การพัฒนาระบบงานประยุกต์บนระบบฐานข้อมูลแบบไคลเอ็นท์เซิร์ฟเวอร์

CLIENT-SERVER DATABASE APPLICATION DEVELOPMENT

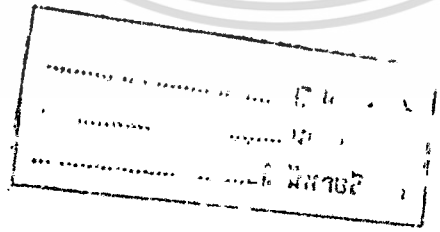
โดย

นางสาวพัชภรณ์ จามิกรณ์ 34104236

นางสาวมินตรา เปี่ยมกุลวนิช 34105272

อาจารย์ที่ปรึกษา

ผศ.ดร.ศุภมิตร จิตตะยโศธร



ปริญญานิพนธ์สำหรับปริญญาวิศวกรรมศาสตร์บัณฑิต

สาขาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2537

ปริญญานิพนธ์ปีการศึกษา 2537

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การพัฒนาระบบงานประยุกต์บนระบบฐานข้อมูลแบบไคลเอ็นท์เซิร์ฟเวอร์

ผู้จัดทำ

1. นางสาวพัชภรณ์ จามิกรณ์ 34104236
2. นางสาวมินตรา เปี่ยมกุลวนิช 34105272

อาจารย์ที่ปรึกษา

(ผศ.ดร.ศุภมิตร จิตตะยโศธร)

การพัฒนาระบบงานประยุกต์บนระบบฐานข้อมูลแบบไคลเอ็นท์เซิร์ฟเวอร์

พัชภรณ์ จามิกรณ์
มินตรา เปี่ยมกุลวนิช
ผศ.ดร.ศุภมิตร จิตตะยโศธร อาจารย์ที่ปรึกษา
ปีการศึกษา 2537

บทคัดย่อ

ในปริญญานิพนธ์ฉบับนี้เรียบเรียงขึ้นจากการศึกษาวิธีการออกแบบระบบงานประยุกต์ โดยได้เลือกการออกแบบที่สนับสนุนวิธีออบเจ็กต์โอเรียนเต็ด เพราะได้ทดลองและเปรียบเทียบข้อดีและข้อเสียกับวิธีการออกแบบที่ใช้กันในสมัยเก่าและพบว่าวิธีออบเจ็กต์โอเรียนเต็ดนั้นมีข้อดีหลายประการ จึงได้เลือกวิธีออกแบบวิธีนี้ สำหรับผลิตภัณฑ์ที่นำมาเป็นเครื่องมือในการพัฒนาระบบงานประยุกต์ในครั้งนี้ ได้ใช้ผลิตภัณฑ์ SQLWindows ของบริษัทกุปตาเทคโนโลยีและ Visual Basic ของไมโครซอฟต์ นอกจากนั้นยังได้ศึกษาถึงหลักการของไคลเอ็นท์เซิร์ฟเวอร์ โดยได้ใช้ SQLWindows เขียนงานประยุกต์ติดต่อกับ SQLBase ซึ่งเป็นตัวจัดการระบบฐานข้อมูลแบบสัมพันธ์ และได้ใช้ Visual Basic เขียนงานประยุกต์ติดต่อกับ Microsoft Access และ Windows NT แล้วทำการเปรียบเทียบระหว่างผลิตภัณฑ์ทั้งสอง

CLIENT-SERVER DATABASE APPLICATION DEVELOPMENT

Patchaporn Jamikorn

Mintra Piamkulvanich

Prof.Ass.Dr.Suppamit Chittayasothorn

1994

Abstract

This thesis is a client-sever database application development. First is studying a methodology to design application based on object oriented concept and based on functional. And then to compare together. The result from studying and test showed that methodology that based on object oriented is better than one. So that we chose the better way for design our application. Besides that we studied concept about client-server and to program with 2 products, one is SQLWindows from Gupta and the other one is Visual Basic from Microsoft. We used SQLWindows to interface with SQLBase (is DBMS) and used Visual Basic to interface with Access and Windows NT. Then compared some features between them.

สารบัญ

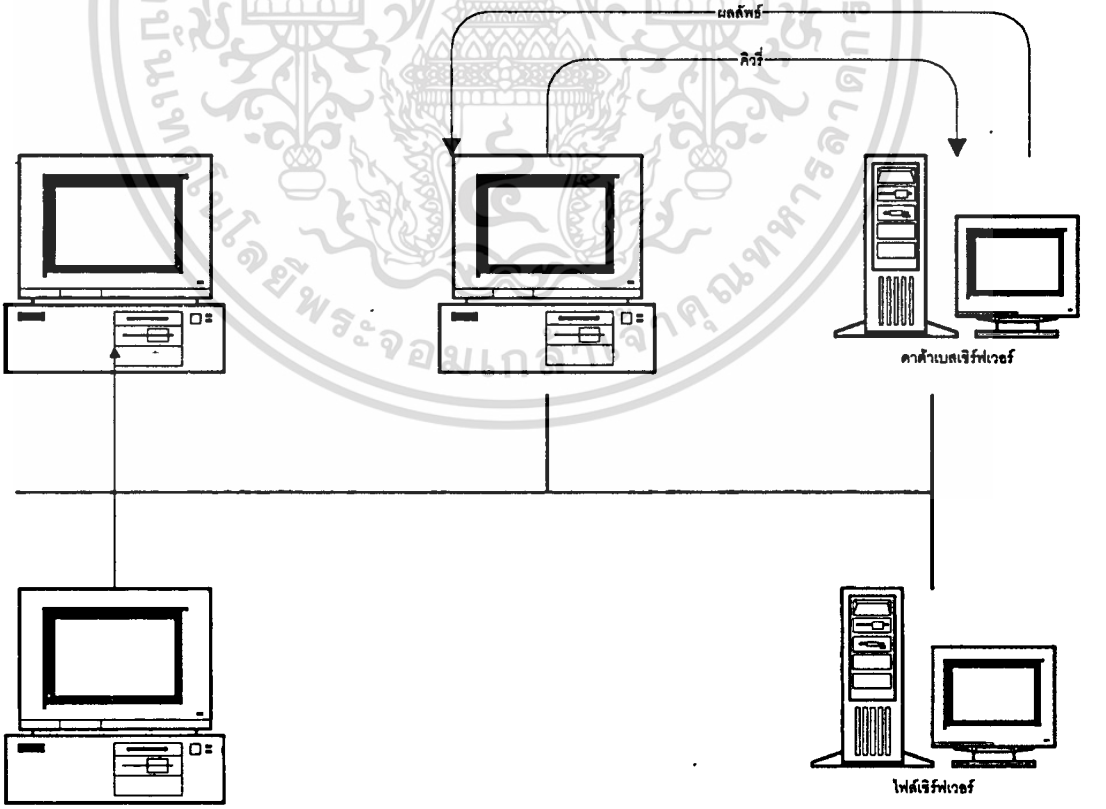
สารบัญรูปภาพ

สารบัญตาราง

บทที่ 2
ทฤษฎีและหลักการ

2.1 หลักการของไคลเอ็นท์เซิร์ฟเวอร์

ไคลเอ็นท์เซิร์ฟเวอร์ เป็นโครงสร้างของระบบคอมพิวเตอร์รูปแบบหนึ่งที่แบ่งแยกการประมวลผลข้อมูลออกเป็น 2 ระบบ โดยฝั่งไคลเอ็นท์ (ผู้ขอใช้บริการ) จะมีระบบฟรอนท์เอ็น (FRONT END SYSTEM) หรือส่วนดาต้าเบสแอพพลิเคชันทำงานอยู่และฝั่งดาต้าเบสเซิร์ฟเวอร์ (ผู้ให้บริการ) จะมีระบบแบคเอ็น (BACK-END SYSTEM) หรือส่วนที่เป็นดีบีเอ็มเอสจริงๆทำงานอยู่ ซึ่งระบบฟรอนท์เอ็นนี้จะจัดการการประมวลผลเกี่ยวกับหน้าจอและอินพุทเอาท์พุทของผู้ใช้และระบบแบคเอ็นจะจัดการการประมวลผลข้อมูลและการทำดิสค์แอคเซส (DISK ACCESS) เช่นเมื่อผู้ใช้บนระบบฟรอนท์เอ็นสร้างคิวรี่(QUERY) เพื่อสอบถามข้อมูลจากดาต้าเบสเซิร์ฟเวอร์ ส่วนฟรอนท์เอ็น (FRONT-END) แอพพลิเคชันจะส่งการร้องขอให้เซิร์ฟเวอร์โดยผ่านระบบเครือข่าย ส่วนเซิร์ฟเวอร์ก็จะทำการค้นหาข้อมูลที่ผู้ใช้ต้องการแล้วส่งข้อมูลกลับไปให้ดังรูปที่ 2.1



รูปที่ 2.1 ระบบไคลเอ็นท์เซิร์ฟเวอร์

วัตถุประสงค์หลัก ของระบบไคลเอ็นท์เซิร์ฟเวอร์คือการอนุญาตให้แอพพลิเคชันของผู้ขอใช้บริการเข้ามาเรียกใช้ข้อมูลที่ถูกจัดการโดยผู้ให้บริการได้ โดยผู้ให้บริการสามารถรันอยู่ในเครื่องที่ตั้งอยู่ในที่ห่างไกลกับเครื่องที่ผู้ขอใช้บริการรันอยู่

โดยทั่วไปแล้วระบบไคลเอ็นท์จะถูกใช้ทำงานกับพีซี และส่วนดาต้าเบสเซิร์ฟเวอร์สามารถทำงานบนเครื่องใดก็ได้ตั้งแต่พีซีไปจนถึงเมนเฟรม

2.1.1 ชนิดของการประมวลผลไคลเอ็นท์เซิร์ฟเวอร์

รูปแบบของแอพพลิเคชันของระบบไคลเอ็นท์เซิร์ฟเวอร์แบ่งได้เป็น 6 ประเภท โดยมีรายละเอียดดังนี้

2.1.1.1 ไคลเอ็นท์เซิร์ฟเวอร์ที่ทำงานบนเครื่องเดียวกัน (STAND-ALONE CLIENT-SERVER)

แอพพลิเคชันประเภทนี้จะมีผู้ขอใช้บริการประมวลผลอยู่บนเครื่องเดียวกับที่ให้บริการทำการประมวลผลอยู่ในรูปที่ 2.2 ลักษณะการทำงานเช่นนี้จะเป็นการบั่นทอนประสิทธิภาพ การประมวลผลสำหรับระบบจัดการฐานข้อมูลลงบ้าง แต่ความเร็วในการสื่อสารระหว่างผู้ขอใช้บริการกับผู้ให้บริการจะสูงมาก ผู้ให้บริการจะยังสามารถที่จะทำงานได้โดยการประมวลผลร่วมกับแอพพลิเคชันอื่น ๆ ของผู้ขอใช้บริการ ในกรณีที่มีผู้ขอใช้บริการและผู้ให้บริการหลาย ๆ ตัวรันอยู่บนฮาร์ดแวร์แพลตฟอร์มเดียวกัน การใช้มัลติโปรเซสเซอร์อาจจะช่วยเพิ่มประสิทธิภาพการทำงานขึ้นได้ แต่ว่าจะไม่สามารถนำเอาเทคโนโลยีด้านการประมวลผลแบบกระจายหรือการประมวลผลฐานข้อมูลแบบกระจายมาใช้ในกรณีนี้ได้เลย

2.1.1.2 แสตนอโลนแลนไคลเอ็นท์เซิร์ฟเวอร์ (STAND-ALONE LAN CLIENT-SERVER)

ระบบไคลเอ็นท์เซิร์ฟเวอร์แบบนี้จะเป็นรูปแบบของไคลเอ็นท์เซิร์ฟเวอร์ในวงแลนวงหนึ่ง มีการทำงานของผู้ขอใช้บริการแต่ละตัวอาจจะรับผิดชอบงานด้านการนำเสนอข้อมูลประมวลผลธุรกิจและลอจิกทางด้านฐานข้อมูลในขณะที่ผู้ให้บริการจะรับผิดชอบในเรื่องของการเรียกใช้ข้อมูลสำหรับผู้ขอใช้บริการภายในวงแลน ข้อเสียของระบบนี้เมื่อเทียบกับระบบนี้ในรูปที่ 2.3 คือการสื่อสารระหว่างผู้ขอใช้บริการกับผู้ให้บริการที่ทำโดยผ่านการเชื่อมต่อของแลนจะช้ากว่าการใช้หน่วยความจำร่วมกันของระบบที่ 1 มาก

2.1.1.3 แมนนวลเอ็กแทรกต์ไคลเอ็นท์เซิร์ฟเวอร์ (MANUAL EXTRACT CLIENT-SERVER)

ในรูปที่ 2.4 จะแสดงให้เห็นถึงรูปแบบของแอพพลิเคชันไคลเอ็นท์เซิร์ฟเวอร์ ที่การประมวลผลกระทำได้โดยเรียกใช้ข้อมูลบางส่วนของทั้งหมดที่ได้ทำการย้ายไปเก็บไว้ในเครื่องของผู้ขอใช้บริการ ข้อมูลส่วนนี้ถูกสร้างขึ้นด้วยวิธีการกระจายข้อมูลแบบแมนนวลเอ็กแทรกต์ ลักษณะการทำงานของแอพ

พลิเคชันสามารถเกิดขึ้นโดยผู้ใช้ส่งคำสั่งไปยังผู้ให้บริการเพื่อเรียกใช้ข้อมูล ซึ่งในกรณีนี้มักจะถูกกำหนดให้ทำการอ่านอย่างเดียว การคัดข้อมูลและทำการย้ายนั้นเป็นสิ่งที่สำคัญและจำเป็นต้องทำ เพราะว่าโดยปกติแล้วข้อมูลทั้งหมดมักจะไม่ได้อยู่ในรูปแบบที่ผู้ใช้ต้องการ ตัวอย่างเช่น ผู้ใช้อาจจะต้องการดูข้อมูลสรุป หรือข้อมูลที่ได้ทำการรวบรวมแล้วมากกว่าที่จะดูข้อมูลโดยละเอียด การรวบรวมข้อมูล หรือทำสรุปจะกระทำที่เครื่องของผู้ขอใช้บริการ ข้อเสียอย่างหนึ่งที่อาจจะเกิดขึ้นคือข้อมูลในส่วนที่เก็บอยู่ที่เครื่องของผู้ขอใช้บริการอาจจะไม่ถูกต้องตรงกับความเป็นจริง ถ้าข้อมูลส่วนดังกล่าวกำลังถูกเรียกใช้โดยผู้ขอใช้บริการ และในขณะเดียวกันก็กำลังถูกเปลี่ยนแปลงที่ผู้ให้บริการด้วย

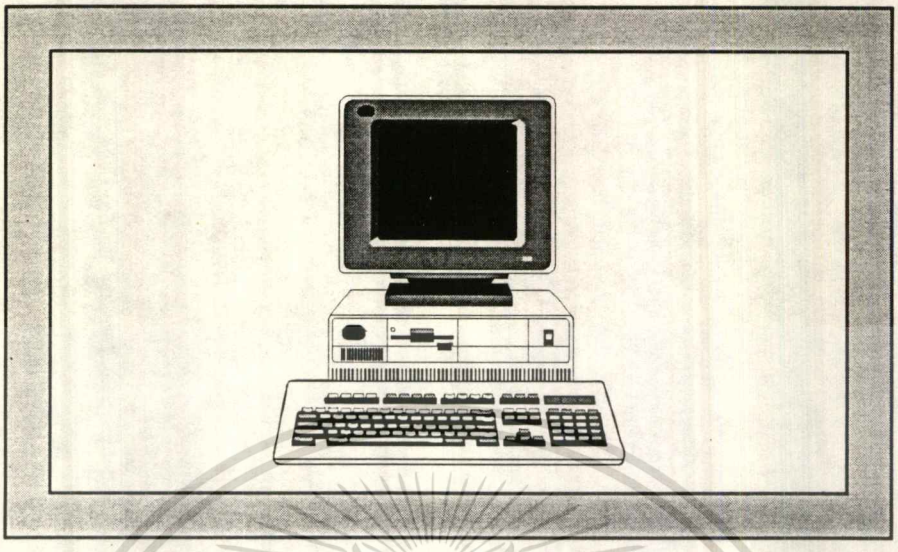
2.1.1.4 ซิงเกิลไซต์อัพเดทไคลเอ็นท์เซิร์ฟเวอร์ (SINGLE-SITE UPDATE CLIENT-SERVER)

ในรูปที่ 2.5 จะแสดงลักษณะแอพพลิเคชันไคลเอ็นท์เซิร์ฟเวอร์ประเภทนี้จะมีความสามารถที่สูงขึ้น โดยมันจะสามารถส่งคำสั่งที่ประกอบด้วยคำสั่งหลายคำสั่งส่งไปยังผู้ให้บริการหลาย ๆ ตัวที่อยู่ห่างไกลได้ แต่ข้อมูลที่ทำการเรียกใช้จากผู้ให้บริการแต่ละตัวมักจะไม่มีความสัมพันธ์กัน ทั้งนี้เนื่องจากว่าผู้ให้บริการแต่ละตัวไม่ได้ต่อเชื่อมกันเป็นเครือข่ายเดียวกันและไม่มีผู้ให้บริการตัวใดทำหน้าที่เป็นตัวกลางในการสื่อสารระหว่างผู้ให้บริการโดยการใช้เส้นทางเครือข่ายผ่านทางผู้ขอใช้บริการอีกทอดหนึ่ง . (TWO PHASE COMMIT PROTOCOL) จากสาเหตุอันนี้ ทำให้การประมวลผลแบบนี้อนุญาตให้ผู้ขอใช้บริการสามารถที่จะทำการแก้ไขข้อมูลที่ผู้ให้บริการได้เพียงตัวเดียวเท่านั้น ถ้าหากข้อมูลที่เก็บอยู่ ณ หน่วยเก็บข้อมูลของผู้ให้บริการต่าง ๆ มีความสัมพันธ์กัน การที่ผู้ใช้ส่งคำสั่งให้มีการแก้ไขข้อมูลที่เก็บอยู่ ณ ผู้ให้บริการตัวอื่นด้วย ถ้าแอพพลิเคชันของผู้ขอใช้บริการสามารถสนับสนุนให้ตัวผู้ขอใช้บริการทำหน้าที่เป็นตัวกลางระหว่างผู้ให้บริการทั้งหลายแล้ว ข้อจำกัดข้างต้นก็สามารถที่จะแก้ไขได้

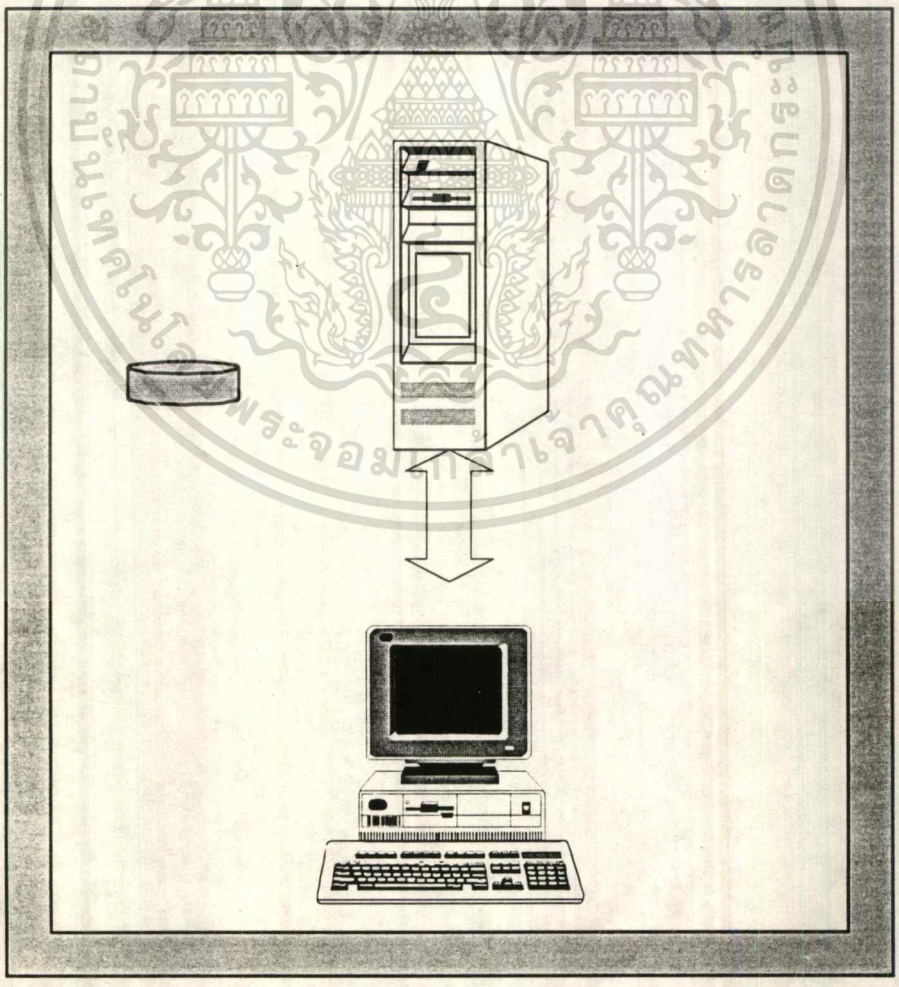
ถึงแม้การประมวลผลแบบนี้จะสามารถแก้ไขข้อมูลที่ผู้ให้บริการได้เพียงหนึ่งตัว แต่ก็ยังมีความเป็นไปได้ที่อาจจะเกิดเดทล็อคขึ้นในเวลาที่มีผู้ใช้หลาย ๆ คนเรียกใช้ข้อมูลพร้อม ๆ กัน ดังนั้นจึงจำเป็นที่จะต้องมีระบบมาทำการควบคุมการเรียกใช้ข้อมูลด้วย การกระจายข้อมูลของไคลเอ็นท์เซิร์ฟเวอร์ประเภทนี้อาจทำได้โดยใช้วิธีแมนนวลเอ็กแทรกต์

2.1.1.5 มัลติไซต์อัพเดทไคลเอ็นท์เซิร์ฟเวอร์ (MULTISITE UPDATE CLIENT-SERVER)

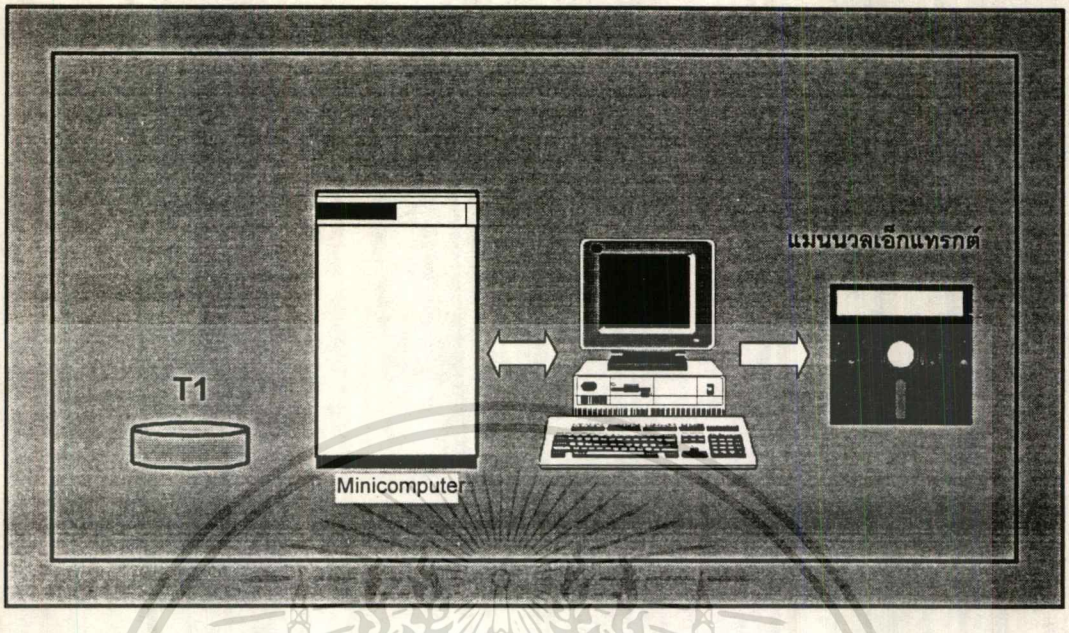
ลักษณะแอพพลิเคชันประเภทนี้ดังแสดงในรูปที่ 2.6 จะสนับสนุนการติดต่อกันระหว่างผู้ให้บริการแต่ละตัว ดังนั้นผู้ใช้จึงสามารถที่จะออกคำสั่งประเภทที่จะแก้ไขข้อมูลที่เก็บอยู่หลายที่ได้ถ้ามองในอีกแง่หนึ่งก็คือ ข้อมูลที่เก็บอยู่ ณ ที่ต่าง ๆ กัน สามารถที่จะมีความสัมพันธ์กันได้ ลักษณะของไคลเอ็นท์เซิร์ฟเวอร์ประเภทนี้จะเป็นประเภทแรกที่มีความสามารถในเรื่องการกระจายฐานข้อมูลและเมื่อมีความสามารถในเรื่องนี้แล้ว การกระจายข้อมูลจะถูกการทำด้วยวิธีสแนพชอต (SNAPSHOTS) จากผู้ใช้บริการฐานข้อมูลแทนที่จะเป็นวิธีแมนนวลเอ็กแทรกต์



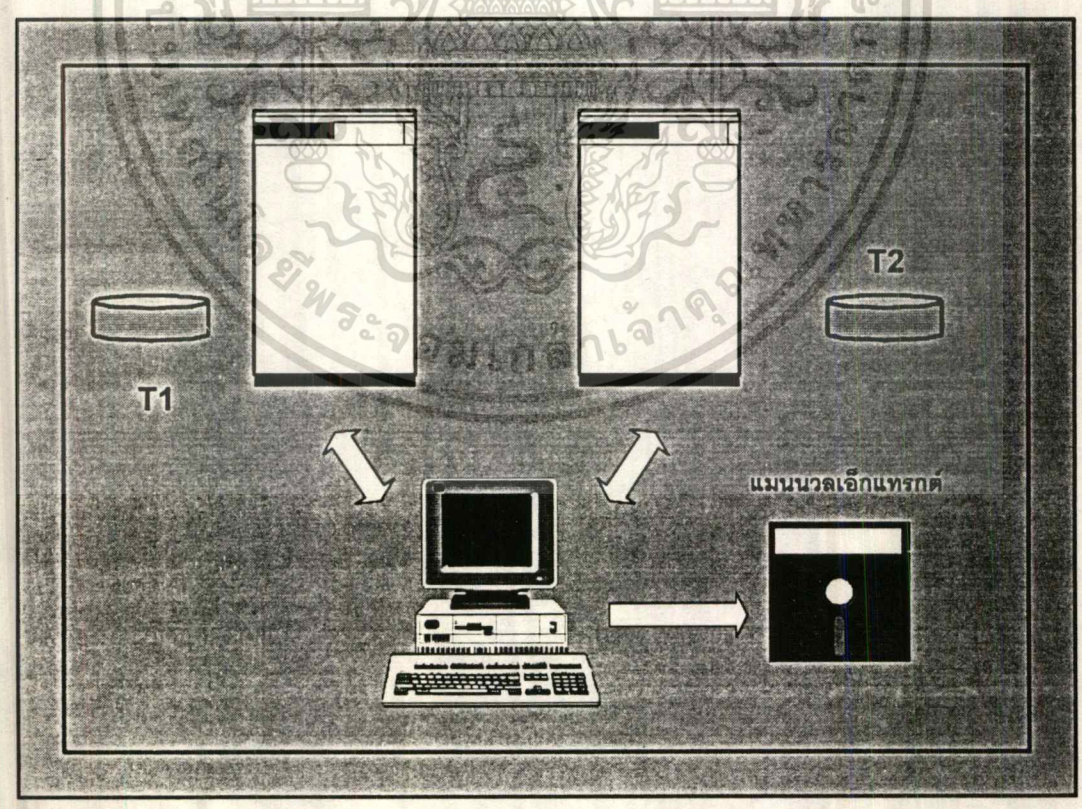
รูปที่ 2.2 ไคลเอ็นท์เซิร์ฟเวอร์ประเภทที่ 1



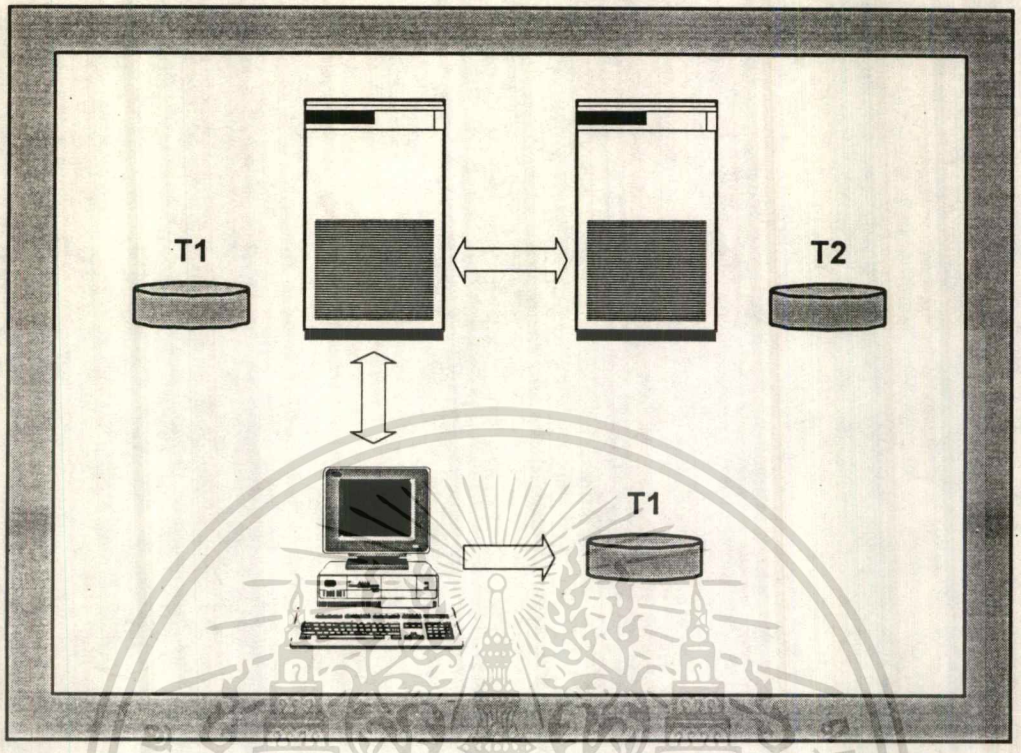
รูปที่ 2.3 ไคลเอ็นเซิร์ฟเวอร์ประเภทที่ 2



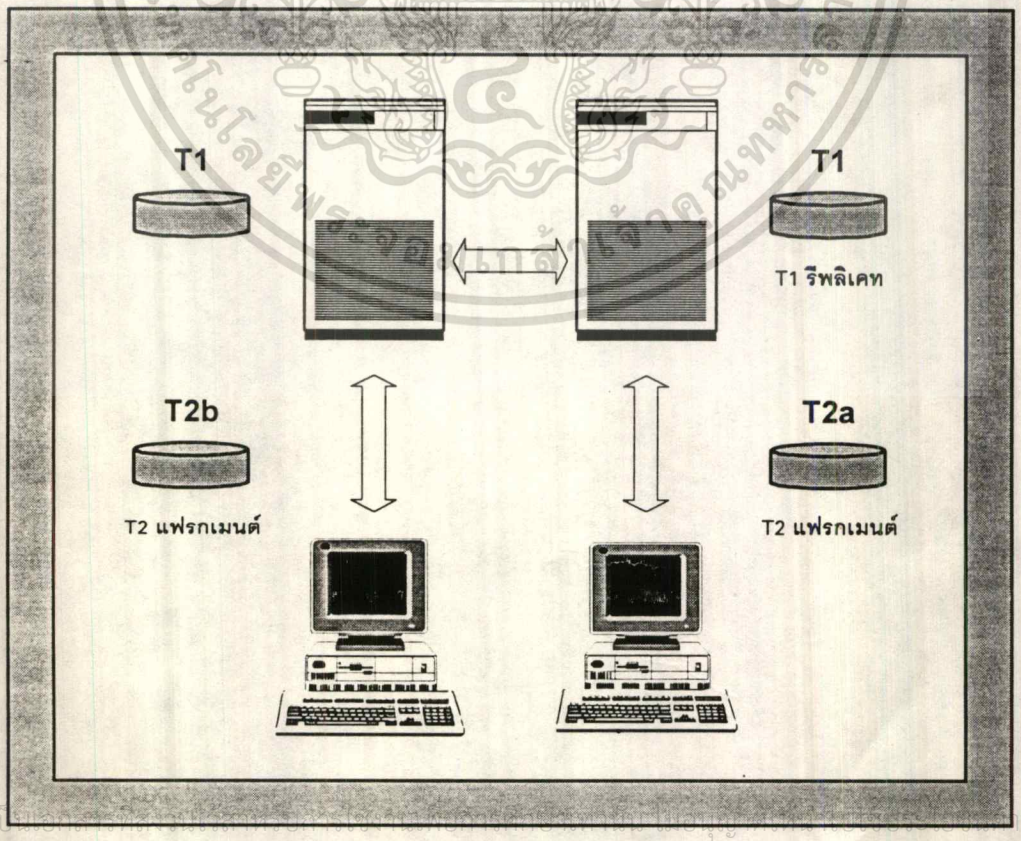
◣ รูปที่ 2.4 ไคลเอ็นท์เซิร์ฟเวอร์ประเภทที่ 3



◣ รูปที่ 2.5 ไคลเอ็นท์เซิร์ฟเวอร์ประเภทที่ 4



◣ รูปที่ 2.6 ไคลเอ็นท์เซิร์ฟเวอร์ประเภทที่ 5



◣ รูปที่ 2.7 ไคลเอ็นท์เซิร์ฟเวอร์ประเภทที่ 6

2.1.1.6 ไคลเอ็นท์เซิร์ฟเวอร์แบบระบบฐานข้อมูลแบบกระจาย (DISTRIBUTED DATABASE CLIENT-SERVER)

เป็นระบบไคลเอ็นท์เซิร์ฟเวอร์ที่ใช้แอพพลิเคชันฐานข้อมูลแบบกระจาย และใช้การประมวลผลแบบดิสทริบิวเต็ดรีเควสต์ (DISTRIBUTED REQUEST) ดังแสดงในรูปที่ 2.7 ลักษณะของไคลเอ็นท์เซิร์ฟเวอร์ประเภทนี้ ผู้ให้บริการฐานข้อมูลจะสนับสนุนทั้งการคัดแบ่งข้อมูล หรือการทำก็อปปี้ข้อมูลทั้งหมดไปเก็บไว้ตามหน่วยเก็บข้อมูลของผู้ให้บริการต่าง ๆ ซึ่งทำให้การอ่านข้อมูลสามารถทำได้ด้วยความรวดเร็วแต่การแก้ไขข้อมูลอาจจะต้องใช้เวลามากกว่า เพราะว่าจะต้องมีการติดต่อกันระหว่างผู้ให้บริการ ซึ่งอาจจะไม่ใช่เพียงแค่ 2 ตัว ดังนั้นเทคโนโลยีทางด้านการสื่อสารจึงมีบทบาทสำคัญในการที่จะขจัดปัญหาในเรื่องของความเร็ว

ความสามารถที่จำเป็นสำหรับแอพพลิเคชันประเภทนี้คือ การที่แอพพลิเคชันจะไม่จำเป็นต้องรู้ตำแหน่งของผู้ให้บริการ หรือตำแหน่งที่เกิดการประมวลผลฐานข้อมูล การควบคุมประสิทธิภาพโดยรวมของระบบ การควบคุมความถูกต้องของข้อมูลที่กระจายเห็นอยู่ตามที่ต่าง ๆ และการควบคุมการทำการกระจายข้อมูลซึ่งเป็นส่วนสำคัญของระบบไคลเอ็นท์เซิร์ฟเวอร์ประเภทนี้

2.1.2 ข้อดีข้อเสียของระบบไคลเอ็นท์เซิร์ฟเวอร์

2.1.2.1 ข้อดีของระบบไคลเอ็นท์เซิร์ฟเวอร์

- เนื่องจากระบบไคลเอ็นท์เซิร์ฟเวอร์มีการแบ่งแยกการประมวลผลออกระหว่างส่วนไคลเอ็นท์และส่วนเซิร์ฟเวอร์ ซึ่งทำให้การประมวลผลข้อมูลจะทำที่เซิร์ฟเวอร์ ทำให้ความเร็วของดีบีเอ็มเอสไม่ขึ้นกับความเร็วของเวิร์คสเตชัน (WORKSTATION) ดังนั้นเวิร์คสเตชันที่ใช้ไม่จำเป็นต้องเป็นเครื่องที่มีประสิทธิภาพสูง แต่สามารถให้ระบบฟรอนท์เอ็นทำงานได้ก็เพียงพอ

  เนื่องจากมีการแบ่งแยกการประมวลผลออกเป็น 2 ฝั่ง ทำให้ลดโหลด (LOAD) ในการติดต่อบนระบบเครือข่ายระหว่างเครือข่ายระหว่างเซิร์ฟเวอร์และไคลเอ็นท์ ถ้าเปรียบเทียบกับระบบโครงสร้าง แบบเซนทรัลไลท์ (CENTRALIZE) จะต้องมีการส่งไฟล์ดาต้าเบสทั้งไฟล์ไปกลับระหว่างผู้ให้บริการและผู้ใช้บริการอยู่ตลอดเวลาที่มีการเรียกใช้ข้อมูล แต่สำหรับระบบไคลเอ็นท์เซิร์ฟเวอร์จะเป็นการส่งแค่คิวรี่ (QUERY) และผลลัพธ์ที่ได้จากดาต้าเบสเซิร์ฟเวอร์ เพราะการใช้ประโยคคำสั่งเอสคิวแอลในแอพพลิเคชันของระบบไคลเอ็นท์เซิร์ฟเวอร์สามารถที่จะสร้างตารางข้อมูลบรรจุผลลัพธ์ที่ได้จากการรวม ตัดทอน และเปลี่ยนแปลงข้อมูลจากตารางข้อมูลจากผู้ให้บริการแล้วค่อยส่งเข้ามายังเครือข่ายเพื่อเป็นการประหยัดการสื่อสาร นอกจากนั้นดาต้าเบสเซิร์ฟเวอร์บางระบบยังสามารถเก็บคำสั่งเอสคิวแอลไว้ในตัวมันเองโดยทำการเก็บไว้ในลักษณะของสตอโพรซีเดอร์ (STORED PROCEDURE) โดยจะอธิบายในหัวข้อหลักการของดีบีเอ็มเอส ซึ่งผู้ใช้จะทำการเรียกใช้โดยออกคำสั่งสั้น ๆ ให้ผู้ให้บริการ

เรียกประโยคคำสั่งนั้น ๆ ออกมาทำงานจะเป็นการช่วยลดปริมาณข้อมูลที่ส่งผ่านเข้าไปในเครือข่ายได้ทางหนึ่ง แต่ก็มีปัญหาอยู่ว่ายังไม่มีการกำหนดมาตรฐานในเรื่องของสตอโพรซีเดอร์ขึ้นมาและระบบจัดการฐานข้อมูลหลายตัวยังไม่สนับสนุนความสามารถในเรื่องนี้

จากการแบ่งแยกออกเป็นไคลเอ็นท์และเซิร์ฟเวอร์ทำให้ส่วนที่ไคลเอ็นท์ทำงานอยู่และแพลตฟอร์มสามารถเป็นอะไรก็ได้ ซึ่งแพลตฟอร์มที่ใช้อาจจะเป็นพีซีที่เข้ากันได้กับพีซีของ IBM , แมคอินทอช (MACINTOSHES), ยูนิกซ์เวิร์คสเตชัน (UNIX WORKSTATION) นอกจากนั้นยังสามารถใช้กับระบบปฏิบัติการได้หลายตัว เช่น ดอส (DOS), พีซีดอส (PC-DOS), ไมโครซอฟท์วินโดว์ (MS WINDOWS), ไอบีเอ็มโอเอสทู (IBM OS/2) หรือ แอบเปิ้ล (APPLE/S SYSTEM 7A) ทำให้เวิร์คสเตชันสามารถใช้แอพพลิเคชันตัวใดก็ได้ในการเข้าถึงดาต้าเบส

- ระบบไคลเอ็นท์เซิร์ฟเวอร์สามารถรักษาดาต้าอินทริกริตี้ (DATA INTEGERITY) ได้โดยดีบีเอ็มเอส จะไม่อนุญาตให้ผู้ใช้เข้าถึงดาต้าเบสจากภายนอก เช่นอาจจะทำการเข้ารหัสไฟล์เพื่อป้องกันผู้ใช้ดูข้อมูลจากภายนอก นอกจากนั้นดีบีเอ็มเอสยังสามารถทำการแบคอัพไปยังเทปแบบเรียลไทม์ (REAL TIME) ได้ คือขณะที่ดาต้าเบสกำลังถูกใช้ก็มีการแบคอัพไปยังเทป การทำดิสค์มิลเลอร์ (DISK MIRRORING) ซึ่งการทำสิ่งเหล่านี้เพื่อรักษาความถูกต้องของข้อมูลจากการเกิดการแครชของระบบหรือระบบเฟลหรือไฟดับ
- สามารถกำหนดขนาดของผู้ขอใช้บริการและผู้ให้บริการได้อย่างอิสระ และสามารถลดและขยายในภายหลังได้
- แอพพลิเคชันต่าง ๆ สามารถใช้ข้อมูลบนผู้ให้บริการร่วมกันได้

2.1.2.2 ข้อเสียของระบบไคลเอ็นท์เซิร์ฟเวอร์

- เสียค่าใช้จ่ายในการดูแลและบำรุงรักษาระบบ
- ฮาร์ดแวร์ที่ทำหน้าที่เป็นดาต้าเบสเซิร์ฟเวอร์จะต้องเป็นเครื่องที่มีประสิทธิภาพสูง ซึ่งทำให้เสียค่าใช้จ่ายสูง
- ซอฟท์แวร์ที่เป็น DBMS จะมีราคาสูง
- มีผลกระทบจากการกระจายข้อมูลต่อประสิทธิภาพของระบบ
- การบริการระบบข้อมูลการทำได้ลำบาก ในระบบที่ข้อมูลทุกอย่างเก็บรวบรวมอยู่ที่ส่วนกลาง (CENTRALIZE SYSTEM) การควบคุมจะกระทำได้สะดวก แต่เมื่อเรากระจายการพัฒนาระบบงาน การประมวลผลแอพพลิเคชันและการจัดเก็บข้อมูลออกไปแล้ว ความง่ายและความสะดวกในการควบคุมจะสูญเสียไปซึ่งจะมีปัญหาในเรื่องต่าง ๆ ดังนี้
    - ◆ การจัดการไลบรารีของโปรแกรมต่าง ๆ ที่เก็บกระจายกันอยู่
    - ◆ การจัดการข้อมูลที่เก็บอยู่ตามที่ต่าง ๆ

- การตรวจสอบและเพิ่มประสิทธิภาพในการทำงานของระบบ
- การสำรองข้อมูลที่เก็บอยู่อย่างกระจายในระบบ
- การจัดการระบบเครือข่าย

2.2 หลักการของระบบการจัดการฐานข้อมูลหรือดีบีเอ็มเอส

ดีบีเอ็มเอส เป็นซอฟแวร์ที่จัดการสตอเรจและการเข้าถึงของข้อมูลในฐานข้อมูล ประกอบด้วยข้อมูลที่สัมพันธ์กันและโปรแกรมที่ใช้สำหรับการเข้าถึงข้อมูล โดยการจัดการที่กล่าวถึงนี้จะครอบคลุมถึงความปลอดภัย (Security) คอนเคอเร็นซี (Concurrency) อินทิกริตี (Intigrity) และการกู้คืน (Recovery) ซึ่งจะกล่าวถึงรายละเอียดในหัวข้อถัดไป ผู้ใช้และโปรแกรมจะทำการเข้าถึงและเก็บข้อมูลโดยการติดต่อกับดีบีเอ็มเอส

2.2.1 จุดประสงค์หลักในการออกแบบดีบีเอ็มเอส

- เพื่อการจัดการข้อมูลที่มีปริมาณมาก ผู้ใช้สามารถดึงและเก็บข้อมูลได้อย่างสะดวกและมีประสิทธิภาพมากยิ่งขึ้น
- จัดการให้ผู้ใช้ให้มองเห็นเฉพาะข้อมูลที่ต้องการเท่านั้น โดยไม่ต้องรู้ว่าข้อมูลนั้นถูกจัดเก็บ ณ ที่ใด และถูกบำรุงรักษา (Maintain) อย่างไร

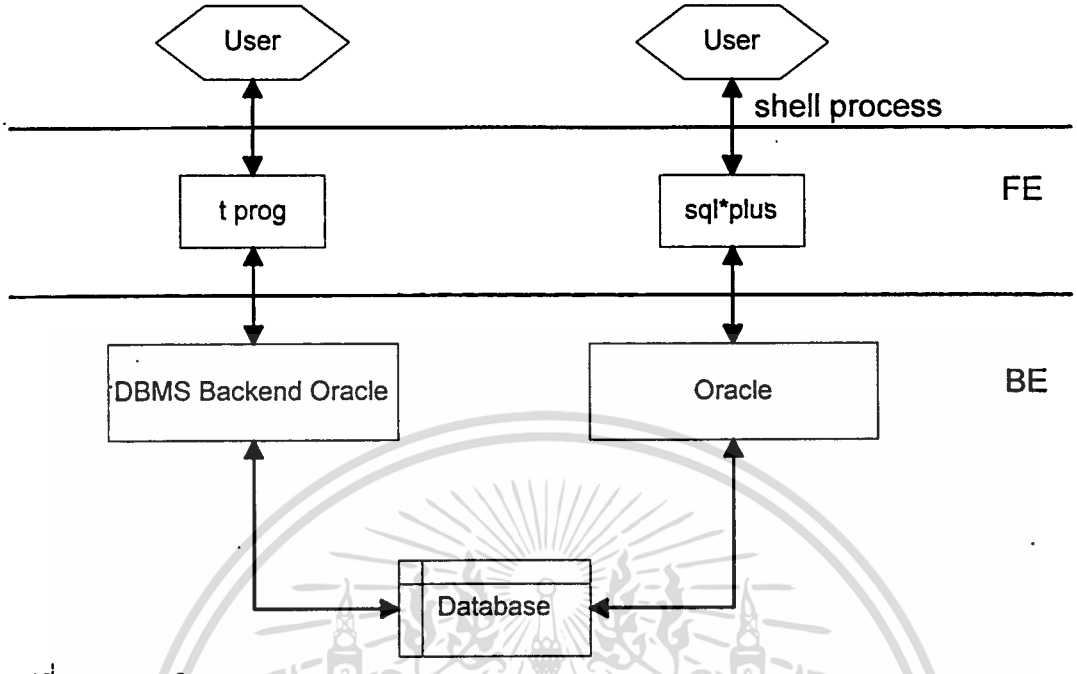
2.2.2 สถาปัตยกรรมของดีบีเอ็มเอส

2.2.2.1 ดูออลโพรเซส (Dual Process Architecture)

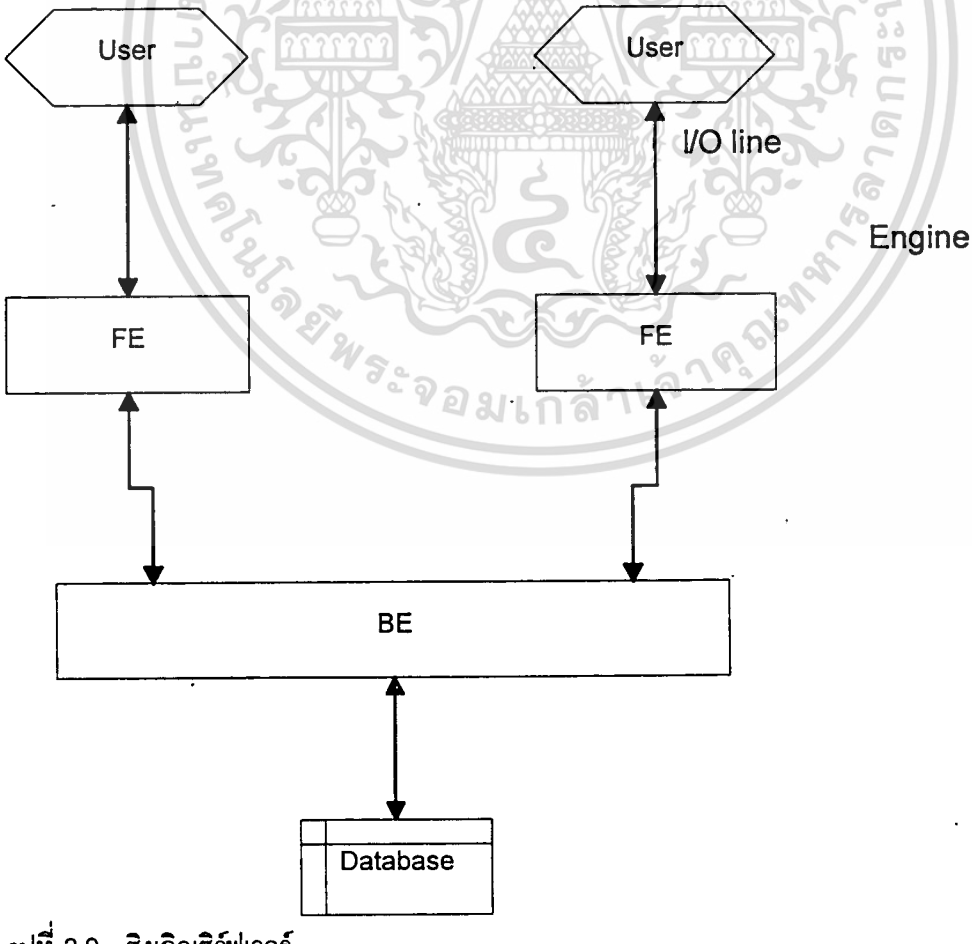
ในรูปที่ 2.8 สถาปัตยกรรมแบบดูออลโพรเซสนี้จะทำการอิมพลีเมนท์ง่ายเพราะระบบปฏิบัติการจะช่วยจัดการการทำเชดดูลิ่ง (Scheduling) และคอนเคอเรนซีคอนโทรล ซึ่งจะเหมาะสมกับระบบปฏิบัติการที่ใช้สวอปเปอร์ (Swapper) แต่วิธีนี้ก็มีข้อเสียคือจำนวนโพรเซสจะมีจำนวนมากทำให้เกิดคอนเทคสวิตชิงสูง และจำนวนแอคทีฟยูสเซอร์น้อย ดังนั้นราคาของดีบีเอ็มเอสประเภทนี้จะมีราคาไม่แพงนัก

2.2.2.2 ซิงเกิลเซิร์ฟเวอร์ (Single Server Architecture)

สถาปัตยกรรมแบบซิงเกิลเซิร์ฟเวอร์นี้ ดีบีเอ็มเอสจะเป็นผู้ดูแลคอนเคอเรนซีคอนโทรลและเชดดูลิงเองเพราะมีแอคทีฟยูสเซอร์หลายคนใช้ดีบีเอ็มเอสตัวเดียวกัน ดีบีเอ็มเอสจะจัดการการทำงานของผู้ใช้โดยจะมองการทำงานของผู้ใช้เป็นเทรด (Thread) ซึ่งเทรดจะเป็นงานย่อยๆหรือการโพรเซสเล็ก ๆ ทำให้ลดจำนวนโพรเซสลงเมื่อเปรียบเทียบกับวิธีแรกเพราะโอเวอร์เฮดจากการคอนเทคสวิตชิงลดลง



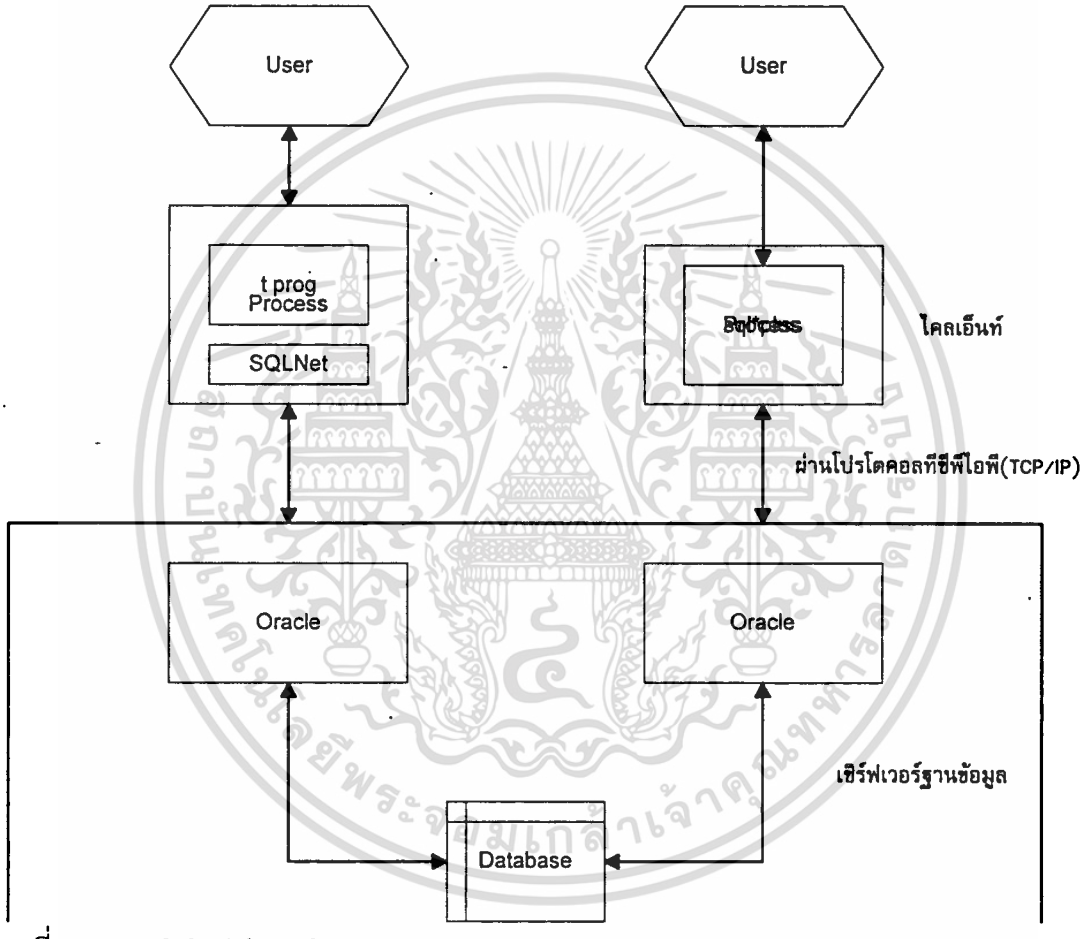
รูปที่ 2.8 ดูออลโพรเซส



รูปที่ 2.9 ซิงเกิลเซิร์ฟเวอร์

นอกจากนั้นจำนวนแอคทีฟยูสเซอร์จะเพิ่มขึ้นและแบคเอ็นสามารถอิมพลีเมนท์บนซิงเกิล ทาสก์แมชชีน (Single Task Machine) ได้ แต่จะอิมพลีเมนท์ยากกว่าแบบแรก ดังแสดงในรูปที่ 2.9

2.2.2.3 มัลติเพิลเซิร์ฟเวอร์ (Multiple Server Architecture)



