การออกแบบกฎการแปลงยูเอ็มแอลคลาสไดอะแกรมเป็นสคีมาฐานข้อมูลเชิงวัตถุ

นายสถิตย์ ประสมพันธ์

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมซอฟต์แวร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2547

ISBN 974-17-6994-6

DESIGN OF RULES FOR TRANSFORMING A UML CLASS DIAGRAM TO OBJECT-ORIENTED DATABASE SCHEMA

Mr.Sathit Prasomphan

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Science in Software Engineering
Department of Computer Engineering
Faculty of Engineering
Chulalongkorn University
Academic Year 2004

ISBN 974-17-6994-6

หัวข้อวิทยานิพนธ์	การออกแบบกฎการแปลงยูเอ็มแอลคลาสไดอะแกรมเป็นสคีมาฐานข้อมูลเชิงวัตถุ
โดย	นายสถิตย์ ประสมพันธ์
สาขาวิชา	วิศวกรรมซอฟต์แวร์
อาจารย์ที่ปรึกษา	รองศาสตราจารย์ ดร. วันชัย ริ้วไพบูลย์

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

..... คณบดีคณะวิศวกรรมศาสตร์
(ศาสตราจารย์ ดร. ดิเรก ลาวัณย์ศิริ)

คณะกรรมการสอบวิทยานิพนธ์

..... ประธานกรรมการ
(ผู้ช่วยศาสตราจารย์ ดร.พรศิริ หมื่นไชยศรี)

..... อาจารย์ที่ปรึกษา
(รองศาสตราจารย์ ดร. วันชัย ริ้วไพบูลย์)

..... กรรมการ
(อาจารย์ นครทิพย์ พร้อมพูล)

..... กรรมการ
(อาจารย์ เชษฐ พัฒโนทัย)

สถิตย์ ประสมพันธ์ : การออกแบบกฎการแปลงยูเอ็มแอลคลาสไดอะแกรมเป็นสคีมาฐานข้อมูลเชิงวัตถุ.(DESIGN OF RULES FOR TRANSFORMING A UML CLASS DIAGRAM TO OBJECT-ORIENTED DATABASE SCHEMA) อ.ที่ปรึกษา: รองศาสตราจารย์ ดร. วันชัย ริ้วไพบูลย์, 166 หน้า. ISBN 974-17-6994-6.

กระบวนการทางด้านวิศวกรรมซอฟต์แวร์ในปัจจุบันพบว่าได้นำเอาระเบียบวิธีและเครื่องมือต่าง ๆ เข้ามาใช้เพื่ออำนวยความสะดวกในระหว่างการพัฒนาซอฟต์แวร์เป็นจำนวนมากแต่เครื่องมือที่นำมาช่วยในการอำนวยความสะดวกสำหรับการพัฒนาในระบบฐานข้อมูลเชิงวัตถุมีจำนวนน้อยและยังไม่มีการพัฒนาทฤษฎีต่าง ๆ ที่ช่วยในการออกแบบสำหรับฐานข้อมูลเชิงวัตถุ ทั้งที่ในความเป็นจริงระบบฐานข้อมูลเชิงวัตถุสามารถออกแบบได้โดยการใช้คลาสไดอะแกรม ซึ่งให้ผลลัพธ์การทำงานที่มีประสิทธิภาพมากกว่า โดยสามารถระบุถึงคุณลักษณะประจำต่างๆ ของวัตถุเมทธอด รวมถึงความสัมพันธ์ระหว่างวัตถุได้อย่างมีประสิทธิภาพ

ในงานวิจัยฉบับนี้ได้ศึกษาถึงความเกี่ยวข้องระหว่างการออกแบบฐานข้อมูลโดยใช้คลาสไดอะแกรมและการแปลงข้อมูลจากคลาสไดอะแกรมดังกล่าวให้อยู่ในรูปของสคีมาฐานข้อมูลเชิงวัตถุ โดยสร้างกฎการแปลงคลาสไดอะแกรมเป็นสคีมาฐานข้อมูลเชิงวัตถุ 11 กฎ สำหรับฐานข้อมูลเชิงวัตถุตามมาตรฐานโอดีเอ็มจี ฐานข้อมูลเชิงวัตถุคาเช่ และฐานข้อมูลเชิงวัตถุแมทิส และสร้างเครื่องมือที่ประยุกต์กฎการแปลงคลาสไดอะแกรมเป็นสคีมาฐานข้อมูลเชิงวัตถุ

ผลการทดสอบเครื่องมือที่ประยุกต์ใช้กฎกับตัวอย่างของคลาสไดอะแกรมด้วยการเปรียบเทียบสคีมาฐานข้อมูลเชิงวัตถุที่ได้จากเครื่องมือกับ สคีมาฐานข้อมูลเชิงวัตถุที่ได้จากการทำการแปลงด้วยตนเองโดยการประยุกต์ใช้กฎ พบว่าเครื่องมือสามารถให้ผลการแปลงสคีมาฐานข้อมูลเชิงวัตถุได้ถูกต้อง ผลของการวิจัยสามารถนำไปใช้งานในเชิงธุรกิจและส่งเสริมให้มีการพัฒนาและมีการใช้ฐานข้อมูลเชิงวัตถุมากขึ้น

ภาควิชา วิศวกรรมคอมพิวเตอร์ ลายมือชื่อนิสิต.....

สาขาวิชา วิศวกรรมซอฟต์แวร์ ลายมือชื่ออาจารย์ที่ปรึกษา.....

ปีการศึกษา 2547

4570691721 : MAJOR SOFTWARE ENGINEERING
KEY WORD: TRANSFORMATION / CLASS DIAGRAM / UML /
OBJECT-ORIENTED DATABASE SCHEMA

SATHIT PRASOMPHAN:DESIGN OF RULES FOR TRANSFORMING A UML CLASS DIAGRAM TO OBJECT-ORIENTED DATABASE SCHEMA.THESIS ADVISOR : ASSOC.PROF.WANCHAI RIVEPIBOON,Ph.D, 166 pp. ISBN 974-17-6994-6.

In present, software engineering process uses a lot of methodologies and tools to develop software. While researchers have developed many database design methodologies for database system, there is no widely accepted methodologies for object-oriented database schema design yet. In fact object-oriented database can be designed by using class diagram which can give more efficiency of output and easy to specify attribute, method and relationship between objects in object-oriented database.

In this research, relationship between object-oriented database design by using class diagram and object-oriented database schema transformation are studied. Transformation rules with 11 rules for ODMG object-oriented database, Cache object-oriented database and Matisse object-oriented database are made. An automated tool was built to transform class diagram into object-oriented database schema.

The tool is tested with class diagrams by comparing its transformation results to the manual transformation results. The comparison result shows that the tool can give the correct results. From this study, we expect that object-oriented database will be used in business and will encourage further studies and development of object-oriented database.

Department Computer Engineering Student's signature.....
Field of study Software Engineering Advisor's signature.....
Academic year 2004

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงไปได้ด้วยความช่วยเหลือจากรองศาสตราจารย์ ดร.วันชัย ริ้วไพบูลย์ อาจารย์ที่ปรึกษาวิทยานิพนธ์ของข้าพเจ้า ขอกราบขอบพระคุณที่ได้ให้คำแนะนำและข้อเสนอแนะต่าง ๆ ตลอดระยะเวลาในการทำวิทยานิพนธ์

ขอกราบขอบพระคุณผู้ช่วยศาสตราจารย์ ดร.พรศิริ หมื่นไชยศรี ซึ่งเป็นประธานกรรมการสอบวิทยานิพนธ์ อาจารย์ นครทิพย์ พร้อมพูล และ อาจารย์ เชษฐ พัฒโนทัย ซึ่งเป็นกรรมการสอบวิทยานิพนธ์ ที่ได้สละเวลาและให้คำแนะนำต่างๆ ที่เป็นประโยชน์อย่างยิ่งต่อการจัดทำวิทยานิพนธ์ฉบับนี้

ขอกราบขอบพระคุณบิดา มารดา พี่ชาย และพี่สาว ที่ให้กำลังใจและแนะนำข้อคิดเห็นที่เป็นประโยชน์

ขอขอบคุณเพื่อนๆ และน้องๆ จากภาควิชาวิทยาการคอมพิวเตอร์ มหาวิทยาลัยขอนแก่น และภาควิชาวิศวกรรมคอมพิวเตอร์ จุฬาลงกรณ์มหาวิทยาลัยทุกคนที่เป็นกำลังใจตลอดการจัดทำวิทยานิพนธ์

ท้ายที่สุดขอขอบคุณสถาบันเทคโนโลยีพระจอมเกล้าพระนครเหนือที่มอบทุนโครงการพัฒนาอาจารย์สำหรับการเรียนระดับปริญญาโทในครั้งนี้

สารบัญ

หน้า

หน้า

หน้า

สารบัญตาราง

ตาราง หน้า

หน้า

หน้า

สารบัญภาพ

ภาพประกอบ หน้า

หน้า

หน้า

สารบัญภาพ (ต่อ)

หน้า

หน้า

หน้า

หน้า

บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

กระบวนการทางด้านวิศวกรรมซอฟต์แวร์ (Software engineering) ในปัจจุบันพบว่าได้นำเอาระเบียบวิธี (Methodology) และเครื่องมือต่าง ๆ เข้ามาใช้เพื่ออำนวยความสะดวกในระหว่างการพัฒนาซอฟต์แวร์เป็นจำนวนมาก โดยเฉพาะอย่างยิ่งเครื่องมือที่นำมาช่วยในกระบวนการพัฒนาในด้านฐานข้อมูลจะมีเครื่องมือสำหรับใช้อำนวยความสะดวกในการพัฒนาในด้านนี้เป็นจำนวนมาก

แต่เป็นที่น่าสังเกตว่าเครื่องมือที่นำมาช่วยในการอำนวยความสะดวกสำหรับการพัฒนาในระบบฐานข้อมูลเชิงวัตถุ (Object-oriented database) มีจำนวนน้อยมาก สาเหตุประการหนึ่งก็เนื่องมาจาก ระบบฐานข้อมูลเชิงวัตถุยังไม่แพร่หลาย และยังไม่มีการใช้งานมากนัก เนื่องจากการออกแบบฐานข้อมูลเชิงวัตถุ (Object-oriented database design) มีความซับซ้อนและยุ่งยากมากกว่าการออกแบบฐานข้อมูลเชิงสัมพันธ์ (Relational database design) [7] และในขณะเดียวกันเมื่อนักวิจัยพยายามที่จะพัฒนาเครื่องมือต่าง ๆ เพื่อใช้ในกระบวนการพัฒนาระบบในฐานข้อมูลจะพบว่าผลงานวิจัยส่วนใหญ่ได้เน้นไปเฉพาะที่การพัฒนาเครื่องมือสำหรับระบบฐานข้อมูลเชิงสัมพันธ์มากกว่าโดยที่เครื่องมือหรือระเบียบวิธีสำหรับระบบฐานข้อมูลเชิงวัตถุได้ถูกพัฒนาไปน้อยมากเมื่อเปรียบเทียบกับผลงานวิจัยที่มี [7] และยังไม่มีการพัฒนากฎต่าง ๆ ที่ช่วยในการออกแบบสำหรับฐานข้อมูลเชิงวัตถุ ทั้ง ๆ ที่ ในความเป็นจริง ระบบฐานข้อมูลเชิงวัตถุสามารถออกแบบได้โดยการใช้คลาสไดอะแกรม(Class diagram) ของยูเอ็มแอล(UML-Unified Modeling Language) ซึ่งให้ผลลัพธ์การทำงานที่มีประสิทธิภาพมากกว่าเนื่องจาก การออกแบบโดยการใช้คลาสไดอะแกรมสามารถที่จะระบุถึงคุณแอทริบิวต์ต่าง ๆ ของวัตถุ รวมถึงความสัมพันธ์ระหว่างวัตถุได้ดีกว่าการออกแบบโดยการใช้เครื่องมืออื่น

เหตุผลอีกประการคือแอพลิเคชันที่พัฒนาด้วยเทคโนโลยีเชิงวัตถุ (Object-oriented technology) จะเป็นระบบจัดเก็บข้อมูลที่สามารถใช้ได้ทั้งระบบจัดการฐานข้อมูลเชิงสัมพันธ์ (RDBMS-Relational Database Management System) และระบบจัดการฐานข้อมูลเชิงวัตถุ (OODBMS-Object Oriented Database Management System) ทั้งสองวิธีมีการจัดเก็บข้อมูลให้ผลที่แตกต่างกันอย่างเห็นได้ชัด ทั้งในแง่ของประสิทธิภาพในการพัฒนา และประสิทธิภาพใน

การทำงานของแอพลิเคชัน[4] อย่างไรก็ตามถ้าระบบจัดการฐานข้อมูลสนับสนุนเทคโนโลยีเชิงวัตถุ การเข้าถึงข้อมูลในฐานข้อมูลเชิงวัตถุก็สามารถทำได้โดยตรง ไม่ต้องมีส่วนของแอพลิชันที่ทำหน้าที่ในการแปลง ความซับซ้อนในการพัฒนาแอพลิเคชันก็จะน้อยลง [4]

จากเหตุผลดังกล่าวผู้วิจัยจึงนำเสนอความเกี่ยวข้องกันระหว่างขั้นตอนการออกแบบฐานข้อมูลโดยใช้คลาสไดอะแกรม และการแปลงข้อมูลจากคลาสไดอะแกรมดังกล่าวให้อยู่ในรูปของ ข้อมูลของฐานข้อมูลเชิงวัตถุ หรือสคีมาของฐานข้อมูลเชิงวัตถุ (Object-oriented database schema) โดยอาศัยภาษานิยามเชิงวัตถุ (ODL-Object Definition Language) ของโอดีเอ็มจี (ODMG-Object Data Management Group) เพื่อช่วยให้กระบวนการในการพัฒนาซอฟต์แวร์ในขั้นตอนของการเก็บข้อมูลเชิงวัตถุ ขั้นตอนของการออกแบบ และขั้นตอนของการพัฒนาระบบ มีความสอดคล้องกันมากขึ้น ซึ่งจะเป็นผลให้การพัฒนาซอฟต์แวร์ สามารถที่จะทำได้ง่ายและมีประสิทธิภาพมากขึ้น โดยที่สามารถนำสคีมาที่ได้ดังกล่าวไปใช้ในการสร้างฐานข้อมูลเชิงวัตถุเพื่อพัฒนาแอพลิเคชันในระบบธุรกิจต่อไป

1.2 **วัตถุประสงค์ของการวิจัย**

เพื่อออกแบบกฎและสร้างเครื่องมือสำหรับการแปลงคลาสไดอะแกรมเป็นสคีมาของฐานข้อมูลเชิงวัตถุบนเว็บแอพลิเคชัน

1.3 **ขอบเขตของการวิจัย**

1. กฎเกณฑ์ที่ใช้ในการสร้างสคีมาสำหรับฐานข้อมูลเชิงวัตถุอาศัยมาตรฐานของโอดีเอ็ม-จีโดยการใช้ภาษานิยามฐานข้อมูลเชิงวัตถุเป็นหลัก

2. กฎเกณฑ์ที่ใช้ในการสร้างสคีมาสำหรับฐานข้อมูลเชิงวัตถุจากคลาสไดอะแกรมจะสามารถใช้ได้กับฐานข้อมูลเชิงวัตถุต่อไปนี้ได้เท่านั้น

- ระบบฐานข้อมูลเชิงวัตถุคาเช่ (Cache)
- ระบบฐานข้อมูลเชิงวัตถุแมทิส(Matisse)

3. ข้อมูลนำเข้าในโปรแกรมจะได้จากแผนภาพคลาสไดอะแกรมเท่านั้นและแผนภาพคลาสไดอะแกรมดังกล่าวจะถูกเก็บข้อมูลที่จะใช้ในการแปลงเป็นสคีมาฐานข้อมูลเชิงวัตถุ โดยการใช้เอ็กซ์เอ็มแอล (XML-Extensible Markup Language) เพื่อทำการเก็บข้อมูลที่จำเป็นในการสร้างเป็นสคีมาของฐานข้อมูลเชิงวัตถุต่อไป โดยเหตุผลของการสร้างโปรแกรมเพื่อเก็บข้อมูลจากคลาสไดอะแกรมเป็นเอ็กซ์เอ็มแอลใหม่เนื่องจากในงานวิจัยก่อน ๆ ที่มีการเก็บข้อมูลเป็นเอ็กซ์เอ็มแอลจะมีข้อมูลที่ต้องการไม่เพียงพอในการสร้างเป็นฐานข้อมูลเชิงวัตถุดังนั้นจึงต้องมีการสร้างกฎ

ในการดึงข้อมูลจากคลาสไดอะแกรมมาเป็นเอ็กซ์เอ็มแอล และนำเอาข้อมูลในเอ็กซ์เอ็มแอลดังกล่าวไปใช้ในการสร้างสคีมาของฐานข้อมูลเชิงวัตถุที่ต้องการ

4. ความถูกต้องของสคีมาของฐานข้อมูลเชิงวัตถุที่สร้างขึ้น ขึ้นอยู่กับความถูกต้องของข้อมูลที่นำเข้าที่ผู้ใช้นำเข้ามาในโปรแกรมในส่วนของคลาสไดอะแกรมโดยเฉพาะ

5. เอกสารข้อกำหนดความต้องการของระบบจะถูกนำมาใช้เพื่อช่วยพิจารณาความถูกต้องของผลลัพธ์ที่ได้เท่านั้นและเป็นข้อมูลเบื้องต้นของการสร้างคลาสไดอะแกรม

6. โปรแกรมจะสร้างสคีมาในรูปของข้อความเท่านั้นในส่วนของการนำเข้าเป็นฐานข้อมูลเชิงวัตถุ ผู้ใช้จะต้องนำผลลัพธ์ที่ได้จากโปรแกรมไปสร้างในระบบฐานข้อมูลเชิงวัตถุแต่ละประเภทเอง โดยผลลัพธ์จากงานวิจัยที่ได้คือสคีมาของฐานข้อมูลเชิงวัตถุ โดยสคีมาเป็นข้อมูลของฐานข้อมูล ส่วนหนึ่งของระบบฐานข้อมูล ที่จะใช้อธิบายข้อมูลซึ่งจะมีรายละเอียดที่แตกต่างกันตามแบบจำลองของข้อมูลที่ใช้

7. กฎเกณฑ์ของการสร้างคลาสไดอะแกรมอาศัยมาตรฐานของยูเอ็มแอลรุ่น 1.1

8. เครื่องมือจะทำงานบนอินเตอร์เน็ตเอ็กโพเลอร์เวอร์ชั่น 6.0 ขึ้นไป

9. ในงานวิจัยนี้จะทดสอบความสามารถในการนำสคีมาของฐานข้อมูลเชิงวัตถุที่ได้จากตัวแอพลิเคชัน ไปทดลองสร้างฐานข้อมูลเชิงวัตถุกับระบบฐานข้อมูลเชิงวัตถุดังนี้

- ระบบฐานข้อมูลเชิงวัตถุคาเช่
- ระบบฐานข้อมูลเชิงวัตถุแมทิส

โดยการทดสอบจะนำสคีมาฐานข้อมูลเชิงวัตถุที่ได้ เปรียบเทียบกับคลาสไดอะแกรม ที่สร้างขึ้นว่ามีข้อมูลของแอทริบิวต์ต่างๆ และชื่อคลาสที่ได้รวมถึงความสัมพันธ์ต่าง ๆ ตรงกับข้อมูลของฐานข้อมูลที่สร้างขึ้นหรือไม่และนำเข้าสู่ระบบฐานข้อมูลแต่ละชนิด

10. เนื่องจากผลลัพธ์ของสคีมาของฐานข้อมูลเชิงวัตถุที่ได้จากการแปลงคลาสไดอะแกรม อยู่ในรูปของมาตรฐานของโอดีเอ็มจี ซึ่งเมื่อนำสคีมาดังกล่าวไปใช้งานจริงอาจจะเกิดข้อผิดพลาดเนื่องจากระบบฐานข้อมูลในเชิงการค้าแต่ละประเภท จะมีรูปแบบการเขียนที่แตกต่างกันบ้างบางส่วน ดังนั้นในงานวิจัยนี้จึงได้พัฒนาเครื่องมือที่สามารถแปลงสคีมาของฐานข้อมูลให้อยู่ในรูปของภาษามาตรฐาน และสคีมาที่สามารถใช้กับระบบฐานข้อมูลทางการค้าได้ โดยในงานวิจัยนี้จะได้ทดลองใช้กับระบบฐานข้อมูลเชิงการค้า 2 ระบบดังกล่าวข้างต้น เพื่อทำการเปรียบเทียบว่าผลลัพธ์ที่ได้จากสคีมาของฐานข้อมูลตามมาตรฐานของโอดีเอ็มจี และสคีมาที่อยู่ในรูปของสคีมาระบบฐานข้อมูลทางการค้าว่ามีความแตกต่างกันมากน้อยเพียงใด

11. วิธีการทดสอบความถูกต้องของกฎและเครื่องมือที่ได้สามารถตรวจสอบได้ โดยการดูผลลัพธ์ที่ได้จากเครื่องมือและพิจารณากฎที่ใช้ในการแปลงเป็นหลักโดยดูจากชุดคำสั่งของการสร้างสคีมาที่ถูกสร้างออกมาว่าแต่ละส่วนของชุดคำสั่งของการสร้างสคีมาครบถ้วนหรือไม่ ถูกต้อง

ตามกฎที่มีหรือไม่ และเมื่อนำชุดคำสั่งของการสร้างสคีมาที่ได้ไปสร้างในระบบฐานข้อมูลเชิงวัตถุแล้วตรวจสอบดูว่า ชื่อคลาส แอทริบิวต์ เมทธอด ความสัมพันธ์ การสืบทอดต่างๆ มีครบตามที่เขียนไว้ในคลาสไดอะแกรมหรือไม่ ส่วนการตรวจสอบความถูกต้องของกฎที่ใช้จะตรวจสอบกับมาตรฐานสำหรับฐานข้อมูลเชิงวัตถุของโอดีเอ็มจีว่า กฎการแปลงที่ได้ครบถ้วนตามมาตรฐานที่โอดีเอ็มจีกำหนดไว้หรือไม่

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1. ได้เครื่องมือและกฎเกณฑ์ที่ช่วยในการออกแบบและสร้างสคีมาสำหรับการพัฒนาระบบฐานข้อมูลเชิงวัตถุ
2. การใช้งานสามารถนำไปใช้งานในเชิงธุรกิจได้หรือเพื่อการศึกษาในการสร้างฐานข้อมูลเชิงวัตถุได้
3. ส่งเสริมและกระตุ้นให้มีการพัฒนา และมีการใช้ฐานข้อมูลเชิงวัตถุมากขึ้น
4. ส่งเสริมให้การพัฒนาแอพลิเคชันโดยเทคโนโลยีเชิงวัตถุเป็นไปได้ง่ายและสะดวกมากยิ่งขึ้น
5. ลดความยุ่งยากในการแปลงข้อกำหนดความต้องการซอฟต์แวร์เป็นฐานข้อมูลในระบบที่ต้องการ

1.5 วิธีการดำเนินการวิจัย

1. ศึกษาถึงเครื่องมืออื่น ๆ ที่เกี่ยวข้องกับการแปลงแผนภาพแบบต่าง ๆ เป็นสคีมาของฐานข้อมูลเชิงวัตถุที่สามารถจัดเก็บในฐานข้อมูลเชิงวัตถุได้
2. ศึกษาถึงระบบฐานข้อมูลเชิงวัตถุประเภทต่าง ๆ ที่มีในปัจจุบัน วิเคราะห์รูปแบบโครงสร้าง และการทำงาน
3. ศึกษาถึงการออกแบบ กฎเกณฑ์ และรายละเอียดต่าง ๆ ของแผนภาพยูเอ็มแอลในส่วนของคลาสไดอะแกรม
4. ศึกษาถึงการออกแบบ กฎเกณฑ์ และรายละเอียดต่าง ๆ ของมาตรฐานระบบฐานข้อมูลเชิงวัตถุ
5. วิเคราะห์ และออกแบบเครื่องมือที่จะพัฒนา
6. พัฒนาเครื่องมือที่ได้ทำการวิเคราะห์และออกแบบ
7. ทดสอบและปรับปรุงเครื่องมือที่ได้พัฒนาแล้ว
8. สรุปผลการวิจัยและข้อเสนอแนะ
9. จัดทำรายงานวิทยานิพนธ์

บทที่ 2

ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 ทฤษฎีที่เกี่ยวข้อง

การสร้างกฎในการแปลงจากรูปแบบหนึ่งเป็นอีกรูปแบบหนึ่ง จำเป็นอย่างยิ่งที่ต้องอาศัยทฤษฎีหรือกฎต่าง ๆ ในการรองรับการสร้างกฎใหม่ ๆ เหล่านั้นดังนั้นในงานวิจัยฉบับนี้ได้อาศัยทฤษฎีที่เกี่ยวข้องเพื่อใช้ในกระบวนการวิเคราะห์และสร้างกฎดังนี้

2.1.1 แนวคิดแบบจำลองเชิงวัตถุ(Object-oriented model concept) [2] [10]

วัตถุ(Object) ถูกนิยามด้วยแอทริบิวต์(Attribute) และเมทธอด (Method) ซึ่งแอทริบิวต์แสดงสถานะของวัตถุ ส่วนเมทธอดเป็นการกระทำที่แสดงพฤติกรรมของวัตถุในสถานะต่าง ๆ โดยที่เมื่อวัตถุแต่ละวัตถุเมื่อถูกสร้างขึ้นจะมีค่าตัวระบุวัตถุ (OID-Object Identifier) ซึ่งเป็นค่าระบุประจำที่ทำให้วัตถุนั้นไม่ซ้ำกัน(Unique) ในระบบ วัตถุในระบบจะมีพฤติกรรมและแอทริบิวต์ของแต่ละวัตถุต่างกัน แต่หากวัตถุใดๆ มีพฤติกรรมและแอทริบิวต์เหมือนกัน เราเรียกว่าเป็นวัตถุชนิดเดียวกัน ซึ่งโครงสร้างของวัตถุก็จะเหมือนกันทั้งแอทริบิวต์และเมทธอด จึงมีการนิยามโครงสร้างของวัตถุชนิดเดียวกันในรูปของคลาส (Class) เปรียบเสมือนแผ่นแบบ (Template) หรือแม่แบบของวัตถุ และเรียกวัตถุของคลาสว่าเป็นอินสแตนซ์ (Instance) ของคลาสนั้น และคลาสสามารถสืบทอดลักษณะ(Inherit) ของตัวเองไปยังคลาสอื่นๆได้เพื่อทำให้เกิดคลาสใหม่ที่มีลักษณะเพิ่มเติมจากคลาสเดิมเรียกคลาสที่ถูกถ่ายทอดลักษณะไปว่าซูเปอร์คลาส (Super class) และเรียกคลาสที่รับการถ่ายทอดลักษณะมาว่า สับคลาส (Subclass) การสืบทอดลักษณะทั่วไปทำให้เกิดลักษณะที่เจาะจงและมีความเป็นทั่วไปน้อยลงกว่าเดิม

2.1.2 แนวคิดระบบการจัดการฐานข้อมูลเชิงวัตถุ (Object-oriented database management system concept) [2]

ระบบการจัดการฐานข้อมูลเชิงวัตถุเป็นระบบการจัดการฐานข้อมูลที่มีการนำเอาแนวคิดแบบจำลองเชิงวัตถุมาช่วยในการจัดเก็บข้อมูล โดยระบบฐานข้อมูลดังกล่าวจะนำเอาคุณสมบัติในเรื่องของความจำเพาะเจาะจงของวัตถุที่มีอยู่ คุณสมบัติของการสืบทอดลักษณะของวัตถุ เข้าช่วย เพราะเหตุที่ระบบการจัดการฐานข้อมูลเชิงวัตถุใช้แบบจำลองเชิงวัตถุในการวางโครงสร้าง

ข้อมูลดังนั้นลักษณะข้อมูลในระบบการจัดการฐานข้อมูลเชิงวัตถุนั้น จะประกอบไปด้วยวัตถุต่าง ๆ ตามโครงสร้างที่วางไว้ การวางโครงสร้างของคลาสในระบบฐานข้อมูลเชิงวัตถุจะเรียกว่าสคีมากราฟ(Schema Graph) ซึ่งเมื่อได้สคีมากราฟแล้ว วัตถุใด ๆในฐานข้อมูลจะมีความสัมพันธ์กันตามสคีมากราฟที่วางไว้

2.1.3 คลาสไดอะแกรม [1][7][8]

ในการพัฒนาระบบซอฟต์แวร์เชิงวัตถุผู้พัฒนาโปรแกรมจะมีการใช้งานวัตถุของคลาสและมีการสร้างความสัมพันธ์ระหว่างคลาสหรือวัตถุเหล่านั้น เช่น การสืบทอดคุณสมบัติของคลาส ดังนั้นในการสร้างแบบจำลองเชิงวัตถุ มีความจำเป็นอย่างยิ่งที่จะต้องสร้างไดอะแกรมที่แสดงถึงองค์ประกอบดังกล่าวทั้งหมดที่เรียกว่าคลาสไดอะแกรม

วัตถุประสงค์ของการสร้างคลาสไดอะแกรม เพื่อแสดงถึงโครงสร้างของระบบอันประกอบไปด้วยคลาส อินเตอร์เฟส และความสัมพันธ์ระหว่างคลาสเหล่านั้น ประกอบไปด้วย

คลาส(Class) คือแบบจำลองของวัตถุแต่ละชนิดในระบบ โดยในแต่ละคลาสประกอบไปด้วย 3 ส่วน ได้แก่

1. ชื่อคลาส
2. แอทริบิวต์ซึ่งประกอบไปด้วยชนิดของการเข้าถึง ชื่อแอทริบิวต์ ประเภทของแอทริบิวต์โดยแอทริบิวต์ภายในคลาสจะมี 2ประเภทคือ
   แอทริบิวต์เดี่ยว (Single-valued attributes) เป็นแอทริบิวต์ที่มีชนิดข้อมูลภายในเป็นชนิดข้อมูลแบบพื้นฐานเช่น Integer,Boolean,Short,Byte เป็นต้น และแอทริบิวต์แบบมัลติแอทริบิวต์ (Multi-valued attributes) เป็นแอทริบิวต์ที่มีชนิดข้อมูลภายในเป็นกลุ่มของชนิดข้อมูลแบบพื้นฐาน เช่น Array List Bag และ Set เป็นต้น
3. ส่วนสำหรับเมทธอดซึ่งประกอบไปด้วย ชนิดของการเข้าถึงเมทธอด ชื่อเมทธอดพารามิเตอร์ ประเภทค่าที่ส่งคืน

โดยสามารถแสดงรูปของคลาสและส่วนประกอบได้ดังนี้

CAR	ชื่อคลาส
-numberOfCar :integer -direction:Direction	แอทริบิวต์
+driveSpeed(speed:integer,direction:Direction)	เมทธอด

รูปที่ 2.1 การกำหนดแอทริบิวต์และเมทธอดในคลาส

อินเทอร์เฟส(Interface) เป็นการอธิบายถึงกลุ่มของเมทธอดที่ใช้ระบุถึงบริการที่เสนอโดยวัตถุของคลาสหนึ่งคลาสใด

มัลติพลิซิตี (Multiplicity) ที่แต่ละด้านของความสัมพันธ์ จะแสดงถึงจำนวนที่เป็นไปได้ของวัตถุในด้านนั้นต่อวัตถุแต่ละตัวในอีกด้านหนึ่ง ตัวอย่างเช่น “1” หมายความถึงมีได้ 1 ตัวเท่านั้น “0..1” หมายความถึงมีได้ 0 หรือ 1 ตัว เป็นต้น

คลอลิไฟเออร์ (Qualifier) เป็นแอทริบิวต์หรือกลุ่มของแอทริบิวต์ที่ค่าของแอทริบิวต์ดังกล่าวจะใช้อ้างอิงการเชื่อมโยงกันระหว่างคลาส 2 คลาสที่สัมพันธ์กัน เป็นแอทริบิวต์ของเส้นความสัมพันธ์แบบแอสโซซิเอชัน คอมโพซิชัน และแอกกริเกชัน โดยความสัมพันธ์ของ คอลลิไฟล์เออร์สามารถสร้างความสัมพันธ์เป็นคอลลิไฟล์แอสโซซิเอชัน(Qualified association) หรือวีคแอสโซซิเอชัน(Weak association) ซึ่งความสัมพันธ์คอลลิไฟล์แอสโซซิเอชัน จะรวมไปถึงความสัมพันธ์แบบคอลลิไฟล์คอมโพซิชัน และคอลลิไฟล์แอกกริเกชัน ด้วย

ความสัมพันธ์ระหว่างคลาส(Relationships) เป็นการอธิบายถึงความสัมพันธ์ระหว่างสิ่งของต่าง ๆ โดยที่ในแต่ละรูปแบบของความสัมพันธ์จะมีการสร้างสัญลักษณ์ในการสื่อความหมายที่แตกต่างกันของแต่ละความสัมพันธ์ จะประกอบด้วย

1 ความสัมพันธ์แบบแอสโซซิเอชัน (Association relationship) เป็นความสัมพันธ์ที่อธิบายการเชื่อมต่อระหว่างวัตถุของคลาสหนึ่งกับวัตถุของอีกคลาสหนึ่งที่สามารถโต้ตอบ (Interact) กันได้

2 ความสัมพันธ์แบบแอสโซซิเอชันคลาส (Association class relationship) เป็นความสัมพันธ์แบบหนึ่ง ที่เส้นที่เชื่อมความสัมพันธ์ระหว่างคลาส 2 คลาสจะมีคลาสเป็นของตัวเอง เพิ่มขึ้นมาในเส้นความสัมพันธ์นั้น ๆ

3 ความสัมพันธ์แบบรีเคอร์ซีพแอสโซซิเอชัน(Recursive association relationship) เป็นรูปแบบหนึ่งของความสัมพันธ์แบบแอสโซซิเอชัน อยู่ในรูปแบบที่คลาสหนึ่งคลาสมีความสัมพันธ์กับตัวของมันเอง

4 ความสัมพันธ์แบบดีเพนเดนซี (Dependency relationship) เป็นความสัมพันธ์ที่คลาสหนึ่งมีคุณสมบัติและพฤติกรรมขึ้นอยู่กับอีกคลาสหนึ่ง เช่นคลาส A มีความสัมพันธ์แบบขึ้นแก่กันกับคลาส B เมื่อเปลี่ยนแปลงค่าในคลาส A จะทำให้คลาส B มีการเปลี่ยนแปลงค่าไปด้วย

5 ความสัมพันธ์แบบแอกกรีเกชัน (Aggregation relationship) เป็นความสัมพันธ์แบบ “ประกอบด้วย (Has-a)” โดยจะมีคลาสซึ่งแสดงถึงสิ่งของที่ใหญ่กว่าที่ประกอบไปด้วยสิ่งของที่เล็กกว่า ความสัมพันธ์ชนิดนี้เป็นกรณีพิเศษของความสัมพันธ์แบบแอสโซซิเอชัน โดยสามารถแสดงได้ด้วยความสัมพันธ์แบบแอสโซซิเอชันที่มีปลายด้านที่เป็นสิ่งของที่ใหญ่กว่าเป็นสี่เหลี่ยมขนมเปียกปูนโปร่ง

6 ความสัมพันธ์แบบคอมโพซิชัน (Composition relationship) เป็นความสัมพันธ์รูปหนึ่งของความสัมพันธ์แบบแอกกรีเกชัน แต่จะแสดงถึงความเป็นเจ้าของที่ชัดเจนยิ่งขึ้น ในความสัมพันธ์แบบคอมโพซิชัน วัตถุที่ใหญ่กว่าจะมีหน้าที่จัดการเกี่ยวกับการสร้างและการทำลายวัตถุที่เล็กกว่า สามารถแสดงได้ด้วยความสัมพันธ์แบบแอสโซซิเอชันที่มีปลายด้านที่เป็นสิ่งของที่ใหญ่กว่าเป็นสี่เหลี่ยมขนมเปียกปูนทึบ

7 ความสัมพันธ์แบบคอลลิไฟล์แอสโซซิเอชัน(Qualified association relationship) เป็นความสัมพันธ์รูปแบบหนึ่งที่มีแอทริบิวต์หรือกลุ่มของแอทริบิวต์ที่ค่าของแอทริบิวต์ดังกล่าวจะใช้อ้างอิงการเชื่อมโยงกันระหว่างคลาส 2 คลาสที่สัมพันธ์กัน เป็นแอทริบิวต์ของเส้นความสัมพันธ์แบบแอสโซซิเอชัน คอมโพซิชันและแอกกริเกชัน

8 ความสัมพันธ์แบบเจเนอรัลไลเซชัน (Generalization relationship) เป็นความสัมพันธ์ที่อธิบายความสัมพันธ์ระหว่างซูเปอร์คลาสและสับคลาส ทำให้เกิดกลไกการสืบทอดคุณสมบัติขึ้นมาได้อาจมองว่าเป็นความสัมพันธ์แบบ “เป็นชนิดหนึ่งของ (Is-a-kind-of)”

9 ความสัมพันธ์แบบเรียลไลเซชัน (Realization relationship) ในคลาสไดอะแกรมจะใช้ความสัมพันธ์ชนิดนี้ในการระบุความสัมพันธ์ระหว่างอินเทอร์เฟสกับคลาส โดยอินเทอร์เฟสจะระบุถึงเมทธอดที่เสนอ และคลาสจะใช้บริการตามที่ระบุไว้ในอินเทอร์เฟสนั้น

.

2.1.4 มาตรฐานสำหรับฐานข้อมูลเชิงวัตถุ (Object-oriented database standard concept)[10]

การพัฒนาซอฟต์แวร์ในระดับต่าง ๆ ได้นำเอามาตรฐานเกี่ยวกับซอฟต์แวร์ต่าง ๆ เข้ามาใช้งานเพื่อช่วยให้การทำงานเป็นไปอย่างมีระบบและสะดวกต่อการพัฒนาซอฟต์แวร์ ในการพัฒนาระบบโดยการใช้ระบบฐานข้อมูลเชิงวัตถุก็ได้มีผู้นำเสนอเกี่ยวกับมาตรฐานสำหรับฐานข้อมูลเชิงวัตถุ เช่นเดียวกัน โดยมาตรฐานดังกล่าวได้ถูกนำเสนอขึ้นมาเพื่อเป็นแม่แบบช่วยให้การออกแบบเกี่ยวกับฐานข้อมูลเชิงวัตถุมีรูปแบบที่เป็นไปในทิศทางเดียวกันโดยในปัจจุบันได้มีการสร้างมาตรฐานสำหรับฐานข้อมูลเชิงวัตถุของโอดีเอ็มจีชื่อว่า โอดีเอ็มจี-2 (ODMG-2) ขึ้นมาในมาตรฐานสำหรับฐานข้อมูลเชิงวัตถุ ดังกล่าวจะประกอบไปด้วย

1. แบบจำลองเชิงวัตถุ เป็นแบบจำลองเบื้องต้นของการอธิบายส่วนประกอบต่างๆ ของวัตถุซึ่งประกอบไปด้วย คุณสมบัติ เมทธอด ความสัมพันธ์ของวัตถุ

2. ภาษาข้อกำหนดเชิงวัตถุ (Object specification language) ประกอบไปด้วย ภาษานิยามเชิงวัตถุ (ODL-Object Definition Language) เป็นภาษามาตรฐานที่ใช้อธิบายถึงโครงสร้างความสัมพันธ์ และลักษณะต่างๆ ของวัตถุ เป็นภาษาข้อกำหนด(Specification language) โดยที่ไม่ขึ้นอยู่กับภาษาโปรแกรม (Programming language) ซึ่งจะทำให้ยืดหยุ่นเป็นอย่างมาก โดยเมื่อนำเอาภาษานิยามเชิงวัตถุมาใช้สร้างสคีมาของฐานข้อมูลเชิงวัตถุจะสามารถนำเอาสคีมาที่ได้ดังกล่าวไปทำการเชื่อมโยงเข้ากับภาษาโปรแกรมแต่ละชนิดได้เช่น ภาษาจาวา ซีพลัสพลัส สมอลทอล์ก ซึ่งเรียกว่าการเชื่อมโยง (Binding) ภาษานิยามเชิงวัตถุจะประกอบไปด้วยการนิยามส่วนประกอบต่าง ๆ ของวัตถุคือ

- ชื่อคลาส (class name)
- การสืบทอดของสับคลาสจากซูเปอร์คลาส เรียกว่า extends
- ชื่อของกลุ่มของวัตถุที่ถูกสร้างจากคลาส เรียกว่า extent
- คีย์หลัก
- แอทริบิวต์
- เมทธอด
- ความสัมพันธ์ระหว่างคลาส

โดยการสร้างภาษานิยามวัตถุจะมีไวยากรณ์(Syntax) ดังรูปที่ 2.2

```
class className extends superclassName
(       extent  extentName      keys    keyName)
{       attribute <attributeType> attributeName;
        …
        relationship <relationType> relationName
        inverse <inverseSpec>;
        …
        method_definitions;
}
```

รูปที่ 2.2 แสดงไวยากรณ์ของภาษานิยามเชิงวัตถุตามมาตรฐานของโอดีเอ็มจี

```
Class MovieStar (extent MovieStars key ssn )
{
  attribute string name;
  attribute string ssn;
  attribute date birthdate;
  short  age();
  attribute struct address {String street, String city } address;
}
```

รูปที่ 2.3 แสดงตัวอย่างของภาษานิยามวัตถุที่ถูกสร้างเป็นสคีมาของฐานข้อมูลเชิงวัตถุ

3. ภาษาสอบถามข้อมูลเชิงวัตถุ (OQL-Object Query Language)

ภาษาสอบถามข้อมูลเชิงวัตถุ เป็นภาษาที่ใช้ในการสืบค้นข้อมูลที่ถูกจัดเก็บในระบบฐานข้อมูลเชิงวัตถุทำให้การเข้าถึงข้อมูลทำได้ง่าย สนับสนุนรูปแบบตามมาตรฐานของโอดีเอ็มจี มีลักษณะเหมือนกับภาษาสอบถามเชิงโครงสร้าง(SQL-Structure Query Language) สามารถสร้างผลลัพธ์ของข้อมูลที่มีความซับซ้อนยุ่งยากมากให้ปรากฏเป็นผลลัพธ์ได้ตามที่ผู้ใช้ต้องการได้ โดยลักษณะของภาษาสอบถามข้อมูลเชิงวัตถุ มีดังนี้

- เป็นภาษาที่ใกล้เคียงกับภาษาสอบถามเชิงโครงสร้าง
- เป็นภาษาที่มีแบบแผนและมีความยืดหยุ่นกับชนิดของข้อมูลที่ต้องการค้นหา
- สามารถใช้ภาษาสอบถามข้อมูลเชิงวัตถุ กับภาษาโปรแกรมทั่ว ๆไปได้

ผลลัพธ์ที่ได้จากการสืบค้นโดยภาษาในการสืบค้นข้อมูลเชิงวัตถุ สามารถให้ผลลัพธ์ในรูปแบบของรายการ (List) เซ็ต(Set) ค่าเดี่ยวๆ (Atomic value) ได้

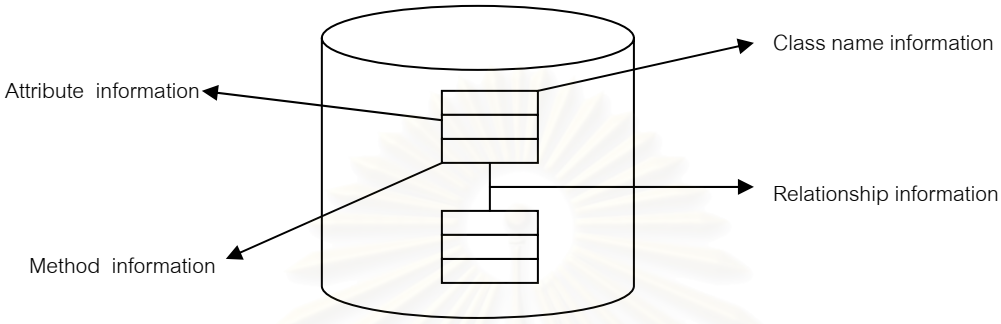
ตัวอย่างการใช้ภาษาสอบถามข้อมูลเชิงวัตถุ ในการค้นหาข้อมูลที่เก็บอยู่ในระบบฐานข้อมูลเชิงวัตถุ โดยข้อมูลที่ต้องการค้นหาคือชื่อของถนนของบ้านดาราที่อยู่จังหวัดกรุงเทพมหานคร สามารถเขียนในรูปของภาษาสอบถามข้อมูลเชิงวัตถุ ได้ดังนี้

```
Select MovieStar.name , MovieStar.address.street
From MovieStar
Where MovieStar.address.city = “bangkok” ;
```

รูปที่ 2.4 แสดงตัวอย่างของภาษาสอบถามข้อมูลเชิงวัตถุ

4. สคีมาฐานข้อมูลเชิงวัตถุ(Object-oriented database schema)

สคีมาฐานข้อมูลเชิงวัตถุ จะอธิบายถึงส่วนประกอบต่างๆ ของวัตถุว่าวัตถุภายในฐานข้อมูลประกอบไปด้วยแอทริบิวต์และเมทธอดอะไรบ้าง และอธิบายความสัมพันธ์ระหว่างวัตถุแต่ละชนิด โดยข้อมูลที่เก็บในสคีมาฐานข้อมูลเชิงวัตถุสามารถอธิบายส่วนประกอบได้ดังรูปต่อไปนี้



รูปที่ 2.5 แสดงส่วนประกอบต่าง ๆ ในฐานข้อมูลเชิงวัตถุ

จากรูปที่ 4.1 จะพบว่าในสคีมาของฐานข้อมูลเชิงวัตถุจะเก็บข้อมูลประเภทต่างๆ ที่จำเป็นต่อการสร้างฐานข้อมูล คือ

- การเก็บข้อมูลชื่อคลาส (Class's name information) ประกอบด้วย ชื่อคลาสในสคีมาฐานข้อมูลเชิงวัตถุและรายละเอียดของคลาสเพิ่มเติมตามแต่ละประเภทของฐานข้อมูล

- การเก็บข้อมูลแอทริบิวต์ (Attribute information) ประกอบด้วย ชื่อแอทริบิวต์ ประเภทของแอทริบิวต์ ค่าเริ่มต้นของแอทริบิวต์ รายละเอียดเพิ่มเติมของแอทริบิวต์ตามแต่ละประเภทของฐานข้อมูล

- การเก็บข้อมูลเมทธอด (Method information) ประกอบด้วย ชื่อของเมทธอด ประเภทข้อมูลของอาร์กูเมนต์ ชื่ออาร์กูเมนต์ ประเภทข้อมูลของการคืนค่ากลับ รายละเอียดเพิ่มเติมของเมทธอด ตามแต่ละประเภทของฐานข้อมูล

- การเก็บข้อมูลความสัมพันธ์ (Relationship information) ประกอบด้วยชื่อคลาสที่สัมพันธ์กับคลาสนั้นๆ มัลติพลิซิตีของคลาส มัลติพลิซิตีของคลาสที่สัมพันธ์กับคลาสนั้น ๆ ชื่อบทบาทของคลาส ชื่อบทบาทของคลาสที่สัมพันธ์กับคลาสนั้น ๆ ชื่อบทบาทแบบย้อนกลับของคลาส ชื่อคลาสของความสัมพันธ์แบบย้อนกลับ

2.2 งานวิจัยที่เกี่ยวข้อง

2.2.1 วิทยานิพนธ์เรื่อง “การแปลงฐานข้อมูลเชิงสัมพันธ์เป็นฐานข้อมูลเชิงวัตถุ” [4]

ในวิทยานิพนธ์ฉบับนี้ได้นำเสนอเครื่องมือและวิธีการที่ใช้เพื่อช่วยในการแปลงฐานข้อมูลในระบบจัดการฐานข้อมูลเชิงสัมพันธ์เป็นฐานข้อมูลในระบบจัดการฐานข้อมูลเชิงวัตถุได้แบ่งขั้นตอนการวิจัยออกเป็น 4 ขั้นตอน คือ

1. เตรียมแบบจำลองเชิงวัตถุเบื้องต้น ในขั้นตอนนี้เป็นการเตรียมคลาสเบื้องต้นจากตาราง-ต่างๆ เป็นการแปลงจากตารางในระบบฐานข้อมูลเชิงสัมพันธ์เป็นฐานข้อมูลเชิงวัตถุ โดยการสร้างเป็นคลาสขึ้นมาก่อนในขั้นตอนนี้จะพิจารณาคีย์เลือก (Candidate keys) และคีย์นอก (Foreign key) ผลลัพธ์ในขั้นตอนนี้คือแบบจำลองเชิงวัตถุเบื้องต้น(Initial object-oriented model)

2. ปรับแบบจำลองเชิงวัตถุ (Refine object-oriented model) ในขั้นตอนนี้จะนำแบบจำลองเชิงวัตถุเบื้องต้นที่ได้มาปรับให้เหมาะสมและดีขึ้นโดยอาศัยข้อมูลจากแบบจำลองข้อมูลเชิงตรรกะ (Logical data model) และข้อมูลจากพจนานุกรมข้อมูล (Data dictionary) ที่ได้จากการวิเคราะห์ระบบโดยพิจารณาคลาสที่สามารถรวมเป็นคลาสเดียวกันได้ ค้นหาคลาสที่มีลักษณะทั่วไป (Generalization class) สร้างคลาสที่เกี่ยวพันกัน (Association class) โดยอาศัยคีย์นอก จัดกลุ่มของคลาสที่มีความสัมพันธ์กันเป็นกลุ่ม ผลลัพธ์ที่ได้คือ แบบจำลองเชิงวัตถุที่ผ่านการปรับแล้วขั้นที่ 1

3. กำหนดเมทธอดให้กับคลาสต่างๆ ที่ได้จากขั้นตอนที่ 2

4. การสร้างฐานข้อมูลและนำข้อมูลเข้า

จากการศึกษาพบว่าสคีมาของฐานข้อมูลที่เลือกใช้ ใช้ในระบบฐานข้อมูลเชิงวัตถุ POET อย่างเดียวทำให้ไม่สามารถเปรียบเทียบและวัดได้ว่าฐานข้อมูลเชิงวัตถุที่ได้จากการแปลงนี้มีคุณภาพมากน้อยเพียงใด เมื่อทำการวัดโดยอาศัยหลักเกณฑ์การวัดซอฟต์แวร์ (Software metrics) และการออกแบบคลาสที่ได้จะมีการแปลงจากอีอาร์ไดอะแกรม (ER diagram-Entity Relationship diagram) ของระบบฐานข้อมูลเชิงสัมพันธ์ ก่อนที่จะนำเอาอีอาร์ไดอะแกรมนั้นๆ มาแปลงเป็นฐานข้อมูลเชิงวัตถุ จึงทำให้เกิดข้อสงสัยว่าอีอาร์ไดอะแกรมดังกล่าวมีคุณสมบัติต่าง ๆ เพียงพอต่อความต้องการของการสร้างระบบฐานข้อมูลเชิงวัตถุหรือไม่ และรูปแบบการสืบทอดวัตถุต่างๆ ทำอย่างมีประสิทธิภาพเพียงพอหรือไม่

แนวคิดที่ได้จากวิทยานิพนธ์เรื่องนี้คือขั้นตอนของการแปลงข้อมูลจากแผนภาพแบบหนึ่งเป็นชุดคำสั่งของการสร้างฐานข้อมูลที่ต้องการซึ่งสามารถนำแนวคิดที่ได้ดังกล่าวไปสร้างเป็นขั้นตอนของการแปลงคลาสไดอะแกรมเป็นสคีมาฐานข้อมูลเชิงวัตถุต่อไป

2.2.2 งานวิจัยเรื่อง “A knowledge base object-oriented database schema generator” [7]

ในงานวิจัยฉบับนี้ได้นำเสนอการออกแบบและการทำงานของเครื่องมือ KERO ที่นำไปใช้ในการสร้างสคีมาของฐานข้อมูลเชิงวัตถุ โดยเครื่องมือ KERO จะแปลงจากอีอาร์ไดอะแกรมเป็นสคีมาของฐานข้อมูลเชิงวัตถุโดยได้นำเสนอกฎของการแปลงข้อมูลดังกล่าวไปเป็นสคีมาของฐานข้อมูลเชิงวัตถุ และการแบ่งระเบียบวิธี (Methodology) ที่ใช้ในการออกแบบและการนำไปใช้งานจริงของเครื่องมือ KERO แบ่งออกเป็น 3 ส่วนใหญ่ ๆ คือ

1.การนำเสนออีอาร์ไดอะแกรม(Entity relationship diagram representation) ที่สามารถนำอีอาร์ไดอะแกรมนี้ ไปเข้ากระบวนการแปลงเป็นสคีมาของฐานข้อมูลเชิงวัตถุได้

2.การนำเสนอสคีมาของระบบฐานข้อมูลเชิงวัตถุ(Object-oriented database schema representation) ในส่วนนี้จะนำเสนอรูปแบบโดยทั่วไปของสคีมาของฐานข้อมูลเชิงวัตถุ

3.การนำเสนอกฎเกณฑ์ของการแปลงจากอีอาร์ไดอะแกรมไปเป็นสคีมาของระบบฐานข้อมูลเชิงวัตถุ(Translation rule from entity relationship diagram to object-oriented database schema representation) จะสร้างกฎของการแปลงจากอีอาร์ไดอะแกรมไปเป็นสคีมาของระบบฐานข้อมูลเชิงวัตถุ

แนวคิดที่ได้จากงานวิจัยนี้คือ ระเบียบวิธีที่ใช้ในการออกแบบและการประยุกต์จากกฎเพื่อนำไปสร้างเป็นเครื่องมือที่ประยุกต์กฎที่ต้องการ ซึ่งสามารถนำแนวคิดที่ได้ดังกล่าวไปสร้างเป็นระเบียบวิธีที่ใช้ในการออกแบบ และการประยุกต์จากกฎเพื่อนำไปสร้างเป็นเครื่องมือที่แปลงคลาสไดอะแกรมเป็นสคีมาฐานข้อมูลเชิงวัตถุต่อไป

2.2.3 วิทยานิพนธ์เรื่อง “การออกแบบกฎการแปลงยูเอ็มแอลซีเควนซ์ไดอะแกรมเป็นชุดคำสั่งภาษาจาวา” [3]

ในวิทยานิพนธ์ฉบับนี้ มีวัตถุประสงค์เพื่อออกแบบกฎการแปลงยูเอ็มแอลซีเควนซ์ไดอะแกรมเป็นชุดคำสั่งภาษาจาวา เพื่อสามารถนำกฎไปประยุกต์ใช้ในการสร้างเครื่องมือสำหรับการแปลงยูเอ็มแอลซีเควนซ์ไดอะแกรมเป็นชุดคำสั่ง ภาษาจาวาต่อไป การออกแบบจะเริ่มจากการออกแบบยูเอ็มแอลเมต้าโมเดลเพื่อใช้ในการแปลงยูเอ็มแอลซีเควนซ์ไดอะแกรม แล้วจึงออกแบบกฎการแปลงยูเอ็มแอลซีเควนซ์ไดอะแกรมเป็นชุดคำสั่งภาษาจาวา 8 กฎ คือ เมต้ารูลสำหรับการแปลงคลาสไดอะแกรมของเมทธอดที่ยูเอ็มแอลซีเควนซ์ไดอะแกรมอธิบาย เมต้ารูลสำหรับการแบ่งซีเควนซ์ เมต้ารูลสำหรับการเรียกเมทธอดที่มีเงื่อนไข และการแตกกิ่ง เมต้ารูลสำหรับการกำหนดค่าให้กับตัวแปร เมต้ารูลสำหรับการกำหนดค่าให้กับตัวชี้ เมต้ารูลสำหรับการสร้างวัตถุใหม่ เมต้ารูลสำหรับการเรียกเมทธอดของวัตถุที่มีอยู่แล้วและเมต้ารูลสำหรับการเรียกเมทธอดของตัววัตถุเอง

แนวคิดที่ได้จากงานวิทยานิพนธ์นี้คือรูปแบบของการสร้างกฎแบบต่างๆ จากซีเควนซ์ไดอะแกรมและรูปแบบของการเก็บข้อมูลจากซีเควนซ์ไดอะแกรมเพื่อแปลงเป็นชุดคำสั่งภาษาจาวา โดยที่จากแนวคิดที่ได้ดังกล่าวสามารถนำไปประยุกต์เพื่อสร้างกฎการแปลงคลาสไดอะแกรมเป็น สคีมาฐานข้อมูลเชิงวัตถุต่อไปได้

2.2.4. งานวิจัยเรื่อง “An extended NIAM conceptual schema model for object Database” [13]

ในงานวิจัยฉบับนี้ได้นำเสนอแบบจำลองสคีมาเชิงแนวคิด(Conceptual schema model) ที่นำเสนอรูปแบบและโครงสร้างที่นำเสนอข้อมูล (Data) และความสัมพันธ์ (Relation) โดยการใช้รูปแบบการนำเสนอแบบกราฟิก ในงานวิจัยฉบับนี้ที่ได้นำเสนอโมเดลของ ENIAM ซึ่งเป็นหนึ่งในแบบจำลองสคีมาเชิงแนวคิด และได้นำเสนอตัวอย่างของแผนภาพใน ENIAM และส่วนประกอบต่างๆ โดยการนำเสนอส่วนประกอบต่างๆ ที่ปรากฏใน ENIAM และได้อธิบายถึงความสัมพันธ์ระหว่างส่วนประกอบต่างๆ ของ ENIAM และได้นำเสนอรูปแบบของการแปลงจากตัว ENIAM ดังกล่าวมาเป็นคลาสไดอะแกรม และได้นำเอาผลลัพธ์จากการแปลงที่ได้ดังกล่าวมาแปลงเป็นสคีมาของฐานข้อมูลเชิงวัตถุโดยมีการแปลงอยู่ในรูปของฐานข้อมูลเชิงวัตถุคาเช่

แนวคิดที่ได้จากวิทยานิพนธ์เรื่องนี้คือขั้นตอนของการแปลงข้อมูลจากแผนภาพแบบหนึ่งเป็นชุดคำสั่งของการสร้างฐานข้อมูลที่ต้องการซึ่งสามารถนำแนวคิดที่ได้ดังกล่าวไปสร้างเป็นขั้นตอนของการแปลงคลาสไดอะแกรมเป็นสคีมาฐานข้อมูลเชิงวัตถุต่อไป

2.2.5 งานวิจัย “XML rule base source code generator for UML case tool” [5]

เป็นงานวิจัยที่มีวัตถุประสงค์เพื่อนำเสนอเค้าโครง (Framework) สำหรับการสร้างเครื่องมือที่สามารถทำให้เกิดโปรแกรมจากคลาสไดอะแกรมที่สามารถเพิ่มขยายได้ ในงานวิจัยนี้ได้ออกแบบส่วนต่อประสานสำหรับโปรแกรมประยุกต์ (Application program interface) ที่ใช้ในการสกัด (Extract) ข้อมูลของคลาสไดอะแกรมจากที่เก็บข้อมูลไว้ให้ ผู้ใช้ที่ต้องการสกัดข้อมูลของแต่ละเครื่องมือจะต้องทำการเพิ่มเติมตรรกะ สำหรับการสกัดข้อมูลจากที่เก็บข้อมูลที่มีรูปแบบเฉพาะนั้น ๆ และมีการนำเสนอเค้าโครงสำหรับการสร้างเครื่องมือที่สามารถทำให้เกิดโปรแกรมในภาษาต่าง ๆ ได้ขึ้นอยู่กับภาษาที่ต้องการแปลง

แนวคิดที่ได้จากวิทยานิพนธ์นี้คือรูปแบบของการเก็บข้อมูลจากคลาสไดอะแกรมเพื่อแปลงเป็นชุดคำสั่งที่ต้องการ จากแนวคิดที่ได้ดังกล่าวสามารถนำไปประยุกต์เพื่อออกแบบรูปแบบของการเก็บข้อมูลจากคลาสไดอะแกรมเพื่อนำไปแปลงเป็นสคีมาฐานข้อมูลเชิงวัตถุต่อไป

บทที่ 3

การออกแบบการเก็บข้อมูลของคลาสไดอะแกรมในเอ็กซ์เอ็มแอลสำหรับการแปลงเป็นสคีมาฐานข้อมูลเชิงวัตถุ

ในบทนี้จะนำเสนอแนวคิดการออกแบบกฎการแปลงยูเอ็มแอลคลาสไดอะแกรมเป็นสคีมาฐานข้อมูลเชิงวัตถุ โดยจะอธิบายภาพรวมของโครงสร้างการทำงานของเครื่องมือ หลังจากนั้นจะอธิบายรูปแบบการเก็บข้อมูลของคลาสไดอะแกรมเพื่อแปลงเป็นฐานข้อมูลเชิงวัตถุ โดยการใช้ภาษาเอ็กซ์เอ็มแอล รายละเอียดของการเก็บข้อมูลในเอ็กซ์เอ็มแอลจะมีการออกแบบดังนี้คือ

1.การออกแบบรูปแบบการเก็บข้อมูลเกี่ยวกับคลาสหรืออินเตอร์เฟส

2.การออกแบบรูปแบบการเก็บข้อมูลเกี่ยวกับความสัมพันธ์ประเภทต่าง ๆ คือ

2.1 ความสัมพันธ์แบบแอสโซซิเอชัน

2.2 ความสัมพันธ์แบบแอสโซซิเอชันคลาส

2.3 ความสัมพันธ์แบบรีเคอร์ซีพแอสโซซิเอชัน

2.4 ความสัมพันธ์แบบดีเพนเดนซี

2.5 ความสัมพันธ์แบบแอกกรีเกชัน

2.6 ความสัมพันธ์แบบคอมโพซิชัน

2.7 ความสัมพันธ์แบบคอลลิไฟล์แอสโซซิเอชัน

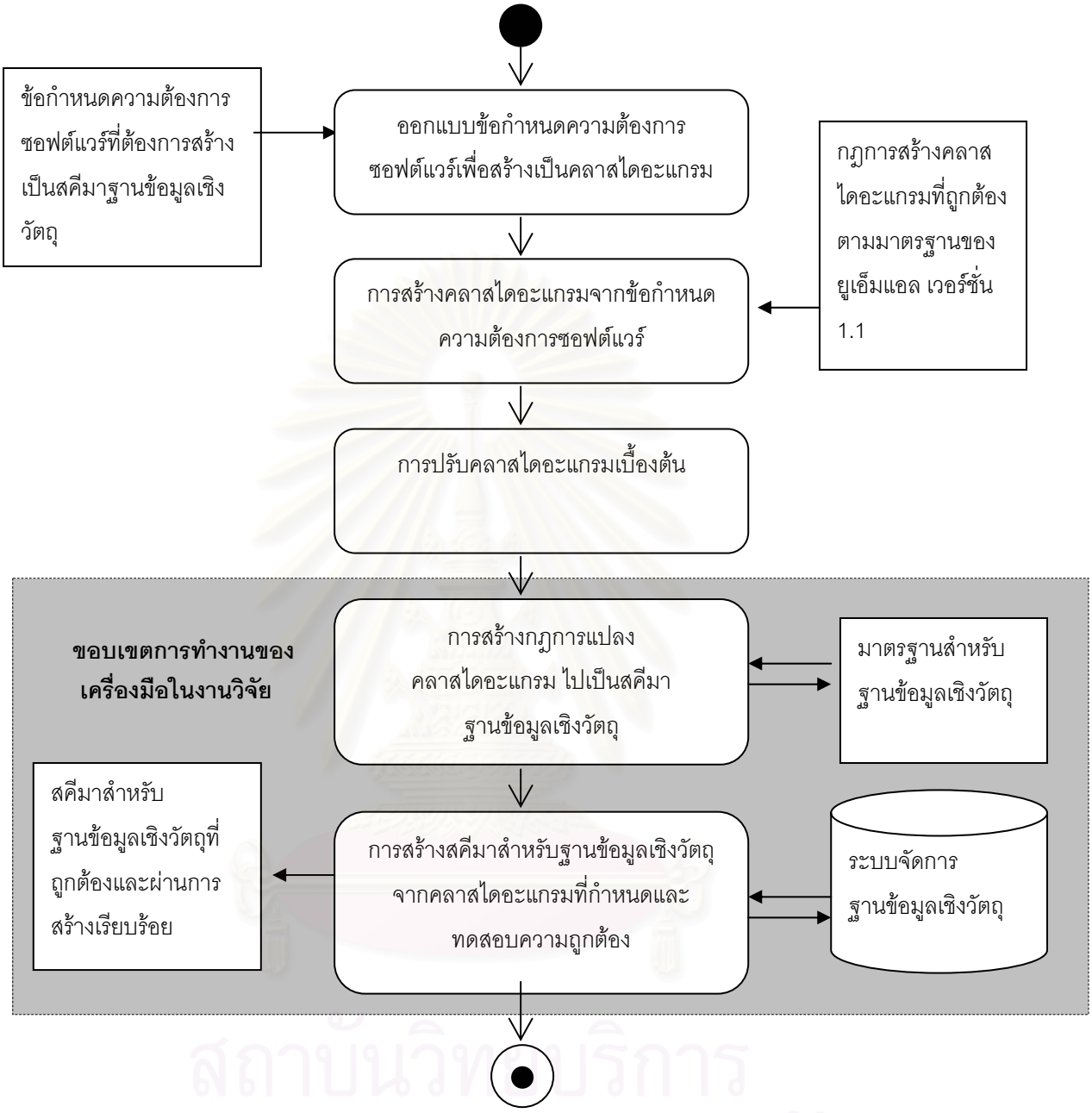
2.8 ความสัมพันธ์แบบเจเนอรัลไลเซชัน

2.9 ความสัมพันธ์แบบเรียลไลเซชัน

สำหรับกฎการแปลงยูเอ็มแอลคลาสไดอะแกรมเป็นสคีมาฐานข้อมูลเชิงวัตถุจะอธิบายในบทถัดไป

3.1 การออกแบบกฎการแปลงยูเอ็มแอลคลาสไดอะแกรมเป็นสคีมาฐานข้อมูลเชิงวัตถุ

กฎ (Rule) และ เครื่องมือ (Tool) ในงานวิจัยนี้เป็นกฎ และเครื่องมือ สำหรับการแปลงคลาสไดอะแกรม เป็นสคีมาของฐานข้อมูลเชิงวัตถุบนเว็บแอพลิเคชันโดยอาศัยมาตรฐานของสองมาตรฐาน ภาษายูเอ็มแอลและมาตรฐานสำหรับฐานข้อมูลเชิงวัตถุของโอดีเอ็มจี ภาพรวมของโครงสร้างการทำงานของเครื่องมือเป็นดังรูปที่ 3.1



รูปที่ 3.1 แสดงภาพรวมการวิจัย

จากรูปสรุปภาพรวมในการวิจัยโดยอาศัยกระบวนการทางวิศวกรรมซอฟต์แวร์ดังนี้

1. ออกแบบข้อกำหนดความต้องการซอฟต์แวร์เพื่อสร้างเป็นคลาสไดอะแกรม (Software requirements specification design)
2. การสร้างคลาสไดอะแกรมจากข้อกำหนดความต้องการซอฟต์แวร์ (Create class diagram from software requirements specification)
3. การปรับคลาสไดอะแกรมเบื้องต้น(Refine class diagram)

4.การสร้างกฎการแปลงคลาสไดอะแกรมเป็นสคีมาของฐานข้อมูลเชิงวัตถุ

(Create rules for transforming class diagram to object-oriented database schema)

5.การสร้างสคีมาสำหรับฐานข้อมูลเชิงวัตถุจากคลาสไดอะแกรมที่กำหนด

(Generate object-oriented database schema from class diagram)

โดยรายละเอียดของขั้นตอนดังกล่าวอธิบายได้ดังนี้

3.1.1 ออกแบบข้อกำหนดความต้องการซอฟต์แวร์เพื่อสร้างเป็นคลาสไดอะแกรม

ในขั้นตอนนี้จะเป็นส่วนของการเก็บรวบรวมข้อมูลของข้อกำหนดความต้องการซอฟต์แวร์ โดยรายละเอียดที่ได้ในแต่ละโปรแกรมที่พัฒนาจะมีความแตกต่างกันขึ้นอยู่กับรายละเอียด ของระบบนั้นๆ ผลลัพธ์ที่ได้ในขั้นตอนนี้จะปรากฏในรูปของเอกสารข้อกำหนดความต้องการซอฟต์แวร์ (Software requirement specification) ซึ่งอาจจะอยู่ในรูปของข้อความหรือแผนผัง แบบต่าง ๆ โดยผลลัพธ์ที่ได้ในขั้นตอนนี้จะถูกนำไปใช้ประกอบการพิจารณาในการสร้างแผนภาพคลาสไดอะแกรมในขั้นตอนต่อไป

3.1.2 การสร้างคลาสไดอะแกรมจากข้อกำหนดความต้องการซอฟต์แวร์

1. ศึกษาถึงโครงสร้างและรายละเอียดต่าง ๆ รวมถึงกฎเกณฑ์ของการสร้างคลาสไดอะแกรมที่จำเป็น และพิจารณาถึงความสัมพันธ์ระหว่างคลาสแบบต่างๆ ที่ปรากฏในแผนภาพ รายละเอียดในส่วนนี้คือ

