การพัฒนาไคลเอนต์/เซิร์ฟเวอร์ชนิด 3-เทียร์
3-TIER CLIENT/SERVER DEVELOPMENT

นาย ธนากร ชวลิตมณเฑียร
นาย นิรันดร์ ลัคนากุล

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2542

การพัฒนาไคลเอนต์/เซิร์ฟเวอร์ชนิด 3-เทียร์
3-TIER CLIENT/SERVER DEVELOPMENT

โดย
นาย ธนากร ชวลิตมณเฑียร
นาย นิรันดร์ ลัดนากุล

อาจารย์ที่ปรึกษา
ดร.วรวัฒน์ ลิ้มโภคา

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2542

ปริญญานิพนธ์ปีการศึกษา 2542

ภาควิชา วิศวกรรมคอมพิวเตอร์

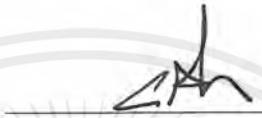
คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การพัฒนาไคลเอนต์/เซิร์ฟเวอร์ชนิด 3-เทียร์

3-TIER CLIENT/SERVER DEVELOPMENT

ผู้จัดทำ

1. นาย ธนากร ชวลิตมณเฑียร รหัสประจำตัว 39014215
2. นาย นิรันดร์ ลัดนากุล รหัสประจำตัว 39014267



อาจารย์ที่ปรึกษา

(ดร.วรวัฒน์ ลิ้มโภคา)

การพัฒนาไคลเอนต์/เซิร์ฟเวอร์ชนิด 3-เทียร์

นายธนากร ชวลิตมณเฑียร 34014215
นายนิรันดร์ ลัคนากุล 39014267
ดร. วรวัฒน์ ลิ้มโภคา อาจารย์ที่ปรึกษา
ปีการศึกษา 2542

บทคัดย่อ

ในปัจจุบันระบบคอมพิวเตอร์ชนิดแยกส่วนผู้ใช้บริการ/ผู้ให้บริการหรือที่เรียกกันว่าระบบไคลเอนต์/เซิร์ฟเวอร์ กำลังมีบทบาทมากในการสร้างแอปพลิเคชั่นของระบบการทำงาน แต่ด้วยข้อเสียบางประการ เช่น การแก้ไขแอปพลิเคชั่นยากและความสามารถของเครื่องคอมพิวเตอร์มีจำกัดทำให้เกิดแนวความคิดในการพัฒนาแอปพลิเคชั่นโดยใช้สถาปัตยกรรมแบบไคลเอนต์/เซิร์ฟเวอร์ชนิด 3-เทียร์ ซึ่งมีข้อดีในด้านการใช้ระบบเครือข่ายมีประสิทธิภาพมากขึ้นและสามารถลดค่าใช้จ่ายในการบำรุงรักษาระบบ นอกจากนี้ยังมีความสามารถในการจัดการฐานข้อมูลระบบกระจายได้อีกด้วย

ปริญญานิพนธ์ฉบับนี้เป็นการนำเสนอแนวทางวิเคราะห์ ออกแบบตามแนวความคิดเชิงวัตถุและพัฒนาระบบตามแบบสถาปัตยกรรมไคลเอนต์/เซิร์ฟเวอร์ชนิด 3-เทียร์โดยใช้เทคโนโลยี COM ที่พัฒนาขึ้นมาโดยวิชวลเบสิก 6.0 และ MTS (Microsoft Transaction Server) ซึ่งจะทำงานติดต่อกับระบบจัดฐานข้อมูลเชิงวัตถุสัมพันธ์ Oracle 8.0.5 ในโครงงานนี้ได้พัฒนาระบบการพาณิชย์อิเล็กทรอนิกส์ขึ้นมาเพื่อทดสอบการทำงานของระบบไคลเอนต์/เซิร์ฟเวอร์ชนิด 3-เทียร์

3-Tier Client/Server Development

Mr. Thanakorn Chawalitmontien
Mr. Niran Luckcanakul
Dr. Worawat Limpoka Advisor

ABSTRACT

Nowadays, The computer system called client/server system becomes important in application development. Because of its problem such as maintainability and limitation of computer's performance the concept of 3-tier client/server architecture was introduced widely. The benefit of this architecture is using network bandwidth efficiency, reducing cost in maintenance and capability to manage distributed database.

This thesis is concerned with analysis and design using object-oriented concept, developing 3-tier client/server application using COM (Component Object Model) technology and MTS (Microsoft Transaction Server) connected to Object-relational database management system (ORDBMS) such as Oracle 8.0.5. In this project we implemented an electronic commerce system to test how 3-tier client/server system work.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้คงไม่อาจเสร็จได้ด้วยดี หากไม่ได้รับความช่วยเหลือ และร่วมมือจากหลาย ๆ ฝ่ายด้วยกัน บุคคลแรกที่ต้องกล่าวถึงเพราะเป็นส่วนสำคัญที่ทำให้วิทยานิพนธ์นี้เสร็จลงได้ก็คือ อาจารย์ วรวัฒน์ ลิ้มโภคา อาจารย์ที่ปรึกษาปริญญานิพนธ์ ที่ให้ความเอาใจใส่ แนะนำ และช่วยเหลือเสมอมา ซึ่งต้องขอขอบพระคุณเป็นอย่างมาก

ขอบคุณเพื่อนทุกคน ที่ห้องปฏิบัติการฐานข้อมูลและออบเจ็กโอเรียนเท็ดที่ร่วมให้ความช่วยเหลือในการทำงาน เพื่อนๆ ห้อง 4 ที่คอยให้กำลังใจ

และต้องขอขอบพระคุณบุคคลสำคัญที่สุดที่ทำให้ข้าพเจ้ามีวันนี้ ก็คือ บิดา มารดา อันเป็นที่เคารพรักยิ่ง ซึ่งได้เลี้ยงดูผู้เขียนมาเป็นอย่างดี พร้อมทั้งให้โอกาสในการศึกษาอย่างเต็มที่ และยังให้กำลังใจเอาใจใส่เสมอมา ในทุก ๆ ด้านอันหาที่เปรียบมิได้ ข้าพเจ้าขอระลึกในพระคุณอันสุดประมาณ และขอกราบขอบพระคุณมา ณ ที่นี้

ธนากร ชวลิตมณเฑียร

นิรันดร์ ลัคนากุล

สารบัญ

สารบัญตาราง

สารบัญรูปภาพ

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มา

ในปัจจุบันระบบคอมพิวเตอร์ชนิดไคลเอนต์/เซิร์ฟเวอร์ได้รับความนิยมมากขึ้นเนื่องจากความสามารถในการนำเสนอระบบเชื่อมต่อกับผู้ใช้ที่ง่ายต่อการใช้ รวมถึงประสิทธิภาพในการให้บริการจัดการข้อมูลที่สูงขึ้นโดยเสียค่าใช้จ่ายน้อยลง นอกจากนี้ยังมีความสามารถในการจัดการฐานข้อมูลแบบกระจายได้อีกด้วย

ส่วนประกอบของแอปพลิเคชั่นหนึ่งๆประกอบด้วย 7 ส่วน ได้แก่ ส่วนให้บริการแสดงผลข้อมูล (Presentation Services) ส่วนตรรกะในการแสดงผล (Presentation Logic) ส่วนตรรกะในการจัดการแอปพลิเคชั่น (Application Logic) ส่วนตรรกะในการจัดการข้อมูล (Data Logic) ส่วนให้บริการข้อมูล (Data Services) หรือระบบจัดการฐานข้อมูล (DBMS : Database Management System) ส่วนให้บริการแฟ้มข้อมูล (Files Services) และส่วนกายภาพของฐานข้อมูล (Physical Database)

ในสมัยเริ่มแรกระบบคอมพิวเตอร์จะเป็นระบบแบบรวมศูนย์การประมวลผล (Centralized Host System) ระบบชนิดนี้จะปฏิบัติงานตั้งแต่ส่วนตรรกะในการแสดงผลจนถึงส่วนให้บริการฐานข้อมูล โดยระบบจะประกอบด้วยเครื่องคอมพิวเตอร์ที่ศูนย์กลางและมีอุปกรณ์เทอร์มินอลประสิทธิภาพต่ำซึ่งทำหน้าที่ให้บริการแสดงผลข้อมูลอย่างเดียว (Dumb Terminal) ระบบคอมพิวเตอร์ชนิดนี้เรียกอีกอย่างว่าสถาปัตยกรรมแบบส่วนเดียว (Single-tier architecture) ปัญหาของระบบคอมพิวเตอร์แบบนี้ได้แก่ประการแรกคือความล่าช้าในการส่งข้อมูลระหว่างคอมพิวเตอร์และเครื่องเทอร์มินอล ประการที่สองคือส่วนแสดงผลใช้งานยาก

ต่อมาเมื่อเครื่องคอมพิวเตอร์ส่วนบุคคล (PC : Personal Computer)ได้ถูกพัฒนาให้มีความสามารถมากขึ้นจึงเกิดระบบคอมพิวเตอร์ที่ย้ายการทำงานส่วนตรรกะในการแสดงผลไปไว้ที่เครื่องคอมพิวเตอร์ส่วนบุคคล ส่วนหน้าที่อื่นยังคงทำงานที่คอมพิวเตอร์ศูนย์กลาง ระบบคอมพิวเตอร์ชนิดนี้เรียกอีกอย่างว่าสถาปัตยกรรมแบบสองส่วน (Two-tier architecture)

เทคโนโลยีทางด้านคอมพิวเตอร์พัฒนาไปอย่างรวดเร็วทำให้ประสิทธิภาพของคอมพิวเตอร์ไม่ถูกใช้อย่างเต็มที่จึงได้เกิดแนวความคิดในการย้ายงานในส่วนตรรกะในการจัดการแอปพลิเคชั่นบางส่วนไปไว้ฝั่งตอมพิวเตอร์ส่วนบุคคล แต่ยังคงงานในส่วนตรรกะในการจัดการแอปพลิเคชั่นบางส่วนที่คอมพิวเตอร์ศูนย์กลาง ระบบคอมพิวเตอร์ชนิดเรียกว่าระบบไคลเอนต์/เซิร์ฟเวอร์(Client/Server architecture) ปัญหาของระบบแบบนี้คือการเพิ่มภาระในการประมวลผลย่อยบนเครื่องคอมพิวเตอร์ส่วนบุคคลทำให้เปลี่ยนแปลงแอปพลิเคชั่นยาก นอกจากนี้ความสามารถของเครื่องไคลเอนต์ยังจำกัดด้วย

จากปัญหาของกล่าวจึงได้เกิดระบบไคลเอนต์/เซิร์ฟเวอร์ชนิด 3-เทียร์ (Three-tier architecture)ซึ่งสามารถแก้ปัญหาของระบบแบบก่อนโดยแยกระบบคอมพิวเตอร์ออกเป็น 3 ส่วนใหญ่ ส่วนแรกคือส่วน

แสดงผลออกจากแอปพลิเคชั่นประกอบด้วยส่วนให้บริการแสดงผลข้อมูลและส่วนตรรกะในการแสดงผล ส่วนที่สองคือส่วนโปรแกรมแอปพลิเคชั่นประกอบด้วยส่วนตรรกะในการจัดการแอปพลิเคชั่นและส่วนตรรกะในการจัดการข้อมูล ส่วนที่สามคือส่วนจัดการฐานข้อมูลประกอบด้วยส่วนให้บริการข้อมูลหรือระบบจัดการฐานข้อมูลและส่วนให้บริการแฟ้มข้อมูลรวมทั้งส่วนกายภาพของฐานข้อมูล

การพัฒนาแอปพลิเคชั่นไคลเอนต์/เซิร์ฟเวอร์ชนิด 3-เทียร์มีแนวทางการพัฒนาได้หลายแนวทาง ยกตัวอย่างเช่น พัฒนาโดยใช้เทคโนโลยี COM (Component Object Model) ร่วมกับ MTS (Microsoft Transaction Server) หรือพัฒนาโดยใช้เทคโนโลยี DCOM (Distributed COM) เป็นต้น แต่ละวิธีจะมีรายละเอียดต่างกันซึ่งจะได้กล่าวถึงต่อไป

1.2 วัตถุประสงค์ของงานวิจัย

1.2.1 ศึกษาและวิเคราะห์หลักการทำงานของระบบที่มีสถาปัตยกรรมแบบ 3-เทียร์

1.2.2 ศึกษาและวิเคราะห์หลักการเขียนโปรแกรมแบบออบเจ็กต์โอเรียลเต็ดและการนำไปใช้

1.2.3 ศึกษาแนวคิดของยูเอ็มแอล (UML :Unified Modeling Language) เพื่อนำไปใช้ในการวิเคราะห์และออกแบบระบบ

1.2.4 ศึกษาการออกแบบและการสร้างระบบฐานข้อมูลเชิงวัตถุสัมพันธ์ (Object-relational Database Management System)

1.2.5 ศึกษาและพัฒนาแอปพลิเคชั่นด้วยวิชวลเบสิก 6 (Visual Basic 6)

1.2.6 ศึกษาและวิเคราะห์แนวทางในการสร้างแอปพลิเคชั่นของระบบที่มีสถาปัตยกรรมแบบ 3-เทียร์

1.3 ขอบเขตของงานวิจัย

งานวิจัยนี้จะสร้างระบบพาณิชย์อิเล็กทรอนิกส์ขึ้น ลักษณะของเว็บไซต์จะเป็นตลาดซื้อขายสินค้า โดยลูกค้าสามารถทำการตั้งร้านที่จะขายสินค้า เลือกรูปแบบของร้านค้า ตั้งชื่อร้าน กำหนดประเภทของสินค้าที่จะขายและกำหนดรายละเอียดของสินค้าได้ ลูกค้าที่เข้ามาเพื่อซื้อสินค้าสามารถค้นหาสินค้าที่ตรงกับความต้องการ เลือกสินค้าที่ต้องการเก็บลงในกระเป๋าสินค้า สามารถบันทึกรายการสินค้าในกระเป๋าสินค้าเพื่อทำการสั่งซื้อภายหลังได้ อีกทั้งยังสามารถติดตามความคืบหน้าในการสั่งซื้อสินค้าโดยผ่านทางเว็บไซต์,ทางอีเมล์หรือทางเพจเจอร์ ระบบสามารถแบ่งการทำงานออกเป็น 2 ส่วนใหญ่คือ ระบบส่วนการทำงานส่วนหน้า (Front end) และระบบส่วนการทำงานส่วนหลัง (Back end) ระบบส่วนการทำงานส่วนหน้าคือส่วนที่ติดต่อกับผู้ใช้ซึ่งได้แก่เว็บไซต์ที่แสดงรายละเอียดของร้านค้า รายละเอียดของสินค้า ส่วนมัลติมีเดียต่างๆ สำหรับระบบส่วนการทำงานส่วนหลังคือส่วนที่จัดการข้อบังคับที่บนแอปพลิเคชั่นเซิร์ฟเวอร์และจัดการระบบฐานข้อมูล เช่น การเพิ่ม ลบ ค้นหาและแก้ไขข้อมูล ขอบเขตของโครงงานนี้คือพัฒนาส่วนระบบการทำงานส่วนหลังเพื่อรองรับการใช้งานจากส่วนการทำงานส่วนหน้าซึ่งถูกพัฒนาโดยโครงงานการพัฒนาซอฟท์แวร์เชิงคอมโพเนนต์

ในการออกระบบการทำงานส่วนหลังจะใช้แนวคิดของยูเอ็มแอลมาช่วยในการวิเคราะห์และออกแบบ ระบบจะถูกสร้างขึ้นตามแบบสถาปัตยกรรมแบบ 3-เทียร์โดยใช้เทคโนโลยี COM และ MTS ซึ่งจะสนับสนุนหลักการพัฒนาแบบออบเจ็กต์โอเรียลเต็ด ระบบสร้างขึ้นเป็นคอมโพเนนต์ย่อยๆทำงานร่วมกันเพื่อทำให้สามารถนำกลับมาใช้ใหม่ (reusable) ส่วนระบบจัดการฐานที่เลือกใช้จะต้องเป็นระบบจัดการฐานข้อมูลเชิงวัตถุสัมพันธ์ซึ่งได้แก่ Oracle 8

โครงงานนี้ยังถือว่าเป็นโครงการที่ทดลองสร้าง เพื่อศึกษาความเป็นไปได้ในเทคโนโลยีที่นำมาใช้งาน ดังนั้นจึงมีข้อจำกัดบางอย่างของระบบ เช่น เวลาที่ใช้ในการติดต่อกับระบบจัดการฐานข้อมูลและประสิทธิภาพของคอมโพเนนต์ เช่น หากต้องการให้คอมโพเนนต์สามารถทำงานได้อย่างรวดเร็วต้องผ่านพารามิเตอร์แทนการเรียกใช้เมธอด แต่อย่างไรก็ตามระบบก็ยังมีความสามารถเพียงพอต่อการทดสอบอย่างแน่นอน

1.4 วิธีการดำเนินงาน

งานวิจัยในโครงงานนี้จะเริ่มด้วยการศึกษาทฤษฎีพื้นฐานต่าง ๆ ที่เกี่ยวข้องกับงานวิจัย ซึ่งก็มีเรื่องหลัก ๆได้แก่ สถาปัตยกรรมไคลเอนต์/เซิร์ฟเวอร์แบบ 3-เทียร์ วิธีการวิเคราะห์และออกแบบระบบเชิงวัตถุตามแนวคิดของยูเอ็มแอล การพัฒนาโปรแกรมเชิงวัตถุ เทคโนโลยี COM และ DCOM การนำ MTS มาใช้ร่วมกับคอมโพเนนต์ COM และระบบจัดการฐานข้อมูลเชิงวัตถุสัมพันธ์

หลังจากนั้นจะเข้าสู่ขั้นออกแบบและพัฒนาระบบพาณิชย์อีเล็กทรอนิกส์ โดยในบทที่ 6 จะกล่าวถึงคำอธิบายปัญหา (Problem Statement) ยูสเคส ซีนาริโอของระบบ โครงสร้างออบเจ็กต์ โครงสร้างคลาส การทำงานของคลาสในส่วนการทำงานส่วนหลังของคอมโพเนนต์ต่างๆ โครงสร้างและข้อกำหนดในฐานข้อมูล การจัดการเรื่องทรานแซคชั่น การกำหนดข้อบังคับในแอบพลิเคชันเซอร์เวอร์และการเตรียมความพร้อมก่อนการรวมกับคอมโพเนนต์ของระบบการทำงานส่วนหน้า

ขั้นตอนต่อมาคือการทดสอบระบบรวมตามหน้าที่ของแต่ละคอมโพเนนต์ ทดสอบการจัดการทรานแซคชั่นแบบทูเฟสคอมมิต ทดสอบการจัดการทรานแซคชั่นกับตารางชนิดเนสเต็ดและทดสอบทดสอบตามการทำงานพื้นฐาน ขั้นตอนสุดท้ายก็จะเป็นการสรุปการทำงาน ผลที่ได้รับจากงานวิจัยชิ้นนี้และแนวทางในการพัฒนางานวิจัยนี้เพิ่มเติม และแนวทางในการนำไปประยุกต์ใช้

บทที่ 2
การพัฒนาโปรแกรมเชิงวัตถุ

2.1 ความหมายของออบเจ็กต์โอเรียนเต็ด (Object-Oriented)

เป็นวิธีการพัฒนาระบบโดยใช้แนวคิดที่ว่ามองระบบเสมือนเป็นวัตถุต่างๆ มาประกอบกันและทำงานร่วมกันโดยสนใจสิ่งที่ป้อนเข้าไปในระบบและสิ่งระบบตอบสนองมา จากหลักการของแนวคิดนี้ทำให้มีการแยกวิธีการทางออบเจ็กต์โอเรียนเต็ดออกเป็น 3 รูปแบบ ดังนี้

1. การวิเคราะห์ (Analysis)
2. การออกแบบ (Design)
3. การโปรแกรม (Program)

โดยสองส่วนแรกมักถูกเรียกรวมกันว่า การวิเคราะห์และออกแบบเชิงวัตถุ (Object-Oriented Analysis and Design) ส่วนขั้นตอนของการโปรแกรมจะถูกเรียกว่า การโปรแกรมเชิงวัตถุ (Object-Oriented Programming) ซึ่งทั้ง 3 ขั้นตอนนี้จะถูกนำมาอธิบายในหัวข้อต่อไปภายในบทนี้

2.1.1 ความแตกต่างระหว่างวิธีการทางออบเจ็กต์ (Object-Oriented Approach) กับวิธีการทางโครงสร้าง (Structured Approach)

2.1.1.1 วิธีการทางโครงสร้าง

เป็นวิธีในการพัฒนาระบบในสมัยก่อน โดยมี 3 รูปแบบคือ การวิเคราะห์ การออกแบบ และการโปรแกรม โดยใน

- ส่วนของการวิเคราะห์และออกแบบจะนำการออกแบบบนลงล่าง (Top-Down Design) การใช้แผนผังการไหลของข้อมูล (Data Flow Diagram) และการนำแนวคิดของการออกแบบแยกส่วน (Modula Design) มาช่วยในการทำงาน ทำให้มีความเข้าใจในระบบที่จะทำการออกแบบได้ดียิ่งขึ้น
- ส่วนของการโปรแกรม ได้ใช้โครงสร้างข้อมูล (Data Structure) ซึ่งมีการกำหนดขอบเขตของโปรแกรม (Block Structure) ซึ่งเป็นการกำหนดขอบเขตของการใช้ตัวแปร รวมถึงกำหนดโครงสร้างของชุดคำสั่ง ซึ่งมีโครงสร้างแบบลำดับ (Sequential) โครงสร้างแบบเลือก (Choice) และโครงสร้างแบบวนซ้ำ (Repeat)

วิธีแบบโครงสร้างนี้ใช้ได้ดีและเป็นที่ยอมรับมากมายแต่เมื่อเวลาผ่านไปมีวิธีการประยุกต์ใช้ในหลายๆ เรื่องที่วิธีการเหล่านี้ไม่ได้ครอบคลุมถึง เช่น การพัฒนาระบบขนาดใหญ่ การพัฒนาระบบที่เป็นเรียวไทม์ (Real Time System) ทำให้เกิดปัญหาขึ้น โดยปัญหาทั่วไปที่พบคือการบำรุงรักษาระบบเพื่อให้สามารถใช้งานต่อไปได้นั้นทำได้ยาก โดยผู้ที่ทำหน้าที่บำรุงรักษานั้นต้องทำความเข้าใจถึงตัวแปรและ

ฟังก์ชันการทำงานของระบบว่ามีเพื่ออะไรและทำหน้าที่อะไรซึ่งใช้เวลาในการศึกษามากและการแก้ข้อผิดพลาดนั้นทำให้เกิดผลกระทำกับส่วนอื่นมากหรือทำให้เกิดข้อผิดพลาดขึ้นที่ส่วนอื่นได้ (Side Effect)

2.1.1.2 วิธีการทางออบเจ็กต์

ช่วยแก้ปัญหาต่างๆที่เกิดขึ้นกับวิธีการทางโครงสร้าง โดยใช้คุณสมบัติพื้นฐานต่างๆ ของออบเจ็กต์ซึ่งจะนำเสนอในหัวข้อต่อไป และใช้การออกแบบล่างขึ้นบน (Bottom-Up Design) โดยทำการมองถึงส่วนประกอบที่เล็กที่สุดของระบบก่อนแล้วจึงประกอบกันขึ้นมาจนเป็นระบบขนาดใหญ่ และหลักการที่สำคัญอีกอย่างหนึ่งของออบเจ็กต์คือแต่ละออบเจ็กต์รับรู้เพียงข้อมูลเข้าและข้อมูลที่ได้จากออบเจ็กต์อื่นเท่านั้นทำให้สามารถทำการเปลี่ยนแปลงออบเจ็กต์ตัวใดก็ได้ โดยไม่ส่งผลกระทบต่อออบเจ็กต์อื่นๆในระบบทำให้การบำรุงรักษาระบบสามารถทำได้ง่ายและรวดเร็ว

2.1.2 ข้อดีของวิธีการทางออบเจ็กต์

2.1.2.1 Consistency of Model Views

ในการพัฒนาแบบโครงสร้างนั้นในการเปลี่ยนระหว่างมุมมองการวิเคราะห์ และมุมมองการออกแบบเป็นเรื่องที่ยาก ทั้งๆ ที่ เรากำลังมองระบบเดียวกัน(ในมุมมองการวิเคราะห์นั้นเราอาจจะใช้ Data Flow diagram หรือ ER Diagram แต่ในการมุมมองการออกแบบนั้นเราใช้ Structure Chart ในการอธิบายการทำงาน) และเมื่อลงในระดับการโปรแกรมนั้นการที่เราจะอธิบายว่าการทำงานของโปรแกรมที่เราเขียนขึ้นมานั้นเป็นไปตามสิ่งที่เราวิเคราะห์และออกแบบหรือไม่ก็เป็นเรื่องที่ยากเช่นกัน ในขณะที่วิธีการทางออบเจ็กต์ในแต่ขั้นตอนการทำงานนั้นเราจะใช้มุมมองเดียวกันตลอดคือใช้คุณสมบัติพื้นฐานของออบเจ็กต์ ทำให้การเปลี่ยนมุมมองระหว่างการวิเคราะห์และการออกแบบสามารถทำได้ง่าย โดยเราสามารถนำออบเจ็กต์ที่เรากำหนดระหว่างการออกแบบ นำมาสร้างเป็นโปรแกรมได้เลย

2.1.2.2 Improve Problem Domain Abstraction

การแยกแยะเอกลักษณ์ (Abstraction) ของปัญหา ในการพัฒนาระบบแบบโครงสร้างมีหลักการคือ ทำการแยกระหว่างโครงสร้างข้อมูล (Structure) กับ วิธีปฏิบัติกับข้อมูลนั้น (behavior) ออกจากกัน ซึ่งเป็นจุดอ่อนของการออกแบบในรูปแบบนี้ เพราะจะทำให้ความสัมพันธ์ระหว่างสิ่งทั้งสองซึ่งควรมีในโลกของความเป็นจริงนั้นหายไป ในส่วนของวิธีการทางออบเจ็กต์จะรักษาความสัมพันธ์ระหว่าง ข้อมูลและการกระทำของข้อมูลไว้ด้วยกัน ดังนั้นการสร้างลักษณะเด่นของปัญหาเพื่อให้ผู้พัฒนามีความเข้าใจในปัญหาของระบบมากขึ้นจึงสามารถทำได้ครอบคลุมกว่าการออกแบบแบบโครงสร้าง อีกทั้งเวลาการเก็บข้อมูลความต้องการผู้ใช้สามารถอธิบายให้ผู้พัฒนาเข้าใจได้ไม่ยากเพราะในมุมมองของผู้ใช้จะมองการทำงานที่เขาทำนั้นในลักษณะของหน้าที่ของ ออบเจ็กต์เช่นกัน

2.1.2.3 Improved Stability in the Presence of Change

ในวิธีแบบโครงสร้างถ้าเกิดมีการแก้ไขโปรแกรมจะมีผลกระทบมากมาย (อาจจะต้องมีการแก้ทั้งโปรแกรมใหม่ทั้งโปรแกรม) แต่ในลักษณะของวิธีการแบบออบเจ็กต์การเปลี่ยนแปลงความต้องการในระบบที่ถูกพัฒนาขึ้นสามารถทำได้ง่ายและการแก้ไขที่เกิดขึ้นจะไม่กระทบกับการทำงานของออบเจ็กต์ตัวอื่น ทำให้ไม่จำเป็นที่ต้องเขียนโปรแกรมใหม่ทั้งหมด

2.1.2.4 Improved Modal Facilities for Reuse

ในวิธีการแบบโครงสร้างนั้นเรามักไม่มองว่าเป็นการนำกลับมาใช้ใหม่ (Reuse) เพราะเป็นการนำส่วนของโปรแกรมเดิมมาแก้ไขทั้งหมดเพื่อให้เป็นไปตามฟังก์ชั่นใหม่ที่เราต้องการ แต่ในวิธีการทางออบเจ็กต์จะมีหลักการ 2 อย่างที่สำคัญที่สนับสนุนการนำกลับมาใช้ใหม่คือ

- Generalization เป็นการเพิ่มหรือขยายองค์ประกอบโดยปราศจากการแก้ไขโปรแกรมเดิมแต่ใช้หลักการที่ว่าจะเขียนเฉพาะส่วนที่แตกต่างเท่านั้น (Programming by Difference)
- Refinement คล้ายกับ Generalization โดยเป็นการอนุญาตให้ออบเจ็กต์ที่ยังถูกกำหนดมาไม่ครบสามารถทำให้สมบูรณ์ได้โดยเพิ่มส่วนที่ยังไม่ครบเข้าไปได้ พื้นฐานของรูปแบบนี้คือการเขียนโปรแกรมที่มีหน้าที่หนึ่งแต่สามารถใช้กับข้อมูลได้หลายชนิด เป็นต้น

2.1.2.5 Better Support for Reliability and Safety Concern

การใช้วิธีการทางออบเจ็กต์นั้นเราจะมีการทำ Abstraction และ Encapsulation ดังนั้นการติดต่อของแต่ละออบเจ็กต์สามารถถูกกำหนดให้เกิดขึ้นภายในข้อกำหนดทางอินเตอร์เฟส(Interface) เท่านั้น ทำให้เกิดความปลอดภัยในเชิงการพัฒนาต่อ และมีเสถียรภาพมากขึ้นเพราะเราสามารถที่จะกำหนดเงื่อนไขก่อนหรือหลังการทำงานหรือติดต่อระหว่างออบเจ็กต์ได้

2.2 คำศัพท์ที่เกี่ยวข้องกับการพัฒนาโปรแกรมเชิงวัตถุ

2.2.1 ออบเจ็กต์

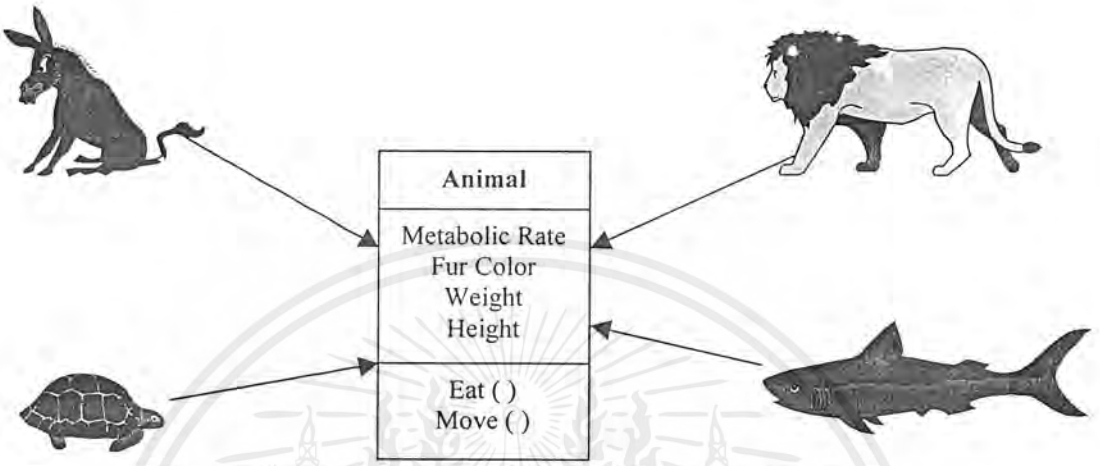
เป็นรูปแบบหนึ่งในการแสดงถึงสิ่งที่มีในโลกความเป็นจริง ซึ่งประกอบด้วยส่วนที่เป็นข้อมูล (Data) กับส่วนที่เป็นการทำงาน (Behavior) ซึ่งอาจมีคุณสมบัติอื่นเช่น สถานะ (State), สิ่งที่ระบุถึงออบเจ็กต์ (Identity), ความรับผิดชอบของออบเจ็กต์(Responsibility) เพื่อใช้ในการระบุถึงออบเจ็กต์เพื่อให้เข้าใจถึงออบเจกต์นั้นชัดเจนขึ้น เช่น

Attributes	Behavior	State	Identity	Responsibility
- ขนาดตัวอักษร - สี - ชื่อ	- เขียน - ลบ - เปลี่ยนสี	ลักษณะตัวอักษรปัจจุบัน	Times New Roman 16 pt.	ใช้ในการอ่าน และเป็นการแสดงความสวยงามของตัวอักษร

ตาราง 2-1 แสดงการระบุถึงวัตถุตัวอักษร

2.2.2 คลาส (Class)

คือกลุ่มของออบเจ็กต์ที่แบ่งตามลักษณะเฉพาะและการการใช้งาน หรือกล่าวได้ว่าเป็นการแบ่งออบเจ็กต์ที่มีคุณสมบัติพื้นฐานที่เหมือนกันไว้ในคลาสเดียวกัน ซึ่งเราเรียกคลาสว่าเป็นต้นแบบของออบเจ็กต์ (Class is template of objects) หรือออบเจ็กต์เป็นอินสแตนท์ของคลาส (Object is instance of class)



รูปที่ 2-1 ตัวอย่างความสัมพันธ์ระหว่างออบเจ็กต์กับคลาส

2.2.3 แอตทริบิวต์(Attribute)

ออบเจ็กต์จำเป็นต้องมีการเก็บข้อมูลของตัวเอง โดยข้อมูลที่ออบเจ็กต์เก็บไว้นั้นอาจใช้ในการประมวลผลเพื่อตอบสนองต่อเหตุการณ์ภายนอก หรือเก็บไว้เพื่อให้ออบเจ็กต์อื่นมาเรียกใช้งาน หรือเก็บไว้เพื่อบอกสถานะของตนเองก็ได้ ดังนั้นจึงสรุปได้ว่าออบเจ็กต์โดยส่วนใหญ่จะต้องเก็บข้อมูลเพื่อให้ออบเจ็กต์สามารถทำหน้าที่รับผิดชอบได้อย่างถูกต้อง

โดยทั่วไปแล้วแอตทริบิวต์ของออบเจ็กต์จะถูกซ่อนไว้จากผู้ที่จะใช้งานออบเจ็กต์ทั้งนี้เนื่องจากว่าการเข้าใช้งานแอตทริบิวต์โดยตรงเปรียบเสมือนเป็นการเข้าถึงโครงสร้างของออบเจ็กต์นั้นโดยตรง ดังนั้นการออกแบบออบเจ็กต์ที่ดีควรที่จะซ่อนแอตทริบิวต์ของออบเจ็กต์ไว้เพื่อไม่ให้ผู้ใช้เห็น

ในบางครั้งการออกแบบออบเจ็กต์ที่ไม่ดีอาจได้ออบเจ็กต์ที่มีแต่แอตทริบิวต์แต่ไม่มีเมธอด ซึ่งในกรณีนี้ผู้ออกแบบต้องย้อนกลับไปดูว่าออบเจ็กต์นั้นควรมีเมธอดอะไรบ้าง เพราะการที่ออบเจ็กต์มีแต่แอตทริบิวต์นั้นก็ไม่ต่างจากโครงสร้างข้อมูลที่เป็นเรคอร์ดเลย

2.2.4 เมธอด (Method)

เป็นการกระทำที่ออบเจ็กต์สามารถทำได้และอยู่ในขอบเขตที่เราสนใจ เช่นออบเจ็กต์สิงโตมีการกระทำมากมายแต่เราสนใจแค่การกินกับการเคลื่อนที่เท่านั้นเราจึงกำหนดให้มีเพียงสองการกระทำเท่านั้น ซึ่งถ้าการกระทำนี้มีลำดับเหตุการณ์ก่อนหลัง (State) เข้ามาพิจารณาด้วย เราจะแบ่งออกได้เป็น 3 ลักษณะดังนี้

2.2.4.1 Simple

สำหรับเมธอดแบบ Simple นี้ออบเจ็กต์จะให้บริการกับการร้องขอเข้ามาและไม่ค้างข้อมูลไว้ในหน่วยความจำสำหรับบริการครั้งต่อไป การกระทำเป็นแบบสมบูรณ์ในตัวเอง คือเหตุการณ์ในอดีตไม่ส่งผลถึงการกระทำในปัจจุบัน ออบเจ็กต์พื้นฐานจะมีชนิดข้อมูลพื้นฐานและโอเปอเรชั่นที่เกี่ยวข้องกับข้อมูลเหล่านี้ ตัวอย่างเช่นออบเจ็กต์ไบนารีทรี (Binary tree) จะมีเมธอดแบบ Simple เช่น InsertItem และ RemoveItem ส่วนตัวอย่างอื่นเช่น Cos(x) ถ้า x เป็นค่าเดิมทุกครั้งผลที่ได้ออกมาจะเป็นค่าเดิมด้วยซึ่งทำให้เห็นว่า Cos(x) นั้นไม่ได้เก็บสถานะของข้อมูลเก่าไว้เลย เราเรียกออบเจ็กต์แบบนี้ว่า Primitive Object

2.2.4.2 Automaton

มีการทำงานเป็นในลักษณะ Finite State Machine คือออบเจ็กต์ชนิดนี้จะมีเซตของสถานะที่มีขอบเขต ออบเจ็กต์หนึ่งอาจมีสถานะ 1 หรือมากกว่าในเวลาใดๆ การที่ออบเจ็กต์จะอยู่ในสถานะใดๆนั้นจะขึ้นอยู่กับอีเวนต์หรือเหตุการณ์ที่เข้ามาจากสภาพแวดล้อมภายนอก และเนื่องจากว่าออบเจ็กต์แบบนี้ทำงานแบบ State Machine เพื่อตอบสนองกับอีเวนต์ภายนอก เราจึงเรียกออบเจ็กต์แบบนี้ว่า Reactive Object

อีเวนต์ภายนอกที่เข้ามาอาจทำให้ออบเจ็กต์เปลี่ยนสถานะ วิธีการทางออบเจ็กต์โอเรียนเต็ดบางวิธีอ้างว่าออบเจ็กต์ทั้งหมดมีสถานะ ตัวอย่างเช่นตัวแปลง A/D แบบ Sample-and-Hold อาจเป็นออบเจ็กต์ที่มีสถานะต่างๆ เช่น Enabled, Sampling, Holding, Disabled

2.2.4.3 Continuous

ออบเจ็กต์ที่มีเมธอดแบบ Continuous จะมีเซตของสถานะที่ไม่จำกัดและไม่มีขอบเขต โดยมีการทำงานของออบเจ็กต์ปัจจุบันขึ้นอยู่กับการทำงานของเมธอดและอินพุตก่อนหน้า ซึ่งความเกี่ยวข้องระหว่างสถานะนี้อาจเป็นความสัมพันธ์แบบต่อเนื่อง ตัวอย่างเช่นในระบบของ Fuzzy และตัวควบคุม PID ซึ่งเป็นเครื่องสร้างจำนวนแบบ Pseudo-random และตัวกรองดิจิตอล การทำงานของเมธอดของระบบเหล่านี้จะขึ้นอยู่กับอินพุตก่อนหน้า

2.2.5 แมสเซจ (Message)

การสื่อสารระหว่างออบเจ็กต์สามารถทำได้โดยการส่งแมสเซจ แมสเซจเป็นการแยกแยะเอกลักษณ์ของข้อมูลหรือข้อมูลการควบคุมที่ส่งมาจากออบเจ็กต์หนึ่งไปยังออบเจ็กต์อื่นๆ ในเชิงการเขียนโปรแกรมนั้นแมสเซจสามารถทำหน้าที่ได้หลายแบบเช่น การเรียกฟังก์ชัน การอินเทอร์รัพท์

ในขั้นตอนการวิเคราะห์ระบบนั้นจะมีการกำหนดแมสเซจ ส่วนในขั้นตอนของการออกแบบนั้นจะกำหนดรูปแบบของการซิงโครไนเซชั่นและความต้องการทางด้านเวลาของแต่ละแมสเซจ และเมื่อออบเจ็กต์ได้รับแมสเซจมา ออบเจ็กต์จะเปลี่ยนแมสเซจที่เข้ามาให้เป็นโอเปอเรชั่น การเปลี่ยนสถานะ คำสั่งหรือข้อมูลตามที่เหมาะสม แมสเซจถูกส่งระหว่างออบเจ็กต์ได้ก็ต่อเมื่อออบเจ็กต์ต้องมีความสัมพันธ์แบบ Association

การใช้แมสเซจทำให้แต่ละออบเจ็กต์มีความเกี่ยวข้องกันน้อยลง ในขั้นตอนของการวิเคราะห์ผู้พัฒนาระบบจะไม่ได้กำหนดรายละเอียดในการติดต่อของแมสเซจในการเรียกใช้ฟังก์ชัน ไทม์เอาท์ แต่การกำหนดรายละเอียดนี้จะทำเมื่อถึงขั้นตอนการออกแบบ

ส่วนอินเตอร์เฟซของออบเจ็กต์เป็นส่วนที่ออบเจ็กต์ใช้ติดต่อกับโลกภายนอก ซึ่งอินเตอร์เฟซจะทำหน้าที่กำหนดเซตของโพรโตคอลสำหรับการสื่อสารกับออบเจ็กต์อื่นๆ ซึ่งโพรโตคอลจะประกอบไปด้วย 3 สิ่งดังนี้

- พรีคอนดิชัน (Precondition) เป็นเงื่อนไขที่มีค่าจริงก่อนที่จะส่งหรือรับแมสเซจ
- ซิกเนเจอร์ (Signature) เป็นรูปแบบในการส่งแมสเซจ
- โพสต์คอนดิชัน (Postcondition) เป็นเงื่อนไขที่มีค่าจริงหลังจากที่ออบเจ็กต์ได้ประมวลผลแมสเซจเรียบร้อยแล้ว

2.2.6 ความรับผิดชอบ (Responsibility)

ความรับผิดชอบของออบเจ็กต์คือหน้าที่และบทบาทของออบเจ็กต์ต่อระบบ โดยส่วนอินเตอร์เฟสและเมธอดของออบเจ็กต์นั้นสามารถใช้เพื่อให้ออบเจ็กต์ทำหน้าที่รับผิดชอบได้ โดย ความรับผิดชอบของออบเจ็กต์ควรจะเป็นสิ่งแรกที่ผู้พัฒนาระบบกำหนดให้กับออบเจ็กต์เพราะถ้าไม่รู้ว่าออบเจ็กต์นั้นมีหน้าที่อย่างไร ผู้พัฒนาจะไม่สามารถกำหนดคุณลักษณะอื่นๆให้กับออบเจ็กต์ได้อย่างถูกต้องและครบถ้วน

2.3 ความสัมพันธ์ระหว่างออบเจ็กต์

โดยปกตินั้นออบเจ็กต์เพียงออบเจ็กต์เดียวไม่สามารถที่จะรับผิดชอบงานทั้งหมดของระบบได้ ออบเจ็กต์จะต้องมีความสัมพันธ์และต้องทำงานร่วมกันเพื่อให้ระบบสามารถทำหน้าที่รับผิดชอบได้ โดยเราแบ่งความสัมพันธ์ระหว่างออบเจ็กต์ออกเป็น 2 แบบคือลิงค์ (Link) และ อะกรีเกชัน (Aggregation)

2.3.1 ลิงค์

ความสัมพันธ์ระหว่างออบเจ็กต์แบบลิงค์เป็นการเชื่อมต่อระหว่างออบเจ็กต์ในทางกายภาพหรือในทางความคิด การที่ออบเจ็กต์ซึ่งมาประกอบกันเป็นระบบจะทำงานร่วมกันได้นั้นออบเจ็กต์เหล่านี้จะต้องมีการสื่อสารระหว่างกัน ออบเจ็กต์จะสื่อสารกันโดยการที่ออบเจ็กต์เรียกใช้บริการหรือเซอร์วิส (Service) จากออบเจ็กต์อื่น โดยออบเจ็กต์ที่เรียกใช้บริการเราเรียกว่า ออบเจ็กต์ไคลเอนต์ ส่วนออบเจ็กต์ที่ให้บริการแก่ออบเจ็กต์อื่นเราเรียกว่าออบเจ็กต์เซิร์ฟเวอร์

2.3.2 อะกรีเกชัน

ความสัมพันธ์แบบอะกรีเกชันเป็นความสัมพันธ์ที่ออบเจ็กต์เป็นส่วนหนึ่งของอีกออบเจ็กต์ โดยอาจเป็นความสัมพันธ์แบบอะกรีเกชันธรรมดาหรือเป็นความสัมพันธ์แบบคอมโพซิชัน (Composition) โดยความสัมพันธ์แบบคอมโพซิชันนั้นแตกต่างจากความสัมพันธ์แบบอะกรีเกชันที่ ออบเจ็กต์ที่ถูกอะกรี

เกชันนั้นสามารถถูกอะกรีเกชันกับออบเจ็กต์อื่นได้ แต่ในกรณีของคอมโพซิชันนั้นออบเจ็กต์ที่ถูกคอมโพซิชันนั้นไม่สามารถถูกคอมโพซิชันด้วยออบเจ็กต์อื่นได้

2.4 คุณสมบัติที่สำคัญของการพัฒนาโปรแกรมเชิงวัตถุ

2.4.1 คุณสมบัติเอนแคปซูเลชัน (Encapsulation)

เอนแคปซูเลชันเป็นการรวมสิ่งต่างๆ เข้าไว้ด้วยกันเป็นหนึ่งหน่วย ซึ่งโดยหลักการนี้เมื่ออยู่ในรูปแบบของออบเจ็กต์แล้วเอนแคปซูเลชันจะเป็นการรวมแอตทริบิวต์และเมธอดของออบเจ็กต์ไว้เป็นหน่วยเดียวกัน ซึ่งคุณสมบัตินี้จะทำการซ่อนโครงสร้างของข้อมูลภายในของออบเจ็กต์จากระบบภายนอก และเป็นการป้องกันการเปลี่ยนแปลงข้อมูลหรือเปลี่ยนแปลงการทำงานของออบเจ็กต์ โดยการปกป้อง ข้อมูลนี้จึงทำให้คุณสมบัตินี้ถูกเรียกอีกชื่อหนึ่งว่าการซ่อนข้อมูล (Information Hiding)

2.4.2 คุณสมบัติการสืบทอดคุณสมบัติ (Inheritance)

คือการที่สามารถนำเอาลักษณะเฉพาะ หรือความสามารถของคลาสที่ได้สร้างขึ้นไว้แล้ว นำไปให้กับคลาสที่สร้างขึ้นมาใหม่ได้ โดยคลาสที่สร้างขึ้นก่อนนั้นจะถูกเรียกเป็นคลาสหลัก (Base Class) ส่วนคลาสที่ได้รับการสืบทอดคุณสมบัติจะถูกเรียกเป็นคลาสลูก (Derived Class) ซึ่งมีการแบ่งการสืบทอดคุณสมบัติออกเป็น 2 ส่วนคือการสืบทอดลักษณะเฉพาะ และการสืบทอดเมธอด

2.4.2.1 การสืบทอดลักษณะเฉพาะ เป็นการสืบทอดแอตทริบิวต์จากคลาสหลักไปสู่คลาสลูก

2.4.2.2 การสืบทอดเมธอด คล้ายกับการสืบทอดลักษณะเฉพาะแต่เป็นการสืบทอดเมธอดแทน

2.4.3 คุณสมบัติโพลีมอร์ฟิซึม (Polymorphism)

เป็นคุณสมบัติที่เกิดขึ้นมาจากการสืบทอดเมธอด โดยออบเจ็กต์ลูกมากกว่าหนึ่งออบเจ็กต์ที่ได้รับการสืบทอดเมธอด ดังนั้นออบเจ็กต์ลูกแต่ละออบเจ็กต์จะมีชื่อเมธอดที่เหมือนกันแต่มีการทำงานที่ไม่เหมือนกัน เช่น คลาสสัตว์ปีกกับคลาสสัตว์น้ำได้รับการสืบทอดคุณสมบัติการเคลื่อนที่มาจากคลาสสัตว์ โดยสัตว์ปีกการเคลื่อนที่จะเป็นการบินแต่สัตว์น้ำการเคลื่อนที่จะเป็นการว่ายน้ำ

2.5 การวิเคราะห์และออกแบบเชิงวัตถุ

ในขั้นตอนการวิเคราะห์และออกแบบวัตถุนี้เราได้ใช้ UML (Unified Modeling Language) ซึ่งเป็นมาตรฐานหนึ่งที่ใช้ในการวิเคราะห์และออกแบบเชิงวัตถุ

โดยหลักการแล้ว UML จะช่วยในการสร้างความเข้าในร่วมกันระหว่างกลุ่มผู้พัฒนาระบบกับผู้ใช้งานระบบ หรืออาจเป็นภายในกลุ่มผู้พัฒนาระบบ โดยกำหนดสัญลักษณ์และคำจำกัดความของสัญลักษณ์ต่างๆ ขึ้นมาเพื่อทำให้มองเห็นภาพของระบบที่กำลังพัฒนาขึ้นมาอย่างชัดเจนขึ้น และ UML สามารถถูกนำไปใช้ในส่วนต่างๆ ของการพัฒนาระบบได้ เช่น การเก็บความต้องการของระบบ (Requirement) การวิเคราะห์และออกแบบระบบ (Analysis and Design) การสร้างระบบ (Implement) การทดสอบระบบ (Testing)

2.5.1 การเก็บความต้องการของระบบ (Requirement Analysis)

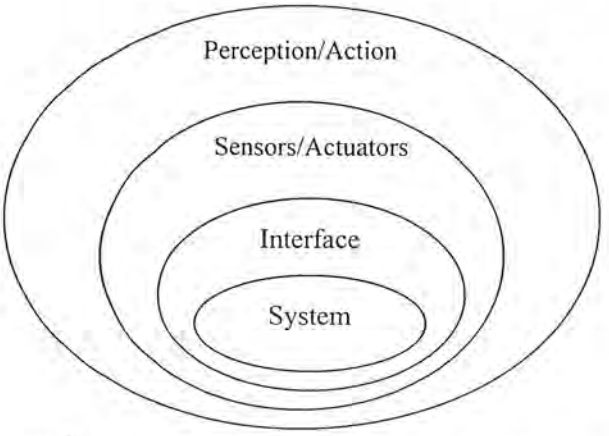
ในขั้นตอนนี้สิ่งที่เราต้องการคือ ออบเจ็กต์ภายนอก (External Object) หรือสภาพแวดล้อมของระบบ (System Environment) เป็นอย่างไร การตอบสนองของระบบต่อผู้ใช้เป็นอย่างไร และความต้องการพื้นฐานที่ผู้ใช้ต้องการให้ระบบมีคืออะไร

โดยปกติระบบจะต้องติดต่อกับสภาวะแวดล้อมภายนอก ซึ่งการที่เราจะรู้ว่าระบบติดต่อกับสภาวะแวดล้อมภายนอกอย่างไรนั้นเราจำเป็นต้องมีการศึกษาความต้องการของระบบ(หรือ Requirement Analysis) ในการวิเคราะห์ระบบแบบโครงสร้างผลที่ได้จากขั้นตอนนี้จะเป็นไดอะแกรม 2 ตัวคือ Context Diagram และ Data Flow Diagram แต่สำหรับวิธีการวิเคราะห์และออกแบบระบบด้วยออบเจ็กต์โอเรียนเต็ดผลที่ได้จากขั้นตอนนี้จะเป็น External Event Context และ Use Case Diagram โดย External Event Diagram จะจำลองให้ระบบเป็นเหมือนกับออบเจ็กต์ใหญ่ตัวหนึ่งซึ่งประกอบด้วยออบเจ็กต์ย่อยภายในและออบเจ็กต์ของระบบนี้จะแสดงการติดต่อสื่อสารทั้งการรับและส่งแมสเซจกับออบเจ็กต์ที่เป็น แอคเตอร์ (Actor) ภายนอก ส่วน Use Case และ Scenario จะแสดงฟังก์ชันการทำงานหลักของระบบและโพรโตคอลในการแลกเปลี่ยนแมสเซจที่จำเป็นเพื่อให้ระบบสามารถติดต่อกับแอคเตอร์ภายนอกได้ตามความถูกต้องตามความต้องการของระบบ

2.5.1.1 External Event Context

System context เป็นแผนที่สำหรับสิ่งที่ผู้พัฒนาหรือผู้ใช้ระบบนั้น ในการพัฒนาระบบแบบโครงสร้างนั้นเราสามารถแทน System context ได้ด้วย Context Diagram โดย Context Diagram เป็นโมเดลที่อยู่ในการพัฒนาระบบแบบโครงสร้างแต่สามารถนำมาใช้กับวิธีการออบเจ็กต์โอเรียนเต็ดได้ โดย Context Diagram นี้จะแสดงสภาพแวดล้อมที่มีต่อระบบ ซึ่งประกอบไปด้วยผู้กระทำหรือแอคเตอร์ การแลกเปลี่ยนแมสเซจและอีเวนต์ระหว่างระบบกับสภาพแวดล้อมภายนอก

