โครงร่างโปรแกรมประยุกต์เชิงวัตถุสำหรับพัฒนาโปรแกรมประยุกต์ฐานข้อมูลเชิงสัมพันธ์

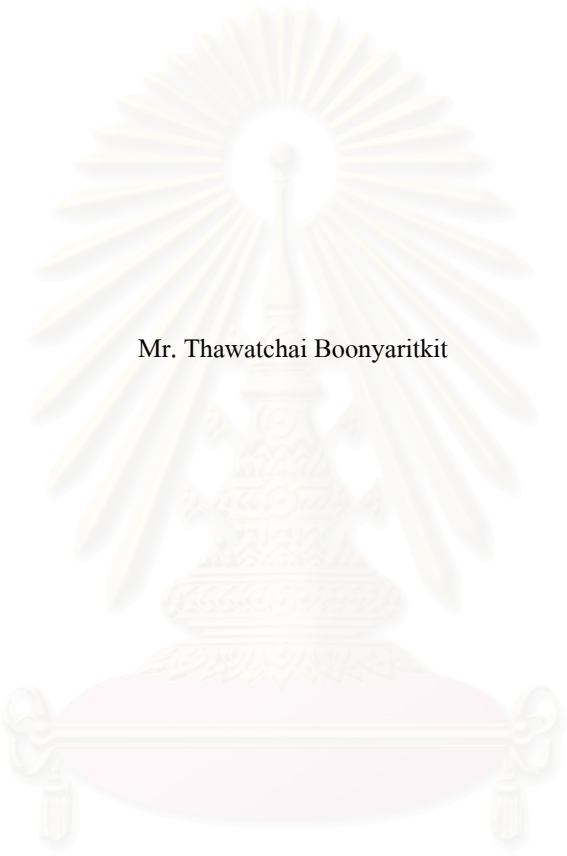
นายธวัชชัย บุญยฤทธิ์กิจ

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิทยาศาสตร์คอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2547
ISBN 974-53-1233-9
ลิขสิทธิ์ของจุฬาลงกรณ์มหาวิทยาลัย

OBJECT-ORIENTED APPLICATION FRAMEWORK FOR DEVELOPING RELATIONAL DATABASE APPLICATION

Mr. Thawatchai Boonyaritkit



สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

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

ISBN 974-53-1233-9

หัวข้อวิทยานิพนธ์	โครงร่างโปรแกรมประยุกต์เชิงวัตถุสำหรับพัฒนาโปรแกรมประยุกต์ฐานข้อมูลเชิงสัมพันธ์
โดย	นาย ธวัชชัย บุญยฤทธิ์กิจ
สาขาวิชา	วิทยาศาสตร์คอมพิวเตอร์
อาจารย์ที่ปรึกษา	ผู้ช่วยศาสตราจารย์ ดร.พรศิริ หมื่นไชยศรี

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

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

คณะกรรมการสอบวิทยานิพนธ์

…… ประธานกรรมการ
(ผู้ช่วยศาสตราจารย์ ดร.ทวิตีย์ เสนีวงศ์ ณ อยุธยา)

……... อาจารย์ที่ปรึกษา
(ผู้ช่วยศาสตราจารย์ ดร.พรศิริ หมื่นไชยศรี)

……กรรมการ
(อาจารย์ นครทิพย์ พร้อมพูล)

……กรรมการ
(อาจารย์ เชษฐ พัฒโนทัย)

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

ธวัชชัย บุญยฤทธิ์กิจ : โครงร่างโปรแกรมประยุกต์เชิงวัตถุสำหรับพัฒนาโปรแกรมประยุกต์ฐานข้อมูลเชิงสัมพันธ์. (OBJECT-ORIENTED APPLICATION FRAMEWORK FOR DEVELOPING RELATIONAL DATABASE APPLICATION) อ. ที่ปรึกษา : ผู้ช่วยศาสตราจารย์ ดร.พรศิริ หมื่นไชยศรี, 168 หน้า.
ISBN 974-53-1233-9.

วิทยานิพนธ์นี้มีวัตถุประสงค์เพื่อออกแบบและพัฒนาโครงร่างโปรแกรมประยุกต์เชิงวัตถุสำหรับการพัฒนาโปรแกรมประยุกต์ฐานข้อมูลเชิงสัมพันธ์ โครงร่างฯ นี้เป็นแนวทางหนึ่งในการนำกลับมาใช้ใหม่ในระดับโปรแกรมประยุกต์ ซึ่งทำให้การพัฒนาโปรแกรมประยุกต์เชิงวัตถุเพื่อจัดเก็บวัตถุในฐานข้อมูลเชิงสัมพันธ์ทำได้โดยง่าย นักออกแบบโปรแกรมประยุกต์เชิงวัตถุซึ่งไม่มีประสบการณ์ในการออกแบบการจัดเก็บวัตถุในฐานข้อมูลเชิงสัมพันธ์และโปรแกรมเมอร์ซึ่งไม่มีประสบการณ์ในการใช้งานฐานข้อมูลเชิงสัมพันธ์ สามารถนำโครงร่างฯไปใช้งาน โดยโครงร่างฯ รองรับการใช้งานฐานข้อมูลเชิงวัตถุในการเพิ่มวัตถุ การปรับปรุงวัตถุ การลบวัตถุ การดึงวัตถุกลับมาใช้งาน ภายใต้ความสัมพันธ์คลาสแบบ ซิงเกิลคลาส การรับทอดคลาส ภาพรวมกลุ่มคลาส และคอมโพสิตชันคลาส และประโยคคำสั่งเอสคิวแอลถูกซ่อนภายใต้โครงร่างฯ โดยโปรแกรมเมอร์ไม่ต้องเขียนคำสั่งเอสคิวแอลไว้ภายในโปรแกรมประยุกต์เชิงวัตถุ

หลังจากพัฒนาโครงร่างฯ และนำไปใช้งานจริงกับภาษา C++ ร่วมกับฐานข้อมูลเชิงสัมพันธ์ MySQL และนำไปพัฒนาระบบการสั่งซื้อสินค้าแบบง่าย พบว่าใช้เวลาในการออกแบบและสร้างชุดคำสั่งโดยรวมน้อยกว่าไม่ใช้โครงร่างฯ เนื่องด้วยการใช้โครงร่างฯ คือการนำกลับมาใช้ใหม่แบบหนึ่งย่อมทำให้การพัฒนาเร็วขึ้น เนื่องจากไม่จำเป็นต้องทำการออกแบบและเขียนชุดคำสั่งที่โครงร่างฯจัดการให้ การบำรุงรักษาทำได้ง่ายกว่าไม่ใช้โครงร่างฯ และไม่พบคำสั่งเอลคิวแอลในรหัสคำสั่งของโปรแกรมประยุกต์เชิงวัตถุ แต่นักวิเคราะห์และโปรแกรมเมอร์จะมีช่วงเวลาในการศึกษาเรียนรู้การใช้งานโครงร่างฯ อยู่ระดับหนึ่งด้วยเช่นกัน

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

ภาควิชา...วิศวกรรมคอมพิวเตอร์..... ลายมือชื่อนิสิต.....
สาขาวิชา..วิทยาศาสตร์คอมพิวเตอร์..... ลายมือชื่ออาจารย์ที่ปรึกษา.....
ปีการศึกษา 2547

4470343221 : MAJOR COMPUTER SCIENCE

KEY WORD: OBJECT-ORIENTED APPLICATION FRAMEWORK / RELATIONAL DATABASE APPLICATION / APPLICATION REUSING / APPLICATION FRAMEWORK DESIGN / PERSISTENCE OBJECT

THAWATCHAI BOONYARITKIT : OBJECT-ORIENTED APPLICATION FRAMEWORK FOR DEVELOPING RELATIONAL DATABASE APPLICATION. THESIS ADVISOR : ASSISTANT PROFESSOR PORNSIRI MUENCHAISRI, Ph.D., 168 pp. ISBN 974-53-1233-9.

The objective of this thesis is to design and develop the object-oriented application framework for relational database application development. This framework is aspect for reusing in application level. With framework, application development will be easier. The novice programmers or designers can use the framework for developing relational database application by inserting, updating, deleting or selecting the objects under the class relationship, which are single class, inheritance class, aggregation class and composition class. Programmers do not have to code SQL statement within application program since SQL statement will be hidden within the framework.

The developed framework that is implemented with C++ language and MySQL the relational database will reduce the time used for designing and programming of the Sale Order application's development. Thus, these inexperienced programmers need not to be specialized in database design and management. Furthermore, By using this framework, the maintenance process will be more convenient than not using the framework and the code will be used without SQL statement. However, the analyst and programmer need more time to learn the framework at the beginning of the usage. Once they are familiar with the framework, they can construct application faster.

Department..Computer Engineering.....Student’s signature.....

Field of study..Computer Science..... Advisor’s signature.....

Academic year 2004

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงไปได้ด้วยความช่วยเหลือจาก ผู้ช่วยศาสตราจารย์ ดร. พรศิริ หมื่นไชยศรี ผู้ช่วยศาสตราจารย์ ทวิตีย์ เสนีวงศ์ ณ อยุธยา อาจารย์ นครทิพย์ พร้อมพูล อาจารย์ เชษฐ พัฒโนทัย ขอขอบพระคุณที่ท่านได้ให้คำแนะนำ และข้อเสนอแนะต่างๆ ตลอดระยะเวลาของการจัดทำวิทยานิพนธ์ สุดท้ายนี้ขอกราบขอบพระคุณบิดา มารดา ขอบคุณเพื่อนๆ ทุกคนที่เป็นกำลังใจ และให้ความสนับสนุนมาโดยตลอด

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

สารบัญ

หน้า

สารบัญตาราง

หน้า

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

สารบัญภาพ

หน้า

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

บทที่ 1

บทนำ

1.1 ความเป็นมา และความสำคัญของปัญหา

การพัฒนาโปรแกรมประยุกต์ด้วยภาษาเชิงวัตถุร่วมกับฐานข้อมูลเชิงวัตถุในปัจจุบันไม่ได้รับความนิยมในการใช้งานมากนัก ถึงแม้ว่าไม่มีงานสำรวจที่ชัดเจน แต่จากการพิจารณาระบบต่างๆ ที่มีอยู่ในปัจจุบัน พบว่าโดยมากยังคงใช้ฐานข้อมูลเชิงสัมพันธ์ร่วมกับภาษาเชิงวัตถุ จะด้วยสาเหตุหลายประการเช่น ฐานข้อมูลเชิงวัตถุมีราคาแพง การพัฒนาเป็นผลิตภัณฑ์สินค้ามีไม่มากนัก การเรียนรู้เพื่อการใช้งานฐานข้อมูลเชิงวัตถุมีความแตกต่างจากการใช้งานฐานข้อมูลเชิงสัมพันธ์ การวิจัยฐานข้อมูลเชิงวัตถุมีมาเป็นเวลานานแต่การนำมาใช้งานยังมีความยุ่งยากอยู่มาก

ปัจจุบันการใช้งานฐานข้อมูลเชิงสัมพันธ์ร่วมกับภาษาเชิงวัตถุในการพัฒนาโปรแกรมประยุกต์มีความนิยมมากกว่าการใช้งานร่วมกับฐานข้อมูลเชิงวัตถุ หากแต่มีปัญหาอยู่หลายประการ เช่น ความซ้ำซ้อนของชุดคำสั่งในการเชื่อมต่อกับฐานข้อมูลเชิงสัมพันธ์ ความยุ่งยากในการออกแบบและเขียนชุดคำสั่งให้ฐานข้อมูลเชิงสัมพันธ์เสมือนฐานข้อมูลเชิงวัตถุในมุมมองของนักพัฒนาโปรแกรมประยุกต์ในเรื่องการจัดเก็บวัตถุ ซึ่งในทางปฏิบัติในการจัดเก็บคุณลักษณะ (Attribute) ของวัตถุเป็นเขตข้อมูล (Field) ของตารางความสัมพันธ์ในฐานข้อมูลเชิงสัมพันธ์ และเมื่อต้องการเรียกใช้วัตถุสามารถนำกลับมาใช้ได้โดยการนำคุณลักษณะที่เก็บไว้ในฐานข้อมูลเชิงสัมพันธ์มากำหนดคุณลักษณะเริ่มต้นให้วัตถุที่สร้างขึ้นใหม่เสมือนเรียกวัตถุมาใช้จากฐานข้อมูลเชิงวัตถุเป็นต้น การพัฒนาโปรแกรมจะมีรหัสคำสั่งในการเพิ่ม การแก้ไข การลบ การเลือก ในลักษณะเดียวกันเป็นจำนวนมากก่อให้เกิดความซ้ำซ้อนของรหัสคำสั่งและการทำงานกับฐานข้อมูลเชิงสัมพันธ์ ทั้งนี้เพื่อลดความซ้ำซ้อนดังกล่าวและสร้างแม่แบบ (Pattern) ในการออกแบบจึงมีความจำเป็นที่จะจัดการเพื่อลดความซ้ำซ้อนดังกล่าว โดยการสร้างโครงร่างโปรแกรมประยุกต์เชิงวัตถุเพื่อให้นักพัฒนาโปรแกรมประยุกต์สามารถใช้ฐานข้อมูลเชิงสัมพันธ์เสมือนฐานข้อมูลเชิงวัตถุ

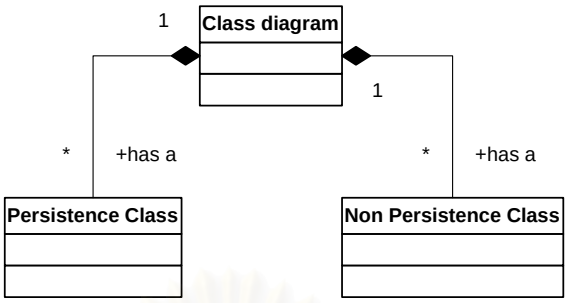
การนำกลับมาใช้ใหม่มีอยู่หลายวิธี เช่น การนำกลับมาใช้ใหม่ของการออกแบบ (Design pattern reusing) การนำกลับมาใช้ใหม่ของชุดรหัสคำสั่ง (Class library reusing) เป็นต้น หากแต่การนำกลับใช้ใหม่ที่กล่าวมานั้นจะไม่ขึ้นต่อกัน ทำให้มีวิธีการนำไปใช้ได้หลากหลาย ทำให้

การควมคุมคุณภาพทำได้ยาก เนื่องจากการออกแบบอาจจะมีคุณภาพ แต่ชุดรหัสคำสั่งอาจไม่มีคุณภาพหรือในทางตรงข้ามกันเป็นต้น การนำกลับมาใช้ใหม่ในระดับที่สูงขึ้น เช่น การใช้โครงร่างโปรแกรมประยุกต์เชิงวัตถุมีลักษณะต่างไปจากที่กล่าวคือรหัสคำสั่งและการออกแบบต้องมีความสัมพันธ์กันซึ่งจะลดข้อจำกัดในเรื่องการควบคุมภาพ

โครงร่างโปรแกรมประยุกต์เชิงวัตถุ ต่อจากนี้จะใช้คำว่า โครงร่างฯ แทนคำว่า โครงร่างโปรแกรมประยุกต์เชิงวัตถุ หมายถึง การนำกลับมาใช้ใหม่ของการออกแบบและชุดคำสั่งที่ได้ออกแบบและพัฒนาไว้เพื่อใช้ในการพัฒนาโปรแกรมประยุกต์ในขอบเขตที่โครงร่างฯนั้นรองรับ โดยการออกแบบในโครงร่างฯจะเตรียมส่วนของการทำงานร่วมกันของคลาสต่างๆในโครงร่างฯ และส่วนของชุดคำสั่งในโครงร่างฯจะเตรียมส่วนของการทำงานต่างๆภายในคลาสต่างๆที่อยู่ภายในโครงร่างฯนั้น ผลจากการเตรียมการทั้งสามทำให้คุณภาพของโปรแกรมประยุกต์ที่ใช้โครงร่างฯใกล้เคียงกัน เนื่องจากใช้วิธีการออกแบบและชุดรหัสคำสั่งเดียวกัน ซึ่งถูกกำหนดการพัฒนาโดยโครงร่างฯ

ดังนั้นการพัฒนาโครงร่างฯเพื่อใช้ในการพัฒนาโปรแกรมประยุกต์ฐานข้อมูลเชิงสัมพันธ์จึงเป็นแนวทางในการนำกลับมาใช้ใหม่ของการออกแบบและชุดคำสั่งที่จำเป็นในการพัฒนาโปรแกรมประยุกต์ด้วยภาษาเชิงวัตถุร่วมกับฐานข้อมูลเชิงสัมพันธ์ เป็นการนำกลับมาใช้ใหม่ทั้งในรูปของการออกแบบและชุดคำสั่ง จึงเป็นการลดต้นทุนในการพัฒนาและเพิ่มคุณภาพของซอฟต์แวร์ (Software)

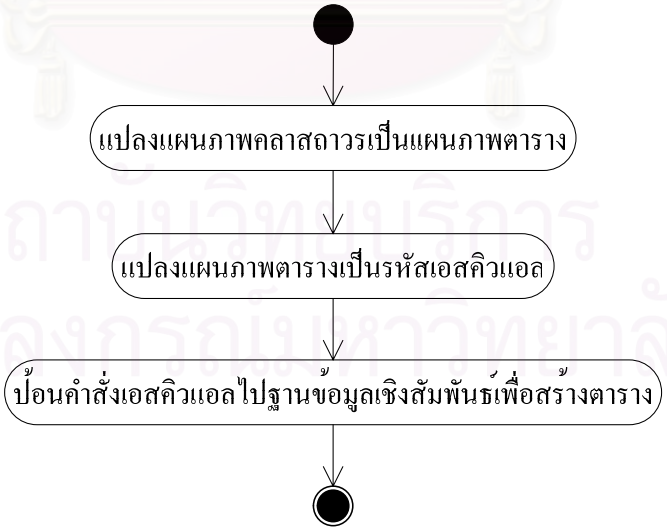
กระบวนการวิเคราะห์และออกแบบเชิงวัตถุ (Object-Oriented Analysis and Design) ทำเพื่อสร้างแบบจำลองเชิงวัตถุ (Object Model) เพื่อนำไปใช้งาน (Implementation) โดยภายใต้แบบจำลองเชิงวัตถุจะประกอบด้วย 2 กลุ่มคือ คลาสไม่ถาวร (Non Persistence Class) และคลาสที่ถาวร (Persistence Class) ดังแสดงในรูปที่ 1.1 คลาสที่ถาวรเป็นคลาสที่ต้องการจัดเก็บเพื่อคงความเป็นถาวรของวัตถุโดยข้อมูลของวัตถุจะคงอยู่ เช่นคลาสลูกค้าที่อ้างถึงคลาสการสั่งซื้อข้อมูบการสั่งซื้อจะถูกบันทึกไว้โดยทำการจัดเก็บลงในฐานข้อมูลเชิงสัมพันธ์เป็นต้น ในการคงสภาพของวัตถุนั้นมีแนวทางในการจัดเก็บวัตถุลงในฐานข้อมูลเชิงสัมพันธ์ ส่วนคลาสที่ไม่ถาวรจะเป็นคลาสที่ไม่ต้องการจัดเก็บในระบบซึ่งสามารถสร้างขึ้นในขณะที่ระบบทำงาน (run time) เช่น คลาสเพื่อการติดต่อกับผู้ใช้ เป็นต้น



รูปที่ 1.1 แผนภาพคลาสประกอบด้วยคลาสไม่ถาวรและคลาสถาวร

เนื่องจากฐานข้อมูลเชิงสัมพันธ์ไม่สนับสนุนเทคโนโลยีเชิงวัตถุ การจัดเก็บวัตถุจึงจัดเก็บเพียงคุณลักษณะของวัตถุเท่านั้น ดังนั้นคลาสที่จัดเก็บนั้นจึงต้องมีเมท็อด (Method) สำหรับการเพิ่ม (Insert) สำหรับการปรับปรุง (Update) สำหรับการลบ (Delete) สำหรับการเลือก (Select) เป็นต้น เพื่อสามารถปฏิสัมพันธ์กับฐานข้อมูลเชิงสัมพันธ์ได้ ซึ่งสรุปเป็นแนวทางการใช้เทคโนโลยีเชิงวัตถุกับฐานข้อมูลเชิงสัมพันธ์แสดงในรูปที่ 1.2 [1] ดังนี้

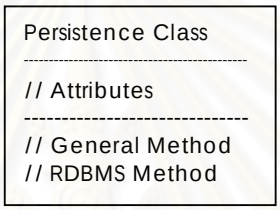
1. สร้างแบบจำลองตาราง (Table Model) ซึ่งเปลี่ยนจากแบบจำลองวัตถุของคลาสถาวร
2. นำแบบจำลองตารางเปลี่ยนเป็นการออกแบบตารางเค้าร่าง (Schema Table)
3. นำรหัสเอสคิวแอลไปสร้างตารางในฐานข้อมูลเชิงสัมพันธ์



รูปที่ 1.2 กิจกรรมในการแปลงแผนภาพคลาสถาวรเพื่อสร้างตารางในฐานข้อมูลเชิงสัมพันธ์

เมื่อพิจารณาส่วนคลาสถาวรในต้นแบบวัตถุจะเป็นดังรูปที่ 1.3 จะพบว่าคลาสถาวรที่ปฏิสัมพันธ์กับฐานข้อมูล มักประกอบด้วย

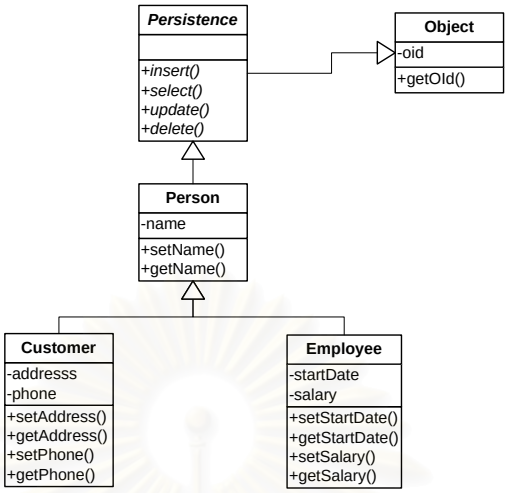
1. เมท็อดทำหน้าที่ทั่วไป (General Method) สำหรับกระทำสิ่งใดสิ่งหนึ่งกับคุณลักษณะ เช่น setName() เพื่อกำหนดค่าชื่อวัตถุ getName() เพื่อขอชื่อวัตถุ เป็นต้น
2. เมท็อดสำหรับปฏิสัมพันธ์กับฐานข้อมูลเชิงสัมพันธ์ (RDBMS Method) เช่น insert() ส่วนที่เป็นการเพิ่มใช้สำหรับเพิ่มคุณลักษณะวัตถุในฐานข้อมูลเชิงสัมพันธ์ update() การปรับปรุงใช้สำหรับปรับปรุงคุณลักษณะวัตถุในฐานข้อมูลเชิงสัมพันธ์ delete() การลบใช้สำหรับลบคุณลักษณะวัตถุในฐานข้อมูลเชิงสัมพันธ์ select() การเลือกใช้สำหรับเลือกคุณลักษณะวัตถุในฐานข้อมูลเชิงสัมพันธ์ เป็นต้น



รูปที่ 1.3 แสดงส่วนประกอบของคลาสถาวรที่ปฏิสัมพันธ์กับฐานข้อมูลเชิงสัมพันธ์

ภาษาเอสคิวแอล (SQL Language) เป็นภาษาที่ใช้ในฐานข้อมูลเชิงสัมพันธ์ ดังนั้นเมื่อเขียนโปรแกรมเพื่อทำงานร่วมกับฐานข้อมูลเชิงสัมพันธ์ จะใช้การฝังภาษาเอสคิวแอล (Embedded SQL) ไว้ในโปรแกรมเพื่อสั่งให้ฐานข้อมูลเชิงสัมพันธ์ทำงานใดๆ เช่นการใช้ภาษาจาวา (JAVA) กับ ฐานข้อมูลเชิงสัมพันธ์ผ่านเจดีบีซี (Java Database Connectivity: JDBC) โดยการฝังคำสั่งเอสคิวแอล (SQL Command) ดังตัวอย่างรูปที่ 1.4

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย



รูปที่ 1.4 ตัวอย่างแผนภาพคลาสที่มีคลาสถาวร

จากรูปที่ 1.4 คลาสลูกค้า (Customer) การรับทอดคุณสมบัติมาจากคลาสบุคคล (Person) และคลาสบุคคลการรับทอดคุณสมบัติมาจากคลาสถาวร (Persistence) คลาสถาวรการรับทอดคุณสมบัติมาจากคลาสวัตถุ (Object) ตามลำดับ คลาสถาวรมีเมท็อดสำหรับเพิ่ม สำหรับปรับปรุง สำหรับเลือก สำหรับลบ เป็นเมท็อดนามธรรม (Abstract method) เมื่อเขียนโปรแกรมภาษาจาวาดังรูปที่ 1.5

```
001: public class Object
002: {
003:     private String oid;
004:
005:     public String getOId();
006:     public void setOId(String newoid);
007: };
008:
009: public Interface Persistence
010: {
011:     public void insert();
012:     public void update();
013:     public void delete();
014:     public void select();
015: };
016:
017: public class Person extends Object implements Persistence
018: {
019:
020:     private String name;
021:
022:     public void setName(String newname);
023:     public String getName();
024:
025:     public void insert();
026:     public void update()
027:     {
028:             String sqlcmd;
029:             ...
030:             sqlcmd = "update Person set name='" + name + "' where oid='" +
oid + "'";
031:             ...
032:     }
033:
034:     public void delete();
035:     public void select();
036: };
037:
038: public class Customer extends Person implements Persistence
039: {
040:     private String address;
```

```
041:     private String phone;
042:
043:     public void setAddress(String newaddr);
044:     public String getAddress();
045:     public void setPhone(String newphone);
046:     public String getPhone();
047:
048:     public void insert()
049:     public void update()
050:     {
051:             String sqlcmd;
052:
053:             ...
054:             super.update();
055:             sqlcmd = "update Customer set address='" + address + "',
phone='" +
056:                                                              phone +"'
where id='" + oid + "'";
057:             ...
058:     }
059:     // extend
060:     public void update_name()
061:     {
062:             String sqlcmd;
063:             ...
064:             super.update();
065:             ...
066:     }
067:     public void update_address()
068:     {
069:             String sqlcmd;
070:
071:             ...
072:             sqlcmd = "update Customer set address='" + address + "' where
id='" + 073:
                        oid + "'";
074:             ...
075:     }
076:     public void update_phone()
077:     {
078:             String sqlcmd;
079:             ...
080:             sqlcmd = "update Customer set phone='" + phone +"' where id='" +
oid + 081:
                                "'";
082:             ...
083:     }
084:
085:     public void delete();
086:     public void select();
087: };
```

รูปที่ 1.5 แสดงโปรแกรมสำหรับแผนภาพคลาสตัวอย่างรูปที่ 1.4

จากตัวอย่างจะแสดงให้เห็นปัญหาในการเขียนโปรแกรมประยุกต์เชิงวัตถุกับฐานข้อมูลเชิงสัมพันธ์ต่างๆ ดังนี้

1. ปัญหาในการเขียนโปรแกรมและการบำรุงรักษา (Maintenance) เพื่อเข้าถึงคุณลักษณะของวัตถุ

ในการออกแบบและเขียนโปรแกรมเพื่อการเข้าถึงคุณลักษณะของคลาสโดยทั่วไปหนึ่งคุณลักษณะมักจะมีสองเมท็อดคือการกำหนดค่า (Set) และการขอค่า (Get) ซึ่งในการออกแบบเมท็อดสำหรับแต่ละคุณลักษณะนั้นจะมีปริมาณมากกล่าวคือเป็น 2 เท่าของคุณลักษณะที่ต้องการให้เข้าถึงจากภายนอกเป็นอย่างน้อย และต้องคำนึงถึงชื่อของเมท็อดที่สอดคล้องกับชื่อคุณลักษณะและชนิดของคุณลักษณะที่ส่งคืน (return) จากเมท็อดและอาร์กิวเมนต์ (argument) มาในเมท็อดจากภายนอกด้วย จากรูปที่ 1.6 จะพบว่ามีเมท็อดสำหรับเข้าถึงคุณลักษณะมีเป็น 2 เท่าคือ 4 จาก

คุณลักษณะ 2 คุณลักษณะคือคุณลักษณะที่อยู่และคุณลักษณะเบอร์โทรศัพท์ ในการออกแบบควรให้เมท็อดมีชื่อที่สอดคล้องคือกำหนดค่าที่อยู่ (setAddress), ขอค่าที่อยู่ (getAddress) สำหรับคุณลักษณะที่อยู่ และกำหนดค่าเบอร์โทรศัพท์ (setPhone) ขอค่าเบอร์โทรศัพท์ (getPhone) สำหรับคุณลักษณะเบอร์โทรศัพท์ และชนิดข้อมูลของคุณลักษณะ เช่น สายอักขระ (String) ดังนั้นอาร์กิวเมนต์จึงเป็นสายอักขระและค่าที่ส่งคืนจึงเป็นสายอักขระเช่นกัน เป็นต้น ดังนั้นเมื่อความต้องการของโปรแกรมประยุกต์มีการเปลี่ยนในการเพิ่มหรือลดจำนวนคุณลักษณะส่งผลให้จำเป็นต้องมีการเพิ่มหรือลดเมท็อดสำหรับเข้าถึงคุณลักษณะเหล่านั้นด้วยซึ่งทำให้มีต้นทุนเพิ่มขึ้นในการบำรุงรักษาระบบ และบางกรณีอาจมีผลกระทบกับโครงสร้างเขตข้อมูลในตารางความสัมพันธ์ของวัตถุนั้นในฐานข้อมูลเชิงสัมพันธ์ด้วย

```
public class Customer extends Person implements Persistence
{
        private String address;
        private String phone;

        public void setAddress(String newaddr);
        public String getAddress();
        public void setPhone(String newphone);
        public String getPhone();
        …
}
```

รูปที่ 1.6 แสดงเมท็อดกำหนดค่าและขอค่าสำหรับคุณลักษณะของคลาสลูกค้า

2. ปัญหาในการปรับปรุงคุณลักษณะวัตถุในฐานข้อมูลเชิงสัมพันธ์โดยเจาะจงคุณลักษณะ

จากรูปที่ 1.5 บรรทัดที่ 60-80 จะพบว่าโปรแกรมเมอร์เขียนคำสั่งเอสคิวแอลฝังในโปรแกรมให้ทำงานในขณะโปรแกรมทำงานนั้นจะไม่สามารถเปลี่ยนแปลงคำสั่งเอสคิวแอลที่ฝังอยู่ได้ จากรูปเมท็อดปรับปรุง (Update method) จะทำหน้าที่ในการปรับปรุงทุกคุณลักษณะของวัตถุในตารางความสัมพันธ์ของวัตถุ ปัญหาที่พบ หากต้องการที่จะปรับปรุงเพียงบางคุณลักษณะของวัตถุจะต้องออกแบบและเขียนโปรแกรมของเมท็อดเพื่อปรับปรุงเฉพาะคุณลักษณะที่ต้องการ เช่น ปรับปรุงเฉพาะที่อยู่ (update_address), ปรับปรุงเฉพาะเบอร์โทรศัพท์ (update_phone) เป็นต้น

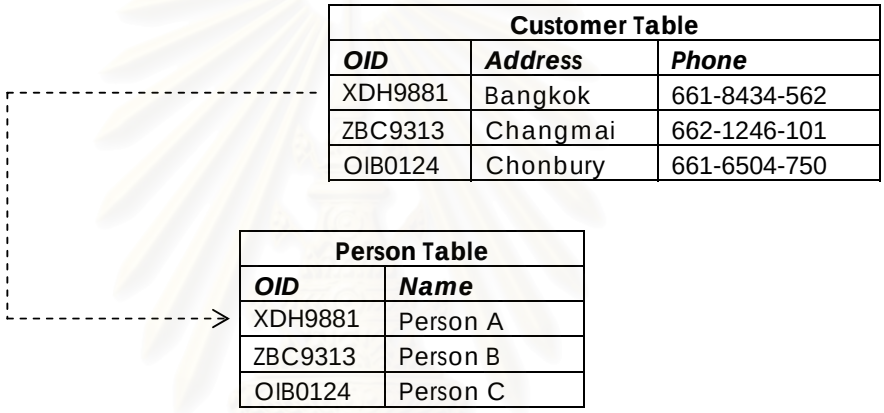
3. ปัญหาการปรับปรุงคุณลักษณะคลาสที่การรับทอดคุณสมบัติมาจากคลาสอื่น

จากตัวอย่างคลาสลูกค้าการรับทอดมาจากคลาสบุคคล จะพบว่าคุณลักษณะชื่อในคลาสบุคคล และคุณลักษณะที่อยู่ในคลาสลูกค้า คลาสทั้งสองถูกเก็บในตารางลูกค้าและตารางบุคคลซึ่งอ้างถึงกันในฐานข้อมูลเชิงสัมพันธ์ดังแสดงในรูปที่ 1.7 ดังนั้นการปรับปรุงใดๆ กับวัตถุลูกค้าต้องพิจารณาว่าต้องปรับปรุงกับตารางบุคคลหรือไม่ เช่นคุณลักษณะชื่อของวัตถุลูกค้าเปลี่ยน

ต้องเขียนโปรแกรมให้ปรับปรุงที่ตารางบุคคลที่อ้างถึงโดยวัตถุลูกค้านั้น แทนการเขียนโปรแกรมเพื่อปรับปรุงตารางลูกค้า เป็นต้น และยังพบปัญหาลักษณะเดียวกันกับเมท็อดอื่น คือการเพิ่มวัตถุลูกค้าในตารางลูกค้าต้องเพิ่มในตารางบุคคลด้วย การลบวัตถุลูกค้าก็เช่นเดียวกัน

4. ปัญหาการนำวัตถุที่จัดเก็บภายในฐานข้อมูลเชิงสัมพันธ์มาใช้งานภายหลัง

การจัดเก็บคุณลักษณะของวัตถุในฐานข้อมูลเชิงสัมพันธ์เป็นดังรูปที่ 1.7 โดยแถวนอนแต่ละแถวคือการเก็บคุณลักษณะวัตถุของแต่ละวัตถุ โดยแถวตั้งแต่ละแถวคือการเก็บคุณลักษณะหนึ่งๆ ของวัตถุ



Customer Table

OID	***Address***	***Phone***
XDH9881	Bangkok	661-8434-562
ZBC9313	Changmai	662-1246-101
OIB0124	Chonbury	661-6504-750

Person Table

OID	***Name***
XDH9881	Person A
ZBC9313	Person B
OIB0124	Person C

รูปที่ 1.7 แสดงลักษณะโครงสร้างการจัดเก็บวัตถุที่การรับทอดคุณสมบัติในฐานข้อมูลเชิงสัมพันธ์

ดังนั้นคุณลักษณะของวัตถุหนึ่งวัตถุจะเก็บในหนึ่งแถวในตารางความสัมพันธ์ หรืออีกนัยหนึ่งตารางความสัมพันธ์ในฐานข้อมูลเชิงสัมพันธ์เก็บวัตถุของคลาสชนิดเดียวกันไว้ ดังนั้นเมื่อต้องการเรียกใช้วัตถุในตารางความสัมพันธ์ จะทำการสร้างวัตถุชนิดนั้นแล้วนำคุณลักษณะที่เก็บในฐานข้อมูลเชิงสัมพันธ์ของวัตถุที่ต้องการใช้มากำหนดค่าเริ่มให้วัตถุที่สร้างใหม่เสมือนเรียกใช้วัตถุจากฐานข้อมูลเชิงวัตถุ

