สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

ฐานข้อมูลเชิงวัตถุและการประยุกต์ในระบบผู้เชี่ยวชาญ

Object-Oriented Databases and their Expert Systems' applications

นายสุรศักดิ์ ลีนะพงศ์พานิช
นายสุรศักดิ์ วัฒนยนต์กิจ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2542

เลขหมู่.....
เลขทะเบียน.....37072
วัน, เดือน, ปี.....30 ธ.ค. 2543

ลิขสิทธิ์ของสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่อนุญาตให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ฐานข้อมูลเชิงวัตถุและการประยุกต์ในระบบผู้เชี่ยวชาญ

Object-Oriented Databases and their Expert Systems' applications

โดย

นายสุรศักดิ์ ลีนะพงศ์พานิช

นายสุรศักดิ์ วัฒนยนต์กิจ

อาจารย์ที่ปรึกษา

รศ.ดร.ศุภมิตร จิตตยะโศธร

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2542

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ปริญญานิพนธ์ปีการศึกษา 2542

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ฐานข้อมูลเชิงวัตถุและการประยุกต์ในระบบผู้เชี่ยวชาญ

Object-Oriented Databases and their Expert Systems' applications

ผู้จัดทำ

1. นายสุรศักดิ์ ลีนะพงศ์พานิช รหัสประจำตัว 39014569
2. นายสุรศักดิ์ วัฒนยนต์กิจ รหัสประจำตัว 39014611

อาจารย์ที่ปรึกษา

(รศ.ดร.ศุภมิตร จิตตยะโศธร)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ฐานข้อมูลเชิงวัตถุและการประยุกต์ในระบบผู้เชี่ยวชาญ

นายสุรศักดิ์ ลีนะพงศ์พานิช 39014569
นายสุรศักดิ์ วัฒนยนต์กิจ 39014611
ปีการศึกษา 2542

บทคัดย่อ

จากการพัฒนาของเทคโนโลยีทางด้านระบบฐานข้อมูลและแนวคิดในแบบเชิงวัตถุได้มีความก้าวหน้ามากขึ้นเรื่อยๆ จนในปัจจุบันได้เริ่มมีผลิตภัณฑ์ในเชิงพาณิชย์ออกมาบ้างแล้ว และในอีกด้านหนึ่ง การวิจัยทางด้านระบบฐานข้อมูลอนุมานที่ถึงแม้ว่ายังไม่มีผลิตภัณฑ์ในเชิงพาณิชย์ แต่ก็มีความก้าวหน้าถึงขนาดที่มีการสร้างระบบต้นแบบขึ้นมามากมายหลายระบบ อีกทั้งยังได้เริ่มมีการพัฒนาที่รวมเอาระบบทั้งสองข้างต้นนี้เข้าไว้ด้วยกัน ทำให้โครงงานนี้เล็งเห็นประโยชน์ที่จะนำเอาเทคโนโลยีทั้งสองอย่างนี้มาประยุกต์เข้าด้วยกันและนำมาใช้ประโยชน์ในระบบผู้เชี่ยวชาญ โดยให้ระบบผู้เชี่ยวชาญใช้ประโยชน์จากฐานข้อมูลที่มีอยู่ ซึ่งอาจเป็นเป็นข้อมูลที่เป็นแบบชนิดข้อเท็จจริงหรือกฎก็ได้ เพื่อให้การใช้งานระบบผู้เชี่ยวชาญเป็นไปอย่างมีประสิทธิภาพมากขึ้นและเป็นการลดคำถามที่ระบบผู้เชี่ยวชาญจะถามผู้ใช้ลงให้เหลือเท่าที่จำเป็น

และในอีกส่วนหนึ่งของโครงงานนี้ที่ได้นำเสนอก็คือ ได้มีนำการเสนอรูปแบบวิธีการที่ใช้ในการติดต่อกันระหว่างฐานข้อมูลเชิงวัตถุและระบบผู้เชี่ยวชาญที่แทนความรู้ในรูปแบบของกฎ ซึ่งมีรูปแบบการแทนความรู้ที่แตกต่างกันด้วย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Object-Oriented Databases and their Expert Systems' applications

Surasak Leenapongpanich

Surasak Watthanayontkit

Assoc. Prof. Dr. Suphamit Chittayasothon Advisor

ABSTRACT

From the result of evolutions in database system technology and Object-Oriented concepts, in present, there are some commercial Object-Oriented database management system's products that have been arisen. On the other hand, in deductive database's research community, although no commercial product does exist, there are a lot of prototypes available now. And there seems a new trend exists from the emerging of both database technologies described above. This project is inspired by the idea of combining these technologies and utilizing them in expert systems by let them use data that have already stored in the database, which may be facts, or rules. For the purpose to archive higher usability of expert systems, and to reduce the questions that expert systems need to ask to the users.

This project also presents the approach and the solution for interfacing between Object-Oriented database and rule-based system which data representations are different.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้คงไม่สามารถสำเร็จสมบูรณ์ลงได้ หากไม่ได้รับการช่วยเหลือจากบุคคลหลายๆ ฝ่ายด้วยกัน ทั้งที่เกี่ยวข้องกับการทำงานของนักศึกษาผู้จัดทำโครงงานโดยตรง และทางอ้อม ทางนักศึกษาผู้จัดทำโครงงานขอขอบพระคุณบุคคลต่างๆ ต่อไปนี้ :

- รศ.ดร.ศุภมิตร จิตตะยโศธร ที่รับเป็นอาจารย์ที่ปรึกษาของวิชาโครงงาน ตลอดจนช่วยดูแลเอาใจใส่ แนะนำและให้คำปรึกษาเกี่ยวกับแนวทางการทำงานของโครงงานมาโดยตลอด (รวมทั้งช่วยเฉลยการบ้านบางข้อที่ฝากให้นักศึกษาไปคิดแต่คิดไม่ออก หรืออาจคิดออกแต่ไม่ยอมคิด)
- อาจารย์บัณฑิต พัสยา สำหรับความช่วยเหลือ และคำแนะนำในเรื่องต่างๆ ทั้งที่เกี่ยวข้องกับโครงงานและที่ไม่เกี่ยว (รวมทั้งการรักษาน้ำใจ โดยการแนะนำโดยสุภาพว่าให้แต่งกายให้ดีกว่าที่เป็นอยู่ในยามที่นักศึกษาผู้ทำโครงงานแต่งกายในชุดที่ไม่มีใครคาดคิดว่าจะมีคนกล้าแต่งเข้ามาเดินในภาควิชาได้)
- บิดา มารดา ผู้บังเกิดเกล้า ที่อุตส่าห์ชุบเลี้ยง สั่งสอน และดูแลจนทำให้มีวันนี้ รวมทั้งเอื้อเฟื้ออิสระในการเลือกทางเดินของชีวิตให้แก่นักศึกษาผู้ทำโครงงาน (ไม่ว่าพวกท่านจะเต็มใจหรือไม่ก็ตาม)
- เจ้าหน้าที่ และบุคลากรที่เกี่ยวข้องในส่วนของห้องสมุดทั้งในส่วนของห้องสมุดคณะวิศวกรรมศาสตร์ สำนักหอสมุดกลาง ของสถาบันที่ทำให้นักศึกษาผู้จัดทำโครงงานรับรู้ว่ายังมีอะไรๆ ในห้องสมุดของสถาบันตัวเองให้ใช้ประโยชน์อีกมาก (และจะดียิ่งกว่านี้ ถ้าอัตราค่าบริการในการทำสำเนาบทความของ IEEE จาก CD-ROM เป็นราคาหน้าละ 1 บาทตั้งแต่ภาคการศึกษาตอนต้น)
- ผู้บริหารห้องสมุดสถาบันเทคโนโลยีแห่งเอเชีย (AIT) ที่เปิดให้บุคคลที่แม้จะเป็นนักศึกษาของสถาบันการศึกษาอื่นๆ เข้าใช้บริการได้โดยไม่เสียค่าธรรมเนียมใดๆ (แต่น่าเสียดายที่อยู่ไกลไปหน่อยเลยเสียค่ารถแพงๆแทน)
- Alan Kay และ Xerox Corp. สำหรับ Smalltalk : ต้นแบบของแนวความคิด Object-Oriented ในเชิงพาณิชย์และเติบโตแพร่หลายมาจนถึงปัจจุบัน (มีหัวข้อโครงงานให้ทำ)
- ผู้ก่อตั้ง Domain WWW ที่จัดทำขึ้นเพื่อส่วนรวมโดยไม่แสวงผลกำไร ทำให้มีแหล่งรวมข้อมูลที่ทุกคนมีสิทธิ์ได้ใช้โดยเท่าเทียมกัน และเป็นแหล่งข้อมูลที่นักศึกษาผู้จัดทำโครงงานได้ใช้ในการหาข้อมูลที่เกี่ยวข้องในการทำโครงงาน (รวมทั้งที่ไม่เกี่ยวข้องแต่อยากหาด้วย)
- เพื่อนๆ ทุกๆ คนทั้งที่ทำงานอยู่ในห้องวิจัยเดียวกัน, ภาควิชาเดียวกัน และคนละภาควิชา รวมถึงรุ่นน้องที่คอยถามไถ่ถึงความคืบหน้าของโครงงานที่ทำอยู่เสมอๆ (ทั้งที่เป็นการถามไถ่ด้วยความห่วงใยจากใจจริง และที่เป็นการถามไถ่ตามมารยาท)
- คนอื่นๆ ที่ยังเหลืออีกทุกๆ คนที่มีส่วนเกี่ยวข้องด้วย แต่ยังไม่ได้อ้างอิงถึงในส่วนข้างต้น (เนื่องจากผู้จัดทำโครงงานนึกไม่ออกแล้ว)

สุรศักดิ์ ลีนะพงศ์พานิช
สุรศักดิ์ วัฒนยนต์กิจ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญภาพและตาราง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มา

จากการพัฒนาทางด้านเทคโนโลยีทางด้านฐานข้อมูลที่ได้พัฒนาขึ้นมาเรื่อยๆ จนในปัจจุบันได้มีผลิตภัณฑ์ในเชิงพาณิชย์ของระบบจัดการฐานข้อมูล (Database Management System: DBMS) เกิดขึ้นและใช้กันอย่างแพร่หลาย ซึ่งระบบจัดการฐานข้อมูลเหล่านี้ได้แบ่งเบาภาระหน้าที่ของผู้ที่มีหน้าที่ในการพัฒนาระบบสารสนเทศอย่างมีประสิทธิภาพ ตัวอย่างของความสามารถของระบบจัดการข้อมูลดังกล่าวนี้ ก็ได้แก่ ความสามารถในการจัดการเรื่องความซ้ำซ้อนของข้อมูล (Redundancy), ความสามารถในการดูแลความสอดคล้องกันของข้อมูล (Consistency), ความสามารถในการจัดการระเบียบข้อบังคับที่ใช้ในการจัดเก็บข้อมูล (Integrity), ความสามารถในการจัดการเรื่องของการจัดสรรข้อมูลให้สามารถใช้งานโดยโปรแกรมหลายๆ ตัวในเวลาเดียวกันได้ (Concurrency), และความสามารถในการจัดการเรื่องมาตรการด้านความปลอดภัยในการใช้ข้อมูลที่อยู่ในฐานข้อมูล (Security)

ไม่เพียงแต่ในเรื่องดังข้างต้นนี้เท่านั้นที่ได้มีการพัฒนา ในส่วนของแนวทางที่ใช้ในการจัดเก็บข้อมูลในฐานข้อมูลก็มีการพัฒนาและคิดค้นขึ้นมาอยู่เรื่อยๆ เช่นกัน ตัวอย่างหนึ่งของแนวทางเหล่านี้ได้แก่ แนวทางที่เป็นการนำเอาแนวความคิดเชิงวัตถุ (Object-Oriented Concept) ซึ่งมีจุดกำเนิดมาจากในส่วนของวิธีการพัฒนาและเขียนโปรแกรมก่อน ต่อมาก็มีการนำมาประยุกต์ใช้ในเรื่องของปัญญาประดิษฐ์ (Artificial Intelligence) จนในที่สุดก็เริ่มมีการนำมาใช้กับระบบฐานข้อมูล ซึ่งทำให้ระบบจัดการฐานข้อมูลที่ใช้วิธีการเก็บข้อมูลแบบนี้สามารถจัดการกับข้อมูลที่มีลักษณะที่ซับซ้อนได้ดีกว่าแบบเดิมซึ่งเป็นแบบ Relational สำหรับอีกแนวทางหนึ่งก็คือการนำเอาแนวทางที่ใช้ในระบบผู้เชี่ยวชาญเพื่อทำให้ระบบฐานข้อมูลมีความสามารถในการเก็บข้อมูลที่อยู่ในรูปแบบของกฎ (Rule) ได้ เป็นผลให้ระบบฐานข้อมูลชนิดนี้มีความสามารถในการอนุมาน (Deduction) ข้อมูลที่เป็นข้อเท็จจริงใหม่ๆ ได้ โดยไม่จำเป็นต้องมีการเก็บข้อมูลตัวนั้นจริงในฐานข้อมูลซึ่งในแนวทางนี้ก็ได้มีการพัฒนาขึ้นมาจนในปัจจุบันได้มีระบบที่เป็นต้นแบบอยู่มากมาย

นอกจากสองแนวทางหนึ่งดังข้างต้นแล้ว อีกแนวทางหนึ่งซึ่งเป็นแนวทางที่พยายามพัฒนาระบบฐานข้อมูลที่รวมเอาแนวทางทั้งสองเข้าไว้ด้วยกันก็มีการพัฒนาขึ้นด้วยเช่นกัน และก็ได้มีระบบต้นแบบให้เห็นกันแล้วด้วย ซึ่งสำหรับในโครงงานนี้จะนำรูปแบบของระบบจัดการฐานข้อมูลดังกล่าวมาประยุกต์ใช้ในระบบผู้เชี่ยวชาญ โดยมีวัตถุประสงค์เพื่อพยายามพัฒนาระบบสถาปัตยกรรมของระบบผู้เชี่ยวชาญที่มีประสิทธิภาพและความสามารถในการนำไปประยุกต์ในการใช้งานจริงได้อย่างกว้างขวางขึ้น โดยพัฒนาส่วนของระบบผู้เชี่ยวชาญให้สามารถใช้งานข้อมูลข้อมูลที่อยู่ในฐานข้อมูลซึ่งในโครงงานนี้ได้ทำการพัฒนาขึ้นให้เป็นแบบเชิงวัตถุที่มีความสามารถในการอนุมานได้ด้วย เป็นการขยายความสามารถของระบบผู้เชี่ยวชาญซึ่งจากเดิมที่สามารถเก็บข้อมูลได้อย่างจำกัดและทำงานได้เพียงแค่ในหน่วยความจำหลักเท่านั้น อีกทั้งยังขาดความสามารถในการเก็บข้อมูลต่างๆ ไว้อย่างถาวร ทำให้ในการใช้งานระบบผู้เชี่ยวชาญบางครั้งค่อนข้างซ้ำซากและต้องทำการถามผู้ใช้ตามขั้นตอนต่างๆ ที่ซ้ำๆ กันซึ่งจริงๆ แล้วเป็นสิ่งที่ไม่จำเป็นเลย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สำหรับการพัฒนาระบบของโครงงานนี้จะเป็นการสร้างระบบสถาปัตยกรรมที่เป็นติดต่อกันระหว่างระบบฐานข้อมูลเชิงวัตถุ และระบบผู้เชี่ยวชาญแบบทำงานด้วยกฎ (Rule-based) ที่มีรูปแบบของข้อมูลที่แตกต่างกัน ซึ่งในเนื้อหาของปริญญานิพนธ์ฉบับนี้ก็ได้นำเสนอแนวทางหนึ่งที่ใช้ในการจัดการกับปัญหานี้ด้วย

1.2 วัตถุประสงค์ของโครงงาน

1.2.1. ศึกษาถึงทฤษฎี หลักการ และแนวทางการใช้งานฐานข้อมูลเชิงวัตถุ โดยเน้นการศึกษาไปที่ผลิตภัณฑ์ Jasmine ซึ่งเป็นโปรแกรมระบบจัดการฐานข้อมูลเชิงวัตถุ

1.2.2. ศึกษาถึงรูปแบบสถาปัตยกรรมที่เป็นการนำเอาระบบฐานข้อมูลมาประยุกต์ใช้ประโยชน์ในการทำงานของระบบผู้เชี่ยวชาญ และพยายามนำเสนอรูปแบบสถาปัตยกรรมใหม่ที่สามารถลดปัญหาหรือข้อจำกัดที่มีอยู่ในระบบสถาปัตยกรรมแบบเดิมๆ ได้

1.2.3 ศึกษาถึงความเป็นไปได้ในการสร้างในส่วนของกฎให้สามารถเก็บอยู่ในระบบจัดการฐานข้อมูลเชิงวัตถุได้ โดยพยายามใช้ประโยชน์จากคุณสมบัติที่มีอยู่ของระบบฐานข้อมูลเชิงวัตถุ

1.2.4. นำเสนอวิธีการที่ใช้ในการแก้ปัญหาที่เกิดขึ้นการติดต่อกันระหว่างระบบผู้เชี่ยวแบบทำงานด้วยกฎกับระบบฐานข้อมูลเชิงวัตถุซึ่งมีรูปแบบข้อมูลที่แตกต่างกัน

1.2.5. ศึกษาถึงรูปแบบของกฎต่างๆ ที่มีอยู่ในระบบสถาปัตยกรรมที่เป็นแบบ Client-Server ว่ากฎใดควรจะเก็บไว้ที่ไหน ในกรณีที่เราสามารถเก็บข้อมูลประเภทกฎได้ทั้งในส่วนของ Client และส่วนของ Server

1.3 ขอบเขตของโครงงาน

โครงงานจะเน้นในด้านการศึกษาและออกแบบรูปแบบสถาปัตยกรรมที่ใช้ในการติดต่อกันระหว่างระบบฐานข้อมูลและระบบผู้เชี่ยวชาญ โดยระบบจัดการฐานข้อมูลที่ทางโครงงานนี้ได้นำมาใช้จะเป็นระบบจัดการฐานข้อมูลเชิงวัตถุ คือ Jasmine และระบบผู้เชี่ยวชาญที่ได้ทำการใช้จะเป็นระบบผู้เชี่ยวชาญแบบที่เป็นเปลือกระบบผู้เชี่ยวชาญแบบทำงานด้วยกฎ คือ Knowledge Pro และจะไม่เน้นในส่วนของการพัฒนาระบบเพื่อนำไปใช้งานจริง แต่จะทำการสร้างเพียงแค่ระบบต้นแบบเพื่อทดสอบว่าระบบที่ได้ทำการศึกษาและทำการออกแบบสามารถใช้งานได้จริง

1.4 วิธีการดำเนินงาน

โครงงานนี้จะเริ่มจากการศึกษาทฤษฎีพื้นฐานต่างๆที่เกี่ยวข้องกับโครงงาน ซึ่งแบ่งออกเป็นสองส่วน คือ ระบบฐานข้อมูลแบบต่างๆ ซึ่งในรายละเอียดของบทจะกล่าวถึงรูปแบบฐานข้อมูลแบบต่างๆ ที่เกี่ยวข้องกับโครงงาน นั่นคือ ระบบฐานข้อมูลเชิงวัตถุ ระบบฐานข้อมูลอนุมาน และระบบฐานข้อมูลอนุมานเชิงวัตถุ โดยในแต่ละส่วน จะมีการอธิบายถึงหลักการคร่าวๆ โครงสร้างและสถาปัตยกรรม รวมทั้งอาจแสดงถึงตัวอย่างของที่เกี่ยวข้องกับระบบฐานข้อมูลตัวนั้นๆ พอสังเขป ในเนื้อหาของบทที่ 2 สำหรับในส่วนของระบบผู้เชี่ยวชาญนั้นจะมีรายละเอียดของระบบผู้เชี่ยวชาญซึ่งก็คือ ส่วนประกอบ รูปแบบของระบบผู้เชี่ยวแบบต่างๆ และข้อจำกัดในการทำงานของระบบผู้เชี่ยวชาญและวิธีการที่ใช้ในการแก้ปัญหา ตลอดจนตัวอย่างของระบบที่เป็นการนำเอาระบบผู้เชี่ยวชาญมาใช้งานร่วมกันกับระบบจัดการฐานข้อมูล จะมีอธิบายอยู่ในบทที่ 3

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ในบทที่ 4 จะเป็นการนำเสนอถึงสถาปัตยกรรมของระบบที่ทางโครงงานได้นำเสนอ รูปแบบการติดต่อระหว่างฐานข้อมูลและระบบผู้เชี่ยวชาญรวมทั้งวิธีการแก้ปัญหาที่เกิดขึ้นในการติดต่อจะเป็นรายละเอียดในบทที่ 5 จากนั้นจะเป็นการนำแนวคิดที่ได้ออกแบบไว้นั้นมาทำการพัฒนาขึ้นเป็นระบบต้นแบบโดยการพิจารณาถึงคุณสมบัติต่างๆ ที่ระบบฐานข้อมูลมีอยู่และนำมาใช้ประโยชน์ และอาจมีการเขียนโปรแกรมเพิ่มเติมเพื่อทำหน้าที่บางอย่าง ในส่วนของบทที่ 7 และบทที่ 8 จะเป็นการทดสอบระบบที่ได้ทำการออกแบบและสร้างขึ้นมาและสรุปวิจารณ์รูปแบบสถาปัตยกรรมที่ได้สร้างขึ้นมา รวมทั้งเสนอแนวทางจะพัฒนาต่อ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 2
ระบบฐานข้อมูลแบบต่างๆ

1.1. บทนำ

จากการวิจัยและพัฒนาทางด้านเทคโนโลยีทางด้านระบบฐานข้อมูลแบบ Relational ที่ได้มีการพัฒนามาช้านาน จนเริ่มมีผลิตภัณฑ์ที่พัฒนาขึ้นโดยมีวัตถุประสงค์เพื่อการใช้งานในเชิงพาณิชย์ออกมากันอย่างแพร่หลาย และมีการพัฒนาต่อมาเรื่อยๆ อย่างสม่ำเสมอ จนในปัจจุบัน ระบบจัดการฐานข้อมูลแบบ Relational (Relational Database Management System : RDBMS) ได้มีการใช้งานกันและเป็นที่รู้จักกันอย่างแพร่หลาย และการพัฒนาเพิ่มเติมต่อมาเรื่อยๆ จนในปัจจุบัน ผลิตภัณฑ์ในส่วนของระบบการจัดการฐานข้อมูลเหล่านี้ได้มีความสามารถต่างๆ ที่เป็นการเอื้อประโยชน์ในการพัฒนาระบบสารสนเทศขนาดใหญ่ๆ มากมายหลายประการและมีความน่าเชื่อถือ (Reliability) ในการทำงานที่สูงด้วย ตัวอย่างของความสามารถของระบบจัดการฐานข้อมูลที่มีอยู่ในปัจจุบันก็ได้แก่ ความสามารถในการจัดการเรื่องของความซ้ำซ้อนของข้อมูล (Redundancy) ความสามารถในการหลีกเลี่ยงและจัดการในเรื่องของความขัดแย้งกันของข้อมูล (Consistency) ความสามารถในการเรียกใช้ข้อมูลและการทำงานร่วมกันของโปรแกรมประยุกต์ใช้งานหลายๆ โปรแกรมในเวลาเดียวกัน (Concurrency) ความสามารถในการจัดการเพื่อรักษาความถูกต้องของตัวข้อมูล (Integrity) และความสามารถในการจัดตั้งมาตรการในการใช้งานต่างๆ เพื่อเหตุผลทางด้านความปลอดภัย (Security)

แต่ข้อจำกัดอย่างหนึ่งของระบบจัดการฐานข้อมูลส่วนใหญ่ในปัจจุบันที่เป็นแบบ Relational นี้ ก็คือความสามารถในการจัดการข้อมูลที่มีลักษณะที่ซับซ้อนซึ่งทางตัวแบบจำลองข้อมูล (Data model) ที่ระบบใช้อยู่นั้นออกแบบมาเพื่อให้สามารถจัดการจัดข้อมูลที่มีลักษณะง่ายๆ ไม่ซับซ้อนซึ่งหากเราต้องทำการเก็บข้อมูลที่มีลักษณะที่ซับซ้อนขึ้นมาแล้ว ในส่วนของทางระบบฐานข้อมูลก็จะต้องใการออกแบบโครงสร้างการเก็บข้อมูลที่ซับซ้อนขึ้นตามไปด้วยและตัวแบบจำลองข้อมูลเองก็ไม่ได้ออกแบบมาให้มีโครงสร้างหรือกลไกใดๆ เพื่อรองรับหรือจัดการในด้านความซับซ้อนเหล่านี้โดยตรง ทำให้ทางผู้พัฒนาและผู้ที่มีหน้าที่ดูแลในส่วนของฐานข้อมูลต้องมาจัดการเอง ทำให้ก่อให้เกิดความผิดพลาดต่างๆ ได้ง่าย และรูปแบบการเก็บข้อมูลต่างๆ ก็ยังไม่สามารถเก็บข้อมูลที่อยู่ในรูปแบบของกฎ (Rule) ได้

จากข้อจำกัดต่างๆ ที่ได้อธิบายมาแล้วข้างต้นนี้ ทำให้แนวทางการวิจัยในด้านของระบบฐานข้อมูลในระยะเวลาต่อมา ได้มุ่งเน้นไปในส่วนที่ใช้ในการจัดการข้อมูลที่ซับซ้อน และความสามารถในการอนุมานกฎ (Deduction) ซึ่งจากแนวทางในการวิจัยเหล่านี้ ก็ได้เกิดแนวความคิดในเรื่องของระบบฐานข้อมูลที่เป็นแบบเชิงวัตถุ (Object-Oriented Database) และระบบฐานข้อมูลอนุมาน (Deductive Database) ซึ่งในเนื้อหาของบทนี้จะประกอบด้วยเนื้อหาในส่วนระบบฐานข้อมูลทั้งสอง โดยในหัวข้อต่อไป จะอธิบายถึงระบบฐานข้อมูลอนุมานก่อน . จากนั้นในหัวข้อต่อๆ ไป จะอธิบายถึงแนวความคิดเชิงวัตถุและการมาประยุกต์ใช้กับระบบฐานข้อมูล และสำหรับในลำดับสุดท้ายที่จะอธิบายในบทนี้ก็คือ แนวความคิดของระบบฐานข้อมูลที่มีการนำเอาแนวคิดของระบบฐานข้อมูลทั้งสองอย่างข้างต้น คือระบบฐานข้อมูลเชิงวัตถุ และระบบฐานข้อมูลอนุมานมารวมกัน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ซึ่งระบบฐานข้อมูลที่ได้มาใหม่นี้ก็มีชื่อเรียกว่าระบบฐานข้อมูลอนุมานเชิงวัตถุ (Deductive Object-Oriented Database)

2.2. ระบบฐานข้อมูลอนุมาน

ระบบฐานข้อมูลอนุมานเป็นระบบที่พยายามขยายความสามารถของเทคโนโลยีทางด้านฐานข้อมูล โดยการพยายามสร้างระบบที่มีความสามารถทั้งในส่วนที่เป็นลักษณะของฐานความรู้ (knowledge-base) และส่วนที่เป็นลักษณะของระบบการจัดการฐานข้อมูล ทำให้เราได้ระบบฐานข้อมูลมีความสามารถทางด้านการอนุมานหาข้อเท็จจริง (Fact) ใหม่ๆ จากข้อเท็จจริงที่มีการเก็บไว้ในส่วนของฐานข้อมูลซึ่งอาจมีการเก็บข้อมูลเหล่านั้นแยกไว้ต่างหากโดยไม่เกี่ยวข้องกับส่วนที่ใช้ในการทำ inference ซึ่งจะหมายถึงกระบวนการที่ใช้ในการหาข้อเท็จจริง โดยมีหลักการมาจากทฤษฎีทางด้านตรรกศาสตร์ (mathematical logic) และระบบฐานข้อมูลแบบนี้ก็จะใช้ทฤษฎีตรรกศาสตร์มาประยุกต์ใช้ในส่วนของการเก็บข้อมูลต่างๆ ด้วย ข้อมูลที่เก็บเหล่านี้ก็ได้แก่ ข้อเท็จจริง, กฎ, ข้อบังคับของข้อมูล (constraint) ซึ่งก็จะมีลักษณะที่คล้ายคลึงกับแบบจำลองข้อมูลแบบ Relational ที่สร้างขึ้นมาจากทฤษฎีทางด้านคณิตศาสตร์ด้วย ทำให้เป็นเกิดเป็นแนวทางการพัฒนาที่พยายามสร้างฐานข้อมูลที่มีความสามารถในการอนุมานขึ้นดังที่ได้อธิบายไปแล้ว

2.2.1. โครงสร้างของระบบฐานข้อมูลอนุมาน

ระบบฐานข้อมูลอนุมานสามารถพิจารณาแยกออกเป็น 2 ส่วน คือ

- intensional database ประกอบด้วย ส่วนของฐานความรู้ที่จะเก็บในส่วนของกฎทั่วไป (general rule) การอ้างสิทธิ รูปแบบลักษณะของข้อมูล และฐานความรู้ผู้เชี่ยวชาญ (expert knowledge) สำหรับในส่วนของฐานข้อมูลจะประกอบด้วยรูปแบบโครงสร้างของข้อมูล ข้อบังคับที่เกี่ยวข้องกับความถูกต้องของข้อมูล (integrity constraint) และรูปแบบของ view ที่จะเก็บไว้ในลักษณะต่างๆ ซึ่งจะเห็นได้ว่าสถานะของระบบในส่วนนี้จะไม่ค่อยเปลี่ยนแปลงมากนัก และค่อนข้างมีเสถียรภาพ
- extensional database คือ ส่วนของฐานข้อมูลที่เก็บข้อมูลในส่วนที่เราสนใจของระบบในปัจจุบัน ซึ่งจะมีการเปลี่ยนแปลงตลอดเวลา ข้อมูลต่างๆเหล่านี้ ได้แก่ ข้อมูลที่มาจากการเพิ่ม ลบ และแก้ไข จากตัวโปรแกรมที่เรียกใช้งานระบบฐานข้อมูล

2.2.2. รูปแบบสถาปัตยกรรม

ในการสร้างระบบฐานข้อมูลอนุมาน สิ่งแรกที่ต้องทำคือสร้างรูปแบบการเชื่อมต่อ (interface) ระหว่างระบบฐานข้อมูลกับภาษาที่ใช้ในการจัดการฐานข้อมูลซึ่งในที่นี้ก็คือ ภาษาประเภท logic programming โดยมีตัวประมวลผลทางภาษา (language processor) ทำหน้าที่เข้าถึงข้อมูลในส่วนที่เราต้องการ ซึ่งตัวประมวลผลทางภาษาอาจจะใช้วิธีการแปลงรูปแบบคำสั่งที่ส่งเข้ามาให้อยู่ในรูปแบบของภาษาที่ใช้จัดการฐานข้อมูล เช่น SQL ในกรณีที่สถาปัตยกรรมที่เราสร้างขึ้นมานั้นเป็นการสร้างระบบๆ หนึ่งขึ้นมาครอบระบบจัดการฐานข้อมูลที่มีอยู่เดิม เพื่อค้นหาข้อเท็จจริงนั้นในฐานข้อมูลต่อไป รูปแบบการแยกระบบจัดการฐานข้อมูลกับระบบผู้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เชื่อมชาญนี้เราเรียกว่า การรวมกันแบบหลวม (loosely coupled) ซึ่งสถาปัตยกรรมแบบนี้จะทำงานได้ดีถ้าทางฝั่งระบบผู้เชี่ยวชาญมีความต้องการข้อมูลจากฐานข้อมูลไม่บ่อยครั้งนัก แต่ถ้าหากทางฝั่งระบบผู้เชี่ยวชาญมีความต้องการข้อมูลจากฐานข้อมูลบ่อยจะทำให้การทำงานของระบบโดยรวมช้าลง

สำหรับรูปแบบสถาปัตยกรรมอีกแบบหนึ่งที่ได้มีการพัฒนาขึ้นเพื่อแก้ไขข้อเสียของรูปแบบสถาปัตยกรรมข้างต้น เป็นรูปแบบสถาปัตยกรรมที่ได้มีการสร้างให้กลไกของการเข้าถึงข้อมูลในระบบฐานข้อมูลกับกลไกของการทำ inference มีการทำงานร่วมกันอย่างมีประสิทธิภาพและเรียกสถาปัตยกรรมนี้ว่า การรวมกับอย่างเหนียวแน่น (tight coupling) โดยการชี้ครรชนีและการเข้าถึงการทำงานย่อยของระบบฐานข้อมูลควรเป็นรูปแบบที่เป็นการเข้าถึงข้อมูลโดยตรง โดยไม่ต้องผ่านตัวกลางที่ทำหน้าที่ในการแปลงในการสืบค้นของระบบฐานข้อมูล อย่างไรก็ตามระบบจัดการฐานข้อมูลและระบบผู้เชี่ยวชาญก็ยังคงมีความเป็นอิสระอยู่ ยกเว้นการทำงานที่ประสานกันในระดับต่ำ

และเมื่อเข้าสู่การพัฒนาในรูปแบบต่อมา จะมีการรวมกันของระบบจนแยกไม่ออก นั่นคือระบบฐานข้อมูลและระบบการโปรแกรมเชิงตรรกไม่มีความเป็นอิสระต่อกัน ดังเช่น ภาษา logic-base ซึ่งทั้งการเชื่อมต่อและระบบต้องถูกออกแบบมาจากระบบฐานข้อมูลอนุมาน

2.2.3. Datalog

Datalog คือ ภาษาโปรแกรมที่คิดค้นขึ้นมาเพื่อใช้ในการจัดการกับระบบฐานข้อมูลอนุมาน โดยแนวคิดของภาษาได้มาจากแนวคิดของภาษา Prolog มาทำการเปลี่ยนแปลงและประยุกต์ใช้เพื่อให้เหมาะสมกับทฤษฎีของระบบฐานข้อมูลแบบ Relational ให้ลักษณะของภาษา Datalog คล้ายกับภาษา Prolog มาก นั่นคือเป็นภาษาที่มีรูปแบบการเขียนโปรแกรมเป็นในลักษณะของ Declarative แต่ที่แตกต่างกันก็คือ ในภาษา Datalog จะตัดในส่วนของโครงสร้างที่มีลักษณะการทำงานเป็นแบบ Procedural ไป เช่น การใช้ cut เป็นต้น และข้อแตกต่างอีกส่วนที่ต่างจากภาษา Prolog ก็คือ การแยกส่วนของการทำงานที่เกี่ยวข้องกับการทำ Inference ออกจากส่วนที่ใช้ในการเก็บข้อมูล ซึ่งก็เป็นข้อควรระวังที่ผู้ที่ทำการพัฒนาต้องระวัง และอีกอย่างก็คือลำดับของ rule ที่เก็บอยู่ในฐานข้อมูลที่เป็นภาษา Datalog จะไม่มีความสำคัญในการทำงานของโปรแกรมเหมือนอย่างในภาษา Prolog ซึ่งจากลักษณะต่างๆ ที่ได้อธิบายมานี้ทำให้ภาษา Datalog เป็นภาษาที่เขียนได้ง่ายกว่าภาษา Prolog แต่การพัฒนาตัวประมวลผลของภาษานั้นจะทำได้ยากกว่า ต่อไปจะอธิบายถึงหลักสำคัญที่จะต้องนำไปใช้ใน Datalog

สำหรับรายละเอียดของภาษา Datalog นี้ จะลักษณะที่คล้ายกับภาษา Prolog มากเนื่องจากมีพื้นฐานมาจากทฤษฎีด้านตรรกศาสตร์เหมือนกัน คือในโครงสร้างของภาษาจะมีส่วนประกอบดังนี้

- ค่าคงที่ (Constant) เป็นค่าพื้นฐานที่สุดที่จะมีอยู่ในของฐานข้อมูล เป็นการประกอบกันของกลุ่มอักขระของภาษาอังกฤษและเครื่องหมายบางชนิดรวมทั้งเครื่องหมายขีดเส้นใต้ (underscore : _) และตัวเลข โดยที่อักษรตัวแรกของกลุ่มอักขระต้องเขียนขึ้นต้นด้วยอักษรตัวเล็กหรือตัวเลข ตัวอย่างเช่น

green, 20, january, 24.5, william_shakespeare

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- ตัวแปร (Variable) ใช้สำหรับการอ้างอิงถึงค่าคงที่ โดยลักษณะของตัวแปรจะต้องเป็นกลุ่มอักขระที่ขึ้นต้นด้วยอักษรตัวใหญ่ หรือเป็นสัญลักษณ์ตัวขีดเส้นใต้ก็ถือว่าเป็นรูปแบบหนึ่งของตัวแปรเช่นกัน โดยจะใช้ในกรณีที่อ้างอิงถึงค่าใดๆ ก็ได้ที่เราไม่สนใจ ตัวอย่างเช่น

X, Person, Department

สำหรับในส่วนของค่าคงที่และตัวแปรนี้อาจเรียกรวมกันได้ว่าเป็นเทอม (Term) ซึ่งเทอมอาจจะเขียนอยู่ในรูปของฟังก์ชันที่มีค่าเทอมเหล่านี้เป็นพารามิเตอร์ได้ด้วย (หมายความว่า ถ้า f เป็นฟังก์ชันชนิด n-ary และ t1, t2, t3, ... เป็นเทอมเช่นเดียวกัน จะได้ f(t1, t2, t3, ...) เป็นเทอมด้วย) แต่สำหรับรูปแบบการใช้สัญลักษณ์ฟังก์ชันแทนถึงเทอมนี้จะไม่มีอยู่ในภาษา Datalog แบบพื้นฐาน แต่จะมีในภาษา Datalog ที่เป็นการเพิ่มส่วนขยายให้มีความสามารถในการใช้สัญลักษณ์ฟังก์ชัน

- พรีดิเคต (Predicate) เป็นรูปแบบที่แสดงถึงความความสัมพันธ์กันระหว่างเทอม

works_in(donald, marketing), author_of(hamlet, shakespeare)
mortal(Human), employee(Name, Department, Telephone#)

เราสามารถเรียกรูปแบบพรีดิเคตที่เป็นลักษณะแบบ P(t1, t2, t3, ...) ที่แสดงดังข้างต้นนี้ได้อีกอย่างว่า อะตอม (Atom)

- ตัวเชื่อม (Connective) ทำหน้าที่ในการรวมนิพจน์ของอะตอมที่ต้องการเชื่อมเข้าไว้ด้วยกัน เพื่อสร้างเป็นโครงสร้างประโยคที่ใหญ่ขึ้นได้ โดยตัวเชื่อมเหล่านี้ได้แก่ AND, IF และ NOT แต่ละตัวจะใช้สัญลักษณ์แทนดังนี้

  " , " แทนตัวเชื่อม AND

  " :- " แทนตัวเชื่อม IF

  " ¬ " แทนตัวเชื่อม NOT (ไม่มีในภาษา Datalog แบบพื้นฐาน)

- อนุประโยค (clause) แบ่งออกได้เป็น 2 ชนิด คือ ชนิดแบบข้อเท็จจริง และชนิดแบบกฎ สำหรับรูปแบบที่เป็นชนิดกฎ มีลักษณะเขียนได้ดังนี้

$$P :- Q_1, Q_2, \ldots, Q_n$$

เมื่อ P, Q_1, Q_2, ..., Q_n แต่ละตัวจะแทนถึงอะตอม ซึ่งอนุประโยคข้างบนสามารถอ่านได้ว่า " P ถ้า Q_1 และ Q_2 และ ... และ Q_n" และในส่วนข้างต้นนี้ เราจะเรียกในส่วนของ P ว่าเป็นส่วนหัว (Head หรือ Conclusion) และเรียกส่วนของ Q_1, Q_2, ..., Q_n ว่าเป็นส่วนของตัว (Body หรือ Premises)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ในกรณีที่ในกฎไม่มีส่วนของ $Q_1, Q_2, \ldots, Q_n$ ก็จะเหลือแค่ส่วนหัวและเครื่องหมาย :- เท่านั้น ในการเขียนกฎที่มีลักษณะแบบนี้เราจะละในส่วนของเครื่องหมายนี้ไว้และจะเขียนเพียงแค่ P เท่านั้น เราจะรูปแบบของกฎแบบนี้ว่า unit clause ซึ่งถ้า unit clause นั้นอนุประโยคที่มีค่าพารามิเตอร์ทั้งหมดเป็นค่าคงที่ เราก็จะเรียกอนุประโยคนั้นว่าเป็นข้อเท็จจริง หมายความว่า จริงๆ แล้วข้อเท็จจริงก็คือรูปแบบของกฎที่เป็นกรณีพิเศษนั่นเอง ตัวอย่างของอนุประโยคสามารถเขียนแสดงได้ดังนี้

P(abc,12), Employee(donald,marketing,215)

ตัวอย่างข้างต้นนี้จะแทนอนุประโยคที่เป็นข้อเท็จจริง ส่วนอนุประโยคที่ยังมีส่วนของ $Q_1, Q_2, \ldots, Q_n$ อยู่จะเป็นกฎ

เมื่อเรานำมาลักษณะเหล่านี้มาประยุกต์ใช้เพื่อทำการเก็บข้อมูลในฐานข้อมูลที่เราได้อธิบายมาแล้วข้างต้น จะได้ลักษณะของข้อมูลที่จัดเก็บอยู่ในฐานข้อมูลดังนี้ ดังตัวอย่างที่แสดงถึงข้อมูลของลูกจ้างในแผนก

สำหรับใน Extensional Database จะมี Base Predicate เป็น

emp(Eno,Name,Dno,Job,Sal,Allow)
dep(Dno,Dname,Location,Manager)

และ Intensional Database ก็ประกอบด้วยกฎที่มีทั้ง Derived Predicate คือพรีดิเคตที่ได้มาจากพรีดิเคตอื่นๆ และข้อบังคับที่เกี่ยวกับตัวข้อมูลดังนี้

ตัวอย่างของ Derived Predicate เช่น

- เราอาจหา Derived Predicate สำหรับบอกพนักงานที่เป็นนายจ้าง

manager(Eno,Name,Dno,Sal) :-
emp(Eno,Name,Dno,_,Sal,_), dep(_,_,_,Eno).

- Derived Predicate ของ "assists" ใช้สัญลักษณ์ assists(X,Y) แทนด้วย X assists Y โดยกำหนดว่า X จะ assists Y ถ้า X ถูกจ้างในแผนกที่มี Y เป็นนายจ้าง

assists(X,Y) :- emp(X,_,Z,_,_,_), dep(Z,_,_,Y).

