สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

การออกแบบและพัฒนาระบบเชิงวัตถุโดยใช้ OONIAM/UML
: กรณีศึกษาระบบหลักสูตร

THE DESIGN AND IMPLEMENTATION OF AN OBJECT – ORIENTED SYSTEM USING OONIAM/UML : THE CASE STUDY OF COURSE SYSTEM

T 0 4 6 1 7 8

นางสาวชญานันท์ พิทักษ์วงศ์โรจน์
นายอรรถสิทธิ์ สุรรังสรรค์

เลขหมู่.....
เลขทะเบียน 46178
วัน, เดือน, ปี 2 0 มี.ค. 2546

.b.....
.i.....

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์
ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2544

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้
b11085037

การออกแบบและพัฒนาระบบเชิงวัตถุโดยใช้ OONIAM/UML : กรณีศึกษาระบบหลักสูตร

THE DESIGN AND IMPLEMENTATION OF AN OBJECT – ORIENTED SYSTEM USING OONIAM/UML : THE CASE STUDY OF COURSE SYSTEM

โดย

นางสาวชญานันท์ พิทักษ์วงศ์โรจน์

นายอรรถสิทธิ์ สุรรังสรรค์

อาจารย์ที่ปรึกษา

รศ.ดร. ศุภมิตร จิตตะยโศธร

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2544

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ปริญญานิพนธ์ปีการศึกษา 2544

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การออกแบบและพัฒนาระบบเชิงวัตถุโดยใช้ OONIAM/UML : กรณีศึกษาระบบหลักสูตร

The Design and Implementation of an Object – Oriented System Using OONIAM/UML : The Case Study of Course System

ผู้จัดทำ

1. นางสาวชญานันท์ พิทักษ์วงศ์โรจน์ รหัสประจำตัว 41014499

1. นายอรรถสิทธิ์ สุรรังสรรค์ รหัสประจำตัว 41014528

อาจารย์ที่ปรึกษา

(รศ.ดร. ศุภมิตร จิตตะยโศธร)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การออกแบบและพัฒนาระบบเชิงวัตถุโดยใช้ OONIAM/UML : กรณีศึกษาระบบหลักสูตร

นางสาวชญานันท์ พิทักษ์วงศ์โรจน์
นายอรรถสิทธิ์ สุรรังสรรค์
รศ.ดร.ศุภมิตร จิตตะยโศธร อาจารย์ที่ปรึกษา
ปีการศึกษา 2544

บทคัดย่อ

ในปัจจุบันเทคโนโลยีทางด้านฐานข้อมูลได้พัฒนาไปมาก ทั้งในแง่ของประสิทธิภาพการทำงานที่เร็วขึ้น ทั้งในแง่ของความสามารถที่มากขึ้น ทำให้ได้รับการนำไปใช้อย่างกว้างขวาง ในส่วนของแนวทางที่ใช้ในการจัดเก็บข้อมูลในฐานข้อมูลก็มีการพัฒนาและคิดค้นขึ้นมาอยู่เรื่อย ๆ เช่นกัน ตัวอย่างหนึ่งของแนวทางเหล่านี้ได้แก่ การนำแนวคิดเชิงวัตถุ (Object-Oriented Concept) ซึ่งมีจุดกำเนิดมาจากส่วนของวิธีการพัฒนาและเขียนโปรแกรมก่อน ต่อมาก็ได้เริ่มมีการนำมาใช้กับระบบฐานข้อมูล เรียกว่าระบบฐานข้อมูลเชิงวัตถุ (Object-Oriented Database) ซึ่งทำให้ระบบจัดการฐานข้อมูลที่ใช้ฐานข้อมูลเชิงวัตถุสามารถจัดเก็บข้อมูลที่มีลักษณะที่ซ้ำซ้อนได้ดีกว่าแบบเดิมซึ่งเป็นแบบฐานข้อมูลสัมพันธ์ (Relational Database)

วิทยานิพนธ์ฉบับนี้จะเน้นในด้านการศึกษาและวิจัยการออกแบบโปรแกรมและฐานข้อมูลเชิงวัตถุ ทำการทดลองสร้างโปรแกรมและฐานข้อมูลที่สามารถปรับปรุงให้ใช้งานในองค์กรที่มีกฎเกณฑ์แตกต่างกันได้ที่เรียกว่า คอมมอนซอฟต์แวร์ (Common Software) โดยอาศัยแบบจำลองที่ชื่อว่า Unified Modeling Language (UML) ในการออกแบบระบบเชิงวัตถุ และใช้แบบจำลองไนแอมเชิงวัตถุ (OONIAM) ในการช่วยออกแบบฐานข้อมูลเชิงวัตถุ และจะไม่เน้นในส่วนของการพัฒนาระบบงานเพื่อนำไปใช้งานจริง แต่จะทำการสร้างเพียงแค่ระบบต้นแบบ และทำการวิเคราะห์ UML และ OONIAM ว่าสามารถนำมาใช้ในการออกแบบระบบเชิงวัตถุได้ดีเพียงใด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

The Design and Implementation of an Object - Oriented System Using OONIAM/UML : The Case Study of Course System

Chayanan Pithukwongrote

Arthasit Surarangsan

Assoc. Prof. Dr. Suphamit chittayasothon Advisor

ABSTRACT

At present, Database Technology is developed rapidly both in performance and ability. Consequently, it is widely used in many applications. The new methods for storing data in the database are also developed. One of these methods is applying the object-oriented concept, which is previously used in software development, to relational database which results in object-relational database. Object-relational database is better for storing data which contains redundancy than relational database.

This thesis focuses on study and research in application design and object-relation database design. The objective of the research is to find the weaknesses or holes in the design tool and the solutions for the problems. The database application built as an example in this thesis will be common software - can be customized to satisfy varies organization rules. The Unified Modeling Language (UML) will be used to design application and the object-oriented NIAM (OONIAM) will be used to design the object-relational database. The purpose of the application is to show how well can the UML and OONIAM be used to design application. The completeness of the application is not the main objective.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้คงไม่อาจเสร็จได้ด้วยดี หากไม่ได้รับความช่วยเหลือ และร่วมมือจากหลาย ๆ ฝ่ายด้วยกัน บุคคลแรกที่ต้องกล่าวถึงเพราะเป็นส่วนสำคัญที่ทำให้วิทยานิพนธ์นี้เสร็จลงได้ก็คือ อาจารย์ ศุภมิตร จิตตะยโศธร อาจารย์ที่ปรึกษาวิทยานิพนธ์ ที่ให้ความเอาใจใส่ แนะนำ และช่วยเหลือเสมอมา ซึ่งต้องขอขอบพระคุณเป็นอย่างมาก

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

ชญานันท์ พิทักษ์วงศ์โรจน์
อรรถสิทธิ์ สุรรังสรรค์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ

หน้าที่

บทคัดย่อภาษาไทย	I
บทคัดย่อภาษาอังกฤษ	II
กิตติกรรมประกาศ	III
สารบัญ	IV
สารบัญตาราง	VIII
สารบัญภาพ	IX
บทที่ 1 บทนำ	1
1.1 ความสำคัญและที่มา	1
1.2 วัตถุประสงค์ของงานวิจัย	2
1.3 ขอบเขตของงานวิจัย	2
1.4 วิธีการดำเนินงาน	2
บทที่ 2 ระบบฐานข้อมูลและฐานข้อมูลเชิงวัตถุ	3
2.1 บทนำ	3
2.2 ฐานข้อมูลเชิงสัมพันธ์และการพัฒนาโปรแกรมด้วยภาษาเชิงวัตถุ	4
2.2.1 ข้อดีของฐานข้อมูลเชิงสัมพันธ์	4
2.2.2 ข้อดีของการพัฒนาโปรแกรมเชิงวัตถุ	4
2.3 ระบบฐานข้อมูลเชิงวัตถุ (OBJECT-ORIENTED DATABASE SYSTEM: OODBS)	4
2.3.1 วัตถุเชิงซ้อน (Complex Object) และคลาส (Class)	4
2.3.2 ตัวชี้วัตถุ (Object Identity)	5
2.3.3 การสืบทอดคุณสมบัติ (Inheritance)	6
2.3.4 โพลีมอร์ฟิซึม	6
2.4 สถาปัตยกรรมของระบบจัดการฐานข้อมูลเชิงวัตถุ (The Architecture of OODBMS)	7
2.5 แบบจำลองฐานข้อมูลเชิงวัตถุ (Object Oriented Database Model)	7
2.5.1 ส่วนประกอบพื้นฐานและสัญลักษณ์ของแบบจำลองฐานข้อมูลเชิงวัตถุ	8
2.5.2 คลาสแอสโซซิเอชั่น (Association classes)	10
2.6 การเปรียบเทียบแบบจำลองฐานข้อมูลเชิงวัตถุกับแบบจำลองฐานข้อมูลเชิงสัมพันธ์	14
2.6.1 ชนิดของข้อมูล (Data Types)	14
2.6.2 ความแน่นอนของข้อมูล (Data Integrity)	14
2.6.3 วิวัฒนาการของโครงสร้าง (Schema Evolution)	15
2.7 ประโยชน์จากการใช้ฐานข้อมูลเชิงวัตถุ	16
บทที่ 3 แบบจำลองสำหรับการวิเคราะห์และออกแบบระบบเชิงวัตถุ	18

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.1 บทนำ 18
3.2 UML (Unified Modeling Language) 18
3.3 มุมมองของ UML 19
3.4 แผนภาพของ UML 20
3.4.1 Use-case Diagram 20
3.4.2 ซีเควนต์ไดอะแกรม (Sequence Diagram) 22
3.4.3 คอลแลโบเรชั่นไดอะแกรม (Collaboration Diagram) 25
3.4.4 สเตทไดอะแกรม (State Diagram) หรือเรียกว่า State Machine ไดอะแกรม 26
3.4.5 แอ็กทิวิตี้ไดอะแกรม (Activity Diagram) 31
3.4.6 คลาสไดอะแกรม (Class Diagram) 37
3.4.7 ออบเจ็กต์ไดอะแกรม (Object Diagram) 38
3.4.8 คอมโพเน็นต์ไดอะแกรม (Component Diagram) 38
3.4.9 ดีพลอยเมนต์ไดอะแกรม (Deployment Diagram) 38
บทที่ 4 แบบจำลองสำหรับการออกแบบฐานข้อมูลเชิงวัตถุ 39
4.1 บทนำ 39
4.2 ความหมายของ NIAM และการใช้ 39
4.2.1 ส่วนประกอบพื้นฐานของไนแอม 40
4.2.2 กฎข้อบังคับความถูกต้องของข้อมูลที่ใช้ในแบบจำลองระดับแนวคิดไนแอม 43
4.2.2.1 Intra fact type constraints (internal uniqueness constraints) 43
4.2.2.2 Inter fact type uniqueness constraints (external uniqueness constraints) 45
4.2.2.3 Mandatory role constraints 45
4.2.2.4 Inclusion mandatory role constraints 45
4.2.2.5 Entity type constraints (Value constraints) 46
4.2.2.6 Subset constraint 46
4.2.2.7 Equality constraints 47
4.2.2.8 Exclusion constraints 48
4.2.2.9 Subtype constraints 48
4.2.2.10 Occurrence frequency constraints 49
4.2.3 The Optimal Normal Form algorithm (ONF อัลกอริธึม) 49
4.3 แบบจำลองแนวคิด OONIAM และการแปลงรูป 50
4.3.1 แบบจำลองแนวคิด OONIAM 51
4.3.1.1 โครงร่างหลัก (Main Schema) 52
4.3.1.2 โครงร่างย่อย 53
4.3.2 การแปลงแบบจำลอง OONIAM ให้เป็นภาษานิยามเชิงวัตถุ ODL 54

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.3.2.1 การแปลงโครงร่างย่อย 56
4.3.2.2 การแปลงโครงร่างหลัก 57
4.3.3 ตัวอย่างของ OONIAM และการแปลงให้เป็นภาษา ODL 57
4.3.4 แปลงตัวอย่าง OONIAM ให้เป็นภาษา ODL 59
บทที่ 5 ระบบจัดการฐานข้อมูลเชิงวัตถุ คาเช่ (Caché) 61
5.1 บทนำ 61
5.2 คุณสมบัติด้านต่างๆ ของ Caché 61
5.3 องค์ประกอบของ Caché 62
5.4 Object Model 62
5.5 การอ้างอิงออบเจ็กต์ 63
5.6 คลาส 63
5.6.1 Registered Class 64
5.6.2 Persistent Class 64
5.6.3 Embeddable Class 64
5.6.4 Non – Registered Class 65
5.6.5 Data Type Class 65
5.6.6 Caché กับการถ่ายทอดคุณสมบัติ 66
5.6.7 Caché กับ Multiple Inheritance 66
5.6.8 Caché กับความสัมพันธ์ (Relationship) 66
5.6.9 การกำหนดความสัมพันธ์ 67
5.6.10 Caché กับ Application 67
บทที่ 6 แนวคิดโปรแกรมประยุกต์กลาง(Common Application Concept) 69
6.1 ระบบโปรแกรมประยุกต์กลาง 69
6.2 กรณีศึกษา : ระบบงานหลักสูตร 69
6.3 การออกแบบโดยใช้ UML 70
6.3.1 Use Case Diagram 70
6.3.2 Sequence Diagram 72
6.3.3 Class Diagram 85
6.3.4 State-Activity Diagram 98
6.3.5 Component Diagram 103
6.3.6 OONIAM Diagram 104
6.4 การพัฒนาโปรแกรมด้วย Delphi 6.0 ร่วมกับ Caché DBMS 111
6.5 การเข้าถึงข้อมูล 113
6.5.1 การเข้าถึงข้อมูลระดับวัตถุ Object 113

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

6.5.2 การเข้าถึงสมาชิกของข้อมูลที่เป็น Array Collection 113
6.5.3 การเข้าถึงสมาชิกของข้อมูลที่เป็นคลาสและเป็น Embeded Class 113
6.5.4 การเข้าถึงสมาชิกของข้อมูลที่เป็นคลาสและเป็น Referenced Class 114
6.5.5 การเข้าถึงข้อมูลด้วยคิวรี (Access with SQL) 114
6.6 การสร้างระบบ Customize โดยใช้ Library ของ Caché 115
6.6.1 การนำเข้าคลาส CDL 116
6.6.2 การส่งออกคลาส CDL,ODL,JAVA 116
6.6.3 การคอมไพล์คลาส 116
บทที่ 7.สรุปผลการวิจัยและข้อเสนอแนะ 117
7.1 สรุปผลการวิจัย 117
7.2 ข้อเสนอแนะ 119
7.3 แนวทางในการพัฒนา 120
ภาคผนวก หน้าต่างของโปรแกรม 121
บรรณานุกรม 129

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญตาราง

	หน้าที่
ตารางที่ 2 – 1 แสดงสัญลักษณ์ที่ใช้แทนความหมายของ Multiplicity of Association	10
ตารางที่ 5 - 1 แสดงการเข้าถึงข้อมูลของ Caché	62
ตารางที่ 5 - 2 แสดงส่วนประกอบของ Caché	62
ตารางที่ 6 - 1 แสดงความหมายของแต่ละยูสเคสและแอ็คเตอร์	71
ตารางที่ 6 - 2 แสดงข้อดี-ข้อเสียของการติดต่อ	112

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญภาพ

หน้าที่

รูปที่ 2 - 1 สถาปัตยกรรมของระบบจัดการฐานข้อมูลเชิงวัตถุ 7

รูปที่ 2 – 2 แสดงสัญลักษณ์และตัวอย่างของคลาส 8

รูปที่ 2 – 3 แสดงสัญลักษณ์ความสัมพันธ์แบบแอสโซซิเอชั่น 9

รูปที่ 2 – 4 แสดงชื่อของแอสโซซิเอชั่น 9

รูปที่ 2 – 5 แสดงสัญลักษณ์ของชื่อบทบาท 9

รูปที่ 2 – 6 แสดงตัวอย่างของ Multiplicity of Associations 10

รูปที่ 2 – 7 แสดงคลาสแอสโซซิเอชั่น 11

รูปที่ 2 – 8 แสดงตัวอย่างของแอตทริบิวต์แอสโซซิเอชั่น 11

รูปที่ 2 – 9 แสดงตัวอย่างความสัมพันธ์แบบ N-ary Associations 11

รูปที่ 2 – 10 แสดงความสัมพันธ์แบบแอกกรีชั่น 12

รูปที่ 2 – 11 แสดงความสัมพันธ์แบบแอกกรีชั่นในลักษณะของการเป็นส่วนประกอบ 12

รูปที่ 2 – 12 แสดงความสัมพันธ์แบบเจนเนอรัลไลเซชั่น 12

รูปที่ 2 – 13 แสดงกฎข้อบังคับความถูกต้องของข้อมูลแบบ Ordered 13

รูปที่ 2 – 14 แสดงกฎข้อบังคับความถูกต้องของข้อมูลแบบ subset 13

รูปที่ 2 – 15 แสดงกฎข้อบังคับความถูกต้องของข้อมูลแบบ exclusive or 13

รูปที่ 2 - 16 แสดงกฎข้อบังคับความถูกต้องแบบ Reflexive association 13

รูปที่ 3 – 1 แผนภาพแสดง UML ไดอะแกรมทั้งหมด 20

รูปที่ 3 – 2 สัญลักษณ์ของ General-purpose concept 20

รูปที่ 3 – 3 สัญลักษณ์ที่ใช้ในยูซเคส ไดอะแกรม 22

รูปที่ 3 - 4 ตัวอย่างยูซเคส ไดอะแกรม 22

รูปที่ 3 – 5 ตัวอย่างซีเควนซ์ไดอะแกรมที่มีการควบคุมการทำงาน, การใช้เงื่อนไข, 23

รูปที่ 3 – 6 ตัวอย่างทรานสิชั่นไทม์ 24

รูปที่ 3 – 7 ตัวอย่างไดอะแกรมคอลแลโบเรชั่น 26

รูปที่ 3 - 8 แสดงตัวอย่างสเตทไดอะแกรม 27

รูปที่ 3 - 9 แสดงตัวอย่างสเตท 28

รูปที่ 3 - 10 แสดงการเชื่อมสเตทด้วยเส้นการทำงาน 28

รูปที่ 3 - 11 แสดงการใส่รายละเอียดบนเส้นการทำงาน 28

รูปที่ 3 – 12 แสดง State Dialing ที่มีการแบ่งสถานะการทำงานย่อยแบบ Sequential 29

รูปที่ 3 - 13 แสดง State Dialing ที่มีการแบ่งสถานะการทำงานย่อยแบบ Concurrent 30

รูปที่ 3 - 14 แสดงลักษณะการทำงานและสัญลักษณ์ที่ใช้ใน History State 30

รูปที่ 3 - 15 แสดงสัญลักษณ์ที่ใช้ในแอ็กทิวิตี้ไดอะแกรม 32

รูปที่ 3 - 16 แสดงการสร้างทางเลือกให้แก่แอ็กทิวิตี้ไดอะแกรม 33

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รูปที่ 3 - 17 แสดงสัญลักษณ์ที่ใช้ในการจัดกลุ่มงานที่ทำพร้อมกันในแอ็กทิวิตี้ไดอะแกรม 34
รูปที่ 3 - 18 แทนเหตุการณ์ (event) ที่เป็นอินพุท 34
รูปที่ 3 - 19 แทนเหตุการณ์ (event) ที่เป็นเอาท์พุท 34
รูปที่ 3 - 20 แสดงตัวอย่างการใช้สัญลักษณ์การส่งสัญญาณในแอ็กทิวิตี้ไดอะแกรม swimlane 35
รูปที่ 3 - 21 แสดงตัวอย่างการใช้งาน Swimlanes ในแอ็กทิวิตี้ไดอะแกรม 36
รูปที่ 4 - 1 แสดงสัญญลักษณ์ของชนิดเอนติตีภาควิชา 40
รูปที่ 4 - 2 แสดงสัญญลักษณ์ของชนิดเลเบิลรหัสภาควิชา 40
รูปที่ 4 - 3 แสดงความสัมพันธ์อ้างอิงแบบ one to one 40
รูปที่ 4 - 4 แสดงการเขียนความสัมพันธ์อ้างอิงแบบ one to one อย่างย่อ 41
รูปที่ 4 - 5 แสดงความจริงแบบ many to one 41
รูปที่ 4 - 6 แสดงความจริงแบบ many to many 41
รูปที่ 4 - 7 แสดงการใช้ intra fact type uniqueness constraint 41
รูปที่ 4 - 8 แสดงการใช้ inter fact type uniqueness constraint 41
รูปที่ 4 - 9 แสดงการใช้ equality constraint 42
รูปที่ 4 - 10 แสดงการใช้ exclusion constraint 42
รูปที่ 4 - 11 แสดงการใช้ subset constraint 42
รูปที่ 4 - 12 แสดงการใช้ subtype constraint 42
รูปที่ 4 - 13 แสดงการใช้ mandatory constraint, lexical constraint 43
รูปที่ 4 - 14 Employee ทุกคนต้องมีรายได้ และทุกคนต้องเสียภาษี 43
รูปที่ 4 - 15 Ternary Fact Type หมายความว่า ข้อมูลการเรียนของนักศึกษาทุกคนจะต้องมีทั้งรหัสวิชาและเกรด 43
รูปที่ 4 - 16 Nested Fact Type มีความหมายเหมือนรูปที่ 4-15 43
รูปที่ 4 - 17 แสดงความสัมพันธ์แบบหนึ่งต่อหลายหน่วย 44
รูปที่ 4 - 18 แสดงความสัมพันธ์แบบหนึ่งต่อหนึ่งหน่วย 44
รูปที่ 4 - 19 แสดงความสัมพันธ์แบบหลายหน่วยต่อหลายหน่วย 44
รูปที่ 4 - 20 แสดง Inter fact type uniqueness constrains 45
รูปที่ 4 - 21 แสดง Mandatory role constraints 45
รูปที่ 4 - 22 แสดง Inclusion mandatory role constraints 46
รูปที่ 4 - 23 แสดง Entity type constraints 46
รูปที่ 4 - 24 แสดง Subset constraints 47
รูปที่ 4 - 24 แสดง Subset constraints (ต่อ) 47
รูปที่ 4 - 25 แสดง Equality constraints 47
รูปที่ 4 – 25 แสดง Equality constraints (ต่อ) 48
รูปที่ 4 - 26 แสดง Exclusion constraints 48

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รูปที่ 4 - 27 แสดง Subtype constraints 49
รูปที่ 4 - 28 แสดง Occurrence frequency constraints 49
รูปที่ 4 - 29 ตัวอย่างแบบจำลองข้อมูล(Conceptual Schema) 50
รูปที่ 4 - 30 รีเลชันของแบบจำลองรูปที่ 4 - 29 50
รูปที่ 4 - 31 ขั้นตอนการทำงานเพื่อนำ OONIAM ไปทำงานร่วมกับฐานข้อมูลเชิงวัตถุ 51
รูปที่ 4 – 32 สัญลักษณ์ของการประมวลผลข้อมูล (เมธอด) 52
รูปที่ 4 – 33 โครงร่างหลักของคลาสหลายคลาส 52
รูปที่ 4 - 34 โครงร่างหลักแบบที่มี Uniqueness Identifier ที่เอนติตี 52
รูปที่ 4 – 35 การถ่ายทอดคุณสมบัติ 53
รูปที่ 4 - 36 โครงร่างย่อยระดับที่ 1 ของคลาส Person 53
รูปที่ 4 - 37 โครงร่างย่อยของคลาส Address 54
รูปที่ 4 - 38 ตัวอย่างโครงร่างหลักของระบบ 54
รูปที่ 4 - 39 ตัวอย่างโครงร่างย่อยระดับที่ 1 55
รูปที่ 4 - 40 ตัวอย่างโครงร่างย่อยระดับที่ 2 55
รูปที่ 4 - 41 ตัวอย่างโครงร่างหลักของระบบ 57
รูปที่ 4 - 42 ตัวอย่างโครงร่างย่อยระดับที่ 1 58
รูปที่ 4 - 43 ตัวอย่างโครงร่างย่อยระดับที่ 2 59
รูปที่ 5 - 1 การติดต่อกับระบบจัดการฐานข้อมูล Caché 61
รูปที่ 5 - 2 OREF และ OID 63
รูปที่ 5 - 3 ประเภทของคลาส 64
รูปที่ 5 - 4 Embedded Object ในหน่วยความจำและดิสก์ 65
รูปที่ 5 - 5 ความสัมพันธ์ระหว่างคลาส Company และคลาส Employee 67
รูปที่ 6 – 1 แสดง Use Case ของระบบงานหลักสูตร 70
รูปที่ 6 – 2 แสดงปฏิสัมพันธ์ของระบบการเพิ่มรายวิชา 72
รูปที่ 6 - 3 แสดงปฏิสัมพันธ์ของระบบการแก้ไขรายวิชา 73
รูปที่ 6 - 4 แสดงปฏิสัมพันธ์ของระบบการเพิ่มข้อมูลคณาจารย์ 74
รูปที่ 6 - 5 แสดงปฏิสัมพันธ์ของระบบการแก้ไขข้อมูลคณาจารย์ 75
รูปที่ 6 - 6 แสดงปฏิสัมพันธ์ของระบบการเพิ่มข้อมูลนักศึกษา 76
รูปที่ 6 - 7 แสดงปฏิสัมพันธ์ของระบบการแก้ไขข้อมูลคณาจารย์ 77
รูปที่ 6 - 8 แสดงปฏิสัมพันธ์ของระบบการเพิ่มข้อมูลคณะ 78
รูปที่ 6 - 9 แสดงปฏิสัมพันธ์ของระบบการแก้ไขข้อมูลคณะ 79
รูปที่ 6 - 10 แสดงปฏิสัมพันธ์ของระบบการเพิ่มข้อมูลภาควิชา 80
รูปที่ 6 - 11 แสดงปฏิสัมพันธ์ของระบบการแก้ไขข้อมูลภาควิชา 81
รูปที่ 6 - 12 แสดงปฏิสัมพันธ์ของระบบการเพิ่มข้อมูลหลักสูตร 82

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รูปที่ 6 - 13 แสดงปฏิสัมพันธ์ของระบบการแก้ไขข้อมูลหลักสูตร 83
รูปที่ 6 - 14 แสดงปฏิสัมพันธ์ของระบบการเพิ่มข้อมูลผลการเรียน 84
รูปที่ 6 – 15 แสดงคลาสฐานข้อมูลในระดับ 1 85
รูปที่ 6 - 16 แสดงคลาสรายวิชาและการสืบทอดคลาส 86
รูปที่ 6 – 17 แสดงคลาสนักศึกษาและการสืบทอด 87
รูปที่ 6 - 18 แสดงคลาสหลักสูตรและการสืบทอด 88
รูปที่ 6 - 19 แสดงคลาสคณาจารย์และการสืบทอด 89
รูปที่ 6 - 20 แสดงคลาสคอนโทรลที่ใช้ใน IDE 90
รูปที่ 6 - 21 แสดงคลาส User Interface ที่ใช้ใน IDE 91
รูปที่ 6 - 22 แสดงคลาส User Interface ของการจัดการหลักสูตร 92
รูปที่ 6 - 23 แสดงคลาสและการติดต่อใน User Interface ของ Explorer (Resource Viewer) 93
รูปที่ 6 - 24 แสดงคลาส User Interface ของคลาสจัดการภาควิชา 94
รูปที่ 6 - 25 แสดงคลาส User Interface ของคลาสจัดการรายวิชา 95
รูปที่ 6 - 26 แสดงคลาส User Interface ของคลาสจัดการนักศึกษา 96
รูปที่ 6 - 27 แสดงคลาส User Interface ของคลาสจัดการคณาจารย์ 97
รูปที่ 6 - 28 แสดงสถานะการทำงานหลักของไดอะล็อกมาตรฐาน 98
รูปที่ 6 – 29 แสดงสถานะการทำงานภายใน Resource viewer 99
รูปที่ 6 - 30 แสดงสถานะส่วนขยายการทำงานในไดอะล็อกจัดการหลักสูตร 100
รูปที่ 6 – 31 แสดงสถานะส่วนขยายการทำงานในไดอะล็อกจัดการรายวิชา 101
รูปที่ 6 – 32 แสดงสถานะส่วนขยายการทำงานในไดอะล็อกจัดการนักศึกษา 102
รูปที่ 6 – 33 แสดงการจัดหมวดหมู่ของคอมโพเนนท์ เพื่อการนำไปอิมพลิเมนท์ 103
รูปที่ 6 – 34 แสดง main schema ในมุมมองของ OONIAM 104
รูปที่ 6 – 35 แสดง Sub schema level 1 ในมุมมองของ OONIAM 105
รูปที่ 6 – 35 แสดง Sub schema level 1 ในมุมมองของ OONIAM (ต่อ) 106
รูปที่ 6 - 35 แสดง Sub schema level 1 ในมุมมองของ OONIAM (ต่อ) 107
รูปที่ 6 - 35 แสดง Sub schema level 1 ในมุมมองของ OONIAM (ต่อ) 108
รูปที่ 6 - 36 แสดง Sub schema level 2 ในมุมมองของ OONIAM 109
รูปที่ 6 - 36 แสดง Sub schema level 2 ในมุมมองของ OONIAM (ต่อ) 110
รูปที่ 7-1 แสดงตัวอย่างปัญหาความสัมพันธ์ระหว่างแอตทริบิวต์ในคลาส 117
รูปที่ 7-2 แสดงตัวอย่างการแก้ปัญหาความสัมพันธ์ระหว่างแอตทริบิวต์ในคลาสด้วย OONIAM 117
รูปที่ 7-3 แสดงตัวอย่างปัญหา FD ระหว่างแอตทริบิวต์ในคลาส 118
รูปที่ 7-4 แสดงตัวอย่างการแก้ปัญหา FD ระหว่างแอตทริบิวต์ในคลาส 118
รูปที่ 7-5 แสดงตัวอย่างปัญหาการกำหนดกฎข้อบังคับใน UML 118
รูปที่ 7-6 แสดงตัวอย่างการแก้ปัญหาการกำหนดกฎข้อบังคับใน UML โดยใช้ User-define list 119

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รูปที่ 7-7 แสดงตัวอย่างการแก้ปัญหาการกำหนดกฎข้อบังคับใน UML โดยใช้ { } 119
รูปที่ 7-8 แสดงตัวอย่างการแก้ปัญหาการกำหนดกฎข้อบังคับใน UML โดยใช้ OONIAM 119

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มา

ในปัจจุบันพัฒนาการทางด้านวิชาฐานข้อมูลนับว่าได้ก้าวหน้าไปมาก มีนักวิทยาศาสตร์ในสาขานี้เป็นจำนวนมากได้ทำการวิจัยและพัฒนาวิธีเก็บข้อมูลจำนวนมากลงบนหน่วยความจำสำรอง (Secondary Memory) โดยมีแบบจำลองข้อมูลในรูปแบบต่างๆ และมีการจัดเก็บที่เป็นระบบ สามารถลดความซ้ำซ้อน (Redundancy) รักษาความถูกต้อง (Integrity) ลดการสูญหายของข่าวสาร (Information Lost) และลดข้อผิดพลาดจากการแก้ไขข้อมูล (Update Anomolies)

นอกจากพัฒนาการทางด้านการออกแบบฐานข้อมูลแล้ว วิชาการทางด้านระบบจัดการฐานข้อมูล (Database Management System : DBMS) ก็ได้รับการพัฒนาไปอย่างมากเช่นเดียวกัน ระบบจัดการฐานข้อมูลได้รับการพัฒนาให้มีคุณภาพสูง และมีสถาปัตยกรรมที่ใกล้เคียงสถาปัตยกรรมมาตรฐานของ ANSI และ ISO มีการแบ่งแยกเป็นระดับแนวคิด (Conceptual Level) เป็นระดับภายนอก (External Level) และระดับภายใน (Internal Level) ที่เด่นชัด มีความเป็นอิสระของข้อมูล (Data Independance) ทั้งทางตรรกและทางกายภาพ

นอกจากนี้ระบบจัดการฐานข้อมูลในปัจจุบันยังมีความปลอดภัย มีระบบเรียกข้อมูลคืนป้องกันการสูญหาย (Recovery) มีระบบใช้ข้อมูลร่วมกันระหว่างผู้ใช้หลายๆ คน (Multiuser) และมีภาษาจัดการข้อมูลที่มีประสิทธิภาพ อย่างน้อยเทียบเท่ากับพีชคณิตรีเลชัน (Relational Algrebra) และใช้เทคโนโลยีทางด้าน Query Optimization เพื่อให้สามารถจัดการกับข้อมูลขนาดใหญ่ได้ด้วยประสิทธิภาพสูง

ไม่เพียงแต่ในเรื่องดังข้างต้นเท่านั้นที่ได้มีการพัฒนา ในส่วนของแนวทางที่ใช้ในการจัดเก็บข้อมูลในฐานข้อมูลก็มีการพัฒนาและคิดค้นขึ้นมาอยู่เรื่อย ๆ เช่นกัน ตัวอย่างหนึ่งของแนวทางเหล่านี้ได้แก่ การนำแนวคิดเชิงวัตถุ (Object-Oriented Concept) ซึ่งมีจุดกำเนิดมาจากส่วนของวิธีการพัฒนาและเขียนโปรแกรมก่อน ต่อมาก็ได้เริ่มมีการนำมาใช้กับระบบฐานข้อมูล เรียกว่า ระบบฐานข้อมูลเชิงวัตถุ (Object-Oriented Database) ซึ่งทำให้ระบบจัดการฐานข้อมูลที่ใช้ฐานข้อมูลเชิงวัตถุสามารถจัดเก็บข้อมูลที่มีลักษณะที่ซ้ำซ้อนได้ดีกว่าแบบเดิมซึ่งเป็นแบบฐานข้อมูลสัมพันธ์ (Relational Database)

เนื่องจากระบบฐานข้อมูลเชิงวัตถุเป็นฐานข้อมูลแบบใหม่ที่นำมาใช้งานไม่แพร่หลายนัก ทำให้ยังไม่มีแบบจำลองสำหรับออกแบบฐานข้อมูลโดยเฉพาะ แต่เป็นการนำเอาแบบจำลองที่ใช้ในการวิเคราะห์และออกแบบฐานข้อมูลเชิงวัตถุมาใช้แทน ตัวอย่างเช่น UML (Unified Modeling Language) ซึ่งถือว่ายังไม่เหมาะสำหรับการออกแบบฐานข้อมูลเชิงวัตถุนัก เนื่องจากเหตุผลหลายประการ ได้แก่

1. ไม่สามารถแสดงความสัมพันธ์ระหว่างแอตทริบิวต์ของคลาสได้
2. ไม่สามารถบ่งบอกการมีได้หลายค่า (Multiplicity Value) ของแอตทริบิวต์ภายในคลาสได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3. ไม่สามารถแยกแยะความแตกต่างของความสัมพันธ์แบบเป็นส่วนหนึ่งของ (Aggregation Relationship) ของแอตทริบิวต์ภายในคลาส ที่อาจเป็นเอมเบดออบเจ็กต์ (Embedded Object) หรือเป็นออบเจ็กต์ที่ถูกอ้างถึงด้วย ID (Reference Object) ก็ได้
4. ไม่สามารถกำหนดกฎข้อบังคับของความถูกต้องของข้อมูล (Integrity Rules) ในฐานข้อมูลได้

ดังนั้น โครงงานวิจัยนี้จึงนำเสนอการใช้ OONIAM มาช่วยในการวิเคราะห์และออกแบบฐานข้อมูลเชิงวัตถุแทนคลาสไดอะแกรม (Class Diagram) ของ UML ส่วนไดอะแกรมอื่น ๆ ของ UML ยังคงนำมาใช้วิเคราะห์และออกแบบระบบเชิงวัตถุได้ โดยโครงงานได้ทำการทดลองสร้างระบบเชิงวัตถุที่สามารถปรับปรุงให้ใช้งานในองค์กรซึ่งมีกฎเกณฑ์แตกต่างกันได้ที่เรียกว่า ซอฟต์แวร์กลาง (Common Software) ซึ่งเป็นการใช้ความสามารถของระบบเชิงวัตถุในด้านโพลีมอร์ฟิซึม (Polymorphism) แต่จะไม่เน้นในส่วนของการพัฒนาระบบงานเพื่อนำไปใช้งานจริง แต่จะทำการสร้างเพียงแค่ระบบต้นแบบเท่านั้น โดยใช้กรณีศึกษาคือระบบหลักสูตร

1.2 วัตถุประสงค์ของงานวิจัย

1.2.1 เพื่อศึกษาการออกแบบระบบเชิงวัตถุที่มีการใช้ฐานข้อมูลเชิงวัตถุด้วย UML และหาวิธีการปรับปรุงให้ UML สามารถรองรับการออกแบบฐานข้อมูลเชิงวัตถุ (Object-Oriented Database) ให้ดียิ่งขึ้น

1.2.2 ทดลองสร้างแอปพลิเคชั่นต้นแบบที่สามารถ Customize ได้สะดวก มาเป็นกรณีศึกษาการปรับปรุง UML โดยอาศัยคุณสมบัติของ Object ทางด้าน Polymorphism และ Inheritance

1.3 ขอบเขตของงานวิจัย

งานวิจัยนี้จะเน้นในด้านการศึกษาแบบจำลอง UML ที่ใช้ในการวิเคราะห์และออกแบบระบบเชิงวัตถุ แล้วทดลองสร้างแอปพลิเคชั่นต้นแบบระบบงานหลักสูตรที่สามารถ Customize ระบบงานได้ง่าย เพื่อทดสอบว่าระบบที่ได้ทำการออกแบบสามารถทำงานได้จริง โดยพัฒนาแอปพลิเคชั่นด้วย Delphi 6.0 และใช้คาเช่ (Caché) เป็นระบบจัดการฐานข้อมูลเชิงวัตถุ โดยจะเน้นในส่วนของการใช้แบบจำลอง UML ว่าสามารถรองรับการออกแบบฐานข้อมูลเชิงวัตถุได้ดีหรือไม่ หรือมีข้อควรปรับปรุงแก้ไขอย่างไร

1.4 วิธีการดำเนินงาน

1.4.1 ศึกษาทฤษฎีพื้นฐานต่างๆ หลักการและการพัฒนาโปรแกรมเชิงวัตถุ

1.4.2 ศึกษาทฤษฎีพื้นฐานต่างๆ หลักการและการพัฒนาฐานข้อมูลเชิงวัตถุ

1.4.3 ศึกษาการใช้งานระบบการจัดการฐานข้อมูลเชิงวัตถุ คาเช่

14.4 ศึกษาเทคโนโลยี COM

1.4.5 นำคาเช่มาประยุกต์ใช้งานร่วมกับการพัฒนาโปรแกรมด้วย Delphi 6.0

1.4.6 ออกแบบและทดลองเพื่อหาข้อสรุป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 2
ระบบฐานข้อมูลและฐานข้อมูลเชิงวัตถุ

2.1 บทนำ

จากการพัฒนาทางด้านเทคโนโลยีทางด้านฐานข้อมูลที่ได้พัฒนาขึ้นมาเรื่อยๆ จนในปัจจุบันได้มีผลิตภัณฑ์เชิงพาณิชย์ของระบบจัดการฐานข้อมูล (Database Management System: DBMS) เกิดขึ้นและใช้กันอย่างแพร่หลาย ซึ่งระบบจัดการฐานข้อมูลเหล่านี้ได้แบ่งเบาภาระหน้าที่ของผู้มีหน้าที่ในการพัฒนาระบบสารสนเทศอย่างมีประสิทธิภาพ ตัวอย่างความสามารถของระบบจัดการข้อมูลดังกล่าวนี้ ได้แก่ ความสามารถในการจัดการเรื่องความซ้ำซ้อนของข้อมูล (Redundancy) ความสามารถในการดูแลความสอดคล้องกันของข้อมูล (Consistency) ความสามารถในการจัดการระเบียบข้อบังคับที่ใช้ในการเก็บข้อมูล (Integrity) ความสามารถในการจัดการเรื่องของการจัดสรรข้อมูลให้สามารถใช้งานโดยโปรแกรมหลายๆ ตัวในเวลาเดียวกันได้ (Concurrency) และความสามารถในการจัดการเรื่องมาตรการด้านความปลอดภัยในการใช้ฐานข้อมูล (Security)

แต่ข้อจำกัดอย่างหนึ่งของระบบจัดการฐานข้อมูลส่วนใหญ่ในปัจจุบันที่เป็นแบบ Relational นี้ ก็คือปัญหาเรื่องความซ้ำซ้อนของข้อมูลที่จัดเก็บ (Complex Object) ซึ่งจะกล่าวถึงต่อไป กล่าวคือฐานข้อมูลดังกล่าว เก็บข้อมูลไว้ในรูปตาราง แต่การกระทำดังกล่าวจะเหมาะกับข้อมูลพื้นฐานที่เป็นตัวเลขและตัวอักษรเท่านั้น แต่หากเป็นข้อมูลที่มีความซ้ำซ้อน เช่น ภาพนิ่ง เสียง หรือภาพเคลื่อนไหว หากจัดเก็บในระบบฐานข้อมูล เมื่อทำการเรียกค้น โดยภาษา SQL ก็จะได้ข้อมูลที่ไม่สามารถทำงานได้ทันที จำเป็นต้องผ่านกระบวนการบางอย่างเพื่อให้ข้อมูลนั้นทำงานได้ เช่น หากจัดเก็บภาพเคลื่อนไหว ระบบจะส่งข้อมูลในลักษณะ BLOB (Binary Large Object Block) ดังนั้นหากผู้ใช้ต้องการที่จะนำข้อมูลดังกล่าวไปแสดงผลก็ต้องมีความรู้ด้านข้อมูลการแปลงไบนารีและมัลติมีเดีย เป็นต้น

ในปัจจุบันข้อมูลไม่ได้อยู่ในรูปของข้อมูลพื้นฐานอีกแล้ว ข้อมูลที่ใช้ในระบบฐานข้อมูลมีความซ้ำซ้อนมากยิ่งขึ้น ดังนั้นการใช้ระบบฐานข้อมูลเชิงสัมพันธ์ จึงไม่สะดวกอีกต่อไปแล้ว ด้วยเหตุนี้จึงได้มีการนำเอาหลักการพัฒนาเชิงวัตถุ (Object-Oriented Concept) มารวมกับเทคโนโลยีระบบฐานข้อมูลเพื่อสร้างเป็นระบบฐานข้อมูลเชิงวัตถุ (Object-Oriented Database: OODB) ซึ่งหัวใจหลักของระบบฐานข้อมูลเชิงวัตถุนี้ก็คือ การมีรหัสเฉพาะในวัตถุ (Object Identification: OID) และการรวมกันระหว่างข้อมูลและการจัดการข้อมูล (Encapsulation) นอกจากนี้ยังมีคุณสมบัติอื่นๆ ที่ช่วยให้การจัดการข้อมูลที่ซับซ้อนได้ เช่นคุณสมบัติการถ่ายทอด (Inheritance) คุณสมบัติการเปลี่ยนแปลงความหมายเมื่อมีการสืบทอด โพลีมอร์ฟิซึม เป็นต้น ซึ่งคุณสมบัติที่กล่าวมานี้ ไม่พบในฐานข้อมูลเชิงสัมพันธ์

2.2 ฐานข้อมูลเชิงสัมพันธ์และการพัฒนาโปรแกรมด้วยภาษาเชิงวัตถุ

ฐานข้อมูลเชิงสัมพันธ์ เกิดขึ้นในปี 1980s ส่วนการพัฒนาโปรแกรมด้วยภาษาเชิงวัตถุ เกิดขึ้นในปลายปี 1980 ถึง 1990s ทั้ง 2 ระบบดูเหมือนจะมีความแตกต่างกันอย่างสิ้นเชิง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.2.1 ข้อดีของฐานข้อมูลเชิงสัมพันธ์

- สามารถจัดการข้อมูลที่มีจำนวนมากได้ดี
- เหมาะกับการค้นคืนข้อมูล แต่ให้การสนับสนุนการปรับปรุงเปลี่ยนแปลงข้อมูลน้อย

2.2.2 ข้อดีของการพัฒนาโปรแกรมเชิงวัตถุ

- สามารถอธิบายความสัมพันธ์ที่ซับซ้อนได้ดี
- สามารถปรับปรุงเปลี่ยนแปลงข้อมูลได้อย่างดีเยี่ยม แต่ไม่ค่อยให้การสนับสนุนการคงสถานะหรือการค้นคืนข้อมูล

หลายคนพยายามรวม 2 ระบบนี้เพื่อการจัดการจำนวนข้อมูลขนาดใหญ่ด้วยความสัมพันธ์ที่ซับซ้อน แต่โชคไม่ดีที่แบบจำลองสัมพันธ์ (Relational model) และแบบจำลองเชิงวัตถุ (object model) แตกต่างกันอย่างยิ่ง และมีการรวมกันได้เพียงอ้อม ๆ เท่านั้น

ฐานข้อมูลเชิงสัมพันธ์ อยู่บนฐานของตาราง 2 มิติซึ่งในแต่ละทัปเปิล (Tuple) ปรากฏเป็น 1 แถว ความสัมพันธ์ระหว่างข้อมูลถูกอธิบายโดยการเปรียบเทียบค่าที่เก็บไว้ในตาราง ภาษาแบบ SQL ยอมให้ตาราง join กันได้ แต่การอธิบายความสัมพันธ์ระหว่างข้อมูลแบบจำลองเชิงวัตถุอยู่บนฐานของการรวมกันอย่างแน่นหนาของโค้ดและข้อมูล มีประเภทข้อมูลที่ยืดหยุ่น, ความสัมพันธ์อย่างเป็นลำดับชั้นของชนิดข้อมูล และการอ้างถึงแบบจำลองสัมพันธ์ทำได้ง่าย แต่มันแตกต่างกันกับแบบจำลองเชิงวัตถุอย่างแท้จริง ฐานข้อมูลเชิงสัมพันธ์ไม่ได้ถูกออกแบบเพื่อเก็บวัตถุและการโปรแกรมการเชื่อมต่อเพื่อเก็บวัตถุในฐานข้อมูลเชิงวัตถุไม่ใช่เรื่องเล็ก ๆ เลย ความหมายของ OOPL และฐานข้อมูลสัมพันธ์ที่ไม่ตรงกันได้นำไปสู่การพัฒนาระบบจัดการฐานข้อมูลเชิงวัตถุที่รองรับแบบจำลองเชิงวัตถุโดยตรง โดยปกติแล้ว OODBs ง่ายกว่ามากที่จะใช้ OOP

2.3 ระบบฐานข้อมูลเชิงวัตถุ (Object-Oriented Database System: OODBs)

2.3.1 วัตถุเชิงซ้อน (Complex Object) และคลาส (Class)

ในยุค 1970s มีผู้พบข้อสังเกตว่าชนิดข้อมูลที่สร้างขึ้นโดยผู้เขียนโปรแกรม (User define data type) โดยใช้อาร์เรย์ (Array) หรือสตรัคเจอร์ (Structures) ที่มีในภาษาพื้นฐาน (Imperative) ทั้งหลาย ไม่สามารถใช้งานได้อย่างปลอดภัยเหมือนกับชนิดข้อมูลพื้นฐานที่มีมากับภาษา เช่น int หรือ float จึงมีการศึกษาและเสนอว่าชนิดข้อมูลที่จะใช้งานได้อย่างปลอดภัยนั้นควรมีลักษณะคือ เมธอด (method) และรายละเอียดของการเก็บแสดงข้อมูลต้องไม่มีความสำคัญต่อการใช้งานข้อมูลชนิดนั้น และต้องไม่สามารถอ้างถึงข้อมูลนั้นได้โดยตรง แต่หากจำเป็นต้องการอ่านหรือเขียนข้อมูลบางส่วนจะต้องทำผ่านทางโอเปอเรชั่น (Operation) ที่กำหนดไว้เท่านั้น ชนิดของข้อมูลที่มีลักษณะดังกล่าวนี้จะถูกเรียกว่า แอ็บสแทรกต์ดาต้าไทป์ (Abstract data type) และวัตถุเชิงซ้อน จากการศึกษาพบว่าภาษาที่สามารถสร้างชนิดข้อมูลแบบนั้นได้ต้องมีกลไกทางภาษาสองอย่างดังต่อไปนี้ คือ (1) การรวมกันระหว่างข้อมูลและการจัดการข้อมูลเป็นกลไกสำหรับนำข้อมูลกับโอเปอเรชั่นมาผูกติดกันเป็นหน่วยหนึ่ง และ (2) การซ่อนข้อมูล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

(Information hiding) เป็นกลไกสำหรับกำหนดการมองเห็นสมาชิกภายในหน่วยที่สร้างขึ้นโดยข้อหนึ่ง คือ ให้สมาชิกแต่ละตัวถูกอ้างถึงภายนอกได้หรือไม่ได้

กลไกทั้งสองถูกพัฒนาให้มีข้อกำหนดให้ภาษาเชิงวัตถุต้องมีอย่างน้อย 4 กลไก คือ

Encapsulation

Information Hiding

Inheritance

Dynamic Binding

ในส่วนที่เกี่ยวข้องกับโครงงานจะมีการอธิบายถึง Inheritance และ Dynamic binding อีกครั้งต่อไป หลักการของวัตถุเชิงซ้อนนี้ จะเป็นการคลายข้อจำกัดใน 1NF (First normal form) ในแบบจำลองสัมพันธ์ วัตถุเชิงซ้อนนี้ สามารถมองได้เป็น Abstraction of structure หรือเรคอร์ด ในภาษาโปรแกรม สิ่งนี้ยอมให้มีกลุ่มของข้อมูล (Repeating group), เรคอร์ดซ้อนเรคอร์ด (Nested record) และสามารถบันทึกข้อมูลในลักษณะทรีได้ โครงสร้างวัตถุเชิงซ้อนอย่างง่ายอาจประกอบไปด้วยวัตถุ ซึ่งแต่ละวัตถุอาจมีชนิดที่แตกต่างกัน ดังนั้นหลักการของวัตถุเชิงซ้อนจะประกอบไปด้วย

กลุ่มของค่าที่แบ่งย่อยอีกไม่ได้ (Set of atomic values)

กลุ่มของแถวหรือความสัมพันธ์ (Set of touples or nested relations)

คอนสตรัคเตอร์และตัวจัดการวัตถุ (General sets and touple constructor)

ตัวอ้างอิงวัตถุ (Object identity)

ข้อมูลสื่อผสม เช่น ลายเซ็น ภาพถ่าย X-Ray เสียง เป็นต้น

ตัวอย่างของวัตถุเชิงซ้อน เช่น ชาวต่างชาติย้ายเข้าเมืองและจำเป็นต้องกรอกหมายเลขโทรศัพท์ลงในแบบฟอร์ม

คำถาม : “คุณมีเบอร์โทรศัพท์ติดต่อหรือไม่ ?”

ตอบ : “แน่นอน ผมทำงานที่อาคาร A และทุกๆ เช้าผมจะอยู่ที่ห้องของผม หมายเลข 001 หมายเลข 002 ขณะอยู่ที่ทำงานตอนบ่าย ผมมีหมายเลข 003 ในรถ และในวันหยุดสามารถติดต่อผมที่หมายเลข 004”

จะเห็นได้ว่าคำถามต้องการเพียงหมายเลขเดียวแต่คำตอบประกอบไปด้วยข้อมูลเชิงซ้อน เป็นต้น

2.3.2 ตัวชี้วัตถุ (Object Identity)

สำหรับในระบบฐานข้อมูลที่เป็นแบบการเน้นการแสดงค่า (Value-Oriented) จะใช้ค่าๆ หนึ่งที่อยู่ในฐานข้อมูลที่เป็นค่าหลักในการแทนถึงวัตถุใดวัตถุหนึ่ง ปัญหาที่พบเมื่อมีการแทนถึงวัตถุในลักษณะนี้ก็คือ ในกรณีที่ค่าหลักที่ใช้ในการแทนถึงวัตถุหรือสิ่งของใดๆ นั้นเป็นค่าที่มีการเปลี่ยนแปลงด้วย จะทำให้เกิดปัญหาของการแทนค่าถึงวัตถุตัวใดตัวหนึ่ง เนื่องจากค่าที่ใช้ในการอ้างอิงนั้นเกิดการเปลี่ยนแปลงไปจากเดิม และต้องแก้ปัญหาด้วยการแก้ไขค่าหลักที่ใช้ในการอ้างอิงนี้ทุกตัว เมื่อมีการเปลี่ยนแปลงเกิดขึ้นเพื่อให้การอ้างอิงถึงค่าของสิ่งใดสิ่งหนึ่งในฐานข้อมูลเป็นไปได้เหมือนเดิม

ในกรณีของระบบฐานข้อมูลเชิงวัตถุ จะแก้ปัญหานี้ด้วยการแทนค่าที่ใช้ในการอ้างอิงถึงค่าของสิ่งใดสิ่งหนึ่ง ซึ่งจะเรียกค่านี้ว่า “ตัวชี้วัตถุ” ลักษณะค่านี้จะแตกต่างไปจากค่าที่เก็บอยู่ในฐานข้อมูลค่าอื่นๆ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ก็คือ จะมีค่าที่ไม่ซ้ำกันเพื่อที่ระบบจัดการฐานข้อมูลใช้ค่าเหล่านี้ในการอ้างอิงวัตถุตัวใดตัวหนึ่งที่เก็บอยู่ในฐานข้อมูลเป็นหลัก ซึ่งในการอ้างอิงนี้เราอาจพบว่าจะมีวัตถุสองตัวใดๆ ที่มีค่าเหมือนกัน แต่มีการอ้างอิงโดยค่าตัวชี้วัตถุคนละตัวก็ได้ ซึ่งค่านี้ระบบจัดการฐานข้อมูลจะใช้ในการอ้างอิงถึงตัววัตถุตัวนั้น เป็นแบบถาวร คือค่าที่ใช้ในการอ้างอิงเหล่านี้จะถูกสร้างและถูกทำลายโดยตัวระบบจัดการฐานข้อมูลเอง และไม่สามารถทำการเปลี่ยนแปลงค่าต่างๆ เหล่านี้ได้ ซึ่งก็จะแตกต่างจากค่าอื่นๆ ที่อยู่ในฐานข้อมูล ส่วนความสามารถที่ผู้ใช้ระบบฐานข้อมูลจะนำค่าเหล่านี้ไปใช้ในการอ้างอิงได้หรือไม่นั้น ซึ่งก็แตกต่างกันไปในแต่ละระบบ

2.3.3 การสืบทอดคุณสมบัติ (Inheritance)

ในการสร้างซอฟต์แวร์ โดยปกติมักต้องมีการเปลี่ยนแปลงข้อกำหนดหรือเพิ่มหน้าที่ใหม่ๆ อยู่เสมอ การแก้ไขโปรแกรมให้ทำงานตามข้อกำหนดใหม่หรือหน้าที่ที่เพิ่มขึ้น โดยต้องให้โปรแกรมทำงานหน้าที่เดิมได้ถูกต้องด้วย ซึ่งเป็นงานที่ยุ่งยากและใช้เวลามาก บางทีอาจมากกว่าสร้างโปรแกรมขึ้นมาใหม่ทั้งหมดเลยก็ได้ ภาษาและฐานข้อมูลเชิงวัตถุสามารถแก้ปัญหานี้ โดยใช้กลไกการนำคลาสที่ถูกสร้างไว้แล้ว มาเพิ่มหรือเปลี่ยนแปลงสมาชิกบางตัวเป็นคลาสใหม่ที่เหมาะสมกับหน้าที่ใหม่ที่ต้องการ คลาสใหม่ที่ได้สมาชิกของคลาสเดิมมาโดยอาจจะมีสมาชิกใหม่เพิ่มขึ้น หรือสมาชิกเดิมของคลาสบางตัวถูกเปลี่ยนไปจากเดิม คลาสเดิมก็ยังใช้งานได้เช่นเดิม แต่คลาสใหม่สามารถทำงานในหน้าที่เดิม และทำงานในหน้าที่ใหม่ได้ด้วย เราเรียกกลไกการสร้างคลาสอย่างนี้ว่า “การสืบทอดคุณสมบัติ” ซึ่งเป็นกลไกหนึ่งของภาษาเชิงวัตถุ

2.3.4 โพลีมอร์ฟิซึม

ภาษาเชิงวัตถุมีกลไก Inheritance และ dynamic binding ทำให้เราสามารถสร้างเมธอดที่มีลักษณะเป็นโพลีมอร์ฟิซึมเมธอด คือ เมธอดที่สามารถจัดการกับอินสแทนซ์ (instance) ของคลาสที่ต่างกันได้ หากเป็นคลาสที่ขยายมาจากคลาสหนึ่งที่กำหนดให้เป็นพารามิเตอร์ของเมธอดนั้น (หนึ่งรูป หลายพฤติกรรม) โพลีมอร์ฟิซึมช่วยให้เราไม่ต้องสร้างเมธอดหนึ่งสำหรับจัดการกับอินสแทนซ์ของคลาสหนึ่ง แต่ใช้เพียงเมธอดเดียวจัดการกับอินสแทนซ์ของคลาสต่างๆ ที่ขยายจากคลาสแม่คลาสหนึ่ง ทำให้เราสามารถนำโปรแกรมที่ทำงานกับคลาสแม่มาใช้กับคลาสลูกได้

แนวความคิดของโพลีมอร์ฟิซึมนั้น ไม่ได้จำกัดอยู่กับรูปแบบของ method เท่านั้น แต่อาจขยายเป็นอาร์เรย์ที่เก็บอินสแทนซ์ของคลาสที่ต่างกัน เพื่อให้สามารถใช้ประโยค for จัดการกับสมาชิกในอาร์เรย์นั้นด้วยโปรแกรมเดียวกัน แต่ผลที่ได้ขึ้นกับสมาชิกนั้นเป็นอินสแทนซ์ของคลาสใด

โพลีมอร์ฟิซึมไม่ใช่การทำ overloading ที่จริงแล้วทั้งสองมีเพียงจุดมุ่งหมายคล้ายกัน คือ ต้องการสร้างเมธอดที่ถูกเรียกด้วยอาร์กิวเมนต์ (argument) หลายแบบได้ นั่นคือทางผู้เรียก (caller) เมธอด เราอาจกล่าวได้ว่าโพลีมอร์ฟิซึมคล้ายกับ overloading คือสามารถเรียกเมธอดหนึ่งด้วยอาร์กิวเมนต์ที่แตกต่างกันไปในการเรียกแต่ละครั้ง แต่ด้านผู้ถูกเรียก (callee) โพลีมอร์ฟิซึมจะแตกต่างจาก overloading อย่างมากคือ ถ้าเป็นโพลีมอร์ฟิซึมจะมีเพียงเมธอดเดียว โดยอาศัย Inheritance และ dynamic binding เพื่อให้รับพารามิเตอร์ที่ต่างคลาสได้ และเกิดการทำงานที่ต่างกันไป แต่สำหรับ overloaded method ต้องมีหลาย ๆ เมธอดสำหรับเรียกแต่ละแบบ และอาศัยคอมไพเลอร์ทำการผูกชื่อว่า ตรงที่เรียกเมธอดที่มีอาร์กิวเมนต์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

แบบนั้นจะให้เมธอดใดถูกกระทำ จำนวนของ overloaded method ถูกกำหนดไว้ตั้งแต่คอมไพล์โปรแกรม และเพิ่มไม่ได้ในภายหลัง แต่เมธอดแบบโพลีมอร์ฟิซึม นั้นสามารถรับพารามิเตอร์ที่แตกต่างกันได้ไม่จำกัด เพราะสามารถขยายคลาสได้โดยไม่ต้องแก้ไขหรือคอมไพล์โปรแกรมเดิม

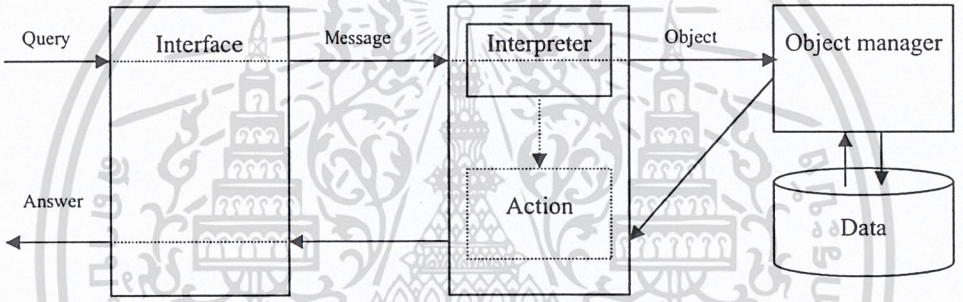
2.4 สถาปัตยกรรมของระบบจัดการฐานข้อมูลเชิงวัตถุ (The Architecture of OODBMS)

ระบบจัดการฐานข้อมูลเชิงวัตถุจะต้องประกอบไปด้วย (รูปที่ 2 - 1)

ส่วนติดต่อกับผู้ใช้ (Interface)

ตัวแปลภาษา (Interpreter) ประกอบไปด้วย message-processing subsystem และ transaction-processing subsystem

ตัวจัดการวัตถุ (Object Manager) ประกอบไปด้วย object subsystem และ storage subsystem



รูปที่ 2 – 1 สถาปัตยกรรมของระบบจัดการฐานข้อมูลเชิงวัตถุ

2.5 แบบจำลองฐานข้อมูลเชิงวัตถุ (Object Oriented Database Model)

