การนำระบบฐานข้อมูลเชิงวัตถุไปประยุกต์ใช้กับระบบสารสนเทศที่มีปัญหาการเปลี่ยนแปลงตัวระบุชี้

Implementation of Object-Oriented Database for Identifier Changing Problem in Information System

โดย

นายเศรษฐพงศ์ วงษ์อินทร์

รหัส 41067111

รายงานนี้เป็นส่วนหนึ่งของวิชาโครงการพัฒนาระบบงาน

หลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ

ภาคการเรียนที่ 1 ปีการศึกษา 2543

คณะเทคโนโลยีสารสนเทศ

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ชื่อหัวข้อ	การนำระบบฐานข้อมูลเชิงวัตถุไปประยุกต์ใช้กับระบบสารสนเทศที่มีปัญหาการเปลี่ยนแปลงตัวระบุชี้
นักศึกษา	นายเศรษฐพงศ์ วงษ์อินทร์
อาจารย์ที่ปรึกษา	รศ.ดร.ศุภมิตร จิตตะยโศธร
ระดับการศึกษา	วิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ
แขนงวิชา	วิทยาการสารสนเทศ
ปีการศึกษา	2543

บทคัดย่อ

การนำฐานข้อมูลเชิงวัตถุ (Object-Oriented Database) มาประยุกต์ใช้กับระบบสารสนเทศมีวัตถุประสงค์เพื่อ ให้การพัฒนาระบบใช้เวลาสั้นลง ต้นทุนต่ำ การใช้ความสามารถการนำกลับมาใช้ได้อีก ต้นทุนในการบำรุงรักษาระบบต่ำลง เปลี่ยนแปลงได้ง่าย และการเปลี่ยนแปลงไม่มีผลกระทบต่อภายนอกอย่างแน่นอน การนำหลักการของ Object-Oriented มาใช้กับฐานข้อมูลจึงทำให้เกิดข้อดีดังที่กล่าวมา โดยเฉพาะอย่างยิ่งปัญหาเกี่ยวกับการแก้ปัญหาการเปลี่ยนแปลงตัวระบุชี้ (Identifier) ซึ่งระบบฐานข้อมูลเชิงสัมพันธ์ (Relation Database) อาจไม่สามารถรองรับได้หรืออาจทำได้ไม่ดีนักสำหรับปัญหานี้ ในที่นี้จะนำระบบฐานข้อมูลเชิงวัตถุมาประยุกต์ใช้กับการเปลี่ยนแปลงตัวระบุชี้ในระบบสารสนเทศ

ผลที่คาดว่าจะได้รับจากการพัฒนา ได้แก่ระบบงานสารสนเทศที่มีความสามารถในการรองรับปัญหาที่มีการเปลี่ยนแปลงตัวระบุชี้ โดยนำแนวคิดของการพัฒนาระบบเชิงวัตถุมาประยุกต์ใช้ในการพัฒนาและใช้ระบบฐานข้อมูลเชิงวัตถุเพื่อช่วยให้ระบบสามารถทำงานได้อย่างเต็มประสิทธิภาพ โดยในงานวิจัยฉบับนี้จะนำเสนอหลักการเบื้องต้นของ Object-Oriented Database แนวคิด วิธีในการแก้ปัญหาดังกล่าว และวิธีการ พัฒนาระบบงานทะเบียนในเชิงวัตถุ ในการออกแบบ OODB โดยจะนำเสนอตั้งแต่หลักการแนวคิดของ Object-Oriented รวมไปถึงขั้นตอนต่าง ๆ ในการพัฒนาระบบงานทะเบียนในเชิงวัตถุ

Title	Implementation of Object-Oriented Database for Identifier Changing Problem in Information System
Student	Mr. Sethapong Wong-In
Advisor	Assoc.Prof.Dr. Suphamit Chittayasothorn
Level of Study	Master of Science in Information Technology
Major	Information Science
Academic Year	2000

ABSTRACT

The implement Implementation of Object-Oriented Database in Information System 's propose is to reduce period time in development, low-cost in maintainance, reuseability and flexibility especially the changing of identifier problem. In Relational Database System (RDB) can not support this problem.

The result of this research is the system that can support changing identifier problem by using Object-Oriented 's Concept and Object-Oriented Database Management System (OODBMS). This research will present the basic concept of Object-Oriented, the changing identifier problem and how to solve this problem with Object-Oriented Database.

กิตติกรรมประกาศ

ในการจัดทำโครงการพัฒนาระบบงานการนำระบบฐานข้อมูลเชิงวัตถุไปประยุกต์ใช้กับระบบสารสนเทศที่มีปัญหาการเปลี่ยนแปลงตัวระบุชี้ที่จัดทำขึ้นนี้ได้รับความสนับสนุนจากหลายฝ่ายด้วยดีในการให้คำปรึกษา แนวทางในการจัดทำ รวมไปถึงกำลังใจซึ่งทำให้การศึกษาโครงการนี้บรรลุผลตามเป้าหมายที่ได้วางไว้ ผู้จัดทำจึงใคร่ขอขอบพระคุณบุคคลดังนี้

1. บิดา มารดา และน้องสาว ที่สนับสนุนและให้กำลังใจอย่างสม่ำเสมอ
2. รศ.ดร.ศุภมิตร จิตตะยโศธร อาจารย์ที่ปรึกษาโครงการนี้ที่ได้ให้คำปรึกษาและแนวทางในการศึกษาโครงการนี้เป็นอย่างดี
3. บริษัท Warden International CO.,LTD ที่เอื้อเฟื้อให้ข้อมูลของ CACHE ,Software CACHE รวมไปถึงการจัดอบรมการใช้งาน CACHE
4. คุณสุรณรงค์ ดามตะศิลา จาก NECTEC ที่ช่วยให้คำปรึกษาเกี่ยวกับ Object-Oriented Concept และ Object-Orient Design
5. คุณสัณห์ ปลื้มสำราญ, คุณนทพล ทรงเจริญ และคุณบริรักษ์ สุขะตุงคะ เพื่อนสมัยมัธยมปลายโรงเรียนสวนกุหลาบวิทยาลัยนนทบุรี ที่เป็นแรงผลักดันในการทำโครงงานนี้
6. เพื่อน และ พี่ ๆ พนักงานธนาคารไทยพาณิชย์ สำนักงานใหญ่ ที่สนับสนุนการศึกษาโครงการนี้เป็นอย่างดี
7. คุณพิมผกา ประเสริฐศิลป์ ที่ให้คำปรึกษาและเป็นกำลังใจในการศึกษาโครงการมาโดยตลอด

เศรษฐพงศ์ วงษ์อินทร์

ผู้จัดทำ

สารบัญ

หน้า

สารบัญ(ต่อ)

หน้า

สารบัญตาราง

หน้า

ตารางที่

สารบัญภาพ

หน้า

ภาพที่

สารบัญภาพ(ต่อ)

หน้า

ภาพที่

สารบัญภาพ(ต่อ)

หน้า

ภาพที่

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

การพัฒนาระบบในปัจจุบัน การในพัฒนาระบบเชิงวัตถุเป็นแนวคิดที่สามารถแก้ปัญหาที่ซับซ้อนได้ โดยอาศัยหลักแนวคิด Object-Oriented ในการนำหลักการนี้ไปพัฒนาระบบต่าง ๆ จะช่วยให้การพัฒนามีประสิทธิภาพมากขึ้นในด้าน ความรวดเร็ว การลดทรัพยากรในการพัฒนา และสิ่งที่เป็นจุดเด่นมากที่สุดของแนวความคิดเชิงวัตถุได้แก่ ความสามารถในการที่จะเปลี่ยนแปลงตัวระบุชี้ (Identifier Changing) ซึ่งเป็นปัญหาที่สามารถเกิดขึ้นได้ในโลกวัตถุ (Real World) และไม่สามารถใช้การระบบฐานข้อมูลเชิงสัมพันธ์มาแก้ปัญหานี้ได้ หรือ ถึงแม้ว่าจะทำได้แต่ก็อาจจะมีความยุ่งยากในการพัฒนามากกว่ามาก ในรายงานสัมมนาฉบับนี้จะเป็นการแสดงถึงวิธีการในการพัฒนาระบบสารสนเทศที่มีปัญหาในเรื่องการเปลี่ยนแปลงตัวระบุชี้โดยใช้ แนวความคิดเชิงวัตถุมาประยุกต์ ซึ่งจะนำ OODB มาช่วยในการพัฒนา

ในปัจจุบันหลาย ๆ ระบบสารสนเทศยังนิยมที่จะใช้ระบบฐานข้อมูลเชิงสัมพันธ์เนื่องจากมีลักษณะโครงสร้างข้อมูลที่ง่าย มักจะอยู่ในรูปแบบของตาราง(Table) ทำให้ผู้พัฒนาและผู้ใช้งานเข้าใจได้ง่าย แต่ระบบฐานข้อมูลเชิงสัมพันธ์นี้ก็ยังมีข้อจำกัดอยู่มาก ในที่นี้จะกล่าวถึงข้อจำกัดด้านตัวระบุชี้(Identifier) ซึ่งในระบบฐานข้อมูลเชิงสัมพันธ์เรียกว่า Primary key ซึ่งจะไม่อนุญาติให้เปลี่ยนแปลงแก้ไขค่าตัวระบุชี้ได้ ทำให้ไม่สามารถรองรับความต้องการของระบบที่ซับซ้อนบางระบบได้

1.2 วัตถุประสงค์ของงานวิจัย

ในโครงการพัฒนาระบบงานมีวัตถุประสงค์ของการวิจัยดังนี้

1. เพื่อศึกษาทฤษฎีและหลักการทำงานของระบบการจัดการฐานข้อมูลเชิงวัตถุ
2. เพื่อศึกษาวิธีการการพัฒนาระบบสารสนเทศโดยนำระบบฐานข้อมูลเชิงวัตถุมาประยุกต์
3. เพื่อศึกษาปัญหาและแนวทางในการแก้ปัญหาการเปลี่ยนแปลงตัวระบุชี้ที่เกิดขึ้นในระบบสารสนเทศ
4. เพื่อพัฒนาตัวอย่างโปรแกรมประยุกต์ในการจัดการปัญหาการเปลี่ยนแปลงตัวระบุชี้ของระบบงานสารสนเทศบนระบบฐานข้อมูลเชิงวัตถุ

1.3 ขอบเขตของงานวิจัย

ศึกษาแนวความคิดและหลักการในการพัฒนาระบบสารสนเทศเชิงวัตถุในการแก้ไขปัญหาการเปลี่ยนแปลงตัวระบุชี้ และพัฒนาระบบสารสนเทศที่มีปัญหาในเรื่องนี้ได้ โดยมีขั้นตอนฝนการวิจัยดังนี้

1. ศึกษาปัญหาการเปลี่ยนแปลงตัวระบุชี้
2. ศึกษาแนวทางการแก้ปัญหาโดยอาศัยแนวคิดเชิงวัตถุมาประยุกต์
3. ทำการออกแบบระบบสารสนเทศที่มีการแก้ไขเปลี่ยนแปลงตัวระบุชี้ โดยใช้หลัก UML ในการออกแบบ
4. ยกตัวอย่างในรูปแบบกรณีศึกษาของปัญหาการใช้ตัวระบุชี้ในระบบสารสนเทศ
5. ทำการสร้างโปรแกรมตัวอย่างซึ่งสามารถแก้ปัญหาการใช้ตัวระบุชี้ในระบบสารสนเทศ โดยใช้ ระบบฐานข้อมูลเชิงวัตถุในการเก็บข้อมูล

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1. ทราบถึงลักษณะของปัญหาการใช้ตัวระบุชี้ในระบบสารสนเทศ
2. ทราบถึงหลักการในการพัฒนาระบบโดยนำระบบฐานข้อมูลเชิงวัตถุมาประยุกต์ใช้
3. สามารถพัฒนาโปรแกรมในการจัดการปัญหาการเปลี่ยนแปลงตัวระบุชี้ได้

1.5 เนื้อหาในงานวิจัย

ในงานวิจัยนี้ได้แบ่งออกเป็นเนื้อหา 7 บท ดังนี้

บทที่ 1 : บทนำ

บทที่ 2 : แนวคิดของ Object-Oriented Database และ OODBMS

บทที่ 3 : แนวทางการพัฒนาระบบสารสนเทศเชิงวัตถุ

บทที่ 4 : ปัญหาการเปลี่ยนแปลงตัวระบุชี้ (Identifier Problem)

บทที่ 5 : แนะนำระบบฐานข้อมูลเชิงวัตถุ CACHE

บทที่ 6 : กรณีศึกษาการนำ OODB มาแก้ปัญหาการเปลี่ยนแปลงตัวระบุชี้

บทที่ 7 : บทสรุปและแนวทางในการพัฒนาในอนาคต

บทที่ 2

แนวคิดของ Object-Oriented Database และ OODBMS

OODB เริ่มเป็นที่รู้จักตั้งแต่กลางปี คศ. 1980 โดยมีวัตถุประสงค์หลักคือ เพื่อที่จะเพิ่มความสามารถในการพัฒนาระบบให้ดีขึ้นในแง่ของ Model และ Tools ต่าง ๆ โดยนำเอาหลักการของ Object-Oriented มาประยุกต์ใช้ ซึ่งหลักการของ Object-Oriented นั้นเป็นแนวคิดที่ทำให้เราสามารถนำเทคโนโลยีซึ่งง่ายในการสร้าง พัฒนาและดูแลรักษาระบบที่มีความซับซ้อนมาก ๆ ได้ โดยในงานวิจัยนี้จะขอพูดถึงหลักการแนวคิดของ Object-Oriented, OODBMS รวมไปถึงแนวทางและวิธีการในการออกแบบและพัฒนาระบบเชิงวัตถุด้วย

2.1 แนวคิดเชิงวัตถุ(Object-Oriented Concept)

แนวคิดของ Object-Oriented คือ การรวมเอากระบวนการการทำงาน (Process, Method) กับข้อมูล (Data, Information) มารวมกันไว้ในสิ่งที่เรียกว่าวัตถุ(Object) โดยการที่แต่ละ Object จะสื่อสารกันได้ต้องมีข้อตกลงในการสื่อสารว่าจะมีลักษณะการสื่อสารในรูปแบบใด เราเรียกการสื่อสารระหว่าง Object ว่า การส่ง Message Object Oriented Database เป็นแนวความคิดที่ผสมผสานและรวบรวมแนวความคิดของ Object-Oriented และ Database เข้าไว้ด้วยกันเพื่อที่จะทำให้เหมาะสมตรงกับความต้องการที่ใช่แค่เพียง การพัฒนา Database Application เท่านั้น แต่ยังรวมไปถึง การทำงานแบบทั่ว ๆ ไปอีกด้วย โดย มีการให้คำนิยามดังนี้

Object-Oriented Database = Object Orientation + Database capabilities

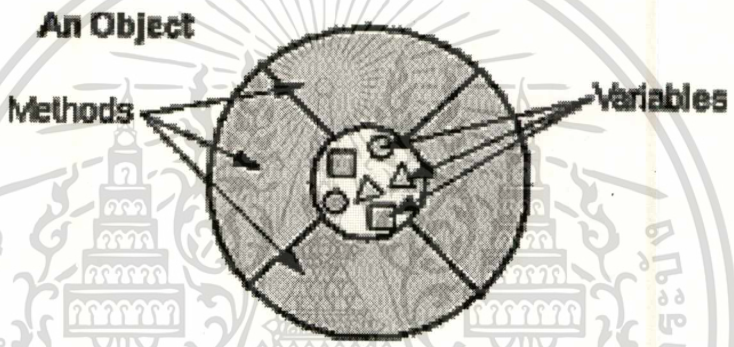
Objected-Oriented = Abstract data type + Inheritance + Objected Identity

Database capabilities = Persistence + Concurrency + Transactions + Recovery + Querying + Versioning + Integrity + Security + Performance

Object-Oriented เป็น Paradigm ที่ใหม่ล่าสุดในการจัดการกับปัญหาโดยมองเป็น Object ซึ่งกำลังเป็นที่นิยมในปัจจุบัน และมีเครื่องมือที่ช่วยอำนวยความสะดวกในการพัฒนามากมาย เช่น Object Model (รวม Data กับ Process เข้าด้วยกัน) ซึ่งมีภาษาที่สามารถใช้ร่วมกับTechnology นี้คือ C++, Smalltalk , Eiffel, Object C, Object Pascal ,Java, Dephi, VB และอื่น ๆ

คำว่า Object (Real World Object) โดยทั่วไปจะหมายถึง วัตถุใดๆจะมีสิ่งสำคัญคือ ลักษณะ, สถานะ และ พฤติกรรม (Attribute, State, Behavior) ส่วน Software Object จะพยายามเลียนแบบ Real World Object โดยแบ่งออกเป็น 2 ส่วน คือ ส่วนที่เป็นสถานะ(Data and variable) และพฤติกรรม (Method, Process, Operation)

Objects An object เป็น Collection ของ Data (Attribute ,Pproperties) และ Function Logic ซึ่ง Data จะบอกถึงคุณสมบัติหรือสถานะของ Object และ Method จะบอกถึงพฤติกรรมต่าง ๆ ของ Object นั้นๆ โดย

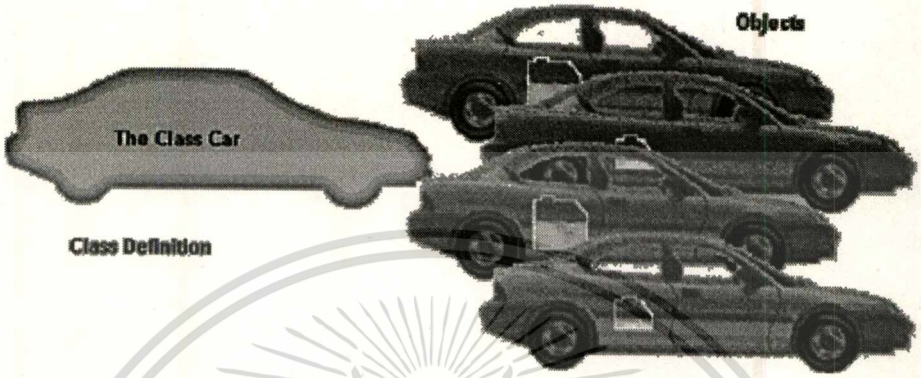


ภาพที่ 2.1 ลักษณะของ Object

- Attribute คือข้อมูลที่เราสนใจเกี่ยวกับ Object นั้น

- Method จะแบ่งเป็น 2 ประเภทคือ interface method (เป็น method ที่ถูกใช้ได้จาก Object อื่น) และ Internal method (เป็น Method ที่จะถูกเรียกใช้ได้เฉพาะภายใน Object ที่เป็นเจ้าของเท่านั้น)

นอกจากนี้แล้วสถานะของ Object(State of Object) ณ ขณะเวลาใดเวลาหนึ่ง ซึ่งจะมีการเปลี่ยนแปลงสถานะเมื่อมีการกระทำจากภายนอกแต่มี Object บางตัวสามารถเปลี่ยนสถานะด้วยตัวเองได้ เรียกว่า Object with life หรือ Actor เช่น นาฬิกา ลูกค้า และยังมีอีกคำหนึ่งที่ควรทราบคือ Class เป็นพิมพ์เขียวของ Object ที่ไม่สามารถนำมาใช้ได้โดยตรง โดยจะมีการบอกถึง Method ที่ใช้ได้โดย Object และมีการแสดง Data Type ที่บอกถึงสถานะของ Object โดยยังไม่ระบุค่าใน Attribute แต่ละตัวซึ่งถ้าเป็น Object จะมีการระบุค่าของ Attribute ที่แน่นอน ถ้า Object เป็น Instance ของ Class ใด Object นั้นจะต้องมี Attribute และ Method เหมือน Class ของมัน



ภาพที่ 2.2 ตัวอย่าง Class กับ Object

การที่จะเป็น Object ได้ต้องมีคุณสมบัติดังนี้

1. Encapsulation/Information hiding

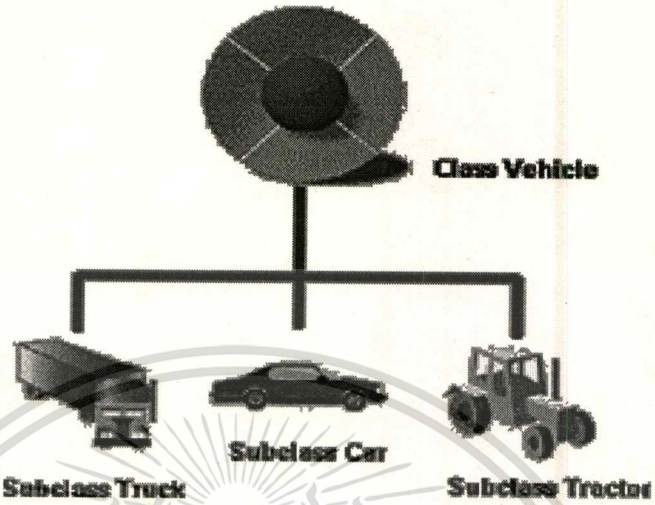
ทฤษฎีนี้ถูกคิดค้นโดย James Rambaugh ซึ่งหมายถึง การแยกลักษณะภายนอก (Interface) ซึ่งสามารถติดต่อกับ Object อื่นๆ ออกจากส่วนที่ Implement ภายในของ Object ซึ่งจะถูกใช้ได้เฉพาะตัว Object นั้นๆ มีข้อดีคือ เพิ่มความสามารถในการปกป้องข้อมูลโดยไม่ให้ Object อื่นเข้ามาทำการเปลี่ยนแปลงข้อมูลก่อนได้รับอนุญาติทั้งนี้ตัว Object ควรรู้และทำการเปลี่ยนค่าด้วยตนเอง ซึ่งเป็นผลให้ทำการดูแลรักษา object ง่ายขึ้น

2. Inheritance

คือหลักการในการที่ Object ทุกๆตัวใน Generalized Collection ได้มีการใช้ข้อมูล (Structure) และพฤติกรรม(Behavior) ร่วมกันซึ่งจะมีความสัมพันธ์แบบ Is-a relationship โดยเรียก Class ที่อยู่เหนือกว่าว่า Parent Class ซึ่งจะถ่ายทอดคุณสมบัติทั้ง Attribute และ Method มายัง Class ที่ต่ำกว่าเรียกว่า Subclass ซึ่งจะมี Attribute และ Method เพิ่มเติมจาก Parent Class มีข้อดีคือ

- เพิ่ม Consistancy เพียงแค่เปลี่ยน Method หรือ Attribute ที่ Super class ซึ่งเป็นผลให้ค่าที่ Subclass เปลี่ยนไปด้วย

- เป็นการส่งเสริมการนำ Object กลับมาใช้ใหม่



ภาพที่ 2.3 แสดงการถ่ายทอดคุณสมบัติ(Inherit)

3. Polymorphism

หมายถึง Having Many Forms โดยการส่ง Message เดียวกันให้กับ Object ที่ต่างกันเป็นผลทำให้ Object แสดงพฤติกรรมที่แตกต่างกัน มีข้อดีคือ เป็นการสนับสนุนการนำโปรแกรมกลับมาใช้ใหม่และสามารถเปลี่ยนแปลง Software ได้ในระหว่างการพัฒนา

ความสัมพันธ์ระหว่าง Objects มี 4 แบบคือ

1. Association

เป็นการอธิบายถึงความสัมพันธ์ระหว่างโครงสร้างโดยทั่วไปกับ Semantic ทั่วไป โดยอ้างถึงความสัมพันธ์ทั้งสองด้านและมีการระบุชื่อความสัมพันธ์อย่างชัดเจน ความสัมพันธ์แบบนี้ประกอบด้วย

- Constraint คือการแสดงข้อมูลที่เกี่ยวกับความสัมพันธ์ระหว่าง Objects

- Link คือ การเชื่อมต่อระหว่าง Objects ที่มีความสัมพันธ์แบบ ถาวร ซึ่งเป็น Instance ของ Association

2. Aggregation

เป็นรูปแบบความสัมพันธ์อย่างหนึ่งของ Association ในการกำหนด Parts ใน Component นั้นโดยมีการแยกระหว่าง Whole กับ Parts ซึ่งจะได้ความสัมพันธ์แบบ Whole - Part หรือ A-Part-of หรืออาจเรียกอีกแบบหนึ่งว่ามีความสัมพันธ์แบบ Has-a Relationship เป็นผลให้ Superclass มีอิทธิพลต่อการดำรงอยู่ของ Subclass คือถ้า Superclass ถูกทำลาย Subclass จะถูกทำลายไปด้วยโดย

อัตโนมัติ นอกจากนี้การพิจารณาถึงการแบ่งระดับของ Aggregation นั้นขึ้นอยู่กับความต้องการของระบบ

3. Generalization

เป็นความสัมพันธ์ในการจัดกลุ่มสิ่งที่เหมือนกันระหว่าง Object หรือ Class ขึ้นมาเป็นอีก Object หนึ่งโดยพิจารณาจากข้อมูล (Structure) หรือ พฤติกรรม (Behavior) ในกลุ่มของ Object นั้นจะมีลักษณะเป็นลักษณะทั่วไปไม่ชี้เฉพาะเจาะจง

4. Depends On

เป็นความสัมพันธ์ที่ชี้ให้เห็นถึง Source หรือ Client ขึ้นอยู่กับ Destination หรือ Supplier เช่นใน Class หนึ่งสามารถมี Method ที่ขึ้นอยู่กับ Class อื่น เวลาทำงานจะมีการอ้างถึง Class นั้นด้วย

Message Passing

คือการที่ Object ติดต่อกันด้วยการส่งข้อความ (Message) ถึงกันและกันซึ่งข้อความจะประกอบด้วยจุดหมายปลายทาง (Destination) ของข้อความนั้นและข้อมูลที่สำคัญ (Argument หรือ Parameter)

Message Passing เปรียบได้กับ Function Call หรือ Procedure call ที่มีใน Structured Programming โดยผ่าน Interface method ของ Object นั้นๆ มีผลทำให้Object ที่เป็นผู้รับข้อความ (Received Object) นั้นกระทำการอย่างใดอย่างหนึ่ง

ภาพที่ 2.4 แสดงการส่ง Message

การพัฒนา Software หรือ Application มักจะประสบปัญหาในเรื่องของการปรับเปลี่ยนความสามารถของ Application อยู่เสมอ จากการที่มีการปรับเปลี่ยนอยู่ตลอดเวลาแม้ในตอนก่อนที่จะส่งมอบก็ยังมีการปรับเปลี่ยนให้เห็นกันอยู่เนือง ๆ ทำให้การพัฒนา Application ต้องประสบกับ

ความล่าช้าเสมอมา การเปลี่ยนแปลงความสามารถเพียงบางอย่างแม้เพียงเล็กน้อย ก็จะทำให้กำหนดเวลาในการส่งมอบต้องยืดออกไป แนวความคิดในการพัฒนาและสร้าง Application Software โดยการทำการประกอบชิ้นส่วนของ Software ขึ้นมาเป็น Software ตัวใหม่จึงได้กำเนิดขึ้น

การพัฒนา Application Software ด้วยการประกอบชิ้นส่วนของ Software ได้พัฒนาจนมาอยู่ในรูปแบบซึ่งเรียกว่า RAD (Rapid Application Development) กล่าวคือ มีการพัฒนาเครื่องมือสำหรับพัฒนา Application ด้วยการนำชิ้นส่วนของ Software มาประกอบกันเพื่อสร้างเป็น Application ตัวใหม่ ประโยชน์ที่ได้จากการพัฒนา Software แบบ RAD ที่เห็นได้อย่างชัดเจนคือ ช่วยประหยัดเวลาและค่าใช้จ่ายในการพัฒนา Software เนื่องจากสามารถเปลี่ยนแปลงหรือปรับปรุงความสามารถของ Application ได้อย่างง่ายดาย โดยใช้วิธีการถอดประกอบชิ้นส่วนต่าง ๆ และเพิ่มเติม Code อีกเล็กน้อยเท่านั้น ซึ่งในระยะเวลาที่ผ่านมาได้พิสูจน์ให้เห็นถึงประโยชน์ของการพัฒนา Software ในรูปแบบนี้แล้วเป็นอย่างดี

ปัจจัยสำคัญที่ทำให้ RAD กลายเป็นเครื่องมือที่มีประโยชน์อย่างมากนั้นก็คือสิ่งที่เรียกว่าชิ้นส่วน Software หรือ Software Component เนื่องจากความสามารถในการนำชิ้นส่วน Software จากผู้ผลิตหลาย ๆ รายมาประกอบใช้งานร่วมกัน เช่น ชอบใจใช้ ปุ่มจากบริษัท A และ ชอบใช้ Listbox จาก บริษัท B ก็สามารถนำมาประกอบใช้งานร่วมกันได้ ซึ่งการที่ เครื่องมือประเภท RAD ยินยอมให้มีการนำ ชิ้นส่วน Software จากผู้ผลิตหลาย ๆ รายเข้ามาใช้ในเครื่องมือของตนได้นั้น เป็นเพราะเครื่องมือ RAD นั้นได้มีการกำหนดมาตรฐานของชิ้นส่วนเหล่านั้นเอาไว้แล้วเป็นอย่างดี เปรียบเสมือนกับการที่ได้มีการกำหนดขนาดของลิ่มและตำแหน่งของลิ่มและรูรับลิ่มในของเล่นเด็ก Lego

RAD ที่ได้รับความนิยมในปัจจุบันได้แก่ Visual Basic , Visual C++ , Oracle Designer 2000 ,Power Builder, Delphi , C++ Builder, IBM VisualAge เครื่องมือต่าง ๆ เหล่านี้เป็นเครื่องมือที่ใช้งานง่าย มีส่วนติดต่อกับผู้ใช้ที่เป็นกราฟฟิค และ สามารถสร้าง Application เล็ก ๆ ออกมาได้ภายในระยะเวลาอันรวดเร็วด้วย การ Click เมาส์เพียงไม่กี่ครั้ง และ เขียน Code สั้น ๆ หรือ อาจจะไม่ต้องเขียนเลย

ในช่วงระยะเวลาที่ผ่านมา ในขณะที่ RAD ได้รับความนิยมอย่างสูง มีผู้ผลิต Software Component ออกมาเป็นจำนวนมาก ชิ้นส่วน Software เหล่านี้มีความสามารถในด้านต่าง ๆ มากมายหลายแบบ แต่ทว่า ความสามารถของชิ้นส่วนเหล่านี้มักเป็นไปในรูปแบบของ GUI เสียเป็นส่วนมาก ในด้านของ Logic หรือ Process ของโปรแกรมแล้วยังขาดผู้ผลิตที่จะสามารถผลิตชิ้นส่วน Software ในด้านนี้ออกมา ทั้งนี้เนื่องจาก Logic ของ Application นั้นมักจะแปรเปลี่ยนไปในแต่ละงานขึ้นกับประเด็นปัญหาที่ต้องการนำ Application นั้นเข้ามารองรับการทำงาน ดังนั้นหน้าที่ในการ

ผลิตชิ้นส่วน Software ที่เป็น Logic จึงเป็นหน้าที่ของนักพัฒนา Software เอง อย่างไรก็ตาม ในช่วงระยะเวลาที่ผ่านมา การพัฒนาชิ้นส่วน Software ที่เป็น Logic มักจะถูกมองข้ามและละเลยไป เนื่องจากต้องสูญเสียเวลาและแรงงานอย่างมหาศาล อีกทั้งสภาพแวดล้อมในการพัฒนาที่ไม่เอื้ออำนวย เนื่องจาก RAD ส่วนมากไม่ได้จัดเตรียมฟังก์ชันในการสร้างชิ้นส่วน Software มาให้ ทำให้ไม่สามารถนำซอร์สโค๊ดที่ได้เคยเขียนไว้มาสร้างเป็นชิ้นส่วน Software ได้ทันที ก่อให้เกิดการทำงานถึงสองขั้นตอน ทำให้การสร้างชิ้นส่วน Software ขึ้นมาเองถูกมองข้ามไป

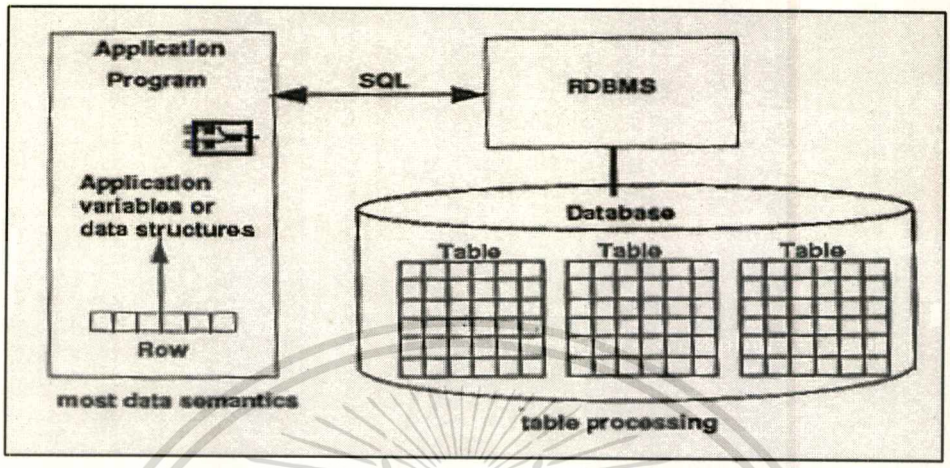
ความพยายามในการสร้างชิ้นส่วน Software สำหรับ Logic หรือ Process ตามอย่างที่สร้างสำหรับ User Interface นั้นมีมาตลอดเวลา ข่าวคราวเกี่ยวกับวิธีการในการสร้าง Logical Software Component นั้น อยู่ในความสนใจของนักพัฒนามาตลอด ทฤษฎีต่าง ๆ ถูกสร้างขึ้นอยู่เสมอ แต่อย่างใดก็ตาม การพัฒนาทางด้านนี้ยังคงเป็นไปอย่างเชื่องช้า อุปสรรคสำคัญคือ เรื่องของงบประมาณและระยะเวลา ซึ่งถ้าหากในการออกแบบ Application ไม่ได้คำนึงถึงเรื่องของการทำให้บางส่วนของ Application กลายไปเป็น ชิ้นส่วน Software แล้ว เวลานำส่วนใดส่วนหนึ่งมาทำเป็นชิ้นส่วน Software จะต้องเสียเวลามาก