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตัวอย่างของข้อบังคับของตัวข้อมูล เช่น

- การที่ลูกจ้างมีเงินเดือนไม่เกินกว่า 40% ของนายจ้าง

$$integ\_check\_1 :- emp(Eno, \_, \_, \_, X, Y),\ Y <= 0.4*X.$$

- ไม่มีลูกจ้างคนใดที่มีเงินเดือนมากกว่านายจ้างของเขา

$$integ\_check\_2(Eno) :- emp(Eno, \_, dno, \_, X, \_),\ manager(\_, \_, Dno, Y),\ X <= Y.$$

2.3. ระบบฐานข้อมูลเชิงวัตถุ

แนวทางหนึ่งที่มีขึ้นเพื่อเป็นการขยายความสามารถเพื่อให้ระบบฐานข้อมูลจัดการข้อมูลที่มีลักษณะซับซ้อนได้ก็คือแนวทางการพัฒนาระบบฐานข้อมูลที่เป็นลักษณะในเชิงวัตถุ โดยสำหรับในแนวคิดเชิงวัตถุนี้มีต้นกำเนิดมาจากการเขียนโปรแกรมในเชิงวัตถุมาก่อน (Object-Oriented Programming) ซึ่งเมื่อนำมาประยุกต์ใช้กับระบบฐานข้อมูลแล้ว ก็มีการเปลี่ยนในส่วนของรายละเอียดปลีกย่อยเล็กน้อย แต่โดยรวมแล้วก็ยังมีลักษณะที่สำคัญๆ คล้ายคลึงกันอยู่ และยังมีความสามารถในส่วนของการทำงานของการจัดการด้านฐานข้อมูลที่ระบบจัดการฐานข้อมูลแบบ Relational มีเหมือนเดิม สำหรับลักษณะโดยรวมที่สำคัญๆ ของระบบฐานข้อมูลเชิงวัตถุสามารถอธิบายโดยสังเขปได้โดยแบ่งเป็นหัวข้อย่อยๆ ได้ดังนี้

2.3.1. Object Identity

สำหรับในระบบฐานข้อมูลที่เป็นแบบเน้นการแสดงค่า (Value-Oriented) จะใช้ค่าๆ หนึ่งที่อยู่ในฐานข้อมูลเป็นค่าหลัก (Key) ในการแทนถึงวัตถุใดวัตถุหนึ่ง ปัญหาที่พบเมื่อมีการแทนถึงวัตถุในลักษณะนี้ก็คือ ในกรณีที่ค่าหลักที่ใช้ในการแทนถึงวัตถุหรือสิ่งของใดๆ นั้นเป็นค่าที่มีการเปลี่ยนแปลงด้วย จะทำให้เกิดปัญหาของการแทนค่าถึงวัตถุตัวใดตัวหนึ่งเนื่องจากค่าที่ใช้ในการอ้างอิงนั้นเกิดการเปลี่ยนแปลงไปจากเดิม และต้องแก้ปัญหาด้วยการแก้ไขค่าหลักที่ใช้ทำการอ้างอิงนี้ทุกตัวเมื่อมีการเปลี่ยนแปลงเกิดขึ้นเพื่อให้การอ้างอิงถึงค่าของสิ่งใดสิ่งหนึ่งในฐานข้อมูลเป็นไปได้เหมือนเดิม

ในกรณีของระบบฐานข้อมูลเชิงวัตถุ จะแก้ปัญหานี้ด้วยการแทนค่าที่ใช้ในการอ้างอิงถึงวัตถุตัวนั้นเป็นค่าต่างหากค่าหนึ่ง ซึ่งจะเรียกค่านี้ว่า Object Identifier ลักษณะของค่านี้จะแตกต่างไปจากค่าที่เก็บอยู่ในฐานข้อมูลค่าอื่นๆ ก็คือ จะมีค่าที่ไม่ซ้ำกันเพื่อที่ระบบจัดการฐานข้อมูลใช้ค่าเหล่านี้ในการอ้างอิงถึง Object ตัวใดตัวหนึ่งที่เก็บอยู่ในฐานข้อมูลเป็นหลัก ซึ่งในอ้างอิงนี้เราอาจพบว่า จะมี Object สองตัวใดๆ ที่มีค่าเหมือนกัน แต่มีการอ้างอิงโดยค่า Object Identifier คนละตัวก็ได้ ซึ่งค่านี้ ระบบฐานข้อมูลจะใช้ในการอ้างอิงถึง Object ตัวนั้นเป็นแบบถาวร คือ ค่าที่ใช้ในการอ้างอิงเหล่านี้จะถูกสร้างและทำลายโดยตัวระบบจัดการฐานข้อมูลเองและจะไม่สามารถทำการเปลี่ยนแปลงค่าต่างๆ เหล่านี้ได้ ซึ่งจะแตกต่างจากค่าอื่นๆ ที่อยู่ในฐานข้อมูล ส่วนความสามารถที่ผู้ใช้ระบบฐานข้อมูลจะนำค่าเหล่านี้ไปใช้ในการอ้างอิงหรือใช้งานได้หรือไม่นั้น จะขึ้นอยู่กับ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ระบบจัดการฐานข้อมูลที่ใช้ว่าจะเปิดโอกาสให้ผู้ใช้ได้เข้าถึงข้อมูลของค่าเหล่านี้หรือไม่ ซึ่งก็แตกต่างกันไปในแต่ละระบบ

2.3.2. Complex Value

สำหรับ Object แต่ละตัวจะมีค่าที่เกี่ยวข้องด้วยอยู่ค่าหนึ่ง ซึ่งอาจเป็นค่าที่มีลักษณะซับซ้อนก็ได้ ซึ่งค่านี้ก็เป็นค่าของ Object Identifier ก็ได้ เป็นผลให้สามารถมี Object สองตัวใดๆ ที่อ้างอิงถึง Object อีกตัวหนึ่งซึ่งเป็นตัวเดียวกันได้ โดยใช้ Object Identifier ที่มีค่าเหมือนกัน และเป็นผลให้เมื่อมีการเปลี่ยนแปลงข้อมูลในตัว Object นั้นๆ ก็จะไม่ส่งผลกระทบไปยัง Object ตัวอื่นๆ ที่มีการอ้างอิงถึง ซึ่งค่าที่เก็บอยู่ใน Object นี้สามารถเรียกได้อีกอย่างว่าเป็นสถานะ (State) ของ Object ตัวนั้นได้ด้วย

2.3.3. Method

Object ที่เก็บอยู่ในระบบฐานข้อมูลเชิงวัตถุนั้นจะมีวิธีการใช้งานโดยผ่านทาง Method นี้ โดย Method จะประกอบด้วย ชื่อ, รูปแบบในการใช้ (Signature), และส่วนที่เป็นตัวเนื้อของ Method

ในการใช้งานนี้จะต้องอ้างอิง Method ตามรูปแบบชื่อและรูปแบบในการใช้ให้ถูกต้องตามที่มีอยู่ ส่วนที่เป็นตัวเนื้อของ Method ซึ่งแสดงถึงรูปแบบการทำงานของ Method นั้นๆ โดยปกติจะถูกเขียนด้วยภาษาที่เป็นส่วนขยายเพิ่มเติมมาจากภาษาโปรแกรม

2.3.4. Class

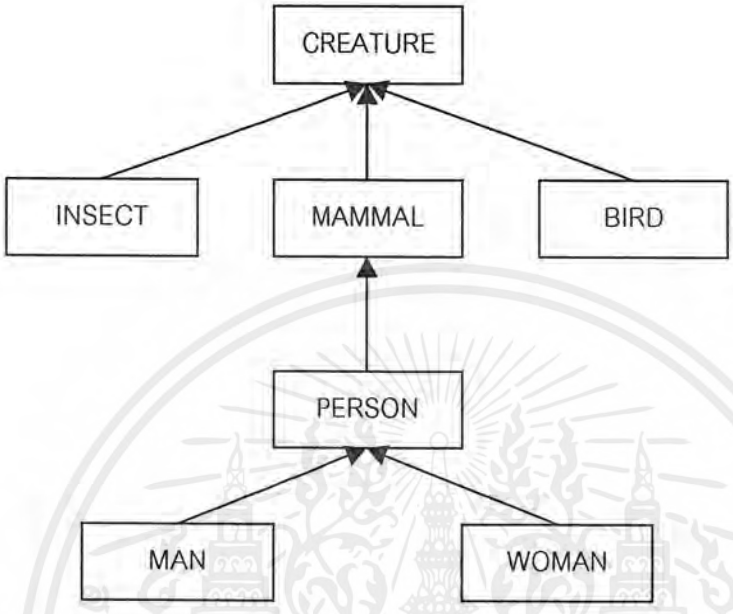
จะหมายถึง กลุ่มของ Object ที่มีโครงสร้างภายในที่เหมือนๆ กัน นั่นคือ จะมีค่ารายละเอียดของค่า Attribute และรายละเอียดของ Method ที่เหมือนๆ กัน สำหรับตัว Object ใดๆ ที่เป็นของ Class ใด Class หนึ่งที่อยู่ในฐานข้อมูลจะเรียก Class นั้นว่าเป็น Class หลัก (Primary Class, Immediate Class หรือ Proper Class) ของ Object นั้นๆ ซึ่งจะเป็น Class ที่ Object ตัวนั้นมีความสัมพันธ์ในรูปแบบที่เป็น ตัวแทน (Instance-of) ของ Class นั้น และใน Class นี้จะเก็บลักษณะต่างๆ ของตัว Object ที่อยู่ใน Class ได้แก่ Attribute และ Method ต่างๆ ในบางกรณีอาจพบว่าเราสามารถให้ Object ตัวหนึ่ง ๆ เป็นตัวแทนของ Class มากกว่า 1 Class ได้ โดยการสืบทอดคุณสมบัติ (Inheritance) ซึ่งจะอธิบายต่อไป แต่อย่างไรก็ตาม Object ตัวนั้นจะต้องมี Class หลักได้เพียงตัวเดียว

2.3.5. การสืบทอดคุณสมบัติ

ในระบบฐานข้อมูลเชิงวัตถุ แนวคิดของการสืบทอดคุณสมบัติทำให้ Object มี Attribute และ Method ที่เจาะจง(Specialized) ลงไปจาก Class ที่เป็น Class ต้น (Superclass หรือ Parent Class) ได้ โดยการสืบทอด Attribute และ Method จาก Class ดังกล่าวมายัง Class ของตัวจะเรียกว่า Subclass สำหรับ Object ที่อยู่ใน Class ใหม่นี้ ก็ยังถือว่าเป็นตัวแทนของ Class ต้นที่สืบทอดมา จากคุณสมบัติที่มีมาจากการสืบทอดนี้ทำให้เราได้ประโยชน์จากการที่เราสามารถเพิ่มศักยภาพทางด้านการนำกลับมาใช้ใหม่ (Reusability) และการขยายเพิ่มเติมคุณสมบัติ (Extensibility) ได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ในส่วนต่อไปจะเป็นการแสดงตัวอย่างของลำดับชั้นของ Class (Class Hierarchies) ที่ได้มาจากการสืบทอดคุณสมบัติ โดยการสืบทอดจะเป็นในลักษณะแบบ Class ที่อยู่ในระดับที่ต่ำกว่าจะสืบทอดค่าคุณสมบัติต่างๆ จาก Class ที่อยู่สูงขึ้นไป ซึ่งจะแสดงได้ดังรูป



รูปที่ 2-1 การสืบทอดคุณสมบัติแบบลำดับชั้น

จากรูปจะเห็นว่ามี Class *Mammal Bird* และ *Insect* ซึ่งเป็น Class ย่อยของ Class *Creature* ส่วน Class Person เป็น Class ย่อยของ Class *Mammal* และ Class *Man* และ *Woman* เป็น Class ย่อยของ Class *Person* ในส่วนรายละเอียดของ Class ดังกล่าวข้างต้นนี้ เราสามารถเขียนตัวอย่างแสดงได้โดยยึดจากลำดับชั้นจากรูปได้ดังตัวอย่างข้างล่างที่จะแสดงต่อไปนี้ ซึ่งเขียนอยู่ในรูปที่ยึดหลักของไวยากรณ์ที่คล้ายๆกับภาษา C++

```
class creature
        properties
                type : string;
                weight : real;
                habitat : ( ... รูปแบบข้อมูลที่แสดงถึงลักษณะที่อยู่อาศัย ...);
                ...
        operations
                create() -> creature;
                predators (creature) -> set (creature);
                life_expectancy (creature) -> integer;
                ...
end creature.
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
class mammal
        inherit creature;
        properties
                Gestation_period : real;
        operations
                ...
end mammal.

class person
        inherit mammal;
        properties
                Surname, firstname : string;
                Date_of_birth : date;
                Origin : country;
end person.

class man
        inherit person;
        properties
                Wife : woman;
                ...
        operations
                ...
end man.

class woman
        inherit person;
        properties
                Husband : man;
                Maiden_name : string;
                ...
end woman.
```

2.3.6. Polymorphism

สำหรับความหมายของ Polymorphism โดยรวมแล้วก็คือ รูปแบบหนึ่งของการสืบทอดคุณสมบัติ โดยจะแบ่งออกได้เป็นสองลักษณะใหญ่ก็คือ ลักษณะแบบ Overloading จะหมายถึงการที่ Method สองตัวใดๆ ใน Class เดียวกันที่มีชื่อเหมือนกัน แต่ละรูปแบบการใช้งานที่แตกต่างกัน และแบบ Overiding ซึ่งหมายถึงความสามารถของโปรแกรมที่อ้างอิงถึง Attribute หรือ Method ที่มีชื่อเหมือนกัน แต่มีการเปลี่ยนแปลงความหมายไปเมื่อมีการสืบทอดค่าคุณสมบัตินั้นมาจาก Class ต้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.4. ระบบฐานข้อมูลอนุมานเชิงวัตถุ

สำหรับอีกแนวทางหนึ่งที่เป็นการขยายความสามารถของระบบจัดการฐานข้อมูลที่เป็นแบบ Relational ซึ่งมีการค้นคว้าวิจัยและพัฒนากันก็คือ แนวทางที่พยายามรวมเอาทั้งจุดเด่นของระบบฐานข้อมูลอนุมานและระบบฐานข้อมูลเชิงวัตถุเข้าด้วยกัน โดยทิศทางของการพัฒนาวิจัยจะเป็นการพยายามหาทฤษฎีที่ใช้อธิบายเรื่องของแนวความคิดเชิงวัตถุ ซึ่งแบ่งออกได้เป็นหลายแนวทางด้วยกัน เช่นการขยายความหมายของภาษาที่เป็น Logic Programming ที่มีอยู่ก่อนแล้ว เช่น Prolog ให้มีแนวความคิดในเชิงวัตถุด้วย โดยการใช้ลักษณะของภาษาที่เป็นแบบเชิงวัตถุ เช่น C++ หรือ Smalltalk เพิ่มเติมเข้ามา หรืออาจคิดทฤษฎีตรรกศาสตร์ใหม่ๆ เพิ่มเติมจากของเดิมเพื่อให้มีความสามารถในส่วนของที่เป็นเชิงวัตถุ และแนวทางอีกอย่างก็คือการคิดค้นทฤษฎีขึ้นมาใหม่ทั้งหมด สำหรับในส่วนของรายละเอียดของแนวทางเหล่านี้ จะขอไม่กล่าวถึงในรายงานฉบับนี้ เนื่องจากในโครงงานจะไม่เน้นในส่วนของการสร้างระบบฐานข้อมูลในส่วนนี้มากนัก แต่จะยกตัวอย่างหนึ่งของภาษาที่ใช้ในการจัดการระบบฐานข้อมูลชนิดนี้ ซึ่งจะเป็นหัวข้อสุดท้ายในบทนี้

- **F-logic [Kifer 1995]**

  สำหรับตัวอย่างหนึ่งของภาษาที่ใช้ในระบบฐานข้อมูลที่เป็นแบบอนุมานเชิงวัตถุ ซึ่งค่อนข้างเป็นที่รู้จักกันดี เนื่องจากมีการพัฒนาที่ยาวนานและ มีรูปแบบการกำหนดความหมายต่างๆ ที่รัดกุม (Well-Defined Semantics) นอกจากนี้ยังมีการพัฒนาต้นแบบเพื่อทดลองการใช้งานขึ้นมาแล้วด้วย ตามรายละเอียดใน [Kandzia, Schlepphorst 1997] แต่ยังมีคุณสมบัติในการทำงานของภาษาไม่ครบถ้วนทุกส่วน ซึ่งรูปแบบของภาษาเป็นการพัฒนาขึ้นมาใหม่ทั้งหมด โดยใช้ทฤษฎีทางด้านตรรกศาสตร์เป็นแนวทางในการพัฒนา สำหรับลักษณะของภาษาโดยทั่วไปมีรายละเอียดดังนี้

  - **Object Identity (OID)** สำหรับในภาษาจะใช้ OID ที่เป็นลักษณะแบบ Logical ซึ่งจะมีลักษณะเหมือนๆ กับ First-Order Term เช่น

    13 หรือ *_256* หรือ *john32*

    และสามารถใช้ฟังก์ชันในการสร้าง OID ที่มีความซับซ้อนมากขึ้นได้ เช่น

    father(mary) หรือ *head(csdept(stonybrook))*

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- **Attribute** จะแทนอยู่ในรูปแบบที่เรียกว่า Molecular Formula ซึ่งมีลักษณะตามตัวอย่าง

john[name → "John Doe"; salary → 20000]
father(mary)[address → "Main St. USA"; spouse → sally]

จากตัวอย่างข้างต้น *john, father(mary)* และ *sally* เป็น OID ของ Object ของบุคคล, *"John Doe", "main St. USA"* เป็น OID ของ Object แบบสตริง (string) และ *20000* เป็น OID ของ Object แบบเลขจำนวนเต็ม (Integer) โดยค่า *name, salary, address* และ *spouse* จะเป็น Attribute มีข้อสังเกตว่าค่าแต่ละค่าที่เป็นสตริงและเลขจำนวนเต็มจะใช้ค่านั้นใช้เป็น OID ด้วย

- **Set-Valued Attribute** จากข้างต้น เรายังพบข้อจำกัดในการแสดงถึง Object ที่มีความซับซ้อนอยู่ ซึ่งจากตัวอย่างที่ผ่านมา เราไม่มีทางที่จะบอกว่า John มีลูกชื่อ Mary, Bob และ Alice เว้นเสียแต่เราจะทำการขยายความสามารถในการจัดการ Object ที่มีความซับซ้อน โดยอนุญาตให้ Attribute มีค่าเป็นเซต (Set-Valued) ได้ เช่น จากตัวอย่างที่แล้ว ถ้าเราต้องการเพิ่มค่า Attribute ที่มีค่าเป็นเซต คือ เซตของลูกๆ ก็สามารถทำได้โดยเขียนเป็น

john[children ->> {mary, bob, alice}]

- **Method** สำหรับในภาษา F-logic จะมอง Method เป็นเหมือนกับค่า Attribute ที่มีค่าพารามิเตอร์

- **ลำดับชั้นของ Class** จะแสดงโดยการสร้างรูปแบบที่ใช้ในการอ้างอิงเพิ่มขึ้นมาใหม่มีลักษณะตามตัวอย่าง

john : student	หมายถึง *john* เป็น Object ของ Class *student*
student :: person	หมายถึง *student* เป็น Class ย่อยของ Class *person*

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- **Higher-order Syntax** ถูกสร้างขึ้นมาเพื่อวัตถุประสงค์ต่างๆ ดังนี้
    1. ให้สามารถกำหนด (Defining), จัดการในเรื่องต่างๆ ของ Class และ Object โดยสามารถใช้ภาษาตัวเดียวกันได้
    2. ให้สามารถในการกำหนด Virtual Class (หรือ View) และคุณสมบัติต่างๆที่ใช้ในกฎอนุมาน
    3. ให้มีความสามารถในส่วนของการสืบทอดคุณสมบัติในส่วนของโครงสร้างข้อมูล
    4. ให้มีความสามารถในการตรวจสอบโครงสร้างข้อมูลได้

นอกจากนี้ก็ยังมีส่วนประกอบอื่นๆ อีกก็คือ เครื่องหมายของตัวที่ใช้ทำการเชื่อมต่อ (Connective) ได้แก่ AND, OR, NOT และ IF

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 3
ระบบผู้เชี่ยวชาญ

3.1 บทนำ

จากคำจำกัดความของโดยศาสตราจารย์ Edward Feigenbaum แห่งมหาวิทยาลัยสแตนฟอร์ด ซึ่งเป็นนักค้นคว้าชั้นแนวหน้าในสาขาวิชาปัญญาประดิษฐ์ (Artificial Intelligence) ได้กล่าวไว้ว่า ระบบผู้เชี่ยวชาญคือโปรแกรมคอมพิวเตอร์ที่มีความฉลาดด้วยการใช้ความรู้และขบวนการอนุมาน (Inference Procedure) ในการแก้ปัญหาที่มีความยุ่งยากขนาดถึงต้องใช้ประสบการณ์ความชำนาญของมนุษย์จึงจะแก้ไขได้

กล่าวคือ ระบบผู้เชี่ยวชาญ คือโปรแกรมคอมพิวเตอร์ที่เก็บทั้งความรู้เกี่ยวกับปัญหาที่จะแก้และขบวนการอนุมานเพื่อนำไปสู่ผลสรุปหรือคำตอบของปัญหานั้น ความรู้ที่เก็บมีทั้งความรู้ที่เป็นความจริงที่อาจจะถูกบันทึกไว้ในรูปของตำราหรือเอกสารทางวิชาการและความรู้ที่ได้จากประสบการณ์ที่อาจจะไม่อยู่ในรูปของตำราหรือเอกสารทางวิชาการ แต่จะต้องดึงมาจากผู้เชี่ยวชาญหรือผู้ชำนาญที่มีประสบการณ์นั้น

ปัญหาที่ระบบผู้เชี่ยวชาญจะแก้ส่วนใหญ่จะเป็นปัญหาที่ยุ่งยากและไม่ค่อยมีโครงสร้าง (Semi-Structured หรือ Ill-Structured Problem) ในปัญหาประเภทนี้คำตอบจะมีโอกาสเป็นได้หลายอย่าง ทั้งนี้ขึ้นอยู่กับสภาพขณะนั้นของปัญหาและข้อมูลที่เข้ามา ปัญหาประเภทนี้อาจจะอุปมาได้เหมือนกับการเล่นหมากรุก การเดินหมากครั้งต่อไปนั้นเดินได้หลายวิธีด้วยกันแต่ตัวหมากที่จะเดินดีที่สุดจะตัดสินจากสภาพของกระดานในขณะนั้นและหมากที่คิดว่าคู่ต่อสู้จะเดินในครั้งต่อไป ในการแก้ปัญหาประเภทนี้เรามักไม่สามารถจะกำหนดขั้นตอนในการแก้ปัญหาอย่างชัดเจนไว้ล่วงหน้าได้ ดังนั้นวิธีการแก้ปัญหาแบบมีที่มาซึ่งเป็นแบบเขียนโปรแกรมเป็นขั้นตอนการแก้ปัญหาหรืออัลกอริธึม (Algorithm) จึงไม่สามารถจะนำมาประยุกต์ใช้ในปัญหาประเภทนี้ได้ ระบบผู้เชี่ยวชาญถึงแม้จะเป็นโปรแกรมคอมพิวเตอร์ชนิดหนึ่ง แต่โครงสร้างและเทคนิคที่ใช้ในการสร้างหรือพัฒนาต่างจากของโปรแกรมที่มีมาและเป้าหมายในการประยุกต์ใช้ก็แตกต่างกัน การประยุกต์ใช้ระบบผู้เชี่ยวชาญที่ประสบความสำเร็จเท่าที่มีมาได้แก่ การวินิจฉัยโรค, การสำรวจทรัพยากรธรณี, การวิเคราะห์โครงสร้างสารอินทรีย์เคมี และการแนะนำระบบคอมพิวเตอร์ เป็นต้น

3.2. โครงสร้างพื้นฐานของระบบผู้เชี่ยวชาญ

ระบบผู้เชี่ยวชาญโดยทั่วไปจะประกอบด้วยส่วนประกอบพื้นฐาน 5 ส่วน ดังแสดงในรูปที่ 1.2 ส่วนที่เป็นหัวใจที่จะขาดเสียมิได้ คือ ฐานความรู้และเครื่องมืออนุมาน (Inference Engine) รายละเอียดโดยย่อของแต่ละส่วนสามารถอธิบายได้ดังนี้

- ฐานความรู้ (Knowledge Base)

ส่วนนี้เปรียบเสมือนกับข้อมูลในซอฟต์แวร์ธรรมดาหรือฐานข้อมูล (Database) ในระบบสารสนเทศ (Information Systems) เป็นส่วนที่ใช้เก็บความรู้ทุกประเภทไม่ว่าจะเป็นความรู้ที่ได้จากตำราหรือความรู้ที่ได้จากประสบการณ์ ปัญหาหลักของฐานความรู้ก็คือ การเลือกวิธีที่จะใช้ในการแสดงความรู้หรือโครงสร้างสำหรับเก็บความรู้ที่เหมาะสม ปัญหานี้เปรียบได้กับการเลือกโครงสร้างข้อมูลหรือโครงสร้างฐานข้อมูลที่เหมาะสมในระบบซอฟท์แวร์แบบธรรมดา

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

- เครื่องมืออนุมาน (Inference Engine)

ส่วนนี้เปรียบได้กับอัลกอริธึม เป็นส่วนที่ควบคุมการใช้ความรู้ในฐานความรู้เพื่อแก้ไขปัญหาอย่างมีประสิทธิภาพ วิธีการอนุมานมีหลายแบบแต่แยกประเภทใหญ่ๆ ได้ 2 ประเภท คือ อนุมานแบบเดินหน้า (Forward-Chaining Inference) และอนุมานแบบย้อนหลัง (Backward-Chaining Inference) ทั้งสองวิธีนี้ต่างก็มีข้อดีและข้อเสีย ทั้งนี้ขึ้นอยู่กับลักษณะของปัญหา ในระบบผู้เชี่ยวชาญบางระบบจะใช้วิธีอนุมานทั้งสองวิธีรวมกัน

- ส่วนดึงความรู้ (Knowledge Acqiusition)

เป็นส่วนของระบบผู้เชี่ยวชาญที่ใช้ในการช่วยดึงเอาความรู้จากตำราหรือฐานข้อมูลและจากผู้เชี่ยวชาญ การดึงเอาความรู้จากตำราหรือฐานข้อมูลนั้นทำได้ไม่ยาก ถ้าหากเราสามารถจัดการความรู้จากแหล่งดังกล่าวให้เป็นระบบ และเข้ากันได้กับโครงสร้างของฐานความรู้ เราก็จะสามารถบรรจุความรู้เหล่านั้นเข้าไปในฐานข้อมูล แต่ทว่าการดึงเอาความรู้จากผู้เชี่ยวชาญนั้นทำได้ยาก จำเป็นต้องใช้เทคนิคต่างๆ เข้าช่วยหรือไม่ก็ทำให้ระบบผู้เชี่ยวชาญสามารถเรียนรู้ด้วยตนเองในบางส่วน ปัจจุบันการเรียนรู้ (Learning) เป็นหัวข้อค้นคว้าที่นักค้นคว้าในสาขาปัญญาประดิษฐ์ให้ความสนใจมากที่สุดสาขาหนึ่ง

- ส่วนอธิบาย (Explanation Subsystem)

ส่วนนี้ทำหน้าที่อธิบายรายละเอียดของขั้นตอนการวินิจฉัยต่อผู้ใช้ว่าข้อสรุปหรือคำตอบนั้นได้มาอย่างไรและทำไม

- ส่วนติดต่อกับผู้ใช้ (User Interface)

เป็นส่วนที่ทำหน้าที่เป็นตัวกลางระหว่างผู้ใช้กับระบบเพื่อทำให้การสื่อสารระหว่างผู้ใช้กับระบบเป็นไปได้อย่างราบรื่น และช่วยให้ผู้ใช้ยอมรับระบบมากขึ้น

ในระบบผู้เชี่ยวชาญบางระบบจะไม่มีส่วนประกอบทั้งห้าส่วนดังกล่าวข้างต้น แต่ที่ขาดไม่ได้แน่ๆ คือ ฐานความรู้และเครื่องมืออนุมาน บางคนถึงกับกล่าวว่า Knowledge Base + Inference Engine = Expert System เลย

3.3. รูปแบบการแทนความรู้ (Knowledge Representation) ชนิดต่างๆ ที่ใช้ในระบบผู้เชี่ยวชาญ

สำหรับในระบบผู้เชี่ยวชาญจะมีรูปแบบการแทนควาามรู้ที่แตกต่างกันไปหลายแบบด้วยกัน ซึ่งรูปแบบต่างๆ ที่สามารถพบเห็นได้ พอจะอธิบายได้โดยสังเขปดังนี้ ซึ่งในปัจจุบันระบบผู้เชี่ยวชาญก็มิได้มีการใช้รูปแบบการแทนความรู้ที่อยู่ในลักษณะอันหนึ่งอันใดเพียงอย่างเดียวก็ได้ แต่อาจมีการใช้รูปแบบการแทนความรู้หลายๆ แบบเข้าด้วยกันหรืออาจมีการดัดแปลงรูปแบบการแทนความรู้แบบนี้เข้าด้วยกันด้านอื่นๆ ก็ได้ เช่น การใช้ Neural Network หรือทฤษฎี Fuzzy เป็นต้น

3.3.1. แบบของกฎ (Rule-Based)

จะเป็นการแสดงความรู้ในรูปแบบของกฎ ซึ่งแทนอยู่ในรูปแบบของ First Order Logic คือ จะแทนรูปแบบความรู้ต่างๆ ในลักษณะของ Horn Clause ซึ่งสามารถเขียนให้อยู่ในรูปแบบที่สามารถทำความเข้าใจได้ง่ายๆ ดังนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้
37072

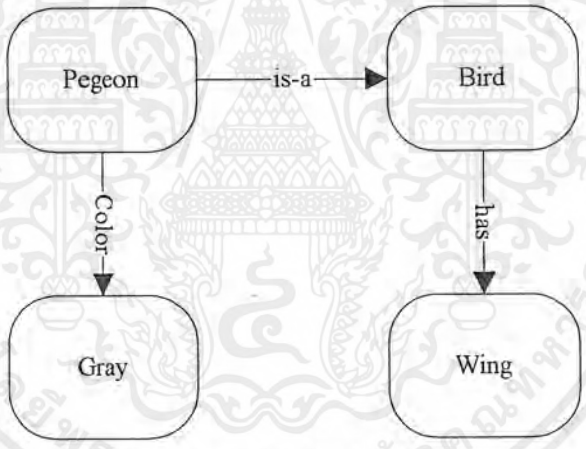
(Condition) → *(Action)* หรือเขียนในรูปแบบที่เข้าใจง่ายๆ ได้อีกแบบ คือ

If (Condition) Then (Action)

คือ เมื่อพบว่าในส่วนของ *(Condition)* เป็นจริงแล้ว ก็ให้ทำในส่วนของ *(Action)* ซึ่งรูปแบบการแทนแบบนี้จะสามารถเว้นไว้และมีเฉพาะในส่วนของ *(Action)* ได้ด้วย ซึ่งถ้าเป็นการแทนแบบนี้จะหมายถึงการแทนความรู้ที่เป็นข้อเท็จจริง (Fact)

3.3.2. แบบใช้ข่ายความหมาย (Semantic Network)

ในแบบนี้จะแทนความรู้โดยใช้การแสดงความสัมพันธ์ระหว่างสิ่งสองสิ่ง (Binary-Relation) ซึ่งอาจแทนได้ง่ายๆ ด้วยการใช้กราฟ โดยส่วนของ Node ในกราฟจะแสดงในส่วนที่เป็น Object, Concept, Event, Action และ State ของสิ่งที่เราต้องการสร้างเป็นฐานความรู้ ส่วนในส่วนของ Arc จะแสดงถึงความสัมพันธ์กันระหว่าง Node ต่างๆ ที่แทนถึงสิ่งที่ได้อธิบายไว้ในข้างต้นแล้ว ซึ่งถ้ารูปแบบของความสัมพันธ์มีทิศทางแล้วก็ต้องใช้ Arc แบบที่แสดงถึงทิศทางของความสัมพันธ์ด้วย (หมายถึงใช้ Arc ที่มีหัวลูกศร) ตัวอย่างของการแทนความรู้แบบนี้แสดงได้ดังรูปที่ 3-1



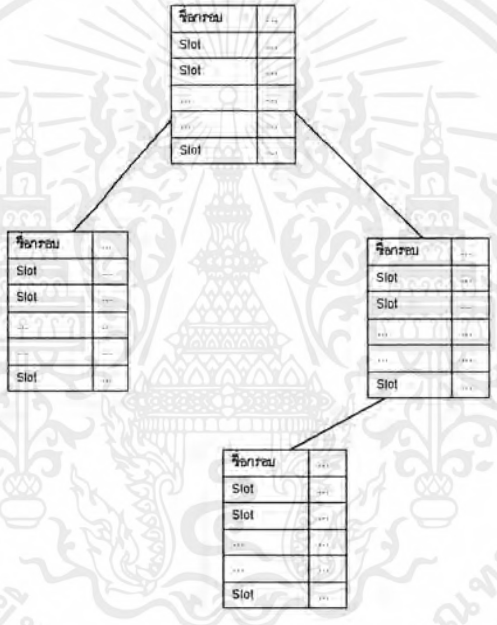
รูปที่ 3-1 ตัวอย่างการแทนความรู้ด้วยข่ายความหมาย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.3.3. แบบใช้กรอบ (Frame-based) และแบบเชิงวัตถุ (Object-Oriented)

การแทนรูปแบบความรู้แบบกรอบนี้จะมีลักษณะแบบเป็นโครงสร้าง คือจะใช้แนวความคิดที่ว่า ในการแทนความรู้หนึ่งๆ นั้นจะประกอบไปด้วยสิ่งย่อยๆ หลายๆ อย่างประกอบเข้าด้วยกัน และในการอ้างอิงถึงความรู้นั้นก็จะอ้างอิงถึงส่วนย่อยๆ ต่างๆ เหล่านี้ทั้งหมด ซึ่งส่วนย่อยๆ ต่างๆ เหล่านี้จะถูกนำมารวมกันเป็นกรอบ (Frame) โดยการแสดงความรู้จะต้องมีกรอบที่แสดงถึงความรู้สามัญบรรจุอยู่ ซึ่งจะเป็นความรู้พื้นฐานในกรณีที่ไม่มีสิ่งบ่งบอกอย่างอื่นก็จะใช้ข้อมูลที่อยู่ในกรอบนี้ไปใช้ และสามารถมีกรอบได้หลายๆ กรอบโดยจะมีการแสดงถึงความสัมพันธ์ที่เกี่ยวข้องกันได้เช่นเดียวกับในกรณีของข่ายความหมาย โดยรูปแบบที่สำคัญจะเป็นการสืบทอดคุณสมบัติ คือ ในกรณีที่กรอบนั้นไม่ได้มีการกำหนดข้อมูลบางอย่างเป็นการจำเพาะก็จะใช้ข้อมูลที่สืบทอดมาจากกรอบอื่นๆ แทน

ตัวอย่างรูปแบบการแทนความรู้ด้วยกรอบสามารถแสดงได้โดยดูจากรูปที่ 3-2



รูปที่3-2 การแทนความรู้ด้วยกรอบ

สำหรับการแทนความรู้แบบเชิงวัตถุในระบบผู้เชี่ยวชาญนั้นเป็นผลมาจากการที่แนวความคิดเชิงวัตถุในวิทยาการคอมพิวเตอร์ในด้านอื่นๆ เป็นที่นิยมขึ้นมา จึงมีการพยายามนำมาประยุกต์เข้าด้วยกันกับการแทนความรู้ด้วยกรอบซึ่งก็มีลักษณะคล้ายๆ กันอยู่แล้วและมีการปรับเปลี่ยนไปมาโดยใช้หลักการของทั้งสองส่วนผสมผสานเข้าด้วยกัน จนบางครั้งเราสามารถเรียกรูปแบบต่างๆ ทั้งสองแบบนี้เข้ากันได้ แต่อย่างไรก็ดี ได้มีผู้ที่ชี้ให้เห็นถึงความแตกต่างของรูปแบบทั้งสองนี้ เช่นใน [Nikolopoulos 1997] โดยให้ดูในส่วนของการแบ่งแยกในส่วนของ Class และ Instance ที่ชัดเจนถ้าเป็นการแทนรูปแบบที่เป็นเชิงวัตถุ ในขณะที่ในแบบกรอบต้องมีการพิจารณาเป็นกรณีๆ ไปว่ากรอบนั้นเป็น Class หรือว่าเป็น Instance ซึ่งบางครั้งกรอบที่ใช้แสดงความรู้ตัวเดียวกันจะเป็น Class ได้และก็อาจเป็น Instance ได้ด้วยเมื่อเราพิจารณาในอีกกรณี

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.4. ข้อจำกัดของระบบผู้เชี่ยวชาญ

ถึงแม้ว่าระบบผู้เชี่ยวชาญจะมีความสามารถในการทำงานเพื่อแก้ปัญหาหรือช่วยในการทำงานบางประเภทได้ยืดหยุ่นและดีกว่าโปรแกรมคอมพิวเตอร์อื่นๆ ก็ตาม แต่ระบบผู้เชี่ยวชาญที่ใช้กันอย่างแพร่หลายอยู่ในปัจจุบันนี้ก็มีข้อจำกัดที่เป็นอุปสรรคที่สำคัญของการพัฒนาตัวระบบก็คือ การที่ระบบผู้เชี่ยวชาญสามารถทำงานกับฐานความรู้ที่ต้องสามารถเก็บไว้ในหน่วยความจำหลักของเครื่องคอมพิวเตอร์เท่านั้น ทำให้ในการพัฒนาระบบผู้เชี่ยวชาญต้องทำกับระบบที่มีขนาดไม่ใหญ่เกินไป เพื่อที่จะให้สามารถเก็บฐานความรู้ไว้ในหน่วยความจำหลักของเครื่องคอมพิวเตอร์ได้ และในฐานความรู้ที่สร้างขึ้นมาจะประกอบไปด้วยรูปแบบความรู้ที่เป็นกฎเสียเป็นส่วนมาก ส่วนข้อมูลที่เป็นข้อเท็จจริงจะถูกกำจัดออกไปจากส่วนของฐานความรู้และจะทำการสอบถามเอาเองจากผู้ใช้ ซึ่งข้อเท็จจริงต่างๆ เหล่านี้เมื่อมีการถามจากผู้ใช้แล้ว ก็ไม่ได้มีการเก็บข้อมูลในส่วนนี้เอาไว้เพื่อนำไปใช้งานเมื่อมีการเรียกใช้งานระบบผู้เชี่ยวชาญในครั้งต่อๆ ไป ทำให้ในการใช้งานระบบผู้เชี่ยวชาญแต่ละครั้งต้องมีการใส่ข้อมูลเพื่อตอบคำถามเดิมๆ อยู่ตลอดเวลาทั้งๆ ที่บางคำถามก็ไม่จำเป็นหรือควรถามเพียงแค่ครั้งแรกที่มีการใช้งานระบบก็พอ ลักษณะการทำงานแบบนี้จะก่อให้เกิดผลเสียในแง่ต่างๆ ต่อไปนี้คือ

- โปรแกรมระบบผู้เชี่ยวชาญมีการถามผู้ใช้มากเกินไป ทำให้เมื่อมีการใช้งานระบบผู้เชี่ยวชาญตัวเดิมนั้นบ่อยครั้งมากเข้า จะก่อให้เกิดความเบื่อหน่ายในการใช้ได้ และทำให้การป้อนข้อมูลของผู้ใช้เพื่อนำไปประกอบการตัดสินใจของระบบเกิดความผิดพลาดขึ้นได้เนื่องจากผู้ใช้ไม่มีความตั้งใจที่จะตอบคำถามที่ระบบต้องการอย่างระมัดระวัง
- คำถามบางคำถามจะเป็นการถามถึงข้อมูลในอดีต ซึ่งจะเป็นข้อมูลที่จะเหมือนกันทุกครั้งหรือมีการเปลี่ยนแปลงที่ค่อนข้างน้อย และทุกครั้งที่มีการเรียกใช้งานระบบผู้เชี่ยวชาญจะต้องมีการถามถึงข้อมูลที่ไม่ค่อยมีการเปลี่ยนแปลงนี้อยู่ตลอดทุกครั้ง บางครั้งข้อมูลเหล่านี้ตัวผู้ใช้เองก็อาจจำไม่ได้เป็นสาเหตุทำให้เกิดเหตุการณ์ที่ผู้ใช้ให้ข้อมูลที่ผิดพลาดขึ้นได้
- คำถามบางอย่างที่ระบบผู้เชี่ยวชาญจะถามกับผู้ใช้บางคำถามจะเป็นการถามถึงข้อมูลที่แม้แต่ตัวผู้ใช้ระบบเองก็ไม่ทราบ คำถามที่อยู่ในข่ายกรณีนี้จะพบเห็นได้อยู่ทั่วไปในกรณีที่เป็นระบบผู้เชี่ยวชาญทางด้านการแพทย์ ซึ่งจะมีการอ้างอิงถึงข้อมูลบางอย่างที่เป็นรายละเอียดเฉพาะทางซึ่งบางครั้งตัวผู้ใช้ก็ไม่ได้มีการทราบมาก่อน เช่น ถ้าระบบผู้เชี่ยวชาญมีการถามถึงชื่อตัวยาที่ผู้ป่วยเคยได้รับมาก่อนหน้านี้ ตัวผู้ป่วยเองซึ่งเป็นผู้ใช้ระบบผู้เชี่ยวชาญก็อาจจะไม่ทราบก็ได้ เพราะรายละเอียดต่างๆ เหล่านี้ทางแพทย์ผู้สั่งยาจะไม่จำเป็นต้องมีการบอกกับผู้ป่วยก็ได้ เป็นต้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.5. การติดต่อของระบบผู้เชี่ยวชาญกับโปรแกรมอื่นๆ

จากการที่ตัวของโปรแกรมระบบผู้เชี่ยวชาญเองนั้นมีข้อจำกัดต่างๆ ดังที่ได้อธิบายมาแล้วข้างต้น ทางผู้สร้างโปรแกรมเปลือกระบบผู้เชี่ยวชาญ (Expert System Shell) ซึ่งจะหมายถึงโปรแกรมระบบผู้เชี่ยวชาญที่มีแต่ส่วนของกลไกเครื่องมืออนุมาน และจะเว้นในส่วนของฐานความรู้ให้ผู้พัฒนาระบบเชี่ยวชาญทำการพัฒนาและเขียนขึ้นมาเอง จึงได้มีการเพิ่มเติมความสามารถในการติดต่อกับโปรแกรมตัวอื่นๆ หรืออย่างน้อยที่สุด ก็ได้เพิ่มเติมในส่วนของการอ่านและเขียนข้อมูลจากแฟ้มข้อมูลได้ เพื่อช่วยเพิ่มประสิทธิภาพการในการใช้งานตัวระบบผู้เชี่ยวชาญได้ดีขึ้น โดยถ้าเป็นระบบผู้เชี่ยวชาญที่มีความสามารถอ่านหรือเขียนแฟ้มข้อมูลได้นั้น ก็จะทำให้ระบบผู้เชี่ยวชาญที่สร้างขึ้นมานั้นสามารถใช้งานได้กว้างขวางมากกว่าเดิม แต่ถ้าเป็นการติดต่อกับโปรแกรมอื่นๆ ซึ่งถ้าเป็นโปรแกรมประเภทกระดาษตาราง (Spread Sheet) ระบบผู้เชี่ยวชาญก็จะใช้ประโยชน์จากโปรแกรมตัวนี้ในการทำการวิเคราะห์ข้อมูลทางด้านสถิติ, ทำการสร้างรูปแบบรายงานเพื่อแสดงผลลัพธ์ของการวิเคราะห์ซึ่งจะมีรูปแบบที่หลากหลายกว่าการแสดงผลด้วยการใช้ตัวระบบผู้เชี่ยวชาญเอง และการเรียกดึงข้อมูลจากแฟ้มข้อมูลที่อยู่ในรูปแบบของกระดาษตารางที่ใช้ในงานของโปรแกรมนั้นเพื่อใช้ในการประมวลผลแทนการถามข้อมูลจากผู้ใช้ หรือถ้าเป็นโปรแกรมอีกแบบที่โปรแกรมเปลือกระบบผู้เชี่ยวชาญบางตัวสามารถทำการติดต่อได้ก็คือโปรแกรมจัดการระบบฐานข้อมูลซึ่งทำให้โปรแกรมระบบผู้เชี่ยวชาญสามารถทำงานโดยดึงข้อมูลจากฐานข้อมูลที่เก็บไว้ โดยฐานข้อมูลตัวนี้สามารถถูกใช้งานโดยโปรแกรมอื่นๆ ได้ด้วยเช่นกัน รูปแบบข้อกำหนดที่ใช้ในการติดต่อกับส่วนภายนอกนี้ก็จะแตกต่างกันไป โดยจะขึ้นอยู่กับโปรแกรมเปลือกระบบผู้เชี่ยวชาญนั้นว่ามีลักษณะเป็นอย่างไร เช่น รูปแบบการแทนความรู้ (Knowledge Representation), ระบบปฏิบัติการที่โปรแกรมนั้นทำงาน, รูปแบบที่ใช้ในการติดต่อที่ทางผู้พัฒนาโปรแกรมเปลือกระบบผู้เชี่ยวชาญนั้นๆ กำหนดมา ซึ่งจะมีรายละเอียดที่ไม่เหมือนกันเลย

3.6. ปัญหาที่พบในกรณีที่ข้อมูลที่ระบบผู้เชี่ยวชาญเรียกใช้มีขนาดใหญ่มากๆ

จากหัวข้อที่แล้วจะพบว่าเราสามารถสร้างระบบผู้เชี่ยวชาญที่มีการเก็บข้อมูลต่างๆ ไว้เพื่อในการใช้งานครั้งหลังๆ จะได้นำข้อมูลเหล่านี้ที่ระบบผู้เชี่ยวชาญได้เคยถามผู้ใช้แล้วในอดีตมาใช้ประโยชน์ในการทำงานครั้งหลังๆ ได้ และไม่ต้องทำการถามผู้ใช้ซ้ำอีก ถ้าโปรแกรมเปลือกระบบผู้เชี่ยวชาญที่เราใช้นั้นมีความสามารถในการติดต่อกับส่วนภายนอกได้

อย่างไรก็ตาม ในกรณีที่ข้อมูลที่ระบบผู้เชี่ยวชาญจะต้องใช้ในการทำงานนั้นมีขนาดใหญ่มากๆ แล้วการเรียกข้อมูลจากภายนอกที่อยู่ในรูปแบบของการอ่าน/เขียน แฟ้มข้อมูลโดยตรง หรือการทำงานกับแฟ้มข้อมูลผ่านทางโปรแกรมกระดาษตารางนั้นจะพบกับความยากลำบากมากขึ้น เนื่องจากว่าทางโปรแกรมเปลือกระบบผู้เชี่ยวชาญหรือโปรแกรมกระดาษตารางมีความสามารถในส่วนของการจัดการข้อมูลที่ไม่ดีพอเนื่องจากโปรแกรมเหล่านี้ไม่ได้พัฒนาขึ้นมาเพื่อให้มีความสามารถในส่วนนี้เป็นหลักอยู่แล้ว ทำให้การพัฒนาโปรแกรมระบบผู้เชี่ยวชาญที่มีขนาดใหญ่ๆ จะทำได้อย่างยากลำบาก และเป็นการลำบากอีกเช่นกันถ้าเราจะนำแฟ้มข้อมูลต่างๆ เหล่านี้ที่ใช้ในการทำงานของระบบผู้เชี่ยวชาญมาใช้ประโยชน์โดยโปรแกรมอื่นๆ ด้วยแทนที่จะถูกใช้งานโดยโปรแกรมเพียงตัวเดียว แต่ถ้ามีการนำเอาระบบจัดการฐานข้อมูลมาใช้ประโยชน์ในด้านนี้ เราก็จะพบว่าปัญหาต่างๆ ที่พบเหล่านี้ก็จะสามารถจัดการได้ง่ายๆ โดยที่การใช้งานในส่วนของระบบผู้เชี่ยวชาญก็ยัง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สามารถขยายขอบเขตได้กว้างขวางและยืดหยุ่นได้มากกว่าเดิมด้วย เนื่องจากข้อมูลที่เก็บไว้ในฐานข้อมูลนั้นเราสามารถนำไปใช้งานในส่วนอื่นๆ ได้ด้วย

3.7. ตัวอย่างระบบผู้เชี่ยวชาญที่มีการติดต่อกับข้อมูลภายนอก

3.7.1. BERMUDA [Ioannidis et al. 1989]

เป็นตัวอย่างหนึ่งของต้นแบบที่นำความสามารถของระบบผู้เชี่ยวชาญมาใช้ร่วมกันกับระบบจัดการฐานข้อมูล ซึ่งทำให้ระบบที่ได้มีความสามารถทั้งในส่วนที่เป็นคุณสมบัติของระบบจัดการฐานข้อมูลและในส่วนของการทำงานที่เป็นข้อได้เปรียบในแบบของระบบผู้เชี่ยวชาญ โดยในส่วนของกฎจะเก็บไว้ที่ส่วนของ Front-End ที่จะทำงานติดต่อกับผู้ใช้ซึ่งสามารถมีได้หลายตัว และแต่ละตัวจะเรียกใช้ข้อมูลในส่วนของข้อเท็จจริงที่เก็บไว้ในส่วนของฐานข้อมูลซึ่งเป็นส่วนที่ใช้ร่วมกัน โดยการเรียกใช้จะผ่านตัวกลางตัวหนึ่งที่ใช้ในการจัดการในเรื่องของการจัดส่งข้อมูล

รูปแบบการแทนความรู้ที่ใช้จะเป็นการแทนอยู่ในรูปของภาษา Prolog และมีการติดต่อกับฐานข้อมูลแบบ Relational โดยมีข้อดีอยู่ที่ผู้ใช้ไม่ต้องทำการจัดการในส่วนของการเรียกข้อมูลจากฐานข้อมูลเองเนื่องจากระบบจะทำหน้าที่จัดการให้เองโดยใช้เครื่องมือแบบ Preprocessor สำหรับข้อจำกัดของระบบนี้ก็คือในระหว่างการทำงานของระบบจะต้องไม่มีการเปลี่ยนแปลงใดๆ เกิดขึ้นในส่วนของฐานข้อมูล

3.7.2. Tanguy [Czejdo et al. 1993]

เป็นต้นแบบของฐานความรู้ที่เป็นการใช้รูปแบบการแทนความรู้แบบเป็น Object-Oriented และมีการแบ่งแยกฐานความรู้ออกเป็นส่วนๆ ด้วย โดยจะมีส่วนที่ทางระบบจะทำการบันทึกลงในหน่วยความจำสำรองด้วยในกรณีที่พิจารณาแล้วพบว่าข้อมูลในส่วนนั้นสามารถนำไปใช้งานในโปรแกรมอื่นๆ ได้หลายงาน

จุดเด่นของระบบฐานความรู้ระบบนี้ก็คือ มีการนำเอาแนวความคิดแบบ Object-Oriented มาใช้ประโยชน์ มีการนำเสนอวิธีการที่ใช้ในการจัดการข้อมูลที่เป็นแบบกลุ่มข้อมูลหรือเป็น Set ซึ่งภาษาโปรแกรมแบบ Object-Oriented ทั่วๆ ไป เช่น C++ ไม่สามารถทำการสนับสนุนโดยตรงได้ และการนำเสนอแนวคิดของการเก็บข้อมูลไว้เพื่อวัตถุประสงค์ในการใช้งานข้อมูลชุดนั้นจากหลายๆ โปรแกรมร่วมกัน รวมทั้งมีการนำเสนอ รูปแบบการเก็บกฎในโครงสร้างการแทนความรู้ที่เป็นแบบ Object-Oriented โดยเก็บให้อยู่ในรูปแบบของ Trigger ที่จะทำงานเมื่อได้รับ message ขึ้นมา

สำหรับข้อจำกัดของระบบนี้ก็คือ ข้อมูลในส่วนที่มีการบันทึกเก็บไว้นั้นไม่ได้จัดเก็บไว้ในระบบจัดการฐานข้อมูลทำให้การจัดการเกี่ยวกับตัวข้อมูลทำได้ไม่สะดวกเท่ากับการใช้ระบบจัดการฐานข้อมูลซึ่งรองรับการทำงานในส่วนนี้ได้มากกว่า และยังเป็นผลให้ขอบเขตการในการนำระบบไปใช้ประโยชน์ค่อนข้างจำกัดกว่าด้วย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.7.3. ระบบฐานความรู้ในระบบสารสนเทศทางการแพทย์ใน [Abawajy,Shepard 1994]

