowners     INT,
        value           INT,
);

CREATE TABLE Fund_owners
        (
        owner_id        INT ,
        owner_name      CHA R(15),
        fund_id         INT,
        numoshares      INT
        FORIENT KEY ( fund_id )
                REFERENT mutual_funds,
                CONSTAINTS fk_fund_id );

Self-Referencing Constraints
CREATE TABLE Advising
        (       advisee CHAR(9),
                Advisor CHAR(9),
                PRIMARY KEY ( advisee )
                        CONTRAINTS pk_advisee,
                FOREIGN KEY (Advisor)
                        REFERENCES advising ( student )
                        CONTRAINTS pk_advisee );
```

Cyclic- Referential Constraints

```
CREATE TABLE Advisor
        (       Advisor CHAR(9),
                PRIMARY KEY ( Advisor )
                        CONTRAINTS pk_Advisor  );


CREATE TABLE Advising
        (       advisee CHAR(9),
                Advisor CHAR(9),
                FOREIGN KEY (Advisor)
                        REFERENCES advising ( student )
                        CONTRAINTS fk_advisee,
        );
```

6.7.7 การอ้างถึงข้อมูลพร้อมๆกัน(Multiple-Path Referential Constraints)

```
CREATE TABLE Staff
        (       staff_member CHAR(9),
                PRIMARY KEY ( staff_member ),
                CONSTRAINTS pk_staff_member );


CREATE TABLE Advising
        (       advisee CHAR(9),
                Advisor CHAR(9),
                FOREIGN KEY (Advisor)
                        REFERENCES advising ( student )
                        CONTRAINTS fk_advisee  );


CREATE TABLE Classes
        (       course_number INT(9),
                teacher CHAR(9),
                FOREIGN KEY ( teacher )
                        REFERENCES staff (staff_member )
                        CONTRAINTS fk_teacher );
```

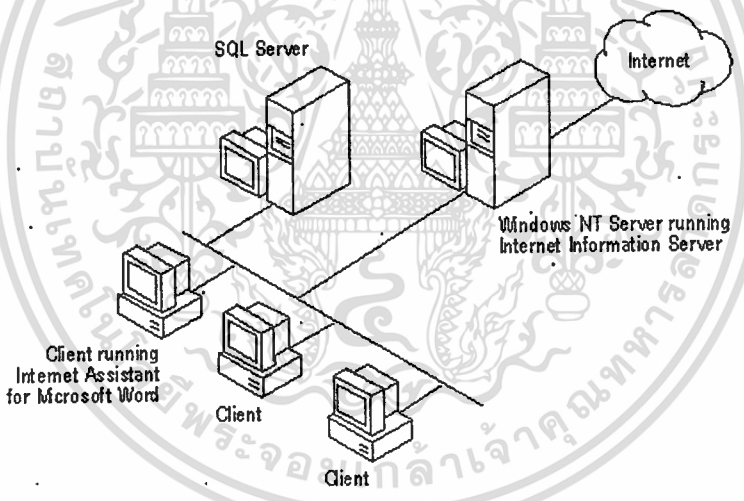
บทที่ 7

การพัฒนาระบบงานบนเครือข่ายอินเตอร์เน็ต

7.1 ระบบเครือข่ายอินเตอร์เน็ต

ระบบเครือข่ายอินเตอร์เน็ต[4],[5],[10] เป็นเทคโนโลยีใหม่ที่มีการพัฒนาปรับปรุงไปอย่างรวดเร็ว จนทำให้ระบบเครือข่ายอินเตอร์เน็ต นี้ในปัจจุบันค่อนข้างจะมีบทบาทต่อชีวิตประจำวันของมนุษย์มากขึ้น ด้วยประสิทธิภาพในการรับส่งข้อมูลได้รวดเร็วนี้เอง ทำให้เกิดแรงจูงใจในการสร้างงานในภาพแบบต่างๆ ที่เราทุกคนควรจะหันหน้ามาศึกษาและติดตามการทำงานของระบบเครือข่ายนี้กันต่อไป

ภาพที่ 7.1



แสดงการใช้เครือข่ายอินเตอร์เน็ต

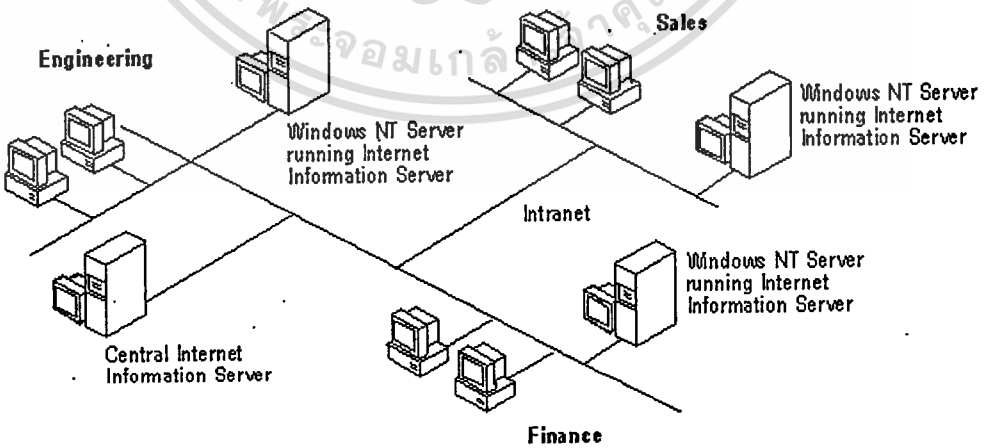
7.1.1 ที่มาของอินเตอร์เน็ต

เครือข่ายอินเตอร์เน็ต ได้เริ่มมีการพัฒนาขึ้นในปี พ.ศ.2512 จากการเป็นเครือข่ายทดลองที่ชื่อว่า ARPA ซึ่งเป็นหน่วยสนับสนุนงานวิจัยทางทหารของสหรัฐอเมริกาโดยได้มีการติดตั้งคอมพิวเตอร์เชื่อมต่อเข้าหากันเป็นครั้งแรกโดยมี HOST หลักเป็น Mini Computer รุ่น 316 ของฮันนี่เวลล์ คอมพิวเตอร์ที่เชื่อมเข้าหากันนั้น เป็นเครื่องหลายชนิดที่มีระบบปฏิบัติการแตกต่างกันและตั้งอยู่บนพื้นที่ต่างกัน 4 แห่ง ซึ่งได้ประสบความสำเร็จและได้เริ่มการใช้งานจริง แต่การขยายเครือข่ายยังติดขัดอยู่ที่โปโตคอล(Protocal)ที่ใช้ทำงานได้อย่างจำกัดจึงมีความพยายามที่จะให้ โปโตคอลสามารถติดต่อสื่อสารกันได้อย่างหลากหลายขึ้นจนมาถึงยุคของ TCP/IP

TCP/IP ทำให้อินเตอร์เน็ตขยายตัวได้อย่างรวดเร็วการจัดทำข้อมูลส่วนกลางจึงเปลี่ยนเป็นพิมพ์เอกสารไฮเปอร์เท็กซ์(Hypertext)บนอินเตอร์เน็ต ไฮเปอร์เท็กซ์ คือข้อความซึ่งมีรหัสควบคุมเพื่อเชื่อมโยงหัวข้อที่มีความสัมพันธ์กันเข้าด้วยกันเฉกเช่นเดียวกันกับ Help ของวินโดวส์ที่คุณสามารถคลิกข้อความหนึ่ง เพื่อดูรายละเอียดเพิ่มเติมของข้อความนั้นบนเว็บการใช้ไฮเปอร์ลิงค์ทำให้ผู้ใช้สามารถดำเนินตามความคิดและหัวข้อต่างๆ จากเพจหนึ่งไปยังอีกเพจหนึ่งได้โดยไม่ต้องสนใจว่าเพจเหล่านั้นจะถูกเก็บอยู่บนคอมพิวเตอร์เดี่ยวๆ (ที่เรียกว่า เว็บเซิร์ฟเวอร์) หรือยู่กระจัดกระจายบนเซิร์ฟเวอร์อื่นๆทั่วโลก(ใน Network การกระโดดข้ามไปมาระหว่างเซิร์ฟเวอร์นี้ เรียกว่า Browsing,Crushing และ Surfing) เมื่อมีเซิร์ฟเวอร์แล้ว สิ่งต่อไปที่ผู้พิมพ์ เว็บ จะต้องทำคือสร้างเพจของตนโดยใช้สิ่งที่เรียกว่า Hypertext Markup Language (HTML) จะสนับสนุนไฮเปอร์ลิงค์ พร้อมด้วยกราฟิกความละเอียดสูง เสียง ภาพ และทำให้ผู้ออกแบบเพจ สามารถจัดลักษณะของหน้าไปตามลำดับ เช่น ชื่อเรื่อง หัวข้อเรื่อง เนื้อหา โดยการใช้ HTML ทำให้ เว็บเพจ สามารถแล้วเกิดความลื่นไหล ใช้ง่ายกว่าเนื้อหาที่คุณพบในบริการออนไลน์อื่น

การสร้างเพจไม่ใช่เรื่องที่ง่ายนัก อันดับแรกคุณจะต้องสร้างไฟล์ HTML ของคุณ เมื่อคุณสร้างเอกสาร HTML เสร็จแล้ว คุณจะต้องนำมันมาใส่ไว้ใน เว็บเซิร์ฟเวอร์ ที่ซึ่งผู้ใช้ เว็บสามารถเข้าถึงได้ คุณสามารถทำงานนี้ได้ด้วยตัวคุณเอง แต่คุณจะต้องใช้เครื่องพีซีที่รันซอฟต์แวร์เว็บเซิร์ฟเวอร์ พร้อมด้วย Router และสายโทรศัพท์ความเร็วสูง

ภาพที่ 7.2



แสดงการเชื่อมโยงเครือข่ายแบบ Intranet

7.1.2 การให้บริการในอินเตอร์เน็ต

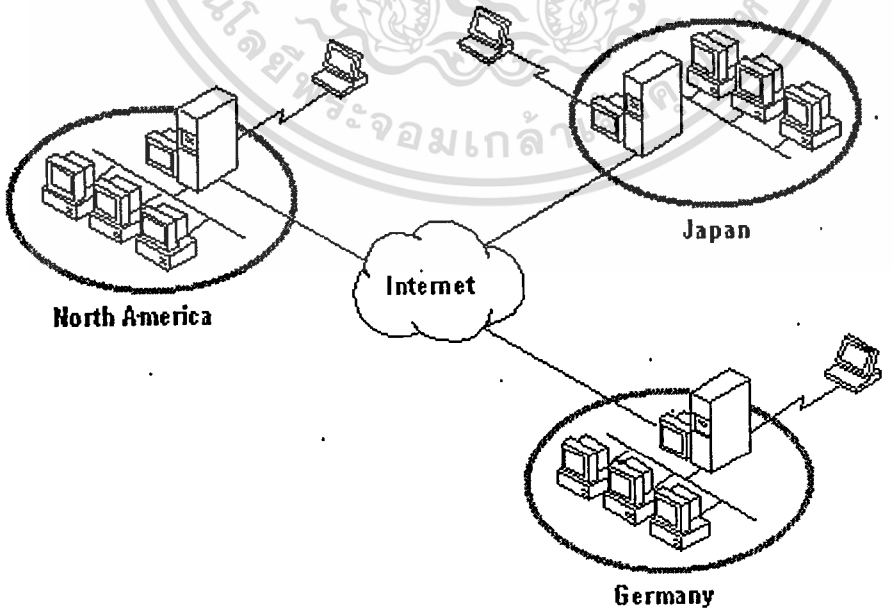
จากปรัชญาของระบบเครือข่ายที่มุ่งหวังให้มีการใช้ทรัพยากรอย่างคุ้มค่าที่สุด จึงสามารถแยกประเภทของการให้บริการหลักๆ ได้ 5 ประเภทได้แก่

- จดหมายอิเลคโทรนิค
- การขนถ่ายแฟ้มข้อมูล
- การใช้โปรแกรมบนเครื่องอื่น
- การบริการค้นหาไฟล์และฐานข้อมูล
- กลุ่มสนทนาและข่าวสาร

7.1.3 เชื่อมโยงเครือข่าย

เมื่อมีเครือข่ายย่อยหลายเครือข่าย ก็หาวิธีการในการเชื่อมโยงเข้าหากัน การเชื่อมโยงเข้าหากันนี้อาจมีการใช้ระบบเชื่อมโยงโดยตรงเป็นเกตเวย์ เชื่อมโยงโดยเราเตอร์หรือบริดจ์ เชื่อมโยงด้วยระบบสื่อสารข้อมูลเพื่อให้เครือข่ายย่อยเชื่อมกันเป็นเครือข่ายเดียวขององค์กร การเชื่อมโยงนี้ใช้มาตรฐานระบบ TCP/IP เป็นหลัก อย่างไรก็ดี ระบบเครือข่ายย่อยที่ใช้ งานอยู่ถ้าเป็นโปรโตคอลอื่นๆ เช่น IPX ก็ยังคงใช้งานของตนเองภายใต้เครือข่ายย่อยๆ นั้น ได้โดยไม่มีปัญหา การใช้งานจึงมีความหลากหลายในระดับย่อยและระดับองค์กร แต่ระดับองค์กรจะใช้ TCP/IP เป็นหลัก

ภาพที่ 7.3



แสดงการเชื่อมโยงเครือข่ายอินเตอร์เน็ต ขนาดใหญ่

7.1.4 การลงทะเบียน (Register)

Register ก็คือการนำ เว็บเพจ ของเราไปใส่ใน Web Site ที่เหมาะสมการหาSite ที่เหมาะสมนั้นขึ้นอยู่กับวิจารณญาณของเราด้วย แต่วิธีการง่ายๆก็คือ ค้นหาจาก Page อื่นๆที่มีอยู่แล้วว่า Page ที่มีเนื้อหาคล้ายๆ กับเราอยู่ใน Site ใด และสามารถค้นหาได้จากวิธีใด เราก็สามารถนำ Site ของเราไปใส่ไว้บ้าง เพราะผู้ที่ค้นหาข้อมูลคล้ายๆ กับเราก็มักจะใช้วิธีค้นหาเช่นเดียวกับเราด้วย การค้นหาจะใช้เครื่องมือที่เรียกว่า "Search Engine" ซึ่งมีอยู่หลายตัว เช่น Yahoo, Magellan ฯลฯ เป็นต้น การ Register อาจมีทั้งต้องเสียค่าบริการและไม่เสียค่าบริการ

Web Site ที่เราสามารถ Register (แบบไม่เสียค่าบริการ) ได้มี 3 ประเภท

- Directory Site เป็น Site ที่เราต้องกรอกข้อมูล ,ต่างๆลงไปเองแล้ว Site เหล่านี้ จะนำข้อมูลที่เรากรอกไปใช้ในการค้นหา เช่น Yahoo,และ TradcWave Galaxy
- reboot Site หรือ Crawler System เป็น Site ที่จะค้นหาข้อมูลจาก เว็บเพจ ของ เราเอง ดังนั้นข้อมูลจริงมีอะไรก็จะใช้ข้อมูลเหล่านั้นในการค้น เช่น WebCrawler และ Lycos
- Announcement Service จะทำหน้าที่ประกาศ โฆษณา เว็บเพจ ของเราไปให้กับ Search System อื่นๆมากมาย(มีทั้งแบบไม่เสียค่าบริการ)เช่น SubmitIT

7.2 ลักษณะของ เว็บเพจ หรือ Home Page ในอินเตอร์เน็ต

เว็บเพจ หรือ Home Page เป็นการแสดง, นำเสนอข้อมูลข้อความหรือภาพภาพภายใต้โค้ดในภาษา HTML ที่ย่อมาจาก Hyper Text Markup Language

ข้อมูลที่อยู่ในภาพของเอกสารที่ปรากฏในอินเตอร์เน็ต นั้นมีอยู่ด้วยกันหลายภาพแบบ ซึ่งอาจมีทั้งการโฆษณา, ประชาสัมพันธ์, การให้บริการทางสังคม, การให้บริการทางธุรกิจ ที่แตกต่างกันไปภายใต้แนวความคิดของผู้สร้างงานที่จะนำเสนอ จากความหลากหลายเหล่านี้จำเป็นต้องมีการกำหนดมาตราฐานในการควบคุมในเบื้องต้นไว้เพื่อให้เอกสารในภาพแบบต่างๆ สามารถเชื่อมต่อหรือมีความสัมพันธ์กันได้ มาตราฐานที่ถูกสร้างขึ้นนี้ จะกำหนดไว้เป็นชุดคำสั่งหรือภาษาที่ใช้ในการใช้เขียนเอกสารที่เราเรียกว่า"HTML"(Hypertext Makeup Language)เครื่องมือหรือโปรมแกรมที่ใช้เขียน HTML เรียกว่า"HTML Editor"ซึ่งในปัจจุบันมีอยู่มากมาย หรืออาจใช้โปรแกรม Text Editor ในการเขียนหรือแก้ไขก็ได้ ไฟล์ข้อมูลที่เขียนจากภาษาHTML นี้เมื่อนำไปจัดเก็บในส่วนให้บริการข้อมูล (เว็บเซิร์ฟเวอร์) แล้วจะเรียกว่า เว็บเพจ หรือ Home Page

เว็บเพจ จากส่วนกลางที่ถูกส่งผ่านระบบเครือข่ายมายังผู้ใช้ ที่ผู้ใช้ จะต้องมีเครื่องมือหรือโปรแกรมที่ใช้ในการเปิดเครื่องมือหรือโปรแกรมนี้เรียกว่า "Web Browser" ซึ่งในปัจจุบันมีอยู่ด้วยกันมากมายแต่ที่นิยมใช้กันมากในขณะนี้คือ Nescape Navigator และ Microsoft Internet Explore

7.2.1 ภาษา HTML

HTML เป็นภาษาที่ใช้ในการเขียน โปรแกรมภาษาหนึ่งของคอมพิวเตอร์ โดยที่เราสามารถใช้ความคิดในภาพแบบและโครงสร้างที่กำหนดได้ การใช้งาน HTML ไม่มีความยุ่งยากซับซ้อนเพียงแต่ผู้ยอมมาต้องกำหนดภาพแบบและโครงสร้างให้เป็นไปตามมาตรฐานของภาษา HTML ก็สามารถใช้งานได้แล้ว คุณลักษณะอีกอย่างหนึ่งของ HTML คือถ้า Browser ไม่สนับสนุนคำสั่งตัวใดมันจะข้ามคำสั่งนั้นไป ฉะนั้นผู้พัฒนาจึงสามารถใช้คำสั่งเพิ่มเติมต่างๆ ได้โดยที่ไม่ต้องกังวลว่าจะมีผลกระทบต่อการใช้งานเว็บบราวเซอร์ที่ไม่สนับสนุนคำสั่งนั้นๆ

วัตถุประสงค์หลังของ HTML ก็คือการส่งข้อมูลภายใต้โครงสร้างที่เป็นมาตราฐานเดียวกันระหว่างผู้ใช้ที่อยู่ในระบบ ข้อดีของ HTML ที่เด่นชัด HTML เป็นภาษาที่ง่ายต่อการเรียนรู้ ซึ่งผู้ใช้ไม่จำเป็นต้องรู้จักคำสั่งทุกตัว และโปรแกรม Editor ทุกตัวก็สามารถสร้างไฟล์ HTML ได้ซึ่งมีผลดีต่อการนำไปพัฒนางานด้านต่างๆ คำสั่งที่ใช้ใน HTML มีขนาดเล็กทำให้ไฟล์ HTML มีขนาดเล็กตามไปด้วยโดยไม่จำเป็นต้องใช้การบีบอัดเข้ามาช่วย ซึ่งจะทำให้สะดวกและประหยัดต่อเนื่องที่ในการจัดเก็บ

ภาพที่ 7.4

