แผนภาพคลาสเชิงเวลา และการแปลงเป็นภาษานิยามเชิงวัตถุ

THE TEMPORAL CLASS DIAGRAM AND TRANSLATION TO OBJECT DEFINITION LANGUAGE

นายยุทธนา ชุ้ยวงศ์ษา
นายเอกฉันท์ ปิติสาครวิทย์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2544

แผนภาพคลาสเชิงเวลา และการแปลงเป็นภาษานิยามเชิงวัตถุ

THE TEMPORAL CLASS DIAGRAM AND TRANSLATION TO OBJECT DEFINITION LANGUAGE

โดย

นาย ยุทธนา ซุ้ยวงศ์ษา

นาย เอกฉันท์ ปิติสาครวิทย์

อาจารย์ที่ปรึกษา

อาจารย์ บัณฑิต พัสยา

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2544

ปริญญานิพนธ์ปีการศึกษา 2544

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง แผนภาพคลาสเชิงเวลาและการแปลงเป็นภาษานิยามเชิงวัตถุ

THE TEMPORAL CLASS DIAGRAM AND TRANSLATION TO OBJECT DEFINITION LANGUAGE

ผู้จัดทำ

1. นายยุทธนา ซุ้ยวงศ์ษา รหัสประจำตัว 42015315
2. นายเอกฉันท์ ปิติสาครวิทย์ รหัสประจำตัว 42015333

_____ อาจารย์ที่ปรึกษา

(อาจารย์ บัณฑิต พัสยา)

แผนภาพคลาสเชิงเวลา และการแปลงเป็นภาษานิยามเชิงวัตถุ

นาย ยุทธนา ซุ้ยวงศ์ษา
นาย เอกฉันท์ ปิติสาครวิทย์
อาจารย์บัณฑิต พัสยา อาจารย์ที่ปรึกษา
ปีการศึกษา 2544

บทคัดย่อ

ยูเอ็มแอลเป็นกระบวนความคิด และแบบจำลองข้อมูลที่สนับสนุนการวิเคราะห์และออกแบบระบบเชิงวัตถุที่ดี ผู้ใช้งานสามารถนำคลาสมาออกแบบฐานข้อมูลเชิงวัตถุโดยใช้แผนภาพคลาส และแปลงรูปเป็นภาษานิยามเชิงวัตถุได้เลย แต่แผนภาพคลาสก็มีข้อจำกัดอยู่ประการหนึ่ง คือในปัจจุบันแผนภาพคลาสไม่สามารถรองรับข้อมูลเชิงเวลาได้ ดังนั้นหากต้องการออกแบบฐานข้อมูลเชิงเวลาโดยใช้แผนภาพคลาส จำเป็นต้องเพิ่มขีดความสามารถของแผนภาพคลาสให้สามารถรองรับข้อมูลเชิงเวลาได้

โครงงานนี้ได้นำเสนอแบบจำลองข้อมูลทีซีดี ซึ่งเป็นแบบจำลองข้อมูลสำหรับออกแบบข้อมูลเชิงเวลาโดยเพิ่มสัญลักษณ์ของเวลาเข้าไปในแผนภาพคลาส นอกจากนี้ยังได้นำเสนออัลกอริธึมในการแปลงรูปจากแผนภาพคลาสเชิงเวลาให้อยู่ในรูปของภาษานิยามเชิงวัตถุเพื่อนำไปสร้างฐานข้อมูลเชิงเวลาบนระบบฐานข้อมูลเชิงวัตถุ

The Temporal Class Diagram and Translation to Object Definition Language

Yutthana Suiwongsa

Akachan Pitisakhonwit

Bundit Passaya Advisor

ABSTRACT

UML is the methodology and the data model that support analysis and designing of good Object-Oriented system. User can use class diagram to design Object-Oriented Database and can translate to Object Definition Language but class diagram has one limitation. Currently, class diagram can not support temporal data. If we want to design Temporal Database by class diagram, we must increase ability of class diagram to support temporal data.

This project presented model of Temporal Class Diagram (TCD) that be data model for designing temporal data that add sign of time into class diagram. In additional, we presented algorithm to translate from Temparal Class Diagram to Object Definition Language for creating Temporal Database on Object-Oriented Database Management System.

กิตติกรรมประกาศ

ปริญญานิพนธ์นี้คงไม่อาจสำเร็จลุล่วงได้ด้วยดี หากไม่ได้รับความช่วยเหลือจากหลาย ๆ ฝ่ายด้วยกัน ซึ่งบุคคลที่สำคัญที่สุดนอกจากผู้ทำปริญญานิพนธ์นี้ คือ อ.บัณฑิต พัสยา อาจารย์ที่ปรึกษาปริญญานิพนธ์ ที่คอยให้คำแนะนำแนวทางการดำเนินงาน, เอาใจใส่ให้ความช่วยเหลือ ทั้งในเรื่องเอกสารทรัพยากรต่าง ๆ และคอยกระตุ้นให้มีผลงานตลอด ทำให้การทำปริญญานิพนธ์เป็นไปได้ด้วยดี ทางผู้จัดทำมีความรู้สึกภาคภูมิใจเป็นอย่างมากที่ได้มีโอกาสเข้ามาทำปริญญานิพนธ์นี้ ขอกราบขอบพระคุณในความกรุณาเป็นอย่างยิ่ง

ขอขอบคุณบริษัท InterSystems ที่เอื้อเฟื้อให้เอกสารและเครื่องมือต่างๆ ทั้งยังได้จัดฝึกอบรมทำให้ผู้จัดทำสามารถทำปริญญานิพนธ์นี้ได้อย่างเต็มความสามารถ ขอบคุณภาควิชาวิศวกรรมคอมพิวเตอร์ที่ได้สนับสนุนสถานที่และทรัพยากรต่างๆ ขอบคุณคณาจารย์ทุกท่านที่ประสิทธิประสาทวิชาความรู้ความสามารถ ขอบคุณเพื่อนๆ ทุกคนในห้องวิจัย OLALA ที่ทำให้บรรยากาศในการทำปริญญานิพนธ์เป็นไปอย่างสนุกสนานในทุกๆ โอกาส ขอบคุณเพื่อนๆ และน้องๆ ในห้องวิจัย ISAG ที่ให้ใช้เครื่องพิมพ์ และให้ความรู้เพิ่มเติมในหลายๆ ด้าน ขอบคุณพี่ๆ ในห้องวิจัย HARDWARE ที่ให้คำแนะนำที่ดีเสมอมา ขอบคุณทุกๆ คนที่คอยเป็นห่วงและถามถึงโปรเจ็กต์ที่ทำถึงไหนแล้ว และขอบคุณสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ซึ่งเป็นสถาบันที่รักยิ่งขอขอบคุณสำหรับโอกาสที่ให้ข้าพเจ้าและเพื่อน ๆ ได้เข้ามาศึกษาเล่าเรียนวิชาจนมีวันนี้

สุดท้ายขอขอบคุณสิ่งศักดิ์สิทธิ์ทั้งหลายในสากลโลกนี้ที่ได้ดลบันดาลให้ข้าพเจ้าได้เกิดมาเป็นลูกของท่าน ก็คือ บิดา มารดา อันเป็นที่เคารพรักยิ่ง ซึ่งท่านได้คอยสนับสนุน อบรมเลี้ยงดู ให้กำลังใจพร้อมทั้งให้โอกาสและการศึกษาที่ดีแก่ข้าพเจ้าอย่างเต็มที่ ข้าพเจ้าขอระลึกในพระคุณอันยิ่งใหญ่นี้ และขอกราบขอบพระคุณมา ณ ที่นี้ด้วย

นาย ยุทธนา ซุ้ยวงศ์ษา
นาย เอกฉันท์ ปิติสาครวิทย์

สารบัญ

หน้าที่

สารบัญ (ต่อ)

หน้าที่

สารบัญ (ต่อ)

หน้าที่

สารบัญ (ต่อ)

หน้าที่

สารบัญ (ต่อ)

หน้าที่

สารบัญ (ต่อ)

หน้าที่

สารบัญตาราง

หน้าที่

สารบัญภาพ

หน้าที่

สารบัญภาพ (ต่อ)

หน้าที่

สารบัญภาพ (ต่อ)

หน้าที่

สารบัญภาพ (ต่อ)

หน้าที่

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มา

ในระบบฐานข้อมูลเดิมนั้นจะเก็บข้อมูลสถานะ (State) เดียว ซึ่งถ้าต้องการเรียกค้นข้อมูลเก่า โดยที่ข้อมูลนั้นมีการเปลี่ยนแปลงไปแล้วก็จะไม่สามารถทำได้ เช่น การเก็บความสูงของนักเรียนเมื่ออายุ 10 ปี เมื่อนักเรียนผู้นั้นอายุ 15 ปีแล้วทำการบันทึกข้อมูลลงไป โดยข้อมูลเมื่อนักเรียนผู้นั้นอายุ 10 ปีก็จะเกิดการสูญหายได้ สามารถแก้ไขโดย ให้เพิ่มสัญลักษณ์ของเวลาเข้าไป

ในยูเอ็มแอล (Unified Modeling Language : UML) นั้นเป็นกระบวนความคิด (Methodology) และโมเดลที่สนับสนุนการวิเคราะห์และออกแบบระบบเชิงวัตถุ โดยเป็นการพัฒนาระบบเชิงวัตถุซึ่งเป็นวิธีการที่จะพัฒนาซอฟต์แวร์ (Software) รูปแบบใหม่โดยผู้พัฒนาระบบจะมองระบบและส่วนประกอบหรือระบบย่อย (Sub System) เป็นวัตถุหรือออบเจ็กต์ (Object) เนื่องจากสิ่งต่างๆ ที่มีอยู่ในโลกความเป็นจริงนั้น ก็ถือได้ว่าเป็นวัตถุอยู่แล้ว และด้วยการมองระบบเป็นวัตถุทำให้ผู้พัฒนาระบบสามารถมองระบบได้ชัดเจนมากยิ่งขึ้น ส่งผมให้การวิเคราะห์และออกแบบระบบทำได้ง่ายและมีความถูกต้องมากขึ้น

แผนภาพคลาสเป็นเครื่องมือตัวหนึ่งของ UML ที่ช่วยในการแสดงคลาส (Class) และความสัมพันธ์ (Relationship) ที่ใช้ในการออกแบบฐานข้อมูลเชิงวัตถุได้ แต่แผนภาพคลาสนั้นยังไม่สามารถรองรับข้อมูลเชิงเวลาได้ ดังนั้นหากต้องการนำแผนภาพคลาสมาออกแบบฐานข้อมูลเชิงเวลา จำเป็นต้องเพิ่มคุณสมบัติบางอย่างเข้าไป โดยรายละเอียดจะนำเสนอในส่วนถัดไป

1.2 วัตถุประสงค์

1.2.1 เพื่อศึกษาฐานข้อมูลเชิงวัตถุ (Object-Oriented Database : OODB)

1.2.2 เพื่อศึกษาฐานข้อมูลเชิงเวลา (Temporal Database)

1.2.3 นำเสนอแผนภาพคลาสเชิงเวลา (Temporal Class Diagram : TCD)

1.2.4 นำเสนอการแปลงแผนภาพคลาสเชิงเวลา ไปเป็นภาษานิยามเชิงวัตถุโอดีแอล (Object Definition Language : ODL)

1.2.5 เพื่อสร้างซอฟต์แวร์สำหรับวาดแผนภาพคลาสเชิงเวลา (TCD Schema Model) ที่สามารถแปลงเป็นภาษานิยามเชิงวัตถุโอดีแอล

1.3 ขอบเขตของโครงงาน

โครงงานนี้เป็นโครงงาน ที่มีลักษณะเป็นการสร้างโปรแกรมประยุกต์ที่ได้จากงานวิจัยโดยเป็นการศึกษาการเพิ่มสัญลักษณ์ของเวลาเข้าไปในแผนภาพคลาส เพื่อมาแก้ปัญหาต่างๆที่เกิดขึ้นกับข้อมูลเชิงเวลา และหลักการแปลงแผนภาพคลาส ที่มีการเพิ่มสัญลักษณ์ของเวลาไปเป็น ภาษานิยามเชิงวัตถุโอดีแอล เพื่อใช้กับฐานข้อมูลเชิงวัตถุ และจะนำหลักการเหล่านั้นมาสร้างเป็นซอฟต์แวร์สำหรับวาดแผนภาพคลาสเชิงเวลาที่มีความสามารถในการแปลงรูปที่วาดไปเป็นภาษานิยามเชิงวัตถุ ซึ่งใช้สร้างฐานข้อมูลบนฐานข้อมูลเชิงวัตถุได้

1.4 วิธีการดำเนินโครงงาน

1.4.1 ศึกษาระบบฐานข้อมูลเชิงวัตถุ

1.4.2 ศึกษาระบบฐานข้อมูลเชิงเวลา

1.4.3 ศึกษายูเอ็มแอล

1.4.4 ศึกษาการพัฒนาแผนภาพคลาสในยูเอ็มแอลให้สามารถรองรับข้อมูลเชิงเวลา

1.4.5 ศึกษาอัลกอริทึม (Algorithm) ในการแปลงแผนภาพคลาสเชิงเวลา ไปเป็นภาษานิยามเชิงวัตถุโอดีแอล

1.4.6 ทำการสร้างโปรแกรม ที่ใช้ในการวาดแผนภาพคลาสเชิงเวลา และสามารถแปลงรูปที่ได้ไปเป็น ภาษานิยามเชิงวัตถุโอดีแอล

บทที่ 2

ยูเอ็มแอล

(Unified Modelling Language : UML)

2.1 ยูเอ็มแอล

ยูเอ็มแอล (Unified Modelling Language:UML) เป็นภาษาในการโมเดลมาตรฐาน เกิดจากการพัฒนาร่วมกันของผู้นำเทคโนโลยีทางด้านวัตถุ 3 คน คือ Grady Booch, Ivar Jacobson และ Jim Rumbaugh

ยูเอ็มแอลนั้นได้รวมแนวความคิดของวิธีการต่างๆเข้าไว้ด้วยกัน จุดประสงค์ของยูเอ็มแอลก็คือต้องการสร้างโมเดลในการโมเดลในการพัฒนาที่เข้าใจและสร้างได้ง่าย แต่สามารถนำไปใช้ได้กับทุกๆ ระบบ

ในยูเอ็มแอลมีโมเดลที่สื่อสารด้วยภาพได้สำหรับระบบหลายๆโมเดล โดยแต่ละโมเดลก็จะแสดงมุมมองต่อระบบที่ไม่เหมือนกัน ซึ่งประกอบด้วย

1. ยูสเคสไดอะแกรม (Use case diagram) ที่แสดงการติดต่อระหว่างระบบกับผู้ใช้
2. คลาสไดอะแกรม (Class diagram) ใช้แสดงโครงสร้างทางตรรกของระบบ
3. ออบเจ็กต์ไดอะแกรม (Object diagram) ใช้แสดงวัตถุและความสัมพันธ์ระหว่างวัตถุ
4. สแตติกไดอะแกรม (Static diagram) ใช้แสดงพฤติกรรมของระบบ
5. คอมโพเนนท์ไดอะแกรม (Component diagram) ใช้แสดงโครงสร้างทางกายภาพของซอฟต์แวร์
6. ดีพลอยเมนท์ไดอะแกรม (Deployment diagram) ที่ใช้แสดงการติดต่อระหว่างซอฟต์แวร์กับฮาร์ดแวร์
7. อินเตอร์แอกทีฟไดอะแกรม (Interaction diagram) ใช้แสดงพฤติการณ์ของระบบ
8. แอกติวิตี้ไดอะแกรม (Activity diagram) ใช้แสดงการไหลของอีเวนต์ (Event) ในยูสเคส

2.2 Extension Mechanisms

ถึงแม้ว่ายูเอ็มแอลนั้นประกอบไปด้วยสัญลักษณ์และเครื่องหมายจำนวนมาก โดยสัญลักษณ์และเครื่องหมายจำนวนมากของยูเอ็มแอลเหล่านี้ ทำให้สามารถสร้างโมเดลและแผนภาพได้อย่างยืดหยุ่น และคลอบคลุมโดเมนของปัญหาทั้งหมด แต่ก็ยังไม่เพียงพอที่จะใช้แทนทุกๆ สิ่งที่มีในโลกความจริง จึงได้สร้างสิ่งที่สามารถมาขยาย หรือรวมเข้าด้วยกัน เพื่อที่จะอธิบายโมเดลใหม่ ซึ่งแตกต่างจากโมเดลมาตรฐานของยูเอ็มแอล ซึ่งกลไกในการขยายนี้คือ สเตอริโอไทพ์ (Stereotype), แทกแวลู (Tagged values) และคอนสเตรนท์ (Contraints)

สเตอริโอไทพ์ (Stereotype) เป็นเมต้าโมเดล (Metamodel) ของยูเอ็มแอล หรือกล่าวได้ว่าเป็น เอนติตี้ (Entity) ซึ่งเป็นส่วนขยายสำหรับยูเอ็มแอล ซึ่งเป็นส่วนที่สำคัญมากสามารถใช้เมต้าโมเดลนี้ในการแทนความหมายของสิ่งเหล่านั้นตามที่ต้องการได้ อย่างไรก็ดีสัญลักษณ์และเครื่องหมายของยูเอ็มแอลทั้งหมดสามารถเขียนในรูปของสเตอริโอไทพ์ ได้เช่นกัน

สำหรับสัญลักษณ์ของสเตอริโอไทพ์นั้นจะเขียนชื่อเอนติตี้ที่จะเป็นสเตอริโอไทพ์ ภายในวงเล็บ <<sterotype name>> และนำไปวางไว้หน้าชื่อของส่วนที่ต้องการจะอธิบาย เช่น "<<type>>tree" เป็นการกำหนดสเตอริโอไทพ์ว่า tree เป็นชนิดของข้อมูลเพื่อใช้ในการเขียนโปรแกรม

แทกแวลู (Tagged value) ใช้อธิบายคุณลักษณะเฉพาะของอีลีเมนท์ (Element) โดยเขียนเป็นคู่ (tag,value) โดย tag คือ ชื่อของ property และ value คือ ค่าของ property นั้น โดยแทกแวลูจะเขียนอยู่ในเครื่องหมายปีกกา {}

คอนสเตรนท์ (Contraint) เป็นเงื่อนไขหรือข้อบังคับของโมเดลอีลีเมนท์ (Model element) ให้เป็นไปตามระบบสำหรับอธิบายโมเดลที่ต้องการ โดยคอนสเตรนท์นั้นจะเขียนอยู่ในเครื่องปีกกา {}

2.3 Static Structure Elements

2.3.1 วัตถุหรือออบเจ็กต์ (Object)

ออบเจ็กต์ใน ยูเอ็มแอล นั้น คือวัตถุ คือ กลุ่มก้อนของเอนติตี้ที่ประกอบไปด้วยคุณสมบัติหรือแอตทริบิวต์ (attribute) พฤติกรรมหรือเมธอด(Behavior or Method) โดยทั้งโครงสร้างและพฤติกรรมทั้งหมดมาจากคลาสของมันเอง

2.3.2 คลาส (Class)

คลาสคือการแยกแยะเอกลักษณ์ของคุณสมบัติพื้นฐานจากกลุ่มของวัตถุที่เหมือนกัน หรืออาจกล่าวได้ว่าคลาสเป็นชนิดของวัตถุ คลาสจะกำหนดแอททริบิวต์และเมธอดของวัตถุ โดยเมื่อนำไปใช้งานจริง คลาสของวัตถุจะกำหนดความรับผิดชอบของวัตถุเฉพาะเมื่อวัตถุที่ถูกสร้างขึ้นจากคลาสทั้งหมดถูกใช้ในสภาวะแวดล้อมและมีหน้าที่ที่เหมือนกัน

คลาสในยูเอ็มแอล นั้นแบ่งออกเป็น 3 ส่วน ได้แก่ Name compartment, Attribute compartment และ Operation compartment

Name compartment
Attribute compartment
Operation compartment

รูปที่ 2-1ลักษณะเบื้องต้นของ Class

1. Name compartment คือ ส่วนบนสุดของสี่เหลี่ยมใช้ใส่ชื่อของคลาส ซึ่งโดยทั่วไปจะใช้ตัวหนาอยู่ตรงกลาง และควรเป็นคำนาม

2. Attribute compartment คือส่วนกลางของสื่อเหลี่ยม โดยใช้แสดงรายการของแอตทริบิวต์โดย แอตทริบิวต์ ที่แสดงนั้นจะต้องระบุชนิดของมัน (Primitive type) และความแตกต่างในการเข้าถึงข้อมูล (Visibility) ด้วย

3. Operation compartment คือส่วนล่างสุดของสี่เหลี่ยมใช้ระบุ ใช้แสดงรายการโอเปอเรชั่น (Operation) ซึ่งจะมีการระบุความแตกต่างในการใช้ (Visibility) เช่นกัน

2.4 ความสัมพันธ์ระหว่างคลาส

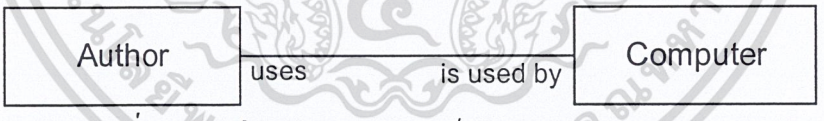
ในคลาสไดอะแกรมประกอบไปด้วยคลาสและ ความสัมพันธ์ระหว่างคลาสเหล่านั้น โดยความสัมพันธ์เหล่านั้น สามารถแบ่งออกได้เป็น 3 ประเภทดังต่อไปนี้

2.4.1 ความสัมพันธ์แบบแอซโซซิเอชั่น (Associations)

เป็นความสัมพันธ์ระหว่างคลาส โดยที่วัตถุของคลาสหนึ่งมีการใช้บริการของอีกวัตถุของอีกคลาสหนึ่ง ซึ่งไม่ได้เป็นเจ้าของวัตถุนั้น หรือกล่าวอีกในหนึ่งว่า ความสัมพันธ์แบบแอซโซซิเอชั่นนั้นจะเกิดขึ้นเมื่อวัตถุของคลาสสองวัตถุมีความเกี่ยวข้องกันหรือมีการสื่อสารกัน ตัวอย่างเช่น ลูกค้าเป็นเจ้าของบัญชี หรือ นักเขียนใช้คอมพิวเตอร์ โดยความสัมพันธ์แบบแอซโซซิเอชั่นแบ่งได้เป็น

2.4.1.1 Normal Association

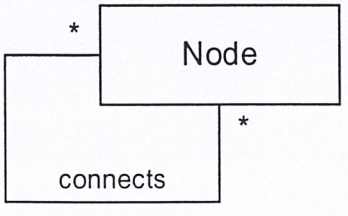
เป็นเพียงการเชื่อมต่อระหว่างคลาสสองคลาส โดยมากจะใช้เส้นตรงเชื่อมระหว่างคลาส และมีชื่อของความสัมพันธ์ระบุไว้บนเส้น โดยส่วนมากจะใช้คะกริยา และมีหัวลูกศรชี้ที่ปลายของความสัมพันธ์ ดังรูป



รูปที่ 2-2 ตัวอย่างของความสัมพันธ์แบบ Normal Association

2.4.1.2 Recursive Association

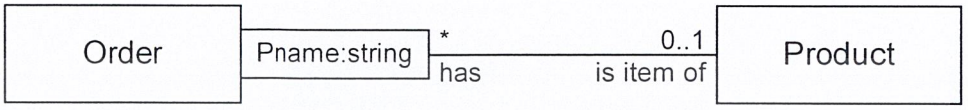
เป็นความสัมพันธ์ระหว่างคลาสที่แทนที่จะสัมพันธ์กับคลาสอื่น แต่กลับมีการติดต่อกับวัตถุซึ่งเป็นคลาสเดียวกัน เช่น เครือข่าย (Network) หนึ่งประกอบไปด้วยโหนด (Node) หลายโหนดติดต่อกันแต่ละโหนด เขียนได้เป็นโหนดคลาส (Node class) มีความสัมพันธ์แบบแอซโซซิเอชั่นกับโหนดคลาส



รูปที่ 2-3 ตัวอย่างของ Recursive Association

2.4.1.3 Qualified Association

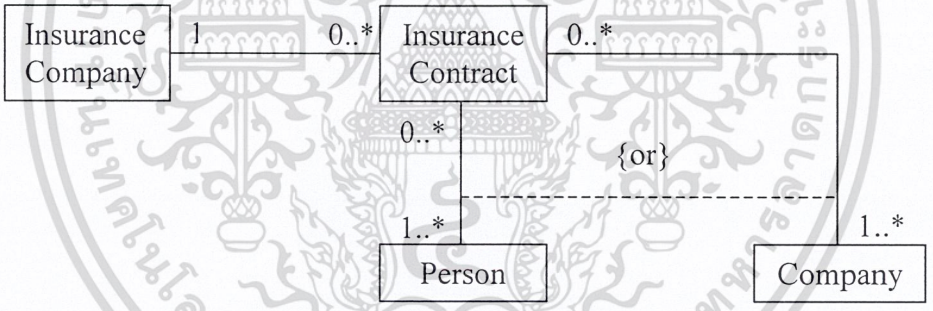
เป็นความสัมพันธ์ระหว่างคลาสที่ถูกใช้กับ ความสัมพันธ์ที่มีการระบุจำนวน แบบหนึ่งต่อหลายหรือหลายต่อหลายแอซโซซิเอชั่น (One-to-many or Many-to-many associations) เป็นระบุว่าวัตถุของคลาสหนึ่งมีการติดต่อกับวัตถุหลายๆวัตถุ ที่เป็นของคลาสชนิดเดียวกันแต่มีคุณสมบัติใดที่มีความแตกต่างกันซึ่งจะไม่มีการซ้ำกันเลย เช่น



รูปที่ 2-4 ตัวอย่างของ Qualified Association

2.4.1.4 Or-Association

เป็นความสัมพันธ์ที่มีเงื่อนไขบนการแอซโซซิเอชั่นที่มีมากกว่า 2 ขึ้นไปซึ่งเป็นการเลือกการแอซโซซิเอชั่นกับคลาสใดคลาสหนึ่งเท่านั้นในเวลาเดียวกัน เช่น Insurance contract คลาสไม่สามารถแอซโซซิเอชั่นทั้ง Company คลาสและ Persen คลาสในเวลาเดียวกันได้

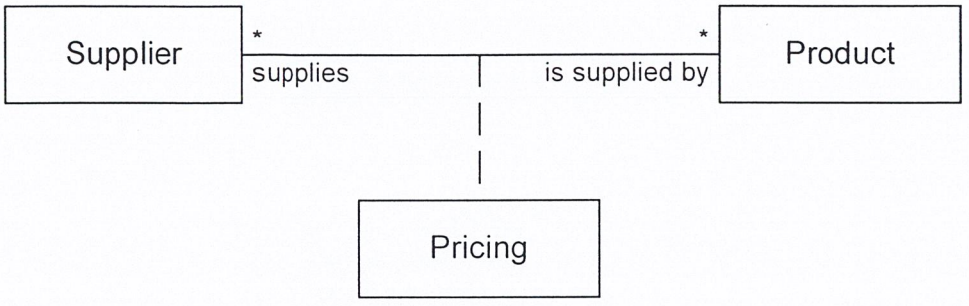


รูปที่ 2-5 ตัวอย่างของ Or-Association

2.4.1.5 แอซโซซิเอชั่นคลาส (Association Class)

เป็นคลาสที่เกิดขึ้นจากความสัมพันธ์แบบแอซโซซิเอชั่นระหว่างคลาสอื่น โดยที่แอซโซซิเอชั่นคลาสเป็นเพียงคลาสปกติที่มี แอตทริบิวต์, โอเปอเรชั่น และแอซโซซิเอชั่นอื่น ซึ่งถูกใช้ในการเพิ่มรายละเอียดพิเศษในการติดต่อระหว่างคลาส

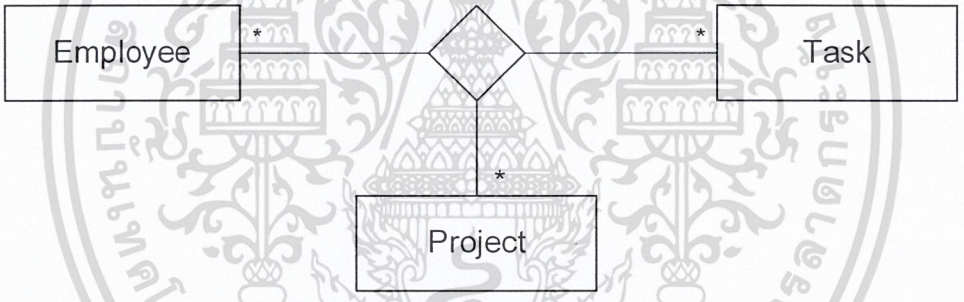
ในระบบเฉพาะ เราสามารถสร้างแอซโซซิเอชั่นคลาสที่ใช้เก็บข้อมูลและมีเมธอดสำหรับทำงานบางอย่างได้ตามระบบที่ต้องการด้วย ตัวอย่างของแอซโซซิเอชั่นคลาส ได้แก่ คลาสสำหรับความสัมพันธ์ระหว่างวัตถุวิชาเรียนกับทะเบียนนักศึกษา โดยแอซโซซิเอทีฟคลาส (Associative Class) ของความสัมพันธ์นี้จะเก็บข้อมูลหรือแอตทริบิวต์ต่างๆ ไว้เช่นสถานะของการลงทะเบียนหรือเกรดที่ได้ เป็นต้น



รูปที่ 2-6 ตัวอย่างของ Association Class

2.4.1.6 ความสัมพันธ์แบบหลายบทบาท (N-ary association)

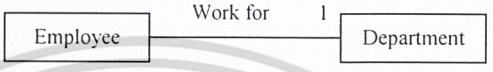
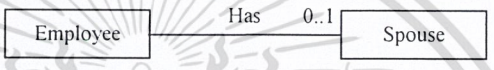
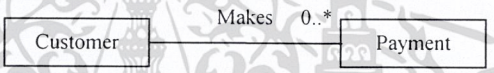
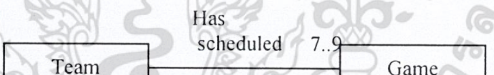
คือแอซโซซิเอชั่นตั้งแต่ 3 คลาส ขึ้นไป โดยที่ทุกอินสแตนซ์ (Instance) ของแอซโซซิเอชั่นเป็นเอ็นทัพเปิ้ล (N-tuple) ของค่าจากคลาสที่เกี่ยวข้อง เช่น โปรเจค (Project) หนึ่งต้องใช้คนงาน (Employee) เพื่อให้ได้ งาน (Task) ออกมา เขียนได้เป็น



รูปที่ 2-7 ตัวอย่างของ N-ary Association

2.4.1.7 Association with Multiplicity

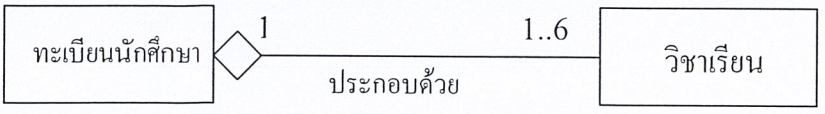
Multiplicity ใช้กำหนดจำนวนอินสแตนซ์ของวัตถุหรือคลาสที่แอซโซซิเอชั่นกับอินสแตนซ์ของวัตถุหรือคลาสอื่นๆ โดยแบ่งเป็น

Multiplicity	**UML Multiplicity Notation**	**Association With Multiplicity**	**Association Meaning**
Exactly 1	1 or leave blank		An employee work for one and only one department.
Zero or one	0..1		An employee has either one or no spouse.
Zero or more	0..* or *		A customer can make no payment up to many payment.
One or More	1..*		A university offers at least 1 couse up to many couse.
Specfic range	7..9		A team has either 7, 8 or 9 games scheduled

ตารางที่ 2-1 Multiplicity

2.4.2 ความสัมพันธ์แบบเป็นส่วนหนึ่งของ (Aggregation)

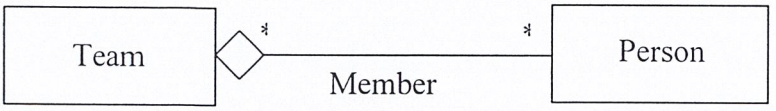
ความสัมพันธ์แบบเป็นส่วนหนึ่งของเป็นความสัมพันธ์แบบแอซโซซิเอชั่นชนิดพิเศษ โดยจะหมายถึงว่าคลาสหนึ่งประกอบด้วยคลาสอีกคลาสหนึ่งทั้งในทางตรรก เช่น ทะเบียนนักศึกษาจะประกอบไปด้วยคลาสของวิชาที่นักศึกษาลงทะเบียน และในทางกายภาพ เช่นคลาสระบบตู้เอทีเอ็ม (ATM) ประกอบไปด้วยคลาสเครื่องจ่ายเงิน คลาสที่ใหญ่กว่าเรียกว่า Owner class หรือ Whole ซึ่งเป็นคลาสที่มีหัวลูกศรรูปสี่เหลี่ยมอยู่ ส่วนคลาสที่เล็กกว่าเรียกว่า Component class หรือ Owned หรือ Part



รูปที่ 2-8 ตัวอย่างของ Aggregation

2.4.2.1 Share aggregation

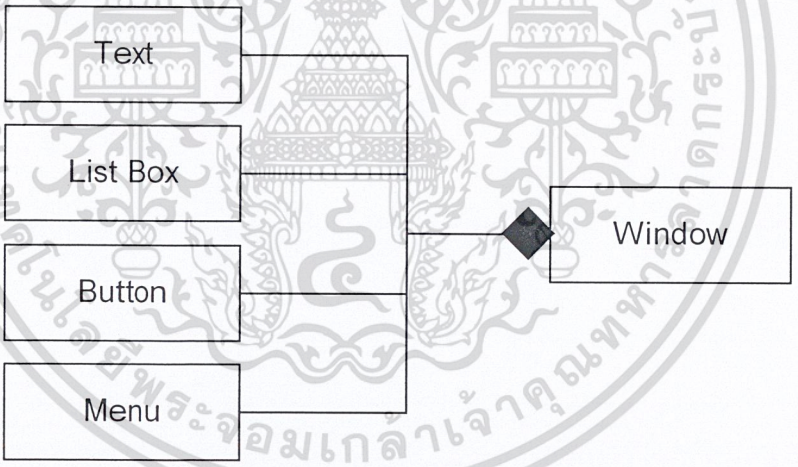
เป็นชนิดพิเศษของความสัมพันธ์แบบเป็นส่วนหนึ่งของโดย วัตถุที่เป็นส่วนหนึ่งของวัตถุอื่นๆ นั้นสามารถมีเจ้าของได้มากกว่า 1 วัตถุ ซึ่งแสดงให้เห็นโดย Multiplicity ของฝั่ง Owner คลาสมีมากกว่า 1 หรือเป็นหลาย (Many) นั่นเอง



รูปที่ 2-9 ตัวอย่างของ Share Aggregation

2.4.2.2 Composition Aggregation

เป็นชนิดพิเศษของความสัมพันธ์แบบเป็นส่วนหนึ่งของคือนอกจากวัตถุที่เป็นเจ้าของ(Composite object) นั้นจะประกอบด้วยวัตถุส่วนประกอบ (Component object) ที่อยู่ภายในแล้ว วัตถุที่เป็นเจ้าของยังมีหน้าที่รับผิดชอบวัตถุส่วนประกอบที่ตัวเองเป็นเจ้าของอยู่ด้วย เช่น วัตถุเจ้าของนั้นมีหน้าที่ในการสร้างและทำลายวัตถุส่วนประกอบ