เป็นการนำเอาฐานความรู้มาประยุกต์ใช้กับระบบฐานข้อมูลต่างๆ ที่มีอยู่เพื่อใช้ประโยชน์ในการสืบค้นเพื่อหาข้อมูลที่เกี่ยวข้อง โดยรูปแบบการแทนความรู้ที่ใช้ในฐานความรู้ชนิดนี้จะมีลักษณะเป็นข่ายความหมายที่มีการใช้ระบบฐานข้อมูลหลายๆ ระบบร่วมกัน โดยในการจัดสร้างระบบจะใช้ประโยชน์จากการพัฒนาในรูปแบบ Object-Oriented ทั้งในส่วนของฐานความรู้ (ใช้ C++) และระบบจัดการฐานข้อมูล (ONTOS) ที่ใช้เพื่อทำการสร้างฐานความรู้ที่เป็นรูปแบบข้างต้น

วิธีการพัฒนาระบบที่นำเสนอขึ้นมานี้จะทำการสร้างในส่วนของระบบฐานข้อมูลขึ้นมาก่อน จากนั้นจึงทำการสร้างส่วนของระบบฐานความรู้ขึ้นมาภายหลัง ซึ่งข้อดีของระบบนี้ก็คือ เรายังสามารถนำระบบฐานข้อมูลที่เป็นส่วนหนึ่งของระบบไปใช้ต่างหากแบบเป็นอิสระได้ แต่ก็ทำให้เกิดข้อเสียด้วยเช่นกัน เนื่องจากหากเราทำการเปลี่ยนแปลงโครงสร้างของฐานข้อมูลตัวใดตัวหนึ่งแล้วก็จะต้องทำการแก้ไขในส่วนของฐานความรู้ด้วย

3.7.4. CLUES [Telezadeh et al. 1995]

เป็นระบบผู้เชี่ยวชาญที่มีการพัฒนาขึ้นและใช้งานจริงเพื่อช่วยในการตัดสินใจในการปล่อยเงินกู้ เนื่องจากในการทำงานเพื่อพิจารณาปล่อยเงินกู้ให้แก่ลูกค้าแต่ละรายนั้นจะใช้เวลาในการพิจารณาที่ยาวนานเนื่องจากต้องทำการพิจารณาข้อมูลที่เกี่ยวข้องซึ่งมีรายละเอียดปลีกย่อยค่อนข้างมาก รูปแบบสถาปัตยกรรมของระบบก็เป็นการใช้รูปแบบการแทนความรู้ด้วยกฎเนื่องจากทางผู้พัฒนาเห็นว่าสามารถที่จะย้อนดูผลการพิจารณาได้ว่าที่ได้ผลการตัดสินใจออกนั้นเป็นแบบนั้นได้มาจากการพิจารณาอย่างไรซึ่งถือว่าเป็นข้อได้เปรียบของระบบผู้เชี่ยวชาญเมื่อเปรียบเทียบกับระบบอื่นๆ ที่สามารถตรวจสอบผลการทำงานได้ ในขณะที่หากเป็นการใช้เทคนิคแบบอื่นๆ เช่น Neural Network หรือ Genetic Algorithm การตรวจสอบความถูกต้องของผลที่ได้นั้นจะทำได้ลำบากหรือทำไม่ได้เลย ซึ่งระบบที่ได้พัฒนาขึ้นมานี้ก็มีการติดต่อกับฐานข้อมูลด้วยเพื่อทำการตรวจดูประวัติการกู้ยืมเงินของลูกค้าเพื่อใช้ประกอบการพิจารณาเพื่อตัดสินใจการปล่อยกู้ในครั้งปัจจุบัน

จุดเด่นของระบบนี้ก็คือจะมีโครงสร้างของระบบที่สามารถเรียกใช้งานจากระยะไกลได้ผ่านทางสายสัญญาณสื่อสารด้วยการใช้โมเด็ม โดยในส่วนของสาขาของเครือข่ายก็ทำการส่งข้อมูลที่ใช้ในการพิจารณาซึ่งจะอยู่ในรูปแบบของแฟ้มข้อมูลที่เป็นรายงานเกี่ยวกับรายละเอียดการขอกู้เข้ามาทำการประมวลผลที่ส่วนกลางซึ่งจะเก็บโปรแกรมระบบผู้เชี่ยวชาญที่ใช้ในการตัดสินใจรวมทั้งฐานข้อมูลที่เก็บข้อมูลที่ต้องใช้ในการพิจารณาด้วยจากนั้นก็จะทำการส่งเฉพาะข้อมูลส่วนที่เป็นผลการพิจารณากลับไปยังสาขาที่ทำการส่งข้อมูลมาในรูปแบบของแฟ้มรายงานซึ่งสามารถเลือกได้หลายแบบ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.7.5. GermWatcher [Steib 1996]

เป็นระบบผู้เชี่ยวชาญเพื่อช่วยวินิจฉัยในงานทางด้านการแพทย์ที่พัฒนาขึ้นและใช้งานจริงเช่นเดียวกับ CLUES โดยในการพัฒนาช่วงแรกๆ นั้นจะยังไม่มีการใช้ระบบจัดการฐานข้อมูลมาใช้งานในระบบ ซึ่งทำให้เกิดปัญหาขึ้นเนื่องจากการบริหารข้อมูลทำได้ยากลำบาก ต่อมาจึงมีการนำระบบจัดการฐานข้อมูลมาใช้งานซึ่งก็ทำให้การบริหารข้อมูลต่างๆ ทำได้ดีขึ้น และยังช่วยให้ขอบเขตความสามารถในของระบบเพิ่มขึ้นด้วย ซึ่งก็เป็นตัวอย่างของข้อดีของการใช้ระบบจัดการฐานข้อมูลเข้ามาใช้ในงานร่วมกับระบบผู้เชี่ยวชาญที่ชัดเจนตัวอย่างหนึ่ง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 4

สถาปัตยกรรมหลักของระบบ

4.1. บทนำ

สำหรับในบทที่ผ่านๆ มานั้นจะเป็นการอธิบายถึงทฤษฎีต่างๆ ที่เกี่ยวข้อง และตัวอย่างของการประยุกต์ใช้ระบบผู้เชี่ยวชาญร่วมกับฐานข้อมูลบางแบบไปแล้ว สำหรับในบทนี้จะเป็นการนำเสนอและอธิบายรูปแบบสถาปัตยกรรมที่ใช้งานในโครงงานนี้ โดยก่อนอื่นในส่วนของบทนี้ จะขออธิบายสรุปถึงรูปแบบสถาปัตยกรรมต่างๆ ที่มีอยู่เสียก่อนในหัวข้อต่อไปของบทนี้ หลังจากนั้นจึงค่อยนำเสนอรูปแบบสถาปัตยกรรมที่เลือกใช้ รวมไปถึงหลักในการพิจารณาต่างๆ ที่เกี่ยวข้องเมื่อได้เลือกใช้รูปแบบสถาปัตยกรรมที่โครงงานได้นำเสนอ และตัวอย่างการนำไปใช้งานจริงด้วย

4.2. รูปแบบสถาปัตยกรรมแบบต่างๆ

สำหรับรูปแบบการแยกสถาปัตยกรรมของระบบที่มีการใช้ระบบผู้เชี่ยวชาญร่วมกันกับระบบจัดการฐานข้อมูลนั้น มีการจัดไว้อยู่หลายรูปแบบด้วยกัน ดังที่ได้มีการอธิบายย่อๆ ไว้จากบทก่อนๆ แล้ว สำหรับในส่วนนี้จะเป็นการอธิบายในส่วนที่เป็นรายละเอียดมากขึ้น โดยหลักการแบ่งรูปแบบจะแบ่งออกเป็นสองแบบใหญ่ๆ ด้วยกัน คือ

- แบบที่สร้างให้ระบบผู้เชี่ยวชาญและระบบฐานข้อมูลอยู่แยกจากกัน (Heterogeneous approach)
- แบบที่สร้างให้ระบบผู้เชี่ยวชาญและระบบฐานข้อมูลรวมอยู่ด้วยกัน (Homogeneous approach)

โดยในแต่ละรูปแบบสถาปัตยกรรมจะมีรายละเอียดต่างๆ เพิ่มเติมดังที่จะได้อธิบายดังนี้

4.2.1. สถาปัตยกรรมแบบแยกส่วน

สำหรับรูปแบบวิธีการแบบนี้จะเป็นการแยกส่วนของระบบผู้เชี่ยวชาญและฐานข้อมูลออกจากกัน โดยรูปแบบที่นิยมใช้กันก็คือการสร้างระบบผู้เชี่ยวชาญเป็นส่วนที่ทำการติดต่อกับผู้ใช้ ในส่วนนี้ซึ่งจะเก็บความรู้ประเภทกฎ ไว้ทั้งหมด และเมื่อมีการอ้างถึงความรู้ประเภทข้อเท็จจริงที่มีการเก็บไว้ในฐานข้อมูล ทางตัวระบบผู้เชี่ยวชาญจะมีการร้องขอข้อมูลจากฝั่งของฐานข้อมูลให้ส่งข้อมูลมา ซึ่งก็เปรียบได้กับการปรับปรุงส่วนของระบบผู้เชี่ยวชาญให้มีความสามารถในการเรียกใช้ข้อมูลจากหน่วยความจำสำรองนั่นเอง เรียกวิธีแบบนี้ว่าเป็นการรวมกันแบบหลวมๆ (Loosely Coupled)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

อย่างไรก็ตาม ในการรวมแบบวิธีนี้ก็มีอุปสรรคอันเนื่องมาจากความแตกต่างของทฤษฎีและกลไกในการทำงานของระบบทั้งสองแบบที่แตกต่างกันเล็กน้อย คือสำหรับในทฤษฎีที่ใช้ในระบบผู้เชี่ยวชาญนั้น เป็นรูปแบบที่มีอัลกอริธึมแบบ Interpreter สำหรับทำงานกับข้อเท็จจริงเพียงทีละตัว (Tuple-Oriented Algorithm) และลำดับ (Order) ของการประมวลหาข้อเท็จจริงก็มีผลต่อการคำนวณหาคำตอบด้วย ในขณะที่ระบบฐานข้อมูลนั้น ตามทฤษฎีแล้ว การเก็บข้อมูลและการประมวลผลจะอยู่รูปของเซ็ต (Set) หรือรีเลชัน (Relation) ซึ่งประกอบด้วยข้อมูลที่อาจมีมากกว่าหนึ่งตัวและไม่มีการคำนึงถึงลำดับของกลุ่มข้อมูล การประมวลผลหาคำตอบของคำถามที่รับเข้ามาจึงเป็นการประมวลผลที่อิงกับข้อมูลชนิดเซ็ต (Set-Oriented Algorithm) ทางผู้ที่ทำการสร้างระบบโดยยึดเอาสถาปัตยกรรมรูปแบบนี้จึงต้องคิดวิธีการในการกำจัดข้อขัดแย้งที่เกิดขึ้นนี้เพื่อให้ระบบทั้งสองสามารถทำงานด้วยกันได้ แต่ผลที่ตามมาก็คือระบบจะมีประสิทธิภาพในการทำงานที่แย่มากเนื่องจากต้องมีการติดต่อกับแหล่งข้อมูลภายนอกทุกครั้งที่มีการต้องการเรียกใช้ข้อเท็จจริงเพื่อทำการประมวลผลซึ่งก็หมายความว่าต้องมีการดึงข้อมูลจากดิสก์ซึ่งจะมีการทำงานที่ช้ามากเมื่อพิจารณาว่าเป็นกระบวนการที่ทำเพียงเพื่อดึงเอาข้อเท็จจริงเพียงตัวเดียวมาใช้ จะเห็นได้เลยว่าเป็นแนวทางที่ไม่คุ้มค่าเลย ดังนั้น หลักสำคัญของสถาปัตยกรรมในรูปแบบนี้จึงอยู่ที่การออกแบบส่วนที่ทำหน้าที่เชื่อมต่อ (Interface) กันระหว่างระบบทั้งสองเพื่อแก้ปัญหาที่เกิดขึ้นดังที่ได้อธิบายไว้แล้วให้มีการทำงานที่ก่อให้เกิดประสิทธิภาพสูงที่สุด แนวทางที่นำมาใช้ในการแก้ปัญหาจะแบ่งออกเป็น 2 แนวทางหลักๆ คือ แบบการใช้ Interpreter และ แบบการใช้ Compiler ซึ่งก็อาจมีบางระบบที่จะใช้แนวทางทั้งสองอย่างนี้ผสมผสานกัน

4.2.1.1 แนวทางแบบการใช้ Interpreter

ในแนวทางนี้จะมีการสร้างการกันหน่วยความจำหลักไว้ส่วนหนึ่งสำหรับเก็บข้อมูลที่ได้เรียกเข้ามาไว้ชั่วคราว โดยข้อมูลนี้อาจเป็นคำตอบของปัญหาย่อยๆ ที่จำเป็นต้องเก็บไว้ใช้ต่อไป ซึ่งถ้าระบบผู้เชี่ยวชาญต้องการใช้ข้อมูลนี้อีกครั้งก็ไม่ต้องเสียเวลาในการหาจากระบบฐานข้อมูลอีก

4.2.1.2. แนวทางแบบการใช้ Compiler

ในแนวทางนี้จะทำการดูรูปแบบของกฎก่อนว่าในการอนุมานคำตอบของกฎนั้น มีข้อเท็จจริงอะไรบ้างที่จำเป็นต้องดึงออกมาจากฐานข้อมูลภายนอก และจะพยายามรวบรวมข้อเท็จจริงที่ต้องการให้ได้มากที่สุดและส่ง Query ให้มีจำนวนน้อยที่สุด เพื่อให้จำนวนครั้งของการติดต่อกับฐานข้อมูลภายนอกเกิดขึ้นน้อยที่สุด วิธีนี้จะทำให้การประสิทธิภาพการทำงานของระบบเพิ่มขึ้นอย่างมาก แต่ในทางปฏิบัติแล้ว เป็นไปได้ยากที่จะสามารถทำการคอมไพล์กฎทั้งหมดให้เสร็จก่อนแล้วค่อยทำการส่ง Query ไป และถ้ามีการออกแบบที่ไม่ดีก็อาจทำให้เกิดการทำงานแบบ Infinite Loop ได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.2.2. สถาปัตยกรรมแบบรวมเป็นส่วนเดียว

ในระบบแบบนี้จะเป็นการพยายามรวมเอาส่วนของที่เก็บข้อมูลข้อเท็จจริงและส่วนที่เป็นกฎไว้ เรียกว่าเป็นการรวมกันอย่างเหนียวแน่น (Tightly-couple) ซึ่งจากการที่ในช่วงหลังๆ เทคโนโลยีทางด้านระบบจัดการฐานข้อมูลเชิงสัมพันธ์ได้มีการพัฒนาขึ้นเร็วกว่าทางด้านระบบผู้เชี่ยวชาญมาก มีการนำระบบไปประยุกต์ใช้งานในเชิงธุรกิจขนาดใหญ่อย่างได้ผลและมีการพัฒนาต่อให้สามารถทำงานเป็นระบบเครือข่าย เช่น โครงสร้างแบบ Client-Server ทำให้ในช่วงหลังๆ รูปแบบการพัฒนาของระบบฐานข้อมูลผู้เชี่ยวชาญส่วนใหญ่ได้พัฒนาให้อยู่ในแบบของสถาปัตยกรรมนี้ ซึ่งก็ได้เป็นรูปแบบหลักที่มีการพัฒนาและวิจัยอย่างต่อเนื่องและสม่ำเสมอมากกว่าแบบอื่นๆ

ระบบสถาปัตยกรรมแบบนี้จะเป็นที่รู้จักกันทั่วไปในชื่อของระบบฐานข้อมูลอนุมาน (Deductive database) หรือระบบฐานข้อมูลชาญฉลาด (Intelligent Database) โดยสถาปัตยกรรมแบบนี้สามารถแบ่งย่อยลงไปได้อีกสองแบบตามวิธีการที่ใช้ในการพัฒนา คือ

4.2.2.1 แบบทำการขยายส่วนของระบบผู้เชี่ยวชาญให้มีความสามารถของระบบฐานข้อมูล (Elementary Deductive Database)

เป็นการสร้างส่วนขยายเพิ่มเติมจากภาษา Prolog ให้มีความสามารถในการจัดการด้านฐานข้อมูลได้เช่นเดียวกับที่ระบบจัดการฐานข้อมูลมี ได้แก่ ส่วนของภาษาที่ใช้ในการกำหนดรูปแบบของข้อมูล (Data Definition Language : DDL), ส่วนที่เกี่ยวข้องกับการดูแลความปลอดภัยของข้อมูล (Security), ส่วนที่เกี่ยวข้องกับความถูกต้องของข้อมูลและข้อบังคับในการจัดเก็บข้อมูล (Data integrity), ความสามารถในการประมวลผลข้อมูลหลายๆ กระบวนการในเวลาเดียวกัน (Concurrency), ความสามารถในการให้บริการผู้ใช้ได้หลายๆ คนพร้อมกัน (User sharing), ความสามารถในการสำรองข้อมูลและกู้ข้อมูลกลับคืนมากรณีที่ระบบเกิดความเสียหาย (Backup and recovery) ซึ่งนับว่าเป็นงานที่ค่อนข้างหนักทีเดียวในการสร้างระบบผู้เชี่ยวชาญให้มีความสามารถเหล่านี้โดยมีประสิทธิภาพที่ใกล้เคียงหรือเทียบเท่ากับระบบจัดการฐานข้อมูล

4.2.2.2. แบบทำการขยายส่วนของระบบฐานข้อมูลให้มีความสามารถในการอนุมาน (Advanced Deductive Database)

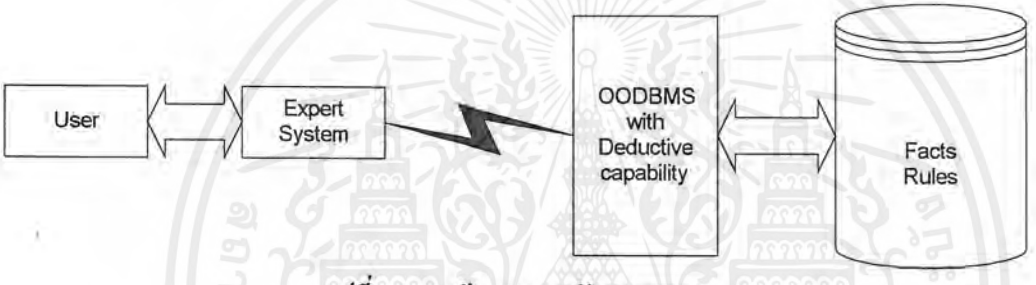
จากการที่ระบบผู้เชี่ยวชาญสามารถตอบคำถามบางรูปแบบซึ่งไม่สามารถหาคำตอบได้ด้วยการใช้ Query ของระบบฐานข้อมูลแบบ Relational ทั่วๆ ไปได้ โดยเฉพาะคำถามแบบ Recursive ทำให้มีการพยายามขยายความสามารถของระบบฐานข้อมูลที่มีอยู่เดิมให้มีความสามารถในส่วนนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

แต่เนื่องจากข้อจำกัดในเรื่องของรูปแบบทฤษฎีของระบบทั้งสองดังที่ได้อธิบายไว้ในข้างต้นแล้ว ทำให้เราไม่สามารถใช้ทฤษฎีทางตรรกศาสตร์ที่ใช้ในระบบผู้เชี่ยวชาญ คือ First-Order Predicate Logic ในการกำหนดส่วนของกฎที่ใช้อนุมานโดยตรงกับฐานข้อมูลได้ จึงมีการดัดแปลงทฤษฎีตรรกศาสตร์ดังกล่าวให้มีความสอดคล้องกับทฤษฎีของแบบจำลองข้อมูลแบบ Relational เพื่อสร้างเป็นภาษาที่ใช้ในการจัดการฐานข้อมูล โดยอาจมีการลดความสามารถทางด้านตรรกศาสตร์ลงด้วย เช่น Datalog ซึ่งมีลักษณะของภาษาตามที่อธิบายไว้ในบทที่ 2 แล้ว

4.3. รูปแบบสถาปัตยกรรมที่ได้นำเสนอในโครงงาน

สำหรับในโครงงานนี้จะทำการนำเสนอแนวความคิดของสถาปัตยกรรมแบบใหม่ขึ้นมาคือจะเป็นรูปแบบสถาปัตยกรรมที่เป็นการผสมผสานรูปแบบของสถาปัตยกรรมแบบหลักๆ ทั้งสองแบบเข้าไว้ด้วยกัน คือ จะเก็บส่วนของกฎทั้งในส่วนของระบบผู้เชี่ยวชาญและในส่วนของระบบฐานข้อมูล ดังรูปที่ 4-1



รูปที่ 4-1 สถาปัตยกรรมหลักของระบบ

ซึ่งจากสถาปัตยกรรมแบบที่ได้นำเสนอนี้จะทำให้เราลดข้อเสียที่เกิดจากสถาปัตยกรรมรูปแบบเดิมๆ คือจากถ้าเป็นแบบที่แยกส่วนของกฎออกมาต่างหากจะทำให้การเรียกบริการขอข้อมูลจากฐานข้อมูลจะมีมาก ซึ่งจะทำให้เกิด Traffic ขึ้นมากเกินความจำเป็น แต่ถ้าหากจะรวมส่วนของกฎทั้งหมดไว้ในฐานข้อมูลก็จะทำให้ฐานข้อมูลหรือฐานความรู้มีขนาดใหญ่เกินไปและจะมีกฎบางกฎที่มีการใช้น้อยเนื่องจากเป็นกฎที่จะใช้เป็นการเฉพาะสำหรับงานบางงานเท่านั้นและมีความสัมพันธ์กับตัวข้อมูลที่ถูกเก็บไว้น้อย ดังนั้นหากเราทำการเก็บเฉพาะกฎที่มีความเกี่ยวข้องกับข้อมูลที่มีการเก็บไว้อยู่ในฐานข้อมูล ในส่วนที่ใช้ติดต่อกับผู้ใช้ก็จะมีกฎที่ใช้เฉพาะสำหรับในงานนั้น และการเรียกบริการขอข้อมูลไปยังส่วนของฐานข้อมูลก็สามารถเรียกผ่านโดยการใช้กฎ ซึ่งจะเป็นการลด Traffic ที่เกิดจากการเรียกใช้บริการฐานข้อมูลได้ เนื่องจากการเรียกผ่านทางกฎที่อยู่ในฐานข้อมูลจะเป็นการเรียกข้อมูลที่ผ่านกระบวนการการประมวลผลบางอย่างมาก่อนแล้ว เช่น หากในส่วนของระบบผู้เชี่ยวชาญต้องการข้อมูลชื่อคนที่เป็นยายของบุคคลๆ หนึ่ง แต่ในฐานข้อมูลจะมีการเก็บข้อมูลเฉพาะคนที่เป็นพ่อและแม่ของคนๆ นี้เท่านั้น ถ้าเป็นในกรณีแบบที่ใช้ฐานข้อมูลธรรมดาทางส่วนของระบบผู้เชี่ยวชาญจะต้องทำการร้องขอบริการจากระบบฐานข้อมูลถึงสองครั้งคือ 1) จะเป็นการขอข้อมูลชื่อแม่ของบุคคลที่ต้องการทราบชื่อยาย จากนั้น 2) จะทำการขอข้อมูลชื่อของคนที่เป็นแม่ของคนที่เราได้ชื่อมาจาก 1) แต่ถ้าเราทำการเก็บส่วนของกฎที่จะใช้ทำการหาคนที่เป็นยายของบุคคลที่เราต้องการก็จะลดจำนวนบริการที่เราจะทำการ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ร้องขอให้เหลือเพียงครั้งเดียว จากตัวอย่างนี้จะเห็นได้ว่าขนาดของข้อมูลที่จะส่งผ่านกลับไปยังส่วนของกฎที่เก็บไว้ที่ส่วนติดต่อกับผู้ใช้เพื่อทำการประมวลผลต่อมีน้อยลง

4.4. ลักษณะ รูปแบบ ของกฎแบบต่างๆ ที่สามารถเก็บไว้ที่ทางด้านฐานข้อมูล

จากตัวอย่างที่ได้แสดงให้เห็นแล้วในหัวข้อที่ผ่านมา คงจะทำให้พอจะเริ่มประโยชน์ของการเก็บกฎต่างๆ ไว้ที่ส่วนของฐานข้อมูลบ้างแล้ว ปัญหาต่อมาที่จะต้องนำมาพิจารณาก็คือ กฎแบบไหนที่เราควรจะเก็บไว้ที่ส่วนของระบบผู้เชี่ยวชาญและส่วนไหนควรจะเก็บไว้ที่ฐานข้อมูล ในส่วนต่อไปจะเป็นการอธิบายถึงแนวทางที่ใช้ในการพิจารณาถึงรูปแบบของกฎต่างๆ และส่วนที่กฎเหล่านี้ควรจะถูกนำไปเก็บไว้โดยจะพิจารณาในมุมมองของฐานข้อมูลเป็นหลัก

สำหรับในส่วนของกฎต่างๆ ที่สามารถจะเก็บไว้ในส่วนของฐานข้อมูลนั้นแนวทางหลักๆ ที่ใช้ในการพิจารณาก็คือ ในตอนแรกจะดูถึงความเกี่ยวข้องของกฎนั้นๆ กับตัวข้อมูลที่เก็บไว้ในฐานข้อมูลหรือไม่ จากนั้นก็พิจารณาต่ออีกว่ากฎนั้นๆ ถ้าใส่เข้าไปในฐานข้อมูลแล้วจะมีโอกาสที่จะได้ใช้กฎในส่วนนั้นบ่อยหรือไม่ คือกฎนั้นมีแนวโน้มของโอกาสที่จะมีการนำไปใช้ในงานส่วนอื่นๆ ได้อีก ไม่ได้ใส่เข้าไปเพื่อนำไปใช้กับงานใดงานหนึ่งเป็นการเฉพาะ มีประโยชน์มากพอที่จะนำมาเก็บไว้ในฐานข้อมูลเพื่อนำไปใช้งานในครั้งต่อๆ ไป และมีการเปลี่ยนแปลงที่ค่อนข้างน้อยหรือจะไม่มีการเปลี่ยนแปลงเลย

จากแนวทางคร่าวๆ ที่ได้กำหนดไว้ข้างต้นนี้ ก็พอจะนำมาแบ่งเป็นลักษณะใหญ่ๆ ของกฎที่สามารถเก็บไว้ในฐานข้อมูล ซึ่งเป็นลักษณะเดียวกันกับกฎรูปแบบต่างๆ ที่ได้นำเสนอไว้ใน [Ceri,Ramakrishnan 1996] คือจะแบ่งเป็น 1) Active Rule และ 2) Deductive Rule ซึ่งก็คือรูปแบบของกฎที่นำมาจากในระบบผู้เชี่ยวชาญเพื่อประยุกต์ใช้กับระบบจัดการฐานข้อมูล สำหรับรายละเอียดของกฎในแต่ละลักษณะ และหลักในการพิจารณาเพื่อนำกฎเหล่านี้มาไว้ในฐานข้อมูล จะมีรายละเอียดอยู่ในหัวข้อต่อไป

4.4.1 Active Rule

คือ กฎในลักษณะที่จะเป็นกระบวนการต่างๆ ที่จะทำงานเมื่อมีเหตุการณ์ (Event) ที่มีเงื่อนไขตรงตามที่กำหนดไว้ ซึ่งมีชื่อเรียกอื่นๆ อีก คือ Trigger และ Demon ซึ่งการใช้งานของกฎต่างๆ เหล่านี้ในระบบฐานข้อมูลก็มีในเรื่องของการรักษาความถูกต้องของข้อมูล (Integrity Maintenance) ซึ่งกฎต่างๆเหล่านี้ถือว่าเป็นกฎที่ค่อนข้างผูกติดกับตัวของข้อมูลอยู่แล้ว เราสามารถพิจารณาได้ง่ายๆ ว่า หากเราแยกกฎเหล่านี้ไปไว้ในส่วนอื่นที่ๆไม่ใช่ตัวของฐานข้อมูลแล้ว เราจะพบว่าเมื่อเรามีการเปลี่ยนแปลงหรือแก้ไขข้อมูลในฐานข้อมูลแล้ว โอกาสที่จะเกิดความไม่ถูกต้อง หรือ กลุ่มของข้อมูลที่เก็บอาจมีความขัดแย้งกันเอง (Inconsistent) เกิดขึ้นในฐานข้อมูลได้ ซึ่งกฎแบบนี้ก็มีเป็นประเภทเดียวกันกับกฎที่มีอยู่ในผลิตภัณฑ์ระบบจัดการฐานข้อมูลในชื่อที่รู้จักกันว่า "Business Rule" ตัวอย่างของกฎต่างๆ เหล่านี้ได้แก่

- ข้อมูลที่จะเก็บไว้ในส่วนที่เป็นเพศของบุคคลจะต้องมีค่าเป็น 'Male' หรือ 'Female' เท่านั้น
- กฎที่ใช้ในการทำ Rollback เมื่อ Transaction ที่กำลังทำงานอยู่เกิดความผิดพลาด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.4.2 Deductive Rule

คือ กฎที่ใช้ในการอนุมานหาข้อเท็จจริงใหม่ๆ จากข้อมูลที่มีการเก็บไว้อยู่แล้วในฐานข้อมูล โดยไม่มีความจำเป็นที่จะต้องเก็บข้อมูลตัวนั้นจริงๆ เมื่อทราบถึงลักษณะของกฎประเภทนี้แล้ว จะพบว่าปัญหาที่เกิดขึ้นก็คือ เรามีหลักการอะไรที่ใช้ในการแยกว่ากฎนี้ควรจะมีอยู่ในฐานข้อมูล ซึ่งสำหรับในบทความชิ้นเดียวกันนี้ ก็ได้อ้างถึงเรื่อง Knowledge Independence ซึ่งจะใช้การพิจารณาว่าหากกฎนั้นมีการใช้โดย Application หลายๆ ตัว ก็ควรเก็บกฎนั้นไว้ที่ฐานข้อมูล โดยในโครงงานฉบับนี้ จะยกตัวอย่างกฎที่ควรเก็บไว้ในฐานข้อมูลและเก็บไว้ในระบบผู้เชี่ยวชาญดังนี้

- ในระบบฐานข้อมูลที่เราพัฒนาขึ้นมานั้น ถ้าเราพบว่าโปรแกรมต่างๆแต่ละโปรแกรมที่มีการเรียกใช้ระบบฐานข้อมูลนี้ (ซึ่งวัตถุประสงค์ของระบบฐานข้อมูลนี้ก็คือสร้างขึ้นมาเพื่อรองรับการให้บริการแก่ระบบผู้เชี่ยวชาญที่ใช้งานในส่วนที่แตกต่างกันไปแต่มีความต้องการข้อมูลที่คล้ายๆ กัน แต่ขอบเขตของโครงงานนี้จะเป็นการนำเสนอรูปแบบที่ใช้ในการติดต่อกันระหว่างฐานข้อมูลและระบบผู้เชี่ยวชาญจึงขอไม่อธิบายในรายละเอียด) ถ้าหากมีการใช้กฎเพื่อเรียกข้อมูลแบบเดียวกันแล้ว ซึ่งกฎประเภทนี้จะเป็นกฎที่ทำงานเกี่ยวข้องกับข้อมูลในอดีตที่เคยมีการถามมาแล้ว และมีการเก็บข้อมูลส่วนนั้นไว้ในฐานข้อมูล ก็ควรเก็บกฎนั้นไว้ที่ด้านของฐานข้อมูล ตัวอย่างของกฎประเภทนี้ เช่น
  - กฎที่จะบอกว่าถ้าพนักงานคนใดเข้าทำงานไม่เคยขาดหรือสายเลยตลอดทั้งเดือน ก็จะมีการเพิ่มเบี้ยขยันไว้ในเงินเดือนที่จะจ่ายให้พนักงานในเดือนนั้น
  - กฎที่จะบอกว่าถ้าอาจารย์ท่านใดมีชั่วโมงสอนเกินจากที่กำหนดก็จะมีการเพิ่มเงินเดือนให้ในเดือนนั้น

- กฎบางอย่างซึ่งจะเป็นกฎประเภทที่ต้องทำงานกับข้อมูลปัจจุบัน คือเป็นข้อมูลที่ต้องถามจากผู้ใช้ (ad hoc) ก็จะเก็บกฎนั้นไว้ที่ส่วนของระบบผู้เชี่ยวชาญ ตัวอย่างของกฎแบบนี้ที่พบบ่อยก็คือกฎประเภทที่เป็นการถามถึงสถานการณ์ปัจจุบันที่ระบบผู้เชี่ยวชาญจำเป็นต้องทราบจากผู้ใช้เพื่อให้คำตอบที่ใช้ในการแก้ปัญหาที่เกิดขึ้น ตัวอย่างเช่น
  - กฎที่จะบอกว่าควรจะจ่ายยาอะไรถ้าผู้ป่วยมีอาการตามที่บอกมา

จากการพิจารณาข้างต้นที่ผ่านมาแล้วนั้น ก็จะได้หลักการที่จะใช้ในการแยกว่ากฎใดที่จะเก็บไว้ในฐานข้อมูล ปัญหาที่ตามมาอีกก็คือ เราจำเป็นหรือไม่ที่จะต้องเก็บกฎเหล่านี้ (หรือในอีกนัยหนึ่งก็คือความรู้) ไว้ในฐานข้อมูล ในเมื่อเราก็สามารถเก็บให้อยู่ในรูปแบบของข้อมูลธรรมดาก็ได้ ในส่วนต่อไปนี้จะเป็นการพิจารณาเปรียบเทียบถึงความแตกต่างระหว่างการเก็บความรู้ในรูปแบบของกฎเ ะการเก็บไว้ในแบบของข้อมูลธรรมดาโดยจะพิจารณาถึงตัวอย่างที่จะใช้ประกอบการอธิบายต่อไป

- การเก็บความรู้ที่จะบอกว่าถ้ามีใครคนหนึ่งชอบเล่นเทนนิสเป็นงานอดิเรก คนๆนั้นจะชอบเล่นแบดมินตันเป็นงานอดิเรกด้วย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เมื่อพิจารณาถึงการเก็บข้อมูลที่เป็นการเก็บด้วยการใช้ข้อมูลแบบธรรมดาสิ่งที่ต้องทำเพื่อให้ข้อความข้างต้นเป็นจริงก็คือ ทำการเพิ่มข้อมูลให้กับทุกๆ คนที่ชอบเล่นเทนนิสเป็นงานอดิเรกว่าชอบเล่นแบดมินตันด้วย ในขณะที่ถ้าเป็นการเก็บด้วยการใช้กฎก็จะมีเพียงแค่การสร้างกฎที่จะเก็บไว้เพียงกฎเดียว และถ้าหากมีการเปลี่ยนแปลงแก้ไขข้อมูลเกี่ยวกับงานอดิเรกที่เป็นการเล่นเทนนิส ในการเก็บข้อมูลแบบธรรมดาก็จำเป็นต้องมีการแก้ไขข้อมูลงานอดิเรกที่เป็นการเล่นแบดมินตันด้วย ซึ่งจะมีโอกาสของความผิดพลาดที่จะเกิดขึ้นจากความไม่ถูกต้องของข้อมูลได้ และยังอาจมองได้ว่าความซ้ำซ้อนอีกแบบหนึ่งเนื่องจากมีการเก็บข้อมูลที่ซ้ำๆ กัน

ลองพิจารณาถึงตัวอย่างต่อไป เพื่อให้เห็นถึงลักษณะของความซ้ำซ้อนที่ชัดเจนขึ้น

- การเก็บความรู้ที่ใช้ในการบอกชื่อยายของคนๆนั้น

จากตัวอย่างนี้จะเห็นได้อย่างชัดเจนว่าถ้าเก็บเป็นข้อมูลแบบธรรมดาแล้วจะทำให้เกิดความซ้ำซ้อนของข้อมูลอย่างมาก (เว้นแต่ว่าฐานข้อมูลนั้นมีการเก็บข้อมูลของบุคคลที่ไม่มีบุคคลใดที่เก็บไว้ในฐานข้อมูลเป็นแม่ลูกกันเลย) และความยุ่งยากก็จะเกิดขึ้นอีกถ้ามีการเปลี่ยนแปลงข้อมูลเกี่ยวกับแม่หรือย่าของบางคน ก็จำเป็นต้องมีการแก้ไขข้อมูลจุดอื่นๆ ด้วย

แต่เมื่อลองพิจารณาถึงกรณีที่เมื่อกฎมีการเปลี่ยนแปลงขึ้น เช่น เกิดมีการเปลี่ยนแปลงขึ้นมาว่า คนที่ชอบเล่นเทนนิสเป็นงานอดิเรกไม่จำเป็นต้องชอบเล่นแบดมินตันเป็นงานอดิเรกเสมอไป ก็จะทำให้ฐานข้อมูลมีการเปลี่ยนแปลงข้อมูลอย่างมาก ดังนั้นในการพิจารณาว่าจะเก็บข้อมูลเป็นกฎหรือไม่ ปัจจัยหนึ่งที่ใช้พิจารณาด้วยก็คือ กฎต่างๆ เหล่านั้นมีโอกาสที่จะเกิดการเปลี่ยนแปลงน้อยหรือไม่มีเลย

4.5. กรณีศึกษา : ตัวอย่างของกฎที่สามารถเก็บไว้ในระบบฐานข้อมูลนักศึกษาและอาจารย์

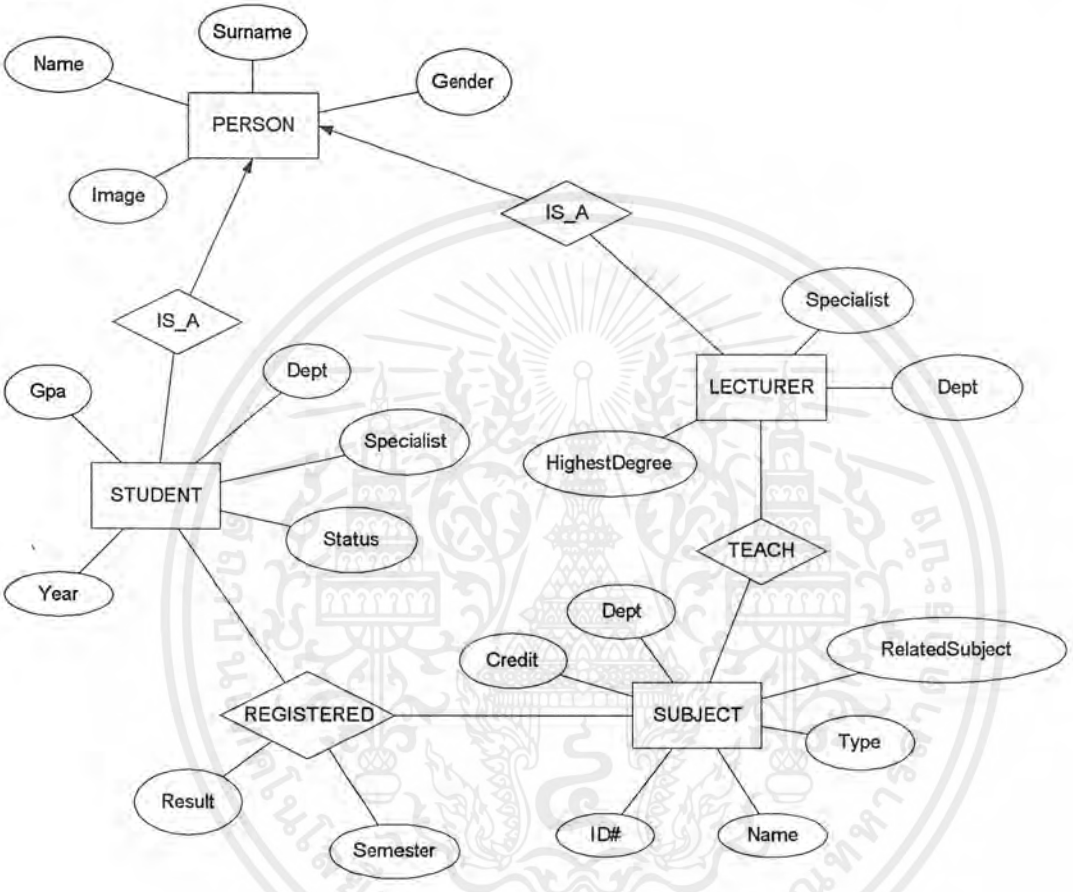
ในส่วนนี้เพื่อให้เป็นการมองเห็นภาพที่ชัดเจนขึ้น ในส่วนนี้จะเป็นการยกตัวอย่างเพื่อแสดงให้เห็นถึงวิธีการที่ใช้ในการพิจารณาเพื่อทำการเก็บกฎที่เหมาะสมสำหรับเก็บไว้ในฐานข้อมูล โดยในตัวอย่างนี้จะพิจารณาเฉพาะในส่วนของกฎที่เป็น Deductive Rule เนื่องจากทางผู้จัดทำโครงงานเห็นว่าถ้าเป็น Active rule ที่เกี่ยวข้องกับข้อมูลที่เก็บอยู่ในฐานข้อมูล ก็ควรจะเก็บไว้ในฐานข้อมูลทั้งหมดอยู่แล้ว ส่วนตัวอย่างเป็นการอธิบายถึงการสร้างฐานข้อมูลหรือฐานความรู้ที่ใช้เก็บข้อมูลเกี่ยวกับนักศึกษา, อาจารย์และวิชาเรียนในสถาบันการศึกษาระดับอุดมศึกษาเพื่อใช้สำหรับเป็นฐานข้อมูลให้กับระบบผู้เชี่ยวชาญอื่นๆ ที่มีความต้องการใช้ข้อมูลที่มีรายละเอียดเกี่ยวข้องกับอาจารย์หรือนักศึกษา และเป็นฐานข้อมูลที่จะทำการสร้างขึ้นเพื่อใช้ทดลองใช้งานในโครงงานวิจัยนี้ ซึ่งรายละเอียดของข้อมูลที่มีในฐานข้อมูลมีดังนี้

- ข้อมูลที่เกี่ยวกับตัวบุคคล ได้แก่ ชื่อ นามสกุล เพศ
- ข้อมูลที่เป็นส่วนของนักศึกษา ได้แก่ ชั้นปี ภาควิชาที่เรียน วิชาที่ได้ลงทะเบียน ผลการเรียนที่ได้ในแต่ละวิชา ผลการเรียนเฉลี่ย ความชำนาญในด้านวิชาการ สถานะทางด้านการศึกษา
- ข้อมูลที่เป็นส่วนของอาจารย์ ได้แก่ ภาควิชาที่สังกัด วิชาที่เปิดสอน ความเชี่ยวชาญในสาขาวิชา

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- ข้อมูลที่เกี่ยวกับรายชื่อวิชา ได้แก่ รหัสวิชา ชื่อวิชา ลักษณะของวิชา (เช่น เป็นวิชาบังคับของภาค, เป็นวิชาเลือกของภาค หรือเป็นวิชาเลือกทางมนุษย์ศาสตร์, สังคมศาสตร์ เป็นต้น

ต่อไปจะทำการแสดงความสัมพันธ์ต่างๆ ของข้อมูลโดยการใช้แผนภาพ EER (Extended Entity-Relationship) ซึ่งเป็นแผนภาพที่เป็นการทำส่วนขยายเพิ่มเติมมาจาก ER แบบธรรมดา ซึ่งก็มีลักษณะที่ไม่แตกต่างจาก ER แบบธรรมดามากนัก หากมีความรู้เกี่ยวกับ ER อยู่แล้ว ก็สามารถทำความเข้าใจแผนภาพแบบนี้ได้โดยง่าย



รูปที่ 4-2 แผนภาพแสดงโครงสร้างของฐานข้อมูล

จากรูปที่ 4-2 ซึ่งได้แสดงแผนภาพที่ได้ทำการแสดงไว้ข้างต้นนี้เราจะทำการสร้างเป็นโครงสร้างของข้อมูลที่จะเก็บไว้ในฐานข้อมูลโดยจะเป็นการเขียนให้อยู่ในรูปแบบของ F-Logic เพื่อเป็นการแสดงให้เห็นถึงความเป็นไปได้ในทางทฤษฎีที่เราสามารถจะเก็บกฎต่างๆไว้ร่วมกับรูปแบบข้อมูลชนิด Object ได้ และจากนั้นจึงแสดงถึงรูปแบบกฎที่ใช้ในการอนุมานหาข้อเท็จจริงต่างๆ จากข้อมูลตัวอื่นที่มีอยู่โดยไม่ต้องเก็บข้อมูลนั้นไว้จริงๆ ในฐานข้อมูล แต่สำหรับขอบเขตของโครงงานนี้จะไม่ครอบคลุมถึงส่วนของการสร้างระบบฐานข้อมูลที่ใช้ทฤษฎีนี้ แต่จะเน้นในส่วนของหลักการของความเป็นไปได้และประโยชน์ในการเก็บส่วนของกฎไว้ที่ฐานข้อมูล และวิธีการที่ใช้ในการติดต่อของระบบผู้เชี่ยวชาญเพื่อขอข้อมูลจากฐานข้อมูลเข้ามาทำงาน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ก่อนอื่นจะเป็นการแสดงถึงลำดับชั้นของ Class ที่แสดงถึงความสัมพันธ์กันระหว่าง Class ต่างๆ ที่ได้ออกแบบมา ซึ่งจะได้ความสัมพันธ์ออกมาดังนี้

student :: person
lecturer :: person

จากตัวอย่างจะเป็นการแสดงถึงความสัมพันธ์ว่าสำหรับ Class *student* จะเป็น Subclass ของ Class *person* และ Class *lecturer* ก็เป็น Subclass ของ Class *person* เช่นกัน