แต่ว่า UML เองไม่ได้สนับสนุน Context Diagram โดยตรง แต่เราสามารถใช้ Stereo Type ของ UML ในการสร้าง Context Diagram ขึ้นมาได้และในหัวข้อนี้ขอเสนอเพียง Context Diagram ในรูปแบบของ Onion Skin เท่านั้นสำหรับผู้ที่สนใจสามารถศึกษาเพิ่มเติมได้จากหนังสือ Real Time UML Developing Efficient Objects For Embedded Systems ซึ่งรายละเอียดของหนังสือนี้สามารถดูได้จากส่วนบรรณานุกรม



รูปที่ 2-2 การเปรียบเทียบระบบแบบ Onion Skin Analogy

Perception / Action เป็นส่วนของการแสดงการกระทำต่างๆที่เกิดขึ้นกับระบบ ซึ่งชั้นนี้ทำให้ผู้ใช้เป็นข้อมูลบนจอมอนิเตอร์หรือสามารถเห็นการกระทำของระบบที่มีต่อสภาพแวดล้อมได้ ตัวอย่างเช่นจอภาพของตู้ ATM ที่แสดงจำนวนเงินที่เหลืออยู่ในบัญชี

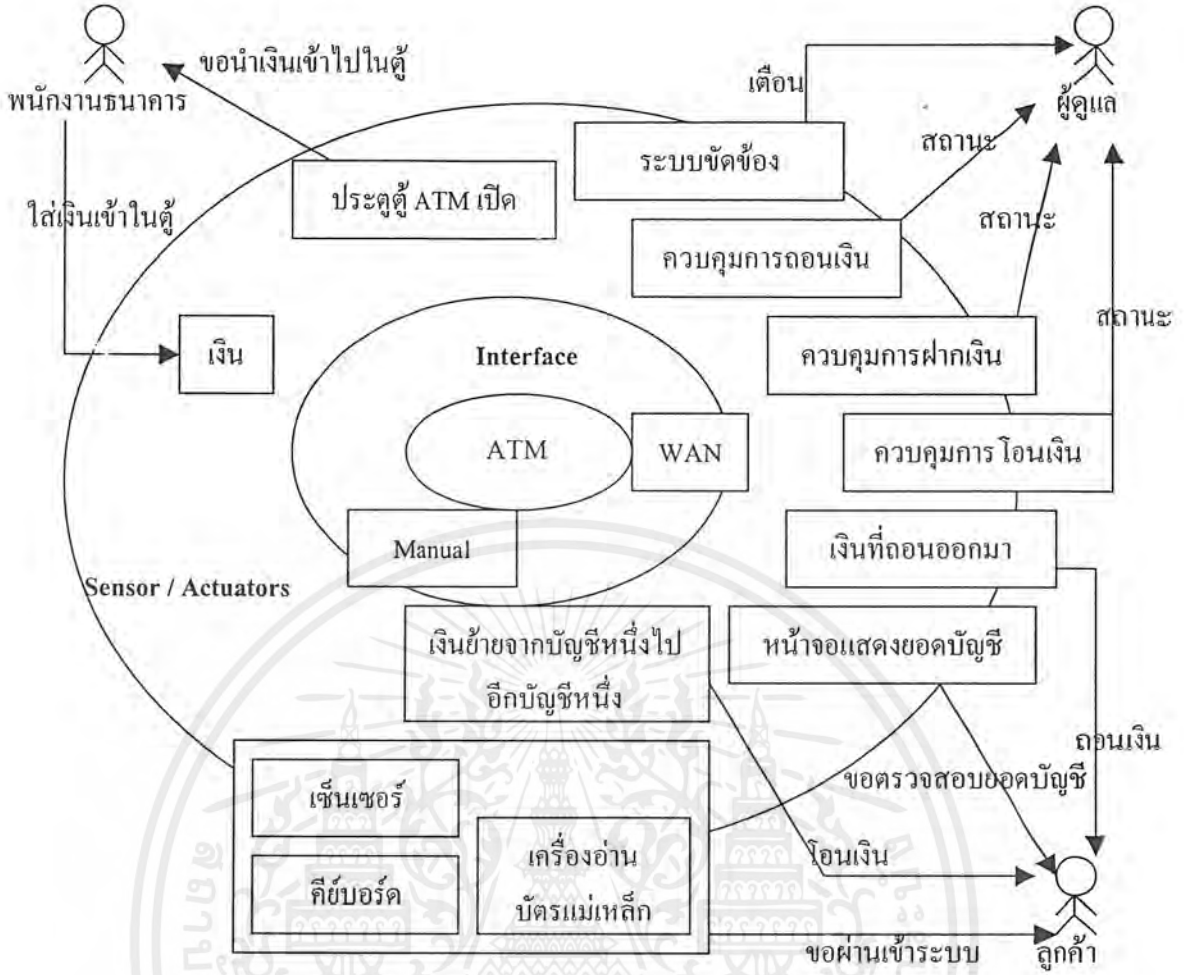
Sensors / Actuators ทำหน้าที่รับข้อมูลหรือให้ผลตอบสนองต่อสภาพแวดล้อม ในตัวอย่างตู้ ATM อุปกรณ์ในชั้นนี้คือเครื่องอ่านบัตรแม่เหล็กหรือเครื่องจ่ายเงินที่จ่ายเงินออกมาตามจำนวนที่ลูกค้าต้องการ

Interfacc เป็นอุปกรณ์ที่ใช้ในการติดต่อสื่อสาร เช่น โมเด็มหรือระบบ WAN

System คือระบบศูนย์กลางที่ทำหน้าที่ตัดสินใจและประมวลผลข้อมูล ซึ่งในกรณีของตู้ ATM ระบบศูนย์กลางอาจไม่ใช่เครื่องเมนเฟรมที่อยู่ที่สาขาใหญ่แต่อาจเป็นระบบคอมพิวเตอร์ที่ควบคุมการทำงานที่อยู่ในตู้ ATM เองก็ได้

Perception / Action	Sensors / Actuators	Interface	System
ตรวจสอบยอดบัญชี	คีย์บอร์ด, จอภาพ, เครื่องอ่านบัตรแม่เหล็ก	WAN	ATM
ถอนเงิน	คีย์บอร์ด, จอภาพ, เครื่องอ่านบัตรแม่เหล็ก, เครื่องจ่ายเงิน, เครื่องนับเงิน	WAN	ATM
โอนเงิน	คีย์บอร์ด, จอภาพ, เครื่องอ่านบัตรแม่เหล็ก, เครื่องนับเงิน	WAN	ATM
เพิ่มธนบัตรในตู้ ATM	กุญแจ, ธนบัตร	Manual	ATM

ตารางที่ 2-2 ตัวอย่างออบเจ็กต์ในระบบของตู้ ATM



รูปที่ 2-3 ตัวอย่างการใช้ Onion Skin ของระบบตู้ ATM

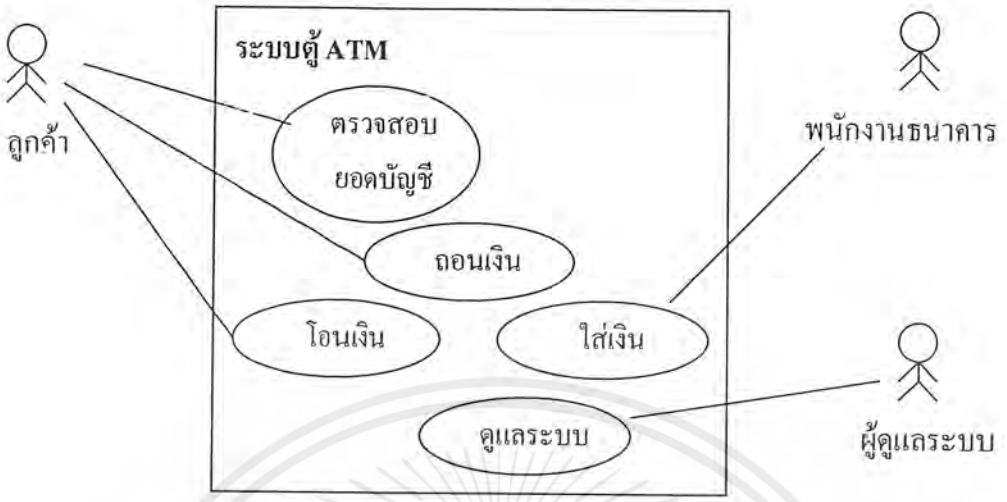
2.5.1.2 Use Case

เป็นไดอะแกรมที่เพิ่มกรณีสำหรับการติดต่อระหว่างระบบกับแอคเตอร์ภายนอก นอกจากนี้ยังแสดงการไหลของอีเวนต์ และสามารถใช้ตรวจสอบความถูกต้องของ Context Diagram ได้อีกด้วย

Use Case Diagram นั้นจะแสดงมุมมองต่อฟังก์ชันการทำงานหลักๆของระบบ ซึ่งผู้พัฒนาระบบสามารถใช้ในการสื่อสารกับผู้ใช้ระบบได้ด้วย การกำหนดฟังก์ชันใน Use Case นั้นเราต้องทำการสร้าง Scenario ของการทำงานที่เกิดขึ้นกับระบบขึ้นมาก่อนเปรียบ Scenario นี้เหมือนออบเจ็กต์แล้ว Use Case นั้นเป็นคลาส โดยเราจะนำ Scenario ที่มีลักษณะคล้ายกันมารวมเป็นหนึ่ง Use Case ขึ้นมาซึ่งการที่เราจะได้ Use Case ออกมาครบทั้งระบบเราจำเป็นต้องกำหนด Scenario ให้ครอบคลุมทุกการทำงานที่จะเกิดขึ้นด้วย นอกจากนี้ในแต่ละ Use Case ยังสามารถมีความสัมพันธ์กันได้ในลักษณะของ Uses และ Extend ดังนี้

- Extends Relationship : เป็น Generalization Relationship โดย Use case ที่ Extend ออกมาจาก Use case อื่นจะมีการเพิ่มการทำงานใหม่เข้าไปใน Use case เดิมโดยอาจไม่ใช้การทำงานทั้งหมดจาก Use Case เดิมที่ Extend มา

- Uses Relationship : เป็นการรวม Use case ที่มีการทำงานที่เหมือนกัน (Common Behavior) ไว้เป็น Use case เดียวแล้วถูกเรียกใช้โดย Use case อื่น



รูปที่ 2-4 Use Case Diagram ของระบบตู้ ATM

วิธีการหา Use Case

จากที่กล่าวมาแล้วว่าควรใช้ Scenario เพื่อทำการหา Use Case โดยเลือก Scenario ที่สำคัญต่อระบบขึ้นมาสร้างเป็น Use Case ดังนั้นจึงมีคำถามว่าจะหา Scenario ที่สำคัญต่อระบบได้อย่างไร ซึ่งมีคำถามดังต่อไปนี้ที่ช่วยในการหา Scenario ได้จากผู้ใช้ระบบ

- ฟังก์ชันการทำงานหลักของระบบคืออะไร
- ฟังก์ชันการทำงานรองของระบบคืออะไร
- ทำไมจึงต้องสร้างระบบขึ้นมา และระบบที่สร้างขึ้นจะเข้ามาเปลี่ยนอะไรและทำไมต้องเปลี่ยน

จากนั้นจะทำการสร้าง Use Case ขึ้นมาจาก

- บทบาทระหว่างออบเจ็กต์ภายนอกกับระบบภายใน Scenario
- การติดต่อสื่อสารทั้งระหว่างออบเจ็กต์ภายนอกต่อระบบหรือระหว่าง UseCase กับ Use Case อื่นในระบบ
- ลำดับการไหลของอีเวนต์และข้อมูลใน Scenario
- นักพัฒนาต้องคำนึงถึงความเปลี่ยนแปลงที่อาจจะเกิดขึ้นกับ Scenario ด้วย

2.5.1.3 Scenarios

Scenario คือตัวตนของ Use Case ในโมเดลของ Scenario นั้นจะกำหนดลำดับแมสเซจของระบบรวมถึงโครงสร้างของออบเจ็กต์ซึ่งทำงานร่วมกันเพื่อให้ระบบสามารถทำงานตามหน้าที่ได้ ถ้าใน Use Case หนึ่งมีหลาย Scenario นั้นหมายความว่าในแต่ละ Scenario นั้นจะแสดงรูปแบบในการติดต่อออบเจ็กต์ที่ต่างกันออกไป ในขั้นตอนการกำหนด Scenario นี้ผู้พัฒนาระบบควรจะทำความเข้าใจกับคุณ

ลักษณะของระบบทั้งหมดเนื่องจากตัว Scenario นั้นเป็นเครื่องมือที่มีประโยชน์สำหรับการตรวจสอบความถูกต้องความต้องการของระบบ

ในกระบวนการวิเคราะห์แบบออบเจ็กต์โอเรียนเต็ดนั้นมีโมเดลสำหรับ Scenario อยู่ 2 โมเดลคือ Sequence Diagram และ Collaborate Diagram โดยทั้งสองไดอะแกรมนี้มีความแตกต่างกันที่วัตถุประสงค์ในการใช้และรูปแบบการนำเสนอ โดย

Sequence Diagram จะเน้นเกี่ยวกับลำดับและเวลาที่ใช้ในการส่งแมสเซจติดต่อกันระหว่างออบเจ็กต์ ซึ่งจุดประสงค์ของการใช้ Sequence Diagram เพื่อต้องการผลในรูปแบบของเวลาเป็นหลัก

Collaboration Diagram จะเน้นเกี่ยวกับลำดับของแมสเซจและความสัมพันธ์ของออบเจ็กต์ต่างๆ ที่ติดต่อกันด้วยแมสเซจ ซึ่งจุดประสงค์ของการใช้ Collaboration Diagram เพื่อแสดงถึงความสัมพันธ์ในการติดต่อกันของออบเจ็กต์

2.5.2 การกำหนดโครงสร้างของออบเจ็กต์

เมื่อเรากำหนดฟังก์ชันการทำงานของระบบกับสิ่งแวดล้อมภายนอกแล้ว ขั้นต่อมานักพัฒนาจะต้องกำหนดออบเจ็กต์และคลาสหลักรวมทั้งความสัมพันธ์ระหว่างออบเจ็กต์และคลาสในระบบ ซึ่งในหัวข้อนี้จะกล่าวขึ้นขั้นตอนต่างๆ เหล่านี้ด้วย

2.5.2.1 การกำหนดออบเจ็กต์

ในการกำหนดออบเจ็กต์นั้นไม่มีหลักการที่แน่นอน ขึ้นอยู่กับประสบการณ์ของนักพัฒนาระบบเองแต่นอกเหนือจากประสบการณ์แล้วยังมีวิธีการหรือกลยุทธ์ที่ใช้ในการหาออบเจ็กต์ด้วย และการใช้กลยุทธ์เหล่านี้ไม่จำเป็นต้องใช้กลยุทธ์ทั้งหมด แต่ขึ้นอยู่กับความเหมาะสมว่านักพัฒนาระบบจะเลือกใช้กลยุทธ์ใด และปัญหาอย่างหนึ่งที่เกิดขึ้นจากการใช้กลยุทธ์นี้คือ ได้ออบเจ็กต์ออกมามากมาย ซึ่งมีออบเจ็กต์จำนวนไม่น้อยที่ซ้ำกัน ดังนั้นการกำหนดออบเจ็กต์จากกลยุทธ์นี้ต้องขึ้นอยู่กับการตัดสินใจของนักพัฒนาระบบด้วย ซึ่งกลยุทธ์ที่กล่าวถึงมีดังนี้

2.5.2.1.1 ขีดเส้นใต้คำนาม

เป็นกลยุทธ์แรกในการกำหนดออบเจ็กต์แบบง่ายๆ โดยให้ตัวผู้วิเคราะห์ระบบขีดเส้นใต้คำนามที่มีอยู่ในคำอธิบายปัญหา(ProblemStatement) จากนั้นทำการตรวจสอบว่าคำนามใดที่สามารถนำมากำหนดเป็นออบเจ็กต์ได้ ซึ่งคำนามที่ได้จะแยกเป็น 3 ประเภทคือ

1. ออบเจ็กต์ที่ผู้พัฒนาระบบสนใจ
2. ออบเจ็กต์ที่ผู้พัฒนาระบบไม่สนใจ
3. แอตทริบิวต์ของออบเจ็กต์

2.5.2.1.2 กำหนดแอกทีฟออบเจ็กต์

ออบเจ็กต์ที่ทำหน้าที่สร้างหรือควบคุมแอกชัน และสร้างหรือวิเคราะห์ข้อมูลนั้น เราเรียกออบเจ็กต์ประเภทนี้ว่าแอกทีฟออบเจ็กต์ (Active Object) แอกทีฟออบเจ็กต์นั้นเป็นออบเจ็กต์ที่ทำงานได้ด้วยตัวเองเพื่อทำแอกชัน ควบคุมการทำงานของออบเจ็กต์อื่น หรือทำหน้าที่สร้างอีเวนต์ โดยทั่วไปแอกทีฟออบเจ็กต์จะเป็นออบเจ็กต์หลักในการสร้างเธรด (Thread) ของออบเจ็กต์อื่นๆ ด้วย

2.5.2.1.3 กำหนดบริการของพาสซิฟออบเจ็กต์

พาสซิฟออบเจ็กต์เป็นออบเจ็กต์ที่ให้บริการหรือเก็บข้อมูลของออบเจ็กต์อื่น ซึ่งจะตรงกันข้ามกับแอกทีฟออบเจ็กต์ ตัวอย่างเช่นเครื่องจ่ายเงินของตู้ ATM จะทำการจ่ายเงินได้เมื่อระบบตู้ ATM เรียกใช้ให้ทำงานเท่านั้นนอกนั้นจะไม่ทำงานจ่ายเงินเองโดยระบบยังไม่สั่ง

2.5.2.1.4 กำหนดสิ่งที่มีอยู่ในโลกจริง

ระบบออบเจ็กต์โอเรียนเต็ดส่วนใหญ่จะสร้างขึ้นมาจากข้อมูลและหน้าที่การทำงานของสิ่งที่มีอยู่ในโลกจริง โดยกลยุทธ์นี้มีหลักการที่ว่าถ้าสิ่งใดที่ระบบติดต่อด้วยและสิ่งนั้นมีหน้าที่รับผิดชอบต่อออบเจ็กต์อื่นสามารถกำหนดสิ่งนั้นเป็นออบเจ็กต์ได้

2.5.2.1.5 กำหนดอุปกรณ์กายภาพ

โดยปกติระบบจะต้องติดต่อกับสิ่งแวดล้อมเพื่อทำหน้าที่รับผิดชอบอยู่แล้ว ระบบอาจประกอบด้วยอุปกรณ์ต่างๆ ซึ่งเราสามารถนำอุปกรณ์ต่างๆที่เราสามารถมองเห็นได้มากำหนดเป็นออบเจ็กต์ได้

2.5.2.1.6 กำหนดจากแนวความคิดหลัก

แนวความคิดหลักหรือ Key concept เป็นการแยกแยะเอกลักษณ์ของระบบที่สำคัญซึ่งมีทั้งแอตทริบิวต์และเมธอดที่ผู้วิเคราะห์ระบบสนใจ การแยกแยะเอกลักษณ์โดยใช้แนวความคิดหลักนั้นไม่จำเป็นต้องเป็นสิ่งที่มีตัวตนจริง แต่ว่ามีความหมายต่อระบบ

2.5.2.1.7 กำหนดจากข้อมูลที่เก็บอยู่

โดยปกติแล้วข้อมูลของระบบจะถูกเก็บอยู่ที่พาสซิฟออบเจ็กต์ หรือในฐานข้อมูล ซึ่งข้อมูลนี้จะเป็นแอตทริบิวต์ในออบเจ็กต์เพื่อให้ออบเจ็กต์เรียกใช้ได้

2.5.2.1.8 กำหนดจากทรานแซคชั่น

ทรานแซคชั่นสามารถเป็นออบเจ็กต์ได้ถ้าทรานแซคชั่นนั้นมีคุณสมบัติของออบเจ็กต์ตามที่ได้กล่าวไว้ข้างต้นของบท ซึ่งทรานแซคชั่นที่เกิดขึ้นนี้เกิดจากการติดต่อกันระหว่างออบเจ็กต์

2.5.2.1.9 กำหนดจากองค์ประกอบที่มองเห็น

นักพัฒนาระบบสามารถใช้ออบเจ็กต์ที่ได้จากวิธีนี้ไปใช้ในการออกแบบส่วนที่ทำการติดต่อกับผู้ใช้ได้ เช่นในระบบปฏิบัติการที่มีส่วนติดต่อกับผู้ใช้แบบกราฟิก (Graphic User Interface) อาจกำหนด ปุ่ม วินโดวส์ ไอคอน และข้อความเป็นออบเจ็กต์ได้

2.5.2.1.10 กำหนดองค์ประกออบที่ทำหน้าที่ควบคุม

องค์ประกอบที่ทำหน้าที่ควบคุมจะทำการควบคุมออบเจ็กต์อื่น องค์ประกอบที่ทำหน้าที่ควบคุมเป็นแอกทีฟออบเจ็กต์ชนิดหนึ่ง หรืออาจเรียกว่า Composite ซึ่งหมายถึงออบเจ็กต์ที่ประกอบด้วยออบเจ็กต์อื่นๆภายใน

2.5.2.1.11 ประยุกต์ใช้ Scenario

เราใช้ Scenario ในการตรวจสอบการกำหนดออบเจ็กต์ที่ได้จากวิธีต่างๆ ซึ่งทำให้เราทราบว่าออบเจ็กต์ใดที่มีความสำคัญกับระบบแล้วเรายังไม่ได้กำหนด

2.5.2.2 กำหนดความสำพันธ์ระหว่างออบเจ็กต์

การกำหนดความสัมพันธ์ระหว่างออบเจ็กต์นั้นส่วนใหญ่ดูจากการส่งแมสเซจเป็นหลัก เพราะว่าทุกๆแมสเซจจะหมายถึงความสัมพันธ์ระหว่างออบเจ็กต์ที่เป็นผู้ส่งแมสเซจกับออบเจ็กต์ผู้รับแมสเซจ ซึ่งมีวิธีการในการกำหนดส่วนต่างๆ ของแมสเซจดังนี้

1. กำหนดแมสเซจ
2. กำหนดเส้นทางของแมสเซจ
3. กำหนดออบเจ็กต์ที่เก็บแมสเซจ
4. กำหนดออบเจ็กต์ที่จัดการกับแมสเซจ
5. ประยุกต์ใช้ Scenario เพื่อตรวจสอบความถูกต้องของการกำหนดความสัมพันธ์

2.5.2.3 การกำหนดแอตทริบิวต์ของออบเจ็กต์

แอตทริบิวต์เป็นคุณสมบัติที่ไม่สามารถแบ่งแยกลงไปอีกได้ ถ้าในการออกแบบพบว่าแอตทริบิวต์ที่กำหนดสามารถแยกย่อยลงไปอีกได้ ควรกำหนดแอตทริบิวต์นั้นเป็นออบเจ็กต์ที่อยู่ภายใน

ในบางระบบจะพบว่าออบเจ็กต์บางตัวไม่จำเป็นต้องมีแอตทริบิวต์ แต่ออบเจ็กต์นั้นมีเพียงแค่เมธอดก็เพียงพอแล้ว เราเรียกออบเจ็กต์ที่มีแต่เมธอดว่า Functoids หรือ Class Utilities

2.5.2.4 การกำหนดเมธอดของออบเจ็กต์

เมธอดสามารถหาได้จากสเตทไดอะแกรม ซึ่งทำให้ได้สเตทแมชีน (Finite State Machine) โดยในหนึ่งคลาสจะมีหนึ่งสเตทแมชีน หรือกล่าวอีกอย่างได้ว่าหนึ่งออบเจ็กต์จะทำงานกับหนึ่งสเตทแมชีนนั่นเอง จากหลักการนี้การที่เราจะสร้างสเตทแมชีนขึ้นมาได้เราต้องรู้ก่อนว่าออบเจ็กต์นั้นต้องทำอะไรได้

บ้างนั่นคือต้องรู้ว่าออบเจ็กต์นั้นมีความรับผิดชอบอะไรบ้าง เรานำความรับผิดชอบนั้นมากำหนดเป็นโอเปอเรชัน (Operation) ซึ่งโอเปอเรชันนี้คือสิ่งที่ทำให้เกิดการเปลี่ยนสเตท ซึ่งโอเปอเรชันนี้เป็นส่วนประกอบของเมธอดซึ่งในแต่ละเมธอดจะประกอบขึ้นมาจากหลายๆโอเปอเรชัน

โอเปอเรชันสามารถแบ่งให้เป็นส่วนอินเตอร์เฟสหรือจะเป็นโอเปอเรชันภายใน (Internal Operation) ก็ได้ โดยถ้าเป็นอินเทอร์เฟสนั้นหมายความว่าออบเจ็กต์อื่นสามารถเรียกใช้งานโอเปอเรชันนั้นได้ แต่ถ้าเป็นโอเปอเรชันภายในคือโอเปอเรชันที่มีไว้ให้ออบเจ็กต์ที่เป็นเจ้าของโอเปอเรชันนั้นเรียกใช้โดยออบเจ็กต์อื่นไม่สามารถเรียกใช้งานได้

สำหรับการเขียนสเตทไดอะแกรมนั้นจะไม่กล่าวถึงในที่นี้ผู้ที่สนใจศึกษาในเรื่องนี้สามารถศึกษาได้ตามบรรณานุกรมที่อยู่ท้ายเล่ม

2.5.2.5 สร้างคลาสไดอะแกรม

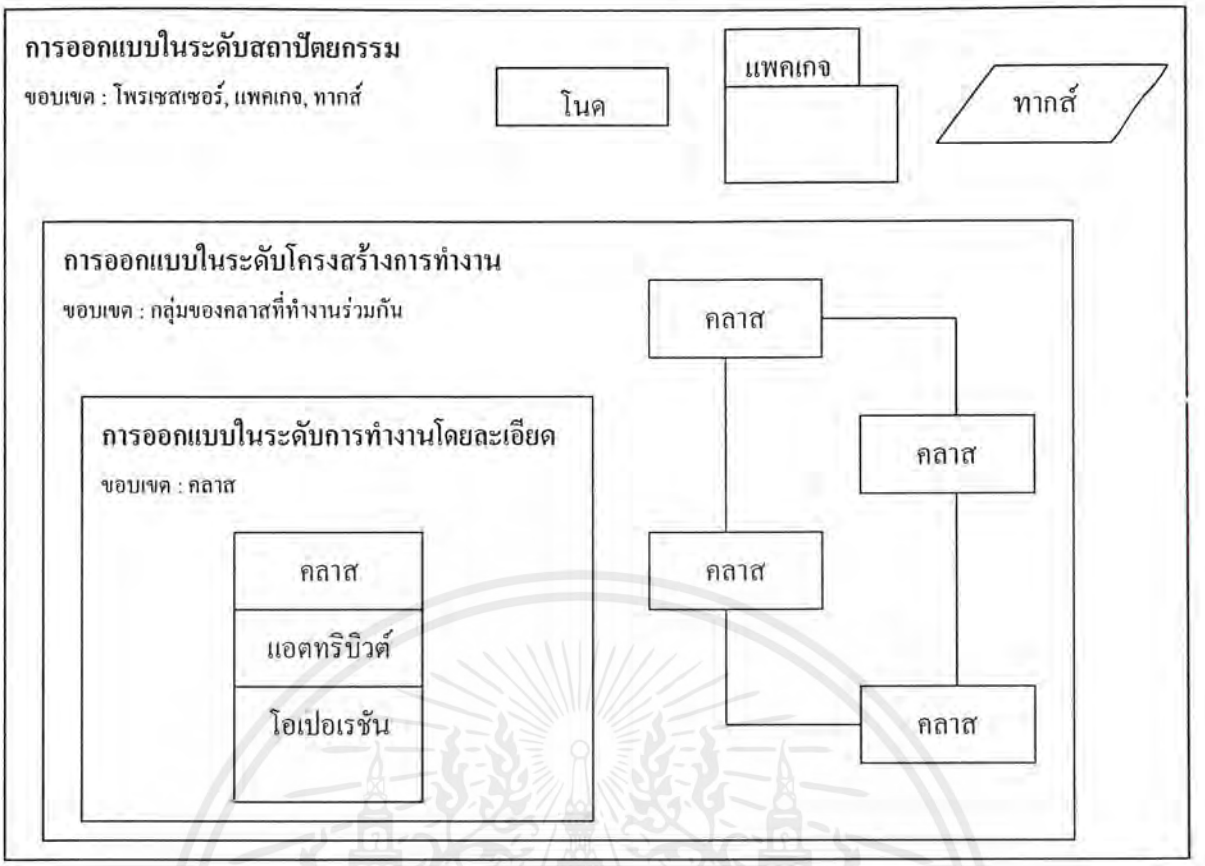
เมื่อเราได้ออบเจ็กต์และความสัมพันธ์ของออบเจ็กต์แล้ว ขั้นต่อไปคือทำการรวมออบเจ็กต์ที่มีเอกลักษณ์ที่เหมือนกันมารวมเป็นคลาส ส่วนความสัมพันธ์ระหว่างออบเจ็กต์ซึ่งมี ลิงค์ และแอกรีเกชันเมื่ออยู่ในรูปแบบคลาสความสัมพันธ์นี้ยังคงอยู่ในคลาสเหมือนกัน ทำให้ลักษณะของคลาสไดอะแกรมที่ได้มีความใกล้เคียงกับออบเจ็กต์ไดอะแกรม

ส่วนความสัมพันธ์ที่จะมีแต่ในคลาสไดอะแกรมแต่ไม่มีในออบเจ็กต์ไดอะแกรมคือ ความสัมพันธ์แบบเจเนรอไลเซชัน (Generalization Relationship) ซึ่งคือคุณสมบัติการสืบทอดคุณสมบัติของคลาส โดยคลาสที่สืบทอดคุณสมบัติเราเรียกว่า ซูเปอร์คลาส และคลาสที่ได้รับการสืบทอดคุณสมบัติเราเรียกว่า สับคลาส เช่นคลาสสิ่งมีชีวิตเป็นซูเปอร์คลาสโดยมีสับคลาสเป็นคลาสพืช และคลาสสัตว์ แต่สับคลาสกลับเป็นสับเซตของซูเปอร์คลาสถึงแม้ว่าสับคลาสจะมีคุณสมบัติมากกว่าซูเปอร์คลาสก็ตาม ทั้งนี้เนื่องจากว่าการที่เราเพิ่มคุณสมบัติให้กับสับคลาสนั้นเป็นการเพิ่มคุณสมบัติพิเศษให้กับสับคลาสเข้าไปส่งผลให้สับคลาสมีความเป็นชนิดที่มีความเฉพาะเจาะจงกว่าซูเปอร์คลาส

2.5.3 การออกแบบเชิงวัตถุ (Object-Oriented Design)

ในการออกแบบระบบนี้จะแบ่งการออกแบบออกเป็น 3 ระดับคือ การออกแบบในระดับสถาปัตยกรรม (Architectural Design), การออกแบบในระดับของโครงสร้างการทำงาน (Mechanistic Design) และการออกแบบในระดับการทำงานโดยละเอียด (Detail Design)

ในขั้นตอนการออกแบบนี้ได้มีแนวทางใหม่สำหรับวิธีการทางออบเจ็กต์โอเรียนเท็ต คือการนำดีไซน์แพทเทิร์น (Design Pattern) ซึ่งเป็นรูปแบบในการออกแบบระบบ ผู้ที่ออกแบบระบบสามารถนำดีไซน์แพทเทิร์นที่มีอยู่ไปประยุกต์ใช้กับระบบได้โดยมั่นใจว่าการประยุกต์ใช้ดีไซน์แพทเทิร์นจะทำให้สามารถออกแบบได้อย่างถูกต้อง ทั้งนี้เนื่องจากดีไซน์แพทเทิร์นนั้นเป็นรูปแบบในการออกแบบที่ได้มีผู้คิดค้นและพิสูจน์ไว้แล้วว่าเหมาะสมสำหรับปัญหาใดๆในระบบ แต่อย่างไรก็ดีการประยุกต์ใช้ดีไซน์แพทเทิร์นนั้นก็ต้องคำนึงถึงความเหมาะสมด้วยเช่นกัน และในส่วนของดีไซน์แพทเทิร์นที่ใช้ในขั้นแต่ละขั้นตอนของการออกแบบจะไม่กล่าวถึงในที่นี้



รูปที่ 2-5 การแบ่งระดับของการออกแบบ

2.5.3.1 การออกแบบในระดับสถาปัตยกรรม

การออกแบบในระดับสถาปัตยกรรมเป็นการออกแบบในมุมมองรวมของระบบว่าต้องประกอบด้วยอุปกรณ์ใดบ้างซึ่งเป็นการมองในลักษณะของการออกแบบบนลงล่างโดยในขั้นตอนนี้เป็นระดับบนสุดของขั้นตอนทั้งหมด

การทำงานในขั้นตอนนี้จะทำการกำหนดแพคเกจหรือส่วนย่อยของระบบ (Subsystems) ที่ประกอบด้วย ความสัมพันธ์และการติดต่อสื่อสารระหว่างแพคเกจที่สำคัญในระบบ โดยที่โครงสร้างที่ได้จะต้องง่าย และทำความเข้าใจได้ง่าย

ในการออกแบบในส่วนนี้ จะเกี่ยวข้องกับทั้งซอฟท์แวร์(มองในลักษณะของทากส์(Task)) และฮาร์ดแวร์ ซึ่งทั้งสองส่วนนี้จะต้องสามารถทำงานร่วมกันได้อย่างดี โดยในการออกแบบ โดยการออกแบบจะแสดงถึงส่วนของแพคเกจ โพรเซสเซอร์ และทากส์

2.5.3.2 การออกแบบในระดับของโครงสร้างการทำงาน

เป็นการออกแบบในระดับการเชื่อมต่อกันของคลาสว่าควรอยู่ในลักษณะใด ซึ่งขั้นตอนนี้จะทำการออกแบบลึกลงมาจากชั้นของการออกแบบในระดับสถาปัตยกรรมคือมองว่าภายในทากส์ที่อยู่ภายในอุปกรณ์ต่างๆ ประกอบด้วยคลาสอะไรบ้างและมีการเชื่อมต่อของแต่ละคลาสอย่างไร

2.5.3.3 การออกแบบในระดับการทำงานโดยละเอียด

เป็นการออกแบบในลักษณะของโครงสร้างข้อมูลภายในคลาส ซึ่งมีความใกล้เคียงกับส่วนของการอิมพลีเมนต์มาก ซึ่งจุดประสงค์ของการออกแบบในขั้นตอนนี้คือวางโครงสร้างภายในคลาสให้มีความถูกต้องและเหมาะสมเพื่อให้แต่ละคลาสมีความถูกต้องและเมื่อนำแต่ละคลาสไปรวมกันเป็นระบบใหญ่จะทำให้ได้ระบบที่ได้ มีความผิดพลาดน้อยลงด้วย

2.6 การพัฒนาโปรแกรมเชิงวัตถุด้วยวิชวลเบสิก 6

วิชวลเบสิก 6 นั้นเป็นออบเจ็กต์เบส (Object-Base Programming Language) ซึ่งสิ่งที่มีความแตกต่างจากออบเจ็กต์โอเรียนเต็ดที่ได้กล่าวมาข้างต้นคือ ออบเจ็กต์เบสนั้นไม่สามารถทำการสืบทอดคุณสมบัติได้อย่างสมบูรณ์ กล่าวคือออบเจ็กต์เบสจะทำการสืบทอดคุณสมบัติได้เฉพาะอินเตอร์เฟสเท่านั้น (Interface Inheritance)ส่วนที่เป็นชุดคำสั่ง (Code) นั้นจะไม่ถูกสืบทอดมาจากคลาสหลัก

แต่การเป็นออบเจ็กต์เบสนั้นก็ไม่ได้หมายความว่าวิชวลเบสิกไม่สามารถใช้การพัฒนาเชิงวัตถุได้ และด้วยคุณสมบัติของการสืบทอดเฉพาะอินเตอร์เฟสของวิชวลเบสิกนี้ทำให้โครงสร้างของออบเจ็กต์ที่ได้มีความยืดหยุ่นมากกว่าเดิม (Loose Coupling) เพราะว่าถ้ามีการสืบทอดคุณสมบัติที่เป็นชุดคำสั่งมาด้วยการเปลี่ยนแปลงแก้ไขที่คลาสหลักนั้นจะส่งผลกระทบต่อทุกคลาสที่สืบทอดมาจากคลาสหลักนั้น และด้วยคุณสมบัตินี้ของวิชวลเบสิกได้ถูกสร้างขึ้นมาเป็นมาตรฐานของการสร้าง COM (Component Object Model) ขึ้นมาโดยมีวัตถุประสงค์ที่ว่า COM ที่ได้ไม่ว่าจะถูกสร้างขึ้นจากภาษาใดก็ตามที่สนับสนุนการสร้าง COM นี้ ต้องทำงานร่วมกันได้ ซึ่งในรายละเอียดของ COM จะกล่าวถึงในบทต่อไป และสำหรับการสืบทอดคุณสมบัติเฉพาะอินเตอร์เฟส การทำโพลีมอร์ฟิซึม และการนำกลับมาใช้ในรูปแบบของอินเตอร์เฟส (Reuse of Interface)โดยใช้วิชวลเบสิกนั้นสามารถดูได้ที่ภาคผนวก การโปรแกรมเชิงวัตถุด้วยวิชวลเบสิก 6

บทที่ 3
เทคโนโลยี COM และ DCOM

3.1 COM (Component Object Model)

คอมโพเนนต์ (Component) คือ ส่วนของโปรแกรมซึ่งสามารถในการทำงานอย่างใดอย่างหนึ่งสำเร็จได้ด้วยตนเองและสามารถนำไปใช้ร่วมกับคอมโพเนนต์อื่นได้โดยที่คอมโพเนนต์จะต้องมีคุณสมบัติเฉพาะของตนเอง มีความยืดหยุ่นในการนำคอมโพเนนต์อื่นมาทดแทนหรือแก้ไขได้โดยง่าย และสามารถนำคอมโพเนนต์ที่สร้างขึ้นมากลับมาใช้ใหม่ได้ แต่ในการสร้างคอมโพเนนต์จะต้องมีมาตรฐานในการสร้างเพื่อให้ทุกคนสามารถสร้างคอมโพเนนต์ที่เข้ากันและติดต่อกันได้ คอมโพเนนต์เป็นส่วนซอฟต์แวร์ที่อยู่ในรูปแบบไบนารีที่สามารถประกอบกับคอมโพเนนต์อื่น ๆ เพื่อสร้างซอฟต์แวร์ที่มีขนาดใหญ่ โดยความสัมพันธ์ระหว่างคอมโพเนนต์จะผ่านทางกลุ่มอินเตอร์เฟส (เมธอดและแอททริบิวต์) จุดสำคัญของคอมโพเนนต์ คือ เมื่อมีการนำคอมโพเนนต์มาประกอบกันแล้วสามารถใช้งานได้เลยโดยไม่ต้องมีการคอมไพล์ใหม่

COM (Component Object Model) เป็นมาตรฐานที่ไมโครซอฟท์คิดขึ้นเพื่อใช้ในการสร้างซอฟต์แวร์คอมโพเนนต์ เทคโนโลยี COM เป็นรายละเอียดของการสร้างและการคอมไพล์ซอฟต์แวร์คอมโพเนนต์ COM ไม่ใช่ภาษาที่ใช้ในการเขียนโปรแกรม ไม่ใช่ library ของโค้ด และไม่ใช่คอมไพล์เลอร์ รายละเอียดของมาตรฐาน COM จะบอกถึงวิธีการสร้างคอมโพเนนต์ว่าทำอย่างไรคอมโพเนนต์เหล่านั้นจึงสามารถทำงานติดต่อระหว่างคอมโพเนนต์ได้และจะบอกถึงวิธีการสร้างคอมโพเนนต์เป็นไบนารี

COM สามารถสร้างคอมโพเนนต์ที่เปลี่ยนแปลงได้ง่ายเพื่อรองรับการทำงานในอนาคต แต่เดิมปัญหาในการพัฒนาซอฟต์แวร์เชิงคอมโพเนนต์มีปัญหาที่สำคัญ คือ ปัญหาด้านเวอร์ชั่น(Versioning) ใน COM จะแก้ปัญหานี้โดยจะไม่อนุญาตให้มีการเปลี่ยนแปลงอินเตอร์เฟสของคอมโพเนนต์การปรับเปลี่ยนหรือเพิ่มความสามารถให้กับคอมโพเนนต์จะเป็นในลักษณะแก้ไขการทำงานของอินเตอร์เฟส หรือการเพิ่มอินเตอร์เฟสใหม่ COM อนุญาตให้คอมโพเนนต์สามารถมีหลายอินเตอร์เฟสได้

COM มีลักษณะที่สำคัญดังนี้

- โค้ดของคอมโพเนนต์ (Component code) เป็นส่วนที่ทำงานตามจุดประสงค์ของคอมโพเนนต์
- อินเตอร์เฟส(Interfaces) เป็นส่วนโปรแกรมติดต่อกับคอมโพเนนต์ อินเตอร์เฟส เป็นสิ่งที่บอกคอมโพเนนต์อื่นว่าคอมโพเนนต์นั้นมีหน้าที่อะไรและเรียกวิธีใช้คอมโพเนนต์อย่างไร
- GUIDs (Globally Unique Identifier) เป็นสิ่งที่ระบุถึงคอมโพเนนต์ COM และอินเตอร์เฟสของคอมโพเนนต์ซึ่ง GUIDs จะเป็นหมายเลขซึ่งไม่ซ้ำกัน
- Binary compatibility หมายถึง คอมโพเนนต์เป็นรูปแบบไบนารีทำให้สามารถนำไปใช้ได้กับหลายๆรูปแบบ

COM คือ การนำคอมโพเนนต์รวมกับอินเตอร์เฟส ในการสร้างคอมโพเนนต์สิ่งที่จะต้องพิจารณาเป็นสิ่งแรกคือ อินเตอร์เฟส เพราะส่วนซับซ้อนของการสร้างโปรแกรมแบบคอมโพเนนต์เบส (component-based) คือการติดต่อระหว่างคอมโพเนนต์ หากเราต้องการจะแก้ไขคอมโพเนนต์แต่ไม่ได้แก้ไขอินเตอร์เฟสของคอมโพเนนต์นั้น เมื่อคอมไพล์เสร็จแล้วสามารถนำมาใช้ได้เลยโดยไม่ต้องแก้ไขส่วนอื่นเพิ่ม แต่ถ้าแก้ไขอินเตอร์เฟสเมื่อคอมไพล์เสร็จจะได้เป็นคอมโพเนนต์ใหม่ซึ่งแตกต่างจากคอมโพเนนต์เดิม (GUIDs เปลี่ยน) ทำให้ต้องแก้ไขการติดต่อระหว่างคอมโพเนนต์ด้วย ดังนั้นการออกแบบอินเตอร์เฟสตั้งแต่เริ่มสร้างคอมโพเนนต์จึงเป็นสิ่งที่ควรให้ความสำคัญ

ประโยชน์ของ COM

- คอมโพเนนต์ COM สามารถทดแทนกันได้ง่าย
- คอมโพเนนต์ COM สามารถเปลี่ยนแปลงตามความต้องการที่เปลี่ยนแปลงได้ เช่น การแก้ไขอัตราภาษี
- คอมโพเนนต์ COM สามารถนำกลับมาใช้ใหม่ได้ COM สนับสนุนการนำกลับมาใช้ใหม่ในระดับอินเตอร์เฟสแต่ไม่สนับสนุนในระดับโค้ด
- คอมโพเนนต์ COM สนับสนุนการพัฒนาแบบขนาน ผู้พัฒนาหลายคนสามารถพัฒนาคอมโพเนนต์ของโปรแกรมไปพร้อมกันได้

ปัญหาของ COM

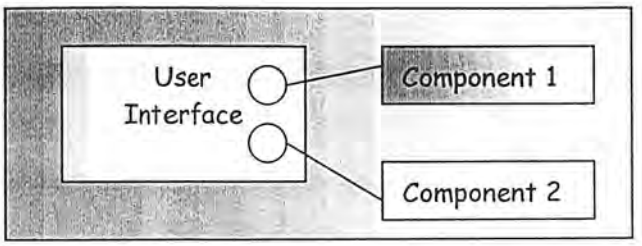
- การเปลี่ยนเวอร์ชันของคอมโพเนนต์ทำให้ต้องแก้ไขในส่วนอื่นๆ
- อินเตอร์เฟสเดิมจะไม่สามารถเปลี่ยนแปลงได้
- อินเตอร์เฟสของ COM จะต้องออกแบบอย่างระมัดระวัง

3.2 DCOM (Distributed COM)

หลักการของ DCOM จะทำงานเหมือน COM ทุกอย่าง แต่แตกต่างที่ DCOM สามารถทำงานได้บนเครื่องคอมพิวเตอร์มากกว่า 1 เครื่องซึ่งหมายความว่าโพรเซส ของ DCOM สามารถทำงานข้าม Process Space ได้ ซึ่งแตกต่างจาก COM ที่ต้องรันอยู่ใน Process Space เดียวกัน

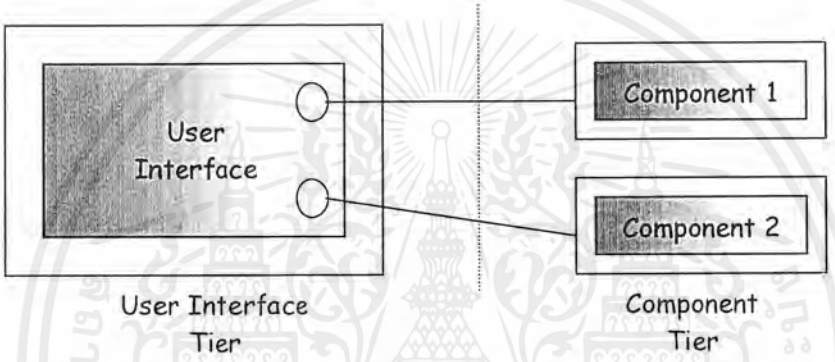
In-Process และ Out-Of-Process

In-Process คือ การที่คอมโพเนนต์ทำงานอยู่ใน Process space เดียวกับแอปพลิเคชั่นที่เรียกใช้คอมโพเนนต์ COM นั้นอยู่ ซึ่งก็คือการแบ่งการใช้ Address Space เดียวกันนั่นเอง ซึ่งคอมโพเนนต์ ที่ทำงานแบบ In-process นั้นจะสร้างเป็นไฟล์ชนิด ActiveX DLL



รูปที่ 3-1 แสดงการทำงานแบบ In-process

Out-Of-Process คือ การที่คอมโพเนนต์สามารถทำงานอยู่บนต่าง Process space กับแอปพลิเคชั่นที่เรียกใช้ได้ ซึ่งทำให้คอมโพเนนต์กับแอปพลิเคชั่นนั้นสามารถอยู่ต่างเครื่องกันได้ ซึ่งการสร้างคอมโพเนนต์ที่ทำงานแบบ Out-Of-Process นั้นสามารถสร้างโดยกำหนดไฟล์เป็นชนิด ActiveX EXE



รูปที่ 3-2 แสดงการทำงานแบบ Out-of-Process

การสร้าง DCOM Component

DCOM สร้างขึ้นโดยกำหนดชนิดของโปรแกรมเป็นแบบActiveX EXE DCOM มีการทำงานแบบ Out-Of-Process ซึ่งแอปพลิเคชั่นจะทำงานอยู่ที่ฝั่งเซิร์ฟเวอร์และที่ฝั่งไคลเอนต์จะทำการสร้าง DCOM Instance ขึ้นมา เมื่อจะนำแอปพลิเคชั่นไปใช้จะต้องทำการรีจิสเตอร์คอมโพเนนต์ DCOM ที่ฝั่งไคลเอนต์ด้วย

การส่งข้อมูลผ่าน DCOM

การส่งข้อมูลผ่าน DCOM มีความแตกต่างกับการส่งข้อมูลผ่านคอมโพเนนต์แบบ COM อยู่ที่เวลาของการตอบสนอง (Response Time) และความหนาแน่นของการใช้เครือข่าย (Traffic) ซึ่งต้องให้ความสำคัญกับการส่งข้อมูลนี้ด้วย

- การส่งข้อมูลเข้าสู่คอมโพเนนต์ DCOM นั้นจะทำโดยการผ่านค่าเป็นอาร์กิวเมนต์ (Argument) เข้าสู่เมธอดของ DCOM และควรใช้การส่งผ่านแบบ ByVal (ถ้าค่าของ อาร์กิวเมนต์ นั้นเปลี่ยนแปลงจะไม่ส่งผลกลับมา) ซึ่งทำให้สามารถยกเลิกการติดต่อได้ (Drop Connection) เมื่อเราส่งข้อมูลเข้าสู่ DCOM แล้วซึ่งเป็นการความหนาแน่นของการใช้เครือข่าย

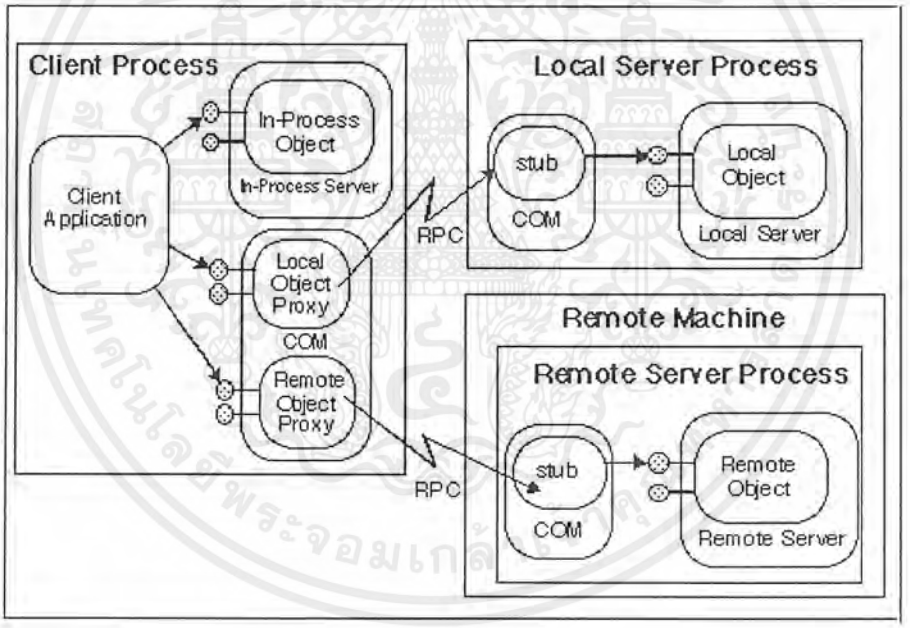
- การส่งข้อมูลออกจากคอมโพเนนต์ DCOM จะใช้การส่งข้อมูลออกจาก ฟังก์ชั่นของ เมธอด ในคอมโพเนนต์ DCOM ซึ่งรูปแบบของการส่งข้อมูลออกที่นิยมใช้มีอยู่ 3 รูปแบบ คือ อาเรย์, สตริง, เรคคอร์ดเซต (Recodset)

Marshaling

เป็นวิธีการในการส่งผ่านพารามิเตอร์ (อาร์กิวเมนต์ที่ส่งไปกับฟังก์ชั่น) ข้ามขอบเขตของโพรเซส ซึ่งถ้าการทำงานเป็นแบบ In-Process อย่างคอมโพเนนต์ COM นั้นจะใช้สแต็กเป็นที่เก็บพารามิเตอร์ แต่ถ้าการทำงานเป็น Out-Of-Process นั้นไม่สามารถทำอย่าง COM ได้จึงต้องใช้ Marshaling

การทำงานของ Marshaling หรือเรียกอีกอย่างว่า Proxy/Stub โดย Proxy Function นั้นจะอยู่ที่ไคลเอนต์ทำหน้าที่เรียบเรียงและจัดส่งข้อมูล ส่วน Stub Function นั้นจะอยู่ที่คอมโพเนนต์ทำหน้าที่รับข้อมูลและส่งให้คอมโพเนนต์

สำหรับการเขียนโค้ดในส่วนของ Proxy/Stub ในวิชวลเบสิกนั้นไม่จำเป็นต้องเขียนเพราะ วิชวลเบสิกทำงานในส่วนนี้ให้แล้ว



รูปที่ 3-3 แสดงการทำงานแบบ Proxy/Stub ของ DCOM

การจัดการความปลอดภัยของ DCOM

ในเรื่องของการจัดการความปลอดภัยของ DCOM นั้นสามารถใช้ DCOM Configuration Utility จัดการได้ ซึ่งจะมีการกำหนดค่าดังนี้

