การเปรียบเทียบระบบฐานข้อมูลเชิงวัตถุกับระบบฐานข้อมูลเชิงสัมพันธ์ในการประยุกต์ใช้กับงานระบบสารสนเทศ

A Comparison of Object-Oriented Database and Relational Database on the Information System Applications

โดย

นายจักรกฤษณ์ กลิ่นสมิทธ์

รหัส 40067045

รายงานนี้เป็นส่วนหนึ่งของวิชาโครงการพัฒนาระบบงาน

หลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ

ภาคเรียนที่ 2 ปีการศึกษา 2541

คณะเทคโนโลยีสารสนเทศ

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ชื่อหัวข้อ	การเปรียบเทียบระบบฐานข้อมูลเชิงวัตถุกับระบบฐานข้อมูลเชิงสัมพันธ์ในการประยุกต์ใช้กับงานระบบสารสนเทศ
นักศึกษา	นายจักรกฤษณ์ กลิ่นสมิทธิ์
อาจารย์ที่ปรึกษา	รศ.ดร.ศุภมิตร จิตตะยโศธร์
ระดับการศึกษา	วิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ
แขนงวิชา	วิทยาการสารสนเทศ
พ.ศ.	2541

บทคัดย่อ

ในระบบงานสารสนเทศที่มีลักษณะของข้อมูลประเภทตัวเลข ตัวอักษร ตัวอักษรที่มีขนาดความยาวมาก หรือในบางครั้งอาจรวมถึงข้อมูลประเภทมัลติมีเดีย จำเป็นต้องมีการพิจารณาความเหมาะถึงรูปแบบของระบบฐานข้อมูลที่ใช้ ซึ่งแนวทางในการศึกษาครั้งนี้จะเน้นการศึกษาถึงปัญหาหรือความไม่เหมาะสมของระบบฐานข้อมูลเชิงสัมพันธ์ในการประยุกต์ใช้กับงานระบบสารสนเทศ พร้อมทั้งแนวทางในการแก้ไขปัญหาดังกล่าวด้วยแนวคิดเชิงวัตถุที่สนับสนุนในระบบฐานข้อมูลเชิงวัตถุ นอกจากนี้ยังทำการศึกษาและพัฒนาตัวอย่างซึ่งเป็นแนวทางในการแก้ปัญหาดังกล่าวด้วยระบบฐานข้อมูลเชิงวัตถุ Jasmine

Title	A Comparison of Object-Oriented Database and Relational Database on the Information System Applications
Student	Mr. Jakkrit Klinsmith
Advisor	Assoc.Prof. Dr. Suphamit Chittayasothorn
Level of Study	Master of Science in Information Technology
Major	Information Science
Year	1998

ABSTRACT

It is essential to consider the appropriation of database system model which are used especially in the information system which its characteristics are digit, character, string, and multimedia data type. The project emphases on the problems or inappropriateness of the relational database which are applied in order to use in the information system. The guidelines of how to solve the mentioned problems by using the object-oriented concept that supports the object-oriented database are also studied. In addition, the study and development of examples of how to solve the mentioned problems by the Jasmine object-oriented database are carried out.

กิตติกรรมประกาศ

โครงงานชิ้นนี้สำเร็จลงได้ด้วยดีเพราะได้รับความกรุณา และกำลังใจที่สมาชิกทุกๆ คนในครอบครัวตลอดช่วงเวลาในการศึกษาและการพัฒนาระบบงาน ขอขอบพระคุณที่ท่านคอยช่วยเหลือในทุกๆ ด้าน

ขอขอบพระคุณสำหรับการให้คำปรึกษาอย่างดียิ่งจาก รศ.ดร. ศุภมิตร จิตตะยโศธร พร้อมทั้งเวลาที่ท่านได้สละให้ในการเตรียมงาน และการให้คำแนะนำต่างๆ

ขอขอบพระคุณสำหรับการให้คำปรึกษาและคำแนะนำจาก อ. รุ่งโรจน์ โพนคำ ที่ท่านได้สละเวลาให้ในการจัดทำรูปแบบรายงาน พร้อมทั้งคำชี้แนะต่างๆ

ขอขอบคุณท่านต่างๆ ที่มีส่วนในการช่วยงานทุกท่าน คือ เจ้าหน้าที่ของคณะเทคโนโลยีสารสนเทศที่คอยให้ข้อมูลข่าวสารต่างๆ

ขอขอบคุณ นางสาวเสาวภา สันติวิภานนท์ ที่คอยเป็นกำลังใจตลอดระยะเวลาการทำโครงงานชิ้นนี้จนกระทั่งโครงงานดังกล่าวสำเร็จลุล่วง

ขอขอบคุณ พี่ๆ เพื่อนๆ คณะเทคโนโลยีสารสนเทศที่คอยให้ความช่วยเหลือในเรื่องข้อมูลและข้อเสนอแนะต่างๆ

และขอขอบพระคุณคณะกรรมการทุกท่านที่ช่วยแก้ไขข้อผิดพลาดเพื่อให้โครงงานชิ้นนี้ถูกต้องและสมบูรณ์ยิ่งขึ้น

นายจักรกฤษณ์ กลิ่นสมิทธิ์

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญภาพ

หน้า

รูปที่

สารบัญภาพ (ต่อ)

หน้า

รูปที่

สารบัญภาพ (ต่อ)

หน้า

รูปที่

บทที่1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ปัจจุบันลักษณะงานในระบบสารสนเทศได้มีการขยายขอบเขตการใช้งานเทคโนโลยีทางด้านฐานข้อมูลในการจัดการกับข้อมูลที่มีความซับซ้อน(Complex Data Type) เพิ่มมากขึ้น เช่น ข้อมูลประเภทกราฟิก(Graphic) และข้อมูลประเภทมัลติมีเดีย(Multimedia) จึงมีการผลักดันระบบการจัดการฐานข้อมูลรูปแบบใหม่มากมาย เพื่อรองรับกับความต้องการดังกล่าว ระบบการจัดการฐานข้อมูลเชิงวัตถุ(Object-Oriented Database Management หรือOODBMS) เป็นระบบการจัดการฐานข้อมูลหนึ่งที่มีการนำเสนอเพื่อจัดการกับข้อมูลที่มีความซับซ้อนนั้น โดยมีแนวคิดในการจัดการข้อมูลแตกต่างจากระบบการจัดการฐานข้อมูลเชิงสัมพันธ์(Relational Database Management หรือ RDBMS) ที่เป็นระบบการจัดการฐานข้อมูลในรูปแบบของความสัมพันธ์(Relation) ซึ่งจะแสดงความสัมพันธ์ดังกล่าวได้ด้วย ตาราง(Table) ทั้งนี้ความต้องการในการจัดการกับข้อมูลลักษณะดังกล่าวนั้น ส่งผลให้การจัดการข้อมูลด้วยระบบการจัดการฐานข้อมูลเชิงสัมพันธ์ไม่เพียงพอและอาจจะไม่เหมาะสม เนื่องมาจากรูปแบบข้อมูล(Data Type) และตัวกระทำการ(Operator) ที่สนับสนุนในระบบการจัดการฐานข้อมูลเชิงสัมพันธ์ไม่เพียงพอกับความต้องการในงานสารสนเทศปัจจุบัน และระบบการจัดการฐานข้อมูลเชิงสัมพันธ์ยังขาดฟังก์ชัน Built-in ในการจัดการข้อมูลที่มีความซับซ้อน รวมทั้งยังขาดความสามารถในการนำเสนอ Structure นอกจากปัญหาดังกล่าวแล้วระบบการจัดการฐานข้อมูลเชิงสัมพันธ์ยังมีปัญหาในเรื่องการดูแลรักษาและการจัดการในเรื่องการควบคุมความถูกต้อง(Integrity Rule) เนื่องจากกระทำโดยโพรซีเดอร์ซึ่งเป็นคนละส่วนกับโครงสร้าง ข้อมูล และยังมีปัญหาเกี่ยวกับการจัดการ Identifier key ซึ่งเป็นการจัดการโดยผู้ใช้หรือผู้พัฒนาเอง

จากปัญหาที่กล่าวข้างต้นนั้นส่งผลให้การพัฒนาระบบงานที่มีการอ้างอิงข้อมูลในลักษณะดังกล่าวมีความยุ่งยากซับซ้อนและใช้เวลาในการพัฒนาเพิ่มมากขึ้นรวมทั้งการดูแลรักษาระบบยังทำได้ไม่สะดวกเนื่องการการควบคุมความถูกต้องเป็นคนละส่วนกับโครงสร้างข้อมูล ทำให้ผู้พัฒนาระบบหรือโปรแกรมเมอร์เป็นผู้รับภาระการจัดการข้อมูลดังกล่าวด้วยตัวเอง ทั้งนี้แนวความคิดเชิงวัตถุที่สนับสนุนในระบบการจัดการฐานข้อมูลเชิงวัตถุ เช่น Abstraction Mechanism, Complex Object และ Object Identity สามารถรองรับและแก้ไขปัญหาดังกล่าวได้ ซึ่งจะช่วยลดความยุ่งยากและเวลาที่ใช้ในการพัฒนาระบบงานของผู้พัฒนาระบบหรือโปรแกรมเมอร์

จากที่กล่าวมางานประมวลผลทรานส์แอ็กชัน(Transaction Processing) ในปัจจุบันโดยทั่วไปจะประกอบด้วยข้อมูลประเภทตัวเลข, ตัวอักษร, กลุ่มของตัวอักษรที่มีขนาดความยาวมาก (Memo) หรือในบางครั้งอาจรวมถึงข้อมูลประเภทมัลติมีเดีย(Multimedia)ด้วยนั้น สิ่งหนึ่งที่ต้องคำนึงถึงคือรูปแบบของระบบการจัดการฐานข้อมูลที่ใช้ซึ่งจะส่งผลกระทบต่อระบบงานที่ทำงานอยู่บนระบบการจัดการฐานข้อมูลดังกล่าว รวมทั้งยังส่งผลถึงความยาก-ง่ายในการดูแลรักษา (Maintenance) ระบบต่อๆ ไปอีกด้วย ดังนั้นการศึกษาถึงรูปแบบ, ลักษณะการทำงานของระบบการจัดการฐานข้อมูลจึงเป็นสิ่งที่สำคัญในการพิจารณาความเหมาะสมระหว่างระบบงานกับระบบการจัดการฐานข้อมูลที่เลือกใช้

1.2 วัตถุประสงค์ของงานวิจัย

ในโครงการพัฒนาระบบงานมีวัตถุประสงค์ของการวิจัยดังนี้

1. เพื่อศึกษาทฤษฎีและหลักการทำงานของระบบการจัดการฐานข้อมูลเชิงวัตถุกับระบบการจัดการฐานข้อมูลเชิงสัมพันธ์
2. เพื่อศึกษาถึงปัญหาของระบบการจัดการฐานข้อมูลเชิงสัมพันธ์ในปัจจุบัน
3. เพื่อศึกษาถึงแนวทางในการแก้ปัญหาของระบบการจัดการฐานข้อมูลเชิงสัมพันธ์ด้วยระบบการจัดการฐานข้อมูลเชิงวัตถุ
4. เพื่อศึกษาหาความแตกต่างและเปรียบเทียบข้อดี-ข้อเสียของการประยุกต์ใช้งานสารสนเทศบนระบบการจัดการฐานข้อมูลเชิงวัตถุกับระบบการจัดการฐานข้อมูลเชิงสัมพันธ์
5. เพื่อสร้างตัวอย่างโปรแกรมประยุกต์ในการจัดการปัญหาดังกล่าวบนระบบการจัดการฐานข้อมูลเชิงวัตถุ

1.3 ขอบเขตของงานวิจัย

ศึกษาระบบการจัดการฐานข้อมูลเชิงวัตถุและระบบการจัดการฐานข้อมูลเชิงสัมพันธ์ ในด้านต่างๆ เพื่อศึกษาถึงปัญหาหรือความไม่เหมาะสมของระบบการจัดการฐานข้อมูลเชิงสัมพันธ์ที่เกิดขึ้นกับงานระบบสารสนเทศในปัจจุบัน พร้อมทั้งแนวทางในการแก้ปัญหาด้วยระบบการจัดการฐานข้อมูลเชิงวัตถุ และสร้างตัวอย่างโปรแกรมประยุกต์เพื่อสนับสนุนแนวความคิดเชิงวัตถุในระบบการจัดการฐานข้อมูลเชิงวัตถุที่ใช้ในการจัดการปัญหาดังกล่าว รวมทั้งศึกษาหาความแตกต่างและเปรียบเทียบข้อดี-ข้อเสียของการประยุกต์ใช้งานสารสนเทศบนระบบการจัดการฐานข้อมูลเชิงวัตถุกับระบบการจัดการฐานข้อมูลเชิงสัมพันธ์

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1. ทราบถึงทฤษฎีและหลักการทำงานของระบบการจัดการฐานข้อมูลเชิงวัตถุกับระบบการจัดการฐานข้อมูลเชิงสัมพันธ์
2. ทราบถึงปัญหาของระบบการจัดการฐานข้อมูลเชิงสัมพันธ์ในปัจจุบัน
3. ทราบถึงแนวทางในการแก้ปัญหาของระบบการจัดการฐานข้อมูลเชิงสัมพันธ์ด้วยระบบการจัดการฐานข้อมูลเชิงวัตถุ
4. ทราบถึงความแตกต่างรวมทั้งข้อดี-ข้อเสียของการประยุกต์ใช้งานสารสนเทศบนระบบการจัดการฐานข้อมูลเชิงวัตถุกับระบบการจัดการฐานข้อมูลเชิงสัมพันธ์
5. สามารถพัฒนาตัวอย่างโปรแกรมประยุกต์ในการจัดการปัญหาดังกล่าวบนระบบการจัดการฐานข้อมูลเชิงวัตถุ

1.5 เนื้อหาในงานวิจัย

เนื้อหางานวิจัยในบทแรกนี้จะกล่าวถึงความเป็นมาและความสำคัญของปัญหา รวมทั้งวัตถุประสงค์ในการทำวิจัย บทที่ 2 จะกล่าวถึงระบบการจัดการฐานข้อมูลเชิงสัมพันธ์ที่ใช้ในปัจจุบันและปัญหาหรือความไม่เหมาะสมของระบบการจัดการฐานข้อมูลเชิงสัมพันธ์ที่เกิดขึ้นกับงานระบบสารสนเทศในปัจจุบัน บทที่ 3 จะกล่าวถึงแนวคิดเชิงวัตถุ บทที่ 4 จะกล่าวถึงระบบการจัดการฐานข้อมูลเชิงวัตถุซึ่งเป็นการนำเอาแนวคิดเชิงวัตถุเข้ามาประยุกต์ใช้กับระบบฐานข้อมูล รวมทั้งการเปรียบเทียบการประยุกต์ใช้งานสารสนเทศบนระบบการจัดการฐานข้อมูลเชิงวัตถุกับระบบการจัดการฐานข้อมูลเชิงสัมพันธ์ ในบทที่ 5 จะกล่าวถึงการแนะนำระบบการจัดการฐานข้อมูลเชิงวัตถุ Jasmine บทที่ 6 แนวทางในการแก้ปัญหาหรือความไม่เหมาะสมของระบบการจัดการฐานข้อมูลเชิงสัมพันธ์ที่เกิดขึ้นกับงานระบบสารสนเทศในปัจจุบันด้วยแนวคิดเชิงวัตถุในระบบการจัดการฐานข้อมูลเชิงวัตถุ พร้อมทั้งการแก้ไขปัญหาดังกล่าวด้วยระบบการจัดการฐานข้อมูลเชิงวัตถุ Jasmine และในบทที่ 7 จะกล่าวถึงตัวอย่างการพัฒนา โปรแกรมประยุกต์บน Jasmine สุดท้ายในบทที่ 8 จะเป็นบทสรุปและแนวทางในการพัฒนาระบบในอนาคต

บทที่ 2
ระบบฐานข้อมูลเชิงสัมพันธ์

ระบบฐานข้อมูลที่นิยมใช้กันนั้นมีแนวความคิดอยู่บนความสัมพันธ์(Relation) โดยมีแบบจำลองที่สนับสนุนแนวคิดดังกล่าวคือแบบจำลองเชิงสัมพันธ์ หรือรีเลชันนัลโมเดล(Relational Model) แต่ในปัจจุบันความต้องการข้อมูลในรูปแบบมัลติมีเดียหรือข้อมูลที่มีความซับซ้อน มีเพิ่มมากขึ้นตามเทคโนโลยีที่สูงขึ้น ซึ่งข้อมูลลักษณะดังกล่าวมีความยุ่งยากและไม่เหมาะสมกับการใช้แบบจำลองเชิงสัมพันธ์

โดยในบทนี้จะเริ่มด้วยการทบทวนแนวคิดและประเด็นสำคัญของระบบฐานข้อมูลเชิงสัมพันธ์ จากนั้นจะเป็นการกล่าวถึงปัญหาของระบบฐานข้อมูลเชิงสัมพันธ์ในงานระบบสารสนเทศปัจจุบัน

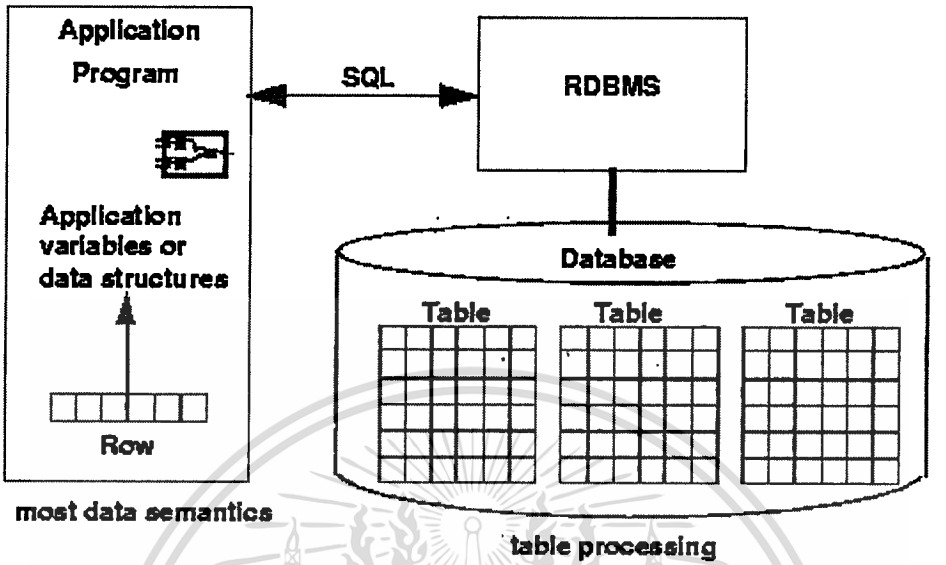
2.1 แนวคิดและประเด็นสำคัญของระบบฐานข้อมูลเชิงสัมพันธ์

แบบจำลองข้อมูลเชิงสัมพันธ์(Relational Data Model) นั้นมีแนวความคิดอยู่บนความสัมพันธ์ ซึ่งจะสนับสนุนเฉพาะโครงสร้างแบบจำลองข้อมูลเชิงสัมพันธ์เท่านั้น ในส่วนของทฤษฎีรีเลชันนัลสามารถทำความเข้าใจได้ง่าย โดยมีพื้นฐานอยู่บนรีเลชันนัลแอลจิบรา(Relational Algebra) และรีเลชันนัลแคลคูลัส(Relational Calculus) ซึ่งจะเกี่ยวข้องกับพื้นฐานทางคณิตศาสตร์ เช่น เซต, ฟังก์ชัน และ ตรรกศาสตร์

2.1.1 โครงสร้างข้อมูลแบบรีเลชันนัล(Relational Data Structure)

- โครงสร้างของรีเลชันนัล(Relational or Table Structure)

ในระบบฐานข้อมูลเชิงสัมพันธ์ข้อมูลทั้งหมดจะถูกเก็บในตาราง 2 มิติโดยตารางดังกล่าวจะถูกเรียกว่ารีเลชัน(Relations) และจะเก็บข้อมูลอยู่ในรูปแถว(Rows) ต่อกันไป ในแต่ละแถวของตารางจะประกอบด้วยฟิลด์(Fields) หรือแอตทริบิวต์(Attributes) ซึ่งต้องตรงกับประเภทของข้อมูลที่รองรับในฐานข้อมูลนั้นๆ ระบบโดยส่วนมากจะไม่สามารถเพิ่มประเภทของข้อมูลชนิดใหม่เข้าไปได้



รูปที่ 2.1 แสดง RDBMS Access

- คีย์หลักและคีย์นอก(Primary and Foreign Keys)

ในแต่ละแถวของตารางเปรียบได้กับ 1 ไอเทม หรือ 1 เรคอร์ด ซึ่งตำแหน่งของแถวอาจจะมีการเปลี่ยนแปลงในกรณีที่มีการเพิ่มหรือลบแถว ไอเทมที่ถูกเก็บในฐานข้อมูลจะไม่มีการซ้ำกันหรือมีเพียงหนึ่งเดียว(Unique) ในตาราง

ในการอ้างหรือระบุถึงไอเทมในตารางจำเป็นต้องเลือกฟิลด์หรือแอตทริบิวต์มากำหนดเป็นคีย์

ชนิดของคีย์ในตาราง

- คีย์หลักหรือ Primary Key คือคีย์ที่ไม่ซ้ำ(Unique)ในตารางความสัมพันธ์ ซึ่งทุกตารางจะต้องมีคีย์หลักดังกล่าว ซึ่งแต่ละตารางจะมีคีย์หลักเพียงคีย์เดียว และจะต้องไม่เป็น Null หรือไม่เป็นค่าว่าง
- คีย์นอกหรือ Foreign Key คือแอตทริบิวต์ หรือ Combination ที่ใช้ในการอ้างอิงข้อมูลในตารางความสัมพันธ์อื่นโดยคีย์นอกจะ Match กับคีย์หลักของตารางความสัมพันธ์ที่อ้างอิงถึง ทั้งนี้คีย์นอกอาจมีค่าเป็น Null ได้

ตัวอย่างของคีย์หลักและคีย์นอก

Invoice

Invoice number	Customer number	Date
1276	1232	3 / 23 / 1992
1289	4567	4 / 1 / 1992

จากตารางข้างต้น Invoice Number จะเป็นคีย์หลักของตาราง Invoice และ Customer Number เป็นคีย์นอกในการอ้างถึงแถวในตาราง Customer

Customers

Customer number	Name
1232	Eddy Murphy
1456	George Bush

ตาราง Invoice อาจจะประกอบด้วยแถวหลายๆ แถว และในกรณีที่มีความต้องการอ้างถึงจำนวนของสินค้า จำเป็นต้องมีการสร้างตารางเกี่ยวกับข้อมูลดังกล่าว ในที่นี้คือ ตาราง Invoice Rows โดยคีย์หลักของตารางจะประกอบด้วยฟิลด์ 2 ฟิลด์คือ Invoice Number และ Row Number

Invoice Rows

Invoice number	Row number	Quantity	Item number
1234	4567	123	1001
1234	1232	1	1032
1289	4567	15	2045

ในการอ้างถึงราคา, ชื่อสินค้า, และจำนวนสินค้าคงคลัง จะอ้างอิงผ่านตาราง Items

Items

Item Number	Name	Price	Number in stock
1001	Elephant Soup	22.34	1066
1032	Nuclear Dental Floss	10.04	157

- การแมนนิพูเลชันข้อมูล(Data Manipulation)

การเพิ่มแถวเข้าไปในตารางสามารถทำได้โดยใช้คำสั่ง INSERT ในกรณีแถวที่มีอยู่ในตารางมีการเปลี่ยนแปลงหรือทีความต้องการลบทิ้งสามารถทำได้โดยใช้คำสั่ง UPDATE และ DELETE ตามลำดับ โดยคำสั่ง SELECT จะเป็นการสอบถามข้อมูล คำสั่ง UPDATE, DELETE และ

SELECT เป็นการเลือกแถวที่มีค่าดังกล่าวและกระทำตามคำสั่งนั้นๆ กับทุกแถวในตารางที่ตรงตามเงื่อนไขดังกล่าว

ตัวอย่าง การเพิ่มแถวในตารางสามารถเขียนโค้ด(code)ได้ดังนี้

```
INSERT
INTO ITEMS (ITEM_NUMBER, NAME, PRICE, QUANTITY)
VALUE ('10036', 'oil of Broccoli', '5.23', '100')
```

ตัวอย่าง การขึ้นราคา 1 ดอลลาร์กับสินค้าที่มีราคาถูกกว่า 1 ดอลลาร์

```
UPDATE ITEMS
SET PRICE = 1 + ITEMS.PRICE
WHERE S.PRICE < 1.00
```

ตัวอย่าง การลบรายการสินค้าทั้งหมดที่มีราคาถูกกว่า 10 ดอลลาร์

```
DELETE
FROM ITEMS
WHERE P.PRICE < 10
```

ในกรณีมีการกระทำกับแถวที่อยู่ต่างตารางกันสามารถใช้การจอย(Join) ซึ่งเป็นส่วนหนึ่งของคำสั่ง SELECT โดยไม่ต้องมีการเก็บลงในอีกตารางหนึ่งโดยตรง สมมติต้องการทราบจำนวนสินค้าที่ชื่อ "Cream of Elephant Soup" สามารถเขียนโค้ดได้ดังนี้

```
SELECT INVOICE_ROWS.INVOICE_NUMBER,
        ITEMS.NAME, INVOICE_ROWS.QUANTITY
        FROM INVOICE_ROWS, ITEMS
        WHERE ITEMS.NAME = 'Cream of Elephant Soup'
```

2.2 ปัญหาของระบบการจัดการฐานข้อมูลเชิงสัมพันธ์ในงานระบบสารสนเทศปัจจุบัน

ปัจจุบันลักษณะงานในระบบสารสนเทศได้มีการขยายขอบเขตการใช้งานเทคโนโลยีทางด้านฐานข้อมูลในการจัดการกับข้อมูลที่มีความซับซ้อน เช่น ข้อมูลประเภทภาพ, เสียง หรือข้อมูลประเภทมัลติมีเดีย ซึ่งมีความต้องการใช้งานข้อมูลลักษณะนี้เพิ่มมากขึ้น ทั้งนี้ความต้องการในการ

จัดการกับข้อมูลลักษณะดังกล่าวนั้น ส่งผลให้การจัดการข้อมูลด้วยระบบการจัดการฐานข้อมูลเชิงสัมพันธ์ไม่เพียงพอและอาจจะไม่เหมาะสม เนื่องมาจาก

2.2.1 ปัญหา Data Type ที่มีใน RDBMS ไม่เพียงพอกับความต้องการ

Data Type ที่มีใน RDBMS (เช่น integer, character string เป็นต้น) นั้นมีความจำเป็นและเหมาะสมกับโปรแกรมประยุกต์ทางธุรกิจที่มีการจัดเก็บข้อมูลไม่ซับซ้อนมากนัก อย่างไรก็ตาม ใน advance application ชุดของ Data Type ดังกล่าวยังไม่เหมาะสมเพียงพอ ตัวอย่างเช่น ใน scientific application ผู้ใช้มีความต้องการใช้รูปแบบข้อมูลหรือ Data Type ในลักษณะ complex number, array และ time series สำหรับใน Business application เองก็มีความต้องการใช้ Data Type ที่มีลักษณะพิเศษเช่นกัน ตัวอย่างเช่นโปรแกรมที่คำนวณหาดอกเบี้ยของพันธบัตร ที่มีการกำหนดให้ทุกๆ เดือนมี 30 วัน ซึ่งโปรแกรมดังกล่าวมีความต้องการ DATE เป็น Data Type ในการหาผลต่างระหว่าง "April 15" – "March 15" ซึ่งจะต้องได้ค่าเป็น 30 มากกว่าที่จะได้ค่าเป็น 31 วัน(กำหนดให้ทุกเดือนมี 30 วัน)

(หมายเหตุ : รายละเอียดและตัวอย่างขออธิบายร่วมกับปัญหาที่ 2 ต่อไป)

2.2.2 ปัญหา Operator ที่สนับสนุนใน RDBMS ไม่เพียงพอกับความต้องการ

Operator (หรือ built-in Operator) ที่สนับสนุนใน RDBMS นั้นไม่เพียงพอและเหมาะสมกับความต้องการในการใช้งานกับโปรแกรมประยุกต์ในปัจจุบันหรือโปรแกรมประยุกต์ที่มีความซับซ้อน ทั้งนี้ Built-in Operator ที่มีใน RDBMS ทั่วๆ ไปแบ่งออกเป็น 3 ประเภทใหญ่ๆ คือ

1. **Calculation Operator** (หรือ Arithmetic Operator) ได้แก่ +, -, *, / ซึ่ง Operator ดังกล่าวมีความเหมาะสมกับโปรแกรมประยุกต์ทางธุรกิจที่มีการประมวลผลไม่ซับซ้อนมากนัก อย่างไรก็ตามใน advance application ชุดของOperator ดังกล่าวยังไม่เหมาะสมเพียงพอ ตัวอย่างเช่น geographic application ผู้ใช้มีความต้องการใช้รูปแบบข้อมูลในลักษณะ points, lines, line groups และ polygons เป็นรูปแบบข้อมูลพื้นฐานและมีการใช้ Operator เกี่ยวกับ intersection, distance และ containment ในการประมวลผลเป็นต้น
2. **Comparison Operator** (หรือ Relation Operator) ได้แก่ <, >, <=, >=, = และ ^= ซึ่ง Operator เหล่านี้สามารถใช้ได้ครอบคลุมกับ Data Type ที่เป็น basic Data Type (เช่น integer, floatingเป็นต้น) ใน RDBMS แต่ยังไม่เหมาะสมกับข้อมูลที่มีลักษณะซับซ้อน

เนื่องจากมีความยุ่งยากในการพิจารณาเงื่อนไขต่างๆ หลายขั้นตอนในการอ้างอิงถึงข้อมูลลักษณะดังกล่าว

3. **Logical Operator** ได้แก่ AND, OR, NOT ซึ่งใช้เชื่อมความสัมพันธ์ของเงื่อนไขระหว่างตัวแปรที่มีการใช้ comparison Operator เช่น (AGE > 10) AND (AGE < 25) เป็นต้น แต่ยังไม่เหมาะสมกับโปรแกรมประยุกต์ที่มีการเปรียบเทียบซับซ้อนมาก เนื่องจากมีความยากของการ Query ซึ่งจำเป็นต้องหาหลักเกณฑ์หรือสูตรในการคำนวณที่ยุ่งยากซับซ้อนขึ้น รวมทั้งยังทำความเข้าใจได้ยากด้วย

(หมายเหตุ : ผู้แต่หนังสือบางท่านได้แบ่ง Operator เป็น 2 ประเภทใหญ่ๆ คือ 1. Arithmetic Operator 2. Comparison Operator และ Logical Operator ทั้งนี้เนื่องจากความสัมพันธ์ของ Operator ทั้ง 2 ในการใช้งาน)

จากปัญหาที่ 2.1.1 และปัญหาที่ 2.2.2 นั้น

โปรแกรมประยุกต์ในปัจจุบันต้องทำการสมมติ(จำลอง) Data Type และ Operator เหล่านั้นขึ้นมาโดยใช้ Data Type และ Operator พื้นฐานที่มีอยู่ใน DBMS นั้นๆ ซึ่งผลที่ได้ค่อนข้างมีความยุ่งยากและซับซ้อนมาก โดยความซับซ้อนดังกล่าวจะขึ้นกับ Data Type และ Operator ที่ต้องทำการสมมติขึ้นมา เช่น ตัวอย่างความสัมพันธ์ง่ายๆ ซึ่งประกอบด้วยข้อมูลของรูปสี่เหลี่ยม(หรือกล่องหรือ BOX) 2 มิติ ถ้าในแต่ละ BOX มีการกำหนด Identifier แล้วแทน BOX ด้วยจุด Coordinate ของ 2 มุมของ BOX นั้น ดังเช่น

```
CREATE TABLE BOX
    ( ID    NUMBER(4),
      X1    FLOAT(8),
      X2    FLOAT(8),
      Y3    FLOAT(8),
      Y2    FLOAT(8) );
```

จากข้างต้นนั้นเมื่อมาพิจารณาถึงการ Query ในที่นี้ขอยกตัวอย่างแบบง่ายๆ ในการหา BOX ทั้งหมด(แบบ non-rotated) ที่มีเกิดการ Overlap หรือมีการทับกับ Unit Square ที่มี Coordinates (0,1,0,1) ซึ่งสามารถเขียนในรูปแบบของ SQL ได้ดังนี้

```
SELECT *
FROM BOX
WHERE NOT (X2 <= 0 OR X1 >= 1 OR Y2 <= 0 OR Y1 >= 1);
```

ปัญหาพื้นฐานในการแสดงดังกล่าวซึ่งก็คือ ความยากของการ Query ในการหาหลักเกณฑ์หรือสูตรในการคำนวณ และยังเข้าใจได้ยากด้วย ซึ่งจะเห็นได้จากการใช้ Query เดิม แต่มีการหา rotate BOX ด้วย ซึ่งจะต้องมีหลักเกณฑ์ในการพิจารณายุ่งยากซับซ้อนขึ้นอีก

จากปัญหาดังกล่าวข้างต้นนั้นถ้าพิจารณาอย่างง่ายๆ คือถ้า DBMS มีการเตรียม BOX ซึ่งเป็น Data Type ไว้ และมี Operator ที่เหมาะสมที่ใช้กับ Data Type นั้นไว้ การกำหนด Relation จะสามารถทำได้ดังนี้

```
CREATE TABLE BOX
        ( ID NUMBER(4),
          DESC BOX) );
```

และในการ Query นั้นสามารถเขียนได้ดังนี้

```
SELECT *
FROM BOX
WHERE DESC OVERLAPS '0, 1, 0, 1';
```

การกำหนด Data Type ขึ้นใหม่ดังกล่าวนั้นจะต้องมีการกำหนดเป็นพิเศษถึงลักษณะของ Type นั้นกับ Operation ที่ใช้ร่วมกับ Type นั้นๆ เช่น **OVERLAPS** Operator ในตัวอย่าง ซึ่งความต้องการของ Type เหล่านั้นมีความสำคัญมาก อย่างเช่นเป็น Type ที่เฉพาะเจาะจงขึ้นมาและ behavior ของ Type นั้นๆ ก็มีความซับซ้อนมาก(เช่นข้อมูลประเภทรูปภาพ) และไม่สามารถอธิบายได้ทั้งหมดโดยใช้ความสามารถทั่วๆไป หรือความสามารถพื้นฐานของ DBMS

(หมายเหตุ : การเปลี่ยนแปลงความต้องการเพื่อสนับสนุนให้มีการกำหนด Type ใหม่ๆ ของ DBMS ได้นั้นไม่ใช่ข้อจำกัด ซึ่งอาจทำได้โดยมีการเปลี่ยนแปลงโครงสร้างภายในของ DBMS เพื่อให้สนับสนุน Data Type นั้นๆ)

วิธีการหนึ่งที่ใช้ในการแก้ปัญหาดังกล่าวนั้นเป็นการกำหนดส่วนของ non-traditional Data Type ต่างๆ และรวมไว้ใน DBMS ตัวอย่างเช่น การเพิ่มเติมได้วัตถุประสงค์พิเศษต่างๆ ใน DBMS นั้นเป็นเรื่องเกี่ยวกับข้อมูลประเภท text, image และ geometric ซึ่งความยากของการเพิ่มเติมนี้เป็นไปตามแต่ละกรณีที่ต้องการกำหนดหรือระบุในโปรแกรมประยุกต์ รวมทั้งข้อจำกัดในหลักการทั่วๆ ไป ตัวอย่างเช่น ความสามารถเกี่ยวกับระยะทาง(spatial) นั้นจะใช้กับข้อมูลประเภท geographic ได้ดีที่สุดแต่จะมีข้อจำกัดเมื่อใช้ใน CAD application นอกจากนั้น *category* ของข้อมูลประเภทหนึ่งๆ เช่น geographic data ซึ่งมีวิธีการมากมายในการแสดงข้อมูลและ manipulate ข้อมูลและในแต่ละวิธีการนั้นอาจจะเป็นวิธีที่ดีที่สุดกับบางโปรแกรมประยุกต์ตามลักษณะเฉพาะของข้อมูลนั้นๆ มันเป็นไปไม่ได้จริงๆ ที่จะเลือก set หนึ่งของ Data Type เป็น built-in Type และเป็นไปไม่ได้ที่จะจัดเตรียมไว้เพื่อความต้องการอย่างกว้างๆ ด้วย ในเวลานี้มันชัดเจนว่าเป็นไม่ได้ที่จะ built-in ตัว Data Type ทั้งหมดที่ใช้งานซึ่งอาจจะมาจากความต้องการของหลายๆ โปรแกรมประยุกต์ใน DBMS เดียว (จากที่กล่าวมานั้นจำเป็นต้องใช้กลไกในการจัดการเกี่ยวกับ user-defined Data Type ในการเพิ่ม Type ที่เป็นไปตามโปรแกรมประยุกต์ที่เปลี่ยนแปลงไป)

2.2.3 ปัญหาการขาด Built-in function ในการจัดการกับข้อมูลที่มีความซับซ้อน (Complex Data Type)

Built-in function ที่สนับสนุนใน RDBMS (เช่น AVG, SUM, MIN, MAX และ COUNT เป็นต้น) เป็น built-in function ที่ใช้จัดการข้อมูลที่เป็น traditional Data Type เท่านั้น ไม่สามารถใช้จัดการข้อมูลที่มีความซับซ้อนได้ ซึ่ง built-in function ดังกล่าวไม่เพียงพอและเหมาะสมกับความต้องการในการใช้งานกับโปรแกรมประยุกต์ในปัจจุบันหรือโปรแกรมประยุกต์ที่มีความซับซ้อน ในกรณีนี้ผู้พัฒนาต้องจัดการข้อมูลที่มความซับซ้อนดังกล่าวด้วยตัวเองโดยการสมมติ(จำลอง) Operator ขึ้นมาโดยใช้ Operator พื้นฐานที่มีอยู่ใน DBMS นั้นๆ และผลที่ได้ค่อนข้างมีความยุ่งยากและซับซ้อนมาก

2.2.4 ปัญหาการขาดความสามารถในการนำเสนอ Structure

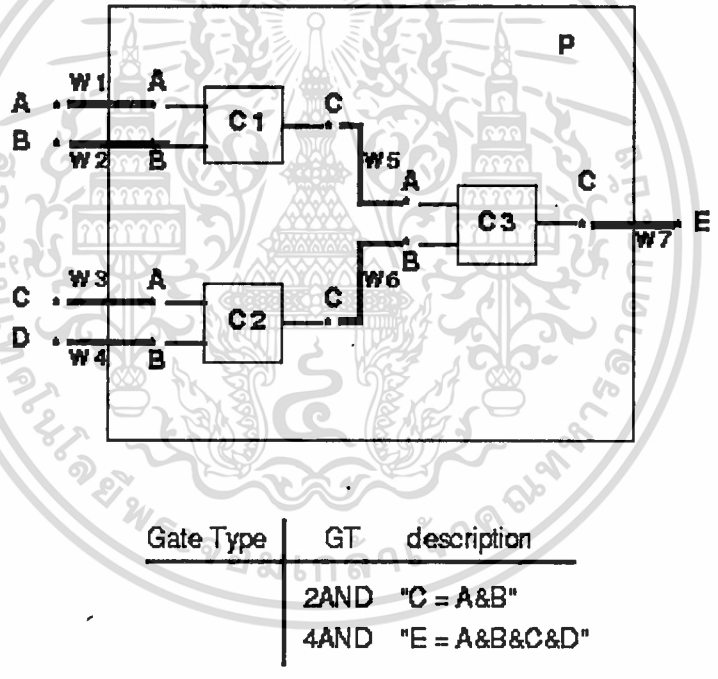
Structure ที่สนับสนุนใน RDBMS จะมีการจัดเก็บข้อมูลในลักษณะเป็น atomic ในขณะที่ DBMS application ใหม่หลายๆ ตัวจะเกี่ยวข้องกับโครงสร้างของออบเจ็กต์ระดับสูงซึ่งประกอบด้วยออบเจ็กต์อื่นๆ (บ่อยครั้งเป็นการอ้างถึง **Complex Object**) ทั้งนี้ Structure ที่สนับสนุนใน RDBMS นั้นยังไม่เหมาะสมเพียงพอกับโครงสร้างข้อมูลที่ซับซ้อนดังกล่าว ตัวอย่างเช่น part ใน part hierarchy อาจจะประกอบด้วย part อื่นๆ หรือใน IC หนึ่งๆ อาจจะประกอบด้วย pins, wires และ

module อื่นๆ หรือใน geographic ที่มี feature ที่ซับซ้อน เช่น industrial park อาจจะประกอบด้วย feature อื่นๆ เช่น buildings, smokestacks, and gardens หรือใน module ของโปรแกรมอาจจะประกอบด้วย module อื่นๆ อีก หรือว่าใน document อาจจะประกอบด้วย section และ front matter และใน section นั้นๆ อาจจะประกอบด้วย section headings, paragraphs ของ text และ figures ต่างๆ

ปัญหาดังกล่าวนั้นส่งผลให้การออกแบบโปรแกรมประยุกต์จะต้องมีการจัดการกับส่วนประกอบต่างๆ ซึ่งค่อนข้างยุ่งยากและมีความซับซ้อน ตัวอย่างเช่น เมื่อมีการลบ IC module ออกจากการออกแบบวงจร การลบนั้นจะต้องกระทำอย่างอัตโนมัติกับทุกๆ component ของ IC นั้นๆ

ซึ่งในบางครั้งเรื่องของ complex object ก็เป็นตัวกำหนดปัญหาเกี่ยวกับความยากง่ายใน RDBMS โดยสามารถแสดงตัวอย่างแบบง่ายๆ ได้ดังรูปที่ 2.2

ตัวอย่าง

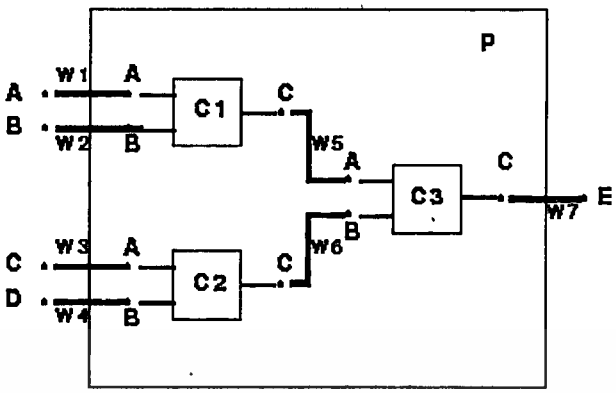


Simple Relational Representation of a Complex Object

รูปที่ 2.2 แสดง Relational อย่างง่ายของ Complex Object

จากรูปแสดงให้เห็นถึง **4-input AND gate** ซึ่งสร้างขึ้นจาก **2-input AND gate** สามอัน กับ relation ซึ่งไม่ได้แสดงถึงรายละเอียดของ gate หรือโครงสร้างภายในของ gate ในฐานข้อมูล แนวทางในการจัดเก็บรายละเอียดของออบเจ็กต์ดังกล่าวใน RDBMS มี 2 แนวทางหลักๆ ซึ่ง **วิธีการแรก** คือการ decompose ออบเจ็กต์(อย่างสมบูรณ์) ให้เก็บในอยู่ในรูปของ tuples หรือ row

โดยที่โครงสร้างที่สมบูรณ์ของออบเจ็กต์สามารถแสดงและ manipulate ได้โดยตรงด้วย DBMS วิธีการดังกล่าวสามารถแสดงได้ดังรูปที่ 2.3



Gate Type	GT	description
	2AND	"C = A&B"
	4AND	"E = A&B&C&D"

Gate Instance	GT	GI	Parent
	2AND	C1	4AND
	2AND	C2	4AND
	2AND	C3	4AND
	4AND	P	.....