```
<HTML>
    <HEAD>
      <META NAME="GENERATOR" Content="Visual Page 1.1a for Windows">
      <TITLE>Simple Select 1 </TITLE> </HEAD>
    <BODY>
    <HR ALIGN="CENTER"><TABLE BORDER="1">
    <?MISQL SQL="SELECT * FROM EMPLOYEE where code ='011';">
            <TR> { <TD>$*</TD> }
    <?/MISQL>       </TR>
    </TABLE> </BODY>
</HTML>
```

แสดง โครงสร้างของภาษา HTML

ข้อเสียของ HTMLเนื่องจาก HTML เป็นภาษามาตราฐานที่มีผู้ใช้นิยมเป็นอย่างมาก ทำให้มีข้อจำกัดในการใช้คำสั่งเนื่องจากไม่สามารถสร้างคำสั่งใหม่ๆ ขึ้นมาใช้เองได้ จึงจำเป็นต้องใช้คำสั่งที่มีอยู่เท่านั้น และโครงสร้างของ HTML เป็นการกำหนดขึ้นจากโครงสร้างที่เป็นตัวอักษรซึ่งอาจทำให้บราวเซอร์ที่ไม่ใช่ชนิดเดียวกันมีการแสดงผลที่แตกต่างกันไปได้

7.2.2 สิ่งต่างๆ ที่บรรจุอยู่ใน เว็บเพจ

- Text : เป็นข้อความ ,ตัวอักษร, สัญลักษณ์ รวมทั้งอักษรพิเศษที่ไม่ปรากฏบน Keyboard ด้วย Word Processing ซึ่งตัวอักษรต่างๆ เหล่านี้สามารถตกแต่งภาพแบบให้สวยงามตามความ ต้องการของผู้ออกแบบได้ภายใต้คำสั่งภาษา HTML
- Graphic : ไฟล์ภาพภาพที่เรียกว่า Graphic File (ภาพภาพที่แสดงบน เว็บ จะเรียกว่า "Image") โดยทั่วไปจะต้องเป็นไฟล์ที่มีการจัดเก็บขนาดเล็กเพื่อความรวดเร็วในการสั่งภาพไปยัง เว็บบราวเซอร์ ต่างๆ ภาพแบบไฟล์ที่นิยมใช้ในการสร้าง GRAPHIC บน เว็บ คือ
  - GIF Format (Graphical Image Format) ใช้เทคนิคที่เรียกว่า"Losses Corporation" คือ เมื่อผ่านการ Compress แล้ว จะมีการสูญหายของข้อมูลน้อยนั่นคือ ภาพจะเหมือนต้นฉบับมากแต่ข้อเสียคือ Software ที่จะสนับสนุน GIF มีน้อยและภาพที่ได้ก็ใช้สีได้สูงสุดเพียง 256 สีเท่านั้น
  - JPEG Format (Joint Photographic Expert Group) ใช้เทคนิคที่เรียกว่า "Losses Compression" คือมีโอกาสที่จะได้ไฟล์ที่ไม่เหมือนต้นฉบับ แต่สิ่งที่ดีมากก็คือใช้สีได้ สูงสุดถึง 16.7 ล้านสีซึ่งทำให้เป็นภาพแบบที่นิยมกันมากสกุลของ JPEG อาจมีได้ทั้ง .JPG,.SPEG, หรือ .JPE
- Multimedia : เป็นภาพภาพที่แสดงความเคลื่อนไหวประกอบเสียง ซึ่งเรามักแยกได้เป็น Audio, Video และ Animation
  - Audio : เป็นคลื่นเสียงที่เกิดขึ้นจากการแปลงสัญญาณจากข้อมูล Digital การใช้ Audio กับเอกสาร HTML นั้นจำเป็นต้องมีอุปกรณ์หรือโปรแกรมที่จะนำมาใช้งานโปรแกรมเหล่านี้ได้ โปรแกรมนั้นเรียกว่า "Plug In" ซึ่งมีอยู่หลายภาพแบบของ Audio โดยปกติมี 3 ชนิด คือ
    - Digitize Audio เป็นภาพแบบของเสียงที่ทำงานโดยถูกแปลงจาก Analog ไปเป็นDigital เพื่อใช้กับคอมพิวเตอร์ได้

●Music File เป็นภาพแบบของตัวโน๊ตดนตรีที่เรียงลำดับกันเพื่อให้เล่นออกมาได้เป็นเสียงเพลง

●Text to Speech เป็นเทคนิคการแปลงข้อความ (Text File) ให้เป็นเสียงพูด (Speech) ได้

●Video : Video ที่นำมาแสดงบน เว็บเพจ สามารถแสดงเป็นภาพยนต์สั้นๆ ช่วยเพิ่มความสนใจให้แก่ เว็บ นั้นๆ เป็นอย่างดี สัญญาณ digital ที่ถูกส่งมาเป็นชุดจะเรียกว่าFrame rate ถ้า Frame Rate มีค่ามากคุณภาพของ Video ก็จะมีมากขึ้นไปด้วย นอกนั้นยังมี Frame Size หรือขนาดของ Frame ก็เป็นส่วนแสดงความคมชัดของภาพด้วยเหมือนกัน ในระบบ Video File มักมีขนาดใหญ่ จึงจำเป็นต้องมีการบีบอัดที่เรียกว่า Codec (โคเด็ค) ซึ่งกระทำจาก Software ที่ผลิตนั้น ซึ่งจะให้ประสิทธิภาพที่แตกต่างกันไป เช่นCinepak(Compac Video),Indeo,JPEG,MPeG Codec เป็นต้น

●Animation : วิธีที่ง่ายที่สุดในการสร้างภาพเคลื่อนไหวบน เว็บ ก็คือ การนำภาพภาพ .GIF หลายภาพ มาเก็บรวมกันเป็น File GIF Animation และส่งให้แสดงออกมาเรียงกันตามลำดับ ทำให้ได้ภาพเสมือนมีการเคลื่อนไหวอยู่ หรืออาจใช้วิธีการเขียน Script ส่งให้ภาพแสดงที่ละภาพตามลำดับก็ได้

●Counter : ใช้นับจำนวนที่มาเยี่ยมเยียน เว็บเพจ ของเรา ซึ่งฟังก์ชันในการนับนี้ จะถูกควบคุมจาก Script ที่ทำงานอยู่บน เว็บเซิร์ฟเวอร์

● Cool Link Link : เป็นการเชื่อมโยงจากจุดหนึ่งของเอกสารที่กำลังแสดงอยู่ใน Browser ใน ปัจจุบัน ไปยังอีกจุดหนึ่งของเอกสารเดียวกัน หรือคนละเอกสารก็ได้ ซึ่งการ Link อาจจะเป็น การ Link กันระหว่าง Text กับภาพภาพก็ได้การ Link แบ่งออกเป็น

- การ Link ไปยัง File อื่น : สำหรับการ Link ไปยังไฟล์อื่นๆ ที่ไม่ใช่ในระบบของเราจะต้องระบบภาพแบบของ URL ให้ถูกต้องซึ่ง URL นี้จะทำงานภายใต้โปโตคอล TCP/JP เช่น http://www.kmiitl.AC.TH สำหรับการ Link ไปยังไฟล์ในระบบเดียวกันต้องระบุ Path (เส้นทาง) ไปยังไฟล์ HTML ให้ถูกต้องภายใต้ภาพแบบคำสั่งของ HTML
- การ Link ไปยังส่วนอื่นภายในเอกสารเดียวกัน : ต้องมีการกำหนด Label หรือชื่อให้กับตำแหน่งที่ต้องการจะ Link ไปก่อน จากนั้นจึงทำการ Link ไปยัง Label ที่ตั้งไว้อีกที่หนึ่ง

- การ Link ไปยังส่วนใดๆ ใน File อื่น : ต้องมีการกำหนด Label ให้กับตำแหน่งภายในเอกสารที่ต้องการจะ Link ไป และเมื่อทำการ Link จะต้องกำหนดชื่อไฟล์ต่อท้ายด้วยตำแหน่งของไฟล์ที่กำหนดโดย Label ด้วย
- การ Link ไปยังบริการอื่นบนอินเตอร์เน็ต เช่น Link ไปยัง FTP, Gapher หรือ Email โดยหลักการทั่วไปจะต้องกำหนด URL ของปลายทางให้ถูกต้องและเพิ่มเติมด้วยคำสั่งหรือภาพแบบในการใช้บริการนั้นๆ โดยเฉพาะ

- Form : เป็นลักษณะของระบบสอบถามอย่างหนึ่งที่ใช้ในอินเตอร์เน็ต ซึ่งมีภาพแบบให้เลือกใช้งานได้ดังนี้
  - Text Area : เป็นภาพแบบการเขียนข้อมูลแบบอิสระ คือผู้เขียนสามารถป้อนข้อมูลลงไปในบริเวณที่กำหนดมากกว่า 1 บรรทัด
  - Select : เป็นการแสดงค่าตัวเลือกเพื่อให้ผู้ใช้เลือก โดยอาจแสดงในภาพของ DROP-downlist หรือแสดงค่าให้เลือกทั้งหมดตามปกติก็ได้
  - Input : เป็นคำสั่งที่ใช้ในการป้อนข้อมูล 1 บรรทัด โดยสามารถจำกัดภาพแบบ หรือ จำนวนอักษรในการป้อนก็ได้ มีเพิ่ม Check Box, Option, Radio

การทำงานของ Form : Foum ที่ถูกสร้างขึ้นในเบื้องต้นนั้น ยังไม่สามารถทำงานได้อย่างสมบูรณ์เนื่องจากยังมิได้มีการระบุลงไปว่าค่าที่เติมลงในช่องกรอกข้อความนั้นจะส่งไปยังที่ใด, ต้องการติดต่อกับฐานข้อมูลหรือไม่ หรือจะให้มีการประมวลผลอย่างไร

การใช้งานและควบคุมข้อมูลเหล่านี้ จะถูกกระทำโดยโปรแกรมย่อยๆ หรือ Script ที่ทำงานอยู่ภายใน เว็บเซิร์ฟเวอร์ ซึ่งจะทำหน้าที่รับคำสั่ง และตรวจสอบค่าได้เพื่อให้นำค่านั้นไปดำเนินการตามคำสั่งใน Script โดย Script ที่ใช้ทำหน้าที่เชื่อมโยงข้อมูลระหว่างผู้ใช้กับ Server ที่นิยมใช้งานปัจจุบันมี 2 ชนิด คือ CGI (Common Gateway Interface) และ IIS (Internet Information Server) ซึ่งจะเป็นการศึกษาในรายละเอียดในช่วงต่อไป

- Frame : เป็นการแบ่งจอภาพออกเป็นส่วนๆ ซึ่งแต่ละส่วนแสดงข้อมูลที่แตกต่างกันเป็นอิสระจากกัน
- Image Maps : เป็นเทคนิคในการนำภาพแบบมาใช้บน เว็บเพจ โดยภาพภาพนั้นจะถูกแบ่งออกเป็นส่วนย่อยๆ (โดยเงามองไม่เห็น) โดยใช้จุดคู่ลำดับเป็นเกณฑ์ในการจัดแยกพื้นที่ของภาพภาพซึ่งเรียกว่า "Hot Spot" ส่วนย่อยแต่ละส่วนจะ

ทำหน้าที่ Link ไปยังเอกสารอื่น ตามที่ออกแบบไว้ การกำหนด Image Map ทำได้ 2 วิธีคือ

- Server - Side Image (SSM) : จะเป็นการทำงานโดย Server ซึ่งจะทำให้ประสิทธิภาพการทำงานของ Server ต้องเสียไปส่วนหนึ่ง วิธีนี้ปัจจุบันไม่นิยมใช้
- Client - Side Image Maps (CSIM) : Client จะเป็นผู้ทำงานเกี่ยวกับ Image Map เองที่จะช่วยแบ่งภาระของ Server ลง ซึ่งจะทำให้ Server ทำงานได้เต็มประสิทธิภาพ
- Server Applets : เป็นโปรแกรมสำเร็จภาพเล็กๆ ที่ใส่ลงไปใน เว็บเพจ เพื่อให้การใช้งาน เว็บเพจ มีประสิทธิภาพมากขึ้น

7.2.3 ภาพแบบของเอกสารในอินเตอร์เน็ต

- ภาพแบบของเอกสารที่เกิดขึ้นในอินเตอร์เน็ต สามารถแบ่งออกเป็น 2 ลักษณะ คือ