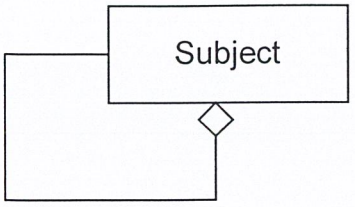


รูปที่ 2-10 ตัวอย่างของ Composition Aggregation

นอกจากนั้นวัตถุส่วนประกอบก็ไม่สามารถมีเจ้าของหลายๆ เจ้าของได้กล่าวง่ายๆ ก็คือวัตถุส่วนประกอบจะมีเจ้าของเพียงวัตถุเดียวเท่านั้น เช่น วัตถุ Window ประกอบด้วย วัตถุ Text, วัตถุ Listbox , วัตถุ Button, วัตถุ Menu เมื่อวัตถุ Window ถูกทำลายทำให้ วัตถุที่เป็นส่วนประกอบ ถูกทำลายไปด้วย

2.4.2.3 Recursive Aggregation

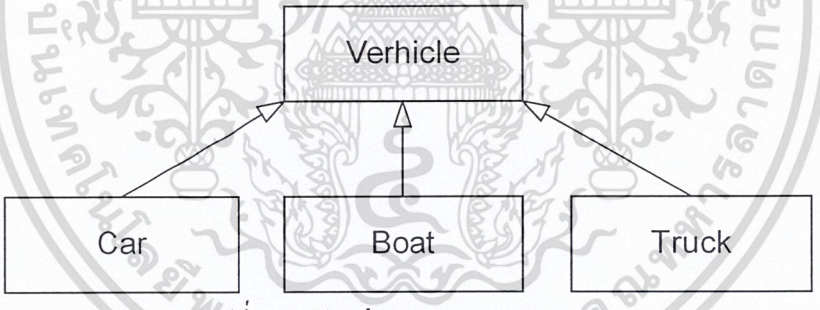
จะเกิดขึ้นก็ต่อเมื่อวัตถุหนึ่งเป็นส่วนหนึ่งของวัตถุที่ถูกสร้างขึ้นมาจากคลาสเดียวกัน ตัวอย่างเช่น วิชาเรียนบางวิชาจำเป็นต้องมีวิชาเรียนก่อนหน้า เราสามารถเขียนให้วัตถุของวิชาเรียนประกอบด้วยวัตถุของวิชาเรียนที่ต้องลงทะเบียนก่อนหน้าไปแล้วได้



รูปที่ 2-11 ตัวอย่างของ Recursive Aggregation

2.4.3 ความสัมพันธ์ในการสืบทอดคุณสมบัติ (Generalization)

เป็นความสัมพันธ์แบบสืบทอดคุณสมบัติ เป็นความสัมพันธ์ระหว่างคลาสที่เกิดขึ้นเมื่อคลาสหนึ่งมีการสืบทอดคุณสมบัติมาจากอีกคลาสหนึ่ง เราสามารถใช้ความสัมพันธ์แบบการสืบทอดคุณสมบัตินี้ในการจัดลำดับขั้นของคลาสได้ โดยคลาสที่อยู่ด้านบนของลำดับชั้นนั้นเราจะเรียกว่าซูเปอร์คลาส (Superclass) ซึ่งเป็นคลาสที่สืบทอดคุณสมบัติและพฤติกรรมกับคลาสที่อยู่ระดับล่างของลำดับชั้นของลำดับชั้นเราเรียกว่าสับคลาส (Subclass) ในการสืบทอดคุณสมบัตินั้นเราสามารถเพิ่มคุณสมบัติให้กับสับคลาสได้ แต่อย่างไรก็ดีการเพิ่มคุณสมบัติให้กับสับคลาสนั้นไม่ได้หมายความว่าซูเปอร์คลาสนั้นเป็น สับเซต (Subset) ของสับคลาส แต่สับคลาสกลับเป็น Subset ของ Superclass ถึงแม้ว่า Subclass จะมีคุณสมบัติมากกว่า Superclass ก็ตาม เช่นสัตว์เลี้ยงลูกด้วยนมเป็นชนิดหนึ่งของสัตว์ หรือ รถ เรือ รถบรรทุก เป็นชนิดหนึ่งของยานพาหนะ



รูปที่ 2-12 ตัวอย่างของ Generalization

บทที่ 3
ฐานข้อมูลเชิงเวลา
(Temporal Database)

3.1 ฐานข้อมูลเชิงเวลา

ในฐานข้อมูลนั้นโดยหลักจะต้องเก็บ Fact ที่เป็นจริงไว้ในฐานข้อมูลเสมอสถานะ (State) เก่า ๆ ของข้อมูลนั้นจะไม่ถูกเก็บไว้ในฐานข้อมูล เกิดว่ามีเหตุการณ์ในการเปลี่ยนแปลง (Update) ข้อมูล (Data) ในแต่ละแถว (Tuple) เกิดขึ้น โดยหลักการเดิมก็จะทำการแก้ไขข้อมูลใหม่ที่ทำการเปลี่ยนแปลงโดยจะเขียนข้อมูลใหม่นั้นทับลงไปในแถว (Row) ที่ทำการเปลี่ยนแปลงทำให้ข้อมูลเดิมเราก็ไม่สามารถไปหาข้อมูลนั้นได้เพราะข้อมูลใหม่ได้ถูกเขียนทับไปแล้วเช่น นาย เอ ตอนเข้ามาเรียนปี 1 ได้ลงทะเยียนรหัสวิชา 001 เป็นวิชาคณิตศาสตร์ แต่พอตอนปี 3 ได้มีการเปลี่ยนวิชา 001 เป็นภาษาอังกฤษเมื่อเราทำการเรียกค้นข้อมูลมาดูก็จะทำให้เราไม่ได้รับข้อมูลที่เคยเป็นจริงในอดีต ก็คือ รหัสวิชา 001 ที่เคยลงทะเบียนเป็นคณิตศาสตร์ ได้หายไปซึ่งรหัส 001 ได้กลายเป็นภาษาอังกฤษ ทำให้เราได้ข้อมูลที่ไม่ถูกต้องเนื่องจากแต่เดิม นาย เอ ได้เคยผ่านการเรียนวิชา คณิตศาสตร์ มาแล้วกลายเป็นว่าข้อมูลใหม่นั้น นาย เอ ไม่ได้เรียนวิชาคณิตศาสตร์

ในส่วนของเวลาที่เกี่ยวข้องกับฐานข้อมูลเชิงเวลานั้น เราจำแนกเป็น 3 อย่างคือ Valid time, Transaction Time, และ User Defined Time

3.1.1 **Valid Time** เวลาที่ข้อมูลเป็นจริง ซึ่งจะแบ่งแยกย่อยออกเป็น
- Valid Time Start คือเวลาเริ่มต้นที่ข้อมูลนั้นเป็นจริง
- Valid Time End คือเวลาสุดท้ายที่ข้อมูลนั้นเป็นจริง

3.1.2 **Transaction Time** คือ เวลาที่ข้อมูลถูกเก็บใน Database โดยสามารถแบ่งออกเป็น
- Transaction Time Start คือเวลาที่เริ่มจัดเก็บข้อมูลลงใน ฐานข้อมูล
- Transaction Time End คือเวลาที่แก้ไขหรือลบข้อมูล

3.1.3 **User Defined Time** คือเวลาที่ผู้ใช้งานสร้างขึ้นเพื่อใช้งาน ซึ่งผู้ใช้งานต้องจัดการและเรียกค้นเอง ระบบฐานข้อมูลจะไม่จัดการให้อย่างอัตโนมัติ หรือกล่าวอีกนัยหนึ่งได้ว่าเวลาชนิดนี้จะเป็นเวลาที่เป็นจริงเสมอ เช่น วันเกิด วันเข้าทำงาน เป็นต้น

สำหรับตัวอย่างของ Valid Time และ Transaction Time เช่น ถ้านาย รักไทย ไทยแท้ เป็นนักศึกษาตั้งแต่วันที่ 1 มิถุนายน พ.ศ. 2540 ถึงวันที่ 30 พฤษภาคม พ.ศ. 2544 แต่ข้อมูลนี้ถูกป้อนเข้าฐานข้อมูลในวันที่ 15 มิถุนายน พ.ศ. 2540 และถูกลบออกจากฐานข้อมูลในวันที่ 25 มิถุนายน พ.ศ. 2544 เราสามารถแจกแจงเวลาได้ดังนี้

- Valid Time Start คือ วันที่ 1 มิถุนายน พ.ศ. 2540

- Valid Time End คือ วันที่ 30 พฤษภาคม พ.ศ. 2544
- Transaction Time Start คือ วันที่ 15 มิถุนายน พ.ศ. 2540
- Transaction Time End คือวันที่ 25 มิถุนายน พ.ศ. 2544

สังเกตว่าช่วงเวลา 3 ช่วงนี้ไม่จำเป็นต้องเป็นช่วงเวลาเดียวกัน ตัวอย่างเช่นสมมติว่ามี Temporal Database ที่เก็บข้อมูลของศตวรรษที่ 20 Valid Time ของข้อมูลคือช่วงเวลาใด ๆ ระหว่างปี 1900 – 1999 ส่วน Transaction time จะเป็นเวลาที่เราเก็บข้อมูลเข้าไปในฐานข้อมูล ถ้า Valid Time Start เป็นอดีตแต่ข้อมูลที่ใส่เข้าไปยังไม่ทราบ Valid Time End เราจึงกำหนดให้ Valid Time End เป็น NOW และให้ Valid Time End เป็น NULL เมื่อ Valid Time Start เป็น อนาคต

$$T_E := \text{NOW}, \text{ if } T_S \leq \text{NOW}$$

$$T_E := \text{NULL}, \text{ if } T_S > \text{NOW}$$

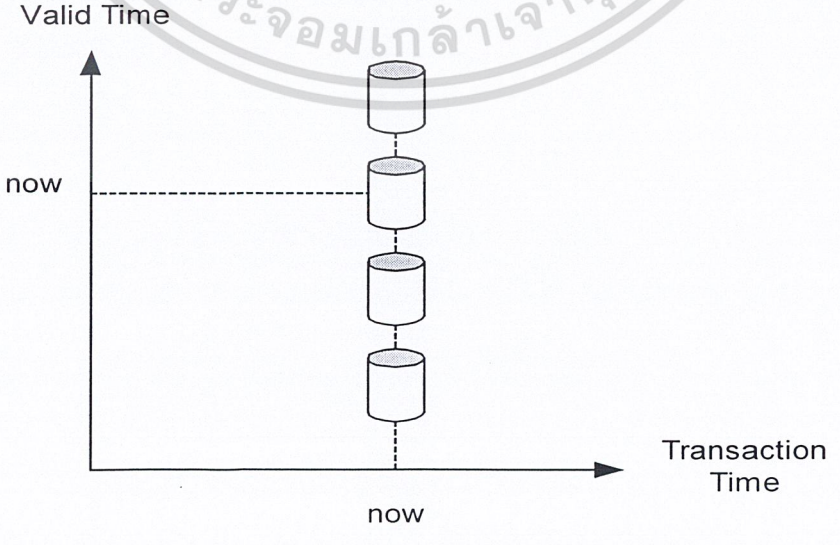
จะเห็นได้ว่า ช่วงเวลา Valid Time และช่วงเวลา Transaction Time ไม่จำเป็นต้องเป็นช่วงเวลาเดียวกันสำหรับข้อมูลตัวเดียวกัน ตัวอย่างเช่น ฐานข้อมูลเก็บชื่อของพนักงานคนหนึ่งไว้ตั้งแต่ ปี 2540 และพนักงานคนนี้เปลี่ยนแปลงชื่อในปี 2543 แต่มาตรวจพบว่ามีชื่อการเปลี่ยนแปลงในปี 2544 จะทำให้ Valid Time กับ Transaction Time เป็น 2543 และ 2544 ตามลำดับ ซึ่งมีค่าไม่เท่ากัน

3.2 ประเภทของฐานข้อมูลเชิงเวลา

ช่วงเวลา Valid Time และ Transaction Time ทำให้เกิดฐานข้อมูลในหลายประเภทดังนี้

3.2.1 ฐานข้อมูลวาลิดไทม์ (Valid-Time Database)

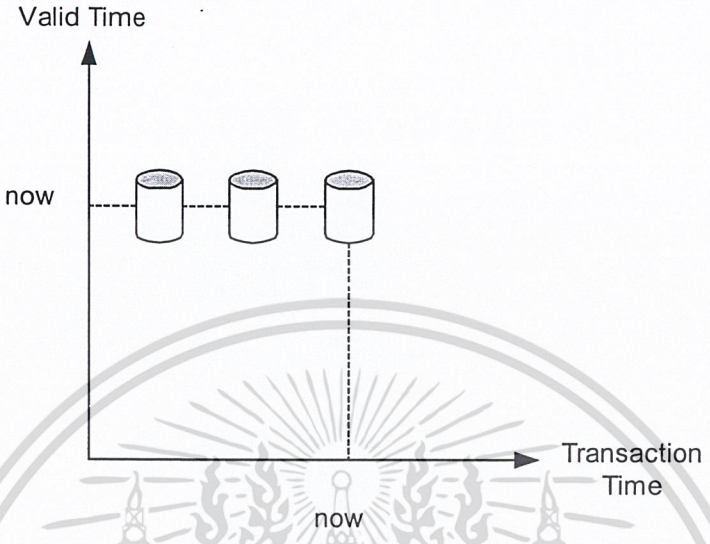
ฐานข้อมูลประวัติ (Historical Database) จะเก็บเฉพาะวาลิดไทม์ ซึ่งจะเก็บข้อมูลในอดีต กับปัจจุบัน ดังแสดงในรูป 3-1



รูปที่ 3-1 ฐานข้อมูลประวัติ

3.2.2 ฐานข้อมูลทรานส์แซคชันไทม์ (Transaction-Time Database)

ฐานข้อมูลย้อนกลับ (Rollback Database) ซึ่งจะเก็บเฉพาะทรานส์แซคชันไทม์แสดงดังรูปที่ 3-2

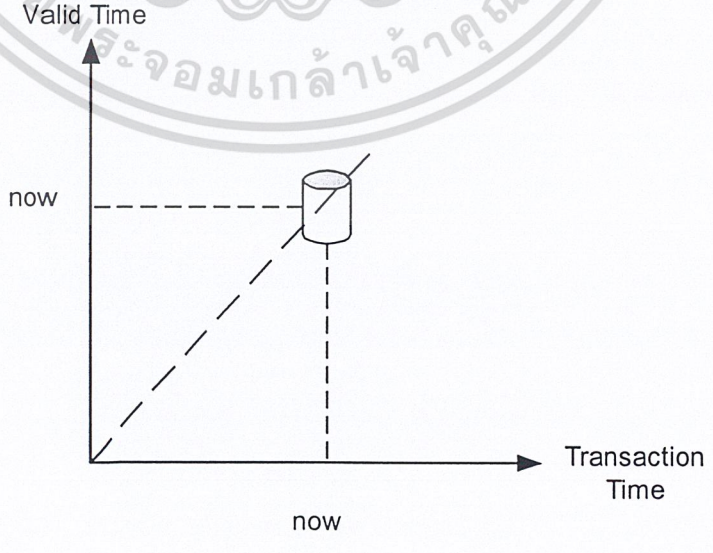


รูปที่ 3-2 ฐานข้อมูลย้อนกลับ

3.2.3 ฐานข้อมูลไบเทมโพรอล (Bitemporal Database)

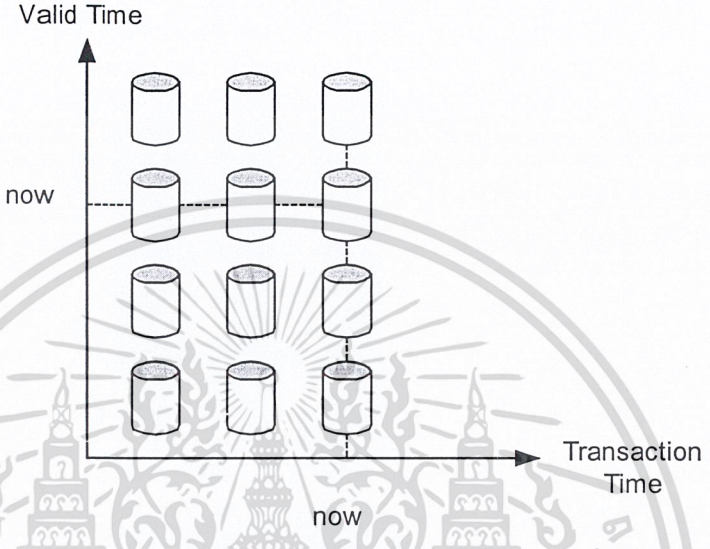
เก็บทั้งวาลิดไทม์ และทรานส์แซคชันไทม์ จะทำให้เก็บข้อมูลในอดีต, อนาคตและปัจจุบันได้

ในระบบจัดการฐานข้อมูลเชิงพาณิชย์จะเก็บเฉพาะข้อมูลที่เป็นจริงในปัจจุบันเท่านั้น ฐานข้อมูลชนิดนี้จะเรียกว่าฐานข้อมูลสแน็ปช็อท (Snapshot Database) โดยฐานข้อมูลสแน็ปช็อทในแบบของ วาลิดไทม์และทรานส์แซคชันไทม์ แสดงดังรูปที่ 3-3



รูปที่ 3-3 ฐานข้อมูลสแน็ปช็อท

และฐานข้อมูลแบบไบเทมโพรอล อาจจะเป็นได้ทั้งตารางสแน็ปช็อท คือเก็บตารางวาลิดไทม์ คือ เก็บว่าข้อมูลเป็นจริงเมื่อใด โดยมีลักษณะฐานข้อมูล หรือเก็บตารางทรานส์แซคชันไทม์ คือ เก็บว่าข้อมูลถูกบันทึกลงในฐานข้อมูลเมื่อใด หรืออาจจะเป็นตารางไบเทมโพรอล ที่เก็บทั้ง วาลิดไทม์และทรานส์แซคชันไทม์ ก็ได้ โดยมีลักษณะของฐานข้อมูลดังรูปที่ 3-4



รูปที่ 3-4 ฐานข้อมูลไบเทมโพรอล

บทที่ 4
ฐานข้อมูลเชิงวัตถุ
(Object-Oriented Database)

4.1 แนวความคิดเชิงวัตถุ (Object-Oriented Conceptual)

รายละเอียดหลักการต่างๆ ของแนวคิดเชิงวัตถุ มีดังนี้

4.1.1 ออบเจ็กต์ (Object)

ออบเจ็กต์ คือ Abstract Representation ของ Real World Entity ที่มี

1. Unique Identity : ออบเจ็กต์ต้องมีสิ่งที่ระบุถึงความแตกต่างจากออบเจ็กต์อื่น
2. Embedded Properties : ออบเจ็กต์ต้องมีคุณสมบัติของการฝัง
3. ความสามารถในการติดต่อสื่อสารกับออบเจ็กต์อื่นและตัวเอง

4.1.2 รหัสเฉพาะของออบเจ็กต์ (Object Identity : OID)

รหัสเฉพาะของออบเจ็กต์ เป็นสิ่งที่ใช้ระบุหรืออ้างอิงออบเจ็กต์ ซึ่งแต่ละออบเจ็กต์จะมี OID ไม่ซ้ำกัน OID จะถูกกำหนดให้โดยระบบตั้งแต่เมื่อออบเจ็กต์นั้นถูกสร้างขึ้นและไม่สามารถเปลี่ยนแปลงได้

OID จะแตกต่างจากคีย์หลักของระบบฐานข้อมูลเชิงสัมพันธ์ เพราะ คีย์หลักจะขึ้นอยู่กับค่าของแอตทริบิวท์ผู้ใช้กำหนด ซึ่งสามารถเปลี่ยนแปลงได้ในภายหลัง แต่ OID ถูกกำหนดโดยระบบซึ่งไม่ขึ้นกับค่าแอตทริบิวท์ของออบเจ็กต์ และเปลี่ยนแปลงไม่ได้ เมื่อออบเจ็กต์ถูกลบไปจากระบบ OID ของออบเจ็กต์นั้นก็จะไม่นำกลับมาใช้ใหม่ ค่าของ OID จะไม่ผูกติดกับ Physical Address ของหน่วยความจำถาวร ทำให้ระบบเชิงวัตถุ เป็น Physical Data Independence

4.1.3 แอตทริบิวท์ (Attribute or Instance Variable)

แอตทริบิวท์ (Attribute or Instance Variable) ก็คือข้อมูลของออบเจ็กต์ อาจเป็นได้ทั้งข้อมูลชนิดพื้นฐานหรือเป็นออบเจ็กต์ก็ได้

4.1.4 สถานะของออบเจ็กต์ (Object State)

สถานะของออบเจ็กต์ขึ้นอยู่กับค่าของแอตทริบิวท์ของออบเจ็กต์ ณ เวลาที่กำหนด ถึงแม้ว่าสถานะของออบเจ็กต์จะเปลี่ยนแปลงไปแต่ OID ยังคงเดิม ถ้าต้องการเปลี่ยนสถานะของออบเจ็กต์ต้องเปลี่ยนค่าของแอตทริบิวต์ โดยจะต้องส่งเมสเซจ (Message) ไปยังออบเจ็กต์ แล้วเมสเซจที่ส่งไปนี้จะเรียกเมธอดที่เกี่ยวข้องให้ทำงาน

4.1.5 เมสเซจและเมธอด (Message and Method)

ตามคุณสมบัติของเอนแคปซูลเลชั่น (Encapsulation) การที่จะกระทำการใดๆ กับออบเจ็กต์ต้องกระทำผ่านเมธอดเท่านั้น เมธอดจะถูกใช้หรือเรียกดูหรือเปลี่ยนค่าแอตทริบิวท์ของออบเจ็กต์ ในการเรียนใช้เมธอดนั้นจะต้องส่งเมสเซจไปยังออบเจ็กต์ เมสเซจที่ส่งจะต้องระบุออบเจ็กต์ที่จะรับเมสเซจ ชื่อเมธอดและพารามิเตอร์ที่เกี่ยวข้อง

4.1.6 การรวมกันของข้อมูลและการจัดการข้อมูล (Encapsulation)

ออบเจ็กต์จะเป็นการรวมเอาแอตทริบิวท์และเมธอดไว้ด้วยกัน การจะเข้าถึงแอตทริบิวต์ต้องทำผ่านเมธอดเท่านั้น ซึ่งจะเป็นการปกป้องข้อมูลของออบเจ็กต์

4.1.7 คลาส (Class)

ในระบบเชิงวัตถุจะนำออบเจ็กต์ที่มีคุณสมบัติเหมือนกันเข้าไว้ด้วยกันเป็นคลาส หรืออาจพูดอีกอย่างได้ว่า คลาสก็คือ Collection ของออบเจ็กต์ที่มีคุณสมบัติเหมือนกันและใช่แอตทริบิวท์และเมธอดร่วมกัน

4.1.8 การสืบทอดคุณสมบัติ (Inheritance)

การสืบทอดคุณสบบัติเป็นการสร้างคลาสใหม่ (Subclass) โดยสืบทอดแอตทริบิวท์และ เมธอดจากคลาสที่มีอยู่แล้ว (Superclass) นอกจากนี้ยังสามารถเพิ่มเติมรายละเอียดให้สับคลาส (Subclass) ได้อีกด้วย การสืบทอดคุณสมบัตินี้มีประโยชน์มากในเรื่องการนำกลับมาใช้ใหม่

4.1.9 Method Overriding and Polymorphism

Method Overriding คือการที่ Subclass สามารถสร้างเมธอดที่มีชื่อซ้ำกับเมธอดที่สืบทอดคุณสมบัติมาจาก Superclass ได้ เพื่อทำให้ Subclass นั้นมีความสามารถเฉพาะเจาะจงมากขึ้น

Polymorphism เป็นการทำให้ออบเจ็กต์ที่ต่างกันสามารถตอบสนองต่อเมสเซจเดียวกันได้ในหลายๆ วิธีการ

4.1.10 Abstract Data Type

Abstract Data Type (ADT) เป็นคุณสมบัติที่ใช้สร้างชนิดของข้อมูลใหม่ขึ้นมา โดยกำหนดโครงสร้างข้อมูลและโอเปอร์เรชันที่ใช้จัดการข้อมูลขึ้นมาจากข้อมูลพื้นฐาน

จะสังเกตได้ว่า ADT และคลาสมีความหมายใกล้เคียงกัน แต่ในระบบเชิงวัตถุคำสองคำนี้มีความหมายต่างกัน โดย Type จะหมายถึงโครงสร้างข้อมูลและเมธอดของคลาส และคลาสหมายถึง Collection ของ Object Instance เมื่อกำหนดคลาสใหม่นี้ขึ้นมาก็เป็นการกำหนด Type ใหม่ด้วย Type ที่กำหนดขึ้นจะถูกใช้เป็นต้นแบบในการสร้างออบเจ็กต์ใหม่ ซึ่งจะถูกจัดการโดยคลาสขณะ run time

ด้วยคุณสมบัติของ ADT และ Inheritance จะสนับสนุนให้มี Complex Object โดย Complex Object ถูกสร้างขึ้นโดยนำออบเจ็กต์อื่นเข้ามารวมไว้ด้วยในรูปของ Set of Complex Relation

4.2 คุณลักษณะของ Object-Oriented Data Model (OODM)

OODM อย่างน้อยที่สุดควรมีลักษณะดังนี้

1. ต้องสนับสนุนการทำ Complex Object
2. Extensible : ต้องมีความสามารถในการกำหนด Data Type และ Operation ที่เกี่ยวข้องขึ้นใหม่ได้
3. ต้องสนับสนุน Encapsulation : รูปแบบของข้อมูลและการจัดการของเมธอดต้องถูกซ่อนจากภายนอก
4. ต้องมีคุณสมบัติ Inheritance : ออบเจ็กต์ต้องสามารถสืบทอดคุณสมบัติ (ข้อมูลและ เมธอด) จากออบเจ็กต์อื่นได้
5. ต้องสนับสนุน Object Identity (OID)

4.3 ระบบจัดการฐานข้อมูลเชิงวัตถุ (Object-Oriented Database Management System: OODBMS)

OODBMS จะทำหน้าที่จัดการกับฐานข้อมูลเชิงวัตถุ โดยใช้คุณสมบัติบางส่วนตามแนวคิดเชิงวัตถุ (OO Concept) และ OODM ที่เป็นเพียงบางส่วน ทั้งนี้เพราะยังไม่มีมาตรฐานที่ใช้กำหนดถึงคุณสมบัติของ OODBMS ที่ต้องสามารถทำได้ ทำให้ OODBMS ที่มีอยู่มีคุณสมบัติแตกต่างกัน อย่างไรดี OODBMS ก็มีคุณสมบัติที่จะเป็นต้องมีอยู่ 13 ข้อด้วยกัน ซึ่งสามารถแบ่งได้เป็น 2 ส่วนดังนี้

ส่วนที่ทำให้เป็น OO System

1. ระบบต้องสนับสนุน Complex Object
2. ระบบต้องสนับสนุน Object Identity
3. ออบเจ็กต์ต้องถูก Encapsulation
4. ระบบต้องสนับสนุน Type และ Class
5. ระบบต้องสนับสนุนการสืบทอดคุณสมบัติ (Inheritance)
6. ระบบต้องหลีกเลี่ยง Premature Binding
7. ระบบต้องเป็น Computation Complete
8. ระบบต้อง Extensible

ส่วนที่ทำให้เป็น DBMS

9. ระบบต้องสามารถจดจำตำแหน่งของข้อมูลได้
10. ระบบต้องสามารถจัดกับฐานข้อมูลที่มีขนาดใหญ่ได้
11. สนับสนุนการทำงานแบบ Concurrency
12. ต้องสามารถกอบกู้ระบบ (Recovery) จากการทำงานที่ผิดพลาดของ Hardware และ Software ได้ ต้องสามารถทำการค้นหาข้อมูลได้ง่าย

บทที่ 5

ฐานข้อมูลเชิงเวลาบนฐานข้อมูลเชิงวัตถุ

(Temporal Object-Oriented Database)

5.1 การนำ OID มาใช้

Object Identity (OID) เป็นสิ่งที่ใช้ระบุหรืออ้างอิงออบเจ็กต์ ซึ่งแต่ละออบเจ็กต์จะมี OID ไม่ซ้ำกัน OID จะถูกกำหนดให้โดยระบบตั้งแต่เมื่อออบเจ็กต์ถูกสร้างขึ้นและจะไม่สามารถเปลี่ยนได้

ปัญหาของการพัฒนาฐานข้อมูลเชิงเวลาบนระบบเชิงสัมพันธ์เปลี่ยน เพราะใน Relational Model จะใช้ External Identifier นี้อาจมีการเปลี่ยนแปลงได้ เช่น ในบางครั้งรหัสสินค้าอาจเปลี่ยนได้

แต่ในระบบที่เป็นเชิงวัตถุทุกออบเจ็กต์จะมี OID เป็นของตัวเอง ซึ่งตัวระบบจะกำหนดให้เมื่อออบเจ็กต์นั้นถูกสร้างขึ้นมา และ OID นี้จะไม่ซ้ำกันอย่างแน่นอน เรียก OID นี้ว่าเป็น Internal Identifier ซึ่งไม่มีส่วนเกี่ยวข้องใดๆ ทั้งสิ้นกับข้อมูล ไม่ว่าข้อมูลจะเปลี่ยนไปอย่างไร OID ก็จะเป็นค่าเดิมสำหรับออบเจ็กต์เดิมเสมอ

ระบบที่เป็นเชิงวัตถุซึ่งมี OID นี้สามารถช่วยแก้ปัญหาเรื่อง Identifier เปลี่ยนของระบบที่เป็นเชิงสัมพันธ์ได้ เพราะแม้ว่า External Identifier เช่น รหัสวิชา จะเปลี่ยนไป แต่ Internal Identifier หรือ OID ก็จะคงเดิมเสมอ

5.2 Extension for Time in Database

ในการประยุกต์ Temporal Application นั้นสิ่งที่ Non-Temporal Database System ต้องสนับสนุนมี 3 ประการคือ

5.2.1 Data structure จะต้องสามารถเก็บข้อมูลเกี่ยวกับเวลาได้

การเพิ่ม date/time attribute เข้าใน data structure นั้นสามารถทำได้ง่ายมาก เราอาจเก็บ Valid Time โดยเพิ่ม attribute VTS (Valid Time Start) และ VTE (Valid Time End) ให้กับ class Date ส่วน Transaction Time ก็เช่นเดียวกัน

แต่สำหรับ operation สำหรับ Temporal จะซับซ้อนกว่า เช่นผลต่างของสองช่วงเวลาอาจได้ผลออกมาเป็นช่วงเวลาหลายๆ ช่วง (Set Of Interval) ซึ่งโดยทั่วไปจะใช้ Temporal Element ซึ่งเป็น Set of Interval เข้ามาช่วยเรื่อง Timestamp

สำหรับ Relation Data Model นั้นส่วนที่ซับซ้อนที่สุดคือ Timestamp ของส่วนของ Data Structure อย่างเช่น Tuple หรือ Attribute Timestamp ซึ่งใน Object Data Model อาจทำได้ 3 แนวทางคือ

1. Timestamp ที่ Attribute หรือ Timestamp ในระดับ Object
2. Timestamp ในระดับ Type

3. Timestamp ในระดับ Identifier

5.2.2 Operation ต่างๆ สำหรับ temporal เพื่อการ query และ modify database

การเพิ่ม Temporal Operation เป็นเรื่องที่มีปัญหามากจาก Temporal Algebra ที่ได้มีการนิยามไว้แล้ว เราอาจนำไป Implement โดยตรงกับ System หรือทำเป็น Layer อยู่บน Non-Temporal System ซึ่งก็จำเป็นต้องมีส่วนขยายของ Query Language ให้สามารถใช้ Temporal Operation ร่วมกับส่วนที่เป็น Non-Temporal อื่นๆได้

5.2.3 Temporal constraint

ใน Temporal Relation Database ตัว System ทำหน้าที่จัดการ Temporal constraint อย่างเช่น Referential Integrity ให้ แต่สำหรับ OODBMS จะให้ผู้เขียนโปรแกรมจัดการกับ Temporal constraint ที่เพิ่มขึ้นมาเอง

บทที่ 6

แผนภาพคลาสเชิงเวลา

(Temporal Class Diagram : TCD)

6.1 แผนภาพคลาสเชิงเวลา TCD

แผนภาพคลาสเชิงเวลา TCD นี้ได้นำเอาแผนภาพคลาสใน UML มาเพิ่มขยายโครงสร้างให้สามารถรองรับเวลาของระบบฐานข้อมูลเชิงเวลาได้ โดยเวลาของระบบฐานข้อมูลเชิงเวลาประกอบด้วยเวลา 3 ชนิดคือ

6.1.1 Transaction Time (TT)

คือเวลาที่ข้อมูลอยู่ในฐานข้อมูลและเวลาที่ข้อมูลได้รับการแก้ไข โดยเวลาแบบนี้ระบบจะจัดหาให้ ผู้ใช้ไม่ต้องป้อนเข้าไป ดังนั้นเวลาแบบนี้จะใช้กับข้อมูลที่มีการเปลี่ยนแปลงไม่บ่อยนัก หรือเปลี่ยนแปลงเมื่อมีความเข้าใจผิด เช่น ความสูงของคน น้ำหนักของคน เป็นต้น

6.1.2 Valid Time (VT)

คือเวลาที่ข้อมูลเป็นจริง ซึ่งผู้ใช้งานจะต้องป้อนเวลาที่ข้อมูลนั้นเป็นจริงเข้าไปในระบบ โดยผู้ใช้งานต้องกำหนดเวลาที่ข้อมูลนั้นเป็นจริง เวลาดังกล่าวจะต้องระบุหน่วยของเวลาด้วย เช่น ปี เดือนหรือวัน หรืออาจระบุเป็นช่วงเวลาก็ได้

6.1.3 Bitemporal (BT)

คือการจัดเก็บทั้ง Transaction Time และ Valid Time เข้าด้วยกันสำหรับข้อมูลที่จะใช้กับเวลาประเภทนี้คือข้อมูลที่ต้องการจัดเก็บเวลาที่ข้อมูลเป็นจริง และเวลาที่จัดเก็บข้อมูลหรือเวลาที่ข้อมูลได้รับการแก้ไข

จากนิยามของเวลาทั้งสามแบบนั้น หากผู้ใช้งานจำเป็นที่จะต้องใช้งานเวลากับข้อมูลหรือใช้งานเวลากับความสัมพันธ์ต่างๆ แล้ว ผู้ใช้ต้องเลือกตามความเหมาะสมเอง

สำหรับสัญลักษณ์ของเวลาที่เพิ่มเข้าไปในแผนภาพคลาส นั้นแสดงดังรูป

รูปที่ 6-1 สัญลักษณ์ทางเวลาของแผนภาพคลาสเชิงเวลา TCD

จากรูปเป็นสัญลักษณ์ของเวลาที่นำมาใช้ใน TCD โดยผู้ใช้สามารถเลือกใช้ประเภทของเวลาตามที่ต้องการออกแบบ

6.2 Temporal Data Type

หน่วยของเวลาที่จะนำมาใช้ในโมเดลนี้ แบ่งออกเป็นหลายๆ ชนิดคือ

- Date วัน เดือน ปี
- Time เป็นเวลาที่บ่งชี้ ชั่วโมง นาที และวินาที
- Timestamp เป็นจุดหนึ่งของเวลาประกอบด้วย วัน เดือน ปี ชั่วโมง นาที และวินาที ใช้บอกเวลาเริ่มต้นของข้อมูล (Timestart) และเวลาสิ้นสุดของข้อมูล (Timeend)
- Interval เป็นช่วงเวลาระหว่าง 2 จุดเวลาต่อเนื่องกัน ประกอบด้วย วัน ชั่วโมง นาที และวินาที
- Timepoint นำ Timestamp มาเพิ่มเติมหน่วยของเวลาเข้าไป เพื่อต้องการให้วางเวลาเฉพาะหน่วยที่ต้องการเท่านั้น มิได้วางเวลาตาม Timepoint
- Period ช่วงเวลาระหว่าง 2 จุดเวลา ที่นำ Interval มาเพิ่มเติมหน่วยของเวลาเข้าไป