Pin Type	GT	PT	I/O
	2AND	A	I
	2AND	B	I
	2AND	C	O
	4AND	A	I
	4AND	B	I
	4AND	C	I
	4AND	D	I
	4AND	E	O

Wire Instance	WI	GT1	GI1	Pin1	GT2	GI2	Pin2	Parent
	W1	4AND	P	A	2AND	C1	A	4AND
	W2	4AND	P	B	2AND	C1	B	4AND
	W3	4AND	P	C	2AND	C2	A	4AND
	W4	4AND	P	D	2AND	C2	B	4AND
	W5	2AND	C1	C	2AND	C3	A	4AND
	W6	2AND	C2	C	2AND	C3	B	4AND
	W7	2AND	C3	C	4AND	P	E	4AND

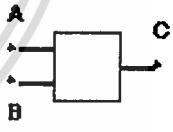
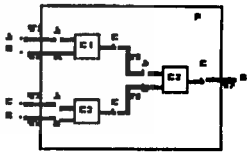
Complete Relational Representation of a Complex Object

รูปที่ 2.3 แสดง Relational อย่างสมบูรณ์ของ Complex Object

ในกรณีดังกล่าวนั้นใน relation ของ Gate Type และ Pin Type ได้อธิบายเกี่ยวกับ Type ทั้ง 2 ชนิดของ gate และ pin ในแต่ละ gate Type ตามลำดับ สำหรับใน relation ของ Gate Instance นั้นชี้ให้เห็นถึง instance แต่ละอันของ **4-input AND gate** คือการสร้างขึ้นจาก instance **2-input AND gate** 3 instance และใน relation ของ Wire Instance ได้แสดงให้เห็นว่า **4-input AND gate** นั้นประกอบด้วย wire 7 wire(ลวด) ซึ่งแต่ละอันจะเชื่อมด้วย pin 1 คู่ **ตัวอย่างเช่น** wire W1 เชื่อมต่อกับ Input pin A (จากภายนอก) ของ **4-input AND gate** ไปยัง pin A ของ component แรกที่เป็น **2-input AND gate** เป็นต้น

จากตัวอย่างดังกล่าวนั้นจะพบว่ามีความยุ่งยากในการออกแบบให้สมบูรณ์เพื่อที่จะแสดงโครงสร้างของ Complex Object ใน RDBMS ซึ่งจะต้องแยกส่วนต่างๆ ให้อยู่ในรูปแบบของ tuple ที่มีความสัมพันธ์กันมากมาย(จากตัวอย่างเป็นเพียง module ง่ายๆ) นอกจากนี้แล้วการ manipulate กับออบเจ็กต์ที่แสดงในรูปแบบดังกล่าวค่อนข้างมีความซับซ้อนและยุ่งยากมาก

วิธีการอีกวิธีหนึ่งที่ใช้กันคือการจัดเก็บออบเจ็กต์ทั้งหมดนั้นให้อยู่ในรูปแบบที่เรียกกันว่า BLOB โดยจะเก็บออบเจ็กต์ในรูปของ Binary ทั้งหมด ซึ่ง BLOB นั้นก็เป็นก็เป็น column หนึ่งใน relation โดยจะมีการใช้ร่วมกับข้อมูลใน column อื่นๆ ที่เป็นส่วนอธิบายเสมอ เช่น part number หรือ object Type ที่ใช้ในการ identify ข้อมูลที่เก็บในรูปแบบ BLOB ดังกล่าว ซึ่งวิธีการดังกล่าวแสดงได้ดังรูปที่ 2.4

Gate Type	GT	description	layout
	2AND	"C = A&B"	
	4AND	"E = A&B&C&D"	

รูปที่ 2.4 แสดง Relational ที่มีการใช้ BLOB นำเสนอข้อมูลลักษณะ Complex Object

จากรูปดังกล่าวใน relation ของ Gate Type นั้นพบว่าค่าใน column "layout" จะแสดงให้เห็นในลักษณะ graphic โดยใน column ดังกล่าวนี้จะเก็บข้อมูลในลักษณะ BLOB

วิธีการนี้ค่อนข้างยอมรับได้อย่างสมบูรณ์ ถ้ามีความต้องการแค่เพียงแสดง layout ของข้อมูลนั้นๆ เท่านั้น อย่างไรก็ตามวิธีการนี้ DBMS ไม่สามารถถูกใช้เพื่อทำการ manipulation ที่เกี่ยวกับ

โครงสร้างของออบเจ็กต์นั้นๆ ได้ตัวอย่างเช่นเราไม่สามารถที่จะ identified ถึงตัวที่เป็น subcomponent ของออบเจ็กต์นั้นได้ หรือไม่สามารถที่จะดึงข้อมูลในส่วนที่เป็น subcomponent ได้เลยเนื่องจาก BLOB มีลักษณะการจัดเก็บเป็น Binary ทั้งหมด

- ข้อจำกัดในเรื่องของ BLOB

จากตัวอย่างที่ผ่านมาเป็นการแก้ไขปัญหาการเก็บรายละเอียดของข้อมูลที่มีลักษณะโครงสร้างซับซ้อนโดยใช้วิธีการเก็บข้อมูลในลักษณะ **BLOBS** หรือ Binary Large Objects ซึ่งสามารถนิยามได้ดังนี้

"*BLOBS* เป็น case พิเศษใน RDBMS ที่สนับสนุนให้มีการเก็บ(store) ลงในฐานข้อมูลและสนับสนุนให้มีการ retrieve ในส่วนของ Data Type ที่เพิ่มขึ้นมา โดยเฉพาะกับข้อมูลชนิด multimedia (เช่น text, images, audio clips, videostreams เป็นต้น) แต่มีข้อจำกัดเนื่องจากไม่สามารถใช้ในการ Query ในส่วน content ของ BLOB ได้ ตัวอย่างเช่น เราสามารถทำได้เพียงเก็บและดึง image ได้เท่านั้น แต่ไม่สามารถ Query ฐานข้อมูลสำหรับ image ที่ match กับสีดังกล่าว เป็นต้น" หรือสามารถยกตัวอย่างโดยเขียนในรูปของ SQL โดยใช้ตัวอย่างที่ผ่านมาข้างต้น ได้ดังนี้

```
SELECT *
FROM GATE_TYPE
WHERE LAYOUT = ????
```

ในส่วนนี้จะเห็นได้ว่าเราไม่สามารถใช้ข้อมูลที่จัดเก็บในลักษณะ BLOB ในการ Query ได้

- Object-Oriented Application กับระบบการจัดการฐานข้อมูล

จากปัญหาทางด้าน structure นั้น ใน object-oriented application ที่มีการอ้างถึงความสัมพันธ์ของ complex data ที่ทำงานบนฐานข้อมูลแบบ relational หรือฐานข้อมูลแบบ object relational นั้น code ที่ถูกเขียนจำเป็นต้อง map โครงสร้างของ complex object ไปสู่รูปแบบการเก็บข้อมูลของ relational database เหมือนกับการแยกส่วนประกอบของรถยนต์ก่อนที่จะเก็บมันในโรงรถในแต่ละคืนและประกอบเข้าไปใหม่ก่อนที่จะใช้มันในทุกๆ เช้า ซึ่งDeveloper ต้องใช้เวลาประมาณ 25% ของการพัฒนา ในการ map โปรแกรม object นั้นไปเป็นฐานข้อมูล ซึ่งเป็นส่วนที่สำคัญ ทั้งนี้ยังทำให้โปรแกรมประยุกต์นั้น maintain ได้ง่ายกว่าด้วย และจากการ map จากฐานข้อมูลไปยังโครงสร้างของโปรแกรมประยุกต์นั้นส่วนมากจะเป็นหน้าที่ของโปรแกรมประยุกต์มากกว่าที่

จะเป็นหน้าที่ของ DBMS ซึ่งลักษณะดังกล่าวจะเพิ่มความเสี่ยงในการละเมิด integrity ได้ ทั้งนี้ในแต่ละโปรแกรมประยุกต์นั้นอาจจะมีการ map ที่มีลักษณะแตกต่างกันไป ในขณะที่ ODBMS นั้นสนับสนุน object-oriented application โดยตรง และสนับสนุนแนวความคิดของ object-oriented ทั้งหมด เช่น การ encapsulation, การ inheritance และ polymorphism เป็นต้น

2.2.5 ปัญหาการจัดการ Integrity rules ที่กระทำโดย procedure ซึ่งเป็นคนละส่วนกับ Data Structure

การจัดการ Integrity rules ใน RDBMS นั้นจะเป็นการกระทำจาก procedure (อาจเป็น store procedure) หรือ program ทางฝั่งโปรแกรมประยุกต์มากกว่าที่จะกระทำโดย RDBMS เอง ซึ่ง integrity rule ดังกล่าวจะถูกแยกเป็นคนละส่วนกับโครงสร้างข้อมูล ส่งผลให้การ maintain มีความยากลำบาก ตัวอย่างเช่น การโฆษณาทางหนังสือพิมพ์ จะมีค่า "change fee" ซึ่งลูกค้าหรือเจ้าของโฆษณาต้องเสียค่าธรรมเนียมถ้ามีความต้องการให้เปลี่ยนแปลงแก้ไขสำเนาหลังจากมีการตกลงกันไปแล้ว แต่ถ้าการเปลี่ยนแปลงแก้ไขเป็นการความต้องการของบรรณาธิการ(โดยสั่งขณะทำสำเนา และมีการเปลี่ยนแปลงรูปแบบเล็กน้อย) ก็จะไม่มีค่าธรรมเนียมดังกล่าว จากการกระทำทั้ง 2 ลักษณะเป็นการเปลี่ยนแปลง Layout ของโฆษณาเหมือนกันแต่จะมีกรณีหนึ่งที่มีการ charge ค่าธรรมเนียมกับลูกค้า ซึ่งการควบคุมความถูกต้องดังกล่าวผู้พัฒนาจะต้องกระทำเองโดยแยกจากโครงสร้างข้อมูลในฐานข้อมูล และในกรณีที่ rules ดังกล่าวถูกจัดการที่ฝั่งโปรแกรมประยุกต์การเปลี่ยนแปลง rules จะทำได้ยากเนื่องจากอาจต้องมีการ recompile ใหม่ เป็นต้น

2.2.6 ปัญหาการจัดการ Identifier keys

ในระบบฐานข้อมูลเชิงสัมพันธ์จะใช้ unique key หรือ identifier key เป็นกลไกในการอ้างถึงข้อมูล ซึ่ง identifier key ก็คือ subset ของ attribute ของ object ซึ่ง unique สำหรับทุก object ใน relation นั้น สำหรับการใช้ identifier key มีปัญหาหลักๆ 3 ปัญหาดังนี้

1. Modifying identifier keys เป็นปัญหาที่ identifier key ไม่สามารถ(ไม่ควรจะ) ยอมให้มีการเปลี่ยนแปลงถึงแม้ว่า identifier key จะเป็นข้อมูลในลักษณะ user-defined
2. Nonuniformity คือปัญหาของ identifier key ที่อยู่ต่างตารางกันมีชนิดข้อมูลต่างกัน (integer, character string, floating point) และการ combination ของ attribute ที่แตกต่างกัน ตัวอย่างเช่น identifier ในตาราง Item คือ ItemNumber ซึ่งมีชนิดของข้อมูลเป็น integer และในตาราง Person มี attribute FirstName และ LastName รวมกันเป็น identifier key ซึ่งมีชนิดของข้อมูลเป็น string ทั้งนี้ความแตกต่างของชนิดข้อมูลที่เก็บ

หรือชนิดข้อมูลของ attribute ที่ใช้ในการชี้ไปยังข้อมูลมีความไม่สอดคล้องกัน และเป็นสาเหตุทำให้มีความยุ่งยากเพิ่มขึ้นเมื่อมีการกระทำกับตารางหลายๆ ตาราง

3. "Unnatural" joins เป็นปัญหาที่ 3 ซึ่งการใช้ identifier key เป็นสาเหตุให้การ join ถูกนำมาใช้ในการดึงข้อมูลแทนที่จะมีการดึงข้อมูลจาก object โดยตรงซึ่งเป็นวิธีที่ง่ายกว่า ตัวอย่างเช่น ถ้ามีเรา relation Employee และ relation Department

EMPLOYEE (NAME, AGE, ADDRESS, SALARY, DEPTNAME)

DEPARTMENT (NAME, BUDGET, LOCATION, ...)

จาก relation ดังกล่าวจะเห็นได้ว่า attribute DeptName จะถูกกำหนดขึ้นเพื่อเชื่อมความสัมพันธ์ ระหว่าง Employee และ Department ในการใช้ identifier key นั้น DeptName จะมีค่าของ identifier key ของ Department ทั้งนี้ในการดึงข้อมูลที่มีการอ้างอิงทั้ง 2 relation ทำให้ต้องมีการ join ระหว่าง 2 relation ดังกล่าว เช่น ถ้าต้องการดึงข้อมูลทั้งหมดซึ่งประกอบด้วย Name ของ employee และ Location ที่ employee ทำงานอยู่ สามารถแสดงด้วยคำสั่ง SQL ได้ดังนี้

```
SELECT EMPLOYEE.NAME, DEPARTMENT.LOCATION
FROM EMPLOYEE, DEPARTMENT
WHERE EMPLOYEE.DEPTNAME = DEPARTMENT.NAME ;
```

บทที่ 3
แนวคิดเชิงวัตถุ

แนวความคิดแบบเชิงวัตถุเป็นกุญแจของกลไกการทำงานที่จะขยายขีดความสามารถของ DBMS ในปัจจุบัน การศึกษาในบทนี้จะทำการเป็นการมองภาพทั่วๆ ไปโดยรวมของคุณลักษณะของ OODBMS ในด้านต่างๆ ซึ่งคุณลักษณะเหล่านี้มีความสอดคล้องกับแนวความคิดเชิงวัตถุ และในส่วนสุดท้ายจะเป็นการกล่าวถึงข้อดีที่เป็นลักษณะทั่วไปในการกล่าวถึงการโปรแกรมเชิงวัตถุ

3.1 แนวคิดเชิงวัตถุ

เดิมแนวความคิดในลักษณะออบเจ็กต์ถูกออกแบบบนภาษา Simula เป็นครั้งแรกในปี ค.ศ. 1967 อย่างไรก็ตามลักษณะการโปรแกรมเชิงวัตถุ ก็ไม่ได้ปรากฏเป็นรูปแบบการโปรแกรมใหม่ จนกระทั่งมีการพัฒนาภาษา Smalltalk ขึ้นมาใช้ และในปัจจุบันก็มีภาษาที่สนับสนุนการโปรแกรมเชิงวัตถุออกมากมายและมีการใช้งานกันอย่างแพร่หลาย

ในอดีตที่ผ่านมาภาษาที่ใช้ในการโปรแกรมส่วนใหญ่จะอยู่ในรูปแบบ Operator/Operand (หรือที่เรียกว่า Procedural Programming) ในการทำงาน โดยจะแยกส่วนที่เป็นโพรซีเจอร์ (Procedure) กับส่วนที่เป็นข้อมูล(Data) ออกจากกัน ในรูปแบบการโปรแกรมดังกล่าวผู้ใช้หรือโปรแกรมเมอร์จะใช้กลุ่มของข้อมูล (Operand) กับกลุ่มของโอเปอเรเตอร์ (Function หรือ Procedure) มาทำงานร่วมกันโดย กลุ่มของข้อมูลจะเปรียบเสมือนพารามิเตอร์(Parameter) ของโอเปอเรเตอร์ในการทำงาน ซึ่งจำเป็นต้องใช้ข้อมูลให้ตรงกับรูปแบบ(Type) ที่กำหนดไว้เพื่อให้การทำงานถูกต้อง ภาษาที่มีรูปแบบการโปรแกรมในลักษณะดังกล่าวเช่นภาษา C, Pascal หรือว่า Fortran เป็นต้น ในกรณีที่มีการใช้โอเปอเรเตอร์ในลักษณะเดียวกัน เช่น print โดยจะกระทำกับ โอเปอแรนด์(Operand) ที่มีลักษณะแตกต่างกันจำเป็นต้องมีการแยกโพรซีเจอร์ในการกำหนดและใช้งาน ตัวอย่างเช่น prnPoint(point) จะเป็นการ print ที่ใช้กับข้อมูลชนิด point หรือ prnRectangle(rect) จะเป็นการใช้กับข้อมูลชนิด rectangle ซึ่งจะเห็นได้ว่าจำเป็นต้องมีการระบุหรือกำหนดรูปแบบของชนิดข้อมูลไว้อย่างชัดเจนและมีการอิมพลีเมนท์(Implement) ที่แตกต่างกันออกไปตามลักษณะของข้อมูลนั้นๆ รวมทั้งการเขียนโปรแกรมในลักษณะดังกล่าวนี้จะพบว่ามีข้อจำกัดอยู่มากในเรื่องของการนำโปรแกรมมาพัฒนาให้ทันสมัยซึ่งทำได้ยาก เพราะต้องทำการศึกษาโปรแกรมเก่าให้เข้าใจ

อย่างแท้จริงก่อนลงมือแก้ไข และไม่มีความปลอดภัยของข้อมูล เพราะทุกโพรเซส(Process) สามารถนำข้อมูลนั้นไปใช้งานได้ตามต้องการ

ภาษาบางภาษาเช่น Simula, Ada, หรือ CLU มีการเพิ่มเติมวิธีการเขียนโปรแกรมจากวิธีการดังกล่าวซึ่งโปรแกรมเมอร์สามารถที่จะระบุ Abstract Operation Name และ Compiler Differentiate ซึ่งจะเป็นการระบุรูทีน(Routine) ที่เรียกใช้ในช่วงที่คอมไพล์ ซึ่งจะมีพื้นฐานอยู่บนชนิดของ โอเปอแรนด์ ตัวอย่างเช่น โปรแกรมเมอร์สามารถเขียน print(point) หรือ print(rect) ได้โดยคอมไพเลอร์ (compiler) จะถือว่าเป็นการอ้างถึงโพรซีเจอร์(Procedure) ต่างกันในแต่ละกรณี

จากรูปแบบดังกล่าวเป็นการริเริ่มแนวคิดการเขียนโปรแกรมเชิงวัตถุขึ้นมา โดยเป็นวิธีใหม่ในการพัฒนาซอฟต์แวร์ เพื่อเพิ่มประสิทธิภาพในการผลิตซอฟต์แวร์ วิธีนี้ทำให้การออกแบบและการเขียนซอฟต์แวร์มีความใกล้ชิดกันมากขึ้น. โดยจะมีรูปแบบการโปรแกรมในลักษณะ Object/Message แทนรูปแบบ Operator/Operand ซึ่งการโปรแกรมในรูปแบบดังกล่าว จะกำหนดสิ่งต่างๆ ที่อยู่ภายในขอบเขตของระบบงานเป็นออบเจ็กต์ซึ่งในเมธอดจะประกอบด้วยข้อมูลหรือแอตทริบิวต์ และเมธอด(กลุ่มของ Operation) รวมกันอยู่ภายในออบเจ็กต์และการเข้าถึงข้อมูลนั้นได้จะต้องผ่านเมธอด(Method) ที่กำหนดไว้เท่านั้น (จากหลักการ Encapsulation) เมธอดเปรียบเทียบได้ทำนองเดียวกันกับโพรซีเจอร์ในโปรแกรมระบบงานแบบเดิมนั่นเอง การเรียกใช้เมธอด หรือการสั่งให้เกิดการกระทำตามที่กำหนดไว้ในเมธอดทำได้โดยการส่งเมสเซจ(Message) ไปยังออบเจ็กต์นั้นๆ ดังนั้นหากมีการเปลี่ยนแปลงวิธีการกระทำ (Function) ใดๆ ของออบเจ็กต์นั้นจะไม่มีผลกระทบกับเมสเซจที่ส่งไปยัง ออบเจ็กต์ นั้นและไม่มีผลกระทบกับ ออบเจ็กต์ อื่นๆ ด้วย ซึ่งนับว่าเป็นผลดีอย่างมากกับระบบงานเพราะโดยปกติแล้วพบว่าระบบงานมักจะมีการเปลี่ยนแปลงวิธีการปฏิบัติอยู่เสมอๆ เมธอดสามารถใช้อธิบายถึงพฤติการณ์ของออบเจ็กต์แต่ละออบเจ็กต์ได้ ออบเจ็กต์ที่มีพฤติกรรมแบบเดียวกันสามารถจัดรวมกลุ่มเป็นกลุ่มเดียวกันได้เรียกว่าคลาส(Class) ในระหว่างคลาสสามารถที่จะถ่ายทอดคุณสมบัติ หรือที่เรียกว่า inheritance จาก คลาส หนึ่งไปยังอีก คลาสหนึ่งได้ โดยเรียก คลาส ที่ถ่ายทอดคุณสมบัติว่า ซูปเปอร์คลาส(Superclass) และเรียกคลาสที่ได้รับการถ่ายทอดคุณสมบัติว่า ซับคลาส(Subclass) ซึ่งทำให้การบำรุงรักษาระบบงานทำได้ง่าย และสามารถขจัดปัญหาต่างๆ ในการเขียนโปรแกรมแบบโพรซีเดอรอล(Procedural) ได้อีกด้วย

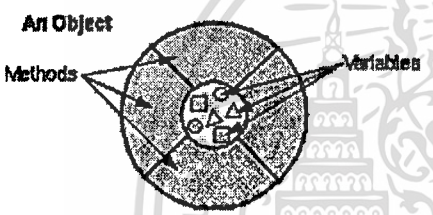
3.2 วิธีการเชิงวัตถุ(Object-Oriented Methodology)

ในหัวข้อที่กล่าวมาเป็นแนวคิดพื้นฐานเชิงวัตถุ ซึ่งจะเห็นได้ว่าแนวคิดดังกล่าวได้กลายมาเป็น Paradigm ใหม่ทีมีการจัดการกับปัญหาโดยมองเป็น ออบเจ็กต์ ซึ่งกำลังเป็นที่นิยมในปัจจุบัน และมี tool ที่สำคัญคือ ออบเจ็กต์โมเดล (รวม Data กับ Process เข้าด้วยกัน) ซึ่งมีภาษาที่สามารถใช้

ร่วมกับเทคโนโลยีนี้คือ C++, Smalltalk , Eiffel, Object C, Object Pascal และ Java ในขณะที่ Delphi และ VB กำลังพยายามทำอยู่

คำว่า "ออบเจ็กต์" (Real World Oobject) โดยทั่วไปจะหมายถึง วัตถุใดๆจะมีสิ่งสำคัญคือ ลักษณะ, สถานะ และ พฤติกรรม (Attribute, State, Behavior) ส่วนออบเจ็กต์ซอฟต์แวร์จะพยายามเลียนแบบ Real World Object โดยแบ่งออกเป็น 2 ส่วน คือ ส่วนที่เป็นสถานะ(Data and Variable) และพฤติกรรม (Method, Process, Operation)

3.2.1 Objects เป็นการรวมกันทางด้านโครงสร้างของข้อมูล(Attribute , Properties) และ Function Logic (หรือ Method) ซึ่งข้อมูลจะบอกถึงคุณสมบัติหรือสถานะของออบเจ็กต์และเมธอดจะบอกถึงพฤติกรรมต่างๆของออบเจ็กต์นั้นๆ โดย



รูปที่ 3.1 แสดงออบเจ็กต์โมเดล

- แอตทริบิวต์หรือตัวแปรของออบเจ็กต์คือข้อมูลที่เราสนใจเกี่ยวกับออบเจ็กต์นั้น
- เมธอดจะแบ่งเป็น 2 ประเภทคือ Interface Method (เป็นเมธอดที่ถูกใช้ได้จากออบเจ็กต์อื่น) และ Internal Method (เป็นเมธอดที่จะถูกเรียกใช้ได้เฉพาะภายในออบเจ็กต์ที่เป็นเจ้าของเท่านั้น)

นอกจากนี้แล้ว State of Object หรือสถานะของออบเจ็กต์ ณ ขณะเวลาใดเวลาหนึ่ง ซึ่งจะมีการเปลี่ยนแปลงสถานะเมื่อมีการกระทำจากภายนอก แต่มีออบเจ็กต์บางตัวสามารถเปลี่ยนสถานะด้วยตัวเองได้ เรียกว่า " Object with life" หรือ "Actor" เช่น นาฬิกา, ลูกค้า เป็นต้น

3.2.2 Class เป็นการจัดกลุ่มของออบเจ็กต์ตามคุณสมบัติ และผู้ใช้สามารถติดต่อกับคลาสโดยผ่านทาง เมธอดภายในคลาส หรือเมธอดในซูปเปอร์คลาส(หรือ Parent Class) เท่านั้น ทั้งนี้คลาสเองก็เป็น ออบเจ็กต์แต่เป็นออบเจ็กต์ที่ไม่สามารถนำมาใช้ได้โดยตรง โดยจะมีการบอกถึงเมธอดที่ใช้ได้โดย ออบเจ็กต์และมีการแสดงประเภทของข้อมูล(Data Type) ที่บอกถึงสถานะของออบเจ็กต์โดยยังไม่ระบุค่าในแอตทริบิวต์แต่ละตัวซึ่งถ้าเป็นออบเจ็กต์จะมีการระบุค่าของแอตทริบิวต์ที่แน่นอน ถ้าออบเจ็กต์เป็นอินสแตนท์(Instance) ของคลาสใดออบเจ็กต์นั้นจะต้องมีแอตทริบิวต์และเมธอด เหมือนคลาส ของมัน

3.2.3 Method เป็นการทำงาน (Operation) ที่สำคัญหรือเป็นวิธีการกระทำที่สามารถทำงานกับออบเจ็กต์ได้ แต่ละคลาสจะมีเมธธอดของตัวเอง โดยการทำงาน จะเริ่มจากการส่งเมสเซจไปยังออบเจ็กต์ที่ต้องการ(Receiver Object) เพื่อเรียกใช้เมธธอดที่อยู่ภายในออบเจ็กต์นั้น

ตัวอย่าง การส่งเมสเซจไปยัง Receiver Object

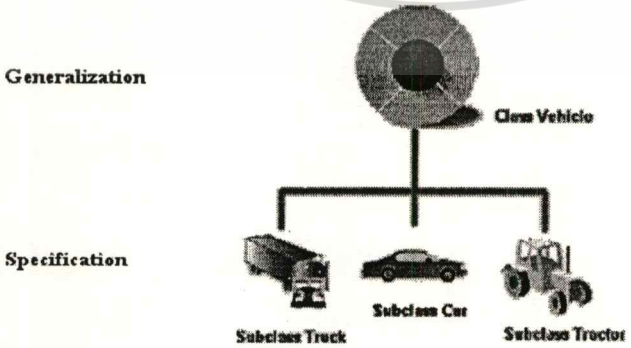
receiver . message_name (a1, a2 , a3)

receiver . message_name: a1 parm1: a2 parm3: a3

3.2.4 Encapsulation/Information hiding ทฤษฎีนี้ถูกคิดค้นโดย James Rambaugh ซึ่งหมายถึง การแยกลักษณะภายนอก (Interface) ซึ่งสามารถติดต่อกับออบเจ็กต์อื่นๆ ออกจากส่วนที่อิมพลีเมนท์ภายในของออบเจ็กต์ซึ่งจะถูกใช้ได้เฉพาะตัวออบเจ็กต์นั้นๆ มีข้อดีคือ เพิ่มความสามารถในการปกป้องข้อมูลโดยไม่ให้ออบเจ็กต์อื่นเข้ามาทำการเปลี่ยนแปลงข้อมูลก่อนได้รับอนุญาติ ทั้งนี้ตัวออบเจ็กต์ควรรู้และทำการเปลี่ยนค่าด้วยตนเองซึ่งเป็นผลให้มีการจัดการออบเจ็กต์ได้ง่ายขึ้น

3.2.5 Inheritance คือหลักการในการที่ออบเจ็กต์ทุกๆตัวใน Generalized Collection ได้มีการใช้ข้อมูล(Structure) และพฤติกรรม(Behavior) ร่วมกันซึ่งจะมีความสัมพันธ์แบบ is-a relationship โดยเรียก คลาสที่อยู่เหนือกว่าว่า ซูปเปอร์คลาส ซึ่งจะถ่ายทอดคุณสมบัติทั้ง แอตทริบิวต์และเมธอดมายัง คลาสที่ต่ำกว่าเรียกว่า ซับคลาส ซึ่งจะมีแอตทริบิวต์และเมธอดเพิ่มเติมจากซูปเปอร์คลาส ซึ่งมีข้อดีดังนี้

- เพิ่ม Consistency เพียงแค่เปลี่ยนเมธอดหรือแอตทริบิวต์ที่ซูปเปอร์คลาสซึ่งเป็นผลให้ค่าที่ซับคลาสเปลี่ยนไปด้วย
- เป็นการส่งเสริมการนำออบเจ็กต์กลับมาใช้ใหม่



รูปที่ 3.2 แสดงลักษณะการสืบทอดตามหลักการ Inheritance

3.2.6 Polymorphism คือ การที่เราส่งเมสเซจที่เหมือนกัน ไปในออบเจ็กต์ที่ต่างกัน แต่ละออบเจ็กต์จะตอบสนองออกมาไม่เหมือนกันตามแต่ชนิดและหน้าที่ของออบเจ็กต์ ความสามารถที่ใช้เมสเซจเหมือนกัน สำหรับการกระทำที่เหมือนกัน ไปยังออบเจ็กต์ต่างชนิดกัน มีลักษณะเหมือนกับการที่มนุษย์คิดในการแก้ปัญหาหนึ่ง ๆ

ตัวอย่าง การ Polymorphism

- AB + CD ได้ผลลัพธ์ABCD(Concat)
- 5 + 3ได้ผลลัพธ์ 8 (Add)

ทั้ง 2 แบบ เป็นการส่งเมสเซจ '+' เข้าไปยังออบเจ็กต์ภายในคลาส String และ Integer ตามลำดับ

3.2.7 Dynamic Binding (บางครั้งเรียกว่า Late Binding) คือ การนำโปรแกรมย่อยๆ มาประกอบให้ใช้งานในขณะ Run Time โดยในขณะ Compile Time นั้นจะเก็บโปรแกรมในรูปแบบของคลาส ต่างๆ ไว้ เพื่อไม่ให้เกิดความยุ่งยาก ซับซ้อน ต่อจากนั้น เมื่อนำมาใช้ในขณะ Run Time เมื่อมีการเรียกใช้คลาสนั้นๆ จะทำการนำคลาสนั้นมาไว้ในส่วนของโปรแกรม และเมื่อใช้งานเสร็จแล้วจะถูกลบออกจากหน่วยความจำ

3.2.8 Override คือ การที่สร้างซับคลาส ขึ้นมาใหม่ และมีการสร้างเมธอดที่ซ้ำกับเมธอดที่เป็นของซูปเปอร์คลาส

3.2.9 Overload คือ การที่สร้างเมธอดชื่อเหมือนกันแต่รับพารามิเตอร์ต่างกัน ภายในคลาสเดียวกัน

3.3 ความสัมพันธ์ระหว่างออบเจ็กต์

ความสัมพันธ์ที่เกิดขึ้นระหว่างออบเจ็กต์มี 4 แบบคือ

3.3.1 Association

เป็นการอธิบายถึงความสัมพันธ์ระหว่างโครงสร้างโดยทั่วไปกับซีแมนติก(Semantic) ทั่วไป โดยอ้างถึงความสัมพันธ์ทั้งสองด้านและมีการระบุชื่อความสัมพันธ์อย่างชัดเจน ความสัมพันธ์แบบนี้ประกอบด้วย

- ข้อจำกัด(Constraint) คือการแสดงข้อมูลที่เกี่ยวกับความสัมพันธ์ระหว่างออบเจ็กต์
- ลิงก์(Link) คือการเชื่อมต่อระหว่างออบเจ็กต์ที่มีความสัมพันธ์แบบถาวรซึ่งเป็นอินสแตนท์ของ Association

3.3.2 Aggregation

เป็นรูปแบบความสัมพันธ์อย่างหนึ่งของ Association ในการกำหนด Parts ในส่วนประกอบนั้นโดยมีการแยกระหว่าง Whole กับ Parts ซึ่งจะได้ความสัมพันธ์แบบ Whole - Part หรือ a-part-of หรืออาจเรียกอีกแบบหนึ่งว่ามีความสัมพันธ์แบบ has-a relationship เป็นผลให้ซูปเปอร์คลาส มีอิทธิพลต่อการดำรงอยู่ของ ซับคลาส คือถ้า ซูปเปอร์คลาส ถูกทำลาย ซับคลาส จะถูกทำลายไปด้วยโดยอัตโนมัติ นอกจากนี้การพิจารณาถึงการแบ่งระดับของ Aggregation นั้นขึ้นอยู่กับความต้องการของระบบ

3.3.3 Generalization

เป็นความสัมพันธ์ในการจัดกลุ่มสิ่งที่เหมือนกันระหว่างคลาสของออบเจ็กต์หรือคลาสขึ้นมาเป็นอีกคลาสของออบเจ็กต์หนึ่งโดยพิจารณาจากข้อมูล(Structure) หรือพฤติกรรม(Behavior) ในกลุ่มของ คลาสของออบเจ็กต์นั้น จะมีลักษณะเป็นลักษณะทั่วไปไม่ชี้เฉพาะเจาะจง

3.3.4 Depends On

เป็นความสัมพันธ์ที่ชี้ให้เห็นถึงคลาสของออบเจ็กต์ที่ร้องขอ(Source หรือ Client) ขึ้นอยู่กับคลาสของออบเจ็กต์ปลายทาง(Destination หรือ Supplier) เช่นในคลาสหนึ่งสามารถมีเมธอดที่ขึ้นอยู่กับคลาสอื่นเวลาทำงานจะมีการอ้างถึงคลาสนั้นด้วย

3.4 การส่งผ่านเมสเซจ(Message Passing)