ในปัจจุบันภาษาหลักที่ใช้การผลิตชิ้นส่วน Software คือ ภาษาประเภท OOP ทั้งนี้เพราะมีความกลมกลืนเป็นเนื้อเดียวกันตั้งแต่ขั้นตอนการวิเคราะห์ ออกแบบ จนถึงการเขียนโปรแกรม ดังนั้นหากการเขียน Application ได้ใช้วิธีการวิเคราะห์ ออกแบบ และเขียนโปรแกรมด้วยเทคโนโลยีเชิงวัตถุแล้ว จะทำให้การพัฒนาชิ้นส่วน Software ที่เป็นด้าน Logic จะสามารถกระทำได้ง่ายขึ้น ภาษา OOP ที่ได้รับความนิยมเป็นอย่างสูงและถูกเลือกในการสร้างชิ้นส่วน Software คือ ภาษา C++ และ ภาษาใหม่ ๆ อย่าง ภาษา Java

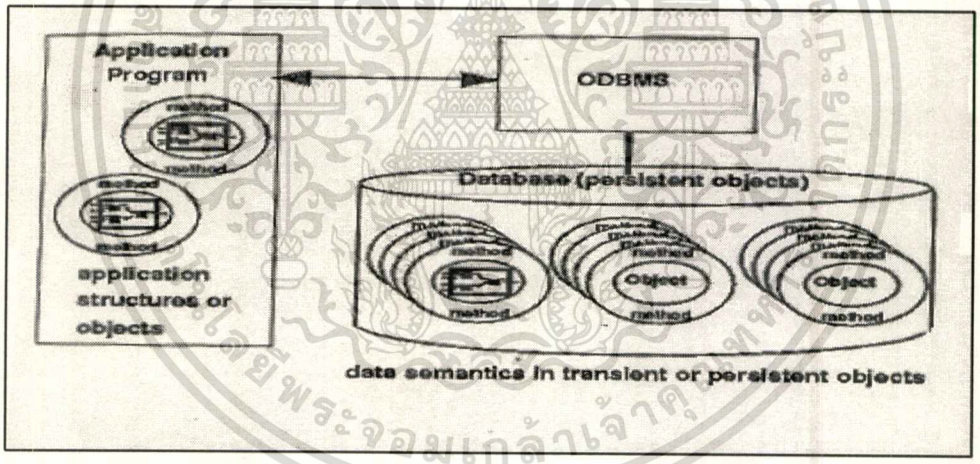
นอกจากภาษาที่จะเลือกมาใช้สร้างชิ้นส่วน Software แล้ว วิธีการในการวิเคราะห์ และออกแบบ ชิ้นส่วน Software ก็นับว่ามีความสำคัญเป็นอย่างยิ่ง เนื่องจากการออกแบบที่ดีจะช่วยประหยัดเวลาในการสร้าง ลดความซ้ำซ้อนในระหว่างการสร้าง และ ก่อให้เกิดความสะดวกและประหยัดในการบำรุงรักษาชิ้นส่วน Software เหล่านั้น

2.2 ระบบการจัดการฐานข้อมูลเชิงวัตถุ(Object-Oriented Database Management System : OODBMS)

ระบบการจัดการฐานข้อมูลเชิงวัตถุ หรือเรียกย่อ ๆ ว่า OOBDMS คือ ระบบที่มีหน้าที่จัดการกับฐานข้อมูลในรูปแบบของ Object ซึ่ง OOBDMS นี้ถูกออกแบบมาเพื่อใช้สำหรับการจัดเก็บข้อมูลในรูปแบบของ Object โดยเฉพาะ ต่างกับ RDMBS ซึ่งถูกออกแบบมาสำหรับการจัดเก็บข้อมูลในลักษณะเป็น Table ดังภาพ



ภาพที่ 2.5 แสดงลักษณะของ RDBMS



ภาพที่ 2.6 แสดงลักษณะของ OODBMS

จากทั้ง 2 รูปแนวคิดโดยทั่วไปของ OODBMS เป็น DBMS ที่ไม่ได้มีไว้ใช้ในการจัดเก็บข้อมูล แต่เป็นการเก็บ Object ซึ่งเป็นการรวมกันของโครงสร้างกับกระบวนการ(Method) ไว้ด้วยกัน โดย OODBMS จะทำการ รวบรวมข้อมูล(Data Integration) , Overall control และ โดยที่ Application จะสามารถเรียกใช้ Object ใดๆ ก็ได้โดยเรียกผ่าน OODBMS ต่างกับ RDBMS ตรงที่ RDBMS จะเก็บ Table และให้ Application เข้ามาเรียกใช้

เนื่องมาจากความเจริญเติบโตทางด้านเทคโนโลยีการพัฒนา Software โดยเฉพาะการพัฒนาในเชิงวัตถุ หรือการพัฒนาโดยใช้ Object-Oriented Programming (OOP) จะเกิดปัญหาในการทำงานร่วมกับ Relational Database เนื่องจาก Relational Database เป็นฐานข้อมูลที่มีรูปแบบ

Relational Model คือ แต่ละ Table มีข้อมูลต่าง ๆ ที่สัมพันธ์กัน ส่วนการพัฒนาโดยใช้ Object-Oriented Programming จะเป็นการมองลักษณะข้อมูลในรูปแบบของ Object Model คือแต่ละ Object มีความสัมพันธ์กันโดยการอ้างถึงกัน ทำให้มีความลำบากในการที่จะเก็บข้อมูลในลักษณะของ Object ลงในฐานข้อมูลที่มีลักษณะเป็น Relation

2.3 ลักษณะพื้นฐานของ OODBMS

ฐานข้อมูลเชิงวัตถุ ตามที่ได้กล่าวไปแล้วในช่วงแรกมีลักษณะพื้นฐานแบ่งเป็นข้อ ๆ ได้ดังนี้

1. Complex Object – OODBMS ต้องมีความสามารถยอมให้ User สามารถที่จะสร้าง Complex Object หรือ Object ที่มีลักษณะซับซ้อนได้ โดยที่ Complex Object จะรวมอยู่หรืออ้างอิงกับ Object อื่น ๆ เพื่อสร้าง Application เกี่ยวกับ Object

2. Extensibility Using Abstract Types or Classes - OODBMS ต้องสนับสนุนรูปแบบของ Type หรือ Class ซึ่งก็คือกลุ่มของ Object ที่มีลักษณะคล้าย ๆ กัน

3. Inheritance หรือการสืบทอด - OODBMS ต้องสนับสนุนหลักการสืบทอด ซึ่งก็คือการที่ Class หนึ่งมีการสืบทอดคุณลักษณะทุกอย่างมาจากอีก Class หนึ่ง และสามารถที่จะมีลักษณะพิเศษอย่างอื่นที่นอกเหนือจากที่สืบทอดมาจากอีก Class ก็ได้

4. Overriding, Overloading - OODBMS ต้องสนับสนุนการทำงานของ Operation Overriding (คือ การที่เราสร้าง Subclass ขึ้นมาใหม่ และมีการสร้าง Method ที่ซ้ำกับ Method ที่เป็นของ Superclass)และ Overloading (คือ การที่เราสร้าง Method ชื่อเหมือนกัน แต่รับ Parameter ต่างกัน ภายใน Class เดียวกัน)

5. Persistence and Secondary Storage Management - การยอมให้มีการคงอยู่ของ Object นั้น ถึงแม้ว่าการประมวลผลนั้นจะเสร็จสิ้นลงแล้วก็ตาม OODBMS ต้องสามารสนับสนุนการคงอยู่ของ Object ทั้งนี้เพื่อความถูกต้องของระบบฐานข้อมูลในการถูกอ้างอิงจาก Object อื่น ๆ ในภายหลัง และOODBMS ยังต้องสามารถสนับสนุน Secondary Storage ในกรณีที่ฐานข้อมูลมีขนาดใหญ่ ซึ่งต้องมีกลไกการต่างๆ เช่น การทำ Index, Clustering, Bufferingเป็นต้น

6. Concurrency and Recovery – OODBMS ต้องสนับสนุนการประมวลผลแบบ Transaction และต้องมีกลไกในการควบคุมการประมวลผลแบบ Concurrency อีกทั้งรวมไปถึงต้องมีกลไกที่สำคัญอีกส่วนหนึ่งคือการ Recovery ด้วย

7. Basic Database Operation การทำงานพื้นฐานของ OODB มีดังนี้

7.1 Storing Object คือการจัดเก็บข้อมูล Object ลงในฐานข้อมูล ซึ่งใน RDBMS ก็สามารถทำได้แต่ต้องมีขั้นตอนในการ Mapping เสียก่อนจึงจะสามารถทำการจัดเก็บข้อมูลแบบ Object ลงในฐานข้อมูลแบบ Relation mได้ ซึ่งเป็นวิธีที่ค่อนข้างยุ่งยาก ซับซ้อนกว่า

7.2 Changing Objects คือ การเปลี่ยนแปลงแก้ไขข้อมูล Object ซึ่งสามารถทำได้โดยการเรียก Object นั้น ๆ จากฐานข้อมูลแล้วทำการแก้ไขได้เลยตามความต้องการ จากนั้นก็ทำการจัดเก็บข้อมูล Object ที่แก้ไขแล้วลงฐานข้อมูลได้ตามเดิม

7.3 Delete Object คือ การลบ Object ออกจากฐานข้อมูล

7.4 Query ในการค้นหาข้อมูลทั้ง RDBMS และ OODBMS จะใช้ Query Language ในการจัดการซึ่ง OODBMS จะได้เปรียบ RDBMS ดังนี้

- ความเร็วในการจัดการข้อมูล ใน RDBMS ข้อมูลมีความสัมพันธ์กันกับข้อมูลในตารางต่าง ๆ ทำให้เกิดความซับซ้อนมากในการค้นหา เมื่อเทียบกับ OODBMS ซึ่งในแต่ละ Objectต่างๆ จะเชื่อมโยงกันโดยใช้การอ้างถึง(Reference) โดยใช้ Link ดังนั้นการที่จะหาขอมูลที่ต้องการก็เพียงแค่ไปที่ Object ที่ต้องการ เท่านั้น

- ในการ Joining เนื่องจาก OODBMS ใช้ Link ในการอ้างถึง Object อื่น ส่วน RDBMS ใช้ Key เป็นตัวเชื่อมความสัมพันธ์ระหว่าง Table จึงทำให้การ Join ของ RDBMS มีความยุ่งยากกว่า

8. Object Identity – Object แต่ละตัวใน OODBMS จะมี Logical Object Identifier(LOID) เป็นเลขที่ระบุ ID ของ Object นั้น ซึ่ง LOID จะต้อง Unique และจะไม่มีการนำกลับมาใช้ใหม่ถึงแม้ว่า Object นั้นจะถูกลบออกไปแล้วก็ตาม

9. Reference Among Object – OODBMS จะมีการอ้างถึง Object อื่น ๆ โดยใช้ Links ซึ่งจะแทนด้วย LOID ของ Object นั่นเอง

10. Version – การเปลี่ยนแปลงของ Object จะเปลี่ยนจากสถานะหนึ่งไปเป็นอีกสถานะหนึ่ง ซึ่ง Version ก็คือกระบวนการในการเก็บผลลัพธ์ที่ได้จาก Object ที่มีการเปลี่ยนแปลงในแต่ละสถานะอย่างรวดเร็ว

11. Ad hoc Query Facility – OODBMS ต้องสนับสนุนการใช้งานเชิงวัตถุที่สอดคล้องกับภาษาที่ใช้ในการ Query ดังเช่นใน RDBMS สนับสนุนการใช้ภาษา SQL ในการทำ Query ซึ่งใน OODB นี้ ใช้ OQL ในการทำ Query โดย OQL เป็นภาษาที่มีลักษณะคล้ายกับภาษา SQL โดยที่ OQL ถูกอธิบายการ Access Object ที่กำหนดด้วยรูปแบบมาตรฐานของ OMG(Object Management Group) การทำ Query ด้วย OQL สามารถอ้างถึงข้อมูลที่ไดกำหนดใน Object Model ได้ โดย OQL มีรูปแบบทั่วไปดังนี้

Query Expression ::=

SELECT constructor(p1,p2,p3,…,pm)

FROM extent list

WHERE condition

Constructor : แทนข้อมูลที่ต้องการ

Extent list : แทน Object หนึ่ง Object หรือมากกว่า(ในกรณีที่มีการรวม Object)

Condition : เป็นส่วนที่ใช้ในการประเมิน Object ถ้า Object เป็นไปตามเงื่อนไข ผลที่ได้จะถูกรวบรวม(กรณีที่มีการรวม Object)และถูกส่งไปยังส่วนของ Constructor ซึ่งในส่วนของ Condition นี้จะอยู่ในรูปแบบของตรรกะ(Boolean) ซึ่งจะสร้างจากการอ้างอิง Attribute และ Method ที่เรียกใช้

2.4 ข้อดีของข้อมูลเชิงวัตถุ

OODB มีลักษณะดังที่ได้กล่าวมาแล้ว โดยมีความสามารถหลัก ๆ ที่จัดเป็นข้อดีได้ดังนี้

1. Extensibility ในส่วนของ Class สามารถที่จะปรับเปลี่ยน แก้ไข เพิ่มเติม ได้ตามต้องการ โดยทีการเปลี่ยนแปลงนั้น ๆ จะถูกจัดการกับ Class ที่มีความสัมพันธ์กับ Class ที่เราแก้ไขโดยอัตโนมัติ นั่นคือเราสามารถที่จะแก้ไขเปลี่ยนแปลง Class ต่าง ๆ ได้โดยที่ไม่มีการส่งผลกระทบถึงส่วนอื่น ๆ ภายในระบบเลย

2. Information Hiding การซ่อนข้อมูล Object จะอนุญาตให้ข้อมูลและ Operation ถูกกำหนดอยู่ภายในโครงสร้างข้อมูลเดี่ยว(Single Model) นั่นคือ ข้อมูลจะถูกกำหนดเฉพาะที่และจะซ่อนไม่ให้คนภายนอกเห็น โดยคนนอกจะสามารถเข้ามาใช้ข้อมูลได้ก็ต่อเมื่อส่งผ่านฟังก์ชั่นด้วยวิธี Message Passing เท่านั้น

3. Flexibility of Type Definition คือ ความสามารถที่อนุญาตให้ผู้ใช้กำหนด Class และ Operation ใหม่ กล่าวคือ ให้ความยืดหยุ่นและการควบคุมกับผู้ใช้ ซึ่งเป็นประโยชน์กับการประยุกต์ใช้ฐานข้อมูลใหม่ ๆ ในการปรับเปลี่ยนข้อมูล

4. Code Reusability เนื่องจาก Object มีคุณสมบัติการสืบทอด (Inheritance) จาก Super Class ไปยัง Sub Class เราจึงไม่จำเป็นต้องทำการกำหนดวิธีการ และ Operation ต่างๆ ใหม่ ทำให้ช่วยลดความทำงานที่ซ้ำซ้อนได้

บทที่ 3
แนวทางการพัฒนาระบบสารสนเทศเชิงวัตถุ

การพัฒนาซอฟต์แวร์เชิงวัตถุ (Object Oriented Software) เป็นแนวคิดในการพัฒนาซอฟต์แวร์รูปแบบหนึ่ง ที่ผนึกเอาข้อมูลและฟังก์ชันการทำงานเข้าด้วยกัน เรียกว่า วัตถุ (Object) และสามารถกำหนดอินเตอร์เฟส (Interface) ระหว่างวัตถุต่างๆ โดยถ้าวัตถุหนึ่งต้องการจะติดต่อกับอีกวัตถุหนึ่ง ต้องติดต่อผ่าน Interface ที่กำหนดไว้เท่านั้น จากหลักการนี้จะพบว่าการเปลี่ยนแปลงแก้ไขฟังก์ชันการทำงาน และข้อมูลใดๆ ในวัตถุหนึ่ง จะมีผลกระทบต่อวัตถุอื่นน้อยมาก ซึ่งจะทำให้การแก้ไขโปรแกรมทำได้สะดวกและรวดเร็วยิ่งขึ้น เป็นการลดต้นทุนในการบำรุงรักษาโปรแกรม นอกจากนี้ยังมีความสามารถในการถ่ายทอดคุณสมบัติ (Inheritance) จากวัตถุหนึ่งไปอีกวัตถุหนึ่งได้ ทำให้สามารถนำซอฟต์แวร์ บางส่วนที่มีอยู่เดิมกลับมาใช้ใหม่ได้ โดยผู้พัฒนาไม่จำเป็นต้อง ทราบว่าซอฟต์แวร์บางส่วนที่นำกลับมา ใช้นั้นเขียนขึ้นมาอย่างไร (How does it work?) เพียงแต่ทราบว่าซอฟต์แวร์ส่วนนั้นทำอะไร (What does it work?) และทำการเพิ่มคุณสมบัติอื่นๆ ที่ผู้พัฒนา ต้องการเข้าไป ทำให้การพัฒนาซอฟต์แวร์ใหม่ๆ ทำได้รวดเร็วขึ้น

3.1 ภาพโดยรวมของการพัฒนาระบบเชิงวัตถุ

เนื่องจากในการพัฒนาซอฟต์แวร์ในระบบเก่าเกิดปัญหาขึ้นในหลายขั้นตอนของการพัฒนา ซึ่งทำ ให้ผู้พัฒนาต้องเสียค่าใช้จ่ายในการพัฒนาและค่าบำรุงระบบค่อนข้างสูงมากเมื่อเทียบกับผลตอบ แทนที่ได้เนื่องจากสามารถใช้ได้เฉพาะงานนั้นๆเท่านั้น (ไม่สามารถนำกลับมาใช้ใหม่ได้อีก) นอกจากนี้เมื่อ ผู้ใช้งานระบบ(User) ต้องการเปลี่ยนความต้องการเกี่ยวกับการทำงานของระบบที่ทำเสร็จแล้วก็ยังเป็น การยุ่งยากเพราะต้องไปเริ่มต้นใหม่อีกจึงมีผู้เชี่ยวชาญหลายท่านได้ทำการ คิดค้นหาแนวทางที่จะแก้ปัญหาเหล่านี้ และพบว่า Object Technology เป็นแนวทางหนึ่งในการพัฒนาซอฟต์แวร์ที่มีประสิทธิภาพและในขณะนี้ได้เป็นที่นิยมในหมู่ผู้พัฒนาระบบในหลายๆองค์กร

3.2 การวิเคราะห์และออกแบบเชิงวัตถุ (Object-Oriented Analysis and Design)

1. ฟังก์ชันนอลโมเดล (Functional Model)

ฟังก์ชันนอลโมเดล เป็นโมเดลที่ใช้ในการแสดงความต้องการของระบบทั้งหมด ช่วยในการอธิบายรายละเอียดหลักๆ ภายในวัตถุ แสดงให้เห็นการไหลของข้อมูลในแต่ละการทำงาน โดย

จะสนใจเพียงแค่ว่ามีงานอะไรบ้างที่ต้องทำ จะยังไม่สนใจว่างานนั้นๆ ทำอย่างไร เช่น ในการพิมพ์รายงานสรุปประจำเดือน จะต้องรู้ว่างานนี้ ใครเป็นผู้ที่รับผิดชอบรายงาน รูปร่างหน้าตาของรายงานเป็นอย่างไร มีข้อมูลอะไรบ้างที่สนใจ และเมื่อออกรายงานแล้ว ใครเป็นผู้ที่รับรายงานนั้นเป็นต้น เครื่องมือที่ใช้ในการแสดงความต้องการของระบบคือ ยูสเคสไดอะแกรม (Use Case Diagram) เป็นไดอะแกรมที่ใช้ในการแสดงความต้องการของระบบทั้งหมดในลักษณะที่ผู้ใช้งานสามารถเข้าใจได้ง่าย โดยจะถูกนำไปใช้ต่อไปใน Phase ต่างๆ ของการวิเคราะห์และออกแบบระบบ

2. Object โมเดล (Object Model)

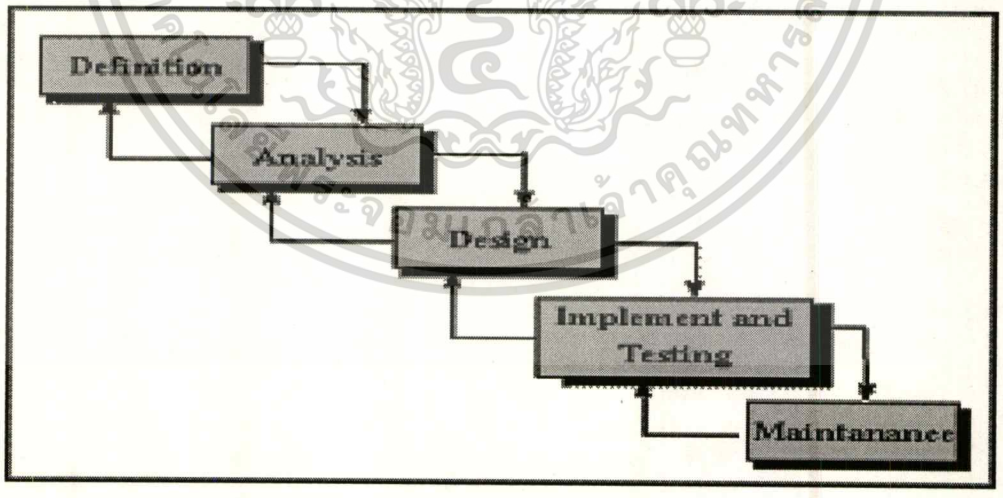
Object โมเดล เป็นโมเดลที่ใช้ในการแสดงโครงสร้างของระบบ โดยจะแสดงในรูปของ Class (Class) ต่างๆ พิจารณาจากความต้องการของระบบ ที่แสดงอยู่ในฟังก์ชันนอลโมเดล เครื่องมือที่ใช้ในการแสดงโครงสร้างของระบบจะมี Component Diagram และClass Diagram ซึ่ง Component Diagram จะแสดงให้เห็นถึงความสัมพันธ์ระหว่างคอมโพเนนท์ต่างๆ ภายในระบบ Class Diagram แสดงให้เห็นถึงคุณลักษณะ (Attribute) คือ ข้อมูล (Data) หรือ ตัวแปร (Varilable), หน้าที่การทำงาน (Operation) คือ Method ภายใน Class และความสัมพันธ์ระหว่าง Class ต่างๆ ภายในระบบ

3. ไดนามิคโมเดล (Dynamic Model)

ไดนามิคโมเดล เป็นโมเดลที่ใช้ในการแสดงถึงการทำงานระหว่าง Object ต่างๆ ตามการส่งข้อความ (Message) หรือเมื่อเหตุการณ์ (Event) ต่างๆ ได้เกิดขึ้น Object ในที่นี้หมายถึง Instance ที่สร้างขึ้นจาก Class ที่ได้ออกแบบไว้ใน Object โมเดล โดยที่แต่ละ Object มีคุณสมบัติ และพฤติกรรมเช่นเดียวกับ Class ต้นแบบ ในการทำงานของระบบจะประกอบขึ้นจากการส่งข้อความไปมาระหว่าง Object เหล่านั้น เมื่อมีการทำงานไปเรื่อยๆ แล้ว Object อาจจะมีการเปลี่ยนสถานะ(State) ไปตามเหตุการณ์ที่เกิดขึ้นได้ เพื่อให้เป็นตามที่กำหนดไว้ในฟังก์ชันนอลโมเดล เครื่องมือที่ใช้คือ Sequence Diagram และ State Diagramโดย Sequence Diagram แสดงให้เห็นถึงลำดับขั้นตอนการทำงาน เมื่อมีการเกิดเหตุการณ์หนึ่งๆ ขึ้นแล้ว Object ต่างๆ มีการทำงานต่อไปอย่างไร วัตถุประสงค์หลักของ Sequence Diagram คือเพื่อให้ผู้เขียนโปรแกรมสามารถพัฒนาระบบได้ตามที่ออกแบบไว้ ดังนั้นจึงนับได้ว่า Sequence Diagram เป็นเครื่องมือที่ผู้เขียนโปรแกรมสามารถนำไปศึกษาและพัฒนาโปรแกรม หรือขยายขีดความสามารถของโปรแกรมต่อไป และ State Diagram เป็นไดอะแกรมที่แสดงให้เห็นถึงสถานะทั้งหมดที่เป็นไปได้ และเหตุการณ์ที่ทำให้เกิดการเปลี่ยนสถานะของ Object แต่ละตัว

3.3 การพัฒนา Software โดยใช้หลัก UML

ในการพัฒนา Softwareเป็นเรื่องที่ค่อนข้างซับซ้อนและมีรายละเอียดค่อนข้างมาก จึงได้มีการแบ่งการพัฒนา Softwareออกเป็น Phase ต่างๆ เรียกว่า Software development life cycle model ซึ่งการแบ่งขั้นตอนการพัฒนาออกเป็น Phase ต่าง ๆ ทำให้การปฏิบัติและติดตามผลของการพัฒนาง่ายขึ้น ซึ่งได้มีการกำหนดมาตรฐาน Software Development Life Cycle (SDLC) ไว้หลายโมเดล เช่น Waterfall model, Rapid prototyping model, Spiral model, Concurent Development Model และ Formal Methods Model เป็นต้น ในที่นี้จะนำ Waterfall Model มาเป็นหลักในการประยุกต์ใช้กับการวิเคราะห์และออกแบบเชิงวัตถุ เพราะ Waterfall Model นั้นเป็นโมเดลที่นิยมและเข้าใจง่าย ซึ่ง Waterfall Model จะแบ่ง Phase การทำงานออกเป็น 5 Phase คือ Phase ของการให้คำนิยามต่าง ๆ ของ Software (Definition), Phase ของการวิเคราะห์ (Analysis), Phase ของการออกแบบ (Design), Phase ของการปฏิบัติและทดสอบ (Implement and testing) และ Phase ของการบำรุงรักษา (Maintenance) ดังรูปที่ 1 สัญลักษณ์ที่ใช้ช่วยในการวิเคราะห์และออกแบบระบบเชิงวัตถุที่ใช้ในการอธิบายนี้จะยึดตามหลักของ UML Notation(Unified Modeling Language Notation) ซึ่งเป็นมาตรฐานในการพัฒนาระบบด้วย Object Oriented Technology จัดทำขึ้นโดย Object Management Group (OMG) โดยแบ่งการออกแบบเป็นขั้นดังนี้



ภาพที่ 3.1 Waterfall model

Phase 1 : การให้คำนิยามของ Software (Definition)

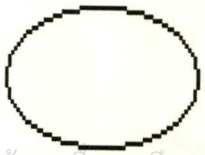
Phase นี้เป็น Phase ที่สำคัญที่สุด เนื่องจากเป็นการศึกษาถึงความต้องการต่างๆ ของระบบ ปัญหาหลักของระบบคืออะไร ขอบเขตของระบบงานที่สนใจมีแค่ไหน ความต้องการของผู้ใช้มีอะไรเพิ่มเติมบ้าง และมีคำศัพท์อะไรบ้างจะต้องมีการให้คำนิยามของศัพท์เหล่านั้นเพื่อความเข้าใจที่ตรงกัน ถ้าใน Phase นี้วิเคราะห์ความต้องการของระบบไม่ชัดเจน ก็จะมีผลถึง Phase ต่อไปของการดำเนินงานด้วย ผลลัพธ์ที่ได้จาก Phase นี้เป็นฟังก์ชันนอลโมเดล คือ Use Case Diagram และพจนานุกรมของข้อมูล (Data Dictionary)

Use Case Diagram เป็นแนวคิดของไอวาร์ จาคอบสัน (Ivar Jacobson) ซึ่ง OMG ได้รวมไว้ในมาตรฐานของ UML ด้วย และได้มีวิธีการอื่นๆ นำแนวคิดนี้ไปใช้อย่างกว้างขวาง Use Case Diagram เป็นสิ่งที่ใช้ในการแสดงความต้องการของระบบทั้งหมดในลักษณะที่ผู้ใช้งานสามารถเข้าใจได้ง่าย ซึ่งเน้นในมุมมองของผู้ใช้ระบบกับการติดต่อระบบ Use Case Diagram จะถูกนำไปใช้ต่อไปในการออกแบบระบบใน Phase ต่างๆ ของการดำเนินงาน สัญลักษณ์หลักที่ใช้ใน Use Case Diagram มีดังนี้

- Actor คือสิ่งที่อยู่ภายนอกระบบที่จะพัฒนาขึ้นทั้งหมด ที่ต้องการแลกเปลี่ยนหรือส่งข้อมูลให้กับระบบ โดยที่ Actor อาจจะเป็นคน ระบบ หรือโปรแกรมอื่นๆ ก็ได้ โดยมากจะเป็นผู้เริ่มทำงานกับ Use Case เช่นผู้ใช้ระบบ เป็นต้น

ภาพที่ 3.2 สัญลักษณ์ของ Actor

- Use Case จะเป็นตัวแทนงานที่เกิดขึ้นในขั้นตอนต่างๆ โดยจะใช้สัญลักษณ์เป็นรูปวงรีหรือวงกลม ถ้า Use Case ใดมีกรอบสี่เหลี่ยมสีเทาล้อมรอบแสดงว่ามี Diagram ย่อยที่ใช้อธิบายรายละเอียดของ Use Case ต่อไป



ภาพที่ 3.3 สัญลักษณ์ของ Use Case

- ระบบ (System) เป็นตัวแทนของระบบ ซึ่งใช้สัญลักษณ์เป็นรูปสี่เหลี่ยม ที่ถูกกระทำโดย Actor ภายในระบบประกอบไปด้วย Use Case ต่างๆ ในที่นี้คือ ระบบสรรหาทรัพยากรบุคคล (Recruitment System)

ภาพที่ 3.4 สัญลักษณ์ของ System

- Communication เป็นการแสดงความสัมพันธ์หรือการติดต่อสื่อสารกัน (การรับและให้ข้อมูลข่าวสารแก่กันและกัน) ระหว่าง Actor และ Use Case ซึ่งอาจจะเป็นการสื่อสารแบบทางเดียวหรือ 2 ทางก็ได้ แสดงด้วยสัญลักษณ์เส้นที่มีหัวลูกศรสีดำ

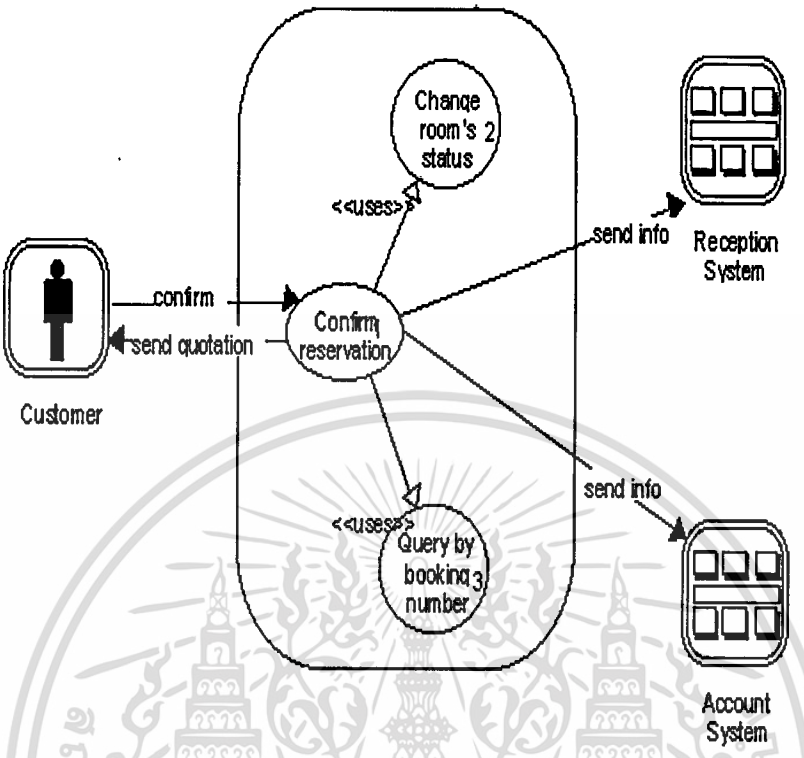
ภาพที่ 3.5 สัญลักษณ์ของ Communication

- Relationship เป็นการแสดงความสัมพันธ์ระหว่าง Use Case กับ Use Case โดยใช้เส้นที่มีหัวลูกศรสีขาว ความสัมพันธ์ระหว่าง Use Caseมีได้ 2 แบบ คือ Extends และ Uses

ภาพที่ 3.6 สัญลักษณ์ของ Relationship

- Extends เป็นการเพิ่มการทำงานให้กับ Use Case โดยการเรียกใช้จากอีก Use Case หนึ่ง ลูกศรจะออกจาก Use Case ที่ถูกเรียกใช้งานไปยัง Use Case ที่ต้องการเพิ่มเติมการทำงาน และจะมีข้อความ <<extends>> ระบุอยู่ข้างๆ เส้น

- Uses เป็นการแสดงให้เห็นถึงการ Inherit หน้าที่การทำงานจาก Use Case หนึ่ง ไปอีก Use Case หนึ่ง ปลายลูกศรจะอยู่ที่ Use Case หลักที่จะถูกถ่ายทอดความสามารถออกไป โดยจะมีข้อความ <<uses>> ปรากฏอยู่ข้างๆ เส้น