1) การพิจารณาสัญลักษณ์ต่าง ๆ ที่จะนำมาใช้ในคลาสไดอะแกรม

2) การพิจารณากฎเกณฑ์ของการสร้างคลาสไดอะแกรม

3) การพิจารณาถึงความสัมพันธ์ของคลาสแบบต่าง ๆ

2. การวิเคราะห์และเตรียมข้อมูลของข้อกำหนดความต้องการซอฟต์แวร์ เพื่อสร้างคลาสไดอะแกรม เพื่อเป็นข้อมูลนำเข้าในกระบวนการแปลงเป็นสคีมาของฐานข้อมูลเชิงวัตถุในขั้นตอนต่อไป โดยที่การสร้างคลาสไดอะแกรมในส่วนนี้จะเป็นการแสดงถึงโครงสร้างของระบบโดยประกอบไปด้วยคลาสของวัตถุต่าง ๆ และความสัมพันธ์ระหว่างคลาส โดยคลาสไดอะแกรมมีความเกี่ยวเนื่องโดยตรงกับการสร้างฐานข้อมูลเชิงวัตถุ[4] การวิเคราะห์ในขั้นตอนนี้อาศัยข้อมูลนำเข้าคือ ข้อกำหนดความต้องการซอฟต์แวร์เบื้องต้นที่ได้จากข้อ 3.1.1 มาใช้พิจารณาในการเตรียมข้อมูลรายละเอียดเพื่อสร้างคลาสไดอะแกรม โดยที่แนวทางในการสร้างคลาสคลาสไดอะแกรมจากข้อกำหนดความต้องการซอฟต์แวร์ จะอยู่ในรูปของข้อความหรือยูสเคสไดอะแกรม จะใช้เทคนิคที่เรียกว่าการแก้ปัญหาแบบศึกษาสำนึก(Heuristic mapping) ซึ่งจะทำให้สามารถค้นหาคลาสต่าง ๆ ที่จะมีในระบบได้เป็นอย่างดีประกอบไปด้วยขั้นตอนดังนี้

1) พิจารณาข้อมูลนำเข้าจากเอกสารความต้องการซอฟต์แวร์

2) พิจารณาถึงกฎเกณฑ์สำหรับการแปลง

3) สร้างแอทริบิวท์ เมทธอดและชื่อคลาสในระบบ

โดยในส่วนของแอทริบิวต์จะประกอบไปด้วยชนิดของการเข้าถึงของแอทริบิวท์ ชื่อของแอทริบิวท์ ประเภทของแอทริบิวต์

ในส่วนของเมทธอดจะประกอบไปด้วยชนิดของการเข้าถึง ชื่อโอเปอเรชั่น พารามิเตอร์ ประเภทของค่าที่ส่งคืน

4) หาความสัมพันธ์ระหว่างคลาส

5) กำหนดอินเตอร์เฟส

6) สร้างคลาสไดอะแกรมที่ได้จากการวิเคราะห์ตามขั้นตอนข้างต้น ลงบนเอกสารเอชทีเอ็มแอล เพื่อใช้เป็นข้อมูลนำเข้าสำหรับการสร้างสคีมาของฐานข้อมูลเชิงวัตถุในขั้นตอนต่อไป ผลลัพธ์ที่ได้จากขั้นตอนนี้คือ คลาสไดอะแกรม

3.1.3 การปรับคลาสไดอะแกรมเบื้องต้น

ในขั้นตอนนี้จะนำเอาคลาสไดอะแกรมที่ได้จากขั้นตอนที่ 3.1.2 มาทำการปรับให้เหมาะสมและดีขึ้นโดยอาศัยการวิเคราะห์ระบบของผู้วิเคราะห์ เพื่อให้ข้อมูลในฐานข้อมูลมีความถูกต้องมากขึ้น โดยมีการพิจารณาถึงสิ่งต่าง ๆ ดังนี้

1) พิจารณาคลาสนำเข้าที่สามารถรวมเป็นคลาสเดียวกันได้

2) พิจารณาแอทริบิวต์ เมทธอดที่อาจจะเพิ่ม หรือลบในระบบ

3) ดูความสัมพันธ์ระหว่างคลาสที่อาจจะมีการเปลี่ยนแปลงหรือมีการปรับปรุงเพิ่มเติม

3.1.4 การสร้างกฎการแปลงคลาสไดอะแกรมเป็นสคีมาของฐานข้อมูลเชิงวัตถุ

ในส่วนของขั้นตอนนี้จะเป็นการสร้างข้อกำหนดต่างๆ ของการแปลงโดยสร้างข้อกำหนดของการแปลงคลาสไดอะแกรมเป็นสคีมาฐานข้อมูลเชิงวัตถุ

3.1.5 การสร้างสคีมาสำหรับฐานข้อมูลเชิงวัตถุจากคลาสไดอะแกรมที่กำหนด

จากข้อกำหนดที่ได้จากขั้นตอนที่ 3.1.4 จะถูกนำเอามาใช้เพื่อเป็นกฎเกณฑ์ในการสร้างสคีมาสำหรับฐานข้อมูลเชิงวัตถุและนำเอาส่วนของคลาสไดอะแกรมที่ได้ในขั้นตอนที่ 3.1.2 มาสร้างเป็น สคีมาของฐานข้อมูลเชิงวัตถุโดยใช้ กฎการสร้างสคีมาสำหรับฐานข้อมูลเชิงวัตถุ และนำเอาส่วนของคลาสไดอะแกรมมาใช้พิจารณาควบคู่กัน ผลลัพธ์ที่ได้จากขั้นตอนนี้คือสคีมาของฐานข้อมูลเชิงวัตถุที่พร้อมที่จะนำเอาไปใช้ในการสร้างฐานข้อมูลเชิงวัตถุต่อไป

3.2 การออกแบบรูปแบบการเก็บข้อมูลเกี่ยวกับคลาสหรืออินเตอร์เฟส

การออกแบบรูปแบบการเก็บข้อมูลเกี่ยวกับคลาสหรืออินเตอร์เฟส มีรูปแบบดังนี้

```
<class [interface] ComponentID=" "  CPNname=" "  Cvisibility=" "  Cxcenter =" "
Cycenter =" " CleftPos=" " CtopPos=" "  Cextent="">
        <attribute CPNname=" "  Visibility=" "  Type=" "  Initialvalue =" " ></attribute>
        <method CPNname=" "  Visibility=" "  RetType=" "> </method>
</class[/interface]>
```

รูปที่ 3.2 แสดงการออกแบบรูปแบบการเก็บข้อมูลเกี่ยวกับคลาสหรืออินเตอร์เฟส

การออกแบบรูปแบบการเก็บข้อมูลเกี่ยวกับคลาสหรืออินเตอร์เฟสมีดังนี้

1. ข้อมูลของคลาสหรืออินเตอร์เฟสอยู่ภายในแท็ก <interface> </interface> หรือแท็ก <class> </class>เช่น ชื่อคลาส การเข้าถึง รหัสของคลาส รายละเอียดตำแหน่งของการวางสัญลักษณ์บนหน้าจอ รายละเอียดประกอบด้วย

   1.1 ComponentID ระบุข้อมูลรหัสของสัญลักษณ์ของคลาสที่ถูกวาดขึ้นในหน้าจอ

   1.2 CPNname ระบุข้อมูลชื่อคลาส

   1.3 Cvisibility ระบุข้อมูลเกี่ยวกับการเข้าถึงข้อมูลของคลาสเช่น public private

   1.4 Cxcenter Cycenter ระบุตำแหน่งกึ่งกลาง x y ของสัญลักษณ์

   1.5 CleftPos CtopPos ระบุตำแหน่งด้านบน และด้านซ้ายของ สัญลักษณ์

   1.6 Cextent ระบุที่เก็บของกลุ่มวัตถุภายในคลาส

2. ข้อมูลของแอทริบิวต์ภายในคลาสอยู่ภายในแท็ก <attribute> </attribute > เช่น ชื่อแอทริบิวต์ การเข้าถึง ประเภทข้อมูล ค่าเริ่มต้น รายละเอียดประกอบด้วย

   2.1 CPNname ระบุข้อมูลเกี่ยวกับชื่อแอทริบิวต์

   2.2 Visiblility ระบุข้อมูลเกี่ยวกับการเข้าถึงข้อมูลของแอทริบิวต์เช่น public private

   2.3 Type ระบุข้อมูลเกี่ยวกับชนิดของแอทริบิวต์ ของแต่ละฐานข้อมูล

   2.4 Initialvalue ระบุค่าเริ่มต้นของแอทริบิวต์

3. ข้อมูลของเมทธอดภายในคลาสอยู่ภายในแท็ก <method></method> เช่น ชื่อเมทธอดการเข้าถึง ประเภทข้อมูลของการส่งค่ากลับคืน รายละเอียดประกอบด้วย

   3.1 CPNname ระบุข้อมูลเกี่ยวกับชื่อเมทธอด

   3.2 Visibility ระบุข้อมูลเกี่ยวกับการเข้าถึงข้อมูลของเมทธอด

3.3 RetType ระบุข้อมูลเกี่ยวกับประเภทข้อมูลของการส่งค่ากลับคืน

3.3 การออกแบบการเก็บข้อมูลเกี่ยวกับความสัมพันธ์ประเภทต่าง ๆ

ในส่วนนี้จะเป็นการออกแบบการเก็บข้อมูลเกี่ยวกับความสัมพันธ์ประเภทต่าง ๆ โดยในการแยกความแตกต่างของการเก็บข้อมูลความสัมพันธ์แต่ละประเภทจะพิจารณาจากสัญลักษณ์ที่ปรากฏในคลาสไดอะแกรมและกำหนดรหัสให้กับสัญลักษณ์นั้น ๆ เมื่อนำไปเก็บในรูปของเอ็กซ์เอ็มแอลจะระบุประเภทของความสัมพันธ์ที่ตำแหน่งแรกของแท็กของเอ็กซ์เอ็มแอลโดยจะระบุชื่อของความสัมพันธ์เพื่อแยกความแตกต่างของความสัมพันธ์แต่ละประเภท โดยแบ่งประเภทของความสัมพันธ์ได้ดังนี้

1) ความสัมพันธ์แบบแอสโซซิเอชัน

```
<association ComponentID=" " SourcePos="  " SourceID="  " SourceType="  " DestID="  "
DestType=" " X1=" " Y1=" " Xstart=" " Ystart=" " X2=" " Y2=" " RolA="  " RolB="  " MultiA="  "
MultiB="  " XrolA="  " XrolB="  " XmultiA="  " XmultiB="  "  YrolA="  " YrolB="  " YmultiA="  "
ymultiB="  " ></association>
```

รูปที่ 3.3 แสดงการออกแบบความสัมพันธ์แบบแอสโซซิเอชัน

จากรูปแบบการเก็บข้อมูลเกี่ยวกับความสัมพันธ์ระหว่างคลาสแบบแอสโซซิเอชัน อยู่ภายในแท็ก <association> </association> ประกอบไปด้วยข้อมูลที่ต้องเก็บต่อไปนี้

1 ComponentID ระบุข้อมูลเกี่ยวกับรหัสของความสัมพันธ์ระหว่างคลาส

2 SourceID DestID ระบุรหัสของคลาสต้นทางและคลาสปลายทาง

3 SourceType ระบุประเภทของสัญลักษณ์ที่อยู่ต้นทาง

4 X1 Y1 X2 Y2 ระบุตำแหน่งเริ่มต้น และตำแหน่งสุดท้าย ของสัญลักษณ์

5 RoleA RoleB ระบุบทบาทของคลาสต้นทางและปลายทาง

6 XroleA YroleA XroleB YroleB ระบุตำแหน่งของบทบาทของคลาสที่อยู่ต้นทางและปลายทาง

7 Multix Multiy ระบุ มัลติพลิซิตี ของคลาสที่อยู่ต้นทางและปลายทาง

8 Xmultia Ymultia Xmultib Ymultib ระบุตำแหน่งของ มัลติพลิซิตี ของคลาสที่อยู่ต้นทางและปลายทาง

9 DestType ระบุประเภทของสัญลักษณ์ที่อยู่ปลายทาง

2) ความสัมพันธ์แบบแอสโซซิเอชันคลาส

<associationclass ComponentID=" " SourcePos=" " SourceID=" " SourceType=" " DestID=" " DestType=" " X1=" " Y1=" " Xstart=" " Ystart=" " X2=" " Y2=" " RolA=" " RolB=" " MultiA=" " MultiB=" " XrolA=" " XrolB=" " XmultiA=" " XmultiB=" " YrolA=" " YrolB=" " YmultiA=" " ymultiB=" " ></associationclass>

รูปที่ 3.4 แสดงการออกแบบความสัมพันธ์แบบแอสโซซิเอชันคลาส

จากรูปแบบการเก็บข้อมูลเกี่ยวกับความสัมพันธ์ระหว่างคลาสแบบแอสโซซิเอชัน อยู่ภายในแท็ก <associationclass> </associationclass> ประกอบไปด้วยข้อมูลที่ต้องเก็บต่อไปนี้

1 ComponentID ระบุข้อมูลเกี่ยวกับรหัสของความสัมพันธ์ระหว่างคลาส

2 SourceID DestID ระบุรหัสของคลาสต้นทางและคลาสปลายทาง

3 SourceType ระบุประเภทของสัญลักษณ์ที่อยู่ต้นทาง

4 X1 Y1 X2 Y2 ระบุตำแหน่งเริ่มต้น และตำแหน่งสุดท้าย ของสัญลักษณ์

5 RoleA RoleB ระบุบทบาทของคลาสต้นทางและปลายทาง

6 XroleA YroleA XroleB YroleB ระบุตำแหน่งของบทบาทของคลาสที่อยู่ต้นทางและปลายทาง

7 Multix Multiy ระบุ มัลติพลิซิตี ของคลาสที่อยู่ต้นทางและปลายทาง

8 Xmultia Ymultia Xmultib Ymultib ระบุตำแหน่งของ มัลติพลิซิตี ของคลาสที่อยู่ต้นทางและปลายทาง

9 DestType ระบุประเภทของสัญลักษณ์ที่อยู่ปลายทาง

3) ความสัมพันธ์แบบรีเคอร์ซีพแอสโซซิเอชัน

<recursiveassociation ComponentID=" " SourcePos=" " SourceID=" " SourceType=" " DestID=" " DestType=" " X1=" " Y1=" " Xstart=" " Ystart=" " X2=" " Y2=" " RolA=" " RolB=" " MultiA=" " MultiB=" " XrolA=" " XrolB=" " XmultiA=" " XmultiB=" " YrolA=" " YrolB=" " YmultiA=" " ymultiB=" " ></recursiveassociation >

รูปที่ 3.5 แสดงการออกแบบความสัมพันธ์แบบรีเคอร์ซีพแอสโซซิเอชัน

จากรูปแบบการเก็บข้อมูลเกี่ยวกับความสัมพันธ์ระหว่างคลาสแบบรีเคอร์ซีพแอสโซซิเอชัน อยู่ภายในแท็ก < recursiveassociation > </ recursiveassociation > ประกอบไปด้วยข้อมูลที่ต้องเก็บต่อไปนี้

1 ComponentID ระบุข้อมูลเกี่ยวกับรหัสของความสัมพันธ์ระหว่างคลาส

2 SourceID DestID ระบุรหัสของคลาสต้นทางและคลาสปลายทาง

3 SourceType ระบุประเภทของสัญลักษณ์ที่อยู่ต้นทาง

4 X1 Y1 X2 Y2 ระบุตำแหน่งเริ่มต้น และตำแหน่งสุดท้าย ของสัญลักษณ์

5 RoleA RoleB ระบุบทบาทของคลาสต้นทางและปลายทาง

6 XroleA YroleA XroleB YroleB ระบุตำแหน่งของบทบาทของคลาสที่อยู่ต้นทางและปลายทาง

7 Multix Multiy ระบุ มัลติพลิซิตี ของคลาสที่อยู่ต้นทางและปลายทาง

8 Xmultia Ymultia Xmultib Ymultib ระบุตำแหน่งของ มัลติพลิซิตี ของคลาสที่อยู่ต้นทางและปลายทาง

9 DestType ระบุประเภทของสัญลักษณ์ที่อยู่ปลายทาง

4) ความสัมพันธ์แบบดีเพนเดนซี

```
<dependency ComponentID=" " SourcePos=" " SourceID=" " SourceType=" "
DestID=" " DestType=" " X1=" " Y1=" " Xstart=" " Ystart=" " X2=" " Y2=" " >
</dependency >
```

รูปที่ 3.6 แสดงการออกแบบความสัมพันธ์แบบดีเพนเดนซี

จากรูปแบบการเก็บข้อมูลเกี่ยวกับความสัมพันธ์ระหว่างคลาสแบบดีเพนเดนซีอยู่ภายในแท็ก < dependency > </ dependency > ประกอบไปด้วยข้อมูลที่ต้องเก็บต่อไปนี้

1 ComponentID ระบุข้อมูลเกี่ยวกับรหัสของความสัมพันธ์ระหว่างคลาส

2 SourceID DestID ระบุรหัสของคลาสต้นทางและคลาสปลายทาง

3 SourceType ระบุประเภทของสัญลักษณ์ที่อยู่ต้นทาง

4 X1 Y1 X2 Y2 ระบุตำแหน่งเริ่มต้น และตำแหน่งสุดท้าย ของสัญลักษณ์

5 DestType ระบุประเภทของสัญลักษณ์ที่อยู่ปลายทาง

5) ความสัมพันธ์แบบแอกกรีเกชัน

```
<aggregation ComponentID=" " SourcePos=" " SourceID=" " SourceType=" " DestID="
" DestType=" " X1=" " Y1=" " Xstart=" " Ystart=" " X2=" " Y2=" " RolA=" " RolB=" " MultiA="
" MultiB=" " XrolA=" " XrolB=" " XmultiA=" " XmultiB=" " YrolA=" " YrolB=" " YmultiA=" "
ymultiB=" " ></aggregation>
```

รูปที่ 3.7 แสดงการออกแบบความสัมพันธ์แบบแอกกรีเกชัน

จากรูปแบบการเก็บข้อมูลเกี่ยวกับความสัมพันธ์ระหว่างคลาสแบบแอกกรีเกชันอยู่ภายในแท็ก < aggregation > </aggregation > ประกอบไปด้วยข้อมูลที่ต้องเก็บต่อไปนี้

1 ComponentID ระบุข้อมูลเกี่ยวกับรหัสของความสัมพันธ์ระหว่างคลาส

2 SourceID DestID ระบุรหัสของคลาสต้นทางและคลาสปลายทาง

3 SourceType ระบุประเภทของสัญลักษณ์ที่อยู่ต้นทาง

4 X1 Y1 X2 Y2 ระบุตำแหน่งเริ่มต้น และตำแหน่งสุดท้าย ของสัญลักษณ์

5 RoleA RoleB ระบุบทบาทของคลาสต้นทางและปลายทาง

6 XroleA YroleA XroleB YroleB ระบุตำแหน่งของบทบาทของคลาสที่อยู่ต้นทางและปลายทาง

7 Multix Multiy ระบุ มัลติพลิซิตี ของคลาสที่อยู่ต้นทางและปลายทาง

8 Xmultia Ymultia Xmultib Ymultib ระบุตำแหน่งของ มัลติพลิซิตี ของคลาสที่อยู่ต้นทางและปลายทาง

9 DestType ระบุประเภทของสัญลักษณ์ที่อยู่ปลายทาง

6) ความสัมพันธ์แบบคอมโพสิชัน

<composition ComponentID=" " SourcePos=" " SourceID=" " SourceType=" " DestID=" " DestType=" " X1=" " Y1=" " Xstart=" " Ystart=" " X2=" " Y2=" " RolA=" " RolB=" " MultiA=" " MultiB=" " XrolA=" " XrolB=" " XmultiA=" " XmultiB=" " YrolA=" " YrolB=" " YmultiA=" " ymultiB=" " ></composition>

รูปที่ 3.8 แสดงการออกแบบความสัมพันธ์แบบคอมโพซิชัน

จากรูปแบบการเก็บข้อมูลเกี่ยวกับความสัมพันธ์ระหว่างคลาสแบบคอมโพซิชันอยู่ภายในแท็ก <composition> </composition > ประกอบไปด้วยข้อมูลที่ต้องเก็บต่อไปนี้

1 ComponentID ระบุข้อมูลเกี่ยวกับรหัสของความสัมพันธ์ระหว่างคลาส

2 SourceID DestID ระบุรหัสของคลาสต้นทางและคลาสปลายทาง

3 SourceType ระบุประเภทของสัญลักษณ์ที่อยู่ต้นทาง

4 X1 Y1 X2 Y2 ระบุตำแหน่งเริ่มต้น และตำแหน่งสุดท้าย ของสัญลักษณ์

5 RoleA RoleB ระบุบทบาทของคลาสต้นทางและปลายทาง

6 XroleA YroleA XroleB YroleB ระบุตำแหน่งของบทบาทของคลาสที่อยู่ต้นทางและปลายทาง

7 Multix Multiy ระบุ มัลติพลิซิตี ของคลาสที่อยู่ต้นทางและปลายทาง

8 Xmultia Ymultia Xmultib Ymultib ระบุตำแหน่งของ มัลติพลิซิตี ของคลาสที่อยู่ต้นทางและปลายทาง

9 DestType ระบุประเภทของสัญลักษณ์ที่อยู่ปลายทาง

7) ความสัมพันธ์แบบคอลลิไฟล์แอสโซซิเอชัน

<qualifiedassociation ComponentID=" " SourcePos=" " SourceID=" " SourceType=" " DestID=" " DestType=" " X1=" " Y1=" " Xstart=" " Ystart=" " X2=" " Y2=" " RolA=" " RolB=" " MultiA=" " MultiB=" " XrolA=" " XrolB=" " XmultiA=" " XmultiB=" " YrolA=" " YrolB=" " YmultiA=" " ymultiB=" " ></qualifiedassociation >

รูปที่ 3.9 แสดงการออกแบบคอลลิไฟล์แอสโซสิเอชัน

จากรูปแบบการเก็บข้อมูลเกี่ยวกับความสัมพันธ์ระหว่างคลาสแบบคอลลิไฟล์แอสโซสิเอชันอยู่ภายในแท็ก < qualifiedassociation > </ qualifiedassociation > ประกอบไปด้วยข้อมูลที่ต้องเก็บต่อไปนี้

1 ComponentID ระบุข้อมูลเกี่ยวกับรหัสของความสัมพันธ์ระหว่างคลาส

2 SourceID DestID ระบุรหัสของคลาสต้นทางและคลาสปลายทาง

3 SourceType ระบุประเภทของสัญลักษณ์ที่อยู่ต้นทาง

4 X1 Y1 X2 Y2 ระบุตำแหน่งเริ่มต้น และตำแหน่งสุดท้าย ของสัญลักษณ์

5 RoleA RoleB ระบุบทบาทของคลาสต้นทางและปลายทาง

6 XroleA YroleA XroleB YroleB ระบุตำแหน่งของบทบาทของคลาสที่อยู่ต้นทางและปลายทาง

7 Multix Multiy ระบุ มัลติพลิซิตี ของคลาสที่อยู่ต้นทางและปลายทาง

8 Xmultia Ymultia Xmultib Ymultib ระบุตำแหน่งของ มัลติพลิซิตี ของคลาสที่อยู่ต้นทางและปลายทาง

9 DestType ระบุประเภทของสัญลักษณ์ที่อยู่ปลายทาง

8) ความสัมพันธ์แบบเจเนอรัลไลเซชัน

<generalization ComponentID=" " SourcePos=" " SourceID =" " SourceType=" " DestId =" " DestType=" “ X1=" " Y1=" " Xstart=" " Ystart=" " X2=" " Y2=" " Xstop=" " Ystop=" " > </generalization>

รูปที่ 3.10 แสดงการออกแบบความสัมพันธ์แบบเจเนอรัลไลเซชัน

จากรูปแบบการเก็บข้อมูลเกี่ยวกับความสัมพันธ์ระหว่างคลาสแบบการสืบทอดคุณสมบัติระหว่างคลาสอยู่ภายใน แท็ก < generalization > </generalization > ประกอบไปด้วยข้อมูลที่ต้องเก็บดังนี้

1 ComponentID ระบุข้อมูลเกี่ยวกับรหัสของความสัมพันธ์ระหว่างคลาส

2 SourceID DestId ระบุรหัสของคลาสต้นทาง และรหัสของคลาสปลายทาง

3 SourceType ระบุประเภทของสัญลักษณ์ที่อยู่ต้นทาง

4 X1 Y1 X2 Y2 ระบุตำแหน่งเริ่มต้น และตำแหน่งสุดท้ายของสัญลักษณ์

5 DestType ระบุประเภทของสัญลักษณ์ที่อยู่ปลายทาง

9) ความสัมพันธ์แบบเรียลไลเซชัน

<realization ComponentID=" " SourcePos=" " SourceID =" " SourceType=" " DestId =" " DestType=" " X1=" " Y1=" " Xstart=" " Ystart=" " X2=" " Y2=" " Xstop=" " Ystop=" " > </realization >

รูปที่ 3.11 แสดงการออกแบบความสัมพันธ์แบบเรียลไลเซชัน

จากรูปแบบการเก็บข้อมูลเกี่ยวกับความสัมพันธ์แบบเรียลไลเซชันอยู่ภายใน แท็ก <realization > </ realization > ประกอบไปด้วยข้อมูลที่ต้องเก็บดังนี้

1 ComponentID ระบุข้อมูลเกี่ยวกับรหัสของความสัมพันธ์ระหว่างคลาส

2 SourceID DestId ระบุรหัสของคลาสต้นทาง และรหัสของคลาสปลายทาง

3 SourceType ระบุประเภทของสัญลักษณ์ที่อยู่ต้นทาง

4 X1 Y1 X2 Y2 ระบุตำแหน่งเริ่มต้น และตำแหน่งสุดท้ายของสัญลักษณ์

5 DestType ระบุประเภทของสัญลักษณ์ที่อยู่ปลายทาง

3.4 สรุปรูปแบบการเก็บข้อมูลของคลาสไดอะแกรมในเอ็กซ์เอ็มแอล

สรุปรูปแบบการเก็บข้อมูลของคลาสไดอะแกรมในเอ็กซ์เอ็มแอลมีรูปแบบดังนี้

```
<oodb_schema>
   <class>
        <attribute></attribute>
        <method></method>
    </ class >
   <relationship></relationship>
</oodb_schema >
```

รูปที่ 3.12 แสดงสรุปรูปแบบการเก็บข้อมูลของคลาสไดอะแกรมในเอ็กซ์เอ็มแอล

เมื่อปรากฏสัญลักษณ์ในยูเอ็มแอลคลาสไดอะแกรม จะเก็บข้อมูลภายในเอ็กซ์เอ็มแอล โดยจะสร้างข้อมูลต่าง ๆ เช่น คลาส แอทริบิวต์ เมทธอด ความสัมพันธ์ต่าง ๆระหว่างคลาส ภายในแท็ก <oodb_schema></oodb_schema> ซึ่งถือว่าเป็นแท็ก สำหรับการเก็บข้อมูล หลังจากนั้นข้อมูลต่าง ๆ ของสัญลักษณ์จะถูกเก็บภายใน แท็ก ต่าง ๆดังนี้

คลาสจะถูกเก็บภายในแท็ก <class></class> โดยอยู่ภายใต้แท็ก <oodb_schema> </oodb_schema>

อินเตอร์เฟสจะถูกเก็บภายในแท็ก <interface> </ interface > โดยอยู่ภายใต้แท็ก <oodb_schema></oodb_schema>

แอทริบิวต์ต่าง ๆ จะถูกเก็บภายในแท็ก <attribute></attribute> โดยอยู่ภายใต้แท็ก <class> </class>

เมทธอดต่างๆ จะถูกเก็บภายในแท็ก <method></method> โดยอยู่ภายใต้แท็ก <class> </class>

ความสัมพันธ์ระหว่างคลาสจะถูกเก็บภายในแท็ก <relationship> </relationship> โดยอยู่ภายใต้แท็ก <oodb_schema> </oodb_schema>

โดยที่แท็ก ต่างๆ เหล่านี้จะมีการเก็บข้อมูลต่าง ๆ ตามที่ได้อธิบายในหัวข้อที่ 3.2-3.4 ตามลำดับข้างต้น สำหรับตัวอย่างรูปแบบของการเก็บข้อมูลจะแสดงในภาคผนวก ค สรุปรูปแบบการเก็บข้อมูลในเอ็กซ์เอ็มแอล

บทที่ 4

กฎการแปลงยูเอ็มแอลคลาสไดอะแกรมเป็นสคีมาฐานข้อมูลเชิงวัตถุ

การสร้างกฎการแปลงคลาสไดอะแกรมเป็นสคีมาฐานข้อมูลเชิงวัตถุในบทที่ 4 จะเป็นกฎที่ใช้แปลงคลาสไดอะแกรมที่อยู่ในรูปของเอ็กซ์เอ็มแอลให้เป็นสคีมาตัวกลาง(Intermediate schema) โดยสคีมาฐานข้อมูลเชิงวัตถุที่ปรากฏในแต่ละกฎที่ถูกสร้างขึ้นในบทที่ 4 จะยังไม่ใช่สคีมาฐานข้อมูลเชิงวัตถุที่สมบูรณ์แต่จะอยู่ในรูปของสคีมาตัวกลางเพื่อที่จะนำไปสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุที่สมบูรณ์ในขั้นตอนถัดไป สคีมาตัวกลางที่ถูกสร้างโดยกฎแต่ละข้อในวิทยานิพนธ์ฉบับนี้จะมีรูปแบบดังนี้

```
class A{
   แอทริบิวต์
   เมทธอด
}
class B{
   แอทริบิวต์
   เมทธอด
}
A:
   ความสัมพันธ์ระหว่างคลาสของคลาส A
B:
   ความสัมพันธ์ระหว่างคลาสของคลาส B
```

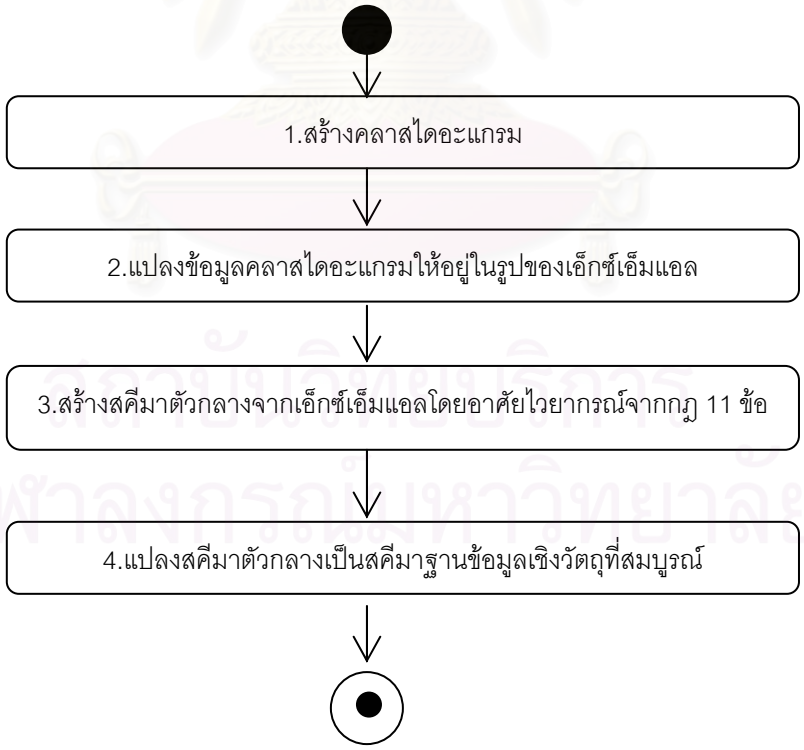
รูปที่ 4.1 แสดงรูปแบบของสคีมาตัวกลางสำหรับสร้างสคีมาฐานข้อมูลเชิงวัตถุ

หลังจากได้สคีมาตัวกลางดังรูปที่ 4.1 ซึ่งเป็นสคีมาตัวกลางที่ได้จากกฎทั้ง 11 ข้อ สคีมาตัวกลางดังกล่าวจะถูกนำไปแปลงเป็นสคีมาฐานข้อมูลเชิงวัตถุอีกครั้งเพื่อให้เป็นสคีมาฐานข้อมูลเชิงวัตถุที่สมบูรณ์โดยการนำความสัมพันธ์ระหว่างคลาสของแต่ละคลาสที่ถูกสร้างขึ้นไปใส่ในคลาสที่ถูกต้อง โดยรูปแบบของสคีมาฐานข้อมูลเชิงวัตถุที่สมบูรณ์ที่ได้จากการแปลงสคีมาตัวกลางมีรูปแบบดังนี้

```
class A{
   แอทริบิวต์
   เมทธอด
   ความสัมพันธ์ระหว่างคลาสของคลาส A
}
class B{
   แอทริบิวต์
   เมทธอด
   ความสัมพันธ์ระหว่างคลาสของคลาส B
}
```

รูปที่ 4.2 แสดงรูปแบบของสคีมาฐานข้อมูลเชิงวัตถุที่สมบูรณ์จากการแปลงสคีมาตัวกลาง

สำหรับภาพรวมของขั้นตอนการแปลงคลาสไดอะแกรมเป็นสคีมาฐานข้อมูลเชิงวัตถุมีดังนี้



รูปที่ 4.3 ภาพรวมของขั้นตอนการแปลงคลาสไดอะแกรมเป็นสคีมาฐานข้อมูลเชิงวัตถุ

การสร้างกฎการแปลงคลาสไดอะแกรมเป็นสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุจะแบ่งกฎสำหรับการแปลงออกเป็น 2 ส่วนคือ

ส่วนที่ 1 การสร้างกฎการแปลงคลาสไดอะแกรมเป็นสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุโดยการสร้างเป็นกฎทั่วไป (General rules) โดยไม่ระบุชนิดของฐานข้อมูล ในส่วนนี้จะอธิบายขั้นตอนการสร้างกฎการแปลง โดยไม่นำเอาไวยากรณ์ของการสร้างของแต่ละฐานข้อมูลมาอธิบายแต่จะอธิบายภาพรวมของฐานข้อมูลแทน

ส่วนที่ 2 การสร้างกฎการแปลงคลาสไดอะแกรมเป็นสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุโดยการประยุกต์จากกฎทั่วไปที่สร้างขึ้นมาในส่วนแรกเข้าประยุกต์กับฐานข้อมูลเชิงวัตถุแต่ละประเภท โดยในส่วนนี้จะมีการประยุกต์การใช้ฐานข้อมูลเชิงวัตถุ 3 ชนิด คือ

1. การสร้างกฎการแปลงคลาสไดอะแกรมเป็นสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุตามมาตรฐานของโอดีเอ็มจี
2. การสร้างกฎการแปลงคลาสไดอะแกรมเป็นสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุของฐานข้อมูลคาเช่
3. การสร้างกฎการแปลงคลาสไดอะแกรมเป็นสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุของฐานข้อมูลแมทิส

สาเหตุที่มีการจำแนกการแปลงออกเป็น 3 กรณีเนื่องจากต้องการที่จะเปรียบเทียบรูปแบบสคีมาฐานข้อมูลเชิงวัตถุที่ได้ในแต่ละกรณี และเหตุผลอีกประการหนึ่งเนื่องจากสคีมาของฐานข้อมูลเชิงวัตถุที่ได้ในแต่ละกรณีจะมีความแตกต่างกันจากการแปลง

จากสาเหตุดังกล่าวผู้วิจัยจึงได้ทำการสร้างกฎการแปลงคลาสไดอะแกรมเป็นสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุออกเป็น 3 กรณี โดยกรณีแรกจะเป็นการสร้างกฎการแปลงสคีมาฐานข้อมูลเชิงวัตถุที่เป็นต้นแบบมาตรฐาน ส่วนอีก 2 กรณีถัดไปจะเป็นสคีมาฐานข้อมูลเชิงวัตถุของฐานข้อมูลที่ใช้ในทางการค้าคือ ฐานข้อมูลคาเช่ และฐานข้อมูลแมทิส ประกอบไปด้วยกฎจำนวน 11 ข้อ คือ

กฎข้อที่ 1 กฎการแปลงคลาสและอินเตอร์เฟสเป็นสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุ (Rules1 -Rule for transform class and interface to intermediate schema of object-oriented database schema)

กฎข้อที่ 2 กฎการแปลงแอทริบิวต์และเมทธอดเป็นสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุ(Rules2- Rule for transform attributes and method to intermediate schema of object-oriented database schema)

กฎข้อที่ 3 กฎการแปลงความสัมพันธ์ระหว่างคลาสแบบแอสโซซิเอชันเป็นสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุ (Rules3- Rule for transform associations without attributes relationship to intermediate schema of object-oriented database schema)

กฎข้อที่ 4 กฎการแปลงความสัมพันธ์ระหว่างคลาสแบบแอสโซซิเอชันคลาสเป็นสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุ (Rules4 - Rule for transform associations class relationship to intermediate schema of object-oriented database schema)

กฎข้อที่ 5 กฎการแปลงความสัมพันธ์ระหว่างคลาสแบบรีเคอร์ซีพแอสโซซิเอชันเป็นสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุ (Rules5 -Rule for transform recursive associations relationship to intermediate schema of object-oriented database schema)

กฎข้อที่ 6 กฎการแปลงความสัมพันธ์ระหว่างคลาสแบบดีเพนเดนซีเป็นสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุ (Rules6 -Rule for transform dependency relationship to intermediate schema of object-oriented database schema)

กฎข้อที่ 7 กฎการแปลงความสัมพันธ์ระหว่างคลาสสัมพันธ์แบบแอกกรีเกชันเป็นสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุ(Rules7 -Rule for transform aggregation relationship to intermediate schema of object-oriented database schema)

กฎข้อที่ 8 กฎการแปลงความสัมพันธ์ระหว่างคลาสแบบคอมโพสิชันเป็นสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุ (Rules8 -Rule for transform composition relationship to intermediate schema of object-oriented database schema)

กฎข้อที่ 9 กฎการแปลงความสัมพันธ์ระหว่างคลาสแบบคอลลิไฟล์แอสโซซิเอชันเป็นสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุ(Rules9 -Rule for transform qualified association relationship to intermediate schema of object-oriented database schema)

กฎข้อที่ 10 กฎการแปลงความสัมพันธ์ระหว่างคลาสแบบเจนเนอรัลไลเซชันเป็นสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุ(Rules10 -Rule for transform generalization relationship to intermediate schema of object-oriented database schema)

กฎข้อที่ 11 กฎการแปลงความสัมพันธ์ระหว่างคลาสแบบเรียลไลเซชันเป็นสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุ(Rules11 -Rule for transform realization relationship to intermediate schema of object-oriented database schema)

การสร้างกฎการแปลงยูเอ็มแอลคลาสไดอะแกรมเป็นสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุในงานวิจัยชิ้นนี้อาศัยไวยากรณ์ในรูปแบบของไวยากรณ์บีเอ็นเอฟ (BNF-Backus Naur Form)[12] เพื่อเป็นต้นแบบสำหรับการสร้างกฎทั้ง 11 ข้อ สาเหตุของการสร้างกฎทั้ง 11 ข้อมาจากความบ่อยของการนำรูปแบบดังกล่าวไปใช้งาน โดยที่กฎทั้ง 11 ข้อดังกล่าวยังไม่ครบทุกกรณีของความสัมพันธ์ที่มีในคลาสไดอะแกรม รูปแบบของสัญลักษณ์และความหมายของสัญลักษณ์ที่ปรากฏในกฎจะอธิบายในตารางที่ 4.1

ตารางที่ 4.1 แสดงความหมายของสัญลักษณ์ที่ใช้ในกฎการแปลงยูเอ็มแอลคลาสไดอะแกรมเป็นสคีมาฐานข้อมูลเชิงวัตถุ

สัญลักษณ์	ความหมายของสัญลักษณ์
<NON-TERMINAL>	<>ใช้แยกความแตกต่างของสัญลักษณ์ที่เป็นนอนเทอร์มินอล (non-terminal symbols) จากสัญลักษณ์ที่เป็นเทอร์มินอล (terminal symbols) โดยข้อความที่อยู่ภายในเครื่องหมาย <> จะเป็นนอนเทอร์มินอล
::=	แสดงการถูกกำหนดเป็น ("is defined as") โดยสัญลักษณ์ที่อยู่ซ้ายมือสามารถกำหนดเป็นสัญลักษณ์ที่อยู่ขวามือได้
|	แสดงความหมายหรือ ("or")
symbols	แสดงสัญลักษณ์ที่เป็นเทอร์มินอลโดยแทนด้วยตัวอักษรเอียง
[ ]	แสดงตัวเลือกที่สามารถจะเลือกหรือไม่เลือกก็ได้(0..1)
< ø >	แสดงเซ็ตว่าง
{ }	ใช้แสดงสัญลักษณ์ของการทำซ้ำ(0..n)
“ ”	ใช้ใส่ที่คำที่ต้องการให้เป็นไวยากรณ์ที่เป็นเทอร์มินอล
<NON-TERMINAL1> **<NON-TERMINAL2>**	ใช้แสดงการเชื่อมต่อกันของสัญลักษณ์ที่เป็นนอนเทอร์มินอล จำนวน 2 ชุด ลำดับของการวางจะมีผลต่อความถูกต้องของสคีมา
<NON-TERMINAL$_{NAME}$>	แสดงสัญลักษณ์ที่เป็นนอนเทอร์มินอลที่มีการอ้างอิงชื่อจากคลาสไดอะแกรม โดยตัวห้อยจะแสดงชื่อของคลาส เมทธอด แอทริบิวต์ จากคลาสไดอะแกรมในที่นี้ **NON-TERMINAL$_{NAME}$** หมายถึงการนิยาม **NON-TERMINAL** ที่มีชื่อ **NAME**
CLASSNAME$_{CLASSNAME}$	ใช้แสดงสัญลักษณ์ที่เป็นเทอร์มินอล โดยข้อมูลที่ได้มาจากข้อมูลภายในคลาสไดอะแกรมโดยตัวห้อยจะแสดงชื่อของคลาส เมทธอด แอทริบิวต์

ข้อกำหนดเบื้องต้นของการสร้างกฎ

กรณีที่มีคลาสมากกว่า 1 คลาสภายในคลาสไดอะแกรม จะแปลงคลาสทุกคลาสให้เสร็จก่อน จากนั้นหากคลาสที่ปรากฏมีความสัมพันธ์ระหว่างคลาส จะแปลงความสัมพันธ์ระหว่างคลาสต่อจากคลาสต่าง ๆ ที่แปลงเสร็จแล้ว โดยความสัมพันธ์ระหว่างคลาสที่แปลงจะระบุชื่อคลาสไว้เพื่อให้สามารถนำไปแปลงเป็นสคีมาที่สมบูรณ์ได้ถูกต้องต่อไป แสดงดังรูปที่ 4.1- 4.2

ไวยากรณ์เริ่มต้นของการสร้างสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุของทั้ง 3 ฐานข้อมูลมีดังนี้

```
<RULESCHEMA-OODB>    ::= {<INTERFACEDEF_NAME>}{<CLASSDEF_NAME>}{<RELATIONSHIP_NAME>}

<RELATIONSHIP_NAME>::= {<RELATIONSHIP-H_NAME>}{<RELATIONSHIP-D_NAME>}

<RELATIONSHIP-H_NAME>::=[<GENERALIZATION>][<REALIZATION>]

<RELATIONSHIP-LIST_NAME> ::= relationship CLASSNAME_NAME|
                             relationship <COLLECTION-TYPE><CLASSNAME_NAME>
<RELATIONSHIP-FLAG_NAME> ::=CLASSNAME_NAME:

<RELATIONSHIP-D_NAME>::=<ASSOCIATION> | <ASSOCIATIONCLASS> |
                        <RECURSIVEASSOCIATION> | <DEPENDENCY> |
                        <AGGREGATION> | <COMPOSITION> |
                        <QUALIFIEDASSOCIATION>
```

รูปที่ 4.4 แสดงไวยากรณ์เริ่มต้นของการสร้างสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุของทั้ง 3 ฐานข้อมูล

คำอธิบายไวยากรณ์รวมของการสร้างสคีมาฐานข้อมูลเชิงวัตถุ มีดังนี้

1. **<RULESCHEMA-OODB>** จะอธิบายรายละเอียดของสคีมาว่าภายในจะประกอบไปด้วยข้อมูลอะไรบ้าง เช่น คลาส (**<CLASSDEF_NAME>**) อินเตอร์เฟส (**<INTERFACEDEF_NAME>**) ความสัมพันธ์ระหว่างคลาส (**<RELATIONSHIP_NAME>**)

2. **<CLASSDEF_NAME>** จะอธิบายรายละเอียดของคลาสคือ

3. **<INTERFACEDEF_NAME>** จะอธิบายรายละเอียดของอินเตอร์เฟส

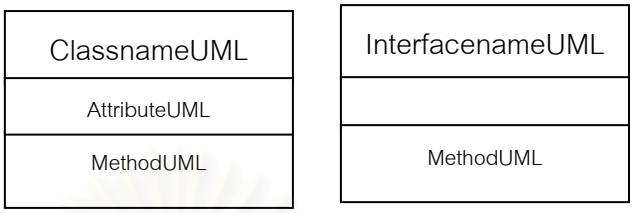
4. **<RELATIONSHIP_NAME>** จะอธิบายรายละเอียดความสัมพันธ์ระหว่างคลาสแบบต่างๆ เช่น

   - **<ASSOCIATION>** อธิบายความความสัมพันธ์แบบแอสโซซิเอชันคลาส
   - **<AGGREGATION>** อธิบายความความสัมพันธ์แบบแอกกรีเกชัน

รายละเอียดต่าง ๆ ของกฎทั้ง 11 ข้อมีดังนี้

4.1 กฎข้อที่ 1 กฎการแปลงคลาสและอินเตอร์เฟสเป็นสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุ

รูปแบบของคลาสไดอะแกรม



รูปที่ 4.5 แสดงคลาสไดอะแกรมที่มีลักษณะเป็นคลาสและอินเตอร์เฟส

ขั้นตอนการแปลงคลาสไดอะแกรมให้อยู่ในรูปแบบของสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุ

1. ชื่อคลาสในคลาสไดอะแกรม(ClassnameUML) จะถูกแปลงเป็นชื่อคลาสใน สคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุ(***CLASSNAME***$_{NAME}$) โดยการระบุคำที่ระบุว่าเป็นคลาส ชื่อคลาส ตามด้วยส่วนที่อธิบายลักษณะเพิ่มเติมของคลาส เช่น ชื่อที่เก็บกลุ่มของวัตถุที่ถูกสร้างจากคลาส (***EXTENTOFCLASS***$_{NAME}$) คีย์ที่ระบุถึงวัตถุแต่ละตัวที่จะทำให้วัตถุนั้น ๆ ยูนิค หรือตัวระบุวัตถุ (***KEYOFCLASS***$_{NAME}$) และค่าอื่น ๆ ตามชนิดของฐานข้อมูล

ชื่ออินเตอร์เฟสในคลาสไดอะแกรม(InterfacenameUML) จะถูกแปลงเป็นชื่ออินเตอร์เฟสในสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุ(***INTERFACENAME***$_{NAME}$) โดยการระบุคำที่ระบุว่าเป็นอินเตอร์เฟส ชื่ออินเตอร์เฟส

2. แอทริบิวต์ต่าง ๆ ในคลาสไดอะแกรม(AttributeUML)จะถูกสร้างเป็นแอทริบิวต์ในสคีมาตัวกลางสำหรับสร้างเป็น สคีมาฐานข้อมูลเชิงวัตถุ (<**ATTRIBUTE**$_{NAME}$>)

3. เมทธอดต่างๆ ในคลาสไดอะแกรม(methodUML)จะถูกสร้างเป็นเมทธอดในสคีมาตัวกลางสำหรับสร้างเป็น สคีมาฐานข้อมูลเชิงวัตถุ(<**METHOD**$_{NAME}$>)

ไวยากรณ์การสร้างสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุ

1. ไวยากรณ์ของการสร้างสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุตามมาตรฐานของโอดีเอ็มจี

```
<INTERFACEDEF_NAME > ::= interface  INTERFACENAME_NAME
                         "{"
                              {<METHOD_NAME>}
                         "}"

<CLASSDEF_NAME> ::=<CLASSHEADER_NAME>
                        "{"
                           {<CLASSBODY_NAME>}
                        "}"
<CLASSHEADER_NAME> ::= class CLASSNAME_NAME
                       (extent  EXTENTOFCLASS_NAME  [key  <KEY-LIST_NAME>])

<KEY-LIST_NAME> ::= <TYPE> KEYOFCLASS_NAME [{,<TYPE> KEYOFCLASS_NAME}]

<CLASSBODY_NAME>::={<ATTRIBUTE_NAME>}
                   {<METHOD_NAME>}
```

รูปที่ 4.6 ไวยากรณ์เบื้องต้นของการสร้างคลาสตามมาตรฐานของโอดีเอ็มจี

2. ไวยากรณ์ของการสร้างสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุของฐานข้อมูลคาเช่

```
<INTERFACEDEF_NAME > ::= interface  INTERFACENAME_NAME
                         "{"
                              {<METHOD_NAME>}
                         "}"

<CLASSDEF_NAME> ::=<CLASSHEADER_NAME>
                        "{"
                           {<CLASSBODY_NAME>}
                        "}"
<CLASSHEADER_NAME> ::= class  Package.CLASSNAME_NAME

<KEY-LIST_NAME> ::= KEYOFCLASS_NAME as %<TYPE> [{,KEYOFCLASS_NAME as %<TYPE>}]

<CLASSBODY_NAME>::={<ATTRIBUTE_NAME>}
                   {<METHOD_NAME>}
```

รูปที่ 4.7 ไวยากรณ์เบื้องต้นของการสร้างคลาสตามมาตรฐานของฐานข้อมูลคาเช่

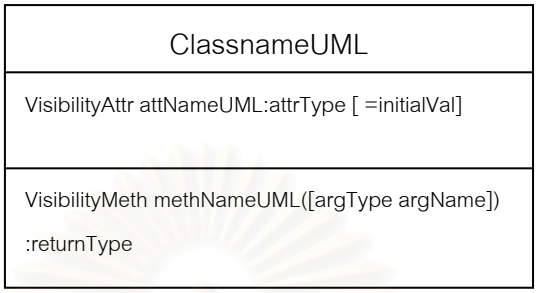
3. ไวยากรณ์ของการสร้างสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุของฐานข้อมูลแมทิส

```
<INTERFACEDEF_NAME > ::= interface  INTERFACENAME_NAME
                         "{"
                              {<METHOD_NAME>}
                         "}"
<CLASSDEF_NAME> ::=<CLASSHEADER_NAME>
                        "{"
                           {<CLASSBODY_NAME>}
                        "}"
<CLASSHEADER_NAME> ::= interface  CLASSNAME_NAME

<KEY-LIST_NAME> ::= <TYPE> KEYOFCLASS_NAME [{,<TYPE> KEYOFCLASS_NAME}]

<CLASSBODY_NAME>::={<ATTRIBUTE_NAME>}
                   {<METHOD_NAME>}
```

รูปที่ 4.8 ไวยากรณ์เบื้องต้นของการสร้างคลาสตามมาตรฐานของฐานข้อมูลแมทิส

4.2 กฎข้อที่ 2 กฎการแปลงแอทริบิวต์และเมทธอดเป็นสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุ

รูปแบบของคลาสไดอะแกรม



รูปที่ 4.9 แสดงคลาสไดอะแกรมที่มีแอทริบิวต์และเมทธอด

ขั้นตอนการแปลงคลาสไดอะแกรมให้อยู่ในรูปแบบของสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุ

1. ชื่อคลาสในคลาสไดอะแกรมจะถูกแปลงเป็นสคีมาตัวกลางสำหรับสร้างเป็น สคีมาฐานข้อมูลเชิงวัตถุตามกฎข้อที่ 1

2. แอทริบิวต์ต่างๆ ในคลาสไดอะแกรมจะถูกสร้างเป็นแอทริบิวต์ในสคีมาตัวกลางสำหรับสร้างเป็น สคีมาฐานข้อมูลเชิงวัตถุโดยการเพิ่มคำเพื่อระบุว่าเป็นแอทริบิวต์ และระบุประเภทของแอทริบิวต์ที่ถูกสร้างโดยสามารถเลือกได้จากประเภทต่อไปนี้ *String Date Integer Char Short* การระบุการเข้าถึงของแอทริบิวต์ ต่าง ๆ จากฐานข้อมูลเชิงวัตถุเช่น *public private protected* (ในฐานข้อมูลเชิงวัตถุจะไม่มีการใช้งานของส่วนนี้) การระบุค่าเริ่มต้นของแอทริบิวต์ การระบุชื่อแอทริบิวต์ และมีรายละเอียดเพิ่มเติมของแต่ละฐานข้อมูล การแปลงมีลักษณะดังนี้

- AttNameUML ในคลาสไดอะแกรมจะถูกแปลงเป็นชื่อแอทริบิวต์ในสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุ(*ATTRIBUTENAME*$_{NAME}$)

- AttrTypeในคลาสไดอะแกรมจะถูกแปลงเป็น ประเภทของแอทริบิวต์ ในสคีมาตัวกลางสำหรับสร้างเป็น สคีมาฐานข้อมูลเชิงวัตถุโดยแบ่งเป็นแอทริบิวต์เดี่ยว หรือแอทริบิวต์แบบมัลติแอทริบิวต์ (<**TYPE**>)

3. เมทธอดต่างๆ ในคลาสไดอะแกรมจะถูกสร้างเป็นเมทธอดในสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุโดยการเพิ่มคำเพื่อระบุว่าเป็นเมทธอด และประเภทข้อมูลการคืนค่ากลับของเมทธอด อาร์กูเม้นต์ต่าง ๆ ของเมทธอด ประเภทข้อมูลของอาร์กูเม้นต์ของ เมทธอด การเข้าถึงของเมทธอด(ในฐานข้อมูลเชิงวัตถุจะไม่มีการใช้งานของส่วนนี้) การแปลงมีดังนี้

- MethNameUMLในคลาสไดอะแกรมจะถูกแปลงเป็น ชื่อของเมทธอดในสคีมาตัวกลางสำหรับสร้างเป็น สคีมาฐานข้อมูลเชิงวัตถุ(*METHODNAME*$_{NAME}$)

- ArgTypeในคลาสไดอะแกรมจะถูกแปลงเป็น ประเภทข้อมูลของอาร์กูเมนต์ในสคีมาตัวกลางสำหรับสร้างเป็น สคีมาฐานข้อมูลเชิงวัตถุ(<**TYPE**>)

- ArgNameในคลาสไดอะแกรมจะถูกแปลงเป็นชื่ออาร์กูเมนต์ในสคีมาตัวกลางสำหรับสร้างเป็น สคีมาฐานข้อมูลเชิงวัตถุ(*ARGUMENTNAME*$_{NAME}$)

ไวยากรณ์การสร้างสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุ

1. ไวยากรณ์ของการสร้างสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุตามมาตรฐานของโอดีเอ็มจี

```
<METHOD_NAME> ::=<TYPE>  METHODNAME_NAME  ([<ARGUMENT-LIST>])

<ARGUMENT-LIST> ::= <TYPE> ARGUMENTNAME_NAME[{,<TYPE> ARGUMENTNAME_NAME}]

<TYPE>::= <PRIMITIVE>|<NON-PRIMITIVE>

<PRIMITIVE>::= String | Date | Integer | Char | Short

<NON-PRIMITIVE>::= <STRUCT>|<COLLECTION>|<CLASS_NAME>

<STRUCT>  ::= STRUCTNAME

<COLLECTION> ::= <COLLECTION-TYPE>"<" <PRIMITIVE> ">" |
                 <COLLECTION-TYPE>"<" <NON-PRIMITIVE> ">"

<COLLECTION-TYPE> ::= set | bag| list

<CLASS_NAME >::=CLASSNAME_NAME

<ATTRIBUTE_NAME> ::= attribute  <TYPE>  ATTRIBUTENAME_NAME;
```

รูปที่ 4.10 แสดงไวยากรณ์เบื้องต้นของการสร้างแอทริบิวต์และเมทธอดของโอดีเอ็มจี

2. ไวยากรณ์ของการสร้างสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุของฐานข้อมูลคาเช่

```
<METHOD_NAME> ::=<TYPE>  METHODNAME_NAME  ([<ARGUMENT-LIST>])

<ARGUMENT-LIST> ::=ARGUMENTNAME_NAME as %<TYPE>[{,ARGUMENTNAME_NAME as %<TYPE> }]

<TYPE>::= <PRIMITIVE>|<NON-PRIMITIVE>

<PRIMITIVE>::= String | Date | Integer | Char | Short

<NON-PRIMITIVE>::= <STRUCT>|<COLLECTION>|<CLASS_NAME>

<STRUCT>  ::= STRUCTNAME

<COLLECTION> ::= <COLLECTION-TYPE>"<" <PRIMITIVE> ">" |
                 <COLLECTION-TYPE>"<" <NON-PRIMITIVE> ">"

<COLLECTION-TYPE> ::= set | bag| list
<CLASS_NAME >::=CLASSNAME_NAME

<ATTRIBUTE_NAME> ::= property   ATTRIBUTENAME_NAME as %<TYPE> ;
```

รูปที่ 4.11 แสดงไวยากรณ์เบื้องต้นของการสร้างแอทริบิวต์และเมทธอดตามมาตรฐานของฐานข้อมูลคาเช่

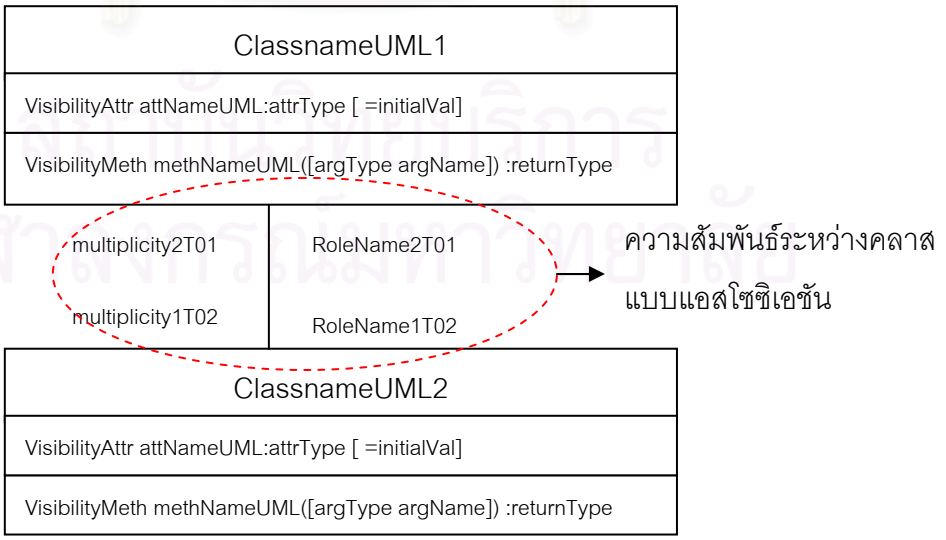
3. ไวยากรณ์ของการสร้างสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุตามมาตรฐานของฐานข้อมูลแมทิส

```
<METHOD_NAME> ::= mt_method"CREATE METHOD  METHODNAME_NAME
                  ([<ARGUMENT-LIST>])
                   RETURNS <TYPE> FOR  CLASSNAME_NAME
                   BEGIN
                     DECLARE  METHODNAME_NAME  <TYPE>;
                     RETURN   METHODNAME_NAME;
                   END;";

<ARGUMENT-LIST> ::= <TYPE> ARGUMENTNAME_NAME[{,<TYPE> ARGUMENTNAME_NAME}]

<TYPE>::= <PRIMITIVE>|<NON-PRIMITIVE>

<PRIMITIVE>::= String | Date | Integer | Char | Short

<NON-PRIMITIVE>::= <STRUCT>|<COLLECTION>|<CLASS_NAME>

<STRUCT>  ::= STRUCTNAME

<COLLECTION> ::= <COLLECTION-TYPE>"<" <PRIMITIVE> ">" |
                 <COLLECTION-TYPE>"<" <NON-PRIMITIVE> ">"

<COLLECTION-TYPE> ::= set | bag| list

<CLASS_NAME >::=CLASSNAME_NAME

<ATTRIBUTE_NAME> ::= attribute  <TYPE>  ATTRIBUTENAME_NAME;
```

รูปที่ 4.12 แสดงไวยากรณ์เบื้องต้นของการสร้างแอทริบิวต์และเมทธอดตามมาตรฐานของฐานข้อมูลแมทิส

4.3 กฎข้อที่ 3 กฎการแปลงความสัมพันธ์ระหว่างคลาสแบบแอสโซซิเอชันเป็นสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุ

รูปแบบของคลาสไดอะแกรม



รูปที่ 4.13 แสดงคลาสไดอะแกรมที่มีลักษณะเป็นแบบแอสโซซิเอชัน

จากรูปที่ 4.13 คลาสไดอะแกรมประกอบด้วยส่วนประกอบดังนี้

1 คลาส ClassnameUML1และคลาส ClassnameUML2 สัมพันธ์กันแบบแอสโซซิเอชัน

2 เส้นความสัมพันธ์ระหว่างคลาส 2 คลาสจะระบุบทบาท(Roles) และ มัลติพลิซิตี (multiplicity) จากรูปที่ 4.13 เส้นความสัมพันธ์มีบทบาทของแต่ละคลาสคือ RoleName1T02 จะเป็นบทบาทของคลาส ClassnameUML1 ที่มีต่อคลาส ClassnameUML2 อยู่ฝั่งคลาส ClassnameUML2 และ RoleName2T01 จะเป็นบทบาทของคลาส ClassnameUML2 ที่มีต่อคลาส ClassnameUML1 อยู่ฝั่งคลาส ClassnameUML1 และประกอบไปด้วย มัลติพลิซิตี ที่อยู่ด้านคลาส ClassnameUML1 เป็น multiplicity2T01 และ มัลติพลิซิตี ที่อยู่ด้านคลาส ClassnameUML2 เป็น multiplicity1T02

ขั้นตอนการแปลงคลาสไดอะแกรมให้อยู่ในรูปแบบของสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุ

ความสัมพันธ์ระหว่างคลาสแบบแอสโซซิเอชันจะถูกสร้าง ดังนี้

1. สร้างความสัมพันธ์ของคลาส ClassnameUML1 และคลาส ClassnameUML2 ซึ่งในสคีมาตัวกลางสำหรับสร้างเป็น สคีมาฐานข้อมูลเชิงวัตถุ คือ **<ASSOCIATION-CLASS1>** และ **<ASSOCIATION-CLASS2>** โดยการสร้างความสัมพันธ์ของคลาส ClassnameUML1 ให้เพิ่มคำที่ระบุความสัมพันธ์ระหว่างคลาส เช่น *relationship* เป็นต้น ระบุชื่อคลาสที่สัมพันธ์กับคลาส ClassnameUML1 ชื่อบทบาทของคลาส ClassnameUML1 ที่มีต่อคลาส ClassnameUML2 ซึ่งในที่นี้คือ RoleName1T02 จากนั้นระบุความสัมพันธ์ย้อนกลับของคลาส ClassnameUML1 โดยการระบุคำที่ระบุถึงความสัมพันธ์แบบย้อนกลับ ตามด้วย ชื่อคลาส ClassnameUML2 และชื่อบทบาทย้อนกลับของคลาส ClassnameUML2 ในที่นี้ชื่อบทบาทของคลาส ClassnameUML2 ที่มีต่อคลาส ClassnameUML1ซึ่งในที่นี้คือ RoleName2T01 และตามด้วยรายละเอียดเพิ่มเติมต่างๆ ตามแต่ละประเภทของฐานข้อมูล

2. สร้างความสัมพันธ์ของคลาส ClassnameUML2 ในลักษณะเดียวกับการสร้างความสัมพันธ์ของคลาส ClassnameUML1

สรุปการแปลงคลาสทั้ง 2 คลาส มีลักษณะดังนี้

ที่คลาส ClassnameUML1

- RoleName1T02 ในคลาสไดอะแกรมจะถูกแปลงเป็น บทบาทของคลาสในสคีมาตัวกลางสำหรับสร้างเป็น สคีมาฐานข้อมูลเชิงวัตถุ ($\underline{RELATIONSHIPNAME_{CLASS1\text{-}2}}$)

- RoleName2T01 ในคลาสไดอะแกรมจะถูกแปลงเป็นบทบาทแบบย้อนกลับของคลาสในสคีมาตัวกลางสำหรับสร้างเป็น สคีมาฐานข้อมูลเชิงวัตถุ($\underline{RELATIONSHIPNAME_{CLASS2\text{-}1}}$)

- ClassnameUML2 ในคลาสไดอะแกรมจะถูกแปลงเป็น ชื่อคลาสของความสัมพันธ์แบบย้อนกลับในสคีมาตัวกลางสำหรับสร้างเป็น สคีมาฐานข้อมูลเชิงวัตถุ(*CLASSNAME*$_{CLASS2-1}$)

- multiplicity1T02 ในคลาสไดอะแกรมจะถูกแปลงเป็น มัลติพลิซิตีของคลาส ในสคีมาตัวกลางสำหรับสร้างเป็น สคีมาฐานข้อมูลเชิงวัตถุ(*MULTIPLICITY*$_{CLASS1-2}$)

- multiplicity2T01 ในคลาสไดอะแกรมจะถูกแปลงเป็น มัลติพลิซิตีของคลาสที่สัมพันธ์กับคลาสในสคีมาตัวกลางสำหรับสร้างเป็น สคีมาฐานข้อมูลเชิงวัตถุ(*MULTIPLICITY*$_{CLASS2-1}$)

ที่คลาส ClassnameUML2

- RoleName2T01 ในคลาสไดอะแกรมจะถูกแปลงเป็น บทบาทของคลาสในสคีมาตัวกลางสำหรับสร้างเป็น สคีมาฐานข้อมูลเชิงวัตถุ (*RELATIONSHIPNAME*$_{CLASS2-1}$)

- RoleName1T02 ในคลาสไดอะแกรมจะถูกแปลงเป็นบทบาทแบบย้อนกลับของคลาสในสคีมาตัวกลางสำหรับสร้างเป็น สคีมาฐานข้อมูลเชิงวัตถุ(*RELATIONSHIPNAME*$_{CLASS1-2}$)

- ClassnameUML1 ในคลาสไดอะแกรมจะถูกแปลงเป็น ชื่อคลาสของความสัมพันธ์แบบย้อนกลับในสคีมาตัวกลางสำหรับสร้างเป็น สคีมาฐานข้อมูลเชิงวัตถุ (*CLASSNAME*$_{CLASS1-2}$)

- multiplicity2T01 ในคลาสไดอะแกรมจะถูกแปลงเป็น มัลติพลิซิตีของคลาส ในสคีมาตัวกลางสำหรับสร้างเป็น สคีมาฐานข้อมูลเชิงวัตถุ(*MULTIPLICITY*$_{CLASS2-1}$)

- multiplicity1T02 ในคลาสไดอะแกรมจะถูกแปลงเป็น มัลติพลิซิตีของคลาสที่สัมพันธ์กับคลาสในสคีมาตัวกลางสำหรับสร้างเป็น สคีมาฐานข้อมูลเชิงวัตถุ(*MULTIPLICITY*$_{CLASS1-2}$)

ไวยากรณ์การสร้างสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุ

1. ไวยากรณ์ของการสร้างสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุตามมาตรฐานของโอดีเอ็มจี

<ASSOCIATION> ::= **<ASSOCIATION-CLASS1><ASSOCIATION-CLASS2>**

<ASSOCIATION-CLASS1>::=*CLASSNAME*$_{CLASS1-2}$*:*
<RELATIONSHIP-LIST$_{CLASS2-1}$**>** *RELATIONSHIPNAME*$_{CLASS1-2}$
inverse *CLASSNAME*$_{CLASS2-1}$*::* *RELATIONSHIPNAME*$_{CLASS2-1}$ *;*

<ASSOCIATION-CLASS2 >::=*CLASSNAME*$_{CLASS2-1}$*:*
<RELATIONSHIP-LIST$_{CLASS1-2}$**>** *RELATIONSHIPNAME*$_{CLASS2-1}$
inverse *CLASSNAME*$_{CLASS1-2}$*::* *RELATIONSHIPNAME*$_{CLASS1-2}$*;*

รูปที่ 4.14 ไวยากรณ์ของการแปลงความสัมพันธ์ระหว่างคลาสแบบแอสโซซิเอชันตามมาตรฐานของโอดีเอ็มจี