การติดต่อกันระหว่างคลาสของออบเจ็กต์นั้นจะใช้วิธีการติดต่อกันด้วยการส่งข้อความ (Message) โดยใช้เทคนิคที่เรียกว่า "Message Passing" โดยข้อความที่ส่งถึงกันนั้นจะประกอบด้วยจุดหมายปลายทาง (Destination) ของข้อความนั้นและข้อมูลที่สำคัญ (Argument หรือ Parameter) Message Passing เปรียบได้กับการเรียกใช้ฟังก์ชันหรือโพรซีเจอร์(Function Call หรือ Procedure Call) ที่มีในการโปรแกรมแบบโครงสร้าง(Structure Programming) โดยผ่าน Interface Method ของคลาสของออบเจ็กต์นั้นๆ มีผลทำให้คลาสของออบเจ็กต์ที่เป็นผู้รับข้อความนั้นกระทำการอย่างใดอย่างหนึ่ง

3.5 ข้อดีที่เป็นลักษณะทั่วไปในการกล่าวถึงการโปรแกรมเชิงวัตถุ

- การอธิบายความหมายของระบบในรูปแบบของออบเจ็กต์ช่วยให้ง่ายต่อการสร้างส่วนประกอบต่างๆ ของซอฟต์แวร์ซึ่งจะสอดคล้องตามลักษณะของโปรแกรมประยุกต์และยังช่วยในการออกแบบรวมทั้งการทำความเข้าใจระบบได้ง่าย
- ประโยชน์ของออบเจ็กต์และเมสเซจช่วยสนับสนุนการออกแบบระบบในลักษณะโมดูลลาร์ (Modular) ทั้งนี้การอิมพลีเมนท์ในแต่ละออบเจ็กต์จะไม่ขึ้นกับการออกแบบภายในของออบเจ็กต์อื่นๆ โดยจะขึ้นกับการตอบสนองกับเมสเซจที่เข้ามาเท่านั้นว่าจะเกิดขึ้นอย่างไร ทั้งนี้เนื่องจากส่วน Private Information ของออบเจ็กต์นั้นสามารถเข้าถึงได้เฉพาะเมธอดของออบเจ็กต์นั้นๆ
- ประโยชน์ของคลาสและการสืบทอดจะเป็นรูปแบบที่ใช้แสดงถึงความสัมพันธ์ของส่วนประกอบต่างๆ ในระบบที่กล่าวถึง ทั้งนี้ยังสนับสนุนการนำส่วนประกอบต่างๆ ที่มีกลับมาใช้ใหม่ และการเพิ่มเติมปรับปรุงแก้ไขระบบเดิม หรือนำไปใช้ในระบบใหม่ได้ง่าย ซึ่งจะช่วยลดระยะเวลาที่ใช้ในการพัฒนาได้

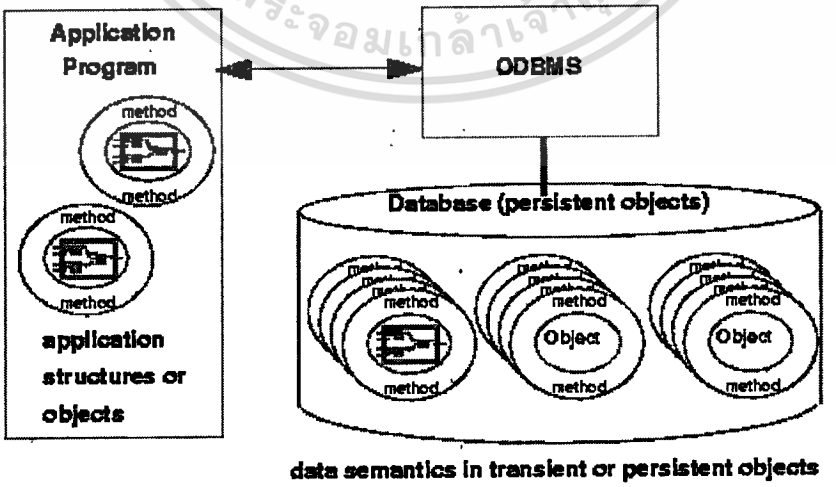
บทที่ 4
ระบบการจัดการฐานข้อมูลเชิงวัตถุ

แนวความคิดแบบเชิงวัตถุเป็นกุญแจของกลไกการทำงานที่จะขยายขีดความสามารถของ DBMS ในปัจจุบัน การศึกษาในบทนี้จะเป็นการมองภาพทั่วๆ ไปของOODBMS ในด้านต่างๆ และในส่วนสุดท้ายจะเป็นการอธิบายถึงหลักเกณฑ์ในการพิจารณาระบบที่มีพื้นฐานอยู่บน OODBMS รวมทั้งการเปรียบเทียบการประยุกต์ใช้งานสารสนเทศบนระบบการจัดการฐานข้อมูลเชิงวัตถุกับระบบการจัดการฐานข้อมูลเชิงสัมพันธ์

4.1 ระบบการจัดการฐานข้อมูลเชิงวัตถุ(Object-Oriented DBMS)

OODBMS(Object-Oriented Database Management Systems) ทั่วๆ ไปได้พยายามนำเอาแนวความคิดของการโปรแกรมเชิงวัตถุไปใช้และเพิ่มคุณลักษณะบางอย่างในการทำงานที่จำเป็นเพื่อสนับสนุนข้อมูลขนาดใหญ่, การใช้งานร่วมกัน รวมทั้งการคงอยู่ของออบเจ็กต์ที่เก็บ(Persistent Object Stores) ซึ่งคุณลักษณะดังกล่าวยังรวมถึงการสนับสนุนการร้องขอแบบ Set-Oriented หรือการสอบถามข้อมูล(Query), ประสิทธิภาพในการประมวลผลบนโครงสร้างของ Secondary Storage ขนาดใหญ่, การควบคุมความถูกต้อง(Concurrency Control) และ การเรียกข้อมูลกลับ(Recovery)

จากแนวความคิดทั่วไป OODBMS นั้นไม่ใช่ DBMS ที่เก็บเฉพาะข้อมูล แต่จะเก็บอยู่ในรูปของออบเจ็กต์ซึ่งสามารถแสดงได้ดังรูปที่ 4.1



รูปที่ 4.1 แสดงลักษณะของ OODBMS

เมื่อทำการเปรียบเทียบกับระบบฐานข้อมูลที่ใช้กันในปัจจุบันหรือ RDBMS นั้นจะมีข้อแตกต่างกับ OODBMS อย่างน้อยดังนี้

1. OODBMS สนับสนุนการกำหนดโครงสร้างข้อมูลและการทำงานโดยผู้ใช้เอง
2. OODBMS สนับสนุนแนวความคิดเกี่ยวกับ Object Identity ซึ่งหมายความว่าออบเจ็กต์จะมีการชี้หรือระบุ อย่างอิสระทั้งภายในออบเจ็กต์เองหรือระหว่างออบเจ็กต์ก็ตาม
3. OODBMS สนับสนุนความสัมพันธ์ของออบเจ็กต์ทางตรง(Direct Object Relationships) ซึ่งหมายความว่าออบเจ็กต์ที่มีความสัมพันธ์กันสามารถแอ็กเซส (Access) ผ่านทางการกระทำกับเมธอดที่มีอยู่ในออบเจ็กต์นั้นๆ โดยทั่วไปแล้วจะเป็นการมองหา Object Identity ของออบเจ็กต์ที่สัมพันธ์กันในส่วนของแอตทริบิวต์ ซึ่งเป็นการกระทำที่เกิดขึ้นภายในออบเจ็กต์ที่สัมพันธ์กันดังกล่าว

พื้นฐานที่อยู่บน Object-Oriented DBMS จะมีหลักเกณฑ์ที่ใช้ในการพิจารณา 2 หลักเกณฑ์ด้วยกันคือ 1) ระบบต้องมีรูปแบบเชิงวัตถุ 2) ตัวระบบต้องเป็น DBMS จากหลักเกณฑ์ดังกล่าวสามารถอธิบายเพิ่มเติมได้โดย หลักเกณฑ์ที่ 1) สามารถแปลงได้เป็น 8 คุณลักษณะดังนี้คือ Complex Object, Object Identity, Encapsulation, Types or Classes, Inheritance, Overriding Combined with Late Binding, Extensibility และ Computation Completeness หลักเกณฑ์ที่ 2) สามารถแปลงได้เป็น 5 คุณลักษณะดังนี้คือ Persistence, Secondary Storage Management, Concurrency, Recovery และ Ad Hoc Query Facility

จากหลักเกณฑ์ข้างต้นนั้นสามารถขยายความและแสดงตัวอย่างในการใช้งานกับข้อมูลประเภทมัลติมีเดีย(Multimedia) ซึ่งเป็นประเภทข้อมูลที่ OODBMS สนับสนุน ได้ดังนี้

4.2 หลักเกณฑ์ในการพิจารณาระบบที่มีพื้นฐานอยู่บน OODBMS

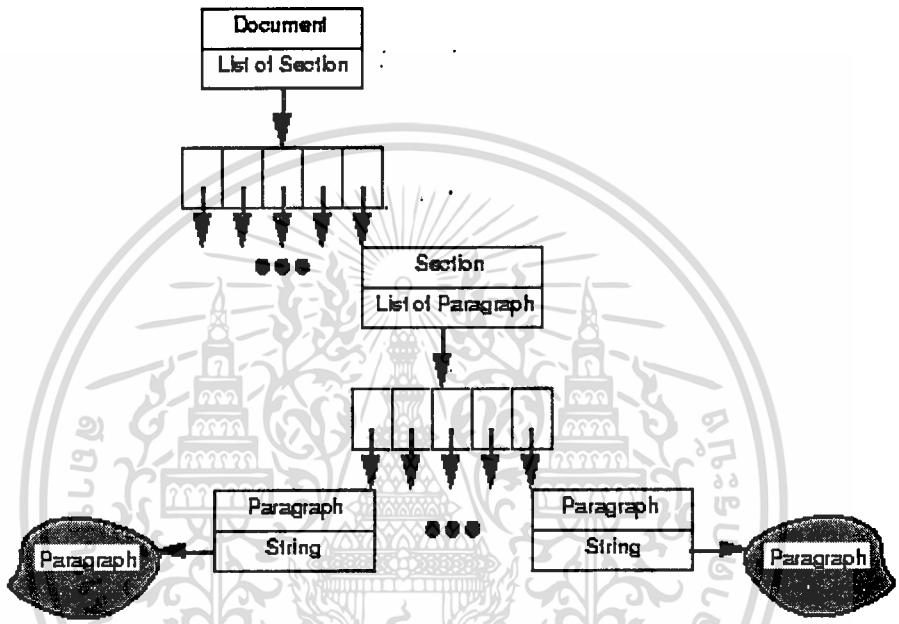
4.2.1 Complex Objects and Object identity

OODBMS ต้องมีความสามารถในการยอมให้ผู้ใช้สามารสร้างออบเจ็กต์ที่มีลักษณะซับซ้อนได้ ซึ่งเรียกว่า Complex Object ซึ่งจะรวมหรืออ้างอิงออบเจ็กต์อื่นๆ เพื่อสร้างโปรแกรมเกี่ยวกับออบเจ็กต์ เช่น เอกสารที่มีความซับซ้อนซ่อนอยู่ภายในโครงสร้าง นอกจากนี้แล้ว OODBMS ควรที่จะสนับสนุนคอนสตักเตอร์(Constructor) ที่จะใช้สร้าง Complex Object

Complex Object เป็นออบเจ็กต์ที่สร้างจากออบเจ็กต์ง่ายๆ โดยการใช้คอนสตักเตอร์ช่วยในการสร้าง สำหรับรูปแบบที่ง่ายที่สุดของออบเจ็กต์(บางครั้งเรียกว่า Atomic Type ซึ่งเป็นรูปแบบที่ไม่มีส่วนประกอบย่อยเลย) เช่น Integer, Character หรือ Boolean สำหรับรูปแบบของคอนสตักเตอร์ของ Complex Object นั้นจะรวมทูเปิล(Tuples), เซต(Sets), แบ็ก(Bags) (เป็นเซตที่อาจจะมีการซ้ำ

กันของสมาชิก), ลิสต์(Lists)และ อาร์เรย์(arrays) การกำหนดกลุ่มของคอนสตักเตอร์นั้นอย่างน้อย OODBMS ควรจะประกอบด้วยคอนสตักเตอร์ดังนี้คือเซต, ลิสต์ และทูเปิล

คอนสตักเตอร์เหล่านี้จะช่วยให้มีการสร้าง Complex Object ได้ เนื่องจากยอมให้มีการสร้างออบเจ็กต์ที่มีการอ้างถึงออบเจ็กต์อื่น สำหรับ Complex Object นั้นบ่อยครั้งจะพบในการประยุกต์ใช้ทางด้านมัลติมีเดีย ซึ่งตัวอย่างของ Complex Object ที่เป็นออบเจ็กต์เอกสารได้แสดงไว้ดังรูปที่ 4.2



รูปที่ 4.2 แสดงออบเจ็กต์เอกสารซึ่งเป็น Complex Object

จากรูปดังกล่าวสามารถอธิบายในลักษณะของอัลกอริธึม(Algorithm) ได้ดังต่อไปนี้

```
Create Type Document
Supertypes: TextObject
Properties:
      title: String;
      author: String;
      components: List of Section;
Operations:
      print;
      checkSpelling: String {errors};
      checkGrammar: String {errors};

Create Type Section
Supertypes: TextObject
Properties:
      title: String;
      number: Integer;
      components: List of Paragraph;
Operations:
      print;
      checkSpelling: String {errors};
      checkGrammar: String {errors};

Create Type Paragraph
Supertypes: TextObject
```

ออบเจ็กต์นั้นจะถูกกล่าวถึงโดยการใช้ Object Identity อย่างเป็นอิสระซึ่งจะตรงข้ามกับทูเปิลใน RDBMS ที่ต้องมีการเข้าถึงโดยการผ่านค่าของคีย์ที่เป็นส่วนหนึ่งของทูเปิลนั้นๆ โดยทั่วไปแล้ว Object Identity นั้นจะถูกสร้างขึ้นเองโดยระบบ ซึ่งจะมีลักษณะอยู่ในรูปของพอยน์เตอร์ เชิงวัตถุ(Object-Oriented Pointers) หรือบางครั้งอาจจะเรียกว่า Surrogates ลักษณะการจัดการดังกล่าวนี้สอดคล้องกับเรื่องของความเป็นหนึ่งเดียว(Uniqueness) และเรื่องของ Referential Integrity ที่จะถูกดูแลและจัดการในการใช้งานด้วย

หลักสำคัญในการใช้ Object Identifier ในการสร้าง Complex Object ซึ่งจะช่วยชี้หรืออ้างอิงไปยังส่วนประกอบอื่นๆ ของ Complex Object สำหรับตัวอย่างจากการอธิบายออบเจ็กต์เอกสาร ข้างต้นนั้น Identifier ที่ถูกใช้เพื่อแทนส่วนของเอกสาร Identifier ที่เป็นรายละเอียดของการใช้ในการแสดงถึงแนวคิดของการใช้โครงสร้างร่วมกัน ซึ่งจะมีความหมายถึงออบเจ็กต์ที่แตกต่างกัน 2 ออบเจ็กต์มีออบเจ็กต์ที่เป็นส่วนประกอบย่อย(Subcomponents) เหมือนกัน

4.2.2 Extensibility Using Abstract Types or Classes

OODBMS ต้องสนับสนุนรูปแบบของ Type หรือ Class ซึ่ง Type ก็คือการอธิบายถึงลักษณะทั่วไปของกลุ่มของออบเจ็กต์ซึ่งมีคุณลักษณะเหมือนๆ กัน ในระบบที่มีพื้นฐานแบบ Type ทั่วๆ ไปนั้น Type จะถูกใช้ในส่วนต้นๆ ของการคอมไพล์และไม่ได้เป็นคลาสแรกของออบเจ็กต์ หรือกล่าวได้ว่า Type ไม่ได้เป็นออบเจ็กต์นั่นเอง ทั้งนี้รูปแบบดังกล่าวจะไม่สามารถแก้ไขเปลี่ยนแปลงในช่วง Run Time ได้ สำหรับ Class นั้นมีลักษณะที่คล้ายคลึงกับ Type แต่สามารถทำงานในขณะ Run Time ได้มากกว่า ในระบบที่มีพื้นฐานแบบ Class ทั่วๆ ไปนั้น Class เป็นออบเจ็กต์ในตัวมันเองซึ่งจะคอยอำนวยความสะดวกในการสร้างออบเจ็กต์ใหม่ เช่นการส่งเมสเซจ "new" ไปยังคลาสของออบเจ็กต์ซึ่งการสร้างออบเจ็กต์ขึ้นมานี้จะเรียกว่า Extension ซึ่งเป็นกลุ่มของอินสแตนท์ของคลาสนั้นๆ ในฐานข้อมูล

การสนับสนุนแนวความคิดใดความคิดหนึ่งข้างต้นนั้นมีความจำเป็นมาก และเป็นมาตราฐานของ OODBMS ในการระบุหรือกำหนดออบเจ็กต์ขึ้นมาใช้งานในระบบฐานข้อมูล และในเรื่องของความสามารถในเรื่อง Extensible นั้นจะเกี่ยวกับการที่ผู้ใช้กำหนดรูปแบบขึ้นมาใหม่ซึ่งความต้องการดังกล่าวนี้มีการสนับสนุนโดยการใช้ Abstract Data Type หรือ ADT ในการช่วยสร้างออบเจ็กต์ใหม่ขึ้นมา

การใช้ ADT นั้นจะพาดพิงถึงแนวคิดในเรื่อง Encapsulation ส่วนที่เป็น Object State หรือแอตทริบิวต์และส่วนที่เป็นพฤติกรรม(Operation) ไว้ตามหลักการในระบบเชิงวัตถุ ในการติดต่อกับออบเจ็กต์นั้นจะติดต่อผ่านส่วนที่เป็น External Interface เท่านั้น สำหรับส่วนที่เป็น Internal

Interface นั้นไม่สามารถที่จะทำการติดต่อได้โดยตรง ซึ่งลักษณะดังกล่าวนี้จะช่วยในเรื่องของความยืดหยุ่นโดยรายละเอียดภายในของออบเจ็กต์นั้นสามารถเปลี่ยนแปลงโดยไม่มีผลกระทบต่อออบเจ็กต์อื่นๆ ที่ติดต่อเข้ามา

ในรูปแบบข้อมูลประเภทมัลติมีเดีย เช่นภาพ และเสียง เป็นตัวอย่างที่ดี ซึ่งสามารถแสดงและอ้างอิงถึงข้อมูลประเภทรูปภาพได้ดังนี้

```
Create Type Image
Supertypes: Object
Properties:
      name: String;
      owner: Person;
      creation: Date;
      lastModified: Date;
      encodingType: String;
      recordingAlgorithm: String;
      rawData: BLOB;
      description: Text;
      pixelWidth: Integer;
      pixelHeight: Integer;
      pixelDepth: Integer;
      colorMap: ColorMap;
      compressionAlgorithm: String;
Operations:
       display (VideoOutputDevice);
       compress (ratio: Real): Image;
       extractSubimage (subimage: Region): Image;
       overlay (Image): Image
       reverse (subimage: Region): Image
       dither (subimage: Region): Image
       magnify (factor: Integer): Image
       reduce (factor: Integer): Image
       match (Image): Boolean;
```

จากข้างต้น match() จะเป็นโอเปอร์เรชันเกี่ยวกับการตัดสินใจว่าภาพ I1 นั้นเหมือนกับภาพ I2 หรือไม่ซึ่งรายละเอียดของอัลกอริธึมที่ใช้นั้นจะถูกซ่อนด้วยการ Encapsulation

การกำหนด Computation Completeness ในภาษาการโปรแกรม เช่นความสามารถในการแสดงฟังก์ชั่นในการคำนวณต่างๆ ซึ่ง Computation Completeness นั้นหายากใน DML(Data Manipulation Language) ในระบบฐานข้อมูลที่ใช้กันอยู่ และ SQL ที่ใช้กันในปัจจุบันนี้ก็ไม่เป็น Computation Completeness แต่ใน OODBMS นั้นสนับสนุนให้มีการเชื่อมกับภาษาที่มีอยู่แล้ว ซึ่ง Computation Completeness นั้นต้องการเนื่องจากต้องใช้เกณฑ์ในการกำหนดพฤติกรรม(Operation หรือ Method) ของออบเจ็กต์ในระบบเมื่อมีการระบุประเภทของข้อมูลโดยผู้ใช้ โดยแสดงการกำหนดโอเปอร์เรชันของข้อมูลประเภทรูปภาพดังข้างต้น

4.2.3 Class หรือ Type Hierarchies

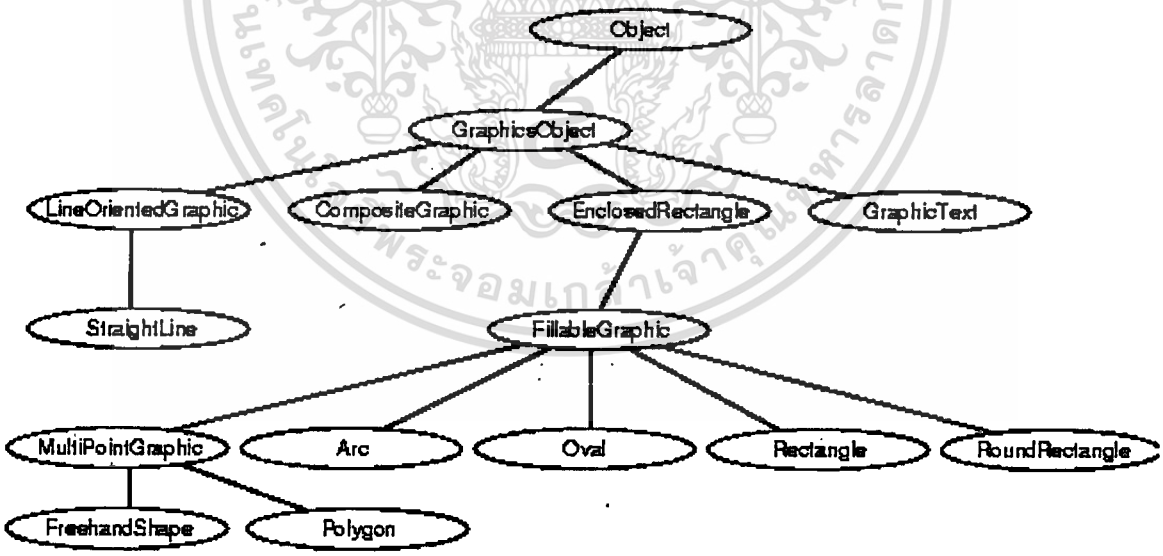
ODBMS ต้องสนับสนุนหลักการ Inheritance ซึ่งเป็นการที่คลาสหนึ่งมีการสืบทอดคุณสมบัติทุกอย่างมาจากอีกคลาสหนึ่งซึ่งคาสที่สืบทอดคุณสมบัติมานั้นสามารถมีคุณสมบัติอื่นเพิ่มเติม

นอกเหนือจากคลาสที่ถูกสืบทอดคุณสมบัติได้ ซึ่งหลักการ Inheritance นั้นเป็นแนวความคิดที่ดีและแสดงถึงรายละเอียดที่น่าสนใจในการกำหนดข้อจำกัดในการสืบทอดนั้นๆ ตัวอย่างเช่น การกำหนดคลาส Teenager ที่เป็น สับคลาสของคลาส Person ที่มีอายุระหว่าง 13 ถึง 19 ปี

ในกรณีของ Multiple Inheritance นั้นเป็นกรณีทั่วๆ ไปของการสืบทอดที่ยินยอมให้ออบเจ็กต์หนึ่งมีการสืบทอดคุณลักษณะและการกระทำจากคลาสที่มากกว่า 1 คลาส ตัวอย่างเช่น คลาส Employee และ คลาส Student ถูกกำหนดให้เป็น สับคลาสของคลาส Person ในการสร้างคลาสใหม่ เช่นการกำหนดคลาส Student-Employee ซึ่งออบเจ็กต์ในคลาสดังกล่าวจะแทนคนที่ทำงานไปด้วย เรียนไปด้วย โดยจะสืบทอดคุณลักษณะและการกระทำจากคลาส Student และคลาส Employee

จากหลักการ Inheritance จะพบว่ามีข้อดีที่ชัดเจนมากในเรื่องของการนำกลับมาใช้ใหม่ (Reusability) เพราะว่าทุกๆ โปรแกรมในแต่ละระดับของ Hierarchy สามารถใช้หลักเกณฑ์ดังกล่าวในการสืบทอดคุณลักษณะและการกระทำจากคลาสที่มีอยู่แล้วได้

Type Hierarchy เป็นส่วนที่สำคัญในการกำหนดรายละเอียดของ Complex Type ที่ใช้สำหรับข้อมูลประเภทมัลติมีเดีย ตัวอย่าง เช่น Hierarchy ของออบเจ็กต์เกี่ยวกับรูปภาพ โดยความสัมพันธ์ระหว่าง Type แสดงดังรูปที่ 4.3 และส่วนที่อธิบาย Type ต่างๆ เหล่านั้นแสดงดังรูปที่ 4.4



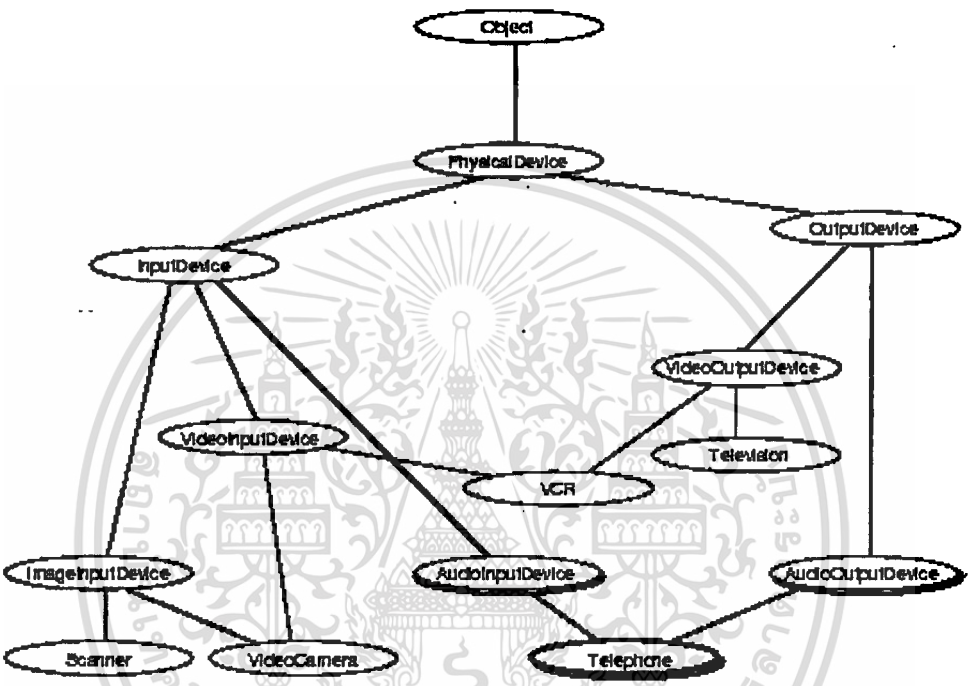
รูปที่ 4.3 แสดง Graphics Type Lattice

```
Create Type GraphicsObject
Supertypes: Object
Properties:
      startPoint: Point;
      color: String (Blue|Red|Magenta|Green|Cyan|Yellow|Black|White);
Operations:
      draw (Window);<Redefine in subtype>
      scale (percent: Integer);<Redefine in subtype>
      duplicate: GraphicsObject;
      rotate (degrees: Integer);<Redefine in subtype>
      flipHorizontal;<Redefine in subtype>
      flipVertical;<Redefine in subtype>
      moveToFront;
      moveToBack;
      moveForward;
      moveBackward;
      overlaps (GraphicsObject): Boolean;

Create Type CompositeGraphic
Supertypes: GraphicsObject
Properties:
      components: List of GraphicsObject;
Operations:
      draw (window1: Window)
            {
            for i=1 to components.numElements
                  components[i].draw (window1);
            }
      group (List of GraphicsObjects): CompositeGraphic;
      ungroup: List of GraphicsObjects;

Create Type EnclosedRectangle
Supertypes: GraphicsObject
Properties:
      relativeEndPt: Point;
Operations:

Create Type LineOrientedGraphic
Supertypes: GraphicsObject
Properties:
      thickness: Integer;
      linePattern: Bitmap;
      dashedLine: Bitmap;
Operations:

Create Type FillableGraphic
Supertypes: EnclosedRectangle
Properties:
      fillPattern: Bitmap
Operations:
```

รูปที่ 4.4 แสดงตัวอย่างการกำหนด Graphics Object Types

สำหรับตัวอย่างในกรณีของ Multiple Inheritance จะแสดงในส่วนที่เป็นแสดงส่วนของ PhysicalDevice โดยแสดงดังรูปที่ 4.5 ซึ่งแสดงถึงลักษณะการสืบทอดจาก Class หรือ Type ที่มากกว่าหนึ่ง เช่นในกรณีของ Type Telephone จะมีการสืบทอดคุณลักษณะจาก Type ของ AudioInputDevice และ AudioOutputDevice และส่วนที่อธิบาย Type ต่างๆ เหล่านั้น แสดงดังรูปที่ 4.6 ซึ่งจะพบว่ามีปัญหาซึ่งสามารถเกิดขึ้นในกรณีของ Multiple Inheritance เกี่ยวกับความไม่ชัดเจนของชื่อ ในกรณีนี้ Type Telephone จะสืบทอดคุณลักษณะ uperfreqresp, lowfreqresp และ

impedance จากทั้ง Type AudioInputDevice และ AudioOutputDevice ซึ่งเมื่อออบเจ็กต์ Telephone อ้างถึงคุณลักษณะดังกล่าวจะเกิดความสับสนและไม่ชัดเจน วิธีการหนึ่งที่เป็นมาตรฐานในแก้ปัญหาดังกล่าวก็คือการเปลี่ยนชื่อ (Renaming) ซึ่งแสดงไว้ในส่วนของการอธิบาย Type Telephone



รูปที่ 4.5 แสดง Portion of PhysicalDevice Type Lattice

```
Create Type PhysicalDevice
Supertypes: Object
Properties:
      make: String;
      model: String;
      serialNumber: String;
      purchasePrice: Dollars;
      purchaseDate: Date;
Operations:

Create Type AudioInputDevice
Supertypes: InputDevice
Properties:
      upperfreqresp: Hertzvalue;
      lowerfreqresp: Hertzvalue;
      impedance: Integer;
Operations:

Create Type AudioOutputDevice
Supertypes: OutputDevice
Properties:
      upperfreqresp: Hertzvalue;
      lowerfreqresp: Hertzvalue;
      impedance: Integer;
      sensitivity: dB;
Operations:

Create Type Telephone
Supertypes: AudioOutputDevice, AudioInputDevice
Properties:
      inputupperfreqresp: AudioInputDevice.upperfreqresp;
      inputlowerfreqresp: AudioInputDevice.lowerfreqresp;
      inputimpedance: AudioInputDevice.impedance;
      outputupperfreqresp: AudioOutputDevice.upperfreqresp;
      outputlowerfreqresp: AudioOutputDevice.lowerfreqresp;
      outputimpedance: AudioOutputDevice.impedance;
Operations:
      ring;
      redial;
      hold;
      answer;
      dial (phoneNumber: String);
```

รูปที่ 4.6 ตัวอย่างการกำหนด PhysicalDevice Object Types

4.2.4 Overriding, Overloading and Late Binding

OODBMS ต้องสนับสนุนโอเปอร์เรชัน Overloading และโอเปอร์เรชัน Overriding ซึ่งบางครั้งจะมีการอ้างอิงกับหลักการของ Polymorphism ซึ่งยอมให้มีการใช้ชื่อของโอเปอร์เรชันเหมือนกันโดยมีลักษณะการทำงานที่แตกต่างกัน ตัวอย่างเช่นการยอมให้มีการประยุกต์ใช้โอเปอร์เรชัน Display กับกลุ่มของออบเจ็กต์ที่เป็นกราฟิกหรือภาพประเภทต่างๆ ที่มีรูปแบบการจัดเก็บแตกต่างกัน โดยมีชื่อของโอเปอร์เรชันเหมือนกัน(ใช้หลักการ Inheritance ในการสืบทอดคุณลักษณะและการกระทำ) แต่กระทำกับประเภทของข้อมูลที่แตกต่างกันออกไป โดยใช้กลไก Run-Time Binding หรือ Late Binding ในการจัดการเกี่ยวกับการใช้โอเปอร์เรชันดังกล่าวให้ถูกต้องตามประเภทของข้อมูลที่เรียกใช้โอเปอร์เรชันนั้นๆ

จากตัวอย่างที่แสดงดังรูปที่ 4.4 ซึ่งจะอธิบาย Type CompositeGraphic โดยในกรณีนี้ถ้ากำหนดออบเจ็กต์ myCar เป็น instance ของ Type CompositeGrapic แล้วการเรียกใช้โอเปอร์เรชัน draw สามารถเขียนได้ดังนี้

```
myCar.draw (window1) ;
```

จากการเรียกใช้โอเปอร์เรชันดังกล่าว กลไก Late Binding จะทำการเรียกโอเปอร์เรชัน draw ใน Type CompositeGraphic ก่อน ถ้าไม่มีจะทำการหาโอเปอร์เรชันดังกล่าวในส่วนที่สูงขึ้นไปในที่นี้จะทำการค้นหาที่ Type GraphicsObject ต่อไป

```
draw (window1:Window)
    {
    for i=1 to components,numElements
       components[i].draw (window1) ;
    }
```

สำหรับโอเปอร์เรชัน Overloading และโอเปอร์เรชัน Overriding นั้นจะเป็นไปตามหลักการเชิงวัตถุทั่วไปซึ่งได้กล่าวไว้ในข้างต้นแล้ว

4.2.5 Persistence and Secondary Storage Management

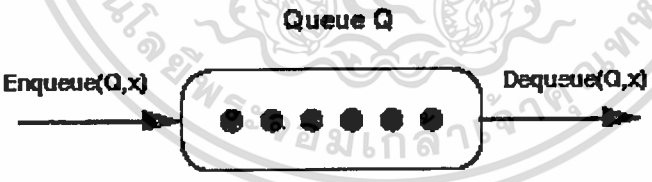
Persistence คือการยอมให้มีการคงอยู่ของออบเจ็กต์แม้ว่าการประมวลผลนั้นจะเสร็จสิ้นลงแล้ว การสนับสนุนการคงอยู่ของออบเจ็กต์เป็นความต้องการที่ชัดเจนของ OODBMS เพื่อความถูกต้องของระบบฐานข้อมูลในการอ้างอิงจากออบเจ็กต์อื่นในภายหลัง และ OODBMS ยังต้องการระบบที่จัดการกับฐานข้อมูลขนาดใหญ่ซึ่ง OODBMS จะต้องมีการสนับสนุนกลไกการจัดการเกี่ยวกับ Secondary Storage เช่น การจัดการ Index, การทำ Clustering, การทำ Buffering, การเข้าถึงจากเส้นทาง(Path) ที่เลือก และการทำ Query Optimization ซึ่งการจัดการดังกล่าวนี้ผู้ใช้ไม่สามารถทราบได้เลยว่ามีการจัดการอย่างไร

ความสามารถดังกล่าวเป็นส่วนสำคัญในการสนับสนุนให้ระบบฐานข้อมูลสามารถเก็บข้อมูลประเภทมัลติมีเดียได้ เช่น โปรแกรมประยุกต์ที่จัดการเกี่ยวกับเอกสารซึ่งอาจจะมีรูปแบบการจัดเก็บเป็นภาพที่เรียกว่า "Clip Art" ในระดับสูงนั้นการจัดการเกี่ยวกับ Secondary Storage จะเป็นส่วนสำคัญในการเพิ่มประสิทธิภาพการทำงานบนข้อมูลประเภทมัลติมีเดีย เช่น เสียง, ภาพ และตัวอักษร (ที่มีจำนวนข้อมูลมาก) ให้มีการทำงานที่ดีขึ้น

4.2.6 Concurrency and Recovery

OODBMS ต้องสนับสนุนการทำงานเกี่ยวกับการประมวลผลทรานส์แอ็กชัน(Transaction Processing) และกลไกในการควบคุมความถูกต้อง(Concurrency Control) ซึ่งมีความจำเป็นในเรื่องประสิทธิภาพและความถูกต้องของการประมวลผลจากการร้องขอที่เข้ามาในขณะนั้นๆ กลไกที่สำคัญอีกส่วนหนึ่งคือกลไกการเรียกคืน(Recovery) ซึ่งกลไกการเรียกคืนจะช่วยในการสนับสนุนการทำงานของกลไกควบคุมความถูกต้อง

ในการประมวลผลทรานส์แอ็กชัน, การควบคุมความถูกต้อง และกลไกการเรียกคืน ต้องสนับสนุนการทำงานในการติดต่อกับรูปแบบข้อมูลที่เป็นมัลติมีเดีย, ข้อมูลชั้นสูงหรือข้อมูลที่มีความซับซ้อนมากโดยจะใช้ข้อดีของการกำหนดเองโดยผู้ใช้(User-Define) ซึ่งสามารถกำหนดรูปแบบของข้อมูลได้เองในการพัฒนารูปแบบของความถูกต้อง ตัวอย่าง เช่น ในการที่ผู้ใช้กำหนดชนิดของข้อมูลแบบ Queue ที่มีการทำงานคือ Enqueue() และ Dequeue() ถ้ากำหนดให้ Q เป็นออบเจ็กต์ของชนิดข้อมูล Queue ดังนั้น Enqueue(Q,X) จะแทนสมาชิก X ที่เป็นข้อมูลเข้าในส่วนท้ายของคิว และ Dequeue(Q,Y) จะแทนการลบสมาชิก Y ที่เป็นข้อมูลออกในส่วนท้ายของคิว Enqueue() และ Dequeue() สามารถพิจารณาถึงกลไกการควบคุมความถูกต้องโดยการเขียนฟังก์ชันการทำงาน เช่นมีการ Lock ค่าของ Q ในการทำงานแต่ละครั้ง ซึ่งการทำงานของทรานส์แอ็กชัน ทั้ง Enqueue() และ Dequeue() ที่กระทำบน Q จะต้องรอให้ทรานส์แอ็กชันอื่นๆ เสร็จสมบูรณ์ก่อน ลักษณะของการทำงานใน Queue สามารถแสดงดังรูปที่ 4.7



รูปที่ 4.7 แสดง Operation on Queue

4.2.7 Ad hoc Query Facility

ในส่วนของ OODBMS ต้องสนับสนุนการใช้งานในเชิงวัตถุที่สอดคล้องกับภาษาที่ใช้ในการสอบถามข้อมูล เช่นเดียวกับภาษาที่ใช้งานในระบบฐานข้อมูลเชิงสัมพันธ์ อย่างไรก็ตามสิ่งที่อำนวยความสะดวกในการสอบถามควรจะมีคือ