- Default Access Permissions : กำหนดสิทธิให้สามารถเข้าใช้งานแอปพลิเคชั่นนั้นได้ โดยไม่มีการเปลี่ยนแปลง

- Default Launch Permissions : กำหนดสิทธิให้สามารถใช้แอปพลิเคชั่นติดต่อกับคอมโพเนนต์นั้นได้ โดยไม่มีการเปลี่ยนแปลง
- Default Configuration Permissions : กำหนดสิทธิให้สามารถทำการเปลี่ยนแปลงคอมโพเนนต์นั้นได้

  ผู้ที่จะสามารถใช้ DCOM Configuration Utility นี้ได้ โดยปกติจะเป็น Administrator

บทที่ 4
ไมโครซอฟท์ทรานแซคชั่นเซิร์ฟเวอร์

4.1 ทรานแซคชั่น

ทรานแซคชั่น (Transaction) คือ งานหรือกลุ่มของงานที่ต้องทำให้เสร็จทั้งหมดหรือไม่ทำเลย โดยทรานแซคชั่นที่สำเร็จจะรับประกันได้ว่างานหรือกลุ่มของงานในทรานแซคชั่นนั้นเสร็จสิ้นทั้งหมด ซึ่งงานอาจจะมีหลายขั้นตอนหรือมีคอมพิวเตอร์หลายเครื่องที่เกี่ยวข้อง ส่วนทรานแซคชั่นที่ไม่สำเร็จจะปฏิบัติเหมือนกับไม่เคยเกิดทรานแซคชั่นนั้น ปกติแล้วทรานแซคชั่นจะถูกควบคุมโดยระบบฐานข้อมูล ทรานแซคชั่นอาจประกอบด้วยหลายขั้นตอน ได้แก่ การดึงข้อมูลจากฐานข้อมูล (Retrieve) การแก้ไขข้อมูล (Modify) หรือการจัดเรียงข้อมูลในฐานข้อมูล(Sort) เมื่อกิจกรรมที่เกี่ยวข้องกับฐานข้อมูลเสร็จสิ้น แอปพลิเคชั่นชันจะต้องสั่งคอมมิต (Commit) ทรานแซคชั่นซึ่งจะบอกแก่ระบบฐานข้อมูลว่าให้แก้ไขข้อมูลอย่างถาวร แต่ถ้ากิจกรรมที่กระทำต่อระบบฐานข้อมูลผิดพลาดจะต้องแก้ไขระบบฐานข้อมูลให้อยู่ในสภาพเดิมก่อนที่จะเริ่มทรานแซคชั่น

4.1.1 คุณสมบัติของทรานแซคชั่น

คุณสมบัติของทรานแซคชั่นประกอบด้วย 4 ข้อ คือ

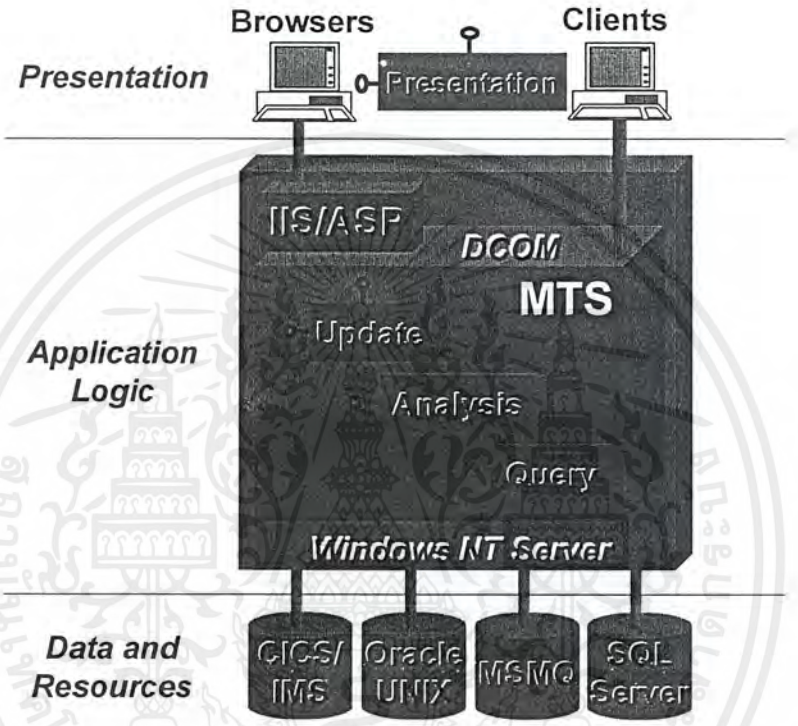
- Atomicity คือ ความเป็นหนึ่งเดียวกันของทรานแซคชั่น ถ้าเสร็จต้องเสร็จทั้งทรานแซคชั่น ถ้าไม่เสร็จต้องเหมือนกับว่าไม่เคยเกิดทรานแซคชั่นขึ้น
- Consistency คือ สถานะของฐานข้อมูลจะต้องตรงกับสถานะที่คอมโพเนนต์ต้องการ
- Isolation คือ ทรานแซคชั่นต้องเป็นอิสระจากกัน โดยไม่ขึ้นกับสถานะของทรานแซคชั่นอื่น
- Durability คือ หากทรานแซคชั่นถูกคอมมิตแล้ว การเปลี่ยนแปลงของทรานแซคชั่นนั้นจะต้องคงอยู่โดยข้อมูลไม่สูญหาย

ในกรณีการเข้าใช้ฐานข้อมูลแห่งเดียวจะควบคุมทรานแซคชั่นของแอพพลิเคชันได้โดยสะดวก แต่ในกรณีที่มีฐานข้อมูลหลายแห่งซึ่งอยู่บนเครื่องคอมพิวเตอร์ต่างเครื่อง การควบคุมทรานแซคชั่นจะทำได้ยาก เช่น ทรานแซคชั่นสามารถคอมมิตฐานข้อมูลแรกสำเร็จ แต่ฐานข้อมูลที่สองไม่สามารถคอมมิตได้ เนื่องจากเครื่องเซิร์ฟเวอร์เสีย หากเกิดกรณีนี้จะต้องสามารถควบคุมให้ทรานแซคชั่นไม่เกิดขึ้นทั้งหมด โดยจะต้องลบข้อมูลที่ได้แก้ไขในฐานข้อมูลแรก การเขียนโค้ดเพื่อมาควบคุมคุมทรานแซคชั่นเป็นสิ่งที่ทำได้ยาก เพราะจะต้องควบคุม ในทุกขั้นตอนทำให้เสียเวลาในการพัฒนามาก ดังนั้นไมโครซอฟท์จึงได้พัฒนา MTS ขึ้นมาช่วยจัดการในด้านการควบคุมคุมทรานแซคชั่นที่เกิดขึ้น

4.2 สถาปัตยกรรมของ MTS (Microsoft Transaction Server)

MTS เป็น Environment ที่ถูกพัฒนาเพื่อใช้ในการสร้างและการแจกจ่าย (deploy) แอปพลิเคชั่น ซึ่งถูกพัฒนาขึ้นบนพื้นฐานของเทคโนโลยี COM

ในการใช้แอปพลิเคชั่นบน MTS คอมโพเนนต์ที่ทำงานด้าน application logic จะทำงานภายใต้การควบคุมของ MTS ที่ติดตั้งอยู่บนเครื่องเซิร์ฟเวอร์และจะถูกเรียกใช้โดยคอมโพเนนต์ที่จัดการด้านแสดงผลซึ่งทำงานอยู่บนเครื่องไคลเอนต์ผ่านทางเทคโนโลยี DCOM



รูปที่ 4-1 แสดงสถาปัตยกรรมของแอปพลิเคชั่นที่ติดตั้งบน MTS

คอมโพเนนต์ที่จัดการด้าน Application logic สามารถเรียกใช้ข้อมูลได้จากหลายฐานข้อมูลซึ่งอาจจะเป็นผลิตภัณฑ์ชนิดเดียวกันหรือต่างชนิดกันก็ได้ การเข้าถึงฐานข้อมูลและการจองทรัพยากรจะถูกจัดการโดย MTS Resource Dispensers ซึ่งจะกล่าวถึงต่อไป จากรูปที่ 4-1แสดงให้เห็นส่วนแสดงผลของไคลเอนต์ซึ่งอาจจะเป็นเบราเซอร์หรือโปรแกรมไคลเอนต์ ถ้าเป็นเบราเซอร์จะเรียกใช้คอมโพเนนต์ที่ติดตั้งบน MTS ผ่านทาง IIS (Internet Information Server) คอมโพเนนต์ที่อยู่บน MTS จะทำหน้าที่ประมวลแก้ไขข้อมูลและเรียกใช้ข้อมูล โดยที่ MTS จะต้องถูกติดตั้งลงบนระบบปฏิบัติการวินโดว์เอ็นทีเซิร์ฟเวอร์

ส่วนประกอบของ MTS

1. MTS execution environment

ทำหน้าที่จัดการการสร้างและทำลายออบเจ็กต์ ,เธรด และจัดการด้านความปลอดภัย ซึ่งจะต้องทำงานติดต่อกับ คอมโพเนนต์, resource dispensers และฐานข้อมูลที่ถูกใช้งานจากทรานแซคชั่น

2. MTS Explorer
   เป็นเครื่องมือที่ใช้ในการจัดการคอมโพเนนต์และแพคเกจ โดยสามารถใช้เพื่อติดตั้งคอมโพเนนต์หรือกระจายคอมโพเนนต์ และสามารถใช้ในการสร้างแพคเกจเพื่อนำไปติดตั้งที่เครื่องเซิร์ฟเวอร์หรือเครื่องไคลเอนต์ รวมทั้งสามารถใช้ติดตามการทำงาน ดูสถิติเกี่ยวกับการใช้งานและโหลดการทำงานของเครื่องเซิร์ฟเวอร์
3. APIs (Application programming interfaces)
   เป็นสิ่งที่นำไปใช้กับคอมโพเนนต์เพื่อให้สามารถทำงานร่วมกับ MTS ได้ โดยคอมโพเนนต์จะส่งข้อความผ่านทาง APIs เพื่อบอกกับ MTS ว่าการทำงานตามทรานแซคชั่นสำเร็จหรือไม่
4. Resource dispensers
   เป็นสิ่งที่จัดการเกี่ยวกับการใช้ทรัพยากร เช่น การเชื่อมจ่อกับฐานข้อมูล การเชื่อมต่อกับเครือข่าย
5. Application components
   คือ คอมโพเนนต์ที่สร้างขึ้นเพื่อทำงานตามจุดประสงค์บางอย่าง

4.3 รูปแบบการสร้างคอมโพเนนต์เพื่อทำงานบน MTS

4.3.1 ข้อกำหนดในการสร้างคอมโพเนนต์เพื่อทำงานร่วมกับ MTS

ผู้พัฒนาสร้างคอมโพเนนต์ของส่วนแสดงผลและส่วนประมวลผลได้โดยใช้เครื่องมือที่สามารถสร้างไฟล์ชนิด DLL (Dynamic-link libraries) การที่จะสร้างคอมโพเนนต์เพื่อใช้งานบน MTS จะต้องสร้างตามข้อกำหนดดังนี้

1. คอมโพเนนต์ของ MTS ต้องเป็นแบบ COM DLLs ซึ่งเป็น in-process ห้ามใช้ DCOM คอมโพเนนต์ซึ่งเป็น out-of-process หากว่ามี DCOM คอมโพเนนต์ อยู่แล้วต้องแก้ไขให้เป็น COM DLLs เสียก่อน
2. คอมโพเนนต์ของ MTS ต้องเขียนตามข้อกำหนดของการเขียนโปรแกรมและการจัดการทรัพยากร ได้แก่
   - เขียนคอมโพเนนต์ให้เหมือนกับการใช้กับไคลเอนต์เดียว ส่วนการใช้งานกับหลายๆ ไคลเอนต์ MTS จะทำหน้าที่จัดการให้
   - เขียนคอมโพเนนต์ให้ร้องขอการใช้ทรัพยากรเมื่อต้องการจะใช้เท่านั้น
   - เขียนคอมโพเนนต์ให้คืนการใช้ทรัพยากรให้เร็วที่สุด
   - คอมโพเนนต์ไม่ควรคงสถานะของข้อมูลไว้หลังจากจบทรานแซคชั่น

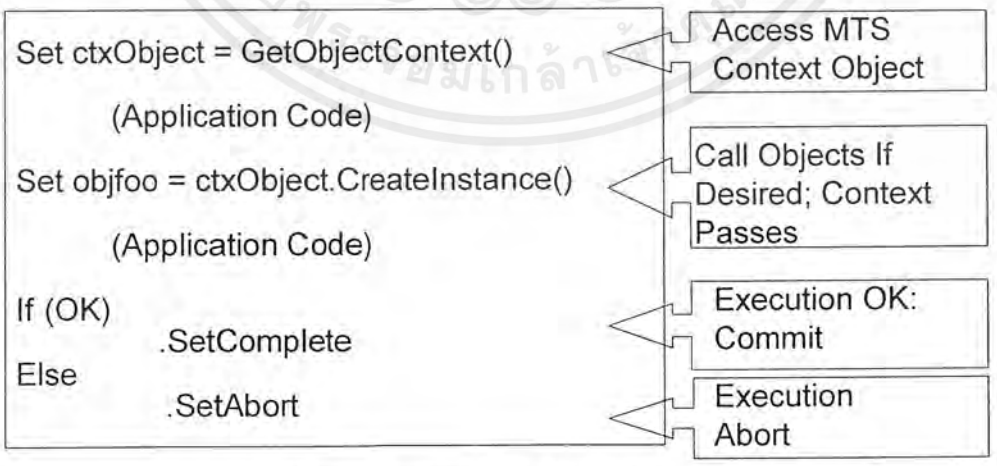
3. คอมโพเนนต์ของ MTS ต้องบอกสถานะแก่ MTS โดยการเรียกใช้ MTS APIs วิธีการเขียนมีดังนี้

- ก่อนที่จะเรียกใช้ APIs จะต้องอ้างอิงเพื่อให้คอมโพเนนต์สามารถใช้ประโยชน์จาก MTS ในการสนับสนุนทรานแซคชั่นและความปลอดภัย เมื่อเราได้สร้าง object context แล้วเราจะสามารถเรียกใช้ APIs call อื่นๆได้ การอ้างอิงทำได้ดังนี้

```
Dim x As ObjectContext
Set x = GetObjectContext()
```

- MTS มีกลไกการใช้ทรัพยากรที่เรียกว่า just-in-time (JIT) ซึ่งหมายถึงใช้เมื่อต้องการจริงเท่านั้น และการคืนทรัพยากรต้องทำทันทีที่เป็นไปได้ โดยเราสามารถเรียกใช้ฟังก์ชันดังนี้
  - เมื่อคอมโพเนนต์ประมวลผลสำเร็จจะต้องเรียกใช้ฟังก์ชัน Setcomplete เพื่อบอก MTS ให้คอมมิตงานทุกอย่างที่อยู่ใน ทรานแซคชั่นและบอกให้ MTS สามารถใช้ทรัพยากรที่ถูกจองโดยคอมโพเนนต์ที่ประมวลผลสำเร็จนั้นได้
  - ถ้าคอมโพเนนต์ไม่สามารถประมวลผลได้สำเร็จจะต้องเรียกใช้ฟังก์ชัน Setabort เพื่อบอก MTS ให้ยกเลิกงานทุกอย่างที่อยู่ในทรานแซคชั่นและกู้ข้อมูลเดิมก่อนทำทรานแซคชั่นนั้น

4.3.2 ตัวอย่างการสร้างคอมโพเนนต์เพื่อทำงานร่วมกับ MTS



รูปที่ 4-2 แสดงตัวอย่างการสร้างคอมโพเนนต์เพื่อใช้บน MTS

4.4 บทบาทของ MTS ในการพัฒนาแอปพลิเคชั่น

4.4.1 บทบาทของ MTS ในการพัฒนาแอปพลิเคชั่นแบบ 3-เทียร์

เทคโนโลยี COM ทำให้การพัฒนาแอปพลิเคชั่นแบบ 3-เทียร์ง่ายขึ้น โดยที่ MTS ถูกนำมาใช้ในส่วนของมิดเดิลเทียร์ซึ่งทำหน้าที่ควบคุมการทำงานของ COM คอมโพเนนต์ที่อยู่ในรูปแบบไฟล์ .DLL เมื่อคอมโพเน นต์ของส่วนแสดงผลเรียกใช้คอมโพเนนต์ของส่วนประมวลผลที่อยู่บนเครื่องเซิร์ฟเวอร์ MTS จะจัดการเกี่ยวกับทรานแซคชั่นที่เกิดขึ้นและจัดการเกี่ยวกับการใช้ทรัพยากรต่างๆให้อย่างอัตโนมัติ โดยผู้พัฒนาแอปพลิเคชั่นไม่ต้องเสียเวลาในการพัฒนามากนักเพียงแต่เรียกใช้ API ของ MTS ในการจัดการทรานแซคชั่น

ประโยชน์ของ MTS ได้แก่

- MTS รับผิดชอบงานด้านทรานแซคชั่นทำให้ในการสร้างคอมโพเนนต์ ผู้พัฒนาไม่ต้องจัดการด้านทรานแซคชั่นเอง
- MTS จัดการด้านการจองและการคืนทรัพยากรต่างๆ เช่น การขอใช้เครือข่าย
- MTS Explorer ทำให้การติดตั้งคอมโพเนนต์บนเครื่องเซิร์ฟเวอร์และไคลเอนต์ง่ายขึ้น

4.4.2 บทบาทของ MTS ในการพัฒนาแอปพลิเคชั่นบนเว็บ

ในปัจจุบันอินเตอร์เน็ตเข้ามามีบทบาทในการดำเนินธุรกิจและการติดต่อสื่อสารมากขึ้น ตัวอย่างเช่น ระบบพาณิชย์อิเล็กทรอนิกส์ซึ่งเป็นระบบที่ทำให้ผู้ใช้ไม่ต้องซื้อขายสินค้าแบบเดิมอีกต่อไป เพียงแค่ใช้เว็บเบราเซอร์ ที่ติดตั้งอยู่บนเครื่องคอมพิวเตอร์และเครือข่ายอินเตอร์เน็ตก็สามารถซื้อสินค้าที่ต้องการได้ ในอดีตเว็บเพจที่ให้บริการเป็นแบบสแตติกซึ่งผู้ใช้ไม่สามารถติดต่อกับเครื่องเซิร์ฟเวอร์ได้ แต่ในปัจจุบันได้พัฒนาให้เว็บเพจสามารถรับข้อมูลจากผู้ใช้ได้เพื่อนำข้อมูลเหล่านั้นมาประมวลผลแล้วส่งผลลัพธ์กลับไปแสดงบนเว็บเบราเซอร์ได้

MTS ถูกนำมาใช้ร่วมกับ IIS 4.0 เพื่อสร้างแอปพลิเคชั่นแบบ 3-เทียร์ที่มีส่วนติดต่อกับผู้ใช้เป็นเว็บเบราเซอร์ IIS จัดการส่วนติดต่อกับผู้ใช้แบบ interactive และจัดการ ASP (Active Server Page) ซึ่งเป็นไฟล์ข้อมูลที่ประกอบด้วยคำสั่งของ HTML และสคริปต์ของวิชวลเบสิกหรือจาวาสคริปต์ ผู้พัฒนาเว็บจะใช้ ASP เป็นมาตรฐานในการสร้างแอปพลิเคชั่น เมื่อผู้ใช้กรอกข้อมูลบนฟอร์มที่มี URL ลงท้ายด้วย .ASP แล้วส่งข้อมูลไปยังเว็บเซิร์ฟเวอร์ IIS จะประมวลผล สคริปต์ของ ASP ซึ่งสามารถทำการคำนวณ, การติดต่อฐานข้อมูลผ่านทาง ODBC และส่งข้อมูล ในรูปแบบแท็ก html เพื่อแสดงผลเบราเซอร์

ประโยชน์ของ MTS ในการพัฒนาแอปพลิเคชั่นบน เว็บ คือ เพิ่มความสามารถให้คอมโพเนนต์ สามารถประมวลผลตาม application logic, ติดต่อกับฐานข้อมูลผ่านทางไดรเวอร์ของ ODBC, ติดต่อกับเครื่องคอมพิวเตอร์เมนเฟรมผ่านทาง SNA Server, รับส่งข้อมูลผ่านทาง MSMQ (Microsoft Message Queue) และสามารถเรียกใช้คอมโพเนนต์อื่นทำให้เกิดการนำกลับมาใช้ (reuse)

เนื่องจาก MTS สนับสนุนทรานแซคชั่นโดยอัตโนมัติ ถ้าเกิดข้อผิดพลาด MTS จะทำการกู้ข้อมูลเดิมก่อนการแก้ไขทำให้ข้อมูลมีความถูกต้องและ MTS ยังสามารถควบคุมให้การตอบสนองแก่ผู้ใช้จำนวนมากเป็นไปอย่างรวดเร็ว

บทที่ 5
ระบบจัดการฐานข้อมูลเชิงวัตถุสัมพันธ์

5.1 ระบบจัดการฐานข้อมูลเชิงวัตถุสัมพันธ์

ระบบจัดการฐานข้อมูลเชิงวัตถุสัมพันธ์ (Object-Relational Database Management System : ORDBMS) เป็นระบบจัดการฐานข้อมูลที่มีความสามารถของระบบจัดการฐานข้อมูลเชิงสัมพันธ์ (Relational Database Management System : RDBMS) และความสามารถของระบบจัดการฐานข้อมูลแบบออบเจ็กต์ (Object-Oriented Database Management System) ไว้ด้วยกันซึ่งแยกอธิบายได้ดังนี้

5.1.1 ความสามารถของระบบจัดการฐานข้อมูลเชิงสัมพันธ์

มีความสามารถในการเก็บ และจัดการ ข้อมูลที่อยู่ในลักษณะความสัมพันธ์แบบรีเลชัน ซึ่งตัวอย่างหนึ่งของความสัมพันธ์แบบรีเลชันนี้คือ ตาราง โดยมีความสัมพันธ์กันระหว่างแถว (Row) และสดมภ์ (Column) ตัวอย่างของระบบจัดการฐานข้อมูลแบบนี้คือ ระบบจัดการฐานข้อมูลที่ใช้ภาษา SQL เป็นหลัก (SQL-Base Database Management System) ซึ่งมีคุณสมบัติดังนี้

1. โครงสร้างข้อมูล (Data Structure หรือ Logical data structure) เป็นในรูปแบบของความสัมพันธ์โดยแสดงออกมาในรูปแบบของตารางซึ่งข้อมูลในตารางจะต้องมีคุณสมบัติดังนี้
   - ข้อมูลในแต่ละแถวไม่ซ้ำกัน
   - ไม่จำเป็นต้องมีการเรียงลำดับของแถวต่างๆ
   - ข้อมูลในแต่ละสดมภ์จะไม่สามารถแบ่งแยกอีกได้
2. กฎบังคับความถูกต้องของข้อมูล (Data Integrity) เป็นกฎบังคับว่าข้อมูลที่ทำการใส่ลงฐานข้อมูลนั้นต้องมีความถูกต้องตามกฎข้อบังคับ (Constrain) ที่ตั้งไว้
3. ภาษาที่ใช้ในการจัดการข้อมูล (Data Manipulation)

5.1.2 ความสามารถของระบบจัดการฐานข้อมูลแบบออบเจ็กต์

มีการเก็บข้อมูล (Data หรือ Property) และการใช้ข้อมูล (Procedure หรือ Method) เข้าด้วยกันอยู่ในลักษณะของออบเจ็กต์ ซึ่งมีคุณสมบัติทางออบเจ็กต์ดังนี้

1. ความสามารถในการซ่อนรายละเอียด (Encapsulation) เป็นการซ่อนการทำงานภายในของออบเจ็กต์ไว้ ซึ่งออบเจ็กต์อื่นจะรู้แค่อินเตอร์เฟส (Interface) เท่านั้น
2. คุณสมบัติการโพลิมอร์ฟีซึม (Polymorphism)
3. ความสัมพันธ์แบบออบเจ็กต์ คือ
   - ความสัมพันธ์แบบแอคกรีเกชั่น (Aggregation)
   - ความสัมพันธ์แบบสืบทอด (Generalization)

ในระบบจัดการฐานข้อมูลเชิงวัตถุสัมพันธ์ที่ทำการศึกษาและนำมาใช้งาน คือระบบจัดการฐานข้อมูลออราเคิล ซึ่งมีความสามารถดังนี้

1. สามารถใช้ชนิดข้อมูลมาตรฐาน (Built-In Datatypes)
2. สามารถใช้ชนิดข้อมูลแบบกำหนดเอง (User-Defined Datatypes) ซึ่งประกอบด้วย
   2.1 ชนิดข้อมูลแบบออบเจ็กต์ (Object Types)
   2.2 ชนิดข้อมูลแบบกลุ่ม (Collection Types)

และทั้งสองหัวข้อนี้จะถูกนำเสนอถึงรายละเอียดทั้งหมดภายในบทนี้

5.2 ชนิดข้อมูลมาตรฐาน

ใช้ในการระบุชนิดของข้อมูลที่ต้องการเก็บลงฐานข้อมูล ซึ่งมีชื่อและลักษณะที่แตกต่างกันในระบบจัดการฐานข้อมูลที่มีอยู่มากมายในปัจจุบัน โดยชนิดข้อมูลมาตรฐานที่จะนำเสนอนี้จะเป็นชนิดข้อมูลมาตรฐานของระบบจัดการฐานข้อมูลออราเคิล

5.2.1 ชนิดข้อมูลแบบตัวอักษร (Character Datatypes)

ใช้สำหรับการเก็บข้อมูลที่เป็นตัวอักษร ซึ่งอยู่ในรูปแบบของการเข้ารหัสตัวอักษร (Character set) ในลักษณะต่างๆ เช่น ASCII, EBCDIC, etc. ซึ่งมีการแบ่งประเภทของชนิดข้อมูลแบบตัวอักษรนี้ย่อยลงไปอีกได้ดังนี้

5.2.1.1 CHAR Datatype

ใช้เก็บข้อความที่มีกำหนดความยาวของข้อความไว้แน่นอน ซึ่งมีความยาวข้อมลอยู่ระหว่าง 1 ถึง 200 ไบต์ ซึ่งเมื่อทำการเพิ่มหรือเปลี่ยนแปลงข้อมูลที่มีความยาวน้อยกว่าที่กำหนดข้อมูลชุดนั้นจะถูกแทนที่ด้วยช่องว่างซึ่งเป็นรหัสแอสกีตัวหนึ่ง

5.2.1.2 VARCHAR2 Datatype

ใช้เก็บข้อความที่มีจำนวนของตัวอักษรในข้อความนั้นเปลี่ยนแปลงได้ สามารถกำหนดจำนวนของตัวอักษรมากที่สุดที่อยู่ในหนึ่งข้อความได้ โดยมีค่าได้ระหว่าง 1 ถึง 4000

5.2.1.3 VARCHAR Datatype

เป็นชนิดข้อมูลที่ใช้ในออราเคิลรุ่นเก่าซึ่งยังคงไว้เพื่อสามารถใช้เก็บข้อมูลที่มีอยู่ในฐานข้อมูลออราเคิลรุ่นเก่นได้ โดยมีคุณสมบัติคล้ายกับ Varchar2

5.2.1.4 NCHAR และ NVARCHAR2 Datatype

ใช้เก็บข้อมูลที่มีการเข้ารหัสแบบ NLS โดย NCHAR จะมีคุณสมบัติเหมือน CHAR และ NVARCHAR2 จะมีคุณสมบัติเหมือน VARCHAR2

5.2.1.5 LONG Datatype

ใช้เก็บข้อความที่มีจำนวนของตัวอักษรในข้อความนั้นเปลี่ยนแปลงได้ และสามารถเก็บจำนวนตัวอักษรสูงสุดได้มากกว่า VARCHAR2

5.2.2 ชนิดข้อมูลแบบตัวเลข (Number Datatype)

ใช้เก็บข้อมูลที่เป็นตัวเลข ซึ่งประกอบด้วยเลขจำนวนเต็ม และเลขจำนวนจริง ซึ่งมีขนาดข้อมูลสูงสุด 21 ไบต์

5.2.3 ชนิดข้อมูลที่เป็นวันที่ (Date Datatype)

ใช้เก็บข้อมูลประเภทเวลา ประกอบด้วยปี เดือน วัน ชั่วโมง นาที วินาที ซึ่งมีขนาดคงที่ 7 ไบต์ สำหรับรูปแบบในการเก็บนั้นสามารถกำหนดได้ ซึ่งจะนำเสนอในส่วนของการเปลี่ยนชนิดข้อมูล

5.2.4 ชนิดข้อมูลที่มีขนาดใหญ่ (LOB Datatypes)

ใช้เก็บข้อมูลขนาดใหญ่ที่ไม่มีโครงสร้างข้อมูลที่แน่นอน เช่น ข้อมูลรูปภาพ ข้อมูลมัลติมีเดีย ซึ่งการเก็บข้อมูลด้วยชนิดของการเก็บข้อมูลนี้ทำให้สามารถเข้าถึงข้อมูลแบบสุ่มได้ (Random access) ในขณะที่การเก็บข้อมูลชนิด LONG จะเข้าถึงข้อมูลแบบเป็นลำดับเท่านั้น (Sequential access) ชนิดข้อมูลประเภทนี้ประกอบด้วย

5.2.4.1 BLOB Datatype

ใช้เก็บข้อมูลไบนารีขนาดใหญ่ที่ไม่มีโครงสร้าง (Unstructured Binary Data)

5.2.4.2 CLOB และ NCLOB Datatype

ใช้เก็บข้อมูลข้อความที่มีจำนวนตัวอักษรมากๆ

5.2.4.3 BFILE Datatype

ใช้เก็บข้อมูลไบนารีขนาดใหญ่โดยทำการเก็บข้อมูลนั้นไว้นอกฐานข้อมูล และเก็บเพียงตัวชี้ข้อมูลนั้นไว้ (Pointer) ซึ่งจะเป็นการเก็บข้อมูลเพื่ออ่านอย่างเดียว

5.2.5 RAW และ LONG RAW Datatypes

ใช้เก็บข้อมูลที่มีขนาดใหญ่โดย RAW มีคุณสมบัติเหมือน VARCHAR2 ส่วน LONG RAW นั้นเหมือน LONG สามารถใช้เก็บข้อมูลที่เป็นมัลติมีเดียได้แต่ประสิทธิภาพจะไม่ดีเท่าใช้ LOB และไม่สามารถทำการเอคซ์พอร์ท (Export)

5.2.6 ROWID Datatype

ใช้เก็บข้อมูลที่เป็นตำแหน่งจริงของแถวในฐานข้อมูล

5.3 ชนิดข้อมูลแบบกำหนดเอง

เกิดจากการสร้างชนิดข้อมูลขึ้นมาจากชนิดข้อมูลมาตรฐาน และชนิดข้อมูลกำหนดเองอื่นๆ ซึ่งได้ถูกแบ่งออกมาสองลักษณะดังนี้

5.3.1 ชนิดข้อมูลแบบออบเจ็กต์

เป็นชนิดของข้อมูลที่ผู้ใช้สามารถทำการสร้างขึ้นมาเอง โดยทำการกำหนดลักษณะของข้อมูลให้มีความใกล้เคียงกับความเป็นจริงมากที่สุด ซึ่งเป็นการใช้ลักษณะเด่นของข้อมูลนั้นเป็นตัวระบุ ซึ่งชนิดข้อมูลแบบออบเจ็กต์นี้ประกอบด้วยส่วนประกอบ 3 ส่วนดังนี้

1. ชื่อ (Name) ใช้ในการระบุถึงชนิดข้อมูลแบบออบเจ็กต์ ซึ่งต้องไม่ซ้ำกันในโครงสร้างที่เก็บข้อมูลแบบออบเจ็กต์เดียวกัน (Schema)
2. ลักษณะต่างๆ (Attributes) คือข้อมูลที่ออบเจ็กต์ควรมีเพื่อทำให้ออบเจ็กต์นั้นมีความใกล้เคียงกับความเป็นจริงให้มากที่สุด ซึ่งจะเป็นชนิดข้อมูลแบบมาตรฐาน หรือชนิดข้อมูลแบบกำหนดเองก็ได้
3. การกระทำ (Method) เป็นฟังก์ชันหรือโพรซีเดอร์ ที่สร้างจากภาษา PL/SQL เพื่อสร้างการกระทำให้กับออบเจ็กต์

ตัวอย่าง การสร้างข้อมูลชนิดออบเจ็กต์

```
CREATE OR REPLACE TYPE person_object AS OBJECT (
   ssn              NUMBER,
   last_name  VARCHAR2(50),
   first_name       VARCHAR2(50) ;
```

5.3.2 ชนิดข้อมูลแบบกลุ่ม

เป็นลักษณะของข้อมูลที่มีชนิดของข้อมูลเหมือนกันและมีจำนวนที่ไม่สามารถระบุได้ว่ามีเท่าไร ซึ่งจะแบ่งเป็น ชนิดข้อมูลแบบอาร์เรย์ (Array Types) หรือชนิดข้อมูลแบบ วีอาร์เรย์ (VArray) และ ชนิดข้อมูลแบบ ตาราง (Table Types) หรือชนิดข้อมูลแบบตารางซ้อนตาราง (Nested Table)

5.3.2.1 ชนิดข้อมูลแบบวีอาร์เรย์

เป็นการเก็บข้อมูลที่มีชนิดของข้อมูลเหมือนกัน และข้อมูลที่อยู่ในแต่ละช่องของอาร์เรย์จะถูกระบุด้วยดัชนี (Index) ซึ่งเป็นลักษณะของเซตของข้อมูลแบบมีลำดับ

ตัวอย่าง การสร้างชนิดข้อมูลวีอาร์เรย์

```
CREATE TYPE Project AS OBJECT(
   project_no NUMBER(2),
   title      VARCHAR2(35),
   cost       NUMBER(7,2))

CREATE TYPE ProjectList AS VARRAY(50) OF Project
```

5.3.2.2 ชนิดข้อมูลแบบตารางซ้อนตาราง

เป็นการสร้างตารางย่อยขึ้นภายในตาราง โดยสามารถสร้างตารางย่อยนี้ได้จากชนิดข้อมูลแบบกำหนดเอง ซึ่งมีลักษณะเป็นกลุ่มของข้อมูลที่ไม่เป็นลำดับ คือไม่มีการใช้ดัชนีเป็นตัวเข้าถึงข้อมูล

ตัวอย่าง การสร้างชนิดข้อมูลตารางซ้อนตาราง

```
CREATE OR REPLACE TYPE class_o AS OBJECT (          -- Create Object
Type
id            NUMBER,
semester            VARCHAR2(10),
name          VARCHAR2(25),
concentration VARCHAR2(25),
professor     VARCHAR2(50) );

CREATE OR REPLACE TYPE class_t AS TABLE OF class_o;  -- Create Table
Type

CREATE TABLE student (
  id          NUMBER,
  ssn         NUMBER,
  lname             VARCHAR2(50),
  fname             VARCHAR2(50),
  classes           class_t )
NESTED TABLE classes STORE AS student_classes;
```

5.4 การเปลี่ยนชนิดของข้อมูล

มีฟังก์ชันที่ใช้ในการเปลี่ยนชนิดข้อมูลดังนี้

- TO_NUMBER()
- TO_CHAR()
- TO_DATE()
- CHARTOROWID()
- ROWIDTOCHAR()
- HEXTORAW()
- RAWTOHEX()

ซึ่งฟังก์ชันกลุ่มนี้เป็นฟังก์ชันของระบบจัดการฐานข้อมูลของออราเคิลไม่ใช้เป็นฟังก์ชันของเอดดิเตอร์ (Editor) ดังนั้นจึงสามารถทำการเรียกใช้ด้วยโปรแกรมที่ใช้ในการพัฒนาอื่นได้ ซึ่งรายละเอียดของแต่ละฟังก์ชันสามารถหาได้จากเอกสารที่มากับระบบจัดการข้อมูลออราเคิลได้ (Help in Oracle)

5.5 การติดต่อฐานข้อมูลออราเคิลด้วยโปรแกรมวิชวลเบสิก

รูปแบบของการติดต่อกับระบบจัดการฐานข้อมูลต่างๆ ของโปรแกรมวิชวลเบสิกนั้นส่วนใหญ่จะใช้ ODBC (Open Database Connectivity) เป็นตัวกลางในการติดต่อเพราะเป็นมาตรฐานกลางในการติดต่อกับระบบจัดการฐานข้อมูลทั่วไป และมีออบเจ็กต์ที่ใช้ในการติดต่อกับฐานข้อมูลดังนี้

5.5.1 DAO (Data Access Object)

เป็นเทคโนโลยีที่แอพพลิเคชันฐานข้อมูลทำการติดต่อกับฐานข้อมูลโดยใช้กลุ่มของออบเจ็กต์ที่เตรียมไว้ให้ในการติดต่อ โดยขอยกตัวอย่างออบเจ็กต์ที่สำคัญดังนี้

5.5.1.1 ออบเจ็กต์ที่ใช้งานใน DAO

- DBEngine เป็นตัวเก็บออบเจ็กต์ DAO ต่างๆไว้ภายใน
- Workspace Object เป็นออบเจ็กต์ที่จัดการเซสชัน (Session) ของผู้ใช้ จัดการกับทรานเซคชัน (Transaction) และเรื่องของความปลอดภัย โดยผู้ใช้งานแต่ละรายจะถูกกำหนดเซสชันโดยจะไม่รบกวนกับผู้ใช้งานรายอื่นๆ
- Database Object เป็นออบเจ็กต์ที่ใช้พรอพเพอร์ตี้ (Property) และเมธอด (Method) ในการจัดการฐานข้อมูล
- Recordset Object เป็นออบเจ็กต์ที่ใช้พรอพเพอร์ตี้ (Property) และเมธอด (Method) ในการจัดการกลุ่มของเรคคอร์ด ซึ่งกลุ่มของเรคคอร์ดอาจได้มาจากตารางเดียว หรือหลายตารางได้
- QueryDef Object เป็นออบเจ็กต์ที่เก็บคำสั่งในภาษา SQL ซึ่งเมื่อเรียกใช้งานออบเจ็กต์นี้ก็จะทำงานตามคำส่งในภาษา SQL ที่กำหนดไว้
- TableDef Object เป็นออบเจ็กต์ที่เก็บโครงสร้างของฐานข้อมูล (Base Table) และ ตารางของฐานข้อมูลอื่นที่เชื่อมโยงถึงกันอยู่ (Linked Table)
- Field Object เป็นออบเจ็กต์ที่ทำหน้าที่ในการจัดการรายละเอียดเกี่ยวกับฟิลด์ข้อมูลของออบเจ็กต์เร็คคอร์ดเซตและคิวรี่เดฟ

5.5.1.2 ข้อดีของ DAO

1. ง่ายต่อการตรวจสอบและจัดการข้อมูลก่อนทำการบันทึกข้อมูลนั้นลงฐานข้อมูล
2. ลดความยุ่งยากในการใช้ฐานข้อมูล เมื่อใช้งานร่วมกับผู้ใช้รายอื่น
3. สนับสนุนการประมวลผลแบบทรานเซคชั่น ซึ่งช่วยในแอปพลิเคชั่นมีความน่าเชื่อถือและมีเสถียรภาพ
4. จัดการและควบคุมเกี่ยวกับข้อผิดพลาดได้ง่าย

5.5.1.3 การเขียนโปรแกรมโดยใช้ DAO

```
Public OrderWs As Workspace
Public OrderDB As Database
Public rstOrder As Recordset

Set OrderWs = CreateWorkspace("MyWorkspace",  "threetier", "public", _
dbUseODBC)
Set OrderDB = OrderWs.OpenDatabase("OracleDSN", dbDriverComplete, True, _
"ODBC;")
Set rstOrder = OrderDB.OpenRecordset(SQL_Statement, dbOpenSnapshot)
```

ต้องทำการอ้างถึง (Reference) Microsoft DAO 3.51 Object library ด้วย ซึ่งมีหลักการดังนี้คือ DAO จะทำงานอยู่บน Workspace ซึ่งมีการเปิดคอนเนคชันกับฐานข้อมูล ซึ่งภายในหนึ่ง Workspace จะมีได้หลายคอนเนคชัน และภายในแต่ละคอนเนคชันจะมีได้หลายเร็คคอร์ดเซต ซึ่งแต่ละเร็คคอร์ดเซตจะระบุถึงฐานข้อมูลในระดับตารางที่ต้องการทำงานด้วย โดย DAO นี้ไม่มีคอนโทรลให้ใช้ดังเช่น Data Control หรือ ADO Data Control ดังนั้นการใช้งานต้องใช้การเขียนโปรแกรมอย่างเดียว

5.5.2 ADO (ActiveX Data Object)

เป็นเทคโนโลยีการติดต่อกับฐานข้อมูลรูปแบบใหม่โดยตัดออบเจ็กต์บางตัวของ DAO ออกเพื่อทำให้สามารถใช้งานได้ง่ายขึ้นและเนื่องจากความเป็นเทคโนโลยี ActiveX ทำให้ ADO ถูกนำมาใช้มากในการเชื่อมต่อฐานข้อมูลผ่านอินเตอร์เน็ต ซึ่งความสามารถอย่างหนึ่งของ ADO คือเป็น OLE DB ด้วยเพื่อสนับสนุนการจัดการกับข้อมูลที่เป็นมัลติมีเดียด้วย

5.5.2.1 ออบเจ็กต์ที่ใช้งานใน ADO

- Connection เป็นออบเจ็กต์ที่ทำหน้าที่ในการเชื่อมต่อกับแหล่งข้อมูล (Data Source) ซึ่งทำการติดต่อกับฐานข้อมูลผ่าน ODBC
- Command เป็นออบเจ็กต์ที่ใช้ในการสั่งงานในรูปแบบของภาษา SQL หรือ Store Procedure ซึ่งมีความจำเป็นว่าแหล่งข้อมูลนั้นต้องยอมรับด้วย
- Parameter เป็นออบเจ็กต์ที่ใช้ในการส่งผ่านพารามิเตอร์ที่จำเป็น เช่น การส่งผ่านพารามิเตอร์ของ Store Procedure
- Recordset เป็นออบเจ็กต์ที่จัดการกับผลลัพธ์ที่ได้ในลักษณะของกลุ่มของข้อมูล
- Field เป็นออบเจ็กต์ที่จัดการกับสคมภ์ที่อยู่ภายในเร็คคอร์ดเซ็ต
- Error เป็นออบเจ็กต์ที่จัดการกับความผิดพลาดที่เกิดขึ้น

5.5.2.2 ข้อดีของ ADO

1 ความซับซ้อนในการติดต่อกับฐานข้อมูลลดลง

2 ใช้เทคโนโลยี ActiveX ทำให้สามารถพัฒนาแอปพลิเคชั่นบนอินเตอร์เน็ตได้ดีขึ้น

3 สนับสนุนการจัดการกับข้อมูลที่เป็นมัลติมีเดียได้

5.5.2.3 การเขียนโปรแกรมโดยใช้ ADO

1. สามารถใช้คอนโทรในการสร้างการติดต่อกับฐานข้อมูลได้โดยทำการเพิ่มคอมโพเนนต์
2. ทำการเรียกใช้ออบเจ็กต์ที่ได้กล่าวมาในการทำการติดต่อกับฐานข้อมูล ดังนี้

```
Dim cn As ADODB.Connection
Dim rs As ADODB.Recordset

Set cn = New ADODB.Connection
cn.ConnectionString = "PROVIDER=MSDASQL;DSN=OracleDSN;" & _
UID=boy;PWD=boy;"
cn.CursorLocation = adUseClient
cn.Open

Set rs = New ADODB.Recordset
rs.CursorType = adOpenKeyset
rs.LockType = adLockOptimistic
rs.Open Str_sql, cn
```

โดยให้ทำการอ้างถึง Microsoft ActiveX Data Object 2.1 ด้วย มีหลักการทำงานดังนี้คือออบเจ็กต์คอนเนคชันจะทำการติดต่อกับฐานข้อมูลซึ่งภายในออบเจ็กต์คอนเนคชันนี้จะมีการเปิดเรคคอร์ดเซตอยู่ภายใน ซึ่งเราสามารถทำการเปิดคอนเนคชันผ่านทางเรคคอร์ดเซตโดยตรงได้โดยเปลี่ยนจากคอนเนคชันออบเจ็กต์เป็นคอนเนคชันสตริงได้ ซึ่งวิธีเลือกใช้ว่าเราควรจะใช้คอนเนคชันออบเจ็กต์หรือไม่นั้นเราพิจารณาที่การเปิดเรคคอร์ดเซตภายในคอนเนคชันหนึ่งถ้าเรามีการเปิดหลายครั้งเราควรทำเป็นคอนเนคชันออบเจ็กต์ เช่นต้องการเรียกดูข้อมูลจากตารางหนึ่งเพื่อทำการเปลี่ยนแปลงแก้ไขข้อมูลอีกตารางหนึ่งอย่างนี้ควรสร้างคอนเนคชันออบเจ็กต์

บทที่ 6
การวิเคราะห์และการออกแบบระบบ

6.1 คำอธิบายปัญหา (Problem Statement)

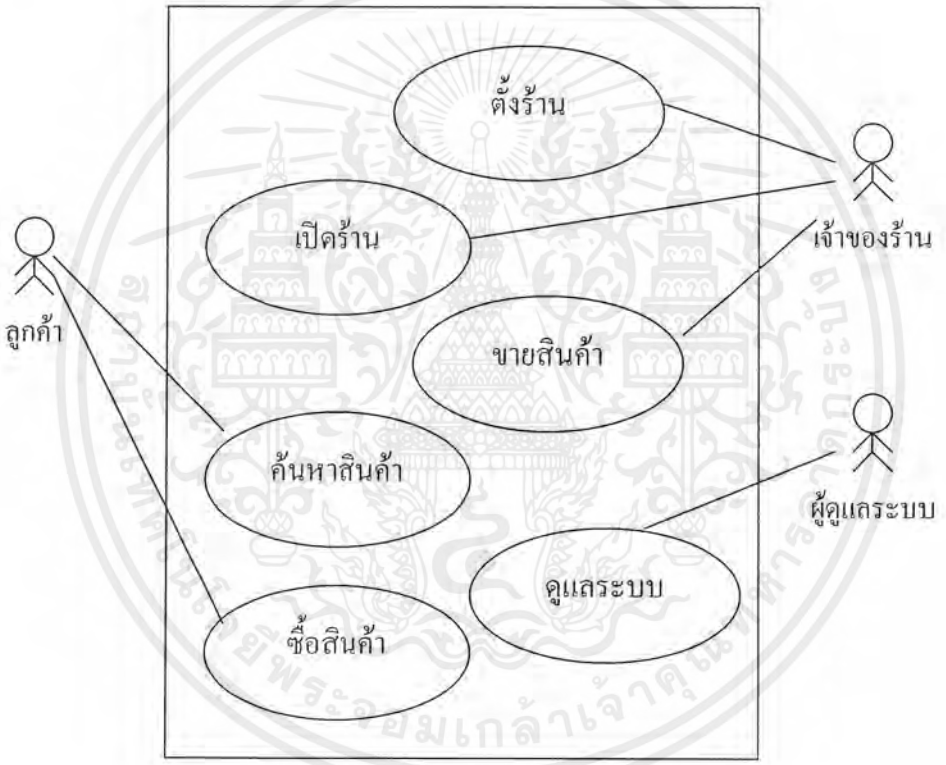
เมื่อเราพิจารณาระบบพาณิชย์อิเล็กทรอนิกส์ (Electronic Commerce) เราจะได้คำอธิบายของระบบออกมาดังนี้

- ลูกค้าสามารถมาเปิดร้าน (เจ้าของร้าน) โดยมีรายละเอียดตามที่ลูกค้าต้องการได้ เช่น รูปแบบเว็บเพจของร้านที่เขาต้องการขายของ
- เจ้าของร้านสามารถจัดการรายละเอียดเกี่ยวกับสินค้าในร้านได้ เช่น การเพิ่มสินค้าเข้าไปในร้าน การเปลี่ยนแปลงข้อมูลของสินค้า หรือข้อมูลต่างๆในร้านของเขา และการลบข้อมูลสินค้าในร้านของเขาได้
- ข้อมูลของสินค้าจะอยู่ในรูปมัลติมีเดียสตรีม (Multimedia Stream) คือ ภาพ เสียง และวิดีโอ
- ร้านค้าที่เปิดขึ้นสามารถขายสินค้าใดๆ ก็ได้โดยมีการเก็บข้อมูลทุกอย่างที่จำเป็นตามแต่ชนิดของสินค้า
- ลูกค้าที่เข้ามาซื้อสินค้าสามารถเรียกดูข้อมูลต่างๆ เกี่ยวกับสินค้าที่ต้องการได้ และถ้ามีความสนใจในสินค้าสามารถทำการเลือกสินค้านั้นเก็บลงในกระเป๋าสินค้าได้ (Shopping Bag) ซึ่งข้อมูลนี้สามารถทำการเก็บไว้เพื่อใช้ในการซื้อครั้งต่อไปได้
- การเข้าใช้ระบบจะมีการควบคุมในระดับระบบสารสนเทศ (Information System Control) โดยมีการกรอกชื่อผู้ใช้และรหัสผ่าน ทั้งในส่วนของเจ้าของร้านและลูกค้าเพื่อเป็นการปกป้องทรัพยากรระบบ
- ในการสื่อสารข้อมูลมีการเข้ารหัสในรูปแบบของ SSL (Secure Socket Layer) 40 บิต
- ระบบการเก็บข้อมูลเป็นระบบจัดการฐานข้อมูลเชิงวัตถุสัมพันธ์
- มีระบบจัดการทรานแซคชั่นทั้งในระดับโลคอล (Local) และโกลบอล์ (Global) หรือทูเฟสคอมมิต (Two-Phase Commit)
- ลูกค้าสามารถหาสินค้าตามคุณสมบัติของสินค้าที่ต้องการได้
- ระบบสามารถยืนยันการสั่งซื้อผ่านระบบเพจเจอร์และระบบส่งข้อความสั้น (Shot Message) ของผู้ให้บริการต่างๆ ได้ เช่น โฟนลิงค์, GSM เป็นต้น และรวมถึงระบบอีเมล์และระบบไอซีคิวได้ด้วย
- ผู้ซื้อสินค้าสามารถติดตามผลการส่งสินค้าได้ว่าขณะนี้อยู่ในขั้นตอนใดแล้ว

6.2 สร้าง Use Case

เรานำคำอธิบายปัญหามาแยกออกเป็นฟังก์ชันการทำงานโดยรวมของระบบ ในรูปแบบของ Use Case ซึ่งจะมี Use Case ดังนี้

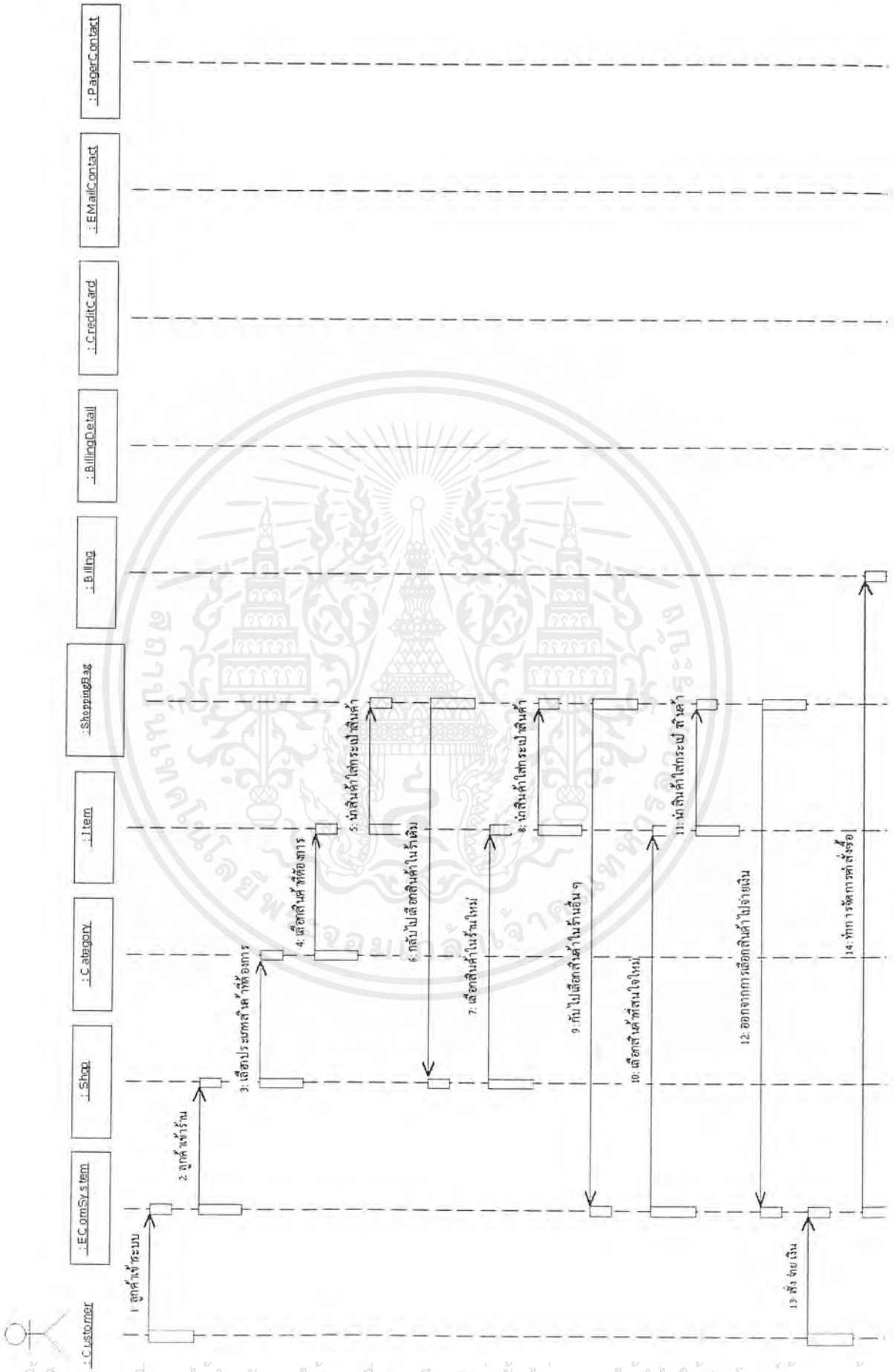
- ตั้งร้าน
- เปิดร้าน
- ขายสินค้า
- ค้นหาสินค้า
- ซื้อสินค้า
- การดูแลระบบ