สำหรับเวลาที่ได้กล่าวมาแล้วนั้น เวลาที่อยู่ในมาตรฐานของ ODMG มีด้วยกัน 4 ประเภท คือ Date, Time, Timestamp และ Interval แต่ Timepoint และ Period ได้มีการสร้างขึ้นมาใหม่เพื่อที่จะใช้กับข้อมูลเชิงเวลาได้

6.3 เวลาที่เพิ่มเข้าไปที่คลาส

เวลาที่เพิ่มเข้าไปที่คลาสนั้น เป็นการจัดเก็บช่วงชีวิตของคลาส ซึ่งหากผู้ใช้งานโมเดลนี้ต้องการให้คลาสใดมีการจัดเก็บช่วงชีวิตของคลาสมีเวลาแบบใดมาเกี่ยวข้อง ผู้ใช้สามารถนำสัญลักษณ์ของเวลาแบบนั้นไปใส่ที่คลาสที่ต้องการ สำหรับการใช้แบบต่างๆ นั้นมีรายละเอียดดังต่อไปนี้

6.3.1 คลาสที่ไม่มีเวลาเข้ามาเกี่ยวข้อง (Static Class)

คลาสประเภทนี้เป็นคลาสที่ออบเจ็กต์ของคลาสนั้นไม่มีเวลาเข้ามาเกี่ยวข้อง เป็นคลาสที่มีการใช้งานตามปกติ ไม่มีการเก็บช่วงชีวิตของออบเจ็กต์ของคลาส จึงไม่ต้องเพิ่มเติมส่วนใดๆ เข้าไปใน UML เลย การกำหนดให้คลาสใดเป็นคลาสที่ไม่มีเวลาเข้ามาเกี่ยวข้องควรพิจารณาจาก สิ่งต่อไปนี้

- ออบเจ็กต์ของคลาสอยู่ในฐานข้อมูลตลอดเวลา
- ออบเจ็กต์ของคลาสนั้นเป็นจริงตลอดไปไม่มีการเปลี่ยนแปลง
- ค่าของแอตทริบิวต์เป็นค่าปัจจุบันในฐานข้อมูลค่าเหล่านั้นเป็นจริงตลอดเวลา

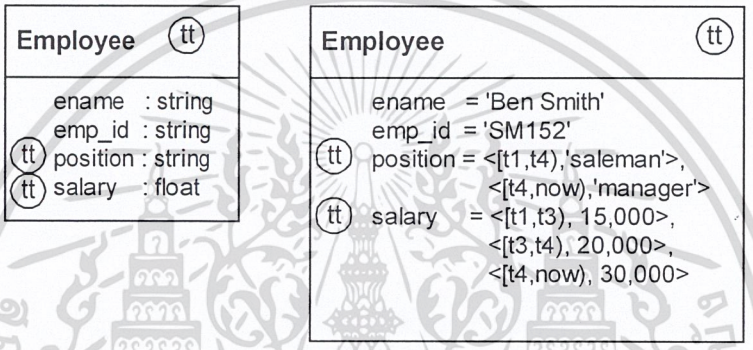
6.3.2 คลาสที่มีการจัดเก็บเวลาที่ออบเจ็กต์อยู่ในฐานข้อมูล (Transaction Time Class)

คลาสประเภทนี้จะมีการเก็บเวลาที่ออบเจ็กต์ของคลาสนั้นอยู่ในฐานข้อมูลหรือเวลาที่ออบเจ็กต์ได้รับการเปลี่ยนแปลง (transaction time lifespan) ไว้ด้วย และแอตทริบิวต์ของคลาสนั้นมีได้เพียง แอตทริบิวต์ที่ไม่มีเวลาเข้ามาเกี่ยวข้อง (Static attribute) หรือแอตทริบิวต์ที่มีการจัดเก็บเวลาที่ข้อมูลนั้นอยู่ในฐานข้อมูล (transaction time class) เท่านั้น

การกำหนดให้คลาสใดเป็นคลาสที่มีการจัดเก็บเวลาที่ออบเจ็กต์อยู่ในฐานข้อมูลควรพิจารณาจากสิ่งต่อไปนี้

- ออบเจ็กต์ของคลาสเป็นจริงตลอดช่วงเวลาที่จัดเก็บ
- ค่าของแอตทริบิวต์ไม่ค่อยมีการเปลี่ยนแปลงหรือการเปลี่ยนแปลงน้อยมาก
- ค่าของแอตทริบิวต์ที่ไม่มีเวลาเข้ามาเกี่ยวข้องเป็นค่าปัจจุบันในฐานข้อมูลตลอดช่วงชีวิตของออบเจ็กต์นั้น

สามารถเขียนคลาสที่มีการจัดเก็บเวลาที่ออบเจ็กต์อยู่ในฐานข้อมูลเป็น TCD โดยเขียนสัญลักษณ์ (tt) ไว้ที่มุมบนด้านขวาของชื่อคลาส ดังรูป 6-2



รูปที่ 6-2 ตัวอย่างของ Transactoin Time class และ Object

จากตัวอย่างในรูปที่ 6-2 จะเห็นว่า Transaction time lifespan ของออบเจ็กต์ คือ [t1,now) นั่นคือ ออบเจ็กต์นี้ถูกสร้างขึ้นที่ time point t1 และอยู่ในฐานข้อมูลเรื่อยมาจนถึงปัจจุบัน ส่วนแอตทริบิวต์ position และ salary เป็นแอตทริบิวต์ที่เก็บเวลาที่ข้อมูลนั้นอยู่ในฐานข้อมูล

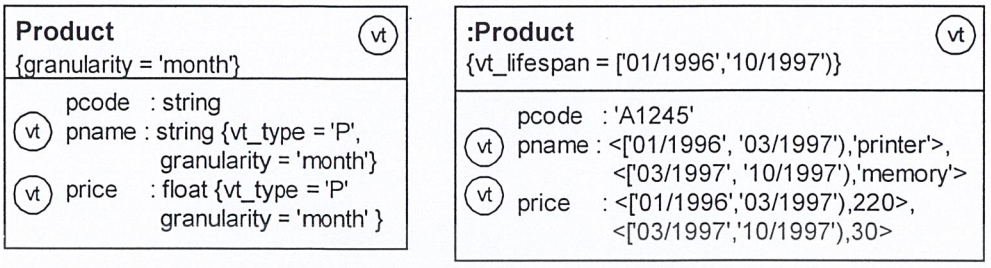
6.3.3 คลาสที่มีการจัดเก็บเวลาที่ออบเจ็กต์นั้นเป็นจริง (Valid Time Class)

คลาสประเภทนี้จะมีการเก็บเวลาที่ออบเจ็กต์ของคลาสนั้นเป็นจริงเข้าไปด้วย โดยผู้ใช้ต้องบอกหน่วยของเวลาเข้าไปด้วย คลาสประเภทนี้จะมีการจัดเก็บช่วงชีวิตที่คลาสนั้นเป็นจริง (valid time lifespan) และผู้ใช้ต้องบอกลักษณะของเวลาที่จัดเก็บ (Time Period) และหน่วยของเวลาที่เล็กที่สุดที่จัดเก็บ (Graualarity) ด้วยตัวเอง แอตทริบิวต์ของคลาสนั้นมีได้เพียง แอตทริบิวต์ที่ไม่มีเวลาเข้ามาเกี่ยวข้อง หรือแอตทริบิวต์ที่มีการจัดเก็บเวลาที่ข้อมูลนั้นเป็นจริงเท่านั้น

การกำหนดให้คลาสใดเป็นคลาสที่มีการจัดเก็บเวลาที่ออบเจ็กต์นั้นเป็นจริง ควรพิจารณาจากสิ่งต่อไปนี้

- ออบเจ็กต์ของคลาสนั้นอยู่ในฐานข้อมูลตั้งแต่เวลาที่ถูกสร้างขึ้นจนถึงปัจจุบัน
- ค่าของแอตทริบิวต์เป็นค่าปัจจุบันในฐานข้อมูลตั้งแต่เวลาที่ถูกสร้างขึ้นจนถึงปัจจุบัน
- ค่าของแอตทริบิวต์ที่ไม่มีเวลาเข้ามาเกี่ยวข้องเป็นจริงตลอดช่วงชีวิตของออบเจ็กต์นั้น

สามารถเขียนคลาสที่มีการจัดเก็บเวลาที่ออบเจ็กต์เป็นจริงเป็น TCD โดยเขียนสัญลักษณ์ (vt) ไว้ที่มุมบนด้านขวาของชื่อคลาส ดังรูป 6-3



รูปที่ 6-3 ตัวอย่างของ Valid Time Class และ Object

จากตัวอย่างในรูปที่ 6-3 จะเห็นว่า Valid time lifespan ของออบเจ็กต์คือ [‘01/1996’,’10/1997’) นั้นคือ ออบเจ็กต์นี้อยู่ในเป็นจริงตลอดช่วงระยะเวลานี้ ส่วนแอตทริบิวต์ pname และ price เป็นแอตทริบิวต์ที่จัดเก็บเวลาที่ข้อมูลนั้นเป็นจริง

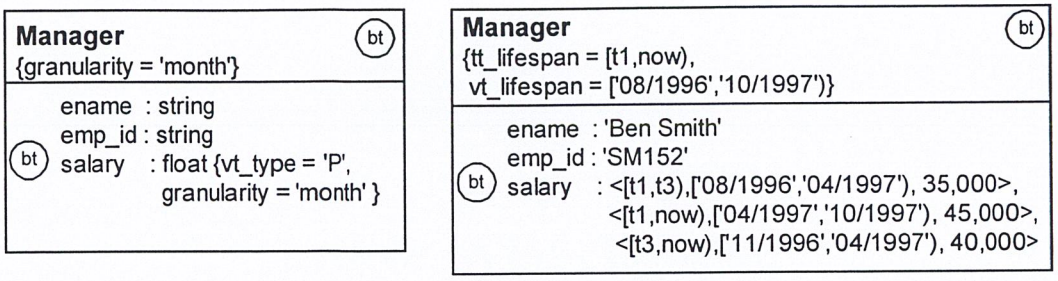
6.3.4 คลาสที่มีการจัดเก็บเวลาที่ออบเจ็กต์เป็นจริง และเวลาที่ออบเจ็กต์นั้นอยู่ในฐานข้อมูล (Bitemporal Time Class)

คลาสประเภทนี้จะจัดเก็บเวลาทั้งสองชนิดลงไปด้วยในฐานข้อมูล เพื่อจัดเก็บทั้งเวลาที่ออบเจ็กต์ของคลาสนั้นเป็นจริง (valid time lifespan) และเวลาที่ออบเจ็กต์นั้นอยู่ในฐานข้อมูลหรือได้รับการเปลี่ยนแปลงแก้ไข (transaction time lifespan) โดยการจัดเก็บช่วงชีวิตที่คลาสนั้นเป็นจริง ผู้ใช้ต้องบอกลักษณะของเวลาที่จัดเก็บ (Time Period) และหน่วยของเวลาที่เล็กที่สุดที่จัดเก็บ (Granularity) ด้วย แอตทริบิวต์ของคลาสนั้นเป็นได้ทั้ง แอตทริบิวต์ที่ไม่มีเวลาเข้ามาเกี่ยวข้อง, แอตทริบิวต์ที่จัดเก็บเวลาที่ข้อมูลนั้นอยู่ในฐานข้อมูล และแอตทริบิวต์ที่จัดเก็บเวลาที่ข้อมูลเป็นจริง

การกำหนดให้คลาสใดเป็นคลาสที่มีการจัดเก็บเวลาที่ออบเจ็กต์เป็นจริง และเวลาที่ออบเจ็กต์นั้นอยู่ในฐานข้อมูล ควรพิจารณาจากสิ่งต่อไปนี้

- ค่าของแอตทริบิวต์ที่ไม่มีเวลาเข้ามาเกี่ยวข้องเป็นค่าปัจจุบันในฐานข้อมูลตลอดช่วงเวลา transaction time lifespan ของออบเจ็กต์และค่าของแอตทริบิวต์นั้นเป็นจริงตลอดช่วงเวลา valid time lifespan ของออบเจ็กต์นั้น
- ค่าของแอตทริบิวต์ที่จัดเก็บเวลาที่ข้อมูลอยู่ในฐานข้อมูล เป็นจริงตลอดช่วงเวลา valid time lifespan ของออบเจ็กต์นั้น
- ค่าของแอตทริบิวต์ที่จัดเก็บเวลาที่ข้อมูลเป็นจริงเป็นค่าปัจจุบันในฐานข้อมูลตลอดช่วงเวลา transaction time lifespan ของออบเจ็กต์นั้น

สามารถเขียนคลาสที่มีการจัดเก็บเวลาที่ออบเจ็กต์เป็นจริง และเวลาที่ออบเจ็กต์นั้นอยู่ในฐานข้อมูลเป็น TCD โดยเขียนสัญลักษณ์ของเวลา bt ไว้ที่มุมบนด้านขวาของชื่อคลาส ดังรูป 6-4



รูปที่ 6-4 ตัวอย่างของ Bitemporal Time Class และ Object

จากตัวอย่างในรูปที่ 6-4 จะเห็นว่า transaction time lifespan ของออบเจ็กต์นี้คือ [t1,now) และ valid time lifespan ของมันคือ ['08/1996','10/1997') โดยส่วนแอตทริบิวต์ salary เป็นแอตทริบิวต์ที่จัดเก็บเวลาที่ข้อมูลเป็นจริงและเวลาที่ข้อมูลอยู่ในฐานข้อมูล

6.4 เวลาที่เพิ่มเข้าไปที่แอตทริบิวต์ของคลาส

เวลาที่เพิ่มเข้าไปที่แอตทริบิวต์ของคลาส เป็นการจัดเก็บเวลาของข้อมูลของแอตทริบิวต์ เพื่อใช้ในการจัดเก็บประวัติของแอตทริบิวต์ ซึ่งรายละเอียดการเพิ่มเวลาเข้าไปที่แอตทริบิวต์ของคลาสได้ดังนี้

6.4.1 แอตทริบิวต์ที่ไม่มีเวลามาเกี่ยวข้อง (Static Attribute)

แอตทริบิวต์ประเภทนี้จะไม่มีรูปแบบของเวลาเข้าไปเกี่ยวข้อง เป็นแอตทริบิวต์ที่ใช้งานกันอยู่ตามปกติ ไม่มีการจัดเก็บสถานะต่างๆ จะจัดเก็บเฉพาะสถานะล่าสุดเท่านั้น โดย Static time Attribute สามารถอยู่ได้ใน Static Class, Valid time Class, Transaction time Class และ Bitemporal time Class

สามารถเขียนแอตทริบิวต์ที่ไม่มีเวลามาเกี่ยวข้องเป็น TCD ได้เลยโดยไม่ต้องเพิ่มเติมส่วนใดๆ เข้าไปในแผนภาพคลาสเลย

ename : string

6.4.2 แอตทริบิวต์ที่จัดเก็บเวลาที่ข้อมูลอยู่ในฐานข้อมูล (Transaction Time Attribute)

แอตทริบิวต์ประเภทนี้จะมีการจัดเก็บเวลาที่ข้อมูลอยู่ในฐานข้อมูล หรือเวลาที่ข้อมูลได้รับการเปลี่ยนแปลงไว้ด้วย สำหรับแอตทริบิวต์ที่เหมาะสมที่จะใช้งานกับเวลาประเภทนี้ก็คือ แอตทริบิวต์ที่ไม่ค่อยมีการเปลี่ยนแปลงมากนัก โดย Transac time Attribute สามารถอยู่ได้ใน Transaction time Class และ Bitemporal time Class

สามารถเขียนแอตทริบิวต์ที่จัดเก็บเวลาที่ข้อมูลอยู่ในฐานข้อมูลเป็น TCD โดยเขียนสัญลักษณ์ (tt) ไว้หน้าแอตทริบิวต์นั้นได้ดังรูป

(tt) position : string

6.4.3 แอตทริบิวต์ที่จัดเก็บเวลาที่ข้อมูลเป็นจริง (Valid Time Attribute)

แอตทริบิวต์ประเภทนี้จะเป็นแอตทริบิวต์ที่ผู้ใช้ต้องป้อนเวลาที่ต้องการให้ข้อมูลนั้นๆ เป็นจริง เข้าไปด้วย ซึ่งเวลาที่กล่าวถึงนี้เป็นเวลาที่ข้อมูลนั้นๆ เป็นจริง โดย Valid time Attribute สามารถอยู่ได้ใน Valid Time Class และ Bitemporal Time Class

สามารถเขียนแอตทริบิวต์ที่เก็บเวลาที่ข้อมูลเป็นจริงเป็น TCD โดยเขียนสัญลักษณ์ (vt) ไว้หน้าแอตทริบิวต์นั้นได้ดังรูป

(vt) price : float {vt_type = 'P', granularity = 'month'}

6.4.4 แอตทริบิวต์ที่จัดเก็บทั้งเวลาที่ข้อมูลเป็นจริง และเวลาที่ข้อมูลอยู่ในฐานข้อมูล (Bitemporal Time Attribute)

แอตทริบิวต์ประเภทนี้จะจัดเก็บเวลาทั้งสองชนิดของข้อมูลลงไปด้วย เพื่อให้การสืบค้นได้ค่าที่ถูกต้องมากขึ้น เนื่องจากการจัดเก็บเวลาที่ข้อมูลเป็นจริงนั้น อาจไม่ทราบว่าบันทึกไว้เมื่อใด และมีการเปลี่ยนแปลงข้อมูลอย่างไร จะทราบเพียงว่าความจริงนั้นเมื่อใดเท่านั้น ส่วนหากจัดเก็บเฉพาะเวลาข้อมูลนั้นอยู่ในฐานข้อมูล หรือเวลาที่ข้อมูลนั้นมีการเปลี่ยนแปลงนั้น ก็จะไม่ทราบเหมือนกันว่าเวลาที่ข้อมูลนั้นเป็นจริงเมื่อใด

สามารถเขียนแอตทริบิวต์ที่จัดเก็บทั้งเวลาที่ข้อมูลเป็นจริง และเวลาที่ข้อมูลอยู่ในฐานข้อมูล เป็น TCD ได้โดยเขียนสัญลักษณ์ ไว้หน้าแอตทริบิวต์นั้น ได้ดังรูป

(bt) salary : float {vt_type = 'P', granularity = 'month' }

6.5 เวลาที่เพิ่มเข้าไปในความสัมพันธ์ระหว่างคลาส

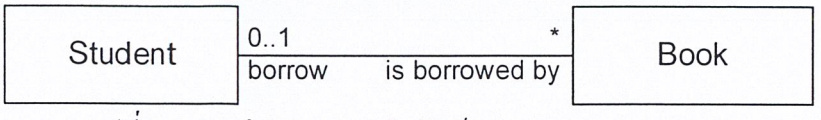
เวลาที่เพิ่มเข้าไปในความสัมพันธ์ระหว่างคลาสนี้จะเป็นเวลาของความสัมพันธ์ ซึ่งหากผู้ใช้งานโมเดลนี้ต้องการให้ความสัมพันธ์ของคลาสมีเวลาแบบใดมาเกี่ยวข้อง ผู้ใช้สามารถนำสัญลักษณ์ของเวลาแบบนั้นๆ ไปใส่ไว้ยังความสัมพันธ์ที่ต้องการ สำหรับการใช้เวลาแบบต่างๆ นั้นรายละเอียดมีดังต่อไปนี้

6.5.1 ความสัมพันธ์ที่ไม่มีเวลาเข้ามาเกี่ยวข้อง (Static Relationship)

ความสัมพันธ์นี้ก็คือความสัมพันธ์เดิม ๆ ที่ใช้งานกันตามปกติ โดยความสัมพันธ์นี้จะไม่มีเวลามาเกี่ยวข้อง เช่น ในระบบห้องสมุด นักศึกษามีความสัมพันธ์กับหนังสือ โดยนักศึกษาคนหนึ่งสามารถยืมหนังสือได้หลายเล่ม แต่หนังสือเล่มหนึ่งจะมีนักศึกษาเพียงคนเดียวเท่านั้นที่สามารถยืมได้ ในเวลาหนึ่ง นั่นคือความสัมพันธ์แบบหนึ่งต่อหลาย (One to many) นั่นเอง ความสัมพันธ์นี้เป็นความสัมพันธ์ที่ใช้งานกันอยู่ทั่วไป ไม่มีส่วนใดเพิ่มเติมจากเดิม โดยความสัมพันธ์แบบนี้จะสามารถเป็นความสัมพันธ์ระหว่าง Static Class, Valid Time Class, Temporal Time Class หรือ Bitemporal Time Class ก็ได้

ดังนั้นหากผู้ใช้งานต้องการตอบคำถามประเภทที่ถามประวัติการยืมหนังสือของนักศึกษา เช่น ต้องการทราบว่านักศึกษาคนใดเคยยืมหนังสือรหัส IT001 บ้าง ผู้ใช้งานจะใช้งานความสัมพันธ์ชนิดนี้ไม่ได้ ความสัมพันธ์ชนิดนี้จะจัดเก็บข้อมูล State ล่าสุดเท่านั้น ไม่สามารถจัดเก็บประวัติได้

สามารถเขียนความสัมพันธ์ที่ไม่มีเวลาเข้ามาเกี่ยวข้องเป็น TCD ได้ดังรูป



รูปที่ 6-5 ตัวอย่างของความสัมพันธ์แบบ Static Relationship

6.5.2 ความสัมพันธ์ที่จัดเก็บเวลาที่ความสัมพันธ์นั้นอยู่ในฐานข้อมูล (Transaction Time Relationship)

ความสัมพันธ์ประเภทนี้จะจัดเก็บประวัติความสัมพันธ์ที่มีต่อกัน โดยจัดเก็บความสัมพันธ์ในหลายๆ ช่วงเวลา ซึ่งผู้ใช้งานสามารถเรียกค้นประวัติเหล่านี้ตามช่วงเวลาที่ต้องการได้ สำหรับเวลาที่จัดเก็บระบบจะใช้เวลาของระบบเอง โดยหลักการในการเลือกใช้ความสัมพันธ์แบบนี้ควรเลือกความสัมพันธ์ ที่มีการเปลี่ยนแปลงไม่บ่อยนัก เช่น จากตัวอย่างระบบห้องสมุด เมื่อมีการจัดเก็บเวลาของความสัมพันธ์ ก็จะทำให้สามารถตอบคำถามที่ต้องการประวัติของการใช้งานได้ โดยความสัมพันธ์แบบนี้จะสามารถเป็นความสัมพันธ์ระหว่าง Transaction Time Class หรือ Bitemporal Time Class ก็ได้

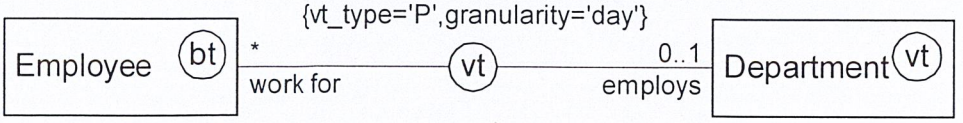
สามารถเขียนความสัมพันธ์แบบมีเวลาที่ความสัมพันธ์อยู่ในฐานข้อมูลเป็น TCD ได้ดังรูป

รูปที่ 6-6 ตัวอย่างของความสัมพันธ์แบบ Transaction Time Relationship

6.5.3 ความสัมพันธ์ที่จัดเก็บเวลาที่ความสัมพันธ์นั้นเป็นจริง (Valid Time Relationship)

ความสัมพันธ์ประเภทนี้จะเหมือนกันความสัมพันธ์ที่จัดเก็บเวลาที่ความสัมพันธ์นั้นอยู่ในฐานข้อมูล เพียงแต่ว่าเวลาที่จัดเก็บสำหรับความสัมพันธ์นั้น เป็นเวลาที่ผู้ใช้งานป้อนเข้าไปเอง ดังนั้นความสัมพันธ์แบบนี้ผู้ใช้งานสามารถวางนโยบายในการเปลี่ยนแปลงเวลาได้ ตัวอย่างเช่น ในบริษัทแต่ละบริษัทได้แบ่งออกเป็นแผนกต่างๆ ดังนั้นพนักงานจะสังกัดแผนกได้หนึ่งแผนกเท่านั้น แต่แผนกหนึ่งจะมีพนักงานได้หลายคน โดยความสัมพันธ์แบบนี้จะสามารถเป็นความสัมพันธ์ระหว่าง Valid Time Class หรือ Bitemporal Time Class ก็ได้

สามารถเขียนความสัมพันธ์ที่จัดเก็บเวลาที่ความสัมพันธ์เป็นจริง เป็น TCD ได้ดังรูป

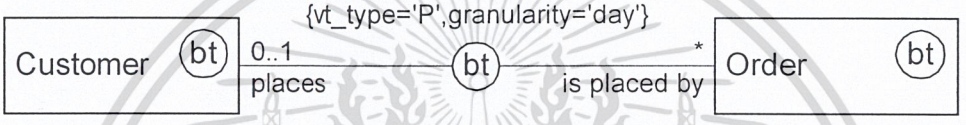


รูปที่ 6-7 ตัวอย่างของความสัมพันธ์แบบ Valid Time Relationship

6.5.4 ความสัมพันธ์ที่จัดเก็บทั้งเวลาที่ความสัมพันธ์เป็นจริง และเวลาที่ความสัมพันธ์นั้นอยู่ในฐานข้อมูล (Bitemporal Time Relationship)

ความสัมพันธ์แบบนี้จะจัดเก็บเวลาทั้งสองลงในฐานข้อมูล เพื่อจัดเก็บทั้งเวลาที่ความสัมพันธ์นั้นเป็นจริง และเวลาที่จัดเก็บหรือเวลาที่ความสัมพันธ์นั้นๆ เปลี่ยนแปลง ดังนั้นจึงสามารถตอบคำถามประเภทที่ต้องการทราบประวัติของความสัมพันธ์และคำถามที่ต้องการทราบช่วงเวลาในการบันทึกความสัมพันธ์ได้อีกด้วย โดยความสัมพันธ์แบบนี้จะสามารถเป็นความสัมพันธ์ระหว่าง Bitemporal Time Class ได้เพียงอย่างเดียว

สามารถเขียนความสัมพันธ์ที่จัดเก็บทั้งเวลาที่ความสัมพันธ์เป็นจริง และเวลาที่ความสัมพันธ์นั้นอยู่ในฐานข้อมูล เป็น TCD ได้ดังรูป

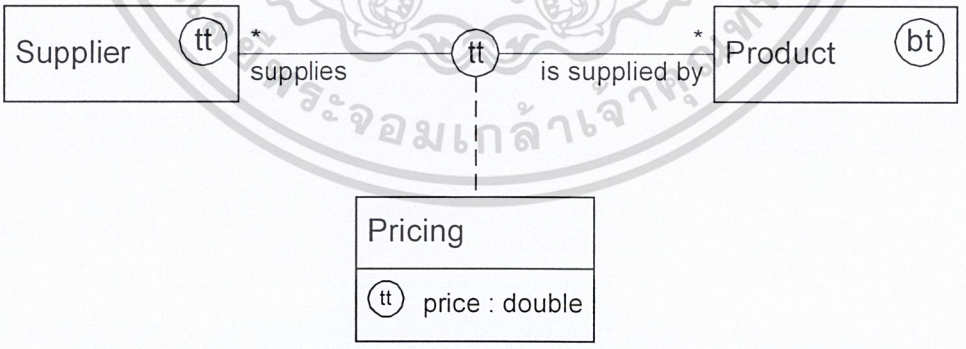


รูปที่ 6-8 ตัวอย่างของ Bitemporal Time Relationship

6.5.5 ความสัมพันธ์แบบ Association Class

ความสัมพันธ์แบบนี้เป็นความสัมพันธ์ที่มีการเก็บเวลาของความสัมพันธ์ที่เป็นแบบ Association Class ซึ่งสามารถจัดเก็บเวลาได้ทั้ง 3 แบบตามที่ต้องการ

สามารถเขียนความสัมพันธ์แบบ Association Class เป็น TCD ได้ดังรูป

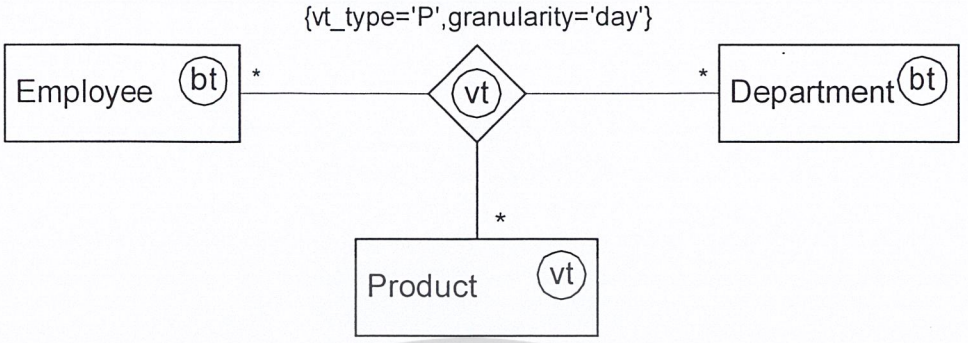


รูปที่ 6-9 ตัวอย่างของความสัมพันธ์แบบ Association Class

6.5.6 ความสัมพันธ์แบบหลายบทบาท (N-ary Relationship)

ความสัมพันธ์แบบนี้เป็นความสัมพันธ์ที่มีมากกว่า 2 บทบาทขึ้นไป ซึ่งสามารถจัดเก็บเวลาของความสัมพันธ์ได้ทั้ง 3 แบบ

สามารถเขียนความสัมพันธ์แบบหลายบทบาทเป็น TCD ได้ดังรูป



รูปที่ 6-10 ตัวอย่างของความสัมพันธ์แบบ N-ary Relationship

6.5.7 Aggregation and Composition association

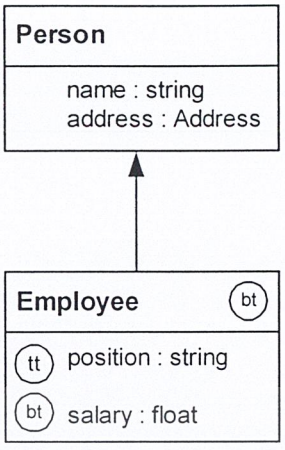
Aggregation และ Composition association ใน TCD นั้น สามารถนำคุณสมบัติทางเวลามาใช้กับ aggregation และ composition association ได้

รูปที่ 6-11 ตัวอย่างของ Aggregation

จากรูปที่ 6-11 แสดง aggregation ระหว่าง Project valid time class และ Task Bitemporal time class ซึ่งเป็น valid time association โดยสมมติว่า Project หนึ่งอาจประกอบด้วย Task หลายๆ Task และ Task หนึ่งอาจจะเป็นส่วนหนึ่งของหลาย Project

6.5.8 Generalization

Generalization relationship ใน TCD นั้น subclass จะมีการแปะเวลาเช่นเดียวกันกับ superclass ของมันเอง อย่างไรก็ดี subclass อาจจะเป็น temporal class ประเภทอื่นก็ได้ โดยเฉพาะ static class อาจจะมี subclass เป็น transaction, valid or bitemporal time class ขณะที่ transaction หรือ valid time class ก็สามารถมี bitemporal time class เป็น subclass ได้ โดยที่ subclass สามารถแปะเวลาลงไปที่ property ที่ inherit มาได้ทำให้ subclass ที่มี static property สามารถเปลี่ยนไปเป็น transaction, valid หรือ bitemporal time property ได้ขณะที่ transaction หรือ valid time property ก็สามารถเปลี่ยนไปเป็น bitemporal time property ได้เช่นเดียวกัน และสามารถเพิ่ม property ใหม่ที่มีการแปะเวลาเข้าไปด้วยก็ได้ ดังรูป

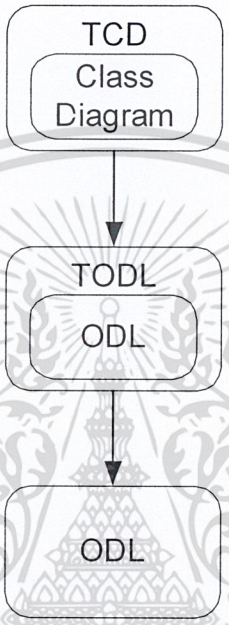


รูปที่ 6-12 ตัวอย่างของ Generalization

จากรูป 6-12 เป็นตัวอย่างของการถ่ายทอดคุณสมบัติ โดยคลาส Employee เป็นคลาสที่จัดเก็บช่วงชีวิตที่ออบเจ็กต์ของคลาสนั้นเป็นจริง และอยู่ในฐานข้อมูลหรือถูกเปลี่ยนแปลงแก้ไข ถูกกำหนดให้เป็นสับคลาส (Subclass) ของคลาส Person ที่เป็นคลาสที่ไม่มีเวลาเข้ามาเกี่ยวข้อง

บทที่ 7
การแปลงรูป

การแปลงแผนภาพคลาสเชิงเวลา (TCD) ให้สามารถทำงานร่วมกับฐานข้อมูลเชิงวัตถุได้นั้นมีขั้นตอนดังรูปที่ 7-1



รูปที่ 7-1 ขั้นตอนการแปลงแผนภาพคลาสเชิงเวลา TCD ไปเป็นภาษานิยามเชิงวัตถุ ODL

จากรูปที่ 7-1 TCD เป็นโมเดลสำหรับการออกแบบฐานข้อมูลเชิงเวลา เป็นโมเดลที่นำแผนภาพคลาสใน UML เดิมมาเพิ่มขยายความสามารถให้รองรับกับแนวคิดเชิงวัตถุและข้อมูลเชิงเวลาได้ สำหรับ TODL นั้นเป็นภาษานิยามเชิงเวลาบนฐานข้อมูลเชิงวัตถุ กล่าวคือได้นำเอา ODL ตามมาตรฐานของ ODMG มาเพิ่มขีดความสามารถให้รองรับข้อมูลที่แปรผันตามเวลาได้ หรือกล่าวอีกนัยหนึ่งก็คือ TODL ก็คือ ODL ที่เพิ่มส่วนของเวลาเข้าไปนั่นเอง โดยโมเดล TCD มีโมเดลแผนภาพคลาสใน UML เป็นโครงสร้างพื้นฐาน

สำหรับขั้นตอนการทำงานนั้นเราจะแปลงโมเดล TCD ให้เป็นภาษานิยามเชิงวัตถุอิงเวลา TODL ก่อนจากนั้นเราก็จะแปลงภาษานิยามเชิงวัตถุอิงเวลา TODL ให้เป็นภาษานิยามเชิงวัตถุ ODL อีกที ที่เราไม่ทำการแปลงโมเดล TCD ให้เป็นภาษานิยามเชิงเวลา ODL ในครั้งเดียวไปเลยก็เพราะว่า ในบางครั้งเราอาจไม่ต้องการแปลงโมเดลของเราให้สนับสนุนกับข้อมูลเชิงเวลา