- ภาษาระดับสูง(High-Level) คือสามารถเข้าใจได้ง่าย
- ประสิทธิภาพ(Efficient) คือในส่วนของ Interface มีการสนับสนุนมาจากการทำ Query Optimize
- ความเป็นอิสระของโปรแกรมประยุกต์(Application-Independent) คือความสามารถในการประยุกต์ใช้กับฐานข้อมูลอื่นๆ ที่เป็นไปได้

ในส่วนนี้มีมาตรฐานที่รองรับคือ SQL3 และ ODMG-93 ซึ่งใน SQL3 นั้นจะเป็นการสนับสนุนการทำงานบน ORDBMS มีลักษณะการทำงานที่เรียกว่า "SQL-Like" และสำหรับมาตรฐานของ ODMG-93 นั้นจะมีการกำหนดมาตรฐานของ OQL ซึ่งเป็นภาษาในการสอบถามข้อมูลที่สนับสนุนการทำงานบน OODBMS

OQL เป็นภาษาที่มีลักษณะคล้ายกับภาษา SQL โดย OQL ถูกใช้อธิบายการแอ็กเซสออบเจ็กต์ที่กำหนดด้วยรูปแบบตามมาตรฐานของ ODMG การสอบถามด้วย OQL สามารถอ้างถึง....ที่กำหนดในออบเจ็กต์โมเดลได้ รูปแบบทั่วไปในการสอบถามข้อมูลแสดงได้ดังนี้

Query Expression ::=

SELECT *constructor(p1,p2,...,pn)*

FROM *extent list*

WHERE *condition*

ในส่วนของ *constructot* จะแทนประเภทของข้อมูลที่ต้องการ ในส่วนของ *extent list* จะแทนออบเจ็กต์หนึ่งออบเจ็กต์หรือมากกว่า(ในกรณีมีการรวมออบเจ็กต์) และในส่วนของ *condition* จะเป็นส่วนที่ใช้ในการประเมินออบเจ็กต์ถ้าออบเจ็กต์เป็นไปตามเงื่อนไข ผลที่ได้จะถูกรวมกัน(กรณีมีการรวมออบเจ็กต์)และส่งกลับไปยังส่วนของ *constructor*

ในส่วนของ *condition* ที่กล่าวนั้นจะมีการกล่าวอยู่ในรูปแบบของบูลีน(Boolean) ซึ่งจะสร้างจากอ้างอิงแอตทริบิวต์และเมธอดที่จะเรียกใช้ ตัวอย่างในการใช้การสอบถามด้วย OQL แสดงได้ดังนี้

OQL Expression ::=

SELECT couple(student:x.name, professor:z.name)

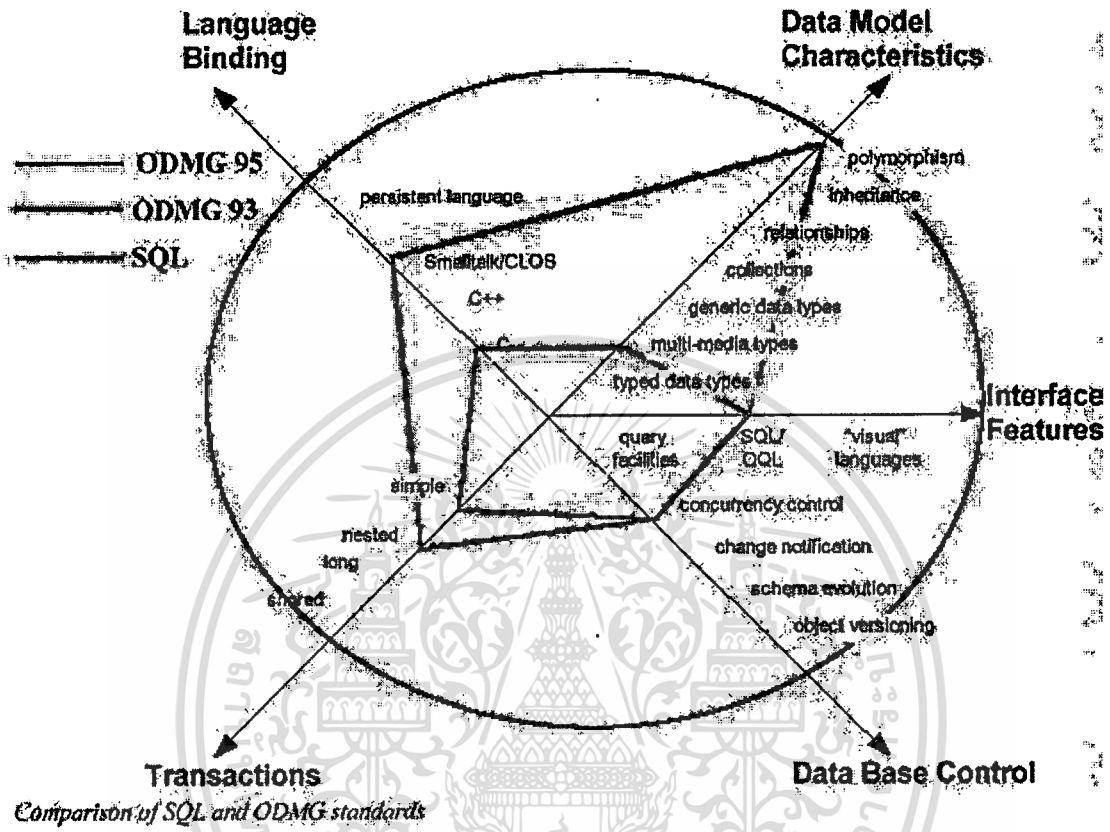
FROM x in Students,

y in x.takes,

z in y.taught

WHERE z.rank = &quotfull professor"

จากที่กล่าวมานั้นสามารถแสดงการเปรียบเทียบมาตรฐานของ SQL และ ODMG ได้ดังรูปที่ 4.8



รูปที่ 4.8 แสดงการเปรียบเทียบมาตรฐานของ SQL กับ ODMG

4.3 ข้อดีของระบบการจัดการฐานข้อมูลเชิงวัตถุ

ในส่วนนี้จะเป็นการกล่าวถึงข้อดีของระบบการจัดการฐานข้อมูลเชิงวัตถุที่นำเอาแนวคิดเชิงวัตถุมาประยุกต์ใช้ในระบบฐานข้อมูล ซึ่งข้อดีต่างๆ ที่จะกล่าวถึงในส่วนนี้เป็นเพียงบางส่วนขององค์ประกอบที่สนับสนุนในระบบการจัดการฐานข้อมูลเชิงวัตถุ

4.3.1 Extensibility

สำหรับ User-Defined Type และ Overloading Operation นั้นส่วนของคลาสและการจัดการความสัมพันธ์จะสามารถปรับเปลี่ยน หรือเพิ่มเติมได้ตามความต้องการของผู้ใช้ การเปลี่ยนแปลงหรือการเพิ่มนี้จะถูกจัดการให้กับคลาสที่มีความสัมพันธ์กันโดยอัตโนมัติ นั่นคือคลาส และ Operation ใหม่ๆ สามารถรวมเข้าไปในระบบโดยปราศจากผลกระทบกับวัตถุอื่นๆ ภายในระบบ

4.3.2 Information Hiding

การซ่อนข้อมูลวัตถุจะอนุญาติให้ข้อมูล และ Operation ถูกกำหนดอยู่ภายในโครงสร้างของมูลเดี่ยว(Single Model) นั่นคือ ข้อมูลจะถูกกำหนดเฉพาะที่(Local) และถูกซ่อนอยู่ภายในวัตถุเอง เมื่อมีการเชื่อมต่อเกิดขึ้นจะมีการแสดงออกมาโดยอัตโนมัติ

4.3.3 Flexibility of Type Definition

ความสามารถที่อนุญาตให้ผู้ใช้กำหนดคลาส และ Operation ใหม่จะให้ความยืดหยุ่นในการจัดการกับผู้ใช้ ซึ่งเป็นประโยชน์กับการประยุกต์ใช้ฐานข้อมูลใหม่ๆ ในการปรับเปลี่ยนข้อมูล

4.4.4 Code Reusability

จากคุณสมบัติการสืบทอดจากคลาสแม่ไปยังคลาสลูก ทำให้ไม่ต้องมีการกำหนดวิธีการและ Operation ขึ้นใหม่ในแต่ละคลาส ซึ่งจะช่วยลดความซ้ำซ้อน และการทำงานที่ไม่ควบคู่กัน

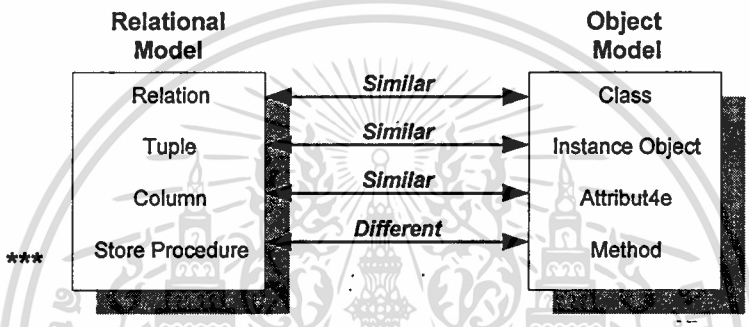
4.4 การเปรียบเทียบระหว่างระบบการจัดการฐานข้อมูลเชิงวัตถุกับระบบการจัดการฐานข้อมูลเชิงสัมพันธ์

ข้อเปรียบเทียบของระบบการจัดการฐานข้อมูลเชิงวัตถุกับระบบการจัดการฐานข้อมูลเชิงสัมพันธ์สามารถสรุปได้ 2 แนวทาง ดังนี้

4.4.1 การเปรียบเทียบแบบจำลองที่ใช้ในระบบการจัดการฐานข้อมูล

แนวทางนี้เป็นแนวทางการเปรียบเทียบแบบจำลองที่ใช้ในระบบการจัดการฐานข้อมูลเชิงวัตถุกับแบบจำลองที่ใช้ในระบบการจัดการฐานข้อมูลเชิงสัมพันธ์

จากที่กล่าวผ่านมาในบทที่ 2 และบทที่ 4 จะพบว่าแบบจำลองของระบบการจัดการฐานข้อมูลเชิงวัตถุ จะใช้แบบจำลองเชิงวัตถุ(Object Model) และแบบจำลองของระบบการจัดการฐานข้อมูลเชิงสัมพันธ์ จะใช้แบบจำลองเชิงสัมพันธ์(Relational Model) โดยสามารถแสดงการเปรียบเทียบได้ดังรูปที่ 4.9



รูปที่ 4.9 แสดงการเปรียบเทียบแบบจำลองระหว่างแบบจำลองเชิงสัมพันธ์กับแบบจำลองเชิงวัตถุ

4.4.2 การเปรียบเทียบแนวความคิดที่ใช้ในระบบการจัดการฐานข้อมูล

4.4.2.1 Data Type

ลักษณะของชนิดของข้อมูลที่ใช้กับระบบการจัดการฐานข้อมูลเชิงสัมพันธ์จะเป็นลักษณะที่เรียกว่า Built-in Data Type เช่น integer, real, string เป็นต้น ซึ่งลักษณะของข้อมูลเหล่านี้ระบบจะเป็นผู้ทำการกำหนดไว้ให้ แต่สำหรับระบบการจัดการฐานข้อมูลเชิงวัตถุชนิดของข้อมูลที่ใช้งานได้จะเป็นทั้ง Built-in Data Type และชนิดข้อมูลที่ผู้ใช้กำหนดขึ้นมาใช้เอง เช่นข้อมูลประเภท Complex Number เป็นต้น

4.4.2.2 Data Integrity

ในระบบการจัดการฐานข้อมูลเชิงวัตถุมีการทำงานลักษณะที่เรียกว่า Referential Integrity โดยการใช้การสื่อความหมายในการอ้างถึงโดยผ่าน Foreign key แต่ในทางตรงกันข้ามในระบบการจัดการฐานข้อมูลเชิงวัตถุ คลาสจะเป็นส่วนที่อธิบายถึง Data Abstraction และรวมไปถึงการกำหนด Specific ให้กับ Operation ต่างๆที่จะถูกประยุกต์ใช้กับ Instance ของคลาสนั้นๆ ซึ่งการกำหนดเช่นนี้จะทำให้ข้อมูลมีความเป็นอิสระต่อกันมากขึ้น นั่นคือ

เมื่อมีการเปลี่ยนแปลง หรือมีการพัฒนาระบบจะไม่ส่งผลกระทบต่อคลาส รวมทั้งกระบวนการทำงานอื่นๆ ซึ่งส่งผลให้ข้อมูลมีความถูกต้อง(Data Integrity)

4.4.2.3 Data Manipulation

สำหรับ Data Manipulation ในระบบการจัดการฐานข้อมูลเชิงสัมพันธ์นั้นมีหลักการอยู่บน Relational Algebra หรือ Relational Calculus โดยหลักการดังกล่าวที่ใช้กันในปัจจุบันคือ SQL-92 ซึ่งเป็นมาตรฐานบนระบบการจัดการฐานข้อมูลเชิงสัมพันธ์ แต่ในระบบการจัดการฐานข้อมูลเชิงวัตถุยังมีมาตรฐานไม่แน่นอน โดยมีมาตรฐานหลักๆ ที่ใช้กันในปัจจุบันคือ มาตรฐานของ ODMG-93, มาตรฐานของ SQL3 และมาตรฐานของ OMG Standards ซึ่งในแต่ละมาตรฐานจะมีหลักการเชิงวัตถุที่คล้ายคลึงกัน แต่ยังมีรายละเอียดของหลักการบางส่วนของที่แตกต่างกันออกไป

บทที่ 5

แนะนำระบบการจัดการฐานข้อมูลเชิงวัตถุ Jasmine

ระบบการจัดการฐานข้อมูลเชิงวัตถุเป็นระบบฐานข้อมูลที่นำเอาแนวคิดเชิงวัตถุมาใช้ในงานระบบฐานข้อมูลเพื่อขยายขีดความสามารถของระบบการจัดการฐานข้อมูลในปัจจุบัน ทั้งนี้เนื่องจากความต้องการใช้งานข้อมูลที่มีความซับซ้อนในงานระบบสารสนเทศปัจจุบันมีเพิ่มมากขึ้นเช่น มีความต้องการใช้ข้อมูลประเภทมัลติมีเดียในงานระบบฐานข้อมูล เป็นต้น โดยในบทนี้จะเป็นการแนะนำระบบการจัดการฐานข้อมูลเชิงวัตถุ Jasmine ของ Computer Associates ที่ร่วมพัฒนากับบริษัท Fujitsu ในประเทศญี่ปุ่น ในแง่มุมต่างๆ

5.1 ระบบการจัดการฐานข้อมูลเชิงวัตถุ Jasmine

Jasmine เป็นระบบการจัดการฐานข้อมูลเชิงวัตถุที่มีความสมบูรณ์พร้อมที่ผู้ใช้หรือผู้พัฒนาต้องการในการพัฒนาโปรแกรมประยุกต์ในแต่ละ Phase ตั้งแต่ระดับต้นแบบ(Prototype) ไปจนกระทั่งระดับผลิตภัณฑ์ โดยผู้ใช้หรือผู้พัฒนาสามารถนำเสนอข้อมูลทางธุรกิจในรูปแบบของมัลติมีเดีย(Multimedia) รวมทั้งการสร้าง Business Logic หรือกระทั่งการเก็บข้อมูลประเภท Complex Data ในระบบฐานข้อมูล นอกจานี้ในระบบการจัดการฐานข้อมูลเชิงวัตถุ Jasmine นั้นผู้ใช้สามารถสร้างฐานข้อมูลเชิงวัตถุเพียงครั้งเดียวและสามารถนำกลับมาใช้ได้ใหม่หรือมีความสามารถที่เรียกว่า "Reusability" ซึ่งผู้ใช้สามารถสร้างออบเจ็กต์ใหม่ได้อย่างรวดเร็วและสามารถกระจาย(Distributed)โปรแกรมประยุกต์ที่มีลักษณะซับซ้อนโดยไม่จำเป็นต้องใช้ประสบการณ์ในการจัดการโปรแกรมประยุกต์ดังกล่าวมากนัก

5.2 องค์ประกอบใน Jasmine (Jasmine Feature)

Jasmine Object Database System (ODB) มีองค์ประกอบมากมายซึ่งทำให้ง่ายต่อการพัฒนาของผู้พัฒนาหรือโปรแกรมเมอร์ในการพัฒนาโปรแกรมประยุกต์เชิงวัตถุ ทั้งในเรื่องของการสร้างและการดูแลรักษาฐานข้อมูลเชิงวัตถุ โดยในส่วนต่อไปจะกล่าวถึงองค์ประกอบต่างๆ เหล่านั้น

5.2.1 Object-Oriented System

ระบบการจัดการฐานข้อมูลเชิงวัตถุ Jasmine ได้รวบรวมลักษณะต่างๆ ที่จำเป็นในระบบฐานข้อมูลเชิงวัตถุเช่น การกำหนดสิ่งต่างๆ ในระบบ, การจัดการการทำงานของระบบและการควบคุมความถูกต้องในการแอ็กเซสคลาสหรือฐานข้อมูลเชิงวัตถุ ซึ่งในระบบการจัดการฐานข้อมูลเชิงวัตถุ Jasmine ดังกล่าวมีองค์ประกอบที่ใช้สนับสนุนข้อมูลที่มีโครงสร้างซับซ้อน(Complex data structure) และข้อมูลที่มีขนาดใหญ่ตาม Type ของข้อมูลนั้นๆ อีกทั้งยังมีกลไกสนับสนุนการ Inheritance และ Multiple Inheritance นอกจากนี้และยังสนับสนุน Property และ Method ในระดับ Instance และ Class-Level รวมทั้งระดับ Collection-Level ด้วย

ในส่วนของ Object Database Query Language (ODQL) ซึ่งองค์ประกอบที่สนับสนุนการเข้าถึงฐานข้อมูลได้อย่างสมบูรณ์ โดยจะรวมการกำหนดข้อมูล, การเข้าถึงข้อมูลดังกล่าวและการเขียนเมธอด โดยในภาษา ODQL จะยอมให้ผู้ใช้สามารถกำหนดและคอมไพล์คลาสกับเมธอด, การกระทำของเมธอดที่กำหนดในฐานข้อมูล, การ Execute Query ต่างๆ ที่กระทำกับฐานข้อมูล รวมทั้งการสร้างและ Update ฐานข้อมูลเชิงวัตถุ และนอกจากนี้เมื่อมีการเขียนโปรแกรมประยุกต์ที่ทำการติดต่อกับระบบการจัดการฐานข้อมูลเชิงวัตถุ Jasmine ผู้ใช้หรือผู้พัฒนาสามารถใช้ภาษา ODQL ได้โดยตรงโดยการ Embedded โค้ด ODQL ลงในโปรแกรมภาษา C หรือ C++ หรือทำการ Execute โค้ด ODQL ผ่าน Jasmine C API

5.2.2 Database Integration

Jasmine ได้ผนวก(Integration) เอาความสามารถต่างๆ ในการสนับสนุนฐานข้อมูลรูปแบบต่างๆ มาไว้อย่างสมบูรณ์ โดยมีองค์ประกอบที่สนับสนุนฐานข้อมูลเชิงสัมพันธ์ เช่น OpenIngres, Oracle, Sybase, Informix, and SQLServer เป็นต้นและนอกจากนี้ยังสนับสนุนฐานข้อมูลในระบบ Mainframe เช่น DB2 เป็นต้น

5.2.3 Multimedia Support

Jasmine ได้จัดเตรียมส่วนของ Multimedia Class Library เพิ่มเติมเพื่อสนับสนุนข้อมูลประเภทมัลติมีเดียต่างๆ เช่น ภาพ, ลำดับเหตุการณ์ของเฟรมภาพเคลื่อนไหว, รูปแบบของ Audio และ Video File ต่างๆ , Rich Text Format (RTF) File และ Page Layout ซึ่งการใช้งาน Multimedia Class Library นั้น ช่วยให้ง่ายในการจัดการส่วน

ประกอบที่เป็นมัลติมีเดียซึ่งเป็น Information ที่มีความนิยมใช้ในโปรแกรมประยุกต์ปัจจุบัน

5.2.4 Distributed Client/Server Architecture

ในโปรแกรมประยุกต์ Jasmine จะมีการทำงาน(Execute) อยู่ทางฝั่ง Client Workstation ซึ่งมีหลักการทำงานอยู่บนสถาปัตยกรรมแบบ Client/Server รวมทั้งยังสนับสนุนการใช้ Plug-ins ใน Web Browser ต่างๆ เช่น Netscape Navigator หรือ Microsoft Internet Explorer เพื่อให้โปรแกรมประยุกต์แบบ Web Enabled ทำการติดต่อผ่านระบบเครือข่าย Internet ไปยังระบบการจัดการฐานข้อมูลเชิงวัตถุ Jasmine ซึ่งทำหน้าที่เป็น Server คอย Execute ตัว Business Logic และจัดเตรียมพื้นที่สำหรับออบเจ็กต์รวมทั้งข้อมูลของโปรแกรมประยุกต์นั้นๆ

5.2.5 Internet-and Intranet Enabled

- โปรแกรมประยุกต์ Client/Server บน Jasmine สามารถติดต่อกับ Jasmine Server ผ่านโพรโตคอลที่ใช้ในการติดต่อสื่อสารมากมายที่มีในปัจจุบัน
- โปรแกรมประยุกต์ Web-Enabled บน Jasmine ใช้โพรโตคอล HTTP ในการติดต่อสื่อสารกับ Jasmine Server ซึ่งจะช่วยเพิ่มประสิทธิภาพในการ Download ข้อมูลต่างๆ
- ในส่วนของ WebLink Tool จะมีองค์ประกอบที่ใช้ในการแอ็กเซสฐานข้อมูล Jasmine ข้ามระบบเครือข่าย Internet โดยการใช้ HTML Page ซึ่งผู้ใช้สามารถใช้ Jasmine พัฒนาการใช้งานทางด้าน Internet หรือ intranet ภายในองค์กรได้อย่างกว้างขวาง

5.2.6 Integrated Application Development Environment

Jasmine มีองค์ประกอบที่ใช้ในการพัฒนา 2 รูปแบบคือการพัฒนาแบบเปิด(Open) โดยใช้ Application Programming Interface (API) และการพัฒนาด้วย Jasmine Studio ซึ่งเป็นการพัฒนาโปรแกรมประยุกต์ในแบบ Windows-based โดยจะมี Tool ที่อนุญาตให้ผู้ใช้พัฒนาโปรแกรมประยุกต์แบบ Client/Server ทางด้านมัลติมีเดียหรือโปรแกรมประยุกต์ในรูปแบบของ Web Browser Plug-ins

- Jasmine C API มีองค์ประกอบที่ใช้ในการแอ็กเซสผ่านโปรแกรมภาษา C/C++ และระบบการพัฒนาโปรแกรมประยุกต์ต่างๆ ที่สนับสนุนการติดต่อกับ Dynamic Link Libraries (DLLs) รวมทั้ง ActiveX Control ที่เตรียมไว้ให้ผู้ใช้ติดต่อกับระบบการจัดการฐานข้อมูลเชิงวัตถุ Jasmine ผ่านทางโปรแกรมประยุกต์ที่พัฒนาขึ้นด้วย Visual Basic และ Tool อื่นๆ ที่สนับสนุน ActiveX ในการพัฒนา

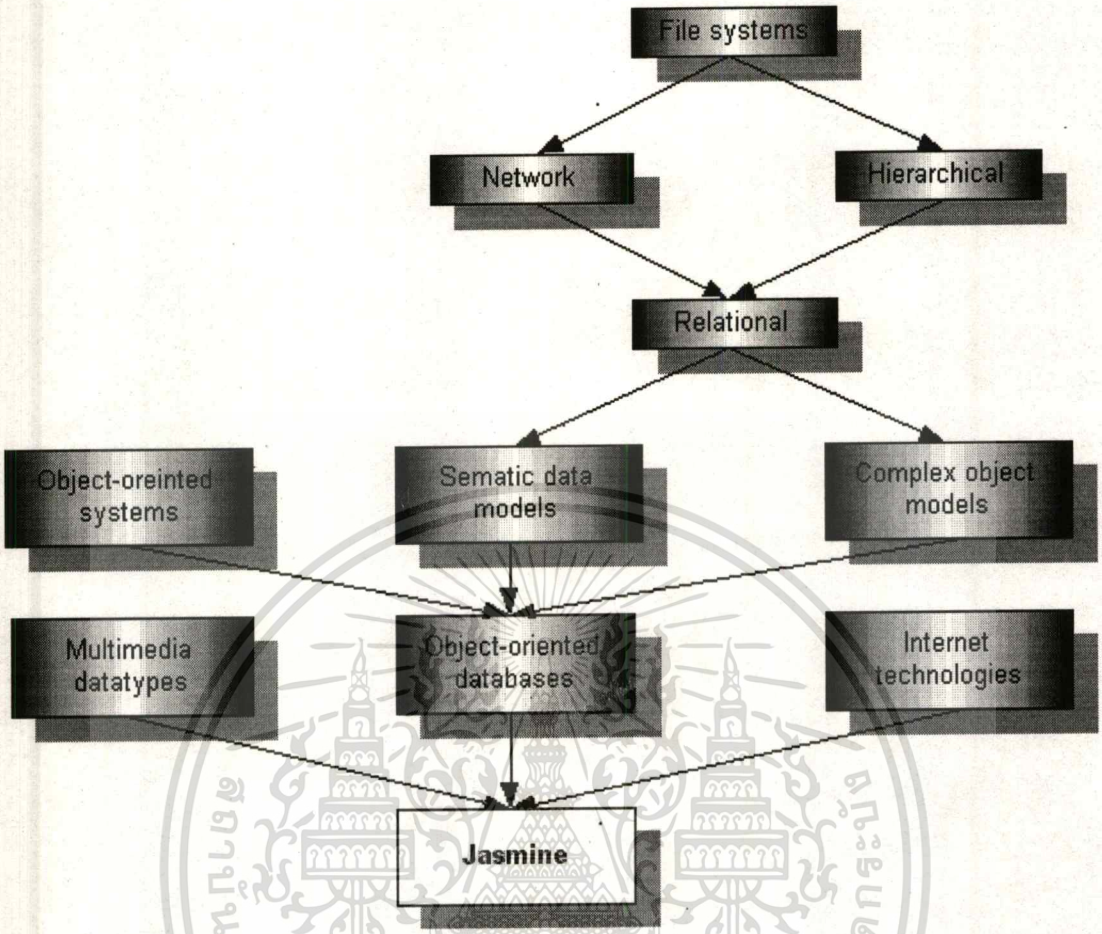
  ในการโปรแกรมด้วยภาษา C API จะสนับสนุนการใช้ภาษา ODQL ที่เตรียมไว้เพื่อใช้ในการแอ็กเซสฐานข้อมูลรวมทั้งยังมีองค์ประกอบเกี่ยวกับฟังก์ชันสำหรับการกำหนดหรือสร้างออบเจ็กต์, การ Query ออบเจ็กต์ และ การ Manipulation ออบเจ็กต์ สำหรับในภาษา Host นั้นอนุญาตให้ผู้ใช้กระทำกับ non-Database-Related Computation หรือ Computation ที่มีการอ้างอิงข้อมูลภายนอก(External Data)

- Jasmine Studio มีองค์ประกอบต่างๆ ที่อนุญาตให้ผู้ใช้ทำการออกแบบโปรแกรมประยุกต์, ทำต้นแบบ(Prototype), ทำการ Debug และ Deploy โปรแกรมประยุกต์ที่สร้างบน Jasmine Studio และยังมีองค์ประกอบที่สนับสนุนการจัดการคลาสทั้งหมดของผู้ใช้ในฐานข้อมูลโดยวิธีการ Drag-and-Drop ซึ่งมีความสะดวกและง่ายในการจัดการ

5.2.7 Complete Data Management

นอกเหนือจากองค์ประกอบที่ใช้ในการพัฒนาโปรแกรมประยุกต์ต่างๆ ที่กล่าวมาแล้วนั้น Jasmine ยังมีการจัดเตรียมกลุ่มของ Service ที่ใช้ในการจัดการฐานข้อมูลที่สมบูรณ์พร้อมดังต่อไปนี้

- Client/Server computing
- Locking and concurrency control
- Transaction management
- Recovery and restart
- Access control
- Database administration



รูปที่ 5.1 แสดงการวิวัฒนาการ(Evolution) ของระบบฐานข้อมูลและ Jasmine

จากรูปที่ 5.1 ข้างต้นนั้นจะเป็นการแสดงถึงวิวัฒนาการของระบบฐานข้อมูลและการผนวกเอาความสามารถต่างๆ ดังกล่าวไว้ในระบบการจัดการฐานข้อมูลเชิงวัตถุ Jasmine

และในส่วนต่อไปจะเป็นการกล่าวถึงการจัดการฐานข้อมูลเชิงวัตถุด้วย Jasmine Studio ซึ่งเป็น Tool ที่มีมาพร้อมกับระบบการจัดการฐานข้อมูลเชิงวัตถุ Jasmine

5.3 Database Administration with Jasmine Studio

Jasmine Studio ได้จัดเตรียมชุดของเครื่องมือที่ใช้ในการทำงานและติดต่อกับระบบการจัดการฐานข้อมูลเชิงวัตถุ Jasmine ในแบบ GUI ซึ่งเครื่องมือดังกล่าวนี้เหมาะสำหรับผู้ใช้ที่ไม่คุ้นเคยกับการใช้ภาษา ODQL และการเขียนโปรแกรมประยุกต์เพื่อทำการติดต่อกับระบบการจัดการฐานข้อมูลเชิงวัตถุ Jasmine โดยฟังก์ชันการทำงานที่สนับสนุนการจัดการต่างๆ ของ Jasmine Studio จะ

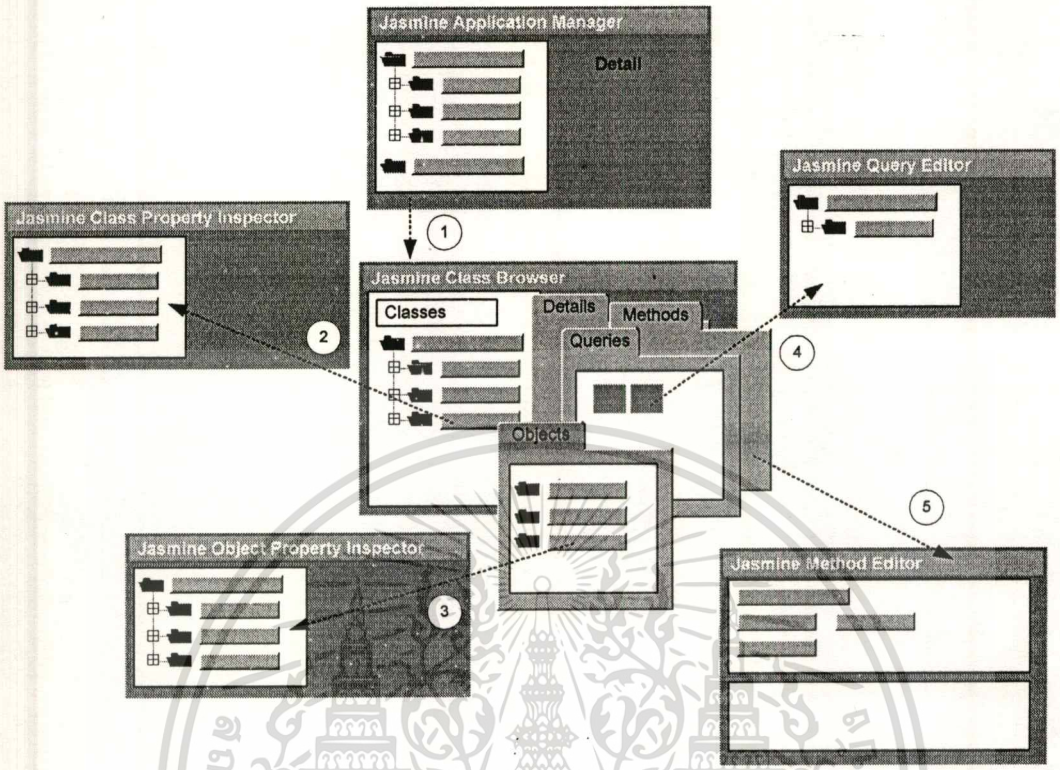
รวมเรื่องของการ Viewing และการทำการเปลี่ยนแปลงทั้งใน Database Schema และ Database Content ทั้งนี้ฟังก์ชันการทำงานที่สำคัญ ซึ่งที่สนับสนุนการทำงานใน Jasmine Studio มีดังนี้

- การ Browse และการพิมพ์คลาสของฐานข้อมูลและ Class Hierarchies
- การเพิ่มคลาสใหม่(การกำหนดและการสร้าง)
- การเปลี่ยนแปลงคลาสที่กำหนดโดยการเพิ่มหรือลบ Property หรือเมธอดของคลาสนั้นๆ
- การเพิ่มออบเจ็กต์ใหม่ให้กับคลาสนั้นๆ (การ Populate ฐานข้อมูล)
- การลบออบเจ็กต์จากฐานข้อมูล
- การเปลี่ยนแปลงค่า Property ของออบเจ็กต์ที่มีอยู่ในฐานข้อมูล
- การกำหนด Database Query ที่ใช้ในโปรแกรมประยุกต์ที่สร้างขึ้นจาก Jasmine Studio

ในการสนับสนุนฟังก์ชันการทำงานหรือ Operation ที่ใช้ในการจัดการฐานข้อมูลดังกล่าวแล้ว Jasmine Studio ยังได้จัดเตรียมหน้าต่างที่ใช้ในการทำงานดังกล่าวไว้มากมายเพื่อสนับสนุนองค์ประกอบต่างๆ ของการทำงานดังกล่าวในการแสดงและการ Manipulate ในแต่ละ Type ที่ระบุของ Item ในฐานข้อมูล ซึ่ง หน้าต่างเหล่านั้นจะประกอบด้วย

- Class Browser
- Class Property Inspector
- Object Property Inspector
- Query Editor
- Method Editor

นอกจากนี้ยังสนับสนุน Multiple Windows ของแต่ละ Type ซึ่งอาจจะถูกเปิดขึ้นพร้อมๆ กัน ทั้งนี้การทำความเข้าใจเกี่ยวกับหน้าที่ของแต่ละหน้าต่างรวมทั้งความสัมพันธ์ของแต่ละหน้าต่างสามารถแสดงได้ดังรูปที่ 5.2 โดยจะแสดงให้เห็นถึงหน้าต่างหลักๆ และความสัมพันธ์ระหว่างหน้าต่างนั้นๆ



รูปที่ 5.2 แสดง Jasmine Studio's Database Administration Windows

จากรูปดังกล่าวแสดงหน้าต่าง Application Manager ซึ่งเป็นหน้าต่างหลักของ Jasmine Studio โดยจะถูกเปิดขึ้นเมื่อมีการเริ่มใช้ Jasmine Studio ทั้งนี้ใน Application Manager จะแสดงถึงโปรแกรมประยุกต์ต่างๆ ที่ติดต่อกับฐานข้อมูลในปัจจุบัน และจากหน้าต่าง Application Manager ดังกล่าวนั้นผู้ใช้หรือผู้พัฒนาสามารถเชื่อมการใช้งานหน้าต่างต่างๆ ใน Jasmine Studio ซึ่งแสดงให้เห็นในรูปที่ 5.2

1. เมื่อกดปุ่ม Database Administration ที่ Toolbar ของ Application Manager หรือเลือก Database Administration จาก File Menu เพื่อเปิดหน้าต่าง Class Browser
2. Double-Clicks คลาสใดๆ ใน Class Hierarchy ที่แสดงในหน้าต่าง Class Browser เพื่อเปิดหน้าต่าง Class Property Inspector เพื่อจะแสดงรายละเอียดเกี่ยวกับ Property ของคลาสที่ถูกเลือกนั้นๆ

3. Double-Clicks ออบเจ็กต์ที่แสดงใน Object Tab ทางด้านขวามือของ Class Browser เพื่อเปิดหน้าต่าง Object Property Inspector เพื่อจะแสดงค่า Property ของออบเจ็กต์ที่ถูกเลือกนั้นๆ
4. Double-Clicks Query ใน Query Tab ทางด้านขวามือของ Class Browser เพื่อเปิดหน้าต่างของ Query Editor เพื่อแสดง Query ในคลาสนั้นๆ
5. Double-Clicks เมธอดใน Method Tab ทางด้านขวามือของ Class Browser เพื่อเปิดหน้าต่างของ Method Editor เพื่อแสดงรายละเอียดของ Method ใน Class นั้นๆ

บทที่ 6
แนวทางในการแก้ไขปัญหา

จากที่กล่าวมาในบทก่อนหน้านี้จะพบว่าปัจจุบันลักษณะงานในระบบสารสนเทศได้มีการขยายขอบเขตการใช้งานเทคโนโลยีทางด้านฐานข้อมูลในการจัดการกับข้อมูลที่มีความซับซ้อน เช่น ข้อมูลประเภทภาพ, เสียง หรือข้อมูลประเภทมัลติมีเดีย ซึ่งมีความต้องการใช้งานข้อมูลลักษณะนี้เพิ่มมากขึ้น ทั้งนี้ความต้องการในการจัดการกับข้อมูลลักษณะดังกล่าวนั้น ส่งผลให้การจัดการข้อมูลด้วยระบบการจัดการฐานข้อมูลเชิงสัมพันธ์ไม่เพียงพอและอาจจะไม่เหมาะสม เนื่องมาจาก

1. Data Type ที่มีใน RDBMS ไม่เพียงพอกับความต้องการ
2. Operator ที่สนับสนุนใน RDBMS ไม่เพียงพอกับความต้องการ
3. RDBMS ขาด Built-in function ในการจัดการข้อมูลที่มีความซับซ้อน
4. RDBMS ขาดความสามารถในการนำเสนอ Structure
5. การจัดการ Integrity rules ที่กระทำโดย procedure ซึ่งเป็นคนละส่วนกับ Data Structure และ,
6. การจัดการ Identifier keys ที่กระทำโดย User

(หมายเหตุ : ปัญหาเรื่องการจัดการ Identifier keys จะไม่ขอกล่าวถึงในงานวิจัยครั้งนี้)

การศึกษาในบทนี้จะเป็นการกล่าวถึงแนวทางในการแก้ไขปัญหาด้วยแนวความคิดเชิงวัตถุที่สนับสนุนในระบบการจัดการฐานข้อมูลเชิงสัมพันธ์ และแนวทางการแก้ไขปัญหาดังกล่าวด้วยระบบจัดการฐานข้อมูลเชิงวัตถุ Jasmine