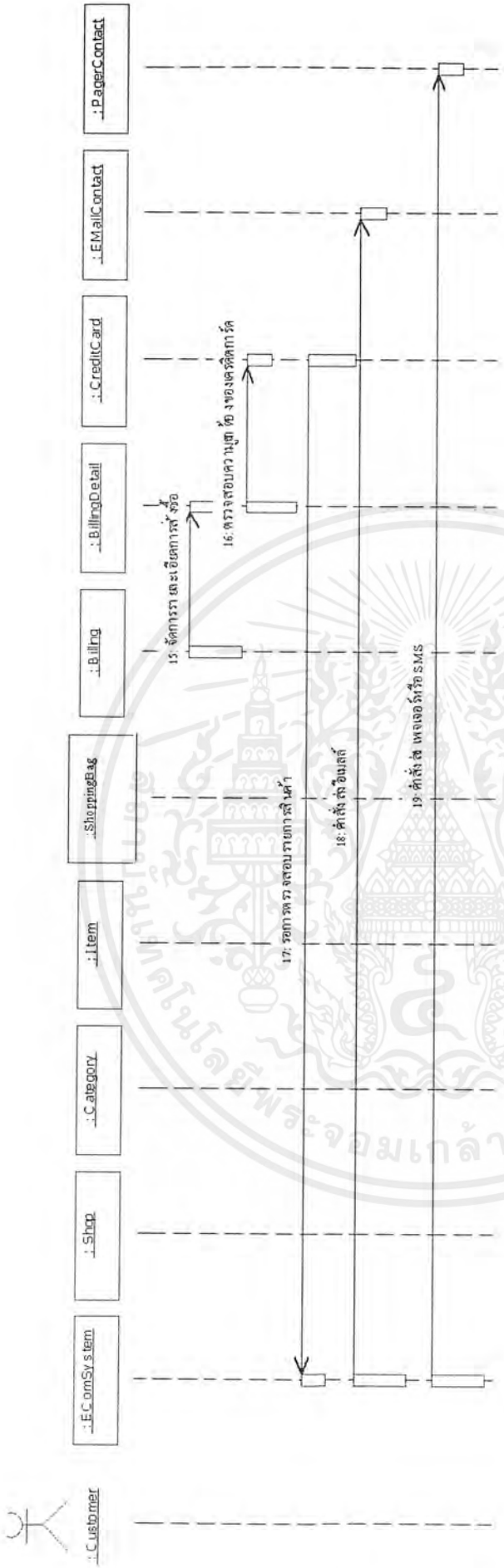
รูป 6-1 แสดง Use Case ของระบบ

6.3 สร้างซินาริโอ

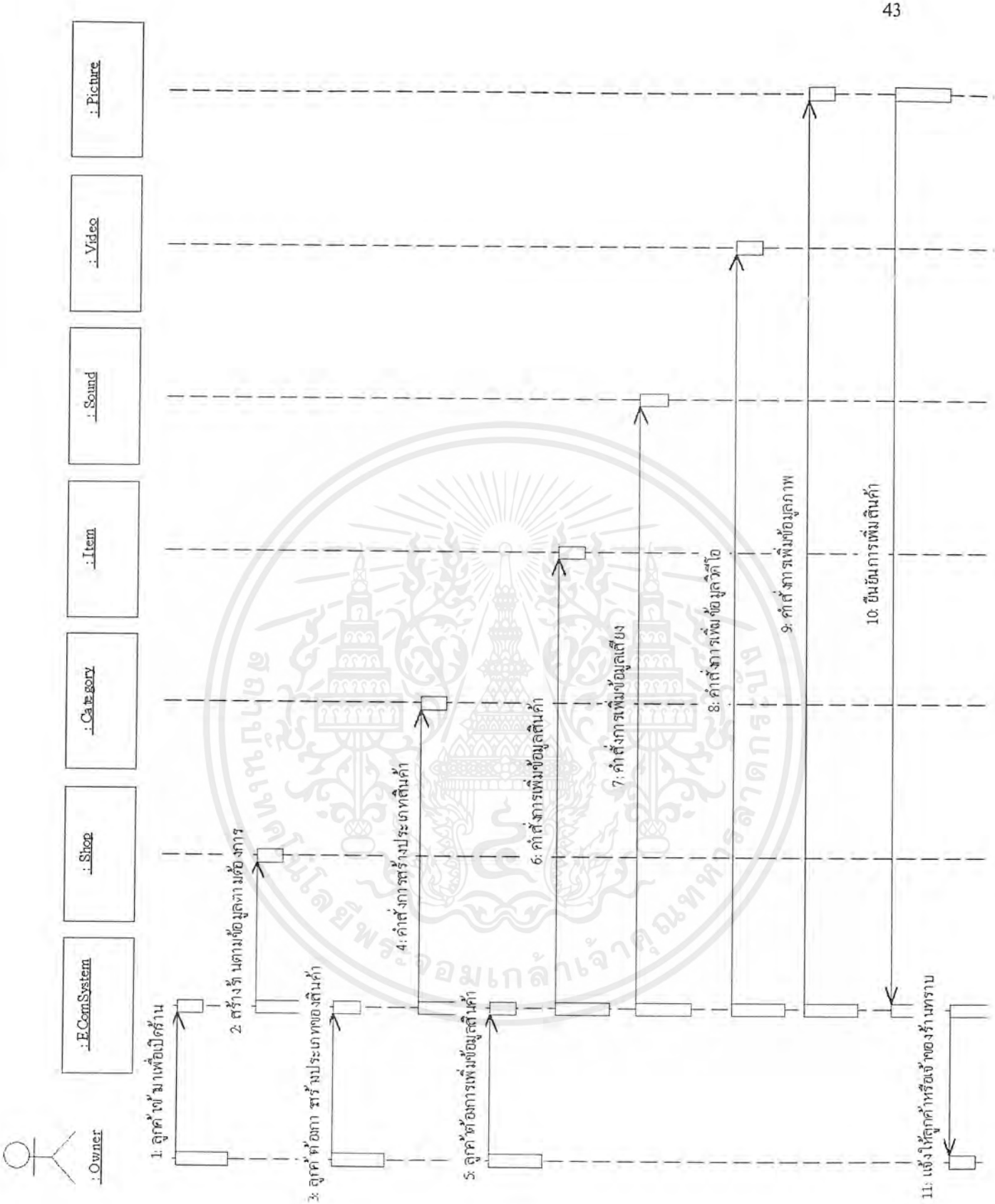
ในที่นี้จะนำเพียง 2 Use Case คือ การซื้อสินค้า และการตั้งร้าน มาสร้างซินาริโอเพื่อใช้ในการตรวจสอบความถูกต้องของ Use Case และช่วยในการกำหนดออบเจ็กต์เบื้องต้น ซึ่งแสดงได้ดังนี้



รูปที่ 6-2 ซินาริโอการซื้อของ (1)



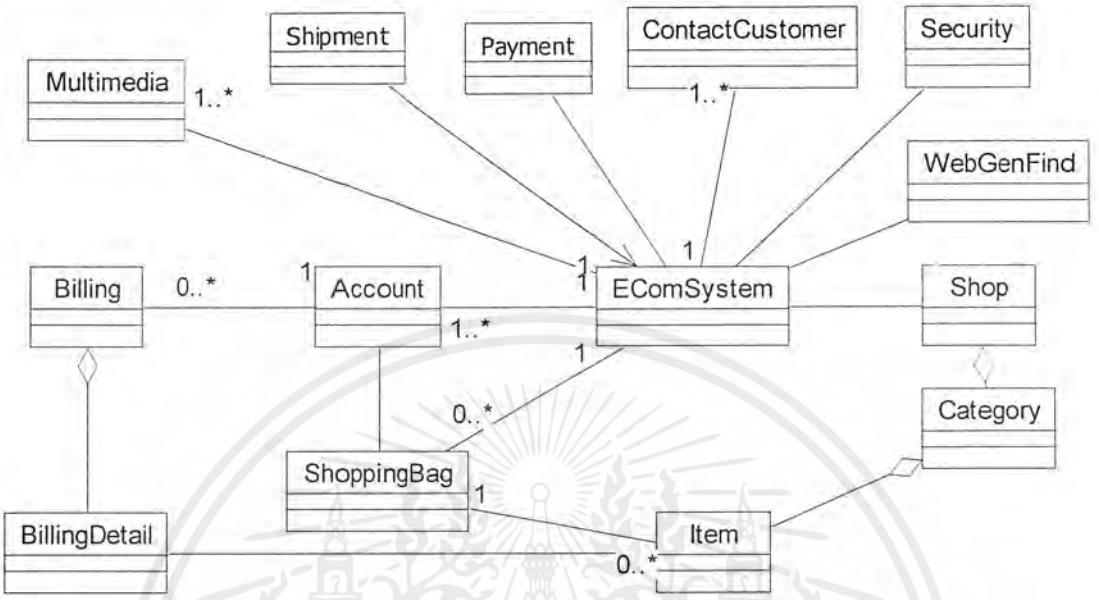
รูปที่ 6-3 ซินาริโอการซื้อของ (2)



รูปที่ 6-4 ซินาริโอการตั้งร้าน

6.4 กำหนดโครงสร้างออบเจ็กต์

จากการวิเคราะห์ระบบทำให้ได้ออบเจ็กต์ไดอะแกรมในระดับการวิเคราะห์เบื้องต้น (First Cut Object Diagram) ออกมาดังนี้

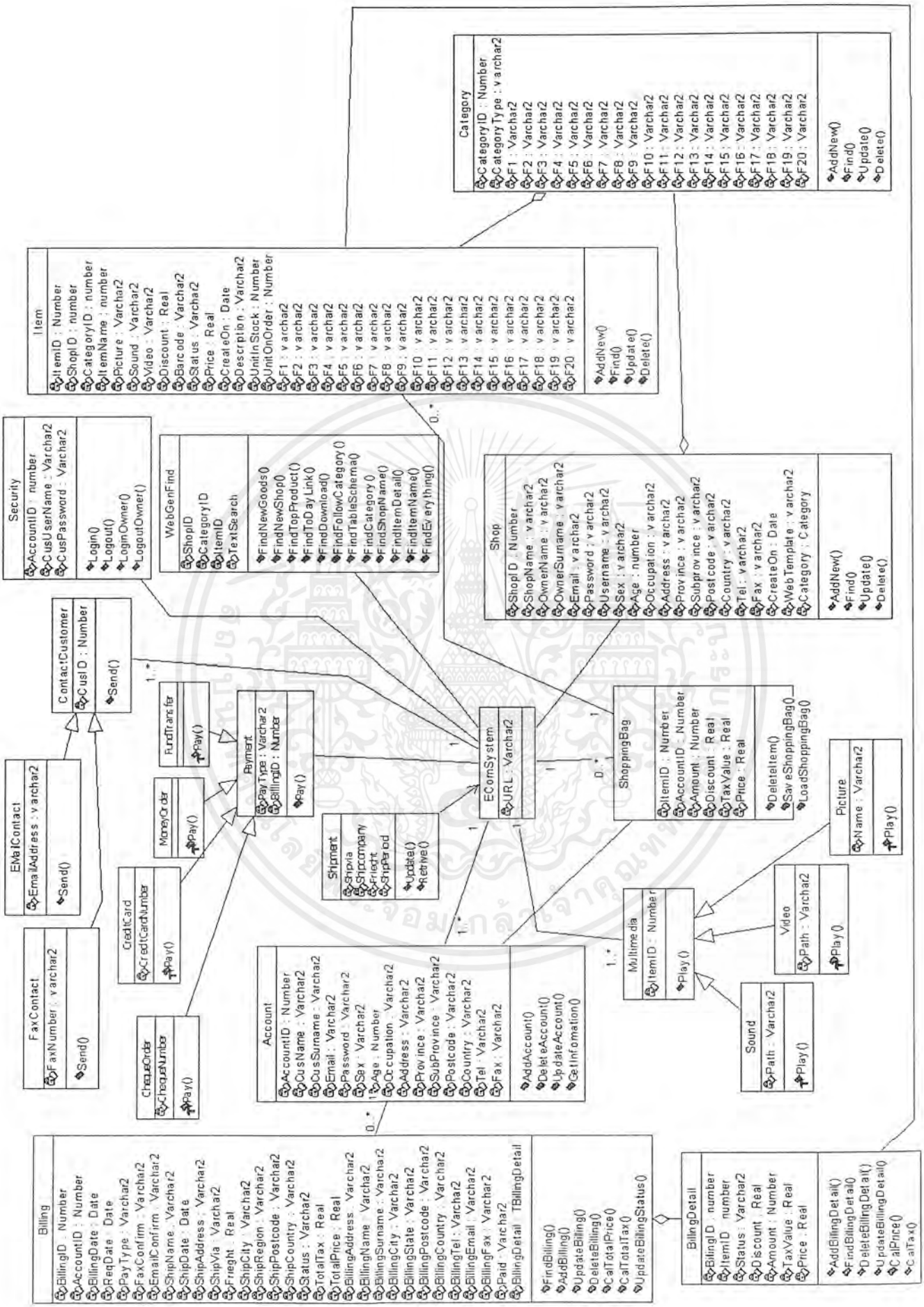


รูปที่ 6-5 แสดงออบเจ็กต์ไดอะแกรมในระดับการวิเคราะห์เบื้องต้น

โดยรายละเอียดของออบเจ็กต์แต่ละตัวจะนำมาอธิบายอีกครั้งในส่วนของการสร้างโครงสร้างคลาสซึ่งอยู่ในส่วนต่อไป

6.5 กำหนดโครงสร้างคลาส

จากออบเจ็กต์ไดอะแกรมเราสามารถสร้างเอกลักษณ์ของระบบเพื่อแสดงโครงสร้างและความสัมพันธ์ต่างๆ ของออบเจ็กต์ในระบบ รวมถึงการสร้างสเตทไดอะแกรมเพื่อดูพฤติกรรมต่างๆ ของคลาสในระบบเพื่อใช้ระบุโอเปอเรชันและเมธอดของคลาสจากที่กล่าวมาเราสามารถสร้างคลาสไดอะแกรมได้ดังรูปที่ 6-6



รูปที่ 6-6 คลาสไดอะแกรมของระบบ

จากคลาสไดอะแกรมที่ได้มานั้นจะแบ่งเป็น 2 ส่วนคือ ส่วนที่ทำงานด้านฟรอนท์เอนด์ (Front-End) และส่วนที่ทำงานด้านแบคก์เอนด์ (Back-End) โดยการทำงานด้านฟรอนท์เอนด์นั้นจะมีหน้าที่คือรับการติดต่อกับผู้ใช้ระบบซึ่งจะเป็นการติดต่อระหว่างบราวส์เซอร์ (Browser) กับเว็บเซิร์ฟเวอร์ (Web Server) โดยเว็บเซิร์ฟเวอร์นี้จะทำการเรียกใช้การทำงานขั้นพื้นฐานจากส่วนแบคก์เอนด์ ซึ่งหน้าที่ในส่วนแบคก์เอนด์นั้นจะทำหน้าที่ในการติดต่อกับระบบฐานข้อมูล (Database Server) เพื่อทำการเรียกใช้ข้อมูลจากระบบฐานข้อมูลเพื่อจัดเตรียมข้อมูลนั้นให้อยู่ในรูปแบบที่สามารถถูกส่งไปให้กับส่วนฟรอนท์เอนด์นำไปใช้ได้

ในหัวข้อต่อไปจะแสดงการทำงานของแต่ละคลาสที่อยู่ในส่วนของแบคก์เอนด์ทั้งหมด รวมถึงการจัดการทรานแซกชัน และการกำหนดกฎการทำงานของระบบ (Business Rules) ที่อยู่ที่แอพลิเคชันเซิร์ฟเวอร์ (Application Server)

โดยภายในปริญญานิพนธ์เล่มนี้จะมีขอบเขตการทำงานครอบคลุมการทำงานทั้งหมดในส่วนแบคก์เอนด์ ส่วนการทำงานในส่วนของฟรอนท์เอนด์จะถูกแยกอยู่กับปริญญานิพนธ์อีกเล่มหนึ่ง

6.6 การทำงานของคลาสในส่วนแบคก์เอนด์

ในหัวข้อนี้จะทำการแบ่งการอธิบายการทำงานของคลาสนี้ในสองรูปแบบคือ ส่วนแรกจะทำการอธิบายถึงการทำงานของทุกคลาสว่าแต่ละคลาสที่สร้างขึ้นมานี้สร้างมาทำไมและมีหน้าที่ความรับผิดชอบอะไรบ้างซึ่งเหมาะสำหรับผู้ที่มีความสนใจในการศึกษาถึงโครงสร้างแท้จริงของส่วน แบคก์เอนด์ และอีกส่วนหนึ่งจะทำการอธิบายในลักษณะการทำงานที่ด้านแบคก์เอนด์ทั้งหมดนั้นสามารถทำอะไรได้บ้างซึ่งจะเป็นรูปธรรมมากกว่าซึ่งเหมาะสำหรับผู้ที่ต้องการเข้าใจถึงการทำงานโดยภาพรวมของระบบ

โดยคลาสในส่วนของแบคก์เอนด์จะมีดังนี้

- Shop
- Category
- Item
- ShoppingBag
- Billing
- BillingDetail
- Account
- Shipment

6.6.1 การทำงานของแต่ละคลาสในส่วนแบคก์เอนด์

6.6.1.1 คลาส SHOP

เป็นคลาสร้านค้าที่ภายในร้านจะมีประเภทของสินค้า (Category) และสินค้า (Item) ภายในร้าน ภายในคลาสนี้มีการรวมเอาข้อมูลของเจ้าของร้านมารวมไว้กับคลาสนี้ด้วย เหตุผลคือ เจ้าของร้านไม่มีเมธอดที่เด่นชัดถึงขั้นที่ต้องแยกออกมาเป็นอีกคลาสหนึ่ง โดยเมธอดที่สำคัญของเจ้าของร้านคือระบุตัวเขา (Authorization) ซึ่งเราได้แบ่งการทำงานในส่วนนี้ออกเป็นคลาสหนึ่งที่จัดการเรื่องความปลอดภัย (Class : Security) อยู่แล้ว

การตั้งร้านนี้ต้องใช้ Stored Procedure เพื่อใช้ในการทำงานกับ Nested Table ด้วยซึ่งรายละเอียดจะอยู่ในภาคผนวก การใช้งาน SQL ในการติดต่อกับตารางลักษณะเนสเต็ด

ซึ่งภายในคลาส Shop นี้จะมีแอตทริบิวต์และเมธอดดังนี้

แอตทริบิวต์

ShopID : Integer	หมายเลขของร้านค้า
ShopName : String	ชื่อของร้านค้า
OwnerName : String	ชื่อของเจ้าของร้าน
OwnerSurname : String	นามสกุลของเจ้าของร้าน
Email : String	อีเมล์ของเจ้าของร้าน
Username : String	ชื่อที่เจ้าของร้านใช้ในการระบุความเป็นเจ้าของร้าน
Password : String	รหัสผ่านในการระบุความเป็นเจ้าของร้าน
Sex : String	เพศของเจ้าของร้าน
Age : Integer	อายุของเจ้าของร้าน
Occupation : String	อาชีพของเจ้าของร้าน
Address : String	ที่อยู่ของเจ้าของร้าน
SubProvince : String	เขตที่เจ้าของร้านอยู่
Province : String	จังหวัดที่เจ้าของร้านอยู่
Postcode : String	รหัสไปรษณีย์ที่อยู่ของเจ้าของร้าน
Country : String	ประเทศที่เจ้าของร้านอยู่
Tel : String	เบอร์โทรศัพท์ของเจ้าของร้าน
Fax : String	เบอร์แฟกซ์ของเจ้าของร้าน
CreateOn : Date	วัน ที่ร้านได้ถูกตั้งขึ้น
WebTemplate : String	รูปแบบหน้าร้าน
Category : Class	คลาสแผนกหรือประเภทสินค้าภายในร้าน

เมธอด

AddNew() : ShopID

ใช้ในการตั้งร้านขึ้นมาใหม่โดยข้อมูลของร้านใหม่ถูกส่งผ่านทางแอตทริบิวต์ และเมื่อการตั้งร้านเสร็จสมบูรณ์จะส่งหมายเลขของร้านออกมา

Find(SQLString : String) : Recordset

ใช้ในการค้นหาข้อมูลของร้านโดยผ่านคำสั่ง SQL เพื่อให้ได้เซตของข้อมูลที่ได้จากฐานข้อมูลออกมา ซึ่งถ้าไม่ส่งคำสั่ง SQL เข้าไปเมธอดนี้จะทำการเรียกข้อมูลทั้งหมดของร้านค้าออกมา

Update()

ใช้ในการเปลี่ยนแปลงแก้ไขข้อมูลของร้านโดยต้องทำการระบุร้านที่ต้องการแก้ไขข้อมูลก่อน โดยการผ่านหมายเลขของร้านเข้าไปทางแอตทริบิวต์ ShopID และส่งข้อมูลที่ต้องการเปลี่ยนแปลงไปทางแอตทริบิวต์ ซึ่งถ้าแอตทริบิวต์ใดไม่มีข้อมูลอยู่แสดงว่าไม่ต้องการเปลี่ยนแปลงข้อมูล

การเปลี่ยนแปลงแก้ไขข้อมูลเฉพาะบางตัวของคลาสนี้เราทำได้โดย ทำการตรวจสอบว่าค่าที่ถูกส่งผ่านแอตทริบิวต์เข้ามานี้เป็นค่าเริ่มต้น (Initial Value)หรือไม่ เช่นถ้าเป็นข้อความค่าเริ่มต้นจะเป็น Empty String ("") ถ้าเป็นตัวเลขจะได้ค่าเป็น 0 ซึ่งเราตรวจสอบจากค่าพวกนี้ก่อนทำการเปลี่ยนแปลงแก้ไขในฐานข้อมูล ซึ่งมีปัญหาที่ว่าถ้าต้องการเปลี่ยนแปลงข้อมูลให้เป็นค่าเริ่มต้นนี้จะไม่สามารถทำได้ เช่นต้องการเปลี่ยนค่าที่เป็นตัวเลขให้มีค่าเป็น 0 เมธอด Update นี้จะไม่สามารถแยกได้ว่าเป็นค่าเริ่มต้นหรือเป็นค่าที่ต้องการให้เป็นจริงๆ ทำให้ไม่มีการเปลี่ยนแปลงเกิดขึ้น

Delete(Criteria : String)

ใช้ในการลบร้านออกจากฐานข้อมูลโดยทำการผ่านเงื่อนไขในการลบ (Criteria) ซึ่งเป็นเงื่อนไขหลัง Where ใน SQL นั่นเองเพื่อทำการระบุเซตของข้อมูลที่ต้องการลบ โดยการลบร้านนี้จะทำการลบแผนกและสินค้าที่อยู่ภายในร้านด้วย ซึ่งเป็นการกำหนดเงื่อนไขในโครงสร้างฐานข้อมูลซึ่งจะนำเสนอในหัวข้อ โครงสร้างข้อมูลและข้อกำหนดในฐานข้อมูล ซึ่งอยู่ภายในบทนี้

6.6.1.2 คลาส CATEGORY

เป็นคลาสที่สร้างขึ้นมาเพื่อรองรับกับการขายสินค้าที่มีความแตกต่างกัน โดยใน 1 ร้านจะมีได้หลายแผนก (Category) และใน 1 แผนกจะมีสินค้าที่มีคุณสมบัติเหมือนกัน

แอตทริบิวต์

CategoryID : Integer	หมายเลขของแผนกสินค้าในร้านโดยทุกร้านจะได้หมายเลขแผนกของสินค้าที่ไม่ซ้ำกัน
CategoryType : String	ชื่อแผนกของสินค้าซึ่งในร้านเดียวกันจะไม่มีชื่อที่ซ้ำกัน
F1 – F20 : String	เป็นแอตทริบิวต์อิสระ 20 ตัวมีไว้ให้เจ้าของร้านกำหนดคุณสมบัติของสินค้าที่มีนอกเหนือจากคุณสมบัติพื้นฐานที่มีอยู่ของสินค้า (ในหัวข้อต่อไปจะแสดงคุณสมบัติพื้นฐานของสินค้า)

เมธอด

AddNew(ShopID : Integer) : CategoryID

ใช้ในการตั้งแผนกขึ้นมาใหม่ภายในร้าน โดยผ่านข้อมูลที่จำเป็นในการตั้งแผนกใหม่ผ่านทางแอตทริบิวต์ สำหรับ ShopID ที่ต้องทำการส่งผ่านเป็นพารามิเตอร์ (Parameter) นั้นเพราะคลาสนี้เป็นแอตทริบิวต์ของคลาส Shop ดังนั้นจึงไม่จำเป็นต้องมีแอตทริบิวต์ ShopID แต่ว่าในการนำข้อมูลใส่ลงฐาน

ข้อมูลนั้นจำเป็นต้องใช้ ShopID ด้วยจึงจำเป็นต้องใช้วิธีนี้ เมื่อทำการตั้งแผนกในร้านสมบูรณ์จะส่งหมายเลขของแผนกนั้นออกมา

Find (ShopID : Integer, Criteria : String) : Recordset

ใช้ในการเรียกดูข้อมูลของแผนกต่างๆ ในร้าน โดยทำการผ่าน ShopID และเงื่อนไขในการเรียกดูข้อมูล ที่ต้องพารามิเตอร์สองตัวนี้เพราะโครงสร้างของข้อมูลระหว่าง Shop กับ Category นี้อยู่ในลักษณะของตารางลักษณะเนสเต็ดดังนั้นการเรียกดูข้อมูลในตารางลักษณะเนสเต็ดนั้นต้องทำการเรียกผ่านจากแถวที่ต้องการจากตารางหลักก่อนจึงสามารถเรียกใช้แถวในตารางลักษณะเนสเต็ดได้ โดยรายละเอียดในส่วนนี้สามารถดูได้จากภาคผนวก การใช้งาน SQL ในการติดต่อกับตารางลักษณะเนสเต็ด

ส่วนการส่งเงื่อนไขในการเรียกดูข้อมูลแทนที่จะส่งคำสั่ง SQL เข้ามานั้น เพราะว่าต้องใช้ คำสั่ง SQL ที่มีความซับซ้อนในการเรียกดูข้อมูลซึ่งไม่สะดวกในการทำงาน และถ้าไม่ใส่เงื่อนไขในการหาเมธอดนี้จะทำการแสดงข้อมูลทั้งหมดของแผนกที่อยู่ในร้านที่ระบุด้วย ShopID ออกมา

Update (ShopID : Integer)

ใช้ในการเปลี่ยนแปลงแก้ไขข้อมูล โดยต้องส่งค่า CategoryID ผ่านมาทางแอตทริบิวต์ด้วย และการที่ต้องผ่าน ShopID เป็นพารามิเตอร์นั้นมีเหตุผลเช่นเดียวกับเมธอด AddNew และคุณสมบัติรวมถึงปัญหาของเมธอด Update นี้ยังคงมีเช่นเดียวกับเมธอด Update ของคลาส Shop

Delete(ShopID : Integer, Criteria : String)

ใช้ในการลบข้อมูลของแผนกออกจากร้าน ซึ่งจำเป็นต้องใช้ ShopID ในการระบุร้านที่ต้องการลบแผนกออกซึ่งการลบนี้จะทำการลบสินค้าที่อยู่ภายใต้แผนกนี้ด้วย

6.6.1.3 คลาส ITEM

เป็นคลาสที่สร้างขึ้นมาเพื่อจัดการข้อมูลของสินค้า แต่เราทำการแยกคลาส Item นี้ออกจากคลาส Category คือไม่ทำคลาส Item เป็นแอตทริบิวต์หนึ่งในคลาส Category เพราะว่า คลาส Item นี้มีมีความเกี่ยวเนื่องกับคลาสอื่นด้วยไม่ใช้เฉพาะคลาส Category

แอตทริบิวต์

ItemID : Integer	หมายเลขของสินค้า ซึ่งสินค้าในทุกร้านจะมีหมายเลขสินค้าที่ไม่ซ้ำกัน
ShopID : Integer	หมายเลขร้าน
CategoryID : Integer	หมายเลขแผนก
ItemName : String	ชื่อสินค้า
Picture : Long Raw	ใช้ในการเก็บรูปภาพลงฐานข้อมูล ดูรายละเอียดในภาคผนวก การเก็บข้อมูลมัลติมีเดีย

Sound : String	เก็บตำแหน่งของไฟล์เสียงที่ได้จากการแปลง Text-To-Speech ซึ่งเป็นคอมโพเนนต์ส่วนของฟรอนท์เอนด์
Video : String	เก็บตำแหน่งของไฟล์วิดีโอที่ใช้ในการส่งข้อมูลแบบสตรีม (Stream Video) ซึ่งใช้ Microsoft Media Server ซึ่งอยู่ในการทำงานส่วนฟรอนท์เอนด์
Discount : Single	ส่วนลดของสินค้า
Barcode : String	เก็บ Barcode ของสินค้า
Status : String	เก็บสถานะของสินค้าซึ่งมีความสัมพันธ์กับ UnitInStock และมีความสัมพันธ์กับในส่วนของการซื้อของด้วย ซึ่งจะนำมาอธิบายในหัวข้อการกำหนดข้อบังคับในแอบพลิเคชั่นเซิร์ฟเวอร์ (Business Rule In Application Server)
Price : Single	ราคาสินค้า
CreateOn : Date	วันที่ใส่ข้อมูลของสินค้า
Description : String	คำอธิบายของสินค้า
UnitInStock : Integer	จำนวนที่สินค้าเหลืออยู่ในคลังสินค้า
UnitOnOrder : Integer	จำนวนสินค้าที่อยู่ในขั้นตอนการสั่งซื้อ
F1 – F20 : String	คุณสมบัติพิเศษ ซึ่งขึ้นอยู่กับแผนกที่สินค้าอยู่ โดยค่าที่อยู่ในคลาส Category นั้นจะเป็นชื่อของคุณสมบัติพิเศษส่วนค่าที่อยู่ในคลาส Item นี้จะเป็นค่าของคุณสมบัตินั้น เช่นในร้านค้าหนึ่งมีแผนกหนังสือ ค่า F1 ในคลาส Category เป็นคำว่า “ผู้แต่ง” และค่า F1 ในคลาส Item จะเป็นชื่อของผู้แต่งหนังสือเล่มนั้น

เมธอด

AddNew() : ItemID

เป็นการเพิ่มสินค้าเข้าไปร้านและแผนกที่ต้องการ โดยการส่งผ่านข้อมูลที่จำเป็นผ่านทางแอตทริบิวต์ ซึ่งเมธอดนี้จะไม่มีส่วนที่ทำการใส่รูปภาพลงฐานข้อมูล เนื่องจากการทำงานในส่วนนี้มีความซับซ้อนเกินกว่าที่จะรวมกับการในข้อมูลปกติในเมธอดนี้ได้ ซึ่งเราได้แบ่งการทำงานในส่วนนี้ให้ส่วนของฟรอทเอ็นท์ทำการควบคุมแทน แต่ว่าวิธีการใส่ข้อมูลที่เป็นรูปภาพนี้เราจะกล่าวถึงใน ภาคผนวก การเก็บข้อมูลมัลติมีเดีย และเมื่อทำการใส่ข้อมูลของสินค้าเรียบร้อยแล้วจะส่งหมายเลขของสินค้าออกมา

Find(SQLString : String) : Recordset

ใช้ในการหาข้อมูลของสินค้าโดยทำการส่งคำสั่ง SQL เพื่อทำการดึงเซตของข้อมูลสินค้าขึ้นมา ถ้าไม่กำหนด SQLString เมธอดนี้จะเรียกข้อมูลของสินค้าทั้งหมดขึ้นมา

Update()

ใช้ในการเปลี่ยนแปลงแก้ไขข้อมูลสินค้าโดยต้องส่งค่า ItemID ผ่านทางแอตทริบิวต์เข้าไปด้วย โดยมีคุณสมบัติเช่นเดียวกับเมธอด Update ในคลาส Shop

Delete(Criteria : String)

ใช้ในการลบสินค้าโดยใช้เงื่อนไขในการลบ

6.6.1.4 คลาส BILLING

สร้างขึ้นมาเพื่อรองรับการสั่งซื้อสินค้า

แอตทริบิวต์

BillingID : Integer	หมายเลขของใบรายการสั่งซื้อ
AccountID : Integer	หมายเลขสมาชิกของลูกค้า
BillingDate : Date	วันออกใบรายการสั่งซื้อ
ReqDate : Date	กำหนดวันที่สินค้าต้องส่งถึงลูกค้า
PayType : String	วิธีการจ่ายเงิน
Paid : String	สถานะว่าจ่ายเงินเสร็จสมบูรณ์แล้วหรือไม่
Freight : Single	ค่าธรรมเนียมในการส่งสินค้า
ShipName : String	ส่งในนาม
ShipDate : Date	วันที่เริ่มส่งสินค้า
ShipAddress : String	ที่อยู่ที่สินค้าต้องส่งไป
ShipVia : String	วิธีการส่งสินค้า เช่นส่งโดยเครื่องบิน ส่งโดยเรือ
ShipCity : String	เมืองหรือเขตที่สินค้าต้องส่งไป
ShipRegion : String	จังหวัดที่สินค้าต้องส่งไป
ShipPostcode : String	รหัสไปรษณีย์ที่สินค้าต้องส่งไป
ShipCountry : String	ประเทศที่สินค้าต้องส่งไป
Status : String	สถานะของสินค้าทั้งหมดในใบรายการสั่งซื้อ
TotalTax : Single	ภาษีรวม
TotalPrice : Single	ราคารวม
BillingAddress : String	ที่อยู่ที่ให้ส่งใบเสร็จไป
BillingName : String	ชื่อที่ให้ลงในใบเสร็จ
BillingSurname : String	นามสกุลที่ให้ลงในใบเสร็จ
BillingCity : String	เมืองหรือเขตของผู้ซื้อที่ลงในใบเสร็จ
BillingState : String	รัฐของผู้ซื้อที่ลงในใบเสร็จ

BillingPostcode : String	รหัสไปรษณีย์ของผู้ซื้อที่ลงในใบเสร็จ
BillingCountry : String	ประเทศของผู้ซื้อที่ลงในใบเสร็จ
BillingTel : String	เบอร์โทรศัพท์ของผู้ซื้อที่ลงในใบเสร็จ
BillingEmail : String	อีเมล์ที่ใช้ในการติดต่อกับผู้ซื้อ
BillingFax : String	เบอร์แฟกซ์ของผู้ซื้อที่ลงในใบเสร็จ
PagerNumber : String	เบอร์วิทยุติดตามตัวที่ใช้ในการติดต่อกับผู้ซื้อ
PagerType : String	ประเภทของเครื่องมือสื่อสาร (เพจเจอร์ โทรศัพท์มือถือ) ที่ใช้ในการติดต่อกับผู้ซื้อ
PagerPassword : String	รหัสผ่านสำหรับขอใช้บริการสำหรับการติดต่อผ่านบริการของเครื่องมือสื่อสารผ่าน Internet
BillingDetail : Class	คลาสรายละเอียดของใบสั่งซื้อว่ามีสินค้าใดบ้างที่ถูกสั่งซื้อ

เมธอด

AddBilling() : BillingID

ใช้ในการสร้างในรายการสั่งซื้อ จากข้อมูลที่ส่งผ่านแอตทริบิวต์มา โดยเมื่อทำการสร้าง รายการสั่งซื้อเสร็จจะส่งหมายเลขในรายการสั่งซื้อที่ได้ออกมา

FindBilling (SQLString : String) : Recordset

ใช้ในการค้นหาข้อมูลของใบรายการสั่งซื้อโดยส่งคำสั่ง SQL เข้าไปแล้วจะได้เซตของข้อมูลใบสั่งซื้อออกมา ซึ่งถ้าไม่มีคำสั่ง SQL เข้าไปจะทำการส่งข้อมูลของใบรายการสั่งซื้อทั้งหมดออกมา

DeleteBilling(Criteria : String)

ใช้ลบใบสั่งซื้อโดยส่งเงื่อนไขในการลบเข้าไป

UpdateBilling()

ใช้ในการเปลี่ยนแปลงข้อมูลของใบรายการสั่งซื้อโดยต้องส่งค่า BillingID ผ่านทางแอตทริบิวต์เข้าไปเพื่อระบุรายการสั่งซื้อที่ต้องการเปลี่ยนแปลงค่า

CalTotalTax() : Single

ใช้ในการคำนวณภาษีจากรายการสินค้าที่อยู่ในใบรายการสั่งซื้อสินค้า โดยต้องผ่าน BillingID เข้าไปทางแอตทริบิวต์ เพื่อใช้ในการระบุใบรายการสั่งซื้อที่ต้องการคำนวณภาษีใหม่ ซึ่งผลที่ได้คือทำการบันทึกค่าภาษีใหม่ที่ได้ลงในใบรายการสั่งซื้อและส่งค่าภาษีที่ได้นั้นออกมา

CalTotalPrice() : Single

ใช้ในการคำนวณราคารวมของสินค้าที่อยู่ในใบรายการสั่งซื้อสินค้า โดยต้องผ่าน BillingID เข้าไปทางแอตทริบิวต์เพื่อทำการระบุถึงใบรายการสั่งซื้อที่ต้องการทำการคำนวณราคาสินค้ารวมใหม่ ซึ่ง จะทำการบันทึกค่าใหม่ที่ได้และส่งค่านั้นออกมาจากเมธอดนี้ด้วย

UpdateBillingStatus()

ใช้ในการเปลี่ยนแปลงสถานะของใบรายการสั่งซื้อว่าอยู่ในสถานะใด ซึ่งจะอธิบายโดยละเอียดอีกครั้งในหัวข้อการกำหนดข้อบังคับในแอบพลิเคชันเซิร์ฟเวอร์

6.6.1.5 คลาส BILLINGDETAIL

สร้างขึ้นมาเพื่อทำการเก็บข้อมูลของสินค้าที่ลูกค้าเลือกซื้อไว้ ซึ่งคลาสนี้เป็นแอตทริบิวต์หนึ่งในคลาส Billing

แอตทริบิวต์

BillingID : Integer	หมายเลขของใบรายการสั่งซื้อย่อยซึ่งมีค่าไม่ซ้ำกัน
ItemID : Integer	หมายเลขของสินค้า
Status : String	สถานะของรายการสั่งซื้อย่อย
Discount : Single	ส่วนลดของรายการสั่งซื้อย่อย
Amount : Integer	จำนวนสินค้าที่สั่งซื้อ
TaxValue : Single	ภาษีของสินค้า
Price : Single	ราคาของสินค้า

เมธอด

AddBillingDetail()

ใช้ในการสร้างรายการสั่งซื้อย่อยโดยต้องผ่านค่า BillingID เข้าไปทางแอตทริบิวต์ด้วยและจะทำการลดค่า UnitInStock และเพิ่มค่า UnitOnOrder ตามจำนวนของสินค้าที่ถูกสั่งซื้อด้วย

สาเหตุที่มีแอตทริบิวต์ BillingID เพราะโครงสร้างของข้อมูลในฐานข้อมูลระหว่าง Billing และ BillingDetail ไม่ได้เป็นแบบตารางลักษณะเนสเต็ดแต่เป็นการ Join ของสองตารางจึงต้องมีคีย์หลักของตาราง Billing เกาะอยู่บนตาราง BillingDetail ซึ่งสาเหตุที่ไม่ใช้ตารางลักษณะเนสเต็ดเพราะว่าเราต้องมีการค้นหาข้อมูลจากตารางย่อยออกมายังตารางใหญ่ ซึ่งตารางลักษณะเนสเต็ดทำไม่ได้ สามารถดูรายละเอียดได้ในภาคผนวก การใช้งาน SQL ในการติดต่อกับตารางลักษณะเนสเต็ด

FindBillingDetail(SQLString : String) : Recordset

ใช้ในการเรียกดูข้อมูลรายการสั่งซื้อย่อย ซึ่งถ้าไม่ส่งคำสั่ง SQL เข้าไปจะแสดงข้อมูลรายการสั่งซื้อย่อยทั้งหมดออกมา

DeleteBillingDetail(Criteria : String)

ใช้ในการลบรายการสั่งซื้อย่อยตามเงื่อนไขที่ส่งเข้าไป

UpdateBillingDetail()

ใช้ในการเปลี่ยนแปลงแก้ไขข้อมูลของแต่ละรายการสั่งซื้อ โดยต้องผ่านค่า BillingID และItemID เข้าไปทางแอตทริบิวต์เพื่อทำการระบุถึงรายการสั่งซื้อย่อยที่ต้องการเปลี่ยนแปลงข้อมูล

CalPrice() : Single

ทำการคำนวณราคาของรายการสินค้าย่อยนั้น โดยจะทำการบันทึกค่าใหม่ที่ได้และส่งค่านั้นออกมาด้วย

CalTax() : Single

ทำการคำนวณภาษีของรายการสินค้าย่อยนั้น โดยจะทำการบันทึกค่าใหม่ที่ได้และส่งค่านั้นออกมาด้วย

6.6.1.6 คลาส SHOPPINGBAG

สร้างขึ้นมาเพื่อเป็นที่พักข้อมูลสินค้าที่ลูกค้าสนใจซื้อ ซึ่งเมื่อมองดูแล้วน่าเป็นคลาสที่มีการทำงานอยู่ฝั่งฟรอนท์เอนด์แต่เราได้เพิ่มคุณสมบัติให้ลูกค้าสามารถทำการเก็บข้อมูลที่เขาเลือกไว้ได้ จึงทำให้คลาสนี้สามารถทำงานทั้งสองฝั่งได้ โดยจะทำงานส่วนแบคก์เอนด์เมื่อลูกค้าสั่งให้เก็บสินค้าที่เขาเลือกไว้โดยยังไม่ได้เข้าสู่ขั้นตอนการจ่ายเงิน

แอตทริบิวต์

ItemID : Integer	หมายเลขสินค้า
AccountID : Integer	หมายเลขของสมาชิก
Amount : Integer	จำนวนสินค้าที่เลือกซื้อ
Discount : Single	ส่วนลดของสินค้าในรูปแบบเปอร์เซ็นต์
TaxRate : Single	ภาษีของสินค้าในรูปแบบเปอร์เซ็นต์
Price : Single	ราคาสินค้า

เมธอด

LoadShoppingBag(AccountID : Integer) : Recordset

ใช้ในการแสดงข้อมูลสินค้าที่เขาเคยเลือกไว้ โดยใช้ AccountID เป็นตัวระบุ

SaveShoppingBag(Recordset) : Boolean

ใช้ในการเก็บข้อมูลสินค้าที่ลูกค้าได้เลือกไว้ โดยข้อมูลที่จะทำการเก็บนั้นจะถูกส่งมาในรูปแบบของตัวแปรชนิด Recordset ซึ่งถ้าทำการเก็บข้อมูลเรียบร้อยจะแสดงค่าตรรกว่าถูกต้องออกไป ซึ่งในกรณีนี้หมายความว่าลูกค้าสามารถออกจากระบบ แล้วผ่านไปหลายๆ วันเขาสามารถกลับมาเรียกดูสินค้าที่เขาเคยเลือกไว้ได้

DeleteShoppingBag(AccountID : Integer)

ใช้ในการลบข้อมูลสินค้าที่ลูกค้าได้เลือกไว้ในฐานข้อมูล โดยการลบนี้จะลบสินค้าทุกชิ้นที่ลูกค้าได้สั่งให้เก็บไว้

6.6.1.7 คลาส ACCOUNT

สร้างขึ้นมาเพื่อทำการเก็บข้อมูลของลูกค้าที่ต้องการซื้อสินค้า โดยเราได้กำหนดเงื่อนไขว่าลูกค้าทุกคนที่ต้องการซื้อของในระบบเราต้องเป็นสมาชิกของระบบทุกคน

แอตทริบิวต์

AccountID : Integer	หมายเลขสมาชิก
CusName : String	ชื่อของสมาชิก
CusSurname : String	นามของสกุลสมาชิก
Email : String	อีเมล์ของสมาชิกซึ่งใช้เป็น Username
Password : String	รหัสผ่านของสมาชิก
Sex : String	เพศ
Age : Integer	อายุ
Occupation : String	อาชีพ
Address : String	ที่อยู่
Province : String	จังหวัด
Subprovince : String	เขต
Postcode : String	รหัสไปรษณีย์
Country : String	ประเทศ
Tel : String	เบอร์โทรศัพท์
Fax : String	เบอร์แฟกซ์

เมธอด

AddAccount() : Integer

ใช้ในการเพิ่มสมาชิกใหม่โดยส่งข้อมูลที่ใช้ผ่านมาทางแอตทริบิวต์ และส่งหมายเลขสมาชิกที่ใหม่ออกมา

DeleteAccount(Criteria : String)

ใช้ในการลบสมาชิกออกจากระบบโดยใช้เงื่อนไขในการลบ

UpdateAccount()

ใช้ในการเปลี่ยนแปลงข้อมูลของสมาชิก ซึ่งต้องส่ง AccountID เข้ามาทางแอตทริบิวต์ด้วยเพื่อใช้ในการระบุว่าต้องการแก้ไขข้อมูลสมาชิกคนใด

GetInformation(SQLString : String) : Recordset

ใช้ในการค้นหาข้อมูลของสมาชิก โดยใช้คำสั่ง SQL ถ้าไม่ได้ส่งคำสั่ง SQL เข้ามาจะทำการแสดงข้อมูลของสมาชิกทุกคนที่มีออกมา

โดยสรุปแล้วจะเห็นได้ว่าเมธอดของแต่ละคลาสจะมีความคล้ายกันดังนั้นจะมีอินเทอร์เฟส และคุณสมบัติการทำงานที่คล้ายกันทำให้ระบบมีความซับซ้อนน้อยลงได้

และการที่ฟังก์ชันการค้นหาข้อมูลส่วนใหญ่จะส่งคำสั่ง SQL ลงไปนั้น เป็นเพราะการทำงานในระดับล่างนั้นควรมีความยืดหยุ่นที่มากพอสมควรจึงทำให้การทำงานในระดับสูงขึ้นไปนั้นออกแบบได้ง่ายขึ้น ทำให้การซ่อนรายละเอียดในระดับล่างลดลง เช่นการใช้คำสั่ง SQL นี้ทำให้คอมโพเนนต์ในระดับบนต้องรู้จักชื่อตารางในฐานข้อมูลด้วย ซึ่งเราจำเป็นต้องยอมแลกกับข้อเสียนี้

6.6.1.8 คลาส SHIPMENT

ใช้ในการจำลองการจัดการกับข้อมูลและรูปแบบการส่งสินค้า

แอตทริบิวต์

ShipVia	รูปแบบการส่งสินค้า เช่นส่งทางเครื่องบิน ส่งทางเรือ
ShipCompany	บริษัทที่รับผิดชอบในการส่งสินค้า
Freight	ค่าขนส่ง
ShipPeriod	ระยะเวลาที่ใช้ในการขนส่ง

เมธอด

Update()

ใช้ในการแก้ไขข้อมูลการส่งสินค้า ซึ่งเป็นการจำลองเท่านั้นจึงมีเพียงบริษัทแห่งเดียวที่รับผิดชอบการขนส่งสินค้า

Retrieve()

ใช้ในการแสดงข้อมูลของการส่งสินค้าโดยแสดงออกมาทางแอตทริบิวต์ของคลาสนี้

6.6.2 ลักษณะการทำงานที่ส่วนแบคก์เอนด์สนับสนุน

เมื่อมองในภาพรวมของระบบแล้วจะแบ่งฟังก์ชันการทำงานหลักออกได้ดังนี้

6.6.2.1 การตั้งร้าน

- ทำการสร้างร้านใหม่ขึ้นมาโดยใช้ Shop.AddNew() ทำการสร้างร้านขึ้นมาใหม่โดยผ่านข้อมูลของร้านที่ต้องการสร้าง ทางแอตทริบิวต์ของคลาส Shop
- ทำการสร้างแผนกของสินค้าในร้าน โดยใช้ Shop.Category.AddNew() (ในการสร้างออบเจ็กต์ขึ้นมาจากคลาส Shop คลาส Category จะถูกสร้างขึ้นมาด้วย) และผ่านข้อมูลของแผนกสินค้าที่ต้องการสร้าง ทางแอตทริบิวต์ของคลาส Category
- ทำการใส่สินค้าต่างๆ ลงในแผนกสร้างไว้ โดยใช้ Item.AddNew()

6.6.2.2 การเปิดร้าน

หลังจากเสร็จสิ้นขั้นตอนการตั้งร้านระบบจะทำการนำเสนอร้านใหม่และสินค้าใหม่ในร้านนั้นเองโดยเจ้าของร้านไม่ต้องจัดการเอง โดยในส่วนนี้เป็นหน้าที่ของคอมโพเนนต์ด้านฟรอนท์เอนด์ซึ่งเรียกใช้การทำงานด้านแบคก์เอนด์ดังนี้

- เรียกใช้ Shop.Find() เพื่อทำการแสดงข้อมูลของร้านค้า
- เรียกใช้ Shop.Category.Find() เพื่อแสดงรายละเอียดของคุณสมบัติที่แต่ละแผนกมี
- เรียกใช้ Item.Find() เพื่อหาข้อมูลของสินค้าภายในร้านและภายในแผนกที่ระบุหรืออาจเป็นรายการสินค้าใหม่ซึ่งรูปแบบของข้อมูลนั้นขึ้นอยู่กับคำสั่ง SQL ที่ส่งเข้ามา

6.6.2.3 การจัดร้าน

เป็นการแก้ไขข้อมูลไม่ว่าจะเป็นการแก้ไขข้อมูลร้านค้า ข้อมูลแผนกสินค้าในร้าน หรือข้อมูลของสินค้าในร้าน ซึ่งสามารถใช้เมธอดของแต่ละคลาสในการแก้ไขข้อมูลในส่วนต่างๆ ได้

- Shop.Update() เพื่อแก้ไขข้อมูลของร้านค้า
- Shop.Category.Update() แก้ไขข้อมูลของแผนกสินค้าที่มีอยู่

- Shop.Category.AddNew() เพิ่มแผนกใหม่ขึ้นภายในร้าน
- Shop.Category.Delete() ลบแผนกเดิม ซึ่งจะเป็นการลบสินค้าที่อยู่ภายในแผนกนั้นด้วย
- Item.Update() แก้ไขข้อมูลของสินค้าที่มีอยู่
- Item.AddNew() เพิ่มสินค้าใหม่
- Item.Delete() ลบสินค้าที่มีอยู่

6.6.2.4 การขายสินค้า

การขายนั้นเกิดขึ้นตั้งแต่ในขั้นตอนที่ลูกค้าเลือกสินค้าใส่ Shopping Bag จนถึงขั้นตอนที่ลูกค้าทำการจ่ายเงิน (ทำใบรายการสั่งซื้อสินค้า) ซึ่งคลาสและเมธอดที่เกี่ยวข้องมีดังนี้