รูปที่ 2.10 มัลติเพิลเซิร์ฟเวอร์

สถาปัตยกรรมแบบมัลติเพิลเซิร์ฟเวอร์นี้รองรับการทำงานแบบมัลติโพรเซสเซอร์ (Multiprocessor) โดยสามารถมีโปรเซสเซอร์ได้มากกว่า 1 โปรเซสเซอร์ รูปที่ 2.10 แสดงสถาปัตยกรรมแบบมัลติเพิลเซิร์ฟเวอร์

2.2.3 ส่วนประกอบที่สำคัญของดีบีเอ็มเอส

ดีบีเอ็มเอสที่ใช้ในการทำโครงงานนี้จะเป็นดีบีเอ็มเอสแบบสัมพันธ์หรืออาจเรียกอีกอย่างหนึ่งว่า เอสคิวแอลเซิร์ฟเวอร์ โดยมีส่วนประกอบที่สำคัญดังนี้

2.2.3.1 ส่วนจัดการไฟล์ (File Manager)

จะจัดการเนื้อที่บนดิสค์และโครงสร้างข้อมูลที่ใช้ในการแสดงข้อมูลที่เก็บบนดิสค์

2.2.3.2 ส่วนจัดการฐานข้อมูล (Database Manager)

จะจัดการการติดต่อระหว่างข้อมูลที่เก็บไว้ที่ระดับล่าง (Low Level) กับแอพพลิเคชันโปรแกรม และจัดการการส่งคิวรี่ (Query)ระหว่างระบบ นอกจากนั้นส่วนจัดการฐานข้อมูลยังทำหน้าที่

- ติดต่อกับส่วนจัดการไฟล์ โดยส่วนจัดการฐานข้อมูลจะจัดการแปลงประโยคดีเอ็มแอล ดังที่จะกล่าวในส่วนพรีคอมไพล์ภาษาสำหรับการมานิพูเลทข้อมูลให้เป็นคำสั่งระบบไฟล์ระดับล่าง (Low Level File System Command) ดังนั้นส่วนจัดการฐานข้อมูลมีหน้าที่โดยตรงในการจัดเก็บ การดึงข้อมูล และการอัพเดทข้อมูลในฐานข้อมูล
- ดูแลอินทิกริตีและความปลอดภัยของข้อมูล

อินทิกริตีของข้อมูล หรือ กฎบังคับความถูกต้องของข้อมูล จะหมายถึง ความถูกต้องของข้อมูล โดยจะตรวจสอบ ว่าค่าไพรมารีคีย์จะต้องไม่เป็นค่านัล (Null) และค่าฟอเรนคีย์จะต้องตรงกับค่าไพรมารีคีย์และจะต้องไม่มีค่านัล

ความปลอดภัยของ ข้อมูล จะเป็นการป้องกันข้อมูลจากการเปิดเผย การเปลี่ยนแปลง หรือการทำลาย

ว่าผู้ใช้กำลังทำในสิ่งที่ผู้ใช้ผู้นั้นมีสิทธิ์ที่จะทำ เช่น การลบข้อมูลในตาราง การดรอบตาราง เป็นต้น และส่วนจัดการฐานข้อมูลจะดูแลอินทิกริตีของข้อมูลเพื่อให้แน่ใจว่าสิ่งที่ผู้ใช้กำลังทำอยู่นั้นมีความถูกต้อง

ในด้านการรักษาความปลอดภัยของข้อมูลในภาษาเอสคิวแอล เราสามารถใช้คำสั่งวิว (View) เพื่อสร้างโครงสร้างตารางใหม่เพื่อจำกัดการมองเห็นข้อมูลของผู้ใช้ได้ นอกจากนั้นยังสามารถใช้คำสั่งแกรนท์เพื่ออนุญาตให้ผู้ใช้คนอื่นสามารถมองเห็นตารางของเราได้

- การทำการกู้คืนและคอนเคอเรนซีคอนโทรล

ในดีบีเอ็มเอสที่เล็กๆมักจะไม่มีการสนับสนุนการทำรีโคเวอรี และ คอนเคอเรนซี กล่าวคือปัญหาคอนเคอเรนซีไม่ค่อยจะเกิดขึ้น ส่วนปัญหาการทำรีโคเวอรีจะกลายเป็นความรับผิดชอบของผู้ใช้ โดยผู้ใช้ต้องทำการแบคอัพฐานข้อมูลไว้เอง และต้องทำการรีดู (redo) งานด้วยตัวเองถ้าระบบเกิดเฟล (failure) ขึ้นมา

ทรานแซคชัน หมายถึงหน่วยการทำงานหน่วยหนึ่งที่เปลี่ยนคอนซิสเตนท์สเตท (consistent state)

ของฐานข้อมูลจากสเตทหนึ่งเป็นอีกคอนซิสเตนท์สเตทหนึ่งโดยไม่จำเป็นต้องรักษาคอนซิสเตนท์สเตททุกจุดในทรานแซคชันนั้น

ระบบที่สนับสนุนการโพรเซสทรานแซคชันจะยืนยันว่า ถ้าทรานแซคชันที่มีการเอ็กซีคิวท์คำสั่งอัพเดตบางคำสั่งอยู่ เกิดการเฟล (fail) ขึ้นก่อนที่จะเทอมิเนท (terminate) การอัพเดตเหล่านั้นจะถูกยกเลิก (undo) คือทรานแซคชันจะทำก็ทำทั้งหมด หรือไม่ก็ไม่ทำเลยเท่านั้น

ส่วนของระบบที่ทำการควบคุมให้ทรานแซคชันเป็นอะตอมมิก คือ ทรานแซคชันเมเนจเจอร์ และมีโอเปอเรชันที่สำคัญคือ คอมมิท (COMMIT) และ โรลแบ็ค (ROLLBACK)

การทำคอมมิท แปลว่า ทรานแซคชันนั้นจบโดยสมบูรณ์แล้ว และบอกทรานแซคชันเมเนจเจอร์ว่างานหน่วยนี้เสร็จแล้วและฐานข้อมูลเข้าสู่ (หรือควรจะเข้าสู่) คอนซิสเตนท์สเตทอีกครั้งและการอัพเดตทุกอย่างที่ถูกทำได้ทำเป็นการถาวรไปแล้ว

การทำโรลแบ็ค แปลว่า ทรานแซคชันนั้นจบโดยไม่เสร็จสมบูรณ์ และบอกทรานแซคชันเมเนจเจอร์ว่ามีบางสิ่งผิดพลาด ฐานข้อมูลอาจไม่อยู่ในคอนซิสเตนท์สเตทและการอัพเดตทั้งหมดที่ทำโดยงานหน่วยนี้จะต้องถูกโรลแบ็ค หรือยกเลิก ตัวอย่างเช่น DB2 จะคอมมิทแบบอัตโนมัติเมื่อโปรแกรมจบและ จะสั่งโรลแบ็คเองเมื่อโปรแกรมไม่เสร็จสมบูรณ์ แต่ถ้าระบบเกิดการเฟลขึ้นมาระหว่างที่กำลังรันโปรแกรม เราต้องทำโรลแบ็คเองเมื่อระบบถูกเริ่มขึ้นใหม่ เพราะฉะนั้นเราจะสามารถละเลยการทำเอกซ์พลิสิทคอมมิทได้แต่เรายังคงต้องการเอกซ์พลิสิทโรลแบ็คอยู่

เราสามารถอันดู (undo) การอัพเดตที่ทำไปแล้วได้โดยใช้ล็อค (log) หรือ เรียกว่า เจอนอล (journal) บนเทปหรือดิสค์ได้

ในระบบรีเลชันนอล (relational system) คำสั่งจัดการข้อมูลจะทำกับเรคคอร์ดหลายๆอันในครั้งเดียว เช่นคำสั่งอัพเดต ถ้าเกิดเฟลระหว่างทำคำสั่งนั้นยังไม่จบ ฐานข้อมูลก็ยังคงไม่ถูกดัพเดต เพราะ เอสคิวแอลสเตทเมนท์ (SQL statement) ต้องเป็นอะตอมมิก

ซิงโครนัสพอยท์ (syschronous point) หรือเรียกว่า ซิงค์พอยน์ คือเขตแดนระหว่างทรานแซคชันที่ทำติดต่อกัน 2 อัน และหมายถึงการจบของทรานแซคชันแรก และเป็นจุดที่ฐานข้อมูลอยู่ในคอนซิสเตนท์สเตท โอเปอเรชันที่สร้างซิงค์พอยท์ ได้แก่ คอมมิท, โรลแบ็ค และโปรแกรมอินนิชิเอชัน (program initiation)

เมื่อซิงค์พอยท์ถูกสร้างขึ้น การอัพเดตที่ทำโดยโปรแกรมหลังจากซิงค์พอยท์จุดที่แล้วจะถูกคอมมิทหรือโรลแบ็ค เคอร์เซอร์ที่เปิดไว้ทั้งหมดจะถูกปิดและดาต้าเบสโพซิชันนิ่ง (database positioning) ทั้งหมดจะถูกลบทิ้งไป เรคคอร์ดล็อคทั้งหมดจะถูกปล่อย

การทำคำสั่งคอมมิทและโรลแบ็คจะเทอมิเนทแค่ทรานแซคชัน ไม่ใช่ทั้งโปรแกรม โปรแกรมประกอบด้วยหลายๆทรานแซคชันที่ทำต่อเนื่องกันโดยมีคำสั่งคอมมิทหรือโรลแบ็ค เพื่อจบการ

ทำงานของทรานแซคชันหนึ่ง และเริ่มทรานแซคชันใหม่ต่อไป ถ้าโปรแกรมประกอบด้วยทรานแซคชันเดียว เราสามารถเขียนโค้ทโดยไม่ต้องมีเอ็กซ์พลิสิทคอมมิท หรือเอ็กซ์พลิสิทโรลแบ็คเลย

เราจะเห็นได้ว่าทรานแซคชันไม่ได้เป็นเพียงแค่หน่วยการทำงานแต่ยังเป็นหน่วยขของการรีโคเวอร์ด้วย ถ้าระบบเกิดแครช (crash) ภายหลังจากคำสั่งคอมมิทเพียงเล็กน้อย อาจเป็นไปได้ว่าการอัพเดตของทรานแซคชันนั้นยังไม่ได้ถูกเขียนลงบนฐานข้อมูลจริง เมื่อทำคำสั่งคอมมิท มันอาจจะยังคงรอการเขียนอยู่ในสตอเรจบัฟเฟอร์ และจะสูญหายไปได้เมื่อระบบเกิดการแครช ถ้าเกิดเหตุการณ์เช่นนี้ เราต้องทำการอัพเดตฐานข้อมูลเองหลังจากระบบเริ่มขึ้นมาใหม่ (redo) โดยอาจดูค่าใหม่ที่ต้องการอัพเดตจากล็อคไฟล์ที่ตรงกัน (ตามข้อกำหนดว่า ล็อคจะต้องถูกเขียนลงสเตเบิลสตอเรจ (stable storage) ก่อนที่จะทำคำสั่งคอมมิทเสร็จ : เรียกกฎนี้ว่า Write-Ahead Log Protocol) ทำให้เราสามารถกู้ทรานแซคชันที่เราทำเสร็จสมบูรณ์ไปแล้ว แต่ไม่ได้ถูกเขียนลงฐานข้อมูลก่อนที่ระบบจะแครช คืนมาได้

ระบบอาจถูกทำการรีโคเวอร์ได้จากหลายๆสาเหตุไม่เพียงแต่จากลอจิคอลเฟลเลอร์ เช่นการเกิดโอเวอร์โฟลว์ในทรานแซคชัน เท่านั้น แต่ยังอาจเกิดจากโกลบอลเฟลเลอร์ เช่นไฟดับ และโลคอลเฟลเลอร์ ซึ่งจะมีผลกระทบแต่กับทรานแซคชันที่มีเฟลเลอร์เกิดขึ้นเท่านั้น ส่วนโกลบอลเฟลเลอร์จะเกิดผลกระทบกับทรานแซคชันทั้งหมดหรือเกือบทั้งหมดที่กำลังดำเนินอยู่และมีเฟลเลอร์เกิดขึ้น โกลบอลเฟลเลอร์แบ่งเป็น 2 ประเภทคือ

ซิสเต็มเฟลเลอร์ *(system failure)* เช่นไฟดับ มีผลกับทุกทรานแซคชันที่กำลังดำเนินอยู่ แต่ไม่ทำลายฐานข้อมูล เรียกว่า ซอร์ฟแครช (soft crash) ทำให้เมนสตอเรจหายหรือดาต้าเบสบัฟเฟอร์หายนั่นเอง และทำให้เราไม่รู้สเตทที่แท้จริงของทรานแซคชันที่กำลังดำเนินอยู่ขณะเกิดเฟลเลอร์อีกต่อไป แต่เรารู้ว่า ทรานแซคชันนั้นจะอยู่ในสเตท 2 สเตทที่เป็นไปได้คือ ยังทำไม่เสร็จสมบูรณ์ กรณีนี้เราต้องทำการอันดู หรือโรลแบ็ค เมื่อระบบเริ่มขึ้นมาใหม่ หรือ อีกกรณีหนึ่งคือ ทำเสร็จแล้วแต่ยังไม่ได้เขียนลงฐานข้อมูลจริง กรณีนี้เราต้องรีดูเมื่อระบบเริ่มขึ้นมาใหม่ ระบบจะรู้ว่าควรอันดู หรือรีดูทรานแซคชันใดๆ ต้องมีการทำเช็คพอยท์ การทำเช็คพอยท์ทุกช่วงเวลาหนึ่ง คือ การเขียนดาต้าเบสบัฟเฟอร์ลงบนฐานข้อมูลจริง และเขียนเช็คพอยท์เรคคอร์ทลงบนล็อค (log) จริงๆ เช็คพอยท์เรคคอร์ทประกอบด้วยลิสต์ของทรานแซคชันทั้งหมดที่กำลังดำเนินอยู่ ณ เวลาที่ทำเช็คพอยท์

มีเดียเฟลเลอร์ (media failure) เช่นหัวอ่านกระทบกับแผ่นดิสค์ หรือดิสค์คอนโทรลเลอร์พัง มีเดียเฟลเลอร์จะมีผลทำลายฐานข้อมูล หรือบางส่วนของฐานข้อมูล และมีผลกระทบกับทรานแซคชันที่กำลังใช้งานส่วนของฐานข้อมูลนั้นอยู่ เรียกว่า ฮาร์ดแครช(hard crash)

การกู้ฐานข้อมูลที่ถูกทำลายไปคืน ทำได้โดยการรีโหลด (reload) หรือรีสตอ (restore) ฐานข้อมูลจากแบ็คอัพก็อปปี้ (หรือดัมพ์ (dump)) แล้วใช้ล็อคทั้งหมดที่เก็บไว้หลังจากการแบ็คอัพครั้งสุดท้าย รวมทั้งอันที่กำลังแอคทีฟอยู่มาทำรีดูทรานแซคชันทั้งหมดที่ทำเสร็จไปแล้วหลังจากแบ็คอัพก็อปปี้ชุดนี้ เราไม่จำเป็นต้องอันดูทรานแซคชันที่ยังดำเนินอยู่ขณะที่เกิดเฟลเลอร์ เพราะการอัพเดตทุกอันของทรานแซคชันเหล่านั้นจะถูกอันดู หรือถูกทำลายไปเรียบร้อยแล้ว

ปัญหา 3 ประการของการทำคอนเคอเรนซี

ดีบีเอ็มเอสโดยส่วนใหญ่เป็นระบบมัลติยูสเซอร์ คืออนุญาตให้ทรานแซคชันหลายๆอันสามารถแอคเซสฐานข้อมูลเดียวกัน ณ เวลาเดียวกันได้ ระบบมัลติยูสเซอร์นี้ต้องการคอนเคอเรนซีคอนโทรล เพื่อให้แน่ใจได้ว่าทรานแซคชันเหล่านั้น จะไม่มีการรบกวนกันระหว่างโอเปอเรชันของแต่ละทรานแซคชัน ถ้าไม่มีคอนเคอเรนซีคอนโทรลจะเกิดปัญหาขึ้นมา 3 ประการที่เด่นชัด แต่เราสามารถแก้ปัญหาเหล่านี้ได้โดยการใช้ล็อคเป็นคอนเคอเรนซีคอนโทรลเมคคานิซึม (concurrency control mechanism)

ปัญหาที่เกิดจากการขาดคอนเคอเรนซีคอนโทรลเมคคานิซึมได้แก่

◆ ปัญหาการอัพเดตสูญหาย (The lost update problem) เช่น เมื่อทรานแซคชัน A ทำการอัพเดตออบเจ็กต์ R แล้วต่อมาทรานแซคชัน B ก็ทำการอัพเดต R อีกครั้ง การอัพเดตของทรานแซคชัน A จึงสูญหายไป

◆ ปัญหาการใช้ข้อมูลที่ยังไม่ได้รับการคอมมิท (Uncommitted dependency problem) เช่นทรานแซคชัน A อ่านค่าของออบเจ็กต์ R ที่ถูกทรานแซคชัน B อัพเดตไปก่อนแล้ว B ยังอาจถูกโรลแบ็คได้ ถ้า B เฟล, ค่าของ R ที่ทรานแซคชัน A อ่านค่าที่ผิดๆไป

◆ ปัญหาการวิเคราะข้อมูลขณะที่ข้อมูลนั้นยังถูกอัพเดตอยู่ (Inconsistent analysis problem) ผลลัพธ์จากการวิเคราะห์จึงไม่ถูกต้อง

ปัญหาเหล่านี้จะทำให้ได้ผลลัพธ์ที่ผิดๆ เพราะคอนเคอเรนท์ทรานแซคชันต่างๆมีการทำคำสั่งที่อาจรบกวนกันได้ กล่าวคือ ทรานแซคชัน 2 อันที่ได้ผลลัพธ์ถูกต้องในตัวมันเองและมีคำสั่งที่เกิดการรบกวนกันได้ เมื่อนำโอเปอเรชันต่างๆของทั้งสองมาเอ็กซีคิวท์แบบคอนเคอเรนท์กัน จะได้ผลลัพธ์ที่ผิดไปจากผลลัพธ์ที่ควรจะเกิดเมื่อนำทรานแซคชันทั้งสองมารันต่อกัน

การล็อค (Locking)

หลักการของการทำล็อคง่ายๆคือ เมื่อทรานแซคชันต้องการความแน่นอนว่าออบเจ็กต์ที่มันสนใจ (มักหมายถึง ดาต้าเบสเรคคอร์ด) จะไม่เปลี่ยนแปลงไปโดยทรานแซคชันอื่นเมื่อมันจะอ่านหรือดัพเดตออบเจ็กต์เดิมนั้น

ผลของการใช้ล็อค คือการล็อคทรานแซคชันอื่นๆออกไปจากออบเจ็กต์ ทำให้เป็นที่แน่นอนว่าออบเจ็กต์ที่ถูกล็อคอยู่นั้นจะอยู่ในสเตทเดิมนานตราบเท่าที่ทรานแซคชันนั้นต้องการ

- เอ็กซ์คลูซีฟล็อค (exclusive locks / X locks) ถ้าทรานแซคชัน A ทำเอ็กซ์ล็อคบนเรคคอร์ด R ถ้าทรานแซคชัน B ต้องการล็อคชนิดใดก็ได้บน R แล้ว B ต้องรอจนกว่า A จะปล่อยล็อค
- แชร์ล็อค (shared locks / S locks) ถ้า A ทำเอสล็อคบนเรคคอร์ด R แล้ว
- เมื่อ B ต้องการทำเอ็กซ์ล็อคบน R, B จะต้องคอยจนกว่า A จะปล่อยล็อค
- เมื่อ B ต้องการทำเอสล็อคบน R, B จะได้เอสล็อคบน R เช่นกัน
- การขอล็อคเรคคอร์ดมักเป็นอิมพลิสิท (Implicit) โดยเฉพาะในระบบรุ่นใหม่ๆส่วนใหญ่ เมื่อทรานแซคชันดึงเรคคอร์ดขึ้นมาจะมีการทำเอสล็อคบนเรคคอร์ดนั้นโดยอัตโนมัติ เมื่อทรานแซคชันอัพเดตเรคคอร์ดมันจะทำเอ็กซ์ล็อคโดยอัตโนมัติบนเรคคอร์ดนั้นเช่นกัน ถ้าทรานแซคชันมีเอสล็อคบนเรคคอร์ดนั้นอยู่แล้ว คำสั่งอัพเดตจะทำการอัพเกรดเอสล็อค เป็นเอ็กซ์ล็อค
- เอ็กซ์ล็อค จะคงอยู่จนถึงซิงค์พอยท์ถัดไป ส่วนเอสล็อค โดยปกติก็จะอยู่จนถึงซิงค์พอยท์เช่นกัน

2.2.3.3 ส่วนประมวลผลคิวรี่ (Query Processor)

จะแปลงประโยคที่เป็นภาษาเอสคิวแอลให้เป็นคำสั่งระดับล่าง (Low Level Instruction) ที่ส่วนจัดการฐานข้อมูลสามารถเข้าใจได้ นอกจากนี้ยังสามารถหากลยุทธ์ที่จะแปลงคิวรี่นั้นให้อยู่ในรูปที่สามารถทำการเอ็กซีคิวท์ได้รวดเร็วและมีประสิทธิภาพยิ่งขึ้น

2.2.3.4 ส่วนพรีคอมไพล์ภาษาสำหรับการมานิพูเลทข้อมูล (Data Manipulation Language (DML) Precompiler)

การมานิพูเลทข้อมูลจะหมายถึง

- การดึงข้อมูลที่เก็บไว้ที่ฐานข้อมูลขึ้นมา
- การอินเสิร์ทข้อมูลใหม่ลงในฐานข้อมูล
- การลบข้อมูลที่เก็บอยู่ในฐานข้อมูลออกไป

ภาษาสำหรับการมานิพูเลทข้อมูล (Data Manipulation Language) หรือดีเอ็มแอล (DML) จะเป็นภาษาที่ช่วยให้ผู้ใช้สามารถภาษานี้ในการเข้าถึงและการมานิพูเลทข้อมูล มี 2 ประเภท

- โพรซีเดอรอล ภาษาประเภทนี้ผู้ใช้ต้องบอกข้อมูลที่ต้องการและวิธีที่จะนำมา
- นอนโพรซีเดอรอล ภาษาประเภทนี้ผู้ใช้ต้องบอกข้อมูลที่ต้องการแต่ไม่ต้องบอกวิธีที่จะนำมา

ซึ่งส่วนพรีคอมไพล์ภาษาสำหรับการมานิพูเลทข้อมูลจะทำหน้าที่แปลงประโยคดีเอ็มแอล (DML, Data Manipulation Language) ที่ติดมากับแอพพลิเคชันโปรแกรมให้เป็นการเรียกโพรซีเดอร์ในภาษาโฮสต์ (Host Language) ส่วนพรีคอมไพล์ภาษาที่ใช้จัดการข้อมูลนี้จะติดต่อกับส่วนประมวลผลคิวรี่เพื่อที่จะสร้างโค้ด (Code) ที่เหมาะสม

2.2.3.5 ส่วนคอมไพล์ภาษาที่ใช้นิยามข้อมูล (Data Definition Language Compiler)

จะทำหน้าที่แปลงประโยคภาษาที่ใช้นิยามข้อมูลให้เป็นเซตของตารางที่ประกอบด้วยเมต้าดาต้า (Metadata)

นอกจากนั้นโครงสร้างข้อมูลยังประกอบด้วยส่วนฟิสิคอลซิสเต็มอิมพลีเมนเทชัน (Physical System Implementation) ซึ่งประกอบด้วย

- ไฟล์ข้อมูล (Data File) จะเก็บตัวฐานข้อมูล
- ดาต้าดิกชันนารี (Data Dictionary) จะเก็บเมตาดาต้าเกี่ยวกับโครงสร้างของฐานข้อมูล
- ตัวชี้ (Index) จะช่วยในการเข้าถึงข้อมูลให้รวดเร็วยิ่งขึ้น

2.3 หลักการของออบเจ็กต์โอเรียนเต็ด

ออบเจ็กต์โอเรียนเต็ดโมเดล (object-oriented model)

ออบเจ็กต์โอเรียนเต็ดโมเดล มีพื้นฐานอยู่บนหลักการของออบเจ็กต์โอเรียนเต็ดโปรแกรมมิ่ง (Object Oriented Programming) การเขียนโปรแกรมแบบนี้เริ่มมีขึ้นโดยใช้ภาษา Simula 67 ต่อมาภาษา C++ และ Smalltalk ก็กลายเป็นภาษาที่ได้รับความนิยมแพร่หลาย

ในระยะแรกเราพัฒนาฐานข้อมูลด้วยระบบจัดการไฟล์ ต่อมาก็พัฒนาเป็นใช้ฐานข้อมูลแบบเน็ตเวิร์ค (Network Databases) โครงสร้าง (hierarchical databases) และรีเลชันนอล (relational databases) ตามลำดับ ต่อมาไม่นานเราได้นำเทคโนโลยีฐานข้อมูลไปใช้กับแอพพลิเคชันต่างๆนอกเหนือจากจุดประสงค์การประมวลข้อมูล แอพพลิเคชันใหม่ๆเหล่านี้ได้แก่

- คอมพิวเตอร์ช่วยการออกแบบ (CAD: Computer-Aided Design)
- คอมพิวเตอร์ช่วยในงานซอร์ฟแวร์เอ็นจิเนียริ่ง (CASE: Computer-Aided Software Engineering)

- ฐานข้อมูลมัลติมีเดีย (Multimedia databases)
- ระบบข่าวสารในสำนักงาน (OIS: Office Information Systems)
- ระบบฐานข้อมูลผู้เชี่ยวชาญ (Expert Database System)

แอพพลิเคชันต่างๆเหล่านี้ไม่ได้รับการสนใจในช่วงปี 1970-1979 (ขณะที่มีการออกแบบระบบฐานข้อมูลรุ่นแรกๆ) ต่อมาเมื่อมีการพัฒนาขนาดของหน่วยความจำหลักของเครื่อง, ความเร็วของหน่วยประมวลผลหลัก, ราคาของฮาร์ดแวร์ลดลง และการเห็นประโยชน์ในระบบจัดการฐานข้อมูลมีมากขึ้น จึงได้นำเทคโนโลยีฐานข้อมูลมาช่วยในงานต่างๆมาก แอพพลิเคชันใหม่ๆเหล่านี้ต้องการดาต้าโมเดลแบบใหม่, ภาษาคิวรี่แบบใหม่, และทรานแซคชันโมเดลแบบใหม่

ออบเจ็กต์ที่ซับซ้อน (complex object) คือ ออบเจ็กต์เดี่ยวตัวหนึ่งในโลกแห่งความเป็นจริง (real world) แต่มันรวมเอาออบเจ็กต์อื่นๆไว้ด้วย ออบเจ็กต์เหล่านี้จะมีโครงสร้างภายในที่ซับซ้อนซึ่งมักจะอยู่ในรูปโครงสร้าง (structural hierarchy) แสดงความสัมพันธ์ระหว่างออบเจ็กต์ภายใน การสร้างโมเดลของออบเจ็กต์ที่ซับซ้อน นำไปสู่การพัฒนาฐานข้อมูลแบบออบเจ็กต์โอเรียนเต็ด (Object Oriented Database) ซึ่งมีพื้นฐานจากหลักการของภาษาโปรแกรมแบบออบเจ็กต์โอเรียนเต็ด และฐานข้อมูลรีเลชันนอลที่ซ้อนกัน (Nested Relational Database) ซึ่งรีเลชันหนึ่งอาจถูกซ้อนอยู่ภายในรีเลชันอื่นๆ

โครงสร้างของออบเจ็กต์ (Object Structure)

หลักการออบเจ็กต์โอเรียนเต็ดมีพื้นฐานคือ การรวม (encapsulation)เอาโค้ ทและข้อมูลเข้าเป็นหน่วยๆหนึ่งที่เรียกว่า *ออบเจ็กต์ (object)* การติดต่อระหว่างออบเจ็กต์หนึ่งและระบบส่วนที่เหลือจะถูกกำหนดด้วยเซตของแมสเซจ (message) ตัวอย่างเช่น ในฐานข้อมูลของเอกสาร การสร้างเอกสารหนึ่งต้องใช้ซอร์ฟแวร์จัดระเบียบตัวอักษรที่มีหลายฟังก์ชัน เมื่อจะพิมพ์เอกสาร ตัวตรวจสอบความถูกต้องจะทำงาน ภายใต้หลักการออบเจ็กต์โอเรียนเต็ด แต่ละเอกสารเป็นออบเจ็กต์ที่ประกอบด้วยตัวอักษร และโค้ทที่จะทำงานบนออบเจ็กต์นั้นเอง ทุกออบเจ็กต์เอกสารจะรับแมสเซจในการสั่งพิมพ์ แต่อาจจะทำงานต่างกัน โดยแต่ละเอกสารจะทำการเอ็กซีคิวท์โค้ทที่เหมาะสมของตัวเอง สิ่งที่เอ็นแคปซูเลชัน (encapsulation) หรือซ่อนอยู่ในออบเจ็กต์เอกสารได้แก่ ข้อมูลว่าจะพิมพ์เอกสารอย่างไร และเราสามารถทำให้ทุกเอกสารมีส่วนติดต่อกับผู้ใช้ (user interface) ที่เหมือนกันได้

ปกติออบเจ็กต์จะเกี่ยวข้องกับ

- เซตของตัวแปรที่บรรจุข้อมูลสำหรับออบเจ็กต์โดยค่าของแต่ละตัวแปรขึ้นอยู่กับออบเจ็กต์
- เซตของแมสเซจที่ออบเจ็กต์ตอบสนอง
- เมททอด (method) ซึ่งคือส่วนของโค้ทที่ถูกเอ็กซีคิวท์เมื่อได้รับแมสเซจ และเมททอดจะส่งค่ากลับเป็นการตอบสนองต่อแมสเซจ

แมสเซจไม่ได้หมายถึง ฟิสิกคอลแมสเซจ (physical message)ในคอมพิวเตอร์เน็ตเวิร์ค แต่หมายถึงการส่งผ่านคำร้องขอระหว่างออบเจ็กต์ต่างๆโดยไม่ได้สนใจในรายละเอียดการอิมพลีเมนท์ (implement) ส่วนติดต่อกับผู้ใช้ของออบเจ็กต์ มีเพียงเซตของแมสเซจที่มันตอบสนอง เราจึงสามารถแก้ไขนิยามของเมททอด และตัวแปรของออบเจ็กต์ได้โดยไม่มีผลต่อออบเจ็กต์อื่นๆ ความสามารถนี้เป็นข้อดีมากอย่างหนึ่งของการเขียนโปรแกรมแบบออบเจ็กต์โอเรียนเต็ด

แผนภาพคลาส (Class Hierarchy)

มีออบเจ็กต์ที่เหมือนๆกันในฐานข้อมูล เรามองในแง่ว่า มันตอบสนองต่อแมสเซจเดียวกันและใช้เมททอดเดียวกัน และมีตัวแปรชื่อและชนิดเดียวกัน จะเป็นการเสียเวลาที่จะกำหนดออบเจ็กต์แต่ละอันแยกกัน เราจึงกรุพออบเจ็กต์ที่เหมือนๆกันนี้สร้างเป็นคลาสแต่ละออบเจ็กต์นั้นเรียกว่า **อินแสตนท์** (instance)ของคลาส ทุกออบเจ็กต์ในคลาสจะแชร์คอมมอนเด็ฟฟินิชชัน (common definition) แต่มันจะแตกต่างกันที่ค่าของตัวแปรในออบเจ็กต์

ตัวอย่างของคลาสในฐานข้อมูลธนาคาร เช่น ลูกค้า, บัญชี(account), การกู้ยืม (loan) จากนิยามของคลาสที่กล่าวมาจะเหมือนนิยามของแอบสแตรทดาต้าไทป์ (abstract data type) แต่คลาสจะมีคุณสมบัติเพิ่มเติมนอกเหนือจากของแอบสแตรทดาต้าไทป์

ฐานข้อมูลแบบออบเจ็กต์โอเรียนเต็ดต้องการคลาสจำนวนมาก แต่คลาสต่างๆอาจคล้ายกัน เช่น สมมติว่าเรามีฐานข้อมูลแบบออบเจ็กต์โอเรียนเต็ดสำหรับงานธนาคาร ซึ่งมีคลาสของลูกค้า และคลาสของลูกจ้างธนาคารที่มีตัวแปรเหมือนกันคือ ชื่อ, ที่อยู่, เบอร์โทรศัพท์ และอื่นๆ แต่อาจมีตัวแปรอื่นๆที่เป็นเฉพาะของลูกค้า เช่น เครดิตเรทติ้ง (credit rating) และตัวแปรเฉพาะของลูกจ้าง เช่นเงินเดือน เราน่าจะนิยามตัวแปรคอมมอน (common variable) ไว้ในที่เดียวกัน ซึ่งทำได้โดยให้ลูกค้าและลูกจ้างรวมอยู่ในกลุ่มเดียวกันเป็นคลาสๆหนึ่ง เราสามารถเขียนแผนภาพคลาส สำหรับงานธนาคารได้

ออบเจ็กต์โอเรียนเต็ดคิวรี่ (Object-Oriented Queries)

ภาษาการเขียนโปรแกรมแบบออบเจ็กต์โอเรียนเต็ดต้องการการติดต่อระหว่างออบเจ็กต์โดยผ่านแมสเซจ ทำให้เกิดข้อจำกัดแก่แอพพลิเคชันฐานข้อมูล ขอยกตัวอย่างคิวรี่ต่อไปนี้ "หาระบบคอมพิวเตอร์ทั้งหมดที่ใช้ชิพที่ขายโดยบริษัทอินเทล"ถ้าเราทำตามหลักการเขียนโปรแกรมแบบออบเจ็กต์โอเรียนเต็ด แมสเซจจะถูกส่งไปให้ทุกอินแสตนท์ของคลาสชิพเพื่อดูว่าผู้ขายของชิพนั้นๆคือใคร ถ้าเรามองการร้องขอ (request) นี้เป็นปัญหาของฐานข้อมูล เราจะต้องสร้างอินเด็กซ์สำหรับคลาสชิพเพื่อให้สามารถค้นหา "บริษัทอินเทล" และได้รับไอเดนติไฟเออร์ของอินแสตนท์ที่มีผู้ขายเป็นบริษัทอินเทล โดยตรงการประมวลผลคิวรี่เป็นอย่างไรนั้นขึ้นกับรีเลชันนอลวิวของฐานข้อมูลแบบออบเจ็กต์นั้น นอกจากนี้เราสามารถส่งคิวรี่ที่มีการจอยน์ (join) กันระหว่างเซตของออบเจ็กต์ก็ได้

ภาษาทำคิวรี่สำหรับระบบฐานข้อมูลแบบออบเจ็กต์โอเรียนเต็ดมีหลายภาษา แต่ยังไม่มีแสตนดาร์ดที่ชัดเจนออกมา

ภาษาออบเจ็กต์โอเรียนเต็ด (Object-Oriented Languages)

ระบบฐานข้อมูลออบเจ็กต์โอเรียนเต็ดมีภาษาเฉพาะในโมเดลของมันเพื่อใช้ในการเขียนโค้ท ภาษาเหล่านี้มีข้อจำกัดให้ใช้แอคเซสได้เฉพาะตัวแปรของอินแสตนท์ที่โลคอล และใช้ส่งแมสเซจไปให้ออบเจ็กต์อื่นๆได้ และมีความสามารถในการทำคิวรี่และเซตโอเรียนเต็ดโอเปอเรชัน (set-oriented operation)

ภาษายุคที่ 4 (Fourth Generation Languages)

การเขียนโปรแกรมฐานข้อมูลโดยใช้เอสคิวแอลเอ็มเบ็ด (SQL embedded) ในภาษาโปรแกรมมิ่งทั่วๆไป (General-Purpose Programming Language) หรือเรียกว่า ภาษาโฮสท์ ทำให้โปรแกรมที่เขียนขึ้นสามารถแอคเซสฐานข้อมูลได้ แต่มันไม่สามารถแสดงผลลัพธ์แก่ผู้ใช้ หรือช่วยสร้างรายงานได้เลย ผลิตภัณฑ์ฐานข้อมูลในตลาดส่วนใหญ่จึงมีภาษาเฉพาะให้ใช้ เพื่อช่วยโปรแกรมเมอร์ในการเสร้างเท็มเพลตหลายๆอันบนหน้าจอสำหรับติดต่อกับผู้ใช้งานหนึ่ง และช่วยโปรแกรมเมอร์ในการสร้างรายงาน ภาษาพิเศษนี้เรียกว่า ภาษายุคที่ 4

ภาษายุคที่ 4 บางภาษายังมีโครงสร้างระดับสูงที่ช่วยในการเขียนลูปอ่านเรคคอร์ดโดยไม่ต้องให้โปรแกรมเมอร์จัดการเรื่องเคอร์เซอร์ อย่างไรก็ตามยังไม่มีมาตราฐานของภาษายุคที่ 4 เหมือนกับเอสคิวแอล และเอ็มเบ็ดเอสคิวแอล เพราะบริษัทต่างๆต่างพัฒนาภาษาของตัวเองขึ้นมา

2.4 โอเพนดาต้าเบสคอนเน็คติวิตี (โอดีบีซี) (Open Database Connectivity, ODBC)

ก่อนที่จะมีการพัฒนาโอดีบีซีขึ้นมาใช้ในการพัฒนาแอพพลิเคชันนั้น การพัฒนาแอพพลิเคชันเมื่อก่อนจะขึ้นกับดีบีเอ็มเอสโดยจะใช้เอ็มเบดเอสคิวแอล ซึ่งประสิทธิภาพของเอ็มเบดเอสคิวแอลนี้จะขึ้นอยู่กับฮาร์ดแวร์และสภาวะแวดล้อมของระบบปฏิบัติการ ซึ่งทำให้ซอสโค้ดจะต้องถูกคอมไพล์ (compile) ใหม่สำหรับแต่ละสภาวะแวดล้อม

การที่แอพพลิเคชันใช้โอดีบีซีเป็นตัวติดต่อเพื่อการเข้าถึงข้อมูลในดีบีเอ็มเอสนั้นจะใช้ภาษาเอสคิวแอล เป็นมาตราฐานสำหรับการเข้าถึงข้อมูล การทำเช่นนี้จะทำให้แอพพลิเคชันสามารถใช้ได้กับดีบีเอ็มเอสที่แตกต่างกันได้ ซึ่งทำให้ผู้พัฒนาแอพพลิเคชันสามารถนำแอพพลิเคชันที่พัฒนาแล้วไปใช้กับดีบีเอ็มเอสตัวใดก็ได้ โดยผู้ใชสามารถเพิ่มเติมโมดุล (Dadabase Driver) ที่จะลิ้งค์แอพพลิเคชันให้ใช้กับดีบีเอ็มเอสที่ต้องการได้

2.4.1 การติดต่อโดยใช้โอดีบีซี (ODBC Interface)

การใช้โอดีบีซีในการติดต่อนั้นจำเป็นต้องมี

- ไลบรารี่ของโอดีบีซีฟังชันคอล ซึ่งจะทำให้แอพพลิเคชันติดต่อกับดีบีเอ็มเอสเพื่อที่จะเอ็กซีคิวท์คำสั่งแล้วดึงข้อมูลขึ้นมา
- เซตมาตราฐานของโค้ดที่แสดงข้อผิดพลาด
- กฎเอสคิวแอล (SQL syntax) บน X/Open และข้อกำหนดเอสคิวแอลแอคเซสกรุ๊ป (SQL Access Group, SAG) เอสคิวแอลซีเออี (SQL CAE)
- วิธีมาตราฐานในการติดต่อและการล็อคเข้าดีบีเอ็มเอส

2.4.2 ส่วนประกอบของโอดีบีซี

โครงสร้างของโอดีบีซีจะประกอบด้วยส่วนประกอบ 4 ส่วน ดังแสดงในรูปที่ 2.11

แอพพลิเคชัน จะทำการประมวลผลและเรียกโอดีบีซีฟังก์ชัน เพื่อส่งประโยคภาษาเอสคิวแอลไปเอ็กซีคิวท์และ ดึงข้อมูลออกมา

ไดรเวอร์เมเนเจอร์ (Driver Manager) จะทำการโหลดไดรเวอร์

ไดรเวอร์ (Driver) จะประมวลผลโอดีบีซีฟังกัชันคอล (ODBC Function Call) โดยส่งคำสั่งเอสคิวแอล ไปยังดาต้าซอสนั้นๆ แล้วส่งผลลัพธ์กลับมายังแอพพลิเคชัน ซึ่งถ้าจำเป็นจริงๆแล้ว ไดร์เวอร์จะทำการเปลี่ยนแปลงคำสั่งนั้นๆของแอพพลิเคชันในกรณีที่ไม่ถูกต้องตามกฎ

ดาต้าซอส (Data source) ประกอบด้วยข้อมูลที่ผู้ใช้ต้องการจะเข้าถึง ระบบปฏิบัติการที่เกี่ยวข้อง ดีบีเอ็มเอส และเน็ตเวิร์คแพลตฟอร์ม (network Platform) (ถ้ามี)

แอพพลิเคชัน แอพพลิเคชันจะใช้การติดต่อกับโอดีบีซี ทำได้ดังนี้

- ทำการร้องขอ (Request) ไปยังดาต้าซอส (Data source)
- ส่งเอสคิวแอลไปยังดาต้าซอส
- กำหนดเนื้อที่และรูปแบบของข้อมูลสำหรับเก็บผลลัพธ์จากการประมวลผลคำสั่งเอสคิวแอล
- ร้องขอผลลัพธ์
- ประมวลผลข้อผิดพลาดที่อาจจะเกิดขึ้น
- รายงานผลลัพธ์กลับไปยังผู้ใช้ (ถ้าจำเป็น)
- ร้องขอการคอมมิทและโรวแบคสำหรับการควบคุมทรานแสคชัน
- ออกจากการติดต่อจากดาต้าซอส

ไดรเวอร์เมเนเจอร์

ไดรเวอร์เมเนเจอร์เป็นไดนามิกลิงค์ไลบรารี่ (ดีแอลแอล) (Dynamic - liink library,DLL) จุดประสงค์หลักของ ไดรเวอร์เมเนเจอร์คือการโหลดไดรเวอร์ โดยจะทำตามลำดับดังนี้

- ใช้ ODBC.INI ไฟล์เพื่อแมพชื่อของดาต้าซอสเพื่อค้นหาไดรเวอร์ดีแอลแอล
- ทำการประมวลผลโอดีบีซีอินนิเชียลไลเซชันคอล (ODBC Initialization call)
- จัดการโอดีบีซีฟังก์ชันสำหรับแต่ละไดรเวอร์
- จัดการพารามิเตอร์ให้ถูกต้องสำหรับการเรียกใช้โอดีบีซี

ไดรเวอร์

ไดรเวอร์เป็นดีแอลแอลที่อิมพลีเมนต์โอดีบีซีฟังก์ชันคอลและทำการติดต่อกับข้อมูลที่ดาต้าซอส โดยไดรเวอร์เมเนเจอร์จะทำการโหลดไดรเวอร์เมื่อแอพพลิเคชันได้เรียกฟังก์ชัน SQLBrowseConnect SQLConnect หรือ SQLDriverConnect โดยไดรเวอร์จะกระทำตามขั้นตอนดังต่อไปนี้เพื่อตอบรับการเรียกโอดีบีซีฟังก์ชันของแอพพลิเคชัน

- สร้างการติดต่อไปยังดาต้าซอส
- ส่งการร้องขอไปยังดาต้าซอส
- แปลงข้อมูลให้เป็นรูปแบบอื่นถ้าเป็นการเรียกโดยแอพพลิเคชัน
- ส่งผลลัพธ์กลับไปยังแอพพลิเคชัน
- ถ้าเกิดข้อผิดพลาดขึ้นจะส่งโค้ดข้อผิดพลาดกลับไปยังแอพพลิเคชันที่เรียกมา

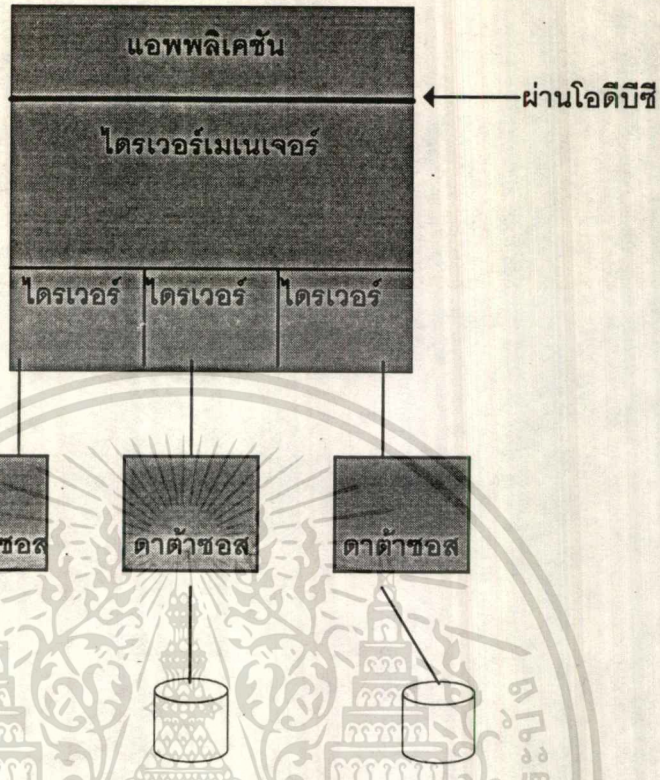
ประกาศและทำการมานิพูเลทเคอเซอร์ถ้าจำเป็นต้องใช้ ซึ่งแอพพลิเคชันจะมองไม่เห็นถ้าไม่มีการร้องขอเพื่อการเข้าถึงข้อมูล

รูปแบบของไดรเวอร์

โอดีบีซีได้กำหนดรูปแบบไดรเวอร์ได้ 2 รูปแบบ

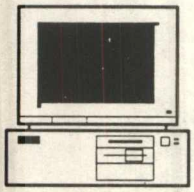
- ชิงเกิลไทเออร์ ไดรเวอร์จะประมวลผลทั้งการเรียกโอดีบีซีและประโยคเอสคิวแอล ดาต้าเบสไฟล์จะถูกประมวลผลโดยไดรเวอร์ ไดรเวอร์จะทำการประมวลผลประโยคเอสคิวแอลแล้วดึงข้อมูลจากฐานข้อมูล ไดรเวอร์ประเภทนี้จะจำกัดเซตของประโยคภาษาเอสคิวแอล ดังรูปที่ 2.12
- มัลติเพิลไทเออร์ ไดรเวอร์จะประมวลผลการเรียกโอดีบีซีและส่งประโยคเอสคิวแอลไปให้ดาต้าซอส โดยมีรูปแบบการติดตั้ง 3 ประเภท ดังนี้
    - การติดตั้งอาจจะอยู่บนเครื่องเดียวกัน
    - มีการใช้ข้ามแพลตฟอร์ม โดยแอพพลิเคชัน ไดรเวอร์ และไดรเวอร์เมเนเจอร์อาจจะอยู่อีกระบบหนึ่งและฐานข้อมูลและซอฟแวร์ที่ใช้ในการเข้าถึงอยู่อีกระบบหนึ่ง
    - เป็นโครงสร้างเกตเวย์ ไดรเวอร์จะส่งคำสั่งเอสคิวแอลไปให้เกตเวย์แล้วเกตเวย์ก็จะส่งต่อให้กับดาต้าซอส

โดยระบบหนึ่งสามารถมีไดรเวอร์ได้ทั้ง 2 รูปแบบ

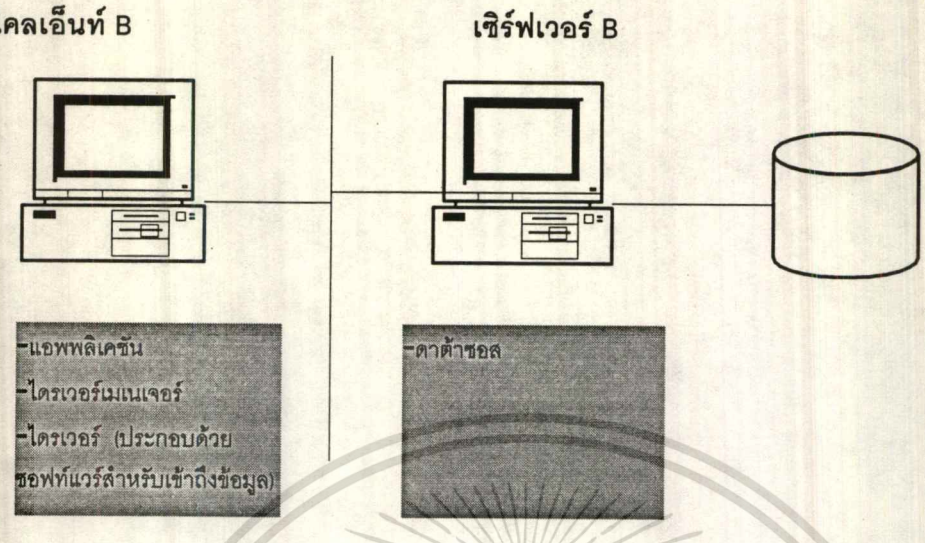


รูปที่ 2.11 แสดงส่วนประกอบของโอดีบีซี

ระบบ A



-แอพพลิเคชัน
-ไดรเวอร์เมเนเจอร์(ประกอบด้วย
ซอฟต์แวร์ที่ใช้ในการเข้าถึงข้อมูล)
-ดาต้าสตอเรจ



รูปที่ 2.12 แสดงไดรเวอร์ประเภทซิงเกิลไทเออร์ บนระบบ A และ B

โดยโดเมนของระบบจัดการห้องสมุดจะเรียกใช้โดเมนของระบบการแสดงผลเพื่อแสดงสภาวะของระบบ และเก็บรวบรวมอินพุทของโอเปอร์เรเตอร์จากคอนโซลหรืออุปกรณ์อินพุท

หลังจากได้กำหนดโดเมนต่าง ๆ ในระบบแล้ว จึงมาวิเคราะห์แต่ละโดเมนโดยใช้แบบจำลองรูปภาพ(Graphical OOA Models) เป็นเครื่องมือในการพัฒนา ซึ่งต้องทำการสร้างแบบจำลองทั้ง 3 เรียงตามลำดับกันไป ดังนี้

- อินฟอร์เมชันโมเดล (Information Model)
- สเตทโมเดล (State Model)
- โพรเซสโมเดล (Process Model)

ถึงแม้ว่าการพัฒนาซอฟแวร์จะสามารถทำได้หลายวิธี (methodologies) และล้วนแต่เป็นวิธีที่น่าสนใจแต่โปรเจ็กต์ส่วนใหญ่โดยเฉพาะโปรเจ็กต์ขนาดใหญ่มักจะต้องการรายงานความคืบหน้าโปรเจ็กต์ด้วย waterfall model ซึ่งเป็นโมเดลพื้นฐานที่ประกอบด้วยเวิร์คเฟส (work phases) ต่าง ๆ ได้แก่
อนาไลซิสเฟส (Analysis Phase) สร้างโมเดลการวิเคราะห์ (OOA Models) ของแอพพลิเคชันโดเมน
ดีไซน์เฟส (Design Phase) สร้างโมเดลการวิเคราะห์ของโดเมนโครงสร้างและเซอร์วิสโดเมน
อิมพลีเมนเตชันเฟส (Implementation Phase) เป็นการแปลงแอพพลิเคชันโดเมน, เซอร์วิสโดเมน และโดเมนโครงสร้างเป็นโค้ด

ในการทำวิทยานิพนธ์ครั้งนี้เราก็จะใช้โมเดลนี้ในการแบ่งขั้นตอนการทำงานออกเป็นส่วนๆ เพื่อให้ง่ายแก่การทำความเข้าใจด้วยเช่นกัน

3.1 การวิเคราะห์แบบออบเจ็กต์โอเรียนเต็ด (Object-Oriented Analysis)

กล่าวโดยรวมแล้ว จะหมายถึง วิธีการเพื่อการวิเคราะห์หาเอ็นติตี้ (entities) ในขอบเขตของปัญหา (Application Domain) แล้วทำความเข้าใจและอธิบายว่าเอ็นติตี้เหล่านั้นติดต่อกันอย่างไร วิธีการนี้มักใช้ในซอฟแวร์เอ็นจิเนียริ่ง (Software engineering) โดยมี 3 ขั้นตอนตามลำดับ

3.1.1 อินฟอร์เมชันโมเดล (Information Models)

เมื่อวิเคราะห์ได้โดเมนแต่ละโดเมนแล้ว ลำดับต่อไปก็จะเป็นการวิเคราะห์หาเอ็นติตี้ ที่มีอยู่ในโดเมนนั้น จากการวิเคราะห์นี้สิ่งที่ได้ออกมาคือออบเจ็กต์ และ แอททริบิวท์ (Attribute) และวิเคราะห์หาความสัมพันธ์ระหว่างเอ็นติตี้ (Relationship) ซึ่งจะมีศัพท์เทคนิคที่ใช้ในโมเดลโดยจะกำหนดคำจำกัดความดังต่อไปนี้

บทที่ 3
วิธีที่ใช้ในการพัฒนาระบบงานประยุกต์

โปรแกรมประยุกต์ระบบงานยืมคืนของห้องสมุด ซึ่งใช้เป็นตัวอย่างในการอธิบายวิธีที่ใช้ในการออกแบบออบเจ็กต์โอเรียนเต็ด (Object-Oriented Analysis & Design) นี้เป็นระบบที่สร้างขึ้นเพื่อใช้ในการยืมและคืนหนังสือหรือวารสาร (รวมแล้วเรียกว่า สิ่งพิมพ์) ในห้องสมุดได้ โดยเมื่อจะมีการยืมให้ใส่รหัสผู้ยืม และรหัสสิ่งพิมพ์และเมื่อจะมีการคืนให้ใส่รหัสสิ่งพิมพ์เท่านั้น และทุกครั้งที่เกิดการยืมหรือคืนจะมีการเปลี่ยนแปลงตารางต่างๆที่เกี่ยวข้องในฐานข้อมูล

วิธีที่ใช้จะใช้หลักการของออบเจ็กต์โอเรียนเต็ด โดยจะทำการวิเคราะห์ปัญหาโดยสร้างโดเมน (Domain) ต่าง ๆ ขึ้นก่อน แต่ละโดเมนจะแยกออกจากกันและมีความสัมพันธ์กันในโดเมนเดียวกันเท่านั้น คุณสมบัติของโดเมนคือ

- โดเมนจะมีออบเจ็กต์ใดๆ (object) เป็นของตัวเองเท่านั้นไม่ซ้ำกับโดเมนอื่น
- ออบเจ็กต์หนึ่งในโดเมนจะถูกกำหนดขึ้นได้เมื่อมีออบเจ็กต์ที่เกี่ยวข้องในโดเมนเดียวกัน เช่น สิ่งพิมพ์ จะไม่มีความหมายถ้าไม่มีผู้ยืม
- ออบเจ็กต์ในโดเมนหนึ่งสามารถกำหนดขึ้นได้โดยไม่ต้องมีออบเจ็กต์ในโดเมนอื่น ๆ เช่น สิ่งพิมพ์หรือนักเรียนสามารถกำหนดให้มีได้โดยไม่จำเป็นต้องมีวินโดวส์หรือไอคอน

โดเมนต่าง ๆ ในระบบสามารถจำแนกออกเป็น 4 ประเภทตามบทบาทหน้าที่ในระบบดังนี้

แอพพลิเคชันโดเมน (Application Domain) ในระบบยืมคืนห้องสมุดมี 1 แอพพลิเคชันโดเมน คือ โดเมนของระบบจัดการห้องสมุด (Library Management Domain)