5. ปัญหาความซับซ้อนและทำซ้ำเสมอของส่วนโปรแกรมสำหรับติดต่อกับฐานข้อมูลเชิงสัมพันธ์

โปรแกรมประยุกต์ฐานข้อมูลเชิงสัมพันธ์ต้องมีส่วนในการติดต่อกับฐานข้อมูลเชิงสัมพันธ์ ซึ่งส่วนดังกล่าวมักเป็นส่วนที่ซับซ้อนและทำซ้ำเสมอ กล่าวคือ เมื่อต้องการเพิ่มข้อมูล การปรับปรุงข้อมูล การลบข้อมูล และการเลือกข้อมูล สิ่งที่ต้องทำคือการสร้างการเชื่อมต่อ (Connection) เพื่อทำการติดต่อกับฐานข้อมูลเชิงสัมพันธ์ก่อนเสมอ โปรแกรมเมอร์อาจจำเป็นต้อง

เสียเวลาในการเรียนรู้การติดต่อกับฐานข้อมูลเชิงสัมพันธ์และสำหรับโปรแกรมที่มีประสบการณ์มักจะต้องเขียนโปรแกรมส่วนดังกล่าวเสมอ

ดังนั้นผู้วิจัยจึงมีความสนใจในการศึกษาวิจัยและออกแบบพัฒนาโครงร่างฯ ดังกล่าวเพื่อให้สามารถแก้ไขปัญหาดังนี้

1. โครงร่างฯสามารถแก้ไขปัญหาเมท็อดสำหรับการกำหนดค่าและการขอค่าคุณลักษณะของวัตถุมีจำนวนมาก จึงออกแบบโครงร่างฯ ให้มีเพียงเมท็อดเดียวสำหรับรับค่าจากผู้ส่งข้อความว่าต้องการกำหนดค่าและมีเพียงเมท็อดเดียวสำหรับขอค่าจากคุณลักษณะใด
2. โครงร่างฯสามารถแก้ไขปัญหาจำนวนเมท็อดสำหรับการเพิ่ม การปรับปรุง ในแต่ละคุณลักษณะของวัตถุที่มีมาก จึงออกแบบโครงร่างฯ ให้มีเพียงเมท็อดเดียวรับค่าจากผู้ส่งข้อความว่าต้องการเพิ่ม ปรับปรุง คุณลักษณะใด
3. โครงร่างฯสามารถพิจารณาสำหรับเรียกเมท็อดของการรับทอดคลาสคุณลักษณะกัน เพื่อแก้ปัญหาการเรียกเมท็อดของคลาสโดยนักพัฒนาโปรแกรมประยุกต์
4. โครงร่างฯสามารถแก้ปัญหาในการนำวัตถุที่จัดเก็บภายในฐานข้อมูลเชิงสัมพันธ์มาใช้งานภายหลัง
5. โครงร่างฯทำการสร้างการและยกเลิกการเชื่อมต่อไปฐานข้อมูลเชิงสัมพันธ์แทนโปรแกรมเมอร์

1.2 วัตถุประสงค์ของการวิจัย

เพื่อศึกษาวิจัยและออกแบบพัฒนาโครงร่างโปรแกรมประยุกต์เชิงวัตถุสำหรับพัฒนาโปรแกรมประยุกต์ฐานข้อมูลเชิงสัมพันธ์

1.3 ขอบเขตของการวิจัย

1. ศึกษาวิจัยและออกแบบพัฒนาโครงร่างโปรแกรมประยุกต์เชิงวัตถุสำหรับระบบงานประยุกต์ฐานข้อมูลเชิงสัมพันธ์เพื่อให้
   - โครงร่างฯสามารถแก้ไขปัญหาจำนวนเมท็อดสำหรับการกำหนดค่าและการขอค่าคุณลักษณะของวัตถุมีจำนวนมาก จึงออกแบบโครงร่างฯให้มีเพียงเมท็อดเดียวสำหรับรับค่าและกำหนดค่าจากคุณลักษณะใด

- โครงร่างฯสามารถแก้ไขปัญหาจำนวนเมท็อดสำหรับการเพิ่ม การปรับปรุง ในแต่ละคุณลักษณะของวัตถุมีจำนวนมาก จึงออกแบบโครงร่างฯ ให้มีเพียงเมท็อดเดียวสำหรับการเพิ่ม การปรับปรุง คุณลักษณะใด
- โครงร่างฯสามารถพิจารณาเรียกเมท็อดของคลาสที่มีความสัมพันธ์แบบการรับทอดคลาสว่าควรเรียกที่คลาสใด เช่น การปรับปรุงชื่อซึ่งเป็นคุณลักษณะของคลาสบุคคล แต่การเรียกเมท็อดดังกล่าวถึงเรียกผ่านคลาสลูกค้า โครงร่างฯ ทราบว่าต้องไปเรียกเมท็อดปรับปรุงชื่อที่คลาสบุคคล
- โครงร่างฯสามารถแก้ปัญหาในการสร้างหรือขอวัตถุจากฐานข้อมูลเชิงสัมพันธ์ เพื่อให้นักพัฒนาโปรแกรมประยุกต์เรียกใช้เพื่อกระทำกับคุณลักษณะของวัตถุในฐานข้อมูลเชิงสัมพันธ์
- โครงร่างฯทำการสร้างการและยกเลิกการเชื่อมต่อไปฐานข้อมูลเชิงสัมพันธ์แทนโปรแกรมเมอร์

2. ทำการสร้างโครงร่างฯจากโครงร่างฯที่ได้ออกแบบไว้
3. ใช้งานโครงร่างฯสำหรับระบบงานประยุกต์ฐานข้อมูลเชิงสัมพันธ์
4. การศึกษาวิจัยโครงร่างฯ จะใช้ภาษาเชิงวัตถุ C++ ภายใต้ระบบปฏิบัติการไมโครซอฟท์วินโดว์ ผ่านตัวกลางในการเชื่อมต่อฐานข้อมูลเชิงสัมพันธ์ โอดีบีซี (Open Database Connectivity :ODBC) ไปยังฐานข้อมูลเชิงสัมพันธ์มายเอสคิวแอล (MySQL)

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1. โครงร่างฯ สามารถนำไปใช้ประโยชน์ในการพัฒนาระบบงานคอมพิวเตอร์ด้วยเทคโนโลยีเชิงวัตถุร่วมกับฐานข้อมูลเชิงสัมพันธ์
2. เป็นแนวทางในการปรับปรุงโครงร่างฯเพื่อทำงานร่วมกับฐานข้อมูลในด้านอื่น
3. เป็นแนวทางในการศึกษาวิจัยและออกแบบและพัฒนาโครงร่างฯ ด้านอื่น
4. โครงร่างฯ ทำให้สิ่งที่ต้องทำอยู่เสมอและซับซ้อนในการพัฒนาโปรแกรมประยุกต์ฐานข้อมูลเชิงสัมพันธ์ลดลง

1.5 ขั้นตอนการวิจัย

1. รวบรวมและศึกษาแนวทางการติดต่อกับฐานข้อมูลเชิงสัมพันธ์ในเชิงการนำไปใช้งานจริง เช่น การเชื่อมต่อฐานข้อมูลเชิงสัมพันธ์ เป็นต้น
2. รวบรวมและศึกษาแนวทางในการจัดเก็บวัตถุลงในฐานข้อมูลเชิงสัมพันธ์ เช่นการกระทำกับคุณลักษณะของการรับทอดคลาส
3. ทำการออกแบบแผนภาพคลาสเบื้องต้น สำหรับการปฏิสัมพันธ์กับฐานข้อมูลเชิงสัมพันธ์
4. ทำการออกแบบโครงร่างฯจากแผนภาพคลาสเบื้องต้นและออกแบบส่วนเมท็อดในการปฏิสัมพันธ์กับฐานข้อมูลเชิงสัมพันธ์
5. สร้างกรณีตัวอย่างและนำโครงร่างฯไปใช้งาน
6. ปรับปรุงโครงร่างฯให้ดีขึ้น
7. วิเคราะห์และสรุปผล
8. จัดทำรายงานวิทยานิพจน์

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

บทที่ 2

ทฤษฎี และงานวิจัยที่เกี่ยวข้อง

2.1 แนวคิด และทฤษฎี

2.1.1. การทำงานหลักของระบบจัดการฐานข้อมูล

ระบบจัดการฐานข้อมูลเชิงสัมพันธ์ คือระบบที่พัฒนาเพื่อการจัดเก็บ เรียกใช้ค้นหาข้อมูล ให้เป็นไปอย่างมีประสิทธิภาพ ระบบงานคอมพิวเตอร์ต่างๆในปัจจุบันมักจะมีการทำงานร่วมกับระบบจัดการฐานข้อมูลเชิงสัมพันธ์ ประโยชน์ในการใช้งานระบบจัดการฐานข้อมูล [2] คือ

- ความไม่ขึ้นกับข้อมูล (Data Independence)
- ระบบงานควรเป็นอิสระเท่าที่เป็นไปได้จากรายละเอียดของการนำเสนอข้อมูลและการจัดเก็บ
- ประสิทธิภาพในการเข้าถึงข้อมูล (Efficient data access)
- ระบบจัดการฐานข้อมูลใช้เทคนิคที่เป็นประโยชน์สูงสุดในการจัดเก็บและเรียกใช้ข้อมูลอย่างมีประสิทธิภาพ
- ความบูรณภาพและความปลอดภัยของข้อมูล (Data integrity and security)
- ระบบจัดการฐานข้อมูลสามารถจัดการข้อบังคับความบูรณภาพบนข้อมูลได้ และจัดการการเข้าถึงเพื่อความปลอดภัยได้
- การดูแลข้อมูล (Data administration)
- การดูแลข้อมูลจากเครื่องบริการ (Server) สามารถทำได้สะดวก ซึ่งดูแลให้มีความซ้ำซ้อนน้อยที่สุด และปรับแต่งการจัดเก็บข้อมูลเพื่อการเรียกใช้ที่มีประสิทธิภาพ
- การเข้าถึงข้อมูลจวบกันและการกู้การชน (Concurrent access and crash recovery)
- ระบบจัดการฐานข้อมูลจัดการการเข้าถึงข้อมูลพร้อมกันโดยผู้ใช้คิดเหมือนใช้ข้อมูลเพียงผู้เดียว
- เวลาในการพัฒนางานประยุกต์ลดลง (Reduced application development time)
- ระบบจัดการฐานข้อมูลรองรับหน้าที่การทำงานที่สำคัญเพื่อให้นักพัฒนาระบบเรียกใช้งาน

2.1.2. การทำนอร์มอลไรเซชัน (Normalization)

เป็นวิธีกำจัดความซ้ำซ้อนของข้อมูล (Redundancy) จากการออกแบบฐานข้อมูลเบื้องต้นเพื่อกำจัดความซ้ำซ้อนของข้อมูล เพื่อลดปัญหาดังนี้ [3]

- ความเป็นบูรณภาพของข้อมูล (Data integrity) ในฐานข้อมูลเชิงสัมพันธ์
- ความผิดปกติในการปรับปรุง (Update anomaly)
- ความผิดปกติในการลบ (Delete anomaly)
- ความผิดปกติในการเพิ่ม (Insertion anomaly)

แนวทางในการทำนอร์มอลไรเซชันคือการแตกหรือรวมความสัมพันธ์ เพื่อให้ความสัมพันธ์ใดๆ มีเฉพาะข้อมูลที่เกี่ยวข้องหรือสัมพันธ์กันในความสัมพันธ์นั้นเท่านั้น

2.1.3. การเปลี่ยนวัตถุของคลาสเพื่อเก็บในฐานข้อมูลเชิงสัมพันธ์ [4]

เป็นวิธีการเปลี่ยนแผนภาพคลาสเป็นแผนภาพความสัมพันธ์ โดยมีการลักษณะการเปลี่ยนดังนี้ พิจารณาการสืบทอดของคลาสเพื่อนำไปจัดเก็บในฐานข้อมูลเชิงสัมพันธ์ด้วยวิธีการเปลี่ยนคลาสไปยังตารางความสัมพันธ์ โดยแต่ละวิธีมีข้อดีข้อเสียดังตารางเปรียบเทียบที่ 2.1 [4]

- ใช้หนึ่งตารางความสัมพันธ์ต่อหนึ่งสายลำดับชั้นการการรับทอด
  หมายถึงการสร้างตารางความสัมพันธ์สำหรับคลาสลำดับชั้นสุดท้าย เพื่อนำคุณลักษณะของวัตถุที่สืบทอดของทุกคลาสในสายการสืบทอดมาอยู่ร่วมในตารางความสัมพันธ์เดียว เช่น คลาสลูกค้าสืบทอดจากคลาสบุคคล ทำการสร้างตารางความสัมพันธ์ลูกค้า นำคุณลักษณะของคลาสบุคคลและคลาสลูกค้ามาสร้างเป็นเขตข้อมูลในตารางความสัมพันธ์ลูกค้า เป็นต้น
- ใช้หนึ่งตารางความสัมพันธ์ต่อหนึ่งคลาสรูปธรรม (Concrete class)
  หมายถึงคลาสรูปธรรมที่สืบทอดจากคลาสนามธรรม ให้เปลี่ยนเป็นตารางความสัมพันธ์และนำคุณลักษณะของคลาสนามธรรมมารวมอยู่ในตารางความสัมพันธ์ด้วย
- ใช้หนึ่งตารางความสัมพันธ์ต่อหนึ่งคลาส
  หมายถึงคลาสทั้งหมดในแผนภาพคลาสแต่ละคลาสเปลี่ยนเป็นตารางความสัมพันธ์

ตารางที่ 2.1 เปรียบเทียบข้อดีข้อเสียในการเลือกวิธีการเปลี่ยนแผนภาพคลาสเป็นตารางความสัมพันธ์

ปัจจัยเพื่อพิจารณา	หนึ่งตารางต่อสายลำดับชั้น (Hierarchy)	หนึ่งตารางต่อคลาสรูปธรรม	หนึ่งตารางต่อคลาส
การทำรายงานแบบด่วน	ง่าย	ปานกลาง	ปานกลาง/ง่าย
ความสะดวกการทำให้ใช้งานได้จริง	ง่าย	ปานกลาง	ยาก
ความสะดวกการเข้าถึงข้อมูล	ง่าย	ง่าย	ปานกลาง/ง่าย
การคอปพ์ปิง (Coupling)	สูงมาก	สูง	ต่ำ
ความเร็วการเข้าถึงข้อมูล	เร็ว	เร็ว	ปานกลาง/เร็ว
สนับสนุนโพลีมอร์พิเซม (Polymorphism)	ปานกลาง	ต่ำ	สูง

2.1.4. โครงร่างโปรแกรมประยุกต์เชิงวัตถุ [5][6][7]

โครงร่างโปรแกรมประยุกต์เชิงวัตถุคือการนำกลับมาใช้ใหม่ของการออกแบบและชุดคำสั่งของระบบงานประยุกต์ใดๆภายใต้ขอบเขตของความสนใจ การนำกลับมาใช้ใหม่ระดับสถาปัตยกรรมเป็นการนำกลับมาใช้ใหม่ในระดับที่สูงกว่าการนำกลับมาใช้ใหม่ในระดับชุดคำสั่งเพียงอย่างเดียว การใช้โครงร่างฯ จึงช่วยลดเวลาในการพัฒนาโปรแกรมประยุกต์ เช่น การพัฒนาโปรแกรมประยุกต์สมุดโทรศัพท์โดยจัดเก็บข้อมูลในฐานข้อมูลเชิงสัมพันธ์โดยใช้เทคโนโลยีเชิงวัตถุ นักพัฒนาสามารถเลือกใช้โครงร่างฯสำหรับโปรแกรมประยุกต์ฐานข้อมูลเชิงสัมพันธ์ซึ่งจัดเตรียมสถาปัตยกรรมหรือการออกแบบและกำหนดแนวทางหลักให้นักพัฒนาใช้งานจึงลดเวลาในการออกแบบใหม่ในส่วนที่เหมือนกันหรือร่วมกันในทุกระบบโปรแกรมประยุกต์ฐานข้อมูลเชิงสัมพันธ์ โครงร่างฯได้จัดเตรียมชุดคำสั่งซึ่งซ่อนอยู่ภายใต้โครงร่างฯเพื่อให้คลาสต่างๆทำงานร่วมกันอย่างเป็นระบบ โดยนักพัฒนาเขียนชุดคำสั่งเท่าที่จำเป็นแก่โครงร่างฯเท่านั้นทำให้นักพัฒนาไม่ต้องเขียนคำสั่งและออกแบบใหม่ทั้งหมด จึงสามารถลดเวลาในการพัฒนาระบบโดยรวมได้ และทำให้โปรแกรมประยุกต์ที่พัฒนาโดยโครงร่างฯเดียวกันมีคุณภาพเดียวกันหรือใกล้เคียงกัน ดังนั้นการเลือกโครงร่างฯที่มีคุณภาพทางซอฟต์แวร์ย่อมทำให้ระบบงานมีคุณภาพด้วยเช่นกัน และการใช้งานโครงร่างฯสามารถใช้โครงร่างฯมากกว่าหนึ่งโครงร่างฯหรือใช้เพียง

บางส่วนในโครงร่างฯได้ ประโยชน์การใช้โครงร่างฯในการพัฒนาโปรแกรมประยุกต์ใดๆ คือ ความสามารถสภาพเป็นส่วนจำเพาะ (Modularity) ความสามารถในการนำกลับมาใช้ใหม่ (Reusability) ความสามารถในการเพิ่มขยายได้ (Extensibility) การควบคุมการผกผัน (Inversion of control)

โครงร่างฯที่ออกแบบจะประกอบด้วยสปอท (Spot) คือ จุดหรือเมท็อดซึ่งโครงร่างฯ กำหนดให้ผู้ใช้โครงร่างฯ สามารถปรับเปลี่ยนให้เหมาะกับการใช้งานเพื่อให้โครงร่างฯทำงานตามที่ปรับเปลี่ยน ประเภทของจุดหรือเมท็อดประกอบด้วย คอร์สปอท (Core-Sport) จุดแกนหรือเมท็อดที่ปรับเปลี่ยนไม่ได้ ซึ่งทุกโปรแกรมประยุกต์ที่ใช้โครงร่างฯ ใช้ร่วมกัน และจุดหรือเมท็อดที่ปรับเปลี่ยนได้ (Hot-spots) คือจุดหรือเมท็อดซึ่งสามารถปรับเปลี่ยนได้ เพื่อให้โครงร่างฯทำงานตามความต้องการของผู้ใช้โครงร่างฯ ซึ่งจุดหรือเมท็อดที่ปรับเปลี่ยนได้สามารถแบ่งได้ 2 ประเภท คือ จุดหรือเมท็อดปรับเปลี่ยนประเภทกล่องขาว (Whitebox Hot-Spot) ซึ่งเป็นลักษณะการใช้งานโครงร่างฯด้วยการปรับและทำให้ใช้งานจริงในบางเมท็อดและคลาสของโครงร่างฯเพื่อนำไปใช้งานในโปรแกรมประยุกต์ และจุดหรือเมท็อดปรับเปลี่ยนประเภทกล่องดำ (Blackbox Hot-Spot) ซึ่งลักษณะการใช้งานจะนำไปใช้เป็นส่วนประกอบของโปรแกรมประยุกต์

2.2 ผลิตภัณฑ์ซอฟท์แวร์และงานวิจัยที่เกี่ยวข้อง

2.2.1. เจดีโอ (Java Data Objects: JDO) [8]

เจดีโอถูกสร้างโดยบริษัท ซัน เป็น เอพีไอ (Application Programming Interface :API) ซึ่งเป็นคลังโปรแกรม (Program library) สำหรับเชื่อมต่อมาตรฐานบนพื้นฐานแบบจำลองนามธรรมจาวา (Java model abstraction) ซึ่งถูกพัฒนาสำหรับภาษาจาวา การใช้เจดีโอมีประโยชน์คือ

- ความสามารถในการทำงานบนหลายสภาพแวดล้อมโดยไม่ต้องแปลรหัสคำสั่งเป็นภาษาเครื่องใหม่หรือเปลี่ยนแปลงรหัส (Portability)
- สามารถใช้งานโดยไม่ขึ้นกับชนิดฐานข้อมูล (Database independence)
- ใช้งานง่ายโดยโปรแกรมเมอร์สนใจเพียงแบบจำลองวัตถุเท่านั้นส่วนเจดีโอจะทำให้วัตถุถาวรให้
- มีประสิทธิภาพสูง (High performance)
- สามารถใช้ร่วมกับอีเจบี (Enterprise Java Bean :EJB) ได้

โดยเจดีโอเป็นคลาสไลบารี่สำหรับการจัดเก็บคลาส จึงมีความเกี่ยวข้องกับโครงร่างฯ โดยเป็นวิธีหนึ่งสำหรับการจัดเก็บวัตถุ

2.2.2. เจมส์สโทนเอสแอปพิเคชันเซอร์เวอร์ (GemStone/S Application Server) [9]

เป็นซอฟต์แวร์ให้บริการโปรแกรมประยุกต์เชิงวัตถุ (Object Application Server) ซึ่งทำหน้าที่เป็นสะพาน (Bridges) ระหว่างโปรแกรมประยุกต์เชิงวัตถุและฐานข้อมูลเชิงสัมพันธ์ สำหรับสร้างโปรแกรมประยุกต์แบบ 3 ระดับ (3-Tier) หรือ สำหรับสถาปัตยกรรมแบบเครื่องขอบริการกับเครื่องให้บริการ (client-server architecture) ที่มีการทำงานร่วมกันแบบกว้าง (Enterprise wide)

2.2.3. วิชาลบีเอสเอฟ (Visual BSF ™) [10]

เป็นโครงร่างวัตถุ-เชิงสัมพันธ์ของภาษาจาวาที่ทำให้นักพัฒนาโปรแกรมประยุกต์ง่ายในการจัดเก็บวัตถุถาวรด้วยภาษาจาวาไปยังฐานข้อมูลเชิงสัมพันธ์

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

บทที่ 3

การวิเคราะห์โครงร่างโปรแกรมประยุกต์เชิงวัตถุ สำหรับพัฒนาโปรแกรมประยุกต์ฐานข้อมูลเชิงสัมพันธ์

บทนี้กล่าวถึงการวิเคราะห์โครงร่างโปรแกรมประยุกต์เชิงวัตถุสำหรับพัฒนาโปรแกรมประยุกต์ฐานข้อมูลเชิงสัมพันธ์

3.1. ขั้นตอนการวิเคราะห์โครงร่างฯ

1. วิเคราะห์แผนภาพตารางความสัมพันธ์ซึ่งเหมาะสมตามหลักการออกแบบฐานข้อมูลเชิงสัมพันธ์และสามารถนำไปใช้กับโครงร่างฯได้
2. วิเคราะห์ลักษณะแผนภาพคลาสที่เหมาะสมกับโครงร่างฯ โดยพิจารณาว่าแผนภาพคลาสลักษณะใดซึ่งมีความสอดคล้องกับหลักการออกแบบฐานข้อมูล เช่น แผนภาพคลาสควรมีลักษณะใดและไม่ควรมีลักษณะใดก่อนการนำมาแปลงเพื่อใช้งานในโครงร่างฯ เป็นต้น
3. วิเคราะห์วิธีการแปลงแผนภาพคลาสไปเป็นแผนภาพตารางความสัมพันธ์ที่เหมาะสมกับโครงร่างฯ โดยพิจารณาจากแผนภาพตารางความสัมพันธ์ที่ได้จากการแปลงว่ามีความสอดคล้องกับหลักการออกแบบฐานข้อมูลเพียงใด
4. วิเคราะห์ลักษณะความสัมพันธ์ระหว่างโปรแกรมประยุกต์เชิงวัตถุกับฐานข้อมูลเชิงสัมพันธ์
5. วิเคราะห์ลักษณะความสัมพันธ์ระหว่างโปรแกรมประยุกต์เชิงวัตถุกับฐานข้อมูลเชิงสัมพันธ์และลักษณะความสัมพันธ์ของคลาส

3.2. การวิเคราะห์ปัญหาการออกแบบโครงร่างฯ

ในการนำโครงร่างฯไปใช้งาน จุดประสงค์เพื่อให้นักพัฒนาโปรแกรมประยุกต์เชิงวัตถุ มีความรู้สึกเสมือนว่าสามารถจัดเก็บวัตถุลงในฐานข้อมูลเชิงสัมพันธ์ภายใต้ลักษณะความสัมพันธ์ของคลาส โดยโครงร่างฯ จะทำงานในการปฏิสัมพันธ์กับฐานข้อมูลเชิงสัมพันธ์ในลักษณะต่างๆ แทนผู้ใช้โครงร่างฯ ปัญหาที่พบในการออกแบบโครงร่างฯ นั้นมีดังนี้

1. การจัดเก็บวัตถุในฐานข้อมูลเชิงสัมพันธ์ไม่สามารถจะเก็บเมท็อดของวัตถุได้ เนื่องจากฐานข้อมูลเชิงสัมพันธ์สามารถจัดเก็บได้เพียงข้อมูลหรือคุณลักษณะของวัตถุเท่านั้น ทำอย่างไรให้นักพัฒนาโปรแกรมประยุกต์เชิงวัตถุมีความรู้สึกเหมือนจัดเก็บวัตถุได้จริง
2. การเพิ่มวัตถุในฐานข้อมูลเชิงสัมพันธ์ต้องคำนึงถึงความสัมพันธ์ของคลาส ดังนี้
   - ประเภทซิงเกิลคลาส (Single class) จะจัดเก็บวัตถุใหม่ลงในตารางความสัมพันธ์เดียว
   - ประเภทการรับทอดคลาส (Inheritance class) การจัดเก็บวัตถุใหม่ลงฐานข้อมูลเชิงสัมพันธ์นั้นจะมีความซับซ้อนมาก เนื่องจากต้องนำคุณลักษณะของแต่ละวัตถุจัดเก็บลงในแต่ละตารางความสัมพันธ์ให้ถูกต้อง เช่น คลาสลูกค้าสืบทอดจากคลาสบุคคล เมื่อนักพัฒนาต้องการจัดเก็บวัตถุลูกค้า โครงร่างฯ ต้องทำการจัดเก็บลงในตารางความสัมพันธ์บุคคลและตารางความสัมพันธ์ลูกค้า
   - ประเภทภาพรวมกลุ่มคลาส (Aggregation class) ในการจัดเก็บนั้นไม่มีความจำเป็นต้องจัดเก็บวัตถุที่รวมกลุ่มกันจะจัดเก็บเฉพาะวัตถุตนเอง เนื่องจากคลาสที่รวมกลุ่มอยู่ด้วยจะถูกสร้างและจัดเก็บก่อนการรวมกลุ่ม ดังนั้นในการจัดเก็บวัตถุต้องคำนึงถึงว่าคลาสที่รวมกลุ่มนั้นถูกสร้างและจัดเก็บไว้แล้วหรือไม่
   - ประเภทคอมโพสิตชันคลาส (Composition class) การจัดเก็บวัตถุ วัตถุต้องทำการสร้างและจัดเก็บวัตถุคอมโพสิตนั้นด้วยเสมอ และต้องทำการจัดเก็บวัตถุคอมโพสิตก่อนและจึงจัดเก็บวัตถุตนเอง

การออกแบบโครงร่างฯให้นักพัฒนาโปรแกรมประยุกต์เชิงวัตถุมีความรู้สึกเหมือนการจัดเก็บวัตถุใหม่เพียงอย่างเดียวไม่ต้องสนใจว่าคลาสมีความสัมพันธ์อย่างไร

3. การปรับปรุงวัตถุในฐานข้อมูลเชิงสัมพันธ์นั้นหมายถึงการปรับปรุงคุณลักษณะของวัตถุ ซึ่งต้องคำนึงถึงความสัมพันธ์ของคลาส ดังนี้
   - ประเภทซิงเกิลคลาสจะปรับปรุงที่ตารางความสัมพันธ์ของวัตถุเพียงตารางเดียว แต่ในการปรับปรุงจำเป็นต้องทราบตำแหน่งที่วัตถุต้องการปรับปรุงด้วย
   - ประเภทการรับทอดคลาสซึ่งคลาสตนเองอาจมีความสัมพันธ์แบบซิงเกิลคลาส หรือภาพรวมกลุ่มคลาสหรือคอมโพสิตชันคลาส ในการปรับปรุงวัตถุในฐานข้อมูลเชิงสัมพันธ์นั้นมีความซับซ้อนมาก เนื่องจากต้องพิจารณาวัตถุนั้นมีการสืบทอดอย่างไร และการทำการปรับปรุงวัตถุต้องพิจารณาว่าต้องทำการปรับปรุงที่ตารางความสัมพันธ์ใดบ้าง เช่น คลาสลูกค้าสืบทอดจากคลาสบุคคล เมื่อ

นักพัฒนาโปรแกรมประยุกต์เชิงวัตถุต้องการปรับปรุงวัตถุลูกค้า โครงร่างฯ ต้องทำการปรับปรุงตารางความสัมพันธ์บุคคล และตารางความสัมพันธ์ลูกค้า

- ประเภทภาพรวมกลุ่มคลาสซึ่งคลาสตนเองอาจมีความสัมพันธ์แบบซิงเกิลคลาสหรือการรับทอดคลาสหรือคอมโพสิตชันคลาส ในการปรับปรุงจะทำการปรับปรุงในตารางความสัมพันธ์ของวัตถุ ส่วนวัตถุที่รวมกลุ่มอยู่จะเป็นเพียงคุณลักษณะหนึ่งของวัตถุตนเองซึ่งในการจัดเก็บในตารางความสัมพันธ์จะจัดเก็บเพียงหมายเลขวัตถุของวัตถุที่รวมกลุ่มเท่านั้น เช่น คลาสลูกค้ารวมกลุ่มกับคลาสพนักงานขาย การปรับปรุงคลาสลูกค้าโครงร่างฯทำการปรับปรุงตามชนิดความสัมพันธ์คลาส แต่เมื่อวัตถุลูกค้าต้องการเปลี่ยนวัตถุพนักงานขายเหมือนเมท็อดกำหนดค่าคุณลักษณะทั่วไป
- ประเภทคอมโพสิตชันคลาสซึ่งคลาสตนเองอาจเป็นซิงเกิลคลาสหรือการรับทอดคลาสหรือภาพรวมกลุ่มคลาส ในการปรับปรุงจะทำการปรับปรุงในตารางความสัมพันธ์ของวัตถุ ส่วนวัตถุคอมโพสิตจะเป็นเพียงคุณลักษณะหนึ่งของวัตถุตนเองซึ่งในการจัดเก็บในตารางความสัมพันธ์ซึ่งจัดเก็บเพียงหมายเลขวัตถุของวัตถุคอมโพสิตเท่านั้น เช่น คลาสการสั่งซื้อคอมโพสิตกับคลาสรายการสั่งซื้อ เมื่อมีการปรับปรุงคลาสการสั่งซื้อโครงร่างฯทำการปรับปรุงตามชนิดความสัมพันธ์คลาส แต่หากต้องการปรับปรุงวัตถุรายการการสั่งซื้อ สามารถส่งข้อความให้วัตถุรายการการสั่งซื้อ เพื่อขอทำการปรับปรุง

ซึ่งต้องออกแบบโครงร่างฯให้นักพัฒนาโปรแกรมประยุกต์เชิงวัตถุรู้สึกเหมือนการปรับปรุงวัตถุเพียงอย่างเดียวไม่ต้องสนใจว่าคลาสมีความสัมพันธ์อย่างไร

4. การลบวัตถุในฐานข้อมูลเชิงสัมพันธ์นั้นหมายถึงการลบคุณลักษณะของวัตถุ ซึ่งต้องคำนึงถึงความสัมพันธ์ของคลาส ดังนี้
   - ประเภทซิงเกิลคลาสจะลบที่ตารางความสัมพันธ์ของวัตถุเพียงตารางเดียว แต่ในการลบจำเป็นต้องทราบตำแหน่งที่วัตถุต้องการปรับปรุงด้วย
   - ประเภทการรับทอดคลาสซึ่งคลาสตนเองอาจเป็นซิงเกิลคลาสหรือภาพรวมกลุ่มคลาสหรือคอมโพสิตชันคลาส ในการลบวัตถุในฐานข้อมูลเชิงสัมพันธ์นั้นมีความซับซ้อนมาก เนื่องจากต้องพิจารณาวัตถุนั้นมีการการรับทอดอย่างไร และการทำการลบวัตถุต้องพิจารณาว่าต้องทำการลบที่ตารางความสัมพันธ์ใดบ้าง เช่น คลาสลูกค้าสืบทอดจากคลาสบุคคล เมื่อนักพัฒนาโปรแกรมประยุกต์เชิงวัตถุต้องการลบวัตถุลูกค้า โครงร่างฯ ต้องทำการลบที่ตารางความสัมพันธ์บุคคล และตารางความสัมพันธ์ลูกค้า เป็นต้น

- ประเภทภาพรวมกลุ่มคลาสซึ่งคลาสอาจเป็นซิงเกิลคลาสหรือการรับทอดคลาสหรือคอมโพสิตชันคลาส ในการลบจะทำการลบวัตถุในตารางความสัมพันธ์ของวัตถุเท่านั้น ไม่ควรทำการลบวัตถุที่รวมกลุ่ม
- ประเภทคอมโพสิตชันคลาสซึ่งคลาสอาจเป็นซิงเกิลคลาสหรือการรับทอดคลาสหรือภาพรวมกลุ่มคลาส ในการลบจะทำการลบวัตถุในตารางความสัมพันธ์ของวัตถุและต้องทำการลบวัตถุคอมโพสิตด้วย เช่น คลาสการสั่งซื้อคอมโพสิตกับคลาสรายการสั่งซื้อ เมื่อมีการลบวัตถุการสั่งซื้อจะต้องทำการลบรายการสั่งซื้อที่ประกอบอยู่ในวัตถุการสั่งซื้อนั้น

ซึ่งต้องออกแบบโครงร่างฯให้นักพัฒนาโปรแกรมประยุกต์เชิงวัตถุรู้สึกเหมือนการลบวัตถุเพียงอย่างเดียวไม่ต้องสนใจว่าคลาสมีความสัมพันธ์อย่างไร