ในบทนี้กล่าวถึง การแปลงรูปเป็นภาษานิยามเชิงวัตถุ (ODL) ซึ่งจะแบ่งส่วนอธิบายดังนี้ หัวข้อ 7.1 จะนำเสนอการแปลงแบบพื้นฐานจากแผนภาพคลาสให้เป็นภาษานิยามเชิงวัตถุ ODL จากนั้นหัวข้อ 7.2 จะนำเสนอการแปลงโมเดล TCD ให้เป็นภาษานิยามเชิงวัตถุอิงเวลา TODL สำหรับหัวข้อ 7.3 จะนำเสนอการแปลงภาษานิยามเชิงวัตถุอิงเวลา TODL ให้เป็นภาษานิยามเชิงวัตถุ ODL

7.1 การแปลงแผนภาพคลาส ให้เป็นภาษานิยามเชิงวัตถุ ODL

ก่อนที่จะกล่าวถึงการใช้เวลาในโมเดลนี้ จะขอกล่าวถึงการแปลงแผนภาพคลาสใน UML ให้เป็นภาษานิยามเชิงวัตถุ ODL ก่อน ซึ่งหลังจากได้มีการเปลี่ยนแปลงขั้นตอนนี้แล้ว เมื่อโมเดลมีเวลาใน TCD มาเกี่ยวข้อง สิ่งที่แปลงไว้ก่อนนี้จะช่วยให้การแปลงจาก TCD ให้เป็น ODL จะง่ายขึ้น สำหรับขั้นตอนในการแปลงนั้นเริ่มต้นต้องแปลงจากรายละเอียดของคลาสแต่ละคลาสก่อน จากนั้นจึงไปแปลงความสัมพันธ์ระหว่างคลาส สำหรับหลักการในการแปลงแผนภาพคลาส ให้เป็นภาษา ODL มีหลักการดังนี้

7.1.1 การแปลงคลาส

ชื่อของคลาสในแผนภาพคลาสจะถูกแปลงไปเป็น ชื่อ interface ในส่วน Interface Header ของภาษา ODL ได้ดังรูป 7-2

รูปที่ 7-2 ตัวอย่างการแปลง Class เป็นภาษา ODL

7.1.2 การแปลงแอตทริบิวต์ของคลาส

แอตทริบิวต์ของคลาสแผนภาพคลาสจะถูกแปลงเป็น attribute ในส่วน Inteface Body ของภาษา ODL และถ้าแอตทริบิวต์ใดมีหลายค่าจะถูกแปลงให้เป็น Set ได้ดังรูป 7-3

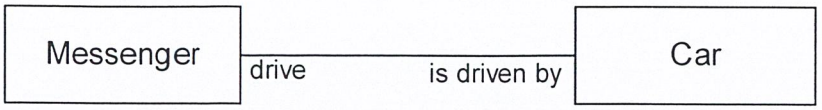
รูปที่ 7-3 ตัวอย่างการแปลง Attribute เป็นภาษา ODL

7.1.3 การแปลงความสัมพันธ์แบบ Association

ความสัมพันธ์ระหว่างคลาสนั้นจะถูกแปลงเป็นพารามิเตอร์ relationships ในส่วน Interface Body ใน ODL ของแต่ละคลาสที่มีความสัมพันธ์กัน โดยสามารถแบ่งตามความสัมพันธ์ระหว่างคลาสที่เกิดขึ้นได้ดังนี้

7.1.3.1 ความสัมพันธ์แบบหนึ่งต่อหนึ่ง (One to one relationship)

ความสัมพันธ์แบบนี้เป็นความสัมพันธ์ที่แต่ละออบเจ็กต์ของคลาสหนึ่งมีความสัมพันธ์กับออบเจ็กต์ของอีกคลาสหนึ่งได้เพียงออบเจ็กต์เดียวเท่านั้น สามารถแปลงความสัมพันธ์แบบหนึ่งต่อหนึ่งเป็น ODL ได้ดังรูป 7-4

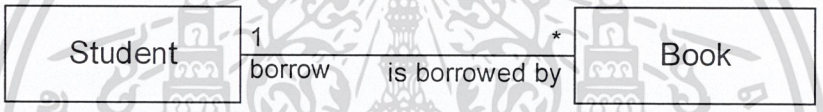



```
interface Messenger                    interface Car
{relationship Car drive                {relationshipMessenger is_driven_by
        inverse Car::is_driven_by;     inverse Messenger::drive;
};                                     };
```

รูปที่ 7-4 ตัวอย่างการแปลงความสัมพันธ์แบบหนึ่งต่อหนึ่ง เป็นภาษา ODL

7.1.3.2 ความสัมพันธ์แบบหนึ่งต่อหลาย (One to many relationship)

ความสัมพันธ์แบบนี้เป็นความสัมพันธ์ที่แต่ละออบเจ็กต์ของคลาสหนึ่งมีความสัมพันธ์กับออบเจ็กต์ของอีกคลาสหนึ่งได้หลายออบเจ็กต์ สามารถแปลงความสัมพันธ์แบบหนึ่งต่อหลาย เป็น ODL ได้ดังรูป 7-5

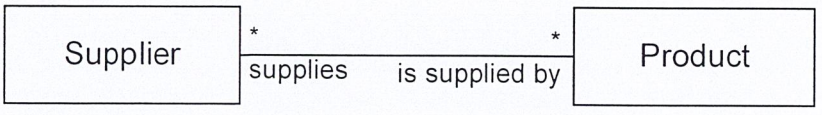



```
interface Student                      interface Book
{relationship set<Book> borrow         {relationship Student is_borrowed_by
        inverse Book::is_driven_by;            inverse Student::drive;
};                                     };
```

รูปที่ 7-5 ตัวอย่างการแปลงความสัมพันธ์แบบหนึ่งต่อหลาย เป็นภาษา ODL

7.1.3.3 ความสัมพันธ์แบบหลายต่อหลาย (Many to many relationship)

ความสัมพันธ์แบบนี้เป็นความสัมพันธ์ที่แต่ละออบเจ็กต์ของคลาสหนึ่งมีความสัมพันธ์กับออบเจ็กต์ของอีกคลาสหนึ่งได้หลายออบเจ็กต์ ได้ทั้งสองฝ่าย สามารถแปลงความสัมพันธ์แบบหลายต่อหลายได้เป็น ODL ได้รูป 7-6




```
interface Supplier                     interface Part
{relationship set<Product> supplies    {relationship  set<Supplier> is_supplied_by
        inverse Product::is_supplied_by;       inverse Supplier::supplies
};                                     };
```

รูปที่ 7-6 ตัวอย่างการแปลงความสัมพันธ์แบบหลายต่อหลาย เป็นภาษา ODL

7.1.4 การแปลงความสัมพันธ์แบบ Aggregation

ความสัมพันธ์แบบนี้เป็นความสัมพันธ์ที่มีลักษณะ Embedded Class หรือแบบเป็นส่วนหนึ่ง โดยการแปลงนั้นจะแปลงคลาส Component class หรือ Owned หรือ Part ไปเป็น Stucture ของคลาสที่มีหัวลูกศรรูปสี่เหลี่ยมชี้ ที่เรียกว่า Owner class หรือ Whole ใช้เป็นแอตทริบิวต์ชนิดนั้น ดังรูป 7-7


```
interface customer
{
        typedef struct Address{unsigned short number; string street; string city_name;};
        attribute string cust_name;
        attribute Address address;
};
```

รูปที่ 7-7 ตัวอย่างการแปลงความสัมพันธ์แบบ Aggregation เป็น ภาษา ODL

7.1.5 การแปลงความสัมพันธ์แบบ Generalization

ความสัมพันธ์แบบนี้เป็นความสัมพันธ์ที่คลาสหนึ่งถ่ายทอดคุณสมบัติมาจากอีกคลาสหนึ่งหรือจากหลายคลาส โดยการแปลงทำโดยคลาสที่เป็นคลาสลูกหรือ Subclass จะระบุ ชื่อคลาสที่เป็นคลาสแม่หรือ Superclass ทุกคลาสไว้ต่อท้ายชื่อคลาส ดังรูป 7-8


```
interface Employee:Person
(..){
};
```

รูปที่ 7-8 ตัวอย่างการแปลงความสัมพันธ์แบบ Generalzation เป็นภาษา ODL

7.2 การแปลงแผนภาพคลาสเชิงเวลา TCD ให้เป็นภาษานิยามเชิงวัตถุอิงเวลา TODL

TODL (Temporal Object Definition Language) เป็นภาษานิยามเชิงวัตถุอิงเวลานี้ได้สร้างขึ้นเพื่อเพิ่มคุณสมบัติเชิงเวลาเข้าไปในภาษานิยามเชิงวัตถุ ODL ซึ่งภาษา ODL นี้เป็นภาษามาตรฐานบนฐานข้อมูลเชิงวัตถุ ที่สามารถทำงานบนฐานข้อมูลเชิงวัตถุได้อยู่แล้ว

ภาษา TODL นี้ได้มีการเพิ่มเติมส่วนของเวลาเข้าไปในภาษา ODL ทำให้ไม่สามารถทำงานได้โดยตรงบนฐานข้อมูลเชิงวัตถุ จะต้องมีการแปลงให้เป็นภาษา ODL เสียก่อน จึงจะสามารถนำ TODL ไปสร้างออบเจ็กต์บนฐานข้อมูลเชิงวัตถุได้

สำหรับหลักการแปลง TCD ให้เป็นภาษานิยามเชิงวัตถุอิงเวลา TODL นั้นต้องทำกระบวนการแปลงแผนภาพคลาส ให้เป็นภาษานิยามเชิงวัตถุ ODL ก่อน จนได้ ODL ของคลาสทั้งหมด จากนั้นนำคลาสทั้งหมดนี้มาเพิ่มส่วนของเวลาเข้าไป โดยกลับไปพิจารณาโมเดลที่สร้างขึ้นอีกครั้ง ซึ่งมีหลักเกณฑ์ดังนี้

7.2.1 คลาส

ใน TCD ได้นำเวลาเพิ่มไว้ที่คลาส ดังนั้นจำเป็นต้องเพิ่มสัญลักษณ์ของเวลาเข้าไปที่คำสั่ง Interface ซึ่งการเพิ่มเวลาไว้ที่คลาสสามารถแบ่งรายละเอียดได้ดังนี้

7.2.1.1 คลาสที่ไม่มีเวลาเข้ามาเกี่ยวข้อง (Static Class)

สำหรับการประกาศคลาสแบบนี้จะเหมือนการแปลงแผนภาพคลาสไปเป็นภาษา ODL ดังที่กล่าวมาแล้ว

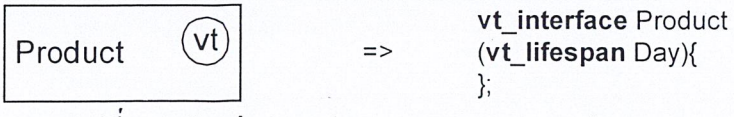
7.2.1.2 คลาสที่เก็บเวลาที่อยู่ในฐานข้อมูล (Transaction Time Class)

คลาสนี้จะมีสัญลักษณ์ ⓣⓣ อยู่ที่มุมขวาบนของชื่อคลาส ซึ่งสามารถแปลงเป็น TODL โดยมีคำว่า tt_interface แทน interface ดังรูป 7-9

รูปที่ 7-9 ตัวอย่างการแปลง Transaction Time Class เป็นภาษา TODL

7.2.1.3 คลาสที่เก็บเวลาที่ข้อมูลเป็นจริง (Valid Time Class)

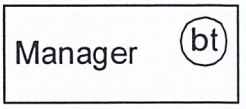
คลาสนี้จะมีสัญลักษณ์ ⓥⓣ อยู่ที่มุมขวาบนของชื่อคลาส ซึ่งสามารถแปลงเป็น TODL โดยมีคำว่า vt_interface แทน interface และกำหนด vt_lifespan ตาม granularity ด้วยดังรูป 7-10



รูปที่ 7-10 ตัวอย่างการแปลง Valid Time Class เป็นภาษา TODL

7.2.1.4 คลาสที่เก็บเวลาที่ข้อมูลเป็นจริงและอยู่ในฐานข้อมูล (Bitemporal Time Class)

คลาสนี้จะมีสัญลักษณ์ (bt) อยู่ที่มุมขวาบนของชื่อคลาส ซึ่งสามารถแปลงเป็นTODL โดยมีคำว่า bt_interface แทน interfaceและกำหนด vt_lifespan ตาม granularity ด้วย ดังรูป 7-11



=>

```
bt_interface Manager
(bt_lifespan Day){
};
```

รูปที่ 7-11 ตัวอย่างการแปลง Bitemporal Time Class เป็นภาษา TODL

7.2.2 แอตทริบิวต์ของคลาส

แอตทริบิวต์ของคลาส ใน TCD นั้น มีการนำเวลาเพิ่มไว้ในรายละเอียดของคลาส ดังนั้นเมื่อนำเวลามาเพิ่มที่แอตทริบิวต์ก็จำเป็นต้องให้เพิ่มสัญลักษณ์ของเวลาเข้าไปที่คำสั่ง attribute ซึ่งการเพิ่มเวลาไว้ที่แอตทริบิวต์สามารถแบ่งรายละเอียดได้ดังนี้

7.2.2.1 แอตทริบิวต์ที่ไม่มีเวลาเข้ามาเกี่ยวข้อง (Static Attributes)

สำหรับการประกาศในแอตทริบิวต์แบบนี้จะเหมือนกับการแปลงแผนภาพคลาสไปเป็นภาษา ODL ดังที่กล่าวมาแล้ว

7.2.2.2 แอตทริบิวต์ที่จัดเก็บข้อมูลและเวลาที่ข้อมูลอยู่ในฐานข้อมูล (Transaction Time Attribute)

แอตทริบิวต์นี้จะมีสัญลักษณ์ (tt) อยู่หน้าแอตทริบิวต์นั้น ซึ่งสามารถแปลงเป็น TODL โดยมีคำว่า tt_attribute แทน attribute ดังนี้

(tt) position : **string**

แปลงไปเป็นภาษา ODL

tt_attribute string position;

7.2.2.3 แอตทริบิวต์ที่จัดเก็บข้อมูลและเวลาที่ข้อมูลเป็นจริง (Valid Time Attribute)

แอตทริบิวต์นี้จะมีสัญลักษณ์ (vt) อยู่หน้าแอตทริบิวต์นั้น ซึ่งสามารถแปลงเป็น TODL โดยมีคำว่า vt_attribute แทน attribute แต่เวลาที่วางเข้าไปนี้ผู้ใช้จำเป็นต้องป้อนเข้าไปเอง จึงต้องมีการระบุว่าวางเวลาประเภทใด และใช้หน่วยของเวลาใดเป็นหน่วยเล็กที่สุด ดังนี้

(vt) price : **float** {**vt_type** = 'P', **granularity** = 'month'}

แปลงไปเป็นภาษา ODL

vt_attribute<P,MONTH> **float** price;

7.2.2.4 แอตทริบิวต์ที่จัดเก็บข้อมูลและเวลาทั้งที่ข้อมูลอยู่ในฐานข้อมูลและข้อมูลเป็นจริง (Bitemporal Time Attribute)

แอตทริบิวต์นี้จะมีสัญลักษณ์ (bt) อยู่หน้าแอตทริบิวต์นั้น ซึ่งสามารถแปลงเป็น TODL โดยมีคำว่า bt_attribute แทน attribute แต่เวลาที่วางเข้าไปนี้ผู้ใช้นำเป็นต้องป้อนเข้าไปเอง จึงต้องมีการระบุว่าวางเวลาประเภทใด และใช้หน่วยของเวลาใดเป็นหน่วยเล็กที่สุด ดังนี้

(bt) salary : **float** {**vt_type** = 'P', **granularity** = 'month' }

แปลงไปเป็นภาษา ODL

bt_attribute<P,MONTH> **float** salary;

7.2.3 ความสัมพันธ์ระหว่างคลาส

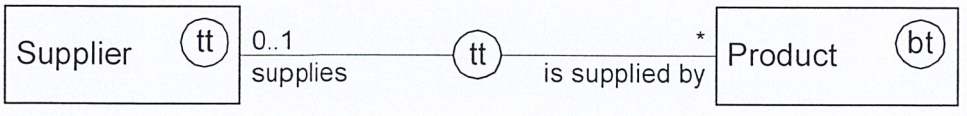
การนำเวลาเข้าไปเพิ่มในความสัมพันธ์ระหว่างคลาสนี้ ทำได้เฉพาะความสัมพันธ์ระหว่างคลาสที่เป็นแบบ binary association เท่านั้น ดังนั้นหากต้องการแปลงความสัมพันธ์ระหว่างคลาสที่เป็นแบบอื่นๆ ให้อยู่ในรูป TODL ต้องทำการแปลงความสัมพันธ์นั้นให้ออกมาเป็นความสัมพันธ์แบบ binary association เสียก่อนจึงจะสามารถแปลงเป็น TODL ได้ ซึ่งสามารถสรุปได้ดังนี้

7.2.3.1 ความสัมพันธ์ที่ไม่มีเวลามาเกี่ยวข้อง (Static Relationship)

ความสัมพันธ์นี้จะไม่มีการจัดเก็บเวลาของความสัมพันธ์ ดังนั้นจึงไม่ต้องคำนึงถึงส่วนที่ต้องเข้าไป ทำให้ความสัมพันธ์ที่ปรากฏเหมือนกับการแปลงแผนภาพคลาสไป ODL ทุกประการ ดังที่กล่าวมาแล้ว

7.2.3.2 ความสัมพันธ์ที่จัดเก็บเวลาที่ความสัมพันธ์นั้นอยู่ในฐานข้อมูล (Transaction Time Relationship)

ความสัมพันธ์นี้จะต้องมีสัญลักษณ์ (tt) ไว้ที่ความสัมพันธ์นั้นๆ ซึ่งสามารถแปลงเป็น TODL โดยมีคำว่า tt_relationship แทน relationship



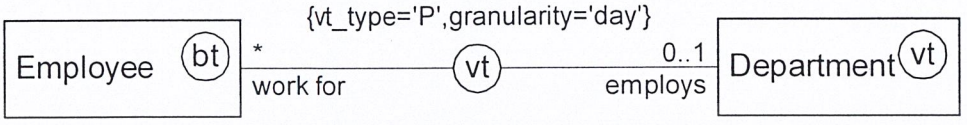
tt_interface Supplier (..)
{**tt_relationship set**<Product> supplies
inverse Product::is_supplied_by;
};

bt_interface Part (..)
{**tt_relationship** <Supplier> is_supplied_by
inverse Supplier::supplies
};

รูปที่ 7-12 ตัวอย่างการแปลง Transaction Time Relationship เป็นภาษา TODL

7.2.3.3 ความสัมพันธ์ที่จัดเก็บเวลาที่ความสัมพันธ์นั้นเป็นจริง (Valid Time Relationship)

ความสัมพันธ์นี้จะต้องมีสัญลักษณ์ (vt) ไว้ที่ความสัมพันธ์นั้นๆ ซึ่งสามารถแปลงเป็น TODL โดยมีคำว่า vt_relationships แทน relationship แต่เวลาที่วางเข้าไปนี้ผู้ใช้นำเป็นต้องป้อนเข้าไปเอง จึงต้องมีการระบุว่าวางเวลาประเภทใด และใช้หน่วยของเวลาใดเป็นหน่วยเล็กที่สุด ดังรูป 7-13

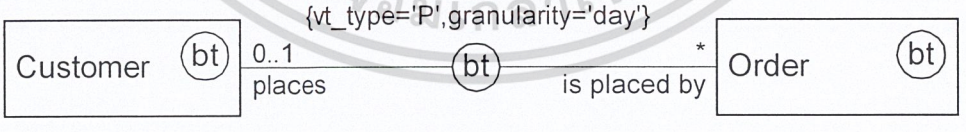



```
bt_interface Employee (..)
{       vt_relationship<P,DAY> Department work_for  inverse Department::employs };
vt_interface Department (..)
{       vt_relationship<P,DAY> Set<Employee> employs inverse Employee::work_for };
```

รูปที่ 7-13 ตัวอย่างการแปลง Valid Time Relationship เป็นภาษา TODL

7.2.3.4 ความสัมพันธ์ที่จัดเก็บทั้งเวลาที่ความสัมพันธ์เป็นจริงและเวลาที่ความสัมพันธ์อยู่ในฐานข้อมูล (Bitemporal Time Relationship)

ความสัมพันธ์นี้จะต้องมีสัญลักษณ์ (bt) ไว้ที่ความสัมพันธ์นั้น ซึ่งสามารถแปลงเป็น TODL โดยมีความว่า bt_relationship แต่เวลาที่วางเข้าไปนี้ผู้ใช้นำเป็นต้องป้อนเข้าไปเอง จึงต้องมีการระบุว่าวางเวลาประเภทใด และใช้หน่วยของเวลาใดเป็นหน่วยเล็กที่สุด ดังรูป 7-14




```
bt_interface Customer (..)
{       bt_relationship<P,DAY> Set<Order> places inverse Order::is_placed_by };
bt_interface Order (..)
{       bt_relationship<P,DAY> Customer is_places_by  inverse Customer::places };
```

รูปที่ 7-14 ตัวอย่างการแปลง Bitemporal Time Relationship เป็นภาษา TODL

7.3 การแปลงภาษานิยามเชิงวัตถุอิงเวลา TODL ให้เป็นภาษานิยามเชิงวัตถุ ODL

ภาษานิยามเชิงวัตถุ เป็นภาษาสำหรับสร้างออบเจ็กต์บนฐานข้อมูลเชิงวัตถุ เพื่อนำออบเจ็กต์ที่สร้างไปทำงานเพื่อจัดเก็บ และเรียกค้นข้อมูลตามต้องการได้ สำหรับรายละเอียดมีดังนี้

7.3.1 คลาส

คลาสที่เป็น TODL มีด้วยกัน 3 ประเภทด้วยกันคือ transaction time class, valid time class และ bitemporal time class ดังนั้นการแปลง TODL เป็น ODL มีดังนี้

7.3.1.1 การแปลง Transaction Time Class รูปแบบใน TODL มีดังนี้

```
tt_interface class-name [<inheritance_spec>]
[<type_property_list>] {…};
```

แปลงไปเป็นภาษา ODL แล้วจะได้

```
interface class-name [<inheritance_spec>]
[<type_property_list>] {
        attribute Temporal_Element<SECOND> tt_lifespan;
        …
};
```

7.3.1.2 การแปลง Valid Time Class รูปแบบใน TODL มีดังนี้

```
vt_interface class-name [<inheritance_spec>]
([<type_property_list>]
vt_lifespan granularity ) {…};
```

แปลงไปเป็นภาษา ODL แล้วจะได้

```
interface class-name [<inheritance_spec>]
[<type_property_list>] {
        attribute Temporal_Element<granularity> vt_lifespan;
        …
};
```

7.3.1.3 การแปลง Bitemporal Time Class รูปแบบใน TODL มีดังนี้

bt_interface class-name [<inheritance_spec>]
([<type_property_list>]
vt_lifespan granularity) {…};

แปลงไปเป็นภาษา ODL แล้วจะได้

interface class-name [<inheritance_spec>]
[<type_property_list>] {
attribute Temporal_Element<SECOND> tt_lifespan;
typedef struct{ **Period<SECOND>** tt_timestamp;
Temporal_Element<granularity> vt_lifespan} VT_Lifespan_Element;
attribute Set<VT_Lifespan element> vt_lifespan;
…
};

7.3.2 แอตทริบิวต์

แอตทริบิวต์ที่เกี่ยวข้องกับเวลาในภาษา TODL มีด้วยกัน 3 ประเภทด้วยกันคือ tt_attribute, vt_attribute, และ bt_attribute ดังนั้นการแปลง TODL เป็น ODL มีดังนี้

7.3.2.1 การแปลง tt_attribute

การประกาศ Transaction Time Attribute ในภาษา TODL มีรูปแบบตามนี้

tt_attribute Type tta;

โดยที่

Type	type ของ attribute
tta	ชื่อ ของ attribute

เมื่อแปลงไปเป็นภาษา ODL แล้วจะได้

attribute Set<struct element{**Type** value, **Period<SECOND>** tt_timestamp}> tta;

7.3.2.2 การแปลง vt_attribute

การประกาศ Valid Time Attribute ในภาษา TODL มีรูปแบบตามนี้

vt_attribute<tsc,gr> Type vta;

โดยที่

tsc ใช้บอกประเภทของ timestamp

gr ใช้บอก granularity

เมื่อแปลงไปเป็นภาษา ODL แล้วจะได้

- ถ้า tsc มีค่าเท่ากับ P (Period) แล้ว

attribute Set<struct element{**Type** value, **Period<gr>** vt_timestamp}> vta;

- ถ้า tsc มีค่าเท่ากับ T (Timepoint) แล้ว

attribute Set<struct element{**Type** value,**Timepoint<gr>** vt_timestamp}> vta;

7.3.2.3 การแปลง bt_attribute

การประกาศ Bitemporal Time Attribute ในภาษา TODL มีรูปแบบตามนี้

bt_attribute<tsc,gr> Type bta;

เมื่อแปลงไปเป็นภาษา ODL แล้วจะได้

- ถ้า tsc มีค่าเท่ากับ P (Period) แล้ว

attribute Set<struct element{**Type** value, **Period<SECOND>** tt_timestamp,
Period<gr> vt_timestamp}> bta;

- ถ้า tsc มีค่าเท่ากับ T (Timepoint) แล้ว

attribute Set<struct element{**Type** value,**Timepoint<SECOND>** tt_timestamp,
Timepoint<gr> vt_timestamp}> bta;

7.3.3 ความสัมพันธ์ (Relationships)

สำหรับความสัมพันธ์ที่มีเวลาเข้ามาเกี่ยวข้องนั้น เมื่อแปลง TODL ให้กลายเป็น ODL แล้วจะมีการสร้างคลาสใหม่เพื่อแสดงความสัมพันธ์ดังตัวอย่าง

พิจารณาความสัมพันธ์ที่จัดเก็บเวลาที่ความสัมพันธ์อยู่ในฐานข้อมูล แบบหนึ่งต่อหนึ่ง (transaction time one-one relationship) ระหว่าง Atomic Object Types ชื่อ A และ B ในภาษา TODL มีรูปแบบดังนี้

Type A:

tt_relationship B r_A

inverse B :: r_B;

Type B:

tt_relationship A r_B

inverse A :: r_A;

โดยที่

r_A ชื่อของ traversal path ของ relationship ที่ประกาศใน interface ของ type A

r_B ชื่อของ traversal path ของ relationship ที่ประกาศใน interface ของ type B

ขั้นตอนการแปลงเป็นภาษา ODL เป็นดังนี้

1. สร้าง type ใหม่เพื่อแสดงความสัมพันธ์เป็นภาษา ODL ตามนี้

interface A_r_A_B_r_B {

attribute Period<SECOND> tt_timestamp;

relationship A i_r_A

inverse A r_A;

relationship B i_r_B

inverse B r_B;

};

2. แปลง transaction time relationship r_A ที่อยู่ใน interface ของ type A ไปเป็น relationship ในภาษา ODL ได้ตามนี้

relationship Set< A_r_A_B_r_B > r_A

inverse A_r_A_B_r_B :: i_ r_A;

3. แปลง transaction time relationship r_B ที่อยู่ใน interface ของ type B ไปเป็น relationship ในภาษา ODL ได้ตามนี้

relationship Set< $A\_r_A\_B\_r_B$ > r_B

inverse $A\_r_A\_B\_r_B$ :: $i\_ r_B$;

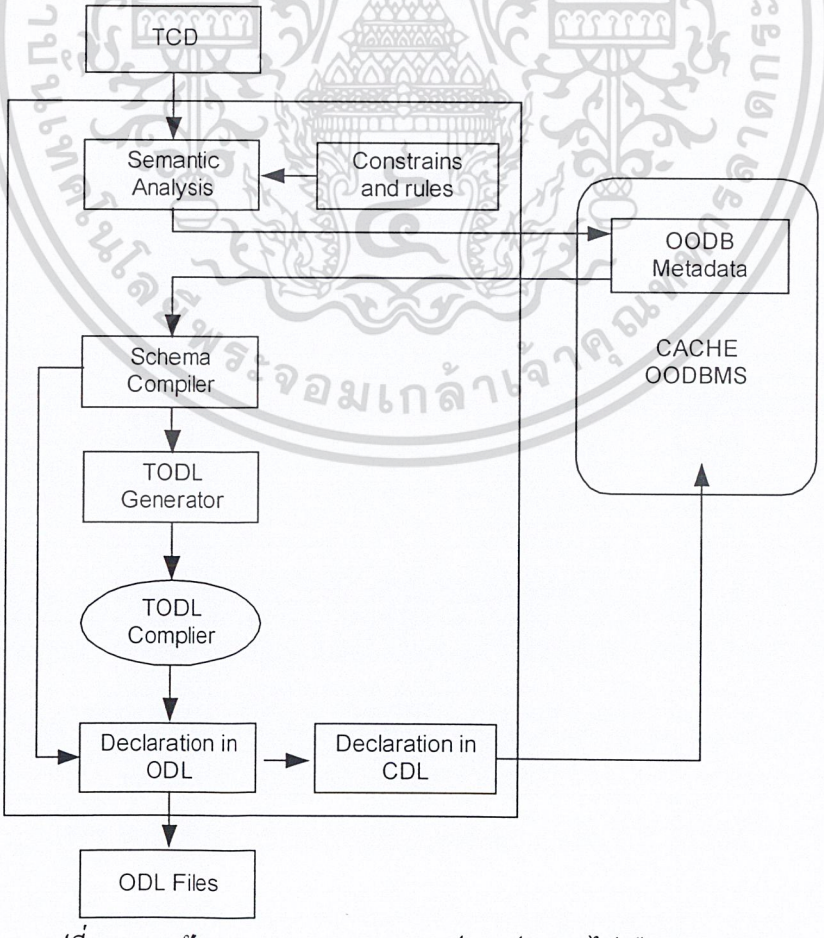
หากความสัมพันธ์ของเวลาแบบอื่นก็สามารถแปลงได้ เพียงเปลี่ยนประเภทของเวลาและหน่วยของเวลาในคลาสของความสัมพันธ์นั้น

บทที่ 8
ระบบอัตโนมัติในการแปลงรูป TCD ให้เป็น ODL

ในบทนี้จะกล่าวถึงการนำสิ่งต่างๆ ที่ได้ศึกษามาทำการพัฒนาโปรแกรมในการสร้างโมเดล TCD และการแปลงโมเดลที่สร้างขึ้นให้เป็นภาษานิยามเชิงวัตถุ (ODL) ซึ่งได้มีการแยกการอธิบายเป็นส่วนๆ คือ ในหัวข้อ 8.1 อธิบายถึงสถาปัตยกรรมของระบบ ซึ่งเป็นส่วนสำคัญที่จะแสดงถึงขั้นตอนการทำงานทั้งหมด ในหัวข้อ 8.2 คืออุปกรณ์ที่ใช้สร้างโปรแกรมนี้ ในหัวข้อ 8.3 ได้อธิบายถึงการเชื่อมต่อโปรแกรมกับระบบจัดการฐานข้อมูล สำหรับหัวข้อ 8.4 ได้อธิบายถึงการออกแบบฐานข้อมูล Meta Data เพื่อจัดเก็บโครงสร้างของ TCD ที่ผู้ใช้สร้างขึ้น เพื่อนำฐานข้อมูล Meta นี้ไปสร้างเป็นภาษา ODL อีกทอดหนึ่ง หัวข้อ 8.5 อธิบายถึงขั้นตอนการประมวลผลของโปรแกรม หัวข้อสุดท้ายคือหัวข้อ 8.6 เป็นตัวอย่างการใช้งานโปรแกรม

8.1 สถาปัตยกรรมของระบบ

สำหรับโปรแกรมประยุกต์ที่สร้างขึ้นเพื่อแปลงรูป TCD ให้กลายเป็นภาษา ODL นั้นมีรายละเอียดดังรูป 8-1



รูปที่ 8-1 สถาปัตยกรรมของระบบการแปลงรูป TCD ไปเป็นภาษา ODL

ซอฟต์แวร์ที่สร้างขึ้นนี้ มีจุดประสงค์ที่จะนำแบบจำลอง TCD ที่ผู้ใช้งานออกแบบมาแปลงให้เป็นภาษา ODL เพื่อสร้าง Schema ของฐานข้อมูลเชิงเวลา และนำ Schema ที่ได้นี้ไปใช้งานกับระบบฐานข้อมูลเชิงวัตถุ สำหรับระบบฐานข้อมูลเชิงวัตถุที่นำมาใช้งานคือ ระบบฐานข้อมูล Caché โดยระบบฐานข้อมูลดังกล่าวนี้จะจัดเก็บ Meta data ของ TCD จากนั้นจะนำ Meta Data นี้ไปผ่านกระบวนการเพื่อแปลงเป็น ODL จากนั้นจะนำภาษา ODL นี้กลับไปทำงานบนระบบฐานข้อมูล Caché อีกครั้ง เพื่อให้ระบบฐานข้อมูลสร้างฐานข้อมูล ตามที่ผู้ใช้งานออกแบบไว้

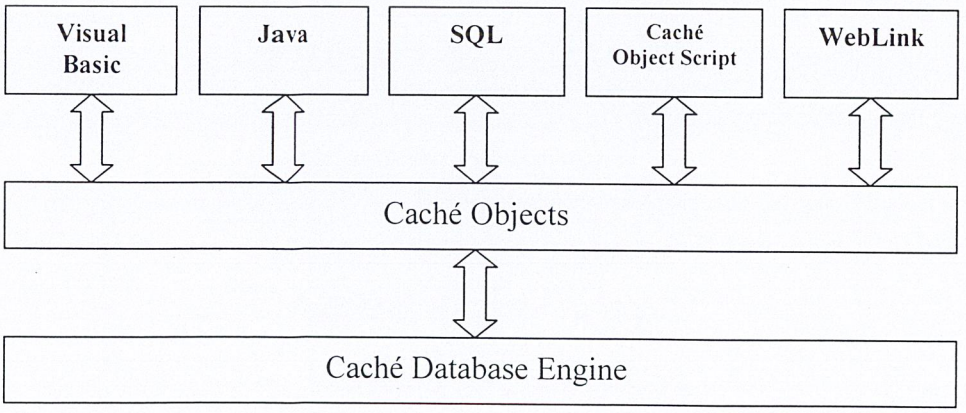
8.2 อุปกรณ์ที่ใช้

สำหรับอุปกรณ์ที่ใช้เพื่อสร้างโปรแกรมนี้นั้นประกอบด้วยฮาร์ดแวร์และซอฟต์แวร์ดังรี้

1. เครื่องคอมพิวเตอร์
   - หน่วยประมวลผลกลาง Pentium III ความเร็ว 600 เมกกะเฮริซ์
   - หน่วยความจำ 256 เมกกะไบต์
   - ฮาร์ดดิสก์ 10 กิกะไบต์
2. ระบบปฏิบัติการ Microsoft Windows 2000 Advanced Server Service Pack 2
3. ระบบฐานข้อมูล InterSystems Caché Post-Relation Database for Window NT(Intel) 4.0
4. โปรแกรม JBuilder 5 Enterprise

สำหรับฐานข้อมูลที่ใช้ในการจัดเก็บ Meta data และนำภาษา ODL มาทดสอบนั้นคือระบบฐานข้อมูล Caché ซึ่งเป็นระบบฐานข้อมูลที่สามารถรองรับโครงสร้างข้อมูลได้ทั้งฐานข้อมูลเชิงสัมพันธ์และฐานข้อมูลเชิงวัตถุ ซึ่งมีรายละเอียดดังนี้