ภาพที่ 3.7 ตัวอย่าง Use Case Diagram การยกเลิกจองห้องพักของโรงแรม Comfort Hotel

Description : ระบบการจองห้องพักของโรงแรม Comfort Hotel อนุญาติให้มีการยกเลิกการจองห้องพักที่มีการจองไว้ได้ โดยผ่านมาทาง Use Case diagram นี้ซึ่งอนุญาติให้ระบบสามารถทำงานยกเลิกการจองห้องพักได้ 2 แบบคือ

1. การยกเลิกการจองห้องพักที่มีการยืนยันแล้วเมื่อลูกค้าต้องการยกเลิกการจองห้องพักที่มีการยืนยันแล้วจะให้ หมายเลขการจองห้องพักแบบยืนยัน ระบบจะทำการยกเลิกการจองห้องพักให้และส่งรายละเอียดการยกเลิกนี้ไปให้ กับระบบบัญชี (Account System) และระบบต้อนรับ (Reception System)

2. การยกเลิกการจองโดยอัตโนมัติ ทุกๆวันพนักงานจะทำการยกเลิกการสำรองห้องพักที่เกินระยะเวลาที่กำหนดการยกเลิกทั้งสองแบบจะมีการเปลี่ยนสถานะของห้องพักกลับไปเป็น "ว่าง" เพื่ออนุญาติให้ห้องพักเหล่านี้สามารถถูกสำรองได้ต่อไป จากรูปอธิบายได้ว่า Actor ของระบบมี 3 กลุ่ม คือ เป็น ลูกค้า (customer) ซึ่งสามารถที่จะทำการยืนยันการจองห้องพัก (Use Case : confirm reservation) ซึ่งจาก Diagram แสดงให้เห็นว่า Use Case : Confirm Reservation มีความสัมพันธ์กับ Use Case : Change room status กับ Use Case : Query by booking number แบบ Uses ซึ่งเหมือนกับว่า Use Case : confirm reservation ได้รับการถ่ายทอดคุณสมบัติมาจากทั้ง Use Case : Make

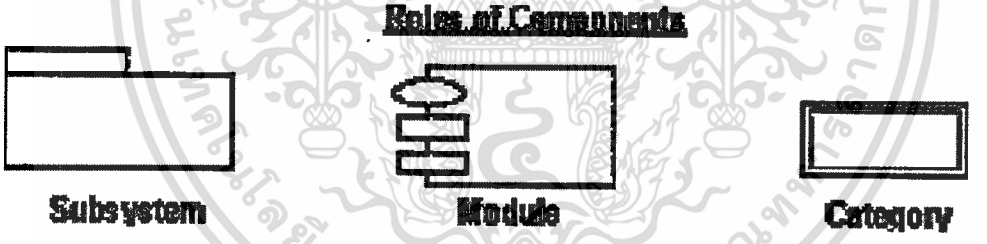
reservation และ Use Case : Query by booking number ซึ่งต่อไปในอนาคตอาจจะมีการทำงานแบบอื่นๆ ที่มีบางส่วนของการทำงานซ้ำกับ

3. Use Case นี้ก็สามารถที่จะถ่ายทอดคุณสมบัติของ Use Case เหล่านี้ได้ ก็จะเป็นการนำกลับมาใช้ใหม่ในขั้นตอนของการวิเคราะห์ระบบ

Phase 2 : การวิเคราะห์ (Analysis)

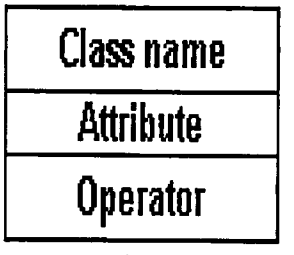
ในเฟสของการวิเคราะห์จะเป็นหลักสำคัญของในการที่จะศึกษาและออกแบบถึงโครงสร้างของระบบงานตามขอบเขตของปัญหาที่สนใจมีอะไร บ้างตาม Use Case Diagram ที่ได้จาก Phase ที่ 1 ใน Phase นี้จะได้ผลลัพธ์เป็น Object โมเดลผู้วิเคราะห์จะทำการรวบรวม Class ที่สัมพันธ์กันขึ้นมาเป็น หมวดหมู่ เรียกว่า Component เพื่อเป็นการลดความซ้ำซ้อนของระบบ ส่วนComponent Diagram แสดงถึงความสัมพันธ์ระหว่าง Component ต่างๆ

Component จะจำแนกได้ 3 แบบ คือ SUBSYSTEM, MODULE และ CATEGORY มีสัญลักษณ์ดังนี้



ภาพที่ 3.8 สัญลักษณ์แสดงประเภทของ Component

Class ใช้สัญลักษณ์เป็นรูปสี่เหลี่ยม จะอธิบายถึงกลุ่มของ Objects ต่างๆ ที่มีคุณลักษณะ (Attribute), หน้าที่การทำงาน (Operation) และความสัมพันธ์(Relationship) ขั้นพื้นฐานซึ่งในแต่ละ Class ประกอบไปด้วยรายละเอียดพื้นฐาน 3 ส่วน ดังนี้



ภาพที่ 3.9 สัญลักษณ์แสดงรายละเอียดของ Class

- Class Name : แสดงชื่อของ Class ที่กำหนดในระบบ

- Attribute : เป็นการกำหนดคุณลักษณะทั้งหมดที่มีภายใน Class บอกถึงรายละเอียดของข้อมูล (Attribute) ภายใน Class เดียวกันจะไม่ซ้ำกัน แต่ชื่อข้อมูลใน Class อาจจะไปซ้ำกับชื่อข้อมูลใน Class อื่นได้

- Operation : เป็นส่วนที่ใช้อธิบายใน Class นั้นมี Method อะไรบ้าง มีการรับค่าชุดของตัวแปร (Argument) อะไรบ้าง หรือมีการส่งค่าออกไปหรือไม่

Attribute และ Operation จะมีการแสดงสิทธิการเข้าถึงข้อมูล และ Operation ของ Class มีรายละเอียดดังนี้

สัญลักษณ์	ความหมาย	คำอธิบาย
-	Private	สามารถเข้าถึงได้เฉพาะตัวเอง, subclasses และ friends
+	Public	สามารถเข้าถึงได้ทุก Class
#	Protected	สามารถเข้าถึงได้เฉพาะตัวเอง และ subclasses
?	Implementation	Scope ถูกตัดสินใจตอน Implement Program แต่ในตัวอย่างที่แสดงในเอกสารนี้หมายถึง package ในภาษา Java คือ สามารถเข้าถึงได้เฉพาะตัวเอง และ classes ที่อยู่ใน package เดียวกัน

ตาราง 3.1 แสดงระดับการเข้าถึงของความสัมพันธ์ระหว่าง Class (Relationship)

UML ได้เตรียมความสัมพันธ์ระหว่าง Class ไว้ จะมีความสัมพันธ์อยู่ 4 แบบ มีสัญลักษณ์ดังนี้

- Association : จะแสดงความสัมพันธ์ระหว่าง Class จะมีได้ทั้งทางเดียว และ 2 ทาง มีสัญลักษณ์ดังนี้

ความสัมพันธ์แบบทางเดียว : 

ความสัมพันธ์แบบสองทาง :

ภาพที่ 3.10 สัญลักษณ์แสดงความสัมพันธ์แบบ Association

- Aggregation : เป็นรูปแบบพิเศษของ association คือเป็นความสัมพันธ์ระหว่าง Whole และ parts ของมัน คือ Whole จะประกอบไปด้วย parts ต่างๆ ของมัน ดังนั้นการคงอยู่ของ Parts จะต้องขึ้นกับ Whole หรือจะเรียกความสัมพันธ์แบบนี้ว่า **Whole-part** ก็ได้ มีสัญลักษณ์ดังนี้

ภาพที่ 3.11 สัญลักษณ์แสดงความสัมพันธ์แบบ by Value และ by Reference

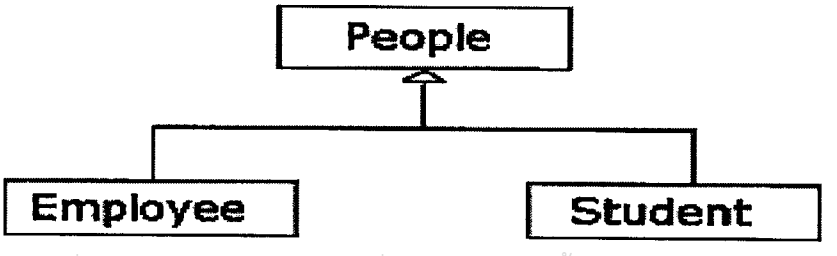
ภาพที่ 3.12 สัญลักษณ์ของความสัมพันธ์แบบ Aggregation

จากรูปอธิบายได้ว่า Whole ประกอบด้วย Part 3 ชิ้น และมีความสัมพันธ์แบบ by Reference

- Depends on : เป็นรูปแบบความสัมพันธ์แบบหนึ่งที่ใช้แสดงความสัมพันธ์กันระหว่าง Class 2 Class ในแง่ที่ Class หนึ่งเรียกใช้บริการของอีก Class หนึ่ง กล่าวคือ Class ของผู้ขอบริการขึ้นอยู่กับบริการของ Class ของผู้ให้บริการ แต่ไม่มีการขึ้นต่อกันภายในโครงสร้างของ Class มีสัญลักษณ์ดังนี้

ภาพที่ 3.13 สัญลักษณ์แสดงความสัมพันธ์แบบ Depend on

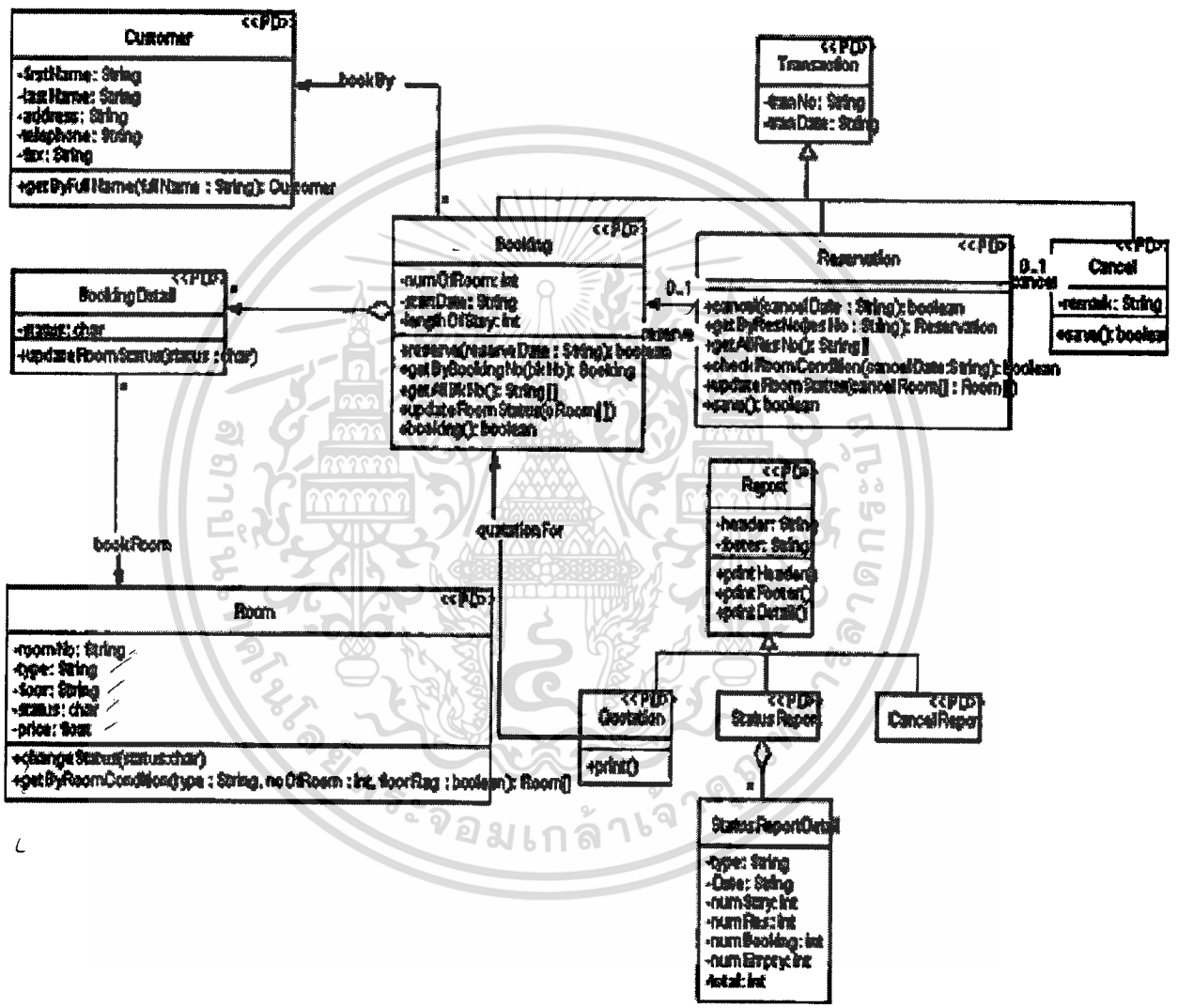
- Generalization : ความสัมพันธ์รูปแบบนี้ใช้แสดงความสัมพันธ์ระหว่าง Class กับ Class ในแง่ที่ Class หนึ่งถ่ายทอดคุณสมบัติและโครงสร้างจากอีก Class หนึ่ง โดยเรียก Class ที่ถูกถ่ายทอดว่า Super Class และเรียก Class ที่ทำการถ่ายทอดว่า Sub Class มีสัญลักษณ์ดังนี้



ภาพที่ 3.14 สัญลักษณ์แสดงความสัมพันธ์แบบ Generalization

จากรูปอธิบายได้ว่า People เป็น Super Class ของ Employee และ Student หรือ Employee และ Student เป็น Sub Class ของ People

<u>ตัวอย่าง</u> Class Diagram ของปัญหาของโรงแรม Comfort Hotel



ภาพที่ 3.15 Class Diagram : Reservatation Hotel

จากรูปจะเป็น Class ทั้งหมดที่ได้จากการวิเคราะห์ จะแสดงถึงความสัมพันธ์ระหว่าง Class เช่น Booking มีความสัมพันธ์กับ BookingDetail แบบ Aggregation ซึ่ง Hotel เป็น Whole ส่วน Room เป็น Part เป็นต้น ซึ่ง Class ที่ได้ในช่วงนี้จะเป็น Class ที่เกี่ยวข้องกับปัญหาหลักของระบบ

Phase 3 : การออกแบบ (Design)

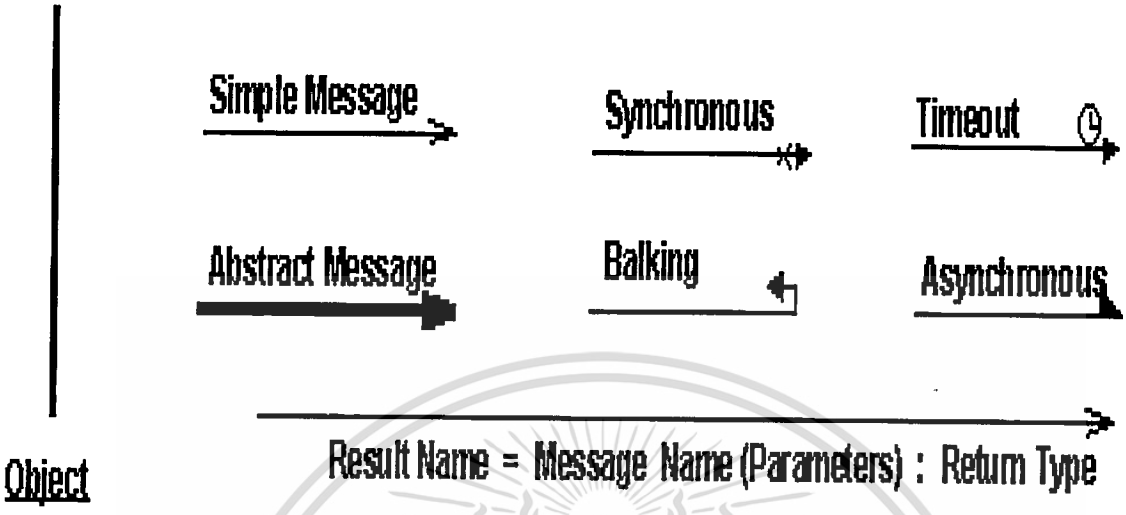
การออกแบบระบบจะเริ่มเมื่อข้อมูลในช่วงของการวิเคราะห์เพียงพอที่จะเริ่มทำการปฏิบัติกับระบบแล้ว ในการออกแบบควรจะพิจารณาถึง

- **Hardware** ว่า ระบบเป็น Centralized หรือ Distributed ถ้าเป็นระบบแบบ Distributed มีการจัดการกับระบบความปลอดภัยอย่างไร จะใช้อะไรเป็นตัวติดต่อสื่อสาร ต้องการ Object เพิ่มเติมเพื่อช่วยอำนวยความสะดวกในการติดต่อสื่อสารหรือไม่ เป็นต้น

- **Software** ว่ามี Class พิเศษอะไรบ้างเพื่อสร้างหน้าต่างส่วนที่ติดต่อกับผู้ใช้(User interface) ต้องการ Class เพิ่มเติมเพื่อช่วยอำนวยความสะดวกในเรื่องของความสัมพันธ์ระหว่าง Class หรือไม่ ถ้าในระบบต้องการใช้ระบบฐานข้อมูล มี Class พิเศษที่ต้องใช้ในการติดต่อระหว่าง DBMS หรือไม่ ต้องมีการลดความซ้ำซ้อนของข้อมูลหรือไม่ ต้องทำการ Normalization หรือไม่มีการใช้ Class ของบุคคลภายนอกหรือไม่ การติดต่อระหว่าง Object เป็นแบบ พร้อมกัน (Synchronously) หรือ เกิดขึ้นไม่พร้อมกัน(Asynchronously) เป็นต้น

ผลลัพธ์ที่ได้จาก Phase นี้ควรจะได้ Class เพิ่มเติม เช่น Class เกี่ยวกับการติดต่อกับผู้ใช้งาน คือหน้าต่าง ต่างๆ (GUI Classes), Class ที่จัดการและติดต่อกับฐานข้อมูล (Database Classes), การสนทนาระหว่าง Class (Dialogue) และประเภทของ Object ที่ต้องการเป็นแบบถาวร(Persistent Object), ความสัมพันธ์ระหว่าง Object , Ttransaction ต่าง ๆ ที่สนใจ Dynamic model จะมี Sequence Diagram และ State Diagram

Sequence Diagram เป็น Diagram ที่แสดงให้เห็นถึงการทำงานระหว่าง Object ต่างๆ ตามการส่งข้อความ(Message) และเมื่อเหตุการณ์(Event) ต่างๆ ได้เกิดขึ้น ซึ่งผู้เขียนโปรแกรมจะใช้ Diagramตัวนี้เป็นตัวช่วยเพื่อที่จะได้เขียนโปรแกรมให้ได้ตามที่ได้ออกแบบไว้ สัญลักษณ์ที่ใช้ใน Sequence Diagram มีดังนี้



ภาพที่ 3.16 สัญลักษณ์ที่ใช้ในSequence Diagram

สัญลักษณ์ที่ใช้ในซีเควนไดอะแกรม Sequence Diagram

- Object : คือ Instance ของ Class ที่ทำหน้าที่รับ-ส่งข้อความ (Message) จาก Object อื่นๆ และทำการตอบสนองตามข้อความนั้นๆ เพื่อให้เกิดการทำงานในขั้นตอนต่างๆ ของระบบ โดยจะใช้สัญลักษณ์เป็นเส้นตรงแนวตั้งและมีชื่อของ Object และ Class กำหนดอยู่ด้านล่าง

- Message : เป็นข้อความที่ส่งไปมาระหว่าง Object เพื่อสั่งให้ Object ทำงานตามที่กำหนด กล่าวได้ว่า Message คือชื่อของ Method ของ Object ที่ถูกเรียก ซึ่ง Message จะประกอบไปด้วยส่วนประกอบต่างๆ ดังต่อไปนี้

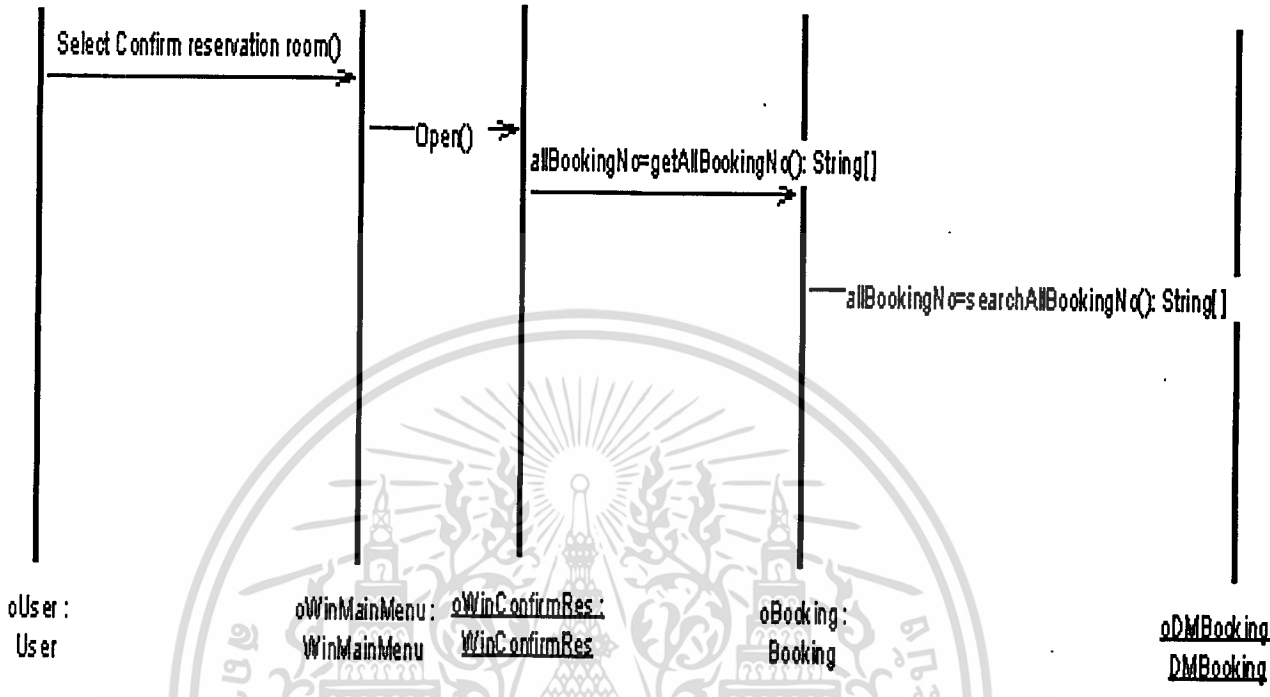
- Result Name : คือ ตัวแปรที่จะจัดเก็บค่าที่ได้จากการส่งข้อความ(Message)

- Parameters : คือ Argument ต่างๆ ที่ส่งให้กับ Object เพื่อทำงานตาม Method ที่กำหนดด้วย Message นั้นๆ

- Return Type : คือ ประเภทของข้อมูลที่ส่งกลับมาหลังจากการทำงานของการส่ง Method นั้นๆ ถ้า Method นั้น กำหนดว่ามีการส่งค่ากลับมา

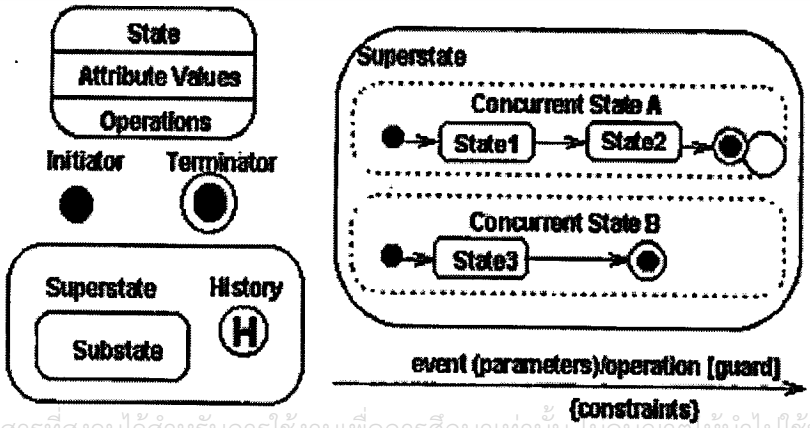
- Constraint : คือ เงื่อนไขในการส่ง Message ในการติดต่อสื่อสารกันระหว่าง Object

ตัวอย่าง Sequence Diagram : การเปิดหน้าต่างเพื่อทำการยืนยัน



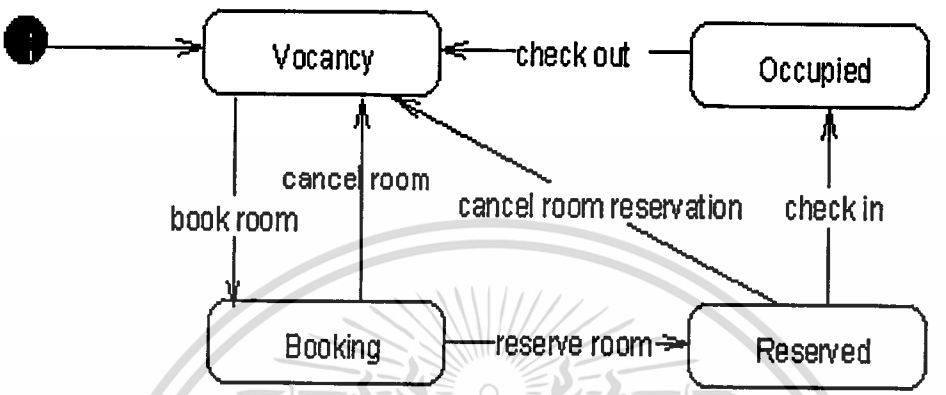
ภาพที่ 3.17 Sequence Diagram : OpenWinConfirmReservation

จากภาพที่ 3.17 สามารถอธิบายได้ว่า เมื่อผู้ใช้ระบบมีการเลือกที่จะทำการยืนยันห้องพักจากเมนูหลัก(WinMainMenu) เมนูหลักจะทำการเรียกให้หน้าต่างการยืนยันห้องพักเปิด และหน้าต่างยืนยันห้องพักต้องแสดงหมายเลขการสำรองห้องพักทั้งหมดมาแสดงที่หน้าจอ โดยจะร้องขอจาก Class บุคกิ้ง (Booking) และ Class บุคกิ้งต้องไปค้นหาหมายเลขการสำรองห้องพักทั้งหมดจาก Class DMBooking Class DMBooking ก็จะไปค้นหาข้อมูลมาจากฐานข้อมูล



ภาพที่ 3.18 สัญลักษณ์ที่ใช้ใน State Diagram

State Diagram จะเป็น **Diagram** ที่แสดงให้เห็นว่า Object หนึ่งๆ สามารถมีสถานะใดได้บ้าง และเกิดเหตุการณ์ใดจึงมีการเปลี่ยนสถานะ สัญลักษณ์ที่ใช้มีดังนี้



ภาพที่ 3.19 State Diagram ของ Objectห้องพัก(Room)

จากรูปเป็น State Diagram ของ Object ห้องพัก โดยที่ห้องพักจะมีสถานะที่เป็นไปทั้งหมด 4 สถานะคือ ว่าง(Vocancey), สำรอง(Booking), จอง(Reserved) และ มีลูกค้าพักอยู่(Occupied) โดยห้องพักจะเริ่มจากว่าง เมื่อมีการจองห้องพักแล้วห้องพักจะกลายเป็นห้องพักที่มีการสำรองแล้ว และเมื่อมีการยกเลิกการสำรองก็กลับเป็นห้องว่างได้ หรือ เมื่อมีการยืนยันการจองแล้วห้องพักที่มีการสำรองแล้วก็จะกลายเป็นห้องพักที่จองแล้ว จากห้องพักที่มีการจองแล้วก็สามารถยกเลิกได้จะกลายเป็นห้องว่าง หรือลูกค้าไม่ยกเลิกแต่เข้าพักก็จะกลายเป็นห้องที่มีลูกค้าพักอยู่ จากห้องที่มีลูกค้าพักอยู่จะกลับไปเป็นห้องว่างได้ต่อเมื่อลูกค้า check out ออกจากห้องนั้น

Phase 4 : Implement และ Testing

Implement และ Testing เป็น Phase ของการเขียนโปรแกรมและทดสอบให้ได้เป็นไปตามการวิเคราะห์ โดยจะใช้การโปรแกรมเชิงวัตถุ (Object Oriented Programming) ซึ่งการแปลงจากการวิเคราะห์และออกแบบไปเป็นการโปรแกรมเชิงวัตถุ ส่วนการ Testing ก็เหมือนเป็นการตรวจสอบคุณภาพของโปรแกรมที่เราทำขึ้นมาก็มีวิธีการตรวจสอบ Object รองรับ (Object Testing)

การแปลงจากการวิเคราะห์และออกแบบไปเป็นการโปรแกรมสามารถเชิงวัตถุทำได้ไม่ยุ่งยากเนื่องจาก ทั้งการวิเคราะห์และออกแบบเชิงวัตถุ กับการโปรแกรมเชิงวัตถุ นั้นสนับสนุนหลักการของ Object-Oriented ด้วย คือ

1. การปกป้องข้อมูล(Encapsulation)
2. การสืบทอดคุณสมบัติ(Inheritance)

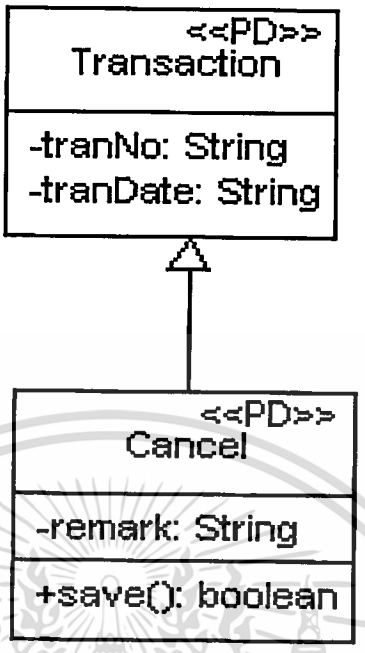
3.การที่ได้รับข้อความ(Message) เดียวแล้วสามารถตอบสนองได้หลายรูปแบบ (Polymorphism)

ปัจจุบันมีภาษาหลายภาษาที่สนับสนุนหลักการของ Object-Oriented เช่น C++, Smalltalk, Eiffel, Object C, Object Pascal (หรือ Delphi) และ Java โดยในที่นี้จะขอยกตัวอย่างด้วยภาษาจาวา (Java) เนื่องจากภาษาจาวาเป็นภาษาโปรแกรมเชิงวัตถุ ซึ่งสนับสนุนทั้ง 3 ข้อข้างต้น และยังสามารถทำงานข้ามระบบปฏิบัติการ(Cross Platform)ได้ด้วย หลักการในการแปลงจากการวิเคราะห์และออกแบบไปเป็นการโปรแกรม มีหลักง่ายๆ ดังนี้

การวิเคราะห์และออกแบบระบบเชิงวัตถุ (UML notation)	การโปรแกรมเชิงวัตถุ (ภาษาจาวา)
Component	เป็น package ในภาษา Java
Class	Class
Attribute	Attribute
method	Method
Attribute ต่างๆ ถ้าเป็น read only	เป็น set method
Inherit	Sub class จะต้อง extends จาก Super Class
ถ้า Component มีการอ้างอิงถึง Module หรือ Component อื่นๆ	จะต้อง Import Package นั้นๆ
Access Modifyer : Public (+) Private (-) Protected (#) Implementation (?)	เป็น Public ในจาวา เป็น Package ในจาวา เป็น Protected ในจาวา คือการกำหนด ณ ขณะ Implement เราอาจจะใช้ private ในภาษาจาวาก็ได้แล้วแต่ขอบเขตกรณีที่เราศึกษา

ตาราง 3.2 การเปรียบเทียบจากการวิเคราะห์ออกแบบไปเป็นการโปรแกรมเชิงวัตถุ(ภาษาจาวา)

ตัวอย่าง Class Cancel จะมี Attribute อยู่ 1 ตัว คือ remark มีประเภทข้อมูลเป็น String และมี Method อยู่ 1 Method คือ save คืนค่า boolean กลับ และ Class Cancel Inherite จาก Class Transaction



ภาพที่ 3.20 แสดง ตัวอย่าง Diagram

แปลงเป็นโปรแกรมภาษาจาวา ได้ดังนี้

```
public class Cancel extends Transaction {
        private String remark;
        public Cancel() {
                super();
        }
        public String get_remark() {
                return remark;
        }
        public void set_remark(String value) {
                remark = value;
        }
        public boolean save() {
                }
} //end class Cancel
```

Phase 5 : Maintenance

ในการ Maintenance นั้นจะทำได้สะดวกกว่า ยกตัวอย่างเช่นถ้ามีการเปลี่ยนแปลงโครงสร้างของ Class Cancel ก็จะทำการเปลี่ยนแปลงและ Implement Class นี้ใหม่เท่านั้น Class อื่นๆ ที่เกี่ยวข้องไม่จำเป็นต้องเปลี่ยนแปลง

จะเห็นว่าตั้งแต่ Phase ที่ 1 ถึง Phase ที่ 5 การวิเคราะห์และออกแบบเชิงวัตถุจะใช้เครื่องมือที่มีลักษณะเดียวกันคืออ้างอิงถึง Object มองทุกอย่างเป็น Object ทำให้เมื่อมีการเปลี่ยนแปลงแก้ไขก็ง่าย สะดวกต่อการจัดการ การส่งทอดระหว่าง Phase หนึ่งถึงอีก Phase หนึ่งจะไม่มีช่องว่างเพราะอ้างอิงถึง Object เหมือนกัน และมี Object-Oriented Case tool ที่สนับสนุนการวิเคราะห์และออกแบบเชิงวัตถุออกหลายค่าย เช่น Rose from Rational, Pladigm plus from Platinum, Object System from CRC, MetaCASE เป็นต้น ดังนั้นวิธีการวิเคราะห์และออกแบบระบบเชิงวัตถุจึงเป็นวิธีการที่น่าสนใจและมีแนวโน้มที่ดี