- ข้อมูลคงที่ (STATIC) เป็นข้อมูลที่ถูกนำเสนอจากผู้ให้บริการเพียงด้านเดียว ภาพแบบที่เกิดขึ้นในหน้าจอของ เว็บเพจ ส่วนใหญ่จะอยู่ในภาพของ Hypertext Markup Language (HTML) ซึ่งมี Hypertext Link เป็นตัวเชื่อมความสัมพันธ์ของข้อมูลภายในหน้าหนึ่ง ไปยังอีกหน้างหนึ่งซึ่งข้อมูลชนิดนี้ผู้ใช้บริการไม่สามารถแก้ไขหรือโต้ตอบกับผู้อื่นได้ ลักษณะที่สำคัญในการทำงานของข้อมูลแบบคงที่นี้คือ ข้อมูลที่เกิดขึ้นบน เว็บเพจ จะต้องเป็นข้อมูลที่ได้มีการเตรียมข้อมูลหรือคำตอบไว้ล่วงหน้าแล้ว เพื่อให้ผู้ใช้ได้รับข้อมูลตามที่ผู้เขียนกำหนดเท่านั้น
- ข้อมูลไม่คงที่ (DYNAMIC) เป็นข้อมูลสามารถโต้ตอบได้ 2 ทาง แบบ Interactive ผู้ใช้บริการหรือผู้ใช้ สามารถส่งคำขอผ่านแบบฟอร์มจาก Web Browser กลับไปยังเครื่องแม่ข่าย (เว็บเซิร์ฟเวอร์) เพื่อให้ได้คำตอบที่เปลี่ยนแปลงไปตามการประมวลผลของกระบวนการทำงานได้ ซึ่งข้อมูลที่เปลี่ยนแปลงไปนี้อาจได้มาจากการทำงานของโปรแกรมที่เรียกว่า Script หรือการเรียกใช้ข้อมูลจากฐานข้อมูลก็ได้

7.3 ส่วนประกอบที่สำคัญของระบบ

การนำส่วนต่างๆ มาประกอบกันเพื่อให้เกิดวิธีการตามขั้นตอนของงานใหม่ที่เรากำหนดนั้น จำเป็นต้องสรรหาสิ่งที่เหมาะสม,ความเข้ากันได้ของแต่ละส่วนภายใต้ผลที่มีประสิทธิภาพตรงตามความต้องการ ซึ่งระบบงานนี้สามารถแบ่งหน้าที่ ที่เป็นกลไกหลักของระบบออกเป็น 4 ส่วนคือ

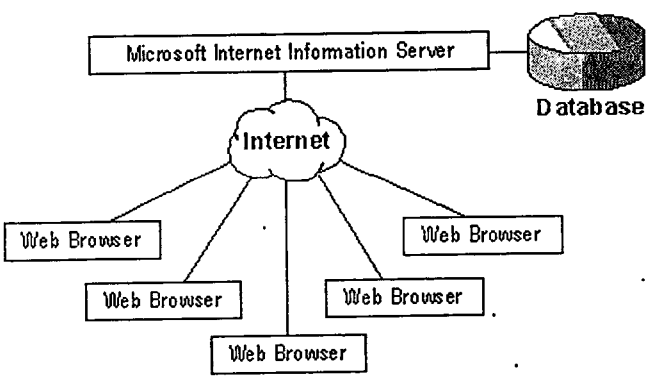
- Database
- Web Server
- Open Database Connectivity (ODBC)
- Internet Information Server (IIS)

7.3.1 ระบบฐานข้อมูล (Database)ระบบฐานข้อมูลเป็นกลไกสำคัญในการจัดเก็บข้อมูล ฉะนั้นการศึกษาในจุดนี้จะเป็นแนวทางหลักในการแก้ปัญหาการใช้ข้อมูลข่าวสารในระบบซึ่งมีเนื้อหาที่สำคัญดังนี้

ฐานข้อมูล (Dabase) คือ การรวบรวมข้อมูลที่สัมพันธ์กัน และกำหนดภาพแบบการจัดเก็บอย่างเป็นระบบ การจัดเก็บเป็นฐานข้อมูลมักจะจัดเก็บไว้ที่หน่วยศูนย์กลาง ทั้งนี้เพื่อให้ผู้ใช้จากหลายๆ หน่วยในองค์กรสามารถเรียกใช้ข้อมูลที่จัดเก็บไว้ได้ตามต้องการของแต่ละหน่วยงาน

การใช้ฐานข้อมูลเพื่อตอบสนองต่อความต้องการสารสนเทศ องค์การส่วนใหญ่จะจัดสร้างระบบฐานข้อมูล (Database System) ให้เป็นส่วนหนึ่งของระบบสารสนเทศขององค์การ ความจริงแล้วฐานข้อมูลเกือบจะถือได้ว่า เป็นส่วนสำคัญที่สุดของระบบสารสนเทศของบริษัทถ้าจะเปรียบเทียบง่ายๆ ฐานข้อมูลก็จะเหมือนกับห้องสมุดคือเป็นห้องสมุดของข้อมูลต่างๆ และเช่นเดียวกับห้องสมุด ฐานข้อมูลไม่ใช่มีแต่ข้อมูลเพียงอย่างเดียว แต่จะต้องมีการจัดระเบียบและกำหนดการจัดการควบคุม และการใช้ข้อมูลในระบบอีกด้วย

ภาพที่ 7.5



แสดงระบบฐานข้อมูลที่ใช้ในระบบเครือข่ายอินเตอร์เน็ต

ปัญหาสำคัญที่มักจะเกิดขึ้นในการสร้างและใช้ฐานข้อมูลภายในองค์การ จำแนกได้เป็น 4 ลักษณะใหญ่ๆ ด้วยกัน คือ

1.มีความซ้ำซ้อนกันในการเก็บข้อมูล (Uncontrolled Data Redundancy)

2.มีการให้คำจำกัดความของข้อมูลแต่ละตัวไม่ตรงกัน(Inconsistent data Definition`Data Definition)

3.มีการจัดการหรือใช้วิธีการประมวลผลของข้อมูลแตกต่างกันออกไป (Inconsistent Data Manipulation)

4.มีการพัฒนาการใช้ข้อมูลไม่เป็นระบบ แต่ละหน่วยงานจัดทำ หรือใช้กันอย่างไม่มีกฎเกณฑ์ (fragmental application development)

กล่าวโดยสรุปจะเห็นได้ว่าการออกแบบสร้างระบบที่จะประมวลข้อมูล ให้กลายเป็นสารสนเทศที่มีความถูกต้องไม่ผิดพลาด มีความหมายและตรงต่อความต้องการของผู้ใช้ระบบ ไม่ใช่เรื่องที่ทำได้สำเร็จในเร็ววัน จะต้องมีการวางแผนพัฒนากันอย่างเป็นขั้นตอน

ระบบจัดการฐานข้อมูล (DBMS-Database Management System) คือ ระบบโปรแกรมที่มีความสามารถในการจัดการข้อมูลในด้านต่างๆ ได้แก่ การให้คำจำกัดความของข้อมูลและเรคอร์ดการกำหนดความสัมพันธ์ระหว่างฟิลด์ต่างๆ ในเรคอร์ดการจัดการประมวล ปรับเปลี่ยนแก้ไขข้อมูล และการจัดการกำหนดควบคุมการใช้ข้อมูลที่มีอยู่ได้อย่างเป็นระบบ

จุดมุ่งหมายสำคัญของระบบ DBMS จำแนกได้เป็น 2 ด้าน คือ

- เพื่อจัดการควบคุม
- เพื่อสนับสนุนการใช้ข้อมูลภายในองค์การอย่างเป็นระบบ

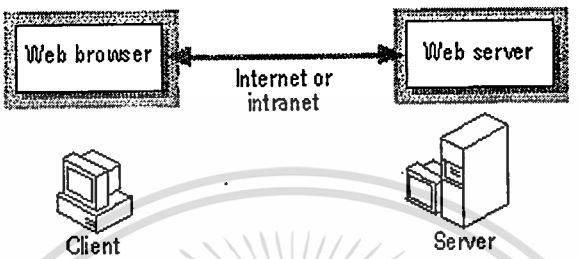
แต่อย่างไรก็ตามระบบงานใดๆ มิใช่จะใช้คอมพิวเตอร์แต่เพียงอย่างเดียว การตัดสินใจบางอย่างอาจอยู่นอกเหนือกฎเกณฑ์ที่คอมพิวเตอร์จะคิดได้ ฉะนั้น การนำคอมพิวเตอร์มาใช้งาน ยังคงเป็นเพียงเครื่องช่วยเสริมของการทำงานในระบบเท่านั้น

7.3.2 เว็บเซิร์ฟเวอร์

- **การตั้งเว็บเซิร์ฟเวอร์**เว็บเซิร์ฟเวอร์เป็นศูนย์กลางขององค์กร ดังนั้นจึงต้องตั้งเว็บเซิร์ฟเวอร์เพื่อเป็นศูนย์กลางขององค์กรพนักงานในองค์กรทุกคนใช้บราวเซอร์บอกผ่านเครื่องพีซีของเครือข่ายตนเองเพื่อเชื่อมต่อเข้าสู่เว็บเซิร์ฟเวอร์ เว็บที่ตั้งเป็นศูนย์กลางยังทำให้เชื่อมโยงต่อเข้าสู่ระบบฐานข้อมูลขององค์กรกับเซิร์ฟเวอร์ตัวอื่นขององค์กร
- **กรอกข้อมูลข่าวสารที่จะเขียนลงบนเว็บเซิร์ฟเวอร์**เพื่อให้เว็บเซิร์ฟเวอร์เป็นศูนย์กลาง องค์กรจำเป็นต้องสร้างข้อมูลข่าวสารในศูนย์กลางที่ทำให้เกิดประโยชน์และเป็นตัวเชื่อมโยง

เข้าสู่ระบบฐานข้อมูลข้อมูลเฉพาะอื่นๆ ในองค์กรการหาข้อมูลข่าวสารขององค์กรและการเขียนเว็บเพจจำเป็นต้องเรียนรู้วิธีการเขียนโดยเฉพาะHTMl ซึ่งเป็น

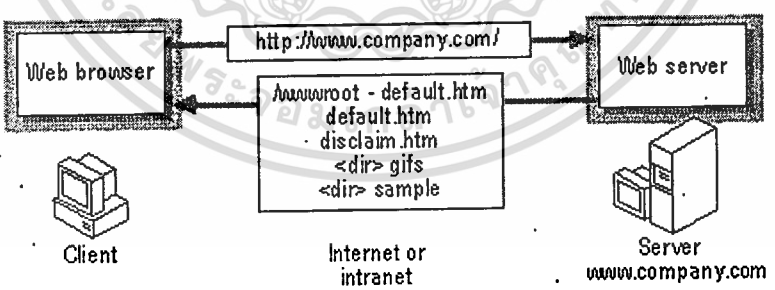
ภาพที่ 7.6



แสดงระบบเว็บเซิร์ฟเวอร์ ที่ใช้ในระบบเครือข่ายอินเตอร์เน็ต

ภาษาสากลสำหรับการเขียน ภาษา HTML เป็นภาษาที่ไม่ยุ่งยาก สามารถเรียนรู้วิธีการเขียนได้เอง การออกแบบเว็บทำให้สวยงามและน่าสนใจเพียงไรก็ได้ ขึ้นอยู่กับศิลปและวิธีการของการสร้างสรรค์ การศึกษาวิธีการเขียน HTML หาได้จากที่ต่างๆ ที่ไว้ในเว็บบนอินเตอร์เน็ตหรือหาจากหนังสือทั่วไปได้

ภาพที่ 7.7



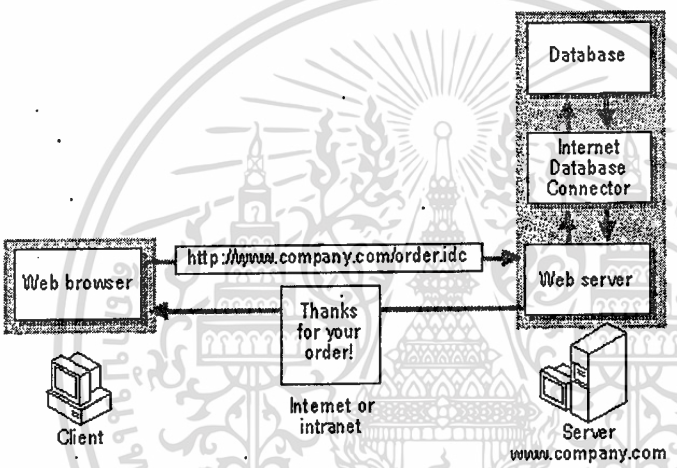
แสดงระบบการสื่อสารของ Web Browser กับ เว็บเซิร์ฟเวอร์

7.3.3 Open Database Connectivity (ODBC)

ODBC หรือ Open Database connectivity เป็นองค์ประกอบใน Windows ServicesArchitecture (WOSA) ของไมโครซอฟต์ซึ่งเป็นความพยายามที่จะให้ windows เป็นระบบเปิดสามารถเชื่อมต่อกับระบบอื่นๆ ได้

ODBC เป็นตัวกลางในการเชื่อมต่อระหว่างแอพพลิเคชันกับตัวจัดการฐานข้อมูลแบบต่างๆ(DBMS) โดยใช้ภาษา SQL (Structured Query Language) เป็นหลักในการสอบถามและทำงานกับข้อมูล ไม่ว่าจะติดต่อกับตัวจัดการฐานข้อมูลใดผู้พัฒนาแอพพลิเคชันด้วย ODBC จะใช้ภาพแบบคำสั่งของ ODBC เหมือนกันโดยไม่ต้องสนใจว่ากำลังเชื่อมต่อกับ DBMS ใดเป็นการลดความสับสนในการติดต่อกับ DBMS หลากแบบ การติดต่อจริงจะเป็นหน้าที่ของไดรเวอร์สำหรับ DBMS แต่ละตัวซึ่งจะถูกเลือกจากคำสั่งว่าเราต้องการเชื่อมกับตัวจัดการฐานข้อมูลใด

ภาพที่ 7.8

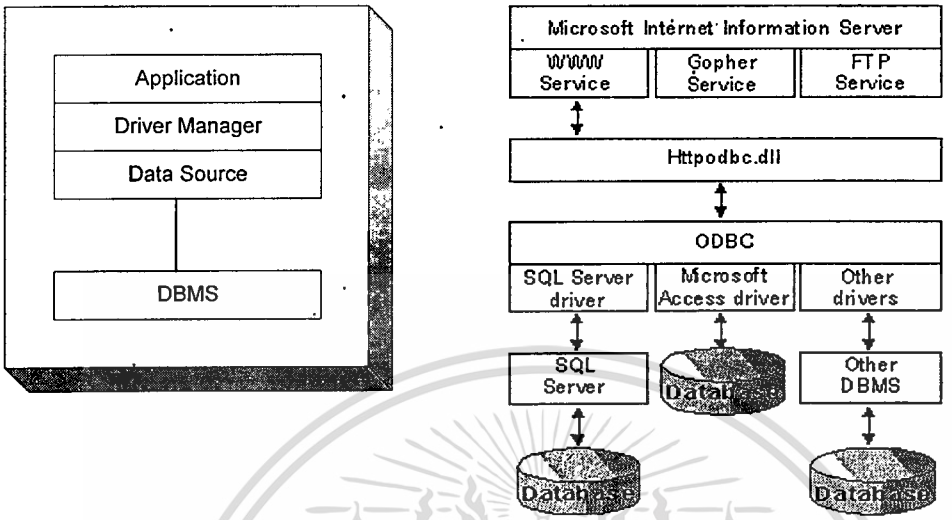


แสดงการต่อระบบฐานข้อมูลผ่าน เว็บเซิร์ฟเวอร์c และ ODBC

องค์ประกอบของ ODBC ประกอบด้วย

- Driver Manager ทำหน้าที่โหลดและจัดการทำงานกับไดรเวอร์ ว่าต้องการติดต่อกับฐานข้อมูลใด ต้องใช้ไดรเวอร์ตัวไหน
- ไดรเวอร์ ทำหน้าที่โปรเซสคำสั่ง ODBC และส่งความต้องการไปทำงานยังฐานข้อมูลซึ่งหากจำเป็นก็จะมีการแปลคำสั่ง ODBC ไปเป็นคำสั่งของ DBMS นั้นๆ ที่ทำงานได้เทียบเท่ากัน

ภาพที่ 7.9



แสดงการทำงานของ ODBC

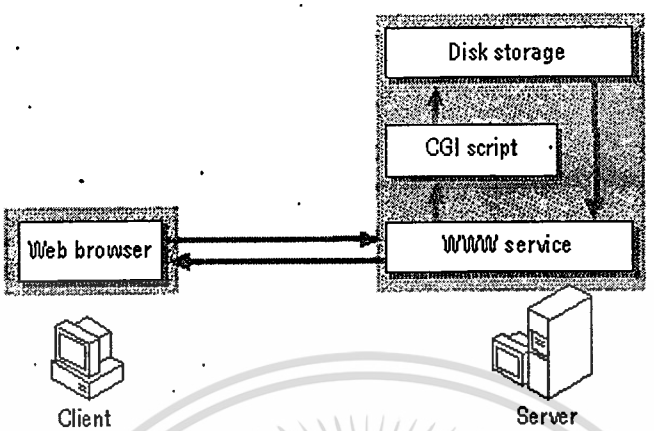
Data Source เป็นแหล่งเก็บข้อมูลและแพลตฟอร์มที่สัมพันธ์กันของแหล่งเก็บข้อมูลนั้นเช่น OS, DBMS หรือระบบเนตเวอร์ค ซึ่งตัว ODBC จะใช้เป็นข้อมูลในการเข้าไปติดต่อทำงานกับระบบจัดการฐานข้อมูล(DBMS)ที่เก็บข้อมูลนั้นไว้

จะเห็นว่าการติดต่อกับ Data Source ต่างแบบกันนั้นเราเพียงระบุแหล่งข้อมมูลให้กับ DriverManager ผ่านคำสั่งของ ODBC เท่านั้น Driver Manager จะเลือก ไดรเวอร์มาใช้งานได้เองโดยที่ผู้พัฒนา แอพพริเควันไม่ต้องไปสนใจถึงภาพแบบคำสั่งที่แท้จริงของตัวจัดการฐานข้อมูล เพียงใช้คำสั่ง ODBC ภาพแบบเดียวเท่านั้น