สำหรับรายละเอียดของ Class แต่ละตัวสามารถแสดงได้ดังนี้

```
person [    name => string;
            surname => string;
            gender => sex;]

lecturer[   dept => string;
            highestDegree => degree;
            teach =>> subject;]

student[    id => string;
            dept => string;
            year => stdYear;
            gpa => real;
            status => stdStatus;
            specialist =>> string;
            registered =>> regssubj;]

subject[    id => string;
            name => string;
            type => subjType;
            credit => number;
            openDept => string;]

regssubj[   year => integer;
            semester => integer;
            subj => subject;
            grade => string;]
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ต่อไปจะเป็นหลักการในการพิจารณาเพื่อดูว่าข้อมูลตัวใดที่สามารถหามาจากข้อมูลตัวอื่นที่มีอยู่ได้ และไม่จำเป็นต้องมีการเก็บไว้จริงในฐานข้อมูล โดยจะพิจารณาถึงความเหมาะสมของการที่จะเก็บกฎตัวนี้ไว้ในระบบฐานข้อมูลด้วย โดยมีหลักการคือหากเป็นกฎซึ่งใช้ในการอนุมานนี้มีความจำเป็นที่จะเก็บไว้อยู่ในฐานข้อมูลได้เนื่องจากจะมีโปรแกรมหลาย ๆ ตัว ซึ่งอาจจะเป็นโปรแกรม Application ทั่วๆ ไป หรืออาจเป็นโปรแกรมระบบผู้เชี่ยวชาญที่ใช้ทำงานในหน้าที่ต่างๆ ที่จะทำการเชื่อมต่อและร้องขอข้อมูลจากฐานข้อมูล และหากเก็บไว้ในรูปแบบ เช่น โปรแกรม Application ตัวอื่นๆ เราจะเรียกใช้ข้อมูลตัวนี้

จาก Class *student* พิจารณาถึง attribute *dept* ซึ่งจะหมายถึงภาควิชา เราจะพบว่าข้อมูลตัวนี้ควรใส่ไว้ในฐานข้อมูลเพราะโดยปกติเราจะพบว่ามีโปรแกรมที่มีการอ้างอิงถึงฐานข้อมูลนี้จะมีการเรียกใช้ข้อมูลภาควิชาของนักศึกษาเป็นประจำอยู่แล้ว โดยข้อมูลตัวนี้เราสามารถทำการอนุมานมาได้จากข้อมูลของรายละเอียดวิชาที่นักศึกษาคนนั้นได้ทำการลงทะเบียนเรียนโดยจะดูรายละเอียดว่าเป็นวิชาของภาคใดและต้องวิชาที่เปิดสอนเฉพาะในภาคเท่านั้น สามารถเขียนให้อยู่ในรูปแบบของ F-logic ได้ดังนี้

$$S[dept\text{->}\text{“}CE\text{”}] \text{ :- } S[registerd \text{->>} Subj[openDept\text{->} \text{“}CE\text{”}]] \wedge Subj[type \text{->} \text{“}RestrictedSubject\text{”}] \wedge S{:}student \wedge Subj{:}subject$$

จาก Class เดียวอีกนี้ พิจารณาถึง Attribute *specialist* ซึ่งจะหมายถึงรายชื่อวิชาที่เป็นวิชาถนัดของนักศึกษาคนนั้น เราจะพบว่าข้อมูลตัวนี้เราควรจะเก็บไว้ในฐานข้อมูลเนื่องจากว่าจะมีโปรแกรมหลายๆ โปรแกรมนำไปใช้งานได้เช่นระบบผู้เชี่ยวชาญที่เกี่ยวกับการเลือกวิชาลงทะเบียนวิชาเลือกของภาควิชา, ระบบผู้เชี่ยวชาญที่เกี่ยวกับการเลือกอาจารย์ที่ปรึกษาวิชาโครงงาน เป็นต้น ซึ่งสามารถทำการอนุมานมาได้จากข้อมูลของรายละเอียดวิชาที่นักศึกษาคนนั้นได้ทำการลงทะเบียนเรียนและเป็นวิชาของภาคและได้ผลการเรียนเป็น A

$$S[specialist \text{->>} Str] \text{ :- } S[registerd \text{->>} R] \wedge R[subj \text{->} Subj[name \text{->} Str]] \wedge Subj[\ openDept \text{->} D] \wedge S[dept \text{->} D] \wedge R[grade \text{->} \text{“}A\text{”}] \wedge S{:}student \wedge Subj{:}subject \wedge R{:}regssubj \wedge D{:}string \wedge Str{:}string$$

จาก Class *lecturer* พิจารณาถึง Attribute *dept* ซึ่งจะหมายถึงภาควิชา เราจะพบว่าข้อมูลตัวนี้เราสามารถทำการอนุมานมาได้จากข้อมูลของรายละเอียดวิชาสอนได้เช่นกัน

$$L[dept\text{->}\text{“}CE\text{”}] \text{ :- } L[registerd \text{->>} Subj[openDept\text{->} \text{“}CE\text{”}]] \wedge Subj[type \text{->} restrictedSubject] \wedge L{:}lecturer \wedge Subj{:}subject$$

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ส่วน Attribute ตัวอื่นๆ ที่เราสามารถทำการอนุมานได้อีกก็คือ Attribute *status,year* แต่เนื่องจากภาษา F-logic ไม่ได้มีความสามารถในการทำงานทางด้านคณิตศาสตร์ (ซึ่งใน [Liu 1999] ก็ได้มีการชี้ให้เห็นว่าไม่มีภาษาของฐานข้อมูลแบบ Deductive ใดๆ ที่มีความสามารถทางด้านการคำนวณอย่างครบถ้วน) แต่ก็สามารถบอกที่มาที่จะบอกว่าค่าใน Attribute เหล่านี้มีค่าเป็นอะไร โดยพิจารณาจากรายละเอียดดังต่อไปนี้

status คือค่าที่เป็นสถานะทางการศึกษาของนักศึกษาโดยจะพิจารณาจากผลการเรียนเฉลี่ย ควรเก็บไว้ในฐานข้อมูลเนื่องจากมีการใช้งานในหลายด้านเช่น การหาผู้ที่ได้เกียรตินิยม, คนที่กำลังถูกภาคทัณฑ์อยู่ การพิจารณาให้ทุนการศึกษา เป็นต้น มีวิธีการอนุมานโดยใช้หลักการดังนี้

โดยจะมีค่าเป็น *"FirstHonor"* เมื่อค่า *gpa* มากกว่า 3.50

มีค่าเป็น *"Honor"* เมื่อค่า *gpa* อยู่ระหว่าง 3.50 และ 3.25

มีค่าเป็น *"Good"* เมื่อค่า *gpa* อยู่ระหว่าง 3.25 และ 2.75

มีค่าเป็น *"Fair"* เมื่อค่า *gpa* อยู่ระหว่าง 2.75 และ 2.50

มีค่าเป็น *"Poor"* เมื่อค่า *gpa* อยู่ระหว่าง 2.50 และ 2.00

มีค่าเป็น *"Probation"* เมื่อค่า *gpa* ต่ำกว่า 2.00

year คือค่าที่บอกชั้นปีของนักศึกษาโดยจะพิจารณาจากผลการเรียนเฉลี่ย เป็นค่าปกติที่ควรเก็บไว้ในฐานข้อมูลอยู่แล้วมีวิธีการอนุมานโดยใช้หลักการดังนี้

จะมีค่าเป็น *"Freshy"* เมื่อผลต่างระหว่างปีที่มากสุดกับปีที่น้อยสุดของรายวิชาที่ลงทะเบียนเป็น 0 และปีสุดท้ายที่ลงทะเบียนไม่มีรายวิชาที่ลงทะเบียนในภาคการเรียนที่ 2

จะมีค่าเป็น *"Sophomore"* เมื่อผลต่างระหว่างปีที่มากสุดกับปีที่น้อยสุดของรายวิชาที่ลงทะเบียนเป็น 0 และปีสุดท้ายที่ลงทะเบียนมีรายวิชาที่ลงทะเบียนในภาคการเรียนที่ 2

จะมีค่าเป็น *"Sophomore"* เมื่อผลต่างระหว่างปีที่มากสุดกับปีที่น้อยสุดของรายวิชาที่ลงทะเบียนเป็น 1 และปีสุดท้ายที่ลงทะเบียนไม่มีรายวิชาที่ลงทะเบียนในภาคการเรียนที่ 2

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จะมีค่าเป็น *"Junior"* เมื่อผลต่างระหว่างปีที่มากสุดกับปีที่น้อยสุดของรายวิชาที่ลงทะเบียนเป็น 1 และปีสุดท้ายที่ลงทะเบียนมีรายวิชาที่ลงทะเบียนในภาคการเรียนที่ 2

จะมีค่าเป็น *"Junior"* เมื่อผลต่างระหว่างปีที่มากสุดกับปีที่น้อยสุดของรายวิชาที่ลงทะเบียนเป็น 2 และปีสุดท้ายที่ลงทะเบียนมีรายวิชาที่ลงทะเบียนในภาคการเรียนที่ 2

จะมีค่าเป็น *"Senior"* เมื่อผลต่างระหว่างปีที่มากสุดกับปีที่น้อยสุดของรายวิชาที่ลงทะเบียนเป็นมากกว่า 2

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 5
การออกแบบโพรโตคอลที่ใช้ในการรับ-ส่งข้อมูล

5.1 บทนำ

ในบทที่แล้วจะเป็นการอธิบายถึงสถาปัตยกรรมของระบบซึ่งแบบที่เราใช้จะมีการเก็บกฎทั้งในส่วนของระบบผู้เชี่ยวชาญและฐานข้อมูลแล้ว รวมทั้งยกตัวอย่างกฎต่าง ๆ สำหรับในบทนี้จะเป็นการอธิบายถึงวิธีการที่โครงงานนี้ใช้ในการแก้ปัญหาที่เกิดขึ้นเนื่องจากรูปแบบโครงสร้างของข้อมูลที่แตกต่างกัน และต้องทำการแปลงรูปแบบของข้อมูลจากที่เป็นแบบ Object มาเป็นแบบ Tuple ซึ่งเป็นรูปแบบข้อมูลที่ระบบฐานข้อมูลแบบ Relational ใช้เก็บข้อมูล และเป็นรูปแบบข้อมูลที่ระบบผู้เชี่ยวชาญแบบกฎ ซึ่งเป็นระบบผู้เชี่ยวชาญที่โครงงานนี้สามารถใช้ได้ด้วย และถือว่าเป็นรายละเอียดที่สำคัญส่วนหนึ่งของโครงงานนี้นอกจากในส่วนของสถาปัตยกรรมของระบบ คือ ในเรื่องของการจัดการข้อมูลที่มีความซับซ้อนคือในข้อมูลที่ส่งกลับไปอาจมีบาง Field ที่มีลักษณะเป็น Repeating Groups ซึ่งเป็นโครงสร้างที่ไม่ผิดรูปแบบของ Object แต่เป็นรูปแบบที่ผิดสำหรับในฐานข้อมูลแบบ Relational และระบบผู้เชี่ยวชาญที่เป็นแบบกฎใช้ในการแทนรูปแบบข้อมูล ลักษณะสำคัญอีกอย่างหนึ่งก็คือการที่ข้อมูลที่ส่งกลับไปให้ระบบผู้เชี่ยวชาญนั้นเป็นข้อมูลที่ได้มาจากการอนุมานขึ้นมาโดยไม่มีการเก็บข้อมูลจริงๆ ไว้ในฐานข้อมูล นอกจากนี้ยังรวมถึงวิธีการที่ทางส่วนของทางระบบผู้เชี่ยวชาญใช้ในการรับข้อมูลซึ่งผ่านการแปลงแล้วซึ่งในโครงงานนี้จะออกแบบโดยจะเน้นในส่วนวิธีการติดต่อที่โปรแกรมเปลือกระบบผู้เชี่ยวชาญที่ทางโครงงานเลือกใช้คือ Knowledge Pro ใช้ในการติดต่อเพื่อดึงข้อมูลจากภายนอกเป็นหลักเนื่องจากโปรแกรมระบบผู้เชี่ยวชาญต่างๆ จะมีวิธีการและรูปแบบการนำข้อมูลเข้าที่แตกต่างกันไป โดยในการพิจารณาเพื่อออกแบบจะค่อยๆ พิจารณาทีละส่วนๆ เพื่อบอกให้ทราบถึงสาเหตุและนำมาใช้เป็นหลักการที่ใช้ในการออกแบบ โดยก่อนอื่นจะกำหนดข้อจำกัดเบื้องต้นเพื่อใช้เป็นบรรทัดฐานของการออกแบบที่จะมีรายละเอียดต่อไปภายในบทนี้ก่อน คือ

- ในระบบฐานข้อมูลจะไม่มีการเก็บค่า Null ซึ่งจะก่อให้เกิดปัญหาในการตีความหมายของข้อมูลเพื่อใช้ในการทำงานของระบบผู้เชี่ยวชาญ เนื่องจากค่า Null จะมีความหมายที่มิได้เป็นทั้งจริง (True) หรือว่าเท็จ (False) ทั้งสองอย่าง และจะก่อให้เกิดปัญหาในการทำงานของระบบผู้เชี่ยวชาญขึ้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

5.2 ข้อมูลที่ทางระบบผู้เชี่ยวชาญต้องการ

ในหัวข้อนี้จะเป็นการพิจารณาถึงรูปแบบข้อมูลที่เราควรจะส่งไป ให้ทางระบบผู้เชี่ยวชาญเพื่อให้ระบบผู้เชี่ยวชาญทำการประมวลผล โดยจะเริ่มจากการพิจารณาการแทนรูปแบบพรีดิเคตที่ใช้ในฐานข้อมูลอนุมานที่สร้างขึ้นมาจากระบบฐานข้อมูลแบบ Relational ก่อน และชี้ให้เห็นถึงอุปสรรคที่พบ ซึ่งการพิจารณานี้จะอ้างอิงจาก [ธนา 2538] ซึ่งมีรายละเอียดดังนี้

สมมติว่าเรามีฐานข้อมูลซึ่งจะเก็บข้อมูลเกี่ยวกับนักศึกษาในมหาวิทยาลัยแห่งหนึ่ง โดยจะเก็บเลขประจำตัว ชื่อ ที่อยู่ หมายเลขโทรศัพท์ เพศ และชั้นปีที่ศึกษาอยู่และให้ Attribute ทุกตัวมีความสัมพันธ์กันแบบ Fully Functional Dependency กับเลขประจำตัวของนักศึกษา เราจะได้โครงสร้างฐานข้อมูลที่ใช้ในการเก็บข้อมูลแบบ Relational ที่อยู่ในรูปแบบของ Fifth Normal Form สำหรับเก็บข้อมูลเหล่านี้ดังที่แสดงไว้ในรูปที่ 5-1 โดยมี attribute ชื่อ Student_ID เป็น Primary key

← P.K. →

Student_ID	Student_Name	Address	Telephone	Sex	Class

รูปที่ 5-1 ตัวอย่างการใช้ n-ary รีเลชันเพื่อแทนเป็น n-ary พรีดิเคต

ซึ่งในระบบฐานข้อมูลแบบอนุมานที่ใช้รูปแบบการแทนความรู้เป็นแบบ Relational จะแทน n-ary Relation แบบนี้ด้วย n-ary Predicate ที่ชื่อ Student ด้วยโดยที่แต่ละ Attribute ของ Relation จะสัมพันธ์กับแต่ละ Argument ของแต่ละพรีดิเคต

อุปสรรคที่เกิดขึ้นก็คือในเก็บข้อมูลที่ใช้ในการอนุมาน หรือกฎซึ่งจะอยู่ในรูปของ Horn Clause แล้วซึ่งจะแสดงให้เห็นดังตัวอย่าง ซึ่งจะเป็นกฎที่บอกว่านักศึกษาทุกคนที่อยู่ชั้นปีที่ 3 ทุกคนลงทะเบียนเรียนวิชา CS314

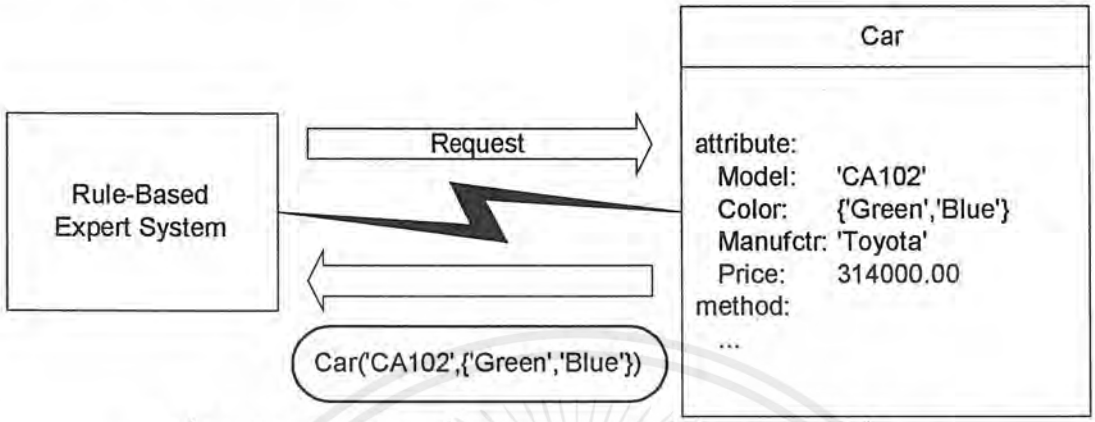
stu_subj(X,'CS314) :- student(X,_,_,_,_,'year3').

ซึ่งจะเห็นได้ว่าการแทนความรู้ในลักษณะนี้ ไม่ใช่ทางเลือกที่ดีนักเพราะในแต่ละ Tuple ของ Relation สามารถที่จะเก็บข้อเท็จจริงได้มากกว่า 1 อย่าง ข้อเท็จจริงที่ไม่เกี่ยวกับกฎจะถูกละไว้ แต่เนื่องจากแต่ละพรีดิเคตถูกกำหนดโดย Relation ดังนั้นผู้ใช้ต้องรู้จำนวนและตำแหน่งของทุก Attribute ในแต่ละ Relation ถึงแม้ว่าจะไม่เกี่ยวกับความจริงที่เป็นเงื่อนไขของกฎก็ตาม

จากตัวอย่างข้างต้นนี้ แสดงให้เห็นว่าเราไม่มีความจำเป็นต้องใช้ Attribute ทุกตัวเพื่อใช้ในการทำงานของระบบผู้เชี่ยวชาญ ในกรณีที่เป็นการทำงานเพื่อดึงข้อมูลจากระบบฐานข้อมูลแบบ Relational แล้วเราก็สามารถใช้ View ซึ่งเป็นลักษณะการดึงข้อมูลจากฐานข้อมูลแบบหนึ่งของฐานข้อมูลแบบ Relational เพื่อดึงเอาเฉพาะ Attribute ที่ต้องการได้ ในกรณีตัวอย่างข้างต้นนี้ เราก็สามารถดึงเอา Attribute เฉพาะที่เราต้องการได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

คือ Student_ID และ Class และนำมาแทนเป็นพรีดิเคตแบบ Binary ได้ และทางโครงงานชิ้นนี้จะนำแนวคิดนี้มาใช้เพื่อเลือกดึงเอา Attribute ที่อยู่ภายใน Object เฉพาะตัวที่ต้องการเท่านั้นมาใช้ในการทำงานของระบบผู้เชี่ยวชาญ โดยจะแสดงให้เห็นจากตัวอย่างในรูปที่ 5-2



รูปที่ 5-2รูปแบบการส่งข้อมูลที่ใช้ลักษณะของ view เพื่อเลือกส่งเฉพาะค่าที่ต้องการ

ซึ่งประโยชน์จากการที่ระบบผู้เชี่ยวชาญเรียกข้อมูลแบบนี้ก็คือข้อมูลบางรูปแบบที่มีลักษณะซับซ้อน ซึ่งไม่สามารถจัดการได้โดยตัวระบบผู้เชี่ยวชาญแต่สามารถเก็บไว้ในระบบฐานข้อมูลเชิงวัตถุ เช่น ข้อมูลที่เป็นภาพ, เสียง, หรือภาพเคลื่อนไหว เราก็ไม่ต้องทำการส่งไปให้ตัวระบบผู้เชี่ยวชาญได้

5.3 การแปลงรูปแบบข้อมูลจากที่มี Repeating Group มาเป็น Tuple แบบที่ฐานข้อมูลแบบ Relational ใช้

ถ้าพิจารณาตัวอย่างการส่งข้อมูลที่แสดงไว้ในรูปที่ 5-2 แล้วจะเห็นได้ว่าข้อมูลที่ส่งกลับมานั้นยังไม่เป็นข้อมูลที่ถูกต้องตามรูปแบบของข้อมูลที่ระบบผู้เชี่ยวชาญแบบกฎ หรือของระบบฐานข้อมูลแบบ Relational เนื่องจากค่า Attribute แต่ละตัวยังไม่ใช่ลักษณะที่เป็น Atomic คือ ข้อมูลที่เก็บไว้ในแต่ละ Attribute ยังสามารถมีข้อมูลที่เป็น Repeating Group ได้ ซึ่งไม่ใช่ลักษณะที่แปลกอะไรสำหรับฐานข้อมูลเชิงวัตถุ ดังนั้นเราจึงยังส่งข้อมูลเหล่านี้ไปยังระบบผู้เชี่ยวชาญโดยตรงไม่ได้ เนื่องจากต้องทำการเปลี่ยนรูปแบบของข้อมูลที่ส่งกลับมาให้เป็น Tuple ที่ในแต่ละ Attribute จะมีข้อมูลที่เป็น Atomic เสียก่อน โดยวิธีที่จะใช้นี้จะใช้ตามตัวอย่างที่ใช้ในหนังสือของ [Date 1995] คือตาราง UCTX ในตารางที่ 5-1

UCTX

COURSE	TEACHERS	TEXTS
Physics	Prof. Green Prof. Brown	Basic Mechanics Principles of Optics
Math	Prof. Green	Basic Mechanics Vector Analysis Trigonometry

ตารางที่ 5-1 ตาราง UCTX

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สามารถแปลงให้เป็นตาราง CTX ซึ่งเป็นตารางที่ Normalized แล้วดังในตารางที่ 5-2

CTX

COURSE	TEACHER	TEXT
Physics	Prof. Green	Basic Mechanics
Physics	Prof. Green	Principles of Optics
Physics	Prof. Brown	Basic Mechanics
Physics	Prof. Brown	Principle of Optics
Math	Prof. Green	Basic Mechanics
Math	Prof. Green	Vector Analysis
Math	Prof. Green	Trigonometry

ตารางที่ 5-2 ตาราง CTX

ซึ่งถึงแม้ว่าการแปลงตารางให้เป็นในรูปแบบนี้จะเป็นวิธีที่ไม่ดีนักเนื่องจากไม่ใช่ตารางที่อยู่ในรูปแบบที่เป็น Forth-Normal Form และมีปัญหาในเรื่องของการเปลี่ยนแปลงข้อมูล แต่เนื่องจากวัตถุประสงค์ของการแปลงนี้ก็เพื่อให้ระบบผู้เชี่ยวชาญสามารถนำข้อมูลไปใช้งานได้เท่านั้น และไม่ได้นำข้อมูลไปทำการแก้ไขหรือเปลี่ยนแปลงอะไรแต่อย่างใด จึงไม่มีผลอะไรกับการทำงานของระบบที่ทางโครงงานออกแบบและสร้างขึ้นมา

5.4. วิธีการที่ระบบผู้เชี่ยวชาญจะทำการเรียกและรับคำตอบ

จากการที่ระบบผู้เชี่ยวชาญที่ทางโครงงานใช้คือ Knowledge Pro นั้น มีรูปแบบการแทนความรู้แบบ Attribute-Value Pair ซึ่งมีลักษณะตามตัวอย่างดังนี้ คือ

Colour is Red

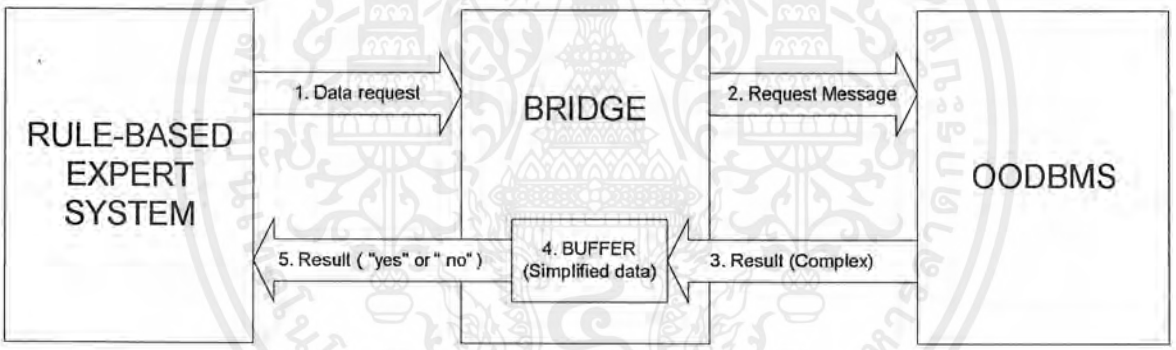
จากตัวอย่างข้างต้น ส่วนที่เป็น Attribute ก็คือ *Colour* และจะมีค่าเป็นค่าที่เราใส่ไว้หลังจาก *is* ก็คือ *Red* ข้อจำกัดการแทนรูปแบบความรู้แบบนี้ก็คือ ทำให้ในการเก็บค่าความจริงของแต่ละ Attribute สามารถเก็บได้เพียงค่าเดียว และทำให้เกิดข้อจำกัดว่า เราจะไม่สามารถใช้ค่าตัวแปรที่เป็นแบบ Symbolic Variable ในส่วนของกฎด้วย การเรียกขอข้อมูลจากฐานข้อมูลจึงต้องเป็นการเรียกขอข้อมูลทีละตัว ซึ่งจากในหัวข้อที่แล้วจะเห็นว่าข้อมูลที่ทางตัวระบบผู้เชี่ยวชาญขอไปนั้น แม้ว่าจะมีการส่งกลับมาเป็นข้อมูล 1 ตัวก็จริง แต่ก็เป็นข้อมูลที่ยังมี Repeating Group อยู่ ซึ่งหากทำการแปลงข้อมูลให้เป็นข้อมูลแบบที่ทำให้ค่า Attribute แต่ละตัวเป็น Atomic แล้วจะทำให้ข้อมูลที่ส่งมาจากฐานข้อมูลกลายเป็นหลาย Tuple ซึ่งจะให้เกิดปัญหาขึ้น ซึ่งสำหรับวิธีการแก้ปัญหาที่ทางโครงงานได้ใช้ก็คือการสร้างพื้นที่สำหรับเก็บข้อมูลเหล่านี้ขึ้นมาโดยจะให้คั่นอยู่ระหว่างตัวระบบผู้เชี่ยวชาญและระบบจัดการฐานข้อมูล ซึ่งตัวกลางตัวนี้จะทำหน้าที่ในการแปลงข้อมูลที่ส่งมาจากระบบฐานข้อมูลด้วย ตัวระบบผู้เชี่ยวชาญจะทำการเรียกข้อมูลจากตัวกลางตัวนี้โดยจะทำการเรียกหาข้อมูลที่ต้องการจากพื้นที่ที่ตัวกลางนั้นได้สร้างขึ้นมาและเก็บข้อมูลที่แปลงแล้วไว้แล้ว ซึ่งทางตัวกลางจะทำการค้นหาว่าพบคำตอบที่ต้องการหรือไม่ และจะส่งคำตอบไปยังระบบผู้เชี่ยวชาญโดยการส่งข้อความว่า "yes" ถ้าหากพบว่าข้อ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

มูลตัวนั้นมีเก็บอยู่ในพื้นที่ที่สร้างขึ้น และหากไม่พบข้อมูลนั้นก็จะทำการส่งคำตอบกลับไปว่า "no" ซึ่งจากทั้งหมดนี้เราก็จะได้รูปแบบที่ใช้ในการติดต่อกันระหว่างระบบผู้เชี่ยวชาญกับฐานข้อมูลเป็นลำดับขั้นตอนดังนี้

1) ทางระบบผู้เชี่ยวชาญต้องการข้อมูลจากฐานข้อมูลและทำการส่งข้อความการร้องขอไปยังตัวกลางที่อยู่ระหว่างระบบผู้เชี่ยวชาญและระบบจัดการฐานข้อมูลซึ่งในส่วนนี้จะมีพื้นที่จะใช้สำหรับเก็บคำตอบที่ส่งมาจากระบบฐานข้อมูลด้วย
2) ตัวกลางจะทำการส่งข้อความที่ระบบผู้เชี่ยวชาญทำการร้องขอต่อไปยังระบบจัดการฐานข้อมูลซึ่งในส่วนนี้อาจมีการเปลี่ยนรูปแบบข้อความที่ใช้ในการร้องขอก็ได้ถ้าจำเป็น
3) ระบบฐานข้อมูลได้รับสัญญาณการร้องขอจากตัวกลางที่ส่งมา และทำการส่งข้อมูลกลับไปยังตัวกลาง ซึ่งข้อมูลที่ส่งกลับมานี้จะมีลักษณะที่เป็นโครงสร้างข้อมูลที่ซับซ้อนอยู่
4) ทางตัวกลางจะทำการแปลงข้อมูลที่ส่งมาจากระบบฐานข้อมูลและเก็บไว้ในพื้นที่ที่จัดเตรียมไว้
5) ตัวกลางทำการสแกนหาข้อมูลที่ทางระบบผู้เชี่ยวชาญต้องการและส่งคำตอบไปยังระบบผู้เชี่ยวชาญโดยจะส่งว่า "yes" ถ้าพบว่ามีข้อมูลในพื้นที่ที่เก็บไว้ และ "no" ถ้าพบว่าไม่มีคำตอบที่ต้องการ

ซึ่งจากลำดับการทำงานข้างต้นนี้เราสามารถนำมาเขียนเป็น Block Diagram ได้ตามรูปที่ 5-2 ดังนี้



รูปที่ 5-3 Block diagram แสดงรูปแบบการติดต่อระหว่างฐานข้อมูลและระบบผู้เชี่ยวชาญ

จากแนวทางเหล่านี้ข้างต้นเราก็พบว่าเราสามารถสร้างวิธีการที่ใช้ในการแก้ปัญหาที่เกิดขึ้นจากความไม่เข้ากันของรูปแบบข้อมูลระหว่างแบบ Object และแบบ Tuple ได้แล้ว และสำหรับในบทต่อไป เราก็จะนำเอาแนวทางการออกแบบนี้ไปใช้ในการสร้างระบบที่เป็นโปรแกรมต้นแบบ เพื่อใช้ในการทดสอบการทำงานเพื่อพิสูจน์แนวความคิดได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 6
การพัฒนาระบบต้นแบบเพื่อใช้ทดสอบ

6.1 บทนำ

สำหรับในเนื้อหาของบทที่ผ่านๆ มาซึ่งได้กล่าวถึงแนวทางที่ใช้ในการออกแบบเรียบร้อยแล้ว สำหรับในบทนี้จะเป็นการนำรูปแบบสถาปัตยกรรมและโพรโตคอลที่ใช้ในการติดต่อกันระหว่างระบบผู้เชี่ยวชาญและระบบจัดการฐานข้อมูลที่ได้ออกแบบไว้แล้วมาใช้ในการสร้างระบบต้นแบบ เพื่อที่จะทำการทดสอบการทำงานและพิสูจน์ความถูกต้องของรูปแบบสถาปัตยกรรมและโพรโตคอลของการติดต่อระหว่างฐานข้อมูลและระบบผู้เชี่ยวชาญ โดยในบทนี้จะเป็นเนื้อหาจะเริ่มที่ส่วนของการพิจารณาวิธีที่จะใส่กฎเข้าไปในส่วนของระบบจัดการฐานข้อมูลก่อน จากนั้นก็จะพิจารณาถึงวิธีการที่จะทำให้รูปแบบโพรโตคอลที่เราได้ออกแบบไว้สามารถทำงานได้โดยจะพิจารณาในทุกๆ ส่วนทั้งในด้านของระบบผู้เชี่ยวชาญและด้านระบบจัดการฐานข้อมูล รวมทั้งแนวทางในการสร้างโปรแกรมที่เป็นตัวกลางซึ่งใช้สำหรับการเปลี่ยนรูปแบบข้อมูลและสำหรับเก็บข้อมูลที่ได้ทำการเปลี่ยนแล้วรวมทั้งการสแกนข้อมูลเพื่อทำการหาคำตอบเพื่อส่งกลับไปยังระบบผู้เชี่ยวชาญ

6.2 วิธีการที่ใช้ในการสร้างกฎสำหรับในส่วนของฐานข้อมูล

สำหรับในการสร้างกฎที่ส่วนของระบบจัดการฐานข้อมูลนั้น จะแบ่งกฎเหล่านั้นออกได้เป็น 2 ประเภทใหญ่ๆ คือตามที่ได้มีการอธิบายมาแล้ว คือ Active Rule และ Deductive Rule ซึ่งจากการพิจารณาคุณสมบัติต่างๆ ที่มีอยู่ในระบบจัดการฐานข้อมูลเชิงวัตถุแล้ว ทางผู้ทำโครงงานพบว่าในโครงสร้างแบบ Object นั้น จะมีส่วนที่เป็น Method ซึ่งใช้สำหรับเก็บกระบวนการที่ใช้ในการทำงานเกี่ยวกับข้อมูลที่อยู่ในฐานข้อมูล ซึ่งเราสามารถนำมาประยุกต์ด้วยการนำกฎต่างๆ เหล่านั้นมาเขียนเป็น Method เหล่านี้เพื่อให้กฎต่างๆเหล่านั้นสามารถทำงานได้ ทั้งกฎที่เป็นแบบ Active Rule และกฎที่เป็นแบบ Deducitve Rule โดยรูปแบบที่ใช้ก็จะสามารถทำได้หลายรูปแบบด้วยกันตามแต่ลักษณะและวัตถุประสงค์ที่จะนำไปใช้งาน ซึ่งจะพิจารณาตามความเหมาะสม สำหรับรูปแบบตัวอย่างที่ได้นำเสนอนี้ก็จะเป็นการแบ่งออกตามรูปแบบการใช้งานในกรณีทั่วไป โดยจะพิจารณาถึงประเด็นหลักๆ ที่เกี่ยวข้องเป็นส่วนใหญ่ แต่สำหรับในการพัฒนาระบบเพื่อใช้งานจริงบางระบบก็อาจมีกรณีที่อยู่นอกเหนือไปจากที่แสดงไว้ในรายงานฉบับนี้ก็ได้ ซึ่งผู้พัฒนาระบบอาจสร้างเพิ่มเติมขึ้นได้หากเห็นว่ามีความจำเป็น สำหรับในส่วนหลักๆ ที่มักจะมีการใช้งานอยู่เสมอก็มีดังนี้

6.2.1 กฎประเภท Active Rule

จะแบ่งออกเป็นสามส่วนใหญ่ๆ ก็คือ กรณีที่เป็นการแก้ไขข้อมูล กรณีที่เป็นการเพิ่มข้อมูลและในกรณีที่เป็นการลบข้อมูล โดยในแต่ละกรณีเหล่านี้ ก็อาจมีการแบ่งเป็นกรณีย่อยๆ ลงไปอีกตามแต่ลักษณะการใช้งาน

- ถ้าเป็นรูปแบบที่ใช้สำหรับเปลี่ยนแปลงข้อมูลที่อยู่ใน Object จะใช้ Method ที่เป็นระดับ Object-Level คือเป็น Method ที่ใช้จัดการ Object แต่ละตัว และจะมีค่า Argument ของ Method เป็นค่าใหม่ที่ต้องการเปลี่ยนแปลง ซึ่งรูปแบบการเรียกใช้ในโครงงานที่ได้ทำการออกแบบเมื่อเขียนให้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

อยู่ในรูปแบบที่ใช้ในระบบจัดการฐานข้อมูล Jasmine ซึ่งเป็นระบบจัดการฐานข้อมูลเชิงวัตถุที่โครงงานใช้จะมีลักษณะคือ

Result = Name.MethodName(Argument);

เมื่อ *Name* คือชื่อตัวแปรที่เป็น Object ที่เราต้องการเปลี่ยนแปลงค่า, *MethodName* คือชื่อ Method ที่ใช้สำหรับในการเปลี่ยนแปลงค่าข้อมูลที่ต้องการ, *Argument* คือค่าที่เราต้องการจะทำการเปลี่ยนใหม่ และ *Result* คือตัวแปรที่ใช้สำหรับเก็บค่าผลลัพธ์ของการเปลี่ยนแปลง ถ้ามีโปรแกรมใดๆ ต้องการเปลี่ยนแปลงค่าข้อมูลเหล่านี้ ก็จะทำการเรียกผ่าน Method เหล่านี้ และตัว Method จะเก็บกระบวนการการตรวจเงื่อนไขข้อบังคับต่างๆ ที่ต้องมีเมื่อมีการแก้ไขข้อมูล ซึ่งถ้าพบว่าไม่มีเงื่อนไขขัดแย้งใดๆ ก็จะเปลี่ยนแปลงค่าตามที่ส่งเข้ามาและส่งค่ากลับเป็นค่าปกติ แต่ถ้าเกิดเงื่อนไขขัดแย้งก็จะไม่ทำการแก้ไขข้อมูลและทำการส่งค่าที่บอกถึงความผิดพลาด

- ถ้าเป็นรูปแบบที่ใช้สำหรับสร้าง Object จะใช้ Method ระดับ Class-Level ซึ่งเป็น Method ที่จะครอบคลุมถึง Object ทุกตัวที่อยู่ใน Class นั้นๆ โดยรูปแบบของ Method ที่ใช้สำหรับสร้าง Object ตัวใหม่ซึ่งอยู่ใน Class นั้นจะมีลักษณะ

Result = Name.MethodName(Arg1,Arg2,Arg3,...);

เมื่อ *Name* คือชื่อ Class ที่เราต้องการสร้าง Object ใหม่, *MethodName* คือชื่อ Method ที่ใช้สำหรับในการสร้างค่าข้อมูลที่ต้องการ, *Arg1,Arg2,...* คือค่า Attribute ต่างๆ ที่เราต้องการให้มีใน Object ที่ต้องการสร้างใหม่, *Result* คือตัวแปรที่ใช้สำหรับเก็บค่าผลลัพธ์ของการเปลี่ยนแปลง ถ้ามีโปรแกรมใดๆ ต้องการสร้าง Object ก็จะทำการเรียกผ่าน Method เหล่านี้ เช่นเดียวกับ Method ที่ใช้ในการแก้ไขข้อมูลคือ ถ้าพบว่าไม่มีเงื่อนไขขัดแย้งใดๆ ก็จะทำการสร้าง Object ตัวนั้นเก็บไว้ในฐานข้อมูลและส่งค่ากลับเป็นค่าปกติ แต่ถ้าเกิดเงื่อนไขขัดแย้งก็จะไม่ทำการสร้างและทำการส่งค่าที่บอกถึงความผิดพลาด

- ถ้าเป็นรูปแบบที่ใช้สำหรับลบ Object จะใช้ Method ระดับ Class-Level เช่นเดียวกับ Method ที่ใช้ในการสร้าง Object โดยรูปแบบเป็นดังนี้

Result = Name.MethodName(Argument);

เมื่อ *Name* คือชื่อ Class ที่เราต้องการสร้าง Object ใหม่, *MethodName* คือชื่อ Method ที่ใช้สำหรับในการสร้างค่าข้อมูลที่ต้องการ, *Argument* คือค่าตัวแปรที่เก็บค่า Object ID ที่เราต้องการทำลบในฐานข้อมูล, *Result* คือตัวแปรที่ใช้สำหรับเก็บค่าผลลัพธ์ของการเปลี่ยนแปลง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ถ้ามีโปรแกรมใดๆ ต้องการลบ Object ก็จะทำการเรียกผ่าน Method เหล่านี้ เช่นเดียวกับ Method ที่ใช้ในการสร้าง Object คือ ถ้าพบว่าไม่มีเงื่อนไขขัดแย้งใดๆ ก็จะทำการลบ Object ตัวนั้นในฐานข้อมูลและส่งค่ากลับเป็นค่าปกติ แต่ถ้าเกิดเงื่อนไขขัดแย้งก็จะไม่ลบและทำการส่งค่าที่บอกถึงความผิดพลาด

6.2.2 กฎประเภท Deductive rule

สำหรับกฎประเภทนี้ก็จะทำการสร้างในรูปแบบของ Method เช่นเดียวกัน ซึ่งหากพิจารณาแล้วกฎประเภทนี้จะใช้ในการอนุมานหาค่า Attribute ที่เกี่ยวข้องกับ Object เฉพาะตัวนั้นๆ ดังนั้นในการสร้าง Method เพื่อใช้ในการอนุมานหาค่าที่ต้องการนั้น ทางโครงงานจะสร้างเป็น Method ระดับ Object-Level โดยจะมีลักษณะเหมือนกันทั้งหมด ดังนี้

Attribute = Name.MethodName();

เมื่อ *Name* คือ ตัวแปรที่เก็บ Object ID ของ Object ที่เราสนใจ, *MethodName* คือชื่อ Method ที่จะใช้แทนค่า Attribute ที่เราต้องการทราบซึ่งจะเป็น Method แบบที่ไม่มีค่า Argument, *Attribute* คือตัวแปรที่ใช้สำหรับเก็บค่า Method ที่เราจะใช้แทนค่า Attribute ที่เราได้ค่ามาจากการอนุมาน (ซึ่งจะไม่มีการเก็บค่าไว้จริงในฐานข้อมูลตัวนั้น) ทำการส่งกลับมา ซึ่งจะต้องมีรูปแบบชนิดข้อมูลแบบเดียวกับค่าที่ Method ตัวนั้นส่งกลับ และอาจเป็นค่าแบบ Repeating Group ก็ได้ ถ้ามีโปรแกรมใดๆ ต้องการทราบค่า Attribute เหล่านี้ ก็จะทำการเรียกผ่าน Method เหล่านี้ และตัว Method ก็จะทำการประมวลผลขั้นตอนการอนุมานข้อมูลและทำการส่งค่ากลับไป

6.3 การสร้างส่วนที่ใช้ส่งผ่านข้อมูลระหว่างระบบผู้เชี่ยวชาญ

สำหรับในส่วนนี้จะแยกพิจารณาสร้างส่วนต่างๆ โดยจะใช้หลักการพิจารณาเป็นส่วนๆ เช่นเดียวกับที่ใช้ในการพิจารณาเพื่อทำการออกแบบดังรายละเอียดในบทที่แล้ว โดยจะแบ่งเป็นดังนี้

6.3.1 ส่วนที่ใช้ในการเลือกเอาข้อมูลเฉพาะตัวที่ระบบผู้เชี่ยวชาญต้องการ

จากการวิเคราะห์เพื่อออกแบบในบทที่แล้วจะพบว่า เราไม่จำเป็นต้องมีการส่งผ่านข้อมูลที่เก็บไว้ใน Object ทั้งหมดไปให้ยังระบบผู้เชี่ยวชาญ แต่สามารถทำการส่งข้อมูลเพียงเฉพาะที่ระบบผู้เชี่ยวชาญต้องการเท่านั้น ซึ่งจากการดูคุณสมบัติของเรื่อง Object-Oriented แล้ว เราจะพบว่าเราสามารถนำ Method ที่มีอยู่ใน Object มาใช้ประโยชน์ในการส่งข้อมูลแบบนี้ได้ โดยการใช้ Method ทำกระบวนการที่ใช้ดึงข้อมูลเฉพาะ Attribute ที่เราต้องการและเก็บไว้ในตัวแปรตัวหนึ่งแล้วส่งกลับมายังระบบผู้เชี่ยวชาญได้ แต่สำหรับวิธีนี้ก็จะมีข้อจำกัดเช่นกันคือ ทางระบบผู้เชี่ยวชาญจะสามารถเรียกรูปแบบข้อมูลที่ต้องมีการสร้างเป็น Method จัดเตรียมไว้อยู่แล้วในฐานข้อมูล ซึ่งถ้าหากทางระบบผู้เชี่ยวชาญต้องการรูปแบบข้อมูลที่นอกเหนือไปจากนี้ ก็จะต้องทำการสร้างเป็น method และจัดเก็บไว้ในฐานข้อมูลเสียก่อน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รูปแบบ Method ที่จะสร้างขึ้นมาเพื่อทำหน้าที่นี้ หากพิจารณาดูแล้วควรจะสร้างให้เป็น Method ในระดับ Class-Level เนื่องจากในการอ้างอิงถึงข้อมูลนั้นจะทำการอ้างอิงถึง Object ทั้งหมดที่มีอยู่ใน Class โดยรูปแบบของ Method มีลักษณะดังนี้

Result = Name.MethodName(arg1,arg2,arg3,...);

เมื่อ *Name* คือชื่อ Class ที่เราต้องการอ้างอิงถึงข้อมูลที่ทางระบบผู้เชี่ยวชาญต้องการ, *MethodName* คือชื่อ Method ที่ใช้สำหรับในการสร้างค่าข้อมูลที่ต้องการ, *Arg1,Arg2,...* คือค่า Attribute ต่างๆ ที่ทางระบบผู้เชี่ยวชาญต้องการทราบว่ามีอยู่ในฐานข้อมูลหรือไม่, *Result* คือตัวแปรที่ใช้สำหรับเก็บค่าผลลัพธ์ที่ได้มาจากทางฐานข้อมูลซึ่งจะยังไม่มีการเปลี่ยนแปลงรูปแบบให้เป็นแบบ Tuple ซึ่งทางตัวกลางที่ทำหน้าที่เชื่อมต่อระหว่างฐานข้อมูลและระบบผู้เชี่ยวชาญ จะทำการเรียก Method เหล่านี้ โดยจะเรียกตามคำสั่งร้องขอข้อมูลที่ทางระบบผู้เชี่ยวชาญส่งเข้ามา และทำการแปลงคำสั่งเหล่านั้นมาเป็นการเรียก Method เหล่านี้ที่เกี่ยวข้อง และทำการเก็บคำตอบที่ได้รับไว้ในพื้นที่หน่วยความจำที่ได้ทำการจัดเตรียมไว้เพื่อทำการแปลงข้อมูลที่ได้รับต่อมาในภายหลัง

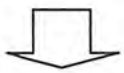
6.3.2 อัลกอริธึมที่ใช้ในการแปลงข้อมูลที่ได้รับมาจากทางระบบฐานข้อมูลมาเป็นข้อมูลแบบ tuple

จากรูปแบบหลักการที่เราได้ออกแบบและจะใช้เพื่อทำการแปลงข้อมูลจากโครงงสร้างที่มี Repeating Group เป็นข้อมูลที่ได้ทำการ Normalized ในบทที่แล้วนั้น ในส่วนนี้เราจะมาพิจารณาถึงรายละเอียดของวิธีการหรืออัลกอริธึมที่จะใช้ในการแปลงข้อมูล โดยก่อนอื่นเราจะมาพิจารณาถึงรายละเอียดของรูปแบบข้อมูลที่ส่งกลับมาว่ามีความเป็นไปได้ที่จะเป็นข้อมูลในรูปแบบใดบ้างก่อน

จากรูปแบบคำถามที่ทางระบบผู้เชี่ยวชาญจะทำการร้องขอไปยังฐานข้อมูลนั้นจะเป็นรูปแบบการถามที่เป็น Predicate แบบไม่มี Repeating Group แต่ระบบฐานข้อมูลอาจจะส่งข้อมูลแบบที่เป็นโครงสร้างแบบที่ซับซ้อนแต่จะมีข้อมูลที่ทางระบบผู้เชี่ยวชาญขอไปอยู่ภายใน หรืออาจส่งข้อมูลเปล่ากลับมาถ้าในฐานข้อมูลไม่มีข้อมูลที่ทางระบบผู้เชี่ยวชาญขอไป ในกรณีที่มีการส่งข้อมูลกลับมาซึ่งจะเกิดขึ้นก็ต่อเมื่อมีคำตอบพบว่าค่า Attribute แต่ละค่าของ Tuple ที่ทำการถามไปตรงกับ Attribute หรือเป็นหนึ่งในเซตของค่า Attribute ของ Object ตัวเดียวกันที่เก็บอยู่ในฐานข้อมูลครบทุกตัว ซึ่งอาจมีมากกว่าหนึ่ง Object ก็ได้ในบางกรณี เนื่องจากทฤษฎีของ Object-Oriented จะพบว่า Object แต่ละตัวอาจมีค่า Attribute ทุกอย่างตรงกันก็ได้ ซึ่งในกรณีนี้ก็จะได้คำตอบที่จะทำการส่งกลับมามากกว่าหนึ่งตัว แต่เนื่องจากสิ่งที่เราต้องการ คือ ต้องการเพียงหาว่ามีคำตอบอยู่ในฐานข้อมูลหรือไม่เท่านั้น ดังนั้นการส่งคำตอบกลับมายังตัวกลางตัวนี้ก็สามารถส่งคำตอบมาแค่ตัวเดียวก็ได้แล้วค่อยมาทำการแปลงข้อมูลเพื่อเก็บไว้ในพื้นที่หน่วยความจำ ซึ่งวิธีการที่ใช้ในการแปลงมีรายละเอียดที่สามารถอธิบายได้โดยดูจากรูปที่ 6-1 ในหน้าต่อไป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

‘A’ ‘B’	‘C’ ‘D’	‘E’ ‘F’



‘A’	‘C’ ‘D’	‘E’ ‘F’
‘B’	‘C’ ‘D’	‘E’ ‘F’

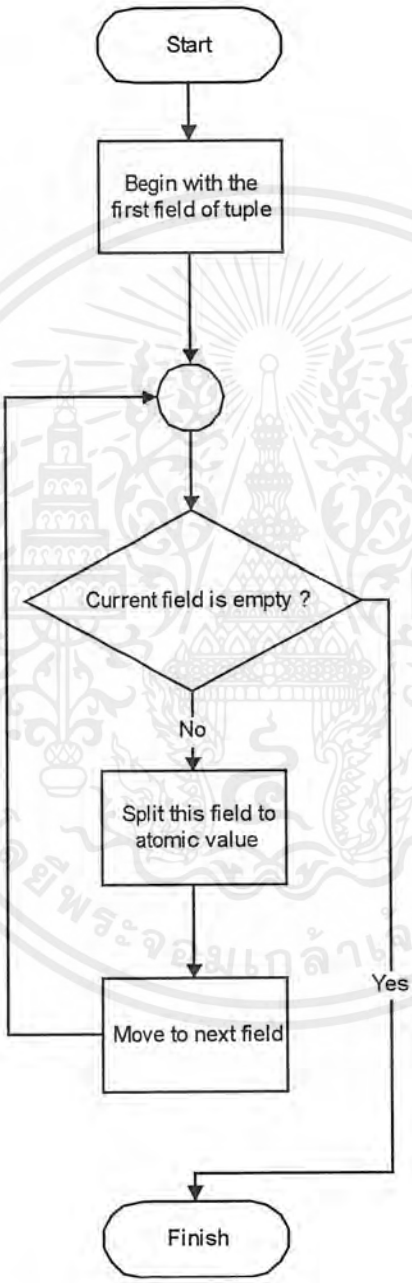
‘A’	‘C’	‘E’ ‘F’
‘A’	‘D’	‘E’ ‘F’
‘B’	‘C’	‘E’ ‘F’
‘B’	‘D’	‘E’ ‘F’

‘A’	‘C’	‘E’
‘A’	‘C’	F’
‘A’	‘D’	‘E’
‘A’	‘D’	‘F’
‘B’	‘C’	‘E’
‘B’	‘C’	‘F’
‘B’	‘D’	‘E’
‘B’	‘D’	‘F’

รูปที่ 6-1 ลำดับการแปลงข้อมูล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากภาพที่เป็นการแสดงวิธีการแปลงไปแล้วนั้น จะพบว่าวิธีการแปลงที่จะใช้ในการแปลงนี้จะเป็นการแปลงโดยจะค่อยเปลี่ยนค่าที่อยู่ใน Field ทางซ้ายให้เป็นชนิดข้อมูลแบบที่เป็น Atomic ก่อนแล้วค่อยๆ ไล่ไปยัง Field ที่อยู่ทางขวาไปเรื่อยๆ จนครบทุก Field ซึ่งจากแนวทางนี้ เราก็จะนำมาเขียนเป็นอัลกอริธึมซึ่งจะแสดงเป็น Flow Chart ได้ดังรูปที่ 6-2



รูปที่ 6-2 Flow chart แสดงการทำงานเพื่อแปลงข้อมูลโดยสังเขป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

6.3.3 วิธีการที่ระบบผู้เชี่ยวชาญใช้ในการเรียกข้อมูลจากระบบจัดการฐานข้อมูล

จากการที่โปรแกรมเปลือกระบบผู้เชี่ยวชาญที่ทางโครงงานได้ใช้คือ Knowledge Pro นั้น มีวิธีการที่ใช้ในการติดต่อกับข้อมูลภายนอกได้เพียงวิธีเดียวเท่านั้น คือ ทาง DDE (Dynamic Data Exchange) และจะไม่มีวิธีการที่ใช้ในการติดต่อกับโปรแกรมระบบจัดการฐานข้อมูลใดๆ โดยตรงรวมทั้งระบบจัดการฐานข้อมูลเชิงวัตถุที่ทางโครงงานได้ใช้ด้วย คือ Jasmine ทำให้ทางโครงงานต้องทำการเขียนโปรแกรมที่ทำหน้าที่เป็น DDE Server เพื่อจัดการด้านการบริการข้อมูลให้แก่ทางระบบผู้เชี่ยวชาญ โดยการพัฒนาโปรแกรมตัวนี้ จะเลือกใช้ภาษา Visual C++ เนื่องจากเล็งเห็นว่า ภาษานี้สามารถจัดการทางด้านปัญหาที่เกิดขึ้นเนื่องจากข้อจำกัดทางด้านรูปแบบการติดต่อกับภายนอกของโปรแกรม Knowledge Pro ได้ อีกทั้งทางระบบจัดการฐานข้อมูล Jasmine ที่ทางโครงงานได้ใช้นั้น ก็ออกแบบโดยยึดติดกับการใช้ภาษา C++ ค่อนข้างมากเนื่องจากในการสร้าง Method เพื่อใช้งานในระบบจัดการฐานข้อมูลนั้น จำเป็นต้องมี compiler ภาษา C++ ด้วยเพื่อให้ method ที่เราสร้างขึ้นมาและจัดเก็บในฐานข้อมูลนั้นสามารถนำไปใช้งานได้ และยังจัดเตรียมชุดคำสั่งที่เป็น API (Application Programmer's Interface)เพื่อใช้ในการติดต่อกับฐานข้อมูล ซึ่งทางผู้จัดทำโครงงานก็ได้ใช้โปรแกรมตัวนี้ในการติดต่อกับทางฐานข้อมูลและจัดการทางด้านเปลี่ยนแปลงรูปแบบข้อมมูลที่ส่งกลับมาด้วย โดยรายละเอียดของวิธีการเชื่อมต่อกับระบบฐานข้อมูล Jasmine โดยใช้ภาษา C++ และการเชื่อมต่อโดยการใช้ DDE ทั้งในทางส่วนของภาษา C++ และ ทางโปรแกรม Knowledge Pro จะมีรายละเอียดแสดงอยู่ในภาคผนวก ข. และ ค. ตามลำดับ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 7
การทดสอบระบบที่ได้ทำการสร้าง

7.1. บทนำ

หลังจากที่เราทำการวิเคราะห์ออกแบบโครงสร้างสถาปัตยกรรม, ออกแบบและสร้างระบบต้นแบบขึ้นมาเรียบร้อยแล้ว ขั้นตอนสุดท้ายก็คือทำการทดสอบระบบที่สร้างขึ้นมาเพื่อพิสูจน์ว่าระบบสามารถทำงานได้ถูกต้องและเป็นไปตามวัตถุประสงค์ที่ต้องการ โดยในส่วนนี้จะอธิบายในส่วนของการสร้างระบบผู้เชี่ยวชาญเพื่อทำการทดลองใช้ประโยชน์จากสถาปัตยกรรมที่คิดค้นขึ้นมานี้และทำการวิเคราะห์เปรียบเทียบ สำหรับรายละเอียดในบทนี้จะอธิบายถึงโครงสร้างของข้อมูลที่เก็บไว้ในฐานข้อมูลเพื่อใช้ในการทดสอบ และหลังจากนั้นจะเป็นการออกแบบสร้างระบบผู้เชี่ยวชาญที่จะนำข้อมูลที่อยู่ในฐานข้อมูลมาใช้ประโยชน์ในการประมวลผล โดยจะเน้นในส่วนของการประยุกต์ใช้กฎประเภท Deducitve Rule ที่เราได้สร้างและเก็บไว้เป็น Method ที่อยู่ในฐานข้อมูลเป็นหลัก และจะไม่เน้นในด้านส่วนของกฎที่เป็น Active Rule เนื่องจากขอบเขตข้อกำหนดที่ได้ใช้ในการออกแบบของโครงงานนี้จะเกี่ยวข้องกับการแก้ไขหรือเปลี่ยนแปลงข้อมูลใดๆ ที่อยู่ในฐานข้อมูลและเนื่องจาก Active Rule บางชนิดก็จะมีอยู่ในกลไกการทำงานของระบบฐานข้อมูลอยู่แล้ว เช่น กฎที่ใช้ในการทำ Rollback และกฎที่ใช้ควบคุมความถูกต้องของข้อมูลต่างๆ เหล่านี้ก็ไม่ได้เป็นสิ่งใหม่อะไรเนื่องจากระบบจัดการฐานข้อมูลในปัจจุบันก็มีการจัดการในเรื่องนี้ได้อยู่แล้ว