บทที่ 4
ปัญหาการเปลี่ยนแปลงตัวระบุชี้ (Identifier Problem)

4.1 Object Identifier (OID)

ตัวระบุชี้(Identifier) เป็นค่าที่ใช้ในการจำแนก Object แต่ละ Object ซึ่ง ค่าตัวระบุชี้ต้องเป็น Unique เท่านั้น คือไม่สามรถที่จะมีค่านี้ซ้ำกันอีกได้ แต่ละ Object จะมี ID ของตนที่ต่างกัน เราเรียกว่า Object Identifier หรือ OID ซึ่งเป็นค่าที่ระบบจะสร้างให้ตั้งแต่เริ่มสร้าง Object และค่า OID นี้จะไม่เปลี่ยนแปลง และจะไม่มีการนำค่า OID ที่ใช้แล้วมาสร้างใหม่ถึงแม้ว่า Object นั้นจะถูกลบทิ้งไปแล้วก็ตาม

4.2 ปัญหาเกี่ยวกับตัวระบุชี้

ในระบบฐานข้อมูลเชิงสัมพันธ์จะใช้ Unique key หรือเรียกว่า Identifier key เป็นกลไกที่ใช้ในการอ้างถึงข้อมูล ซึ่งการใช้ Identifier key ของฐานข้อมูลเชิงสัมพันธ์มีปัญหาต่าง ๆ ดังนี้

4.2.1 Modifying Identifier Key

การเปลี่ยนแปลงแก้ไขตัวระบุชี้ ในฐานข้อมูลเชิงสัมพันธ์ ไม่ยอมให้มีการเปลี่ยนแปลงแก้ไข Identifier Key หรือ Identify key ไม่สามารถทำการเปลี่ยนแปลงแก้ไขได้ เนื่องจากจะทำให้เกิดปัญหาต่าง ๆ ยกตัวอย่างเช่น ระบบงานแห่งหนึ่งใช้ ชื่อฝ่ายเป็น Identify key เพื่อใช้ระบุว่าพนักงานคนใดทำงานอยู่ฝ่ายใด ตอมาทางบริษัทเปลี่ยนนโยบายทำให้มีการเปลี่ยนชื่อฝ่าย ซึ่งจะส่งผลให้เกิดความไม่ต่อเนื่องของตัวระบุชี้ (Discontinuity Identity) อีกทั้งเมื่อมีการเปลี่ยนแปลง Identify key ดังกล่าวต้องมีการใช้ Referential Integrity เพื่อดูแลความถูกต้องของข้อมูลที่อ้างถึงด้วย(Consistency) กล่าวคือ เมื่อมีการเปลี่ยนแปลง Identify key ในระบบฐานข้อมูลเชิงสัมพันธ์ต้องมีกฎควบคุมความถูกต้องเรียกว่า Referential Integrity Rule ซึ่งมีในระบบฐานข้อมูลเชิงสัมพันธ์แทบทุก Product อยู่แล้ว แต่ก็ยังเป็นเรื่องที่ควบคุมดูแลได้ยาก เช่น เมื่อมีการเปลี่ยนชื่อฝ่าย ต้องตามแก้ไขข้อมูลของพนักงานที่อยู่ในฝ่ายนั้น ๆ อีกทั้งข้อมูลอื่น ๆ ที่ใช้ชื่อฝ่ายในการอ้างถึง

4.2.2 Noununiformity

คือ การที่ Identifier key ที่อยู่ต่างตารางกันมีรูปแบบข้อมูลที่ต่างกัน ปัญหานี้เป็นปัญหาที่เลือก Identify key มาจากตารางที่มีชนิดของข้อมูลที่ต่างกัน เช่น บริษัท ก ใช้ รหัสพนักงานเป็นตัวเลข(NUMERIC TYPE) แต่บริษัท ข ใช้เป็นอักษร(STRING TYPE) เมื่อต้องการจะรวมบริษัท ก และบริษัท ข เข้าเป็นบริษัทเดียวกัน จะเกิดปัญหาการระบุรหัสของพนักงานเนื่องจาก ใช้ Identify key คนละ Type กัน

ใน Object มีคุณสมบัติ OID ซึ่งเป็น Unique จะมีเพียงรูปแบบเดียวที่เหมือนกันทุก ๆ Object เนื่องจากระบบจะเป็นตัวสร้าง OID นี้ขึ้นมา กล่าวคือในระบบฐานข้อมูลเชิงสัมพันธ์ผู้ออกแบบเป็นคนกำหนดสร้าง Identify key ขึ้นมาเอง ดังนั้นเขาสามารถจะออกแบบให้มีลักษณะอย่างไรก็ได้ ซึ่งถ้าผู้ออกแบบทำการออกแบบให้ Identify key ต่างกันดังตัวอย่างข้างต้น ก็จะทำให้เกิดปัญหาดังกล่าว

4.2.3. Unnatural Join

เป็นปัญหาที่เกิดจากการ Join ของ Table คือการอ้างถึงข้อมูลที่อยู่ต่างตารางกัน ในระบบฐานข้อมูลเชิงสัมพันธ์จะใช้วิธีการ Join เพื่อดึงข้อมูลนั้นมาใช้ซึ่งการ Join นั้น ถ้าข้อมูลมีจำนวนมากจะทำให้ใช้เวลาและทรัพยากรมากในการ Join Table กัน เช่น

Employee relation

Officer(Name, Age, Address, Salary, DeptName)

Department relation

Department(Name, Budget, Location ,…)

เมื่อเราต้องการ List รายชื่อพนักงานทุกคนพร้อมทั้งสถานที่ที่ทำงาน จะเขียนเป็น SQL ได้ดังนี้

```
SELECT Officer.Name, Department.Location
FROM Officer, Department
WHERE Officer.DeptName = Department.Name
```

จากตัวอย่างข้างบนแสดงให้เห็นว่า สิ่งที่ต้องการจะทราบจริง ๆ คือชื่อพนักงานและสถานที่ทำงาน ซึ่งจากการทำ Normalization ของระบบฐานข้อมูลเชิงสัมพันธ์ จะทำให้ข้อมูลที่อยู่ต่างตารางถูกจำกัดคือไม่สามารถจัดการกับข้อมูลในตารางอื่น ๆ ได้ หรือกล่าวได้ว่า ไม่อนุญาติให้กำหนดจัดการกับ Tuple, Relation หรือ Complex Object Type อื่น ๆ ของ Attribute ได้ จากการที่เราใช้การ

Normalization กับระบบฐานข้อมูลเชิงสัมพันธ์ เพื่อจัดการกับปัญหาการ Insert, Delete และ Update หรือที่เรียกว่า Anamalize จะทำให้สูญเสีย Semantic ที่เชื่อมระหว่าง Object ในฐานข้อมูลไป ซึ่งในระบบฐานข้อมูลเชิงสัมพันธ์ได้ใช้ Foreign key มาใช้ร่วมกับ SQL เพื่อใช้งานแทน Semantic ที่สูญเสียไปแทน ซึ่งในการใช้ระบบฐานข้อมูลเชิงสัมพันธ์ และในระบบฐานข้อมูลเชิงสัมพันธ์การที่จะสร้าง Complex Object จะต้องสร้าง Relation มากกว่า 1 Relation มาทำการ Join กัน เพื่อที่จะให้เกิด Complex Object ขึ้นมา ยกตัวอย่าง Complex Object เช่น

Object ของพนักงานที่ทำงานในหน่วยงานหนึ่ง ซึ่งจะต้องประกอบด้วย ชื่อ ตำแหน่ง เงินเดือน ประวัติการทำงาน ซึ่งในประวัติการทำงานประกอบด้วย ปีที่ทำงาน ผลงาน ในตัวอย่างนี้ถ้าใช้ระบบฐานข้อมูลเชิงสัมพันธ์จะต้องสร้างเป็น 2 ตารางได้แก่

Employee relation

Officer(Name, Position, Salary)

Profile relation

Profile(Year, Product, OfficeName)

จะเห็นได้ว่าจะต้องใช้การ Join ทั้ง 2 Relation เพื่อที่จะทำการดูประวัติของพนักงานคนนี้เป็นยังไง ซึ่งถ้าปัญหายิ่งซับซ้อนมากก็จะยิ่งใช้ Relation มากตามไปด้วย

ปัญหาการเปลี่ยนแปลงตัวระบุชี้เป็นปัญหาที่ทำให้การพัฒนาระบบแบบเดิมมีปัญหาเนื่องจากการพัฒนาระบบแบบเดิมจะใช้การออกแบบในเชิงความสัมพันธ์(Relation) ซึ่งมีการจัดเก็บข้อมูลในลักษณะของตาราง(Table) โดยที่ในแต่ละ Table จำเป็นต้องมีตัวระบุชี้(Identifier) หรือเราเรียกว่าคีย์หลัก(Primary Key) ซึ่งจากนิยาม Primary key นี้คือ เป็นค่าที่สามารถระบุข้อมูลในตาราง ซึ่งค่านี้จะต้องไม่ซ้ำกัน(Unique) การระบบที่ต้องมีการเปลี่ยนแปลงค่า Primary key นี้ จะมีปัญหาอย่างมากในเรื่องการ Update ข้อมูล ในงานวิจัยนี้จะขอยกตัวอย่างเป็นกรณี ๆ ไป และจะทำการนำเอาแนวความคิดเชิงวัตถุมาประยุกต์ในการจัดการกับปัญหาเป็นกรณีไปเช่นกัน

- <u>กรณีที่ 1</u> : การที่นักศึกษาชื่อ นาย A มีรหัสประจำตัว 41067111 กำลังศึกษาอยู่คณะวิศวกรรมศาสตร์ ในปี 2541 พอปี 2542 นาย A ต้องการที่จะเปลี่ยนคณะที่เรียน เป็นคณะเทคโนโลยีสารสนเทศแทน นาย A จะมีรหัสนักศึกษาเปลี่ยนไปเป็น 42067111 ถ้าเราใช้ระบบที่เป็นเชิงสัมพันธ์ จะเห็นได้ว่า ต้องมีการ Add ข้อมูลของนาย A เข้าไปในฐานข้อมูลใหม่ ทั้ง ๆ ที่นาย A ยังคงเป็นนักศึกษาคนเดิมที่เคยมีประวัติอยู่แล้ว เพียงแต่เปลี่ยนแปลงที่ตัวระบุชี้ ในที่นี้คือ รหัสนักศึกษา

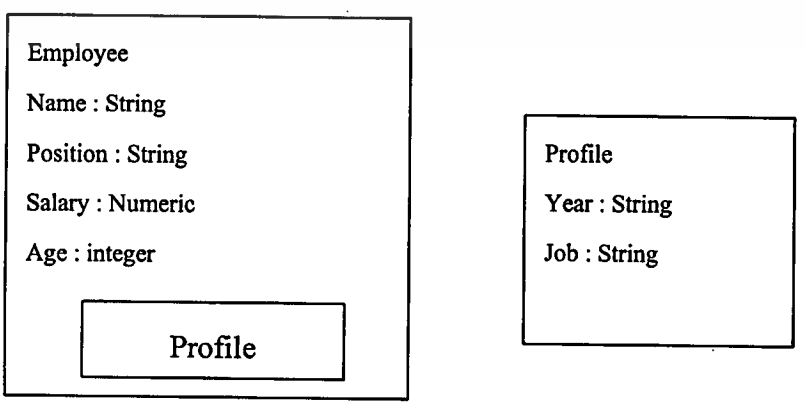
- กรณีที่ 2 : นาย B มีประวัติการทำงานมา 10 ปี พอปี ที่ 11 นาย B เปลี่ยนชื่อเป็น นาย C แทน เมื่อต้องการจะดูประวัติของนาย C ย้อนหลังไป 5 ปี การทำงานด้วยฐานข้อมูลเชิงสัมพันธ์ จะไม่สามารถค้นหาได้(กรณีนี้ให้ชื่อ เป็น Primary key ในระบบฐานข้อมูลเชิงสัมพันธ์)

4.3 การนำแนวคิดของ Object-Oriented มาใช้ในการแก้ปัญหา

จากคุณสมบัติของ Object-Oriented ดังที่ได้กล่าวมาข้างต้น เห็นได้ว่า Object-Oriented มีคุณสมบัติ OID(Object Identifier) ซึ่งในแต่ละ Object มี OID ที่เป็น Unique และจะไม่มีการนำ OID นั้นกลับมาใช้ซ้ำอีกไม่ว่า Object ที่ใช้ OIDนั้นจะถูกลบออกไปแล้วก็ตาม

ในระบบฐานข้อมูลเชิงสัมพันธ์ การที่เราจะอ้างถึงข้อมูลในตารางเราจะอ้างถึง Primary key ของตารางนั้น ๆ ซึ่งถ้าเราทำการเปลี่ยนแปลงค่า Primary key จะทำให้เกิดความเปลี่ยนแปลงซึ่งจะก่อให้เกิดความยุ่งยาก เช่น ในตัวอย่างที่ 1 ในข้างต้น เราใช้ รหัสนักศึกษาเป็น Primary key จะเห็นได้ว่าหากมีการเปลี่ยนแปลงรหัสนักศึกษาย่อมหมายถึงเป็นนักศึกษาคนใหม่ทั้ง ๆ ที่ในความจริงก็ยังคงเป็น นาย A คนเดิม

การใช้ OID (Object Identifier) มาเป็นตัวระบุชี้ของหลักการเชิงวัตถุ ก็คล้ายกับการมี Primary key ของระบบฐานข้อมูลเชิงสัมพันธ์ ต่างกันคือ เราไม่สามารถเข้าถึง OID ได้ เนื่องจาก OID จะเป็นข้อมูลที่ระบุว่า Object นั้น ๆ มีหมายเลขใด โดยค่าของ OID นั้นจะถูก Generate ขึ้นเมื่อมีการสร้าง Object นั้น ๆ ขึ้นมาโดยอัตโนมัติ ซึ่งในส่วนนี้ OODBMS จะทำหน้าที่ในการ Generate OIDให้กับ Object ซึ่งเราไม่สามารถทราบได้ว่า Object นั้น ๆ มี OID เป็นอะไร หรือกล่าวได้ว่าเราจะสามารถมองเห็น OID ได้ ในการทำงานเมื่อเราจะทำงานกับ Object ใด เช่นจากตัวอย่าง Complex Object ที่กล่าวมาแล้วในตอนต้น ถ้าเราออกแบบโดยใช้หลักการในเชิงวัตถุมาใช้เราจะได้ Object Employee ดังนี้



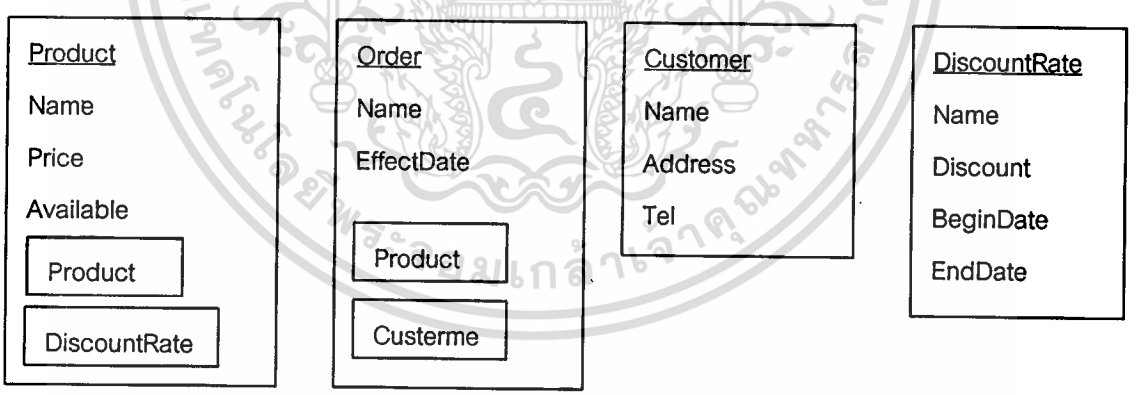
ภาพที่ 4.1 แสดง Object Employee และ Object Profile

จากรูปที่ 2 เห็นว่าลักษณะของ Object จะสามารถนำ Object อื่นมาเป็น Attribute ได้ โดยจะอ้างถึงด้วย OID เราเรียก Object ในที่นี้ซึ่งก็คือ Object ที่เก็บประวัติการทำงานของพนักงานว่า Embeded Object ในการที่เราจะอ้างถึงข้อมูลใน Object Profile จะอ้างถึงได้โดยใช้ OID ตามที่ได้กล่าวมาในตอนต้น ซึ่งถ้าเป็นระบบฐานข้อมูลเชิงสัมพันธ์ใช้การ Join 2 Relation จะมีความยุ่งยากมาก หรือในบางปัญหาไม่สามารถทำงานได้เลย

ทำการแก้ไข ชื่อของนักศึกษา เวลา Update Data จะเป็นการ Update ข้อมูลที่เกี่ยวของกับ Object นั้นโดยอัตโนมัติ ในตัวอย่าง แม้ว่าเราจะเปลี่ยนชื่อจากนาย A เป็นนาย B ก็ตาม แต่ Object ของนาย A ก็ยังเป็น Object เดิม ดังนั้นเมื่อต้องการดูประวัติหรือข้อมูลอื่น ๆ ของนาย B ก็สามารถทำได้

4.4 ตัวอย่างการการแก้ปัญหาโดยใช้ระบบฐานข้อมูลเชิงวัตถุ

ในการที่จะแก้ปัญหาดังกล่าวจะขอยกตัวอย่างการทำงานโดยทั่วไปของระบบฐานข้อมูลนั่นก็คือการ Insertion, Deletion และการ Update มาประยุกต์ใช้กับระบบงานหนึ่ง ในที่นี้ของยกตัวอย่างระบบการสั่งซื้อของของร้านค้าแห่งหนึ่ง โดยที่จะทำการบันทึกรายการสั่งซื้อที่ลูกค้าสั่งมา โดยจะมีลักษณะของ Object ดังนี้



ภาพที่ 4.2 แสดง Class ของการสั่งซื้อสินค้า

จะเห็นได้ว่ามี 3 Object ได้แก่ Product, Order และ DiscountRate มีรายละเอียดดังนี้

1. Product เป็นข้อมูลของสินค้าที่มีในร้านประกอบด้วย ชื่อสินค้า(Name) ราคา(Price) และสินค้านั้นมีอยู่ในร้านหรือว่าหมดแล้ว(Available) และรายการส่วนลด(DiscountRate)

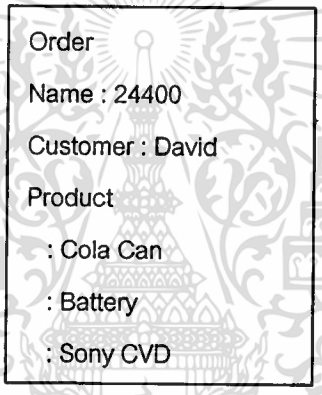
2. Order เป็นข้อมูลของรายการสินค้าที่ลูกค้าสั่งซื้อประกอบด้วย ชื่อรายการสั่งซื้อ(Name) วันที่ที่สั่งซื้อ(EffectDate) และสินค้าที่สั่ง(Product) และลูกค้าที่สั่ง(Customer)

3. DiscountRate เป็นข้อมูลรายการการลดราคา ประกอบด้วย ชื่อรายการ(Name) อัตราส่วนลด(Discount) วันเริ่มต้นรายการ(BeginDate) และ วันสิ้นสุดรายการ(EndDate)

4. Customer เป็นข้อมูลของลูกค้าประกอบด้วย ชื่อ(Name) ที่อยู่(Address)และหมายเลขโทรศัพท์(Tel)

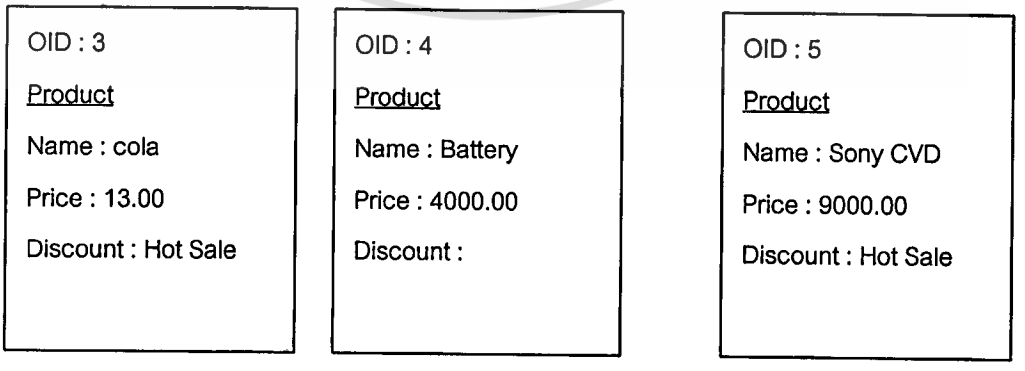
ระบบนี้ลูกค้าสามารถสั่งซื้อสินค้าได้ทีละหลาย ๆ Product และแต่ละ Product ก็อาจมีรายการส่วนลดที่ต่างกับไป ซึ่งถ้าเราใช้ระบบฐานข้อมูลเชิงสัมพันธ์มาออกแบบระบบนี้

1. Insertion – การเพิ่มรายการสินค้าที่สั่ง ในการทำงานของ Object ใช้ OID เป็นตัวช่วยในการอ้างถึงอีก Object หนึ่ง ในที่นี้การสั่งสินค้าเป็นการเพิ่ม Product เข้ามาในรายการสั่งซื้อ(Order) มีหลักการดังนี้



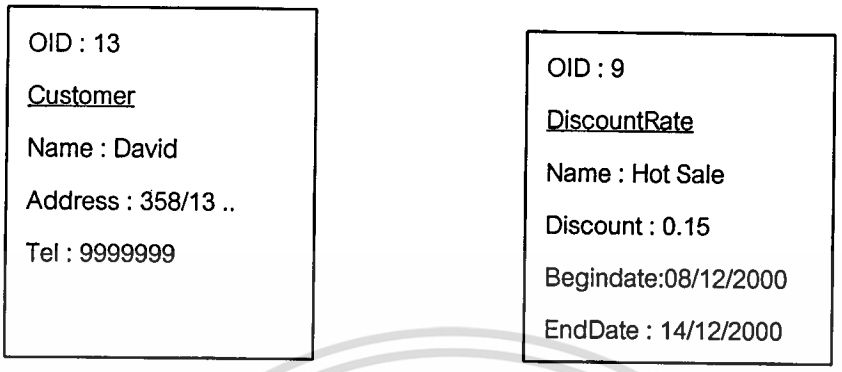
ภาพที่ 4.3 แสดง Class Order

จากภาพลูกค้าชื่อ David ได้สั่งซื้อสินค้า 3 อย่าง ดังที่ได้กล่าวไว้แล้วว่าการอ้างถึง Object อื่น จะใช้ OID ดังนี้



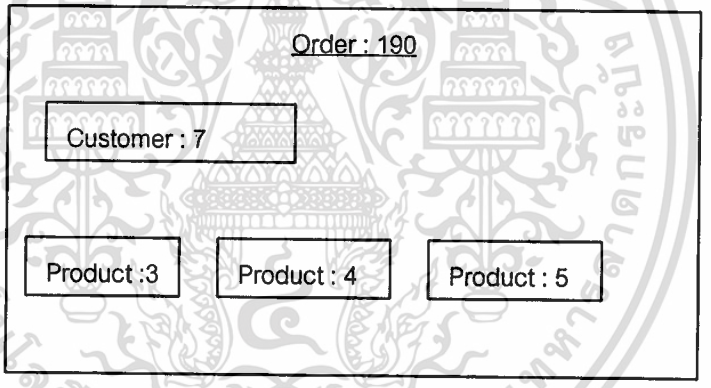
ภาพที่ 4.4 แสดงการสั่งซื้อสินค้าของ David

จากภาพทั้ง 3 Product จะมี OID ที่ต่างกัน ซึ่ง OID นี้ระบบจะเป็นตัวการสร้างให้



ภาพที่ 4.5 แสดง Instance ของ Class Customer กับ DiscountRate

เมื่อทำการสั่งรายการสินค้าดังกล่าวจะได้ Object Order ดังนี้



ภาพที่ 4.6 แสดงการเก็บค่าใน Class Order

Object Order จะเก็บข้อมูลในลักษณะดังภาพ คือ จะทำการอ้างถึงข้อมูลที่อยู่ต่าง Object กันไปด้วย OID

2. Delete – การยกเลิกการสินค้าที่สั่ง ในการทำงานของ Object ใช้ OID เป็นตัวช่วยในการอ้างถึงเช่นเดียวกัน จะเห็นได้ว่าข้อมูลที่ถูกลบจะไม่ส่งผลไปถึงข้อมูลอื่น เนื่องจากจะเป็นการลบเฉพาะข้อมูล Object นั้นเท่านั้นจะไม่ส่งผลไปถึง Object อื่น ๆ ในตัวอย่างการลบรายการสินค้าของนาย David ก็ไม่ทำให้ข้อมูลส่วนอื่น ๆ เช่น ข้อมูลของรายการสินค้า หรือ ข้อมูลลูกค้าเสียหายไป และถึงแม้ว่าจะทำการลบรายการสั่งซื้อดังตัวอย่างที่มี OID เป็น 109 แล้วก็ตาม แต่ระบบจะไม่มีการนำ OID 109 มาใช้งานอีกครั้ง

3. Update – ในส่วนของการเปลี่ยนแปลงแก้ไข ซึ่งเป็นส่วนที่งานวิจัยนี้เน้นเป็นพิเศษ ก็จะคล้าย ๆ กับการ Delete คือจะไม่ส่งผลกระทบไปยัง Object อื่น เราสามารถทำการแก้ไขได้ทุก

Property ใน Object เนื่องจากแต่ละ Object มี OID ที่ไม่สามารถเปลี่ยนแปลงได้อยู่แล้ว ซึ่งถ้าเป็นระบบฐานข้อมูลเชิงสัมพันธ์เราสามารถแก้ไขอะไรก็ได้ยกเว้น Identity key เนื่องจากจะส่งผลกระทบต่อ Table อื่น ๆ มากมาย เช่นถ้าระบบนี้ออกแบบโดยใช้ระบบฐานข้อมูลเชิงสัมพันธ์จะได้ Relation ต่าง ๆ ดังนี้

DiscountRate Relation

Discount(Name, Discount, BeginDate, EndDate)

Product Relation

Product(Name, Price, Available, DiscountName)

Order Relation

Order(Name , ProductName, CustomerName)

Customer Relation

Customer(Name, Address, Tel)

ในตัวอย่างถ้าเกิดเรามีการเปลี่ยนแปลง Identity key บางค่าเช่น นาย David เปลี่ยนชื่อเป็น Steve เราจะต้องทำการแก้ไขข้อมูลใน Order Relation ด้วย หรือถ้าเกิด เปลี่ยนชื่อรายการอัตราส่วนลดก็ต้องทำการแก้ Product Relation ตามด้วย

จากตัวอย่างที่ยกมาจะเห็นได้ว่าในการทำงานขั้นพื้นฐานการเปลี่ยนแปลง Identity key ของระบบฐานข้อมูลเชิงสัมพันธ์จะทำให้เกิดความยุ่งยากมากเมื่อเทียบกับระบบฐานข้อมูลเชิงวัตถุ และยังมีบางปัญหาที่ระบบฐานข้อมูลเชิงสัมพันธ์ไม่สามารถจัดการได้หรือต้องใช้ความยุ่งยากมากเช่น จากตัวอย่างเมื่อเราต้องการทราบว่าตอนนี้มีลูกค้าที่อยู่ละแวกใดที่นิยมสั่งสินค้าใดบ้างที่ลด 10 – 15 % จากคำถามนี้เราต้องทำการ Join Table หลาย ๆ Table ซึ่งถ้าข้อมูลมีจำนวนมากจะใช้เวลานานมากดังที่กล่าวไปแล้วดังนั้น ระบบฐานข้อมูลเชิงสัมพันธ์จึงไม่เหมาะที่จะจัดการกับระบบที่มีความซับซ้อนมาก ๆ หรือมีลักษณะเป็น Complex Object

บทที่ 5
แนะนำระบบฐานข้อมูลเชิงวัตถุ CACHE

CACHE เป็น ระบบฐานข้อมูลเชิงวัตถุที่พัฒนาโดย บริษัท InterSystems Corporation ในปี 1999 โดยเป็นระบบฐานข้อมูลที่สามารถรองรับการทำงานเชิงวัตถุได้เต็มประสิทธิภาพ อีกทั้งมีความง่ายในการพัฒนา

5.1 แนะนำระบบฐานข้อมูลเชิงวัตถุ CACHE ขั้นต้น

CACHE มีความสามารถในการรองรับการทำงานได้หลากหลาย Platform ทั้งในด้าน Operating System และ Hardware ดูได้จากตารางดังนี้

Operating System	Hardware
Compaq Tru64 UNIX 4.0F and 5.0 (Digital UNIX)	Alpha computers
Digital OpenVMS 7.1 and 7.2	Alpha computers
DG/UX 4.2	AViiON (Intel) computers
HP/UX 11.0	Hewlett-Packard computers
IBM AIX 4.3.2, 4.3.3	IBM PowerPC computers
IBM AIX 4.3.2	IBM RS/6000 computers
Red Hat Linux 6.1	Intel-based computers
SCO UNIX 5.0, 5.02, 5.0.4, and 5.0.5	Intel-based computers
SCO UnixWare 7.1	Intel-based computers
Sun Solaris 2.7	Intel-based and SPARC-based computers
Windows NT 4.0 (with Service Pack 4, 5, 6, or 6A)	Intel-based and Alpha-based computers
Windows 2000	Intel-based computers
Windows 95 and 98	Intel-based computers

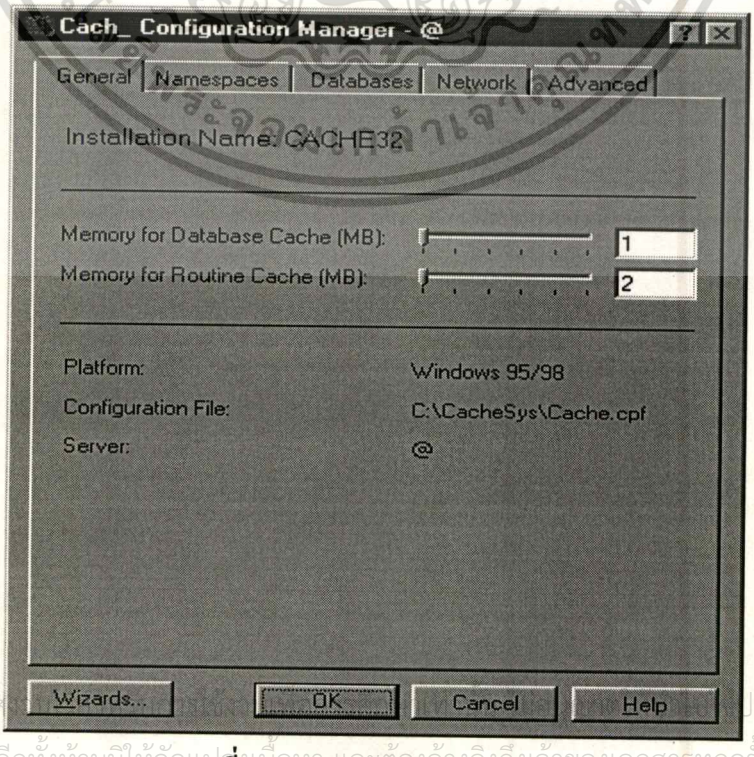
จากตารางแสดงให้เห็นว่า CACHE สามารถใช้งานได้หลากหลาย Platform ในงานวิจัยนี้ได้ใช้ Windows 95 เป็น Operating System

5.2 Component ต่าง ๆ ใน CACHE

ในตัว CACHE เองจะประกอบด้วยส่วนต่าง ๆ แบ่งเป็น Component ต่าง ดังนี้

1. CACHE Configuration: ทำหน้าที่ในการควบคุมจัดการการทำงานของระบบฐานข้อมูล ซึ่งสามารถจัดแบ่งการจัดการเป็น Section ซึ่งเราสามารถกำหนดค่าต่าง ๆ ได้ ดังนี้

- SQL section - SQL system-wide settings
- License section -WAN (Wide-area Network) License Allocation
- General section -กำหนดจำนวนงาน(JOB)ในตอนเริ่มต้น และกำหนดเวลาที่จะ Time out
- Process section - สามารถเพิ่มหน่วยความจำได้ถึง 16 MB ต่อ Process
- ViewPoint section - ดูข้อมูลการประมวลผลของ Server
- Network section - DTM collation, เราสามารถกำหนด DTM-NetBIOS Server เพื่อที่จะใช้ DTM Compatible Collation type for DTM globals
- Journal section – สามารถกำหนดจำนวนวันก่อนที่จะทำการเคลียร์ข้อผิดพลาดต่างๆ (Error) ออกจากระบบ
- Miscellaneous section - ทำหน้าที่ในการสร้าง Log File เพื่อใช้ในการ Rollback