- ShoppingBag.SaveShoppingBag ใช้ในการเก็บข้อมูลสินค้าที่ลูกค้าเลือก ไว้ในระบบซึ่งเมื่อเขาออกจากระบบไปและเมื่อเขาเข้ามาในระบบอีกครั้งเขาสามารถเรียกดูข้อมูลนี้ได้
- ShoppingBag.LoadShoppingBag ใช้ในการเรียกดูข้อมูลสินค้าที่เขาได้เคยเก็บไว้
- Billing.AddBilling ใช้ในการสร้างใบรายการสั่งซื้อในกรณีที่ลูกค้าต้องการซื้อสินค้า
- Billing.AddBillingDetail ใช้ในการใส่ข้อมูลของสินค้าที่อยู่ในใบรายการสั่งซื้อ

6.6.3 โครงสร้างและข้อกำหนดในฐานข้อมูล

รูปแบบของโครงสร้างฐานข้อมูลที่ได้จะมีลักษณะเหมือนกับคลาสไดอะแกรมที่ไม่มีเมธอด ที่ได้อย่างนี้เป็นเพราะว่า หลักการของการออกแบบเชิงวัตถุเรานำทั้งข้อมูลและการกระทำมารวมกันเป็นออบเจ็กต์ออกมา ดังนั้นส่วนที่เป็นข้อมูลคือส่วนที่เป็นแอตทริบิวต์

โดยในหัวข้อนี้จะอธิบายถึงตารางทั้งหมดที่สร้างขึ้นมาและข้อกำหนดในฐานข้อมูล เพื่อใช้งานในด้านแบคก์เอนด์ รวมถึงการนำความสามารถของฐานข้อมูลออราเคิลมาใช้

6.6.3.1 ตาราง Shop

ในตาราง Shop นี้จะประกอบด้วยตารางลักษณะเนสเต็ด ซึ่งคือตาราง Category ดังนั้นจึงต้องสร้างออบเจ็กต์ไทพของ Category ขึ้นมาแล้วสร้างเป็นเทเบิลไทพ์ ต่อจากนั้นนำเทเบิลไทพนี้ไปรวมกับออบเจ็กต์ไทพของ Shop แล้วจึงเปลี่ยนมาเป็นตารางดังนี้

```
Create or Replace Type TCategory As Object (
  CategoryID Number, CategoryType Varchar2(30),
  F1 Varchar2(30), F2 Varchar2(30), F3 Varchar2(30), F4 Varchar2(30),
  F5 Varchar2(30), F6 Varchar2(30), F7 Varchar2(30), F8 Varchar2(30),
  F9 Varchar2(30), F10 Varchar2(30), F11 Varchar2(30), F12 Varchar2(30),
  F13 Varchar2(30), F14 Varchar2(30), F15 Varchar2(30), F16 Varchar2(30),
  F17 Varchar2(30), F18 Varchar2(30), F19 Varchar2(30), F20 Varchar2(30) );
/

CREATE TYPE TCategory_T AS TABLE OF TCategory;
/

CREATE Or Replace Type Shop as object(
  ShopID NUMBER, ShopNAME VARCHAR2(30), OwnerName Varchar2(30),
```

```
  OwnerSURNAME VARCHAR2(30), EMAIL VARCHAR2(30),
  PASSWORD VARCHAR2(20), UserName Varchar2(20),
  SEX VARCHAR2(10), AGE NUMBER(3,0), OCCUPATION VARCHAR2(30),
  ADDRESS VARCHAR2(30), PROVINCE VARCHAR2(20), SUBPROVINCE VARCHAR2(20),
  POSTCODE VARCHAR2(20), Country VARCHAR2(30), Tel VARCHAR2(30),
  Fax VARCHAR2(30), CreateOn Date, WebTemplate Varchar2(30),
  Category TCategory_T );
/

Create Table TblShop of Shop
Nested Table Category Store As NCategory;
```

ข้อบังคับของตาราง Shop

- กำหนดคีย์หลัก

```
ALTER TABLE tblShop
ADD PRIMARY KEY (ShopID);
```

- Username และ ShopName ห้ามซ้ำ

```
Alter Table tblShop
ADD UNIQUE(Username);

Alter Table tblShop
ADD UNIQUE(Shopname);
```

6.6.3.2 ตาราง Category

เป็นตารางชนิดเนสเต็ดที่อยู่ภายในตาราง Shop สามารถใช้ชื่อ NCategory อ้างถึงตารางนี้ได้แต่จะไม่รู้ความสัมพันธ์กับตาราง Shop สามารถดูรายละเอียดได้ที่ภาคผนวก การใช้งาน SQL ในการติดต่อกับตารางลักษณะเนสเต็ด

ข้อบังคับของตาราง Category

- CategoryID ห้ามซ้ำ

```
Alter Table NCategory
ADD Primary Key (CategoryID);
```

- กำหนดเงื่อนไขการลบแบบต่อเนื่อง (Cascade Delete) ซึ่งทำในระดับฐานข้อมูลนั้นไม่สามารถกำหนดข้อกำหนดนี้ได้ จึงต้องเลื่อนข้อกำหนดนี้มาอยู่ที่แอบพลิเคชันเซิร์ฟเวอร์

6.6.3.3 ตาราง Item

ตารางนี้ไม่จำเป็นต้อง สร้างเป็นออบเจ็กต์ไทพก่อนเพราะไม่ต้องการใช้คุณสมบัติพิเศษของออราเคิล จะได้ตารางดังนี้

```
Create Table TblItem (
  ItemID NUMBER, ShopID Number, CategoryID Number, ItemName Varchar2(30),
  Picture Long Raw, Sound Varchar2(50), Video Varchar2(50),
  discount real, Barcode VARCHAR2(30), Status VARCHAR2(30), Price Real,
  CreateOn Date, Description VARCHAR2(255), UnitInStock NUMBER,
  UnitOnOrder NUMBER,
```

```
F1 Varchar2(50), F2 Varchar2(50), F3 Varchar2(50), F4 Varchar2(50),
F5 Varchar2(50), F6 Varchar2(50), F7 Varchar2(50), F8 Varchar2(50),
F9 Varchar2(50), F10 Varchar2(50), F11 Varchar2(50), F12 Varchar2(50),
F13 Varchar2(50), F14 Varchar2(50), F15 Varchar2(50), F16 Varchar2(50),
F17 Varchar2(50), F18 Varchar2(50), F19 Varchar2(50), F20 Varchar2(50)  );
```

ข้อบังคับของตาราง Item

- กำหนดคีย์หลัก

```
Alter Table tblItem
ADD Primary key(ItemID);
```

- กำหนดเงื่อนไขการลบแบบต่อเนื่อง

```
Alter Table tblItem
Add ( Foreign key (ShopID) References tblShop(Shopid)
      ON DELETE CASCADE);
```

6.6.3.4 ตาราง Account

สามารถสร้างเป็นตารางได้เลยดังนี้

```
Create Table TblAccount (
  AccountID Number, CusName Varchar2(30), CusSurname Varchar2(30),
  EMAIL VARCHAR2(50), PASSWORD VARCHAR2(20), SEX VARCHAR2(10),
  AGE NUMBER(3,0), OCCUPATION VARCHAR2(30), ADDRESS VARCHAR2(50),
  PROVINCE VARCHAR2(20), SUBPROVINCE VARCHAR2(20), POSTCODE VARCHAR2(20),
  Country VARCHAR2(50), Tel VARCHAR2(30), Fax VARCHAR2(30)  );
```

ข้อบังคับของตาราง Account

- กำหนดคีย์หลัก

```
Alter Table tblAccount
Add Primary key(AccountID);
```

- อีเมล์ของลูกค้าต้องไม่ซ้ำเพราะใช้เป็น Username ของลูกค้า

```
ALTER TABLE tblaccount
ADD UNIQUE(Email);
```

6.6.3.5 ตาราง Billing

ตาราง Billing นี้ตามหลักการของการออกแบบแล้วต้องมีตาราง BillingDetail เป็นตารางชนิดเนสเต็ดอยู่ภายใต้ตารางนี้ด้วย แต่เนื่องจากลักษณะของตารางลักษณะเนสเต็ดนี้ใช้กับความสัมพันธ์ที่เป็นแบบหนึ่งไปหลาย (One-To-Many) และทิศทางการเรียกดูข้อมูลจะเป็นแบบทางเดียว คือต้องหาจากตารางใหญ่ก่อนจึงเข้าไปหาในตารางย่อยได้ เท่านั้น

ซึ่งความสัมพันธ์ของ Billing กับ BillingDetail นั้นเป็นความสัมพันธ์แบบหลายไปหลาย (Many-To-Many) และมีการเรียกดูข้อมูลจากตารางย่อยออกมายังตารางใหญ่ (ถามจาก BillingDetail เพื่อหา Billing) ดังนั้นเราจึงไม่ใช้ความสัมพันธ์แบบเนสเต็ด

```
Create Table TblBilling(
  BillingID Number, AccountID Number, BillingDate Date, ReqDate Date,
```

```
PayType Varchar2(20), FaxConfirm Varchar2(3), EmailConfirm Varchar2(3),
ShipName Varchar2(30), ShipDate date, ShipAddress Varchar2(30),
ShipVia Varchar2(20), Frieght Real, ShipCity varchar2(30),
ShipRegion Varchar2(30), ShipPostcode Varchar2(20),
ShipCountry Varchar2(30), Status Varchar2(30), TotalTax Real,
TotalPrice Real, BillingAddress Varchar2(50), BillingName Varchar2(30),
BillingSurname Varchar2(30), BillingCity Varchar2(30),
BillingState Varchar2(30), BillingPostCode Varchar2(20),
BillingCountry Varchar2(30), BillingTel Varchar2(30),
BillingEmail Varchar2(50), BillingFax Varchar2(30),
Paid Varchar2(3) Default 'no'  );
```

ข้อบังคับของตาราง Billing

- กำหนดคีย์หลัก

```
Alter table tblBilling
Add Primary key(BillingID);
```

6.6.3.6 ตาราง BillingDetail

```
Create Table TblBillingDetail (
  BillingID Number, ItemID Number, Status Varchar2(30), Discount Real,
  Amount Number, TaxValue Real, Price Real );
```

ข้อบังคับของตาราง BillingDetail

- กำหนดคีย์หลักซึ่งเป็นการรวมกันของ BillingID และ ItemID (Combine key)

```
Alter Table tblBillingDetail
Add Primary key(BillingID, ItemID);
```

- กำหนดเงื่อนไขการลบแบบต่อเนื่อง

```
Alter Table tblBillingDetail
Add ( Foreign key (BillingID) References tblBilling(BillingID)
      ON DELETE CASCADE);
```

6.6.3.7 ตาราง ShoppingBag

เป็นตารางที่ใช้เก็บข้อมูลสินค้าที่มาจากฝั่งฟรอนท์เอนด์เท่านั้นจึงไม่มีการกำหนดข้อบังคับของตารางนี้

```
Create Table TblShoppingBag (
  ItemID Number, AccountID Number, Amount Number, Discount Real,
  TaxValue Real, Price Real );
```

6.6.3.8 ตารางอื่นๆ

เป็นตารางที่ใช้เก็บค่าคงที่รวมถึงส่วนที่ระบบไม่ได้ครอบคลุมถึง ซึ่งเป็นเพียงการจำลองการทำงานเท่านั้นจะมีดังนี้

6.6.3.8.1 ตารางค่าคงที่

เก็บอัตราภาษี และเก็บหมายเลขของสินค้า เนื่องจากว่าสินค้านั้นสามารถถูกเพิ่มหรือลบจากเจ้าของร้านได้เราจึงต้องทำให้หมายเลขสินค้านั้นไม่มีทางซ้ำกันเลยจึงนำมาเก็บในตารางนี้

เนื่องจากอัลกอริทึมในการสร้างหมายเลขสินค้านี้เราใช้ค่าสูงสุดของหมายเลขที่มีอยู่ในปัจจุบันแล้วเพิ่มค่าขึ้นไปอีกค่าหนึ่ง (Select Max(ID) From ...) ซึ่งอัลกอริทึมนี้ใช้ได้ในกรณีที่มีแทบไม่มีการลบเรคคอร์ดออกจากฐานข้อมูล แต่ใช้กับข้อมูลสินค้าไม่ได้เพราะมีการเปลี่ยนแปลงบ่อย

ส่วนหมายเลขของร้านค้า แผนก หมายเลขของลูกค้า และหมายเลขใบสั่งซื้อสินค้านั้นจะมีโอกาสที่จะมีการซ้ำกันน้อยกว่าหรือมีการถูกลบออกน้อยกว่าเราจึงใช้อัลกอริทึมแบบเก่าได้

```
Create Table TblConsts (
  TaxRate Real, ItemID Number);
```

6.6.3.8.2 ตารางจำลองการจ่ายเงินผ่านบัตรเครดิต

เป็นตารางที่ใช้ในการตรวจสอบบัตรเครดิตซึ่งเป็นตารางของธนาคารที่ระบบของเราจะต้องทำการตรวจสอบบัตรเครดิตของลูกค้าในขั้นตอนของการทำใบสั่งซื้อสินค้าว่าลูกค้ามีวงเงินเพียงพอที่จะทำการซื้อของได้หรือไม่ ซึ่งเป็นการทำงานของคลาส Payment ซึ่งเป็นคอมโพเนนต์ด้านฟรอนท์เอนด์ รูปแบบของตารางจะเป็นดังนี้ และไม่ได้กำหนดข้อบังคับของตารางนี้เพราะทำเพื่อจำลองการทำงานเท่านั้น

```
Create Table SimPayment (
  CreditCardNumber Varchar2(50), BankName Varchar2(30),
  ExpMonth Varchar2(10), ExpYear Varchar2(5) );
```

6.7 การจัดการเรื่องทรานแซคชั่น

เราได้ใช้ไมโครซอฟท์ทรานแซคชั่นเซิร์ฟเวอร์ในการจัดการเรื่องทรานแซคชั่น ดังที่กล่าวไปแล้วในบทแรกๆ โดยความสามารถและการใช้งานทรานแซคชั่นเซิร์ฟเวอร์นี้เหมาะกับระบบฐานข้อมูลแบบกระจายโดยฐานข้อมูลเหล่านั้นไม่ได้มาจากบริษัทเดียวกัน และให้แต่ละระบบฐานข้อมูลทำการคอมมิตภายในเอง (Local Commit) โดยทรานแซคชั่นเซิร์ฟเวอร์จะจัดการการคอมมิตรวม (Global Commit) เมื่อทุกระบบฐานข้อมูลมีการคอมมิตภายในทุกระบบแล้ว ซึ่งเป็นหลักการของทูเฟสคอมมิต (Two-Phase Commit)

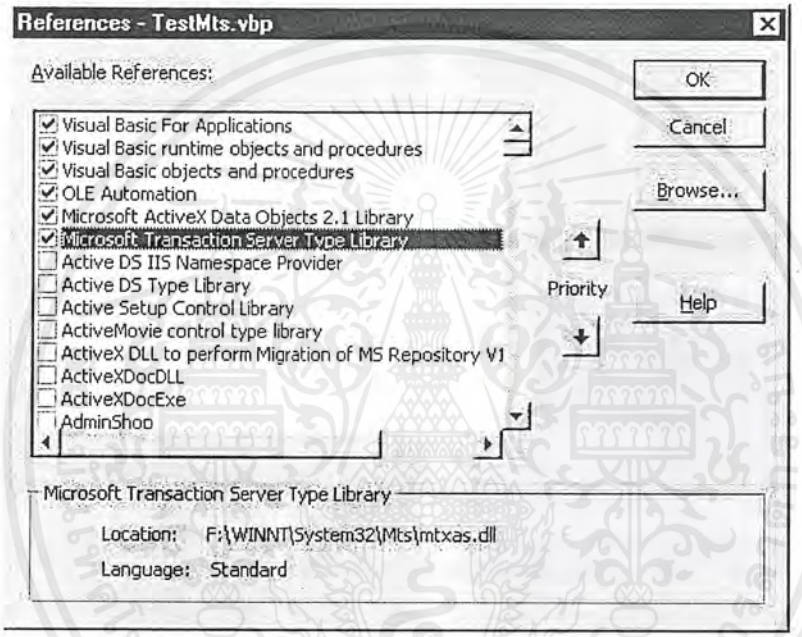
สำหรับในการทำโครงงานนี้เราออกแบบให้มีระบบฐานข้อมูลเพียงที่เดียวแต่เรายังใช้ไมโครซอฟท์ทรานแซคชั่นเซิร์ฟเวอร์เพราะ

- เราต้องการความสามารถในการสร้างการทำงานแบบ DCOM ซึ่งจำเป็นสำหรับการทำงานไคลเอนต์เซิร์ฟเวอร์ชนิด 3 ระดับ (Three-Tiered Architecture)
- เป็นการวางโครงสร้างการขยายของระบบว่าสามารถรองรับกับการทำระบบฐานข้อมูลแบบกระจาย

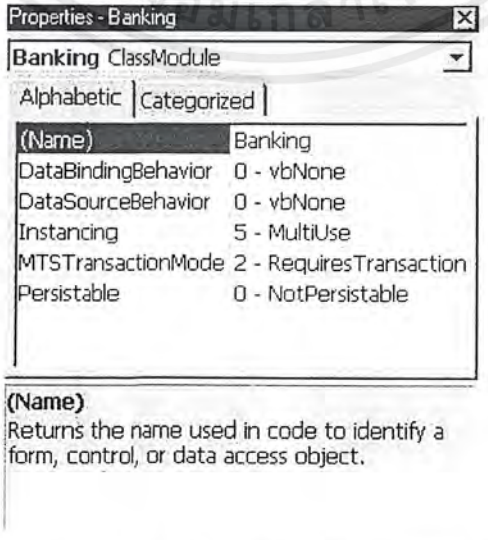
สำหรับในหัวข้อนี้จะกล่าวถึงการจัดการเรื่องทรานแซคชั่นในระดับโปรแกรมว่าทำได้อย่างไรซึ่งก่อนอื่นต้องทำการติดตั้งให้ไมโครซอฟท์ทรานแซคชั่นเซิร์ฟเวอร์นั้นใช้งานกับระบบจัดการฐานข้อมูลออราเคิลก่อน โดยจะแสดงรายละเอียดไว้ในภาคผนวก การติดตั้งไมโครซอฟท์ทรานแซคชั่นเซิร์ฟเวอร์

ขั้นตอนการโปรแกรมโดยใช้คอมโพเนนต์ของไมโครซอฟท์ทรานแซคชั่นเซิร์ฟเวอร์มีดังนี้

- สร้างคอมโพเนนต์ที่ต้องการใช้งานทรานแซคชั่นโดยสร้างเป็น ActiveX DLL
- ทำการอ้างเพื่อเรียกใช้ไมโครซอฟท์ทรานแซคชั่นเซิร์ฟเวอร์คอมโพเนนต์
- กำหนดคุณลักษณะ (Property) ของคอมโพเนนต์ที่เราสร้างขึ้นให้เรียกใช้การจัดการทรานแซคชั่น ซึ่งต้องเลือก MTSTransactionMode เป็น RequiresTransaction



รูปที่ 6-7 การอ้างถึงไมโครซอฟท์ทรานแซคชั่นเซิร์ฟเวอร์คอมโพเนนต์



รูปที่ 6-8 การกำหนดค่าคุณสมบัติของคอมโพเนนต์ให้สามารถใช้ทรานแซคชั่นได้

- ทำการเขียนโปรแกรมควบคุมดังนี้

```
Public Sub Delete(ByVal Criteria As String)
Dim tx As ObjectContext
' --- Start Transaction ---
Set tx = GetObjectContext()
On Error GoTo DeleteShopError        ' Set Error Handler
' ----- Begin in Transaction
' การทำงานที่อยู่ภายในทรานแซคชัน
' -- Commit Transaction --
tx.SetComplete

Exit Sub                ' Exit
DeleteShopError:
' - Abort Transaction
    tx.SetAbort
    Err.Raise Err.Number, , Err.Description        ' Return Error
End Sub
```

- เมื่อสร้างคอมโพเนนต์เสร็จแล้วให้กำหนดค่า Version Compatibility เป็น Binary Compatibility ด้วย
- ทำการติดตั้งคอมโพเนนต์ที่ได้ลงในไมโครซอฟท์ทรานแซคชั่นเซิร์ฟเวอร์ ซึ่งสามารถดูรายละเอียดได้ในภาคผนวก การติดตั้งคอมโพเนนต์ลงในไมโครซอฟท์ทรานแซคชั่นเซิร์ฟเวอร์

ส่วนผลการทำงานของคอมโพเนนต์ที่เรียกใช้ทรานแซคชั่นนี้จะนำเสนอในบทต่อไปคือ ผลการทดสอบระบบ

6.8 การกำหนดข้อบังคับในแอบพลิเคชันเซิร์ฟเวอร์

เหตุผลที่เราเลื่อนข้อบังคับบางส่วนขึ้นมาไว้ที่แอบพลิเคชันเซิร์ฟเวอร์มีดังนี้

- แบ่งเบาการทำงานของระบบฐานข้อมูล
- การกำหนดข้อบังคับบนแอปพลิเคชั่นเซิร์ฟเวอร์ทำได้ง่าย และสามารถสร้างข้อบังคับที่มีความซับซ้อนได้

โดยข้อบังคับที่อยู่ในแอปพลิเคชั่นเซิร์ฟเวอร์ของระบบนี้คือ

6.8.1 การลบต่อเนื่องของ Category กับ Item

Category นั้นเป็นข้อมูลแผนกสินค้าในร้าน ถ้าแผนกของสินค้าถูกลบสินค้าที่อยู่ภายใต้แผนกนั้นจะถูกลบไปด้วย ซึ่งการกำหนดข้อบังคับที่ระบบฐานข้อมูลทำไม่ได้เพราะตาราง Category นั้นเป็นตารางแบบเนสเต็ดสามารถสร้างคีย์หลักได้แต่ไม่สามารถนำคีย์หลักนี้ไปสร้างเป็นฟอร์เรนคีย์ (Foreign key) ของตาราง Item ได้ จึงต้องนำคำสั่งการลบข้อมูลสินค้าไปใส่ไว้ในเมธอด Delete() ของคลาส Category

6.8.2 การเปลี่ยนสถานะของสินค้าและใบรายการสั่งซื้อ

คลาสที่มีข้อมูลเกี่ยวกับสถานะจะมีดังนี้คือ คลาส Item คลาส Billing และคลาส BillingDetail ซึ่งจะมีรูปแบบของสถานะดังนี้

6.8.2.1 Item.Status

- Available เมื่อ Item.UnitInStock มีค่ามากกว่า 0
- Out Of Stock เมื่อ Item.UnitInStock มีค่าน้อยกว่าหรือเท่ากับ 0 ที่กำหนดให้มีค่าน้อยกว่า 0 ได้เพราะเจ้าของร้านจะได้รู้ว่ามีลูกค้าสั่งซื้อสินค้าแล้วไม่มีสินค้าให้กับลูกค้าเป็นจำนวนเท่าใด

6.8.2.2 BillingDetail.Status

มีสถานะเช่นเดียวกับสินค้าที่อยู่ใน BillingDetail เช่น ถ้าสินค้าที่อยู่ใน BillingDetail มีสถานะเป็น Available จะได้ BillingDetail.Status เป็น Available ด้วย

6.8.2.3 Billing.Status

Available for Send All	สินค้าทุกชิ้นภายในรายการสั่งซื้อนั้นพร้อมส่ง และทำการจ่ายเงินแล้ว
Wait for Send All	สินค้าทุกชิ้นภายในรายการสั่งซื้อนั้นพร้อมส่ง แต่ยังไม่ได้ทำการจ่ายเงิน
Available for Send Some	สินค้าบางชิ้นภายในรายการสั่งซื้อนั้นพร้อมส่ง และทำการจ่ายเงินแล้ว
Wait for Send Some	สินค้าบางชิ้นภายในรายการสั่งซื้อนั้นพร้อมส่ง แต่ยังไม่ได้ทำการจ่ายเงิน

6.9 การเตรียมความพร้อมก่อนการรวมกับคอมโพเนนต์ฝั่งฟรอนท์เอนด์

คอมโพเนนต์ทุกตัวที่ได้จากส่วนข องแบคก์เอนด์นี้ต้องทำการทดสอบให้แน่ใจว่าสามารถทำงานได้อย่างถูกต้องเพื่อว่าจะช่วยลดความผิดพลาด และช่วยลดเวลาที่ต้องใช้ในการค้นหาความผิดพลาดเมื่อนำไปรวมกับคอมโพเนนต์ด้านฟรอนท์เอนด์

โดยทางด้านของฟรอนท์เอนด์คอมโพเนนต์ที่ได้ส่วนใหญ่จะเป็น WebClass Application หรือ IIS Application ซึ่งเปรียบเสมือนเป็นไคลเอนต์มาเรียกใช้บริการจากด้านแบคก์เอนด์ซึ่งเปรียบเสมือนเป็นเซิร์ฟเวอร์ ดังนั้นเราจึงได้จำลองไคลเอนต์ขึ้นมาโดยเป็นเพียงโปรแกรม EXE ธรรมดา โดยมีจำนวนเท่ากับจำนวนคอมโพเนนต์ด้านแบคก์เอนด์ซึ่งในหนึ่งคอมโพเนนต์จะประกอบด้วยคลาสได้มากกว่า 1 คลาส และสำหรับในส่วนของแบคก์เอนด์จะมีคอมโพเนนต์ดังนี้ โดยในแต่ละคอมโพเนนต์จะมีเป็น 1 DLL

6.9.1 คอมโพเนนต์ร้านค้า

ประกอบด้วยคลาสดังนี้

- Shop
- Category
- Item

6.9.2 คอมโพเนนต์สมาชิกของระบบ

ประกอบด้วยคลาส Account เพียงคลาสเดียว

6.9.3 คอมโพเนนต์การสั่งซื้อสินค้า

ประกอบด้วยคลาสดังนี้

- Billing
- BillingDetail

6.9.4 คอมโพเนนต์กระเป๋าสินค้า

ประกอบด้วยคลาส ShoppingBag เพียงคลาสเดียว

6.9.5 คอมโพเนนต์การส่งสินค้า

ประกอบด้วยคลาส Shipment เพียงคลาสเดียว

วิธีในการรวมคลาสต่างๆ มาเป็นคอมโพเนนต์นั้นมีดังนี้

- คลาสในคอมโพเนนต์มีการทำงานร่วมกัน
- การรวมนั้นต้องไม่ให้เกิดความซับซ้อนมากนัก
- เพื่อใช้ในการแบ่งงานกันทำได้

เราสามารถรวมทุกคลาสที่มีการทำงานด้าน แบคก์เอนด์เข้าเป็นคอมโพเนนต์เดียวเลยก็ได้ ซึ่งทำให้เราได้คอมโพเนนต์ตัวใหญ่หนึ่งตัวซึ่งจะกลายเป็น DLL ตัวใหญ่หนึ่งตัว

เมื่อเราได้คอมโพเนนต์ต่างๆ แล้วต่อไปเราต้องทำการสร้างโปรแกรมมาเพื่อทำการทดสอบคอมโพเนนต์ ซึ่งเราสามารถทำการแปลงโปรแกรมที่ใช้ในการทดสอบนั้นให้เป็นคอมโพเนนต์ในรูปแบบของ ActiveX Document ซึ่งจะกลายเป็นคอมโพเนนต์อีกส่วนหนึ่งของด้านฟรอนท์เอนด์ ซึ่งใช้สำหรับผู้ดูแลระบบนี้ใช้ในการควบคุมดูแลระบบนี้ต่อไปได้

บทที่ 7
ผลการทดสอบระบบ

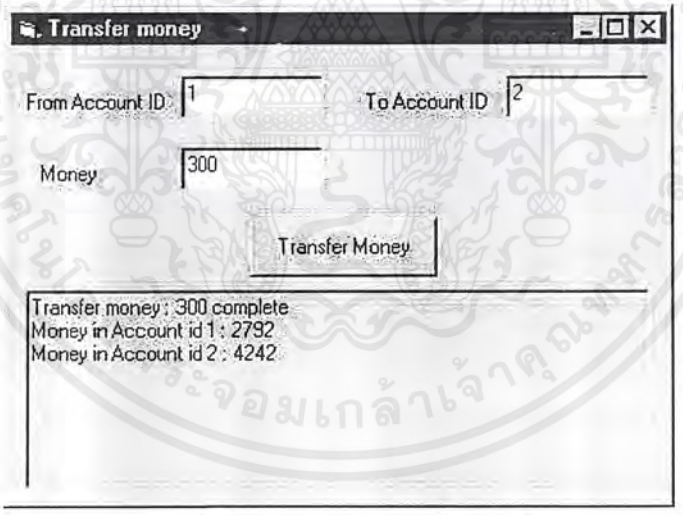
7.1 การทดลอบการจัดการทรานแซคชั่นแบบทูเฟสคอมมิต

เป็นการทดสอบการโรลแบคก์ (Roll Back) ของระบบฐานข้อมูลออราเคิล โดยการควบคุมการโรลแบคก์นี้ถูกควบคุมโดยทรานแซคชั่นเซิร์ฟเวอร์ โดยมีรูปแบบการเขียนโปรแกรมการควบคุมตามที่ได้นำเสนอในบทที่แล้วในหัวข้อการจัดการกับทรานแซคชั่น

โดยการทดสอบนี้เราจำลองขึ้นมาโดยมีโครงสร้างการทำงานภายในในส่วนที่มีการจัดการเรื่องทรานแซคชั่นเหมือนกับระบบที่เราพัฒนาขึ้นมา โดยเหตุผลที่สร้างโปรแกรมอีกตัวขึ้นมาทดสอบนี้เพราะว่าการทดสอบให้ระบบที่พัฒนานั้นเป็นระบบขนาดใหญ่การทำให้เกิดการล่มของระบบเป็นเรื่องที่เสี่ยงพอสมควร

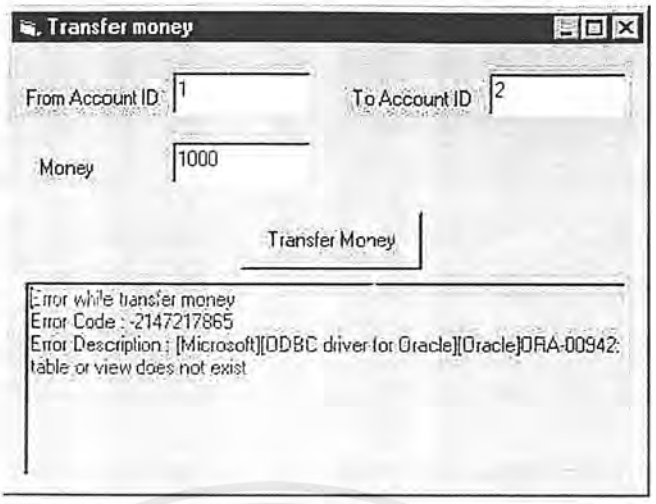
โปรแกรมที่พัฒนาขึ้นมาเป็นโปรแกรมการโอนเงินของธนาคารซึ่ง

- ขั้นตอนแรกเราทำการโอนเงินภายใต้ระบบฐานข้อมูลที่เดียว โดยโอนระหว่างตาราง TblAccount1 กับ TblAccount2 ซึ่งในกรณีนี้คือทรานแซคชั่นเสร็จสมบูรณ์ได้ผลดังนี้



รูปที่ 7-1 แสดงการโอนเงินที่สมบูรณ์

- ทดลองทำให้เกิดความผิดพลาดขึ้นโดยทำการลบตาราง TblAccount2 ออกจากฐานข้อมูล ซึ่งจะมีการทำงานดังนี้ การลดเงินออกจากบัญชีที่ 1 เสร็จสมบูรณ์ แต่การเพิ่มเงินในบัญชีที่ 2 ไม่สามารถทำได้ เป็นผลทำให้ต้องทำการโรลแบคก์ให้ค่าในบัญชีที่ 1 มีค่าดังเดิม ซึ่งการทดสอบได้ผลดังที่คาดไว้คือเงินในบัญชีที่ 1 มีค่าเท่าเดิม



รูปที่ 7-2 แสดงความผิดพลาดที่เกิดในระหว่างการโอนเงิน

- ต่อไปเราทำการทดสอบให้ทั้งสองบัญชีนี้อยู่ต่างระบบฐานข้อมูลกันแต่ยังคงใช้ฐานข้อมูลออราเคิลเหมือนกัน โดยเราเพียงทำการเปลี่ยนที่การติดต่อกับระบบฐานข้อมูลที่ระดับคอมโพเนนต์โดยเราคงใช้โปรแกรมที่ทำการทดสอบตัวเดิมได้ โดยมีเงื่อนไขในการทดสอบดังเดิมคือ ทำการลบบัญชีที่ 2 ในฐานข้อมูลอีกที่หนึ่ง การทำงานที่ได้จะเหมือนเดิมคือทำการลดเงินจากบัญชีที่ 1 ซึ่งสำเร็จและเป็นการคอมมิตภายในฐานข้อมูลนั้น (Local Commit) แต่การเพิ่มเงินในบัญชีที่ 2 นั้นไม่สามารถทำได้จึงไม่คอมมิตภายใน ดังนั้นทำให้ไม่คอมมิตทั้งหมด (Global commit) ผลคือต้องทำการโรลแบคก์ทั้งสองฝั่ง ทำให้เงินในบัญชีที่ 1 ยังคงเท่าเดิมอยู่

- ส่วนการใช้ระบบฐานข้อมูลที่ต่างกันในการทำทูเฟสคอมมิตโดยมีการควบคุมอยู่ที่ทรานแซคชั่นเซิร์ฟเวอร์นั้นไม่ได้ทำการทดลองเพราะไม่มีอุปกรณ์เพียงพอ แต่ทางทฤษฎีแล้วต้องทำได้เพราะระบบฐานข้อมูลที่มีความสามารถในการทำระบบฐานข้อมูลแบบกระจายจะมีมาตรฐานที่ใช้ในการทำทูเฟสคอมมิต และทรานแซคชั่นเซิร์ฟเวอร์จะทำการติดต่อกับฐานข้อมูลโดยใช้มาตรฐานนี้

7.2 การจัดการทรานแซคชั่นกับตารางชนิดเนสเต็ด

การทดสอบเรื่องทรานแซคชั่นด้วยวิธีข้างต้นนั้นมีปัญหากับชนิดตารางแบบเนสเต็ด ประการแรกน่าจะเกิดจากวิชวลเบสิกเองนั้นไม่ได้รองรับกับชนิดของตารางประเภทนี้ตั้งแต่ต้น ยกตัวอย่างเช่นการใช้ ADO ในการติดต่อกับตารางชนิดเนสเต็ดนี้ไม่สามารถทำได้โดยตรงจำเป็นต้องใช้คำสั่ง SQL แบบพิเศษในการทำให้ข้อมูลที่ส่งกลับมาจากฐานข้อมูลออราเคิลนั้นดูเหมือนเป็นข้อมูลที่ได้จากตารางปกติ ซึ่งการใช้คำสั่งสามารถดูได้จากภาคผนวก การใช้งาน SQL ในการติดต่อกับตารางลักษณะเนสเต็ด

การแก้ปัญหาด้วยวิธีนี้ทำให้เราสามารถเรียกดูข้อมูลจากตารางชนิดเนสเต็ดนี้ได้แต่เราไม่สามารถทำการเพิ่ม ลบ หรือเปลี่ยนแปลงข้อมูลโดยใช้เมธอดของ ADO ได้ เช่นเมธอด AddNew, Delete ซึ่งแนว

ทางการแก้ปัญหานี้คือใช้คำสั่ง SQL ในการเพิ่ม ลบ และแก้ไขข้อมูล โดยเราส่งคำสั่ง SQL นี้ในขณะที่ทำการเปิดเรคอร์ดเซต (Recordset.open SQLString, ActiveConnection)

จากที่กล่าวมาทำให้เกิดปัญหาขึ้นมาอีกว่าเมื่อเราใช้ไมโครซอฟท์ทรานแซคชั่นเซิร์ฟเวอร์คอมโพเนนต์ในการจัดการกับทรานแซคชั่นในการใส่ข้อมูลใหม่ลงตารางที่เป็นชนิดเนสเต็ด ซึ่งสามารถจัดการได้โดยคอมโพเนนต์ที่เราสร้างขึ้นเรียกใช้ เมธอดคอมมิตของทรานแซคชั่นคอมโพเนนต์นี้แล้ว และการทำงานที่ได้เรียบร้อยดีไม่มีความผิดพลาดเกิดขึ้น แต่ข้อมูลนั้นไม่ได้ถูกเพิ่มเข้าไปต้องใช้เวลาประมาณ 10 – 20 วินาทีจึงจะถูกเพิ่มเข้าไปและสำหรับในกรณีของการลบข้อมูลออกจากตารางก็เช่นกัน แต่สำหรับการเปลี่ยนแปลงแก้ไขข้อมูลกลับไม่เป็น

แนวทางการแก้ปัญหานี้คือ เรียกใช้การจัดการทรานแซคชั่นของ ADO แทนซึ่งจะอยู่ภายใต้การควบคุมของออบเจ็กต์คอนเนคชั่น (Connection Object) ของ ADO ซึ่งมีการเรียกใช้ดังนี้

- ทำการอ้างถึงคอมโพเนนต์ Microsoft ActiveX Data Objects 2.1 Library เพื่อเรียกใช้ ADO
- โปรแกรมที่ได้มีการทำงานเป็นดังนี้

```
Private Sub TestADOTransaction()
Dim cn As Connection

Set cn = New Connection

On Error GoTo ErrHandler
cn.open              ' Start Connection to Database Server
cn.BeginTrans        ' Start Transaction
' Code for work in transaction

cn.CommitTrans       ' Commit Transaction

Exit Sub

ErrHandler:          ' Error Occur
 cn.RollbackTrans    ' Abort Transaction

End Sub
```

จากการเปลี่ยนมาใช้การจัดการทรานแซคชั่นแบบนี้ทำให้สามารถแก้ปัญหาที่เกิดขึ้นได้ แต่การจัดการทรานแซคชั่นแบบนี้ยังคงมีข้อจำกัดที่ว่าทรานแซคชั่นที่วิธีนี้จะทำได้คือต้องมีการติดต่อกับฐานข้อมูลหรือการเปิดคอนเนคชั่นนั้นเพียงครั้งเดียวใน 1 คอนเนคชั่น โดยในระหว่างที่อยู่ภายในทรานแซคชั่นจะไม่สามารถทำการเปลี่ยนการติดต่อไปยังฐานข้อมูลอื่นได้ ซึ่งการใช้คอนเนคชั่นออบเจ็กต์สองตัวเป็นแนวทางที่น่าจะทำได้แต่เราได้ทำแล้วแต่การวางโครงสร้างของโปรแกรมจะมีความซับซ้อนมากจึงขอแนะนำว่าไม่ควรใช้วิธีนี้

7.3 การทดสอบตามการทำงานพื้นฐาน

การทดสอบนี้จะทดสอบเพียงการทำงานพื้นฐานหรือเป็นการทดสอบการทำงานของเมธอดของแต่ละคลาสว่าตรงตามความต้องการในเบื้องต้น และมีการทำงานที่ถูกต้องหรือไม่อย่างไร

สำหรับการทดสอบการทำงานรวมของระบบต้องใช้คอมโพเนนต์ด้านของฟรอนท์เอนด์ประกอบการทดสอบด้วย ซึ่งการทดสอบรวมทั้งระบบนี้จะอยู่ในส่วนของการทดสอบระบบของด้านฟรอนท์เอนด์ ซึ่งในปริญญานิพนธ์เล่มนี้จะไม่กล่าวถึง

7.3.1 ทำการทดสอบโปรแกรมที่ใช้ในการทดสอบ

โปรแกรมที่ใช้ในการทดสอบคอมโพเนนต์ต่างๆ ด้านแบคก์เอนด์นั้นจำเป็นที่ต้องทำการตรวจสอบความถูกต้องก่อนที่จะนำไปใช้ทดสอบคอมโพเนนต์ ซึ่งเราแบ่งการทดสอบออกเป็นขั้นตอนต่างๆ ดังนี้

7.3.1.1 ทดสอบอินพุทที่รับเข้ามาจากผู้ใช้

ในที่นี้คือผู้พัฒนาระบบเพราะว่าโปรแกรมที่ใช้ทดสอบนี้ไม่ได้เป็นโปรแกรมที่จะนำไปใช้จริง โดยรูปแบบของการตรวจสอบจะเน้นไปที่การจัดการกับชนิดของข้อมูลว่ามีชนิดตรงกับชนิดข้อมูลที่เราต้องการหรือไม่ เช่น การรับข้อความและเปลี่ยนให้เป็นตัวเลข หรือการเปลี่ยนตัวเลขจำนวนเต็มเป็นตัวเลขจำนวนจริง

7.3.1.2 ทดสอบอินพุทที่ส่งเข้าคอมโพเนนต์

ส่วนใหญ่ปัญหาที่พบส่วนใหญ่จะเกิดความผิดพลาดที่ชนิดของข้อมูลนั้นไม่ตรงกับที่ชนิดข้อมูลที่คอมโพเนนต์ต้องการ และปัญหาอีกประการคือใส่ข้อมูลไม่ครบตามที่คอมโพเนนต์ต้องการ

7.3.1.3 ทดสอบเอาท์พุทที่ได้ออกมาจากคอมโพเนนต์

การนำเอาท์พุทจากคอมโพเนนต์มาใช้นั้นเราต้องระบุให้ได้ว่า ข้อมูลอกมาจากคอมโพเนนต์นั้นส่งแบบ ByRef หรือ ByVal โดยการส่งค่าข้อมูลที่เป็นออบเจ็กต์ออกมาโดยใช้คำสั่ง Set นั้นจะเป็นการผ่านแบบ ByRef และการตรวจสอบชนิดของข้อมูลที่ส่งออกมานั้นมีความสำคัญในขั้นตอนนี้ด้วย

7.3.1.4 ทดสอบเอาท์พุทที่ส่งออกไปให้กับผู้ใช้

เนื่องจากการทดสอบการทำงานของคอมโพเนนต์นั้นเราไม่สามารถรู้ถึงการทำงานภายในคอมโพเนนต์ได้ดังนั้นผลที่ได้จากโปรแกรมทดสอบนี้เป็นสิ่งที่สำคัญ ซึ่งการส่งเอาท์พุทออกมาใช้ผู้ใช้นี้ส่วนใหญ่ไม่ค่อยมีปัญหาเพราะชนิดข้อมูลที่เป็นข้อความนั้นมีความครอบคลุมถึงชนิดข้อมูลที่เป็นตัวเลขด้วย ซึ่งปัญหาที่เกิดขึ้นส่วนใหญ่จะเกิดกับชนิดข้อมูลที่เป็นเรคอร์ดเซต เช่น ในเรคอร์ดเซตไม่มีสดมภ์ของข้อมูลที่ต้องการ หรือในเรคอร์ดเซตไม่มีข้อมูลอยู่ ซึ่งต้องทำการตรวจสอบความถูกต้องของข้อมูลก่อนนำข้อมูลนั้นไปใช้ด้วย

7.3.2 ทำการทดสอบคอมโพเนนต์ด้านแบคก์เอนด์

เมื่อเราแน่ใจว่าโปรแกรมที่เราใช้ในการทดสอบมีความถูกต้องแล้ว ขั้นต่อไปเราจะทำการทดสอบคอมโพเนนต์บ้าง โดยเราแบ่งเป็นขั้นตอนต่างๆ ดังนี้

7.3.2.1 การทดสอบอินพุทที่เข้าคอมโพเนนต์

อินพุทที่เข้าสู่คอมโพเนนต์นั้นจะอยู่ในรูปของแอททริบิวต์และพารามิเตอร์ที่ป้อนเข้าในบางเมธอดซึ่งปัญหาส่วนใหญ่ที่เกิดขึ้นคือ ชนิดของข้อมูลไม่ตรงกับที่ระบุไว้ หรือจำนวนข้อมูลไม่ครบ หรือทำการกำหนดรูปแบบของข้อมูลไม่ตรงกัน ซึ่งส่วนใหญ่จะเป็นปัญหากับข้อมูลชนิดเรคอร์ดเซต

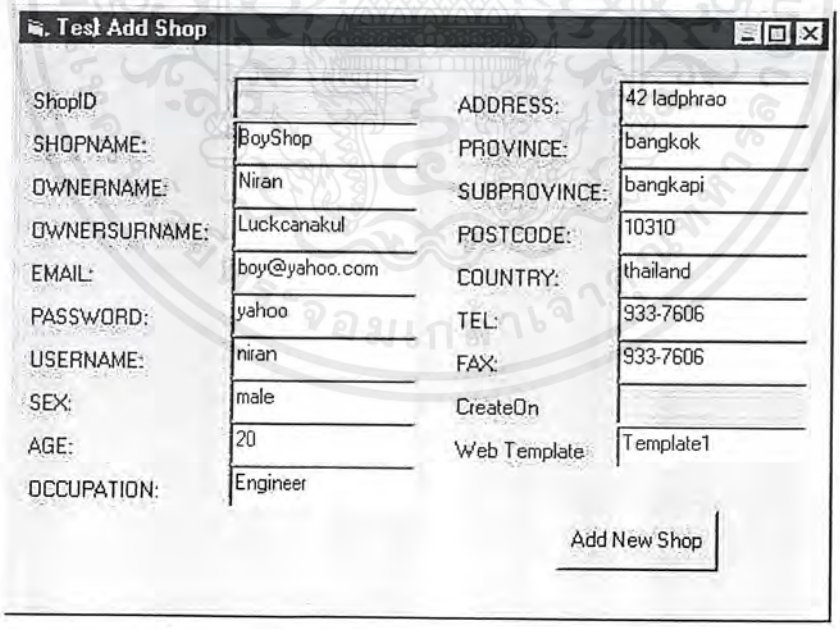
7.3.2.2 การทดสอบเอาท์พุทที่ส่งออกจากคอมโพเนนต์

เป็นการทดสอบข้อมูลที่ถูกส่งออกมาจากคอมโพเนนต์ว่ามีความถูกต้องตามที่ระบุไว้หรือไม่

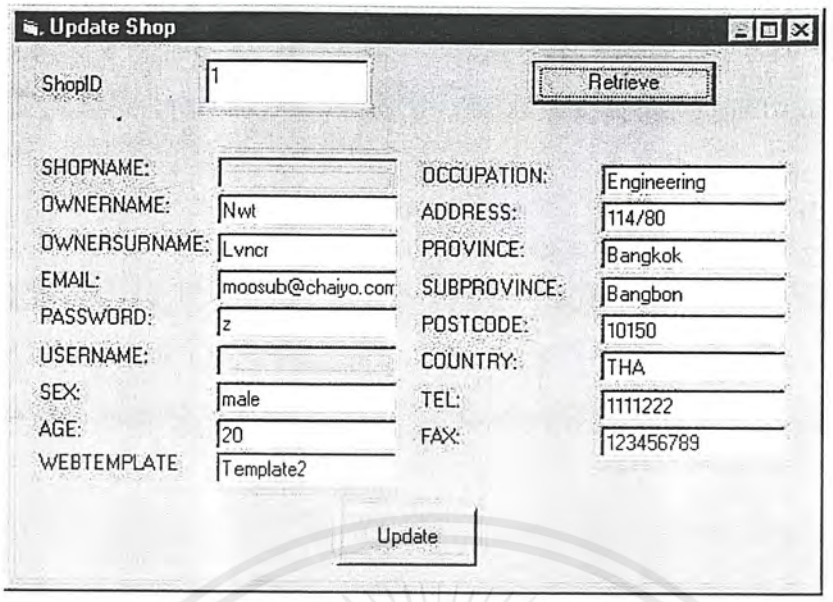
7.3.3 ผลของโปรแกรมที่ใช้ในการทดสอบ

แบ่งผลของการทดสอบนี้ออกตามเมธอดของแต่ละคลาสในการทำงานของด้านแบคก์เอนด์ แบ่งเป็นคลาสต่างๆ ได้ดังนี้

7.3.3.1 คลาส SHOP

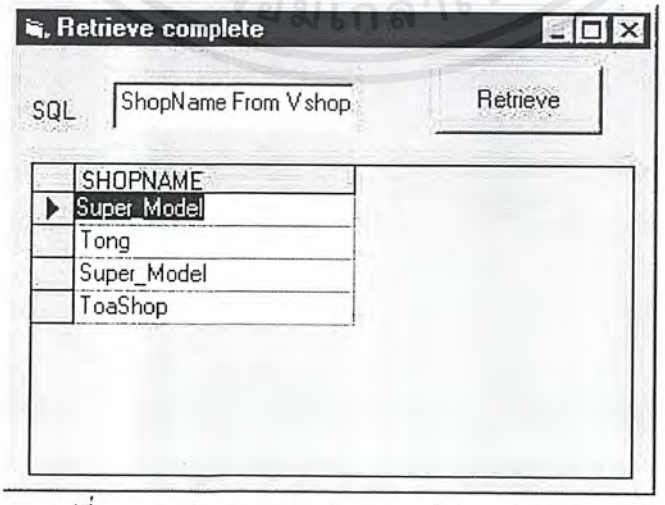


รูปที่ 7-3 การทำงานของเมธอด AddNew() ในคลาส Shop



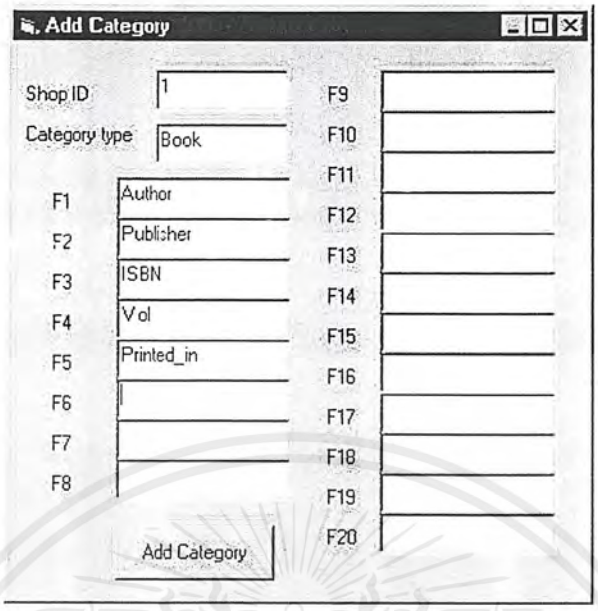
รูปที่ 7-4 การทำงานของเมธอด Update() ในคลาส Shop

รูปที่ 7-5 การทำงานของเมธอด Delete() ในคลาส Shop

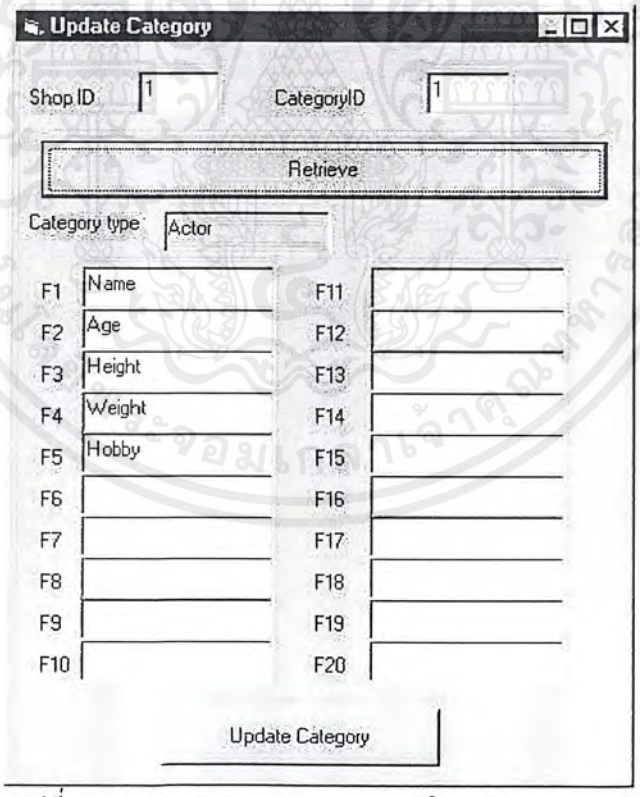


รูปที่ 7-6 การทำงานของเมธอด Find() ในคลาส Shop

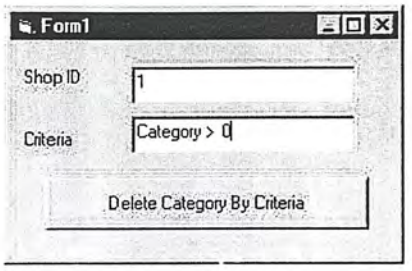
7.3.3.2 คลาส CATEGORY



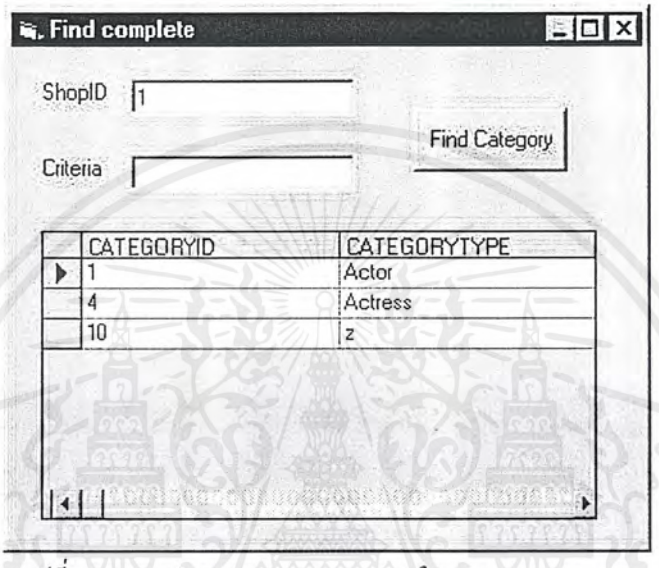
รูปที่ 7-7 การทำงานของเมธอด Addnew() ในคลาส Category