5. การเลือกหรือการดึงวัตถุที่จัดเก็บในฐานข้อมูลเชิงสัมพันธ์มาใช้ เนื่องจากการจัดเก็บวัตถุนั้นจัดเก็บเพียงคุณลักษณะเท่านั้น ดังนั้นจะออกแบบอย่างไรให้เสมือนวัตถุถูกนำกลับมาใช้เสมือนว่าเป็นวัตถุเดิมโดยมีคุณลักษณะค่าเดิมและหน้าที่การทำงานเช่นเดิม และด้วยความสัมพันธ์ของคลาสการสร้างวัตถุขึ้นมาใหม่ต้องสร้างจากวัตถุที่มีความสัมพันธ์กันมาเป็นวัตถุใหม่ ดังนี้
   - ประเภทซิงเกิลคลาสจะสามารถเลือกวัตถุได้จากตารางความสัมพันธ์เดียว หากแต่ในการเลือกวัตถุต้องทราบตัวบ่งชี้วัตถุ เช่น หมายเลขวัตถุ, ค่าคุณลักษณะของวัตถุ เป็นต้น เพื่อใช้ในการเลือกวัตถุ
   - ประเภทการรับทอดคลาสซึ่งคลาสตนเองอาจเป็นซิงเกิลคลาสหรือภาพรวมกลุ่มคลาสหรือคอมโพสิตชันคลาส ในการเลือกวัตถุในฐานข้อมูลเชิงสัมพันธ์นั้นมีความซับซ้อนมาก เนื่องจากต้องพิจารณาว่าวัตถุนั้นมีการสืบทอดอย่างไร การเลือกวัตถุต้องพิจารณาว่าต้องเลือกที่ตารางความสัมพันธ์ใดบ้าง เช่น คลาสลูกค้าสืบทอดจากคลาสบุคคล เมื่อนักพัฒนาโปรแกรมประยุกต์เชิงวัตถุต้องการเลือกวัตถุลูกค้า โครงร่างฯ ต้องทำการเลือกวัตถุบุคคลที่ตารางความสัมพันธ์บุคคล และเลือกวัตถุลูกค้าตารางความสัมพันธ์ลูกค้า แล้วนำมารวมกันเป็นวัตถุลูกค้า
   - ประเภทภาพรวมกลุ่มคลาสซึ่งคลาสอาจเป็นซิงเกิลคลาสหรือการรับทอดคลาสหรือคอมโพสิตชันคลาส ในการเลือกจะทำการเลือกวัตถุในตารางความสัมพันธ์ของวัตถุและทำการเลือกวัตถุที่รวมกลุ่มกันมาด้วยเพื่อให้คลาสตนเองสามารถรวมกลุ่มได้ เช่น คลาสลูกค้ารวมกลุ่มคลาสพนักงานขาย เมื่อมีการเลือกคลาสลูกค้าจะทำการเลือกวัตถุพนักงานขายที่รวมกลุ่มด้วย

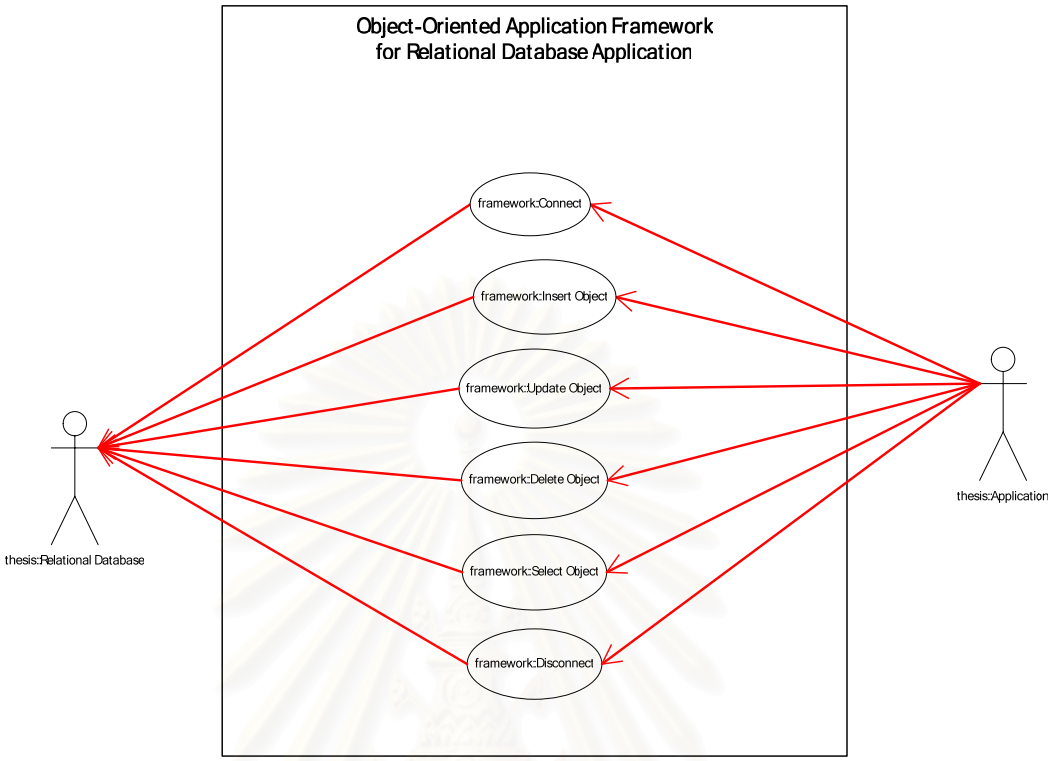
– ประเภทคอมโพสิตชันคลาสอาจเป็นซิงเกิลคลาสหรือการรับทอดคลาสหรือภาพรวมกลุ่มคลาส ในการเลือกวัตถุในตารางความสัมพันธ์ของวัตถุและต้องเลือกวัตถุคอมโพสิตด้วย เช่น คลาสการสั่งซื้อคอมโพสิตกับคลาสรายการสั่งซื้อ เมื่อมีการเลือกวัตถุการสั่งซื้อจะต้องทำการเลือกรายการสั่งซื้อที่คอมโพสิตกับวัตถุการสั่งซื้อนั้นด้วย

ซึ่งต้องออกแบบให้นักพัฒนาโปรแกรมประยุกต์เชิงวัตถุรู้สึกเหมือนเลือกวัตถุเพียงอย่างเดียวไม่ต้องสนใจว่าคลาสมีความสัมพันธ์อย่างไร

3.3. ปฏิสัมพันธ์ระหว่างโปรแกรมประยุกต์เชิงวัตถุกับฐานข้อมูลเชิงสัมพันธ์

การปฏิสัมพันธ์ระหว่างโปรแกรมประยุกต์เชิงวัตถุกับฐานข้อมูลเชิงสัมพันธ์ จากการวิเคราะห์การทำงานของโปรแกรมประยุกต์เชิงวัตถุกับฐานข้อมูลเชิงสัมพันธ์โดยทั่วไป ซึ่งสรุปเป็นกลุ่มลักษณะการทำงานร่วมกันได้ดังนี้

1. การเชื่อมต่อฐานข้อมูลเชิงสัมพันธ์
   คือเมื่อวัตถุของคลาสต้องการสร้างการเชื่อมต่อกับฐานข้อมูลเชิงสัมพันธ์ เพื่อกระทำสิ่งใดสิ่งหนึ่ง เช่น เพิ่มวัตถุ, ปรับปรุงวัตถุ เป็นต้น
2. การเพิ่มวัตถุในฐานข้อมูลเชิงสัมพันธ์
   คือวัตถุของคลาสต้องการจัดเก็บวัตถุใหม่ไว้ในฐานข้อมูลเชิงสัมพันธ์ หมายถึง การจัดเก็บคุณลักษณะของวัตถุใหม่ในฐานข้อมูลเชิงสัมพันธ์
3. การปรับปรุงวัตถุในฐานข้อมูลเชิงสัมพันธ์
   คือวัตถุของคลาสต้องการปรับปรุงวัตถุในฐานข้อมูลเชิงสัมพันธ์ หมายถึง วัตถุต้องการปรับปรุงคุณลักษณะของวัตถุของตนในฐานข้อมูลเชิงสัมพันธ์
4. การลบวัตถุในฐานข้อมูลเชิงสัมพันธ์
   คือวัตถุของคลาสต้องการลบวัตถุในฐานข้อมูลเชิงสัมพันธ์ หมายถึง วัตถุต้องการลบคุณลักษณะของวัตถุของตนในฐานข้อมูลเชิงสัมพันธ์
5. การเลือกวัตถุจากฐานข้อมูลเชิงสัมพันธ์
   คือการเลือกวัตถุหรือชุดของวัตถุจากฐานข้อมูลภายใต้เงื่อนไขที่กำหนด หมายถึง โปรแกรมประยุกต์เชิงวัตถุต้องการดึงคุณลักษณะของวัตถุหนึ่งหรือชุดวัตถุที่เก็บในฐานข้อมูลเชิงสัมพันธ์มาใช้งาน



รูปที่ 3.1 แผนภาพยูสเคสของโครงร่างฯ

3.4. ปฏิสัมพันธ์ระหว่างโปรแกรมประยุกต์เชิงวัตถุกับฐานข้อมูลเชิงสัมพันธ์และความสัมพันธ์ของคลาส

ความสัมพันธ์คลาสส่งผลให้คลาสมีลักษณะการทำงานที่ต่างกันซึ่งมีผลต่อการจัดเก็บวัตถุในฐานข้อมูลเชิงสัมพันธ์ด้วยซึ่งสรุปความสัมพันธ์ระหว่างคลาสได้ดังนี้ คือ

1. ซิงเกิลคลาสคือคลาสซึ่งไม่มีความสัมพันธ์กับคลาสอื่น
2. การรับทอดคลาสคือคลาสซึ่งมีสืบทอดกัน
3. ภาพรวมกลุ่มคลาสคือคลาสอยู่รวมกันแต่ไม่ได้เป็นส่วนประกอบของคลาส
4. คอมโพสิตชันคลาสคือคลาสที่คอมโพสิตกับคลาสอื่น

เมื่อพิจารณาความสัมพันธ์ของคลาสกับการปฏิสัมพันธ์ระหว่างโปรแกรมประยุกต์เชิงวัตถุกับฐานข้อมูลเชิงสัมพันธ์ สรุปได้เป็นลักษณะดังตารางที่ 3.1 จากตารางที่ 3.1 จะพบว่ามีรูปแบบอย่างน้อย 40 รูปแบบ เนื่องจากการรับทอดคลาสซึ่งคลาสตนเองอาจเป็นซิงเกิลคลาส ภาพรวมกลุ่มคลาสหรือคอมโพสิตชันคลาส เช่นเดียวกันภาพรวมกลุ่มคลาสซึ่งคลาสตนเองอาจเป็นซิงเกิลคลาส การรับทอดคลาสหรือคอมโพสิตชันคลาส และคอมโพสิตชันคลาสซึ่งคลาส

ตนเองอาจเป็นซิงเกิลคลาส การรับทอดคลาสหรือภาพรวมกลุ่มคลาส หากเป็นเช่นนั้นรูปแบบจะมีความซับซ้อนมากขึ้นในการปฏิสัมพันธ์กัน

ตารางที่ 3.1 แสดงความสัมพันธ์ของคลาส โปรแกรมประยุกต์เชิงวัตถุและฐานข้อมูลเชิงสัมพันธ์

	การเพิ่ม	การปรับปรุง	การลบ	การเลือก
ซิงเกิลคลาส	X	X	X	X
การรับทอดคลาส				
ซิงเกิลคลาส	X	X	X	X
ภาพรวมกลุ่มคลาส	X	X	X	X
คอมโพสิตชันคลาส	X	X	X	X
ภาพรวมกลุ่มคลาส				
ซิงเกิลคลาส	X	X	X	X
การรับทอดคลาส	X	X	X	X
คอมโพสิตชันคลาส	X	X	X	X
คอมโพสิตชันคลาส				
ซิงเกิลคลาส	X	X	X	X
การรับทอดคลาส	X	X	X	X
ภาพรวมกลุ่มคลาส	X	X	X	X

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

บทที่ 4

การออกแบบโครงร่างโปรแกรมประยุกต์เชิงวัตถุ สำหรับพัฒนาโปรแกรมประยุกต์ฐานข้อมูลเชิงสัมพันธ์

บทนี้กล่าวถึงการออกแบบโครงร่างโปรแกรมประยุกต์เชิงวัตถุสำหรับพัฒนาโปรแกรมประยุกต์ฐานข้อมูลเชิงสัมพันธ์

4.1. ขั้นตอนการออกแบบโครงร่างฯ

1. แผนภาพตารางความสัมพันธ์ที่เหมาะสมกับโครงร่างฯควรมีระดับการทำนอร์มอลไรเซชันระดับ 3 เนื่องจากการทำนอร์มอลไรเซชันระดับนี้มีความเพียงพอและเป็นที่ยอมรับตามหลักการออกแบบฐานข้อมูลเชิงสัมพันธ์
2. แผนภาพคลาสที่จะใช้งานกับโครงร่างฯ ต้องจัดรูปแบบความหลากหลาย (Multiplicity) เป็นแบบหนึ่งต่อหนึ่ง (one-to-one) หรือ หนึ่งต่อมากมาย (one-to-many) เท่านั้น จากปัญหาการความไม่สอดคล้องหลักการออกแบบฐานข้อมูลเชิงสัมพันธ์ เนื่องการความสัมพันธ์แบบ มากมายต่อมากมาย (many-to-many) ผิดหลักการออกแบบฐานข้อมูลเชิงสัมพันธ์ที่ดี
3. การแปลงแผนภาพคลาสเป็นแผนภาพตารางความสัมพันธ์เป็นลักษณะแบบหนึ่งตารางต่อหนึ่งคลาสเนื่องจากแผนภาพตารางความสัมพันธ์ที่ได้จากการแปลงในลักษณะดังกล่าวมีความใกล้เคียงหลักการออกแบบระบบที่ดีและการนอร์มอลไรเซชันระดับ 3 ของการออกแบบฐานข้อมูลเชิงสัมพันธ์ กล่าวคือ การทำดึงวัตถุเพื่อทำรายงานแบบด่วนสามารถได้โดยง่ายถึงปานกลางซึ่งขึ้นอยู่กับระดับการสืบทอดว่าลึกเพียงใด ความสะดวกในการเข้าถึงข้อมูลทำได้โดยง่ายถึงปานกลางซึ่งขึ้นอยู่กับระดับการสืบทอดว่าลึกเพียงใดเช่นกัน ในด้านความเร็วการเข้าถึงข้อมูลอยู่ระดับเร็วถึงปานกลางซึ่งขึ้นอยู่กับระดับการสืบทอดว่าลึกเพียงใดด้วยเช่นกัน เนื่องจากหากมีความลึกในการสืบทอดมากการเชื่อมโยงข้อมูลจะมีความลึกด้วยทำให้ความเร็วในการเข้าถึงข้อมูลลดลง ในการสนับสนุนการมีหลายรูปแบบสนับสนุนสูง เนื่องจากการแปลงหนึ่งคลาสเป็นหนึ่งตารางที่ให้มีสามารถนำตารางได้ในหลายรูปเช่น คลาสบุคคลในรูปคลาสลูกค้า คลาสบุคคลในรูปคลาสพนักงาน ในด้านการร่วมคู่หรืออยู่ร่วมกันต่ำ เนื่องจากคลาสถูกแยกเป็นตารางของคลาสแต่ละคลาส ทำให้การอยู่ร่วมกันหรือซ้ำซ้อนกันต่ำ แต่ด้วยวิธีการทำใช้งานจริงทำได้ยาก

4. ทำการออกแบบให้โครงร่างฯ ให้สามารถปฏิสัมพันธ์ระหว่างโปรแกรมประยุกต์เชิงวัตถุกับฐานข้อมูล เช่น การเชื่อมต่อ การยกเลิกการเชื่อมต่อ การเพิ่มวัตถุ การเลือกวัตถุ การลบวัตถุ การปรับปรุงวัตถุ เป็นต้น
5. ทำการออกแบบให้โครงร่างฯ ให้สามารถปฏิสัมพันธ์ระหว่างโปรแกรมประยุกต์เชิงวัตถุกับฐานข้อมูลและความสัมพันธ์ของคลาส เช่น การเพิ่มวัตถุ การเลือกวัตถุ การลบวัตถุ การปรับปรุงวัตถุ เมื่อคลาสมีความสัมพันธ์ แบบ ซิงเกิลคลาส การรับทอดคลาส ภาพรวมกลุ่มคลาส คอมโพสิตชันคลาสเป็นต้น

4.2. ข้อสมมุติฐานเบื้องต้นในการออกแบบโครงร่างฯ

1. เนื่องจากโครงร่างฯใช้ฐานข้อมูลเชิงสัมพันธ์เพื่อจัดเก็บคุณลักษณะวัตถุเท่านั้น จึงไม่มีความจำเป็นต้องสนใจความสัมพันธ์ในที่เกิดขึ้นระหว่างตารางความสัมพันธ์ในฐานข้อมูลเชิงสัมพันธ์

2. การจัดเก็บวัตถุในฐานข้อมูลเชิงสัมพันธ์ หมายถึง การจัดเก็บเฉพาะคุณลักษณะของวัตถุเท่านั้น โดยหนึ่งวัตถุจะถูกจัดเก็บเป็นหนึ่งระเบียน (Record) ในตารางความสัมพันธ์

3. การดึงวัตถุในฐานข้อมูลเชิงสัมพันธ์มาใช้ หมายถึง การสร้างวัตถุใหม่แล้วนำคุณลักษณะของวัตถุในฐานข้อมูลเชิงสัมพันธ์มากำหนดค่าให้วัตถุ

4. แผนภาพคลาสที่จะใช้งานในโครงร่างฯ ต้องจัดรูปความหลากหลายให้อยู่ในแบบหนึ่งต่อหนึ่งหรือ หนึ่งต่อมากมายเท่านั้น

5. เปลี่ยนแผนภาพคลาสไปเป็นแผนภาพความสัมพันธ์จะใช้รูปแบบการแปลงโดยใช้หนึ่งตารางความสัมพันธ์ต่อหนึ่งคลาส เนื่องจากมีความถูกต้องในการออกแบบแผนภาพความสัมพันธ์ตามหลักการออกแบบฐานข้อมูลเชิงสัมพันธ์มากที่สุด

6. กำหนดให้เพิ่มเขตข้อมูลในตารางความสัมพันธ์สำหรับเก็บหมายเลขวัตถุชื่อ OID (Object ID) ลงในตารางความสัมพันธ์ทุกตารางความสัมพันธ์และกำหนดคุณสมบัติภายใต้โครงร่างฯ ดังนี้
   - หมายเลขวัตถุต้องไม่ซ้ำภายใต้วัตถุชนิดเดียวกันที่เก็บอยู่ภายใต้ตารางความสัมพันธ์เดียวกัน
   - หมายเลขวัตถุต้องเป็นหมายเลขเดียวกันเมื่อวัตถุนั้นมีความสัมพันธ์กัน หรืออาจเรียกว่าหมายเลขความสัมพันธ์ของวัตถุ

4.3. การออกแบบโครงร่างฯ

เนื่องด้วยการแปลงแผนภาพคลาสไปเป็นแผนภาพความสัมพันธ์แบบหนึ่งคลาสต่อหนึ่งตารางความสัมพันธ์ ทำให้วัตถุหนึ่งวัตถุอาจแยกกันเก็บอยู่ในตารางความสัมพันธ์ และเมื่อนำวัตถุกลับมาใช้จะต้องทำอย่างไรให้วัตถุที่แยกกันกลับมาเป็นเช่นเดิม จากปัญหาดังกล่าวเป็นปัญหาในการจัดการกับวัตถุที่อยู่ในฐานข้อมูลเชิงสัมพันธ์จึงเกิดการออกแบบโครงร่างฯ

4.3.1. การแยกวัตถุเพื่อเก็บ รวมวัตถุเมื่อใช้ (Splitting and Merging Object) ในการแยกวัตถุหมายถึงการแยกสองส่วน คือ ส่วนแรกคือการแยกคุณลักษณะวัตถุออกจากวัตถุเพื่อจัดเก็บในฐานข้อมูลเชิงสัมพันธ์ และส่วนที่สองคือการแยกคุณลักษณะของวัตถุออกตามความสัมพันธ์คลาส ตามลักษณะความสัมพันธ์แบบซิงเกิลคลาสดังรูป 4.1 ความสัมพันธ์แบบการรับทอดคลาสดังรูป 4.2 ความสัมพันธ์แบบภาพรวมกลุ่มคลาสดังรูป 4.3 และรูปที่ 4.4 สำหรับคอมโพสิตชันคลาส

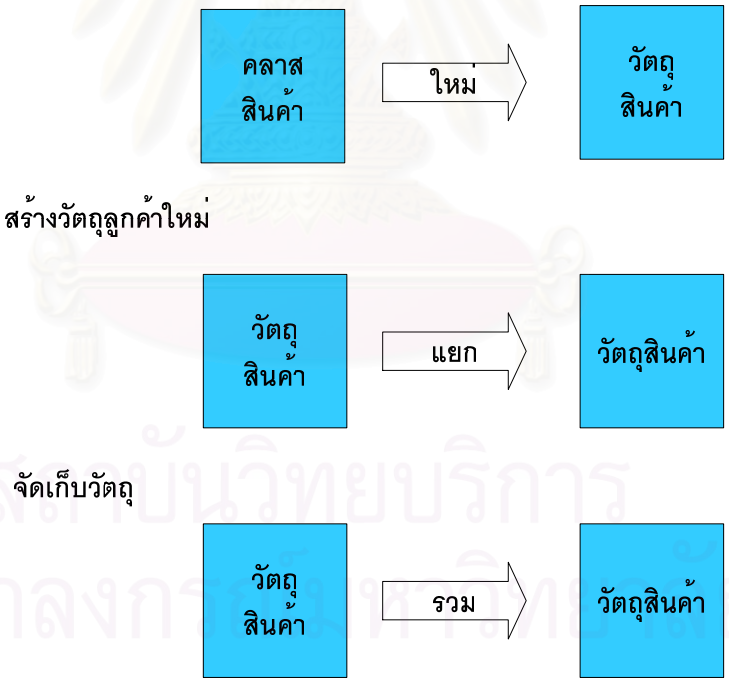
จากรูปที่ 4.1 คลาสสินค้าเป็นซิงเกิลคลาส เมื่อสร้างวัตถุสินค้าใหม่ วัตถุจะประกอบด้วยส่วนของคุณลักษณะคลาสสินค้าเพียงอย่างเดียว และเมื่อต้องการทำวัตถุให้เป็นวัตถุถาวรหรือจัดเก็บวัตถุ จึงไม่ต้องทำการแยกคุณลักษณะของวัตถุใดๆออกจากกัน ซึ่งสามารถทำการจัดเก็บลงตารางความสัมพันธ์ของสินค้าได้โดยตรง และเมื่อต้องการนำวัตถุสินค้าที่จัดเก็บนั้นกลับมาใช้ สามารถดึงคุณลักษณะของวัตถุสินค้าเพียงวัตถุเดียวได้เสมือนเป็นวัตถุเดิม ซึ่งต้องมีออกแบบการจัดเก็บเพื่อให้จัดเก็บคุณลักษณะของวัตถุได้โดยตรงที่ตารางความสัมพันธ์เพียงตารางเดียว

จากรูปที่ 4.2 คลาสลูกค้าการรับทอดจากคลาสบุคคล เมื่อสร้างวัตถุลูกค้าใหม่ วัตถุจะประกอบด้วยส่วนของคุณลักษณะคลาสบุคคลและส่วนของคุณลักษณะคลาสลูกค้า เมื่อต้องการทำวัตถุให้เป็นวัตถุถาวรหรือจัดเก็บวัตถุ จะต้องทำการแยกคุณลักษณะของวัตถุบุคคลกับวัตถุลูกค้าออกจากกันแล้วทำการจัดเก็บลงตารางความสัมพันธ์ของบุคคลและลูกค้า และเมื่อต้องการนำวัตถุลูกค้าที่จัดเก็บกลับมาใช้ ต้องทำการรวมคุณลักษณะของวัตถุบุคคลและคุณลักษณะของวัตถุลูกค้าที่ทำการแยกออกจากการกันนำมารวมกันเสมือนเป็นวัตถุเดิม ซึ่งต้องมีออกแบบการจัดเก็บเพื่อให้ทราบว่าคุณลักษณะของวัตถุที่จะมาประกอบกันอยู่ที่ระเบียนใดในตารางความสัมพันธ์ใด

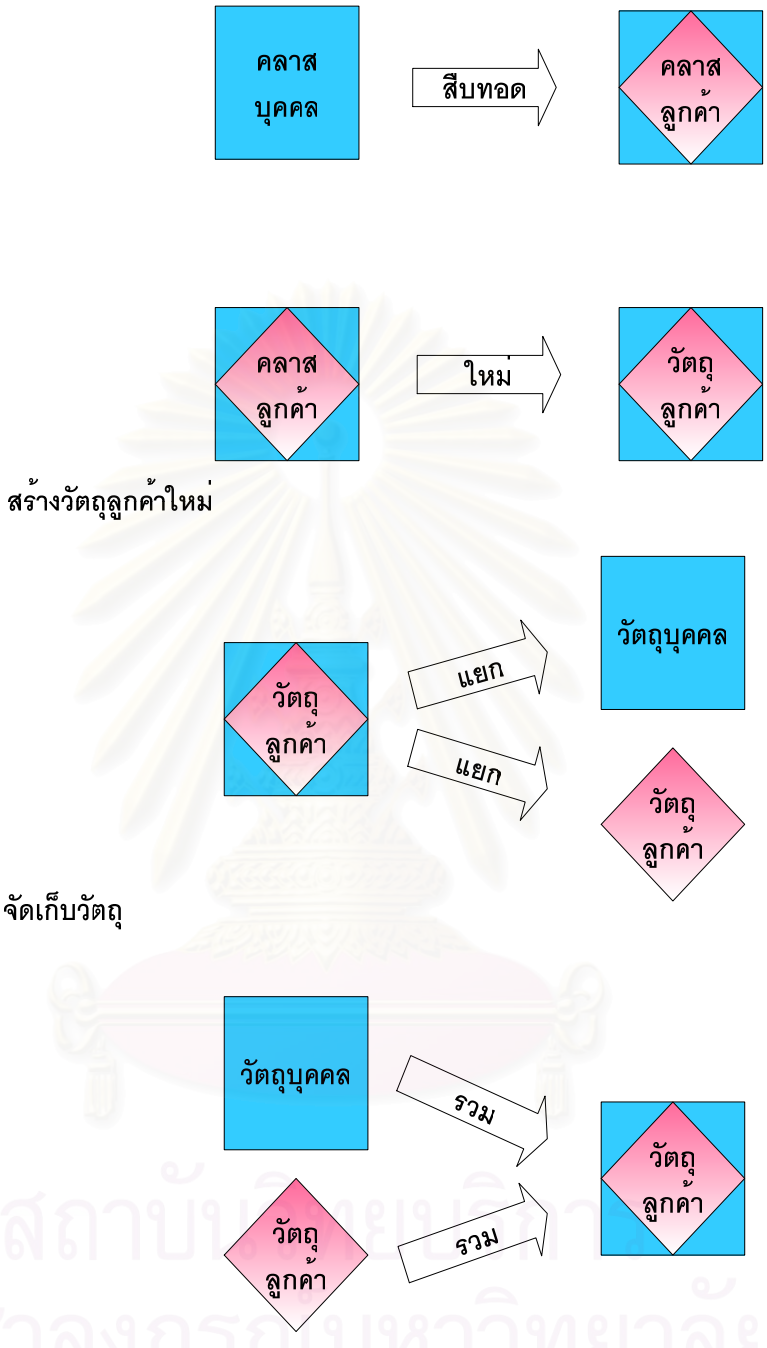
จากรูปที่ 4.3 คลาสลูกค้ารวมกลุ่มกับคลาสพนักงานขาย เมื่อสร้างวัตถุลูกค้าใหม่ วัตถุจะประกอบด้วยส่วนของคุณลักษณะคลาสลูกค้าและคุณลักษณะซึ่งใช้เก็บหมายเลขวัตถุพนักงานขายที่รวมกลุ่มกัน และเมื่อต้องการทำวัตถุให้เป็นวัตถุถาวรหรือจัดเก็บวัตถุ จะทำการ

จัดเก็บคุณลักษณะของวัตถุลูกค้าเพียงอย่างเดียวลงตารางความสัมพันธ์ของลูกค้า โดยไม่ต้องทำการจัดเก็บวัตถุพนักงานขายที่รวมกลุ่มและเมื่อต้องการนำวัตถุลูกค้าที่จัดเก็บกลับมาใช้ จะต้องทำการดึงคุณลักษณะของวัตถุลูกค้าและดึงวัตถุพนักงานขายที่รวมกลุ่มเพื่อให้สามารถใช้ภายในวัตถุลูกค้า การดึงลักษณะนี้ต่างจากความสัมพันธ์แบบคอมโพสิตชันหรือการรับทอดในส่วนที่ดึงวัตถุทั้งวัตถุไม่ใช้เพียงคุณลักษณะของหมายเลขวัตถุที่รวมกันเท่านั้น ซึ่งต้องมีออกแบบการจัดเก็บเพื่อให้ทราบว่าคุณลักษณะของวัตถุที่รวมกันอยู่ที่ระเบียนใดในตารางความสัมพันธ์ใด

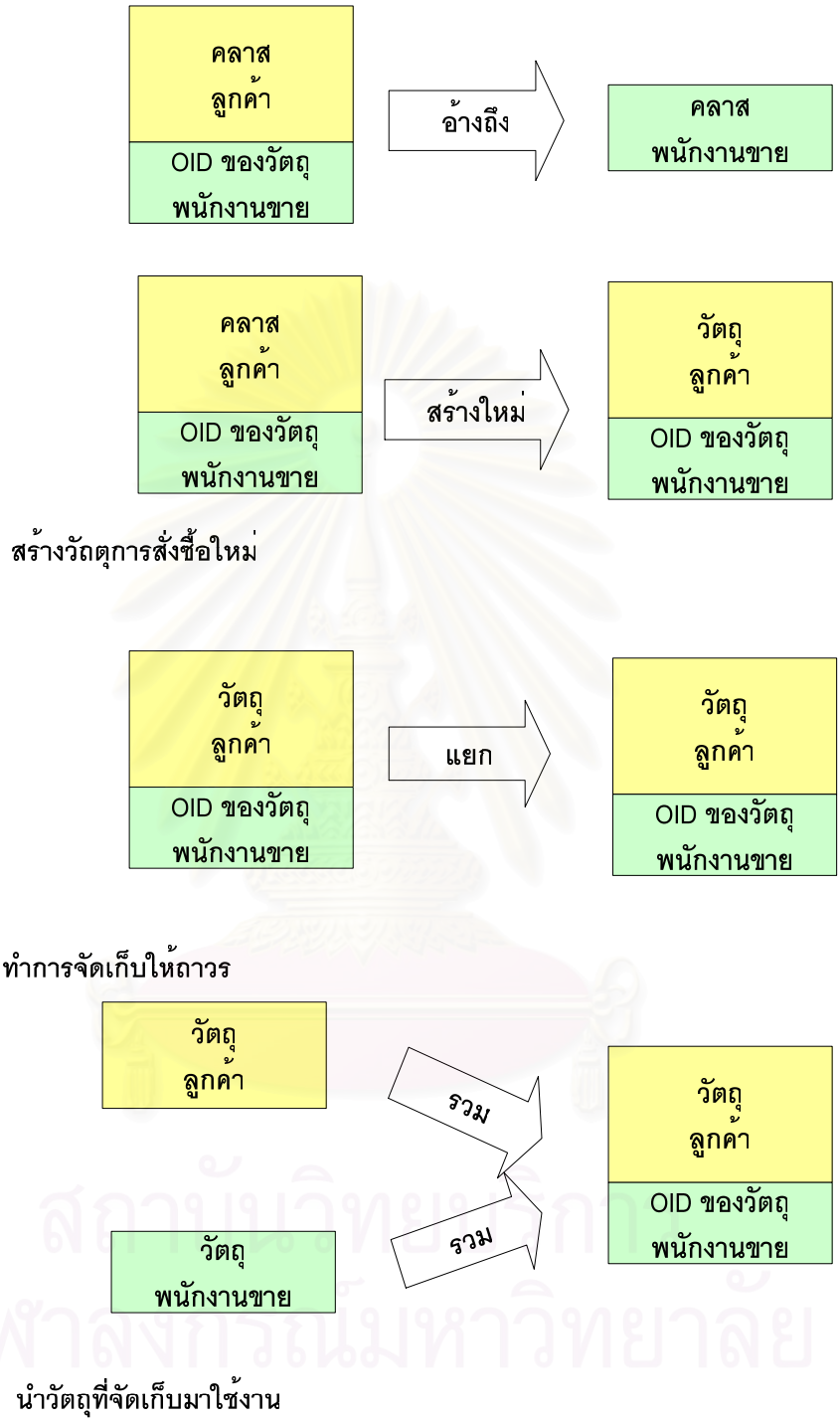
จากรูปที่ 4.4 คลาสการสั่งซื้อประกอบด้วยรายการสั่งซื้อ เมื่อสร้างวัตถุการสั่งซื้อใหม่ วัตถุจะประกอบด้วยวัตถุรายการสั่งซื้อด้วย เช่นเดียวกับความสัมพันธ์คลาสแบบการรับทอด เมื่อพิจารณาเฉพาะส่วนที่ต้องการจัดเก็บ คือคุณลักษณะของวัตถุการสั่งซื้อซึ่งประกอบไปด้วยคุณลักษณะของวัตถุรายการสั่งซื้อ และเมื่อต้องการการจัดเก็บจะทำการแยกวัตถุการสั่งซื้อออกจากวัตถุรายการการสั่งซื้อ การนำวัตถุกลับมาใช้ใหม่จะทำการรวมคุณลักษณะของวัตถุทั้งสองเข้าด้วยกันเสมือนเป็นวัตถุเดิม



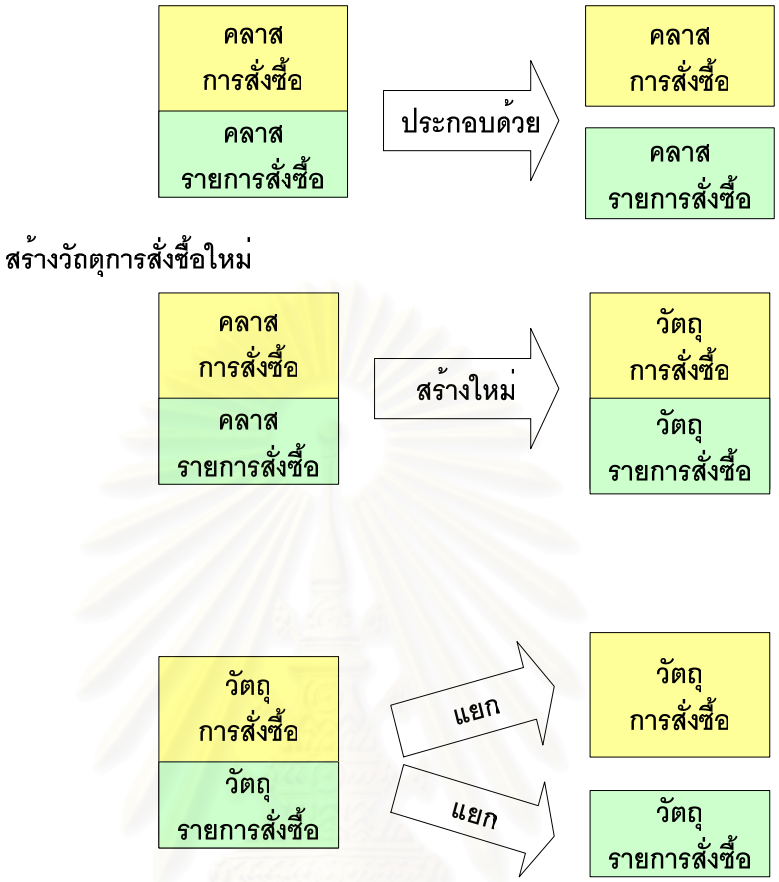
รูปที่ 4.1 แผนภาพแสดงการแยกและรวมวัตถุของซิงเกิลคลาส