ภาพที่ 5.1 CACHE Configuration

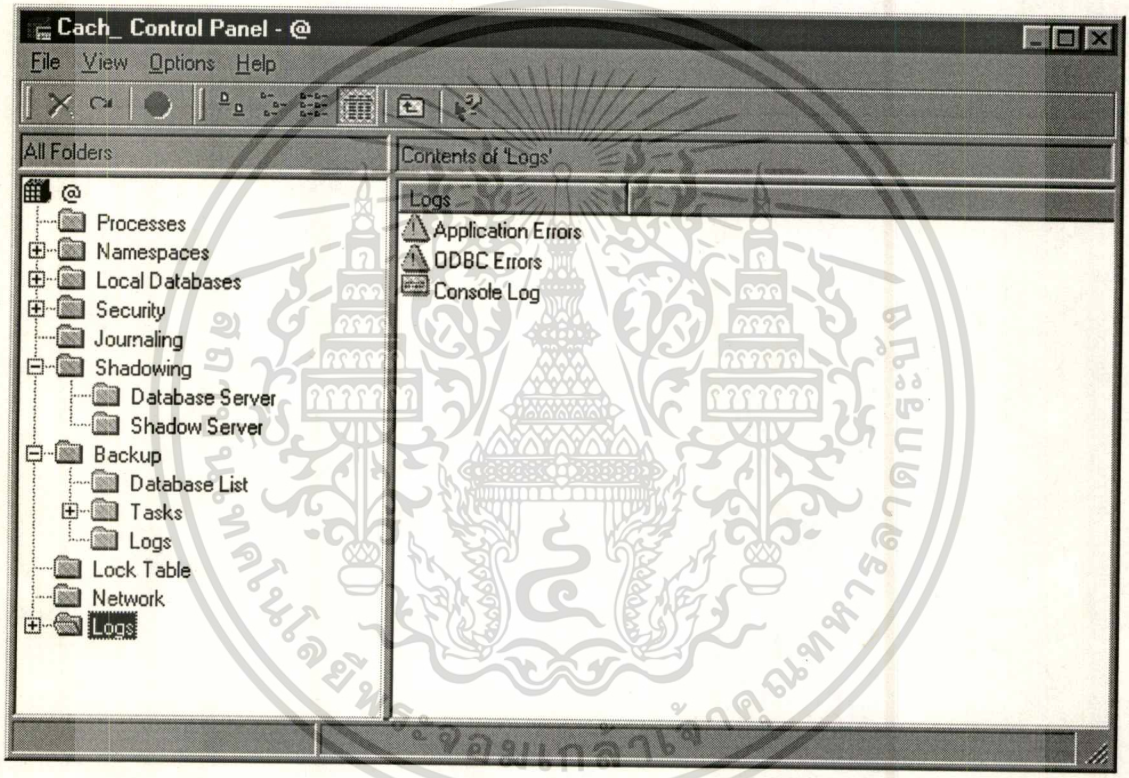
2. Control Panel : ประกอบด้วย Section ต่าง ๆ ดังนี้

- The Namespaces section - สามารถให้เราทำการลบ Namespaces

- The Local Databases section – ประกอบด้วย Delete option รวมทั้ง แสดงเวลาล่าสุด และทำการตรวจสอบความถูกต้องของข้อมูล(Integrity Check)

- Print Section - ส่วนของการพิมพ์

- ส่วนที่แสดงข้อมูลต่าง ๆ ของการประมวลผล



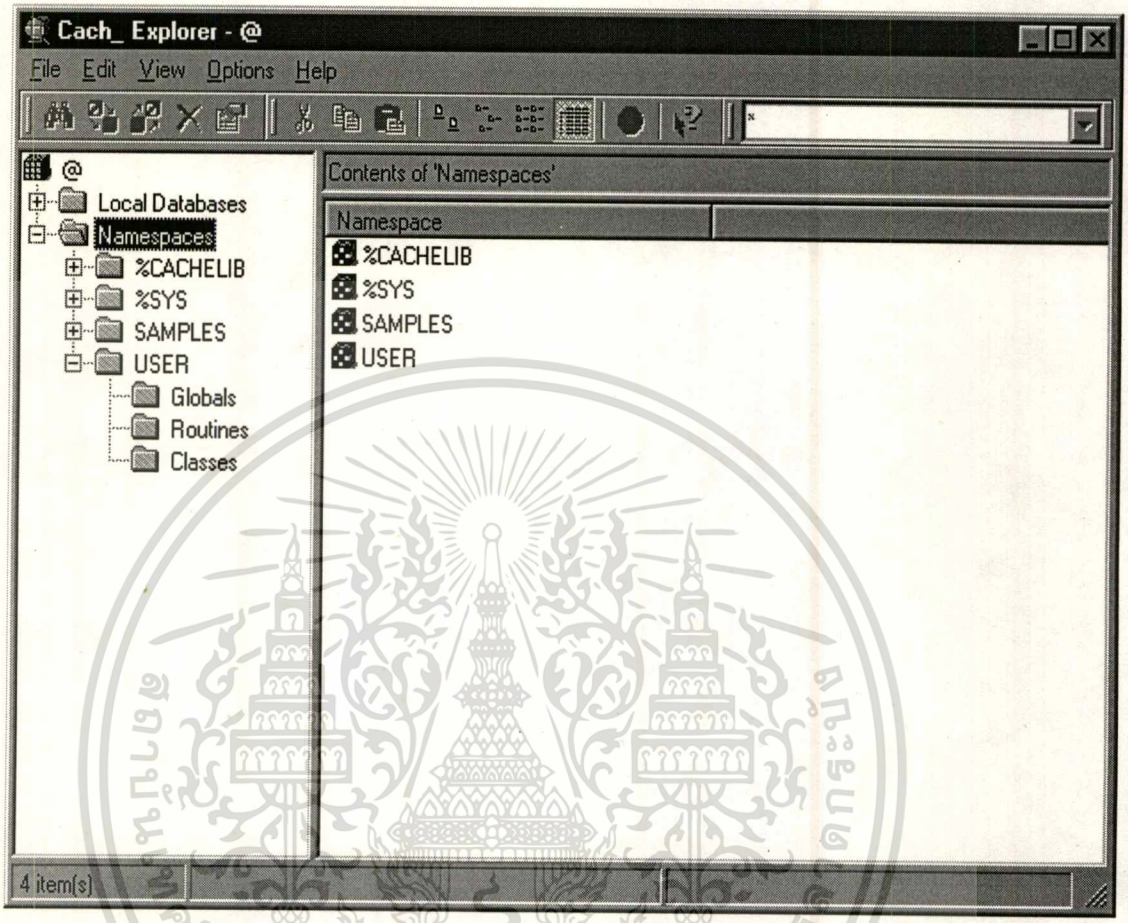
ภาพที่ 5.2 Control Panel

3. CACHE Explorer : มีลักษณะต่าง ๆ ดังนี้

- ในระดับ Routines: สามารถที่จะเทำการค้นหาและทำการแทนที่(Find/Replace) และ ทำการ Compile functionality ได้ เสมือนกับความสามารถในการที่จะกำหนด Mode ของ Language และ การแสดงจำนวน n บรรทัดแรก

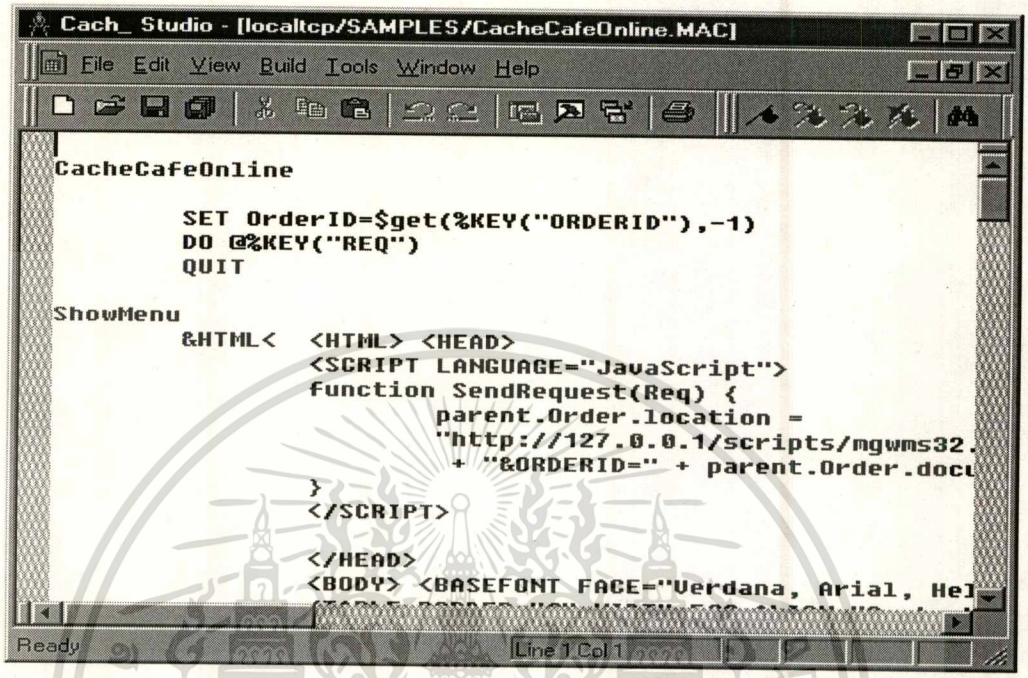
- ในระดับ Globals – สามารถที่จะทำการค้นหาและพิมพ์ได้เสมือนกับการ cut-and-paste operations ใน the Global Viewer

- ในระดับ Classes - สามารถทำการพิมพ์ได้

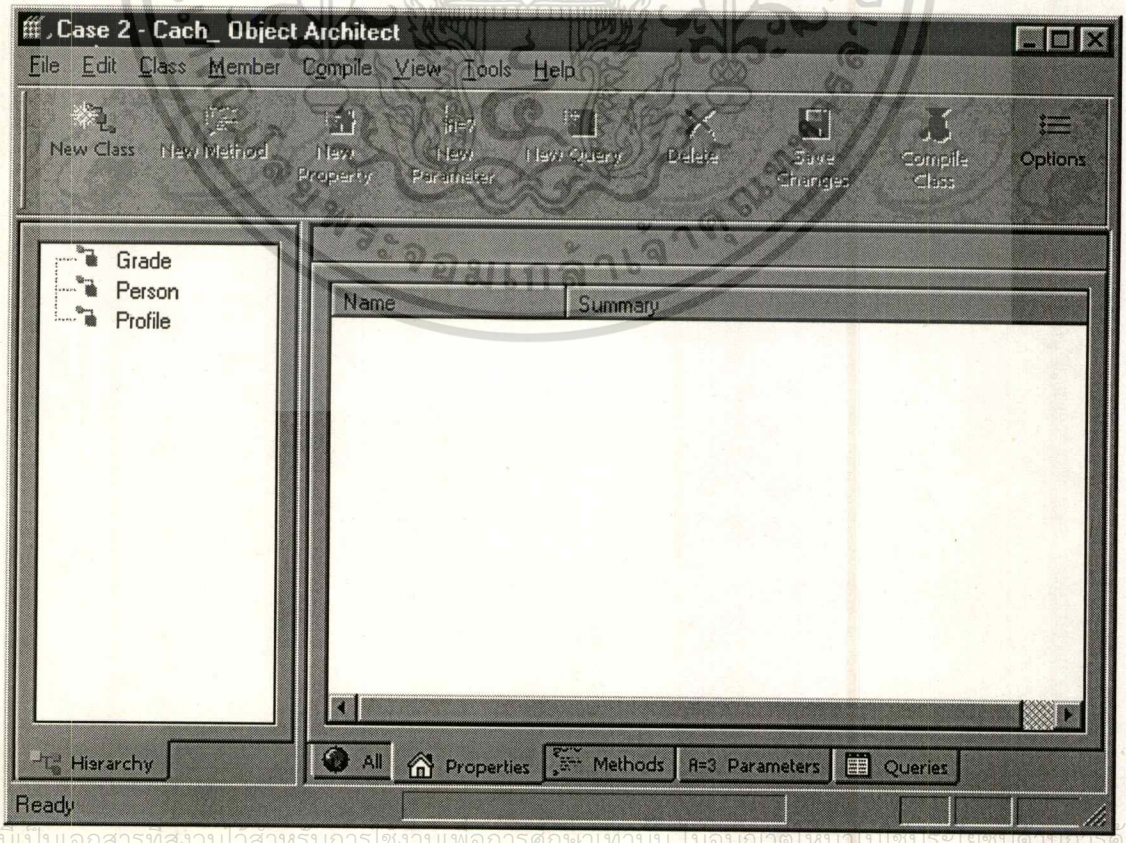


ภาพที่ 5.3 CACHE Explorer

4. CACHE Studio และ Object Architecture : เป็นเครื่องมือที่ใช้ในขั้นตอนการออกแบบเชิงวัตถุ เราสามารถออกแบบโดยอิงหลักของ UML คือออกแบบ Class Diagram โดยเราสามารถระบุ Property และ Method ของ Object รวมไปถึงสามารถสร้าง Query ที่จะเรียกดูข้อมูลจากฐานข้อมูลได้อีกด้วย จากนั้นสามารถทำการ Compile Object ที่เราออกแบบเสร็จแล้วให้เป็น Class ที่พร้อมจะนำไปใช้งานได้เก็บอยู่ในฐานข้อมูลของ CACHE ส่วนใน CACHE Studio เป็นเสมือน Tools ในการเขียน ObjectScript ซึ่งเราจะไม่เน้นในงานวิจัยนี้ โดยจะมีหน้าจอดังนี้



ภาพที่ 5.4 CACHE Studio



ภาพที่ 5.5 Object Architecture

5. CACHE SQL Manager

6. CACHE ObjectScript

โดยในงานวิจัยนี้จะ เน้นการใช้งานในส่วนของ CACHE Architecture ซึ่งใช้ในการสร้าง Class เป็นส่วนใหญ่ จะไม่กล่าวถึงในส่วนอื่น ๆ

5.3 การพัฒนาระบบงานโดยใช้ CACHE

ในส่วนนี้จะนำเสนอการพัฒนาระบบโดยใช้ CACHE เป็นเครื่องมือในการพัฒนา เช่นเดียวกันกับการพัฒนา Application ส่วนใหญ่ซึ่งจะประกอบด้วยส่วนต่าง ๆ ดังนี้

- Data - ส่วนของข้อมูล
- Logic - ส่วนของ Business Rule
- User Interface - ส่วนที่ติดต่อกับผู้ใช้
- Output - ส่วนของข้อมูลที่ได้มาหลังจากผ่านการประมวลผลแล้ว

CACHE ใช้หลักการพัฒนาเชิงวัตถุโดยจะนำเอา Data และ Logic มารวมกันเป็น Object โดยจะมีการกำหนด Property เป็นส่วนของ Data และ กำหนด Method เป็นส่วนของ Logic เก็บไว้ใน Class โดยเมื่อทำการเปรียบเทียบกับระบบฐานข้อมูลทั่ว ๆ ไปจะสามารถสร้างเป็นตารางเปรียบเทียบได้ดังนี้

Caché	raditional Relational	
Rich Properties	Simple Columns	- ในเทคโนโลยีแบบ Relational การนิยามข้อมูลจะนิยามเป็น Column ซึ่งต้องเป็นข้อมูลอย่างธรรมดา และเป็น Single-valued data types - Caché สามารถสร้างลักษณะของข้อมูลให้สามารถรองรับการทำงานให้ดียิ่งขึ้นโดยสามารถที่จะนิยามลักษณะแบบใหม่ที่จะเพิ่ม Object หนึ่ง ๆ เข้าไว้ในอี Object หนึ่งได้ หรือที่เรียกว่า Complex Object
Collections	Extra Tables	- Caché - แต่ละรายการสามารถที่จะประกอบด้วยหลาย ๆ Item ได้ โดยการใช้ Collection - Relational database เราต้องทำการแบ่ง Table แล้วทำการ Join Table เพื่อที่จะรองรับการทำงานแบบ Complex Object ได้
Methods	Loose Logic	- ในระบบฐานข้อมูลเชิงสัมพันธ์ไม่มีส่วนของการกำหนด Logic

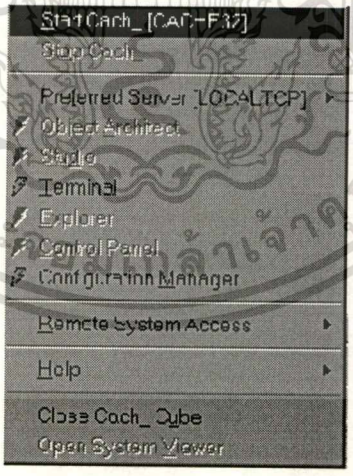
ตาราง 5.2 แสดงการเปรียบเทียบ CACHEกับ Traditional

ในส่วน User Interface เมื่อเราสร้าง Class ได้แล้วเราสามารถที่จะนำ Tools ต่าง ๆ เข้ามาเข้ามาช่วยสร้าง Application ได้หลากหลาย เช่น Visual Basic หรือ Delphi ในกรณีที่ Application ที่เราพัฒนาทำงานในลักษณะ Windows และสามารถใช้ Java Tools ในการพัฒนา Application ที่มีการทำงานในลักษณะ Browser ในส่วนของ Output ในการที่เราต้องการข้อมูลจากระบบเราสามารถที่จะใช้ SQL ได้ทำนองเดียวกันกับที่ใช้งานกับ Relation โดยมที่เราจะเขียน SQL เองหรือสามารถใช้ Tools อื่น ๆ ช่วยในการ Generate ก็ได้ เช่น Crystal Report, Business Object หรือ MS Access

5.4 ขั้นตอนการพัฒนาระบบงานโดย CACHE

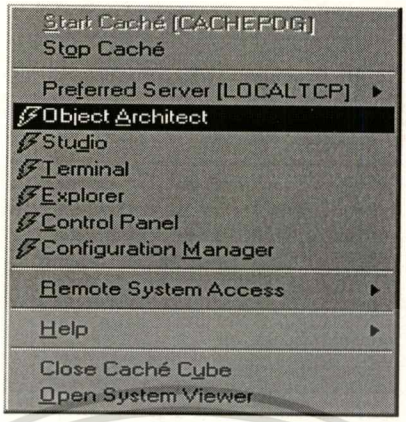
ขั้นตอนในการพัฒนาระบบโดยใช้ CACHE จะเริ่มจากเราต้องมี Class Diagram ที่ออกแบบมาก่อนแล้ว ซึ่งเราสามารถที่จะใช้ Rational Rose เป็นเครื่องมือช่วยในการออกแบบได้ หลังจากที่เราได้ Object Model ที่ออกแบบแล้ว เราจะทำการสร้างระบบงานโดยมีขั้นตอนดังต่อไปนี้

ขั้นที่ 1 Starting CACHE : ทำการ Start CACHE โดยการ Click ขวา ที่ Menu Bar ของ Window ดังภาพ เพื่อทำการ Start CACHE Engine



ภาพที่ 5.6 การ Start CACHE

ขั้นที่ 2 Object Architecture: หลังจากที่เราเปิด CACHE แล้ว ขั้นตอนต่อมาเราจะทำการสร้าง Class โดยใช้ Object Architecture โดยทำตามรูปภาพ

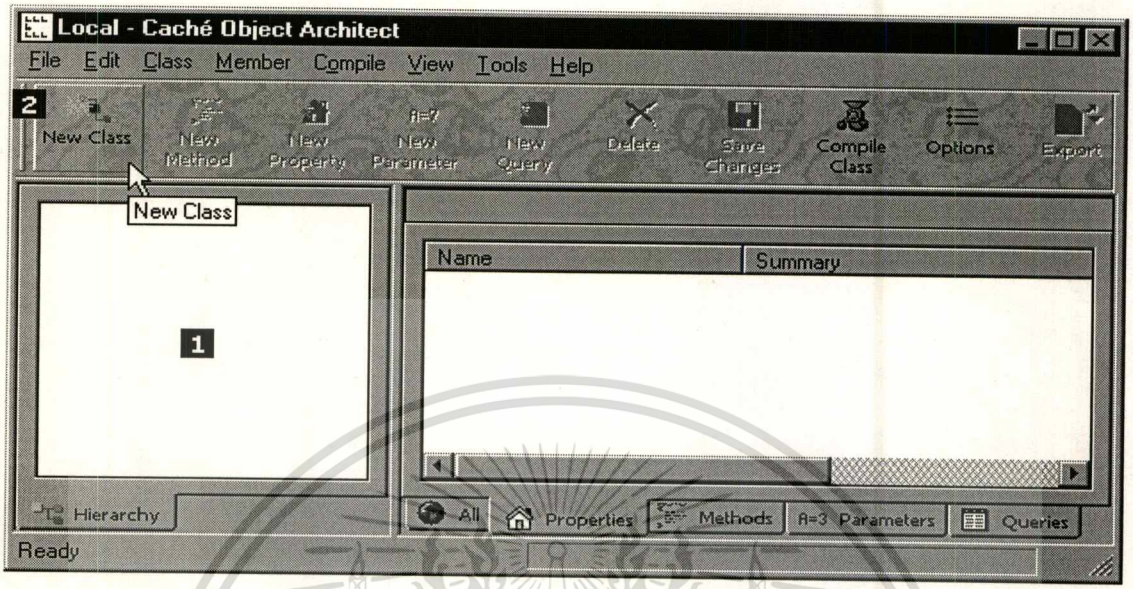


ภาพที่ 5.7 การเข้าสู่ Object Architecture

ขั้นที่ 3 Connect Archicture : หลังจากที่เราเข้าสู่ Object Architecture แล้วจะปรากฏหน้าจอดังภาพ เป็นการให้เราเลือกว่าจะทำการ Connect กับ Caché dictionary located ที่ใดซึ่งเราสามารถกำหนดเองได้โดยการระบุ IP Address ของเครื่องที่เราต้องการจะ Connect ในที่นี้เราเลือก architecture Connect ชื่อ SAMPLES ซึ่งมี Location อยู่ที่เครื่องของเราเอง(IP Address : 127.0.0.1)

ภาพที่ 5.8 Architecture Connect

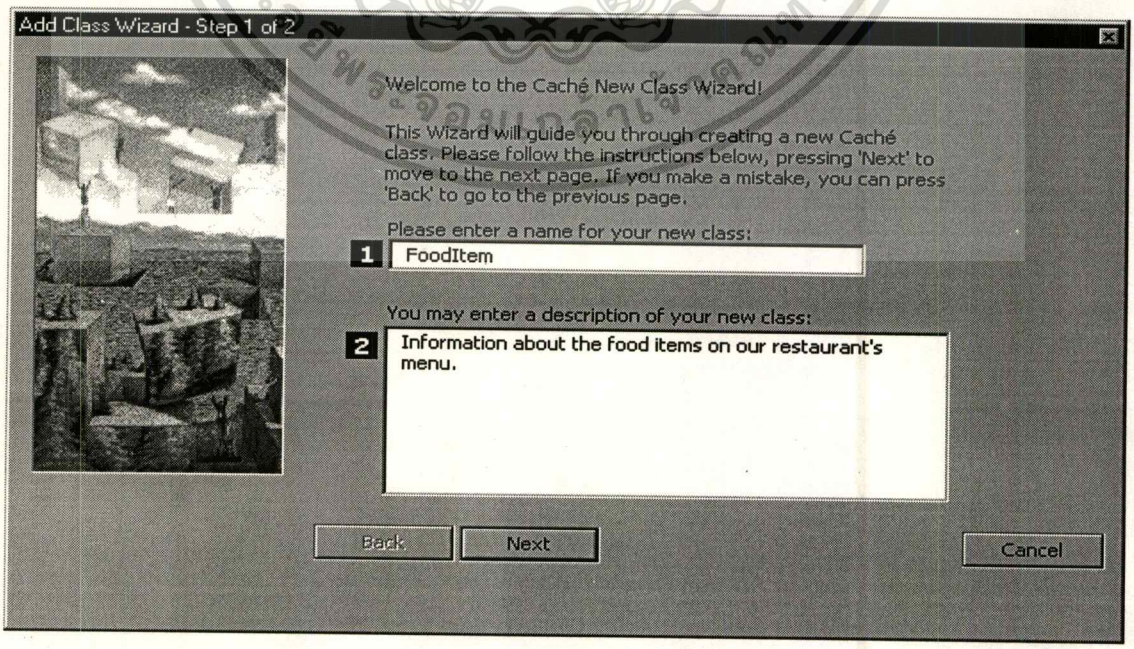
ขั้นที่ 4 การสร้าง Class : หลังจากที่เรา Connect เข้าสู่ Object Architecture แล้วจะปรากฏหน้าจอดังภาพ



ภาพที่ 5.9 Object Architecture

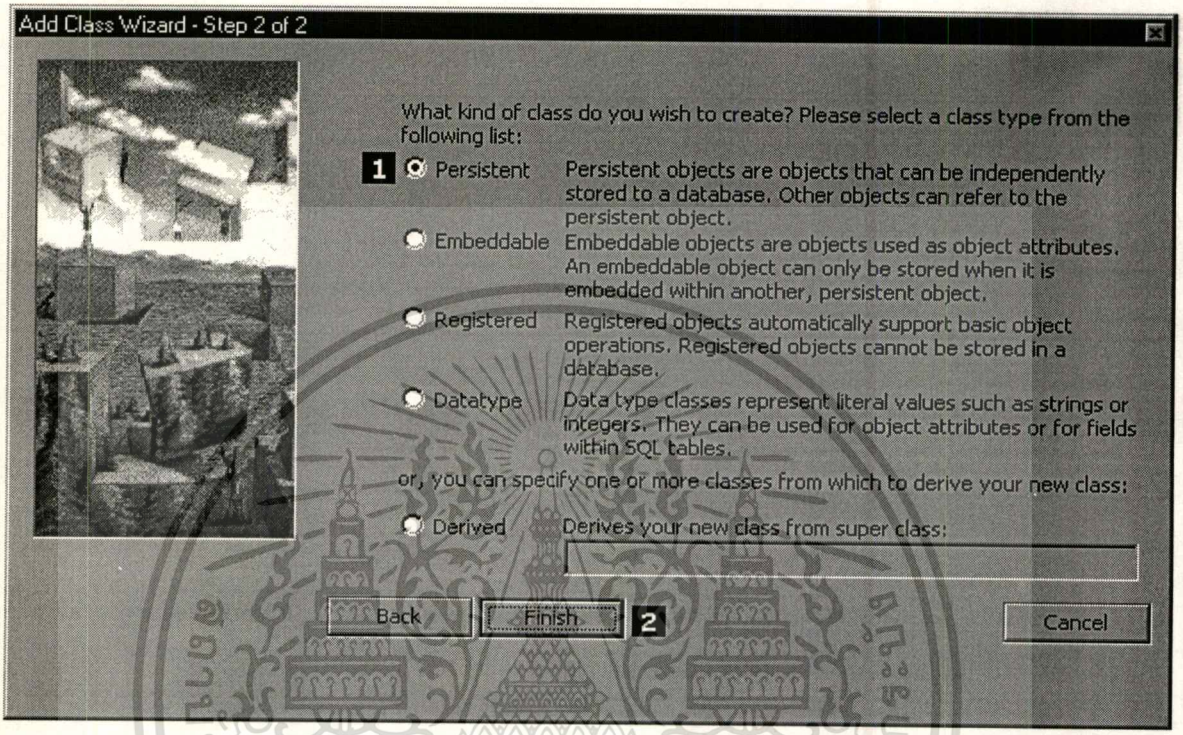
จากภาพเราต้องการที่จะสร้าง Class ใหม่ ทำได้โดยการ Click ที่หมายเลข 2 จากนั้นก็จะเข้าสู่การสร้าง Class โดย CACHE จะมี Wizard เป็นตัวช่วยอำนวยความสะดวกในการสร้างทำให้เข้าใจได้ง่าย โดยจะมีขั้นตอนย่อยดังนี้

1.1 กำหนดชื่อ Class



ภาพที่ 5.10 การกำหนดชื่อ Class และ Describtion

1.2 กำหนดชนิดของ Class

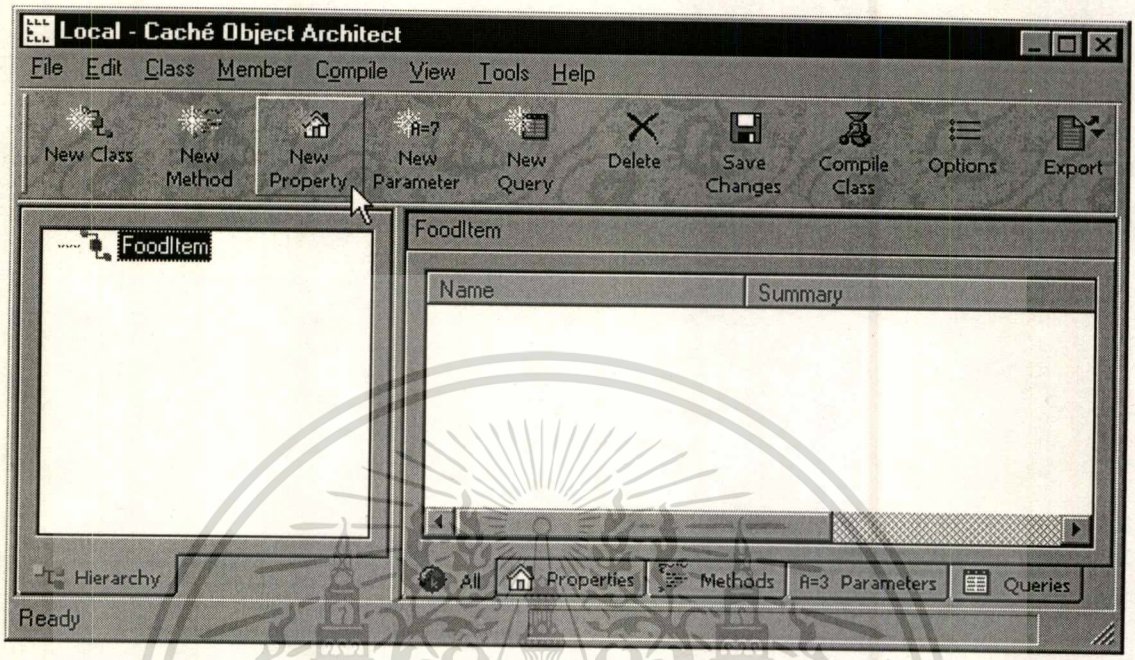


ภาพที่ 5.11 กำหนดชนิดของ Class

การกำหนดชนิดของ Class แบ่งได้เป็น 4 ประเภท ดังนี้

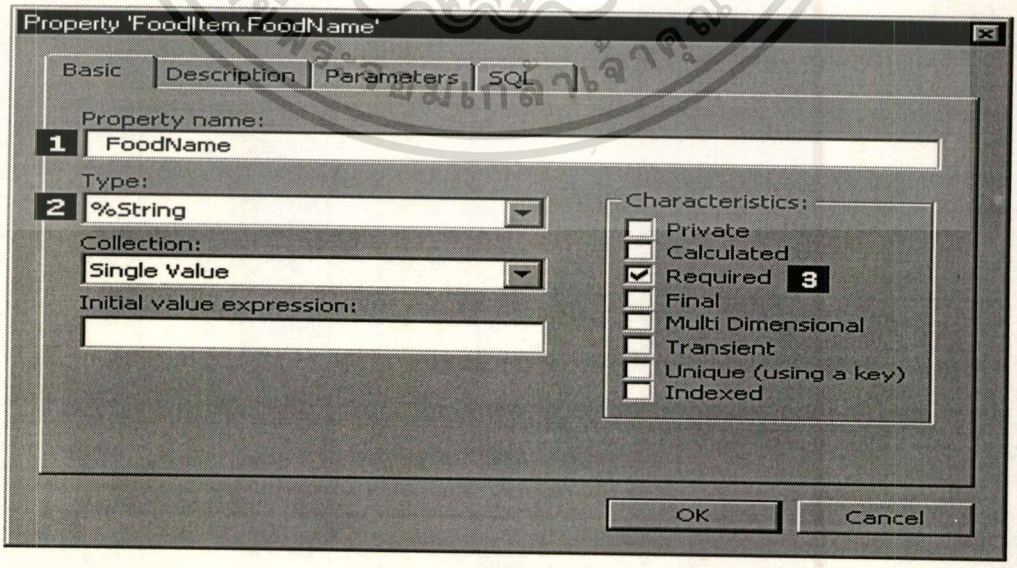
1. Persistent – เป็น Object ที่เป็นอิสระ ไม่ขึ้นกับ Object ใด ๆ
2. Embeddable – เป็น Object ที่ไม่สามารถอยู่อย่างโดดเดี่ยวได้ คือ ด้วยประกอบอยู่ใน Object เท่านั้น
3. Registered – เป็น Object ที่ไม่สามารถเก็บในฐานข้อมูลได้ เป็น Object ที่ใช้ในการ Support การทำงานขั้นพื้นฐานของ Object อื่น ๆ
4. Datatype – จะอยู่ในรูปแบบของ Literal Value
5. Derived – เป็น Object ที่ Inherit มาจาก SuperClass

ในตัวอย่าง เป็นการสร้าง Class ชื่อ FoodItem โดยเป็น Persistent เมื่อทุกอย่างเสร็จแล้วจะได้ผลดังภาพ จากนั้นจะเป็นการสร้าง Property เป็นขั้นตอนต่อไป