เซอร์วิสโดเมน (Service Domain) เช่นยูสเซอร์อินเทอร์เฟสโดเมน (User Interface Domain), โดเมนที่ทำโพรเซสอินพุทเอาท์พุท (Process Input/Output (PIO) Domain)

โดเมนโครงสร้าง (Architectural Domain) เช่นในระบบยืมคืนห้องสมุดมี 1 โดเมนโครงสร้างคือกลไกหรือโครงสร้างสำหรับจัดการข้อมูลและควบคุมการดำเนินไปของระบบทั้งหมด เช่น FSM class

อิมพลีเมนเตชันโดเมน (Implementation Domain) เช่น ภาษาโปรแกรม (Programming Language Domain), เน็ตเวิร์ค (Network Domain), ระบบปฏิบัติการ (Operating System Domain)

โดเมนต่าง ๆจะมีความสัมพันธ์กันแบบไคลเอ็นท์เซิร์ฟเวอร์ คือผู้ใช้และผู้ถูกเรียกใช้ตามลำดับ เราจะแสดงความสัมพันธ์ของโดเมนต่าง ๆในรูปแผนภาพโดเมน (Domain Chart) เช่นโดเมนของระบบการจัดการห้องสมุด (Library Management Domain) ก็จะเกี่ยวกับสิ่งพิมพ์, นักเรียน ขณะที่ยูสเซอร์อินเตอร์เฟสโดเมน (User Interface Domain) ก็จะเกี่ยวกับวินโดวส์ ระบบการแสดงผล และไอคอน เป็นต้น

ออบเจ็กต์ คือ เซตของอินแสตนท์ ซึ่งของทั้งหมดจะอยู่ในรูปของเซตของสิ่งที่มีคุณลักษณะเดียวกัน เช่น หนังสือวิทยาศาสตร์ และนิตยสารคอมพิวเตอร์ ก็จะอยู่ในออบเจ็กต์สิ่งพิมพ์เช่นเดียวกัน ออบเจ็กต์ต่างๆจะต้องมีชื่อที่ไม่ซ้ำกันและเป็นคำที่มีความหมายรวมสมาชิกทั้งหมดไว้ด้วยกัน

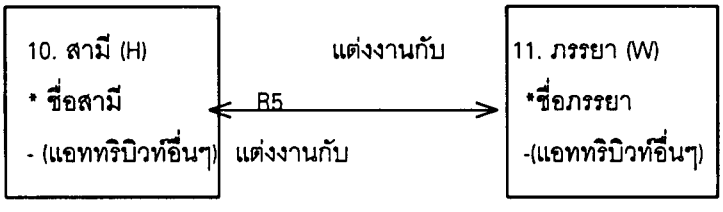
อินแสตนท์ (Instance) คือ สิ่งที่มีคุณลักษณะเดียวกัน อยู่เซตเดียวกันและอยู่ภายใต้กฎและระเบียบเดียวกัน เช่น หนังสือเล่มที่มีเลขทะเบียน 021600

แอททริบิวท์ คือ คุณลักษณะหนึ่ง ๆ ที่ทุกอินแสตนท์มีเหมือนกัน ในการกล่าวถึงแอททริบิวท์จะเขียนในรูปแบบ <object name>.<attribute name> เช่น student.id#

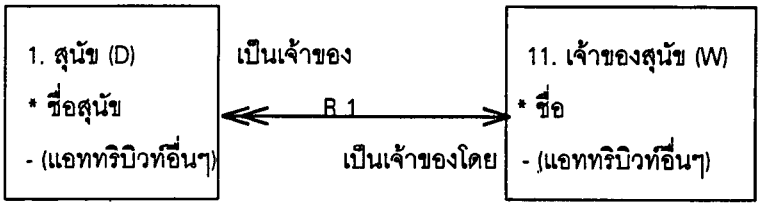
ไอเดนติไฟเออร์ (Identifier) คือเซตของแอททริบิวท์ จะมีค่าไม่ซ้ำกัน (Unique) ในแต่ละอินแสตนท์ของแต่ละออบเจ็กต์ เช่นการตั้งรหัสนักเรียนแต่ละคน และไอเดนติไฟเออร์อาจมีมากกว่า 1 ตัวก็ได้ แต่จะต้องเลือกไอเดนติไฟเออร์ขึ้นมา 1 ตัวให้เป็นพรีเฟอร์ไอเดนติไฟเออร์ (Preferred Identifier)

ความสัมพันธ์ (Relationship) คือการแสดงความเกี่ยวข้องระหว่างออบเจ็กต์ ดังแสดงในรูปที่ 3.1
ความสัมพันธ์แบบไม่มีเงื่อนไข (Unconditional Relationship) ทุก ๆอินแสตนท์(Instance) จะต้องมีส่วนเกี่ยวข้องในความสัมพันธ์นั้น ได้แก่ความสัมพันธ์ วันทูวัน (one to one), เมนี่ทูวัน (many to one) และ เมนี่ทูเมนี่ (many to many) โดยจะแสดงในรูปที่ 3.2 , 3.3 และ 3.4 ตามลำดับ

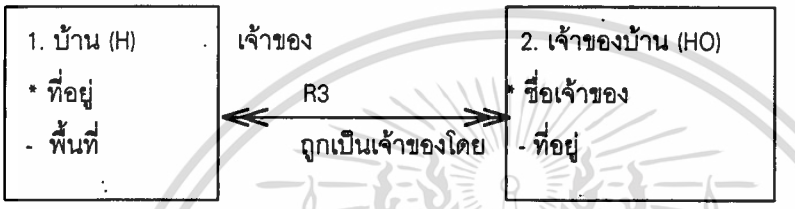
รูปที่ 3.1 แสดงความสัมพันธ์ที่เกี่ยวข้องระหว่างออบเจ็กต์



รูปที่ 3.2 แสดงความสัมพันธ์แบบวันทูวัน (one to one)

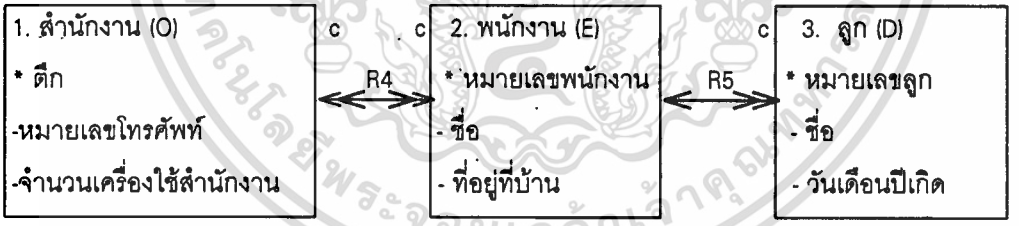


รูปที่ 3..3 แสดงความสัมพันธ์แบบเมนี่ทูวัน (many to one)



รูปที่ 3.4 แสดงความสัมพันธ์แบบเมนี่ทูเมนี่ (many to many)

ความสัมพันธ์แบบมีเงื่อนไข (Conditional Relationship) ในความสัมพันธ์แบบมีเงื่อนไข อาจมีอินแสตนท์บางตัวที่ไม่ต้องมีส่วนเกี่ยวข้องกับความสัมพันธ์ก็ได้ จะแสดงได้โดยใส่ C ที่ปลายลูกศรข้างที่ต้องการ ดังตัวอย่างในรูปที่ 3.5

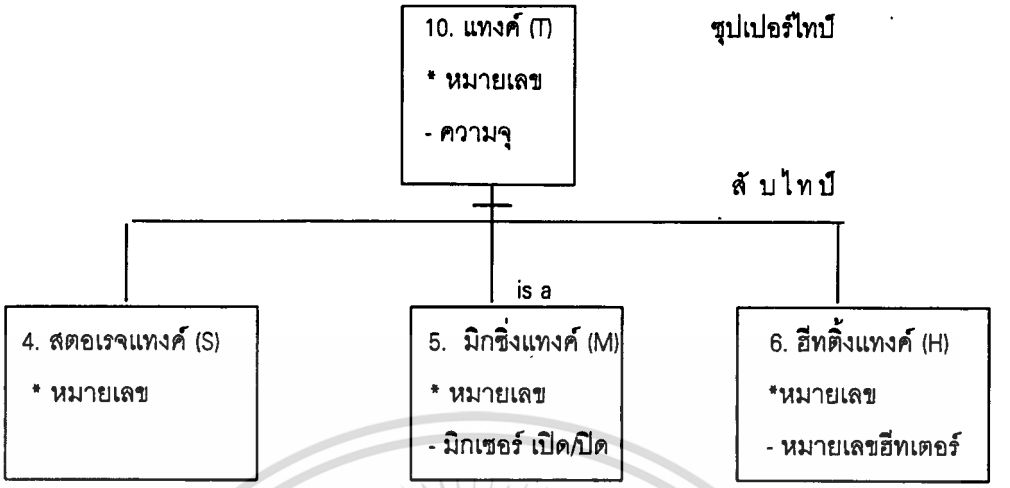


รูปที่ 3.5 แสดงความสัมพันธ์แบบมีเงื่อนไข (Conditional Relationship)

สับไทป์และซุปเปอร์ไทป์ (Subtypes and Supertypes) ในปัญหาต่าง ๆ จะพบว่ามีออบเจ็กต์ที่สามารถใช้แอททริบิวท์ร่วมกันได้ ซึ่งในกรณีนี้สามารถแสดงความสัมพันธ์เป็นสับไทป์-ซุปเปอร์ไทป์ แสดงดังรูปที่ 3.6

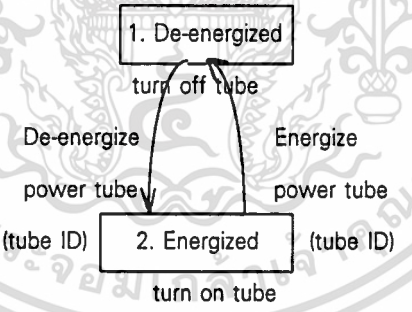
3.1.2 สเตทโมเดล (State Models)

ในขั้นตอนที่ 2 นี้จะเน้นเรื่องพฤติกรรมของออบเจ็กต์และความสัมพันธ์กันในช่วงเวลาทั้งหมด **ไลฟ์ไซเคิล** (Lifecycle) คือ ภาพแสดงพฤติกรรมของออบเจ็กต์ ซึ่งเป็นเช่นเดียวกันในทุกอินแสตนท์ด้วย มีความหมายเดียวกับสเตทโมเดล (state model)



รูปที่ 3.6 แสดงสับไทป์และซุปเปอร์ไทป์ของความสัมพันธ์ระหว่างออบเจ็กต์ที่ใช้แอททริบิวท์ร่วมกัน

3.1.2.1 เซตของสเตท (Set of state) ในไลฟ์ไซเคิล (Lifecycle) ของออบเจ็กต์ โดยจะแสดงสเตทในรูปดังนี้



รูปที่ 3.7 แสดงสเตทโมเดล

โดยมีคำศัพท์เทคนิคดังต่อไปนี้

ชื่อของสเตท และ หมายเลข (state names and numbers) แต่ละสเตทจะถูกตั้งชื่อและหมายเลขโดยจะต้องไม่ซ้ำกันภายในสเตทโมเดลหนึ่ง

สเตทเริ่มต้น (creation state) คือ สเตทแรกที่อินแสตนท์ถูกสร้างขึ้นในระบบ ซึ่งในสเตทโมเดลอาจจะมีสเตทมากกว่า 1 สเตท ที่เป็นสเตทเริ่มต้น และการเปลี่ยนสเตท (transition) เข้าสู่สเตทเริ่มต้นนี้จะไม่ได้มาจากสเตทใดเลย

สเตทสุดท้าย (final state) ในสเตทโมเดล สเตทที่เป็นสเตทสุดท้ายของไลฟ์ไซเคิล (lifecycle)ของอินแสตนท์เราจะเรียกว่าสเตทสุดท้าย (fina l state) สเตทสุดท้ายนี้อาจเกิดจาก 2 สถานการณ์ดังนี้

- อินแสตนท์หยุดการทำงานที่อยู่ในขอบข่ายที่สนใจแล้ว แต่ยังคงมีอยู่ต่อไป เช่น สเตทที่ 2 ในสเตทโมเดลของ assigner
- อินแสตนท์รวมทั้งสเตทแมชชีนของมันตายไป หรือ สูญสิ้นไป โดยแสดงสเตทนี้ด้วยการวาดกล่องด้วยเส้นประในสเตททรานสิชันไดอะแกรม (state transition diagram: STD) เช่น สเตทที่ 5 ในสเตทโมเดลของนักศึกษา (student) และ สเตทโมเดลของสิ่งพิมพ์ (publishing)

สเตทปัจจุบัน (current state) ในทุกๆขณะ อินแสตนท์ที่ต่างกันของออบเจ็กต์ สามารถมีสเตทที่ต่างกันได้ ซึ่งสเตทของอินแสตนท์ในขณะนั้นจะเรียกว่า สเตทปัจจุบัน (current state) ซึ่งจะแสดงในรูปของแอททริบิวท์หนึ่งคือ สเตตัส (status)

3.1.2.2 **เซตของเหตุการณ์** (Event) แต่ละเหตุการณ์ (Event) จะเป็นตัวบอกให้ออบเจ็กต์มีการเปลี่ยนสเตท

เหตุการณ์ (Event) เป็นคำจำกัดความของเหตุการณ์หรือสัญญาณที่เกิดขึ้นจริงและ เป็นการแสดงว่ามีออบเจ็กต์บางตัวในระบบกำลังจะเปลี่ยนสเตทของมันไปจากสเตทหนึ่งไปสู่อีกสเตทหนึ่ง

เหตุการณ์หนึ่งๆ คือ สัญญาณเพื่อควบคุมการเปลี่ยนสเตทของออบเจ็กต์ไปยังสเตทหนึ่ง ซึ่งจะหมายรวมถึงข้อมูลที่แอกชันที่สเตทนั้นต้องการด้วย เช่น เตาไมโครเวฟ สามารถเกิดเหตุการณ์ต่างๆเช่น ปุ่มถูกกด,อบเสร็จแล้ว,ประตูเปิด,ประตูปิด

เหตุการณ์ใดเหตุการณ์หนึ่งต้องมีเลเบล (label) ของมันแตกต่างจากเหตุการณ์อื่นๆ โดยเฉพาะเหตุการณ์ 2 เหตุการณ์ที่เหมือนกัน แต่ยังเป็นคนละเหตุการณ์ ก็จะต้องมีเลเบลต่างกันด้วย

ข้อมูลของเหตุการณ์ (event data) แบ่งเป็น 2 ชนิด

ข้อมูลระบุ (identifier data) คือ เซตของแอททริบิวท์รวมทั้งพรีเฟอร์ไอเดนติไฟเออร์ (perferred identifier) ด้วยเพื่อบ่งบอกอินแสตนท์ที่จะรับเหตุการณ์นั้นๆ และสามารถบอกสเตทแมชชีนที่จะรับเหตุการณ์นั้นๆด้วย

ข้อมูลบริการ (supplement data) คือ แอททริบิวท์ของออบเจ็กต์ใดก็ได้ ไม่จำเป็นต้องเป็นของออบเจ็กต์ที่จะรับเหตุการณ์นั้น

ข้อกำหนดบางประการ

ทุกเหตุการณ์ที่ทำให้เกิดการเปลี่ยนสเตทไปยังสเตทหนึ่งๆจะต้องบรรจุข้อมูลเหมือนๆกัน

ถ้าเหตุการณ์ทำให้เกิดการเปลี่ยนสเตทไปยังสเตทที่ไม่ใช่สเตทเริ่มต้น (creation state) เหตุการณ์นั้นจะต้องบรรจุข้อมูลคือ ข้อมูลระบุ (identifier data)ของอินแสตนท์ที่เหตุการณ์นั้นส่งไปด้วย

เหตุการณ์ที่ทำให้เกิดการเปลี่ยนสเตทเข้าไปสู่สเตทเริ่มต้น (creation state) จะไม่บรรจุข้อมูลระบุ (identifier data) ใดๆ ถ้าสเตทโมเดลที่รับเหตุการณ์นั้นสร้างไอเดนติไฟเออร์ (identifier) ในแอคชันของสเตทเริ่มต้น (creation state)

3.1.2.3 กฎการเปลี่ยนสเตท (Transition Rules) จะเป็นตัวกำหนดสเตทใหม่ที่จะเกิดขึ้นเมื่อออบเจ็กต์ได้รับเหตุการณ์ (Event)

แอคชัน (Actions) เป็นการกระทำที่จะต้องทำเมื่อออบเจ็กต์มาถึงสเตทนั้น และ จะมี 1 แอคชันต่อ 1 สเตทเท่านั้น

แอคชัน (action) คือการทำงานที่เกิดขึ้นเมื่ออินแสตนท์นั้นเปลี่ยนสเตทเข้าไปสู่สเตทหนึ่ง และ แอคชันต้องถูกทำให้เสร็จก่อนที่เหตุการณ์อื่นๆจะมาทำให้อินแสตนท์เปลี่ยนสเตทต่อไปอีก

แอคชันในแต่ละสเตทแมชชีนสามารถทำงานไปพร้อมๆกัน (simultaneously) ได้

ข้อแนะนำในการเขียนสเตทโมเดล

- ทุกสเตทมีจุดประสงค์ในการกำหนดแตกต่างกัน ถึงแม้ว่าลักษณะที่บ่งบอกภายนอกจะเหมือนกัน เช่น สเตทมิกซิงโพรดักต์ (Mixing Product) และ รินซิง (Rinsing) ทั้งสองสเตทจะมีสภาพมิกเซอร์เปิด และ ทุกวาล์วปิดเหมือนกัน แต่จุดประสงค์ในการกำหนดต่างกันจึงเป็นคนละสเตทได้ โดยใช้ชื่อเป็นตัวบอกความแตกต่าง
- ทุกสเตทมีสภาพแวดล้อม ถ้าเราเพ่งเล็งที่เครื่องบินที่กำลังวิ่งไปบนรันเวย์ เราไม่สามารถรู้ได้ว่าทำไมมันถึงกำลังวิ่งอยู่ แต่ถ้าดูที่สเตทก่อนหน้าหรือต่อไปจะพบว่า เครื่องบินที่กำลังบินอยู่นั้นกำลังพร้อมที่จะบินขึ้นหรือกำลังจะไปจอดที่เกท
- ควรให้ความสนใจที่แอคชันโดยทำให้แต่ละสเตทแตกต่างกันด้วยแอคชัน ซึ่งต้องทำงานได้ตามความต้องการ

ออบเจ็กต์ใดที่ต้องเขียนไลฟ์ไซเคิล

- เมื่ออินแสตนท์ของออบเจ็กต์นั้นถูกสร้างขึ้นหรือทำลายในขณะรัน
- เมื่อมีการรวบรวมแอททริบิวท์ของอินแสตนท์ไปเรื่อยๆ ตามสเตทต่างๆของไลฟ์ไซเคิลนั้นๆ
- เมื่อเป็นไลฟ์ไซเคิลที่มีการทำงาน (operation)
- เมื่อออบเจ็กต์ถูกกระทำตามลำดับขั้นตอนในโรงงาน

- เมื่อออบเจ็กต์แสดงการร้องขอหรือมีงานที่ต้องถูกทำ
- เมื่ออินแสตนท์ของออบเจ็กต์เกี่ยวข้องกับความสัมพันธ์ (relationships)
- เมื่อไลฟ์ไซเคิลของมันสัมพันธ์กับของออบเจ็กต์อื่นๆ

ออบเจ็กต์ใดที่ไม่ต้องเขียนไลฟ์ไซเคิล

- ออบเจ็กต์นั้นเป็นพาสซีฟ (passive) เช่น ตึก
- ออบเจ็กต์ที่เป็นตัวกำหนดคุณสักษณะ (specification object) เช่น มาตรฐานคุณภาพ, ตำราอาหาร, สารประกอบ

แบบจำลองแสดงการติดต่อระหว่างออบเจ็กต์ (object communication model)

หมายถึง ภาพแสดงการติดต่อสื่อสารด้วยเหตุการณ์ (event) ระหว่างสเตทโมเดลต่างๆ และออบเจ็กต์ต่างๆภายนอกระบบ (external entities) (เช่น โอเปอร์เรเตอร์, อุปกรณ์ทางกายภาพ, ออบเจ็กต์ในระบบย่อยอื่นๆ)

ชนิดของเหตุการณ์ที่มีในแบบจำลองนี้

- เหตุการณ์จากภายนอก (external event) สร้างโดยสิ่งภายนอกระบบ และรับโดยสเตทโมเดลใดๆในระบบ แบ่งเป็น
  - เหตุการณ์ที่ไม่ได้ขอ (unsolicited event) ไม่ได้เกิดขึ้นโดยการกระทำก่อนหน้านี้ในระบบ เช่น B1: Make batch in tank ถูกสร้างขึ้นโดยโอเปอร์เรเตอร์ เมื่อต้องการสร้างแบทช์ โดยระบบไม่ได้ถามความต้องการของโอเปอร์เรเตอร์เลย และไม่มีการแจ้งล่วงหน้าก่อนที่จะส่งเหตุการณ์นี้มา
  - เหตุการณ์ที่ขอ (solicited event) สร้างเพื่อตอบสนองการกระทำบางอย่างในระบบก่อนหน้านี้ เช่น ถ้ามีคำสั่งปิด (event) ไปยังวาล์ว วาล์วจะมีการตอบสนองกลับว่า "วาล์วได้ปิดลงแล้ว"
- เหตุการณ์จากภายใน (internal event) สร้างโดยสเตทโมเดลใดๆในระบบ

3.1.3 โพรเซสโมเดล

หรือเรียกว่า **แอคชันดาตาโฟลว์ไดอะแกรม** (action data flow diagram: ADFD) (**เอดีเอฟดี**) หมายถึง ภาพแสดงโพรเซส (processes) ในแอคชันและการติดต่อสื่อสารระหว่างกัน, แต่ละสเตทของสเตทโมเดลหนึ่งจะมี 1 แอคชันที่เกี่ยวข้อง ซึ่งประกอบด้วยหลายๆโพรเซส โพรเซสต่างๆที่ได้จากขั้นตอนนี้จะนำไปแปลงเป็นโอเปอเรชันของคลาสได้โดยตรงในการดีซายน์ (Design) เอดีเอฟดีนี้ยึดหลักดาต้าโฟลว์ไดอะแกรมของ ยัวดอน (Yourdon)

โพรเซส (process) หมายถึง การประมวลผลหนึ่งๆ ในแอคชัน โพรเซสส่วนใหญ่ต้องการอินพุทเพื่อที่จะทำงานต่างๆ และสร้างเอาท์พุทออกมาเป็นผลลัพธ์

การไหลของข้อมูล (data flow) ถ้าโพรเซสต้องการอินพุท แล้วข้อมูลที่จะส่งให้กับโพรเซสเรียกว่า การไหลของข้อมูลไปยังโพรเซส (data flow directed to the process) ถ้าโพรเซสสร้างข้อมูลเป็นเอาท์พุท ข้อมูลนั้นจะเรียกว่า การไหลของข้อมูลออกจากโพรเซส (data flow directed away from the process)

ข้อมูลถาวร (persistent data) คือ ข้อมูลที่ยังคงอยู่หลังจากเกิดการทำแอคชันเสร็จแล้ว ใน เอดีเอฟดี แสดงข้อมูลถาวรนี้ในรูปดาตาสโตร์ (data store) ซึ่งทางกายภาพจะคือตารางหนึ่งในดาตาเบส หรือ คือไฟล์หนึ่ง หรือ คือเซตของตัวแปรระบบ

ชนิดของโพรเซส

แบ่งตามจุดประสงค์ของโพรเซสและ การติดต่อกับดาตาสโตร์ (data store)

แอคเซสเซอร์ (accessors) คือ โพรเซสที่มีจุดประสงค์เพื่อแอคเซส (access) ข้อมูลในดาตาสโตร์ แบ่งเป็น

- เพื่อสร้าง (create accessor) สร้างอินแสตนท์ของออบเจ็กต์ขึ้นใหม่
- เพื่ออ่าน (read accessor) อ่านแอททริบิวท์ของออบเจ็กต์หนึ่ง
- เพื่อเขียน (write accessor) แก้ไขค่าแอททริบิวท์ของออบเจ็กต์หนึ่ง
- เพื่อทำลาย (delete accessor) ทำลายอินแสตนท์ของออบเจ็กต์

ชนิดของโพรเซสมีดังนี้

- แอคเซสเซอร์จะถูกกำหนดให้กับออบเจ็กต์ซึ่งตรงกับดาตาสโตร์ที่มันแอคเซส
- ผู้สร้างเหตุการณ์ (event generator) จะถูกกำหนดให้กับออบเจ็กต์ที่ได้รับเหตุการณ์นั้น
- การเปลี่ยนแปลง (transformation) คือ โพรเซสที่มีจุดประสงค์เพื่อประมวลผลหรือ เปลี่ยนแปลงข้อมูล (คือเปลี่ยนอินพุทให้อยู่ในรูปแบบใหม่แล้วเอาท์พุทออกมา)
- การทดสอบ (test) คือโพรเซสที่ทำการทดสอบเงื่อนไขและทำให้เงื่อนไขนั้นสามารถควบคุมเอาพุทที่ออกมาได้

ชนิดของโดเมน (Domain) ได้แก่

- แอพพลิเคชันโดเมน (Application Domain) เป็นโดเมนที่ผู้ใช้มองเห็น
- เซอร์วิสโดเมน (Service Domain) เป็นโดเมนที่มีฟังก์ชันและกระบวนการเป็นส่วนของมันเองไม่ขึ้นกับใคร เช่น ยูสเซอร์อินเตอร์เฟสโดเมน เป็นต้น
- โดเมนโครงสร้าง (Architectural Domain) เป็นโดเมนที่มีกระบวนการและโครงสร้างในการจัดการข้อ

มูลและควบคุมระบบเป็นของตัวเองไม่ขึ้นกับใคร ออบเจ็กต์ที่อยู่ในโดเมนนี้จะมีส่วนที่เป็นโครงสร้างของข้อมูลและโค้ดประกอบอยู่ด้วย

- อิมพลีเมนเทชันโดเมน (Implementation Domain) เป็นโดเมนที่ประกอบด้วยภาษาที่ใช้เขียนโปรแกรม เน็ตเวิร์ค, ระบบปฏิบัติการ และคลาสไลบรารี่ที่ใช้ร่วมกัน

3.2 การแปลงจากออบเจ็กต์โอเรียนเต็ดอนาไลซิสเป็นออบเจ็กต์โอเรียนเต็ดดีไซน์ (Transforming Object-Oriented Analysis tnto Object-Oriented Design)

แต่ละโปรแกรมในระบบจะประกอบด้วย (แสดงในรูปที่ 3.8)

- โปรแกรมหลัก (Main Program)
- คลาสโครงสร้าง 4 คลาส (Four architectural classes)
- แอพพลิเคชันคลาส (Application Class)

จากรูปที่ 3.8 จะเห็นว่าคลาสโครงสร้างจะประกอบด้วย ทรานสิชัน,ไฟไนท์สเตทโมเดล (Finite State Model),แอคทีฟอินแสตนท์ (Active Instance) โดยแสดงดังรูปที่ 3.9, 3.10, 3.11 โดยคลาสแอพพลิเคชันจะอินเฮอริดคลาสแอคทีฟอินแสตนท์ ซึ่งคลาสแอพพลิเคชันจะประกอบด้วยคลาสพาสซีฟ ดังแสดงในรูปที่ 3.12 และแอคทีฟ ดังแสดงในรูปที่ 3.13 โดยคลาสที่เป็นแอคทีฟจะต้องมีคลาสแอสซายเนอร์ดังแสดงในรูปที่ 3.14

กลไกของสเตทแมชชีนประกอบด้วย 3 คลาสโครงสร้าง ดังนี้

3.2.1 ทรานซิชันคลาส (transition class)

จะรวมเอาข้อมูลที่อธิบายถึงการเปลี่ยนสเตททุกอันของทุกแบบจำลองสถานะในโปรแกรม แต่ละอินแสตนท์ของการเปลี่ยนสเตทจะตรงกับเซล (ไม่รวมถึง 'can't happen' เซล) ในตารางการเปลี่ยนสเตท (state transition diagram)

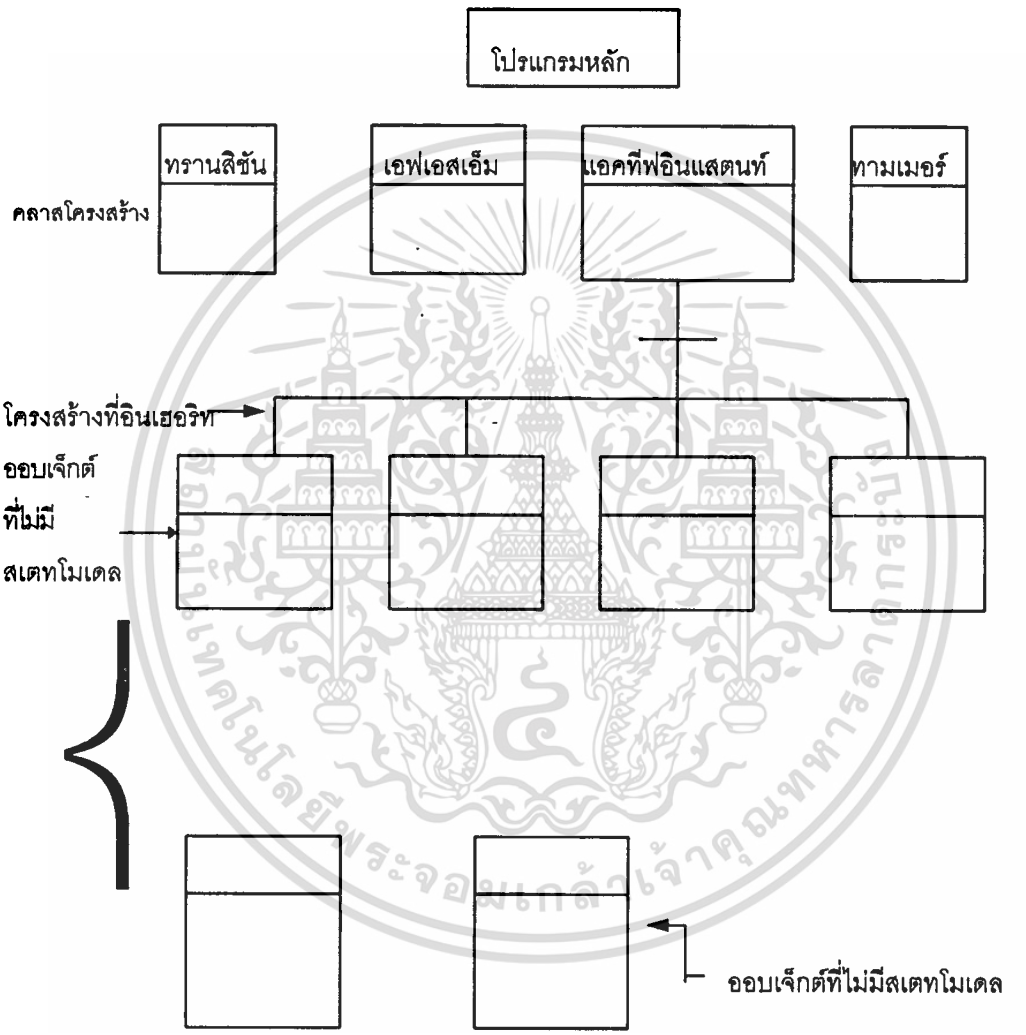
ถ้าเป็น สเตทใหม่ เซลจะมีหมายเลขสเตทใหม่อยู่ในเซล

ถ้าเป็น ไม่สนใจอีเวนท์ เซลหมายเลขสเตทใหม่จะเท่ากับ 0

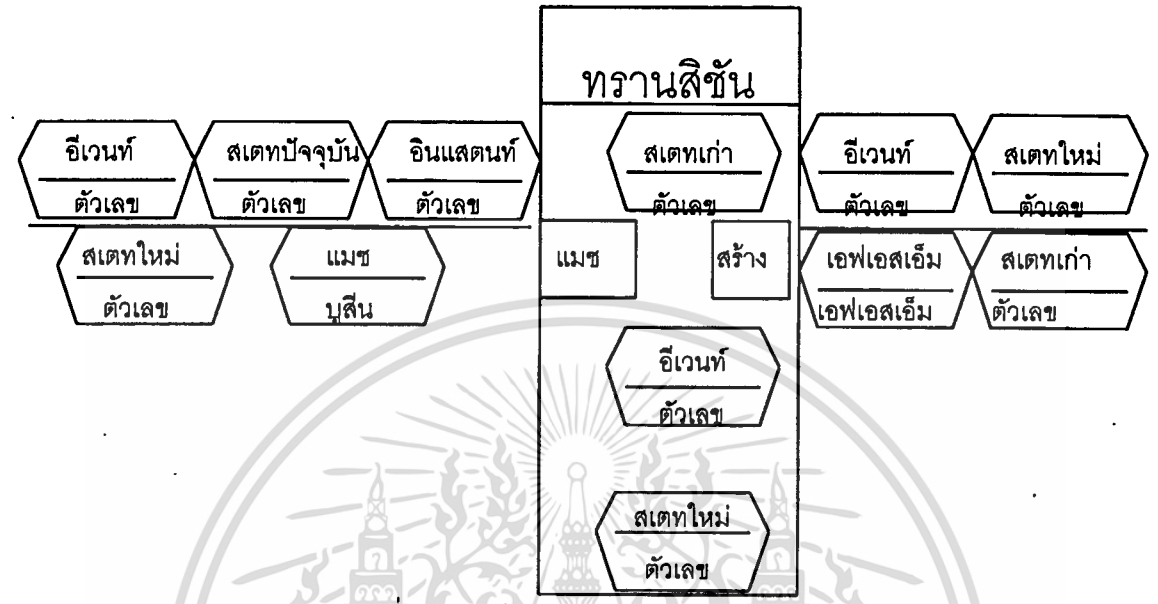
3.2.2 คลาสเอฟเอสเอ็ม (Finite State Machine Class)

เป็นตัวรวบรวมทุกทรานสิชันในแบบจำลองสถานะหนึ่งไว้ด้วยกัน โดรงสร้างข้อมูลอินแสตนท์ของเอฟเอสเอ็มและทรานสิชันจะขึ้นอยู่กับสภาพแวดล้อมในการพัฒนาโปรแกรม (development environment) คือภาษาและคลาสไลบรารีที่ใช้ (language and class libraries in use) ในทางทฤษฎี โครง

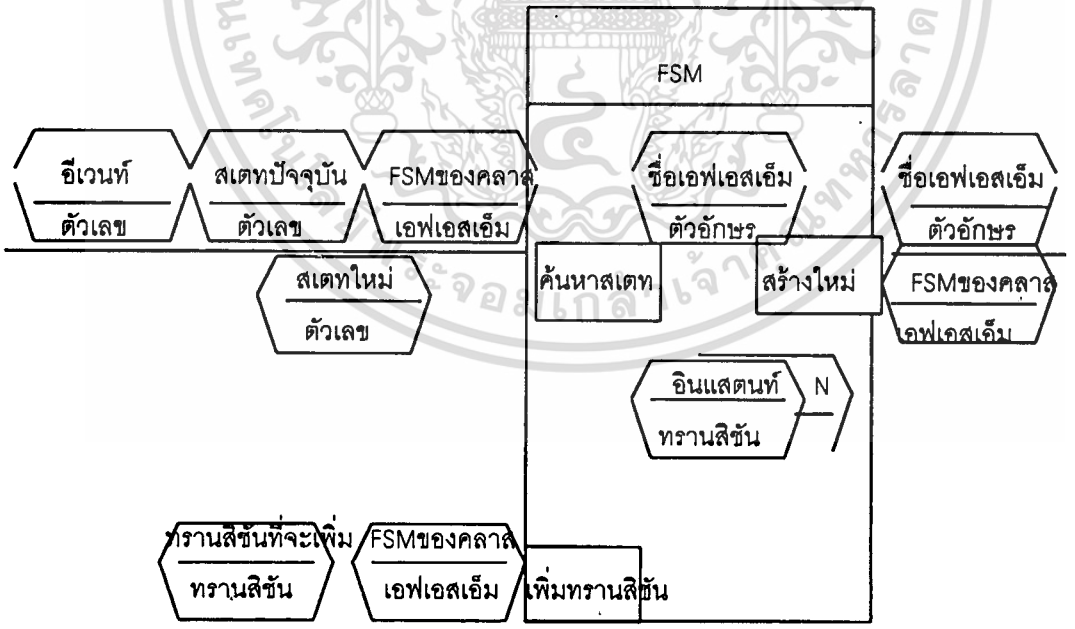
สร้างข้อมูลนี้จะสามารถแสดงได้ดังรูปที่ 3.15 และอาจจะทำได้โดยการให้คลาสเอฟเอสเอ็มเป็นผู้สืบทอด (inherited) มาจากลิสต์คลาส (list class)



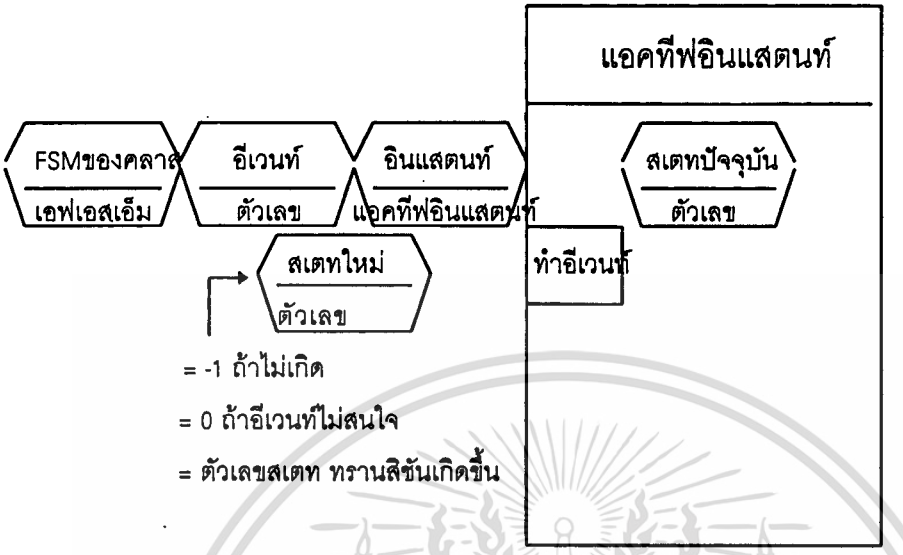
รูปที่ 3.8 แสดงโครงสร้างของส่วนประกอบของโปรแกรม



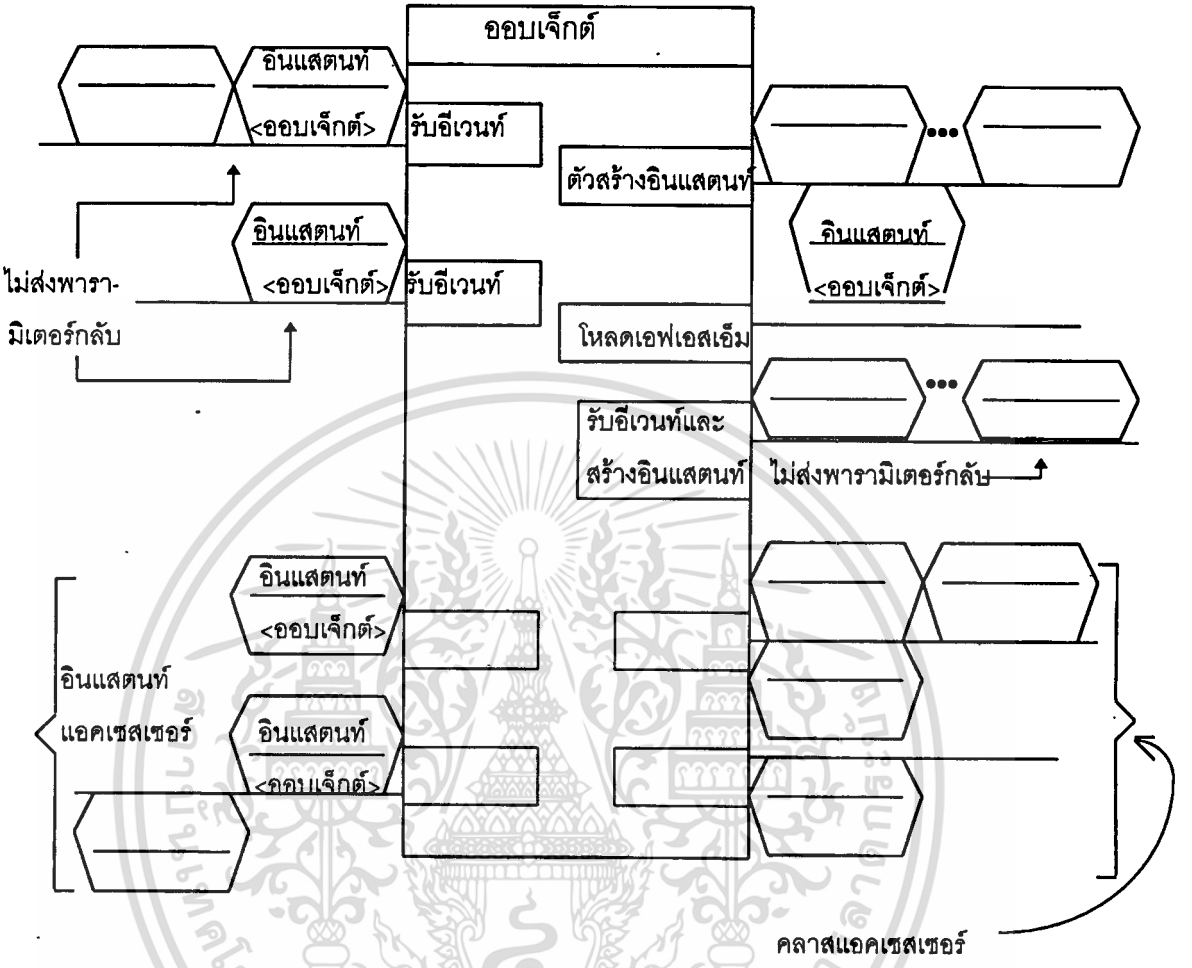
รูปที่ 3.9 แสดงไดอะแกรมของคลาสทรานสิชัน



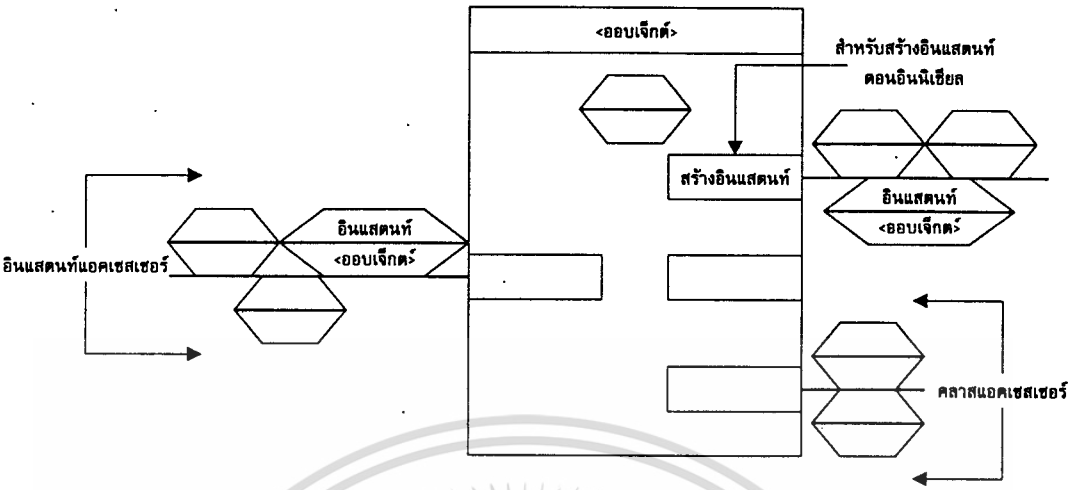
รูปที่ 3.10 แสดงไดอะแกรมของคลาสไฟไนท์สเตทโมเดล (Finite State Model)



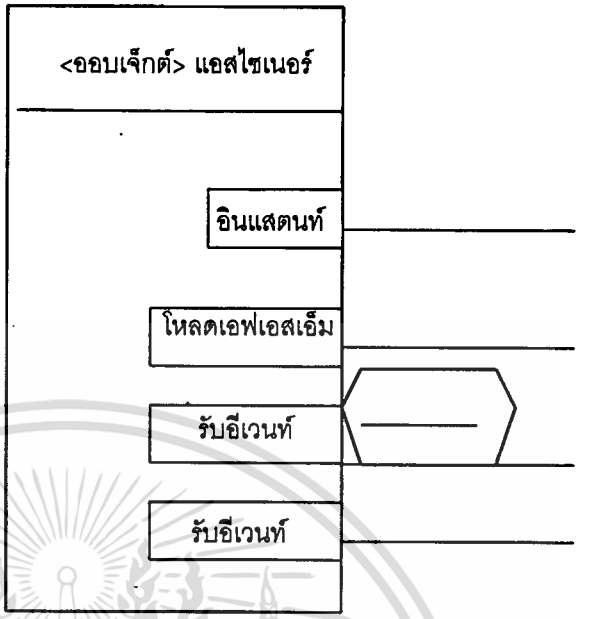
รูปที่ 3.11 แสดงไดอะแกรมของคลาสแอคทีฟอินแสตนท์



รูปที่ 3.12 รูปแบบคลาสไดอะแกรมของแอคทีฟคลาส

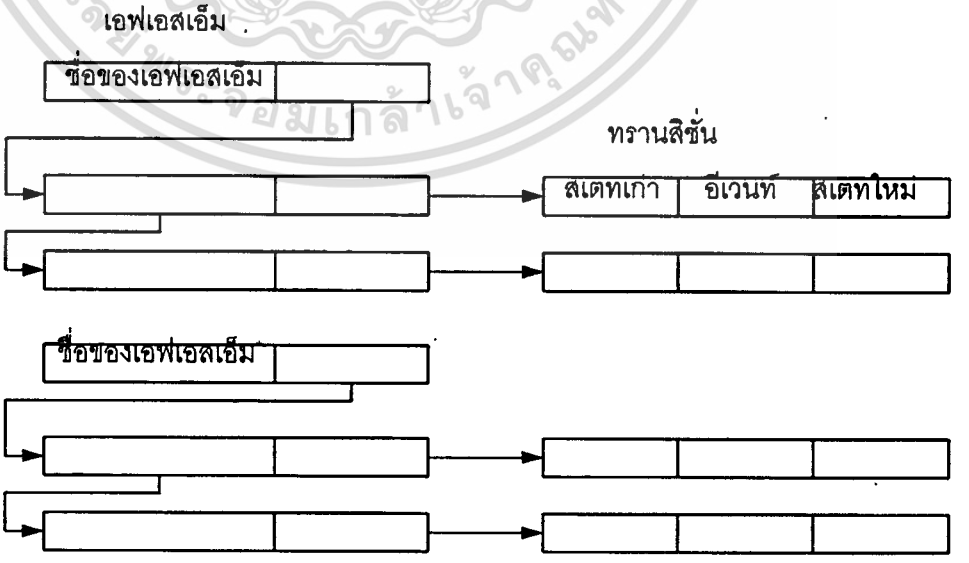


รูปที่ 3.13 แสดงคลาสไดอะแกรมของแพสซีฟคลาส



รูปที่ 3.14 รูปแบบคลาสไดอะแกรมของแอสไซเนอร์คลาส

3.2.3 **แอคทีฟอินแสตนท์** เป็นคลาสนามธรรมซึ่งทุกอินแสตนท์ที่มีสเตทแมชีนจะสืบทอดสเตทปัจจุบัน (current state) จากมันไป



รูปที่ 3.15 โครงสร้างข้อมูลสำหรับเอฟเอสเอ็มและทรานสิชั่นคลาส

ทฤษฎีการทำงาน: การท่องไป(traverse) ในสเตทแมชชีน

เมื่อแอพพลิเคชันคลาสได้รับเหตุการณ์ใดเหตุการณ์หนึ่ง มันจะไปเรียกใช้ โอเปอเรชันทำอีเวนท์ (Do Event) ของแอคทีฟอินแสตนท์ดังรูปที่ 3.16 ในการเรียกใช้นี้แอพพลิเคชันคลาสจะส่งอินพุทพารามิเตอร์คือ หมายเลขเหตุการณ์ที่มันได้รับ, แฮนเดิลของเอฟเอสเอ็มที่จะใช้ในการท่องไป และอินแสตนท์ที่ได้รับเหตุการณ์นั้นดังรูปที่ 3.17 การทำอีเวนท์ (Do Event method) จะไปดึงเอาสเตทปัจจุบันของแอคทีฟอินแสตนท์มา แล้วเรียกใช้เอฟเอสเอ็มโอเปอร์เรชันคือ การค้นหาหรือการท่องไป(Traverse) พร้อมทั้งผ่านอินพุทพารามิเตอร์ คือ สเตทปัจจุบัน, เหตุการณ์ที่ได้รับ และ แฮนเดิลของเอฟเอสเอ็มที่จะใช้ในการท่องไปด้วย

โอเปอร์เรชันการท่องไปของเอฟเอสเอ็ม (FSM.Traverse) ดังรูปที่ 3.18 จะไปเรียกใช้โอเปอร์เรชันแมชของทรานสิชันคลาส (Transition.Match) โดยผ่านพารามิเตอร์ไปให้อีก จนกระทั่งโอเปอร์เรชันแมชของทรานสิชันคลาสรายงานผลว่าพบการเปลี่ยนสเตทที่ต้องการหาแล้วโดยรายงานออกมาเป็นเอาท์พุทพารามิเตอร์ชนิดบูลีนที่มีค่าเป็นจริง และจะส่งสเตทใหม่ที่พบออกมากลับไปให้โอเปอร์เรชันการท่องไปของเอฟเอสเอ็ม ซึ่งจะส่งกลับไปยังโอเปอร์เรชัน ด้วย ในโอเปอร์เรชันการทำอีเวนท์ของแอคทีฟอินแสตนท์จะทำการอัพเดทสเตทปัจจุบันของแอคทีฟอินแสตนท์ พร้อมทั้งส่งสเตทใหม่ไปให้แอพพลิเคชันคลาสที่เรียกใช้มันด้วย

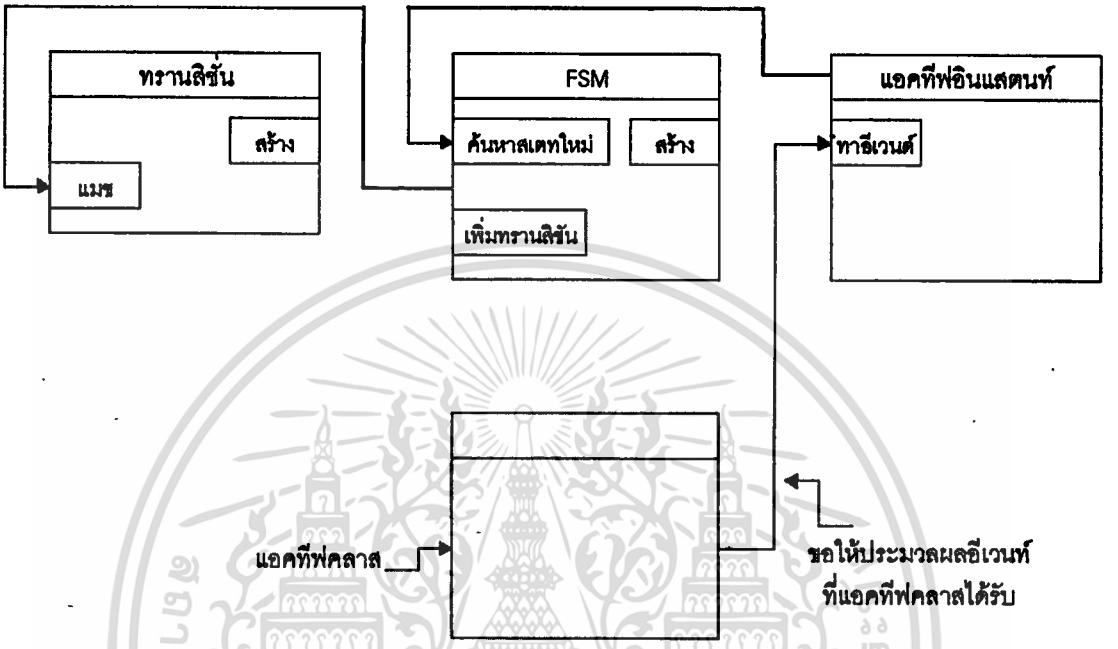
ถ้าโอเปอร์เรชันแมชของทรานสิชันคลาสไม่พบการเปลี่ยนสเตทที่ต้องการ จะทำให้การเปลี่ยนสเตทนั้นตรงกับการไม่สามารถเกิดขึ้น เซลในตารางการเปลี่ยนสเตท(state transition table) คือการแสดงว่าเกิดความผิดพลาดขึ้นในการวิเคราะห์หรือในการอิมพลีเมนต์ (implementation)

ทฤษฎีการทำงาน: การอินนิเชียล (initialization)

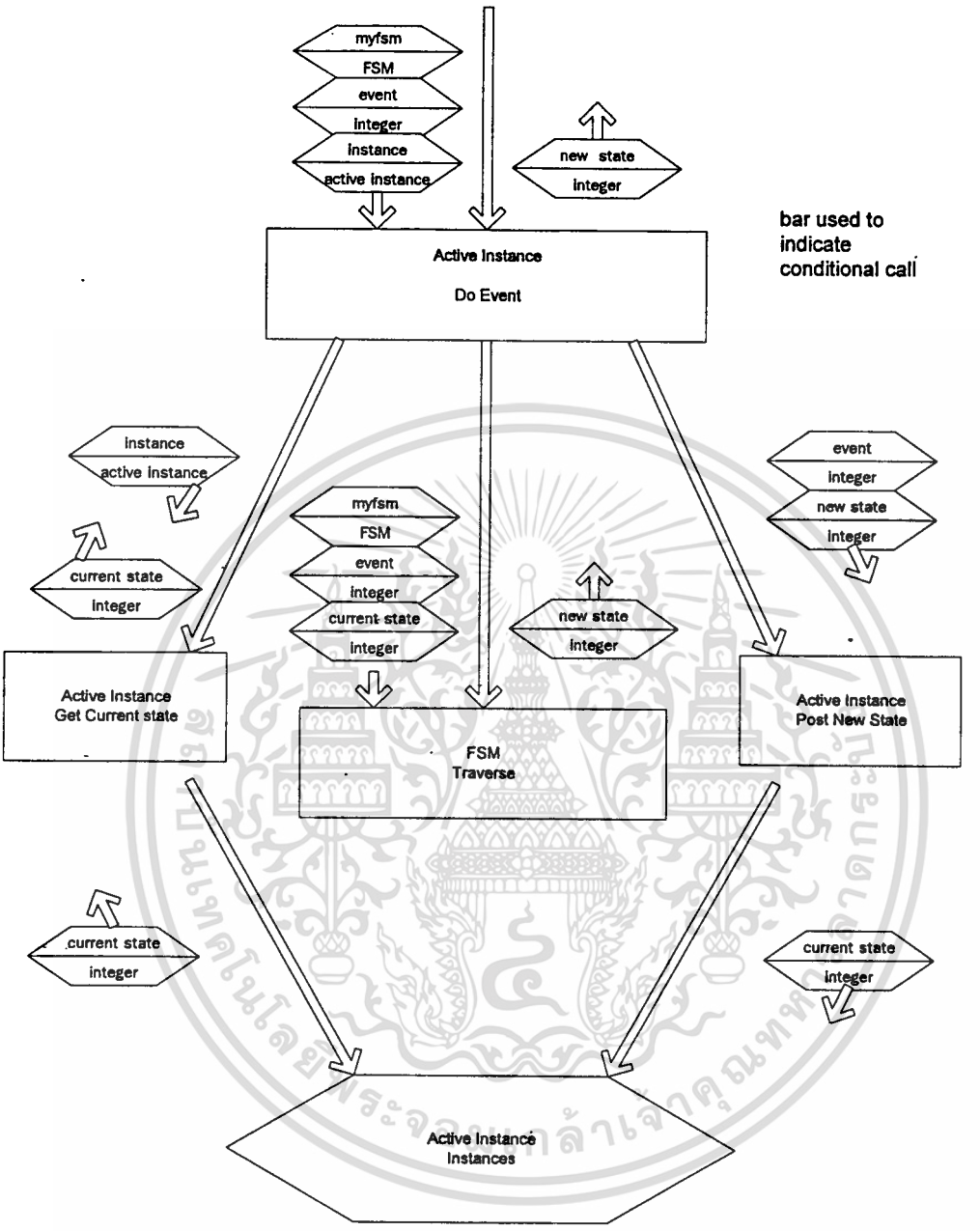
การอินนิเชียลของกลไกของสเตทแมชีนประกอบด้วยการสร้างอินแสตนท์ของเอฟเอสเอ็มและทรานสิชันคลาส ซึ่งจะทำโดยแอพพลิเคชันคลาส โอเปอร์เรชันที่เตรียมไว้ในการอินนิเชียลมี 3 โอเปอร์เรชันได้แก่ โอเปอร์เรชันการสร้างใหม่, การเพิ่มทรานสิชันของเอฟเอสเอ็มและโอเปอร์เรชันการสร้างใหม่ของทรานสิชัน