แบบจำลองฐานข้อมูลเชิงวัตถุเป็นวิธีการใหม่ที่ใช้ในการออกแบบฐานข้อมูล เพื่อให้สามารถจัดเก็บข้อมูลที่มีลักษณะซับซ้อน (Complex Data Type) ได้ เช่น ข้อมูลภาพและเสียง รวมไปถึงข้อมูลที่เป็น Abstract Data Type อีกด้วย ซึ่งรองรับกับการออกแบบระบบงานเชิงวัตถุ (Object-Oriented Design) โดยในการออกแบบงานใดๆ ในลักษณะเชิงวัตถุนี้จะใช้เทคนิคของแบบจำลองเชิงวัตถุ (Object-Oriented Modeling Technique : OMT) ตั้งแต่การวิเคราะห์ระบบ, การออกแบบระบบงาน, การออกแบบฐานข้อมูล รวมไปถึงการพัฒนาระบบด้วย ซึ่งสามารถบ่งบอกถึงขั้นตอนการทำงานทั้งหมดภายในระบบได้ตามแผนภาพที่นำมาใช้ในการออกแบบ

แบบจำลองฐานข้อมูลเชิงวัตถุที่นำมาทำการศึกษาสำหรับงานวิจัยนี้คือ UML ซึ่งเป็นแบบจำลองที่มีการรวบรวมมาจากหลักการของ Booch, Rumbaugh และ Jacobson โดยมีการกำหนดเป็นมาตรฐานในการใช้งานจาก OMG (Object Management Group) ตั้งแต่ราวต้นปี ค.ศ. 1997 และเป็นที่ยอมรับและมีการใช้งานกันอย่างแพร่หลายของแบบจำลองเชิงวัตถุ ตั้งแต่การวิเคราะห์ระบบการออกแบบระบบงาน การออกแบบฐานข้อมูล รวมไปถึงการพัฒนาระบบด้วย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.5.1 ส่วนประกอบพื้นฐานและสัญลักษณ์ของแบบจำลองฐานข้อมูลเชิงวัตถุ

คลาส

คลาส คือ การบอกลักษณะกลุ่มของออบเจ็กต์ซึ่งมีคุณสมบัติ, พฤติกรรม, ลักษณะของความสัมพันธ์กับออบเจ็กต์อื่นๆ และโครงสร้างที่เหมือนกัน ประเภทของคลาสอาจจะมีการกำหนดได้หลายลักษณะดังนี้ คลาสที่เป็นสิ่งของหรือวัตถุที่มีอยู่จริง เช่น คน, บริษัท, รถยนต์ เป็นต้น หรือคลาสที่อยู่ในลักษณะของเอนติตีระดับแนวคิด (Conceptual Entity) เช่น ตารางสอน, ตารางการบิน, สมการ เป็นต้น นอกจากนี้ยังมีคลาสที่กำหนดไว้เพื่อใช้เป็นเครื่องมือ เช่น แอเรย์ (Array), ไบนารี่ทรี (Binary Tree) เป็นต้น องค์ประกอบของคลาสสามารถแบ่งออกได้ 3 ส่วนคือ

ชื่อคลาส เป็นการระบุชื่อคลาส โดยทุกๆ คลาสต้องมีการกำหนดชื่อ

แอตทริบิวต์ เป็นการระบุคุณสมบัติต่างๆ ของคลาส โดยที่แต่ละคลาสอาจจะมีแอตทริบิวต์หรือไม่มีก็ได้

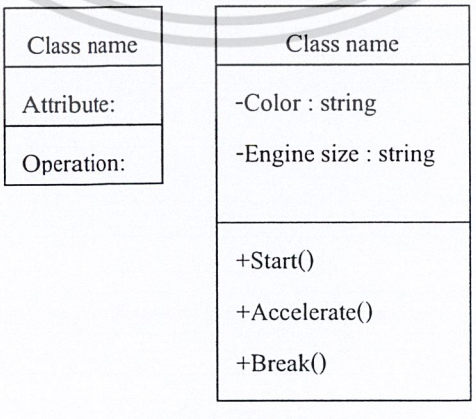
โอเปอเรชั่นหรือเมธอด เป็นการกำหนดฟังก์ชัน หรือขั้นตอน (Procedure) ซึ่งจะถูกกระทำจากออบเจ็กต์ที่เป็นสมาชิกของคลาสนั้น ๆ

จากการที่คลาสมีคุณสมบัติของการเอนแคปซูเลชั่น (Encapsulation) ทำให้คลาสสามารถแบ่งระดับของการเข้าถึงองค์ประกอบต่างๆ ภายในคลาสที่กล่าวมาแล้วได้โดยจะมีการกำหนดสัญลักษณ์เพื่อแสดงคุณสมบัติดังกล่าวอยู่หน้าองค์ประกอบต่างๆ ของคลาส การเอนแคปซูเลชั่นสามารถแบ่งออกได้ 3 ระดับคือ

สัญลักษณ์ + (Public) หมายถึงองค์ประกอบที่มีสัญลักษณ์ดังกล่าวจะสามารถให้คลาสอื่นเข้าถึงและเรียกใช้งาน

สัญลักษณ์ – (Private) หมายถึงองค์ประกอบที่มีสัญลักษณ์ดังกล่าวจะสามารถเรียกใช้หรืออ้างถึงได้เพียงคลาสเดียวเท่านั้น และไม่สามารถอ้างถึงหรือเรียกใช้จากคลาสลูกได้

สัญลักษณ์ # (Protected) หมายถึงองค์ประกอบที่มีสัญลักษณ์ดังกล่าวจะสามารถเรียกใช้หรืออ้างถึงได้เพียงคลาสเดียว แต่อนุญาตให้คลาสลูกสามารถใช้งานได้ด้วย



รูปที่ 2 - 2 แสดงสัญลักษณ์และตัวอย่างของคลาส

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากรูปที่ 2 - 2 จะแสดงโดยใช้ช่องสี่เหลี่ยมดังรูป โดยช่องแรกจะระบุชื่อคลาสโดยจะต้องมีการกำหนดทุกครั้ง ช่องที่สองจะเป็นการกำหนดแอตทริบิวต์ของคลาสโดยรูปแบบของแอตทริบิวต์จะใช้รูปแบบดังนี้

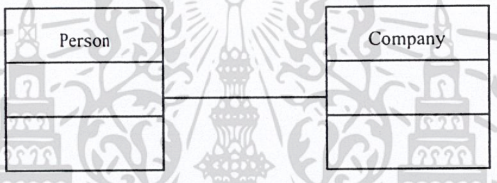
```
Attribute_Name : Attribute_Type = Initial_Value
```

และช่องสุดท้ายเป็นการกำหนดโอเปอเรชั่นของคลาส ซึ่งจะใช้รูปแบบในการเขียนดังนี้

```
Operation_Name(Argument_Name:
Argument_Type= Default_Value,..):Return_Type
```

ความสัมพันธ์แบบแอสโซซิเอชั่น (Association Relationships)

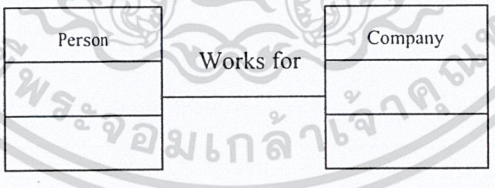
เป็นการแสดงความสัมพันธ์ระหว่างคลาสของออบเจ็กต์ โดยความสัมพันธ์ดังกล่าวใช้เชื่อมออบเจ็กต์ที่เป็นออบเจ็กต์อิสระของคลาส 2 คลาส สัญลักษณ์ที่ใช้คือเส้นตรงเชื่อมระหว่างคลาส 2 คลาส ดังรูปที่ 2 – 3



รูปที่ 2 – 3 แสดงสัญลักษณ์ความสัมพันธ์แบบแอสโซซิเอชั่น

ชื่อแอสโซซิเอชั่น (Association Naming)

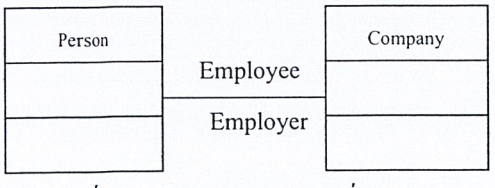
เป็นการบอกถึงรายละเอียดของความสัมพันธ์ระหว่างคลาส โดยจะกำหนดชื่อแอสโซซิเอชั่นอยู่ตรงกลางของเส้นโดยชื่อดังกล่าวมักจะอยู่ในรูปของคำกริยา



รูปที่ 2 – 4 แสดงชื่อของแอสโซซิเอชั่น

ชื่อบทบาท (Role Naming)

ในแต่ละปลายของแอสโซซิเอชั่นอาจมีการกำหนดชื่อของบทบาทได้ โดยชื่อดังกล่าวนั้นจะอธิบายลักษณะของคลาสหนึ่งๆ มองเห็นคลาสฝั่งตรงข้ามอย่างไร ชื่อบทบาทนี้มักจะแสดงอยู่ในรูปของ noun ดังรูปที่ 2-5



รูปที่ 2 – 5 แสดงสัญลักษณ์ของชื่อบทบาท

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

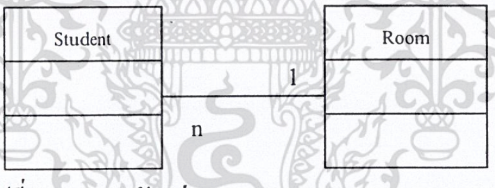
Multitplicity of Associations

ในแต่ละบทบาทของแอสโซซิเอชั่นนั้นมีค่าได้หลายค่า เพื่อระบุจำนวนออบเจ็กต์ที่อยู่ในคลาสที่กำหนดนั้นๆ ว่า มีการเชื่อมต่อกับออบเจ็กต์ของอีกคลาสหนึ่ง โดยการกำหนดขอบเขตสามารถใช้สัญลักษณ์ดังนี้

สัญลักษณ์	ความหมาย
1	มีค่าได้เพียงหนึ่งออบเจ็กต์เท่านั้น
0..1	มีค่าได้เพียงศูนย์หรือหนึ่งออบเจ็กต์
M..N	มีค่าจาก M ถึง N ออบเจ็กต์ (M และ N คือจำนวนนับ)
*	มีค่าจากศูนย์ถึงจำนวนเต็มบวกใดๆ
0..*	มีค่าจากศูนย์ถึงจำนวนเต็มบวกใดๆ
1..*	มีค่าจากศูนย์ถึงจำนวนเต็มบวกใดๆ

ตารางที่ 2 – 1 แสดงสัญลักษณ์ที่ใช้แทนความหมายของ Multiplicity of Associations

การระบุ Multiplicity of Associations เปรียบเสมือนการกำหนดกฎข้อบังคับความถูกต้องของข้อมูลในลักษณะที่กล่าวถึงการมีอยู่ของข้อมูล ดังภาพแสดงตัวอย่างของ Multiplicity values ในรูปที่ 2 – 6



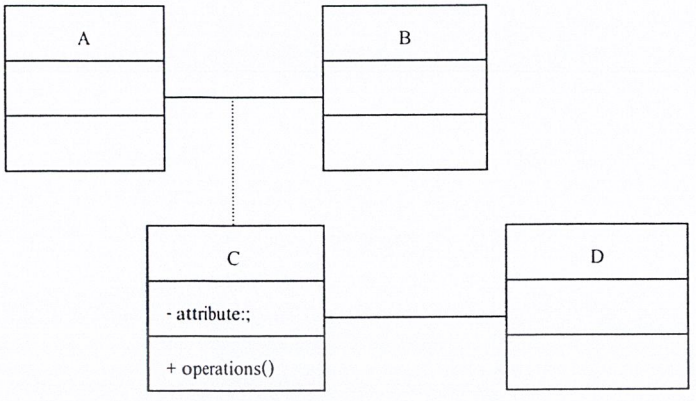
รูปที่ 2 – 6 แสดงตัวอย่างของ Multiplicity of Associations

จากภาพจะเห็นว่านักเรียน 1 คนจะอยู่ในห้องได้ห้องเดียว แต่ห้องหนึ่งห้องสามารถบรรจุนักเรียนได้หลายคน เป็นต้น

2.5.2 คลาสแอสโซซิเอชั่น (Association classes)

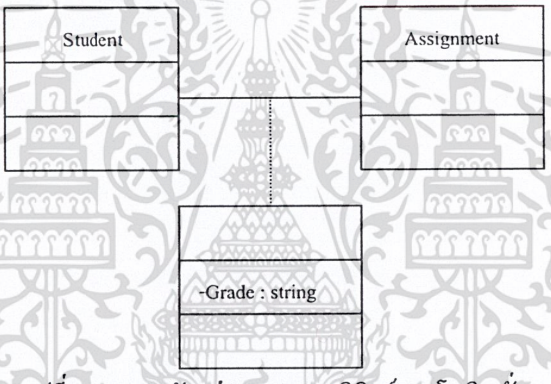
ความสัมพันธ์แบบแอสโซซิเอชั่นนั้นสามารถที่จะนำเสนอเป็นคลาสได้ เพื่อที่จะสามารถเพิ่มแอตทริบิวต์และโอเปอเรชั่นที่เกี่ยวข้องกับความสัมพันธ์นั้นได้ คลาสประเภทนี้จะมีคุณสมบัติเหมือนกับคลาสอื่นๆ ได้ด้วย สัญลักษณ์ที่ใช้ในการแสดงคลาสแอสโซซิเอชั่นจะใช้เป็นลักษณะของเส้นประเพื่อเชื่อมต่อไปยังคลาสดังกล่าว สามารถแสดงได้รูปที่ 2 - 7

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2 – 7 แสดงคลาสแอสโซซิเอชั่น

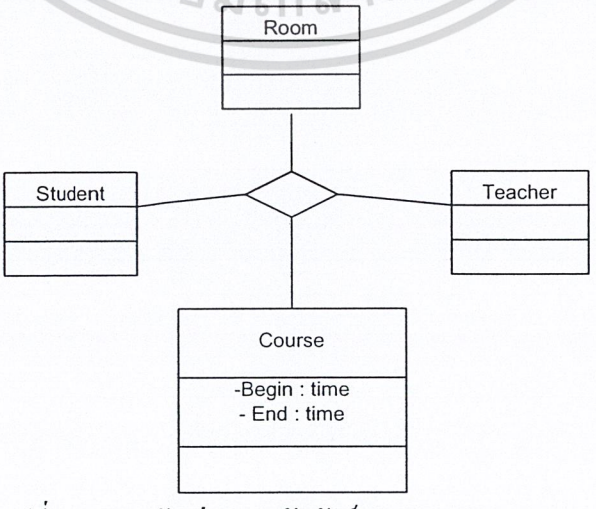
หากคลาสแอสโซซิเอชั่นที่ไม่มีความสัมพันธ์เชื่อมต่อกับคลาสอื่นๆ เราอาจเรียกว่า แอตทริบิวต์แอสโซซิเอชั่น(attributed association) ได้ ดังแสดงในรูปที่ 2 – 8



รูปที่ 2 - 8 แสดงตัวอย่างของแอตทริบิวต์แอสโซซิเอชั่น

N-ary Associations

เป็นการแสดงความสัมพันธ์แบบแอสโซซิเอชั่นระหว่างคลาสที่มีมากกว่า 2 คลาสขึ้นไป โดยจะใช้สัญลักษณ์ diamond เป็นตัวเชื่อมความสัมพันธ์ ดังแสดงได้ในรูปที่ 2 - 9

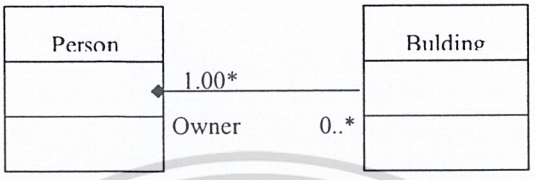


รูปที่ 2 - 9 แสดงตัวอย่างความสัมพันธ์แบบ N-ary Associations

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

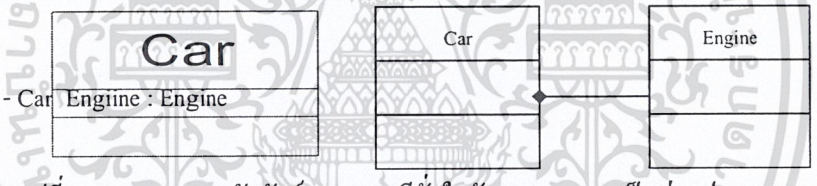
ความสัมพันธ์แบบเอกกรีเกชั่น (Aggregation Relationships)

เป็นการแสดงความสัมพันธ์ในลักษณะที่ไม่สามาตรกล่าวคือ ปลายด้านหนึ่งของความสัมพันธ์จะมีบทบาทมากกว่าปลายอีกด้านหนึ่ง ดังนั้นจึงมักมีการกำหนดบทบาทเพียงด้านเดียวเท่านั้น สัญลักษณ์ที่ใช้จะเพิ่มสี่เหลี่ยมขนมเปียกปูนที่ปลายด้านหนึ่งของเส้น ดังแสดงได้ในรูปที่ 2 – 10



รูปที่ 2 - 10 แสดงความสัมพันธ์แบบแอกกรีชั่น

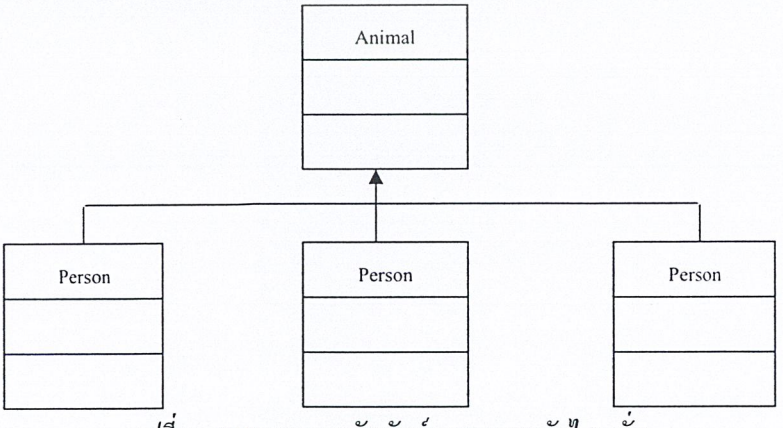
โดยทั่วไปมักใช้ความสัมพันธ์แบบแอกกรีชั่นในการบ่งบอกถึงการเป็นส่วนประกอบ (Composition) หรือใช้แทนความหมายของคำว่า “is a part of” หรือ “is made up of “ โดยแสดงให้เห็นว่าออบเจ็กต์ของคลาสที่ประกอบขึ้นมาจากออบเจ็กต์อื่นๆ ที่อยู่ในคลาสอื่น ซึ่งจะใช้สัญลักษณ์เป็นขนมเปียกปูนสีดำแทน ดังแสดงในรูปที่ 2 – 11



รูปที่ 2 - 11 แสดงความสัมพันธ์แบบแอกกรีชั่นในลักษณะของการเป็นส่วนประกอบ

ความสัมพันธ์แบบเจนเนอรัลไลเซชั่น (Generalization Relationship)

เป็นความสัมพันธ์ที่ใช้ในการแสดงว่าคลาสสามารถที่จะบ่งบอกประเภทของสมาชิกของคลาสอีกคลาสหนึ่งได้ ความสัมพันธ์แบบนี้จะใช้ในการแทนความหมายของ “is a” หรือ “is a kind of” ซึ่งจะใช้สัญลักษณ์ในการแทนความสัมพันธ์โดยเป็นลูกศรชี้จากคลาสที่มีรายละเอียดเฉพาะมากกว่าไปยังคลาสทั่วไป โดยจะเรียกคลาสทั่วไปว่า ซุปเปอร์คลาส (Superclass) และคลาสที่มีรายละเอียดมากกว่าว่า ซับคลาส (Subclasses) ดังแสดงได้ในรูปที่ 2 – 12

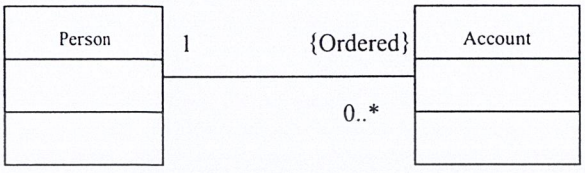


รูปที่ 2 - 12 แสดงความสัมพันธ์แบบเจนเนอรัลไลเซชั่น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

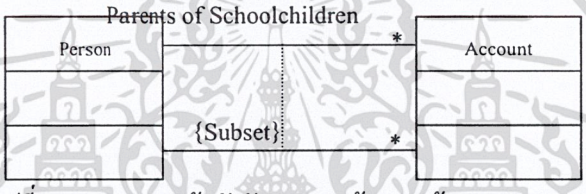
กฎข้อบังคับความถูกต้องของข้อมูล (Integrity Constriants)

แบบจำลองฐานข้อมูลเชิงวัตถุนั้นจะมีการกำหนดสิ่งที่ใช้ในการควบคุมความถูกต้องของข้อมูลบนแบบจำลองได้โดยจะกำหนดกฎข้อบังคับความถูกต้องของข้อมูลไว้ภายในเครื่องหมาย {} และมักจะนำมาวางแทนที่ชื่อของบทบาทดังแสดงได้ในรูปที่ 2 - 13



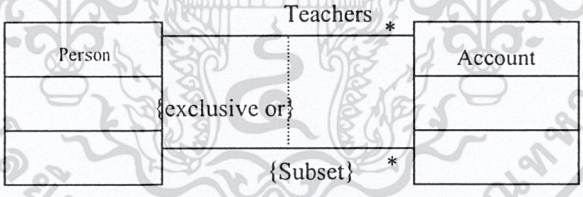
รูปที่ 2 - 13 แสดงกฎข้อบังคับความถูกต้องของข้อมูลแบบ Ordered

แสดงกฎข้อบังคับความถูกต้องของข้อมูล {Ordered} โดยเป็นการบอกลักษณะของออบเจ็กต์ที่อยู่ภายในคลาส Account ซึ่งเป็นเซตของออบเจ็กต์เหล่านั้นจะมีการเรียงลำดับของออบเจ็กต์ด้วย



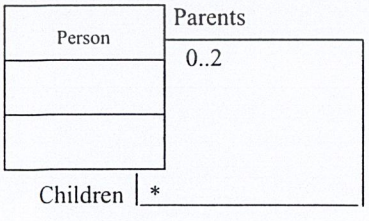
รูปที่ 2 - 14 แสดงกฎข้อบังคับความถูกต้องของข้อมูลแบบ subset

จากภาพแสดงกฎข้อบังคับความถูกต้องเพื่อระบุว่ากลุ่มของออบเจ็กต์ในคลาสเป็นซับเซตกับกลุ่มของออบเจ็กต์อีกกลุ่มหนึ่งในคลาสเดียวกัน



รูปที่ 2 - 15 แสดงกฎข้อบังคับความถูกต้องของข้อมูลแบบ exclusive or

จากภาพแสดงให้เห็นว่ามีการกำหนดกฎข้อบังคับความถูกต้อง เพื่อระบุว่าออบเจ็กต์ที่ทำการกำหนดนั้น สามารถมีความสัมพันธ์ได้เพียงความสัมพันธ์เดียวเท่านั้นจากความสัมพันธ์ที่มีทั้งหมด



รูปที่ 2 - 16 แสดงกฎข้อบังคับความถูกต้องแบบ Reflexive association

จากภาพแสดงให้เห็นถึงความสำคัญในการระบุชื่อบทบาทสำหรับความสัมพันธ์แบบ reflexive association ซึ่งใช้ในการควบคุมความถูกต้องของข้อมูลที่ปรากฏอยู่

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.6 การเปรียบเทียบแบบจำลองฐานข้อมูลเชิงวัตถุกับแบบจำลองฐานข้อมูลเชิงสัมพันธ์

การเปรียบเทียบในที่นี้จะนำเสนอเกี่ยวกับการจัดการข้อมูลในลักษณะที่บ่งบอกถึงความสามารถของแบบจำลองข้อมูลที่รองรับกับความต้องการที่มีการเปลี่ยนแปลงอยู่ตลอดเวลา ซึ่งสามารถสรุปได้ดังนี้คือ

2.6.1 ชนิดของข้อมูล (Data Types)

ในระบบจัดการฐานข้อมูลเชิงสัมพันธ์จะทำการจัดการเกี่ยวกับโครงสร้างของข้อมูลในรูปแบบรีเลชั่นเท่านั้น โดยคำจำกัดความของแอตทริบิวต์ของรีเลชั่นที่ผ่านการทำนอร์มอลไรซ์มาแล้วนั้นจะไม่สามารถแบ่งแยกออกเป็นส่วนย่อยๆ ได้อีก และมีรูปแบบชนิดของข้อมูลอยู่ในลักษณะอย่างง่าย เช่น Integer, Real, String, Date เป็นต้น โอเปอเรชั่นบนรีเลชั่นได้ถูกจำกัดเพื่อที่จะทำการสืบค้น หรือปรับปรุงทัปเปิลที่ถูกระบุโดยค่าของแอตทริบิวต์เท่านั้น เหตุผลหลักในงานลักษณะนี้เพื่อที่จะขยายความสามารถของรูปแบบของแบบจำลองเชิงสัมพันธ์ขณะที่ยังคงลักษณะที่ให้ง่ายต่อการใช้อยู่

สำหรับฐานข้อมูลเชิงวัตถุนั้นจะทำการจัดเก็บคำจำกัดความของคลาสและตัวอย่างของคลาสเหล่านั้น คลาสนั้นก็เปรียบเสมือนกับรีเลชั่นที่ใช้งานบนฐานข้อมูลเชิงสัมพันธ์ แต่มีลักษณะพิเศษที่สำคัญซึ่งเพิ่มขึ้นมาก็คือ คลาสจะทำการเอนแคปซูเลชั่นพฤติกรรมทั้งหมดของออบเจ็กต์โดยการรวมโอเปอเรชั่นและโครงสร้างของข้อมูลไว้ด้วยกัน คุณสมบัติของคลาสนั้นไม่จำเป็นต้องเป็นชนิดของข้อมูลในรูปแบบอย่างง่าย แต่สามารถอ้างถึงคลาสอื่นๆ ที่มีความซับซ้อนใด ๆ ได้ ในระบบเชิงวัตถุสามารถที่จะทำการกำหนดขอบเขตของคลาสตามความถี่ของชนิดข้อมูลที่มีการใช้งาน ผู้ใช้สามารถปรับปรุงคลาสเหล่านี้ได้ตามความต้องการของโปรแกรมประยุกต์ที่ใช้งานนั้น ๆ โดยใช้คุณสมบัติที่มีอยู่อย่างง่ายเช่น การจัดกลุ่ม (Generality) และ การสืบทอด

2.6.2 ความแน่นอนของข้อมูล (Data Integrity)

ฐานข้อมูลเชิงสัมพันธ์ มีความต้องการในการที่จะควบคุม Referential Integrity ของข้อมูลแต่บ่อยครั้งที่จุดมุ่งหมายนี้มักจะเป็นหน้าที่ของผู้พัฒนาโปรแกรมประยุกต์ ในบางระบบนั้นกำหนดเงื่อนไขสำหรับกฎข้อบังคับความถูกต้องของข้อมูลในเรื่องของ Referential Integrity แยกออกมาเฉพาะ ในส่วนนี้นั้นเป็นความรับผิดชอบของผู้พัฒนาฐานข้อมูลที่จะแน่ใจได้ว่ากฎข้อบังคับความถูกต้องทั้งหมดได้ถูกกำหนดรวมอยู่เรียบร้อยแล้ว โดยที่ระบบจะทำการควบคุมกฎข้อบังคับความถูกต้องเหล่านั้นในช่วงที่ระบบทำงาน แต่การทำลักษณะนี้เป็นการเสียเวลาและสิ้นเปลืองทรัพยากรค่อนข้างมาก

ตัวอย่างของปัญหาที่เกิดขึ้น กับฐานข้อมูลเชิงสัมพันธ์เช่น ฐานข้อมูลสัมพันธ์ไม่สามารถที่จะแสดงความสัมพันธ์ซึ่งเป็นความสัมพันธ์แบบหนึ่งต่อหนึ่งหน่วย (one-to-one relationships) หรือความสัมพันธ์แบบหนึ่งต่อหลายหน่วย (one-to-many relationships) ได้ กฎข้อบังคับดังกล่าวจะถูกสร้างและจัดเก็บไว้ในโปรแกรมประยุกต์ (Application Code) ที่ใช้งานร่วมกับฐานข้อมูลเชิงสัมพันธ์ แต่โดยทั่วไปแล้วโค้ดที่สร้างขึ้นนั้นไม่ได้ถูกใช้ร่วมกันระหว่างระหว่างหลายๆ โปรแกรมประยุกต์ ดังนั้นจึงเป็นการยากที่จะแน่ใจได้ว่าข้อมูลดังกล่าวจะถูกปรับปรุงให้คงสภาพความถูกต้องได้ตลอดเวลา

ในทางตรงกันข้ามแบบจำลองเชิงวัตถุนั้น คลาสจะมีการกำหนดในลักษณะของ data abstraction และ abstraction นี้จะทำการรวบรวมรายละเอียดของโอเปอเรชั่น (methods) ซึ่งสามารถนำไปใช้งานกับ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตัวอย่างของคลาสได้ โดยนิยามของฐานข้อมูลในความหมายของ Abstraction ดังกล่าวมีความเป็นอิสระของข้อมูลสูงมาก นั่นคือเป็นไปได้ที่จะทำการเปลี่ยนวิธีดำเนินการในคลาสโดยที่ปราศจากผลกระทบต่อคลาส หรือทรานแซคชั่น (Transactions) ที่มีการใช้ของ abstract นั้นๆ ดังนั้นโอเปอเรชั่นซึ่งถูกกำหนดขึ้นโดยผู้ใช้สามารถสร้างและจัดเก็บไว้ในฐานข้อมูลโอเปอเรชั่นหนึ่งๆ บนออบเจ็กต์สามารถเกิดขึ้นได้หลายๆ ฐานข้อมูล โดยที่โอเปอเรชั่นและทุกๆ ทรานแซชั่นที่ต้องขอโอเปอเรชั่นจะใช้โค้ดเหมือนกัน ซึ่งโค้ดดังกล่าวจะถูกเก็บอยู่ในรูปของกระบวนการ (Procedure) โดยจัดเก็บไว้ในคลาสจึงถือว่าเป็นข้อได้เปรียบอย่างมากเกี่ยวกับการดูแลความถูกต้องของข้อมูล เนื่องจากการตรวจสอบความสอดคล้องอาจถูกรวมเข้าไว้ในกระบวนการและจัดเป็น โอเปอเรชั่นหนึ่งเหมือนกับเป็นส่วนหนึ่งของฐานข้อมูล ดังนั้นส่วนใหญ่โค้ดของโปรแกรมประยุกต์จึงอยู่ภายใต้การควบคุมของผู้ดูแลฐานข้อมูล (Database Administrator) และสามารถถูกจัดการโดยยูทิลิตี้ของการจัดการฐานข้อมูลทั้งหมด

ความแน่นอนของเอนติตี (Entity Integrity) จะถูกจัดการด้วยวิธีการที่แตกต่างกันในระบบเชิงวัตถุ ทุกๆ ออบเจ็กต์มี Unique Identity และออบเจ็กต์อื่นๆ สามารถอ้างถึง identity นั้นได้ ออบเจ็กต์ยังคงรักษา identity นั้นไว้ได้ตลอดไม่ว่าออบเจ็กต์จะมีการเปลี่ยนสภาวะไปอย่างไรก็ตาม ซึ่งจะตรงข้ามกับแบบจำลองเชิงสัมพันธ์ ซึ่งคุณสมบัติของเอนติตีจะต้องมีอย่างเพียงพอในการที่จะแยกแยะความแตกต่างของเอนติตีหนึ่งๆ ออกจากเอนติตีอื่น ๆ ได้ ซึ่งก็คือจะต้องมีอย่างน้อยหนึ่งแอตทริบิวต์ขึ้นไปซึ่งค่าของแอตทริบิวต์นั้นต้องมีเพียงหนึ่งค่าและต้องไม่มีการเปลี่ยนแปลงคุณสมบัติใดๆ ซึ่งมักจะไม่ตรงกับสภาพความเป็นจริง และมักต้องทำการกำหนด Artificial identifiers สำหรับเอนติตีในฐานข้อมูลเชิงสัมพันธ์เสมอ

2.6.3.วิวัฒนาการของโครงสร้าง (Schema Evolution)

ระบบจัดการฐานข้อมูลมีแนวโน้มเพื่อที่จะวิวัฒนาการจำนวนชนิดของออบเจ็กต์พื้นฐานโดยรอบซึ่งเป็นรูปแบบของหลักการของระบบ ขณะที่วิวัฒนาการของระบบนั้น มุมมองของข้อมูลแนวโน้มใหม่ถูกเพิ่มโดยการควบคุมหรือการขยายหลักการของออบเจ็กต์ที่มีอยู่ และโปรแกรมประยุกต์แนวใหม่ในมุมมองเหล่านั้นจะถูกสร้างโดยใช้โมดูลของโค้ดที่มีอยู่แล้วเป็นพื้นฐาน ส่วนระบบจัดการฐานข้อมูลเชิงสัมพันธ์มีแนวโน้มกำหนดขอบเขตอย่างมากสำหรับการขยายหรือการเปลี่ยนแปลงของโครงสร้างของข้อมูลที่มีอยู่ ตัวอย่างเช่น ระบบธุรกิจเชิงสัมพันธ์อนุญาตให้มีการเปลี่ยนแปลงของการสร้าง การลบของรีเลชั่นอย่างแบบธรรมดา และการเพิ่มคอลัมน์ใหม่ของรีเลชั่นการเปลี่ยนชนิดของโดเมนมักจะก่อให้เกิดการสร้างใหม่ของรีเลชั่นที่ต้องการอยู่เสมอ และจะต้องทำการเปลี่ยนแปลงโปรแกรมประยุกต์ที่จะทำการใช้งานกับรีเลชั่นนั้น ๆ ด้วย สิ่งเหล่านี้แสดงให้เห็นความจริงที่ว่าโครงสร้างของข้อมูลประกอบขึ้นมาจากรูปแบบของฐานข้อมูล (Database Schema) และโปรแกรมประยุกต์ซึ่งมีความเกี่ยวข้องกันน้อยมาก

ความเกี่ยวข้องกันอย่างมากระหว่างโปรแกรมและข้อมูลในแบบจำลองเชิงวัตถุแสดงถึงขอบเขตที่กว้างขวางสำหรับวิวัฒนาการของโครงสร้างตลอด โดยการขยายและการกำหนดใหม่ของโครงสร้างข้อมูลที่มีอยู่แล้วและยังแสดงให้เห็นถึงประสิทธิภาพของการนำมาใช้ใหม่ของโค้ดของโปรแกรมประยุกต์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

อีกด้วย ความสมบูรณ์ของแบบจำลองเชิงวัตถุอนุญาตรูปแบบของการวิวัฒนาการโครงสร้างได้มีการกำหนดขึ้นอย่างตายตัวและมีเหตุผล

2.7 ประโยชน์จากการใช้ฐานข้อมูลเชิงวัตถุ

ฐานข้อมูลเชิงวัตถุได้ผสมความสามารถทางด้านความเร็วของฐานข้อมูลบนระบบเครือข่ายเข้ากับความยืดหยุ่นของฐานข้อมูลเชิงสัมพันธ์ และเพิ่มความสามารถในการทำงานให้กับการทำหน่วยการทำงานทรานแซคชั่นระยะไกล โดยการใช้ persistent lock และการทำเวอร์ชัน (version) แบบอัตโนมัติ ซึ่งเป็นพื้นฐานให้ฐานข้อมูลเชิงวัตถุสามารถทำการประมวลผลแบบกระจายได้ นอกจากนี้การทำงานด้วยวิธีการส่งผ่านข้อมูลที่ใช้อยู่ในการโปรแกรมเชิงวัตถุนั้น ยังช่วยอธิบายการทำงานแบบกระจายได้อีกด้วย

ฐานข้อมูลเชิงวัตถุยังให้ประสิทธิภาพในการทำงานกับวัตถุ และความสัมพันธ์ระหว่างวัตถุที่ซับซ้อนได้ดีกว่า ทั้งนี้เนื่องมาจากการทำงานของฐานข้อมูลเชิงวัตถุไม่จำเป็นต้องมีการรวม (Join) ตาราง ซึ่งเป็นการทำงานที่ช้าที่สุดในการทำงานของระบบฐานข้อมูลเชิงสัมพันธ์ เพราะฐานข้อมูลเชิงวัตถุจะเก็บวัตถุไว้ด้วยกัน การค้นหาวัตถุจึงทำได้ในครั้งเดียว แม้ว่าจะเป็นวัตถุขนาดใหญ่ การทำงานของโปรแกรมประยุกต์ในระบบเชิงสัมพันธ์นั้น โปรแกรมประยุกต์จะได้ข้อมูลจากตารางที่ไม่ได้เชื่อมต่อกันในขณะทำงาน ในขณะที่โปรแกรมประยุกต์ในระบบฐานข้อมูลเชิงวัตถุ ข้อมูลจะถูกเก็บโดยตรง ดังนั้นแทนที่จะต้องมีการรวมตารางกัน โปรแกรมประยุกต์สามารถค้นหาวัตถุได้โดยใช้ตัวชี้ที่อยู่ในวัตถุทำ นอกจากนี้การแยกเก็บข้อมูลของวัตถุในรูปคุณลักษณะ (attributes) หรือตารางของฐานข้อมูลเชิงสัมพันธ์นั้น ยังเป็นการเพิ่มความคับคั่งของข้อมูล (data congestion) ระหว่างโปรแกรมประยุกต์กับฐานข้อมูลในการรวบรวมเพื่อให้ได้มาซึ่งวัตถุ และในทางกลับกันเพื่อเก็บวัตถุลงฐานข้อมูลด้วย การรวมตารางเพื่อให้ได้มาซึ่งวัตถุจะยิ่งมีปัญหามากขึ้น เมื่อฐานข้อมูลต้องทำการเก็บข้อมูลที่มีความซับซ้อนสูง เช่น รูปภาพ เอกสาร บัญชี และอื่นๆ นอกจากนี้การเก็บวัตถุของฐานข้อมูลเชิงวัตถุยังประหยัดพื้นที่มากกว่าระบบฐานข้อมูลเชิงสัมพันธ์ที่ต้องมีการสร้าง index จำนวนมากกว่าอีกด้วย

การอธิบายด้วยวัตถุสามารถรองรับโปรแกรมประยุกต์ทางด้านสื่อผสม (Multimedia) ซึ่งประกอบด้วยวัตถุที่มีคุณสมบัติที่หลากหลาย และมีการเปลี่ยนแปลงตามเวลา เช่น ภาพเคลื่อนไหว เสียง ข้อความหรือสื่อชนิดอื่นๆ ด้วยการสร้างแบบจำลอง (model) และจัดเก็บได้ภายในแนวคิด (conceptual) เดียวกัน

การนำกลับมาใช้ใหม่ได้ของวัตถุและความสามารถในการเพิ่มเติมคุณสมบัติให้กับวัตถุที่มีอยู่เดิมเป็นประโยชน์อีกส่วนหนึ่งที่ได้จากโครงสร้างแบบ Classification ของฐานข้อมูลเชิงวัตถุ ฐานข้อมูลเชิงวัตถุจะมีการตรวจสอบความถูกต้องในการอ้างอิงอยู่ในตัว เนื่องจากการอ้างอิงวัตถุอื่นนั้น ไม่ใช้การอ้างอิงถึงเท่านั้น แต่เป็นการบอกถึงที่อยู่จริงของวัตถุโดยตรง จึงมีความถูกต้องอยู่ในตัวมันเองอยู่แล้ว นอกจากนี้ฐานข้อมูลเชิงวัตถุยังแสดงถึงความสัมพันธ์ของวัตถุได้มากกว่าระบบอื่นด้วย ความสามารถในการสร้างชนิดของข้อมูลและรวมเอาข้อบังคับต่างๆ เข้าไว้ในวัตถุ อีกทั้งยังมีความสามารถในการจัดการข้อยกเว้นและการข้ามผ่านข้อบังคับด้วย

ในระบบเชิงสัมพันธ์นั้น การกระจายกันเก็บของวัตถุ เป็นปัญหาที่สำคัญอย่างหนึ่งของ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

ผู้ออกแบบ เพราะผู้ออกแบบจะต้องให้แน่ใจว่าได้ตารางที่มีการรวมกันบ่อย ๆ จะถูกเก็บที่เดียวกันและอยู่ในพื้นที่ที่ใกล้เคียงกันเพื่อลดปริมาณการเข้าถึงดิสก์ แต่ในฐานข้อมูลเชิงวัตถุนั้น วัตถุที่เป็นกลุ่มเดียวกันจะถูกเก็บไว้ที่เดียวกันเหมือนกับการประกอบกันของวัตถุ โครงสร้างของคลาส จะถูกนำมาใช้เป็นหลักในการพิจารณาการแยกวัตถุออกเป็นกลุ่มในการเก็บ

ดังที่กล่าวไปแล้วว่าโปรแกรมประยุกต์ที่อยู่ในระบบฐานข้อมูลเชิงสัมพันธ์ใช้ข้อมูลที่มีโครงสร้างไม่เหมือนกัน คือ โปรแกรมประยุกต์ใช้ตัวแปร ในขณะที่ฐานข้อมูลใช้ตาราง ดังนั้นการทำงานกับฐานข้อมูล จะต้องมีภาษาอีกภาษาหนึ่งมาช่วยในการทำงาน นั่นก็คือ Data Manipulation Language (DML) ซึ่งทำให้ผู้เขียนโปรแกรมใช้เขียนภาษาถึงสองภาษาก็คือ ภาษาของตัวโปรแกรมประยุกต์นั้นกับ DML ในขณะที่ระบบฐานข้อมูลเชิงสัมพันธ์นั้น โปรแกรมประยุกต์และฐานข้อมูล จะใช้ตัวแปรในรูปแบบเดียวกัน จึงไม่จำเป็นต้องมี DML ซึ่งเป็นการลดภาระให้กับผู้พัฒนาระบบ

ความสามารถในการสร้าง Library ของคลาสของโปรแกรมประยุกต์และฐานข้อมูลที่มีลักษณะคล้ายๆ กัน ช่วยให้การพัฒนาระบบเป็นไปได้เร็วขึ้น เพราะไม่ต้องเสียเวลาในการสร้าง class เดิมที่เคยทำมาก่อนแล้ว

ฐานข้อมูลเชิงวัตถุสามารถแปลงสถาปัตยกรรมไปเครื่องขอบริการ/เครื่องให้บริการ (Client/server) และการวัตถุทำงานแบบกระจาย (Distributed COM) ได้ง่าย เพราะการส่งผ่านข้อมูลไปยัง method ที่ถูกซ่อน (encapsulation) อยู่ในตัววัตถุนั้น แสดงให้เห็นถึงสถาปัตยกรรมแบบเครื่องขอบริการ/เครื่องให้บริการอยู่แล้ว จึงไม่จำเป็นต้องศึกษาการทำงานของวัตถุนั้นๆ อีก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

46178

บทที่ 3
แบบจำลองสำหรับการวิเคราะห์และออกแบบระบบเชิงวัตถุ UML

3.1 บทนำ

การพัฒนาระบบแบบโครงสร้างนั้น จะพยายามให้นักพัฒนาระบบแก้ปัญหาด้วยการแบ่งปัญหาออกเป็นส่วน ๆ แล้วแก้ปัญหาในแต่ละส่วนด้วยอัลกอริทึม (Algorithm) และโครงสร้างข้อมูล (Data Structure) เฉพาะสำหรับการพัฒนาระบบด้วยวิธีเชิงวัตถุก็เช่นเดียวกัน แต่จะมีมุมมองต่อปัญหาเป็นวัตถุประกอบกับการเขียนโปรแกรมด้วยภาษาในการเขียนโปรแกรมเชิงวัตถุ การพัฒนาระบบด้วยวิธีเชิงวัตถุไม่เพียงจะเป็นแนวความคิดใหม่ในการเขียนโปรแกรมเท่านั้น ข้อดีของแบบจำลองเชิงวัตถุยังสามารถนำไปประยุกต์กับการออกแบบฐานข้อมูลเชิงวัตถุได้ เนื่องจากหลักการของการพัฒนาโปรแกรมเชิงวัตถุเป็นแนวคิดในการจัดการกับความซ้ำซ้อนของสิ่งต่าง ๆ อย่างมีระบบ ซึ่งสามารถนำไปวิเคราะห์และออกแบบได้อย่างกว้างขวาง

การวิเคราะห์และออกแบบระบบด้วยวิธีการเชิงวัตถุเป็นวิวัฒนาการ ไม่ใช่เป็นการปฏิวัติการวิเคราะห์และออกแบบระบบที่มีอยู่ เนื่องจากการพัฒนาระบบด้วยวิธีเชิงวัตถุนั้นยังคงข้อดีในการพัฒนาระบบแบบเก่าแต่ก็ได้เพิ่มข้อดีใหม่ ๆ เข้าไปด้วย ในการพัฒนาระบบแบบเก่า จะพบกับความซ้ำซ้อนเนื่องจากการแบ่งระบบออกเป็นส่วนย่อย เมื่อส่วนย่อยเหล่านี้มีมากขึ้น ส่งผลให้การดูแลและการแก้ไขระบบเป็นไปได้ยาก ในขณะที่หลักการของเชิงวัตถุนั้น มีการแบ่งปัญหาออกเป็นวัตถุ ซึ่งมีความผูกพันธ์กันน้อย ทำให้การดูแลและแก้ไขส่วนต่าง ๆ ของระบบหรือวัตถุสามารถทำได้ง่าย และมีผลกระทบต่อส่วนอื่น ๆ ของระบบน้อยที่สุด

แนวความคิดของการพัฒนาระบบด้วยวิธีเชิงวัตถุเกิดขึ้นครั้งแรกเมื่อประมาณปี 1970 โดยเริ่มจากการออกแบบสถาปัตยกรรมฮาร์ดแวร์ของคอมพิวเตอร์ นักพัฒนาฮาร์ดแวร์ได้ใช้แนวคิดขอการพัฒนาระบบด้วยวิธีเชิงวัตถุช่วยในการลดช่องว่างระหว่างการแยกแยะเอกลักษณ์ในระดับสูง ซึ่งเป็นระดับการเขียนโปรแกรม และแยกเอกลักษณ์ในระดับล่างเป็นระดับการทำงานของฮาร์ดแวร์ ด้วยการนำหลักการของการพัฒนาระบบด้วยวิธีเชิงวัตถุมาใช้ในการออกแบบและพัฒนา สามารถตรวจสอบความผิดพลาดได้ง่าย ช่วยเพิ่มประสิทธิภาพในการปฏิบัติคำสั่ง ลดชนิดของคำสั่งของฮาร์ดแวร์ มีการคอมไพล์ภาษาระดับสูงเป็นภาษาระดับล่างที่ไม่ซับซ้อน และยังช่วยลดพื้นที่ในการเก็บข้อมูลด้วย

3.2 UML (Unified Modeling Language)

เป็นภาษามาตรฐานทางอุตสาหกรรมผลิตโปรแกรมประยุกต์ ใช้สำหรับแสดงรายละเอียด จำลอง สร้าง และจัดการเอกสารต่างๆ ในการผลิตโปรแกรมประยุกต์ โดยทำให้การออกแบบโปรแกรมประยุกต์หรือการสร้างพิมพ์เขียวทำได้ง่าย นิยามโดย Grady Booch, Ivar Jacobson และ Jim Rumbaugh

Object - Oriented คืออะไร

"Object-Oriented" เป็นแนวคิดหรือระเบียบวิธีคิดของการพัฒนาระบบงาน โดยจะมองระบบ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ออกเป็นกลุ่มของวัตถุที่มีปฏิกิริยาต่อกัน โดยการรวมเอาข้อมูลและฟังก์ชันการทำงานเข้าไว้ด้วยกันในวัตถุและกำหนดวิธีการติดต่อกันระหว่างวัตถุ โดยถ้าวัตถุหนึ่งต้องการติดต่อผ่านวัตถุหนึ่งจะต้องติดต่อผ่านอินเทอร์เฟซที่กำหนดไว้เท่านั้น

ดังนั้นการวิเคราะห์และออกแบบเชิงวัตถุที่เหมาะสมควรจะใช้ภาษาที่สนับสนุนแนวคิดนี้ นั่นคือ UML ซึ่งหากเป็น DFD และ ERD จะเหมาะสมกับแนวคิดแบบ Relational มากกว่า เนื่องจากมองวัตถุหรือข้อมูลแยกออกจากฟังก์ชันและไม่มีการอธิบายถึงเหตุการณ์ที่เกิดขึ้นกับวัตถุ ซึ่งไม่สอดคล้องกันกับเครื่องมือพัฒนาโปรแกรม

3.3 มุมมองของ UML

ในการออกแบบระบบที่มีขนาดใหญ่และมีความซับซ้อนมากๆ นั้นจะทำให้ผู้ออกแบบระบบไม่สามารถที่จะออกแบบระบบได้ครบถ้วน ดังนั้นจึงต้องมีการมองระบบเป็นมุมมองต่างๆ เพื่อทำให้ง่ายในการออกแบบ ดังนั้นระบบจึงมี View ที่ต่างๆกัน ซึ่งแต่ละ View จะแสดงมุมมองเฉพาะของระบบซึ่งอธิบายรวมกันเป็นระบบที่สมบูรณ์ ซึ่งจะประกอบด้วย View ต่างๆดังนี้

มุมมองยูสเคส(Use Case View)

อธิบายการทำงานต่างๆ ของระบบที่ถูกมองจากภายนอกหรือผู้ใช้ระบบ Use-case view ซึ่งอธิบายโดย ยูซเคสไดอะแกรม (Use-case diagram) เป็นมุมมองสำหรับลูกค้า , ผู้ออกแบบ , ผู้พัฒนาระบบ และ ผู้ทดสอบระบบ

มุมมองทางโลจิก(Logical View)

อธิบายการทำงานต่างๆที่ถูกออกแบบไว้ภายในระบบ ว่าระบบจะมีบริการอะไรให้กับผู้ใช้บ้าง โดยจะแสดงโครงสร้างแบบสถิต (dynamic collaboration) ซึ่งจะเกิดขึ้นเมื่อออบเจ็กต์ ส่งแมสเซสระหว่างกันในการทำงาน

โครงสร้างแบบสถิตจะอธิบายโดยใช้ คลาสไดอะแกรม (Class diagram) และ ออบเจ็กต์ไดอะแกรม (Object diagram) ส่วนการทำงานร่วมกันแบบไดนามิกจะอธิบายโดยใช้ สเตทไดอะแกรม (State diagram), ซีเควนต์ไดอะแกรม (Sequence diagram), คอลแลโบเรชั่นไดอะแกรม (Collaboration diagram) และ แอ็คติวิตี้ไดอะแกรม (Activity diagram)

มุมมองคอมโพเนนท์(Component View)

อธิบายการสร้างและความขึ้นต่อกันของโมดูล (Module) ที่ใช้ในการพัฒนาระบบ โดยใช้คอมโพเนนท์ไดอะแกรม (Component diagram) ในการอธิบาย

มุมมองดีพลอยเมนท์(Deployment View)

อธิบายการจัดวางระบบให้เหมาะสมในด้านกายภาพ (Physical) แสดงด้วยคอมพิวเตอร์และโหนด (nodes) ต่างๆ เพื่อให้ระบบมีเสถียรภาพมากขึ้นโดยใช้ ดีพลอยเม้นไดอะแกรม (Deployment Diagram) ในการอธิบาย

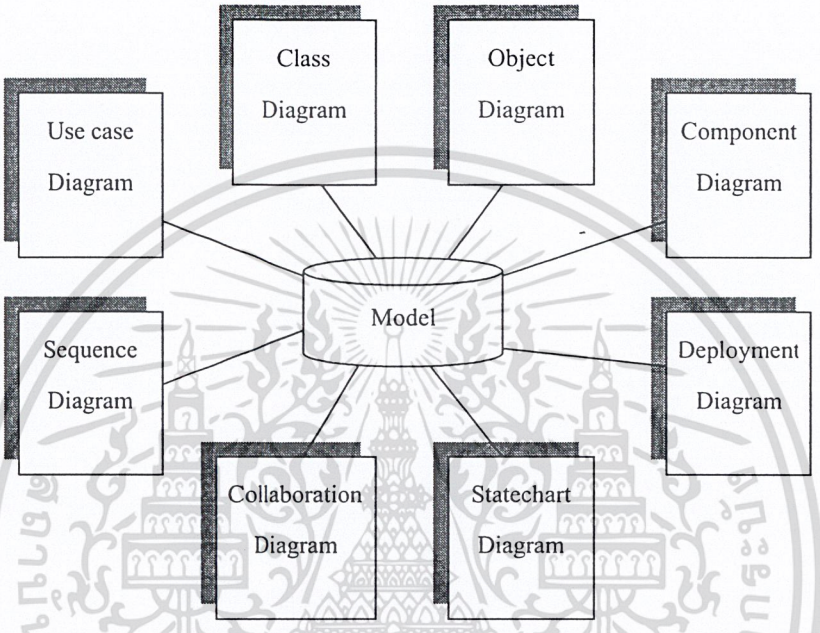
มุมมองการทำงาน(Process View)

แสดงการทำงานร่วมกันและการติดต่อกันของส่วนต่างๆในระบบ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.4 แผนภาพของ UML

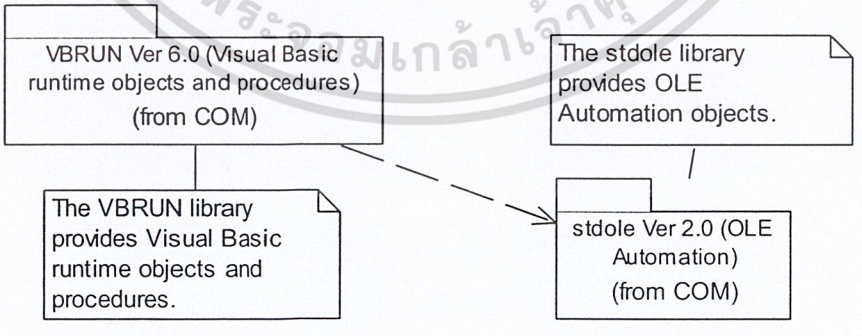
UML ประกอบด้วยไดอะแกรมทั้งหมด 8 ไดอะแกรม ดังแสดงในรูปที่ 3 - 1 เพื่อใช้ในการจำลองระบบงาน เปรียบได้กับการมองในแง่มุมต่างๆ เพื่อให้สามารถเข้าใจระบบงานให้มากที่สุด โดยที่ผู้จำลองแบบไม่จำเป็นต้องใช้ทุกไดอะแกรม โดยสามารถเลือกใช้ไดอะแกรมที่เหมาะสม



รูปที่ 3 - 1 แผนภาพแสดง UML ไดอะแกรมทั้งหมด

General-purpose Concept

การรวมกลุ่มของไดอะแกรมชนิดต่างๆ เป็นแพ็กเกจที่เกี่ยวข้องกัน ซึ่งมีสัญลักษณ์ที่ใช้ได้แก่ Package,Dependency และ Note ดังรูปที่ 3 - 2



รูปที่ 3 - 2 สัญลักษณ์ของ General-purpose concept

3.4.1 Use-case Diagram

ใช้มองภาพรวมของระบบและความต้องการต่างๆ คล้ายๆ กับ DFD ซึ่งสามารถบอกได้ว่าใครเกี่ยวข้องกับงานระบบใด ระบบอะไรและมีงานหลักๆ อะไรบ้าง ยูซเคสไดอะแกรมแสดงถึงตัวกระทำ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

(Actors), ยูซเคส และความสัมพันธ์ระหว่างยูซเคสกับตัวกระทำยูซเคสใช้แทนการทำงานที่ระบบทำได้ เช่นระบบย่อยหรือคลาส โดยมองจากตัวกระทำภายนอกที่มีการติดต่อกับระบบ

แต่ละยูซเคสไดอะแกรมคือกราฟของตัวกระทำ, ยูซเคส, ส่วนติดต่อหรืออินเทอร์เฟซ (Interface)(ถ้ามี) และความสัมพันธ์ระหว่างสิ่งเหล่านี้

ยูสเคส

ยูซเคสใช้แทนหน่วยการทำงานที่ระบบ หรือระบบย่อย หรือคลาสมีให้ ซึ่งแสดงให้เห็นโดยการแลกเปลี่ยนข้อมูลระหว่างระบบกับตัวกระทำภายนอก

จุดเอ็กเทนชั่น(extension point) เป็นจุดอ้างอิงภายในยูซเคสซึ่งอาจมีการทำงานของยูซเคสอื่นเข้ามาแทรกได้ แต่ละจุดเอ็กเทนชั่นจะมีชื่อที่แตกต่างกันในยูซเคสหนึ่งๆและมีคำบรรยายตำแหน่งการทำงานภายในพฤติกรรมของยูซเคส

ความสัมพันธ์ของยูสเคส(Use Case Relationship)

ความสัมพันธ์(Relationship)มาตรฐานระหว่างตัวกระทำกับยูซเคสและระหว่างยูซเคสกับยูซเคสมีอยู่หลายแบบ

แอสโซสิเอชั่น– Association: การกระทำร่วมกันระหว่างตัวกระทำกับยูซเคสแสดงในไดอะแกรมโดยเส้นตรง

เอ็กเท็นด์ – Extend:ความสัมพันธ์แบบขยาย(Extend) ระหว่างยูซเคส A กับยูซเคส B หมายความว่าตัวอย่าง(Instance) ของ B อาจถูกขยายโดยพฤติกรรมที่กำหนดใน A แสดงในไดอะแกรมโดยเส้นประที่มีลูกศรหัวเปิดจากยูซเคสที่เป็นตัวให้ extension ไปยัง use case ที่เป็นฐานโดยจะมีคีย์เวิร์ด <<extend>> กำกับไว้ ส่วนเงื่อนไขของความสัมพันธ์อาจใส่ไว้ใกล้กับ

คีย์เวิร์ด <<extend>>

เจนเนอรัลไลเซชั่น – Generalization: ความสัมพันธ์แบบเจนเนอรัลไลเซชั่นจากยูซเคส A ไปยัง ยูซเคส B หมายถึง A เป็นพ่อแม่(parent) หรืออยู่เหนือ B แสดงในไดอะแกรมโดยเส้นตรงที่มีหัวลูกศรปิดและกลวง หัวลูกศรชี้ที่ยูซเคสที่เป็นพ่อ

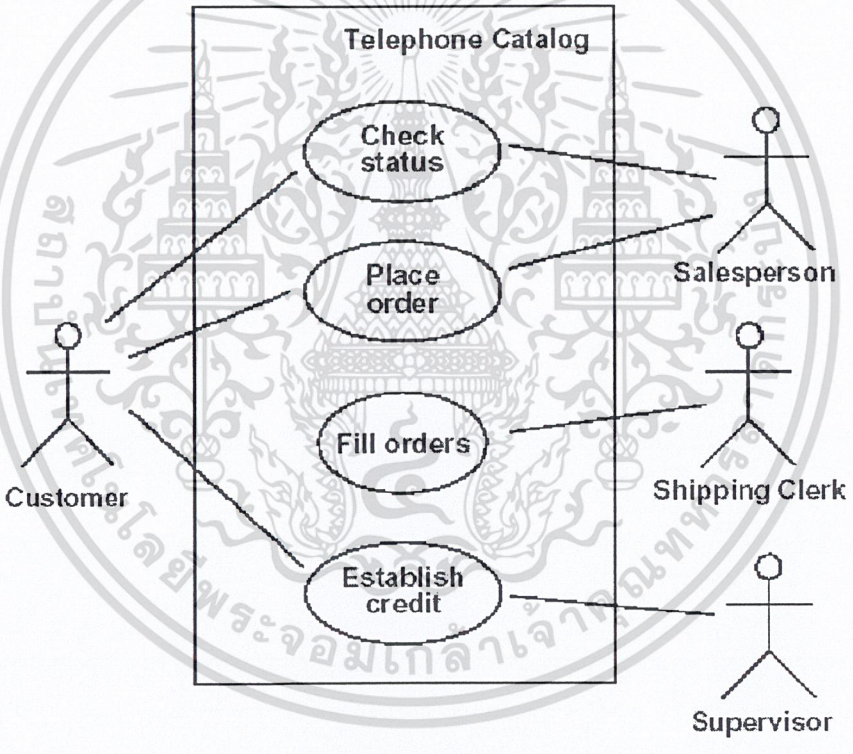
อินคลูด – Include: ความสัมพันธ์แบบรวมเข้าด้วยกันจากยูซเคส A ไปยังยูซเคส B หมายถึงภายใน A อาจรวมเอาพฤติกรรมที่ระบุโดย B เข้าไปด้วย แสดงในไดอะแกรมโดยประที่มีลูกศรหัวเปิดจากยูซเคสที่เป็นฐานไปยังยูซเคสที่ถูกรวมเข้าด้วยกันโดยจะมีคีย์เวิร์ด <<include >> กำกับ

	Use case
	ตัวกระทำหรือ Actor

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

————	Association
- - - - - - - - -> <<extend>>	Extend
- - - - - - - - -> <<include>>	Include
————▷	Generalization