รูปที่ 7-8 การทำงานของเมธอด Update() ในคลาส Category

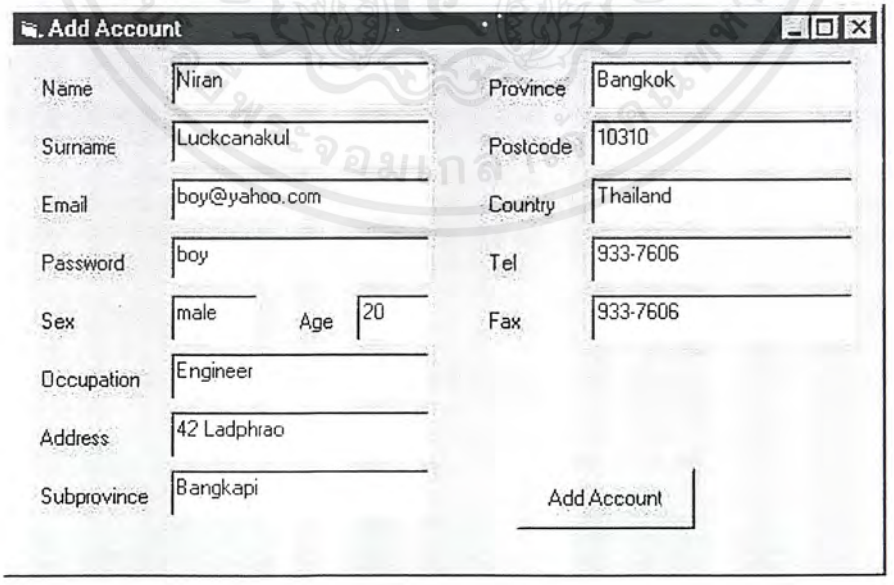


รูปที่ 7-9 การทำงานของเมธอด Delete() ในคลาส Category

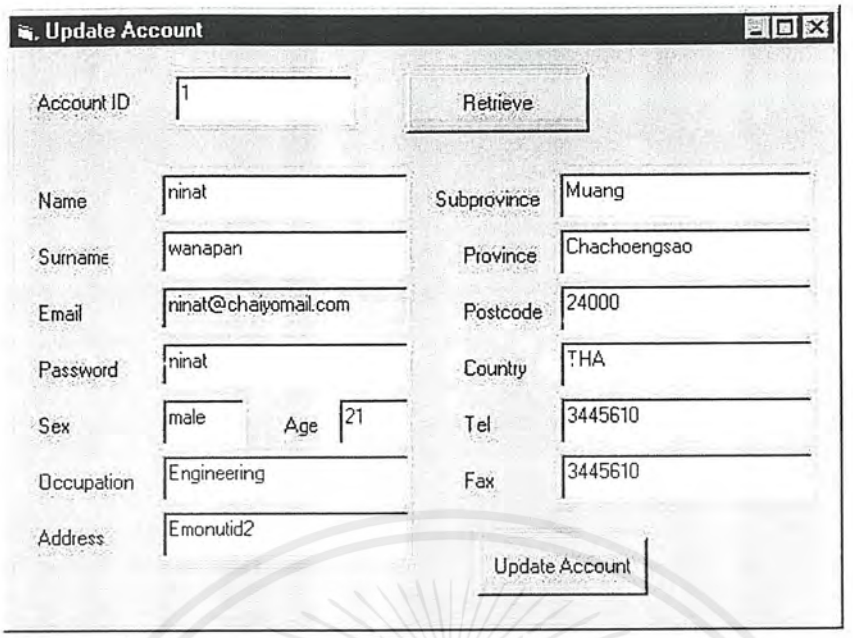


รูปที่ 7-10 การทำงานของเมธอด Find() ในคลาส Category

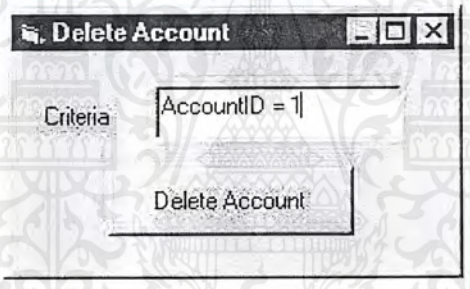
7.3.3.3 คลาส ACCOUNT



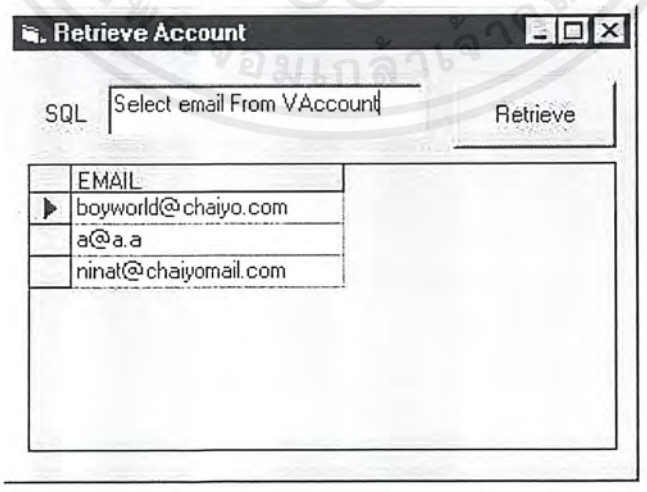
รูปที่ 7-11 การทำงานของเมธอด AddAccount() ในคลาส Account



รูปที่ 7-12 การทำงานของเมธอด UpdateAccount() ในคลาส Account

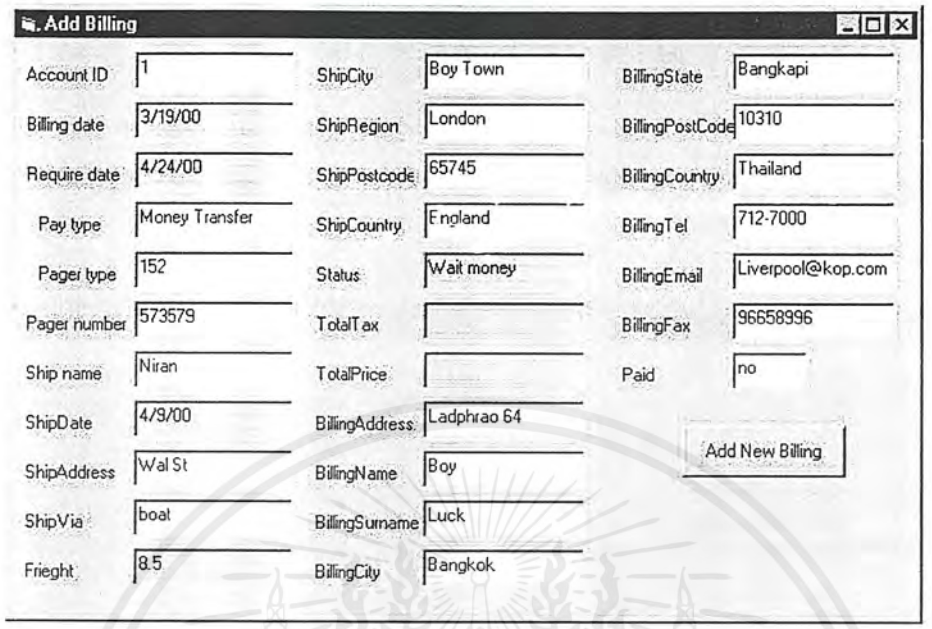


รูปที่ 7-13 การทำงานของเมธอด DeleteAccount() ในคลาส Account

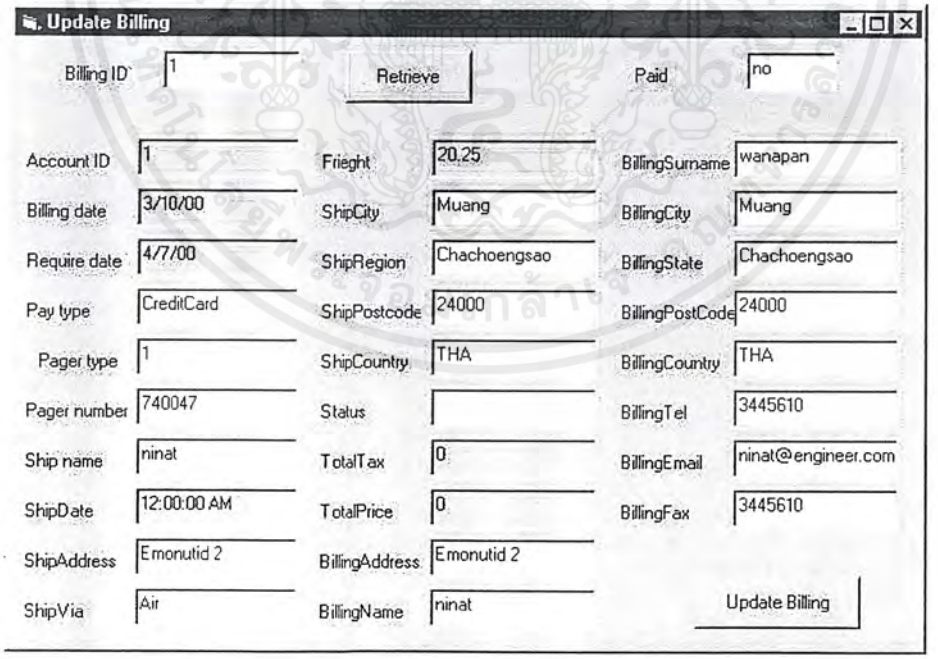


รูปที่ 7-14 การทำงานของเมธอด GetInformation() ในคลาส Account

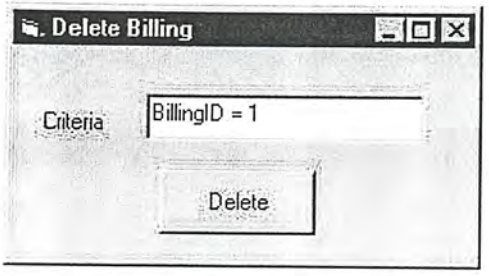
7.3.3.4 คลาส BILLING



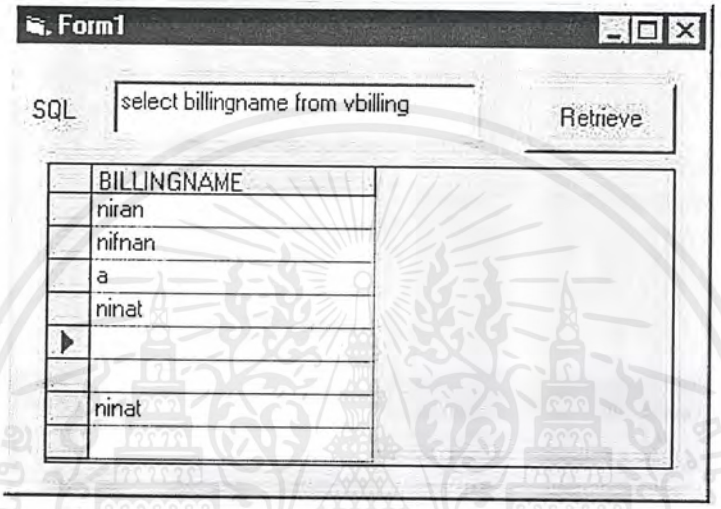
รูปที่ 7-15 การทำงานของเมธอด AddBilling() ในคลาส Billing



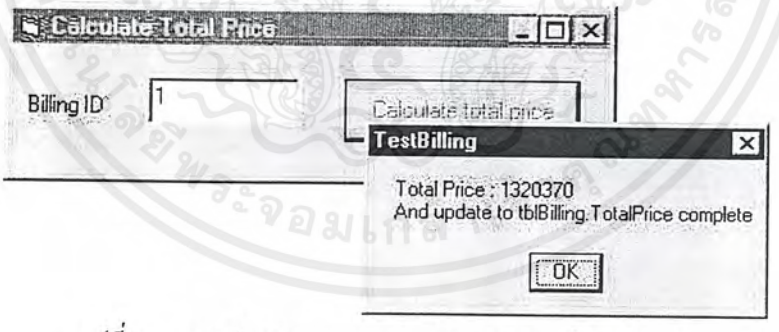
รูปที่ 7-16 การทำงานของเมธอด UpdateBilling() ในคลาส Billing



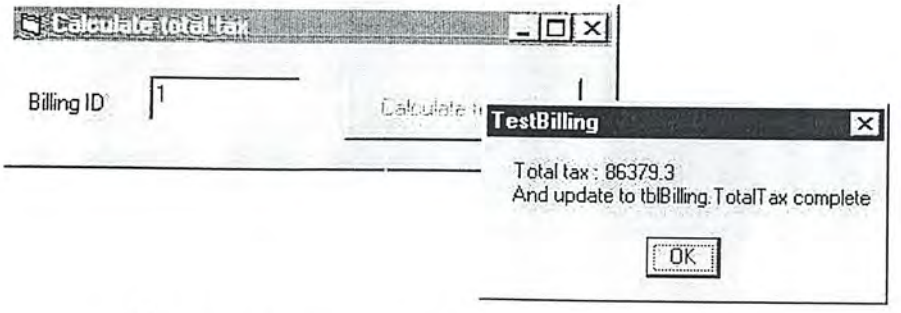
รูปที่ 7-17 การทำงานของเมธอด DeleteBilling() ในคลาส Billing



รูปที่ 7-18 การทำงานของเมธอด FindBilling() ในคลาส Billing

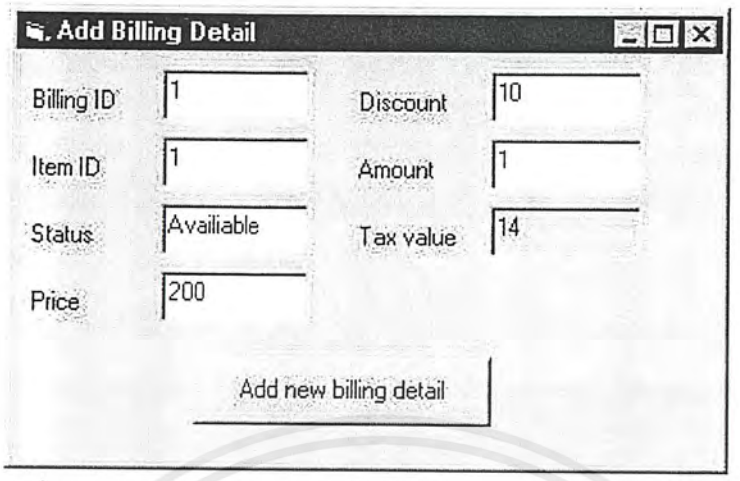


รูปที่ 7-19 การทำงานของเมธอด CalTotalPrice() ในคลาส Billing

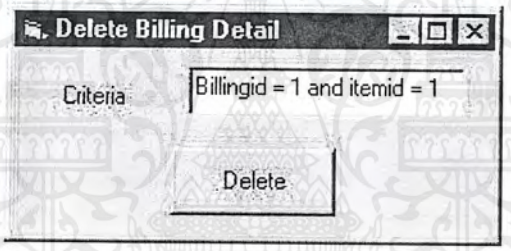


รูปที่ 7-20 การทำงานของเมธอด CalTotalTax() ในคลาส Billing

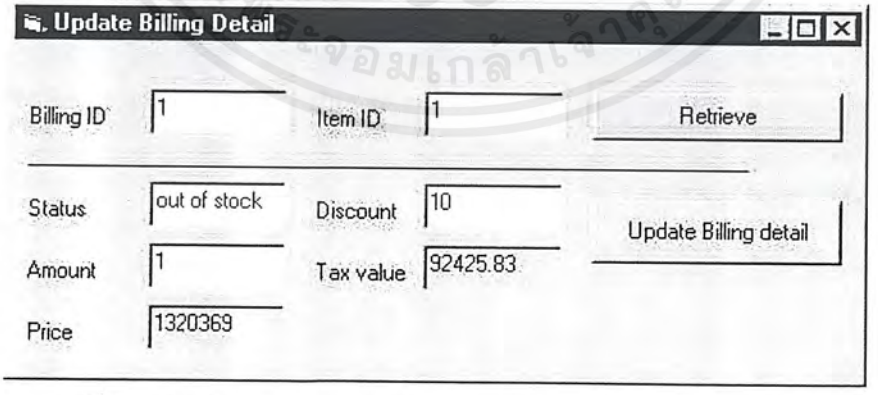
7.3.3.5 คลาส BILLINGDETAIL



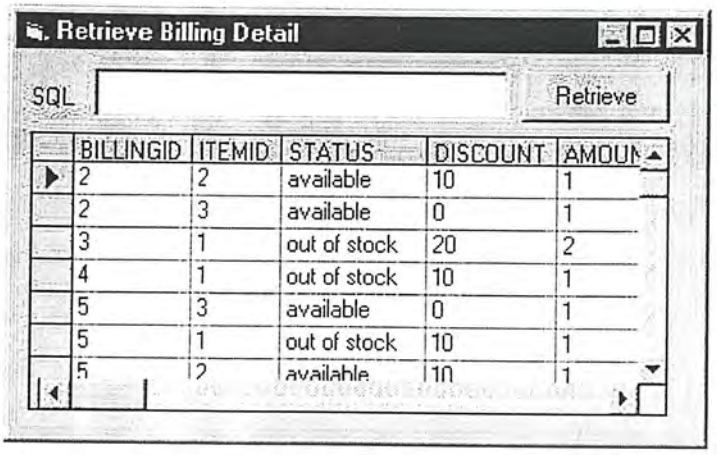
รูปที่ 7-21 การทำงานของเมธอด AddBillingDetail() ในคลาส BillingDetail



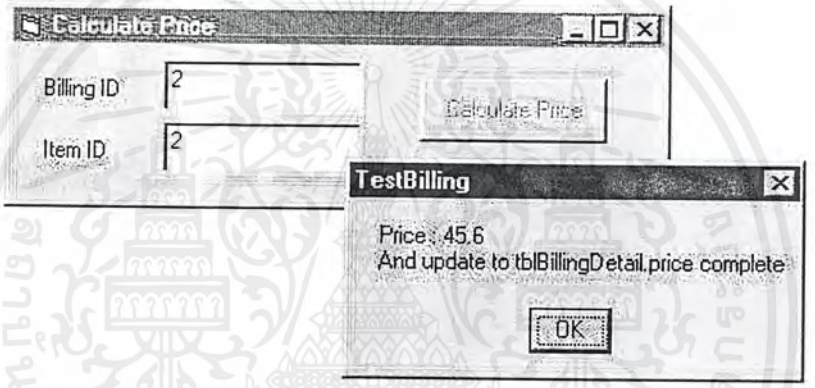
รูปที่ 7-22 การทำงานของเมธอด DeleteBillingDetail() ในคลาส BillingDetail



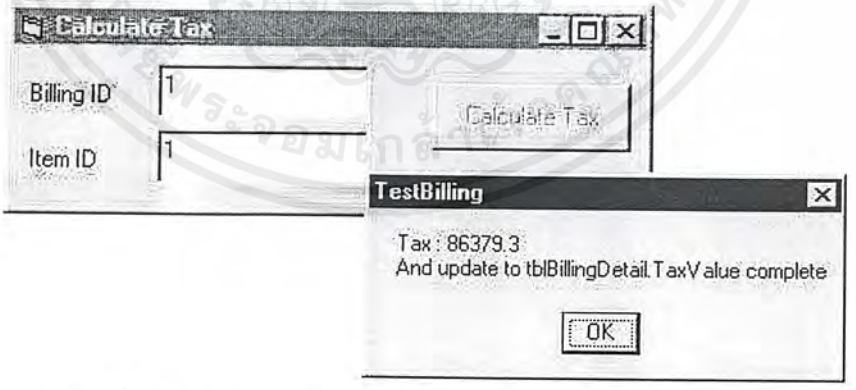
รูปที่ 7-23 การทำงานของเมธอด UpdateBillingDetail() ในคลาส BillingDetail



รูปที่ 7-24 การทำงานของเมธอด FindBillingDetail() ในคลาส BillingDetail



รูปที่ 7-25 การทำงานของเมธอด CalPrice() ในคลาส BillingDetail



รูปที่ 7-26 การทำงานของเมธอด CalTax() ในคลาส BillingDetail

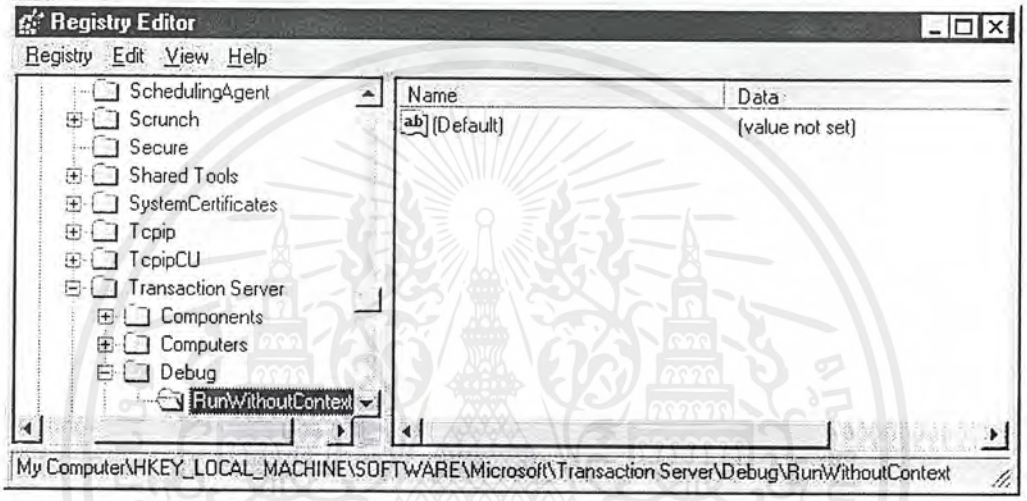
คลาสนอกเหนือจากนี้เป็นคลาสที่มีการทำงานร่วมกับการทำงานด้านฟรอนท์เอนด์ซึ่งไม่จำเป็นต้องสร้างโปรแกรมขึ้นมาทดสอบสามารถทำการทดสอบร่วมกับคอมโพเนนต์ของฟรอนท์เอนด์ได้ ซึ่งจะอยู่ในส่วนของการทดสอบคอมโพเนนต์ด้านฟรอนท์เอนด์

7.4 การดีบักคอมโพเนนต์ที่อยู่ในไมโครซอฟท์ทรานแซคชั่นเซิร์ฟเวอร์

มีขั้นตอนการทำงานดังนี้

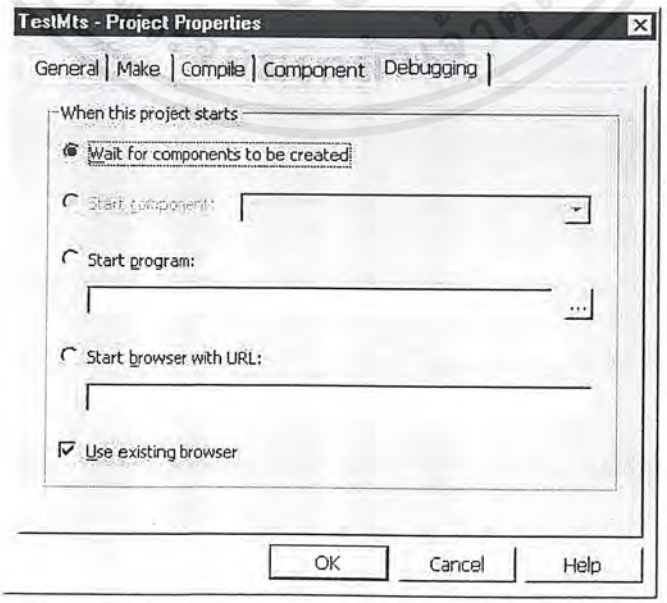
- ทำการเพิ่ม Register โดยเพิ่มคีย์เข้าไปดังนี้

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Transaction Server\debug\ _
RunWithOutContext
```



รูปที่ 7-27 การเพิ่มคีย์ใน Register เพื่อให้สามารถทำการดีบักคอมโพเนนต์ใน MTS ได้

- กำหนดค่าคุณสมบัติของคอมโพเนนต์ที่เรียกใช้ MTS ในส่วนของ Debugging ให้เลือก Wait for components to be created



รูปที่ 7-28 การกำหนดคุณสมบัติของคอมโพเนนต์ที่ต้องการใช้ MTS

- กำหนด Break Point ในคอมโพเนนต์ที่ต้องการดีบัก และในส่วนที่ต้องการเริ่มทำการดีบัก
- ใช้ไคลเอนต์เรียกใช้คอมโพเนนต์ซึ่งจะเข้าสู่การดีบักได้

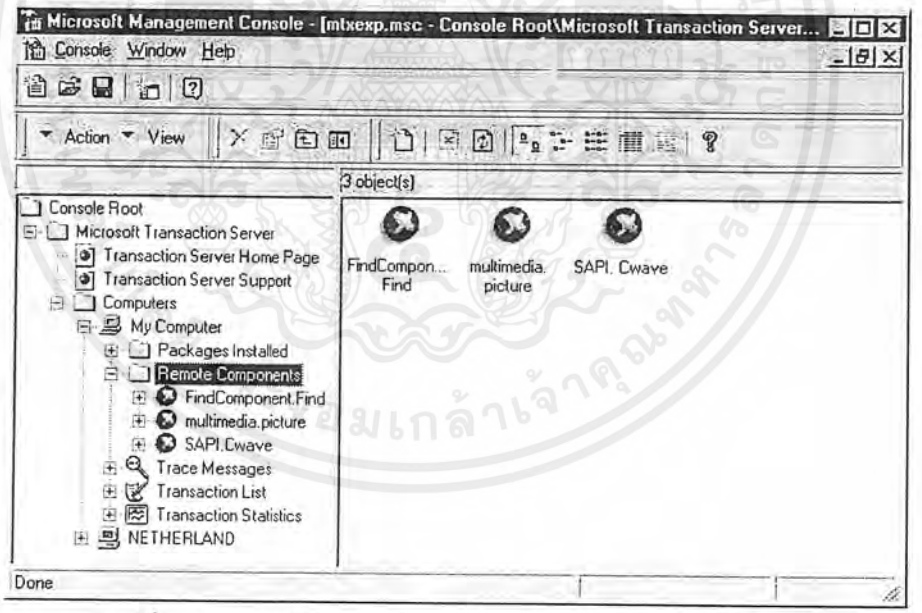
7.5 ปัญหาที่เกิดกับการใช้ไมโครซอฟท์ทรานแซคชั่นเซิร์ฟเวอร์

โดยปัญหาที่จะอธิบายในหัวข้อนี้จะไม่ใช่ส่วนของการจัดการทรานแซคชั่นแต่จะเป็นปัญหาทั่วไปที่เกิดขึ้นในระหว่างการพัฒนาโปรแกรม ซึ่งการใช้งานทรานแซคชั่นเซิร์ฟเวอร์นี้

7.5.1 ปัญหาไม่สามารถเรียกใช้คอมโพเนนต์จากเซิร์ฟเวอร์ได้

เราใช้ความสามารถของไมโครซอฟท์ทรานแซคชั่นเซิร์ฟเวอร์ในการสร้างการทำงานของ COM ให้สามารถทำงานแบบระยะไกลได้โดยเรามีวิธีการใช้งานนี้ 2 รูปแบบ

- การเอ็กซ์พอร์ท (Export) คอมโพเนนต์ที่เราทำการติดตั้งบนทรานแซคชั่นเซิร์ฟเวอร์ เพื่อไปติดตั้งที่เครื่องไคลเอนต์เพื่อทำให้เกิดการทำงานแบบระยะไกลได้
- การใช้ทรานแซคชั่นเซิร์ฟเวอร์แชร์คอมโพเนนต์ในลักษณะรีโมตคอมโพเนนต์โดยวิธีการนี้เครื่องไคลเอนต์ต้องทำการติดตั้งทรานแซคชั่นเซิร์ฟเวอร์ด้วย ซึ่งต้องทำการ Trust Relationships ระหว่างเครื่องไมโครซอฟท์วินโดว์เอ็นทีเซิร์ฟเวอร์ด้วย



รูปที่ 7-29 การแชร์รีโมตคอมโพเนนต์ระหว่างทรานแซคชั่นเซิร์ฟเวอร์

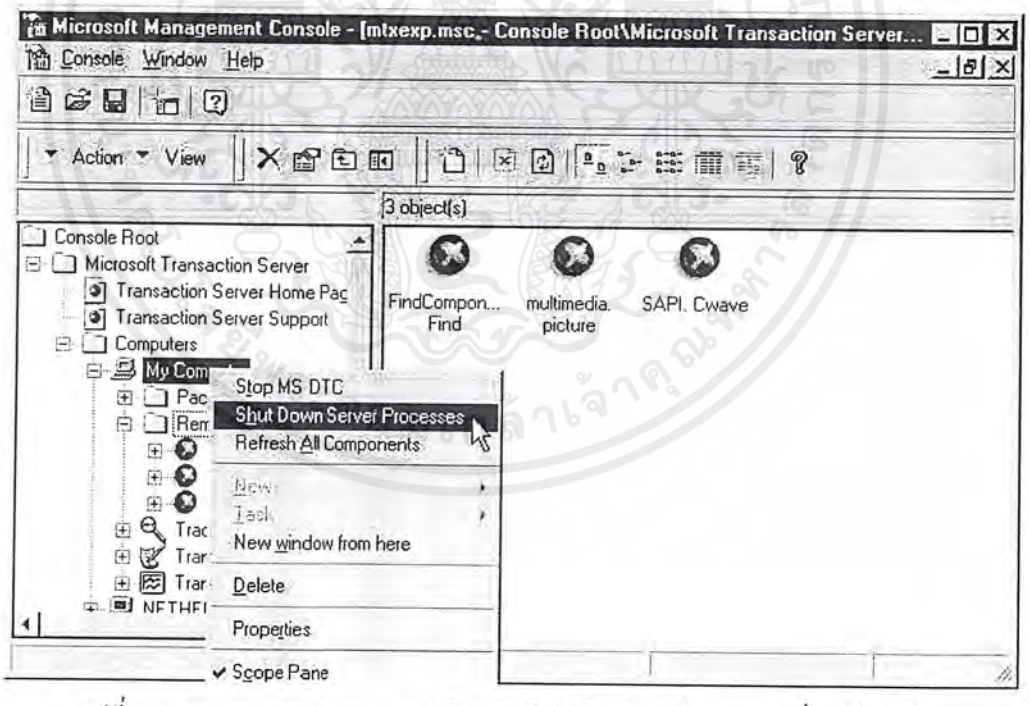
โดยทั้งสองรูปแบบนี้จำเป็นต้องมีคอมโพเนนต์ที่อยู่ในรูปของ DLL ไปติดตั้งที่เครื่องไคลเอนต์ ซึ่งปัญหาเกิดขึ้นเมื่อเราทำการพัฒนาคอมโพเนนต์แล้วเราสร้างไฟล์ DLL ขึ้นมาใหม่ถึงแม้ว่าอินเตอร์เฟสของคอมโพเนนต์จะไม่มีการเปลี่ยนแปลงแต่ยังเกิดปัญหาขึ้นได้ ซึ่งส่วนใหญ่จะขึ้นว่า Permission Denined ซึ่งมีวิธีการแก้ไขดังนี้

- ทำการรีเฟรช (Refresh) คอมโพเนนต์ทั้งเครื่องที่เป็นเจ้าของคอมโพเนนต์และเครื่องที่ขอใช้คอมโพเนนต์นั้นแบบรีโมท
- ถ้ายังไม่ได้ผลให้ทำการลบคอมโพเนนต์ที่เครื่องขอใช้คอมโพเนนต์นั้นแบบรีโมทและทำการใส่คอมโพเนนต์นั้นแบบรีโมทใหม่ โดยก่อนที่จะทำการเพิ่มคอมโพเนนต์ใหม่ที่เป็นแบบรีโมทนั้น เครื่องที่เป็นเจ้าของคอมโพเนนต์ควรทำการรีเฟรชคอมโพเนนต์ที่จะให้เครื่องอื่นเรียกใช้แบบรีโมทก่อนด้วย

ซึ่งการรีเฟรชนี้เป็นการเรียกดูตำแหน่งที่อยู่ของไฟล์คอมโพเนนต์ในรูปแบบของไฟล์ DLL ที่ถูกเก็บไว้ในรีจิสเตอร์โดยค่านี้จะเกิดการเปลี่ยนแปลงเมื่อเราทำการสร้าง DLL ขึ้นมาใหม่โดยอาจทับกับตัวเดิมที่มีอยู่หรือไม่ก็ตาม

7.5.2 ปัญหาโพรเซสของไมโครซอฟท์ทรานแซคชั่นเซิร์ฟเวอร์ค้างเมื่อเกิดความผิดพลาด

ซึ่งเมื่อเกิดความผิดพลาดขึ้นจะทำให้ไมโครซอฟท์ทรานแซคชั่นเซิร์ฟเวอร์ไม่ยอมให้มีการเปลี่ยนแปลงไฟล์ DLL คือมีโพรเซสชื่อ Mtx ค้างอยู่ทำให้ไม่สามารถทำการลบไฟล์ DLL นี้เพื่อสร้างเป็น DLL ใหม่ได้ทางแก้ไขคือทำการหยุดการทำงานของของทุกโพรเซสในไมโครซอฟท์ทรานแซคชั่นเซิร์ฟเวอร์



รูปที่ 7-30 การหยุดการทำงานของทุกโพรเซสในไมโครซอฟท์ทรานแซคชั่นเซิร์ฟเวอร์

บทที่ 8
บทวิจารณ์และสรุป

8.1 การเปรียบเทียบทฤษฎีกับผลการทำงาน

ทฤษฎีที่นำมาใช้ในการทำปริญญานิพนธ์นี้คือการโปรแกรมเชิงวัตถุ การจัดการกับทรานแซคชั่น การใช้คำสั่ง SQL ในการเรียกข้อมูลและใช้ในการทำงานกับระบบฐานข้อมูล โดยทฤษฎีต่างๆที่ได้ทำการศึกษามานี้ได้ถูกนำมาใช้ในการสร้างระบบการพาณิชย์เชิงอิเล็คทรอนิคนี้ขึ้นมา โดยโครงสร้างภายในของระบบนี้เป็นรูปแบบของเทคโนโลยีคอมโพเนนต์ซึ่งมีพื้นฐานมาจากออบเจ็กต์โอเรียนเท็ด

ส่วนเรื่องของการจัดการทรานแซคชั่นนั้นเราจะต้องใช้ความรู้พื้นฐานเรื่องความหมายและหลักการของการใช้ทรานแซคชั่นก่อนเราจึงสามารถเขียนโปรแกรมเพื่อทำการควบคุมและใช้งานได้อย่างถูกต้อง ซึ่งเราได้นำความรู้นี้เป็นพื้นฐานในการใช้ไมโครซอฟท์ทรานแซคชั่นเซิร์ฟเวอร์ด้วย

ส่วนความรู้ในเรื่องของระบบฐานข้อมูลนั้นมีความจำเป็นมากเพราะว่าระบบฐานข้อมูลที่ใช้นี้เป็นระบบฐานข้อมูลเชิงวัตถุสัมพันธ์ ซึ่งถ้าไม่มีความรู้เกี่ยวกับระบบฐานข้อมูลเชิงสัมพันธ์มาก่อนนั้นจะทำความเข้าใจกับระบบฐานข้อมูลเชิงวัตถุสัมพันธ์นี้ได้ยากมาก

สำหรับความรู้ทางด้านอื่นๆ เช่นการเขียนโปรแกรมโดยใช้ภาษาเบสิก PL/SQL และการใช้ทูลต่างๆ ออราเคิลมีให้มา นั้นเป็นส่วนย่อยที่สามารถทำความเข้าใจได้ ซึ่งเราไม่ควรใช้เวลามากเพราะเราควรมองว่าอะไรที่เป็นความรู้ที่จำเป็น และอะไรเป็นสิ่งที่ต้องรู้

8.2 การทำงานที่ได้กับวัตถุประสงค์ที่ตั้งไว้ก่อนทำโครงงานนี้

โดยวัตถุประสงค์ที่วางไว้ในขั้นตอนแรกของการทำโครงงานนี้คือ ทำการสร้างแอบพลิเคชันที่มีลักษณะการทำงานแบบไคลเอนต์เซิร์ฟเวอร์ที่มีสถาปัตยกรรมแบบสามเทียร์ ซึ่งเลือกใช้ทูลในการเขียนโปรแกรมคือวิชวลเบสิก 6.0 ซึ่งทางผู้พัฒนาโปรแกรมวิชวลเบสิกนี้ได้วางโครงสร้างของวิชวลเบสิกให้สร้างการทำงานแบบคอมโพเนนต์ และใช้ไมโครซอฟท์ทรานแซคชั่นในการสร้างแอบพลิเคชันไคลเอนต์เซิร์ฟเวอร์แบบสามเทียรนี้ขึ้นมาได้

ผลที่ใช้ในการวัดว่าเราได้มีการทำงานตรงตามวัตถุประสงค์ที่วางไว้หรือไม่คือผลงานที่ได้ออกมาซึ่งคือ ระบบพาณิชย์เชิงอิเล็กทรอนิกส์นี้ ซึ่งโครงงานนี้ได้ครอบคลุมในส่วนของการทำงานระดับล่างทั้งหมดของระบบหรือคือการทำงานด้านแบคก์เอนด์ ซึ่งคำจำกัดความของการทำงานด้านแบคก์เอนด์นี้คือเป็นการทำงานที่ทำการติดต่อกับระบบฐานข้อมูลโดยตรง และเป็นส่วนที่สนับสนุนให้เกิดการทำงานในระดับหน้าหรือในระดับฟรอนท์เอนด์ด้วย

จากการดำเนินงานและได้ผลงานออกมานี้เราคิดว่าเราได้ทำงานตามวัตถุประสงค์ที่เราได้วางไว้ตั้งแต่ต้นแล้ว

8.3 แนวทางการพัฒนาต่อ

แนวทางการพัฒนาต่อนี้เราสามารถมองได้หลายรูปแบบ คือเราอาจนำระบบที่ได้นี้ไปทำการพัฒนาต่อทั้งระบบ หรือเราอาจนำส่วนที่มีการทำงานด้านฟรอนท์เอนด์หรือแบคก์เอนด์เพียงส่วนเดียวไปใช้แล้วพัฒนาส่วนที่เราไม่ได้นำไปให้มีประสิทธิภาพและความสามารถที่มากกว่าเดิมขึ้นมา หรือเราอาจนำแนวความคิดในการพัฒนาระบบในรูปแบบนี้ขึ้นมาใหม่ เพราะว่าระบบนี้อาจไม่มีความสามารถหรือความเสถียรพอที่จะทำเป็นระบบในเชิงธุรกิจได้เพราะว่าเราไม่ได้คำนึงถึงประสิทธิภาพในการพัฒนา แต่เราเน้นการเชื่อมต่อเทคโนโลยีและความรู้พื้นฐานที่ใช้ในการประยุกต์เทคโนโลยีต่างๆที่เราสนใจ

โดยหลักการของคอมโพเนนต์นั้นมีหลักการเดียวกับออบเจ็กต์โอเรียนเท็ด คือทำเพื่อเน้นการนำกลับมาใช้และการพัฒนาต่อ โดยเราสามารถนำหลักการนี้ไปใช้ในทางปฏิบัติได้แต่หลักการนี้มีข้อแม้อยู่ว่าคอมโพเนนต์ที่นำไปใช้นั้นต้องมีความถูกต้องมากและมีความหลากหลายในการใช้งาน แต่คอมโพเนนต์ที่ทำการพัฒนานี้อาจมีคุณสมบัติไม่ครบถ้วนตามที่ได้กล่าวมา

สำหรับแนวทางการนำทั้งระบบไปพัฒนาต่อนั้น สำหรับสิ่งที่ระบบยังขาดอยู่คือ

- การจ่ายเงินที่ต้องทำการติดต่อกับระบบธนาคารเพื่อใช้ในการตรวจสอบบัตรเครดิต
- การทำส่วนของการส่งสินค้าที่ควรพัฒนาให้มีบริษัทที่ให้บริการในการส่งที่มากขึ้นพร้อมกับการกำหนดข้อบังคับต่างๆ กับบริษัทที่ให้บริการด้วย
- ส่วนของการยืนยันการสั่งซื้อสินค้าที่เป็นแบบเรียลไทม์คือสามารถทำการส่งข้อมูลยืนยันการสั่งซื้อผ่านทางเพจเจอร์หรือข้อความผ่านบริษัทมือถือ ที่ในระบบปัจจุบันยังทำได้ไม่ดีนักคือเราควรทำการติดต่อผ่าน Socket แต่ที่เป็นอยู่เราใช้การจำลองการส่งเพจทาง HTML ผ่านทาง cgi แทน

สำหรับแนวทางการนำส่วนของแบคก์เอนด์ไปพัฒนาต่อ

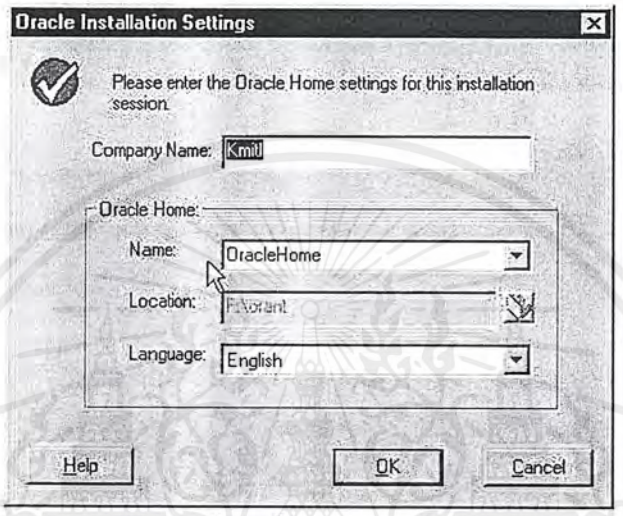
- การเพิ่มข้อบังคับในแอบพลิเคชันเซิร์ฟเวอร์เพื่อสร้างความสมบูรณ์ยิ่งขึ้นให้กับระบบ โดยข้อบังคับในแอบพลิเคชันเซิร์ฟเวอร์ที่มีอยู่เดิมนั้นสามารถดูได้ในบทของการพัฒนาระบบพาณิชย์เชิงอิเล็กทรอนิกส์

สำหรับแนวทางการนำส่วนของ ฟรอนท์เอนด์ไปพัฒนาต่อนั้นสามารถศึกษาได้จากปริญญานิพนธ์อีกเล่มซึ่งมีชื่อว่าการพัฒนาซอฟท์แวร์เชิงคอมโพเนนต์ ซึ่งครอบคลุมถึงการทำงานทั้งหมดของด้านฟรอนท์เอนด์

ภาคผนวก ก.
การติดตั้งระบบจัดการฐานข้อมูลออราเคิล 8

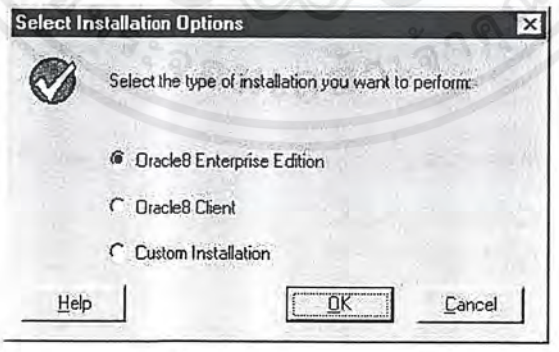
ก.1 Install Oracle 8 Enterprise Edition

1 ทำการเลือกไดเรคทอรีและ ชื่อ Oracle home



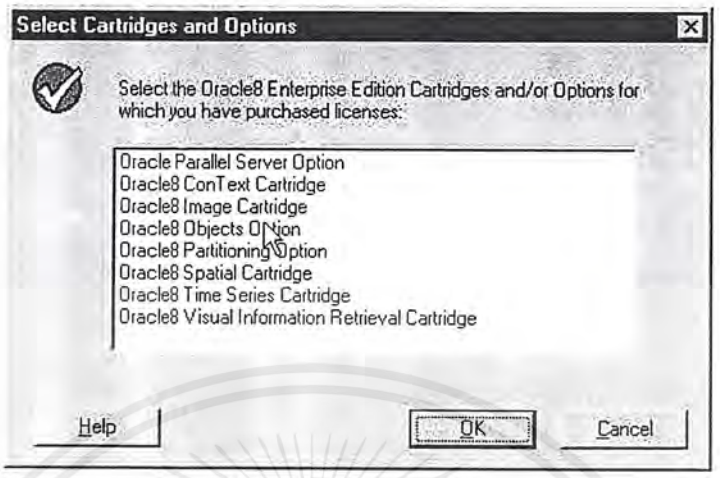
รูปที่ ก-1 ขั้นตอนการลงที่ 1

2 ทำการเลือกรูปแบบที่จะลงซึ่งเลือกแบบในการติดตั้งระบบออราเคิลโดยถ้าเป็นเครื่องที่ทำหน้าที่เป็นเซิร์ฟเวอร์สามารถทำการติดตั้งเป็น Enterprise Edition หรือ Client ก็ได้ส่วนหัวข้อการเลือกรูปแบบการติดตั้งเองนั้นควรศึกษาจากข้อมูลที่ระบบออราเคิลมีมาให้



รูปที่ ก-2 ขั้นตอนการลงที่ 2

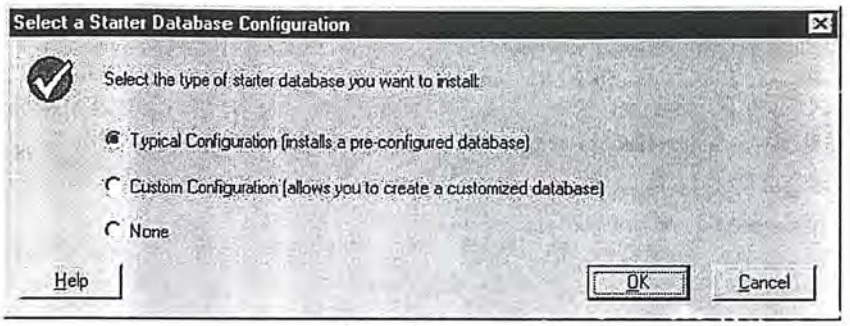
3 เลือก Cartridge หรือรูปแบบการทำงานพิเศษเพื่อเพิ่มความสามารถพิเศษ เช่นการเก็บข้อมูลในลักษณะมัลติมีเดียให้กับระบบฐานข้อมูล



รูปที่ ก-3 ขั้นตอนการลงที่ 3

- Oracle Parallel Server Option เพิ่มความสามารถให้ทำงานกับ Oracle instance หลายๆ ตัวบน Oracle database ตัวเดียวได้
- Oracle8 ConText Cartridge สามารถจัดการกับการดึงข้อมูลที่เป็น Text และใส่ Text ที่เป็นภาษาอื่นได้ ซึ่งทำให้การ Query และทำการแสดงผลในรูปแบบของภาษาอื่นหรือใน Format อื่นได้โดยใช้ SQL และ PL/SQL
- Oracle8 Image Cartridge สามารถจัดการกับการเก็บข้อมูลในรูปแบบของ Image
- Oracle8 Objects Option ทำการสร้างและจัดการกับ Object ได้ดีเท่ากับการ Integrates Object ด้วย Standard Relational
- Oracle8 Partitioning Option
- Oracle8 Spatial Cartridge สามารถเก็บช่องว่างเข้าไปใน Relational Database Model ได้
- Oracle8 Time Series Cartridge สามารถเก็บและเรียกข้อมูลที่เป็น Timestamp
- Oracle8 Visual Information Retrieval Cartridge ใช้กับข้อมูลที่เป็น Image ซึ่งจะมี Attribute ที่สำคัญคือ Color, Texture, Structure, Composition ซึ่งช่วยให้สามารถหา Image ที่เหมือนกันได้

4 ทำการสร้างฐานข้อมูลเริ่มต้น หรือเป็นกลุ่มของไฟล์ข้อมูลที่ถูกจัดการได้ด้วยระบบโอเอส ซึ่งหมายความว่าโอเอสสามารถเข้าถึงไฟล์นั้นได้ ซึ่งจะเป็นไฟล์ .ora ซึ่งมีรูปแบบคือ



รูปที่ ก-4 ขั้นตอนการลงที่ 4

- Typical Install Starter Database และ Server Components ต่างๆ
- Custom เราทำการสร้าง Starter Database เองซึ่งจะโปรแกรมจะเรียก Oracle Database Assistant มาทำการสร้างในขณะที่จบการ Install Oracle8
- None ไม่สร้าง Starter Database

5 Install Legato Storage Manager ซึ่งจัดการเกี่ยวกับการ Backup ข้อมูลซึ่งอาจไม่ใช้ตัวนี้ก็ได้ โดยมี Oracle's backup solution program (BSP) จัดการได้เหมือนกันซึ่งทำการเลือกให้ Install Legato Storage

6 Install Oracle8 HTML Document ลงใน HD

7 แสดง Oracle8 Document เสร็จแล้วก็จบการ Install

ก.2 Oracle Database Assistant

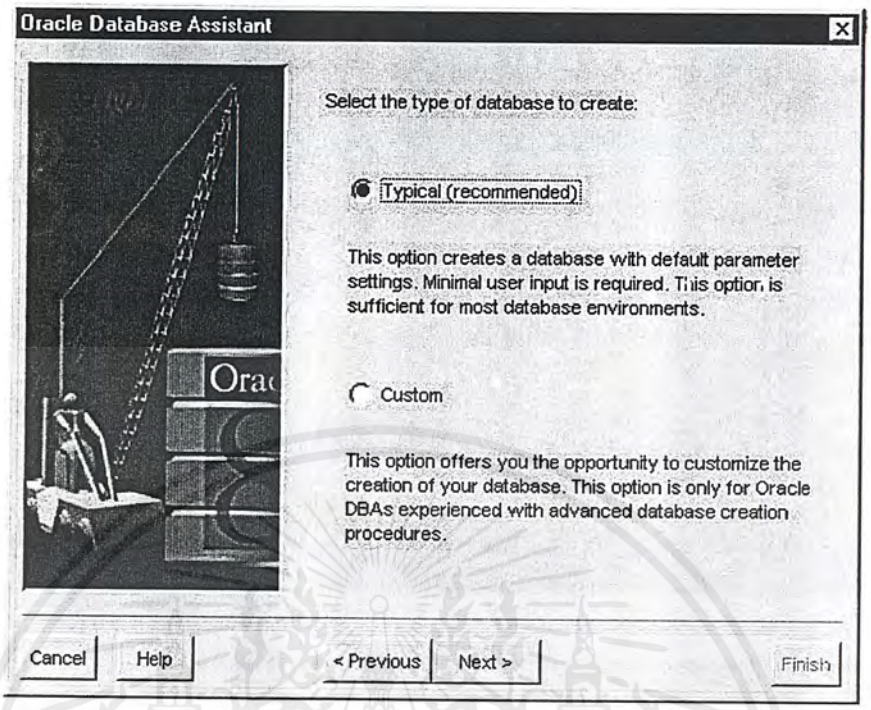
ใช้ในการสร้างฐานข้อมูลขึ้นมา

1 เลือกการสร้างฐานข้อมูล

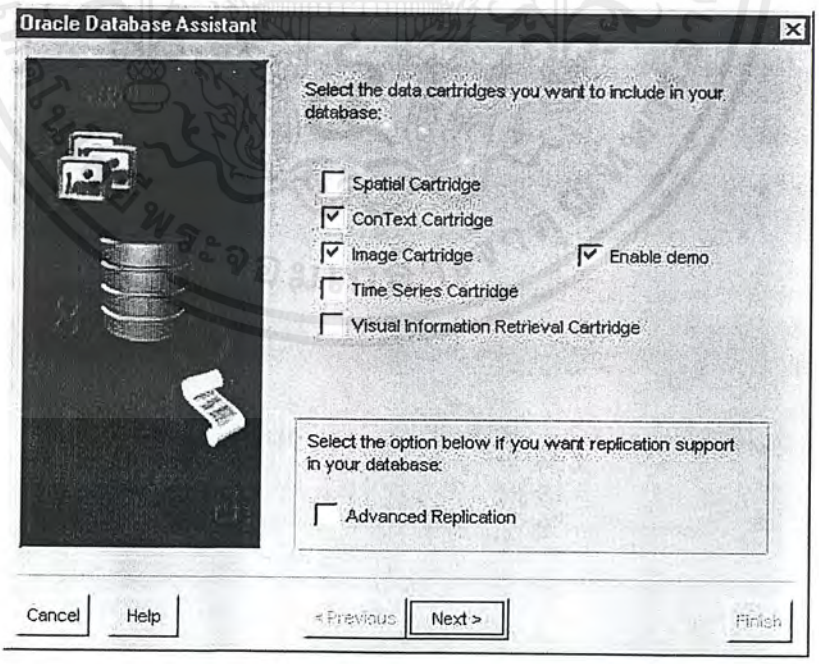
2 เลือกชนิดของการสร้างซึ่งเป็นแบบ Typical หรือ Custom โดย
Typical ทำการ Create Database ตามที่ Default ไว้ใน InitSid.ora
Custom จะต้องทำการเลือกค่า Parameter ต่างๆเอง เช่น Data, Control, Redo log file size, Tablespace size

3 ทำการเลือก Cartridge ที่จะใช้ใน Database ที่สร้างขึ้นซึ่งทำการเลือกไว้ในขั้นตอนการ Install Oracle

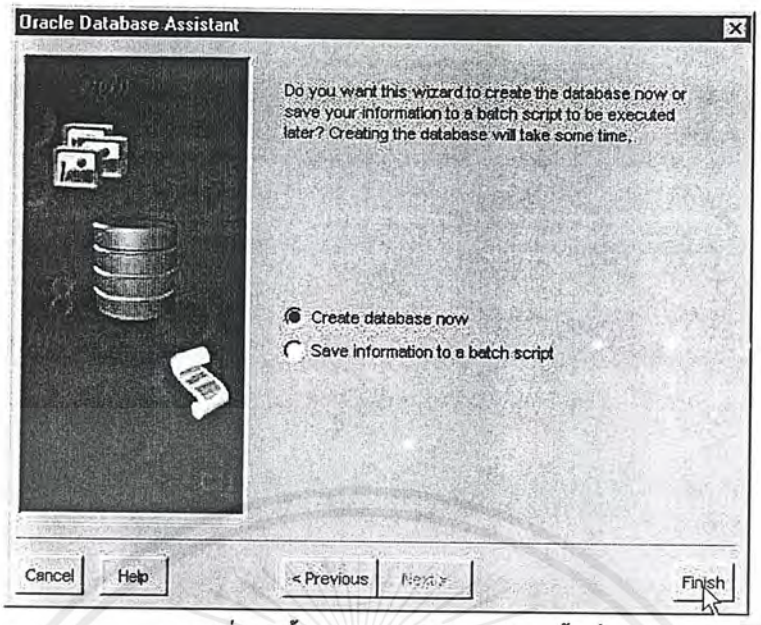
4 เริ่มทำการ Create Database



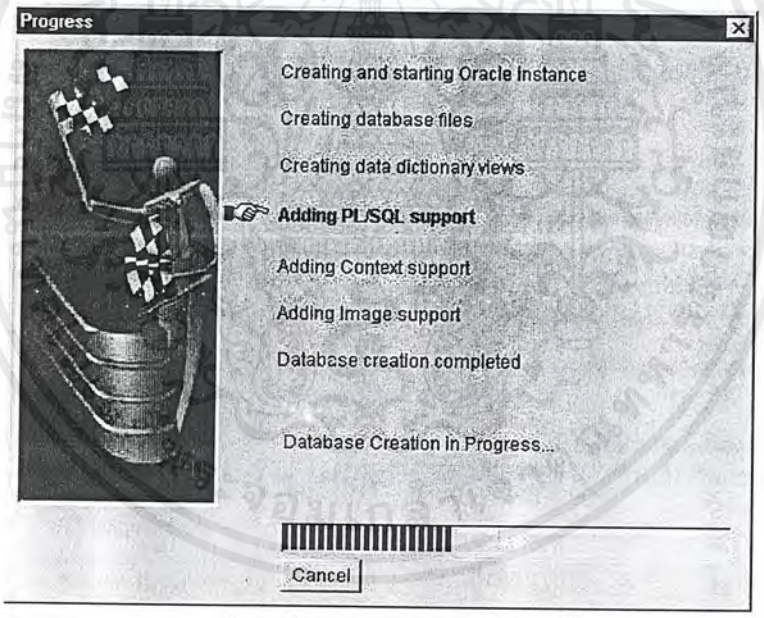
รูปที่ ก-5 ขั้นตอนการสร้างฐานข้อมูลขั้นที่ 1



รูปที่ ก-6 ขั้นตอนการสร้างฐานข้อมูลขั้นที่ 2



รูปที่ ก-7 ขั้นตอนการสร้างฐานข้อมูลขั้นที่ 3



รูปที่ ก-8 ขั้นตอนการสร้างฐานข้อมูลขั้นที่ 4

ก.3 ผู้ใช้ที่มีให้ในการจัดการกับระบบฐานข้อมูลในขั้นตอนเริ่มต้น

User Accounts	Passwords	Description
Internal	Default : oracle Custom : สามารถกำหนดในขั้นตอนการ Install โดยเลือก Custom Option install	จัดการ Database Administration Task ซึ่งรวมถึง Starting up และ Shutting down Database Internal account นี้เป็นเพียง alias ของ SYS Account และมีสิทธิ์ของ SYSDBA
Sys	Change_on_install	สามารถจัดการกับ Database role ดังนี้ ▪ CONNECT ▪ RESOURCE ▪ DBA ▪ AQ_AMINISTRATOR_ROLE ▪ AQ_USER_ROLE ▪ DELETE_CATALOG_ROLE ▪ EXECUTE_CATALOG_ROLE ▪ EXP_FULL_DATABASE ▪ IMP_FULL_DATABASE ▪ RECOVERY_CATALOG_OWNER ▪ SELECT_CATALOG_ROLE ▪ SNMPAGENT ▪ CTXADMIN 1 ▪ CTXAPP 1 ▪ CTXUSER 1
System	Manager	DBA User name
Scott	Tiger	มีบทบาทในการ Connect และ Resource
Demo	Demo	มีบทบาทในการ Connect และ Resource ซึ่งถ้าไม่จำเป็นควร Drop ทิ้ง
DBSNMP	Dbsnmp	ใช้กับ Oracle Enterprise Manager มีหน้าที่ในการ Connect, Resource และ SNMPAgent
CTXSYS	Ctxsys	ใช้กับ ConText Administrator User name ซึ่งมีหน้าที่ในการ Connect, Resource และ DBA

* Password นั้นเป็นตัวเล็กหมด

ภาคผนวก ข.
การติดตั้งไมโครซอฟท์ทรานแซคชั่นเซิร์ฟเวอร์

ในบทนี้จะแสดงการติดตั้งไมโครซอฟท์ทรานแซคชันเซิร์ฟเวอร์ให้สามารถทำงานเกี่ยวกับทรานแซคชันของระบบฐานข้อมูลออราเคิล 8 ได้โดยค่าเริ่มต้นที่ได้ทำการติดตั้งไมโครซอฟท์ทรานแซคชันเซิร์ฟเวอร์ตั้งไว้ให้ทำงานกับออราเคิล 7.3 สำหรับการลงไมโครซอฟท์ทรานแซคชันเซิร์ฟเวอร์จาก Option Pack ที่ได้มาจาก Visual Studio 6.0 Enterprise Edition แผ่น 2 นั้นมีการลงและการติดตั้งที่ง่ายจึงไม่กล่าวในบทนี้ ซึ่งบทความที่จะเสนอต่อไปนี้ได้มาจาก http://www.microsoft.com

Setting up Oracle Support

To set up Oracle to work with transactional Microsoft Transaction Server component, do the following:

1. Install Oracle Database Server software.

   Install the appropriate Oracle software on your database server system.

   If your Microsoft Transaction Server application is accessing an Oracle database on either Windows NT or Unix, ensure that the latest Oracle patch is installed on that system. You can obtain the latest Oracle patches for Windows NT from the Oracle FTP site. Go to ftp://oracle-ftp.oracle.com/ and select "server", "wgt-tech", "server", and "WindowsNT"

2. Install the Oracle Client software.

   Install the latest Oracle 7.3 or Oracle 8 client software on your Microsoft Transaction Server system.

   Make sure that the latest Oracle 7.3 or Oracle 8 Client software patch release is installed on the system containing your Microsoft Transaction Server components. Customers often upgrade these Oracle patch releases on the system containing their Oracle database but fail to install the Oracle patch release on the system containing their Microsoft Transaction Server components. Oracle has corrected several bugs that affect their XA transactional client support; therefore, installing the latest Oracle patch release on your Microsoft Transaction Server system is essential. You can obtain the latest Oracle patches Windows NT from the Oracle FTP site. Go to ftp://oracle-ftp.oracle.com/ and select "server", "wgt-tech", "server", and "windowsNT"

   If you are using Oracle 7.3 Client software, make sure that the correct version of the Oracle Ociw32.dll is installed as described in the Required Software section.

3. Install Microsoft Transaction Server 2.0.

   Install Microsoft Transaction Server 2.0 by installing the

Windows NT 4.0 Option Pack release. When you install Microsoft Transaction Server 2.0, the following software will be installed:

- Microsoft Transaction Server 2.0, including the Microsoft OCI Interface DLL (MTXOCI.DLL)
- Microsoft ODBC 3.5 Driver Manager
- Microsoft ODBC Driver for Oracle 2.0
- ADO 1.5

4. Install Windows NT 4.0 Service Pack 4 or Windows NT 4.0 Service Pack 5

   Install the Windows NT 4.0 Service Pack 4 or the Windows NT 4.0 Service Pack 5 release. These releases include corrections for several XA related MS DTC problems.

   You must install the Windows NT 4.0 Service Pack 4 or Windows NT 4.0 Service Pack 5 releases after you install the Windows NT4 Option Pack release. This is essential because the NT 4.0 Service Pack setup program will only update MTS and MS DTC when the Windows NT4 Option Pack has previously been installed.

5. Install Microsoft Data Access Components version 2.0 or later.

   You can install the Microsoft Data Access Components 2.0 release from the following Web URL by following the "Download" link to the Universal Data Access Downloads section:

   http://www.microsoft.com/data/

6. Update the Oracle Client Software Registry Keys.

   NOTE: If you are using the Oracle 7.3 Client software, you can skip this step.

   If you are using the Oracle 8 Client software, you must modify the values of two registry keys. Under the following registry key are two string-named values that specify the names of the Oracle 7.3 Client software .dlls:

   HKEY_LOCAL_MACHINE\SOFTWARE\ Microsoft\Transaction Server\Local Computer\My Computer

   String-Named Values:

   OracleXaLib "xa73.dll"
   OracleSqlLib "sqllib18.dll"

   Change these values to specify the names of the Oracle 8 Client software .dlls:

   OracleXaLib "xa80.dll"
   OracleSqlLib "sqllib80.dll"

7. Delete the Dtcxatm.log file.

   If you have never installed the Microsoft Transaction Server

2.0 Beta release, skip this step.

If you previously installed the Microsoft Transaction Server 2.0 Beta release, use Windows Explorer to determine if the Dtcxatm.log file is present on your system. If so, stop the Microsoft DTC service and delete the Dtcxatm.log file.

You should only delete the Dtcxatm.log file once when you first upgrade from the Microsoft Transaction Server 2.0 Beta release. You should never delete the Dtcxatm.log file thereafter, because it may contain vital recovery information.

8. Enable Oracle XA Transaction Support.

If you are using Oracle 7.3:

a. Make sure that V$XATRANS$ exists. This view should have been created when the XA library was installed. If this view does not exist, your Oracle system administrator must create it by running the Oracle-supplied script named "XAVIEW.SQL". This file can be found in C:\ORANT\RDBMS73\ADMIN. This SQL script must be executed by the Oracle user "SYS".

b. The Oracle system administrator must grant SELECT access to the public on the $XATRANS$ view. This can be done as follows:

```
Grant Select on V$XATRANS$ to public
```

If you are using Oracle8:

a. Oracle8 should have created both the V$XATRANS$ and the DBA_PENDING_TRANSACTIONS views. You should not need to create either of these views.

b. The Oracle system administrator must grant SELECT access to the public for the DBA_PENDING_TRANSACTIONS view. This can be done as follows:
Grant Select on DBA_PENDING_TRANSACTIONS to public.

9. Configure Enough Concurrent Distributed Transactions.

In the Oracle Instance Manager, click **Advanced Mode** on the **View** menu and select "Initialization Parameters" in the left pane. In the right pane, select **Advanced Tuning** and increase the "distributed_transactions" parameter to allow more concurrent MTS transactions to update the database at a single time.

10. Configure Integrated Security.

Integrated security allows an Oracle database to rely upon Windows NT authentication to validate database users. This permits a user to log in to Oracle without supplying a separate login ID or password. Users can maintain one login ID and password for both Windows NT and Oracle.

If your Microsoft Transaction Server components always supply a login ID and password when connecting to Oracle databases, then you are not using integrated security. This is true whether

your applications specify the login ID and password directly or indirectly through a data source name (DSN). In either event, you are not using integrated security and can ignore this step.

If you use integrated security, you must configure Microsoft Distributed Transaction Coordinator to run under a login ID and password authorized to connect to your Oracle database. This is required because during database recovery, Microsoft Distributed Transaction Coordinator opens your Oracle database to tell it the outcome of in doubt transactions.

You can configure the login ID for the Microsoft Distributed Transaction Coordinator as follows. From the **Start** menu, choose Settings and then select Control Panel. Start the Services applet on the control panel. Select MSDTC. Select "Log On As" and specify a login ID and password. Use the Oracle security administration tools to make sure that the login ID you specify is authorized to open your Oracle database.

For more information on Oracle's integrated Windows NT security facilities, consult your Oracle documentation.

11. Configure Oracle Multi-Threaded Server Support.

You must configure the Oracle "Multi-Threaded Server" feature if you want to open a database link to a remote Oracle database. This is essential because when using XA transaction support, the Oracle database must be able to move the XA transaction between processes (in the general case) so it cannot have any operating system file descriptors open but rather it must connect to the remote database using a virtual circuit. Support for Virtual circuits is available only with Oracle "Multi-Threaded Server".

If the Oracle "Multi-Threaded Server" feature is not configured properly, Oracle reports the following error:

ORA-24777: Cannot create migratable transaction

NOTE: Oracle refers to the "Multi-Threaded Server" feature using the acronym "MTS". Microsoft refers to "Microsoft Transaction Server" using the acronym "MTS". The only relation between these two terms is the one described in the preceding section. Microsoft apologizes for the confusion.

12. Configure Oracle to support more connections.

If you want to create more than a few dozen connections to an Oracle database, you must configure the Oracle server to support additional database connections. Please see the "Configuring Oracle to Support a Large Number of Connections" section for more information.

Testing Installation and Configuration of MTS Support for Oracle

After installing and configuring Oracle support, you must validate your Oracle installation using the Oracle test program installed with MTS. The Oracle test program uses Oracle's OCI XA interfaces in much the same way that MTS uses them.

The Oracle test program determines whether you can connect to an Oracle database using Oracle's XA facility. The Oracle test program uses standard Oracle interfaces and transaction facilities. It makes no use of Microsoft Transaction Server or Microsoft Distributed Transaction Coordinator. Therefore, failure of the test program indicates an improper installation or configuration of your Oracle system. If the Oracle test program fails, reinstall and reconfigure Oracle, or contact the Oracle Support Organization for assistance.

To obtain the Oracle test program:

The Oracle test program for Oracle7 is installed when Microsoft Transaction Server is installed. You may obtain both the source and object code for the Oracle test program for Oracle 7.3 from ftp://ftp.microsoft.com/bussys/viper/Utilities/OracleTestProgram/Oracle7.3/
You may obtain both the source and object code for the Oracle test program for Oracle8 from ftp://ftp.microsoft.com/bussys/viper/Utilities/OracleTestProgram/Oracle8/

You may compile the source code for either Oracle 7.3 or Oracle8 using the C++ compiler. If you recompile the Oracle test program, Microsoft recommends you call the resulting program "TestOracleXaConfig.exe". That is the name Microsoft gives it and the instructions that follow assume that you use that name.

To run the Oracle test program follow these steps:

1. Verify that you have installed all of the correct versions of the software as described in the Required Software section.

2. Create an ODBC DSN that refers to your Oracle database. Make sure that your DSN uses the new Microsoft Oracle ODBC driver.

3. Ensure that you have enabled Oracle XA support as described in EnableOracleXATransactionSupport.

4. Delete all existing Oracle trace files from the computer containing the MTS components that access the Oracle database. The easiest way to do this is to use the Windows Explorer to locate and delete all *.trc files.

   If the Oracle test program fails, the trace files may help you determine the source of the problem. By deleting all obsolete trace files, you make it easier to find any newly created ones.

5. If you never installed the Microsoft Transaction Server 2.0 beta release, you can skip this step.

   If you have previously installed the Microsoft Transaction Server 2.0 beta release, use the Windows Explorer to determine if the Dtcxatm.log file is present on your system. If so, stop the Microsoft DTC service and delete the Dtcxatm.log file.

   You should only delete the Dtcxatm.log file once when you first upgrade from the Microsoft Transaction Server 2.0 beta release. You should never delete the Dtcxatm.log file thereafter, because it may contain vital recovery information.

6. From the MS-DOS Command prompt run the Oracle test program (TestOracleXAConfig.exe) and supply your Oracle server user ID, password and service_name. For example:
   c:>TestOracleXaConfig.exe -U<user id> -P<Password> -S<Service_Name as contained in the TNS file>.

7. If you run the test program without any parameters, it displays help information that describes the required parameters. The test program displays information about each Oracle operation it performs and indicates whether each operation was successful.

8. If the Oracle test program is able to connect to your Oracle database server without error, then it is very likely that MTS works with Oracle also. If the Oracle test program reports any errors, follow these steps:

   - Document the exact error message that the Oracle test program displays.
   - Examine the Oracle trace file produced when running the Oracle test program. The Oracle trace information is located in the *.trc file. The Oracle trace file contains extended error information that is extremely helpful in diagnosing problems.
   - Contact your Oracle support representative for assistance.

โปรแกรมที่ใช้ในการทดสอบการทำงานทรานแซคชัน

แสดงเพียงการสร้างคอมโพเนนต์ที่ทำการติดตั้งลง MTS เท่านั้นในด้านของโปรแกรมเรียกใช้คอมโพเนนต์นี้เราหวังว่าผู้อ่านสามารถทำการเขียนได้เอง

```
Const strConn1 = "PROVIDER=MSDASQL;DSN=OracleDSN;UID=boy;PWD=boy;"
Const strConn2 = "PROVIDER=MSDASQL;DSN=OracleDSN;UID=boy;PWD=boy;"

Public Sub Transfer(ByVal ToAccountID As Integer, ByVal Money As
Integer)
Dim cn As ADODB.Connection
Dim rs As ADODB.Recordset
Dim tx As ObjectContext
Dim strSQL As String

Set tx = GetObjectContext()

' ----- Check AccountID ------
If mvarAccountID <= 0 Then
   tx.SetAbort
   Err.Raise InvalidAccountID, , "Invalid Account ID"
   Set tx = Nothing
   Exit Sub
End If

On Error GoTo TransferError
```

```
' ----- Connect to Account1 ------
Set cn = New ADODB.Connection
cn.ConnectionString = strConn1  ' Public const StrConn in Module shop
cn.CursorLocation = adUseClient
cn.Open

Set rs = New ADODB.Recordset
rs.CursorType = adOpenKeyset
rs.LockType = adLockOptimistic

strSQL = "Update tblAccount1 " & _
         "Set money = money - " & Money & _
         "Where AccountID = " & mvarAccountID
Set rs = cn.Execute(strSQL)

cn.Close

cn.ConnectionString = strConn2
cn.Open

strSQL = "Update tblAccount2 " & _
         "Set money = money + " & Money & _
         "Where AccountID = " & ToAccountID
Set rs = cn.Execute(strSQL)

tx.SetComplete
Set tx = Nothing

Exit Sub
TransferError:
    tx.SetAbort
    Err.Raise Err.Number, , Err.Description
    Set tx = Nothing

End Sub
```

ซึ่งในระดับฐานข้อมูลมีการสร้างดังนี้

```
create table tblAccount1(
AccountID Number, AccountName Varchar2(30), Money number);

create table tblAccount2(
AccountID Number, AccountName Varchar2(30), Money number);

insert into tblaccount1
values (1, 'niran', 1000);

insert into tblaccount2
values (2, 'niran', 3000);
```

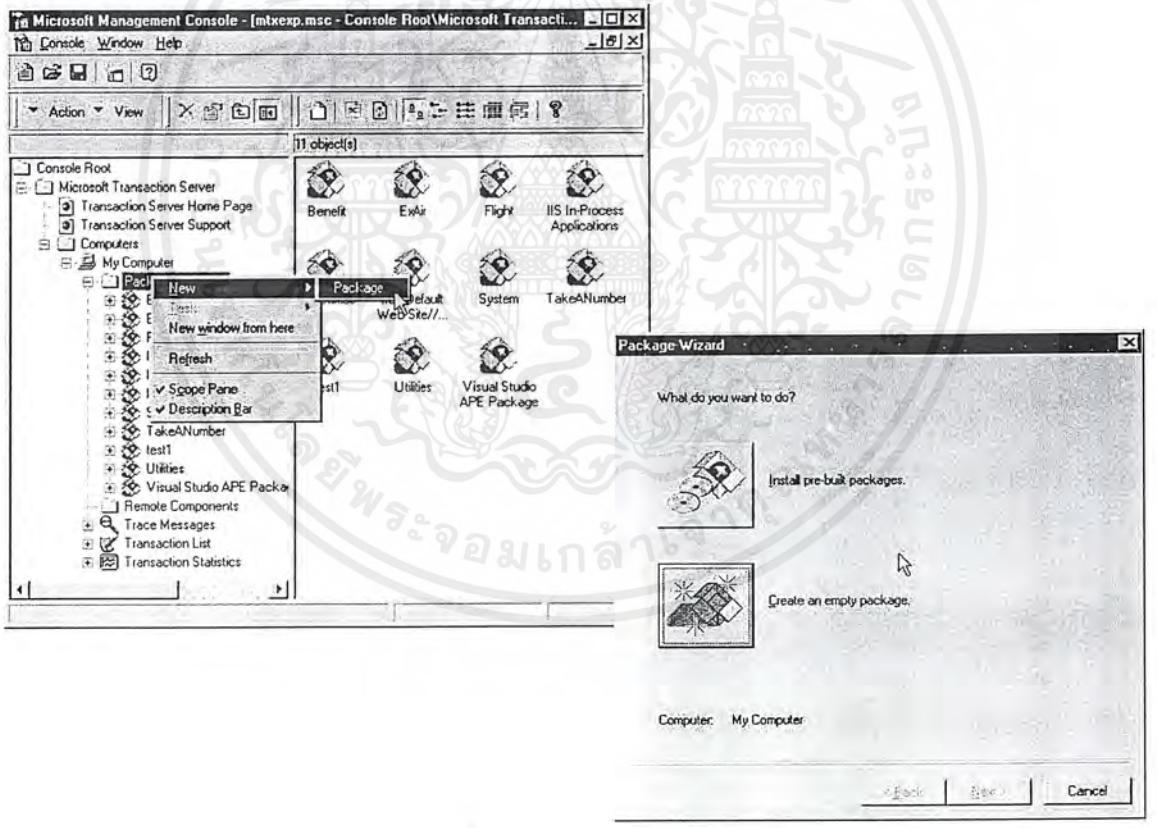
ภาคผนวก ค.
การติดตั้งคอมโพเนนต์บน
Microsoft Transaction Server

ขั้นตอนการติดการติดตั้งคอมโพเนนต์บน Microsoft Transaction Server

ขั้นตอนที่ 1 เขียน COM component ที่ทำงานฝั่ง server โดยเลือกสร้างโปรเจกต์ชนิด ActiveX dll

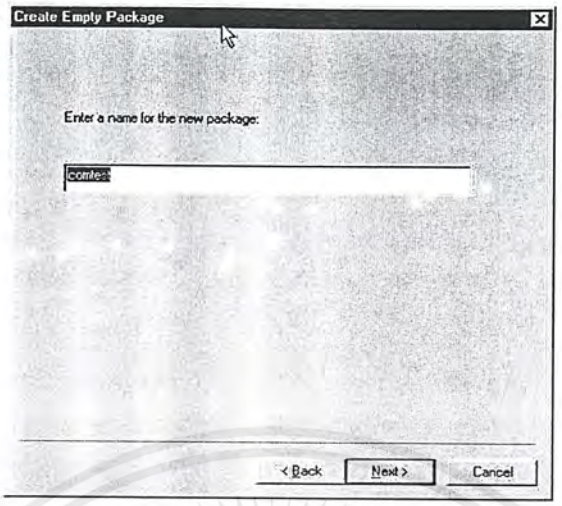
ขั้นตอนที่ 2 นำเอา COM component ไปติดตั้งลงใน Microsoft Transaction Server (MTS)

2.1 สร้างแพคเกจว่าง โดยเลือก create an empty package ดังรูป ค-1



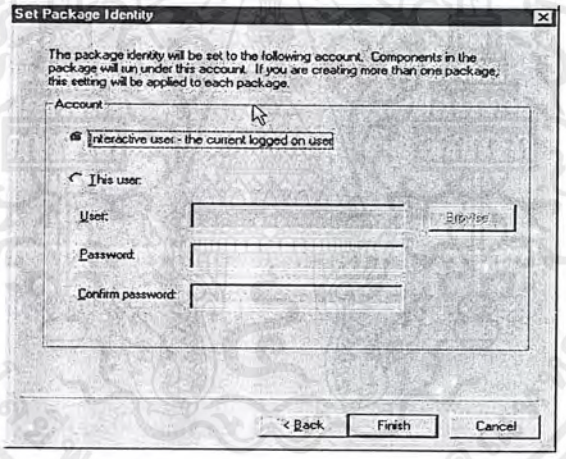
รูปที่ ค-1 แสดงการเลือกสร้างแพคเกจใหม่

2.2 กำหนดชื่อแพคเกจที่ต้องการสร้าง



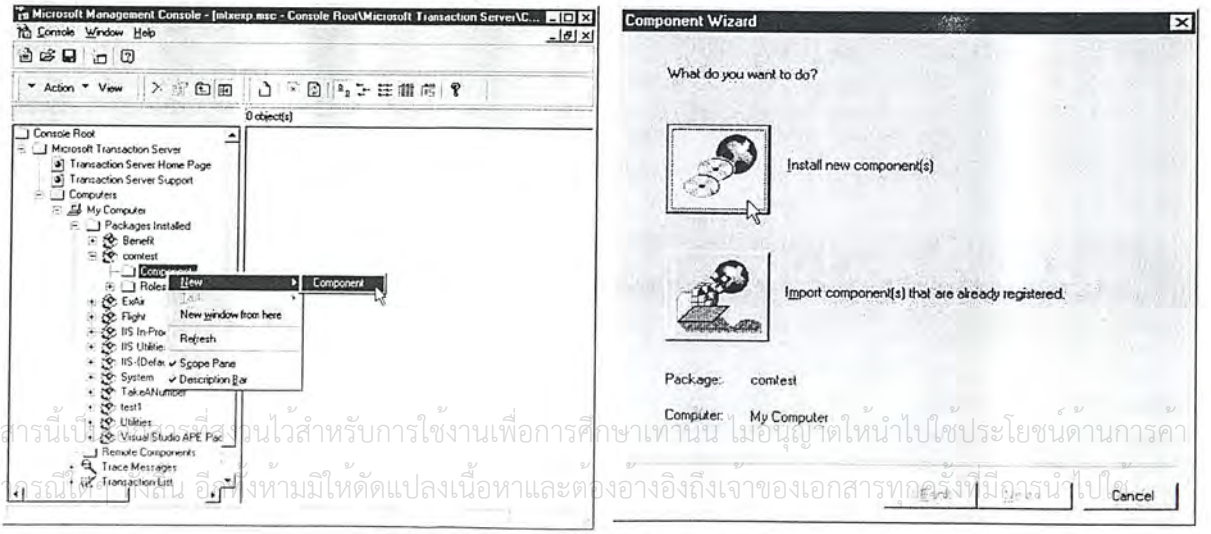
รูปที่ ค-2 แสดงการกำหนดชื่อแพคเกจ

2.3 กำหนด account ของการเรียกใช้คอมโพเนนต์ ในตัวอย่างนี้เลือก แบบ interactive user



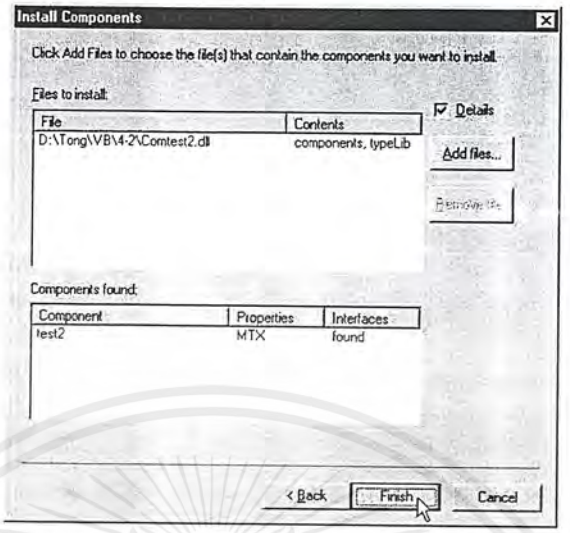
รูปที่ ค-3 แสดงการกำหนด account

2.4 หลังจากสร้างแพคเกจแล้วจะต้องเพิ่มคอมโพเนนท์ลงไปในแพคเกจ โดยเลือก install new component ดังรูป ค-4



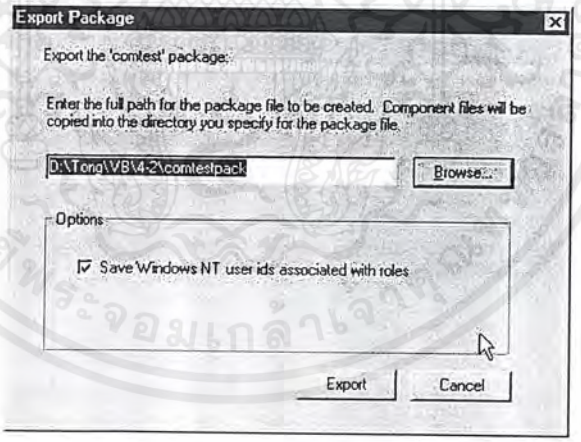
รูปที่ ค-4 แสดงการเพิ่มคอมโพเนนท์ลงไปในแพคเกจ

2.5 กำหนด path ของไฟล์ .dll



รูป ค-5 แสดงการกำหนด Path ของคอมโพเนนต์ที่เป็นไฟล์ .dll

2.6 จากนั้นจะต้องสร้างไฟล์ติดตั้งเพื่อนำไปติดตั้งที่ฝั่งไคลเอนต์โดยกดคลิกขวาที่ชื่อ ของแพคเกจ แล้วกด export จะปรากฏดังรูป แล้วเลือก path ที่ต้องการสร้างไฟล์ติดตั้ง



รูปที่ ค-6 แสดงการเลือก Path ที่ต้องการสร้างไฟล์ติดตั้ง

ขั้นตอนที่ 3 การเขียนฟอร์มฝั่งไคลเอนต์โดยจะต้องอ้างอิงคอมโพเนนท์ที่สร้างไว้ฝั่งเซิฟเวอร์

ขั้นตอนที่ 4 นำไฟล์ติดตั้งที่ได้จากการ Export และ ไฟล์แอปพลิเคชั่น .exe ที่คอมไพลจาก ฟอร์มฝั่งไคลเอนต์ ไปรันโดยจะต้องรันไฟล์ติดตั้งก่อนเพื่อบอกให้ไคล์เอนต์รู้ว่าเซิฟเวอร์อยู่ที่ใด

รายละเอียดลักษณะของ MTS

Microsoft Transaction Server—Complete Features at a Glance	
Feature	**Description and Benefits**
Ease-of-Use	
Simplified Three-Tiered Application Development	MTS completely separates three-tiered infrastructure from presentation and application logic. This lets MTS developers build their applications as a collection of single-user COM components and then deploy components in the appropriate tier. Components that run on middle-tiered servers automatically support multi-user operation.
COM Component Support	MTS supports any tool that can produce COM DLLs, including Microsoft Visual Basic, Microsoft Visual C++® development system, and Microsoft Visual J++.
Simple Application Programming Interfaces	MTS developers have to learn only a simple set of APIs to deploy their components in the MTS run-time environment. Extensive knowledge of COM APIs and Win32 APIs is not required.
Support for Component Packages	MTS lets developers combine related components into packages. Packages provide an easy way to secure, manage, and deploy components as a group.
MTS Explorer	The MTS Explorer is a GUI-based systems management console that makes it easy for administrators to create and manage packages, configure component properties such as security and transaction behavior, and monitor/manage operating servers. The MTS Explorer supports advanced ease-of-use features such as drag-and-drop editing and multiple item selection.
Microsoft Management Console Support	The MTS Explorer management interface can run as a snap-in to the Microsoft Management Console (MMC). MMC integration provides users with a consolidated management environment with a consistent look and feel.
Automatic Client Installation Utility	MTS provides a utility that simplifies and automates the process of configuring client machines that need to access MTS-based components running on a server.
Comprehensive Functionality	
Automatic Transaction Management	MTS provides automatic transaction management services to components. Developers or administrators use simple *property page* interfaces to specify what level of transaction protection is required by a given component. At run time, MTS performs all required transaction management automatically. This dramatically simplifies development and facilitates component reuse.
Database Connection Pooling	MTS allocates connections to database resources from a pool. Components only require connections when they are active. This lowers the total number of database connections required to support clients, increasing database performance.

Microsoft Transaction Server—Complete Features at a Glance

Feature	Description and Benefits
Comprehensive Functionality (Cont.)	
Transactional ODBC Driver for Oracle and DB2 Databases	MTS includes an ODBC 3.0-compliant driver for Oracle version 7.3 and later and DB2. ODBC 3.0 compliance lets developers build MTS applications that access these databases (on any platform), along with access to other resources such as SQL Server and Microsoft Message Queue Server, in transactional units of work for full data integrity protection. ODBC 3.0 compliance also lets MTS perform connection pooling of Oracle and DB2 database sessions for improved performance.
Support for Multiple Databases and Resource Managers	Components running under MTS can access multiple databases and other resources such as message queuing systems and mainframe applications, simultaneously and with full data integrity protections. MTS automatically manages transactions so that all databases and other resources included in a request commit or abort as a unit.
No MTS Client Footprint	MTS applications require no libraries or run-time environment on client machines beyond that provided by COM and Distributed COM (DCOM).
Windows NT Workstation and Windows 95/98 Support	MTS supports operation on Windows NT Workstation and Windows 95/98 so that companies can deploy stand-alone versions of their MTS applications.
Process Isolation	MTS Administrators can configure packages to run in their own process to ensure that failures do not spread to other packages.
Automatic Thread Pooling	As requests come from clients, MTS automatically assigns threads to components from a pre-allocated pool. When a component finishes executing, MTS reclaims the thread. This reduces the overhead of thread creation/deletion for better performance.
Automatic Object Instance Management	MTS creates instances of components on the server only when requested and reclaims memory resources when the component finishes executing. This form *of just-in-time (JIT) activation* and *as-soon-as-possible* deactivation minimizes memory requirements on server machines.
Shared Property Manager	MTS provides the Shared Property Manager to simplify programming of components that need to save state information or share state with other components.
Integration with Windows NT Features and Microsoft Products	
Integration with Microsoft SQL Server 6.5	Microsoft SQL Server 6.5 provides an OBDC 3.0 driver that provides extremely fast and efficient transactional integration, based on Microsoft Distributed Transaction Coordinator (DTC) to MTS applications. Because MTS also uses DTC for transaction management, administrators can use the same utilities to manage transactions between MTS and SQL Server, simplifying systems administration

Microsoft Transaction Server—Complete Features at a Glance	
Integration with Windows NT Features and Microsoft Products (Cont.)	
IIS Integration	Microsoft Internet Information Server version.4.0 is integrated with MTS and uses MTS for many run-time services such as transaction management. MTS integration makes it easy for IIS Active Server Pages to call MTS components that access databases, mainframe applications, and message queues with full data integrity protection. MTS integration also provides IIS with process isolation to prevent individual failures from affecting other parts of a Web site, enhanced run-time services such as thread and connection pooling for better performance, and easier component management.
MSMQ Integration	MTS provides integration with Microsoft Message Queue Server (MSMQ). MSMQ integration lets MTS-based applications communicate in a reliable, loosely coupled fashion. MSMQ operations (such as send and receive) enlist in MTS transactions automatically for data integrity protection
SNA Server 4.0 Integration	MTS can use Microsoft SNA Server 4.0 and the COM Transaction Integrator to include mainframe applications (e.g., running under CICS and IMS) and their data in transactions managed by MTS. MTS-based applications can update one or more databases, send or receive messages with MSMQ, and invoke a mainframe application that updates data (e.g., in VSAM), and all updates, including those on the mainframe, will commit or abort as a whole. Microsoft SNA Server 4.0 also makes it easy to generate COM-based interfaces to mainframe applications for simplified MTS programming.
Support for Windows NT Clustering Services	MTS supports operation on clustered Windows NT Servers. Clustering enables administrators to configure MTS packages for automatic failover and fault-tolerant, high-availability operation.
Windows NT Security Integration	MTS provides a *role-based* security mechanism that is integrated with the Windows NT security environment. Roles let developers design security protections into their components without knowing how the components will be ultimately deployed. Administrators map roles to users when they install components into MTS.

ภาคผนวก ง.

การโปรแกรมเชิงวัตถุด้วยวิชวลเบสิก 6

ในบทนี้แสดงคุณสมบัติการนำวิชวลเบสิกในการสร้างความสัมพันธ์แบบออบเจ็กต์ที่สำคัญซึ่งคือความสัมพันธ์แบบโพลีมอร์ฟิซึม ซึ่งได้มาจาก MSDN Library Visual Studio 6.0

1. How Visual Basic Provides Polymorphism

Most object-oriented programming systems provide polymorphism through *inheritance*. That is, the hypothetical Flea and Tyrannosaur classes might both inherit from an Animal class. Each class would override the Animal class's Bite method, in order to provide its own bite characteristics.

The polymorphism comes from the fact that you could call the Bite method of an object belonging to any class that derived from Animal, without knowing which class the object belonged to.

Providing Polymorphism with Interfaces

Visual Basic doesn't use inheritance to provide polymorphism. Visual Basic provides polymorphism through multiple ActiveX *interfaces*. In the Component Object Model (COM) that forms the infrastructure of the ActiveX specification, multiple interfaces allow systems of software components to evolve without breaking existing code.

An *interface* is a set of related properties and methods. Much of the ActiveX specification is concerned with implementing standard interfaces to obtain system services or to provide functionality to other programs.

In Visual Basic, you would create an Animal interface and implement it in your Flea and Tyrannosaur classes. You could then invoke the Bite method of either kind of object, without knowing which kind it was.

Polymorphism and Performance

Polymorphism is important for performance reasons. To see this, consider the following function:

```
Public Sub GetFood(ByVal Critter As Object, _
ByVal Food As Object)
   Dim dblDistance As Double
   ' Code to calculate distance to food (omitted).
   Critter.Move dblDistance    ' Late bound
   Critter.Bite Food           ' Late bound
End Sub
```

The Move and Bite methods are *late bound* to Critter. Late binding happens when Visual Basic can't determine at compile time what kind of object a variable will contain. In this example, the Critter

argument is declared As Object, so at run time it could contain a reference to any kind of object — like a Car or a Rock.

Because it can't tell what the object will be, Visual Basic compiles some extra code to ask the object if it supports the method you've called. If the object supports the method, this extra code invokes it; if not, the extra code raises an error. Every method or property call incurs this additional overhead.

By contrast, interfaces allow *early binding*. When Visual Basic knows at compile time what interface is being called, it can check the type library to see if that interface supports the method. Visual Basic can then compile in a direct jump to the method, using a virtual function table (vtable). This is many times faster than late binding.

Now suppose the Move and Bite methods belong to an Animal interface, and that all animal classes implement this interface. The Critter argument can now be declared As Animal, and the Move and Bite methods will be early bound:

```
Public Sub GetFood(ByVal Critter As Animal, _
ByVal Food As Object)
   Dim dblDistance As Double
   ' Code to calculate distance to food (omitted).
   Critter.Move dblDistance    ' Early bound (vtable).
   Critter.Bite Food           ' Early bound (vtable).
End Sub
```

For More Information "Creating and Implementing an Interface" creates an Animal interface and implements it in Flea and Tyrannosaur classes.

2. Creating and Implementing an Interface

As explained in "How Visual Basic Provides Polymorphism," an interface is a set of properties and methods. In the following code example, you'll create an Animal interface and implement it in two classes, Flea and Tyrannosaur.

You can create the Animal interface by adding a class module to your project, naming it Animal, and inserting the following code:

```
Public Sub Move(ByVal Distance As Double)

End Sub

Public Sub Bite(ByVal What As Object)

End Sub
```

Notice that there's no code in these methods. Animal is an *abstract class*, containing no implementation code. An abstract class isn't meant for creating objects — its purpose is to provide the template for an interface you add to other classes. (Although, as it turns out, sometimes it's useful to implement the interface of a class that isn't abstract; this is discussed later in this topic.)

> **Note** Properly speaking, an abstract class is one from which you can't create objects. You can always create objects from Visual

> Basic classes, even if they contain no code; thus they are not truly abstract.

Now you can add two more class modules, naming one of them Flea and the other Tyrannosaur. To implement the Animal interface in the Flea class, you use the Implements statement:

```
Option Explicit
Implements Animal
```

As soon as you've added this line of code, you can click the left-hand (Object) drop down in the code window. One of the entries will be Animal. When you select it, the right-hand (Procedure) drop down will show the methods of the Animal interface.

Select each method in turn, to create empty procedure templates for all the methods. The templates will have the correct arguments and data types, as defined in the Animal class. Each procedure name will have the prefix Animal_ to identify the interface.

> **Important** An interface is like a contract. By implementing the interface, a class agrees to respond when any property or method of the interface is invoked. Therefore, you must implement *all* the properties and methods of an interface.

You can now add the following code to the Flea class:

```
Private Sub Animal_Move(ByVal Distance As Double)
   ' (Code to jump some number of inches omitted.)
   Debug.Print "Flea moved"
End Sub

Private Sub Animal_Bite(ByVal What As Object)
   ' (Code to suck blood omitted.)
   Debug.Print "Flea bit a " & TypeName(What)
End Sub
```

You may be wondering why the procedures are declared Private. If they were Public, the procedures Animal_Jump and Animal_Bite would be part of the Flea interface, and we'd be stuck in the same bind we were in originally, declaring the Critter argument As Object so it could contain either a Flea or a Tyrannosaur.

Multiple Interfaces

The Flea class now has two interfaces: The Animal interface you've just implemented, which has two members, and the default Flea interface, which has no members. Later in this example you'll add a member to one of the default interfaces.

You can implement the Animal interface similarly for the Tyrannosaur class:

```
Option Explicit
Implements Animal

Private Sub Animal_Move(ByVal Distance As Double)
   ' (Code to pounce some number of yards omitted.)
   Debug.Print "Tyrannosaur moved"
End Sub
```

```
Private Sub Animal_Bite(ByVal What As Object)
   ' (Code to take a pound of flesh omitted.)
   Debug.Print "Tyrannosaur bit a " & TypeName(What)
End Sub
```

Exercising the Tyrannosaur and the Flea

Add the following code to the Load event of Form1:

```
Private Sub Form_Load()
   Dim fl As Flea
   Dim ty As Tyrannosaur
   Dim anim As Animal

   Set fl = New Flea
   Set ty = New Tyrannosaur
   ' First give the Flea a shot.
   Set anim = fl
   Call anim.Bite(ty)   'Flea bites dinosaur.
   ' Now the Tyrannosaur gets a turn.
   Set anim = ty
   Call anim.Bite(fl)   'Dinosaur bites flea.
End Sub
```

Press F8 to step through the code. Notice the messages in the Immediate window. When the variable anim contains a reference to the Flea, the Flea's implementation of Bite is invoked, and likewise for the Tyrannosaur.

The variable anim can contain a reference to any object that implements the Animal interface. In fact, it can *only* contain references to such objects. If you attempt to assign a Form or PictureBox object to anim, an error will occur.

The Bite method is early bound when you call it through anim, because Visual Basic knows at compile time that whatever object is assigned to anim will have a Bite method.

Passing Tyrannosaurs and Fleas to Procedures

Remember the GetFood procedure from "How Visual Basic Provides Polymorphism?" You can add the *second* version of the GetFood procedure — the one that illustrates polymorphism — to Form1, and replace the code in the Load event with the following:

```
Private Sub Form_Load()
   Dim fl As Flea
   Dim ty As Tyrannosaur

   Set fl = New Flea
   Set ty = New Tyrannosaur
   'Flea dines on dinosaur.
   Call GetFood(fl, ty)
   ' And vice versa.
   Call GetFood(ty, fl)
End Sub
```

Stepping through this code shows how an object reference that you pass to an argument of another interface type is converted into a

reference to the second interface (in this case, Animal). What happens is that Visual Basic queries the object to find out whether it supports the second interface. If the object does, it returns a reference to the interface, and Visual Basic places that reference in the argument variable. If the object does not support the second interface, an error occurs.

Implementing Methods That Return Values

Suppose the Move method returned a value. After all, you know how far you want an Animal to move, but an individual specimen might not be able to move that far. It might be old and decrepit, or there might be a wall in the way. The return value of the Move method could be used to tell you how far the Animal actually moved.

```
Public Function Move(ByVal Distance As Double) _
As Double

End Function
```

When you implement this method in the Tyrannosaur class, you assign the return value to the procedure name, just as you would for any other Function procedure:

```
Private Function Animal_Move(ByVal Distance _
As Double) As Double
   Dim dblDistanceMoved As Double
   ' Code to calculate how far to pounce (based on
   ' age, state of health, and obstacles) is omitted.
   ' This example assumes that the result has been
   ' placed in the variable dblDistanceMoved.
   Debug.Print "Tyrannosaur moved"; dblDistanceMoved
   Animal_Move = dblDistanceMoved
End Function
```

To assign the return value, use the full procedure name, including the interface prefix.

For More Information The interfaces you implement can have properties as well as methods. "Implementing Properties" discusses some differences in the way properties are implemented.

3. Implementing Properties

This topic continues the code example begun in "Creating and Implementing an Interface," adding properties to the Animal interface that was implemented in the Flea and Tyrannosaur classes. You may find it helpful to read that topic before beginning this one.

Suppose we give the Animal class an Age property, by adding a Public variable to the Declarations section:

```
Option Explicit
Public Age As Double
```

The Procedure drop downs in the code modules for the Tyrannosaur and Flea classes now contain property procedures for implementing the Age property, as shown in Figure 9.10.

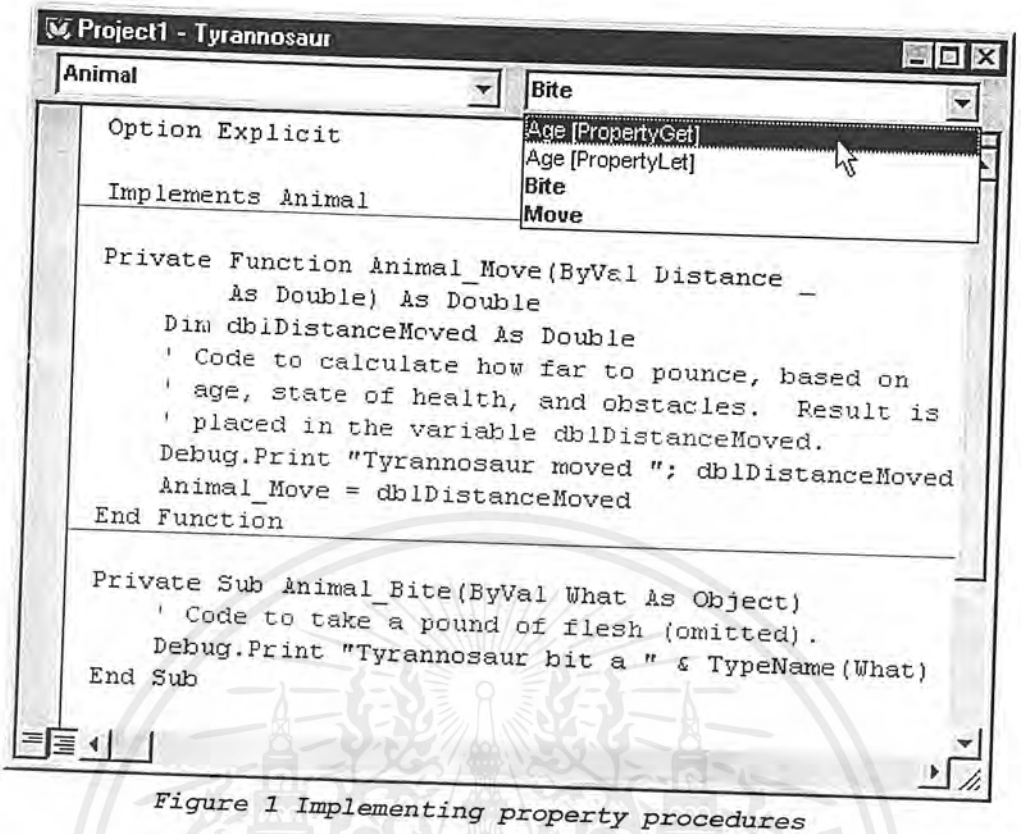


Figure 1 Implementing property procedures

This illustrates a point made in "Adding Properties to a Class" earlier in this chapter. Using a public variable to implement a property is strictly a convenience for the programmer. Behind the scenes, Visual Basic implements the property as a pair of property procedures.

You must implement both procedures. The property procedures are easily implemented by storing the value in a private data member, as shown here:

```
Private mdblAge As Double

Private Property Get Animal_Age() As Double
   Animal_Age = mdblAge
End Property

Private Property Let Animal_Age(ByVal RHS As Double)
   mdblAge = RHS
End Property
```

The private data member is an implementation detail, so you have to add it yourself.

> **Note** When Implements provides the template for a Property Set or Property Let, it has no way of determining the name of the last argument, so it substitutes the name RHS, as shown in the code example above.

There's no data validation on a property implemented as a public data member, but that doesn't mean you can't add validation code to the Property Let for Animal_Age. For example, you might want to restrict the values to ages appropriate for a Tyrannosaur or a Flea, respectively.

In fact, this shows the independence of interface and implementation. As long as the interface matches the description in the type library, the implementation can be anything.
Before you go on to the next step, remove the implementation of the read-write Age property from both class modules.

Implementing a Read-Only Property

Of course, allowing the age of an animal to be set arbitrarily is bad object design. The object should know its own age, and provide it to the user as a read-only property. Remove the public variable Age from the Animal class, and add the template for a read-only age property, like this:

```
Public Property Get Age() As Double

End Property
```

Now the Procedure drop downs in the code windows for the Tyrannosaur and Flea classes contain only a single entry, Age [PropertyGet]. You might implement this for the Tyrannosaur as follows:

```
Private mdblBirth As Double

Private Property Get Animal_Age() As Double
   Animal_Age = Now - mdblBirth
End Property
```

The code above returns the age of the Tyrannosaur in days. You could set mdblBirth in the Initialize event of the Tyrannosaur class, as here:

```
Private Sub Class_Initialize()
   mdblBirth = Now
End Sub
```

And of course you could return the property value in more commonly used units, such as dog years.

For More Information We've been tossing interfaces and objects around like they were the same thing, seemingly putting references to objects into one object variable, and references to interfaces into another. "Time Out for a Brief Discussion of Objects and Interfaces" clears matters up.

4. Time Out for a Brief Discussion of Objects and Interfaces

This topic completes the code example begun in "Creating and Implementing an Interface," and continued in "Implementing Properties." You may find it helpful to read those topics before beginning this one.

The Tyrannosaur and Flea code example seems to play fast and loose with interfaces and objects. References to objects are assigned to one object variable, and references to interfaces to another.

In fact, *all of the references are object references*. A reference to an interface is also a reference to the object that implements the

interface. Furthermore, an object may have multiple interfaces, but it's still the same object underneath.

In Visual Basic, each class has a default interface that has the same name as the class. Well, almost the same. By convention, an underscore is prefixed to the class name. The underscore indicates that this interface is hidden in the type library.

Thus the Tyrannosaur class has a default interface called _Tyrannosaur. Because Tyrannosaur also implements Animal, the class has a second interface named Animal.

However, underneath it all, the object is still a Tyrannosaur. Place a command button on Form1, and add the following code:

```
Private Sub Command1_Click()
   Dim ty As Tyrannosaur
   Dim anim As Animal

   Set ty = New Tyrannosaur
   Set anim = ty
   MsgBox TypeName(anim)
End Sub
```

You might expect the message box to display "Animal," but in fact it displays "Tyrannosaur."

Querying for Interfaces

When you assign a Tyrannosaur object to variable of type Animal, Visual Basic asks the Tyrannosaur object if it supports the Animal interface. (The method used for this is called QueryInterface, or QI for short; you may sometimes hear QI used as a verb.) If the answer is no, an error occurs.

If the answer is yes, the object is assigned to the variable. Only the methods and properties of the Animal interface can be accessed through this variable.

Generic Object Variables and Interfaces

What happens if you assign the object reference to a generic object variable, as in the following code?