การอินนิเชียลเริ่มเมื่อแอพพลิเคชันคลาสเรียกใช้โอเปอร์เรชันสร้างใหม่ของเอฟเอสเอ็มดังรูปที่ 3.19 โอเปอร์เรชันการสร้างใหม่จะรับชื่อของไฟไนท์สเตทโมเดลแล้วส่งแฮนเดิลของอินแสตนท์ของเอฟเอสเอ็ม ที่เพิ่งสร้างขึ้นมาใหม่กลับไปให้แอพพลิเคชันคลาส

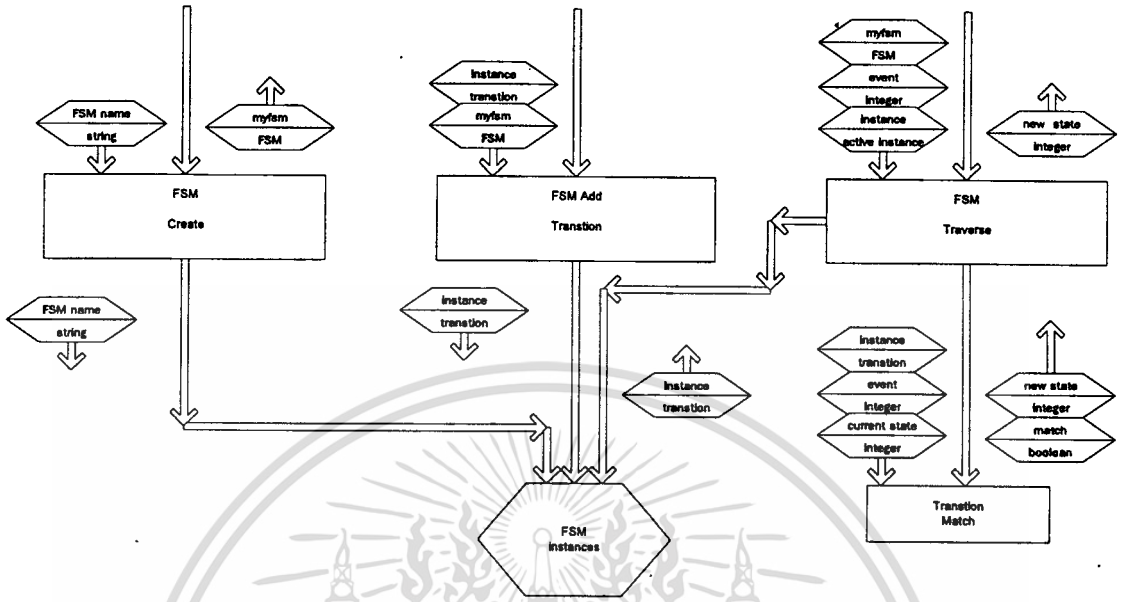
หลังจากแอพพลิเคชันคลาสเรียกใช้การสร้างใหม่ของเอฟเอสเอ็มแล้วก็จะเรียกใช้โอเปอร์เรชันการสร้างใหม่ของทรานสิชัน โดยโอเปอร์เรชันการสร้างใหม่จะสร้างการเปลี่ยนสเตทของแต่ละเซลในตารางการเปลี่ยนสเตท (state transition table) และ จะเรียกโอเปอร์เรชัน การเพิ่มทรานสิชัน ของเอฟเอสเอ็มเพื่อเชื่อม (link) ทรานสิชันต่างๆเข้ากับโครงสร้างข้อมูลของเอฟเอสเอ็ม แล้วการทำงานจึงถูกส่งกลับไปยังแอพพลิเคชันคลาส



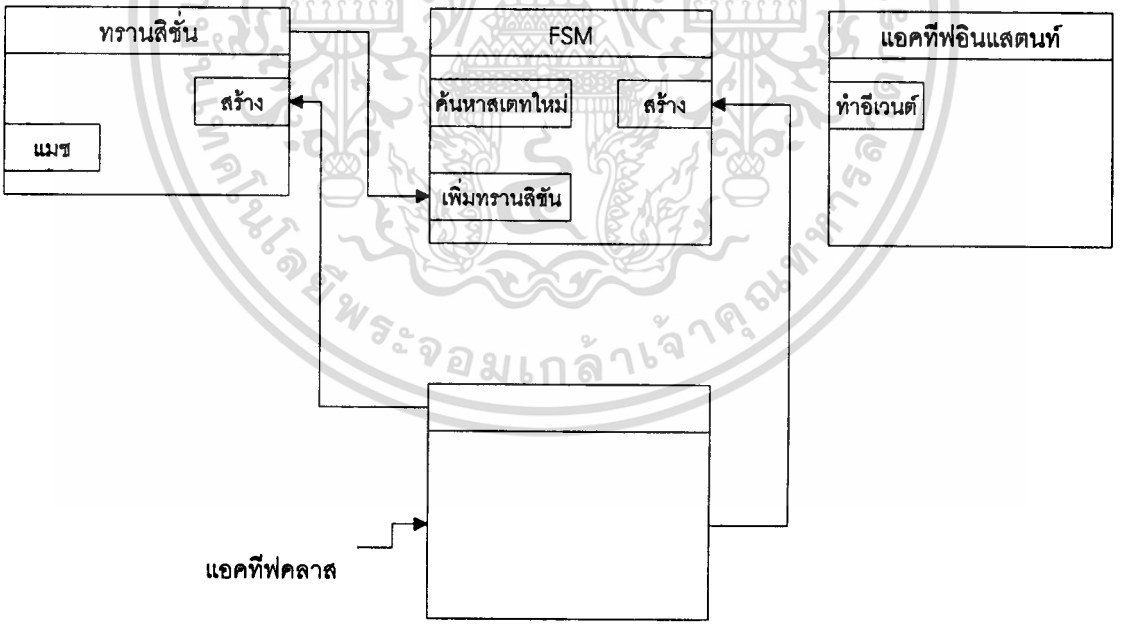
รูปที่ 3.16 แสดงกลไกการประมวลผลอีเวนท์



รูปที่ 3.17 แสดงผังโครงสร้างของคลาสแอคทีฟอินแสตนท์



รูปที่ 3.18 แสดงผังโครงสร้างของเอฟเอสเอ็มคลาส



รูปที่ 3.19 แสดงกลไกของสเตทแมชชีนในการทำอินนิเชียล

3.2.4 แผนภาพคลาส (Class Diagram) สำหรับแอพพลิเคชันคลาส

ประเภทของแอพพลิเคชันคลาส

3.2.4.1 พาสซีฟคลาส (passive class) คือคลาสสำหรับออบเจ็กต์ที่ไม่มีแบบจำลองสถานะ (state model or lifecycle)

3.2.4.2 แอคทีฟคลาส (active class) คือคลาสสำหรับออบเจ็กต์ที่มีสเตทแมชชีนของแต่ละอินแสตนท์ หรือ ออบเจ็กต์ที่มีแบบจำลองสถานะ

3.2.4.3 แอสซายเนอร์ (assigner) คือ คลาสที่ตรงกันกับแบบจำลองสถานะของแอสซายน์เนอร์ ในโอโอเอ (OOA)

3.2.4.1 พาสซีฟคลาส (passive class)

แผนภาพคลาสของพาสซีฟคลาสประกอบด้วย

- **ชื่อของคลาส** (class name) ใช้ชื่อเดียวกับออบเจ็กต์
- **องค์ประกอบของอินแสตนท์** (instance components)
- **แอคเซสเซอร์** (accessors) ดูที่แบบจำลองการติดต่อระหว่างออบเจ็กต์ (object communication model) หรือ ตารางโพรเซสของทุกสเตท (state process table) เพื่อหาแอคเซสเซอร์ที่จะกำหนดให้ออบเจ็กต์ที่กำลังพิจารณาอยู่ และสร้างโอเปอร์เรชันในแผนภาพคลาสสำหรับแต่ละแอคเซสเซอร์นั้นๆ

  โดยสร้างเป็นอินแสตนท์โอเปอร์เรชัน (instance operation) สำหรับแอคเซสเซอร์ที่แอคเซสข้อมูลของอินแสตนท์หนึ่งๆของออบเจ็กต์ หรือมิฉะนั้นก็สร้างเป็นคลาสโอเปอร์เรชัน

  การกำหนดอินพุทและเอาท์พุทพารามิเตอร์ของโอเปอร์เรชันจะตรงกันกับอินพุทและเอาท์พุทของแอคเซสเซอร์นั้นในเอดีเอฟดี (ADFD)
- **ตัวสร้างอินแสตนท์** (a constuctor for pre-existing instances) ในการทำโอโอเอ (OOA) มักสมมติว่าอินแสตนท์ของแต่ละออบเจ็กต์มีอยู่แล้วไม่ต้องมีการสร้างอินแสตนท์ขึ้น

3.2.4.2 แอคทีฟคลาส (active class)

แผนภาพคลาสของแอคทีฟคลาสประกอบด้วย

- **ชื่อของคลาส**

  ใช้ชื่อเดียวกับชื่อออบเจ็กต์
- **องค์ประกอบของอินแสตนท์** การกำหนดองค์ประกอบของอินแสตนท์สำหรับแอคทีฟคลาสเหมือนกับการกำหนดองค์ประกอบของอินแสตนท์ของพาสซีฟคลาส แต่จะไม่มีแอททริบิวท์ที่บ่งบอกถึงสเตทปัจจุบัน(current state attribute) เป็นองค์ประกอบ (component) ด้วย เพราะอินแสตนท์ของแอคทีฟคลาสจะได้รับองค์ประกอบที่บ่งบอกถึงสเตทปัจจุบัน (current state component) สืบทอดมาจากแอคทีฟอินแสตนท์คลาส

- **แอคเซสเซอร์**
- **ตัวรับเหตุการณ์** (event taker) กำหนดโอเปอเรชันที่ตรงกับตัวสร้างเหตุการณ์ (event generator) แต่ละตัวที่แสดงในตารางโพรเซสของทุกสเตท (state process table) เพราะได้รับการกำหนดให้กับออบเจ็กต์ที่อยู่ภายใต้การพิจารณาแล้ว โอเปอเรชันเหล่านั้นเรียกว่า ตัวรับเหตุการณ์ (event taker)

  ◆ ถ้าเหตุการณ์ที่สร้างขึ้นโดยตัวสร้างเหตุการณ์ (Event generator) ไม่ได้ทำให้เกิดการสร้างอินแสตนท์ใหม่ของออบเจ็กต์นั้น ให้กำหนดตัวรับเหตุการณ์ที่ตรงกันนั้นเป็นอินแสตนท์โอเปอเรชัน และตั้งชื่อว่า Take Event <event label>

  ◆ ถ้าเหตุการณ์ที่สร้างขึ้นโดยตัวสร้างเหตุการณ์ ทำให้เกิดการสร้างอินแสตนท์ใหม่ของออบเจ็กต์ขึ้น ให้กำหนดตัวรับเหตุการณ์ที่ตรงกันนั้นเป็นคลาสโอเปอเรชัน และตั้งชื่อว่า Take and Create <event label>

  กำหนดอินพุทพารามิเตอร์ให้ตรงกับข้อมูลของเหตุการณ์ (event data) แต่ละข้อมูลตัวรับเหตุการณ์จะไม่มีเอาท์พุทพารามิเตอร์ เพราะ เหตุการณ์ใน OOA เป็นการติดต่อสื่อสารกันแบบอสมวาร (asynchronous communication) จึงไม่สามารถสร้างสมวาร (synchronous output) เอาท์พุทได้

  จุดประสงค์ของตัวรับเหตุการณ์คือ รับเหตุการณ์ และตัดสินใจว่าจะทำแอคชันอะไรต่อ แล้วก็จัดการทำแอคชันนั้น ซึ่งจะทำให้อินแสตนท์เกิดการเปลี่ยนสเตท
- **ตัวอินนิเชียล** (initializer) แต่ละแอคทีฟคลาสต้องกำหนดคลาสโอเปอเรชันที่ชื่อ Load FSM โอเปอเรชันนี้จะถูกเรียกใช้โดยเมนโมดูลตอนโปรแกรมทำอินนิเชียล (initiation time) เพื่อทำให้แอคทีฟคลาสสร้าง FSM ของมันขึ้นมาทำให้มันสามารถรับเหตุการณ์ต่างๆได้

3.2.4.3 **แอสซายเนอร์คลาส** (Assigner Class)

แผนภาพคลาสของแอสซายน์เนอร์คลาสประกอบด้วย

- **ชื่อของคลาส** คือชื่อเดียวกับแบบจำลองสถานะของมัน
- **ตัวรับเหตุการณ์** เหตุการณ์ที่แอสซายเนอร์ได้รับไม่ใช่เหตุการณ์ที่ส่งให้อินแสตนท์ของมันโดยตรง จึงไม่ต้องมีแฮนเดิลเป็นอินพุทให้กับตัวรับเหตุการณ์ เพราะฉะนั้นตัวรับเหตุการณ์เหล่านั้นจึงเป็นคลาสโอเปอเรชัน
- **ตัวอินนิเชียล**
- **ตัวสร้างอินแสตนท์** จะสร้างอินแสตนท์หนึ่งของแอสซายน์เนอร์คลาส ข้อสังเกต คือโอเปอเรชันนี้ก็ไม่ต้องการอินพุทพารามิเตอร์

แอสซายเนอร์คลาสจะไม่มีองค์ประกอบจะไม่มีองค์ประกอบของอินแสตนท์ (instance component) และไม่มีแอคเซสเซอร์

3.2.5 ผังโครงสร้างของแอพพลิเคชันคลาส (Class Structure Charts for the Application Classes)

การออกแบบภายในของแอพลิเคชันคลาสจะแสดงโดยผังโครงสร้างของคลาส ผังโดรงสร้างของคลาสนั้นจะมีขนาดใหญ่มากจึงแสดงเป็นรูปย่อยๆที่มีความเกี่ยวเนื่องกันอยู่

3.2.5.1 ผังโครงสร้างของคลาสสำหรับพาสซีฟคลาส

เหมือนกับการเขียนแผนภาพของคลาส (class diagram)โดยมีข้อสังเกต คือ อินพุทและเอาท์พุท ของแต่ละโอเปอร์เรชัน สามารถก็อปปี้ได้จากแผนภาพของคลาส (class diagram)

3.2.5.2 ผังโครงสร้างของคลาสสำหรับแอคทีฟคลาส

ผังโครงสร้างของคลาสสำหรับแอคทีฟคลาสจะซับซ้อนกว่าผังโครงสร้างของคลาสสำหรับพาสซีฟคลาส โดยจะแบ่งการอธิบายออกเป็น 4 ส่วน ดังนี้

3.2.5.2.1 ตัวอินนิเชียล

มีโอเปอร์เรชัน 2 ตัวที่เตรียมไว้เพื่อทำการอินนิเชียลแอคทีฟคลาสได้แก่ Institute และการโหลดเอฟเอสเอ็ม ส่วนหนึ่งของผังโครงสร้างของคลาสที่แสดงโอเปอร์เรชันอินนิเชียล (initialization) อยู่ในรูปที่ 3.20 ประโยชน์ของข้อมูลของคลาส (class data) เช่นเอฟเอสเอ็มของคลาส คือ ตัวแปรที่ใช้เป็นแฮนเดิลของ เอฟเอสเอ็มของคลาสนั้น เพราะมันใช้เป็นข้อมูลของทุกอินแสตนท์ของคลาส

3.2.5.2.2 แอคเซสเซอร์

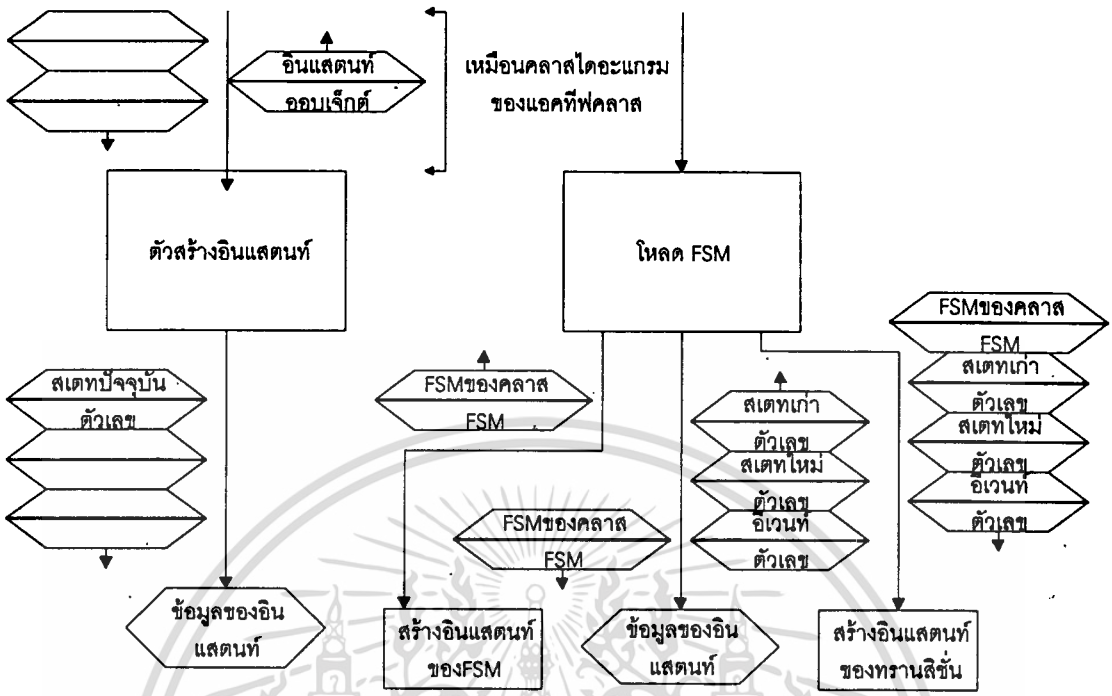
ส่วนหนึ่งของผังโครงสร้างของคลาสที่แสดงแอคเซสเซอร์จะเขียนได้เช่นเดียวกับแผนภาพของคลาส (class diagram)

3.2.5.2.3 ตัวรับเหตุการณ์

ส่วนหนึ่งของผังโครงสร้างของคลาสสำหรับโอเปอร์เรชันรับอีเวนท์ โอเปอร์เรชันนี้จะหาแฮนเดิลของเอฟเอสเอ็มซึ่งเป็นของคลาสนั้น และเรียกใช้ โอเปอร์เรชัน การทำอีเวนท์ของแอคทีฟอินแสตนท์ เพื่อหาสเตทใหม่ที่จะได้จากการรับเหตุการณ์ที่เข้ามา แล้วจึงทำแอคชันที่เกี่ยวข้องกับสเตทใหม่นั้น ขอให้สังเกตว่า ถ้าหมายเลขของสเตทใหม่เป็น 0 จะไม่มีการทำแอคชันใดๆ และถ้าหมายเลขของสเตทใหม่น้อยกว่า 0 จะหมายถึง เหตุการณ์ที่ไม่สามารถเกิดขึ้นได้

โอเปอร์เรชันทำอีเวนท์และสร้างใหม่ทำให้เกิดการสร้างอินแสตนท์ใหม่โดยใช้ข้อมูลของเหตุการณ์ที่ได้รับ เนื่องจากการสร้างอินแสตนท์ใหม่นี้จะทำในแอคชันของสเตทเริ่มต้น (creation state) โอเปอร์เรชันทำอีเวนท์และสร้างใหม่จึงต้องส่งผ่านการควบคุมไปยังแอคชันของสเตทเริ่มต้น

ส่วนหนึ่งของผังโครงสร้างของคลาสสำหรับโอเปอร์เรชัน Take and Create แสดงในรูปที่ 3.21



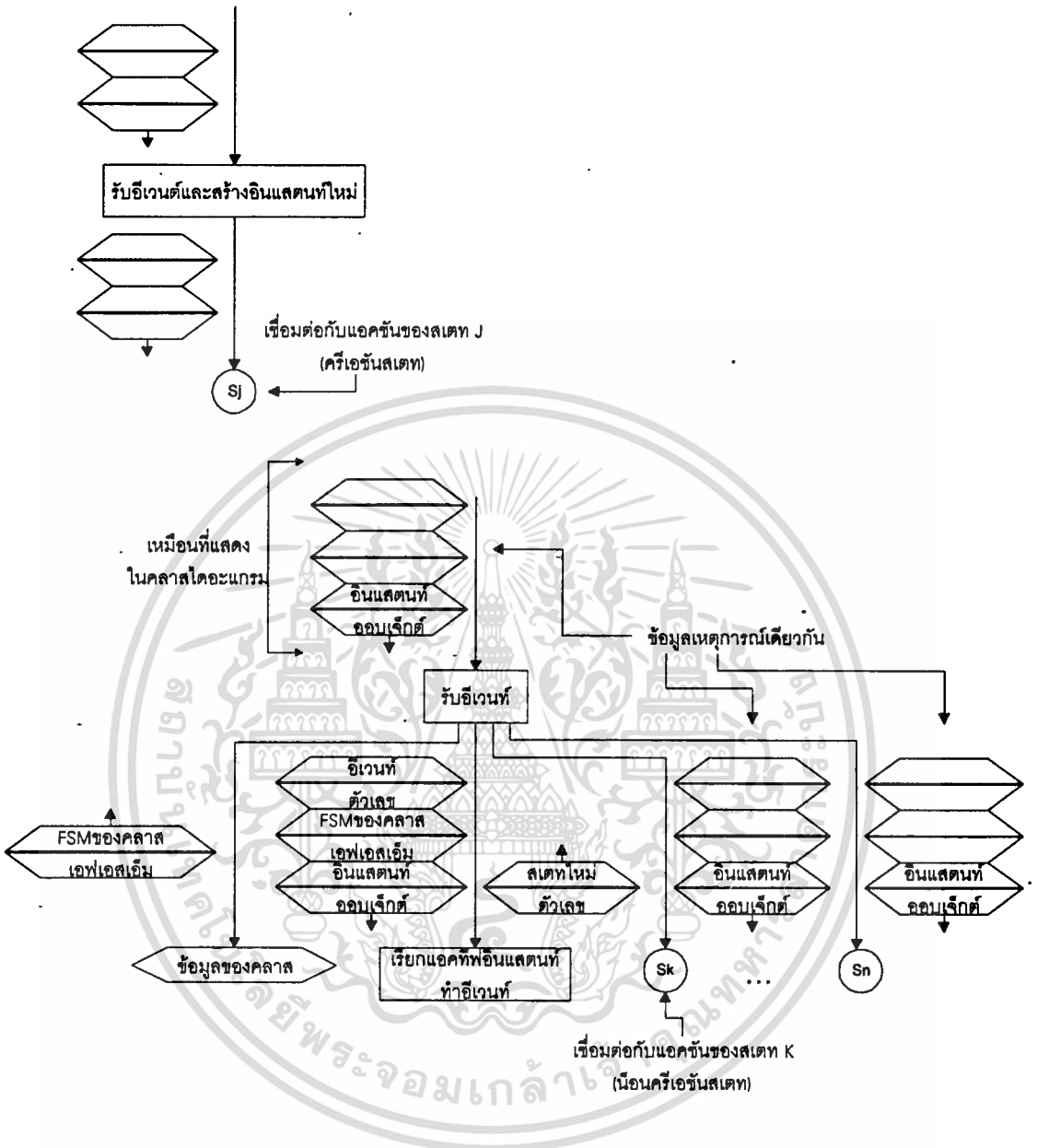
รูปที่ 3.20 การทำอินนิเชียลของแอคทีฟคลาส

3.2.5.2.4 แอคชัน

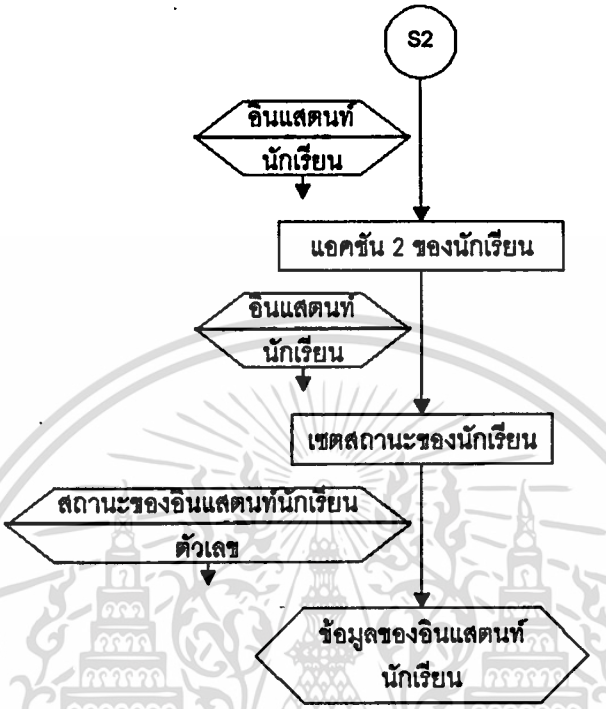
ส่วนหนึ่งของผังโครงสร้างของคลาสที่เกี่ยวกับแอคชัน ถูกสร้างเหมือนกับเอดีแอฟดี เริ่มต้นด้วยการสร้างโอเปอร์เรชันที่ไม่ได้กำหนดในคลาส (unpublished operation) ซึ่งชื่อว่า <object>.Action <k> ซึ่ง k คือ หมายเลขสเตทที่สัมพันธ์กับแอคชันนั้น โอเปอร์เรชันนี้จะถูกเรียกใช้ได้จากตัวรับเหตุการณ์ในคลาสเดียวกันเท่านั้น เพราะฉะนั้นจึงเป็นโอเปอร์เรชันที่ไม่ได้ถูกกำหนดไว้ในคลาส

โอเปอร์เรชัน Action <k> คือโมดูลรับข้อมูลของเหตุการณ์ทุกๆอันที่เตรียมไว้ให้เป็นข้อมูลอินพุทแก่ตัวรับเหตุการณ์ โมดูลนี้จะไปเรียกใช้โมดูลที่สัมพันธ์กับแอคเซสเซอร์ (accessors), การทดสอบ (tests), (transformations) และ ตัวสร้างเหตุการณ์ (event generators) ใน เอดีเอฟดี

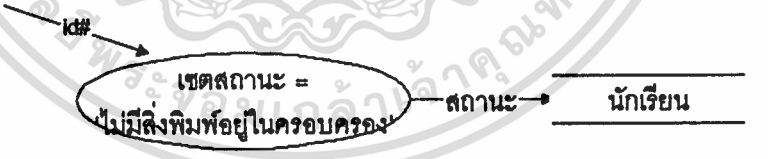
จุดประสงค์ของโมดูล Action <k> คือ เพื่อเรียกใช้โมดูลต่างๆ เหล่านี้ตามลำดับที่ต้องการ และเพื่อรับเอาท์พุทของโมดูลหนึ่งไปใช้เป็นอินพุทของโมดูลถัดไป เรียกได้ว่าโมดูล Action <k> ทำตัวเป็นเสมือนไวร์ริ่ง (wiring) บน เอดีเอฟดี เห็นได้จากการเปรียบเทียบรูปที่ 3.22 และ เอดีเอฟดีที่ตรงกัน ในรูปที่ 3.23



รูปที่ 3.21 แสดงโอเปอเรชันรับอีเวนท์และสร้างอินแสตนท์และโอเปอเรชันรับอีเวนท์ของแอคทีฟคลาส



รูปที่ 3.22 ส่วนหนึ่งของผังโครงสร้างของคลาสนักเรียนเปรียบเทียบADFD ที่ตรงกันในรูปที่ 3.23



รูปที่ 3.23 แสดง ADFD ของสเตทที่ 2 ของออบเจ็กต์นักเรียน

การสร้างโมดูลซึ่งสัมพันธ์กับโพรเซสในเอดีเอฟดี

- ถ้าโพรเซส คือ แอคเซสเซอร์ที่ถูกกำหนดให้กับคลาสอื่น ให้วาดโอเปอร์เรชันนั้นเป็นรูปห้าเหลี่ยม
- ถ้าโพรเซส คือ แอคเซสเซอร์ที่ถูกกำหนดให้กับคลาสนี้ ให้วาดเส้นการเรียกใช้จากโมดูล Action <k> ไปยังโมดูลแอคเซสเซอร์ หรือสามารถทำก็อปปี้ของโมดูลแอคเซสเซอร์นี้ไว้ในส่วนที่แสดงแอคชันของผังโครงสร้างของคลาส (class structure chart) และอธิบายด้วยว่าโอเปอร์เรชันนี้มีการแสดงไว้ในที่อื่นอีก

- ถ้าโพรเซส คือ การทดสอบ (test) หรือ (transformation) โมดูลที่สัมพันธ์กันคือ โอเปอร์เรชันที่ไม่ได้ถูกกำหนดในคลาสนั้น (unpublished operation) กำหนดอินพุท และ เอาท์พุทพารามิเตอร์ของโมดูลให้สัมพันธ์กับข้อมูลอินพุทและเอาท์พุท ที่แสดงในเอดีเอฟดี
- ถ้าโพรเซส คือ การทดสอบ (test) ให้กำหนดเอาท์พุทพารามิเตอร์ซึ่งบรรจุข้อมูลที่บอกผลลัพธ์จากการเทสต์ด้วย นี่เป็นการเปลี่ยนรูปแบบของคอนโทรลเอาท์พุท (control output) ของเอดีเอฟดี เป็นข้อมูลแทน โมดูล Action <k> จะใช้เอาท์พุทพารามิเตอร์นี้เพื่อตัดสินใจว่าจะเรียกใช้โมดูลใดต่อไป
- ถ้าโพรเซส คือ ตัวสร้างเหตุการณ์ ให้แสดงการเรียกใช้ตัวรับเหตุการณ์ที่สัมพันธ์กันไม่ว่าจะอยู่ในคลาสนี้หรือคลาสอื่นๆ

3.2.6 ผังโครงสร้างของคลาสสำหรับแอสซายเนอร์คลาส

เหมือนกับของแอคทีฟคลาสมาก โดยประกอบด้วย 3 ส่วน ได้แก่

3.2.6.1 ตัวอินนิเชียล

แสดงในรูปที่ 3.24 สังเกตว่าแฮนเดิลสำหรับอินแสตนท์ของคลาสจะเก็บไว้เป็นข้อมูลของคลาส (class data)

3.2.6.2 ตัวรับเหตุการณ์

แสดงในรูปที่ 3.25

3.2.6.3 แอคชัน ถูกสร้างขึ้นสอดคล้องกับกฎของแอคชันสำหรับแอคทีฟคลาส

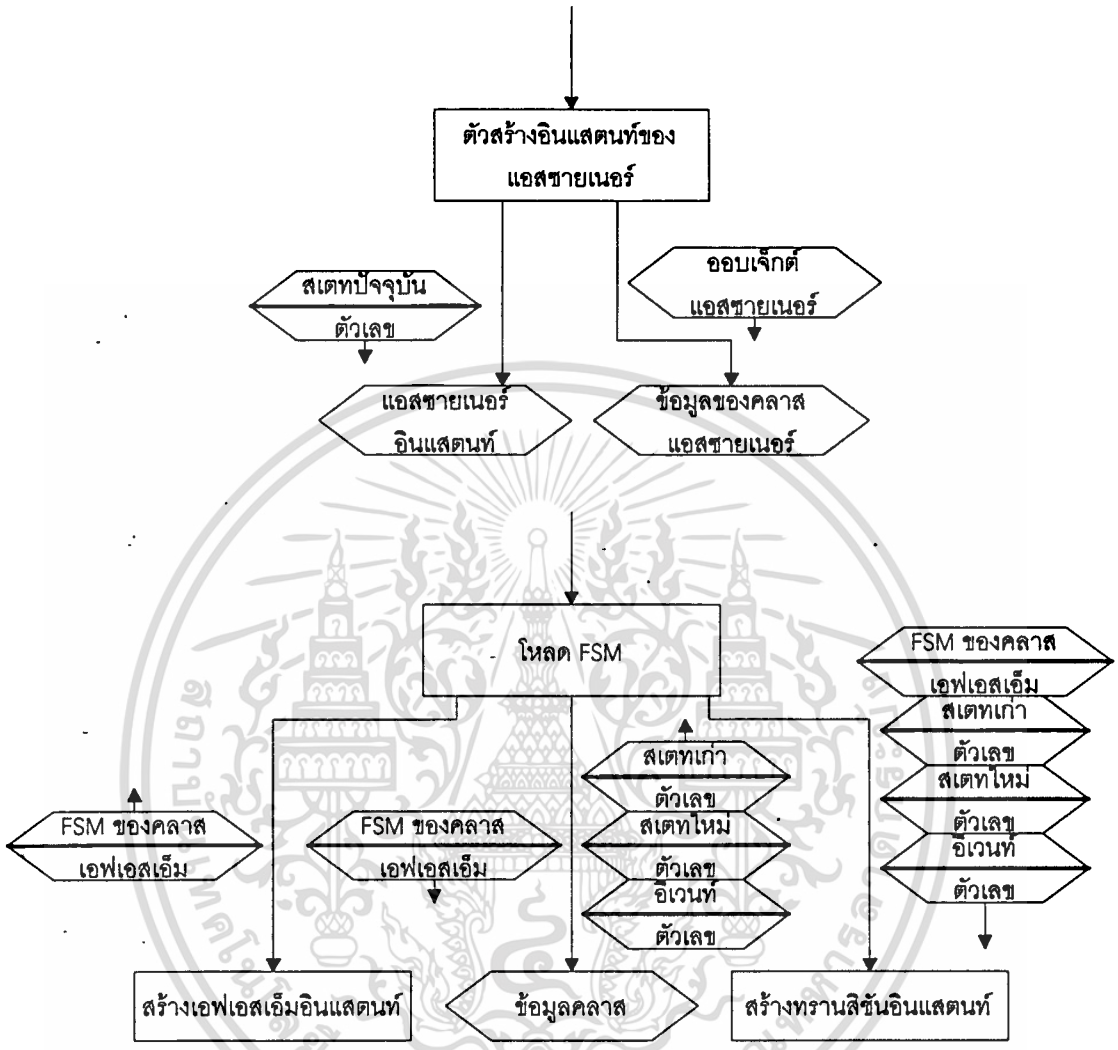
3.2.7 เมนโปรแกรม (Main Program)

มีหน้าที่ 3 ประการคือ

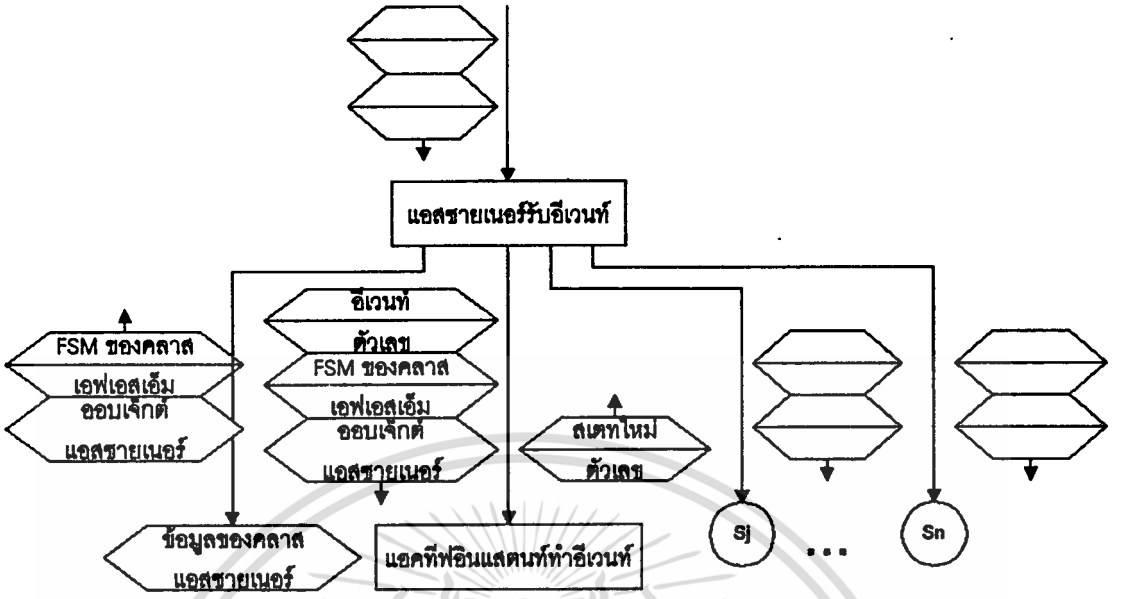
- เรียกใช้โอเปอร์เรชันอินนิเชียลของคลาสต่างๆ
- สร้างเหตุการณ์ภายนอก (external event) ที่เป็นตัวเริ่มต้นหรือ เชื่อมต่อการควบคุม (initiate or continue a thread of control)
- สร้างเหตุการณ์ของนาฬิกา (timer event)

3.2.8 ไดนามิคออฟรีเลชันชิพ (Dynamics of Relationships)

ความสัมพันธ์ (associations) ระหว่างสิ่งต่างๆในโลก (real world)ซึ่งวิวัฒนาการไปตามกาลเวลาด้วยกฎ (rules) ต่างๆ และนโยบาย (policies) ต่างๆที่เกี่ยวข้อง ณ เวลาต่างๆ ในช่วงระยะเวลาของความสัมพันธ์นั้น



รูปที่ 3.24 แสดงการทำอินนิเชียลของแอสซายเนอร์คลาส



รูปที่ 3.25 แสดงการทำโอเปอเรชันรับอีเวนท์ของแอสซายเนอร์คลาส

เมื่อเรากำหนดความสัมพันธ์ในโลกแห่งความจริง (real-world association) เป็นรีเลชันชิพ (relationship) ในอินฟอเมชันโมเดล (information model) เราจะทราบถึงรูปลักษณะซึ่งคงที่ (static aspect) ของความสัมพันธ์นั้น สิ่งที่เราไม่สามารถทราบได้ คือการเคลื่อนไหวของรีเลชันชิพ (dynamics of the relationship) และทราบสิ่งต่างๆอีกดังนี้

- รีเลชันชิพวิวัฒนาการไปตามกาลเวลาอย่างไร
- อินแสตนท์ของรีเลชันชิพ ถูกสร้าง หรือทำลายภายใต้สภาพแวดล้อมอย่างไร
- อินแสตนท์ของออบเจ็กต์ถูกเลือกให้เข้าร่วมในรีเลชันชิพ อย่างไร

คำถามเหล่านี้หาคำตอบได้ในสเตทโมเดล (state models)

ไลฟ์ไซเคิลของรีเลชันชิพ (Lifecycles of Relationships)

ความสัมพันธ์ระหว่างสิ่งต่างๆในโลกดำเนินไปตามสเตทต่างๆโดยใช้กฎและนโยบายต่างๆ เพราะฉะนั้นความสัมพันธ์ จึงถูกสรุปความหมายเป็นรีเลชันชิพ ในรูปแบบของแอสโซซิเอทีฟออบเจ็กต์ (associative object) กับแอททริบิวท์ที่แสดงสเตทปัจจุบัน (current state attribute) และสเตทโมเดล (state model) จึงถูกสร้างขึ้นเพื่อแสดงไลฟ์ไซเคิลของรีเลชันชิพ

สเตทโมเดลอธิบายถึงพฤติกรรมของออบเจ็กต์ของรีเลชันชิพ

เช่น ผู้แทนจำหน่ายขายอุปกรณ์ทางอิเล็กทรอนิกส์หลายประเภท ในแผนกการขายทำการประมวลผลใบสั่งซื้อ (purchase orders) ที่ได้รับจากลูกค้า เพื่อจำแนกใบสั่งซื้อแต่ละใบเป็นรายการสั่งซื้อสินค้าแต่ละชนิด ในอินฟอร์เมชันโมเดลดังรูปที่ 3.26 รีเลชันชิพ Customer ได้สั่งซื้อสินค้า ซึ่งคือสินค้าในแอสโซซิเอทีฟออบเจ็กต์ (associative object) ในการทำงานของผู้แทนจำหน่าย สินค้าแต่ละอันจะถูกโพรเซสแยกจากกัน อุปกรณ์ที่ถูกสั่งซื้อจะได้รับมาจากสต็อกและถูกนำไปแพ็ค (packed) ส่งและแจ้งราคาสินค้าแก่ผู้ซื้อ วิธีการนี้สามารถทำได้เพราะสินค้าแต่ละอันสามารถหามาได้และส่งได้ทันที ถึงแม้ว่าในการสั่งซื้อสินค้าจะเป็นความสัมพันธ์ (relationship) แต่สเตทโมเดลของมันก็จะสร้างเหมือนกับสเตทโมเดลของออบเจ็กต์อื่น ๆ

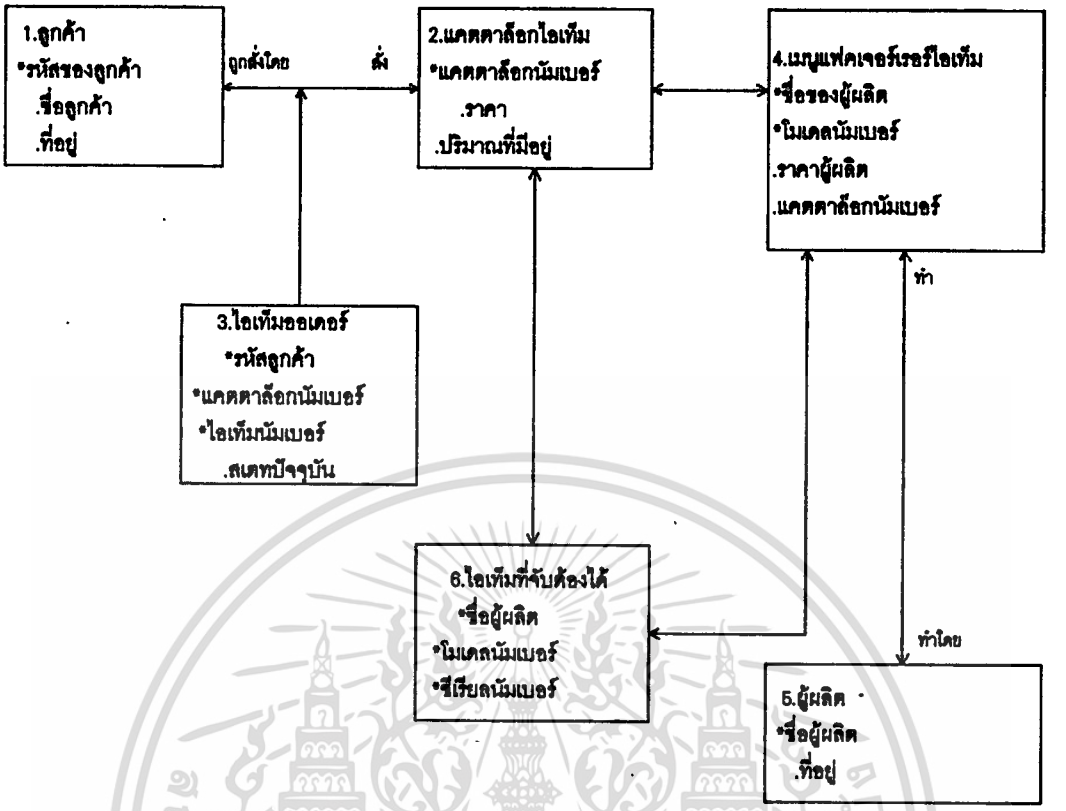
ความสัมพันธ์แบบไดนามิกที่ไม่มีไลฟ์ไซเคิล (Dynamic Relationships without Lifecycle)

ถึงแม้ว่าความสัมพันธ์จะไม่มีไลฟ์ไซเคิลแ ต่เป็นไปได้ว่ามันอาจจะมีคุณสมบัติแบบไดนามิก ถ้าอินแสตนท์ของความสัมพันธ์นั้นถูกสร้างและลบในช่วงเวลาของการวิเคราะห์ เช่น ในกิจการธนาคาร ซึ่งมีอินฟอร์เมชันโมเดลดังรูป 3.27 ความสัมพันธ์ระหว่างตัวลูกค้าและเลขบัญชีของเขาจะคงที่และไม่มีสเตทใด ๆ แต่อย่างไรก็ตามเมื่อมีการสร้างเลขบัญชีของลูกค้าใหม่หรือตัวลูกค้าเอง อินแสตนท์ของความสัมพันธ์ CUSTOMER OWNS ACCOUNT จะถูกสร้างหรือลบไปด้วย

เนื่องจากความสัมพันธ์นี้มีคุณสมบัติแบบไดนามิกอย่างเดียว ซึ่งก็คือการสร้างและลบของอินแสตนท์ของความสัมพันธ์ จึงไม่จำเป็นจะต้องสร้างสเตทโมเดลของความสัมพันธ์นี้ แต่จะให้สเตทโมเดลของออบเจ็กต์ที่เกี่ยวข้องเป็นตัวสร้างและลบอินแสตนท์ของความสัมพันธ์นี้แทน

ความสัมพันธ์ที่เกี่ยวข้องกับการแข่งขัน

การแข่งขัน คือ กรณีที่มีอินแสตนท์มากกว่า 1 ตัวของออบเจ็กต์ขอใช้บริการจากอีกออบเจ็กต์หนึ่งพร้อม ๆกัน เมื่อเกิดกรณีเช่นนี้จะต้องแก้ไขทำให้เป็นซีเรียลไลซ์(Serialization) โดยสร้างสเตทโมเดลสำหรับความสัมพันธ์ที่เกิดการแข่งขัน เริ่มโดยการแอสโซสิเอทีฟออบเจ็กต์ (Associative Object) จากความสัมพันธ์นั้นในอินฟอร์เมชันโมเดล แล้วสร้างสเตทโมเดลซึ่งมีหน้าที่สร้างอินแสตนท์ของความสัมพันธ์



รูปที่ 3.26 แสดงอินฟอเมชันโมเดลของการดำเนินงานของผู้แทนจำหน่าย

รูปที่ 3.27 อินฟอเมชันโมเดลของธนาคาร

บทที่ 4
เปรียบเทียบวิธีการพัฒนาระบบงานประยุกต์

4.1 เปรียบเทียบวิธีการพัฒนาซอฟต์แวร์ (Comparison of Methodologies) ระหว่างวิธีออบเจ็กต์โอเรียนเต็ดและวิธีแบบฟังก์ชันนอล (Object-Oriented Development V.S. Function-Oriented Methodology)

4.1.1 Object Lifecycles Modeling the World in States เป็นวิธีซอฟต์แวร์เอ็นจิเนียริ่ง

วิธีซอฟต์แวร์เอ็นจิเนียริ่งเป็นโพรเซสในการสร้างซอฟต์แวร์โดยใช้เทคนิคและโนเตชัน (notations) ต่างๆ methodology หมายถึง ซีรีส์ (a series) ของสเต็ปการสร้างซอฟต์แวร์รวมทั้งเทคนิคและโนเตชันที่เกี่ยวข้องกับแต่ละสเต็ป สเต็ปการสร้างซอฟต์แวร์จะแสดงเป็นไลฟ์ไซเคิล (Lifecycle) ซึ่งประกอบด้วยเฟสต่างๆ (phases) ในการพัฒนาซอฟต์แวร์ ไลฟ์ไซเคิลของซอฟต์แวร์เริ่มต้นจาก การกำหนดปัญหา (formulation of problem) การวิเคราะห์ (analysis) การออกแบบ (design) การอิมพลีเมนท์ (implement) การทดสอบซอฟต์แวร์ (testing) และโอเปอเรชันนอลเฟส (operaional phase) ซึ่งคือการบำรุงรักษา (maintainance) และการเพิ่มเติม (enhancement) ความสามารถของซอฟต์แวร์

Object Lifecycles Modeling the World in States Methodology นี้สนับสนุนไลฟ์ไซเคิลของซอฟต์แวร์รวมถึงการทดสอบและบำรุงรักษาก็สามารถทำได้โดยวิธีนี้ วิธีแบบออบเจ็กต์โอเรียนเต็ดจะทำให้การออกแบบง่ายต่อการทดสอบและบำรุงรักษา และเพิ่มเติมมากกว่าวิธีอื่นๆ เพราะออบเจ็กต์คลาสต่างๆ เป็นหน่วยธรรมชาติของหลักการโมดูล่า (a natural unit of modularity)

ผู้พัฒนาบางคนชอบวิธีพัฒนาซอฟต์แวร์โดยสร้างโปรโตไทป์ (Prototype) หลักการคือ การสร้างบางส่วนของซอฟต์แวร์ก่อน แล้วนำมาประเมินค่าการใช้งานโดยผู้ใช้ แล้วจึงปรับปรุงใหม่ให้ตรงความต้องการ ทำเช่นนี้ไปเรื่อยๆจนได้ระบบที่ตรงกับความต้องการของผู้ใช้มากที่สุด โดยผ่านขั้นตอนการสร้างซอฟต์แวร์ซ้ำแล้วซ้ำอีก

ในทางตรงข้าม ก็มีผู้พัฒนาซอฟต์แวร์นิยมใช้วิธีไลฟ์ไซเคิล (Lifecycle) ซึ่งซอฟต์แวร์จะถูกกำหนดคุณสมบัติ (specification) ที่ต้องการครั้งเดียว ออกแบบครั้งทีเดียว และอิมพลีเมนท์ครั้งเดียวทั้งหมดด้วย

Object Lifecycles Modeling the World in States Methodology สนับสนุนการสร้างซอฟต์แวร์ทั้งสองวิธีข้างต้น วิธีการแบบออบเจ็กต์โอเรียนเต็ด มีหลักการอยู่ที่ออบเจ็กต์ในโลกแห่งความจริง (real-world objects) ซึ่งเมื่อถูกนำมาสร้างซอฟต์แวร์ออบเจ็กต์ (software objects) เราสามารถสร้างพฤติกรรมต่างๆเพิ่มเติมกี่รอบก็ได้ ดังนั้นจึงเหมาะทั้งกับการสร้างเป็นโปรโตไทป์ และซอฟต์แวร์ทั้งหมดจริงๆ ใน

ทางตรงข้าม เมื่อใช้วิธีการแบบฟังก์ชันนอลสร้างเป็นโปรโตไทป์ซอฟต์แวร์ขึ้นมาแล้ว จะนำฟังก์ชันแบบโปรโตไทป์เหล่านี้ไปใช้ในการอิมพลีเมนท์ทั้งระบบไม่ได้

4.1.2 STRUCTURED ANALYSIS/STRUCTURED DESIGN (SA/SD)

วิธีซอฟต์แวร์เอ็นจิเนียริ่งที่ใช้กันมากที่สุดในปัจจุบันคือวิธีที่มีพื้นฐานอยู่บนดาต้าโฟล์วไดอะแกรม (data flow diagrams) เราจะพูดกันเรื่อง SA/SD ซึ่งเป็นตัวแทนของวิธีดาต้าโฟล์วไดอะแกรม Yourdon, DeMarco, Constantine, Page-Jones และคนอื่นๆ เขียนเรื่อง SA/SD ส่วน Ward และ Mellor ได้เพิ่มเติมการทำ real-time ให้กับ SA/SD ทำให้ SA/SD เป็นที่แพร่หลาย และนำไปประยุกต์กับปัญหาหลายอย่างได้ และมีเอกสารประกอบให้ศึกษามากมาย

SA/SD ระหว่างเฟสการวิเคราะห์ จะใช้โนเตชันต่างๆอธิบายระบบได้แก่

1. ดาต้าโฟล์วไดอะแกรม

แสดงการเปลี่ยนแปลงของข้อมูล ขณะไหลผ่านระบบและเป็นจุดสำคัญของ SA/SD

ดาต้าโฟล์วไดอะแกรมประกอบด้วย โพรเซส, ดาต้าโฟล์ว, แอคเตอร์ (actors) และดาต้าสตอร์ (data stores) เริ่มจากดาต้าโฟล์วไดอะแกรมระดับสูง (หยาบ) และแบ่งโพรเซสที่ซับซ้อนออกเป็นสับไดอะแกรมย่อยๆไปเรื่อยๆ จนกระทั่งได้โพรเซสที่เล็ก และง่ายต่อการอิมพลีเมนท์ การแตกย่อยโพรเซสก็จะหยุด

2. การกำหนดคุณลักษณะของโพรเซส (process specification)

เมื่อได้โพรเซสที่เล็กพอแล้ว จะมีการเขียนคุณลักษณะของโพรเซสแต่ละอัน โดยอาจอธิบายได้ด้วย ตารางการตัดสินใจ (decision tables), ซูโดโค้ท (pseudocode) หรือเทคนิคอื่นๆ

3. ดาต้าดิกชันนารี (data dictionary)

จะมีรายละเอียดที่ดาต้าโฟล์วไดอะแกรมไม่มี ดาต้าดิกจะกำหนดความหมายของดาต้าโฟล์ว, ดาต้าสตอร์ และชื่อต่างๆในระบบ

4. ไดอะแกรมการเปลี่ยนสเตท (state transition diagram)

แสดงพฤติกรรมที่เปลี่ยนไปตามเวลา ไดอะแกรมการเปลี่ยนสเตทส่วนใหญ่จะอธิบายคอนโทรลโพรเซส หรือเวลาที่ฟังก์ชันถูกเอ็กซีคิวท์ และการแอคเซสข้อมูลเมื่อมีอีเวนท์ (events) มาทริก (trig)

5. เอ็นติตี้รีเลชันชิพไดอะแกรม (entity-relationship (ER) diagram)

เน้นรีเลชันชิพระหว่างดาต้าสตอร์ (ซึ่งอาจถูกแสดงในการกำหนดคุณลักษณะของโพรเซสเท่านั้น)ข้อมูลแต่ละอันใน ER คือดาต้าสตอร์ในดาต้าโฟล์วไดอะแกรม อินฟอเมชันโมเดลใน Object Lifecycles Modeling the World in States Methodology นี้ก็เป็นรูปแบบหนึ่งที่ได้รับการพัฒนาเพิ่มเติมจาก ER

ส่วนในเฟสการออกแบบจะเพิ่มเติมรายละเอียดเข้าไปในโมเดลต่างๆที่ได้จากการวิเคราะห์และจะแปลงโพรเซสระดับล่างในดาต้าโฟล์วไดอะแกรมแต่ละอันเป็นฟังก์ชันในผังโครงสร้าง (structure chart) ซึ่งใช้ภาษาที่เขียนโปรแกรมจริงๆ และผังโครงสร้างนี้จะแสดงการเรียกใช้โพรซีเยอร์ต่างๆเป็นรูปต้นไม้ (tree)