รูปที่ 3 - 3 สัญลักษณ์ที่ใช้ในยูซเคสไดอะแกรม



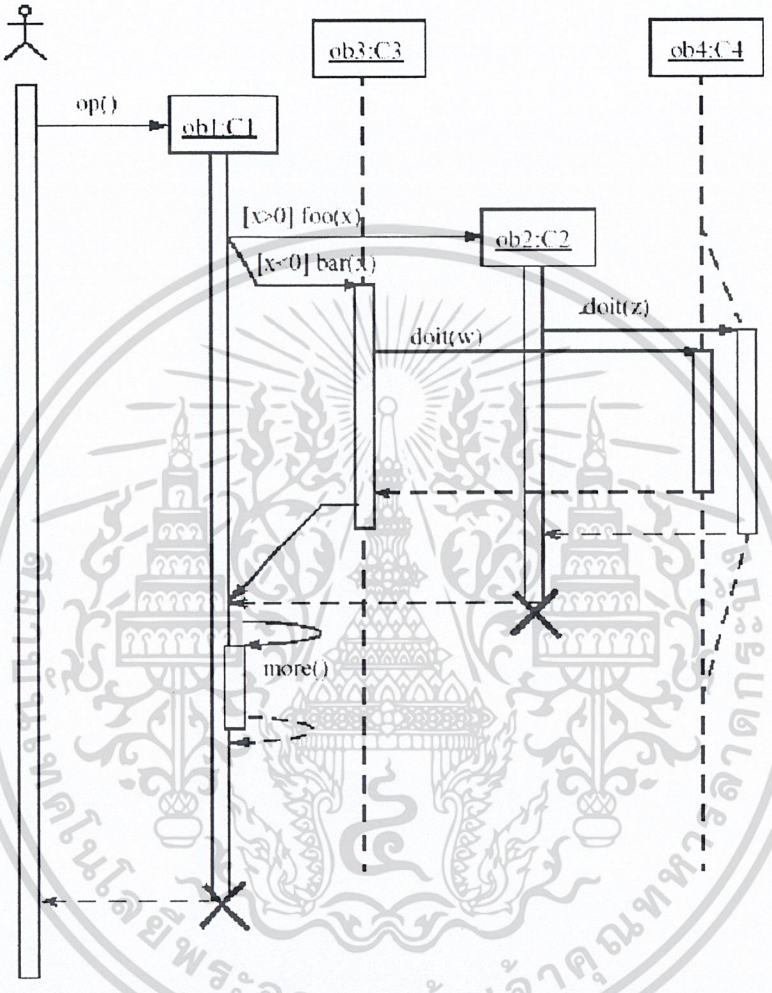
รูปที่ 3 – 4 ตัวอย่างยูซเคสไดอะแกรม

3.4.2 ซีเควนต์ไดอะแกรม (Sequence Diagram)

ซีเควนต์ไดอะแกรมใช้แสดงแทนการกระทำต่อกัน(Interaction) ระหว่างออบเจ็กต์ในการทำงานให้ได้ตามจุดประสงค์หนึ่ง เป็นไดอะแกรมชนิด Interaction ซึ่งขยายการทำงานของแต่ละยูซเคส สำหรับแสดงการทำงานระหว่างวัตถุต่าง ๆ ที่ส่งข้อความถึงกัน ซึ่งผู้เขียนโปรแกรมจะใช้ไดอะแกรมตัวนี้ช่วยเพื่อที่จะได้เขียนโปรแกรมตามที่ออกแบบไว้ได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ซีเควนต์ไดอะแกรมมี 2 มิติ คือ มิติในแนวตั้งแทนเวลา และมิติในแนวนอนแทนออบเจ็กต์ต่าง ๆ (ลำดับการเรียงของออบเจ็กต์ไม่มีผลต่อไดอะแกรม) มีการใช้ลูกศรชนิดต่างๆแทน "เมสเซจและตัวกระตุ้น" (Message and Stimulus)



รูปที่ 3 - 5 ตัวอย่างซีเควนต์ไดอะแกรมที่มีการควบคุมการทำงาน, การใช้เงื่อนไข, การเรียกหาตัวเอง(Recursion), การสร้าง และการทำลาย

เส้นชีวิตของออบเจ็กต์(Object Lifeline)

แสดงการมีอยู่ของออบเจ็กต์ในช่วงเวลาหนึ่งเขียนแทนด้วยเส้นประในแนวตั้ง ถ้าออบเจ็กต์ถูกสร้างขึ้นและถูกทำลายในระหว่างช่วงเวลาในไดอะแกรมแล้ว เส้นชีวิตของมันจะต้องเริ่มและสิ้นสุดลงอย่างเหมาะสม ถ้าออบเจ็กต์ถูกสร้างขึ้นในไดอะแกรมลูกศรที่เป็นตัวกระตุ้นให้สร้างออบเจ็กต์จะชี้ไปที่สัญลักษณ์ออบเจ็กต์ของออบเจ็กต์ที่ถูกสร้าง จากรูปด้านบนคือเส้น op() ที่สร้างออบเจ็กต์ ob1 และถ้าออบเจ็กต์ถูกทำลายระหว่างช่วงเวลาในไดอะแกรม การถูกทำลายจะแทนด้วยเครื่องหมาย "X" ที่ลูกศรที่สั่งให้ทำลายออบเจ็กต์นั้นหรือที่ลูกศรออกจากออบเจ็กต์สุดท้ายในกรณีที่ออบเจ็กต์ทำลายตัวเอง

แอ็คติเวชั่น(Activation)

แสดงช่วงเวลาที่ออบเจ็กต์มีการทำงาน ใช้แทนการทำงานของออบเจ็กต์ในช่วงเวลาหนึ่ง

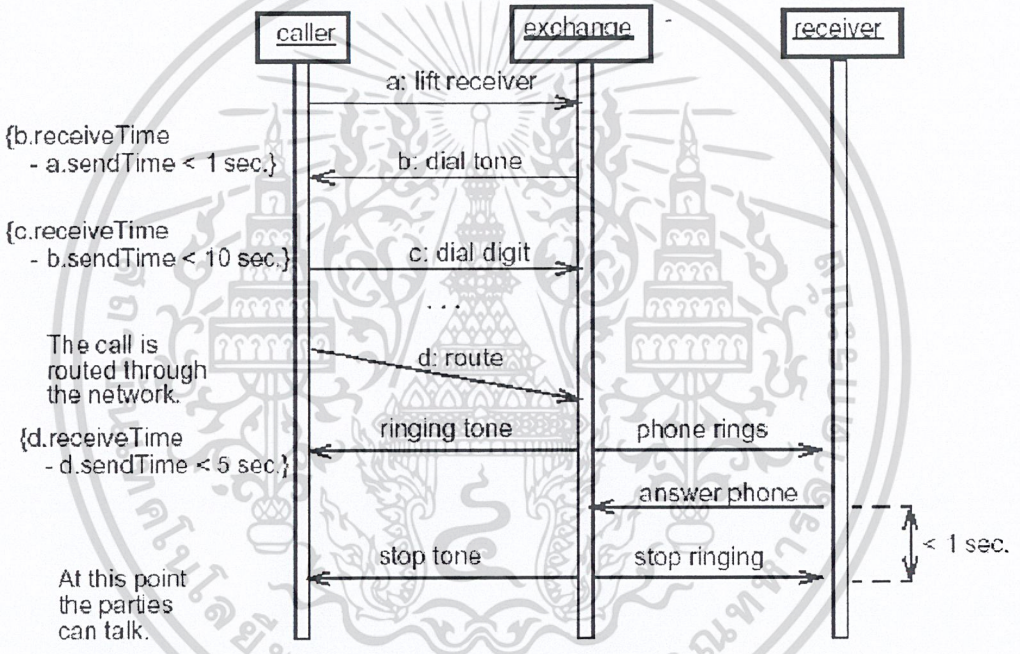
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รวมทั้งความสัมพันธ์ของการควบคุมระหว่างช่วงการทำงานและผู้เรียกให้ทำงาน เขียนแสดงสี่เหลี่ยมบางๆซึ่งด้านบนคือจุดเวลาเริ่มต้นและด้านล่างคือจุดสิ้นสุด ในการไหลของการควบคุมแบบโพรซีเยอร์ด้านบนของสัญลักษณ์แอ็คติเวชั่นจะอยู่ที่หัวลูกศรที่สั่งให้เกิดการทำงานและด้านล่างคือหางลูกศรของการออกจากโพรซีเยอร์

ในกรณีที่มีการเรียกออบเจ็กต์ที่มีแอ็คติเวชั่นอยู่แล้วซ้ำ จะมีอีกแอ็คติเวชั่นหนึ่งเพิ่มขึ้นมาทางด้านขวาของอันแรก

ทรานสิชั่นไทม์(Transition Times)

อาจระบุเวลาแบบต่างๆให้กับเมสเซจ เช่น “sending time” หรือ “receiving time”ซึ่งสามารถนำไปใช้เพื่อการบอกอายุของเมสเซจได้



รูปที่ 3 - 6 ตัวอย่างทรานสิชั่นไทม์

เมสเซจและตัวกระตุ้น(Message and Stimulus)

ตัวกระตุ้น(Stimulus) คือการสื่อสารระหว่างสองออบเจ็กต์ซึ่งส่งข้อมูลโดยคาดว่าจะเกิดการกระทำขึ้นตัวกระตุ้นจะเรียกให้เกิดการทำงาน การสร้างสัญญาณ(Signal) หรือทำให้ออบเจ็กต์ถูกสร้างหรือถูกทำลาย

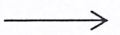
เมสเซจคือการกำหนดของตัวกระตุ้น เช่น ระบุบทบาทซึ่งออบเจ็กต์ตัวส่งและรับเมสเซจจะต้องทำงานให้สอดคล้องตาม หรือการกระทำซึ่งเมื่อถูกสั่งให้ทำแล้ว
จะส่ง Stimulus ซึ่งสอดคล้องกับเมสเซจออกมา

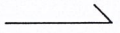
ตัวกระตุ้นจะถูกเขียนแทนด้วยลูกศรจากเส้นชีวิตของออบเจ็กต์หนึ่งไปยังเส้นชีวิตของอีกออบเจ็กต์หนึ่งหรือกลับเข้าหาตัวเองก็ได้ ลูกศรจะถูกกำกับด้วยชื่อของตัวกระตุ้น (โอเปอเรชั่นหรือสัญญาน) พร้อมด้วยอาร์กิวเมนต์

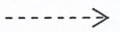
ชนิดของลูกศรแบบต่างๆ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การเรียกโพรซีเยอร์หรือกลุ่มการควบคุมหนึ่ง การทำงานภายในทั้งหมดต้องเสร็จสิ้นก่อนที่กลับไปทำงานในระดับด้านนอก

 แสดงการเข้าสู่ลำดับต่อไปในลำดับการทำงาน

 แสดงตัวกระตุ้นแบบ Asynchronous ใช้แสดงการสื่อสารระหว่างสองออบเจ็กต์แบบ Asynchronous ในลำดับการทำงานของโพรซีเยอร์

 การออกจากโพรซีเยอร์

การแยกการทำงาน – Branching: แสดงโดยลูกศรหลายเส้นออกจากจุดเดียวกัน แต่ละเส้นกำกับโดยเงื่อนไขการเลือกทำโดยทุกเส้นจะต้องแตกต่างกัน(เลือกได้ทางเดียว)

การทำงานซ้ำ – Iteration: กลุ่มของลูกศรที่เชื่อมกันนำมารวมกันและกำกับไว้ว่าเป็น Iteration แสดงถึงการส่งกลุ่มของ Stimuli สามารถเกิดได้หลายครั้ง

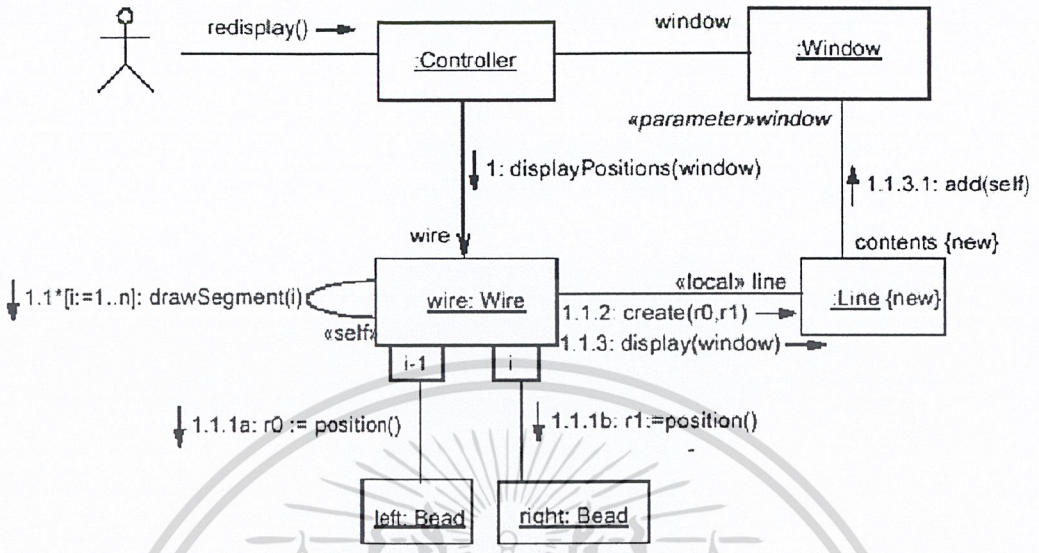
3.4.3 คอลแลโบเรชั่นไดอะแกรม (Collaboration Diagram)

เป็นไดอะแกรมชนิดเดียวกับซีเควนต์ไดอะแกรม เพราะเป็นการแสดงการทำงานร่วมกันระหว่างวัตถุโดยที่ซีเควนต์ไดอะแกรมจะแสดงถึงการแลกเปลี่ยนข่าวสาร ส่วนคอลแลโบเรชั่นไดอะแกรมเป็นการแสดงความสัมพันธ์ภายในวัตถุ โดยมีวิธีการเลือกใช้คือ ถ้าเป็นการกำหนดช่วงเวลาที่แน่นอนและใช้เวลาเป็นสิ่งสำคัญ มีลำดับก่อนหลังให้ใช้ซีเควนต์ไดอะแกรมแต่ถ้าเป็นการให้ความสำคัญกับความสัมพันธ์ภายในวัตถุให้เลือกใช้ไดอะแกรมนี้

คอลแลโบเรชั่นไดอะแกรมแสดงแทนคอลแลโบเรชั่นซึ่งประกอบด้วยเซตของบทบาบของออบเจ็กต์และความสัมพันธ์ระหว่างออบเจ็กต์ ไดอะแกรมนี้อาจใช้แสดงการกระทำต่อกันระหว่างออบเจ็กต์โดยการระบุเซตของเมสเซจที่บอกการกระทำต่อกันระหว่างออบเจ็กต์ที่มีบทบาทอยู่ภายในคอลแลโบเรชั่นเพื่อบรรลุจุดประสงค์หนึ่ง

คอลแลโบเรชั่นไดอะแกรมจะแสดงออกมาเป็นกราฟที่เชื่อมต่อระหว่างออบเจ็กต์ โดยทั่วไปคอลแลโบเรชั่นไดอะแกรมถูกใช้เพื่อช่วยในการออกแบบโพรซีเยอร์ มีการใช้ลูกศรเพื่อบอกการทำงานโดยตัวเส้นลูกศรแทนตัวกระตุ้นซึ่งถูกส่งไปในทิศทางที่หัวลูกศรชี้ ส่วนประกอบในไดอะแกรมส่วนใหญ่จะเหมือนกับซีเควนต์ไดอะแกรม ทั้งซีเควนซ์ไดอะแกรมและคอลแลโบเรชั่นไดอะแกรมต่างก็ใช้บอกการกระทำต่อกัน(Interation) โดยซีเควนซ์ไดอะแกรมแสดงลำดับของตัวกระตุ้นและสามารถเข้าใจได้ง่ายในการพิจารณาด้านเวลา ส่วนคอลแลโบเรชั่นไดอะแกรมจะแสดงความสัมพันธ์ตัวอย่าง(Instance) ช่วยให้ง่ายต่อการเข้าใจผลกระทบต่างๆที่เกิดขึ้นกับตัวอย่างนั้นๆและช่วยในการออกแบบโพรซีเยอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3 – 7 ตัวอย่างไดอะแกรมคอลแลโบเรชั่น

3.4.4 สเตทไดอะแกรม (State Diagram) หรือเรียกว่า State Machine ไดอะแกรม

เป็นไดอะแกรมที่แสดงถึงวงจรชีวิตของวัตถุโดยแสดงเหตุการณ์ต่างๆ ที่มีผลกระทบต่อสถานะของวัตถุ ซึ่งอาจจะมีจุดเริ่มต้นและสิ้นสุดได้หลาย ๆ จุด

ลักษณะทั่วไปของสเตทไดอะแกรม

สเตทไดอะแกรมหรือ Statechart ไดอะแกรมหรือบางครั้งเรียกว่า State Machine เป็นสิ่งที่ใช้แสดงการเปลี่ยนสถานภาพ (state) ของออบเจ็กต์ตั้งแต่เริ่มต้นจนสิ้นสุดการเปลี่ยนแปลงในรอบหนึ่งๆ (1 sequence) โดยทั่วไปแล้วสเตทไดอะแกรมจะใช้ในการอธิบายพฤติกรรมของคลาสแต่ละคลาส แต่ก็สามารถนำมาอธิบายพฤติกรรมของไดอะแกรมหรือแบบจำลองอื่นๆ ได้ด้วยเช่นกัน เช่น use-cases, actors, subsystems, operations และ methods เป็นต้น

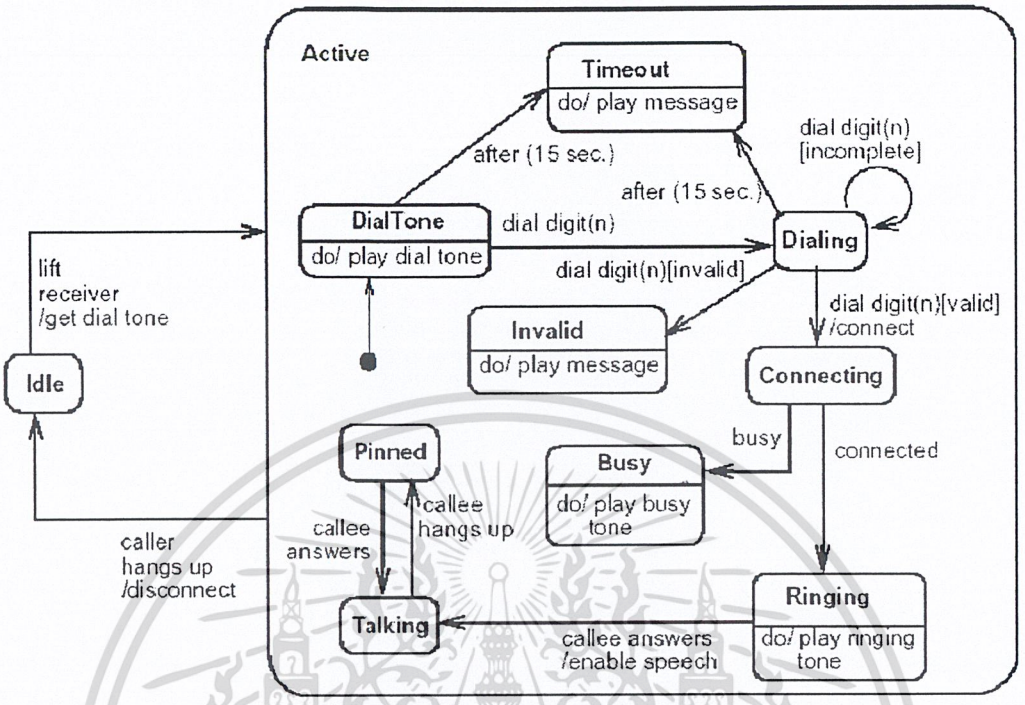
ลักษณะของการเปลี่ยนสถานภาพ เช่น

- เมื่อเรากดสวิทช์ไฟ หลอดไฟฟ้าจะเปลี่ยนสถานะจากมืดเป็นสว่าง
- เมื่อเรากดปุ่มรีโมทคอนโทรลเพื่อเปลี่ยนช่อง โทรทัศน์จะเปลี่ยนสถานะจากรายการช่องหนึ่งเป็นรายการอีกช่องหนึ่ง
- เมื่อเวลาผ่านไประยะหนึ่ง เครื่องซักผ้าจะเปลี่ยนสถานะจาก “ซัก” เป็น “ปั่น”

สถานภาพของออบเจ็กต์ ณ เวลาใดเวลาหนึ่ง จะมีการเปลี่ยนแปลงสถานะเมื่อมีการกระทำจากภาพนอก เช่น หลอดไฟจะมืดหรือสว่างได้ก็ต่อเมื่อเราปิดหรือเปิดสวิทช์ไฟ เป็นต้น แต่มีออบเจ็กต์บางอย่างสามารถเปลี่ยนสถานะได้ด้วยตัวเอง เราจะเรียกออบเจ็กต์ดังกล่าวว่า “Object with life” หรือ “Actor” เช่น นาฬิกา ลูกค้า เป็นต้น

ตัวอย่างของสเตทไดอะแกรมแสดงได้ดังรูป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3 – 8 แสดงตัวอย่างสเตทไดอะแกรม

สเตท

สเตทเป็นเงื่อนไขระหว่างการทำงานของออบเจ็กต์ หรือเป็นสิ่งที่แสดงการตอบสนองเงื่อนไขการกระทำบางอย่าง หรือ รอคอยเพื่อรับเหตุการณ์เข้ามา สัญลักษณ์ที่ใช้แสดงสเตทจะเป็นรูปสี่เหลี่ยมหัวมน โดยภายในจะประกอบด้วยชื่อสเตทซึ่งจะถูกแบ่งด้วยเส้นภายในสเตทนั้น ๆ และภายในสเตทยังสามารถถูกแบ่งออกเป็นส่วนย่อย ๆ อื่นได้อีก โดยใช้เส้นตรงในการแบ่ง ซึ่งมีดังต่อไปนี้

- ส่วนของชื่อสเตท (Name compartment)
- ส่วนแสดงทรานสิชั่นภายใน (Internal transition compartment) ในส่วนนี้จะแสดงรายการทำงานทั้งหมด (action) ที่สามารถถูกกระทำได้ในสเตทนั้นๆ โดยแสดงได้ดังนี้

action-label '/' action-expression

โดยที่จะมีชื่อแอ็กชั่นหลักๆ ที่ถูกสงวนไว้เพื่อจุดประสงค์หลักๆ ซึ่งไม่สามารถนำไปใช้เป็นชื่อของเหตุการณ์ได้ มีดังต่อไปนี้

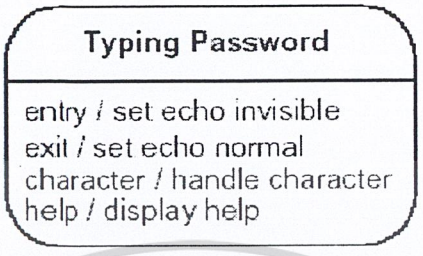
- entry เป็นส่วนแสดงแอ็กชั่น จะอธิบายการทำงานเมื่อมีการเข้าสู่สเตทนี้
- exit เป็นส่วนแสดงแอ็กชั่น จะอธิบายการทำงานเมื่อมีการออกจากสเตทนี้
- do เป็นส่วนที่แสดงถึงกิจกรรม ที่ถูกกระทำภายในสเตทนี้ ซึ่งกิจกรรมนี้จะถูกกระทำไปจนกว่าที่ action-expression จะเสร็จสมบูรณ์
- include ส่วนนี้เป็นส่วนที่ใช้เรียกสเตทย่อย ซึ่ง action-expression จะประกอบด้วยชื่อของสเตทย่อยที่จะถูกเรียกนั้น

รูปแบบทั่วไปของรายการในทรานสิชั่นภายใน มีดังต่อไปนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

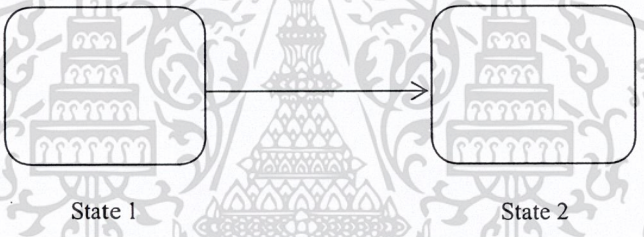
event-name '(' comma-separated-parameter-list ')' '[' guard-condition']' '/' action-expression

โดยที่ event-name สามารถที่จะปรากฏได้หลายครั้งต่อหนึ่งสเตท ถ้า guard conditions แตกต่างกัน ตัวอย่างของสเตทแสดงได้ดังรูป



รูปที่ 3 – 9 แสดงตัวอย่างสเตท

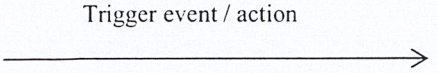
เราสามารถเอาสเตทการทำงานมาเชื่อมโยงถึงกันได้ด้วยกลไกที่เรียกว่า "transition line" โดยใช้สัญลักษณ์เป็นลูกศร ลักษณะของการเชื่อมโยงระหว่างสเตทด้วยเส้นการทำงาน (transition line) จะมีลักษณะดังรูป



รูปที่ 3 – 10 แสดงการเชื่อมสเตทด้วยเส้นการทำงาน

ดังที่กล่าวมาแล้วว่าเส้นการทำงาน คือองค์ประกอบหนึ่งของสเตทไดอะแกรม โดยจะเชื่อมโยงสเตท สถานะการทำงานต่างๆ รวมทั้งแสดงให้เห็นถึงลำดับการทำงานด้วย ซึ่งในส่วนของเส้นการทำงานนั้น เราสามารถใส่รายละเอียดบางอย่างเพิ่มเติมเข้าไปได้ กล่าวคือ เราอาจจะมีการใส่เหตุการณ์ที่ทำให้เกิดทรานสิชั่นหนึ่งๆ ขึ้น ซึ่งเราเรียกเหตุการณ์นั้นว่า "trigger event"

นอกจากนี้เรายังสามารถใส่การคำนวณบางอย่างหรือการทำงานบางอย่างที่ก่อให้เกิดการเปลี่ยนแปลงสถานะการทำงาน และไม่ว่าเราจะใส่เหตุการณ์หรือการทำงานบางอย่างเพิ่มเติมลงใน transition line เราจะใส่เอาไว้ข้างๆ transition line ตัวอย่างการกำหนด trigger event หรือ action มีลักษณะดังรูป



รูป 3 - 11 แสดงการใส่รายละเอียดบนเส้นการทำงาน

แต่ในบางครั้งการเปลี่ยนสถานะการทำงานก็เป็นไปได้อัตโนมัติ โดยอาจจะอาศัยผลลัพธ์จากสถานะการทำงานที่ผ่านมาทำให้เกิดสถานะใหม่ก็ได้ ลักษณะของการเปลี่ยนแปลงสถานะการทำงานโดยไม่มีเหตุการณ์หรือการทำงานเข้าไปกระตุ้น เราจะเรียกว่า "triggerless transition"

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

นอกจากนี้แล้วยังมีอีกวิธีการหนึ่งที่จะทำให้เกิดการเปลี่ยนแปลงสถานะการทำงาน ได้แก่การกำหนด “Guard Condition” ซึ่งเป็นลักษณะของการกำหนดเงื่อนไขทางตรรกะบางอย่าง เช่น อาจจะใช้ระยะเวลาเป็นตัวกำหนดการเปลี่ยนแปลงสถานะ เป็นต้น

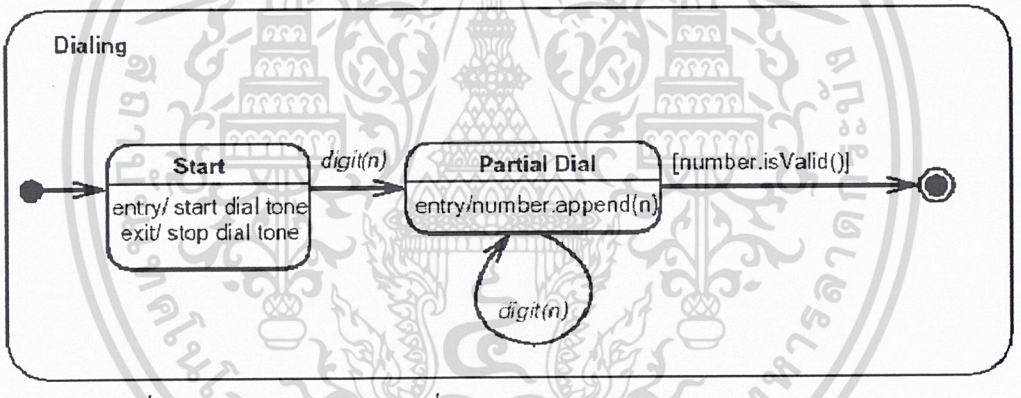
Composite States

เนื่องจากในสเตทหนึ่งๆ นั้นอาจจะสามารถแสดงถึงการทำงานที่ละเอียดได้อีก ดังนั้นเราจึงอาจมีการเพิ่มเติมรายละเอียดในแต่ละสเตทว่ามีการทำงานย่อยอะไรซ่อนอยู่บ้าง โดยเราจะเรียกสถานะการทำงานย่อยๆ ที่ซ่อนอยู่นั้นว่า “ซับสเตท (Substates)”

สถานการณ์ทำงานหรือซับสเตทนั้น แบ่งเป็น 2 ประเภทคือ

1. Sequential

ลักษณะของ Sequential จะเป็นการเปลี่ยนสถานะการทำงานอย่างเป็นลำดับ จากสถานะหนึ่งไปยังสถานะหนึ่ง ลองพิจารณารูปตัวอย่างดังต่อไปนี้

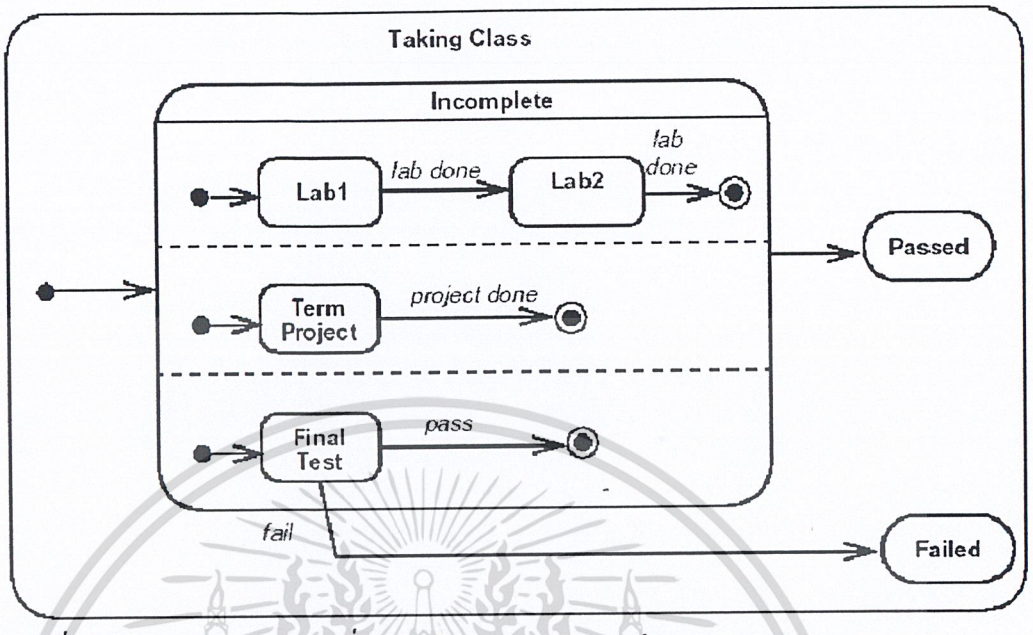


รูปที่ 3 – 12 แสดง State Dialing ที่มีการแบ่งสถานะการทำงานย่อยแบบ Sequential

2. Concurrent

ลักษณะของ Concurrent นั้น สถานะการทำงานจะเปลี่ยนแปลงหลายอย่างพร้อมๆ กัน หนึ่ง ลองพิจารณารูปตัวอย่างดังต่อไปนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

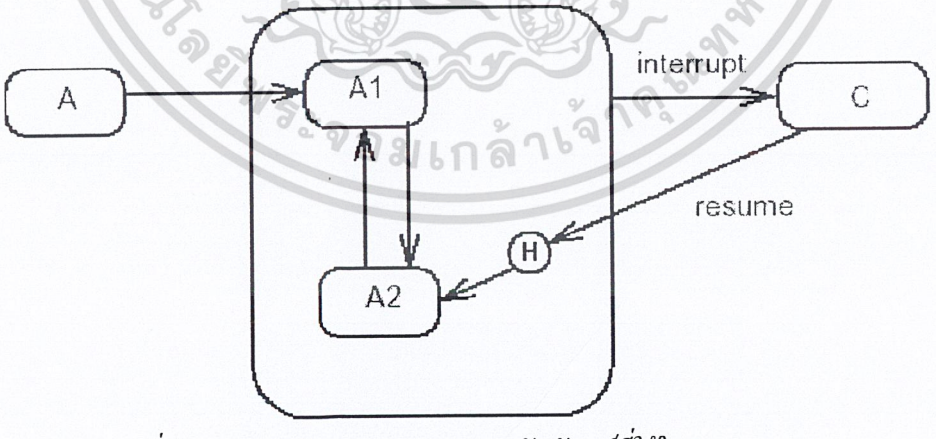


รูปที่ 3 – 13 แสดง State Dialing ที่มีการแบ่งสถานะการทำงานย่อยแบบ Concurrent

History State

การที่เกิดการเปลี่ยนสถานะจากสถานะหนึ่งไปยังอีกสถานะหนึ่ง และมีการเปลี่ยนแปลงจากสถานะดังกล่าวกลับมายังสถานะเดิม เราจะมีส่วนที่เรียกว่า "History State" เพื่อเป็นตัวบอกว่าเมื่อมีการเปลี่ยนแปลงกลับมาสถานะเดิม จะให้กลับมาที่จุดไหน

สัญลักษณ์ของ History State จะเป็นรูปวงกลมที่มีอักษรตัว H อยู่ภายในวงกลม แสดงได้ดังรูปตัวอย่างต่อไปนี้



รูปที่ 3 – 14 แสดงลักษณะการทำงานและสัญลักษณ์ที่ใช้ใน History State

สเตทไดอะแกรมแสดงให้เห็นถึงวงจรชีวิตของออบเจ็กต์ ระบบงานย่อย และระบบงานทั้งหมด โดยบ่งบอกว่าเหตุการณ์หรือภาวะต่างๆ มีผลต่อการเปลี่ยนแปลงของระบบงานอย่างไร ซึ่งสเตทไดอะแกรมหนึ่งอาจมีจุดเริ่มต้นและจุดสิ้นสุดได้หลายจุด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การใช้สเตทไดอะแกรมจะช่วยให้เราเข้าใจถึงพฤติกรรมของระบบมากขึ้น ในขณะที่ไดอะแกรมอื่น ๆ อาจจะแสดงพฤติกรรมของระบบคืออะไร แต่จะไม่ลงไปในรายละเอียดมากนัก และเมื่อเราสามารถรู้ถึงพฤติกรรมของระบบได้ครบถ้วน เราก็จะสามารถตอบสนองต่อความต้องการของผู้ใช้งานได้อย่างตรงจุด

3.4.5 แอ็กทิวิตี้ไดอะแกรม (Activity Diagram)

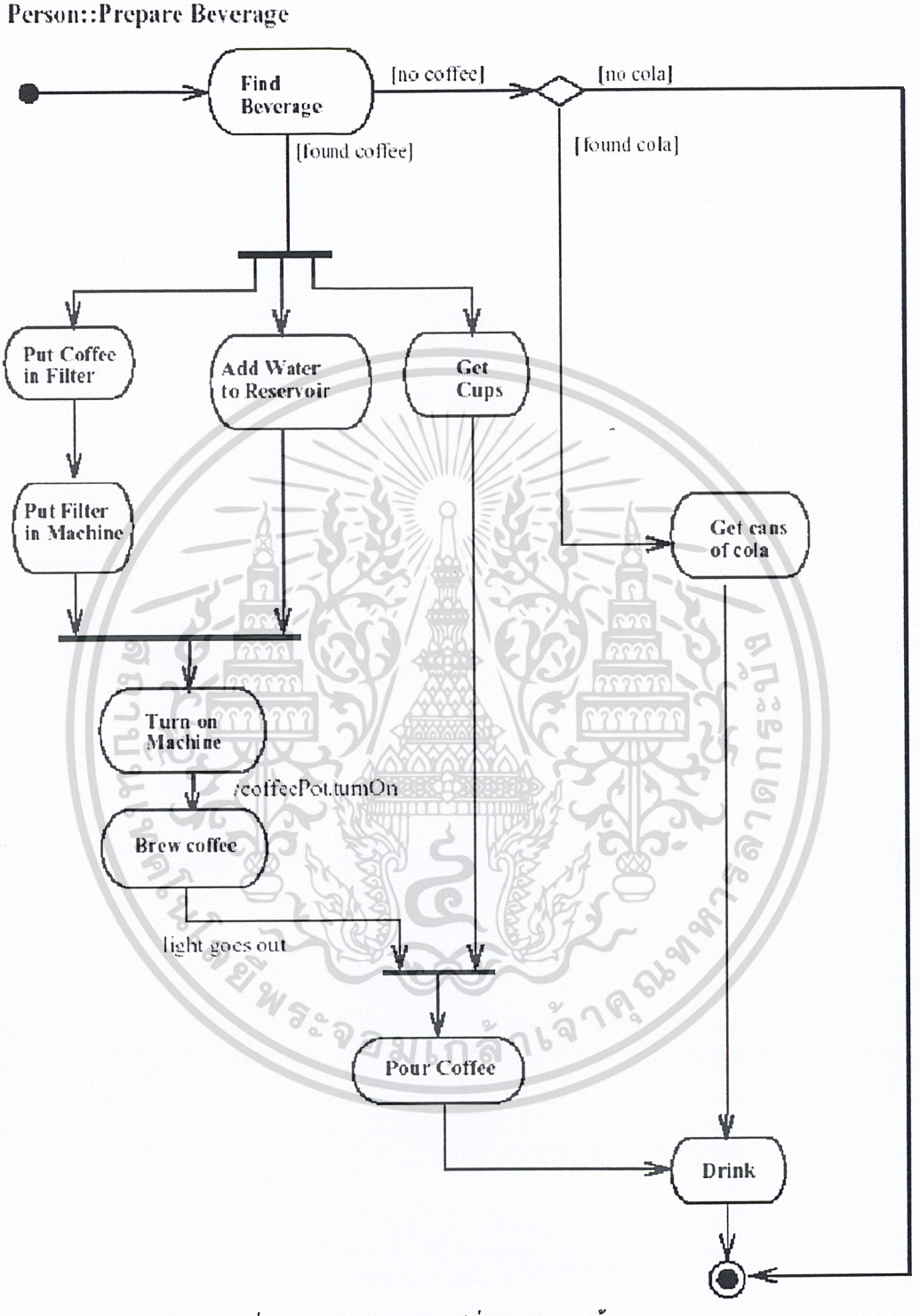
แอ็กทิวิตี้ไดอะแกรม จะมีลักษณะเดียวกันกับโฟลว์ชาร์ต (Flowchart) คือใช้สำหรับแสดงขั้นตอนการทำงานของระบบ แต่ต่างกันตรงที่แอ็กทิวิตี้ไดอะแกรม สามารถแสดงกิจกรรมที่ทำงานเป็นแบบขนานได้ เช่นงานที่ต้องแตกตัวเป็นหลาย Thread และรันไปพร้อมๆ กัน โดยที่ขั้นตอนในการทำงานแต่ละขั้นตอนเราจะเรียกว่า "แอ็กทิวิตี้"

แอ็กทิวิตี้ อาจเป็นลักษณะของการทำงานต่างๆ ได้แก่

- การคำนวณผลลัพธ์บางอย่าง
- การเปลี่ยนแปลงสถานะ (state) ของระบบ
- การส่งค่าบางอย่างกลับคืนมา
- การเรียกให้โอเปอเรชันอื่นๆ ทำงาน
- การส่งสัญญาณ
- การสร้างหรือการทำลายออบเจ็กต์

Acitvity Diagram จะต้องมีจุดเริ่มต้นกับจุดสิ้นสุด และในระหว่างจุดเริ่มต้นกับจุดสิ้นสุด ก็จะมีขั้นตอนหรือแอ็กทิวิตี้ต่างๆ ของระบบ ดังรูปตัวอย่างต่อไปนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3 – 15 แสดงสัญลักษณ์ที่ใช้ในแอ็กทิวิตี้ไดอะแกรม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากไดอะแกรมข้างต้น วงกลมทึบที่อยู่บนสุดแทนจุดเริ่มต้นของไดอะแกรมรูปสี่เหลี่ยมขอบมนจะแทนกิจกรรม หรือแอ็กทิวิตี้ของระบบที่เรากำลังสนใจ และวงกลมทึบที่อยู่ภายในวงกลมอีกทีจะแทนจุดสิ้นสุดของไดอะแกรมโดยปกติแล้วเราจะเขียนแอ็กทิวิตี้ไดอะแกรมโดยอ่านจากบนลงล่าง

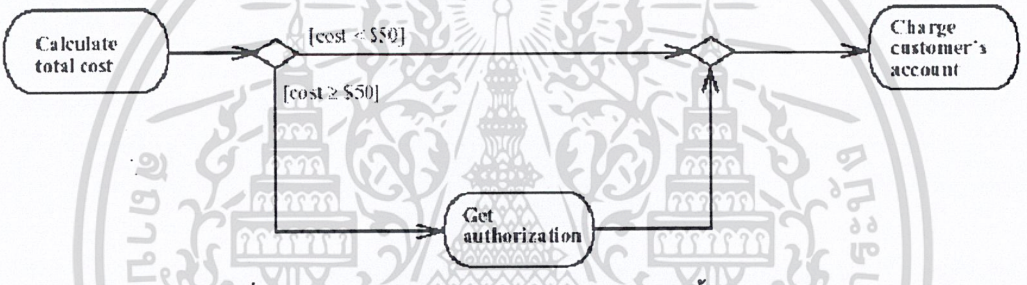
รูปแบบการใช้แอ็กทิวิตี้ไดอะแกรม มีได้หลายแบบ ได้แก่

1. แบบมีทางเลือกให้ตัดสินใจ

เราสามารถกำหนดทางเลือกให้แก่แอ็กทิวิตี้ไดอะแกรม ได้ 2 วิธีคือ

- ลากลูกศรของแต่ละทางเลือกไปยังแอ็กทิวิตี้ผลลัพธ์ของทางเลือกโดยตรง
- ลากลูกศรให้ลูกศรของแต่ละทางเลือกผ่านรูปสี่เหลี่ยมขนมเปียกปูนเสียก่อน (ลักษณะเดียวกันที่เขียนโฟลว์ชาร์ต)

แสดงได้ดังรูปตัวอย่างต่อไปนี้

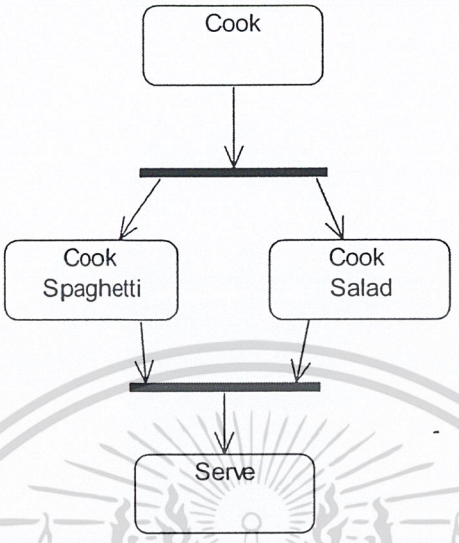


รูปที่ 3 – 16 แสดงการสร้างทางเลือกให้แก่แอ็กทิวิตี้ไดอะแกรม

2. แบบที่มีการทำงานพร้อมๆ กันหลายงาน

ในกรณีที่เรามีงานหลายงานที่มีการทำงานไปพร้อมๆ กัน ในการเขียนแอ็กทิวิตี้ไดอะแกรม เราจะใช้เส้นตรงแนวนอนเส้นหนามาเป็นสัญลักษณ์ที่ใช้จัดกลุ่มงานที่มีการทำพร้อมๆ กัน โดยมีลักษณะดังตัวอย่างต่อไปนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



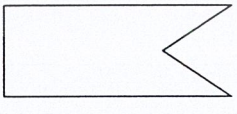
รูปที่ 3 – 17 แสดงสัญลักษณ์ที่ใช้ในการจัดกลุ่มงานที่ทำพร้อมกันในแอ็กทิวิตี้ไดอะแกรม

3. แอ็กทิวิตี้ไดอะแกรมสำหรับการส่งสัญญาณ

ในกระบวนการทำงานอาจเป็นไปได้ว่าจะมีการส่งสัญญาณบางอย่างในระหว่างการทำงาน เมื่อเกิดการส่ง-รับสัญญาณ เราก็จะเรียกว่าเกิด "แอ็กทิวิตี้" ขึ้นเช่นเดียวกัน

ในการเขียน แอ็กทิวิตี้ไดอะแกรม สำหรับการส่งสัญญาณ เราจะใช้รูปหลายเหลี่ยมแทนแอ็กทิวิตี้ที่มีการส่งสัญญาณ แสดงได้ดังรูป

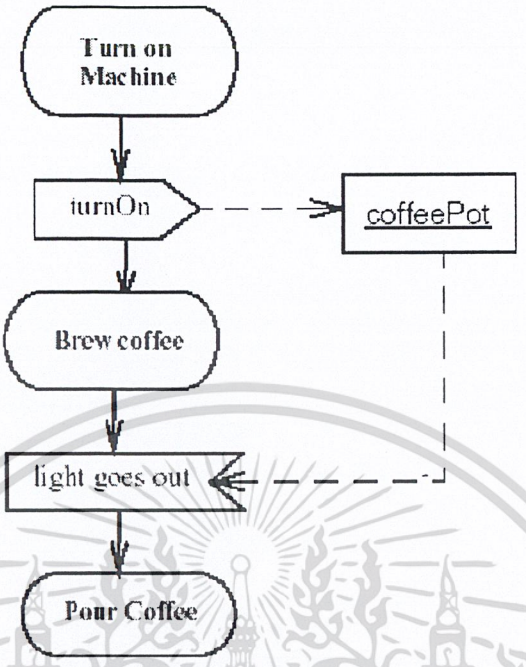
รูปที่ 3 – 18 แทนเหตุการณ์ (event) ที่เป็นอินพุท



รูปที่ 3 – 19 แทนเหตุการณ์ (event) ที่เป็นเอาท์พุท

รูปต่อไปเป็นตัวอย่างของการใช้แอ็กทิวิตี้ไดอะแกรม แสดงการส่งสัญญาณ โดยระบบที่สนใจคือการชงกาแฟโดยใช้กระติกน้ำร้อน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3 - 20 แสดงตัวอย่างการใช้สัญลักษณ์การส่งสัญญาณในแอ็กทิวิตี้ไดอะแกรม

Swimlanes

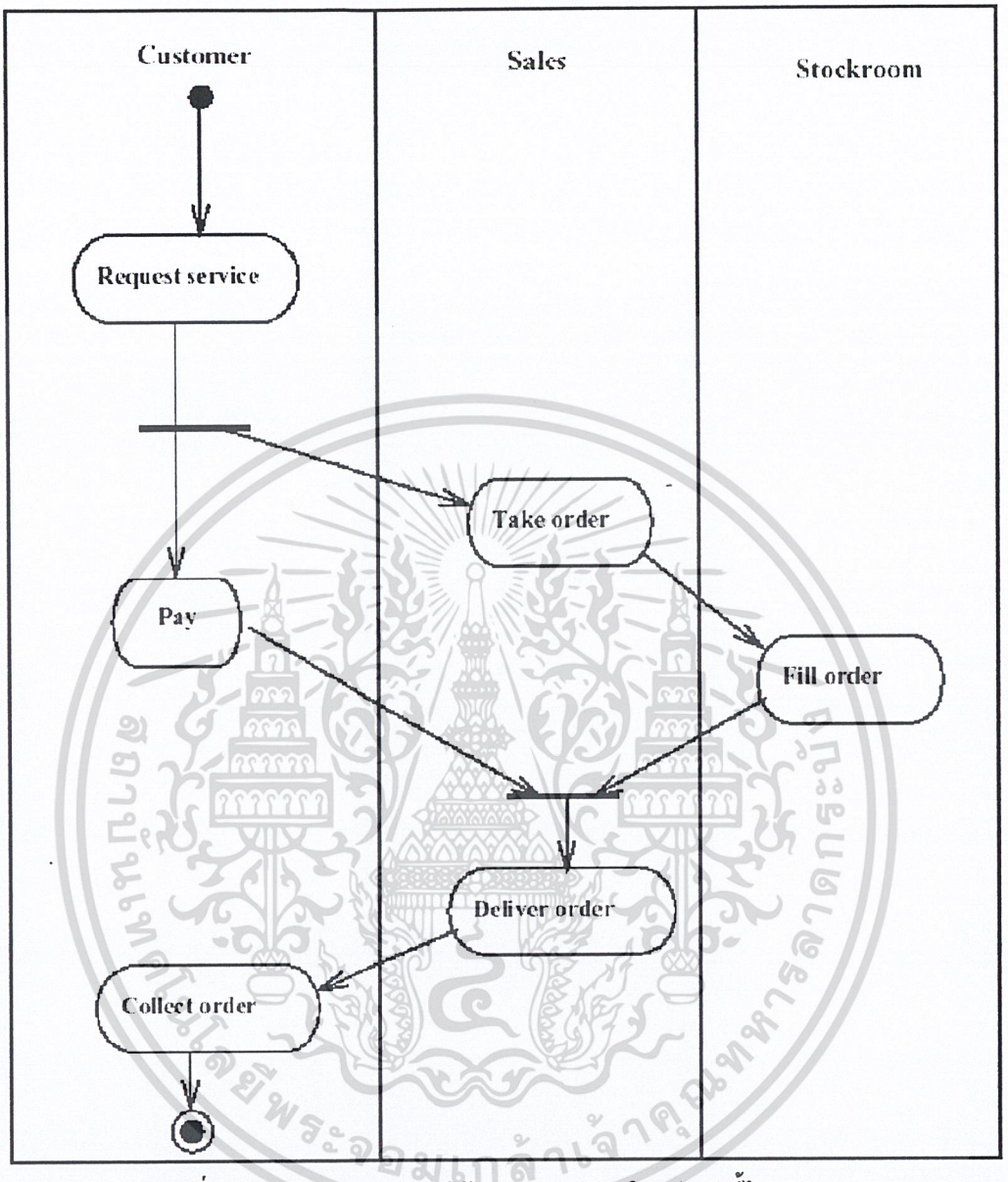
คุณลักษณะอย่างหนึ่งของการใช้แอ็กทิวิตี้ไดอะแกรมคือเราสามารถแสดงให้เห็นได้ว่าใครเป็นผู้มีหน้าที่รับผิดชอบในแต่ละแอ็กทิวิตี้ในกระบวนการทำงานหนึ่ง ๆ

หลักการของการแสดงหน้าที่ความรับผิดชอบของแต่ละแอ็กทิวิตี้ เราจะทำโดยการแบ่งกลุ่มของการรับผิดชอบเป็นกลุ่มๆ ซึ่งแอ็กทิวิตี้ไดอะแกรมมีลักษณะการแบ่งกลุ่มการรับผิดชอบด้วย เรียกกลไกดังกล่าวว่า “Swimlanes”

ในแต่ละ Swimlanes จะมีการกำหนดชื่อกำกับเอาไว้ เช่น ในกระบวนการของการสั่งซื้อสินค้าเราอาจแบ่งกลุ่มของคนที่มีส่วนเกี่ยวข้องเป็น 3 ส่วน หรือ 3 Swimlanes ได้แก่ ลูกค้า, ฝ่ายขาย และคลังสินค้า เป็นต้น

แอ็กทิวิตี้หนึ่งๆ จะอยู่ภายใน 1 Swimlane เท่านั้น แต่การติดต่อหรือส่งผ่านระหว่างแอ็กทิวิตี้สามารถเกิดขึ้นข้ามจาก Swimlane หนึ่งไปยังอีก Swimlane หนึ่งได้ ดังรูป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3 – 21 แสดงตัวอย่างการใช้งาน Swimlanes ในแอ็กทิวิตี้ไดอะแกรม

จะเห็นว่าแอ็กทิวิตี้ไดอะแกรม มีลักษณะคล้ายคลึงกับโฟลว์ชาร์ตมาก โดยจะแสดงถึงลำดับขั้นตอนการทำงาน จุดที่มีการตัดสินใจ และการแตกการทำงานเป็นส่วนย่อยๆ ด้วยลักษณะดังกล่าว แอ็กทิวิตี้ไดอะแกรมจึงมีประโยชน์มากสำหรับใช้อธิบายถึงการทำงานของออบเจ็กต์ต่างๆ รวมไปถึงกระบวนการทำงานต่างๆ ด้วย

เราอาจสงสัยว่าทั้งแอ็กทิวิตี้ไดอะแกรมและสเตทไดอะแกรม ต่างก็ใช้แสดงขั้นตอนการทำงานซึ่งระบบงานเกิดการเปลี่ยนแปลง ดังนั้นเราจะมีหลักอย่างไรว่าเมื่อไรควรจะใช้แอ็กทิวิตี้ไดอะแกรมหรือเมื่อไรเราควรใช้สเตทไดอะแกรม

ข้อแตกต่างของทั้งสองไดอะแกรมอยู่ที่ลักษณะของการเปลี่ยนแปลงของระบบ ถ้าเป็นการเปลี่ยนแปลงหรือเหตุการณ์ต่างๆ อันเป็นผลมาจากการกระทำภายในระบบเองซึ่งมีลักษณะต่อเนื่องกัน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ แสดงต่อสาธารณะ ทำซ้ำ หรือเพื่อการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

และมักมีจุดที่ระบบงานต้องมีการตัดสินใจ ให้ใช้แอ็กทิวิตี้ไดอะแกรมแต่ถ้าเป็นการเปลี่ยนแปลงหรือการกระทำที่เกิดขึ้นในลักษณะไม่ต่อเนื่องกัน ให้ใช้สเตทไดอะแกรม

เมื่อเราสร้างแอ็กทิวิตี้ไดอะแกรมขึ้นมา เราต้องไม่ลืมว่าแอ็กทิวิตี้ไดอะแกรมเป็นเพียงแบบจำลองหนึ่งของภาษา UML ที่พยายามสร้างมุมมองแบบไดนามิกของระบบ ดังนั้น จึงไม่มีแอ็กทิวิตี้ไดอะแกรมใดจะสามารถอธิบายทุกๆ อย่างของระบบได้ภายใน Diagram เดียว เราจำเป็นต้องใช้แอ็กทิวิตี้ไดอะแกรมมาช่วยมากกว่า 1 ไดอะแกรม จึงจะเห็นกระบวนการทำงานได้อย่างครบถ้วน

แอ็กทิวิตี้ไดอะแกรมเป็นไดอะแกรมที่ค่อนข้างสร้างแล้วเข้าใจได้ง่าย อย่างไรก็ตาม แอ็กทิวิตี้ไดอะแกรมที่ดีควรมีคุณสมบัติต่อไปนี้

- มุ่งเน้นไปที่การติดต่อสื่อสารของระบบเชิงไดนามิก กล่าวคือ เป็นการติดต่อที่สภาพของระบบมีการเปลี่ยนแปลงไม่หยุดนิ่ง
- เฉพาะส่วนที่มีความสำคัญต่อกระบวนการทำงานเท่านั้น
- แสดงรายละเอียดในแต่ละระดับของการทำงาน ซึ่งรายละเอียดดังกล่าวจะเลือกแสดงเฉพาะที่มีความสำคัญต่อการเข้าใจการทำงานของระบบเท่านั้น
- ถ้าการทำงานส่วนใดมีความสำคัญแอ็กทิวิตี้ไดอะแกรมก็ควรแสดงให้เห็น ไม่ควรละเอาไว้หรือแสดงอย่างย่นย่อ

3.4.6 คลาสไดอะแกรม (Class Diagram)

โดยการนำคลาสต่างๆ ที่ใช้ในซีเควนต์ไดอะแกรมหรือคอลแลโบเรชั่นไดอะแกรมมาอธิบายถึงโครงสร้างที่ประกอบด้วยแอตทริบิวต์ต่าง ๆ และ ความสัมพันธ์ระหว่างคลาสและนำส่วนของเมสเซจมาสร้างฟังก์ชันของคลาส โดยโครงสร้างที่มีข้อมูลรวมกับพฤติกรรมของข้อมูลซึ่งจะนำไปเป็นแนวทางของการสร้างคลาสในการพัฒนาโปรแกรมเชิงวัตถุได้โดยแต่ละคลาสจะประกอบไปด้วย 3 ส่วน ดังรูป ส่วนแสดงความสัมพันธ์ระหว่างคลาสมี 4 แบบ

1. แอสโซซิเอชั่น จะแสดงความสัมพันธ์ระหว่างคลาส มีได้ทั้งทางเดียวและสองทาง

2. เอกกรีเกชั่น เป็นรูปแบบพิเศษของแอสโซซิเอชั่น คือเป็นความสัมพันธ์ระหว่าง Whole ของมัน คือ Whole จะประกอบไปด้วย parts ต่างๆ ของมัน ดังนั้นการคงอยู่ของ parts จะต้องขึ้นอยู่กับ Whole หรือเรียกความสัมพันธ์ว่า "Whole-part" ก็ได้

3. Depends On เป็นรูปแบบความสัมพันธ์แบบหนึ่งที่ใช้แสดงความสัมพันธ์ระหว่างคลาสสองคลาส ในแง่ที่คลาสหนึ่งเรียกใช้บริการของอีกคลาสหนึ่ง กล่าวคือ คลาสของผู้ขอบริการขึ้นอยู่กับบริการของคลาสผู้ให้บริการ แต่ไม่มีการขึ้นต่อกัน

4. Generalization ความสัมพันธ์รูปแบบนี้ใช้แสดงความสัมพันธ์ระหว่างคลาสกับคลาสในแง่ที่คลาสถ่ายทอดคุณสมบัติและโครงสร้างจากอีกคลาสหนึ่ง โดยเรียกคลาสที่ถูกถ่ายทอดว่า "Super class" และเรียกคลาสที่ทำการถ่ายทอดว่า "Sub class"

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.4.7 ออบเจ็กต์ไดอะแกรม (Object Diagram)

เป็นไดอะแกรมสำหรับแสดงความสัมพันธ์ของวัตถุหรืออินสแทนซ์ในคลาสต่างๆ โดยมีสัญลักษณ์เหมือนกับ class diagram ยกเว้นการแสดงชื่อ

3.4.8 คอมโพเน็นต์ไดอะแกรม (Component Diagram)

เป็นไดอะแกรมที่แสดงโครงสร้างและความเกี่ยวข้องกันของซอฟต์แวร์ โดยคอมโพเน็นต์ประกอบไปด้วย source code และ runtime หรือ executable component คอมโพเน็นต์ไดอะแกรมคือกราฟที่แสดงองค์ประกอบต่างๆ ของระบบที่เชื่อมโยงกันโดยใช้ความสัมพันธ์ในลักษณะ dependency โดยจะแสดงในรูปของเส้นประที่มีหัวลูกศรชี้จากคอมโพเน็นต์ลูกไปยังคอมโพเน็นต์หลัก

3.4.9 ดีพลอยเมนต์ไดอะแกรม (Deployment Diagram)

เป็นไดอะแกรมสำหรับแสดงที่อยู่ของส่วนประมวลผลและซอฟแวร์ของระบบงาน ส่วนซอฟแวร์อื่นๆ ที่ใช้งานจะแสดงไว้ในคอมโพเน็นต์ไดอะแกรมแล้วออบเจ็กต์ไดอะแกรม

เป็นไดอะแกรมสำหรับแสดงความสัมพันธ์ของวัตถุหรืออินสแทนซ์ในคลาสต่างๆ โดยมีสัญลักษณ์เหมือนกับคลาสไดอะแกรมยกเว้นการแสดงชื่อ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 4
แบบจำลองสำหรับการออกแบบฐานข้อมูลเชิงวัตถุ OONIAM

4.1 บทนำ

จากการศึกษาค้นคว้าเกี่ยวกับแบบจำลองสำหรับการออกแบบฐานข้อมูลเชิงวัตถุยังไม่พบแบบจำลองมาตรฐานสำหรับการออกแบบโดยตรง ดังนั้นโครงงานนี้จึงเลือกใช้ OONIAM ซึ่งเป็นแบบจำลองที่พัฒนามาจาก NIAM ที่อาจารย์ศุภมิตร และอาจารย์บัณฑิตร่วมกันคิดขึ้น เพื่อแก้ปัญหาการออกแบบด้วย UML ดังที่เกริ่นไว้ โดยในหัวข้อที่ 4.2 จะอธิบายถึง NIAM แบบจำลอง จากนั้นหัวข้อที่ 4.3 จะอธิบายถึงวิธีการใช้ OONIAM แบบจำลอง