2. ไวยากรณ์ของการสร้างสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุของฐานข้อมูลคาเช่

```
<ASSOCIATION> ::= <ASSOCIATION-CLASS1><ASSOCIATION-CLASS2>

<ASSOCIATION-CLASS1>::=CLASSNAME_CLASS1-2:
      <RELATIONSHIP-LIST_CLASS2-1> RELATIONSHIPNAME_CLASS1-2 as CLASSNAME_CLASS2-1
      [Cardinality= MULTIPLICITY_CLASS1-2 ,inverse RELATIONSHIPNAME_CLASS2-1] ;

<ASSOCIATION-CLASS2 >::=CLASSNAME_CLASS2-1:
      <RELATIONSHIP-LIST_CLASS1-2> RELATIONSHIPNAME_CLASS2-1 as CLASSNAME_CLASS1-2
      [Cardinality= MULTIPLICITY_CLASS2-1 ,inverse RELATIONSHIPNAME_CLASS1-2] ;
```

รูปที่ 4.15 ไวยากรณ์ของการแปลงความสัมพันธ์ระหว่างคลาสแบบแอสโซซิเอชันตามมาตรฐานของฐานข้อมูลเชิงวัตถุคาเช่

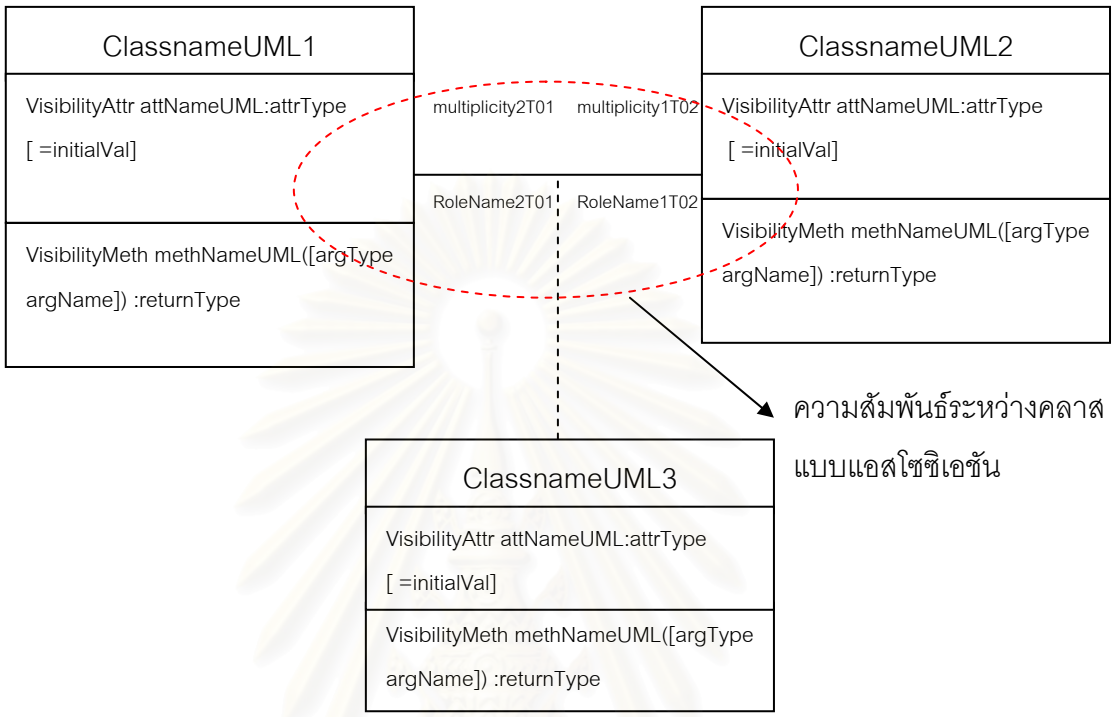
3. ไวยากรณ์ของการสร้างสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุของฐานข้อมูลแมทิส

```
<ASSOCIATION> ::= <ASSOCIATION-CLASS1><ASSOCIATION-CLASS2>

<ASSOCIATION-CLASS1>::=CLASSNAME_CLASS1-2:
    <RELATIONSHIP-LIST_CLASS2-1> RELATIONSHIPNAME_CLASS1-2[MULTIPLICITY_CLASS1-2]
    inverse CLASSNAME_CLASS2-1:: RELATIONSHIPNAME_CLASS2-1 ;

<ASSOCIATION-CLASS2 >::=CLASSNAME_CLASS2-1:
    <RELATIONSHIP-LIST_CLASS1-2> RELATIONSHIPNAME_CLASS2-1[MULTIPLICITY_CLASS2-1]
    inverse CLASSNAME_CLASS1-2:: RELATIONSHIPNAME_CLASS1-2;
```

รูปที่ 4.16 ไวยากรณ์ของการแปลงความสัมพันธ์ระหว่างคลาสแบบแอสโซซิเอชันตามมาตรฐานของฐานข้อมูลเชิงวัตถุแมทิส

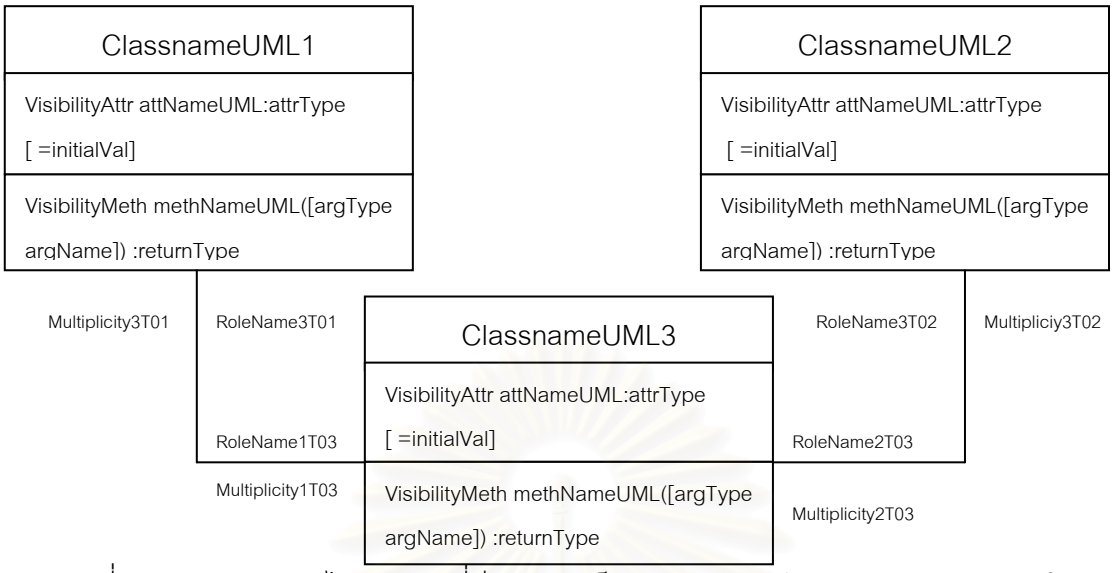
4.4 กฎข้อที่ 4 กฎการแปลงความสัมพันธ์ระหว่างคลาสแบบแอสโซซิเอชันคลาสเป็นสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุ

รูปแบบของคลาสไดอะแกรม



รูปที่ 4.17 แสดงคลาสไดอะแกรมที่มีความสัมพันธ์ระหว่างคลาสแบบแอสโซซิเอชันคลาส

ความสัมพันธ์ระหว่างคลาสที่มีลักษณะเป็นแอสโซซิเอชันคลาส คือความสัมพันธ์ระหว่างคลาสสองคลาสที่มีอีกคลาสเชื่อมความสัมพันธ์ระหว่าง 2 คลาส เมื่อมีความสัมพันธ์โดยคลาสที่มาเชื่อมความสัมพันธ์จะมีแอทริบิวต์และเมทธอดเป็นของตัวเองดังรูปที่ 4.17 โดยในการสร้างความสัมพันธ์แบบแอสโซซิเอชันคลาสจะแยกความสัมพันธ์ใหม่ ดังนี้



รูปที่ 4.18 แสดงคลาสไดอะแกรมที่มีลักษณะเป็นความสัมพันธ์ระหว่างคลาสแบบแอสโซซิเอชันคลาสที่แยกความสัมพันธ์ออกเป็น 2 ชุดความสัมพันธ์

จากรูปที่ 4.18 คลาสไดอะแกรมประกอบด้วยส่วนประกอบดังนี้

1. คลาส ClassnameUML1 และคลาส ClassnameUML2

2. คลาส ClassnameUML3 ที่ถูกสร้างขึ้นมาเป็นตัวเชื่อมความสัมพันธ์ระหว่างคลาส ClassnameUML1 และคลาส ClassnameUML2 มีความสัมพันธ์กันโดยมีแอทริบิวต์เข้ามาเกี่ยวข้องด้วย

3. เส้นความสัมพันธ์ที่เชื่อมโยงความสัมพันธ์ระหว่างคลาส 2 คลาส จะระบุบทบาทของคลาส และ มัลติพลิซิตี จากรูป

มัลติพลิซิตีประกอบไปด้วยมัลติพลิซิตีของคลาส ClassnameUML1 มีลักษณะเป็น Multiplicity1T03 และ มัลติพลิซิตีของคลาส ClassnameUML2 มีลักษณะเป็น Multiplicity2T03

ขั้นตอนการแปลงคลาสไดอะแกรมให้อยู่ในรูปแบบของสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุ

ความสัมพันธ์ระหว่างคลาสแบบแอสโซซิเอชันคลาสจะถูกสร้าง ดังนี้

1. สร้างความสัมพันธ์ระหว่างคลาส ClassnameUML1 และคลาส ClassnameUML3 และสร้างความสัมพันธ์ระหว่างคลาส ClassnameUML2 และคลาส ClassnameUML3

โดยการสร้างความสัมพันธ์ของคลาส ClassnameUML1 ให้ เพิ่มคำที่ระบุความสัมพันธ์ระหว่างคลาส ระบุชื่อคลาสที่สัมพันธ์กับคลาส ClassnameUML1 โดยกำหนดให้เป็นเซ็ตของคลาส ระบุชื่อบทบาทของคลาส ClassnameUML1 ที่มีต่อคลาส ClassnameUML3 ซึ่งในที่นี้คือ RoleName1T03 จากนั้นระบุความสัมพันธ์ย้อนกลับของคลาส ClassnameUML1 โดยการระบุคำที่ระบุถึงความสัมพันธ์แบบย้อนกลับ ตามด้วย ชื่อคลาส ClassnameUML3 และ

ชื่อบทบาทย้อนกลับของคลาส ClassnameUML3 ในที่นี้ชื่อบทบาทของคลาส ClassnameUML3 ที่มีต่อคลาส ClassnameUML1 ซึ่งในที่นี้คือ RoleName3T01 และตามด้วยรายละเอียดเพิ่มเติมต่าง ๆ ตามแต่ละประเภทของฐานข้อมูล

2. สร้างความสัมพันธ์ของคลาส ClassnameUML2 และ ClassnameUML3 ในลักษณะเดียวกับการสร้างความสัมพันธ์ของคลาส ClassnameUML1

ที่คลาส ClassnameUML2 และ ClassnameUML1

- RoleName1T03 ในคลาสไดอะแกรมจะถูกแปลงเป็นบทบาทของคลาสในสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุ($RELATIONSHIPNAME_{CLASS1-3}$)
- RoleName2T03 ในคลาสไดอะแกรมจะถูกแปลงเป็นบทบาทของคลาสในสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุ($RELATIONSHIPNAME_{CLASS2-3}$)
- RoleName3T01 และ RoleName3T02 ในคลาสไดอะแกรมจะถูกแปลงเป็นบทบาทแบบย้อนกลับของคลาสในสคีมาตัวกลางสำหรับสร้างเป็น สคีมาฐานข้อมูลเชิงวัตถุ ($RELATIONSHIPNAME_{CLASS3-1}$ และ $RELATIONSHIPNAME_{CLASS3-2}$)
- ClassnameUML3 ในคลาสไดอะแกรมจะถูกแปลงเป็น ชื่อคลาสของความสัมพันธ์แบบย้อนกลับในสคีมาตัวกลางสำหรับสร้างเป็น สคีมาฐานข้อมูลเชิงวัตถุ($CLASSNAME_{CLASS3-1}$ และ $CLASSNAME_{CLASS3-2}$)
- multiplicity1T03 และ multiplicity2T03 ในคลาสไดอะแกรมจะถูกแปลงเป็น มัลติพลิซิตีของคลาส ในสคีมาตัวกลางสำหรับสร้างเป็น สคีมาฐานข้อมูลเชิงวัตถุ($MULTIPLICITY_{CLASS1-3}$ และ $MULTIPLICITY_{CLASS2-3}$)
- multiplicity3T01 และ multiplicity3T02 ในคลาสไดอะแกรมจะถูกแปลงเป็น มัลติพลิซิตีของคลาสที่สัมพันธ์กับคลาสในสคีมาตัวกลางสำหรับสร้างเป็น สคีมาฐานข้อมูลเชิงวัตถุ ($MULTIPLICITY_{CLASS3-1}$ และ $MULTIPLICITY_{CLASS3-2}$)

ที่คลาส ClassnameUML3

- RoleName3T01 และ RoleName3T02 ในคลาสไดอะแกรมจะถูกแปลงเป็น บทบาทของคลาสในสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุ($RELATIONSHIPNAME_{CLASS3-1}$ $RELATIONSHIPNAME_{CLASS3-2}$)
- RoleName1T03 และ RoleName2T03 ในคลาสไดอะแกรมจะถูกแปลงเป็น บทบาทแบบย้อนกลับของคลาสในสคีมาตัวกลางสำหรับสร้างเป็น สคีมาฐานข้อมูลเชิงวัตถุ ($RELATIONSHIPNAME_{CLASS1-3}$ และ $RELATIONSHIPNAME_{CLASS2-3}$)

- ClassnameUML1 และ ClassnameUML2 ในคลาสไดอะแกรมจะถูกแปลงเป็น ชื่อคลาสของความสัมพันธ์แบบย้อนกลับในสคีมาตัวกลางสำหรับสร้างเป็น สคีมาฐานข้อมูลเชิงวัตถุ ($CLASSNAME_{CLASS1-3}$และ $CLASSNAME_{CLASS2-3}$)

- multiplicity3T01 และ multiplicity3T02 ในคลาสไดอะแกรมจะถูกแปลงเป็น มัลติพลิซิตีของคลาส ในสคีมาตัวกลางสำหรับสร้างเป็น สคีมาฐานข้อมูลเชิงวัตถุ($MULTIPLICITY_{CLASS3-1}$ $MULTIPLICITY_{CLASS3-2}$)

- multiplicity1T03 และ multiplicity2T03 ในคลาสไดอะแกรมจะถูกแปลงเป็นมัลติพลิซิตีของคลาสที่สัมพันธ์กับคลาสในสคีมาตัวกลางสำหรับสร้างเป็น สคีมาฐานข้อมูลเชิงวัตถุ ($MULTIPLICITY_{CLASS1-3}$ และ $MULTIPLICITY_{CLASS2-3}$)

ไวยากรณ์การสร้างสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุ

1. ไวยากรณ์ของการสร้างสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุตามมาตรฐานของโอดีเอ็มจี

```
<ASSOCIATIONCLASS>   ::= <ASSOCIATIONCLASS-CLASS1>
                         <ASSOCIATIONCLASS-CLASS2>
                         <ASSOCIATIONCLASS-CLASS3>

<ASSOCIATIONCLASS-CLASS1>::=CLASSNAME_CLASS1-3:
                    <RELATIONSHIP-LIST_CLASS3-1> RELATIONSHIPNAME_CLASS1-3
                    inverse  CLASSNAME_CLASS3-1:: RELATIONSHIPNAME_CLASS3-1;

<ASSOCIATIONCLASS-CLASS2>::= CLASSNAME_CLASS2-3:
                    <RELATIONSHIP-LIST_CLASS3-2> RELATIONSHIPNAME_CLASS2-3
                    inverse  CLASSNAME_CLASS3-2:: RELATIONSHIPNAME_CLASS3-2;

<ASSOCIATIONCLASS-CLASS3>::= CLASSNAME_CLASS3-1:
                    <RELATIONSHIP-LIST_CLASS1-3> RELATIONSHIPNAME_CLASS3-1
                    inverse  CLASSNAME_CLASS1-3:: RELATIONSHIPNAME_CLASS1-3;

                    <RELATIONSHIP-LIST_CLASS2-3> RELATIONSHIPNAME_CLASS3-2
                    inverse  CLASSNAME_CLASS2-3:: RELATIONSHIPNAME_CLASS2-3;
```

รูปที่ 4.19 ไวยากรณ์ของการสร้างความสัมพันธ์ระหว่างคลาสแบบแอสโซซิเอชันคลาสตามมาตรฐานของโอดีเอ็มจี

2. ไวยากรณ์ของการสร้างสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุของฐานข้อมูลคาเซ่

```
<ASSOCIATIONCLASS>  ::= <ASSOCIATIONCLASS-CLASS1>
                        <ASSOCIATIONCLASS-CLASS2>
                        <ASSOCIATIONCLASS-CLASS3>

<ASSOCIATIONCLASS-CLASS1>::=CLASSNAME_CLASS1-3:
      <RELATIONSHIP-LIST CLASS3-1>  RELATIONSHIPNAME_CLASS1-3 as  CLASSNAME_CLASS3-1
      [Cardinality= MULTIPLICITY_CLASS1-3  ,inverse  RELATIONSHIPNAME_CLASS3-1] ;

<ASSOCIATIONCLASS-CLASS2>::=CLASSNAME_CLASS2-3:
      <RELATIONSHIP-LIST CLASS3-2>  RELATIONSHIPNAME_CLASS2-3 as  CLASSNAME_CLASS3-2
      [Cardinality= MULTIPLICITY_CLASS2-3  ,inverse  RELATIONSHIPNAME_CLASS3-2] ;

<ASSOCIATIONCLASS-CLASS3>::=CLASSNAME_CLASS3-1:
      <RELATIONSHIP-LIST CLASS1-3>  RELATIONSHIPNAME_CLASS3-1 as  CLASSNAME_CLASS1-3
      [Cardinality= MULTIPLICITY_CLASS3-1  ,inverse  RELATIONSHIPNAME_CLASS1-3];

      <RELATIONSHIP-LIST CLASS2-3>  RELATIONSHIPNAME_CLASS3-2 as  CLASSNAME_CLASS2-3
      [Cardinality= MULTIPLICITY_CLASS3-2  ,inverse  RELATIONSHIPNAME_CLASS2-3];
```

รูปที่ 4.20 ไวยากรณ์ของการสร้างความสัมพันธ์แบบแอสโซซิเอชันคลาสของฐานข้อมูลคาเซ่

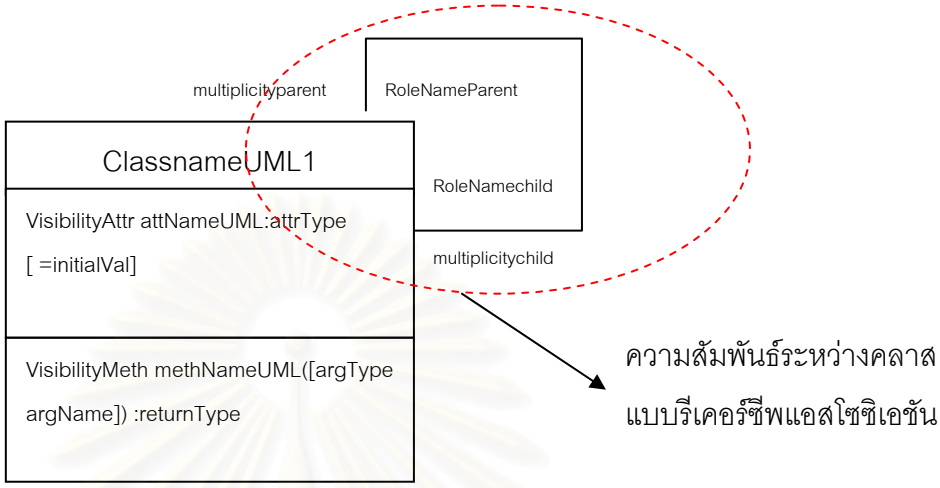
3. ไวยากรณ์ของการสร้างสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุของฐานข้อมูลแมทิส

```
<ASSOCIATIONCLASS>  ::= <ASSOCIATIONCLASS-CLASS1>
                        <ASSOCIATIONCLASS-CLASS2>
                        <ASSOCIATIONCLASS-CLASS3>

<ASSOCIATIONCLASS-CLASS1>::=CLASSNAME_CLASS1-3:
      <RELATIONSHIP-LIST_CLASS3-1>  RELATIONSHIPNAME_CLASS1-3[MULTIPLICITY_CLASS1-3]
      inverse  CLASSNAME_CLASS3-1:: RELATIONSHIPNAME_CLASS3-1;

<ASSOCIATIONCLASS-CLASS2>::=CLASSNAME_CLASS2-3:
       <RELATIONSHIP-LIST_CLASS3-2>  RELATIONSHIPNAME_CLASS2-3[MULTIPLICITY_CLASS2-3]
        inverse  CLASSNAME_CLASS3-2:: RELATIONSHIPNAME_CLASS3-2;

<ASSOCIATIONCLASS-CLASS3>::=CLASSNAME_CLASS3-1:
        <RELATIONSHIP-LIST_CLASS1-3>  RELATIONSHIPNAME_CLASS3-1[MULTIPLICITY_CLASS3-1]
          inverse  CLASSNAME_CLASS1-3:: RELATIONSHIPNAME_CLASS1-3;

        <RELATIONSHIP-LIST_CLASS2-3>  RELATIONSHIPNAME_CLASS3-2[MULTIPLICITY_CLASS3-2]
         inverse  CLASSNAME_CLASS2-3:: RELATIONSHIPNAME_CLASS2-3;
```

รูปที่ 4.21 ไวยากรณ์ของการสร้างความสัมพันธ์แบบแอสโซซิเอชันคลาสของฐานข้อมูลแมทิส

4.5 กฎข้อที่ 5 กฎการแปลงความสัมพันธ์ระหว่างคลาสแบบรีเคอร์ซีพแอสโซซิเอชันเป็นสคีมาฐาน ข้อมูลเชิงวัตถุ

รูปแบบของคลาสไดอะแกรม



รูปที่ 4.22 แสดงคลาสไดอะแกรมที่มีสัมพันธ์ระหว่างคลาสแบบรีเคอร์ซีพแอสโซซิเอชัน

ความสัมพันธ์ระหว่างคลาสที่มีลักษณะแบบรีเคอร์ซีพแอสโซซิเอชันคือ ความสัมพันธ์ระหว่างคลาสที่มีความสัมพันธ์กับตัวคลาสนั้น ๆ เองโดยมีลักษณะความสัมพันธ์แบบแม่กับลูก โดยประกอบไปด้วยส่วนประกอบ ดังนี้

1. คลาส ClassnameUML1

2. เส้นความสัมพันธ์ที่เชื่อมโยงความสัมพันธ์ระหว่างคลาส จะระบุบทบาทของคลาส และ มัลติพลิซิตี ประกอบไปด้วย มัลติพลิซิตี ของคลาส ClassnameUML1 ที่อยู่ตรงส่วน parent เป็น multiplicityparent และ มัลติพลิซิตี ของคลาส ClassnameUML1 ที่อยู่ตรงส่วน child จะมีลักษณะเป็น multiplicitychild โดยมีชื่อของบทบาทของคลาสที่อยู่ตรงส่วน child มีชื่อว่า RoleNamechild และมีชื่อของบทบาทของคลาสที่อยู่ตรงส่วน parent มีชื่อว่า RoleNameParent

ขั้นตอนการแปลงคลาสไดอะแกรมให้อยู่ในรูปแบบของสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุ

ความสัมพันธ์ระหว่างคลาสแบบรีเคอร์ซีพแอสโซซิเอชันจะถูกสร้าง ดังนี้

1. สร้างความสัมพันธ์ของคลาส ClassnameUML1 ในฝั่ง child ให้เพิ่มคำที่ระบุความสัมพันธ์ระหว่างคลาส ระบุชื่อคลาสที่สัมพันธ์กับคลาส ClassnameUML1 โดยระบุเป็นเซ็ต ระบุชื่อบทบาทฝั่ง child ซึ่งในที่นี้คือ Rolenamechild จากนั้นระบุความสัมพันธ์ย้อนกลับของคลาส ClassnameUML1 โดยการระบุคำที่ระบุถึงความสัมพันธ์แบบย้อนกลับ ตามด้วย ชื่อคลาส ClassnameUML1 และชื่อบทบาทย้อนกลับของคลาส ClassnameUML1ในที่นี้ชื่อบทบาทฝั่ง

parent ซึ่งในที่นี้คือ Rolenameparent และตามด้วยรายละเอียดเพิ่มเติมต่าง ๆตามแต่ละประเภทของฐานข้อมูล

2. สร้างความสัมพันธ์ของคลาส ClassnameUML1 ในฝั่ง parent ในลักษณะเดียวกับการสร้างความสัมพันธ์ของคลาส ClassnameUML1 ในฝั่ง child โดยการแปลงคลาส มีลักษณะดังนี้

ที่คลาส ClassnameUML1 ในฝั่ง parent

- RoleNameChild ในคลาสไดอะแกรมจะถูกแปลงเป็น บทบาทของคลาสในสคีมาตัวกลางสำหรับสร้างเป็น สคีมาฐานข้อมูลเชิงวัตถุ ($RELATIONSHIPNAME_{CLASS1(CHILD)}$)

- RoleNameParent ในคลาสไดอะแกรมจะถูกแปลงเป็นบทบาทแบบย้อนกลับของคลาสในสคีมาตัวกลางสำหรับสร้างเป็น สคีมาฐานข้อมูลเชิงวัตถุ($RELATIONSHIPNAME_{CLASS1(PARENT)}$)

- ClassnameUML1 ในคลาสไดอะแกรมจะถูกแปลงเป็น ชื่อคลาสของความสัมพันธ์แบบย้อนกลับในสคีมาตัวกลางสำหรับสร้างเป็น สคีมาฐานข้อมูลเชิงวัตถุ ($CLASSNAME_{CLASS1}$)

- multiplicitychild ในคลาสไดอะแกรมจะถูกแปลงเป็น มัลติพลิซิตีของคลาส ในสคีมาตัวกลางสำหรับสร้างเป็น สคีมาฐานข้อมูลเชิงวัตถุ($MULTIPLICITY_{CLASS1}$)

- multiplicityparent ในคลาสไดอะแกรมจะถูกแปลงเป็น มัลติพลิซิตีของคลาสที่สัมพันธ์กับคลาสในสคีมาตัวกลางสำหรับสร้างเป็น สคีมาฐานข้อมูลเชิงวัตถุ($MULTIPLICITY_{CLASS1}$)

ที่คลาส ClassnameUML1 ในฝั่ง child

- RoleNameParent ในคลาสไดอะแกรมจะถูกแปลงเป็น บทบาทของคลาสในสคีมาตัวกลางสำหรับสร้างเป็น สคีมาฐานข้อมูลเชิงวัตถุ($RELATIONSHIPNAME_{CLASS1(PARENT)}$)

- RoleNameChild ในคลาสไดอะแกรมจะถูกแปลงเป็นบทบาทแบบย้อนกลับของคลาสในสคีมาตัวกลางสำหรับสร้างเป็น สคีมาฐานข้อมูลเชิงวัตถุ($RELATIONSHIPNAME_{CLASS1(CHILD)}$)

- ClassnameUML1 ในคลาสไดอะแกรมจะถูกแปลงเป็น ชื่อคลาสของความสัมพันธ์แบบย้อนกลับในสคีมาตัวกลางสำหรับสร้างเป็น สคีมาฐานข้อมูลเชิงวัตถุ($CLASSNAME_{CLASS1}$)

- multiplicityparent ในคลาสไดอะแกรมจะถูกแปลงเป็น มัลติพลิซิตีของคลาส ในสคีมาตัวกลางสำหรับสร้างเป็น สคีมาฐานข้อมูลเชิงวัตถุ($MULTIPLICITY_{CLASS1}$)

- multiplicitychild ในคลาสไดอะแกรมจะถูกแปลงเป็น มัลติพลิซิตีของคลาสที่สัมพันธ์กับคลาสในสคีมาตัวกลางสำหรับสร้างเป็น สคีมาฐานข้อมูลเชิงวัตถุ($MULTIPLICITY_{CLASS1}$)

ไวยากรณ์การสร้างสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุ

1. ไวยากรณ์ของการสร้างสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุตามมาตรฐานของโอดีเอ็มจี

```
<RECURSIVEASSOCIATION> ::= CLASSNAME_CLASS1:
            <RELATIONSHIP-LIST_CLASS1> RELATIONSHIPNAME_CLASS1(PARENT)
                  inverse  CLASSNAME_CLASS1:: RELATIONSHIPNAME_CLASS1(CHILD) ;

            <RELATIONSHIP-LIST_CLASS1> RELATIONSHIPNAME_CLASS1(CHILD)
             inverse  CLASSNAME_CLASS1:: RELATIONSHIPNAME_CLASS1(PARENT) ;
```

รูปที่ 4.23 ไวยากรณ์เบื้องต้นของการสร้างความสัมพันธ์แบบรีเคอร์ซีพตามมาตรฐานของโอดีเอ็มจี

2. ไวยากรณ์ของการสร้างสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุของฐานข้อมูลคาเช่

```
<RECURSIVEASSOCIATION> ::= CLASSNAME_CLASS1:
      <RELATIONSHIP-LIST CLASS1> RELATIONSHIPNAME_CLASS1(PARENT) as CLASSNAME_CLASS1
      [Cardinality= MULTIPLICITY CLASS1,inverse RELATIONSHIPNAME_CLASS1(CHILD)] ;

      <RELATIONSHIP-LIST CLASS1> RELATIONSHIPNAME_CLASS1(CHILD) as CLASSNAME_CLASS1
      [Cardinality= MULTIPLICITY CLASS1 ,inverse RELATIONSHIPNAME_CLASS1(PARENT)];
```

รูปที่ 4.24 ไวยากรณ์เบื้องต้นของการสร้างความสัมพันธ์แบบรีเคอร์ซีพของฐานข้อมูลคาเช่

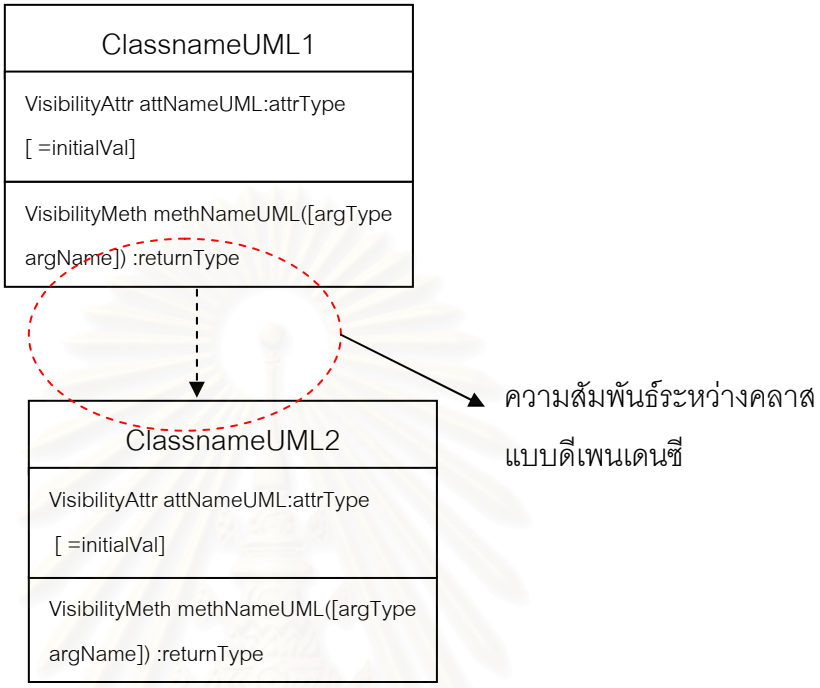
3. ไวยากรณ์ของการสร้างสคีมาตัวกลางสำหรับสร้างเป็น สคีมาฐานข้อมูลเชิงวัตถุของฐานข้อมูลแมทิส

```
<RECURSIVEASSOCIATION> ::=CLASSNAME_CLASS1:
   <RELATIONSHIP-LIST_CLASS1> RELATIONSHIPNAME_CLASS1(PARENT)[MULTIPLICITY_CLASS1(PARENT)]
         inverse  CLASSNAME_CLASS1:: RELATIONSHIPNAME_CLASS1(CHILD) ;

   <RELATIONSHIP-LIST_CLASS1> RELATIONSHIPNAME_CLASS1(CHILD)[MULTIPLICITY_CLASS1(CHILD)]
         inverse  CLASSNAME_CLASS1:: RELATIONSHIPNAME_CLASS1(PARENT) ;
```

รูปที่ 4.25 ไวยากรณ์เบื้องต้นของการสร้างความสัมพันธ์แบบรีเคอร์ซีพของฐานข้อมูลแมทิส

4.6 กฎข้อที่ 6 กฎการแปลงความสัมพันธ์ระหว่างคลาสแบบดีเพนเดนซีเป็นสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุ

รูปแบบของคลาสไดอะแกรม



รูปที่ 4.26 แสดงรูปแบบคลาสไดอะแกรมที่มีความสัมพันธ์แบบดีเพนเดนซี

ความสัมพันธ์ระหว่างคลาสที่เป็นความสัมพันธ์แบบดีเพนเดนซี คือ ความสัมพันธ์ที่แสดงถึงการขึ้นต่อกันระหว่างคลาสกับคลาสโดยหัวลูกศรจะชี้ไปที่คลาสที่เมื่อมีการเปลี่ยนแปลงแล้วจะส่งผลต่อคลาสที่อยู่ปลายลูกศรด้วย โดยที่แอทริบิวต์ของคลาสที่อยู่ตรงส่วนปลายลูกศรจะถูกสร้างมาจากแอทริบิวต์ของคลาสที่อยู่ตรงหัวลูกศร

จากรูปดังกล่าวจะประกอบไปด้วยส่วนประกอบต่าง ๆดังนี้คือ

1. คลาส ClassnameUML1 และคลาส ClassnameUML2ที่มีความสัมพันธ์กัน โดยแอทริบิวต์ของคลาส ClassnameUML1 ถูกสร้างมาจากแอทริบิวต์ที่อยู่คลาส ClassnameUML2

2. เส้นความสัมพันธ์ที่เชื่อมโยงความสัมพันธ์ระหว่างคลาส 2 คลาส

ขั้นตอนการแปลงคลาสไดอะแกรมให้อยู่ในรูปแบบของสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุ

ความสัมพันธ์ระหว่างคลาสแบบดีเพนเดนซีจะถูกสร้าง ดังนี้

1. ที่คลาส ClassnameUML2 แอทริบิวต์ในยูเอ็มแอลคลาสไดอะแกรมจะถูกแปลงเป็นแอทริบิวต์ในสคีมาตัวกลางสำหรับสร้างเป็น สคีมาฐานข้อมูลเชิงวัตถุ ตามกฎข้อที่ 2 ตามปกติ

2. ที่คลาส ClassnameUML1 แอทริบิวต์ในยูเอ็มแอลคลาสไดอะแกรมจะถูกแปลงเป็นแอทริบิวต์ในสคีมาตัวกลางสำหรับสร้างเป็น สคีมาฐานข้อมูลเชิงวัตถุ โดยการสร้างแอทริบิวต์ขึ้นมาเพิ่ม 1 แอทริบิวต์ โดยที่แอทริบิวต์ดังกล่าวมีชนิดข้อมูลเป็นคลาส ClassnameUML2 และระบุชื่อแอทริบิวต์ที่สร้างขึ้นมา

ไวยากรณ์การสร้างสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุ

1. ไวยากรณ์ของการสร้างสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุตามมาตรฐานของโอดีเอ็มจี

```
<DEPENDENCY>  ::= CLASSNAME_CLASS1-2:
                    attribute  <CLASS_CLASS2-1>  ATTRIBUTENAME_CLASS1-2;
```

รูปที่ 4.27 แสดงไวยากรณ์เบื้องต้นของการสร้างความสัมพันธ์แบบดีเพนเดนซีตามมาตรฐานของโอดีเอ็มจี

2. ไวยากรณ์ของการสร้างสคีมาตัวกลางสำหรับสร้างเป็น สคีมาฐานข้อมูลเชิงวัตถุของฐานข้อมูลคาเช่

```
<DEPENDENCY>  ::= CLASSNAME_CLASS1-2:
                    property  ATTRIBUTENAME_CLASS1-2  as % <CLASS_CLASS2-1> ;
```

รูปที่ 4.28 แสดงไวยากรณ์เบื้องต้นของการสร้างความสัมพันธ์แบบดีเพนเดนซีตามมาตรฐานของฐานข้อมูลคาเช่

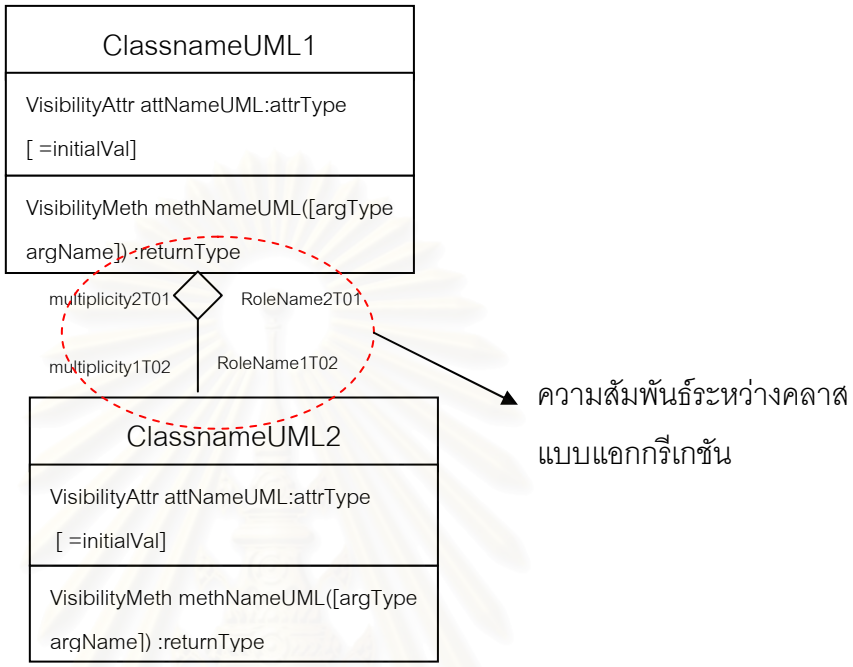
3. ไวยากรณ์ของการสร้างสคีมาตัวกลางสำหรับสร้างเป็น สคีมาฐานข้อมูลเชิงวัตถุตามมาตรฐานของฐานข้อมูลแมทิส

```
<DEPENDENCY>  ::= CLASSNAME_CLASS1-2:
                    attribute  <CLASS_CLASS2-1>  ATTRIBUTENAME_CLASS1-2;
```

รูปที่ 4.29 แสดงไวยากรณ์เบื้องต้นของการสร้างความสัมพันธ์แบบดีเพนเดนซีตามมาตรฐานของฐานข้อมูลแมทิส

4.7 กฎข้อที่ 7 กฎการแปลงความสัมพันธ์ระหว่างคลาสแบบแอกกรีเกชันเป็นสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุ

รูปแบบของคลาสไดอะแกรม



รูปที่ 4.30 แสดงรูปแบบคลาสไดอะแกรมที่มีความสัมพันธ์แบบแอกกรีเกชัน

ความสัมพันธ์ระหว่างคลาสที่เป็นความสัมพันธ์แบบแอกกรีเกชัน คือ ความสัมพันธ์แบบแอสโซซิเอชันแบบหนึ่งใช้แสดงความสัมพันธ์ในการเป็นส่วนหนึ่งของกันและกันระหว่างคลาส

จากรูปดังกล่าวจะประกอบไปด้วยส่วนประกอบต่าง ๆดังนี้คือ

1. คลาส ClassnameUML1 และคลาส ClassnameUML2

2. เส้นความสัมพันธ์ที่เชื่อมโยงความสัมพันธ์ระหว่างคลาส 2 คลาส จะระบุ บทบาท และ มัลติพลิซิตี และมีเครื่องหมายที่แสดงถึงการเป็นส่วนหนึ่ง(part) ของคลาส และส่วนรวมทั้งหมด (whole) ของคลาส โดยด้านหัวที่เป็นข้าวหลามตัดจะชี้ไปที่คลาสที่แสดงส่วนรวมทั้งหมด ด้านปลายของเส้นความสัมพันธ์จะชี้ไปที่คลาสที่เป็นส่วนหนึ่ง ของด้านหัว และ มัลติพลิซิตี จะประกอบไปด้วย มัลติพลิซิตี ของคลาส ClassnameUML2 ที่เป็นส่วนหนึ่ง จะมีลักษณะเป็น multiplicity1T02 และ มัลติพลิซิตี ของคลาส ClassnameUML1 ที่แสดงส่วนรวมทั้งหมด จะมีลักษณะเป็น multiplicity1T02

ขั้นตอนการแปลงคลาสไดอะแกรมให้อยู่ในรูปแบบของสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุ

ความสัมพันธ์ระหว่างคลาสแบบแอกกรีเกชันจะถูกสร้าง ดังนี้

1. ความสัมพันธ์ระหว่างคลาสของคลาส ClassnameUML1 และ ClassnameUML2 ในคลาสไดอะแกรมจะถูกแปลงเป็นสคีมาตัวกลางสำหรับสร้างเป็นของสคีมาตัวกลางสำหรับสร้างเป็น สคีมาฐานข้อมูลเชิงวัตถุในลักษณะเดียวกับกฎข้อที่ 3

2. ที่คลาส ClassnameUML1 แอทริบิวต์ในยูเอ็มแอลคลาสไดอะแกรมจะถูกแปลงเป็นแอทริบิวต์ในสคีมาตัวกลางสำหรับสร้างเป็น สคีมาฐานข้อมูลเชิงวัตถุ โดยการสร้างแอทริบิวต์ขึ้นมาเพิ่ม 1 แอทริบิวต์ โดยที่แอทริบิวต์ดังกล่าวมีชนิดข้อมูลเป็นคลาส ClassnameUML2 และระบุชื่อแอทริบิวต์ที่สร้างขึ้นมา

ไวยากรณ์การสร้างสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุ

1. ไวยากรณ์ของการสร้างสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุตามมาตรฐานของโอดีเอ็มจี

```
<AGGREGATION> ::= <AGGREGATION-CLASS1><AGGREGATION-CLASS2>

<AGGREGATION-CLASS1> ::= CLASSNAME_CLASS1-2:
                         <RELATIONSHIP-LIST_CLASS2-1>  RELATIONSHIPNAME_CLASS1-2
                         inverse  CLASSNAME_CLASS2-1:: RELATIONSHIPNAME_CLASS2-1;

                         attribute  <CLASS_CLASS2-1>  ATTRIBUTENAME_CLASS1-2;

<AGGREGATION-CLASS2> ::= CLASSNAME_CLASS2-1:
                         <RELATIONSHIP-LIST_CLASS1-2>  RELATIONSHIPNAME_CLASS2-1
                         inverse  CLASSNAME_CLASS1-2:: RELATIONSHIPNAME_CLASS1-2;
```

รูปที่ 4.31 แสดงไวยากรณ์เบื้องต้นของการสร้างความสัมพันธ์แบบแอกกรีเกชันตามมาตรฐานของโอดีเอ็มจี

2. ไวยากรณ์ของการสร้างสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุของฐานข้อมูลคาเช่

```
<AGGREGATION> ::= <AGGREGATION-CLASS1><AGGREGATION-CLASS2>

<AGGREGATION-CLASS1> ::= CLASSNAME_CLASS1-2:
      <RELATIONSHIP-LIST_CLASS2-1> RELATIONSHIPNAME_CLASS1-2 as CLASSNAME_CLASS2-1
      [Cardinality= MULTIPLICITY_CLASS1-2, inverse RELATIONSHIPNAME_CLASS2-1] ;

      property ATTRIBUTENAME_CLASS1-2 as % <CLASS_CLASS2-1> ;

<AGGREGATION-CLASS2> ::= CLASSNAME_CLASS2-1:
      <RELATIONSHIP-LIST_CLASS1-2> RELATIONSHIPNAME_CLASS2-1 as CLASSNAME_CLASS1-2
      [Cardinality= MULTIPLICITY_CLASS2-1, inverse RELATIONSHIPNAME_CLASS1-2] ;
```

รูปที่ 4.32 แสดงไวยากรณ์เบื้องต้นของการสร้างความสัมพันธ์แบบแอกกรีเกชันตามมาตรฐานของฐานข้อมูลคาเช่

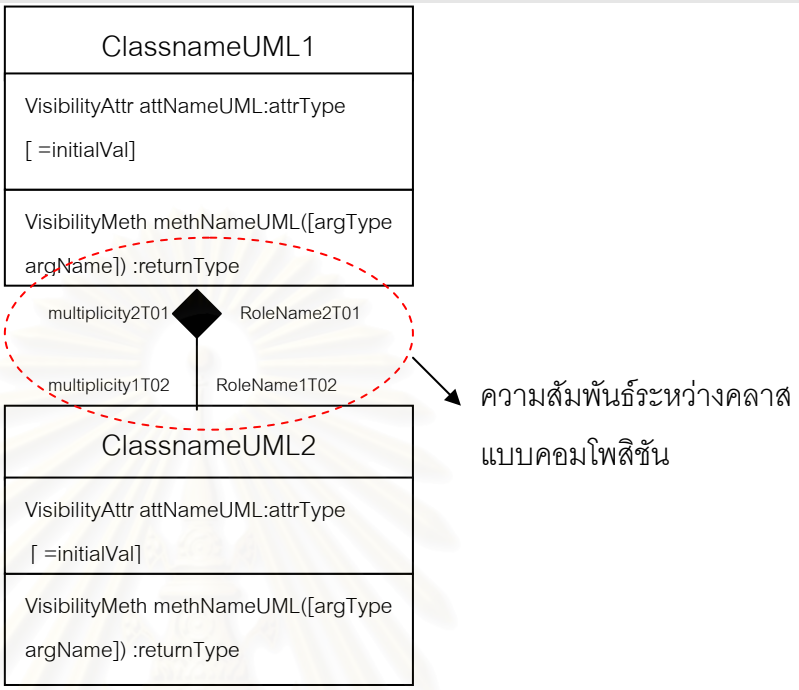
3. ไวยากรณ์ของการสร้างสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุตามมาตรฐานของฐานข้อมูลแมทิส

```
<AGGREGATION-CLASS1> ::= CLASSNAME_CLASS1-2:
      <RELATIONSHIP-LIST_CLASS2-1> RELATIONSHIPNAME_CLASS1-2[MULTIPLICITY_CLASS1-2]
      inverse CLASSNAME_CLASS2-1:: RELATIONSHIPNAME_CLASS2-1;

      attribute <CLASS_CLASS2-1> ATTRIBUTENAME_CLASS1-2;

<AGGREGATION-CLASS2> ::= CLASSNAME_CLASS2-1:
      <RELATIONSHIP-LIST_CLASS1-2> RELATIONSHIPNAME_CLASS2-1[MULTIPLICITY_CLASS2-1]
      inverse CLASSNAME_CLASS1-2:: RELATIONSHIPNAME_CLASS1-2;
```

รูปที่ 4.33 แสดงไวยากรณ์เบื้องต้นของการสร้างความสัมพันธ์แบบแอกกรีเกชันตามมาตรฐานของฐานข้อมูลแมทิส

4.8 กฎข้อที่ 8 กฎการแปลงความสัมพันธ์ระหว่างคลาสแบบคอมโพสิชันเป็นสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุ

รูปแบบของคลาสไดอะแกรม



รูปที่ 4.34 แสดงรูปแบบคลาสไดอะแกรมที่มีความสัมพันธ์แบบคอมโพสิชัน

ความสัมพันธ์ระหว่างคลาสที่เป็นความสัมพันธ์แบบคอมโพสิชันคือ ความสัมพันธ์แบบแอกกรีเกชันแบบหนึ่งใช้แสดงความสัมพันธ์ในการเป็นส่วนหนึ่งของกันและกันระหว่างคลาสกับคลาส โดยด้านหัวที่เป็นข้าวหลามตัดทึบจะชี้ไปที่คลาสที่แสดงส่วนรวมทั้งหมดด้านปลายของเส้นความสัมพันธ์จะชี้ไปที่คลาสที่เป็นส่วนหนึ่ง ของด้านหัว และคลาสที่เป็นส่วนหนึ่งนั้นจะอยู่หรือหายไปพร้อม ๆ กับคลาสที่เป็นส่วนรวมทั้งหมด

จากรูปดังกล่าวจะประกอบไปด้วยส่วนประกอบต่าง ๆดังนี้คือ

1. คลาส ClassnameUML1 และคลาส ClassnameUML2

2. เส้นความสัมพันธ์ที่เชื่อมโยงความสัมพันธ์ระหว่างคลาส 2 คลาสจะระบุและ มัลติพลิซิตี และจะมีเครื่องหมายที่แสดงถึงการเป็นส่วนหนึ่งของคลาสแต่ละคลาส และส่วนรวมทั้งหมดของคลาส โดยด้านหัวที่เป็นข้าวหลามตัดทึบจะชี้ไปที่คลาสที่แสดงส่วนรวมทั้งหมด ด้านปลายของเส้นความสัมพันธ์จะชี้ไปที่คลาสที่เป็นส่วนหนึ่ง ของด้านหัว และ มัลติพลิซิตี จะประกอบไปด้วยมัลติพลิซิตี ของคลาส ClassnameUML2 ที่เป็นส่วนหนึ่ง จะมีลักษณะเป็น multiplicity2T01 และมัลติพลิซิตี ของคลาส ClassnameUML1 ที่แสดงส่วนรวมทั้งหมด จะมีลักษณะเป็น multiplicity1T02

ขั้นตอนการแปลงคลาสไดอะแกรมให้อยู่ในรูปแบบของสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุ

ความสัมพันธ์ระหว่างคลาสแบบคอมโพสิชันคลาสจะถูกสร้าง ดังนี้

1. ความสัมพันธ์ระหว่างคลาสของคลาส ClassnameUML1 และ ClassnameUML2 ในคลาสไดอะแกรมจะถูกแปลงเป็นสคีมาตัวกลางสำหรับสร้างเป็น สคีมาฐานข้อมูลเชิงวัตถุในลักษณะเดียวกับกฎข้อที่ 3

2. ที่คลาส ClassnameUML1 แอทริบิวต์ในยูเอ็มแอลคลาสไดอะแกรมจะถูกแปลงเป็นแอทริบิวต์ในสคีมาตัวกลางสำหรับสร้างเป็น สคีมาฐานข้อมูลเชิงวัตถุ โดยการสร้างแอทริบิวต์ขึ้นมาเพิ่ม 1 แอทริบิวต์ โดยที่แอทริบิวต์ดังกล่าวมีชนิดข้อมูลเป็นคลาส ClassnameUML2 และระบุชื่อแอทริบิวต์ที่สร้างขึ้นมา

3. เพิ่มคีย์ของคลาส ClassnameUML2 ที่คลาส ClassnameUML1 เพื่อให้สามารถทำลายคลาส ClassnameUML2 เมื่อคลาส ClassnameUML1 ถูกทำลาย

ไวยากรณ์การสร้างสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุ

1. ไวยากรณ์ของการสร้างสคีมาตัวกลางสำหรับสร้างเป็น สคีมาฐานข้อมูลเชิงวัตถุตามมาตรฐานของโอดีเอ็มจี

```
<COMPOSITION> ::= <COMPOSITION-CLASS1> <COMPOSITION-CLASS2>

<COMPOSITION-CLASS1>::=CLASSNAME_CLASS1-2:
                       <RELATIONSHIP-LIST_CLASS2-1>   RELATIONSHIPNAME_CLASS1-2
                       inverse  CLASSNAME_CLASS2-1:: RELATIONSHIPNAME_CLASS2-1;

                       attribute <CLASS_CLASS2-1>  ATTRIBUTENAME_CLASS1-2;

                       attribute  <KEY-LIST_NAME>;

<COMPOSITION -CLASS2>::=CLASSNAME_CLASS2-1:
                       <RELATIONSHIP-LIST CLASS1-2>   RELATIONSHIPNAME_CLASS2-1
                       inverse  CLASSNAME_CLASS1-2:: RELATIONSHIPNAME_CLASS1-2;
```

รูปที่ 4.35 แสดงไวยากรณ์เบื้องต้นของการสร้างความสัมพันธ์แบบคอมโพซิชันตามมาตรฐานของโอดีเอ็มจี

2. ไวยากรณ์ของการสร้างสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุของฐานข้อมูลคาเช่

```
<COMPOSITION> ::= <COMPOSITION-CLASS1> <COMPOSITION-CLASS2>

<COMPOSITION-CLASS1>::=CLASSNAME_CLASS1-2:
      <RELATIONSHIP-LIST_CLASS2-1> RELATIONSHIPNAME_CLASS1-2 as CLASSNAME_CLASS2-1
      [Cardinality= MULTIPLICITY_CLASS1-2,inverse RELATIONSHIPNAME_CLASS2-1];

      property   ATTRIBUTENAME_CLASS1-2  as % <CLASS_CLASS2-1>  ;

      property  <KEY-LIST_NAME>;

<COMPOSITION -CLASS2>::=CLASSNAME_CLASS2-1:
      <RELATIONSHIP-LIST_CLASS1-2>   RELATIONSHIPNAME_CLASS2-1 as  CLASSNAME_CLASS1-2
[Cardinality= MULTIPLICITY_CLASS2-1,inverse  RELATIONSHIPNAME_CLASS1-2] ;
```

รูปที่ 4.36 แสดงไวยากรณ์เบื้องต้นของการสร้างความสัมพันธ์แบบคอมโพซิชันตามมาตรฐานของฐานข้อมูลคาเช่

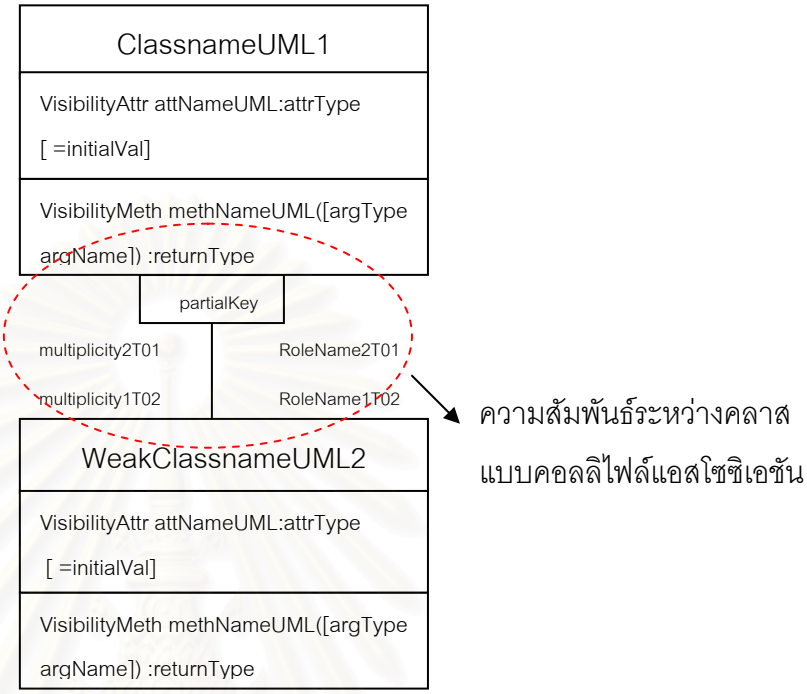
3. ไวยากรณ์ของการสร้างสคีมาตัวกลางสำหรับสร้างเป็น สคีมาฐานข้อมูลเชิงวัตถุตามมาตรฐานของฐานข้อมูลแมทิส

```
<COMPOSITION> ::= <COMPOSITION-CLASS1> <COMPOSITION-CLASS2>


<COMPOSITION-CLASS1>::=CLASSNAME_CLASS1-2:
     <RELATIONSHIP-LIST_CLASS2-1>   RELATIONSHIPNAME_CLASS1-2[MULTIPLICITY_CLASS1-2]
      inverse  CLASSNAME_CLASS2-1:: RELATIONSHIPNAME_CLASS2-1;

      attribute  <TYPE>  KEYOFCLASS_CLASS2-1;

      attribute <CLASS_CLASS2-1>  ATTRIBUTENAME _CLASS1-2;

<COMPOSITION -CLASS2>::=CLASSNAME_CLASS2-1:
     <RELATIONSHIP-LIST_CLASS1-2>   RELATIONSHIPNAME_CLASS2-1[MULTIPLICITY_CLASS2-1]
      inverse  CLASSNAME_CLASS1-2:: RELATIONSHIPNAME_CLASS1-2;
```

รูปที่ 4.37 แสดงไวยากรณ์เบื้องต้นของการสร้างความสัมพันธ์แบบคอมโพซิชันตามมาตรฐานของฐานข้อมูลแมทิส

4.9 กฎข้อที่ 9 กฎการแปลงความสัมพันธ์ระหว่างคลาสแบบคอลลิไฟล์แอสโซซิเอชัน เป็นสคีมาตัวกลางสำหรับสร้างเป็น สคีมาฐานข้อมูลเชิงวัตถุ

รูปแบบของคลาสไดอะแกรม



รูปที่ 4.38 แสดงคลาสไดอะแกรมที่มีความสัมพันธ์แบบคอลลิไฟล์แอสโซซิเอชัน

ความสัมพันธ์ระหว่างคลาสที่เป็นความสัมพันธ์แบบคอลลิไฟล์แอสโซซิเอชัน คือ ความสัมพันธ์ที่มีแอทริบิวต์หรือกลุ่มของแอทริบิวต์ที่ค่าของแอทริบิวต์ดังกล่าวจะใช้อ้างอิงการเชื่อมโยงกันระหว่างคลาส 2 คลาสที่สัมพันธ์กัน ใช้แสดงความสัมพันธ์ในการเป็นส่วนหนึ่งของกันและกันระหว่างคลาส จะมีการระบุถึงคีย์ที่คลาสวีคจะนำไปใช้ในการอ้างอิงภายในคลาสวีคนั้น ๆ ความสัมพันธ์ดังกล่าวจะรวมไปถึงความสัมพันธ์แบบคอลลิไฟล์คอมโพซิชัน และคอลลิไฟล์แอกกรีเกชัน

จากรูปดังกล่าวจะประกอบไปด้วยส่วนประกอบต่าง ๆดังนี้คือ

1. คลาส ClassnameUML1 และคลาส WeakClassnameUML2
2. คีย์ที่เป็นตัวระบุถึงคลาสที่เป็นส่วนทั้งหมด(patial key)
3. เส้นความสัมพันธ์ที่เชื่อมโยงความสัมพันธ์ระหว่างคลาส 2 คลาส จะระบุชื่อความสัมพันธ์และ มัลติพลิซิตี ประกอบไปด้วย มัลติพลิซิตี ของคลาส ClassnameUML1 ที่เป็นส่วนรวมทั้งหมด จะมีลักษณะเป็น multiplicity1T02 และ มัลติพลิซิตี ของคลาส WeakClassnameUML2 ที่แสดงส่วนหนึ่งจะมีลักษณะเป็น multiplicity2T01

ขั้นตอนการแปลงคลาสไดอะแกรมให้อยู่ในรูปแบบของสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุ

ความสัมพันธ์ระหว่างคลาสสองคลาสแบบคอลลิไฟล์แอสโซซิเอชัน จะถูกสร้าง ดังนี้

1. ความสัมพันธ์ระหว่างคลาสของคลาส ClassnameUML1 WeakClassnameUML2 ในคลาสไดอะแกรมจะถูกแปลงเป็นสคีมาตัวกลางสำหรับสร้างเป็น สคีมาฐานข้อมูลเชิงวัตถุในลักษณะเดียวกับกฎข้อที่ 3

2. ที่คลาส ClassnameUML2 เพิ่มแอทริบิวต์ที่มาจากคลาส ClassnameUML1

3. เพิ่มคีย์ที่เป็นตัวระบุถึงคลาสที่เป็นส่วนทั้งหมด ที่คลาส ClassnameUML2

ไวยากรณ์การสร้างสคีมาตัวกลางสำหรับสร้างเป็น สคีมาฐานข้อมูลเชิงวัตถุ

1. ไวยากรณ์ของการสร้างสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุตามมาตรฐานของโอดีเอ็มจี

```
<QUALIFIEDASSOCIATION> ::= <QUALIFIEDASSOCIATION-CLASS1>
                           <QUALIFIEDASSOCIATION-CLASS2>
<QUALIFIEDASSOCIATION-CLASS1>::=CLASSNAME_CLASS1-2:
                      <RELATIONSHIP-LIST_CLASS2-1> RELATIONSHIPNAME_CLASS1-2
                      inverse  CLASSNAME_CLASS1-2:: RELATIONSHIPNAME_CLASS2-1;

<QUALIFIEDASSOCIATION-CLASS2>::=CLASSNAME_CLASS2-1:
                      <RELATIONSHIP-LIST_CLASS1-2>   RELATIONSHIPNAME_CLASS2-1
                      inverse  CLASSNAME_CLASS2-1:: RELATIONSHIPNAME_CLASS1-2;

                      attribute <TYPE>  PARTIALKEY_CLASS1-2;

                      {<ATTRIBUTE_CLASS1-2>}
```

รูปที่ 4.39 ไวยากรณ์เบื้องต้นของการสร้างความสัมพันธ์แบบคอลลิไฟล์แอสโซซิเอชันตามมาตรฐานของโอดีเอ็มจี

2. ไวยากรณ์ของการสร้างสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุของฐานข้อมูลคาเช่

```
<QUALIFIEDASSOCIATION> ::= <QUALIFIEDASSOCIATION-CLASS1>
                           <QUALIFIEDASSOCIATION-CLASS2>
<QUALIFIEDASSOCIATION-CLASS1>::=CLASSNAME_CLASS1-2:
      <RELATIONSHIP-LIST_CLASS2-1> RELATIONSHIPNAME_CLASS1-2 as  CLASSNAME_CLASS2-1
      [Cardinality= MULTIPLICITY_CLASS1-2,inverse  RELATIONSHIPNAME_CLASS2-1] ;

<QUALIFIEDASSOCIATION-CLASS2>::=CLASSNAME_CLASS2-1:
      <RELATIONSHIP-LIST_CLASS2-1> RELATIONSHIPNAME_CLASS2-1 as  CLASSNAME_CLASS1-2
      [Cardinality= MULTIPLICITY_CLASS2-1,inverse  RELATIONSHIPNAME_CLASS1-2] ;

      property   PARTIALKEY_CLASS1-2  as % <TYPE> ;

      {<ATTRIBUTE_CLASS1-2>}
```

รูปที่ 4.40 ไวยากรณ์เบื้องต้นของการสร้างความสัมพันธ์แบบคอลลิไฟล์แอสโซซิเอชันของฐานข้อมูลคาเช่

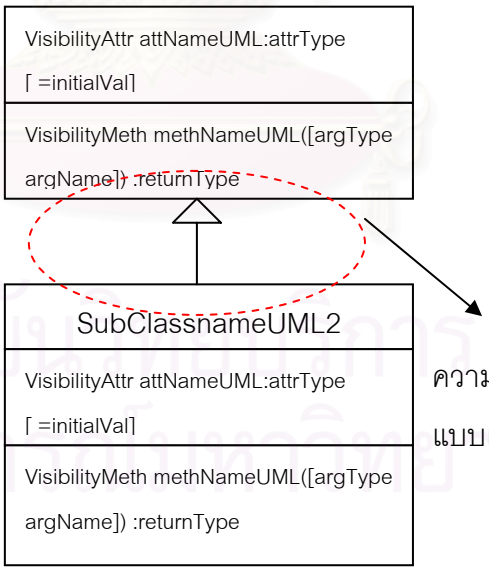
3. ไวยากรณ์ของการสร้างสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุของฐานข้อมูลแมทิส

```
<QUALIFIEDASSOCIATION> ::= <QUALIFIEDASSOCIATION-CLASS1>
                           <QUALIFIEDASSOCIATION-CLASS2>

<QUALIFIEDASSOCIATION-CLASS1>::=CLASSNAME_CLASS1-2:
        <RELATIONSHIP-LIST_CLASS2-1> RELATIONSHIPNAME_CLASS1-2[MULTIPLICITY_CLASS1-2]
        inverse  CLASSNAME_CLASS2-1:: RELATIONSHIPNAME_CLASS2-1;

<QUALIFIEDASSOCIATION-CLASS2>::=CLASSNAME_CLASS2-1:
        <RELATIONSHIP-LIST_CLASS1-2>  RELATIONSHIPNAME_CLASS2-1[MULTIPLICITY_CLASS2-1]
        inverse  CLASSNAME_CLASS1-2:: RELATIONSHIPNAME_CLASS1-2;

        attribute <TYPE>  PARTIALKEY_CLASS1-2;

        {<ATTRIBUTE_CLASS1-2>}
```

รูปที่ 4.41 ไวยากรณ์เบื้องต้นของการสร้างความสัมพันธ์แบบคอลลิไฟล์แอสโซซิเอชันของฐานข้อมูลแมทิส

4.10 กฎข้อที่ 10 กฎการแปลงความสัมพันธ์ระหว่างคลาสแบบเจนเนอรัลไลเซชันเป็นสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุ

รูปแบบของคลาสไดอะแกรม



ความสัมพันธ์ระหว่างคลาสแบบเจนเนอรัลไลเซชัน

รูปที่ 4.42 แสดงรูปแบบตามยูเอ็มแอลคลาสไดอะแกรมที่มีความสัมพันธ์ระหว่างคลาสแบบเจนเนอรัลไลเซชัน

ความสัมพันธ์ระหว่างคลาสแบบเจนเนอรัลไลเซชันคือ การสร้างความสัมพันธ์ให้คลาสสับคลาส มีเมทธอดและแอทริบิวต์เหมือนกับซูเปอร์คลาส เป็นการทำให้มีลักษณะทั่วไป ทำให้

กลไกการสืบทอดคุณสมบัติ(Inheritance) เกิดขึ้นได้โดยที่สับคลาสจะมีการสืบทอดคุณลักษณะต่าง ๆ ของซูเปอร์คลาสทุกประการ

จากรูปดังกล่าวจะประกอบไปด้วยส่วนประกอบต่าง ๆดังนี้คือ

1. ส่วนที่เรียกว่าซูปเปอร์คลาส คือ คลาส SupClassnameUML1 เป็นคลาสต้นแบบที่จะถ่ายทอดคุณสมบัติไปให้กับคลาสอื่น ๆ

2. ส่วนที่เรียกว่าสับคลาส คือ คลาส SubClassnameUML2 เป็นคลาสที่ต้องการสืบทอดคุณสมบัติจากคลาสที่เป็นซูปเอร์คลาส

ขั้นตอนการแปลงคลาสไดอะแกรมให้อยู่ในรูปแบบของสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุ

1. แปลงคลาสที่เป็นซูปเปอร์คลาส คือ SupClassnameUML1 ปรากฏในคลาสไดอะแกรมตามกฎข้อที่ 1

2. ส่วนที่เป็นสับคลาสคือ คลาส SubClassnameUML2 ให้แปลงความสัมพันธ์ตามกฎข้อที่ 1 ตามปกติ แต่ให้เพิ่มส่วนของการสืบทอดลงในคลาสที่เป็นสับคลาส โดยเพิ่มคำที่ระบุการสืบทอดเช่น *extends* เป็นต้น และระบุชื่อคลาสที่ต้องการสืบทอดมา และเพิ่มรายละเอียดเพิ่มเติมตามแต่ละประเภทของฐานข้อมูล

ไวยากรณ์การสร้างสคีมาตัวกลางสำหรับสร้างเป็น สคีมาฐานข้อมูลเชิงวัตถุ

1. ไวยากรณ์ของการสร้างสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุตามมาตรฐานของโอดีเอ็มจี

```
<GENERALIZATION>::=CLASSNAME_CLASS1-2:
                        extends   SUPERCLASSNAME_NAME
```

รูปที่ 4.43 ไวยากรณ์เบื้องต้นของการสร้างความสัมพันธ์แบบเจนเนอรัลไลเซชันตามมาตรฐานของโอดีเอ็มจี

2. ไวยากรณ์ของการสร้างสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุของฐานข้อมูลคาเช่

```
<GENERALIZATION>::=CLASSNAME_CLASS1-2:
          extends %Persistent[ClassType = persistent,ProcedureBlock] |
          extends %SUPERCLASSNAME_CLASSNAME%Persistent
             [ClassType = persistent,ProcedureBlock]
```

รูปที่ 4.44 ไวยากรณ์เบื้องต้นของการสร้างความสัมพันธ์แบบเจนเนอรัลไลเซชันของฐานข้อมูลคาเช่