4.1.3 ผลกระทบของวิธีการแบบออบเจ็กต์โอเรียนเต็ด

ความแตกต่างจากการสร้างซอฟต์แวร์วิธีเก่า (Function-Oriented Methodology) จะทำให้การใช้วิธีแบบออบเจ็กต์โอเรียนเต็ดมีผลกระทบกับโพรเซสในการพัฒนาซอฟต์แวร์ และตัวซอฟต์แวร์ที่สร้างขึ้นโดยวิธีนี้ด้วย

1. เปลี่ยนไปเน้นการพัฒนาซอฟต์แวร์ที่การวิเคราะห์ก่อน

วิธีแบบออบเจ็กต์โอเรียนเต็ดต่างกับวิธีการแบบเก่า คือเน้นที่เฟสการวิเคราะห์ของไลฟ์ไซเคิล เราต้องใช้เวลาวิเคราะห์และออกแบบมากที่สุด แต่ซอฟต์แวร์ที่ได้จะดีกว่า ในแง่ของการปรับปรุงแก้ไขได้ง่าย

2. เน้นโครงสร้างของข้อมูลมากกว่าฟังก์ชัน

ทำให้โพรเซสการสร้างซอฟต์แวร์มีพื้นฐานที่ดี และมีหลักการของออบเจ็กต์ชัดเจนไปตลอดโพรเซสส่วนหลักการอื่นๆ เช่น ฟังก์ชัน (functions), รีเลชันชิพ (relationships) และ อีเวนท์ (events) จะถูกจัดการอยู่รอบๆออบเจ็กต์ ทำให้โนเตชัน (notations) ต่างๆระหว่างเฟสการวิเคราะห์ไม่สูญหาย หรือเปลี่ยนรูปไป เมื่อเข้าสู่เฟสการออกแบบ และอิมพลีเมนท์

โครงสร้างข้อมูลของแอพพลิเคชันและรีเลชันชิพระหว่างโครงสร้างข้อมูลเหล่านั้น จะอ่อนไหวต่อการเปลี่ยนแปลงความต้องการของผู้ใช้น้อยกว่าฟังก์ชันหรือโอเปอเรชันที่กระทำกับข้อมูล การจัดการระบบโดยใช้หลักการออบเจ็กต์แทนที่การจัดการโดยใช้ฟังก์ชันต่างๆ จึงทำให้โพรเซสการพัฒนาซอฟต์แวร์ถาวรมากกว่า นอกจากนี้ออบเจ็กต์ซึ่งมีพับบลิกอินเตอร์เฟส (public interface) และซ่อนการอิมพลีเมนท์ไว้ในไพรเวท (private) จึงไม่เปลี่ยนแปลงไปง่าย เมื่อมีการปรับปรุงซอฟต์แวร์

3. โพรเซสการพัฒนาที่ไม่มีรอยต่อ

วิธีการแบบออบเจ็กต์โอเรียนเต็ด จะกำหนดเซตของออบเจ็กต์ในแอพพลิเคชันโดเมนตั้งแต่แรกและเพิ่มเติมออบเจ็กต์ไปเรื่อยๆได้ตามต้องการในระหว่างการพัฒนา

อินฟอเมชันโมเดลที่ถูกสร้างขึ้นในเฟสการวิเคราะห์จะถูกนำไปใช้ต่อในเฟสการออกแบบ และอิมพลีเมนท์ และอาจจะมีการทบทวนโมเดลต่างๆอีกในระดับรายละเอียด ไม่ใช่เปลี่ยนแปลงโมเดลนี้เป็นโมเดลอื่นๆ โพรเซสการพัฒนาด้วยวิธีออบเจ็กต์โอเรียนเต็ดจึงเรียกว่า โพรเซสที่ไร้รอยต่อ เพราะ โนเต

ชันของเฟสหนึ่งๆจะไม่ถูกแทนที่ด้วยโนเตชันอื่นๆในเฟสอื่นๆ หรือมีการใช้โนเตชันของเฟสหนึ่งๆ ต่อเนื่องไปตลอดโพรเซส โดยไม่ถูก เปลี่ยนไปเป็นโนเตชันแบบอื่นๆ

4. ทำซ้ำได้หลาย ๆรอบ

ถึงแม้ว่าเฟสต่างๆในการพัฒนาซอฟต์แวร์ด้วย Object Lifecycles Modeling the World in States Methodology จะเป็นลิเนียร์ (linear) แต่โพรเซสการพัฒนาสามารถถูกทำซ้ำได้ ความไร้รอยต่อของการพัฒนาซอฟต์แวร์แบบออบเจ็กต์โอเรียนเต็ด ทำให้ทำการพัฒนาซอฟต์แวร์ในระดับรายละเอียดซ้ำได้หลายๆรอบ และง่ายขึ้น และแต่ละรอบของการทำซ้ำจะเป็นการเพิ่มเติม หรือ แคลลิฟายฟีเจอร์ (clarify features) ต่างๆ มากกว่าการปรับปรุงงานที่ทำไปแล้ว ดังนั้นจะมีโอกาสเกิดความไม่สอดคล้องกัน (inconsistencies) และ ความผิดพลาด (errors) น้อยมาก

4.1.4 SA/SD เปรียบเทียบกับ Object Lifecycles Modeling the World in States Methodology

ความแตกต่างระหว่าง 2 วิธีนี้ ได้แก่

1. เรื่องสไตล์ (style) และการเน้น (emphasis) ที่แตกต่างกัน

วิธีแบบออบเจ็กต์โอเรียนเต็ดและ SA/SD มีโมเดล (models) คล้ายกัน และสนับสนุนวิวทั้ง 3 ระบบเหมือนกันได้แก่ อินฟอเมชันโมเดล (information model), สเตทโมเดล (state model), และ โพรเซสโมเดล (process model)

ใน SA/SD จะเน้นโพรเซส โมเดล รองลงมาก็สเตทโมเดล และอินฟอเมชันโมเดลสำคัญน้อยที่สุด ในทางตรงข้าม Object Lifecycles Modeling the World in States Methodology เน้นอินฟอเมชันโมเดลเป็นสำคัญ และสเตทโมเดล และ สุดท้ายคือ โพรเซสโมเดล

วิธีแบบออบเจ็กต์โอเรียนเต็ด เน้นอินฟอเมชันโมเดล, หลักออบเจ็กต์ในโลกแห่งความจริง (real-world objects) และรีเลชันชิพของออบเจ็กต์เหล่านั้น ทำให้เราเข้าใจพฤติกรรมแบบไดนามิก และฟังก์ชันนอลของออบเจ็กต์ ในทางตรงกันข้าม SA/SD เน้นหลักการแตกแยกย่อยระบบเป็นฟังก์ชันต่างๆ

2. SA/SD จัดระบบเป็นโพรซีเยอร์ต่างๆ ในทางตรงข้าม Object Lifecycles Modeling the World in States Methodology จัดระบบเป็นออบเจ็กต์ในโลกแห่งความเป็นจริง (real-world objects) หรือ ออบเจ็กต์ในโลกของผู้ใช้ (conceptual objects) การเปลี่ยนแปลงความต้องการของผู้ใช้ส่วนใหญ่เป็นการเปลี่ยนในฟังก์ชันมากกว่าในออบเจ็กต์ ดังนั้นการเปลี่ยนแปลงเหล่านี้จะทำให้การออกแบบเป็นโพรซีเยอร์ต่างๆเสียหายได้

ในทางตรงข้าม การเปลี่ยนแปลงในฟังก์ชันเหล่านี้จะทำได้ง่ายในการออบแบบแบบออบเจ็กต์โอเรียนเต็ด โดยการเพิ่มเติมหรือแก้ไขโอเปอเรชัน โดยไม่ทำให้โครงสร้างพื้นฐานของออบเจ็กต์เปลี่ยนไปด้วย

SA/SD จึงเหมาะกับปัญหาที่เน้นความสำคัญที่ฟังก์ชัน และปัญหาที่ส่วนของฟังก์ชันซับซ้อนกว่าข้อมูล

3. การออกแบบแบบ SA/SD มีขอบเขตของระบบที่ชัดเจนแน่นอน

โครงสร้างของการออกแบบแบบ SA/SD ถูกจำกัดอยู่ในขอบเขตของระบบดังนั้น ทำให้ยากในการเพิ่มเติมการออกแบบนี้ไปยังขอบเขตของระบบใหม่

ในทางตรงข้ามเราสามารถเพิ่มเติมการออกแบบแบบออบเจ็กต์โอเรียนเต็ดได้ง่าย โดยแค่เพิ่มออบเจ็กต์ใหม่ และรีเลชันชิพกับออบเจ็กต์อื่นๆในระบบเข้าไปเท่านั้น

การออกแบบแบบออบเจ็กต์โอเรียนเต็ดจึงมีความยืดหยุ่นต่อการเปลี่ยนแปลงและเพิ่มเติมได้มากกว่า

4. วิธีแบบฟังก์ชันนอลของ Yourdon และ DeMarco เน้นการแบ่งระบบออกเป็นฟังก์ชันย่อยๆ จะเห็นได้ว่ามันเป็นวิธีที่ตรงที่สุดในการพัฒนาระบบ แต่ระบบที่ได้อาจเปราะบางเกินไป เช่นถ้ามีผู้ใช้เปลี่ยนความต้องการ ระบบอาจจะต้องเปลี่ยนแปลงโครงสร้างขนานใหญ่ ในทางตรงกันข้ามวิธีแบบ ออบเจ็กต์โอเรียนเต็ด เน้นการออกแบบระบบจากแอพพลิเคชันโดเมนเป็นออบเจ็กต์ต่างๆ จากนั้นค่อยใส่โพรซีเยอร์ให้มัน ถึงแม้ว่าวิธีนี้จะอ้อมกว่า แต่มันจะไม่ต้องเปลี่ยนแปลงมากเมื่อผู้ใช้เปลี่ยนความต้องการไป และที่สำคัญเพราะ พื้นฐานของวิธีการแบบออบเจ็กต์โอเรียนเต็ดอยู่บนแอพพลเคชันโดเมนอยู่แล้ว ไม่ใช่เพียงแต่มีพื้นฐานอยู่บนความต้องการของผู้ใช้หรือปัญหาของผู้ใช้เพียงปัญหาเดียวเท่านั้น

5. ความต่อเนื่องโดยตรงระหว่างออบเจ็กต์ในการออกแบบแบบออบเจ็กต์โอเรียนเต็ด และออบเจ็กต์ในโดเมนของปัญหา (problem domain) ทำให้ระบบง่ายต่อการทำความเข้าใจ และ ทำให้การออกแบบที่สร้างขึ้นมีลักษณะเป็นไปตามสัญชาติญาณ ผู้บำรุงรักษาซอฟต์แวร์จึงสามารถดูโค้ทของซอฟต์แวร์และ ความต้องการของผู้ใช้ประกอบกันไปได้ และบำรุงรักษาซอฟต์แวร์ได้ง่ายโดยที่เราไม่ต้องเป็นส่วนหนึ่งในทีมของผู้ออกแบบ

ใน SA/SD การแตกโพรเซสหนึ่งออกเป็นโพรเซสย่อยๆ เป็นไปตามความต้องการของผู้พัฒนาเอง ผู้พัฒนาต่างๆกันจะทำการแตกโพรเซสไม่เหมือนกัน ในการออกแบบแบบออบเจ็กต์โอเรียนเต็ด, ผู้พัฒนาที่พัฒนาโปรแกรมที่แตกต่างกันในโดเมนของปัญหาเดียวกันมักมีแนวโน้มที่จะเลือกออบเจ็กต์ต่างๆออก

มาคล้ายกัน การเลือกที่คล้ายกันเช่นนี้จะเป็นการเพิ่มความสามารถในการใช้งานใหม่ของออบเจ็กต์จากโปรเจ็คหนึ่งไปใช้ในโปรเจ็คอื่นๆได้อีก

6. วิธีการแบบออบเจ็กต์โอเรียนเต็ดจะทำให้การรวมฐานข้อมูลเข้ากับโปรแกรมมิ่งโค้ทได้ดีกว่า จากหลักการของออบเจ็กต์ เราสามารถใช้ออบเจ็กต์เพื่อสร้างโมเดลของฐานข้อมูล หรือสร้างโมเดลของโครงสร้างของโปรแกรมได้ นอกจากนี้การค้นคว้าในเรื่องฐานข้อมูลแบบออบเจ็กต์โอเรียนเต็ดจะทำให้การสร้างโมเดลทำได้ง่ายขึ้น

ในทางตรงกันข้าม การออกแบบแบบโพรซีเยอร์จะงุ่มง่ามเมื่อต้องจัดการกับฐานข้อมูล และมันยากที่จะเมิร์จ (merge) โค้ทซึ่งจัดการกับฟังก์ชันเข้ากับฐานข้อมูลซึ่งจัดการกับข้อมูล

มีหลายเหตุผลที่วิธีการของดาต้าโฟล์วมีการใช้งานกว้างขวางกว่า โปรแกรมเมอร์มักจะคิดในเทอมของฟังก์ชัน ดังนั้นวิธีการของดาต้าโฟล์วจึงเรียนรู้ได้ง่ายกว่า อีกเหตุผลหนึ่งคือ ประวัติศาสตร์ เพราะ SA/SD เป็นหนึ่งในวิธีการแรกที่ถูกคิดค้นขึ้นอย่างเป็นทางการเพื่อพัฒนาซอฟต์แวร์และระบบ

เราเชื่อว่า ข้อดีของวิธีการแบบออบเจ็กต์โอเรียนเต็ด และความสุกงอมของเทคโนโลยีออบเจ็กต์โอเรียนเต็ดจะเป็นตัวโปรโมทการนำไปใช้วิเคราะห์, ออกแบบ, และอิมพลีเมนท์ระบบได้

ข้อดีของภาษาออบเจ็กต์โอเรียนเต็ด (OBJECT-ORIENTED LANGUAGE)เมื่อเปรียบเทียบกับภาษาคอนเว็นชันแนล (CONVENTIONAL LANGUAGE)

เปรียบเทียบในด้านของคุณภาพ (QUALITY) ของซอฟต์แวร์โดยใช้หลักในหัวข้อดังต่อไปนี้

⇒ ความถูกต้อง (CORRECTNESS)

⇒ ความยืดหยุ่นและความน่าเชื่อถือของโปรแกรมจะต้องครอบคลุมทุกเงื่อนไขที่เกิดขึ้นเพื่อป้องกันความผิดพลาด

⇒ การบำรุงรักษา โปรแกรมต้องบำรุงรักษาง่ายและสามารถเพิ่มความสามารถได้เมื่อต้องการ

⇒ การนำกลับมาใช้ใหม่ โปรแกรมควรสร้างจากโมดูลที่สามารถนำกลับมาใช้ใหม่ได้อีก

⇒ โปรแกรมควรเป็นระบบเปิด คือสามารถใช้ได้กับระบบอื่น ๆ ด้วย

⇒ ประสิทธิภาพ

⇒ โปรแกรมควรใช้ได้กันได้กับฮาร์ดแวร์และระบบปฏิบัติการที่ใช้

⇒ ความสามารถในการตรวจสอบได้ (VERIFIABILITY)

⇒ ความปลอดภัย สามารถซ่อนรายละเอียดของข้อมูล , โนวเลช (KNOWLEDGE) หรือฟังก์ชันได้

⇒ ความถูกต้องตามกฎที่กำหนดไว้ (INTEGRITY)

⇒ เฟรนลีเนส (FRIENDLINESS)

⇒ เดสคริบอบิลิตี (DESCRIBABILITY)

⇒ อันเดอร์สแตนอบิลิตี (UNDERSTANDABILITY)

จากคุณสมบัติในหัวข้อที่กล่าวมา ภาษาออบเจ็กต์โอเรียนเต็ดจะสนับสนุนดังนี้

- เมื่อสร้างระบบด้วยภาษาที่เป็นออบเจ็กต์โอเรียนเต็ด จะง่ายต่อการพัฒนาระบบในอนาคต เพราะเมื่อทำการเปลี่ยนแปลงแก้ไขก็จะมีผลกระทบหรือเปลี่ยนแปลงเฉพาะบางส่วนเท่านั้นไม่ได้กระทบทั้งหมด ซึ่งตรงกับคุณสมบัติในด้านการบำรุงรักษา
- ด้วยคุณสมบัติของโพลีมอฟฟิซึม, ไดนามิกไบดิงและอินเฮอริท ทำให้ผู้พัฒนาซอฟต์แวร์สามารถสร้างคลาสไลบรารีซึ่งนำมาใช้ใหม่ได้ (REUSABLE) ทำให้เกิดความสะดวกรวดเร็วซึ่งตรงกับคุณสมบัติในด้านการนำกลับมาใช้ใหม่
- เมื่อนำคลาสที่ประกาศไว้แล้วมาใช้ทำให้โปรแกรมที่สร้างขึ้นมีข้อผิดพลาดน้อยลง เนื่องจากมีการตรวจสอบข้อผิดพลาดก่อนแล้ว
- เนื่องจากการติดต่อระหว่างออบเจ็กต์จะเป็นเพียงการส่งแมสเสจกันเท่านั้น ทำให้การอินเตอร์เฟสระหว่างโมดูลกับระบบภายนอกเป็นไปได้ง่ายขึ้น
- ด้วยคุณสมบัติเอ็นแคปซูเลชัน ทำให้ลดความซับซ้อนในระบบที่แบ่งงานออกเป็นส่วนๆ (PARTITIONING SYSTEM)
- ในงานที่ต้องแบ่งออกเป็นส่วนย่อย ๆ การใช้วิธีออบเจ็กต์โอเรียนเต็ดจะเหมาะสมกว่าการใช้ทอปดาวน์ฟังก์ชัน (TOP-DOWN FUNCTIONAL)
- ด้วยคุณสมบัติอินฟอร์เมชันไฮดิง จะช่วยสร้างความปลอดภัยให้แก่ระบบ
- ขั้นตอนการสร้างโปรแกรมเป็นไปอย่างมีระเบียบแบบแผน ทำให้โปรแกรมที่ได้มีความถูกต้อง
- ในหัวข้อการบำรุงรักษา, การนำกลับมาใช้ใหม่ และความสามารถในการตรวจสอบได้ ภาษาออบเจ็กต์โอเรียนเต็ดก็ได้สนับสนุนในกรณีที่โปรแกรมที่สร้างขึ้นง่ายที่จะเพิ่มเติมและพร้อมที่จะถูกใช้งาน แต่ในภาษาคอนเว็นชันแนล เช่นในภาษาซี ในกรณีของ CASE STATEMENT ตามโค้ดต่อไปนี้

```
Switch (n) {
        case 1: error = 49;
                printf("Warning - Error 49"); continue;
        case 2: error = 71;
                printf("Execution halting due to Error 71"); break;
        case 3;
```

```
                break;
        }
```

จะเห็นว่าโค้ดเช่นนี้ขาดคุณสมบัติการนำไปใช้ใหม่และ EXTENSIBILITY ไป

- เปรียบเทียบในด้านของความเป็นโมดูล (MODULARITY) ของซอฟต์แวร์สามารถแบ่งออกเป็น 5 หัวข้อดังต่อไปนี้
  - ◆ ดีคอมโพสสิบิลิตี (DECOMPOSABILITY)
  - ◆ คอมแพทอบิลิตี (COMPOSABILITY)
  - ◆ อันเดอร์สแตนอบิริตี (UNDERSTANDABILITY)
  - ◆ คอนทินูอิตี (CONTINUITY)
  - ◆ โพรเท็กชัน (PROTECTION)

ดีคอมโพสสิบิลิตี ระบบสามารถแบ่งออกเป็นส่วน ๆ ได้เพื่อสะดวกในการจัดการและการเปลี่ยนแปลง ดังนั้นสามารถแบ่งทีมในการพัฒนาโปรแกรมได้ ถ้าหากใช้ภาษาคอนเว็นชันแนลซึ่งเป็นการแบ่งระบบแบบทอปดาวน์ จะทำได้ในฟังก์ชันระดับแรกเท่านั้น เช่นระบบปฏิบัติการอาจมีความจำเป็นต้องสร้างโมดูลซึ่งมีการติดต่อกับโมดูลอื่นยุ่งยากและซับซ้อนมาก แต่ถ้าใช้ภาษาออบเจ็กต์โอเรียนเต็ดซึ่งเป็นการแบ่งแบบบอททอมอัพ (BOTTOM-UP) และด้วยคุณสมบัติอินฟอร์เมชันไฮดิง จะทำให้การติดต่อง่ายขึ้น

คอมแพทอบิลิตี เป็นความสามารถที่จะเพิ่มรายละเอียดเข้าไปในระบบโดยอิสระ ไม่ทำให้ส่วนอื่นกระทบกระเทือน เช่นการเพิ่มฟังก์ชันเข้าไปในระบบ ซึ่งภาษาออบเจ็กต์โอเรียนเต็ดก็ได้สนับสนุนคุณสมบัติข้อนี้เช่นกัน

อันเดอร์สแตนอบิริตี ช่วยให้ผู้ใช้ทำการเรียนรู้ระบบได้เร็วยิ่งขึ้นเมื่อได้พิจารณาโปรแกรมเนื่องจากการติดต่อสื่อสารใรภาษาออบเจ็กต์โอเรียนเต็ดนั้นจะเป็นเพียงการรับส่งแมสเสจระหว่างออบเจ็กต์เท่านั้น

คอนทินูอิตี เมื่อทีการเปลี่ยนแปลงมันจะมีผลกระทบกับบางส่วนเท่านั้น ซึ่งภาษาออบเจ็กต์โอเรียนเต็ดได้สนับสนุนอยู่แล้ว

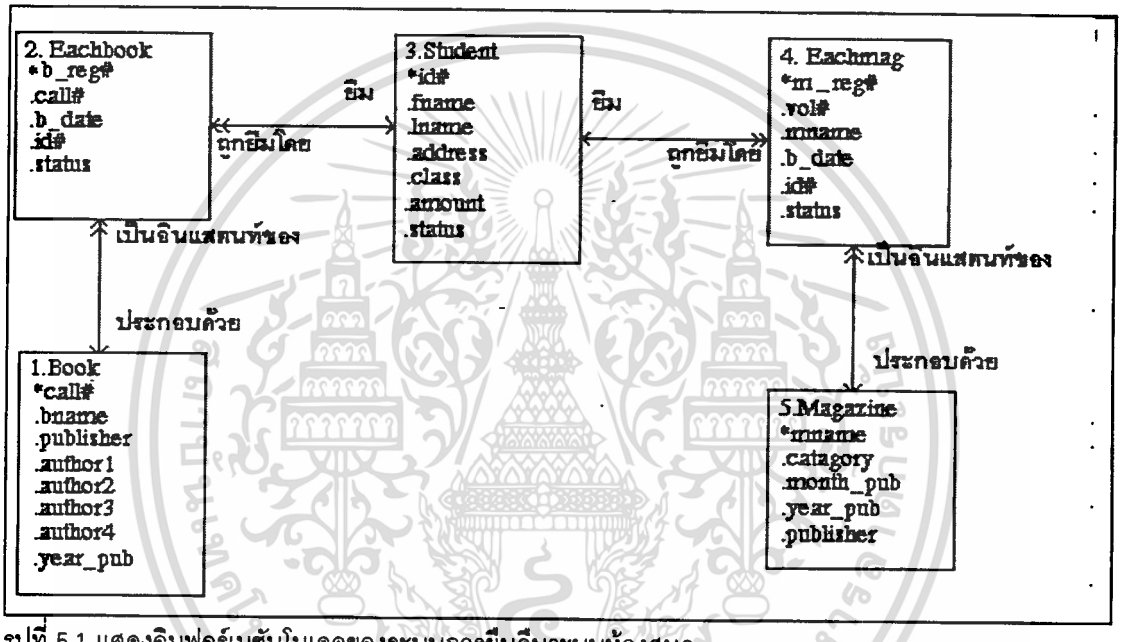
โพรเท็กชัน เป็นเงื่อนไขในการเกิดข้อผิดพลาดและการยกเว้นของโมดูลเพื่อใช้ในการกำหนดขอบเขตของโมดูล

นอกจากทางด้านคุณภาพและความเป็นโมดูลแล้ว ภาษาออบเจ็กต์โอเรียนเต็ดเมื่อนำมาใช้บนระบบเครือข่าย จะช่วยลดโหลดในการติดต่อสื่อสารกันระหว่างออบเจ็กต์ได้เพราะส่งเพียงแค่แมสเสจ

บทที่ 5
การอิมพลีเมนต์ระบบงาน

5.1 แอพพลิเคชันระบบการยืมคืนห้องสมุด

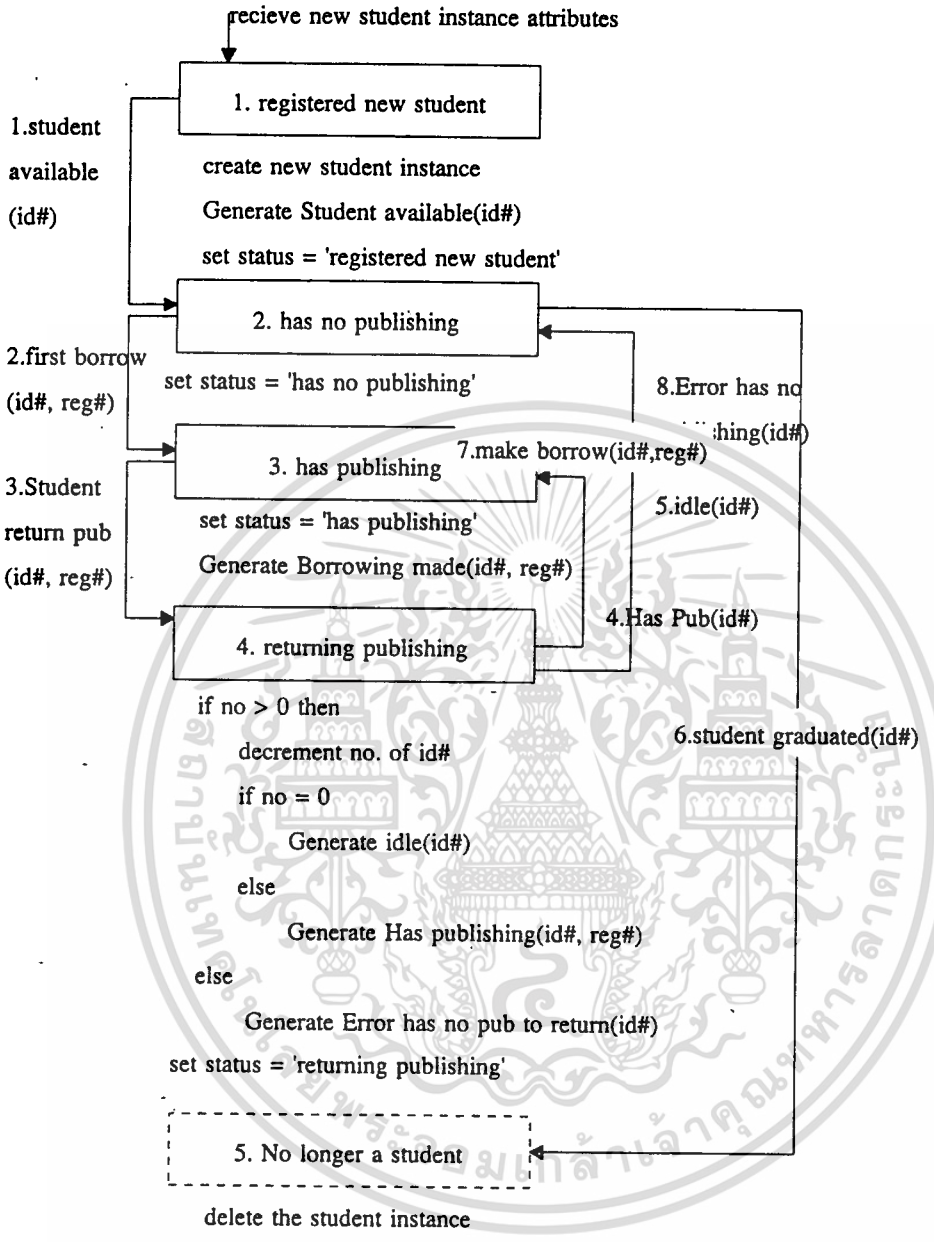
5.1.1 อินฟอร์เมชันโมเดลของระบบการยืมคืนห้องสมุด แสดงดังรูปที่ 5.1



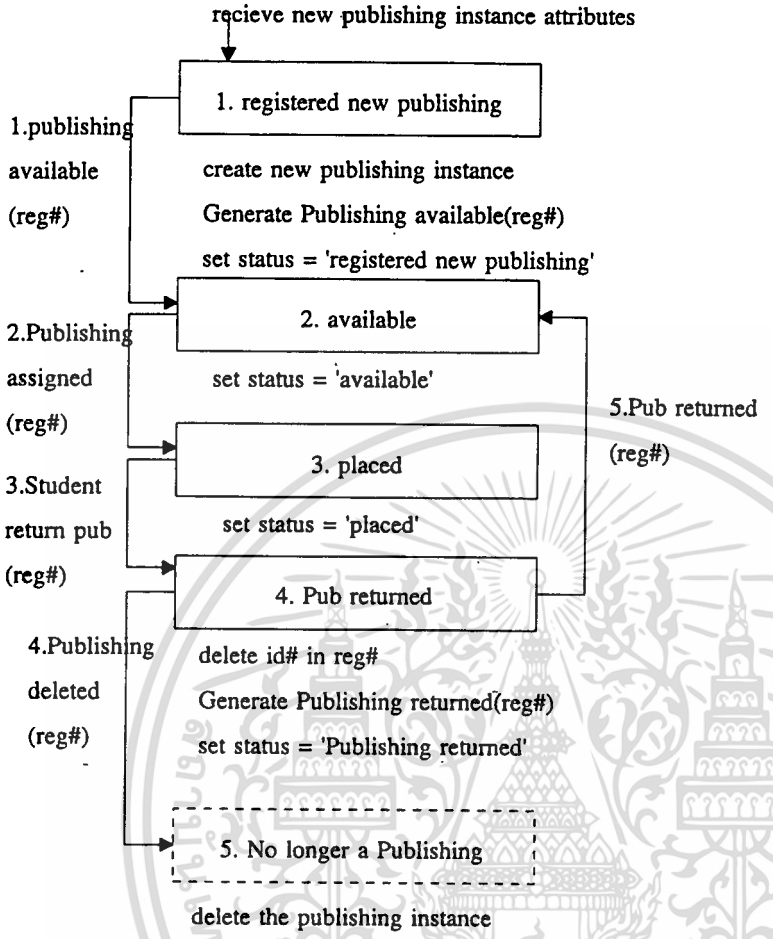
รูปที่ 5.1 แสดงอินฟอร์เมชันโมเดลของระบบการยืมคืนระบบห้องสมุด

5.1.2 สเตทโมเดลของออบเจ็กต์ที่ได้จากอินฟอร์เมชันโมเดล

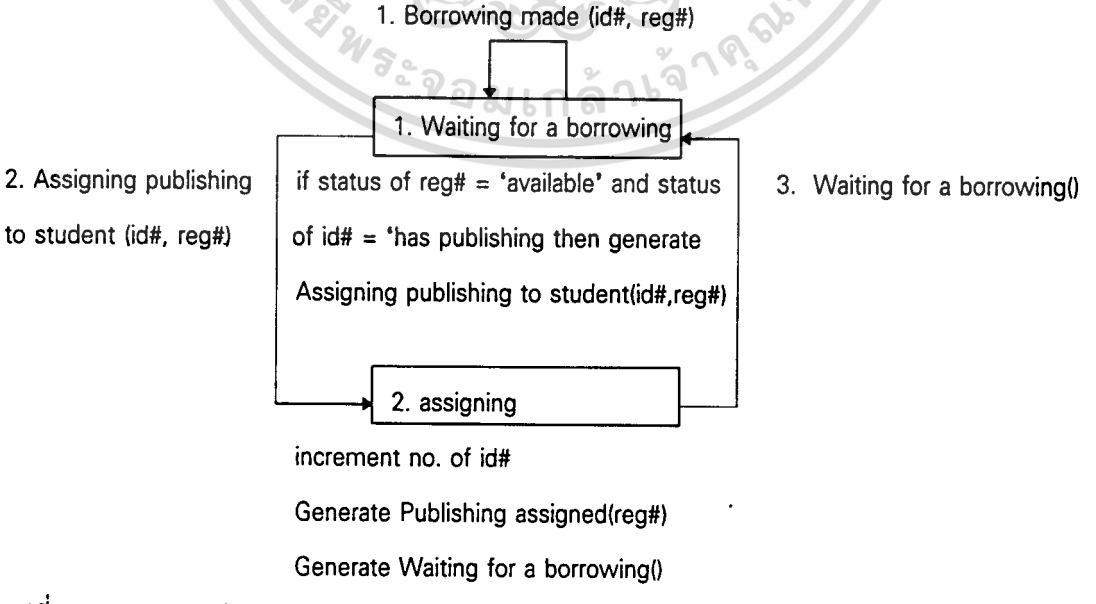
เมื่อได้อินฟอร์โมเดลแล้วนำออบเจ็กต์แต่ละตัวมาเขียนสเตทต่าง ๆ ที่เกิดขึ้น โดยรูปที่ 5.2 แสดงสเตทโมเดลของนักศึกษา, รูปที่ 5.3 แสดงสเตทโมเดลของสิ่งพิมพ์ และรูปที่ 5.4 แสดงสเตทโมเดลของแอสซายเนอร์



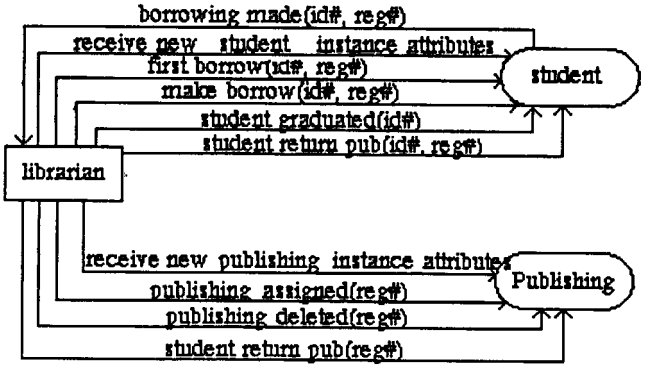
รูปที่ 5.2 แสดงสเตทโมเดลของนักศึกษา



รูปที่ 5.3 แสดงสเตทโมเดลของสิ่งพิมพ์



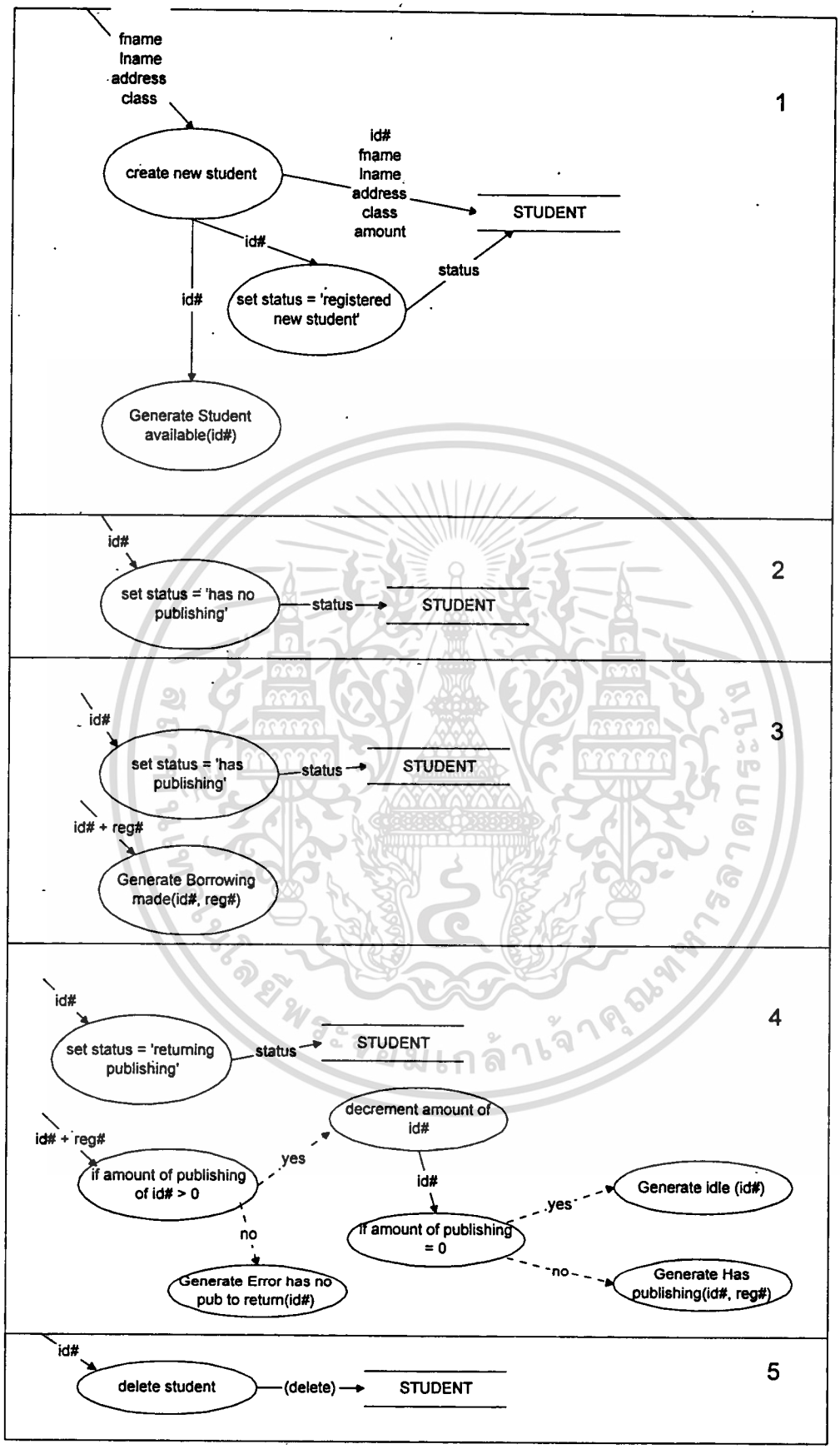
รูปที่ 5.4 แสดงสเตทโมเดลของแอสซายเนอร์



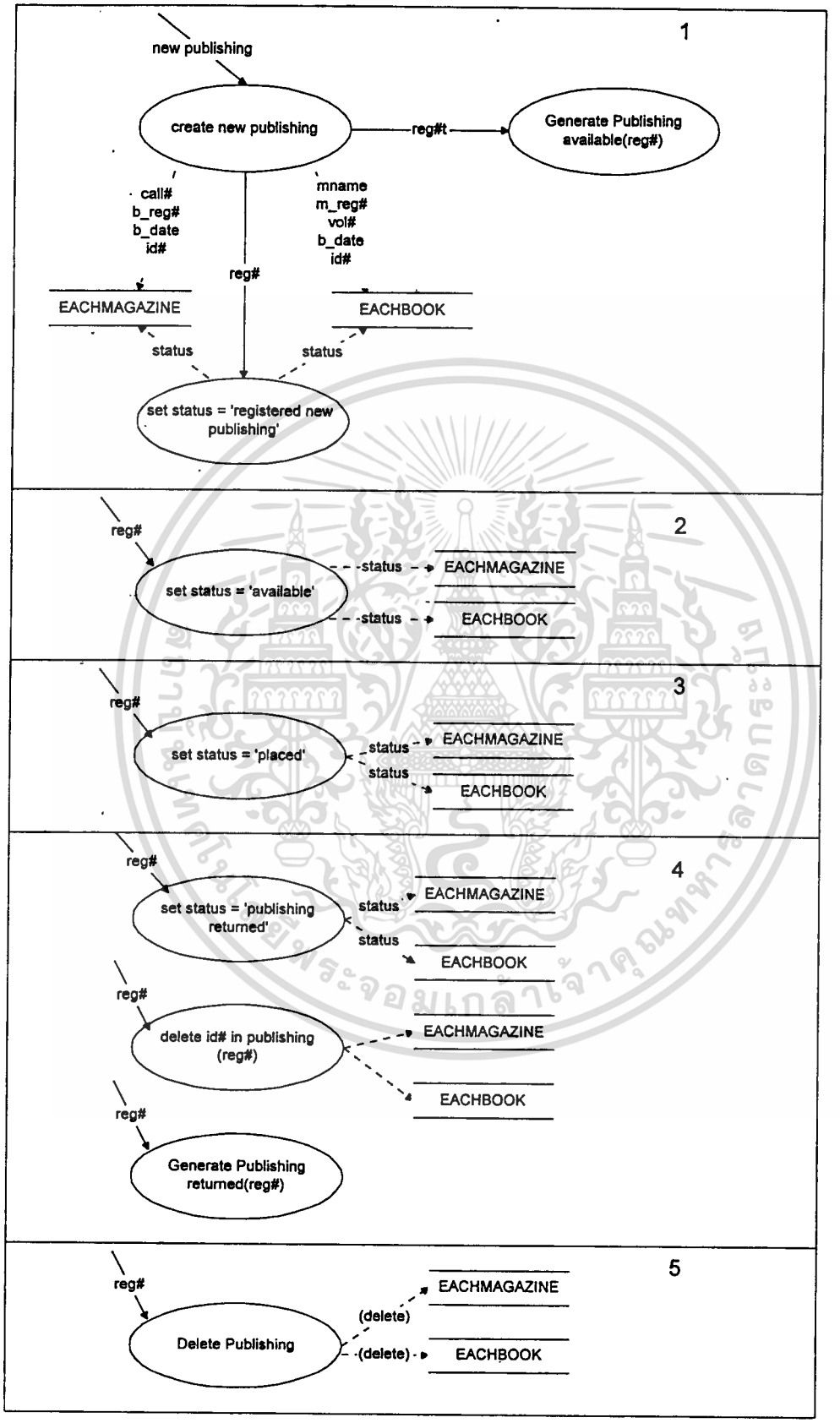
รูปที่ 5.5 แสดงแบบจำลองการติดต่อระหว่างออบเจ็กต์ (Object Communication Model)

5.1.3 โพรเซสโมเดล

เมื่อได้สเตทโมเดลแล้วนำแต่ละสเตทมาเขียนโพรเซสโมเดลดังรูปที่ 5.6, 5.7, และ 5.8 โดยจะแสดงโพรเซสโมเดลของ นักศึกษา, สิ่งพิมพ์ และ แอสซายเนอร์ และรูปที่ 5.9 จะแสดงฐานข้อมูลของระบบงานห้องสมุด



รูปที่ 5.6 ภาพแสดงโพรเซสโมเดลของทุกสเตทตามลำดับของ student

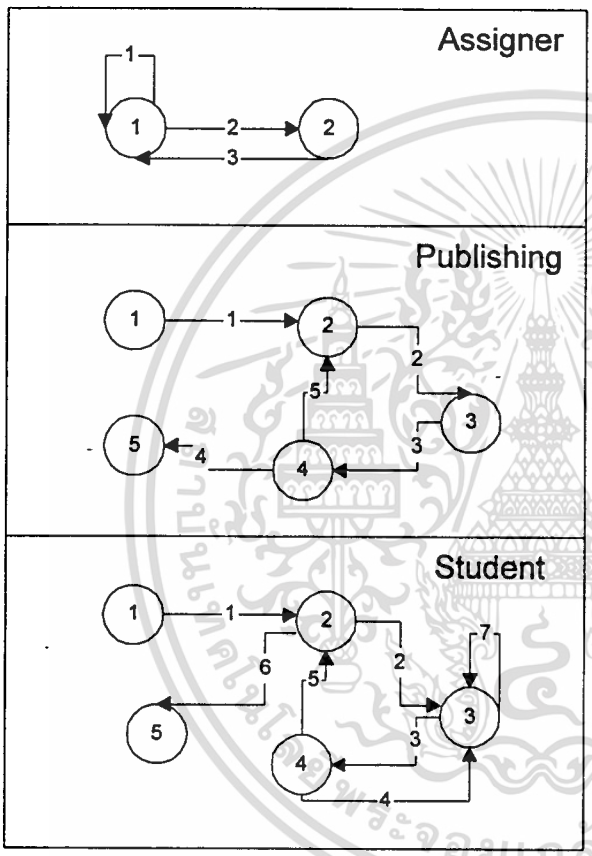


รูปที่ 5.7 ภาพแสดงโพรเซสโมเดลของทุกสเตทของ Publishing ตามลำดับ

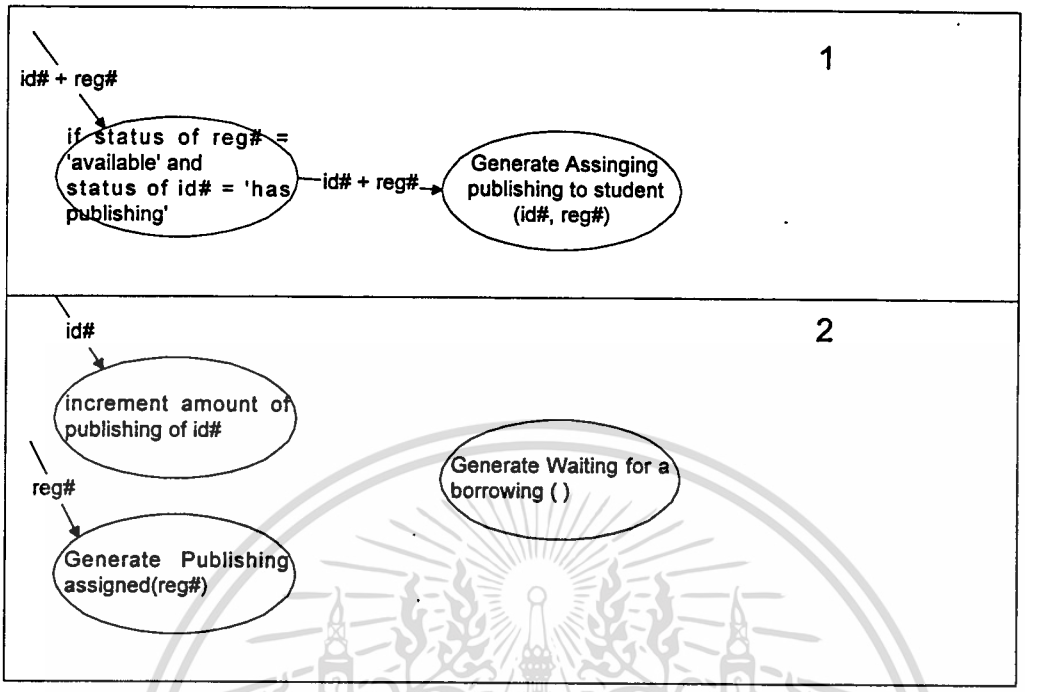
FSM

FSMNAME	CURRENTSTATE	EVENT	NEWSTATE
char	smallint	smallint	smallint

รูปที่ 5.9 แสดงฐานข้อมูลของระบบงานห้องสมุด



รูปที่ 5.10 แสดงไฟไนท์สเตทแมชชีนของคลาสต่างๆ



รูปที่ 5.8 ภาพแสดงโพรเซสโมเดลของทุกสเตทของ assigner ตามลำดับ

STUDENT

ID#	FNAME	LNAME	ADDRESS	CLASS	AMOUNT	STATUS
float	char 10	char 20	char 30	char 2	float	smallint

BOOK

CALL#	BNAME	PUBLISHER	AUTHOR1	AUTHOR2	AUTHOR3	AUTHOR4	YEAR_PUB
char 18	char 40	char 30	char 30	char 30	char 30	char 30	integer

EACHBOOK

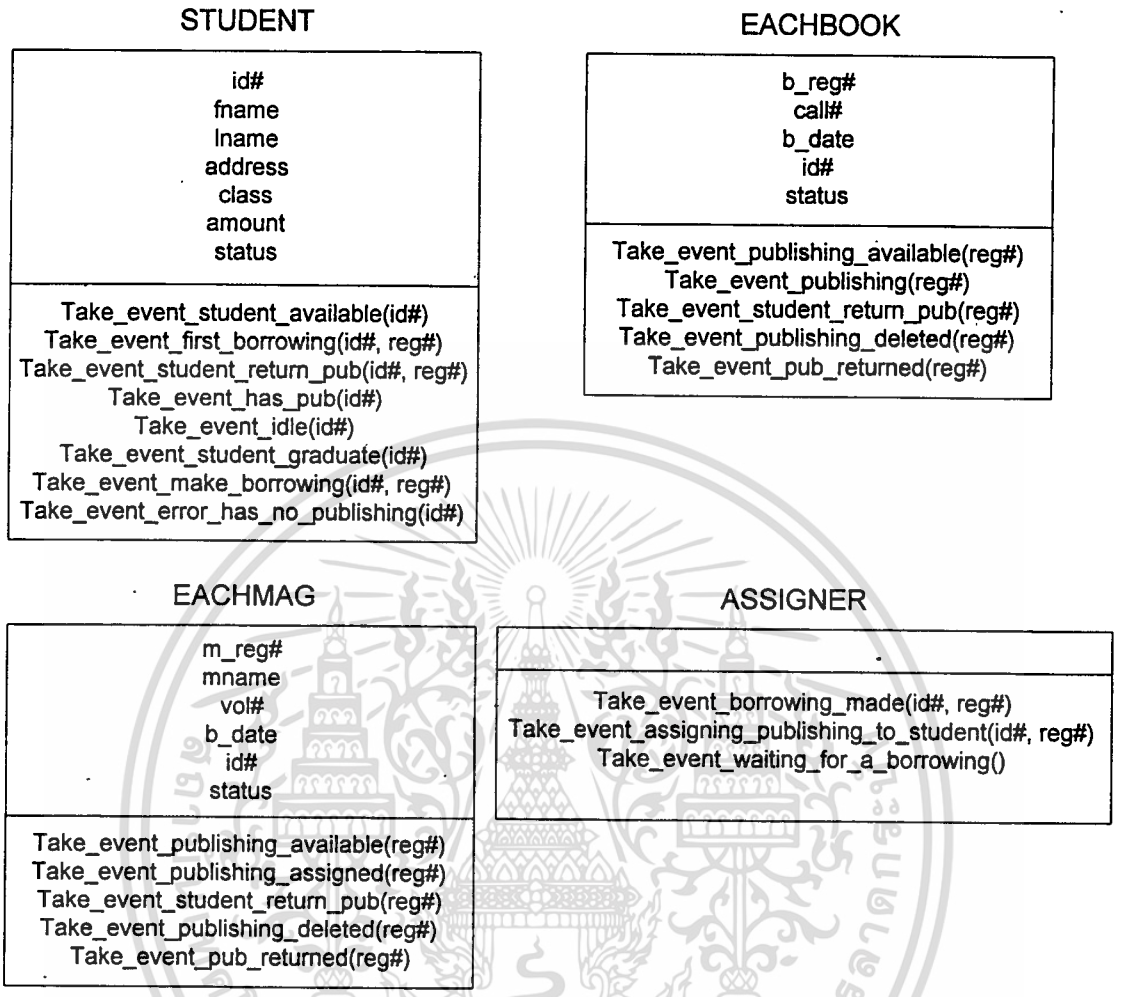
B_REG#	CALL#	B_DATE	ID#	STATUS
char 6	char 18	date	float	smallint

MAGAZINE

MNAME	CATAGORY	MONTH_PUB	YEAR_PUB	PUBLISHER
char 30	smallint	smallint	integer	char 30

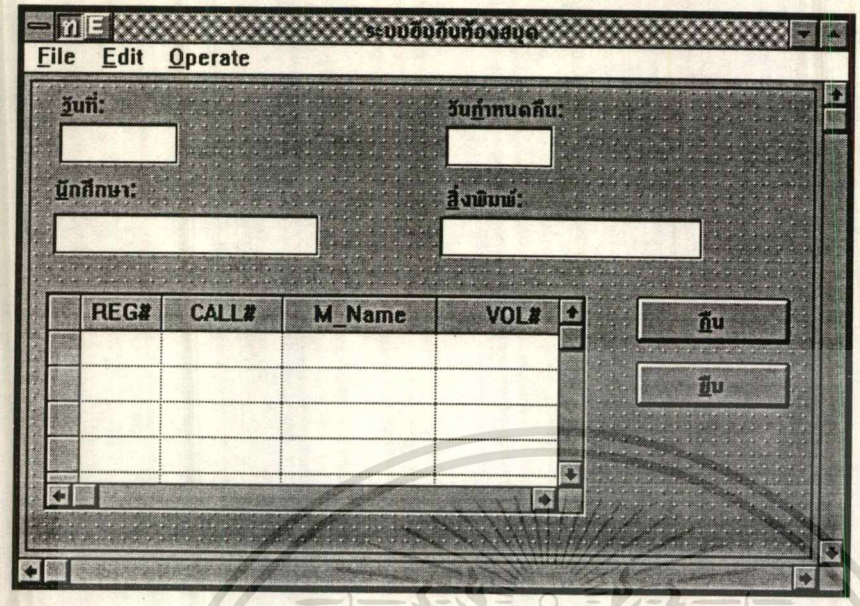
EACHMAG

M_REG#	MNAME	VOL#	B_DATE	ID#	STATUS
float	char 30	integer	date	float	smallint



รูปที่ 5.11 แสดงแอพพลิเคชันคลาส, แอคทีฟคลาส

รูปที่ 5.12 แสดงแอพพลิเคชันคลาส, แพสซีฟคลาส



รูปที่ 5.13 แสดงหน้าจอของระบบยืมคืนห้องสมุด ซึ่งสามารถยืมและคืนสิ่งพิมพ์ได้ดังที่ได้กล่าวไปแล้วในตอนต้นของบทนี้ และมีเมนู Operate ให้เลือก Borrow หรือ Submission

การยืมสิ่งพิมพ์

เมื่อเริ่มโปรแกรม เราจะเซตให้เลือกเมนู Borrow โดยอัตโนมัติและจะเซตโฟกัสอยู่ที่ฟิลด์นักศึกษาเพื่อให้ผู้ใช้ใส่รหัสนักศึกษาที่จะยืมสิ่งพิมพ์และรหัสสิ่งพิมพ์ที่จะยืม เมื่อใส่รหัสนักศึกษาในฟิลด์แล้ว ระบบจะเช็คว่ามีนักศึกษาคนนั้นอยู่จริงหรือไม่ในฐานข้อมูล ถ้ามีก็จะแสดงสิ่งพิมพ์ที่นักศึกษาคนนั้นยืมไปทั้งหมดออกมาในตารางบนฟอร์มหลัก ส่วนถ้านักศึกษาคนนี้ไม่มีอยู่ในฐานข้อมูล ก็จะมีแมสเซจเตือนจากระบบทันที และเมื่อใส่รหัสสิ่งพิมพ์ในฟิลด์แล้ว ถ้าสิ่งพิมพ์ไม่มีคนยืมอยู่ และนักศึกษามีสิทธิ์ที่จะยืมหนังสือได้อีก (ยังยืมสิ่งพิมพ์ไม่ครบ 3 เล่ม) จะสามารถกดปุ่มยืมได้ หรือมิฉะนั้นระบบก็จะเตือนว่า สิ่งพิมพ์ได้ถูกยืมไปแล้ว หรือนักศึกษาคนนี้ไม่มีสิทธิ์ยืมได้อีก เมื่อกดปุ่มยืมแล้วจะมีการอัพเดตฐานข้อมูลโดยใส่รหัสนักศึกษาที่ยืมสิ่งพิมพ์นั้นและวันที่ยืม ลงในคอลัมน์ผู้ยืมสิ่งพิมพ์และคอลัมน์วันยืมตามลำดับในตาราง eachbook หรือ eachmag และมีการบวกเพิ่มจำนวนสิ่งพิมพ์ที่ยืมเข้าไปในคอลัมน์จำนวนสิ่งพิมพ์ที่ยืมในตารางนักศึกษา