ระบบจัดการฐานข้อมูลเชิงวัตถุที่ใช้ (ODBMS) ได้แก่ Caché (อ่านว่า คา-เช่) พัฒนาโดย บริษัท InterSystems Corporation ซึ่งเป็นระบบจัดการฐานข้อมูลที่สนับสนุนเทคโนโลยีเชิงวัตถุ (Object-Oriented Technology) สามารถจำลองโครงสร้างของข้อมูลที่มีความซับซ้อนให้คงสภาพตามที่เป็นจริงโดยไม่ต้องเปลี่ยนรูปร่างของโครงสร้างข้อมูล ทำให้ลดความสับสนในการกำหนดความสัมพันธ์ของข้อมูลไปได้มาก สำหรับการใช้งาน Caché สามารถทำงานได้บนหลากหลายแพลตฟอร์ม (Hardware Platform) และหลากหลายระบบปฏิบัติการ (Operating System) อาทิเช่น Window 95/98/NT/2000, UNIX และ Linux เป็นต้น



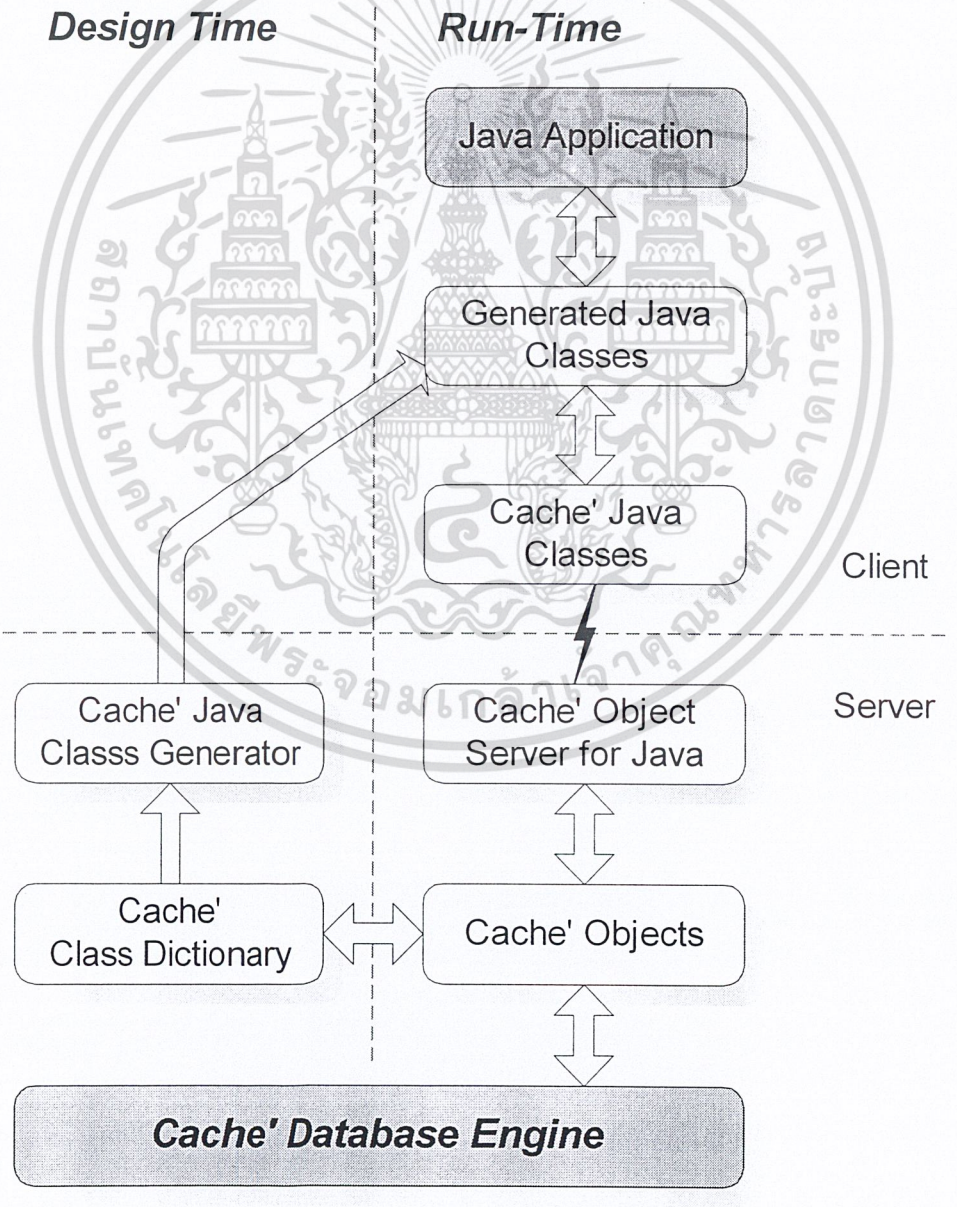
รูปที่ 8-2 การติดต่อกับระบบจัดการฐานข้อมูล Caché

8.3 การเชื่อมต่อ Caché และ Java Application (Caché Java Binding)

ระบบอัตโนมัติในการสร้าง TCD และแปลงรูปไปเป็น ODL นี้สนับสนุนการทำงานแบบไคลเอนต์/เซิร์ฟเวอร์ โดยฝั่งไคลเอนต์จะพัฒนาโดยใช้ภาษาจาวา และทางฝั่งดาต้าเบสเซิร์ฟเวอร์ (Database Server) ใช้ InterSystems Caché ซึ่งการจัดการกับฐานข้อมูลพัฒนาโดยใช้ Caché Java Binding ซึ่งเป็นเครื่องมือที่ทาง Caché ได้จัดเตรียมไว้ให้ใช้ในการพัฒนาโปรแกรมประยุกต์ภาษาจาวา ที่ต้องการติดต่อกับระบบฐานข้อมูล Caché

Caché Java Binding จะมี Caché Object Server สำหรับจาวาที่ทำงานร่วมกันกับ Caché เพื่อ

- สร้าง Java Class เพื่อใช้ในการเข้าถึงออบเจ็กต์ในฐานข้อมูล Caché
- เพิ่มประสิทธิภาพในการติดต่อระหว่างฐานข้อมูล Caché และ Java-based Client



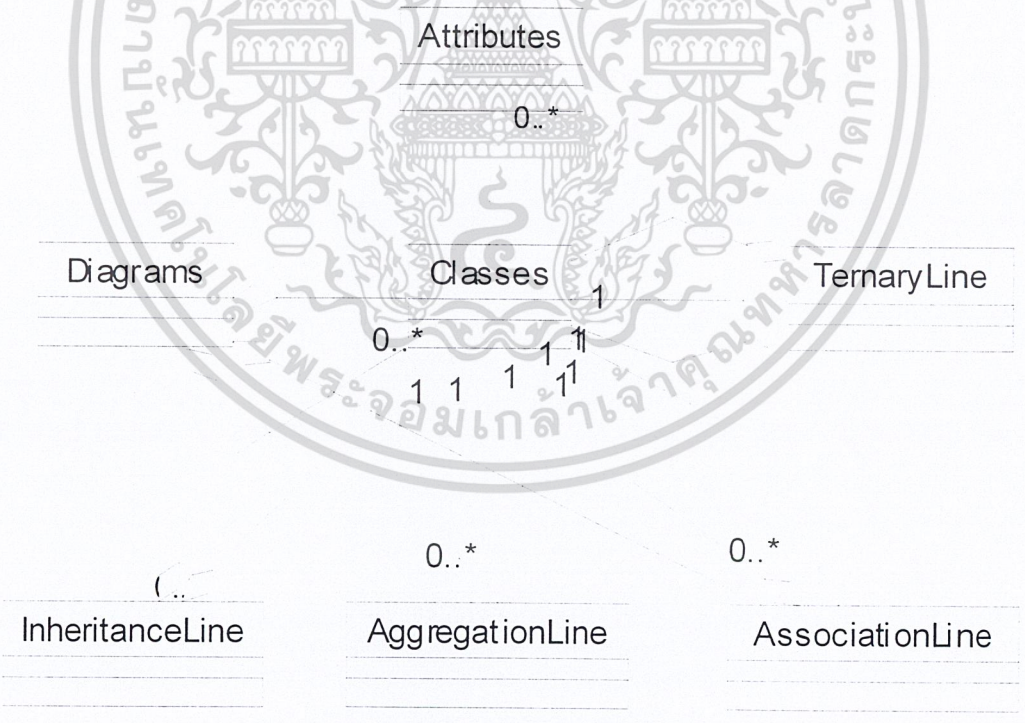
รูปที่ 8-3 การติดต่อระบบจัดการฐานข้อมูล Caché กับโปรแกรมภาษา Java

จากรูป 8-3 Caché Java Binding ประกอบไปด้วยส่วนประกอบต่างๆ ดังต่อไปนี้

- Caché Java Class Generator เป็นคอมไพเลอร์ที่ทำหน้าที่คอมไพล์ Caché Class ให้เป็น Java Class จากคลาสที่ประกาศไว้ใน Caché Class Dictionary
- Caché Java Package เป็น Package ของ Java Class ซึ่งทำงานกับ Java Class ที่ถูกสร้างขึ้นจาก Caché Java Class Generator และ ช่วยให้ Java Class เหล่านั้นสามารถใช้งานออบเจ็กต์ที่เก็บอยู่ในฐานข้อมูล Caché ได้โดยตรงผ่านทาง Java
- Caché Object Server สำหรับจาวาเป็น Gateway ประสิทธิภาพสูงซึ่งคอยจัดการการติดต่อระหว่าง Java Client และ Caché Database Server โดยใช้มาตรฐาน Protocol TCP/IP ในการเชื่อมต่อเครือข่าย

8.4 ฐานข้อมูล Meta

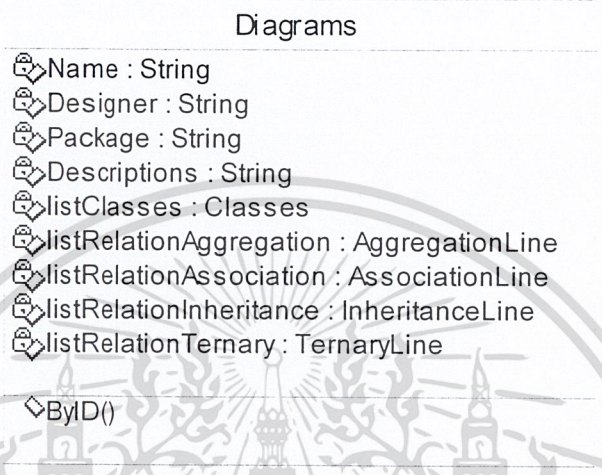
ฐานข้อมูล Meta เป็นฐานข้อมูลที่จัดเก็บ TCD ที่ผู้ใช้สร้าง โดยจัดเก็บรายละเอียดต่างๆ ของคลาส และความสัมพันธ์ระหว่างคลาส รวมถึงการจัดเก็บการถ่ายทอดคุณสมบัติของคลาสต่างๆ อีกด้วย ซึ่งฐานข้อมูล Meta นี้ ได้รับการออกแบบโดยใช้คลาสไดอะแกรม ซึ่งปรากฏในรูปที่ 2-5



รูปที่ 8-4 คลาสไดอะแกรมของฐานข้อมูล Meta

สำหรับรายละเอียดของแต่ละคลาสจะแสดงในหัวข้อต่อไปนี้

1. คลาส : Diagrams



รูปที่ 8-5 คลาส : Diagrams

คลาสชื่อ Diagrams ใช้เก็บข้อมูลไดอะแกรมที่วาดขึ้นทั้งหมดไว้ในฐานข้อมูล Meata โดยมีแอตทริบิวต์และเมธอดดังนี้

1.1 แอตทริบิวต์ Name : String ใช้เก็บชื่อของคลาสไดอะแกรม

1.2 แอตทริบิวต์ Designer : String ใช้เก็บชื่อของผู้ออกแบบคลาสไดอะแกรม

1.3 แอตทริบิวต์ Package : String ใช้เก็บชื่อแพจเกจสำหรับการ Export คลาสไดอะแกรม

1.4 แอตทริบิวต์ Descriptions : String ใช้เก็บคำอธิบายสำหรับคลาสไดอะแกรม

1.5 แอตทริบิวต์ listClasses : Classes ใช้เก็บคลาสทั้งหมดในแต่ละไดอะแกรมที่วาดขึ้น โดยเก็บเป็น List of Object

1.6 แอตทริบิวต์ listRelationAssociation :AssociationLine ใช้เก็บ Relationship แบบ Association ทั้งหมดในแต่ละไดอะแกรม โดยเก็บเป็น List of Object

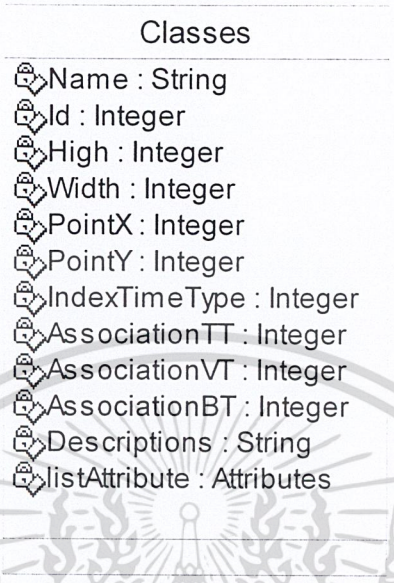
1.7 แอตทริบิวต์ listRelationAggregation : AggregationLine ใช้เก็บ Relationship แบบ Aggregation ทั้งหมดในแต่ละไดอะแกรม โดยเก็บเป็น List of Object

1.8 แอตทริบิวต์ listRelationInheritance : InheritanceLine ใช้เก็บ Relationship แบบ Inheritance ทั้งหมดในแต่ละไดอะแกรม โดยเก็บเป็น List of Object

1.9 แอตทริบิวต์ listRelationTernary : TernaryLine ใช้เก็บ Relationship แบบ Ternary Association ทั้งหมดในแต่ละไดอะแกรม โดยเก็บเป็น List Of Object

1.10 คิวรี ByID() เป็นคิวรีเมธอด ที่ใช้คิวรีออบเจกต์ทั้งหมดของคลาสนี้ออกมาเป็น Result Set

2. คลาส : Classes



รูปที่ 8-6 คลาส : Classes

คลาสชื่อ Classes เป็นคลาสที่ใช้จัดเก็บรายละเอียดต่างๆ ของคลาสทุกๆ คลาสที่วาดขึ้นมา ของทุกไดอะแกรม โดยมีแอตทริบิวต์และเมธอดดังนี้

2.1 แอตทริบิวต์ Name : String ใช้เก็บชื่อของคลาส

2.2 แอตทริบิวต์ Id : Integer ใช้เก็บหมายเลขของคลาสที่อยู่ในคลาสไดอะแกรม

2.3 แอตทริบิวต์ High : Integer ใช้เก็บความสูงของรูปคลาสที่วาด

2.4 แอตทริบิวต์ Width : Integer ใช้เก็บความกว้างของรูปคลาสที่วาด

2.5 แอตทริบิวต์ PointX : Integer ใช้เก็บพิกัด X ของตำแหน่งที่วางคลาส

2.6 แอตทริบิวต์ PointY : Integer ใช้เก็บพิกัด Y ของตำแหน่งที่วางคลาส

2.7 แอตทริบิวต์ IndexTimeType : Integer ใช้เก็บชนิดของข้อมูลเชิงเวลาของคลาส

2.8 แอตทริบิวต์ AssociationTT : Integer ใช้เก็บจำนวนความสัมพันธ์กับทรานแซคชั่นไทม์คลาส

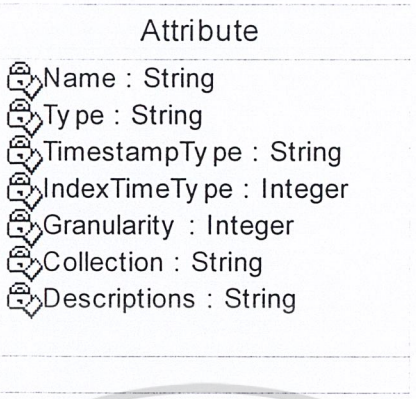
2.9 แอตทริบิวต์ AssociationVT : Integer ใช้เก็บจำนวนความสัมพันธ์กับเวลิดไทม์คลาส

2.10 แอตทริบิวต์ AssociationBT : Integer ใช้เก็บจำนวนความสัมพันธ์กับไบเทมโพรัลไทม์คลาส

2.11 แอตทริบิวต์ Descriptions : String ใช้เก็บคำอธิบายสำหรับคลาส

2.12 แอตทริบิวต์ listAttribute : Attributes ใช้เก็บ Attribute ที่มีในคลาสทั้งหมด โดยเก็บเป็น List Of Object

3. คลาส : Attributes



รูปที่ 8-7 คลาส : Attribute

คลาส : Attribute เป็นคลาสที่ใช้เก็บแอตทริบิวต์ของละคลาสทุกๆ คลาส โดยมีแอตทริบิวต์และเมธอดดังนี้

3.1 แอตทริบิวต์ Name : String ใช้เก็บชื่อของแอตทริบิวต์

3.2 แอตทริบิวต์ Type : String ใช้เก็บชนิดของแอตทริบิวต์

3.3 แอตทริบิวต์ TimestampType : String ใช้เก็บชนิดของช่วงเวลาที่ใช้ในข้อมูลเชิงเวลา

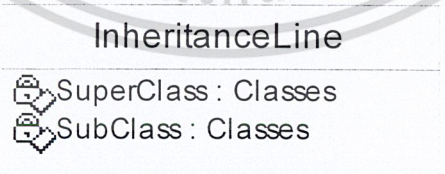
3.4 แอตทริบิวต์ IndexTimeType : Integer ใช้เก็บชนิดของข้อมูลเชิงเวลาของแอตทริบิวต์

3.5 แอตทริบิวต์ Granularity : Integer ใช้เก็บหน่วยของเวลาที่ใช้กับข้อมูลเชิงเวลา

3.6 แอตทริบิวต์ Collection: String ใช้เก็บลักษณะของแอตทริบิวต์ที่จัดเก็บ

3.7 แอตทริบิวต์ Descriptions: String ใช้เก็บคำอธิบายสำหรับแอตทริบิวต์

4. คลาส : InheritanceLine



รูปที่ 8-8 คลาส : InheritanceLine

คลาสชื่อ InheritanceLine ใช้จัดเก็บ Relationship แบบ Generization ระหว่างคลาสทุกคลาสของทุกคลาสไดอะแกรมที่วาดขึ้นมาทั้งหมด โดยจะมีแอตทริบิวต์ดังนี้

4.1 แอตทริบิวต์ SuperClass : Classes ใช้เก็บออบเจ็กต์ที่เป็น SuperClass

4.2 แอตทริบิวต์ SubClass : Classes ใช้เก็บออบเจ็กต์ที่เป็น SubClass

5. คลาส : AggregationLine

รูปที่ 8-9 คลาส : AggregationLine

คลาส : AggregationLine ใช้จัดเก็บ Relationship แบบ Aggregation ระหว่างคลาสต่างๆ ของทุกไดอะแกรมที่วาดขึ้น โดยมีแอตทริบิวต์ดังนี้

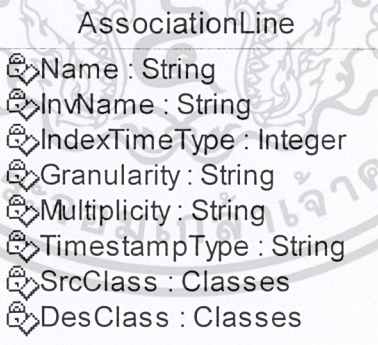
5.1 แอตทริบิวต์ Multiplicity : String ใช้เก็บ Multiplicity ของความสัมพันธ์ระหว่างคลาส

5.2 แอตทริบิวต์ RoleName : String ใช้เก็บชื่อบทบาท

5.3 แอตทริบิวต์ AggregationClass : Classes ใช้เก็บออบเจ็กต์ของคลาสที่ถูก Aggregation

5.4 แอตทริบิวต์ SrcClass : Classes ใช้เก็บออบเจ็กต์ของคลาสที่ทำ Aggregation

6. คลาส : AssociationLine



รูปที่ 8-10 คลาส: AssociationLine

คลาส : AssociationLine ใช้เก็บ Relationship แบบ Binary Association ระหว่างคลาสที่วาดขึ้นของทุกไดอะแกรม โดยมีแอตทริบิวต์ดังนี้

6.1 แอตทริบิวต์ Name : String ใช้เก็บชื่อความสัมพันธ์จากคลาส A ไป คลาส B

6.2 แอตทริบิวต์ InvName : String ใช้เก็บชื่อความสัมพันธ์จากคลาส B กลับมา คลาส A

6.3 แอตทริบิวต์ IndexTimeType : Integer ใช้เก็บชนิดของข้อมูลเชิงเวลาที่เพิ่มบนความสัมพันธ์นี้

6.4 แอตทริบิวต์ Granularity : String ใช้เก็บหน่วยของเวลาที่ใช้กับข้อมูลเชิงเวลา

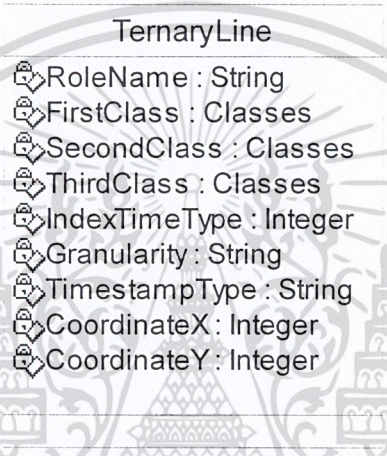
6.5 แอตทริบิวต์ Mulityplicity : String ใช้เก็บ Multiplicity ของความสัมพันธ์ระหว่างคลาส

6.6 แอตทริบิวต์ TimestampType : String ใช้เก็บชนิดของช่วงเวลาที่ใช้ในข้อมูลเชิงเวลา

6.7 แอตทริบิวต์ SrcClass : Classes ใช้เก็บออบเจ็กต์ของคลาสตัวแรก

6.8 แอตทริบิวต์ DesClass : Classes ใช้เก็บออบเจ็กต์ของคลาสตัวที่สอง

7. คลาส : TernaryLine



รูปที่ 8-11 คลาส: TernaryLine

คลาสชื่อ TernaryLine ใช้เก็บ Relationship แบบ Ternary Association ระหว่างคลาสที่วาดขึ้นของทุกไดอะแกรม โดยมีแอตทริบิวต์ดังนี้

7.1 แอตทริบิวต์ RoleName : String เป็นชื่อบทบาทของ Relationship

7.2 แอตทริบิวต์ FirstClass : Classes เป็นคลาสตัวแรกที่มี Relationship

7.3 แอตทริบิวต์ SecondClass : Classes เป็นคลาสตัวที่สองที่มี Relationship

7.4 แอตทริบิวต์ ThirdClass : Classes เป็นคลาสตัวที่สามที่มี Relationship

7.5 แอตทริบิวต์ IndexTimeType : Integer เป็นชนิดของข้อมูลเชิงเวลาที่เพิ่มใน Relationship

7.6 แอตทริบิวต์ Granularity : String เป็นหน่วยของเวลาที่ใช้กับข้อมูลเชิงเวลา

7.7 แอตทริบิวต์ TimestampType : String เป็นชนิดของช่วงเวลาที่ใช้ในข้อมูลเชิงเวลา

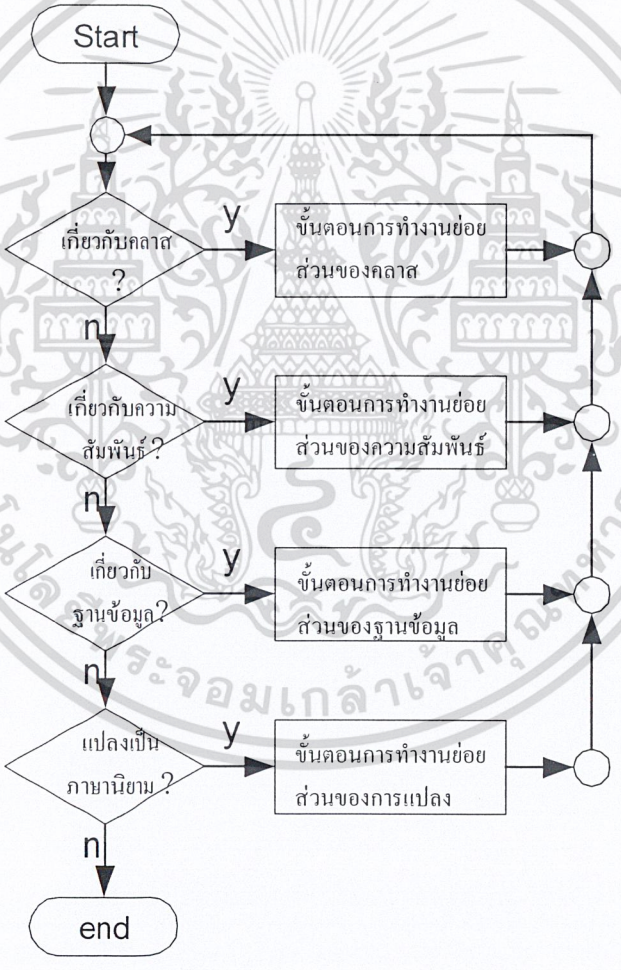
7.8 แอตทริบิวต์ CoordinateX : String เป็นตำแหน่งรูปบนแกน X

7.9 แอตทริบิวต์ CoordinateY : String เป็นตำแหน่งรูปบนแกน Y

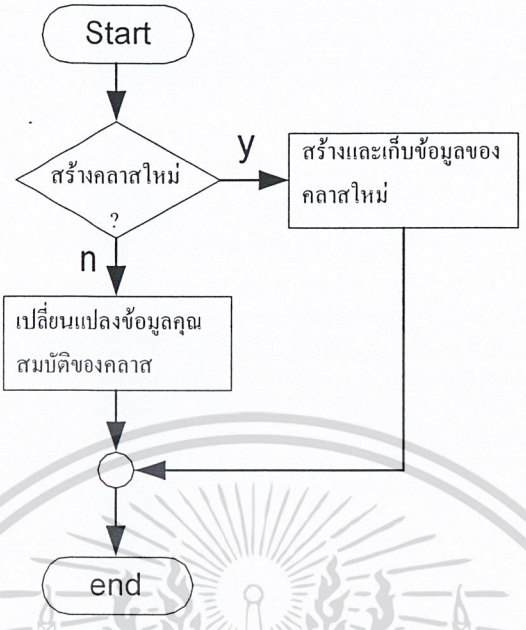
8.5 ขั้นตอนการประมวลผล

ขั้นตอนในการประมวลผลหลักแสดงดังรูปที่ 1 ซึ่งสามารถแยกเป็นส่วน ๆ ได้ตามลักษณะ Input ที่มาจากผู้ใช้ ซึ่งในแต่ละลักษณะ ก็จะมีขั้นตอนการทำงานย่อยออกไปอีก ดังนี้

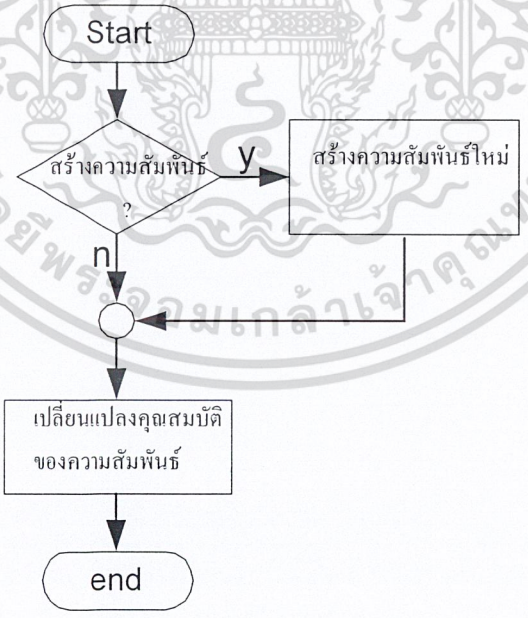
- ขั้นตอนการทำงานย่อยส่วนของคลาส แสดงดังรูปที่ 2
- ขั้นตอนการทำงานย่อยส่วนของความสัมพันธ์ แสดงดังรูปที่ 3
- ขั้นตอนการทำงานย่อยส่วนของฐานข้อมูล แสดงดังรูปที่ 4
- ขั้นตอนการทำงานย่อยส่วนของการแปลง แสดงดังรูปที่ 5



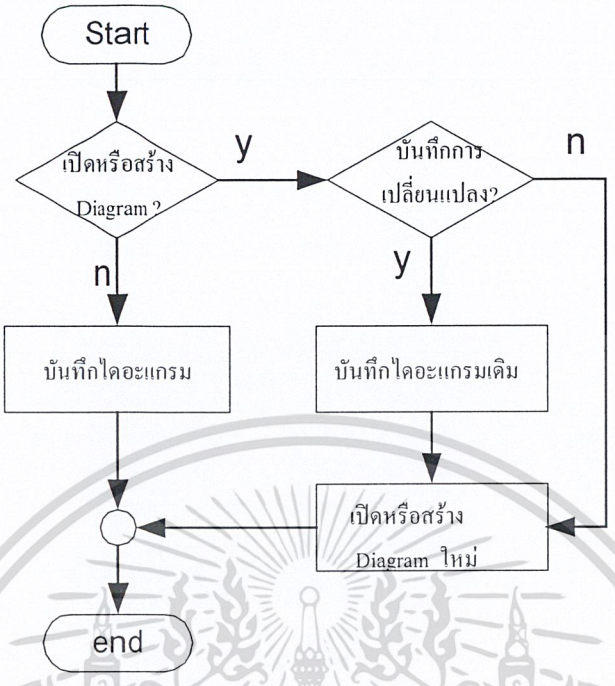
รูปที่ 8-12 ขั้นตอนการประมวลผลหลัก



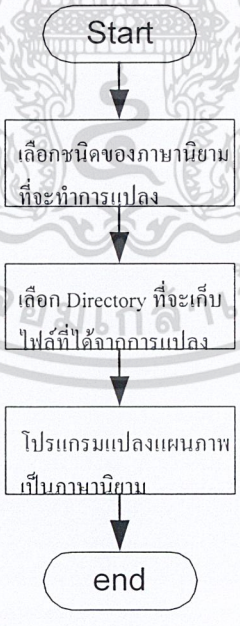
รูปที่ 8-13 ขั้นตอนการทำงานย่อยส่วนของคลาส



รูปที่ 8-14 ขั้นตอนการทำงานย่อยส่วนของความสัมพันธ์



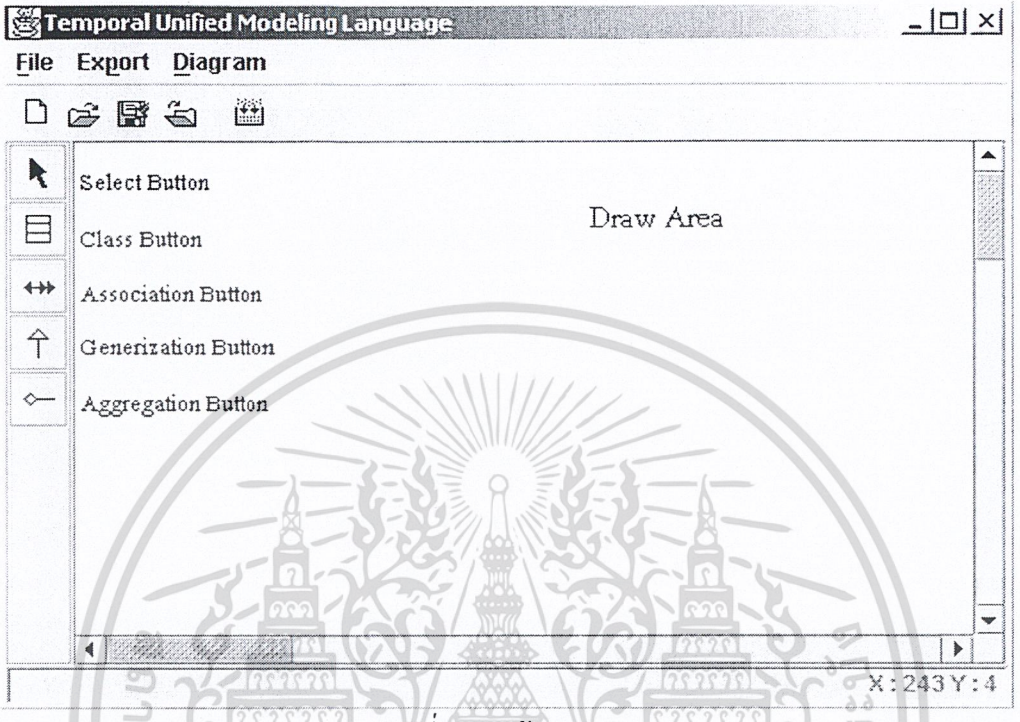
รูปที่ 8-15 ขั้นตอนการทำงานย่อยส่วนของฐานข้อมูล



รูปที่ 8-16 ขั้นตอนการทำงานย่อยส่วนของการแปลง TCD

8.6 ตัวอย่างการใช้งาน

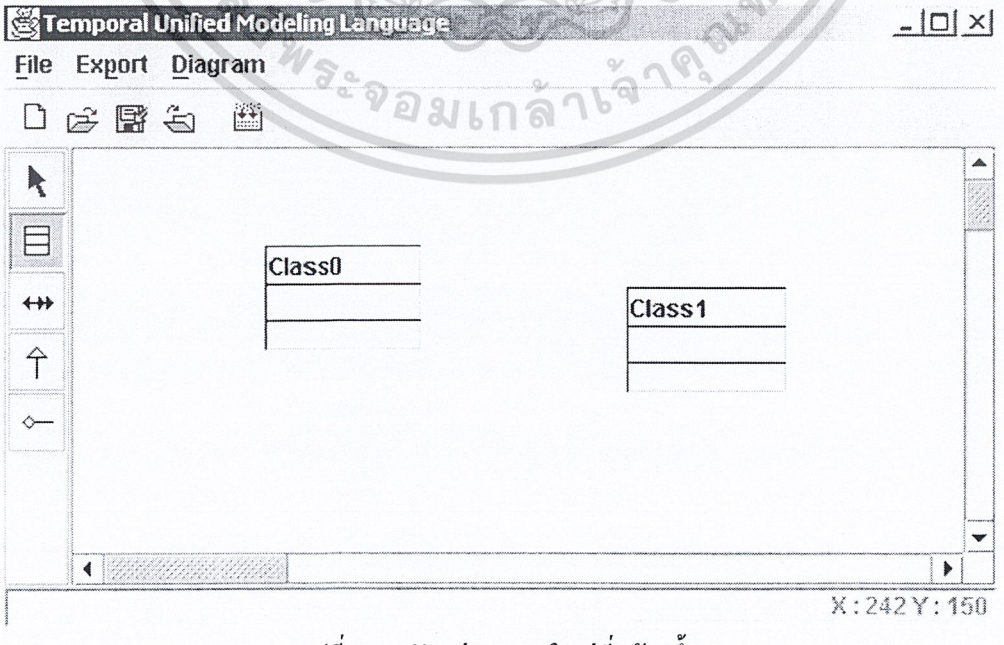
เมื่อเปิดโปรแกรมเข้าสู่ระบบอัตโนมัติในการแปลงรูป TCD ไปเป็นภาษา ODL ก็จะปรากฏหน้าจอดังรูป 8-17