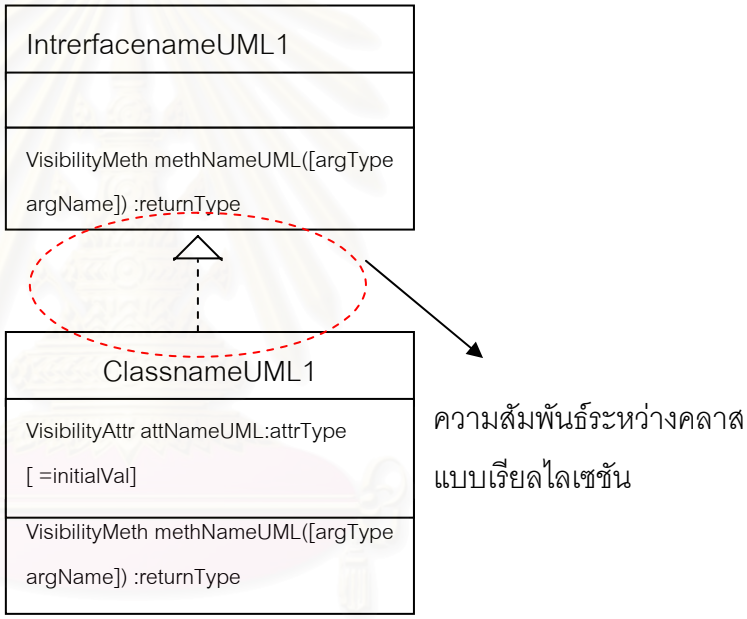
3. ไวยากรณ์ของการสร้างสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุของฐานข้อมูลแมทิส

```
<GENERALIZATION>::=CLASSNAME_CLASS1-2:
                   ":"SUPERCLASSNAME_NAME:Persistent
```

รูปที่ 4.45 ไวยากรณ์เบื้องต้นของการสร้างความสัมพันธ์แบบเจนเนอรัลไลเซชันของฐานข้อมูลแมทิส

4.11 กฎข้อที่ 11 กฎการแปลงความสัมพันธ์ระหว่างคลาสแบบเรียลไลเซชันเป็นสคีมาตัวกลางสำหรับสร้างเป็น สคีมาฐานข้อมูลเชิงวัตถุ

รูปแบบของคลาสไดอะแกรม



รูปที่ 4.46 แสดงคลาสไดอะแกรมที่มีความสัมพันธ์ระหว่างคลาสแบบเรียลไลเซชัน

ขั้นตอนการแปลงคลาสไดอะแกรมให้อยู่ในรูปแบบของสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุ

1. แปลงอินเตอร์เฟส IntrerfacenameUML1 โดยการระบุชื่ออินเตอร์เฟส รวมถึงเมทธอดของอินเตอร์เฟส เป็นสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุ

2. ส่วนที่เป็นการสืบทอดของอินเตอร์เฟสคือ คลาส ClassnameUML1 ให้แปลงความสัมพันธ์ตามกฎข้อที่ 1 ตามปกติ โดยเพิ่มคำที่ระบุการสืบทอด และระบุชื่ออินเตอร์เฟสที่ต้องการสืบทอด และเพิ่มรายละเอียดเพิ่มเติมตามแต่ละประเภทของฐานข้อมูล

ไวยากรณ์การสร้างสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุ

1. ไวยากรณ์ของการสร้างสคีมาตัวกลางสำหรับสร้างเป็น สคีมาฐานข้อมูลเชิงวัตถุตามมาตรฐานของโอดีเอ็มจี

```
<REALIZATION>::=CLASSNAME_CLASS1-2:
                ":"INTERFACENAME_NAME[{":"INTERFACENAME_NAME}]
```

รูปที่ 4.47 ไวยากรณ์เบื้องต้นของการสร้างความสัมพันธ์แบบเรียลไลเซชันตามมาตรฐานของโอดีเอ็มจี

2. ไวยากรณ์ของการสร้างสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุของฐานข้อมูลคาเช่

```
<REALIZATION>::=CLASSNAME_CLASS1-2:
                  ":"INTERFACENAME_NAME[{":"INTERFACENAME_NAME}]
```

รูปที่ 4.48 ไวยากรณ์เบื้องต้นของการสร้างความสัมพันธ์แบบเรียลไลเซชันตามมาตรฐานของฐานข้อมูลคาเช่

3. ไวยากรณ์ของการสร้างสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุของฐานข้อมูลแมทิส

```
<REALIZATION>::=CLASSNAME_CLASS1-2:
                ":"INTERFACENAME_NAME[{":"INTERFACENAME_NAME}]
```

รูปที่ 4.49 ไวยากรณ์เบื้องต้นของการสร้างความสัมพันธ์แบบเรียลไลเซชันตามมาตรฐานของฐานข้อมูลแมทิส

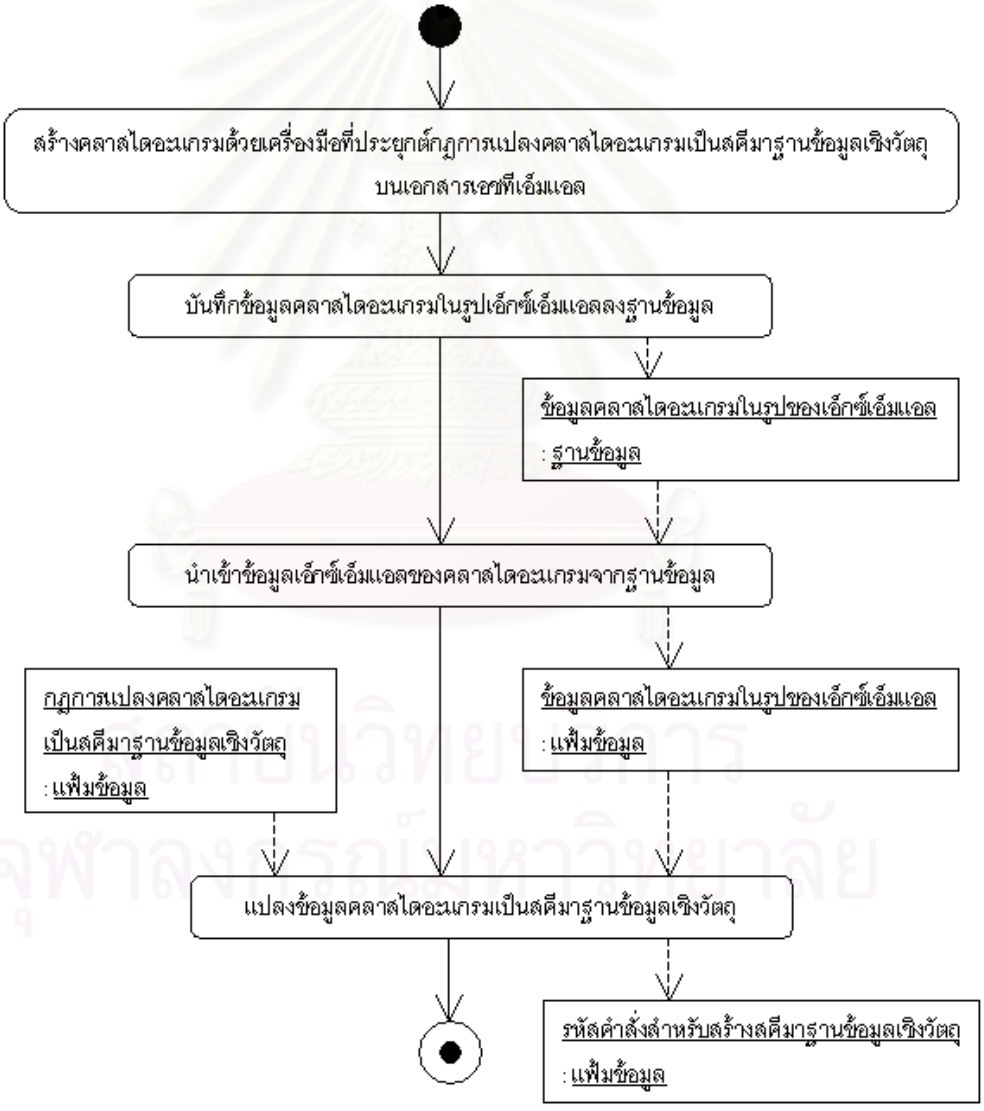
สำหรับการประยุกต์ใช้กฎทั้ง 11 ข้อของทั้ง 3 กรณีจะแสดงไว้ในภาคผนวก ก ตัวอย่างการประยุกต์ใช้กฎการแปลง ส่วนภาพรวมของไวยากรณ์จะแสดงไว้ในภาคผนวก ง ภาพรวมของไวยากรณ์ในการสร้างสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุ

บทที่ 5

การออกแบบและพัฒนาเครื่องมือที่ประยุกต์ใช้กฎการแปลงยูเอ็มแอลคลาสไดอะแกรมเป็นสคีมาฐานข้อมูลเชิงวัตถุ

ในบทนี้จะเป็นการกล่าวถึงการออกแบบและการพัฒนาเครื่องมือที่ประยุกต์ใช้กฎการแปลงยูเอ็มแอลคลาสไดอะแกรมเป็นสคีมาฐานข้อมูลเชิงวัตถุที่ได้ออกแบบในบทที่ 3 และบทที่ 4

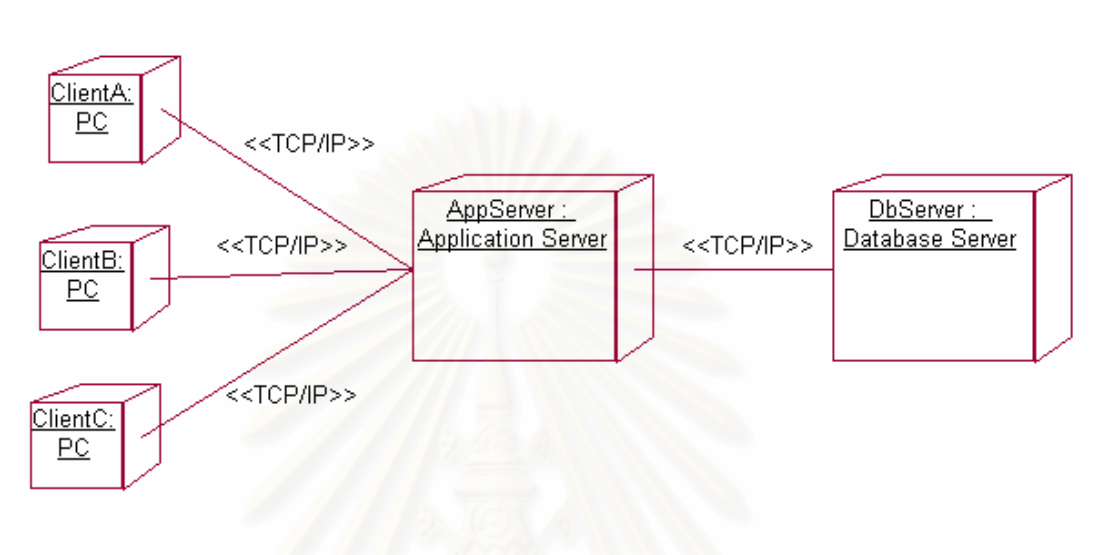
5.1 การออกแบบโครงสร้างพื้นฐานที่ใช้ในการพัฒนาเครื่องมือที่ประยุกต์ใช้กฎการแปลงยูเอ็มแอลคลาสไดอะแกรมเป็นสคีมาฐานข้อมูลเชิงวัตถุ



รูปที่ 5.1 ภาพรวมของกระบวนการแปลงคลาสไดอะแกรมเป็นสคีมาฐานข้อมูลเชิงวัตถุ

รูปที่ 5.1 เป็นภาพรวมของกระบวนการแปลงคลาสไดอะแกรมเป็นสคีมาฐานข้อมูลเชิงวัตถุซึ่งได้ประยุกต์กฎการแปลงคลาสไดอะแกรมเป็นสคีมาฐานข้อมูลเชิงวัตถุที่ได้ออกแบบไว้แล้วในบทที่ 3

5.2 สถาปัตยกรรมที่ใช้ในการพัฒนาเครื่องมือ



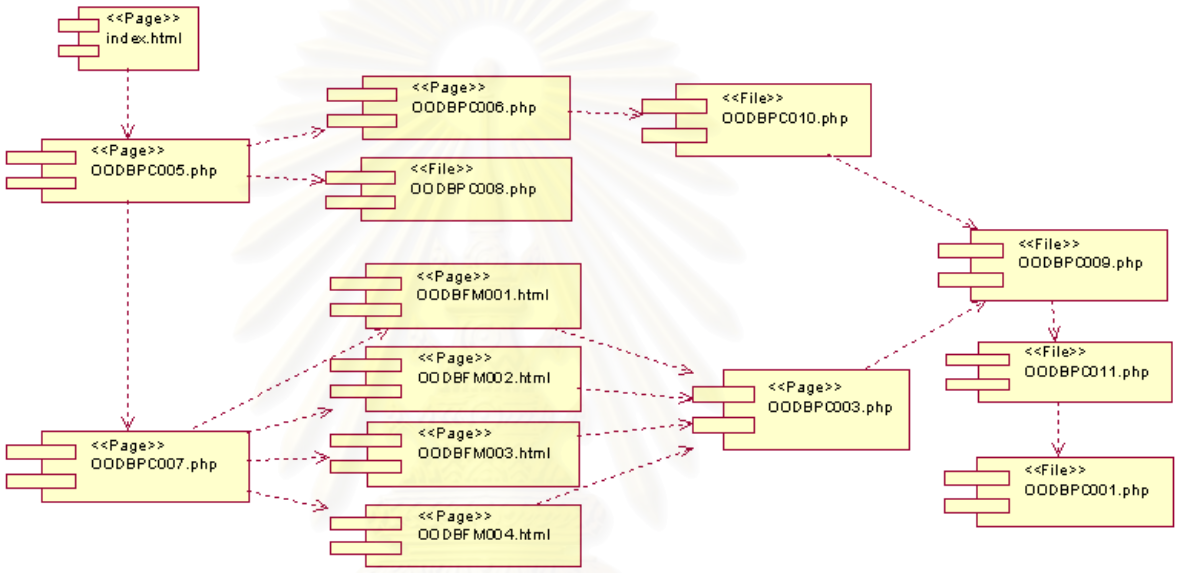
รูปที่ 5.2 ดีพลอยเมนต์ไดอะแกรมแสดงสถาปัตยกรรมการทำงานของเครื่องมือ

จากรูปที่ 5.2 แสดงให้เห็นถึงสถาปัตยกรรมของการทำงานของเครื่องมือ โดยผู้ใช้จะมีการทำงานผ่านเว็บบราวเซอร์ ในรูปแบบเว็บแอพลิเคชัน ซึ่งลักษณะการทำงานเป็นการทำงานแบบผ่านเซิร์ฟเวอร์ (Server side script) โดยการทำงานฝั่งไคลแอนท์ (Client) จะเป็นฝั่งที่ผู้ใช้เครื่องมือออกแบบคลาสไดอะแกรมและป้อนข้อมูลรายละเอียดต่าง ๆ สำหรับการสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุ โดยข้อมูลบางส่วนจะส่งไปที่ฝั่งเซิร์ฟเวอร์เพื่อให้ประมวลผล ส่วนการทำงานฝั่งเซิร์ฟเวอร์จะเป็นการทำงานในส่วนของการประยุกต์ใช้กฎการแปลงคลาสไดอะแกรมเป็นสคีมาฐานข้อมูลเชิงวัตถุ รวมถึงการรับและส่งข้อมูลภายหลังการประมวลผลเพื่อส่งให้ไคลแอนท์แสดงผล ส่วนดาต้าเบสเซิร์ฟเวอร์จะเป็นการทำงานในส่วนของการเก็บข้อมูลเบื้องต้นของคลาสไดอะแกรมรวมถึงสคีมาฐานข้อมูลเชิงวัตถุที่ได้จากการแปลง เรียกการทำงานในลักษณะนี้ว่า 3-tiers model application สรุปส่วนประกอบของการทำงานในลักษณะนี้ คือ

1. ส่วนการแสดงผล(Presentation logic) ทำหน้าที่รับข้อมูลคลาสไดอะแกรมจากผู้ใช้ รวมถึงการแสดงผลลัพธ์ต่าง ๆที่ได้จากการประมวลผล
2. ส่วนการประมวลผล (Businesss logic) ทำหน้าที่แปลงคลาสไดอะแกรมที่ได้จากผู้ใช้โดยการประยุกต์กฎที่ออกแบบไว้แล้วในบทที่ 3

3. ส่วนฐานข้อมูล(Database logic) ทำหน้าที่เก็บข้อมูลเบื้องต้นของคลาสไดอะแกรม รวมถึงสคีมาฐานข้อมูลเชิงวัตถุที่ได้จากการแปลง

ในส่วนของคอมโพเนนต์ต่าง ๆ ที่สร้างภายในเครื่องมือที่ประยุกต์กฎการแปลงยูเอ็มแอลคลาสไดอะแกรมเป็นสคีมาฐานข้อมูลเชิงวัตถุ สามารถแสดงได้ดังรูปที่ 5.3



รูปที่ 5.3 คอมโนเนนต์ไดอะแกรมแสดงคอมโพเนนต์ต่าง ๆ ที่สร้างภายในเครื่องมือที่ประยุกต์กฎการแปลงยูเอ็มแอลคลาสไดอะแกรมเป็นสคีมาฐานข้อมูลเชิงวัตถุ

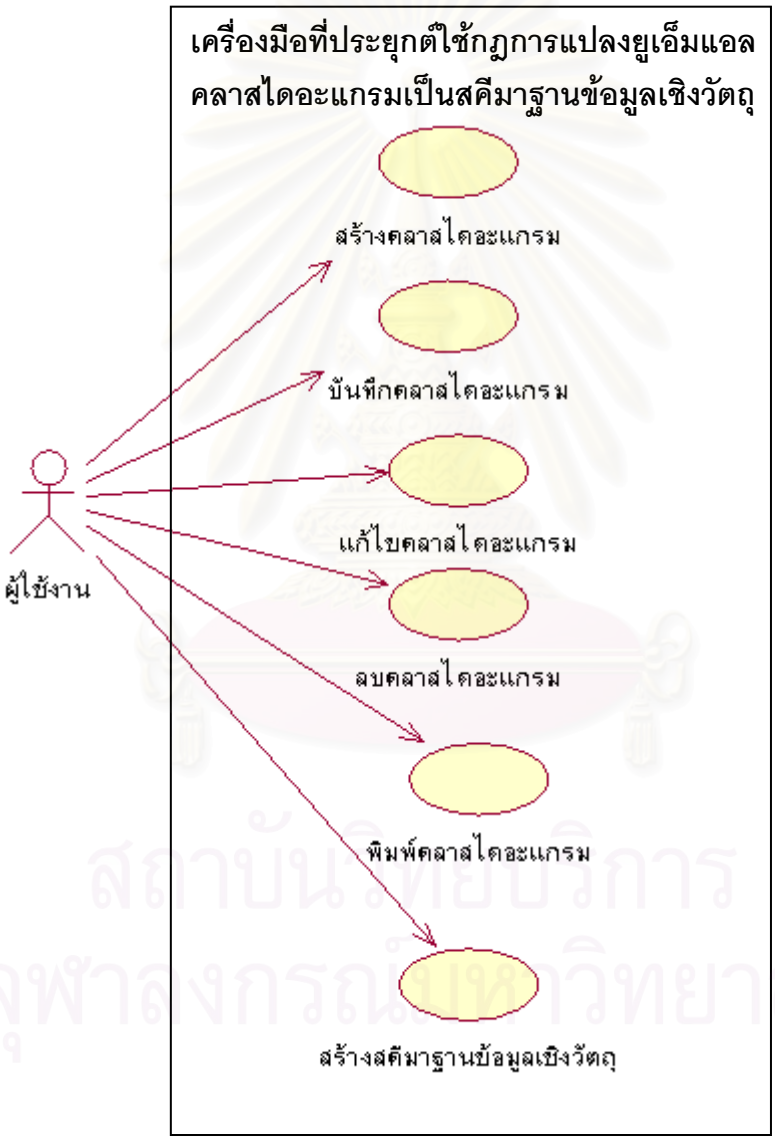
จากรูปที่ 5.3 คอมโพเนนต์ต่าง ๆ ที่สร้างภายในเครื่องมือที่ประยุกต์กฎการแปลงยูเอ็มแอลคลาสไดอะแกรมเป็นสคีมาฐานข้อมูลเชิงวัตถุ มีส่วนประกอบและรายละเอียดดังตารางที่ 5.1

ตารางที่ 5.1 ส่วนประกอบและรายละเอียดของคอมโพเนนต์ต่าง ๆ ที่สร้างภายในเครื่องมือที่ประยุกต์กฎการแปลงยูเอ็มแอลคลาสไดอะแกรมเป็นสคีมาฐานข้อมูลเชิงวัตถุ

ชื่อคอมโพเนนต์	ประเภทของคอมโพเนนต์	รายละเอียดของคอมโพเนนต์
index.html	<<Page>>	หน้าแรกของเครื่องมือ
OODBPC001.php	<<File>>	ไฟล์สำหรับการแปลงคลาสไดอะแกรมเป็นสคีมาฐานข้อมูลเชิงวัตถุ
OODBPC002.php	<<Page>>	หน้าจอแสดงการเปิดคลาสไดอะแกรม
OODBPC003.php	<<Page>>	หน้าจอแสดงการยืนยันการบันทึกคลาสไดอะแกรม
OODBPC004.php	<<Page>>	หน้าจอแสดงการลบคลาสไดอะแกรม
OODBPC005.php	<<Page>>	หน้าจอแสดงรายชื่อโปรเจ็ค
OODBPC006.php	<<Page>>	หน้าจอแสดงการเปิดโปรเจ็ค
OODBPC007.php	<<Page>>	หน้าจอแสดงการสร้างโปรเจ็คใหม่
OODBPC008.php	<<Page>>	หน้าจอแสดงการลบโปรเจ็ค
OODBPC009.php	<<File>>	ไฟล์สำหรับการบันทึกคลาสไดอะแกรม
OODBPC010.php	<<File>>	ไฟล์สำหรับการเปิดคลาสไดอะแกรม
OODBPC011.php	<<File>>	ไฟล์สำหรับการแก้ไขคลาสไดอะแกรม
OODBFM001.html	<<Page>>	หน้าจอการเลือกประเภทฐานข้อมูลสำหรับแปลง
OODBFM002.html	<<Page>>	หน้าจอการเพิ่มข้อมูลของคลาส
OODBFM003.html	<<Page>>	หน้าจอการเพิ่มข้อมูลของอินเตอร์เฟส
OODBFM004.html	<<Page>>	หน้าจอการเพิ่มรายละเอียดของความสัมพันธ์เมทธอด และแอทริบิวต์ต่าง ๆ

5.3 การออกแบบการใช้งานของเครื่องมือที่ประยุกต์ใช้กฎการแปลงยูเอ็มแอลคลาสไดอะแกรมเป็นสคีมาฐานข้อมูลเชิงวัตถุ

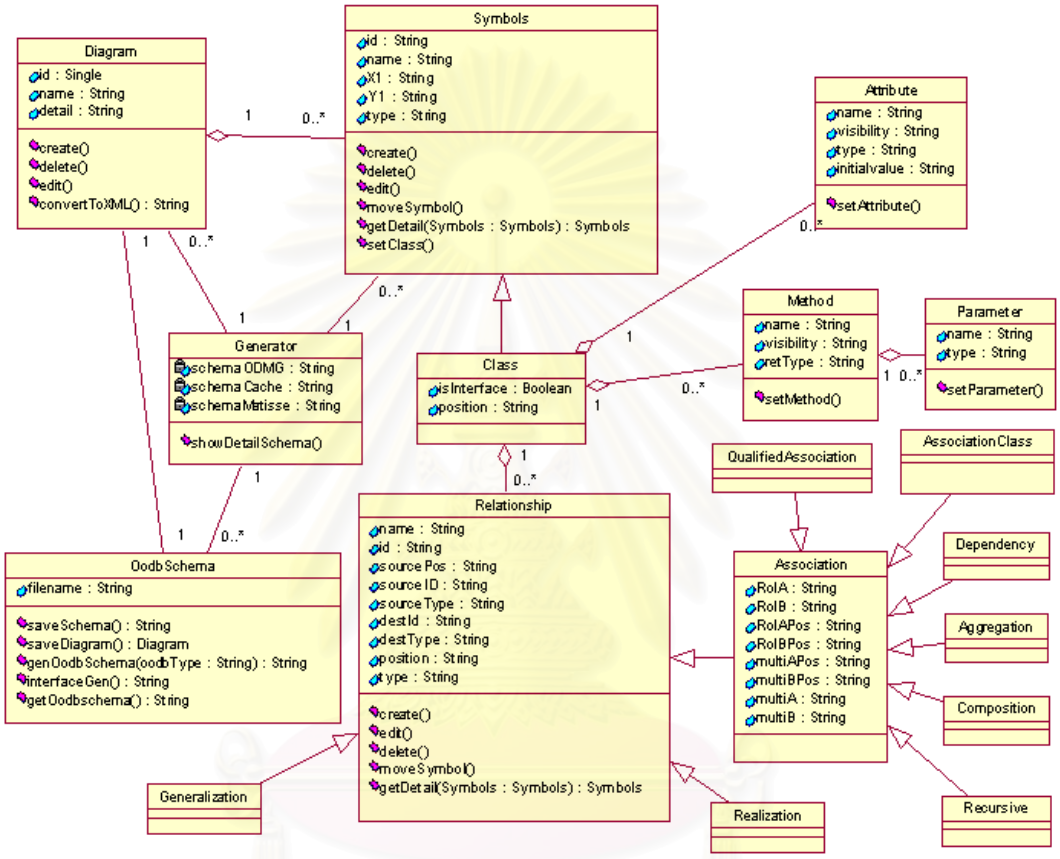
จากรูปที่ 5.4 แสดงให้เห็นถึงยูสเคสไดอะแกรมของเครื่องมือที่ประยุกต์กฎการแปลงคลาสไดอะแกรมเป็นสคีมาฐานข้อมูลเชิงวัตถุโดยจะแสดงฟังก์ชันการทำงานทั้งหมดที่เครื่องมือดังกล่าวสามารถทำได้โดยประกอบไปด้วย ฟังก์ชันหลักของการทำงานของเครื่องมือคือ การสร้างคลาสไดอะแกรม การบันทึกคลาสไดอะแกรม การแก้ไขคลาสไดอะแกรม การลบคลาสไดอะแกรม การพิมพ์คลาสไดอะแกรม และการสร้างสคีมาฐานข้อมูลเชิงวัตถุ



รูปที่ 5.4 แสดงยูสเคสไดอะแกรมของเครื่องมือที่ประยุกต์กฎการแปลงคลาสไดอะแกรมเป็นสคีมาฐานข้อมูลเชิงวัตถุ

5.4 คลาสไดอะแกรมของเครื่องมือที่ประยุกต์การใช้กฎการแปลงคลาสไดอะแกรมเป็นสคีมาฐานข้อมูลเชิงวัตถุ

คลาสไดอะแกรมสำหรับการสร้างเครื่องมือการแปลงคลาสไดอะแกรมเป็นสคีมาฐานข้อมูลเชิงวัตถุ แสดงดังรูปที่ 5.5



รูปที่ 5.5 คลาสไดอะแกรมของเครื่องมือที่ประยุกต์การใช้กฎการแปลงยูเอ็มแอลคลาสไดอะแกรมเป็นสคีมาฐานข้อมูลเชิงวัตถุ

ตารางที่ 5.2 ถึง 5.18 แสดงรายละเอียดของแต่ละส่วนย่อยในคลาสไดอะแกรมของเครื่องมือที่ประยุกต์การใช้กฎการแปลงยูเอ็มแอลคลาสไดอะแกรมเป็นสคีมาฐานข้อมูลเชิงวัตถุ

ตารางที่ 5.2 รายละเอียดของส่วนย่อย “Class”

ชื่อส่วนย่อย	Class
คำอธิบาย	เก็บรายละเอียดของคลาส ในคลาสไดอะแกรมสืบทอดมาจาก Symbols
แอทริบิวต์	
- isInterface	สำหรับเช็คประเภทของสัญลักษณ์ที่เป็นคลาสหรืออินเตอร์เฟส
- position	ตำแหน่งของคลาส

ตารางที่ 5.3 รายละเอียดของส่วนย่อย “Generator”

ชื่อส่วนย่อย	Generator
คำอธิบาย	คลาสที่ทำหน้าที่สร้างสคีมาฐานข้อมูลเชิงวัตถุ
แอทริบิวต์	
- schemaODMG	เก็บสคีมาฐานข้อมูลเชิงวัตถุตามมาตรฐานโอดีเอ็มจี
- schemaCache	เก็บสคีมาฐานข้อมูลเชิงวัตถุของฐานข้อมูลคาเช่
- schemaMatisse	เก็บสคีมาฐานข้อมูลเชิงวัตถุของฐานข้อมูลแมทิส
เมทธอด	
- showDetailSchema()	สำหรับการแสดงรายละเอียดสคีมาฐานข้อมูลเชิงวัตถุ

ตารางที่ 5.4 รายละเอียดของส่วนย่อย “OodbSchema”

ชื่อส่วนย่อย	OodbSchema
คำอธิบาย	เก็บรายละเอียดของสคีมาฐานข้อมูลเชิงวัตถุที่ได้จากการแปลงจากคลาสไดอะแกรม
แอทริบิวต์	
- filename	ชื่อไฟล์ของสคีมาที่ได้จาการแปลง
เมทธอด	
- saveSchema()	สำหรับการบันทึกสคีมาฐานข้อมูลเชิงวัตถุ
- interfaceGen()	การแปลงอินเทอร์เฟสเป็นสคีมาฐานข้อมูลเชิงวัตถุ
- genOodbSchema (oodbType:String)	การแปลงคลาสเป็นสคีมาฐานข้อมูลเชิงวัตถุ
- getOodbschema()	การดึงข้อมูลสคีมาที่ได้ภายหลังการแปลง

ตารางที่ 5.5 รายละเอียดของส่วนย่อย “Method”

ชื่อส่วนย่อย	Method
คำอธิบาย	เก็บรายละเอียดของเมทธอด ในคลาสไดอะแกรมเป็นส่วนหนึ่งของ Class
แอทริบิวต์	
- name	ชื่อของเมทธอด
- visibility	ความสามารถการมองเห็นของเมทธอด
- retType	ประเภทของข้อมูลส่งกลับ
- setMethod()	การกำหนดรายละเอียดของเมทธอด

ตารางที่ 5.6 รายละเอียดของส่วนย่อย “Diagram”

ชื่อส่วนย่อย	Diagram
คำอธิบาย	เก็บรายละเอียดของคลาสไดอะแกรม
แอทริบิวต์	
- id	รหัสของคลาสไดอะแกรมที่ถูกสร้าง
- name	ชื่อของคลาสไดอะแกรม
- detail	ข้อมูลเพิ่มเติมของคลาสไดอะแกรม
เมทธอด	
- convertToXML()	การแปลงข้อมูลสัญลักษณ์ให้อยู่ในรูปของเอ็กซ์เอ็มแอล
- create()	การสร้างไดอะแกรม
- delete()	การลบไดอะแกรม
- edit()	การแก้ไขไดอะแกรม

ตารางที่ 5.7 รายละเอียดของส่วนย่อย “Symbols”

ชื่อส่วนย่อย	Symbols
คำอธิบาย	เก็บรายละเอียดของสัญลักษณ์ต่าง ๆ ในคลาสไดอะแกรม
แอทริบิวต์	
- id	รหัสของสัญลักษณ์ต่างๆ ของคลาสไดอะแกรมที่ถูกสร้าง
- name	ชื่อของสัญลักษณ์ต่างๆ ของคลาสไดอะแกรมที่ถูกสร้าง
- X1	ตำแหน่งของสัญลักษณ์
- Y1	ตำแหน่งของสัญลักษณ์
- type	ประเภทของสัญลักษณ์
เมทธอด	
- moveSymbol()	การย้ายสัญลักษณ์
- setClass()	การกำหนดรายละเอียดของคลาส
- create()	การสร้างสัญลักษณ์
- delete()	การลบสัญลักษณ์
- edit()	การแก้ไขสัญลักษณ์
- getDetail (Symbols:Symbols)	การดึงรายละเอียดของสัญลักษณ์แต่ละประเภท

ตารางที่ 5.8 รายละเอียดของส่วนย่อย “Relationship”

ชื่อส่วนย่อย	Relationship
คำอธิบาย	เก็บรายละเอียดของความสัมพันธ์ระหว่างคลาส ในคลาสไดอะแกรมเป็นส่วนหนึ่งของ Class
แอทริบิวต์	
- id	รหัสของสัญลักษณ์
- name	ชื่อของสัญลักษณ์
- sourcePos	ตำแหน่งของสัญลักษณ์ต้นทาง
- sourceID	รหัสของสัญลักษณ์ต้นทาง
- sourceType	ตำแหน่งของสัญลักษณ์ปลายทาง
- destId	รหัสของสัญลักษณ์ปลายทาง
- destType	ประเภทของคลาสปลายทาง
- position	ตำแหน่งของสัญลักษณ์
- type	ประเภทของสัญลักษณ์
เมทธอด	
- moveSymbol()	การย้ายสัญลักษณ์
- getDetail (Symbols:Symbols)	การดึงรายละเอียดของสัญลักษณ์
- create()	การสร้างสัญลักษณ์
- delete()	การลบสัญลักษณ์
- edit()	การแก้ไขสัญลักษณ์

ตารางที่ 5.9 รายละเอียดของส่วนย่อย “Attribute”

ชื่อส่วนย่อย	Attribute
คำอธิบาย	เก็บรายละเอียดของแอทริบิวต์ในคลาสไดอะแกรมเป็นส่วนหนึ่งของ Class
แอทริบิวต์	
- name	ชื่อของแอทริบิวต์
- visibility	ความสามารถการมองเห็นของแอทริบิวต์
- type	ประเภทของแอทริบิวต์
- initialvalue	ค่าเริ่มต้นของแอทริบิวต์
- set Attribute ()	การกำหนดรายละเอียดของแอทริบิวต์

ตารางที่ 5.10 รายละเอียดของส่วนย่อย “Association”

ชื่อส่วนย่อย	Association
คำอธิบาย	เก็บรายละเอียดของความสัมพันธ์แบบแอสโซสิเอชันสืบทอดมาจาก Relationship
แอทริบิวต์	
- RolA	บทบาทของคลาสที่ 1
- RolB	บทบาทของคลาสที่ 2
- RolAPos	ตำแหน่งของบทบาทของคลาสที่ 1
- RolBPos	ตำแหน่งของบทบาทของคลาสที่ 1
- multiAPos	ตำแหน่งของมัลติพลิซิตีของคลาสที่ 1
- multiBPos	ตำแหน่งของมัลติพลิซิตีของคลาสที่ 2
- multiA	มัลติพลิซิตีของคลาสที่ 1
- multiB	มัลติพลิซิตีของคลาสที่ 2

ตารางที่ 5.11 รายละเอียดของส่วนย่อย “Parameter”

ชื่อส่วนย่อย	Parameter
คำอธิบาย	เก็บรายละเอียดของของอาร์กูเมนต์ของเมทธอดในคลาสไดอะแกรมเป็นส่วนหนึ่งของ Method
แอทริบิวต์	
- name	ชื่อของอาร์กูเมนต์ของเมทธอด
- type	ประเภทข้อมูลของอาร์กูเมนต์ของเมทธอด

ตารางที่ 5.12 รายละเอียดของส่วนย่อย “Realization”

ชื่อส่วนย่อย	Realization
คำอธิบาย	เก็บรายละเอียดของความสัมพันธ์แบบเรียลไลเซชันสืบทอดมาจาก Relationship

ตารางที่ 5.13 รายละเอียดของส่วนย่อย “Dependency”

ชื่อส่วนย่อย	Dependency
คำอธิบาย	เก็บรายละเอียดของความสัมพันธ์แบบดีเพนเดนซีสืบทอดมาจาก Relationship

ตารางที่ 5.14 รายละเอียดของส่วนย่อย “Generalization”

ชื่อส่วนย่อย	Generalization
คำอธิบาย	เก็บรายละเอียดของความสัมพันธ์แบบเจนเนอรอลไลเซชันสืบทอดมาจาก Relationship

ตารางที่ 5.15 รายละเอียดของส่วนย่อย “QualifiedAssociation”

ชื่อส่วนย่อย	QualifiedAssociation
คำอธิบาย	เก็บรายละเอียดของความสัมพันธ์แบบคอลลิไฟล์สืบทอดมาจาก Association

ตารางที่ 5.16 รายละเอียดของส่วนย่อย “AssociationClass”

ชื่อส่วนย่อย	AssociationClass
คำอธิบาย	เก็บรายละเอียดของความสัมพันธ์แบบแอสโซสิเอชันคลาส สืบทอดมาจาก Association

ตารางที่ 5.17 รายละเอียดของส่วนย่อย “Aggregation”

ชื่อส่วนย่อย	Aggregation
คำอธิบาย	เก็บรายละเอียดของความสัมพันธ์แบบแอกกรีเกชันสืบทอดมาจาก Association

ตารางที่ 5.18 รายละเอียดของส่วนย่อย “Composition”

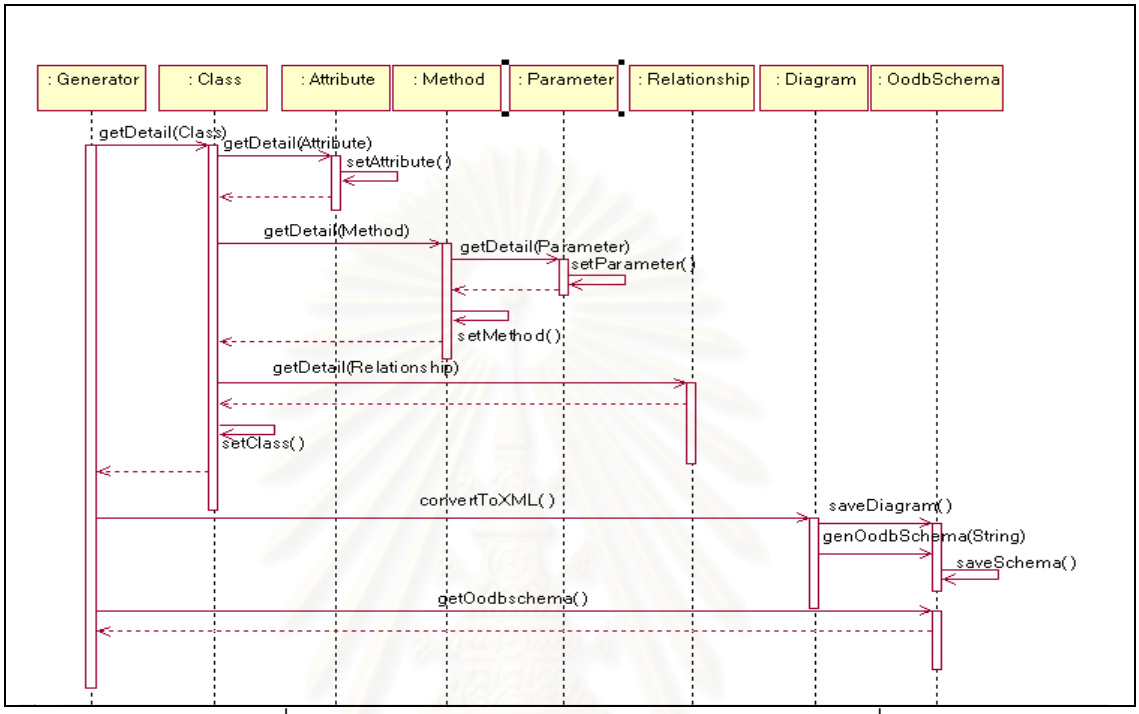
ชื่อส่วนย่อย	Composition
คำอธิบาย	เก็บรายละเอียดของความสัมพันธ์แบบคอมโพสิชัน สืบทอดมาจาก Association

ตารางที่ 5.19 รายละเอียดของส่วนย่อย “Recursive”

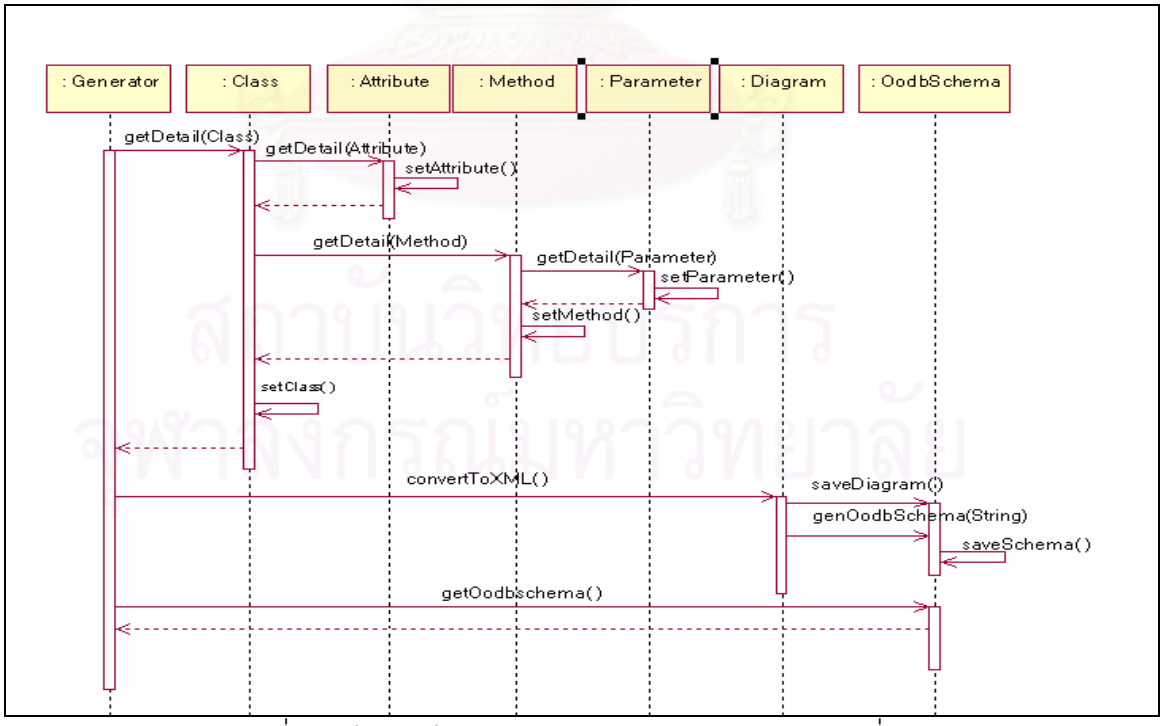
ชื่อส่วนย่อย	Recursive
คำอธิบาย	เก็บรายละเอียดของความสัมพันธ์แบบรีเคอร์ซีพ สืบทอดมาจาก Association

5.5 ซีเควนซ์ไดอะแกรมของกฎการแปลงคลาสไดอะแกรมเป็นสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุของกฎทั้ง 11 ข้อ

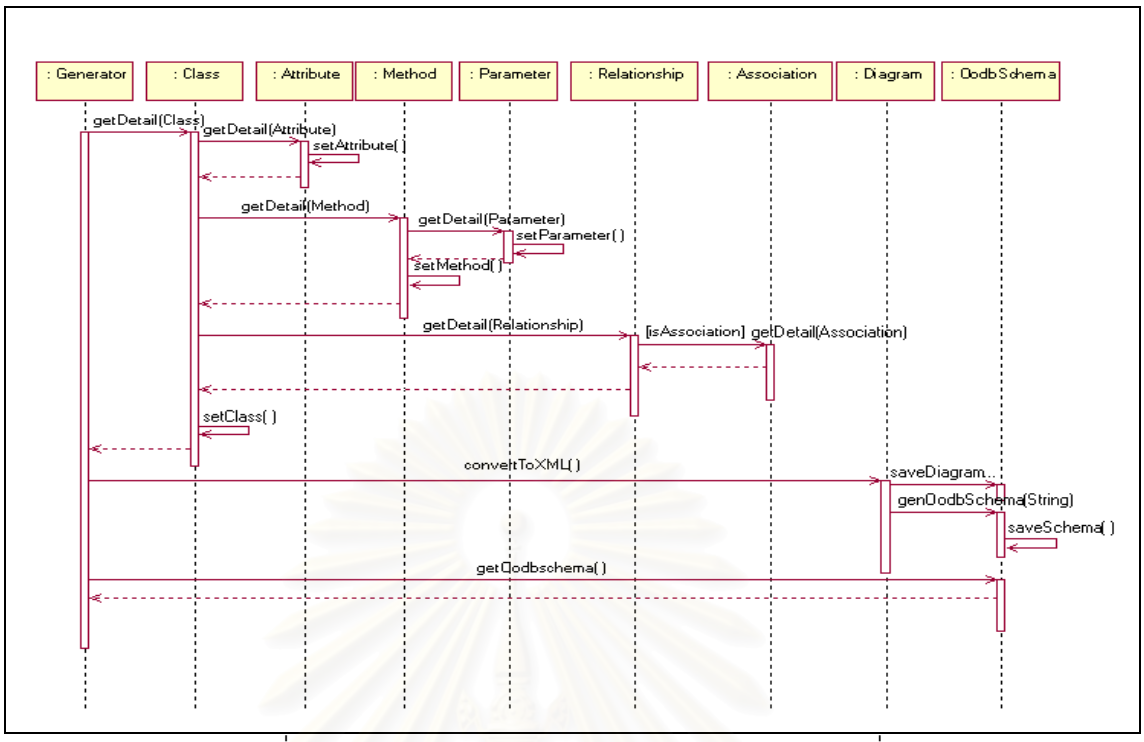
จากกฎการแปลงคลาสไดอะแกรมเป็นสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุทั้ง 11 ข้อในบทที่ 4 สร้างเป็นซีเควนซ์ไดอะแกรมของกฎแต่ละข้อได้ดังนี้



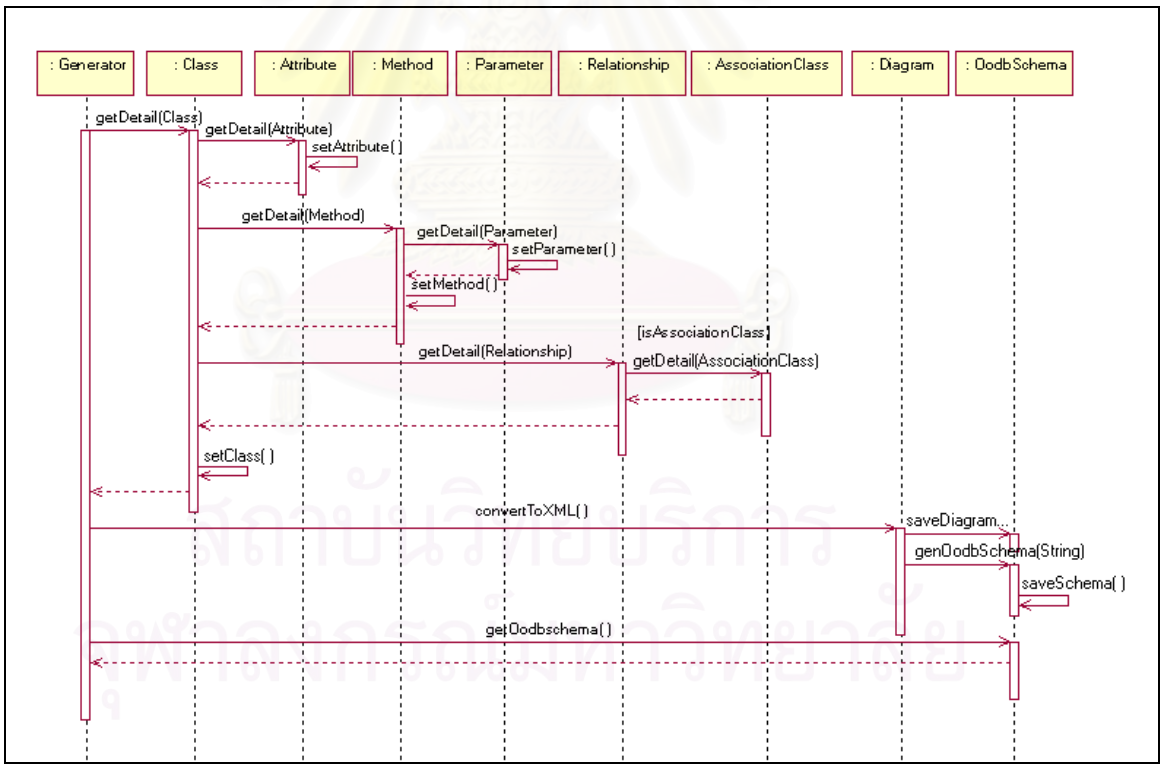
รูปที่ 5.6 ซีเควนซ์ไดอะแกรมแสดงการทำงานของกฎข้อที่ 1



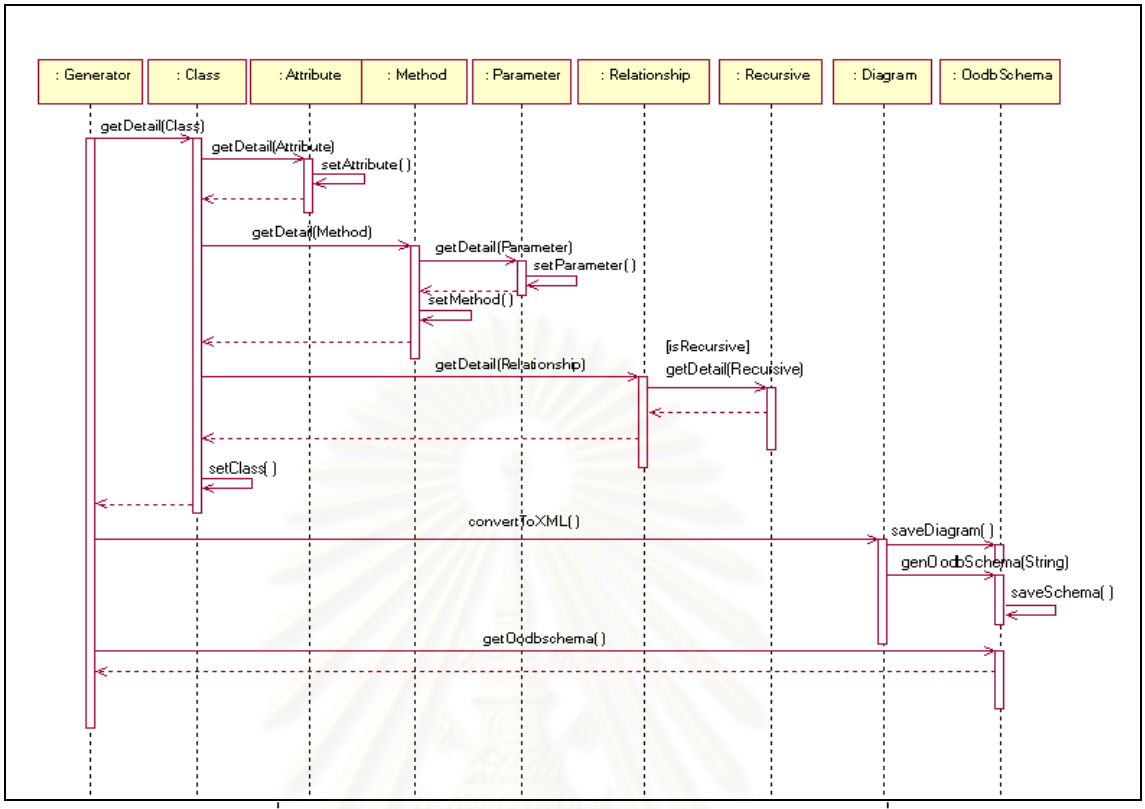
รูปที่ 5.7 ซีเควนซ์ไดอะแกรมแสดงการทำงานของกฎข้อที่ 2



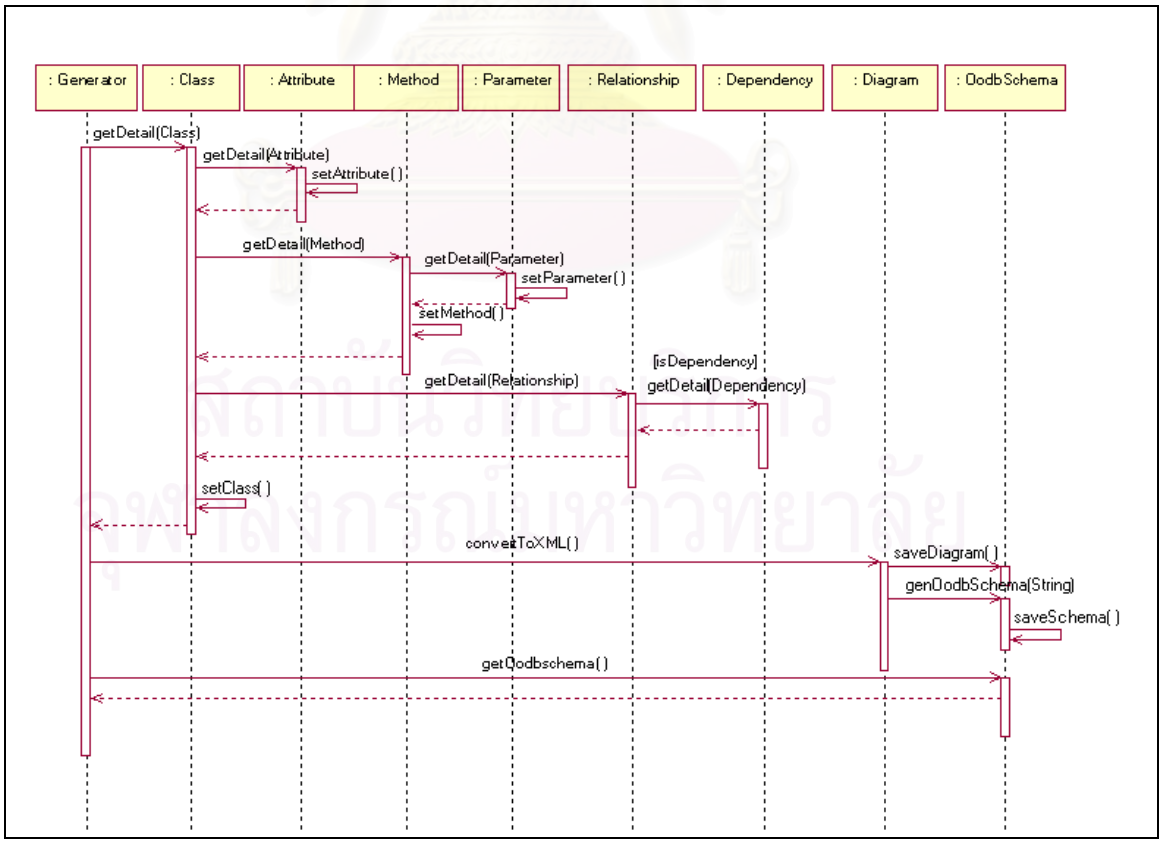
รูปที่ 5.8 ซีเควนซ์ไดอะแกรมแสดงการทำงานของกฎข้อที่ 3



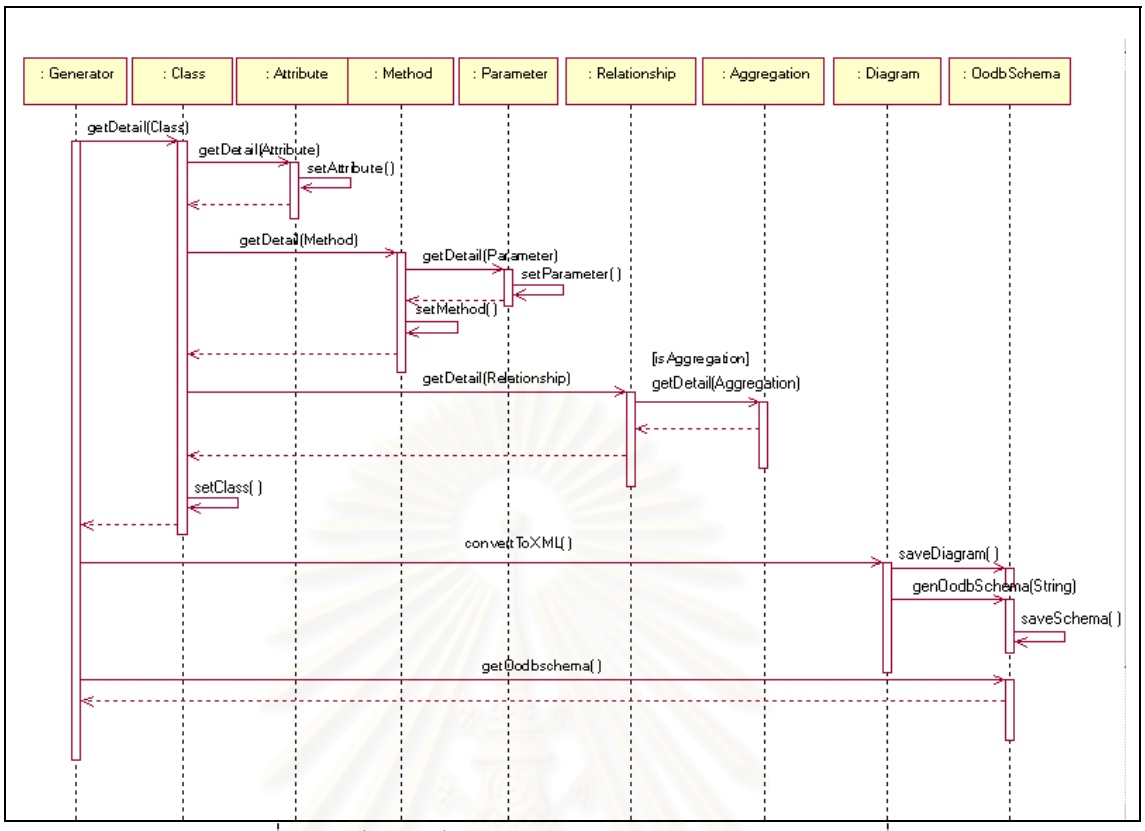
รูปที่ 5.9 ซีเควนซ์ไดอะแกรมแสดงการทำงานของกฎข้อที่ 4



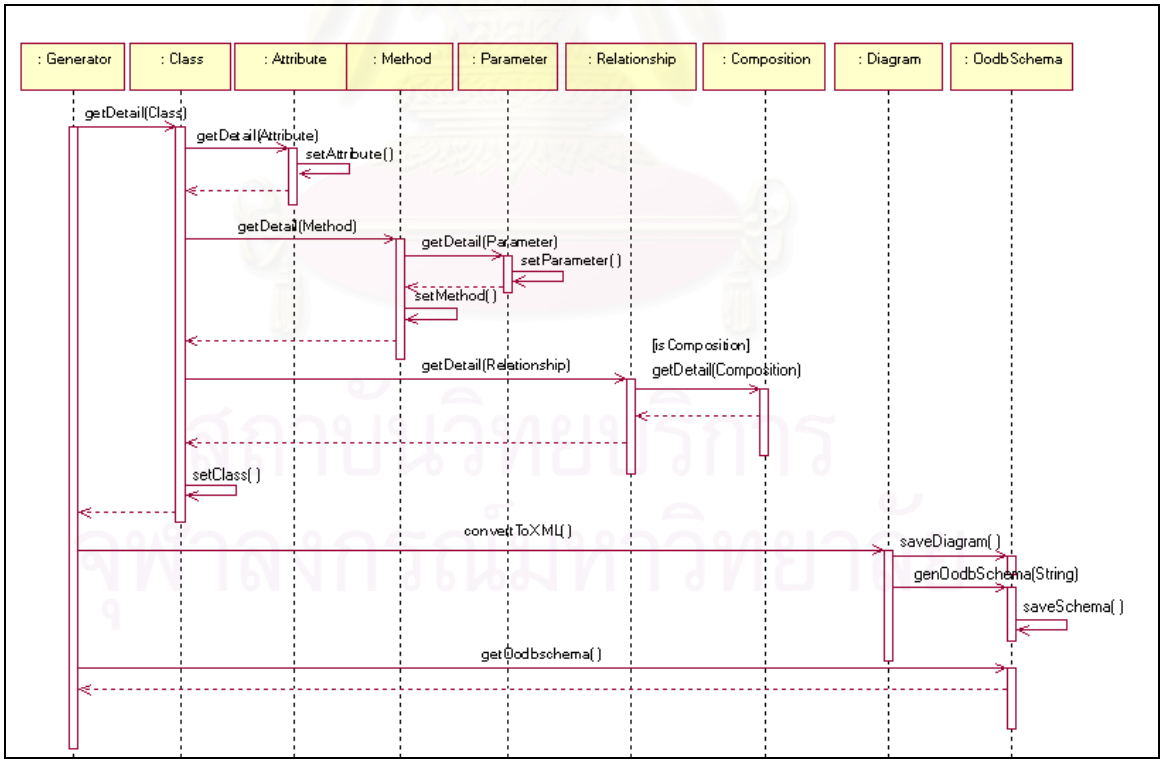
รูปที่ 5.10 ซีเควนซ์ไดอะแกรมแสดงการทำงานของกฎข้อที่ 5



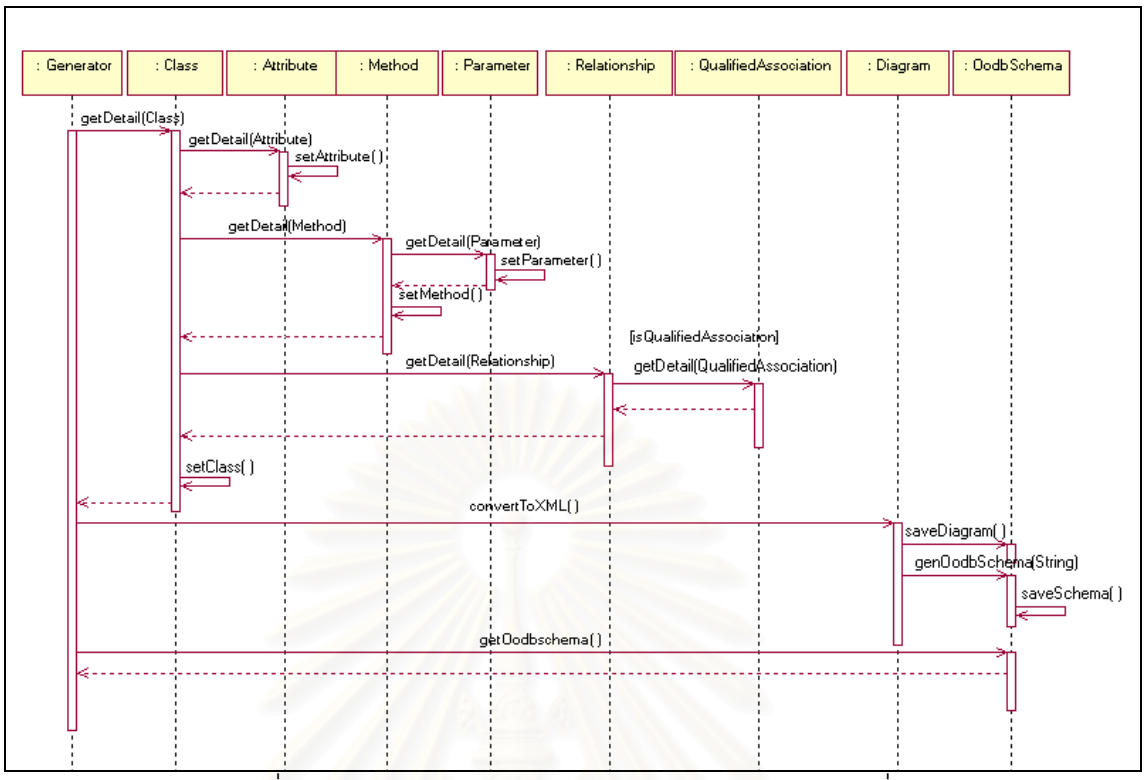
รูปที่ 5.11 ซีเควนซ์ไดอะแกรมแสดงการทำงานของกฎข้อที่ 6



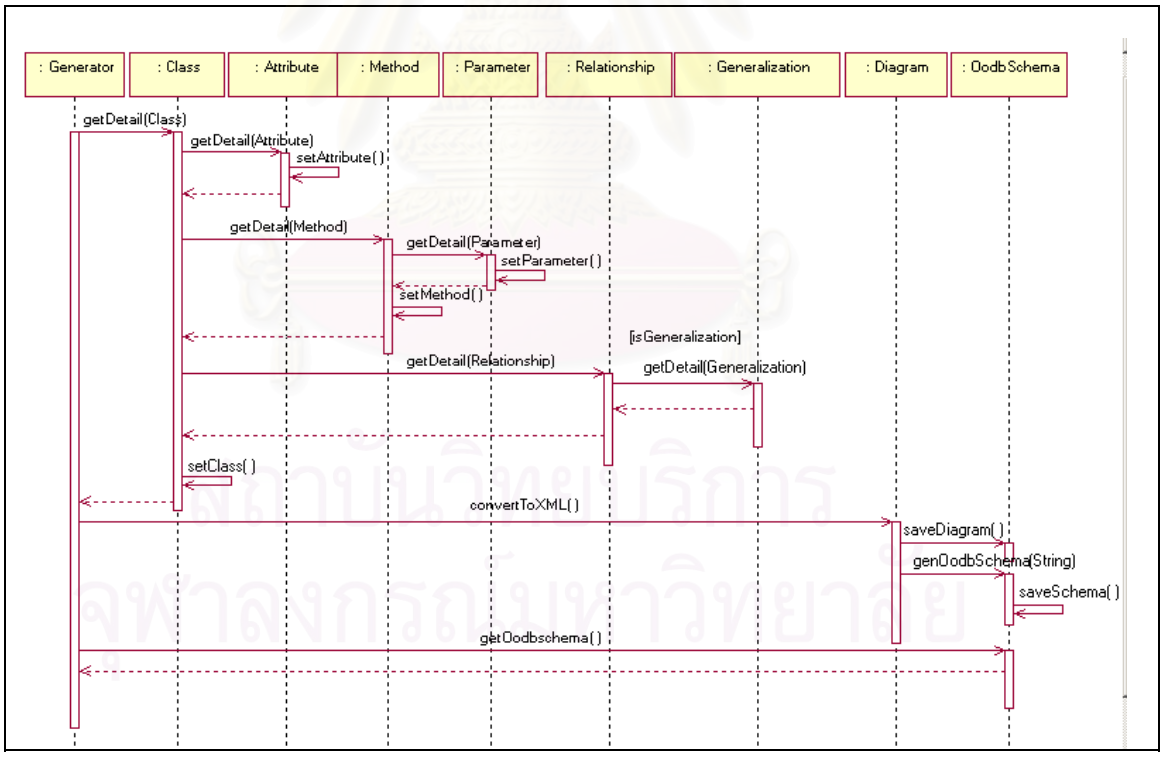
รูปที่ 5.12 ซีเควนซ์ไดอะแกรมแสดงการทำงานของกฎข้อที่ 7



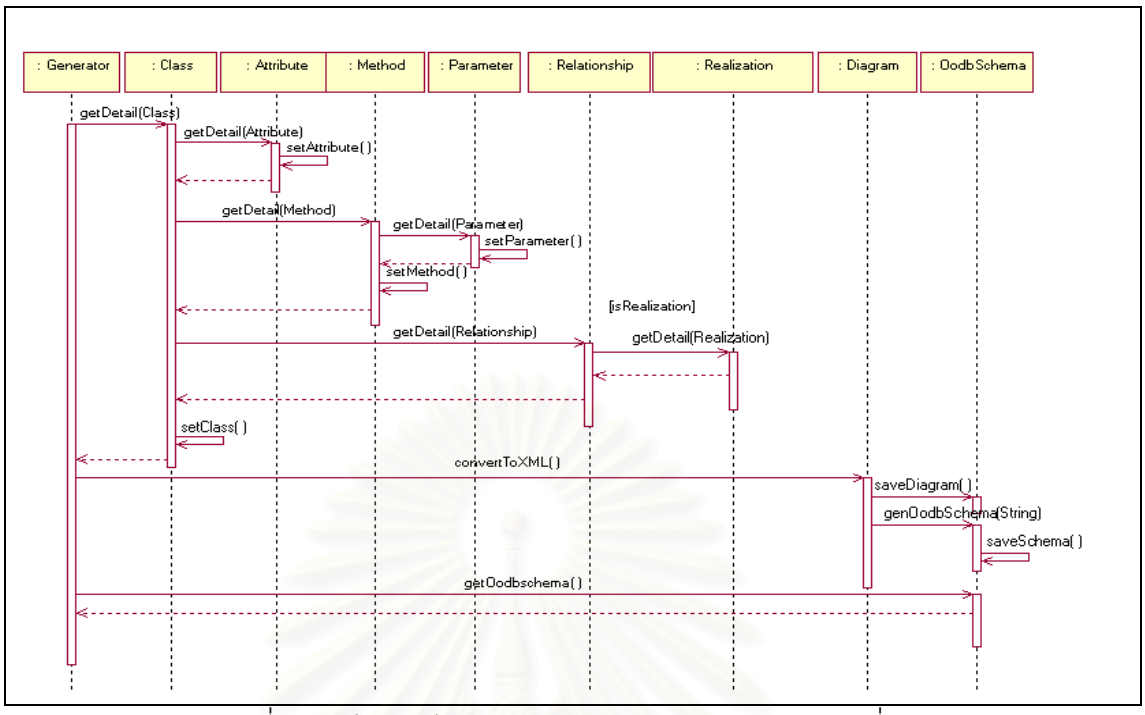
รูปที่ 5.13 ซีเควนซ์ไดอะแกรมแสดงการทำงานของกฎข้อที่ 8