6.1 แนวทางในการแก้ไขปัญหาด้วยแนวความคิด Object-Oriented Database

จากแนวความคิดเชิงวัตถุที่สนับสนุนในระบบการจัดการฐานข้อมูลเชิงวัตถุ(กล่าวไว้ในบทที่ 3 และบทที่ 4) สามารถใช้เป็นแนวทางในการแก้ปัญหาหรือความไม่เหมาะสมที่เกิดขึ้นบนระบบการจัดการฐานข้อมูลเชิงสัมพันธ์ในงานระบบสารสนเทศปัจจุบันได้ดังนี้

6.1.1 แนวทางในการแก้ไขปัญหาเรื่อง Data Type, Operator, Built-in function และ Integrity rules ใน RDBMS (ปัญหาที่ 1, 2, 3 และ ปัญหาที่ 5)

จากปัญหาเรื่อง Data Type, Operator, Built-in function และ Integrity rules ใน RDBMS นั้นสามารถแก้ไขได้โดยใช้ Abstraction Mechanisms หรือ Abstract Data Type (ADT) ซึ่งเป็นคุณลักษณะของโครงสร้างออบเจ็กต์(Object Structure) ที่สนับสนุนใน OODBMS โดย data abstraction นั้นผู้ใช้สามารถกำหนด Type ของ Data Object ขึ้นมาใหม่ซึ่งเห็นว่า Type ที่สร้างขึ้นมาใหม่นั้นเป็นประโยชน์ในการแก้ปัญหาในแต่ละระดับของการใช้ รวมทั้งการกำหนด Operation ต่างๆ ในการใช้งานสำหรับ Type ที่สร้างขึ้นมาใหม่นั้น ทั้งนี้การนิยาม ADT นั้นจะมีการอ้างอิงกลไกการ Encapsulation ซึ่งกลุ่มของแอตทริบิวต์ (อาจเป็นออบเจ็กต์หรือกลุ่มของ Instance Variable) และกลุ่มของ Operations จะถูก Encapsulate ไว้ในออบเจ็กต์เดียวกัน ซึ่งในการใช้งานโปรแกรมเมอร์จะสนใจเพียงพฤติกรรมของ Data Object ว่า Information ใดที่จะส่งเข้าไปยังออบเจ็กต์นั้นๆ และจะได้ผลลัพธ์ในรูปแบบใด โดยโปรแกรมเมอร์ไม่จำเป็นต้องสนใจว่า Data Object นั้นจะมีการทำงานอย่างไร

ในส่วนของพฤติกรรมของ Data Object นั้นจะถูกกล่าวถึงให้เหมือนธรรมชาติมากที่สุด โดยแสดงในเทอมของกลุ่ม Operation ซึ่งแสดงความหมายของออบเจ็กต์เหล่านั้น ซึ่งเซตดังกล่าวจะรวม Operation ในการสร้างออบเจ็กต์, การได้รับ Information จากออบเจ็กต์เหล่านั้น และอาจจะมีการเปลี่ยนแปลงแก้ไขออบเจ็กต์เหล่านั้นด้วย ตัวอย่างเช่น ***Pop และ Push เป็น Operation ของ Stack หรือ Operation ของ Integer คือ Operation ทาง Arithmetic เป็นต้น (ซึ่งอาจมีการกำหนด Operation เพิ่มเติมเช่น Remainder และ Quotient ตามความเหมาะสมของงานนั้นๆ)*** ดังนั้น Data Abstraction จะประกอบด้วยกลุ่มของออบเจ็กต์และกลุ่มของ Operation ซึ่งอธิบายถึงลักษณะ พฤติกรรมของออบเจ็กต์เหล่านั้น

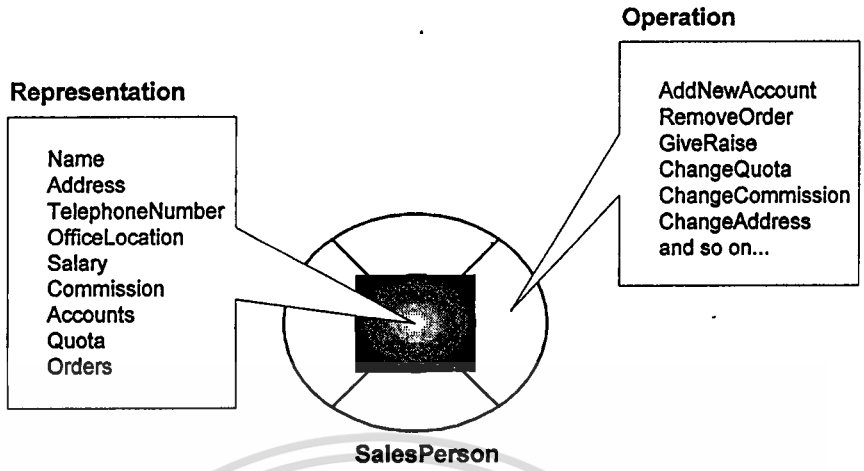
ทั้งนี้ยังรวมถึงการ Manipulation ของ Instance ของ Data Type ซึ่งจะถูกกระทำผ่าน Operation ที่มีความสัมพันธ์กับ Data Type เท่านั้น ADT จะมีการเตรียมกลไกซึ่งมีการแบ่งแยกอย่างชัดเจนระหว่างส่วนที่เป็น Interface และส่วนของอิมพลีเมนท์ของ Data Type

การอิมพลีเมนท์ของ ADT จะประกอบด้วย

- ส่วนของ Representation ซึ่งก็คือส่วนของโครงสร้างข้อมูล
- ส่วนของ Operation ซึ่งก็คือส่วนของอัลกอริธึม

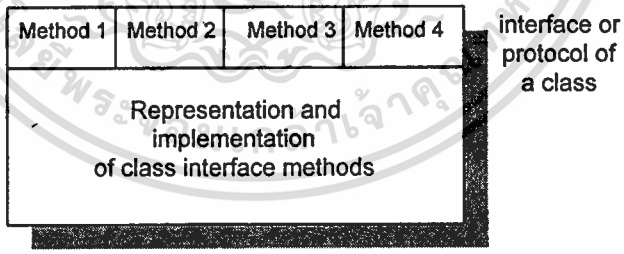
การติดต่อกับ ADT นั้นจะกระทำผ่าน Operation ที่สัมพันธ์กับ ADT นั้นๆ

ตัวอย่าง การ Definition ของ ADT SalesPerson



รูปที่ 6.1 แสดงการกำหนด ADT SalesPerson

ลักษณะจริงๆ ของ Data Structure ในส่วน Representation ของ ADT นั้นจะไม่สามารถมองเห็นได้จากผู้ใช้และในส่วนของอัลกอริธึมที่ใช้อิมพลีเมนท์แต่ละ Operation ใน ADT นั้นๆ จะถูก Encapsulate ไว้ภายใน ADT นั้น ทั้งนี้คุณลักษณะของ Information-Hiding หรือ Encapsulation นั้นจะประกอบด้วยส่วน Public ซึ่งทำหน้าที่เป็นส่วน Interface และส่วน Private ซึ่งซ่อนรายละเอียดต่างๆ ของการ Representation และส่วนของการอิมพลีเมนท์ไว้จากผู้ใช้ โดยแสดงได้ดังรูปที่ 6.2



รูปที่ 6.2 แสดงกลไกการ Encapsulation หรือ Information hiding

จากแนวคิดเชิงวัตถุที่สนับสนุนในระบบการจัดการฐานข้อมูลเชิงวัตถุดังกล่าวนั้น สามารถรองรับกับปัญหาต่างๆ ในเรื่อง Data Type และ Operator ที่สนับสนุนใน RDBMS ไม่เพียงพอกับความต้องการ โดยผู้ใช้สามารถทำการสร้าง Data Type ที่จำเป็นขึ้นมาใหม่ได้พร้อมทั้งสร้าง Operator ที่ใช้งานกับ Data Type ที่สร้างขึ้นมานั้นๆ และในกรณีปัญหาในเรื่องการขาด Built-in

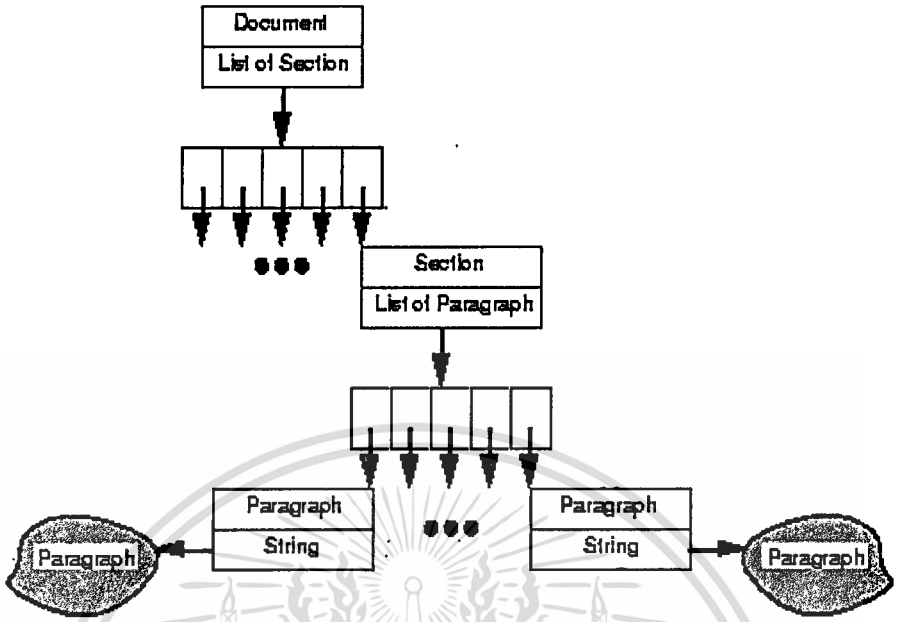
function ในการจัดการกับข้อมูลที่มีความซับซ้อนแนวคิดดังกล่าวสามารถรองรับโดยผู้ใช้สามารถสร้างฟังก์ชันการใช้งานกับข้อมูลที่มีความซับซ้อนเพิ่มเติมได้เองตามความเหมาะสมของงาน ซึ่งจะรวมอยู่ในส่วนของ Operation หรือเมธอดในออบเจ็กต์นั้นๆ ทั้งนี้จากแนวคิดดังกล่าวมีการอ้างอิงกลไกการ Encapsulation ซึ่งจะรวมกลุ่มของแอตทริบิวต์และกลุ่มของ Operations ที่ใช้จัดการออบเจ็กต์ดังกล่าวไว้ด้วยกัน และในส่วนของ Operation เองนั้นผู้ใช้สามารถกำหนด Integrity rule ต่างๆ ไว้เพื่อควบคุมความถูกต้องในการทำงานของออบเจ็กต์นั้นๆ ซึ่งการรวม Integrity rule ไว้ใน Object Structure ส่งผลให้ง่ายต่อการดูแลรักษาระบบ รวมทั้งมีความยืดหยุ่นในการจัดการและพัฒนาระบบงาน

6.1.2 แนวทางในการแก้ไขปัญหาการขาดความสามารถในการนำเสนอ Structure ของ RDBMS (ปัญหาที่ 4)

จากปัญหาการขาดความสามารถในการนำเสนอ Structure ของ RDBMS สามารถแก้ไขได้โดยใช้แนวคิดเกี่ยวกับ Complex Objects และ Object identity ที่สนับสนุนใน OODBMS ซึ่ง OODBMS จะมีกลไกที่สามารถยอมให้ผู้ใช้สามารถสร้างออบเจ็กต์ที่มีลักษณะซับซ้อนได้ ซึ่งเรียกว่า Complex Object โดยจะมีการรวมหรืออ้างอิงกับออบเจ็กต์อื่นๆ เช่น เอกสารที่มีความซับซ้อนซ่อนอยู่ภายในโครงสร้างของเอกสารนั้นๆ ทั้งนี้ OODBMS จะเตรียมคอนสตักเตอร์(Constructor) ต่างๆ ที่ช่วยในการสร้าง Complex Object ไว้

สำหรับ Complex Object นั้นเป็นออบเจ็กต์ที่สร้างจากออบเจ็กต์ง่ายๆ โดยการใช้คอนสตักเตอร์ช่วยในการสร้าง สำหรับรูปแบบที่ง่ายที่สุดของออบเจ็กต์(บางครั้งเรียกว่า Atomic Type ซึ่งเป็นรูปแบบที่ไม่มีส่วนประกอบย่อยเลยหรือ Subcomponents เลย) เช่น Integer, Character หรือ Boolean สำหรับรูปแบบของคอนสตักเตอร์ ของ Complex Object นั้นจะประกอบด้วยทูเปิล(Tuples), เซต (Sets), แบ็ก(Bags) (เป็นเซตที่อาจจะมีการซ้ำกันของสมาชิก), ลิสต์(Lists)และ อาร์เรย์(Arrays) โดยที่คอนสตักเตอร์เซต, ลิสต์ และทูเปิล เป็นกลุ่มของคอนสตักเตอร์อย่างน้อยที่ OODBMS ควรจะสนับสนุน ทั้งนี้อาจมีการเตรียมคอนสตักเตอร์ Union ซึ่งใช้ในการกำหนดค่าของแอตทริบิวต์ที่มาจากแอตทริบิวต์หนึ่งในกลุ่มของ Type ไว้ในบาง OODBMS

คอนสตักเตอร์เหล่านี้จะช่วยให้มีการสร้าง Complex Object ได้ เนื่องจากยอมให้มีการสร้างออบเจ็กต์ที่มีการอ้างถึงออบเจ็กต์อื่น สำหรับ Complex Object นั้นบ่อยครั้งจะพบในการประยุกต์ใช้ทางด้านมัลติมีเดีย ซึ่งตัวอย่างของ Complex Object ที่เป็นออบเจ็กต์เอกสารได้แสดงไว้ดังรูปที่ 6.3



รูปที่ 6.3 ออบเจ็กต์เอกสารซึ่งเป็น Complex Object

จากรูปดังกล่าวสามารถอธิบายในลักษณะของอัลกอริธึม(Algorithm) ได้ดังต่อไปนี้

```
Create Type Document
Supertypes: TextObject
Properties:
      title: String;
      author: String;
      components: List of Section;
Operations:
      print;
      checkSpelling: String {errors};
      checkGrammar: String {errors};

Create Type Section
Supertypes: TextObject
Properties:
      title: String;
      number: Integer;
      components: List of Paragraph;
Operations:
      print;
      checkSpelling: String {errors};
      checkGrammar: String {errors};

Create Type Paragraph
Supertypes: TextObject
```

```
Properties:
      paragraphText: TextString;
Operations:
      print;
      checkSpelling: String {errors};
      checkGrammar: String {errors};
      getSentence (index: Integer): TextString;
      changeFont (String);
```

ในแต่ละออบเจ็กต์จะมี Identity ซึ่งใช้ในการอ้างถึงได้อย่างอิสระสำหรับแต่ละ State ของออบเจ็กต์ ทั้งนี้จะพบว่า Object Identity จะมีแนวคิดตรงกันข้ามกับทูเปิลใน RDBMS ที่ต้องมีการเข้าถึงโดยการผ่านค่าของคีย์ที่เป็นส่วนหนึ่งของทูเปิลนั้นๆ โดยทั่วไปแล้ว Object Identity นั้นจะถูกสร้างขึ้นอย่างชัดเจนโดยระบบ ซึ่งจะมีลักษณะอยู่ในรูปของพอยน์เตอร์เชิงวัตถุ(Object-Oriented Pointers) หรือบางครั้งอาจจะเรียกว่า Surrogates ซึ่งในส่วนนี้ผู้ใช้ไม่จำเป็นต้องกำหนด คีย์ที่ใช้ในการอ้างอิงซึ่งส่วนประกอบต่างๆ ลักษณะการจัดการดังกล่าวนี้มีความสอดคล้องกับเรื่องของความเป็นหนึ่งเดียว(Uniqueness) และเรื่องของ Referential Integrity ที่ถูกดูแลและจัดการในระบบการใช้งานด้วย

หลักสำคัญของการใช้ Object Identifier ในการสร้าง Complex Object เพื่อที่จะช่วยชี้หรืออ้างอิงไปยังส่วนประกอบอื่นๆ ใน Complex Object ตัวอย่างเช่น ออบเจ็กต์เอกสารที่กล่าวมาข้างต้นนั้นจะมีลิสต์ของ Identifier ที่ถูกใช้แทนส่วนต่างๆ ของเอกสาร ซึ่ง Identifier ดังกล่าวเป็นส่วนที่ถูกใช้ในการกล่าวถึงแนวความคิด “Share Structure” ซึ่งจะหมายถึงการสร้างออบเจ็กต์ที่แตกต่างกัน 2 ออบเจ็กต์(State เดียวกัน) และออบเจ็กต์ดังกล่าวมีส่วนประกอบย่อย(Subcomponents) เดียวกันทั้ง 2 ออบเจ็กต์

จากตัวอย่างของ gate ที่กล่าวไว้ในส่วนของปัญหาข้างต้นนั้น (กล่าวในบทที่ 2) สามารถกำหนด Entity type รวมทั้ง Operation ซึ่งแสดงในรูปของ Complex Object ได้ดังนี้

Entity GATE TYPE is ENTITY

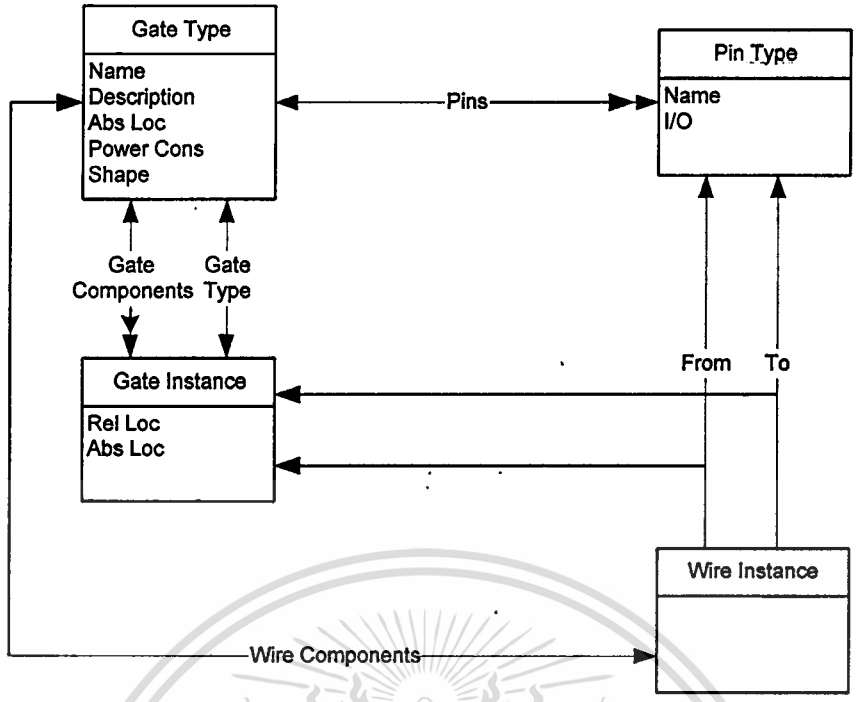
Function Name(GATE TYPE) -> STRING

Function Description(GATE TYPE) -> STRING

Function Pins(GATE TYPE) -> set of PIN TYPE

Function Gate Components(GATE TYPE) -> set of GATE INSTANCE

Function Wire Components(GATE TYPE) -> set of WIRE INSTANCE



รูปที่ 6.4 แสดง Complex Object ใน the PROBE Data Model

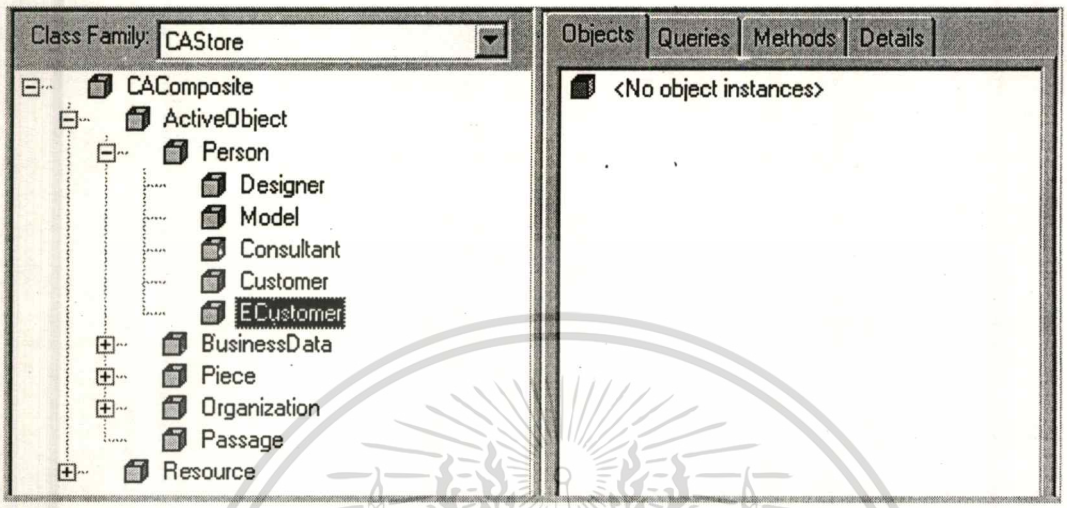
6.2 แนวทางในการแก้ปัญหาด้วยระบบการจัดการฐานข้อมูลเชิงวัตถุ Jasmine

จากที่กล่าวไว้ในบทที่ 5 แสดงให้เห็นว่าระบบการจัดการฐานข้อมูลเชิงวัตถุ Jasmine เป็น "Pure Object-Oriented Database Management" ซึ่งสามารถแก้ไขปัญหาหรือความไม่เหมาะสมที่เกิดขึ้นบนระบบการจัดการฐานข้อมูลเชิงสัมพันธ์ในงานระบบสารสนเทศปัจจุบันด้วยแนวคิดเชิงวัตถุที่สนับสนุนในระบบการจัดการฐานข้อมูลเชิงวัตถุ Jasmine ได้เช่นเดียวกับแนวทางในการแก้ปัญหาด้วยแนวคิดของ Object–Oriented Database ที่กล่าวมาแล้วข้างต้นของบทนี้ โดยแนวทางการแก้ไขปัญหาดังกล่าวด้วย Jasmine สามารถทำได้ดังนี้

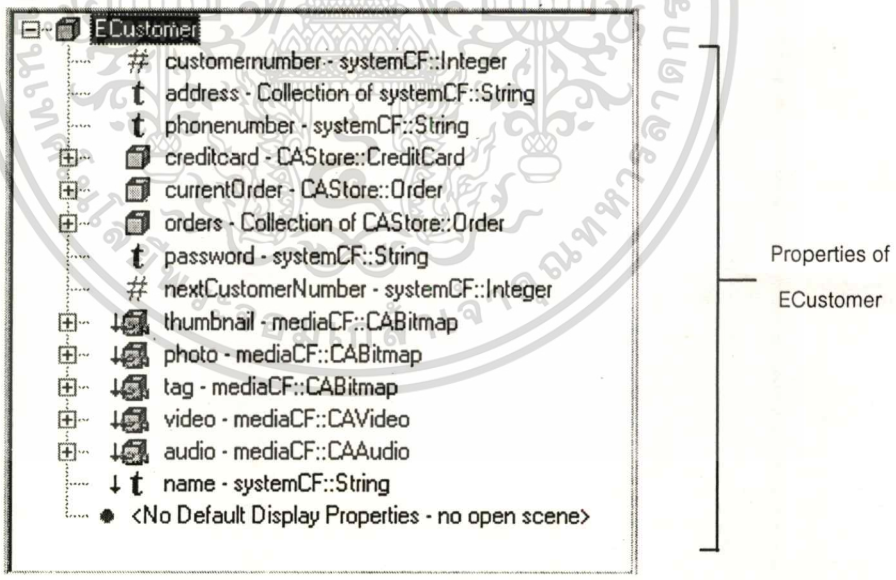
6.2.1 แนวทางในการแก้ไขปัญหาเรื่อง Data Type, Operator, Built-in function และ Integrity rules ใน RDBMS (ปัญหาที่ 1, 2, 3 และ ปัญหาที่ 5)

ในระบบการจัดการฐานข้อมูลเชิงวัตถุ Jasmine ผู้ใช้สามารถสร้างหรือกำหนด Type ของ Data Object ขึ้นมาใหม่รวมทั้งกำหนด Operator ที่ใช้งานกับ Type หรือออบเจ็กต์นั้นๆ ตามความเหมาะสมของปัญหา ทั้งนี้การกำหนด Built-in function และ Integrity rule ของออบเจ็กต์นั้นจะรวมอยู่ในส่วนของการกำหนด Operator ด้วยเช่นกัน และในการกำหนด Operator ดังกล่าวนั้นสามารถ

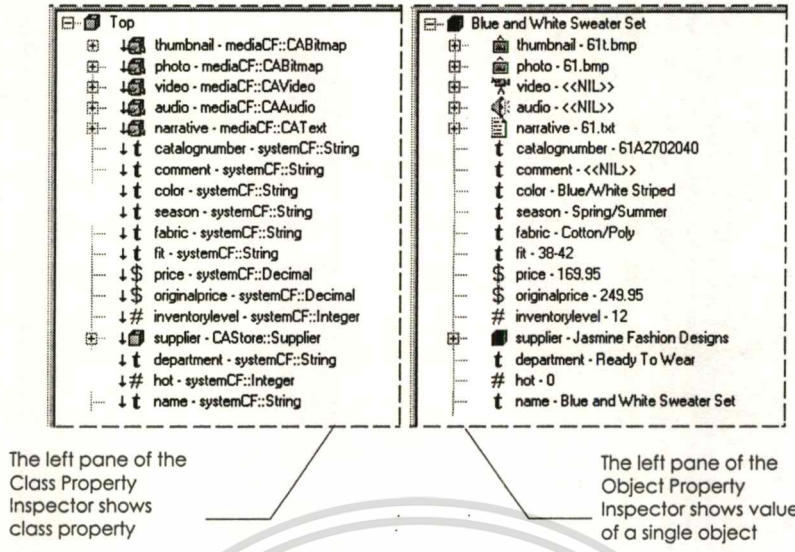
กำหนดไว้ในส่วนของเมธอดที่จัดเตรียมไว้ในแต่ละออบเจ็กต์หรือคลาสที่สร้างขึ้น โดยการกำหนด Type และ Operator ในระบบการจัดการฐานข้อมูลเชิงวัตถุ Jasmine สามารถแสดงได้ดังรูปที่ 6.5 ถึง รูปที่ 6.7



รูปที่ 6.5 แสดงการสร้างคลาสในระบบการจัดการฐานข้อมูลเชิงวัตถุ Jasmine

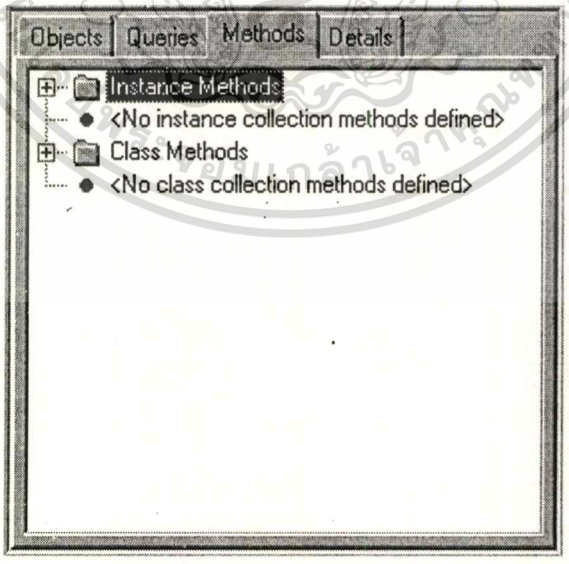


รูปที่ 6.6 แสดงการกำหนดรายละเอียดของ Property ในคลาส ECustomer



รูปที่ 6.7 แสดงการเปรียบเทียบ Property ของคลาสกับค่าของ Property ในออบเจ็กต์ของคลาส

จากรูปที่ผ่านมาแสดงการกำหนดรายละเอียด Propertyในคลาสของออบเจ็กต์ ซึ่งผู้ใช้สามารถกำหนด Type ที่ต้องการใช้งานขึ้นมาตามความเหมาะสมของปัญหาตามแนวความคิดเชิงวัตถุและในส่วนต่อไปจะเป็นการกล่าวถึงการสร้าง Operator ที่นำใช้งานกับ Type ที่สร้างขึ้นมานั้น โดยผู้ใช้สามารถกำหนด Operator ต่างๆ ในส่วนเมธอดของคลาสที่สร้างขึ้นมานั้นๆ โดยการกำหนด Operator ในระบบการจัดการฐานข้อมูลเชิงวัตถุ Jasmine สามารถแสดงได้ดังรูปที่ 6.8

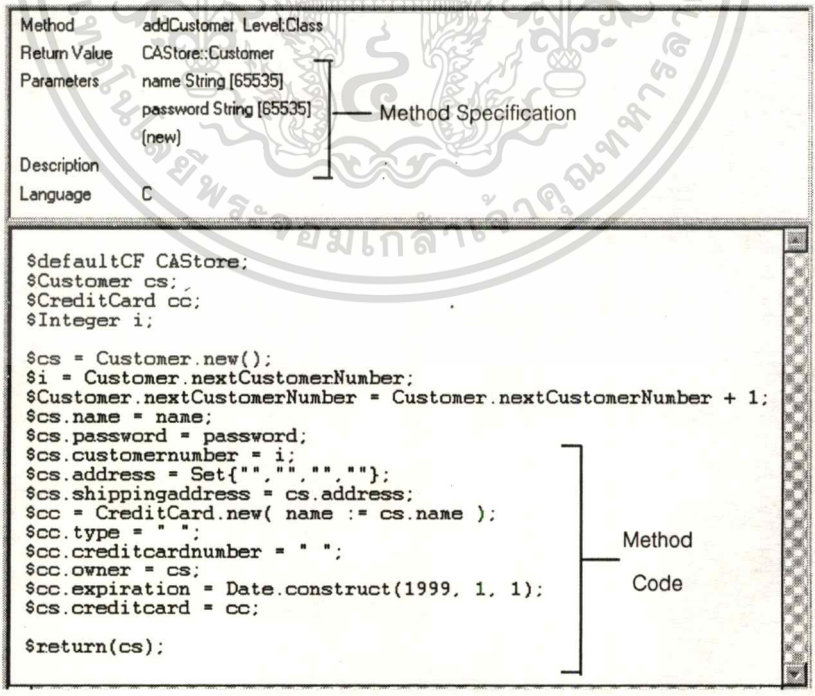


รูปที่ 6.8 แสดงส่วนของเมธอดที่ใช้ในการกำหนด Operator สำหรับใช้งานกับออบเจ็กต์ในคลาสนั้นๆ

จากรูปจะสังเกตเห็นว่าผู้ใช้สามารถกำหนดประเภทของเมธอดได้คือ Instance Method กับ Class Method แต่นอกจากเมธอดดังกล่าวแล้วผู้ใช้ยังสามารถกำหนดเมธอดได้อีกรูปแบบหนึ่งคือ Collection Method ซึ่งความหมายของเมธอดแต่ละประเภทสามารถอธิบายได้ดังนี้

- Instance-Level Methods (Instance Methods) เป็นเมธอดที่ใช้เฉพาะกับออบเจ็กต์ ตัวอย่างเช่นการกำหนดเมธอดที่ใช้ในการคำนวณหาระยะเวลาการทำงานของพนักงานแต่ละคนในบริษัทหรือองค์กร
- Class-Level Methods (Class Methods) เป็นเมธอดที่ใช้กับออบเจ็กต์ภายในคลาสทั้งหมด ตัวอย่างเช่นการกำหนดเมธอดที่ใช้ในการสร้างออบเจ็กต์ใหม่หรือการสร้าง Instance ของคลาสโดยใช้ข้อมูลที่เก็บอยู่ใน External file
- Collection-Level Methods (Collection Methods) สำหรับ Collection คือการเลือกออบเจ็กต์ที่มีรูปแบบเหมือนๆ กัน เช่น Collection ของตัวเลข หรือ Collection ของบุคลากร ซึ่งทั้ง Instance-Level Methods และ Class-Level Methods สามารถกำหนดและประยุกต์ใช้กับ Collection ได้โดยจะเรียกว่า Collection-Level Method ตัวอย่างเช่นการกำหนดเมธอดที่ใช้ในการเพิ่มค่าตอบแทนให้กับ Collection ของพนักงาน

ทั้งนี้การกำหนดรายละเอียดของแต่ละ Operator ในเมธอดแบ่งเป็น 2 ส่วนคือส่วนของ Method Specification และส่วนของ Method Code โดยรายละเอียดสามารถแสดงได้ดังรูปที่ 6.9



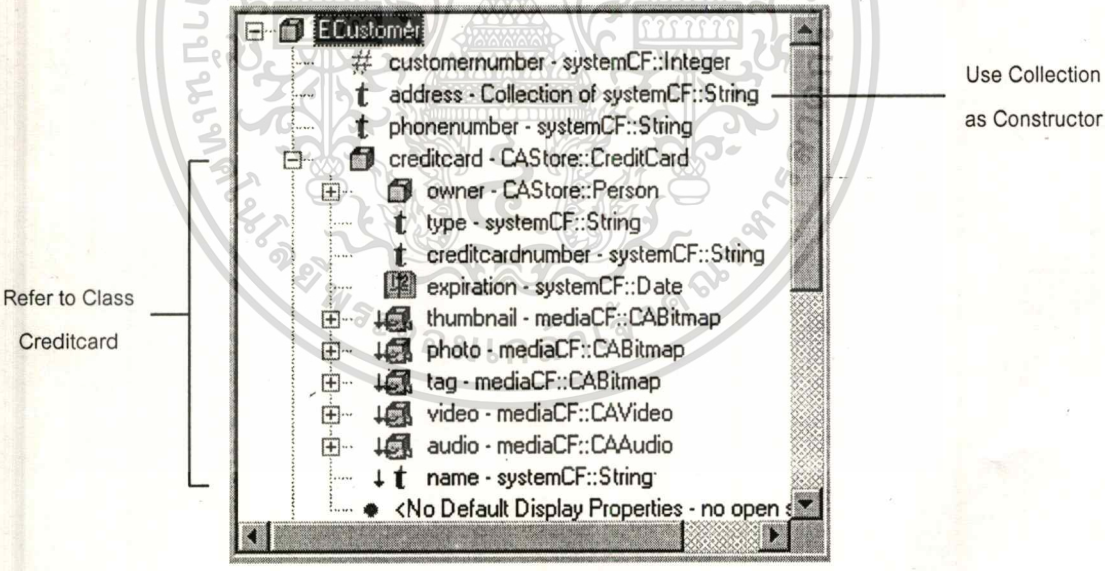
รูปที่ 6.9 แสดงการกำหนดรายละเอียดของเมธอด

6.2.1 แนวทางในการแก้ไขปัญหาการขาดความสามารถในการนำเสนอ Structure ของ RDBMS (ปัญหาที่ 4)

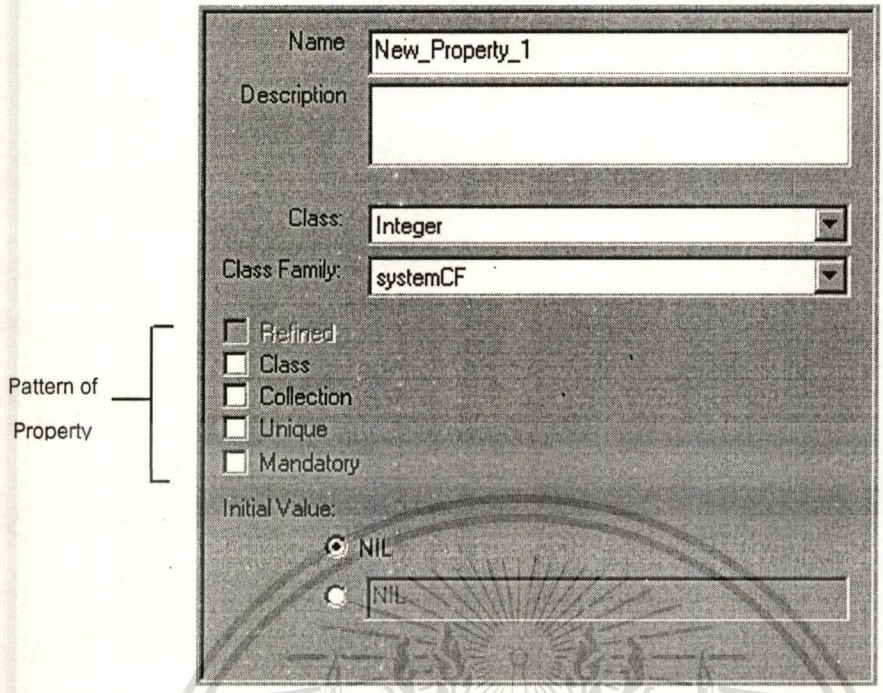
ในระบบการจัดการฐานข้อมูลเชิงวัตถุ Jasmine ผู้ใช้สามารถนำเสนอข้อมูลในลักษณะ Structure ได้ตามแนวคิดเกี่ยวกับ Complex Object และ Object Identity ที่สนับสนุนใน OODBMS นอกจากนี้ในระบบการฐานข้อมูลเชิงวัตถุ Jasmine ได้เตรียมคอนสตักเตอร์ที่ช่วยในการสร้าง Complex Object ไว้ดังนี้คือทูเปิล(Tuples), เซต(Sets), แบ็ก(Bags), ลิสต์(Lists)และ อาร์เรย์(Arrays) และในการอ้างอิงออบเจ็กต์ที่เป็นส่วนประกอบนั้นจะมี Object Identity เป็นกลไกในการจัดการดูแลออบเจ็กต์ต่างๆ ที่อ้างถึง

สำหรับ Object Identity ในระบบการจัดการฐานข้อมูลเชิงวัตถุ Jasmine จะมีโครงสร้างเป็น Logical Object Identity ที่ประกอบด้วย Database ID, Class ID และ Instance ID ซึ่งโดยทั่วไปแล้ว Object Identity นั้นจะถูกสร้างขึ้นและดูแลโดยระบบฐานข้อมูลนั้นๆ ทำให้ผู้ใช้ไม่จำเป็นต้องกำหนดคีย์ที่ใช้ในการอ้างอิงซึ่งส่วนประกอบต่างๆ ด้วยตัวเอง

และในการสร้าง Complex Object ดังกล่าวนั้นในระบบการจัดการฐานข้อมูลเชิงวัตถุ Jasmine สามารถแสดงได้ดังรูปที่ 6.10 และรูปที่ 6.11



รูปที่ 6.10 แสดงโครงสร้างของออบเจ็กต์ ECustomer ที่มีลักษณะเป็น Complex Object



รูปที่ 6.11 แสดงการกำหนดลักษณะของ Property ภายในโครงสร้างของออบเจ็กต์ ECustomer ที่มีลักษณะเป็น Complex Object