7.3.4 ส่วนให้บริการข้อมูลในอินเตอร์เน็ต (Internet Information Server)

การศึกษาในทางนี้เปรียบเสมือนเป็นตัวเชื่อมการทำงานของทุกส่วนให้มีความสัมพันธ์ อย่างมีประสิทธิ ภาพ ระบบอินเตอร์เน็ต นับว่าเป็นระบบที่เกิดขึ้นได้ไม่นาน แต่มีผู้พัฒนาเทคโนโลยีที่ใช้กับระบบนี้อย่างรวดเร็วและมากมาย Application ต่างๆ จึงมีการพัฒนาอย่างรวดเร็วตามไปด้วย การใช้งานอินเตอร์เน็ต แบบ Interactive หรือ Web Dynamic นั้นนับว่าเป็นปัจจัยที่สำคัญปัจจัยหนึ่งของInternet การใช้งานและควบคุมข้อมูลเหล่านี้กระทำโดยโปรแกรมย่อยๆ หรือ Scripts ที่เรียกว่า"Common Fateway Interface (CGI)" ที่ติดตั้งอยู่บน เว็บเซิร์ฟเวอร์

ภาพที่ 7.10



แสดงการทำงานของระบบ CGI

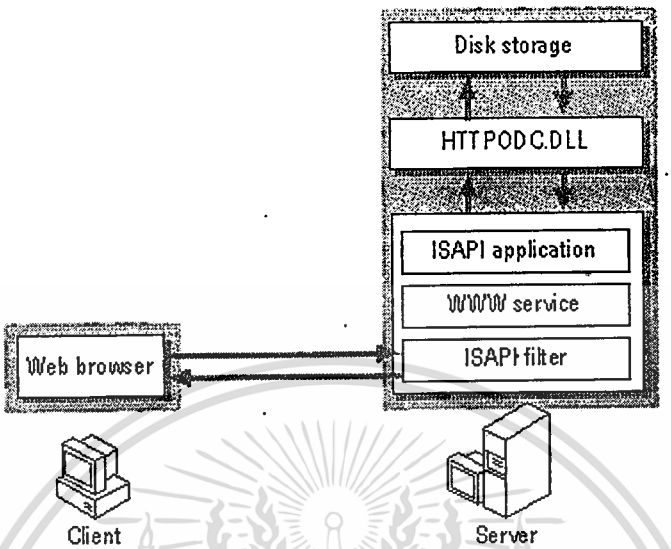
CGI เป็นโปรแกรมตัวกลางที่ทำหน้าที่เชื่อมโยงข้อมูลระหว่างเครื่องผู้ใช้กับ เว็บเซิร์ฟเวอร์ ที่มีการโต้ตอบข้อมูลในขั้นที่ซับซ้อนกว่า HTML โดยสามารถรวบรวมข้อมูลที่ผู้ใช้ส่งมาเพื่อประมวลผล เมื่อเสร็จแล้วก็ส่งผลลัพท์กลับไปยังผู้ใช้ โปรแกรม CGI นี้สามารถพัฒนาได้จากภาษาต่างๆ เช่น ภาษา C, Perl เป็นต้น

Internet Information Server (IIS) เป็น เว็บเซิร์ฟเวอร์ ตัวหนึ่งที่ถูกพัฒนาขึ้นโดยบริษัท Microsoft ที่ถูกผลิตขึ้นมาให้สามารถทำงานได้ในภาพแบบดังนี้

- การกระจายข้อมูลในรูปแบบของการประกาศ(ผู้บริโภคเป็นผู้รับข้อมูลด้านเดียว)
- การโฆษณา (ผู้บริโภคและผู้ผลิตสามารถโต้ตอบหรือแลกเปลี่ยนข้อมูลข่าวสารระหว่างกัน)
- สามารถติดต่อใช้ข้อมูลจากฐานข้อมูลได้

IIS ได้มีการพัฒนางานในลักษณะของ CGI เป็นของตนเองชุดโปรแกรมที่มากับ IIS นี้เรียกว่า Microsoft Internet Server Application Interface (ISAPI) แต่ยังสามารถทำงานในภาพแบบของ CGI ได้เช่นกัน

ภาพที่ 7.11



แสดงการทำงานของระบบ ISAPI

ISAPI (Internet Server Application Program Interface) เป็นช่องทางการติดต่อสื่อสารที่จะถ่ายทอดข้อมูลจากระบบ Hypertext ไปเป็นระบบฐานข้อมูล โดยผ่านตัวควบคุม ที่เรียกว่า Internet Database Connector (IDC) หรือ Httpodc.dll และนำข้อมูลจากระบบฐานข้อมูลกลับมายัง Hypertext อีกครั้งหนึ่ง ซึ่งมีการทำงานดังนี้

ผู้ใช้จะส่งค่าผ่านทางฟอร์มที่สร้างขึ้นจาก Web Brower ไปยังเว็บเซิร์ฟเวอร์ ซึ่งจะทำให้ ISAPI ที่ติดตั้งอยู่กับเว็บเซิร์ฟเวอร์ทำงานชุดคำสั่งที่อยู่บน ISAPI มีลักษณะเดียวกับคำสั่งของ SQL (ไฟล์.IDC) จากนั้น ISAPI จะติดต่อกับตัวฐานข้อมูลโดยเรียกผ่าน ODBC ซึ่งจะติดตั้งอยู่ภายใต้ HTTPODC.DLL ค่าที่ได้กลับมาอยู่ในภาพของ (ไฟล์.HTX) และจะถูกส่งไปยังผู้ใช้โดยการควบคุมของ Web Browser อีกทีหนึ่ง

CGI (Common Gatewey Interface) เป็นวิธีการที่ทำให้ผู้ใช้ ที่ใช้บริการ เว็บ เรียกใช้โปรแกรมอื่นที่ไม่มีให้บริการใน เว็บเซิร์ฟเวอร์ ได้ ซึ่งมีการทำงานคือผู้ใช้จะส่งคำขอจาก Web Browser ไปยัง เว็บเซิร์ฟเวอร์ และเว็บเซิร์ฟเวอร์ จะไปเรียกการทำงานของ CGI Scrip (เป็นสคริปที่ถูกเขียนขึ้นเพื่อใช้งานเฉพาะฐานข้อมูลนั้นๆ) ให้ทำงานกับฐานข้อมูลที่ได้จะถูกส่งกลับไปยัง เว็บเซิร์ฟเวอร์ เพื่อส่งกลับไปยังผู้ใช้ โดยไม่ผ่าน CGI อีก

ข้อเสียของ CGI คือ เวลาเรียกใช้ Application ก็จะต้อง Run Program ใหม่ทุกครั้ง หากมีหลายผู้ใช้ ทำการเรียกใช้โปรแกรมตัวเดียวกันก็จะต้องมีการ Execuse หลายครั้ง ซึ่งโปรแกรมจะถูกโหลดลงสู่หน่วยความจำหลักเป็นจำนวนมาก และจะทำให้หน่วยความจำลดประสิทธิภาพลง

บทที่ 8

กรณีศึกษาการออกแบบระบบฐานข้อมูลเชิงวัตถุสัมพันธ์ในระบบงานคลังสินค้า

8.1 ระบบงานคลังสินค้า

ระบบคลังสินค้านั้นเป็นระบบย่อยระบบหนึ่ง ในระบบข้อมูลของบริษัท ร้านค้า หรือหน่วยงานองค์กรอื่นๆ ที่มีการจัดการเกี่ยวกับการซื้อหรือจำหน่ายสินค้า ซึ่งประกอบด้วยระบบย่อยต่างๆ เช่น ระบบการขาย ระบบการซื้อ การผลิต การบัญชี และอื่นๆ โดยทั่วไปนั้นระบบคลังสินค้ามักมีคลังสินค้าเป็นขององค์กรเอง การหาตัวเลขที่เหมาะสมของจำนวนสินค้าในคลังนั้น เป็นสิ่งสำคัญสำหรับผลประโยชน์ขององค์กร ดังนั้นการบริหารและควบคุมคลังสินค้าจึงเป็นส่วนหนึ่งของระบบบริหารข้อมูล ซึ่งจะทำให้เกิดประโยชน์ดังนี้ :-

1. สามารถให้ค่าของจำนวนสินค้าได้ใกล้เคียงความเป็นจริงที่สุด เพื่อไม่ให้เกิดการขาดสต็อกขึ้น ผลที่ได้ก็คือความพึงพอใจของลูกค้า
2. ต้องป้องกันการล้นสต็อก ซึ่งการเพิ่มจำนวนสินค้าในคลังจนเกินความต้องการ จะทำให้เกิดการสูญเสียค่าใช้จ่ายและเนื้อที่สำหรับวางสินค้า

การที่จะหาค่าของจำนวนสินค้าในคลังที่เหมาะสมที่สุดได้นั้น ต้องพิจารณาถึงการสั่งซื้อวัตถุดิบหรือสินค้าเข้าสต็อกว่าจะมีกี่ครั้ง และครั้งละเท่าไร ซึ่งก็คือจำนวนของสินค้าในคลังที่ควรจะต้องมีการสั่งซื้อแต่ละครั้งเวลาที่ใช้ในระหว่างการสั่งซื้อจนกระทั่งได้รับสินค้านั้น อัตราของสินค้าที่ถูกใช้ไป และจำนวนสินค้าที่ควรค้างไว้ในสต็อก เพื่อป้องกันการขาดสต็อกซึ่งเป็นสาเหตุให้เกิดการล่าช้าในการบริการลูกค้าต่อการสั่งซื้อ ณ จุดนี้ต้องมีการใช้ความรู้ทางด้านการวาดกราฟ สถิติ เลขคณิต และเทคนิคการคาดการณ์ ระบบของระยะเวลาและการสั่งซื้อจะเป็นส่วนของการเพิ่มตัวเลขในคลังสินค้า มีวิธีมากมายในการคำนวณจำนวนสินค้าในคลัง แต่ส่วนใหญ่ต้องคำนึงถึงค่าใช้จ่าย 2 ชนิด คือ

1. ค่าใช้จ่ายเมื่อเกิดการขาดสต็อกไม่มีสินค้าให้กับลูกค้า
2. ค่าใช้จ่ายเมื่อมีสินค้าล้นคลัง

จากค่าใช้จ่ายสำหรับการขาดสต็อกนั้นจะรวมถึงอัตราการเปลี่ยนผลิตซึ่งจะรวมถึงค่าใช้จ่ายที่เพิ่มขึ้นในการจัดซื้อ การขนส่ง การสูญเสียในสั่งซื้อจากลูกค้าเนื่องจากผลิตหรือหาสินค้าให้กับลูกค้าไม่ทัน ขาดความสัมพันธ์ระหว่างลูกค้า หรืออีกแง่หนึ่ง ค่าใช้จ่ายในการลงทุน ซึ่งรวมถึงค่าเช่าที่ ภาษี ประกันภัย ค่าเสียหายของสินค้าในสต็อกหรือสินค้าล้าสมัย ซึ่งล้วนแต่เป็นค่าใช้จ่ายที่ต้องสูญเสียทั้งสิ้น

คลังสินค้าหนึ่งๆ จะประกอบด้วยส่วนของการนำเข้าและส่วนของการนำออก ของรายการสินค้า ซึ่งตัวเลขจะต้องถูกแก้ไขทุกครั้งที่มีการเปลี่ยนแปลง เพื่อให้ทราบถึงจำนวนที่มีอยู่ในขณะนั้น ซึ่งข้อมูลจะได้ถูกนำไปใช้ในด้านอื่นๆ อีกด้วย

8.2 การออกแบบระบบงานควบคุมคลังสินค้า

การออกแบบระบบงานนี้สามารถแบ่งการออกแบบได้ 2 ส่วนคือ

8.2.1 การออกแบบฐานข้อมูล

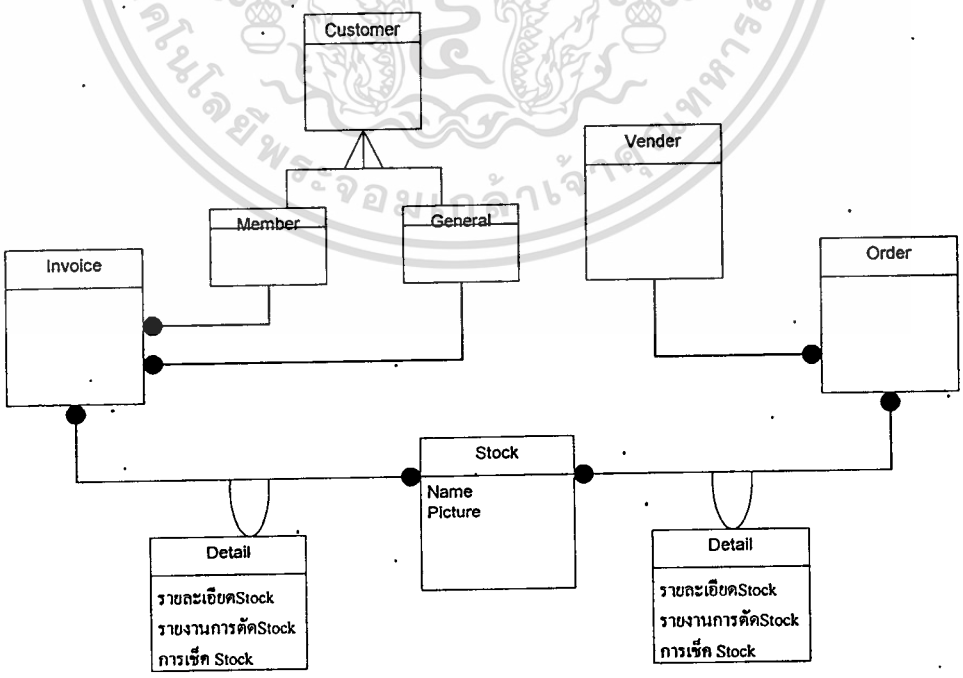
8.2.2 การออกแบบโปรแกรม HTML ที่ใช้งานผ่าน Internet

8.2.3 การออกแบบระบบงานด้วยวิธีการแบบออบเจ็กต์โมเดลติ้งโดย OMT

8.3 การออกแบบฐานข้อมูลด้วย Object Model

การออกแบบในส่วนนี้เป็นการออกแบบระบบงานคลังสินค้าโดยนำหลักการของออบเจ็กต์ออร์เรียนเต็ดมาทำการออกแบบ และใช้วิธีการออกแบบของ ออบเจ็กต์โมเดลลิ่งเทคนิค (Object Modeling Technic)

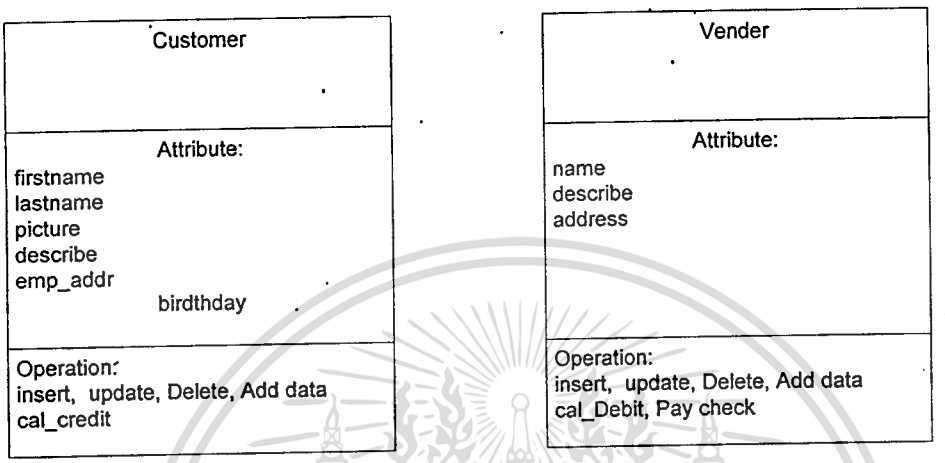
ภาพที่ 8.1



แสดงการออกแบบคลาสด้วยวิธีการของ Object Modeling

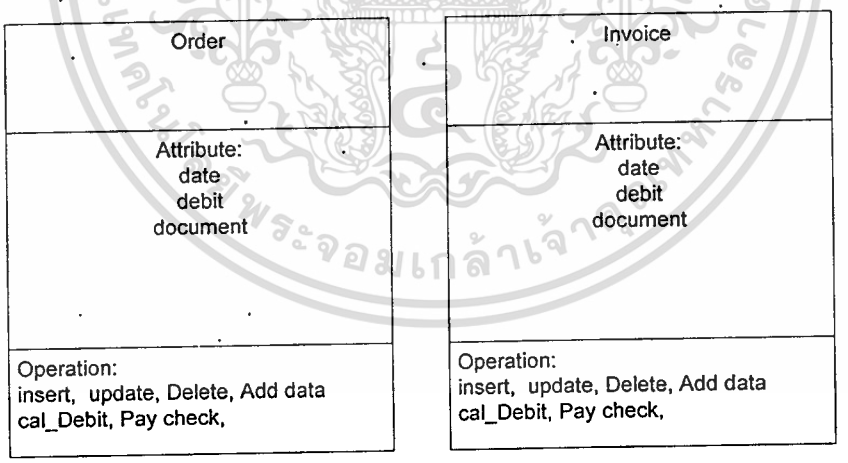
8.4 รายละเอียดของคลาสที่ได้จากการออกแบบของระบบสินค้าคงคลัง

ภาพที่ 8.2



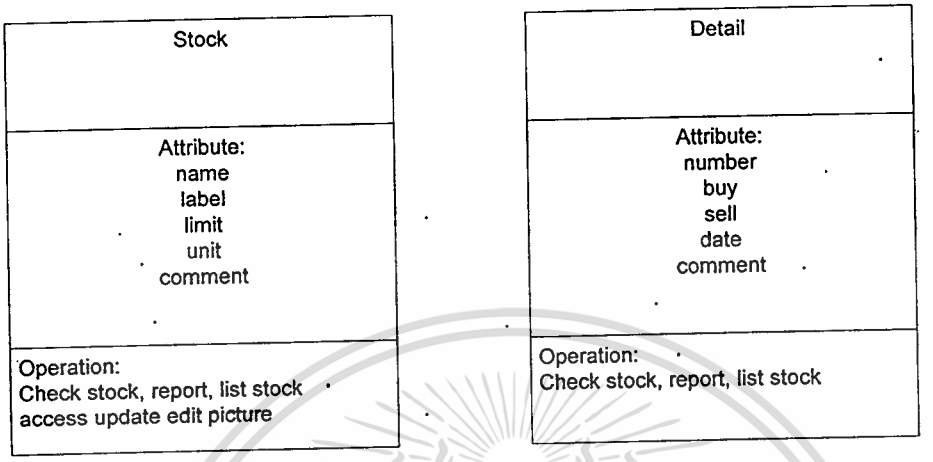
แสดงรายละเอียดของคลาส Customer และ คลาส Vender

ภาพที่ 8.3



แสดงรายละเอียดของคลาส Order และ คลาส Invoice

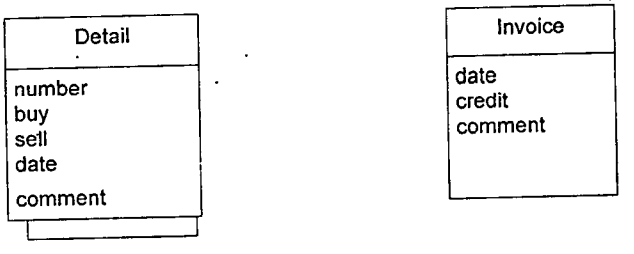
ภาพที่ 8.4



แสดงรายละเอียดของคลาส Detail และ คลาส Stock

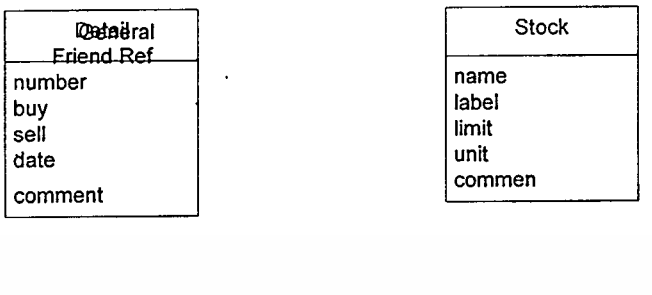
ในการออกแบบฐานข้อมูลโดยการใช้มาตราฐานของ SQL3 นั้นสามารถทำการแปลงออกมาเป็นตารางต่างๆ ได้ดังนี้

ภาพที่ 8.5



รายละเอียดของตาราง

ภาพที่ 8.5(ต่อ)



แสดงของตาราง

8.5 การออกแบบตารางด้วยการ TRANFER มาจาก CLASS จะได้ดังนี้

8.5.1. ออกแบบ ตารางเป็นแบบ Name row type คือมีการกำหนดชื่อให้กับ row type ด้วย