รูปที่ 8-17หน้าจอหลัก

8.6.1 สร้างคลาสใหม่

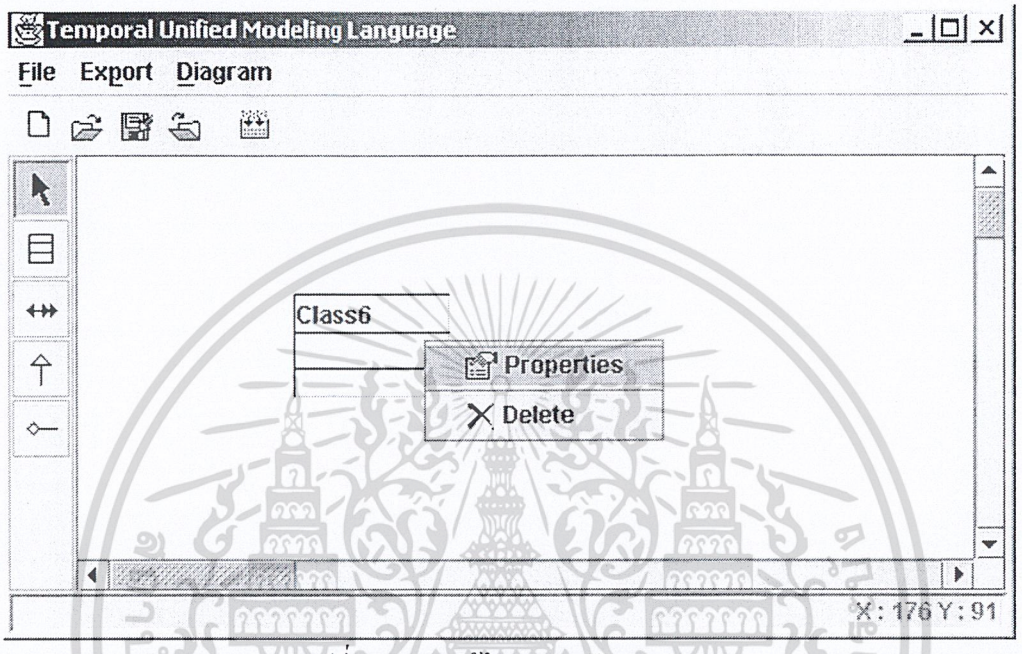
สร้างคลาสใหม่ โดยคลิกที่ปุ่ม Class Button จากนั้นเลื่อนเมาส์ Pointer ไปใน Draw Area ตำแหน่งที่ต้องการสร้างคลาสแล้วคลิกเมาส์อีกครั้ง จะได้รูปคลาสปรากฏให้เห็นดังแสดงในรูปที่ 8-18



รูปที่ 8-18 ตัวอย่างคลาสใหม่ที่สร้างขึ้น

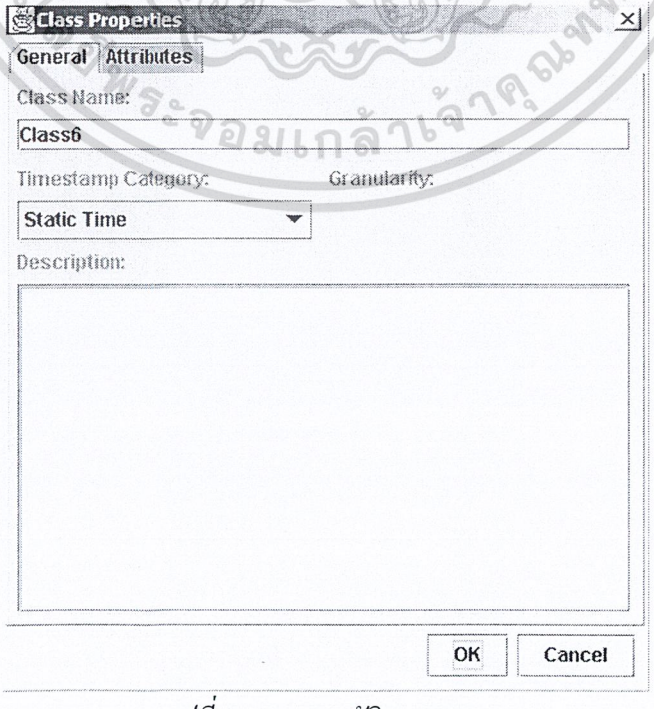
8.6.2 การเพิ่มแอตทริบิวต์ และแก้ไขคุณสมบัติของคลาส

เมื่อต้องการแก้ไขคุณสมบัติของคลาส โดยการคลิกที่ปุ่ม Select Button จากนั้นดับเบิ้ลคลิกที่คลาส หรือคลิกขวาที่คลาส เรียก Popup Menu ให้แสดงขึ้นมาดังรูปที่ 8-19 แล้วเลือกหัวข้อ Properties จะปรากฏ Class Properties Dialog ขึ้นมา ดังที่แสดงในรูปที่ 8-20



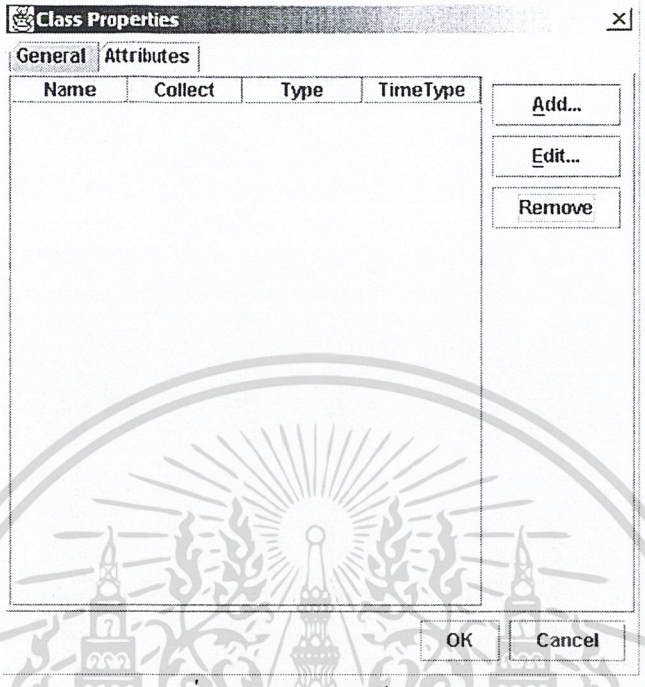
รูปที่ 8-19 การแก้ไขคุณสมบัติของคลาส

เลือกแท็บ General เพื่อแก้ไขคุณสมบัติต่างๆ ของคลาสแสดงดังรูปที่ 8-20



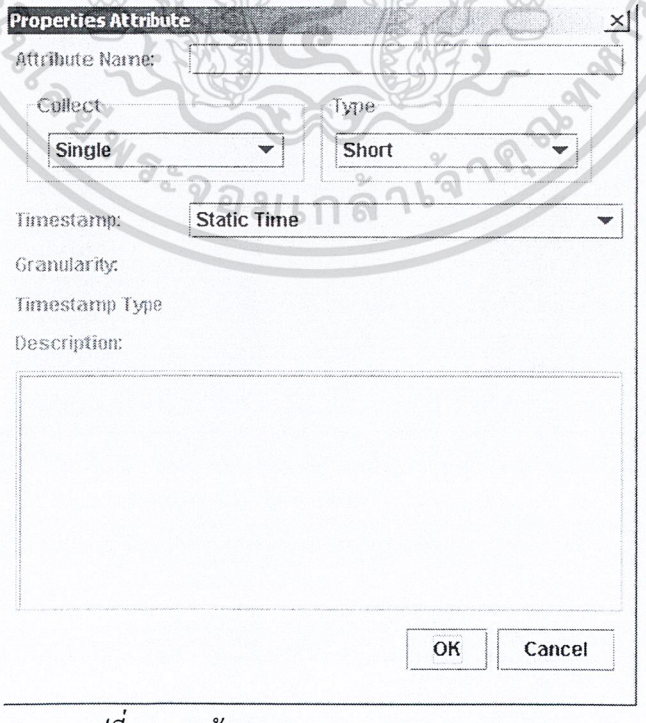
รูปที่ 8-20 คุณสมบัติของคลาส

เลือกแท็บ Attributes เพื่อเพิ่ม ลบ และแก้ไขแอตทริบิวต์ของคลาสแสดงดังรูปที่ 8-21



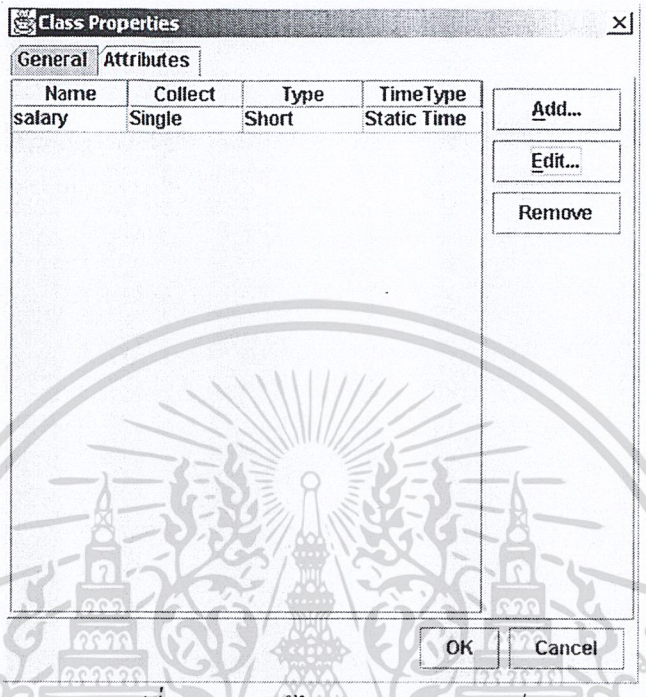
รูปที่ 8-21 แอตทริบิวต์ของคลาส

เพิ่มแอตทริบิวต์ใหม่โดยคลิกที่ปุ่ม Add จะปรากฏ Attribute Properties Dialog ขึ้นมาให้ป้อนรายละเอียดของแอตทริบิวต์แล้วคลิกที่ปุ่ม OK แสดงดังรูปที่ 8-22



รูปที่ 8-22 หน้าจอ Attribute Properties Dialog

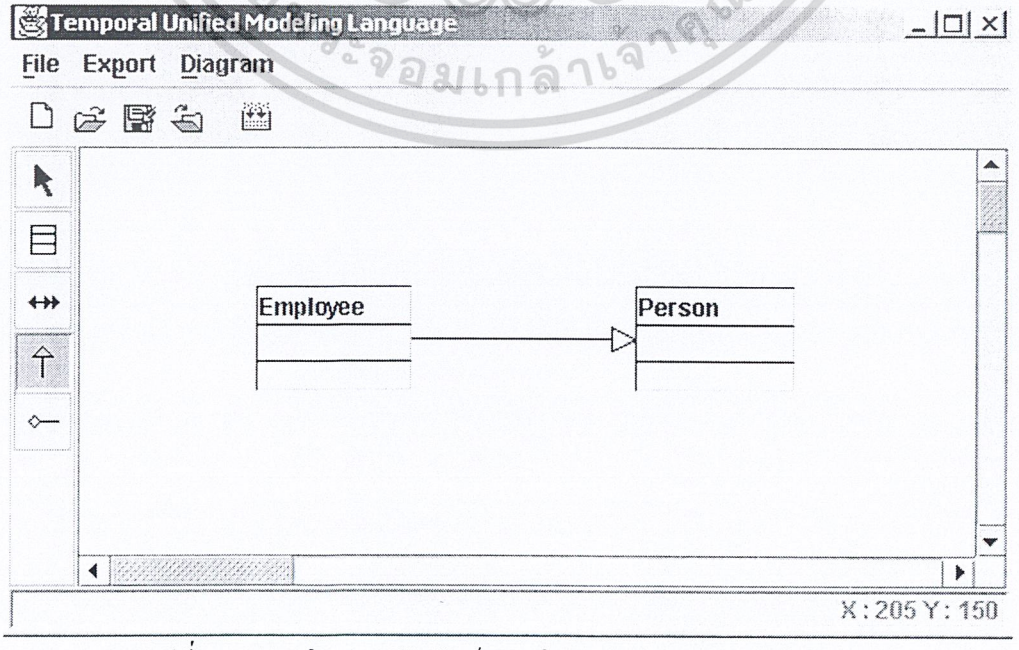
แก้ไขหรือลบ แอตทริบิวต์ที่ต้องการโดย คลิกเลือกที่ชื่อแอตทริบิวต์ของคลาสที่แท็บ Attributes นั้น แล้วคลิกปุ่ม Edit เมื่อต้องการแก้ไขจะปรากฏ Attribute Properties Dialog ของแอตทริบิวต์นั้น หรือ คลิกปุ่ม Remove เมื่อต้องการลบแอตทริบิวต์ จะแสดงดังรูป 8-23



รูปที่ 8-23 การแก้ไขและลบแอตทริบิวต์

8.6.3 การสร้างความสัมพันธ์แบบถ่ายทอดคุณสมบัติ

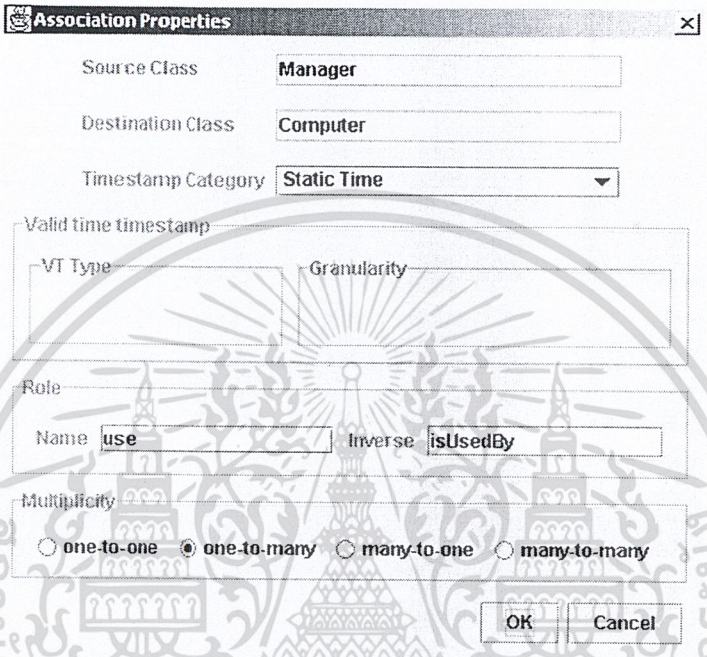
สร้างความสัมพันธ์แบบถ่ายทอดคุณสมบัติจากคลาสแม่ไปคลาสลูก โดยคลิกที่ปุ่ม Generalization จากนั้นเลือกคลาสต้นแบบ (Super Class) โดยคลิกขวาค้างไว้ แล้วเลื่อน Mouse Pointer ไปที่คลาสที่จะสืบทอดคุณสมบัติ (Sub Class) แล้วปล่อยปุ่มเมาส์ที่คลิกค้างไว้ จะได้เส้นแสดงความสัมพันธ์ Generalization ระหว่าง 2 คลาส แสดงดังรูปที่ 8-24



รูปที่ 8-24 ตัวอย่างความสัมพันธ์แบบถ่ายทอดคุณสมบัติระหว่างคลาส

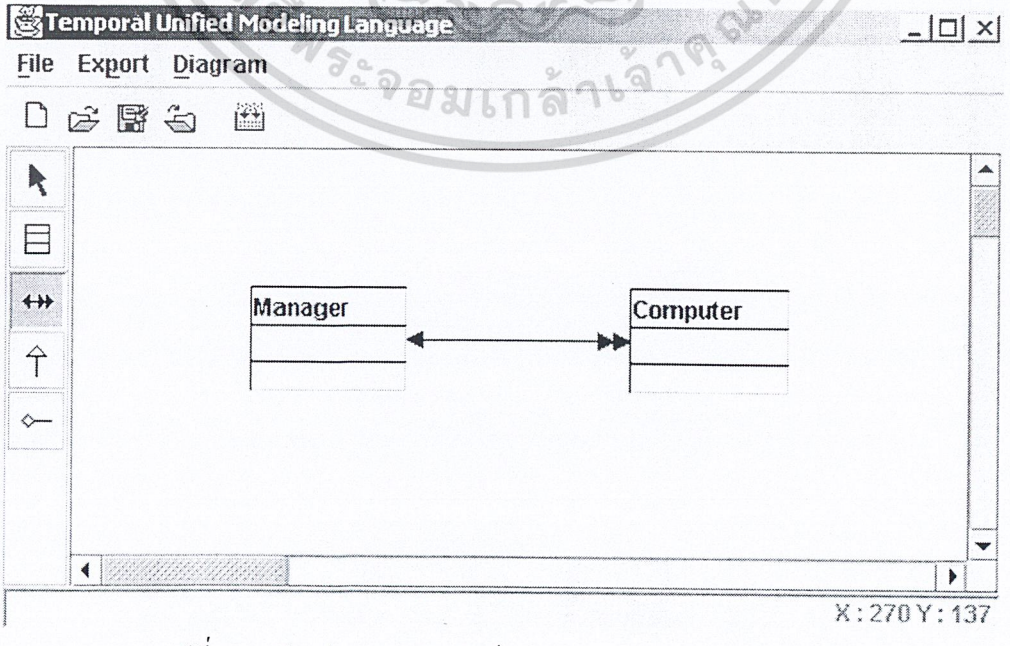
8.6.4 การสร้างความสัมพันธ์แบบ Association

สร้างความสัมพันธ์แบบ Association ระหว่าง 2 คลาส โดยคลิกที่ปุ่ม Association Button จากนั้นเลือกคลาสต้นทางโดยคลิกขวาค้างไว้ แล้วเลื่อน Mouse Pointer ไปที่คลาสปลายทาง แล้วปล่อยปุ่มเมาส์ จะปรากฏ Association Properties Dialog ขึ้นมาให้ป้อนข้อมูลรายละเอียดของความสัมพันธ์ แสดงดังรูปที่ 8-25



รูปที่ 8-25 หน้าจอ Association Properties Dialog

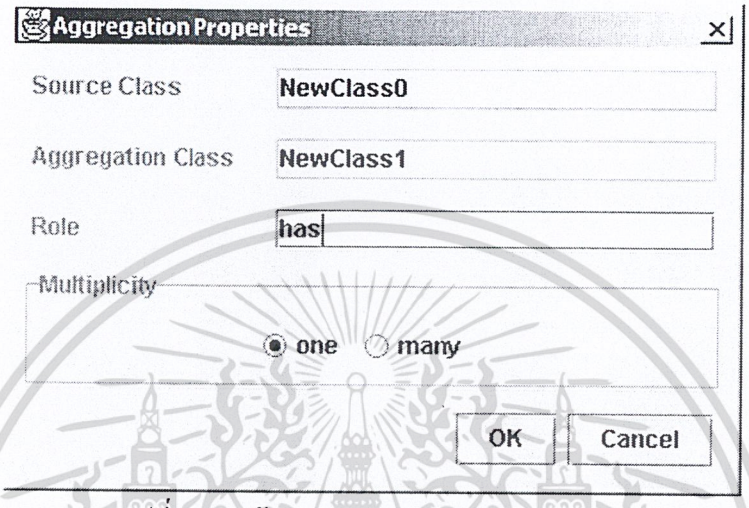
หลังจากคลิกที่ปุ่ม OK จะได้รูปความสัมพันธ์แบบ Association ระหว่างคลาส แสดงดังรูปที่ 8-26



รูปที่ 8-26 ตัวอย่างความสัมพันธ์แบบ Association ระหว่างคลาส

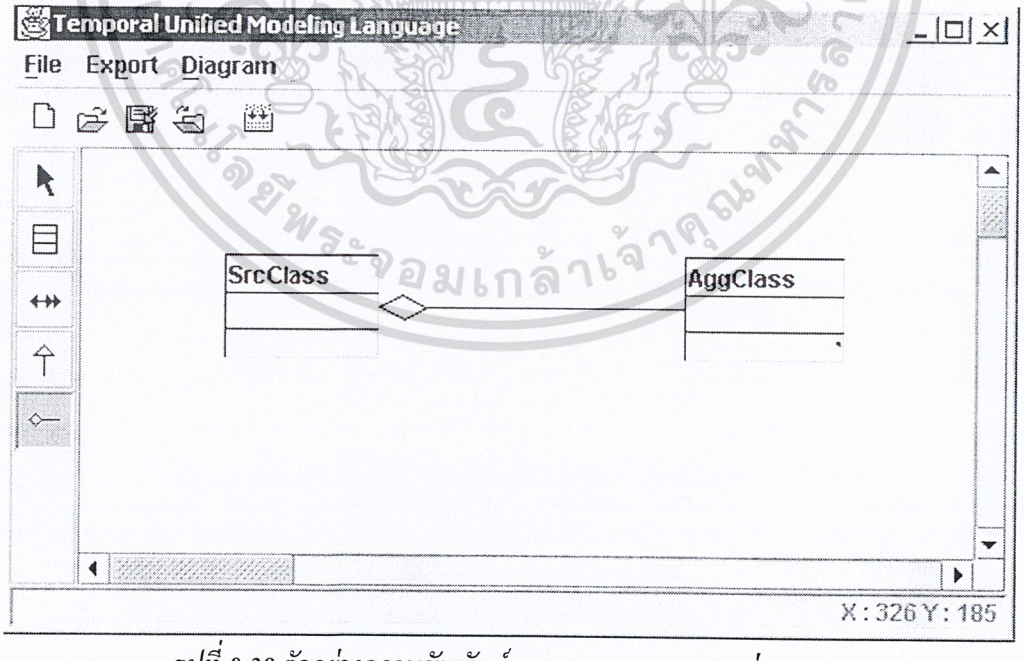
8.6.5 การสร้างความสัมพันธ์แบบ Aggregation

สร้างความสัมพันธ์แบบ Aggregation ระหว่าง 2 คลาส โดยคลิกเลือกที่ปุ่ม Aggregation จากนั้นเลือกคลาสต้นทางโดยขวาค้างไว้ แล้วเลื่อน Mouse Pointer ไปที่คลาสปลายทาง แล้วปล่อยปุ่มเมาส์ จะปรากฏ Aggregation Properties Dialog ขึ้นมาให้ป้อนรายละเอียดของความสัมพันธ์ แสดงดังรูปที่ 8-27



รูปที่ 8-27 หน้าจอ Aggregation Properties Dialog

หลังจากคลิกที่ปุ่ม OK จะได้รูปความสัมพันธ์แบบ Aggregation ระหว่างคลาส ดังรูปที่ 8-28



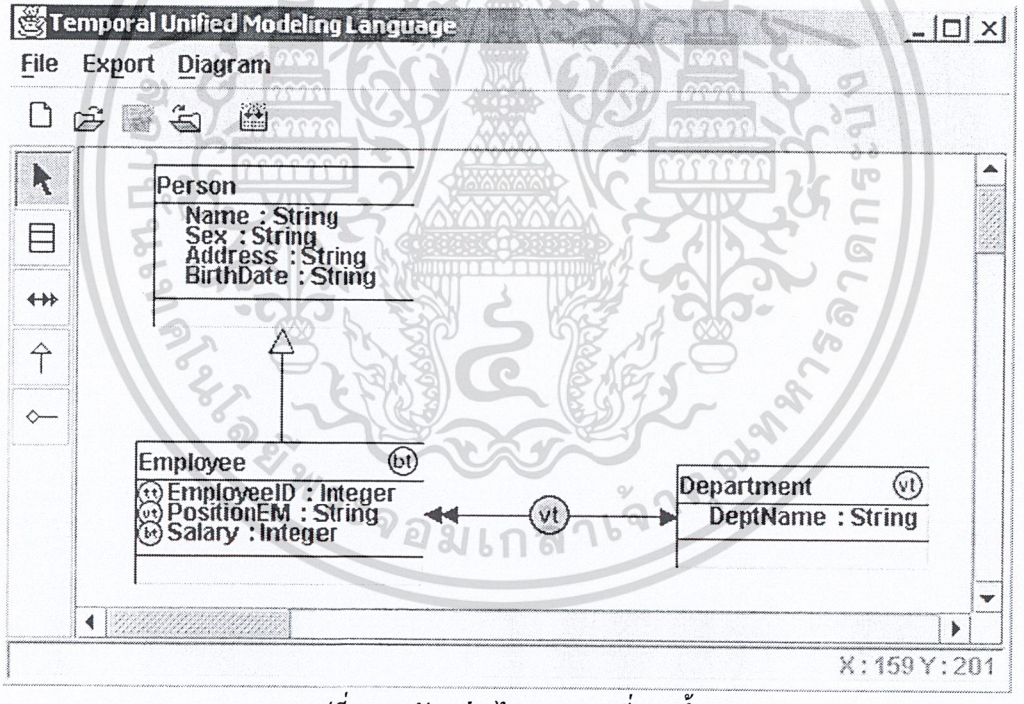
รูปที่ 8-28 ตัวอย่างความสัมพันธ์แบบ Aggregation ระหว่างคลาส

8.6.6 การแปลง TCD ไปเป็นภาษา ODL

เมื่อต้องการแปลงไดอะแกรมที่วาดขึ้นไปเป็นภาษานิยามเชิงวัตถุ ทำได้โดยการ Export ออกมาเป็น ไฟล์ได้เพื่อที่จะนำไปใช้งานต่อไป โดยการคลิกเลือกที่เมนู Export ซึ่งจะมีหัวข้อย่อยให้เลือกดังนี้

- ODL (Object Definition Language) เป็นการ Export ออกมาเป็นภาษามาตรฐานในการสร้างคลาสเพื่อใช้ในระบบ ฐานข้อมูลเชิงวัตถุ ตามมาตรฐาน ODMG 2.0
- TODL (Temporal Object Definition Language) เป็นการ Export ออกมาเป็นภาษามาตรฐานเช่นเดียวกันแต่มีการเพิ่มคุณสมบัติทางเวลาเข้าไปด้วย
- CDL (Class Definition Language) เป็นการ Export ออกมาเป็นภาษาที่ใช้ในฐานข้อมูล Caché โดยสามารถนำไฟล์ที่ Export ออกมานี้ไป Import เข้าฐานข้อมูล Caché ได้
- TCDL (Temporal Class Definition Language) เช่นเดียวกับ CDL แต่จะมีการเพิ่มคุณสมบัติทางเวลาเข้าไปด้วย

ตัวอย่างไดอะแกรมที่ทำการแปลงไปเป็นภาษานิยามเชิงวัตถุ แสดงดังรูป 8-29



รูปที่ 8-29 ตัวอย่างไดอะแกรมที่วาดขึ้น

เมื่อทำการแปลงเป็นภาษานิยามเชิงวัตถุแล้วจะได้ดังนี้

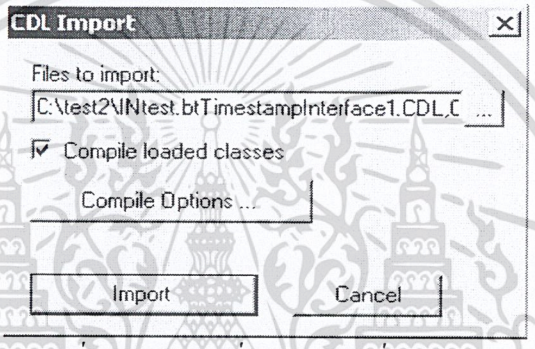
```
interface Employee : Person
( extent Employees )
{
   attribute Temporal_Element<SECOND> tt_lifespan;
   typedef struct{Period<SECOND> tt_timestamp;
```

```
    Temporal_Element<DAY> vt_lifespan} VT_Lifespan_Element;
    attribute Set<VT_Lifespan_Element> vt_lifespan;
    attribute Set<struct element{Integer value,Period<SECOND> tt_timestamp}> EmployeeID;
    attribute Set<struct element{String value,Period<DAY> vt_timestamp}> PositionEM;
    attribute Set<struct element{Integer value,Period<SECOND> tt_timestamp,
                                 Period<DAY> vt_timestamp}> Salary;
    relationship<P,YEAR> Set<Employee_workFor_Department_employ> workFor
        inverse Employee_workFor_Department_employ::i_workFor;
};

interface Person
( extent Persons )
{
    attribute String Name;
    attribute String Sex;
    attribute String Address;
    attribute String BirthDate;
};

interface Department
( extent Departments )
{
    attribute Temporal_Element<YEAR> vt_lifespan;
    attribute String DeptName;
    relationship<P,YEAR> Set<Employee_workFor_Department_employ> employ
        inverse Employee_workFor_Department_employ::i_employ;
};

interface Employee_workFor_Department_employ
( extent Employee_workFor_Department_employs )
{
    attribute Temporal_Element<YEAR> vt_lifespan;
    relationship Employee i_workFor
```

```
        inverse Employee::workFor;
    relationship Department i_employ
        inverse Department::employ;
};
```

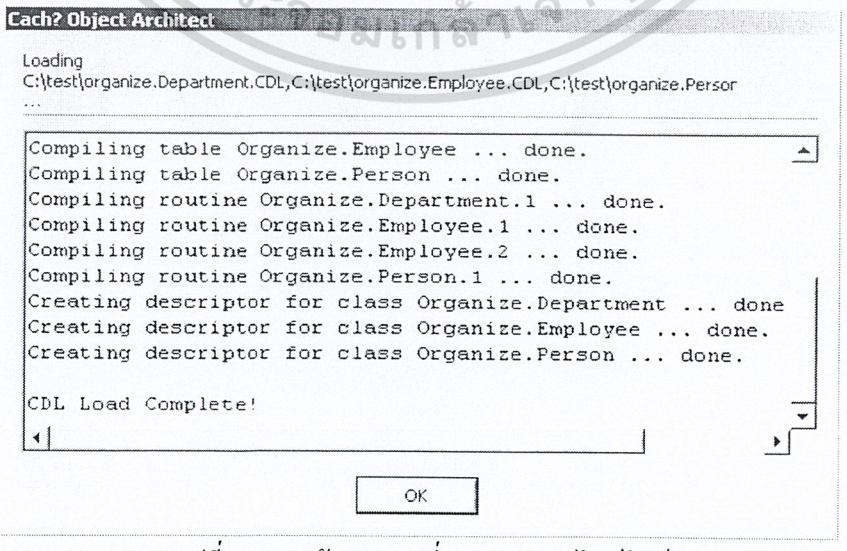
8.6.7 การนำภาษานิยามเชิงวัตถุไปใช้ในฐานข้อมูล Caché

เปิดโปรแกรม Object Architect ซึ่งเป็นเครื่องมือที่ Caché มีมาให้ ติดต่อผ่าน Connection Name จากนั้นเลือกเมนู File > Import > Import CDL จะปรากฏ CDL Import Dialog ขึ้นมาแสดงดังรูปที่ 8-30



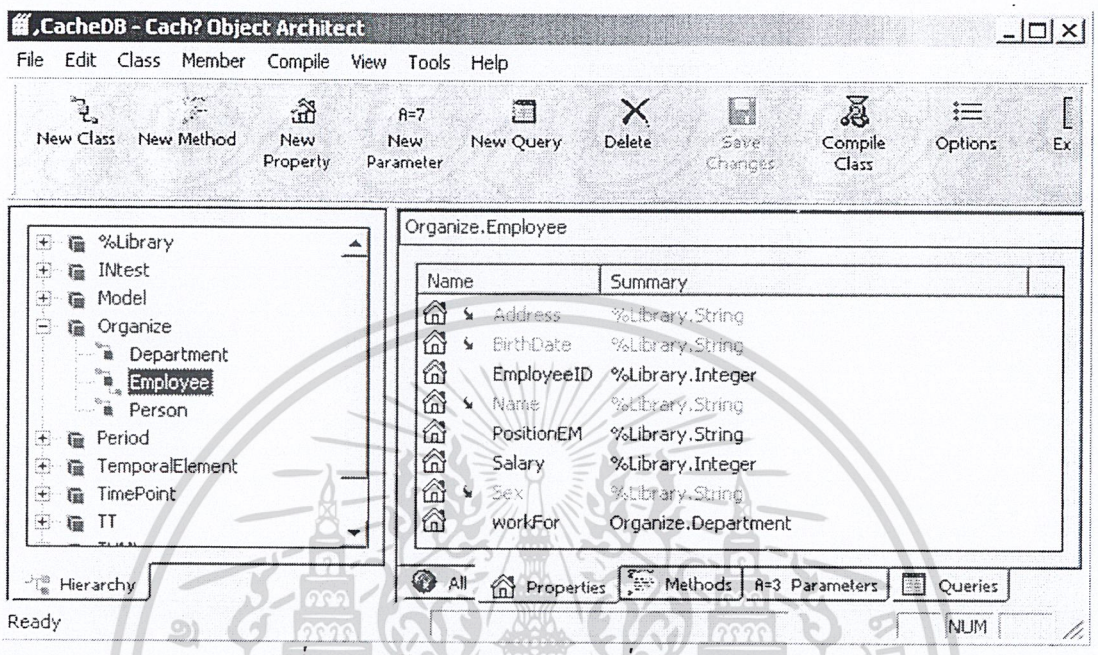
รูปที่ 8-30 หน้าจอที่ใช้เลือกไฟล์ที่จะ Import

เลือกไฟล์ที่ต้องการ Import โดยสามารถเลือกได้ทีละหลายไฟล์ แล้วคลิกปุ่ม Import จากนั้น Caché จะทำการคอมไพล์ไฟล์ที่เลือกไว้ เมื่อ Caché คอมไพล์เสร็จ Dialog ที่ Caché แสดงขณะคอมไพล์จะแสดงว่า "CDL Load Complete !" ดังรูปที่ 8-31



รูปที่ 8-31 หน้าจอขณะที่ Caché คอมไพล์ไฟล์

เมื่อ Import เรียบร้อยแล้ว จะได้คลาสตามที่วาดไว้ในไดอะแกรมที่เราออกแบบ ดังรูปที่ 8-32



รูปที่ 8-32 หน้าจอแสดง Package ที่ได้ จากการ Import

บทที่ 9
สรุปและวิจารณ์

9.1 สิ่งที่ทำได้

1. สามารถสร้างคลาสไดอะแกรมที่สามารถเพิ่มข้อมูลเชิงเวลาให้กับคลาสไดอะแกรมได้
2. สามารถแปลงรูปคลาสไดอะแกรมไปเป็นภาษานิยามเชิงวัตถุตามมาตรฐาน ODMG 2.0 และภาษานิยามคลาส เพื่อนำไปใช้กับระบบฐานข้อมูล InterSystems Caché ได้ ทั้งแบบที่มีการเพิ่มข้อมูลเชิงเวลา และไม่มีข้อมูลเชิงเวลา
3. สามารถจัดเก็บคลาสไดอะแกรมที่ได้ทำการออกแบบไว้ และนำกลับมาใช้งานใหม่ได้

9.2 สิ่งที่ทำไม่ได้

1. สำหรับความสัมพันธ์แบบ one-to-one และ many-to-many นั้น บนฐานข้อมูล Caché ไม่รองรับ ดังนั้นในกรณีที่ไดอะแกรมที่ออกแบบมีความสัมพันธ์ในลักษณะดังกล่าว จะไม่สามารถ Export เป็นภาษานิยามคลาส สำหรับ Caché ได้