รูปที่ 5.14 ซีเควนซ์ไดอะแกรมแสดงการทำงานของกฎข้อที่ 9



รูปที่ 5.15 ซีเควนซ์ไดอะแกรมแสดงการทำงานของกฎข้อที่ 10



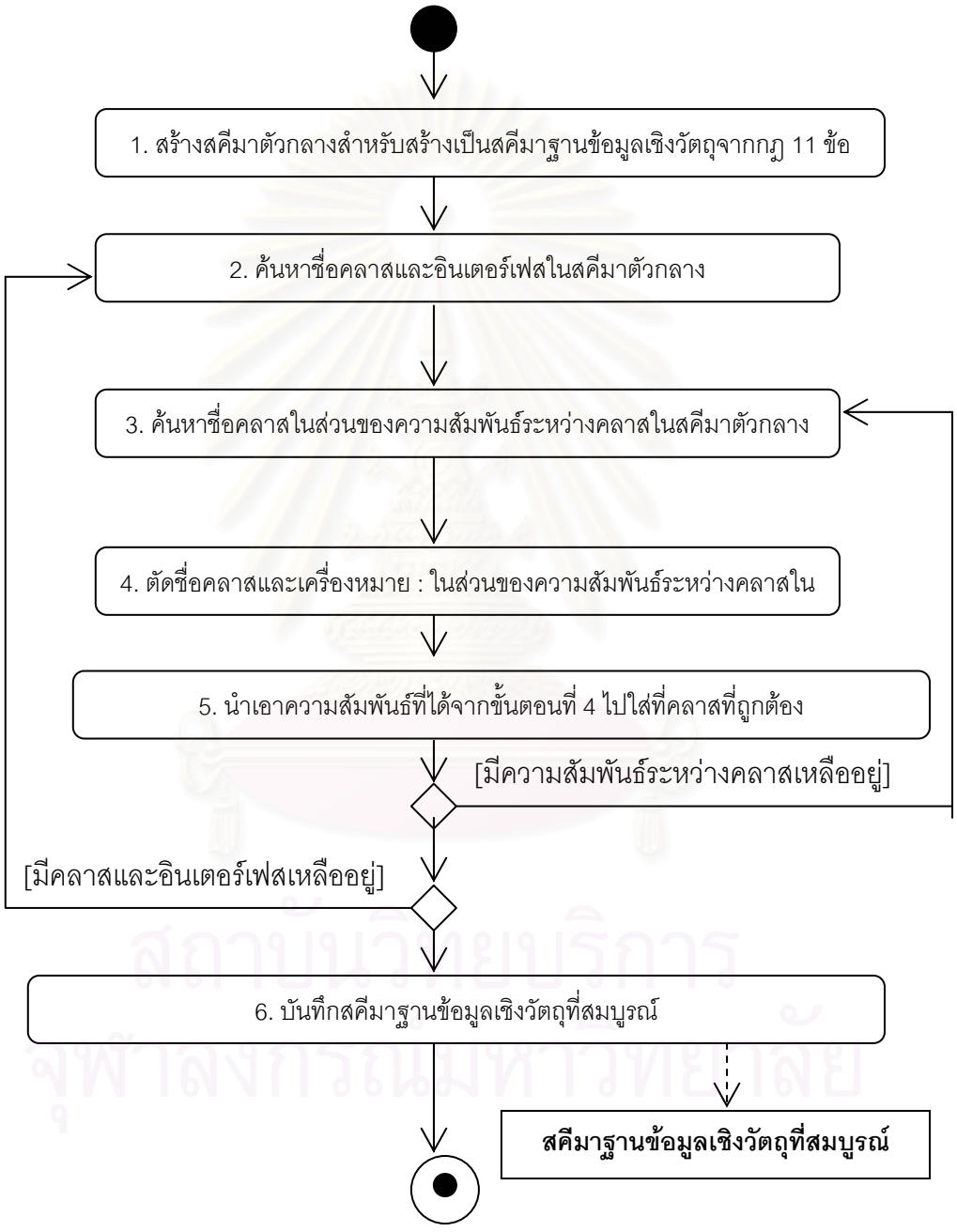
รูปที่ 5.16 ซีเควนซ์ไดอะแกรมแสดงการทำงานของกฎข้อที่ 11

จากรูปที่ 5.6-5.16 จะนำเสนอซีเควนซ์ไดอะแกรมแสดงการทำงานของกฎข้อที่ 1–ข้อที่ 11 ซึ่งการทำงานของกฎทั้ง 11 ข้อจะมีลักษณะค้ายกัน โดยในกฎข้อที่ 1 และข้อที่ 2 การทำงานของกฎจะเริ่มจากคลาส Generator เรียกใช้เมทธอด getDetail() จากคลาส Method คลาส Attribute คลาส Parameter คลาส Relationship เพื่อเก็บข้อมูลต่างๆ ที่จะนำไปใช้ในการประยุกต์กฎการแปลงยูเอ็มแอลคลาสไดอะแกรมเป็น สคีมาตัวกลางสำหรับสร้างสคีมาฐานข้อมูลเชิงวัตถุ โดยภายในคลาสแต่ละคลาสจะเรียกเมทธอดของตัวเองเพื่อกำหนดรายละเอียดต่าง ๆ ที่มีเพื่อส่งกลับไปให้คลาส Generator โดยเมทธอดที่ใช้กำหนดรายละเอียดของคลาสเช่น setAttribute() setMethod() setParametor() เป็นต้น ซึ่งเมื่อได้รายละเอียดครบ คลาส Generator จะเรียกใช้เมทธอด convertToXML() เพื่อกำหนดรายละเอียดของข้อมูลที่ได้ให้อยู่ในรูปของเอ็กซ์เอ็มแอล จากนั้นจะเรียกเมทธอดเพื่อบันทึกข้อมูลที่ได้ และจะเรียกเมทธอด genOodbSchema() เพื่อสร้างเป็นสคีมาของฐานข้อมูลเชิงวัตถุแต่ละชนิด โดยภายในเมทธอดดังกล่าวจะทำการแปลงสคีมาตัวกลางที่ได้จากกฎทั้ง 11 ข้อเป็นสคีมาฐานข้อมูลเชิงวัตถุ (โดยรายละเอียดของการแปลงจากสคีมาตัวกลางที่ได้จากกฎทั้ง 11 ข้อเป็นสคีมาฐานข้อมูลเชิงวัตถุจะอธิบายในหัวข้อต่อไป) หลังจากนั้นจะเรียกเมทธอด saveSchema() เพื่อบันทึกข้อมูลของสคีมาของฐานข้อมูลเชิงวัตถุที่สมบูรณ์ที่ได้ และเมื่อต้องการรายละเอียดของสคีมาจะเรียกใช้เมทธอด getOodbSchema()

ในส่วนของกฎข้อที่ 3 - ข้อที่ 11 จะมีการทำงานลักษณะเดียวกับกฎข้อที่ 1 และ 2 จะมีส่วนที่แตกต่างคือจะเพิ่มคลาสขึ้นมาแต่ละกฎ โดยเป็นคลาสที่เกี่ยวกับความสัมพันธ์ต่างๆ ที่กฎข้อ

นั้น ต้องใช้ เช่น กฎข้อที่ 3 จะใช้คลาส Association เพื่อกำหนดรายละเอียดของความสัมพันธ์ที่มีอยู่ และส่งรายละเอียดให้คลาส Generator ต่อไปเพื่อสร้างเป็นเอ็กซ์เอ็มแอลและสคีมาฐานข้อมูลเชิงวัตถุต่อไป คลาสที่เพิ่มจะปรากฏในแต่ละกฎของกฎข้อที่ 3– ข้อที่ 11 ดังรูปที่ 5.8- 5.16

5.6 ขั้นตอนการแปลงสคีมาตัวกลางที่ได้จากกฎทั้ง 11 ข้อเป็นสคีมาฐานข้อมูลเชิงวัตถุ

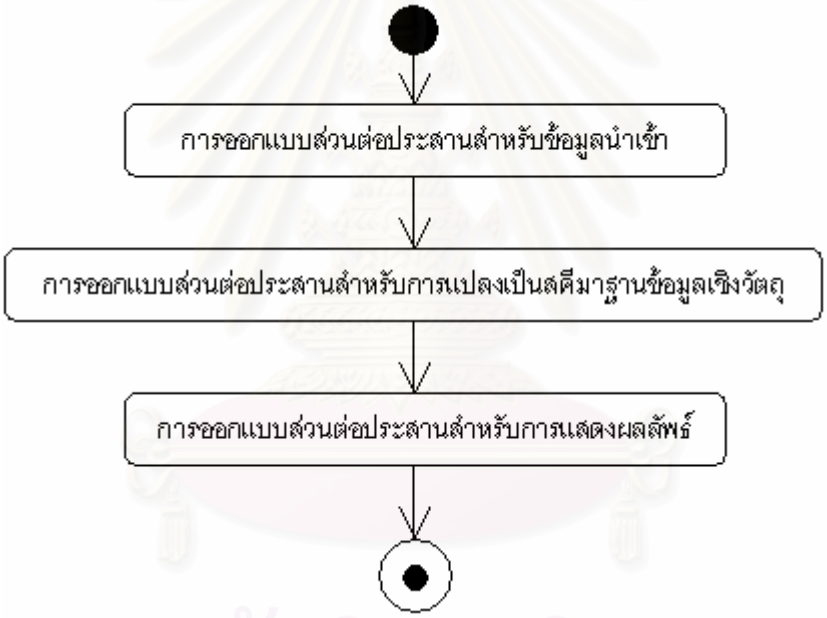


รูปที่ 5.17 ขั้นตอนการแปลงสคีมาตัวกลางที่ได้จากกฎทั้ง 11 ข้อเป็นสคีมาฐานข้อมูลเชิงวัตถุ

จากรูปที่ 5.17 จะแสดงขั้นตอนการแปลงสคีมาตัวกลางเพื่อสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุที่สมบูรณ์ โดยการทำงานจะเริ่มจากค้นหาชื่อคลาสและอินเตอร์เฟสในสคีมาตัวกลาง จากนั้นค้นหาชื่อคลาสในส่วนของความสัมพันธ์ระหว่างคลาสในสคีมาตัวกลาง เมื่อได้ชื่อคลาสและเครื่องหมาย : ให้ตัดชื่อคลาสและเครื่องหมาย : ในส่วนของความสัมพันธ์ระหว่างคลาสออกจากนั้นนำเอาความสัมพันธ์ที่ได้ ไปใส่ที่คลาสที่ถูกต้องและบันทึกสคีมาฐานข้อมูลเชิงวัตถุที่สมบูรณ์ต่อไป

5.7 การออกแบบส่วนต่อประสานของเครื่องมือ

การออกแบบส่วนต่อประสานของเครื่องมือที่ประยุกต์ใช้กฎการแปลงคลาสไดอะแกรมเป็นสคีมาฐานข้อมูลเชิงวัตถุจะประกอบไปด้วยกิจกรรมต่าง ๆ ดังนี้



รูปที่ 5.18 การออกแบบส่วนต่อประสานของเครื่องมือที่ประยุกต์ใช้กฎการแปลงคลาสไดอะแกรมเป็นสคีมาฐานข้อมูลเชิงวัตถุ

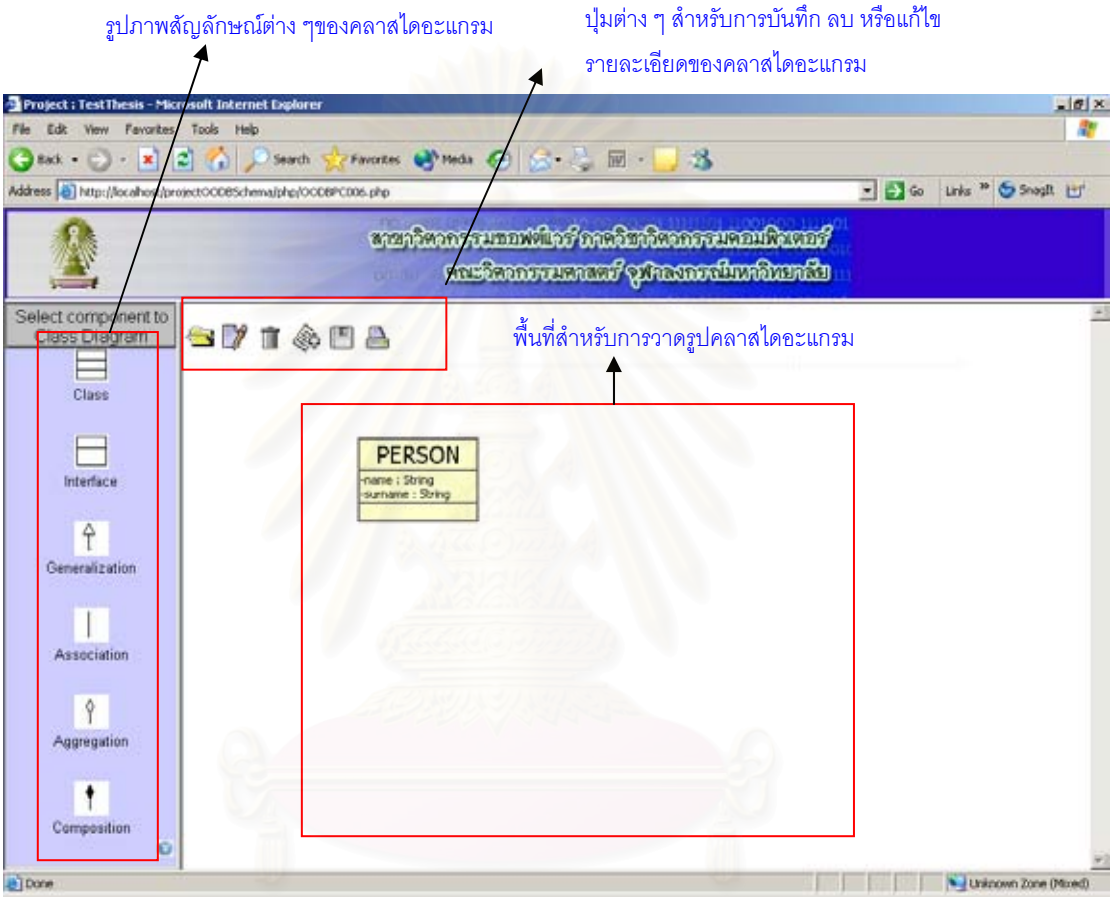
จากรูปที่ 5.18 การออกแบบส่วนต่อประสานทั้งหมดทั้ง 3 ส่วน จะทำงานภายใต้เว็บบราวเซอร์ จากรูปการออกแบบส่วนต่อประสานทั้ง 3 มีรายละเอียดดังนี้

1. การออกแบบส่วนต่อประสานสำหรับข้อมูลนำเข้า เป็นส่วนที่จะรับข้อมูลนำเข้าจากผู้ใช้ จะเป็นการรับข้อมูลที่จะต้องใช้ในการทำการแปลงจากเครื่องมือ มาทำการจัดเตรียมให้อยู่ในรูปของข้อมูลเอ็กซ์เอ็มแอลที่เหมาะสม สำหรับการออกแบบในส่วนนี้จะประกอบไปด้วย

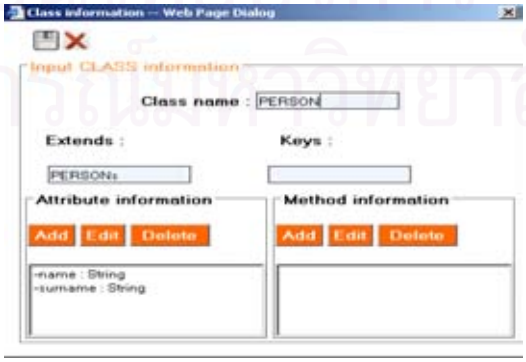
- รูปภาพสัญลักษณ์ต่าง ๆของคลาสไดอะแกรม

- พื้นที่สำหรับการวาดรูปคลาสไดอะแกรม เป็นส่วนที่จะรับข้อมูลนำเข้าของคลาสไดอะแกรมจากผู้ใช้
- ปุ่มต่าง ๆ สำหรับการบันทึก ลบ หรือแก้ไขรายละเอียดของคลาสไดอะแกรม
- หน้าต่างสำหรับเพิ่มข้อมูลเพิ่มเติมของคลาส อินเตอร์เฟส และความสัมพันธ์ต่าง ๆ

ส่วนต่อประสานสำหรับข้อมูลนำเข้าสามารถแสดงได้ดังรูปที่ 5.19 - 5.20



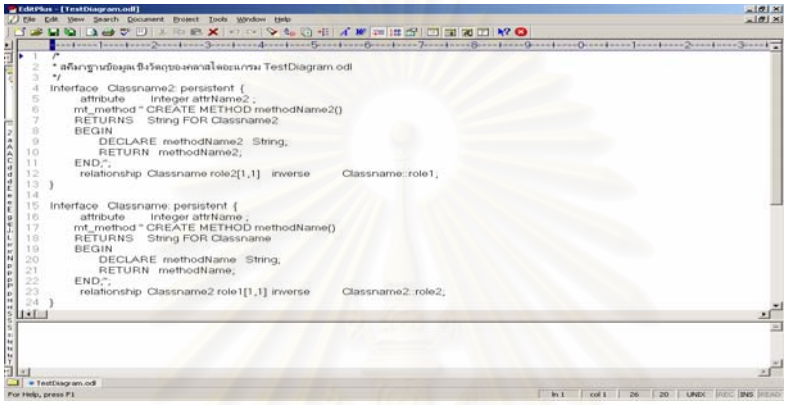
รูปที่ 5.19 การออกแบบส่วนต่อประสานสำหรับข้อมูลนำเข้า



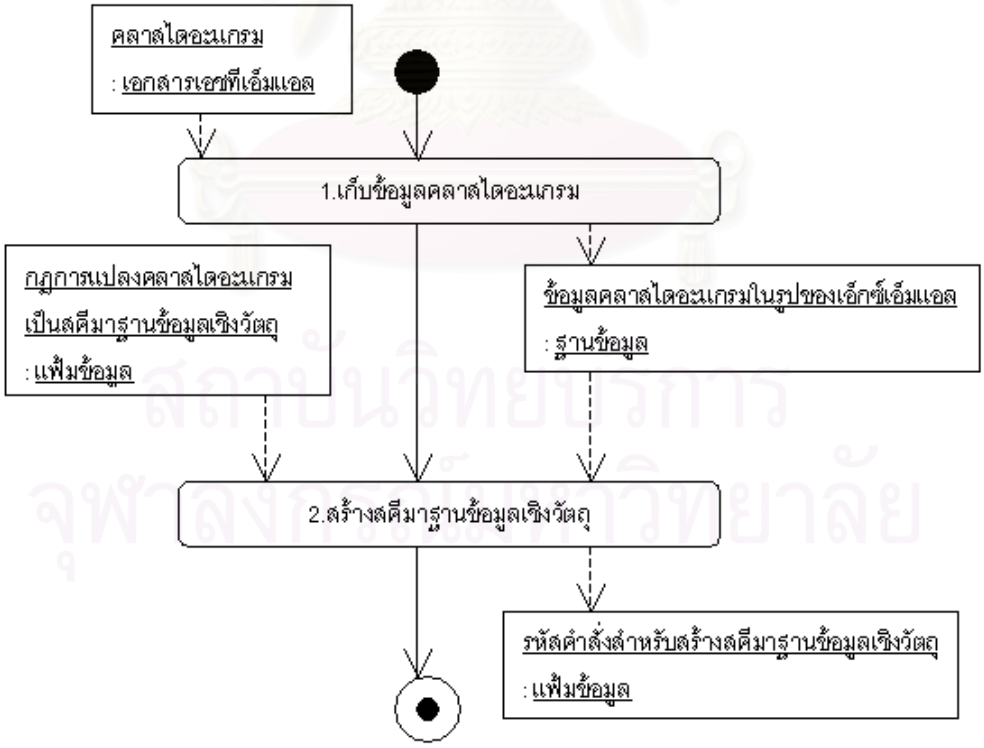
รูปที่ 5.20 การออกแบบส่วนต่อประสานสำหรับข้อมูลนำเข้าส่วนของหน้าต่างสำหรับเพิ่มข้อมูลเพิ่มเติมของคลาส

2. การออกแบบส่วนต่อประสานสำหรับการแปลงเป็นสคีมาฐานข้อมูลเชิงวัตถุ เป็นส่วนที่จะทำหน้าที่สร้างสคีมาฐานข้อมูลเชิงวัตถุที่ได้จากข้อมูลนำเข้าในส่วนแรก โดยการทำงานของการแปลงเป็นสคีมาฐานข้อมูลเชิงวัตถุแสดงดังรูปที่ 5.22 - รูปที่ 5.23

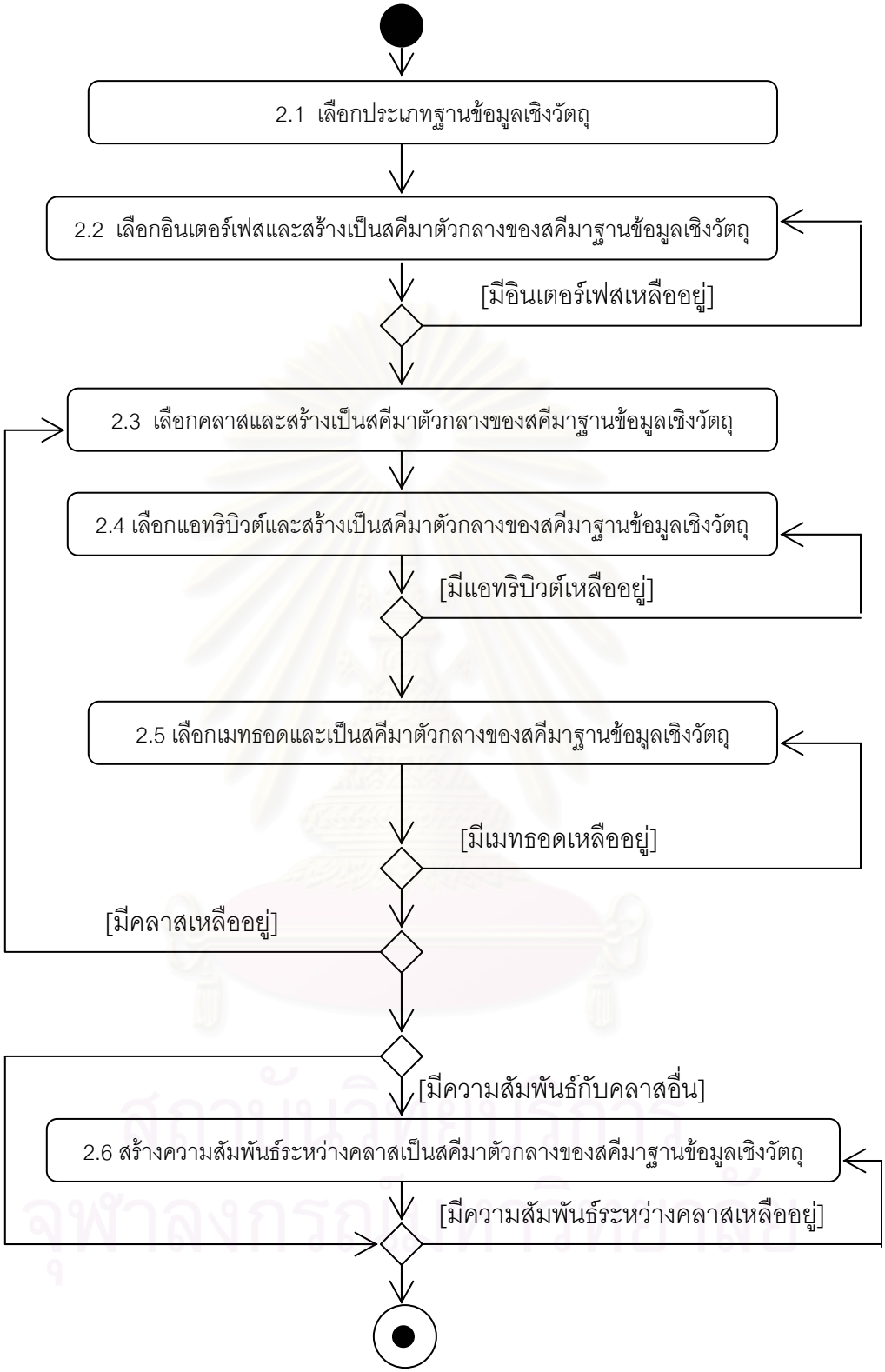
3. การออกแบบส่วนต่อประสานสำหรับการแสดงผลลัพธ์ เป็นส่วนที่ทำหน้าที่แสดงผลลัพธ์ที่ได้จากการแปลง ผลลัพธ์ที่ได้จะเป็นสคีมาฐานข้อมูลเชิงวัตถุที่อยู่ในรูปของไฟล์นามสกุล .odl ผู้ใช้สามารถนำไปสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุต่อไปได้แสดงดังรูปที่ 5.21



รูปที่ 5.21 การออกแบบส่วนต่อประสานสำหรับการแสดงผลลัพธ์



รูปที่ 5.22 การออกแบบส่วนต่อประสานสำหรับการแปลงเป็นสคีมาฐานข้อมูลเชิงวัตถุ



รูปที่ 5.23 การแปลงคลาสเป็นสคีมาตัวกลางของสคีมาฐานข้อมูลเชิงวัตถุ

5.8 การออกแบบโครงสร้างข้อมูลเอ็กซ์เอ็มแอลที่ใช้ในเครื่องมือ

การออกแบบโครงสร้างข้อมูลเอ็กซ์เอ็มแอลที่ใช้ในเครื่องมือสำหรับการแปลงคลาสไดอะแกรมเป็นสคีมาฐานข้อมูลเชิงวัตถุจะมีรายละเอียดตามที่ได้อธิบายไว้ในบทที่ 3 สรุปโครงสร้างข้อมูลเอ็กซ์เอ็มแอลที่ออกแบบสำหรับเครื่องมือมีดังนี้

```
<oodb_schema>
      <class [interface] ComponentID=" "  CPNname=" "  Cvisibility=" "  Cxcenter =" " Cycenter =" "
CleftPos=" " CtopPos=" "  Cextent="">
                  <attribute CPNname=" "   Visibility="  "  Type="  "  Initialvalue ="  " ></attribute>
                  <method CPNname=" "   Visibility="  "   RetType="  "></method>
       </class[/interface]>
      <relationship>
          <association ComponentID=" " SourcePos="  " SourceID="  " SourceType="  " DestID="  "
DestType=" " X1=" " Y1=" " Xstart=" " Ystart=" " X2=" " Y2=" " RolA="  " RolB="  " MultiA="  " MultiB="  "
XrolA="  " XrolB="  " XmultiA="  " XmultiB="  "  YrolA="  " YrolB="  " YmultiA="  " ymultiB="  " >
          </association>
          <associationclass ComponentID=" " SourcePos="  " SourceID="  " SourceType="  " DestID="  "
DestType=" " X1=" " Y1=" " Xstart=" " Ystart=" " X2=" " Y2=" " RolA="  " RolB="  " MultiA="  " MultiB="  "
XrolA="  " XrolB="  " XmultiA="  " XmultiB="  "  YrolA="  " YrolB="  " YmultiA="  " ymultiB="  " >
          </associationclass >
          <recursiveassociation  ComponentID=" " SourcePos="  " SourceID="  " SourceType="  "
DestID="  " DestType=" " X1=" " Y1=" " Xstart=" " Ystart=" " X2=" " Y2=" " RolA="  " RolB="  " MultiA="  "
MultiB="  " XrolA="  " XrolB="  " XmultiA="  " XmultiB="  "  YrolA="  " YrolB="  " YmultiA="  " ymultiB="  " >
          </recursiveassociation >
          <dependency  ComponentID=" " SourcePos="  " SourceID="  " SourceType="  " DestID="  "
DestType=" " X1=" " Y1=" " Xstart=" " Ystart=" " X2=" " Y2=" " ></dependency >
          <aggregation  ComponentID=" " SourcePos="  " SourceID="  " SourceType="  " DestID="  "
DestType=" " X1=" " Y1=" " Xstart=" " Ystart=" " X2=" " Y2=" " RolA="  " RolB="  " MultiA="  " MultiB="  "
XrolA="  " XrolB="  " XmultiA="  " XmultiB="  "  YrolA="  " YrolB="  " YmultiA="  " ymultiB="  " >
          </aggregation>
          <composition ComponentID=" " SourcePos="  " SourceID="  " SourceType="  " DestID="  "
DestType=" " X1=" " Y1=" " Xstart=" " Ystart=" " X2=" " Y2=" " RolA="  " RolB="  " MultiA="  " MultiB="  "
XrolA="  " XrolB="  " XmultiA="  " XmultiB="  "  YrolA="  " YrolB="  " YmultiA="  " ymultiB="  " >
          </composition>
          <qualifiedassociation  ComponentID=" " SourcePos="  " SourceID="  " SourceType="  "
DestID="  " DestType=" " X1=" " Y1=" " Xstart=" " Ystart=" " X2=" " Y2=" " RolA="  " RolB="  " MultiA="  "
MultiB="  " XrolA="  " XrolB="  " XmultiA="  " XmultiB="  "  YrolA="  " YrolB="  " YmultiA="  " ymultiB="  " >
          </qualifiedassociation >
          <generalization ComponentID=" " SourcePos=" " SourceID =" " SourceType="  " DestId =" "
DestType=" " X1=" " Y1=" " Xstart=" " Ystart=" " X2="  " Y2="  "  Xstop=" " Ystop=" " > </generalization>
          <realization ComponentID=" " SourcePos=" " SourceID =" " SourceType="  " DestId =" "
DestType=" " X1=" " Y1=" " Xstart=" " Ystart=" " X2="  " Y2="  "  Xstop=" " Ystop=" " > </ realization >
      </relationship>
</oodb_schema>
```

รูปที่ 5.24 การออกแบบโครงสร้างข้อมูลเอ็กซ์เอ็มแอลที่ใช้ในเครื่องมือ

5.9 **เครื่องมือที่ใช้ในการพัฒนา**

ในการพัฒนาเครื่องมือที่ประยุกต์ใช้กฎการแปลงคลาสไดอะแกรมเป็นสคีมาฐานข้อมูลเชิงวัตถุ ผู้วิจัยได้ใช้เครื่องมือและภาษาต่อไปนี้

1. ดรีมวีฟเวอร์ (Dreamweaver) สำหรับการสร้างส่วนต่อประสานกับผู้ใช้เครื่องมือ
2. ภาษาพีเอชพี(PHP)และภาษาจาวาสคริปต์(Javascript) เป็นภาษาที่ใช้ในการเขียนข้อความคำสั่งที่ประยุกต์กฎการแปลงที่ออกแบบไว้แล้วให้เป็นชุดคำสั่งที่สมบูรณ์
3. อปาเช่เซิร์ฟเวอร์(Apach Server) สำหรับทำหน้าที่เป็นเว็บเซอร์เวอร์ และดาต้าเบสเซิร์ฟเวอร์ในขณะที่ผู้ใช้ใช้งาน
4. ฐานข้อมูลเชิงวัตถุแมทิส และฐานข้อมูลเชิงสวัตถุคาเช่ เป็นฐานข้อมูลที่ใช้สำหรับการสร้างสคีมาฐานข้อมูลเชิงวัตถุที่ได้จากเครื่องมือ
5. เอ็กซ์เอ็มแอลสปาย(XML Spy) สำหรับการตรวจสอบโครงสร้างข้อมูลคลาสไดอะแกรมบนเอกสารเอ็ชทีเอ็มแอลหลังจากที่แปลงเป็นเอ็กซ์เอ็มแอลเรียบร้อย

เหตุผลของการใช้ภาษาพีเอชพี และ ดรีมวีฟเวอร์ เป็นภาษาและเครื่องมือที่ใช้พัฒนาเครื่องมือที่ประยุกต์ใช้กฎการแปลงคลาสไดอะแกรมเป็นสคีมาฐานข้อมูลเชิงวัตถุ เนื่องจากการเขียนโปรแกรมด้วยส่วนประกอบต่าง ๆ เช่น เท็กซ์บ็อกซ์ สามารถมองเห็นทันทีในขณะเขียนโปรแกรม ทำให้สะดวกและรวดเร็วกว่า

การพัฒนาเริ่มจากการออกแบบส่วนต่อประสานดังรูปที่ 5.19-5.20 หลังจากนั้นจะเพิ่มคำสั่งที่ประยุกต์จากกฎที่สร้างขึ้นมา เพื่อให้โปรแกรมทำงานได้ตามต้องการ หาข้อผิดพลาดและแก้ไขต่อไป

รายละเอียดของการใช้เครื่องมือจะแสดงในภาคผนวก จ

บทที่ 6

การทดสอบและการประเมินผลการออกแบบกฎการแปลงคลาสไดอะแกรมเป็นสคีมาฐานข้อมูลเชิงวัตถุ

ในบทนี้จะอธิบายรายละเอียดของการทดสอบและการประเมินผลการออกแบบกฎการแปลงยูเอ็มแอลคลาสไดอะแกรมเป็นสคีมาฐานข้อมูลเชิงวัตถุที่ได้อธิบายไว้ในบทที่ 3 โดยจะประยุกต์ใช้กฎต่าง ๆ ที่ได้อธิบายไว้นำมาทดสอบกับคลาสไดอะแกรมที่เขียนถูกต้องตามมาตรฐานของยูเอ็มแอลโดยจะนำเสนอการประยุกต์ใช้กฎกับฐานข้อมูลทั้ง 3 ชนิด โดยคลาสไดอะแกรมที่ใช้จะใช้คลาสไดอะแกรมชุดเดียวกันทั้งหมดทั้ง 3 กรณี การทดสอบจะมี 3 การทดสอบคือ

การทดสอบที่ 1 การแปลงคลาสไดอะแกรมโดยการประยุกต์ใช้กฎทั้ง 11 ข้อ ของฐานข้อมูลทั้ง 3 ชนิด โดยใช้เครื่องมือที่สร้างเป็นตัวแปลงคลาสไดอะแกรมที่สร้างขึ้น

การทดสอบที่ 2 การแปลงคลาสไดอะแกรมโดยการประยุกต์ใช้กฎทั้ง 11 ข้อ ของฐานข้อมูลทั้ง 3 ชนิด โดยการประยุกต์ใช้กฎโดยการแปลงด้วยตัวเอง

การทดสอบที่ 3 การทดสอบใช้คำสั่งเพื่อดึงข้อมูลจากฐานข้อมูลที่สร้างขึ้นเพื่อตรวจสอบความถูกต้องของฐานข้อมูล

รายละเอียดของผลการทดสอบ และผลการวิเคราะห์ที่ได้จะอธิบายในลำดับถัดไป

รายละเอียดของเนื้อหาในบทนี้จะแบ่งออกเป็น 2 ส่วน คือ

1. การทดสอบการแปลงคลาสไดอะแกรมเป็นสคีมาฐานข้อมูลเชิงวัตถุของกฎทั้ง 11 ข้อ ของฐานข้อมูลทั้ง 3 ชนิด

2. การประเมินผลและวิเคราะห์ผลที่ได้จากการทดสอบการแปลงคลาสไดอะแกรมเป็นสคีมาฐานข้อมูลเชิงวัตถุ

สคีมาฐานข้อมูลเชิงวัตถุที่ปรากฏในบทนี้จะเป็นสคีมาฐานข้อมูลเชิงวัตถุที่สมบูรณ์ที่ผ่านการแปลงจากสคีมาตัวกลางเรียบร้อยแล้ว

6.1. การทดสอบการแปลงคลาสไดอะแกรมเป็นสคีมาฐานข้อมูลเชิงวัตถุของฐานข้อมูลทั้ง 3 ชนิด

ในส่วนของการทดสอบกฎการแปลงคลาสไดอะแกรมเป็นสคีมาฐานข้อมูลเชิงวัตถุจะมีรายละเอียดดังนี้

หลังจากผู้วิเคราะห์ได้วิเคราะห์ระบบการจัดการข้อมูลโครงการภายในบริษัท จะเขียนเป็นข้อกำหนดความต้องการซอฟต์แวร์(Software requirements specification) อย่างย่อดังรูปที่ 6.1

ระบบการจัดการข้อมูลโครงการภายในบริษัท เป็นระบบที่อธิบายถึงรายละเอียดของโครงการที่รับผิดชอบโดยแผนกต่าง ๆภายในบริษัท

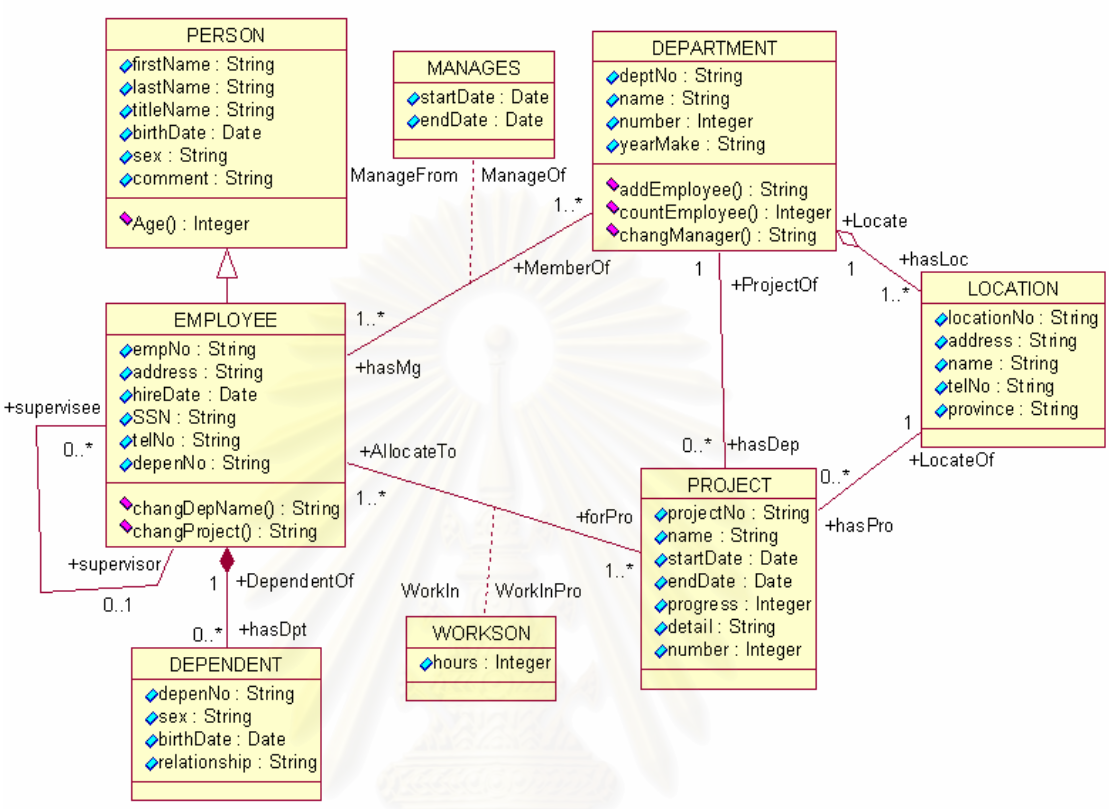
โครงการที่ถูกสร้างขึ้นจะมีข้อมูลวันเริ่มต้นและสิ้นสุดโครงการและจะมีชื่อและรหัสของโครงการ รวมถึงรายละเอียดความก้าวหน้าของการทำโครงการ และรายละเอียด ของโครงการ โครงการจะอยู่ภายใต้แผนกที่กำหนดภายในบริษัท ในโครงการสามารถเพิ่มหรือลดพนักงานเข้ามาภายในโครงการได้ โครงการจะต้องอยู่ภายในสถานที่ที่แผนกตั้งอยู่ โดยสถานที่ 1 ที่จะมีโครงการได้มากว่า 1 โครงการ และโครงการ 1 โครงการจะมีสถานที่ที่อยู่ของโครงการได้เพียง 1 สถานที่ โครงการสามารถที่จะเพิ่มโครงการย่อย ๆ ภายในโครงการนั้น ๆได้ และสามารถเปลี่ยนตัวผู้จัดการของโครงการได้ และผู้จัดการโครงการจะต้องเป็นผู้จัดของแผนกที่โครงการนั้นอยู่

แผนกสามารถเพิ่มพนักงานเข้ามาภายในแผนกได้ และสามารถเปลี่ยนผู้จัดการแผนกได้ และแผนกจะมีข้อมูลที่ตั้งของแผนก โดยแผนก 1 แผนกจะมีข้อมูลที่ตั้งของแผนกโดยแผนก 1 แผนกสามารถมีที่ตั้งได้หลายสถานที่ และสถานที่ 1 สถานที่สามารถมีแผนกได้มากว่า 1 แผนก

พนักงานจะทำงานภายในโครงการและแผนกที่กำหนด และสามารถเปลี่ยนโครงการและแผนกที่สังกัดอยู่ได้ พนักงาน 1 คนจะมีหัวหน้าได้ 1 คน และหัวหน้า 1 คนสามารถมีพนักงานได้มากกว่า 1 คนพนักงาน 1 คนจะต้องมีบุคคลค้ำประกันอย่างน้อย 1 คน การทำงานภายใต้โครงการของพนักงานจะมีจำนวนชั่วโมงบอกถึงระยะเวลาของการทำโครงการ

รูปที่ 6.1 ข้อกำหนดความต้องการซอฟต์แวร์ระบบการจัดการข้อมูลโครงการภายในบริษัท

จากนั้นนำข้อกำหนดความต้องการซอฟต์แวร์ที่ได้ดังรูปที่ 6.1 สร้างเป็นคลาสไดอะแกรม ได้ดังรูปที่ 6.2

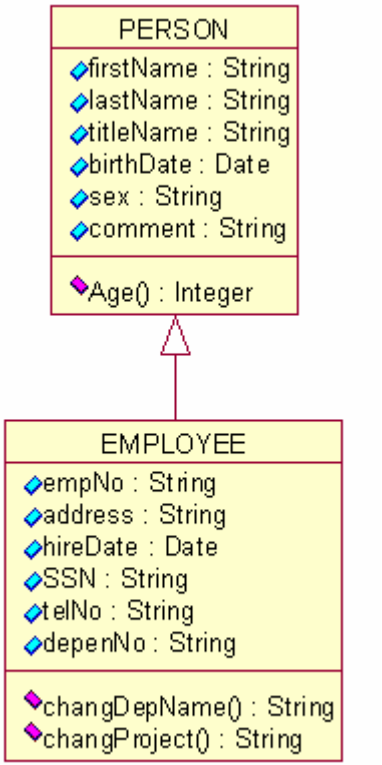


รูปที่ 6.2 แสดงคลาสไดอะแกรมระบบการจัดการข้อมูลโครงการภายในบริษัท[10]

จากคลาสไดอะแกรมระบบการจัดการข้อมูลโครงการภายในบริษัทรูปที่ 6.2 แยกเป็นส่วนประกอบย่อยเพื่อประยุกต์ใช้กฎได้ดังตารางที่ 6.1- ตารางที่ 6.8

ตารางที่ 6.1 ส่วนประกอบย่อยคลาสไดอะแกรมระบบการจัดการข้อมูลโครงการภายในบริษัทส่วนที่ 1

สัญลักษณ์ในคลาสไดอะแกรม	สคีมาฐานข้อมูลของโอดีเอ็มจี	สคีมาของฐานข้อมูลคาเช่	สคีมาของฐานข้อมูลแมทิส	กฎที่ใช้

	Class PERSON
	(extent PERSONs)
	{
	attribute string firstName;
	attribute string lastName;
	attribute string titleName;
	attribute date birthDate;
	attribute string sex ;
	attribute string comment;
	integer Age():
	};
		Class Test.Person Extends
		%Persistent [ClassType =
		persistent, ProcedureBlock]
		{
		Property firstName As %String;
		Property lastName As %String;
		Property titleName As %String;
		Property birthDate As %Date;
		Property sex As %String;
		Property comment As %String;
		Method Age() As %Integer{}
		};
			interface PERSON: persistent
			{
			attribute String firstName;
			attribute String lastName;
			attribute String titleName;
			attribute Date birthDate;
			attribute String sex ;
			attribute String comment;
			mt_method
			"CREATE METHOD Age()
			RETURNS Integer FOR
			PERSON
			BEGIN
			DECLARE Age Integer;
			RETURN Age;;END;" ;
			};
				■ กฎข้อ1
				■ กฎข้อ2
				☐ กฎข้อ3
				☐ กฎข้อ4
				☐ กฎข้อ5
				☐ กฎข้อ6
				☐ กฎข้อ7
				☐ กฎข้อ8
				☐ กฎข้อ9
				■ กฎข้อ10
				☐ กฎข้อ11

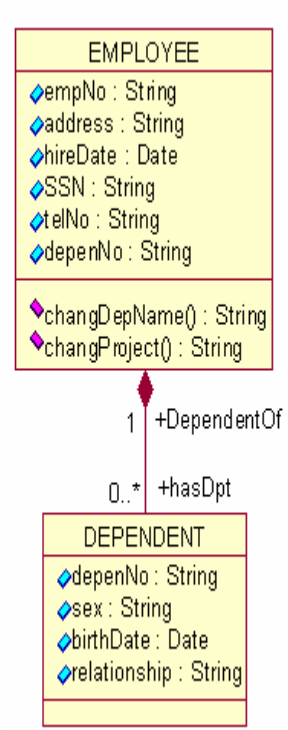
ตารางที่ 6.1 ส่วนประกอบย่อยคลาสไดอะแกรมระบบการจัดการข้อมูลโครงการภายในบริษัทส่วนที่ 1(ต่อ)

สัญลักษณ์ในคลาสไดอะแกรม	สคีมาฐานข้อมูลของโอดีเอ็มจี	สคีมาของฐานข้อมูลคาเช่	สคีมาของฐานข้อมูลแมทิส	กฎที่ใช้

	class EMPLOYEE
	extends PERSON
	(extent EMPLOYEEs)
	{
	attribute string empNo;
	attribute string address;
	attribute date hireDate;
	attribute string SSN;
	attribute string telNo;
	attribute string depenNo;
	string changDepName ():
	string changProject ():
	relationship EMPLOYEE
	supervisor inverse
	EMPLOYEE:: supervisee;
	relationship
	set<EMPLOYEE>
	supervisee inverse
	EMPLOYEE:: supervisor;
	}
		Class Test.EMPLOYEE Extends
		Person [ClassType = persistent,
		ProcedureBlock]
		{
		Property empNo As %String;
		Property address As %String;
		Property hireDate As %Date;
		Property SSN As %String;
		Property telNo As %String;
		Property depenNo As %String;
		Method changDepName()
		As %String{}
		Method changPoject()
		As %String{}
		Relationship supervisor As
		EMPLOYEE [Cardinality = one,
		Inverse = supervisee];
		Relationship supervisee As
		EMPLOYEE[Cardinality = many,
		Inverse = supervisor];
		}
			interface EMPLOYEE:PERSON:
			persistent
			{ attribute String empNo;
			attribute String address;
			attribute Date hireDate;
			attribute String SSN;
			attribute String telNo;
			attribute String depenNo;
			mt_method
			"CREATE METHOD
			changDepName() RETURNS
			String FOR EMPLOYEE
			BEGIN
			DECLARE
			changDepName String;
			RETURN changDepName;
			END;";
			mt_method
			"CREATE METHOD
			changProject() RETURNS
			String FOR EMPLOYEE
			BEGIN
			DECLARE
			changProject String;
			RETURN changProject;
			END;";
			relationship EMPLOYEE
			supervisor[0, 1] inverse
			EMPLOYEE::supervisee;
			relationship Set<EMPLOYEE>
			supervisee inverse
			EMPLOYEE::supervisor;};	

ตารางที่ 6.2 ส่วนประกอบย่อยคลาสไดอะแกรมระบบการจัดการข้อมูลโครงการภายในบริษัทส่วนที่ 2

สัญลักษณ์ในคลาสไดอะแกรม	สคีมาฐานข้อมูลของโอดีเอ็มจี	สคีมาของฐานข้อมูลคาเช่	สคีมาของฐานข้อมูลแมทิส	กฎที่ใช้

	class EMPLOYEE extends PERSON
	(extent EMPLOYEEs)
	{
	attribute string empNo;
	attribute string address;
	attribute date hireDate;
	attribute string SSN;
	attribute string telNo;
	attribute string depenNo;
	attribute DEPENDENT dependent;
	string changDepName ():
	string changProject ():
	relationship DEPENDENT
	hasDpt inverse
	DEPENDENT :: dependentOf;
	relationship EMPLOYEE
	supervisor inverse
	EMPLOYEE:: supervisee;
	relationship set<EMPLOYEE>
	supervisee inverse
	EMPLOYEE:: supervisor;
	}
		Class Package.EMPLOYEE Extends
		Person[ClassType = persistent,
		ProcedureBlock]
		{
		Property empNo As %String;
		Property address As %String;
		Property hireDate As %Date;
		Property SSN As %String;
		Property telNo As %String;
		Property depenNo As %String;
		Property dependent
		As %DEPENDENT;
		Method changDepName()
		As %String{}
		Method changPoject()
		As %String{}
		Relationship hasDpt As
		DEPENDENT [Cardinality =
		one,
		Inverse = DepentdentOf];
		Relationship supervisor As
		EMPLOYEE [Cardinality = one,
		Inverse = supervisee];
		Relationship supervisee As
		EMPLOYEE [Cardinality =
		many, Inverse = supervisor];
		}
			interface EMPLOYEE:PERSON:
			persistent
			{ attribute String empNo;
			attribute String address;
			attribute Date hireDate;
			attribute String SSN;
			attribute String telNo;
			attribute String depenNo;
			attribute DEPENDENT dependent;
			mt_method"CREATE METHOD
			changDepName() RETURNS
			String FOR EMPLOYEE
			BEGIN DECLARE
			changDepName String;
			RETURN changDepName; END;";
			mt_method"CREATE METHOD
			changProject() RETURNS
			String FOR EMPLOYEE
			BEGIN DECLARE
			changProject String;
			RETURN changProject; END;";
			relationship EMPLOYEE
			supervisor[0, 1] inverse
			EMPLOYEE::supervisee;
			relationship Set<EMPLOYEE>
			supervisee inverse
			EMPLOYEE::supervisor;
			relationship Set<WORKSON>
			worksIn [1,-1] inverse
			WORKSON::allocateTo;
			relationship Set<DEPENDENT>
			hasDpt inverse
			DEPENDENT::dependentOf;};
				■ กฎข้อ1
				■ กฎข้อ2
				☐ กฎข้อ3
				☐ กฎข้อ4
				☐ กฎข้อ5
				☐ กฎข้อ6
				☐ กฎข้อ7
				■ กฎข้อ8
				☐ กฎข้อ9
				☐ กฎข้อ10
				☐ กฎข้อ11

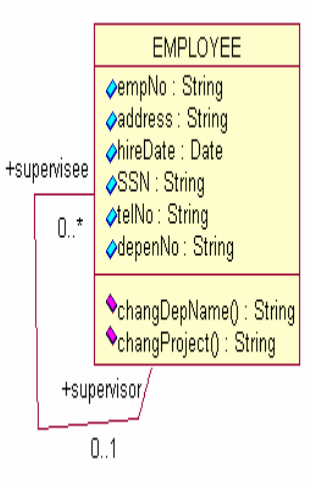
ตารางที่ 6.2 ส่วนประกอบย่อยคลาสไดอะแกรมระบบการจัดการข้อมูลโครงการภายในบริษัทส่วนที่ 2(ต่อ)

สัญลักษณ์ในคลาสไดอะแกรม	สคีมาฐานข้อมูลของโอดีเอ็มจี	สคีมาของฐานข้อมูลคาเช่	สคีมาของฐานข้อมูลแมทิส	กฎที่ใช้

	class DEPENDENT
	(extent DEPENDENTs)
	{
	attribute string depenNo;
	attribute date birthDate;
	attribute string sex ;
	attribute string relationship;
	relationship EMPLOYEE
	dependentOf inverse
	EMPLOYEE::hasDpt;
	}
		Class Package.DEPENDENT
		Extends %Persistent [ClassType =
		persistent,ProcedureBlock]
		{
		Property depenNo As %String;
		Property birthDate As %Date;
		Property sex As %String;
		Property relationship As %String;
		Relationship DepentdentOf
		As EMPLOYEE
		[Cardinality =one,
		Inverse = hasDpt];
		}
			interface DEPENDENT: persistent
			{
			attribute String depenNo;
			attribute Date birthDate;
			attribute String sex ;
			attribute String relateDpt;
			relationship EMPLOYEE
			dependentOf inverse
			EMPLOYEE::hasDpt;
			};	

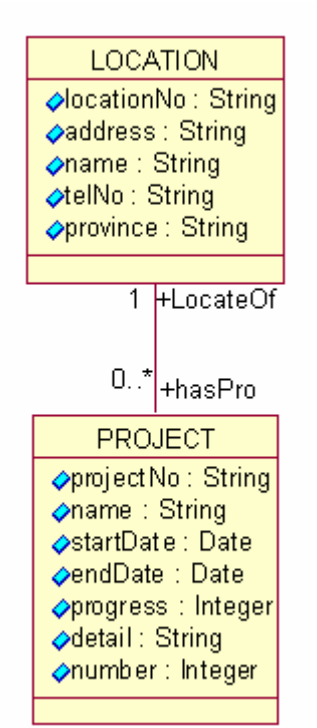
ตารางที่ 6.3 ส่วนประกอบย่อยคลาสไดอะแกรมระบบการจัดการข้อมูลโครงการภายในบริษัทส่วนที่ 3

สัญลักษณ์ในคลาสไดอะแกรม	สคีมาฐานข้อมูลโอดีเอ็มจี	สคีมาของฐานข้อมูลคาเช่	สคีมาของฐานข้อมูลแมทิส	กฎที่ใช้

	class EMPLOYEE extends PERSON(extent EMPLOYEEs)
	{
	attribute string empNo;
	attribute string address;
	attribute string hireDate;
	attribute date SSN;
	attribute string telNo;
	attribute string sex;
	attribute string depenNo;
	string changDepName():
	string changDepName():
	relationship EMPLOYEE
	supervisor inverse
	EMPLOYEE:: supervisee;
	relationship set<EMPLOYEE>
	supervisee inverse
	EMPLOYEE:: supervisor;
	
	}
		Class Package.EMPLOYEE Extends Person[ClassType=persistent, ProcedureBlock]
		{
		Property empNo As %String;
		Property address As %String;
		Property hireDate As %Date;
		Property SSN As %String;
		Property telNo As %String;
		Property depenNo As %String;
		Method changDepName()
		As %String{}
		Method changPoject() As %String{}
		Relationship hasDpt As
		DEPENDENT [Cardinality = one,
		Inverse = DepentdentOf];
		Relationship supervisor As
		EMPLOYEE [Cardinality = one,
		Inverse = supervisee];
		Relationship supervisee As
		EMPLOYEE [Cardinality = many,
		Inverse = supervisor];
		Relationship WorksIn As
		WORKSON [Cardinality = many,
		Inverse = AllocateTo];
		Relationship manageFrom As
		MANAGES [Cardinality = many,
		Inverse = hasMg];
		}
			interface EMPLOYEE:PERSON:
			persistent
			{ attribute String empNo;
			attribute String address;
			attribute Date hireDate;
			attribute String depenNo;
			attribute String SSN;
			attribute String telNo;
			mt_method"CREATE METHOD
			changDepName() RETURNS
			String FOR EMPLOYEE BEGIN
			DECLARE changDepName String;
			RETURN changDepName; END;";
			mt_method "CREATE METHOD
			changProject() RETURNS
			String FOR EMPLOYEE
			BEGIN DECLARE
			changProject String;
			RETURN changProject; END;";
			relationship EMPLOYEE
			supervisor[0, 1] inverse
			EMPLOYEE::supervisee;
			relationship Set<EMPLOYEE>
			supervisee inverse
			EMPLOYEE::supervisor;
			relationship Set<WORKSON>
			worksIn [1,-1] inverse
			WORKSON::allocateTo;
			relationship Set<MANAGES>
			manageFrom [1,-1] inverse
			MANAGES::hasMg;
			relationship Set<DEPENDENT>
			hasDpt inverse
			DEPENDENT::dependentOf; };
				■ กฎข้อ1
				■ กฎข้อ2
				☐ กฎข้อ3
				☐ กฎข้อ4
				■ กฎข้อ5
				☐ กฎข้อ6
				☐ กฎข้อ7
				☐ กฎข้อ8
				☐ กฎข้อ9
				☐ กฎข้อ10
				☐ กฎข้อ11

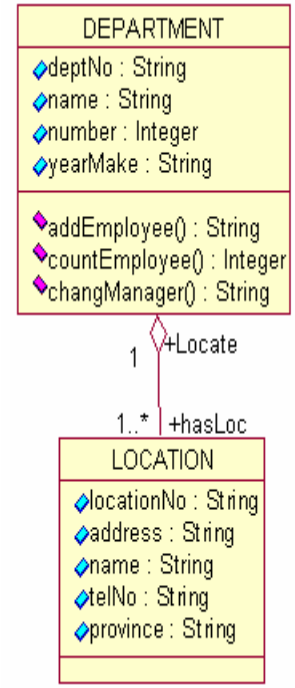
ตารางที่ 6.4 ส่วนประกอบย่อยคลาสไดอะแกรมระบบการจัดการข้อมูลโครงการภายในบริษัทส่วนที่ 4

สัญลักษณ์ในคลาสไดอะแกรม	สคีมาฐานข้อมูลของโอดีเอ็มจี	สคีมาของฐานข้อมูลคาเช่	สคีมาของฐานข้อมูลแมทิส	กฎที่ใช้

	class LOCATION
	(extent LOCATIONs)
	{
	attribute string locationNo;
	attribute string address;
	attribute string name;
	attribute string telNo;
	attribute string province;
	relationship set<PROJECT>
	hasPro inverse
	PROJECT:: DepentdentOf;
	}
	class PROJECT
	(extent PROJECTs)
	{ attribute string projectNo;
	attribute string name;
	attribute date startdate;
	attribute date endDate;
	attribute integer progress;
	attribute string detail;
	attribute integer number;
	relationship LOCATION
	locateOf inverse
	LOCATION:: hasPro;
	}
		Class Package.PROJECT
		Extends %Persistent [
		ClassType= persistent,
		ProcedureBlock]
		{
		Property projectNo As %String;
		Property name As %String;
		Property startdate As %Date;
		Property endDate As %Date;
		Property progress As %Integer;
		Property detail As %String;
		Property number As %Integer;
		Relationship locateOf As
		LOCATION [Cardinality =
		one, Inverse = hasPro];
		}
		Class Package.LOCATION
		Extends %Persistent [ClassType
		= persistent, ProcedureBlock]
		{
		Property locationNo As %String;
		Property address As %String;
		Property name As %String;
		Property telNo As %String;
		Property province As %String;
		Relationship hasPro As
		PROJECT
		[Cardinality = many,
		Inverse = locateOf];
		}
			interface LOCATION: persistent
			{
			attribute String locationNo;
			attribute String address;
			attribute String name;
			attribute String telNo;
			attribute String province;
			relationship Set<PROJECT>
			hasPro inverse PROJECT::
			locateOf;
			};
			interface PROJECT: persistent
			{
			attribute String projectNo;
			attribute String name;
			attribute Date startDate;
			attribute Date endDate;
			attribute Integer progress;
			attribute String detail;
			attribute Integer number;
			relationship LOCATION
			locateOf inverse
			LOCATION::hasPro;
			};
				■ กฎข้อ1
				■ กฎข้อ2
				■ กฎข้อ3
				☐ กฎข้อ4
				☐ กฎข้อ5
				☐ กฎข้อ6
				☐ กฎข้อ7
				☐ กฎข้อ8
				☐ กฎข้อ9
				☐ กฎข้อ10
				☐ กฎข้อ11

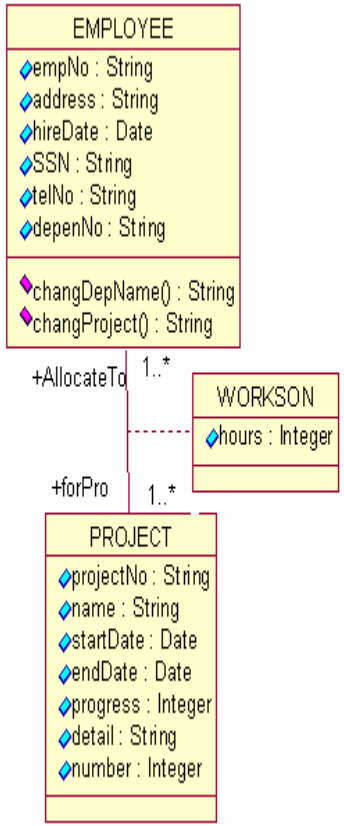
ตารางที่ 6.5 ส่วนประกอบย่อยคลาสไดอะแกรมระบบการจัดการข้อมูลโครงการภายในบริษัทส่วนที่ 5

สัญลักษณ์คลาสไดอะแกรม	สคีมาฐานข้อมูลของโอดีเอ็มจี	สคีมาของฐานข้อมูลคาเช่	สคีมาของฐานข้อมูลแมทิส	กฎที่ใช้

	class DEPARTMENT
	(extent DEPARTMENTs)
	{
	attribute string deptNo;
	attribute string name;
	attribute int number;
	attribute string yearMake;
	attribute LOCATION location;
	string addEmployee ():
	string coutEmployee ():
	string changManger ():
	relationship LOCATION has
	inverse LOCATION::
	Locate;
	}
	class LOCATION
	(extent LOCATIONs)
	{
	attribute string locationNo;
	attribute string address;
	attribute string name;
	attribute string telNo;
	attribute string province;
	relationship
	set< DEPARTMENT >
	Locate inverse
	DEPARTMENT:: has;
	}
		Class Package. DEPARTMENT
		Extends %persistent[ClassType =
		persistent, ProcedureBlock]
		{
		Property deptNo as %string;
		Property name as %string;
		Property number as % int
		Property yearMake as %string;
		Property location
		as %LOCATION ;
		string addEmployee ():
		string coutEmployee ():
		string changManger ():
		Relationship has as LOCATION
		[cardinality=one,
		inverse= Locate];
		}
		Class Package.LOCATION
		Extends persistent[ClassType =
		persistent, ProcedureBlock]
		{
		Property locationNo as %string;
		Property address as %string;
		Property name as %string;
		Property telNo as %string;
		Property province as %string;
		Relationship Locate as
		DEPARTMENT
		[cardinality=one,
		inverse= has];
		}
			interface DEPARTMENT: persistent
			{
			attribute Integer number;
			attribute String name;
			attribute String deptNo;
			attribute String yearMake;
			attribute LOCATION location;
			mt_method"CREATE METHOD
			addEmployee() RETURNS String
			FOR DEPARTMENT BEGIN
			DECLARE addEmployee String;
			RETURN addEmployee; END;";
			mt_method"CREATE METHOD
			coutEmployee() RETURNS String
			FOR DEPARTMENT BEGIN
			DECLARE coutEmployee String;
			RETURN coutEmployee; END;";
			mt_method "CREATE METHOD
			changManger() RETURNS String
			FOR DEPARTMENT BEGIN
			DECLARE changManger String;
			RETURN changManger; END;";
			relationship Set<LOCATION>
			hasLoc[1,-1] inverse LOCATION::
			locate; };
			interface LOCATION: persistent{
			attribute String locationNo;
			attribute String address;
			attribute String province;
			attribute String name;
			attribute String telNo;
			relationship DEPARTMENT locate
			inverse DEPARTMENT::hasLoc;};
				■ กฎข้อ1
				■ กฎข้อ2
				☐ กฎข้อ3
				☐ กฎข้อ4
				☐ กฎข้อ5
				☐ กฎข้อ6
				■ กฎข้อ7
				☐ กฎข้อ8
				☐ กฎข้อ9
				☐ กฎข้อ10
				☐ กฎข้อ11

ตารางที่ 6.6 ส่วนประกอบย่อยคลาสไดอะแกรมระบบการจัดการข้อมูลโครงการภายในบริษัทส่วนที่ 6

สัญลักษณ์ในคลาสไดอะแกรม	สคีมาฐานข้อมูลของโอดีเอ็มจี	สคีมาของฐานข้อมูลคาเช่	สคีมาของฐานข้อมูลแมทิส	กฎที่ใช้

	class PROJECT
	(extent PROJECTs)
	{
	attribute string projectNo;
	attribute string name;
	attribute date startdate;
	attribute date endDate;
	attribute integer progress;
	attribute string detail;
	attribute integer number;
	relationship set<WORKSON>
	worksInPro inverse
	WORKSON:: forPro;
	}
	class WORKSON
	(extent WORKSONs)
	{
	attribute integer hours;
	relationship EMPLOYEE
	AllocateTo inverse
	EMPLOYEE :: worksIn;
	relationship Project forPro
	inverse Project::
	worksInPro;
	}
		Class Test.PROJECT Extends
		%Persistent [ClassType= persistent,
		ProcedureBlock]
		{
		Property projectNo As %String;
		Property name As %String;
		Property startdate As %Date;
		Property endDate As %Date;
		Property progress As %Integer;
		Property detail As %String;
		Property number As %Integer;
		Relationship WorksInPro As
		WORKSON [Cardinality =
		many, Inverse = forPro];
		}
		Class Test.WORKSON Extends
		%Persistent [ClassType =
		persistent,ProcedureBlock]
		{
		Property hours As %Integer;
		Relationship AllocateTo As
		EMPLOYEE [Cardinality =
		one, Inverse = WorksIn];
		Relationship forPro As
		PROJECT [Cardinality =
		one, Inverse = WorksInPro];
		}
			interface PROJECT: persistent
			{
			attribute String projectNo;
			attribute String name;
			attribute Date startDate;
			attribute Date endDate;
			attribute Integer progress;
			attribute String detail;
			attribute Integer number;
			relationshipSet<WORKSON>
			worksInPro[1,-1] inverse
			WORKSON::forPro;
			};
			interface WORKSON: persistent
			{
			attribute Integer hours;
			relationship PROJECT forPro
			inversePROJECT:: worksInPro;
			relationship EMPLOYEE
			allocateTo inverse
			EMPLOYEE::worksIn;
			};
				■ กฎข้อ1
				■ กฎข้อ2
				☐ กฎข้อ3
				■ กฎข้อ4
				☐ กฎข้อ5
				☐ กฎข้อ6
				☐ กฎข้อ7
				☐ กฎข้อ8
				☐ กฎข้อ9
				☐ กฎข้อ10
				☐ กฎข้อ11

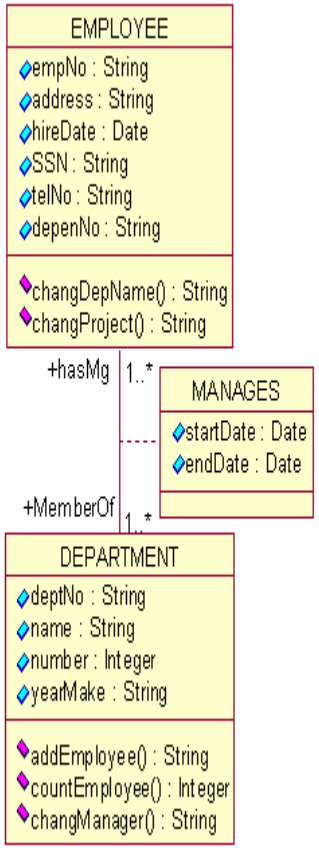
ตารางที่ 6.6 ส่วนประกอบย่อยคลาสไดอะแกรมระบบการจัดการข้อมูลโครงการภายในบริษัทส่วนที่ 6(ต่อ)