4.2 ความหมายของ NIAM และการใช้

ไนแอม (NIAM : Nijssen's Information Analysis Methodology) เป็นวิธีการในการออกแบบฐานข้อมูล โดยการแสดงความหมาย, ความสัมพันธ์ และข้อจำกัดต่างๆ ของข้อมูลด้วยแบบจำลองข้อมูลที่ประกอบไปด้วยสัญลักษณ์ต่างๆ เนื่องจากแนวคิดที่ให้โครงร่างแนวความคิด (Conceptual Schema) มีพื้นฐานมาจากโครงสร้างภาษาธรรมชาติ ใช้รูปประโยคที่มีประธาน กริยา กรรม วิธีแสดงรูปแบบความสัมพันธ์ของข้อมูลและข้อจำกัดของข้อมูลได้อย่างชัดเจน นอกจากนั้นยังสามารถแปลงโครงร่างแนวความคิดเป็นโครงร่างฐานข้อมูลสัมพันธ์ ซึ่งจะอยู่ในรูปของ Fifth Normal Form ได้โดยตรง และเนื่องจากวิธีการนี้ใช้รูปสัญลักษณ์ที่แสดงความสัมพันธ์ของข้อมูลและง่ายต่อการเข้าใจ ดังนั้นจึงสะดวกในการออกแบบฐานข้อมูลของระบบงานใหญ่ ๆ

ไนแอมมีขั้นตอนในการออกแบบอยู่ 9 ขั้นตอน (CSDP 9 steps : Conceptual Schema Design Procedure)

1. กำหนดขอบเขตของงาน (Universe of Discourse : UOD) และความจริงที่เกิดขึ้นภายในขอบเขตของงานที่กำหนดไว้

2. วาดไดอะแกรมโครงร่างแนวความคิด (Conceptual Schema Diagram) โดยคร่าวๆ จากความจริงในขอบเขตของงาน

3. จัดรูปของโครงร่าง (Schema) ให้เป็นระเบียบและหาชนิดความจริงที่ได้รับข้อมูลมาจากชนิดความจริง

4. เติมสัญลักษณ์แสดง Uniqueness constraints

5. ตรวจสอบความถูกต้องของชนิดความจริง

6. เติมสัญลักษณ์แสดง Lexical, Mandatory Role, Subtype constraints

7. ตรวจสอบ Unique Identifier ของแต่ละชนิดเอนติตี

8. เติมสัญลักษณ์แสดง Equality, Exclusion, Subset constraints

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

9. ตรวจสอบความสมบรูณ์ของโครงร่างแนวความคิดว่าต้องสอดคล้องกับตัวอย่างข้อมูลและไม่มีความซ้ำซ้อนของข้อมูล

4.2.1 ส่วนประกอบพื้นฐานของไนแอม

ส่วนประกอบพื้นฐาน ประกอบไปด้วย

ชนิดเอนติตี (Entity Type) หมายถึง เซตของสิ่งที่สนใจทั้งที่อยู่ในรูปของนามธรรม หรือรูปธรรม ซึ่งอาจเป็นสิ่งที่จับต้องได้หรือไม่ได้ เช่น คน, ภาควิชา, บริษัท, รถยนต์ เป็นต้น

ชนิดเลเบิล (Lable Type, Value Type) หมายถึง เซตของสิ่งที่ใช้บ่งบอกความแตกต่าง หรือชื่อของแต่ละเอนติตีที่กำหนด เช่น ชื่อ, นามสกุล, รหัสประจำตัว, ทะเบียนรถยนต์ เป็นต้น

บทบาท (Role) หมายถึง ความสัมพันธ์ที่เกี่ยวข้องกับชนิดเอนติตีที่สัมผัสอยู่ เช่น เอนติตีนักศึกษา แสดงบทบาท เป็นผู้ลงทะเบียนเรียนในวิชานั้น ๆ เป็นต้น

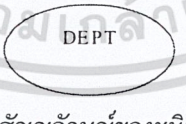
ประโยคความจริงมูลฐาน (Element Fact Type) หรืออาจเรียกว่าชนิดความจริง (Fact Type) หมายถึง เซตของความสัมพันธ์ระหว่างสมาชิกของชนิดเอนติตีตั้งแต่ 2 เอนติตีขึ้นไป โดยขนาดของชนิดความจริงจะขึ้นอยู่กับจำนวนบทบาทที่เกี่ยวข้อง โดยที่ชนิดความจริงที่มีจำนวน 2 บทบาท จะเรียกว่า Binary fact type ส่วนชนิดความจริงที่มี 3 บทบาท จะเรียกว่า Ternary fact type สำหรับชนิดความจริงที่มีมากกว่า 3 บทบาทขึ้นไป จะรวมเรียกว่า n-ary fact type

ชนิดอ้างอิง (Reference Type) หมายถึง เซตของความสัมพันธ์ระหว่างสมาชิกของชนิดเอนติตีกับสมาชิกของชนิดเลเบิลที่มีอยู่

ชนิดความจริงแบบเนส (Nested Fact Type) หมายถึง ชนิดเอนติตีที่แสดงความสัมพันธ์ในการกำหนดกลุ่มของชนิดความจริงที่มีตั้งแต่ 2 บทบาทขึ้นไป

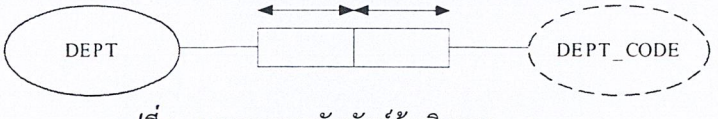
กฎข้อบังคับความถูกต้องของข้อมูล (Integrity Constraints) หมายถึง สิ่งที่ใช้แสดงกฎที่ใช้ในการบังคับควบคุมความถูกต้องของข้อมูล

สัญลักษณ์และตัวอย่างการใช้ส่วนประกอบพื้นฐานของแบบจำลองไนแอมแสดงไว้ดังรูป



รูปที่ 4 - 1 แสดงสัญญลักษณ์ของชนิดเอนติตีภาควิชา

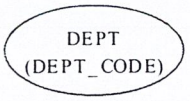
รูปที่ 4 - 2 แสดงสัญญลักษณ์ของชนิดเลเบิลรหัสภาควิชา



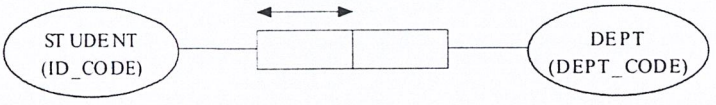
รูปที่ 4 - 3 แสดงความสัมพันธ์อ้างอิงแบบ one to one

หมายความว่า ภาควิชาใดๆ จะมีรหัสภาควิชาเพียงรหัสเดียวเท่านั้นและไม่ซ้ำกับภาควิชาอื่น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

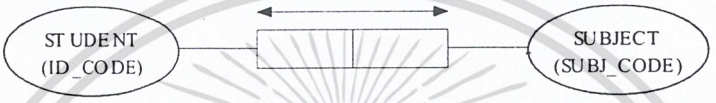


รูปที่ 4 - 4 แสดงการเขียนความสัมพันธ์อ้างอิงแบบ one to one อย่างย่อ



รูปที่ 4 - 5 แสดงความจริงแบบ many to one

หมายความว่านักศึกษาหนึ่งคนจะสังกัดภาควิชาได้เพียงภาคเดียว แต่ภาควิชาใดๆ สามารถมีนักศึกษาในสังกัดได้มากกว่าหนึ่งคน



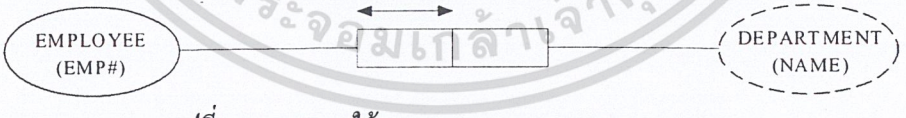
รูปที่ 4 - 6 แสดงความจริงแบบ many to many

หมายความว่านักศึกษาหนึ่งคนสามารถลงทะเบียนเรียนได้หลายวิชาและแต่ละวิชาที่เปิดสอนสามารถรับจำนวนได้มากกว่าหนึ่งคนแต่นักศึกษาที่ลงทะเบียนเรียนวิชาใดๆ แล้ว จะลงทะเบียนซ้ำวิชาเดิมไม่ได้ (ตัวอย่างนี้ไม่เป็นจริงในทางปฏิบัติ)

ชนิดเอนติตีเป็นเซต ซึ่งมีสมาชิกเป็นเอนติตีอินสแทนซ์ เช่น ภาควิชา A (ภาควิชาโทรคมนาคม) ภาควิชา B (ภาควิชาวิศวกรรมคอมพิวเตอร์) เป็นตัวอย่างเอนติตีของชนิดเอนติตีภาควิชา

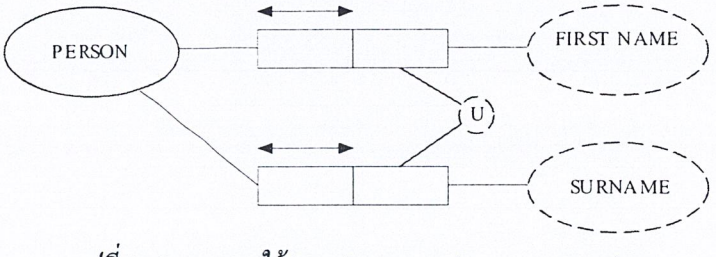
เครื่องหมายความสัมพันธ์ที่เป็นส่วนเชื่อมโยงระหว่างชนิดเอนติตี และชนิดเอนติตีหรือชนิดเลเบิลนั้น เรียกว่า บทบาท (role) จะเขียนความหมายของบทบาทนั้นไว้ภายในหรือข้าง ๆ สัญลักษณ์ของมัน

การแปลงข้อมูลที่วิเคราะห์มาให้อยู่ในรูปแบบจำลอง ก่อนอื่นต้องนำข้อมูลมากำหนดเป็นชนิดเอนติตีและเลเบิลให้เรียบร้อยเสียก่อน จึงนำชนิดเอนติตีที่ได้มาเขียนเป็นประโยคความจริงมูลฐาน (Elementary Fact) แล้วเอาความจริงทั้งหมดที่ได้มาเขียนเป็นแบบจำลอง และเติมข้อจำกัดต่าง ๆ ลง ไปตามความเป็นจริงในขอบเขตของงาน ตัวอย่างการใช้ข้อจำกัดต่างๆ แสดงไว้ดังรูปที่ 4 - 7 และรูปที่ 4 - 8



รูปที่ 4 - 7 แสดงการใช้ intra fact type uniqueness constraint

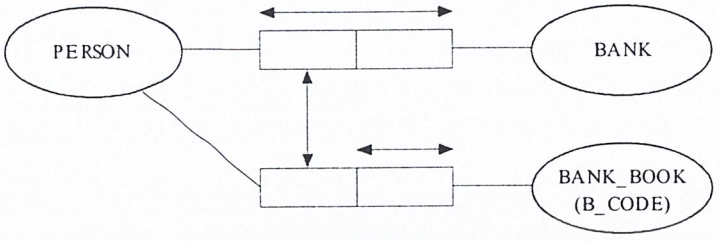
หมายความว่า Employee หนึ่งคนจะมีที่ทำงานได้ที่เดียวเท่านั้น



รูปที่ 4 - 8 แสดงการใช้ inter fact type uniqueness constraint

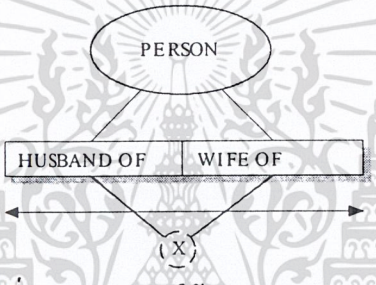
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

หมายความว่า บุคคลหนึ่งจะมีชื่อ 1 ชื่อ นามสกุล 1 นามสกุล ชื่อของบางคนอาจจะซ้ำกัน และนามสกุลจะต้องไม่ซ้ำกัน



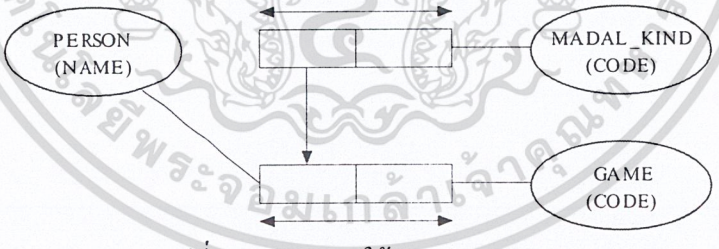
รูปที่ 4 - 9 แสดงการใช้ equality constraint

หมายความว่า ถ้าบุคคลหนึ่งเป็นลูกค้าของธนาคารใดแล้ว บุคคลนั้นต้องมีสมุดบัญชีของธนาคารนั้นด้วย หรือในทางกลับกัน ถ้าบุคคลใดมีสมุดบัญชีของธนาคารใดแล้ว ก็ต้องเป็นลูกค้าของธนาคารนั้นด้วย



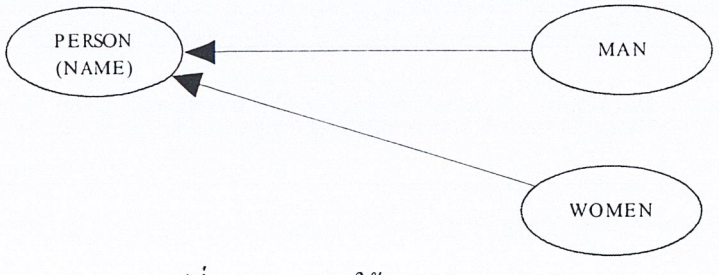
รูปที่ 4 - 10 แสดงการใช้ exclusion constraint

หมายความว่า บุคคลใดเป็นภรรยาของอีกบุคคลหนึ่งแล้ว บุคคลนั้นต้องไม่เป็นสามีของบุคคลใดๆ ในทางกลับกัน บุคคลที่เป็นสามีของบุคคลหนึ่งแล้วจะต้องไม่เป็นภรรยาของบุคคลใดด้วย



รูปที่ 4 - 11 แสดงการใช้ subset constraint

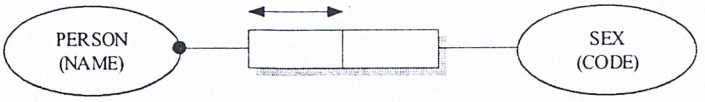
หมายความว่า บุคคลที่ชนะกีฬาการแข่งขันกีฬาทุกคนจะต้องเป็นบุคคลที่เล่นกีฬา แต่บุคคลที่เล่นกีฬาไม่จำเป็นต้องชนะการแข่งขันกีฬาทุกคน



รูปที่ 4 - 12 แสดงการใช้ subtype constraint

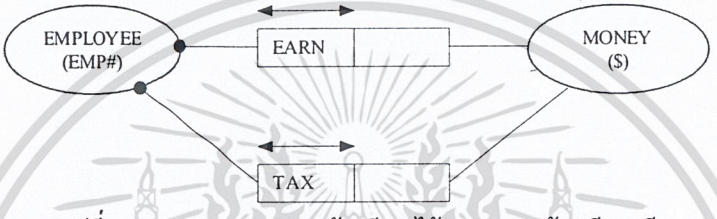
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

หมายความว่า ตัวอย่างเอนติตีทุกตัวของชนิดเอนติตีผู้ชาย และชนิดเอนติตีผู้หญิง ต่างก็เป็นสมาชิกของชนิดเอนติตีบุคคล



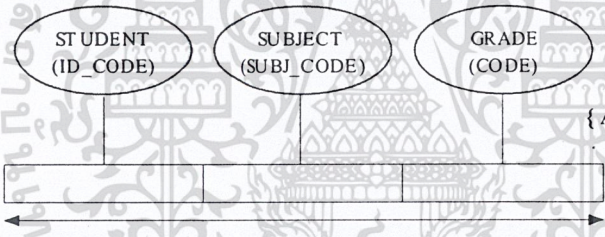
รูปที่ 4-13 แสดงการใช้ mandatory constraint, lexical constraint

หมายความว่าบุคคลทุกคนต้องมีเพศและสมาชิกของชนิดเอนติตีเพศมีเพียง M (Male) และ F (Female) เท่านั้น

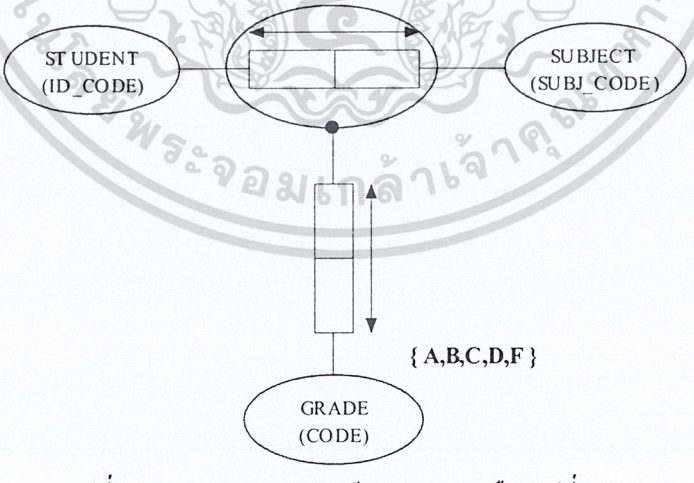


รูปที่ 4 - 14 Employee ทุกคนต้องมีรายได้ และทุกคนต้องเสียภาษี

นอกจากตัวอย่างความจริงแบบ Binary Fact Type ที่แสดงไว้ข้างต้นแล้ว ยังมีตัวอย่างชนิดอื่นอีกเช่น



รูปที่ 4 - 15 Ternary Fact Type หมายความว่า ข้อมูลการเรียนของนักศึกษาทุกคนจะต้องมีทั้งรหัสวิชาและเกรด



รูปที่ 4 - 16 Nested Fact Type มีความหมายเหมือนรูปที่ 4-15

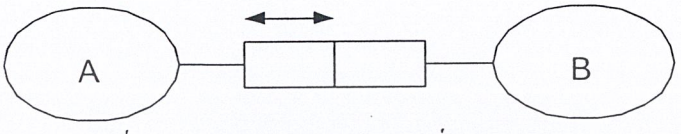
4.2.2 กฎข้อบังคับความถูกต้องของข้อมูลที่ใช้ในแบบจำลองระดับแนวคิดไนแอม

4.2.2.1 Intra fact type constraints (internal uniqueness constraints)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เป็นกฎข้อบังคับความถูกต้อง เพื่อทำการกำหนดบทบาทที่ใช้แสดงความสัมพันธ์ระหว่างสมาชิกของชนิดเอนติตีหนึ่งกับสมาชิกของชนิดเอนติตีอื่น หรือกับสมาชิกของชนิดเลเบิล โดยสามารถแบ่งเป็นรูปแบบต่าง ๆ ได้ดังต่อไปนี้

การกำหนดความสัมพันธ์แบบหนึ่งต่อหลายหน่วย (one to many relationship) ซึ่งสามารถแสดงบนแผนภาพได้ดังรูป



รูปที่4 - 17 แสดงความสัมพันธ์แบบหนึ่งต่อหลายหน่วย

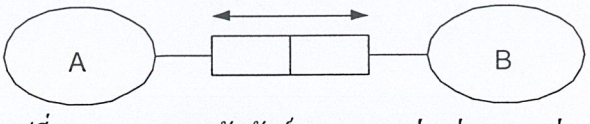
ลักษณะเช่นนี้สามารถแสดงได้ว่า ชนิดเอนติตี A จะแสดงความสัมพันธ์กับ ชนิดเอนติตี หรือ ชนิดเลเบิล B ได้อย่างมากที่สุดเพียงหนึ่งความสัมพันธ์เท่านั้น แต่ในทางกลับกัน ชนิดเอนติตี หรือ ชนิดเลเบิล B จะแสดงความสัมพันธ์กันชนิดเอนติตี A ได้หลายความสัมพันธ์ โดยกฎข้อบังคับความถูกต้องจะต้องทำการควบคุมไม่ให้เกิดการซ้ำซ้อนของข้อมูลในคอลัมน์ A ขึ้นได้ เช่น คนหนึ่งคนจะมีมารดาได้เพียงคนเดียวเท่านั้น แต่ในทางกลับกัน คนเพียงคนเดียวอาจเป็นมารดาของคนหลายคนได้

การกำหนดความสัมพันธ์แบบหนึ่งต่อหนึ่งหน่วย (one to one relasionship) ซึ่งสามารถแสดงบนแผนภาพได้ดังรูป

รูปที่ 4 - 18 แสดงความสัมพันธ์แบบหนึ่งต่อหนึ่งหน่วย

ลักษณะเช่นนี้สามารถแสดงได้ว่า ชนิดเอนติตี A จะแสดงความสัมพันธ์กับ ชนิดเอนติตี หรือ ชนิดเลเบิล B ได้เพียงหนึ่งความสัมพันธ์เท่านั้น โดยกฎข้อบังคับจะทำการควบคุมไม่ให้เกิดความสัมพันธ์ของข้อมูลมากกว่าหนึ่งความสัมพันธ์ เช่น คนหนึ่งคนจะมีเลขประจำตัวได้เพียงหมายเลขเดียวเท่านั้น และในทางกลับกัน หมายเลขประจำตัวหนึ่งหมายเลขจะต้องหมายถึงคนเพียงคนเดียว

การกำหนดความสัมพันธ์แบบหลายต่อหลายหน่วย (many to many relationship) ซึ่งสามารถแสดงบนแผนภาพได้ดังรูป



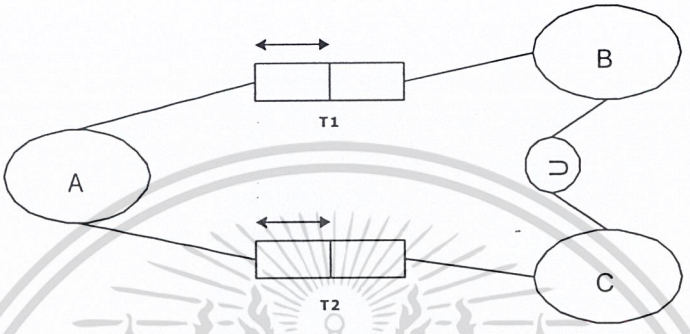
รูปที่ 4 - 19 แสดงความสัมพันธ์แบบหลายหน่วยต่อหลายหน่วย

ลักษณะเช่นนี้สามารถแสดงได้ว่า ชนิดเอนติตี A จะแสดงความสัมพันธ์กับ ชนิดเอนติตี B ได้หลายความสัมพันธ์ และในทางกลับกัน ชนิดเอนติตี B ก็จะแสดงความสัมพันธ์กับ ชนิดเอนติตี A ได้หลายความสัมพันธ์เช่นกัน โดยกฎข้อบังคับความถูกต้องจะต้องทำการควบคุมความสัมพันธ์ A และ B ไม่ให้เกิดความซ้ำซ้อนเกิดขึ้นได้ เช่น นักศึกษาหนึ่งคนอาจลงทะเบียนเรียนได้หลายวิชา และวิชาใดๆ ก็สามารถรองรับนักศึกษาได้หลายคน แต่นักศึกษาหนึ่งคนจะไม่สามารถลงทะเบียนวิชาใดๆ ได้มากกว่า 1 ครั้งของการลงทะเบียน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.2.2.2 Inter fact type uniqueness constraints (external uniqueness constraints)

เป็นกฎข้อบังคับความถูกต้องที่แสดงให้เห็นว่าชนิดเอนติตีใดๆ มีความสัมพันธ์กับชนิดเลเบิล หรือ ชนิดเอนติตี ได้มากกว่าหนึ่ง โดยในทางกลับกัน ชนิดเลเบิล หรือ ชนิดเอนติตี เหล่านั้น สามารถบ่งบอกถึงลักษณะเฉพาะของชนิดเอนติตีนั้นได้ดังแสดงในแผนภาพดังนี้



รูปที่ 4 - 20 แสดง Inter fact type uniqueness constrains

ลักษณะเช่นนี้สามารถแสดงได้ว่ากฎข้อบังคับความถูกต้องจะทำการควบคุม หากนำ T1 join T2 แล้วผลที่ได้ BC จะไม่เกิดความซ้ำซ้อนกันขึ้น เช่น คนหนึ่งคนอาจมีชื่อหรือนามสกุลซ้ำกันได้ แต่ถ้า รวมทั้งชื่อและนามสกุลแล้วจะไม่เกิดความซ้ำซ้อนดังนั้นจะสามารถบ่งบอกได้ว่าเป็นการระบุถึงคนใด

4.2.2.3 Mandatory role constraints

เป็นกฎข้อบังคับความถูกต้องที่ใช้ในการควบคุมเพื่อแสดงให้เห็นถึงการมีอยู่ของข้อมูลว่าต้องมีการบันทึกข้อมูลทุกครั้งที่เกิดมีความสัมพันธ์เกิดขึ้น สามารถแสดงได้ในแผนภาพดังนี้

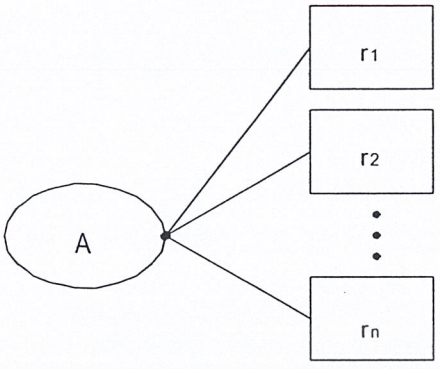
รูปที่ 4 - 21 แสดง Mandatory role constraints

จากภาพจุดทึบที่เชื่อมต่อระหว่าง ชนิดเอนติตี กับ Role นั้น แสดงให้เห็นว่าสมาชิกทุกตัวใน ชนิดเอนติตี A จะต้องถูกบันทึกข้อมูลเมื่อมีบทบาท r เกิดขึ้น โดยแสดงให้เห็นว่า pop(A) = pop(r) เช่น นักศึกษาทุกคนต้องมีการบันทึกชื่อและนามสกุล เป็นต้น

4.2.2.4 Inclusion mandatory role constraints

เป็นกฎข้อบังคับความถูกต้องที่แสดงให้เห็นถึงทางเลือกของบทบาทในกลุ่มของความสัมพันธ์ ที่มีอยู่ว่าต้องมีการบันทึกข้อมูลอย่างน้อยบทบาทใดบทบาทหนึ่งของชนิดเอนติตีนั้น ดังแสดงในแผนภาพได้ดังนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

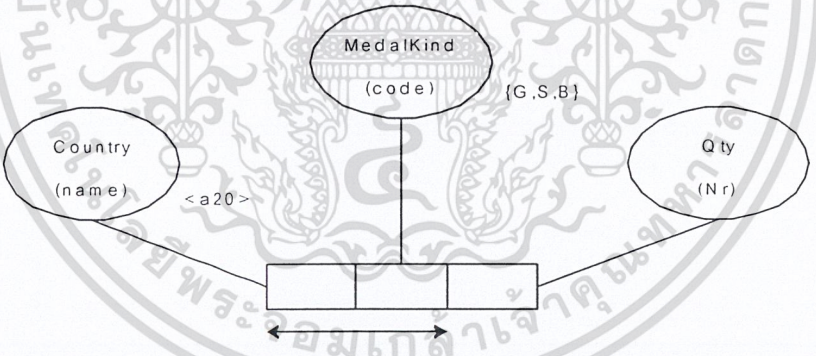


*รูปที่ 4 - 22 แสดง **Inclusion mandatory role constraints***

จากภาพสามารถแสดงกฎข้อบังคับความถูกต้องของข้อมูล คือสมาชิกของชนิดเอนติตี A ใดๆ ต้องมีการบันทึกความสัมพันธ์เกิดขึ้นความสัมพันธ์ในความสัมพันธ์หนึ่ง ซึ่งแสดงได้ว่า $pop(A) = pop(r_1) \cup pop(r_2) \cup ... \cup pop(r_n)$ เช่น บุคคลใดๆ จะต้องมีการระบุข้อมูลของบุตร หรือข้อมูลของบิดามารดาของบุคคลนั้นๆ อย่างน้อยที่สุดหนึ่งข้อมูล

4.2.2.5 Entity type constraints (Value constraints)

เป็นกฎข้อบังคับความถูกต้อง ที่ใช้ในการกำหนดค่าของสมาชิกภายในเซตของข้อมูลที่เป็นไปได้ของชนิดเลเบิล หรือ ชนิดเอนติตีหนึ่งๆ รวมไปถึงการกำหนดชนิดของข้อมูลในเซตด้วย ดังแสดงได้ในแผนภาพดังนี้



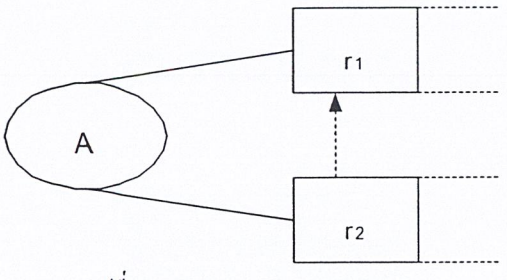
*รูปที่ 4-23 แสดง **Entity type constraints***

จากภาพนั้นจะมีกฎข้อคับความถูกต้องของข้อมูลเพื่อทำการระบุชนิดของเหรียญรางวัลในการแข่งขันกีฬา สามารถแยกออกได้เป็น เหรียญทองแดง, เหรียญเงิน, เหรียญทอง และระบุถึงจำนวนของเหรียญรางวัลที่ได้ว่าต้องอยู่ในช่วง 1 ถึง 200 เหรียญ รวมทั้งยังสามารถระบุชนิดของข้อมูลได้ด้วย ดังที่แสดงให้เห็นว่าชื่อประเทศนั้นกำหนดให้จัดเก็บได้มากที่สุด 20 ตัวอักษร

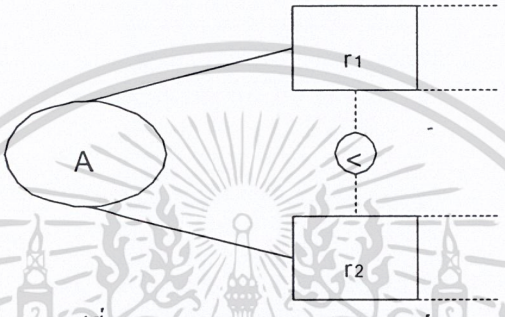
4.2.2.6 Subset constraint

เป็นกฎข้อบังคับความถูกต้องของข้อมูล ที่แสดงความสัมพันธ์ที่เป็นส่วนหนึ่งของความสัมพันธ์ที่มีอยู่ แต่จะมีลักษณะความสัมพันธ์ไปในทางเดียวดังแสดงความสัมพันธ์ได้โดยใช้สัญลักษณ์คือ $A \rightarrow B$ ซึ่งสามารถแสดงในแผนภาพได้ดังนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4 - 24 แสดง Subset constraints

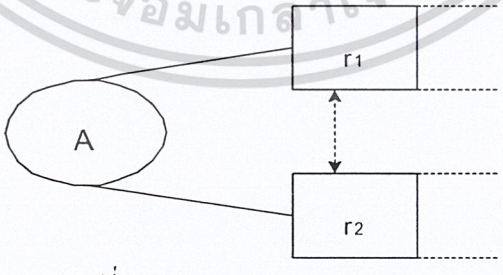


รูปที่ 4 - 24 แสดง Subset constraints (ต่อ)

ลักษณะดังกล่าวนี้แสดงให้เห็นกฎข้อบังคับความถูกต้องของข้อมูลว่า สมาชิกแต่ละตัวของชนิดเอนติตี A หากมีการบันทึกความสัมพันธ์ r2 แล้ว ต้องมีการบันทึกความสัมพันธ์ r1 ด้วย แต่ในทางกลับกัน สมาชิกแต่ละตัวของชนิดเอนติตี A หากมีการบันทึกความสัมพันธ์ r1 แล้วไม่จำเป็นต้องมีการบันทึกความสัมพันธ์ r2 ก็ได้ เช่น บุคคลที่ชนะการแข่งขันกีฬา แสดงว่าต้องเป็นนักกีฬา แต่ผู้ที่เป็นนักกีฬาไม่จำเป็นต้องเป็นผู้ชนะการแข่งขันทุกคน

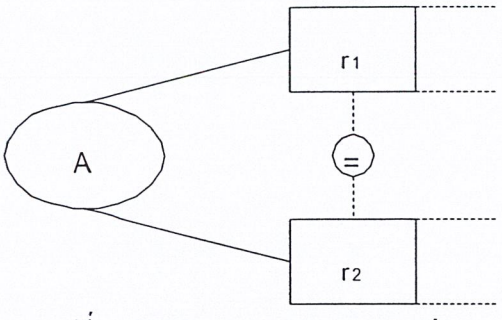
4.2.2.7 Equality constraints

เป็นกฎข้อบังคับความถูกต้องที่แสดงให้เห็นว่า ชนิดเอนติตีเหล่านั้นจะต้องมีการถูกบันทึกข้อมูลควบคู่กันเสมอไป ใช้สัญลักษณ์แสดงความสัมพันธ์ได้คือ A↔B ซึ่งสามารถแสดงในแผนภาพได้ดังนี้



รูปที่ 4 - 25 แสดง Equality constraints

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

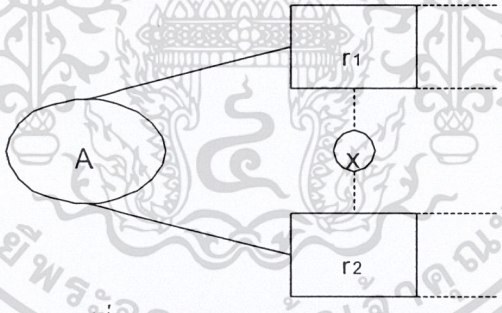


รูปที่ 4 - 25 แสดง Equality constraints (ต่อ)

ลักษณะดังกล่าวนี้ สามารถแสดงถึงกฎข้อบังคับความถูกต้องของข้อมูลว่า หากมีการบันทึกข้อมูลความสัมพันธ์ r1 ก็ต้องมีการบันทึกข้อมูลความสัมพันธ์ r2 ของสมาชิกของชนิดเอนติตี A ด้วย เช่น หากบุคคลใดจะทำการบันทึกระยะเวลาของการออกกำลังกาย ก็จะต้องทำการบันทึกข้อมูลของอัตราการเต้นของหัวใจด้วย และในทางกลับกัน หากมีการบันทึกข้อมูลอัตราการเต้นของหัวใจ ก็จะต้องทำการบันทึกข้อมูลระยะเวลาการออกกำลังกายด้วยเช่นกัน

4.2.2.8 Exclusion constraints

เป็นกฎข้อบังคับความถูกต้องที่มีลักษณะตรงข้ามกับ Equality constraints คือ แสดงความสัมพันธ์ที่ระบุว่าหากมีความสัมพันธ์แบบหนึ่งเกิดขึ้นจะต้องไม่มีความสัมพันธ์อีกแบบหนึ่งเกิดขึ้นโดยเด็ดขาด ซึ่งสามารถแสดงในแผนภาพได้ดังนี้



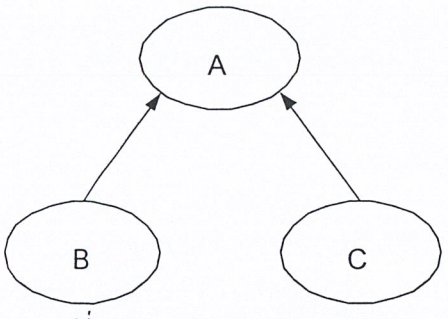
รูปที่ 4 - 26 แสดง Exclusion constraints

ลักษณะดังกล่าวนี้ แสดงให้เห็นกฎข้อบังคับความถูกต้องว่า หากมีการบันทึกข้อมูลความสัมพันธ์ r1 ของสมาชิกของชนิดเอนติตี A ใด จะต้องไม่มีการบันทึกข้อมูลความสัมพันธ์ r2 ของสมาชิกของชนิดเอนติตี A นั้นโดยเด็ดขาด เช่น ถ้าบุคคลใดถูกเลือกให้เป็นกรรมการในการตัดสินเกมนั้น บุคคลนั้นจะไม่มีสิทธ์เป็นผู้แข่งขันในเกมอย่างเด็ดขาด

4.2.2.9 Subtype constraints

เป็นกฎข้อบังคับความถูกต้องที่ระบุถึงการแบ่งกลุ่มของสมาชิกของชนิดเอนติตีที่มีอยู่อย่างชัดเจน ซึ่งสมาชิกของชนิดเอนติตีที่แบ่งแยกออกจากชนิดเอนติตีที่เป็น Super Type นั้น จะต้องมีลักษณะและคุณสมบัติที่แตกต่างกันอย่างชัดเจน ดังสามารถแสดงในแผนภาพได้ดังนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

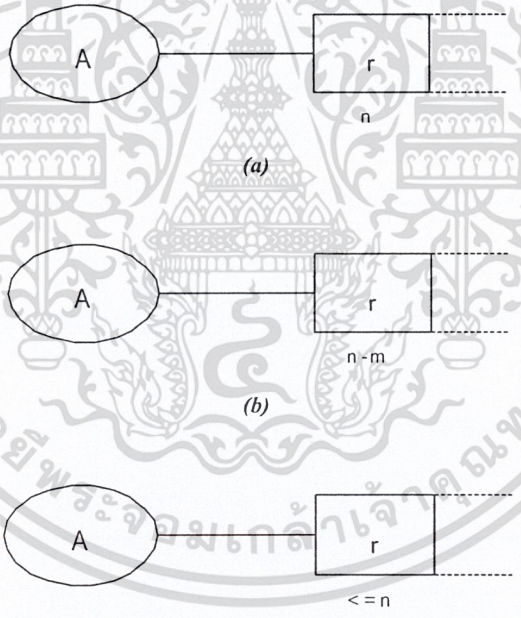


รูปที่ 4 - 27 แสดง Subtype constraints

ลักษณะดังกล่าวนี้แสดงให้เห็นว่า สมาชิกของชนิดเอนติตี A โดยจะเรียกว่า Super Type นั้นสามารถแบ่งออกเป็น 2 กลุ่มได้คือ กลุ่มของชนิดเอนติตี B และกลุ่มของชนิดเอนติตี C ซึ่งเรียกว่า Subtype เช่น ชนิดเอนติตีของบุคคล สามารถแบ่งออกเป็น Subtype ผู้ชายและผู้หญิงได้

4.2.2.10 Occurrence frequency constraints

เป็นกฎข้อบังคับความถูกต้องของข้อมูลที่ใช้ในการระบุจำนวนครั้งที่สมาชิกของชนิดเอนติตีใดๆ จะสามารถแสดงบทบาทใดบทบาทหนึ่งได้ ซึ่งสามารถแสดงในแผนภาพได้ดังนี้



รูปที่ 4 - 28 แสดง Occurrence frequency constraints

จากรูปที่ 4 - 28 (a) เป็นการแสดงกฎข้อบังคับความถูกต้องของข้อมูลโดยที่แต่ละชนิดเอนติตี A จะมีการแสดงบทบาทในคอลัมน์ r เป็นจำนวน n ครั้ง จากรูปที่ 4 - 28 (b) เป็นการแสดงกฎข้อบังคับความถูกต้องของข้อมูลโดยที่แต่ละ ชนิดเอนติตี A ในการแสดงบทบาทในคอลัมน์ r ได้อย่างน้อยที่สุด n ครั้งและมากที่สุด m ครั้ง และจากรูปที่ 4 - 28 (c) เป็นการแสดงกฎข้อบังคับความถูกต้องของข้อมูลโดยที่แต่ละ ชนิดเอนติตี A ในการแสดงบทบาทในคอลัมน์ r ได้อย่างน้อยที่สุด n ครั้ง เช่น ชมรมใดชมรมหนึ่งจะต้องมีสมาชิกอย่างน้อย 20 คน แต่จำนวนสูงสุดที่รับได้ต้องไม่เกิน 200 คน เป็นต้น

4.2.3 The Optimal Normal Form algorithm (ONF อัลกอริธึม)

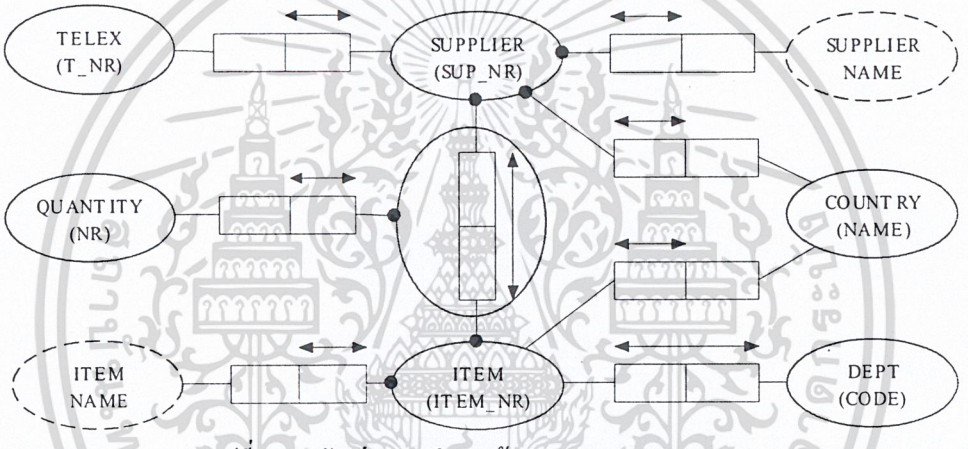
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เป็นวิธีการจัดกลุ่มความจริงใน Conceptual Schema ให้เป็น Relational Database Schema โดยมีหลักการโดยสรุปดังต่อไปนี้

1. สร้าง 1 รีเลชัน สำหรับชนิดความจริงแบบไบนารีซึ่งมีความสัมพันธ์แบบ many to many โดยที่ Unique Identifier ของชนิดเอนติตีที่เกี่ยวข้องทั้งหมดเป็น Primary Key

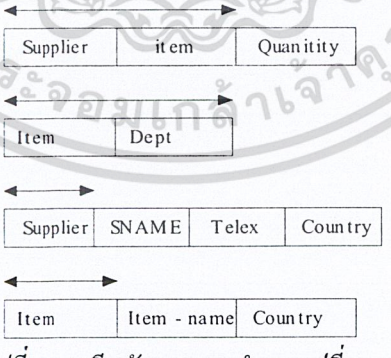
2. สร้าง 1 รีเลชัน สำหรับแต่ละชนิดความจริงแบบ n-ary โดยที่ Unique Identifier ของชนิดเอนติตีซึ่งมี role ถูกบังคับด้วย Uniqueness Constraint เดียวกันเป็น Candidate Key

3. พิจารณาชนิดเอนติตีที่เกี่ยวข้องกับความจริงแบบไบนารีที่มีความสัมพันธ์เป็นแบบ one to one หรือ many to one โดยที่ role ของชนิดเอนติตีเหล่านั้นถูกบังคับด้วย Uniqueness Constraint ให้สร้างรีเลชันโดยมี Unique Identifier ของชนิดเอนติตีเหล่านั้นเป็น Candidate Key



รูปที่ 4 - 29 ตัวอย่างแบบจำลองข้อมูล(Conceptual Schema)

จาก Conceptual Schema ในรูปที่ 4 - 29 เมื่อใช้ ONF อัลกอริธึม จะได้รีเลชันดังต่อไปนี้



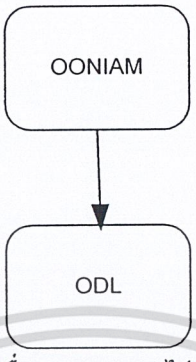
รูปที่ 4 – 30 รีเลชันของแบบจำลองรูปที่ 4 - 29

โดยมีเครื่องหมาย ◄—► แอตทริบิวต์หรือกลุ่มของแอตทริบิวต์ที่เป็น Primary key และเครื่องหมาย ◄----► บน candidate key

4.3 แบบจำลองแนวคิด OONIAM และการแปลงรูป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การสร้างแบบจำลองแนวคิด OONIAM ให้สามารถทำงานร่วมกับฐานข้อมูลเชิงวัตถุได้นั้นมีขั้นตอนดังรูปที่ 4 - 31



รูปที่ 4 - 31 ขั้นตอนการทำงานเพื่อนำ OONIAM ไปทำงานร่วมกับฐานข้อมูลเชิงวัตถุ

จากรูปที่ 4 - 31 OONIAM เป็นแบบจำลองสำหรับการออกแบบฐานข้อมูลเชิงวัตถุ เป็นแบบจำลองที่นำ NIAM เดิมมาเพิ่มขยายความสามารถให้รองรับแนวคิดเชิงวัตถุได้ สำหรับ ODL เป็นภาษานิยามฐานข้อมูลเชิงวัตถุ

ในบทนี้กล่าวถึง OONIAM และการแปลงรูปเป็นภาษานิยามเชิงวัตถุ (ODL) ซึ่งจะแบ่งส่วนในการอธิบายดังนี้ หัวข้อถัดไป เป็นการนำเสนอแบบจำลอง OONIAM เพื่อทำให้ NIAM ที่มีอยู่เดิมสามารถรองรับแนวคิดเชิงวัตถุได้ ซึ่งในหัวข้อนี้จะได้กล่าวถึง โครงร่างหลัก (Main Schema) และโครงร่างย่อย (Sub Schema) ที่เป็นส่วนสำคัญในการสร้าง OONIAM โดยโครงร่างทั้งสองจะนำเสนอความสัมพันธ์ต่างๆ ของแอตทริบิวต์ และความสัมพันธ์ของออบเจ็กต์ ในหัวข้อถัดไป จะกล่าวถึงการแปลง OONIAM ให้เป็น ODL โดยจะนำโครงร่างหลักและโครงร่างย่อยมาผ่านการประมวลผลเพื่อแปลงโครงร่างทั้งสองให้เป็นภาษา ODL

4.3.1 แบบจำลองแนวคิด OONIAM

เมื่อผู้ใช้งานต้องการที่จะออกแบบฐานข้อมูลบนระบบฐานข้อมูลเชิงวัตถุ ผู้ใช้งานต้องอาศัยเครื่องมือในการออกแบบฐานข้อมูล ซึ่งในปัจจุบันเครื่องมือที่เป็นที่นิยมนำมาช่วยในการออกแบบฐานข้อมูลมีด้วยกัน 2 เครื่องมือคือ ER และ NIAM ซึ่งเครื่องมือทั้งสองก็มีข้อดีข้อเสียที่แตกต่างกันออกไป การที่ผู้ใช้งานจะใช้เครื่องมือใดนั้นก็แล้วแต่ความถนัด คุ้นเคยและประสบการณ์ของแต่ละคน แต่มีข้อสังเกตอยู่ประการหนึ่งก็คือ NIAM นั้นสามารถออกแบบฐานข้อมูลได้ถึง 5NF เลย แต่ ER ไม่สามารถทำได้เพราะ ER ไม่มีการระบุถึงความสัมพันธ์ระหว่างแอตทริบิวต์ จึงทำให้ ER สามารถออกแบบฐานข้อมูลได้แค่ 3NF เท่านั้น ซึ่งในการออกแบบฐานข้อมูลที่ใช้งานทั่วไปนั้น ส่วนมากแล้วการออกแบบแค่ 3NF ก็สามารถทำงานได้แล้ว

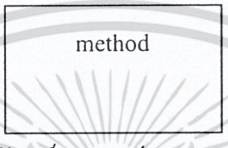
การใช้งาน ER และ NIAM นั้นจะใช้กับระบบฐานข้อมูลเชิงสัมพันธ์เท่านั้น แต่ในปัจจุบันระบบฐานข้อมูลมิได้มีเพียงระบบฐานข้อมูลเชิงสัมพันธ์เท่านั้น แต่มีระบบฐานข้อมูลเชิงวัตถุสัมพันธ์และระบบฐานข้อมูลเชิงวัตถุอีกด้วย ซึ่งระบบฐานข้อมูลทั้งสองได้รับการพัฒนาให้มีขีดความสามารถที่สูงขึ้น โดยมีแนวคิดให้สามารถรองรับข้อมูลและกระบวนการของข้อมูลที่สลับซับซ้อนมากขึ้นได้ เมื่อมีระบบ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ฐานข้อมูลแบบใหม่ขึ้นมาแล้ว ก็ต้องมีการเครื่องมือในการออกแบบใหม่ด้วย ซึ่ง ER และ NIAM ที่มีแต่เดิมไม่สามารถใช้งานกับระบบฐานข้อมูลใหม่ทั้งสองนี้

ในหัวข้อนี้จะกล่าวถึงการนำเอา NIAM ที่มีแต่เดิมมาเพิ่มขยายความสามารถให้รองรับหลักการเชิงวัตถุได้ ซึ่งเครื่องมือใหม่นี้เรียกว่า แบบจำลองไนแอมเชิงวัตถุ (Object - Oriented NIAM Schema Model) หรือ เรียกย่อๆ ว่า OONIAM เมื่อผู้ใช้งานสร้างแบบจำลองของฐานข้อมูลเชิงวัตถุโดยใช้ OONIAM แล้ว ผู้ใช้งานสามารถแปลงแบบจำลองที่สร้างขึ้นให้เป็นภาษานิยามเชิงวัตถุได้เลย

สำหรับหลักการสร้าง OONIAM นั้น จะนำสัญลักษณ์เดิมที่มีอยู่ใน NIAM มาใช้ร่วมกับสัญลักษณ์ใหม่ที่เพิ่มเติมเข้าไปดังรูปที่ 4 - 32 เพื่อให้ NIAM ใหม่นี้รองรับหลักการเชิงวัตถุได้

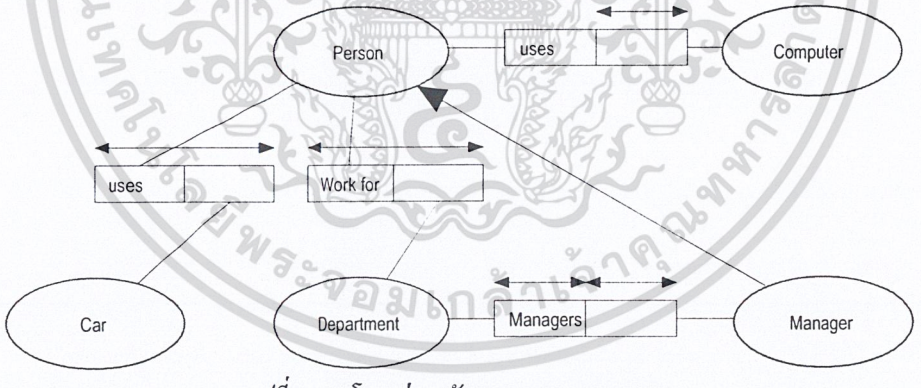


รูปที่ 4 - 32 สัญลักษณ์ของการประมวลผลข้อมูล (เมธอด)

สำหรับสัญลักษณ์ที่เพิ่มเข้าไปในรูปที่ 4-32 นั้นคือ กระบวนการประมวลผลนี้จะมีไว้ให้ผู้ใช้งานที่เพิ่มเข้าไปที่เอนติตีที่ต้องการ เพื่อให้เมธอดที่เพิ่มเข้าไปนี้จัดการกับข้อมูลที่ต้องการได้โดยรายละเอียดจะกล่าวถึงในหัวข้อ โครงร่างย่อย OONIAM แบ่งส่วนสำคัญออกเป็นสองส่วนดังนี้

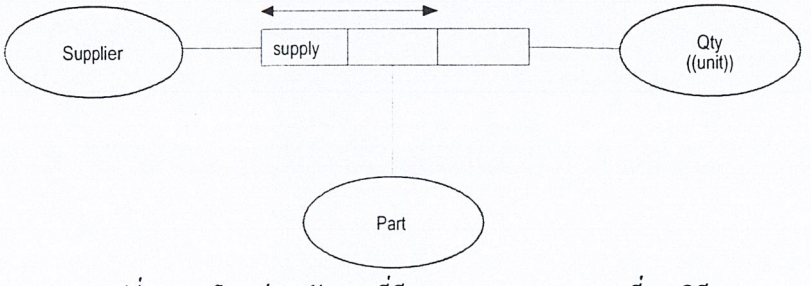
4.3.1.1 โครงร่างหลัก (Main Schema)

โครงร่างหลักนี้เป็นภาพรวมของความสัมพันธ์ระหว่างคลาส ดังนั้นในแต่ละคลาสจะไม่มีการแสดงแอตทริบิวต์ และไม่มี Uniqueness Identifier ดังตัวอย่างในรูปที่ 4 - 33



รูปที่ 4 - 33 โครงร่างหลักของคลาสหลายคลาส

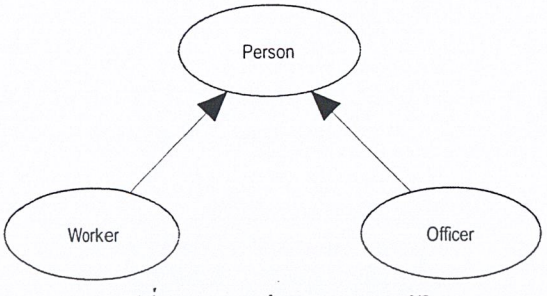
หากมีความสัมพันธ์กันแบบ N-ary ก็จะสามารถมี Uniquness Identifier ได้ที่เอนติตี ดังรูปที่ 4 - 34



รูปที่ 4 - 34 โครงร่างหลักแบบที่มี Uniqueness Identifier ที่เอนติตี

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

นอกจากนี้ การอธิบายถึงความสัมพันธ์ระหว่างคลาสแล้ว ในโครงร่างหลักยังได้กล่าวถึง การถ่ายทอดคุณสมบัติของคลาสอีกด้วย ดังรูปที่ 4 - 35

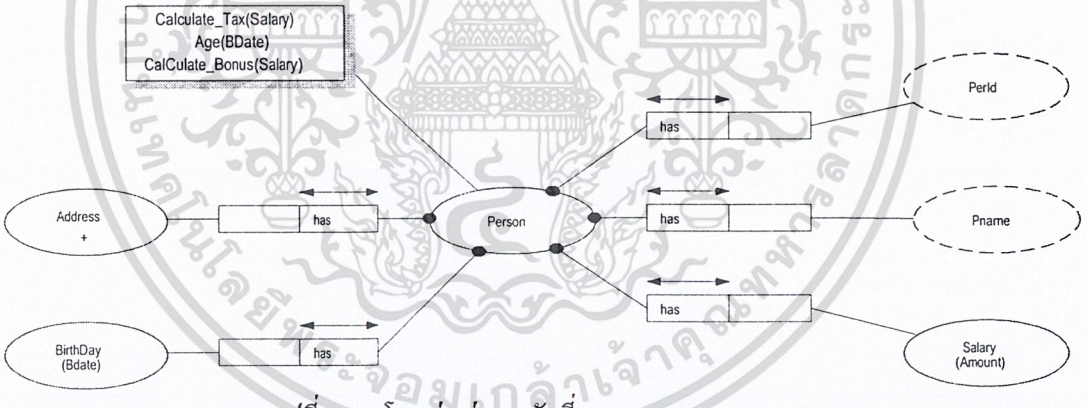


รูปที่ 4 - 35 การถ่ายทอดคุณสมบัติ

สำหรับโครงร่างหลักนี้จะแสดงความสัมพันธ์ระหว่างคลาสแบบหนึ่งต่อหนึ่ง หนึ่งต่อหลาย หลายต่อหนึ่ง หรือ หลายต่อหลายก็ได้ แล้วแต่การออกแบบคลาสนั้นๆ ซึ่งแอตทริบิวต์ของแต่ละคลาสและ Uniqueness Identifier นั้นจะแสดงในโครงร่างย่อย

4.3.1.2 โครงร่างย่อย

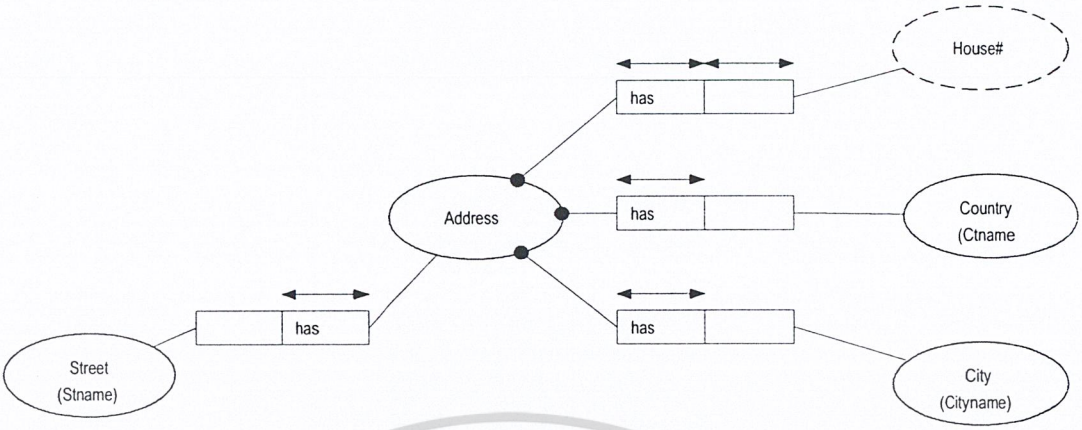
โครงร่างย่อยเป็นการแสดงรายละเอียดของคลาส ซึ่งแต่ละคลาสนั้นอาจเป็นคลาสที่ซับซ้อนก็ได้ กล่าวคือ คลาสดังกล่าวอาจมีคลาสอื่นเป็นส่วนประกอบก็ได้ โดยโครงร่างย่อยนี้ประกอบด้วย แอตทริบิวต์, เมธอด และคลาสที่อยู่ในคลาส ดังรูปที่ 4 - 36



รูปที่ 4 - 36 โครงร่างย่อยระดับที่ 1 ของคลาส Person

จากรูปที่ 4 - 36 เป็นการเขียนเพิ่มขยายโครงร่างหลักในรูปที่ 4 - 35 กล่าวคือในรูปที่ 4 - 35 ได้บอกถึงความสัมพันธ์ของคลาส Person ที่มีต่อคลาสอื่น ๆ แต่ยังไม่ได้อธิบายถึงรายละเอียดของคลาส Person เมื่อต้องการอธิบายรายละเอียดต่าง ๆ ของคลาสดังกล่าวจำเป็นต้องอธิบายในโครงร่างย่อยเท่านั้น จากโครงร่างย่อยในรูปที่ 4 - 36 ซึ่งเป็นโครงร่างย่อยระดับที่ 1 จะเห็นว่าคลาส Person จะมีแอตทริบิวต์ต่างๆ ดังแสดงในรูป และมีเมธอดสำหรับจัดการกับคลาสนี้ด้วยกัน 3 เมธอด นอกจากนี้แล้วสัญลักษณ์ที่แสดงชื่อคลาสหลักนี้จะไม่มี Uniqueness Identifier ที่สำคัญคลาสนี้ยังมีคลาส Address เป็นคลาสย่อย (Embedded Class) อีก โดยคลาสย่อยนี้แสดงโดยการใช้เครื่องหมายบวก (+) ไว้ใต้ชื่อคลาส ซึ่งหากต้องการเขียนโครงร่างย่อยของคลาส Address และคลาส Land สามารถเขียนได้ดังรูปที่ 4 - 37 ซึ่งเป็นโครงร่างย่อยระดับที่ 2

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



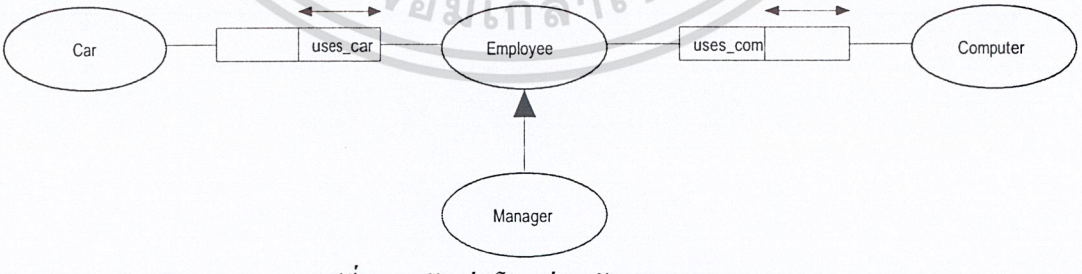
รูปที่ 4 - 37 โครงร่างย่อยของคลาส Address

จากรูปที่ 4 - 37 และรูปที่ 4 - 36 จะเห็นได้ว่าโครงร่างย่อยของคลาส Person มีด้วยกัน 2 ระดับคือ ระดับแรก เป็นการอธิบายคลาส Person (รูปที่ 4 - 36) ส่วนระดับที่ 2 เป็นการอธิบายคลาสที่เกี่ยวเนื่องกับคลาส Person (รูปที่ 4 - 37) ซึ่งจะสังเกตได้ว่าในโครงร่างย่อยแต่ละระดับนั้นหากต้องการอธิบายรายละเอียดของคลาสใด คลาสนั้นจะไม่มีการแสดง Uniqueness Identifier ในแบบจำลอง ส่วนคลาสที่เกี่ยวเนื่องนั้นจะแสดงในโครงร่างย่อยในระดับถัด ๆ ไป

สำหรับโครงร่างย่อยทั่ว ๆ ไปนั้นอาจมีหลายระดับ ขึ้นอยู่กับความซับซ้อนของข้อมูลและการออกแบบของผู้ออกแบบ