รูปที่ 4.2 แผนภาพแสดงการแยกและรวมวัตถุของการรับทอดคลาส



รูปที่ 4.3 แผนภาพแสดงการแยกและรวมวัตถุของภาพรวมกลุ่มคลาส



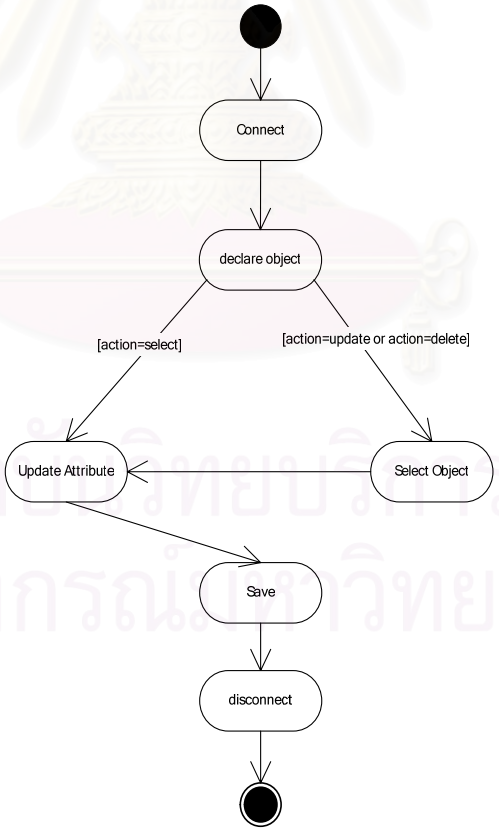
รูปที่ 4.4 แผนภาพแสดงการแยกและรวมวัตถุของคอมโพสิตชันคลาส

4.4. การทำงานโดยรวมของโครงร่างฯ

4.4.1. แผนภาพกิจกรรมขั้นตอนการใช้งานโครงร่างฯ

จากรูปที่ 4.5 อธิบายลักษณะการใช้งานโครงร่างฯ ภายในโปรแกรมประยุกต์ โดยโปรแกรมจะดำเนินขั้นตอนโดยรวมดังนี้

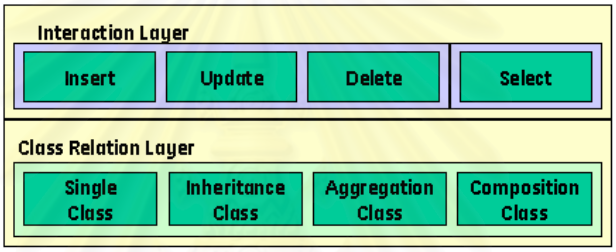
1. โปรแกรมเมอร์เขียนโปรแกรมเพื่อสร้างวัตถุเชื่อมต่อเพื่อใช้ในการเชื่อมต่อไปยังฐานข้อมูลเชิงสัมพันธ์ที่ใช้ในการจัดเก็บวัตถุ
2. โปรแกรมเมอร์เขียนโปรแกรมเพื่อประกาศตัวแปรชนิดวัตถุที่ต้องการใช้งาน
3. โปรแกรมเมอร์พิจารณาว่าต้องการปรับปรุงวัตถุหรือไม่
   3.1 ถ้าใช่ โปรแกรมเมอร์เขียนโปรแกรมเพื่อเลือกวัตถุที่ต้องการปรับปรุง
4. โปรแกรมเมอร์เขียนโปรแกรมเพื่อทำการกำหนดค่าคุณลักษณะตามต้องการ
5. โปรแกรมเมอร์เขียนโปรแกรมเพื่อทำการบันทึก
6. โปรแกรมเมอร์เขียนโปรแกรมเพื่อทำการยกเลิกการเชื่อมต่อ



รูปที่ 4.5 แผนภาพกิจกรรมโดยรวมของโครงร่างฯ

4.4.2. แผนภาพภาพชั้นการทำงานของโครงร่างฯ

จากรูปที่ 4.6 แสดงให้เห็นว่าโครงร่างฯทำงานโดยพิจารณาวัตถุว่าปฏิสัมพันธ์กับฐานข้อมูลเชิงสัมพันธ์อย่างไร คือ การเพิ่ม การปรับปรุง การลบหรือการเลือกวัตถุ ต่อจากนั้นโครงร่างฯพิจารณาชนิดของความสัมพันธ์ของคลาสว่าเป็นชนิดใด คือ ซิงเกิลคลาส การรับทอดคลาส ภาพรวมกลุ่มคลาสหรือคอมโพสิตชันคลาส โครงร่างฯจะปฏิสัมพันธ์ตามความสัมพันธ์คลาส โดยชั้นการทำงานทั้งสองจะทำงานเป็นลำดับกันกล่าวคือ โครงร่างฯพิจารณาว่าวัตถุต้องปฏิสัมพันธ์กับฐานข้อมูลเชิงสัมพันธ์อย่างไร เมื่อทราบแล้วจึงส่งให้ชั้นความสัมพันธ์คลาสพิจารณาว่าวัตถุนั้นมีความสัมพันธ์แบบใด แล้วแสดงพฤติกรรมตามนั้น



รูปที่ 4.6 แผนภาพชั้นการทำงานของโครงร่างฯ

คลาสเป็นกลไกหลักในการปฏิสัมพันธ์ของโครงร่างฯ คือ คลาส DbObject โครงร่างฯ กำหนดให้โปรแกรมเมอร์ต้องสร้างเมท็อดใหม่แทนที่เมท็อดนามธรรมของคลาส โดยเป็นไปข้อกำหนดของโครงร่างฯ เพื่อให้กลไกของโครงร่างฯทำงานได้ โดยหลักการโครงร่างฯ ทำหน้าที่บริหารจัดการข้อมูลของวัตถุให้เป็นไปตามการออกแบบที่กำหนดในโครงร่างฯ โดยการสร้างประโยคคำสั่งเอสคิวแอลไว้ภายในโครงร่างฯ แล้วเพื่อส่งให้ฐานข้อมูลประมวลผลให้ข้อมูลของวัตถุตรงตามการออกแบบการจัดเก็บวัตถุที่ใช้ในโครงร่างฯ กล่าวคือเมื่อต้องการใช้วัตถุต้องทำการร่วมข้อมูลของวัตถุซึ่งกระจายในตารางความสัมพันธ์ตามความสัมพันธ์ของคลาส โดยเป็นหน้าที่ของชั้นการทำงานความสัมพันธ์คลาสตามชั้นการทำงานปฏิสัมพันธ์ประเภทการเลือก และทำหน้าที่ควบคุมการจัดการกับข้อมูลวัตถุโดยการสร้างประโยคเอสคิวแอลในการเพิ่ม ปรับปรุง และลบวัตถุ โดยเป็นหน้าที่ของชั้นการทำงานปฏิสัมพันธ์เพื่อจัดการกับข้อมูลซึ่งกระจายอยู่ในตารางความสัมพันธ์ตามความสัมพันธ์ของคลาส ซึ่งพิจารณาโดยชั้นการทำงานของความสัมพันธ์คลาส

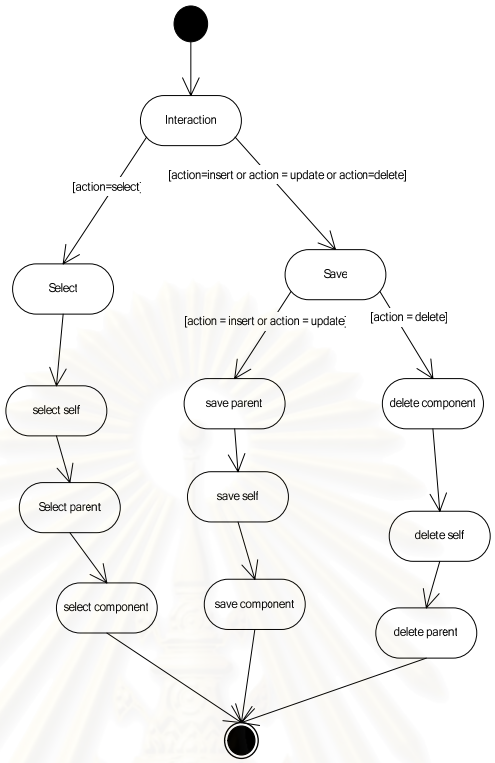
4.4.2.1. การทำงานของการดึงวัตถุกลับมาใช้

โดยกลไกการทำงานของโครงร่างฯ เพื่อดึงวัตถุมาใช้งานผ่านทางเมท็อด Select ของคลาส DbObject เพื่อให้สามารถทำงานกับความสัมพันธ์คลาสชนิดต่างๆ ภายใต้การเลือกนั้น โครงร่างฯ มีกลไกการสร้างวัตถุใหม่โดยการดึงคุณลักษณะของวัตถุตนเองก่อน จากนั้นทำการดึงคุณลักษณะของวัตถุคอมโพสิตถ้ามี ซึ่งมีเพียงกรณีคอมโพสิตชันคลาส และทำการดึงคุณลักษณะของวัตถุแม่ถ้ามี ซึ่งมีเพียงกรณีการรับทอดคลาส ซึ่งจะได้ข้อมูลของวัตถุทั้งหมดมาประกอบเป็นวัตถุเสมือนวัตถุเดิม ดังรูปที่ 4.7

4.4.2.2. การทำงานของการจัดการข้อมูลวัตถุ

โดยกลไกการทำงานของโครงร่างฯ เพื่อปรับปรุงคุณลักษณะวัตถุผ่านเมท็อด Save ของคลาส DbObject โดยขั้นตอนโครงร่างฯ พิจารณาสถานะของวัตถุว่าเป็นวัตถุใหม่โครงร่างฯจะทำการเพิ่มวัตถุ ถ้าเป็นวัตถุที่ดำเนินชีวิตอยู่ โครงร่างฯ จะทำการปรับปรุง และถ้าสถานะของวัตถุเป็นตาย โครงร่างฯ จะทำการลบวัตถุ เพื่อให้สามารถทำงานกับความสัมพันธ์คลาสชนิดต่างๆ ภายใต้การปฏิสัมพันธ์แบบการเพิ่มและการปรับปรุงวัตถุ โครงร่างฯ มีกลไกการเพิ่มหรือปรับปรุงวัตถุแม่ก่อนสำหรับกรณีการรับทอดคลาส จากนั้นทำการเพิ่มหรือปรับปรุงวัตถุตนเองแล้วทำการเพิ่มหรือปรับปรุงวัตถุคอมโพสิตสำหรับกรณีที่เป็นคอมโพสิตชันคลาส และกรณีการปฏิสัมพันธ์แบบการลบวัตถุ โครงร่างฯ จะทำการลบวัตถุคอมโพสิตก่อนจากนั้นจึงทำการลบวัตถุตนเอง และวัตถุแม่ตามลำดับ ดังรูปที่ 4.5

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย



รูปที่ 4.7 แผนภาพกิจกรรมการทำงานของโครงร่างฯกับความสัมพันธ์คลาส

4.5. โครงร่างฯโดยรวม

จากแนวความคิดและการวิเคราะห์ปัญหาและความต้องการของโครงร่างฯ นำไปสู่การออกแบบโครงร่างฯ ซึ่งผ่านการทดสอบโดยการทำไปสร้างให้เป็นจริงแล้วนำไปพัฒนาโปรแกรมประยุกต์จริงแล้วทำการประเมินโครงร่างฯว่ามีปัญหาอย่างไร เพื่อนำกลับไปปรับปรุงโครงร่างฯเพื่อออกเป็นรุ่นต่อไปจนได้โครงร่างฯ ที่ต้องการ ซึ่งในการออกแบบโครงร่างฯ ไม่สามารถทำได้เพียงการออกแบบเพียงครั้งเดียว ซึ่งหลังจากภายกระบวนการดังกล่าวมาหลายครั้ง จึงได้ส่วนต่างๆของโครงร่างฯ ดังนี้

4.5.1. แผนภาพคลาสโครงร่างฯ

ภาพรวมของคลาสในโครงร่างฯ ซึ่งมีลักษณะการใช้งานโครงร่างฯด้วยการปรับและสร้างเมท็อดใหม่แทนที่เมท็อดเดิมตามโครงร่างฯกำหนดในบางเมท็อดและคลาสของโครงร่างเพื่อนำไปใช้งานในโปรแกรมประยุกต์ ซึ่งคลาสทั้งหมดภายใต้โครงร่างฯ ประกอบด้วย 6 คลาส ดังรูปที่ 4.8

4.5.1.1. คลาส DbApp

เป็นคลาสนามธรรมซึ่งทำหน้าที่เป็นคลาสควบคุมการขอเชื่อมต่อกับฐานข้อมูลเชิงสัมพันธ์ การสร้างวัตถุเชื่อมต่อ การขอยกเลิกการเชื่อมต่อ และการทำลายการเชื่อมต่อ คลาสนี้จะขึ้นอยู่กับคลาสการเชื่อมต่อ DbConnect ซึ่งเมื่อโปรแกรมประยุกต์ต้องการสร้างวัตถุเชื่อมที่ต่างออกไป ต้องมีการสร้างคลาสใหม่ซึ่งสืบทอดจากคลาส เพื่อให้โปรแกรมเมอร์ปรับเมท็อดของโครงร่างฯ ให้เหมาะกับโปรแกรมยุกต์เชิงวัตถุ โดยคลาสดังกล่าวประกอบด้วยเมท็อดดังนี้

เมท็อดคอร์ฮอทสปอทของคลาสดังนี้

1. LoginDB(in user : std::string, in password : std::string) : bool

เป็นเมท็อดที่โครงร่างฯ กำหนดให้เรียกเมื่อเริ่มทำงานโปรแกรมประยุกต์ ซึ่งทำหน้าที่ในการเชื่อมต่อไปยังฐานข้อมูลเชิงสัมพันธ์ โดยส่งค่าชื่อผู้ใช้และรหัสผ่าน

2. LogoutDB() : void

เป็นเมท็อดที่โครงร่างฯ กำหนดให้เรียกเมื่อออกจากโปรแกรมประยุกต์ ทำหน้าที่ยกเลิกการเชื่อมต่อจากฐานข้อมูลเชิงสัมพันธ์

3. InitialConnector(in user : std::string, in password : std::string) : void

เป็นเมท็อดซึ่งถูกเรียกโดยเมท็อด LoginDB เพื่อเริ่มต้นการเชื่อมต่อ

4. Connect() : bool

เป็นเมท็อดซึ่งถูกเรียกโดยเมท็อด InitialConnector เพื่อเชื่อมต่อฐานข้อมูลเชิงสัมพันธ์

เมท็อดไวท์บอกซ์ฮอทสปอทของคลาสดังนี้

5. GetConnect() : DbConnect

เป็นเมท็อดจะคืนวัตถุการเชื่อมต่อของโปรแกรมประยุกต์ขณะนั้น

6. NewConnect() : DbConnect

เป็นเมท็อดสร้างวัตถุเชื่อมต่อใหม่แล้วส่งวัตถุที่สร้างคืน เมท็อดนี้ถูกเรียกโดยโครงร่างฯ

7. DestroyConnect() : void

เป็นเมท็อดทำลายวัตถุเชื่อมต่อถูกใช้อยู่ขณะนั้น เมท็อดนี้ถูกเรียกโดยโครงร่างฯ

4.5.1.2. คลาส DbConnect

เป็นคลาสนามธรรม ซึ่งทำหน้าที่สำหรับเชื่อมต่อไปยังฐานข้อมูลเชิงสัมพันธ์ เมื่อนำไปใช้งานจริงต้องทำการปรับให้เหมาะกับการเชื่อมต่อ เช่น เมื่อต้องการเชื่อมต่อผ่านมาตรฐานโอดีบีซี (Open Database Connectivity: ODBC) ต้องให้นักพัฒนาที่มีประสบการณ์เกี่ยวกับการเชื่อมต่อฐานข้อมูลผ่านมาตรฐานโอดีบีซี นำโครงร่างฯ ไปพัฒนาซึ่งจะได้คลาสใหม่ ซึ่งทำให้โครงร่างฯ สามารถเชื่อมผ่านมาตรฐานโอดีบีซีได้ เป็นต้น คลาสนี้เป็นคลาสประเภทไวท์บอกซ์ฮอทสปอทเพื่อให้โปรแกรมเมอร์ปรับเมท็อดของโครงร่างฯ ให้เหมาะกับโปรแกรมยุกต์เชิงวัตถุ โดยคลาสดังกล่าวประกอบด้วยเมท็อดดังนี้

เมท็อดคอร์ฮอทสปอท ของคลาสดังนี้

1. SetUser(in user : std::string) : void

   เป็นเมท็อดใช้กำหนดผู้ใช้ในการเชื่อมต่อไปยังฐานข้อมูลเชิงสัมพันธ์

2. GetUser() : std::string

   เป็นเมท็อดคืนค่าผู้ใช้

3. SetPassword(in password : std::string) : void

   เป็นเมท็อดกำหนดรหัสผ่านเพื่อใช้ในการเชื่อมต่อ

4. GetPassword() : std::string

   เป็นเมท็อดคืนค่ารหัสผ่าน

5. IsConnect() : bool

   เป็นเมท็อดทำหน้าที่ตรวจสอบสถานะของการเชื่อมต่อฐานข้อมูลเชิงสัมพันธ์ หากคืนค่าเป็นจริง แสดงว่าเชื่อมต่ออยู่ หากเป็นเท็จ แสดงว่าไม่ได้เชื่อมต่อ

เมท็อดไวท์บอกซ์ฮอทสปอทของคลาสดังนี้

6. ExecuteSQL(in cmd : std::string) : bool

   เป็นเมท็อดซึ่งโครงร่างฯเรียกใช้ เมื่อต้องการส่งคำสั่งเอสคิวแอลไปเพื่อทำกลุ่มภาษาจัดการข้อมูล (Data Manipulate Language: DML) เช่น การเพิ่ม, การลบ, การปรับปรุงระเบียน เป็นต้น ที่ฐานข้อมูลเชิงสัมพันธ์

7. QuerySQL(in cmd : std::string, in pRecSet : std::vector<Record>) : bool

   เป็นเมท็อดซึ่งโครงร่างฯเรียกใช้ เมื่อต้องการส่งคำสั่งเอสคิวแอลไปเพื่อทำการเลือกระเบียนที่ฐานข้อมูลเชิงสัมพันธ์ แล้วส่งชุดของวัตถุระเบียนที่เลือกได้

8. Connect() : bool

เป็นเมท็อดทำการเชื่อมต่อฐานข้อมูลสัมพันธ์ หากเชื่อมต่อได้จะคืนค่าเป็นจริง หากเชื่อมต่อไม่สำเร็จจะคืนค่าเป็นเท็จ

9. Disconnect() : void

เป็นเมท็อดทำการยกเลิกการเชื่อมต่อไปฐานข้อมูลเชิงสัมพันธ์

4.5.1.3. คลาส DbObject

เป็นคลาสนามธรรม เพื่อทำหน้าที่ในการเพิ่ม การปรับปรุง การลบ และการเลือกวัตถุไปยังฐานข้อมูลเชิงสัมพันธ์ เมื่อคลาสใดต้องการจัดเก็บในวัตถุโดยสืบทอดจากคลาสนี้ ซึ่งเป็นคลาสประเภทไวท์บอกซ์ฮอทสปอทเพื่อให้โปรแกรมเมอร์ปรับเมท็อดของโครงร่างฯ ให้เหมาะกับโปรแกรมยุกต์เชิงวัตถุ โดยคลาสดังกล่าวประกอบด้วยเมท็อดดังนี้

เมท็อดคอร์ฮอทสปอท ของคลาสดังนี้

1. SaveRDB() : bool

เป็นเมท็อดทำหน้าที่ในการสร้างประโยคเอสคิวแอลสำหรับการเพิ่ม การปรับปรุง การลบ ของวัตถุแล้วทำการส่งให้ฐานข้อมูลเชิงสัมพันธ์ประมวลผลหากสำเร็จจะคืนค่าเป็นจริงหากไม่จะคืนค่าเป็นเท็จ

2. SelectRDB(in criteria : std::string, out pObjectSet : std::vector<DbObject>) : bool

เป็นเมท็อดทำหน้าที่เลือกระเบียนที่ต้องการตามเงื่อนไขจากฐานข้อมูลเชิงสัมพันธ์แล้วทำการสร้างวัตถุแล้วส่งคืนค่าผู้ใช้หากสำเร็จจะคืนค่าเป็นจริงหากไม่จะคืนค่าเป็นเท็จ

3. GetRecordToObject(in oid : long) : bool

เป็นเมท็อดซึ่งให้นำค่าคุณลักษณะที่เก็บในฐานข้อมูลเชิงสัมพันธ์ และมีหมายเลขวัตถุตามที่ต้องการมากำหนดให้กับวัตถุหากสำเร็จจะคืนค่าเป็นจริงหากไม่จะคืนค่าเป็นเท็จ

4. SelectRecordToObject(in criteria : std::string) : bool

เป็นเมท็อดระดับล่างซึ่งทำหน้าที่เหมือนกับ GetRecordToObject ต่างตรงการรับค่าเป็นเงื่อนไขแทนหมายเลขวัตถุ หากสำเร็จจะคืนค่าเป็นจริงหากไม่จะคืนค่าเป็นเท็จ

5. SelectParentRDB(inout pObjectSet : std::vector<DbObject>) : bool

เป็นเมท็อดระดับล่างทำหน้าที่พิจารณาว่ามีวัตถุของคลาสแม่หรือไม่หากมีจะทำการดึงค่าคุณลักษณะของวัตถุแม่มากำหนดให้กับวัตถุโดยพิจารณาที่หมายเลขของวัตถุแม่กับลูกตรงกัน หากสำเร็จจะคืนค่าเป็นจริงหากไม่จะคืนค่าเป็นเท็จ

6. SelectComponentRDB(in pObjectSet : std::vector<DbObject>) : bool

เป็นเมท็อดระดับล่างทำหน้าที่พิจารณาว่ามีวัตถุคอมโพเนนท์หรือไม่หากมีจะทำการดึงค่าคุณลักษณะของวัตถุคอมโพเนนท์มากำหนดให้กับวัตถุโดยพิจารณาหมายเลขของวัตถุที่ตรงกับหมายเลขของวัตถุคอมโพเนนท์ที่ตรงกัน หากสำเร็จจะคืนค่าเป็นจริงหากไม่จะคืนค่าเป็นเท็จ

7. GenSQL(in criteria : std::string) : std::string

เป็นเมท็อดทำหน้าที่ในการสร้างประโยคคำสั่งในการเลือกระเบียนจากฐานข้อมูลเชิงสัมพันธ์

8. PushAssociate(in pObject : DbObject) : int

เป็นเมท็อดรับค่าวัตถุซึ่งเป็นวัตถุแอสโซซิเอดเพื่อนำไปเก็บภายในวัตถุซึ่งจะส่งค่าจำนวนของวัตถุแอสโซซิเอดที่อยู่ภายในวัตถุ

9. PopAssociate() : DbObject

เป็นเมท็อดเมื่อเรียกจะคืนค่าวัตถุแอสโซซิเอดซึ่งถูกจัดเก็บล่าสุด

10. DestroyAllAssociate() : void

เป็นเมท็อดทำหน้าที่ทำลายวัตถุแอสโซซิเอดที่อยู่ในวัตถุทั้งหมด

11. SetConnect(in pConnect : DbConnect) : void

เป็นเมท็อดการกำหนดวัตถุเชื่อมต่อให้กับวัตถุ

12. GetConnect() : DbConnect

เป็นเมท็อดคืนวัตถุการเชื่อมต่อที่ใช้อยู่ภายในวัตถุขณะนั้น

13. GetOID() : long

เป็นเมท็อดคืนค่าหมายเลขวัตถุ

14. SetOID(in oid : long) : void

เป็นเมท็อดใช้กำหนดค่าหมายเลขวัตถุ

15. GetLifeStatus() : ObjectLife

เป็นเมท็อดคืนค่าสถานะของวัตถุว่าเกิดใหม่ มีชีวิตหรือตาย

16. SetLifeStatus(in life : ObjectLife) : void

เป็นเมท็อดกำหนดสถานะของวัตถุ

17. Delete() : bool

เป็นเมท็อดที่ต้องเรียกเมื่อต้องการลบวัตถุจากฐานข้อมูลก่อนเรียกเมท็อดบันทึก

18. Save() : bool

เป็นเมท็อดทำหน้าที่เพื่อกระทำสิ่งใดสิ่งหนึงระหว่างวัตถุกับฐานข้อมูลเชิงสัมพันธ์เช่น หากเป็นวัตถุเกิดใหม่เมท็อดจะทำการเพิ่มวัตถุ หากวัตถุมีสถานะชีวิตเป็นมีชีวิตเมท็อดจะทำการปรับปรุงวัตถุ และหากวัตถุมีสถานะชีวิตเป็นตาย เมท็อดจะทำการลบวัตถุออกจากฐานข้อมูลเชิงสัมพันธ์

19. Select(in criteria : std::string, out pObjectSet : std::vector<DbObject>) : bool

เป็นเมท็อดที่รับค่าเงื่อนไขที่ต้องการเลือกวัตถุของคลาสแล้วส่งคืนกลับมา หากสำเร็จจะคืนค่าเป็นจริงหากไม่จะคืนค่าเป็นเท็จ

20. RecordToObject(inout pObjectSet : std::vector<DbObject>, in pRecord : Record, inout pObject : DbObject>) : bool

เป็นเมท็อดระดับล่างทำหน้าที่การโอนย้ายคุณลักษณะจากวัตถุระเบียนไปวัตถุแล้วทำการใส่ลงในชุดของวัตถุอีกครั้ง หากสำเร็จจะคืนค่าเป็นจริงหากไม่จะคืนค่าเป็นเท็จ

21. GetMemberValue() : bool

เป็นเมท็อดระดับบนทำหน้าที่เลือกวัตถุจากค่าหมายเลขวัตถุซึ่งกำหนดอยู่วัตถุขณะนั้น หากสำเร็จจะคืนค่าเป็นจริงหากไม่จะคืนค่าเป็นเท็จ

22. SetNewOID() : void

เป็นเมท็อดที่ทำหน้าที่กำหนดหมายเลขวัตถุใหม่ตามตรรกที่กำหนดไว้โดยต้องเป็นหมายเลขที่ไม่ซ้ำ

เมท็อดไวท์บอกซ์ฮอทสปอทของคลาสดังนี้

23. GetNickName() : std::string

เป็นเมท็อดคืนค่าชื่อของวัตถุซึ่งเป็นชื่อใดก็ได้ขึ้นอยู่กับผู้สร้างคลาสใหม่กำหนด

24. GetString(in index : int) : std::string

เป็นเมท็อดคืนค่าสายอักขระซึ่งเป็นค่าของคุณลักษณะตามตัวเลขดัชนีที่กำหนด เช่น คลาสบุคคลมีคุณลักษณะ 2 คุณลักษณะ คือ ชื่อและที่อยู่ อาจกำหนดให้ 1 แทนคุณลักษณะชื่อ และ 2 แทนคุณลักษณะที่อยู่ ดังนั้นเมื่อเมท็อดได้ค่า 1 จะส่งค่าของชื่อเป็นสายอักขระกลับไปให้ เป็นต้น

25. GetString(in name : std::string) : std::string

เป็นเมท็อดต้องคืนค่าสายอักขระซึ่งเป็นค่าของคุณลักษณะตามชื่อที่กำหนด เช่น คลาสบุคคลมีคุณลักษณะ 2 คุณลักษณะ คือ ชื่อและที่อยู่ อาจกำหนดให้ name แทนคุณลักษณะชื่อ และ addr แทนคุณลักษณะที่อยู่ ดังนั้นเมื่อเมท็อดรับค่า name จะส่งค่าของชื่อเป็นสายอักขระกลับไปให้ เป็นต้น

26. GetTableName() : std::string

เป็นเมท็อดคืนค่าชื่อของตารางความสัมพันธ์ที่จัดเก็บวัตถุนี้

27. GetFieldInfo(in index : int) : ObjectField

เป็นเมท็อดที่รับค่าดัชนีและคืนค่าวัตถุซึ่งบอกว่ารายละเอียดเขตข้อมูลตามดัชนีมีชื่อและชนิด เช่น คลาสบุคคลมีคุณลักษณะ 2 คุณลักษณะ คือ ชื่อและที่อยู่ อาจกำหนดให้ 1 แทนคุณลักษณะชื่อ ซึ่งมีชื่อเขตข้อมูลในตารางความสัมพันธ์ว่า name และชนิดข้อมูลเป็นตัวอักษร เป็นต้น

28. GetFieldInfo(in name : std::string) : ObjectField

เป็นเมท็อดที่รับค่าชื่อและคืนค่าวัตถุซึ่งบอกรายละเอียดเขตข้อมูลตามดัชนีมีชื่อและชนิด เช่น คลาสบุคคลมีคุณลักษณะ 2 คุณลักษณะ คือ ชื่อและที่อยู่ อาจกำหนดให้ name แทนคุณลักษณะชื่อ ซึ่งมีชื่อเขตข้อมูลในตารางความสัมพันธ์ว่า name และชนิดข้อมูลเป็นตัวอักษร เป็นต้น

29. GetFieldCount() : int

เป็นเมท็อดคืนค่าจำนวนเขตข้อมูลที่อยู่ตารางความสัมพันธ์ของคลาส

30. NewObject() : DbObject

เป็นเมท็อดที่สร้างวัตถุใหม่แล้วคืนค่าวัตถุใหม่กลับมา

31. SetObjectValue(in index : int, in v : std::string) : void

เป็นเมท็อดในการกำหนดค่าให้วัตถุโดยที่คุณลักษณะแต่ละคุณลักษณะด้วยค่าบ่งชี้ และรับค่าที่ต้องการกำหนดให้วัตถุ

32. SelectParent() : bool

เป็นเมท็อดซึ่งทำการเรียกเพื่อนำดึงค่าคุณลักษณะของวัตถุแม่มากำหนดให้วัตถุ

33. SaveParent() : bool

เป็นเมท็อดซึ่งเรียกเพื่อทำการบันทึกส่วนของวัตถุแม่ในวัตถุลูกไปยังตารางความสัมพันธ์ของแม่ เช่น คลาสลูกค้าสืบทอดจากคลาสบุคคล เมท็อดจะทำการบันทึกส่วน

ของคุณลักษณะของคลาสบุคคลซึ่งอยู่ในคลาสลูกค้าไปยังตารางความสัมพันธ์ของวัตถุบุคคลเป็นต้น

34. CloneMemberValue(in other : DbObject) : void

เป็นเมท็อดทำการกำหนดค่าคุณลักษณะของวัตถุให้เหมือนกับวัตถุที่ส่งเข้ามา

35. SelectComponent() : bool

เป็นเมท็อดทำหน้าที่เลือกวัตถุคอมโพเนนท์ของวัตถุเมื่อวัตถุนั้นมีวัตถุคอมโพเนนท์

36. SaveComponent() : bool

เป็นเมท็อดทำการบันทึกวัตถุคอมโพเนนท์ของวัตถุ

4.5.1.4. คลาส Record

เป็นคลาสเพื่อสร้าง ซึ่งทำหน้าที่เก็บชุดของวัตถุ Column ของระเบียนหนึ่งในฐานข้อมูลเชิงสัมพันธ์ซึ่งใช้ในโครงร่างฯ โดยคลาสดังกล่าวประกอบด้วยเมท็อดดังนี้

1. GetColumn(in index : int) : Column

เป็นเมท็อดคืนค่าวัตถุ Column ตามค่าดัชนี

2. AddColumn(in pColumn : Column) : void

เป็นเมท็อดเพิ่มวัตถุ Column ลงไปในวัตถุ

3. GetCount() : long

เป็นเมท็อดที่คืนค่าจำนวนวัตถุ Column ซึ่งอยู่ในวัตถุ

4. DeleteAllColumn() : void

เป็นเมท็อดทำการลบวัตถุ Column ที่อยู่ในวัตถุทั้งหมด

4.5.1.5. คลาส Column

เป็นคลาสเพื่อสร้าง ซึ่งทำหน้าที่เก็บค่าของคุณลักษณะใดๆของวัตถุซึ่งใช้ในโครงร่างฯ โดยคลาสดังกล่าวประกอบด้วยเมท็อดดังนี้

1. SetValue(in value : std::string) : void

เป็นเมท็อดในการกำหนดค่าให้วัตถุ

2. GetValue() : std::string

เป็นเมท็อดคืนค่าค่าในวัตถุ

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

4.5.1.6. คลาส ObjectField

เป็นคลาสเพื่อสร้าง ซึ่งทำหน้าที่เก็บรายละเอียดของคุณลักษณะใดๆของวัตถุ เช่น ชื่อเขตข้อมูล (Field) ซึ่งเก็บค่าของคุณลักษณะ และชนิดของข้อมูล เช่น ตัวเลข, ตัวอักษร เป็นต้น ซึ่งใช้ในโครงร่างฯ โดยคลาสดังกล่าวประกอบด้วยเมท็อดดังนี้

1. SetName(in name : std::string) : void

   เป็นเมท็อดใช้กำหนดชื่อเขตข้อมูล

2. GetName() : std::string

   เป็นเมท็อดคืนค่าชื่อเขตข้อมูล

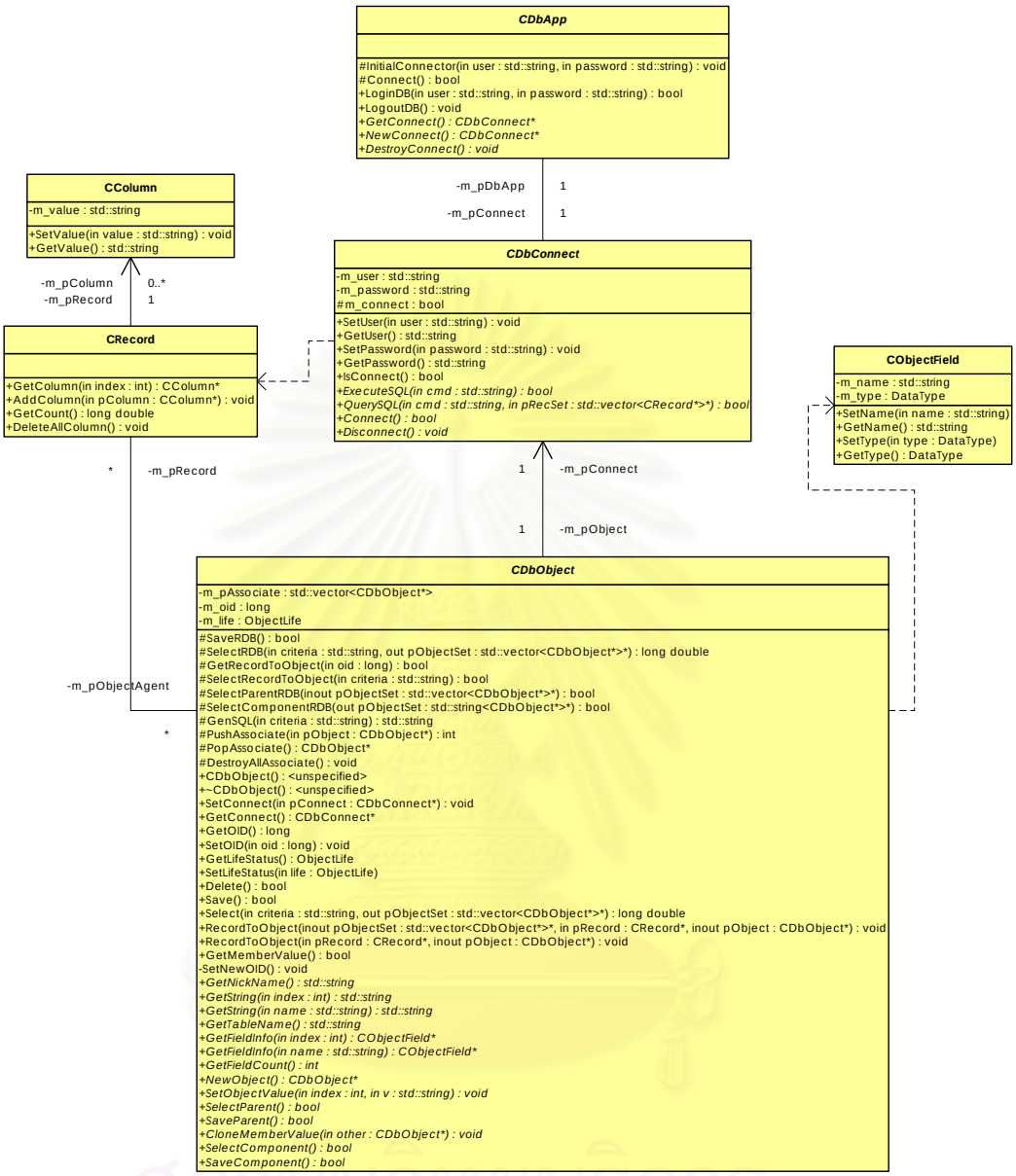
3. SetType(in type : DataType) : void

   เป็นเมท็อดกำหนดชนิดข้อมูล

4. GetType() : DataType

   เป็นเมท็อดคืนค่าชนิดข้อมูล

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย



รูปที่ 4.8 แผนภาพคลาสของโครงร่างฯ

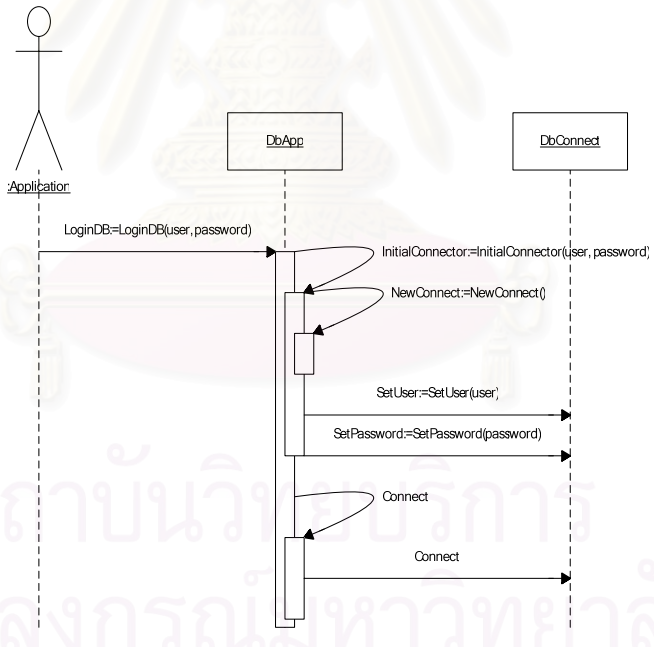
สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