สัญลักษณ์ในคลาสไดอะแกรม	สคีมาฐานข้อมูลของโอดีเอ็มจี	สคีมาของฐานข้อมูลคาเช่	สคีมาของฐานข้อมูลแมทิส	กฎที่ใช้

	class EMPLOYEE extends PERSON
	(extent EMPLOYEEs)
	{
	attribute string empNo;
	attribute string address;
	attribute date hireDate;
	attribute string SSN;
	attribute string telNo;
	attribute string depenNo;
	string changDepName ():
	string changProject ():
	relationship EMPLOYEE
	supervisor inverse
	EMPLOYEE:: supervisee;
	relationship set<EMPLOYEE>
	supervisee inverse
	EMPLOYEE:: supervisor;
	relationship set<WORKSON>
	worksIn inverse
	WORKSON:: AllocateTo;
	}
		Class Test.EMPLOYEE Extends
		Person [ClassType =
		persistent,ProcedureBlock]
		{
		Property empNo As %String;
		Property address As %String;
		Property hireDate As %Date;
		Property SSN As %String;
		Property telNo As %String;
		Property depenNo As %String;
		Method changDepName()
		As %String{}
		Method changPoject()
		As %String{}
		Relationship supervisor As
		EMPLOYEE[Cardinality =
		one, Inverse = supervisee];
		Relationship supervisee As
		EMPLOYEE [Cardinality =
		many, Inverse = supervisor];
		Relationship WorksIn As
		WORKSON [Cardinality =
		many, Inverse = AllocateTo];
		}
			Interface EMPLOYEE:PERSON:
			persistent
			{
			attribute String empNo;
			attribute String address;
			attribute Date hireDate;
			attribute String SSN;
			attribute String telNo;
			attribute String depenNo;
			mt_method"CREATE METHOD
			changDepName() RETURNS
			String FOR EMPLOYEE BEGIN
			DECLARE changDepName String;
			RETURN changDepName; END;";
			mt_method"CREATE METHOD
			changProject() RETURNS String
			FOR EMPLOYEE BEGIN
			DECLARE changProject String;
			RETURN changProject; END;";
			relationship EMPLOYEE
			supervisor[0, 1] inverse
			EMPLOYEE::supervisee;
			relationship Set<EMPLOYEE>
			supervisee inverse
			EMPLOYEE::supervisor;
			relationship Set<WORKSON>
			worksIn [1,-1] inverse
			WORKSON::allocateTo;
			};	

ตารางที่ 6.7 ส่วนประกอบย่อยคลาสไดอะแกรมระบบการจัดการข้อมูลโครงการภายในบริษัทส่วนที่ 7

สัญลักษณ์ในคลาสไดอะแกรม	สคีมาฐานข้อมูลของโอดีเอ็มจี	สคีมาของฐานข้อมูลคาเช่	สคีมาของฐานข้อมูลแมทิส	กฎที่ใช้

	class DEPARTMENT
	(extent DEPARTMENTs)
	{
	attribute string deptNo;
	attribute string name;
	attribute integer number;
	attribute string yearMake;
	string addEmployee ():
	string coutEmployee ():
	string changManger ():
	relationship set< MANAGES>
	manageOf inverse
	MANAGES:: MemberOf;
	}
	class MANAGES
	(extent WORKS_ONs)
	{
	attribute date startDate;
	attribute date endDate;
	relationship DEPARTMENT
	MemberOf inverse
	DEPARTMENT:: manageOf;
	relationship EMPLOYEE
	hasMg inverse
	EMPLOYEE::manageFrom;
	}
		Class Package.DEPARTMENT
		Extends %Persistent [ClassType =
		persistent, ProcedureBlock]
		{
		Property deptNo As %String;
		Property name As %String;
		Property number As %Integer;
		Property yearMake As %String;
		Method addEmployee()
		As %String{}
		Method coutEmployee()
		As %String{}
		Method changManger()
		As %String{}
		Relationship ManageOf As
		MANAGES [Cardinality =many,
		Inverse = MemberOf];
		}
		Class Package.Test.MANAGES
		Extends %Persistent [ClassType =
		persistent, ProcedureBlock]
		{
		Property startDate As %Date;
		Property endDate As %Date;
		Relationship MemberOf As
		DEPARTMENT [Cardinality =
		one, Inverse = ManageOf];
		Relationship hasMg As
		EMPLOYEE [Cardinality =
		one, Inverse = manageFrom];
		}
			interface DEPARTMENT: persistent
			{
			attribute String deptNo;
			attribute String name;
			attribute Integer number;
			attribute String yearMake;
			mt_method "CREATE METHOD
			addEmployee() RETURNS String
			FOR DEPARTMENT BEGIN
			DECLARE addEmployee String;
			RETURN addEmployee; END;";
			mt_method "CREATE METHOD
			coutEmployee() RETURNS String
			FOR DEPARTMENT BEGIN
			DECLARE coutEmployee String;
			RETURN coutEmployee; END;";
			mt_method "CREATE METHOD
			changManger() RETURNS String
			FOR DEPARTMENT BEGIN
			DECLARE changManger String;
			RETURN changManger; END;";
			relationship Set<MANAGES>
			manageOf [1,-1] inverse
			MANAGES::memberOf;}
			interface MANAGES: persistent
			{ attribute Date startDate;
			attribute Date endDate;
			relationship DEPARTMENT
			memberOf inverse
			DEPARTMENT::manageOf;
			relationship EMPLOYEE hasMg
			inverse EMPLOYEE::
			manageFrom;};
				■ กฎข้อ1
				■ กฎข้อ2
				☐ กฎข้อ3
				■ กฎข้อ4
				☐ กฎข้อ5
				☐ กฎข้อ6
				☐ กฎข้อ7
				☐ กฎข้อ8
				☐ กฎข้อ9
				☐ กฎข้อ10
				☐ กฎข้อ11

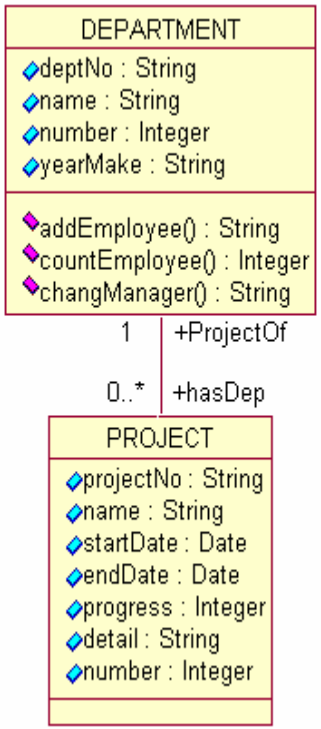
ตารางที่ 6.7 ส่วนประกอบย่อยคลาสไดอะแกรมระบบการจัดการข้อมูลโครงการภายในบริษัทส่วนที่ 7 (ต่อ)

สัญลักษณ์ในคลาสไดอะแกรม	สคีมาฐานข้อมูลของโอดีเอ็มจี	สคีมาของฐานข้อมูลคาเช่	สคีมาของฐานข้อมูลแมทิส	กฎที่ใช้

	class EMPLOYEE extends PERSON
	(extent EMPLOYEEs)
	{ attribute string empNo;
	attribute string address;
	attribute date hireDate;
	attribute string SSN;
	attribute string telNo;
	attribute string depenNo;
	string changDepName ():
	string changProject ():
	relationship EMPLOYEE
	supervisor inverse
	EMPLOYEE:: supervisee;
	relationship set<EMPLOYEE>
	supervisee inverse
	EMPLOYEE:: supervisor;
	relationship set< MANAGES >
	manageFrom inverse
	MANAGES:: hasMg;
	}
		Class Package.Test.EMPLOYEE
		Extends Person [ClassType =
		persistent,ProcedureBlock]
		{
		Property empNo As %String;
		Property address As %String;
		Property hireDate As %Date;
		Property SSN As %String;
		Property telNo As %String;
		Property depenNo As %String;
		Method changDepName()
		As %String{}
		Method changPoject()
		As %String{}
		Relationship supervisor As
		EMPLOYEE [Cardinality =
		one, Inverse = supervisee];
		Relationship supervisee As
		EMPLOYEE [Cardinality =
		many, Inverse = supervisor];
		Relationship manageFrom As
		MANAGES [Cardinality =
		many, Inverse = hasMg];
		}
			interfaceEMPLOYEE:PERSON :
			persistent
			{ attribute String empNo;
			attribute String address;
			attribute Date hireDate;
			attribute String SSN;
			attribute String telNo;
			attribute String depenNo;
			mt_method"CREATE METHOD
			changDepName() RETURNS
			String FOR EMPLOYEE BEGIN
			DECLARE changDepName
			String;
			RETURN changDepName;
			END;";
			mt_method "CREATE METHOD
			changProject() RETURNS
			String FOR EMPLOYEE
			BEGIN
			DECLARE changProject
			String;
			RETURN changProject;
			END;";
			relationship EMPLOYEE
			supervisor[0, 1] inverse
			EMPLOYEE::supervisee;
			relationship Set<EMPLOYEE>
			supervisee inverse
			EMPLOYEE::supervisor;
			relationship Set<MANAGES>
			manageFrom[1,-1] inverse
			MANAGES::hasMg;}	

ตารางที่ 6.8 ส่วนประกอบย่อยคลาสไดอะแกรมระบบการจัดการข้อมูลโครงการภายในบริษัทส่วนที่ 8

สัญลักษณ์ในคลาสไดอะแกรม	สคีมาฐานข้อมูลของโอดีเอ็มจี	สคีมาของฐานข้อมูลคาเช่	สคีมาของฐานข้อมูลแมทิส	กฎที่ใช้

	class DEPARTMENT
	(extent DEPARTMENTs)
	{
	attribute string deptNo;
	attribute string name;
	attribute integer number;
	attribute string yearMake;
	string addEmployee ():
	string coutEmployee ():
	string changManger ():
	relationship set<PROJECT>
	hasDep inverse PROJECT::
	ProjectOf;
	}
		Class Package.DEPARTMENT
		Extends %Persistent [ClassType =
		persistent, ProcedureBlock]
		{
		Property deptNo As %String;
		Property name As %String;
		Property number As %Integer;
		Property yearMake As %String;
		Method addEmployee()
		As %String{}
		Method coutEmployee()
		As %String{}
		Method changManger()
		As %String{}
		Relationship hasDep As
		PROJECT [Cardinality =
		many, Inverse = ProjectOf];
		}
			interface DEPARTMENT: persistent
			{ attribute String deptNo;
			attribute String name;
			attribute Integer number;
			attribute String yearMake;
			mt_method"CREATE METHOD
			addEmployee() RETURNS String
			FOR DEPARTMENT
			BEGIN
			DECLARE addEmployee String;
			RETURN addEmployee; END;";
			mt_method "CREATE METHOD
			coutEmployee() RETURNS String
			FOR DEPARTMENT
			BEGIN
			DECLARE coutEmployee String;
			RETURN coutEmployee; END;";
			mt_method "CREATE METHOD
			changManger() RETURNS String
			FOR DEPARTMENT
			BEGIN
			DECLARE changManger String;
			RETURN changManger; END;";
			relationship Set<PROJECT>
			hasDep[1,-1] inverse
			PROJECT::projectOf;
			};
				■ กฎข้อ1
				■ กฎข้อ2
				■ กฎข้อ3
				☐ กฎข้อ4
				☐ กฎข้อ5
				☐ กฎข้อ6
				☐ กฎข้อ7
				☐ กฎข้อ8
				☐ กฎข้อ9
				☐ กฎข้อ10
				☐ กฎข้อ11

ตารางที่ 6.8 ส่วนประกอบย่อยคลาสไดอะแกรมระบบการจัดการข้อมูลโครงการภายในบริษัทส่วนที่ 8(ต่อ)

สัญลักษณ์ในคลาสไดอะแกรม	สคีมาฐานข้อมูลของโอดีเอ็มจี	สคีมาของฐานข้อมูลคาเช่	สคีมาของฐานข้อมูลแมทิส	กฎที่ใช้

	class PROJECT
	(extent PROJECTs)
	{
	attribute string projectNo;
	attribute string name;
	attribute date startdate;
	attribute date endDate;
	attribute integer progress;
	attribute string detail;
	attribute integer number;
	relationship DEPARTMENT
	ProjectOf inverse
	DEPARTMENT:: hasDep;
	}
		Class Package.PROJECT Extends
		%Persistent [ClassType= persistent,
		ProcedureBlock]
		{
		Property projectNo As %String;
		Property name As %String;
		Property startdate As %Date;
		Property endDate As %Date;
		Property progress As %Integer;
		Property detail As %String;
		Property number As %Integer;
		Relationship ProjectOf As
		DEPARTMENT [Cardinality =
		one, Inverse = hasDep];
		
		}
			interface PROJECT: persistent
			{
			attribute String projectNo;
			attribute String name;
			attribute Date startDate;
			attribute Date endDate;
			attribute Integer progress;
			attribute String detail;
			attribute Integer number;
			relationship DEPARTMENT
			projectOf inverse
			DEPARTMENT::hasDep;
			
			};	

จากนั้นนำคลาสไดอะแกรมที่ออกแบบเสร็จดังรูปที่ 6.2 นำเข้าเครื่องมือเพื่อทดสอบผลลัพธ์จากการแปลง จะได้ผลการทดสอบดังนี้

กรณีที่ 1 สคีมาฐานข้อมูลเชิงวัตถุตามมาตรฐานโอดีเอ็มจี

```
class PERSON(extent PERSONs)
{    attribute string firstName;
     attribute string lastName;
     attribute string titleName;
     attribute date birthDate;
     attribute string sex ;
     attribute string comment;
     integer Age():
 }
  class EMPLOYEE extends PERSON    ( extent EMPLOYEEs)
{     attribute string empNo;
      attribute string address;
      attribute date hireDate;
      attribute string SSN;
      attribute string telNo;
      attribute string depenNo;
      attribute DEPENDENT dependent;
      string changDepName ():
      string changProject ():
      relationship DEPENDENT hasDpt inverse    DEPENDENT :: dependentOf;
      relationship EMPLOYEE supervisor inverse    EMPLOYEE:: supervisee;
      relationship set<EMPLOYEE> supervisee inverse    EMPLOYEE:: supervisor;
      relationship set<WORKSON>  worksIn  inverse    WORKSON:: AllocateTo;
      relationship set< MANAGES >  manageFrom inverse    MANAGES:: hasMg;
}
class DEPENDENT (extent DEPENDENTs)
{    attribute string depenNo;
     attribute date birthDate;
     attribute string sex ;
     attribute string relationship;
     relationship  EMPLOYEE dependentOf  inverse    EMPLOYEE::hasDpt;
}
 class LOCATION (extent LOCATIONs)
{    attribute string locationNo;
     attribute string address;
     attribute string name;
     attribute string telNo;
     attribute string province;
     relationship set<PROJECT> hasPro inverse    PROJECT:: DepentdentOf;
     relationship set< DEPARTMENT >  Locate inverse    DEPARTMENT:: hasLoc ;
}
class PROJECT   ( extent PROJECTs)
{       attribute string projectNo;
       attribute string name;
       attribute date startdate;
       attribute date endDate;
       attribute integer progress;
       attribute string detail;
       attribute integer number;
       relationship LOCATION  locateOf inverse    LOCATION:: hasPro;
       relationship set<WORKSON>  worksInPro  inverse    WORKSON:: forPro;
       relationship DEPARTMENT ProjectOf  inverse    DEPARTMENT:: hasDep;
 }
class DEPARTMENT(extent DEPARTMENTs)
{     attribute string deptNo;
      attribute string name;
      attribute integer  number;
      attribute string yearMake;
      attribute LOCATION location;
      string addEmployee ():
```

```
    string coutEmployee ():
    string changManger ():
    relationship set<LOCATION>  hasLoc  inverse    LOCATION:: Locate;
    relationship set< MANAGES >  manageOf  inverse    MANAGES:: MemberOf;
    relationship set<PROJECT>   hasDep inverse    PROJECT:: ProjectOf;
}
class WORKSON ( extent WORKSONs)
{    attribute integer hours;
     relationship EMPLOYEE  AllocateTo  inverse    EMPLOYEE :: worksIn;
     relationship Project  forPro  inverse    Project  :: worksInPro;
}
class MANAGES ( extent MANAGESs)
{    attribute date startDate;
     attribute date endDate;
     relationship DEPARTMENT MemberOf  inverse    DEPARTMENT:: manageOf;
     relationship EMPLOYEE hasMg  inverse    EMPLOYEE::manageFrom;
}
```

รูปที่ 6.3 แสดงรูปแบบสคีมาฐานข้อมูลเชิงวัตถุตามมาตรฐานโอดีเอ็มจี

กรณีที่ 2 สคีมาฐานข้อมูลเชิงวัตถุฐานข้อมูลคาเช่

```
Class Package.DEPARTMENT Extends %Persistent [ClassType = persistent, ProcedureBlock ]
{
        Property deptNo As %String;
        Property name As %String;
        Property number As %Integer;
        Property yearMake As %String;
        Property location As %LOCATION;
        Method addEmployee() As %String{}
        Method coutEmployee() As %String{}
        Method changManger() As %String{}
        Relationship hasLoc As LOCATION [ Cardinality = many, Inverse = Locate ];
        Relationship ManageOf As MANAGES [ Cardinality =many, Inverse = MemberOf ];
        Relationship hasDep As PROJECT [ Cardinality = many, Inverse = ProjectOf ];
}
Class Package.DEPENDENT
Extends %Persistent[ClassType=persistent,ProcedureBlock]
{       Property depenNo As %String;
        Property birthDate As %Date;
        Property sex As %String;
        Property relationship As %String;
        Relationship DepentdentOf As EMPLOYEE [Cardinality =one,Inverse = hasDpt ];
}
Class Package.EMPLOYEE Extends Person [ ClassType = persistent,ProcedureBlock ]
{
        Property empNo As %String;
        Property address As %String;
        Property hireDate As %Date;
        Property SSN As %String;
        Property telNo As %String;
        Property depenNo As %String;
        Property dependent As %DEPENDENT;
        Method changDepName() As %String{}
        Method changPoject() As %String{}
        Relationship hasDpt As DEPENDENT [ Cardinality = one, Inverse = DepentdentOf ];
        Relationship supervisor As EMPLOYEE [ Cardinality = one, Inverse = supervisee ];
        Relationship supervisee As EMPLOYEE [ Cardinality = many, Inverse = supervisor ];
        Relationship WorksIn As WORKSON [ Cardinality = many, Inverse = AllocateTo ];
        Relationship manageFrom As MANAGES [ Cardinality = many, Inverse = hasMg ];
}
```

```
Class Package.LOCATION
Extends %Persistent [ClassType = persistent, ProcedureBlock]
{       Property locationNo As %String;
        Property address As %String;
        Property name As %String;
        Property telNo As %String;
        Property province As %String;
        Relationship hasPro As PROJECT [ Cardinality = many, Inverse = locateOf ];
        Relationship Locate As DEPARTMENT [ Cardinality = many, Inverse = hasLoc ];
}
Class Package.MANAGES
Extends %Persistent [ClassType = persistent, ProcedureBlock]
{       Property startDate As %Date;
        Property endDate As %Date;
        Relationship MemberOf As DEPARTMENT [ Cardinality = one, Inverse = ManageOf ];
        Relationship hasMg As EMPLOYEE [ Cardinality = one, Inverse = manageFrom ];
}
Class Package.PROJECT
Extends %Persistent [ ClassType= persistent, ProcedureBlock ]
{       Property projectNo As %String;
        Property name As %String;
        Property startdate As %Date;
        Property endDate As %Date;
        Property progress As %Integer;
        Property detail As %String;
        Property number As %Integer;
        Relationship locateOf As LOCATION [ Cardinality = one, Inverse = hasPro ];
        Relationship WorksInPro As WORKSON [ Cardinality = many, Inverse = forPro ];
        Relationship ProjectOf As DEPARTMENT [ Cardinality = one, Inverse = hasDep ];
}
Class Package.Person Extends %Persistent [ ClassType = persistent, ProcedureBlock]
{       Property firstName As %String;
        Property lastName As %String;
        Property titleName As %String;
        Property birthDate As %Date;
        Property sex As %String;
        Property comment As %String;
        Method Age() As %Integer{}
}
Class Package.WORKSON
Extends %Persistent [ ClassType = persistent,ProcedureBlock]
{       Property hours As %Integer;
        Relationship AllocateTo As EMPLOYEE [ Cardinality = one, Inverse = WorksIn ];
        Relationship forPro As PROJECT [ Cardinality = one, Inverse = WorksInPro ];
}
```

รูปที่ 6.4 แสดงรูปแบบสคีมาฐานข้อมูลเชิงวัตถุตามมาตรฐานสำหรับฐานข้อมูลเชิงวัตถุของฐานข้อมูลคาเช่

กรณีที่ 3 สคีมาฐานข้อมูลเชิงวัตถุฐานข้อมูลแมทิส

```
interface DEPARTMENT: persistent{
        attribute String deptNo;
        attribute String name;
        attribute Integer  number;
        attribute String yearMake;
        attribute LOCATION location;
        mt_method "CREATE METHOD addEmployee()
        RETURNS STRING    FOR DEPARTMENT
        BEGIN
                DECLARE addEmployee  STRING;
                RETURN addEmployee;
        END;";
        mt_method "CREATE METHOD coutEmployee()
        RETURNS STRING    FOR DEPARTMENT
```

```
		BEGIN
			DECLARE coutEmployee  STRING;
			RETURN coutEmployee;
		END;";
		mt_method "CREATE METHOD changManger()
		RETURNS STRING    FOR DEPARTMENT
		BEGIN
			DECLARE changManger  STRING;
			RETURN changManger;
		END;";
		relationship Set<LOCATION>  hasLoc[1,-1]  inverse    LOCATION::locate;
		relationship Set<MANAGES>  manageOf[1,-1]  inverse    MANAGES::memberOf;
		relationship Set<PROJECT>   hasDep[1,-1] inverse    PROJECT::projectOf;
};
 interface LOCATION: persistent{
		attribute String locationNo;
		attribute String address;
		attribute String name;
		attribute String telNo;
		attribute String province;
		relationship Set<PROJECT>  hasPro  inverse PROJECT::locateOf;
		relationship  DEPARTMENT  locate  inverse    DEPARTMENT::hasLoc;
};
interface PROJECT: persistent {
		attribute String projectNo;
		attribute String name;
		attribute Date startDate;
		attribute Date endDate;
		attribute Integer progress;
		attribute String detail;
		attribute Integer number;
		relationship LOCATION  locateOf  inverse    LOCATION::hasPro;
		relationship Set<WORKSON>  worksInPro[1,-1]  inverse    WORKSON::forPro;
		relationship DEPARTMENT projectOf  inverse    DEPARTMENT::hasDep;
};
interface MANAGES: persistent{
		attribute Date startDate;
		attribute Date endDate;
		relationship DEPARTMENT memberOf  inverse    DEPARTMENT::manageOf;
		relationship EMPLOYEE hasMg  inverse    EMPLOYEE::manageFrom;
};
 interface WORKSON: persistent {
		attribute Integer hours;
		relationship PROJECT  forPro  inverse    PROJECT::worksInPro;
		relationship EMPLOYEE  allocateTo  inverse    EMPLOYEE::worksIn; };
interface PERSON: persistent{
		attribute String firstName;
		attribute String lastName;
		attribute String titleName;
		attribute Date birthDate;
		attribute String sex ;
		attribute String comment;
		mt_method "CREATE METHOD Age()
		RETURNS Integer    FOR PERSON
		BEGIN
			DECLARE Age  Integer    ;
			RETURN Age;
		END;";
};
interface EMPLOYEE : PERSON : persistent{
		attribute String empNo;
		attribute String address;
		attribute Date hireDate;
		attribute String SSN;
		attribute String telNo;
		attribute String depenNo;
		attribute DEPENDENT dependent;
```

```
        mt_method "CREATE METHOD changDepName()
        RETURNS STRING    FOR EMPLOYEE
        BEGIN

                DECLARE changDepName  STRING;
                RETURN changDepName;
        END;";
        mt_method "CREATE METHOD changProject()
        RETURNS STRING    FOR EMPLOYEE
        BEGIN
                DECLARE changProject  STRING;
                RETURN changProject;
        END;";
        relationship EMPLOYEE supervisor[0, 1] inverse  EMPLOYEE::supervisee;
        relationship Set<EMPLOYEE> supervisee  inverse    EMPLOYEE::supervisor;
        relationship Set<WORKSON>  worksIn[1,-1]  inverse    WORKSON::allocateTo;
        relationship Set<MANAGES>  manageFrom[1,-1]  inverse    MANAGES::hasMg;
        relationship Set<DEPENDENT>  hasDpt  inverse DEPENDENT::dependentOf;
};
interface DEPENDENT: persistent{
        attribute String depenNo;
        attribute Date birthDate;
        attribute String sex ;
        attribute String relateDpt;
        relationship EMPLOYEE dependentOf   inverse   EMPLOYEE::hasDpt;
};
```

รูปที่ 6.5 แสดงรูปแบบสคีมาฐานข้อมูลเชิงวัตถุตามมาตรฐานสำหรับฐานข้อมูลเชิงวัตถุของฐานข้อมูลแมทิส

จากการทดสอบโดยการเปรียบเทียบผลลัพธ์ที่ได้จากการแปลงคลาสไดอะแกรมเป็นสคีมาฐานข้อมูลเชิงวัตถุตามกฎที่สร้างขึ้นตามตารางที่ 6.1 – ตารางที่ 6.8 กับ สคีมาฐานข้อมูลเชิงวัตถุทั้ง 3 ฐานข้อมูลที่ได้จากเครื่องมือ ตามรูปที่ 6.3 - รูปที่ 6.5 จะพบว่าสคีมาฐานข้อมูลเชิงวัตถุที่ได้ ให้ผลลัพธ์ที่เหมือนกัน จึงสรุปได้ว่ากฎที่สร้างขึ้นสามารถนำไปประยุกต์ใช้กับเครื่องมือที่ประยุกต์ใช้กฎการแปลงยูเอ็มแอลคลาสไดอะแกรมเป็นสคีมาฐานข้อมูลเชิงวัตถุได้อย่างถูกต้อง

ในส่วนของตัวอย่างของคลาสไดอะแกรม ที่ได้ประยุกต์ใช้กฎทั้ง 11 ข้อ ใน 1 คลาสไดอะแกรมจะแสดงตัวอย่างของคลาสไดอะแกรมและผลลัพธ์ที่ได้ในภาคผนวก ข

6.2. การตรวจสอบความถูกต้องของการแปลงคลาสไดอะแกรมเป็นสคีมาฐานข้อมูลเชิงวัตถุ

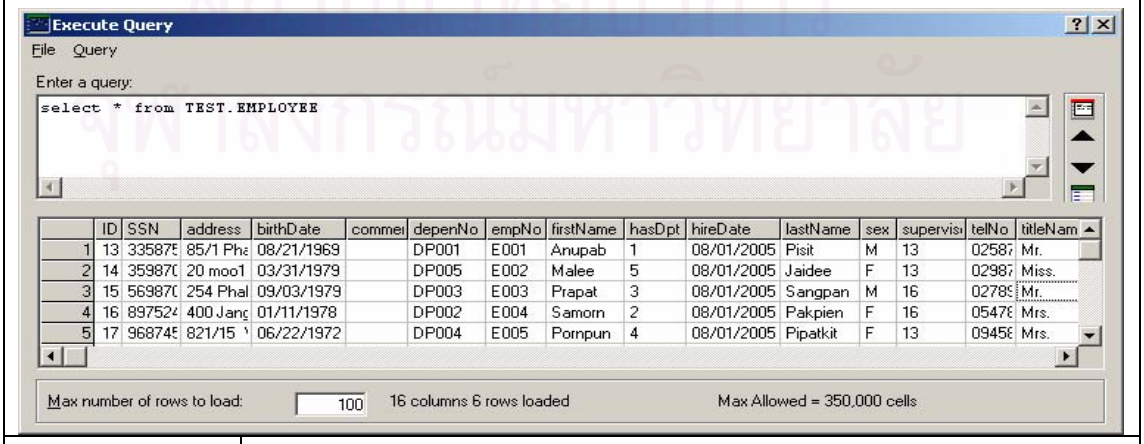
ในหัวข้อนี้จะทำการตรวจสอบความถูกต้องของการแปลงคลาสไดอะแกรมเป็นสคีมาฐานข้อมูลเชิงวัตถุโดยการทดสอบดึงข้อมูลจากฐานข้อมูลที่ถูกสร้างขึ้น จากเครื่องมือที่ประยุกต์กฎการแปลงคลาสไดอะแกรมเป็นสคีมาฐานข้อมูลเชิงวัตถุ เพื่อทดสอบว่าฐานข้อมูลแต่ละชนิดสามารถแสดงผลลัพธ์จากการดึงข้อมูลออกมาได้ถูกต้องตามที่ออกแบบไว้หรือไม่ การทดสอบจะใช้ภาษาสำหรับค้นหาข้อมูลเชิงวัตถุ โดยการระบุชื่อแอทริบิวต์ที่ต้องการจากคลาสไดอะแกรม

การทดสอบจะทดสอบกับฐานข้อมูล 2 ชนิดเท่านั้นคือ ฐานข้อมูลคาเช่ และฐานข้อมูลแมทิส ส่วนฐานข้อมูลตามมาตรฐานโอดีเอ็มจีจะไม่ทดสอบเนื่องจากฐานข้อมูลดังกล่าวเป็นมาตรฐานสำหรับการสร้างฐานข้อมูลไม่สามารถสร้างฐานข้อมูลได้โดยตรง

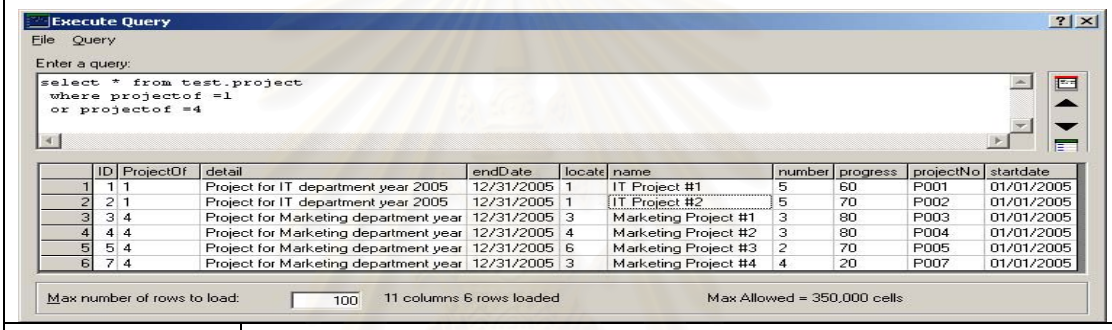
6.2.1 การทดสอบความถูกต้องของสคีมาฐานข้อมูลของฐานข้อมูลคาเช่

การทดสอบความถูกต้องของสคีมาฐานข้อมูลของฐานข้อมูลคาเช่สามารถแสดงได้ตามตารางที่ 6.9 –ตารางที่ 6.13 ดังนี้

ตารางที่ 6.9 แสดงการทดสอบความถูกต้องของสคีมาฐานข้อมูลการทดสอบที่ 1

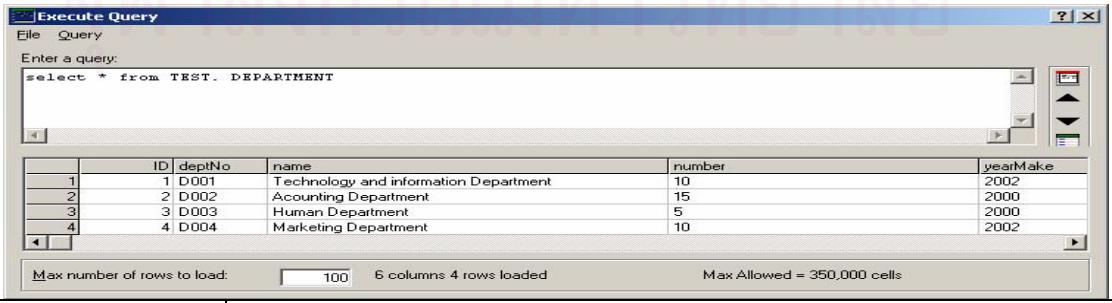
<table>
<tr><td colspan="8">TEST REPORT</td></tr>
<tr><td>ID:</td><td>TestCase001</td><td>Date:</td><td>10/03/2005</td><td>By:</td><td>Sathit</td><td>Database</td><td>Cache’</td></tr>
<tr><td>การทดสอบ</td><td colspan="7">ทดสอบดึงข้อมูลโดยใช้คำสั่ง : Select * From TEST.EMPLOYEE</td></tr>
<tr><td>ผลลัพธ์ที่ต้องการ</td><td colspan="7">แสดงข้อมูลจำนวน 6 เร็คคอร์ด โดยแอทริบิวต์ต่าง ๆที่แสดงจะเป็นแอทริบิวต์ที่ปรากฏในคลาสไดอะแกรม ของคลาส PERSON และ EMPLOYEE และมีแอทริบิวต์ที่เกิดจากความสัมพันธ์ระหว่างคลาส PERSON, EMPLOYEE, DEPARTMENT, PROJECT, DEPENDENTและ WORKSON</td></tr>
<tr><td>ผลการทดสอบ</td><td colspan="7"> 
</td></tr>
<tr><td>สรุปผล</td><td colspan="7">การแสดงผลของข้อมูลถูกต้อง</td></tr>
</table>

ตารางที่ 6.10 แสดงการทดสอบความถูกต้องของสคีมาฐานข้อมูล การทดสอบที่ 2

TEST REPORT							
ID:	TestCase002	Date:	10/03/2005	By:	Sathit	Database	Cache'
การทดสอบ	ทดสอบดึงข้อมูลโดยใช้คำสั่ง : Select * From TEST.PROJECT Where projectof =1 or projectof =4						
ผลลัพธ์ที่ต้องการ	แสดงข้อมูลจำนวน 6 เร็คคอร์ด โดยแสดงแอทริบิวต์ที่ปรากฏในคลาสไดอะแกรม ของคลาส PROJECT และมีแอทริบิวต์ที่เกิดจากความสัมพันธ์ระหว่างคลาส DEPARTMENT และ PROJECT โดยแสดงรายละเอียดของคลาส PROJECT ที่อยู่แผนก = 4						
ผลการทดสอบ							
สรุปผล	การแสดงผลของข้อมูลถูกต้อง						

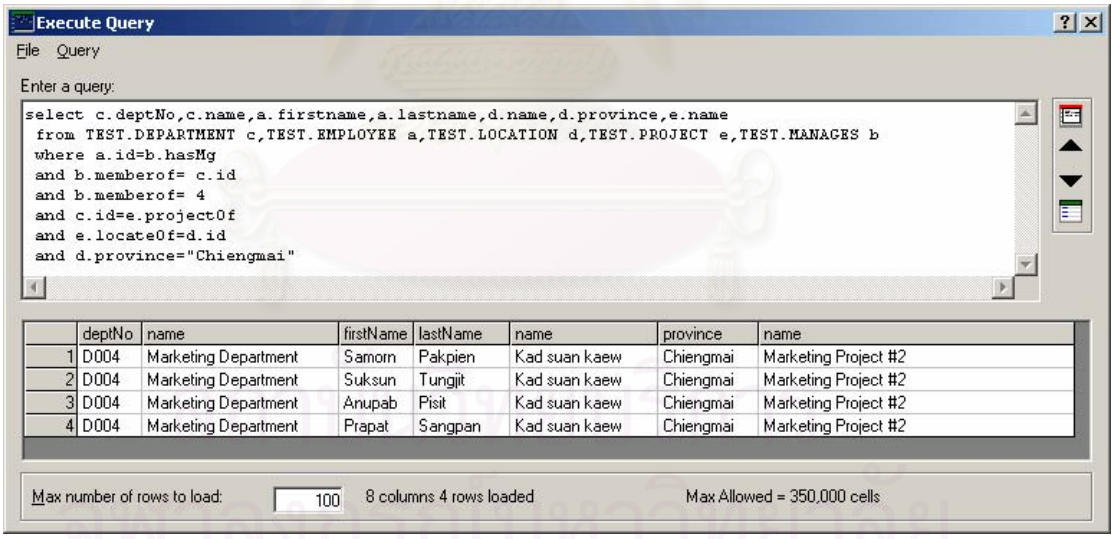
	ID	ProjectOf	detail	endDate	locate	name	number	progress	projectNo	startdate
1	1	1	Project for IT department year 2005	12/31/2005	1	IT Project #1	5	60	P001	01/01/2005
2	2	1	Project for IT department year 2005	12/31/2005	1	IT Project #2	5	70	P002	01/01/2005
3	3	4	Project for Marketing department year	12/31/2005	3	Marketing Project #1	3	80	P003	01/01/2005
4	4	4	Project for Marketing department year	12/31/2005	4	Marketing Project #2	3	80	P004	01/01/2005
5	5	4	Project for Marketing department year	12/31/2005	6	Marketing Project #3	2	70	P005	01/01/2005
6	7	4	Project for Marketing department year	12/31/2005	3	Marketing Project #4	4	20	P007	01/01/2005

ตารางที่ 6.11 แสดงการทดสอบความถูกต้องของสคีมาฐานข้อมูล การทดสอบที่ 3

TEST REPORT							
ID:	TestCase003	Date:	10/03/2005	By:	Sathit	Database	Cache'
การทดสอบ	ทดสอบดึงข้อมูลโดยใช้คำสั่ง : Select * from TEST. DEPARTMENT						
ผลลัพธ์ที่ต้องการ	แสดงข้อมูลจำนวน 4 เร็คคอร์ด โดยแอทริบิวต์ต่าง ๆที่แสดงจะเป็นแอทริบิวต์ที่ปรากฏในคลาสไดอะแกรม ของคลาส DEPARTMENT						
ผลการทดสอบ							
สรุปผล	การแสดงผลของข้อมูลถูกต้อง						

	ID	deptNo	name	number	yearMake
1	1	D001	Technology and information Department	10	2002
2	2	D002	Acounting Department	15	2000
3	3	D003	Human Department	5	2000
4	4	D004	Marketing Department	10	2002

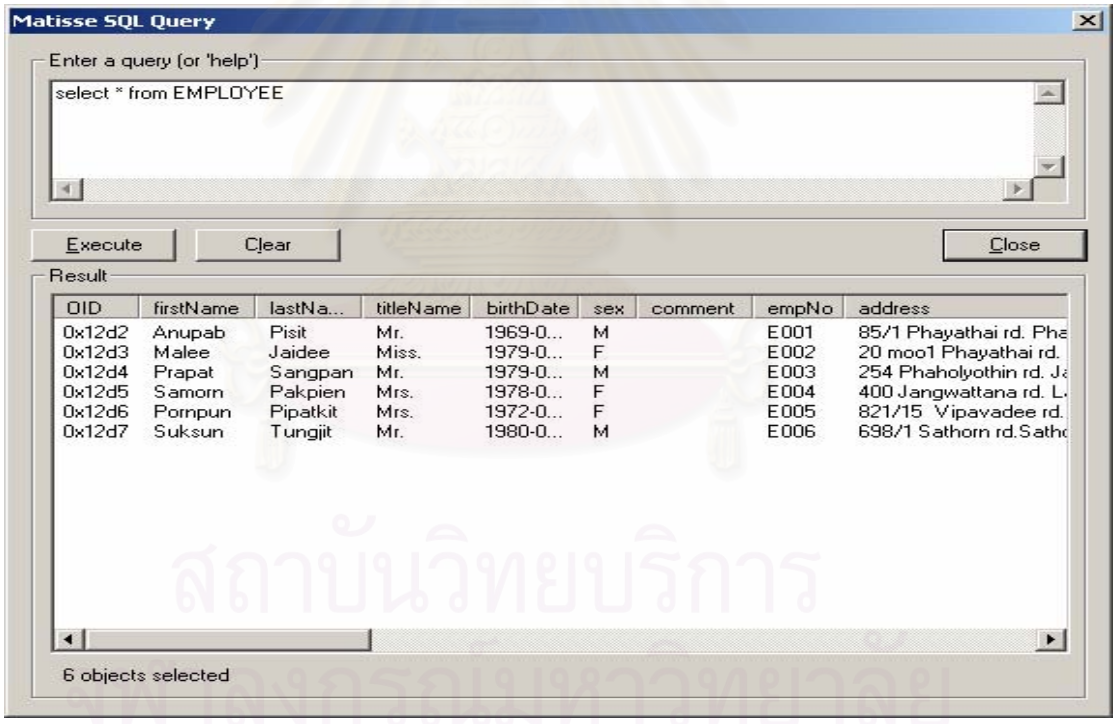
ตารางที่ 6.12 แสดงการทดสอบความถูกต้องของสคีมาฐานข้อมูล การทดสอบที่ 4

<table>
<tr><td colspan="8">TEST REPORT</td></tr>
<tr><td>ID:</td><td colspan="2">TestCase004</td><td>Date:</td><td>10/03/2005</td><td>By:</td><td>Sathit</td><td>Database</td><td>Cache'</td></tr>
<tr><td>การทดสอบ</td><td colspan="8">ทดสอบดึงข้อมูลโดยใช้คำสั่ง :
Select c.deptNo,c.name,a.firstname,a.lastname,
d.name,d.province,e.name
From TEST.DEPARTMENT c,TEST.EMPLOYEE a,
TEST.LOCATION d,TEST.PROJECT e,TEST.MANAGES b
Where a.id=b.hasMg and b.memberof= c.id and b.memberof= 1
and c.id=e.projectOf and e.locateOf=d.id
and d.province="Chiengmai"</td></tr>
<tr><td>ผลลัพธ์ที่ต้องการ</td><td colspan="8">แสดงข้อมูลจำนวน 4 เร็คคอร์ด โดยแอทริบิวต์ต่าง ๆที่แสดงจะเป็นแอทริบิวต์ที่ปรากฏในคลาสไดอะแกรม โดยการเลือกบางแอทริบิวต์ขึ้นมาแสดงและเร็คคอร์ด จะแสดงรายละเอียดของโครงการที่มีสถานที่ตั้ง = Chiengmai</td></tr>
<tr><td colspan="9">ผลการทดสอบ
 
</td></tr>
<tr><td>สรุปผล</td><td colspan="8">การแสดงผลของข้อมูลถูกต้อง</td></tr>
</table>

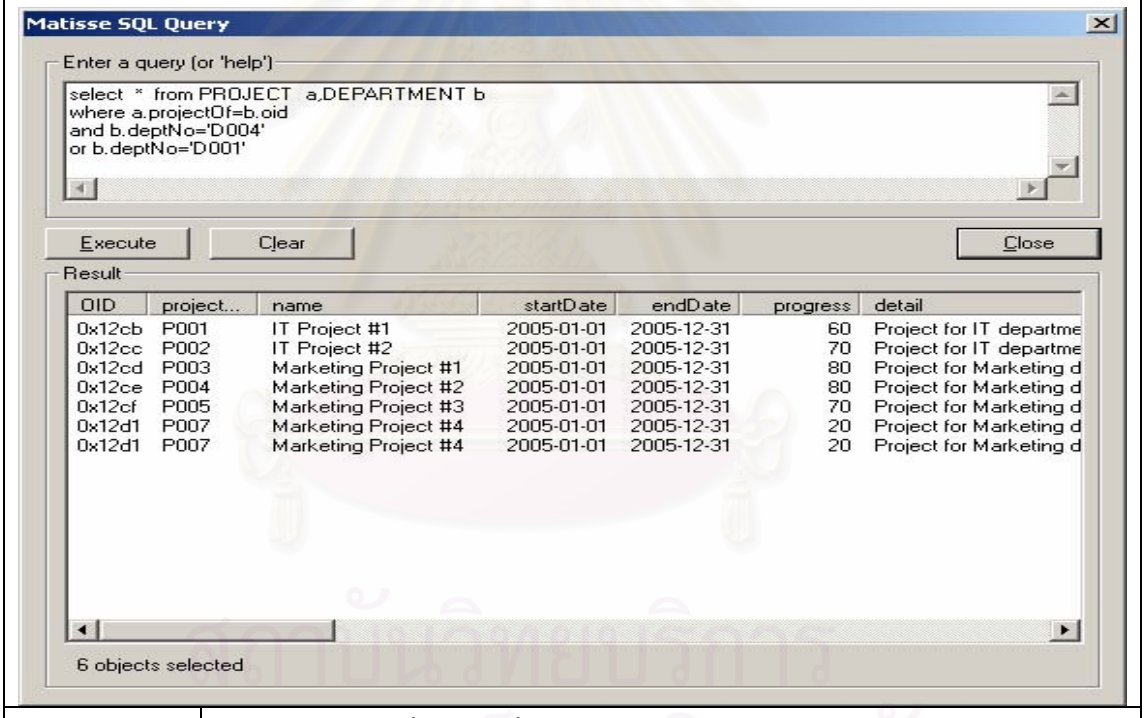
6.2.2 การทดสอบความถูกต้องของสคีมาฐานข้อมูลของฐานข้อมูลแมทิส

การทดสอบความถูกต้องของสคีมาฐานข้อมูลของฐานข้อมูล แมทิส สามารถแสดงได้ตามตารางที่ 6.13 –ตารางที่ 6.16 ดังนี้

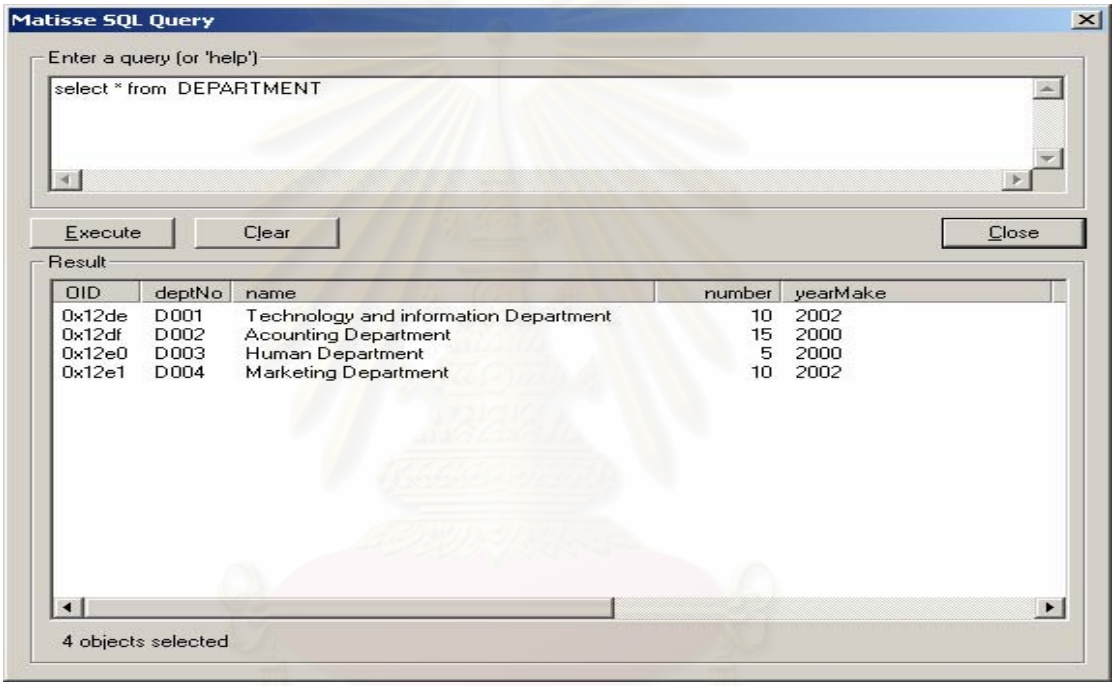
ตารางที่ 6.13 แสดงการทดสอบความถูกต้องของสคีมาฐานข้อมูล การทดสอบที่5

<table>
<tr><td colspan="8">TEST REPORT</td></tr>
<tr><td>ID:</td><td>TestCase005</td><td>Date:</td><td>10/03/2005</td><td>By:</td><td>Sathit</td><td>Database</td><td>Matisse</td></tr>
<tr><td>การทดสอบ</td><td colspan="7">ทดสอบดึงข้อมูลโดยใช้คำสั่ง : Select * From EMPLOYEE</td></tr>
<tr><td>ผลลัพธ์ที่ต้องการ</td><td colspan="7">แสดงข้อมูลจำนวน 6 เร็คคอร์ด โดยแอทริบิวต์ต่าง ๆที่แสดงจะเป็นแอทริบิวต์ที่ปรากฏในคลาสไดอะแกรม ของคลาส PERSON และ EMPLOYEE และมีแอทริบิวต์ที่เกิดจากความสัมพันธ์ระหว่างคลาส PERSON, EMPLOYEE, DEPARTMENT, PROJECT, DEPENDENTและ WORKSON</td></tr>
<tr><td colspan="8">ผลการทดสอบ
 
</td></tr>
<tr><td>สรุปผล</td><td colspan="7">การแสดงผลของข้อมูลถูกต้อง</td></tr>
</table>

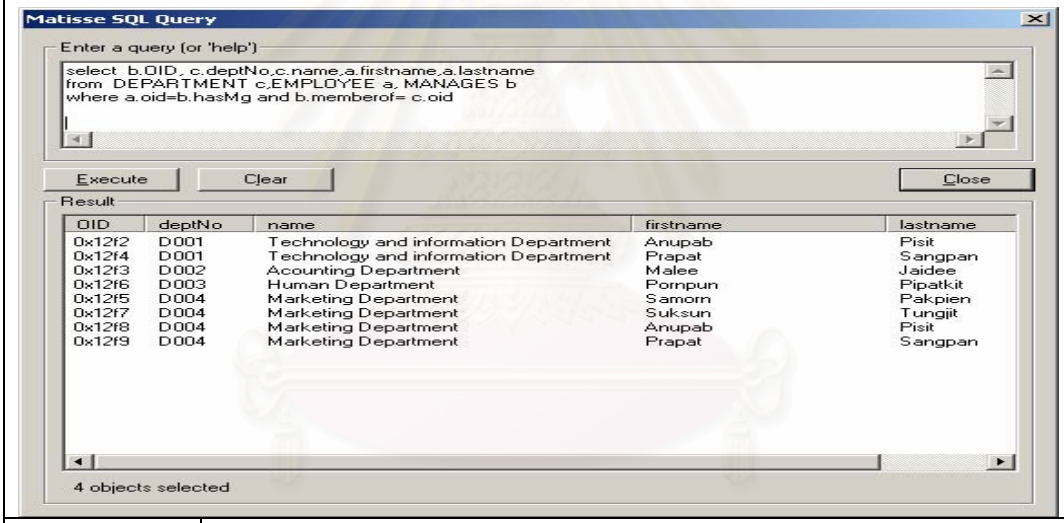
ตารางที่ 6.14 แสดงการทดสอบความถูกต้องของสคีมาฐานข้อมูล การทดสอบที่ 6

<table>
<tr><td colspan="8">TEST REPORT</td></tr>
<tr><td>ID:</td><td>TestCase006</td><td>Date:</td><td>10/03/2005</td><td>By:</td><td>Sathit</td><td>Database</td><td>Matisse</td></tr>
<tr><td>การทดสอบ</td><td colspan="7">ทดสอบดึงข้อมูลโดยใช้คำสั่ง : Select * From PROJECT a,DEPARTMENT b Where a.projectOf=b.oid and b.deptNo='D004'or b.deptNo='D001'</td></tr>
<tr><td>ผลลัพธ์ที่ต้องการ</td><td colspan="7">แสดงข้อมูลจำนวน 6 เร็คคอร์ด โดยแอทริบิวต์ต่าง ๆที่แสดงจะเป็นแอทริบิวต์ที่ปรากฏในคลาสไดอะแกรม ของคลาส PROJECT และมีแอทริบิวต์ที่เกิดจากความสัมพันธ์ระหว่างคลาส DEPARTMENT และ PROJECT โดยแสดงรายละเอียดของคลาส PROJECT ที่อยู่แผนก = D004 ,D001</td></tr>
<tr><td colspan="8">ผลการทดสอบ
 
</td></tr>
<tr><td>สรุปผล</td><td colspan="7">การแสดงผลของข้อมูลถูกต้อง</td></tr>
</table>

ตารางที่ 6.15 แสดงการทดสอบความถูกต้องของสคีมาฐานข้อมูล การทดสอบที่ 7

<table>
<tr><td colspan="8">TEST REPORT</td></tr>
<tr><td>ID:</td><td>TestCase007</td><td>Date:</td><td>10/03/2005</td><td>By:</td><td>Sathit</td><td>Database</td><td>Matisse</td></tr>
<tr><td>การทดสอบ</td><td colspan="7">ทดสอบดึงข้อมูลโดยใช้คำสั่ง : Select * From DEPARTMENT</td></tr>
<tr><td>ผลลัพธ์ที่ต้องการ</td><td colspan="7">แสดงข้อมูลจำนวน 4 เร็คคอร์ด โดยแอทริบิวต์ต่าง ๆที่แสดงจะเป็นแอทริบิวต์ที่ปรากฏในคลาสไดอะแกรม ของคลาส DEPARTMENT</td></tr>
<tr><td colspan="8">ผลการทดสอบ
 
</td></tr>
<tr><td>สรุปผล</td><td colspan="7">การแสดงผลของข้อมูลถูกต้อง</td></tr>
</table>

ตารางที่ 6.16 แสดงการทดสอบความถูกต้องของสคีมาฐานข้อมูล การทดสอบที่ 8

<table>
<tr><td colspan="8">TEST REPORT</td></tr>
<tr><td>ID:</td><td>TestCase008</td><td>Date:</td><td>10/03/2005</td><td>By:</td><td>Sathit</td><td>Database</td><td>Matisse</td></tr>
<tr><td>การทดสอบ</td><td colspan="7">ทดสอบดึงข้อมูลโดยใช้คำสั่ง :
Select b.OID, c.deptNo,c.name,a.firstname,a.lastname
From DEPARTMENT c,EMPLOYEE a, MANAGES b
Where a.oid=b.hasMg and b.memberof= c.oid</td></tr>
<tr><td>ผลลัพธ์ที่ต้องการ</td><td colspan="7">แสดงข้อมูลจำนวน 8 เร็คคอร์ด โดยแอทริบิวต์ต่าง ๆที่แสดงจะเป็นแอทริบิวต์ที่ปรากฏในคลาสไดอะแกรม ของคลาส MANAGE และมีแอทริบิวต์ที่เกิดจากความสัมพันธ์ระหว่างคลาส DEPARTMENT, EMPLOYEE และ MANAGE โดยการเลือกบางแอทริบิวต์ขึ้นมาแสดง</td></tr>
<tr><td colspan="8">ผลการทดสอบ
 
</td></tr>
<tr><td>สรุปผล</td><td colspan="7">การแสดงผลของข้อมูลถูกต้อง</td></tr>
</table>

จากการทดสอบได้ข้อสรุปดังนี้ เมื่อทดสอบความถูกต้องของส่วนประกอบต่าง ๆ ที่สร้างในฐานข้อมูลเชิงวัตถุจาก คลาสไดอะแกรมโดยการทดสอบดึงข้อมูลจากฐานข้อมูลที่สร้างจากสคีมาฐานข้อมูลเชิงวัตถุที่ได้จากการแปลง โดยการทดสอบดังกล่าวจะเป็นการยืนยันว่าเครื่องมือสามารถแปลงชื่อหรือความสัมพันธ์ต่างๆจากคลาสไดอะแกรมถูกต้อง จะพบว่าการดึงข้อมูลจากฐานข้อมูลโดยการระบุชื่อฟิลด์ที่ปรากฏในคลาสไดอะแกรม พบว่าข้อมูลที่ดึงออกมามีความถูกต้อง ดังแสดงในตารางที่ 6.9-6.16

ดังนั้นจึงสรุปได้ว่ากฎการแปลงทั้ง 11 ข้อสามารถแปลงข้อมูลเป็นสคีมาฐานข้อมูลเชิงวัตถุได้ถูกต้องและสามารถนำสคีมาดังกล่าวไปใช้งานได้จริงในฐานข้อมูล

6.3. การประเมินผลและวิเคราะห์ผลที่ได้จากการทดสอบการแปลงคลาสไดอะแกรมเป็นสคีมาฐานข้อมูลเชิงวัตถุ

หลังจากที่ทดสอบการแปลงคลาสไดอะแกรมเป็นสคีมาฐานข้อมูลเชิงวัตถุกับ เครื่องมือที่ประยุกต์ใช้กฎการแปลงยูเอ็มแอลคลาสไดอะแกรม จะพิจารณาผลของการทดสอบจากหัวข้อ ดังนี้

1.จำนวนคลาสที่ถูกสร้างทั้งหมด

2.จำนวนแอทริบิวต์ที่ถูกสร้างทั้งหมด

3.จำนวนเมทธอดที่ถูกสร้างทั้งหมด

4.จำนวนความสัมพันธ์ที่ถูกสร้างทั้งหมด

5.ความถูกต้องจากการทดสอบดึงข้อมูลจากฐานข้อมูลที่สร้างทั้งหมด

การทดสอบในหัวข้อนี้จะทดสอบว่าเมื่อแปลงจากคลาสไดอะแกรมเป็นสคีมาฐานข้อมูลเชิงวัตถุแล้ว จำนวนของคลาส แอทริบิวต์ เมทธอด ความสัมพันธ์ที่ถูกสร้างจากคลาสไดอะแกรมแล้วเมื่อไปสร้างเป็นสคีมาของฐานข้อมูลเชิงวัตถุแล้ว ข้อมูลที่อยู่ภายในคลาสไดอะแกรมและภายในฐานข้อมูลมีความสอดคล้องกันหรือไม่ สรุปเป็นตารางได้ดังนี้

1.จำนวนคลาสที่ถูกสร้างทั้งหมด

การทดสอบจำนวนคลาสที่ถูกสร้างทั้งหมด จะทดสอบว่า เมื่อแปลงคลาสไดอะแกรม เป็นสคีมาฐานข้อมูลเชิงวัตถุแล้วจำนวนคลาสทั้งหมดที่อยู่ในคลาสไดอะแกรมมีจำนวนเท่ากับจำนวนคลาสที่อยู่ในสคีมาฐานข้อมูลเชิงวัตถุหรือไม่ ซึ่งจากการทดสอบคลาสไดอะแกรมที่มีจำนวนคลาสทั้งหมด 8 คลาส และเปรียบเทียบกับคลาสในสคีมาฐานข้อมูลเชิงวัตถุได้ผลดังตารางที่ 6.17 โดยตัวเลขภายในตารางแสดงถึงจำนวนคลาสที่ได้จากการทดสอบนับจำนวนคลาสแต่ละประเภท

ตารางที่ 6.17 แสดงผลลัพธ์ที่ได้จากการทดสอบจำนวนคลาสที่ถูกสร้างทั้งหมด

รายละเอียดการทดลอง	ชนิดของฐานข้อมูลเชิงวัตถุ		
	มาตรฐานโอดีเอ็มจี	ฐานข้อมูลคาเช่	ฐานข้อมูลแมทิส
1.จำนวนคลาสในคลาสไดอะแกรม	8	8	8
2.จำนวนคลาสในฐานข้อมูล	-	8	8
3.จำนวนคลาสที่ได้จากเครื่องมือ	8	8	8
4.ความถูกต้องของการแปลง	100%	100%	100%

2.จำนวนแอทริบิวต์ที่ถูกสร้างทั้งหมด

การทดสอบจำนวนแอทริบิวต์ที่ถูกสร้างทั้งหมด จะทดสอบว่า เมื่อแปลงคลาสไดอะแกรมเป็นสคีมาฐานข้อมูลเชิงวัตถุแล้วจำนวนแอทริบิวต์ทั้งหมดที่อยู่ในคลาสไดอะแกรมมีจำนวนเท่ากับจำนวนแอทริบิวต์ที่อยู่ในสคีมาฐานข้อมูลเชิงวัตถุหรือไม่ ซึ่งจากการทดสอบคลาสไดอะแกรมที่มีจำนวนแอทริบิวต์ทั้งหมด 36 แอทริบิวต์และเปรียบเทียบกับแอทริบิวต์ในสคีมาฐานข้อมูลเชิงวัตถุได้ผลดังตารางที่ 6.18 ตัวเลขภายในตารางแสดงถึงจำนวนแอทริบิวต์ที่ได้จากการทดสอบนับจำนวนแอทริบิวต์ของแต่ละคลาส

ตารางที่ 6.18 แสดงผลลัพธ์ที่ได้จากการทดสอบจำนวนแอริบิวต์ที่ถูกสร้างทั้งหมด

จำนวนแอทริบิวต์ที่ถูกสร้างในคลาส	จำนวนในคลาสไดอะแกรม	ชนิดของฐานข้อมูลเชิงวัตถุ					
		มาตรฐานโอดีเอ็มจี		ฐานข้อมูลคาเช่		ฐานข้อมูลแมทิส	
		ในฐานข้อมูล	จากเครื่องมือ	ในฐานข้อมูล	จากเครื่องมือ	ในฐานข้อมูล	จากเครื่องมือ
PERSON	6	-	6	6	6	6	6
EMPLOYEE	6	-	6	6	6	6	6
DEPENDENT	4	-	4	4	4	4	4
LOCATION	5	-	5	5	5	5	5
PROJECT	7	-	7	7	7	7	7
DEPARTMENT	4	-	4	4	4	4	4
WORKSON	1	-	1	1	1	1	1
MANAGES	2	-	2	2	2	2	2
จำนวนแอทริบิวต์ทั้งหมด	35	-	35	35	35	35	35
ความถูกต้องของการแปลง	100 %	-	100 %	100 %	100 %	100 %	100 %

3.จำนวนเมทธอดที่ถูกสร้างทั้งหมด

การทดสอบจำนวนเมทธอดที่ถูกสร้างทั้งหมด จะทดสอบว่า เมื่อแปลงคลาสไดอะแกรมเป็นสคีมาฐานข้อมูลเชิงวัตถุแล้วจำนวนเมทธอดทั้งหมดที่อยู่ในคลาสไดอะแกรมมีจำนวนเท่ากับเมทธอดที่อยู่ในสคีมาฐานข้อมูลเชิงวัตถุหรือไม่ ซึ่งจากการทดสอบคลาสไดอะแกรมที่มีจำนวนเมทธอดทั้งหมด 9 เมทธอดและเปรียบเทียบกับเมทธอดในสคีมาฐานข้อมูลเชิงวัตถุได้ผลดังตารางที่ 6.19 ตัวเลขภายในตารางแสดงถึงจำนวนเมทธอดที่ได้จากการทดสอบนับจำนวนเมทธอด ของแต่ละคลาส

ตารางที่ 6.19 แสดงผลลัพธ์ที่ได้จากการทดสอบจำนวนเมทธอดที่ถูกสร้างทั้งหมด

จำนวนเมทธอดที่ถูกสร้างในคลาส	จำนวนในคลาสไดอะแกรม	ชนิดของฐานข้อมูลเชิงวัตถุ					
		มาตรฐานโอดีเอ็มจี		ฐานข้อมูลคาเช่		ฐานข้อมูล แมทิส	
		ในฐานข้อมูล	จากเครื่องมือ	ในฐานข้อมูล	จากเครื่องมือ	ในฐานข้อมูล	จากเครื่องมือ
PERSON	1	-	1	1	1	1	1
EMPLOYEE	2	-	2	2	2	2	2
DEPENDENT	0	-	0	0	0	0	0
LOCATION	0	-	0	0	0	0	0
PROJECT	0	-	0	0	0	0	0
DEPARTMENT	3	-	3	3	3	3	3
WORKSON	0	-	0	0	0	0	0
MANAGES	0	-	0	0	0	0	0
จำนวนเมทธอดทั้งหมด	6	-	6	6	6	6	6
ความถูกต้องของการแปลง	100 %	-	100 %	100 %	100 %	100 %	100 %

4.จำนวนความสัมพันธ์ที่ถูกสร้างทั้งหมด

การทดสอบจำนวนความสัมพันธ์ที่ถูกสร้างทั้งหมด จะทดสอบว่า เมื่อแปลงคลาสไดอะแกรม เป็นสคีมาฐานข้อมูลเชิงวัตถุแล้วจำนวนความสัมพันธ์ทั้งหมดที่อยู่ในคลาสไดอะแกรมมีจำนวนเท่ากับความสัมพันธ์ที่อยู่ในสคีมาฐานข้อมูลเชิงวัตถุหรือไม่ ซึ่งจากการทดสอบคลาสไดอะแกรมที่มีจำนวนความสัมพันธ์ทั้งหมด 19 ความสัมพันธ์และเปรียบเทียบกับความสัมพันธ์ใน สคีมาฐานข้อมูลเชิงวัตถุได้ผลดังตารางที่ 6.20 ตัวเลขภายในตารางแสดงถึง จำนวนความสัมพันธ์ระหว่างคลาสที่ได้จากการทดสอบนับจำนวนความสัมพันธ์ระหว่างคลาส ของแต่ละคลาส

ตารางที่ 6.20 แสดงผลลัพธ์ที่ได้จากการทดสอบจำนวนความสัมพันธ์ที่ถูกสร้างทั้งหมด

จำนวนความสัมพันธ์ที่ถูกสร้างในคลาส	จำนวนในคลาสไดอะ-แกรม	ชนิดของฐานข้อมูลเชิงวัตถุ					
		มาตรฐานโอดีเอ็มจี		ฐานข้อมูลคาเช่		ฐานข้อมูล แมทิส	
		ในฐานข้อมูล	จากเครื่องมือ	ในฐานข้อมูล	จากเครื่องมือ	ในฐานข้อมูล	จากเครื่องมือ
PERSON	0	-	0	0	0	0	0
EMPLOYEE	6	-	6	6	6	6	6
DEPENDENT	1	-	1	1	1	1	1
LOCATION	2	-	2	2	2	2	2
PROJECT	3	-	3	3	3	3	3
DEPARTMENT	3	-	3	3	3	3	3
WORKS_ON	2	-	2	2	2	2	2
MANAGES	2	-	2	2	2	2	2
จำนวนความสัมพันธ์ทั้งหมด	19	-	19	19	19	19	19
ความถูกต้องของการแปลง	100 %	-	100 %	100 %	100 %	100 %	100 %

5.ความถูกต้องจากการทดสอบดึงข้อมูลจากฐานข้อมูลที่สร้างทั้งหมด

ตารางที่ 6.21 แสดงผลลัพธ์ที่ได้จากการทดสอบความถูกต้องจากการทดสอบดึงข้อมูลจากฐานข้อมูลที่สร้าง

รายละเอียดการทดลอง	ชนิดของฐานข้อมูลเชิงวัตถุ	
	ฐานข้อมูลคาเช่	ฐานข้อมูล แมทิส
ความถูกต้องของการทดสอบดึงข้อมูลจากฐานข้อมูล	การแสดงผลของข้อมูลถูกต้อง	การแสดงผลของข้อมูลถูกต้อง

จากตารางที่ 6.17 -6.21 ได้ข้อสรุปดังนี้

เมื่อทดสอบกฎการแปลงคลาสไดอะแกรมเป็นสคีมาฐานข้อมูลเชิงวัตถุโดยอาศัยคลาสไดอะแกรมดังรูปที่ 6.2 จะพบว่า จำนวนคลาส แอทริบิวต์ เมทธอด ความสัมพันธ์ต่าง ๆ ที่ปรากฏในคลาสไดอะแกรม เมื่อทดสอบโดยการนับจำนวนของส่วนประกอบต่าง ๆที่ปรากฏในคลาสไดอะแกรมและในสคีมาฐานข้อมูลเชิงวัตถุจะพบว่ามีจำนวนเท่ากันดังแสดงในตารางที่ 6.17 -6.21 และเมื่อทดสอบความถูกต้องของส่วนประกอบต่าง ๆ ที่สร้างในฐานข้อมูลเชิงวัตถุจากคลาสไดอะแกรมโดยการทดสอบดึงข้อมูลจากฐานข้อมูลจะพบว่าข้อมูลที่ดึงออกมามีความถูกต้องทั้งหมดทั้ง 3 กรณี ดังแสดงในตารางที่ 6.9- 6.16 ดังนั้นจึงสรุปได้ว่ากฎการแปลงทั้ง 11 ข้อสามารถแปลงข้อมูลเป็นสคีมาฐานข้อมูลเชิงวัตถุได้ถูกต้องและสามารถใช้งานได้จริงทั้ง 3 ชนิดของฐานข้อมูล

บทที่ 7

สรุปผลการวิจัยและข้อเสนอแนะ

7.1 สรุปผลการวิจัย

ในวิทยานิพนธ์ฉบับนี้ ได้ทำการออกแบบกฎการแปลงยูเอ็มแอลคลาสไดอะแกรม เป็นสคีมาตัวกลางของสคีมาฐานข้อมูลเชิงวัตถุเพื่อให้สามารถนำกฎไปประยุกต์ใช้ในการสร้างเครื่องมือสำหรับการแปลงยูเอ็มแอลคลาสไดอะแกรมเป็นสคีมาฐานข้อมูลเชิงวัตถุ โดยได้ออกแบบกฎการแปลงคลาสไดอะแกรม เป็นสคีมาตัวกลางของสคีมาฐานข้อมูลเชิงวัตถุจำนวน 11 ข้อ ของฐานข้อมูลทั้ง 3 ประเภทคือ

1.ฐานข้อมูลเชิงวัตถุตามมาตรฐานของโอดีเอ็มจี

2.ฐานข้อมูลเชิงวัตถุของฐานข้อมูลคาเช่

3.ฐานข้อมูลเชิงวัตถุของฐานข้อมูลแมทิส

โดยกฎการแปลงทั้ง 11 ข้อ ประกอบไปด้วย

กฎข้อที่ 1 กฎการแปลงคลาสและอินเตอร์เฟสเป็นสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุ

กฎข้อที่ 2 กฎการแปลงแอทริบิวต์และเมทธอดเป็นสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุ

กฎข้อที่ 3 กฎการแปลงความสัมพันธ์ระหว่างคลาสแบบแอสโซซิเอชันเป็นสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุ

กฎข้อที่ 4 กฎการแปลงความสัมพันธ์ระหว่างคลาสแบบแอสโซซิเอชันคลาสเป็นสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุ

กฎข้อที่ 5 กฎการแปลงความสัมพันธ์ระหว่างคลาสแบบรีเคอร์ซีพแอสโซซิเอชันเป็นสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุ

กฎข้อที่ 6 กฎการแปลงความสัมพันธ์ระหว่างคลาสแบบดีเพนเดนซีเป็นสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุ

กฎข้อที่ 7 กฎการแปลงความสัมพันธ์ระหว่างคลาสสัมพันธ์แบบแอกกรีเกชันเป็นสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุ

กฎข้อที่ 8 กฎการแปลงความสัมพันธ์ระหว่างคลาสแบบคอมโพสิชันเป็นสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุ

กฎข้อที่ 9 กฎการแปลงความสัมพันธ์ระหว่างคลาสแบบคอลลิไฟล์แอสโซซิเอชันเป็นสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุ

กฎข้อที่ 10 กฎการแปลงความสัมพันธ์ระหว่างคลาสแบบเจนเนอรัลไลเซชันเป็นสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุ

กฎข้อที่ 11 กฎการแปลงความสัมพันธ์ระหว่างคลาสแบบเรียลไลเซชันเป็นสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุ

จากนั้นได้พัฒนาเครื่องมือที่ประยุกต์ใช้กฎการแปลงยูเอ็มแอลคลาสไดอะแกรม เป็นสคีมาฐานข้อมูลเชิงวัตถุเพื่อแสดงว่ากฎที่ได้ออกแบบไว้สามารถนำมาประยุกต์ใช้งานได้จริง โดยรายละเอียดการทดสอบมีดังนี้

การทดสอบที่ 1 การแปลงคลาสไดอะแกรมโดยการประยุกต์ใช้กฎทั้ง 11 ข้อ ของฐานข้อมูลทั้ง 3 ชนิด โดยใช้เครื่องมือที่สร้างเป็นตัวแปลงคลาสไดอะแกรมที่สร้างขึ้น

การทดสอบที่ 2 การแปลงคลาสไดอะแกรมโดยการประยุกต์ใช้กฎทั้ง 11 ข้อ ของฐานข้อมูลทั้ง 3 ชนิด โดยการประยุกต์ใช้กฎโดยการแปลงด้วยตัวเอง

การทดสอบที่ 3 การทดสอบใช้คำสั่งเพื่อดึงข้อมูลจากฐานข้อมูลที่สร้างขึ้นเพื่อตรวจสอบความถูกต้องของสคีมาของฐานข้อมูล

โดยการตรวจสอบความถูกต้องของการแปลงจะมี 2 ส่วนคือ

1. การตรวจสอบความถูกต้องของการแปลงคลาสไดอะแกรมเป็นสคีมาฐานข้อมูลเชิงวัตถุ โดยการทดสอบดึงข้อมูลจากฐานข้อมูลที่ถูกสร้างขึ้นเพื่อทดสอบว่าฐานข้อมูลแต่ละชนิดสามารถแสดงผลลัพธ์จากการดึงข้อมูลออกมาได้ถูกต้องตามที่ออกแบบไว้หรือไม่ การทดสอบจะใช้ภาษาสำหรับค้นหาข้อมูลของฐานข้อมูลแต่ละชนิด โดยการระบุชื่อแอทริบิวต์ที่ต้องการจากคลาสไดอะแกรมในภาษาสำหรับค้นหาข้อมูลของฐานข้อมูลแต่ละชนิด

2.การพิจารณาถึงผลของการทดสอบจาก

1. จำนวนคลาสที่ถูกสร้างทั้งหมด
2. จำนวนแอทริบิวต์ที่ถูกสร้างทั้งหมด
3. จำนวนเมทธอดที่ถูกสร้างทั้งหมด
4. จำนวนความสัมพันธ์ที่ถูกสร้างทั้งหมด
5. ความถูกต้องจากการทดสอบดึงข้อมูลจากฐานข้อมูลที่สร้างทั้งหมด

จากการทดสอบข้างต้นได้ข้อสรุปดังนี้

เมื่อทดสอบกฎการแปลงคลาสไดอะแกรมเป็นสคีมาฐานข้อมูลเชิงวัตถุโดยอาศัยคลาสไดอะแกรมดังรูปที่ 6.2 จะพบว่า จำนวนคลาส แอทริบิวต์ เมทธอด ความสัมพันธ์ต่าง ๆ ที่ปรากฏในคลาสไดอะแกรม เมื่อทดสอบโดยการนับจำนวนของส่วนประกอบต่าง ๆ ที่ปรากฏในคลาสไดอะแกรมและในสคีมาฐานข้อมูลเชิงวัตถุจะพบว่ามีจำนวนเท่ากันดังแสดงในตารางที่ 6.13-6.17 และเมื่อทดสอบความถูกต้องของส่วนประกอบต่าง ๆ ที่สร้างในฐานข้อมูลเชิงวัตถุจากคลาสไดอะ แกรมโดยการทดสอบดึงข้อมูลจากฐานข้อมูลจะพบว่าข้อมูลที่ดึงออกมามีความถูกต้องทั้งหมดทั้ง 3 กรณี ดังแสดงในตารางที่ 6.17 ดังนั้นจึงสรุปได้ว่ากฎการแปลงทั้ง 11 ข้อสามารถแปลงข้อมูลเป็นสคีมาฐานข้อมูลเชิงวัตถุได้ถูกต้องและสามารถใช้งานได้จริงทั้ง 3 ชนิดของฐานข้อมูล

7.2 ข้อเสนอแนะ

1. เนื่องจากกฎการแปลงยูเอ็มแอลคลาสไดอะแกรมเป็นสคีมาฐานข้อมูลเชิงวัตถุ ได้นำเอาเฉพาะคลาสไดอะแกรมมาใช้ในการพิจารณา เพื่อสร้างเป็นฐานข้อมูล ดังนั้นข้อมูลต่างๆ ที่ถูกสร้างขึ้นในฐานข้อมูลอาจจะยังไม่เพียงพอตามที่ต้องการมากนักเช่นการทำงานของเมทธอดภายในวัตถุ ดังนั้นจึงควรนำเอาไดอะแกรมประเภทอื่น ๆ มาช่วยในการแปลงเป็นฐานข้อมูลด้วย

2. เนื่องจากสคีมาฐานข้อมูลที่ได้หลังจากการแปลงโดยการใช้เครื่องมือ จะยังไม่ถูกนำไปสร้างเป็นฐานข้อมูลทันที ต้องนำเอาสคีมาที่ได้ดังกล่าวแปลงเป็นฐานข้อมูลเองโดยผู้ใช้งาน ดังนั้นควรสร้างเครื่องมือเพื่อนำเอาสคีมาฐานข้อมูลที่ได้จากการแปลงโดยการใช้เครื่องมือดังกล่าวนำเข้าไปสร้างเป็นฐานข้อมูลโดยอัตโนมัติ

3. เนื่องจากสคีมาฐานข้อมูลเชิงวัตถุที่ได้จากการแปลงโดยใช้เครื่องมือดังกล่าว จะมีแค่การสร้างชื่อเมทธอดที่จำเป็นต้องใช้ภายในตัววัตถุที่อยู่ในฐานข้อมูลเท่านั้น ยังไม่มีการอธิบายรายละเอียดการทำงานของเมทธอด ดังนั้นจึงควรมีการสร้างเครื่องมือที่สามารถอธิบายการทำงานของเมทธอดภายในคลาสด้วย

4. เนื่องจากสคีมาฐานข้อมูลเชิงวัตถุที่ได้ ยังไม่ครอบคลุมฐานข้อมูลเชิงวัตถุในเชิงพาณิชย์ชนิดอื่น ๆมากนัก ดังนั้นจึงน่าจะมีการสร้างกฎที่ครอบคลุมฐานข้อมูลเชิงวัตถุประเภทอื่น ๆด้วย โดยการสร้างเป็นกฎมาตรฐานที่ฐานข้อมูลทุกชนิดสามารถนำมาตรฐานนี้มาใช้งานได้

5. เนื่องจากสคีมาฐานข้อมูลเชิงวัตถุที่ได้มีการสร้างความสัมพันธ์ระหว่างคลาสเกิดขึ้น แต่ความสัมพันธ์ระหว่างคลาสอาจจะไม่ครอบคลุมทุกกรณี ดังนั้นจึงควรสร้างกฎที่สามารถครอบคลุมประเภทความสัมพันธ์แบบต่าง ๆให้เพิ่มมากขึ้น ครอบคลุมความสัมพันธ์ตามมาตรฐานของยูเอ็มแอลที่มีอยู่

รายการอ้างอิง

1. ชาลี วรกุลพิพัฒน์ และเทพฤทธิ์ บัณฑิตวัฒนาวงศ์. UML ภาษามาตรฐานเพื่อผู้พัฒนาซอฟต์แวร์. กรุงเทพฯ : ซีเอ็ดยูเคชัน , 2544.