```
CREATE ROW TYPE name_emp
        (       firstname       char( 25 ),
                lastname        char( 25 ),
                picture         BLOB,
                describe        char( 30 ) );

CREATE ROWE TYPE address_emp
        (       number char( 10 ),
                soi     char( 20 ),
                road    char( 20 ),
                squad   char( 20 ),
                villege char( 20 ),
                aumpore         char( 20 ),
                province        char( 20 ),
                zip             integer,
```

```
        telephone       char( 11 ) );

CREATE TABLE customer
    (       cus_id          integer  UNIQUE   PRIMARY KEY,
            emp_name        name_emp,
            emp_addr        address_emp,
            emp_birdthday   date  default  today,
            emp_account     set( char(15)  not null ) );
```

ในตารางของ Customer จะมีการใช้ complex data type ในส่วนของ emp_blood ซึ่งเป็น type แบบ collection type ชนิด set type

8.5.2. ออกแบบ ตารางเป็นแบบ inheritance คือมีการสืบทอดคุณสมบัติของคลาสแม่ให้กับคลาสลูก

```
CREATE ROWE TYPE official
    (       position        char( 30 ),
            base_salary     numeric( 12,2 ) DEFUAL 0,
    ) UNDER customer;

CREATE ROWE TYPE sale
    (       base_salary     numeric( 12,2 ) DEFUAL 0.0 ,
            bonus           numeric( 12,2 ) DEFUAL 0.0
    ) UNDER customer;

CREATE ROWE TYPE member (
            credit  numeric( 12,2 ) DEFUAL 0,
            debit   numeric( 12,2 ) DEFUAL 0
                    CHECK (debit <= credit )
    ) UNDER customer;
```

8.5.3. ออกแบบ ตารางเป็นแบบ unname row type คือไม่มีการกำหนดชื่อให้กับ row type

```
CREATE TABLE inventory
    (       inv_id  integer  UNIQUE  PRIMARY  KEY,
            name    char( 30 ),
            label   char( 15 ),
            limit   float( 9,2 ),
            unit    char( 15 ),
            list_stock  ROW
                    (
                        number      numeric( 12,2 ),
                        buy         float( 10,2 ),
                        sell        float( 10,2 ),
                        date        date,
                        inv_code    char( 10 ) not null
                    ),
            comment             char( 15 ) );

CREATE ROWE TYPE detail
    (       inv_id      integer  not null,
            number      numeric( 12,2 ),
            buy         float( 10,2 ),
            sell        float( 10,2 ),
            date        date,
            inv_code    char( 10 ) not null  );


CREATE TABLE invoice
    (       invoice_id  char( 10 ) not null,
            cus_id      integer,
            date        datetime,
            credit      integer,
```

```
        comment         char( 5 ),
        detail          detail );
```

ในตารางของ invoice จะมีการ กำหนดชนิดของข้อมูลเป็นแบบ datetime คือมีการกำหนดทั้งค่าของวันที่และเวลาพร้อมกัน

ตารางทั้งหมดที่ได้รับการออกแบบคือ

```
CREATE ROW TYPE name_emp
    (   firstname       char( 25 ),
        lastname        char( 25 ),
        picture         BLOB,
        describe        char( 30 ) );

CREATE ROWE TYPE address_emp
    (   number char( 10 ),
        soi     char( 20 ),
        road    char( 20 ),
        squad   char( 20 ),
        villege char( 20 ),
        aumpore         char( 20 ),
        province        char( 20 ),
        zip             integer,
        telephone       char( 11 ) );

CREATE TABLE customer
    (   cus_id          integer  UNIQUE   PRIMARY KEY,
        emp_name        name_emp,
        emp_addr        address_emp,
        emp_birdthday   date  default  today,
        emp_account     set( char(15)  not null ) );
```

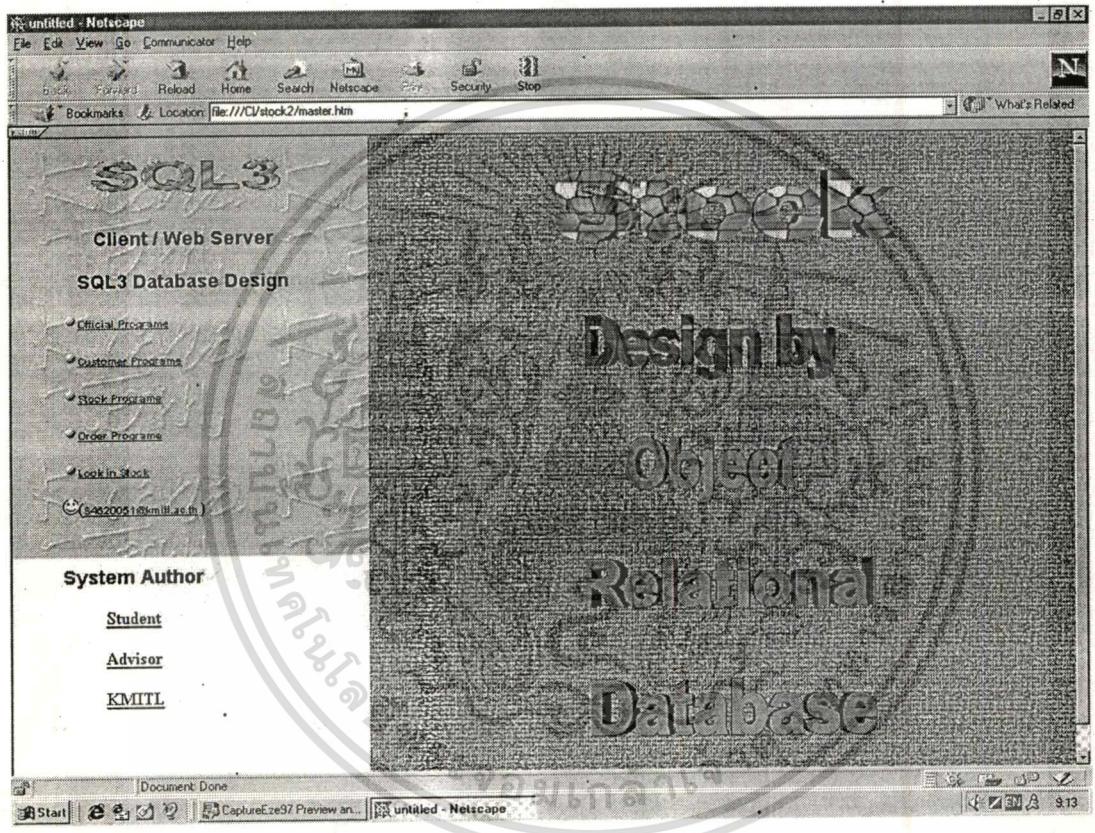
```
CREATE ROWE TYPE official
    (       position        char( 30 ),
            base_salary     numeric( 12,2 ) DEFUAL 0,
    ) UNDER customer;

CREATE ROWE TYPE sale
    (       base_salary     numeric( 12,2 ) DEFUAL 0.0 ,
            bonus           numeric( 12,2 ) DEFUAL 0.0
    ) UNDER customer;

CREATE ROWE TYPE member (
            credit    numeric( 12,2 ) DEFUAL 0,
            debit     numeric( 12,2 ) DEFUAL 0
                      CHECK (debit <= credit )
    ) UNDER customer;

CREATE TABLE inventory
    (       inv_id  integer  UNIQUE  PRIMARY  KEY,
            name    char( 30 ),
            label   char( 15 ),
            limit   float( 9,2 ),
            unit    char( 15 ),
            list_stock  ROW
                    (
                        number          numeric( 12,2 ),
                        buy             float( 10,2 ),
                        sell            float( 10,2 ),
                        date            date,
                        inv_code        char( 10 ) not null
                    ),
```

```
        comment          char( 15 ) );

CREATE ROWE TYPE detail
    (   inv_id           integer  not null,
        number           numeric( 12,2 ),
        buy              float( 10,2 ),
        sell             float( 10,2 ),
        date             date,
        inv_code         char( 10 ) not null );