4.6. การทำงานโดยรวมของโครงร่างฯ

แผนภาพขั้นตอนตามหน้าที่ของโครงร่างฯ โดยสังเขป

4.6.1. แผนภาพขั้นตอนการเชื่อมต่อฐานข้อมูลเชิงสัมพันธ์

ตามรูปที่ 4.9 อธิบายขั้นตอนการเชื่อมต่อ จากรูปแสดงให้เห็นว่า โปรแกรมประยุกต์เริ่มด้วยการเรียกเมท็อด LoginDB พร้อมชื่อผู้ใช้และรหัสผ่าน เพื่อขอเชื่อมต่อกับฐานข้อมูลเชิงสัมพันธ์ หากเชื่อมต่อสำเร็จจะส่งค่าจริงกลับมา แต่ภายในโครงร่างฯ จะทำงานด้วยการเรียกเมท็อด LoginDB จะเรียกเมท็อด InitialConnector เพื่อกำหนดค่าเริ่มต้นก่อนการเชื่อมต่อ ซึ่งภายใต้เมท็อด InitialConnector จะทำการเรียก NewConnect เพื่อขอสร้างวัตถุเชื่อมต่อและทำการกำหนดค่าผู้ใช้และรหัสผ่านให้กับวัตถุเชื่อมต่อที่สร้างใหม่ จากนั้นการเรียกเมท็อด Connect ทำการเชื่อมไปยังฐานข้อมูล หากสำเร็จจะคืนค่าเป็นจริงกลับมา



รูปที่ 4.9 แผนภาพขั้นตอนการเชื่อมต่อฐานข้อมูลเชิงสัมพันธ์โดยใช้โครงร่างฯ

4.6.2. แผนภาพขั้นตอนการการเพิ่มวัตถุในฐานข้อมูลเชิงสัมพันธ์

ตามรูปที่ 4.10 แสดงการเพิ่มวัตถุ จากรูปโปรแกรมประยุกต์จะต้องทำการสร้างวัตถุที่ต้องการจัดเก็บใหม่ซึ่งโครงร่างฯจะกำหนดให้วัตถุมีสถานะชีวิตเป็นเกิดใหม่และสิ่งที่โครง

ร่างฯกำหนดต้องทำการกำหนดวัตถุเชื่อมต่อซึ่งถูกใช้ในโปรแกรมประยุกต์ขณะนั้นให้กับวัตถุด้วย จากนั้นทำการกำหนดค่าคุณลักษณะต่างๆ ก่อนการจัดเก็บเพื่อที่โครงร่างฯจะได้จัดเก็บค่าของคุณลักษณะของวัตถุนั้น เมื่อต้องการจัดเก็บเพียงเรียกเมท็อด Save โครงร่างฯจะต้องทำการตามกลไกภายใต้โครงร่างฯ โดยหลักคือโครงร่างฯจะทำการสร้างประโยคคำสั่งเอสคิวแอลแทรก (Insert SQL command) ซึ่งเป็นหน้าที่ของคลาส DbObject โดยพิจารณาว่าวัตถุมีความสัมพันธ์แบบใดแบบซิงเกิลคลาสจะทำการบันทึกวัตถุลงตารางความสัมพันธ์เพียงตารางเดียว แบบการรับทอดคลาส โครงร่างฯ จะเรียกเมท็อดซึ่งนักพัฒนาโปรแกรมประยุกต์จัดเตรียมไว้ให้ว่ามีความสัมพันธ์กับคลาสใด และทำการบันทึกวัตถุของคลาสแม่และสายการการรับทอดจนกระทั่งทำการบันทึกวัตถุตนเองลงฐานข้อมูลเชิงสัมพันธ์ แบบภาพรวมกลุ่มคลาส โครงร่างฯ จะทำการบันทึกเฉพาะวัตถุตนเองเท่านั้น เนื่องจากวัตถุที่รวมกลุ่มกันได้มีการบันทึกก่อนการใช้ และแบบคอมโพสิตชันคลาส การสร้างวัตถุใหม่คลาสที่คอมโพสิตอยู่จะถูกสร้างขึ้นด้วย และการจัดเก็บวัตถุ โครงร่างฯ จะทำการจัดเก็บวัตถุคอมโพสิตด้วยและจึงจัดเก็บวัตถุตนเอง จากรูปเมท็อด Save จะทำการเรียกเมท็อด SaveParent เพื่อทำการบันทึกวัตถุแม่ก่อน จากนั้นทำการเรียกเมท็อด SaveRDB โดยหลักจะทำการสร้างประโยคเอสคิวแอลจากการเรียกเมท็อด GetTableName เพื่อขอชื่อตารางความสัมพันธ์ที่เก็บวัตถุ เรียกเมท็อด GetFieldCount เพื่อขอจำนวนเขตข้อมูลเพื่อใช้ในการวนรอบในการสร้างประโยคเอสคิวแอล เรียกเมท็อด GetFieldInfo เพื่อขอรายละเอียดขอแต่ละเขตข้อมูลว่ามีชื่อและชนิดข้อมูลเป็นอย่างไร และเรียกเมท็อด GetString เพื่อขอค่าของคุณลักษณะ ทั้งหมดใช้เพื่อสร้างประโยคเอสคิวแอลแล้วส่งไปประมวลผลที่ฐานข้อมูลเชิงสัมพันธ์ผ่านวัตถุเชื่อมต่อ

4.6.3. แผนภาพขั้นตอนการปรับปรุงวัตถุในฐานข้อมูลเชิงสัมพันธ์

แผนภาพการปรับปรุงนั้นจะประกอบด้วยส่วนขั้นตอนการเลือกวัตถุด้วย เนื่องจากการปรับปรุงวัตถุใด วัตถุนั้นต้องจัดเก็บไว้ก่อนหน้า ตามรูปที่ 4.11 แสดงการปรับปรุงวัตถุโปรแกรมประยุกต์ทำการสร้างวัตถุชนิดเดียวกับที่ต้องการปรับปรุง โดยโครงร่างฯ กำหนดสถานะชีวิตเป็นวัตถุสร้างใหม่ จากนั้นทำการกำหนดวัตถุการเชื่อมต่อและทำการเรียกเมท็อด Select เพื่อทำการเรียกวัตถุที่ต้องการ โดยส่งเงื่อนไขที่ต้องการเลือก ภายในเมท็อดนี้จะทำการเรียกเมท็อด SelectRDB ภายในเมท็อดทำการเรียกเมท็อด GenSQL เพื่อสร้างประโยคคำสั่งเอสคิวแอล และส่งประโยคคำสั่งเอสคิวแอลไปประมวลผลผ่านเมท็อด QuerySQL ซึ่งจะได้ชุดของระเบียบซึ่งเก็บค่าของคุณลักษณะของวัตถุที่ต้องการเลือก จากนั้นโครงร่างฯจะทำการเรียกเมท็อด RecordToObject ซึ่งทำหน้าที่ในการนำค่าคุณลักษณะมากำหนดให้วัตถุ จากนั้นทำการเรียกเมท็อด SelectComponentRDB ซึ่งทำหน้าที่ในการเรียกเมท็อด SelectComponent ของวัตถุแต่ละวัตถุในชุดของวัตถุที่ส่งเข้ามา ขั้นต่อไปทำการเรียกเมท็อด SelectParentRDB ภายในจะทำการเรียกเมท็อด

SelectParent ของวัตถุแต่ละวัตถุในชุดของวัตถุที่ส่งเข้ามา เป็นการสิ้นสุดขั้นตอนการเลือกวัตถุ ซึ่งในขั้นตอนการเลือกวัตถุจะเปลี่ยนสถานะชีวิตวัตถุเป็นมีชีวิตจากที่มีสถานะชีวิตเป็นเกิดใหม่ในตอนสร้างวัตถุ ซึ่งเป็นสิ่งที่บอกให้โครงร่างฯทราบว่าเป็นการปรับปรุงวัตถุในขณะที่เรียกเมท็อด Save เมื่อต้องการจัดเก็บเพียงเรียกเมท็อด Save โครงร่างฯจะต้องทำการตามกลไกภายใต้โครงร่างฯ โดยหลักคือโครงร่างฯจะทำการสร้างประโยคคำสั่งเอสคิวแอลปรับปรุง (Update SQL command) ซึ่งเป็นหน้าที่ของคลาส DbObject โดยพิจารณาว่าวัตถุมีความสัมพันธ์แบบใด แบบซิงเกิลคลาสจะทำการปรับปรุงวัตถุลงตารางความสัมพันธ์เพียงตารางเดียว แบบการรับทอดคลาส โครงร่างฯ จะเรียกเมท็อดซึ่งนักพัฒนาโปรแกรมประยุกต์จัดเตรียมไว้ให้ว่ามีความสัมพันธ์กับคลาสใด และทำการปรับปรุงวัตถุของคลาสแม่และสายการการรับทอดจนกระทั่งทำการปรับปรุงวัตถุตนเองในฐานข้อมูลเชิงสัมพันธ์ แบบภาพรวมกลุ่มคลาส โปรแกรมเมอร์จะระบุให้โครงร่างฯ ทำการปรับปรุงวัตถุโดยวัตถุที่รวมกันจะปรับปรุงคุณลักษณะที่หมายเลขวัตถุที่รวมกลุ่มของวัตถุ และแบบคอมโพสิตชันคลาส โปรแกรมเมอร์ต้องออกแบบให้มีเมท็อดเพื่อทำการเรียกเมท็อดเพื่อการปรับปรุงวัตถุคอมโพสิตได้ จากรูปเมท็อด Save จะทำการเรียกเมท็อด SaveParent เพื่อทำการบันทึกวัตถุแม่ก่อน จากนั้นทำการเรียกเมท็อด SaveRDB โดยหลักจะทำการสร้างประโยคเอสคิวแอลจากการเรียกเมท็อด GetTableName เพื่อขอชื่อตารางความสัมพันธ์ที่เก็บวัตถุ เรียกเมท็อด GetFieldCount เพื่อดูว่ามีกี่เขตข้อมูลเพื่อใช้ในการวนรอบในการสร้างประโยคเอสคิวแอล เรียกเมท็อด GetFieldInfo เพื่อขอรายละเอียดขอแต่ละเขตข้อมูลว่ามีชื่อและชนิดข้อมูลเป็นอย่างไร และเรียกเมท็อด GetString เพื่อขอค่าของคุณลักษณะ ทั้งหมดใช้เพื่อสร้างประโยคเอสคิวแอลแล้วส่งไปประมวลผลที่ฐานข้อมูลเชิงสัมพันธ์ผ่านวัตถุเชื่อมต่อ ต่อจากนั้นโครงร่างฯเรียกเมท็อด SaveComponent เพื่อทำกระทำกับวัตถุคอมโพเนนท์

4.6.4. แผนภาพขั้นตอนการเลือกวัตถุในฐานข้อมูลเชิงสัมพันธ์

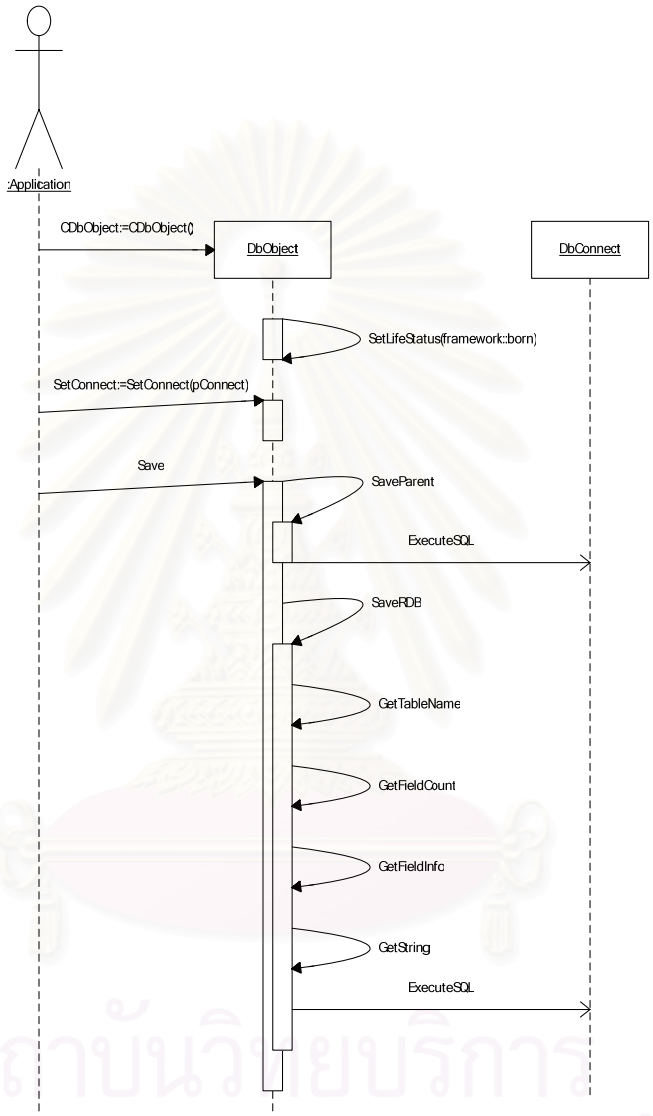
ตามรูปที่ 4.12 โปรแกรมประยุกต์ทำการสร้างวัตถุชนิดเดียวกับที่ต้องการเลือกซึ่งโครงร่างฯ จะกำหนดเป็นวัตถุสร้างใหม่ จากนั้นทำการกำหนดวัตถุการเชื่อมต่อและทำการเรียกเมท็อด Select เพื่อทำการเรียกวัตถุที่ต้องการโดยส่งเงื่อนไขที่ต้องการเลือก ภายในเมท็อดนี้จะทำการเรียกเมท็อด SelectRDB ภายในเมท็อดทำการเรียกเมท็อด GenSQL เพื่อสร้างประโยคคำสั่งเอสคิวแอล และส่งประโยคคำสั่งเอสคิวแอลไปประมวลผลผ่านเมท็อด QuerySQL ซึ่งจะได้ชุดของระเบียบซึ่งเก็บค่าของคุณลักษณะของวัตถุที่ต้องการเลือก จากนั้นโครงร่างฯจะทำการเรียกเมท็อด RecordToObject ซึ่งทำหน้าที่ในการนำค่าคุณลักษณะมากำหนดให้วัตถุ จากนั้นทำการเรียกเมท็อด SelectComponentRDB ซึ่งทำหน้าที่ในการเรียกเมท็อด SelectComponent ของวัตถุแต่ละวัตถุในชุดของวัตถุที่ส่งเข้ามา ขั้นต่อไปทำการเรียกเมท็อด SelectParentRDB ภายในจะทำการเรียกเมท็อด

SelectParent ของวัตถุแต่ละวัตถุในชุดของวัตถุที่ส่งเข้ามา เป็นการสิ้นสุดขั้นตอนการเลือกวัตถุ ซึ่งในขั้นตอนการเลือกวัตถุจะเปลี่ยนสถานะชีวิตวัตถุเป็นมีชีวิตจากที่มีสถานะชีวิตเป็นเกิดใหม่ในตอนสร้างวัตถุ

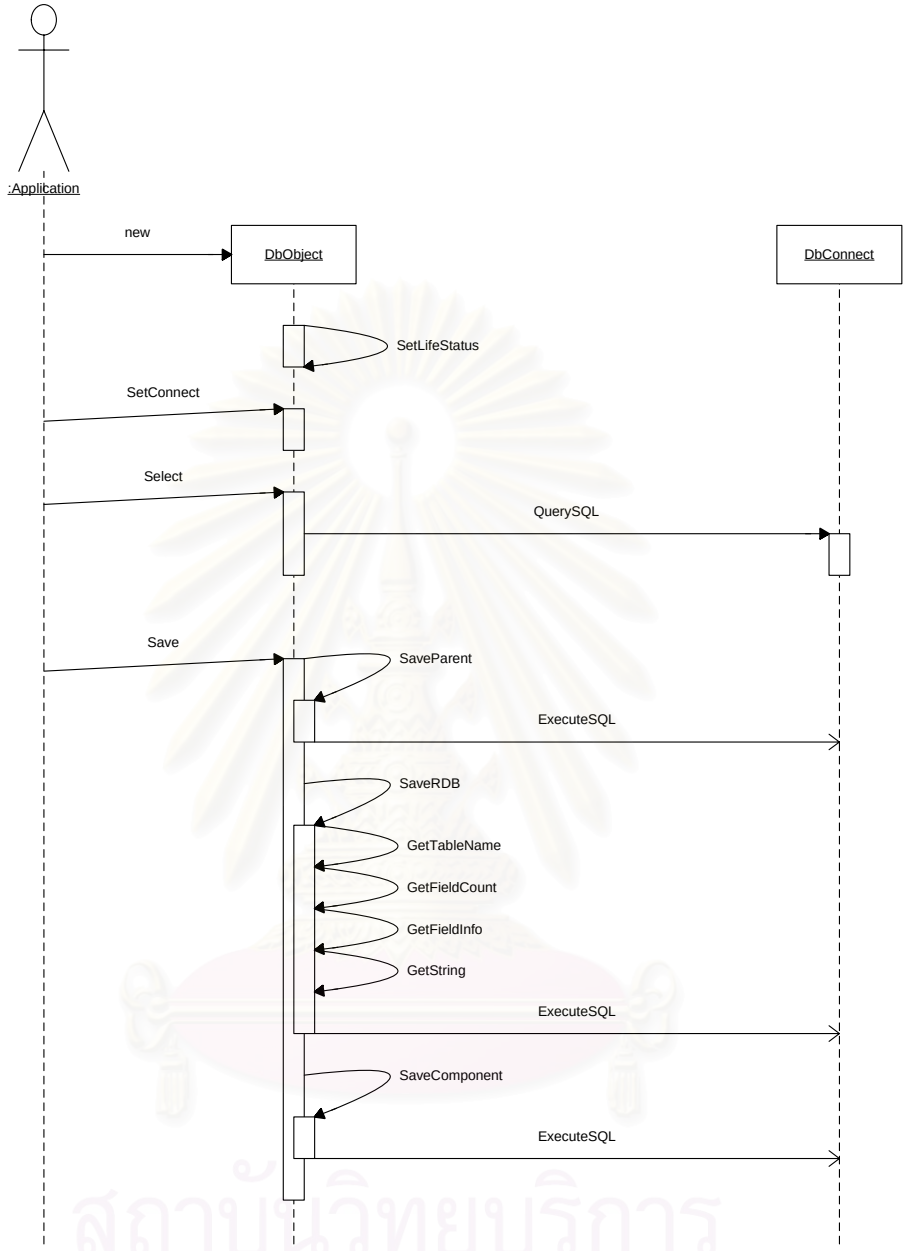
4.6.5. แผนภาพขั้นตอนการลบวัตถุในฐานข้อมูลเชิงสัมพันธ์

แผนภาพการลบนั้นจะประกอบด้วยส่วนขั้นตอนการเลือกวัตถุด้วย เนื่องจากการลบวัตถุใด วัตถุนั้นต้องจัดเก็บไว้ก่อนหน้า ตามรูปที่ 4.13 แสดงการลบวัตถุ โปรแกรมประยุกต์ทำการสร้างวัตถุชนิดเดียวกับที่ต้องการลบซึ่งโครงร่างฯ จะกำหนดเป็นวัตถุสร้างใหม่ จากนั้นทำการกำหนดวัตถุการเชื่อมต่อและทำการเรียกเมท็อด Select เพื่อทำการเรียกวัตถุที่ต้องการโดยส่งเงื่อนไขที่ต้องการเลือก ภายในเมท็อดนี้จะทำการเรียกเมท็อด SelectRDB ภายในเมท็อดทำการเรียกเมท็อด GenSQL เพื่อสร้างประโยคคำสั่งเอสคิวแอล และส่งประโยคคำสั่งเอสคิวแอลไปประมวลผลผ่านเมท็อด QuerySQL ซึ่งจะได้ชุดของระเบียบซึ่งเก็บค่าของคุณลักษณะของวัตถุที่ต้องการเลือก จากนั้นโครงร่างฯจะทำการเรียกเมท็อด RecordToObject ซึ่งทำหน้าที่ในการนำค่าคุณลักษณะมากำหนดให้วัตถุ จากนั้นทำการเรียกเมท็อด SelectComponentRDB ซึ่งทำหน้าที่ในการเรียกเมท็อด SelectComponent ของวัตถุแต่ละวัตถุในชุดของวัตถุที่ส่งเข้ามา ขั้นต่อไปทำการเรียกเมท็อด SelectParentRDB ภายในจะทำการเรียกเมท็อด SelectParent ของวัตถุแต่ละวัตถุในชุดของวัตถุที่ส่งเข้ามา เป็นการสิ้นสุดขั้นตอนการเลือกวัตถุ ซึ่งในขั้นตอนการเลือกวัตถุจะเปลี่ยนสถานะชีวิตวัตถุเป็นมีชีวิตจากที่มีสถานะชีวิตเป็นเกิดใหม่ในตอนสร้างวัตถุ ขั้นตอนนี้เป็นขั้นตอนที่ต่างจากการปรับปรุงวัตถุคือทำการเรียกเมท็อด Delete จากนั้นทำการเรียกเมท็อด Save เมื่อต้องการลบเพียงเรียกเมท็อด Save โครงร่างฯจะต้องทำการตามกลไกภายใต้โครงร่างฯ โดยหลักคือโครงร่างฯจะทำการสร้างประโยคคำสั่งเอสคิวแอลลบ (Delete SQL commnad) ซึ่งเป็นหน้าที่ของคลาส DbObject โดยพิจารณาว่าวัตถุมีความสัมพันธ์แบบใด แบบซิงเกิลคลาสจะทำการลบวัตถุในตารางความสัมพันธ์เพียงตารางเดียว แบบการรับทอดคลาส โครงร่างฯ จะเรียกเมท็อดซึ่งนักพัฒนาโปรแกรมประยุกต์จัดเตรียมไว้ให้ว่ามีความสัมพันธ์กับคลาสใด และทำการลบวัตถุของคลาสแม่และสายการการรับทอดจนกระทั่งทำการลบวัตถุตนเองในฐานข้อมูลเชิงสัมพันธ์ แบบภาพรวมกลุ่มคลาส โครงร่างฯ จะไม่ทำการลบวัตถุที่รวมกลุ่ม และแบบคอมโพสิตชันคลาสโครงร่างฯ จะทำการลบวัตถุคอมโพสิต แล้วจึงลบวัตถุตนเอง จากรูปเมท็อด Save จะทำการเรียกเมท็อด SaveComponent เพื่อทำการลบวัตถุองค์ประกอบก่อน จากนั้นทำการเรียกเมท็อด SaveRDB โดยหลักจะทำการสร้างประโยคเอสคิวแอลจากการเรียกเมท็อด GetTableName เพื่อขอชื่อตารางความสัมพันธ์ที่เก็บวัตถุ เรียกเมท็อด GetFieldCount เพื่อดูว่ามีกี่เขตข้อมูลเพื่อใช้ในการวนรอบในการสร้างประโยคเอสคิวแอล เรียกเมท็อด GetFieldInfo เพื่อขอรายละเอียดขอแต่ละเขตข้อมูลว่ามี

ชื่อและชนิดข้อมูลเป็นอย่างไร และเรียกเมท็อด GetString เพื่อขอค่าของคุณลักษณะ ทั้งหมดใช้เพื่อสร้างประโยคเอสคิวแอลแล้วส่งไปประมวลผลที่ฐานข้อมูลเชิงสัมพันธ์ผ่านวัตถุเชื่อมต่อ ต่อจากนั้นโครงร่างฯ เรียกเมท็อด SaveParent เพื่อลบวัตถุแม่

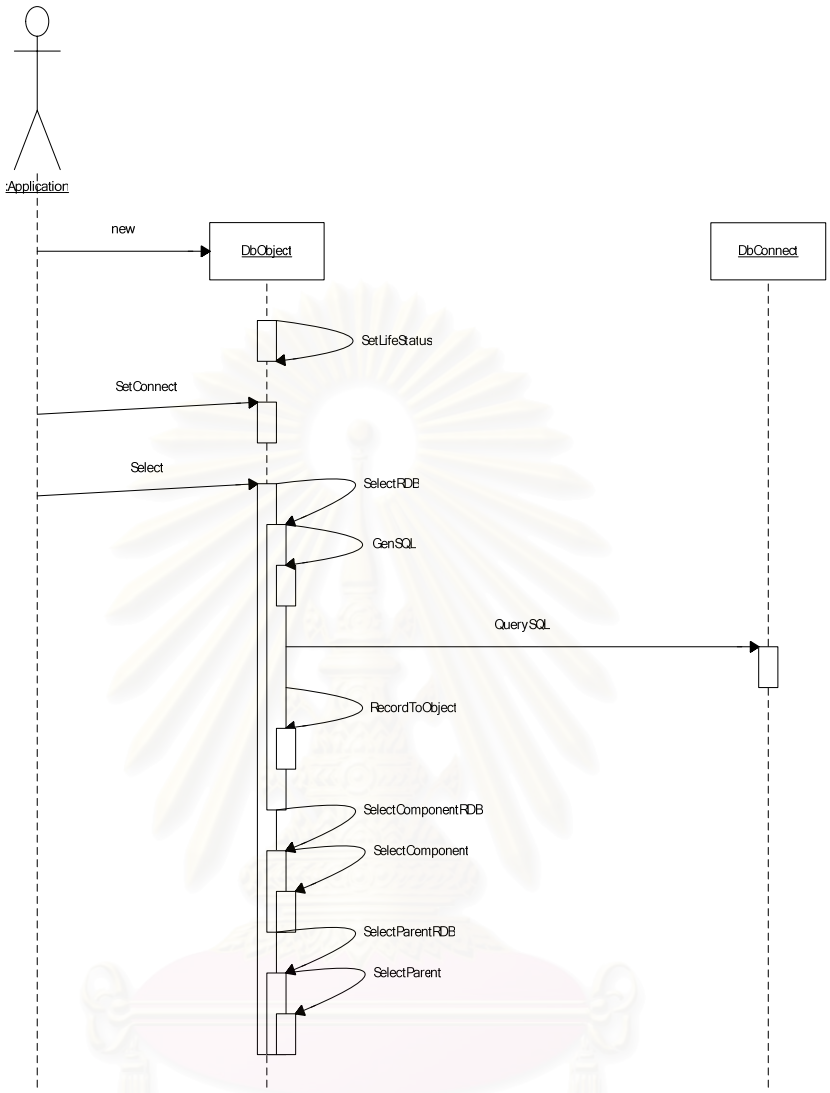


รูปที่ 4.10 แผนภาพขั้นตอนการเพิ่มวัตถุในฐานข้อมูลเชิงสัมพันธ์โดยใช้โครงร่างฯ



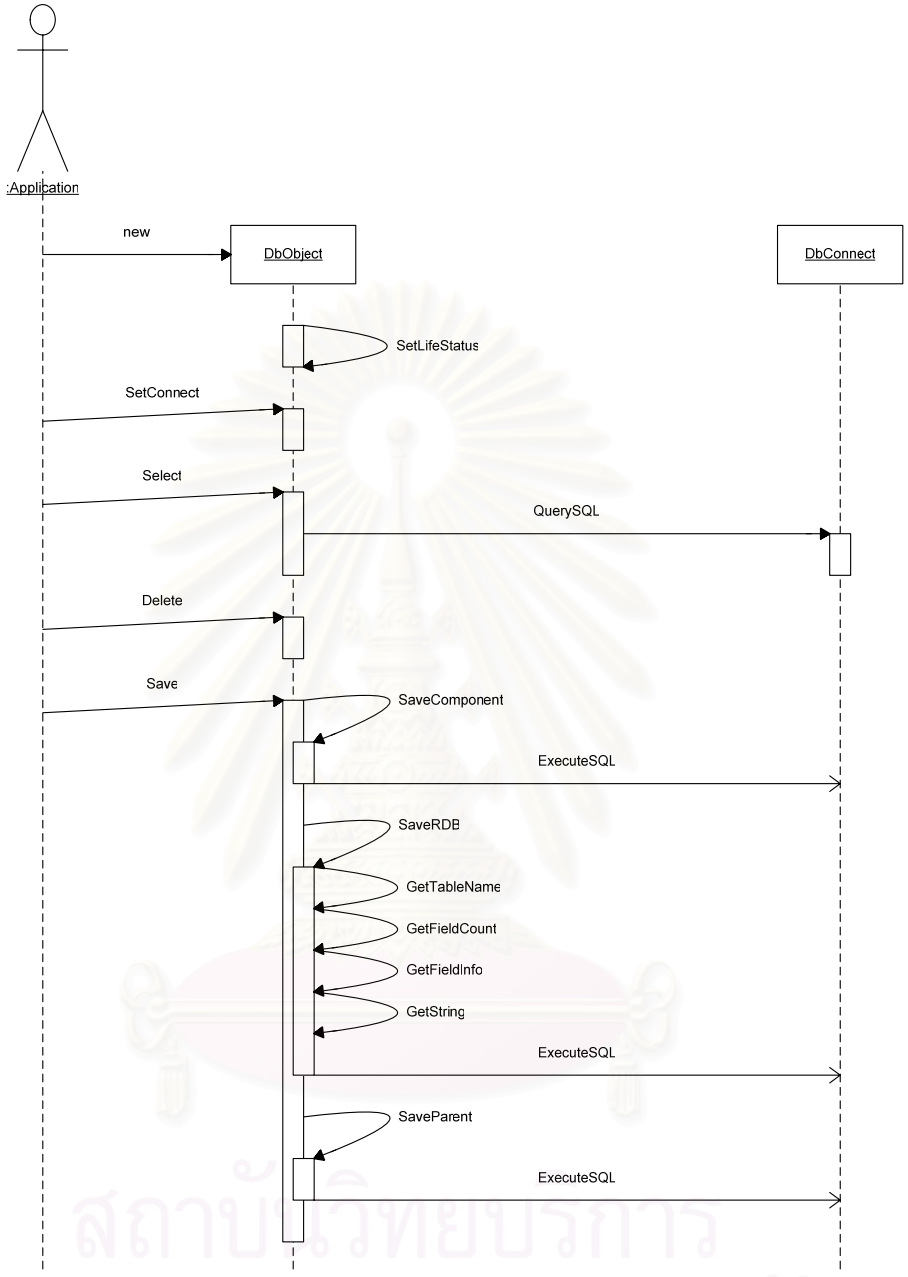
รูปที่ 4.11 แผนภาพขั้นตอนการปรับปรุงวัตถุในฐานข้อมูลเชิงสัมพันธ์โดยใช้โครงร่างฯ

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย



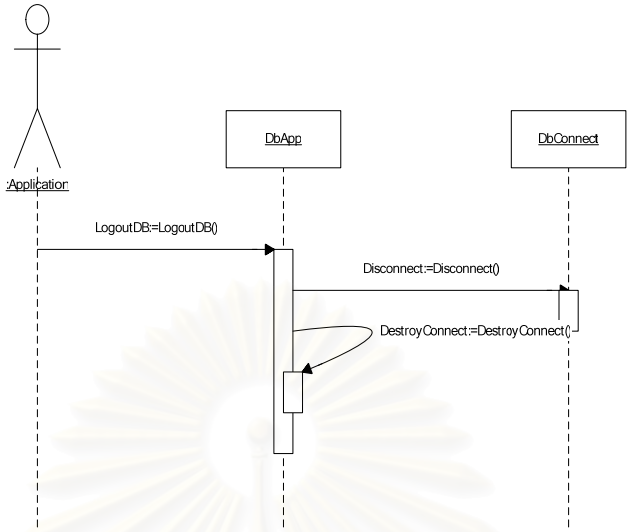
รูปที่ 4.12 แผนภาพขั้นตอนการเลือกวัตถุในฐานข้อมูลเชิงสัมพันธ์โดยใช้โครงร่างฯ

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย



รูปที่ 4.13 แผนภาพขั้นตอนการลบวัตถุในฐานข้อมูลเชิงสัมพันธ์โดยใช้โครงร่างฯ

รูปที่ 4.14 แสดงการยกเลิกการเชื่อมต่อฐานข้อมูลเชิงสัมพันธ์ เมื่อโปรแกรมประยุกต์ต้องการยกเลิกการเชื่อมต่อ โปรแกรมประยุกต์ทำเรียกเมท็อด LogoutDB เพื่อออกจากระบบฐานข้อมูลเชิงสัมพันธ์ ภายในเมท็อดทำการเรียกเมท็อด Disconnect ของวัตถุเชื่อมต่อ และเรียกเมท็อด DestroyConnect เพื่อทำลายวัตถุเชื่อมต่อ