7.2. ระบบผู้เชี่ยวชาญที่ทำการออกแบบและสร้างขึ้นเพื่อใช้ในการทดสอบ

ในส่วนของระบบผู้เชี่ยวชาญที่จะทำการสร้างขึ้นมาจะไม่เน้นในเรื่องของกระบวนการถอดความรู้เท่าไรนัก เนื่องจากโครงงานนี้เป็นโครงงานวิจัย ที่ไม่เน้นในส่วนของการนำเอาระบบไปประยุกต์ใช้งานได้จริง ที่ต้องมีการตรวจสอบความถูกต้องของคำตอบที่ได้ ดังนั้นรูปแบบการให้คำตอบและเหตุผลที่ใช้ในการพิจารณาอาจจะผิดเพี้ยนไปจากความเป็นจริง ซึ่งระบบที่ใช้ในการทดสอบนี้จะเป็นระบบผู้เชี่ยวชาญที่ใช้ช่วยในการตัดสินใจเลือกอาจารย์ที่ปรึกษาวิชาโครงงานในชั้นปีที่ 4 ของภาควิชาวิศวกรรมคอมพิวเตอร์ โดยจะมีการใช้ประโยชน์จากฐานข้อมูลที่เราสร้างขึ้นมาในส่วนของข้อมูลของนักศึกษา โดยการทำงานจะมีขั้นตอนโดยสังเขปดังนี้

- ให้ผู้ใช้ใส่รหัสนักศึกษาของตัวเอง
- ระบบผู้เชี่ยวชาญจะทำการตรวจสอบว่ารหัสนักศึกษานั้นเป็นนักศึกษาปี 4 ของภาควิชาวิศวกรรมคอมพิวเตอร์หรือไม่ โดยจะเรียกข้อมูลจากระบบฐานข้อมูลถ้าพบว่าใช่คือมีในฐานข้อมูลก็จะให้ทำงานต่อ แต่ถ้าไม่ก็จะหยุดการทำงานและแสดงข้อความว่ารหัสผู้ใช้ไม่ถูกต้อง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- จากนั้นก็จะให้ผู้ใช้ทำการเลือกหัวข้อโครงงานที่สนใจ 2 หัวข้อ แล้วจะพิจารณาตามความเหมาะสมโดยจะดูจากความถนัดของนักศึกษาที่เก็บเอาไว้ในฐานข้อมูล
- เมื่อได้หัวข้อที่ต้องการแล้วก็จะทำการเลือกอาจารย์ที่ปรึกษาซึ่งจะความชำนาญในด้านนั้นๆ โดยจะมีเกณฑ์ที่ใช้ในการพิจารณาจากความถนัดของนักศึกษาโดยดูจากผลการเรียนของวิชาที่เกี่ยวข้องเป็นหลัก ซึ่งเมื่อได้วิชาที่เหมาะสมกับนักศึกษาคนนั้นแล้วก็จะนำมาพิจารณาหาอาจารย์ที่ปรึกษาที่เหมาะสม ในกรณีที่มีอาจารย์ที่มีความชำนาญมากกว่า 1 คน ก็อาจจะมีการพิจารณาข้อมูลตัวอื่นๆ อีกตามความจำเป็น
- จากนั้นโปรแกรมจะแสดงคำตอบให้แก่ผู้ใช้ว่าอาจารย์ที่ทางระบบแนะนำคืออาจารย์ท่านใด

7.3. โครงสร้างข้อมูลที่จะเก็บไว้ในฐานข้อมูล

สำหรับในการทดสอบนี้จะใช้ข้อมูลตามโครงสร้างที่ได้ใช้เป็นตัวอย่างประกอบการอธิบายในบทที่ 4 ซึ่งจากโครงสร้างที่ได้ออกแบบไปแล้วนี้เราจะจัดเก็บเป็นรูปแบบข้อมูลแบบ Object ในฐานข้อมูล โดยจะเน้นในส่วนของ Attribute หรือเรียกในระบบฐานข้อมูลที่โครงงานใช้ว่าเป็น Property และ Method ที่สร้างขึ้นมาเพื่อใช้ในการเก็บส่วนของกฎที่ใช้ในการอนุมานและจะไว้ที่ใช้ในส่วนของฐานข้อมูลได้ดังนี้

class Person
instance-level property
Name : *String;*
Surname : *String;*
Gender : *String;*

class Person::Student
instance-level property
ID : *String;*
Registered : *collections of RegsSubj;*
instance-level method
Year(); ชั้นปีที่นักศึกษาคนนั้นอยู่
Dept(); ภาควิชาที่นักศึกษาคนนั้นอยู่
Status(); สถานะการศึกษาของนักศึกษาคนนั้น
Specialist(); ความถนัดในสาขาวิชาของนักศึกษาคนนั้น
Gpa(); ผลการศึกษาเฉลี่ยของนักศึกษาคนนั้น

class Person::Lecturer
instance-level property
HighestDegree : *String;*
Teach : *collections of Subject;*
instance-level method
Dept(); ภาควิชาที่อาจารย์ท่านนั้นสังกัด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

class Subject
instance-level property

ID	*: String;*
Name	*: String;*
Type	*: String;*
Credit	*: Integer;*
OpenDept	*: String;*

class RegsSubj
instance-level property

YearReg	*: Integer;*
Semester	*: Integer;*
Subj	*: Subject;*
Grade	*: Real;*

7.4. กฎที่จะเก็บไว้ที่ส่วนของระบบผู้เชี่ยวชาญ

จากในหัวข้อที่แล้ว ก็ได้มีการอธิบายถึงกฎที่จะเก็บไว้ที่ส่วนของฐานข้อมูลแล้ว สำหรับในหัวข้อนี้ จะเป็นการอธิบายถึงกฎที่จะเก็บไว้ในส่วนของระบบผู้เชี่ยวชาญ ซึ่งจะประกอบด้วยกฎประเภทต่างๆ ดังนี้

- กฎที่ใช้ในการหาว่าเป็นนักศึกษาที่เข้าข่ายต้องใช้ระบบผู้เชี่ยวชาญหรือไม่ คือ กฎที่แสดงอยู่ในตัวอย่างข้างล่าง

  if นักศึกษาคนนั้นเป็นนักศึกษาชั้นปีที่ 4 ภาควิชาวิศวกรรมคอมพิวเตอร์
  then อนุญาตให้ผู้ใช้ใช้ระบบได้

- กฎที่ใช้ในการตัดสินใจถึงเนื้อหาของหัวข้อโครงงานที่เหมาะสมกับนักศึกษาคนนั้น เนื่องจากในระบบผู้เชี่ยวชาญที่ทำการสร้างขึ้นมานี้ จะให้ผู้ใช้ได้ทำการเลือกหัวข้อที่สนใจมาก่อน 2 เรื่อง จากนั้นจึงทำการตรวจสอบว่าผู้ใช้คนนั้นมีความเหมาะสมที่จะทำโครงงานในหัวข้อใดโดยจะดูจากวิชาที่ผู้ใช้คนนั้นมีความถนัด สำหรับในตัวอย่างข้างล่างนี้จะเป็นส่วนหนึ่งของกฎในกลุ่มนี้

  if นักศึกษาคนนั้นถนัดวิชา Microprocessor Interfacing
  then เนื้อหาวิชาโครงงานที่เหมาะสมกับนักศึกษาคือเรื่อง Hardware

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

if นักศึกษาคนนั้นถนัดวิชา Computer Network
then เนื้อหาวิชาโครงงานที่เหมาะสมกับนักศึกษาคือเรื่อง Network

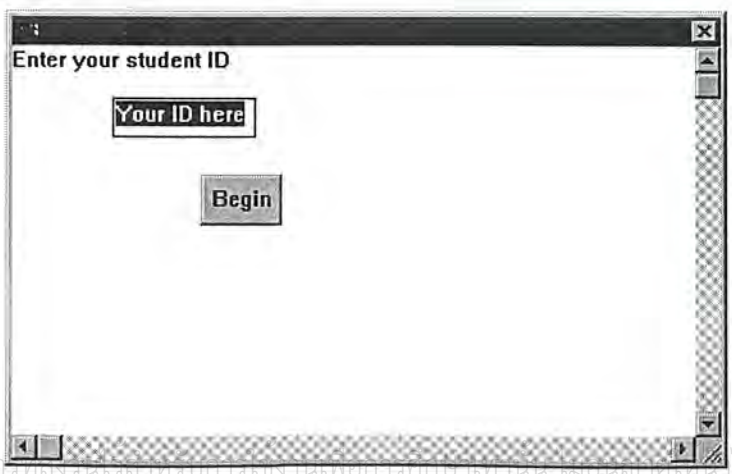
if นักศึกษาคนนั้นถนัดวิชา Database systems
then เนื้อหาวิชาโครงงานที่เหมาะสมกับนักศึกษาคือเรื่อง Database

- กฎที่ใช้ในการตัดสินใจว่าหัวข้อที่ทางระบบผู้เชี่ยวชาญจะใช้เป็นหัวข้อหลักที่จะใช้พิจารณาเพื่อเลือกหาอาจารย์ที่ปรึกษาที่มีความถนัดในด้านนั้นๆ โดยจะมีรายละเอียดคือ หากหัวข้อที่ผู้ใช้มีความสนใจเป็นอันดับแรกเป็นหัวข้อที่ทางระบบพบว่าเป็นหัวข้อที่ผู้ใช้คนนั้นมีความถนัดด้วยก็จะใช้หัวข้อนั้นเป็นหลักในการหาอาจารย์ที่ปรึกษา แต่ถ้าไม่ ก็จะพิจารณาถึงหัวข้อที่ผู้ใช้สนใจรองลงมาว่าเป็นหัวข้อที่ผู้ใช้คนนั้นถนัดด้วยหรือไม่ แต่หากพบว่าผู้ใช้ไม่ได้มีความถนัดในหัวข้อใดหัวข้อหนึ่งเลย ทางระบบก็จะตัดสินให้เอาหัวข้อหลักที่ผู้ใช้คนนั้นสนใจเป็นอันดับแรกในการเลือกหาอาจารย์ที่ปรึกษา

- กฎที่ใช้ในการเลือกหาอาจารย์ที่ปรึกษา โดยจะมีการพิจารณาถึงรายละเอียดปลีกย่อยต่างๆ ที่นอกเหนือไปจากหัวข้อของโครงงานที่ผู้ใช้มีความสนใจ โดยในส่วนนี้จะเป็นการพิจารณาถึงข้อมูลอื่นๆ ที่เกี่ยวข้องกับผู้ใช้อีก ซึ่งอาจมีผลในการตัดสินเลือกอาจารย์ที่ปรึกษาซึ่งอาจมีความถนัดในหัวข้อเดียวกัน แต่มีความต้องการนักศึกษาที่มีลักษณะส่วนตัวที่แตกต่างกัน ซึ่งข้อมูลที่ใช้ในการพิจารณาเพิ่มเติมในส่วนนี้ก็ได้แก่ ระดับผลการเรียนเฉลี่ย บุคลิกลักษณะการทำงานส่วนตัว

7.5. ตัวอย่างการทำงานของระบบที่ทำการพัฒนา

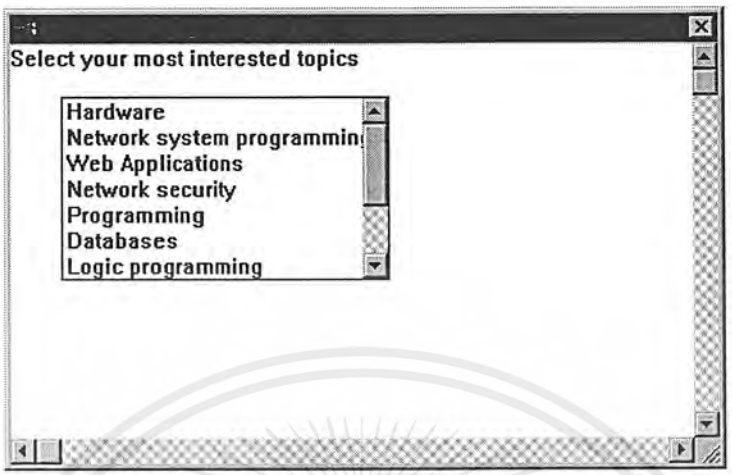
ในส่วนนี้จะเป็นการทดลองการใช้งานระบบที่ทำการสร้างขึ้นมาเพื่อทำการทดสอบการทำงานของระบบโดยในตอนแรกในส่วนของระบบผู้เชี่ยวชาญจะปรากฏกล่องข้อความเพื่อให้ป้อนคำสั่ง ดังรูปที่ 7-1



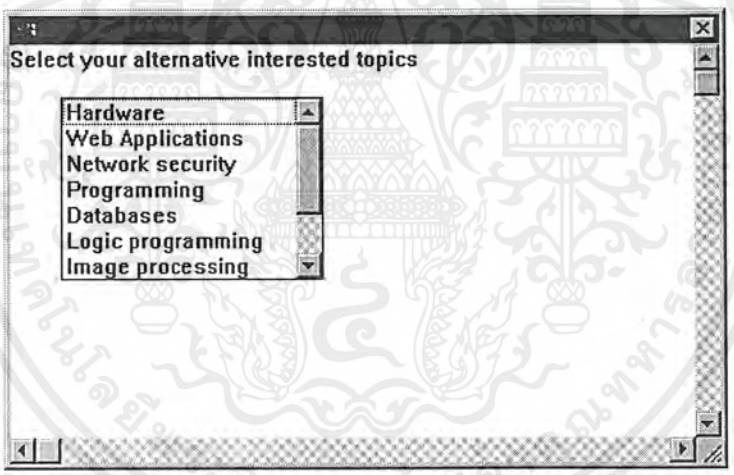
รูปที่ 7-1 การทำงานของโปรแกรมที่ให้ผู้ใช้ป้อนข้อความ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งไม่อนุญาตให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเอกสารทุกครั้งที่มีการนำไปใช้

เมื่อทำการป้อนรหัสเข้าไปแล้วในส่วนนี้ฐานข้อมูลจะส่งข้อความร้องขอไปยังฐานข้อมูลว่ารหัสนี้เป็นผู้ใช้ที่ถูกต้องหรือไม่ ซึ่งถ้าพบว่าถูกต้องแล้วก็จะปรากฏข้อความให้ทำการเลือกเนื้อหาที่สนใจ 2 เรื่อง ดังรูปที่ 7-2 และ 7-3



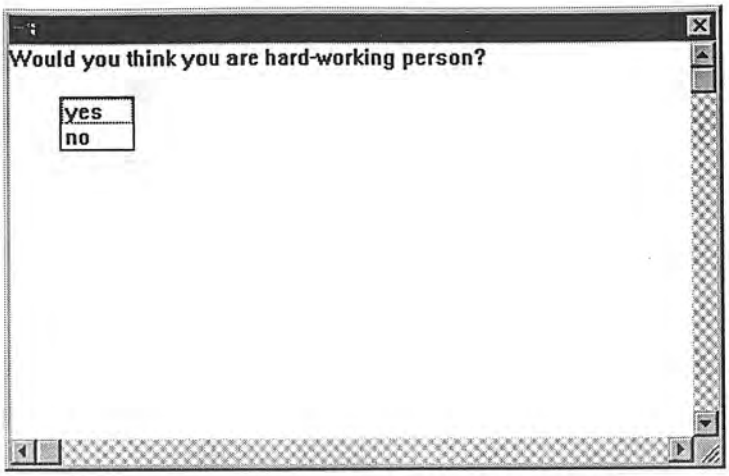
รูปที่ 7-2 หน้าจอที่แสดงการให้เลือกหัวข้อที่ต้องการ



รูปที่ 7-3 หน้าจอที่แสดงการให้เลือกหัวข้อที่สนใจอันดับที่สอง

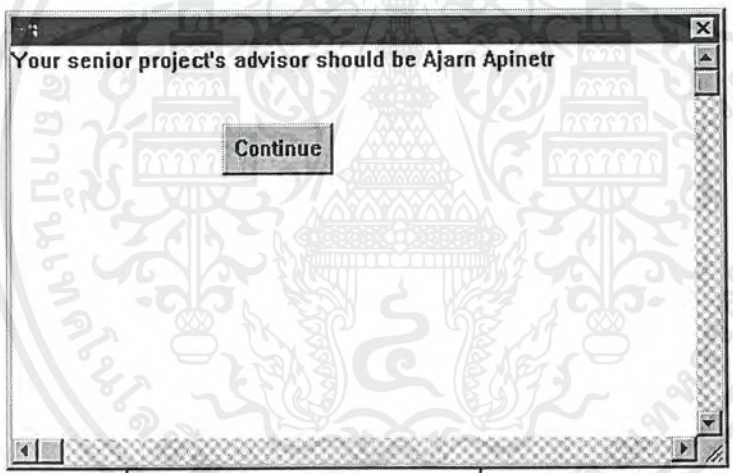
เมื่อทำการเลือกหัวข้อที่สนใจแล้วโปรแกรมระบบผู้เชี่ยวชาญก็จะทำการเรียกขอข้อมูลวิชาที่นักศึกษาคนนั้นชำนาญเพื่อนำมาเลือกเอาหัวข้อที่เหมาะสมหลังจากนั้นจึงทำการเลือกหาอาจารย์ที่ปรึกษาซึ่งมีความชำนาญในด้านนั้นโดยอาจมีการถามคำถามจากผู้ใช้เพิ่มเติมอีกเล็กน้อย ตามรูปที่ 7-4 ในหน้าต่อไป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 7-4 หน้าจอแสดงการถามคำถามอื่นๆ เพื่อเลือกคำตอบที่เหมาะสม

หลังจากที่ได้ข้อมูลที่ระบบผู้เชี่ยวชาญต้องการครบแล้วก็จะแสดงคำตอบเหมาะมาให้แก่ผู้ใช้ดัง รูปที่ 7-5



รูปที่ 7-5 หน้าจอแสดงการแสดงคำตอบที่เหมาะให้แก่ผู้ใช้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 8

สรุปผล,วิจารณ์ และแนวทางการพัฒนาต่อ

8.1 สรุปผลงานวิจัย

จากวัตถุประสงค์ซึ่งเป็นที่มาของโครงงานวิจัยชิ้นนี้ ก็คือเพื่อหาวิธีการประยุกต์ใช้ข้อมูลที่เก็บอยู่ระบบจัดการฐานข้อมูลเพื่อนำมาใช้กับระบบผู้เชี่ยวชาญสำหรับในการให้ข้อมูลเพื่อช่วยให้การทำงานของระบบผู้เชี่ยวชาญมีประสิทธิภาพมากขึ้น คือ ลดการถามคำถามที่ไม่ควรจะถามผู้ใช้ เนื่องจากจะทำให้มีการถามคำถามเกิดขึ้นมากเกินไป และเกิดความซ้ำซาก เนื่องจากทุกครั้งที่เริ่มการทำงานของระบบผู้เชี่ยวชาญใหม่ก็จะต้องมีการถามคำถามนั้นอีก ซึ่งคำถามประเภทนี้ ก็ คือ คำถามประเภทที่เป็นข้อมูลในอดีตซึ่งจะมีประโยชน์อยู่แล้วหากได้มีการเก็บเอาไว้ในฐานข้อมูล ซึ่งตามปกติตัวโปรแกรมเปลือกระบบผู้เชี่ยวชาญนั้นจะมีความสามารถในการติดต่อกับแหล่งข้อมูลภายนอกได้อย่างจำกัดอยู่แล้ว ซึ่งทางโครงงานก็ได้คิดค้นหาวิธีการที่ใช้ในการเชื่อมต่อกับระบบจัดการฐานข้อมูล เพื่อให้เราสามารถทำการบริหารข้อมูลได้ง่ายขึ้น และยังสามารถนำข้อมูลเหล่านั้นที่เก็บอยู่ในระบบฐานข้อมูลไปใช้ประโยชน์โดยโปรแกรมอื่นๆ ที่มีการเรียกใช้ฐานข้อมูลเช่นเดียวกัน

นอกจากนี้ การที่ระบบได้มีการถามคำถามจากผู้ใช้น้อยลงนั้น ก็ยังเป็นประโยชน์ คือ ช่วยให้ระบบผู้เชี่ยวชาญสามารถทำงานได้อย่างถูกต้องมากขึ้น เนื่องจากสาเหตุในกรณีดังต่อไปนี้

- การถามคำถามกับผู้ใช้มากๆ จะเป็นการเพิ่มโอกาสที่จะเกิดความผิดพลาดอันเนื่องมาจากผู้ใช้ตอบคำถามกับตัวโปรแกรมผิดๆ เนื่องจากเกิดความเบื่อหน่ายที่จะตอบคำถามกับโปรแกรม
- การถามคำถามบางอย่าง ซึ่งเป็นข้อมูลในอดีต บางครั้งอาจพบว่า ผู้ใช้เองอาจจำไม่ได้ เนื่องจากเวลาผ่านมานานมากเกินไป ทำให้มีการตอบคำถามที่ผิดๆ ได้
- คำถามบางคำถาม ซึ่งอาจจะเกี่ยวข้องกับตัวผู้ใช้ แต่ตัวผู้ใช้อาจไม่รู้ เนื่องจากอาจเป็นความรู้ที่เป็นความรู้เฉพาะทาง ซึ่งก็จะทำให้ผู้ใช้ตอบคำถามที่ผิดๆ ก็ได้

และนอกจากนี้ในขอบเขตของโครงงานที่ได้มีการนำเสนอการใส่กฎไว้ในฐานข้อมูลเพื่อไว้ใช้ในการอนุมานข้อมูลซึ่งไม่มีการเก็บไว้ในฐานข้อมูลจริงๆ นั้น ทางโครงงานก็ได้ทดลองทำการหาความเป็นไปได้โดยการเก็บไว้ในรูปของ Method ที่จะเรียกใช้เมื่อต้องการข้อมูลตัวนั้น ซึ่งก็พบว่าสามารถทำงานได้ดีโดยไม่พบข้อผิดพลาดอะไร หากมีการกำหนดเงื่อนไขที่ใช้ในการอนุมานที่รัดกุมพอ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

8.2 วิจารณ์งานวิจัย

จากงานวิจัยที่ได้ทำการค้นคว้าและคิดค้นขึ้นมานั้น เมื่อเปรียบเทียบกับระบบอื่นๆ ที่มีอยู่ก่อนแล้วสิ่งที่ผลงานของงานวิจัยชิ้นนี้ที่แตกต่างและดีกว่าสถาปัตยกรรมของระบบผู้เชี่ยวชาญอื่นๆ ก็คือ

- ความสามารถในการติดต่อกันระหว่างระบบผู้เชี่ยวชาญแบบ Rule-Based และระบบจัดการฐานข้อมูลเชิงวัตถุซึ่งมีลักษณะโครงสร้างข้อมูลที่แตกต่างกันก็คือระบบผู้เชี่ยวชาญที่เป็นแบบกฎ นั้นสามารถทำงานกับข้อมูลที่เป็นแบบพรีดิเคต ซึ่งจริงๆ แล้ว ระบบฐานข้อมูลแบบ Relational ซึ่งมีการเก็บข้อมูลแบบเป็น Tuple เท่านั้นน่าจะเหมาะสมกว่า แต่ในโครงงานนี้ก็ได้แสดงให้เห็นว่าระบบจัดการฐานข้อมูลเชิงวัตถุนั้นจะเก็บข้อมูลที่เป็นแบบที่ซับซ้อนว่า คือ ในแต่ละ Attribute อาจมีค่าเป็น Repeating Group หรืออาจเป็นโครงสร้าง Object ตัวอื่นอีก ก็สามารถใช้ในการติดต่อกับระบบผู้เชี่ยวชาญแบบนี้ได้เช่นกัน
- ความสามารถในการจัดเก็บกฎที่ใช้ในการอนุมานข้อมูลตัวใหม่ๆ ไว้ในฐานข้อมูลและถูกเรียกใช้โดยโปรแกรมอื่นๆ ที่ติดต่อกับฐานข้อมูลได้ ทำให้เกิดประโยชน์ก็คือ สามารถประหยัดเนื้อที่ที่ใช้ในการเก็บข้อมูลได้ ซึ่งการเรียกใช้ข้อมูลแบบนี้ก็ไม่ได้จำกัดเฉพาะกับโปรแกรมระบบผู้เชี่ยวชาญเพียงอย่างเดียวเท่านั้น แต่ยังรวมไปถึงโปรแกรมทั่วไปอื่นๆ ด้วย
- ลดความยุ่งยากในการจัดการกับข้อมูลที่รับมาจากฐานข้อมูลซึ่งบางครั้งทางระบบจัดการฐานข้อมูลจะมีการส่งข้อมูลที่ไม่ได้เป็นค่าเดียวแต่จะเป็นเซ็ตของชุดข้อมูล โดยจะทำให้เป็นเสมือนว่าระบบผู้เชี่ยวชาญกำลังทำงานกับข้อมูลตัวเดียว และจะให้โปรแกรมที่เป็นตัวกลางสำหรับทำหน้าที่ในการติดต่อกับฐานข้อมูลเป็นตัวจัดการเอาเอง ทำให้สามารถใช้งานกับระบบผู้เชี่ยวได้หลากหลายรูปแบบขึ้น

และสำหรับส่วนที่ระบบสถาปัตยกรรมของการติดต่อระหว่างระบบผู้เชี่ยวชาญและฐานข้อมูลที่ได้ออกแบบในโครงงานนี้ไม่ได้มีความสามารถครอบคลุมถึงในส่วนนั้น แต่พบว่ามีระบบผู้เชี่ยวชาญอื่นๆ ที่มีความสามารถครอบคลุมถึง คือ

- ความสามารถในการเปลี่ยนแปลงข้อมูลที่อยู่ในฐานข้อมูลโดยระบบผู้เชี่ยวชาญ
- ความสามารถในการติดต่อกับระบบฐานข้อมูลโดยที่ผู้พัฒนาโปรแกรมไม่จำเป็นต้องทราบในขณะที่ทำการพัฒนาระบบผู้เชี่ยวชาญ โดยจะถือว่าข้อมูลต่างๆ เหล่านั้นทางระบบผู้เชี่ยวจะทำการจัดการค้นหาเอาเองว่าจะดึงมาจากฐานข้อมูล หรือจะต้องถามจากผู้ใช้
- ความสามารถในการเรียกดูผลของการทำงานของระบบผู้เชี่ยวชาญที่เคยทำงานไว้ในอดีตซึ่งความสามารถนี้ต้องอาศัยความสามารถในการบันทึกผลการทำงานของระบบผู้เชี่ยวชาญลงในฐานข้อมูลด้วย
- ความสามารถในการจัดการดึงข้อมูลที่เก็บอยู่ฐานข้อมูลซึ่งมีรูปแบบข้อมูลที่ซับซ้อน เช่น ข้อมูลที่เป็นแฟ้มข้อมูลแบบ Multimedia เป็นต้น
- ความสามารถในส่วนของฐานข้อมูลในการจัดการกับกฎที่เป็นแบบ Recursive ได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

8.3 แนวทางการพัฒนาต่อ

สำหรับแนวทางในการพัฒนาต่อนั้นอาจแยกได้เป็น 2 แนวทางหลักๆ ก็คือ

- พัฒนาในเน้นในด้านประสิทธิภาพในการทำงานของระบบ โดยจะพัฒนาในส่วนที่ใช้ในการส่งผ่านข้อมูลให้มีความสามารถในการตรวจดูว่าการขอข้อมูลที่ทางส่วนของระบบผู้เชี่ยวชาญนั้นเป็นการขอข้อมูลเดิมหรือข้อมูลที่เกี่ยวข้องกันซึ่งมีเก็บอยู่ในพื้นที่หน่วยความจำชั่วคราวหรือไม่เนื่องมาจากการขอข้อมูลครั้งก่อนๆ หรือไม่ ถ้าหากมีก็ไม่ต้องทำการส่งการขอข้อมูลจากฐานข้อมูล หรือทำการวิเคราะห์ว่าหากมีการขอข้อมูลแบบใดแบบหนึ่งแล้ว รูปแบบข้อมูลแบบใดที่จะทางระบบผู้เชี่ยวน่าจะทำการขอเป็นตัวต่อไปก็จะทำการส่งข้อมูลตัวนั้นไปด้วยเลยเพื่อเป็นการลด Overhead ที่เกิดจากการติดต่อกับฐานข้อมูล ซึ่งในส่วนนี้จะเป็นการพัฒนาส่วนของโปรแกรมที่ใช้ในการติดต่อกับระบบฐานข้อมูล
- พัฒนาในส่วนของระบบที่ใช้ในการพัฒนาสร้างระบบผู้เชี่ยวชาญโดยไม่ต้องให้ผู้พัฒนาระบบผู้เชี่ยวต้องทำการจัดการส่วนที่ใช้ในการเชื่อมต่อกับฐานข้อมูลแต่จะให้ดูเหมือนว่าข้อมูลทุกอย่างจะอยู่ที่ตัวระบบผู้เชี่ยวชาญอยู่แล้ว โดยวิธีการพัฒนาอาจใช้วิธีสร้างตัว Preprocessor ขึ้นมาเพื่อทำการประมวลผลโปรแกรมที่ผู้ใช้เขียนขึ้นมาและเปลี่ยนเป็นตัวโปรแกรมที่ใช้สำหรับทำงานจริงๆ ซึ่งในส่วนนี้จะเน้นในการพัฒนาส่วนของระบบที่ใช้ในการพัฒนาระบบผู้เชี่ยวชาญเพื่อให้สามารถใช้ประโยชน์จากระบบสถาปัตยกรรมที่ทางโครงงานได้จัดทำขึ้น
- พัฒนาในส่วนที่ทำให้ระบบผู้เชี่ยวชาญมีความสามารถในด้านการเปลี่ยนแปลงแก้ไขข้อมูลที่อยู่ในฐานข้อมูลได้ ซึ่งจะทำให้ระบบนี้มีรูปแบบการใช้ประโยชน์ที่กว้างขึ้นไปอีก เช่น อาจใช้เป็น Intelligent Agent ที่ใช้ในการแก้ไขข้อมูลอะไรบางอย่างในฐานข้อมูล โดยสามารถปล่อยให้โปรแกรมทำงานเองแบบอัตโนมัติ โดยไม่ต้องให้คนมาทำหน้าที่นี้เอง
- พัฒนาในส่วนที่ให้ระบบผู้เชี่ยวชาญ สามารถรับหรือจัดการกับข้อมูลจากฐานข้อมูลซึ่งมีรูปแบบข้อมูลที่ซับซ้อนได้ เช่น ข้อมูลที่เป็นแฟ้มข้อมูลแบบ Multimedia เป็นต้น
- พัฒนาให้ส่วนของฐานข้อมูลมีความสามารถในการจัดการกับกฎที่เป็นแบบ Recursive ได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาคผนวก ก.
ระบบจัดการฐานข้อมูลเชิงวัตถุ : Jasmine

ก.1. ภาพโดยรวมของระบบจัดการฐานข้อมูล Jasmine

ในส่วนของบทที่ 2 จะมีการอธิบายถึงหลักการพื้นฐานและส่วนประกอบเบื้องต้นต่างๆ ที่จะพบอยู่ในระบบฐานข้อมูลเชิงวัตถุไปแล้ว สำหรับในส่วนของภาคผนวกนี้จะเป็นการอธิบายถึงตัวผลิตภัณฑ์ระบบจัดการฐานข้อมูลเชิงวัตถุที่ทางโครงงงานได้ใช้ก็คือ Jasmine version 1.20 โดยในส่วนนี้จะอธิบายถึงหลักการต่างๆ เบื้องก่อนในมุมมองจากตัวผลิตภัณฑ์ จากนั้นจะค่อยๆ อธิบายถึงวิธีการติดตั้ง และการใช้งานต่อไป

ในส่วนแรกที่จะอธิบายถึงก็คือลักษณะของ Object ในมุมมองของ Jasmine จะมีองค์ประกอบพื้นฐาน 2 ส่วนหลักคือ

Properties – เป็นสิ่งที่บอกถึงค่าหรือข้อมูลบน Object นั้นๆ เช่น พนักงานจะมี ชื่อ, หมายเลขประกันสังคม และเงินเดือน เป็น Property

Methods – เป็นการแสดง behavior ของ Object เช่น พนักงานสามารถเข้าทำโครงงานใหม่ๆ หรือเปลี่ยนงาน

ซึ่งในแต่ละส่วนสามารถอธิบายในส่วนของรายละเอียดได้ดังนี้

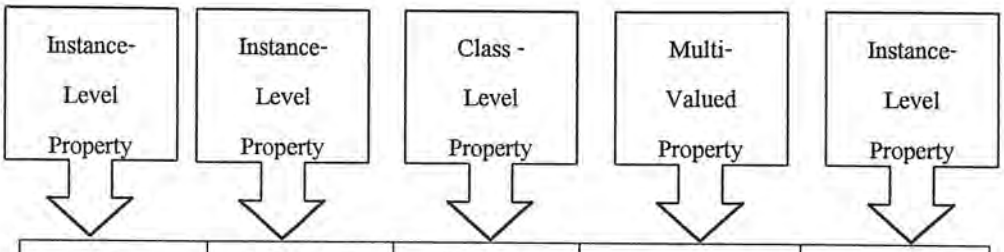
ก.1.1. Properties

การที่ Object มี Properties จะช่วยให้เรารู้ว่า Object นั้นมีลักษณะอย่างไรและช่วยแบ่ง Object ต่างๆ ออกเป็นกลุ่มได้ Properties ของ Object หนึ่งก็ถูกกำหนดเป็นส่วนหนึ่งของข้อมูลภายใน Class และข้อมูลของ Properties ต่างๆ ภายใน Class ก็ถูกเรียกว่า Metadata

ใน Jasmine ได้แบ่ง Properties ออกเป็น 5 ส่วนคือ

1. Instance-level properties
2. Class-level properties
3. Multi-valued properties
4. Attribute properties
5. Relationship properties

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



Name	Extension	Payment Schedule	Languages Spoken	Manager
John Brown	222	Semi-monthly	English, French	Frank Nelson
Fred Smith	201		English, Spanish	Mary Johnson
Jane Jones	208		English, French, Italian	Bob Williams

รูปที่ ก-1 แสดงตัวอย่าง Property แต่ละชนิด

ก.1.1.1. Instance-Level Properties

เป็นคุณสมบัติของแต่ละ Object เช่น จากรูปที่ ก-1 ข้างบน Instance-Level Properties คือ *name* และ *extension*

ก.1.1.2. Class-Level Properties

เป็นคุณสมบัติของ Class รวมทั้งเป็นคุณสมบัติของ Object ทั้งหมดใน Class นั้นด้วย Class-Level Properties ใช้ในกรณีที่ Instance ทั้งหลายใน Class ต้องการใช้ค่านี้ร่วมกัน

ก.1.1.3. Multi-Values Properties

ทั้ง Instance และ Class-Level Properties มีคุณสมบัติเป็น Multi-Valued ได้ คือสามารถมีค่าแสดง Property เดียวได้มากกว่า 1 ค่า

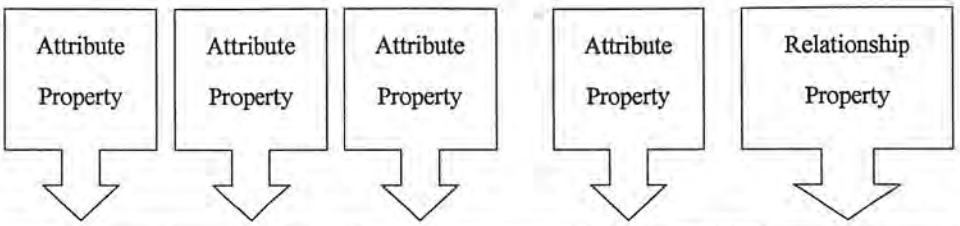
ก.1.1.4. Attribute and Relationship Properties

Properties สามารถแบ่งออกเป็นได้อีกเป็น Attribute และ Relationship โดย Attribute คือ Property ที่มีเป็นรูปแบบข้อมูลที่มีลักษณะพื้นฐาน ไม่ซับซ้อน เช่น number, string หรือ date ส่วน Relationship คือ Property ที่เป็นรูปแบบข้อมูลที่มีลักษณะเป็น Object ซับซ้อน และการอ้างอิงถึงจะใช้ตัว OID เพื่อแสดงความสัมพันธ์กับ Object อื่น แสดงดังรูปที่ ก-2

ก.1.1.5. Many-to-Many Relationships

คล้ายกับ Relationship Properties แต่มีข้อดีเหนือ relational database เนื่องจากแบบของ relational database ต้องการคีย์เมื่อต้องการเชื่อมตารางกับตารางอื่น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



Name	Extension	Payment Schedule	Languages Spoken	Manager
John Brown	222	Semi-monthly	English, French	Frank Nelson
Fred Smith	201		English, Spanish	Mary Johnson
Jane Jones	208		English, French, Italian	Bob Williams

รูปที่ ก-2 แสดง Property แบบ Attribute และ Relationship

ก.1.2. Method

ใน Jasmine มีการแบ่ง Method ออกเป็น 3 แบบ คือ

1. Instance-level methods
2. Class-level methods
3. Collection-level methods

ก.1.2.1. Instance-Level Methods

เป็น Methods ที่อยู่ในระดับ Object เช่น ใน Method ที่จะบอกถึงปีที่แต่ละ employee ทำงานให้กับองค์กร

ก.1.2.2. Class-Level Methods

เป็น Methods ที่อยู่ในระดับ Class เช่น Method ที่ทำหน้าที่ในการสร้าง Object ใหม่หรือ Method ที่สามารถสร้าง instance ให้กับ Class โดยใช้การเก็บข้อมูลใน external file

ก.1.2.3. Collection-Level Methods

คำว่า Collection คือ การรวม Object ที่มีลักษณะเดียวกันเข้าไว้ด้วยกัน ทั้ง Instance- และ Class-Level Methods ก็สามารถเป็น Collection ได้ ตัวอย่างของ Collection-Level Methods เช่น ใน Method ที่ทำหน้าที่ในการเพิ่มเงินเดือนให้กับกลุ่มของ employee

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

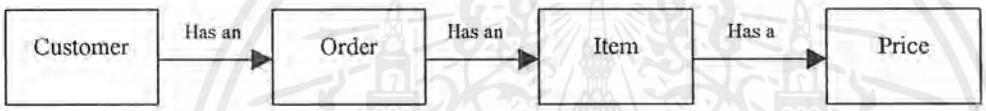
ก.2. Inheritance

คือความสามารถในการอนุญาตให้สร้าง Class ใหม่ (ที่เรียกว่า Class ลูก) ที่มี Properties และ Methods ทั้งหมดที่อยู่ใน Class ต้นอยู่แล้ว โดยสามารถเพิ่ม Properties และ Methods ใหม่ๆ หรือทำการเปลี่ยนแปลงแก้ไขได้

ใน Jasmine ได้สนับสนุน Multiple Inheritance นั่นคือการที่ Subclass สามารถ Inherit Properties และ Methods จาก Superclass อื่นได้ ดังนั้นบางครั้งจึงทำให้สับสนเนื่องมาจากการที่ Superclass สามารถมี Method หรือ Property ชื่อซ้ำกันได้ Jasmine ทำการป้องกันความสับสนนี้โดยการยืนยันว่า Method หรือ Property ใดเป็นของใคร

ก.3. Aggregation

ใช้อธิบายความสัมพันธ์ระหว่าง Object โดยเกิดจากการที่ Object หนึ่งได้อ้างอิงกับ Object อื่น ใน Inheritance ถูกอธิบายโดยใช้วลี "is-a" ส่วนใน Aggregation ใช้วลี "has-a" ในการอธิบาย ดังแสดงในรูปที่ ก-3



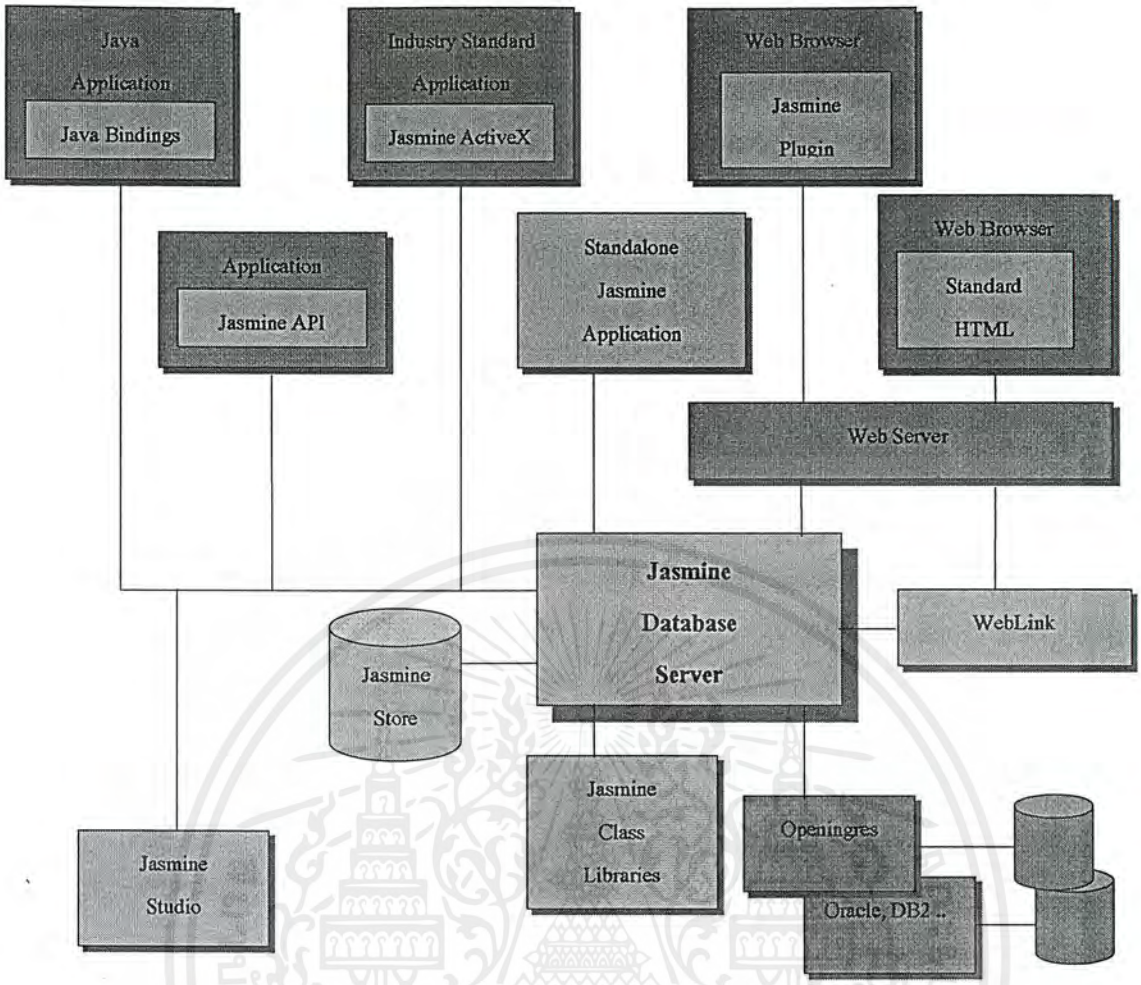
รูปที่ ก-3 แสดงความสัมพันธ์แบบ Aggregation

ก.4. การสร้างระบบที่ใช้เก็บข้อมูลด้วย Jasmine

สำหรับรูปแบบสิ่งแวดล้อมในระบบจัดการฐานข้อมูล Jasmine ที่ผู้พัฒนาสามารถทำการสร้างระบบสารสนเทศหรือที่ต้องเกี่ยวข้องด้วยนั้นจะประกอบด้วยส่วนประกอบต่างๆ ดังที่จะแสดงในรูปที่ ก-4 ในหน้าต่อไป และรายละเอียดของแต่ละส่วนก็มีดังนี้

- *Jasmine Store* -- แหล่งเก็บข้อมูล(Physical data)ของ Jasmine
- *Jasmine Database Server* -- เป็น Server ที่ทำหน้าที่เก็บข้อมูลประเภท Multimedia เพื่อใช้กับแอพพลิเคชัน และยังเป็นแหล่งจัดการฐานข้อมูล รวมทั้งการกระทำของ Method จะเกิดขึ้นที่นี่
- *Jasmine Client* -- มีการเชื่อมต่อกับ Jasmine database server และมีแอพพลิเคชันของ Jasmine รันอยู่
- *Jasmine Class Libraries* -- เป็นกลุ่มของ Class families ที่นำมาใช้กับ Jasmine
- *Jasmine Studio* -- เป็น GUI-based authoring tool เพื่อใช้สร้างแอพพลิเคชันของ Jasmine และใช้เข้าถึง jasmine database
- *Jasmine API* -- เป็น function calls ที่ใช้จัดการกับ Jasmine database โดยผ่าน C/C++ หรือภาษาอื่นๆ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



- *Java Bindings* -- การอินเตอร์เฟสระหว่าง Jasmine กับ Java และสามารถสร้าง Java class ให้เป็น Jasmine class ได้ และอนุญาตให้มีการเข้าถึงฐานข้อมูลผ่าน Java
- *Jasmine ActiveX* -- เพื่อให้ง่ายต่อการทำงานร่วมกับระบบปฏิบัติการจึงสามารถเข้าถึงข้อมูลผ่าน Visual Basic หรือเครื่องมือพัฒนาที่สนับสนุน ActiveX
- *Jasmine Plug-in* -- เป็น Web browser plug-in ที่อนุญาตให้ผู้ใช้สามารถรันแอพพลิเคชันของ Jasmine และเข้าถึงฐานข้อมูลของ Jasmine ผ่าน World Wide Web
- *WebLink* -- เป็น HTML renderer ที่อนุญาตให้เข้าถึง Jasmine objects จาก World Wide Web โดยไม่ต้องติดตั้ง Jasmine plug-in
- *Relational data Access* -- เป็นเซ็ตของ Jasmine classes ที่ทำให้แอพพลิเคชันของ Jasmine สามารถเข้าถึงและจัดการข้อมูลบนระบบฐานข้อมูลอื่นได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ก.5. การติดตั้งและใช้งานโปรแกรม

ก.5.1 ความต้องการของระบบ

สำหรับในการติดตั้งโปรแกรมระบบจัดการฐานข้อมูล Jasmine จะติดตั้งได้ในระบบปฏิบัติสองระบบ คือ Windows NT และ Solaris สำหรับระบบปฏิบัติการที่โครงงานนี้เลือกใช้ก็คือ ระบบปฏิบัติการ Windows 95/NT ซึ่งจะต้องใช้ระบบคอมพิวเตอร์ขั้นต่ำดังนี้

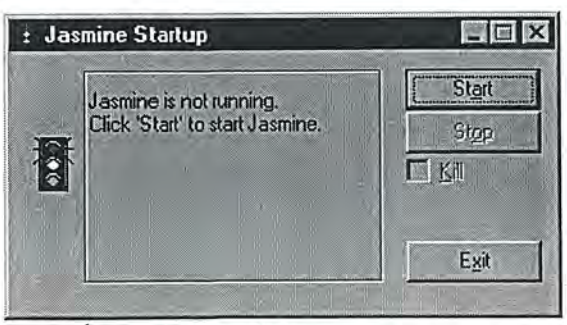
- เครื่องคอมพิวเตอร์ PC ที่ใช้ CPU รุ่น Pentium
- หน่วยความจำของเครื่อง 32 MB เป็นอย่างต่ำ (แนะนำให้ใช้ขนาด 64 MB)
- พื้นที่ว่างบนฮาร์ดดิสก์ 200 MB เป็นอย่างต่ำ (แนะนำว่าควรมี 300 MB)
- ระบบปฏิบัติการ Microsoft Windows NT version 4.0 ภาษาอังกฤษ สำหรับการลงระบบเพื่อให้เครื่องทำหน้าที่เป็น Database server และเป็น Windows 95 หรือ NT ก็ได้ในกรณีที่เป็นการติดตั้งเพื่อให้เครื่องทำหน้าที่เป็นตัว Client
- โปรแกรม Microsoft Visual C++ version 4.2 ขึ้นไป สำหรับใช้คอมไพล์ Method ที่จะเก็บไว้ในฐานข้อมูล
- โพรโตคอล TCP/IP สำหรับการเชื่อมต่อเพื่อใช้งานแบบระยะไกล

สำหรับในการติดตั้งโปแกรมเพื่อใช้งานนั้น จะต้องทำการติดตั้งจากแผ่น CD-ROM ซึ่งใช้ในการติดตั้งโปรแกรม โดยโปรแกรมจะทำงานเองโดยอัตโนมัติเมื่อใส่แผ่น CD-ROM นี้ จากนั้นก็ให้ปฏิบัติตามขั้นตอนต่างๆ ซึ่งจะทำงานในรูปแบบของโปรแกรม Installation Wizard ไปเรื่อยๆ จนเสร็จสมบูรณ์ก็เป็นเสร็จสิ้นขั้นตอนการติดตั้งโปรแกรม

ก.5.2. การเริ่มต้นและหยุดการทำงานของการให้บริการฐานข้อมูล

สำหรับในส่วนของการติดตั้ง เราจะสามารเลือกได้ว่าต้องา ารให้ระบบทำการเริ่มต้นให้บริการทางด้านฐานข้อมูลโดยอัตโนมัติได้ตั้งแต่เริ่มการทำงานของระบบปฏิบัติการได้เลยหรือไม่ ในกรณีที่เราไม่ได้ตั้งค่าไว้แบบนี้ก็สามารถทำการเริ่มการทำงานเองได้ โดยมีขั้นตอนดังนี้

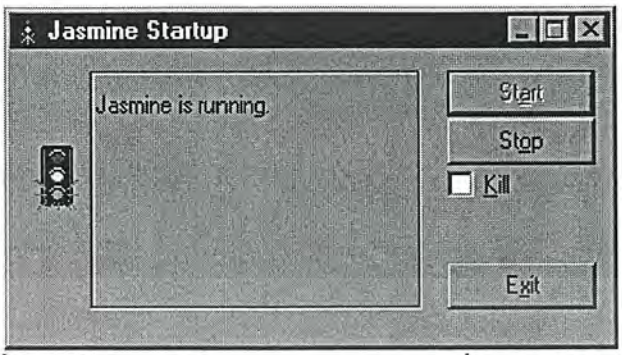
- เลือกไปที่ Start Menu ของ Windows และเลือก Program Group ของ Jasmine และเลือก Jasmine Startup จะปรากฏหน้าต่างของโปรแกรมนี้ขึ้นมา ดังรูปที่ ก-5 แล้วคลิกที่ปุ่ม Start



รูปที่ ก-5 หน้าต่างของโปรแกรม Jasmine Startup

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

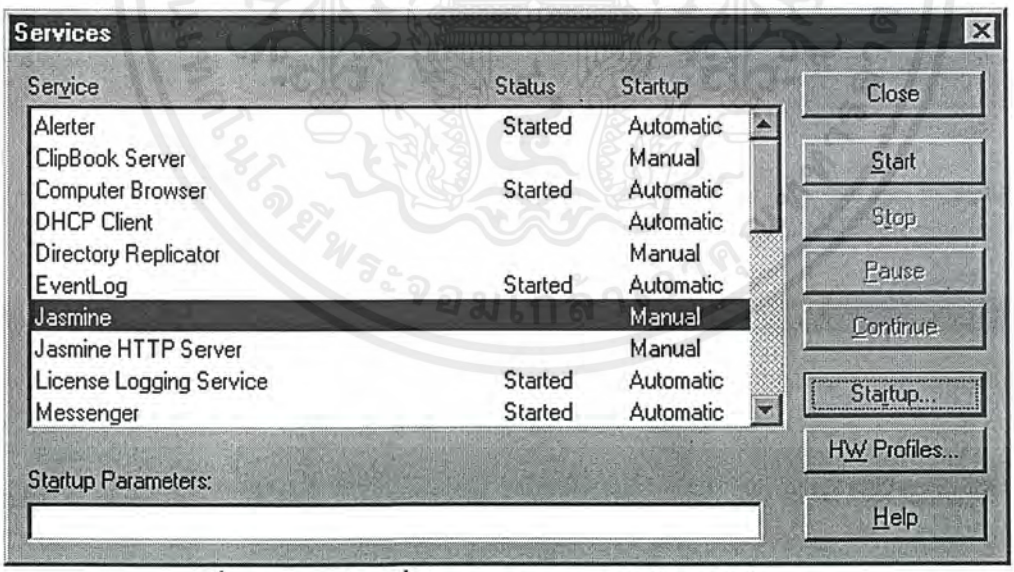
- โปรแกรมจะทำการเริ่มการให้บริการฐานข้อมูลและ Dialog Box ของโปรแกรมจะเปลี่ยนไปเป็นลักษณะดังรูปที่ ก-6



รูปที่ ก-6 หน้าต่างของโปรแกรม Jasmine Startup เมื่อกำลังเปิดให้บริการ

ในกรณีที่เราไม่ได้เลือกการ StartUp แบบอัตโนมัติซึ่งจะต้องทำการเปิดบริการผ่านทางโปรแกรม Jasmine StartUp เองดังที่ได้แสดงไว้ในตอนแรก แต่ต้องการให้มีการ StartUp บริการโดยอัตโนมัติตั้งแต่ตอนเปิดเครื่องเราก็สามารถเปลี่ยนได้โดยมีขั้นตอนวิธีการดังนี้

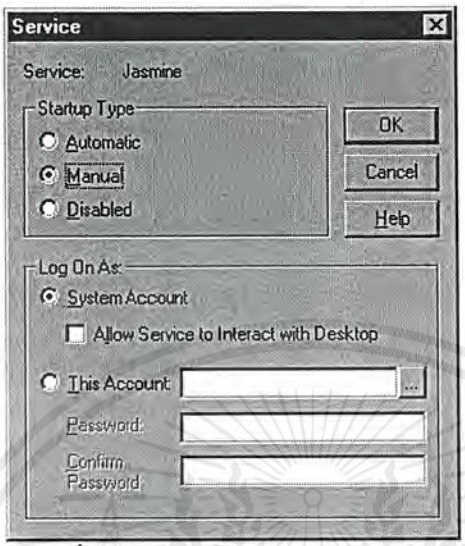
- ไปที่เมนูของ Control Panel และเลือก Services
- จะปรากฏรายการของบริการที่ตัวโปรแกรม Windows NT ให้บริการได้ ซึ่งมีลักษณะดังรูป ก-7 ซึ่งจากตรงนี้ให้เลือกไปที่บริการ Jasmine แล้วคลิกที่ปุ่ม Startup



รูปที่ ก-7 บริการต่างๆ ที่ Windows NT สามารถให้บริการได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- จะปรากฏหน้าต่างแสดงรายละเอียดบริการของ Jasmine ดังรูปที่ ก-8 ซึ่งหากต้องการเปลี่ยนแปลงรายละเอียดของบริการก็สามารถเลือกปรับแต่งได้



รูปที่ ก-8 หน้าต่างที่ใช้ในการปรับแต่งรายละเอียดของบริการ Jasmine

ก.5.4. ตัวอย่างการใช้งานโปรแกรมเพื่อทำการสร้างและเก็บข้อมูล

- การสร้าง Store ซึ่งหมายถึงการกำหนด Directory ที่ใช้ในการเก็บข้อมูลที่เกี่ยวข้องกับฐานข้อมูลที่สร้างขึ้นมา และการสร้าง Class Family ซึ่งเป็นการสร้างกลุ่ม Class ที่ใช้ในการเก็บข้อมูลเพื่อใช้ในการเก็บข้อมูลต่างๆ อย่างเป็นสัดส่วน จะใช้คำสั่งซึ่งเป็นคำสั่งแบบ Command Prompt มีตัวอย่างการสร้างดังนี้

createstore TestStore c:\test
createcf testCF TestStore

จากข้างต้นจะเป็นการสร้าง Store ที่ชื่อว่า *TestStore* ขึ้นมา และมี Class Family ที่ชื่อ *testCF* ใช้พื้นที่ส่วนนี้ในการเก็บข้อมูล

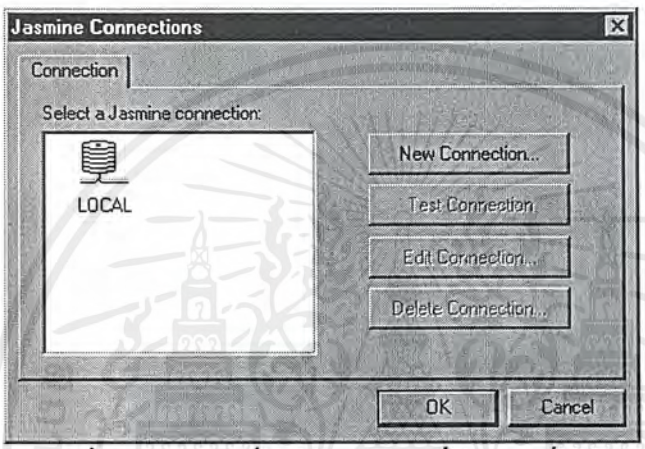
ก.5.5. การสร้าง Class เพื่อใช้ในการเก็บข้อมูลต่างๆ

ในโปรแกรมระบบจัดการฐานข้อมูล Jasmine นี้ ได้จัดเตรียมวิธีการสร้าง Class เพื่อใช้ในการเก็บข้อมูลไว้หลายวิธีด้วยกัน สำหรับในภาคผนวกส่วนนี้จะอธิบายแนวทางที่ใช้ในการจัดการฐานข้อมูลสองแบบด้วยกัน คือ ผ่านทาง ตัว Command-line Interpreter ซึ่งจะเป็นโปรแกรมที่ทำงานโดยการใช้ภาษา ODQL ซึ่งเป็นภาษาที่ใช้ในการจัดการฐานข้อมูลของโปรแกรมระบบจัดการฐานข้อมูลตัวนี้ และอีกแบบก็คือ การสร้าง Class ผ่านทางตัว Class Browser ของตัวโปรแกรม Jasmine Studio ซึ่งเป็นโปรแกรมที่ติดมากับชุดของระบบจัดการฐานข้อมูล Jasmine ซึ่งมีลักษณะการทำงานแบบ GUI ทำให้ง่ายต่อการใช้มาก สำหรับในการจัดการฐานข้อมูล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

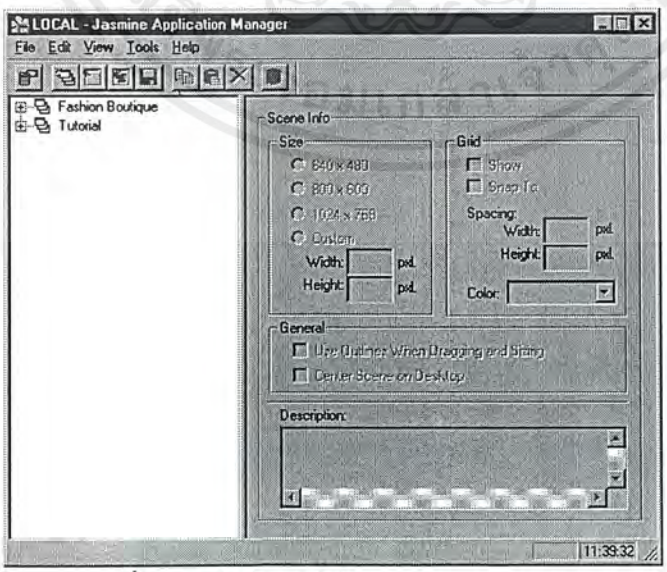
ที่เหลือ คือผ่านทางภาษาโปรแกรมอื่นๆ ได้แก่ Visual BASIC และ JAVA จะไม่ขออธิบาย แต่สำหรับในส่วนของการติดต่อโดยการใช้ C++ ซึ่งเป็นวิธีที่โครงงานนี้ใช้ในการติดต่อกับฐานข้อมูลก็จะมีการใช้ภาษา ODQL ตัวนี้เช่นเดียวกัน แต่จะมีรายละเอียดปลีกย่อยอื่นๆ เพิ่มเข้ามา เพื่อให้ตัวภาษาทั้งสองส่วนสามารถอ้างอิงถึงข้อมูลกันได้ รายละเอียดของการติดต่อฐานข้อมูลโดยใช้ C++ จะอธิบายในภาคผนวก ข. สำหรับในส่วนนี้จะอธิบายถึงการสร้าง Class โดยการใช้ Jasmine Studio ก่อน ซึ่งขั้นตอนการใช้งานเป็นดังนี้

- เรียกโปรแกรม Jasmine Studio โดยคลิกที่ Start Menu เลือก Program Group Jasmine แล้วเลือก Jasmine Studio จะปรากฏหน้าต่างของโปรแกรมและหน้าต่างเพื่อให้เลือกว่าต้องการทำงานโดยเชื่อมต่อกับ Database Server ตัวไหนดังรูป ก-9



รูปที่ ก-9 หน้าต่างเพื่อให้เลือก Serverที่ต้องการเชื่อมต่อ

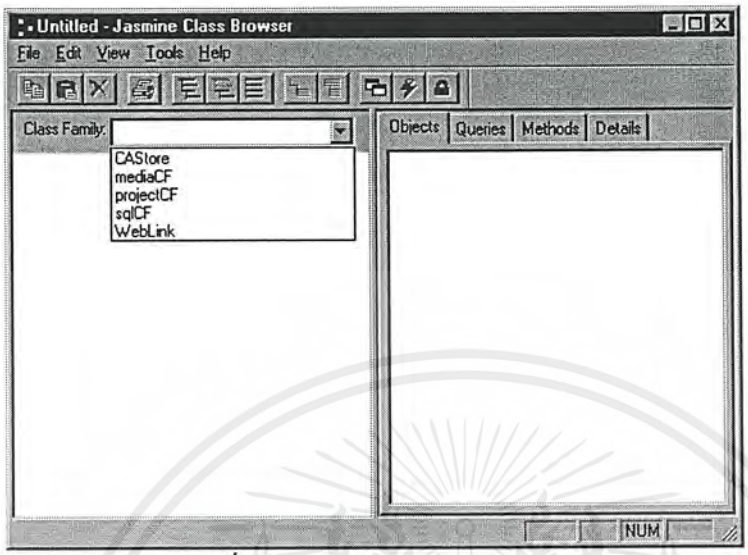
- หลังจากที่ได้เลือก Server ที่ต้องการเชื่อมต่อแล้วการทำงานจะอยู่ในส่วนของหน้าต่าง Jasmine Studio ดังรูป ก-10 เลือกไปที่ Menu File แล้วเลือก Database Administration



รูปที่ ก-10 หน้าต่างของโปรแกรม Jasmine Studio

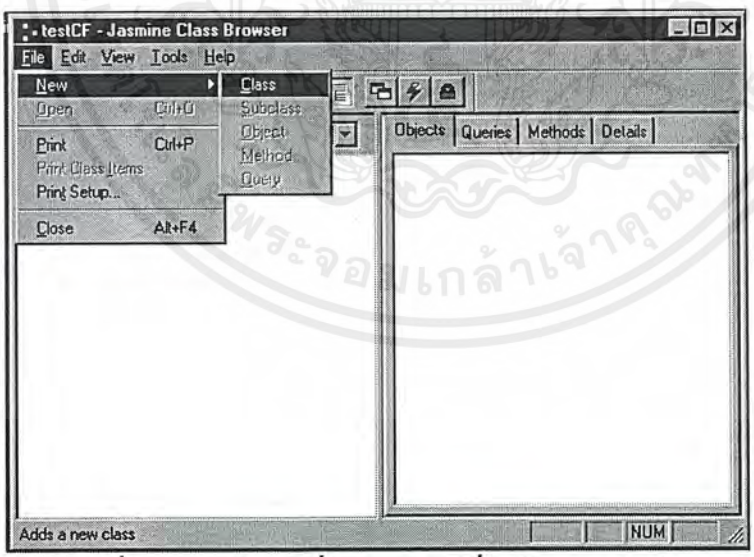
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- จะปรากฏหน้าต่าง Class Browser ขึ้นมา ให้เราทำการเลือก Class Family ที่เราต้องการสร้าง Class ดังรูปที่ ก-11



รูปที่ ก-11 หน้าต่างของ Class Browser

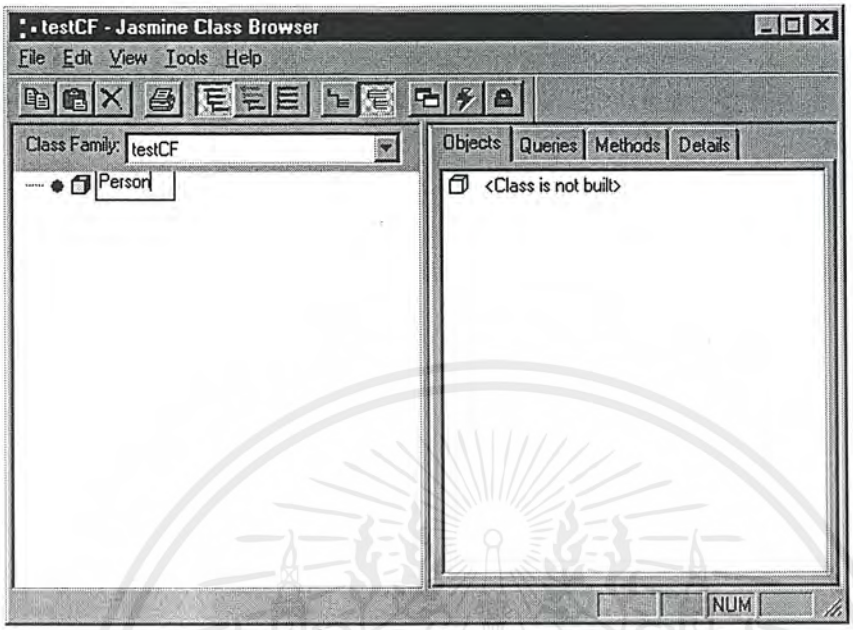
- ต่อจากนี้ไปจะเป็นรายละเอียดของการสร้าง Class โดยผ่านทาง Class Browser ตัวนี้ โดยในขั้นตอนแรกให้เลือกไปที่ Menu แล้วเลือกที่ File > New > Class ดังรูป ก-12



รูปที่ ก-12 การเลือกคำสั่งจาก Menu เพื่อสร้าง Class ใหม่

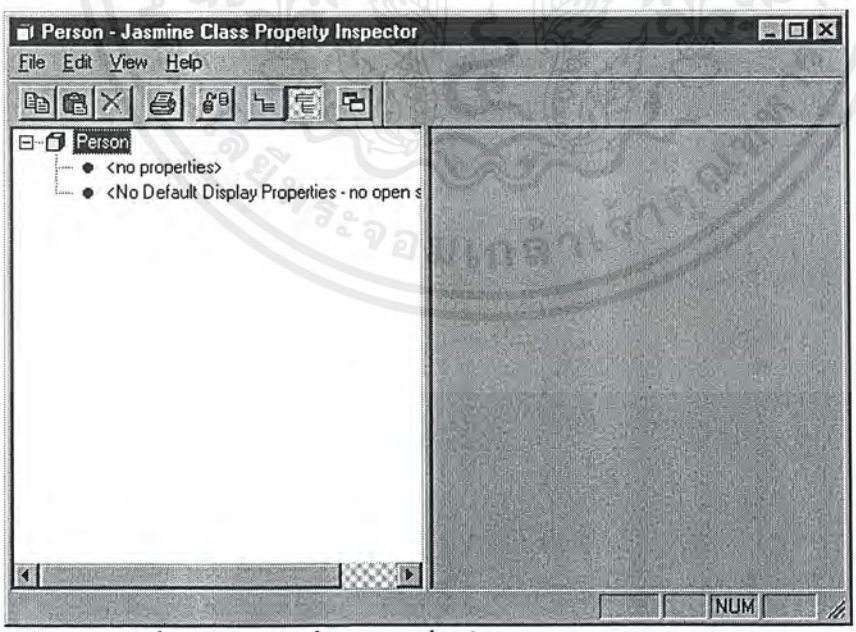
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- จะปรากฏ Class ใหม่ขึ้นมาชื่อว่า New_Class_X เมื่อ X เป็นตัวเลขใดๆ ให้เราทำการเปลี่ยนชื่อ Class นี้ให้เป็นชื่อตามที่เราต้องการ



รูปที่ ก-13 การสร้าง Class ใหม่

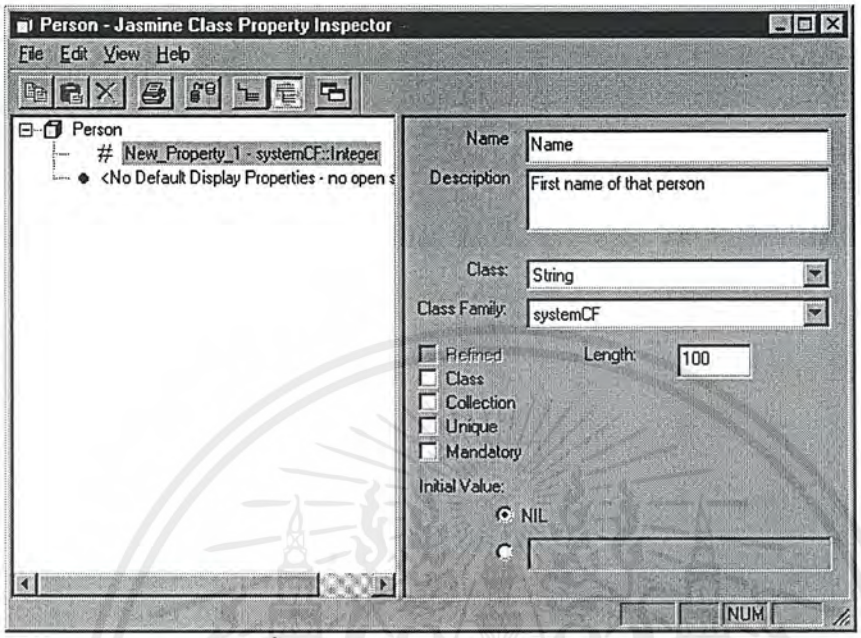
- หลังจากได้ชื่อ Class ที่เราต้องการแล้ว ให้ทำการ Double Click ที่ Class นี้ เพื่อเปิดหน้าต่างที่ใช้ทำการกำหนดค่า Property ต่างๆ ของ Class นั้นๆ ดังรูป ก-14



รูปที่ ก-14 หน้าต่างที่ใช้ทำงานเพื่อกำหนด Property ให้กับ Class

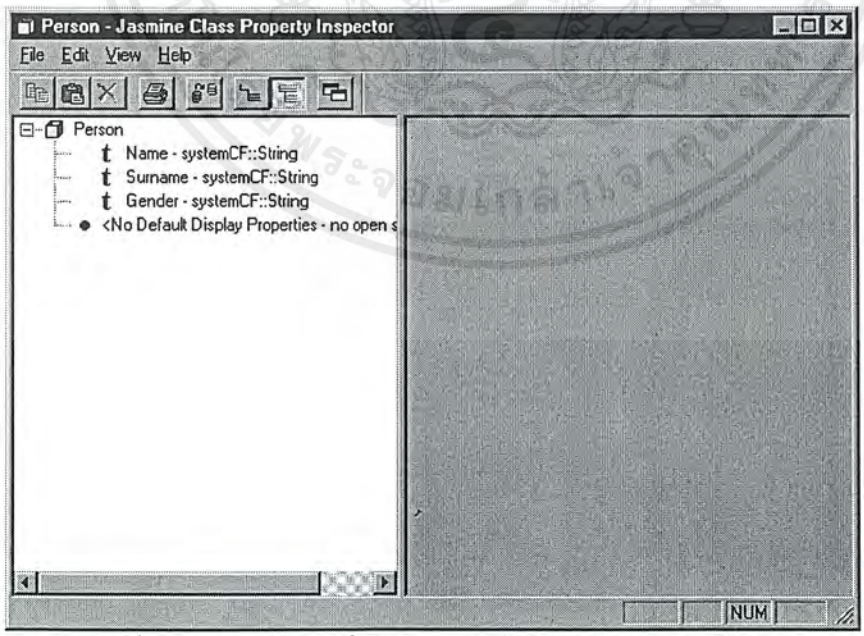
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- สำหรับการสร้าง Property ตัวใหม่ขึ้นมา จะใช้วิธีการ Click Mouse ปุ่มขวาเพื่อเรียก Menu จากนั้นให้เลือก New > Property แล้วทำการพิมพ์ชื่อ Property , รายละเอียดและรูปแบบข้อมูลตามต้องการ ดังรูป ก-15



รูปที่ ก-15 การกำหนดรูปแบบให้กับ Property

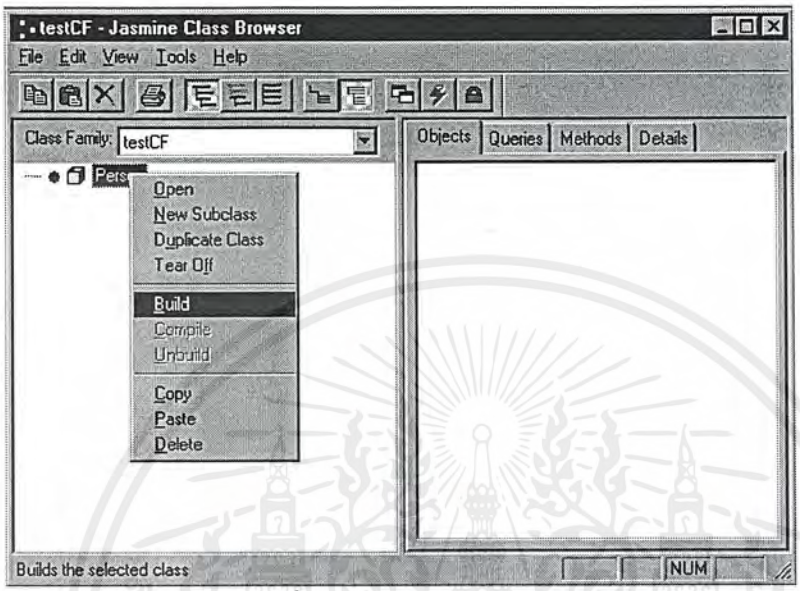
- เมื่อทำการกำหนดรูปแบบให้กับ Property เสร็จแล้ว ต่อไปก็ทำการกำหนด Property ตัวใหม่ไปเรื่อยๆ จนได้ Class ที่สมบูรณ์ดังรูป ก-16



รูปที่ ก-16 Class Person ที่มีค่า Property ต่างๆ ครบถ้วนสมบูรณ์แล้ว

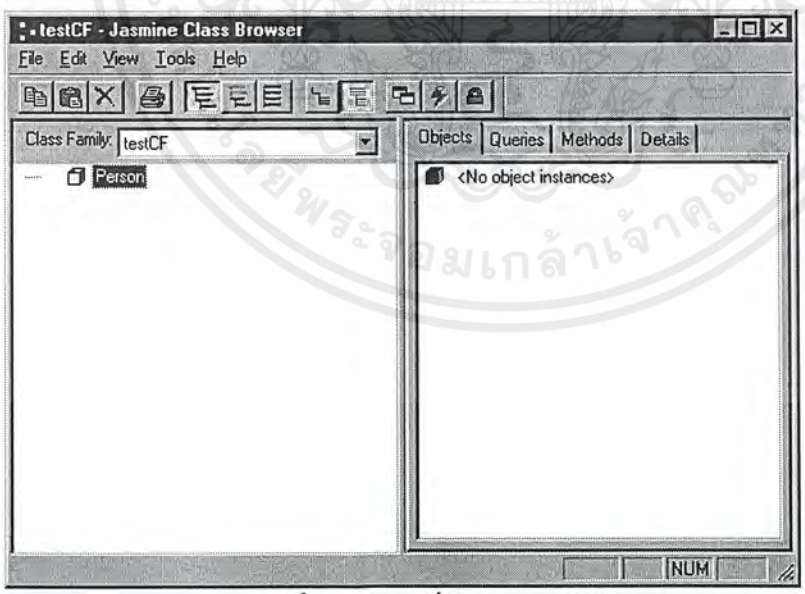
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- เมื่อสร้าง Class ที่มีลักษณะตามต้องการแล้วก็ปิดหน้าต่างที่ใช้สร้าง Property จะกลับมายังหน้าต่าง Class Browser อีกครั้งหนึ่ง แล้วเลือก Class ที่เราสร้างเสร็จแล้ว Click Mouse ปุ่มขวาเลือก Menu Build เพื่อทำการเก็บข้อมูลของ Class นั้นในฐานข้อมูล เพื่อที่จะได้สร้าง Object ที่เป็นของ Class นั้นได้ ดังรูป ก-17



รูปที่ ก-17 การสร้าง Build Class

- สังเกตวงกลมสีแดงที่อยู่หน้าชื่อ Class จะหายไปดังรูป ก-18



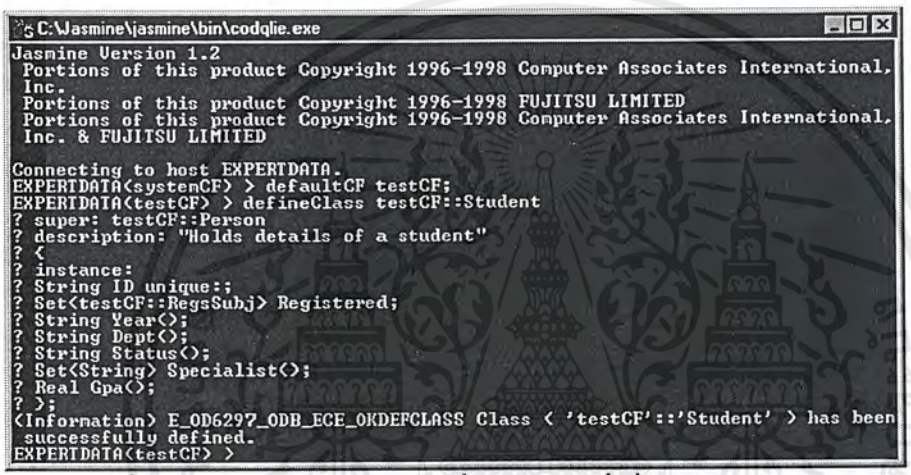
รูปที่ ก-18 Class ที่สร้างสมบูรณ์แล้ว

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- อีกตัวอย่างหนึ่งก็คือการสร้างโดยผ่านทาง ODQL โดยในตอนแรกให้ทำการเรียกโปรแกรม codqlie จะปรากฏหน้าจอของโปรแกรม ในตอนแรกให้พิมพ์คำสั่ง

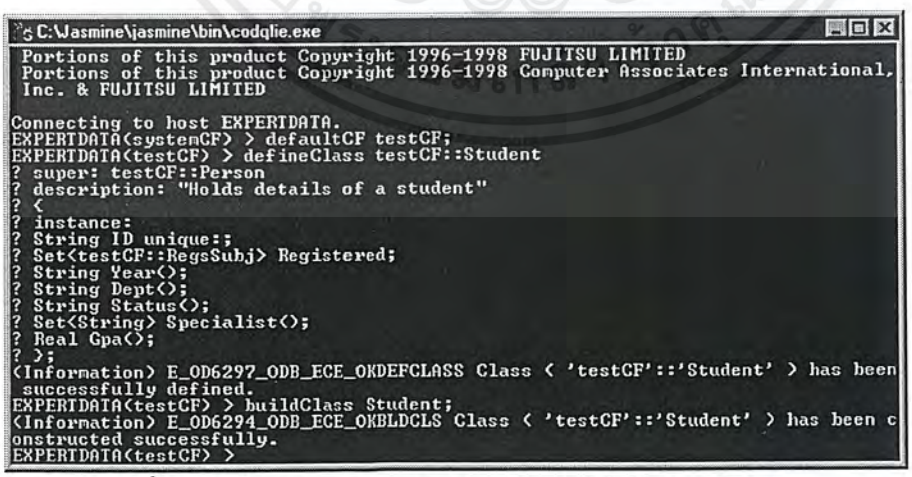
defaultCF testCF;

จะเป็นการกำหนด Class Family ที่ใช้ในการทำงานเพื่อความสะดวกในการอ้างอิงถึง Class ต่างๆ ทำการพิมพ์คำสั่ง defineClass เพื่อทำการสร้าง Class โดยในตัวอย่างที่แสดงในรูป ก-19 เป็นการสร้าง Class ชื่อ Student สำหรับรายละเอียดของคำสั่งจะไม่อธิบายในภาคผนวกนี้ แต่จะสามารถดูได้ใน Online Documentation



รูปที่ ก-19 หน้าจอโปรแกรม codqlie เมื่อทำการใส่คำสั่งที่ใช้สร้าง Class Student

- หลังจากนั้นทำการสร้าง Class ได้โดยการใช้คำสั่ง buildClass



รูปที่ ก-20 หน้าจอแสดงการ Build Class โดยการใช้โปรแกรม codqlie

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

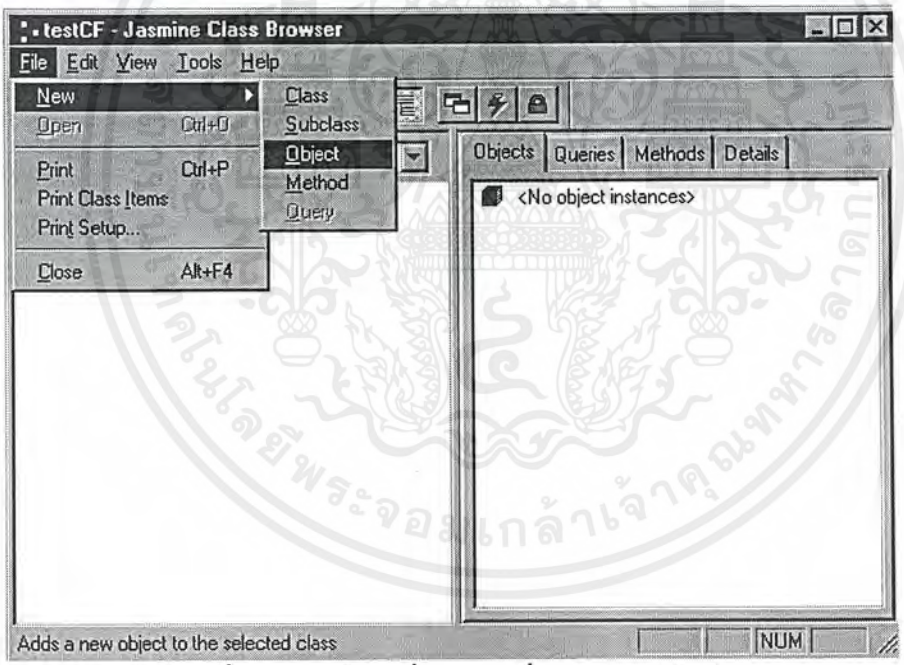
ก.5.6. การสร้าง Object

สำหรับในวิธีการสร้าง Object สามารถสร้างได้โดยทั้งผ่านทางการใช้ภาษา ODQL และผ่านทางการใช้โปรแกรม Jasmine Studio สำหรับการสร้าง Object ผ่านทางภาษา ODQL นั้นจะมีข้อแตกต่างจากการสร้างโดยการใช้ภาษา SQL ก็คือในการสร้างนั้น จำเป็นต้องมีการประกาศตัวแปรขึ้นมารองรับ Object ที่จะสร้างขึ้นมาใหม่ก่อนแล้วจึงทำการสร้างได้ โดยรูปแบบคำสั่งจะมีลักษณะดังนี้