2. มารุต ศิลปสุนทร. การออกแบบวิธีการสร้างแฟ้มสัญลักษณ์แสดงตนสำหรับการสอบถามข้อมูลในระบบการจัดการฐานข้อมูลเชิงวัตถุ. วิทยานิพนธ์ปริญญามหาบัณฑิต สาขาวิทยาศาสตร์คอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย , 2543.

3. มฑุปายาส ทองมาก . การออกแบบกฎการแปลงยูเอ็มแอลซีเควนซ์ไดอะแกรมเป็นชุดคำสั่งภาษาจาวา. วิทยานิพนธ์ปริญญามหาบัณฑิต สาขาวิทยาศาสตร์คอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย , 2545.

4. อภิรักษ์ โชติกิตติพร. การแปลงฐานข้อมูลเชิงสัมพันธ์เป็นฐานข้อมูลเชิงวัตถุ. วิทยานิพนธ์ปริญญามหาบัณฑิต สาขาวิทยาศาสตร์คอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย , 2539.

5. Dong Hyuk Park,Soo Dong Kim. XML Rule Based Source Code Generator for UML Case Tool. Asia-Pacific Software Engineering Conference(APSEC2001), (2001) : 53-60.

6. Grady Booch, James Rumbaugh, and Ivar Jacobson. The Unified Modeling Language User Guide. Massachusetts : Addison-Wesley, 1999.

7. Il-Yeol Song and Heather M.Godsey. A Knowlede Based Object-oriented Database Schema Generator. IEEE Computer (July ,1993) : 160 - 167 .

8. James Rumbaugh, Ivar Jacobson, and Grady Booch . The Unified Modeling Language Reference Manual. Massachusetts : Addison-Wesley, 1999.

9. Kim ,W . Object –Oriented Database :Definition and research Direction. IEEE Transaction on Knowledge and data Engineering (1993) : 327-341.

10. Ramez Elmarsri and Shamkant B.Navathe. Fundamentals Of Database Systems. Massachusetts : Addison-Wesley, 1997.

11. Schach R.S. Classical and Object –Oriented Software engineering with UML and JAVA. 2 nd ed. USA. : McGraw-Hill, 1993.

12. Sethi,Ravi.Programming languages:concepts &construct. Massachusetts : Addison-Wesley, 1996.

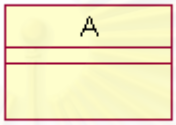
13. Sutheera Puntheeranurak and Supamit Chittayasothon. An Extended NIAM Conceptual schema model for object database. IEEE Computer , (2001) : 250-257.

ภาคผนวก

ภาคผนวก ก.

ตัวอย่างการประยุกต์ใช้กฎการแปลงคลาสไดอะแกรม เป็นสคีมาฐานข้อมูลเชิงวัตถุ

กฎข้อที่ 1 กฎการแปลงคลาสและอินเตอร์เฟสเป็นสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุ



รูปที่ ก.1 แสดงคลาสไดอะแกรมที่มีลักษณะเป็นคลาส

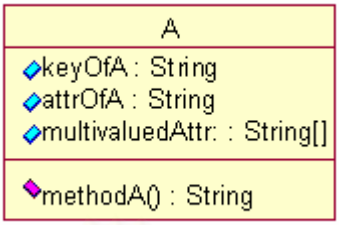
จากกฎข้อที่ 1 และไวยากรณ์จะได้สคีมาตัวกลางของสคีมาฐานข้อมูลเชิงวัตถุ นำมาแปลงเป็นสคีมาฐานข้อมูลเชิงวัตถุที่สมบูรณ์ตามประเภทของฐานข้อมูลแต่ละชนิดได้ดังนี้

ตารางที่ ก.1 แสดงสคีมาฐานข้อมูลเชิงวัตถุของคลาสไดอะแกรมจากกฎข้อที่1

กรณีที่ 1สคีมาฐานข้อมูลเชิงวัตถุตามมาตรฐานโอดีเอ็มจี	กรณีที่ 2 สคีมาฐานข้อมูลเชิงวัตถุฐานข้อมูลคาเช่	กรณีที่ 3 สคีมาฐานข้อมูลเชิงวัตถุฐานข้อมูลแมทิส

class A(extent As)
{
 	
}
	Class Package.A Extends
	%Persistent [ClassType =
	persistent, ProcedureBlock]
	{
	}
		Interface A: persistent
		{
		
		}

กฎข้อที่ 2 กฎการแปลงแอทริบิวต์และเมทธอดเป็นสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุ



รูปที่ ก.2 แสดงคลาสไดอะแกรมที่มีแอทริบิวต์และเมทธอด

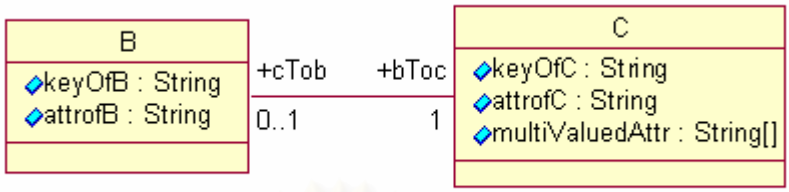
จากกฎข้อที่ 2 และไวยากรณ์จะได้สคีมาตัวกลางของสคีมาฐานข้อมูลเชิงวัตถุ นำมาแปลงเป็นสคีมาฐานข้อมูลเชิงวัตถุที่สมบูรณ์ตามประเภทของฐานข้อมูลแต่ละชนิดได้ดังนี้

ตารางที่ ก.2 แสดงสคีมาฐานข้อมูลเชิงวัตถุของคลาสไดอะแกรมจากกฎข้อที่ 2

กรณีที่ 1สคีมาฐานข้อมูลเชิงวัตถุตามมาตรฐานโอดีเอ็มจี	กรณีที่ 2 สคีมาฐานข้อมูลเชิงวัตถุฐานข้อมูลคาเช่	กรณีที่ 3 สคีมาฐานข้อมูลเชิงวัตถุฐานข้อมูลแมทิส

class A
(extent As key keyOfA)
{
attribute string keyOfA;
attribute string attrOfA;
attribute set<string>
multivaluedAttr;
string methodA();
}
	Class Package.A Extends
	%Persistent [ClassType =
	persistent, ProcedureBlock]
	{
	Property keyOfA As %String;
	Property attrOfA As %String;
	Property multivaluedAttr As
	%String[Collection=array];
	Method methodA()
	As %String{}
	}
		Interface A: persistent
		{
		Attribute string keyOfA;
		Attribute string attrOfA;
		Attribute Set<String>
		multiValuedAttr;
		mt_method
		"CREATE METHOD methodA()
		RETURNS String FOR A
		BEGIN
		DECLARE methodA String;
		RETURN methodA;
		END;";
		}

กฎข้อที่ 3 กฎการแปลงความสัมพันธ์ระหว่างคลาสแบบแอสโซซิเอชันเป็นสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุ



รูปที่ ก.3 แสดงคลาสไดอะแกรมที่มีความสัมพันธ์ระหว่างคลาสแบบแอสโซซิเอชัน

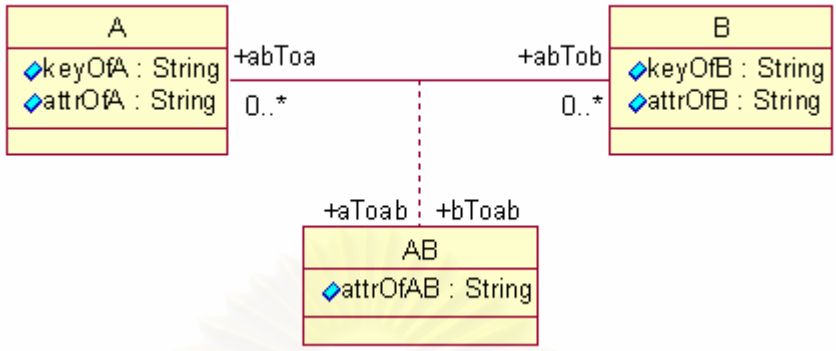
จากกฎข้อที่ 3 และไวยากรณ์จะได้สคีมาตัวกลางของสคีมาฐานข้อมูลเชิงวัตถุ นำมาแปลงเป็นสคีมาฐานข้อมูลเชิงวัตถุที่สมบูรณ์ตามประเภทของฐานข้อมูลแต่ละชนิดได้ดังนี้

ตารางที่ ก.3 แสดงสคีมาฐานข้อมูลเชิงวัตถุของคลาสไดอะแกรมจากกฎข้อที่ 3

กรณีที่ 1สคีมาฐานข้อมูลเชิงวัตถุตามมาตรฐานโอดีเอ็มจี	กรณีที่ 2 สคีมาฐานข้อมูลเชิงวัตถุฐานข้อมูลคาเช่	กรณีที่ 3 สคีมาฐานข้อมูลเชิงวัตถุฐานข้อมูลแมทิส

class B
(extent Bs
key keyOfB)
{
attribute string attrOfB;
attribute string keyOfB;
relationship C bTOc inverse C::cTOb;
}
class C
(extent extentOfC
keyOfC)
{
attribute string attrOfC;
attribute string keyOfC;
attribute set<String> multiValuedAttr;
relationship B cTOb inverse B::bTOc;
}
	Class Package.B Extends
	%Persistent [ClassType = persistent, ProcedureBlock]
	{
	Property keyOfB As%String;
	Property attrOfB As %String;
	Relationship bToc As C
	[Cardinality = one,
	Inverse = cTob];
	}
	Class Package.C Extends
	%Persistent [ClassType = persistent, ProcedureBlock]
	{
	Property keyOfC As%String;
	Property attrOfC As %String;
	Relationship cTob As B
	[Cardinality = one,
	Inverse = bToc];
	}
		Interface B:persistent
		{
		attribute String attrOfB;
		attribute String keyOfB;
		relationship C bTOc[1,1]
		inverse C::cTOb;
		}
		Interface C:persistent
		{
		attribute string attrOfC;
		attribute string keyOfC;
		attribute Set<String> multiValuedAttr;
		relationship B cTOb[0,1]
		inverse B::bTOc;
		}

กฎข้อที่ 4 กฎการแปลงความสัมพันธ์ระหว่างคลาสแบบแอสโซซิเอชันคลาสเป็นสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุ



รูปที่ ก.4 แสดงคลาสไดอะแกรมที่มีความสัมพันธ์ระหว่างคลาสแบบแอสโซซิเอชันคลาส

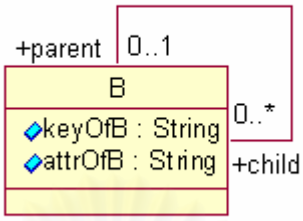
จากกฎข้อที่ 4 และไวยากรณ์จะได้สคีมาตัวกลางของสคีมาฐานข้อมูลเชิงวัตถุ นำมาแปลงเป็นสคีมาฐานข้อมูลเชิงวัตถุที่สมบูรณ์ตามประเภทของฐานข้อมูลแต่ละชนิดได้ดังนี้

ตารางที่ ก.4 แสดงสคีมาฐานข้อมูลเชิงวัตถุของคลาสไดอะแกรมจากกฎข้อที่ 4

กรณีที่ 1สคีมาฐานข้อมูลเชิงวัตถุตามมาตรฐานโอดีเอ็มจี	กรณีที่ 2 สคีมาฐานข้อมูลเชิงวัตถุฐานข้อมูลคาเช่	กรณีที่ 3 สคีมาฐานข้อมูลเชิงวัตถุฐานข้อมูลแมทิส

class A
(extent As key keyOfA)
{
attribute string keyOfA;
attribute string attrOfA;
relationship set<AB> aTOab
inverse AB::abTOa;
}
class B
(extent Bs key keyOfB)
{
attribute string keyOfB;
attribute string attrOfB;
relationship set<AB> bTOab
inverse AB::abTOb;
}
 	
class AB(extent ABs)
{
attribute string attrOfAB;
relationship A abTOa inverse
A::aTOab;
relationship B abTOb inverse
B::bTOab;
}
	Class Package.A Extends
	%Persistent [ClassType =
	persistent, ProcedureBlock]
	{
	Property keyOfA As %String;
	Property attrOfA As %String;
	Relationship aToab As AB
	[Cardinality = many,
	Inverse = abToa];
	}
	Class Package.B Extends
	%Persistent [ClassType =
	persistent, ProcedureBlock]
	{
	Property keyOfB As %String;
	Property attrOfB As %String;
	Relationship bToab As AB
	[Cardinality = many,
	Inverse = abTob];
	}
	Class Package.AB Extends
	%Persistent [ClassType =
	persistent, ProcedureBlock]
	{
	Property attrOfAB As %String;
	Relationship abToa As A
	[Cardinality = one,
	Inverse = aToab];
	Relationship abTob As B
	[Cardinality = one,
	Inverse = bToab];
	}
		Interface A:persistent
		{
		Attribute String keyOfA;
		Attribute String attrOfA;
		Relationship Set <AB>
		aTOab[0,-1] inverse
		AB::abTOa;
		}
		Interface B:persistent
		{
		Attribute String keyOfB;
		Attribute String attrOfB;
		relationship Set <AB>
		bTOab[0,-1] inverse
		AB::abTOb;
		}
		Interface AB:persistent
		{
		attribute String attrOfAB;
		relationship A abTOa[1,1]
		inverse A::aTOab;
		relationship B abTOb[1,1]
		inverse B::bTOab;
		}

กฎข้อที่ 5 กฎการแปลงความสัมพันธ์ระหว่างคลาสแบบรีเคอร์ซีพแอสโซซิเอชันเป็นสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุ



รูปที่ ก.5 แสดงคลาสไดอะแกรมที่มีความสัมพันธ์ระหว่างคลาสแบบรีเคอร์ซีพแอสโซซิเอชัน

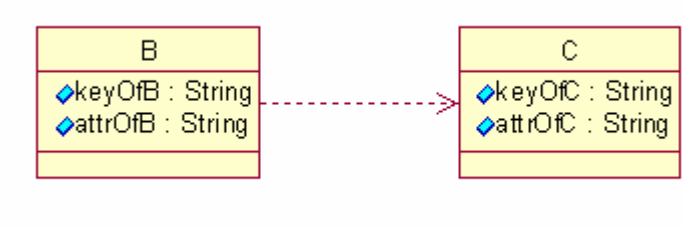
จากกฎข้อที่ 5 และไวยากรณ์จะได้สคีมาตัวกลางของสคีมาฐานข้อมูลเชิงวัตถุ นำมาแปลงเป็นสคีมาฐานข้อมูลเชิงวัตถุที่สมบูรณ์ตามประเภทของฐานข้อมูลแต่ละชนิดได้ดังนี้

ตารางที่ ก.5 แสดงสคีมาฐานข้อมูลเชิงวัตถุของคลาสไดอะแกรมจากกฎข้อที่ 5

กรณีที่ 1สคีมาฐานข้อมูลเชิงวัตถุตามมาตรฐานโอดีเอ็มจี	กรณีที่ 2 สคีมาฐานข้อมูลเชิงวัตถุฐานข้อมูลคาเช่	กรณีที่ 3 สคีมาฐานข้อมูลเชิงวัตถุฐานข้อมูลแมทิส

class B
(extent Bs key keyOfB)
{
attribute string keyOfB;
attribute string attrOfB;
relationship B parent inverse
B::child;
relationship set<B> child
inverse B::parent;
}
	Class Package.B Extends
	%Persistent [ClassType =
	persistent, ProcedureBlock]
	{
	Property keyOfB As %String;
	Property attrOfB As %String;
	Relationship parent As B
	[Cardinality = one,
	Inverse = child];
	Relationship child As B
	[Cardinality = many,
	Inverse = parent];
	}
		interface B:persistent
		{
		Attribute String keyOfB;
		Attribute String attrOfB;
		relationship B parent [0,1]
		inverse B::child;
		relationship Set<B>
		child[0,-1] B::parent;
		}

กฎข้อที่ 6 กฎการแปลงความสัมพันธ์ระหว่างคลาสแบบดีเพนเดนซีเป็นสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุ



รูปที่ ก.6 แสดงคลาสไดอะแกรมที่มีความสัมพันธ์แบบดีเพนเดนซี

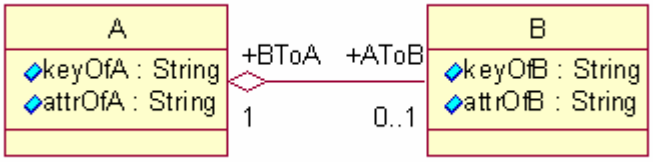
จากกฎข้อที่ 6 และไวยากรณ์จะได้สคีมาตัวกลางของสคีมาฐานข้อมูลเชิงวัตถุ นำมาแปลงเป็นสคีมาฐานข้อมูลเชิงวัตถุที่สมบูรณ์ตามประเภทของฐานข้อมูลแต่ละชนิดได้ดังนี้

ตารางที่ ก.6 แสดงสคีมาฐานข้อมูลเชิงวัตถุของคลาสไดอะแกรมจากกฎข้อที่ 6

กรณีที่ 1สคีมาฐานข้อมูลเชิงวัตถุตามมาตรฐานโอดีเอ็มจี	กรณีที่ 2 สคีมาฐานข้อมูลเชิงวัตถุฐานข้อมูลคาเช่	กรณีที่ 3 สคีมาฐานข้อมูลเชิงวัตถุฐานข้อมูลแมทิส

class B
(extent Bs key keyOfB)
{
attribute string keyOfB;
attribute string attrOfB;
attribute C bc;
}
class C
(extent Cs keyOfC)
{
attribute string keyOfC;
attribute string attrOfC;
}
	Class Package.B Extends
	%Persistent [ClassType =
	persistent, ProcedureBlock]
	{
	Property keyOfB As %String;
	Property attrOfB As %String;
	Property bc As %C;
	}
	Class Package.C Extends
	%Persistent [ClassType =
	persistent, ProcedureBlock]
	{
	Property keyOfC As %String;
	Property attrOfC As %String;
	}
		interface B:persistent
		{
		attribute String keyOfB;
		attribute String attrOfB;
		attribute C bc;
		}
		interface C:persistent
		{
		attribute String keyOfC;
		attribute String attrOfC;
		}

กฎข้อที่ 7 กฎการแปลงความสัมพันธ์ระหว่างคลาสสัมพันธ์แบบแอกกรีเกชันเป็นสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุ



รูปที่ ก.7 แสดงรูปแบบตามยูเอ็มแอลคลาสไดอะแกรมที่มีความสัมพันธ์แบบแอกกรีเกชัน

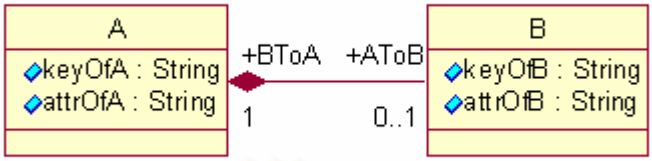
จากกฎข้อที่ 7 และไวยากรณ์จะได้สคีมาตัวกลางของสคีมาฐานข้อมูลเชิงวัตถุ นำมาแปลงเป็นสคีมาฐานข้อมูลเชิงวัตถุที่สมบูรณ์ตามประเภทของฐานข้อมูลแต่ละชนิดได้ดังนี้

ตารางที่ ก.7 แสดงสคีมาฐานข้อมูลเชิงวัตถุของคลาสไดอะแกรมจากกฎข้อที่ 7

กรณีที่ 1สคีมาฐานข้อมูลเชิงวัตถุตามมาตรฐานโอดีเอ็มจี	กรณีที่ 2 สคีมาฐานข้อมูลเชิงวัตถุฐานข้อมูลคาเช่	กรณีที่ 3 สคีมาฐานข้อมูลเชิงวัตถุฐานข้อมูลแมทิส

class B
(extent Bs key keyOfB)
{
attribute string keyOfB;
attribute string attrOfB;
relationship A bToA inverse
A:: AToB;
}
class A
(extent As key keyOfA)
{
attribute string keyOfA;
attribute string attrOfA;
attribute B b;
relationship B AToB
inverse B::bToA;
}
	Class Package.B Extends
	%Persistent [ClassType =
	persistent, ProcedureBlock]
	{
	Property keyOfB as %String;
	Property attrOfB As %String;
	Relationship bToA As A
	[Cardinality = one,
	Inverse = AToB];
	}
	Class Package.A Extends
	%Persistent [ClassType =
	persistent, ProcedureBlock]
	{
	Property keyOfA As %String;
	Property attrOfA As %String;
	Property b As %B;
	Relationship AToB As B
	[Cardinality = one,
	Inverse = bToA];
	
	}
		interface B:persistent
		{
		attribute String keyOfB;
		attribute String attrOfB;
		relationship A bToA [1,1]
		inverse A:: AToB;
		}
		interface A:persistent
		{
		attribute String keyOfA;
		attribute String attrOfA;
		attribute B b;
		relationship B AToB[0,1]
		inverse B:: bToA;
		}

กฎข้อที่ 8 กฎการแปลงความสัมพันธ์ระหว่างคลาสแบบคอมโพสิชันเป็นสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุ



รูปที่ ก.8 แสดงรูปแบบตามยูเอ็มแอลคลาสไดอะแกรมที่มีความสัมพันธ์แบบคอมโพสิชัน

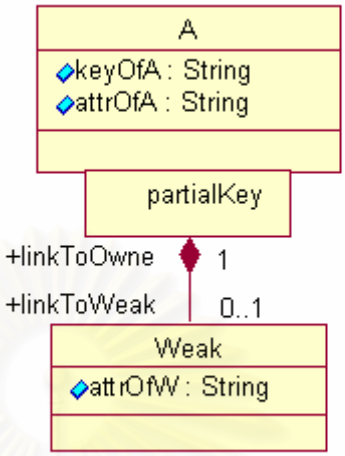
จากกฎข้อที่ 8 และไวยากรณ์จะได้สคีมาตัวกลางของสคีมาฐานข้อมูลเชิงวัตถุ นำมาแปลงเป็นสคีมาฐานข้อมูลเชิงวัตถุที่สมบูรณ์ตามประเภทของฐานข้อมูลแต่ละชนิดได้ดังนี้

ตารางที่ ก.8 แสดงสคีมาฐานข้อมูลเชิงวัตถุของคลาสไดอะแกรมจากกฎข้อที่ 8

กรณีที่ 1สคีมาฐานข้อมูลเชิงวัตถุตามมาตรฐานโอดีเอ็มจี	กรณีที่ 2 สคีมาฐานข้อมูลเชิงวัตถุฐานข้อมูลคาเช่	กรณีที่ 3 สคีมาฐานข้อมูลเชิงวัตถุฐานข้อมูลแมทิส

class B
(extent Bs key keyOfB)
{
attribute string keyOfB;
attribute string attrOfB;
relationship A bToA
inverse A:: AToB;
}
class A
(extent As key keyOfA)
{
attribute string keyOfA;
attribute string attrOfA;
attribute B b;
attribute string keyOfB;
relationship B AToB
inverse B::bToA;
}
	Class Package.B Extends
	%Persistent [ClassType =
	persistent, ProcedureBlock]
	{
	Property keyOfB as %String;
	Property attrOfB As %String;
	Relationship bToA As A
	[Cardinality = one,
	Inverse = AToB];
	}
	Class Package.A Extends
	%Persistent [ClassType =
	persistent, ProcedureBlock]
	{
	Property keyOfA As %String;
	Property attrOfA As %String;
	Property b As % B;
	Property keyOfB As %String;
	Relationship AToB As B
	[Cardinality = one,
	Inverse = bToA];
	}
		interface B:persistent
		{
		attribute String keyOfB;
		attribute String attrOfB;
		relationship A bToA [1,1]
		inverse A:: AToB;
		}
		interface A:persistent
		{
		attribute String keyOfA;
		attribute String attrOfA;
		attribute B b;
		attribute string keyOfB;
		relationship B AToB[0,1]
		inverse B:: bToA;
		}

กฎข้อที่ 9 กฎการแปลงความสัมพันธ์ระหว่างคลาสแบบคอลลิไฟล์แอสโซซิเอชันเป็นสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุ



รูปที่ ก.9 แสดงคลาสไดอะแกรมที่มีความสัมพันธ์แบบคอลลิไฟล์แอสโซซิเอชัน

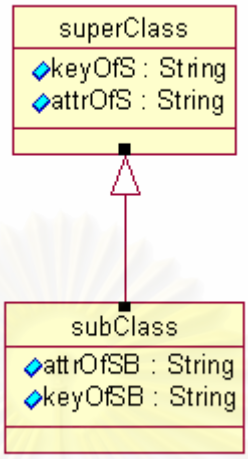
จากกฎข้อที่ 9 และไวยากรณ์จะได้สคีมาตัวกลางของสคีมาฐานข้อมูลเชิงวัตถุ นำมาแปลงเป็นสคีมาฐานข้อมูลเชิงวัตถุที่สมบูรณ์ตามประเภทของฐานข้อมูลแต่ละชนิดได้ดังนี้

ตารางที่ ก.9 แสดงสคีมาฐานข้อมูลเชิงวัตถุของคลาสไดอะแกรมจากกฎข้อที่ 9

กรณีที่ 1สคีมาฐานข้อมูลเชิงวัตถุตามมาตรฐานโอดีเอ็มจี	กรณีที่ 2 สคีมาฐานข้อมูลเชิงวัตถุฐานข้อมูลคาเช่	กรณีที่ 3 สคีมาฐานข้อมูลเชิงวัตถุฐานข้อมูลแมทิส

class A
(extent As key keyOfA)
{
attribute string keyOfA;
attribute string attrOfA;
relationship set<Weak>
linkToWeak inverse
Weak::linkToOwner;
}
class Weak
(extent Weaks key
(partialKey, keyOfA))
{
attribute string partialKey;
attribute string keyOfA;
attribute string attrOfW;
relationship A linkToOwner
inverse A:: linkToWeak;
}
	Class Package.A Extends
	%Persistent [ClassType =
	persistent, ProcedureBlock]
	{
	Property keyOfA As %String;
	Property attrOfA As %String;
	Relationship linkToWeak As Weak
	[Cardinality = one,
	Inverse = linkToOwner];
	}
	Class Package. Weak Extends
	%Persistent [ClassType =
	persistent, ProcedureBlock]
	{
	Property keyOfA As %String;
	Property attrOfW As%String;
	Property partialKey As%String;
	Relationship linkToOwner As A
	[Cardinality = one,
	Inverse = linkToWeak];
	}
		interface A:persistent
		{
		attribute String keyOfA;
		attribute String attrOfA;
		relationship Set<Weak>
		linkToWeak[0,1] inverse
		Weak::linkToOwner;
		}
		interface Weak:persistent
		{
		attribute String partialKey;
		attribute String keyOfA;
		attribute String attrOfW;
		relationship A
		linkToOwner[1,1] inverse
		A:: linkToWeak;
		}

กฎข้อที่ 10 กฎการแปลงความสัมพันธ์ระหว่างคลาสแบบเจนเนอรัลไลเซชันเป็นสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุ



รูปที่ ก.10 แสดงรูปแบบตามคลาสไดอะแกรมที่มีความสัมพันธ์ระหว่างคลาสแบบเจนเนอรัลไลเซชัน

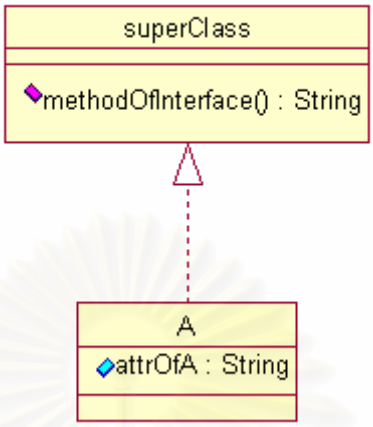
จากกฎข้อที่ 10 และไวยากรณ์จะได้สคีมาตัวกลางของสคีมาฐานข้อมูลเชิงวัตถุ นำมาแปลงเป็นสคีมาฐานข้อมูลเชิงวัตถุที่สมบูรณ์ตามประเภทของฐานข้อมูลแต่ละชนิดได้ดังนี้

ตารางที่ ก.10 แสดงสคีมาฐานข้อมูลเชิงวัตถุของคลาสไดอะแกรมจากกฎข้อที่ 10

กรณีที่ 1สคีมาฐานข้อมูลเชิงวัตถุตามมาตรฐานโอดีเอ็มจี	กรณีที่ 2 สคีมาฐานข้อมูลเชิงวัตถุฐานข้อมูลคาเช่	กรณีที่ 3 สคีมาฐานข้อมูลเชิงวัตถุฐานข้อมูลแมทิส

class superClass
(extent superClasss
key keyOfS)
{
attribute string attrOfS;
attribute string keyOfS;
}
class subClass
extends superClass
(extent subClasss
key keySB)
{
attribute string attrSB;
attribute string keySB;
}
	Class Package.superClass
	Extends %Persistent [ClassType = persistent, ProcedureBlock]
	{
	Property attrOfS As %String;
	Property keyOfS As %String;
	}
	Class Package.subclass Extends % superClass [ClassType = persistent, ProcedureBlock]
	{
	Property keySB As %String;
	Property attrSB As %String;
	}
		interface superClass:persistent
		{
		attribute String attrOfS;
		attribute String keyOfS;
		}
		interface subClass:
		superClass:persistent
		{
		attribute String attrSB;
		attribute String keySB;
		}

กฎข้อที่ 11 กฎการแปลงความสัมพันธ์ระหว่างคลาสแบบเรียลไลเซชันเป็นสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุ



รูปที่ ก.11 แสดงรูปแบบตามคลาสไดอะแกรมที่มีความสัมพันธ์ระหว่างคลาสแบบเรียลไลเซชัน

จากกฎข้อที่ 11 และไวยากรณ์จะได้สคีมาตัวกลางของสคีมาฐานข้อมูลเชิงวัตถุ นำมาแปลงเป็นสคีมาฐานข้อมูลเชิงวัตถุที่สมบูรณ์ตามประเภทของฐานข้อมูลแต่ละชนิดได้ดังนี้

ตารางที่ ก.11 แสดงสคีมาฐานข้อมูลเชิงวัตถุของคลาสไดอะแกรมจากกฎข้อที่ 11

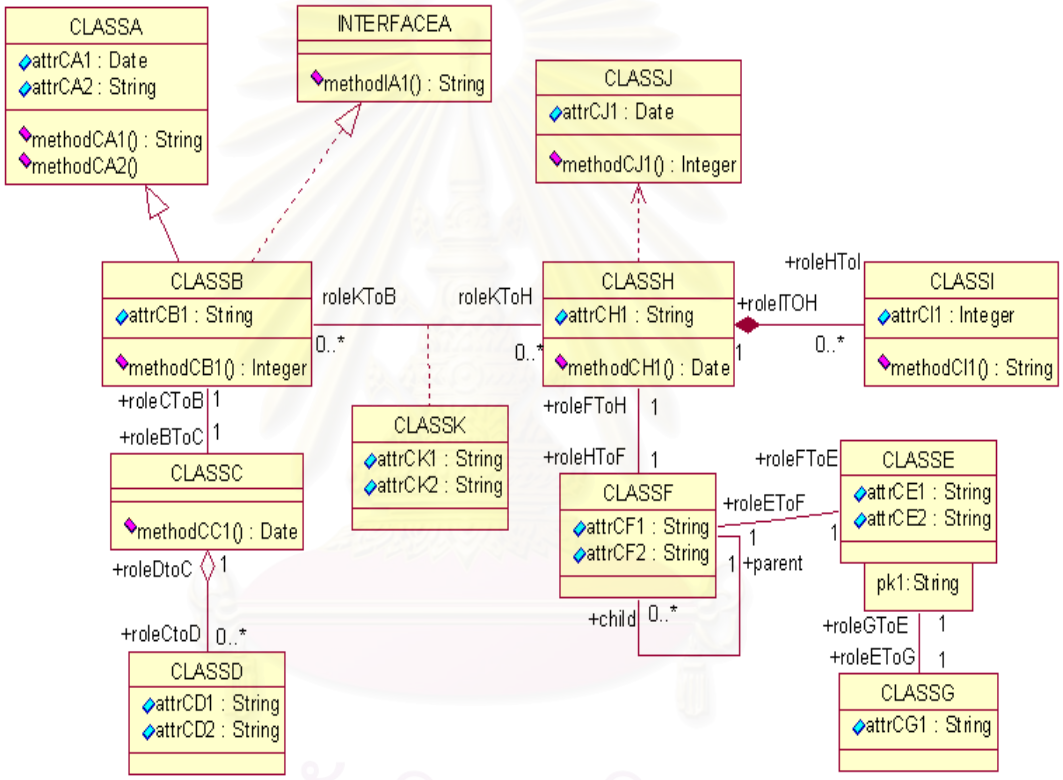
กรณีที่ 1สคีมาฐานข้อมูลเชิงวัตถุตามมาตรฐานโอดีเอ็มจี	กรณีที่ 2 สคีมาฐานข้อมูลเชิงวัตถุฐานข้อมูลคาเช่	กรณีที่ 3 สคีมาฐานข้อมูลเชิงวัตถุฐานข้อมูลแมทิส

interface InterfaceA
{
string methodinterface();
}
class A: InterfaceA
(extent As key keyOfA)
{
attribute string attrOfA;
}
	Interface Package.InterfaceA
	{
	method Methodinterface()
	As %String{ }
	}
	Class Package.A :InterfaceA
	[ClassType = persistent,
	ProcedureBlock]
	{
	Property attrOfA As String;
	}
		interface InterfaceA
		{
		mt_method
		"CREATE METHOD
		Methodinterface ()
		RETURNS STRING FOR
		InterfaceA
		BEGIN
		DECLARE
		Methodinterface STRING;
		RETURN Methodinterface;
		END;";
		}
		interface A:InterfaceA:
		persistent
		{
		attribute String attrOfA;
		}

ภาคผนวก ข

ตัวอย่างการประยุกต์ใช้กฎทั้ง 11 ข้อกับระบบที่ออกแบบโดยคลาสไดอะแกรม

1.รูปแบบตามคลาสไดอะแกรม



รูปที่ ข.1 แสดงระบบที่ออกแบบโดยคลาสไดอะแกรม

2. รูปแบบสคีมาฐานข้อมูลเชิงวัตถุของโอดีเอ็มจีโดยการใช้ภาษานิยามเชิงวัตถุ

```
class CLASSA(extent CLASSAs)
{    attribute date attrCA1;
     attribute string attrCA2;
     string  methodCA1():
     string  methodCA2():
 }
  class INTERFACEA ( extent  INTERFACEAs)
{     string  methodIA1():
}
class CLASSB extends CLASSA: INTERFACEA   (extent CLASSBs)
{    attribute string attrCB1;
     string  methodCB1():
     relationship  set<CLASSK>  roleBToK  inverse    CLASSK:: roleKToB;
     relationship  CLASSC  roleBToC  inverse    CLASSC:: roleCToB ;
}
 class CLASSC (extent CLASSCs)
{    date methodCC1():
     attribute CLASSD   classd;
     relationship  set<CLASSD>  roleCToD  inverse    CLASSD:: roleDToC;
     relationship  CLASSB  roleCToB  inverse    CLASSB:: roleBToC ;
}
class CLASSD   ( extent CLASSDs)
{      attribute string attrCD1;
       attribute string attrCD2;
       relationship  CLASSC  roleDToC  inverse    CLASSC:: roleCToD ;
 }
class CLASSE(extent CLASSEs)
{     attribute string attrCE1;
      attribute string attrCE2;
      relationship CLASSF  roleEToF  inverse    CLASSF  :: roleFToE  ;
      relationship CLASSG  roleEToG  inverse    CLASSG  :: roleGToE  ;
 }
class CLASSF ( extent CLASSFs)
 {     attribute string attrCF1;
       attribute string attrCF2;
       relationship set<CLASSF> child  inverse    CLASSF:: parent;
       relationship CLASSF parent inverse    CLASSF:: child  ;
       relationship CLASSE roleFToE  inverse    CLASSE:: roleEToF  ;
       relationship CLASSH roleFToH  inverse    CLASSH:: roleHToF  ;
}
class CLASSG ( extent CLASSGs)
 {     attribute string attrCG1;
       attribute string pk1;
       relationship CLASSE  roleGToE  inverse    CLASSE  :: roleEToG  ;
}
class CLASSH ( extent  CLASSHs)
 {     attribute string attrCH1;
       attribute CLASSJ classj;
       attribute CLASSI classi;
       attribute string attrCI1;
       date  methodCH1():
       relationship set<CLASSI>  roleHToI  inverse    CLASSI:: roleIToH  ;
       relationship set<CLASSK>  roleHToK  inverse    CLASSK:: roleKToH  ;
       relationship CLASSF  roleHToF  inverse    CLASSF  :: roleFToH  ;
  }
class CLASSI ( extent  CLASSIs)
 {     attribute string attrCI1;
       relationship CLASSH roleIToH  inverse    CLASSH:: roleHToI  ;
}
class CLASSJ ( extent CLASSJs)
 {     attribute date attrCJ1;
       integer  methodCJ1():
}
class CLASSK ( extent CLASSK s)
```

```
{    attribute string attrCK1;
      attribute string attrCK2;
     relationship CLASSB  roleKToB  inverse    CLASSB  :: roleBToK  ;
     relationship CLASSH  roleKToH  inverse    CLASSH  :: roleHToK  ;
}
```

รูปที่ ข.2 แสดงรูปแบบสคีมาฐานข้อมูลเชิงวัตถุของโอดีเอ็มจีโดยการใช้ภาษานิยามเชิงวัตถุ

3. รูปแบบสคีมาฐานข้อมูลเชิงวัตถุของฐานข้อมูลคาเช่

```
class Package.CLASSA Extends %Persistent [ClassType = persistent, ProcedureBlock ]
{   Property attrCA1 As % date;
     Property attrCA2 As %String;
     Method methodCA1() As %String{};
     Method methodCA2 () As %String{};
 }
  class I Package.NTERFACEA Extends %Persistent [ClassType = persistent, ProcedureBlock ]
{    Method  methodIA1() As %String{};
}
class Package.CLASSB extends CLASSA: INTERFACEA   (extent CLASSBs)
{   Property attrCB1 As %String;
     Method  methodCB1() As %String{};
      Relationship  roleBToK As CLASSK [ Cardinality = many, Inverse = roleKToB];
      Relationship  roleBToC  As CLASSC  [ Cardinality = one, Inverse = roleCToB];
}
 class Package.CLASSC Extends %Persistent [ClassType = persistent, ProcedureBlock ]
{    Property classd As % CLASSD  ;
     Method  methodCC1()As %date{};
      Relationship  roleCToD  As CLASSD [ Cardinality = many, Inverse = roleDToC];
      Relationship  roleCToB  As CLASSB  [ Cardinality = one, Inverse = roleBToC];
}
class Package.CLASSD   Extends %Persistent [ClassType = persistent, ProcedureBlock ]
{      Property attrCD1 As %String;
       Property attrCD2 As %String;
       Relationship  roleDToC  As CLASSC  [ Cardinality = one, Inverse = roleCToD];
 }
class Package.CLASSE Extends %Persistent [ClassType = persistent, ProcedureBlock ]
{     Property attrCE1 As %String;
      Property attrCE2 As %String;
      Relationship  roleEToF  As CLASSF  [ Cardinality = one, Inverse = roleFToE  ];
      Relationship  roleEToG  As CLASSG  [ Cardinality = one, Inverse = roleGToE  ];
 }
class Package.CLASSF Extends %Persistent [ClassType = persistent, ProcedureBlock ]
 {     Property attrCF1 As %String;
       Property attrCF2 As %String;
      Relationship  child  As CLASSF [ Cardinality = many, Inverse = parent];
      Relationship  parent  As CLASSF [ Cardinality = one, Inverse = child  ];
      Relationship  roleFToE  As CLASSE [ Cardinality = one, Inverse = roleEToF  ];
      Relationship  roleFToH  As CLASSH [ Cardinality = one, Inverse = roleHToF  ];
  }
class Package.CLASSG Extends %Persistent [ClassType = persistent, ProcedureBlock ]
 {     Property attrCG1 As %String;
       Property pk1 As %String;
       Relationship  roleGToE  As CLASSE  [ Cardinality = one, Inverse = roleEToG  ];
}
class Package.CLASSH Extends %Persistent [ClassType = persistent, ProcedureBlock ]
 {    Property attrCH1 As%String;
      Property classj As %CLASSJ;
      Property classi As %CLASSJ;
      Property attrCI1 As%String;
      Method  methodCH1() As %date{};
      Relationship  roleHToI  As CLASSI [ Cardinality = many, Inverse = roleIToH  ];
      Relationship  roleHToK  As CLASSK [ Cardinality = many, Inverse = roleKToH  ];
      Relationship  roleHToF  As CLASSF  [ Cardinality = one, Inverse = roleFToH  ];
}
class Package.CLASSI Extends %Persistent [ClassType = persistent, ProcedureBlock ]
 {     Property attrCI1 As %String;
```

```
      Relationship  roleIToH  As CLASSH [ Cardinality = one, Inverse = roleHToI  ];
}
class Package.CLASSJ Extends %Persistent [ClassType = persistent, ProcedureBlock ]
{     Property attrCJ1 As % date;
      Method  methodCJ1() As % integer {};
}
class Package.CLASSK Extends %Persistent [ClassType = persistent, ProcedureBlock ]
{    Property attrCK1 As %String;
      Property attrCK2 As %String;
      Relationship  roleKToB  As CLASSB  [ Cardinality = one, Inverse = roleBToK  ];
      Relationship  roleKToH  As CLASSH  [ Cardinality = one, Inverse = roleHToK  ];
}
```

รูปที่ ข.3 แสดงรูปแบบสคีมาฐานข้อมูลเชิงวัตถุของฐานข้อมูลคาเช่

4. รูปแบบสคีมาฐานข้อมูลเชิงวัตถุของฐานข้อมูลแมทิส

```
interface CLASSA: persistent
{   attribute date attrCA1;
    attribute string attrCA2;
    mt_method "CREATE METHOD methodCA1()
          RETURNS STRING    FOR CLASSA
          BEGIN
                DECLARE methodCA1 STRING;
                RETURN  methodCA1;
          END;";
     mt_method "CREATE METHOD methodCA2()
          RETURNS STRING    FOR CLASSA
          BEGIN
                DECLARE methodCA2 STRING;
                RETURN methodCA2;
          END;";
};
 interface: persistent
{     mt_method "CREATE METHOD methodIA1 ()
          RETURNS STRING    FOR INTERFACEA
          BEGIN
                DECLARE methodIA1 STRING;
                RETURN methodIA1;
          END;";
};
interface CLASSB : CLASSA: INTERFACEA   : persistent
{   attribute string attrCB1;
    mt_method "CREATE METHOD methodCB1 ()
          RETURNS STRING    FOR CLASSB
          BEGIN
                DECLARE  methodCB1 STRING;
                RETURN  methodCB1;
          END;";
    relationship  set<CLASSK>  roleBToK  inverse    CLASSK:: roleKToB;
    relationship  CLASSC  roleBToC  inverse    CLASSC:: roleCToB ;
};
 interface CLASSC : persistent
{   attribute CLASSD  classd  ;
    mt_method "CREATE METHOD methodCC1 ()
          RETURNS DATE    FOR CLASSC
          BEGIN
                DECLARE  methodCC1 DATE;
                RETURN  methodCC1;
          END;";
    relationship  set<CLASSD>  roleCToD[1,-1]   inverse    CLASSD:: roleDToC;
    relationship  CLASSB  roleCToB[1,1]  inverse    CLASSB:: roleBToC ;
};
interface CLASSD  : persistent
{     attribute string attrCD1;
      attribute string attrCD2;
      relationship  CLASSC  roleDToC[1,1]    inverse    CLASSC:: roleCToD ;
```

```
};
interface CLASSE: persistent
{    attribute string attrCE1;
     attribute string attrCE2;
     relationship CLASSF  roleEToF[1,1]   inverse   CLASSF  :: roleFToE  ;
     relationship CLASSG  roleEToG[1,1]   inverse   CLASSG  :: roleGToE  ;
};
interface CLASSF : persistent
{    attribute string attrCF1;
     attribute string attrCF2;
     relationship set<CLASSF> child[1,-1]   inverse   CLASSF:: parent;
     relationship CLASSF parent [1,1]   inverse   CLASSF:: child  ;
     relationship CLASSE roleFToE[1,1]   inverse   CLASSE:: roleEToF  ;
     relationship CLASSH roleFToH[1,1]   inverse   CLASSH:: roleHToF  ;
};
interface CLASSG : persistent
{    attribute string attrCG1;
     attribute string pk1;
     relationship CLASSE  roleGToE  inverse   CLASSE  :: roleEToG  ;
};
interface CLASSH : persistent
{    attribute string attrCH1;
     attribute CLASSJ classj;
     attribute CLASSI classi;
     attribute string attrCI1;
     mt_method "CREATE METHOD methodCH1 ()
          RETURNS DATE   FOR CLASSH
          BEGIN
               DECLARE  methodCH1DATE;
               RETURN  methodCH1;
          END;";
     relationship set<CLASSI>  roleHToI[1,-1]   inverse   CLASSI:: roleIToH  ;
     relationship set<CLASSK>  roleHToK[1,-1]   inverse   CLASSK:: roleKToH  ;
     relationship CLASSF  roleHToF[1,1]   inverse   CLASSF  :: roleFToH  ;
 };
interface CLASSI : persistent
{    attribute string attrCI1;
     relationship CLASSH roleIToH[1,1]   inverse   CLASSH:: roleHToI  ;
};
interface CLASSJ : persistent
{    attribute date attrCJ1;
     mt_method "CREATE METHOD methodCJ1 ()
          RETURNS integer  FOR CLASSJ
          BEGIN
               DECLARE  methodCJ1 integer  ;
               RETURN  methodCJ1;
          END;";
};
interface CLASSK : persistent
{    attribute string attrCK1;
     attribute string attrCK2;
     relationship CLASSB  roleKToB[1,1]   inverse   CLASSB  :: roleBToK  ;
     relationship CLASSH  roleKToH[1,1]   inverse   CLASSH  :: roleHToK  ;
};
```

รูปที่ ข.4 แสดงรูปแบบสคีมาฐานข้อมูลเชิงวัตถุของฐานข้อมูลแมทิส

ภาคผนวก ค

สรุปรูปแบบการเก็บข้อมูลในเอ็กซ์เอ็มแอล

```
<oodb_schema>
      <class [interface] ComponentID=" "  CPNname=" "  Cvisibility=" "  Cxcenter =" " Cycenter =" "
CleftPos=" " CtopPos=" "  Cextent="">
                  <attribute CPNname=" "   Visibility="  "  Type="  "  Initialvalue ="  " ></attribute>
                  <method CPNname=" "   Visibility="  "   RetType="  "></method>
       </class[/interface]>
      <relationship>
          <association ComponentID=" " SourcePos="  " SourceID="  " SourceType="  " DestID="  "
DestType=" " X1=" " Y1=" " Xstart=" " Ystart=" " X2=" " Y2=" " RolA="  " RolB="  " MultiA="  " MultiB="  "
XrolA="  " XrolB="  " XmultiA="  " XmultiB="  "  YrolA="  " YrolB="  " YmultiA="  " ymultiB="  " >
          </association>
          <associationclass ComponentID=" " SourcePos="  " SourceID="  " SourceType="  " DestID="  "
DestType=" " X1=" " Y1=" " Xstart=" " Ystart=" " X2=" " Y2=" " RolA="  " RolB="  " MultiA="  " MultiB="  "
XrolA="  " XrolB="  " XmultiA="  " XmultiB="  "  YrolA="  " YrolB="  " YmultiA="  " ymultiB="  " >
          </associationclass >
          <recursiveassociation  ComponentID=" " SourcePos="  " SourceID="  " SourceType="  "
DestID="  " DestType=" " X1=" " Y1=" " Xstart=" " Ystart=" " X2=" " Y2=" " RolA="  " RolB="  " MultiA="  "
MultiB="  " XrolA="  " XrolB="  " XmultiA="  " XmultiB="  "  YrolA="  " YrolB="  " YmultiA="  " ymultiB="  " >
          </recursiveassociation >
          <dependency  ComponentID=" " SourcePos="  " SourceID="  " SourceType="  " DestID="  "
DestType=" " X1=" " Y1=" " Xstart=" " Ystart=" " X2=" " Y2=" " ></dependency >
          <aggregation  ComponentID=" " SourcePos="  " SourceID="  " SourceType="  " DestID="  "
DestType=" " X1=" " Y1=" " Xstart=" " Ystart=" " X2=" " Y2=" " RolA="  " RolB="  " MultiA="  " MultiB="  "
XrolA="  " XrolB="  " XmultiA="  " XmultiB="  "  YrolA="  " YrolB="  " YmultiA="  " ymultiB="  " >
          </aggregation>
          <composition ComponentID=" " SourcePos="  " SourceID="  " SourceType="  " DestID="  "
DestType=" " X1=" " Y1=" " Xstart=" " Ystart=" " X2=" " Y2=" " RolA="  " RolB="  " MultiA="  " MultiB="  "
XrolA="  " XrolB="  " XmultiA="  " XmultiB="  "  YrolA="  " YrolB="  " YmultiA="  " ymultiB="  " >
          </composition>
          <qualifiedassociation  ComponentID=" " SourcePos="  " SourceID="  " SourceType="  "
DestID="  " DestType=" " X1=" " Y1=" " Xstart=" " Ystart=" " X2=" " Y2=" " RolA="  " RolB="  " MultiA="  "
MultiB="  " XrolA="  " XrolB="  " XmultiA="  " XmultiB="  "  YrolA="  " YrolB="  " YmultiA="  " ymultiB="  " >
          </qualifiedassociation >
          <generalization ComponentID=" " SourcePos=" " SourceID =" " SourceType="  " DestId =" "
DestType=" " X1=" " Y1=" " Xstart=" " Ystart=" " X2="  " Y2="  "  Xstop=" " Ystop=" " > </generalization>
          <realization ComponentID=" " SourcePos=" " SourceID =" " SourceType="  " DestId =" "
DestType=" " X1=" " Y1=" " Xstart=" " Ystart=" " X2="  " Y2="  "  Xstop=" " Ystop=" " > </ realization >
       </relationship>
</oodb_schema>
```

รูปที่ ค.1 สรุปรูปแบบการเก็บข้อมูลในเอ็กซ์เอ็มแอล

สรุปรูปแบบการเก็บข้อมูลของคลาสไดอะแกรมในเอ็กซ์เอ็มแอล

```
<oodb_schema>
   <class>
          <attribute></attribute>
          <method></method>
    </class >
   <relationship></relationship>
</oodb_schema >
```

รูปที่ ค.2 แสดงสรุปรูปแบบการเก็บข้อมูลของคลาสไดอะแกรมในเอ็กซ์เอ็มแอล

ภาคผนวก ง

ภาพรวมของไวยากรณ์ในการสร้างสคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุ

```
<RULESCHEMA-OODB>   ::= {<INTERFACEDEF_NAME>}{<CLASSDEF_NAME>}{<RELATIONSHIP_NAME>}

<INTERFACEDEF_NAME > ::= interface  INTERFACENAME_NAME
                         "{"
                              {<METHOD_NAME>}
                         "}"

<METHOD_NAME> ::=<TYPE>  METHODNAME_NAME  ([<ARGUMENT-LIST>])

<ARGUMENT-LIST> ::= <TYPE> ARGUMENTNAME_NAME[{,<TYPE> ARGUMENTNAME_NAME}]

<TYPE>::= <PRIMITIVE>|<NON-PRIMITIVE>

<PRIMITIVE>::= String | Date | Integer | Char | Short

<NON-PRIMITIVE>::= <STRUCT>|<COLLECTION>|<CLASS_NAME>

<STRUCT>  ::= STRUCTNAME

<COLLECTION> ::= <COLLECTION-TYPE>"<" <PRIMITIVE> ">" |
                 <COLLECTION-TYPE>"<" <NON-PRIMITIVE> ">"

<COLLECTION-TYPE> ::= set | bag| list

<CLASS_NAME >::=CLASSNAME_NAME

<CLASSDEF_NAME> ::=<CLASSHEADER_NAME>
                    "{"
                       {<CLASSBODY_NAME>}
                    "}"
<CLASSHEADER_NAME> ::= class CLASSNAME_NAME
                     (extent  EXTENTOFCLASS_NAME  [key  <KEY-LIST_NAME>])

<KEY-LIST_NAME> ::= <TYPE> KEYOFCLASS_NAME [{,<TYPE> KEYOFCLASS_NAME}]

<CLASSBODY_NAME>::={<ATTRIBUTE_NAME>}
                   {<METHOD_NAME>}

<ATTRIBUTE_NAME> ::= attribute  <TYPE>  ATTRIBUTENAME_NAME;

<RELATIONSHIP_NAME>::= {<RELATIONSHIP-H_NAME>}{<RELATIONSHIP-D_NAME>}

<RELATIONSHIP-H_NAME>::=[<GENERALIZATION>][<REALIZATION>]

<GENERALIZATION>::=CLASSNAME_CLASS1-2:
                   extends  SUPERCLASSNAME_NAME

<REALIZATION>::=CLASSNAME_CLASS1-2:
                ":"INTERFACENAME_INAME[{":"INTERFACENAME_INAME}]

<RELATIONSHIP-D_NAME>::=<ASSOCIATION> | <ASSOCIATIONCLASS> |
                        <RECURSIVEASSOCIATION> | <DEPENDENCY> |
                        <AGGREGATION> | <COMPOSITION> |
                        <QUALIFIEDASSOCIATION>
<RELATIONSHIP-LIST_NAME> ::= relationship CLASSNAME_NAME|
                             relationship <COLLECTION-TYPE><CLASSNAME_NAME>
```

```
<ASSOCIATION> ::= <ASSOCIATION-CLASS1><ASSOCIATION-CLASS2>

<ASSOCIATION-CLASS1>::=CLASSNAME_CLASS1-2:
                       <RELATIONSHIP-LIST_CLASS2-1>   RELATIONSHIPNAME_CLASS1-2
                      inverse  CLASSNAME_CLASS2-1:: RELATIONSHIPNAME_CLASS2-1 ;

<ASSOCIATION-CLASS2 >::=CLASSNAME_CLASS2-1:
                       <RELATIONSHIP-LIST_CLASS1-2>  RELATIONSHIPNAME_CLASS2-1
                      inverse  CLASSNAME_CLASS1-2:: RELATIONSHIPNAME_CLASS1-2;

<ASSOCIATIONCLASS>  ::= <ASSOCIATIONCLASS-CLASS1>
                        <ASSOCIATIONCLASS-CLASS2>
                        <ASSOCIATIONCLASS-CLASS3>

<ASSOCIATIONCLASS-CLASS1>::=CLASSNAME_CLASS1-3:
                       <RELATIONSHIP-LIST_CLASS3-1> RELATIONSHIPNAME_CLASS1-3
                       inverse  CLASSNAME_CLASS3-1:: RELATIONSHIPNAME_CLASS3-1;

<ASSOCIATIONCLASS-CLASS2>::=CLASSNAME_CLASS2-3:
                       <RELATIONSHIP-LIST_CLASS3-2> RELATIONSHIPNAME_CLASS2-3
                       inverse  CLASSNAME_CLASS3-2:: RELATIONSHIPNAME_CLASS3-2;

<ASSOCIATIONCLASS-CLASS3>::=CLASSNAME_CLASS3-1:
                       <RELATIONSHIP-LIST_CLASS1-3> RELATIONSHIPNAME_CLASS3-1
                       inverse  CLASSNAME_CLASS1-3:: RELATIONSHIPNAME_CLASS1-3;

                       <RELATIONSHIP-LIST_CLASS2-3> RELATIONSHIPNAME_CLASS3-2
                       inverse  CLASSNAME_CLASS2-3:: RELATIONSHIPNAME_CLASS2-3;

<RECURSIVEASSOCIATION> ::= CLASSNAME_CLASS1:
                       <RELATIONSHIP-LIST_CLASS1> RELATIONSHIPNAME_CLASS1(PARENT)
                       inverse  CLASSNAME_CLASS1:: RELATIONSHIPNAME_CLASS1(CHILD) ;

                       <RELATIONSHIP-LIST_CLASS1> RELATIONSHIPNAME_CLASS1(CHILD)
                       inverse  CLASSNAME_CLASS1:: RELATIONSHIPNAME_CLASS1(PARENT) ;

<DEPENDENCY>  ::= CLASSNAME_CLASS1-2:
                     attribute  <CLASS_CLASS2-1>  ATTRIBUTENAME_CLASS1-2;

<AGGREGATION> ::= <AGGREGATION-CLASS1><AGGREGATION-CLASS2>

<AGGREGATION-CLASS1> ::= CLASSNAME_CLASS1-2:
                       <RELATIONSHIP-LIST_CLASS2-1>  RELATIONSHIPNAME_CLASS1-2
                       inverse  CLASSNAME_CLASS2-1:: RELATIONSHIPNAME_CLASS2-1;
                       attribute  <CLASS_CLASS2-1>  ATTRIBUTENAME_CLASS1-2;

<AGGREGATION-CLASS2> ::= CLASSNAME_CLASS2-1:
                       <RELATIONSHIP-LIST_CLASS1-2>  RELATIONSHIPNAME_CLASS2-1
                       inverse  CLASSNAME_CLASS1-2:: RELATIONSHIPNAME_CLASS1-2;

<COMPOSITION> ::= <COMPOSITION-CLASS1> <COMPOSITION-CLASS2>

<COMPOSITION-CLASS1>::=CLASSNAME_CLASS1-2:
                       <RELATIONSHIP-LIST_CLASS2-1>   RELATIONSHIPNAME_CLASS1-2
                       inverse  CLASSNAME_CLASS2-1:: RELATIONSHIPNAME_CLASS2-1;

                       attribute <CLASS_CLASS2-1>  ATTRIBUTENAME _CLASS1-2;

                       attribute  <KEY-LIST_NAME>;

<COMPOSITION –CLASS2>::=CLASSNAME_CLASS1-2:
                       <RELATIONSHIP-LIST_CLASS1-2>   RELATIONSHIPNAME_CLASS2-1
                       inverse  CLASSNAME_CLASS1-2:: RELATIONSHIPNAME_CLASS1-2;

<QUALIFIEDASSOCIATION> ::= <QUALIFIEDASSOCIATION-CLASS1>
```

```
                              <QUALIFIEDASSOCIATION-CLASS2>

<QUALIFIEDASSOCIATION-CLASS1>::=CLASSNAME_CLASS1-2:
                        <RELATIONSHIP-LIST_CLASS2-1> RELATIONSHIPNAME_CLASS1-2
                        inverse  CLASSNAME_CLASS2-1:: RELATIONSHIPNAME_CLASS2-1;

<QUALIFIEDASSOCIATION-CLASS2>::=CLASSNAME_CLASS2-1:
                        <RELATIONSHIP-LIST_CLASS1-2>   RELATIONSHIPNAME_CLASS2-1
                        inverse  CLASSNAME_CLASS1-2:: RELATIONSHIPNAME_CLASS1-2;

                        attribute <TYPE>  PARTIALKEY_CLASS1-2;

                        {<ATTRIBUTE_CLASS1-2>}
```