รูปที่ 4.14 แผนภาพขั้นตอนการตัดการเชื่อมต่อฐานข้อมูลเชิงสัมพันธ์โดยใช้โครงร่างฯ

4.7. การใช้งานโครงร่างฯ สำหรับนักพัฒนาเพื่อใช้งานจริง

การใช้งานโครงร่างฯสำหรับนักพัฒนาเพื่อใช้งานจริง (Application Framework Implementer) โดยนำการโครงร่างฯ ที่ออกแบบจากนักออกแบบโครงร่างโปรแกรมประยุกต์เชิงวัตถุ มาสร้างให้สามารถใช้งานได้ให้เหมาะกับสภาพแวดล้อมในการพัฒนาโปรแกรมประยุกต์ เช่น จะสร้างโครงร่างฯ จากต้นแบบโครงร่างฯ สำหรับภาษา C++ กับมาตรฐานโอดีบีซีเพื่อให้โปรแกรมเมอร์ภาษา C++ ซึ่งใช้ฐานข้อมูลเชิงสัมพันธ์ ผ่านมาตรฐานโอดีบีซิได้ในไปใช้งาน เป็นต้น โดยขั้นตอนในการสร้างเป็นดังนี้

1. สร้างคลาสใหม่ซึ่งการรับทอดมาจาก DbConnect จากนั้นการใส่รหัสคำสั่งในเมท็อดดังนี้
   - เมท็อด ExcuteSQL

     โปรแกรมเมอร์ต้องกำหนดให้เมท็อดรับประโยคคำสั่งเอสคิวแอลมาและเขียนโปรแกรมเพื่อส่งให้กับฐานข้อมูลเชิงสัมพันธ์ประมวลผล และส่งผลค่าจริงเมื่อสำเร็จ หากไม่สำเร็จส่งค่าเท็จ ซึ่งการเขียนโปรแกรมส่วนนี้ใช้ประสบการณ์ของนักสร้างเพื่อใส่คำสั่งในการส่งประโยคคำสั่งเอสคิวแอลเพื่อจัดการข้อมูลไปประมวลผล
   - เมท็อด QuerySQL

     โปรแกรมเมอร์ต้องกำหนดให้เมท็อดรับประโยคคำสั่งเอสคิวแอลมาและเขียนโปรแกรมเพื่อส่งให้กับฐานข้อมูลเชิงสัมพันธ์ประมวลผล และส่งผลค่าจริงเมื่อสำเร็จ หากไม่สำเร็จส่งค่าเท็จ และหากสำเร็จจะส่งผลของชุดวัตถุที่เลือกได้ โดยให้นำวัตถุที่เลือกได้ใส่ใน

โครงสร้างข้อมูลแบบชุดตัวแปร ซึ่งการเขียนโปรแกรมส่วนนี้ใช้ประสบการณ์ของนักสร้างเพื่อใส่คำสั่งในการส่งประโยคคำสั่งเอสคิวแอลเพื่อเลือกไปประมวลผล

– เมท็อด Connect

โปรแกรมเมอร์ต้องกำหนดให้เมท็อดทำการเชื่อมต่อไปยังฐานข้อมูลเชิงสัมพันธ์เขียนคำสั่งเพื่อส่งให้กับฐานข้อมูลเชิงสัมพันธ์ประมวลผล และส่งผลค่าจริงเมื่อสำเร็จ หากไม่สำเร็จส่งค่าเท็จ ซึ่งการเขียนโปรแกรมส่วนนี้ใช้ประสบการณ์ของนักสร้างเพื่อใส่คำสั่งให้เชื่อมไปยังฐานข้อมูลเชิงสัมพันธ์

– เมท็อด Disconnect

โปรแกรมเมอร์ต้องกำหนดให้เมท็อดทำการยกเลิกการเชื่อมต่อไปยังฐานข้อมูลเชิงสัมพันธ์เขียนคำสั่งเพื่อส่งให้กับฐานข้อมูลเชิงสัมพันธ์ประมวลผล และส่งผลค่าจริงเมื่อสำเร็จ หากไม่สำเร็จส่งค่าเท็จ ซึ่งการเขียนโปรแกรมส่วนนี้ใช้ประสบการณ์ของนักสร้างเพื่อใส่คำสั่งให้เชื่อมไปยังฐานข้อมูลเชิงสัมพันธ์

– เมท็อดอื่นเพื่อความเหมาะสมกับการเชื่อมต่อ

ซึ่งอาจเป็นเมท็อดที่นักสร้างออกแบบเพิ่มเติมเพื่อให้ใช้ได้กับฐานข้อมูลเชิงสัมพันธ์ที่เลือกใช้ หรือวิธีการเชื่อมต่อ

2. สร้างคลาสใหม่ซึ่งการรับทอดมาจาก DbApp จากนั้นเขียนโปรแกรมในเมท็อดดังนี้

– เมท็อด GetConnect

โปรแกรมเมอร์ต้องกำหนดให้เมท็อดคืนค่าวัตถุเชื่อมต่อซึ่งเป็นชนิดเดียวกับที่ใช้ในโปรแกรมประยุกต์ขณะนั้นซึ่งวัตถุของคลาสซึ่งการรับทอดมาจาก DbConnect

– เมท็อด NewConnect

โปรแกรมเมอร์ต้องกำหนดให้เมท็อดสร้างวัตถุเชื่อมต่อชนิดเดียวกับที่ใช้ในโปรแกรมประยุกต์ซึ่งวัตถุของคลาสซึ่งการรับทอดมาจาก DbConnect

– เมท็อด DestroyConnect

โปรแกรมเมอร์ต้องกำหนดให้เมท็อดสำหรับการทำลายการเชื่อมต่อ

4.8. การใช้งานโครงร่างฯ สำหรับโปรแกรมเมอร์

การใช้งานโครงร่างฯ สำหรับโปรแกรมเมอร์ (Application Framework User) จะมีคลาสที่ใช้จากโครงร่างฯ โดยตรงอยู่เพียงหนึ่งคลาสเท่านั้น คือ คลาส DbObject ส่วนอีก 2 คลาส ซึ่งเกิดจากนักสร้างโครงร่างฯ คือ คลาส DbApp และคลาส DbConnect โปรแกรมเมอร์จะพิจารณาว่าจะคลาสทั้งสองว่านักสร้างใดสร้างมาให้เหมาะกับการใช้งาน เช่น นักสร้างโครงร่างฯ สร้างโครงร่างฯ สำหรับภาษา C++ ผ่านมาตรฐานโอดีบีซีหากโปรแกรมเมอร์จะพัฒนาโปรแกรมประยุกต์ด้วย

ภาษา C++ เช่นกัน แล้วติดต่อผ่านมาตรฐานโอดีบีซีเช่นกัน ก็สามารถนำโครงร่างฯ มาใช้ได้งานทันที หรือ นักสร้างโครงร่างฯ สร้างโครงร่างฯ สำหรับภาษา Java และใช้กับฐานข้อมูลเชิงสัมพันธ์โอราเคิล (Oracle) หากโปรแกรมเมอร์ต้องการพัฒนาโปรแกรมประยุกต์ด้วยภาษา Java กับฐานข้อมูลเชิงสัมพันธ์โอราเคิล เช่นกัน สามารถนำมาใช้ได้

สำหรับการใช้งานโครงร่างฯ ในส่วนคลาส DbObject ของโครงร่างฯ เป็นไปดังนี้

1. เมื่อได้แผนภาพคลาสของโปรแกรมประยุกต์แล้ว ทำการสร้างตารางความสัมพันธ์โดยหนึ่งคลาสต่อหนึ่งตาราง แล้วทำการเพิ่มเขตข้อมูล oid ในทุกตารางความสัมพันธ์เพื่อเก็บหมายเลขวัตถุ
2. พิจารณาตามขั้นตอนต่อไปนี้ ทำทีละคลาสจนหมด ซึ่งแต่ละคลาสจะต้องพิจารณาที่แผนภาพคลาสว่าคลาสที่กำลังดำเนินการอยู่นั้นมีความสัมพันธ์ใด เนื่องจากบางเมท็อดจะเขียนโปรแกรมไม่เหมือนกัน ซึ่งโครงร่างฯ มีข้อกำหนดให้โปรแกรมเมอร์ เขียนโปรแกรมให้รองรับโครงร่างฯไม่เหมือนกัน
3. ทำการเขียนโปรแกรมสำหรับคลาสนั้น โดยสืบทอดจากคลาส DbObject ของโครงร่างฯ
4. ทำการพิจารณาว่าคลาสมีความสัมพันธ์แบบใด โดยมีหลักพิจารณาดังนี้
   - ซิงเกิลคลาส คือ คลาสมีไม่มีความสัมพันธ์กับคลาสใดเลย หรือเป็นคลาสซึ่งอยู่บนสุดของสายการการรับทอด
   - การรับทอดคลาส คือ คลาสที่การรับทอดมาจากคลาสอื่น
   - ภาพรวมกลุ่มคลาส คือ คลาสซึ่งมีการอ้างถึงคลาสอื่นโดยที่คลาสที่อ้างถึงนั้นไม่มีวงจรชีวิตเดียวกับคลาส
   - คอมโพสิตชันคลาส คือ คลาสซึ่งมีคลาสอื่นประกอบโดยมีวงจรชีวิตเดียวกับคลาส
5. เมื่อทราบว่าเป็นคลาสความสัมพันธ์แบบใด เพื่อการพิจารณาว่าจะต้องทำการเขียนโปรแกรมอย่างไร ในเมท็อดซึ่งโครงร่างฯ กำหนด โดยโครงร่างฯ กำหนดเมท็อดเป็น 2 กลุ่มคือ กลุ่มเมท็อดซึ่งไม่ขึ้นกับความสัมพันธ์คลาส และกลุ่มซึ่งขึ้นอยู่ความสัมพันธ์คลาส ซึ่งสรุปได้ดังตาราง 4.1

ตารางที่ 4.1 ตารางสรุปกลุ่มเมท็อดของคลาส DbObject ตามความสัมพันธ์คลาส

เมท็อดไม่ขึ้นกับความสัมพันธ์คลาส	เมท็อดขึ้นกับความสัมพันธ์คลาส
GetTableName	SetObjectValue
GetFieldCount	SelectParent
NewObject	SaveParent
GetString(index)	SelectComponent
GetString(name)	SaveComponent
GetFieldInfo(index)	
GetFieldInfo(name)	
CloneMemberValue	

กลุ่มเมท็อดซึ่งไม่ขึ้นกับความสัมพันธ์คลาส

– เมท็อด GetTableName

โปรแกรมเมอร์ต้องกำหนดให้เมท็อดคืนค่า ชื่อตารางความสัมพันธ์ซึ่งเก็บวัตถุ

– เมท็อด GetFieldCount

โปรแกรมเมอร์ต้องกำหนดให้เมท็อดคืนค่า จำนวนคุณลักษณะที่จัดเก็บในคลาส

– เมท็อด NewObject

โปรแกรมเมอร์ต้องกำหนดให้เมท็อดสร้างวัตถุของคลาสแล้วคืนวัตถุนั้น

– เมท็อด GetString แบบรับดัชนี

โปรแกรมเมอร์ต้องกำหนดให้เมท็อดรับค่าดัชนีเพื่อตรวจสอบว่าเป็นตัวแทนของค่าคุณลักษณะใดแล้วคืนค่าของคุณลักษณะนั้นเป็นสายอักขระกลับไป ซึ่งการกำหนดค่าดัชนีตัวแทนนั้น โปรแกรมเมอร์สามารถกำหนดได้เองตามความเหมาะสม

– เมท็อด GetString แบบรับชื่อลักษณะ

โปรแกรมเมอร์ต้องกำหนดให้เมท็อดรับค่าชื่อคุณลักษณะเพื่อตรวจสอบว่าเป็นตัวแทนของค่าคุณลักษณะใดแล้วคืนค่าของคุณลักษณะนั้นเป็นสายอักขระกลับไป ซึ่งการกำหนดค่าชื่อตัวแทนนั้น โปรแกรมเมอร์สามารถกำหนดได้เองตามความเหมาะสม

– เมท็อด GetFieldInfo แบบรับดัชนี

โปรแกรมเมอร์ต้องกำหนดให้เมท็อดรับค่าดัชนีเพื่อตรวจสอบว่าเป็นตัวแทนของข้อมูลลักษณะของฟิลด์ในฐานข้อมูลใดแล้วทำการสร้างวัตถุ ObjectField แล้วกำหนดค่าว่าชื่อฟิลด์อะไร และมีชนิดข้อมูลประเภทตัวอักษรหรือตัวเลข จากนั้นคืนค่าวัตถุนั้นกลับไป ซึ่งการกำหนดค่าดัชนีตัวแทนนั้น โปรแกรมเมอร์สามารถกำหนดได้เองตามความเหมาะสม

– เมท็อด GetFieldInfo แบบรับชื่อ

โปรแกรมเมอร์ต้องกำหนดให้เมท็อดรับค่าชื่อเพื่อตรวจสอบว่าเป็นตัวแทนของข้อมูลลักษณะของฟิลด์ในฐานข้อมูลใดแล้วทำการสร้างวัตถุ ObjectField แล้วกำหนดค่าว่าชื่อฟิลด์อะไร และมีชนิดข้อมูลประเภทตัวอักษรหรือตัวเลข จากนั้นคืนค่าวัตถุนั้นกลับไป ซึ่งการกำหนดค่าชื่อตัวแทนนั้น โปรแกรมเมอร์สามารถกำหนดได้เองตามความเหมาะสม

– เมท็อด CloneMemberValue

โปรแกรมเมอร์ต้องกำหนดให้เมท็อดทำการกำหนดค่าคุณลักษณะทุกค่าให้เหมือนกับวัตถุที่ส่งให้ โดยโครงร่างฯ กำหนดให้เรียกเมท็อด

กลุ่มซึ่งขึ้นอยู่ความสัมพันธ์คลาส

– เมท็อด SetObjectValue

โปรแกรมเมอร์ต้องกำหนดให้เมท็อดทำการกำหนดค่าให้คุณลักษณะวัตถุโดยรับดัชนีตัวแทนคุณลักษณะ และค่าที่จะกำหนดให้ โดยโปรแกรมเมอร์สามารถกำหนดดัชนีตัวแทนคุณลักษณะได้เองตามความเหมาะสม โดยโครงร่างฯ กำหนดให้เขียนโปรแกรมตามคลาสสัมพันธ์ดังนี้

1. ให้พิจารณาว่าคุณลักษณะใดเป็นค่าที่ได้จากวัตถุอื่นให้ทำการสร้างเมท็อดพิเศษเพิ่มเพื่อให้สามารถดึงวัตถุนั้นได้จากหมายเลขวัตถุ ซึ่งเมท็อดนี้จะทำการสร้างวัตถุชนิดเดียวกัน และทำการกำหนดหมายเลขวัตถุด้วยหมายเลขวัตถุตนเอง จากนั้นเรียกเมท็อด GetMemberValue เพื่อกำหนดค่าคุณลักษณะด้วยการดึงจากฐานข้อมูล สิ่งที่โครงร่างฯ กำหนดคือวัตถุที่สร้างใหม่ต้องจัดเก็บในรายชื่อของวัตถุแอสโซซิเอดเพื่อใช้ในการทำลายหลังเลิกใช้งาน โดยกระทำผ่านเมท็อด PushAssociate

2. จากนั้นเมื่อถึงคุณลักษณะที่เก็บค่าวัตถุแอสโซซิเอดให้เรียกเมท็อดพิเศษดังกล่าวเพื่อสร้างวัตถุแอสโซซิเอด แล้วกำหนดวัตถุนั้นให้วัตถุอีกครั้ง

– เมท็อด SelectParent

โปรแกรมเมอร์ต้องกำหนดให้เมท็อดทำให้เลือกวัตถุแม่ โดยโครงร่างฯ กำหนดให้เขียนโปรแกรมตามคลาสสัมพันธ์ดังนี้

1. กรณีที่การรับทอดคลาสการจากคลาสอื่นยกเว้นคลาส DbObject ให้ทำการสร้างวัตถุของคลาสแม่จากนั้นทำการกำหนดค่าคุณลักษณะของวัตถุตนเองให้วัตถุแม่ที่สร้างขึ้นด้วยเมท็อด CloneMemberValue จากนั้นทำการดึงค่าลักษณะของวัตถุแม่ด้วยเมท็อด GetMemberValue จากทำการกำหนดค่าคุณลักษณะกลับไปยังวัตถุตนเองด้วยเมท็อด GetMemberValue อีกครั้ง
2. ทำลายวัตถุที่สร้างใหม่
3. ให้ทำการเรียกเมท็อดนี้ของคลาสแม่ เพื่อทำในรุ่นก่อนหน้า

– เมท็อด SaveParent

โปรแกรมเมอร์ต้องกำหนดให้เมท็อดทำให้บันทึกวัตถุแม่ซึ่งหมายถึงการเพิ่ม การปรับปรุง การลบวัตถุ โดยโครงร่างฯ กำหนดให้เขียนโปรแกรมตามคลาสสัมพันธ์ดังนี้

1. กรณีที่การรับทอดคลาสการจากคลาสอื่นยกเว้นคลาส DbObject ให้ทำการสร้างวัตถุของคลาสแม่จากนั้นทำการกำหนดค่าคุณลักษณะของวัตถุตนเองให้วัตถุแม่ที่สร้างขึ้นด้วยเมท็อด CloneMemberValue จากนั้นทำการบันทึกด้วยเมท็อด Save
2. ทำลายวัตถุที่สร้างใหม่

– เมท็อด SelectComponent

โปรแกรมเมอร์ต้องกำหนดให้เมท็อดการเลือกวัตถุคอมโพเนนท์กลับมา โดยโครงร่างกำหนดให้เรียกเมท็อดนี้ของวัตถุแม่ด้วย

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

– เมท็อด SaveComponent

โปรแกรมเมอร์ต้องกำหนดให้เมท็อดการเรียกเมท็อดของวัตถุคอมโพเนนท์ทั้งหมด โดยการเรียกซ้ำ หากไม่มีวัตถุคอมโพเนนท์ในคลาสไม่ต้องกระทำสิ่งใด

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

บทที่ 5

การพัฒนาโครงร่างโปรแกรมประยุกต์เชิงวัตถุ
สำหรับพัฒนาโปรแกรมประยุกต์ฐานข้อมูลเชิงสัมพันธ์

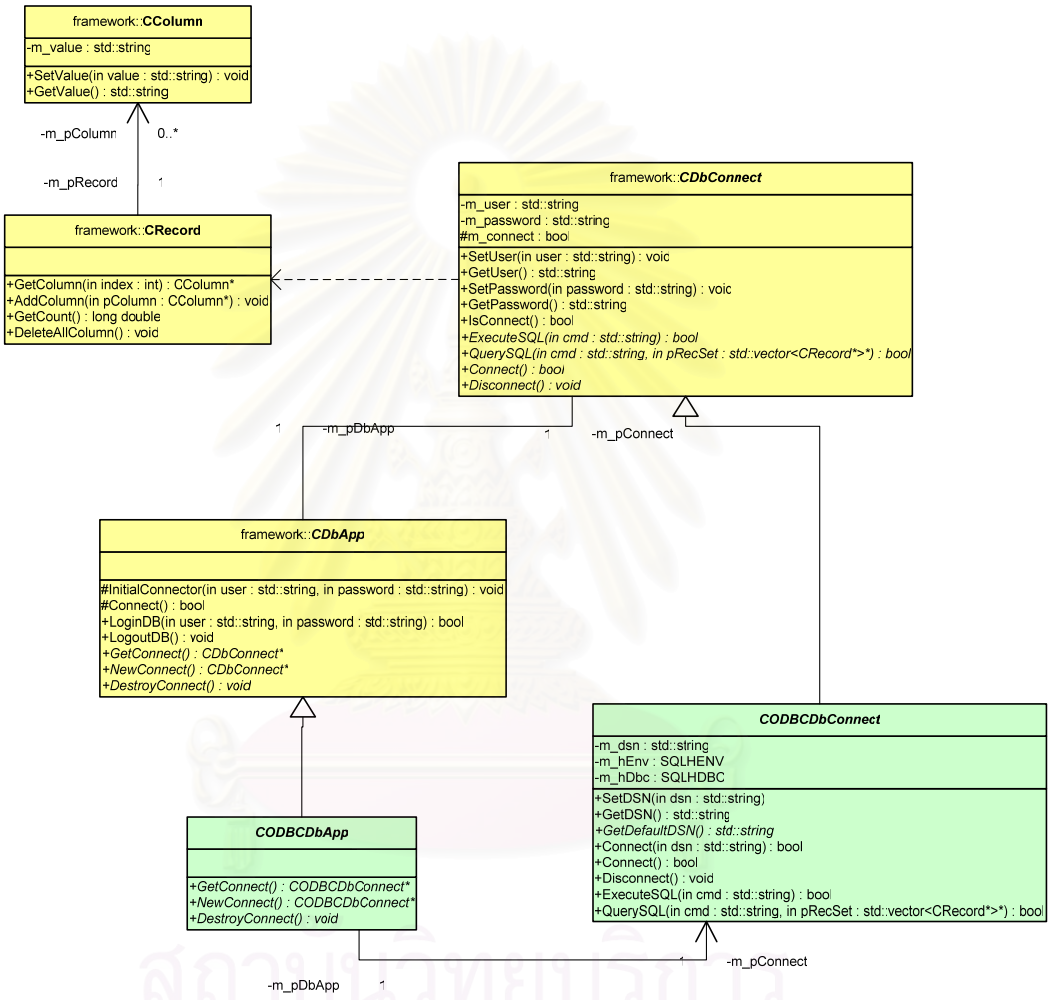
ในบทนี้จะกล่าวถึงการนำโครงร่างฯ ที่ออกแบบไปพัฒนาใช้งานจริง โดยในบทนี้กล่าวถึงพัฒนาโครงร่างฯ สำหรับภาษา C++ โดยผ่านตัวกลางในการเชื่อมต่อฐานข้อมูลเชิงสัมพันธ์โอดีบีซี (Open Database Connectivity: ODBC) ไปยังฐานข้อมูลเชิงสัมพันธ์ใดๆ ที่สนับสนุนการเชื่อมต่อผ่านโอดีบีซี เช่น Oracle Database, Microsoft SQL Server และ MySQL เป็นต้น โดยโครงร่างฯ ที่พัฒนาทำงานภายใต้โอดีบีซี สำหรับระบบปฏิบัติการไมโครซอฟท์วินโดว์ ในภาคผนวก ข แสดงรหัสคำสั่งของคลาสต่างๆ

5.1. ขั้นตอนในการพัฒนาโครงร่างฯ

1. ทำการเลือกภาษาเชิงวัตถุในการพัฒนาโดยเลือก ภาษา C++
2. ทำการเลือกฐานข้อมูลเชิงสัมพันธ์ในการพัฒนาโดยเลือก MySQL
3. ทำการเลือกวิธีการปฏิสัมพันธ์กับฐานข้อมูลเชิงสัมพันธ์ในการพัฒนาโดยเลือกมาตรฐานการเชื่อมต่อแบบโอดีบีซี
4. นำโครงร่างฯที่พัฒนาไปใช้งานในโปรแกรมยุกต์
5. ทำการพัฒนาสำหรับการเพิ่มวัตถุซึ่งมีคลาสความสัมพันธ์แบบซิงเกิลคลาส การรับทอดคลาส ภาพรวมกลุ่มคลาส คอมโพสิตชันคลาส
6. ทำการพัฒนาสำหรับการปรับปรุงวัตถุซึ่งมีคลาสความสัมพันธ์แบบซิงเกิลคลาส การรับทอดคลาส ภาพรวมกลุ่มคลาส คอมโพสิตชันคลาส
7. ทำการพัฒนาสำหรับการลบวัตถุซึ่งมีคลาสความสัมพันธ์แบบซิงเกิลคลาส การรับทอดคลาส ภาพรวมกลุ่มคลาส คอมโพสิตชันคลาส
8. ทำการพัฒนาสำหรับการเลือกวัตถุซึ่งมีคลาสความสัมพันธ์แบบซิงเกิลคลาส การรับทอดคลาส ภาพรวมกลุ่มคลาส คอมโพสิตชันคลาส
9. ทำการพัฒนาสำหรับการเชื่อมต่อไปยังฐานข้อมูลเชิงสัมพันธ์
10. ทำการพัฒนาสำหรับการยกเลิกการเชื่อมต่อไปยังฐานข้อมูลเชิงสัมพันธ์

5.2. ขั้นตอนในการพัฒนาเพื่อสร้างโครงร่างฯ

รูปที่ 5.1 แสดงแผนภาพคลาสซึ่งได้จากการนำโครงร่างฯ ที่ออกแบบไปพัฒนาเพื่อใช้งานโดยมีขั้นตอนดังนี้จะกล่าวต่อไป ซึ่งสามารถดูรหัสคำสั่งได้ในภาคผนวก ข



รูปที่ 5.1 แผนภาพคลาสของโครงร่างฯ สำหรับเชื่อมต่อฐานข้อมูลเชิงสัมพันธ์มาตรฐานโอดีบีซี

1. พัฒนาคลาสโปรแกรมประยุกต์เชิงวัตถุสำหรับโปรแกรมประยุกต์เชิงวัตถุซึ่งใช้มาตรฐานโอดีบีซี เป็นตัวกลางในการเชื่อมต่อไปยังฐานข้อมูลเชิงสัมพันธ์ โดยการการรับทอดจากคลาส CDbApp ไปเป็น CODBCDbApp ดังรูปที่ 5.2

```
0001 // Static Model
0002
0003
0004 #ifndef __ODBCDBAPP__
0005 #define __ODBCDBAPP__
0006
0007 #ifdef OOFARDB_EXPORTS
0008 #define OOFARDB_API __declspec(dllexport)
0009 #else
0010 #define OOFARDB_API __declspec(dllimport)
```

```
0011 #endif
0012
0013 // Include files
0014 #include "..\framework\DbApp.h"
0015 #include "ODBCDbConnect.h"
0016
0017 namespace odbc
0018 {
0019     class OOFARDB_API CODBCDbApp : public framework::CDbApp
0020     {
0021
0022     private:
0023
0024     protected:
0025
0026             virtual CODBCDbConnect* NewConnect();
0027
0028             virtual void DestroyConnect();
0029
0030     public:
0031
0032             virtual CODBCDbConnect* GetConnect();
0033
0034     };// END CLASS DEFINITION CODBCDbApp
0035 } // odbc
0036
0037
0038 #endif // __ODBCDBAPP__
```

รูปที่ 5.2 ตัวอย่างภาษา C++ ของคลาส ODBCDbApp

2. พัฒนาเมท็อด GetConnect ใหม่แทนที่เมท็อดของ CDbApp โดยให้คืนค่า CODBCDbConnect แทน DbConnect ดังรูป 5.3

```
0012     CODBCDbConnect* CODBCDbApp::GetConnect()
0013     {
0014             return NULL;
0015     }
0016
```

รูปที่ 5.3 ตัวอย่างภาษา C++ ของเมท็อด GetConnect ของคลาส ODBCDbApp

3. พัฒนาเมท็อด NewConnect ใหม่แทนที่เมท็อดของ CDbApp ซึ่งคืนค่า CODBCDbConnect แทน DbConnect ดังรูป 5.4

```
0007     CODBCDbConnect* CODBCDbApp::NewConnect()
0008     {
0009             return NULL;
0010     }
0011
```

รูปที่ 5.4 ตัวอย่างภาษา C++ ของเมท็อด NewConnect ของคลาส ODBCDbApp

4. พัฒนาเมท็อด DestroyConnect ใหม่แทนที่เมท็อดของ CDbApp ซึ่งทำลาย CODBCDbConnect แทน DbConnect ดังรูป 5.5

```
0017     void CODBCDbApp::DestroyConnect()
0018     {
0019
0020     }
0021
```

รูปที่ 5.5 ตัวอย่างภาษา C++ ของเมท็อด DestroyConnect ของคลาส ODBCDbApp

5. พัฒนาคลาสใหม่ ODBCDbConnect ซึ่งการรับทอดจากคลาส DbConnect เพื่อใช้สำหรับการเชื่อมต่อผ่านมาตรฐานโอดีบีซี ดังรูปที่ 5.6

```
0001 // Static Model
0002
```

```
0003
0004 #ifndef __ODBCDBCONNECT__
0005 #define __ODBCDBCONNECT__
0006
0007 #ifdef OOFARDB_EXPORTS
0008 #define OOFARDB_API __declspec(dllexport)
0009 #else
0010 #define OOFARDB_API __declspec(dllimport)
0011 #endif
0012
0013 // Include files
0014 #include <windows.h>
0015 #include <sql.h>
0016 #include <sqlext.h>
0017
0018 #include "..\framework\DbConnect.h"
0019 #include <vector>
0020 #include <string>
0021
0022 using namespace framework;
0023 namespace odbc
0024 {
0025     class OOFARDB_API CODBCDbConnect : public framework::CDbConnect
0026     {
0027
0028     private:
0029
0030             std::string m_dsn;
0031
0032             SQLHENV m_hEnv;
0033
0034             SQLHDBC m_hDbc;
0035
0036     public:
0037
0038             void SetDSN(std::string dsn);
0039
0040             std::string GetDSN();
0041
0042             virtual std::string GetDefaultDSN();
0043
0044             virtual bool Connect(std::string dsn);
0045
0046             virtual bool Connect();
0047
0048             virtual void Disconnect();
0049
0050             virtual bool ExecuteSQL(std::string cmd);
0051
0052             virtual bool QuerySQL(std::string cmd,
0053                                                 std::vector<CRecord*>*
pRecSet);
0054
0055     };// END CLASS DEFINITION CODBCDbConnect
0056 } // odbc
0057
0058
0059 #endif // __ODBCDBCONNECT__
```

รูปที่ 5.6 ตัวอย่างภาษา C++ ของคลาส ODBCDbConnect

6. เนื่องจากการเชื่อมต่อผ่านโอดีบีซี ต้องมีการระบุชื่อแหล่งข้อมูล (Data Source Name) ดังนั้นจึงเพิ่มคุณลักษณะ m_dsn สำหรับเก็บชื่อแหล่งข้อมูล, และจำเป็นต้องมีตัวแปรเพื่อเก็บค่าสภาพแวดล้อมในการติดต่อและรายละเอียดของฐานข้อมูลที่ใช้ จึงเพิ่มคุณลักษณะ m_hEnv สำหรับเก็บค่าสภาพแวดล้อมฐานข้อมูล และ m_hDbc สำหรับเก็บค่ารายละเอียดฐานข้อมูล ดังรูปที่ 5.7

```
0029
0030             std::string m_dsn;
0031
0032             SQLHENV m_hEnv;
0033
0034             SQLHDBC m_hDbc;
0035
```

รูปที่ 5.7 ตัวอย่างภาษา C++ คุณลักษณะของการเชื่อมผ่านโอดีบีซีของคลาส ODBCDbConnect

7. เพิ่มเมท็อด SetDSN สำหรับกำหนดค่าชื่อแหล่งข้อมูล ดังรูปที่ 5.8

```
0007    void CODBCDbConnect::SetDSN(std::string dsn)
0008    {
0009            m_dsn = dsn;
0010    }
0011
```

รูปที่ 5.8 ตัวอย่างภาษา C++ เมท็อดของ SetDSN คลาส ODBCDbConnect

8. เพิ่มเมท็อด GetDSN สำหรับคืนค่าชื่อแหล่งข้อมูล ดังรูปที่ 5.9

```
0012    std::string CODBCDbConnect::GetDSN()
0013    {
0014            return m_dsn;
0015    }
0016
```

รูปที่ 5.9 ตัวอย่างภาษา C++ เมท็อดของ GetDSN คลาส ODBCDbConnect

9. เพิ่มเมท็อด GetDefaultDSN สำหรับคืนค่าชื่อแหล่งข้อมูลปัจจุบัน โดยเป็นเมท็อดซึ่งต้องสร้างใหม่เมื่อใช้งาน ดังรูปที่ 5.10

```
0017    std::string CODBCDbConnect::GetDefaultDSN()
0018    {
0019            return ""; // dsn name
0020    }
0021
```

รูปที่ 5.10 ตัวอย่างภาษา C++ เมท็อดของ GetDefaultDSN คลาส ODBCDbConnect

10. พัฒนาเมท็อด Connect ใหม่ซึ่งรับค่าชื่อแหล่งข้อมูล โดยใช้ชุดคำสั่งสำหรับสร้างการเชื่อมต่อแบบ โอดีบีซี ดังรูปที่ 5.11

```
0022    bool CODBCDbConnect::Connect(std::string dsn)
0023    {
0024            SQLRETURN retcode;
0025
0026            SetDSN(dsn);
0027
0028            m_connect = false;
0029            retcode = SQLAllocHandle(SQL_HANDLE_ENV, SQL_NULL_HANDLE,
&m_hEnv);
0030            if (retcode == SQL_SUCCESS || retcode == SQL_SUCCESS_WITH_INFO)
{
0031                    retcode = SQLSetEnvAttr(m_hEnv, SQL_ATTR_ODBC_VERSION,
(void*)SQL_OV_ODBC3, 0);
0032                    if (retcode == SQL_SUCCESS || retcode ==
SQL_SUCCESS_WITH_INFO) {
0033                            retcode = SQLAllocHandle(SQL_HANDLE_DBC,
m_hEnv, &m_hDbc);
0034                            if (retcode == SQL_SUCCESS || retcode ==
SQL_SUCCESS_WITH_INFO) {
0035                                    SQLSetConnectAttr(m_hDbc,
SQL_LOGIN_TIMEOUT, (void*)5, 0);
0036                                    retcode = SQLConnect(m_hDbc,
(SQLCHAR*)GetDSN().data(), SQL_NTS, (SQLCHAR*)GetUser().data(), SQL_NTS,
(SQLCHAR*)GetPassword().data(), SQL_NTS);
0037                                    if (retcode == SQL_SUCCESS || retcode
== SQL_SUCCESS_WITH_INFO)
0038                                            m_connect = true;
0039                            }
0040                    }
0041            }
0042
0043            return IsConnect();
0044    }
0045
```

รูปที่ 5.11 ตัวอย่างภาษา C++ เมท็อดของ Connect รับชื่อแหล่งข้อมูล คลาส ODBCDbConnect

11. พัฒนาเมท็อด Connect ใหม่แทนที่ เมท็อดของ DbConnect โดยใช้ชุดคำสั่งสำหรับสร้างการเชื่อมต่อฐานข้อมูลมาตรฐานโอดีบีซี ดังรูปที่ 5.12

```
0046    bool CODBCDbConnect::Connect()
0047    {
0048            return Connect(GetDefaultDSN());
0049    }
0050
```

รูปที่ 5.12 ตัวอย่างภาษา C++ เมท็อดของ Connect คลาส ODBCDbConnect

12. พัฒนาเมท็อด Disconnect ใหม่แทนที่ เมท็อดของ DbConnect โดยเป็นชุดคำสั่งสำหรับสร้างการยกเลิกการเชื่อมต่อด้วยมาตรฐานโอดีบีซี ดังรูปที่ 5.13

```
0051    void CODBCDbConnect::Disconnect()
0052    {
0053            if (IsConnect()) {
0054                    if (m_hDbc) {
0055                            SQLDisconnect(m_hDbc);
0056                            SQLFreeHandle(SQL_HANDLE_DBC, m_hDbc);
0057                    }
0058                    if (m_hEnv)
0059                            SQLFreeHandle(SQL_HANDLE_ENV, m_hEnv);
0060                    m_connect = false;
0061            }
0062    }
0063
```