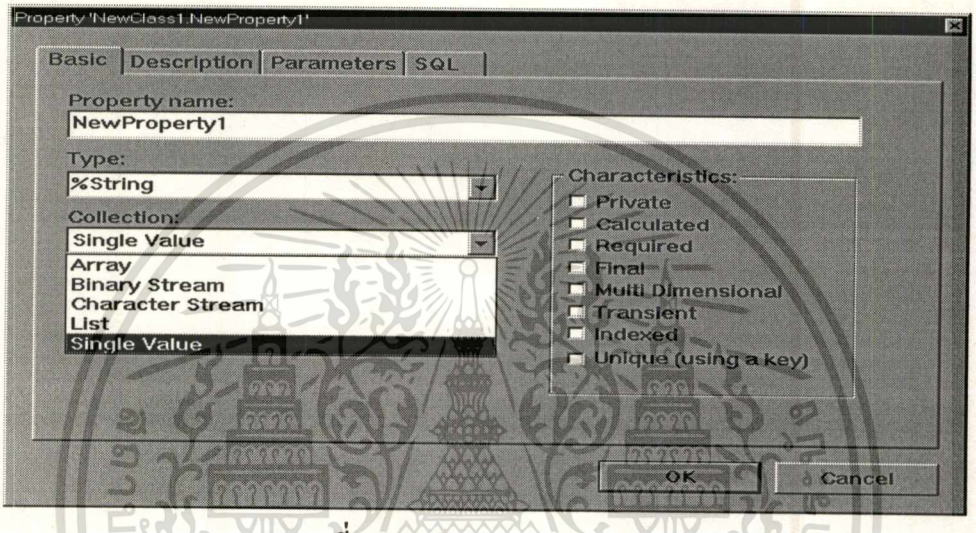
ภาพที่ 5.12 Class ที่สร้างเสร็จแล้ว

ขั้นที่ 5 การสร้าง Property : หลังจากที่เราได้สร้าง Class แล้ว ขั้นต่อไปคือการกำหนด Property ให้กับ Class ที่เราสร้าง โดยจากภาพข้างบนเมื่อ Click ที่ New Property แล้วจะเข้าสู่ Wizard การสร้าง Property ดังภาพ



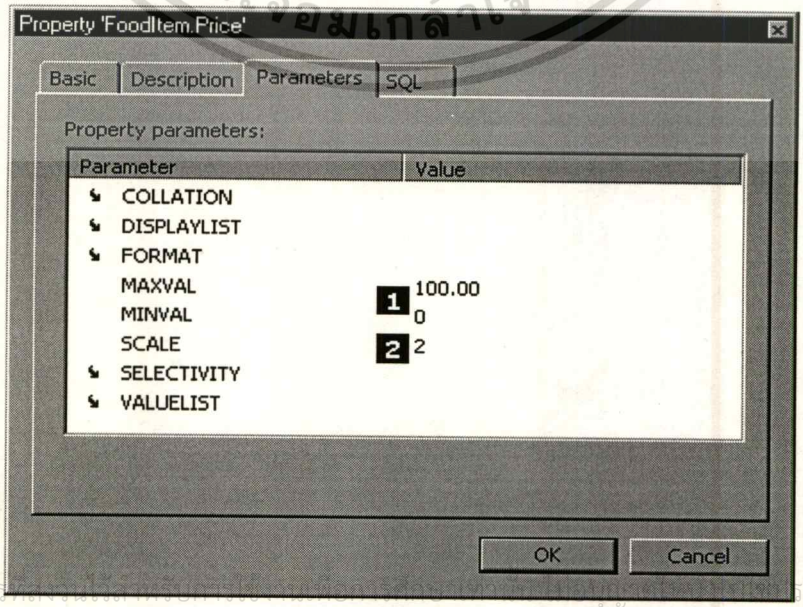
ภาพที่ 5.13 การสร้าง Property

จากภาพที่ 5.8 เป็นการสร้าง Property โดยกำหนด Property Name และกำหนดชนิดของ Property นั้น โดย CACHE มีชนิดของ Property ซึ่งสามารถรองรับการทำงานได้แทบทุกประเภท นอกจากนี้ยังมีส่วนของ Collection ซึ่งเราสามารถเลือกได้ว่า Property นั้นๆ จะมีค่าเป็นแบบใดดังภาพ



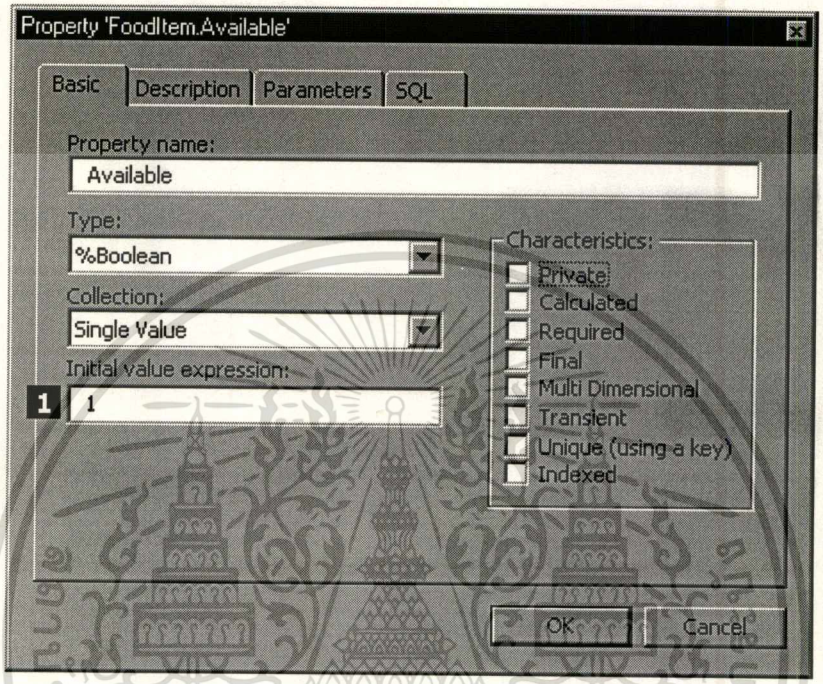
ภาพที่ 5.14 ชนิดของ Collection

นอกจากนี้เรายังสามารถกำหนดลักษณะของ Characteristic ได้อีกด้วย และเมื่อเราเลือกไปที่ Tab ชื่อ Parameter เราสามารถกำหนด ค่าต่าง ๆ ได้ เช่นค่ามากที่สุด(MAXVAL) ค่าน้อยที่สุด (MINVAL) ดังภาพ



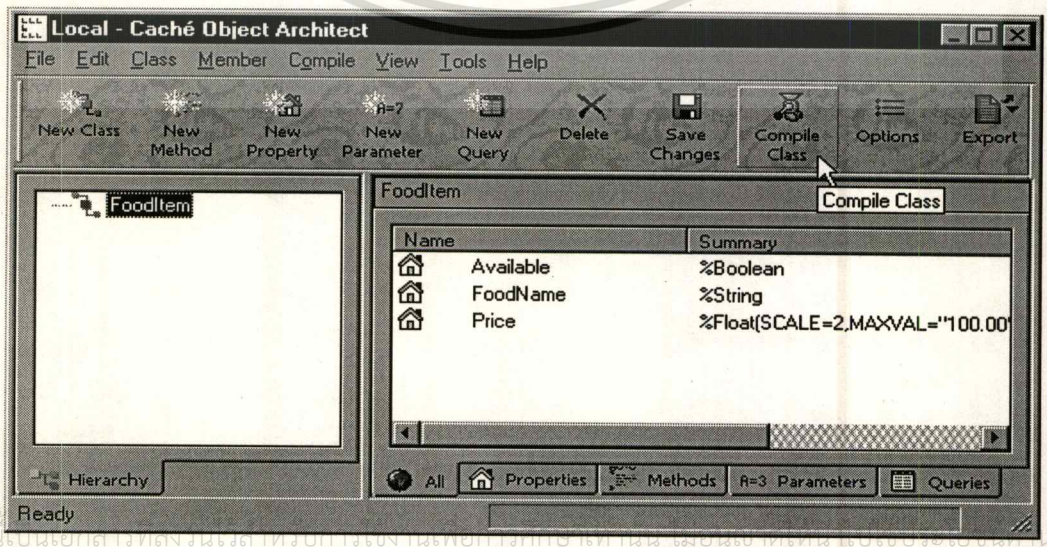
ภาพที่ 5.15 การกำหนด Parameter

จากภาพเป็นการกำหนดค่ามากที่สุด ค่าต่ำที่สุด และค่า Scale นอกจากนี้เรายังสามารถที่จะกำหนดค่าเริ่มต้น(Initial Value) ได้โดยทำได้ดังภาพ ดังนี้



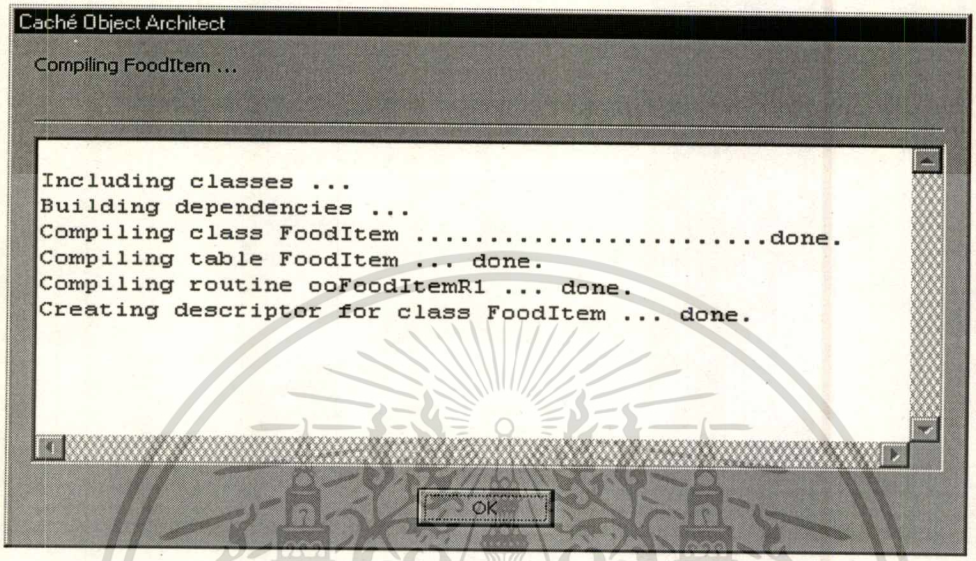
ภาพที่ 5.16 การกำหนดค่าเริ่มต้น

ขั้นที่ 6 การ Compile Class : หลังจากที่เราได้สร้างและกำหนดค่าต่าง ๆ ของ Property ของ Class ทั้งหมดตามทที่เราได้ออกแบบไว้แล้ว เราจะต้องทำการ Compile Class นั้น ๆ เพื่อที่จะนำไปใช้งาน ดังภาพ



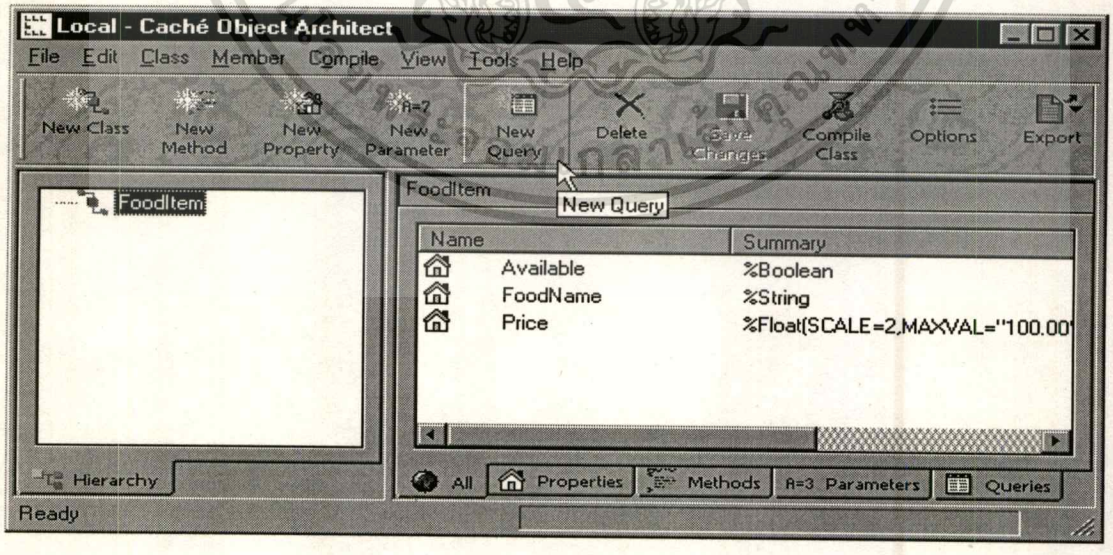
ภาพที่ 5.17 Class FoodItem ที่สร้างเสร็จแล้ว

เมื่อเราได้ Class ตามที่ต้องการแล้ว ต้องทำการ Compile โดยนำ Mouse ไป Click ที่ "Compile Class" ดังภาพข้างบน จากนั้นจะได้ผลดังนี้



ภาพที่ 5.18 Compile Class

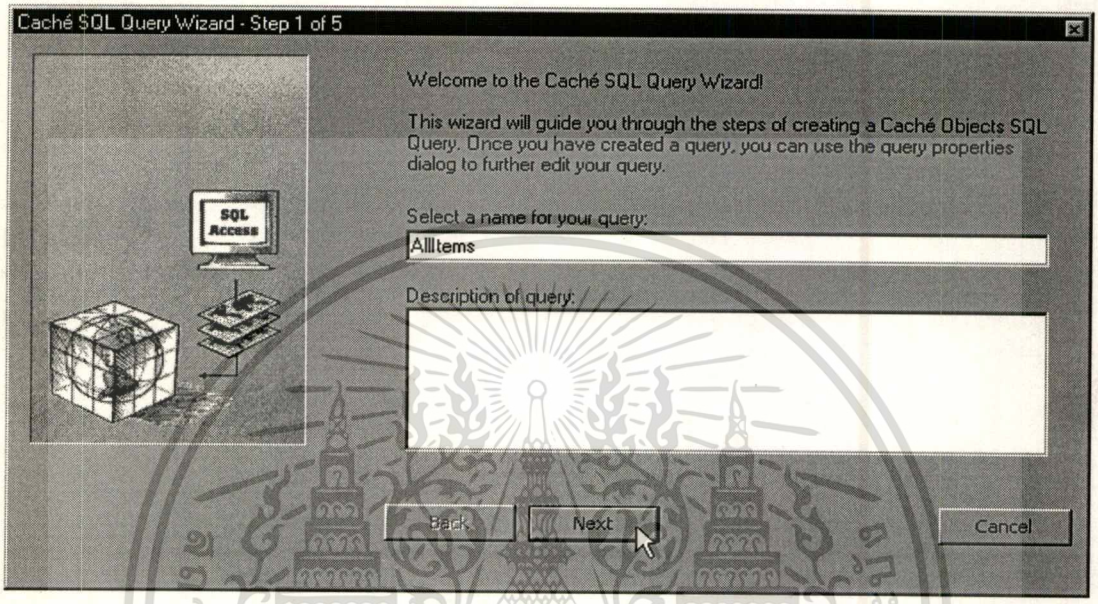
ขั้นที่ 7 การสร้าง Query เบื้องต้น : หลังจากที่เราได้สร้าง Class แล้ว เราสามารถที่จะใช้ CACHE ในการสร้าง Query ได้ โดยนำ Mouse ไป Click ที่ "New Query" ดังภาพ



ภาพที่ 5.19 การสร้าง Query

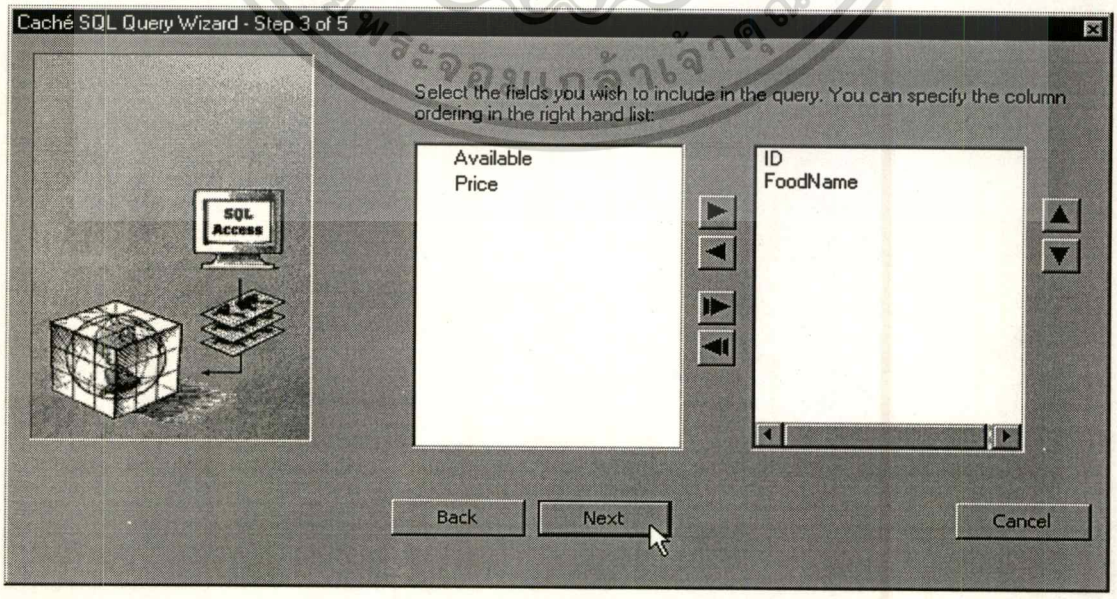
CACHE จะมี Wizard ในการสร้าง Query โดยจะมีขั้นตอนดังนี้

ขั้นที่ 7.1 : กำหนดชื่อและ Description



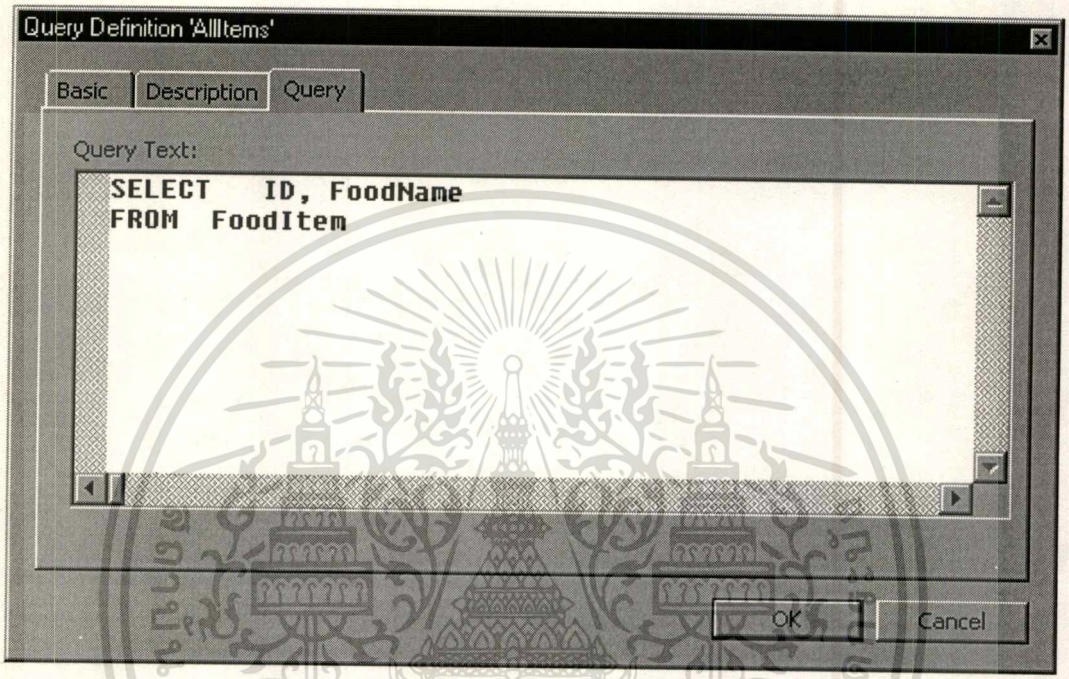
ภาพที่ 5.20 ระบุชื่อ และ Description New Query

ขั้นที่ 7.2 เลือก Property ที่จะใช้ในการทำ Query



ภาพที่ 5.21 เลือก Property ในการสร้าง Query

ขั้นที่ 7.3 : เมื่อเราทำการเลือก Property เสร็จแล้ว สามารถที่จะดู SQL Statement ที่เราสร้างไว้ได้ดังภาพ



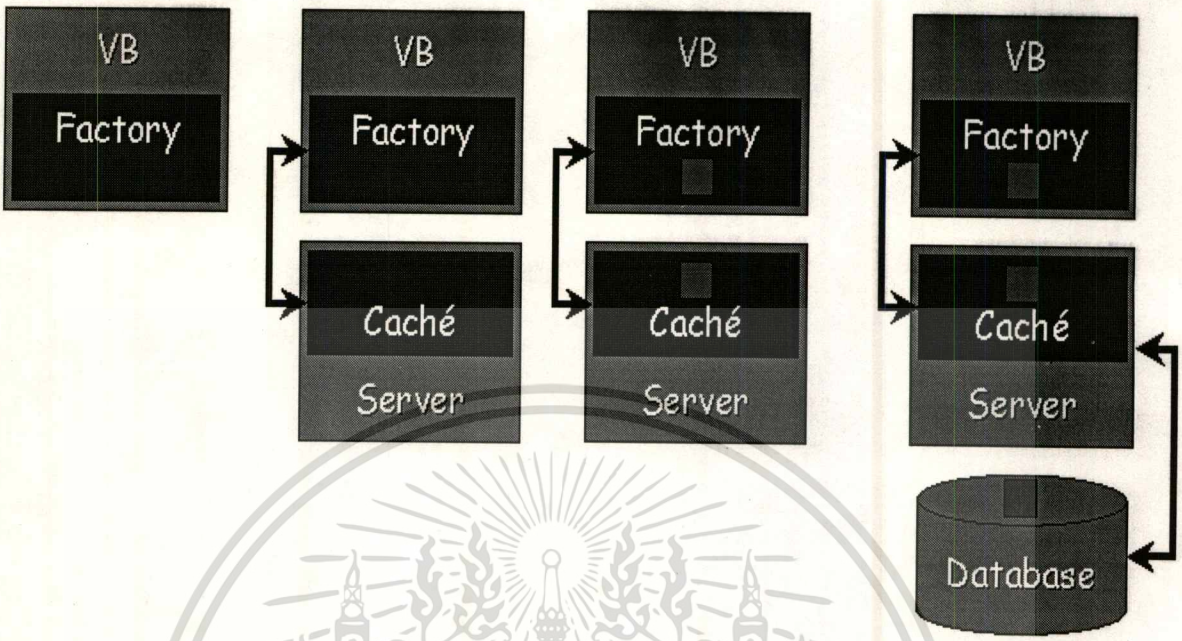
ภาพที่ 5.22 SQL ที่ได้จาก Wizard

เมื่อสร้าง Query เสร็จแล้ว ถ้าต้องการที่จะใช้งาน Query ที่สร้างนั้น เราต้องทำการ Compile Class ใหม่อีกครั้งตามขั้นตอนที่ได้กล่าวมาแล้วข้างต้น

5.5 การสร้าง User Interface โดยใช้ Visual Basic

ในการที่เราจะใช้งาน Class ที่เราสร้างไว้แล้วนั้น มีเครื่องมือต่าง ๆ มากมายในการที่จะสร้าง User Interface ในงานวิจัยนี้ขอนำเสนอการสร้าง User Interface โดยใช้ Visual Basic เนื่องจากเป็นภาษาที่พัฒนาได้ง่าย โดยมีหลักการดังนี้

ในการติดต่อกับ VB Application จะใช้ CACHE FACTORY เป็นตัวช่วยในการติดต่อกับ CACHE Server(ซึ่งในที่นี้ใช้ในเครื่องเดียวกัน) โดยที่ Factory จะทำการสร้าง CACHE ในลักษณะของ Object ให้ VB รู้จัก ดังภาพ

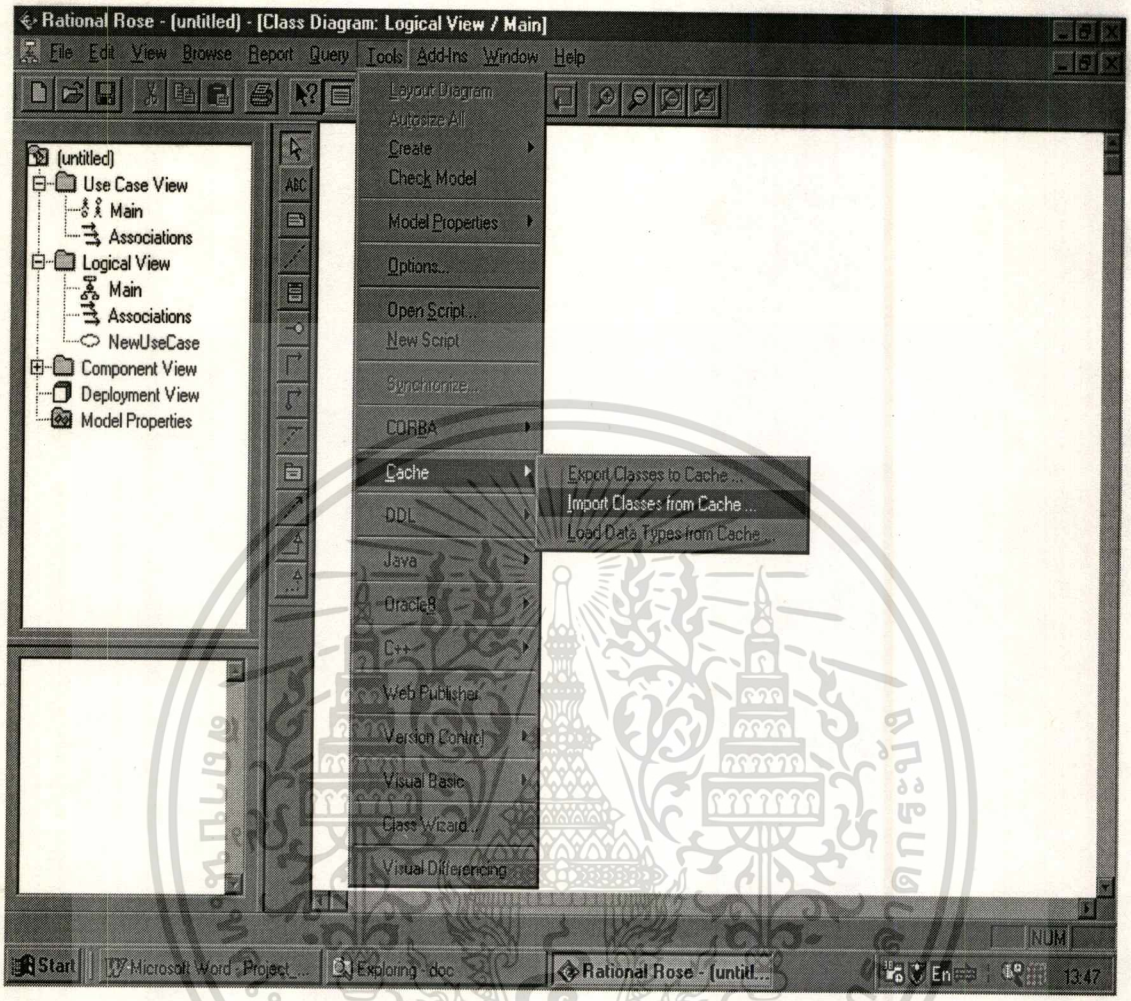


ภาพที่ 5.23 การทำงานของ CACHE FACTORY

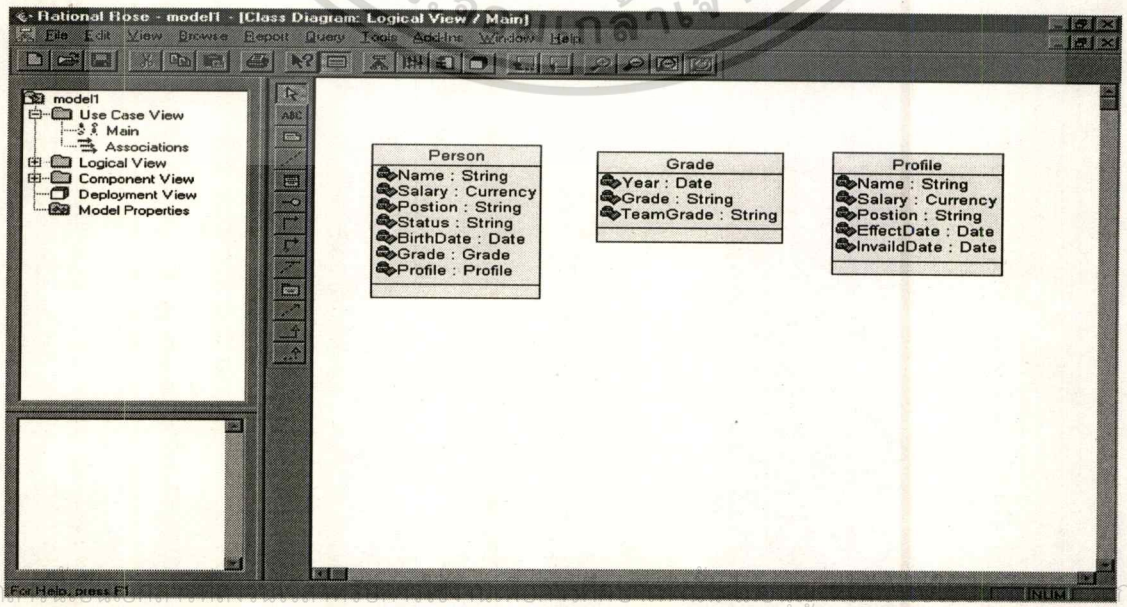
โดยที่ในตอนแรกใน VB จะทำการ Declare FactoryObject ก่อน จากนั้นจะทำการระบุว่าให้ FactoryObject นั้น Connect ไปที่ Location ใด Server ใด เมื่อสามารถทำการเชื่อมต่อกันได้แล้ว เมื่อมีการ สร้าง ลบ หรือมีการแก้ไขเปลี่ยนแปลงข้อมูลที่ส่วนของ VB ทาง Factory Object จะทำการติดต่อกับ CACHE Server ซึ่งตัว CACHE Server จะทำการ Update ฐานข้อมูลให้ดังภาพข้างต้น ซึ่งตัวอย่างการ Statement ของ VB ในการใช้ Factory จะมีรายละเอียดในภาคผนวก

5.6 การออกแบบเชิงวัตถุโดยใช้งานร่วมกับ Rational Rose

ในงานวิจัยนี้ได้เลือกใช้เครื่องมือที่ช่วยในการออกแบบเชิงวัตถุคือ Rational Rose ของบริษัท Rational Software ซึ่งเราสามารถที่จะใช้งานร่วมกับ CACHE และ Application อื่น ๆ ได้มากมายได้ ในส่วนนี้จะสาธิตการใช้งานของ Rational Rose พอสังเขปเพื่อนำไปใช้ในการออกแบบ Class และนำ Class ที่ได้ไปใช้งานกับ CACHE ได้เลย โดยทำงานร่วมกับ CACHE มี 3 ส่วน ดังภาพ

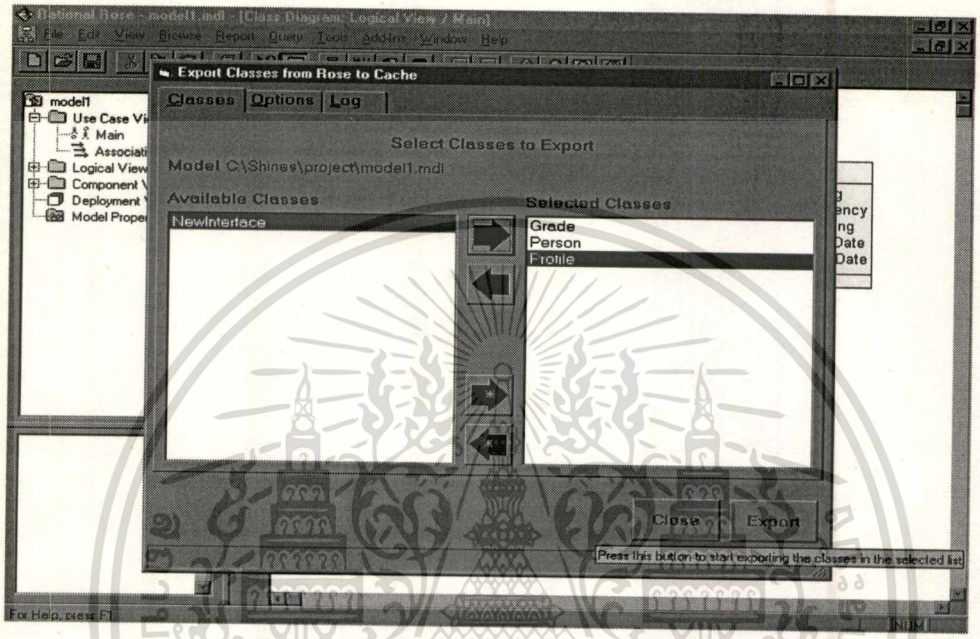


ภาพที่ 5.24 การทำงานร่วมกับ CACHE



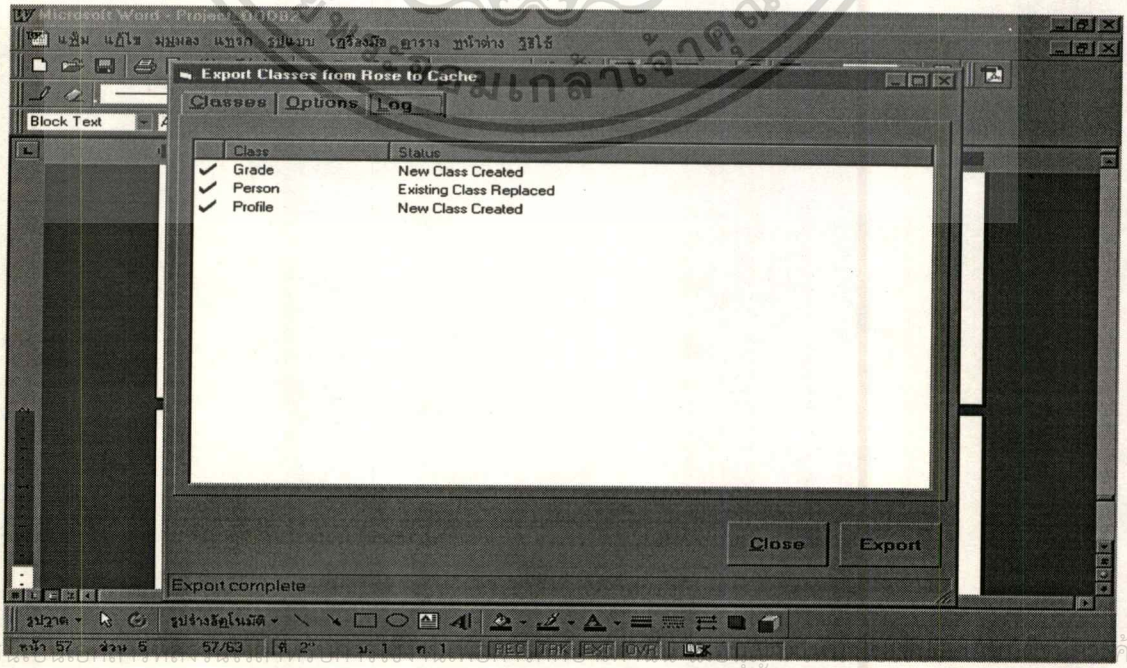
ภาพที่ 5.25 ตัวอย่างการออกแบบ Class โดย Ration Rose