นอกจากนี้แล้วยังสามารถใช้ภาษา ODQL(Object Database Query Language) ที่สนับสนุนในระบบการจัดการฐานข้อมูลเชิงวัตถุ Jasmine ในการสร้าง Complex Object ดังกล่าว โดยสามารถเขียนอัลกอริธึมได้ดังนี้

```
DefineClass     User
super :  applicationObject
        description : "The class implementing users"
{
 class :
        User createObject(Session Session, String lastName,
                          String firstName, String mi,
                          String loginName, String password);
/* This method will retrieve all users except the
** users that have the status of  "Out of Service".
```

```
*/
        User set getAllInServiceUsers(Session Session);
/*
** Retrieve the user with a given name.
*/
        User getUserWithName(Session Session, String name);
 instance :
        String lastName default : " ";
        String firstName default : " ";
        String middleInitail        default : " ";
        Department      userDepartmentl     :
        set<Manager>    managers;
        Manager                 mainManager;
        Integer salary;
        String rank     default : "MTS";
        BasicAddress    address;
        Company                 userCompany;
        Date            dateHired;
        Real hourRate   default : 0;

        Void setUserName(Session Session,
                        String lastName, String firstName,
                        String middleInitial);
/*
** The last, first, and middle name through these accessors
** are obtained and stored in parameters
*/
        String getFirstName(Session Session);
        String getLastName(Session Session);
        String getMiddleInitial(Session Session);
```

```
        String getFullName(Session Session);
        Void setHomeAddress(Session Session,
                            BasicAddress newaddress);
        BasicAddress getHomeAddress(Session Session);
        Void setHourRate(Session Session, Real hourRate);
        Real getHourRate(Session Session);
/* Other methods */
        Set<Manager> getManagers();
        Real evaluateBonus();
        Manager getMainManager();
...
    };
```

บทที่ 7

ตัวอย่างการพัฒนาโปรแกรมประยุกต์บน Jasmine

ในบทที่ผ่านมาได้แสดงถึงแนวทางในการแก้ปัญหาหรือความไม่เหมาะสมของระบบการจัดการฐานข้อมูลเชิงสัมพันธ์ที่เกิดขึ้นกับงานระบบสารสนเทศในปัจจุบันด้วยแนวคิดเชิงวัตถุในระบบการจัดการฐานข้อมูลเชิงวัตถุ รวมทั้งแนวทางในการแก้ไขปัญหาดังกล่าวด้วยระบบจัดการฐานข้อมูลเชิงวัตถุ Jasmine

การศึกษาในบทนี้จะเป็นการกล่าวถึงตัวอย่างการพัฒนาโปรแกรมประยุกต์บนระบบการจัดการฐานข้อมูลเชิงวัตถุ Jasmine โดยจะใช้ Jasmine Studio ในการพัฒนาตัวอย่างโปรแกรมประยุกต์ โดยจะอธิบายถึงการสร้างคลาส, การสร้างเมธอด และการสร้าง Query รวมทั้งความสัมพันธ์ระหว่างออบเจ็กต์ที่สร้างขึ้นบนระบบการจัดการฐานข้อมูลเชิงวัตถุ Jasmine สุดท้ายจะเป็นตัวอย่างการสร้างโปรแกรมประยุกต์ดังกล่าว

7.1 แนวทางในการพัฒนาโปรแกรมประยุกต์บน Jasmine

การพัฒนาโปรแกรมประยุกต์บนระบบการจัดการฐานข้อมูลเชิงวัตถุ Jasmine สามารถทำได้ 2 วิธีการหลักๆ ด้วยกันคือ (จากที่กล่าวผ่านมาแล้วในบทที่ 5)

- การพัฒนาโปรแกรมประยุกต์ด้วย Jasmine Studio
- การพัฒนาโปรแกรมประยุกต์ด้วย Jasmine C API

โดยในบทนี้จะมุ่งประเด็นไปที่การพัฒนาโปรแกรมประยุกต์ด้วย Jasmine Studio ซึ่งมีในหลักการพัฒนาค่อนข้างสะดวกและรวดเร็ว เหมาะกับผู้ใช้หรือผู้พัฒนาที่ไม่ถนัดการเขียนโปรแกรมติดต่อกับ Dynamic Link Libraries (DLLs) ทั้งนี้การพัฒนาโปรแกรมประยุกต์ด้วย Jasmine Studio ยังมีลักษณะเป็นแบบ drag-and-drop, point-and-click ซึ่งเพิ่มความสะดวกสบายในการพัฒนา และนอกจากนี้ยังสามารถพัฒนาโปรแกรมประยุกต์ที่มีอยู่ไปเป็นโปรแกรมประยุกต์บนระบบเครือข่าย Internet หรือที่เรียกว่า "Web Enabled Application" ได้ง่ายอีกด้วย

7.2 สภาพแวดล้อมในการทำงานบน Jasmine Studio

สภาพแวดล้อมในการทำงานบน Jasmine Studio นั้นผู้ใช้หรือผู้พัฒนาสามารถใช้คุณสมบัติต่างๆ ที่สนับสนุนใน Jasmine Studio เช่น สามารถสร้างและปรับปรุงโปรแกรมประยุกต์ที่พัฒนาขึ้น

ด้วย Jasmine Studio, สามารถเขียนเมธอดเพื่อทำการ Manipulate ฐานข้อมูลเชิงวัตถุ,สามารถเขียน Query ที่ใช้ในการดึง(Retrieve) ออบเจ็กต์นั้นๆ จากฐานข้อมูล รวมทั้งการจัดการคลาสต่างๆ และจัดการกับออบเจ็กต์ที่ถูกจัดเก็บในฐานข้อมูล Jasmine

7.2.1 Jasmine Studio Windows

หน้าต่างการทำงานหลักๆ ใน Jasmine Studio จะประกอบด้วยหน้าต่างดังต่อไปนี้

- Jasmine Application Manager
- Class Browser
- Class Property Inspector
- Object Property Inspector
- Query Editor
- Method Editor

โดยรายละเอียดการทำงานของหน้าต่างข้างต้นนั้นได้อธิบายไว้ในบทที่ 5 ซึ่งเป็นบทที่กล่าวถึงการแนะนำระบบการจัดการฐานข้อมูลเชิงวัตถุ Jasmine

7.3 ตัวอย่างขั้นตอนในการพัฒนาโปรแกรมประยุกต์ด้วย Jasmine Studio

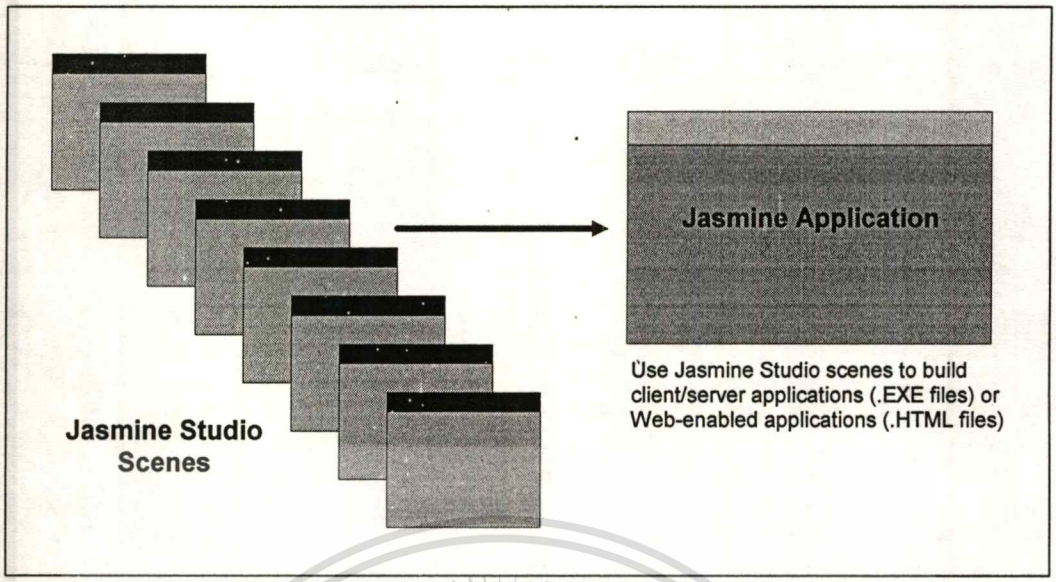
ในส่วนนี้จะเป็นการกล่าวถึงตัวอย่างขั้นตอนในการพัฒนาโปรแกรมประยุกต์ด้วย Jasmine Studio โดยเริ่มแรกจะเป็นการกล่าวถึงขั้นตอนพื้นฐานที่ใช้ในการพัฒนาโปรแกรมประยุกต์ด้วย Jasmine Studio จากนั้นจะเป็นการอธิบายการสร้างคลาสในระบบการจัดการฐานข้อมูลเชิงวัตถุ Jasmine, การใช้คอนสตักเตอร์ในการสร้างออบเจ็กต์ที่มีความซับซ้อนหรือใช้สร้างข้อมูลที่มีลักษณะเป็น Structure รวมทั้งการสร้างสัมพันธ์(Relationship) ระหว่างคลาสหรือออบเจ็กต์ที่กำหนดขึ้นใหม่ จากนั้นจะเป็นการอธิบายการส์ร้างเมธอดหรือกลุ่มของ Operation ที่ใช้ในคลาสนั้นๆ และอธิบายการเขียน Query ที่ใช้ในการสอบถามหรือดึงออบเจ็กต์นั้นๆ จากฐานข้อมูล สุดท้ายจะอธิบายการสร้างโปรแกรมประยุกต์ที่มีการติดต่อกับคลาส, ออบเจ็กต์ภายในคลาส, เมธอด รวมทั้งการใช้ Query ที่ได้ทดลองสร้างขึ้นมาตามลำดับ

7.3.1 ขั้นตอนพื้นฐานในการพัฒนาโปรแกรมประยุกต์ด้วย Jasmine Studio

ขั้นตอนง่ายๆ ซึ่งเป็นพื้นฐานในการพัฒนาโปรแกรมประยุกต์ด้วย Jasmine Studio นั้น ประกอบด้วยขั้นตอนดังต่อไปนี้

1. Start Jasmine
2. Start Jasmine Studio
3. ทำการเชื่อมต่อไปยังระบบการจัดการฐานข้อมูลเชิงวัตถุ Jasmine
4. สร้าง "Application" ใหม่ในหน้าต่าง Application Manager
5. สร้าง "Scene" ใหม่(จำนวนหนึ่งหรือมากว่า) ในส่วนของ "Application"
6. แอ็กเซส Item ในฐานข้อมูลและทำการลาก(Drag) Item เหล่านั้นจาก Jasmine Studio Browser มาไว้บน Scene
7. กำหนดพฤติกรรมหรือ Behavior ของออบเจ็กต์ใน "Behavior Dialog"
8. สร้างคลาส Property ในหน้าต่าง Class Property Inspector
9. ทำการ "Build" คลาสที่สร้างขึ้น(กรณีเป็นคลาสใหม่)ในหน้าต่าง Class Browser
10. สร้างออบเจ็กต์ใน Class Browser
11. กำหนดค่าของออบเจ็กต์ Property ในหน้าต่าง Object Property Inspector
12. (Optional) สร้าง Query ในหน้าต่าง Query Editor
13. (Optional) สร้าง Method และ Compile Method ในหน้าต่าง Method Editor
14. ทำการทดสอบ Scene ใน Run Mode
15. ดีบัก(Debug) Scene ในหน้าต่าง Debug
16. บันทึกการเปลี่ยนแปลงของ Scene
17. ทำการ Compile เพื่อใช้งานในโปรแกรมประยุกต์
18. ทำการ "Build" Executable File(.EXE) หรือ Publish, Install, และ Run "Web Enabled Application"

ทั้งนี้ใน Jasmine Studio จะใช้ Scene ในการประกอบขึ้นมาเป็นโปรแกรมประยุกต์ ซึ่งในแต่ละ Scene จะมีลักษณะเป็น "Page" โดยในโปรแกรมประยุกต์หนึ่งสามารถประกอบด้วย Scene หลายๆ Scene นอกจากนี้ผู้ใช้หรือผู้พัฒนาสามารถใช้ Scene ใน Jasmine Studio สร้างโปรแกรมประยุกต์ Client/Server (.EXE file) หรือสร้างเป็น Web Enabled Application (.HTML file)



รูปที่ 7.1 แสดงการใช้ Sceneใน Jasmine Studio เพื่อสร้างโปรแกรมประยุกต์บน Jasmine

7.3.2 การสร้างคลาสใหม่ในระบบการจัดการฐานข้อมูลเชิงวัตถุ Jasmine

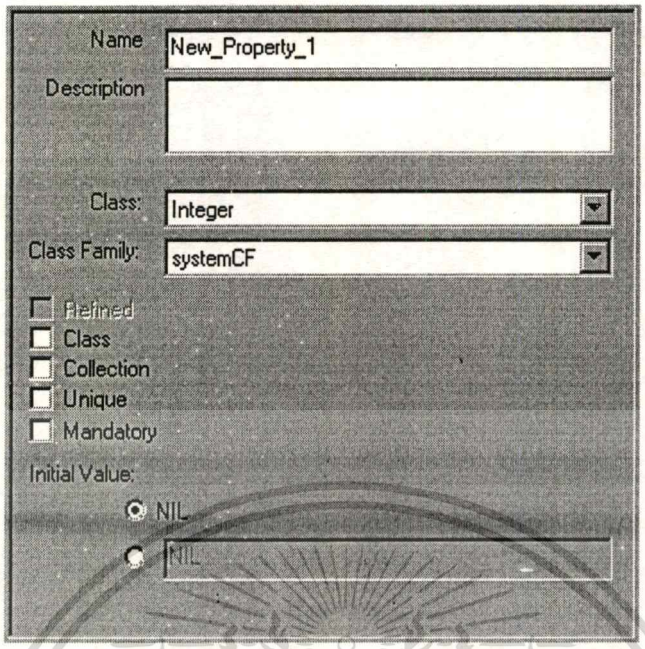
ในตัวอย่างนี้จะเป็นการทดลองสร้างคลาส ECustomer ซึ่งเป็นคลาสของลูกค้าที่มีการติดต่อซื้อขายทางระบบเครือข่าย Internet ด้วย Jasmine Studio โดยขั้นตอนในการสร้างคลาสใหม่สามารถทำได้ดังนี้

1. ทำการเปิด หน้าต่าง Jasmine Class Browser โดยการเลือก File/Database Administration (หรือ Click ที่ปุ่ม Database Administration)
2. ทำการเลือก Class Family (ในระบบการจัดการฐานข้อมูลเชิงวัตถุ Jasmine ได้จัดเตรียม Class Family ไว้ดังนี้คือ Local widget, CAStore, mediaCF, sqlCF และ WebLink) ซึ่งในตัวอย่างนี้เลือก CAStore
3. ทำการเลือกคลาสย่อยใน CAStore ซึ่งจะประกอบด้วย 2 คลาสหลักๆ คือ คลาส ActiveObject และคลาส Resource (ผู้ใช้อาจกำหนดกลุ่มหรือคลาสย่อยใหม่นอกเหนือจากนี้ได้) โดยคลาส ActiveObject จะเป็นคลาสของออบเจ็กต์ใน Real-World เช่น คน, สัตว์, สิ่งของ เป็นต้น ส่วนคลาส Resource จะเป็นคลาสของออบเจ็กต์ที่ใช้ประกอบการสร้าง User Interface ในแต่ละScene ของโปรแกรมประยุกต์ ซึ่งในตัวอย่างนี้จะเลือกคลาส ActiveObject
4. ทำการเลือกคลาส Person ซึ่งเป็นคลาสย่อยของคลาส ActiveObject

5. ทำการสร้างคลาสใหม่ซึ่งในตัวอย่างนี้คือคลาส ECustomer โดยการเลือก File/New/Subclass (หรือ Right-Click ที่คลาส Person แล้วทำการเลือก Subclass จาก Context menu)
6. ทำการกำหนด Property ต่างๆ ในคลาส ECustomer โดยการ Double-Click ที่คลาส ECustomer (หรือ Right-Click ที่คลาส ECustomer แล้วทำการเลือก Open จาก Context menu) โดยการกำหนด Property ต่างๆ ของคลาส ECustomer สามารถแสดงได้ดังรูปที่ 7.2

รูปที่ 7.2 แสดง Property ของคลาส ECustomer

7. ในการกำหนด Property สามารถทำได้โดยการ Right-Click ที่ว่างหรือ Property อื่นที่มีในคลาสย่อย ECustomer แล้วทำการเลือก New Property จาก Context menu จากนั้นทำการกำหนดองค์ประกอบต่างๆ ของ Property ที่ Pane ทางด้านขวามือ ซึ่งสามารถแสดงได้ดังรูปที่ 7.3



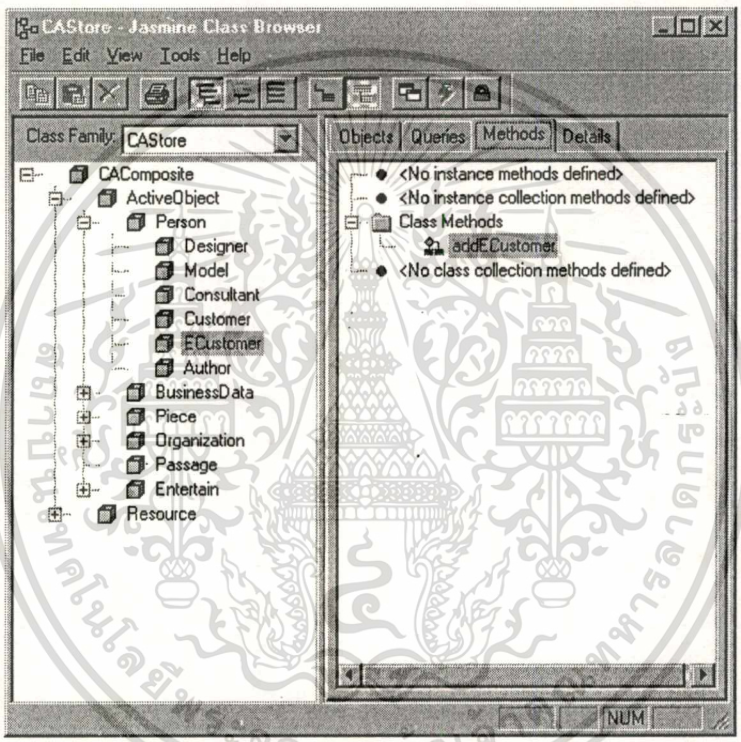
รูปที่ 7.3 แสดงการกำหนดองค์ประกอบของ Property

8. จากการกำหนด Property ดังกล่าวข้างต้น สามารถกำหนด Property ในลักษณะ Collection ซึ่งสนับสนุนคอนสตักเตอร์ที่ใช้ในการสร้างออบเจ็กต์ที่มีความซับซ้อน หรือสนับสนุนการนำเสนอข้อมูลลักษณะ Structure ซึ่งในตัวอย่างนี้จะทำการกำหนดให้ Property "address" มีการนำเสนอข้อมูลลักษณะ Collection (สังเกตได้จากรูปแสดง Property ของคลาส ECustomer ในส่วนของ address)
9. จากการกำหนด Property ดังกล่าวข้างต้น สามารถกำหนด Property ในลักษณะที่มีการอ้างอิงคลาสหรือออบเจ็กต์อื่น หรือสามารถกำหนดความสัมพันธ์ระหว่างออบเจ็กต์หรือคลาสนั้นๆ ได้ ตัวอย่างเช่น Property "creditcard" มีการอ้างอิงกับคลาส creditcard ซึ่งเป็นคลาสย่อยของคลาส Business Data อีกทีหนึ่ง เป็นต้น (นอกจากนี้อาจมีการอ้างอิงข้าม Class Family ได้เช่น Property "photo" ซึ่งมีความสัมพันธ์กับออบเจ็กต์หรือคลาสจาก Class Family "mediaCF" เป็นต้น
10. ทำการ Build คลาสดังกล่าวหลังจากกำหนด Property ของคลาสเสร็จสมบูรณ์(อาจทำการ Build ก่อนกำหนด Property ก็ได้) โดยการ Right-Click ที่คลาสECustomer แล้วทำการเลือก Build จาก Context menu

7.3.2 การสร้างเมธอดในระบบการจัดการฐานข้อมูลเชิงวัตถุ Jasmine

ในตัวอย่างนี้จะเป็นการทดลองสร้างเมธอด addECustomer ด้วย Jasmine Studio ซึ่งเมธอดดังกล่าวเป็นเมธอดในระดับคลาส(Class-Level Method) ของคลาส ECustomer ซึ่งเมธอด addECustomer นี้มีหน้าที่ในการสร้างออบเจ็กต์ใหม่ในคลาส ECustomer โดยขั้นตอนในการสร้างเมธอดดังกล่าว สามารถทำได้ดังนี้

1. เลือกคลาสย่อย ECustomer (ในกรณีเปิดหน้าต่าง Jasmine Class Browser ไว้แล้ว)
2. ทำการเลือก Tab Method ทาง Pane ขวามือ ซึ่งสามารถแสดงได้ดังรูปที่ 7.4

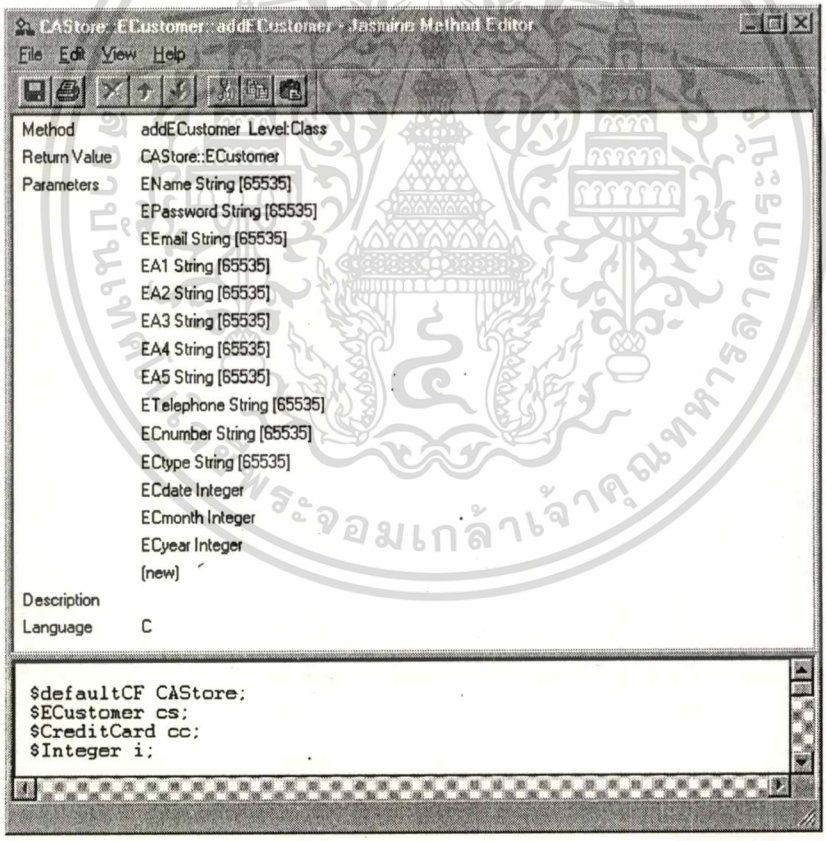


รูปที่ 7.4 แสดง Tab Method

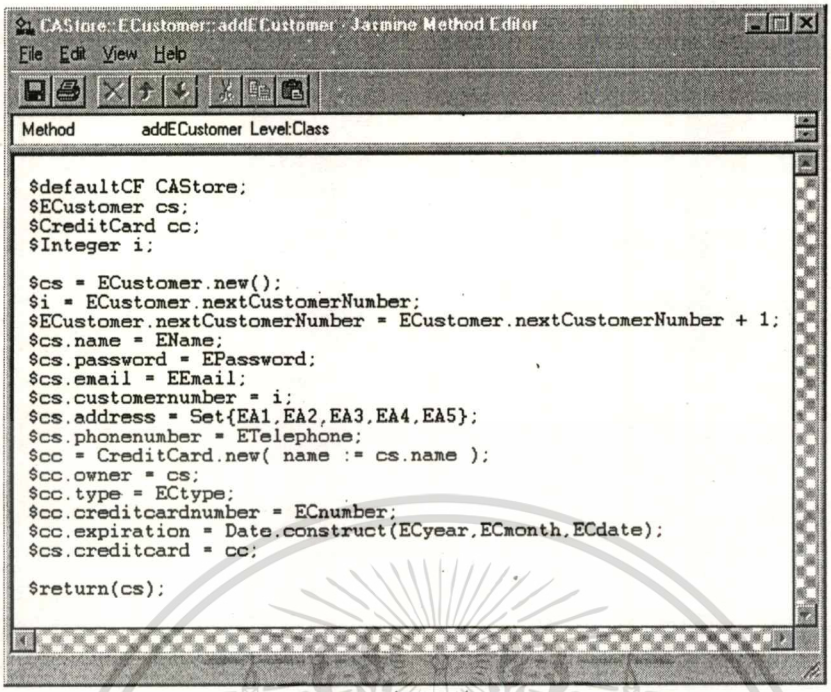
3. ทำการสร้างเมธอด addECustomer โดยการ Right-Click ที่ Class Methods แล้วทำการเลือก New Method จาก Context menu
4. ทำการเปิดเมธอด addECustomer เพื่อกำหนดการทำงานของเมธอดดังกล่าวโดยการเลือก File/Open (หรือ Double-Click ที่เมธอดดังกล่าว)
5. ในส่วนของ Method Editor ผู้ใช้หรือผู้พัฒนาสามารถใช้ Editor ดังกล่าวได้ดังนี้
   - สามารถโปรแกรมการทำงาน, แก้ไขและจัดเก็บเมธอด
   - สามารถกำหนดชื่อของเมธอดและกำหนด Type ของเมธอด

- สามารถเลือก Type ของค่าที่จะได้กลับมา(Returned) ตามความต้องการหลังจากการ Execute เมธอด
- สามารถกำหนดพารามิเตอร์ของเมธอด
- สามารถอธิบายวัตถุประสงค์การทำงานของเมธอด
- สามารถเลือกภาษาที่ใช้ในการโปรแกรมเช่น ภาษา C/C++ ซึ่งใช้ในการเขียนการทำงานของเมธอดดังกล่าว
- สามารถดูโค้ดของเมธอดที่อยู่เดิม
- สามารถใช้คำสั่ง Copy, Cut และ Paste ภายในส่วน Editor ดังกล่าว
- สามารถพิมพ์รายละเอียดของเมธอด

6. กำหนดรายละเอียดต่างๆ ในหน้าต่าง Method Editor ซึ่งสามารถแสดงได้ดังรูปที่ 7.5



รูปที่ 7.5 แสดงการกำหนดรายละเอียดของ Method



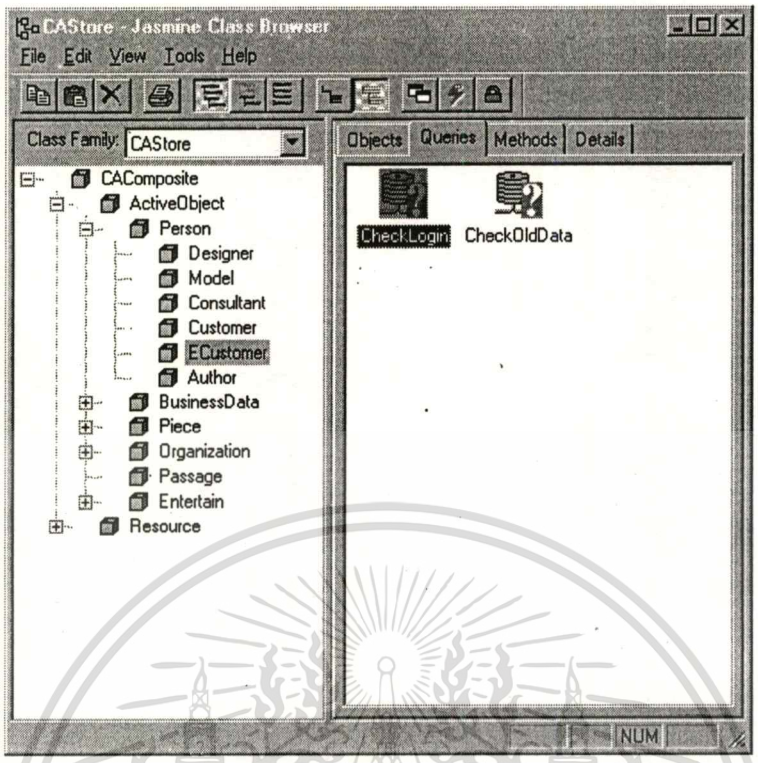
รูปที่ 7.5 แสดงการกำหนดรายละเอียดของ Method (ต่อ)

7. ทำการ Compile เมธอดที่สร้างขึ้น โดยเลือกที่คลาสของเมธอดนั้นๆ ในที่นี้จะเลือกคลาส ECustomer ซึ่งเป็นคลาสที่เก็บเมธอดดังกล่าวไว้ โดย Right-Click ที่คลาสดังกล่าวแล้วทำการเลือก Compile จาก Context menu

7.3.4 การสร้าง Query ในระบบการจัดการฐานข้อมูลเชิงวัตถุ Jasmine

ในตัวอย่างนี้จะเป็นการทดลองสร้าง Query ที่ใช้ในการตรวจสอบผู้เข้าใช้ระบบว่ามีสิทธ์ในการเข้าใช้ระบบหรือไม่อย่างไร ด้วย Jasmine Studio โดยการตรวจสอบจะกระทำบนคลาส ECustomer ซึ่งเป็นคลาสที่เก็บออบเจ็กต์ลูกค้าแต่ละคนซึ่งมีสิทธิ์เข้าใช้บริการระบบ โดยขั้นตอนในการสร้าง Query ดังกล่าว สามารถทำได้ดังนี้

1. เลือกคลาสย่อย ECustomer (ในกรณีเปิดหน้าต่าง Jasmine Class Browser ไว้แล้ว)
2. ทำการเลือก Tab Query ทาง Panel ขวามือ ซึ่งสามารถแสดงได้ดังรูปที่ 7.6



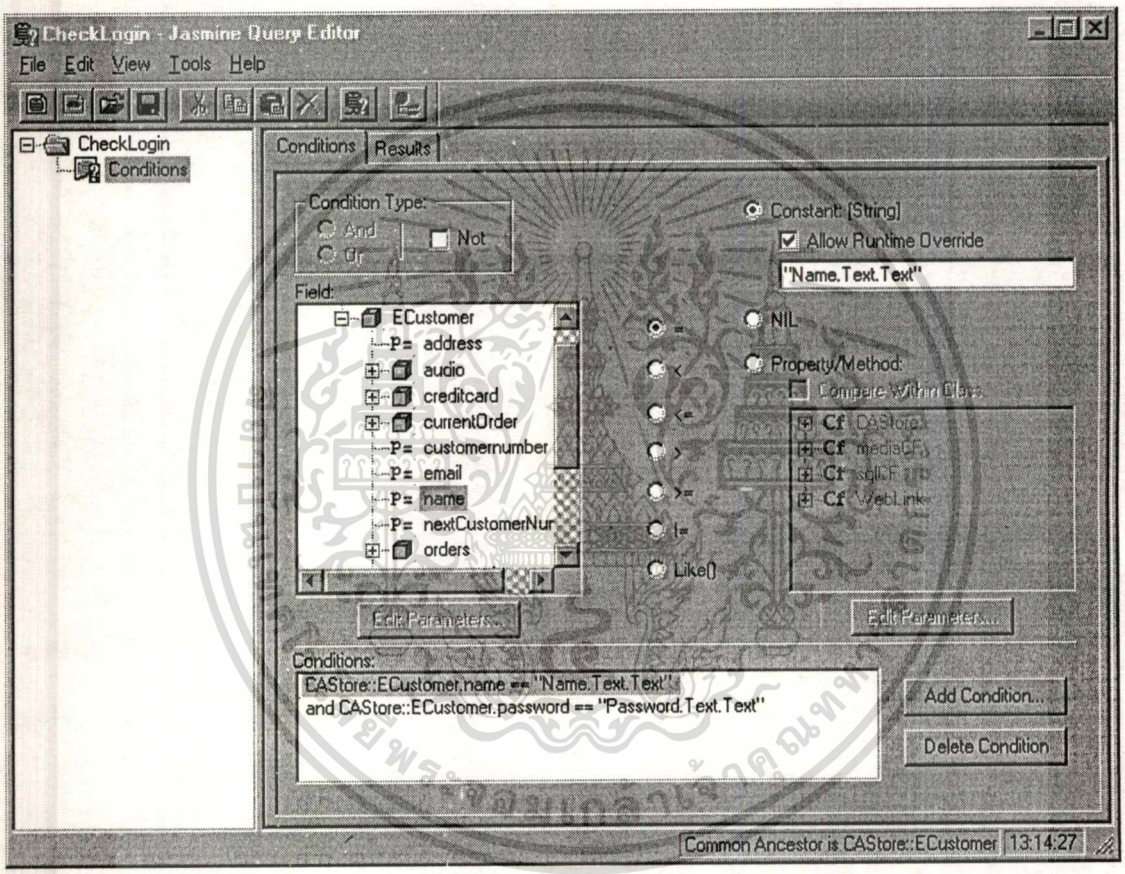
รูปที่ 7.6 แสดง Tab Query

3. ทำการเลือก File/New/Query (หรือ Right-Click ที่พื้นที่ว่างใน Pane ของTab Query แล้วทำการเลือก New Query จาก Context menu)

   หมายเหตุ : ก่อนการเปิด หน้าต่าง Query Editor และการสร้าง Query ใหม่ จำเป็นต้องเปิด Scene ของโปรแกรมประยุกต์นั้นๆ เนื่องจากการกำหนด Query จะเป็นการกำหนดไว้ใช้เฉพาะในโปแกรมประยุกต์ที่ต้องการใช้งาน Query นั้นๆ
4. ทำการเปิด Query Editor เพื่อกำหนดรายละเอียดที่ใช้ในการ Query ดังกล่าว โดยการเลือก File/Open (หรือ Right-Click ที่ Query ดังกล่าวแล้วทำการเลือก Open จาก Context menu)
5. ในส่วนของ Query Editor ผู้ใช้หรือผู้พัฒนาสามารถใช้ Editor ดังกล่าวได้ดังนี้
   - สามารถสร้าง(Construct) Query ใหม่หรือเปลี่ยนแปลงแก้ไข Query ที่มีอยู่ในระบบนั้นๆ แล้ว
   - สามารถดูผลที่ได้จากการ Query (โดยการเลือก Tab Result ในส่วนของ Query Editor)
   - สามารถดู ODQL Statement ที่ใช้ในการ Query ดังกล่าว

- สามารถทำการ Import และ Export Query ที่มีอยู่ในระบบนั้นๆ ได้
- สามารถสร้าง SubQuery ได้
- สามารถดึงออบเจ็กต์จากคลาสที่ถูกเลือกและออบเจ็กต์ทั้งหมดของคลาสย่อยหรือเฉพาะคลาสที่ถูกเลือกนั้นๆ

6. กำหนดรายละเอียดการ Query ดังกล่าวใน Query Editor ซึ่งสามารถแสดงได้ดังรูปที่ 7.7



รูปที่ 7.7 แสดงการกำหนดรายละเอียดของการ Query

7. ทำการบันทึก Query ที่สร้างขึ้นดังกล่าวโดยเลือก File/Save ในหน้าต่าง Query Editor ที่ใช้ในการสร้าง Query นั้นๆ

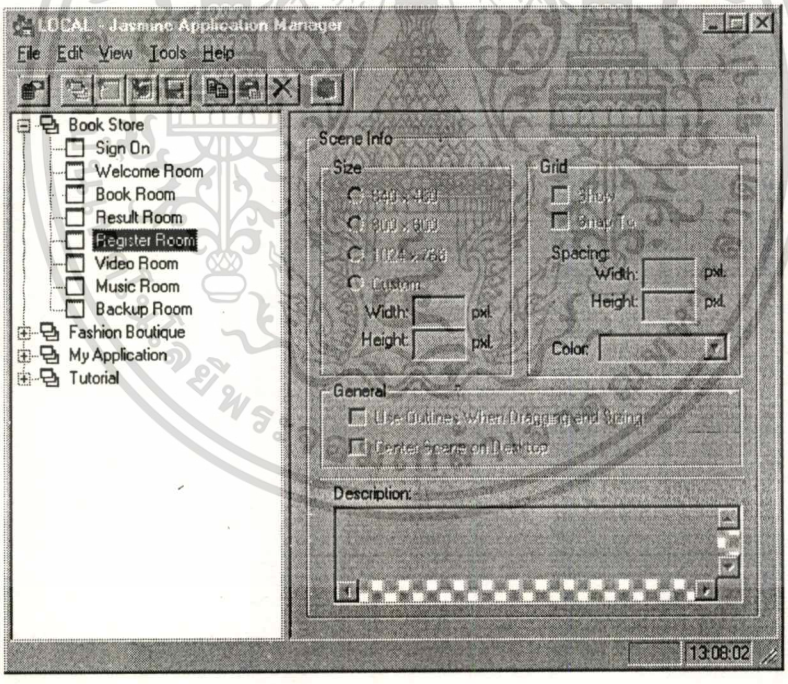
7.3.5 ตัวอย่างการสร้างโปรแกรมประยุกต์อย่างง่ายในระบบการจัดการฐานข้อมูลเชิงวัตถุ Jasmine (Create Simple Application on Jasmine Database)

ตัวอย่างการสร้างโปรแกรมประยุกต์อย่างง่ายด้วย Jasmine Studio ในส่วนนี้จะเป็นการอธิบายถึงการติดต่อกับคลาส, ออบเจ็กต์ภายในคลาส, เมธอด และการใช้ Query ที่ได้ทดลองสร้างขึ้นมา ซึ่งจะแบ่งเป็นการอธิบายเป็น 2 ส่วนคือ ส่วนของตัวอย่างการใช้เมธอดในการสร้างออบเจ็กต์ใหม่ในฐานข้อมูล และ ส่วนของตัวอย่างการ Query เพื่อตรวจสอบสิทธิ์ในการใช้งานระบบดังกล่าว