รูปที่ 5.13 ตัวอย่างภาษา C++ เมท็อดของ Disconnect คลาส ODBCDbConnect

13. พัฒนาเมท็อด ExecuteSQL ใหม่แทนที่ เมท็อดของ DbConnect โดยเป็นชุดคำสั่งสำหรับการส่งชุดคำสั่งเอสคิวแอลไปประมวลผลที่ฐานข้อมูลเชิงสัมพันธ์ด้วยมาตรฐานโอดีบีซี ดังรูปที่ 5.14

```
0064 bool CODBCDbConnect::ExecuteSQL(std::string cmd)
0065 {
0066    SQLHSTMT hstmt;
0067    SQLRETURN retcode;
0068    bool bResult = false;
0069
0070    if (IsConnect()) {
0071            retcode = SQLAllocHandle(SQL_HANDLE_STMT, m_hDbc, &hstmt);
0072            if (retcode == SQL_SUCCESS || retcode == SQL_SUCCESS_WITH_INFO)
{
0073            retcode = SQLExecDirect(hstmt, (SQLCHAR*)cmd.data(), SQL_NTS);
0074                    if (retcode == SQL_SUCCESS || retcode == SQL_NO_DATA)
0075                            bResult = true;
0076                    SQLFreeHandle(SQL_HANDLE_STMT, hstmt);
0077            }
0078    }
0079    return bResult;
0080 }
0081
```

รูปที่ 5.14 ตัวอย่างภาษา C++ เมท็อดของ ExecuteSQL คลาส ODBCDbConnect

14. พัฒนาเมท็อด QuerySQL ใหม่แทนที่ เมท็อดของ DbConnect โดยเป็นชุดคำสั่งสำหรับส่งคำสั่งเอสคิวแอลเพื่อการเลือกโดยเฉพาะไปยังฐานข้อมูลเชิงสัมพันธ์ด้วยมาตรฐานโอดีบีซี แล้วส่งชุดของระเบียนของวัตถุที่เลือกได้คืนมา ดังรูปที่ 5.15

```
0082 bool CODBCDbConnect::QuerySQL(std::string cmd,
0083                                    std::vector<CRecord*>* pRecSet)
0084 {
0085    SQLHSTMT hstmt;
0086    SQLRETURN retcode;
0087    bool bResult = false;
0088    bool bEnd = false;
0089    SQLCHAR          v[512]="";
```

```
0090    std::string s;
0091    SQLINTEGER       cb;
0092    SQLSMALLINT cbColAttr, nField;
0093    SQLINTEGER       nFieldCount;
0094    SQLINTEGER       nFieldLength;
0095    CRecord *pRec;
0096    CColumn *pCol;
0097
0098    if (IsConnect()) {
0099            retcode = SQLAllocHandle(SQL_HANDLE_STMT, m_hDbc, &hstmt);
0100            if (retcode == SQL_SUCCESS || retcode == SQL_SUCCESS_WITH_INFO)
{
0101 SQLSetStmtAttr(hstmt, SQL_ATTR_CONCURRENCY,
(SQLPOINTER)SQL_CONCUR_READ_ONLY, 0);
0102    SQLSetStmtAttr(hstmt, SQL_ATTR_CURSOR_TYPE,
(SQLPOINTER)SQL_CURSOR_STATIC, 0);
0103            retcode = SQLExecDirect(hstmt, (SQLCHAR*)cmd.data(), SQL_NTS);
0104                    if (retcode == SQL_SUCCESS) {
0105                            nFieldCount = 0;
0106                            retcode = SQLColAttribute(hstmt, 0,
SQL_DESC_COUNT, NULL, sizeof(nFieldCount), &cbColAttr, &nFieldCount);
0107                            if (retcode == SQL_SUCCESS || retcode ==
SQL_SUCCESS_WITH_INFO) {
0108                                    bResult = true;
0109                                    while (!bEnd) {
0110                                            retcode = SQLFetch(hstmt);
0111                                            if (retcode == SQL_SUCCESS ||
retcode == SQL_SUCCESS_WITH_INFO) {
0112                                                    pRec = new CRecord();

0113                                                    for (nField = 1;
nField <= nFieldCount; nField++) {
0114                                                    pCol = new CColumn();
0115                                                    retcode =
SQLColAttribute(hstmt, nField, SQL_DESC_OCTET_LENGTH, NULL, sizeof(nFieldLength),
&cbColAttr, &nFieldLength);
0116
        SQLGetData(hstmt, nField, SQL_C_CHAR, &v, 512, &cb);
0117                                                    s = (LPCTSTR)v;
0118                                                    pCol->SetValue(s);
0119                                                    pRec-
>AddColumn(pCol);
0120                                            }
0121                                            pRecSet->push_back(pRec);
0122                                            }
0123                                            else bEnd = true;
0124                                            }
0125                                    }
0126                            }
0127                            SQLFreeHandle(SQL_HANDLE_STMT, hstmt);
0128                    }
0129            }
0130            return bResult;
0131
0132    }
```

รูปที่ 5.15 ตัวอย่างภาษา C++ เมท็อดของ QuerySQL คลาส ODBCDbConnect

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

บทที่ 6

การใช้งานโครงร่างโปรแกรมประยุกต์เชิงวัตถุ สำหรับพัฒนาโปรแกรมประยุกต์ฐานข้อมูลเชิงสัมพันธ์

บทนี้กล่าวถึงการนำโครงร่างฯ ที่พัฒนาในบทที่ที่แล้วมาใช้งานจริง โดยยกกรณีศึกษาระบบการสั่งซื้อสินค้าแบบง่ายมาเป็นตัวอย่าง โดยจะขออธิบายเฉพาะในส่วนที่เกี่ยวข้องการนำโครงร่างฯ มาใช้งานเท่านั้น ซึ่งสามารถดูผลของการพัฒนาโปรแกรมประยุกต์ โดยภาคผนวก ง. แสดงตัวอย่างหน้าจอ ผลของชุดรหัสคำสั่งที่ได้จากในภาคผนวก ค และผลของวัตถุที่จัดเก็บในตารางในภาคผนวก จ

ในการพัฒนาระบบการสั่งซื้อสินค้าแบบง่ายซึ่งทำงานภายใต้ระบบปฏิบัติการไมโครซอฟท์วินโดว์ โดยใช้เครื่องมือในการพัฒนาดังนี้

1. Microsoft Visual C++ .NET
2. คลาสไลบรารี MFC (Microsoft Foundation Class) สำหรับส่วนติดต่อผู้ใช้
3. ระบบฐานข้อมูล MySQL

6.1. ขั้นตอนการใช้งานโครงร่างฯ

1. ทำการเลือกระบบงานซึ่งสามารถใช้งานโครงร่างฯได้ทุกรูปแบบ ในที่นี้คือระบบการสั่งซื้อสินค้าแบบง่าย
2. ทำการใช้งานการเชื่อมต่อฐานข้อมูลเชิงสัมพันธ์ซึ่งทดสอบขณะเริ่มใช้ระบบ
3. ทำการใช้งานการเพิ่มวัตถุในฐานข้อมูลเชิงสัมพันธ์สำหรับความสัมพันธ์คลาสแบบคลาสเดี่ยวซึ่งเลือกคลาสสินค้าเป็นตัวแทน
4. ทำการใช้งานการเพิ่มวัตถุในฐานข้อมูลเชิงสัมพันธ์สำหรับความสัมพันธ์คลาสแบบการรับทอดคลาสซึ่งเลือกคลาสพนักงานขายเป็นตัวแทน
5. ทำการใช้งานการเพิ่มวัตถุในฐานข้อมูลเชิงสัมพันธ์สำหรับความสัมพันธ์คลาสแบบภาพรวมกลุ่มคลาสซึ่งเลือกคลาสลูกค้าเป็นตัวแทน
6. ทำการใช้งานการเพิ่มวัตถุในฐานข้อมูลเชิงสัมพันธ์สำหรับความสัมพันธ์คลาสแบบคอมโพสิตชันคลาสซึ่งเลือกคลาสการสั่งซื้อเป็นตัวแทน

7. ทำการใช้งานการปรับปรุงวัตถุในฐานข้อมูลเชิงสัมพันธ์สำหรับความสัมพันธ์คลาสแบบคลาสเดี่ยวซึ่งเลือกคลาสสินค้าเป็นตัวแทน
8. ทำการใช้งานการปรับปรุงวัตถุในฐานข้อมูลเชิงสัมพันธ์สำหรับความสัมพันธ์คลาสแบบการรับทอดคลาสซึ่งเลือกคลาสพนักงานขายเป็นตัวแทน
9. ทำการใช้งานการปรับปรุงวัตถุในฐานข้อมูลเชิงสัมพันธ์สำหรับความสัมพันธ์คลาสแบบภาพรวมกลุ่มคลาสซึ่งเลือกคลาสลูกค้าเป็นตัวแทน
10. ทำการใช้งานการปรับปรุงวัตถุในฐานข้อมูลเชิงสัมพันธ์สำหรับความสัมพันธ์คลาสแบบคอมโพสิตชันคลาสซึ่งเลือกคลาสการสั่งซื้อเป็นตัวแทน
11. ทำการใช้งานการลบวัตถุในฐานข้อมูลเชิงสัมพันธ์สำหรับความสัมพันธ์คลาสแบบคลาสเดี่ยวซึ่งเลือกคลาสสินค้าเป็นตัวแทน
12. ทำการใช้งานการลบวัตถุในฐานข้อมูลเชิงสัมพันธ์สำหรับความสัมพันธ์คลาสแบบคลาสซึ่งการรับทอดซึ่งเลือกคลาสพนักงานขายเป็นตัวแทน
13. ทำการใช้งานการลบวัตถุในฐานข้อมูลเชิงสัมพันธ์สำหรับความสัมพันธ์คลาสแบบภาพรวมกลุ่มคลาสซึ่งเลือกคลาสลูกค้าเป็นตัวแทน
14. ทำการใช้งานการลบวัตถุในฐานข้อมูลเชิงสัมพันธ์สำหรับความสัมพันธ์คลาสแบบคอมโพสิตชันคลาสซึ่งเลือกคลาสการสั่งซื้อเป็นตัวแทน
15. ทำการใช้งานการยกเลิกการเชื่อมต่อไปยังฐานข้อมูลเชิงสัมพันธ์ซึ่งทดสอบขณะจบการใช้ระบบ

6.2. แผนภาพคลาสระบบการสั่งซื้อสินค้าแบบง่าย

ซึ่งแสดงดังรูปที่ 6.1 โดยประกอบด้วยคลาสดังนี้

6.2.1. คลาสบุคคล (Person)

6.2.2. คลาสลูกค้า (Customer)

6.2.3. คลาสพนักงาน (Employee)

6.2.4. คลาสพนักงานขาย (Sales)

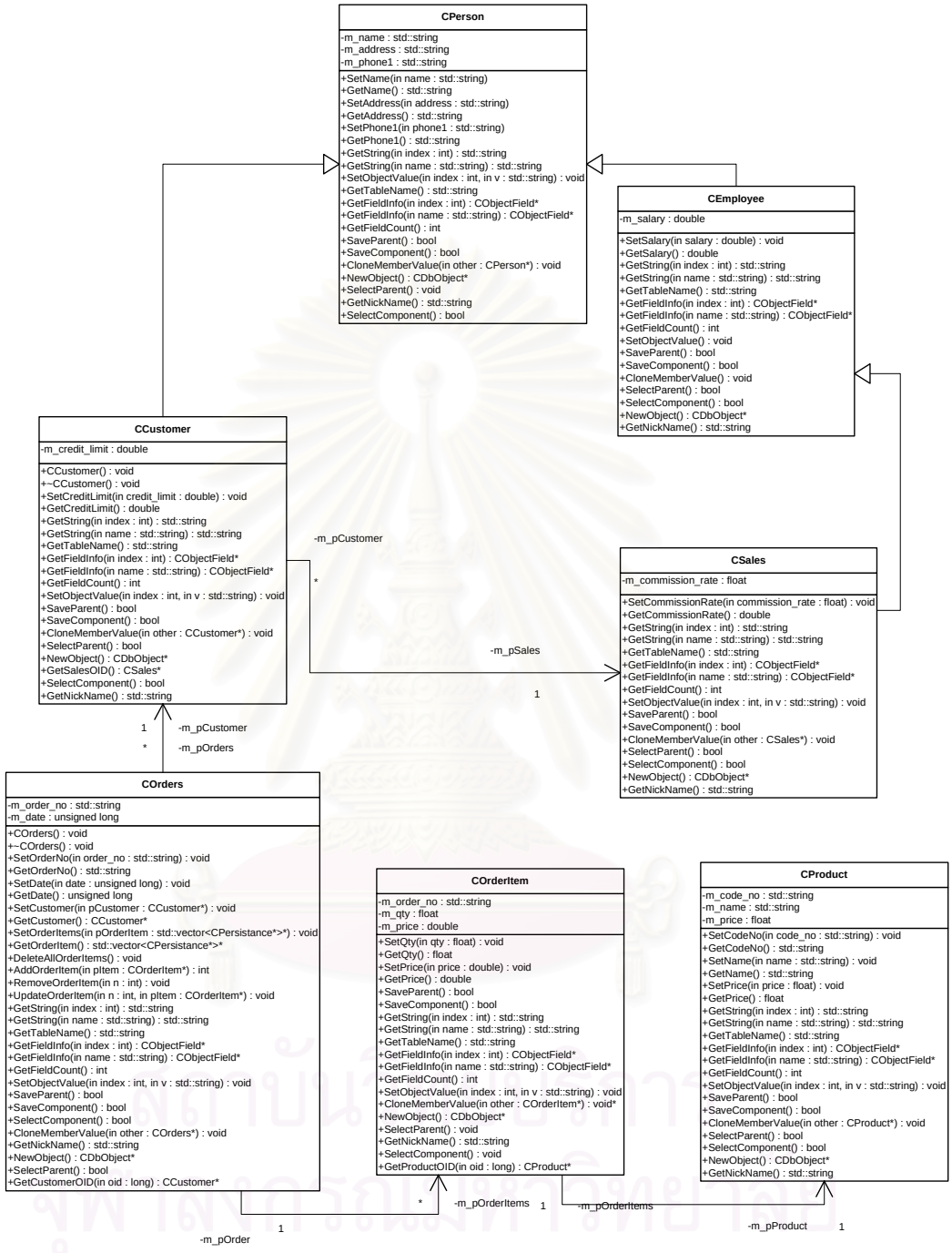
6.2.5. คลาสการสั่งซื้อ (Orders)

6.2.6. คลาสรายการสั่งซื้อ (OrderItem)

6.2.7. คลาสสินค้า (Product)

6.2.8. คลาสโปรแกรมประยุกต์ระบบการสั่งซื้อ (SalesOrderODBCDbApp)

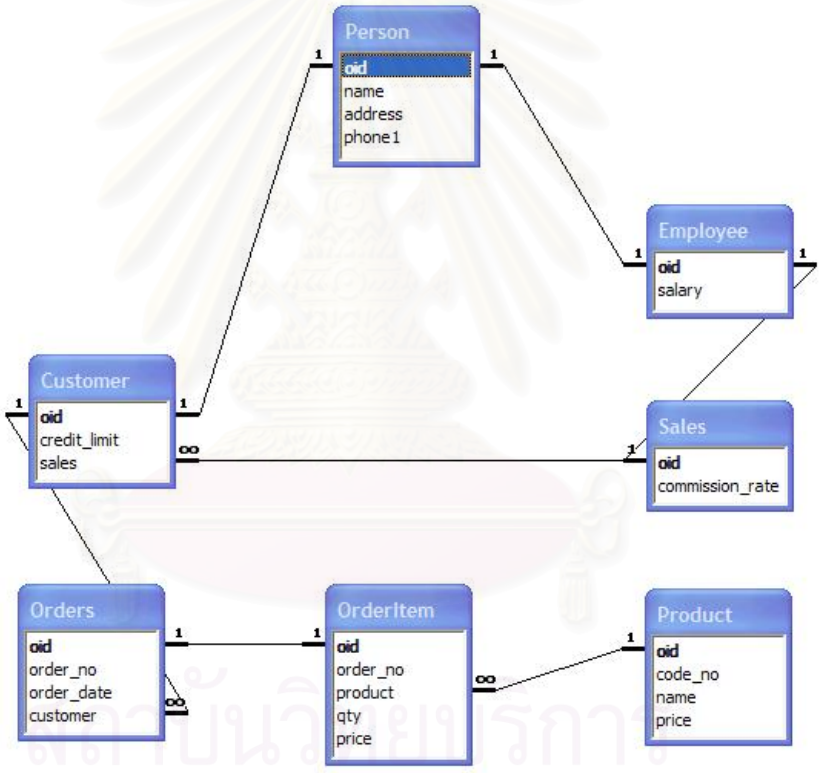
6.2.9. คลาสการเชื่อมต่อระบบการสั่งซื้อ (SalesOrderODBCDbConnect)



รูปที่ 6.1 แผนภาพคลาสระบบการสั่งซื้อสินค้าแบบง่าย

6.3. แผนภาพตารางความสัมพันธ์ของระบบการสั่งซื้อสินค้าแบบง่าย

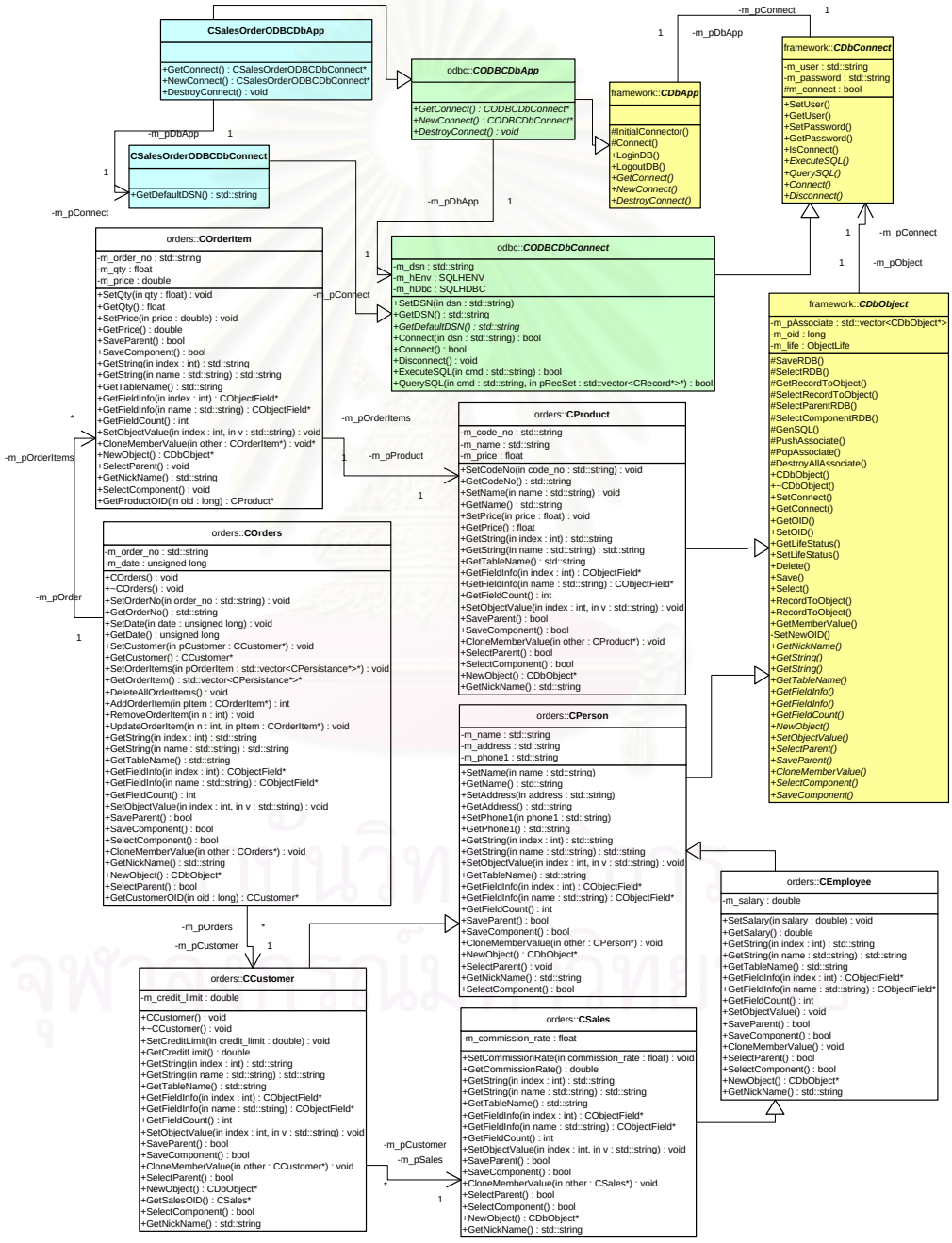
จากแผนภาพคลาสของในรูปที่ 6.1 จากข้อกำหนดของโครงร่างฯ ให้ทำการแปลงแผนภาพคลาสไปเป็นแผนภาพตารางความสัมพันธ์ โดยหลักการแปลงคือหนึ่งคลาสเป็นหนึ่งตารางความสัมพันธ์ดังแสดงในรูปที่ 6.2 แสดงให้เห็นในรูปที่ 6.1 แผนภาพคลาสประกอบด้วย 7 คลาสดังนั้นจึงได้ตารางความสัมพันธ์ 7 ตารางความสัมพันธ์ โดยในการโครงร่างฯ ไม่ได้สนใจว่าต้องสร้างความสัมพันธ์ระหว่างตารางหรือไม่ เนื่องจากโครงร่างฯ มองฐานข้อมูลเชิงสัมพันธ์เป็นเพียงที่เก็บข้อมูลของวัตถุเท่านั้น



รูปที่ 6.2 แผนภาพตารางความสัมพันธ์ระบบการสั่งซื้อสินค้าแบบง่าย

6.4. แผนภาพคลาสระบบการสั่งซื้อสินค้าแบบง่ายเมื่อใช้ร่วมกับโครงร่างฯ

ในรูปที่ 6.3 แสดงให้เห็นความสัมพันธ์ของคลาสของโปรแกรมประยุกต์กับโครงร่างฯ เท่านั้น ในการใช้งานโครงร่างฯ จริงไม่มีความจำเป็นต้องออกแบบแผนภาพคลาสโดยนำแผนภาพคลาสของโครงร่างมาแสดงด้วย



รูปที่ 6.3 แผนภาพคลาสระบบการสั่งซื้อสินค้าแบบง่ายร่วมกับโครงร่างฯ

6.5. การใช้งานโครงร่างฯ กับความสัมพันธ์คลาส

ในการใช้งานนั้นให้พิจารณาที่คลาสที่ต้องการจัดเก็บของระบบการสั่งซื้อสินค้าแบบง่าย โดยพิจารณาว่าคลาสมีลักษณะความสัมพันธ์เป็นแบบใด คือ เป็นซิงเกิลคลาส การรับทอดคลาส ภาพรวมกลุ่มคลาส คอมโพสิตชันคลาส กัน เมื่อได้แล้วให้ทำตามขั้นตอนตัวอย่างตามลักษณะความสัมพันธ์ของคลาสนั้น

6.4.1. ลักษณะแผนภาพคลาสแบบซิงเกิลคลาส (Single Class)
หมายถึงคลาสซึ่งไม่มีคลาสใดประกอบหรือขึ้นกับคลาสอื่น
เช่น คลาสสินค้า (CProduct)

```
0001 // Static Model
0002
0003
0004 #ifndef __PRODUCT__
0005 #define __PRODUCT__
0006
0007
0008 // Include files
0009 #include "..\..\..\oofardb\thesis\framework\Persistance.h"
0010 #include "..\..\..\oofardb\thesis\framework\ObjectField.h"
0011
0012 using namespace framework;
0013
0014 namespace orders
0015 {
0016     class CProduct : public framework::DbObject
0017     {
0018
0019     private:
0020
0021             std::string m_code_no;
0022
0023             std::string m_name;
0024
0025             float m_price;
0026
0027     public:
0028
0029             void SetCodeNo(std::string code_no);
0030
0031             std::string GetCodeNo();
0032
0033             void SetName(std::string name);
0034
0035             std::string GetName();
0036
0037             void SetPrice(float price);
0038
0039             float GetPrice();
0040
0041             virtual bool SaveParent();
0042
0043             virtual bool SaveComponent();
0044
0045             virtual std::string GetString(int index);
0046
0047             virtual std::string GetString(std::string name);
0048
0049             virtual std::string GetTableName();
0050
0051             virtual CObjectField* GetFieldInfo(int index);
0052
0053             virtual CObjectField* GetFieldInfo(std::string name);
0054
0055             virtual int GetFieldCount();
0056
0057             virtual void SetObjectValue(int index, std::string v);
0058
0059             virtual void CloneMemberValue(CProduct *other);
0060
0061             virtual DbObject* NewObject();
0062
0063             virtual bool SelectParent();
0064
0065             virtual std::string GetNickName();
0066
0067             virtual bool SelectComponent();
```

```
0068
0069
0070    };// END CLASS DEFINITION CProduct
0071 } // orders
0072
0073
0074 #endif // __PRODUCT__
```

รูปที่ 6.4 ตัวอย่างภาษา C++ ของคลาสสินค้า

1. สร้างคลาส CProduct โดยการรับทอดจาก DbObject
2. ออกแบบคุณลักษณะและเมท็อดตามความต้องการทางธุรกิจ
3. การรับทอดเมท็อดของ DbObject สำหรับโครงร่างฯ เรียกใช้

3.1. เมท็อด SaveParent เนื่องจากไม่มีคลาสแม่ จึงคืนค่า false ให้โครงร่างฯ เพื่อบอกว่าไม่มีคลาสแม่

```
0140    bool CProduct::SaveParent()
0141    {
0142            bool r = true;
0143
0144            return r;
0145    }
0146
```

รูปที่ 6.5 ตัวอย่างภาษา C++ ของเมท็อด SaveParent ของคลาสสินค้า

3.2. เมท็อด SaveComponent เนื่องจากไม่มีคลาสประกอบ จึงคืนค่า false ให้โครงร่างฯ เพื่อบอกว่าไม่มีคลาสประกอบ

```
0147    bool CProduct::SaveComponent()
0148    {
0149            bool r = true;
0150
0151            return r;
0152    }
0153
```

รูปที่ 6.6 ตัวอย่างภาษา C++ ของเมท็อด SaveComponent ของคลาสสินค้า

3.3. เมท็อด GetString ทำหน้าที่คืนค่าของคุณลักษณะตามดัชนี เมื่อโครงร่างฯ ต้องการ โดยรับดัชนีคุณลักษณะ

เนื่องจากคลาสนี้มีคุณลักษณะที่ต้องการจัดเก็บในคลาส 3 คุณลักษณะ คือ รหัสสินค้า, ชื่อสินค้า, ราคา จึงออกแบบให้ดัชนีที่ 1 คือ รหัสสินค้า, ดัชนีที่ 2 คือ ชื่อสินค้า, ดัชนีที่ 3 คือ ราคา ดังนั้นเมื่อโครงร่างฯ ส่งค่าดัชนีใดมาก็คืนค่าของคุณลักษณะนั้นในชนิดสตริงดังตัวอย่าง

```
0037    std::string CProduct::GetString(int index)
0038    {
0039            std::string value="";
0040
0041            switch (index) {
0042                    case 1:  // code_no
0043                            value = GetCodeNo();
0044                            break;
0045                    case 2: // name
0046                            value = GetName();
0047                            break;
0048                    case 3: // price
0049                            char c[20];
0050                            sprintf(c, "%.2f", GetPrice());
```

```
0051                                value = c;
0052                                break;
0053            }
0054
0055            return value;
0056    }
0057
```

รูปที่ 6.7 ตัวอย่างภาษา C++ ของเมท็อด GetString รับดัชนี ของคลาสสินค้า

3.4. เมท็อด GetString ทำหน้าที่คืนค่าของคุณลักษณะตามดัชนี เมื่อโครงร่างฯ ต้องการ โดยรับชื่อคุณลักษณะ

เนื่องจากคลาสนี้มีคุณลักษณะที่ต้องการจัดเก็บในคลาส 3 คุณลักษณะ คือ รหัสสินค้า, ชื่อสินค้า, ราคา จึงออกแบบให้ชื่อ code_no คือ รหัสสินค้า, ชื่อ name คือ ชื่อสินค้า, ชื่อ price คือ ราคา ดังนั้น เมื่อโครงร่างฯ ส่งค่าดัชนีใดมาก็คืนค่าของคุณลักษณะนั้นในชนิดสตริงดังตัวอย่าง

```
0058    std::string CProduct::GetString(std::string name)
0059    {
0060            int index=-1;
0061
0062            if (name == "code_no")
0063                    index = 0;
0064            else if (name == "name")
0065                    index = 1;
0066            else if (name == "price")
0067                    index = 2;
0068
0069            return GetString(index);
0070    }
0071
```

รูปที่ 6.8 ตัวอย่างภาษา C++ ของเมท็อด GetString รับชื่อ ของคลาสสินค้า

3.5. เมท็อด GetTableName หน้าที่คืนค่า ชื่อตารางที่เก็บสินค้า เมื่อโครงร่างฯ ต้องการทราบ

```
0072    std::string CProduct::GetTableName()
0073    {
0074            return std::string("Product");
0075    }
0076
```

รูปที่ 6.9 ตัวอย่างภาษา C++ ของเมท็อด GetTableName ของคลาสสินค้า

3.6. เมท็อด GetFieldInfo หน้าที่คืนค่าวัตถุ CObjectField ซึ่งเป็นวัตถุรายละเอียดของคุณลักษณะในตาราง ซึ่งรับค่าดัชนี เมื่อโครงร่างฯ ต้องการทราบ

```
0077    CObjectField* CProduct::GetFieldInfo(int index)
0078    {
0079            CObjectField* pField = NULL;
0080            if (index >= 1 && index <= GetFieldCount()) {
0081                    pField = new CObjectField();
0082                    switch (index) {
0083                    case 1:
0084                            pField->SetName("code_no");
0085                            pField->SetType(framework::character);
0086                            break;
0087                    case 2:
0088                            pField->SetName("name");
0089                            pField->SetType(framework::character);
0090                            break;
0091                    case 3:
0092                            pField->SetName("price");
0093                            pField->SetType(framework::numeric);
0094                            break;
0095                    }
```

```
0096            }
0097
0098            return pField;
0099    }
0100
```

รูปที่ 6.10 ตัวอย่างภาษา C++ ของเมท็อด GetFieldInfo รับดัชนี ของคลาสสินค้า

3.7. เมท็อด GetFieldInfo หน้าที่คืนค่าวัตถุ CObjectField ซึ่งเป็นวัตถุรายละเอียดของคุณลักษณะในตาราง ซึ่งรับค่าชื่อคุณลักษณะ เมื่อโครงร่างฯ ต้องการทราบ

```
0101    CObjectField* CProduct::GetFieldInfo(std::string name)
0102    {
0103            int fldIndex = -1;
0104
0105            if (name == std::string("code_no"))
0106                    fldIndex = 1;
0107            else if (name == std::string("name"))
0108                    fldIndex = 2;
0109            else if (name == std::string("price"))
0110                    fldIndex = 3;
0111
0112            return GetFieldInfo(fldIndex);
0113    }
0114
```

รูปที่ 6.11 ตัวอย่างภาษา C++ ของเมท็อด GetFieldInfo รับชื่อ ของคลาสสินค้า

3.8. เมท็อด GetFieldCount หน้าที่คืนค่า จำนวนคุณลักษณะที่ต้องการจัดเก็บ เมื่อโครงร่างฯ ต้องการทราบ

```
0115    int CProduct::GetFieldCount()
0116    {
0117            return 3;
0118    }
0119
```

รูปที่ 6.12 ตัวอย่างภาษา C++ ของเมท็อด GetFieldCount รับชื่อ ของคลาสสินค้า

3.9. เมท็อด SetObjectValue หน้าที่กำหนดค่าให้คุณลักษณะ โดยรับดัชนีที่ต้องการกำหนดค่าและค่าที่ต้องการกำหนดให้คุณลักษณะ โดยค่าดัชนี 0 สำรองสำหรับโครงร่างฯ ในกำหนดค่าหมายเลขวัตถุ นอกนั้นสามารถกำหนดตามลำดับความต้องการว่าดัชนีใดแทนคุณลักษณะใด เมื่อโครงร่างฯ ต้องการกำหนด

```
0120    void CProduct::SetObjectValue(int index, std::string v)
0121    {
0122            if (index >= 0 && index <= GetFieldCount()) {
0123                    switch (index) {
0124                    case 0:
0125                            SetOID(atol(v.data()));
0126                            break;
0127                    case 1:
0128                            SetCodeNo(v);
0129                            break;
0130                    case 2:
0131                            SetName(v);
0132                            break;
0133                    case 3:
0134                            SetPrice((float)atof(v.data()));
0135                            break;
0136                    }
0137            }
0138    }
0139
```

รูปที่ 6.13 ตัวอย่างภาษา C++ ของเมท็อด SetObjectValue รับดัชนี ของคลาสสินค้า

3.10. เมท็อด CloneMemberValue หน้าที่สำเนาเฉพาะค่าในคุณลักษณะของวัตถุหนึ่งให้อีกวัตถุหนึ่ง โดยโครงร่างฯกำหนดให้เรียกเมท็อด CloneMemberValue ของคลาสแม่ด้วยเสมอ ดังบรรทัด 160

```
0154    void CProduct::CloneMemberValue(CProduct* other)
0155    {
0156            SetCodeNo(other->GetCodeNo());
0157            SetName(other->GetName());
0158            SetPrice(other->GetPrice());
0159
0160            __super::CloneMemberValue(other);
0161    }
0162
```

รูปที่ 6.14 ตัวอย่างภาษา C++ ของเมท็อด CloneMemberValue ของคลาสสินค้า

3.11. เมท็อด NewObject ทำหน้าที่สร้างวัตถุของสินค้าใหม่เมื่อโครงร่างฯ ต้องการ โดยต้องการกำหนดวัตถุสำหรับการเชื่อมต่อขณะนั้นให้วัตถุใหม่เสมอ ดังบรรทัดที่ 184 และต้องแปลงเป็นวัตถุของคลาส DbObject เสมอ ดังบรรทัด 185

```
0181    DbObject* CProduct::NewObject()
0182    {
0183            CProduct *pObj = new CProduct();
0184            pObj->SetConnect(GetConnect());
0185            return static_cast<DbObject*>(pObj);
0186    }
0187
```

รูปที่ 6.15 ตัวอย่างภาษา C++ ของเมท็อด NewObject ของคลาสสินค้า

3.12. เมท็อด SelectParent ทำหน้าที่อ่านค่าคุณลักษณะของวัตถุของคลาสแม่ เมื่อโครงร่างฯ ต้องการ โดยโครงร่างกำหนดให้เรียกเมท็อดของคลาสแม่เสมอ ดังบรรทัด 167

```
0163    bool CProduct::SelectParent()
0164    {
0165            bool r = false;
0166
0167            r = __super::SelectParent();
0168
0169            return r;
0170    }
0171
```

รูปที่ 6.16 ตัวอย่างภาษา C++ ของเมท็อด SelectParent ของคลาสสินค้า

3.13. เมท็อด GetNickName ทำหน้าที่คือค่าชื่อของวัตถุ สำหรับคลาสสินค้า ใช้ชื่อสินค้า

```
0188    std::string CProduct::GetNickName()
0189    {
0190            return GetName();
0191    }
0192
```

รูปที่ 6.17 ตัวอย่างภาษา C++ ของเมท็อด GetNickName ของคลาสสินค้า

3.14. เมท็อด SelectComponent ทำหน้าที่อ่านค่าคุณลักษณะของวัตถุประกอบ เมื่อโครงร่างฯ ต้องการ โดยโครงร่างกำหนดให้เรียกเมท็อดของคลาสแม่เสมอ ดังบรรทัด 177

```
0172    bool CProduct::SelectComponent()
0173    {
0174            bool r = false;
0175
0176            r = __super::SelectComponent();
0177
0178            return r;
0179    }
0180
```