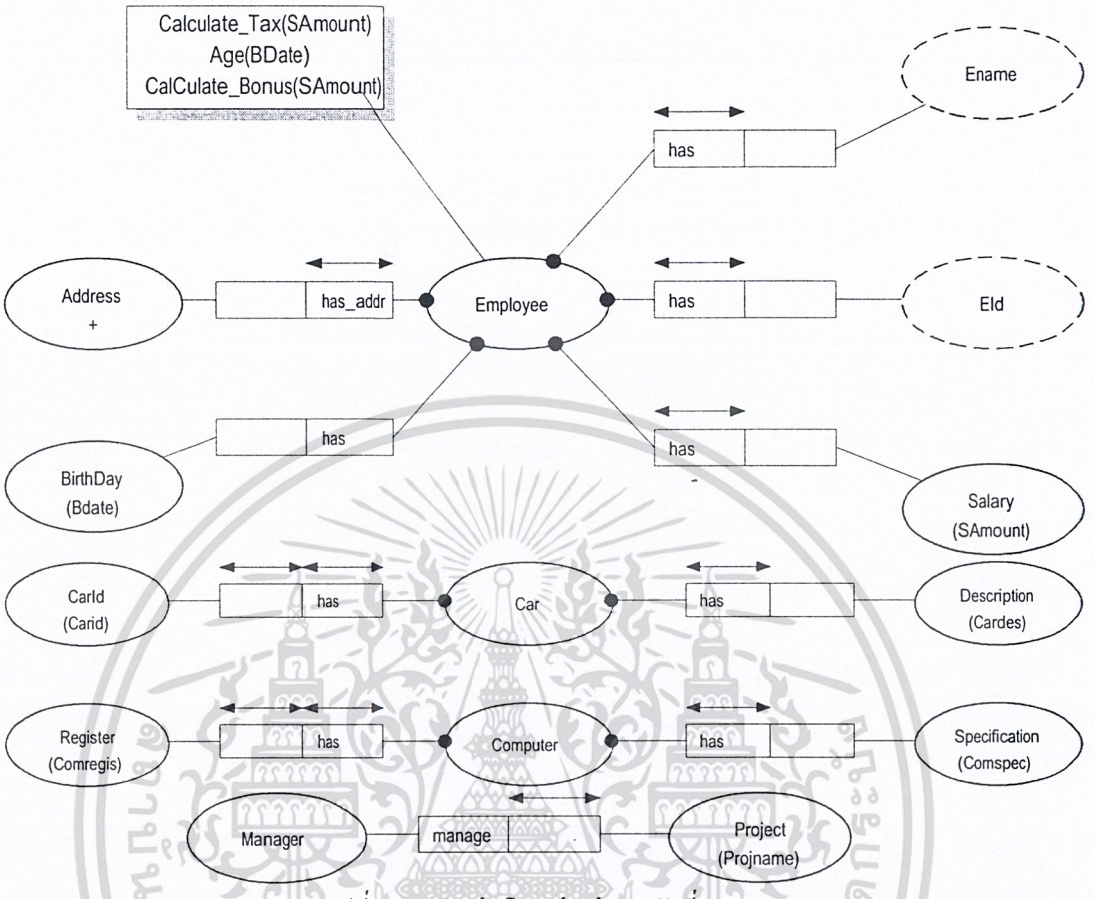
4.3.2 การแปลงแบบจำลอง OONIAM ให้เป็นภาษานิยามเชิงวัตถุ ODL

จะขอกล่าวถึงการแปลงแบบจำลอง OONIAM ซึ่งเป็นแบบจำลอง NIAM เชิงวัตถุให้เป็นภาษานิยามเชิงวัตถุ ODL สำหรับขั้นตอนการแปลงนั้นเริ่มต้นต้องแปลงจากโครงร่างย่อยก่อน จากนั้นจึงไปแปลงในโครงร่างหลัก เนื่องจากในโครงร่างย่อยได้อธิบายโครงสร้างของคลาสไว้อย่างละเอียด ดังนั้นเราจำเป็นต้องสร้างคลาสต่างๆ ก่อน จึงจะนำโครงร่างหลักมาใช้งานร่วมเพื่ออธิบายถึงความสัมพันธ์ระหว่างคลาส ซึ่งการแปลง OONIAM ให้เป็น ODL ดังที่กล่าวมาแล้วนั้นแสดงตัวอย่างในรูปที่ 4 - 38, รูปที่ 4 - 39 และรูปที่ 4 - 40

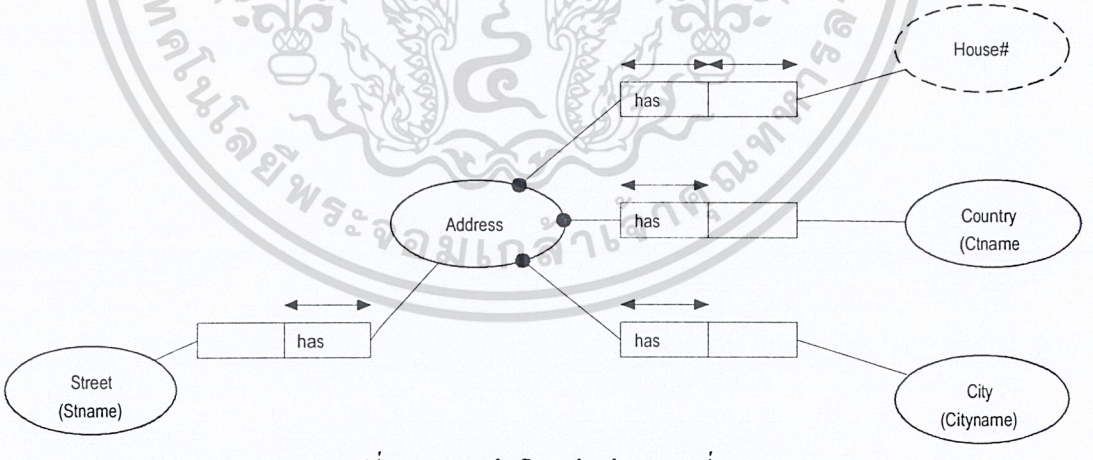


รูปที่ 4 - 38 ตัวอย่างโครงร่างหลักของระบบ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4 - 39 ตัวอย่างโครงร่างย่อยระดับที่ 1



รูปที่ 4 - 40 ตัวอย่างโครงร่างย่อยระดับที่ 2

จากรูปที่ 4 - 38, รูปที่ 4 - 39 และ รูปที่ 4 - 40 สามารถแปลงเป็นภาษานิยามเชิงวัตถุได้ดังนี้

```
interface Employee
(     extent Employees)
{
      typedef Struct AddressStruct {
                  sring Houseid,
                  sring Ctname,
                  string Cityname,
                  string Stnae;}
      attribute string Eid;
      atribute string Ename;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        attribute float Salary;
        attribute date Bdate;
        attribute set<AddressStruct> Address;
        relationship Car uses_car inverse Car::is_used_by;
        relationship set<Computer> uses_com inverse
Computer::is_used_by;
        float Calculate_Tax(in float Samount);
        float Age(in float Bdate);
        float Calculate_Bonus(in float Samount)
};

interface Manager:Employee
(       extent Managers)
{
        attribute string ProName;
};

interface Computer
(       extent Computers)
{
        attribute string Comregis;
        attribute string Comspec;
        relationship Employee is_used_by inverse Employee::uses_com;
};

interface Car
(       extent Cars)
{
        attribute string Carid;
        attribute string Cardes;
        relationship set<Employee> is_used_by inverse
Employee::uses_car;
};
```

สำหรับหลักการในการแปลง OONIAM ให้เป็นภาษา ODL มีหลักการดังนี้

4.3.2.1 การแปลงโครงร่างย่อย

โครงร่างย่อยเป็นการอธิบายรายละเอียดของแต่ละคลาส ซึ่งจะอธิบายแอตทริบิวต์ คลาสย่อยและคลาสอ้างอิง โดยจะต้องแยกเป็นกรณีๆ ไป ซึ่งคลาสทั้งหมดได้ถูกสร้างขึ้นแล้วในขั้นตอนการแปลงโครงร่างหลัก ดังนั้นในโครงร่างย่อยนี้เป็นส่วนของการเพิ่มรายละเอียดของแอตทริบิวต์ สร้างคลาสย่อยและสร้างคลาสอ้างอิงเท่านั้น สำหรับอัลกอริธึมในการแปลงโครงร่างย่อยมีดังนี้

นำโครงร่างย่อยมาสร้างเป็นคลาสใหม่จัดเก็บลงใน Meta Class โดยสร้างแอตทริบิวต์ของคลาสโดยใช้ Uniqueness Identifier เป็นชื่อแอตทริบิวต์ โดยสามารถแบ่งลักษณะของชนิดความจริงที่คลาสหลักมีกับแอตทริบิวต์ ได้ดังนี้

ชนิดความจริงแบบหนึ่งต่อหนึ่ง แอตทริบิวต์นั้นเป็น Simple

ชนิดความจริงแบบหนึ่งต่อหลาย แอตทริบิวต์นั้นเป็น Collection

ชนิดความจริงแบบหลายต่อหนึ่ง แอตทริบิวต์นั้นเป็น Simple

ชนิดความจริงแบบหลายต่อหลาย แอตทริบิวต์นั้นเป็น Collection

หากคลาสใดมีคลาสย่อย นำคลาสที่สร้างไว้ใน Meta Class มาสร้างเป็นคลาสย่อยของคลาสนั้น โดยสามารถแบ่งลักษณะของชนิดความจริงที่คลาสหลักมีกับคลาสย่อยนั้นได้ดังนี้

ชนิดความจริงแบบหนึ่งต่อหนึ่ง คลาสย่อยนั้นเป็น Simple

ชนิดความจริงแบบหนึ่งต่อหลาย คลาสย่อยนั้นเป็น Collection

ชนิดความจริงแบบหลายต่อหนึ่ง คลาสย่อยนั้นเป็น Simple

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ชนิดความจริงแบบหลายต่อหลาย คลาสย่อยนั้นเป็น Collection

4.3.2.2 การแปลงโครงร่างหลัก

โครงร่างหลักนี้ได้อธิบายถึงความสัมพันธ์ในแต่ละคลาส รวมถึงการถ่ายทอดคุณสมบัติของคลาสอีกด้วย นอกจากนี้ยังอธิบายถึงความสัมพันธ์แบบ N-ary ด้วย สำหรับอัลกอริธึมในการแปลงโครงร่างหลักมีดังนี้

กำหนดคุณสมบัติให้กับคลาสที่มีการถ่ายทอดคุณสมบัติ

กำหนดความสัมพันธ์ระหว่างคลาส ดังนี้

ชนิดความจริงแบบหนึ่งต่อหนึ่ง กำหนดให้ทั้งสองคลาสมีความสัมพันธ์กันแบบ Reference ทั้งสองคลาส

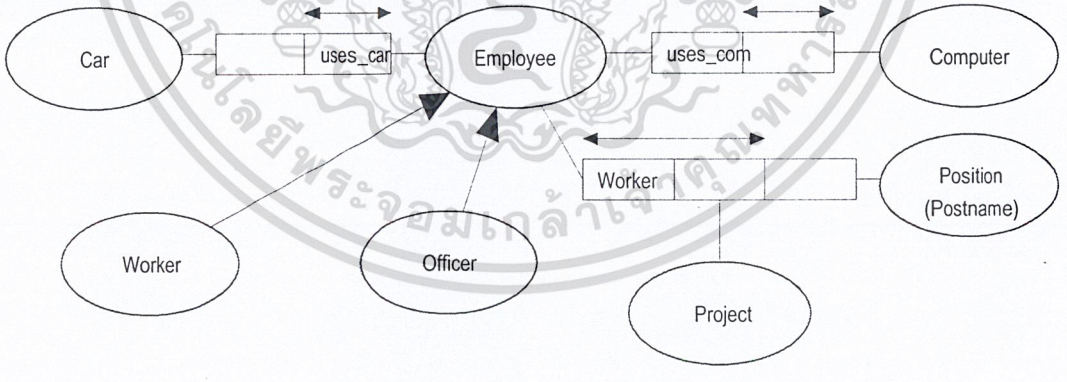
ชนิดความจริงแบบหนึ่งต่อหลาย กำหนดให้ทั้งสองคลาสมีความสัมพันธ์กันแบบ Collection ของ Reference และ Reference ตามลำดับ

ชนิดความจริงแบบหลายต่อหนึ่ง กำหนดให้ทั้งสองคลาสมีความสัมพันธ์กันแบบ Reference และ Collection ของ Reference ตามลำดับ

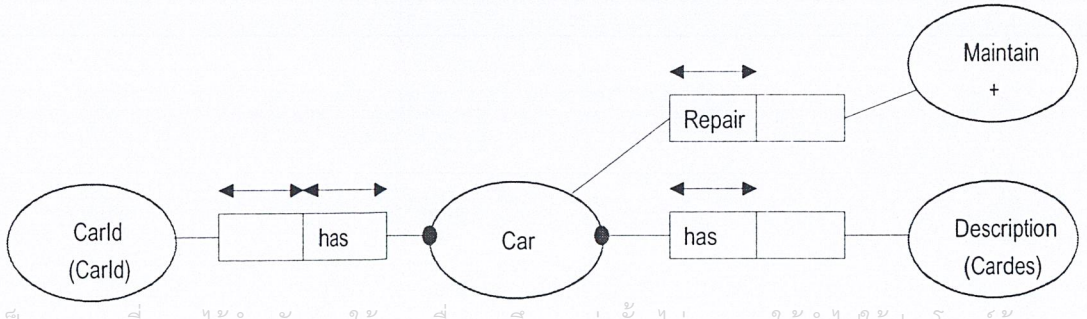
ชนิดความจริงแบบหลายต่อหลาย กำหนดให้ทั้งสองคลาสมีความสัมพันธ์กันแบบ Collection ของ Reference และ Collection ของ Reference ตามลำดับ

หากมีความสัมพันธ์แบบหลายบทบาท ให้สร้างคลาสใหม่เพื่อจัดเก็บ Simple Object ที่เกี่ยวข้องกับ N-ary โดยสร้างความสัมพันธ์หนึ่งต่อหนึ่งไปยังคลาส Complex และกำหนดให้คลาส Complex มีความสัมพันธ์แบบหนึ่งต่อหลายไปยังคลาสที่สร้างใหม่

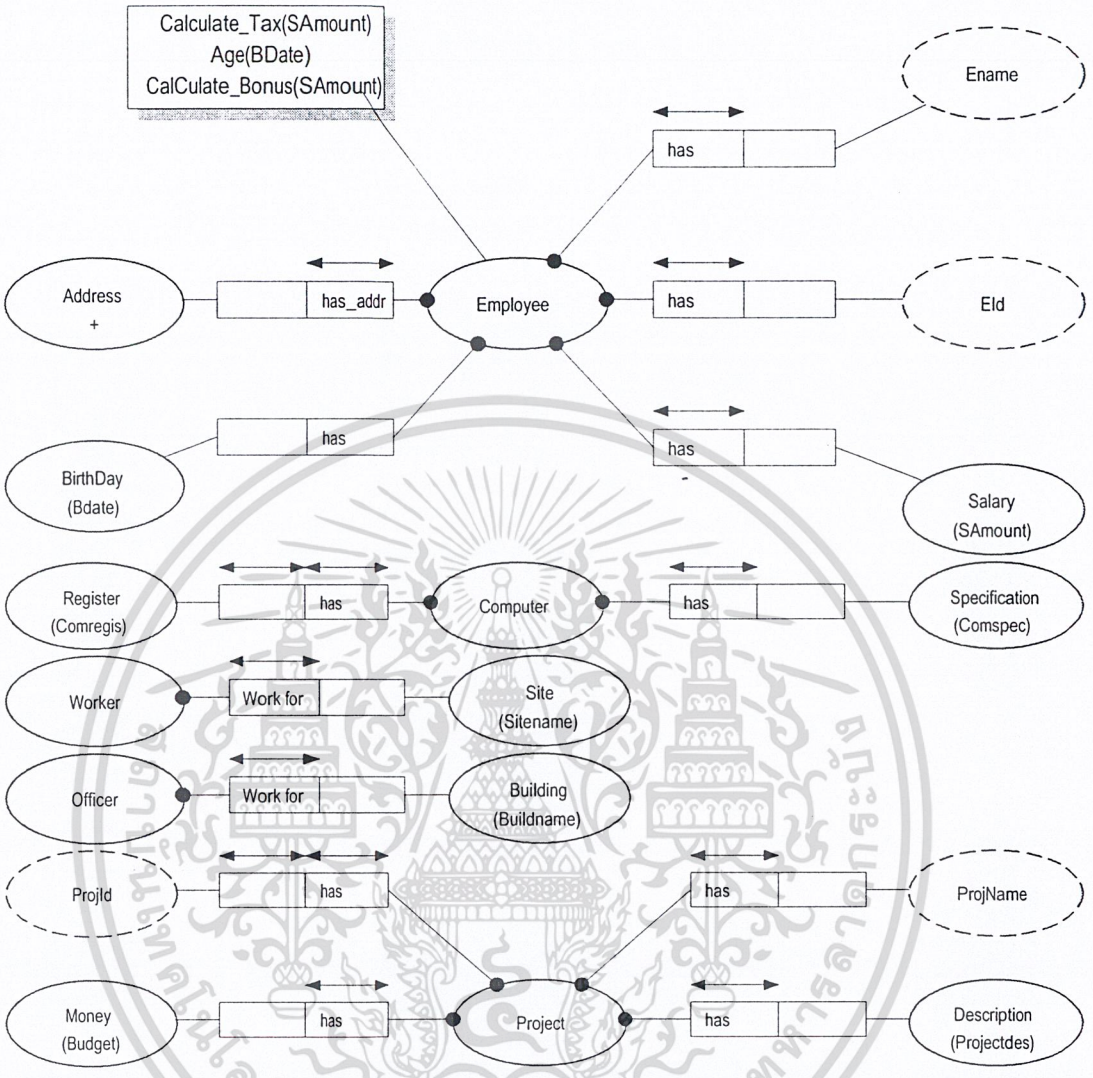
4.3.3 ตัวอย่างของ OONIAM และการแปลงให้เป็นภาษา ODL



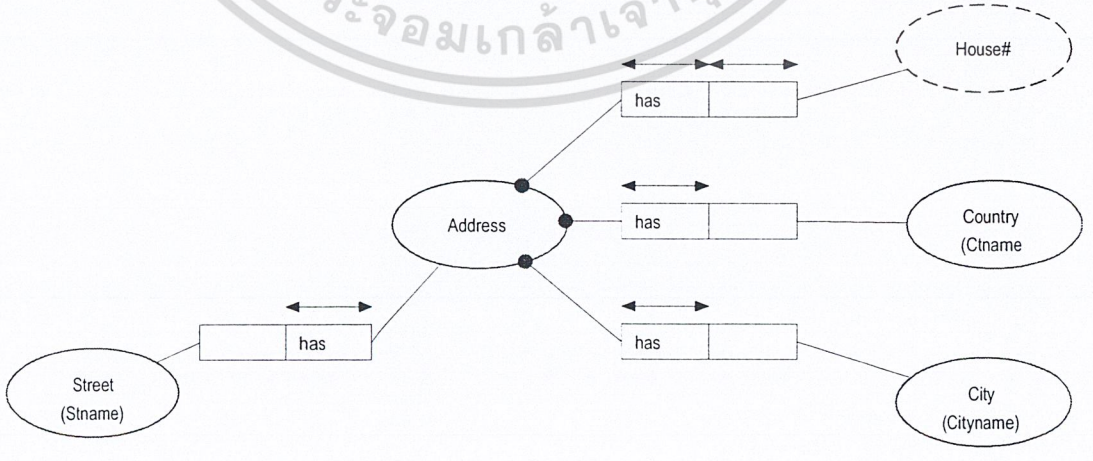
รูปที่ 4 - 41 ตัวอย่างโครงร่างหลักของระบบ



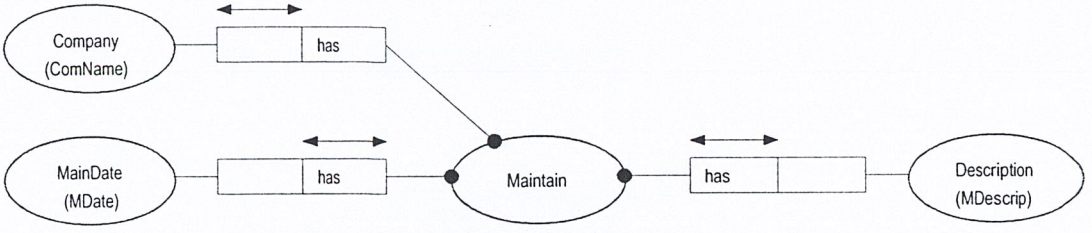
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4 - 42 ตัวอย่างโครงร่างย่อยระดับที่ 1



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4 - 43 ตัวอย่างโครงร่างย่อยระดับที่ 2

4.3.4 แปลงตัวอย่าง OONIAM ให้เป็นภาษา ODL

จากตัวอย่างด้านบนสามารถแปลง OONIAM ให้เป็นภาษา ODL ได้ดังนี้

```
interface Employee
(extent Employees)
{
      typedef Struct AddressStruct {
                  string Houseid,
                  string Ctname,
                  string Cityname,
                  string Stname;}

      attribute string Eid;
      attribute string Ename;
      attribute float Salary;
      attribute date Bdate;

      attribute set<AddressStruct> Address;

      relationship Car uses_car inverse Car::is_used_by;
      relationship set<Computer> uses_com inverse
Computer::is_used_by;
      relationship set<Position> work_for inverse
Position::Employees;

      float Calculate_Tax(in float Samount);
      float Age(in float Bdate);
      float Calculate_Bonus(in float Samount)
};

interface Worker:Employee
(     extent Workers)
{
      attribute string SiteName;
};

interface Officer:Employee
(     extent Officers)
{
      attribute string Buildname;
};

interface Computer
(     extent Computers)
{
      attribute string Comregis;
      attribute string Comspec;

      relationship Employee is_used_by inverse Employee::uses_com;
};

interface Car
(     extent Cars)
{
      typedet struct MaintainStruct{
            date Mdate,
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

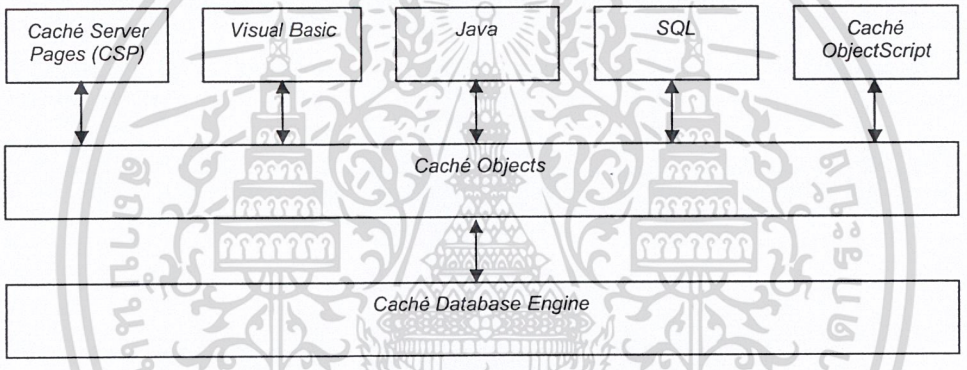
```
            string Mdescrip,
            string ComName;
            };
      attribute string Carid;
      attribute string Cardes;
      attribute maintainStruct Maintain;

      relationship set<Employee> is_used_by inverse
Employee::uses_car;
};

interface Project
(     extent Projects)
{
      attribute string ProjId;
      attribute string ProjName;
      attribute string Projectdes;
      attribute float Budget;

      relationship set <Position> Employees inverse
Position::Projects;
};

interface Position
(     extent Positions)
{
      relationship Employee Employees inverse Employee::Work_for;
      relationship Project Projects inverse Project::Employees;
      Attribute string Posname;

};
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 5
ระบบจัดการฐานข้อมูลเชิงวัตถุ คาเช่ (Caché)

5.1 บทนำ

ระบบจัดการฐานข้อมูลเชิงวัตถุที่ใช้ ได้แก่ Caché (อ่านว่า คา-เช่) พัฒนาโดย InterSystems Corporation ซึ่งเป็นระบบจัดการฐานข้อมูลที่สนับสนุนเทคโนโลยีเชิงวัตถุ (Object – Oriented Technology) สามารถจำลองโครงสร้างของข้อมูลที่มีความซับซ้อนให้คงสภาพตามที่เป็นจริงโดยไม่ต้องเปลี่ยนรูปร่างของโครงสร้างข้อมูล ทำให้ลดความสับสนในการกำหนดความสัมพันธ์ของข้อมูลไปได้มาก สำหรับการใช้งาน Caché สามารถทำงานได้บนหลากหลายแพลตฟอร์ม (Hardware Platform) และหลากหลายระบบปฏิบัติการ (Operating System) อาทิเช่น Window 95/98/NT, Unix และ Linux เป็นต้น



รูปที่ 5 - 1 การติดต่อกับระบบจัดการฐานข้อมูล Caché

5.2 คุณสมบัติด้านต่างๆ ของ Caché

1. Caché ใช้ ODBCs ที่เรียกว่า Ultra Fast ODBC เพื่อให้สามารถใช้ภาษา SQL เป็นตัวติดต่อกับระบบฐานข้อมูลได้ และใช้ Visual Caché เพื่อให้ออบเจ็กต์ของ Caché สามารถใช้ได้กับ Visual Basic และเครื่องมือในการพัฒนา GUI อื่นๆ นอกจากนั้นยังมี Weblink ซึ่งทำให้สามารถเชื่อมต่อฐานข้อมูล Caché ผ่านทางเครือข่ายอินเทอร์เน็ตได้ด้วย
2. Caché Object Script เป็นส่วนที่ใช้ในการพัฒนาโปรแกรมให้เป็นแบบออบเจ็กต์ได้ เพื่อเพิ่มความยืดหยุ่นในการใช้งาน และยังสามารถใช้ JAVA หรือ C++ ในการแก้ไขโปรแกรมได้อีกด้วย
3. วิธีการเข้าถึงข้อมูล สามารถเข้าถึงข้อมูลได้ 3 แบบ ดังตารางที่ 5 – 1
4. Transaction Database Engine ซึ่งเป็นตัวจัดการการทำงานของแอพพลิเคชั่น
5. Distributed Caché Protocol ช่วยในการทำงานทางด้านการติดต่อของระบบเครือข่าย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เครื่องมือ	รายละเอียด
Caché Multidimensional Access	เป็นการเข้าถึงข้อมูลแบบ Global variable ซึ่งจัดเก็บแบบ Multidimentional array
Caché SQL	เป็นการเข้าถึงข้อมูลโดยผ่านทางภาษา SQL
Caché Objects	เป็นการเข้าถึงข้อมูลด้วย JAVA, C++ และเทคโนโลยีในการพัฒนาทางด้านออบเจ็กต์อื่น ๆ

ตารางที่ 5 - 1 แสดงการเข้าถึงข้อมูลของ Caché

5.3 องค์ประกอบของ Caché

เครื่องมือ	รายละเอียด
Caché Object Server	ทำให้สามารถใช้งานออบเจ็กต์ของ Caché ได้โดยตรงผ่านทาง JAVA หรือ C++
Caché SQL Server	สามารถเข้าถึงข้อมูลผ่านทาง SQL และ ODBC
Caché Studio	ใช้ในการพัฒนา GUI ในการกำหนดคลาสของออบเจ็กต์และสร้างส่วนประกอบของแอพพลิเคชัน
Caché Distributed Caché Protocol (DCP)	สามารถกระจายฐานข้อมูลทั่วระบบเครือข่ายในการตอบรับต่อความต้องการ เพื่อให้ประสิทธิภาพดีที่สุด
Caché Object Script	เป็นภาษาในการเขียนโปรแกรมที่มีมาสำหรับแอพพลิเคชันด้านการประมวลผล ทรานแซคชั่นและควบคุมออบเจ็กต์ในระบบ
Caché Weblink	รองรับการเชื่อมต่อระหว่าง Caché กับเว็บเซิร์ฟเวอร์ สำหรับประมวลผลทรานแซคชั่นผ่านทางเว็บ
Visual Caché	เป็นเครื่องมือสร้างฟอร์มวิชวลเบสิกของไมโครซอฟท์อย่างอัตโนมัติ และยังให้การเชื่อมต่อระหว่างออบเจ็กต์ของ Caché กับวิชวลเบสิกได้อัตโนมัติ

ตารางที่ 5 - 2 แสดงส่วนประกอบของ Caché

5.4 Object Model

Caché เป็นระบบจัดการฐานข้อมูลเชิงวัตถุ ดังนั้นจึงจัดการและออกแบบระบบเชิงวัตถุทั้งหมด ซึ่งแบบจำลองวัตถุคาเช่ (Caché Object Model) สนับสนุนลักษณะดังต่อไปนี้

- คลาสหลักใน Caché จะเป็นออบเจ็กต์คลาส (Object Class) และดาต้าไทป์คลาส (Data Type Class) ที่รวมทั้ง Literal ด้วย โดยออบเจ็กต์ จะมีตัวชี้เฉพาะ (Unique Identifier) ส่วน Literal จะไม่มีตัวชี้เฉพาะ มีเพียงค่า (Value) เท่านั้น
- ออบเจ็กต์จะมีพฤติกรรมโดยอัตโนมัติซึ่งจัดการ โดย Caché หรือถูกกำหนดโดยผู้ใช้ก็ได้
- Persistent Object สามารถถูกจัดเก็บในฐานข้อมูล Caché ได้

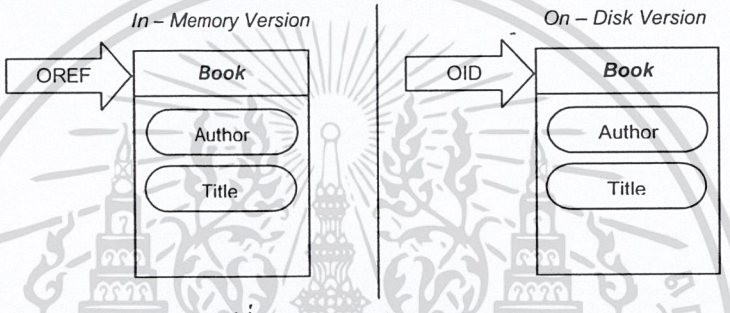
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

5.5 การอ้างอิงออบเจ็กต์

ในระบบจัดการของ Caché ออบเจ็กต์อาจอยู่ในดิสก์หรือในหน่วยความจำ ออบเจ็กต์ที่อยู่ในดิสก์จะเป็นออบเจ็กต์ที่ถูกจัดเก็บลงฐานข้อมูล ส่วนออบเจ็กต์ที่อยู่ในหน่วยความจำนั้นจะถูกโหลดดึงมาจากฐานข้อมูลและสามารถควบคุมได้ เราสามารถอ้างถึงออบเจ็กต์เหล่านี้ได้ 2 รูปแบบขึ้นอยู่กับว่าจะอ้างออบเจ็กต์นั้นหากมันถูกโหลดหรือถูกจัดเก็บอยู่ การอ้างอิงนั้นได้แก่

- OREF (Object Reference)
- OID (Object Identifier)

ดังแสดงในรูปที่ 5 - 2



รูปที่ 5 – 2 OREF และ OID

- OREF เป็นการอ้างถึงออบเจ็กต์ที่อยู่ในความจำ แต่ละครั้งที่ออบเจ็กต์ถูกโหลดขึ้นมาอยู่บนหน่วยความจำ มันอาจจะมีค่า OREF ที่แตกต่างกันได้
- OID เป็นการอ้างถึงออบเจ็กต์ที่อยู่ในดิสก์ เป็นตัวชี้ออบเจ็กต์แบบ Persistent และมีค่าไม่ซ้ำ (Unique) ซึ่งถูกเก็บอยู่ในดิสก์ เมื่อออบเจ็กต์ถูกสร้างขึ้นและได้รับ OID แล้ว ก็จะค่านั้นไม่เปลี่ยนแปลง

เราสามารถใช้ OID ของออบเจ็กต์ Persistent นี้โหลดดึงออบเจ็กต์ขึ้นมาอยู่ในหน่วยความจำ จากนั้นระบบสามารถแทนออบเจ็กต์ตัวนั้นด้วย OREF ทำให้แอพพลิเคชันสามารถอ้างออบเจ็กต์และเข้าถึงค่าต่าง ๆ ของออบเจ็กต์นั้น ๆ ได้

5.6 คลาส

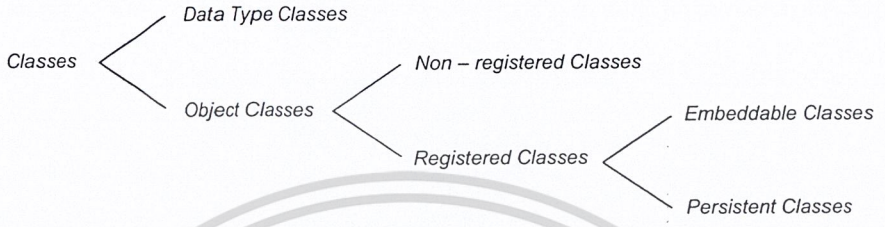
คลาส ใน Caché มี 5 ประเภท ได้แก่

- Registered Class
- Persistent Class
- Embeddable Class
- Non - Registered Class
- Data Type Class

คลาสต่างๆ เหล่านี้ถูกจัดเป็นดาต้าไทป์คลาสและออบเจ็กต์คลาส โดยดาต้าไทป์คลาสจะแทน Literal Value เช่น สตริง (string), ตัวเลข (Integer) หรือวันที่ (Date) เป็นต้น ออบเจ็กต์ส่วนใหญ่จะสืบ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ทอดคุณสมบัติมาจากคลาสของระบบที่เรียกว่า %RegisteredObject ซึ่งจัดการและมีเมธอดหลักๆ ของออบเจ็กต์ให้มา ส่วนคลาสที่ไม่ได้สืบทอดคุณสมบัติมาจาก %RegisteredObject เรียกว่า Non – Registed class ซึ่งคลาสชนิดนี้จะมีเพียงลักษณะพื้นฐานเท่านั้น เราจะต้องสร้างเมธอดให้แก่ออบเจ็กต์เหล่านี้เข้าไปเอง คลาสต่างๆ ใน Caché แสดงดังรูปที่ 5 – 3



รูปที่ 5 - 3 ประเภทของคลาส

5.6.1 Registered Class

อินสแทนซ์ของคลาสนี้สืบทอดคุณสมบัติมาจาก %RegisteredObject หรือเรียกว่าเป็น Registered Object โดยออบเจ็กต์เหล่านี้จะมีเซตของเมธอดพื้นฐานทั้งหมดที่ใช้จัดการออบเจ็กต์ที่อยู่ในหน่วยความจำ เราสามารถอ้างถึงออบเจ็กต์นี้โดยผ่าน OREF ได้

5.6.2 Persistent Class

อินสแทนซ์ องคลาสนี้สืบทอดคุณสมบัติมาจาก %Persistent และถูกเรียกว่า persistent object ซึ่งออบเจ็กต์เหล่านี้เป็น registered object ด้วย ออบเจ็กต์นี้มีความสามารถที่จะจัดการและเก็บตัวมันเองลงในฐานข้อมูลได้ จากถูกโหลดสู่หน่วยความจำ ออบเจ็กต์อื่นที่มันอ้างถึงจะไม่ถูกโหลดมาด้วย เพราะออบเจ็กต์เป็นอิสระต่อกันและเพียงแต่ชี้หากันเท่านั้น

หาก Persistent Object ถูกใช้เป็นแอตทริบิวต์หนึ่งออบเจ็กต์อื่น มันจะถูกชี้มาจากออบเจ็กต์นั้นๆ ยกตัวอย่างเช่น ให้ Doctor เป็น persistent object และเรากำหนดให้ Attribute ในออบเจ็กต์ตัวหนึ่งมีชนิดเป็น Doctor จะได้เป็น

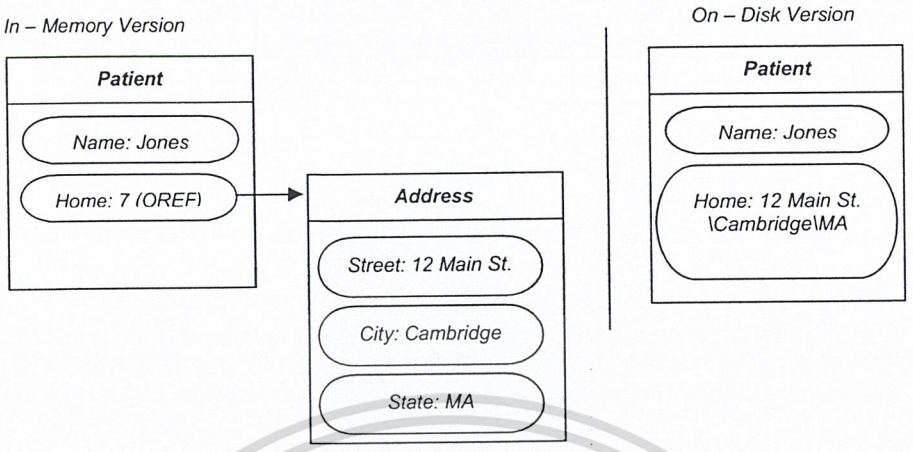
```
ATTRIBUTE TheDoc {TYPE = Doctor;}
```

ซึ่งแอตทริบิวต์e TheDoc จะชี้ไปยังออบเจ็กต์ชนิด Doctor

5.6.3 Embeddable Class

อินสแทนซ์ของคลาสนี้สืบทอดคุณสมบัติมาจาก %SerialObject และถูกเรียกว่า Embedded Object ซึ่งออบเจ็กต์เหล่านี้อ้างอิงได้อย่างอิสระเมื่อถูกโหลดสู่หน่วยความจำ แต่เมื่อจัดเก็บลงฐานข้อมูลแล้ว ออบเจ็กต์เหล่านี้จะต้องฝังตัวอยู่ภายใน Persistent Object เท่านั้น Embedded Object มีรูปแบบ 2 ลักษณะดังรูปที่ 5 - 4

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 5 - 4 Embedded Object ในหน่วยความจำและดิสก์

- ในหน่วยความจำ Embedded Object ถูกแทนด้วยออบเจ็กต์อิสระคล้าย ๆ กับ Persistent Object หากแอตทริบิวต์ใดอ้างชนิดเป็น Embedded Object แล้วการจัดการในหน่วยความจำก็สามารถกำหนดเป็น OREF ได้
- ในดิสก์ Embedded Object ถูกจัดเก็บเป็นเหมือนแอตทริบิวต์หนึ่งและถูกบรรจุลง Persistent Object ตัวหนึ่ง โดยมันจะไม่มี OID เป็นของตัวเองและไม่สามารถถูกอ้างโดยออบเจ็กต์ตัวอื่น ๆ อีกได้เลย ค่าของ Embedded Object จะถูกเก็บลงฐานข้อมูลคล้ายๆ เป็นเพียงสตริงในแอตทริบิวต์ของออบเจ็กต์หลักเท่านั้น เหตุนี้จึงเรียกว่าเป็น SerialObject

ยกตัวอย่างเช่น

```
CLASS Address
{
      SUPER = %SerialObject;
      ATTRIBUTE Street { TYPE = %String(MAXLEN=80); }
      ATTRIBUTE City { TYPE = %String(MAXLEN=3); }
      ATTRIBUTE State { TYPE = %String(MAXLEN=2); }
}
```

เมื่อกำหนดให้ออบเจ็กต์จากคลาส Address เป็น Embeded Object Attribute สำหรับ Persistent Object เช่นคลาส Patient จะได้

```
CLASS Patient
{
      SUPER = %Persistent;
      PERSISTENT;
      ATTRIBUTE Name { TYPE = %String; }
      ATTRIBUTE Home { TYPE = %Address; }
}
```

5.6.4 Non – Registered Class

อินสแทนซ์ของคลาสนี้เป็นออบเจ็กต์ที่เหลือ ที่ไม่ได้สืบทอดคุณสมบัติมาจากคลาส %RegisteredObject อาจเกิดขึ้นจากการโปรแกรมของผู้ใช้ขึ้นเอง และเราจำเป็นต้องกำหนดค่าสำหรับ OREF ขึ้นมาเองเพราะระบบไม่ได้เตรียมหรือรีจีสเตอร์ (Register) ค่าให้

5.6.5 Data Type Class

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

คลาสนี้กำหนดและควบคุมค่าของ Literal โดยดาต้าไทป์ ไม่มีตัวชี้อิสระเหมือนออบเจ็กต์ทั่วไป และไม่สามารถมีอินสแทนซ์ได้ ซึ่งมันเป็นเพียงแอตทริบิวต์ของออบเจ็กต์ซึ่งบรรจุมันเอาไว้ และตัวมันเองก็ไม่มี Property ด้วย ดาต้าไทป์คลาสจะแทน Literal Value เช่น สตริง, ตัวเลขเต็ม, วันที่, ตัวเลขทศนิยม (Float) และชนิดของข้อมูลพื้นฐานต่างๆ

5.6.6 Caché กับการถ่ายทอดคุณสมบัติ

แบบจำลองวัตถุคาเช่ อนุญาตให้เราสามารถออกแบบคลาสด้วยการสืบทอดคุณสมบัติจากคลาสอื่นมาได้โดยการใช้คีย์เวิร์ด SUPER คลาสที่ได้จะมี Specification เหมือนกับ Superclass ทุกประการ รวมทั้ง Property, Method และ Class Parameter ด้วย ยกตัวอย่างเช่นคลาส Person และมีคลาส Employee มาทำการสืบทอดคุณสมบัติจากคลาส Person ดังนี้

```
CLASS Person
{
      SUPER = %Persistent;
      PERSISTENT;
      ATTRIBUTE Name { TYPE = %String; }
      ATTRIBUTE DOB { TYPE = %Address; }
}
CLASS Employee
{
      SUPER = Person;
      PERSISTENT;
      ATTRIBUTE Salary { TYPE = %Integer; }
      ATTRIBUTE Department { TYPE = %String; }
}
```

5.6.7 Caché กับ Multiple Inheritance

คลาสสามารถสืบทอดคุณสมบัติมาได้จากหลาย ๆ Superclass โดยใช้คีย์เวิร์ด SUPER เช่นเดียวกัน เมื่อทำการคอมไพล์คลาส หากพบว่าสมาชิกต่างๆ เช่น ชื่อเมธอดของ Superclass มีชื่อซ้ำกัน Caché จะกำหนดให้คลาสที่สืบทอดลำดับหลังถูกเรียกให้งานก่อน ยกตัวอย่างเช่น คลาส X ทำการสืบทอดคุณสมบัติจากคลาส A, B และ C จะได้ว่า

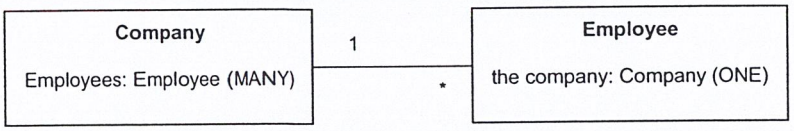
```
CLASS X
{
      SUPER = A, B, C;
}
```

ในกรณีนี้ค่าต่างๆ ของคลาส X จะสืบทอดคุณสมบัติและค่าต่างๆ จากคลาส A, B และ C ตามลำดับ (ดูจากลิสต์ก่อนหลัง) หากพบว่าคลาส B มีสมาชิกชื่อซ้ำกับคลาส A ที่ทำการสืบทอดคุณสมบัติมาแล้ว สมาชิกจากคลาส A จะถูกทับ (Override) ด้วยสมาชิกของคลาส B ทันที

5.6.8 Caché กับความสัมพันธ์ (Relationship)

ความสัมพันธ์เป็น Property ชนิดพิเศษซึ่งกำหนดความเกี่ยวข้องกันของออบเจ็กต์สองตัว เช่น คลาส Company มีความสัมพันธ์กับคลาส Employee ดังรูปที่ ในกรณีนี้อาจมีออบเจ็กต์ Employee ศูนย์หรือมากกว่าหนึ่งตัวที่เกี่ยวข้องกับออบเจ็กต์ Company แต่ละตัว

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 5 - 5 ความสัมพันธ์ระหว่างคลาส Company และคลาส Employee

สำหรับ Caché เวอร์ชัน 3.2.1.7 นี้ ความสัมพันธ์มีคุณลักษณะหลักๆ ดังนี้

- ความสัมพันธ์ถูกกำหนดขึ้นระหว่าง Persistent Class 2 คลาสเท่านั้น
- ความสัมพันธ์เป็น Bi - directional ซึ่งต้องกำหนดความสัมพันธ์ในทั้ง 2 ฝั่งคลาส
- สนับสนุนความสัมพันธ์แบบ One – to – many

5.6.9 การกำหนดความสัมพันธ์

มีคีย์เวิร์ดที่เกี่ยวข้องอยู่ 3 ตัว ได้แก่

- TYPE เป็นชนิด (ชื่อคลาส) ของคลาสที่เกี่ยวข้องและต้องเป็น Persistent Class
- INVERSE ชื่อของความสัมพันธ์ของอีกคลาสที่มีความสัมพันธ์ต่อกัน
- CARDINALITY มีค่าเป็น ONE หรือ MANY

จากรูปที่ ได้ดังนี้

```
CLASS Company
{
        SUPER = %Persistent;
        PERSISTENT;
        ATTRIBUTE Name { TYPE = %String; }

        //a Company has MANY Employees
        RELATIONSHIP Employees { TYPE = %Employee; INVERSE = the
        company; CARDINALITY = MANY;}
}

CLASS Employee
{
        SUPER = %Persistent;
        PERSISTENT;
        ATTRIBUTE Name { TYPE = %String; }
        ATTRIBUTE TItle { TYPE = %String; }

        //an Employee has ONE Company
        RELATIONSHIP TheCompany { TYPE = Company; INVERSE =
        Employees; CARDINALITY = ONE;}
}
```

5.6.10 Caché กับ Application

ในส่วนของการพัฒนาแอพพลิเคชัน (Implementing the Application) นั้น Caché ได้แบ่งออกเป็นขั้นตอนใหญ่ ๆ ดังนี้

- Setting Up a Namespace เป็นการกำหนดพื้นที่ในการจัดเก็บข้อมูลของโปรแกรมประยุกต์ โดยที่จะเป็นการกำหนดทาง Logical เท่านั้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- Creating the Application and its Classes สามารถสร้างคลาสโดยใช้ Cache' Object Architecture สามารถกำหนดชื่อ คำอธิบายและชนิดของคลาสได้ สำหรับ Cache' จะมี 3 คลาสที่ใช้ประจำ คือ Persistent Class (ออบเจ็กต์ที่ได้จะเป็นอิสระ เก็บข้อมูลได้เอง), Embeddable Class (ออบเจ็กต์ของคลาสนี้จะอยู่ในออบเจ็กต์ของคลาสอื่นอีกที) และสุดท้าย คือ Data type Class เป็นคลาสที่ไม่ใช้เก็บข้อมูลแต่มีไว้เพื่อควบคุมการทำงานของแอพพลิเคชัน
- Properties เป็นการระบุข้อมูลของคลาสที่ได้สร้างได้แล้วซึ่งจะต้องระบุชนิดของข้อมูลและเงื่อนไขต่างๆ ของ Preperty ไว้ด้วยเช่น unique, indexed, require เป็นต้น ขั้นตอนนี้ใช้ Cache' Object Architecture ช่วยในการทำงาน
- Method หลังจากกำหนดค่า Properties ของคลาสแล้ว จะต้องทำการกำหนดเมธอดของคลาสต่างๆ ขึ้นมาโดยใช้ภาษา Cache' ObjectScript เขียนขึ้นมาว่าจะให้คลาสที่เรากำหนดมานั้นสามารถทำงานหรือมีพฤติกรรมอย่างไรบ้าง
- Creating Query สร้างขึ้นเพื่อแสดงข้อมูลที่มีอยู่ในฐานข้อมูลของแอพพลิเคชันแทนที่จะเข้าไปดูข้อมูลโดยตรงกับออบเจ็กต์แต่ละตัวผ่านทาง OID โดยใช้ Cache' SQL Query หรือ Cache' ObjectScript

ในส่วนของการสร้าง Application Front – end สำหรับผู้ใช้นั้น สามารถทำได้โดยใช้ ActiveX, C++, Delphi, PowerBuilder ซึ่งจะติดต่อกับ Cache' ผ่านทาง Cache' ObjectServer แต่วิธีที่นิยมและใช้งานง่ายที่สุดคือการสร้างผ่านทาง Visual Basic

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 6

แนวคิดโปรแกรมประยุกต์กลาง(Common Application Concept)

6.1 ระบบโปรแกรมประยุกต์กลาง

แนวคิดระบบโปรแกรมประยุกต์กลางถูกนำมากล่าวถึงการสร้างโปรแกรมประยุกต์ เพื่อให้สามารถนำโปรแกรมประยุกต์นี้ไปใช้ได้กับระบบงานในหลายๆ องค์กรที่มีกฎเกณฑ์การทำงานที่หลากหลาย ซึ่งแต่เดิมการที่จะทำเช่นนี้ได้ สามารถทำได้สร้างไฟล์คอนฟิกกูเรชันสำหรับเก็บค่าตัวเลือกต่างๆ หรืออีกวิธีหนึ่งคือการนำโค้ดมาแก้ไขให้เหมาะสมกับระบบงานที่ต้องการแล้วสร้างใหม่ (Rebuild) แต่วิธีหลังนี้ จะต้องพึ่งความสามารถของผู้พัฒนาโปรแกรมเป็นหลัก

แนวคิดระบบโปรแกรมประยุกต์กลาง เข้ามามีบทบาทเพื่อแก้ปัญหาในกรณีหลัง เพื่อให้หลากหลายองค์กร สามารถแก้กฎ เพิ่มกฎ หรือละเลยกฎ ได้โดยไม่กระทบถึงโปรแกรมประยุกต์ แนวคิดนี้ได้นำคุณลักษณะของฐานข้อมูลเชิงวัตถุมาใช้ เนื่องจากมีคุณสมบัติเชิงวัตถุ เราจะเขียนกฎการตรวจสอบค่า กฎการเปลี่ยนแปลงค่า หรือวิธีการคำนวณค่าได้ภายในตัวจัดการฐานข้อมูลเชิงวัตถุ (OODBMS) ซึ่งในปัจจุบันการแก้ปัญหาโปรแกรมประยุกต์ทำงานไม่เข้ากับความต้องการของระบบงาน (system requirement) ทำให้ต้องนำโปรแกรมกลับมาพัฒนาใหม่ ซึ่งเมื่อพัฒนาเสร็จแล้วจะต้องนำมาลง (install) ให้กับทุกๆ เครื่องที่ต้องใช้ระบบงานนี้ ซึ่งเป็นเรื่องที่ลำบาก ถ้าเครื่องที่ใช้โปรแกรมประยุกต์ไม่ได้อยู่ภายในองค์กรหรือจำนวนเครื่องมีจำนวนมาก

แนวคิดระบบโปรแกรมประยุกต์กลางจะแก้ไขกฎเกณฑ์ต่างๆ ที่ตัวจัดการฐานข้อมูล ทำให้ผู้ใช้ไม่รับทราบถึงการแก้ไข ไม่ได้รับผลกระทบต่างๆ เนื่องจากการแก้ไข และโปรแกรมที่เครื่องต่างๆ ไม่จำเป็นต้องลงใหม่ (ถ้าไม่มีการเปลี่ยนอินเทอร์เฟสที่ตัวฐานข้อมูลเชิงวัตถุ) การวิจัยนี้จะนำระบบสารสนเทศหลักสูตรมาทดสอบความเป็นไปได้สำหรับการนำไปประยุกต์ใช้จริง

6.2 กรณีศึกษา : ระบบงานหลักสูตร

ระบบงานหลักสูตร เป็นระบบงานที่จัดการกับหลักสูตร รายวิชาที่เปิดสอน การจัดตารางเรียน การพิจารณาจบหลักสูตร และ การตรวจสอบการลงทะเบียนของนักศึกษาเพื่อให้เป็นไปตามกฎของแต่ละสถาบัน การออกแบบระบบโดยวิธีเชิงวัตถุมีรายละเอียดขงอแต่ขั้นตอนดังนี้

ความต้องการของระบบ (Requirements of System)

ระบบงานประกอบไปด้วนความต้องการทางด้านความสามารถของระบบงาน และความต้องการทางด้านการติดต่อกับผู้ใช้

ความต้องการของระบบด้านความสามารถของระบบ

ระบบต้องสามารถจัดการกับหลักสูตร โดยการเพิ่ม แก้ไข และลบหลักสูตรได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ระบบต้องสามารถจัดการกับข้อมูลต่างๆ ที่จำเป็นต้องใช้ร่วมกับระบบหลักสูตรได้ เช่น รายวิชาที่เปิดสอน คณาจารย์ นักศึกษา และการจัดการในระดับหมวดหมู่ของหลักสูตร

ระบบต้องสามารถพิจารณาจบหลักสูตรได้ โดยนักศึกษาแต่ละคน/ภาควิชา/คณะ อาจมีความแตกต่างกันในการใช้วิธีพิจารณาฯ

ระบบต้องสามารถกำหนดโดเมนของข้อมูลเข้า (Data-Input) ได้

ระบบต้องสามารถแก้ไขวิธีการพิจารณาจบฯ โดยกระทำที่ DBMS (โดยแก้ไขที่ Caché) และไม่กระทบ Front-End Application

ความต้องการของระบบด้านการติดต่อกับผู้ใช้

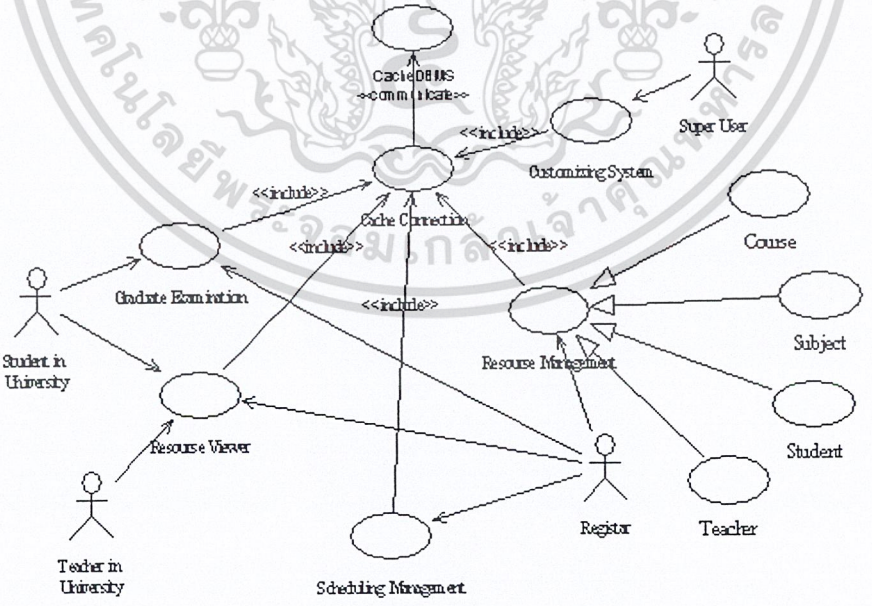
โปรแกรมประยุกต์ต้องมีการใช้งานที่ง่าย โดยสามารถมองเห็นข้อมูลทุกอย่างได้ในหน้าต่างเดียว (เลียนแบบ Window Explorer)

โปรแกรมประยุกต์ต้องสามารถสลับหน้าต่างการใช้งาน (กรณีเปิดหลายระบบงาน) ได้ง่ายโดยเลียนแบบวิธีของ EditPlus

โปรแกรมประยุกต์ต้องมีอินเทอเฟสแบบ WindowXP

6.3 การออกแบบโดยใช้ UML

6.3.1 Use Case Diagram



รูปที่ 6 -1 แสดง Use Case ของระบบงานหลักสูตร

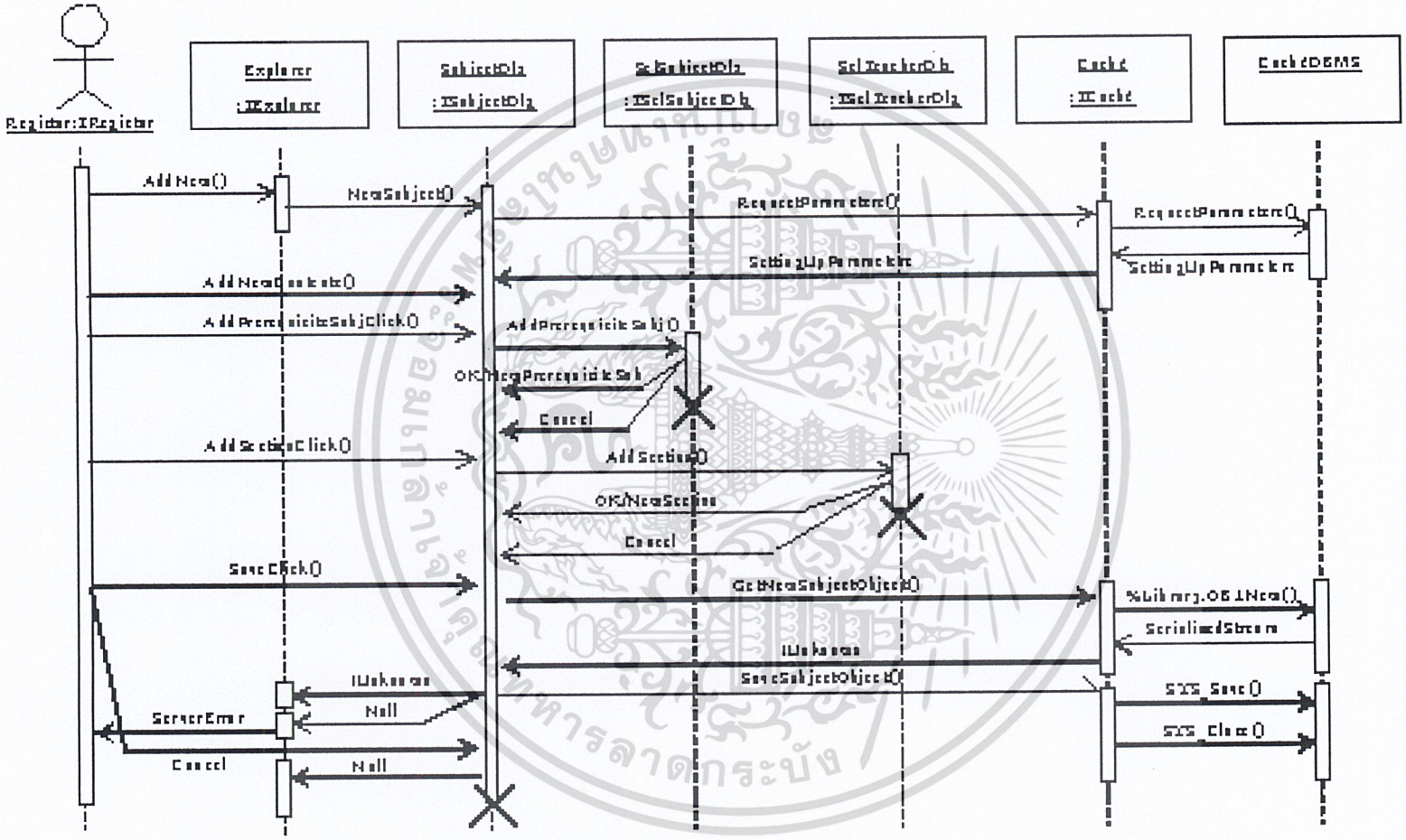
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ชื่อสัญลักษณ์	ความหมาย
CachéDBMS (usecase)	แสดงระบบฐานข้อมูลเชิงวัตถุ Caché ในส่วนที่เป็นฐานข้อมูล
Registar	นายทะเบียนหรือเจ้าหน้าที่สำหรับระบบงาน
Student	นักศึกษาที่ต้องการลงทะเบียนหรือดูข้อมูล
Teacher	คณาจารย์ที่ต้องการดูข้อมูล
Super User	ผู้ดูแลระบบ ที่สามารถ customize ระบบได้
Graduation Examination	งานสำหรับการพิจารณาจบหลักสูตรเพื่อตอบคำถามว่า นักศึกษาที่เข้าประเมิน ผ่านเกณฑ์ของหลักสูตรหรือไม่
Register Constraint Checking	งานตรวจสอบการลงทะเบียนเพื่อตอบคำถามว่า สามารถลงได้หรือไม่ได้ตามกฎของแต่ละสถาบัน
Resource Viewer	งานแสดงข้อมูลทั้งหมดเพื่อแสดงข้อมูลในระบบคร่าวๆ
Scheduling Management	งานจัดการตารางการเรียนของแต่ละรายวิชา/กลุ่ม/คณาจารย์ผู้สอน
Resource Management	งานจัดการข้อมูลต่างๆ โดยรวมของระบบ
Customizing System	งานจัดการ Customize ระบบสำหรับ Super User
Caché Connection	งานช่วยระบบงานต่างๆ ในการติดต่อกับ CachéDBMS