7.3.5.1 ตัวอย่างการใช้เมธอดในการสร้างออบเจ็กต์ใหม่ในฐานข้อมูล

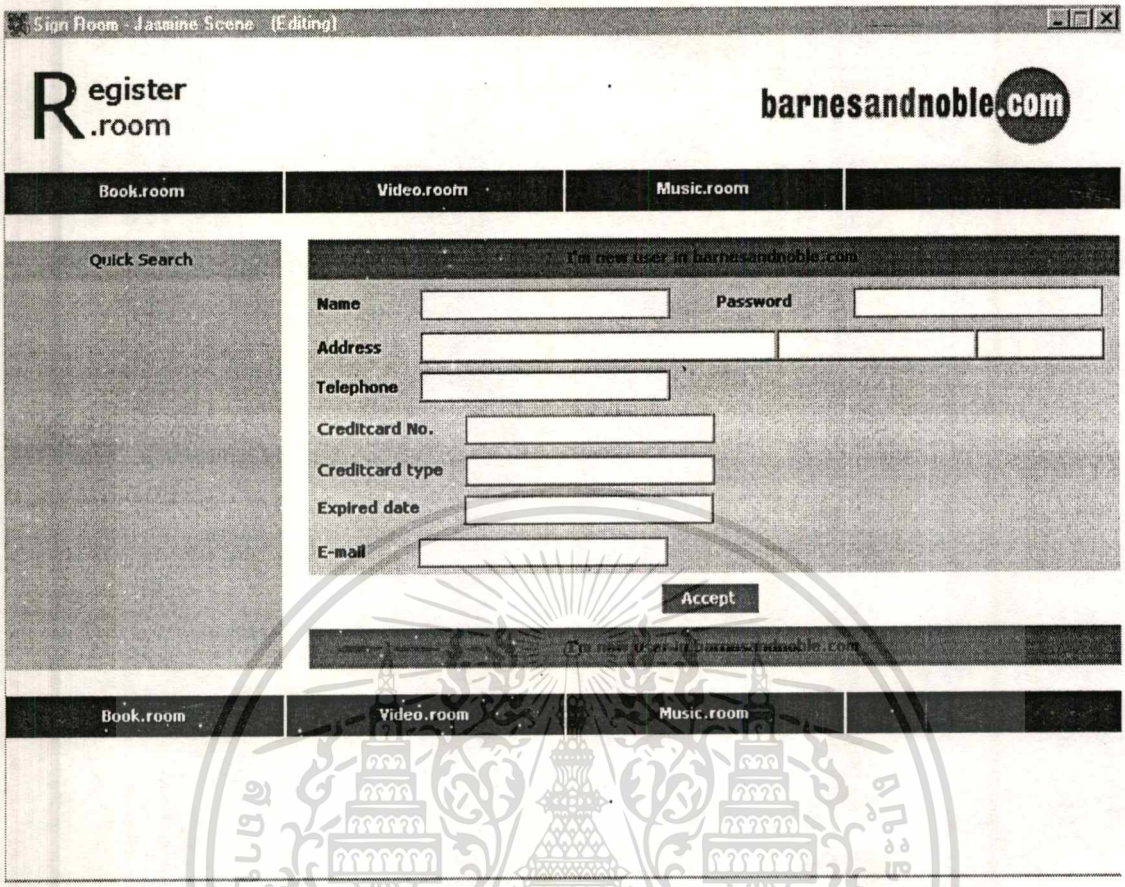
ในการสร้างตัวอย่างโปรแกรมประยุกต์อย่างง่ายโดยการใช้เมธอดในการสร้างหรือเพิ่มออบเจ็กต์ในฐานข้อมูล โดยขั้นตอนในการพัฒนาดังกล่าวสามารถทำได้ดังนี้

1. กำหนด Scene ที่ใช้ในตัวอย่างโปรแกรมประยุกต์โดยการสร้าง Scene ใหม่ได้ที่ หน้าต่าง Jasmine Application Manager ซึ่งสามารถแสดงได้ดังรูปที่ 7.8



รูปที่ 7.8 แสดงการกำหนด Scene ที่ใช้ในการพัฒนาโปรแกรมประยุกต์

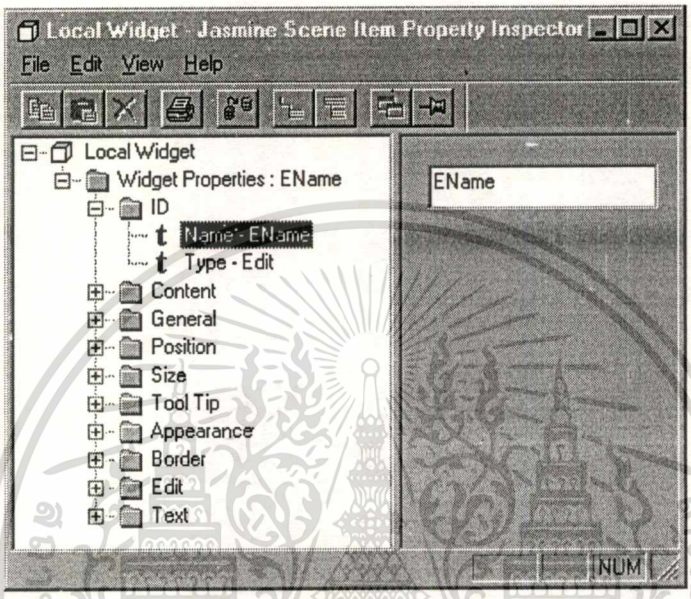
2. เปิด Scene ที่สร้างขึ้นโดยการเลือก File/Open และทำการกำหนดองค์ประกอบและลักษณะต่างๆ ภายใน Scene ดังกล่าว เช่น สีของพื้นหลัง, รูปแบบของ Interface ที่จะติดต่อกับผู้ใช้ ซึ่งสามารถแสดงได้ดังรูปที่ 7.9



รูปที่ 7.9 แสดง Scene ตัวอย่างการใช้เมธอดในการสร้างออบเจ็กต์ใหม่ในฐานข้อมูล

3. เปิดหน้าต่าง Jasmine Class Browser โดยการเลือก File/Database Administration (หรือการ Click ที่ปุ่ม Database Administration ที่ Tools Bar)
4. ในการกำหนด Interface นั้นสามารถกระทำได้โดยใช้ออบเจ็กต์ต่างๆ ที่มีใน Class Family (เช่น Local Widget) ที่สนับสนุนในระบบการจัดการฐานข้อมูลเชิงวัตถุ Jasmine
5. กำหนดส่วนที่ทำการรับข้อมูลเช่น ชื่อ, ที่อยู่ หรือ E-mail ของลูกค้า ซึ่งในที่นี้ใช้ออบเจ็กต์ Edit ใน Local Widget (ซึ่งเป็น Class Family ที่สนับสนุนในระบบการจัดการฐานข้อมูลเชิงวัตถุ Jasmine) โดยการลาก(Drag) ออบเจ็กต์ดังกล่าวมาวางบน Scene ที่ต้องการ
6. ทำการกำหนดชื่อของออบเจ็กต์ Edit ที่ใช้ในการรับข้อมูลจากลูกค้าให้ตรงตามชื่อของพารามิเตอร์ที่ใช้ในเมธอด(ในที่นี้จะใช้เมธอด addECustomer) โดยการ Right-Click ที่ออบเจ็กต์นั้นๆ บน Scene แล้วเลือก Object Property Inspector

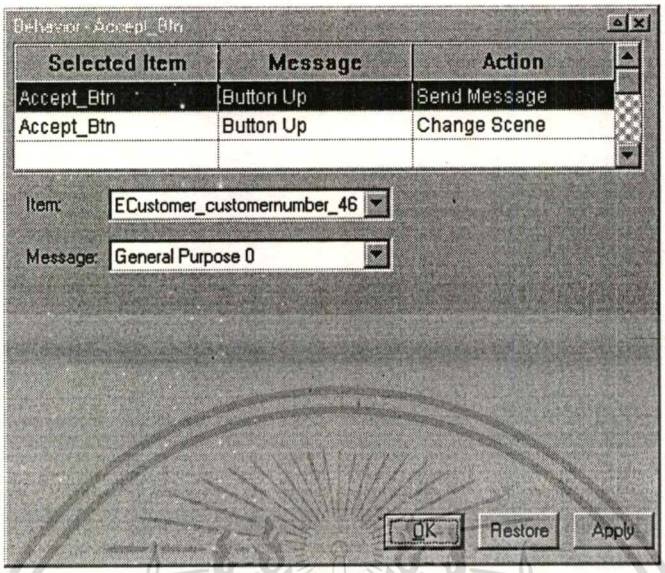
จาก Context menu เพื่อเข้าไปแก้ไขชื่อของออบเจ็กต์ ซึ่งการแก้ไขจะกระทำในส่วน Property ID (นอกจากนี้แล้วอาจเปลี่ยนแปลง Property ต่างๆ ตามความเหมาะสมเช่นลักษณะการ Transparency หรือรูปแบบของตัวอักษรที่ใช้เป็นต้น) ทั้งนี้การกำหนดชื่อของออบเจ็กต์ Edit สามารถแสดงได้ดังรูปที่ 7.10



รูปที่ 7.10 แสดงการกำหนดชื่อของออบเจ็กต์ Edit

7. ทำการสร้างปุ่มที่ใช้ในการยืนยันข้อมูลโดยใช้ออบเจ็กต์ Button ใน Class Family Local Widget
8. ทำการลากคลาส ECustomer จาก Jasmine Class Browser มาไว้บน Scene ที่สร้างขึ้นนั้น
9. ทำการกำหนดพฤติกรรม(Behavior) ของปุ่มที่ใช้ในการยืนยันโดย
   9.1 Right-Click ที่ออบเจ็กต์ปุ่มดังกล่าวแล้วเลือก Behavior จาก Context menu
   9.2 กำหนด Message ที่ใช้โดยเลือก Button Message/Button up
   9.3 กำหนด Action ของ Message ดังกล่าวโดยเลือก Message Actions/Send Message และทำการกำหนด Item ซึ่งหมายถึงออบเจ็กต์ใน Scene ที่รับ Message ดังกล่าว ซึ่งในที่นี้เลือก customer_number_x (x มีค่าไม่แน่นอนขึ้นกับขั้นตอนกำหนดลักษณะของ Interface โดยค่าของ x จะมีลักษณะ

เป็นตัวเลขลำดับ) และเลือก General Purpose 0 จาก Message Drop down list ด้านล่าง ซึ่งสามารถแสดงได้ดังรูปที่ 7.11



รูปที่ 7.11 แสดงการกำหนดพฤติกรรม(Behavior) ของออบเจ็กต์ Button (ตัวอย่างการใช้เมธอดในการสร้างออบเจ็กต์ใหม่ในฐานข้อมูล)

9.4 กำหนด Message ที่ใช้โดยเลือก Button Message/Button up

9.5 กำหนด Action ของ Message ดังกล่าวโดยเลือก Scene Actions/Change Sceneและทำการกำหนดค่า Scene ที่มีการอ้างอิงถึงโดยเลือก Book Room Sceneใน Drop down list ด้านล่าง

9.6 กดปุ่ม Apply และ OK ตามลำดับเพื่อกลับไปยัง Scene

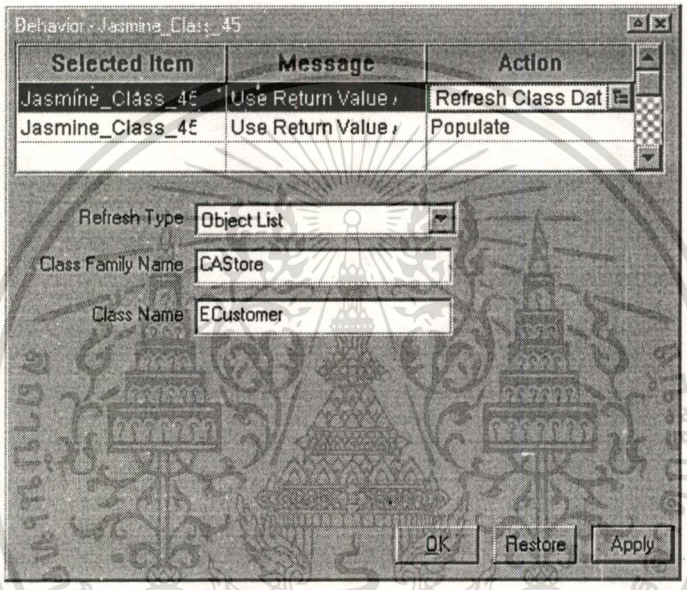
10. ทำการกำหนดพฤติกรรม(Behavior) ของ Class Anchor (Class ที่ลากมาวางบน Scene ดังกล่าว) โดย

10.1 Right-Click ที่ออบเจ็กต์ Class Anchor ใน Scene แล้วเลือก Behavior จาก Context menu

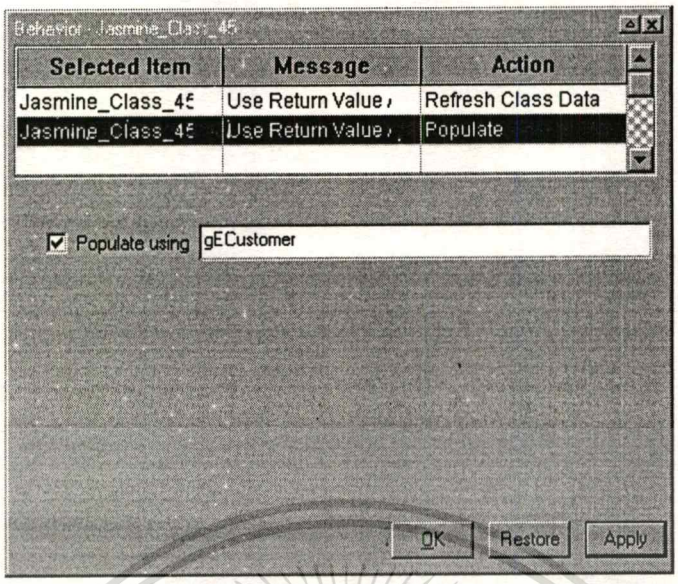
10.2 กำหนด Message ที่ใช้โดยเลือก General Purpose/General Purpose 0

10.3 กำหนด Action ของ Message ดังกล่าวโดยเลือก Class Method Actions /addECustomer และทำการกำหนดพารามิเตอรที่ใช้ในเมธอดดังกล่าวด้านล่าง โดยกำหนดค่าพารามิเตอร์ของ Name คือ Name.Text.Text และค่าพารามิเตอร์ของ Password คือ Password.Text.Text ทำการกำหนดจน

กระทั่งครบทุกค่าพารามิเตอร์(การกำหนด xxx.Text.Text เป็นการกำหนดให้ใช้ค่าของ Property Tex.Text ในออบเจ็กต์ Editor ที่ได้กำหนดไว้ข้างต้น) และเลือก ECustomer Class Anchor(เช่น Jasmine_Class_7)ใน Drop down list ด้านล่าง สุดท้ายทำการกำหนดตัวแปร Global ซึ่งในตัวอย่างนี้กำหนดตัวแปร Global ชื่อ gECustomer เพื่อเก็บค่าของออบเจ็กต์ที่ได้จากการใช้เมธอด addECustomer โดยกำหนดพฤติกรรมดังกล่าวสามารถแสดงได้ดังรูปที่ 7.12



รูปที่ 7.12 แสดงการกำหนดพฤติกรรม(Behavior) ของออบเจ็กต์ Class Anchor



รูปที่ 7.12 แสดงการกำหนดพฤติกรรม(Behavior) ของออบเจ็กต์ Class Anchor (ต่อ)

10.4 กำหนด Message ที่ใช้โดยเลือก Begin/End /Kickstart

10.5 กำหนด Action ของ Message ดังกล่าวโดยเลือก Disable/Enable Actions/Hide Scene Item

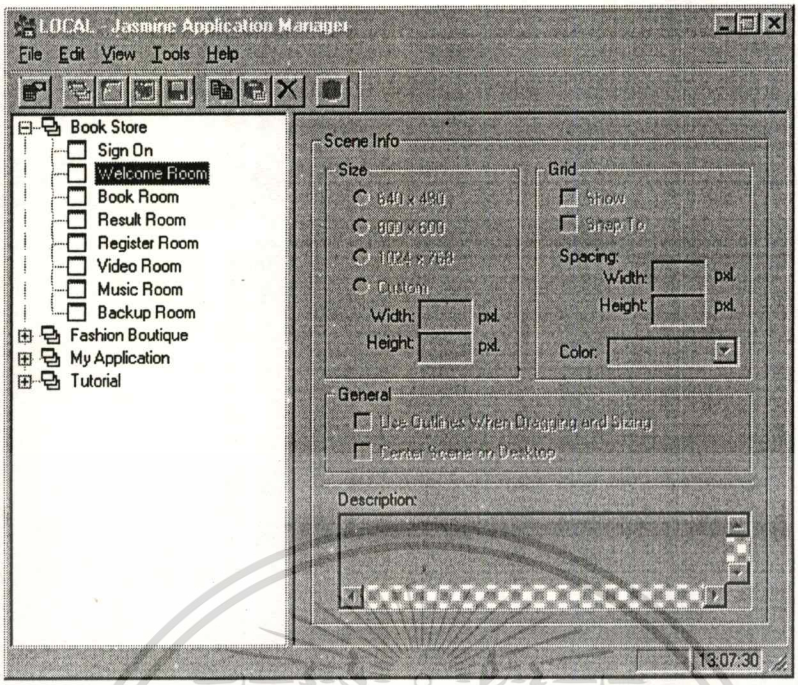
10.6 กดปุ่ม Apply และ OK ตามลำดับเพื่อกลับไปยัง Scene

11. ทำการทดสอบใช้งานโดยการเลือก Tools/Run ในหน้าต่าง Application Manager ทั้งนี้ก่อนการทดสอบใช้งานจำเป็นต้องบันทึกข้อมูลโดยการเลือก File/Save ก่อนเสมอ
12. ทดสอบการใช้งาน Scene ดังกล่าว
13. จบการทดสอบโดยการเลือก Tools/Stop ในหน้าต่าง Application Manager จากนั้นทำการตรวจสอบออบเจ็กต์ต่างๆที่สร้างขึ้นใหม่ในคลาส ECustomer (ในหน้าต่าง Jasmine Class Browser)

7.3.5.2 ตัวอย่างการใช้ Query ในการตรวจสอบสิทธิ์ในการใช้งานระบบ

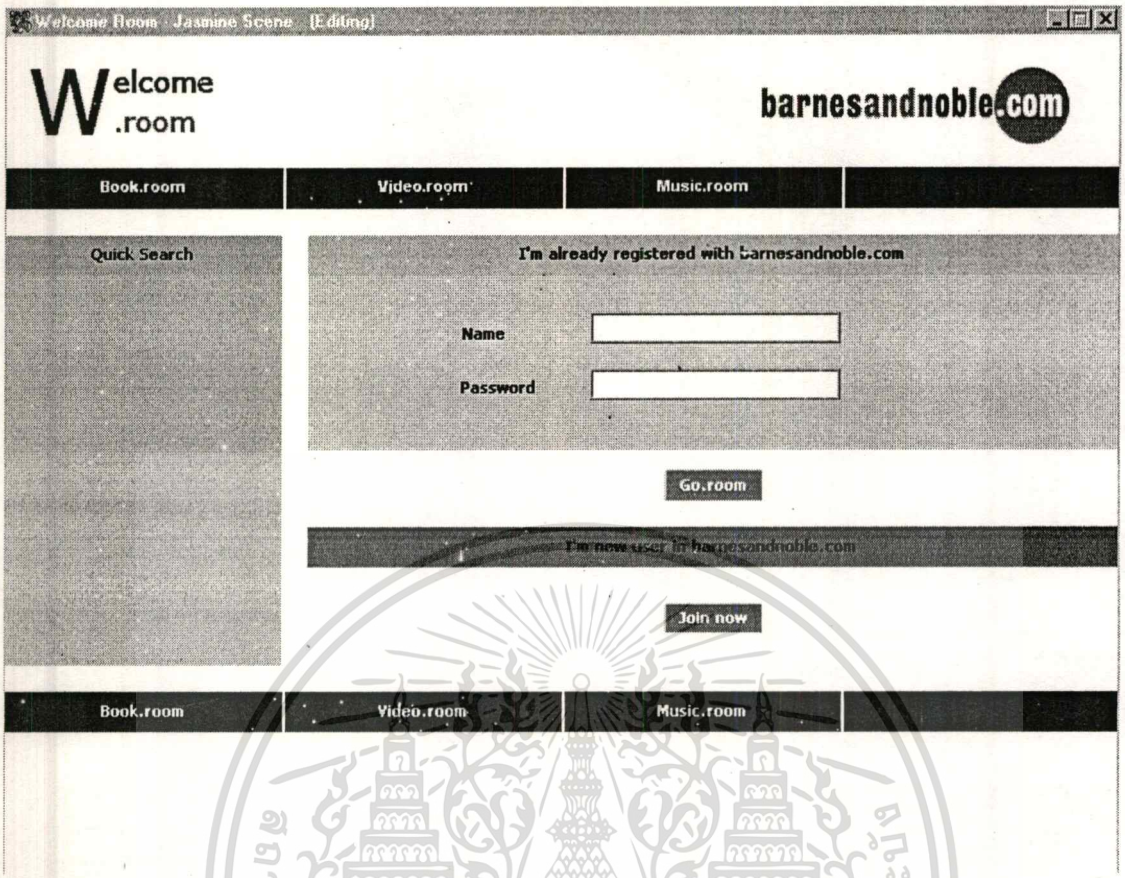
ในการสร้างตัวอย่างโปรแกรมประยุกต์อย่างง่าย โดยใช้ Query ช่วยตรวจสอบสิทธิ์ในการเข้าใช้ระบบดังกล่าว โดยขั้นตอนในการพัฒนาสามารถทำได้ดังนี้

1. กำหนด Scene ที่ใช้ในตัวอย่างโปรแกรมประยุกต์โดยการสร้าง Scene ใหม่ได้ที่ หน้าต่าง Jasmine Application Manager ซึ่งสามารถแสดงได้ดังรูปที่ 7.13



รูปที่ 7.13 แสดงการกำหนด Scene ที่ใช้ในการพัฒนาโปรแกรมประยุกต์

2. เปิด Scene ที่สร้างขึ้นโดยการเลือก File/Open และทำการกำหนดองค์ประกอบและลักษณะต่างๆ ภายใน Scene ดังกล่าว เช่น สีของพื้นหลัง, รูปแบบของ Interface ที่จะติดต่อกับผู้ใช้ ซึ่งสามารถแสดงได้ดังรูปที่ 7.14

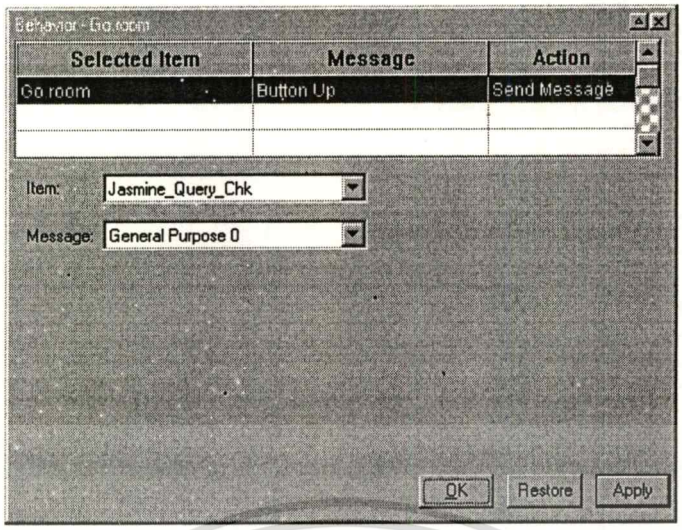


รูปที่ 7.14 แสดง Scene ตัวอย่างการใช้ Query ในการตรวจสอบสิทธิ์การใช้งานระบบ

3. เปิด หน้าต่าง Jasmine Class Browser โดยการเลือก File/Database Administration (หรือการ Click ที่ปุ่ม Database Administration ที่ Tools Bar)
4. ในการกำหนด Interface นั้นสามารถกระทำได้โดยใช้ออบเจ็กต์ที่มีอยู่ใน Class Family (เช่น Local Widget) ที่สนับสนุนในระบบการจัดการฐานข้อมูลเชิงวัตถุ Jasmine
5. กำหนดส่วนที่ทำการรับข้อมูลเช่น ชื่อ และ รหัสผ่านของลูกค้า ซึ่งในที่นี้ใช้ออบเจ็กต์ Edit ใน Local Widget (ซึ่งเป็น Class Family ที่สนับสนุนในระบบการจัดการฐานข้อมูลเชิงวัตถุ Jasmine) โดยการลากออบเจ็กต์ดังกล่าวมาวางบน Scene ที่ต้องการ
6. ทำการกำหนดชื่อของออบเจ็กต์ Edit ที่ใช้รับข้อมูลจากลูกค้าให้ตรงตามชื่อของพารามิเตอร์ที่ใช้ใน Query (ในกรณีที่พารามิเตอร์ที่กำหนดใน Query Editor ไม่ยอมให้มีการ Overwrite) โดยการ Right-Click ที่ออบเจ็กต์นั้นๆ บน

Scene แล้วเลือก Object Property Inspector จาก Context menu เพื่อเข้าไปแก้ไขชื่อของออบเจ็กต์ ซึ่งจะแก้ไขจะกระทำในส่วน Property ID (นอกจากนี้แล้วอาจเปลี่ยนแปลง Property ต่างๆ ตามความเหมาะสมเช่นลักษณะการ Transparency หรือรูปแบบของตัวอักษรที่ใช้ เป็นต้น)

7. ทำการสร้างปุ่มที่ใช้ในการยืนยันชื่อและรหัสผ่านโดยใช้ออบเจ็กต์ Button ใน Class Family Local Widget
8. ทำการลากคลาส ECustomer จาก Jasmine Class Browser มาไว้บน Scene ที่สร้างขึ้นนั้น
9. ทำการลาก Query ที่สร้างไว้ในตอนต้นจาก Tab Query (จากหน้าต่าง Jasmine Class Browser)
10. ทำการกำหนดพฤติกรรม(Behavior) ของปุ่มที่ใช้ในการยืนยันโดย
    10.1 Right-Click ที่ออบเจ็กต์ปุ่มดังกล่าวแล้วเลือก Behavior จาก Context menu
    10.2 กำหนด Message ที่ใช้โดยเลือก Button Message/Button up
    10.3 กำหนด Action ของ Message ดังกล่าวโดยเลือก Message Actions/Send Message และทำการกำหนด Item ซึ่งหมายถึงออบเจ็กต์ใน Scene ที่รับ Message ดังกล่าว ซึ่งในที่นี้เลือก Jasmine_Query_x (x มีค่าไม่แน่นอนขึ้นกับขั้นตอนกำหนดลักษณะของ Interface โดยค่าของ x จะมีลักษณะเป็นตัวเลขลำดับ) และเลือก General Purpose 0 จาก Message Drop down list ด้านล่าง ซึ่งสามารถแสดงได้ดังรูปที่ 7.15



รูปที่ 7.15 แสดงการกำหนดพฤติกรรม(Behavior) ของออบเจ็กต์ Button (ตัวอย่างการใช้ Query ในการตรวจสอบสิทธิ์การใช้งานระบบ)

10.4 กดปุ่ม Apply และ OK ตามลำดับเพื่อกลับไปยัง Scene

11. ทำการกำหนดพฤติกรรม(Behavior) ของ Query Anchor (Query ที่ลากมาวางบน Scene ดังกล่าว) โดย

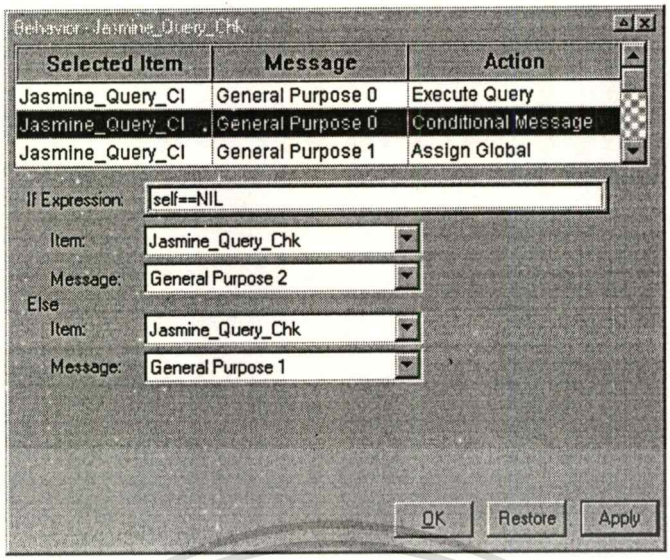
11.1 Right-Click ที่ออบเจ็กต์ Query Anchor ดังกล่าวแล้วเลือก Behavior จาก Context menu

11.2 กำหนด Message ที่ใช้โดยเลือก General Purpose/General Purpose 0

11.3 กำหนด Action ของ Message ดังกล่าวโดยเลือก Jasmine Actions/Execute Query และทำการกำหนดค่าพารามิเตอร์ด้านล่าง โดยกำหนดค่าพารามิเตอร์ของ EName คือ EName.Text.Text และค่าพารามิเตอร์ของ EPassword คือ EPassword.Text.Text

11.4 กำหนด Message ที่ใช้โดยเลือก General Purpose/General Purpose 0

11.5 กำหนด Action ของ Message ดังกล่าวโดยเลือก Condition Actions/ Conditional Message และทำการกำหนดเงื่อนไขต่างๆ ของ Conditional Message ด้านล่าง ซึ่งสามารแสดงได้ดังรูปที่ 7.16



รูปที่ 7.16 แสดงการกำหนดพฤติกรรม(Behavior) ของออบเจ็กต์ Query Anchor ในส่วนของ Condition Message

11.6 กำหนด Message ที่ใช้โดยเลือก General Purpose/General Purpose 1

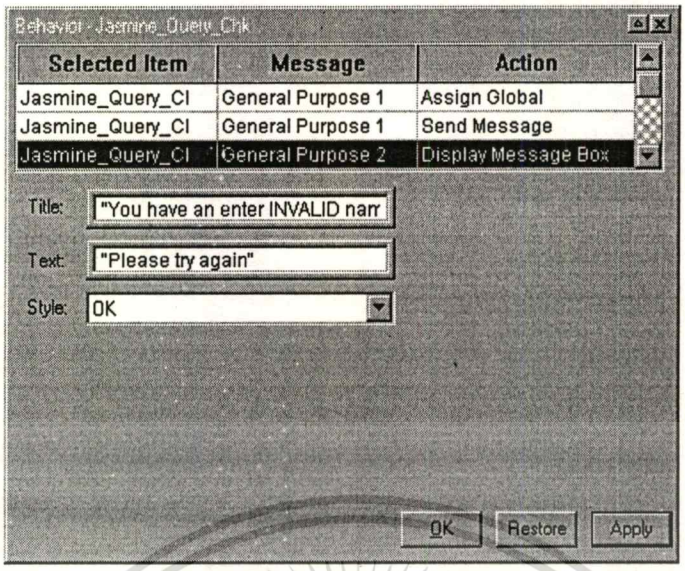
11.7 กำหนด Action ของ Message โดยเลือก Condition Actions/Assign Global และทำการกำหนดค่าพารามิเตอร์ด้านล่าง โดยกำหนดค่าพารามิเตอร์ของ Variable name ที่กำหนดให้เป็น Global Variable คือ login และค่าพารามิเตอร์ของ Expression คือ self

11.8 กำหนด Message ที่ใช้โดยเลือก General Purpose/General Purpose 1

11.9 กำหนด Action ของ Message โดยเลือก Message Actions/Send Message และทำการกำหนด Item ซึ่งแทนออบเจ็กต์ใน Scene ที่รับ Message ดังกล่าว ซึ่งในที่นี้เลือก Jasmine_Class_x และสุดท้ายเลือก General Purpose 0 จาก Message Drop down list ด้านล่าง

11.10 กำหนด Message ที่ใช้โดยเลือก General Purpose/General Purpose 2

11.11 กำหนด Action ของ Message โดยเลือกหน้าต่าง Actions/Display Message Box และทำการกำหนดค่าพารามิเตอร์ต่างๆ ด้านล่าง ซึ่งสามารถแสดงได้ดังรูปที่ 7.17



รูปที่ 7.17 แสดงการกำหนดพฤติกรรม(Behavior) ของออบเจ็กต์ Query Anchor ในส่วนของ Action Display Box

11.12 กดปุ่ม Apply และ OK ตามลำดับเพื่อกลับไปยัง Scene

12. การกำหนดพฤติกรรม(Behavior) ของ Class Anchor (Class ที่ลากมาวางบน Scene ดังกล่าว) โดย

12.1 Right-Click ที่ออบเจ็กต์ Class Anchor ดังกล่าวแล้วเลือก Behavior จาก Context menu

12.2 กำหนด Message ที่ใช้โดยเลือก General Purpose/General Purpose 0

12.3 กำหนด Action ของ Message ดังกล่าวโดยเลือก Jasmine Actions /Populate และเลือก Populate using พร้อมทั้งกำหนดค่าพารามิเตอร์ที่ใช้ คือ login

12.4 กำหนด Message ที่ใช้โดยเลือก General Purpose/General Purpose 0

12.5 กำหนด Action ของ Message ดังกล่าวโดยเลือก Timer Actions/Start Timer และทำการกำหนดค่าพารามิเตอร์ที่ใช้เท่ากับ 5

12.6 กำหนด Message ที่ใช้โดยเลือก Timer Messages/Timer Happened

12.7 กำหนด Action ของ Message ดังกล่าวโดยเลือก Timer Actions/Cancel Timer

12.8 กำหนด Message ที่ใช้โดยเลือก Timer Messages/Timer Happened

12.9 กำหนด Action ของ Message ดังกล่าวโดยเลือก Scene Actions/Change Sceneและทำการกำหนดค่า Scene ที่มีการอ้างอิงถึงโดยเลือก Book Room Sceneใน Drop down list ด้านล่าง

12.10 กดปุ่ม Apply และ OK ตามลำดับเพื่อกลับไปยัง Scene

13. ทำการทดสอบใช้งานโดยการเลือก Tools/Run บน หน้าต่าง Application Manager ทั้งนี้ก่อนการทดสอบใช้งานจำเป็นต้องบันทึกข้อมูลโดยการเลือก File/Save ก่อนเสมอ
14. ทดสอบใช้งาน Scene ดังกล่าว
15. จบการทดสอบโดยการเลือก Tools/Stop ในหน้าต่าง Application Manager

บทที่ 8

บทสรุปและแนวทางการพัฒนาระบบในอนาคต

8.1 บทสรุป

ระบบการจัดการฐานข้อมูลเชิงวัตถุ Jasmine เป็นตัวอย่างหนึ่งของระบบการจัดการฐานข้อมูลเชิงวัตถุ ซึ่งสามารถแก้ปัญหาหรือความไม่เหมาะสมต่างๆ ในระบบการจัดการฐานข้อมูลเชิงสัมพันธ์ โดยกลไกที่สนับสนุนการทำงานเชิงวัตถุ เช่น Abstraction Mechanism ซึ่งเป็นกลไกที่สนับสนุนการกำหนดรูปแบบข้อมูลรวมทั้งการกำหนดการจัดการในการทำงานของรูปแบบข้อมูลดังกล่าวโดยผู้ใช้เอง(User-Defined) นอกจากนี้ยังสามารถใช้แนวคิดเกี่ยวกับ Complex Object และ Object Identity ในการจัดการข้อมูลที่มีความซับซ้อนหรือข้อมูลที่มีลักษณะเป็นโครงสร้าง

จากที่กล่าวมาเป็นเพียงแนวความคิดที่ใช้ในการจัดการปัญหาที่เกิดขึ้นในระบบการจัดการฐานข้อมูลเชิงสัมพันธ์ในงานระบบสารสนเทศ ซึ่งข้อดีต่างๆ ของแนวคิดเชิงวัตถุที่สนับสนุนในระบบการจัดการฐานข้อมูลเชิงวัตถุนอกจากที่กล่าวมาแล้วยังมีอีกมากมาย เช่น ความสามารถในการ Extensibility, ความสามารถในการ Reusability และความยืดหยุ่นของ Type Definition เป็นต้น

สำหรับตัวอย่างโปรแกรมประยุกต์ที่พัฒนาด้วย Jasmine Studio นั้นเป็นตัวอย่างการนำเอาแนวคิดเชิงวัตถุที่สนับสนุนในระบบการจัดการฐานข้อมูลเชิงวัตถุ มาแก้ไขปัญหาหรือความไม่เหมาะสมต่างๆ ในระบบการจัดการฐานข้อมูลเชิงสัมพันธ์ ซึ่งปัญหาหรือความไม่เหมาะสมดังกล่าวเป็นปัญหาที่พบในงานระบบสารสนเทศในปัจจุบัน

8.2 แนวทางการพัฒนาระบบในอนาคต

แนวทางในการพัฒนาระบบในอนาคต. อาจทำได้โดยการเพิ่มหน้าที่การทำงานต่างๆ ของระบบให้สมบูรณ์ครบถ้วน ทั้งในด้านการตรวจสอบข้อมูลต่างๆ ในส่วนการสั่งซื้อสินค้า, การจัดการกลุ่มข้อมูลในระบบ รวมทั้งการพัฒนาให้เป็นโปรแกรมประยุกต์ Web Enabled Application โดยสนับสนุนการทำงานต่างๆ บนระบบเครือข่าย Internet ได้อย่างสมบูรณ์แบบ เป็นต้น

บรรณานุกรม

Bancilhon, F. " Object Databases." ACM Computing Surveys, vol. 28 no. 1(March 1996) pp. 137-140.

Date, C. J. An Introduction to Database Systems, Sixth Edition. New York: Addison-Wesley, 1995.

Dayal, U. " Simplifying Complex Objects: The PROBE Approach to Modeling and Querying Them." Computer Corporation of America. Massachusetts: Cambridge, 1997.

Hughes, G. J. Object-Oriented Database. New York: Prentice Hall, 1991.

Ishikawa, H. and others. "An Object-Oriented Database System Jasmine: Implementation, Application, and Extension." IEEE Transaction on Knowledge and Data Engineering. vol. 8 no. 2(April 1996) pp. 285-304.

Ishikawa, H. and others. "The Model, Language, and Implementation of an Object-Oriented Multimedia Knowledge Base Management System." ACM Transaction on Database Systems. vol. 18 no. 1(March 1993) pp. 1-50.

Jasmine version 1.2 Getting Started, Tutorial and Concepts. [CD-ROM]. Abstract from : Computer Associates File: Jasmine Developer Edition: CDR03110329E, 1998.

Kato, Y. and others. "Data Object creation and Display Techniques for the Huge Database of Subscriber Cable Networks." Optical Subscriber Cable Systems Laborator NTT Telecommunication Field Systems R&D Center. Ibaraki: Tokai, 1997.

Khoshafian, S. Object-Oriented Databases. New York: John Wiley and Sons, 1993.

Khoshafian, S. and others. The Jasmine Object Database Multimedia Applications for the Web. California: Morgan Kaufmann Publishers, 1999.

Liskov, B. and others. "Abstraction Mechanisms in CLU." SIGPLAN/SIGOPS/SICSOFT Conference on Language Design for Reliable Software. Raleigh N.C., 1997.