ส่วนที่ 1 : Export Class to CACHE – เป็นการนำ Class ที่ออกแบบเสร็จเรียบร้อยแล้วจาก Rational Rose ไป Compile ใน CACHE เพื่อใช้งานจริงได้ โดยเลือก Tools → CACHE → Export Class to CACHE จะปรากฏหน้าจอดังนี้



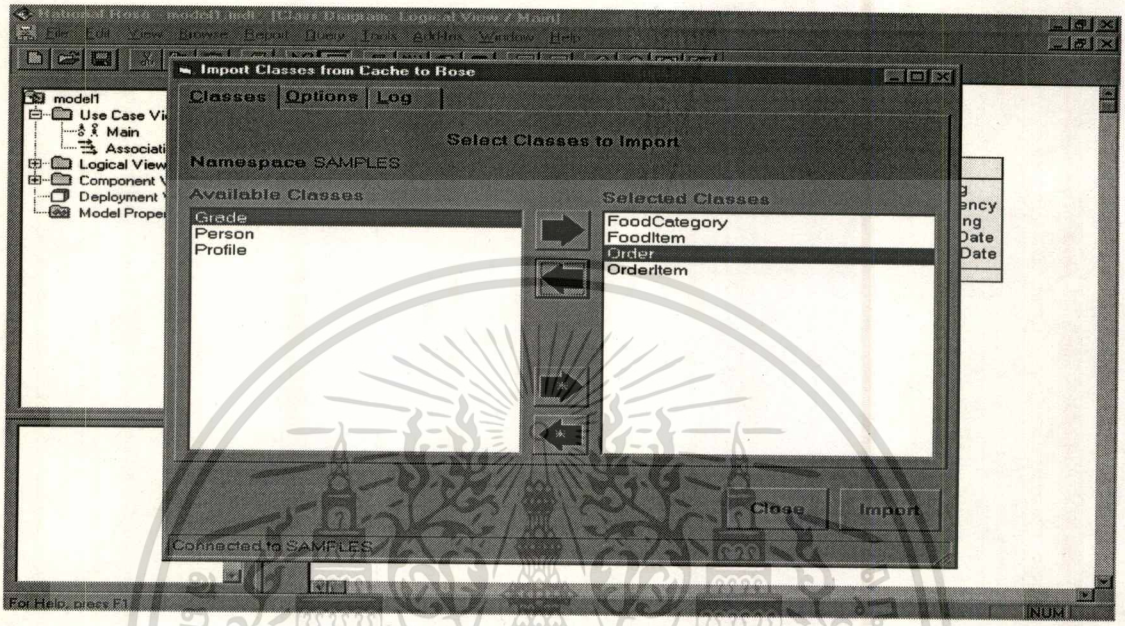
ภาพที่ 5.26 แสดงการเลือกว่าเราจะ Export Class ใด

จากนั้นจะทำการ Generate Class ที่เราเลือก เมื่อเสร็จแล้วจะปรากฏดังภาพ



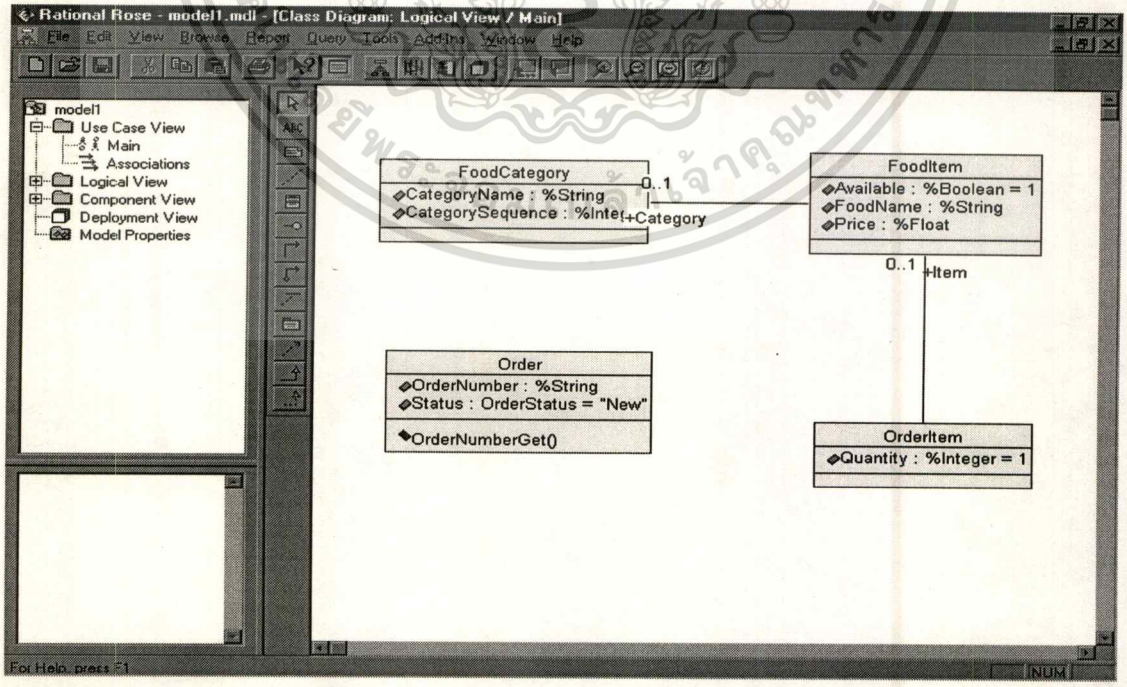
ภาพที่ 5.27 การ Export to CACHE เสร็จสมบูรณ์

ส่วนที่ 2 : Import From CACHE เป็นการนำ Class จาก CACHE มาแสดงในรูปแบบ Diagramเพื่อใช้ทำเป็นพิมพ์เขียวในการออกแบบระบบได้ ดังภาพ



ภาพที่ 5.28 แสดงการ Import From CACHE

จากภาพจะได้ Diagram ของ Class ที่เรา Import ดังภาพ



ภาพที่ 5.29 การ Import Complete

ในบทนี้ได้นำเสนอถึงเครื่องมือที่จะใช้ในการออกแบบและพัฒนาระบบเชิงวัตถุแต่เพียงสังเขป เพื่อที่จะได้ทราบถึงแนวทางในการสร้างระบบเชิงวัตถุซึ่งจะมีทั้งแนวทางและวิธีการในการพัฒนาที่ต่างจากการพัฒนาระบบในแบบเดิม ๆ

บทที่ 6
กรณีศึกษาการนำ OODB มาแก้ปัญหาการเปลี่ยนแปลงตัวระบุชี้

จากบทที่แล้วเราได้รู้จัก OODBMS กันพอสังเขป ในส่วนของบทที่ 6 นี้ จะยกตัวอย่างกรณีศึกษาของระบบสารสนเทศในสถาบันการศึกษาแห่งหนึ่งที่มีปัญหาในเรื่องของตัวระบุชี้ โดยจะนำเสนอการนำระบบฐานข้อมูลเชิงวัตถุในการนำไปประยุกต์ใช้แก้ไขปัญหาที่เกี่ยวกับตัวระบุชี้ ซึ่งได้กล่าวมาในบทที่ 4 แล้วว่าปัญหาที่จะเกิดมี 3 แบบ คือ ปัญหาการเปลี่ยนแปลงตัวระบุชี้(Modifying Identifier Key) ปัญหารูปแบบตัวระบุชี้ไม่เหมาะสม(Nonuniformity) และปัญหาการ Join ซึ่งในกรณีศึกษานี้จะขอยกตัวอย่างการทำงานของระบบที่ประกอบด้วยปัญหาทั้ง 3 แบบอยู่ด้วยกัน

6.1 กรณีศึกษาปัญหาเกี่ยวกับตัวระบุชี้ในระบบงาน

กรณีศึกษาที่จะยกมานี้เป็นระบบทะเบียนของสถาบันการศึกษาแห่งหนึ่งทำหน้าที่เก็บประวัติของนักศึกษารวมทั้งประวัติการศึกษาของนักศึกษา รวมไปถึงระบบตรวจสอบผลการเรียนด้วย ซึ่งทั้งหมดจะรองรับในกรณีที่นักศึกษามีการเปลี่ยนแปลงข้อมูลต่าง ๆ เช่น เปลี่ยนชื่อ-นามสกุล ที่อยู่ เปลี่ยนรหัสนักศึกษา(ในกรณีที่เปลี่ยนคณะ หรือสอบเข้าใหม่) และอื่น ๆ อีกทั้งในระบบตรวจสอบผลการเรียนจะทำการคำนวณว่านักศึกษาจะต้องลงทะเบียนอีกเท่าไรในรายวิชาใดจึงจะสำเร็จการศึกษา โดยจะทำการตรวจสอบจากหลักสูตรที่กำหนดมา ซึ่งสามารถแจกแจงปัญหาตามประเภทได้ดังนี้

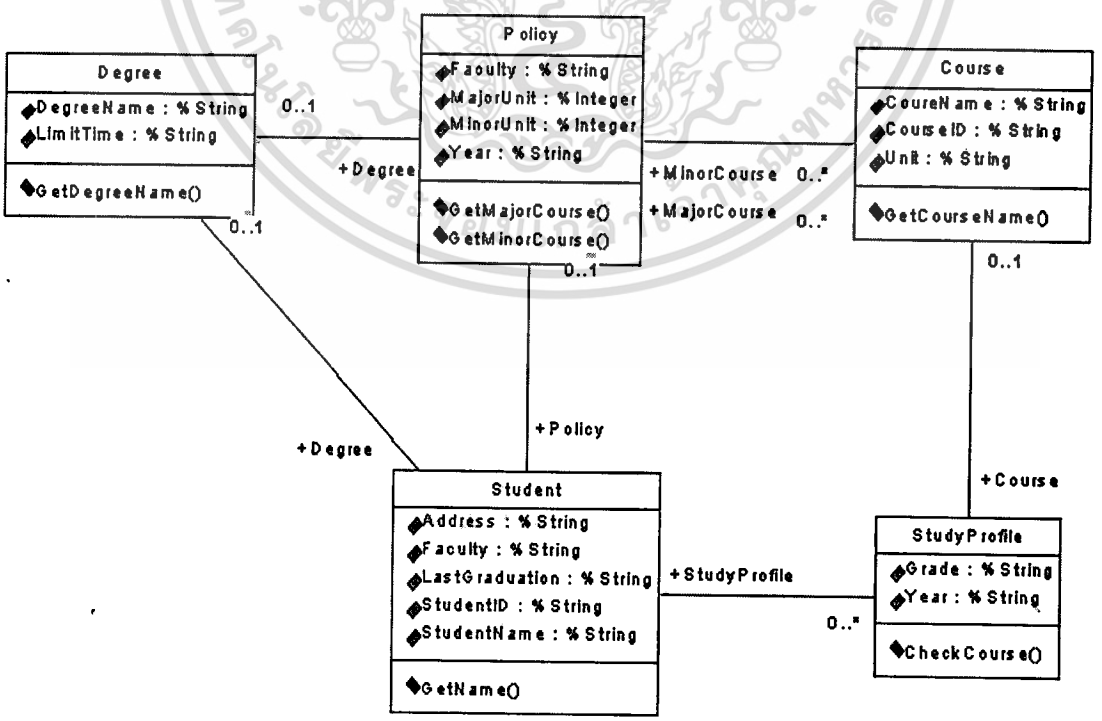
1. ปัญหาการเปลี่ยนแปลงแก้ไขตัวระบุชี้(Modifying Identifier Key) จะเห็นได้ว่าในกรณีที่นักศึกษามีการเปลี่ยนแปลงรหัสนักศึกษาหรือชื่อ-นามกุลก็แล้วแต่ ถ้าเป็นในระบบฐานข้อมูลเชิงสัมพันธ์ หากว่าค่าใด ๆ ก็ตามใช้เป็นตัวระบุชี้ เมื่อมีการเปลี่ยนแปลงค่านั้น ๆ จะทำให้ค่าอื่น ๆ ได้รับผลกระทบไปด้วย ตัวอย่างเช่น กรณีที่ใช้รหัสนักศึกษาเป็นตัวระบุชี้ เมื่อนักศึกษาคนหนึ่งเรียนในคณะวิทยาศาสตร์ได้ 2 ปี แล้วโดน Retire ปีต่อมาเขาเข้าสอบใหม่อีกครั้งและติดในคณะเดิม แต่เปลี่ยนรหัสนักศึกษาไป ถ้าใน Relation อื่น ๆ ที่ใช้รหัสนักศึกษาในการอ้างถึงต้องทำการ Update ข้อมูลรหัสนักศึกษานี้ด้วย ซึ่งจะต้องมีการตรวจสอบข้อมูลก่อนที่จะทำการ Update ที่เรียกว่า Referential Integrity Rule ดังที่เคยกล่าวไว้ในบทที่ 4

2. ปัญหารูปแบบของตัวระบุชี้ไม่เหมาะสม(Noununiformity) ในระบบฐานข้อมูลเชิงสัมพันธ์ยกตัวอย่างในกรณีที่นักศึกษาคนหนึ่งทำการเปลี่ยนแปลงคณะ ระบบจะทำการเก็บประวัติการเปลี่ยนแปลงนี้ไว้ในระบบโดยจะต้องมีเรื่องของเวลามาเกี่ยวข้องคือ ต้องมีการเก็บบันทึกว่านักศึกษาคนนั้นอยู่ในคณะเดิมจนถึงวันที่เท่าไร ในกรณีที่เราต้องการจะดูว่า นักศึกษาคนนี้ตอนเรียนอยู่ที่คณะเดิมเขามีผลการเรียนเป็นอย่างไร จะต้องใช้การอ้างถึงโดยมีแกนเวลามาเกี่ยวข้อง คือนอกจากจะใช้ตัวระบุชี้ในการอ้างถึงข้อมูลอย่างเดียวยังต้องใช้เวลามาอ้างถึงด้วย ซึ่งในระบบฐานข้อมูลเชิงวัตถุในการอ้างถึง Object อื่น ๆ เราสามารถใช้ OID อ้างถึงได้เลย

3. ปัญหาในการ Join เช่นถ้าเราต้องการจะทราบว่านักศึกษาคนนี้จะต้องเรียนอีกกี่วิชาจึงจะสำเร็จการศึกษา หรือจะต้องลงในรายวิชาใดจึงจะสำเร็จการศึกษา เป็นต้น ในระบบฐานข้อมูลเชิงสัมพันธ์ ถ้าเราต้องการที่จะดูข้อมูลที่มีการอ้างถึงกันมากกว่า 1 ตารางจะใช้วิธีการ Join ซึ่งเป็นการทำงานที่ซับซ้อนมากถ้ามีการ Join กันหลาย ๆ ตาราง

6.2 การออกแบบโดยใช้หลักการออกแบบเชิงวัตถุ

จากที่ได้ทราบถึงปัญหาของระบบงานเราสามารถออกแบบระบบเชิงวัตถุโดยใช้หลักของ UML มาทำการสร้างเป็น Class Diagram ได้ดังนี้



ภาพที่ 6.1 แสดง Class Diagram ของกรณีศึกษา

จาก Class Diagram จะประกอบด้วย Class ต่าง ๆ 5 Class ดังภาพ ประกอบด้วย

Class Student – เป็น Class ของนักศึกษาที่มีการเก็บข้อมูลต่าง ๆ ได้แก่ ชื่อ คณะ วันเกิด ที่อยู่ รหัสนักศึกษา ประวัติการเปลี่ยนแปลงข้อมูลต่าง ๆ ของนักศึกษา รวมทั้งประวัติการศึกษา

Class StudyProfile – เป็น Class ที่เก็บข้อมูลประวัติการศึกษาของนักศึกษาในแต่ละปี

Class Policy – เป็น Class ที่เก็บข้อมูลหลักสูตรการศึกษา ที่ระบุว่าในแต่ละปีต้องเรียนวิชาใดบ้างจึงจะสำเร็จการศึกษา

Class Course – เป็น Class ที่เก็บข้อมูลวิชาประกอบด้วย ชื่อวิชา จำนวนหน่วยกิต และรหัสวิชา

Class Degree – เป็น Class ที่เก็บข้อมูลวิชาประกอบด้วย ระดับชั้นเช่นปริญญาตรี ปริญญาโท เป็นต้น รวมทั้ง ระยะเวลาในการศึกษา

6.3 การพัฒนาโปรแกรมเชิงวัตถุเพื่อแก้ไขปัญหา

หลังจากที่เราได้ทำการออกแบบ Class Diagram แล้ว ขั้นต่อไปเราจะนำ Class Diagram ที่ได้ไปสร้าง Class โดยใช้ CACHE ด้วยวิธีการที่กล่าวมาแล้วในบทที่ 5 โดยจะแบ่งการทำงานของการใช้ Class ดังนี้

1. การเพิ่มข้อมูลเข้าสู่ระบบ (Insertion) ในกรณีศึกษานี้มีการป้อนข้อมูลเข้าสู่ระบบ ดังนี้
- การป้อนข้อมูลนักศึกษา
- การป้อนข้อมูลหลักสูตร
- การป้อนข้อมูลวิชา

2. การแก้ไขข้อมูล(Update) เป็นการเปลี่ยนแปลงข้อมูลเดิมที่มีในระบบ ประกอบด้วย
- การแก้ไขข้อมูลนักศึกษา
- การแก้ไขข้อมูลหลักสูตร
- การแก้ไขข้อมูลวิชา

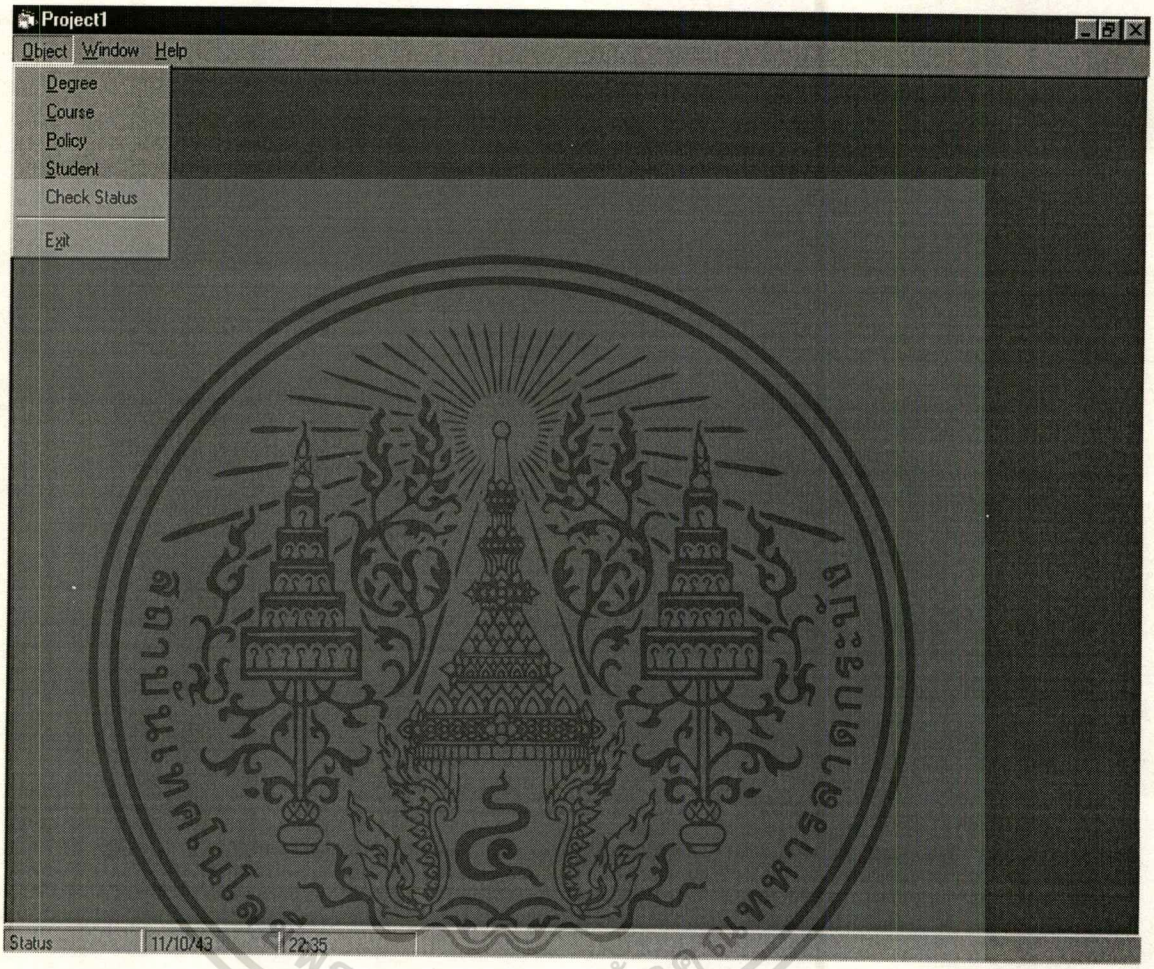
3. การตรวจสอบการสำเร็จการศึกษา โดยจะแสดงวิชาที่ยังไม่ได้เรียนหรือยังไม่ผ่านของนักศึกษา

4. การแสดงประวัติการศึกษาของนักศึกษาตั้งแต่เริ่มเข้าเรียนจนปัจจุบัน

ซึ่งสามารถพัฒนาเป็นโปรแกรมโดยใช้ Visual Basic Version 6 ได้โดยจะนำเสนอในลักษณะของ User Interface ดังนี้

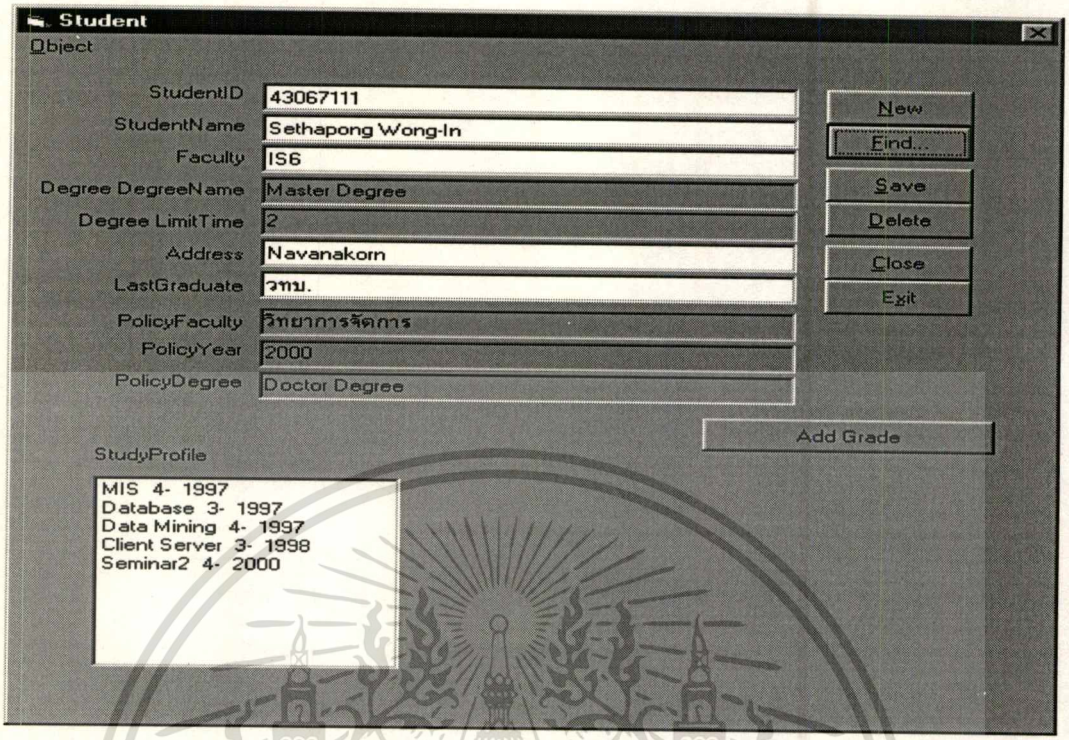
6.2.1. การเพิ่มข้อมูลเข้าสู่ระบบ (Insertion)

เมื่อเข้าสู่ระบบจะมีหน้าจอหลักดังนี้



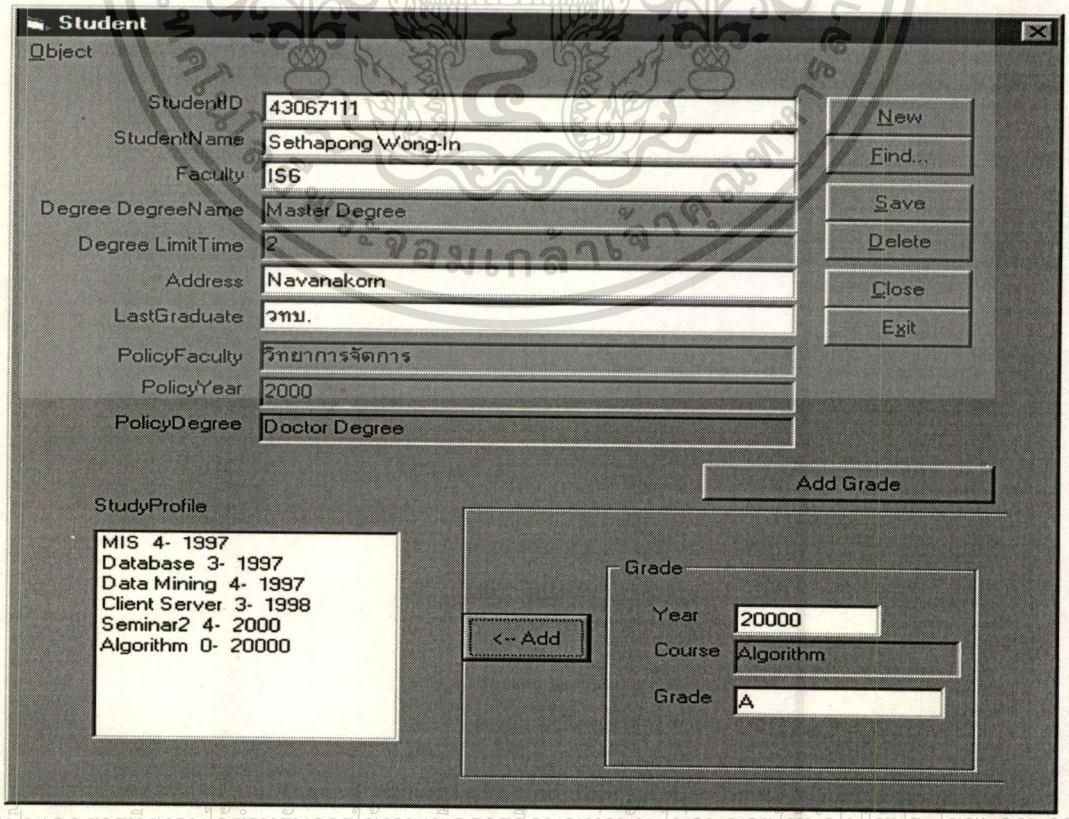
ภาพที่ 6.2 หน้าจอหลักของโปรแกรม CaseStudy

ที่ Menu bar ในโปรแกรมประกอบด้วย Object Windows และ Help ซึ่งใน Object จะประกอบด้วย Degree, Course, Policy, Student และ Check Status ในกรณีที่เราจะเพิ่มข้อมูลเข้าสู่ระบบ ซึ่งประกอบด้วย ข้อมูลนักศึกษา ข้อมูลหลักสูตรและข้อมูลวิชา ซึ่งสามารถแสดงดงหน้าจอต่าง ๆ ได้ดังนี้



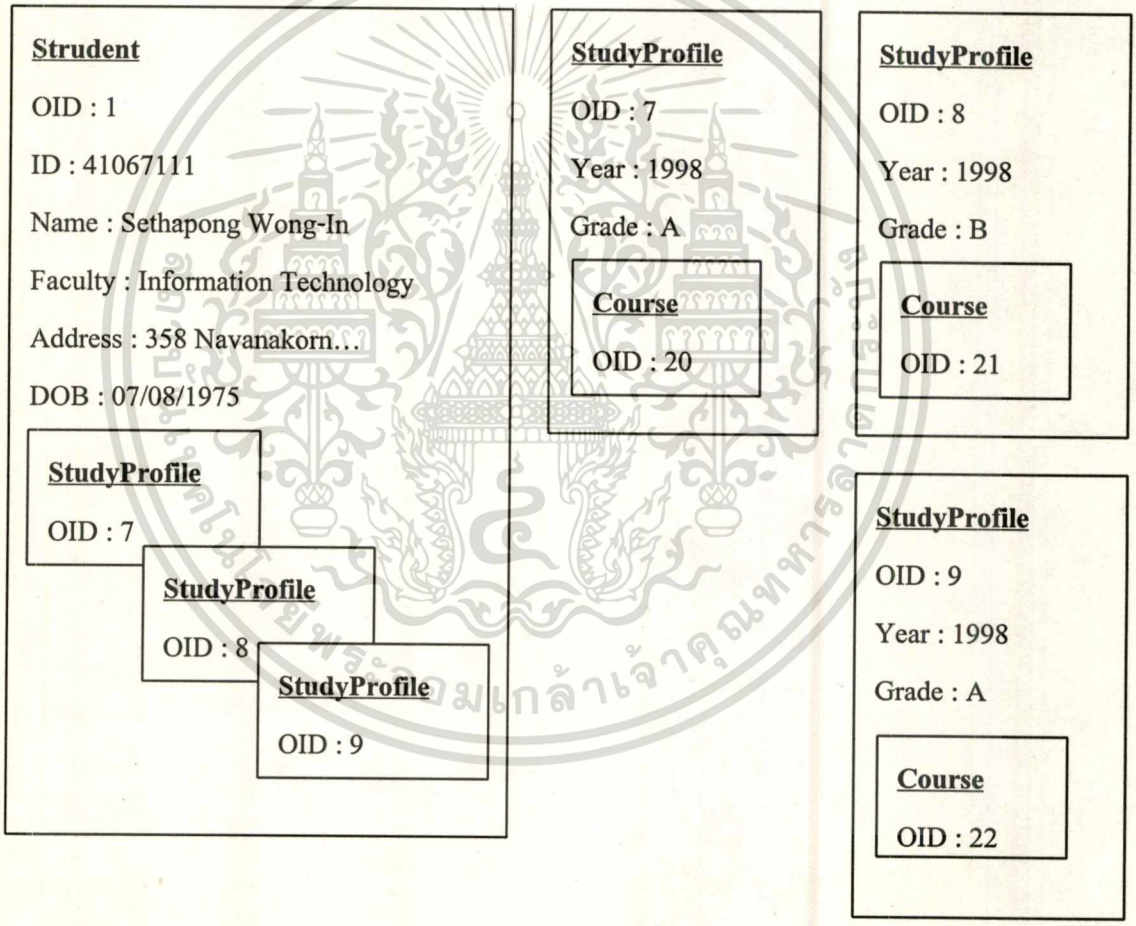
ภาพที่ 6.3 การป้อนข้อมูลนักศึกษา

ในกรณีที่ต้องการบันทึกผลการเรียนของนักศึกษาคนนี้ทำได้โดยกดปุ่ม “ Add Grade” จะปรากฏหน้าจอดังนี้



ภาพที่ 6. 4 การป้อนข้อมูลผลการศึกษาเข้าสู่ระบบ

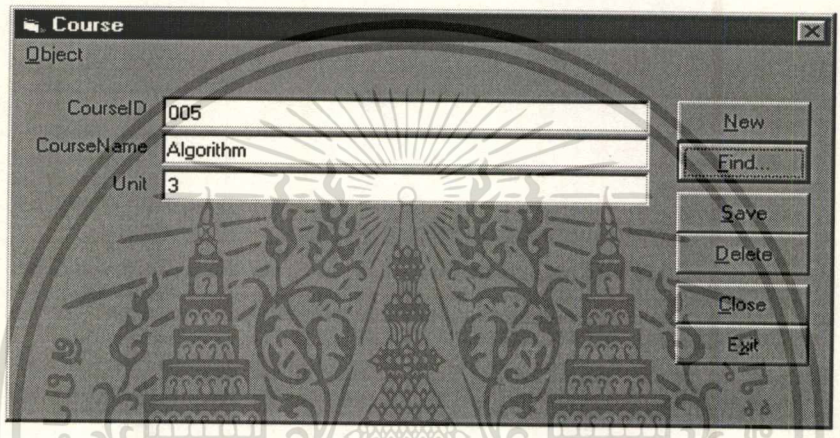
ในส่วนของการป้อนข้อมูลของนักศึกษาเข้าสู่ระบบเป็นการสร้าง Object Student ขึ้นมาใหม่โดยจะมี OID ที่ระบบทำการสร้างให้เป็นตัวระบุชี้ซึ่งระบบจะสร้าง OID ทุกครั้งที่มีการ ADD New Object และเมื่อเราทำการป้อนข้อมูลต่าง ๆ ของนักศึกษาเสร็จแล้วการจัดเก็บข้อมูลของ Object Student จะมีลักษณะการเก็บข้อมูลเป็น Object โดยมีลักษณะดังภาพที่ 6.5 และใน Object StudyProfile เมื่อเราทำการป้อนผลการศึกษาระบบจะทำการสร้าง OID ให้กับ Object StudyProfile เช่นเดียวกัน