การคืนสิ่งพิมพ์

เมื่อเราต้องการคืนสิ่งพิมพ์ ให้เลือกเมนู Operate, Submission ซึ่งจะเซตโฟกัสอยู่ที่ฟิลด์สิ่งพิมพ์เพื่อให้ผู้ใช้ใส่รหัสสิ่งพิมพ์ที่จะคืน ถ้าสิ่งพิมพ์นั้นมีอยู่จริงระบบก็จะแสดงในตารางบนฟอร์มหลัก และเช็คว่าสิ่งพิมพ์นั้นยังไม่ได้ถูกคืนก็จะสามารถกดปุ่มคืนได้ เมื่อกดปุ่มคืนจะมีการลบคอลัมน์ผู้ยืมสิ่งพิมพ์เล่มนั้นในตาราง eachbook หรือ eachmag และลดจำนวนสิ่งพิมพ์ที่นักศึกษาเป็นผู้ยืมลงหนึ่งเล่ม

การทำงานของระบบจะเป็นไปตามสเตทโมเดลของแต่ละออบเจ็กต์ และมีการอัพเดตสถานะของอินแสตนท์ลงในฐานข้อมูลด้วย

5.2 แอพพลิเคชันการพยากรณ์ดวงชะตาด้วยเลข 7 ตัว

การพยากรณ์ดวงชะตาด้วยเลข 7 ตัว จากตำราพรหมชาติ ฉบับชาวบ้าน จะใช้วันเดือนปีเกิดทางจันทรคติของผู้ต้องการพยากรณ์แต่ละคน มาเขียนเป็นฐานทั้งสามได้แก่ ฐานวัน, ฐานเดือน, และฐานปี เพื่อเป็นข้อมูลในการทำนาย ผู้ต้องการพยากรณ์มักไม่ทราบวันเดือนปีเกิดทางจันทรคติของตนเอง ดังนั้นในการเขียน แอพพลิเคชันจริง เราจะคีย์ข้อมูลของปฏิทิน 100 ปีลงในฐานข้อมูลบางส่วนที่สุ่มว่าจะมีผู้เกิดมาก เพื่อความรวดเร็วในการทำงาน หรือนอกเหนือจากนั้นเราจะใช้การเปิดปฏิทิน 100 ปี เอง เพื่อค้นหาวันเดือนปีเกิดทางจันทรคติให้ ตัวอย่างผู้ที่เกิดวันเสาร์ เดือนสาม ปีเถาะ จะนำมาเขียนเป็นเมตริกซ์ 3 แถว 7 คอลัมน์ ได้ดังรูป

7 1 2 3 4 5 6

อัตตะ หินะ ธะนัง ปิตา มาตา โภคา มัชฌิมา

3 4 5 6 7 1 2

ตนุ กดุมภะ สหัชชะ พันธุ ปุตตะ หริ ปัตตนิ

4 5 6 7 1 2 3

มรณะ ศุภะ กรรมะ ลาภะ พยายะ ทาสี ทาสา

5.2.1 การสร้างเมตริกซ์ 3*7

การเขียนตัวเลขในแต่ละแถวจะมีแค่ตัวเลข 1,2,3,..,7

ถ้าเกิดวันอาทิตย์ก็ตรงกับเลข 1 และรันหมายเลขไปเรื่อยๆจนถึงวันเสาร์ก็ตรงกับหมายเลข 7 เช่น คนเกิดวันศุกร์ ตรงกับหมายเลข 6 จึงใช้เลข 6 เป็นหมายเลขเริ่มต้นและตั้งฐานวันได้ ดังนี้

6 7 1 2 3 4 5

อัตตะ หินะ ธะนัง ปิตา มาตา โภคา มัชฌิมา

ถ้าเกิดเดือนอ้ายนับเป็นเลข 1 และรันหมายเลขไปเรื่อยๆ ถ้าเลขเกิน 7 ก็ให้ลบด้วย 7 เช่น คนที่เกิดเดือน 5 และเดือน 12 จะตรงกับหมายเลข 5 จึงใช้เลข 5 เป็นหมายเลข เริ่มต้นและตั้งฐานเดือนได้ดังนี้

5 6 7 1 2 3 4

ตนุ กดุมภะ สหัชชะ พันธุ ปุตตะ หริ ปัตตนิ

การเขียนเลขของปีในฐานที่ 3 ก็เช่นเดียวกันกับเดือนโดยนับปีชวดเป็นเลข 1 เช่น คนเกิดปีขาล (ปีที่ 3) หรือปีระกา(ปีที่ 10) จะตรงกับหมายเลข 3 และตั้งฐานปีได้ดังนี้

3 4 5 6 7 8 9

มรณะ ศุภะ กรรมะ ลาภะ พยายะ ทาสี ทาสา

5.2.2 ความหมายของชื่อเรื่องในแต่ละฐาน

อักษรกำกับในแต่ละคอลัมน์และแถวจะหมายถึงชื่อเรื่องที่ผู้ใช้เลือกดูคำทำนายได้ ได้แก่

<u>ชื่อเรื่องในฐานวัน</u>

'อัตตะ' หมายถึง ตัวตน ระบุถึงกำเนิดของตนว่าเป็นอย่างไร จะดีหรือร้ายอย่างไร

'หินะ' หมายถึง ความชั่วร้ายที่อาจบังเกิดขึ้นแก่ตน แต่ก็ไม่ใช่ภัยอันตรายร้ายแรงเท่าใดนัก

'ธะนัง' หมายถึง เงินทองที่หามาได้ จะมากหรือน้อยเพียงใดเหมาะสมแก่ตนหรือไม่อย่างไร

'ปิตา' หมายถึง พ่อผู้บังเกิดเกล้าหรือญาติพี่น้องทางฝ่ายบิดา ตลอดจนผู้ที่นับถือเสมอด้วยบิดา

'มาตา' หมายถึง แม่ผู้บังเกิดเกล้า หรือแม่ผู้ให้กำเนิด หรือญาติพี่น้องทางฝ่ายแม่ ตลอดจนผู้ที่ตน นับถือเสมอด้วยแม่ของตน

'โภคา' หมายถึง ทรัพย์สมบัติ อันหมายถึงที่ดิน เรือกสวนไร่นา ตลอดจนสิ่งของอันได้แก่แก้ว แหวน ทอง และรูปพรรณต่างๆ

'มัชฌิมา' หมายถึง ปานกลาง คือไม่มีส่วนชั่วร้าย หรือส่วนที่ดีแต่ประการใด

<u>ชื่อเรื่องในฐานเดือน</u>

'ตนุ' หมายถึง ตนเอง เป็นการบอกให้ทราบถึงตนเองเป็นสำคัญ

'กฎุมภะ' หมายถึง ทรัพย์สินเงินทองว่าจะมีมากน้อยเพียงใด

'พันธุ' หมายถึง ญาติพี่น้องฝ่ายบิดามารดาของตน หรือพี่น้องร่วมบิดาหรือมารดาฝ่ายใดฝ่ายหนึ่ง ในแบบที่เรียกว่าญาติวงศ์พงศ์พันธุ์

'ปุตตะ' หมายถึง เชื้อสายที่สืบสกุลของตน อันได้แก่ลูกชายลูกสาวของตน

'หริ' หมายถึง ข้าศึกศัตรู ทั้งนี้คืออุปสรรคที่เข้ามาขัดขวางหรือภัยอันตรายที่จะเข้ามาพัวพันใน ชีวิตของตน

'ปัตตนิ' หมายถึง เนื้อของตน ถ้าเป็นชายก็หมายถึงภรรยาของตนแต่ถ้าเป็นหญิงก็หมายถึงสามีของตน

<u>ชื่อเรื่องในฐานปี</u>

'มรณะ' หมายถึง ความตาย หรือโชคร้ายที่มีความรุนแรงมาก

'ศุภะ' หมายถึง ความดี ความสวย ความงาม ซึ่งเกี่ยวเนื่องไปถึงเกียรตินิยมและความสุขสมบูรณ์ อีกด้วย

'กรรมะ' หมายถึง การกระทำ ทั้งนี้ก็คือกิจการในหน้าที่ของตน ตลอดจนความขยันหมั่นเพียรในการประกอบกิจการของตนอีกด้วย

'ลาภะ' หมายถึง สิ่งที่ตนได้มา หรือความสำเร็จในกิจการงานต่างๆอันเป็นส่วนที่ดีหรือเป็นมงคลของตน

'พยายะ' หมายถึง ความเจ็บไข้ได้ป่วย ตลอดจนกระทั่งอุปสรรคที่อาจจะมาขัดขวางในกิจการหรือชีวิตของตน

'ทาสี' หมายถึง หญิงรับใช้ หรือคนรับใช้ตนที่เป็นเพศหญิง

'ทาสา' หมายถึง ชายรับใช้ หรือคนรับใช้ตนที่เป็นเพศชาย

ซึ่งผู้ใช้จะเป็นคนเลือกดูชื่อเรื่องใดก็ได้

5.2.3 การพยากรณ์

แบ่งเป็น 2 วิธี

1. การค้นหาคำทำนายจากชื่อเรื่องที่ต้องการดู และหมายเลขของมัน จากชื่อเรื่อง 21 เรื่องและมีคำทำนายสำหรับแต่ละชื่อเรื่องนั้น 7 หมายเลขอยู่ในฐานข้อมูลตาราง FORTUNE (ดูฐานข้อมู0ลได้ที่ท้ายบท) โดยค้นหาคำทำนายของเราได้จากชื่อเรื่องและหมายเลขของมันตามเมตริกซ์

2. การใช้เลขตกถึงกัน การทำนายด้วยเลขตกถึงกันแบ่งออกเป็น

<u>การทำนายเมื่อเลือกดูชื่อเรื่องในฐานวัน</u>

เราจะดูเลขในฐานทั้งสามประกอบกัน

เช่นจากตัวอย่างของเมตริกซ์ที่แสดงไปแล้วตอนต้นจะเห็นว่า

เลข 3 ของปิตา เป็นตัวเลขเดียวกับเลข 3 ของตนุและเลข 3 ของทาสา เช่นนี้ก็เรียกว่า

'ปิตาตกที่ตนุและทาสา'

เลข 4 ของมาตา เป็นตัวเลขเดียวกับเลข 4 ของกดุมภะและเลข 4 ของมรณะ เช่นนี้ก็เรียกว่า

'มาตาตกที่กดุมภะและมรณะ'

เลข 5 ของโภคา เป็นตัวเลขเดียวกับเลข 5 ของสหัชชะและเลข 5 ของศุภะ เช่นนี้ก็เรียกว่า

'โภคาตกที่สหัชชะและศุภะ'

เมื่อเราค้นหาฝอยทำนายในฐานข้อมูลตาราง FALLING จะทำให้ได้ฝอยทำนายเพิ่มจากการดูวิธีที่ 1 อีกชื่อเรื่องละ 2 ฝอยทำนายตามการตกถึงกัน

<u>การทำนายเมื่อเลือกดูชื่อเรื่องในฐานเดือน</u>

เราจะดูเลขในฐานเดือนและฐานปีประกอบกัน

เช่นจากตัวอย่างของเมตริกซ์ที่แสดงไปแล้วตอนต้นจะเห็นว่า

เลข 3 ของตนุ เป็นตัวเลขเดียวกับเลข 3 ของทาสา เช่นนี้ก็เรียกว่า

'ตนุตกที่ทาสา'

เลข 4 ของกดุมภะ เป็นตัวเลขเดียวกับเลข 4 ของมรณะ เช่นนี้ก็เรียกว่า

'กดุมภะตกที่มรณะ'

เลข 5 ของสหัชชะ เป็นตัวเลขเดียวกับเลข 5 ของศุภะ เช่นนี้ก็เรียกว่า

'สหัชชะตกที่ศุภะ'

เมื่อเราค้นหาฝอยทำนายในฐานข้อมูลตาราง FALLING จะทำให้ได้ฝอยทำนายเพิ่มจากการดูวิธีที่ 1 อีกชื่อเรื่องละ 1 ฝอยทำนายตามการตกถึงกัน

นอกจากการทำนายวิธีที่ 1 แล้ว เราต้องทราบการทำนายที่ตัวเลขตกที่เดียวกันต่อไปอีกด้วย เพราะการทำนายจากตัวเลขเดียวย่อมได้ผลไม่แน่นอน แต่ถ้ามีสองฐานก็เพื่อความแม่นยำขึ้นกว่าเดิม

5.2.4 อัลกอริทึมในการพยากรณ์

5.2.4.1 ถ้าผู้ใช้เลือกชื่อเรื่องใดชื่อเรื่องหนึ่งที่อยู่ในฐานวัน

เราจะได้เลขตัวหนึ่งออกมาจากชื่อเรื่องนั้น และนำไปค้นหาคำทำนายจากตาราง FORTUNE คอลัมน์ LOOK หมายถึงชื่อเรื่อง และ NUM หมายถึงเลขที่ตรงกับชื่อเรื่องนั้น และ FORTUNE คือคำทำนาย

ต่อจากนั้นเราก็จะใช้ตัวเลขเดียวกันนี้ ไปค้นหาชื่อเรื่องในฐานเดือน และฐานปีของเมตริกซ์ที่มีตัวเลขตรงกับเลขนี้ เมื่อได้ชื่อเรื่องที่ 2 และ 3 มาแล้วก็นำไปค้นหาคำทำนายสำหรับชื่อเรื่อง นั้นๆ เช่นเดียวกันในตาราง FORTUNE ณ จุดนี้เราจะได้คำทำนาย 3 บท ที่สอดคล้องกัน เป็นคำทำนายสำหรับชื่อเรื่องที่เราเลือก

ต่อไปเราจะค้นหาคำทำนายอีกแบบหนึ่ง คือการใช้เลขตกถึงกัน ซึ่งทำเพื่อความแม่นยำในการพยากรณ์ เมื่อเราเลือกชื่อเรื่องที่อยู่ในฐานวัน เราจะได้ชื่อเรื่องในฐานเดือน และฐานปีที่มีเลขเดียวกัน และจะนำไปค้นหาคำทำนายในตาราง FALLING : คอลัมน์ LOOK หมายถึง ชื่อเรื่องที่เราเลือกดู คอลัมน์ FALLTO หมายถึง ชื่อเรื่องที่ตกเลขเดียวกับชื่อเรื่องใน LOOK ณ จุดนี้เราจะได้คำทำนายมาอีก 2 บท รวมเป็น 5 บท

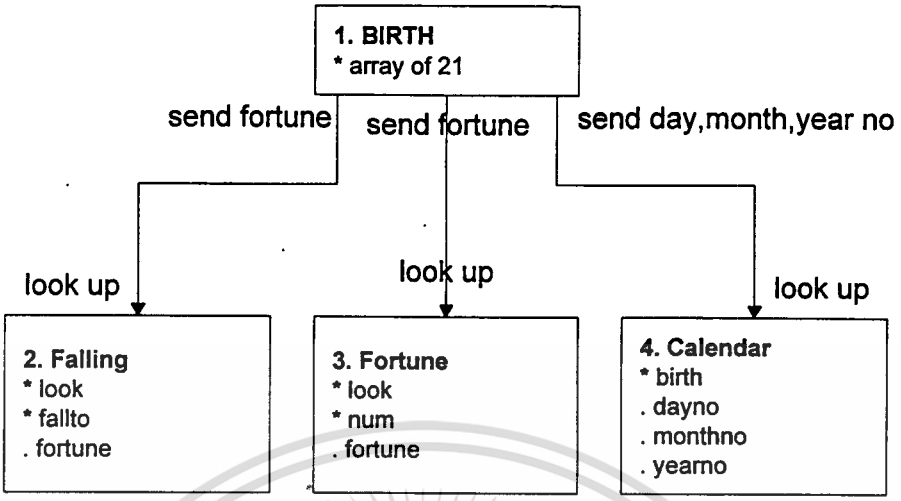
5.2.4.2 ถ้าผู้ใช้เลือกชื่อเรื่องใดชื่อเรื่องหนึ่งที่อยู่ในฐานเดือน

เราจะได้เลขตัวหนึ่งออกมา และ นำไปหาคำทำนายจากตาราง FORTUNE ต่อจากนั้นเราก็จะใช้เลขนี้ไปค้นหาชื่อเรื่องในฐานปีที่มีตัวเลขตรงกับเลขนี้ เมื่อได้มาแล้วก็จะนำไปค้นหาคำทำนายสำหรับชื่อเรื่องนั้นๆ ณ จุดนี้เราจะได้คำทำนาย 2 บทเป็นคำทำนายสำหรับชื่อเรื่องที่เราเลือก

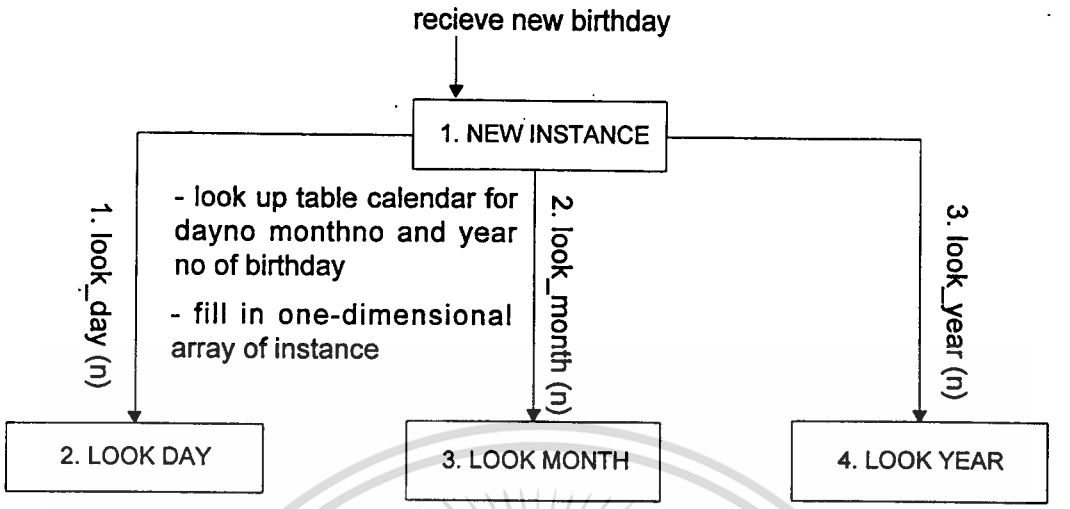
ต่อไปเราจะค้นหาคำทำนายโดยการใช้เลขตกถึงกัน เมื่อเราเลือกชื่อเรื่องที่อยู่ในฐานเดือน เราจะได้ชื่อเรื่องในฐานปีที่มีเลขเดียวกัน และจะนำไปค้นหาคำทำนายในตาราง FALLING ณ จุดนี้เราจะได้คำทำนายเพิ่มขึ้นอีก 1 บท รวมเป็น 3 บท

5.2.4.3 ถ้าผู้ใช้เลือกชื่อเรื่องใดชื่อเรื่องหนึ่งที่อยู่ในฐานปี

เราจะได้เลขตัวหนึ่งออกมา และนำไป ค้นหาคำทำนายจากตาราง FORTUNE และจะได้คำทำนายออกมา 1 บท



รูปที่ 5.14 แสดงอินฟอเมชันโมเดล

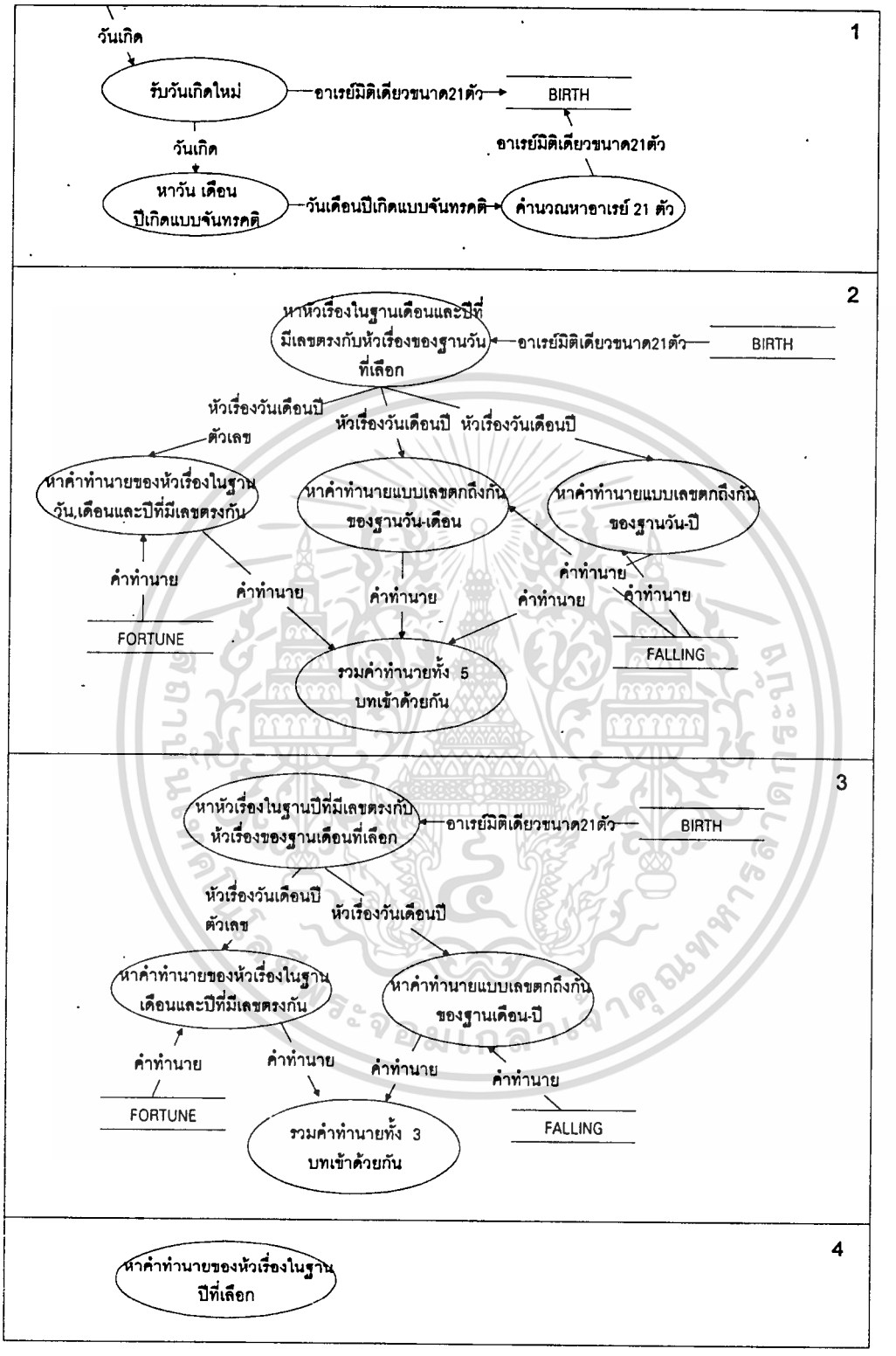


- find other two columns that match to number of the looked-day column
- retreive data from fortune table with looked-day column, month column, and year column
- retreive data from falling table with looked-day column as LOOK and month column as FALLTO
- retreive data from falling table with looked-day column as LOOK and year column as FALLTO
- combine 5 data together with in one matter

- find other column that match to number of the looked-month column
- retreive data from fortune table with looked-month column, and year column
- retreive data from falling table with looked-month column as LOOK and year column as FALLTO
- combine 3 data together with in one matter

- retreive data from fortun table with looked-yea column

รูปที่ 5.15 แสดงสเตทโมเดลของออบเจ็กต์วันเกิด



รูปที่ 5.16 แสดงโพรเซสโมเดลของทุกสเตทของออบเจ็กต์วันเกิด

DATABASE FORTUNE

CALENDAR

BIRTH	DAYNO	MONTHNO	YEARNO
date	char 1	char 2	char 2

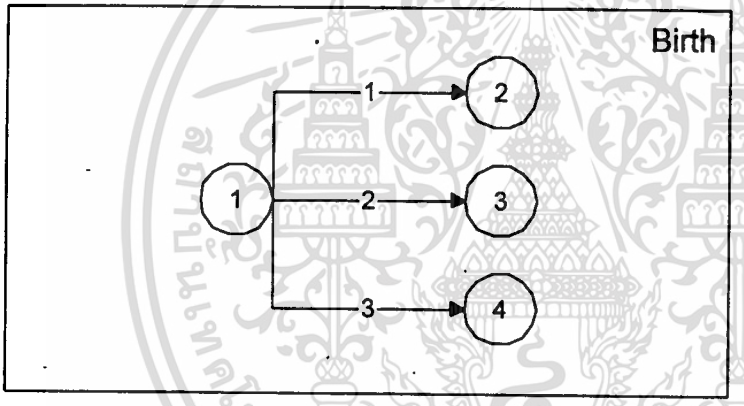
FALLING

LOOK	FALLTO	FORTUNE
float	float	long varchar

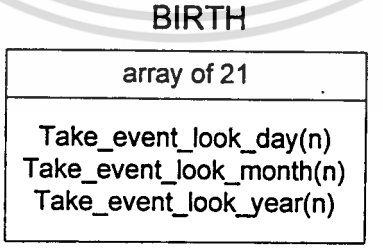
FORTUNE

LOOK	NUM	FORTUNE
float	char 1	long varchar

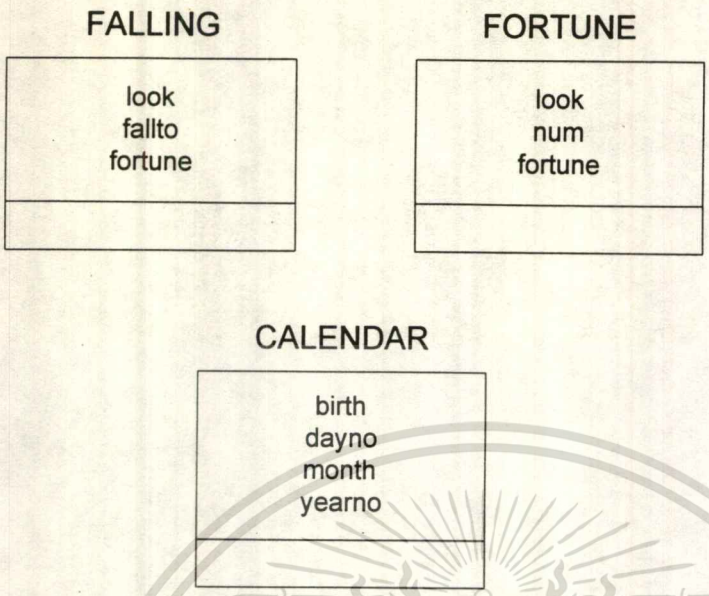
รูปที่ 5.17 แสดงฐานข้อมูลของระบบพยากรณ์ดวงชะตา



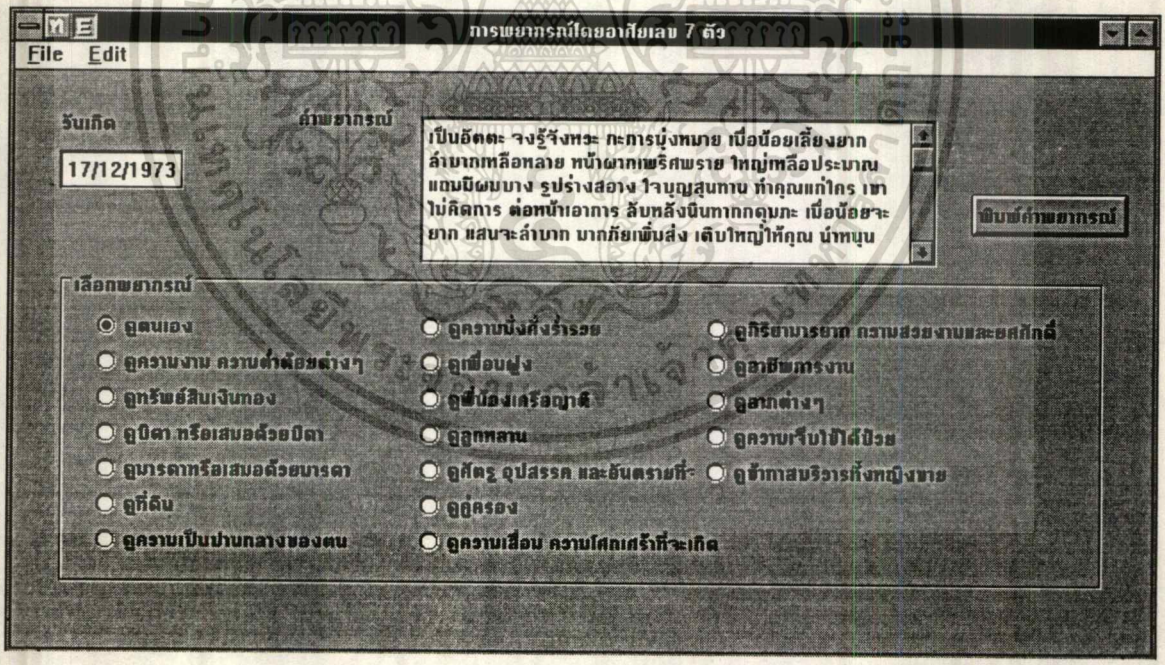
รูปที่ 5.18 แสดงไฟไนท์สเตทแมชชีนของคลาสวันเกิด



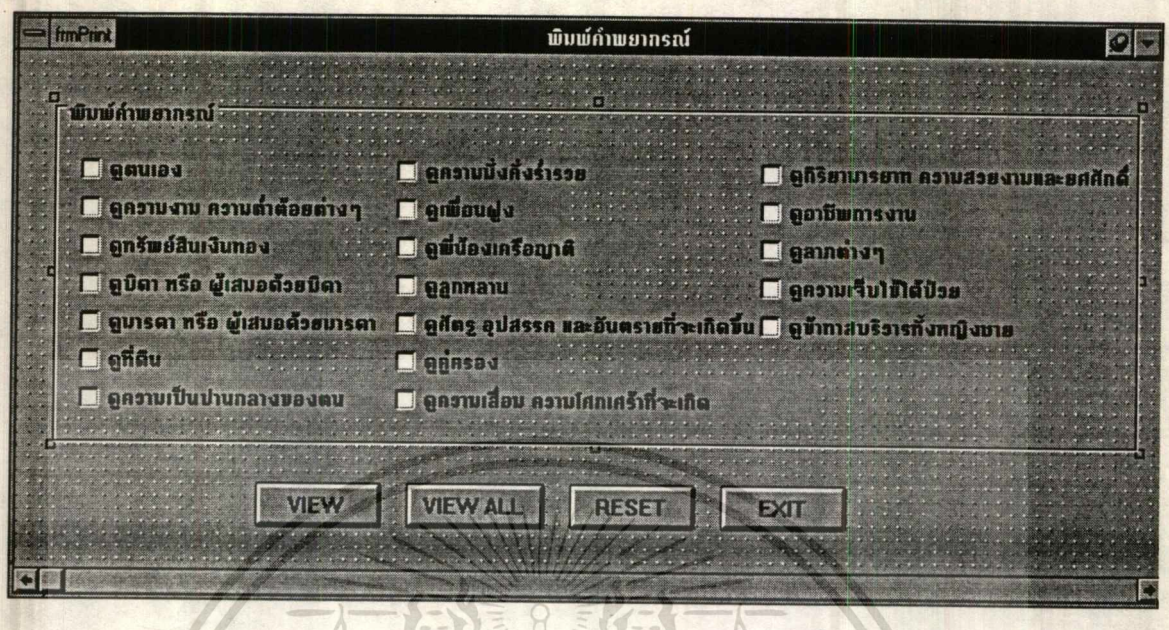
รูปที่ 5.19 แสดงแอพพลิเคชันคลาส, แอคทีฟคลาส



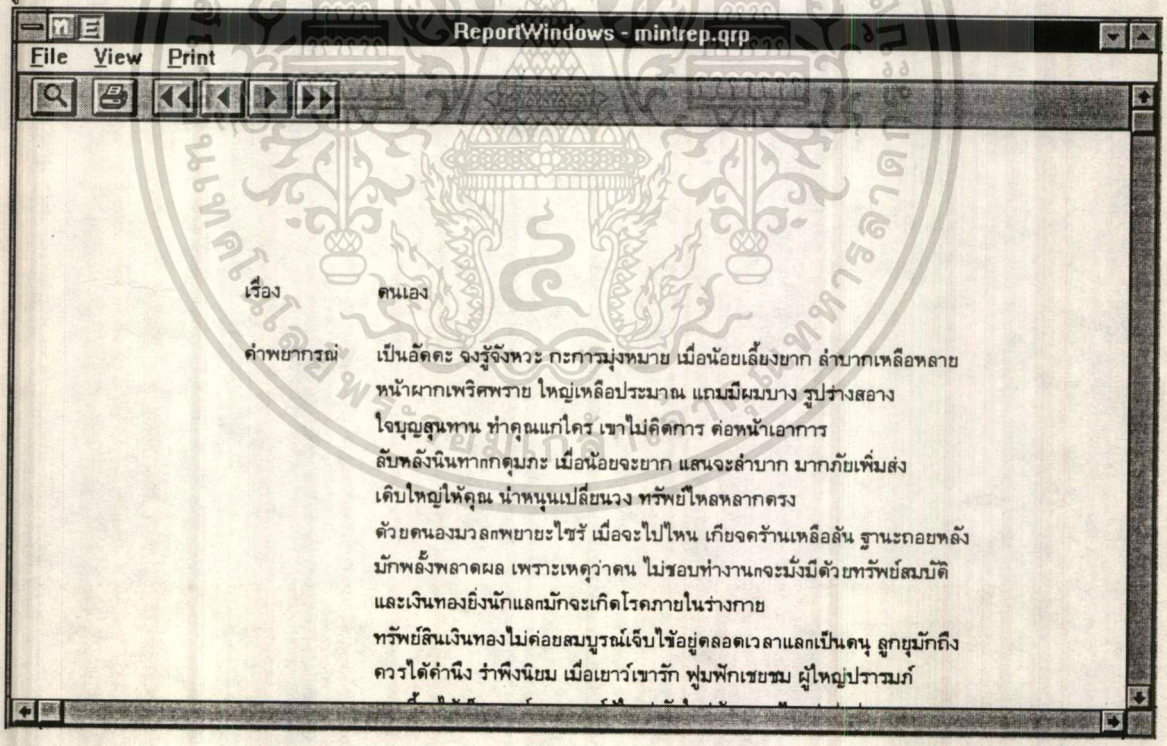
รูปที่ 5.20 แสดงแอพพลิเคชันคลาส, แพสซีฟคลาส



รูป 5.21 หน้าจอการพยากรณ์ให้ใส่วันเกิดแล้วเลือกดูการพยากรณ์จากปุ่มเรดิโอต่างๆที่เมื่อกดปุ่มหนึ่งหมายถึงเลือกและปุ่มอื่นๆจะไม่ถูกเลือกเลย แล้วดูคำพยากรณ์ที่ออกมาได้ในฟิลด์มัลติไลน์เท็กซ์ (multiline text field)
เมื่อต้องการพิมพ์หรือดูคำพยากรณ์ให้กดปุ่ม "พิมพ์คำพยากรณ์"



รูปที่ 5.22 แสดงหน้าจอการเลือกพิมพ์คำพยากรณ์ตามชื่อเรื่องโดยเลือกได้ทีละหลายๆอัน เมื่อกดปุ่มวิว หรือวิวทั้งหมดจะเข้าสู่หน้าต่างของตัวสร้างรายงาน ซึ่งจะแสดงคำพยากรณ์ทั้งหมดที่เลือก ถ้ากดปุ่มรีเซ็ท จะทำการเคลียร์เช็คบอกซ์ทั้งหมด หรือกดปุ่มเอ็กซ์ซิทจะออกจากหน้าต่างนี้กลับไปสู่หน้าต่าง รูป 5.21 เดิม



รูป 5.23 แสดงหน้าจอของตัวสร้างรายงานเพื่อพิมพ์รายงานเมื่อต้องการ หรือดูคำพยากรณ์ทั้งหมดที่เลือก

บทที่ 6
แนะนำซอฟต์แวร์

6.1 เครื่องมือที่ใช้ในการพัฒนางานประยุกต์

เครื่องมือที่ใช้ในการพัฒนาแอพพลิเคชันในครั้งนี้จะใช้เอสคิวแอลวินโดวส์และวิชวลเบสิก โดยใช้เอสคิวแอลวินโดวส์ติดต่อกับเอสคิวแอลเบส และใช้วิชวลเบสิกติดต่อกับไมโครซอฟต์แอคเซสและวินโดวส์เอ็นที เราใช้เครื่องมือทั้งสองเพื่อทดสอบว่าผลที่ได้จากการวิเคราะห์และออกแบบแบบออบเจ็กต์โอเรียนเต็ทนั้นสามารถนำไปอิมพลีเมนท์บนเครื่องมือใดๆก็ได้ และเพื่อศึกษาทูลใหม่ๆในการพัฒนาไคลเอ็นท์เซิร์ฟเวอร์แอพพลิเคชัน โดยจะแนะนำการพัฒนาแอพพลิเคชันบนเครื่องมือทั้งสองพอสังเขปดังนี้

6.1.1 เอสคิวแอลวินโดวส์ (SQLWindows)

เอสคิวแอลวินโดวส์ถูกพัฒนาขึ้นเมื่อปี 1988 เป็นระบบพัฒนาไคลเอ็นท์เซิร์ฟเวอร์แอพพลิเคชันบนไมโครซอร์ฟวินโดวส์ที่ใช้กันอย่างกว้างขวาง เนื่องจากมียูสเซอร์อินเตอร์เฟสที่ใช้งานง่าย มีประสิทธิภาพ และมี 4GL ทูลเพื่อเขียนไคลเอ็นท์เซิร์ฟเวอร์แอพพลิเคชันในเอสคิวแอลวินโดวส์เวอร์ชัน 5 เวอร์ชันใหม่ล่าสุดนี้ (2537) ยังเพิ่มฟีเจอร์ใหม่คือ ควิกออบเจ็กต์ โดยสร้างขึ้นมาเพื่อผู้ใช้จะได้ไม่ต้องเขียนโค้ดเองแม้แต่บรรทัดเดียว และอีกประการหนึ่งคือ ภาษาเอสคิวแอลวินโดวส์แอพพลิเคชัน (SAL:SQLWindows Application Language) เป็นภาษาที่ใช้งานได้คล่องตัวเหมือนการเขียนโปรแกรมภาษา C หรือ C++ ความสามารถของแอพพลิเคชันที่ผู้ใช้เขียนขึ้นจึงมีข้อจำกัดแต่เพียงความคิดสร้างสรรค์ของผู้เขียนเท่านั้น

6.1.1.1 เอสคิวแอลวินโดวส์เวอร์ชัน 5 และ ชุดผลิตภัณฑ์ไคลเอ็นท์เซิร์ฟเวอร์ของกุพต้า (the Gupta Client/Server Suite)

บริษัทกุพต้าพัฒนาชุดของผลิตภัณฑ์ต่างๆบนไคลเอ็นท์เซิร์ฟเวอร์เรียกว่า ชุดผลิตภัณฑ์ไคลเอ็นท์เซิร์ฟเวอร์ของกุพต้า ประกอบด้วย

- **เอสคิวแอลวินโดวส์**

  เป็นระบบออบเจ็กต์โอเรียนเต็ทสำหรับพัฒนาไคลเอ็นท์เซิร์ฟเวอร์แอพพลิเคชันบนไมโครซอฟต์วินโดว์และ แพลตฟอร์ม GUI อื่นๆ โครงสร้างควิกออบเจ็กต์ และ ควิกฟอร์ม (QuickForms) ทำให้เอสคิวแอลวินโดวส์ ใช้งานง่าย ภาษายุคที่ 4 ที่ทันสมัย, คอมไพเลอร์ที่เปลี่ยนภาษายุคที่ 4 เป็นภาษา C และการเขียนโปรแกรมแบบออบเจ็กต์โอเรียนเต็ทที่แท้จริง ทั้งหมดนี้จะทำให้เราสามารถพัฒนาไคลเอ็นท์เซิร์ฟเวอร์แอพพลิเคชันได้อย่างง่ายดาย

- **ตัวสร้างรายงาน (Quest Reporter) และ เควส (Quest)**

ตัวสร้างรายงานมีการติดต่อกับข้อมูลที่ใช้ในการกำหนดรายงานด้วยรูปแบบกราฟฟิก, มีคิวรี่ (query) โดยไม่ต้องมีความรู้เรื่องภาษาเอสคิวแอลก็ได้ และมีเครื่องมือทำรายงานที่ใช้งานง่ายโดยสัญชาติญาณ ส่วนเควสก็เช่นเดียวกัน และยังเตรียมความสามารถอื่นๆให้ผู้ใช้ระดับสูงใช้ เช่น การสร้างฟอร์ม การสร้างและจัดการตารางเอสคิวแอล และทำงานกับนิยามฐานข้อมูล

- **เอสคิวแอลเบส**

เป็นมัลติยูสเซอร์รีเลชันนัลดาต้าเบสเซิร์ฟเวอร์ ที่พัฒนาขึ้นบนหลายแพล็ตฟอร์ม ได้แก่ ดอส, ไมโครซอร์ฟวินโดวส์, โอเอสทู, ยูนิกซ์ และ เน็ตแวร์ มีเอสคิวแอลทอล์คเป็นอินเตอร์แอคทีฟยูสเซอร์อินเตอร์เฟส สำหรับเอสคิวแอลเบส ที่อนุญาตให้เราใส่คำสั่งภาษาเอสคิวแอล หรือ คำสั่งของเอสคิวแอลทอล์คเอง เพื่อจัดการกับฐานข้อมูล เรามักจะเรียก เอสคิวแอลทอล์คสำหรับวินโดวส์ ว่า วินทอล์ค และ วินทอล์คสามารถใช้กับฐานข้อมูลอื่นๆที่ไม่ใช่เอสคิวแอลเบสได้ด้วย

- **เอสคิวแอลคอนโซล** (SQLConsole)

เพื่อการจัดการฐานข้อมูลและมีเครื่องมือตรวจจับติดตามสำหรับ เอสคิวแอลเบสเซิร์ฟเวอร์

- **เอสคิวแอลเน็ตเวิร์ค** (SQLNetwork)

ชุดของผลิตภัณฑ์ที่เตรียมการเรื่องการเชื่อมต่อกับ IBM DB2, Oracle, Sybase และ Microsoft SQL Server, Ingres, Informix, OS/2 Database Manager, IBM AS/400, Cincom Supra, HP Allbase/SQL และ ฐานข้อมูลภาษาเอสคิวแอลอื่นๆ ผ่านทาง ODBC

ใน เอสคิวแอลวินโดวส์เวอร์ชัน 5 ผู้เขียนโปรแกรมสามารถเขียน GUI แอพพลิเคชัน ด้วยวิธีชี้แล้วคลิก (point-and-click) ของ ไมโครซอร์ฟวินโดวส์

เอสคิวแอลวินโดวส์ มีใช้ได้ตั้งแต่ผู้พัฒนาแอพพลิเคชันที่ใช้บนเครื่องเดี่ยว ไปจนถึงการแปลงแอพพลิเคชันไปเป็นทีมของผู้พัฒนาแอพพลิเคชันแบบมัลติยูสเซอร์

นอกเหนือจากที่กล่าวมาแล้ว ชุดผลิตภัณฑ์ไคลเอ็นท์เซิร์ฟเวอร์ของกุพต้ายังประกอบด้วยซอร์ฟแวร์เชื่อมต่อระบบเอ็นเตอร์ไพรส์ (enterprise connectivity software) ซึ่งทำหน้าที่นำเอาข้อมูลจากส่วนกลางมาประมวลผล และ กลยุทธ์การรวมทูลต่างๆสำหรับระบบเอ็นเตอร์ไพรส์ (TIE:Tools Integration for the Enterprise Strategy) ของกุพต้า ทำให้การรวมเทคโนโลยีที่ใช้ในการพัฒนาแอพพลิเคชันแต่ละอันที่ไม่เกี่ยวข้องกันเป็นแอพพลิเคชันที่สร้างด้วยผลิตภัณฑ์ของกุพต้าได้ง่ายขึ้น เช่น เคสทูล(CASE tools) และ กรุพแวร์ (groupware)

6.1.1.2 ฟีเจอร์ที่สำคัญของ เอสคิวแอลวินโดวส์

ความง่ายในการใช้งานของควิกออบเจ็กต์ เราสามารถใช้ ควิกออบเจ็กต์ ซึ่งเป็นส่วนที่สร้างไว้ก่อนแล้ว สร้างแอพพลิเคชันโดยไม่ต้องรู้เรื่องเอสคิวแอล และไม่ต้องเขียนโค้ดเลยแม้แต่บรรทัดเดียว และ

เราไม่ต้องเป็นผู้เชี่ยวชาญในการเขียนโปรแกรมไคลเอ็นท์เซิร์ฟเวอร์ หรือออบเจ็กต์โอเรียนเต็ทก็สามารถใช้ ควิกออบเจ็กต์ นี้ได้ และเราสามารถสร้าง หรือ แก้ไข ควิกออบเจ็กต์ เองได้โดยใช้ SAL (SQLWindows Application Language)

หน้าต่างการออกแบบ (Design Window) เป็นสิ่งแวดล้อมแบบวิชวลในการเขียนโปรแกรม (visual programming environment), มีทูลพาเหรด สำหรับออกแบบหน้าจอแอพพลิเคชัน เราสามารถชี้ไปที่ออบเจ็กต์ที่ต้องการแล้วคลิกเพื่อแปะออบเจ็กต์นั้น เช่น ปุ่มกด(push button) บนฟอร์ม แล้วใช้ความสามารถลากแล้วปล่อย (drag-and-drop) เพื่อย้ายหรือเปลี่ยนขนาดมัน

หน้าต่างการออกแบบยังมีคัสตอมไมเซอร์ (customizer) สำหรับเปลี่ยนแอททริบิวท์ (attributes) ของออบเจ็กต์ เช่น เราสามารถเปลี่ยนคุณสมบัติของฟิลด์ข้อมูล (data field) จากชิดขวาเป็นชิดซ้าย

แอพพลิเคชันเอาท์ลายน์(แอพพลิเคชัน Outline) คือ เอาท์ลายน์ของแอพพลิเคชันทั้งหมดที่สามารถอ่านได้และซ่อนเป็นส่วนๆได้ เมื่อเราเพิ่มหรือเปลี่ยนแปลงแอพพลิเคชันออบเจ็กต์ใน Design Window, แอพพลิเคชันเอาท์ลายน์ จะถูกอัพเดตโดยอัตโนมัติ และสามารถให้เราก็อปปี้, คัท และเพสท์ส่วนของโค้ด หรือ ออบเจ็กต์แบบวิชวล เช่น ฟอร์ม หรือ ไดอะล็อกบ็อกซ์

ภาษาเอสคิวแอลวินโดวส์แอพพลิเคชัน (SAL) เมื่อเขียนแอพพลิเคชันที่ซับซ้อน, เราต้องมีการเขียนโค้ดเองด้วย SAL เป็นภาษายุคที่ 4 SAL เป็นออบเจ็กต์โอเรียนเต็ท แต่เราไม่ต้องใช้ออบเจ็กต์โอเรียนเต็ทฟีเจอร์เลยเนื่องจาก SAL เตรียมฟังก์ชันไว้ให้เรียกใช้มากมาย และเราสามารถสร้างฟังก์ชันขึ้นใช้เองได้ รวมทั้งยังสามารถเก็บลงในไลบรารีไว้เรียกใช้ต่อไปได้ นอกจากนี้เราสามารถเรียกใช้ ไมโครซอร์ฟวินโดวส์ API ฟังก์ชัน หรือฟังก์ชันจาก DLL (Dynamic Link Library) ซึ่งสร้างไว้แล้วก็ได้

ออบเจ็กต์โอเรียนเต็ทที่ง่ายๆ เราสามารถใช้เทคนิคของ OOP เช่น การสืบทอดเพื่อขยายความสามารถของ ควิกออบเจ็กต์

เอาท์ลายน์ออบชันบาร์ (Outline option bar) มีประโยชน์มากสำหรับผู้เขียนโปรแกรม, ออพชันบาร์แสดงทางเลือกต่างๆของคำสั่งภาษา SAL, ฟังก์ชัน, ตัวแปร และค่าคงที่ โดยขึ้นกับว่าเราอยู่ที่ส่วนไหนของโปรแกรม เมื่อต้องการทางเลือกใดก็ชี้ไป แล้วคลิกเมาส์ สเตทเมนท์ที่เลือกนั้นก็จะถูกเพิ่มเข้ามาในโค้ดของเราอย่างอัตโนมัติ สำหรับการใช้ฟังก์ชัน ทั้งที่มีให้ใช้อยู่แล้วและที่สร้างขึ้นเอง ออพชันบาร์ จะแสดงชนิดของข้อมูลของทุกพารามิเตอร์เพื่อช่วยเราตัดสินใจ

ดีบักเกอร์ (Debugger) สามารถตั้งเบรคพอยท์ได้หลายแห่ง, ทำสเต็ปได้, ดูค่าของตัวแปร และทำโค้ดอนิเมชันได้ (code animation)

ตัวออกแบบรายงาน (ReportWindows) ตัวออกแบบรายงานที่เป็นกราฟฟิก สามารถสร้างรายงานแบบทาบุล่า (tabular), ครอสแทบ (cross-tab) และ คอนโทรลเบรค (control-break) และรายงานที่สร้างด้วยตัวสร้างรายงานและเควสจะคอมแพททิเบิลกัน

เอสคิวแอลวินโดวส์คอมไพเลอร์ (SQLWindows Compiler) สร้างโค้ดภาษา C สำหรับบางส่วนของแอพพลิเคชัน

การเขียนโปรแกรมเป็นทีม (Team Programming) เมื่อเรากำลังพัฒนาแอพพลิเคชันเป็นทีม ซึ่งผู้เขียนโปรแกรมหลายๆคนทำงานบนแต่ละส่วนของโค้ด เราน่าจะใช้ทีมวินโดวส์ (TeamWindows) เป็นตัวจัดการแต่ละส่วนของแอพพลิเคชันโค้ด ซึ่งเรียกว่า โมดูล (module), โมดูล อาจหมายถึงอะไรก็ได้ เช่น เอสคิวแอลวินโดวส์แอพพลิเคชัน, เอสคิวแอลวินโดวส์ไลบรารี, ไฟล์บิตแมพ, หรือ เอกสารของเวิร์ด นอกจากนี้ ทีมวินโดวส์ยังเตรียมที่เก็บ (repository) สำหรับซอร์สโค้ด และการควบคุมเวอร์ชัน, การเช็คอินเช็คเอาท์ เพื่อให้สมาชิกในทีมสามารถใช้ และ แก้ไขโมดูลร่วมกันได้

ทีมวินโดวส์ ยังทำหน้าที่บำรุงรักษา ดาต้าดิกส่วนกลาง ได้แก่ ชื่อตารางและชื่อคอลัมน์ต่างๆ และ ความสัมพันธ์ระหว่าง ไพรมารีคีย์ (primary key) กับ ฟอเรนคีย์(foreign key)

6.1.1.3 เริ่มต้นการเขียน เอสคิวแอลวินโดวส์แอพพลิเคชัน

จุดประสงค์ของส่วนนี้เพื่อทำให้ผู้อ่านคุ้นเคยกับเอสคิวแอลวินโดวส์ โดยกล่าวถึงสิ่งต่าง ๆดังนี้

- แอพพลิเคชันอะไรบ้างที่สามารถพัฒนาด้วยเอสคิวแอลวินโดวส์ได้
- องค์ประกอบอย่างง่ายๆของเอสคิวแอลวินโดวส์และการใช้งาน เช่น ทูลพาเหรต (tool palette), คัสตอมไมซ์เซอร์ (customizer), แอพพลิเคชันเอาท์ลายน์ (application outline), เอาท์ลายน์ออบชั่น (outline options bar/box), ดีบักเกอร์ (debugger)
- วิธีการสร้างแอพพลิเคชันโดยใช้ควิกฟอร์ม (QuickForm)
- วิธีการเขียนแอพพลิเคชันแบบฟังก์ชันนอลโดยใช้ควิกออบเจ็กต์ (QuickObjects) และไม่ต้องเขียนโค้ดเองเลยแม้แต่บรรทัดเดียว การใช้แอพพลิเคชันแบบนี้ ผู้ใช้สามารถเลือกดูเรคคอร์ดต่างๆ, อินเสิร์ทเรคคอร์ดใหม่, ลบหรือแก้ไขเรคคอร์ดเดิม และเมื่อทำการเปลี่ยนแปลงข้อมูลในตารางตามต้องการแล้ว ก็สามารถแอพพลายการเปลี่ยนแปลงนั้นลงฐานข้อมูล หรือละเลยการเปลี่ยนแปลงไปได้

เอสคิวแอลวินโดวส์ เตรียมชุดของฟังก์ชันและแมสเสจไว้เกือบครบความต้องการของผู้เขียนโปรแกรม และเรายังสามารถใช้ฟังก์ชันจาก DLL (Dynamic Link Library) เช่น ของไมโครซอร์ฟวินโดวส์ และประมวลผลไมโครซอร์ฟวินโดวส์แมสเซจ และใช้คัสตอมคอนโทรล (custom controls) และ วิชวลเบสิก(VBX)คอนโทรล ในเอสคิวแอลวินโดวส์แอพพลิเคชัน

เอสคิวแอลวินโดวส์ ใช้เขียนแอพพลิเคชันที่ช่วยในการตัดสินใจ เช่น Mission critical Application **ตัวอย่างของแอพพลิเคชันทางธุรกิจที่ใช้เอสคิวแอลวินโดวส์ได้แก่**

ระบบการรับออเดอร์ (Order Entry System) พนักงานขายสามารถใช้ระบบนี้เพื่อรับออเดอร์จากลูกค้าและสร้างใบอินวอยส์ให้

ระบบติดตามดูแลคนไข้ (Patient treatment tracking system) แพทย์สามารถใส่การวิเคราะห์และคำแนะนำในการรักษา เพื่อให้ผู้ขายยาสามารถใช้ระบบเดียวกันในการจ่ายยา

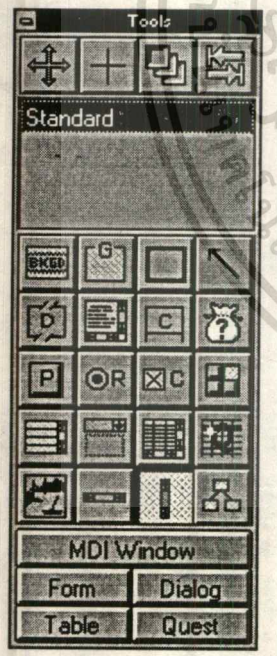
ระบบบัญชี (Account โมดูลs) เช่น ระบบบัญชีลูกหนี้ หรือเจ้าหนี้, บัญชีแยกประเภทและอื่นๆ

ระบบจัดการทรัพยากรบุคคลและเงินเดือน โดยอาจจะมีรูปของพนักงานแต่ละคนด้วย

ระบบติดตามการทำงานของพนักงานขาย และการขาย

เมื่อมีจำนวนผู้ใช้ระบบเพิ่มมากขึ้น เราสามารถอัพเกรดเซิร์ฟเวอร์โดยใช้ CPU ที่มีพลังมากขึ้น หรือใช้แพลตฟอร์มที่มีประสิทธิภาพสูงขึ้น (เช่น ใช้ เอสคิวแอลเบส NLM แทนที่เอสคิวแอลเบสสำหรับดอส) หรือใช้ฐานข้อมูลอื่นๆ (เช่น ใช้ DB2 วิ่งบนไอบีเอ็มเมนเฟรมคอมพิวเตอร์แทนที่เอสคิวแอลเบส NLM ที่วิ่งบนเครื่องพีซีที่มีซีพียูเพนเทียม) การอัพเกรดเซิร์ฟเวอร์นี้จะไม่กระทบกระเทือนแอพพลิเคชันที่รันบนเครื่องไคลเอ็นท์ คือไม่ต้องมีการเปลี่ยนแปลงโค้ด หรือเปลี่ยนแปลงน้อยมาก นี่เป็นข้อดีมากอย่างหนึ่งของโครงสร้างไคลเอ็นท์เซิร์ฟเวอร์