```
Person p;
p = Person.new(Name := "Surasak",
               Surname := "Watthanayontkit",
               Gender := "Male");
```

และถ้าเป็นการสร้างผ่านทาง Class Browser ของ Jasmine Studio ก็สำหรับ Object ที่มีค่า Property ต่างๆ ที่เหมือนกันกับตัวอย่างภาษา ODQL ข้างต้นก็สามารถสร้างได้ตามขั้นตอนดังนี้

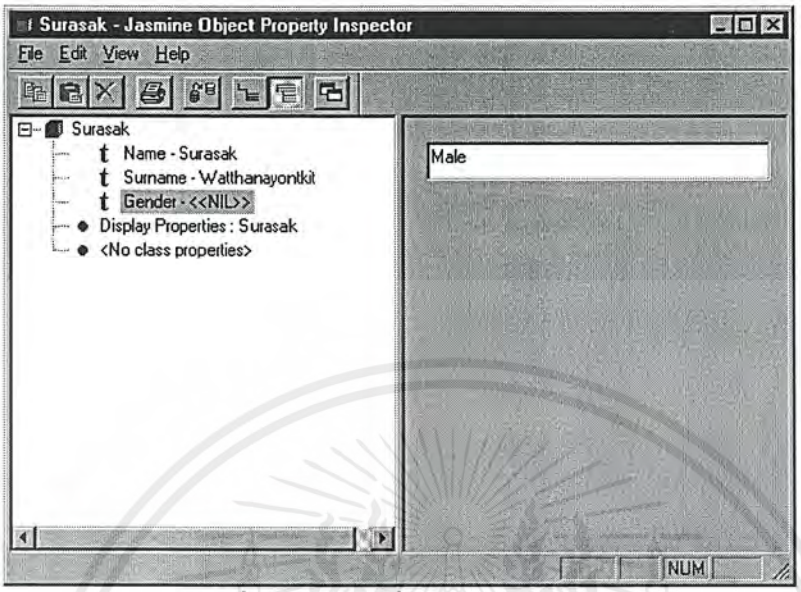
- จากหน้าต่าง Class Browser เรียก Menu คำสั่ง New > Object ดังรูป ก-21



รูปที่ ก-21 การเลือกคำสั่งจากเมนูเพื่อสร้าง Object ใหม่

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- จะปรากฏหน้าต่างใหม่ขึ้นมาเพื่อใช้ในการกำหนดค่า Property ที่ใช้ในการสร้าง Object ตัวใหม่ดังรูป ก-22

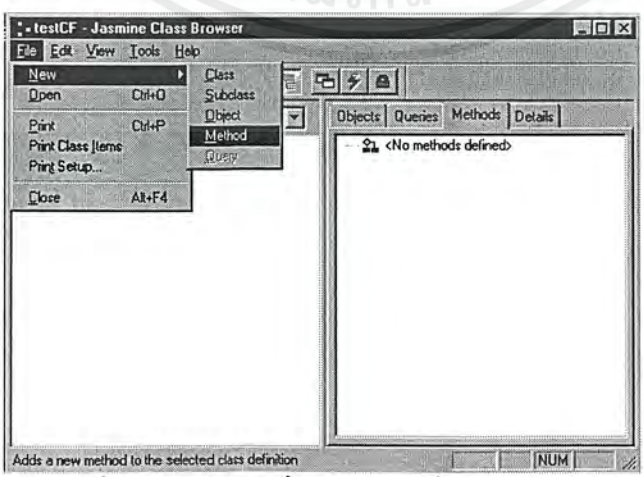


รูปที่ ก-22 หน้าต่างที่ใช้ในการสร้าง Object

ก.5.7. การสร้าง Method

สำหรับการสร้าง Method ของ Jasmine ก็สามารถสร้างได้หลายวิธีเช่นเดียวกับการสร้าง Class และการใช้งานอื่นๆ แต่สำหรับในภาคผนวกส่วนนี้จะแสดง ถึงวิธีการสร้างโดยผ่านทาง Class Browser ของโปรแกรม Jasmine Studio ซึ่งเป็นวิธีที่สะดวกกว่าการใช้โปรแกรม codqlie เนื่องจากมีตัว Editor ไว้สำหรับการพัฒนาทำให้การแก้ไขต่างๆ ทำได้ง่ายกว่า สำหรับขั้นตอนการสร้างมีดังนี้

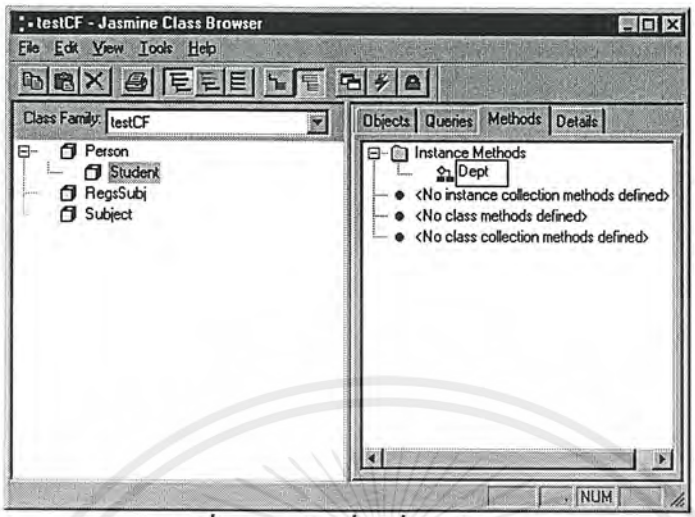
- จากหน้าต่าง Class Browser Click Mouse เลือก Class ที่ต้องการสร้าง Method แล้วเรียก Menu เลือกคำสั่ง New > Method ดังรูป ก-23



รูปที่ ก-23 การเลือกคำสั่งจาก Menu เพื่อสร้าง Method

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- จะปรากฏ Method ขึ้นมาชื่อว่า New_MethodX เมื่อ X เป็นตัวเลขใดๆ ให้เราทำการเปลี่ยนชื่อ Method ตามต้องการ ดูรูป ก-24



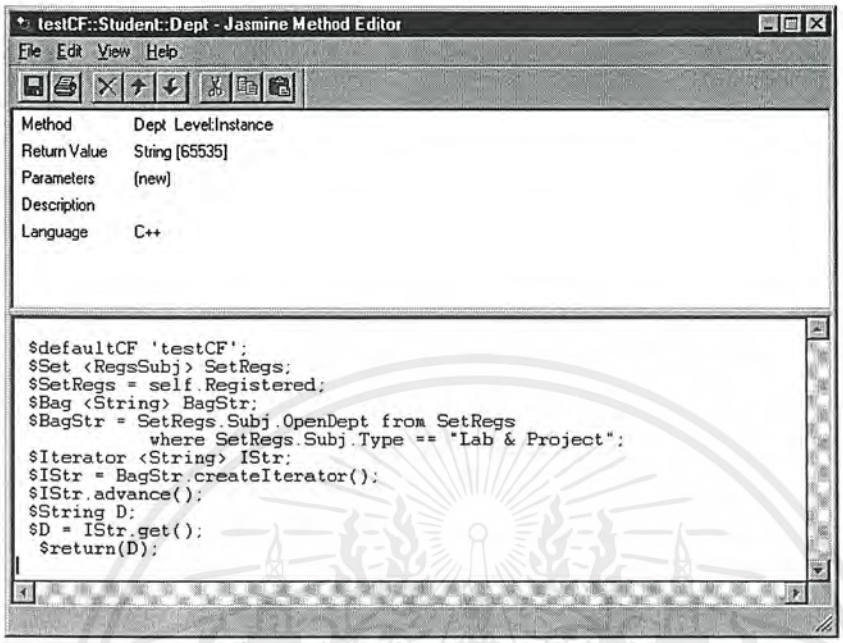
รูปที่ ก-24 การเปลี่ยนชื่อ Method

- จากในหน้าต่าง Class Browser เลือก Method ที่ต้องการแก้ไขหรือเขียนใหม่ แล้วเรียก Menu เลือกคำสั่ง Open จะปรากฏหน้าต่าง Editor ที่ใช้ในการสร้าง Method ดังรูป ก-25

รูปที่ ก-25 การเปิดหน้าต่างเพื่อทำการเขียน Method

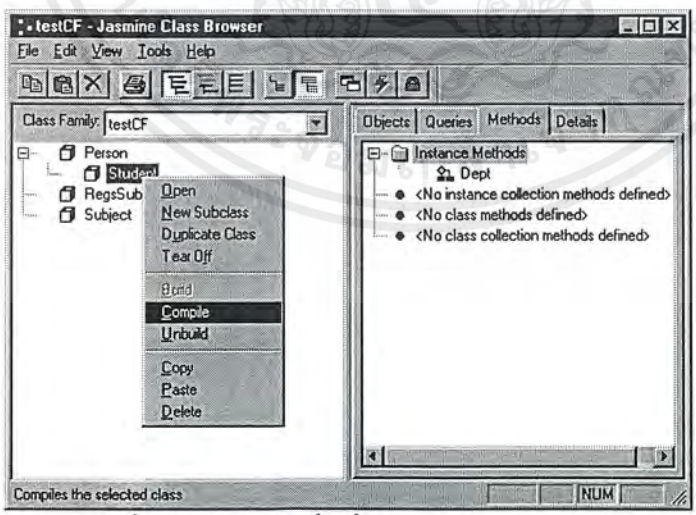
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ร สร้าง Method โดยกำหนดรายละเอียดต่างๆ ที่ต้องการ จากตัวอย่างในรูป ก-26 จะเป็นการสร้าง Method ชื่อ Dept() ของ Class Student



รูปที่ ก-26 รายละเอียดของ Method Dept()

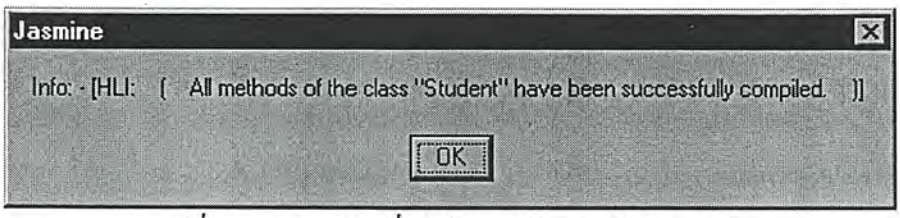
- ทำการบันทึกข้อมูลที่เราได้ทำการเขียนเอาไว้แล้วปิดหน้าต่าง Editor จากนั้นกลับไปยังหน้าต่าง Class Browser อีกครั้งแล้วเลือก Class ที่เราได้ทำการสร้าง Method นั้นๆ Click Mouse ปุ่มขวาแล้วเลือกคำสั่ง Compile ดังรูป ก-27



รูปที่ ก-27 การเลือกคำสั่งเพื่อทำการ Compile Method

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- โปรแกรมจะทำการ Compile Method ทั้งหมดที่อยู่ใน Class นั้นๆ ซึ่งถ้าการ Compile สำเร็จโดยไม่มีข้อผิดพลาดใดๆ ก็จะปรากฏ Dialog Box ลักษณะดังรูปที่ ก-28



รูปที่ ก-28 Dialog Box ที่แจ้งว่าการ Compile ทำได้สมบูรณ์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาคผนวก ข.

วิธีการติดต่อกับระบบจัดการฐานข้อมูล Jasmine

สำหรับวิธีการติดต่อกับฐานข้อมูล Jasmine นั้นทางผลิตภัณฑ์ได้ให้วิธีการที่จะใช้ในการติดต่ออยู่หลายแบบ แต่ในงานวิจัยชิ้นนี้จะมุ่งความสนใจไปที่ส่วนของการใช้ภาษา ODQL ซึ่งเป็นภาษาทางด้านฐานข้อมูลที่ทางผู้พัฒนาโปรแกรมได้พัฒนาขึ้นเพื่อใช้สำหรับการทำงานกับระบบจัดการฐานข้อมูลตัวนี้ และวิธีการสร้างโปรแกรมประยุกต์ใช้งานโดยการใช้ C API ที่ทางผู้ผลิตจัดเตรียมไว้ให้เป็นหลักซึ่งต้องมีการใช้งานร่วมกันกับในส่วนของภาษา ODQL ด้วย ซึ่งในส่วนของการติดต่อทั้งสองแบบที่ทางงานวิจัยได้ทำการศึกษาก็เป็นวิธีหลักที่ใช้ในการทำการติดต่อกับฐานข้อมูลของผลิตภัณฑ์ชิ้นนี้ด้วย

ในส่วนของการติดต่อกับฐานข้อมูลโดยการใช้วิธีการข้างต้นดังกล่าวนั้นก็สามารถนำมาแบ่งแยกออกได้ตามลักษณะของงานที่นำไปใช้ซึ่งจะมีรูปแบบในการเฉพาะที่แตกต่างกันไป แต่ก่อนที่จะอธิบายในส่วนของการใช้งาน ในเนื้อหาส่วนต่อไปจะขออธิบายถึงลักษณะของภาษา ODQL โดยสังเขปเสียก่อนเพื่อให้เกิดความเข้าใจในส่วนของการนำไปใช้งานได้ดียิ่งขึ้น

ข.1. ลักษณะของภาษา ODQL

ดังที่ได้อธิบายไว้ในข้างต้นแล้วว่า ภาษาตัวนี้เป็นภาษาที่ใช้ในการทำงานทางด้านฐานข้อมูลที่สร้างขึ้นมาเพื่อใช้โปรแกรมจัดการระบบจัดการฐานข้อมูลตัวนี้ ซึ่งจะมีลักษณะที่คล้ายกับภาษาซี ในส่วนของภาษาก็มีลักษณะที่ปรากฏอยู่ในภาษาโปรแกรมอื่นๆ อยู่ครบถ้วน คือ

- ความสามารถในการทำการประกาศการใช้ตัวแปร (declare), กำหนดค่าที่จะใช้ ในตัวแปร (assign) และวิธีการที่จะใช้งานตัวแปรนั้นๆ
- ความสามารถในการทำงานแบบเลือกทำตามเงื่อนไข (Conditional)
- ความสามารถในการทำงานแบบวนซ้ำ (Iteration)

ลักษณะของวิธีการที่ตัวภาษาใช้เพื่อทำงานดังกล่าวข้างต้นนั้นก็จะมีลักษณะที่คล้ายคลึงกับภาษาซีอยู่มากซึ่งสามารถดูได้จากวิธีการของภาษาที่ใช้ในการแสดงในส่วนต่างๆ ดังตัวอย่างต่อไปนี้

- การประกาศตัวแปร มีรูปแบบการประกาศโดยการขึ้นต้นด้วยลักษณะของข้อมูล (Type) และชื่อของตัวแปรที่จะใช้ตามด้วยเครื่องหมายเซมิโคลอนเป็นการจบข้อความ

  Person p;

  หมายถึงการกำหนดให้ตัวแปรชื่อ p เป็นตัวแปรที่ใช้สำหรับการตัวแทน (Instance) ของคลาส (Class) ชื่อ Person หรือคลาสที่สืบทอดมาจากคลาสนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Bag <Person> pp;

หมายถึงการกำหนดให้ตัวแปรชื่อ pp เป็นตัวแปรชนิด Bag ที่เก็บกลุ่มของตัวแปรซึ่งตัวแปรทุกตัวเป็นตัวแทนจากคลาสของ Person หรือคลาสที่สืบทอดมาจากคลาสนี้ หรือค่าพิเศษคือ NIL

- การกำหนดค่าต่างๆ ให้ตัวแปรมีลักษณะเช่นเดียวกับภาษาโปรแกรมอื่นๆ ดูได้จากตัวอย่าง