```
Private Sub Command1_Click()
   Dim ty As Tyrannosaur
   Dim anim As Animal
   Dim obj As Object

   Set ty = New Tyrannosaur
   Set anim = ty
   Set obj = anim
   MsgBox TypeName(obj)
End Sub
```

The result is again Tyrannosaur. Now, what interface do you get when you call properties and methods through the variable obj? Add the following method to the Tyrannosaur class:

```
Public Sub Growl()
```

```
    Debug.Print "Rrrrrr"
End Sub
```

The Growl method belongs to the Tyrannosaur object's default interface. In the code for the command button's Click event, replace the MsgBox statement with the following two lines of code:

```
obj.Move 42
obj.Growl
```

When you run the project and click the button, execution stops on the Growl method, with the error "Object does not support this property or method." Clearly, the interface is still Animal.

This is something to bear in mind when using variables of type Object with objects that have multiple interfaces. The interface the variable will access is the *last interface assigned*. For example:

```
Private Sub Command1_Click()
    Dim ty As Tyrannosaur
    Dim anim As Animal
    Dim obj As Object

    Set ty = New Tyrannosaur
    Set anim = ty
    Set obj = anim
    obj.Move 42      ' Succeeds
    obj.Growl        ' Fails

    Set obj = ty
    obj.Move 42      ' Fails
    obj.Growl        ' Succeeds
End Sub
```

Fortunately, there's very little reason to use the slower, late-bound Object data type with objects that have multiple interfaces. One of the main reasons for using multiple interfaces is to gain the advantage of early binding through polymorphism.

Other Sources of Interfaces

Visual Basic class modules are not your only source of interfaces to implement. You can implement any interface contained in a type library, as long as that interface supports Automation.

If you have the Professional or Enterprise Edition of Visual Basic, you can create your own type libraries of abstract classes. These type libraries can be used in many projects, as described in "General Principles of Component Design" in *Creating ActiveX Components* in the *Component Tools Guide*.

The Professional and Enterprise editions also include the MkTypLib (Make Type Library) utility in the Tools directory. If you've used this utility with Microsoft Visual C++, you may find it a more congenial way to create interfaces.

Using Interfaces in Your Project

To use an interface in your project, click References on the Project menu to open the References dialog box. If the type library is

registered, it will appear in the list of references, and you can check it. If the type library is not in the list, you can use the Browse button to locate it.
Once you have a reference to a type library, you can use Implements to implement any Automation interfaces the type library contains.

For More Information You're not limited to implementing abstract interfaces. "The Many (Inter)Faces of Code Reuse" describes how you can implement an interface and selectively reuse the properties and methods of the class that provides the interface.

5. The Many (Inter)Faces of Code Reuse

There are two main forms of code reuse — binary and source. Binary code reuse is accomplished by creating and using an object, while source code reuse is achieved by inheritance, which isn't supported by Visual Basic. (Source code reuse can also be achieved by copying and modifying the source code, but this technique is nothing new, and has many well-known problems.)

Visual Basic has been a pioneer of binary code reuse — controls being the classic example. You reuse the code in a control by placing an instance of the control on your form. This is known as a *containment* relationship or a *has-a* relationship; that is, the form *contains* or *has a* CommandButton.

For More Information Containment relationships are discussed in "Object Models" later in this chapter.

Delegating to an Implemented Object

Implements provides a powerful new means of code reuse. You can implement an abstract class (as discussed in "Creating and Implementing an Interface"), or you can implement the interface of a fully functional class. You can create the *inner object* (that is, the implemented object) in the Initialize event of the *outer object* (that is, the one that implements the inner object's interface).

As noted in "Creating and Implementing an Interface," an interface is like a contract — you must implement all the members of the inner object's interface in the outer object's class module.
However, you can be very selective in the way you delegate to the properties and methods of the inner object. In one method you might delegate directly to the inner object, passing the arguments unchanged, while in another method you might execute some code of your own before calling the inner object — and in a third method you might execute only your own code, ignoring the inner object altogether!

For example, suppose you have a OneManBand class and a Cacophony class, both of which generate sounds. You'd like to add the functionality of the Cacophony class to the OneManBand class, and reuse some of the implementation of the Cacophony class's methods.

```
' OneManBand implements the Cacophony interface.
Implements Cacophony

' Object variable to keep the reference in.
Private mcac As Cacophony
```

```
Private Sub Class_Initialize()
   ' Create the object.
   Set mcac = New Cacophony
End Sub
```

You can now go to the Object drop down and select Cacophony, and then get procedure templates for the methods of the Cacophony interface. To implement these methods, you can delegate to the Cacophony object. For example, the Beep method might look like this:

```
Private Sub Cacophony_Beep(ByVal Frequency As Double, _
ByVal Duration As Double)
   ' Delegate to the inner Cacophony object.
   Call mcac.Beep(Frequency, Duration)
End Sub
```

The implementation above is very simple. The outer object (OneManBand) delegates directly to the inner (Cacophony), reusing the Cacophony object's Beep method without any changes. This is a good thing, but it's only the beginning.

The Implements statement is a very powerful tool for code reuse, because it gives you enormous flexibility. You might decide to alter the effects of the OneManBand class's Beep method, by inserting your own code before (or after) the call to the inner Cacophony object:

```
Private Sub Cacophony_Beep(ByVal Frequency As Double, _
ByVal Duration As Double)
   ' Bump everything up an octave.
   Frequency = Frequency * 2
   ' Based on another property of the OneManBand
   ' class, Staccato, cut the duration of each beep.
   If Staccato Then Duration = Duration * 7 / 8
   Call mcac.Beep(Frequency, Duration)
   ' You can even call other methods of OneManBand.
   If Staccato Then Pause(Duration * 1 / 8)
End Sub
```

For some of the methods, your implementation may delegate directly to the inner Cacophony object, while for others you may interpose your own code before and after delegating – or even omit delegation altogether, using entirely your own code to implement a method.

Because the OneManBand class implements the Cacophony interface, you can use it with any musical application that calls that interface. Your implementation details are hidden from the calling application, but the resulting sounds are all your own.

> **Note** COM provides another mechanism for binary code reuse, called *aggregation*. In aggregation, an entire interface is reused, without any changes, and the implementation is provided by an instance of the class being aggregated. Visual Basic does not support this form of code reuse.

Doesn't This Get Tedious?

Writing delegation code can indeed become tedious, especially if most of the outer object's properties and methods simply delegate directly to the corresponding properties and methods of the inner object.

If you have the Professional or Enterprise Edition of Visual Basic, you can use the Visual Basic Extensibility model to create your own delegation wizard to automate the task, similar to the Class Wizard that's included in the Professional and Enterprise editions.

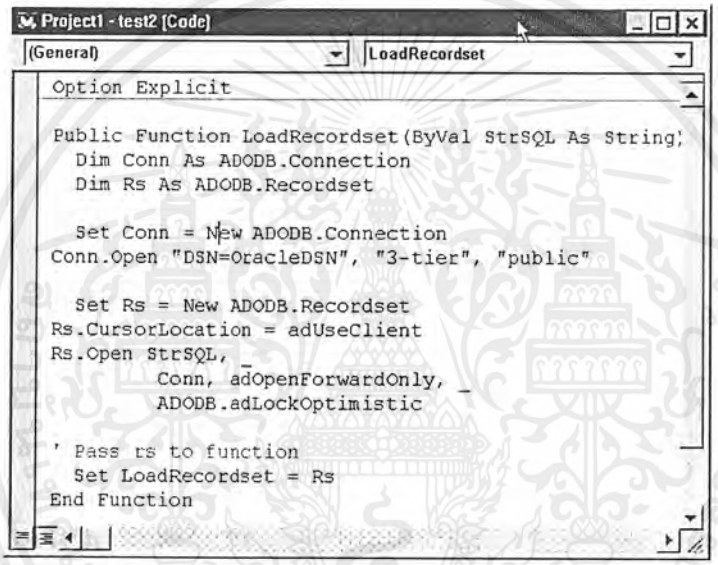
For More Information The use of polymorphism and multiple interfaces in component software is discussed in "General Principles of Component Design" in *Creating ActiveX Components* in the *Component Tools Guide*.

Using the Extensibility Model is documented in *Extending the Visual Basic Environment with Add-Ins* in the *Component Tools Guide*.

ภาคผนวก จ.
ตัวอย่างการสร้างแอปพลิเคชั่น 3-Tier
ด้วยเทคโนโลยี COM และ DCOM

1. การใช้ COM และ MTS ในการสร้างเป็นแอปพลิเคชั่นแบบ 3-tier

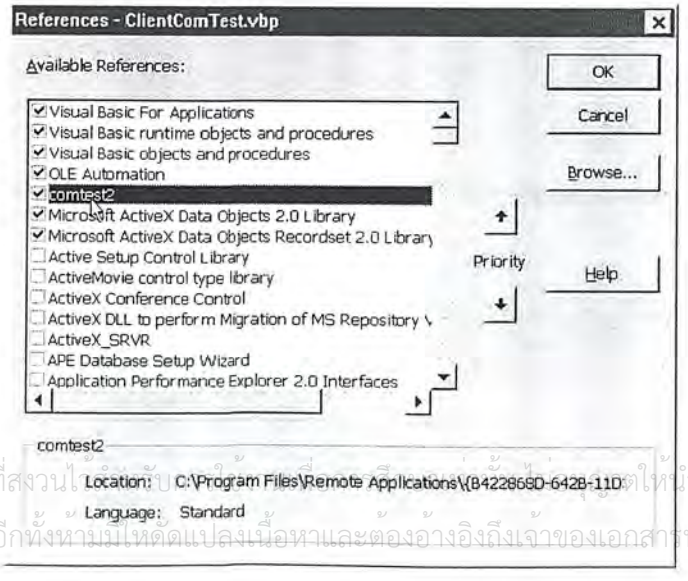
ขั้นตอนที่ 1 เขียนคอมโพเนนต์ COM ที่ทำงานฝั่งเซิร์ฟเวอร์โดยเลือกสร้างโปรเจกต์ชนิด ActiveX DLL แล้วทำการเขียนโค้ดในโมดูลของคลาสดังรูป จ-1 แล้วจากนั้นสร้างให้เป็นไฟล์ .DLL



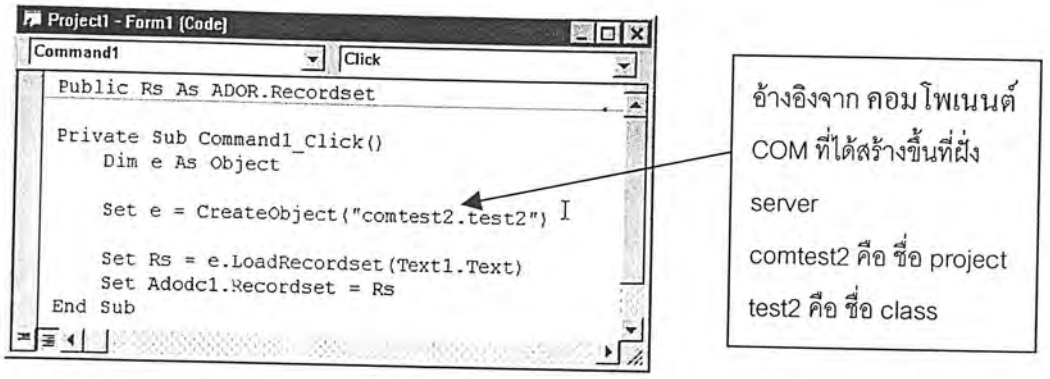
รูปที่ จ-1 แสดงโค้ดของคอมโพเนนต์ COM

ขั้นตอนที่ 2 นำเอาคอมโพเนนต์ COM ไป ติดตั้งลงบน MTS

ขั้นตอนที่ 3 ทำการเขียนฟอร์มฝั่งไคลเอนต์ โดยจะต้องอ้างอิงถึงคอมโพเนนต์ COMที่ติดตั้งไว้บน MTS ก่อนโดยเลือก Project > reference แล้วเลือกคอมโพเนนต์ COM ที่สร้างไว้ที่ฝั่งเซิร์ฟเวอร์



รูปที่ จ-2 แสดงการเลือกใช้ Reference จากคอมโพเนนต์ COM ที่สร้างไว้ฝั่งเซิร์ฟเวอร์



รูปที่ จ-3 แสดงโค้ดของแอปพลิเคชั่นฝั่งไคลเอนต์

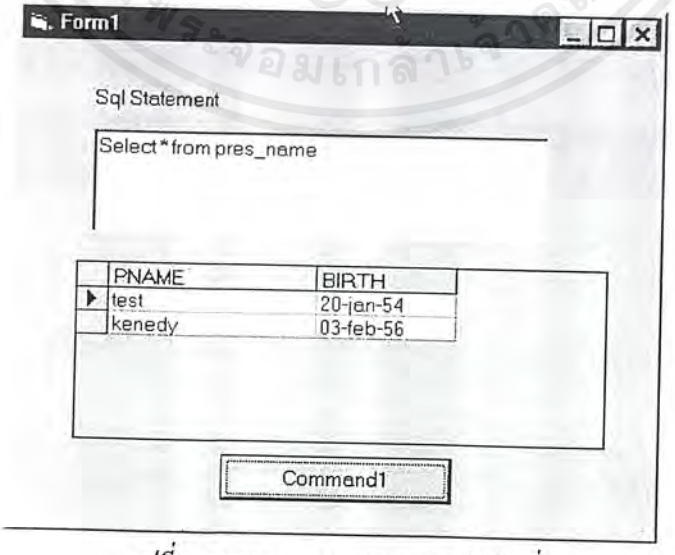
การทดลอง Run โปรแกรม

1. นำไฟล์ติดตั้งที่ได้จากการ export และ file .exe ที่คอมไพล์จากฟอร์มฝั่งไคลเอนต์ ไปทำการรันโดยจะต้องเรียกไฟล์ที่ใช้ติดตั้งก่อนเพื่อบอกให้ไคเอนต์ทราบว่าเซิร์ฟเวอร์ นั้นอยู่ที่ใด ซึ่งภายในไฟล์ติดตั้งจะประกอบด้วยไฟล์ต่างๆ ดังนี้

Name	Modified	Size
comtestpack.inf	6/9/42 14...	3,808
Comtest2.dll	25/8/42 1...	20,480
setupapi.dll	14/11/40 ...	339,728
cfgmgr32.dll	14/11/40 ...	23,552
w95inf16.dll	14/11/40 ...	2,304
w95inf32.dll	14/11/40 ...	4,608
Advpack.dll	14/11/40 ...	66,320

รูปที่ จ-4 แสดงไฟล์ที่อยู่ภายในแพคเกจไฟล์ติดตั้ง

2. ทำการทดสอบแอปพลิเคชั่นได้ผลดังนี้

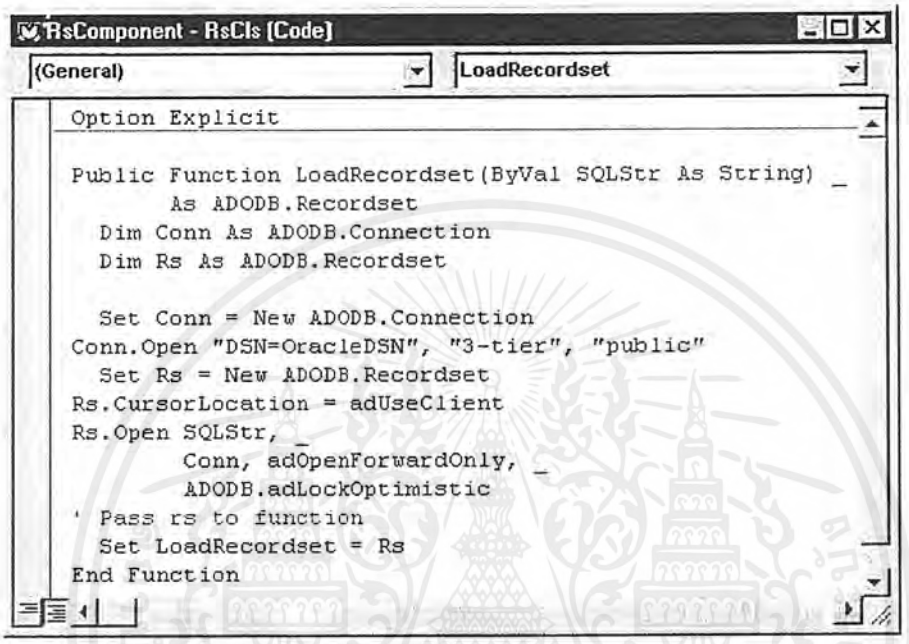


รูปที่ จ-5 แสดงผลการทดสอบแอปพลิเคชั่น

2. การใช้ DCOM ในการสร้างแอปพลิเคชั่นแบบ 3-tier

การสร้างคอมโพเนนต์ DCOM นั้นจะเหมือนกับคอมโพเนนต์ COM แต่ต่างกันที่ DCOM จะสามารถติดต่อกันข้าม Process space ได้ซึ่งจะเรียกเป็น Out-of-Process Servers ซึ่งใช้ ActiveX EXE ในการสร้างคอมโพเนนต์ DCOM ฝั่งแอปพลิเคชั่นเซิร์ฟเวอร์ทำได้ดังนี้

1. ทำการเขียนคลาสใน ActiveX EXE ดังนี้

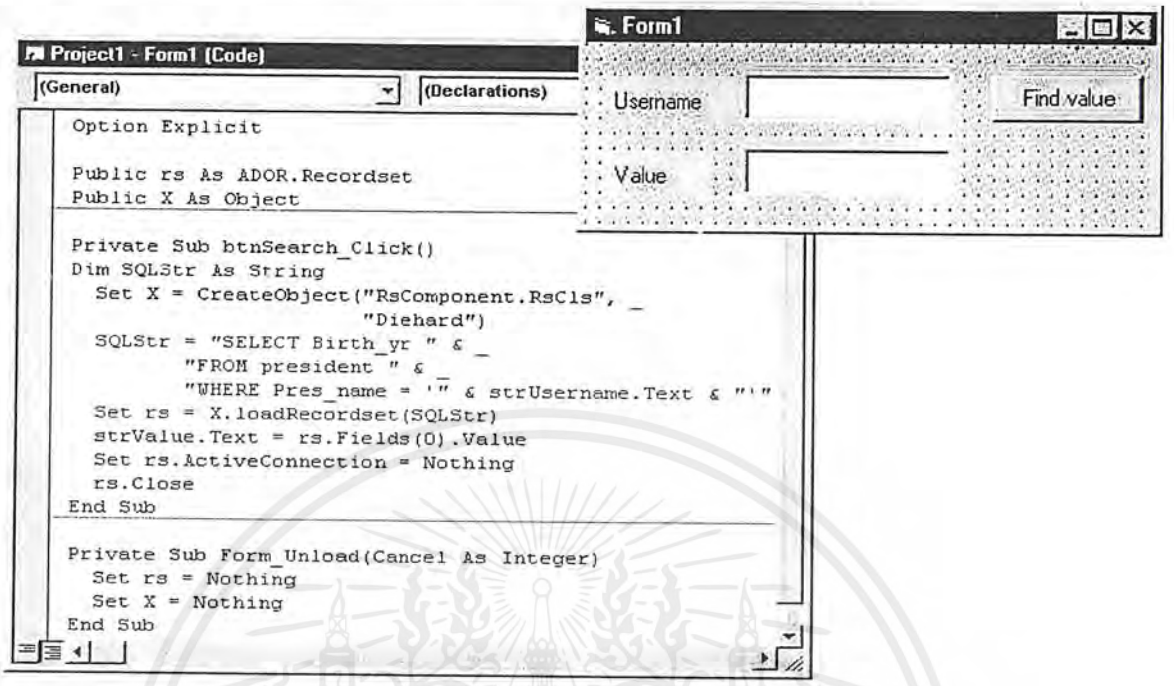


รูปที่ จ-6 แสดงการเขียนโค้ดของของคอมโพเนนต์ DCOM ฝั่งเซิร์ฟเวอร์

การติดต่อกับคลาสจะทำผ่านอินเตอร์เฟส ซึ่งจะเป็นกลุ่มของเมธอดที่ใช้ในการติดต่อกับ คลาส นี้ การทำงานของฟังก์ชัน LoadRecordset ดังรูปคือทำการรับคำสั่งภาษา SQL เข้ามาแล้วทำการส่งเรคอร์ดเซตออกไปซึ่งการกำหนดฐานข้อมูลไว้แล้วทำให้ไม่สามารถเลือกฐานข้อมูลอื่นได้เอง สาเหตุที่ทำการออกแบบเช่นนี้เพราะแนวทางการนำกลับมาใช้ใหม่ที่จะใช้นั้นจะมีขอบเขตแค่ในโปรเจกต์นี้เท่านั้น ดังนั้นจึงไม่มีความจำเป็นในการเลือกฐานข้อมูลได้เอง

เราสามารถใช้ Connection string ในการเลือกเครื่องที่เป็นเซิร์ฟเวอร์ของฐานข้อมูลได้ซึ่งในที่นี้กำหนดให้เซิร์ฟเวอร์ของแอปพลิเคชั่นและเซิร์ฟเวอร์ของฐานข้อมูลอยู่บนเครื่องเดียวกัน

2. ทำการเขียนโปรแกรมแอปพลิเคชั่นฝั่งไคลเอนต์เพื่อทำการทดสอบดังนี้



รูปที่ จ-7 แสดงการสร้างแอปพลิเคชั่นของไคลเอนต์

การเรียกใช้คอมโพเนนต์ DCOM ที่อยู่บนเซิร์ฟเวอร์ นั้นจะใช้การอ้างถึงออปเจกต์แบบ Late binding คือ ประกาศตัวแปลขึ้นมาเป็นออปเจกต์ทั่วไปก่อน (Generic Object) แล้วจึงทำการประกาศว่าจะให้เป็น ออปเจกต์ของอะไร โดยใช้การ CreateObject ซึ่งเป็นวิธีที่ใช้ในการ Create Remote Object โดยเราสามารถสร้างทำการเรียกใช้ Remote Object โดยการใส่ชื่อของเซิร์ฟเวอร์ (Argument ตัวที่สองของ CreateObject ซึ่งที่เซิร์ฟเวอร์ต้องมีคอมโพเนนต์ DCOM ที่ชื่อ RsComponent.RsCls ด้วยซึ่งสร้างขึ้นมาจาก ActiveX EXE ที่สร้างขึ้นและทำการสร้างให้เป็น Execute file และทำการ Execute ที่เซิร์ฟเวอร์ซึ่งตรวจสอบได้จากโปรแกรมคอนฟิกูเรชั่นของ DCOM (DCOMCNFG.EXE)

ภาคผนวก ฉ.

การเก็บข้อมูลมัลติมีเดีย

ในบทนี้จะทำการเสนอวิธีการในการใช้วิชวลเบสิก 6 ในการเก็บข้อมูลลักษณะมัลติมีเดียลงฐานข้อมูลออราเคิล 8 โดยความสามารถของออราเคิล 8 แล้วมีวิธีการมากมายในการเก็บข้อมูลชนิดนี้แค่ว่าวิชวลเบสิกนั้นไม่รู้จัดชนิดข้อมูลที่ออราเคิล 8 ใช้ในการเก็บข้อมูลมัลติมีเดีย ซึ่งวิธีการที่จะนำเสนอนี้ได้มาจาก http://www.microsoft.com

HOWTO: Use ADO GetChunk/AppendChunk with Oracle for BLOB Data

---

The information in this article applies to:

- ActiveX Data Objects (ADO), versions 1.5, 2.0, 2.1 SP2
- Microsoft Visual Basic Enterprise Edition for Windows, versions 5.0, 6.0

SUMMARY

The purpose of this article is to demonstrate how to save and retrieve Binary Large Object (BLOB) data to a LONG RAW datatype column in an Oracle 7.3 database using the ActiveX Data Objects (ADO) methods GetChunk and AppendChunk.

LONG RAW datatypes are used to store binary data of variable size up to 2 Gb in length. Only one LONG RAW column may be defined per table. LONG RAW columns may not be used in subqueries, functions, expressions, WHERE clauses, or indexes. A table containing a LONG RAW column may not be clustered. Only one LONG RAW column may be defined per table and you cannot have both a LONG and a LONG RAW column define in the same table.

It should be noted that it is not recommended to store BLOB data or LONG text data in a table. A more efficient way is to store file pointers in the table that locates the actual files containing the data.

MORE INFORMATION

The following project has a Picture box, CommonDialog control, and three Command buttons on the start up form. Results and status are displayed in the Debug window or the Form's Caption. You must modify the connection string to match the settings of your Oracle installation.

NOTE: For Visual Basic 5.0 users, you need to acquire and install the Microsoft Data Access Components (MDAC) for the sample in this article. Please refer to the article listed in REFERENCES section for information on installing MDAC 2.0. MDAC 2.0 contains ActiveX Data Objects (ADO) version 2.0 and the Microsoft ODBC Driver for Oracle version 2.5.

For Visual Basic 6.0 users, ADO 2.0 and the Microsoft ODBC for Oracle driver version 2.5 installs with Visual Basic 6.0.

This project uses a table called BlobTable. Following is the script used to create the table and add one row to the table:

```
CREATE TABLE BLOBTABLE (

 MYID                NUMBER(2) NOT NULL PRIMARY KEY,
 BLOBFLD                  LONG RAW
);
/
INSERT INTO BLOBTABLE (MYID) VALUES (1);
/
Commit;
```

Application Description

The Visual Basic application has a Picture box control to view the selected picture file (it defaults to .bmp or .ico files), a CommonDialog control to pick a picture file and three Command buttons to control the application flow.

The AppendChunk Command button, when clicked, brings up the Open File dialog box allowing you to select a .bmp or .ico file. The code behind the button takes that file and stores it to the BlobTable LONG RAW column using the AppendChunk method.

The GetChunk button, when clicked, retrieves the BLOB data in the LONG RAW column, converts the binary data to a BMP file and displays that file in the Picture box control. The third button is to Exit the application.

Create the application by following these steps:

1. Open a new project in Visual Basic. Form1 is created by default.

2. Place a Picture box and CommonDialog control along with three Command buttons on the new form. You may need to add the CommonDialog control to your project. On the Project menu, point to Components, and then select the Microsoft Common Dialog Control 5.0 or the 6.0 version if you are using Visual Basic 6.0.

3. On the Project menu, point to References, and then select Microsoft ActiveX Data Objects 2.x Library.

4. Place the following code in the General Declarations section of Form1:

```
' This application demonstrates using ADO with the AppendChunk
' and GetChunk methods against an Oracle 7.3 database.
'
Option Explicit
Dim Cn As ADODB.Connection
Dim Rs As ADODB.Recordset
Dim strConn As String
Dim strSQL As String
Dim FileLength As Long  'Used in Command1 and Command2
procedures.
Dim Numblocks As Integer
Dim LeftOver As Long
Dim i As Integer
Const BlockSize = 100000 'This size can be experimented with for
                         'performance and reliability.
```

```
Private Sub Form_Load()
   Command1.Caption = "AppendChunk"
   Command2.Caption = "GetChunk"
   Command3.Caption = "Exit"
   Command2.Enabled = False

   'Make Connection
   Set Cn = New ADODB.Connection
   strConn = "UID=MyUID;PWD=MyPassword;" & _
             "driver={Microsoft ODBC for Oracle};" & _
             "SERVER=MyServer;"
   Cn.Open strConn
   Debug.Print Cn.ConnectionString

End Sub

Public Sub Command1_Click()
   ' AppendChunk button
   ' This procedure prompts for a BMP file,
   ' converts that file to a Byte array,
   ' and saves the Byte Array to the table
   ' using the Appendchunk method.
   '
   Dim PictBmp As String
   Dim ByteData() As Byte    'Byte array for Blob data.
   Dim SourceFile As Integer

   ' Open the BlobTable table.
   strSQL = "Select MyID, BLOBfld from BLOBTABLE WHERE MyID = 1"
   Set Rs = New ADODB.Recordset
   Rs.CursorType = adOpenKeyset
   Rs.LockType = adLockOptimistic
   Rs.Open strSQL, Cn

   ' Retrieve the picture and update the record.
   CommonDialog1.Filter = "(*.bmp;*.ico)|*.bmp;*.ico"
   CommonDialog1.ShowOpen
   PictBmp = CommonDialog1.filename
   Me.MousePointer = vbHourglass
   Me.Caption = "Retrieving the picture"

   ' Save Picture image to the table column.
   SourceFile = FreeFile
   Open PictBmp For Binary Access Read As SourceFile

   FileLength = LOF(SourceFile)  ' Get the length of the file.
   Debug.Print "Filelength is " & FileLength

   If FileLength = 0 Then

      Close SourceFile
      MsgBox PictBmp & " empty or not found."
      Exit Sub
   Else

      Numblocks = FileLength / BlockSize
      LeftOver = FileLength Mod BlockSize

      ReDim ByteData(LeftOver)
      Get SourceFile, , ByteData()
```

```
        Rs(1).AppendChunk ByteData()

        ReDim ByteData(BlockSize)
        For i = 1 To Numblocks
            Get SourceFile, , ByteData()
            Rs(1).AppendChunk ByteData()
        Next i

        Rs.Update   'Commit the new data.

       Close SourceFile
    End If

    Me.Caption = "Picture Retrieved"
    Command2.Enabled = True
    Me.MousePointer = vbNormal
End Sub

Private Sub Command2_Click()
    ' GetChunk Button
    ' This procedure retrieves the picture image
    ' from the table using the GetChunk method,
    ' converts the data to a file and
    ' displays that file in the Picture box.
    '
    Dim ByteData() As Byte    'Byte array for picture file.
    Dim DestFileNum As Integer
    Dim DiskFile As String

    Me.MousePointer = vbHourglass
    Me.Caption = "Creating Picture File"

    ' Remove any existing destination file.
    DiskFile = App.Path & "\image1.bmp"
    If Len(Dir$(DiskFile)) > 0 Then
       Kill DiskFile
    End If

    DestFileNum = FreeFile
    Open DiskFile For Binary As DestFileNum

    Numblocks = FileLength / BlockSize
    LeftOver = FileLength Mod BlockSize

    ByteData() = Rs(1).GetChunk(LeftOver)
    Put DestFileNum, , ByteData()

    For i = 1 To Numblocks
        ByteData() = Rs(1).GetChunk(BlockSize)
        Put DestFileNum, , ByteData()
    Next i

    Close DestFileNum

    Picture1.Visible = True
    Picture1.Picture = LoadPicture(App.Path & "\image1.bmp")
    Rs.Close
    Debug.Print "Complete"
    Me.Caption = "Success!"
    Me.MousePointer = vbNormal
End Sub
```

```
Private Sub Command3_Click()
'Exit button.
Cn.Close   .
Unload Me
End Sub
```

Run the Project and click the AppendChunk button. Change the directory to pick a .bmp or .ico file. Click the file of choice and wait for the GetChunk button to be enabled. After the GetChunk button is enabled, click it and you should see the picture you selected appear in the Picture box control.

The Debug window will have the size of the file you selected along with the ADO connect string.

REFERENCES

For more information, please see the following articles in the Microsoft Knowledge Base:

Q192743 HOWTO: Use ADO GetChunk/AppendChunk with Oracle for TEXT Data

Q175018 HOWTO: Acquire and Install the Microsoft Oracle ODBC Driver

Additional query words:

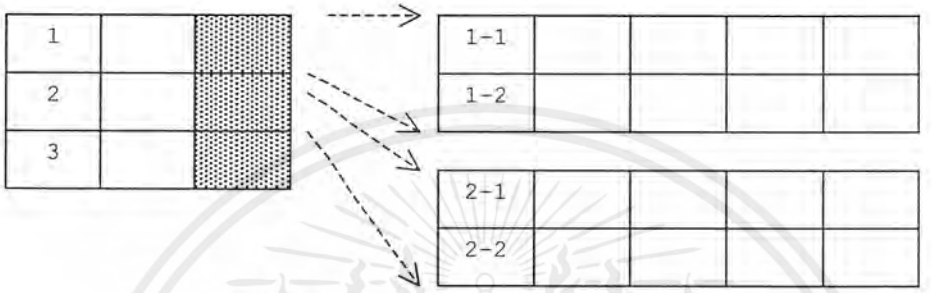
```
Keywords          : kbADO150 kbADO200 kbDatabase kbVBp500 kbVBp600
kbGrpVBDB kbGrpMDAC kbDSupport kbADO210sp2 kbMDAC210SP2
Version           : WINDOWS:1.5,2.0,2.1 SP2,5.0,6.0
Platform          : WINDOWS
Issue type        : kbhowto
```

Last Reviewed: November 5, 1999

ภาคผนวก ช.
การใช้งาน SQL ในการติดต่อกับตารางลักษณะเนสเต็ด

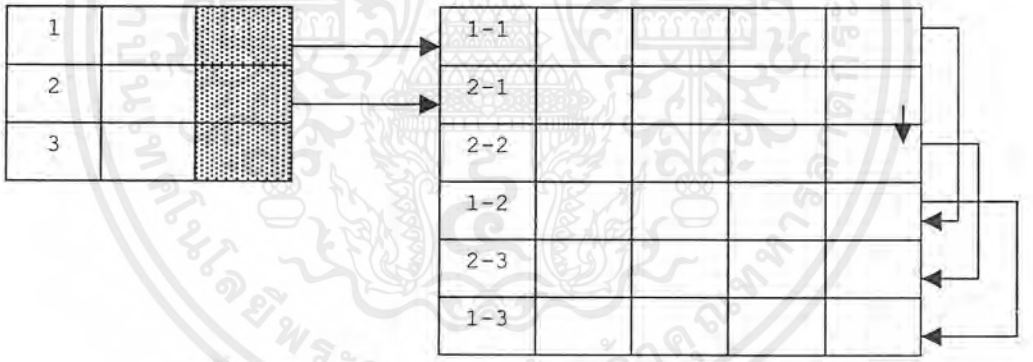
ฉ.1 ภาพรวมของตารางชนิดเนสเต็ด

โครงสร้างของตารางในลักษณะเนสเต็ดตามแนวความคิดจะเป็นดังนี้



รูปที่ ช-1 ลักษณะโครงสร้างของตารางชนิดเนสเต็ด

แต่จากการทดลองแล้วจะได้ลักษณะตารางแบบเนสเต็ดเป็นดังนี้



รูปที่ ช-2 ลักษณะโครงสร้างของตารางชนิดเนสเต็ดที่เป็นจริง

จากรูปที่ 2 นี้ทำให้เรามองเห็นได้ว่าถ้าเราหาข้อมูลจากตารางย่อยก่อนแล้วค้นหาว่ามาจากตารางใหญ่แถวใด จะไม่สามารถทำการค้นหาได้ และถ้าเราต้องการรู้ความสัมพันธ์ระหว่างตารางย่อยกับตารางใหญ่เรามีวิธีเดียวคือ ทำการค้นหาจากตารางใหญ่แล้วทำการค้นหาข้อมูลจากตารางย่อยอีกครั้งหนึ่งซึ่งเรียกได้ว่าตารางชนิดเนสเต็ดนี้สนับสนุนความสัมพันธ์แบบ One-To-Many และเป็นแบบค้นหาข้อมูลทางเดียวเท่านั้น

ฉ.2 การสร้างตารางแบบเนสเต็ด

ขั้นตอนการสร้างจะถูกอธิบายโดยตัวอย่าง ตารางที่ใช้ในการเก็บข้อมูลของนักศึกษากับวิชาที่นักศึกษาเรียน โดยเราให้ตารางใหญ่เก็บข้อมูลนักศึกษาและตารางย่อยเก็บข้อมูลของวิชาที่เขาเรียนโดยรูปแบบนี้จะมีความแตกต่างจากรูปแบบที่เสนอไว้ในบทที่ 5 ซึ่งทั้งสองรูปแบบนี้สามารถใช้ได้เหมือนกัน

- สร้าง Object Type ของข้อมูลที่อยู่ในตารางย่อยขึ้นมาก่อน

```
CREATE OR REPLACE TYPE Subject_O AS OBJECT (
  name varchar2(30),
  prof varchar2(30) );
/
```

- สร้าง Table Type ของ Type ที่ได้จากขั้นตอนก่อน

```
CREATE OR REPLACE TYPE Subject_T AS TABLE OF Subject_O;
/
```

- สร้าง Object Type ของข้อมูลที่อยู่ในตารางหลักโดยข้อมูลที่เป็น Nested นั้นให้ประกาศใช้ Table Type ที่สร้างจากขั้นตอนที่แล้ว

```
CREATE OR REPLACE TYPE Student AS OBJECT (
  id number(4),
  name varchar2(30),
  subject subject_t );
/
```

- สุดท้ายทำการสร้างตารางจาก Object Type จากขั้นตอนที่แล้ว

```
CREATE TABLE TblStudent OF Student
NESTED TABLE subject Store AS NSubject;
```

Note โดยการสร้าง Type หรือ Stored Procedure ต้องใส่ ("/") ไว้หลังสุดด้วยซึ่งในการสร้างตารางไม่ต้องใช้

ฉ.3 การจัดการข้อมูลในตารางที่ประกอบด้วยตารางชนิดเนสเต็ด

การใส่ข้อมูลในตารางใหญ่ในแต่ละแถวทุกครั้ง ต้องทำการอ้างถึงตารางที่เป็นเนสเต็ดทุกครั้งด้วย ซึ่งเหมือนเป็นการสร้างความสัมพันธ์ระหว่างตารางใหญ่กับตารางที่เป็นเนสเต็ด โดยใช้คำสั่งนี้

```
Insert into TblStudent
Values (1, 'niran', Subject_T(Subject_O('OS', 'Prof B')));
```

และเราสามารถใส่ข้อมูลลงในตารางที่เป็นเนสเต็ดโดยอ้างจากตารางใหญ่เข้าไปดังนี้

```
Insert into the(Select subject From tblstudent where id = 1)
values ('Database', 'Prof S');
```

และต่อจากนี้เราสามารถทำการใส่ข้อมูลลงตารางที่เป็นเนสเต็ดเท่าไรก็ได้โดยใช้คำสั่งนี้เช่นกัน และในสำหรับการเรียกดูข้อมูลทั้งหมดทั้งในตารางใหญ่และตารางย่อยจะมีรูปแบบดังนี้

```
SELECT * FROM TBLSTUDENT;
Result:
 ID  NAME          SUBJECT(NAME, PROF)
---  ------------  -----------------------------------
 1   niran         SUBJECT_T(SUBJECT_O('OS', 'Prof B'),
                   SUBJECT_O('Database', 'Prof S'))
```

หรือใช้คำสั่งนี้

```
Select T1.id, T1.name, Cursor (
       Select *
       From Table(T1.Subject))
From tblstudent T1;
```

Result:

```
ID NAME        CURSOR(SELECT*FROMTA
---- ---------- --------------------
   1 niran      CURSOR STATEMENT : 3

CURSOR STATEMENT : 3

NAME         PROF
------------ -----------
OS           Prof B
Database     Prof
```

ส่วนการเรียกดูเฉพาะข้อมูลในตารางย่อยทำได้ดังนี้

```
Select * From The(Select subject From tblStudent where id=1);
```

ฉ.4 การจัดการข้อมูลที่อยู่ในตารางที่มีตารางชนิดเนสเต็ดอยู่ภายในโดยวิชวลเบสิก

วิชวลเบสิกนั้นไม่สามารถเรียกใช้ หรือเข้าใน Object Type ที่มีของออราเคิลได้ ดังนั้นการที่เราได้ชุดของข้อมูลออกมาในลักษณะ

```
ID  NAME          SUBJECT(NAME, PROF)
--- ------------  -----------------------------------
 1   niran        SUBJECT_T(SUBJECT_O('OS', 'Prof B'),
                  SUBJECT_O('Database', 'Prof S'))
```

หรือ

```
ID NAME        CURSOR(SELECT*FROMTA
---- ---------- --------------------
   1 niran      CURSOR STATEMENT : 3

CURSOR STATEMENT : 3

NAME         PROF
------------ -----------
OS           Prof B
Database     Prof
```

จึงไม่สามารถทำได้โดยสิ่งที่วิชวลเบสิกเข้าใจคือเซ็ตของข้อมูลในลักษณะปกติไม่มี Repeating Group ทำให้การเรียกดูข้อมูลจากตารางที่เป็นลักษณะเนสเต็ดนี้ถูกจำกัดลง

- การเรียกดูข้อมูลจากตารางใหญ่ต้องเรียกดูสดมภ์ที่ไม่มีชนิดข้อมูลเป็นเนสเต็ด ซึ่งใช้ View เข้ามาช่วยได้
- การเรียกดูข้อมูลที่อยู่ในตารางเนสเต็ดใช้คำสั่งรูปแบบนี้ได้อย่างเดียว

```
Select * From The(Select subject From tblStudent where id=1);
```

นอกจากการเรียกดูข้อมูลแล้วการ Insert, Delete, Update นั้นก็มีปัญหาด้วยโดยเราไม่สามารถทำผ่าน Method ของ ADO Object ได้ถึงแม้ว่าเราจะเป็น Recordset เป็นแบบที่เราได้ทำการแก้ไขในข้างต้นแล้วก็ตาม (ในการทำงานกับตารางใหญ่ยังคงทำงานได้เมื่อใช้ View แต่มีปัญหาด้านการ Insert) ซึ่งเรามีทางแก้ไขด้วยการส่งคำสั่ง SQL ที่ใช้ในการ Insert, Delete, Update ไปทาง Method Open ของ Recordset ซึ่งมีรูปแบบคำสั่ง SQL ดังนี้

- Insert ที่ตารางใหญ่เราใช้ Stored Procedure ช่วยมีรูปแบบการเขียนดังนี้

```
Create Or Replace Procedure InitStudent (ID IN Number) IS
Begin
  Insert Into TblStudent (ID, Subject)
  Values (ID, Subject_T(Subject_O(NULL, NULL)));
End;
/
```

และใช้ ADO ในการติดต่อดังนี้

```
Dim cn As ADODB.Connection
Dim rs As ADODB.Recordset
Dim Qy As New ADODB.Command
Dim IntShopID As Integer

Set cn = New ADODB.Connection
cn.ConnectionString = "PROVIDER=MSDASQL;DSN=OracleDSN;UID=Ecomm;PWD=Public;"
cn.CursorLocation = adUseClient
cn.Open
Set rs = New ADODB.Recordset

    Set Qy.ActiveConnection = cn
    Qy.CommandType = adCmdStoredProc
    Qy.CommandText = "InitStudent"
     ' -- Set parameter for SP InitStudent --
    Qy(0) = IntShopID
    Set rs = Qy.Execute()
```

- Insert ข้อมูลของตารางย่อย

```
Insert into the(Select subject From tblstudent where id = 1)
values ('Database', 'Prof S');
```

- Delete ข้อมูลในตารางย่อย

```
Delete the(Select subject From tblstudent where id = 1)
Where lower(name)=lower('Database');
```

- Update ข้อมูลในตารางย่อย

```
Update The (Select subject From TblStudent Where ID=1)
Set Name = 'mis',
    Prof = 'Prof P'
Where Upper(name) = Upper('mis');
```

ภาคผนวก ฌ

เทคนิคและคำแนะนำในการพัฒนาระบบ

1. การเปิดเรคคอร์ดเซตเพื่อทำการ Insert ข้อมูลนั้นต้องมีการกำหนดพรอพเพอร์ตีดังนี้
   - CursorLocation ของคอนเนคชันออบเจ็กต์หรือของเรคคอร์เซตออบเจ็กต์ ให้เป็น adUseClient
   - CursorType ของเรคคอร์ดเซตออบเจ็กต์ให้เป็น adOpenKeyset
   - LockType ของเรคคอร์ดเซตออบเจ็กต์ให้เป็น adLockOptimistic

   ซึ่งถ้าเราไม่กำหนดค่า Default จะเป็นการเปิดแบบ Read-Only

2. การเลือกใช้คอนเนคชันออบเจ็กต์หรือเรคคอร์ดเซตออบเจ็กต์ในการเปิดคอนเนคชัน
   - ถ้าเราใช้คอนเนคชันออบเจ็กต์เราสามารถ Drop Connection นั้นเพื่อลด Network Traffic ได้หรือสามารถเปิดเรคคอร์ดเซตออบเจ็กต์ได้มากกว่า 1 ตัวภายใต้คอนเนคชันออบเจ็กต์นั้น ซึ่งมีรูปแบบดังนี้

```
Set cn = New ADODB.Connection
cn.ConnectionString = StrConn  ' Public const StrConn in Module shop
cn.CursorLocation = adUseClient
cn.Open

Set rs = New ADODB.Recordset
rs.CursorType = adOpenKeyset      ' Open for insert
rs.LockType = adLockOptimistic
rs.Open "Select * From VShop Where ShopID = " & CStr(mvarShopID), cn
```

   - สำหรับการเปิดคอนเนคชันโดยใช้เรคคอร์ดเซตออบเจ็กต์มีดังนี้

```
Set rs = New ADODB.Recordset
rs.CursorLocation = adUseClient
rs.CursorType = adOpenKeyset
rs.LockType = adLockReadOnly              ' StrConn in Module Shop
rs.Open "Select * From VShop, StrConn, , , adCmdText
```

3. การเรียกใช้ทรานแซคชันถ้าต้องการให้มีทรานแซคชันปรากฏใน Transaction Statistics ต้องทำการกำหนดพรอพเพอร์ตี MTSTransactionMode เป็น RequiresTransaction และทำการติดตั้งคอมโพเนนต์นั้นลงใน MTS แล้วการกำหนดให้เกิดการ Abort นั้นต้องทำก่อนที่จะเกิด Err.Raise ดังนี้

```
tx.SetAbort
Err.Raise Err.Number, , Err.Description
```

4. การตั้ง Username ในออราเคิลนั้นไม่ควรขึ้นต้นด้วยตัวเลข เช่น 3-Tier นั้นไม่ควรตั้งเพราะจะมีปัญหาในการสร้าง Type

5. ใช้ ODBC ของ Microsoft ODBC for Oracle Version 2.573 ไม่ควรใช้ Oracle ODBC Driver Version 8.00.05

6. Oracle Enterprise Server นั้นมี Tools ต่างๆ ของ Oracle Client รวมอยู่ด้วยแล้วไม่จำเป็นต้องลง Oracle Client ถ้าต้องการพัฒนาโปรแกรมบน Oracle Enterprise Server ซึ่งในกรณีนี้โปรแกรมที่เขียนขึ้นเพื่อทำการติดต่อนั้นไม่ต้องใช้ Service ของ Oracle ซึ่งมีไว้ระบุ IP ของเครื่อง Server ที่ต้องการติดต่อด้วย ซึ่งกำหนดได้จาก Net8 Easy Config โดยการกำหนดและเรียกใช้ Service นี้ทำได้โดยเขียนไว้ใน Connection String ดังนี้

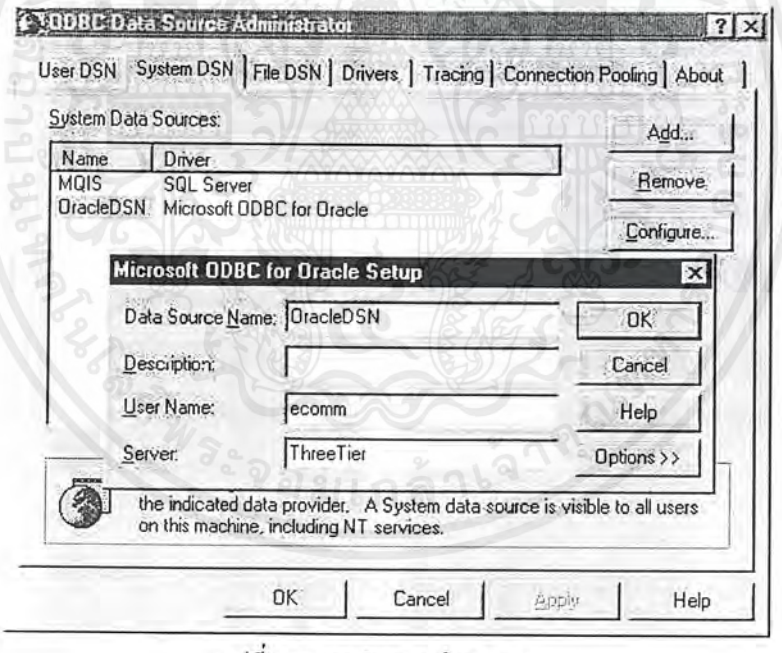
```
PROVIDER=MSDASQL;Driver=Microsoft ODBC for Oracle;Server=ThreeTier;
UID=Ecomm;PWD=public;
```

หรือ

```
PROVIDER=MSDASQL;DSN=OracleDSN;UID=boy;PWD=boy;Server=ThreeTier;
```

หรือ

ใช้คอนเนคชันสตริงเหมือนเดิมแต่ทำการกำหนดใน System DSN ดังนี้



รูปที่ ฌ-1 การกำหนดค่า ODBC

โดยชื่อ ThreeTier เราได้กำหนดใน Net8 Easy Config

บรรณานุกรม

หนังสืออ้างอิง

[1] Hans-Erik Eriksson, Magnus Penker, *"UML Toolkit"*. (Wiley Computer Publishing), 1998

[2] Ash Rofail, Tony Martin, *"Building N-Tier Applications with COM and Visual Basic 6.0"*. (Wiley Computer Publishing), 1999

[3] Rob Thayer, "*Visual Basic 6 Unleashed*", First Edition, Sams Publishing, 1998.

[4] Guy Eddon and Henry Eddon, "*Programming Components with Microsoft Visual Basic 6.0*", Microsoft Press, 1998.

[5] สัจจะ จรัสรุ่งรวีวร, "*คู่มือการสร้างแอปพลิเคชั่นด้วย Visual Basic 6 Basic & Advance*", Infopress, 2542

[5] สัจจะ จรัสรุ่งรวีวร, "*คู่มือการสร้างแอปพลิเคชั่นด้วย Delphi 4*", Infopress, 2542

[7] ธาริน สิทธิธรรมชารี, "*คู่มือการเขียนโปรแกรม Microsoft Visual Basic version 6.0*", สำนักพิมพ์ ซัคเซส มีเดีย

[8] Bruce Powel Douglass, "*Real-Time UML: Developing Efficient Objects for Embedded Systems*", Addison-Wesley, 1998

[10] John Clark Craig, Jeff Webb, เรียบเรียงโดย ชัชวาล ศุภเกษม, "*การเขียนโปรแกรมบนวินโดวส์ด้วย Microsoft Visual Basic 6.0 ภาคปฏิบัติ*", บริษัท ซีเอ็ดยูเคชั่น จำกัด (มหาชน), 2542

[11] Willian G. Page Jr. and Nathan Hughes, *"Using Oracle 8"*, QUE, 1998

[12] ผศ.ดร. ศุภมิตร จิตตะยโศธร, *"เอกสารประกอบการอบรมสัมมนาหลักสูตร Distributed Database Systems"*

[13] นางสาวจันทรกานต์ จุหอม, นางสาวทิพย์เนตร แก้วปัตตะ, "*การพัฒนาโปรแกรมประยุกต์ในเชิงวัตถุ*"

เว็บไซต์อ้างอิง

[14] http://www.oracle.com

[15] http://www.msdn.microsoft.com