9.3 ข้อดี

1. เป็นเครื่องมือที่ช่วยให้ผู้ใช้ สามารถออกแบบคลาสไดอะแกรมที่เพิ่มคุณสมบัติทางเวลาได้ง่ายเนื่องจากโปรแกรมมีลักษณะเป็น Graphic User Interface (GUI)
2. โปรแกรมสามารถ Export ไดอะแกรมที่ได้ออกแบบไว้ไปเป็นภาษานิยามเชิงวัตถุที่มีมาตรฐานรับรอง สามารถนำไปใช้ประโยชน์ได้ โดยที่ผู้ใช้ไม่จำเป็นที่จะต้องมีความรู้เกี่ยวกับภาษานิยามเชิงวัตถุเลย ตัวอย่างเช่น ถ้าผู้ต้องการนำไปใช้กับระบบฐานข้อมูล Caché ก็เพียงแต่ทำการ Export ไดอะแกรมออกมาเป็น Class Definition Langauge (CDL) ไฟล์แล้วนำไป Import เข้าไปในระบบฐานข้อมูล Caché ได้ทันที
3. ใช้สถาปัตยกรรมในการเชื่อมต่อแบบไคลเอนต์/เซิร์ฟเวอร์ โดยมีการจัดเก็บข้อมูลลงบนดาต้าเบสเซิร์ฟเวอร์ ทำให้การจัดการกับข้อมูลที่จัดเก็บนั้นมีความสะดวกและประสิทธิภาพมากขึ้น

9.4 ข้อเสีย

1. การนำภาษานิยามเชิงวัตถุไปใช้งานกับระบบฐานข้อมูลเชิงวัตถุอื่นๆ ที่ยังไม่รองรับมาตรฐาน ODMG 2.0 ผู้ใช้ต้องมีความรู้ในการแปลงภาษานิยามเชิงวัตถุไปเป็นภาษานิยามที่ระบบฐานข้อมูลเชิงวัตถุนั้นรองรับ
2. การจัดเก็บข้อมูลของโปรแกรมประยุกต์ต้องใช้งานร่วมกับระบบฐานข้อมูล Caché เท่านั้น จึงไม่สามารถนำโปรแกรมประยุกต์ไปใช้งานกับระบบฐานข้อมูลอื่นได้

3. ในเรื่องของการแสดงผลของไดอะแกรมนั้นอาจยังไม่ดีในบางจุดเช่น การแสดงบทบาทที่คลาสหนึ่งกระทำต่ออีกคลาสหนึ่ง นั้นยังไม่มีการแสดงให้เห็น

9.5 ข้อเสนอแนะ

สำหรับผู้ที่ต้องการจะพัฒนาควรจะศึกษาหลักทฤษฎีที่เกี่ยวข้องให้ดีเสียก่อน เช่นในเรื่องความสัมพันธ์แบบต่างๆ การแปลงจากรูปสัญลักษณ์ให้มาเป็นในลักษณะของภาษาที่จะนำไปใช้งานต่อ ลักษณะของโครงสร้างภาษาของระบบฐานข้อมูลเชิงวัตถุ

ส่วนที่ควรจะปรับปรุงก็คือส่วนที่เป็นข้อเสียและสิ่งที่ทำไม่ได้ เช่น

1. การแสดงผลของไดอะแกรมควรจะให้มีการแสดงบทบาทที่แต่ละคลาสมีความสัมพันธ์ต่อกันด้วยเพื่อให้ผู้ที่ออกแบบง่ายต่อการตรวจสอบ
2. การตรวจสอบคำเฉพาะของระบบฐานข้อมูลเชิงวัตถุควรตรวจสอบได้ ซึ่งจะทำให้ผู้ที่ออกแบบสามารถแก้ไขคำนั้นในไดอะแกรมได้ทันที
3. ในเรื่องของการจัดเก็บข้อมูลอาจให้สามารถทำออกมาเป็นไฟล์ข้อมูลได้ด้วย เพื่อเพิ่มความสะดวกในการใช้งาน
4. การกำหนด Operation ที่จะใช้ในการเรียกใช้เพื่อทำการบันทึกหรือเรียกค้นข้อมูลที่อยู่ในโครงสร้างคลาสนั้น ๆ
5. ความสามารถในการแปลงภาษานิยามเชิงวัตถุไปเป็นภาษานิยามใดที่ระบบฐานข้อมูลที่ยังไม่รองรับมาตรฐาน ODMG 2.0 สามารถนำไปใช้งานได้ ได้หลายๆ ระบบ ทั้งที่เป็นระบบฐานข้อมูลเชิงวัตถุ ฐานข้อมูลรีเลชั่นแนล หรือฐานข้อมูลเชิงวัตถุสัมพันธ์

ภาคผนวก ก

การติดตั้งระบบฐานข้อมูล Caché บนระบบปฏิบัติการ Windows

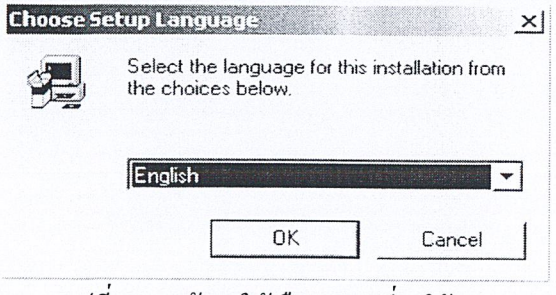
1. การเตรียมการติดตั้ง

ความต้องการของระบบในการติดตั้งระบบฐานข้อมูล Caché มีดังต่อไปนี้

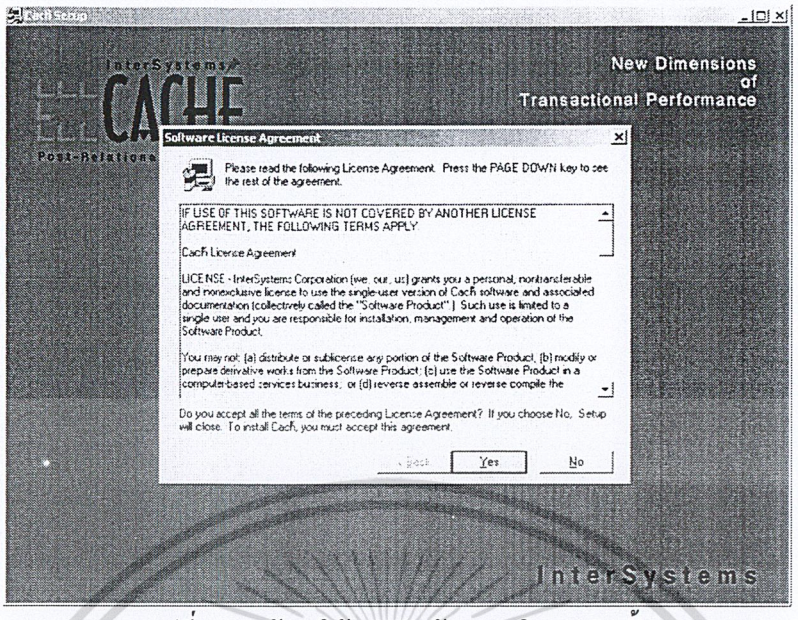
- Hardware
  - ใช้หน่วยประมวลผลกลางอย่างน้อย 486 ความเร็ว 50 mhz (ควรจะใช้ Pentium หรือ Alpha เพื่อการทำงานที่ดีขึ้น)
  - หน่วยความจำอย่างน้อย 32 Mb
  - พื้นที่ว่างบน Hard disk อย่างน้อย 110 Mb ซึ่งยังไม่ได้รวมพื้นที่ที่ใช้เก็บข้อมูล
- Software
  - ระบบปฏิบัติการที่รองรับ Window 95/8 Windows NT 4.0 (with Microsoft Service Pack 5 or higher), Windows ME, Windows 2000
  - ระบบการเชื่อมต่อเครือข่ายแบบ TCP/IP
- เงื่อนไขของ Directory ที่จะทำการติดตั้งมีดังต่อไปนี้
  - จะต้องไม่มีเครื่องหมาย caret (^) ใน pathname
  - ต้องเป็นตัวอักษรตามมาตรฐาน US ASCII
  - ต้องไม่ใช่ Root Directory (เช่น C:\ )
  - ต้องไม่ใช่ ***drive*:\Program Files**
  - มีตัวอักษรใน pathname มากกว่า 32 ตัวอักษร

2. ขั้นตอนการติดตั้ง

2.1 Run Setup.exe เพื่อทำการติดตั้ง จะมีหน้าจอขึ้นมาให้เลือกภาษาที่จะทำการติดตั้งแสดงในรูปที่ ก-1 ให้เลือก English แล้วกด OK จะมีหน้าจอขึ้นมาถามว่าต้องการทำตามข้อตกลงหรือไม่ ให้กด Yes แสดงในรูปที่ ก-2

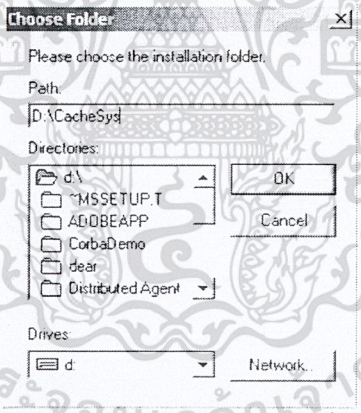


รูปที่ ก-1 หน้าจอให้เลือกภาษาที่จะใช้



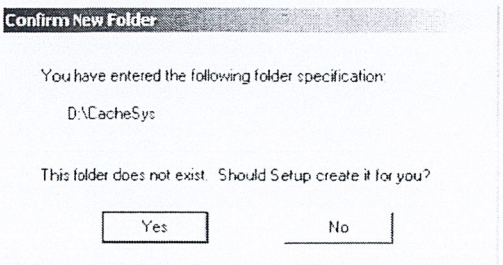
รูปที่ ก-2 หน้าจอให้ยอมรับข้อตกลงในการติดตั้ง

2.2 จากนั้นจะแสดงหน้าจอให้เลือก Directory ที่จะทำการติดตั้ง แสดงในรูปที่ ก-3 เมื่อเลือกแล้วให้กด OK



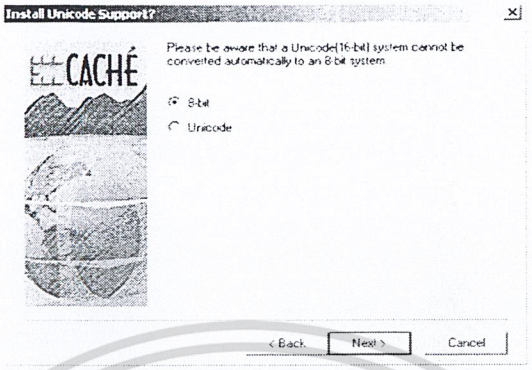
รูปที่ ก-3 หน้าจอให้เลือก Directory ที่จะติดตั้ง

2.3 หาก Directory ที่ต้องการจะติดตั้งไม่มีอยู่โปรแกรมจะแสดงหน้าจอขึ้นมาถามว่าต้องการที่จะสร้าง Directory นี้หรือไม่ ดังแสดงในรูปที่ ก-4 ถ้าต้องการให้กด Yes



รูปที่ ก-4 หน้าจอให้เลือกสร้าง Directory ใหม่เมื่อ Directory ที่ต้องการติดตั้งนั้นไม่มีอยู่

2.4 จากนั้นจะมีหน้าจอแสดงขึ้นมาให้เลือกรหัสในการบันทึกข้อมูลที่จะใช้ให้เราเลือก 8 bit ดังแสดงในรูป ก-5



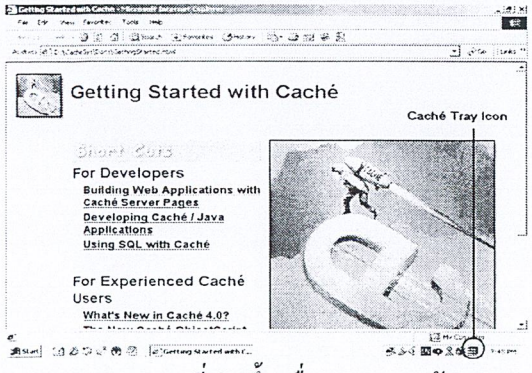
รูปที่ ก-5 หน้าจอให้เลือกรหัสในการบันทึกข้อมูล

2.5 จากนั้นจะมีหน้าจอแสดงขึ้นมาแสดงรายละเอียดเกี่ยวกับการ Install ที่ได้กรอกไปให้เรากด Next เพื่อเข้าสู่ขั้นตอนการติดตั้ง

2.6 เมื่อติดตั้งเสร็จแล้วจะมีหน้าจอแสดงขึ้นมาดังรูปที่ ก-6 ให้กด Finish

รูปที่ ก-6 หน้าจอเมื่อการติดตั้งสมบูรณ์

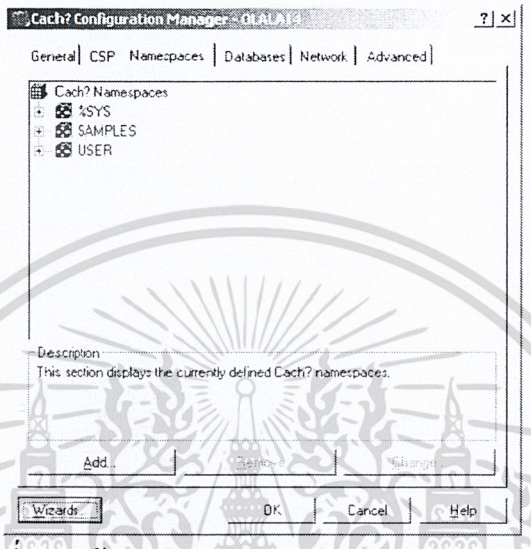
2.7 เมื่อการติดตั้งสมบูรณ์ Caché จะทำการ Start ตัวเองขึ้นมาโดยผู้ใช้สามารถเรียกใช้ระบบฐานข้อมูล Caché ได้จาก Caché Tray Icon ที่เห็นดังรูปที่ ก-7 ได้



รูปที่ ก-7 Caché Tray Icon ที่เกิดขึ้นเมื่อระบบฐานข้อมูล Caché เริ่มทำงาน

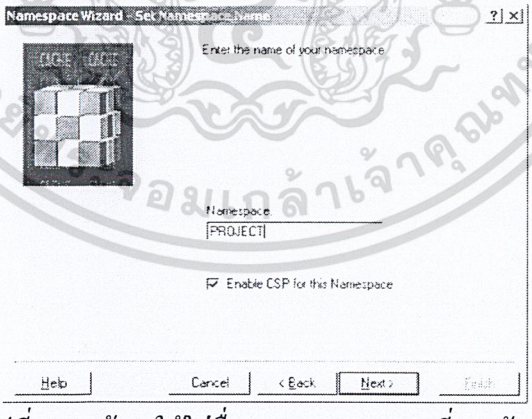
3. การตั้งค่าระบบฐานข้อมูล Caché เพื่อให้สามารถใช้กับโปรแกรม TUML ได้

3.1 ขั้นแรกให้เราใช้เมาส์คลิกขวาที่ Caché Tray Icon เลือกที่หัวข้อ Configuration Manager จากนั้นเลือกไปที่แท็ป Namespace ดังรูปที่ ก-8



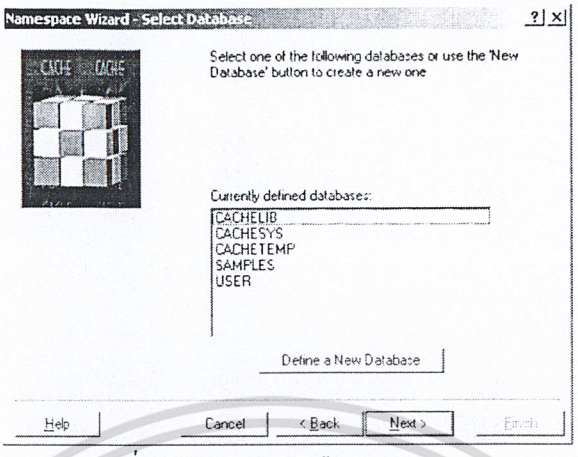
รูปที่ ก-8 หน้าจอ Configuration Manager ของ Caché

3.2 เลือกที่ Add เพื่อเพิ่ม Namespaces จะมีหน้าจอขึ้นมาให้กรอกชื่อของ Namespaces ที่จะสร้าง โดยให้ตั้งชื่อว่า PROJECT ดังแสดงในรูปที่ ก-9 แล้วคลิก Next

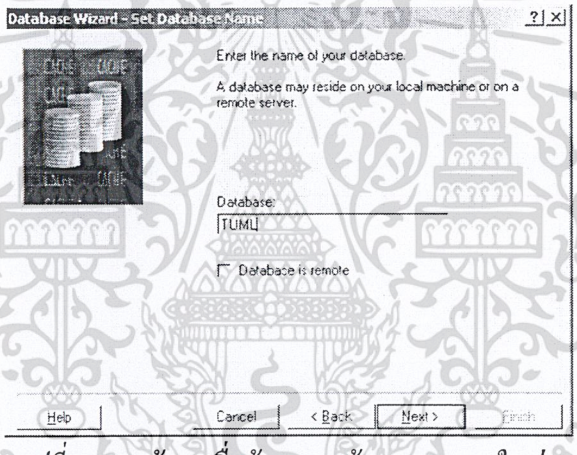


รูปที่ ก-9 หน้าจอให้ใส่ชื่อของ Namespaces ที่จะสร้าง

3.3 จากนั้นจะมีหน้าจอแสดงขึ้นมาให้เลือก Database ที่จะใช้เก็บข้อมูลดังรูปที่ ก-10 ในกรณีที่มีอยู่แล้วให้เลือก Database นั้นแล้วคลิก Next ในกรณีที่ยังไม่มีให้คลิกที่ Define a New Database เพื่อสร้างDatabase ขึ้นมาใหม่สำหรับเก็บข้อมูลใหม่ดังรูป ก-11 โดยในที่นี้ตั้งชื่อว่า TUML (สามารถตั้งชื่ออื่นได้)

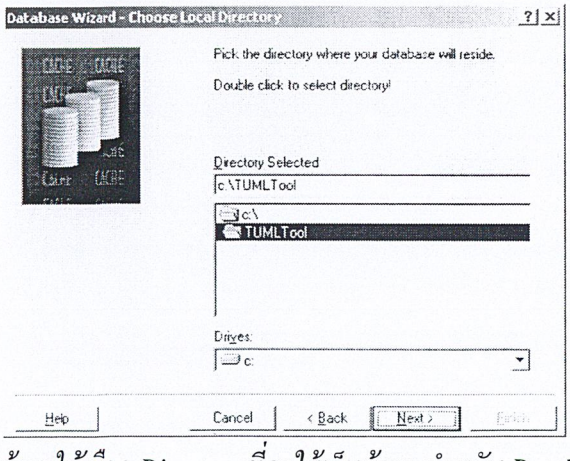


รูปที่ ก-10 หน้าจอให้เลือก Database



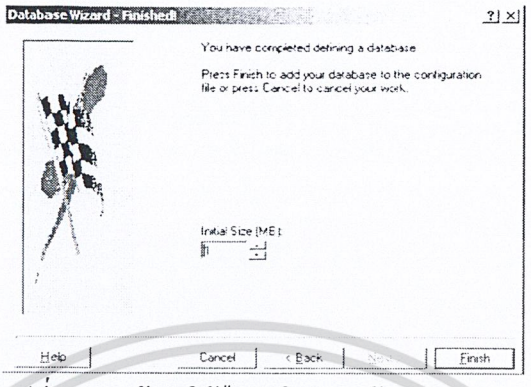
รูปที่ ก-11 หน้าจอเมื่อต้องการสร้าง Database ใหม่

3.4 จากนั้นจะมีหน้าจอแสดงขึ้นมาเพื่อให้เลือก Directory ที่จะเก็บข้อมูลของ Database ที่สร้างขึ้น ในที่นี้เลือกไปที่ C:\TUMLTool แสดงในรูปที่ ก-12 จากนั้นให้คลิก Next



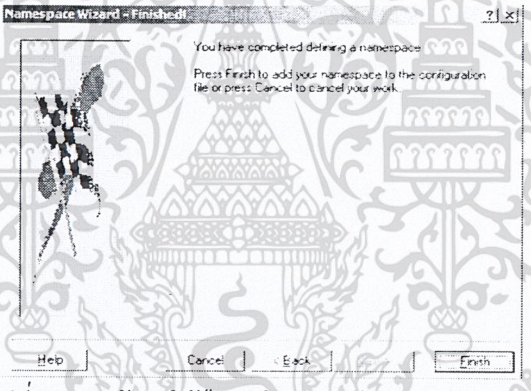
รูปที่ ก-12 หน้าจอให้เลือก Directory ที่จะใช้เก็บข้อมูลสำหรับ Database ที่สร้าง

3.5 โปรแกรมจะแสดงหน้าจอเพื่อขอยืนยันการสร้าง Database และถามค่า Initial Size ให้เราคลิกที่ Finish



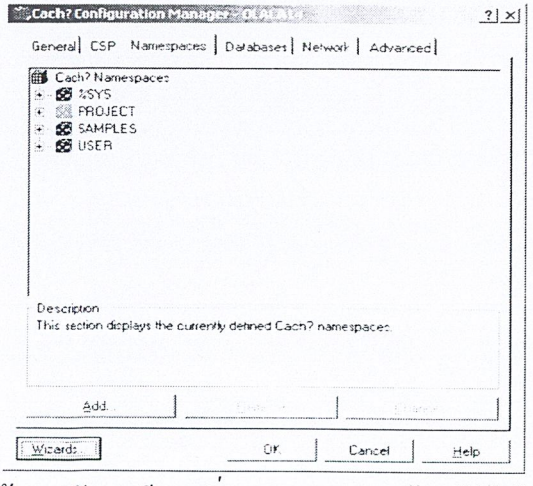
รูปที่ ก-13 หน้าจอให้ยืนยันในการสร้าง Database

3.6 โปรแกรมแสดงหน้าจอเพื่อขอยืนยันการสร้าง Namespaces ให้เราคลิก Finish



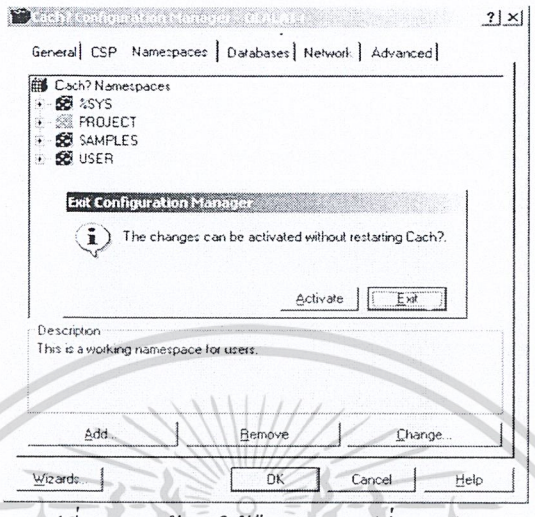
รูปที่ ก-14 หน้าจอให้ยืนยันในการสร้าง Namespaces

3.7 เมื่อสร้างแล้วเราจะเห็นชื่อของ Namespaces ที่เราสร้างขึ้นแต่ Namespaces นี้ยังไม่สามารถใช้งานเราต้องทำการ Activated เสียก่อนโดยการคลิก OK แสดงในรูปที่ ก-15



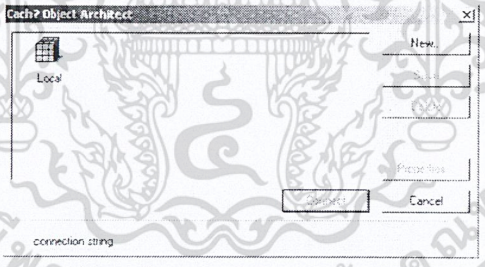
รูปที่ ก-15 หน้าจอหลังจากมีการเพิ่ม Namespaces แล้วแต่ยังไม่ได้ Activated

3.8 โปรแกรมจะแสดงหน้าจอเพื่อถามว่าต้องการให้สิ่งที่เปลี่ยนแปลงไปนั้นมีผลหรือไม่ ให้เราคลิกที่ Activated



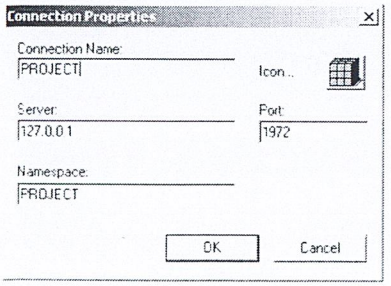
รูปที่ ก-16 หน้าจอให้ยืนยันการเปลี่ยนแปลง

3.9 หลังจากที่เราสร้าง Namespaces แล้วให้คลิกขวาที่ Caché Tray Icon เลือกที่หัวข้อ Object Architect โปรแกรมจะแสดงหน้าจอดังรูปที่ ก-17 เนื่องจากตอนนี้ยังไม่มีการสร้าง Connection ให้เชื่อมต่อกับ Namespaces ที่เราสร้างขึ้นมา เราจะสร้างโดยการคลิกที่ New



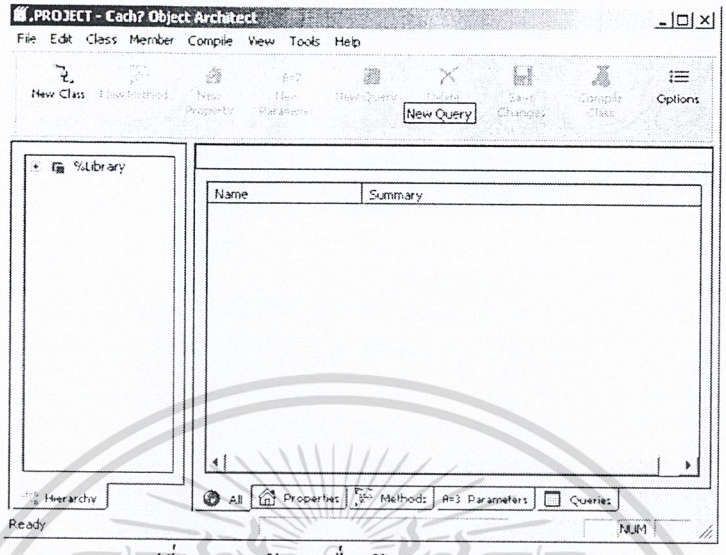
รูปที่ ก-17 หน้าจอก่อนเข้าสู่โปรแกรม Object Architect

3.10 โปรแกรมจะแสดง Dialog ขึ้นมาให้กรอกข้อมูล โดยที่ข้อมูลส่วนที่เป็น Connection Name กับ Namespace ให้กรอกชื่อ Namespace ที่ต้องการเชื่อมต่อด้วยคือ PROJECT (ข้อมูลส่วนของ Connection Name สามารถเป็นอย่างอื่นได้) ดังรูปที่ ก-18 จากนั้นให้กด OK เพื่อยืนยันการสร้าง Connection



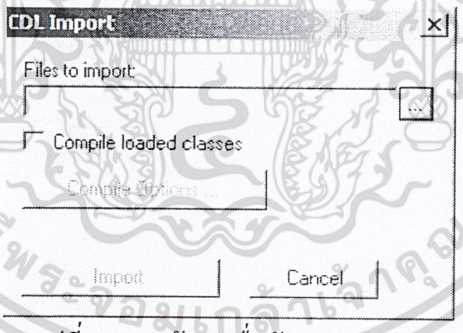
รูปที่ ก-18 หน้าจอให้กรอกข้อมูลการสร้าง Connection

3.11 จากนั้นทำการ Connect เข้าไปยัง Namespace ที่ชื่อ PROJECT แสดงดังรูปที่ ก-19



รูปที่ ก-19 หน้าจอเมื่อเข้า Object Architect

3.12 จากนั้นให้เราทำการ Import ไฟล์ที่จำเป็นต้องใช้สำหรับโปรแกรม TUMLTool โดยการเลือกที่เมนู File เลือกหัวข้อ Import เลือกที่ Import CDL จะมีหน้าจอแสดงขึ้นมาเพื่อให้ทำการเลือกไฟล์ที่จะ Import แสดงดังรูปที่ ก-20 ให้กดที่ปุ่ม ...



รูปที่ ก-20 หน้าจอเมื่อต้องการ Import

3.13 โปรแกรมแสดงหน้าจอขึ้นมาให้ทำการเลือกไฟล์ให้เราไปที่ Directory ที่เก็บไฟล์แล้วทำการเลือกโดยในการ Import นั้นสามารถ Import ทีละหลายไฟล์ได้ดังรูปที่ ก-21 เมื่อเลือกไฟล์แล้วให้ทำกด Open สำหรับไฟล์ที่จำเป็นมีดังต่อไปนี้

- ไฟล์สำหรับการบันทึกไดอะแกรม

  Model.AggregationLine.cdl, Model.AssociationLine.cdl, Model.Attribute.cdl,

  Model.Classes.cdl, Model.Diagrams.cdl, Model.InheritanceLine.cdl

- ไฟล์ที่ใช้ในการ Import ไฟล์ที่ได้จากโปรแกรม TUML ในกรณีที่ต้องการคุณสมบัติทางเวลาด้วย

  - ไฟล์ ของ Package Period

Period.GRYEAR.CDL, Period.GRMONTH.CDL, Period.GRDAY.CDL,

Period.GRHOUR.CDL,Period.GRMINUTE.CDL, Period.GRSECOND.CDL

- ไฟล์ของ Package Timepoint

TimePoint.GRYEAR.CDL, TimePoint.GRMONTH.CDL,

TimePoint.GRDAY.CDL, TimePoint.GRHOUR.CDL,

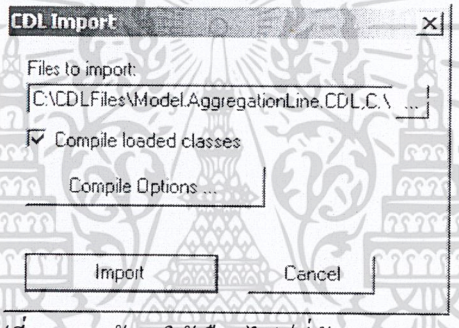
TimePoint.GRMINUTE.CDL, TimePoint.GRSECOND.CDL,

- ไฟล์ของ Package TemporalElement

TemporalElement.GRYEAR.CDL, TemporalElement.GRMONTH.CDL,

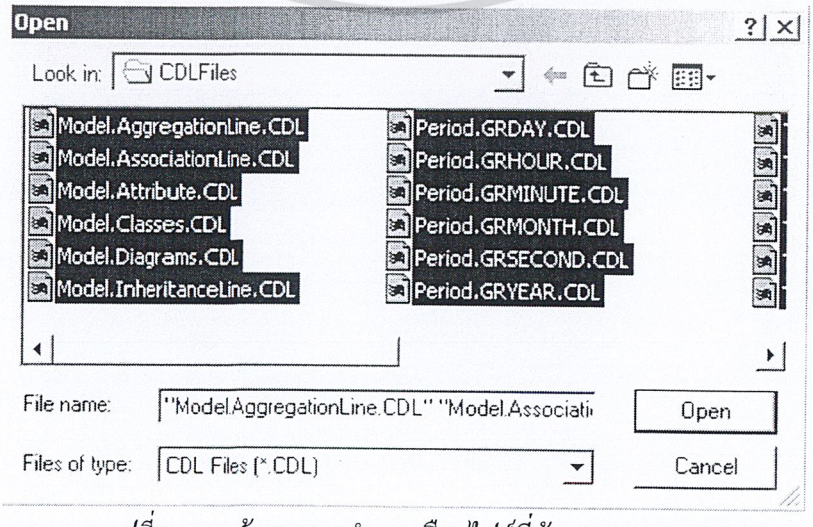
TemporalElement.GRDAY.CDL, TemporalElement.GRHOUR.CDL,

TemporalElement.GRMINUTE.CDL, TemporalElement.GRSECOND.CDL

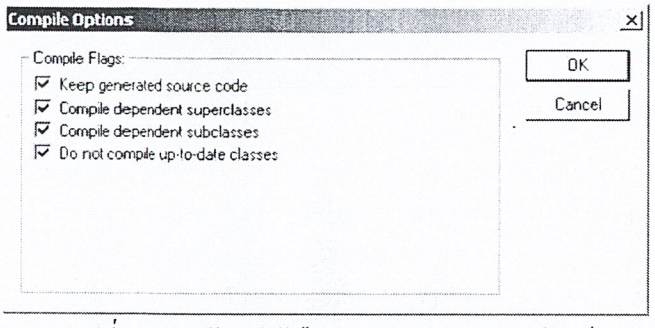


รูปที่ ก-21 หน้าจอให้เลือกไฟล์ที่ต้องการ Import

3.14 หลังจากที่เลือกไฟล์แล้วโปรแกรมจะกลับมาที่หน้าจอรูปที่ ก-20 แต่จะมีการแสดงชื่อไฟล์ที่จะทำการคอมไพล์ด้วย ให้เรา check เลือก Compile loaded classes ดังแสดงในรูปที่ ก-22 จากนั้นคลิกที่ Compile Options โปรแกรมจะแสดงหน้าจอให้เรากำหนด Option ในการคอมไพล์ให้เรา check เลือกดังรูปที่ ก-23 จากนั้นให้กด OK

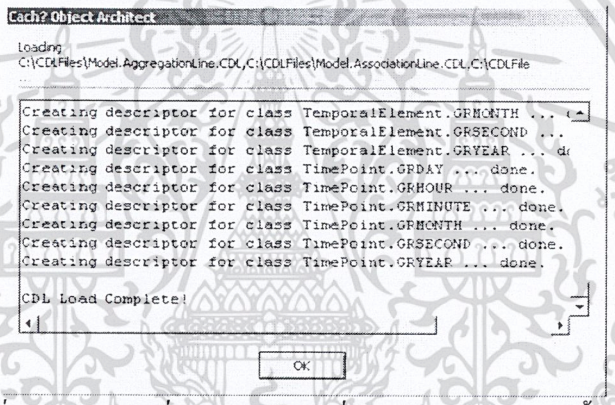


รูปที่ ก-22 หน้าจอขณะทำการเลือกไฟล์ที่ต้องการ Import



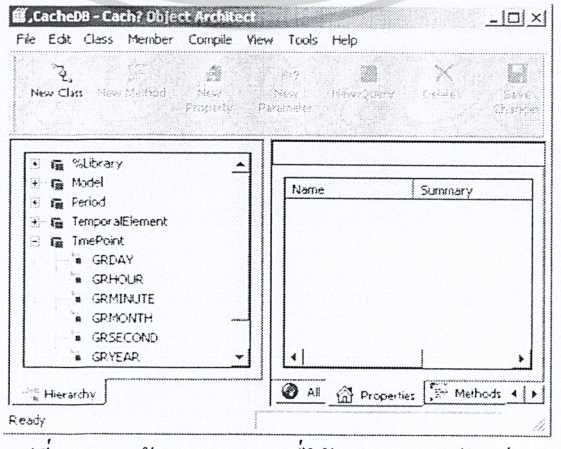
รูปที่ ก-23 หน้าจอให้เลือก Option ในการคอมไพล์

3.15 โปรแกรมจะย้อนกลับมาที่รูป ก-22 ให้เราทำการกด Import เพื่อทำการคอมไพล์ไฟล์ที่เลือกไว้ โดยขณะที่กำลังคอมไพล์โปรแกรมจะแสดงหน้าจอขึ้นมาและเมื่อคอมไพล์เสร็จแล้วจะมีข้อความว่า "CDL Load Complete" แสดงดังรูปที่ ก-24 ให้คลิกที่ OK เพื่อปิดหน้าจอนี้



รูปที่ ก-24 หน้าจอเมื่อคอมไพล์ไฟล์ที่ต้องการ Import เสร็จสิ้น

3.16 เมื่อคอมไพล์เสร็จสิ้นจะเห็น Package ที่ได้จากการคอมไพล์ในหน้าจอของ Object Architecture ซึ่งในที่นี้มี Package ที่เกิดขึ้นคือ Model, Period, TimePoint, TemporalElement และในแต่ละ Package ก็จะมีคลาสที่เราได้ Import เข้าไปดังแสดงในรูปที่ ก-25