```
Integer n,m;
Bag <Integer> nn;
String s;
n =17;                    (กำหนดให้ n มีค่าเป็น 17)
m = NIL;                  (กำหนดให้ m มีค่าเป็น NIL)
nn = Bag{12,14,17};       (กำหนดให้ nn ประกอบด้วยค่า 12,14 และ 17)
s = "Canada";             (กำหนดให้ s มีค่าเป็น "Canada")
```

- วิธีการใช้งานตัวแปรในส่วนที่เป็น Expression ดังตัวอย่าง

```
Boolean b;
Integer n;
Person P;
Person class m;

...

b = (n > 5 and p.spendingLimit() < 100.00) or
                                (p.getInsToClass() == m);
```

เป็นกำหนดให้ b มีค่าเป็นจริงเมื่อ n มีค่ามากกว่า 5 และค่า spendingLimit() ในตัวแปร p น้อยกว่า 100 ทั้งสองอย่างหรือเมื่อค่า getInsToClass ในตัวแปร p มีค่าเท่ากับค่าในตัวแปร m

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- ตัวอย่างวิธีการทำงานในส่วนของการทำงานแบบเลือกทำตามเงื่อนไข

```
if (n > 0) {
  msg = "Value of n is negative\n";
  msg.print()'
}
else {
  p = n * m;
};
```

จะเป็นการทำงานที่จะพิมพ์ข้อความ Value of n is negative เมื่อค่าใน n มากกว่าศูนย์ มิเช่นนั้นก็จะทำการกำหนดค่าของ p ให้เท่ากับค่าของ n คูณกับค่าของ m

- ตัวอย่างของวิธีการทำงานแบบวนซ้ำ

```
while (n < 5) {
  msg = String.format("Value is %s\n",n);
  msg.print();
  n = n + 1;
};
```

เป็นการทำงานในเงื่อนที่ว่าจะทำงานในส่วนของวงเล็บปีกกา ({ - }) ไปเรื่อยๆ จนกว่าค่าใน n จะน้อยกว่า 5

อีกตัวอย่างหนึ่งของรูปแบบการสแกนค่าในตัวแปร

```
Bag <Integer> nn;
Integer n,m;
nn = Bag{2,3,5,7};
scan (nn,n) {
  m = 2 * n;
m.print();
};
```

เป็นการทำงาน โดยให้ค่า n ถูกแทนค่าด้วยค่าที่อยู่ใน nn แต่ละค่าและทำงานไปเรื่อยๆ จนค่าในกลุ่มตัวแปรถูกแทนครบหมดทุกตัว

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- ตัวอย่างของการทำงานเกี่ยวกับด้านฐานข้อมูล

การทำงานแบบ Query

```
List <Person> pp;
pp =  Person from Person alone
  where Person.grade == "10";
```

เป็นการสร้างให้ข้อมูลใน pp มาจากข้อมูลในคลาส Person เพียงอย่างเดียว ซึ่งค่าที่นำมาใส่ในตัวแปร pp จะต้องมีค่า grade เท่ากับ 10

พิจารณาอีกตัวอย่างของการทำงานแบบคิวรี

```
T[String name,mediaCF::Bitmap photo] set ts;
ts = [x.name, x.photo] from Top x
  where x.hot > 0;
```

เป็นการกำหนดค่าข้อมูลให้กับตัวแปร ts ซึ่งเป็นตัวแปรชนิด set ของ tuple ที่ประกอบด้วยตัวแปร name เป็น String และ photo เป็น Bitmap ซึ่งเป็นรูปแบบ Class ที่อยู่ใน class family ชื่อ mediaCF โดยจะเก็บค่า name และ photo จาก instance ของคลาส Top ที่มีค่า hot มากกว่า 0

การสร้าง/ลบออบเจ็กต์ (Object)

```
Location loc;
loc = Location.new(name := "London");
...
loc.delete();
```

ข.2. รูปแบบการใช้งานภาษาODQLแบบต่างๆ

จากในส่วนที่แล้วเราก็ได้ทราบถึงลักษณะของภาษาODQLโดยคร่าวๆแล้ว ต่อไปในส่วนนี้จะเป็นการอธิบายถึงการใช้งานในรูปแบบต่างๆ ซึ่งมีรูปแบบต่างๆ ดังต่อไปนี้

ข.2.1. แบบ Interpreted ODQL

เป็นรูปแบบการป้อนคำสั่งที่เป็นภาษาODQLหรือเรียกไฟล์ที่ภายในมีของโค้ด (Code) ของภาษา ODQL ผ่านตัวโปรแกรม ODQL อินเตอร์พรีเตอร์ (ODQL Interpreter) ที่มีลักษณะการทำงานเป็นแบบคอมมานด์ไลน์ (Command Line) ผ่านทางตัวเทอร์มินอล (Terminal) ซึ่งในที่นี้ก็คือตัวโปรแกรมเอง เมื่อพิมพ์คำสั่งต่างๆ เข้าไปแล้วโปรแกรมจะแสดงคำตอบของคำสั่งที่ผู้ใช้ใส่เข้าไป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ข.2.2 แบบ Embedded ODQL

เป็นลักษณะการเขียนโปรแกรมโดยการผสมโค้ดของภาษา ODQL เข้ากับส่วนของภาษาหลัก (Host Language)ซึ่งในที่นี้ก็คือภาษาซีหรือซีพลัสพลัส การผสมโค้ดในส่วนของภาษาทั้งสองนั้นสามารถทำได้อย่างอิสระโดยจะเขียนโค้ดของภาษา ODQL ในส่วนไหนก็ได้ แต่มีข้อจำกัดว่าถ้าเป็นโค้ดของส่วนที่เป็นภาษา ODQL จะต้องขึ้นต้นบรรทัดด้วยเครื่องหมายดอลลาร์ ($) ดังตัวอย่าง

```
$Boolean b <bval,bstat>;
$String s[40] <sval,sstat>;
$if (b) {
 $p = Person.find("John Carter");
 $s = p.surname;
 if (sstat == ODB_STATNIL) {
  printf("Surname is NIL\n");
 }
 else {
  printf("Surname is %s\n",sval);
 }
};
```

สำหรับในการเขียนโค้ดของทั้งสองภาษาผสมกันเช่นนี้ ถ้าในส่วนของภาษาหลักต้องการที่จะอ้างอิงหรือเรียกใช้ตัวแปรที่อยู่ในส่วนของภาษา ODQL ทางผู้ผลิตก็มีการจัดเตรียมกลไกที่ใช้ในการแมป (map) โดยการใช้ตัวแปรในภาษาหลักหนึ่งคู่ต่อตัวแปรใน ODQL หนึ่งตัว ตัวแปรที่ใช้ในภาษาหลักตัวหนึ่งจะใช้สำหรับการแมปค่าสถานะของตัวแปรอีกตัวหนึ่งที่แมปมา ซึ่งจะมีค่าที่เป็นไปได้ดังนี้

ODB_STATVALID : สามารถใช้ได้ตามปกติ

ODB_STATNIL : หมายความว่ามีค่าเป็น NIL

ODB_STATERROR : หมายความว่ามีค่าเป็น ERROR

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ส่วนตัวแปรอีกตัวหนึ่งจะใช้สำหรับเก็บค่าจริงของตัวแปรซึ่งประเภทของตัวแปรที่สอดคล้องกันในส่วนของภาษาหลักและของภาษา ODQL จะมีรูปแบบการแมปกันดังนี้

ODQL:	C/C++
Integer	long
Real	double
String	char*
ByteSequence	ODB_BYTESEQ
Boolean	char
Date	ODB_DATE
คลาสแบบกำหนดเอง	ODB_OBJECT_ID

สำหรับตัวอย่างวิธีการนำไปใช้เป็นดังนี้

```
$Integer n <nval,nstat>;
```

จากตัวอย่าง ตัวแปร n ที่อยู่ใน ODQL จะถูกแมปด้วยตัวแปรในภาษาหลักสองตัวคือ nval ซึ่งจะใช้ในการเก็บค่าจริงๆของตัวแปร n ซึ่งชนิดของ nval ก็จะเป็นแบบ long ตามรูปแบบการแมปตามที่แสดงในตารางข้างต้นโดยอัตโนมัติ

พิจารณาการใช้โดยละเอียดกับตัวอย่างอีกตัวอย่าง

```
$Bag<Person> pp;
$Person p;
$String s<sval,sstat>
$scan(pp,p) {
   $s=p.surname;
   if (sstat == ODB_STATVALID) {
      if (islower(*sval)) {

         ...

      }
   }
};
```

จากตัวอย่างเมื่อมีการประกาศตัวแปรพร้อมกับตัวแปรที่ใช้ในการแมปแล้วก็สามารถนำตัวแปรนั้นไปใช้ในส่วนของภาษาหลักได้ทันที

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

อย่างไรก็ตาม สำหรับตัวแปรในรูปแบบอื่นๆ ของ ODQL ที่ไม่ได้กำหนดไว้ในตารางจะไม่สามารถทำการแมปมายังภาษาหลักได้

สำหรับการคอมไพล์โค้ดของภาษาหลักที่มีการผสมในส่วนของภาษา ODQL เข้าไปด้วยจะมีแปลงโค้ดในส่วนที่เป็นภาษา ODQL ให้เป็นภาษาหลักเสียก่อนซึ่งคอมไพเลอร์แต่ละตัวจะมีลำดับขั้นตอนการทำงานในส่วนของการคอมไพล์ส่วนของ Header (ไฟล์ที่มีส่วนขยายเป็น .h) ทำให้เกิดข้อจำกัดของวิธีการนี้ก็คือ ทำให้ไม่สามารถเขียนส่วนของภาษา embedded ODQL นี้ในส่วนของ Header file ได้

ข.2.3. แบบใช้ผ่านทาง C API

สำหรับการพัฒนาโปรแกรมที่มีการเรียกใช้ถึงตัวระบบจัดการฐานข้อมูล Jasmine ทางโปรแกรมจัดการก็มีจะจัดเตรียมส่วนของ API ที่ใช้ในการติดต่อกับฐานข้อมูลไว้ให้ เพื่อให้ผู้ที่พัฒนาโปรแกรมด้วยภาษา C หรือ C++ ได้รับความสะดวกในการเรียกใช้บริการที่ตัวโปรแกรมจัดเตรียมไว้ให้ โดยรูปแบบในการใช้ API เหล่านี้ จะเป็นการใช้งานร่วมกับภาษา ODQL โดยจะส่งภาษา ODQL ไปยังส่วนของ Server ในรูปของ String ซึ่งจะเป็นอาร์กิวเมนต์ตัวหนึ่งที่ประกอบอยู่ในฟังก์ชันที่เตรียมมาให้

ลักษณะของโปรแกรมที่มีการเรียกใช้บริการของฐานข้อมูลประกอบด้วยส่วนต่างๆ ต่อไปนี้อยู่ในโปรแกรม

- มีการเรียก include header file ชื่อ codqlhdr.h ดังตัวอย่าง โปรแกรมทุกตัวที่มีการใช้ C API ต้องมีการกำหนดส่วนนี้ไว้

```
#include <codqlhdr.h>
```

- มีส่วนที่ใช้เรียกเพื่อทำการเปิดการติดต่อและยุติการติดต่อกับฐานข้อมูล ถ้ามีการเรียกใช้ฐานข้อมูล ต้องมีส่วนนี้ในโปรแกรมด้วย โดยมีลักษณะการใช้ดังตัวอย่าง

```
odbSessH    sh;
odbStatus   odbrc;
char        *user;
char        *pw;

...

odbrc = odbSesStart(&sh, "jasmine/jasmine",
                                    user,pw, "envfile.env");

...

odbSesEnd(sh);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากตัวอย่าง ในฟังก์ชัน odbSesStart() จะมีการผ่านค่าอาร์กิวเมนต์อยู่หลายตัวด้วยกัน ตัวแปร sh คือตัว Session Handle ซึ่งจะเป็นตัวที่เก็บค่าที่ใช้ในการติดต่อกับฐานข้อมูล ซึ่งทุกครั้งที่มีการติดต่อต้องติดต่อผ่านตัวแปรตัวนี้ทุกครั้ง ส่วนอาร์กิวเมนต์ตัวอื่นๆ จะบ่งบอกถึง Directory ที่เป็นที่ที่ระบบจัดการฐานข้อมูลที่ต้องการติดต่อได้ทำการติดตั้งไว้, ข้อมูล account และ password ของผู้ที่ขอทำการติดต่อ และชื่อของ environment file ซึ่งเป็นไฟล์ที่ใช้เก็บข้อมูลที่เกี่ยวกับลักษณะจำเพาะที่โปรแกรมนั้นๆ ต้องการเพื่อทำงานกับฐานข้อมูล และฟังก์ชันจะมีการคืนค่ากลับออกมาเป็นตัวแปรแบบ odbStatus ซึ่งจะเก็บสถานะของการติดต่อกับฐานข้อมูลในครั้งนั้น ซึ่งจะทำให้เราสามารถตรวจสอบได้ว่าการติดต่อกับฐานข้อมูลประสบความสำเร็จหรือเกิดความผิดพลาดอะไรบ้าง ซึ่งฟังก์ชันที่ใช้ในการติดต่อกับฐานข้อมูลที่ทางผู้ผลิตจัดมาให้นั้นจะมีการคืนค่านี้กลับมาทุกตัว

ส่วนฟังก์ชัน odbSesEnd() จะเป็นฟังก์ชันที่ใช้ในการยุติการทำงานกับฐานข้อมูลซึ่งควรจะมีการเรียกฟังก์ชันนี้ทุกครั้งที่จบการทำงานเพื่อป้องกันความเสียหายที่เกิดขึ้นกับข้อมูลในฐานข้อมูล

- มีส่วนฟังก์ชันที่ใช้ในการส่งโค้ดของภาษา ODQL เพื่อสั่งงานตัวจัดการระบบฐานข้อมูลนั้นจะส่งผ่านทางฟังก์ชัน odbExecODQL() ดังตัวอย่าง

```
odbrc = odbExecODQL(sh,"ts = [x.name,x,photo]
                            from Top x
                            where x.hot > 0;",0,0);
```

- สำหรับในการติดต่อกับระบบฐานข้อมูลนั้นสิ่งสำคัญที่ควรทำก่อนที่จะทำการปฏิบัติการฐานข้อมูลใดๆ ก็คือ การใช้คำสั่งที่บ่งบอกว่าจะเริ่มต้นทำงานกับข้อมูลซึ่งควรเริ่มต้นและจบท้ายด้วยลักษณะการใช้คำสั่งดังนี้

```
odbrc = odbExecODQL(sh,"Transaction.start();",0,0);

...

odbrc = odbExecODQL(sh,"Transaction.end();",0,0);
```

- ส่วนของการแลกเปลี่ยนค่าข้อมูลระหว่างตัวแปรของโปรแกรมและตัวแปรของ ODQL ในฐานข้อมูลจะสามารถทำได้ด้วยการใช้ฟังก์ชัน odbGetVar() และ odbSetVar() โดยก่อนที่ตัวแปรในโปรแกรมจะสามารถนำไปใช้รับค่าหรือส่งค่ากับตัวแปรใน ODQL โดยต้องมีการใช้งานร่วมกับฟังก์ชันอีกกลุ่มคือ odbAllocData() และ odbFreeData() มีรูปแบบดังนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
odbData data1,data2;

...

odbrc = odbExecODQL(sh,"Integer x,y;",0,0);
data1 = odbAllocData();
data2 = odbAllocData();
odbrc = odbGetVar(sh,"x",data1);

...

odbrc = odbSetVar(sh,"y",data2);
odbFreeData(data1);
odbFreeData(data2);
```

จากตัวอย่างข้างต้นจะเป็นการเก็บค่าตัวแปร x ไว้ใน data1 และ ส่งค่าใน data2 กลับไปยังตัวแปร y

- ส่วนที่ทำการเปลี่ยนข้อมูลเป็นรูปแบบข้อมูลมาตรฐานที่ใช้ในโปรแกรม ส่วนนี้จะเป็นรูปแบบที่ต้องใช้ฟังก์ชันที่แตกต่างกันไป ขึ้นอยู่กับว่าต้องการเปลี่ยนค่าของตัวแปรนั้นไปเป็นแบบไหน จากตัวอย่างจะเป็นการเปลี่ยนค่าให้เป็นค่าแบบ Double

```
double      dReal;

...

if (odbGetType(realData == ODBREAL) {
        dReal = odbGetReal(realData)
}

...
```

- ในการทำงานเพื่อติดต่อกับฐานข้อมูลนั้น บ่อยครั้งที่การทำงานจะต้องมีการการเกี่ยวข้องกับกลุ่มของข้อมูลที่มีจำนวนหลายๆ ตัว (Collection) ซึ่งในทางตัวภาษาหลักหรือภาษา C นั้นไม่มีรูปแบบชนิดข้อมูลที่สามารถรองรับข้อมูลแบบนี้ได้โดยตรง ซึ่งรูปแบบวิธีการที่ใช้สำหรับการจัดการแปลงค่าข้อมูลจะใช้การสแกนข้อมูลที่อยู่ในตัวแปรนั้นแล้วทำการแปลงค่ากลับมายังภาษาหลักทีละตัว โดยในภาษาหลักอาจใช้ตัวแปรชนิด array มารองรับก็ได้ ซึ่งลักษณะการสแกนนั้นมีอยู่หลายวิธีด้วยกัน โดยสามารถดูรายละเอียดได้ใน Online documentation ซึ่งจะมีให้มาพร้อมกับตัวผลิตภัณฑ์ สำหรับในตัวอย่างที่จะแสดงในหน้าต่อไปจะเป็นการสแกนแบบไล่ทีละตัวแปร

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
odbScanId ScanId;                    ( ตัวแปรที่ใช้ในการจัดการการสแกน )
odbData Result;

...

odbrc = odbScanOpen(sh, "Set <String> StrSet;",0,0);

...

Result = odbAllocData();
odbrc = odbScanOpen(sh, "StrSet", &ScanId ,0);
for (int I = 0;I <= max;I++) {
        odbrc = odbScanNext(sh,ScanId,Result);

...

}
odbrc = odbScanClose(sh,ScanId);
odbrc = odbFreeData(Result);

...
```

- และบ่อยครั้งอีกเช่นกันที่การทำงานติดต่อกับฐานข้อมูล ทางระบบฐานข้อมูลจะมีการส่งค่ากลับมาเป็นชนิดแบบที่เป็นโครงสร้างซึ่งการอ้างอิงถึงข้อมูลในส่วนนี้จะทำได้โดยการใช้คำสั่งในรูปแบบต่อไปนี้

```
odbData field1,field2,Result;

...

field1 = odbFieldAt(Result, 0);    ( เก็บข้อมูลฟิลด์แรกของข้อมูล Result)
field2 = odbFieldAt(Result, 1);    ( เก็บข้อมูลฟิลด์ที่สองของข้อมูล Result)

...
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาคผนวก ค.

Dynamic Data Exchange (DDE)

ค.1. หลักการของ DDE

DDE (Dynamic Data Exchange) เป็นความสามารถส่วนหนึ่งของวินโดว์ ที่ให้ความสนับสนุนด้านการแลกเปลี่ยนข้อมูลและการพูดคุย (Conversation) ระหว่าง Application ต่างๆ ที่รันบน Windows ซึ่งความสามารถของ DDE นี้ได้รับการพัฒนาให้สามารถทำงานและให้บริการต่างๆสำหรับรองรับความต้องการสื่อสารระหว่างแอพพลิเคชันมากขึ้น

การแลกเปลี่ยนข้อมูลระหว่าง Application โดยผ่านทาง DDE ก็คล้ายๆกับการพูดคุยกันระหว่าง 2 บุคคล ดังนั้นขบวนการที่ DDE ใช้ในการแลกเปลี่ยนข้อมูลระหว่างกันนี้ว่า การพูดคุยระหว่าง Application ส่วน Application ทั้งสองที่มีการแลกเปลี่ยนข้อมูลก็จะถูกเรียกว่า Application ต้นทาง (Source Application) และ Application ปลายทาง (Destination Application) โดย Application ที่ทำหน้าที่กำหนดค่าเริ่มต้นสำหรับเรียกร้องการพูดคุยจะถูกเรียกว่า Application ปลายทาง ส่วน Application ที่สนองต่อการเรียกร้องนี้ก็จะถูกเรียกว่า Application ต้นทาง

ค.2. การติดต่อกับภายนอกโดยใช้ DDE ของโปรแกรมเปลือกระบบผู้เชี่ยวชาญ Knowledge Pro

สำหรับรูปแบบวิธีการที่ใช้ในการติดต่อกับภายนอกของส่วนระบบผู้เชี่ยวชาญ Knowledge Pro นั้น สามารถทำการทั้งในการส่งข้อมูลและรับข้อมูล แต่สำหรับในโครงงานนี้ การใช้งานจะใช้เพียงในส่วนที่ทำการรับข้อมูลเพียงอย่าง ในส่วนนี้จึงเป็นการอธิบายเฉพาะรูปแบบนี้เท่านั้น โดยคำสั่งที่เกี่ยวข้องและรูปแบบลำดับการใช้จะมีรายละเอียดดังนี้

คำสั่งในการเปิดการใช้บริการ DDE จะทำได้โดยการใช้คำสั่ง

dde_open(DDE_TOPIC,APPLICATION,SUBJECT)

เมื่อ *DDE_TOPIC* คือ Topic ที่เป็น Handle การติดต่อกับภายนอกโดยผ่านทางบริการ DDE

APPLICATION คือ ชื่อบริการของ Application ที่ทางโปรแกรมต้องการทำการติดต่อ

SUBJECT คือ ค่าพารามิเตอร์ที่จะทำการส่งไปเพื่อบ่งบอกถึงรูปแบบบริการที่ทางระบบผู้เชี่ยวสนใจจะติดต่อ

โดยรูปแบบในการใช้ในโปรแกรมจะใช้ในการประกาศรูปแบบของ Topic เพื่อให้เป็น Topic ที่ใช้ในการอ้างอิงเพื่อทำการติดต่อส่งและรับข้อมูลดังตัวอย่าง

DDEComm is dde_open(CommTopic, 'SERVER', 'Request')

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จะเป็นการเปิดหัวข้อ Topic ที่ใช้ในการขอหรือให้บริการจากโปรแกรมที่ใช้ชื่อในการให้บริการหรือรับบริการ *'SERVER'* ในหัวข้อของ *'Request'*

จากนั้นในการขอบริการจากฐานข้อมูลจะใช้คำสั่ง

dde_request(DDE_HANDLE,REQUEST,FORMAT)

เมื่อ *DDE_HANDLE* คือ Topic ที่เป็น Channel Handle ที่ได้มาจากคำสั่ง dde_open

REQUEST คือ รูปแบบข้อมูลที่จะเป็นการร้องขอที่จะส่งออกไป

FORMAT คือ ค่าพารามิเตอร์ที่จะทำการส่งไปเพื่อบ่งบอกถึงรูปข้อมูลที่ทางระบบผู้เชี่ยวชาญจะรับกลับมาซึ่งหากไม่ได้กำหนดไว้ก็จะเป็น text

ซึ่งหากใช้ร่วมกับคำสั่งตัวอย่างข้างต้นที่เป็นการประกาศ Topic ที่ใช้ในการติดต่อแล้วก็จะได้เป็น

dde_request(?DDEComm,'Request message')

ก็จะเป็นการส่งข้อความ *'Request message'* ไปยังโปรแกรมที่เราทำการเปิดการติดต่อไปแล้วข้างต้นโดยบอกว่าผลลัพธ์ที่จะส่งกลับมาจะเป็นรูปแบบ Text

ในส่วนสุดท้ายคือส่วนที่จะทำการรับข้อมูลผลลัพธ์กลับมายังฝั่งของระบบผู้เชี่ยวชาญ ซึ่งจะถูกนำมาเก็บไว้ใน Topic ที่เป็นพารามิเตอร์ในคำสั่ง *dde_open* ซึ่ง Topic ที่ทำการเก็บคำตอบไว้นี้ จะเป็น Topic ที่ประกอบด้วยค่าพารามิเตอร์ 3 ตัว ดังตัวอย่าง

DDE_TOPIC(INFORMATION,DDE_EVENT,DDE_HANDLE)

ซึ่งค่าพารามิเตอร์แต่ละตัวมีความหมายดังนี้

info จะเป็น list ประกอบด้วยค่าสามส่วนคือดังนี้

[DATA,REQUEST,FORMAT]

ในส่วน *DATA* จะเก็บค่าที่เป็นค่าข้อมูลที่ส่งกลับมาจริงๆ ส่วนในส่วน *REQUEST* และส่วน *FORMAT* นั้นจะเป็น ข้อความที่เราส่งไปขอข้อมูลนั้นและรูปแบบข้อมูลที่จะส่งกลับมา

ส่วน *DDE_EVENT,DDE_HANDLE* จะเป็นค่าพารามิเตอร์ที่ใช้ควบคุมการส่งข้อมูลและบ่งบอกความถูกต้องของการส่งข้อมูล

และเมื่อทำการส่งข้อมูลต่างๆ เสร็จสิ้นแล้วและต้องการปิดบริการที่เปิดขึ้นมา ก็ทำได้โดยการใช้คำสั่ง

DDE_CLOSE(DDE_HANDLE)

เมื่อ *DDE_HANDLE* คือ Topic ที่ใช้ในการ Handle การติดต่อที่เปิดขึ้นโดยคำสั่ง *dde_open*

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากนี้ไปเป็นจะเป็นการนำลำดับคำสั่งทั้งหมดมาใช้เพื่อทำการเรียกข้อมูลจากบริการภายนอกในรอบ 1 ครั้ง ซึ่งจะมีลำดับการใช้คำสั่งตามตัวอย่าง

```
DDEComm is dde_open(CommTopic, 'Server', 'Request').
dde_request(?DDEComm, 'Request Message').
wait().

...

topic CommTopic (info, event, handle).
        ...
        dde_close(Comm1).
        continue ().
end.
```

ค.3. การสร้างApplication ที่เป็น DDE Server ด้วย C++

การสร้าง DDE Server ในโครงงานนี้ใช้ภาษา C++ และใช้ตัวคอมไพเลอร์ของไมโครซอฟท์ คือ Microsoft Visual C++ เวอร์ชัน 4.0 เนื่องจากทางไมโครซอฟท์ได้เตรียมไลบรารีที่ช่วยในการเขียนโปรแกรมที่เกี่ยวข้องได้ง่ายขึ้นชื่อ DDE Management Library (DDEML) ซึ่ง DDEML นี้เป็นเซ็ตของฟังก์ชันที่ทำขึ้นมาเพื่อใช้เชื่อมต่อระหว่างแอพพลิเคชันผ่านทาง DDE

แนวทางในการสร้าง DDE Server ที่จะแสดงให้เห็นในส่วนนี้จะเป็นขั้นตอนที่ยกเอามาจาก [Stanfield,Arvesen 1996] ซึ่งจะมีรายละเอียดการสร้าง คือ สร้าง Class ของ Object ที่สนับสนุน DDE ขึ้นมาก่อนและให้ชื่อว่า CDDEObj จากนั้นก็ทำตามขั้นตอนดังต่อไปนี้

สร้าง Project ใหม่ขึ้นมาโดยใช้ MFC APPWizard (exe)

ให้สร้าง Project เป็น Dialog-Based Version

สร้าง Class ต่อไปนี้:

Application: CDDEApp ใน DDE.h และ DDE.cpp

Dialog: CDDEDlg ใน DDEDlg.h และ DDEDlg.cpp

คุณสมบัติที่ใช้:

- About box
- 3D Controls
- MFC40.DDL

ทำการประกาศ Class CDDEObj:

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ทำการสร้างแฟมข้อมลใหม่ชื่อ DDEObj.h แล้วทำการประกาศ Class CDDEObj ดังนี้

```
#include <ddeml.h>
class CDDEObj
{
public:
        static  HDDEDATA CALLBACK EPORT DdeCallback(UINT iType,
UINT iFmt,
                HCONV hCconv,
                HSZ hsz1, HSZ hsz2,
                HDDEDATA hData,
                DWORD dwData1,
                DWORD dwData2);
        static  CDDEObj* fakeThis;
        // สร้างการติดต่อกับ DDE
        CDDEObj();
private:
        // ประกาศตัวแปรที่เก็บค่า ID ของ Instance DDE
        DWORD idInst;
        // ประกาศชื่อของ Service
        Cstring AppName;
};
```

การกำหนดค่าใน Class CDDEObj

สร้างแฟ้มข้อมูลใหม่ขึ้นมาชื่อ DDEObj.cpp โดยจะนำโค้ดบางส่วนมาพิจารณา

```
HDDEDATA CALLBACK EXPORT CDDEObj::DdeCallback
        (UINT iType, UINT iFmt,
        HCONV       hCconv, HSZ hsz1, HSZ hsz2, HDDEDATA hData,
        DWORD       dwData1, DWORD dwData2)
{
// Topic = hsz1
// Item = hsz2
        char szBuffer[32];
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
switch(iType)        {
        // ทำการจัดการ 'connect' transaction
        case XTYP_CONNECT:
        // ทำการรับชื่อของ Application
        DdeQueryString(fakeThis->idInst, hsz2,
                SzBuffer, sizeof(szBuffer), 0);
        // ถ้าชื่อของ Application ไม่ได้เป็นชื่อของ Server ตัวนี้ จะ return FALSE
        If (fakeThis->AppName != szBuffer)
                Return FALSE;
        // รับชื่อจาก Topic
        DdeQueryString(fakeThis->idInst, hsz1,
                SzBuffer, sizeof (szBuffer), 0);
        //ทำการเปรียบเทียบค่าที่ได้ว่าตรงกับบริการของโปรแกรมหรือไม่ ถ้าไม่ก็จะส่งค่า
        //เท็จ
                        Return FALSE;

Return (HDDEDATA) TRUE;
        Break;
        Case XTYP_REQUEST:
        // รับชื่อจาก Topic
        DdeQueryString(fakeThis->idInst, hsz1,
                SzBuffer, sizeof(szBuffer), 0);
        If (strcmp(szBuffer, "TopicName") = = 0)
        {
                // รับชื่อจาก Item
                DdeQueryString(fakeThis->idInst, hsz2,
                        SzBuffer, sizeof(szBuffer), 0);
                ...
                // return ค่ากลับไปยังโปรแกรมที่ทำการขอบริการ
                Return DdeCreateDataHandle(fakeThis->idInst,
                        (LPBYTE) ค่าที่จะส่งกลับ,
                        ความยาวของค่าที่จะส่งกลับ,
                        0, hsz2, CF_TEXT, 0);
        }
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ประกาศ Object CDDEObj

ทำการเพิ่มโค้ดข้างล่างนี้ในบรรทัดเริ่มต้นของไฟล์ DDE.CPP

#include "ddeobj.h"

แล้วทำการประกาศออบเจกต์ CDDEObj ดังนี้ในส่วนของ Class WinMain

CDDEObj theDDEObj

รวมไลบรารี DDEML เข้าไปใน Project

ปกติแล้วไลบรารีของ DDE ไม่ได้รวมอยู่ใน linker เมื่อเราใช้ App Studio ดังนั้นจึงต้องเปลี่ยน linker ให้มีไลบรารี DDEML.LIB อยู่ด้วย

ทำการรัน Application

เมื่อรัน Application ตัวนี้จะเป็น DDE Server ที่ใช้ในการให้บริการแก่ DDE Client

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาคผนวก ง.