Maier, D. Comments on The Third-Generation Data Base System Manifestos. Oregon Graduate Institute, 1991.

Maier, D. and J. Stenia. Research in Object-Oriented Database System: Combining Language and Database Advances in an Object-Oriented Development. California: Morgan Kaufmann Publishers, 1990.

McClure, S. "Object Database vs. Object-Relational Databases." IDC Bulletin #14821E, 1997.

Monala, F. "An Evaluation of Object-Oriented DBMS Developments 1994 Edition." GTE Laboratories Incorporated Report. Waltham, 1994.

Robie, J. and Dirk Bartels. "A Comparison between Relational and Object-Oriented Databases for Object-Oriented application development." White Paper, 1995.

Saake, G., S. Conrad, I. Schmitt and C.Turker. "Object-Oriented Database Design: What is the Difference with Relation Database Design?" Object World Frankfurt'95-Conference Notes, Wednesday, Software Development Track SD.6, 1995.

Stonebreaker, M. "Inclusion of New Types in Relational Data Base Systems." Proc. Intl. Conf. On Data Engineering. : IEEE, 1986.

Sujithan, K. R. "An Object Model of Data, Based on the ODMG Industry Standard for Database Applications." The Institution of Electrical Engineers. U.K. : The Institution of Electrical Engineers (IEE), 1995.

Vanden Berg, C. A. ODMG user guide 1.0, 1995.

ภาคผนวก

ภาคผนวก ก

Installing and Setting up a Windows NT or Windows 95 Jasmine Client

ในบทนี้จะเป็นการกล่าวถึงความต้องการของระบบ, การติดตั้ง(Installing) รวมทั้งการเริ่มการทำงานและการจบการทำงานของ Jasmine บน PC Workstation ที่ใช้ระบบปฏิบัติการ Windows NT 4.0 หรือ Windows 95

1. System Requirements

1.1 Hardware Requirements:

- เครื่องคอมพิวเตอร์ Multimedia PC ที่ใช้หน่วยประมวลผลกลาง Pentium
- หน่วนความจำสำรอง(RAM) 32 MB (แนะนำให้ใช้ 64 MB)
- พื้นที่ว่างบนหน่วยความจำหลัก(Hard Disk) 200 MB (แนะนำให้เหลือพื้นที่ว่างบนหน่วยความจำหลัก 300 MB)

การกำหนดพื้นที่ว่างในหน่วยความจำหลักจะขึ้นอยู่กับประเภทของการ Install ซึ่งจะมีการแสดงถึงพื้นที่ว่างที่ต้องการและพื้นที่จริงที่เหลืออยู่ในขณะนั้น ในขั้นตอนการ Install ทั้งนี้ถ้ามีพื้นที่ว่างในหน่วยความจำหลักดังกล่าวมาจะส่งผลให้สามารถใช้ประสิทธิภาพของผลิตภัณฑ์ดังกล่าวได้อย่างเต็มที่

1.2 Software Requirements:

- ในการติดตั้งองค์ประกอบเกี่ยวกับระบบฐานข้อมูลเชิงวัตถุมีความจำเป็นต้องติดตั้งบนระบบปฏิบัติการ Windows NT 4.0 และสำหรับองค์ประกอบอื่นๆ นอกจากนี้มีสามารถลงบนระบบปฏิบัติการ Windows NT 4.0 หรือ Windows 95
- ในการแอ็กเซสระบบการจัดการฐานข้อมูลเชิงวัตถุ Jasmine แบบ Remote นั้นจำเป็นต้องใช้ Protocol TCP/IP ในการเชื่อมต่อในระบบเครือข่าย
- มีความต้องการ Microsoft Visual C++ Compiler เวอร์ชัน 4.2 หรือสูงกว่าเพื่อใช้ในการคอมไพล์เมธอดในระบบการจัดการฐานข้อมูลเชิงวัตถุ Jasmine และเพื่อใช้เครื่องมือในการพัฒนา Jasmine C API

2. Basic Install Steps

ขั้นตอนในการติดตั้ง Jasmine สามารถทำตามขั้นตอนได้ดังนี้

1. Logon เข้าสู่ระบบปฏิบัติการ Windows NT 4.0 หรือ Windows 95
2. ใส่แผ่น CD-ROM ของ Jasmine ในช่องอ่าน CD
3. ถ้า Jasmine Setup Wizard ไม่สามารถทำงานได้อย่างอัตโนมัติให้ทำการ Double-Click ที่ CD-ROM Drive Letter
4. ในกรณีที่มีปัญหาในการลง Jasmine สามารถดูรายละเอียดต่างๆ ได้ใน Online Help

3. Starting and Stopping Jasmine

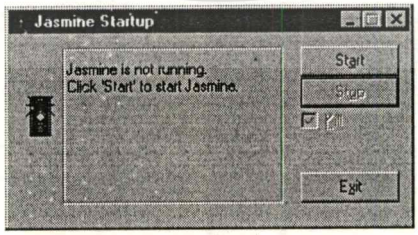
Jasmine จำเป็นต้องเริ่มต้น(Start) การทำงานเพื่อ

- Run โปแกรมประยุกต์ Jasmine end-user
- Run คำสั่ง(Command) Jasmine System เช่น codqlie ซึ่งจะเป็นการเริ่มการใช้งาน ODQL Interpreter)
- ใช้ Jasmine Studio ในการสร้างโปรแกรมประยุกต์บน Jasmine หรือการจัดการข้อมูลในฐานข้อมูล Jasmine

3.1 Starting Jasmine Manually

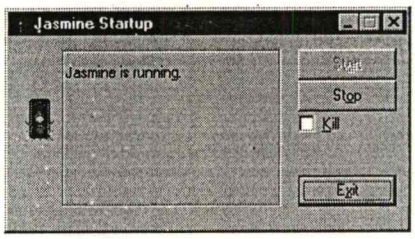
ในกรณีที่ไม่ได้ทำการเริ่มต้นบริการหรือการ Start Service ของ Jasmine อย่างอัตโนมัติเมื่อระบบเริ่มทำงาน สามารถ Start Service ดังกล่าวได้ตามขั้นตอนดังนี้

1. Click Start บน Taskbar จากนั้นเลือก Program, Jasmine, Jasmine Startup เพื่อแสดง Jasmine Startup Dialog



รูปที่ ก.1 แสดง Jasmine Startup Dialog (ก่อนการ Start Service)

2. Click Start จากนั้น Dialog จะมีสัญญาณการเปลี่ยนแปลงสถานะ Stop-> Start



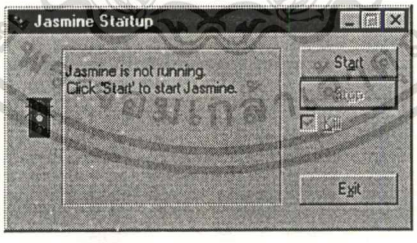
รูปที่ ก.2 แสดง Jasmine Startup Dialog (หลังการ Start Service)

3. Click Exit เพื่อปิด Dialog ดังกล่าว

3.2 Stopping Jasmine

ในการ Start Service ของ Jasmine นั้นจะเป็นการ Run Jasmine มีการหยุดหรือ Stop Service ดังกล่าวซึ่งเป็นการ Stop แบบ Manual หรืออาจจะ Stop เนื่องมาจากการ Shut Down ระบบ ทั้งนี้การ Stop Jasmine แบบ Manual สามารถทำตามขั้นตอนได้ดังนี้

1. Click Start บน Taskbar จากนั้นเลือก Program, Jasmine, Jasmine Startup เพื่อแสดง Jasmine Startup Dialog
2. Click Stop จากนั้น Dialog จะมีสัญญาณการเปลี่ยนแปลงสถานะ Start-> Stop



รูปที่ ก.3 แสดง Jasmine Startup Dialog (หลังการ Stop Service)

3. Click Exit เพื่อปิด Dialog ดังกล่าว

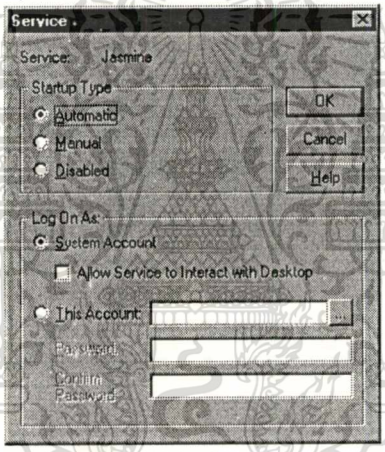
Important!!! ในส่วนของ Kill Check Box นั้นจะถูกใช้เมื่อมีความผิดปรกติอันเนื่องมาจาก Shut Down Jasmine แต่ก็สามรถใช้ได้ตามความเหมาะสมกับเหตุ

การณ์ที่เกิดขึ้นต่างๆ เช่น การ Hang หรือการที่ไม่สามารถปิดการทำงานด้วยปุ่ม Stop ซึ่งสามารถทำการปิดได้โดยการเลือก Kill Check Box และ Click Stop

3.3 Automatic the Jasmine Startup

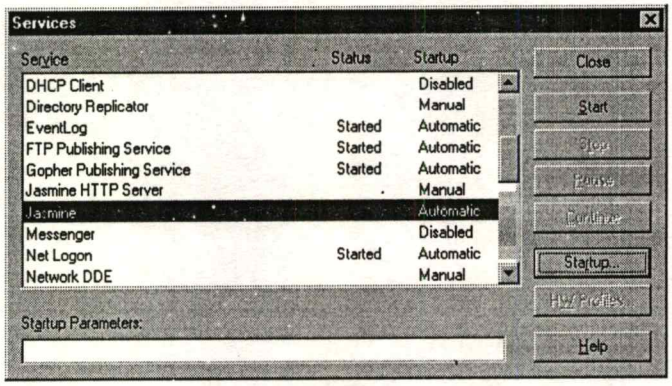
ในการเริ่มใช้งาน Jasmine สามารถกำหนดให้มีการ Start Service ของ Jasmine เมื่อเริ่มการใช้งานระบบ โดยที่ไม่จำเป็นต้องทำการ Start Service ดังกล่าวในแบบ Manual ทั้งนี้ การ Start Serviceแบบ Automatic สามารถทำตามขั้นตอนได้ดังนี้

1. Double-Click ที่ไอคอน Service ใน Control Panel.
2. เลือก Jasmine จากรายการของ Service
3. Click ปุ่ม Startup เพื่อแสดง Service Dialog



รูปที่ ก.4 แสดง Service Dialog

4. เลือก Automatic ใน Startup Type จากนั้นทำการ Click OK
   ซึ่งจะปรากฏ Service Dialog ที่แสดงค่า Automatic ในคอลัมน์ Startup ของ Jasmine Service



รูปที่ ก.5 แสดงสถานะของ Jasmine Service

5. Click Close เพื่อปิด Service Dialog ดังกล่าว
6. Service ที่มีความสัมพันธ์กับ Jasmine Service คือ Jasmine HTTP Server และ WebLink Server ซึ่งถ้ามีการติดตั้งองค์ประกอบดังกล่าว ซึ่งจะถ้ามีการเลือกให้ Jasmine Service เริ่มการทำงานแบบ Automatic แล้วนั้น Service ทั้ง 2 ดังกล่าว จะเปลี่ยนเป็น Automatic ด้วย แต่ถ้าไม่ต้องการให้ Service ดังกล่าวเป็นแบบ Automatic สามารถทำได้เหมือนการกำหนดใน Jasmine Service โดยเลือก Service ที่เหมาะสมในขั้นตอนที่ 2

ภาคผนวก ข

ตัวอย่างระบบงานต้นแบบ(Prototype)

ในส่วนนี้จะเป็นการอธิบายลักษณะการทำงานของระบบต้นแบบ(Prototype) ที่ใช้แนวความคิดเชิงวัตถุที่สนับสนุนในระบบการจัดการฐานข้อมูลเชิงวัตถุ ซึ่งเป็นแนวทางในการแก้ไขปัญหาหรือความไม่เหมาะสมที่พบในระบบการจัดการฐานข้อมูลเชิงสัมพันธ์ โดยในที่นี้เป็นต้นแบบที่พัฒนาขึ้นด้วย Jasmine Studio ซึ่งเป็นเครื่องมือที่ใช้ในการพัฒนาโปรแกรมประยุกต์บนระบบการจัดการฐานข้อมูลเชิงวัตถุ Jasmine

1. ลักษณะการทำงานของต้นแบบ

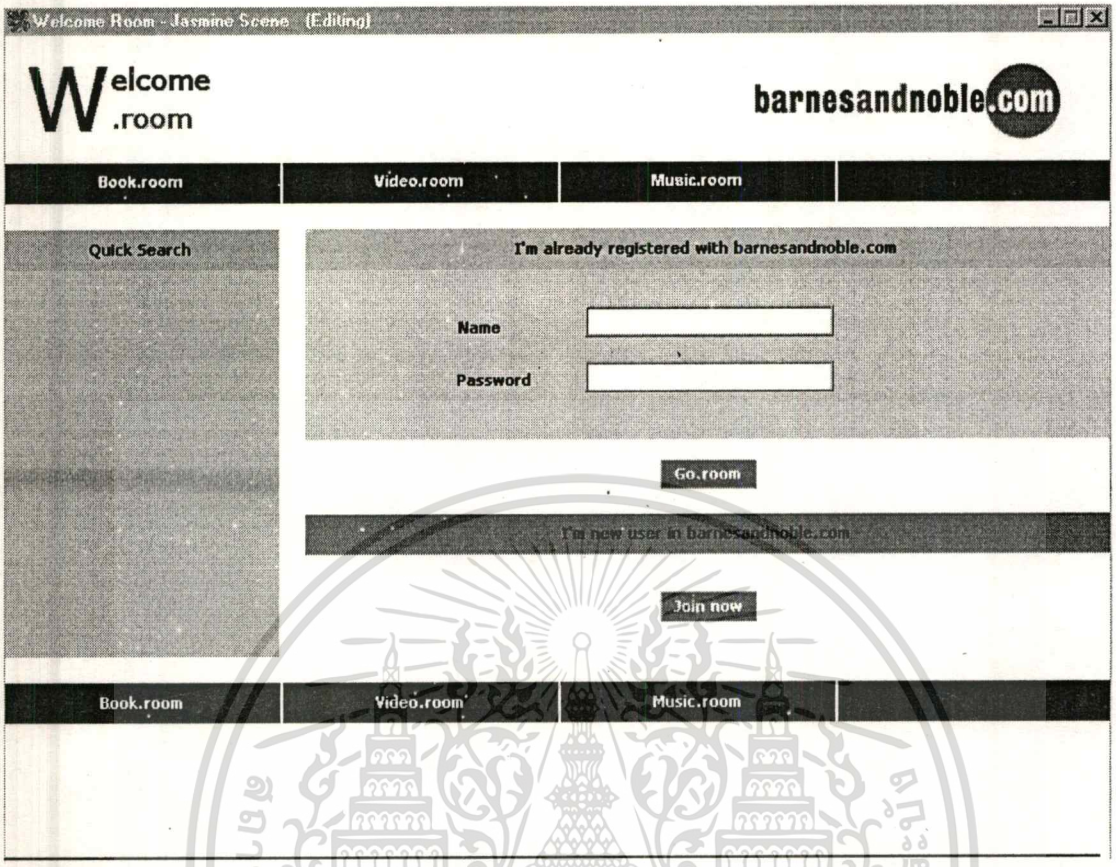
การทำงานของต้นแบบแบ่งออกเป็น 3 ส่วนด้วยกันคือ

- ส่วนการทำงานเกี่ยวกับตรวจสอบสิทธิ์ของผู้เข้าใช้ระบบ
- ส่วนการทำงานเกี่ยวกับการป้อนข้อมูลใหม่เข้าสู่ระบบ
- ส่วนการทำงานเกี่ยวกับการสืบหาข้อมูล

2. รายละเอียดการทำงานของต้นแบบ

2.1 รายละเอียดการทำงานเกี่ยวกับการตรวจสอบสิทธิ์ผู้เข้าใช้ระบบ

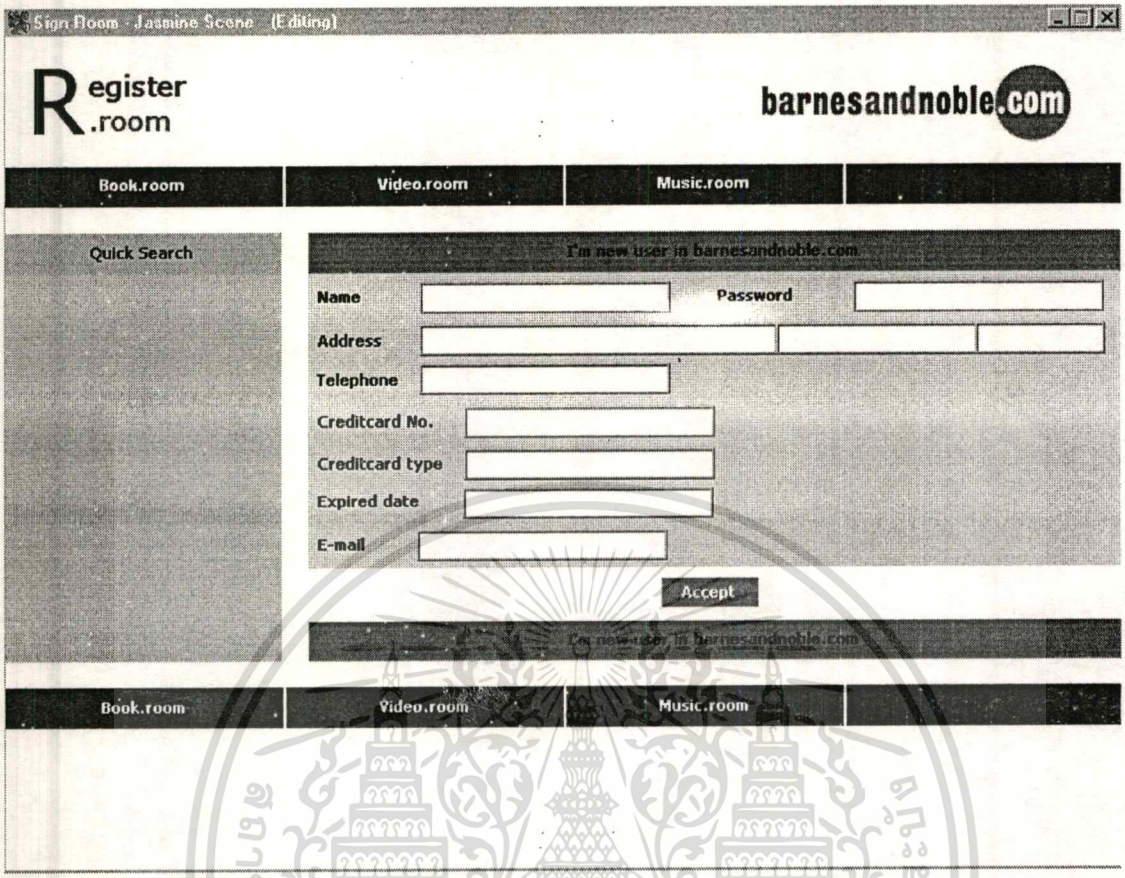
การทำงานในส่วนนี้ของต้นแบบจะเป็นการตรวจสอบสิทธิ์ของผู้เข้าใช้ระบบ ซึ่งจะใช้ลักษณะการทำงานในแบบ Name/Password โดยผู้ใช้ที่ได้ทำการลงทะเบียนแล้วสามารถเข้าไปใช้งานระบบต้นแบบดังกล่าวได้ ทั้งนี้หน้าจอการทำงานในส่วนดังกล่าวสามารถแสดงได้รูปที่ ข.1



รูปที่ ข.1 แสดงส่วนการทำงานเกี่ยวกับการตรวจสอบสิทธิ์ของผู้เข้าใช้ระบบ

2.2 รายละเอียดการทำงานเกี่ยวกับการป้อนข้อมูลใหม่เข้าสู่ระบบ

การทำงานในส่วนนี้ของต้นแบบจะเป็นการป้อนข้อมูลของผู้ที่เข้ามาทำการลงทะเบียนเพื่อขอใช้บริการระบบต้นแบบดังกล่าว โดยผู้ลงทะเบียนเลือกปุ่ม Join ในส่วนของการตรวจสอบสิทธิ์ เพื่อเข้าสู่หน้าจอการลงทะเบียน ทั้งนี้หน้าจอการทำงานในส่วนดังกล่าวสามารถแสดงได้รูปที่ ข.2



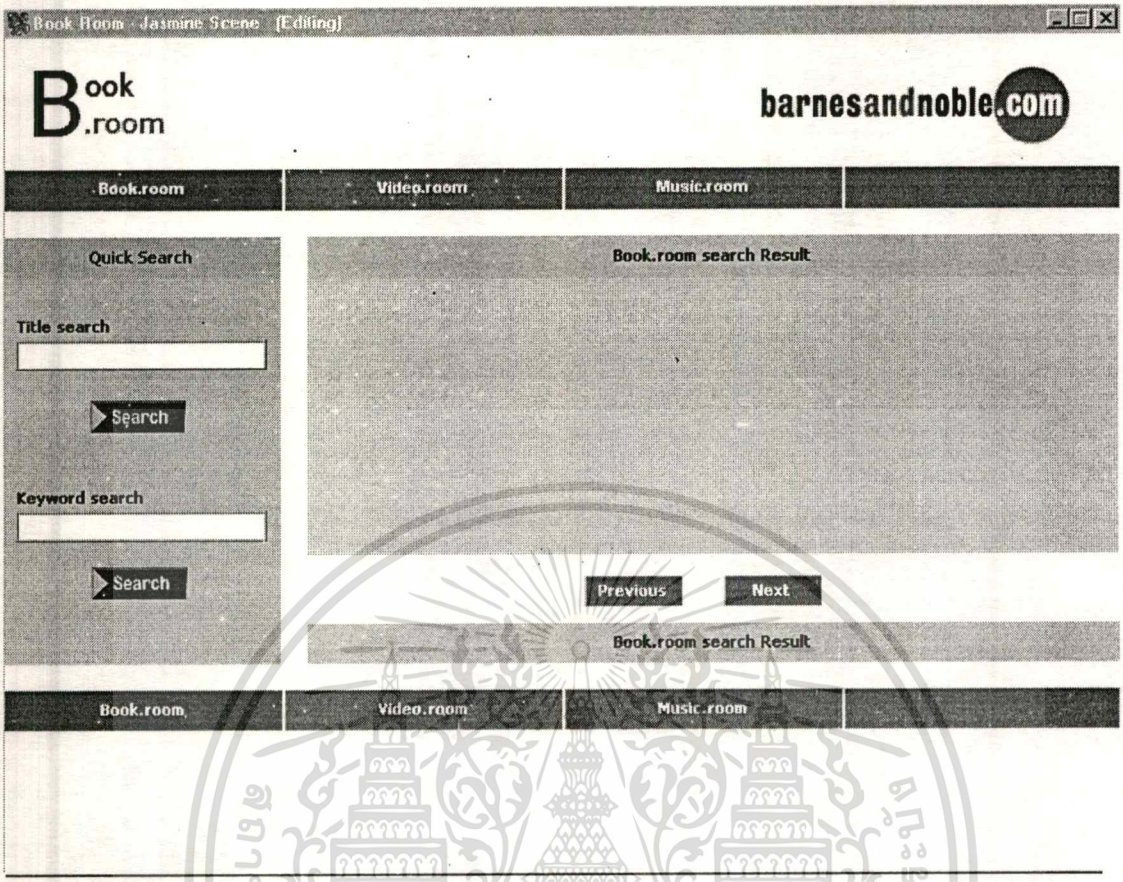
รูปที่ ข.2 แสดงส่วนการทำงานเกี่ยวกับการป้อนข้อมูลใหม่เข้าสู่ระบบ

2.3 รายละเอียดการทำงานเกี่ยวกับการสืบหาข้อมูล

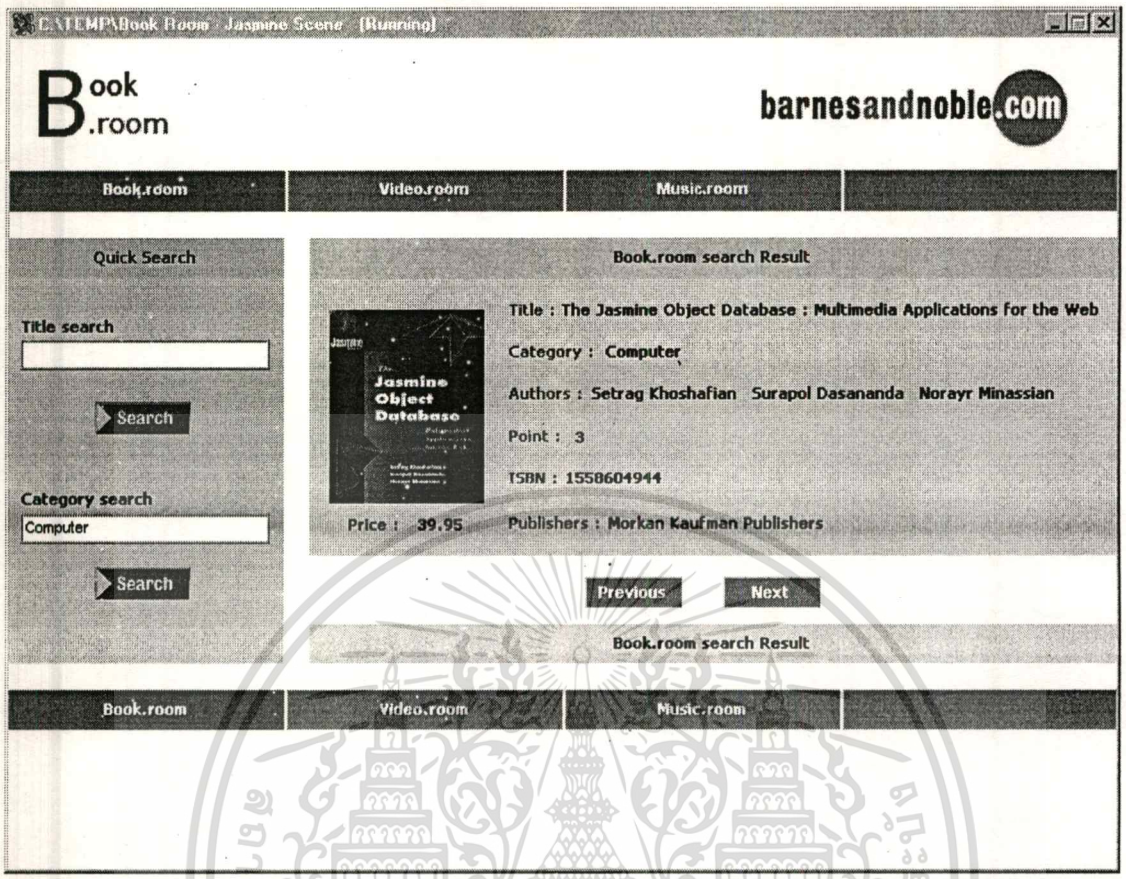
การทำงานในส่วนนี้ของต้นแบบจะเป็นการให้บริการในการสืบหาข้อมูล เกี่ยวกับหนังสือ, วีดีโอ และเพลง ซึ่งสมาชิกสามารถทำการสืบหาข้อมูลได้ 2 แนวทางด้วยกันคือ

- การสืบหาข้อมูลจากรายชื่อ(Title)
- การสืบหาข้อมูลตามหมวดหมู่(Category)

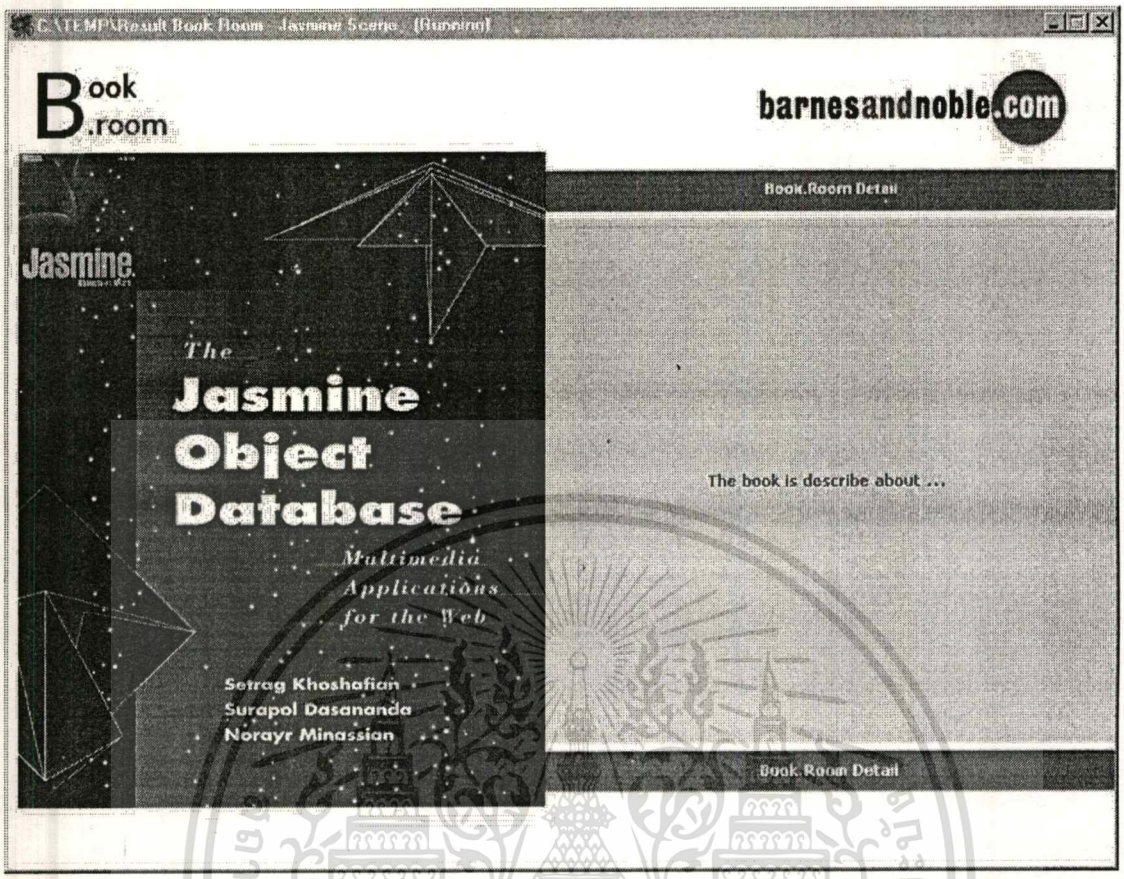
โดยการสืบหาดังกล่าวผู้ใช้สามารถเรียกดูข้อมูลตัวอย่างเช่น รายละเอียดโดยย่อของหนังสือ, Clip วีดีโอตัวอย่าง และเพลงตัวอย่าง ได้โดยการเลือก Detail ของข้อมูลที่สืบหาได้นั้นๆ ทั้งนี้หน้าจอการทำงานในส่วนดังกล่าวสามารถแสดงได้รูปที่ ข.3 ถึง รูปที่ ข.7



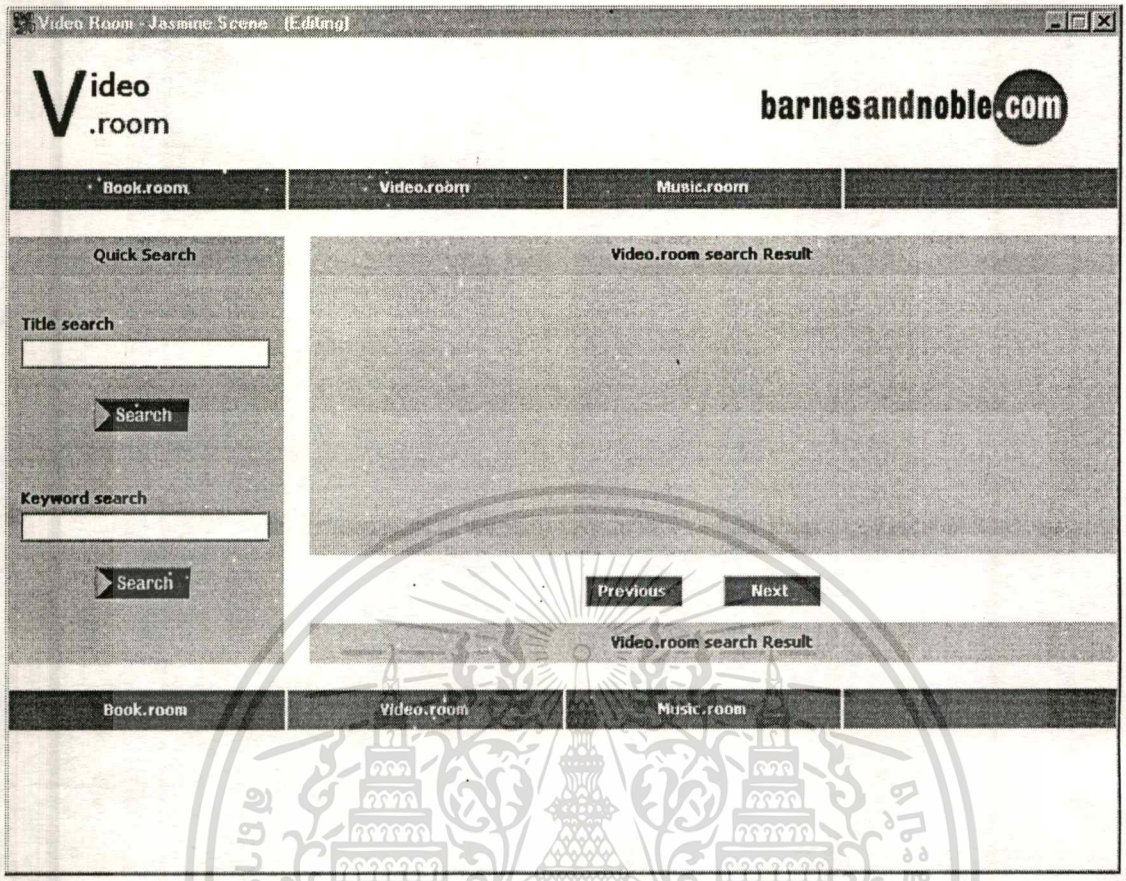
รูปที่ ข.3 แสดงส่วนการทำงานเกี่ยวกับการสืบหาข้อมูลในส่วนของหนังสือ



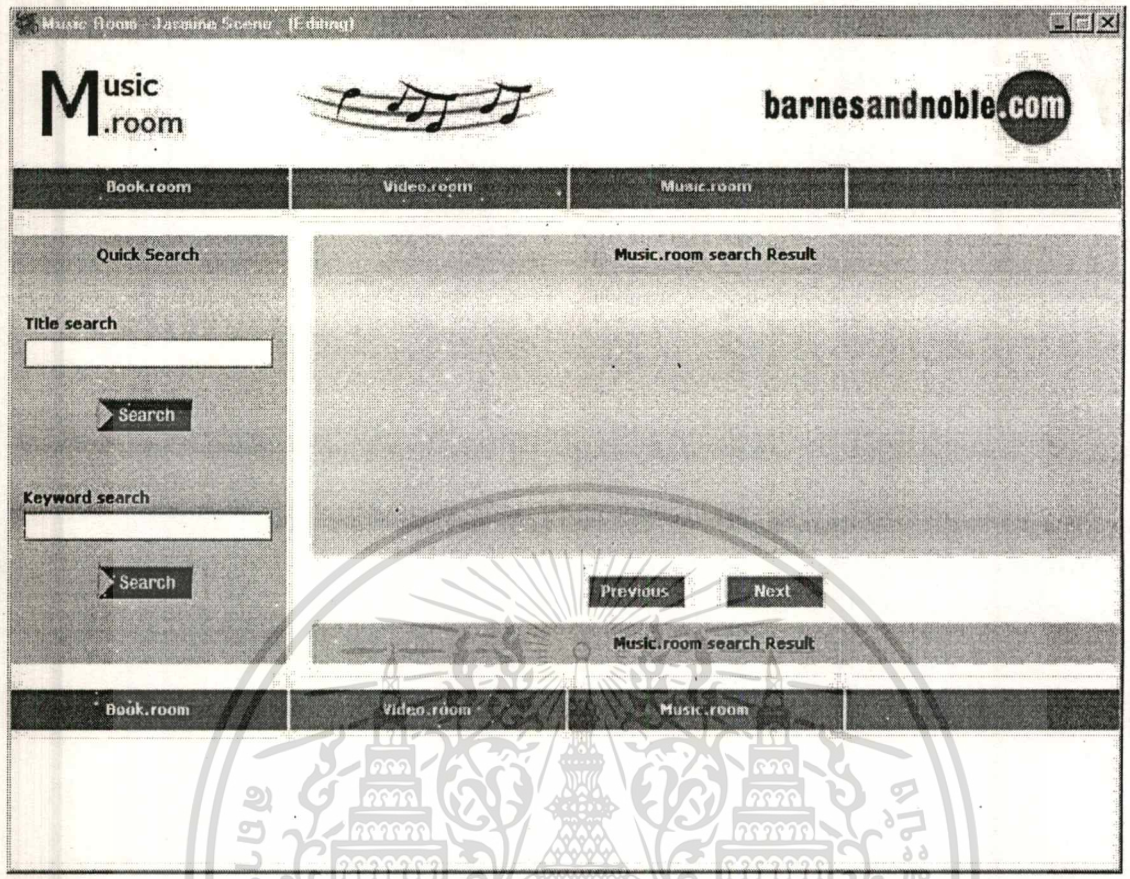
รูปที่ ข.4 แสดงตัวอย่างของผลลัพธ์ที่ได้จากการสืบหาข้อมูลในส่วนของหนังสือ



รูปที่ ข.5 แสดงตัวอย่างของรายละเอียดที่ได้จากการสืบหาข้อมูลในส่วนของหนังสือ



รูปที่ ข.6 แสดงส่วนการทำงานเกี่ยวกับการสืบหาข้อมูลในส่วนของวีดีโอ



รูปที่ ข.7 แสดงส่วนการทำงานเกี่ยวกับการสืบหาข้อมูลในส่วนของเพลง

ประวัติผู้เขียน

ชื่อผู้เขียน	นายจักรกฤษณ์ กลิ่นสมิทธิ์
วันเดือนปีเกิด	11 พฤศจิกายน 2517
สถานที่เกิด	จังหวัดกรุงเทพมหานคร
วุฒิการศึกษาระดับปริญญาตรี	วท.บ.(สถิติประยุกต์)
สถานที่สำเร็จการศึกษา	คณะวิทยาศาสตร์ประยุกต์ สถาบันเทคโนโลยีพระจอมเกล้าพระนครเหนือ
ปีที่สำเร็จการศึกษา	ปีการศึกษา 2539