CREATE TABLE invoice
    (   invoice_id       char( 10 ) not null,
        cus_id           integer,
        date             datetime,
        credit           integer,
        comment          char( 5 ),
        detail           detail  );
```

8.6 การพัฒนาระบบงานคลังสินค้าด้วย Web

ลักษณะของระบบ ที่พัฒนามีลักษณะคือ การใช้งานระบบ ผู้ใช้สามารถใช้งานระบบได้จากเครื่องคอมพิวเตอร์ทุกเครื่องที่สามารถใช้งาน Web ได้ ระบบทำงานบนบริการ Web ของอินเตอร์เน็ต จึงเลือกให้วิธีการพัฒนาระบบด้วยภาษา Java

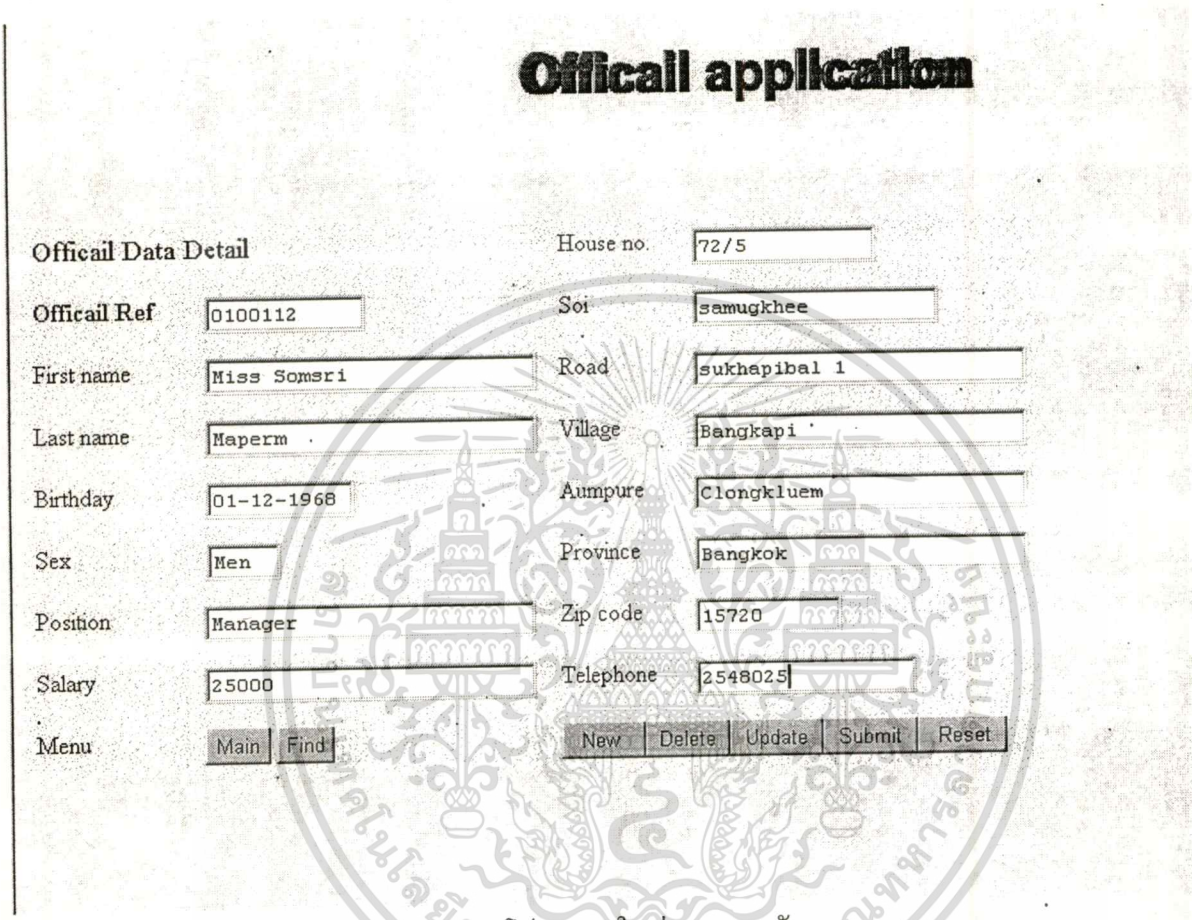
ภาพที่ 8.6



แสดงโปรแกรมระบบสินค้าคงคลังผ่านเว็บ

เป็นโปรแกรมเมนูหลักของระบบสินค้าคงคลัง ที่ใช้ฐานข้อมูลเป็นแบบระบบฐานข้อมูลเชิงออบเจ็กต์โดยการพัฒนาระบบงาน ด้วยภาษาจาวาสคริปผ่านเว็บเซิร์ฟเวอร์ไปยังเว็บเซิร์ฟเวอร์และฐานข้อมูล โดยใช้ระบบการออกแบบ แบบออบเจ็กต์โอเรียนเต็ดใช้วิธีของ OMT

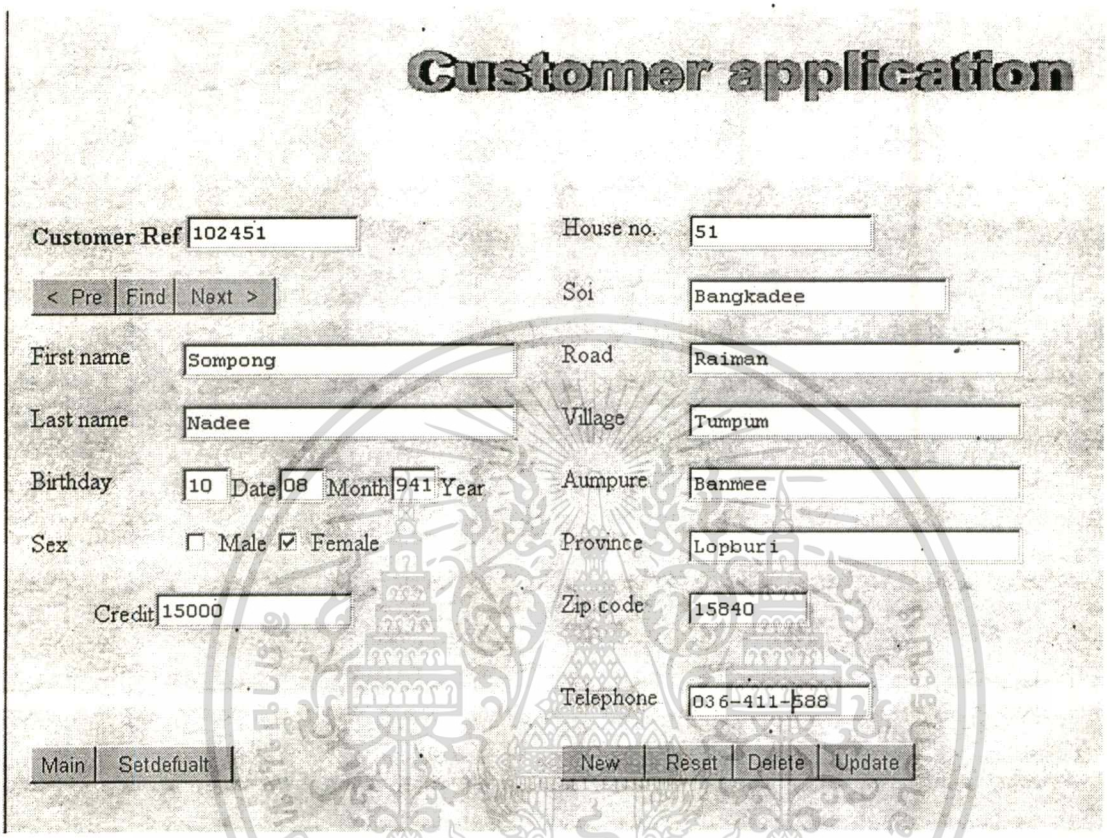
ภาพที่ 8.7



แสดงโปรแกรมในส่วนของพนักงาน

เป็นรายละเอียดของโปรแกรมพนักงานที่ประกอบไปด้วยตำแหน่ง,เงินเดือน,ที่อยู่ พนักงานจะเป็นผู้ที่คอยตรวจเช็คสินค้า ใบสั่งซื้อและจัดส่งสินค้าไปให้ลูกค้า

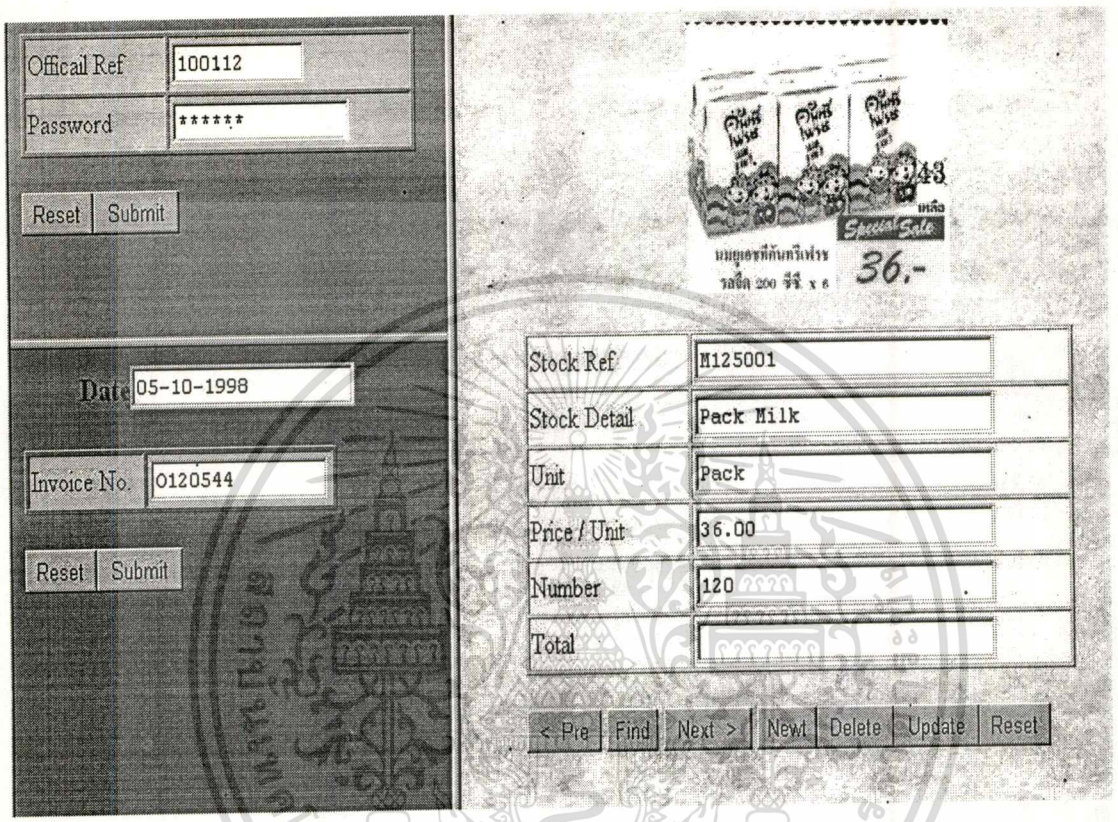
ภาพที่ 8.8



แสดงรายละเอียดของโปรแกรมลูกค้า

เป็นรายละเอียดของโปรแกรมลูกค้าที่เป็นสมาชิกรวมทั้งที่อยู่และการกำหนดยอดเงินสินเชื่อที่ให้กับลูกค้าเพื่อให้สมาชิกลูกค้าสามารถซื้อสินค้าระบบเงินเชื่อได้

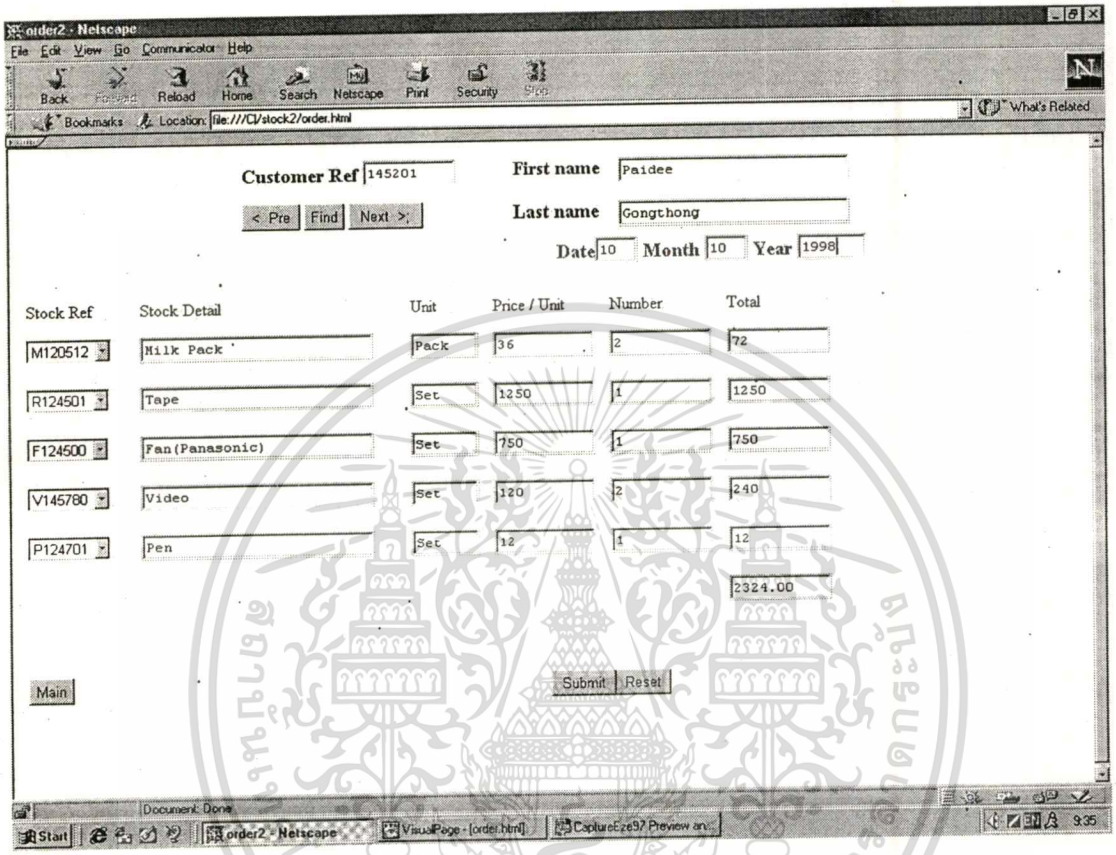
ภาพที่ 8.9



แสดงรายละเอียดของโปรแกรมสินค้า

เป็นโปรแกรมเกี่ยวกับสินค้าที่มีรายละเอียดของสินค้าคงคลัง ในการป้อนสินค้าทั้งสินค้าใหม่และสืนค้าเพิ่มรวมทั้งการตรวจเช็คสินค้าต่างๆ

ภาพที่ 8.10



แสดงรายละเอียดของโปรแกรมการขาย

ในการทำงานของโปรแกรม เมื่อได้รับใบสั่งซื้อจากสมาชิกจะได้แสดงรหัสของสินค้าขึ้นมาให้สมาชิกเลือก หลังจากเราเลือกรหัสสินค้าได้แล้วจะแสดงชื่อสินค้าและราคาขึ้นมาหลังจากนั้นจะให้ใส่จำนวนที่ต้องการ แล้วเริ่มใส่รหัสสินค้าตัวใหม่ หลังจากนั้นก็ส่งใบสั่งซื้อไป พนักงานได้รับแล้ว จะทำการตัดสต็อคและจัดส่งสินค้าไปยังผู้ซื้อ

บทที่ 9
สรุป

9.1 ข้อดีของภาพแบบข้อมูลเชิงออบเจ็กต์

9.1.1 Extensibiility สำหรับ User Define Type และ Overloaded Operations นั้น ส่วนของคลาสและการจัดการความสัมพันธ์จะสามารถปรับเปลี่ยน หรือเพิ่มได้ตามความต้องการ การเปลี่ยนแปลงหรือการเพิ่มนี้จะถูกจัดการให้กับคลาสที่มีความสัมพันธ์กันโดยอัตโนมัติ นั่นคือคลาส และโอเปอเรชั่นใหม่ๆ สามารถรวมเข้าไปในระบบ โดยปราศจากผลกระทบกับออบเจ็กต์อื่นๆ ภายในระบบ

9.1.2 Information Hiding การซ่อนข้อมูลออบเจ็กต์จะอนุญาติให้ข้อมูล และโอเปอเรชั่นถูกกำหนดอยู่ภายในโครงสร้างข้อมูลเดี่ยว (Single Model) นั่นคือ ข้อมูลจะถูกกำหนดเฉพาะที่ (Local) และถูกซ่อนอยู่ภายในออบเจ็กต์เอง เมื่อมีการเชื่อต่อเกิดขึ้น จะมีการแสดงออกมาโดยอัตโนมัติ

9.1.3 Flexibility of Type Definition เป็นความสามารถที่อนุญาตให้ผู้ใช้สร้างคลาส และโอเปอเรชั่นใหม่ๆขึ้นมาให้มีความยืดหยุ่นและการควบคุมให้กับผู้ใช้ ซึ่งเป็นประโยชน์กับการประยุกต์ใช้กับฐานข้อมูลใหม่ๆ

จากคุณสมบัติการถ่ายทอดจากคลาสแม่ไปยังคลาสลูก ทำให้ไม่ต้องมีการกำหนดวิธีการและโอเปอเรชั่นขึ้นใหม่ในแต่ละคลาส ซึ่งจะช่วยลดความซ้ำซ้อน และการทำงานที่ไม่ควบคู่กัน

9.2 การเปรียบเทียบการออกแบบแบบออบเจ็กต์โอเรียนเต็ดกับการออกแบบทั่วๆไป

การออกแบบระบบโดยใช้แนวคิดทางออปเจ็กต์ เมื่อนำไปพัฒนาระบบงานกับงานที่เป็นแบบออบเจ็กต์สามารถที่จะนำไปใช้ได้เลยโดยไม่ต้องปรับปรุงหรือเปลี่ยนแปลงใดๆ แต่ถ้าหากนำไปใช้กับระบบงานที่ไม่เป็นออบเจ็กต์แล้ว ไม่สามารถที่จะนำไปใช้ได้ทันที แต่ต้องปรับปรุงหรือเปลี่ยนแปลงข้อมูลหรือเมทธอดบางอย่าง เพื่อให้ใช้ได้กับข้อมูลหรือเมทธอดที่เหมาะสมกับระบบที่นำมาใช้ ทำให้ยังไม่ได้ใช้ความสามารถในเชิงออปเจ็กต์อย่างครบครัน เช่นข้อมูลในส่วนของ Function ของ OMT นั้น ยังใช้งานไม่ได้กับฐานข้อมูลแบบ Relational เนื่องจากฐานข้อมูลแบบ Relational ไม่มีความสามารถในการ Implement ส่วนของ Function ได้ต้องนำ Function ดังกล่าวไป Implement ในส่วนของโปรแกรมแทน

9.3 การเปรียบเทียบการใช้ Tools แบบออบเจ็กต์โอเรียนเต็ดกับ Tools ทั่วๆไป

การ Implement ด้วยเครื่องมือต่างๆ ในปัจจุบันมีเครื่องมือเป็นจำนวนมากที่เป็นเครื่องมือแบบออบเจ็กต์ โดยส่วนมากแล้วจะเป็นแบบกราฟฟิค(GUI)ทำให้การออกแบบทำได้โดยง่าย รวมทั้งคำสั่งต่างๆ ที่เครื่องมือแบบออบเจ็กต์สนับสนุนนั้นง่ายและสะดวกต่อการพัฒนาระบบงาน นอกจากนั้นยังได้มีการเพิ่มเติมและสนับสนุนรูปแบบของคำสั่งที่ติดต่อกับระฐานข้อมูลทำให้การพัฒนาระบบงานร่วมกันกับฐานข้อมูลทำได้ง่ายและมีประสิทธิภาพ นอกจากนี้เครื่องมือเหล่านี้ยังมีเทคโนโลยี่ที่มีความก้าวหน้าตลอดเวลา ทำให้เราสามารถที่จะปรุงระบบงานโดยใช้เครื่องมือใหม่ๆ ได้

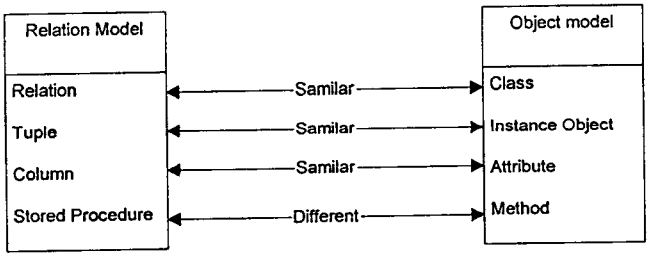
ส่วนการ Implement ด้วยระบบธรรมดานั้น ทำได้ยากทั้งในส่วนของโปรแกรมต่างๆ และเมทธอด จึงมีต้นทุนที่ใช้ในการทำงานมาก นอกจากนี้ยังเสียเวลามากอีกด้วย การพัฒนาในส่วนนี้จึงเหมาะกับผู้ที่มีความสามารถในการเขียนโปรแกรมที่มีความชำนาญ ที่สำคัญการพัฒนาบางส่วนที่ต้องการใช้คุณสมบัติของออบเจ็กต์นั้นไม่สามารถทำได้ การพัฒนาระบบงานส่วนใหญ่จะขึ้นอยู่กับผู้พัฒนาถ้ามีการขาดบุคคลากรในการพัฒนาไป ผู้พัฒนาใหม่จะต้องเสียเวลามากในการปรับปรุง

9.4 การเปรียบเทียบระหว่างฐานข้อมูลเชิงออบเจ็กต์กับฐานข้อมูลเชิงสัมพันธ์

ข้อเปรียบเทียบของ Object Oriented Model กับ Relational Model สามารถสรุปได้ดังนี้ใน 2 ทิศทางคือ

- ความเหมือน และแตกต่างทางด้านรูปแบบสามารถแสดงออกมาได้ดังรูป
- ความเหมือน และแตกต่างทางด้านแนวความคิด
  - Data Type
  - Data Integrity
  - Data Manipulation

ภาพที่ 9.1



แสดงการเปรียบเทียบระหว่างฐานข้อมูลเชิงออบเจ็กต์กับฐานข้อมูลเชิงสัมพันธ์

9.4.1 Data Type

ลักษณะของชนิดข้อมูลที่ใช้กับระบบฐานข้อมูลเชิงสัมพันธ์จะเป็นลักษณะที่เรียกว่า Builtin Data Type เช่น Intreal string เป็นต้น ซึ่งลักษณะของชนิดข้อมูลเหล่านี้ระบบจะเป็นผู้ทำการกำหนดให้ แต่สำหรับระบบฐานข้อมูลเชิงออบเจ็กต์นั้นชนิดของข้อมูลที่สามารถใช้งานได้จะเป็นทั้ง Builtin Data Type และชนิดของข้อมูลที่ผู้ใช้สามารถทำการกำหนดขึ้นมาเองได้ เช่น User Define Type ที่เป็น Complex Data Type เป็นต้น

9.4.2 Data Integrity

Relational System การทำงานลักษณะที่เรียกว่า Referential Integrity โดยการใช้การสื่อความหมายในการอ้างถึงโดยผ่าน Foreign Key แต่ในทางตรงกันข้ามระบบ OODB คลาสจะเป็นตัวบรรยายถึง Data Abstraction และรวมไปถึงการกำหนด Specific ให้กับโอเปอเรชั่นต่างๆ ที่จะถูกประยุกต์ใช้กับ Instance ของคลาสซึ่งการกำหนดเช่นนี้จะทำให้ข้อมูลมีความเป็นอิสระต่อกันมากขึ้น นั่นคือเมื่อมีการเปลี่ยนแปลง หรือมีการพัฒนาระบบจะไม่มีส่งผลกระทบต่อคลาสหรือกระบวนการทำงานอื่นๆ ซึ่งส่งผลให้ข้อมูลมีความถูกต้อง (Data Integrity)

9.4.3 Data Manipulation

การจัดการข้อมูลสามารถเปลี่ยนภาพไปอยู่ในรูปแบบของ Query Processing System ได้ซึ่งระบบนี้จะต้องให้ความสะดวกต่อการสร้าง การลบ การปรับเปลี่ยน และการเข้าถึงข้อมูลที่จำเป็นได้ สำหรับระบบ Relational DB นั้น Query Language จะใช้แนวความคิดในการจัดการข้อมูลที่ต่างกันออกไป

SQL2 นั้นจะมีพื้นฐานมาจาก Relational Calculus สำหรับการจัดการข้อมูลใน OODB นั้นกระทำโดยการเชื่อมต่อคลาส และผ่านโครงสร้างของภาษาโปรแกรมที่ล้อมรอบด้วยการส่งไปยังคลาส นั่นคือการส่งข้อความ (Message) ไปยังออบเจ็กต์ที่ทำงานด้วย ซึ่งภาพแบบของ SQL ที่ทำงานกับ OODB คือ SQL3 โดยสนับสนุนการทำงานในส่วนของ object ด้วย

9.5 สรุปการวิจัย

1. จากการวิจัยระบบฐานข้อมูลแบบ ORDB เป็นระบบฐานข้อมูลที่เป็นแบบรีเลชั่นแนล โดยมีการเพิ่มเติมคุณสมบัติของ Object Oriented ทำให้มีการใช้งานได้กว้างขวางขึ้น มีความยืดยุ่นสูง และใช้ออกแบบระบบฐานข้อมูลในรูปแบบของ Multimedia ได้ โดยรูปแบบของการเก็บข้อมูลที่เป็นแบบ Multimedia นั้น มีการเก็บข้อมูลได้หลายชนิดเช่น Music, Vedio, Picture, File ฯลฯ เนื่องจากข้อมูลที่จัดเก็บนั้นเป็นข้อมูลที่มีขนาดใหญ่และมีความละเอียดสูง ไม่สามารถที่จะจัดเก็บโดยระบบฐานข้อมูลแบบธรรมดาหรือรีเลชั่นแนลได้

2. ในการทำวิจัยนี้ได้ออกแบบฐานข้อมูลให้ทำงานได้ในระบบของ Web Server ผ่านทาง Internet ทำให้เราสามารถที่จะทำงานกับระบบในระยะทางไกลได้สะดวก โดยใช้อุปกรณ์ที่มีการใช้งานทั่วๆไปที่สามารถใช้งาน Internet ได้ทำให้ไม่จำเป็นต้องเสียค่าใช้จ่ายเพิ่มเติม

3. การออกแบบระบบฐานข้อมูลแบบ ORDB นี้ ได้นำวิธีการออกแบบของ OMT มาใช้ในการออกแบบ CLASS โดยมีมาตราฐานของข้อมูลเป็นแบบ SQL3 โดยใช้ระบบฐานข้อมูลของ Informix ที่รองรับมาตราฐานของ SQL3 มาใช้การออกแบบ ทำให้การออกแบบทำได้สะดวกและรวดเร็วได้คุณลักษณะและคุณสมบัติของระบบงานตามที่ต้องการ ในระบบฐานข้อมูลของ Informix ที่เป็น Database Server แบบ ORDB และสนับสนุนมาตราฐานข้อมูลแบบ SQL3 นั้น ตัวของระบบฐานข้อมูลเองมีการใช้งานพื้นที่ของ Hardisk น้อย แต่มีความต้องการหน่วยความจำที่เครื่อง Server ที่สูงจึงจะทำงานได้ดี นอกจากนั้น Informix ยังสามารถทำงานร่วมกันในแบบ Multi Processor ได้

4. ระบบฐานข้อมูลแบบไคลเอนต์/เซิร์ฟเวอร์ มีประโยชน์คือ ระบบไคลเอนต์/เซิร์ฟเวอร์มาจากการแบ่งแยกการจัดการระหว่างระบบไคลเอนต์และดาต้าเบสเซิร์ฟเวอร์ เนื่องจากการจัดการฐานข้อมูลถูกทำที่ส่วนแบ็กเอนต์ ความเร็วของ DBMS ไม่ได้ผูกติดกับความเร็วของเวิร์กสเตชัน ผลลัพธ์ก็คือ เวิร์กสเตชันจำเป็นต้องสามารถรันซอฟต์แวร์ฟรอนต์เอนด์เท่านั้น เป็นการขยายการใช้งานของพีซีที่เล็ก หรือเก่ากว่าที่ไม่มีความสามารถพอที่จะรัน DBMS ที่ซับซ้อนได้