Source Code

ง.1. ส่วนของระบบผู้เชี่ยวชาญ

จะเป็นโปรแกรมที่เขียนขึ้นบนโปรแกรมเปลือกระบบผู้เชี่ยวชาญ Knowledge Pro คือแฟ้มข้อมูลที่ชื่อ Project.kb

```
if NOT(exists(w1)) then w1 is window().

text ('#eEnter your student ID').
TextBox is edit_line('Your ID here',,10,3,13).
set_focus(?TextBox).
button('Begin',ChkUser,18,6).

topic ChkUser.
 StdID is get_text(?TextBox).

 if ?Valid_User is yes
  then ?FindAdvisor
  else ?Show_Error.

 topic Show_Error.
  text('#eYou are not valid user.').
  ?EndProcess.
 end. (*Show Error*)

 topic Valid_User.
  set_number_of_values(Answer,1).

  if ?StdCESenior is yes
   then Valid_User is yes
   else Valid_User is no.

  topic StdCESenior.
   CheckCESenior is dde_open(CESeniorTopic,DBCall,GETS).
   RMsg is concat('Student-isDeptYear("',?StdID,'","CE","Senior")').
   dde_request(?CheckCESenior, ?RMsg).
   wait().

   topic CESeniorTopic (CESeniorInfo, CESeniorEvent, CESeniorHandle).
    StdCESenior is first(?CESeniorinfo).
    dde_close(CheckCESenior).
    continue().
   end. (*CESeniorTopic*)
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
   end. (*StdCESenior*)

  end. (*Valid_User*)

end. (*ChkUser*)

topic FindAdvisor.
 set_number_of_values(Answer,1).
 Result is  concat('#eYour senior project"s advisor should be Ajarn ',?Advisor).
 text(?Result).
 ?EndProcess.

 topic Advisor.
  set_number_of_values(Answer,1).

  if ?SelectedArticle is 'Hardware'
  and ?HardWorking is yes
    then Advisor is Apinetr.

  if ?SelectedArticle is 'Hardware'
  and ?HardWorking is no
    then Advisor is Somsak.

  if ?SelectedArticle is 'Network system programming'
  and ?HardWorking is yes
    then Advisor is Apinetr.

  if ?SelectedArticle is 'Network system programming'
  and ?HardWorking is no
    then Advisor is Chutimet.

  if ?SelectedArticle is 'Web Applications'
    then Advisor is Navaporn.

  if ?SelectedArticle is 'Network security'
    then Advisor is Akaradet.

  if ?SelectedArticle is 'Programming'
  and ?GoodStatus is yes
    then Advisor is Worawat.

  if ?SelectedArticle is 'Programming'
  and ?GoodStatus is no
    then Advisor is Krittawan.

  if ?SelectedArticle is 'Databases'
  and ?GoodStatus is yes
    then Advisor is Suphamit.
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
if ?SelectedArticle is 'Databases'
and ?GoodStatus is no
  then Advisor is Bundit.

if ?SelectedArticle is 'Logic programming'
  then Advisor is Visit.

if ?SelectedArticle is 'Image processing'
and ?SelfMotivated is yes
  then Advisor is Chom.

if ?SelectedArticle is 'Image processing'
and ?SelfMotivated is no
  then Advisor is Boonthee.

if ?SelectedArticle is 'Speech recognition'
  then Advisor is Somkiat.

if ?SelectedArticle is 'IS development'
and ?GoodStatus is yes
  then Advisor is Worawat.

if ?SelectedArticle is 'IS development'
and ?GoodStatus is no
  then Advisor is Bunjong.

if ?SelectedArticle is 'System architecture'
  then Advisor is Bunjong.

topic SelectedArticle.
  set_number_of_values(Answer,1).

  AllArticle is ['Hardware','Network system programming','Web Applications',
        'Network security','Programming','Databases',
        'Logic programming','Image processing','Speech recognition',
        'IS development','System architectures'].

  ask ('#eSelect your most interested topics',MostInterest,?AllArticle).
  RemainingArticles is different(?AllArticle,?MostInterest).
  ask ('#eSelect your alternative interested topics',AltInterest,?RemainingArticles).

  if ?GoodAtMostIntr is yes
    then SelectedArticle is ?MostInterest.

  if ?GoodAtMostIntr is no
  and ?GoodAtAltIntr is yes
    then SelectedArticle is ?AltInterest.
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
if ?GoodAtMostIntr is no
and ?GoodAtAltIntr is no
 then SelectedArticle is ?MostInterest.

topic GoodAtMostIntr.
 set_number_of_values(Answer,1).

 GoodAtMostIntr is no.

 if ?MostInterest is 'Hardware'
 and ?GoodAtMicroIntf is yes
  then GoodAtMostIntr is yes.

 if ?MostInterest is 'Network system programming'
 and ?GoodAtNet is yes
 and ?GoodAtDA is yes
  then GoodAtMostIntr is yes.

 if ?MostInterest is 'Web Application'
 and ?GoodAtNet is yes
  then GoodAtMostIntr is yes.

 if ?MostInterest is 'Network security'
 and ?GoodAtNet is yes
  then GoodAtMostIntr is yes.

 if ?MostInterest is 'Programming'
 and ?GoodAtDA is yes
  then GoodAtMostIntr is yes.

 if ?MostInterest is 'Databases'
 and ?GoodAtDB is yes
  then GoodAtMostIntr is yes.

 if ?MostInterest is 'Logic programming'
 and ?GoodAtAI is yes
  then GoodAtMostIntr is yes.

 if ?MostInterest is 'Image processing'
 and ?GoodAtAI is yes
  then GoodAtMostIntr is yes.

 if ?MostInterest is 'Speech recognition'
 and ?GoodAtAI is yes
  then GoodAtMostIntr is yes.
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
  if ?MostInterest is 'IS development'
  and ?GoodAtIS is yes
  and ?GoodAtDA is yes
    then GoodAtMostIntr is yes.

  if ?MostInterest is 'System architecture'
  and ?GoodAtCompOr is yes
    then GoodAtMostIntr is yes.

end. (*GoodAtMostIntr*)

topic GoodAtAltIntr.
  set_number_of_values(Answer,1).

  GoodAtAltIntr is no.

  if ?AltInterest is 'Hardware'
  and ?GoodAtMicroIntf is yes
    then GoodAtAltIntr is yes.

  if ?AltInterest is 'Network system programming'
  and ?GoodAtNet is yes
  and ?GoodAtDA is yes
    then GoodAtAltIntr is yes.

  if ?AltInterest is 'Web Application'
  and ?GoodAtNet is yes
    then GoodAtAltIntr is yes.

  if ?AltInterest is 'Network security'
  and ?GoodAtNet is yes
    then GoodAtAltIntr is yes.

  if ?AltInterest is 'Programming'
  and ?GoodAtDA is yes
    then GoodAtAltIntr is yes.

  if ?AltInterest is 'Databases'
  and ?GoodAtDB is yes
    then GoodAtAltIntr is yes.

  if ?AltInterest is 'Logic programming'
  and ?GoodAtAI is yes
    then GoodAtAltIntr is yes.

  if ?AltInterest is 'Image processing'
  and ?GoodAtAI is yes
    then GoodAtAltIntr is yes.
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
  if ?AltInterest is 'Speech recognition'
  and ?GoodAtAI is yes
    then GoodAtAltIntr is yes.

  if ?AltInterest is 'IS development'
  and ?GoodAtIS is yes
  and ?GoodAtDA is yes
    then GoodAtAltIntr is yes.

  if ?AltInterest is 'System architecture'
  and ?GoodAtCompOr is yes
    then GoodAtAltIntr is yes.

end. (*GoodAtAltIntr*)

topic GoodAtMicroIntf.
  CheckGoodAtMicroIntf is dde_open(GoodAtMicroIntfTopic,DBCall,GETS).
  RMsg is concat('Student-HasSpecial("',?StdID,'","Microprocessor Interfacing")').
  dde_request(?CheckGoodAtMicroIntf, ?RMsg).
  wait().

  topic GoodAtMicroIntfTopic (GoodAtMicroIntfInfo, GoodAtMicroIntfEvent,
GoodAtMicroIntfHandle).
    GoodAtMicroIntf is first(?GoodAtMicroIntfInfo).
    dde_close(CheckGoodAtMicroIntf).
    continue().
  end. (*GoodAtMicroIntfTopic*)

end. (*GoodAtMicroIntf*)

topic GoodAtNet.
  CheckGoodAtNet is dde_open(GoodAtNetTopic,DBCall,GETS).
  RMsg is concat('Student-HasSpecial("',?StdID,'","Computer Networks")').
  dde_request(?CheckGoodAtNet, ?RMsg).
  wait().

  topic GoodAtNetTopic (GoodAtNetInfo, GoodAtNetEvent, GoodAtNetHandle).
    GoodAtNet is first(?GoodAtNetInfo).
    dde_close(CheckGoodAtNet).
    continue().
  end. (*GoodAtNetTopic*)

end. (*GoodAtNet*)

topic GoodAtDA.
  CheckGoodAtDA is dde_open(GoodAtDATopic,DBCall,GETS).
  RMsg is concat('Student-HasSpecial("',?StdID,'","Data structure & Algorithm")').
  dde_request(?CheckGoodAtDA, ?RMsg).
  wait().
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
  topic GoodAtDATopic (GoodAtDAInfo, GoodAtDAEvent, GoodAtDAHandle).
    GoodAtDA is first(?GoodAtDAInfo).
    dde_close(CheckGoodAtDA).
    continue().
  end. (*GoodAtDATopic*)

end. (*GoodAtDA*)

topic GoodAtDB.
  CheckGoodAtDB is dde_open(GoodAtDBTopic,DBCall,GETS).
  RMsg is concat('Student-HasSpecial("',?StdID,'","Database Systems")').
  dde_request(?CheckGoodAtDB, ?RMsg).
  wait().

  topic GoodAtDBTopic (GoodAtDBInfo, GoodAtDBEvent, GoodAtDBHandle).
    GoodAtDA is first(?GoodAtDBInfo).
    dde_close(CheckGoodAtDB).
    continue().
  end. (*GoodAtDBTopic*)

end. (*GoodAtDB*)

topic GoodAtAI.
  CheckGoodAtAI is dde_open(GoodAtAITopic,DBCall,GETS).
  RMsg is concat('Student-HasSpecial("',?StdID,'","Artificial Intelligent")').
  dde_request(?CheckGoodAtAI, ?RMsg).
  wait().

  topic GoodAtAITopic (GoodAtAIInfo, GoodAtAIEvent, GoodAtAIHandle).
    GoodAtAI is first(?GoodAtAIInfo).
    dde_close(CheckGoodAtAI).
    continue().
  end. (*GoodAtAITopic*)

end. (*GoodAtAI*)

topic GoodAtIS.
  CheckGoodAtIS is dde_open(GoodAtISTopic,DBCall,GETS).
  RMsg is concat('Student-HasSpecial("',?StdID,'","Information systems Analysis &
Design")').
  dde_request(?CheckGoodAtIS, ?RMsg).
  wait().

  topic GoodAtISTopic (GoodAtISInfo, GoodAtISEvent, GoodAtISHandle).
    GoodAtIS is first(?GoodAtISInfo).
    dde_close(CheckGoodAtIS).
    continue().
  end. (*GoodAtISTopic*)

end. (*GoodAtIS*)
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    topic GoodAtCompOr.
      CheckGoodAtCompOr is dde_open(GoodAtCompOrTopic,DBCall,GETS).
      RMsg is concat('Student-HasSpecial("',?StdID,'","Computer organization &
Architecture")').
      dde_request(?CheckGoodAtCompOr, ?RMsg).
      wait().

      topic GoodAtCompOrTopic (GoodAtCompOrInfo, GoodAtCompOrEvent,
GoodAtCompOrHandle).
        GoodAtCompOr is first(?GoodAtCompOrInfo).
        dde_close(CheckGoodAtCompOr).
        continue().
      end. (*GoodAtCompOrTopic*)

    end. (*GoodAtCompOr*)

  end. (*SeletedArticle*)

  topic GoodStatus.
    set_number_of_values(Answer, 1).

    if ?FirstHonorGPA is yes
      then GoodStatus is yes.

    if ?HonorGPA is yes
      then GoodStatus is yes.

    if ?GoodGPA is yes
      then GoodStatus is no.

    if ?PoorGPA is yes
      then GoodStatus is no.

    if ?ProGPA is yes
      then GoodStatus is no.

    topic FirstHonorGPA.
      CheckFirstHonor is dde_open(FirstHonorTopic,DBCall,GETS).
      RMsg is concat('Student-inStatus("',?StdID,'","FirstHonor")').
      dde_request(?CheckFirstHonor, ?RMsg).
      wait().

      topic FirstHonorTopic (FirstHonorInfo, FirstHonorEvent, FirstHonorHandle).
        FirstHonorGPA is first(?FirstHonorInfo).
        dde_close(CheckFirstHonor).
        continue().
      end. (*FirstHonorTopic*)

    end. (*FirstHonorGPA*)
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
topic HonorGPA.
  CheckHonor is dde_open(HonorTopic,DBCall,GETS).
  RMsg is concat('Student-inStatus("',?StdID,'","Honor")').
  dde_request(?CheckHonor, ?RMsg).
  wait().

  topic HonorTopic (HonorInfo, HonorEvent, HonorHandle).
    HonorGPA is first(?HonorInfo).
    dde_close(CheckHonor).
    continue().
  end. (*HonorTopic*)

end. (*HonorGPA*)

topic GoodGPA.
  CheckGood is dde_open(GoodTopic,DBCall,GETS).
  RMsg is concat('Student-inStatus("',?StdID,'","Good")').
  dde_request(?CheckGood, ?RMsg).
  wait().

  topic GoodTopic (GoodInfo, GoodEvent, GoodHandle).
    GoodGPA is first(?GoodInfo).
    dde_close(CheckGood).
    continue().
  end. (*GoodTopic*)

end. (*GoodGPA*)

topic PoorGPA.
  CheckPoor is dde_open(PoorTopic,DBCall,GETS).
  RMsg is concat('Student-inStatus("',?StdID,'","Poor")').
  dde_request(?CheckPoor, ?RMsg).
  wait().

  topic PoorTopic (PoorInfo, PoorEvent, PoorHandle).
    PoorGPA is first(?PoorInfo).
    dde_close(CheckPoor).
    continue().
  end. (*PoorTopic*)

end. (*PoorGPA*)

topic ProGPA.
  CheckPro is dde_open(ProTopic,DBCall,GETS).
  RMsg is concat('Student-inStatus("',?StdID,'","Probation")').
  dde_request(?CheckPro, ?RMsg).
  wait().
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
      topic ProTopic (ProInfo, ProEvent, ProHandle).
       ProGPA is first(?ProInfo).
       dde_close(CheckPro).
       continue().
      end. (*ProTopic*)

    end. (*ProGPA*)

  end. (*GoodStatus*)

  topic HardWorking.
   ask ('Would you think you are hard-working person?',choice,[yes,no]).
   HardWorking is ?choice.
  end. (*HardWorking*)

  topic SelfMotivated.
   ask ('Would you think you are self-motivated person?',choice,[yes,no]).
   SelfMotivated is ?choice.
  end. (*SelfMotivated*)

 end. (*Advisor*)

end. (*FindAdvisor*)

topic EndProcess.
 button (Continue,:Next,20,4).

 topic Next.
  ask ('#eAnother advise ?',want,[Yes,No]).
  if ?want is Yes
   then reset ([ChkUser,FindAdvisor,EndProcess]) and
    do (!main)
   else close_window (?w1).
 end. (*Next*)
end. (*EndProcess*)
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ง.2. ส่วนของโปรแกรมตัวกลางที่ใช้ในการเชื่อมต่อ

เป็นโปรแกรมที่เขียนขึ้นด้วยภาษา C++ ซึ่งอยู่ในรูปของ Workspace ที่พัฒนาบน Microsoft Visual C++ 6.0 ชื่อ Bridge.dsw ประกอบด้วยแฟ้มข้อมูลต่างๆ ดังนี้

Bridge.h

```
// Bridge.h : main header file for the BRIDGE application
//

#if
!defined(AFX_BRIDGE_H__B6C02665_DFD3_11D3_AF75_000000000000__INCLUD
ED_)
#define
AFX_BRIDGE_H__B6C02665_DFD3_11D3_AF75_000000000000__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#ifndef __AFXWIN_H__
        #error include 'stdafx.h' before including this file for PCH
#endif

#include "resource.h"       // main symbols

/////////////////////////////////////////////////////////////////////////////
// CBridgeApp:
// See Bridge.cpp for the implementation of this class
//

class CBridgeApp : public CWinApp
{
public:
        CBridgeApp();

// Overrides
        // ClassWizard generated virtual function overrides
        //{{AFX_VIRTUAL(CBridgeApp)
        public:
        virtual BOOL InitInstance();
        //}}AFX_VIRTUAL
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
// Implementation

        //{{AFX_MSG(CBridgeApp)
                // NOTE - the ClassWizard will add and remove member functions here.
                //    DO NOT EDIT what you see in these blocks of generated code !
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
};


/////////////////////////////////////////////////////////////////////////////

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the
previous line.

#endif //
!defined(AFX_BRIDGE_H__B6C02665_DFD3_11D3_AF75_000000000000__INCLUD
ED_)
```

BridgeDlg.h

```
// BridgeDlg.h : header file
//

#if
!defined(AFX_BRIDGEDLG_H__B6C02667_DFD3_11D3_AF75_000000000000
__INCLUDED_)
#define
AFX_BRIDGEDLG_H__B6C02667_DFD3_11D3_AF75_000000000000__INCL
UDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

/////////////////////////////////////////////////////////////////////////////
// CBridgeDlg dialog

class CBridgeDlg : public CDialog
{
// Construction
public:
        CBridgeDlg(CWnd* pParent = NULL);  // standard constructor
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
// Dialog Data
        //{{AFX_DATA(CBridgeDlg)
        enum { IDD = IDD_BRIDGE_DIALOG };
                // NOTE: the ClassWizard will add data members here
        //}}AFX_DATA

        // ClassWizard generated virtual function overrides
        //{{AFX_VIRTUAL(CBridgeDlg)
        protected:
        virtual void DoDataExchange(CDataExchange* pDX);        // DDX/DDV
support
        //}}AFX_VIRTUAL

// Implementation
protected:
        HICON m_hIcon;

        // Generated message map functions
        //{{AFX_MSG(CBridgeDlg)
        virtual BOOL OnInitDialog();
        afx_msg void OnSysCommand(UINT nID, LPARAM lParam);
        afx_msg void OnPaint();
        afx_msg HCURSOR OnQueryDragIcon();
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
};

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the
previous line.

#endif //
!defined(AFX_BRIDGEDLG_H__B6C02667_DFD3_11D3_AF75_000000000000
__INCLUDED_)
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

BrObj.h

```
#include <ddeml.h>
#define MAXBUFFER 50

//the Declaration of Buffer class

class Buffer
{
public:
        Buffer(int Row,int Col,int Depth);
        void AddMember(int Row,int Col,int Depth,CString Result);
        bool ScanBuffer(CString SrchStr[]);
        void MakeFlat(int col);
        CString GetData(int Row,int Col,int Depth);

private:
        int LastPos;
        int CurrentPos;
        int TotalRow;
        int TotalCol;
        int TotalDepth;
        CString Store[MAXBUFFER][8][5];
};

//the Declaration of BridgeOBJ class

class BridgeOBJ
{
public:
        //Static callback member function.
        static HDDEDATA CALLBACK EXPORT DdeCallback(UINT iType,UINT iFmt,
                        HCONV hCconv,
                        HSZ hsz1,HSZ hsz2,
                        HDDEDATA hData,
                        DWORD dwData1,
                        DWORD dwData2);

        //Extra copy of the first object's 'this' pointer for use
        //by the Ddecallback static member function.
        static BridgeOBJ* fakeThis;

        //Constructor sets up fakeThis pointer,
        //Establishes DDE connection.
        BridgeOBJ();
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
private:
        //DDE instance ID
        DWORD idInst;

        //Service Name
        CString AppName;

        //Operation for making array of data  uerying
        static CString MakeArrStr(char *InMessage);

        //Operation for making flat table
        static Buffer MakeFlat(Buffer InBuffer);

        //Database access operation
        static Buffer DBAccess(char *Re  uest);

        //Database access for Multimedia file data type
        static CString DBGetObj(char *CallStmt);

};
```

Resource.h

```
//{{NO_DEPENDENCIES}}
// Microsoft Visual C++ generated include file.
// Used by BRIDGE.RC
//
#define IDR_MAINFRAME                   128
#define IDM_ABOUTBOX                    0x0010
#define IDD_ABOUTBOX                    100
#define IDS_ABOUTBOX                    101
#define IDD_BRIDGE_DIALOG               102

// Next default values for new objects
//
#ifdef APSTUDIO_INVOKED
#ifndef APSTUDIO_READONLY_SYMBOLS

#define _APS_NEXT_RESOURCE_VALUE  129
#define _APS_NEXT_CONTROL_VALUE         1000
#define _APS_NEXT_SYMED_VALUE           101
#define _APS_NEXT_COMMAND_VALUE         32771
#endif
#endif
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

StdAfx.h

```
// stdafx.h : include file for standard system include files,
//  or project specific include files that are used frequently, but
//      are changed infrequently
//

#if
!defined(AFX_STDAFX_H__B6C02669_DFD3_11D3_AF75_000000000000__INCLUD
ED_)
#define
AFX_STDAFX_H__B6C02669_DFD3_11D3_AF75_000000000000__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#define VC_EXTRALEAN            // Exclude rarely-used stuff from Windows headers

#include <string.h>
#include <afxwin.h>         // MFC core and standard components
#include <afxext.h>         // MFC extensions
#include <afxdisp.h>        // MFC Automation classes
#include <afxdtctl.h>           // MFC support for Internet Explorer 4 Common Controls
#include "codqlhdr.h"           // Header file for the accessing DB
#ifndef _AFX_NO_AFXCMN_SUPPORT
#include <afxcmn.h>                 // MFC support for Windows Common Controls
#endif // _AFX_NO_AFXCMN_SUPPORT


//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the
previous line.

#endif //
!defined(AFX_STDAFX_H__B6C02669_DFD3_11D3_AF75_000000000000__INCLUD
ED_)
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
/  bg d&j

// Bridge.cpp : Defines the class behaviors for the application.
//

#include "stdafx.h"
#include "Bridge.h"
#include "BridgeDlg.h"
#include "BrOBJ.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CBridgeApp

BEGIN_MESSAGE_MAP(CBridgeApp, CWinApp)
        //{{AFX_MSG_MAP(CBridgeApp)
                // NOTE - the ClassWizard will add and remove mapping macros here.
                //    DO NOT EDIT what you see in these blocks of generated code!
        //}}AFX_MSG
        ON_COMMAND(ID_HELP, CWinApp::OnHelp)
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CBridgeApp construction

CBridgeApp::CBridgeApp()
{
        // TODO: add construction code here,
        // Place all significant initialization in InitInstance
}

/////////////////////////////////////////////////////////////////////////////
// The one and only CBridgeApp object

CBridgeApp theApp;

/////////////////////////////////////////////////////////////////////////////
// The one and only BridgeOBJ object

BridgeOBJ theBridgeOBJ;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
//////////////////////////////////////////////////////////////////////////
// CBridgeApp initialization

BOOL CBridgeApp::InitInstance()
{
	AfxEnableControlContainer();

	// Standard initialization
	// If you are not using these features and wish to reduce the size
	//  of your final executable, you should remove from the following
	//  the specific initialization routines you do not need.

#ifdef _AFXDLL
	Enable3dControls();			// Call this when using MFC in a shared
DLL
#else
	Enable3dControlsStatic();	// Call this when linking to MFC statically
#endif

	CBridgeDlg dlg;
	m_pMainWnd = &dlg;
	int nResponse = dlg.DoModal();
	if (nResponse == IDOK)
	{
		// TODO: Place code here to handle when the dialog is
		//  dismissed with OK
	}
	else if (nResponse == IDCANCEL)
	{
		// TODO: Place code here to handle when the dialog is
		//  dismissed with Cancel
	}

	// Since the dialog has been closed, return FALSE so that we exit the
	//  application, rather than start the application's message pump.
	return FALSE;
}
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

BridgeDlg.cpp

```
// BridgeDlg.cpp : implementation file
//

#include "stdafx.h"
#include "Bridge.h"
#include "BridgeDlg.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CAboutDlg dialog used for App About

class CAboutDlg : public CDialog
{
public:
	CAboutDlg();

// Dialog Data
	//{{AFX_DATA(CAboutDlg)
	enum { IDD = IDD_ABOUTBOX };
	//}}AFX_DATA

	// ClassWizard generated virtual function overrides
	//{{AFX_VIRTUAL(CAboutDlg)
	protected:
	virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
	//}}AFX_VIRTUAL

// Implementation
protected:
	//{{AFX_MSG(CAboutDlg)
	//}}AFX_MSG
	DECLARE_MESSAGE_MAP()
};

CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
	//{{AFX_DATA_INIT(CAboutDlg)
	//}}AFX_DATA_INIT
}
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CAboutDlg)
	//}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
	//{{AFX_MSG_MAP(CAboutDlg)
		// No message handlers
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CBridgeDlg dialog

CBridgeDlg::CBridgeDlg(CWnd* pParent /*=NULL*/)
	: CDialog(CBridgeDlg::IDD, pParent)
{
	//{{AFX_DATA_INIT(CBridgeDlg)
		// NOTE: the ClassWizard will add member initialization here
	//}}AFX_DATA_INIT
	// Note that LoadIcon does not require a subsequent DestroyIcon in Win32
	m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
}

void CBridgeDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CBridgeDlg)
		// NOTE: the ClassWizard will add DDX and DDV calls here
	//}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CBridgeDlg, CDialog)
	//{{AFX_MSG_MAP(CBridgeDlg)
	ON_WM_SYSCOMMAND()
	ON_WM_PAINT()
	ON_WM_QUERYDRAGICON()
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
////////////////////////////////////////////////////////////////////////
// CBridgeDlg message handlers

BOOL CBridgeDlg::OnInitDialog()
{
	CDialog::OnInitDialog();

	// Add "About..." menu item to system menu.

	// IDM_ABOUTBOX must be in the system command range.
	ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
	ASSERT(IDM_ABOUTBOX < 0xF000);

	CMenu* pSysMenu = GetSystemMenu(FALSE);
	if (pSysMenu != NULL)
	{
		CString strAboutMenu;
		strAboutMenu.LoadString(IDS_ABOUTBOX);
		if (!strAboutMenu.IsEmpty())
		{
			pSysMenu->AppendMenu(MF_SEPARATOR);
			pSysMenu->AppendMenu(MF_STRING, IDM_ABOUTBOX,
strAboutMenu);
		}
	}

	// Set the icon for this dialog.  The framework does this automatically
	//  when the application's main window is not a dialog
	SetIcon(m_hIcon, TRUE);			// Set big icon
	SetIcon(m_hIcon, FALSE);		// Set small icon

	// TODO: Add extra initialization here

	return TRUE;  // return TRUE  unless you set the focus to a control
}

void CBridgeDlg::OnSysCommand(UINT nID, LPARAM lParam)
{
	if ((nID & 0xFFF0) == IDM_ABOUTBOX)
	{
		CAboutDlg dlgAbout;
		dlgAbout.DoModal();
	}
	else
	{
		CDialog::OnSysCommand(nID, lParam);
	}
}
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
// If you add a minimize button to your dialog, you will need the code below
//  to draw the icon.  For MFC applications using the document/view model,
//  this is automatically done for you by the framework.

void CBridgeDlg::OnPaint()
{
	if (IsIconic())
	{
		CPaintDC dc(this); // device context for painting

		SendMessage(WM_ICONERASEBKGND, (WPARAM) dc.GetSafeHdc(),
0);

		// Center icon in client rectangle
		int cxIcon = GetSystemMetrics(SM_CXICON);
		int cyIcon = GetSystemMetrics(SM_CYICON);
		CRect rect;
		GetClientRect(&rect);
		int x = (rect.Width() - cxIcon + 1) / 2;
		int y = (rect.Height() - cyIcon + 1) / 2;

		// Draw the icon
		dc.DrawIcon(x, y, m_hIcon);
	}
	else
	{
		CDialog::OnPaint();
	}
}

// The system calls this to obtain the cursor to display while the user drags
//  the minimized window.
HCURSOR CBridgeDlg::OnQueryDragIcon()
{
	return (HCURSOR) m_hIcon;
}
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

BrObj.cpp

```
#include "stdafx.h"
#include "BrOBJ.h"
#define MAXBUFFER 50
#define YES "yes"
#define NO "no"

//Variable for storing the result from the database server

Buffer ResultSet(MAXBUFFER,8,5);

//Variable for storing old calling message
CString OldCall = "";

Buffer::Buffer(int Row,int Col,int Depth)
{
        TotalRow = Row;
        TotalCol = Col;
        TotalDepth = Depth;
}

void Buffer::AddMember(int Row,int Col,int Depth,CString Result)
{
        Store[Row][Col][Depth] = Result;
        //MessageBox(NULL,Store[Row][Col][Depth],"Adding...",MB_OK);
}

CString Buffer::GetData(int Row,int Col,int Depth)
{
        if ((Row < TotalRow) &&
                (Col < TotalCol) &&
                (Depth < TotalDepth))
                return Store[Row][Col][Depth];
        else
                return "";
}
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
bool Buffer::ScanBuffer(CString SrchStr[])
{
	for (int i = 0;i < TotalRow;i++)
	{
		int j = 0;
		if (strcmp(Store[i][j][0],SrchStr[j]) == 0)
			while ((j < TotalCol) &&
				(strcmp(Store[i][j++][0],SrchStr[j]) == 0) ) {
			}
		if (j >= TotalCol){
			return TRUE;
		}
	}

	return FALSE;
}

//Define the static fakeThis object.
BridgeOBJ* BridgeOBJ::fakeThis = NULL;

//Constructor of object BridgeOBJ
BridgeOBJ::BridgeOBJ()
{
	fakeThis = this;

	//Setup DDE
	DdeInitialize(&idInst,DdeCallback,APPCLASS_STANDARD|
		CBF_FAIL_ADVISES|
		CBF_FAIL_EXECUTES|
		CBF_FAIL_POKES|
		CBF_SKIP_REGISTRATIONS|
		CBF_SKIP_UNREGISTRATIONS,0L);

	//Register this program as a DDE server with the service name
	//DBCall.

	HSZ hszService;
	AppName = "DBCall";

	hszService = DdeCreateStringHandle(idInst,AppName,0);
	DdeNameService(idInst,hszService,NULL,DNS_REGISTER);
}
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
CString BridgeOBJ::MakeArrStr(char *InMessage)
{
        char *seps = "\"()";
        char *TempChar;
        CString ResultArr[8];
        int i;

        for (i = 0;i < 8;i++) {
                ResultArr[i] = "";
        }

        TempChar = strtok(InMessage,seps);
        TempChar = strtok(NULL,seps);

        i = 0;
        while (TempChar != "") {
                ResultArr[i++] = strtok(NULL,seps);
                TempChar = strtok(NULL,seps);
        }
        return ResultArr[0];
}

Buffer BridgeOBJ::MakeFlat(Buffer InBuffer)
{
        Buffer ConvertedBuffer(MAXBUFFER,8,5);
        Buffer TempBuffer = InBuffer;
        int i = 0;
        while (strcmp(TempBuffer.GetData(0,i,0),"") != 0) {
                int CurRow = 0;
                int j = 0;
                while (strcmp(TempBuffer.GetData(j,i,0),"") != 0) {
                        int k = 0;
                        while (strcmp(TempBuffer.GetData(j,i,k),"") != 0) {

                                for (int l = 0;l < i;l++) {

        ConvertedBuffer.AddMember(CurRow,l,0,TempBuffer.GetData(j,l,0));
                                }


        ConvertedBuffer.AddMember(CurRow,i,0,TempBuffer.GetData(j,i,k));

                                int m = i + 1;
                                while (strcmp(TempBuffer.GetData(j,m,0),"") != 0) {
                                        int n = 0;
                                        while (strcmp(TempBuffer.GetData(j,m,n),"") != 0){
                                                ConvertedBuffer.AddMember
(CurRow,m,n,TempBuffer.GetData(j,m,n));
                                                n++;
                                        }
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                        m++;
                    }

                    k++;
                    CurRow++;
                }
                j++;
            }
            i++;
            TempBuffer = ConvertedBuffer;
            ConvertedBuffer = InBuffer;
        }
        return TempBuffer;
}

//process for access the database
Buffer BridgeOBJ::DBAccess(char *Request)
{
        //Variables for Return value
        Buffer ResultBuffer(MAXBUFFER,8,5);


        //Variables neccessary for open the connection session
        odbSessH   sh;
        odbStatus  odbrc;

        //Variables for scan and store result from the database server
        odbScanId       scanId;
        odbData                 IntData,ResultMember;
        long            MajorQty,MinorQty;

        //Building Statement for execution at DB server
        CString ExeStmnt = "";
        CString MemberStr;
        ExeStmnt += "SStrSet = ";
        ExeStmnt += Request;
        ExeStmnt += ";";
        char *TempStr = ExeStmnt.GetBuffer(0);

        //Open the database connection session
        odbrc = odbSesStart(&sh, "jasmine/jasmine","","","");
        odbrc = odbExecODQL(sh, "defaultCF projectCF;", 0, 0);
        odbrc = odbExecODQL(sh, "Transaction.start();", 0, 0);

        //Execute the database request
        odbrc = odbExecODQL(sh,"List <StringSet> SStrSet;",0,0);
        odbrc = odbExecODQL(sh,TempStr,0,0);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
//Collecting data from the server to buffer
odbrc = odbExecODQL(sh,"Integer Amount;",0,0);
odbrc = odbExecODQL(sh,"Amount = SStrSet.count();",0,0);
odbrc = odbExecODQL(sh,"Iterator <StringSet> ISStr;",0,0);
odbrc = odbExecODQL(sh,"ISStr = SStrSet.createIterator();",0,0);
odbrc = odbExecODQL(sh,"ISStr.advance();",0,0);
IntData = odbAllocData();
odbrc = odbGetVar(sh, "Amount", IntData);
MajorQty = odbGetInt(IntData);
odbrc = odbExecODQL(sh,"Set <String> SStr;",0,0);

for (int i = 1;i <= MajorQty;i++)
{
        //Beginning the scan
        odbrc = odbExecODQL(sh,"SStr = ISStr.get().Retrieve();",0,0);
        odbrc = odbExecODQL(sh,"Amount = SStr.count();",0,0);
        odbrc = odbScanOpen(sh,"SStr",&scanId,0);
        IntData = odbAllocData();
        ResultMember = odbAllocData();
        odbrc = odbGetVar(sh, "Amount", IntData);
        MinorQty = odbGetInt(IntData);
        for (int j = 0; j < MinorQty; j++)
        {
                odbrc = odbScanNext(sh, scanId, ResultMember);
                MemberStr = "";
                MemberStr += odbGetStr(ResultMember);
                ResultBuffer.AddMember(0,i-1,j,MemberStr);
        }
        odbrc = odbScanClose(sh,scanId);

        //Fetch Next Elements in Result
        odbrc = odbExecODQL(sh,"ISStr.advance();",0,0);
}

odbrc = odbFreeData(IntData);
odbrc = odbFreeData(ResultMember);

//Close the database connection
odbrc = odbExecODQL(sh, "StringSet.Clear();", 0, 0);
odbrc = odbExecODQL(sh, "Transaction.end();", 0, 0);
odbSesEnd(sh);
return ResultBuffer;
}
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
//Operation for get multimedia file from database
CString BridgeOBJ::DBGetObj(char *CallStmt)
{
        //Variables neccessary for open the connection session
        odbSessH   sh;
        odbStatus  odbrc;

        odbData ObjFile,ObjName;
        char *FileName;

        //Open the database connection session
        odbrc = odbSesStart(&sh, "jasmine/jasmine","","","");
        odbrc = odbExecODQL(sh, "defaultCF projectCF;", 0, 0);
        odbrc = odbExecODQL(sh, "Transaction.start();", 0, 0);

        odbrc = odbExecODQL(sh, "MMResult ObjData;",0,0);
        ObjFile = odbAllocData();
        ObjName = odbAllocData();
        odbrc = odbExecODQL(sh, CallStmt,0,0);
        odbrc = odbExecODQL(sh, "mediaCF::PaintBrush File;",0,0);
        odbrc = odbExecODQL(sh, "File = ObjData.GetMMDat();",0,0);
        odbrc = odbExecODQL(sh, "String Name;",0,0);
        odbrc = odbExecODQL(sh, "Name = ObjData.GetName();",0,0);

        odbrc = odbGetVar(sh, "File" ,ObjFile);
        odbrc = odbGetVar(sh, "Name" ,ObjName);
        FileName = odbGetStr(ObjName);

        odbrc = odbGetMMDataToFile(sh, ObjFile , FileName);

        odbrc = odbFreeData(ObjFile);
        odbrc = odbFreeData(ObjName);
        //Close the connection
        odbrc = odbExecODQL(sh, "Transaction.end();", 0, 0);
        odbSesEnd(sh);
        return FileName;

}
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
//Static callback member function.
HDDEDATA CALLBACK EXPORT BridgeOBJ::DdeCallback(UINT iType, UINT iFmt,
                    HCONV hConv,HSZ hsz1,HSZ hsz2,HDDEDATA hData,
                    DWORD dwData1,DWORD dwData2)
{
        //Topic = hsz1;
        //Item = hsz2;

        char szBuffer[64];
        char* RequestedClass = "";
        char* RequestedMethod = "";
        char* CompOp = "";                          //operator for comparison
        char* CompVal = "";                         //value for comparison
        char* seps = "-\t";                         //characters for the
separation
        CString MessageCall = "";

        switch(iType) {
                //Handle the 'connect' transaction.
                case XTYP_CONNECT:
                        //Get the application name.
                        DdeQueryString(fakeThis -> idInst,hsz2,
                                szBuffer,sizeof(szBuffer),0);

                        //If the application name is not supported by this server
                        //return FALSE.
                        if (fakeThis -> AppName != szBuffer)
                                return FALSE;

                        //Get the topic name.
                        DdeQueryString(fakeThis -> idInst,hsz1,
                                szBuffer, sizeof(szBuffer),0);

                        //If the topic is not either 'EXIST' or 'GET',
                        //return with an error.
                        if (strcmp(szBuffer,"GETS"))
                                        return FALSE;

                        //Return TRUE to indicate a successful connection.
                        return (HDDEDATA) TRUE;
                        break;

                //Process the 'request' transaction type.
                case XTYP_REQUEST:
                        //Get the topic name.
                        DdeQueryString(fakeThis -> idInst,hsz1,
                                szBuffer,sizeof(szBuffer),0);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
//Is the topic "GETS"?
if (strcmp(szBuffer,"GETS") == 0)
{
        //Get the item name.
        DdeQueryString(fakeThis -> idInst,hsz2,
                szBuffer, sizeof(szBuffer),0);

        CString Instr = szBuffer;

        //Separate Requested item into tokens
        RequestedClass = strtok(szBuffer,seps);
        RequestedMethod = strtok(NULL,seps);

        //Building Calling Message
        MessageCall += RequestedClass;
        MessageCall += ".";
        MessageCall += RequestedMethod;

        CString QueryStream[8];
        char *sep = "\"(),";
        char *TempChar;
        CString ResultArr[8];
        int k;

        for (k = 0;k < 8;k++) {
                ResultArr[k] = "";
        }

        char *InMessage = Instr.GetBuffer(0);

        CString DBRQ = strtok(InMessage,sep);
        TempChar = strtok(NULL,sep);

        k = 0;
        while (k < 8) {
                ResultArr[k++] = TempChar;
                TempChar = strtok(NULL,sep);
        }


        CString Result = "";
        char *Request = "";
        //Compare current calling message with the old one
        if (strcmp(OldCall,(LPCTSTR) DBRQ) == 0)
        {
                bool Found = ResultSet.ScanBuffer(ResultArr);
                if (Found)
                        Result += "yes";
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        else {
                Request = MessageCall.GetBuffer(0);
                ResultSet = DBAccess(Request);

                Buffer TempBuff(MAXBUFFER,8,5);
                TempBuff = MakeFlat(ResultSet);
                ResultSet = TempBuff;

                Found = ResultSet.ScanBuffer(ResultArr);
                if (Found)
                        Result += "yes";
                else
                        Result += "no";
        }
        //Retreive old data from the buffer
        //Result += (LPCSTR) ResultSet.GetData();
}
else    //Send the message to the database server
{
        //copy current message to old message
        OldCall = "";
        OldCall += DBRQ;

        //Access the database operation
        Request = MessageCall.GetBuffer(0);
        ResultSet = DBAccess(Request);

        Buffer TempBuff(MAXBUFFER,8,5);
        TempBuff = MakeFlat(ResultSet);
        ResultSet = TempBuff;

        bool Found = ResultSet.ScanBuffer(ResultArr);
        if (Found)
                Result += "yes";
        else
                Result += "no";
}

//convert type of calling message to common string
const char* ReturnedMessage = Result;

//Return result to the application.
return DdeCreateDataHandle(fakeThis -> idInst,
        (LPBYTE) ReturnedMessage,
        strlen(ReturnedMessage)+1,
        0,hsz2,CF_TEXT,0);
}
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                    break;
        } //end switch
        return NULL;
}
```

stdafx.cpp

```
// stdafx.cpp : source file that includes just the standard includes
//      Bridge.pch will be the pre-compiled header
//      stdafx.obj will contain the pre-compiled type information

#include "stdafx.h"
```

ง.3. ส่วนของระบบฐานข้อมูล

ประกอบด้วย Class ต่างๆ และ Method ดังนี้

Class Lecturer

Method Dept()

```
defineProcedure String projectCF::Lecturer::instance:Dept()
language : "c++"

{

 $defaultCF 'projectCF';
 $Set <String> Sp;
 $Set <Subject> SSubj;
 $SSubj = self.Teach;
 $Sp = SSubj.Name
            from SSubj
            where SSubj.Type == "RestrictedSubj";
 $return(Sp);

};
```

Method Specialist()

```
defineProcedure Bag<String> projectCF::Lecturer::instance:Specialist()
language :  "c++"

{

 $defaultCF 'projectCF';
 $Set <String> Sp;
 $Set <Subject> SSubj;
 $SSubj = self.Teach;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
$Sp = SSubj.Name
        from SSubj
        where SSubj.Type == "RestrictedSubj";
 $return(Sp);

};
```

Class Student

Method Dept()

```
defineProcedure String projectCF::Student::instance:Dept()
language:  "c++"
{

 $defaultCF 'projectCF';
 $Set <RegsSubj> SetRegs;
 $SetRegs = self.Registered;
 $Bag <String> BagStr;
 $BagStr = SetRegs.Subj.SubjectOf from SetRegs
                    where SetRegs.Subj.Type == "Lab & Project";
 $Iterator <String> IStr;
 $IStr = BagStr.createIterator();
 $IStr.advance();
 $String D;
 $D = IStr.get();
  $return(D);

};
```

Method GPA()

```
defineProcedure Real ProjectCF::Student::instance:GPA()
language :  "c++"

{

 $defaultCF 'projectCF';
 $Real GPAC;
 $Integer TotalCredit;
 $GPAC = 0;
 $TotalCredit = 0;
 $Bag <RegsSubj> BR;
 $RegsSubj R;
 $BR = self.Registered;
 $scan (BR,R) {
        $GPAC = GPAC + (R.Result * R.Subj.Credit);
        $TotalCredit = TotalCredit + R.Subj.Credit;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
};
$Real ReturnVal;
$ReturnVal = GPAC / TotalCredit;
$return (ReturnVal);

};
```

Method HasSpecial (StdID,Subj)

```
defineProcedure List<StringSet> projectCF::Student::class:
HasSpecial(String StdID,String Subj)
Language: "c++"

{

$defaultCF 'projectCF';
$Set <Student> S1;
$Set <Student> S2;
$Set <Student> S3;
$S1 = Student from Student where Student.ID == StdID;
$S2 = Student from Student where Student.Specialist().hasElement(Subj) == TRUE;
$S3 = S1.intersect(S2);
$List <String> LS1;
$List <String> LS2;
$List <String> LS3;
$List <StringSet> LSS1;
$List <StringSet> LSS2;
$LS1 = S3.ID from S3;
$Iterator <Student> IS;
$IS = S3.createIterator();
$IS.advance();
$Student Std;
$Std = IS.get();
$LS1 = List{};
$LS2 = LS1.add(Std.ID);
$LS3 = Std.Specialist();
$StringSet SS1;
$StringSet SS2;
$SS1 = StringSet.Create();
$SS1.AddSetElement(LS2);
$SS2 = StringSet.Create();
$SS2.AddSetElement(LS3);
$LSS1 = List{};
$LSS2 = LSS1.add(SS1);
$LSS1 = LSS2.add(SS2);
$return(LSS1);

};
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Method inStatus(StdID,Stat)

```
defineProcedure List<StringSet> projectCF::Student::class:
inStatus(String StdID,String Stat)
language : "c++"

{

$defaultCF 'projectCF';
$Set <Student> S1;
$Set <Student> S2;
$Set <Student> S3;
$S1 = Student from Student where Student.ID == StdID;
$S2 = Student from Student where Student.Status() == Stat;
$S3 = S1.intersect(S2);
$List <String> LS1;
$List <String> LS2;
$List <StringSet> LSS1;
$List <StringSet> LSS2;
$LS1 = S3.ID from S3;
$LS2 = S3.Status() from S3;
$StringSet SS1;
$StringSet SS2;
$SS1 = StringSet.Create();
$SS1.AddSetElement(LS1);
$SS2 = StringSet.Create();
$SS2.AddSetElement(LS2);
$LSS1 = List{};
$LSS2 = LSS1.add(SS1);
$LSS1 = LSS2.add(SS2);
$return(LSS1);

};
```

Method isDeptYear(StdID,Department,Yr)

```
defineProcedure List<StringSet> projectCF::Student::class:
isDeptYear(String StdID,String Department,String Yr)
language : "c++"

{

$defaultCF 'projectCF';
$Set <Student> S1;
$Set <Student> S2;
$Set <Student> S3;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
$Set <Student> S4;
$Set <Student> S5;
$S1 = Student from Student where Student.ID == StdID;
$S2 = Student from Student where Student.Dept() == Department;
$S3 = Student from Student where Student.Year() == Yr;
$S4 = S1.intersect(S2);
$S5 = S4.intersect(S3);
$List <String> LS1;
$List <String> LS2;
$List <String> LS3;
$List <StringSet> LSS1;
$List <StringSet> LSS2;
$List <StringSet> LSS3;
$LS1 = S5.ID from S5;
$LS2 = S5.Dept() from S5;
$LS3 = S5.Year() from S5;
$StringSet SS1;
$StringSet SS2;
$StringSet SS3;
$SS1 = StringSet.Create();
$SS1.AddSetElement(LS1);
$SS2 = StringSet.Create();
$SS2.AddSetElement(LS2);
$SS3 = StringSet.Create();
$SS3.AddSetElement(LS3);
$LSS1 = List{};
$LSS2 = LSS1.add(SS1);
$LSS1 = LSS2.add(SS2);
$LSS2 = LSS1.add(SS3);
 $return(LSS2);

};
```

Method Specialist()

```
defineProcedure Set <String> projectCF::Student::instance:Specialist()
language :  "c++"

{

$defaultCF 'projectCF';
$Set <String> Sp;
$Set <RegsSubj> SRS;
$String D;
$D = self.Dept();
$SRS = self.Registered;
$Sp = SRS.Subj.Name
            from SRS
            where SRS.GetResult() == "A"
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                and SRS.Subj.SubjectOf == D
                and SRS.Subj.Type != "Lab & Project";
  $return (Sp);

};
```

Method Status()

```
defineProcedure String projectCF::Student::instance:Status()
language :  "c++"
{

 $defaultCF 'projectCF';
 $Real R;
 $R = self.GPA();
 $if (R >= 3.50) {
        $return ("FirstHonor");
 };
 $if (R >= 3.25) {
        $return ("Honor");
 };
 $if (R >= 2.75) {
        $return ("Good");
 };
 $if (R >= 2.00) {
        $return ("Poor");
 };
  $return("Probation");

};
```

Method Year()

```
defineProcedure String projectCF::Student::instance:Year()
language: "c++"

{

 $defaultCF 'projectCF';
 $Bag <Integer> BInt;
 $Set <RegsSubj> SRS;
 $SRS = self.Registered;
 $BInt = SRS.RegsYear from SRS;
 $Integer MinYear;
 $Integer MaxYear;
 $Integer YrsStudy;
 $MinYear = BInt.min();
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
$MaxYear = BInt.max();
$Bag <Integer> BSem;
$BSem = SRS.Semester from SRS where SRS.RegsYear == MaxYear;
$Integer Sem;
$Sem = BSem.max();
$YrsStudy = MaxYear - MinYear + Sem;
$if (YrsStudy == 4) {
        $return ("Senior");
};
$if (YrsStudy == 3) {
        $return ("Junior");
};
$if (YrsStudy == 2) {
        $return ("Sophomore");
};
 $return("Freshy");

};
```

Class StringSet

Method AddSetElement(DataSet)

```
defineProcedure Void projectCF::StringSet::instance:
AddSetElement(Set <String> DataSet)
language: "c++"

{

$defaultCF 'projectCF';
$List <String> StrSet;
$StrSet = self.Elements;
$self.Elements = StrSet.append(DataSet);
$return;

};
```

Method Clear()

```
defineProcedure Void projectCF::StringSet::instance:Clear()
language : "c++"

{

$defaultCF 'projectCF';
$StringSet S;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
$List <StringSet> SS;
$SS = StringSet from StringSet;
$scan (SS,S) {
        $S.delete();
};
 $return;

};
```

Method Create()

```
defineProcedure Void projectCF::StringSet::class:Create()
language: "c++"

{

 $defaultCF 'projectCF';
 $StringSet StrSet;
 $StrSet = StringSet.new(Elements := Set {});
  $return(StrSet);

};
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บรรณานุกรม

[ธนา 2538] ธนา หงษ์สุวรรณ, *ระบบฐานข้อมูลอนุมานที่ใช้แบบจำลองระดับแนวคิด*, วิทยานิพนธ์วิศวกรรมศาสตรมหาบัณฑิต สาขาวิศวกรรมไฟฟ้า, บัณฑิตวิทยาลัย สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง, 2538.

[วิลาศ, บุญเจริญ 2535] วิลาศ วูวงศ์, บุญเจริญ ศิริเนาวกุล, *ระบบผู้เชี่ยวชาญ*, ศูนย์เทคโนโลยีอิเล็กทรอนิกส์และคอมพิวเตอร์แห่งชาติ, 2535.

[วิทยา 2541] วิทยา วัชระวิทยากุล (เรียบเรียง), Kernighan, B.W., Ritchie, D.M., *ภาษาและการโปรแกรม C*, บริษัท ซีเอ็ดยูเคชั่น จำกัด (มหาชน), 2541.

[Abawajy,Shepherd 1994] Abawajy, J., Shepherd, M., *Framework for the Design of Coupled Knowledge/Data Base Medical Information Systems*, Seventh Annual IEEE Symposium on Computer-Based Medical System, Proceeding on, 1994, pp 1-5.

[Ceri,Ramakrishnan 1996] Ceri, S., Ramakrishnan, R. , *Rule in Database Systems*, ACM Computing Surveys, Vol. 28, No.1, March 1996, pp 109-111.

[Czejdo et al. 1993] Czejdo, B., Eick, C.F., Taylor, M., *Integrating Sets, Rules and Data in an Object-Oriented Environment*, IEEE Expert (Intelligent system & their application), Vol 8, No. 1, February 1993, pp 59- 66.

[Date 1995] Date,C.J., *An Introduction to Database Sysetms* , 6th edition, Addison-Wesley , 1995.

[Ioannidis et al 1989] Ioannidis, Y.E., Chen, J., Friedman, M.A., Tsangaris, M.M., *BERMUDA-An Architecture Perspective on Interfacing Prolog to a database Machine*, Expert database systems, the Second International Conference, Proceeding from, Benjamin/Cumming, 1989, pp 229-255.

[Kandzia, Schlepphorst 1997] Kandzia, P.-Th., Schlepphorst, C., *Florid – A Prototype for F-logic*, 12th German Workshop on Logic Programming (WLP'97), Proceeding from, 1997, pp 1-7.

[Kifer 1995] Kifer, M., *Deductive and Object Data Languages: A Quest for Integration*, Deductive and Object-Oriented Databases, 4th International Conference, Proceeding from, Published as Lecture Notes on Computer Science, Springer-Verlag, 1995.

[Liu 1999] Liu, M., *Deductive Database Languages: Problems and Solutions*, ACM Computing Surveys, Vol. 31, No. 1, March 1999, pp 27-62.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

[Nikolopoulos 1997] Nikolopoulos, C., *Expert Systems*, Marcel Dekker,Inc., 1997

[Stanfield,Arvesen] Stanfield,S,. Arvesne,R. ,*Visual C++ 4 How-to*, Waite Group press, 1996

[Steib 1996] Steib, S., *Expert Database Systems*, CS530: Information Systems and Database Design, Lecture notes on, http://informatics.wustl.edu/marrs/cs530/Sherry1/Sherry1.ps, 1996

[Stroustrup 1986] Stroustrup, B., *The C++ Programming Language*, Addison-Wesley, 1986

[Telebzadeh et al. 1995] Telebzadeh, H., Mandutianu, S., Winner, C.F., *Countrywide Loan-Underwriting Expert System*, AI Magazine, Spring 1995, AAAI, pp 51-64.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้