ทูลพาเหรด



รูป 6.1 แสดง ทูลพาเหรด

คัสตอมไมเซอร์

แต่ละออบเจ็กต์จะมีแอททริบิวต์ซึ่งกำหนดคุณลักษณะของออบเจ็กต์ที่ปรากฏ เช่น สี, รูปแบบของตัวอักษร, และที่ตั้งของออบเจ็กต์ ออบเจ็กต์แอททริบิวต์สามารถกำหนดผ่าน คัสตอมไมเซอร์ ของออบเจ็กต์

ควิกออบเจ็กต์

ควิกออบเจ็กต์ คือกรุพของพรีดีฟายด์ออบเจ็กต์ (pre-defined objects) ออบเจ็กต์เหล่านี้ทำให้เราสามารถพัฒนาแอพพลิเคชันได้อย่างรวดเร็วโดยมีชุดของวิชวล (visual) และ นอนวิชวลออบเจ็กต์ (non-visual objects) เตรียมไว้แล้วดังนี้

- ดาต้าซอร์ส (data sources)

  ดาต้าซอร์สควิกออบเจ็กต์ ทำหน้าที่เตรียมการเชื่อมต่อกับข้อมูลจากแหล่งข้อมูล เช่น ข้อมูลของเอสคิวแอล (SQL data), ข้อมูลของจดหมายอิเล็คทรอนิกส์ (email data) หรือ ข้อมูลของโลตัสโน้ต (Lotus Notes data), ส่วนใหญ่เราจะทำให้ ดาต้าซอร์สควิกออบเจ็กต์มองไม่เห็นหลังจากกำหนดมันขึ้นมาแล้ว และมันจะทำการแอคเซส และอัพเดตข้อมูลอยู่ภายหลัง

- วิชวลไลซ์เซอร์ (visualizers)

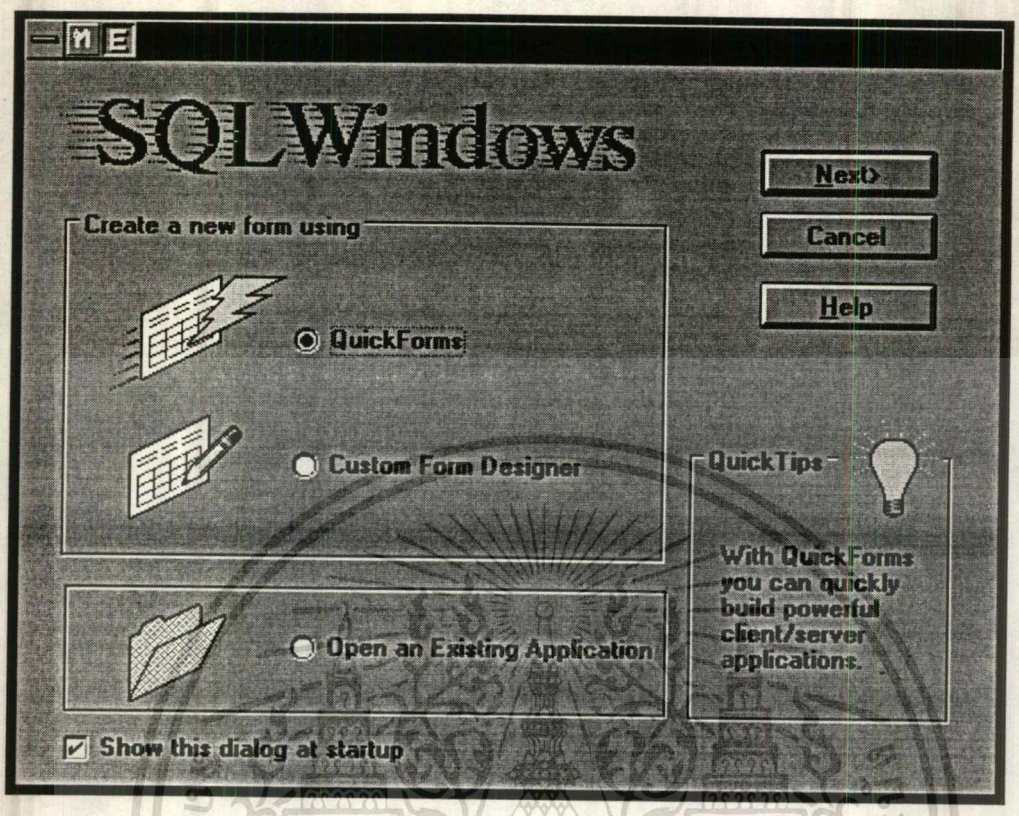
  ใช้แสดงข้อมูลเช่น ปุ่มเรดิโอ (radio buttons) หรือ ลิสต์บ็อกซ์ (list boxes) เช่นถ้าเราต้องการแปะวิธีการจ่ายเงินลงบนฟอร์ม และมีวิธีอยู่ 2 ทาง ได้แก่ จ่ายเงินสด, จ่ายด้วยเครดิตการ์ด เรารู้แน่ว่าต้องใช้กรุพของปุ่มเรดิโอ 2 อัน และในกรณีเช่นนี้เราสามารถใช้ ควิกออบเจ็กต์ ได้ทันที

- คอมมานเดอร์ (commanders)

ใช้จัดการกับข้อมูลเช่น การดูเรคคอร์ดก่อนหน้านี้, การดึงข้อมูล, การอินเสิร์ทเรคคอร์ดใหม่, การลบเรคคอร์ดที่มีอยู่แล้ว, การยอมรับความเปลี่ยนแปลง (applying changes), หรือ การละเลยความเปลี่ยนแปลง (discarding changes) คอมมานเดอร์สั่งให้ ดาต้าซอร์สทำหน้าที่เหล่านี้

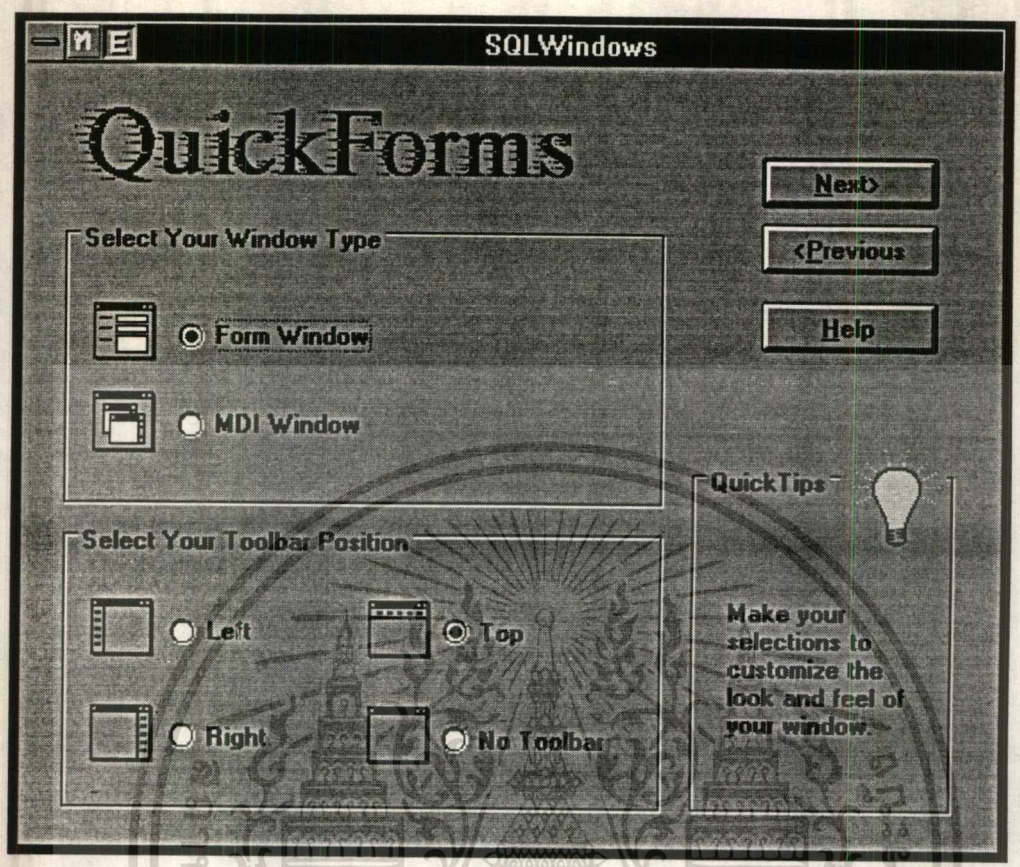
6.1.1.4 การสร้างเอสคิวแอลวินโดวส์แอพพลิเคชัน

เมื่อเปิดเอสคิวแอลวินโดวส์ มันจะแสดงไดอะล็อกบ็อกซ์ดังรูป 6.2 ณ จุดนี้เราสามารถสร้างแอพพลิเคชันใหม่หรือ เปิดแอพพลิเคชันเก่าก็ได้



รูปที่ 6.2 แสดงหน้าจอเมื่อเปิดเอสคิวแอลวินโดวส์

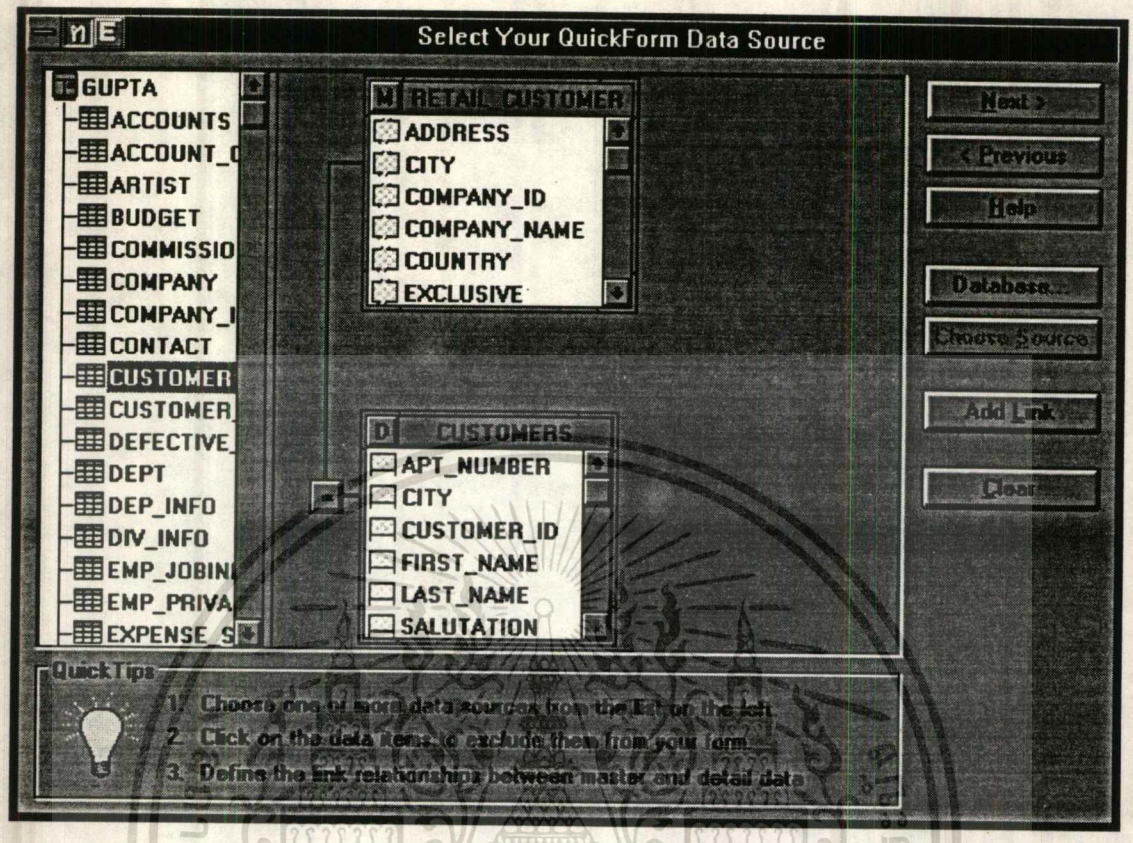
- **ถ้าเราเลือกสร้างควิกฟอร์ม,** เอสคิวแอลวินโดวส์จะนำทางให้เราไปตามสเต็ปต่างๆ ในการสร้างแอพพลิเคชันโดยการใช้ควิกออบเจ็กต์ เมื่อกดปุ่ม Next> ซึ่งต่อไปจะขึ้นหน้าจอที่ถามความต้องการว่าจะสร้างฟอร์มหลักเป็นอย่างไร ดังรูปที่ 6.3 แล้วเอสคิวแอลวินโดวส์ จะถามชื่อฐานข้อมูลที่จะใช้งาน, ชื่อผู้ใช้, และรหัสผ่าน ถ้าเราต้องการทำงานกับฐานข้อมูลมากกว่า 1 อัน เราก็สามารถเพิ่มฐานข้อมูลอื่นๆให้แอพพลิเคชันของเราได้



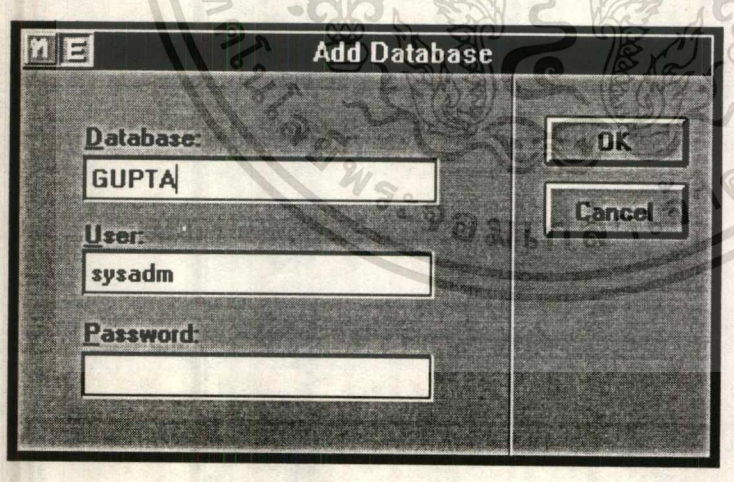
รูปที่ 6.3 แสดงหน้าจอถัดมาในการสร้างควิกฟอร์ม

การกำหนดดาต้าซอร์ส

เลือกตารางและคอลัมน์ที่ต้องการใช้ดังรูปที่ 6.4 หรือเราสามารถเลือกฐานข้อมูลอื่นๆ ที่ต้องการได้โดยคลิกปุ่ม Database... จะปรากฏไดอะล็อกบ็อกซ์ให้เพิ่มฐานข้อมูลอื่นดังรูปที่ 6.5



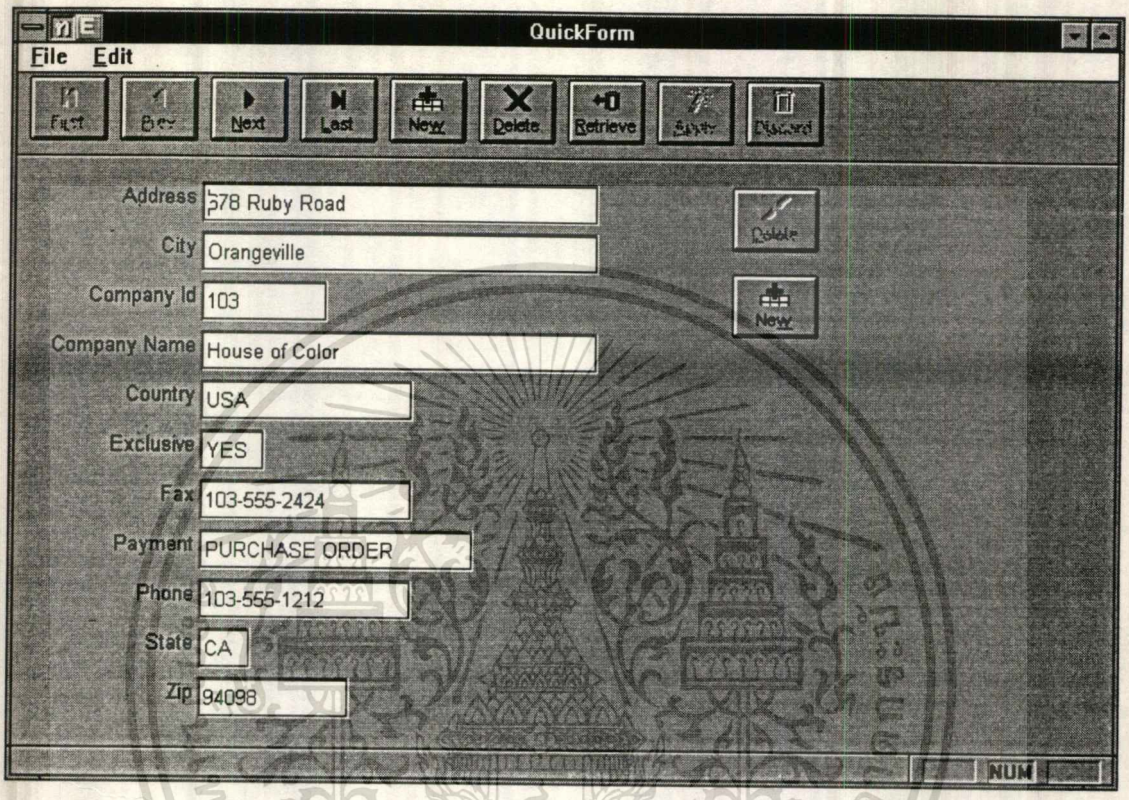
รูปที่ 6.4 การเลือกตารางและคอลัมภ์ที่ต้องการใช้ในการสร้างควิกฟอร์ม ซึ่งสามารถทำการจอยน์ (join) ได้



รูปที่ 6.5 การเพิ่มฐานข้อมูลอื่นๆเป็นดาต้าซอร์ส

หลังจากกำหนดดาต้าซอร์สแล้ว เอสคิวแอลวินโดวส์ จะสร้าง ฟอร์มวินโดวส์ และแปะฟิลด์ข้อมูล(คือ วิชวลไลเซอร์) สำหรับแต่ละคอลัมน์ของตารางพร้อมกับเลเบลของมัน และ เอสคิวแอลวิน

โดวส์ ยังแปะปุ่มกด (คือ คอมมานเดอร์) สำหรับการเลือกดู (browsing), การอินเสิร์ท (inserting), การลบ (deleting) และอื่นๆ แล้วจะได้ ฟอร์มวินโดวส์ ดังรูป 6.6

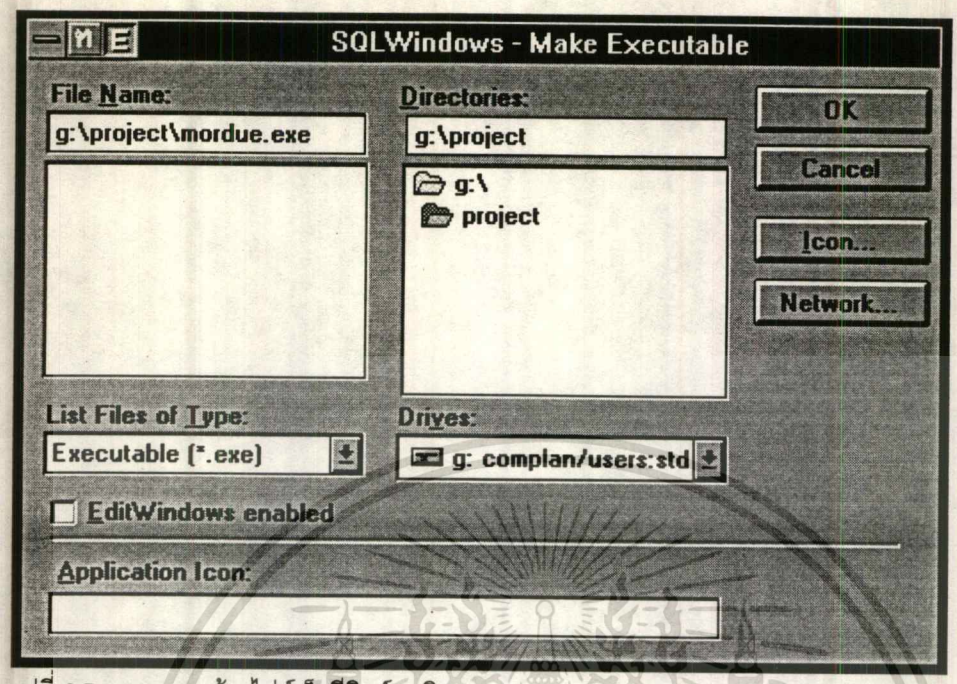


รูป 6.6 แสดง ฟอร์มวินโดวส์ หลัก

หลังจากนั้นเราสามารถเปลี่ยนแปลงแก้ไขแอททริบิวต์ของออบเจ็กต์ต่างๆใน ฟอร์มวินโดวส์ ได้ รวมทั้งตัวฟอร์มวินโดวส์ ด้วย โดยใช้ คัสตอมไมเซอร์

การรันแอพพลิเคชัน

ขณะที่สร้างแอพพลิเคชัน เราอยู่ในโหมดการออกแบบ มี 2 ทางในการรันแอพพลิเคชัน คือ ทางแรก สร้างไฟล์เอ็กซีคิวท์เทเบิล (executable) และรันมันจากภายนอกเอสคิวแอลวินโดวส์ ดังแสดงในรูปที่ 6.7 เราจะใช้ทางเลือกนี้เมื่อแอพพลิเคชันเสร็จสมบูรณ์แล้ว และพร้อมที่จะแจกจ่ายออกไป เมื่อเรายังอยู่ในขั้นตอนของการพัฒนา และแก้บักของแอพพลิเคชัน เราอาจใช้ทางเลือกที่สอง คือ รันแอพพลิเคชันจากในเอสคิวแอลวินโดวส์ เองโดยเลือกโหมดผู้ใช้ ด้วยการเลือกเมนู Run, User Mode



รูปที่ 6.7 แสดงการสร้างไฟล์เอ็กซีคิวท์เทเบิล

ก่อนรันแอพพลิเคชัน ถ้ามีการเปลี่ยนแปลงแอพพลิเคชันตั้งแต่ครั้งสุดท้ายที่เราเซฟไว้ เอสคิวแอลวินโดวส์จะถามก่อนว่า ต้องการเซฟก่อนหรือไม่ หลังจากที่เซฟไฟล์แล้ว จึงทำการคอมไพล์ และรันแอพพลิเคชัน ถ้าต้องการกลับเข้าสู่โหมดการออกแบบอีกครั้งให้เลือกเมนู File, Exit หรือ เลือก Run, User Mode อีกครั้ง

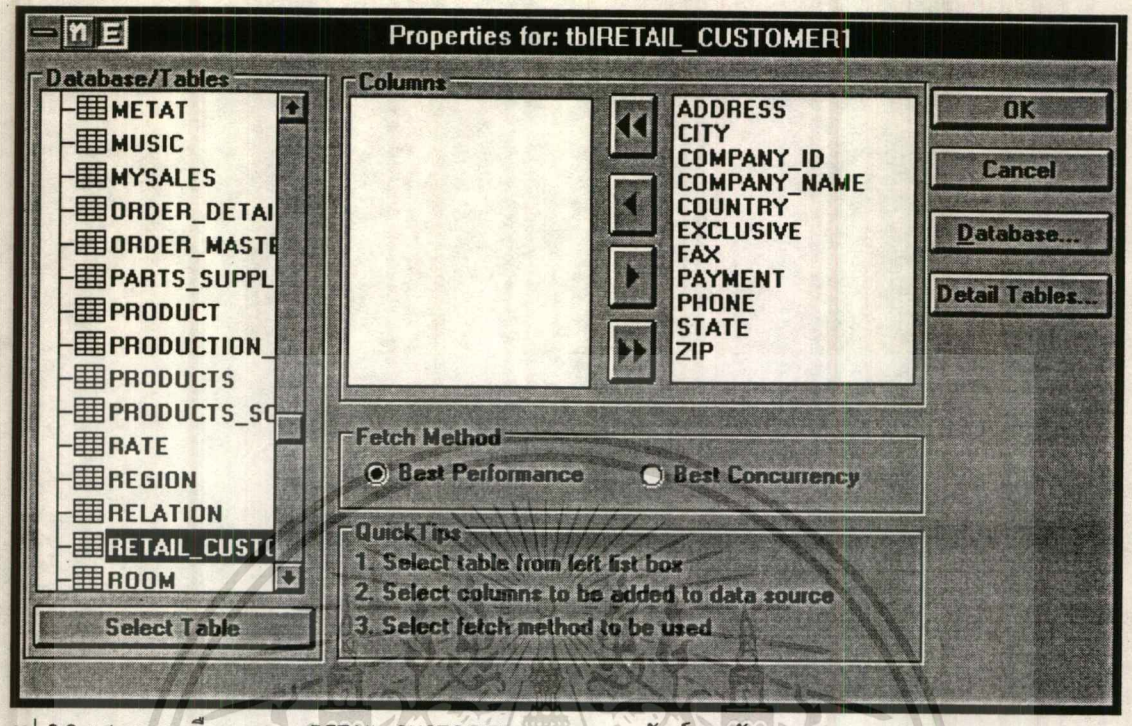
◆ **การใช้ ควิกออบเจ็กต์**

เมื่อเราเปิด เอสคิวแอลวินโดวส์ ขึ้นมาในตอนแรก ถ้าเราเลือก Custom Form Designer มันจะสร้าง ฟอร์มวินโดวส์ ว่างๆขึ้นมาให้อันใหม่

การแปะดาต้าซอร์สบน ฟอร์มวินโดวส์

แอพพลิเคชันนี้จะประกอบด้วยตาราง RETAIL_CUSTOMER เมื่อต้องการแปะดาต้าซอร์สนี้บนฟอร์มวินโดวส์ เราต้องเลือกทูลรูปตารางจากทูลพาเหรด เมื่อเรากดทูลนั้น ทูลพาเหรดจะแสดงคลาสต่างๆที่เกี่ยวข้องสำหรับออบเจ็กต์นี้ในลิสต์บ็อกซ์ (List Box) ที่ส่วนบนของทูลพาเหรด

หลังจากเลือกแล้ว เคอร์เซอร์จะเปลี่ยนเป็นรูปทูลที่เราเลือก แล้วเราสามารถแปะอินแสตนท์ (instance) นี้ที่ใดก็ได้บน ฟอร์มวินโดวส์ โดยคลิกเมาส์ เสร็จแล้วจะเกิดไดอะล็อกบอกซ์ให้กำหนดพร็อบเพอตี้ของตารางที่เราเลือก ดังรูป 6.8



รูป 6.8 แสดงการเลือกตาราง RETAIL_CUSTOMER และทุกๆคอลัมน์ของมัน

การทำให้มองไม่เห็นดาต้าซอร์ส

ส่วนใหญ่เราต้องการซ่อนดาต้าซอร์ส เพื่อให้ดาต้าซอร์สแอคเซสและจัดการข้อมูลอยู่ข้างหลัง แล้วเราก็กำหนดวิชวลไลเซอร์ขึ้นมาเพื่อแสดงข้อมูล และคอมมานเดอร์เพื่อส่งคำสั่งไปยังดาต้าซอร์สให้ทำโอเปอเรชันที่ต้องการ เราสามารถซ่อนดาต้าซอร์ส (table window) โดยใช้ คัสตอมไมเซอร์ เปลี่ยน Visible เป็น No

การแปะวิชวลไลเซอร์บน ฟอร์มวินโดวส์

จาก ทูลพาเหรด เลือกฟิลด์ข้อมูล, เลือก cQuickField จากลิสต์ของคลาสที่เห็น จากนั้นเราจะสามารถเลือกชื่อตารางได้จาก คอมโบบ็อกซ์ (combo box) ของชื่อตาราง และเลือกคอลัมน์ที่จะให้ฟิลด์นี้แสดงได้ที่ลิสต์ของชื่อคอลัมน์ จากนั้นเราก็ลากอินแสตนท์ที่เราเลือกไปแปะบน ฟอร์มวินโดวส์

สำหรับ คอลัมน์ PAYMENT เราเลือกแสดงข้อมูลเป็นกรุพของปุ่มเรดิโอ โดยเลือกปุ่มเรดิโอ แล้วเลือก cQuickRadioGroup แล้วเลือกแสดงคอลัมน์ PAYMENT

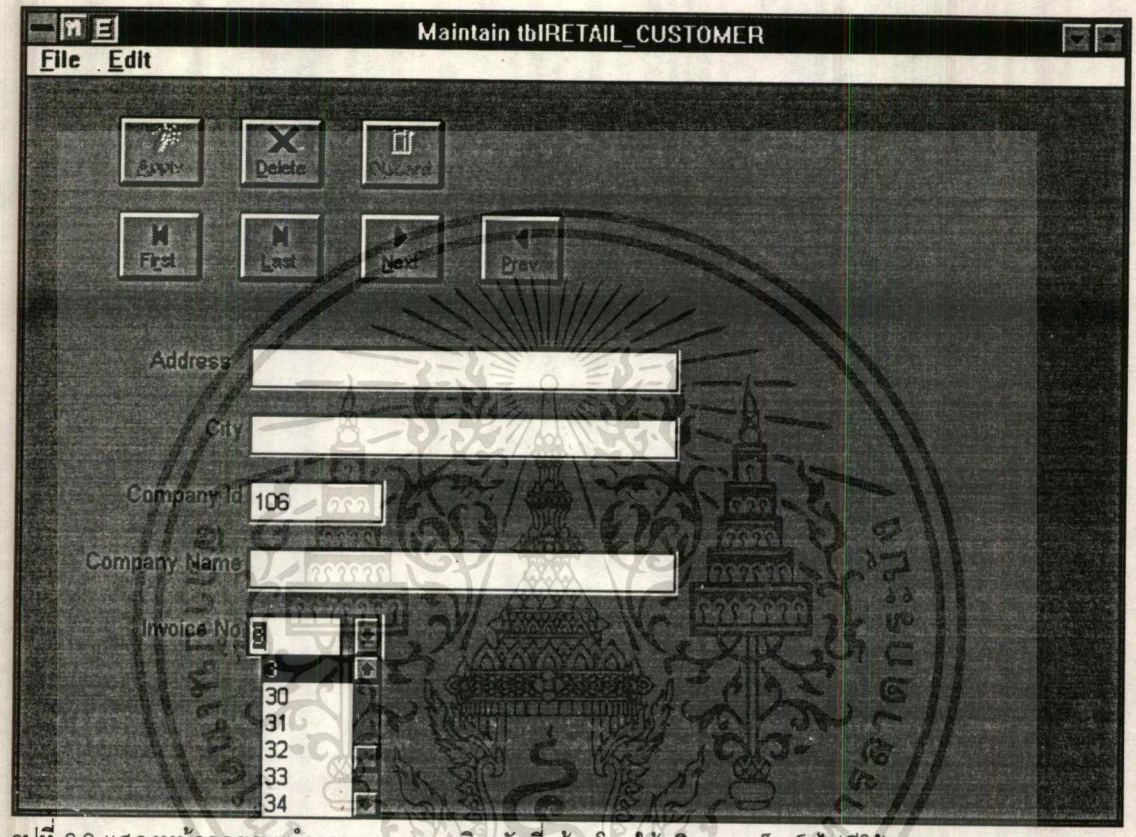
สำหรับ คอลัมน์ EXCLUSIVE เราเลือกแสดงข้อมูลเป็นเช็คบอกซ์ (check box) วิชวลไลซ์เซอร์ มีค่า YES หรือ NO เท่านั้น โดยเลือก เช็คบอกซ์ , cQuickCheckBox

การกำหนดพร็อบเพอตี้ของควิกออบเจ็กต์

เลือกเช็คบอกซ์ และใช้คัสตอมไมเซอร์ เรื่อง Quick Check Box... เพื่อกำหนดพร็อบเพอตี้ของเช็คบอกซ์ แล้วเลือก Yes/No สำหรับค่าของ เช็คบอกซ์ ที่ถูกเช็ค และไม่ถูกเช็คตามลำดับ

การแปะคอมมานเดอร์บน ฟอร์มวินโดวส์

เลือกปุ่มกด และ cQuickCommander จาก ทูลพาเหรด จะปรากฏลิสต์ของคอมมานเดอร์ทั้งหมด บนทูลพาเหรด เราสามารถเลือกแล้วแปะทีละอันบนทูลบาร์ ของฟอร์มวินโดวส์ หรือ เลือกทีเดียวแล้วแปะก็ได้ เมื่อสร้างฟอร์มที่เราต้องการเสร็จแล้วและเริ่มรันแอพพลิเคชันดังรูป 6.9



รูปที่ 6.9 แสดงหน้าจอขณะทำงานของแอพพลิเคชันที่สร้างโดยใช้ควิกออบเจ็กต์ (ไม่มีโค้ดเลย)

◆ **การเขียนโค้ดด้วยตนเอง**

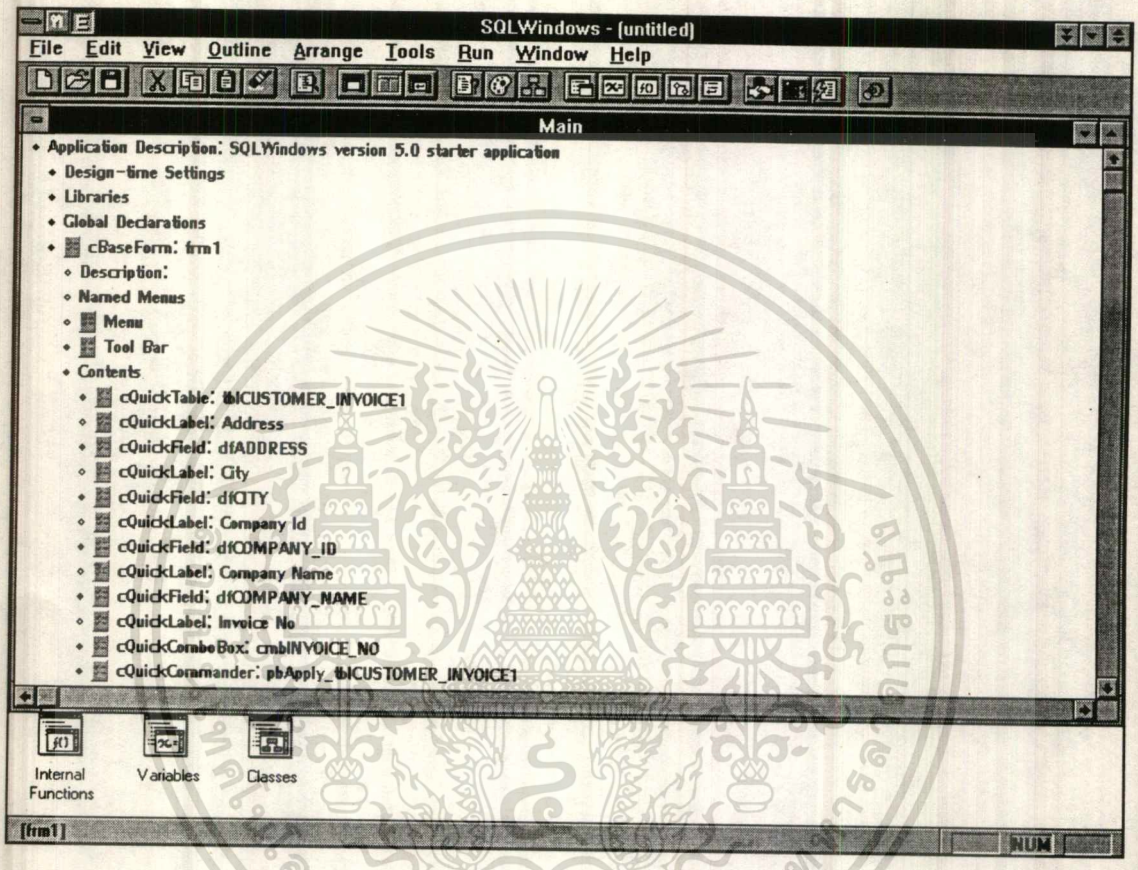
เราได้พัฒนาแอพพลิเคชันมาแล้ว 2 แบบโดยที่ไม่ต้องเขียนโค้ดเองแม้แต่น้อย ที่เราได้ทำไปแล้วจะเห็นว่า ควิกออบเจ็กต์ คือ สมาร์ทออบเจ็กต์ (smart objects) หมายถึง มันรู้ว่าจะแอคเซส และจัดการดาต้าซอร์สได้อย่างไร และมันเป็นคลาสที่ เอสคิวแอลวินโดวส์ กำหนดไว้ให้แล้ว และเราสามารถสร้างควิกออบเจ็กต์ ขึ้นเองได้ หรือเพิ่มเติมความสามารถของ ควิกออบเจ็กต์ ที่มีอยู่เดิมก็ได้

ถ้าเราต้องการแก้ไข ควิกออบเจ็กต์ ที่มีอยู่ หรือสร้าง ควิกออบเจ็กต์ ใหม่ หรือพัฒนาแอพพลิเคชันเองโดยไม่ใช้ ควิกออบเจ็กต์ เราก็ต้องเขียนโค้ดเองโดยใช้ภาษาของเอสคิวแอลวินโดวส์ ที่เรียกว่า SAL (เอสคิวแอลวินโดวส์ แอพพลิเคชัน Language) ซึ่งจะมีฟังก์ชันให้เรียกใช้ได้มากมายโดยขึ้นต้นด้วย Sal...

แอพพลิเคชันเอาท์ลายน์

เอสคิวแอลวินโดวส์ มีเอดิเตอร์สำหรับแก้ไขแอพพลิเคชัน โดยเฉพาะของมันเอง ซึ่งต่างจากเอดิเตอร์โดยทั่วๆไป หรือเวิร์ดโปรเซสเซอร์ ตรงที่มันสามารถเก็บโค้ดเป็นแบบโครงสร้าง

ประโยชน์ของเอดิเตอร์นี้คือ เราสามารถซ่อนสเตทเมนท์ ที่อยู่ภายใต้สเตทเมนท์หนึ่งลงได้ ถ้ายังไม่ต้องการดู แต่ละระดับจะแสดงด้วยรูปเพชร ซึ่งถ้าเป็นเพชรที่ทึบแสดงว่า มีสเตทเมนท์อื่นๆที่ซ่อนอยู่ใต้สเตทเมนท์นั้น ซึ่งจะเรียกดูได้โดยดับเบิลคลิกที่เพชรนั้น ดังรูปที่ 6.10



รูปที่ 6.10 แสดงเอาท์ลายน์ของแอพพลิเคชัน

เอาท์ลายน์วิว (Outline Views)

ถึงแม้ว่าจะมีเอาท์ลายน์เดียวสำหรับแอพพลิเคชันทั้งหมดแต่เราสามารถสร้างวิว (views/windows) ใหม่ได้ เพื่อเขียนโค้ดสำหรับส่วนหนึ่งของแอพพลิเคชันนั้นๆเท่านั้น เช่น เราสร้างวินโดวส์ใหม่ เพื่อแก้ไขเฉพาะล็อคอินไดอะล็อกบ็อกซ์ (dlgLogin) โดยเลือกเมนู Window, New Outline View...

แอพพลิเคชันไลบรารี

แอพพลิเคชันของเอสคิวแอลวินโดวส์ ปกติจะอยู่ในไฟล์ .APP ซึ่งสามารถรวมเอาโค้ดจากไฟล์อื่นๆที่เรียกว่า แอพพลิเคชันไลบรารี (อยู่ในไฟล์ .APL) เข้าไว้ด้วยกัน เนื่องจาก

ไลบรารีเป็นที่เก็บรวบรวมฟังก์ชัน, ไดอะล็อกบ็อกซ์ หรืออื่นๆที่ใช้งานบ่อยๆไว้ในไฟล์เพื่อให้แอพพลิเคชันหลายๆตัวสามารถรวม (include) เอาไลบรารีนั้นไว้ในตัวเองได้

เพื่อแบ่งแอพพลิเคชันขนาดใหญ่เป็นหลายๆไลบรารี ทำให้โปรแกรมเมอร์ต่างๆสามารถทำงานกับไฟล์ไลบรารีต่างๆได้หลายไฟล์ในเวลาเดียวกัน

เราสามารถสร้าง .APL ไฟล์ ได้ด้วยวิธีเดียวกับการสร้างแอพพลิเคชันไฟล์ได้ เพียงแต่ใส่ไฟล์เอ็กซ์เทนชัน .APL เป็นชื่อไฟล์เท่านั้น แล้วเราสามารถรวม .APL ไฟล์ เข้ากับแอพพลิเคชันโดยการเพิ่มข้อความ File Include: ... เข้าไปภายใต้ส่วนไลบรารี ในแอพพลิเคชันเอาท์ลายน์ ได้ โดยเราอาจบอกพาธของไฟล์ (full pathname) ตรงๆลงไปด้วย หรือ เลือกเมนู File, Preferences, General... และบอกพาธของไฟล์ก็ได้

เราสามารถแก้ไขไฟล์ไลบรารีได้โดยใช้เอดิเตอร์ตามปกติ หรือ เลือกไลบรารีที่ต้องการแก้ไขในแอพพลิเคชันเอาท์ลายน์ แล้วเลือกเมนู File, Edit Item หรือกด F5 ก็ได้

เอาท์ลายน์ออบชัน

เมื่อต้องการดู เอาท์ลายน์ออพชันบาร์ ให้เลือกเมนู View, Outline Options... หรือกด F2 หรือดูเอาท์ลายน์ออบชันในไดอะล็อกบ็อกซ์โดยเลือกเมนู File, Preferences, Outline Options...

เอาท์ลายน์ออพชันบาร์ จะแสดงไอเท็มต่างๆที่สามารถเพิ่มเข้าไปใน เอาท์ลายน์ ณ จุดที่เราอยู่ขณะนั้น (context-sensitive) เราใช้ เอาท์ลายน์ออพชันบาร์ เพื่อ

เลือกเมนู

เลือกฟังก์ชัน, ค่าคงที่, และตัวแปร

เลือก SAL statements

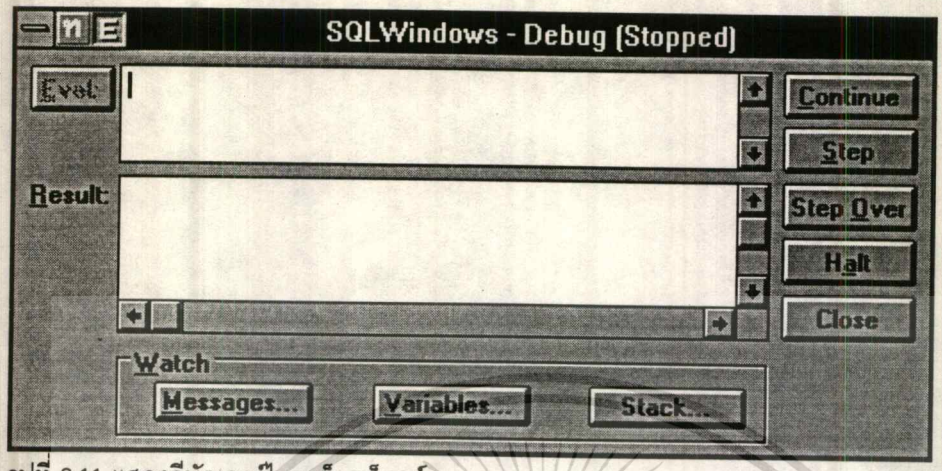
เลือกพารามิเตอร์สำหรับฟังก์ชันที่เราเรียก

สำหรับการเลือกใช้ฟังก์ชัน จะปรากฏทั้งฟังก์ชันของระบบ (system functions), หรือฟังก์ชันของผู้ใช้ (user functions) ด้วย

สำหรับการเลือกพารามิเตอร์ในฟังก์ชัน จะปรากฏทั้งตัวแปร, ตัวแปรระบบ, ค่าคงที่, ชื่อของวินโดวส์, แหล่งข้อมูล (resources), พารามิเตอร์ของฟังก์ชัน และ พารามิเตอร์ของวินโดวส์ (เมื่ออยู่ที่การกำหนดฟังก์ชัน หรือการกำหนดวินโดวส์) หรือเบสคลาสต่างๆ (base classes)

การดีบักแอพพลิเคชัน

เอสคิวแอลวินโดวส์ อนุญาตให้เราเซตเบรคพอยท์ (break point) ที่สเตทเมนท์ใดก็ได้ เมื่อเรารันแอพพลิเคชันโดยใช้โหมดผู้ใช้ การเอ็กซีคิวท์จะหยุดก่อนที่จะทำสเตทเมนท์นั้น และให้เราตรวจสอบค่าเอกซ์เพรสชัน และค่าตัวแปร และติดตามดูแมสเซจต่างๆที่เกิดขึ้นได้ หรือสามารถทำการเอ็กซีคิวท์ต่อไปได้ หรือเอ็กซีคิวท์สเตทเมนท์ขณะนั้น (step) หรือสเต็ปโอเวอร์ (step over) สเตทเมนท์นั้น หรือ หยุดแอพพลิเคชัน หรือปิดไดอะล็อกบ็อกซ์ก็ได้ โดยจะทำการเหล่านี้ได้จากดีบักเกอร์ไดอะล็อกบ็อกซ์ แสดงในรูป



รูปที่ 6.11 แสดงดีบักเกอร์ไดอะล็อกบ็อกซ์

เอสคิวแอลวินโดวส์คอมไพเลอร์

ทำหน้าที่ แปลอินเทอนอลฟังก์ชัน (internal function), ตัวแปรโกลบอล(global variables) และค่าคงที่ในแอพพลิเคชันเป็นสเตทเมนท์ในภาษา C แล้วคอมไพล์สเตทเมนท์นั้นเป็น DLL ในเวอร์ชันต่อไปของ เอสคิวแอลวินโดวส์คอมไพเลอร์ จะไม่ได้มีความสามารถเพียงแค่แปลอินเทอนอลฟังก์ชัน เท่านั้น

ทำหน้าที่สร้าง แอพพลิเคชันเอาท์ลายน์ อันใหม่ (ในไฟล์ .APC) ที่มีนิยามฟังก์ชันภายนอก (external function definition) สำหรับแต่ละฟังก์ชันใน DLL เราสามารถใช้ เอาท์ลายน์ นี้เป็นเหมือนไลบรารีในแอพพลิเคชันที่เรียกใช้ฟังก์ชันเหล่านั้น

การกระจายแอพพลิเคชันไปให้ผู้อื่น

ถ้าเราใช้ เอสคิวแอลวินโดวส์ 4.1 หรือเวอร์ชันต่อๆไป ทุกไฟล์ที่จำเป็นต้องใช้เมื่อรันแอพพลิเคชัน จะอยู่ในไดเรคตอรี DEPLOY เรียบร้อยแล้ว ถ้าต้องการตรวจสอบว่า มีชุดของไฟล์ครบหรือไม่ ก็สามารถตรวจสอบได้ในคู่มือ Power Programming with SQLWindows

ไอโซเลชันเลเวล

- **เคอร์เซอร์สเตบิลิตี้** (CS:Cursor Stability) ไอโซเลชันเลเวลนี้จะล็อคเฉพาะเพจที่เรากำลังโพรเซสอยู่ โดยมันจะยังคงมีแชร์ล็อค หรือเอ็กซ์คลูซีฟล็อคอยู่จนกระทั่งทรานแซคชันคอมมิท กล่าวคือเอสคิวแอลเบสจะล็อคเพจหนึ่งนานตราบเท่าที่เคอร์เซอร์ยังอยู่บนเพจนั้น ผู้ใช้คนอื่นๆจะสามารถโพรเซสเพจอื่นๆที่เราแอคเซสระหว่างทำทรานแซคชันได้โดยไม่ต้องรอให้เราคอมมิทก่อน ข้อมูลที่เราอ่านจากเพจหนึ่งจึงสามารถถูกผู้ใช้คนอื่นๆแก้ไขได้เมื่อเราย้ายเคอร์เซอร์ไปที่เพจใหม่ เมื่อมีการเอ็กซีคิวท์คำสั่ง SqlFetchNext หรือ SqlFetchPrevious เอสคิวแอลเบสจะส่งเรคคอร์ดหนึ่งของผลลัพธ์ข้ามเน็ตเวอร์กไปให้ไคลเอ็นท์เก็บไว้ในอินพุทแมสเซจบัฟเฟอร์ (input messege buffer)

การใช้เคอร์เซอร์สเตบิลิตี้เมื่อเราต้องการอัพเดตทีละเรคคอร์ด (record at a time) ให้ใช้คำสั่ง CURRENT OF <cursor> เมื่อเอสคิวแอลวินโดวส์รับเรคคอร์ดหนึ่งเข้ามาไว้ในอินพุทแมสเซจบัฟเฟอร์แล้ว เพจนั้นจะถูกแชร์ล็อคไว้ ซึ่งแปลว่า จะไม่มีทรานแซคชันใดๆสามารถอัพเดตมันได้

- **รีลีสล็อก** (RL:Release Locks) ไอโซเลชันเลเวลนี้เป็นการเพิ่มการทำคอนเคอเรนซีทรานแซคชันโดยการปล่อยแชร์ล็อคทั้งหมดเมื่อไคลเอ็นท์ได้รับเรคคอร์ดจากเอสคิวแอลเบสแล้ว ตรงข้ามกับการใช้เคอร์เซอร์สเตบิลิตี้ ซึ่งเมื่อเราย้ายเคอร์เซอร์ออกจากเพจหนึ่งแล้ว เอสคิวแอลเบสจะดร็อพ (drop) แชร์ล็อคทิ้ง แต่อย่างไรก็ตามถ้าเรคคอร์ดจากเพจนั้นยังคงอยู่ในอินพุทแมสเซจบัฟเฟอร์ของเรา เพจนั้นก็ยังคงถูกล็อคต่อไป

ไอโซเลชันเลเวลนี้จะมีการส่งผลลัพธ์ข้ามเน็ตเวิร์คทีละหลายๆเรคคอร์ด เป็นการลดการจราจรบนเน็ตเวิร์คได้

เราควรใช้ไอโซเลชันเลเวลนี้สำหรับการดูแอพพลิเคชันซึ่งจะแสดงเซตของเรคคอร์ดหลายๆอันแก่ผู้ใช้

- **อ่านอย่างเดียว** (RO:Read Only) ไอโซเลชันเลเวลนี้ไม่ทำล็อคบนฐานข้อมูลเลย และถ้าทรานแซคชันของเราอยู่ในโหมดนี้ เราจะทำการอ่านข้อมูลได้เท่านั้น เอสคิวแอลวินโดวส์จะไม่อนุญาตให้เราเอ็กซีคิวท์ภาษากำหนดข้อมูล (DDL: Data Definition Language) และภาษาจัดการข้อมูล (DML: Data Manipulation Language)

ถ้าเราร้องขอเพจที่ถูกล็อคโดยทรานแซคชันอื่นๆอยู่ เอสคิวแอลเบสจะสร้างก็อปปี้หรือวิวของเพจอันเก่าก่อนที่ทรานแซคชันแซคชันจะล็อคเพจไว้ จากไฟล์ประวัติศาสตร์ที่อ่านอย่างเดียว (read-only history ไฟล์) ให้เรา

ไฟล์ประวัติศาสตร์ที่อ่านอย่างเดียวเป็นตัวจัดการกับก็อปปี้หลายๆแบบของฐานข้อมูลเพจตามที่ถูกเปลี่ยนแปลงไป

- **อ่านซ้ำได้หลายครั้ง** (RR:Read Repeatablity) ไอโซเลชันเลเวลนี้ล็อคทุกเพจที่เราแอคเซสจนกระทั่งเราคอมมิททรานแซคชัน ถ้าเราอ่านเพจเดิมอีกครั้งระหว่างทำทรานแซคชัน เราก็จะได้ข้อมูลเช่นเดียวกับตอนที่อ่านครั้งแรก โหมดนี้จึงการันตีได้ว่าข้อมูลที่เราแอคเซสจะสอดคล้องกัน (consistent) ทุกเวลาของทรานแซคชัน โหมดนี้เป็นโหมดที่เอสคิวแอลเบสตั้งไว้อยู่แล้ว (default)

ไอโซเลชันเลเวลนี้มีการรับเรคคอร์ดทีละหลายๆเรคคอร์ดข้ามเน็ตเวอร์ก และทุกแชร์ล็อคจะยังคงอยู่โดยไม่คำนึงถึงขนาดของอินพุทแมสเซจบัฟเฟอร์ จนกระทั่งแอพพลิเคชันสั่งคอมมิท หรือโรลแบ็ค

6.1.2 แนวทางการพัฒนาแอพพลิเคชันบนวินโดวส์โดยใช้วิชวลเบสิก

จากการพัฒนาของระบบปฏิบัติการจากดอส (DOS) ไปสู่วินโดวส์ (Windows) การพัฒนาแอพพลิเคชันต่างๆซึ่งเดิมทีเคยใช้เครื่องมือในการพัฒนาเพียงแค่ แอสเซมเบลอร์, ซี คอมไพเลอร์, ปาสคาล คอมไพเลอร์ ฯลฯ กับโปรแกรมเอดิเตอร์ ก็เพียงพอแล้วสำหรับแอพพลิเคชันบนดอสซึ่งมีสภาวะแวดล้อมที่ไม่ซับซ้อนอะไรมากนัก แต่ในวินโดวส์ซึ่งมีสภาวะแวดล้อมแบบกราฟิก เป็นระบบมัลติทาส์กกิ้ง (multitasking) ฯลฯ เครื่องมือในการพัฒนาที่เคยใช้กันอยู่คงจะไม่เหมาะสมแล้วหากนำมาใช้ในการพัฒนาแอพพลิเคชันบนวินโดวส์เพราะจะทำให้ใช้ทรัพยากรทั้งกำลังคนและเวลาเป็นอย่างมาก จากจุดนี้เองจึงเป็นจุดเริ่มต้นของแนวคิดในการสร้างเครื่องมือต่างๆเพื่อมาช่วยสำหรับการพัฒนาแอพพลิเคชันบนวินโดวส์ให้สะดวกและง่ายยิ่งขึ้น

6.1.2.1 เครื่องมือที่สำคัญของวิชวลเบสิก

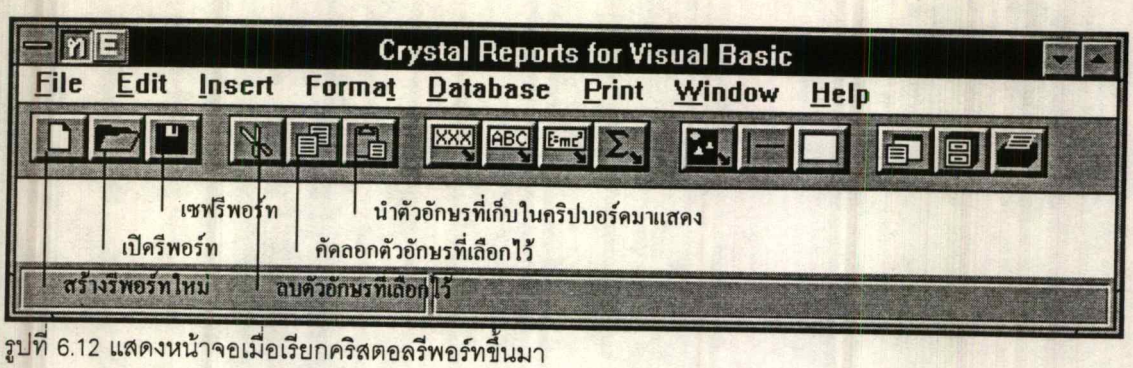
วิชวลเบสิกเป็นหนึ่งในเครื่องมือดังกล่าว ซึ่งแนวทางในการพัฒนาแอพพลิเคชันจะใช้ในการดึง คอมโพเน้นท์ (Component) ที่สมบูรณ์ภายในตัวเอง (Completed Component) ต่างๆที่มีอยู่มาประกอบเข้าด้วยกัน ซึ่งมีข้อดีคือสามารถพัฒนาแอพพลิเคชันได้อย่างง่ายดายและรวดเร็ว โดยมีเครื่องมือที่สำคัญดังนี้