การแบ่งงานเช่นนี้ ยังช่วยลดโหลดให้กับเน็ตเวิร์กสเตชันด้วย แทนที่จะต้องส่งไฟล์ฐานข้อมูลทั้งหมดไป และกลับผ่านสายแลน การจราจรบนเน็ตเวิร์กถูกลดลงเหลือแค่คิวรี และการตอบสนองจากดาต้าเบสเซิร์ฟเวอร์เท่านั้น ดาต้าเบสเซิร์ฟเวอร์บางตัวยังสามารถเก็บและรันโพรซีเดอร์และคิวรีบนตัวเซิร์ฟเวอร์เองด้วยซึ่งเป็นการลดปัญหาการจราจรลงอีก บนระบบเน็ตเวิร์กขนาดใหญ่ที่มีเวิร์กสเตชันจำนวนมา การลดปัญหาจราจรบนเน็ตเวิร์กนี้ยังเพิ่มคุณค่าให้มากกว่า เมื่อเทียบกับค่าใช้จ่ายที่ต้องใช้เพื่อเปลี่ยนไปใช้ระบบไคลเอนต์/เซิร์ฟเวอร์

ข้อดีอีกข้อหนึ่งในการแยกไคลเอนต์ออกจากเซิร์ฟเวอร์ก็คือ การไม่ขึ้นกับตัวเวิร์กสเตชัน ผู้ใช้จะไม่ถูกจำกัดอยู่ที่ระบบหรือแพลตฟอร์มหนึ่ง ในระบบ C/S เวิร์กสเตชันสามารถเป็นเครื่องพีซี IBM-Compatible เครื่องแมคอินทอช หรือเวิร์กสเตชันยูนิกซ์ หรือการรวมกันของสิ่งเหล่านี้ และสามารถทำงานบนระบบปฏิบัติการได้หลายตัว เช่น MS/PC-DOS, MS Windows, IBM OS/2 หรือ Apple's System7 ผลลัพธ์จึงนำไปสู่การไม่ขึ้นอยู่กับแอปพลิเคชั่น เวิร์กสเตชันไม่จำเป็นต้องใช้ซอฟต์แวร์แอปพลิเคชั่นฐานข้อมูลเดียวกัน ผู้ใช้ยังคงสามารถใช้ซอฟต์แวร์ที่คุ้นเคยเพื่อเข้าถึงฐานข้อมูลและผู้พัฒนาสามารถออกแบบส่วนฟรอนต์เอนด์สำหรับเวิร์กสเตชันเฉพาะ หรือสำหรับผู้ใช้เฉพาะได้

5. ฐานข้อมูลเวิร์ลไวด์เว็บ(WWW) คือจุดรวมของเทคโนโลยีในการจัดการข้อมูลของระบบฐานข้อมูลและเทคโนโลยีในการกระจายข้อมูลของอินเตอร์เน็ต

ฐานข้อมูลเวิร์ลไวด์เว็บจึงเป็นแอพพลิเคชั่น ซึ่งช่วยเพิ่มประสิทธิภาพของทั้งสองเทคโนโลยี และกำลังเป็นที่สนใจเป็นอย่างมากในปัจจุบันและมีแนวโน้มว่าฐานข้อมูลเวิร์ลไวด์เว็บจะมีการเพิ่มขึ้นในการใช้งานขึ้นอีกเป็นจำนวนมาก

แม้ว่าปัจจุบันการเชื่อมต่อระบบฐานข้อมูล เข้ากับระบบเวิร์ลไวด์เว็บยังไม่มีเครื่องมือ (Tools) ที่เป็นมาตรฐานกลางที่แน่ชัด แต่อย่างไรก็ตามผู้ผลิต ซอฟท์แวร์จำนวนมากต่างให้ความสำคัญกับเทคนิคในการเชื่อมต่อระบบฐานข้อมูลเข้ากับเวิร์ดไวด์เว็บ เป็นอย่างมาก ดังจะเห็นได้จากมี Tools ที่ช่วยในการพัฒนาระบบฐานข้อมูลเวิร์ลไวด์เว็บเป็นจำนวนมากในปัจจุบัน จึงไม่เป็นที่น่าสงสัยว่าระบบฐานข้อมูลเวิร์ลไวด์เว็บ คือเทคโนโลยีที่จะได้รับการพัฒนาและใช้งานในหน่วยงานต่างๆ ทั่วโลกอย่างแพร่หลายทั้งในปัจจุบัน และจะทวีจำนวนมากขึ้นในอนาคต

9.6 สรุปปัญหาที่เกิดขึ้น

1. ปัญหาที่เกิดขึ้นในการใช้งานคือ การทำงานผ่าน Internet ในระยะทางที่ไกลนั้นความรวดเร็วหรือความล่าช้าในการทำงานจะขึ้นอยู่กับ ระบบของ Internet ซึ่งอยู่นอกเหนือการควบคุมการทำงานของเราทำให้การทำงานของระบบเป็นไปด้วยความล่าช้า

2. การส่งผ่านข้อมูลนั้นทำได้โดยผ่าน Browser ซึ่ง Browser นั้นมีอยู่หลายชนิดแต่ละชนิดมีการใช้เทคโนโลยีที่แตกต่างกัน ทำให้ ไม่สามารถที่จะใช้กับ Browser ได้ทุกชนิดซึ่ง Browser ที่นิยมใช้มีอยู่ด้วยกัน 2 ค่ายคือ Nescape Navigrator และ Internet Exploer โดยจะต้องเป็นเวอร์ชั่นที่ใหม่ ในขณะที่เวอร์ชั่นเก่านั้นไม่สามารถนำมาใช้งานได้

3. ในการติดตั้งระบบฐานข้อมูลซึ่งใช้ Operating System เป็นแบบ Windows NT นั้น ใช้ Windows NT เวอร์ชั้น 4.0 ตัวของระบบเองต้องการ Hardware สูงคือ ต้องการเมมโมรี่(Memmory) กว่า 64 Mbyte และจะต้องเพิ่มขึ้นเมื่อมีการเพิ่มผู้ใช้จำนวนมาก ในการเก็บข้อมูล หน่วยเก็บข้อมูล(Hardisk) ควรมีความจุมากกว่า 2 Gbyte หากมีการใช้งานทางด้าน Multimedia มากๆ ควรจะเพิ่มขนาดของความจุของ Hardisk มากขึ้น

4. การรักษาความปลอดภัย ข้อที่ควรคำนึงถึงเป็นอย่างมากประการหนึ่งในการพัฒนาระบบฐานข้อมูลเวิร์ลไวด์เว็บ ก็คือ การรักษาความปลอดภัย (Security) เนื่องจากระบบฐานข้อมูลที่อยู่ในระบบฐานข้อมูลเวิร์ลไวด์เว็บ มีโอกาสที่จะถูกเข้าถึงได้จากผู้ใช้ต่างๆ มากมาย จากทั่วทุกมุมโลก

และทุกเวลา ดังนั้นการรักษาความปลอดภัยของข้อมูลที่อยู่ในระบบฐานข้อมูลเวิร์ลเวด์เว็บ จึงเป็นสิ่งที่ต้องตระหนักให้มากเมื่อทำการพัฒนาระบบฐานข้อมูลเวิร์ลไลด์เว็บ

ตามที่กล่าวไว้ข้างต้นแล้วว่าระบบฐานข้อมูลเวิร์ลไวด์เว็บเป็นระบบที่เกิดจากการรวมเอาเทคโนโลยีระบบเวิร์ลไวด์เว็บเข้ากับเทคโนโลยีของระบบฐานข้อมูล ดังนั้นฐานข้อมูลเวิร์ลไวด์เว็บจึงเป็นที่รวมของข้อดีของเทคโนโลยีทั้งสองเข้าไว้ด้วยกัน และในขณะเดียวกันในกรณีของการรักษาความปลอดภัยฐานข้อมูลเวิร์ลไวด์เว็บก็ได้รับจุดอ่อนในการเก็บรักษาความปลอดภัยของทั้งสองระบบด้วยเช่นกัน

ดังนั้นเประเด็นการรักษาความปลอดภัยของระบบฐานข้อมูลเวิร์ลไวด์เว็บจึงแบ่งออกได้เป็น 2 ส่วนใหญ่ คือ

1. การรักษาความปลอดภัยในระบบเครือข่ายคอมพิวเตอร์
2. การรักษาความปลอดภัยของระบบฐานข้อมูล

9.7 แนวทางการพัฒนาต่อ

1. นำการออกแบบโปรแกรมประยุกต์โดยใช้แนวคิดทางออปเจ็กต์ ไปใช้กับงานเก่าที่ใช้งานอยู่ เพื่อเพิ่มประสิทธิภาพและรองรับรูปแบบข้อมูลได้มากขึ้น
2. สามารถนำแนวคิดนี้มาใช้กับโปรแกรมประยุกต์ทางด้านมัลติมีเดียได้นำแนวคิดทางด้านออปเจ็กต์นี้ไปใช้งานกับโปรแกรมประยุกต์ด้านอื่นได้ เช่น โปรแกรมด้านระบบปฏิบัติงาน ระบบควบคุม ระบบการขนส่ง และอื่น ๆ

บรรณานุกรม

[1] J. Rumbaugh,M Blaha ,"OBJECT ORIENTED MODELING AND DESIGN", PRENTICE HALL 1991

[2] Ian Graham, "OBJECT ORIENTED METHODE", ADDISON WESLEY, 1994

[3] J. McNally, J, Parjesh,et all "INFORMIX UNLEASHED", SAMS PUBLISHING, 1997

[4] Piroz Mohseni, "WEB DATABASE PRIMER PLUS", WAITING GROUP PRESS, 1996

[5] Tom Sheldon , "THE WINDOWS NT WEB SERVER HANDBOOK", McGraw-HILL, 1996

[6] Informix Thailand, " INFORMIX 9.14 MANUAL ",Informix, 1997

[7] Won Kim ,"MODERN DATABASE SYSTEM", ACM PRESS, 1995

[8] Jeffrey D. Ullman, "A First Course in Database System " , PRENTICE HALL 1991

[9] Nelson M. Mallos, "An Overview of the SQL3 Standard ", IBM, 1996

[10] กิตติ ภักดีวัฒนากุลม, "สร้าง. WEB PAGE ด้วย HTML ", ห้างหุ้นส่วนจำกัดไทยเจริญการพิมพ์, 2540

[11] สุชาติ วิชาช่วย "การสร้างภาษานิยามฐานข้อมูลที่ใช้แบบจำลองเชิงแนวคิดไนแอม","การประชุมวิชาการและวิศวกรรมคอมพิวเตอร์แห่งชาติ 2541 มหาวิทยาลัยเกษตรศาสตร์",2541

[12] สุชาติ วิชาช่วย "การพัฒนาระบบงานสารสนเทศขนาดใหย่โดยใช้เครื่องมือซอฟต์แวร์ในยุคที่ 4 ","วรสารคอมพิวเตอร์แห่งประเทศไทย",2536

ภาคผนวก

มาตราฐานของ SQL3

SQL3/Foundation and SQL3/Bindings Features

This section enumerates the features that make up SQL3/Foundation (3) and SQL3/Bindings (4), over and above those of SQL-92 and SQL-92/PSM. There is no significance to the order in whichthe features appear. A summary of these features appears first, followed by a more detailed definition of the features.

SQL3/Foundation and SQL3/Bindings Features

1. Timestamps in information schema for configuration management
2. Extensions to Embedded SQL exception declarations
3. BOOLEAN data type
4. BLOB, CLOB, and NCLOB data types
5. ROW data type
6. Collection data types
7. CASCADE option for DROP COLLATION largeobjects
8. Abstract data type (ADT)
9. Abstract data type (ADT) subtypes
10. Distinct data type
11. User-defined operators
12. Updatable joined tables
13. WITH in <query expression>
14. Recursive query
15. SIMILAR predicate
16. DISTINCT predicate
17. Boolean predicat

SQL3/Foundation and SQL3/Bindings Features

25. SENSITIVE cursors
26. START TRANSACTION statement
27. LOCAL option for SET TRANSACTION statement
28. Chained transactions
29. Savepoints
30. SELECT privilege with column granularity
31. Static and Dynamic execution rights
32. Functional Dependencies
33. OVERLAY function for characters and

18. Optional interval qualifier

19. LIKE clause in table definition

20. Subtables

21. Referential action RESTRICT

22. Comparable data types for referential constraints

23.Triggers

24.WITH HOLD cursors PAGE 6

1. Timestamps in information schema for configuration management

This appears to only have been added to clause 19.3.36, "ROUTINES base table", of SQL3/Foundation. This definition uses clause 19.2.48."TIME_STAMP domain", of SQL3/Foundation.

2. Extensions to Embedded SQL exception declarations.

SQLSTATE,CONSTRAINT, SQLEXCEPTION, SQLWARNING have been added to clause 13.2,"<embedded exception declaration>", of SQL3/Bindings.

EXEC SQL WHENEVER SQLSTATE (02,001) CONTINUE;

3. BOOLEAN data type.

Boolean literals.TRUE, FALSE, and UNKNOWN.CAST extension. Boolean value expressions (IMPLIES, IFF, and IMPLIEDBY). Aggregate operators (EVERU. AMY, SOME).

4. BLOB, CLOB, and NCLOB data types.

Large object locators (holdable and not holdable). May be used in <like predicate>,<position expression>,<comparison predicate> (With an <equals operator>or<not equals operator>),or <quantified comparison predicate>(With the <equals operator>or<not equals operator>). Clause 13.12," <hold locator statement>", and clause13.13,"<free locator statement>".

5. ROW data type.

ROW reference. Field reference.

Allow ROW to be specified in a <row value constructor> in clause 7.1,"<row value constructor>", of SQL3/Foundation.