รูปที่ ง.1 สรุปไวยากรณ์การแปลงคลาสไดอะแกรมเป็นสคีมาตัวกลางของสคีมาฐานข้อมูลเชิงวัตถุตามมาตรฐานโอดีเอ็มจี

```
<RULESCHEMA-OODB>   ::= {<INTERFACEDEF_NAME>}{<CLASSDEF_NAME>}{<RELATIONSHIP_NAME>}

<INTERFACEDEF_NAME > ::= interface  INTERFACENAME_NAME
                         "{"
                              {<METHOD_NAME>}
                         "}"

<METHOD_NAME> ::=<TYPE>  METHODNAME_NAME  ([<ARGUMENT-LIST>])

<ARGUMENT-LIST> ::=ARGUMENTNAME_NAME as %<TYPE>[{,ARGUMENTNAME_NAME as %<TYPE> }]

<TYPE>::= <PRIMITIVE>|<NON-PRIMITIVE>

<PRIMITIVE>::= String | Date | Integer | Char | Short

<NON-PRIMITIVE>::= <STRUCT>|<COLLECTION>|<CLASS_NAME>

<STRUCT>  ::= STRUCTNAME

<COLLECTION> ::= <COLLECTION-TYPE>"<" <PRIMITIVE> ">" |
                 <COLLECTION-TYPE>"<" <NON-PRIMITIVE> ">"

<COLLECTION-TYPE> ::= set | bag| list

<CLASS_NAME >::=CLASSNAME_NAME

<CLASSDEF_NAME> ::=<CLASSHEADER_NAME>
                         "{"
                            {<CLASSBODY_NAME>}
                         "}"
<CLASSHEADER_NAME> ::= class  Package.CLASSNAME_NAME

<KEY-LIST_NAME> ::= KEYOFCLASS_NAME as %<TYPE> [{,KEYOFCLASS_NAME as %<TYPE>}]

<CLASSBODY_NAME>::={<ATTRIBUTE_NAME>}
                   {<METHOD_NAME>}

<ATTRIBUTE_NAME> ::= property   ATTRIBUTENAME_NAME as %<TYPE> ;

<RELATIONSHIP_NAME>::= {<RELATIONSHIP-H_NAME>}{<RELATIONSHIP-D_NAME>}

<RELATIONSHIP-H_NAME>::=[<GENERALIZATION>][<REALIZATION>]

<GENERALIZATION>::=CLASSNAME_CLASS1-2:
        extends %Persistent[ClassType = persistent,ProcedureBlock] |
        extends %SUPERCLASSNAME_CLASSNAME%Persistent
```

```
            [ClassType = persistent,ProcedureBlock]

<REALIZATION>::=CLASSNAME_CLASS1-2:
                  ":"INTERFACENAME_INAME[{":"INTERFACENAME_INAME}]

<RELATIONSHIP-D_NAME>::=<ASSOCIATION> | <ASSOCIATIONCLASS> |
                        <RECURSIVEASSOCIATION> | <DEPENDENCY> |
                        <AGGREGATION> | <COMPOSITION> |
                        <QUALIFIEDASSOCIATION>
<RELATIONSHIP-LIST_NAME> ::= relationship CLASSNAME_NAME|
                             relationship <COLLECTION-TYPE><CLASSNAME_NAME>

<ASSOCIATION> ::= <ASSOCIATION-CLASS1><ASSOCIATION-CLASS2>

<ASSOCIATION-CLASS1>::=CLASSNAME_CLASS1-2:
       <RELATIONSHIP-LIST_CLASS2-1> RELATIONSHIPNAME_CLASS1-2 as CLASSNAME_CLASS2-1
       [Cardinality= MULTIPLICITY_CLASS1-2 ,inverse RELATIONSHIPNAME_CLASS2-1] ;

<ASSOCIATION-CLASS2 >::= CLASSNAME_CLASS2-1:
       <RELATIONSHIP-LIST_CLASS1-2> RELATIONSHIPNAME_CLASS2-1 as CLASSNAME_CLASS1-2
       [Cardinality= MULTIPLICITY_CLASS2-1 ,inverse RELATIONSHIPNAME_CLASS2-1] ;

<ASSOCIATIONCLASS>  ::= <ASSOCIATIONCLASS-CLASS1>
                        <ASSOCIATIONCLASS-CLASS2>
                        <ASSOCIATIONCLASS-CLASS3>

<ASSOCIATIONCLASS-CLASS1>::=CLASSNAME_CLASS1-3:
       <RELATIONSHIP-LIST_CLASS3-1>  RELATIONSHIPNAME_CLASS1-3 as CLASSNAME_CLASS3-1
       [Cardinality= MULTIPLICITY_CLASS1-3 ,inverse RELATIONSHIPNAME_CLASS3-1] ;

<ASSOCIATIONCLASS-CLASS2>::=CLASSNAME_CLASS2-3:
       <RELATIONSHIP-LIST_CLASS3-2>  RELATIONSHIPNAME_CLASS2-3 as CLASSNAME_CLASS3-2
       [Cardinality= MULTIPLICITY_CLASS2-3 ,inverse RELATIONSHIPNAME_CLASS3-2] ;

<ASSOCIATIONCLASS-CLASS3>::=CLASSNAME_CLASS3-1:
       <RELATIONSHIP-LIST_CLASS1-3>  RELATIONSHIPNAME_CLASS3-1 as CLASSNAME_CLASS1-3
       [Cardinality= MULTIPLICITY_CLASS3-1 ,inverse RELATIONSHIPNAME_CLASS1-3];

       <RELATIONSHIP-LIST _CLASS2-3>  RELATIONSHIPNAME_CLASS3-2 as CLASSNAME_CLASS2-3
       [Cardinality= MULTIPLICITY_CLASS3-2 ,inverse RELATIONSHIPNAME_CLASS2-3];

<RECURSIVEASSOCIATION> ::= CLASSNAME_CLASS1:
       <RELATIONSHIP-LIST_CLASS1> RELATIONSHIPNAME_CLASS1(PARENT) as CLASSNAME_CLASS1
       [Cardinality= MULTIPLICITY_CLASS1,inverse RELATIONSHIPNAME_CLASS1(CHILD)] ;

       <RELATIONSHIP-LIST_CLASS1> RELATIONSHIPNAME_CLASS1(CHILD) as CLASSNAME_CLASS1
       [Cardinality= MULTIPLICITY_CLASS1 ,inverse RELATIONSHIPNAME_CLASS1(PARENT)];

<DEPENDENCY>  ::= CLASSNAME_CLASS1-2:
        property  ATTRIBUTENAME_CLASS1-2  as % <CLASS_CLASS2-1> ;

<AGGREGATION> ::= <AGGREGATION-CLASS1><AGGREGATION-CLASS2>

<AGGREGATION-CLASS1> ::= CLASSNAME_CLASS1-2:
       <RELATIONSHIP-LIST_CLASS2-1> RELATIONSHIPNAME_CLASS1-2 as CLASSNAME_CLASS2-1
       [Cardinality= MULTIPLICITY_CLASS1-2,inverse RELATIONSHIPNAME_CLASS2-1] ;

       property  ATTRIBUTENAME_CLASS1-2  as % <CLASS_CLASS2-1>  ;

<AGGREGATION-CLASS2> ::= CLASSNAME_CLASS2-1:
       <RELATIONSHIP-LIST_CLASS1-2> RELATIONSHIPNAME_CLASS2-1 as CLASSNAME_CLASS1-2
       [Cardinality= MULTIPLICITY_CLASS2-1,inverse RELATIONSHIPNAME_CLASS1-2] ;


<COMPOSITION> ::= <COMPOSITION-CLASS1> <COMPOSITION-CLASS2>
```

```
<COMPOSITION-CLASS1>::=CLASSNAME_CLASS1-2:
      <RELATIONSHIP-LIST_CLASS2-1> RELATIONSHIPNAME_CLASS1-2 as CLASSNAME_CLASS2-1
      [Cardinality= MULTIPLICITY_CLASS1-2,inverse RELATIONSHIPNAME_CLASS2-1];

      property  ATTRIBUTENAME_CLASS1-2  as % <CLASS_CLASS2-1>;

      property  <KEY-LIST_NAME>;

<COMPOSITION –CLASS2>::=CLASSNAME_CLASS2-1:
      <RELATIONSHIP-LIST_CLASS1-2>  RELATIONSHIPNAME_CLASS2-1 as  CLASSNAME_CLASS1-2
      [Cardinality= MULTIPLICITY_CLASS2-1,inverse RELATIONSHIPNAME_CLASS1-2] ;

<QUALIFIEDASSOCIATION> ::= <QUALIFIEDASSOCIATION-CLASS1>
                           <QUALIFIEDASSOCIATION-CLASS2>

<QUALIFIEDASSOCIATION-CLASS1>::=CLASSNAME_CLASS1-2:
      <RELATIONSHIP-LIST CLASS2-1> RELATIONSHIPNAME_CLASS1-2 as  CLASSNAME_CLASS2-1
      [Cardinality= MULTIPLICITY_CLASS1-2,inverse RELATIONSHIPNAME_CLASS2-1] ;

<QUALIFIEDASSOCIATION-CLASS2>::=CLASSNAME_CLASS2-1:

      <RELATIONSHIP-LIST_CLASS1-2> RELATIONSHIPNAME_CLASS2-1 as  CLASSNAME_CLASS1-2
      [Cardinality= MULTIPLICITY_CLASS2-1,inverse RELATIONSHIPNAME_CLASS1-2] ;

      property  PARTIALKEY_CLASS1-2  as % <TYPE> ;

      {<ATTRIBUTE_CLASS1-2>}
```

รูปที่ ง .2 สรุปไวยากรณ์การแปลงคลาสไดอะแกรมเป็นสคีมาตัวกลางของสคีมาฐานข้อมูลเชิงวัตถุของฐานข้อมูลคาเซ่

```
<RULESCHEMA-OODB>    ::= {<INTERFACEDEF_NAME>}{<CLASSDEF_NAME>}{<RELATIONSHIP_NAME>}

<INTERFACEDEF_NAME > ::= interface  INTERFACENAME_NAME
                         "{"
                              {<METHOD_NAME>}
                         "}"

<METHOD_NAME> ::= mt_method"CREATE METHOD  METHODNAME_NAME
                  ([<ARGUMENT-LIST>])
                   RETURNS <TYPE> FOR  CLASSNAME_NAME
                   BEGIN
                     DECLARE  METHODNAME_NAME  <TYPE>;
                     RETURN   METHODNAME_NAME;
                   END;";

<ARGUMENT-LIST> ::= <TYPE> ARGUMENTNAME_NAME[{,<TYPE> ARGUMENTNAME_NAME}]

<TYPE>::= <PRIMITIVE>|<NON-PRIMITIVE>

<PRIMITIVE>::= String | Date | Integer | Char | Short

<NON-PRIMITIVE>::= <STRUCT>|<COLLECTION>|<CLASS_NAME>

<STRUCT>  ::= STRUCTNAME

<COLLECTION> ::= <COLLECTION-TYPE>"<" <PRIMITIVE> ">" |
                 <COLLECTION-TYPE>"<" <NON-PRIMITIVE> ">"

<COLLECTION-TYPE> ::= set | bag| list
```

```
<CLASS_NAME >::=CLASSNAME_NAME

<CLASSDEF_NAME> ::=<CLASSHEADER_NAME>
                        "{"
                           {<CLASSBODY_NAME>}
                        "}"
<CLASSHEADER_NAME> ::= interface  CLASSNAME_NAME

<KEY-LIST_NAME> ::= <TYPE> KEYOFCLASS_NAME [{,<TYPE> KEYOFCLASS_NAME}]

<CLASSBODY_NAME>::={<ATTRIBUTE_NAME>}
                   {<METHOD_NAME>}

<ATTRIBUTE_NAME> ::= attribute  <TYPE>  ATTRIBUTENAME_NAME;

<RELATIONSHIP_NAME>::= {<RELATIONSHIP-H_NAME>}{<RELATIONSHIP-D_NAME>}

<RELATIONSHIP-H_NAME>::=[<GENERALIZATION>][<REALIZATION>]

<GENERALIZATION>::=CLASSNAME_CLASS1-2:
                    ":"SUPERCLASSNAME_NAME

<REALIZATION>::=CLASSNAME_CLASS1-2:
                  ":"INTERFACENAME_NAME[{":"INTERFACENAME_NAME}]

<RELATIONSHIP-D_NAME>::=<ASSOCIATION> | <ASSOCIATIONCLASS> |
                        <RECURSIVEASSOCIATION> | <DEPENDENCY> |
                        <AGGREGATION> | <COMPOSITION> |
                        <QUALIFIEDASSOCIATION>
<RELATIONSHIP-LIST_NAME> ::= relationship CLASSNAME_NAME|
                             relationship <COLLECTION-TYPE><CLASSNAME_NAME>

<ASSOCIATION> ::= <ASSOCIATION-CLASS1><ASSOCIATION-CLASS2>

<ASSOCIATION-CLASS1>::=CLASSNAME_CLASS1-2:
    <RELATIONSHIP-LIST_CLASS2-1> RELATIONSHIPNAME_CLASS1-2[MULTIPLICITY_CLASS1-2]
    inverse  CLASSNAME_CLASS2-1:: RELATIONSHIPNAME_CLASS2-1 ;

<ASSOCIATION-CLASS2 >::=CLASSNAME_CLASS2-1:
    <RELATIONSHIP-LIST_CLASS1-2> RELATIONSHIPNAME_CLASS2-1[MULTIPLICITY_CLASS2-1]
    inverse  CLASSNAME_CLASS1-2:: RELATIONSHIPNAME_CLASS1-2;

<ASSOCIATIONCLASS>  ::= <ASSOCIATIONCLASS-CLASS1>
                        <ASSOCIATIONCLASS-CLASS2>
                        <ASSOCIATIONCLASS-CLASS3>

<ASSOCIATIONCLASS-CLASS1>::=CLASSNAME_CLASS1-3:
      <RELATIONSHIP-LIST_CLASS3-1> RELATIONSHIPNAME_CLASS1-3[MULTIPLICITY_CLASS1-3]
      inverse  CLASSNAME_CLASS3-1:: RELATIONSHIPNAME_CLASS3-1;

<ASSOCIATIONCLASS-CLASS2>::=CLASSNAME_CLASS2-3:
     <RELATIONSHIP-LIST_CLASS3-2> RELATIONSHIPNAME_CLASS2-3[MULTIPLICITY_CLASS2-3]
      inverse  CLASSNAME_CLASS3-2:: RELATIONSHIPNAME_CLASS3-2;

<ASSOCIATIONCLASS-CLASS3>::=CLASSNAME_CLASS3-1:
        <RELATIONSHIP-LIST_CLASS1-3> RELATIONSHIPNAME_CLASS3-1[MULTIPLICITY_CLASS3-1]
         inverse  CLASSNAME_CLASS1-3:: RELATIONSHIPNAME_CLASS1-3;

        <RELATIONSHIP-LIST_CLASS2-3> RELATIONSHIPNAME_CLASS3-2[MULTIPLICITY_CLASS3-2]
         inverse  CLASSNAME_CLASS2-3:: RELATIONSHIPNAME_CLASS2-3;

<RECURSIVEASSOCIATION> ::= CLASSNAME_CLASS1:
   <RELATIONSHIP-LIST_CLASS1> RELATIONSHIPNAME_CLASS1(PARENT)[MULTIPLICITY_CLASS1(PARENT)]
         inverse  CLASSNAME_CLASS1:: RELATIONSHIPNAME_CLASS1(CHILD) ;


   <RELATIONSHIP-LIST_CLASS1> RELATIONSHIPNAME_CLASS1(CHILD)[MULTIPLICITY_CLASS1(CHILD)]
```

```
          inverse  CLASSNAME_CLASS1:: RELATIONSHIPNAME_CLASS1(PARENT) ;

<DEPENDENCY>  ::= CLASSNAME_CLASS1-2:
                    attribute  <CLASS_CLASS2-1>  ATTRIBUTENAME_CLASS1-2;

<AGGREGATION> ::= <AGGREGATION-CLASS1><AGGREGATION-CLASS2>

<AGGREGATION-CLASS1> ::= CLASSNAME_CLASS1-2:
      <RELATIONSHIP-LIST_CLASS2-1>  RELATIONSHIPNAME_CLASS1-2[MULTIPLICITY_CLASS1-2]
       inverse  CLASSNAME_CLASS2-1:: RELATIONSHIPNAME_CLASS2-1;

       attribute  <CLASS_CLASS2-1>  ATTRIBUTENAME _CLASS1-2;

<AGGREGATION-CLASS2> ::=CLASSNAME_CLASS2-1:
      <RELATIONSHIP-LIST_CLASS1-2>  RELATIONSHIPNAME_CLASS2-1[MULTIPLICITY_CLASS2-1]
       inverse  CLASSNAME_CLASS1-2:: RELATIONSHIPNAME_CLASS1-2;

<COMPOSITION> ::= <COMPOSITION-CLASS1> <COMPOSITION-CLASS2>

<COMPOSITION-CLASS1>::=CLASSNAME_CLASS1-2:
      <RELATIONSHIP-LIST_CLASS2-1>  RELATIONSHIPNAME_CLASS1-2[MULTIPLICITY_CLASS1-2]
       inverse  CLASSNAME_CLASS2-1:: RELATIONSHIPNAME_CLASS2-1;

       attribute  <TYPE>  KEYOFCLASS_CLASS2-1;

       attribute <CLASS_CLASS2-1>  ATTRIBUTENAME _CLASS1-2;

<COMPOSITION -CLASS2>::=CLASSNAME_CLASS2-1:
      <RELATIONSHIP-LIST_CLASS1-2>  RELATIONSHIPNAME_CLASS2-1[MULTIPLICITY_CLASS2-1]
       inverse  CLASSNAME_CLASS1-2:: RELATIONSHIPNAME_CLASS1-2;

<QUALIFIEDASSOCIATION> ::= <QUALIFIEDASSOCIATION-CLASS1>
                           <QUALIFIEDASSOCIATION-CLASS2>

<QUALIFIEDASSOCIATION-CLASS1>::=CLASSNAME_CLASS1-2:
       <RELATIONSHIP-LIST_CLASS2-1> RELATIONSHIPNAME_CLASS1-2[MULTIPLICITY_CLASS1-2]
       inverse  CLASSNAME_CLASS2-1:: RELATIONSHIPNAME_CLASS2-1;

<QUALIFIEDASSOCIATION-CLASS2>::=CLASSNAME_CLASS2-1:
      <RELATIONSHIP-LIST_CLASS1-2>  RELATIONSHIPNAME_CLASS2-1[MULTIPLICITY_CLASS2-1]
       inverse  CLASSNAME_CLASS1-2:: RELATIONSHIPNAME_CLASS1-2;

       attribute <TYPE>  PARTIALKEY_CLASS1-2;

       {<ATTRIBUTE_CLASS1-2>}
```

รูปที่ ง.3 สรุปไวยากรณ์การแปลงคลาสไดอะแกรมเป็นสคีมาตัวกลางของสคีมาฐานข้อมูลเชิงวัตถุของฐานข้อมูลแมทิส

ภาคผนวก จ

การใช้เครื่องมือที่ประยุกต์การใช้กฎการแปลงยูเอ็มแอลคลาสไดอะแกรมเป็นสคีมาฐานข้อมูลเชิงวัตถุ

เนื้อหาในบทนี้จะอธิบายถึงวิธีการใช้งานเครื่องมือที่ประยุกต์การใช้กฎการแปลงยูเอ็มแอลคลาสไดอะแกรมเป็นฐานข้อมูลเชิงวัตถุ โดยจะอธิบายถึงการเรียกใช้เครื่องมือและส่วนประกอบต่าง ๆ ที่ปรากฏในเครื่องมือการแปลงยูเอ็มแอลคลาสไดอะแกรมเป็นฐานข้อมูลเชิงวัตถุ รวมถึงวิธีการใช้งานเครื่องมือและขั้นตอนการสร้างสคีมาฐานข้อมูลเชิงวัตถุ

จ.1 การเรียกใช้เครื่องมือและส่วนประกอบต่าง ๆที่ปรากฏในเครื่องมือ

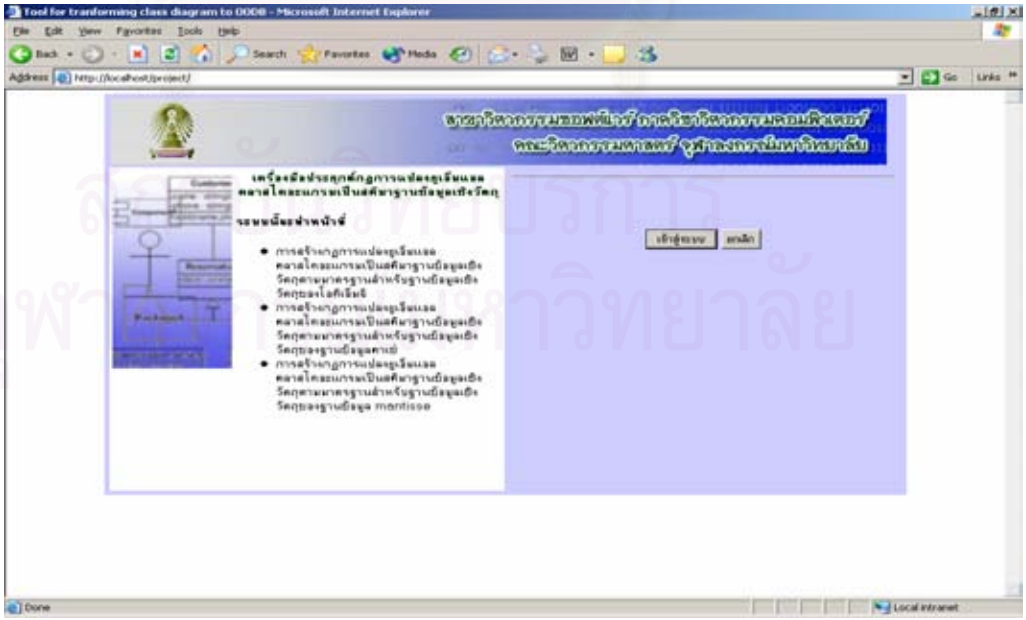
จ.1.1 การเรียกใช้เครื่องมือ

1.เนื่องจากเครื่องมือที่ประยุกต์การใช้กฎการแปลงยูเอ็มแอลคลาสไดอะแกรมเป็นฐานข้อมูลเชิงวัตถุเป็นเว็บแอพลิเคชัน ดังนั้นการเรียกใช้เครื่องมือต้องติดต่อผ่านระบบเครือข่ายโดยพิมพ์ ที่อยู่ (URL Address) เพื่อติดต่อกับเครื่องมือ

2. ผู้ใช้งานสามารถติดตั้งโปรแกรมที่เครื่องคอมพิวเตอร์ของผู้ใช้งานเอง และติดต่อกับเครื่องมือโดยไม่ต้องผ่านเครือข่ายได้โดยพิมพ์ที่อยู่เพื่อติดต่อกับเครื่องมือที่

http://localhost/project/

3. เครื่องมือจะแสดงหน้าจอสำหรับการทำงานหน้าจอแรกขึ้นมาดังรูปที่ จ.1



รูปที่ จ.1 แสดงหน้าจอสำหรับการทำงานหน้าจอแรกของเครื่องมือ

จ.1.2 ส่วนประกอบต่าง ๆ ในเครื่องมือ

1. แถบกลุ่มสัญลักษณ์ตามมาตรฐานของ ยูเอ็มแอลที่ใช้สำหรับการวาดคลาสไดอะแกรมมีดังนี้

1.  แสดงสัญลักษณ์ของคลาส
2.  แสดงสัญลักษณ์ของอินเตอร์เฟส
3. ความสัมพันธ์ระหว่างคลาส แสดงด้วยสัญลักษณ์ดังนี้
   - 3.1  แสดงสัญลักษณ์ของการสืบทอดระหว่างคลาส
   - 3.2  แสดงความสัมพันธ์ระหว่างคลาสแบบคอมโพสิชัน
   - 3.3 แสดงความสัมพันธ์ระหว่างคลาสแบบแอสโซซิเอชัน
   - 3.4  แสดงความสัมพันธ์ระหว่างคลาสแบบแอกกรีเกชัน
   - 3.5  แสดงความสัมพันธ์ระหว่างคลาสแบบดีเพนเดนซี
   - 3.6  แสดงความสัมพันธ์ของสืบทอดระหว่างคลาสกับอินเตอร์เฟส

1. แถบกลุ่มสัญลักษณ์สำหรับการจัดการคำสั่งพื้นฐานต่าง ๆของเครื่องมือ มีดังนี้
   1)  แสดงปุ่มสำหรับการเปิดคลาสไดอะแกรมที่สร้างไว้
   2)  แสดงปุ่มสำหรับบันทึกรูปภาพคลาสไดอะแกรมที่สร้าง
   3)  แสดงปุ่มสำหรับแก้ไขคลาสไดอะแกรมที่สร้าง
   4)  แสดงปุ่มสำหรับสร้างสคีมาฐานข้อมูลเชิงวัตถุจากคลาสไดอะแกรม
   5)  แสดงปุ่มสำหรับลบคลาสไดอะแกรมที่สร้าง
   6)  แสดงปุ่มสำหรับปิดหน้าต่างของโปรแกรม
   7)  แสดงปุ่มสำหรับการพิมพ์คลาสไดอะแกรม

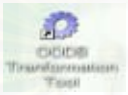
จ.2 วิธีการใช้งานเครื่องมือและขั้นตอนการสร้างสคีมาฐานข้อมูลเชิงวัตถุ

หลังจากผู้ใช้งานเครื่องมือการแปลงยูเอ็มแอลคลาสไดอะแกรมเป็นสคีมาฐานข้อมูลเชิงวัตถุได้เก็บความต้องการของผู้ใช้ระบบและนำข้อกำหนดความต้องการซอฟต์แวร์ มาออกแบบเป็นคลาสไดอะแกรมเรียบร้อยเมื่อต้องการสร้างสคีมาฐานข้อมูลเชิงวัตถุจากคลาสไดอะแกรมสามารถทำได้ตามขั้นตอนดังนี้

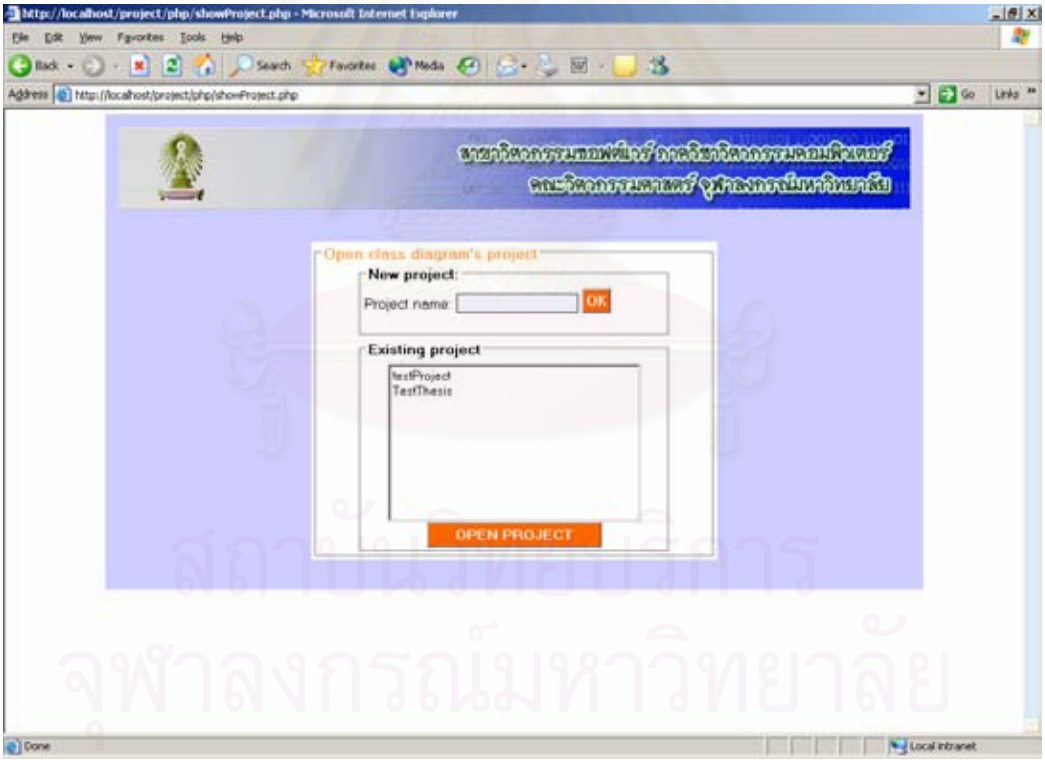
1.ตรวจสอบความถูกต้องของการเก็บมูลข้อกำหนดความต้องการซอฟต์แวร์

2.ออกแบบคลาสไดอะแกรมตามข้อกำหนดความต้องการซอฟต์แวร์

3.นำคลาสไดอะแกรมที่ได้ออกแบบไว้ มาเข้าสู่เครื่องมือ ดังนี้

3.1 คลิกที่ปุ่ม  เพื่อเข้าสู่หน้าแรกของเครื่องมือหรือคลิกที่ไอคอนอินเตอร์เน็ตเอ็กซ์โพเลอร์ แล้วพิมพ์ที่อยู่ของเครื่องมือในหน้าช่อง Address ดังรูปที่ จ.1

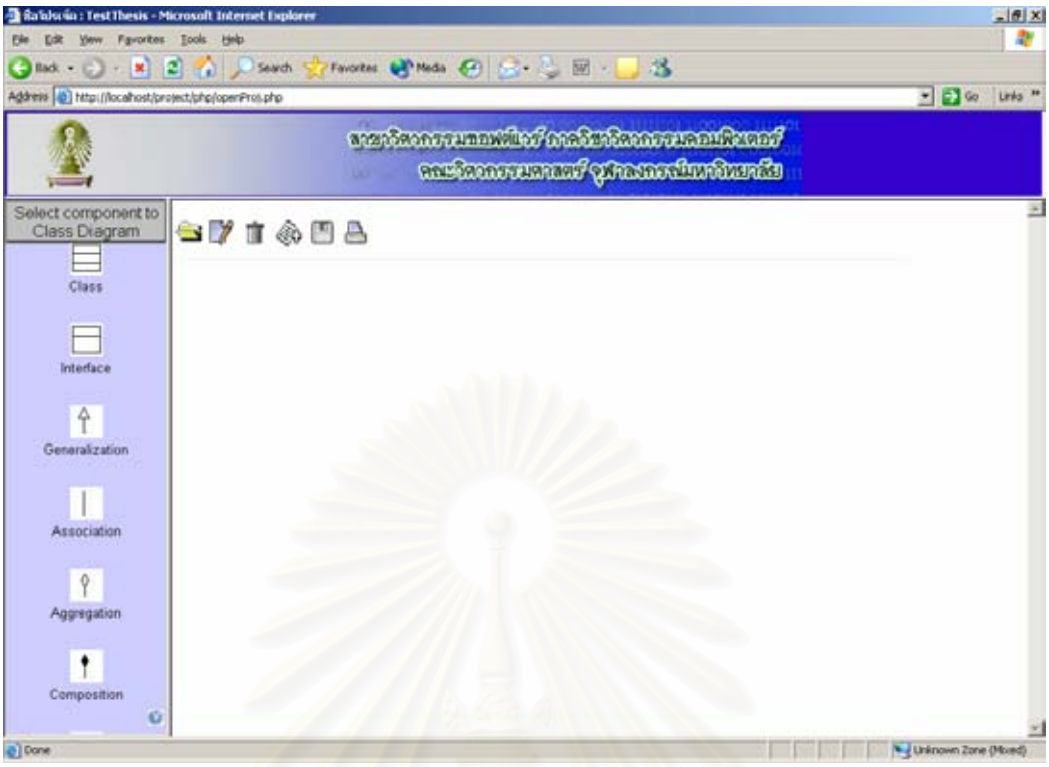
3.2 กดปุ่มตกลงที่หน้าจอแรกของเครื่องมือเพื่อเข้าสู่โปรแกรมจะปรากฏหน้าจอสำหรับเลือกชื่อโปรเจ็คที่ได้สร้างไว้แล้ว ดังรูปที่ จ.2



รูปที่ จ.2 แสดงหน้าจอสำหรับการเลือกโปรเจ็คหรือสร้างโปรเจ็คใหม่

3.3 เลือกชื่อโปรเจ็คของคลาสไดอะแกรมที่ได้เคยสร้างไว้ก่อนหน้าดังรูปที่ จ.2 หรือคลิกที่ปุ่มสร้างโปรเจ็คใหม่ เพื่อสร้างโปรเจ็คใหม่ที่ต้องการ

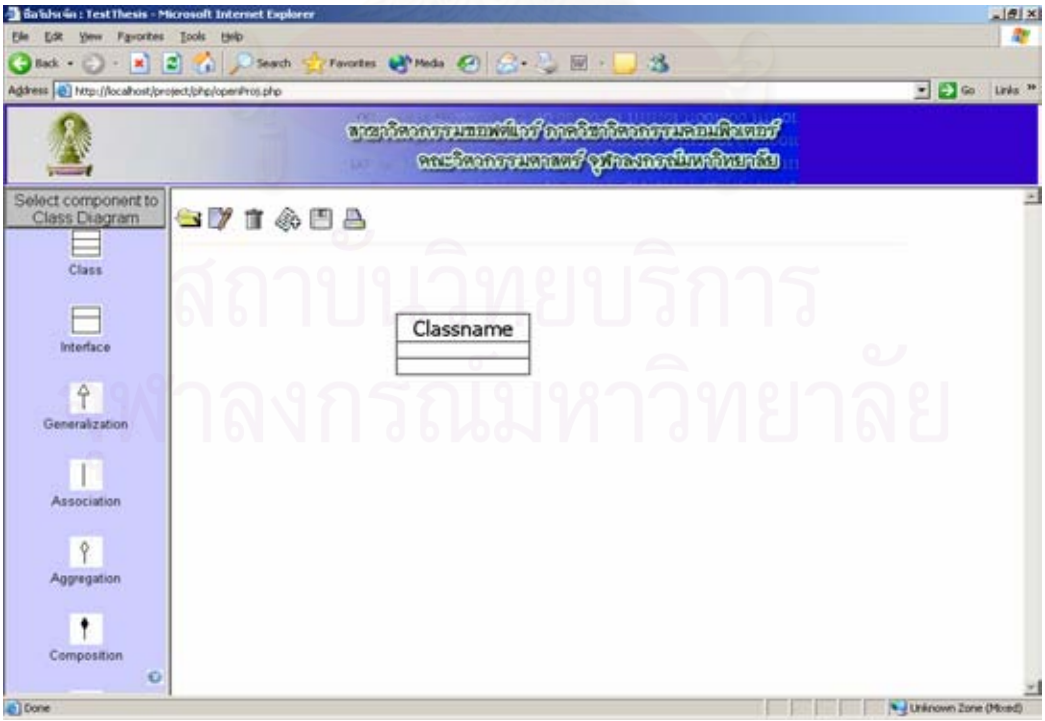
3.4 จะปรากฏหน้าจอหลักสำหรับวาดรูปคลาสไดอะแกรมดังรูปที่ จ.3



รูปที่ จ.3 แสดงหน้าจอหลักสำหรับวาดรูปคลาสไดอะแกรม

3.5 วาดคลาสไดอะแกรมตามที่ได้ออกแบบไว้ตามขั้นตอนดังนี้

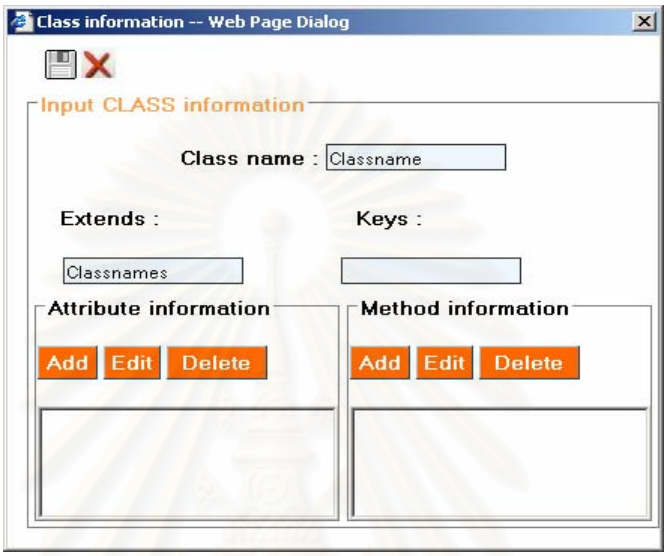
3.5.1 เลือกสัญลักษณ์ที่ต้องการวาดโดยการคลิกที่สัญลักษณ์ด้านซ้าย จากนั้นวางสัญลักษณ์ที่เลือกไว้ลงในตำแหน่งที่ต้องการ ดังรูปที่ จ.4



รูปที่ จ.4 แสดงตัวอย่างคลาสที่ถูกวาดจากเครื่องมือ

รายละเอียดของการวาดสัญลักษณ์แต่ละประเภทมีดังนี้

1.คลาส เมื่อผู้ใช้ต้องการวาดคลาสให้เคลิกที่รูป  จากนั้นนำมาวางบนหน้าจอที่ตำแหน่งที่ต้องการ ดับเบิ้ลคลิกที่รูปสัญลักษณ์จะปรากฏหน้าสำหรับการกรอกรายละเอียดของคลาสดังรูป

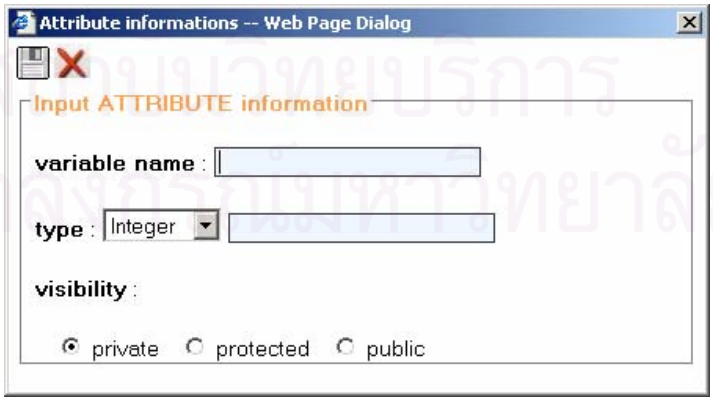


รูปที่ จ.5 แสดงหน้าจอสำหรับการกรอกรายละเอียดของคลาส

กรอกข้อมูลรายละเอียดของคลาสตามที่ได้ออกแบบไว้โดยข้อมูลต้องกรอกประกอบด้วย

1.ข้อมูลชื่อคลาส และข้อมูลระดับการมองเห็นของคลาส

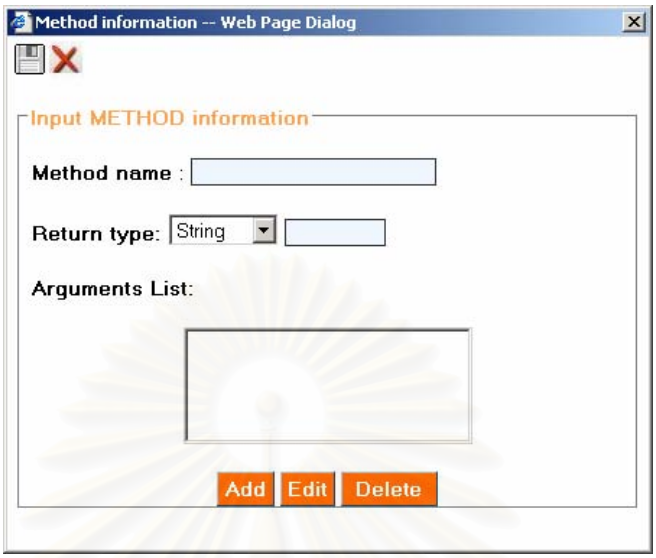
2.ข้อมูลแอทริบิวต์ โดยเมื่อผู้ใช้ต้องการระบุรายละเอียดของแอทริบิวต์ให้คลิกที่ Attribute จะปรากฏหน้าจอสำหรับการกรอกข้อมูลแอทริบิวต์ ดังรูป



รูปที่ จ.6 แสดงหน้าจอสำหรับการกรอกข้อมูลแอทริบิวต์

กรอกข้อมูล ชื่อแอทริบิวต์ ชนิดข้อมูลของแอทริบิวต์ ระดับการมองเห็นให้ครบ จากนั้นกดปุ่มตกลง

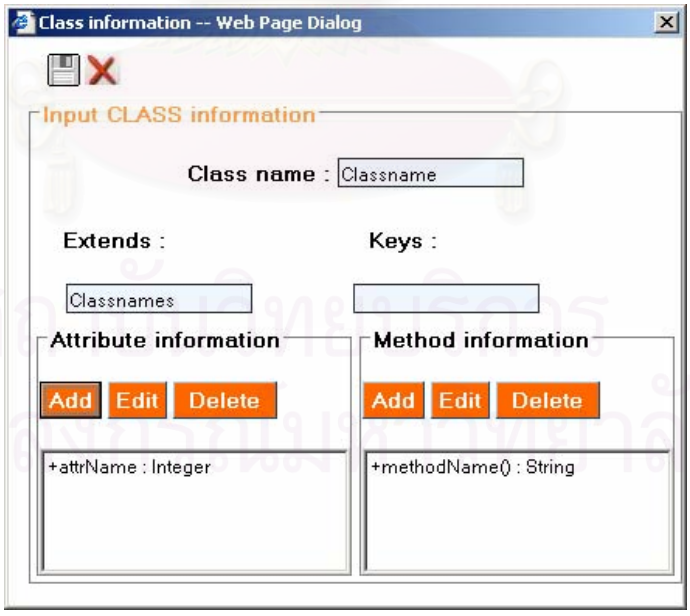
3.ข้อมูลเมทธอด เมื่อผู้ใช้ต้องการระบุรายละเอียดของเมทธอดให้คลิกที่ method จะปรากฏหน้าจอสำหรับการกรอกข้อมูลเมทธอด ดังรูป



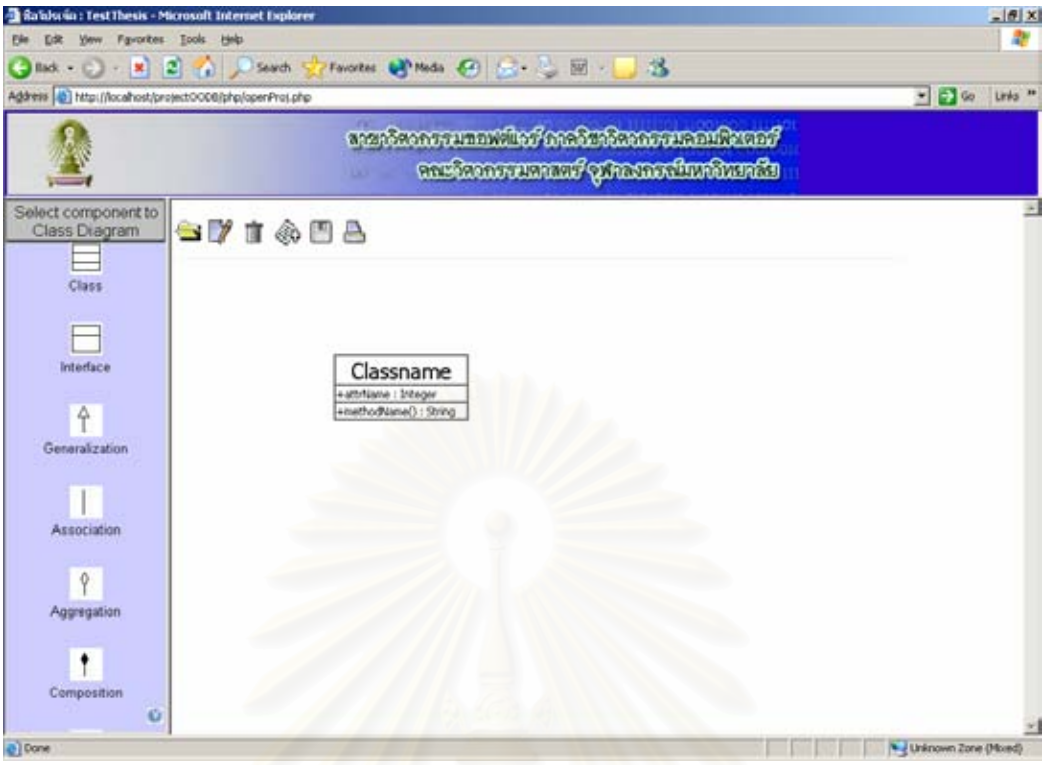
รูปที่ จ.7 แสดงหน้าจอสำหรับการกรอกข้อมูลเมทธอด

กรอกข้อมูล ชื่อเมทธอด ชนิดข้อมูลของเมทธอด ระดับการมองเห็นให้ครบ จากนั้นกดปุ่มตกลง

จากนั้นกดปุ่มตกลง ข้อมูลชื่อคลาส แอทริบิวต์ เมทธอด ระดับการมองเห็นจะถูกเพิ่มลงในสัญลักษณ์คลาสไดอะแกรม ดังรูป



รูปที่ จ.8 แสดงหน้าจอสำหรับการกรอกรายละเอียดของคลาสที่กรอกเรียบร้อย



รูปที่ จ.9 แสดงตัวอย่างคลาสที่ถูกวาดจากเครื่องมือ

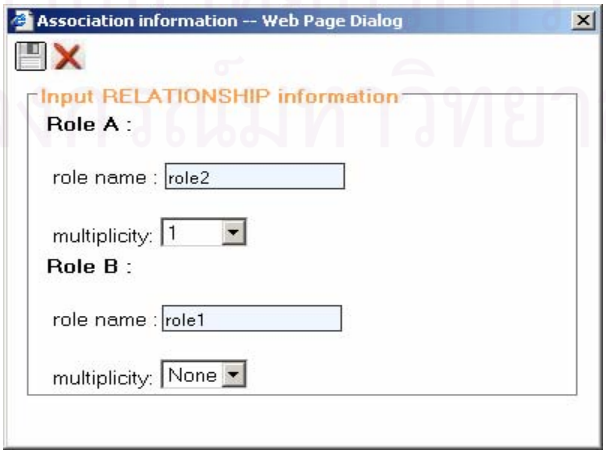
2.ความสัมพันธ์ระหว่างคลาส เมื่อผู้ใช้ต้องการวาดความสัมพันธ์ระหว่างคลาส 2 คลาสสามารถทำได้ดังนี้

2.1 วาดคลาสตามจำนวนที่ต้องการ

2.2 เลือกความสัมพันธ์ที่ต้องการจากเมนูด้านซ้ายโดยการคลิกที่สัญลักษณ์แต่ละรูปจากนั้น นำเมาส์มาคลิกที่คลาสที่ต้องการสร้างความสัมพันธ์ของคลาส 2 คลาส ที่ต้องการ

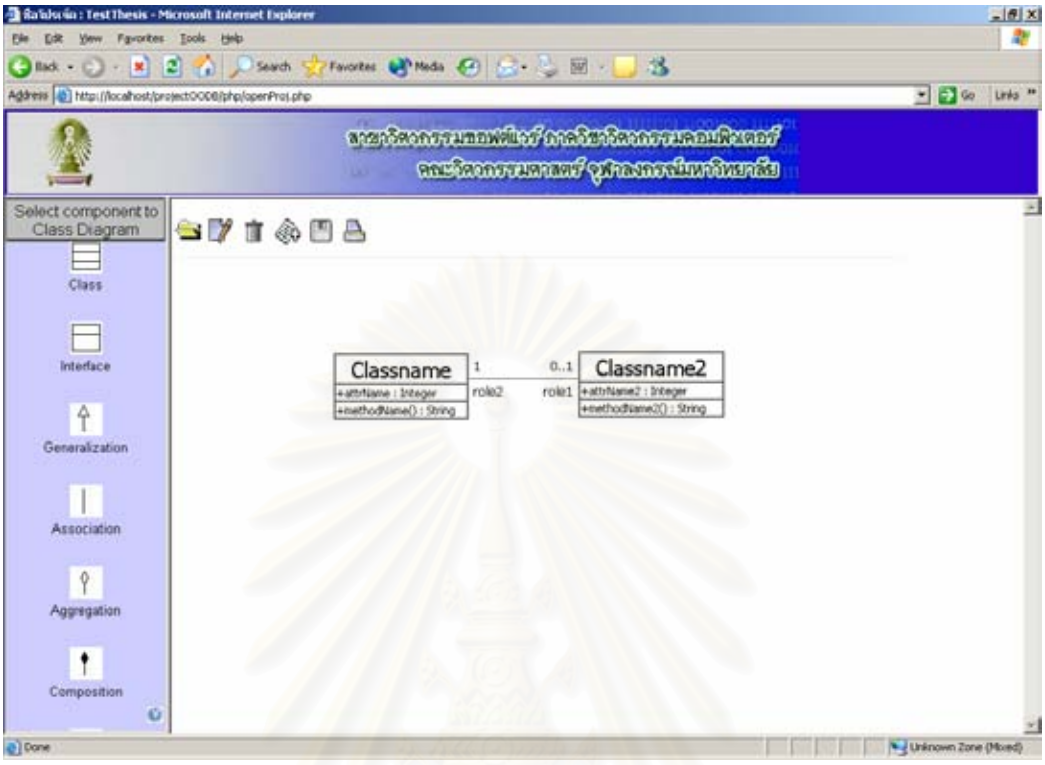
2.3 ระบุรายละเอียดของความสัมพันธ์ของคลาส โดยสามารถทำได้ดังนี้

1.คลิกที่เส้นความสัมพันธ์จะปรากฏหน้าจอสำหรับการระบุรายละเอียดความสัมพันธ์ดังรูป



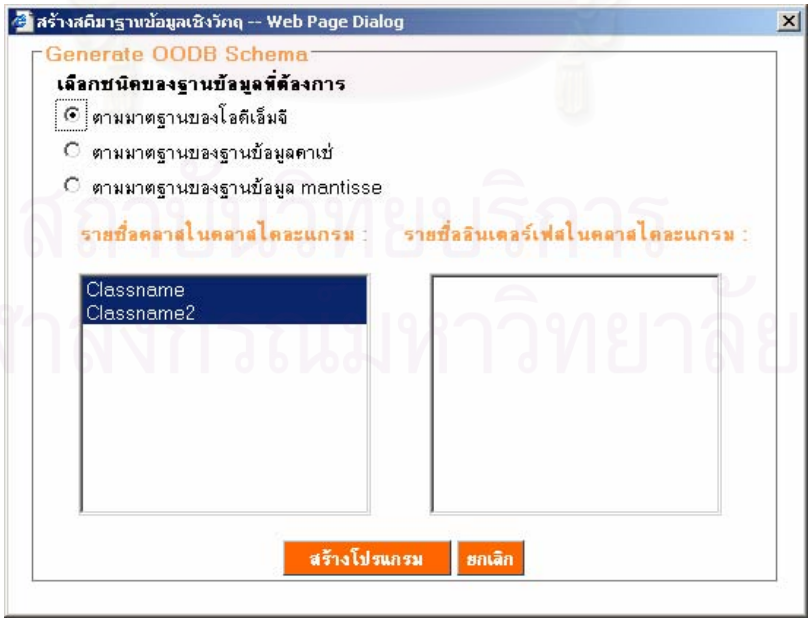
รูปที่ จ.10 แสดงหน้าจอสำหรับการระบุรายละเอียดความสัมพันธ์

ให้ผู้ใช้งานกรอกรายละเอียดความสัมพันธ์โดยข้อมูลที่ต้องระบุประกอบไปด้วยชื่อบทบาท มัลติพลิซิตี จากนั้นกดปุ่ม ตกลง ข้อมูลที่ได้จะปรากฏที่เส้นความสัมพันธ์ที่ได้เลือกไว้แล้วดังรูป



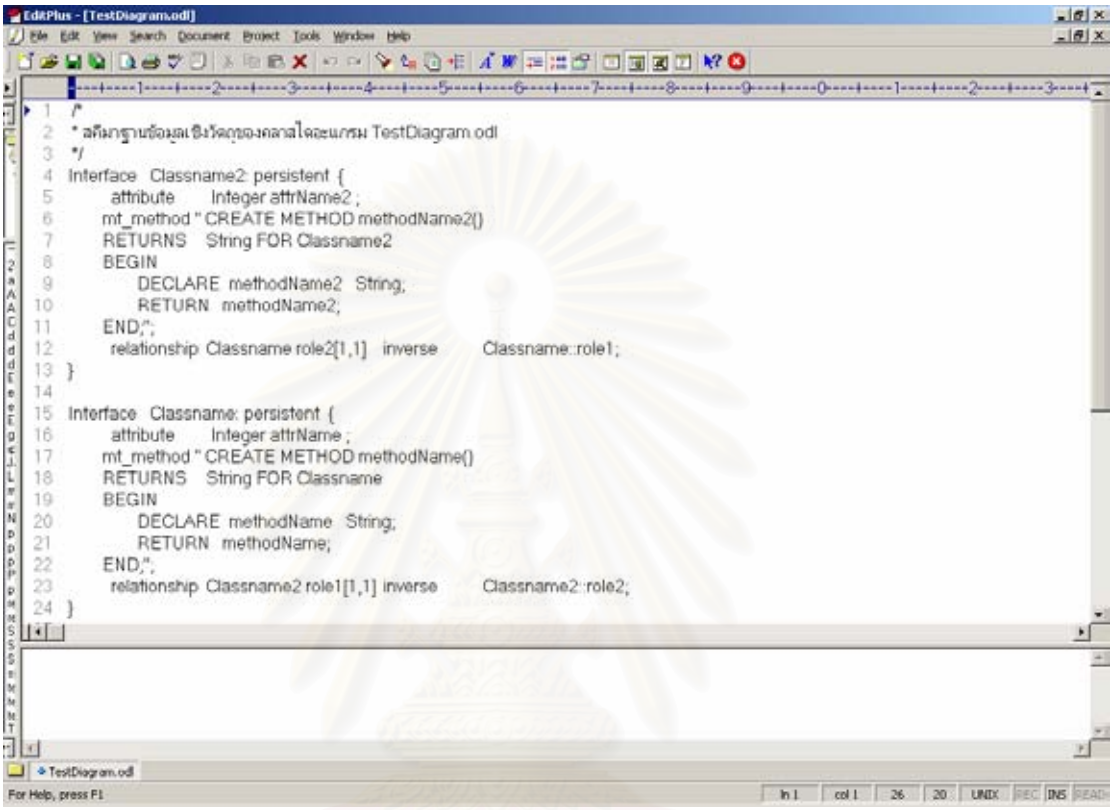
รูปที่ จ.11 แสดงตัวอย่างคลาสที่ถูกวาดจากเครื่องมือ

4.เมื่อวาดสัญลักษณ์ดังกล่าวเรียบร้อยกดปุ่ม  เพื่อบันทึกไดอะแกรม และกดปุ่ม  เพื่อสร้างเป็นสคีมาฐานข้อมูลจะปรากฏหน้าจอสำหรับเลือกชนิดฐานข้อมูลดังรูป



รูปที่ จ.12 แสดงหน้าจอสำหรับเลือกชนิดฐานข้อมูล

5.โปรแกรมจะสร้างสคีมาฐานข้อมูลเชิงวัตถุจากคลาสไดอะแกรมที่ได้ออกแบบไว้ตามชนิดของฐานข้อมูลที่เลือกไว้ให้ผู้ใช้กดปุ่มเพื่อบันทึกสคีมาที่ได้และนำสคีมาดังกล่าวไปสร้างฐานข้อมูลแต่ละชนิด ดังรูป



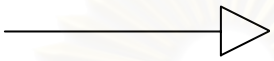
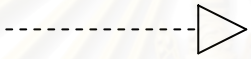
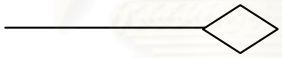
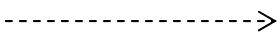
รูปที่ จ.13 แสดงสคีมาฐานข้อมูลเชิงวัตถุของฐานข้อมูลแมทิส ที่ได้จากเครื่องมือ

ภาคผนวก ฉ

รูปภาพสัญลักษณ์ต่าง ๆของคลาสไดอะแกรม

1. ความสัมพันธ์ระหว่างคลาส

ตารางที่ ฉ.1 แสดงสัญลักษณ์ความสัมพันธ์แบบต่าง ๆ ของคลาสไดอะแกรม

ชื่อความสัมพันธ์	สัญลักษณ์	รายละเอียดของสัญลักษณ์
Generalization		ใช้แสดงการสืบทอดระหว่างคลาสกับคลาส หรืออินเตอร์เฟสกับอินเตอร์เฟสโดยหัวลูกศรจะชี้ไปที่คลาสแม่ และหางลูกศรจะชี้ไปที่คลาสลูก
Realization		ใช้แสดงการสืบทอดระหว่างอินเตอร์เฟสกับคลาส
Association		ใช้แสดงความสัมพันธ์ระหว่างคลาสกับคลาส โดยมีบทบาทของคลาสอยู่ปลายแต่ละด้านของความสัมพันธ์
Aggregation		เป็นความสัมพันธ์แบบแอสโซซิเอชันชนิดหนึ่ง ใช้แสดงความสัมพันธ์ในการเป็นส่วนหนึ่งของกันและกันระหว่าง คลาสกับ คลาสโดยด้านหัวที่เป็นข้าวหลามตัดจะชี้ไปที่ คลาสที่แสดงส่วนรวมทั้งหมดด้านปลายจะชี้ไปที่คลาสที่ส่วนหนึ่งของด้านหัว
Composition		เป็นความสัมพันธ์แบบแอกกริเกชันแบบหนึ่ง โดยคลาสที่เป็นส่วนหนึ่งนั้น จะอยู่หรือหายไปพร้อมกับคลาสที่เป็นส่วนรวมทั้งหมด
Dependency		เป็นความสัมพันธ์ที่แสดงถึงการขึ้นต่อกันระหว่าง คลาสกับ คลาสโดยหัวลูกศรจะชี้ไปที่คลาสที่เมื่อมีการเปลี่ยนแปลงแล้วจะส่งผลต่อคลาสที่อยู่ปลายลูกศรด้วย

2. ระดับการมองเห็น

จะแสดงถึงความสามารถในการมองเห็นแอทริบิวต์หรือเมทธอด มี 3 แบบ คือ

+ แทน public

แทน protected

- แทน private

3. มัลติพลิซิตี

มัลติพลิซิตี จะเขียนอยู่ปลายแต่ละด้านของความสัมพันธ์ จะใช้แสดงจำนวนวัตถุของคลาสที่ปลายด้านหนึ่งที่มีความสัมพันธ์กับ 1 วัตถุของคลาสที่ปลายอีกด้านหนึ่ง มีดังนี้

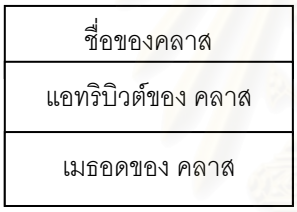
0..1 แทน zero or one

* แทน many

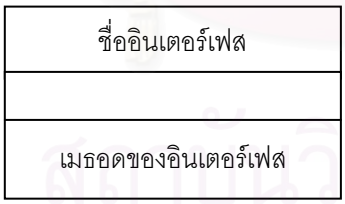
0..* แทน zero or many

1..* แทน one or many

4. คลาส แสดงด้วยสัญลักษณ์ดังนี้



5. อินเตอร์เฟส แสดงด้วยสัญลักษณ์ดังนี้



ภาคผนวก ซ

ตัวอย่างการตรวจสอบความถูกต้องของกฎ

จากกฎที่สร้างขึ้นทั้ง 11 ข้อและคลาสไดอะแกรมที่สร้างตามรูปที่ 6.2 สามารถนำมาตรวจสอบความถูกต้องของกฎได้ โดยการทดสอบกับตัวอย่างของคลาสบางคลาสของคลาสไดอะแกรม

โดยในที่นี้จะตรวจสอบความถูกต้องของกฎตามมาตรฐานโอดีเอ็มจีกับคลาส PERSON และคลาส EMPLOYEE และ DEPENDENT ดังนี้

```
<RULESCHEMA-OODB>    ::= {<CLASSDEF_NAME>}{<RELATIONSHIP_NAME>}

<CLASSDEF_NAME> ::=<CLASSHEADER_NAME>
                        "{"
                           {<CLASSBODY_NAME>}
                        "}"

<CLASSHEADER_NAME> ::= class PERSON
                       (extent  PERSONs)

<CLASSBODY_NAME>::={<ATTRIBUTE_NAME>}
                   {<METHOD_NAME>}

<METHOD_NAME> ::=<TYPE>  Age();
<TYPE>::= <PRIMITIVE>
<PRIMITIVE>::= String
== ><METHOD_NAME> ::= String  Age()

<ATTRIBUTE_NAME> ::= attribute  <TYPE>  firstName;
<TYPE>::= <PRIMITIVE>
<PRIMITIVE>::= String
== ><ATTRIBUTE_NAME> ::= attribute  String firstName;

<ATTRIBUTE_NAME> ::= attribute  <TYPE>  lastName;
<TYPE>::= <PRIMITIVE>
<PRIMITIVE>::= String
== ><ATTRIBUTE_NAME> ::= attribute  String lastName;

<ATTRIBUTE_NAME> ::= attribute  <TYPE>  titleName;
<TYPE>::= <PRIMITIVE>
<PRIMITIVE>::= String
== ><ATTRIBUTE_NAME> ::= attribute  String titleName;
...

<CLASSDEF_NAME> ::=<CLASSHEADER_NAME>
                        "{"
                           {<CLASSBODY_NAME>}
                        "}"

<CLASSHEADER_NAME> ::= class DEPENDENT
                       (extent  DEPENDENTs)

<CLASSBODY_NAME>::={<ATTRIBUTE_NAME>}
                   {<METHOD_NAME>}
```

```
<ATTRIBUTE_NAME> ::= attribute  <TYPE>  depenNo;
<TYPE>::= <PRIMITIVE>
<PRIMITIVE>::= String
== ><ATTRIBUTE_NAME> ::= attribute  String depenNo;

<ATTRIBUTE_NAME> ::= attribute  <TYPE>  birthDate;
<TYPE>::= <PRIMITIVE>
<PRIMITIVE>::= String
== ><ATTRIBUTE_NAME> ::= attribute  date birthDate;

<ATTRIBUTE_NAME> ::= attribute  <TYPE>  sex;
<TYPE>::= <PRIMITIVE>
<PRIMITIVE>::= String
== ><ATTRIBUTE_NAME> ::= attribute  String sex;
...

<CLASSDEF_NAME> ::=<CLASSHEADER_NAME>
                      "{"
                         {<CLASSBODY_NAME>}
                      "}"

<CLASSHEADER_NAME> ::= class EMPLOYEE
                    (extent  EMPLOYEEs)

<CLASSBODY_NAME>::={<ATTRIBUTE_NAME>}
                   {<METHOD_NAME>}

<METHOD_NAME> ::=<TYPE>  changDepName();
<TYPE>::= <PRIMITIVE>
<PRIMITIVE>::= String
== ><METHOD_NAME> ::= String  changDepName();

<METHOD_NAME> ::=<TYPE>  changProject();
<TYPE>::= <PRIMITIVE>
<PRIMITIVE>::= String
== ><METHOD_NAME> ::= String  changProject()

<ATTRIBUTE_NAME> ::= attribute  <TYPE>  empNo;
<TYPE>::= <PRIMITIVE>
<PRIMITIVE>::= String
== ><ATTRIBUTE_NAME> ::= attribute  String empNo;

<ATTRIBUTE_NAME> ::= attribute  <TYPE>  address;
<TYPE>::= <PRIMITIVE>
<PRIMITIVE>::= String
== ><ATTRIBUTE_NAME> ::= attribute  String address;

<ATTRIBUTE_NAME> ::= attribute  <TYPE>  hireDate;
<TYPE>::= <PRIMITIVE>
<PRIMITIVE>::= String
== ><ATTRIBUTE_NAME> ::= attribute  Date hireDate;
...

<RELATIONSHIP_NAME>::= {<RELATIONSHIP-H_NAME>}{<RELATIONSHIP-D_NAME>}

<RELATIONSHIP-H_NAME>::=[<GENERALIZATION>][<REALIZATION>]

<GENERALIZATION>::=EMPLOYEE:
                     extends  SUPERCLASSNAME_NAME


== ><GENERALIZATION> ::= EMPLOYEE:
                         extends  PERSON
```

```
<RELATIONSHIP-D_NAME>::= <RECURSIVEASSOCIATION>

<RELATIONSHIP-LIST_NAME> ::= relationship <COLLECTION-TYPE><CLASSNAME_NAME>

<RECURSIVEASSOCIATION> ::= EMPLOYEE:
                        <RELATIONSHIP-LIST_CLASS1> supervisor
                       inverse  EMPLOYEE:: supervisee ;

<RELATIONSHIP-LIST_NAME> ::= relationship EMPLOYEE

                        <RELATIONSHIP-LIST_CLASS1> supervisee
                       inverse  EMPLOYEE:: supervisor;

<RELATIONSHIP-LIST_NAME> ::= relationship set<EMPLOYEE>

== ><RECURSIVEASSOCIATION> ::=
       EMPLOYEE:
       relationship set<EMPLOYEE> supervisee
              inverse  EMPLOYEE:: supervisor;
      relationship EMPLOYEE supervisor
              inverse  EMPLOYEE:: supervisee ;

<COMPOSITION> ::= <COMPOSITION-CLASS1> <COMPOSITION-CLASS2>

<COMPOSITION-CLASS1>::= EMPLOYEE:
                        <RELATIONSHIP-LIST_CLASS2-1>   hasDpt
                       inverse  DEPENDENT:: dependentOf;

<RELATIONSHIP-LIST_NAME> ::= relationship DEPENDENT

                        attribute DEPENDENT DEPENDENTs;

                        attribute  string depenNo;

== ><COMPOSITION-CLASS1>::=
       EMPLOYEE:
         relationship DEPENDENT hasDpt
              inverse  DEPENDENT:: dependentOf;
         attribute DEPENDENT DEPENDENTs;
         attribute  string depenNo;

<COMPOSITION -CLASS2>::=DEPENDENT:
                        <RELATIONSHIP-LIST_CLASS1-2>   dependentOf
                       inverse  EMPLOYEE:: hasDpt;

<RELATIONSHIP-LIST_NAME> ::= relationship EMPLOYEE

== ><COMPOSITION-CLASS2>::=
       DEPENDENT:
       relationship EMPLOYEE dependentOf
       inverse  EMPLOYEE:: hasDpt;

== ><COMPOSITION> ::=
       EMPLOYEE:
         relationship DEPENDENT hasDpt
              inverse  DEPENDENT:: dependentOf;
         attribute DEPENDENT DEPENDENTs;
         attribute  string depenNo;
       DEPENDENT:
         relationship EMPLOYEE dependentOf
              inverse  EMPLOYEE:: hasDpt;
```

จะได้สคีมาตัวกลางสำหรับสร้างเป็นสคีมาฐานข้อมูลเชิงวัตถุ ดังนี้

```
== ><RULESCHEMA-OODB>   ::=
class  PERSON(extent  PERSONS)
{
       attribute string firstName;
       attribute string lastName;
       attribute string titleName;
       attribute Date  birthDate;
       attribute String sex;
       attribute String comment;
       Integer  Age();
}

class  EMPLOYEE extends  PERSON (extent  EMPLOYEES)
{
      attribute string empNo;
      attribute string address;
      attribute date hireDate;
      attribute string SSN;
      attribute string telNo;
      attribute string depenNo;
      string changDepName():
      string changProject():
}

class  DEPENDENT (extent DEPENDENTS  )
{
       attribute string depenNo;
       attribute date birthDate;
        attribute string sex ;
       attribute string relationship;
}


EMPLOYEE:
    extends  PERSON


EMPLOYEE:
    relationship  EMPLOYEE  supervisor inverse  EMPLOYEE:: supervisee;
    relationship set< EMPLOYEE>  supervisee inverse  EMPLOYEE:: supervisor ;

EMPLOYEE:
   relationship  DEPENDENT  hasDpt  inverse  DEPENDENT:: dependentOf;
   attribute DEPENDENT  dependent;
   attribute string depenNo;

DEPENDENT:
   relationship  EMPLOYEE dependentOf  inverse    EMPLOYEE::hasDpt;
```

นำสคีมาตัวกลางที่ได้ไปแปลง เพื่อนำความสัมพันธ์ของแต่ละคลาสไปลงที่คลาสที่ถูกต้องจะได้สคีมาฐานข้อมูลเชิงวัตถุที่สมบูรณ์ดังนี้

```
class EMPLOYEE extends PERSON (extent EMPLOYEEs)
{
  attribute string empNo;
  attribute string address;
  attribute date hireDate;
  attribute string SSN;
  attribute string telNo;
  attribute string depenNo;
  string changDepName ():
  string changProject ():
  attribute DEPENDENT dependent;
  attribute string depenNo;
  relationship DEPENDENT hasDpt inverse DEPENDENT ::dependentOf;
  relationship EMPLOYEE supervisor inverse EMPLOYEE::supervisee;
  relationship set<EMPLOYEE> supervisee inverse EMPLOYEE::supervisor;
}
class PERSON(extent PERSONs)
{
  attribute string firstName;
  attribute string lastName;
  attribute string titleName;
  attribute date birthDate;
  attribute string sex ;
  attribute string comment;
  integer Age():
}
class DEPENDENT (extent DEPENDENTs)
{
  attribute string depenNo;
  attribute date birthDate;
  attribute string sex ;
  attribute string relationship;
  relationship EMPLOYEE dependentOf inverse EMPLOYEE::hasDpt;
}
```

รูปที่ ช.1 แสดงตัวอย่างการตรวจสอบความถูกต้องของกฎ

ประวัติผู้เขียนวิทยานิพนธ์

นายสถิตย์ ประสมพันธ์ เกิดวันที่ 21 พฤษภาคม พ.ศ. 2522 สำเร็จการศึกษาปริญญาตรีวิทยาศาสตรบัณฑิตสาขาวิทยาการคอมพิวเตอร์จากมหาวิทยาลัยขอนแก่น ในปีการศึกษา 2544 จากนั้นเข้าศึกษาในหลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิชาวิศวกรรมซอฟต์แวร์ ที่คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ในปีการศึกษา 2545 โดยได้รับทุนโครงการพัฒนาอาจารย์สาขาวิทยาการคอมพิวเตอร์ จากสถาบันเทคโนโลยีพระจอมเกล้าพระนครเหนือ