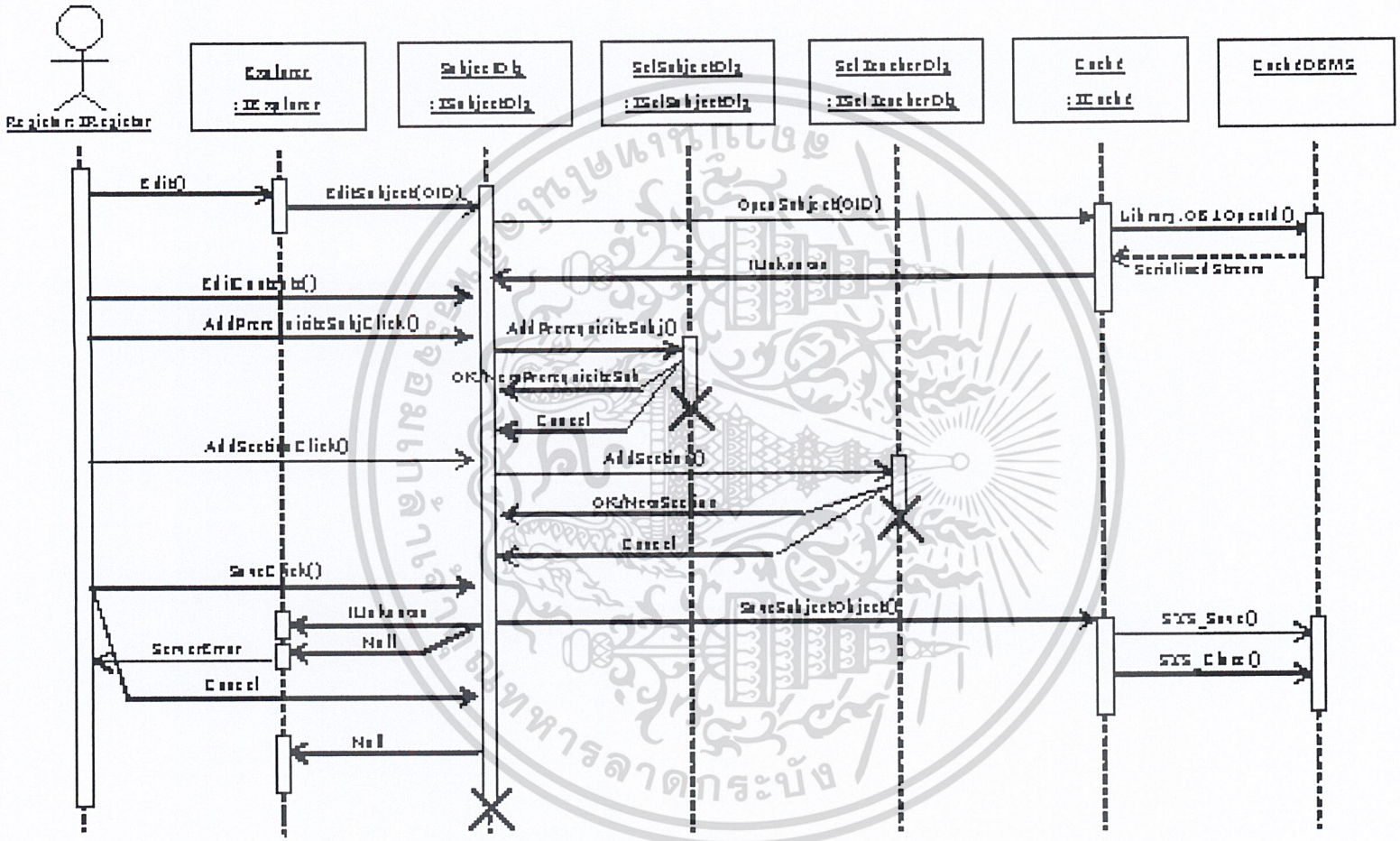
ตารางที่ 6 - 1 แสดงความหมายของแต่ละยูสเคสและแอ็คเตอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

6.3.2 Sequence Diagram

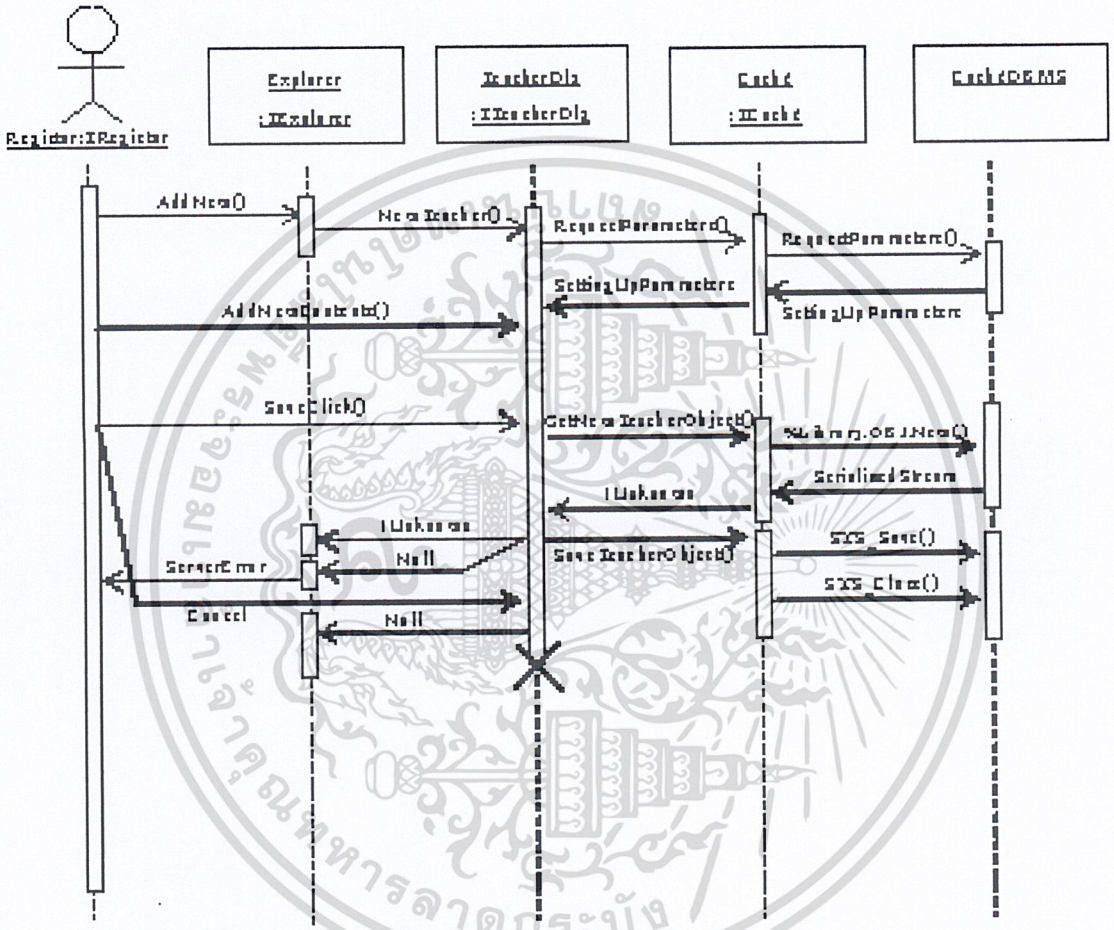


รูปที่ 6 – 2 แสดงปฏิสัมพันธ์ของระบบการเพิ่มรายวิชา



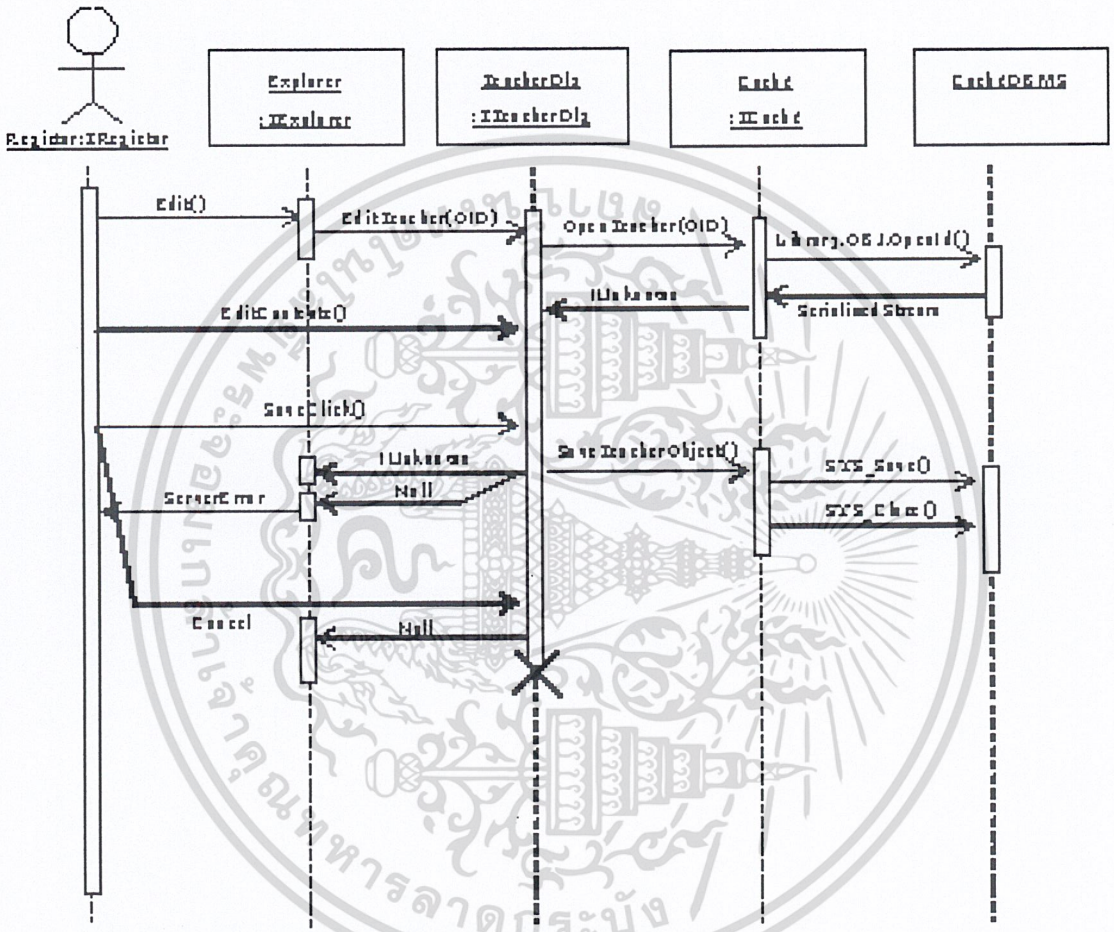
รูปที่ 6 - 3 แสดงปฏิสัมพันธ์ของระบบการแก้ไขรายวิชา

ไม่ควรนำไปใช้ประโยชน์ด้านการค้า



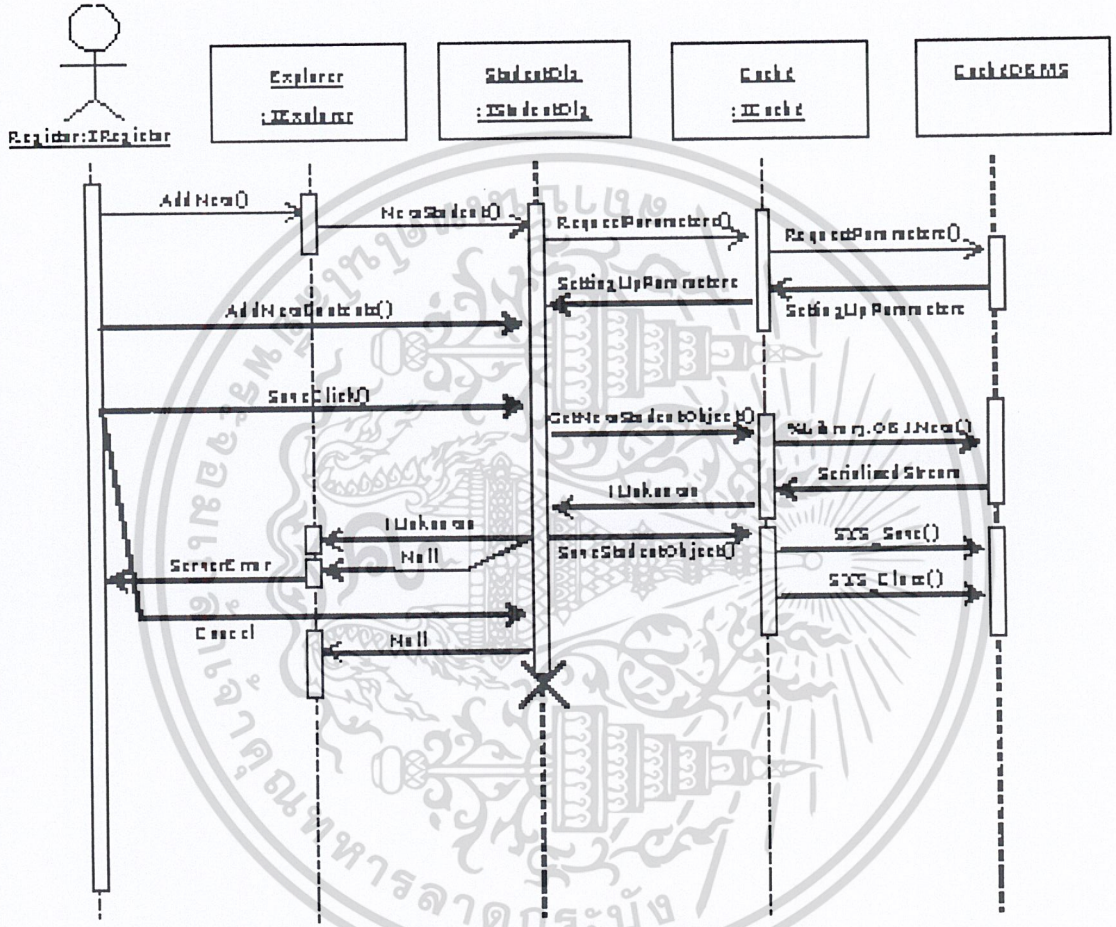
รูปที่ 6 - 4 แสดงปฏิสัมพันธ์ของระบบการเพิ่มข้อมูลคณาจารย์

ลิขสิทธิ์ของมหาวิทยาลัยราชภัฏกาญจนบุรี ไม่อนุญาตให้ทำซ้ำ ดัดแปลง เผยแพร่ หรือนำไปใช้ประโยชน์ด้านการค้า



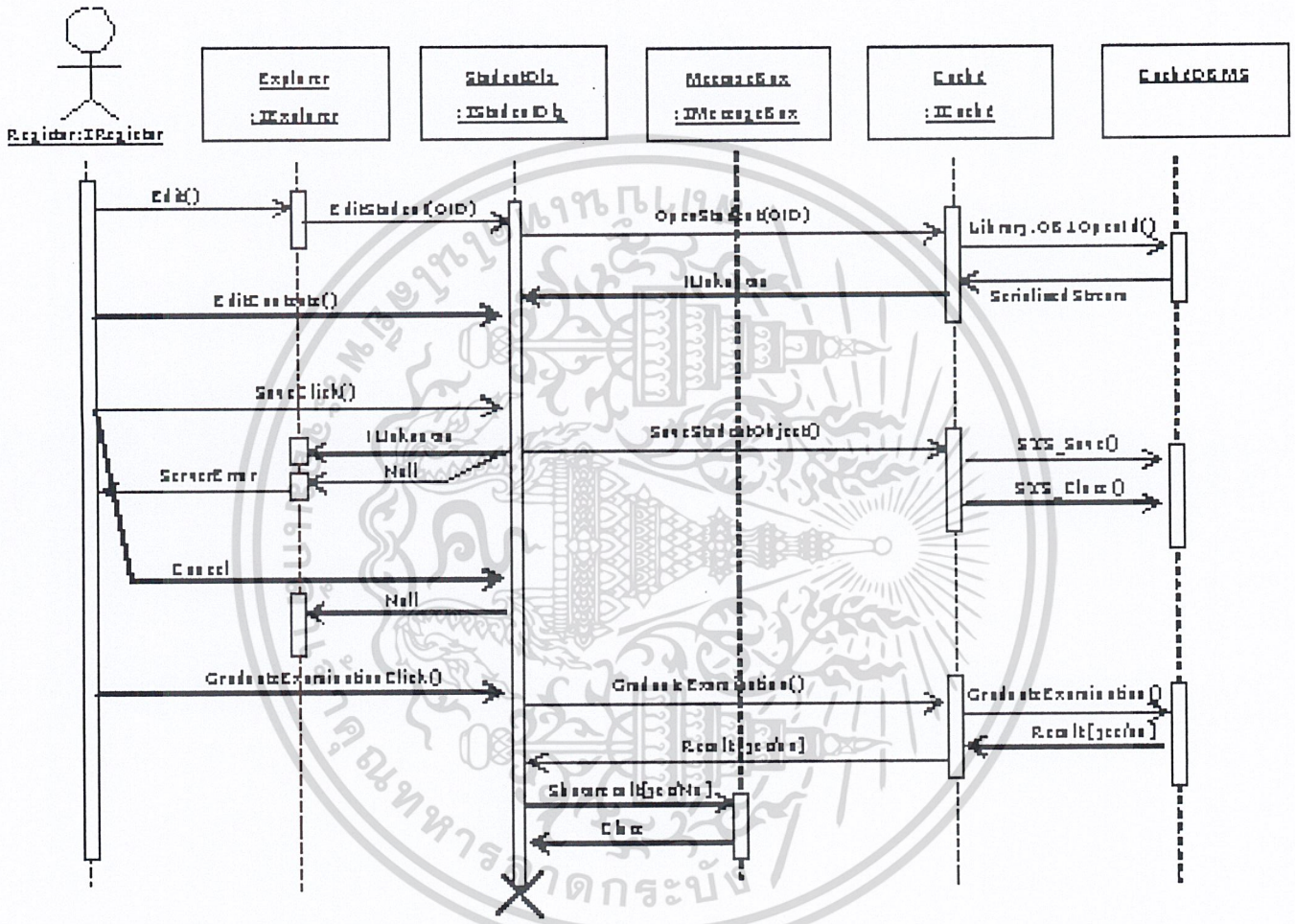
รูปที่ 6 - 5 แสดงปฏิสัมพันธ์ของระบบการแก้ไขข้อมูลคณาจารย์

ลิขสิทธิ์ของมหาวิทยาลัยราชภัฏรำไพพรรณี ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

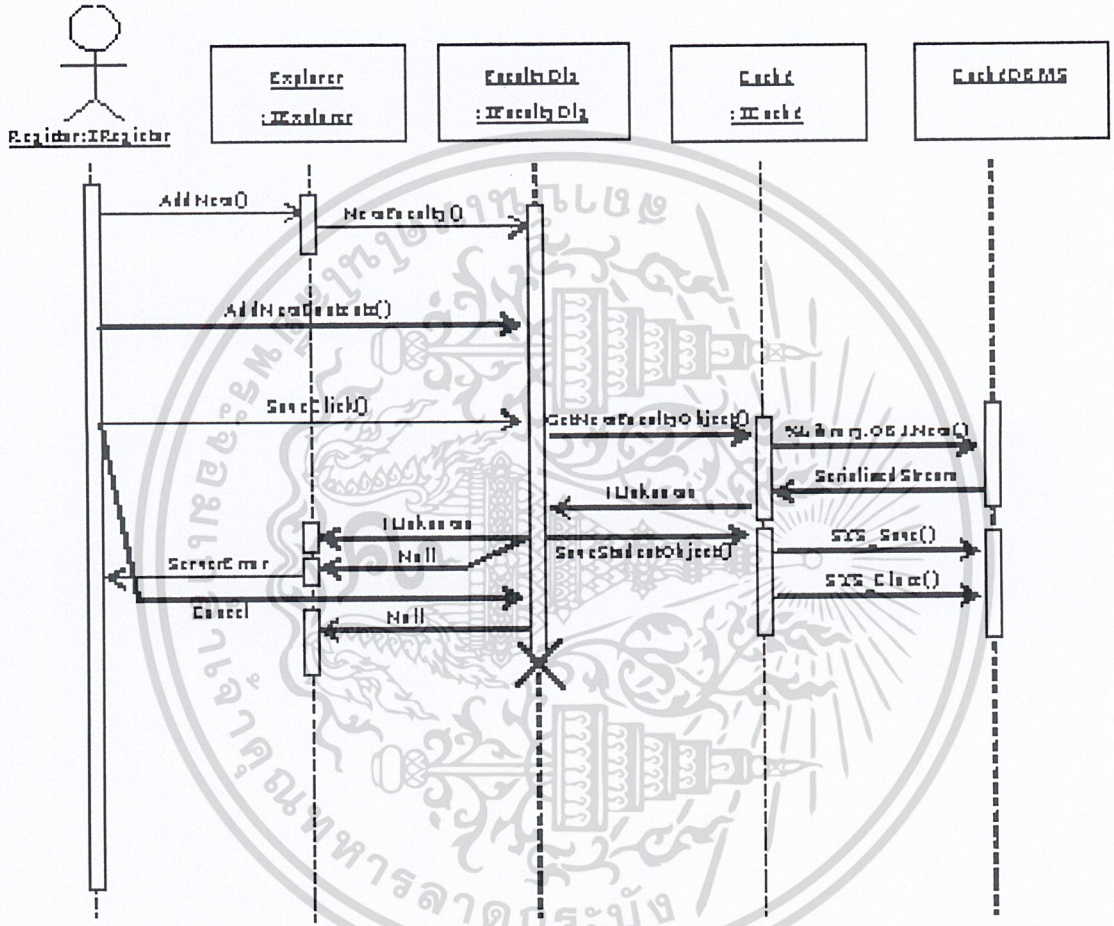


รูปที่ 6 - 6 แสดงปฏิสัมพันธ์ของระบบการเพิ่มข้อมูลนักศึกษา

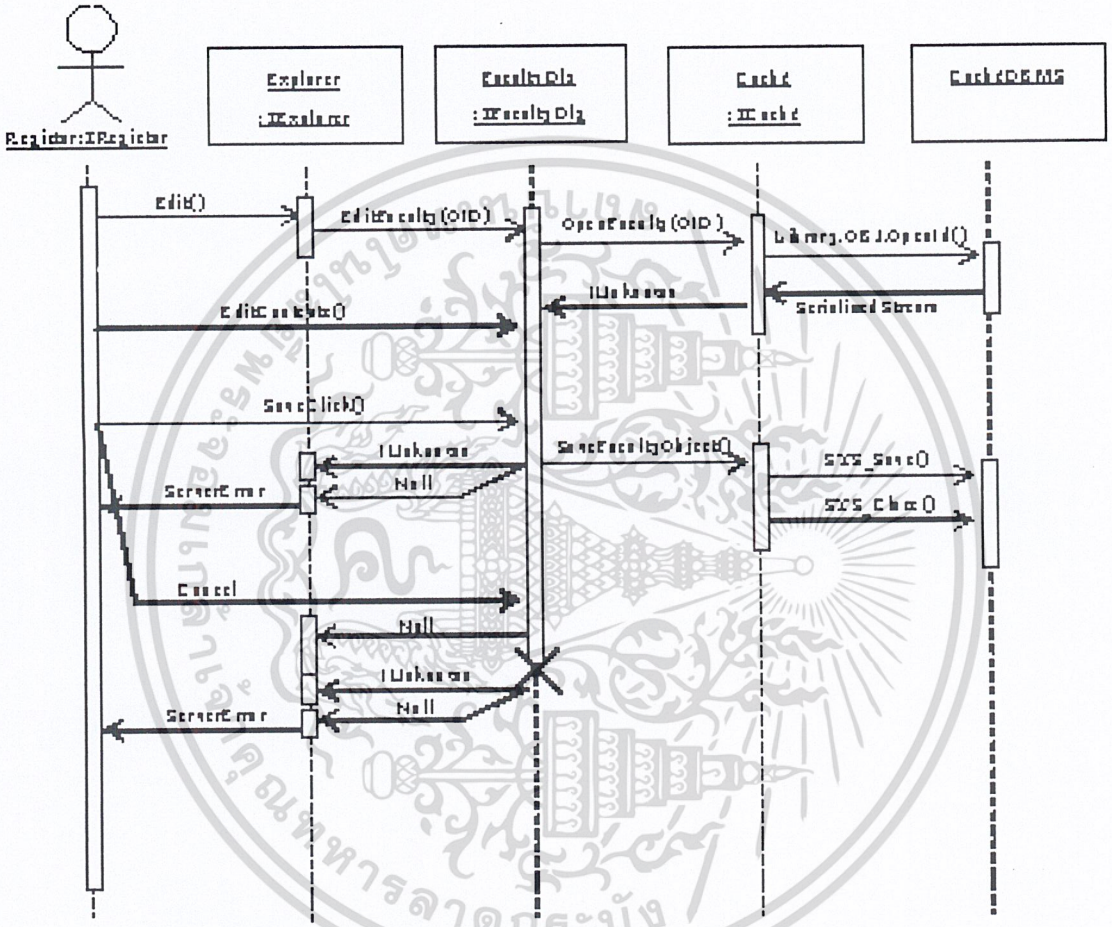
ลิขสิทธิ์ของมหาวิทยาลัยราชภัฏราชนครินทร์ ไม่อนุญาตให้ทำซ้ำ ดัดแปลง เผยแพร่ หรือนำไปใช้ประโยชน์ด้านการค้า



รูปที่ 6 - 7 แสดงปฏิสัมพันธ์ของระบบการแก้ไขข้อมูลนักศึกษา

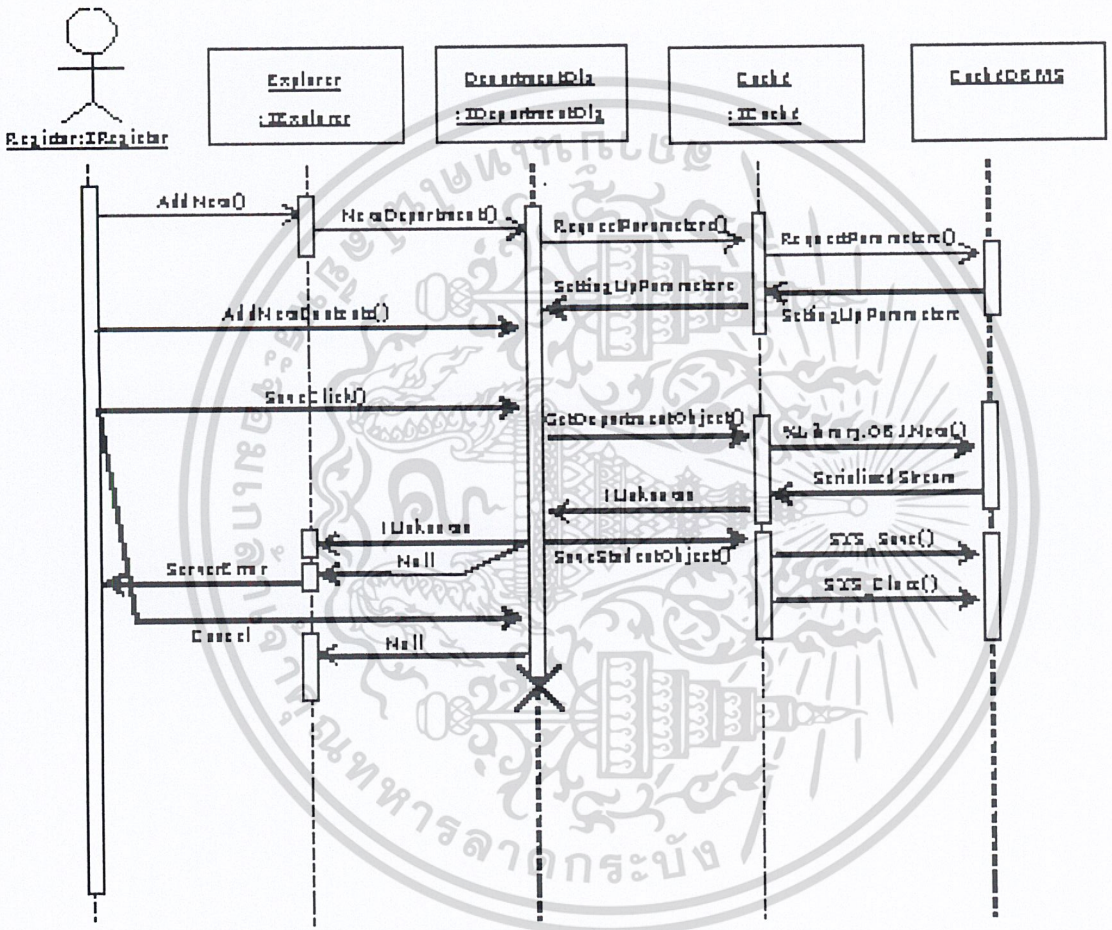


รูปที่ 6 - 8 แสดงปฏิสัมพันธ์ของระบบการเพิ่มข้อมูลคณะ



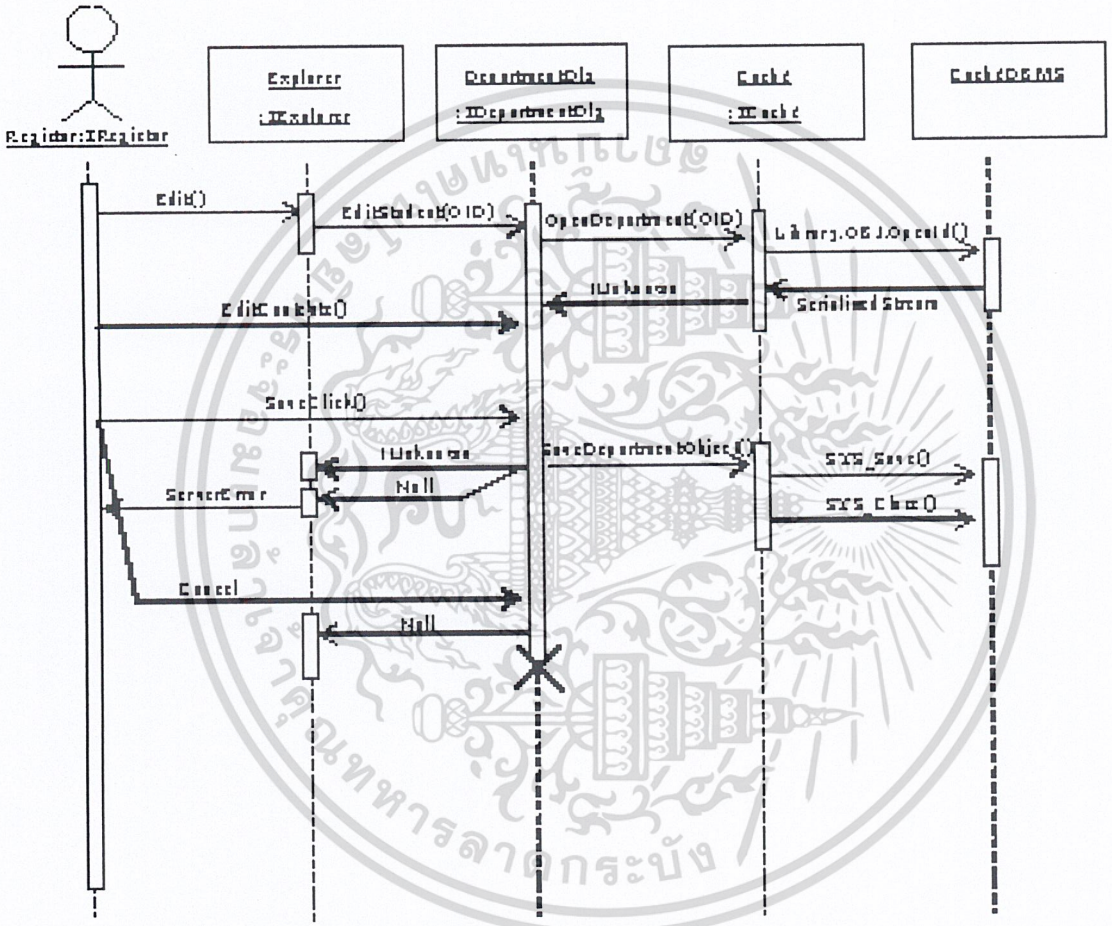
รูปที่ 6 - 9 แสดงปฏิสัมพันธ์ของระบบการแก้ไขข้อมูลคณะ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งไม่อนุญาตให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

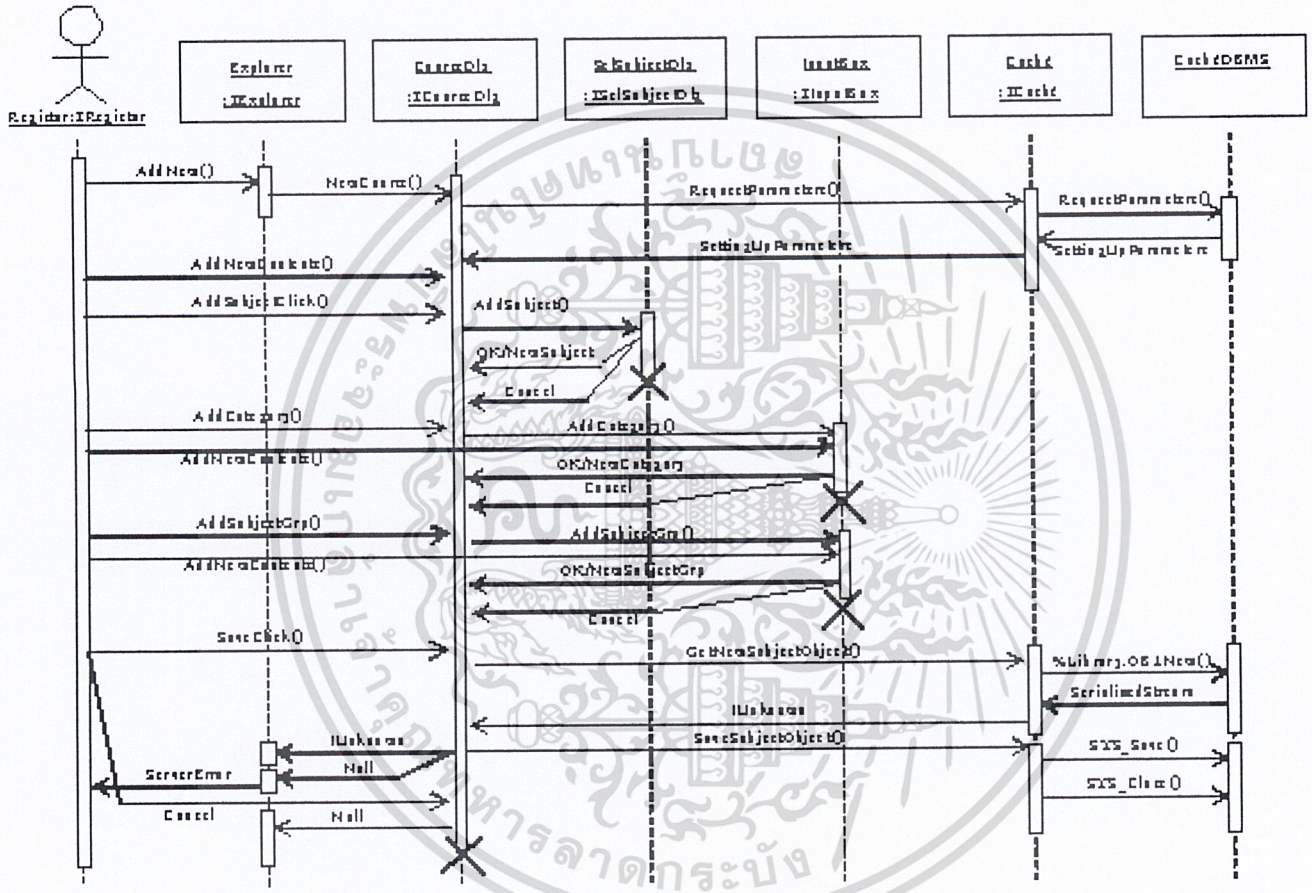


รูปที่ 6 - 10 แสดงปฏิสัมพันธ์ของระบบการเพิ่มข้อมูลภาควิชา

ลิขสิทธิ์ของจุฬาลงกรณ์มหาวิทยาลัย
ห้ามมิให้นำไปใช้ประโยชน์ด้านการค้า

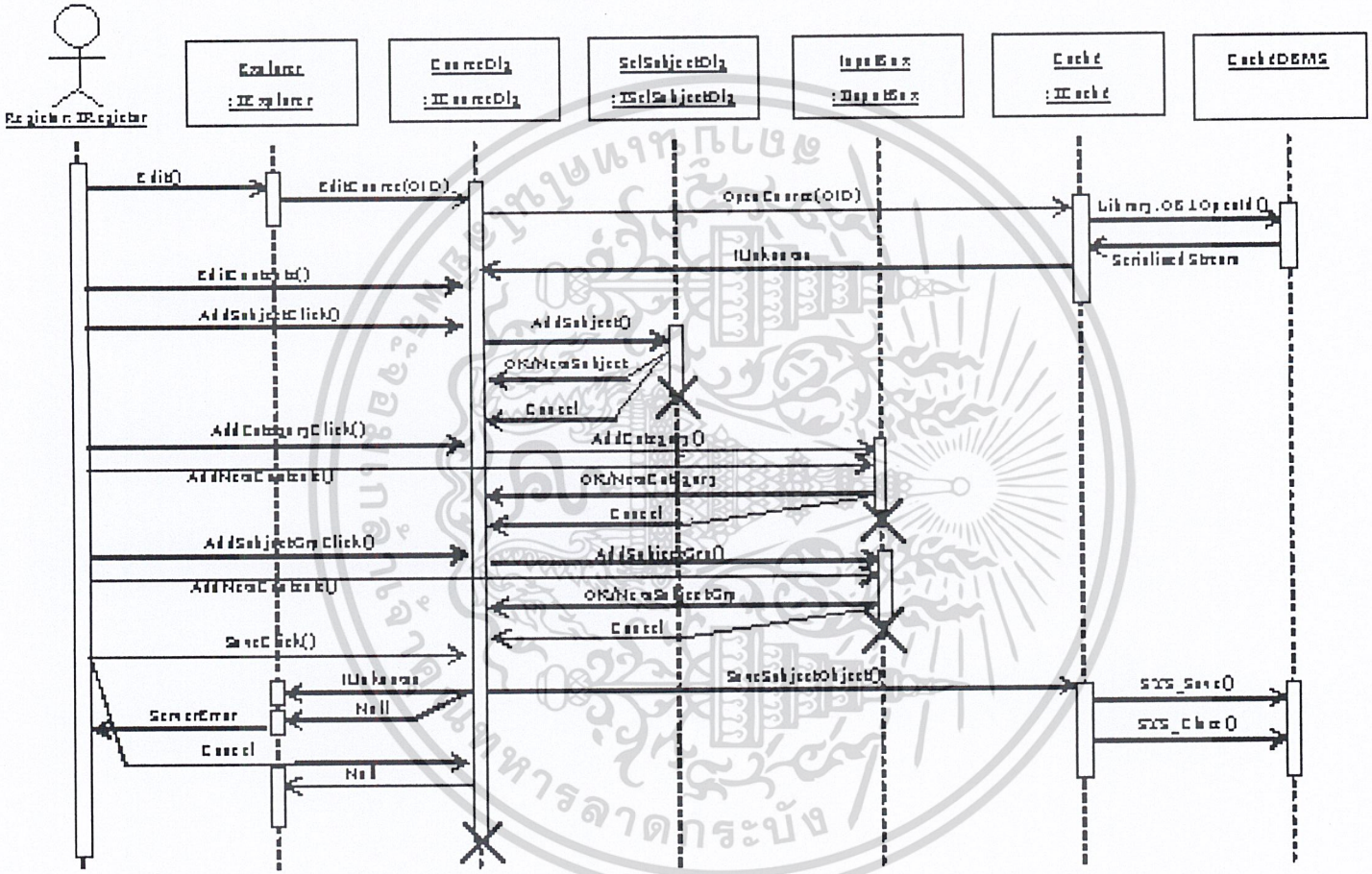


รูปที่ 6 - 11 แสดงปฏิสัมพันธ์ของระบบการแก้ไขข้อมูลภาควิชา



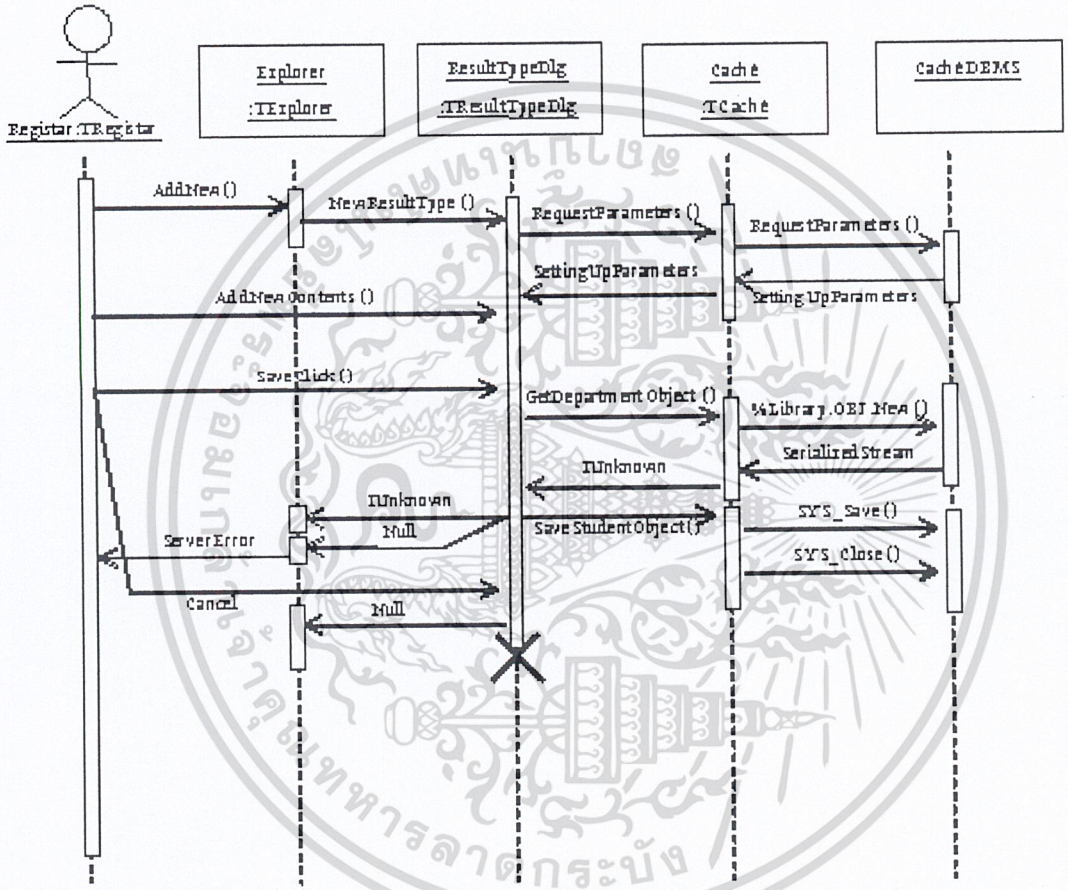
รูปที่ 6 – 12 แสดงปฏิสัมพันธ์ของระบบการเพิ่มข้อมูลหลักสูตร

ลิขสิทธิ์ของจุฬาลงกรณ์มหาวิทยาลัย ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า



รูปที่ 6 - 13 แสดงปฏิสัมพันธ์ของระบบการแก้ไขข้อมูลหลักสูตร

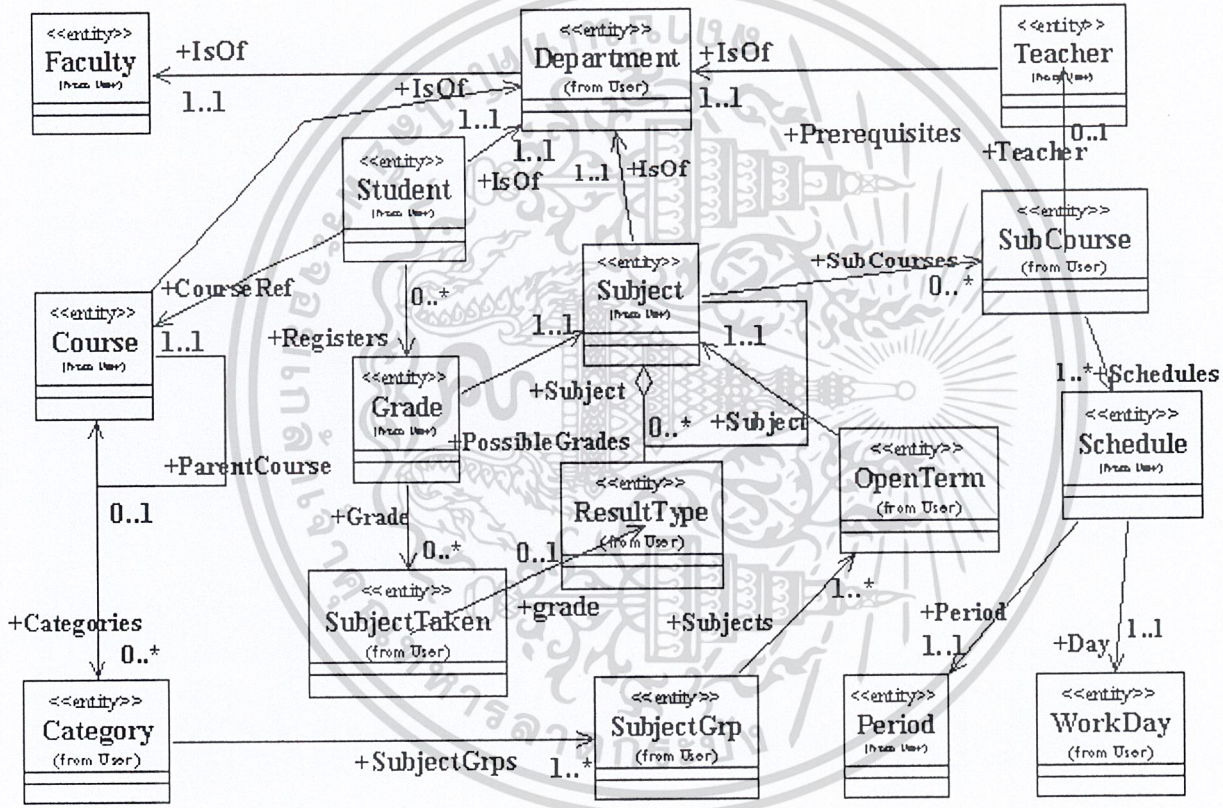
ลิขสิทธิ์มหาวิทยาลัยราชภัฏรำไพพรรณี ไม่อนุญาตให้ทำซ้ำ ดัดแปลง เผยแพร่ หรือนำไปใช้ประโยชน์ด้านการค้า



รูปที่ 6 - 14 แสดงปฏิสัมพันธ์ของระบบการเพิ่มข้อมูลผลการเรียน

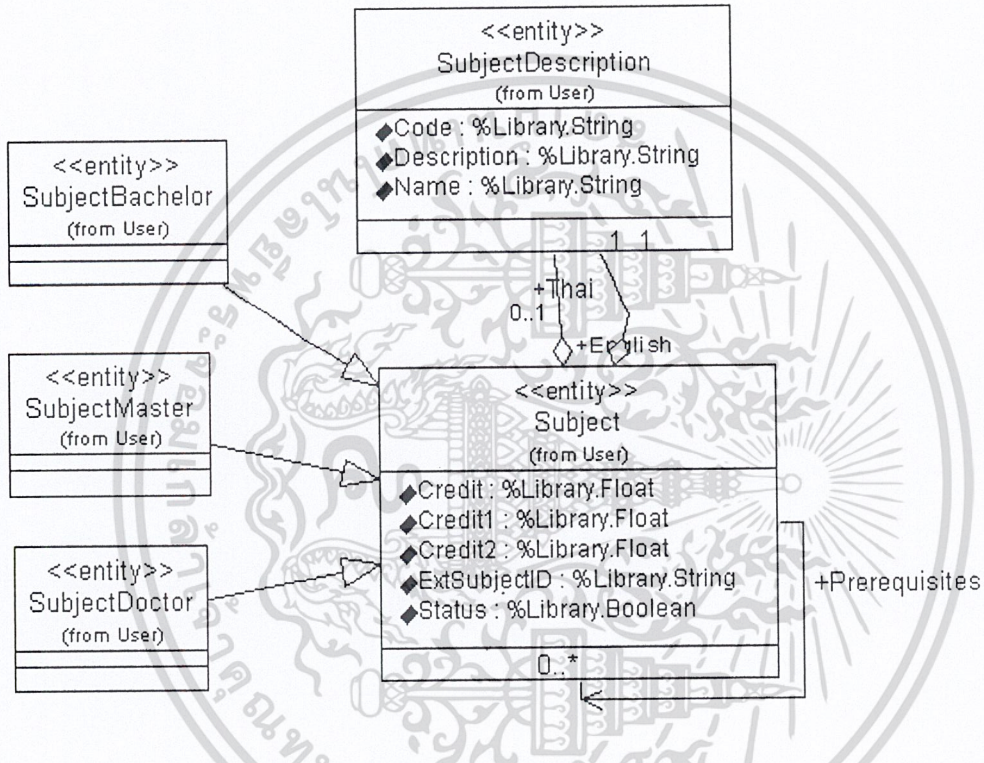
6.3.3 Class Diagram

Class Diagram – แสดงคลาสข้อมูลในระดับที่ 1 ที่ใช้ในระบบ



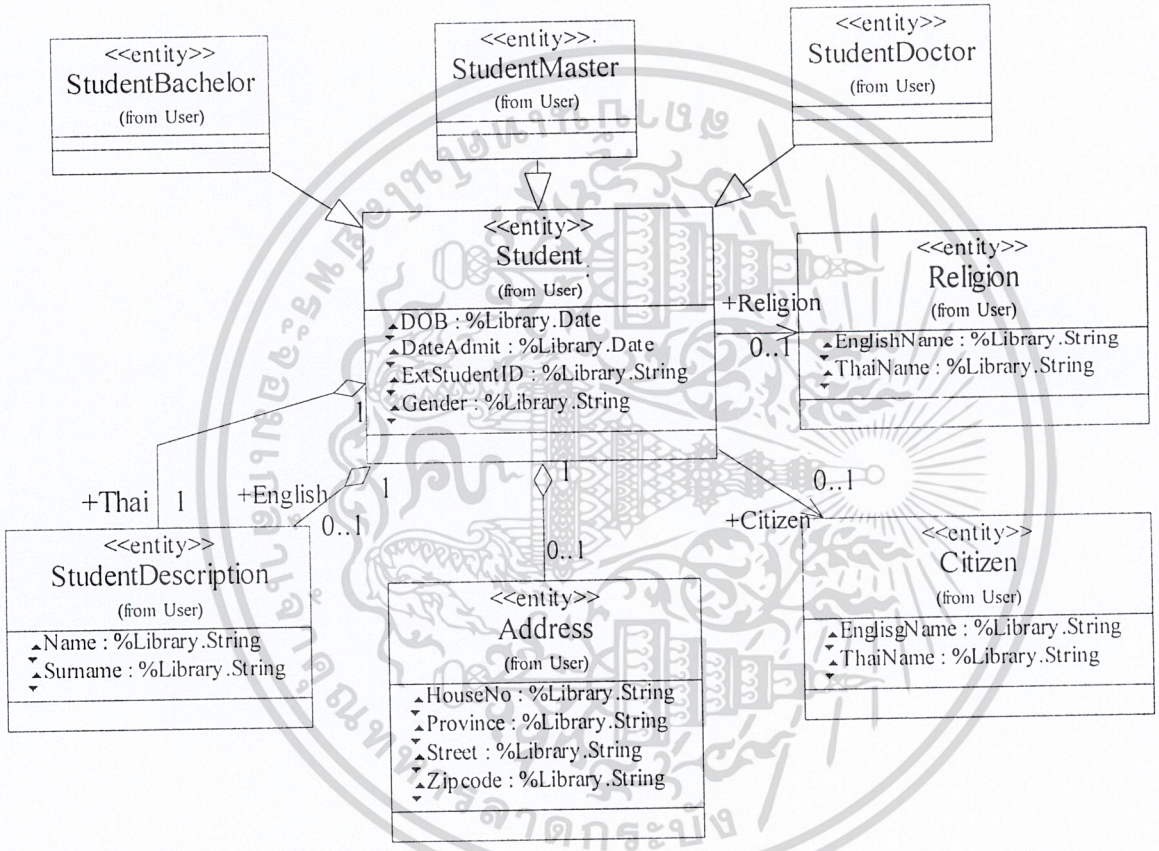
รูปที่ 6 – 15 แสดงคลาสฐานข้อมูลในระดับ 1

Class Diagram – แสดงคลาสข้อมูลในระดับที่ 2 ที่ใช้ในระบบ



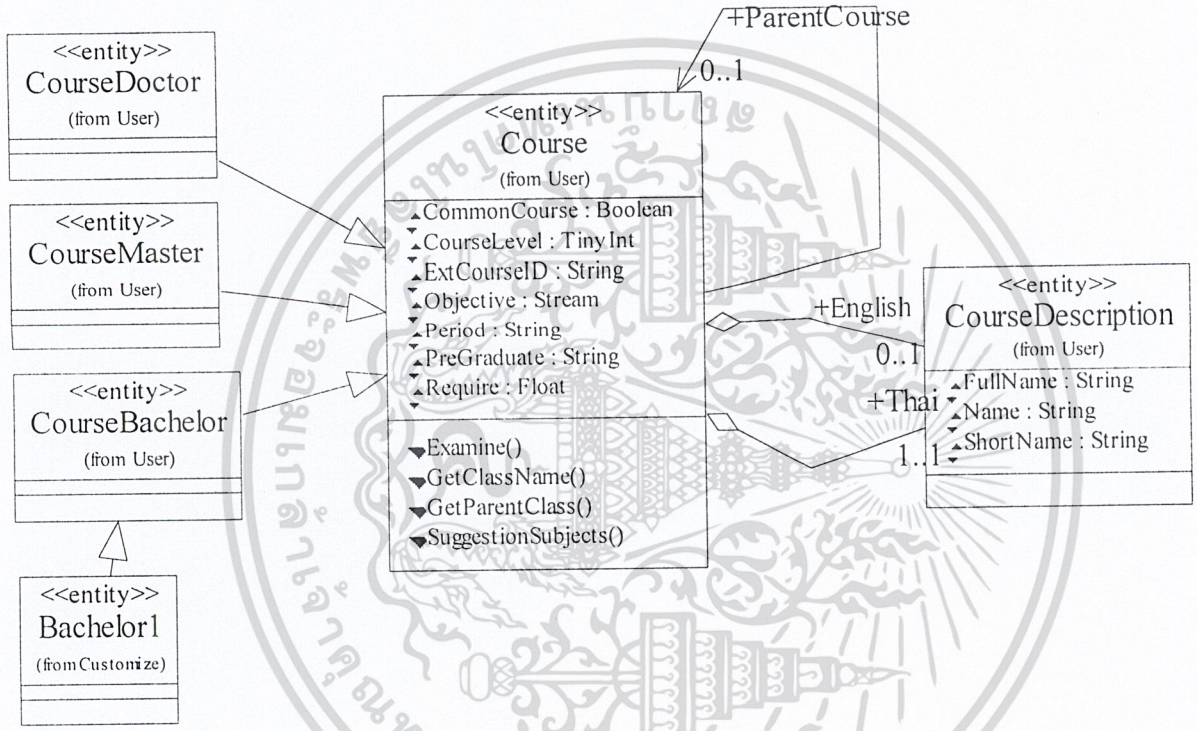
รูปที่ 6 - 16 แสดงคลาสรายวิชาและการสืบทอดคลาส

Class Diagram – แสดงคลาสข้อมูลในระดับที่ 2 ที่ใช้ในระบบ



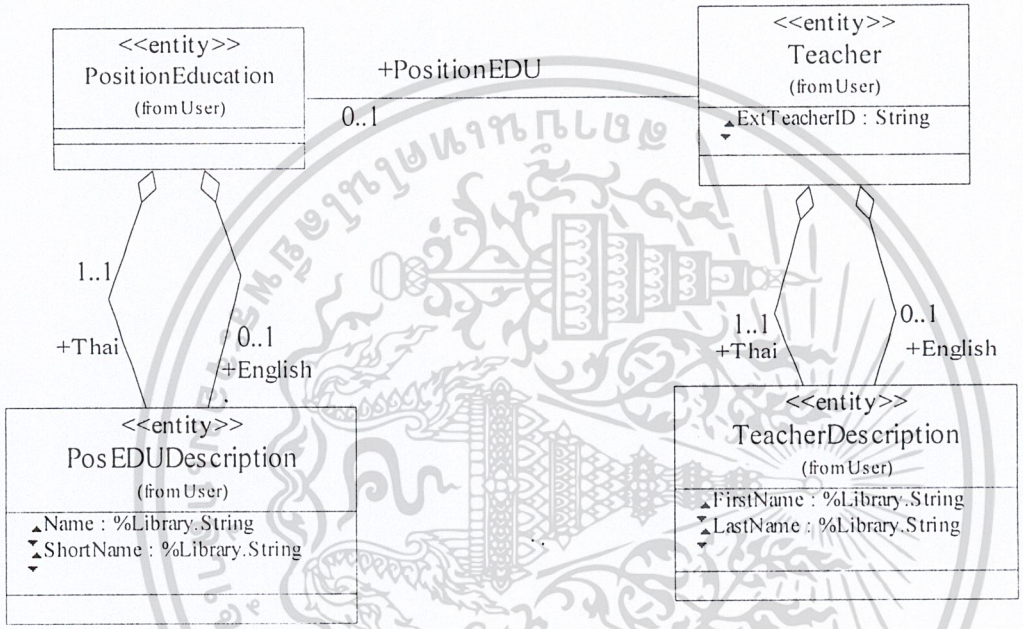
รูปที่ 6 – 17 แสดงคลาสนักศึกษาและการสืบทอด

Class Diagram – แสดงคลาสข้อมูลในระดับที่ 2 ที่ใช้ในระบบ



รูปที่ 6 - 18 แสดงคลาสหลักสูตรและการสืบทอด

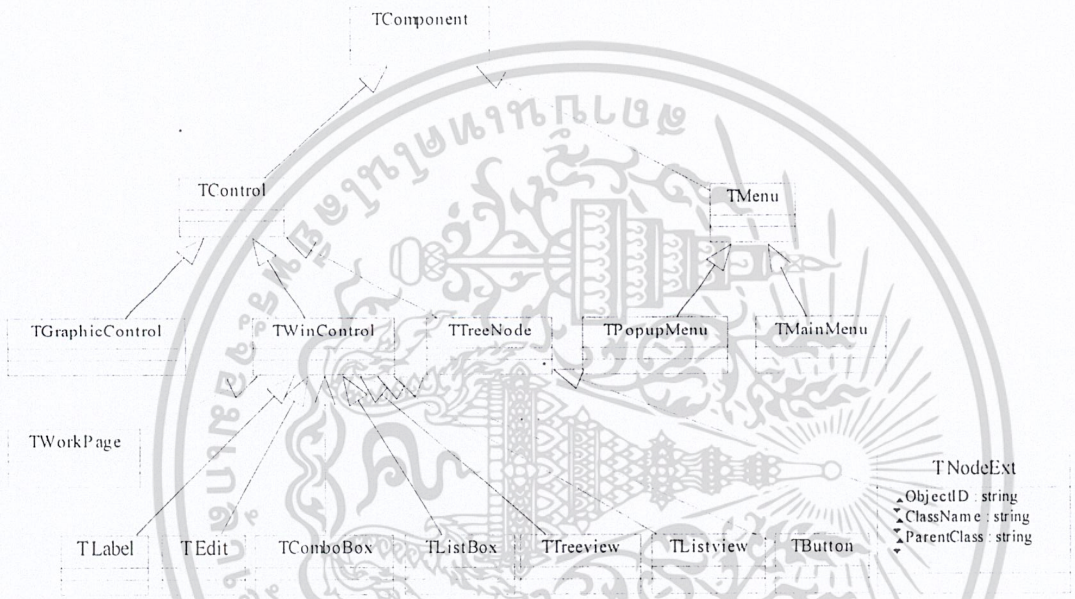
Class Diagram – แสดงคลาสข้อมูลในระดับที่ 2 ที่ใช้ในระบบ



รูปที่ 6 - 19 แสดงคลาสคณาจารย์และการสืบทอด

ลิขสิทธิ์ของมหาวิทยาลัยราชภัฏสวนสุนันทา ไม่อนุญาตให้ทำซ้ำ ดัดแปลง เผยแพร่ หรือนำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น ก่อนได้รับอนุญาตเป็นลายลักษณ์อักษรจากมหาวิทยาลัย