```
SELECT      *
FROM        employees
```

```
WHERE        ROW ( lname , fname)  =  ROW ( 'Cannan' , 'Steve' )
;
```

6. Collection data types.

SET , LIST, and MULTISET. Comparison of collections. TABLE (x) as a synonym for MULTISET (ROW (X)). <collection derived table>in<tablereference>.<collection reference>. Set operators (S_UNION,S_INTERSECT ,and S_EXCEPT). Collection aggregate operators (S_UNION, S-INTERSECT, and S_EXCEPT). CARDINALITY expression. ELEMENT value function. LIST value functions (SUBLIST and CONCATENATE). EMPTY as cast operand. EMPTY as a <row value constructor>element and in clause 11.9,"< default clause>", of SQL3/Foundation. SET,MULTISET, and LIST constructors.

Examples:

```
ALTER TABLE employees ADD COLUMN
        hobbies SET  (CHARACTER VARYING (20)) ;
ALTER TABLE employees ADD COLUMN
        dependents TABLE  (fname CHARACTER VARYING (20),
                           Lname  CHARACTER VARYING (20),
                           bodate DATE) ;
SELECT    COUNT (*)
FORM      TABLE  ( SELECT  S_UNION (hobbies)
                   FROM          employees
                 ) hobbies (hobby) ;
SELECT    e.lname , e.fname
FROM      employees e
WHERE     CARDINALITY  (hobbies) >= 3 ;
```

SET, MULTISET, and LIST constructors.

```
UPDATE    employees
SET       hobbies  = SET  ('Fly fishing', 'Origami' ) ;
```

A<query specification>may contain a <collection expression>.

```
SELECT    *
```

```
FROM      hobbies IN ( SECLECT   hobbies
                       FROM      employees
                       WHERE     lname = ' Cannan ' )
          dependents IN ( SELECT   dependents
                          FROM     employees
                          WHERE    lname = ' Kulkarni ' )
```

ITEM may be specified in a <query specification> or <select statement: single row>.

```
BEGIN
DECLARE   result_set MULTISET ( CHARACTER VARYING (30) ) ;
SET   result_set = ( SECLECT  ITEM lname
                     FROM     employees
                     WHERE    hair_color = ' Black ' ) ;
END ;
```

THE in <subquery> (needs to be expanded).

<collection value expression> may be specified in an IN predicate, an <exis predicate>, a <unique predicate>, or a <match predicate>.

```
SELECT   *
FROM     employees
WHERE    hobbies IN ( SELECT   hobbies
                      FROM     employees
                      WHERE    lname = ' Mattos '
                      S_INTERSECT
                      SELECT   hobbies
                      FROM     employees
                      WHERE    lname = ' Richie '
                    ) ;
```

7. CASCADE option for DROP COLLATION.

<drop behavior> may be specified in clause 11.37, "<drop collation statement>", of SQL3/Foundation.

```
DROP  COLLATION  latinl_insensitive  CASCADE ;
```

8. Abstract data type (ADT).

Clause 11.45, "<abstract data type body>", of SQL3/Foundation, without the specification of <subtype clause>. CREATE and DROP , ordering clause, operator name list, CAST, encapsulation levels, constructors, attribute reference. ADT locators (holdable and not holdable). Component reference

Clause 11.47, "<drop data type statement>", of SQL3/Foundation (also appears in DISTINCT data type).

9. Abstract data type (ADT) subtypes.

UNDER privilege. TREAT subtype as supertype.

<genralized expression> in clause 10.4, ">routine invocation>", of SQL3/Foundation.

TYPE predicate as specified in clause 8016, "<type predicate>", of SQL3/Foundation.

RESULT may be specified for a <parameter declaration> in clause 12.5, "<SQL-invokedroutine>", of SQL3/Foundation.

10. Distinct data type.

Specified in clause 11.46, "<distinct type definition>", of SQL3/Foundation.

CREATE DISTINCT TYPE weight_lbs AS DECIMAL (3) ;

ALTER TABLE employees ADD COLUMN weight weight_lbs ;

Clause 11.47, "<drop data type statement>", of SQL3/Foundation (also appears in ADT data type).

11. User-defined operators.

Clause 11.48, "<operators definition>", of SQL3/Foundation. <add operators definition> in clause 11.2, "<alter schema statement>", of SQL3/Foundation.

12. Updatable joined tables.

SR 12)d and 12)e of 7.9, "<query specification>", of SQL-92 do not appear to have corresponding rules in SQL3/Foundation.

d) if the <table expression> immediately contained in QS immediately contains a <where clause> WC, then no leaf generally underlying table of QS shall be a generally underlying of ant <query expression> contained in WC,

e) The <table expression> immediately contained in QS does not include a <group by clause> or <having clause>.immediately contained in QS does not include a <group by clause> or a <having clause>.

13. WITH in <query expression>

<with clause> when RECURSIVE has not been specified.

```
SELECT    *
FROM      ( WITH top_east_coast_sales AS
                (SELECT   *
                 FORM     employees_east
                 WHERE    job = ' SALES '
                   AND    compensation > 250000 ),
              top_west_coast_sales AS
                ( SELECT   *
                  FROM     employees_west
                  WHERE    job = 'SALES '
                    AND    compensation > 250000 )
            top_east_coast_sales  UNION  top_west_coast_sales
          ) AS top_sales  ;
```

14. Recursive query.

<with clause> or <view definition> when RECURSIVE has been specified.

```
WITH RECURSIVE Reachable_From  (Source, Destin ) AS
        ( SELECT      Source, Destin
                      FROM          Flights
                      WHERE         Source = ' Paris '
                      UNION
                            SELECT      in.Source,  out.Destin
                            FROM        Reachable_From in,  Flights out
                            WHERE       in.Destin = out.Source
        )
SELECT  *  FROM Reachable_From
```

Functional Dependencies.

Keys in Descriptor Area.

33. OVERLAY function for characters and large objects.

<character overlay function>and<blob overlay function> in clause 6.10, " <string value function>", of SQL3/Foundation.

```
SELECT              '+1. '
                OVERLAY ( phone number
                        PLACING '.'
                          FROM  4
                          FOR 1  )
                PLACING '.'
                FROM 8
                FOR  1
FROM    employees  ;
```

34. SQL-invoked routines.

SQL-paths. CURRENT_PATH value spectification. CURRENT_PATH in clause 11.9 <default clause>", of SQL3/Foundation. Procedures and function. SQL routine and external routine. Polymorphism. Subject-roytine determinatiion. Return statement. EXECUTE privilege. Clause 11.49 <drop routine statement>.

35. Roles.

Clause 11.51, "<role definition>", of SQL3/Foundation,11.52, "<grant role statement>", of SQL3/Foundation, 11.53, "<revoke role statement>", of SQL3/Foundation. Grant and revoke of WITH ADMIN OPTION. Clause 17.3,"<set role statement>", of SQL3/Foundation.

```
CREATE ROLE payroll ;
GRANT  PAYROLL To vickers WITH ADMIN OPTION ;
GRANT UPDATE (salary) ON TABLE employees  To payroll ;
...
SET ROLE payroll  ;
```

36. Upper SQL Flagging.

37. Bracketed comments.

<bracketed comment>in clause 5.2, "<token>and<separator>", of SQL3/Foundation.

/* This is a comment */

38. User-defined aggregate operators.

<routine name> may be specified for <set function type> in clause 6.8,"<set function specification>", of SQL3/Foundation.

39. Quantified predicate.

Existential and universal quantification may be specified in clause 8.13,"<quantified predicate>", of SQL3/Foundation.

```
SELECT      *
FROM        employees e
WHERE               FOR SOME hobby IN e.hobbies (hobby LIKE '0%') ;
```

40. ALL SCHEMA PRIVILEGES.

ALL SCHEMA PRIVILEGES may be specified for <privileges> in clouse 10.5." <privileges>", in SQL3/Foundation.

41. Value expression in order by clause.

Removal of the SQL-92 restriction that<sort key> must be a <column name> or an <unsigned integer>.

```
SELECT          *
FROM                employees
ORDER BY CARDINALITY  (dependents) ;
```

42. Table name not required in <delete statement: positioned>or<update statement: positioned>.

Remove the SQL-92 requirement that "FROM<table reference>" be specified in clause 13.6," <delete statement: positioned>", of SQL3/Foundation and clause 13.9,"<update statement: positioned>", of SQL3/Foundation.

```
DECLARE curs1 CURSOR FOR SELECT  *  FROM employees  ;
OPEN curs1  ;
FETCH curs1 INTO  ... ;
UPDATE
SET             status  = "Sabbatical'
WHERE CURRENT OF curs1 ;
```

```
FETCH curs1 INTO ... ;
DELETE WHERE CURRENT OF curs1 ;
CLOSE curs1 ;
```

43. INSERT into a cursor.

A cursor name may be specified instead of a table name in clause 13.8."<insert statement>", of SQL3/Foundation.

```
DECLARE curs1 CURSOR FOR SELECT * FROM employees ;
OPEN curs1 ;
FETCH curs1 INTO .. ;
INSERT INTO CURSOR curs1VALUES ('Darwen','Hugh',...) ;
CLOSE curs1;
```

44. ROW may be specified in an update statement.

ROW may be specified for an <update target> in clause 13.9,"<update statement: positioned>", of SQL3/Foundation.

```
DECLARE curs1 CURSOR FOR SELECT * FROM employees ;
OPEN curs1 ;
FETCH curs1 INTO ..;
UPDATE      employees
SET         ROW = (...)
WHERE CURRENT OF curs1:
CLOSE curs1 ;
```

ประวัติผู้เขียน

นายสุชาติ วิชาช่วย เกิดเมื่อวันที่ 24 มิถุนายน 2501 จบการศึกษาระดับมัธยมศึกษาจากโรงเรียนพิบูลวิทยาลัย จังหวัดลพบุรี ระดับ ปวช. จากวิทยาลัยเทคนิคลพบุรี สาขาวิชาอิเล็คทรอนิกส์ ระดับปวส. จากเทคโนโลยีวิทยาเขตเทคนิคตาก สาขาอิเล็กทรอนิกส์ และระดับปริญญาตรี จากสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ครุศาสตร์อุตสาหกรรมบัณฑิต ภาควิชาครุศาสตร์อุตสาหกรรม สาขาวิศวกรรมโทรคมนาคม คณะครุศาสตร์อุตสาหกรรมและวิทยาศาสตร์ ในปีพ.ศ. 2531 เข้าศึกษาต่อในหลักสูตรปริญญาวิศวกรรมศาสตร์มหาบัณฑิต สาขาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์ เมื่อ พ.ศ. 2534 ปัจจุบันรับราชการครู ตำแหน่งอาจารย์ 2 ระดับ 6 สังกัดวิทยาลัยเทคนิคลพบุรี กองวิทยาลัยเทคนิค กรมอาชีวศึกษา กระทรวงศึกษาธิการ