ภาพที่ 6.5 ลักษณะการเก็บข้อมูลของ Student Class

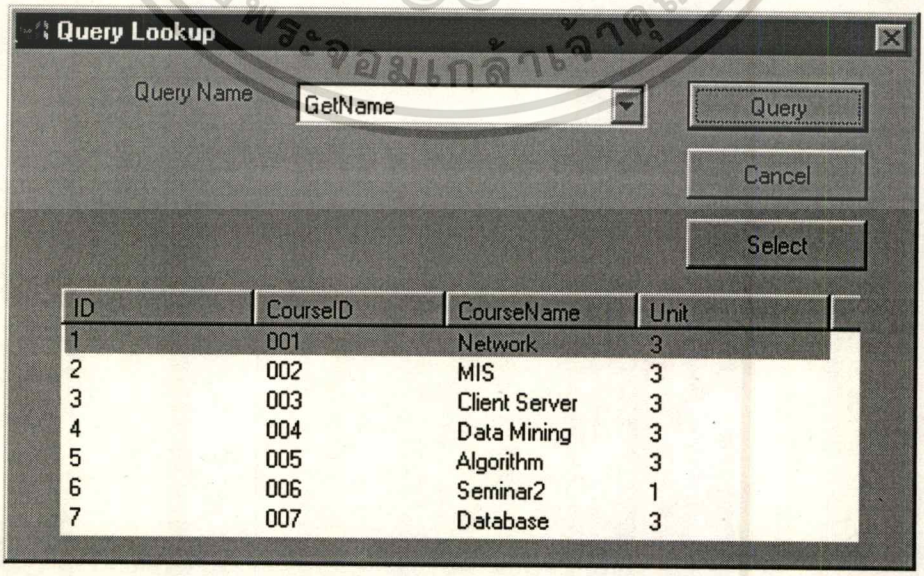
จากภาพข้างบนจะแสดงให้เห็นถึงลักษณะการเก็บข้อมูลของ Class Student และ Class StudyProfile ซึ่งจะเห็นว่าการที่จะอ้างถึง Object ใด ๆ จะใช้ OID เป็นตัวอ้างถึง เราสวามารถทำการแก้ไขเปลี่ยนแปลงข้อมูลได้ทุก ๆ Attribute อยู่ใน Object ได้ตามอิสระ ดังนั้นปัญหาการเปลี่ยนแปลงตัวระบุชี้จึงไม่เป็นปัญหาอีกต่อไป

ในส่วนของการนำข้อมูลของหลักสูตร(Policy Class)และข้อมูลวิชา(Course Class)ก็จะมีลักษณะการเก็บข้อมูลในรูปแบบเดียวกันโดยจะมีหน้าจอในการป้อนข้อมูลดังภาพที่ 6.6



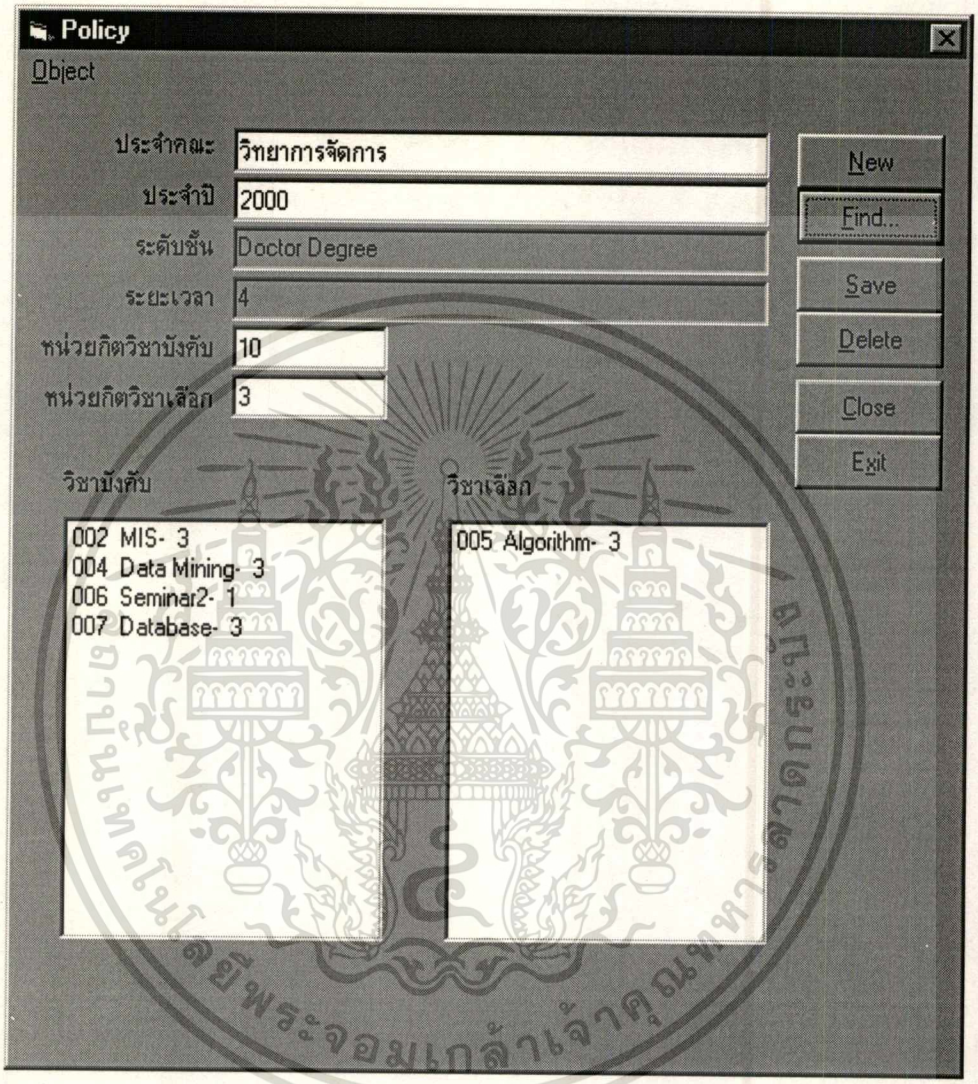
ภาพที่ 6.6 การป้อนข้อมูลวิชาเข้าสู่ระบบ

จากภาพที่ 6.6 เป็นหน้าจอสำหรับการป้อนข้อมูลและแก้ไขข้อมูลของเราสามารถที่จะ Query ข้อมูลขึ้นมาดูและแก้ไขได้ โดยกดปุ่ม "Find" จะปรากฏหน้าจอดังนี้



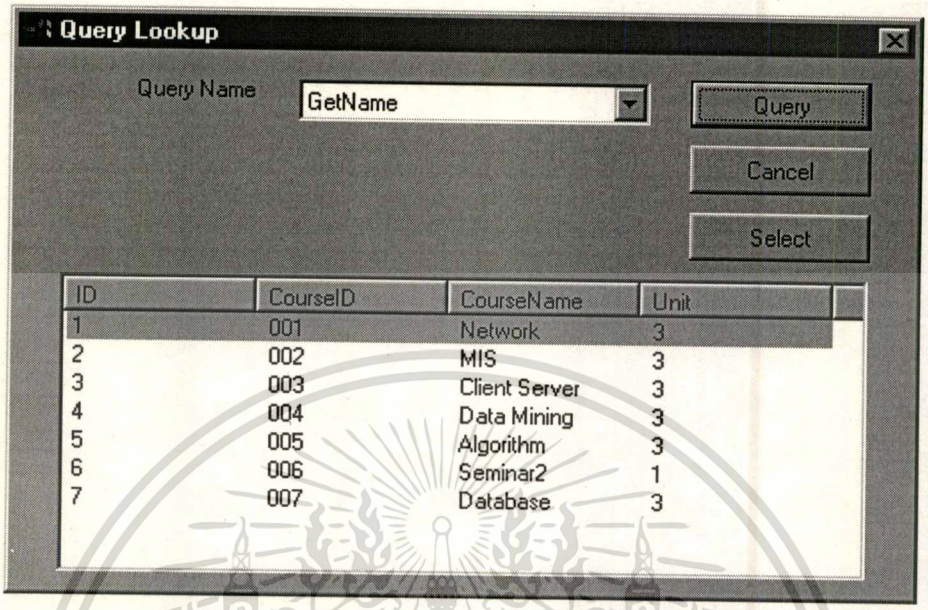
ภาพที่ 6.7 การเรียกดูรายการรายวิชา

ในส่วนของหลักสูตรวิชาจะมีหน้าจอการป้อนข้อมูลและแก้ไขดังนี้



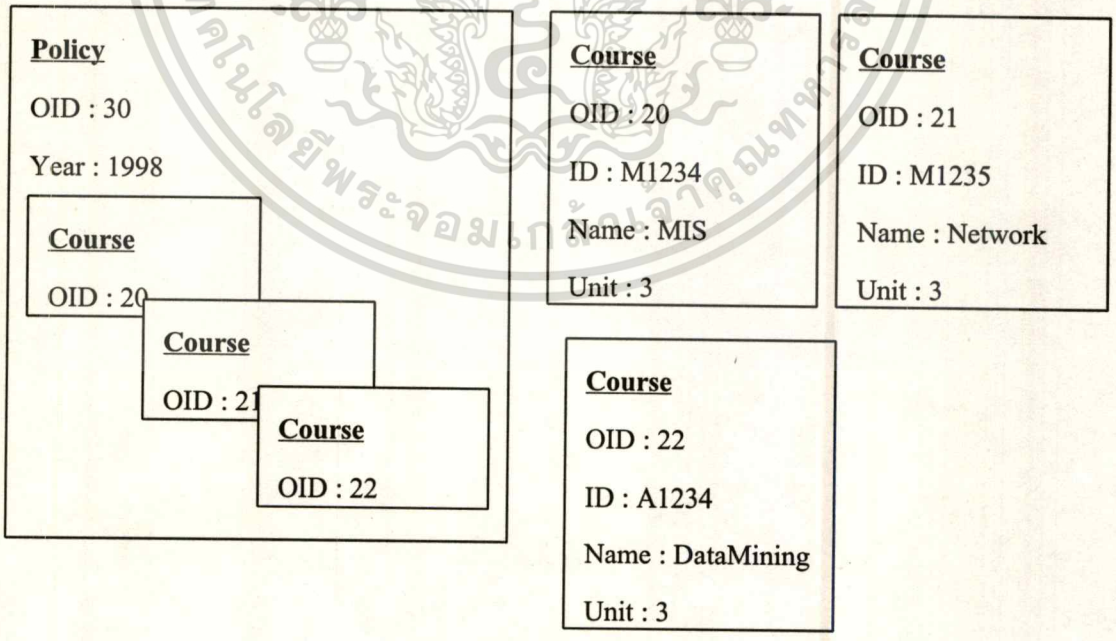
ภาพที่ 6.8 การป้อนข้อมูลหลักสูตรเข้าสู่ระบบ

ในการป้อนข้อมูลหลักสูตรเข้าสู่ระบบเราจะต้องทำการระบุว่าในปีนี้จะมีวิชาอะไรที่ต้องเรียนบ้าง เป็นหลักสูตรของระดับชั้นใด คณะใด และต้องระบุวิชาบังคับ และวิชาเลือกว่าต้องลงกี่หน่วยกิต และมีวิชาใดบ้าง โดยการกดปุ่ม Click ขวา ที่ กรอบของทั้งวิชาเลือก และวิชาบังคับ และในกรณีที่ต้องการยกเลิกทำได้โดยการกดปุ่ม “Remove Course” โดยหลังจากที่เลือก “Add Course” แล้ว จะมีหน้าจอแสดงข้อมูลรายวิชาให้เลือกดังภาพ



ภาพที่ 6.9 การเลือกวิชามาใส่ในหลักสูตร

ในการเลือกรายวิชามาใส่ในหลักสูตร ระบบจะแสดงรายการวิชาทั้งหมดที่มีให้เราสามารถเลือกมาใส่ในระบบได้ โดยลักษณะของการเก็บข้อมูลจะมีลักษณะดังนี้



ภาพที่ 6.10 ลักษณะการเก็บข้อมูลของ Policy Class และ Course

6.2.2. การแก้ไขข้อมูล(Update)

ในระบบนี้สามารถทำการแก้ไขข้อมูลต่างๆ ได้ ไม่ว่าจะเป็นข้อมูลของนักศึกษา ข้อมูลวิชา รวมไปถึงข้อมูลของหลักสูตรด้วย โดยจะนำเสนอแนวทางและลักษณะการทำงานในเชิงวัตถุดังนี้

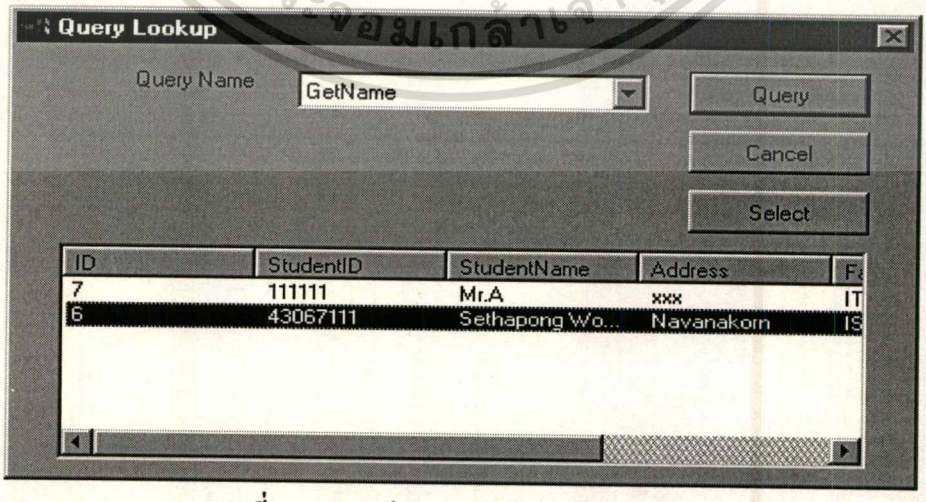
- การแก้ไขข้อมูลของหลักสูตร การเปลี่ยนแปลงหลักสูตรได้แก่การเปลี่ยนแปลงวิชาในหลักสูตร ทำได้โดยการเพิ่มรายวิชาเข้าในหลักสูตรหรือทำการนำรายวิชาออกจากระบบ ซึ่งลักษณะการเก็บข้อมูลจะมีลักษณะเกียวกันกับการนำข้อมูลเข้าสู่ระบบ

- การแก้ไขข้อมูลวิชา เช่นเมื่อมีการเปลี่ยนแปลงหลักสูตร โดยทั่วไปรหัสวิชาจะมีการเปลี่ยนแปลงไปด้วย ซึ่งในระบบสามารถทำได้โดย เรียกข้อมูลเก่าขึ้นมา ทำการแก้ไข แล้วทำการ Update ลงในฐานข้อมูล

จะเห็นได้ว่าในการแก้ไขค่าต่าง ๆ ในแต่ละ Object จะไม่ส่งผลกระทบถึง Object อื่น ๆ ดังเช่นในระบบฐานข้อมูลเชิงสัมพันธ์ ที่ทำการ Update Table ที่มีการอ้างถึงด้วยค่าที่เราต้องการจะแก้ไขนั้น

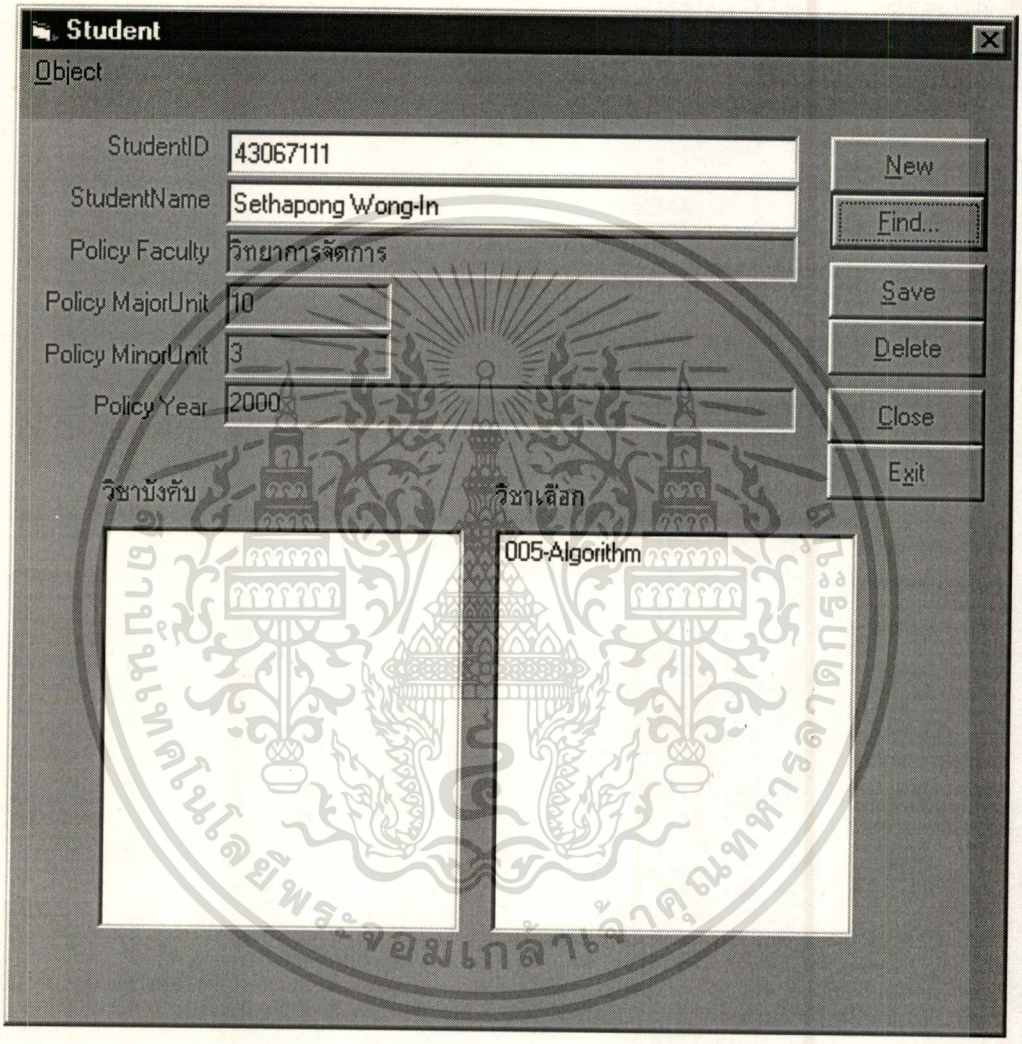
6.2.3. การตรวจสอบการสำเร็จการศึกษา

ในการที่จะดูว่านักศึกษาคนนี้จะจบการศึกษาหรือไม่หรือจะดูว่ายังขาดวิชาใดที่ยังไม่ได้ลงเรียน ถ้าเป็นในระบบฐานข้อมูลเชิงสัมพันธ์ จะต้องใช้การ Join Table ซึ่งเป็นการยุ่งยาก แต่ในระบบฐานข้อมูลเชิงวัตถุซึ่งอ้างถึงข้อมูลใน Object อื่นโดยใช้ OID ในกรณีศึกษานี้การที่นักศึกษาจะดูว่าตนเองเรียนครบตามที่หลักสูตรกำหนดหรือไม่ทำได้โดย ทำการตรวจสอบประวัติของตนเอง โดยจะทำการเปรียบเทียบกับข้อมูลวิชาใน Class Policy โดยการใช้ Method GetCourseName() ของ Object Policy มาทำการเปรียบเทียบกับค่าที่ได้จากการใช้ Method GetCourse ของ Object Student ในโปรแกรมกรณีศึกษานี้จะมีหน้าจอดังนี้



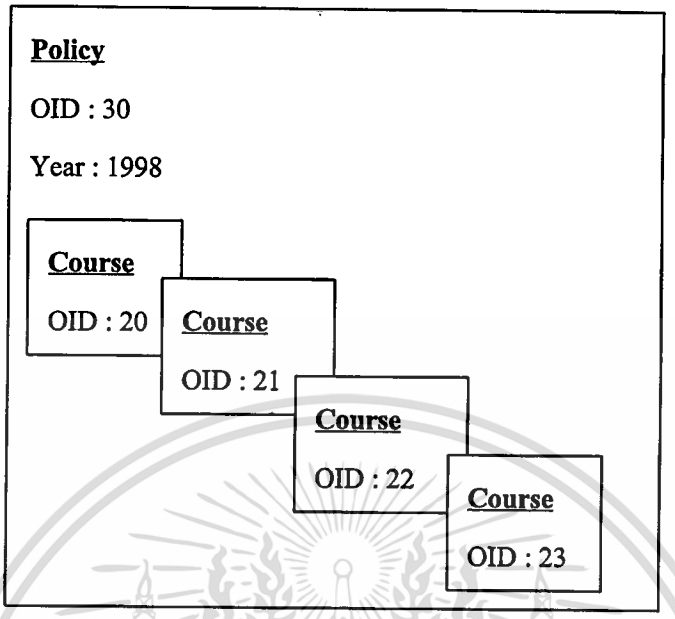
ภาพที่ 6.11 การเรียกดูประวัติการศึกษา

จากภาพที่ 6.11 หน้าจอประวัติการศึกษา เราสามารถเลือกดูว่านักศึกษาคนนี้จะต้องลงเรียนวิชาใดบ้างจึงจะจบการศึกษาโดยเลือกจากรายชื่อนักศึกษาในระบบ จะปรากฏหน้าจอดังนี้

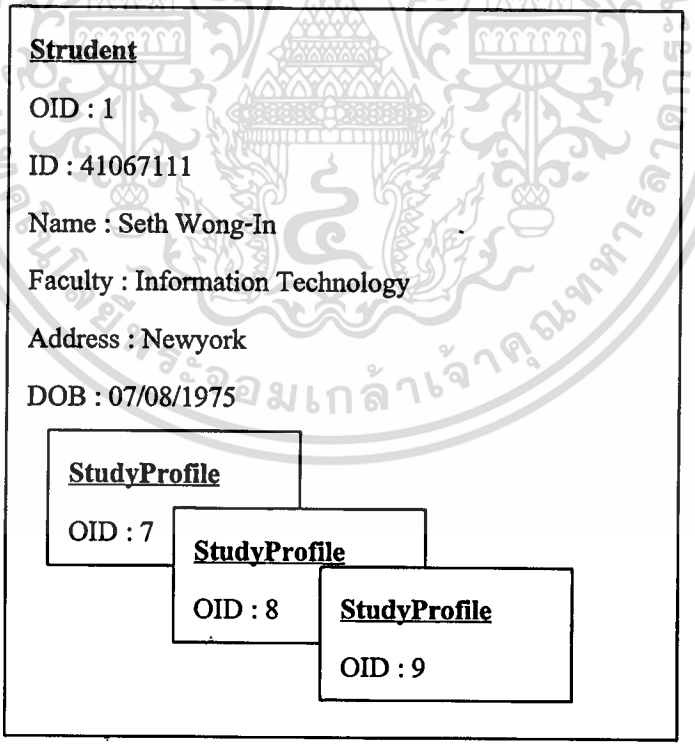


ภาพที่ 6.12 หน้าจอแสดงรายวิชาที่ยังไม่ได้ลงเรียน

จากภาพที่ 6.12 จะแสดงรายวิชาที่ยังไม่ได้ลงเรียนหรือยังไม่ผ่าน ซึ่งได้มาจากการที่เรานำข้อมูลใน Class Policy มาเปรียบเทียบกับข้อมูลใน Class Student ซึ่งอาจเขียนได้ในลักษณะดังนี้



ภาพที่ 6.13 การเก็บข้อมูลของ Class Policy



ภาพที่ 6.14 ลักษณะการเก็บข้อมูลของ Class Student

จากภาพที่ 6.13 และ 6.14 จะเห็นได้ว่าเมื่อมีการเรียกใช้ Method GetCourseName ใน Class Course ที่อยู่ใน Policy จะทำการแสดงรายชื่อวิชาทั้งหมดในหลักสูตรประจำปีนั้นจาก Class Course โดยอ้างถึงด้วย OID ของ Class Course ที่อยู่ภายใน Class Policy ส่วนใน Class Student จะทำการดึงข้อมูลจาก Class StudyProfile ซึ่งจะไปทำการดึงเอาข้อมูล CourseName ใน Class Course มาแสดงเช่นเดียวกัน

จะเห็นได้ว่าการทำงานโดยใช้ตัวระบุชี้ในกรณีศึกษานี้ แสดงให้เห็นถึงกรแก้ปัญหาการใช้ตัวระบุชี้โดยครอบคลุมทั้ง 3 ปัญหา ซึ่งสามารถสรุปกรณีปัญหาได้ดังนี้

- ปัญหาการแก้ไขตัวระบุชี้ – ในระบบฐานข้อมูลเชิงวัตถุใช้ OID ในการอ้างถึง Object อื่น ดังนั้นเราสามารถที่จะเปลี่ยนค่าต่าง ๆ ใน Object ได้โดยไม่ส่งผลกระทบถึง Object อื่น ๆ ตัวอย่าง เช่น นักศึกษาเปลี่ยนชื่อ นามสกุล ที่อยู่

- ปัญหาการใช้ตัวระบุชี้ที่ไม่เหมาะสม ดังเช่นเมื่อนักศึกษาทำการเปลี่ยนแปลง Attribute ใด ๆ ที่ไม่ใช่ตัวระบุชี้ เมื่อมีการเก็บบันทึกผลการเปลี่ยนแปลงนั้นจะใช้แกนเวลามาเกี่ยวข้อง ในกรณีที่ต้องการจะดูข้อมูลย้อนหลังในระบบฐานข้อมูลเชิงสัมพันธ์จะต้องทำการอ้างถึงโดยใช้แกนเวลาเข้ามาเกี่ยวข้องด้วย แต่ในระบบฐานข้อมูลเชิงสัมพันธ์ใช้ OID อย่างเดียว

- ปัญหาการJoin ในส่วนของวิชา กรณีที่นักศึกษาได้ลงทะเบียนเรียนวิชา MIS รหัส M1234 ในปี 1990 แต่สอบไม่ผ่าน ต่อมาในปี 2000 ได้มีการปรับปรุงหลักสูตรทำให้วิชา Network เปลี่ยนชื่อและรหัสวิชาไปเป็น วิชา Network Management และมีรหัสวิชาเป็น N1234 ถ้านักศึกษาคนนี้จะทำการตรวจสอบว่าตนเองยังต้องเรียนวิชาใดจึงจะสำเร็จการศึกษา ระบบจะต้องสามารถแสดงว่านักศึกษาคนนี้ยังต้องลงเรียนในวิชา N1234 ถึงแม้ว่าจะมีการเปลี่ยนชื่อและรหัสไปแล้วก็ตาม ในระบบฐานข้อมูลเชิงวัตถุสนับสนุนการทำงานแบบ Complex Object ซึ่งเป็น Object ที่สามารถประกอบด้วย Object อื่นอยู่ภายในได้ ดังนั้นเมื่อเราทำการอ้างถึงข้อมูลที่อยู่ใน Object ย่อย ๆ เหล่านี้เสามารถเข้าถึงข้อมูลได้เลย ในขณะที่ระบบฐานข้อมูลเชิงสัมพันธ์ต้องอาศัยหลักการ Join Table ที่ยุ่งยาก

กรณีศึกษาในงานวิจัยนี้เป็นเพียงส่วนหนึ่งของระบบสารสนเทศที่มีปัญหาเกี่ยวกับการใช้งานตัวระบุชี้ ซึ่งในระบบฐานข้อมูลเชิงสัมพันธ์ไม่สามารถแก้ปัญหาเหล่านี้ได้หรืออาจต้องใช้ความยุ่งยากในการออกแบบและจัดการ

บทที่ 7

บทสรุปและแนวทางการพัฒนาในอนาคต

ปัญหาการเปลี่ยนแปลงแก้ไขตัวระบุชี้ (Identifier) ในระบบสารสนเทศเป็นหนึ่งในปัญหาการใช้ตัวระบุชี้ซึ่งจะก่อให้เกิดปัญหาได้ในระบบสารสนเทศที่พัฒนาโดยใช้ระบบฐานข้อมูลเชิงสัมพันธ์ในลักษณะต่าง ๆ ตามที่ได้กล่าวมาแล้ว การนำระบบฐานข้อมูลเชิงวัตถุมาแก้ปัญหาการใช้ตัวระบุชี้นี้ สามารถรองรับปัญหาดังกล่าวได้โดยอาศัยคุณสมบัติของ Object คือ คุณสมบัติ Object Identifier มาใช้ในการอ้างถึงข้อมูลซึ่งอยู่ในวัตถุได้เป็นอย่างดี อีกทั้งยังสามารถรองรับการทำงานที่ซับซ้อนของระบบงานจริงได้ดีกว่าอีกด้วย รวมไปถึงยังสามารถแก้ไขปัญหาต่าง ๆ ในการใช้ตัวระบุชี้ไม่ว่าจะเป็นการเลือกใช้ตัวระบุชี้ และปัญหาการ Join โดยในงานวิจัยนี้ได้นำเสนอถึงหลักการของการพัฒนาระบบเชิงวัตถุโดยใช้ระบบฐานข้อมูลเชิงวัตถุในการเก็บข้อมูลซึ่งจะทำให้การทำงานเป็นเชิงวัตถุโดยแท้จริง ซึ่งในปัจจุบันมีหลายระบบงานที่ยังใช้ระบบฐานข้อมูลเชิงสัมพันธ์ซึ่งเมื่อเกิดปัญหาดังที่กล่าวมาในงานวิจัย ทำให้ระบบงานนั้นไม่สามารถรองรับความต้องการนั้นได้

ในปัจจุบันการพัฒนาระบบงานโดยใช้หลักการของ Object-Oriented ไม่ว่าจะเป็น Object-Oriented Design, Object-Oriented Programming หรือ Object-Oriented Database มีแนวโน้มที่จะเป็นที่นิยมในอนาคตจากข้อดีที่ได้นำเสนอมาแล้วในตอนต้น ทั้งนี้ ในปัจจุบันมีเครื่องมือในการออกแบบและพัฒนาระบบในเชิงวัตถุมากมาย เช่น Rational Rose, UML ซึ่งสามารถทำการออกแบบระบบเชิงวัตถุตั้งแต่ขั้นตอนการออกแบบจนถึง การ Generate เป็น Source Code ได้เลย แต่การที่จะให้ระบบเป็นระบบในเชิงวัตถุอย่างแท้จริงได้นั้น จำเป็นต้องมีการใช้ฐานข้อมูลเชิงวัตถุในการเก็บ Class ต่าง ๆ ไว้ด้วย ซึ่งก็ต้องมี OODBMS ที่สามารถรองรับและสนับสนุนคุณสมบัติต่าง ๆ ของ Object-Oriented ได้ดังที่กล่าวแล้วข้างต้น ซึ่งในปัจจุบัน OODBMS ที่นิยมใช้กันได้แก่ Justmine,O2,ObjectStore,GemStone,Ontos,Versant,Objectivity และที่ผู้ทำได้นำมาใช้ในการพัฒนาได้แก่ Cache ซึ่งมีข้อดีดังเหตุผลที่กล่าวมาแล้วข้างต้น นอกจากนี้การออกแบบและพัฒนาเชิงวัตถุยังสามารถที่จะออกแบบระบบที่มีความซับซ้อนได้ดีกว่าเนื่องจากจะมีลักษณะเป็น Real World มากกว่า

บรรณานุกรม

Alfon, Kemper and Guido Moerkotte. 1994. Object-Oriented Database Management : Applications in Engineering and Computer Science. Englewood Cliffs, NJ. : Prentice Hall.

Batanov, Dentcho N. 1996. Fundamentals of Object-Oriented Analysis, Design and Programming. Bangkok.

CACHE Post-Relation DBMS. 1999. CACHE.[Online]. Available :URL:HTTP://www.e-dbms.com .

George Wilkie. 1994. Object-Oriented Software Engineering, The MARI Group.

Hans-Erik Eriksson and Magnus Penker. 1998. UML Toolkit, Wiley Computer Publishing.

Kazuhiko, Kato and Musada Takashi. 1992. "Persistent Caching : An Implementation Techique for Complex Object with Object Identity." IEEE Transactions on Software Engineering. 18(July 1992): 631-645.

Kevin Lano and Howard Hooughton. 1994. Object-Oriented Specification Case Studies, Prentice Hall.

Khoshafian, Setrag N.., and George P. Copeland. 1996. "Object Identity." Proceeding of the 1st OOPSLA Conference(Portland, Ore.,1986).ACM,Newyork,406-416.

Manola, Frank. 1994. An Evaluation of Object-Oriented DBMS Developments. Waltham,MA. : GTE Laboratories Incorporated.

Steve McClare. 1997. "Object Database VS Object-Relational Databases." IDC Bullentin#14821E.

ประวัติผู้เขียน

ชื่อผู้เขียน	นายเศรษฐพงศ์ วงษ์อินทร์
วันเดือนปีเกิด	7 สิงหาคม 2518
สถานที่เกิด	กรุงเทพมหานคร
วุฒิการศึกษาระดับปริญญาตรี	วท.บ.(สถิติประยุกต์)
สถานที่สำเร็จการศึกษา	คณะวิทยาศาสตร์ประยุกต์ สถาบันเทคโนโลยีพระจอมเกล้าพระนครเหนือ
ปีที่สำเร็จการศึกษาในระดับปริญญาตรี	2539