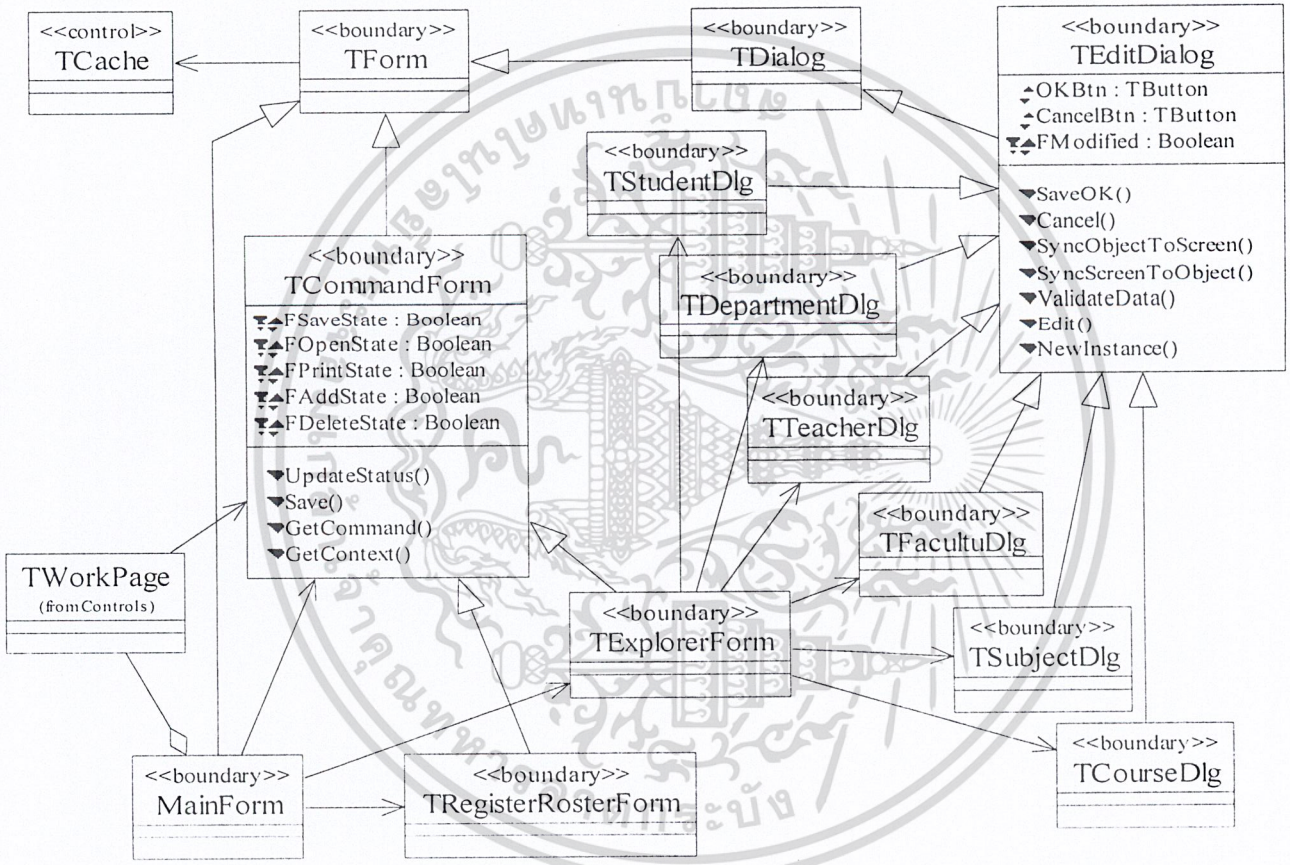
Class Diagram – แสดงคลาสคอนโทรลอินเทอเฟสสำหรับติดต่อผู้ใช้ บน Delphi-IDE



รูปที่ 6 - 20 แสดงคลาสคอนโทรลที่ใช้ใน IDE

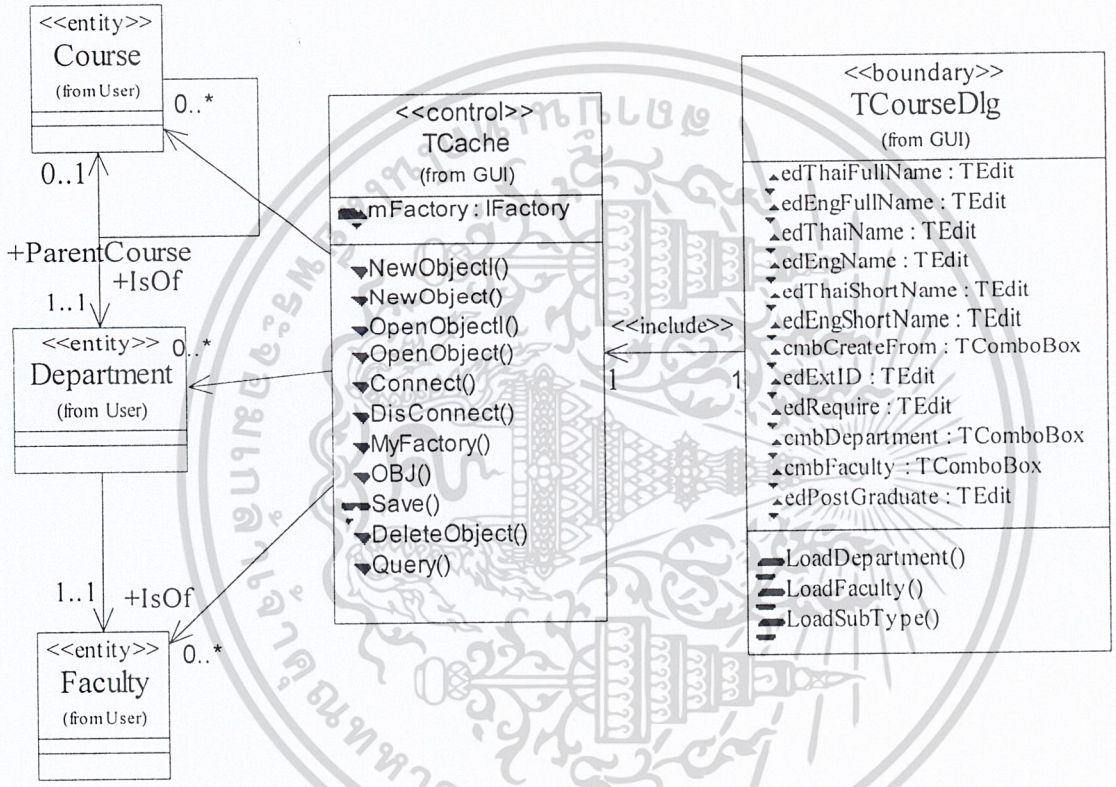
ลิขสิทธิ์ของมหาวิทยาลัยราชภัฏธนบุรี ไม่อนุญาตให้ทำซ้ำ ดัดแปลง หรือนำไปใช้ประโยชน์ด้านการค้า

Class Diagram – แสดงคลาสวินโดว์อินเทอเฟสสำหรับติดต่อผู้ใช้ บน Delphi-IDE-1



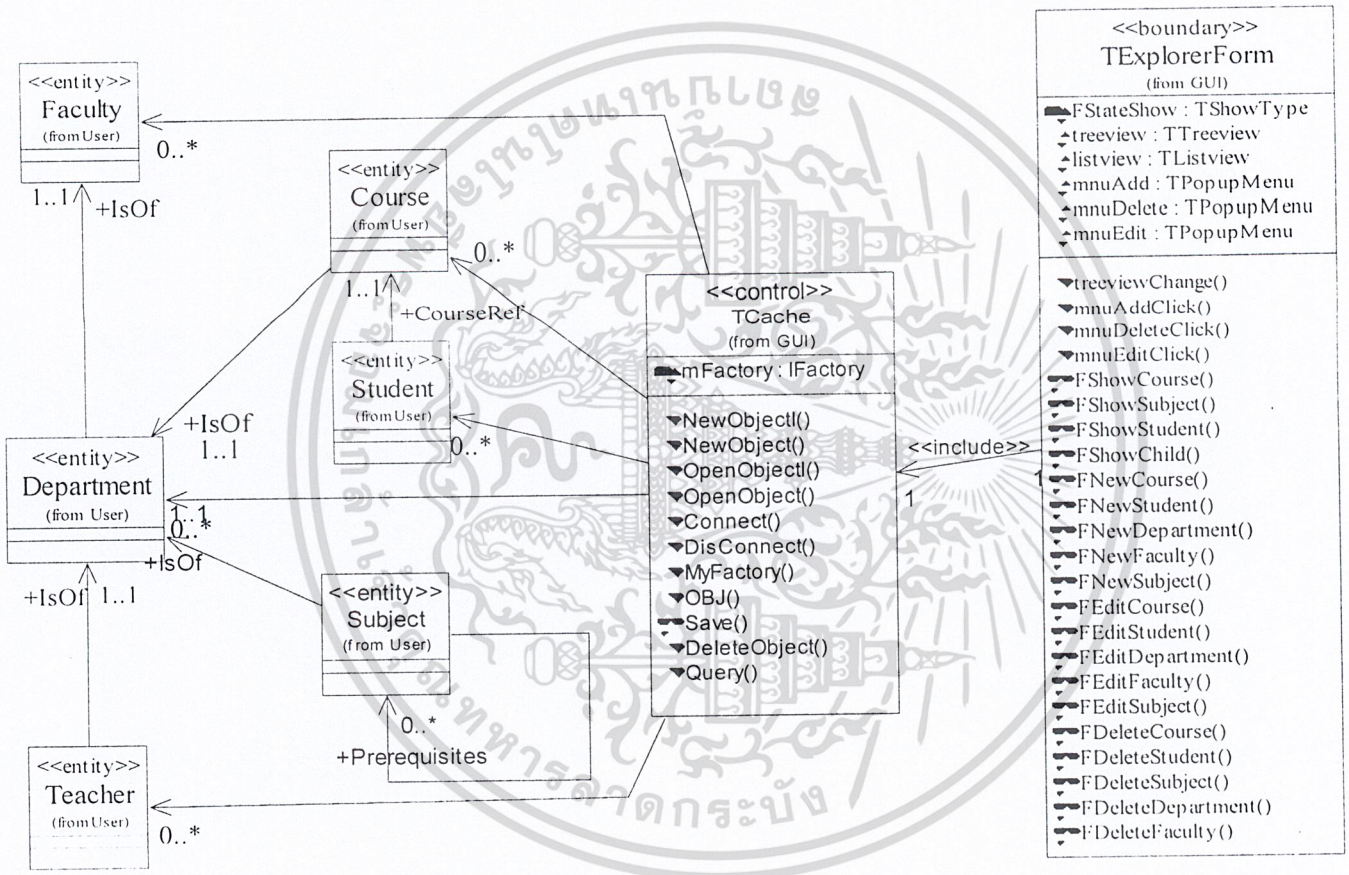
รูปที่ 6 - 21 แสดงคลาส User Interface ที่ใช้ใน IDE

Class Diagram – แสดงคลาสวินโดว์อินเทอเฟสสำหรับติดต่อผู้ใช้ บน Delphi-IDE-2



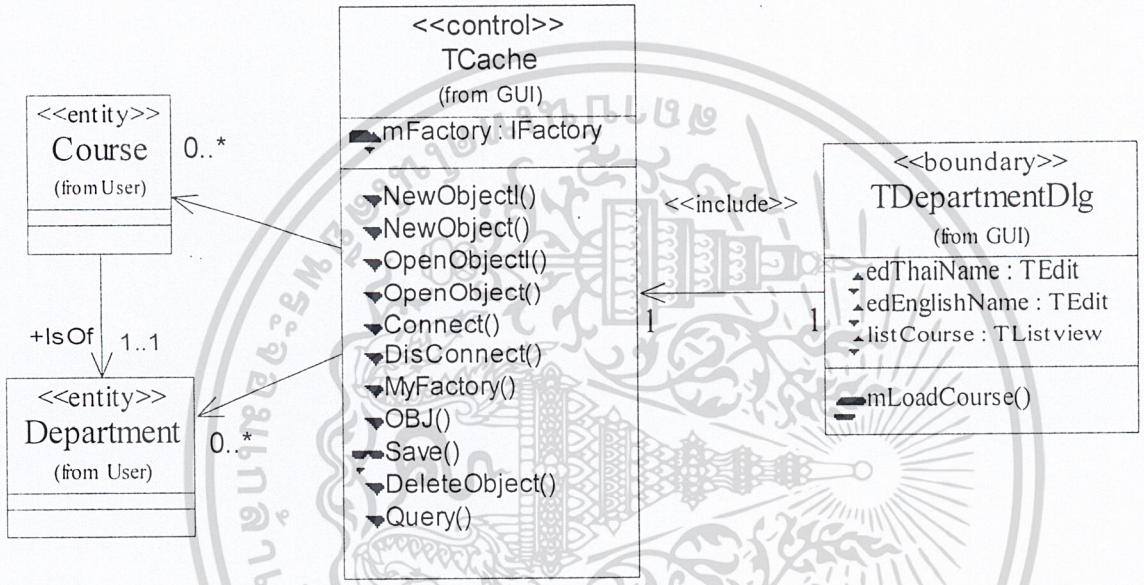
รูปที่ 6 - 22 แสดงคลาส User Interface ของการจัดการหลักสูตร

Class Diagram – แสดงคลาสวินโดว์อินเทอเฟสสำหรับติดต่อผู้ใช้ บน Delphi-IDE-3



รูปที่ 6 - 23 แสดงคลาสและการติดต่อใน User Interface ของ Explorer (Resource Viewer)

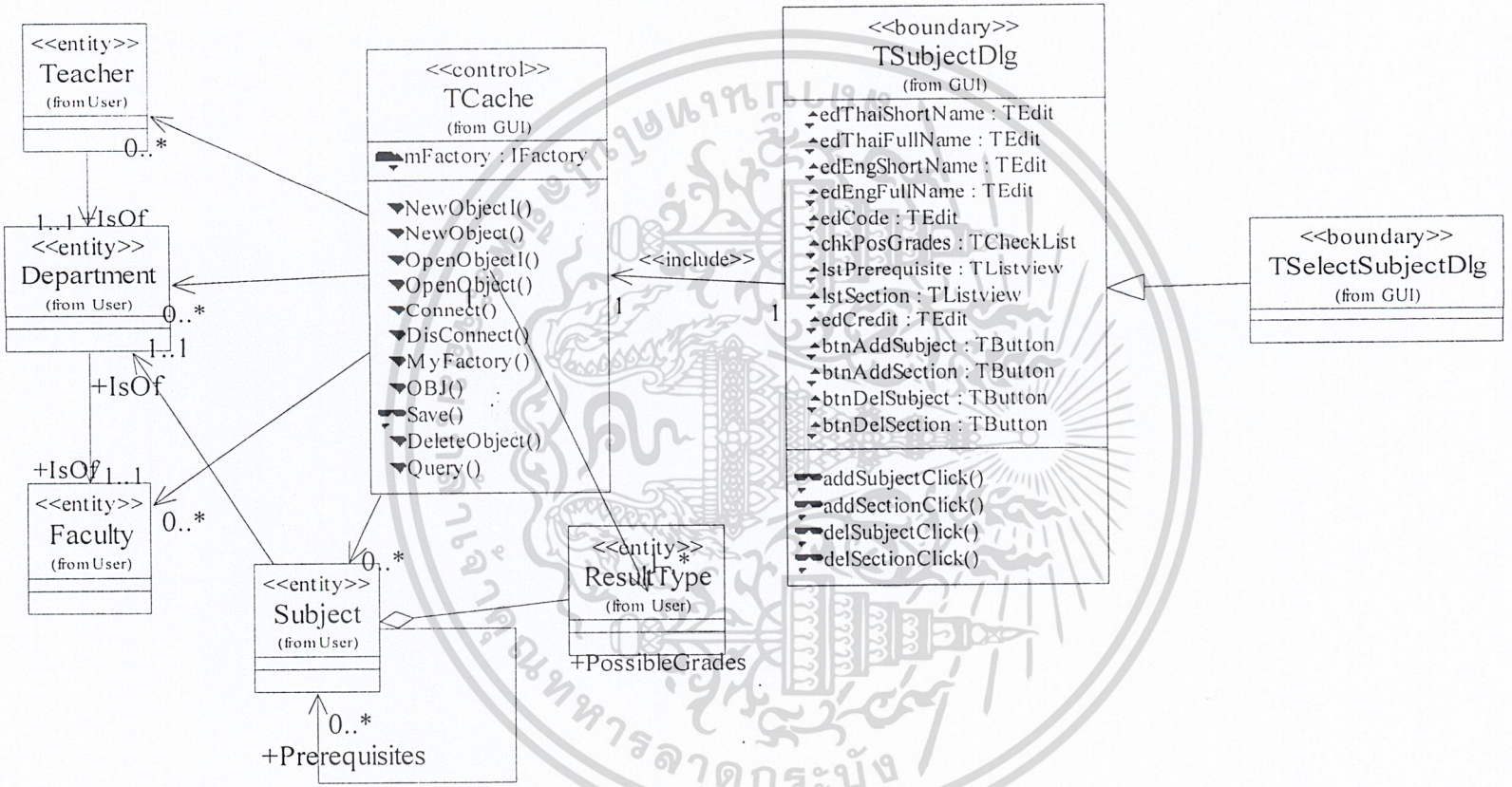
Class Diagram – แสดงคลาสวินโดว์อินเทอเฟสสำหรับติดต่อผู้ใช้ บน Delphi-IDE-4



รูปที่ 6 - 24 แสดงคลาส User Interface ของคลาสจัดการภาควิชา

ลิขสิทธิ์ของจุฬาลงกรณ์มหาวิทยาลัย ห้ามนำไปใช้ประโยชน์ด้านการค้า

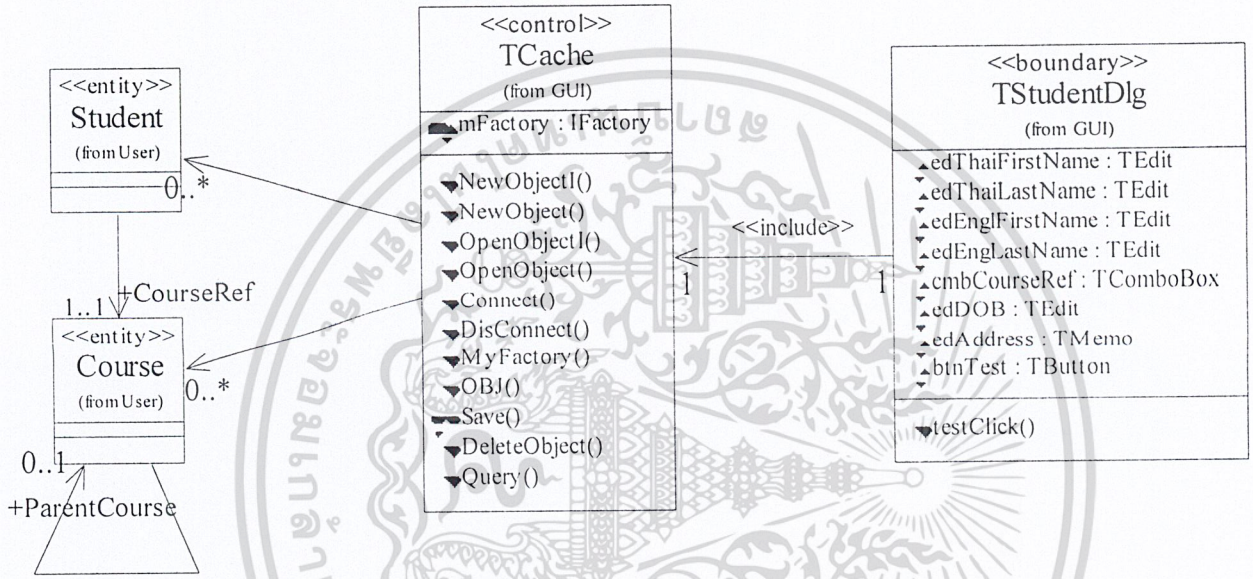
Class Diagram – แสดงคลาสวินโดว์อินเทอเฟสสำหรับติดต่อผู้ใช้ บน Delphi-IDE-5



รูปที่ 6 - 25 แสดงคลาสของคลาสจัดการรายวิชา

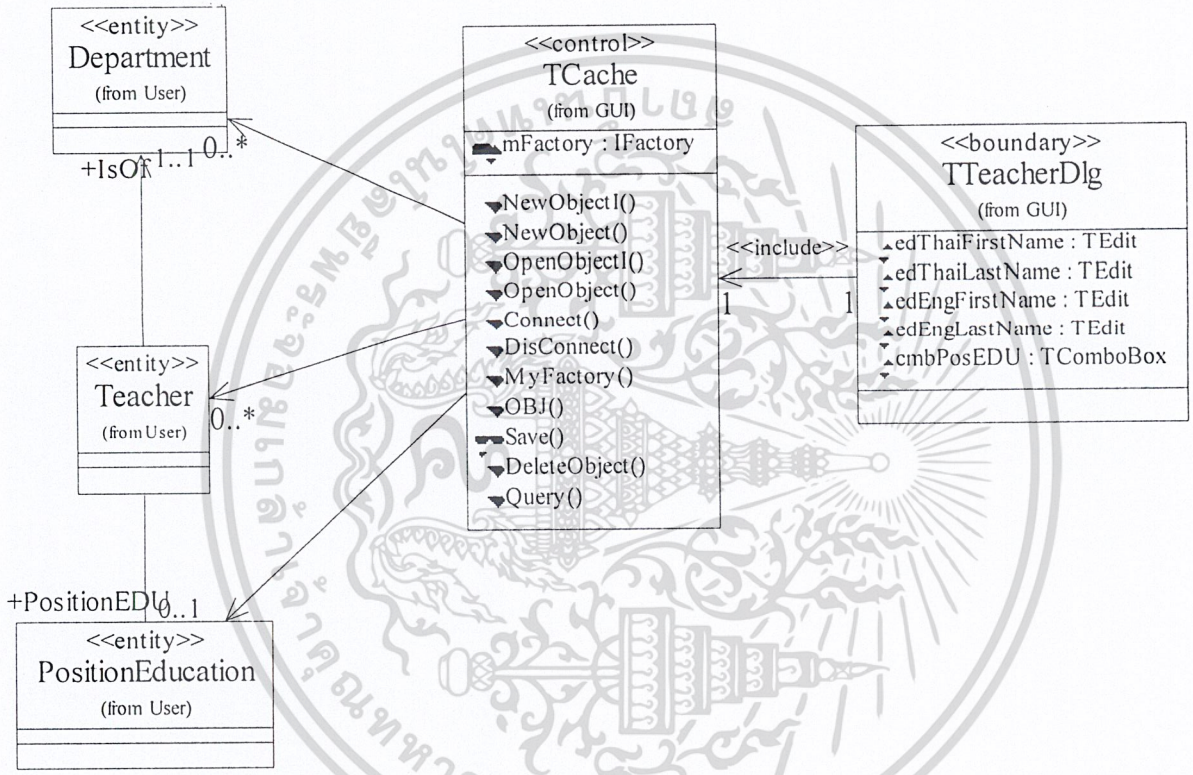
ลิขสิทธิ์ของมหาวิทยาลัยราชภัฏรำไพพรรณี ไม่อนุญาตให้ทำซ้ำ ดัดแปลง เผยแพร่ หรือนำไปใช้ประโยชน์ในทางการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น ก่อนได้รับอนุญาตเป็นลายลักษณ์อักษรจากมหาวิทยาลัย

Class Diagram – แสดงคลาสวินโดว์อินเทอเฟสสำหรับติดต่อผู้ใช้ บน Delphi-IDE-6



รูปที่ 6 – 26 แสดงคลาส User Interface ของคลาสจัดการนักศึกษา

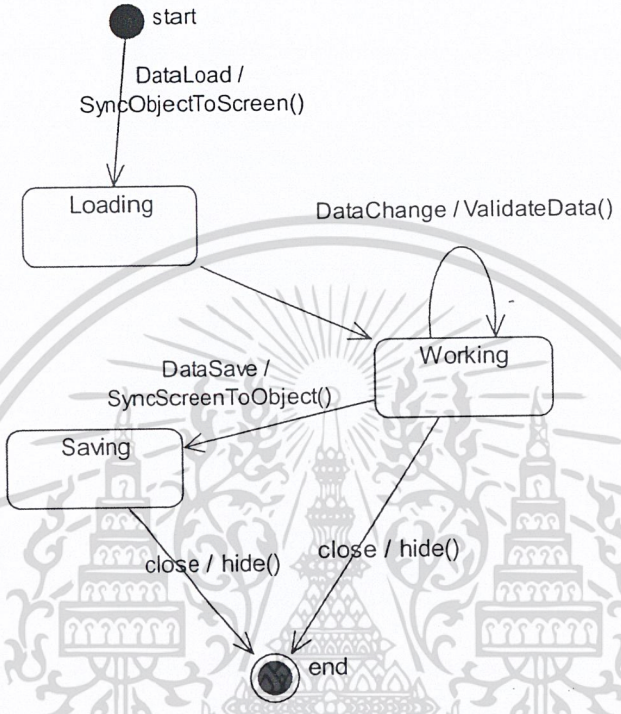
Class Diagram – แสดงคลาสวินโดว์อินเทอเฟสสำหรับติดต่อผู้ใช้ บน Delphi-IDE-7



รูปที่ 6 - 27 แสดงคลาส User Interface ของคลาสจัดการคณาจารย์

6.3.4 State-Activity Diagram

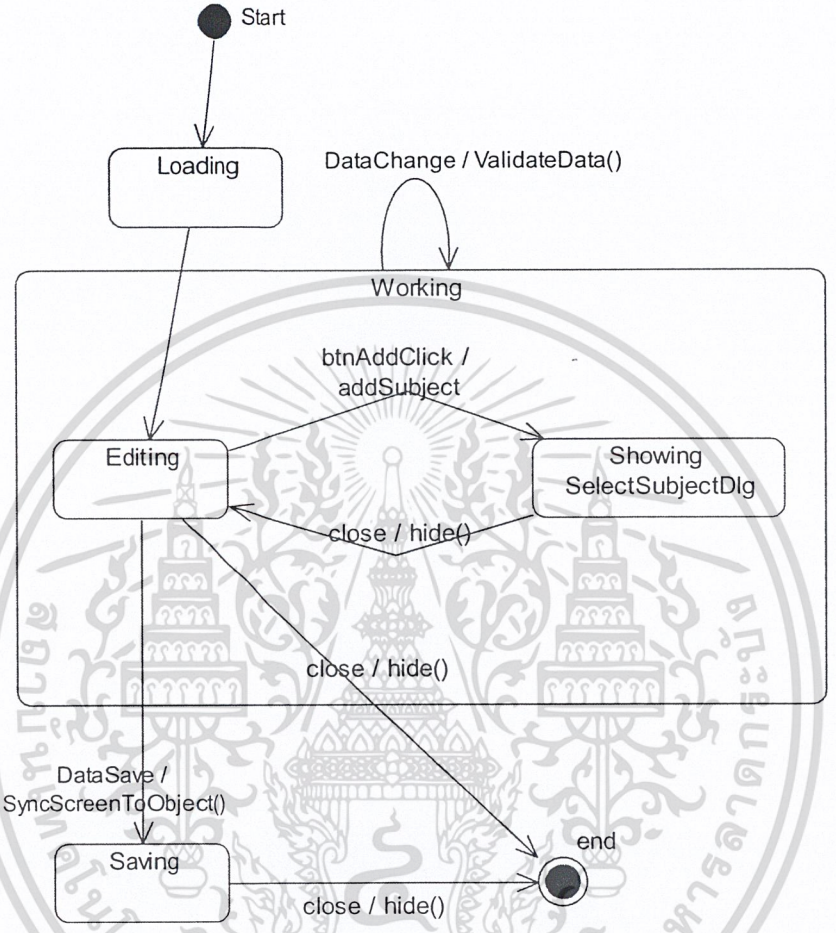
State-Activity Diagram – สถานะการทำงานภายใน TEditDialog



รูปที่ 6 - 28 แสดงสถานะการทำงานหลักของไดอะล็อกมาตรฐาน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

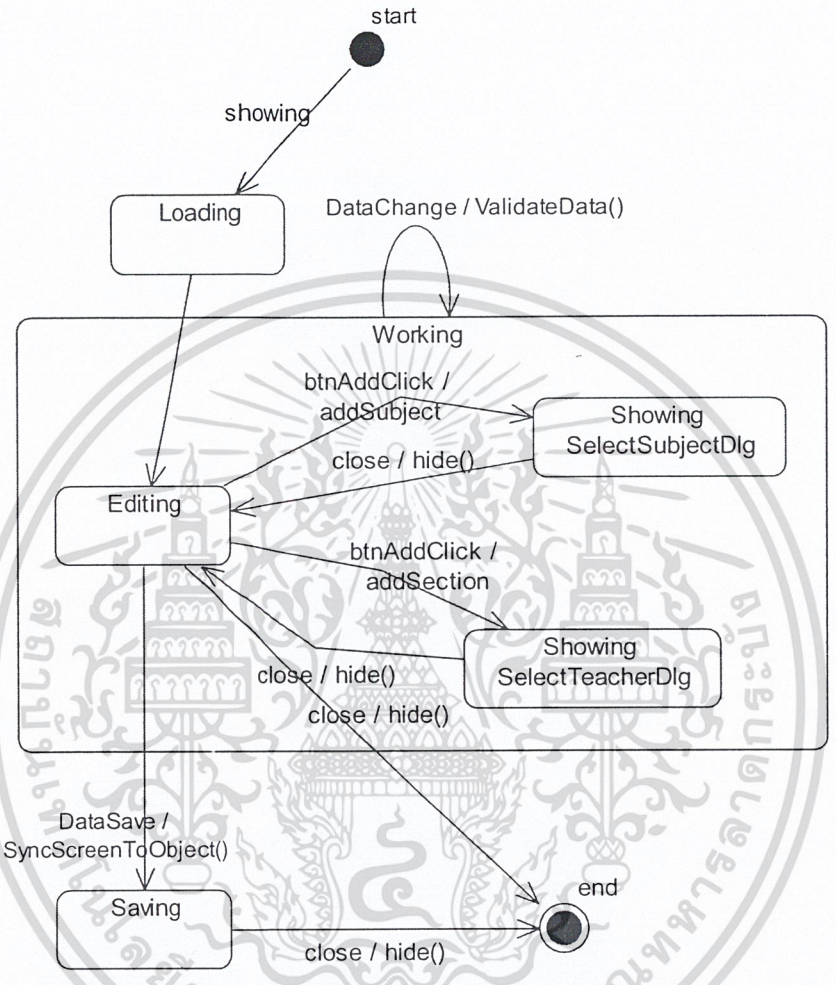
State-Activity Diagram – สถานะการทำงานภายใน TcourseDlg



รูปที่ 6 – 30 แสดงสถานะส่วนขยายการทำงานในไดอะล็อกจัดการหลักสูตร

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

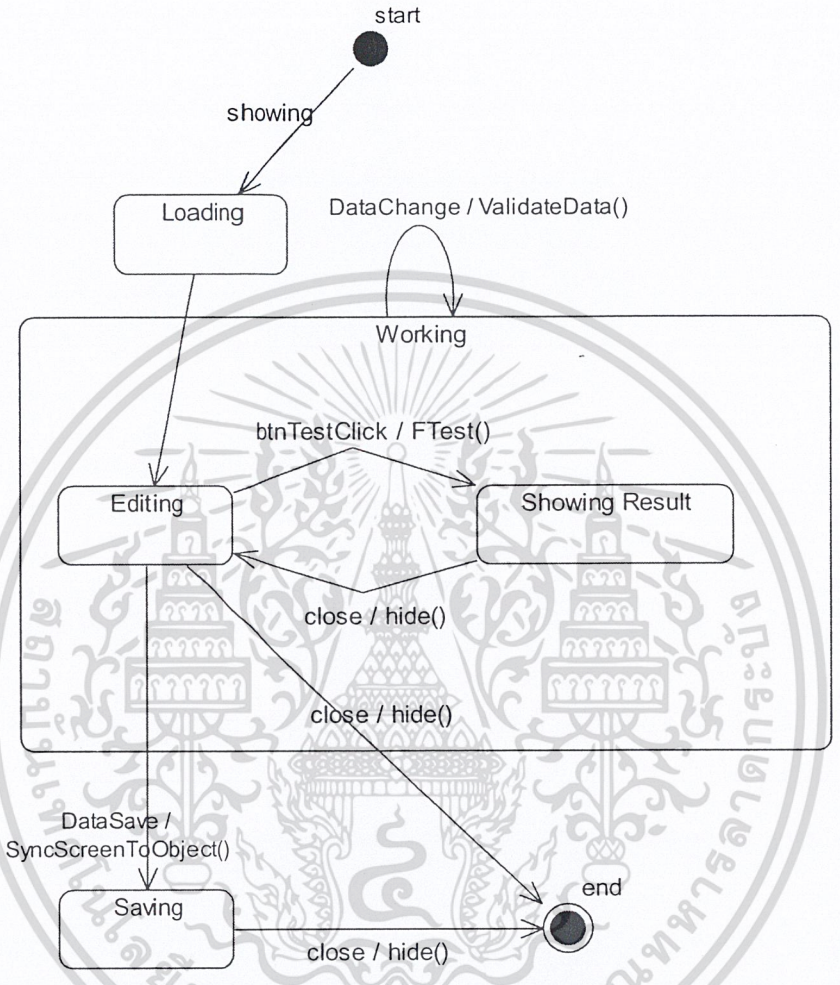
State-Activity Diagram – สถานะการทำงานภายใน TsubjectDlg



รูปที่ 6 – 31 แสดงสถานะส่วนขยายการทำงานในไดอะล็อกจัดการรายวิชา

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

State-Activity Diagram – สถานะการทำงานภายใน TStudentDlg

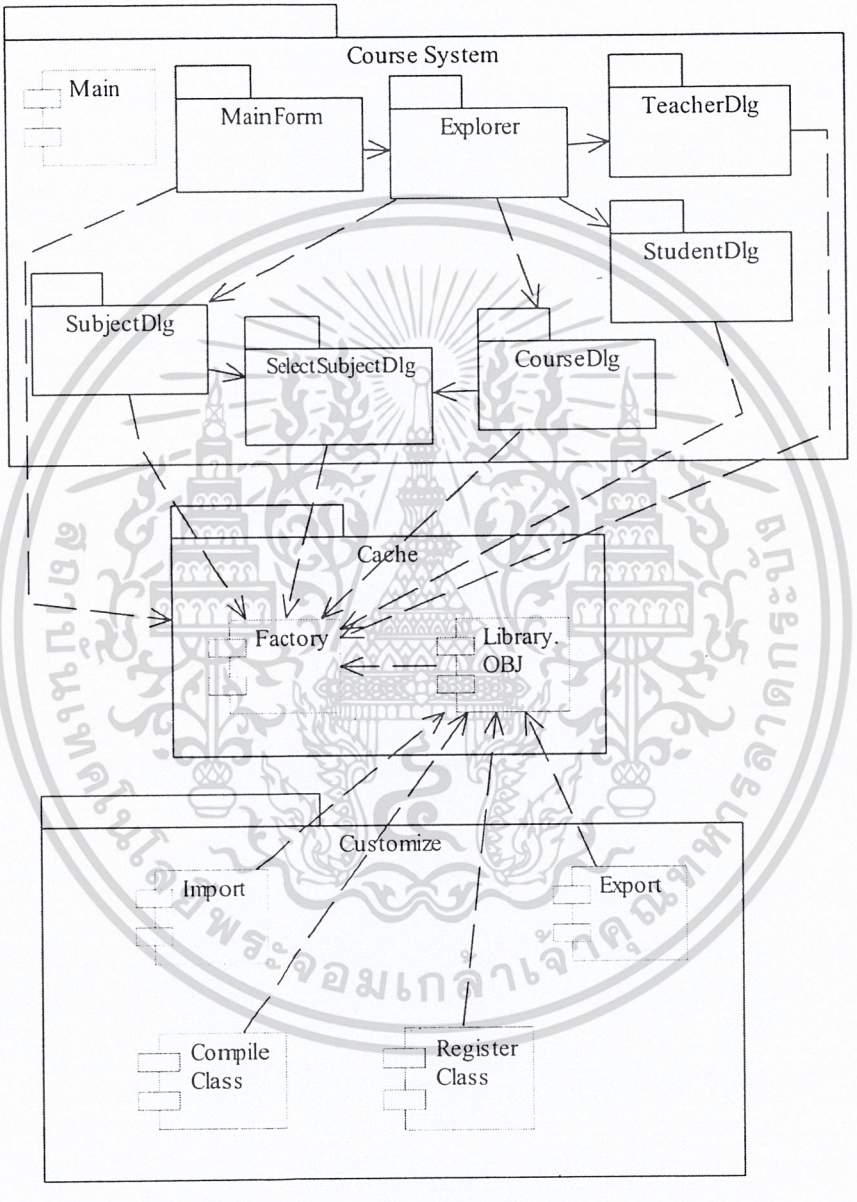


รูปที่ 6 – 32 แสดงสถานะส่วนขยายการทำงานในไดอะล็อกจัดการนักศึกษา

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

6.3.5 Component Diagram

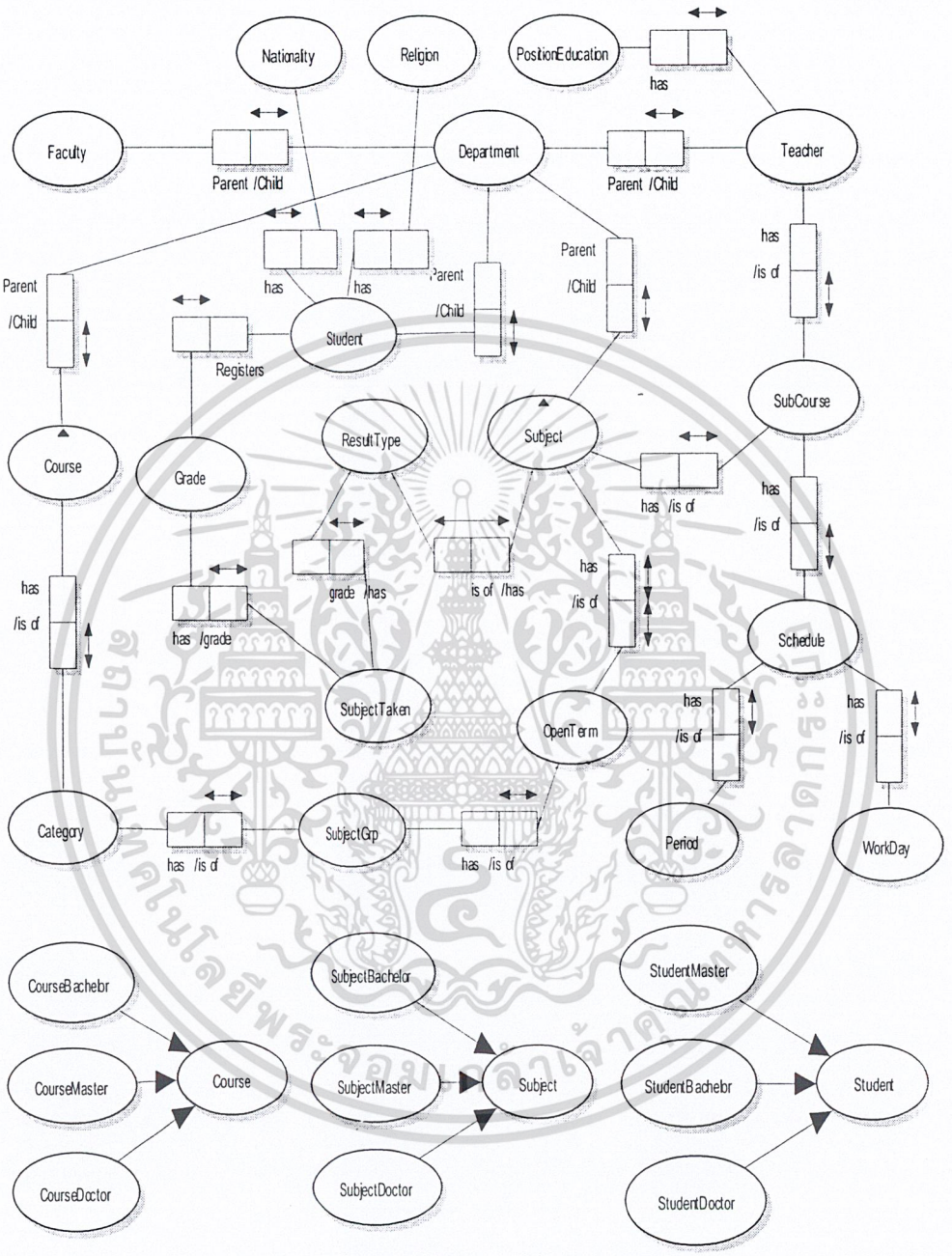
แสดงการจัดหมวดหมู่ เพื่อการพัฒนาในช่วง implimentation



รูปที่ 6 – 33 แสดงการจัดหมวดหมู่ของคอมโพเนนท์ เพื่อการนำไปอิมพลิเมนท์

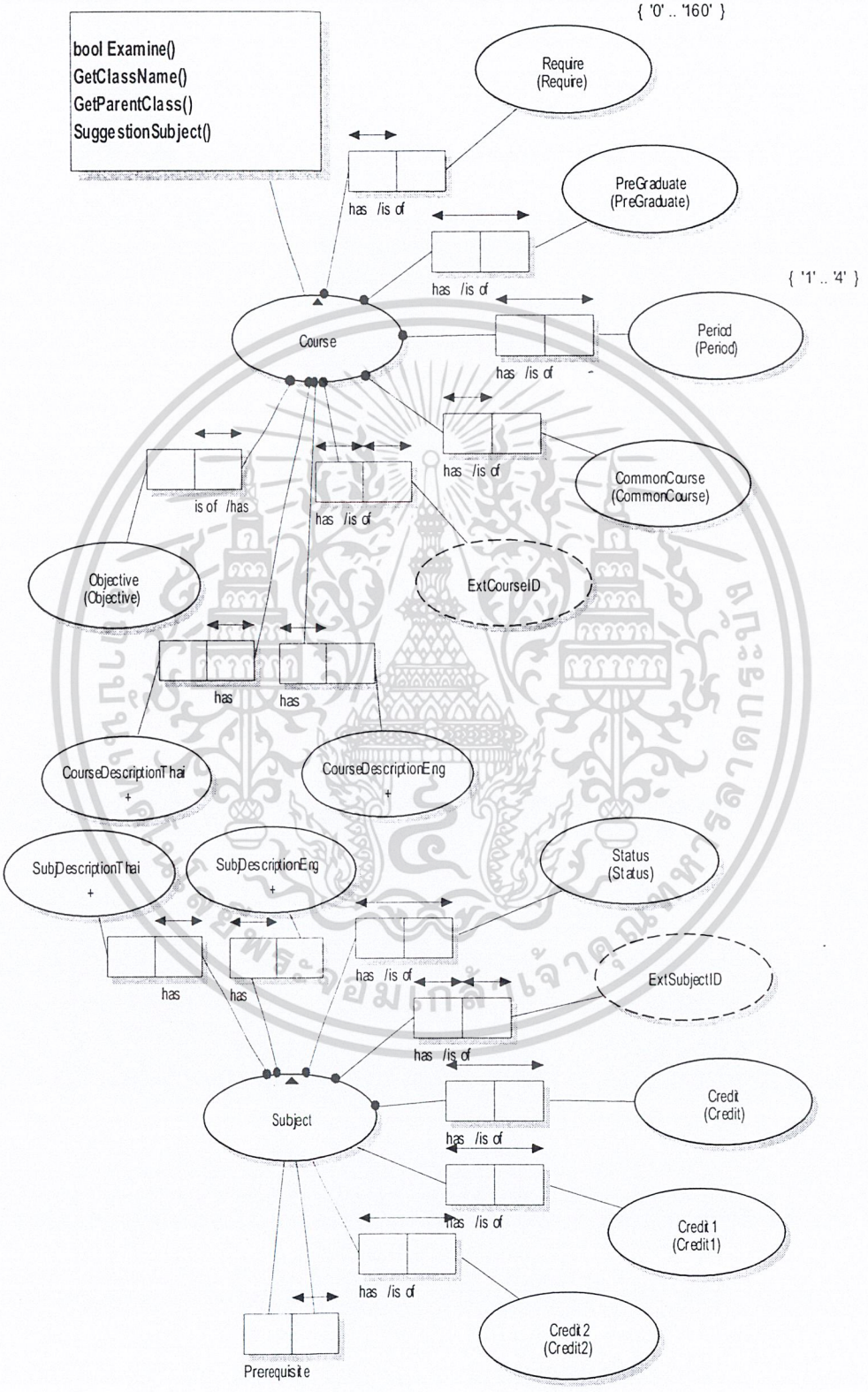
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

6.3.6 OONIAM Diagram



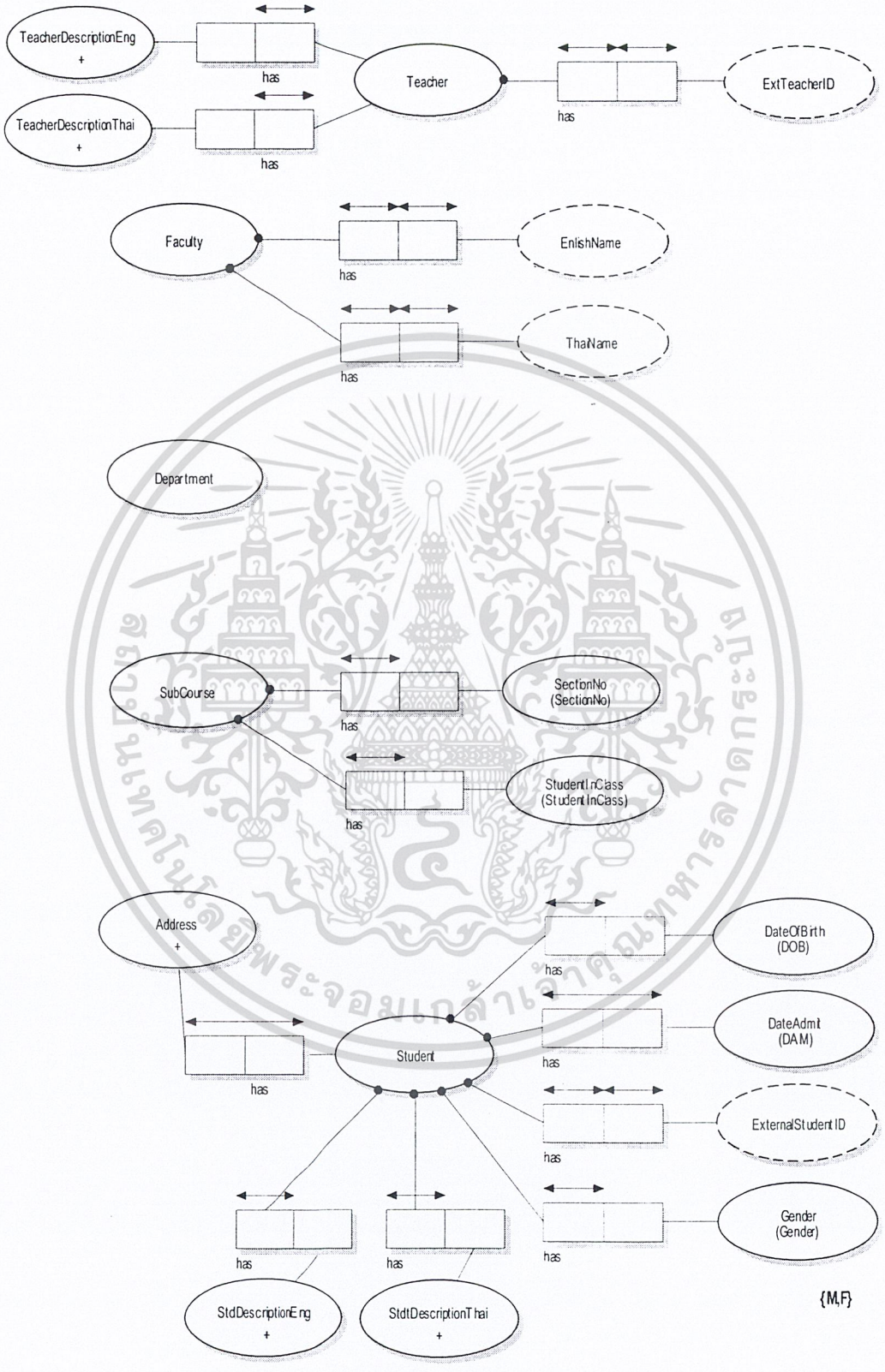
รูปที่ 6 – 34 แสดง Main Schema ในมุมมองของ OONIAM

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



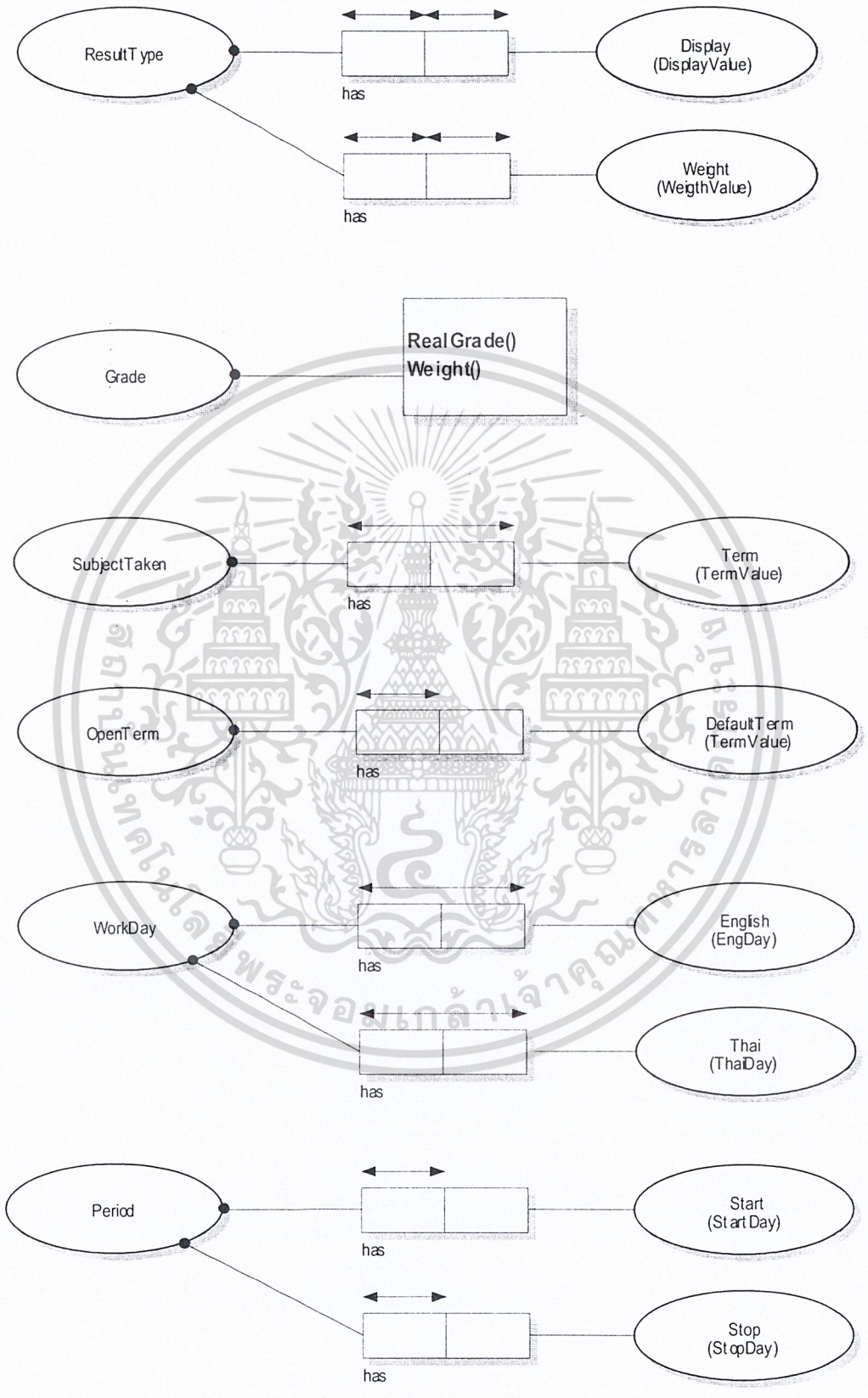
รูปที่ 6 - 35 แสดง Sub Schema level 1 ในมุมมองของ OONIAM

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



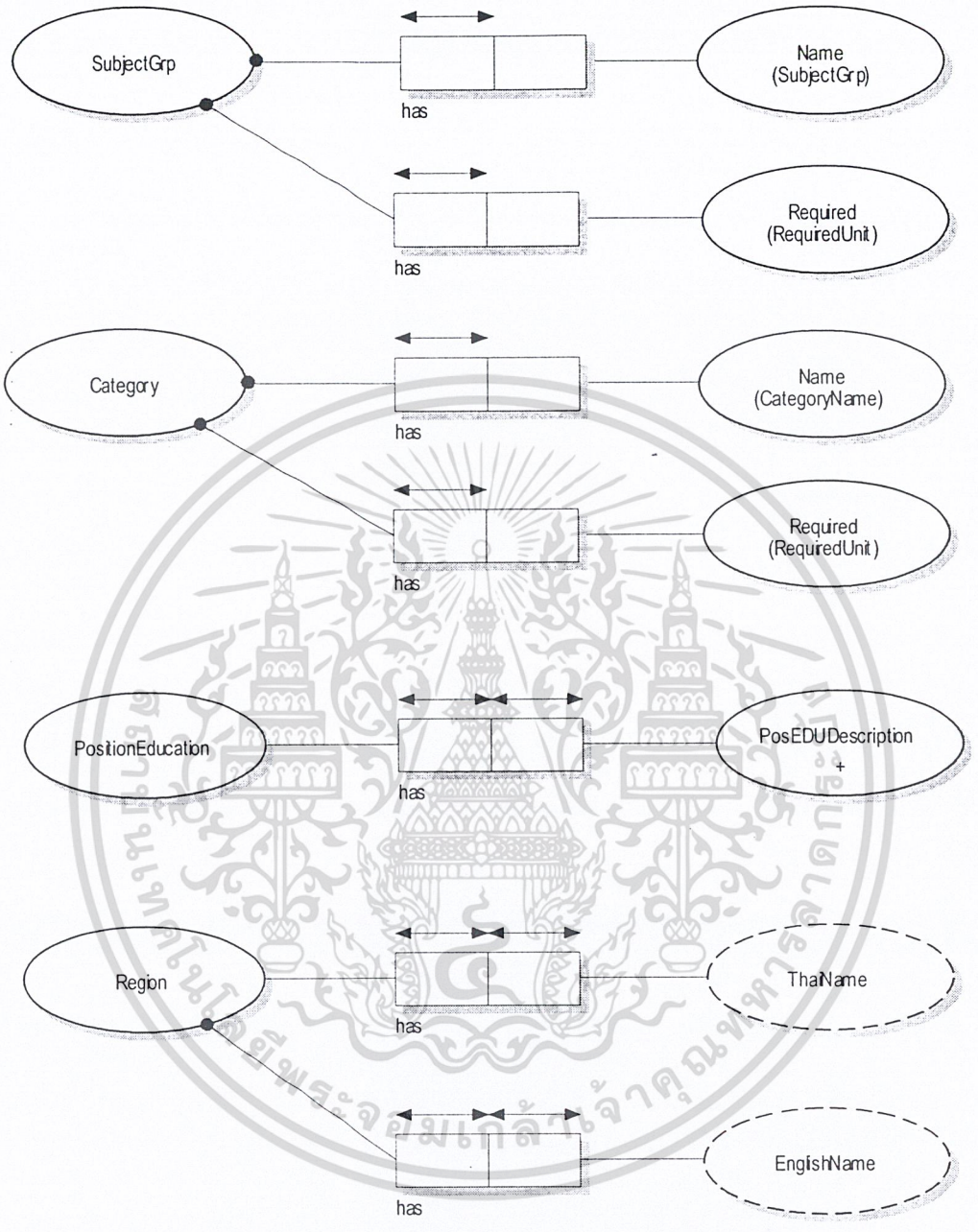
รูปที่ 6 - 35 แสดง Sub schema level 1 ในมุมมองของ OONIAM (ต่อ)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



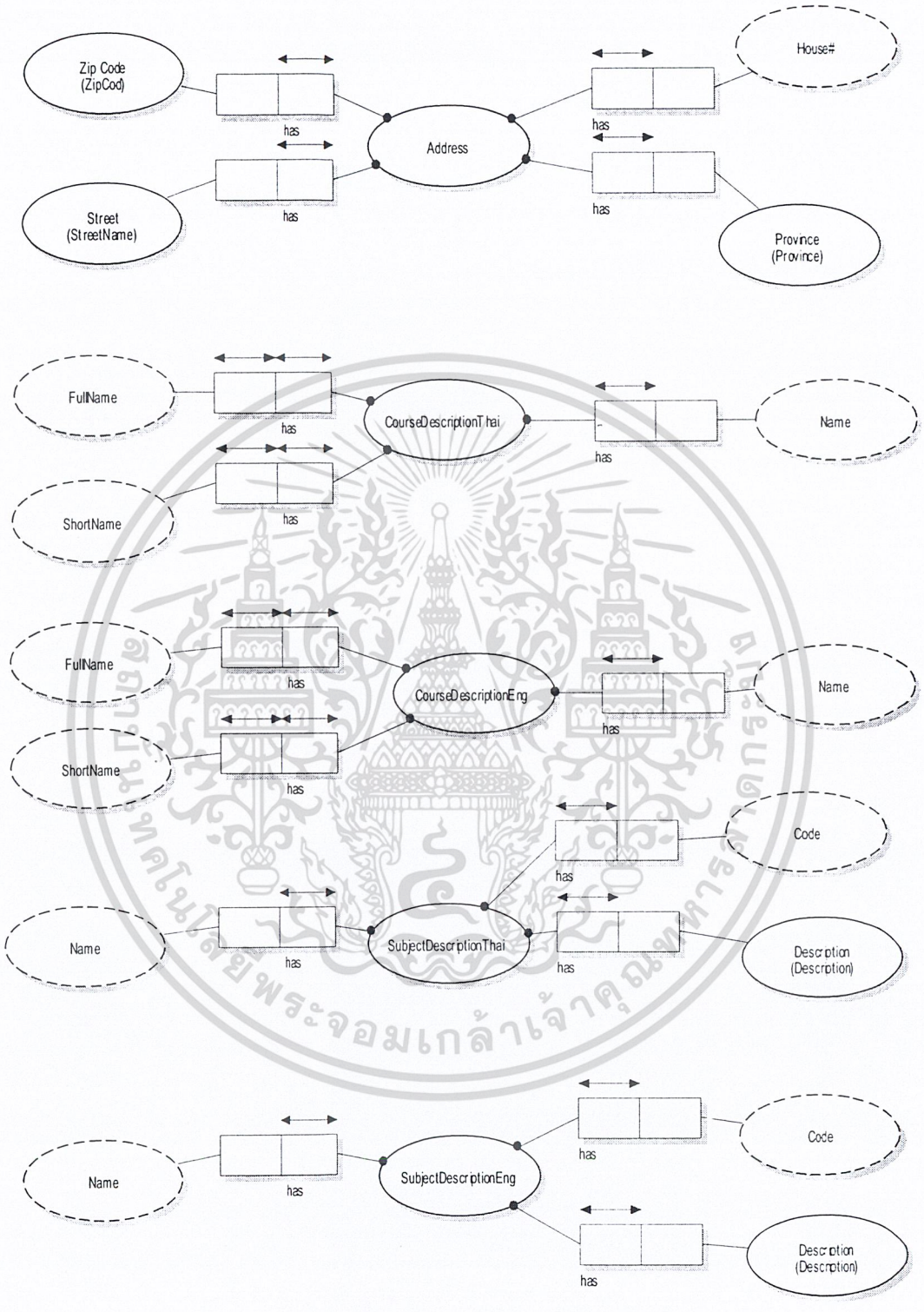
รูปที่ 6 -35 แสดง Sub schema level 1 ในมุมมองของ OONIAM (ต่อ)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



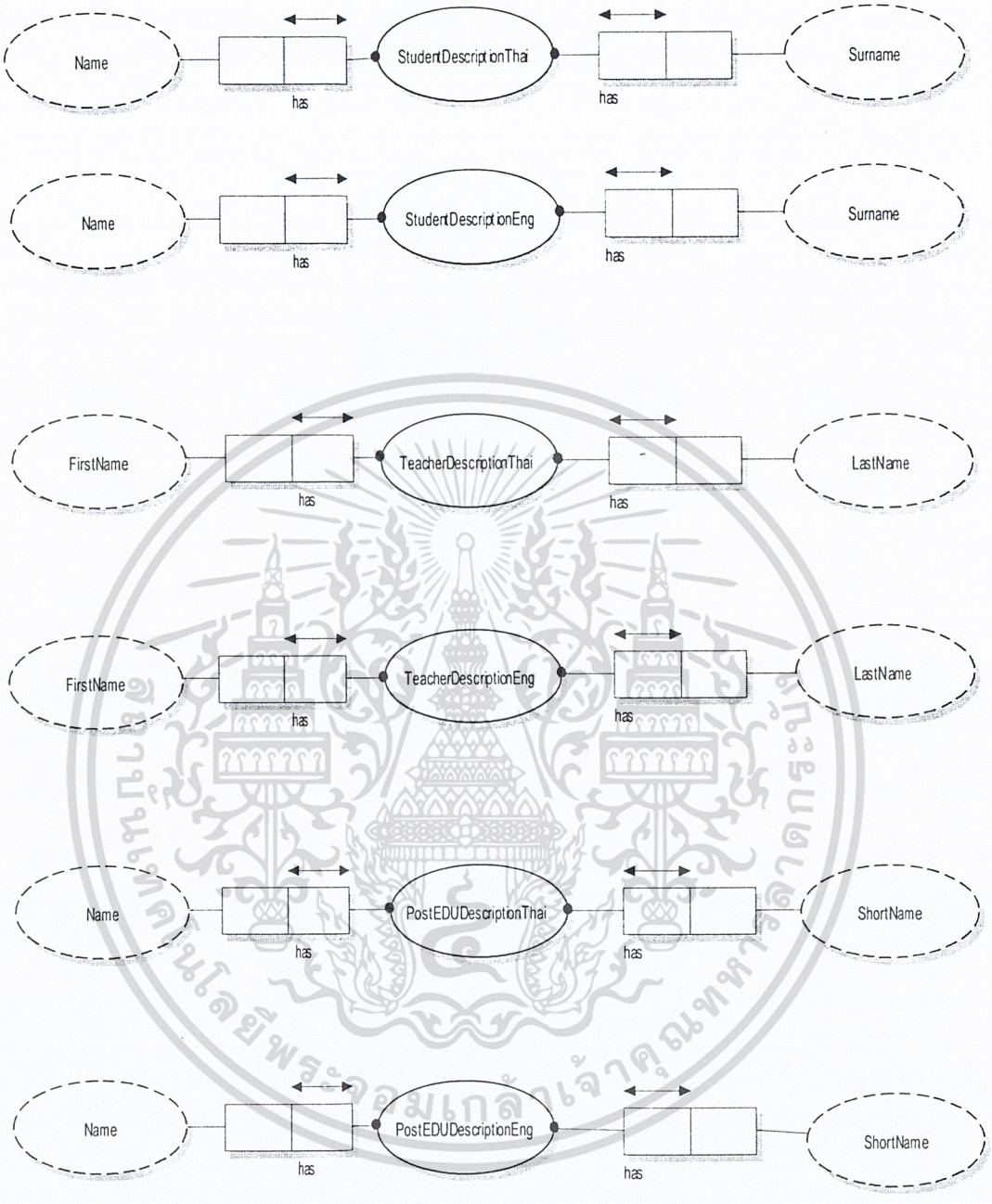
รูปที่ 6 - 35 แสดง Sub schema level 1 ในมุมมองของ OONIAM (ต่อ)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 6 - 36 แสดง Sub schema level 2 ในมุมมองของ OONIAM