6.1.2.1.1 **คริสตอลรีพอร์ท** (Crystal Report)

เป็นเครื่องมือสำหรับการทำรายงาน ลิสต์ (List) และแบบฟอร์มจดหมาย โดยใช้ข้อมูลจากฐานข้อมูลที่มีอยู่ คริสตอลรีพอร์ทเป็นเครื่องมือที่ถูกออกแบบมาให้ใช้งานได้กับข้อมูลทุกรูปแบบ เช่น ตัวเลข (number) เคอเรนซี่ (currency) ตัวอักษร วันที่ และ บูลีน (Boolean) นอกจากนั้นยังเป็นเครื่องมือแบบบิวท์อิน (Build-in Tool) ที่สามารถจัดการกับข้อมูลได้ตามที่ผู้ใช้ต้องการ เช่น ทำการคำนวณและเปรียบเทียบค่าของข้อมูล การเปลี่ยนรูปแบบของข้อมูลให้เป็นรูปแบบอื่น ฯลฯ

คริสตอลรีพอร์ทสามารถใช้ข้อมูลจากไฟล์ฐานข้อมูลทั่วไป เช่น ดีเบสเวอร์ชัน 3, 3+, และ 4 ฟอกซ์โปร คลิปเปอร์ พาราด็อกซ์ แอคเซสเวอร์ชัน 1.0 และ 1.1 และ บีทรีฟ

เราสามารถเรียกให้คริสตอลรีพอร์ทพิมพ์รายงานได้จากส่วนการเขียนโปรแกรมในวิชวลเบสิกดีไซน์โดยใช้คริสตอลคัสตอมคอนโทรล (Crystal Custom Control) โดยเมื่อเรียกคริสตอลรีพอร์ทขึ้นมาแล้วได้ดังรูปที่ 6.12

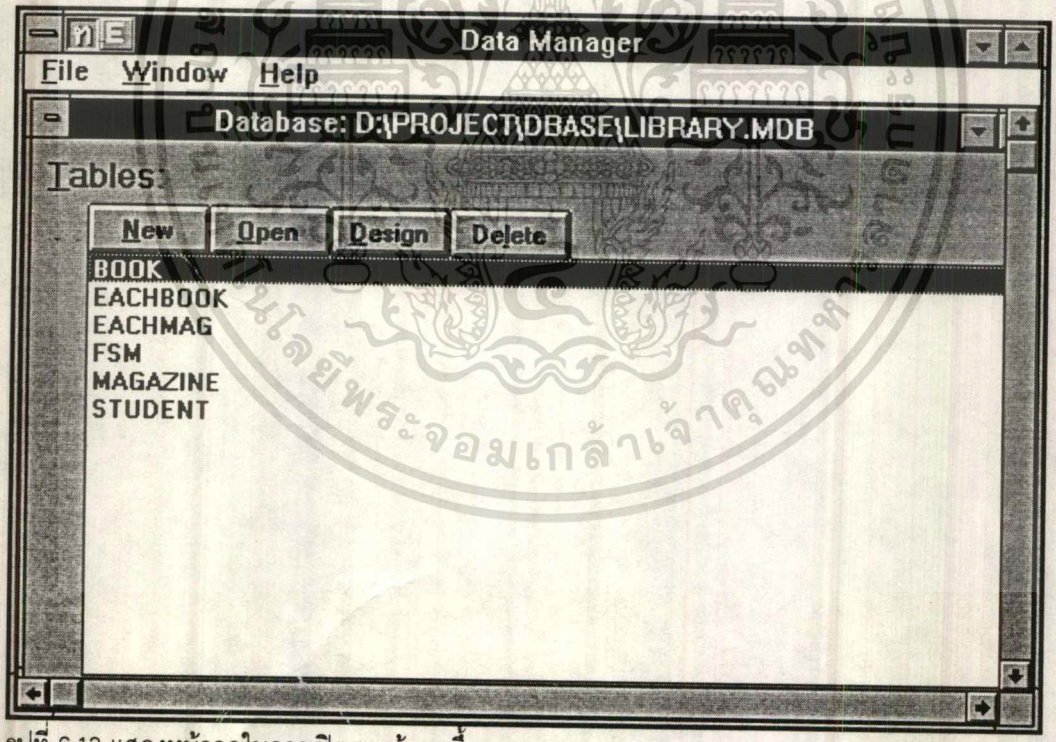


รูปที่ 6.12 แสดงหน้าจอเมื่อเรียกคริสตอลรีพอร์ทขึ้นมา

6.1.2.1.2 **ตัวจัดการฐานข้อมูล** (Data manager)

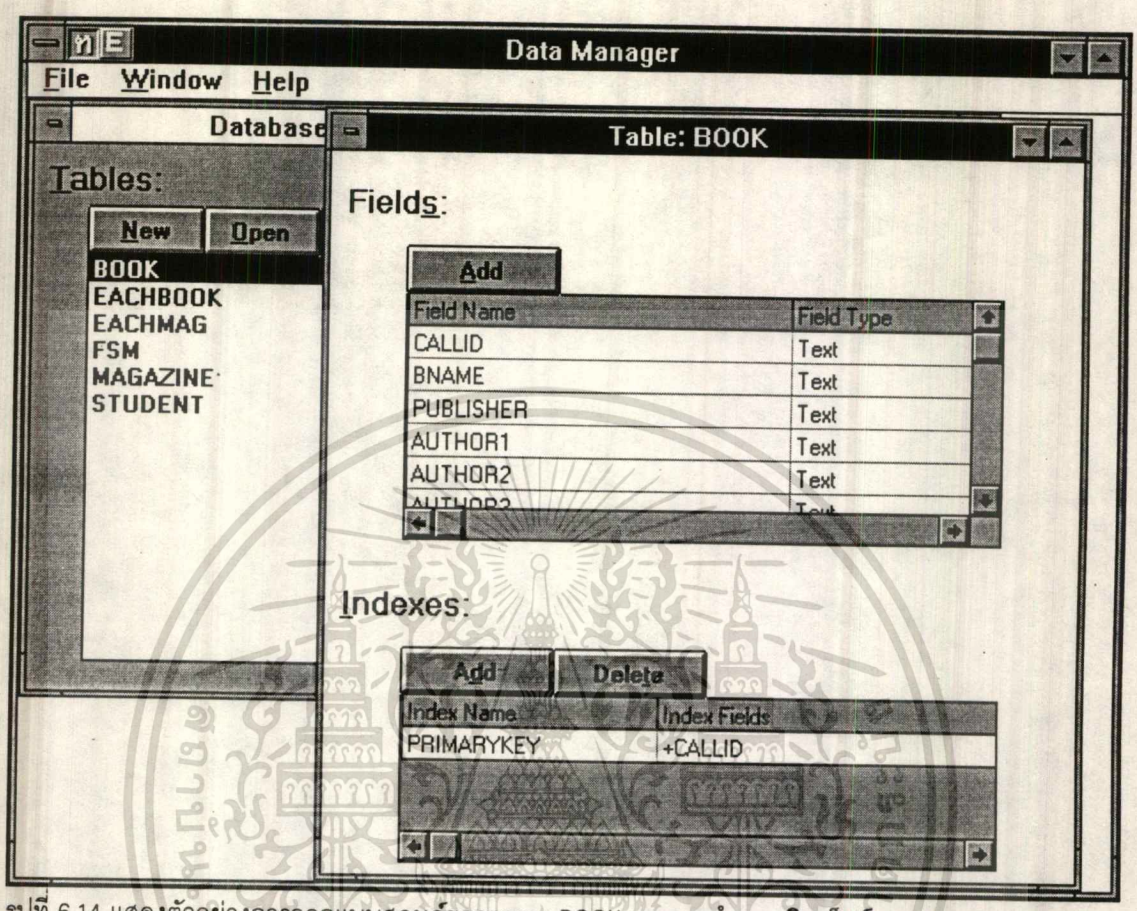
จะเป็นเครื่องมือที่ใช้สำหรับสร้างฐานข้อมูลตัวใหม่หรือทำการมานิพูเลท (Manipulation) กับข้อมูลในฐานข้อมูลที่มีอยู่แล้ว ตัวจัดการฐานข้อมูลสามารถทำงานกับไฟล์ฐานข้อมูลทั่วไปเช่นเดียวกับคริสตอลรีพอร์ท

เมื่อเรียกขึ้นมาตัวจัดการฐานข้อมูลแล้วเปิดฐานข้อมูลขึ้นมาจะขึ้นหน้าจอดังนี้



รูปที่ 6.13 แสดงหน้าจอในการเปิดฐานข้อมูลขึ้นมา

ในกรณีที่ต้องการสร้างฐานข้อมูลตัวใหม่ เราจะต้องสร้างตารางขึ้นมาแล้วกำหนดรูปแบบข้อมูลให้กับแต่ละสดมภ์แล้วทำการกำหนดตัวชี้ (index) รูปที่ 6.14 แสดงตัวอย่างการออกแบบสดมภ์ของตาราง BOOK



รูปที่ 6.14 แสดงตัวอย่างการออกแบบสคมภ์ของตาราง BOOK และการกำหนดอินเด็กซ์

แต่ตัวจัดการฐานข้อมูลที่วิชวลเบสิกให้มานี้เป็นการจัดการอย่างง่ายๆ เพราะยังขาดฟีเจอร์ต่างๆ ที่จำเป็นสำหรับการสร้างคีย์ เช่น การกำหนดฟอเรนคีย์ คอมมอนคีย์ การเชื่อมโยงระหว่างตาราง ฯลฯ

6.1.2.1.3 **วิชวลเบสิกดีไซน์** (Visual Basic Design)

เป็นเครื่องมือที่ใช้สำหรับพัฒนาแอพพลิเคชัน โดยขั้นตอนที่ใช้ในการพัฒนาแอพพลิเคชันของวิชวลเบสิกนั้นจะประกอบด้วยขั้นตอนหลักๆ 3 ขั้นตอนคือ

- ขั้นตอนการสร้างส่วนติดต่อกับผู้ใช้ (Interface)
- ขั้นตอนการกำหนดพร็อบเพอร์ตี้ (Property)
- ขั้นตอนการเขียนและแก้ไขปรับปรุงโค้ด (Code)

ขั้นตอนการสร้างส่วนติดต่อกับผู้ใช้

ขั้นตอนการสร้างส่วนติดต่อกับผู้ใช้นี้เป็นขั้นตอนแรกในการพัฒนาแอพพลิเคชันของวิชวลเบสิกซึ่งในขั้นตอนนี้จะประกอบด้วยขั้นตอนย่อยๆดังนี้

- ขั้นตอนการสร้างโปรเจ็กต์ (Project)
- ขั้นตอนการสร้างฟอร์ม (Form)
- ขั้นตอนการประกอบคอนโทรล (Control) เข้ากับฟอร์ม

- ขั้นตอนการสร้างโปรเจ็กต์

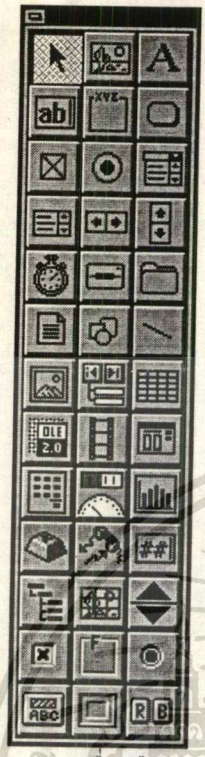
การสร้างโปรเจ็กต์เพื่อเป็นการรวบรวมส่วนต่างๆของแอพพลิเคชันเข้ามาไว้เป็นกลุ่มเดียวกัน

- ขั้นตอนการสร้างฟอร์ม

ขั้นตอนนี้จะเป็นการสร้างฟอร์ม (ฟอร์มหมายถึง วินโดวส์หรือไดอลอกบอกซ์(dialog box) ที่ใช้สำหรับส่วนติดต่อกับผู้ใช้ของแอพพลิเคชัน)

สำหรับแต่ละวินโดวส์ในแอพพลิเคชันซึ่งฟอร์มแต่ละฟอร์มจะหมายถึงวินโดวส์ในแอพพลิเคชัน ฟอร์มนั้นถือเป็นออบเจ็กต์ชนิดหนึ่งในวิชวลเบสิกและนอกจากฟอร์มธรรมดาแล้วยังมีฟอร์มชนิดพิเศษซึ่งสามารถมีฟอร์มอื่นๆในตัวเองได้อีกซึ่งเรียกว่า เอ็มดีไอ ไชด์ฟอร์ม (MDI child form)

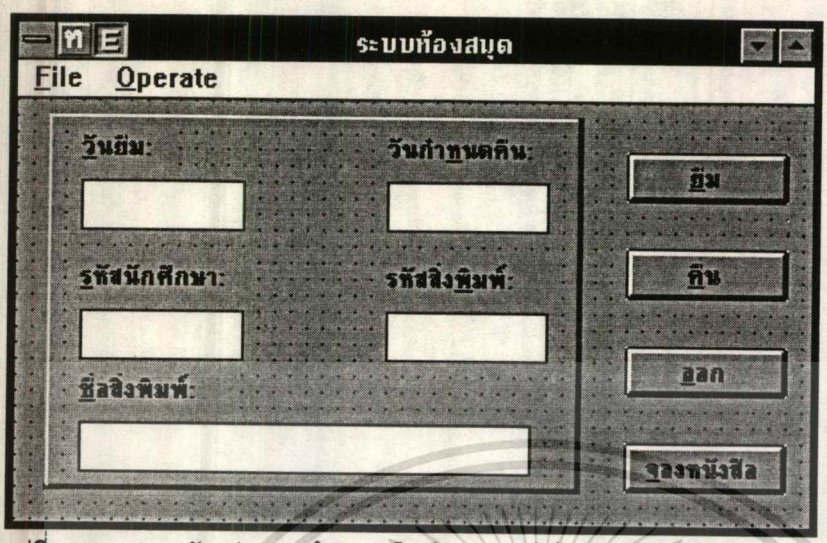
เราจะใช้กล่องเครื่องมือเป็นเครื่องมือในการสร้างออบเจ็กต์ที่ต้องการบนฟอร์มโดยคลิกบนไอคอนในกล่องเครื่องมือ ดังรูปที่ 6.15 ที่ต้องการแล้วทำการวาดลงบนฟอร์ม



รูปที่ 6.15 แสดงกล่องเครื่องมือของวิชวลเบสิก

◆ ขั้นตอนการประกอบคอนโทรลเข้ากับฟอร์ม

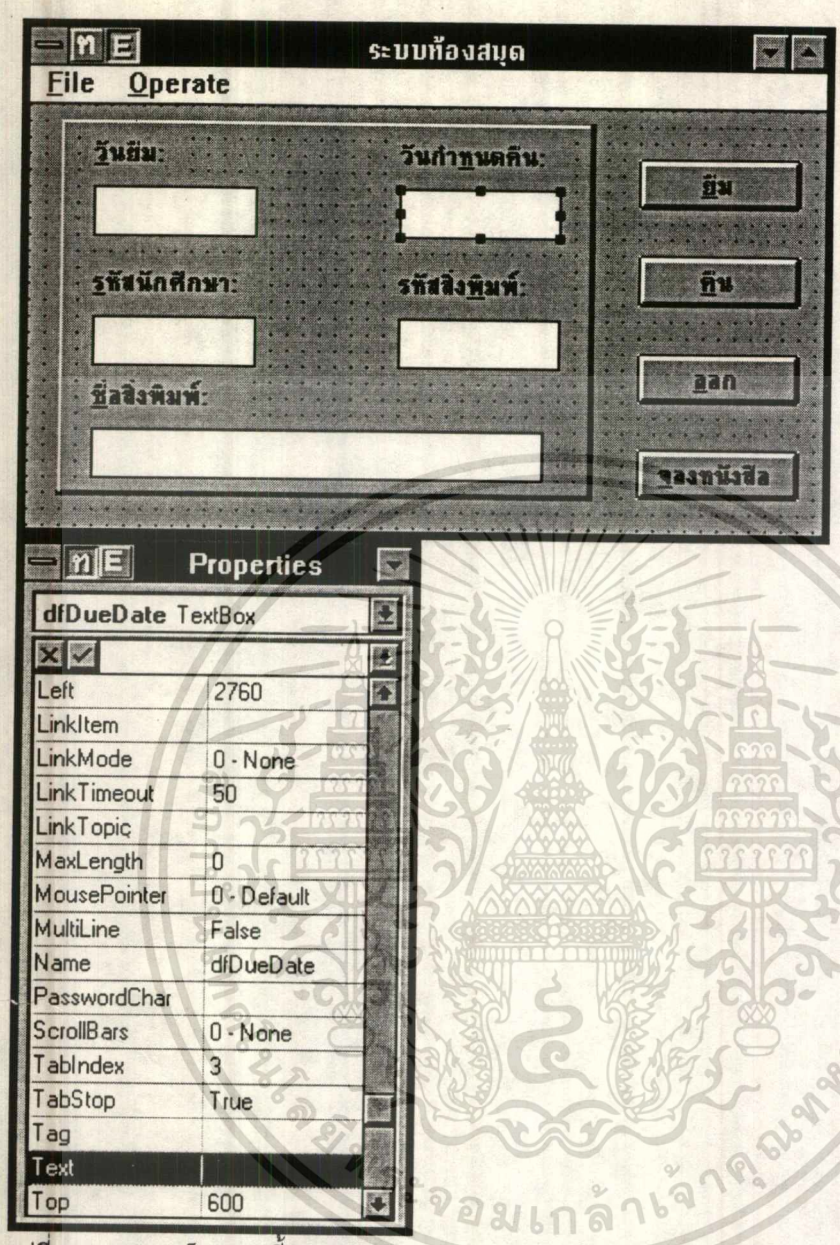
ขั้นตอนนี้จะเป็นขั้นตอนที่ทำการประกอบคอนโทรล (คอนโทรล หมายถึง ออบเจ็กต์ ซึ่งใช้ประกอบเข้าไปในฟอร์ม เพื่อทำให้สามารถสนับสนุนการทำการติดต่อระหว่างผู้ใช้กับแอพพลิเคชัน คอนโทรลแต่ละตัวจะมีพร็อบเพอร์ตี้และเมททอดเป็นคุณสมบัติเฉพาะของตัวเอง ทำให้เกิดความแตกต่างสำหรับการใช้งาน) ที่จะใช้เข้ากับฟอร์ม คอนโทรลต่างๆจะใช้ในการรับอินพุทจากผู้ใช้ซึ่งจะมาในรูปของอีเวนท์ (อีเวนท์ หมายถึง แอคชันซึ่งรับรู้โดยออบเจ็กต์ เช่นการคลิกเมาส์(mouse) การกดคีย์หรืออาจมาจากระบบก็ได้) การแสดงเอาท์พุท ในการที่จะให้คอนโทรลตอบสนองต่อ อีเวนท์ต่างๆเราจะต้องเขียนโค้ดสำหรับการตอบสนองต่ออีเวนท์นั้นๆ ซึ่งลักษณะเช่นนี้เราเรียกว่าเป็นการโปรแกรมแบบอีเวนท์ดริฟเวน (Event Driven Programming) ตัวอย่างของคอนโทรลที่เราพบเห็นบ่อยๆจากแอพพลิเคชันบนวินโดวส์อย่างเช่น เมนูบาร์ (Menu bar) คอมมานด์บัททอน (Command button) เช็คบอกซ์ (Check box) ฯลฯ ดังเช่นรูปที่ 6.16 แสดงการสร้างฟอร์มเพื่อใช้ติดต่อกับผู้ใช้



รูปที่ 6.16 แสดงตัวอย่างการนำออบเจ็กต์ประกอบเข้าไปในฟอร์ม

ขั้นตอนการกำหนดพร็อบเพอร์ตี้

พร็อบเพอร์ตี้คือสถานะ (state) ของออบเจ็กต์ ออบเจ็กต์แต่ละออบเจ็กต์ก็มีพร็อบเพอร์ตี้เป็นของตัวเอง ตัวอย่างของพร็อบเพอร์ตี้เช่น สี ขนาด ตำแหน่ง ฯลฯ การกำหนดพร็อบเพอร์ตี้สามารถกระทำได้ทั้งระหว่างการออกแบบ (Design time) และระหว่างการรัน (Run time) การกำหนดพร็อบเพอร์ตี้นั้นอาจเพื่อเป็นการแสดงข้อความหรือภาพไปสู่ผู้ใช้ การเลื่อนตำแหน่งของคอนโทรล การให้คอนโทรลปฏิเสธการรับอีเวนท์ทุกอีเวนท์ หรือการกำหนดช่วงเวลาสำหรับทำการขัดจังหวะการทำงาน(Interrupt) ฯลฯ รูปที่ 6.17 แสดงการกำหนดพร็อบเพอร์ตี้ในระหว่างการออกแบบ ทำได้โดยการคลิกที่ออบเจ็กต์ที่ต้องการ แล้วเปิดหน้าต่างพร็อบเพอตี้เพื่อกำหนดคุณสมบัติ ในตัวอย่างจะเป็นการกำหนดพร็อบเพอตี้ของออบเจ็กต์เท็กต์บอกซ์



รูปที่ 6.17 แสดงพร็อบเพอตี้ของออบเจ็กต์เท็กต์บ็อกซ์

ขั้นตอนการเขียนและแก้ไขปรับปรุงโค้ด

ดังที่กล่าวมาจากข้างต้น ในวิชวลเบสิกการเขียนโค้ดจะเขียนเพื่อให้แอพพลิเคชันมีการตอบสนองต่ออีเวนท์ที่เกิดกับคอนโทรลหรือฟอร์มเท่านั้น เมื่อเวลาที่เกิดอีเวนท์ขึ้น โค้ดที่เขียนขึ้นเพื่อตอบสนองต่ออีเวนท์ (Event Procedure)จะถูกเรียกขึ้นมาทำงาน (Execute)

6.1.2.2 การใช้วิชวลเบสิกในการพัฒนาฟรอนท์เอ็นแอพพลิเคชัน

หนึ่งในข้อดีข้อหนึ่งของวิชวลเบสิกคือการสนับสนุนการเข้าถึงฐานข้อมูลแบบสัมพันธ์ทั้งโดยตรงหรือผ่านระบบจัดการฐานข้อมูลสัมพันธ์ ซึ่งทำให้สะดวกในการพัฒนาฟรอนท์เอ็นแอพพลิ

เคชันกว่าการที่เราจะสร้างฟังก์ชันในการใช้งานของเราเอง ตัวอย่างของฐานข้อมูลสัมพันธ์และระบบจัดการฐานข้อมูลสัมพันธ์ที่วิชวลเบสิกสนับสนุนซึ่งใช้กันอย่างแพร่หลายเช่น แอคเซส ฟอกซ์โปร ดีเบส ออราเคิลฯลฯ รวมทั้งเอสคิวแอลเซิร์ฟเวอร์ด้วย

ในวิชวลเบสิกมีอยู่ 2 วิธีในการติดต่อกับระบบฐานข้อมูล คือการใช้ออบเจ็กต์สำหรับเข้าถึงข้อมูล (Data access Object) และ การใช้ดาต้าเบสคอนโทรล (Data Control) ทั้งสองแบบนี้จะมีข้อดีแตกต่างกัน ซึ่งจะกล่าวถึงเฉพาะแต่การใช้ออบเจ็กต์สำหรับเข้าถึงข้อมูลเท่านั้น เพราะมีความเหมาะสมกว่าสำหรับการพัฒนาฟรอนท์เอ็นแอพพลิเคชันบนระบบจัดการฐานข้อมูลสัมพันธ์ที่สนับสนุนการทำงานแบบมัลติยูสเซอร์อย่างเช่นออราเคิล เอสคิวแอลเซิร์ฟเวอร์ ฯลฯ เป็นต้น

ออบเจ็กต์สำหรับเข้าถึงข้อมูล

วิชวลเบสิกก็เป็นอีกหนึ่งที่ใช้แนวทางการพัฒนาแอพพลิเคชันแบบออบเจ็กต์โอเรียนเต็ดโปรแกรมมิ่ง(opject oriented programming) การมองสิ่งต่างๆของวิชวลเบสิกจึงมองเสมือนเป็นออบเจ็กต์ซึ่งรวมไปถึงฐานข้อมูลด้วย โดยจะมองฐานข้อมูลเสมือนเป็นออบเจ็กต์ตัวหนึ่งซึ่งมีชื่อว่าดาต้าเบสและจะมองข้อมูลที่มีอยู่ในฐานข้อมูลเสมือนเป็นออบเจ็กต์อีกตัวหนึ่งซึ่งเรียกว่า เรคคอร์ดเซต (Recordset) และการเข้าถึงฐานข้อมูลเพื่อการใช้งานทั้งหมดจะต้องผ่านกระทำผ่านออบเจ็กต์นั้น (หมายถึงการใช้เมททอดของออบเจ็กต์นั้น)

การเข้าถึงระบบฐานข้อมูลเพื่อนำข้อมูลที่มีอยู่มาใช้งานหรือแก้ไขเราจำเป็นต้องสร้าง เรคคอร์ดออบเจ็กต์ (Recordset object) ก่อนซึ่งในวิชวลเบสิกมีเรคคอร์ดออบเจ็กต์อยู่ 3 ชนิดด้วยกัน คือ ไดนาเซต (Dynaset) สแนพชอต (Snapshot) ตาราง ซึ่งมีรายละเอียดดังนี้

ไดนาเซตออบเจ็กต์

เป็นรูปแบบหนึ่งของเรคคอร์ดเซตที่เป็นผลลัพธ์ที่ได้จากการทำคิวรี่ (Query) สามารถมีเรคคอร์ดที่ทำการอัพเดทได้ และไดนาเซตจะประกอบด้วย แถว (Row) และ ฟิวด์ (Field) เหมือนตาราง ซึ่งฟิวด์ของไดนาเซตอาจจะมาจากตารางมากกว่าหนึ่งตารางในฐานข้อมูลก็ได้ เรคคอร์ดเซตของไดนาเซตจะเป็นแบบไดนามิก (Dynamic) ซึ่งเราสามารถเพิ่มเรคคอร์ดใหม่ ทำการเปลี่ยนแปลงข้อมูลในเรคคอร์ด รวมทั้งสามารถลบเรคคอร์ดออกได้

สแนพชอตออบเจ็กต์

เป็นรูปแบบหนึ่งของเรคคอร์ดเซต ที่เป็นผลลัพธ์ที่ได้จากการทำคิวรี่ (Query) แต่ไม่สามารถมีเรคคอร์ดที่ทำการอัพเดทได้และสแนพชอตจะประกอบด้วย แถว (Row) และฟิวด์ (Field) เหมือน

ตาราง ซึ่งฟิวด์ของสแนพชอตอาจจะมาจากตารางมากกว่าหนึ่งตารางในฐานข้อมูลก็ได้ เรคคอร์ดเซตของสแนพชอตจะเป็นแบบสตาติก (Static) ซึ่งเราไม่สามารถเพิ่มเรคคอร์ดใหม่ ทำการเปลี่ยนแปลงข้อมูลในเรคคอร์ด รวมทั้งสามารถลบเรคคอร์ดออกได้

เทเบิ้ลออบเจ็กต์ (Table Object)

เป็นรูปแบบหนึ่งของเรคคอร์ดเซตที่เป็นส่วนหนึ่งของฐานข้อมูล เทเบิ้ลออบเจ็กต์เป็นการแสดงผลระดับลอจิกคอล (Logical) ของฟิสิคอลเทเบิ้ล ประกอบด้วย แถว (Row) และ ฟิวด์ (Field)

ออบเจ็กต์ในแต่ละชนิดนั้นก็มีขอบเขตการใช้งานและความสามารถรวมถึงประสิทธิภาพที่แตกต่างกัน ซึ่งเราใช้สแนพชอตออบเจ็กต์เพราะเหมาะสมที่สุดสำหรับการพัฒนาฟรอนท์เอ็นแอพพลิเคชันในงานของเราซึ่งมันมีความสามารถในการติดต่อกับระบบจัดการฐานข้อมูลสัมพันธ์โดยใช้โอดีบีซี (ODBC, Open DataBase Connectivity) และยังสามารถทำการเปลี่ยนแปลงข้อมูลในฐานข้อมูลได้อีก ความสามารถเหล่านี้ล้วนจำเป็นทั้งสิ้นในงาน ซึ่งในเรคคอร์ดออบเจ็กต์ตัวอื่นๆยังขาดความสามารถเหล่านี้บางตัว

ขั้นตอนการสร้างไดนาเซตออบเจ็กต์มีดังนี้

- สร้างดาต้าเบสออบเจ็กต์โดยใช้ฟังก์ชันชื่อ OpenDatabase
- สร้างเรคคอร์ดเซตออบเจ็กต์จากดาต้าเบสออบเจ็กต์โดยใช้เมททอดชื่อ CreateDynaset ของดาต้าเบสออบเจ็กต์

หากจะมองแล้ววิชวลเบสิกนั้นจัดได้เป็นภาษาในยุคที่ 3 (3th Language) การเข้าถึงข้อมูลนั้นยังคงเป็นแบบเรคคอร์ดโอเรียนเต็ด (Record oriented) หรือ เรคคอร์ดแอทอะทาม (Record at a time) อยู่ จึงจำเป็นต้องมีพอย์นเตอร์ที่เป็นตัวบอกถึงแถวปัจจุบันในข้อมูลทั้งหมด ข้อมูลของแต่ละสดมในแถวปัจจุบันจะเป็นพร็อบเพอร์ตี้ของเรคคอร์ดเซต เราจึงสามารถอ้างถึงข้อมูลได้โดยการอ้างถึงพร็อบเพอร์ตี้ของเรคคอร์ดเซตนั้นเอง และในการเลื่อนตำแหน่งของพอย์นเตอร์เพื่อเปลี่ยนแถวที่อ้างถึงในปัจจุบันนั้นก็สามารถทำได้โดยใช้เมททอดชื่อ Movefirst Movelast Moveprevoiuse Movenext ของไดนาเซต อีกทั้งเรายังสามารถทำการค้นหาแถวที่ต้องการจากข้อมูลทั้งหมดได้โดยใช้เมททอดชื่อ Findfirst Findlast Findpreviouse Findnext ของ ไดนาเซตออบเจ็กต์อีกเช่นกัน

เมททอดชื่อ AddNew Delete Edit Update ของไดนาเซตออบเจ็กต์ทั้ง 3 นี้เป็น เมททอดที่ใช้ในการเปลี่ยนแปลงข้อมูลในฐานข้อมูลซึ่งใช้เวลาทำการเพิ่ม การลบ และการแก้ไข ข้อมูล และที่พิเศษคือหากใช้ดาต้าเบสออบเจ็กต์แล้วเราสามารถทำงานแบบทรานแซคชันได้ซึ่งจะมีเมททอดอยู่

3 เมททอดที่จำเป็นต้องใช้เพื่อให้เกิดการทำงานแบบทรานแซคชันคือเมททอดชื่อ BeginTrans CommitTrans และ Rollback ของไดนาเซตออบเจ็กต์

ออบเจ็กต์ต่างๆตามข้างต้น เวลามีการสร้างขึ้นมาหรืออยู่ในขณะใช้งานก็ตามจะทำให้เกิดการ ล็อคทรัพยากรขึ้น ดังนั้นการที่ให้ดีคือหากเราเลิกใช้งานออบเจ็กต์เหล่านี้แล้วก็ควรจะทำการทำลาย ซึ่งวิธีที่จะทำลายนั้นคือการใช้เมททอดชื่อ Close ของออบเจ็กต์นั้นๆ และมีกฎอยู่ข้อหนึ่งคือก่อนที่เราจะทำการทำลายดาต้าเบสออบเจ็กต์นั้น เราต้องทำลายเรคคอร์ดเซตของดาต้าเบสออบเจ็กต์นั้นก่อนมิฉะนั้นแล้วการเพิ่ม และการแก้ไขข้อมูลที่ทำไปจะไม่เป็นผล ฟังก์ชันและเมททอดที่กล่าวถึงมีรายละเอียดดังนี้

<u>ฟังก์ชัน OpenDatabase</u>

OpenDatabase เป็นฟังก์ชันที่ทำการสร้างดาต้าเบสออบเจ็กต์

กฎการใช้ของฟังก์ชัน

Set databaseobject = OpenDatabase(databasename[,exclusive[,readonly[,connect]]])

พารามิเตอร์ databasename คือ Registered data source name ส่วน connect จะใช้ดังนี้

ODBC;Dsn=Server;Uid=User;Pwd=Password

ซึ่ง Server หมายถึงชื่อโอดีบีซีเซิร์ฟเวอร์ User และ Password หมายถึง User ID และรหัสผ่านที่ใช้ในการ Login และนอกจากนี้ยังมีอีกสองส่วนคือเอ็กคลูสีฟ ซึ่งถ้ากำหนดเป็น true จะทำให้ผู้ใช้คนอื่นๆ ไม่สามารถเปิดฐานข้อมูลนี้ได้ แต่ถ้ามีผู้ใช้คนอื่นเปิดฐานข้อมูลแบบเอ็กคลูสีฟอยู่จะทำให้ไม่สามารถเปิดฐานข้อมูลได้ ส่วน readonly ถ้ากำหนดให้เป็น true จะเป็นการเปิดฐานข้อมูลแบบ readonly ทำให้ไม่สามารถทำการเปลี่ยนแปลงฐานข้อมูลได้

<u>เมททอด Close</u>

เมื่อหลังจากสิ้นสุดใช้ฐานข้อมูลแล้วก็ควรจะทำการ close เพื่อเป็นการยกเลิก resource ที่จองไว้อยู่ ยกเลิกการล็อคหรือใช้ทรัพยากรร่วมกันที่ฐานข้อมูล และโรลแบคทรานแซคชันที่ยังไม่คอมมิท

กฎการใช้

databaseobject.close

dynasetobject.close

ในกรณีที่มี เรคคอร์ดเซต ใน database object จะต้องทำการ close เรคคอร์ดเซตออบเจ็กต์ทุกตัวก่อนจึงจะสามารถ close database ได้โดยการใช้ close method

เมททอด CreateDynaset

CreateDyanset เป็น method ที่ใช้ในการสร้าง Dynaset object จาก Database object หรือ เรคคอร์ดเซตออบเจ็กต์

Syntax ของ method

Set dynasetobject = database.CreateDynaset(source [, options])

Set dynasetobject = { recordset | querydef }.CreateDynaset([options])

เมื่อกำหนดให้

dynasetobject คือ ตัวแปร object ชนิด Dynaset

database คือ ตัวแปร object ชนิด Database

เรคคอร์ดเซต คือ ตัวแปร object ชนิด เรคคอร์ดเซต ต่างๆ

querydef คือ ตัวแปร object ชนิด QueryDef

source คือ String expression ที่เป็นชื่อของ เรคคอร์ดเซต ที่มีอยู่แล้ว ,QueryDef หรือ SQL statement

options คือ numeric expression ที่บอกถึงตัวเลือกที่ทำการตั้งค่าซึ่งอาจตั้งค่าทีละหลายตัวเลือกได้โดยการนำค่าต่างๆที่ตั้งไว้นำมาบวกกัน

* ค่าที่ใช้ในการตั้งค่าตัวเลือกสามารถดูได้จาก Help ใน Visual basic

เมททอด BeginTrans ,CommitTrans ,Rollback Methods

BeginTransbegins เป็น method ที่ใช้ในการเริ่มทำ transaction

CommitTrans เป็น method ที่ใช้ในการจบการทำ transaction ที่กำลังทำอยู่

Rollback เป็น method ที่ใช้ในการจบการทำ transaction ที่กำลังทำอยู่และทำให้ฐานข้อมูลกลับไปอยู่ในสถานะเสมือนตอนเริ่มแรกที่เริ่มทำ transaction

กฎการใช้

database.BeginTrans

database.CommitTrans

database.Rollback

เมททอด AddNew

เป็นเมททอดที่ทำเคลียบัฟเฟอร์เพื่อเตรียมเนื้อที่สำหรับการสร้างเรคคอร์ดขึ้นมาใหม่ใน Dynaset

syntax ของ method

recordset.AddNew

Method FindFirst Find Last , Find Next, FindPrevious

เป็น method ที่ทำการกำหนดเรคคอร์ดปัจจุบันให้เป็นเรคคอร์ดตำแหน่ง เริ่มต้น , ตำแหน่งสุดท้าย , ตำแหน่งถัดไป หรือ ตำแหน่งก่อนหน้า ที่เป็นไปตามเงื่อนไขที่กำหนดไว้ (criteria)

syntax ของ method

recordset.FindFirst criteria

recordset.FindLast criteria

recordset.FindNext criteria

recordset.FindPrevious criteria

เมื่อกำหนดให้

recordset คือตัวแปรที่เป็น ไดนาเซตหรือสแนพชอต

criteria คือ string expression (เป็น where Clause ใน SQL String โดยที่ไม่มีคำว่า "Where") ที่กำหนดเรคคอร์ดที่ต้องการ

ถ้ามีเรคคอร์ดที่มีคุณสมบัติตรงตาม criteria มากกว่า 1 เรคคอร์ด FindFirst จะกำหนดให้เรคคอร์ดปัจจุบันเป็นเรคคอร์ดแรก FindNext จะกำหนดให้เรคคอร์ดถัดจากเรคคอร์ดถัดไป

ถ้าไม่มีเรคคอร์ดใดเลยที่ตรงตามเงื่อนไขที่ต้องการ เรคคอร์ดปัจจุบันก็ยังไม่เปลี่ยนแปลง

เมททอด MoveFirst , MoveLast , MoveNext , MovePrevious

เป็น method ที่ทำการย้ายเรคคอร์ดปัจจุบันไปยังตำแหน่งเริ่มต้น , ตำแหน่งสุดท้าย , ตำแหน่งถัดไป หรือ ตำแหน่งก่อนหน้าของเรคคอร์ดเซตที่กำหนดไว้

syntax

recordset.MoveFirst

recordset.MoveLast

recordset.MoveNext

recordset.MovePrevious

เมททอด Delete

เป็น method ที่ใช้ในการลบเรคคอร์ดปัจจุบันใน dynaset

syntax

recordset.Delete

เมททอด Edit

เป็น mehtod ที่ทำการเปิดเรคคอร์ดปัจจุบันในเรคคอร์ดเซตเพื่ออนุญาตให้ทำการเปลี่ยนแปลงข้อมูลในเรคคอร์ดเซตได้และจะทำการล็อคเพจปัจจุบัน (เป็นฐานข้อมูลไฟล์ที่เก็บค่าข้อมูลของเรคคอร์ด)ด้วย เพรสสิมิสติคล็อคกิ้ง (Perrimistic Locking) ซึ่งเป็นรูปแบบการล็อคเพจที่มีเรคคอร์ดที่สามารถเปลี่ยนแปลงได้แบบหนึ่ง โดยจะไม่อนุญาตให้คนอื่นเปลี่ยนแปลงแล้วจะปล่อยการล็อคเมื่อเราใช้ method อัพเดท (Update) ซึ่งจะกล่าวถึงใน method ถัดไป โดย method Edit จะทำการก็อบปี้เรคคอร์ดปัจจุบันลงบนบัฟเฟอร์แล้วให้เราทำการเปลี่ยนแปลงค่าในบัฟเฟอร์

syntax

recordset.Edit

ในกรณีที่ไม่ได้ใช้ด้าต้าคอนโทรลและใช้ method Find หรือ Move หรือ Close ขณะที่กำลังทำการ Edit หรือ AddNew จะยกเลิกการเปลี่ยนแปลงที่กระทำกับเรคคอร์ดก่อนหน้าที่จะใช้ method เหล่านั้นและไม่มีข้อผิดพลาดเกิดขึ้น

- การใช้ method Edit นี้จะเกิดข้อผิดพลาดขึ้นในกรณีดังต่อไปนี้
- ไม่มีเรคคอร์ดปัจจุบัน
- ฐานข้อมูลหรือเรคคอร์ดเซตถูกเปิดแบบอ่านได้อย่างเดียว (Read Only)
- ไม่มีฟิลด์ใดในเรคคอร์ดที่อนุญาตให้อัพเดทได้
- ฐานข้อมูลหรือเรคคอร์ดเซตถูกเปิดแบบเอ็กคลูซีพ (Exclusive) ซึ่งถูกใช้โดยคนอื่น

เมททอด Update

เมื่อเราทำการเปลี่ยนแปลงข้อมูลในเรคคอร์ด ซึ่งจะไม่ลงไปเขียนในตารางจริง ๆ แต่เรคคอร์ดที่กำลังแก้ไขเปลี่ยนแปลงอยู่นั้นจะถูกเก็บในบัฟเฟอร์ ซึ่งเมื่อเราใช้ method Update แล้วจะทำการเซฟการเปลี่ยนแปลงทั้งหมดลงในตารางหรือใน Dynaset

กฎการใช้

recordset.Update

โดยก่อนที่จะใช้ method Update นั้นเราจะต้องใช้เมททอด Edit ก่อน

6.2 การเปรียบเทียบคุณลักษณะบางประการของผลิตภัณฑ์

หลังจากได้ทดลองเขียนแอพพลิเคชันแล้ว ได้พบความแตกต่างถึงคุณลักษณะบางประการของตัวผลิตภัณฑ์ทั้งผลิตภัณฑ์ที่ใช้เป็นเครื่องมือในการสร้างแอพพลิเคชันและผลิตภัณฑ์ที่เป็นแบคเอ็นดีบีเอ็มเอส โดยได้เปรียบเทียบในด้านต่าง ๆ ดังต่อไปนี้

6.2.1 ผลิตภัณฑ์ที่เป็นแอพพลิเคชัน ในด้าน

6.2.1.1 ความง่ายในการใช้งาน ดังแสดงในตารางที่ 6.1

6.2.1.2 การเข้าถึงฐานข้อมูล ดังแสดงในตารางที่ 6.2

6.2.1.3 การเขียนกราฟและการทำรายงาน ดังแสดงในตารางที่ 6.3

6.2.1.4 สนับสนุนคุณลักษณะของไมโครซอฟต์วินโดวส์ ดังแสดงในตารางที่ 6.4

6.2.1.5 การเขียนโปรแกรม โดยคุณลักษณะการสนับสนุนการเขียนโปรแกรมแบบออบเจ็กต์โอเรียนเต็ดนั้น ในกรณีของวิชวลเบสิกไม่สามารถประกาศคลาสได้ ส่วนคุณลักษณะอื่น ๆ จะแสดงในตารางที่ 6.5

6.2.2 ผลิตภัณฑ์ที่เป็นแบคเอ็นดีบีเอ็มเอส ในด้าน

6.2.2.1 การจัดการทรานแซคชันของแบคเอ็นดีบีเอ็มเอส ดังแสดงในตารางที่ 6.6

6.2.2.2 โปรแกรมมิงอินเตอร์เฟสของแบคเอ็นดีบีเอ็มเอส ดังแสดงในตารางที่ 6.7

6.2.2.3 ดาต้าสตอเรจของแบคเอ็นดีบีเอ็มเอส ดังแสดงในตารางที่ 6.8

6.2.2.4 ดีบีเอ็มเอสอินทิกริตี้ของแบคเอ็นดีบีเอ็มเอส ดังแสดงในตารางที่ 6.9

ความง่ายในการใช้งาน	เอสคิวแอลวินโดวส์	วิชวลเบสิก
สร้าง Application โดยไม่ต้องเขียนโปรแกรม	✓	✓
การสร้างดาต้าเบสแอพพลิเคชัน สำหรับงานทางด้านดาต้าเบส เช่น คิวรี่ (Query), อัพเดท, อินเสิร์ท, การลบ โดยไม่ต้องเขียนโปรแกรม	✓	✓
สร้างหน้าจอได้หลายหน้าจอโดยไม่ต้องเขียนโปรแกรม	✓	✓
สร้างเมนูโดยไม่ต้องเขียนโปรแกรม	✓	✓
สร้างกราฟโดยไม่ต้องเขียนโปรแกรม	✓	✓
สร้างรายงานโดยไม่ต้องเขียนโปรแกรม	✓	✓
การขอความช่วยเหลือแบบคอนเท็กซ์เซนซิทีฟ	✓	✓

ออนไลน์ (context-sensitive on-line help)		
สร้าง Application ที่ประกอบด้วยระบบจดหมายอิเล็กทรอนิกส์ (E-mail) โดยไม่ต้องเขียนโปรแกรม	✓	✗

ตารางที่ 6.1 แสดงคุณลักษณะในด้านความง่ายในด้านการใช้งานของเครื่องมือที่ใช้ในการพัฒนาแอพพลิเคชัน

Database Access	เอสคิวแอลวินโดวส์	วิชวลเบสิก
สามารถเข้าถึง Database ได้	✓	✓
เข้าถึงโดยใช้ ODBC	✓	✓
สนับสนุนการทำไดนามิกเอสคิวแอล	✓	✓
สนับสนุนการทำสตอโพรซีเดอร์ (Store Procedure)	✓	✓

ตารางที่ 6.2 แสดงคุณลักษณะของเครื่องมือที่ใช้ในการพัฒนาแอพพลเคชันในด้านการเข้าถึงฐานข้อมูล

Graphing and Reporting	เอสคิวแอลวินโดวส์	วิชวลเบสิก
สามารถแสดงกราฟได้หลายรูปแบบ (Line, bar, Pie ฯลฯ)	✓	✓
สามารถเปลี่ยนรูปแบบกราฟขณะรันโปรแกรมได้	✓	✓
สามารถมีภาพกราฟิกในรายงานได้	✓	✓
Mailing label report	✓	✓
Form letter report	✓	✓
Reusable Report Templates	✓	✓
สามารถคำนวณได้ (Formula)	✓	✓

ตารางที่ 6.3 แสดงคุณลักษณะการเขียนกราฟและการทำรายงานของเครื่องมือที่ใช้ในการสร้างแอพพลิเคชัน

สนับสนุนคุณลักษณะของไมโครไมโครซอฟต์วินโดวส์	เอสคิวแอลวินโดวส์	วิชวลเบสิก
DDE	✓	✓
OLE	✓	✓
MDI	✓	✓

ตารางที่ 6.4 แสดงการสนับสนุนคุณลักษณะบางประการของไมโครซอฟต์วินโดวส์ของเครื่องมือที่ใช้ในการสร้างแอพพลิเคชัน

การเขียนโค้ด	เอสคิวแอลวินโดวส์	วิชวลเบสิก
ความสัมพันธ์ระหว่างสิ่งแวดล้อมในการออกแบบและการเขียนโค้ด (design and coding environments)	✓	✓
สามารถดูโค้ดในรายละเอียดทุกระดับได้	✓	✓
มีเครื่องช่วยในการเขียนโค้ด	✓	✓
สามารถเลือกจัดการโค้ดของแต่ละออบเจ็กต์หรือโค้ดทั้งหมดในแอพพลิเคชันก็ได้	✓	✓
การพิมพ์โค้ดออกมาเป็นเท็กซ์ไฟล์	✓	✓
การแปลแอพพลิเคชันเป็นภาษากลางระหว่างประเทศ	✓	✗

ตารางที่ 6.5 แสดงคุณลักษณะด้านการเขียนโปรแกรมของเครื่องมือที่ใช้ในการสร้างแอพพลิเคชัน

การจัดการทรานแซคชัน	SQLBase	Microsoft SQL Server
การล็อคระดับเรคคอร์ด	✓	✓

การล็อคระดับเพจ	✓	✓
การล็อคระดับตาราง	✓	✓
การทำให้ฐานข้อมูลอ่านได้อย่างเดียว	✓	✓

ตารางที่ 6.6 แสดงคุณลักษณะด้านการจัดการทรานแซคชันของแบคเอ็นดีบีเอ็มเอส

การโปรแกรม	SQLBase	Microsoft SQL Server
สนับสนุนการทำ Stored Procedure	✓	✓
จัดการการทำ Message และ Error	✗	✓
สนับสนุนการประมวลผล Record at a time	✗	✓
ผู้ใช้สามารถเรียก Store Procedure ที่ Remote Server	✗	✓

ตารางที่ 6.7 แสดงคุณลักษณะด้านโปรแกรมมิงอินเตอร์เฟสของแบคเอ็นดีบีเอ็มเอส

คุณลักษณะของดาตาสตอเรจ	SQLBase	Microsoft SQL Server
สนับสนุนการทำ Clustered Index	✗	✓
สนับสนุนการทำ hashed index	✗	✓
มี Tool สำหรับดู Performance Feature	✓	✓*

ตารางที่ 6.8 แสดงคุณลักษณะด้านดาต้าสตอเรจของแบคเอ็นดีบีเอ็มเอส

ดีบีเอ็มเอสอินทริกริตี	SQLBase	Microsoft SQL Server
Triggers	(ไม่ได้ Support)	✓

ตารางที่ 6.9 แสดงคุณลักษณะด้านดีบีเอ็มเอสอินทิกริตี้ของแบคเอ็นดีบีเอ็มเอส

บทที่ 7
บทวิจารณ์และบทสรุป

จากการที่ได้ศึกษาเครื่องมือที่ใช้พัฒนาแอพพลิเคชันของซอฟต์แวร์กุปตา พบว่าเครื่องมือนั้นสนับสนุนการออกแบบทั้งวิธีฟังก์ชันนอลและวิธีออบเจ็กต์โอเรียนเต็ด แต่ไม่มีเครื่องมือที่ช่วยในการออกแบบแอพพลิเคชัน จึงได้ศึกษาวิธีออกแบบแอพพลิเคชันโดยเลือกวิธีออบเจ็กต์โอเรียนเต็ดเพราะต้องการเปรียบเทียบวิธีการออกแบบเดิมที่เคยใช้อยู่ นอกจากนั้นยังเป็นการเรียนรู้เทคโนโลยีใหม่ๆ ด้วย วิธีที่เลือกมานี้คือ Object Lifecycles Modeling the world in states และได้ทดลองนำมาออกแบบ แอพพลิเคชันระบบงานห้องสมุด เมื่อพบว่าสามารถใช้งานได้จริง จึงได้ทดลองพัฒนาแอพพลิเคชันโดยใช้เครื่องมือตัวอื่น ซึ่งได้เลือกใช้ Visual Basic เวอร์ชัน 3.0 ผลปรากฏว่าก็สามารถใช้งานได้จริง ดังนั้นวิธีที่ได้เลือกใช้สามารถนำไปใช้อิมพลีเมนท์ระบบงานจริงได้ แต่ก็มีข้อจำกัดคือยึดติดกับระบบฐานข้อมูลแบบสัมพันธ์มากเกินไป

ทางด้านซอฟต์แวร์ ซอฟต์แวร์ที่ใช้ในการพัฒนาระบบงานประยุกต์คือ SQLWindows 5.0 ของบริษัทกุปตาและ Microsoft Visual Basic 3.0 เมื่อได้ใช้เครื่องมือทั้งสองแล้ว SQLWindows มีความสามารถกว่า Visual Basic ในหลาย ๆ ด้าน เช่น เป็นภาษา 4GL ทำให้ใช้ง่าย, มีเครื่องมือช่วยในการเขียนโปรแกรม ซึ่ง Visual Basic เองเป็นภาษา 3GL ทำให้การเรียนรู้และใช้งานยากกว่า (ในกรณีของผู้เริ่มต้น) แต่ถ้าผู้ใช้มีประสบการณ์แล้ว การพัฒนาด้วย Visual Basic ก็อาจคล่องกว่า ดังได้ทำหัวข้อการเปรียบเทียบในบทที่ 6 นอกจากนั้นเมื่อเปรียบเทียบกันทางด้านราคาแล้ว SQLWindows มีราคาแพงกว่า Visual Basic อยู่มาก ถ้าต้องการพัฒนาระบบงานเล็กๆ ไม่ใหญ่มาก ใช้ Visual Basic จะเหมาะสมกว่า

ทางด้านการทำงาน ในการศึกษาวิธีออกแบบโดยใช้วิธีออบเจ็กต์โอเรียนเต็ดนั้น แหล่งข้อมูลยังมีไม่มากเท่าที่ควร อาจเป็นเพราะยังเป็นเทคโนโลยีที่ใหม่อยู่ โดยวิธีที่ได้คัดเลือกมานั้นยังน้อยเกินไป และหนังสือที่เขียนออกมาก็ยังไม่ละเอียดเท่าที่ควร ต้องติดตามอ่านในเล่มอื่น ๆ ประกอบ

ทางด้านการติดตั้งซอฟต์แวร์ โดย SQLWindows 5.0 จะทำงานบน Microsoft Windows โดยเลือกติดตั้งระบบไคลเอ็นท์เซิร์ฟเวอร์โดยใช้เน็ตแวร์ เวอร์ชัน 3.11 ก็ไม่เป็นปัญหาแต่อย่างใด

เอกสารอ้างอิง

1. Perter J. Robinson , "Hierarchical Object Oriented Design" , Prentice Hall .
2. James Martin and James J. Odell, "Object-Oriented Analysis and Design", Prentice Hall.
3. Delatte-M.Hettz J.F. Muller, "HOOD Reference Manual 3.1", Prentice Hall.
4. William Gietz, "SQLWindows Programming", Gupta Corporation.
5. "SQLTalk/Windows User's Guide", Gupta Corporation.
6. "SQLBase SQLTalk Languate Reference Manual", Gupta Corporation.
7. Peter J. Robinson ," Object-Oriented Design", Prentice Hall.
8. Sally Shlaer and Stephen J. Mellor "Object Lifecycles Modeling the World in States", Yourdon Press.
9. Paul Harmon and Brian Sawyer , "Object Craft a Graphical Programming Tool for Object-Oriented Application", Addison Wesley.
10. Peter Coad and Edward Yourdon, " Object-Oriented Analysis", Addison Wesley.
11. Naphtali Rishe, "Database Design Fundamentals", Addison Wesley.
12. James Rumbaugh, Michael Blahs, William Premerlani, Ferderick Eddy, and William Lorensen, "Object-Oriented modeling and Design", Addison Wesley.
13. "ODBC 2.0 Programmer 's Reference and SDK Guide" Microsoft Press, 808 p, 1993.
14. พิษณุเพทางค์, "ตำราพรหมชาติ ฉบับชาวบ้าน", สมเจตการพิมพ์, 700 หน้า, 2531

กิตติกรรมประกาศ

การทำปริญญานิพนธ์ในครั้งนี้ ผู้จัดทำขอขอบคุณ

1. ผศ.ดร. ศุภมิตร จิตตะยโศธร ที่ให้คำแนะนำปรึกษาตลอดมา
2. พี่ตาและบริษัทจักรวาลที่มอบซอฟต์แวร์ของบริษัทกุปตาเทคโนโลยี ให้ใช้ในการทำปริญญานิพนธ์ในครั้งนี้
3. พี่ภุชงค์, พี่ทัก, พี่แอน และ พี่นก ที่ช่วยติดต่อซอฟต์แวร์จากบริษัทจักรวาลและคอยตอบปัญหาในการติดตั้งซอฟต์แวร์
4. คุณอัศวิน ซุปเปอร์ไวเซอร์ที่ภาควิชา
5. คุณสุวิชัย ที่ช่วยตอบปัญหาในการเขียนโปรแกรม
6. เพื่อน ๆ กลุ่มทำโปรเจ็คต์