รูปที่ ก-25 หน้าจอแสดงผลที่ได้จากการคอมไพล์

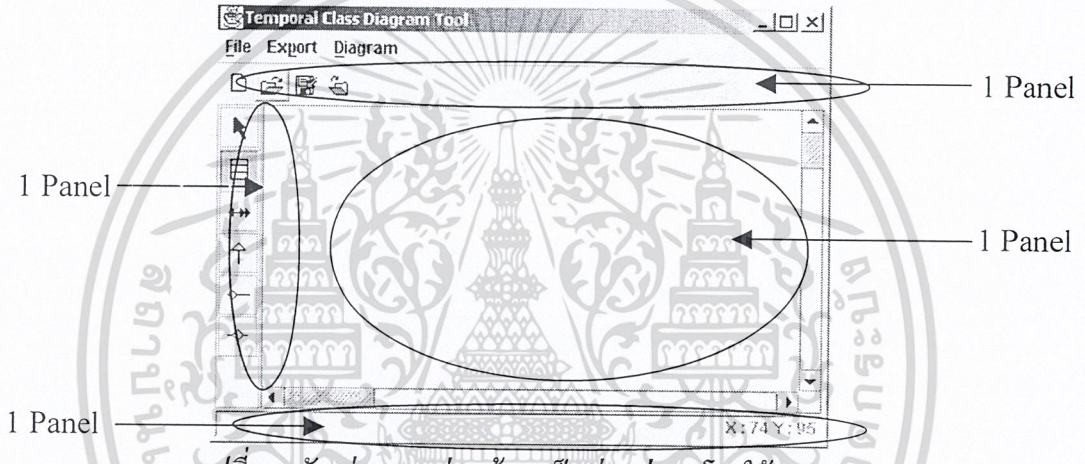
3.17 เสร็จสิ้นการติดตั้ง

ภาคผนวก ข

เทคนิคการเขียนโปรแกรม Java ในงาน Graphic

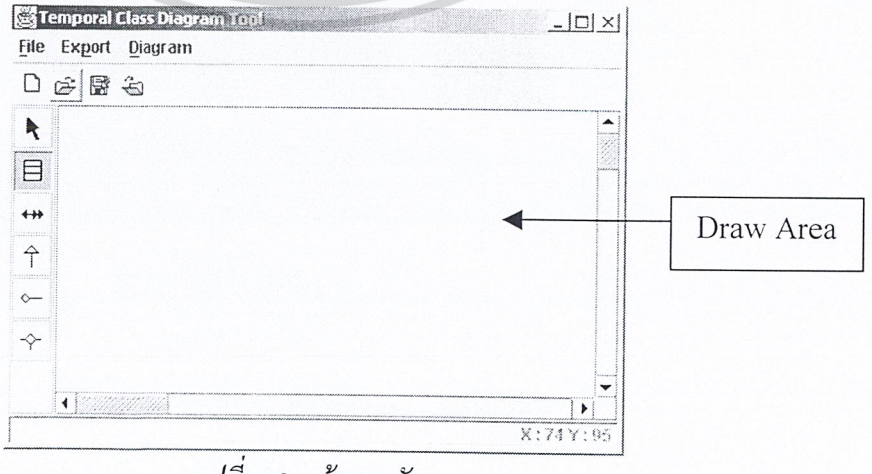
1. หลักการออกแบบ

โปรแกรมที่สร้างมีลักษณะเป็น Graphic User Interface (GUI) ซึ่งสามารถสร้างโดยการใช้ Frame และ Dialog ซึ่งภายใน Frame และ Dialog ก็จะทำการแบ่งส่วนโดยใช้ Panel เพื่อให้ง่ายต่อจัดวางปุ่ม และ Label ต่าง ๆ ที่จะแสดงบนหน้าจอ



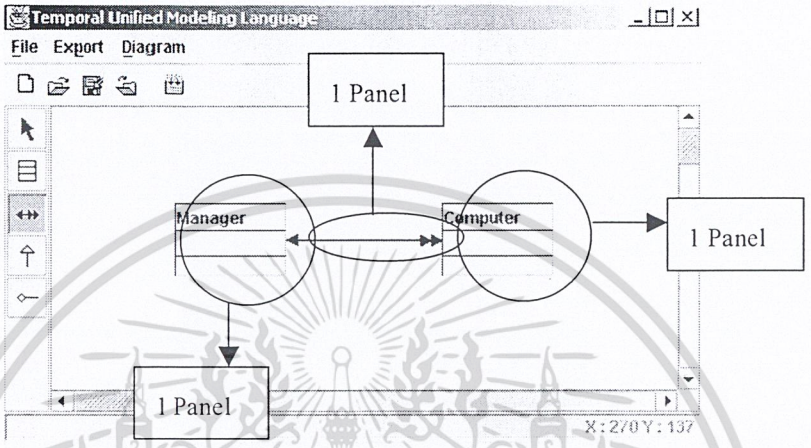
รูปที่ ข-1 ตัวอย่างการแบ่งหน้าจอเป็นส่วนต่าง ๆ โดยใช้ Panel

จากหน้าจอหลักของโปรแกรมจะเห็นว่ามีส่วนที่ใช้ในการวาดแผนภาพ (โดยจะขอเรียกส่วนนี้ว่า Draw Area) โดยในส่วนนี้จะใช้ Text Area ร่วมกับ JScrollPane โดยกำหนดให้ Text Area ไม่สามารถเขียนข้อความได้



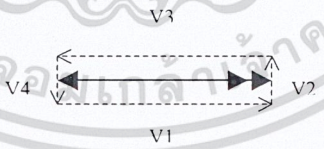
รูปที่ ข-2 หน้าจอหลัก

เมื่อมีการสร้างคลาสหรือความสัมพันธ์ การแสดงผลจะใช้การสร้าง Panel ขึ้นมาเพื่อแสดงคุณสมบัติของคลาสหรือความสัมพันธ์นั้น ๆ โดยจะทำการสร้าง 1 Panel ต่อ 1 คลาสหรือ 1 ความสัมพันธ์ ดังรูปที่ ข-3 แล้วเพิ่ม Panel นั้น ๆ เข้าไปใน Draw Area ซึ่งถ้าคลาสหรือเส้นความสัมพันธ์มีการเปลี่ยนแปลงขนาดก็ใช้วิธีการ setBound ของ Panel นั้นใหม่



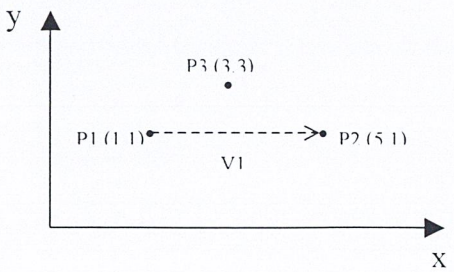
รูปที่ ข- 3 หน้าจอในการแบ่งแผนภาพออกเป็น Panel

หลังจากที่ทำการสร้างแผนภาพแล้วอาจต้องมีการย้ายตำแหน่งของคลาส รวมถึงการเปลี่ยนแปลงคุณสมบัติของคลาสและความสัมพันธ์ ดังนั้นจึงต้องมีการตรวจสอบว่า คลาสหรือความสัมพันธ์ใด ที่ถูกเลือกเพื่อเปลี่ยนคุณสมบัติ ในการตรวจสอบนั้นเราใช้หลักการ Computer Graphic คือ เราจะทำการจำลอง Vector ขึ้นมาล้อมรอบ Panel แต่ละตัวไว้ดังรูปที่ ข-4 จากนั้นก็ตรวจสอบจุดที่ผู้ใช้คลิก ว่าถูกล้อมรอบด้วย Vector ของ Panel นั้น ๆ หรือไม่ (จุดที่กด อยู่ทางซ้ายของ Vector ทั้งหมด) ซึ่งสามารถตรวจสอบว่าจุดอยู่ทางซ้ายของ Vector หรือไม่โดยการใช้หลักการ Cross Product



รูปที่ ข-4 ตัวอย่างการสร้าง Vector ล้อมรอบเส้นแสดงความสัมพันธ์

2. หลักการ Cross Product

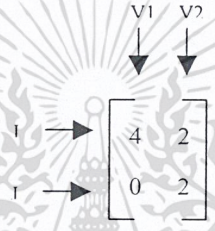


รูปที่ ข-5 ตัวอย่างจุดที่ใช้ในการคำนวณ Cross Product

กำหนดให้มี Vector และจุด ดังรูปที่ ข-5 เราสามารถหาได้ว่า จุด P3 นั้นอยู่ทางซ้ายของ Vector v1 หรือไม่ โดยการสร้าง Vector v2 ขึ้นมาโดยให้มีจุดหัวเป็นจุดเดียวกันกับจุดหัวของ Vector v1 (จุด P1) และจุดปลายเป็นจุดที่เราจะทำการตรวจสอบ (จุด P3) เพราะฉะนั้นเราจะมี 2 Vector คือ

- Vector v1 เกิดจากจุด P1 ไป P2 ซึ่งมีค่าเป็น 4I + 0J
- Vector v2 เกดจากจุด P1 ไป P3 ซึ่งมีค่าเป็น 2I + 2J

จากนั้นเราก็นำค่าที่ได้มาใส่ลงใน Matrix เพื่อทำการตรวจสอบดังเช่นรูปที่ ข-6 และทำการหาค่า Determinant ของ Matrix ซึ่งถ้าผลที่ได้มีค่าเป็นบวกแสดงว่าจุดอยู่ทางซ้ายของ Vector แต่ถ้าผลที่ได้มีค่าเป็นลบ แสดงว่าจุดอยู่ทางขวาของ Vector



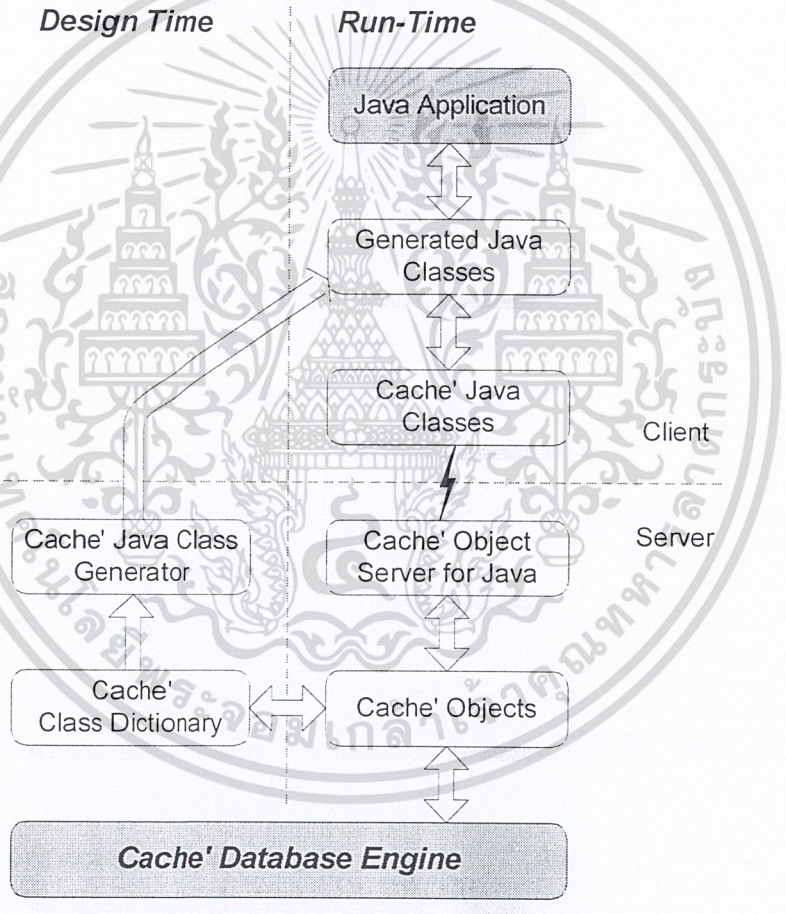
รูปที่ ข-6 ตัวอย่างการใส่ค่าลง Matrix จาก Vector ที่ได้

ภาคผนวก ค

การติดต่อระหว่าง Caché และ Java (Caché Java Binding)

การเขียนโปรแกรมติดต่อฐานข้อมูล Caché กับ Java Application จะต้องใช้ Caché Objects Server สำหรับ Java ในการทำงานร่วมกันกับ Caché ดังนี้

- สร้าง Java Class เพื่อใช้ในการเข้าถึงออบเจ็กต์ในฐานข้อมูล Caché
- เพิ่มประสิทธิภาพในการติดต่อระหว่างฐานข้อมูล Caché และ Java-based Client



รูปที่ ค-1 การติดต่อระหว่าง Caché Object Server สำหรับ Java

Caché Java Binding ประกอบไปด้วยคอมโพเน้นต์ต่างๆ ดังนี้

- Caché Java Class Generator เป็นคอมไพเลอร์ที่ทำหน้าที่คอมไพล์ Caché Class ให้เป็น Java Class จากคลาสที่ประกาศไว้ใน Caché Class Dictionary
- Caché Java Package เป็น Package ของ Java Class ซึ่งทำงานกับ Java Class ที่ถูกสร้างขึ้นจาก Caché Java Class Generator และ ช่วยให้ Java Class เหล่านั้นสามารถติดต่อกับออบเจ็กต์ที่เก็บอยู่ในฐานข้อมูล Caché ได้

- Caché Object Server สำหรับ Java เป็น Gateway ประสิทธิภาพสูงซึ่งคอยจัดการการติดต่อระหว่าง Java Client และ Caché Database Server โดยใช้มาตรฐาน Protocol TCP/IP ในการเชื่อมต่อเครือข่าย

1. การปรับแต่งการทำงานเพื่อใช้งาน Caché Object Server

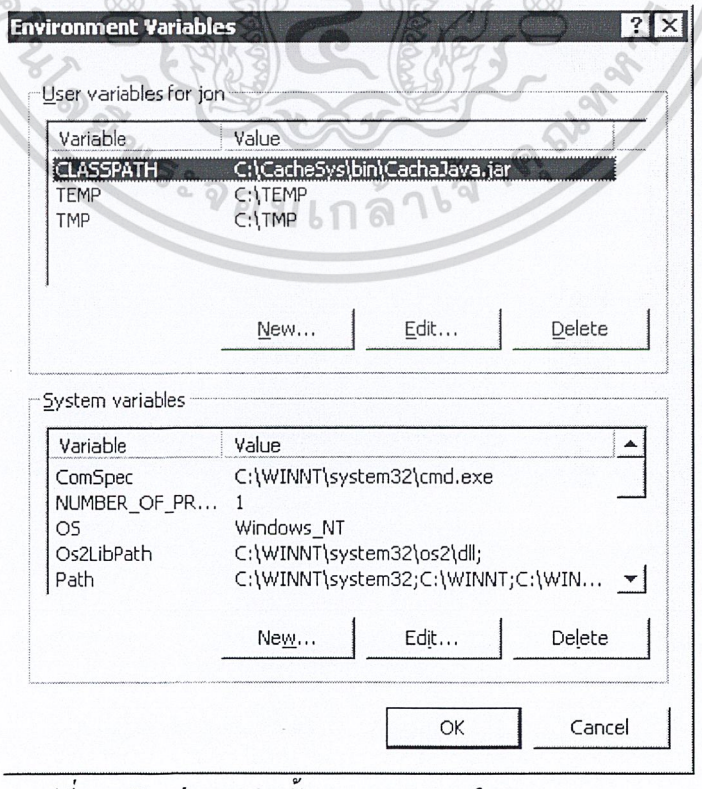
Caché Object Server สำหรับ Java ต้องการ JDK 1.1 หรือสูงกว่า เพื่อที่จะสามารถใช้งาน CacheJava.jar ซึ่งเป็น Package ที่ Caché มีมาให้ โดยในระบบปฏิบัติการ Windows ไฟล์นี้จะเก็บที่อยู่ Bin Subdirectory ของ Directory ที่ลง Caché เอาไว้ ที่ <cache_root>\bin\CacheJava.jar ถ้าในการติดตั้งบนระบบปฏิบัติการ Windows ได้ติดตั้งตาม Default Directory ไฟล์นี้ก็จะอยู่ที่ C:\CacheSys\bin\CacheJava.jar

เมื่อได้ตำแหน่งที่เก็บไฟล์แล้วก็ต้องเพิ่มไฟล์นี้เข้าไปที่ CLASSPATH เพื่อใช้อ้างคลาสในนั้นได้ ขณะ Java Runtime

2. การติดตั้ง CLASSPATH ใน Windows 2000

การติดตั้งตัวแปร CLASSPATH บนระบบปฏิบัติการ Windows 2000 ทำได้ดังนี้

1. เลือกไอคอน System ใน Windows Control Panel
2. เลือกแท็บ Advanced คลิกที่ปุ่ม Environment Variables Button
3. เพิ่ม Full Path ของไฟล์ CacheJava.jar เข้าไปในที่ท้ายตัวแปร CLASSPATH โดยใช้เครื่องหมาย Semicolon (;) ในการแบ่ง ถ้าของเดิมไม่มี CLASSPATH ให้ทำการสร้างขึ้นมาโดยคลิกที่ปุ่ม New



รูปที่ ค-2 ตัวอย่างการติดตั้ง CLASSPATH ใน Windows 2000

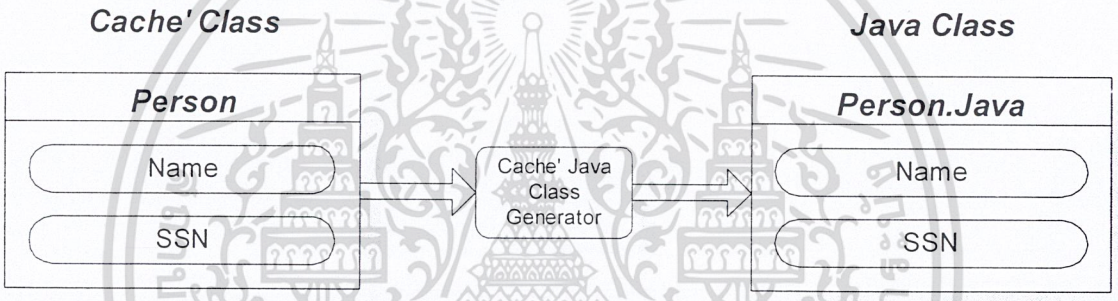
3. Caché Java Classes

ในการทำ Java Binding เพื่อเข้าถึง Caché Object นั้นก่อนที่จะทำการเข้าถึง instance ใดๆก็ตาม จะต้องทำการ Export คลาสใน Caché ไปเป็น Java Class เสียก่อน ซึ่งจะแตกต่างจากการทำ ActiveX Binding ที่ไม่ทำการ Export คลาสไปเป็น ActiveX Classes

ในชุดติดตั้ง Caché นั้นได้ให้ Utility Class โดยได้อธิบายไว้ในเรื่อง Caché Java Package ที่จะกล่าวถึงต่อไป

4. การสร้าง Java Class จาก Caché

Caché Java Class Generator จะสร้าง Java Class จากคลาสที่ประกาศเอาไว้ใน Caché Class Dictionary อย่างอัตโนมัติ โดยที่ Java Class ที่สร้างขึ้นจะมีชื่อเดียวกันกับ Caché Class ที่ประกาศไว้ดังรูป



รูปที่ ค-3 Java Class ที่ได้จาก Caché Class

Java Class ที่ได้นี้สามารถนำไปใช้งานกับ Java Application ได้เช่นเดียวกันกับ Java Class อื่นๆ ทั่วไปและในการสั่งให้ Caché สร้าง Java Class นั้นสามารถทำได้ 2 ทาง คือ

- โดยการใช้คำสั่ง Generate Java ใน Caché Object Architect
- โดยการใช้คำสั่ง ExportJavaList^%apiOBJ()

ทุก Caché Class จะถูก Export ออกเป็นหลายๆ ไฟล์ตามชื่อคลาสคลาสนั้น (classname.java) และเก็บไว้ใน Default Directory ไม่สามารถเปลี่ยนชื่อไฟล์เหล่านั้นได้ก่อนการคอมไพล์

5. โครงสร้างของ Caché Java Classes

Caché Java Generator จะทำการสร้าง Java Class จาก Caché Class ที่ประกาศเอาไว้ใน Caché Class Dictionary โดยใช้กฎดังต่อไปนี้ในการแปลง

- Property ต่างๆ ของคลาสใดจะถูกแปลงเป็นตัวแปรชนิด Protected member ของ Java Class นั้นๆ และจะสร้างเมธอดเพื่อใช้เข้าถึงข้อมูลให้แก่ Property แต่ละตัวด้วย
- Method ต่างๆ ของคลาสใดก็จะถูกแปลงให้เป็นเมธอดของ Java Class โดยการเรียกใช้เมธอดจาก Java นั้นผลลัพธ์จะได้มาจากการที่เมธอดถูก Execute อยู่ใน Caché Database Server ที่ทำการติดต่ออยู่

ถ้าชื่อของ Method หรือ Property ใดขึ้นต้นด้วยสัญลักษณ์ “%” มันจะถูกแปลงไปเป็นสัญลักษณ์ “sys_” แทน ตัวอย่างเช่น

```
Do.pat.%Save() ;   // Caché ObjectScript
```

จะกลายเป็น

```
pat.sys_Save(int depth) // Java
```

หรืออีกตัวอย่างหนึ่งเช่นสมมติว่าเราประกาศคลาส Person ไว้ใน Caché เป็นดังนี้

```
CLASS Person{
        SUPER = %Persistent;
        PERSISTENT;

        ATTRIBUTE Name{ TYPE = %String; }
        ATTRIBUTE SSN { TYPE = %String; }
        ATTRIBUTE DOB { TYPE = %Date; }
        ATTRIBUTE Spouse { TYPE = Person; }

        METHOD Triple(x:%Integer) {
                RETURNTYPE = %Integer;
                CODE = {
                        QUIT x*3
                }
    }
}
```

จากการประกาศข้างต้น Caché Java Class Generator จะ Generate ได้ Java Class ต่อไปนี้

```
public class Person extends Persistent {
// attribute
private DateAttr m_DOB;
private StringAttr m_Name;
private StringAttr m_SSN;
private TransientAttr m_Spouse;
```

```
// attribute accessor methods
public Date getDOB() throws CacheException {
        return m_DOB.get();
}
public void setDOB(Date value) throws CacheException {
        m_DOB.set(value);
}
public String getName() throws CacheException {
        return m_Name.get();
}
public void setName(String value) throws CacheException {
        m_Name.set(value);
}
public String getSSN() throws CacheException {
        return m_SSN.get();
}
public void setSSN(String value) throws CacheException {
        m_SSN.set(value);
}
public Person getSpouse() throws CacheException {
        return (Person) m_Spouse.get();
}
public void setSpouse(Person value) throws CacheException {
        m_Spouse.set ((Transient) value );
}
// methode

public int Triple(int x) throws CacheException {
        SysList _arg;
        _checkObject(true);
        _args = new SysList();
        _args.set(x,0);
        _saveDate();
        m_objectServer.invokeMethod(m_oRef,"Triple",_arg,true);
        return _args.getInt( 0 );
}
```

6. Caché Java Package

Caché Java Package คือ Java Package ที่ Caché เตรียมเอาไว้ให้ใช้สำหรับทำงานร่วมกับ Java Class ที่ถูกสร้างมาจาก Caché Class Generator เพื่อช่วยให้ Java Class เหล่านั้นสามารถติดต่อกับออบเจ็กต์ที่เก็บไว้ใน Caché Database ได้ Caché Java Package ประกอบไปด้วยคลาสต่างๆ ดังต่อไปนี้

- COM.intersys.object ใช้จัดการเกี่ยวกับ Instance ของ Caché Object
- COM.intersys.utils.CacheException ใช้แจ้งข้อผิดพลาดที่เกิดขึ้น
- COM.intersys.utils.SysList ใช้จัดการข้อมูลที่มีรูปแบบเป็น List

7. การใช้ Caché Objects จาก Java

7.1 Caché Object Server สำหรับ Java

Caché Object Server สำหรับ Java เป็น Gateway ประสิทธิภาพสูง ช่วยจัดการให้เกิด High Speed Communication ระหว่าง Client-based Caché Java Classes และ Caché Database

7.2 การติดต่อกับ Server

สิ่งแรกที่ต้องทำ คือในส่วนของ Client โปรแกรมต้องทำการสร้าง Caché Factory Object ขึ้นมาเสียก่อน และทำการติดต่อไปที่เครื่อง Server ที่กำลังรันนิ่ง Caché อยู่ ใน Java สามารถสร้าง Instance ของ Caché Factory Object และทำการติดต่อ Caché ได้โดยใช้โค้ดดังต่อไปนี้

```
String connect = "cn_iptcp:127.0.0.1[1972]:TCD";        // กำหนด Connection String
ObjectFactory factory = null;                  // ประกาศตัวแปรสำหรับ Object Factory ที่จะใช้กับ Application
factory = new ObjectFactory(connect); // ทำการสร้างและเปิด Connection
```

การกำหนดค่า connection string (connect) เพื่อใช้สำหรับการติดต่อจะประกอบด้วยส่วนของ Connection Protocol ต้องเป็น cn_iptcp เสมอ ตามด้วยส่วนของ IP Address ของเครื่อง Server และหมายเลข Port อยู่ภายในเครื่องหมาย "[-]" โดย Default จะเป็น Port 1972 และสุดท้ายคือส่วนของ Namespace ซึ่งใช้ระบุที่เก็บ Caché Object ที่ต้องการจะติดต่อ โดยแต่ละส่วนนั้นจะถูกแบ่งด้วยเครื่อง " : " ดังโค้ดข้างต้น

7.2 การยกเลิกการติดต่อกับ Server

สามารถทำได้โดยใช้เมธอด _Close() ของ Caché Factory Object เพื่อปิดออบเจ็กต์ที่สร้างขึ้นแล้วทำการฟรีหน่วยความจำคืนให้แก่ระบบ โดยใช้โค้ดดังต่อไปนี้

```
factory._Close();
factory = null;
```

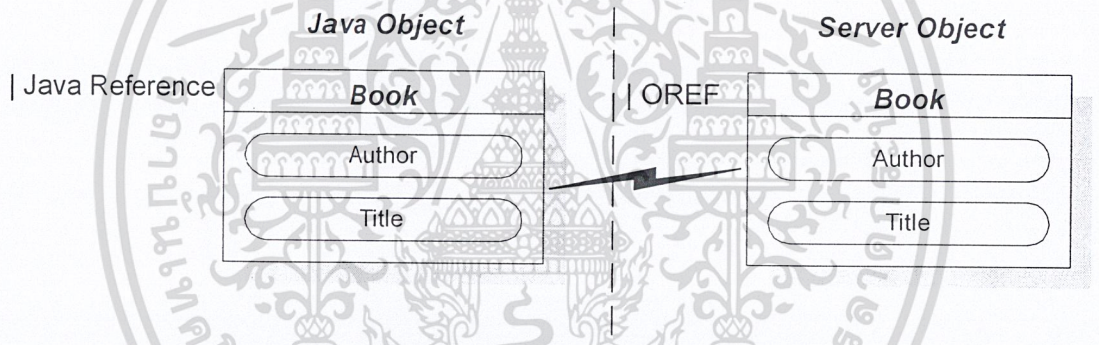
7.3 การสร้าง Object Instance ใน Java

ในการสร้าง Instance ของ Caché Object ใน Java เราสามารถทำได้โดยการใช้โอเปอเรเตอร์ New ผ่านค่าของ Caché Factory Object เข้าไป ตัวอย่างโค้ดการสร้าง Instance ใหม่ของออบเจ็กต์ Person ได้ดังนี้

```
person = new Person(factory);
```

person เป็น Instance ใหม่ของออบเจ็กต์ Person โดยแอตทริบิวต์ของออบเจ็กต์ Person จะมีค่าเป็นค่า Default

โดยทั่วไปเราสามารถเปิดออบเจ็กต์ที่มีอยู่แล้วจาก Caché Database ในกรณีนี้ทำได้โดยส่งค่า ID ซึ่งมีค่าเป็น String ของออบเจ็กต์ตัวที่ต้องการเปิดที่เก็บอยู่ในฐานข้อมูล เมื่อทำการสร้างออบเจ็กต์ Instance



รูป ค-4 ความสัมพันธ์ระหว่าง Java Object และ Server Object

โค้ดที่ใช้โหลด Instance ที่มีอยู่แล้วของออบเจ็กต์ Person จากฐานข้อมูล ได้ดังนี้

```
String id;
person = new Person(factory,id);
```

ในกรณีนี้ แอตทริบิวต์ของ person จะมีค่าที่เก็บเอาไว้ในฐานข้อมูลขณะนั้น และจำไว้ว่าทั้ง 2 กรณีนี้ออบเจ็กต์ที่เกิดขึ้นจะถูกสร้างอยู่บน Server ดังนั้นเมธอดของออบเจ็กต์ก็จะรันอยู่บน Server เช่นกัน

7.4 การใช้ Caché Object ใน Java Application

หลังจากที่ทำการสร้าง Instance ของ Caché Java Object แล้วเราก็จะสามารถใช้ออบเจ็กต์ดังกล่าวเสมือนออบเจ็กต์ใน Java ทั่วไป ยกตัวอย่างเช่น การกำหนดค่าและการอ่านค่าจากแอตทริบิวต์ของออบเจ็กต์

```
String name;
name = person.getName();
person.setName(name);
```

จะเห็นว่าไม่สามารถเข้าถึงแอตทริบิวต์ได้โดยตรง จะต้องใช้เมธอดที่ถูกสร้างขึ้นมาพร้อมกับ Java Class ซึ่ง Caché Class Generator สร้างให้เองซึ่งเรียกว่า Accessor Method เพื่อที่จะทำการเข้าถึงแอตทริบิวต์นั้น และสามารถเรียกใช้เมธอดอื่นๆ ของออบเจ็กต์ดังกล่าวได้ ซึ่งทุกเมธอดที่เรียกจะถูกปฏิบัติที่ทางฝั่ง Server เช่น

```
person.Admit()
```

เรายังสามารถอ้างถึงออบเจ็กต์อื่นๆ ได้ในลักษณะเดียวกันกับแอตทริบิวต์อื่นๆ เช่น

```
name =  person.getSpouse().getName();
```

7.5 การบันทึกค่าของออบเจ็กต์

เราสามารถบันทึกค่า Persistent Object เก็บลงไปที่ฐานข้อมูลได้โดยใช้ เมธอดชื่อ %Save() หรือถ้าใช้ใน Java จะเป็น _save()

```
person._save()          // บันทึกออบเจ็กต์ลงในฐานข้อมูล
person._close()         // ปิดออบเจ็กต์ทางฝั่ง Server ลง
person = null           // Set ค่าออบเจ็กต์ให้เป็น Null
```

สังเกตว่าเราต้องเรียกเมธอด _close() เมื่อเราใช้ออบเจ็กต์นั้นเสร็จแล้ว ซึ่งทำให้ออบเจ็กต์ทางฝั่ง Server ถูกปิดลง จึงต้องทำการ Set ค่าออบเจ็กต์ให้ชี้ไปที่ Null pointer เพื่อเป็นการคืนหน่วยความจำให้กลับระบบ

ข้อแตกต่างกันระหว่างการเรียกใช้เมธอดใน Java และใน Caché Object Script คือ Caché Object Script นั้นเมธอดจุขึ้นต้นด้วย "%" ส่วนใน Java จะขึ้นต้นด้วย "_" ยกตัวอย่างเช่นใน Caché Object Script เรียกใช้ person.%Save() แต่ใน Java จะเรียกโดยใช้ person._save()

7.6 การเรียก Query ใน Java

ใน Java นั้นการ Query ที่สร้างขึ้นจากคลาสในฐานข้อมูลสามารถเรียกใช้ได้โดยใช้ออบเจ็กต์ของคลาส ResultSet ซึ่งออบเจ็กต์ของคลาส ResultSet จะต้องอยู่ใน Java เท่านั้น ไม่มีความเกี่ยวข้องกับออบเจ็กต์ใน Caché ทุกออบเจ็กต์ของคลาส ResultSet จะต้องกำหนดชื่อของคลาสที่เก็บ Query และชื่อของ Query ที่ได้สร้างเอาไว้แล้ว ดังกล่าว ตัวอย่างการรัน Query ที่อยู่ในคลาส Person ที่ชื่อ ByName

```
String id;
String name;
ResultSer rset;
        // กำหนด Factory, Class และ Query สำหรับ ResultSet
rset = new ResultSet(factory,"Person","ByName");
        // Execute Query
rset.execute();
while(rset.next()) {
        // อ่านค่า Id ที่อยู่ที่คอลัมน์แรก
        id = rset.getString();
        // อ่านค่า Name ที่อยู่ที่คอลัมน์ที่สอง
        name = rset.getString(1);
        // พิมพ์ค่า Id และ Name ที่ได้ของแต่ละ Person
        System.out.println("ID:" + id + ", Name : " + name);
}
        // ปิดการ query
rset.Close();
```

จากตัวอย่างการใช้งานข้างบน สามารถเรียกใช้งาน Query ชื่อ ByName ที่ถูกกำหนดในคลาส Person ซึ่ง Query นี้จะส่งค่าของออบเจ็กต์ Person ทั้งหมด หลังจากทำการ Execute แล้วก็สามารถเรียกดูออบเจ็กต์ทีละตัวได้ โดยเรียกใช้เมธอด rset.Next() จนครบหมดทุกตัวแล้วจึงทำการปิดออบเจ็กต์ ResultSet ด้วยเมธอด rset.Close()

บรรณานุกรม

[1] Marianthi Svinterikou and Babis Theodoulids (1997) : "*The Temporal Unified Modeling Language (TUML)*", TimeLab Technical Report, October 1997, pp1-51

[2] Marianthi Svinterikou and Babis Theodoulids (1997) : "*Mapping TUML to TAU TODL*", Time Lab Technical Report, November 1997, pp 1-16

[3] Ioannis Kakoudakis (1996) : "*The TAU Temporal Object Model*", A thesis submitted to the University of Manchester, 1996, Chapter 5 pp 1-21

[4] Date C.J. (2000) : "*An introduction to Database System Vol. 1, 7 Edition*", Addison Wesley, Massachusetts, 2000

[5] Abraham Silberschatz ,Henry F. Korth and S. Sudarshan (1996) : "*Database System Concepts, 3 Edition*", McGraw-HILL, 1996, Chapter 8 pp 251-274

[6] Abdullah Uz Tansel (1993) : "Temporal databases : theory, design, and implementation", The Benjamin/Cummings Publishing Company, California, 1993

[7] Bundit Pasaya (2001) : "*The TOONIAM CONEPTUAL SCHEMA MODEL AND A TRANSFROMATION TO AN OBJECT ORIENTED DATABASE SCHEMA*", A thesis submitted to KINGMONGHUT'S INSTITUTE OF TECHNOLOGY LADKRABANG, 2001, pp 37-104

[8] นายธีรวุธ สันติสุธีรวุฒิ และนายประยงค์ ทานนท์ (2543) : "*การจัดการระบบสารสนเทศเชิงเวลาบนฐานข้อมูลเชิงวัตถุสัมพันธ์*", ปริญญานิพนธ์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง, ปีการศึกษา 2543, หน้า 4-8

[9] วีระศักดิ์ ซึงถาวร,ดร. (2541) : "*Fundamental of JAVA Programming Volume 1*", Sum Publishing, กรุงเทพฯ, 2541

[10] วีระศักดิ์ ซึงถาวร,ดร. (2543) : "*Fundamental of JAVA Programming Volume 2*", ซีเอ็ดยูเคชั่น, กรุงเทพฯ, 2543

[11] กิตติ ภักดีวัฒนะกุล (2544) : "JAVA ฉบับโปรแกรมเมอร์", KTP Comp & Consult, กรุงเทพฯ, 2544