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 6 - 36 แสดง Sub schema level 2 ในมุมมองของ OONIAM (ต่อ)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

6.4 การพัฒนาโปรแกรมด้วย Delphi 6.0 ร่วมกับ Caché DBMS

การติดต่อระหว่าง Delphi และ Caché สามารถทำได้ด้วยกัน 2 วิธี ซึ่งแต่ละวิธีก็มีข้อดีและข้อเสียแตกต่างกันดังนี้

การติดต่อกับ Caché โดยใช้ OLE Variant

การติดต่อด้วยวิธีนี้ เป็นวิธีที่ง่ายที่สุด ผู้พัฒนาฯ ไม่จำเป็นต้องทราบอินเทอร์เฟส (Interface) ของวัตถุ (Object) ดังนั้น ภาษาพรรณนาวัตถุ (Object Definition Language : ODL หรือ Interface Definition Language : IDL) จึงไม่จำเป็นสำหรับวิธีนี้

ตัวอย่างการติดต่อวัตถุด้วยวิธีที่หนึ่ง

```
var    mFacoty : Variant;
       mObject : Variant;
begin
       mFacoty := CoFacory.Create;
       mFactory.ConnectDlg;
       mObject := mFactory.OpenID('STUDENT','3');
       .
       .
       mObject.SYS_Close(0);
end;
```

การติดต่อด้วยวิธีนี้ ง่ายและสะดวก เหมาะสำหรับการพัฒนาเพื่อสร้าง Prototype เท่านั้น การสร้างโปรแกรมประยุกต์ที่มีประสิทธิภาพสูง จะไม่เหมาะ เพราะว่าตัวแปร Variant นั้นใช้หน่วยความจำในการเก็บถึง 64 KB โดยไม่สนใจว่าขนาดของวัตถุจริงนั้นมีขนาดเท่าใด และการแปลภาษาของตัวพัฒนา (IDE) ก็จะข้ามไวยากรณ์ส่วนนี้ไป โดยการผ่านค่าฟังก์ชั่น จะผ่านเป็นสายของอักขระ (String) ตั้งแต่หลังจุดเป็นต้นไป ทำให้เกิดความช้าในขณะรันไทม์ (Runtime) ถ้าผู้พัฒนาต้องการดึงข้อมูลครั้งละมากๆ

การติดต่อกับ Caché โดยใช้ COM (Component Object Model)

การติดต่อด้วยวิธีนี้ จำเป็นต้องมี ODL เพื่อให้ IDE รู้จักอินเทอร์เฟสของออบเจ็คต์นั้นๆ ดังนั้นจำเป็นต้องสร้างไฟล์ ODL ก่อนแล้วคอมไพล์เป็น Type Library เพื่อให้ IDE นำไปใช้ได้

ตัวอย่างไฟล์ ODL ที่สร้างจาก VC++

```
unit User_Dummy_TLB;

const
  User_DummyMajorVersion = 1;
  User_DummyMinorVersion = 0;

  LIBID_User_Dummy: TGUID = '{C96E41CE-CF3F-49FE-A939-12368B800F27}';

  DIID_CDummy: TGUID = '{B0D82FCA-1D08-4A21-B7A3-004A37E361ED}';
  CLASS_Dummy: TGUID = '{6923EAA4-DE84-47F7-BB8F-DB6DEB5C81B0}';
type

  CDummy = dispinterface;

  Dummy = CDummy;

  CDummy = dispinterface
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    ['{B0D82FCA-1D08-4A21-B7A3-004A37E361ED}']
    property DeleteTag: Integer dispid 41;
    function SYS_Close: OleVariant; dispid 2;
    function SYS_Exists(oid: OleVariant): OleVariant; dispid 47;
    function SYS_ExistsId(const id: WideString): OleVariant; dispid 48;
    function SYS_Id: WideString; dispid 25;
    procedure SYS_IncrementCount; dispid 3;
    function SYS_New(const initvalue: WideString): IDispatch; dispid 6;
    function SYS_NormalizeObject: OleVariant; dispid 7;
    function SYS_Oid: OleVariant; dispid 30;
end;
```

ตัวอย่างการติดต่อโดยวิธีที่สอง

```
uses User_Dummy_TLB;

var   mFactory : Factory;
      mObject : Dummy;

begin
      mFactory := Factory(CoFactory.Create);
      mObject := Dummy(mFactory.OpenID('STUDENT','3');
            .
            .
      mObject.SYS_Close(0);
end;
```

จากตัวอย่าง วิธีการติดต่อทั้งสองวิธีให้ผลการลัพธ์เหมือนกัน แต่มีความแตกต่างกัน สรุปได้ดังนี้

	OLE Variant	COM
1	การเข้าถึงวัตถุข้อมูลมีขนาดคงที่ 64 KB	การเข้าถึงข้อมูลมีขนาดเท่าขนาดจริงของวัตถุ
2	ใช้เวลาในการเข้าถึงข้อมูลช้า	ใช้เวลาในการเข้าถึงเร็วกว่า
3	ไม่สามารถตรวจ Syntax ได้ในขณะพัฒนา ทำให้สามารถเกิด error ได้ในขณะ runtime	สามารถตรวจสอบ Syntax ได้ในขณะ runtime
4	การพัฒนาทำได้ง่าย รวดเร็วและมีเครื่องมือช่วยมากมาย	การพัฒนาต้องทำหลายขั้นตอนและอาจจะไม่สนับสนุนได้บางภาษา
5	ไม่จำเป็นต้องทราบ ODL	จำเป็นต้องทราบ ODL ก่อนเพื่อใช้สำหรับติดต่อ
6	สนับสนุน Inheritance โดยอัตโนมัติ	ไม่สนับสนุน Inheritance เนื่องจากการอ้างอิงใช้การอ้างอิงด้วย GUID แทนการอ้างอิงด้วย Class และ Super Class
7	ไม่จำเป็นต้องมีความรู้ด้าน Distributed Application	จำเป็นต้องมีความรู้ด้าน Distributed Application

ตารางที่ 6 - 2 แสดงข้อดี-ข้อเสียของการติดต่อ

ขั้นตอนการสร้าง Type Library เพื่อใช้ใน Delphi

จาก Caché Architecture เลือกคลาสที่ต้องการแล้ว Export จะได้ไฟล์ xxx.ODL

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

นำไฟล์ที่ได้ Import เข้า Delphi จาก Project -> Import TypeLibrary

คลิ้ก <<Create Unit>>

จากนั้นจะได้ไฟล์ xxx_TLB.PAS

6.5 การเข้าถึงข้อมูล

6.5.1 การเข้าถึงข้อมูลระดับวัตถุ Object

เมื่อสร้างวัตถุจาก Caché แล้ว ผู้พัฒนาสามารถเรียกแอตทริบิวท์และเมทธอดของวัตถุได้ดังตัวอย่าง

```
uses User_Student_TLB;

var   mFactory : Factory;
      mObject : Student;
      fName : string;
      fDOB : TDateTime;
      fAge : Single;
begin
      mFactory := Factory(CoFactory.Create);
      mObject := Student(mFactory.OpenID('STUDENT','3');
      fName := mObject.FirstName;    // access attribute
      fDOB := mObject.DOB;           // access attribute
      fAge := mObject.GetAge();      // invoke method
      mObject.SYS_Close(0);
end;
```

6.5.2 การเข้าถึงสมาชิกของข้อมูลที่เป็น Array Collection

ผู้พัฒนาไม่สามารถเข้าถึงข้อมูลที่เป็นคอลเล็คชั่นได้โดยตรงเหมือนแอตทริบิวท์ทั่วไป เราจำเป็นต้องแปลง (Casting) ให้เป็นข้อมูลชนิด SYS_ArrayOfObjects (อยู่ในยูนิต ArrayOfObjects_TLB.pas) ก่อน แล้วเรียกใช้เมทธอดของ SYS_ArrayOfObjects ในการเข้าถึงข้อมูล ดังตัวอย่างแสดงการอ่านข้อมูลและการเขียนข้อมูลกลับ

```
var   posGrds : SYS_ArrayOfObjects;
      key : WideString;

begin
  posGrds := SYS_ArrayOfObjects( IUnknown(mObject.PossibleGrades) );
  key := '';
  posgrd := posGrds.GetNext(key);
      while Length(key) > 0 do
      begin
            Combo1.Items.Add (posgrd.Display);  // read
            posgrds.SetAt (grd,grd.Display);    // write
            posgrd := posGrds.GetNext(key);     // move next
      end;
  posgrds.SYS_Close;
end;
```

6.5.3 การเข้าถึงสมาชิกของข้อมูลที่เป็นคลาสและเป็น Embeded Class

ผู้พัฒนาสามารถเข้าถึงอิลิเมนต์ของแอตทริบิวท์แต่ละตัวได้โดยการเรียกชื่อแอตทริบิวท์นั้นแล้วตามด้วยจุดแล้วตามด้วยอิลิเมนต์หรือเมทธอดที่ต้องการ (Cascade Dot Syntax) ดังตัวอย่าง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
var   mStudent : Student;
      fHouseNo : string;
      fStreet : string;
      fZipCode : string;
      fCity : string;
begin
      fHouseNo := mStudent.Address.HouseNo;      // access element
      fStreet := mStudent.Address.Street;
      fZipCode := mStudent.Address.ZipCode;
      fCity := mStudent.Address.GetCityFromZipCode(); //invoke method
end;
```

6.5.4 การเข้าถึงสมาชิกของข้อมูลที่เป็นคลาสและเป็น Referenced Class

ในการพัฒนาโปรแกรมเชิงวัตถุ สิ่งที่ต้องระวังคือการอ้างอิงถึงวัตถุที่ยังไม่ถูกสร้าง (Instantiate) เพราะ จะทำให้เกิด Access Violation เว้นแต่วัตถุข้อมูลจาก Caché เท่านั้นที่ Caché DBMS รับประกันการเข้าถึงโดยจะสร้างวัตถุทุกตัวที่อาจจะถูกอ้างถึงเพื่อป้องกันปัญหานี้ ดังนั้นผู้พัฒนาสามารถเข้าถึง Referenced Object ได้ทันทีโดยไม่จำเป็นต้อง Instantiate เอง (แต่จำเป็นต้องมีการคาสติ้งเสมอ) ผู้พัฒนาสามารถเข้าถึงอิลิเมนต์ของแอตทริบิวท์แต่ละตัวได้โดยการเรียกชื่อแอตทริบิวท์นั้นแล้วตามด้วยจุดแล้วตามด้วยอิลิเมนต์หรือเมธอดที่ต้องการดังตัวอย่าง

```
var   mStudent : Student;
      mDepartment : Department;
      mFaculty : Faculty;
      fName : string;
      fDepartment : string;
      fFaculty : string;
begin
      fName := mStudent.FirstName;
      fDepartment := Department(mStudent.IsOf).Name;
      fFaculty := Faculty(Department(mStudent.IsOf).IsOf).Name;
end;
```

6.5.5 การเข้าถึงข้อมูลด้วยคิวรี (Access with SQL)

Caché DBMS มีความสามารถในการเข้าถึงข้อมูลด้วย SQL (Structural Query Language) ทำให้มีทางเลือกหนึ่งในการเข้าถึงข้อมูลครั้งละมากๆ ได้ โดยใช้คลาสที่ชื่อ ResultSet (อยู่ใน Caché Object.pas) โดยระบุชื่อคลาสและชื่อคิวรีของคลาสนั้นที่ต้องการ ดังตัวอย่าง

```
var   rs : ResultSet;

begin
      rs := mFactory.ResultSet('STUDENT','DETAILS');
      rs.Execute();     // open recordset
      while not rs.Next() do  // if not EOF
      begin
            writeLn(rs.GetData(1),rs.GetData(2));
      end;
      rs.Close;
end;
```

ด้วยความสามารถนี้ ผู้พัฒนายังสามารถ Join ระหว่างคลาสได้ โดยใช้ ID ดังตัวอย่างภาษา SQL ต่อไปนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
/* in Caché  */
SQL = SELECT COURSE.THAI_NAME, COURSE.THAI_FULLNAME,;
      DEPARTMENT.THAINAME;
      FROM COURSE, DEPARTMENT;
      WHERE COURSE.ISOF = DEPARTMENT.ID;
      ORDER BY DEPARTMENT.THAINAME;
```

จากคำสั่งข้างต้นเป็นการ join ระหว่างคลาส Course และ Department โดยมี Course.IsOf เป็น foreign key และ Department.ID เป็น Primary key (ในมุมมองของฐานข้อมูลเชิงสัมพันธ์) นอกจากนี้เรายังสามารถผ่านพารามิเตอร์เพื่อเลือกเงื่อนไขในคำสั่ง SQL ได้ด้วยดังตัวอย่าง

```
/* in Caché  */
DetailIsOf =
      SELECT COURSE.THAI_NAME, COURSE.THAI_FULLNAME,;
      DEPARTMENT.THAINAME;
      FROM COURSE, DEPARTMENT;
      WHERE COURSE.ISOF = DEPARTMENT.ID;
      AND DEPARTMENT.ID =: department;
      AND COURSE.COURSELEVEL =: level;
      ORDER BY DEPARTMENT.THAINAME;

// in Delphi
var   rs : ResultSet;
      id : string;
      level : integer;

begin
      rs := mFactory.ResultSet('STUDENT','DetailIsOf');
      id := '3';
      level := 1;
      rs.Execute(id,IntToStr(level));
      while not es
      begin
            writeLn(rs.GetData(1),rs.GetData(2));
      end;
      rs.Close;
end;
```

จากโค้ดตัวอย่างข้างต้น เป็นการผ่านพารามิเตอร์ชื่อ Department และ level เข้าไปยังคำสั่ง SQL สังเกตุว่า จะมีเครื่องหมาย ':' อยู่หน้า พารามิเตอร์เสมอ

6.6 การสร้างระบบ Customize โดยใช้ Library ของ Caché

การสร้างระบบที่สามารถเปลี่ยนแปลงได้ โดยการเปลี่ยนแปลงพฤติกรรมของคลาสผ่านการสืบทอดคุณสมบัติ (Inheritance) ในระบบงานนี้ เราจะเน้นไปที่การเปลี่ยนแปลงเพียงพฤติกรรมเท่านั้น โดยไม่มีการเพิ่มคุณสมบัติและเมทธอด และการแก้พฤติกรรมของคลาส จำเป็นต้องมีการคอมไพล์ใหม่ทุกครั้งก่อนทำการเรียกใช้

ดังนั้นผู้พัฒนาระบบงาน Customize จึงจำเป็นต้องศึกษากลไกการทำงานของตัว Caché DBMS ด้วยว่ามีวิธีการในการจัดการสิ่งเหล่านี้อย่างไร

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

6.6.1 การนำเข้าคลาส CDL

เป็นการนำคลาสในไฟล์ชนิด CDL จากภายนอกเข้ามาใน namespace ที่ต้องการ ในขั้นตอนนี้ยังไม่มีการคอมไพล์ โดยมีการทำงานดังตัวอย่าง

```
var   cdl: Variant;
begin
      cdl := mFactory.New('%ClassDefinition');
      cdl.ImportCDL('C:\project\cdl\myclass.cdl','');
end;
```

6.6.2 การส่งออกคลาส CDL,ODL,JAVA

เป็นการนำคลาสที่อยู่ใน Namespace ออกมาในรูปแบบของ CDL,ODL หรือ JAVA โดยมีวิธีการดังตัวอย่าง

```
var   cdl: Variant;
begin
      cdl := mFactory.New('%ClassDefinition');
      cdl.ExportCDL('myclass.cdl','C:\project\cdl\myclass.cdl');
      cdl.ExportODL('myclass.cdl','C:\project\cdl\myclass.odl');
      cdl.ExportJava('myclass.cdl','C:\project\cdl\myclass.java');
end;
```

6.6.3 การคอมไพล์คลาส

เป็นการคอมไพล์คลาสที่นำเข้ามาหรือเขียนขึ้นมาใหม่ โดยถ้าคลาสนั้นคอมไพล์ผ่าน จะถูกลงทะเบียนใน Namespace และจะนำไปใช้ได้ ดังตัวอย่าง

```
var   cdl: Variant;
      status : integer;
begin
      cdl := mFactory.New('%ClassDefinition');
      Status:= cdl.CompileClasses('user.myclass','');
end;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

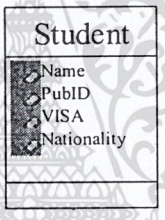
บทที่ 7
สรุปผลการวิจัยและข้อเสนอแนะ

7.1 สรุปผลการวิจัย

ในงานวิจัยได้ศึกษาการออกแบบระบบงานด้วย UML ซึ่งเป็นโมเดลสำหรับออกแบบโปรแกรมเชิงวัตถุและฐานข้อมูลเชิงสัมพันธ์ โดยยังไม่เหมาะสมกับการออกแบบฐานข้อมูลเชิงวัตถุ ซึ่งโมเดลของ UML ที่อาจจะเป็นข้อบกพร่องในการสื่อความหมายทางสัญลักษณ์ เพื่อนำไปอิมพลิเมนท์จริงนั้น ยังขาดคุณลักษณะบางอย่างอยู่ในการแมป (mapping) อยู่ ดังจะได้พบปัญหาดังต่อไปนี้

- ปัญหาความสัมพันธ์ระหว่างแอตทริบิวท์ในคลาสและระหว่างแอตทริบิวท์กับคลาส

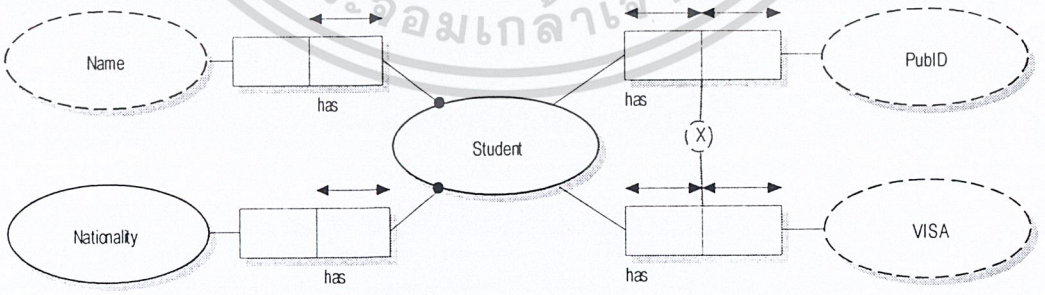
ปัญหานี้เกิดจากผู้อิมพลิเมนต์ไม่ทราบความสัมพันธ์ระหว่างแอตทริบิวท์สองตัวหรือมากกว่านั้นว่ามีความสัมพันธ์ หรือมีกฎในการป้อนอินพุตอย่างไร ดังตัวอย่างต่อไปนี้



รูปที่ 7 - 1 แสดงตัวอย่างปัญหาความสัมพันธ์ระหว่าง แอตทริบิวต์ในคลาส

จากตัวอย่าง ถ้ากำหนดว่านักศึกษาที่เป็นคนไทย ให้กรอกหมายเลขบัตรประชาชน ไม่เช่นนั้นให้กรอกหมายเลข VISA ซึ่งในโมเดลนี้ ไม่สามารถบอกได้

ทางแก้ไขสามารถทำได้โดยใช้ OONIAM

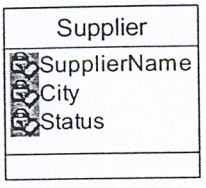


รูปที่ 7 - 2 แสดงตัวอย่างการแก้ปัญหาความสัมพันธ์ระหว่างแอตทริบิวต์ในคลาสด้วย OONIAM

- คลาสไดอะแกรมไม่ได้นำเสนอ Function Dependency (FD)

ซึ่งอาจทำให้เกิดค่าของแอตทริบิวท์ที่ซ้ำซ้อนได้ ดังตัวอย่างต่อไปนี้

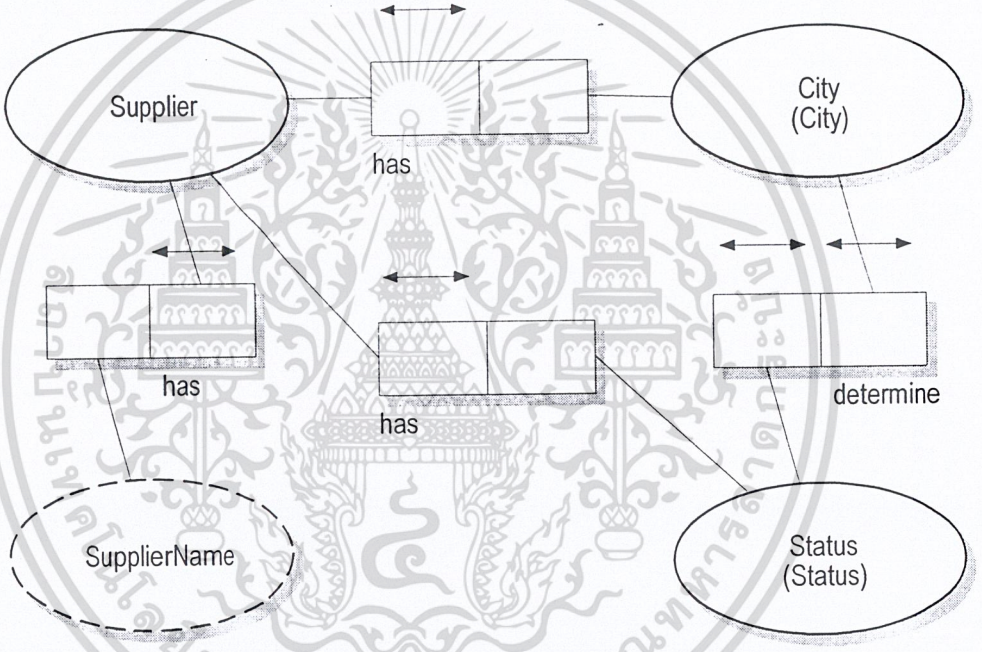
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 7 - 3 แสดงตัวอย่างปัญหา FD ระหว่างแอตทริบิวต์ในคลาส

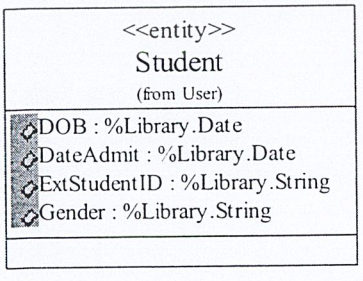
จากตัวอย่างจะเห็นว่าค่าของ city จะ determine state เสมอ เนื่องจากเราไม่ทราบความสัมพันธ์ของแอตทริบิวท์ทั้งสองในคลาส

ทางแก้ไขสามารถแสดง FD ใน OONIAM ได้



รูปที่ 7 - 4 แสดงตัวอย่างการแก้ปัญหา FD ระหว่างแอตทริบิวต์ในคลาส

- UML กำหนดกฎข้อบังคับ (Constraint) ของข้อมูลได้ไม่ชัดเจนนัก

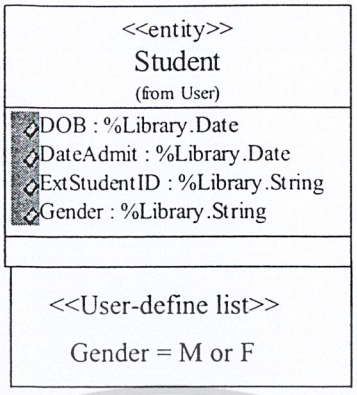


รูปที่ 7 - 5 แสดงตัวอย่างปัญหาการกำหนดกฎข้อบังคับใน UML

วิธีแก้ไข

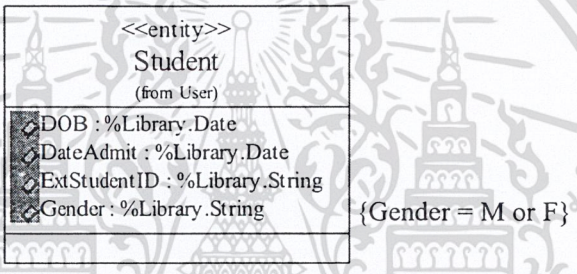
วิธีที่ 1 สามารถอธิบายกฎข้อบังคับไว้ในส่วนของ User-define list ได้

เอกสารนี้เป็นเอกสาร... ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



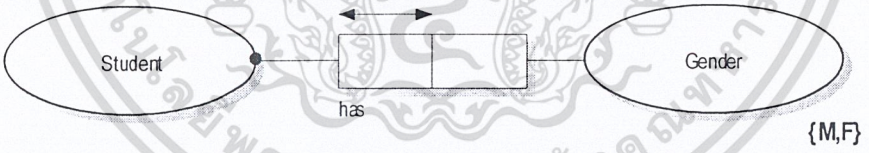
รูปที่ 7 - 6 แสดงตัวอย่างการแก้ปัญหาการกำหนดกฎข้อบังคับใน UML โดยใช้ User-define list

วิธีที่ 2 สามารถอธิบายกฎข้อบังคับไว้ในวงเล็บปีกกา { } แล้วเขียนกำกับไว้ข้าง ๆ คลาสนั้น



รูปที่ 7 - 7 แสดงตัวอย่างการแก้ปัญหาการกำหนดกฎข้อบังคับใน UML โดยใช้ { }

วิธีที่ 3 แก้ไขโดยการเขียนคลาสด้วย OONIAM ซึ่งสามารถแสดงกฎข้อบังคับได้ชัดเจนกว่า



รูปที่ 7 - 8 แสดงตัวอย่างการแก้ปัญหาการกำหนดกฎข้อบังคับใน UML โดยใช้ OONIAM

จากการค้นคว้าพบว่าใน UML สามารถอธิบายกฎข้อบังคับที่มีในฐานข้อมูลได้โดยการใช้ Object Constraint Language (OCL) แต่เนื่องจากว่า OCL จะเป็นการอธิบายข้อบังคับในเชิงภาษาคล้าย ๆ psuedo code ซึ่งจะไม่ได้แนวคิดที่ชัดเจนเท่ากับการใช้สัญลักษณ์รูปภาพเหมือนใน OONIAM

7.2 ข้อเสนอแนะ

จากการศึกษาการออกแบบระบบโปรแกรมและฐานข้อมูลด้วย UML และ OONIAM ทำให้พบว่า UML ยังมีจุดอ่อนในหลายๆ ประการในการที่จะแปลงสัญลักษณ์จากโมเดลไปเป็นโปรแกรมประยุกต์หรือฐานข้อมูลเชิงวัตถุดังที่ผู้ออกแบบได้ประสบมา

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ปัญหาการมองไดอะแกรมของแต่ละระบบงาน บางครั้งขึ้นอยู่กับว่า ผู้พัฒนาต้องการพัฒนาด้วยเครื่องมือภาษาใด แต่อย่างไรก็ดีกระบวนการออกแบบและพัฒนาเชิงวัตถุโดยเครื่องมือเชิงวัตถุ จำเป็นที่ผู้พัฒนาต้องทราบความหมายของไดอะแกรมต่างๆ ของโมเดลก่อน จากงานวิจัยฉบับนี้ ผู้วิจัยจะเน้นที่ไดอะแกรมที่มีความคาบเกี่ยวระหว่างโปรแกรมและฐานข้อมูลซึ่งก็คือคลาสไดอะแกรม และเมื่อพบสิ่งที่ขาดหายไป ผู้วิจัยจึงได้เพิ่ม OONIAM เข้าไปเพื่อเพิ่มนิยามความหมายให้กับตัวโมเดลมากยิ่งขึ้น เพื่อให้ได้การแปลงมีประสิทธิภาพสูงขึ้น

7.3 แนวทางในการพัฒนา

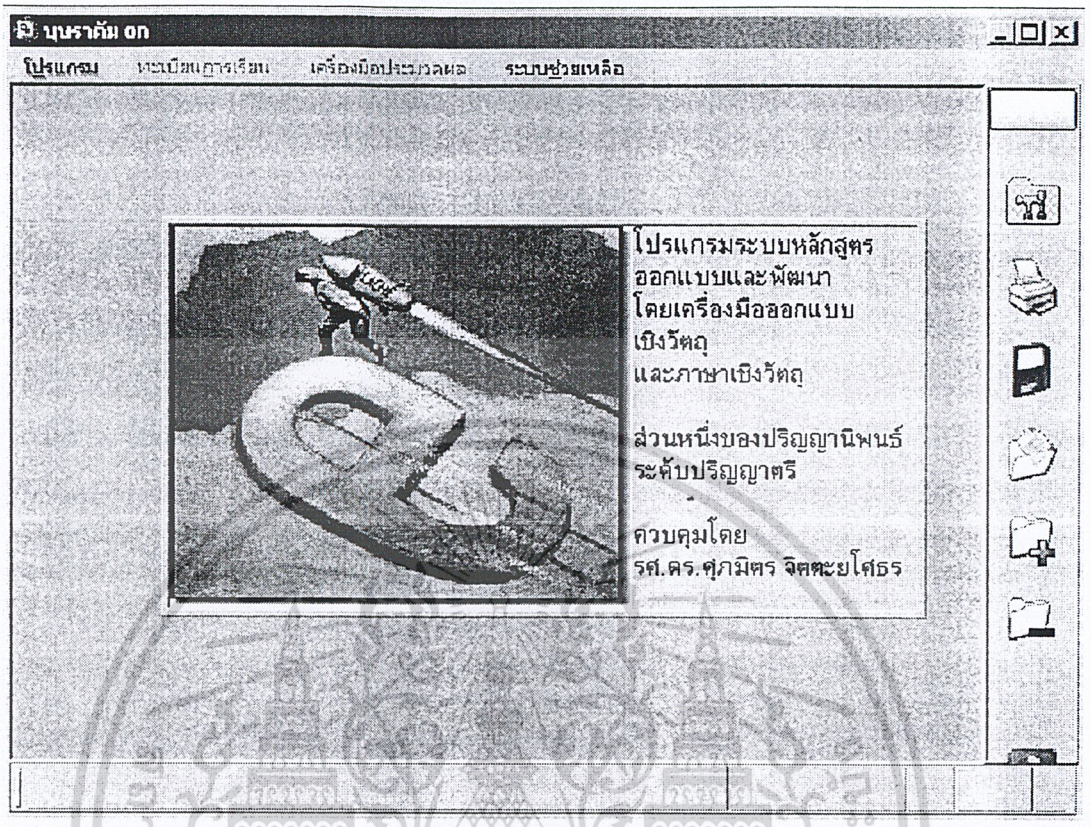
- ปรับปรุง UML ให้รองรับการออกแบบฐานข้อมูลเชิงวัตถุได้ดียิ่งขึ้น โดยไม่ต้องใช้แบบจำลองอื่น เช่น OONIAM มาช่วยแก้ปัญหา
- พัฒนาแอพลิเคชั่นต้นแบบที่สามารถ Customize ได้ง่าย ให้ใช้งานในระบบจริงได้อย่างสมบูรณ์แบบ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

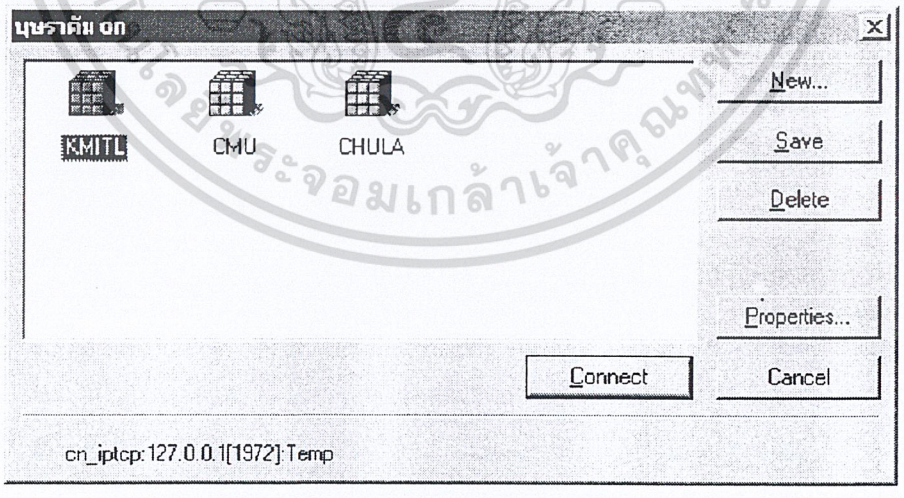
ภาคผนวก

หน้าต่างของโปรแกรม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

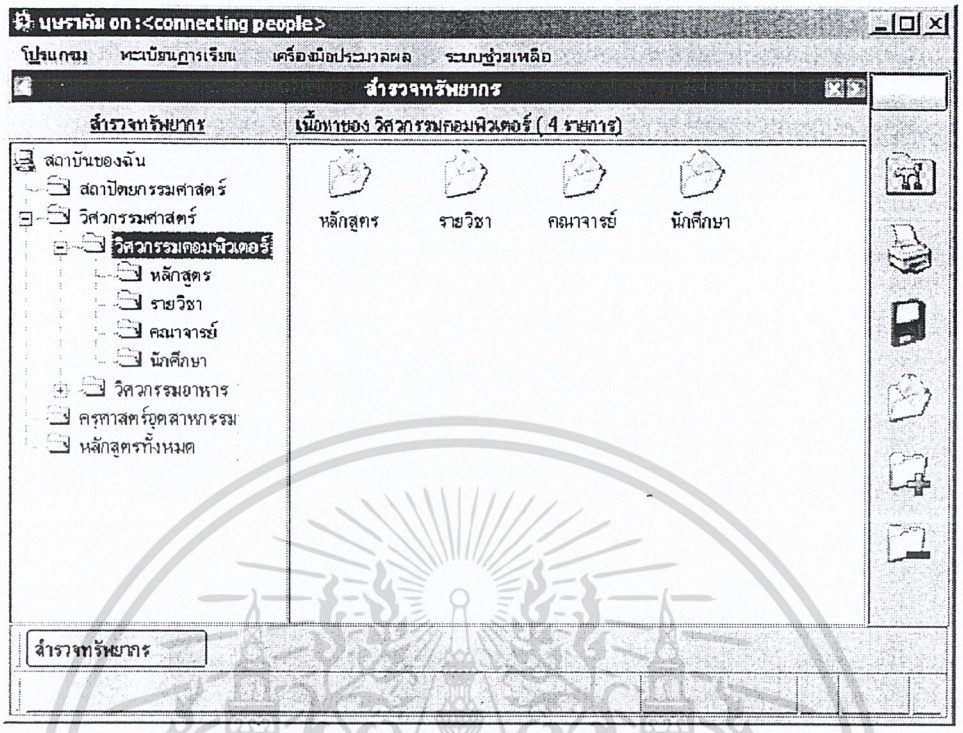


หน้าต่างโปรแกรมหลัก

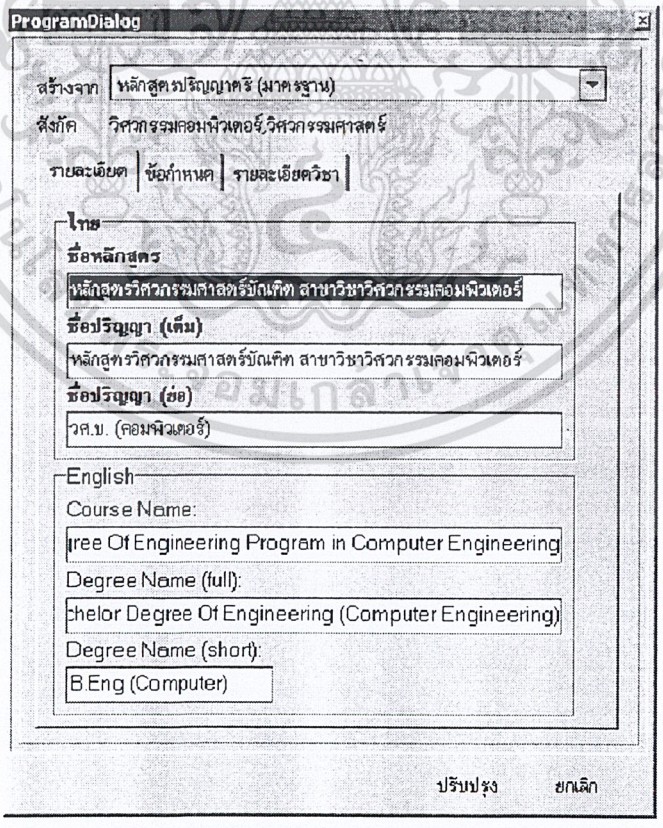


หน้าต่างการติดต่อกับ Caché

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

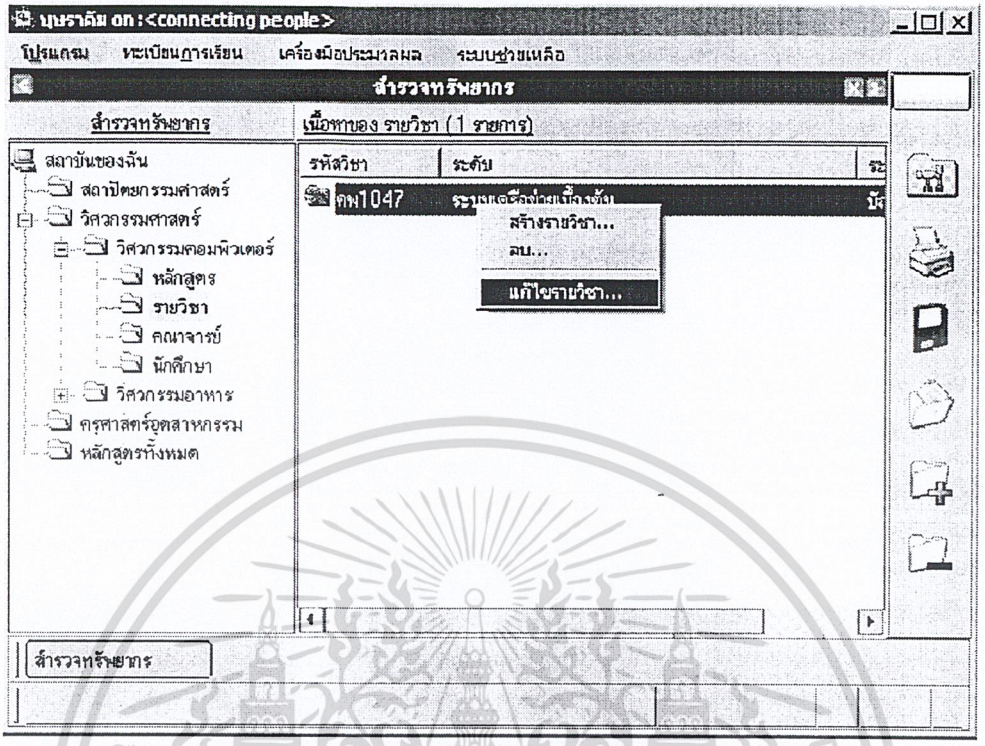


หน้าต่างแสดงทรัพยากรของระบบ

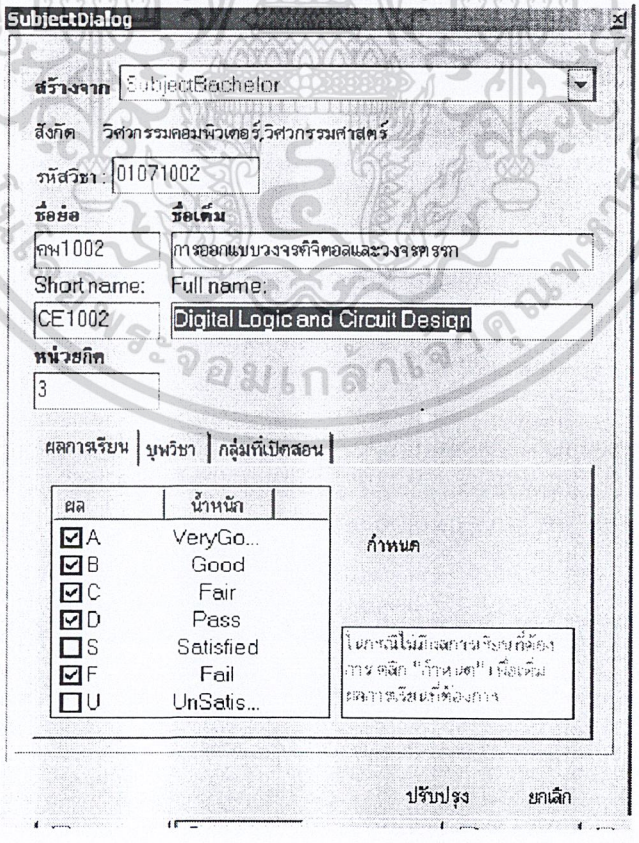


หน้าต่างแสดงการจัดการหลักสูตร

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

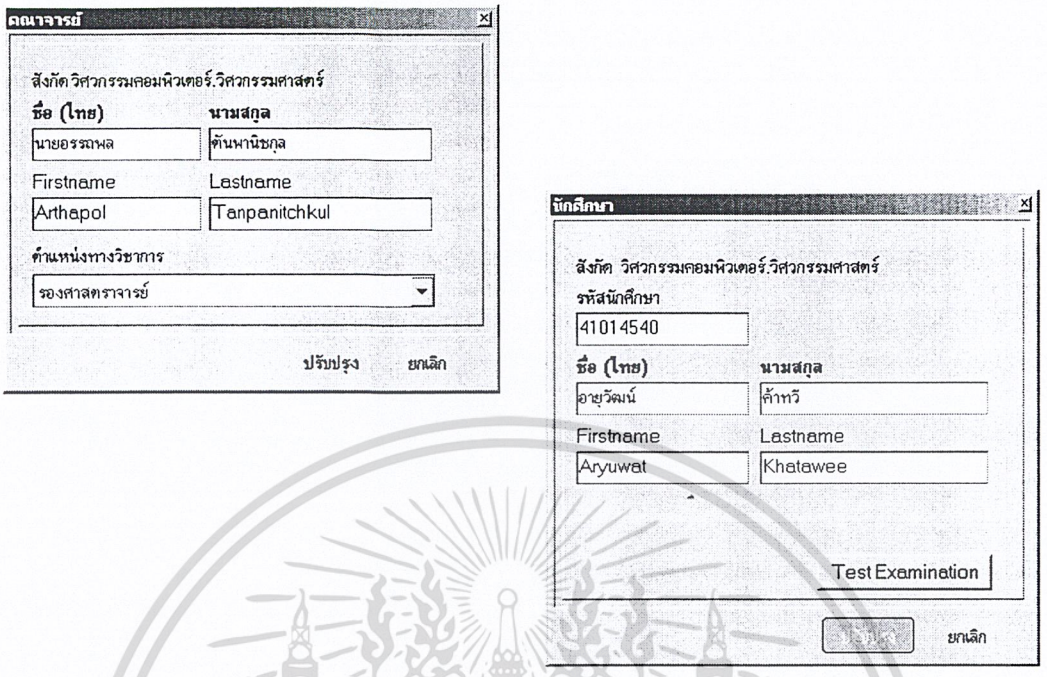


หน้าต่างทรัพยากรแสดงการเข้าจัดการกับข้อมูลด้วยเมนู

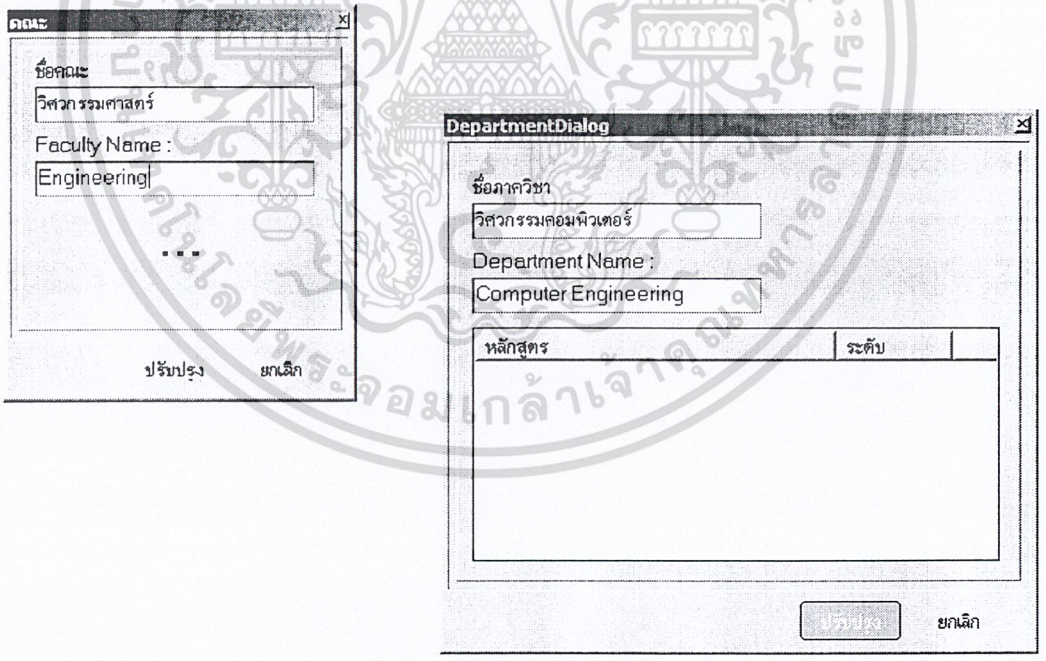


หน้าต่างการจัดการรายวิชา

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

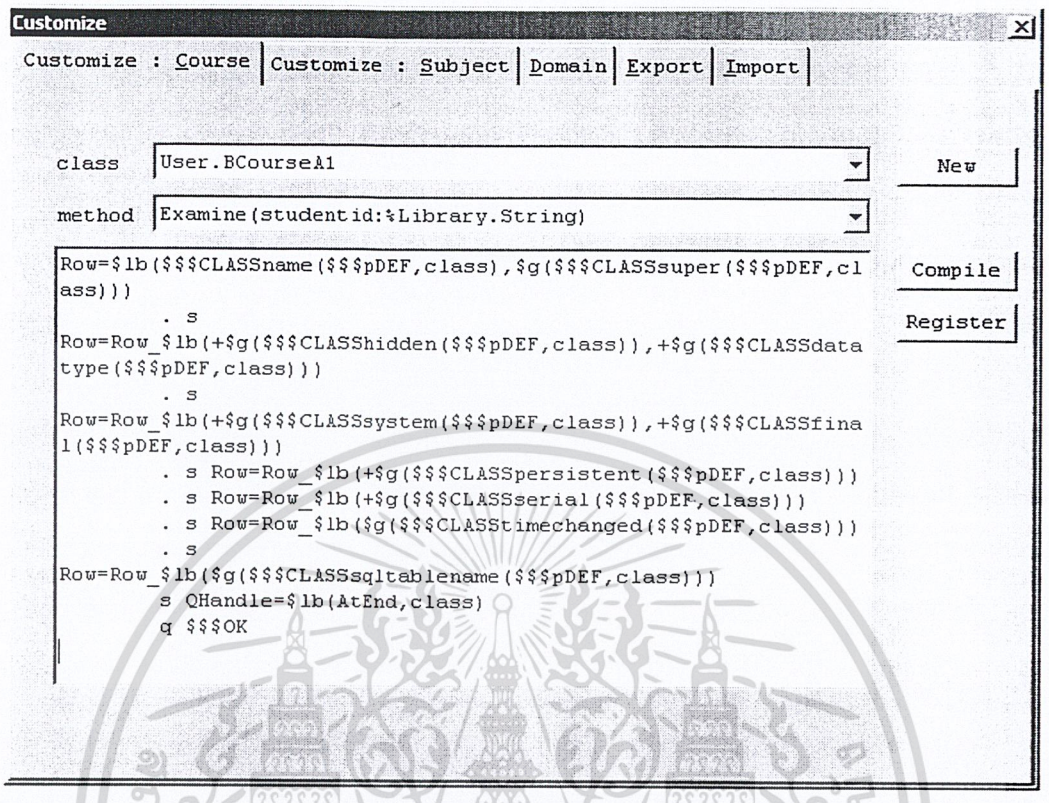


หน้าต่างแสดงการจัดการนักศึกษาและคณาจารย์

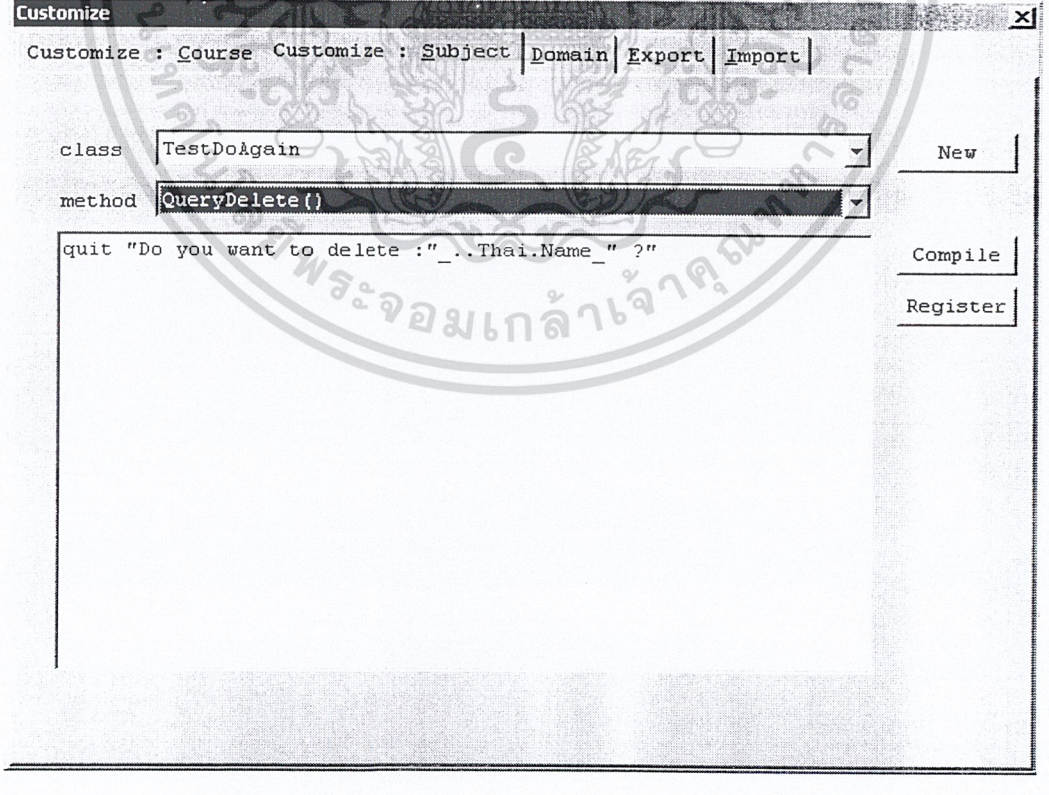


หน้าต่างสำหรับจัดการทรัพยากรทั่วไป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

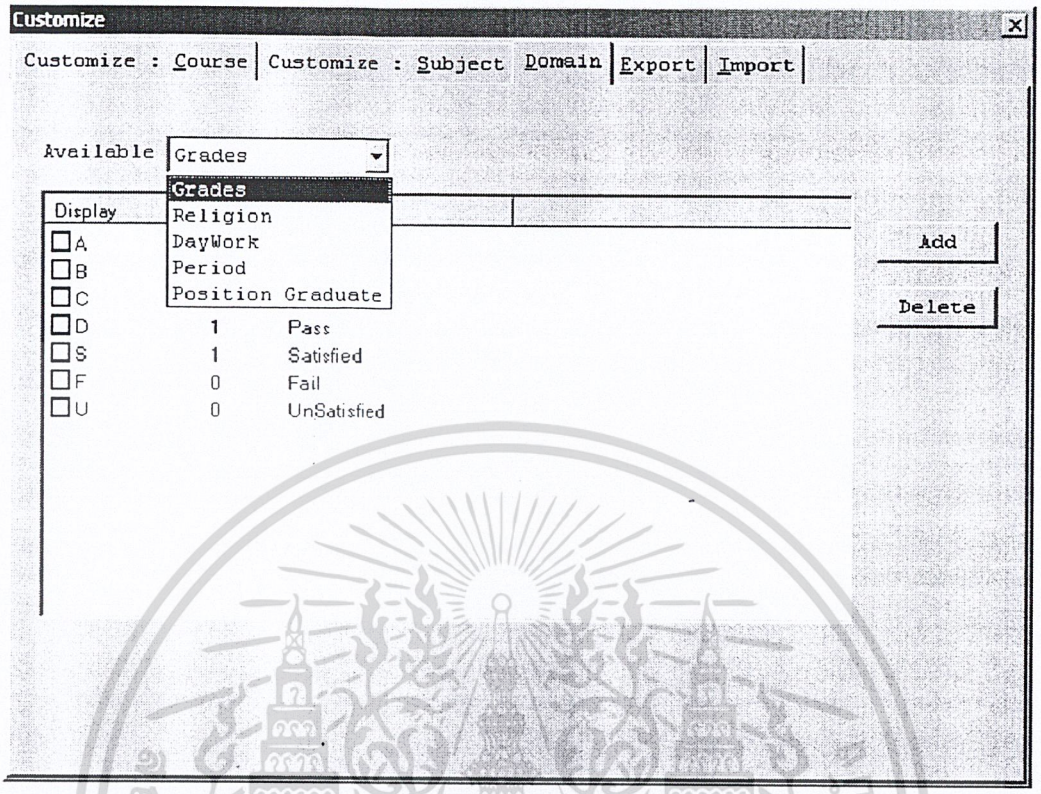


หน้าต่างโปรแกรมในส่วน Customize หลักสูตร

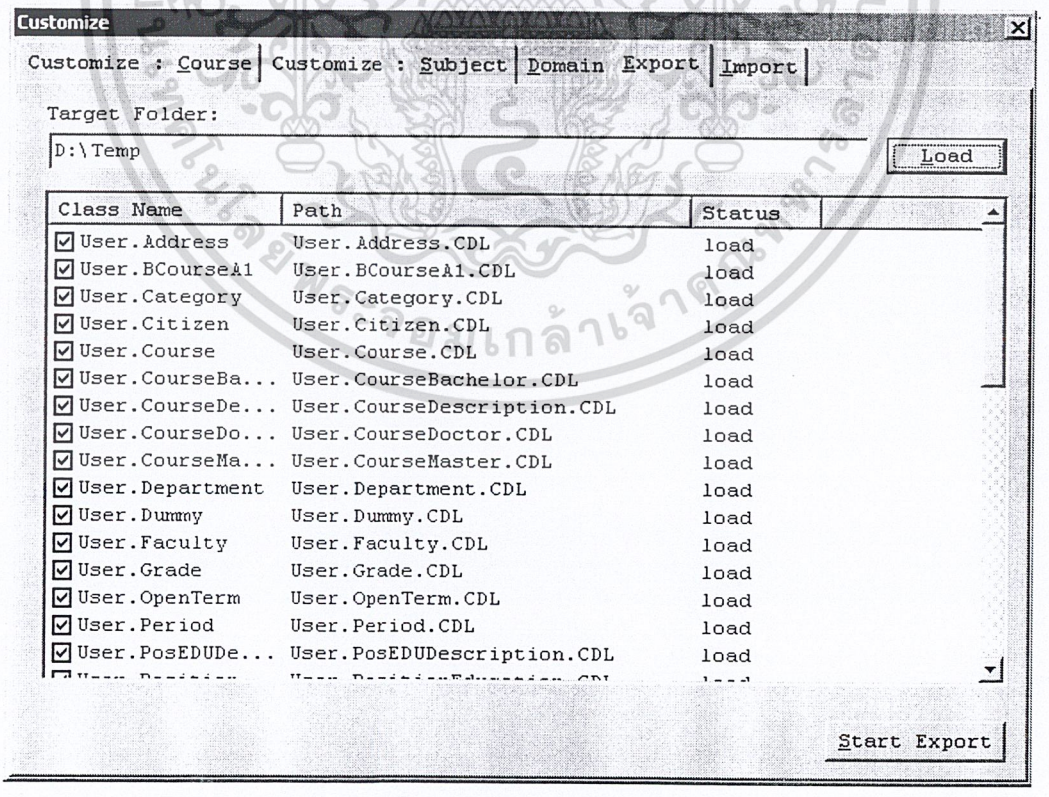


หน้าต่างโปรแกรมในส่วน Customize รายวิชา

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

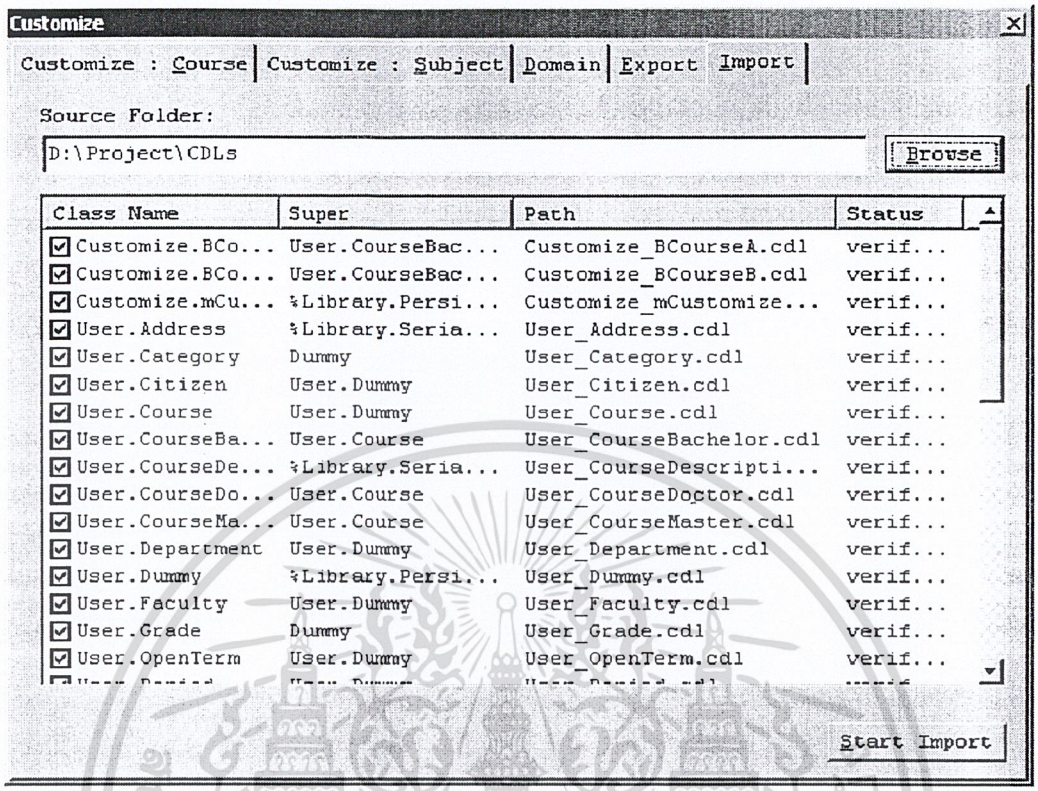


หน้าต่างแสดงการจัดการโดเมนของระบบ



หน้าต่างแสดงการส่งคลาสออกไปที่ไฟล์ CDL

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



หน้าต่างแสดงการส่งคลาสออกไปที่ไฟล์ CDL

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บรรณานุกรม

[1] Bundit Pasaya, "*The TOONIAM Conceptual Schema Model and a Transformation to an Object Oriented Database Schema*", 2544

[2] Martin Fowler, Kendall Scott, "*UML Distilled Second Edition A Brief Guide to the Standard Object Modeling Language*", Addison-Wesley Inc.,1998

[3] สุนทริน วงศ์ศิริกุล, "*พัฒนาโมเดลยุคใหม่ UML Unified Modeling Language*", Success Media CO., LTD

[4] "*OMG Unified Modeling Language Specification*", version 1.3, Object Management Group, Inc., June, 1999

[5] Eric J. Naiburg, Robert A. Maksimchuk, "*UML for Database Design*", Addison-Wesley Inc., 2001

[6] Institute of Thai Information Technology, "*Seminar Relational Database Design*", June 1994

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้