รูปที่ 6.18 ตัวอย่างภาษา C++ ของเมท็อด SelectComponent ของคลาสสินค้า

6.4.2. ลักษณะแผนภาพคลาสแบบการรับทอดคลาส (Inheritance Class)

หมายถึงคลาสซึ่งมีการการรับทอดจากคลาสอื่น

เช่น คลาสพนักงาน (CEmployee) การรับทอดจากคลาสบุคคล (CPerson)

```
0001 // Static Model
0002
0003
0004 #ifndef __PERSON__
0005 #define __PERSON__
0006
0007
0008 // Include files
0009 #include "..\..\..\oofardb\thesis\framework\ObjectField.h"
0010 #include "..\..\..\oofardb\thesis\framework\Persistance.h"
0011
0012 using namespace framework;
0013
0014 namespace orders
0015 {
0016    class CPerson : public framework::DbObject
0017    {
0018
0019    private:
0020
0021            std::string m_name;
0022
0023            std::string m_address;
0024
0025            std::string m_phone1;
0026
0027    public:
0028
0029            void SetName(std::string name);
0030
0031            std::string GetName();
0032
0033            void SetAddress(std::string address);
0034
0035            std::string GetAddress();
0036
0037            void SetPhone1(std::string phone1);
0038
0039            std::string GetPhone1();
0040
0041            virtual std::string GetString(int index);
0042
0043            virtual std::string GetString(std::string name);
0044
0045            virtual void SetObjectValue(int index, std::string v);
0046
0047            virtual std::string GetTableName();
0048
0049            virtual CObjectField* GetFieldInfo(int index);
0050
0051            virtual CObjectField* GetFieldInfo(std::string name);
0052
0053            virtual int GetFieldCount();
0054
```

```
0055             virtual bool SaveParent();
0056
0057             virtual bool SaveComponent();
0058
0059             virtual void CloneMemberValue(CPerson *other);
0060
0061             virtual DbObject *NewObject();
0062
0063             virtual bool SelectParent();
0064
0065             virtual std::string GetNickName();
0066
0067             virtual bool SelectComponent();
0068
0069     };// END CLASS DEFINITION CPerson
0070 } // orders
0071
0072
0073 #endif // __PERSON__
```

รูปที่ 6.19 ตัวอย่างภาษา C++ ของคลาสบุคคล

```
0001 // Static Model
0002
0003
0004 #ifndef __EMPLOYEE__
0005 #define __EMPLOYEE__
0006
0007
0008 // Include files
0009 #include "Person.h"
0010 namespace orders
0011 {
0012     class CEmployee : public CPerson
0013     {
0014
0015     private:
0016
0017             double m_salary;
0018
0019     public:
0020
0021             void SetSalary(double salary);
0022
0023             double GetSalary();
0024
0025             // framework
0026
0027             virtual std::string GetString(int index);
0028
0029             virtual std::string GetString(std::string name);
0030
0031             virtual std::string GetTableName();
0032
0033             virtual CObjectField* GetFieldInfo(int index);
0034
0035             virtual CObjectField* GetFieldInfo(std::string name);
0036
0037             virtual int GetFieldCount();
0038
0039             virtual void SetObjectValue(int index, std::string v);
0040
0041             virtual bool SaveParent();
0042
0043             virtual bool SaveComponent();
0044
0045             virtual void CloneMemberValue(CEmployee *other);
0046
0047             virtual DbObject* NewObject();
0048
0049             virtual bool SelectParent();
0050
0051             virtual bool SelectComponent();
0052
0053             virtual std::string GetNickName();
0054
0055     };// END CLASS DEFINITION CEmployee
0056 } // orders
0057
0058
0059 #endif // __EMPLOYEE__
```

รูปที่ 6.20 ตัวอย่างภาษา C++ ของคลาสพนักงาน

1. สร้างคลาสบุคคล CPerson โดยการรับทอดจาก DbObject เนื่องจากคลาสบุคคล เป็นคลาสซึ่งไม่มีคลาสแม่เหมือนคลาสสินค้าจึงไม่ขอกล่าวถึง
2. สร้างคลาสบุคคล CEmployee โดยการรับทอดจาก CPerson
3. ออกแบบคุณลักษณะและเมท็อดตามความต้องการทางธุรกิจ
4. การรับทอดเมท็อดของ DbObject สำหรับโครงร่างฯ เรียกใช้

4.1. เมท็อด GetString ทำหน้าที่คืนค่าของคุณลักษณะตามดัชนี เมื่อโครงร่างฯ ต้องการ โดยรับดัชนีคุณลักษณะ

เนื่องจากคลาสนี้มีคุณลักษณะที่ต้องการจัดเก็บในคลาส 1 คุณลักษณะ คือ เงินเดือน จึงออกแบบให้ดัชนีที่ 1 คือ เงินเดือน ดังนั้น เมื่อโครงร่างฯ ส่งค่าดัชนีใดมาก็คืนค่าของคุณลักษณะนั้นในชนิดสตริงดังตัวอย่าง

```
0017    std::string CEmployee::GetString(int index)
0018    {
0019            std::string value="";
0020
0021            switch (index) {
0022                    case 1:  // salary
0023                            char c[20];
0024                            sprintf(c, "%.2f", GetSalary());
0025                            value = c;
0026                            break;
0027            }
0028
0029            return value;
0030    }
0031
```

รูปที่ 6.21 ตัวอย่างภาษา C++ ของเมท็อด GetString รับดัชนี ของคลาสพนักงาน

4.2. เมท็อด GetString ทำหน้าที่คืนค่าของคุณลักษณะตามดัชนี เมื่อโครงร่างฯ ต้องการ โดยรับชื่อคุณลักษณะ

เนื่องจากคลาสนี้มีคุณลักษณะที่ต้องการจัดเก็บในคลาส 1 คุณลักษณะ คือ เงินเดือน จึงออกแบบให้ชื่อ salary คือ เงินเดือน ดังนั้น เมื่อโครงร่างฯ ส่งค่าดัชนีใดมาก็คืนค่าของคุณลักษณะนั้นในชนิดสตริงดังตัวอย่าง

```
0032    std::string CEmployee::GetString(std::string name)
0033    {
0034            int index=-1;
0035
0036            if (name == "salary")
0037                    index = 1;
0038
0039            return GetString(index);
0040    }
0041
```

รูปที่ 6.22 ตัวอย่างภาษา C++ ของเมท็อด GetString รับชื่อ ของคลาสพนักงาน

4.3. เมท็อด GetTableName หน้าที่คืนค่า ชื่อตารางที่เก็บสินค้า เมื่อโครงร่างฯ ต้องการทราบ

```
0042    std::string CEmployee::GetTableName()
0043    {
0044            return std::string("Employee");
0045    }
0046
```

รูปที่ 6.23 ตัวอย่างภาษา C++ ของเมท็อด GetTableName ของคลาสพนักงาน

4.4. เมท็อด GetFieldInfo หน้าที่คืนค่าวัตถุ CObjectField ซึ่งเป็นวัตถุรายละเอียดของคุณลักษณะในตาราง ซึ่งรับค่าดัชนี เมื่อโครงร่างฯ ต้องการทราบ

```
0047    CObjectField* CEmployee::GetFieldInfo(int index)
0048    {
0049            CObjectField* pField = NULL;
0050            if (index >= 1 && index <= GetFieldCount()) {
0051                    pField = new CObjectField();
0052                    switch (index) {
0053                    case 1:
0054                            pField->SetName("salary");
0055                            pField->SetType(framework::DataType::numeric);
0056                            break;
0057                    }
0058            }
0059
0060            return pField;
0061    }
0062
```

รูปที่ 6.23 ตัวอย่างภาษา C++ ของเมท็อด GetFieldInfo รับดัชนี ของคลาสพนักงาน

4.5. เมท็อด GetFieldInfo หน้าที่คืนค่าวัตถุ CObjectField ซึ่งเป็นวัตถุรายละเอียดของคุณลักษณะในตาราง ซึ่งรับค่าชื่อคุณลักษณะ เมื่อโครงร่างฯ ต้องการทราบ

```
0063    CObjectField* CEmployee::GetFieldInfo(std::string name)
0064    {
0065            int fldIndex = -1;
0066
0067            if (name == std::string("salary"))
0068                    fldIndex = 1;
0069
0070            return GetFieldInfo(fldIndex);
0071    }
0072
```

รูปที่ 6.24 ตัวอย่างภาษา C++ ของเมท็อด GetFieldInfo รับชื่อ ของคลาสพนักงาน

4.6. เมท็อด GetFieldCount หน้าที่คืนค่า จำนวนคุณลักษณะที่ต้องการจัดเก็บ เมื่อโครงร่างฯ ต้องการทราบ

```
0073    int CEmployee::GetFieldCount()
0074    {
0075            return 1;
0076    }
0077
```

รูปที่ 6.25 ตัวอย่างภาษา C++ ของเมท็อด GetFieldCount ของคลาสพนักงาน

4.7. เมท็อด SetObjectValue หน้าที่กำหนดค่าให้คุณลักษณะ โดยรับดัชนีที่ต้องการกำหนดค่าและค่าที่ต้องการกำหนดให้คุณลักษณะ เมื่อโครงร่างฯ ต้องการกำหนด

```
0078    void CEmployee::SetObjectValue(int index, std::string v)
0079    {
0080            if (index >= 0 && index <= GetFieldCount()) {
0081                    switch (index) {
0082                            // framework
0083                    case 0:
0084                            SetOID(atol(v.data()));
0085                            break;
0086                            // person
0087                    case 1:
0088                            SetSalary(atof(v.data()));
0089                            break;
0090                    }
0091            }
0092    }
0093
```

รูปที่ 6.26 ตัวอย่างภาษา C++ ของเมท็อด SetObjectValue ของคลาสพนักงาน

4.8. เมท็อด SaveParent

เนื่องจากคลาสพนักงานมีคลาสบุคคลเป็นคลาสแม่ โครงร่างฯ กำหนดให้สร้างคลาสแม่ชั่วคราวแล้วสำเนาค่าของคุณลักษณะจากคลาสลูกไปยังคลาสแม่แล้วทำการบันทึกคลาสแม่ก่อน แล้วทำลายคลาสแม่ชั่วคราว ดังตัวอย่าง

```
0094    bool CEmployee::SaveParent()
0095    {
0096            bool r = false;
0097
0098            // parent
0099            CPerson *pObj = new CPerson();
0100            pObj->CloneMemberValue((CPerson*)this);
0101            r = pObj->Save();
0102            delete pObj;
0103
0104            return r;
0105    }
0106
```

รูปที่ 6.27 ตัวอย่างภาษา C++ ของเมท็อด SaveParent ของคลาสพนักงาน

4.9. เมท็อด SaveComponent เนื่องจากไม่มีคลาสประกอบ จึงคืนค่า false ให้โครงร่างฯ เพื่อบอกว่าไม่มีคลาสประกอบ

```
0107    bool CEmployee::SaveComponent()
0108    {
0109            bool r = true;
0110
0111            return r;
0112    }
0113
```

รูปที่ 6.28 ตัวอย่างภาษา C++ ของเมท็อด SaveComponent ของคลาสพนักงาน

4.10. เมท็อด CloneMemberValue หน้าที่สำเนาเฉพาะค่าในคุณลักษณะของวัตถุหนึ่งให้อีกวัตถุหนึ่ง โดยโครงร่างฯกำหนดให้เรียกเมท็อด CloneMemberValue ของคลาสแม่ด้วยเสมอ ดังบรรทัด 160

```
0114    void CEmployee::CloneMemberValue(CEmployee *other)
0115    {
0116            SetSalary(other->GetSalary());
0117
0118            __super::CloneMemberValue(other);
0119    }
0120
```

รูปที่ 6.29 ตัวอย่างภาษา C++ ของเมท็อด CloneMemberValue ของคลาสพนักงาน

4.11. เมท็อด NewObject ทำหน้าที่สร้างวัตถุพนักงานใหม่เมื่อโครงร่างฯ ต้องการ โดยต้องการกำหนดวัตถุสำหรับการเชื่อมต่อขณะนั้นให้วัตถุใหม่เสมอ ดังบรรทัดที่ 184 และต้องแปลงเป็นวัตถุของคลาส DbObject เสมอ ดังบรรทัด 185

```
0148    DbObject* CEmployee::NewObject()
0149    {
0150            CEmployee *pObj = new CEmployee();
0151            pObj->SetConnect(GetConnect());
0152            return static_cast<DbObject*>(pObj);
0153    }
0154
```

รูปที่ 6.30 ตัวอย่างภาษา C++ ของเมท็อด NewObject ของคลาสพนักงาน

4.12. เมท็อด SelectParent ทำหน้าที่อ่านค่าคุณลักษณะของวัตถุของคลาสแม่ เนื่องจากคลาสพนักงานการรับทอดมาจากคลาสบุคคล จึงทำการสร้างวัตถุของคลาสแม่ทำการสำเนาค่าคุณลักษณะจากวัตถุไป จากนั้นให้วัตถุคลาสแม่ทำการดึงค่าคุณลักษณะของวัตถุของคลาสแม่โดยเมท็อด GetMemberValue จากนั้นจึงทำการสำเนาค่าคุณลักษณะกลับไปยังวัตถุ แล้วทำลายวัตถุของคลาสแม่ชั่วคราว เมื่อโครงร่างฯ ต้องการ โดยโครงร่างกำหนดให้เรียกเมท็อดของคลาสแม่เสมอ ดังบรรทัด 134

```
0121    bool CEmployee::SelectParent()
0122    {
0123            bool r = false;
0124
0125            // parent
0126            CPerson *pObj = new CPerson();
0127            pObj->CloneMemberValue((CPerson*)this);
0128            pObj->GetMemberValue();
0129
0130            ((CPerson*)this)->CloneMemberValue(pObj);
0131
0132            delete pObj;
0133
0134            __super::SelectParent();
0135
0136            return r;
0137    }
0138
```

รูปที่ 6.31 ตัวอย่างภาษา C++ ของเมท็อด SelectParent ของคลาสพนักงาน

4.13. เมท็อด GetNickName ทำหน้าที่คือค่าชื่อของวัตถุ สำหรับคลาสพนักงานใช้ชื่อพนักงาน

```
0155    std::string CEmployee::GetNickName()
0156    {
0157            return GetName();
0158
0159    }
0160
```

รูปที่ 6.32 ตัวอย่างภาษา C++ ของเมท็อด GetNickName ของคลาสพนักงาน

4.14. เมท็อด SelectComponent ทำหน้าที่อ่านค่าคุณลักษณะของวัตถุประกอบเมื่อโครงร่างฯ ต้องการ โดยโครงร่างกำหนดให้เรียกเมท็อดของคลาสแม่เสมอ ดังบรรทัด 143

```
0139    bool CEmployee::SelectComponent()
0140    {
0141            bool r = false;
0142
0143            r = __super::SelectComponent();
0144
0145            return r;
0146    }
0147
```

รูปที่ 6.33 ตัวอย่างภาษา C++ ของเมท็อด SelectComponent ของคลาสพนักงาน

6.4.3. ลักษณะแผนภาพคลาสแบบภาพรวมกลุ่มคลาส (Aggregation Class) หมายถึง คลาสซึ่งอ้างถึงคลาสอื่น

เช่น คลาสลูกค้า (CCustomer) มีคลาสพนักงานขาย กล่าวคือ ลูกค้าต้องพนักงานขายดูแล ดังนั้นคลาสลูกค้าจึงอ้างถึงคลาสพนักงานขาย แต่คลาสพนักงานขายไม่ได้มีวงจรชีวิตเดียวกับคลาสลูกค้า เป็นต้น

```
0001 // Static Model
0002
0003
0004 #ifndef __CUSTOMER__
0005 #define __CUSTOMER__
0006
0007
0008 // Include files
0009 #include "Person.h"
0010 #include "Sales.h"
0011 namespace orders
0012 {
0013    class CCustomer : public CPerson
0014    {
0015
0016    private:
0017
0018            double m_credit_limit;
0019
0020            orders::CSales* m_pSales;
0021
0022    public:
0023            CCustomer();
0024
0025            ~CCustomer();
0026
0027            void SetCreditLimit(double credit_limit);
0028
0029            double GetCreditLimit();
0030
0031            void SetSales(CSales* pSales);
0032
0033            CSales* GetSales();
0034
0035            virtual std::string GetString(int index);
0036
0037            virtual std::string GetString(std::string name);
0038
0039            virtual void SetObjectValue(int index, std::string v);
```

```
0040
0041            virtual std::string GetTableName();
0042
0043            virtual CObjectField* GetFieldInfo(int index);
0044
0045            virtual CObjectField* GetFieldInfo(std::string name);
0046
0047            virtual int GetFieldCount();
0048
0049            virtual bool SaveParent();
0050
0051            virtual bool SaveComponent();
0052
0053            virtual void CloneMemberValue(CCustomer *other);
0054
0055            virtual DbObject* NewObject();
0056
0057            virtual bool SelectParent();
0058
0059            virtual std::string GetNickName();
0060
0061            // get component
0062
0063            virtual bool SelectComponent();
0064
0065            CSales* GetSalesOID(long oid);
0066
0067    };// END CLASS DEFINITION CCustomer
0068 } // orders
0069
0070
0071 #endif // __CUSTOMER__
```

รูปที่ 6.34 ตัวอย่างภาษา C++ ของคลาสลูกค้า

1. สร้างคลาสบุคคล CCustomer โดยการรับทอดจาก CPerson
2. ออกแบบคุณลักษณะและเมท็อดตามความต้องการทางธุรกิจ
3. การรับทอดเมท็อดของ DbObject สำหรับโครงร่างฯ เรียกใช้

    3.1. เมท็อด GetString ทำหน้าที่คืนค่าของคุณลักษณะตามดัชนี เมื่อโครงร่างฯ ต้องการ โดยรับดัชนีคุณลักษณะ

เนื่องจากคลาสนี้มีคุณลักษณะที่ต้องการจัดเก็บในคลาส 2 คุณลักษณะ คือ วงเงินสินเชื่อและพนักงานขาย จึงออกแบบให้ดัชนีที่ 1 คือ วงเงินสินเชื่อ และ 2 คือ พนักงานขาย ดังตัวอย่าง

```
0035    std::string CCustomer::GetString(int index)
0036    {
0037            std::string value="";
0038            char c[20];
0039
0040            switch (index) {
0041                    case 1:  // credit limit
0042                             sprintf(c, "%.2f", GetCreditLimit());
0043                             value = c;
0044                             break;
0045                    case 2:  // sales
0046                             sprintf(c, "%d", GetSales()->GetOID());
0047                             value = c;
0048                             break;
0049            }
0050
0051            return value;
0052    }
0053
```

รูปที่ 6.35 ตัวอย่างภาษา C++ ของเมท็อด GetString รับดัชนี ของคลาสลูกค้า

3.2. เมท็อด GetString ทำหน้าที่คืนค่าของคุณลักษณะตามดัชนี เมื่อโครงร่างฯ ต้องการ โดยรับชื่อคุณลักษณะ

เนื่องจากคลาสนี้มีคุณลักษณะที่ต้องการจัดเก็บในคลาส 2 คุณลักษณะ คือ วงเงินสินเชื่อและพนักงานขาย จึงออกแบบให้ชื่อ credit_limit คือ วงเงินสินเชื่อ และ ชื่อ sales คือ พนักงานขาย ดังตัวอย่าง

```
0054    std::string CCustomer::GetString(std::string name)
0055    {
0056            int index=-1;
0057
0058            if (name == "credit_limit")
0059                    index = 1;
0060            else if (name == "sales")
0061                    index = 2;
0062
0063            return GetString(index);
0064    }
0065
```

รูปที่ 6.36 ตัวอย่างภาษา C++ ของเมท็อด GetString รับชื่อ ของคลาสลูกค้า

3.3. เมท็อด GetTableName หน้าที่คืนค่า ชื่อตารางลูกค้า เมื่อโครงร่างฯ ต้องการทราบ

```
0066    std::string CCustomer::GetTableName()
0067    {
0068            return std::string("Customer");
0069    }
0070
```

รูปที่ 6.37 ตัวอย่างภาษา C++ ของเมท็อด GetTableName ของคลาสลูกค้า

3.4. เมท็อด GetFieldInfo หน้าที่คืนค่าวัตถุ CObjectField ซึ่งเป็นวัตถุรายละเอียดของคุณลักษณะในตาราง ซึ่งรับค่าดัชนี เมื่อโครงร่างฯ ต้องการทราบ

```
0071    CObjectField* CCustomer::GetFieldInfo(int index)
0072    {
0073            CObjectField* pField = NULL;
0074            if (index >= 1 && index <= GetFieldCount()) {
0075                    pField = new CObjectField();
0076                    switch (index) {
0077                    case 1:
0078                            pField->SetName("credit_limit");
0079                            pField->SetType(framework::DataType::numeric);
0080                            break;
0081                    case 2:
0082                            pField->SetName("sales");
0083                            pField->SetType(framework::DataType::numeric);
0084                            break;
0085                    }
0086            }
0087
0088            return pField;
0089    }
0090
```

รูปที่ 6.38 ตัวอย่างภาษา C++ ของเมท็อด GetFieldInfo รับดัชนี ของคลาสลูกค้า

3.5. เมท็อด GetFieldInfo หน้าที่คืนค่าวัตถุ CObjectField ซึ่งเป็นวัตถุรายละเอียดของคุณลักษณะในตาราง ซึ่งรับค่าชื่อคุณลักษณะ เมื่อโครงร่างฯ ต้องการทราบ

```
0091    CObjectField* CCustomer::GetFieldInfo(std::string name)
```

```
0092    {
0093            int fldIndex = -1;
0094
0095            if (name == std::string("credit_limit"))
0096                    fldIndex = 1;
0097            else if (name == std::string("sales"))
0098                    fldIndex = 2;
0099
0100            return GetFieldInfo(fldIndex);
0101    }
0102
```

รูปที่ 6.39 ตัวอย่างภาษา C++ ของเมท็อด GetFieldInfo รับชื่อ ของคลาสลูกค้า

3.6. เมท็อด GetFieldCount หน้าที่คืนค่า จำนวนคุณลักษณะที่ต้องการจัดเก็บ เมื่อโครงร่างฯ ต้องการทราบ

```
0103    int CCustomer::GetFieldCount()
0104    {
0105            return 2;
0106    }
0107
```

รูปที่ 6.40 ตัวอย่างภาษา C++ ของเมท็อด GetFieldCount ของคลาสลูกค้า

3.7. เมท็อด SetObjectValue หน้าที่กำหนดค่าให้คุณลักษณะ โดยรับดัชนีที่ต้องการกำหนดค่าและค่าที่ต้องการกำหนดให้คุณลักษณะ เมื่อโครงร่างฯ ต้องการกำหนด

```
0108    void CCustomer::SetObjectValue(int index, std::string v)
0109    {
0110            if (index >= 0 && index <= GetFieldCount()) {
0111                    switch (index) {
0112                            // framework
0113                    case 0:
0114                            SetOID(atol(v.data()));
0115                            break;
0116                    case 1:
0117                            SetCreditLimit(atof(v.data()));
0118                            break;
0119                    case 2:
0120                            SetSales(GetSalesOID(atol(v.data())));
0121                            break;
0122                    }
0123            }
0124    }
0125
```

รูปที่ 6.41 ตัวอย่างภาษา C++ ของเมท็อด SetObjectValue ของคลาสลูกค้า

3.8. เมท็อด SaveParent

เนื่องจากคลาสลูกค้ามีคลาสบุคคลเป็นคลาสแม่ โครงร่างฯ กำหนดให้สร้างคลาสแม่ชั่วคราวแล้วสำเนาค่าของคุณลักษณะจากคลาสลูกไปยังคลาสแม่แล้วทำการบันทึกคลาสแม่ก่อน แล้วทำคลาสแม่ชั่วคราว ดังตัวอย่าง

```
0126    bool CCustomer::SaveParent()
0127    {
0128            bool r = false;
0129
0130            // parent
0131            CPerson *pObj = new CPerson();
0132            pObj->CloneMemberValue((CPerson*)this);
0133            r = pObj->Save();
0134            delete pObj;
0135
0136            return r;
```

```
0137    }
0138
```

รูปที่ 6.42 ตัวอย่างภาษา C++ ของเมท็อด SaveParent ของคลาสลูกค้า

3.9. เมท็อด SaveComponent เนื่องจากไม่มีคลาสประกอบ จึงคืนค่า false ให้โครงร่างฯ เพื่อบอกว่าไม่มีคลาสประกอบ

```
0139    bool CCustomer::SaveComponent()
0140    {
0141            bool r = true;
0142            return r;
0143    }
0144
```

รูปที่ 6.43 ตัวอย่างภาษา C++ ของเมท็อด SaveComponent ของคลาสลูกค้า

3.10. เมท็อด CloneMemberValue หน้าที่สำเนาเฉพาะค่าในคุณลักษณะของวัตถุหนึ่งให้อีกวัตถุหนึ่ง สังเกตว่าต้องทำการกำหนดให้ครบทุกคุณลักษณะ โดยโครงร่างฯกำหนดให้เรียกเมท็อด CloneMemberValue ของคลาสแม่ด้วยเสมอ ดังบรรทัด 160

```
0145    void CCustomer::CloneMemberValue(CCustomer *other)
0146    {
0147            SetCreditLimit(other->GetCreditLimit());
0148            SetSales(other->GetSales());
0149
0150            __super::CloneMemberValue(other);
0151    }
0152
```

รูปที่ 6.44 ตัวอย่างภาษา C++ ของเมท็อด CloneMemberValue ของคลาสลูกค้า

3.11. เมท็อด NewObject ทำหน้าที่สร้างวัตถุลูกค้าใหม่เมื่อโครงร่างฯ ต้องการ โดยต้องการกำหนดวัตถุสำหรับการเชื่อมต่อขณะนั้นให้วัตถุใหม่เสมอ ดังบรรทัดที่ 176 และต้องแปลงเป็นวัตถุของคลาส DbObject เสมอ ดังบรรทัด 178

```
0173    DbObject* CCustomer::NewObject()
0174    {
0175            CCustomer *pObj = new CCustomer();
0176            pObj->SetConnect(GetConnect());
0177            return static_cast<DbObject*>(pObj);
0178    }
0179
```

รูปที่ 6.45 ตัวอย่างภาษา C++ ของเมท็อด NewObject ของคลาสลูกค้า

3.12. เมท็อด SelectParent ทำหน้าที่อ่านค่าคุณลักษณะของวัตถุของคลาสแม่ เนื่องจากคลาสลูกค้าการรับทอดมาจากคลาสบุคคล จึงทำการสร้างวัตถุของคลาสแม่ทำการสำเนาค่าคุณลักษณะจากวัตถุไป จากนั้นให้วัตถุคลาสแม่ทำการดึงค่าคุณลักษณะของวัตถุของคลาสแม่โดยเมท็อด GetMemberValue จากนั้นเปลี่ยนวัตถุให้เป็นวัตถุบุคคลและทำการสำเนาค่าคุณลักษณะกลับไปยังวัตถุ แล้วทำลาย

วัตถุของคลาสแม่ชั่วคราว เมื่อโครงร่างฯ ต้องการ โดยโครงร่างกำหนดให้เรียกเมท็อดของคลาสแม่เสมอ ดังบรรทัด 168

```
0153    bool CCustomer::SelectParent()
0154    {
0155            bool r = false;
0156
0157            // parent
0158            CPerson *pObj = new CPerson();
0159            pObj->CloneMemberValue((CPerson*)this);
0160            pObj->GetMemberValue();
0161
0162            // customer
0163            CPerson *pCusPerson = (CPerson*)this;
0164            pCusPerson->CloneMemberValue(pObj);
0165
0166            delete pObj;
0167
0168            __super::SelectParent();
0169
0170            return r;
0171    }
0172
```

รูปที่ 6.46 ตัวอย่างภาษา C++ ของเมท็อด SelectParent ของคลาสลูกค้า

3.13. เมท็อด GetNickName ทำหน้าที่คือค่าชื่อของวัตถุ สำหรับคลาสพนักงาน ใช้ชื่อพนักงาน

```
0201    std::string CCustomer::GetNickName()
0202    {
0203            return GetName();
0204
0205    }
0206
```

รูปที่ 6.47 ตัวอย่างภาษา C++ ของเมท็อด GetNickName ของคลาสลูกค้า

3.14. เมท็อด SelectComponent ทำหน้าที่อ่านค่าคุณลักษณะของวัตถุประกอบ เมื่อโครงร่างฯ ต้องการ โดยโครงร่างกำหนดให้เรียกเมท็อดของคลาสแม่เสมอ ดังบรรทัด 196

```
0192    bool CCustomer::SelectComponent()
0193    {
0194            bool r = false;
0195
0196            r = __super::SelectComponent();
0197
0198            return r;
0199    }
0200
```

รูปที่ 6.48 ตัวอย่างภาษา C++ ของเมท็อด SelectComponent ของคลาสลูกค้า

3.15. เมท็อด GetSalesOID เป็นเมท็อดเพิ่มสำหรับคลาสที่มีการอ้างถึงคลาสอื่นเพื่อใช้ในการสร้างวัตถุที่อ้างถึงโดยใช้หมายเลขวัตถุ และโครงร่างฯ กำหนดให้เรียก PushAssociate เพื่อโครงร่างฯ ว่ามีการสร้างวัตถุขึ้นมา

```
0180    CSales* CCustomer::GetSalesOID(long oid)
0181    {
0182            CSales *sales = new CSales();
0183
0184            sales->SetConnect(GetConnect());
```

```
0185            sales->SetOID(oid);
0186            if (sales->GetMemberValue())
0187                    PushAssociate(sales); // framework
0188
0189            return sales;
0190    }
0191
```

รูปที่ 6.49 ตัวอย่างภาษา C++ ของเมท็อด GetSaleOID ของคลาสลูกค้า

6.4.4. ลักษณะแผนภาพคลาสแบบคอมโพสิตชันคลาส (Composition Class)

หมายถึง คลาสซึ่งมีคลาสอื่นเป็นองค์ประกอบ

เช่น คลาสการสั่งซื้อ (COrders) มีประกอบคลาสรายการสั่งซื้อ (COrderItem) กล่าวคือ การสั่งซื้อจะประกอบด้วยรายการสั่งซื้อ โดยวงจรชีวิตการเกิดการตายมีความสัมพันธ์กัน ด้วยการสั่งซื้อเกิดก่อนรายการการสั่งซื้อ และ การสั่งซื้อจะตายก่อนการสั่งซื้อ

```
0001 // Static Model
0002
0003
0004 #ifndef __ORDERS__
0005 #define __ORDERS__
0006
0007
0008 // Include files
0009 #include "..\..\..\oofardb\thesis\framework\Persistance.h"
0010 #include "Customer.h"
0011 #include "OrderItem.h"
0012 #include <vector>
0013 #include <string>
0014
0015 namespace orders
0016 {
0017     class COrderItem;
0018     class COrders : public framework::DbObject
0019     {
0020
0021     private:
0022
0023             std::string m_order_no;
0024
0025             unsigned long m_date;
0026
0027             std::vector<DbObject*> m_pOrderItems;
0028
0029             orders::CCustomer* m_pCustomer;
0030
0031     public:
0032
0033             COrders();
0034
0035             ~COrders();
0036
0037             void SetOrderNo(std::string order_no);
0038
0039             std::string GetOrderNo();
0040
0041             void SetDate(unsigned long date);
0042
0043             unsigned long GetDate();
0044
0045             void SetCustomer(CCustomer *pCustomer);
0046
0047             CCustomer *GetCustomer();
0048
0049             void SetOrderItems(std::vector<DbObject*>* pOrderItem);
0050
0051             std::vector<DbObject*>* GetOrderItems();
0052
0053             void DeleteAllOrderItems();
0054
0055             int AddOrderItem(COrderItem *pItem);
0056
0057             void RemoveOrderItem(int n);
0058
0059             void UpdateOrderItem(int n, COrderItem *pItem);
0060
0061             virtual std::string GetString(int index);
0062
```

```
0063            virtual std::string GetString(std::string name);
0064
0065            virtual std::string GetTableName();
0066
0067            virtual CObjectField* GetFieldInfo(int index);
0068
0069            virtual CObjectField* GetFieldInfo(std::string name);
0070
0071            virtual int GetFieldCount();
0072
0073            virtual DbObject* NewObject();
0074
0075            virtual void SetObjectValue(int index, std::string v);
0076
0077            virtual bool SelectParent();
0078
0079            virtual bool SaveParent();
0080
0081            virtual bool SaveComponent();
0082
0083            virtual void CloneMemberValue(COrders* other);
0084
0085            virtual std::string GetNickName();
0086
0087            // get component
0088
0089            virtual bool SelectComponent();
0090
0091            CCustomer* GetCustomerOID(long oid);
0092
0093    };// END CLASS DEFINITION COrders
0094 } // orders
0095
0096
0097 #endif // __ORDERS__
```

รูปที่ 6.50 ตัวอย่างภาษา C++ ของคลาสการสั่งซื้อ

1. สร้างคลาสการสั่งซื้อ COrders โดยการรับทอดจาก DbObject
2. ออกแบบคุณลักษณะและเมท็อดตามความต้องการทางธุรกิจ
3. เพิ่มตัวแปรเพื่อเก็บชุดของวัตถุรายการสั่งซื้อ ดังบรรทัด 27
4. การรับทอดเมท็อดของ DbObject สำหรับโครงร่างฯ เรียกใช้

    4.1. เมท็อด GetString ทำหน้าที่คืนค่าของคุณลักษณะตามดัชนี เมื่อโครงร่างฯ ต้องการ โดยรับดัชนีคุณลักษณะ

เนื่องจากคลาสนี้มีคุณลักษณะที่ต้องการจัดเก็บในคลาส 3 คุณลักษณะ คือ เลขที่การสั่งซื้อ, วันที่สั่งซื้อ และลูกค้า จึงออกแบบให้ดัชนีที่ 1 คือ เลขที่การสั่งซื้อ, 2 คือ วันที่สั่งซื้อ และ 3 คือ ลูกค้า ดังตัวอย่าง

```
0105    std::string COrders::GetString(int index)
0106    {
0107            std::string value="";
0108            char c[20];
0109
0110            switch (index) {
0111                    case 1:  // order no
0112                            value = GetOrderNo();
0113                            break;
0114                    case 2:  // order date
0115                            sprintf(c, "%d", GetDate());
0116                            value = c;
0117                            break;
0118                    case 3:  // customer
0119                            sprintf(c, "%d", GetCustomer()->GetOID());
0120                            value = c;
0121                            break;
0122            }
0123
0124            return value;
0125    }
0126
```

รูปที่ 6.51 ตัวอย่างภาษา C++ ของเมท็อด GetString รับดัชนี คลาสการสั่งซื้อ

4.2. เมท็อด GetString ทำหน้าที่คืนค่าของคุณลักษณะตามดัชนี เมื่อโครงร่างฯ ต้องการ โดยรับชื่อคุณลักษณะ

เนื่องจากคลาสนี้มีคุณลักษณะที่ต้องการจัดเก็บในคลาส 2 คุณลักษณะ คือ วงเงินสินเชื่อและพนักงานขาย จึงออกแบบให้ชื่อ credit_limit คือ วงเงินสินเชื่อ และ ชื่อ sales คือ พนักงานขาย ดังตัวอย่าง

```
0127    std::string COrders::GetString(std::string name)
0128    {
0129            int index=-1;
0130
0131            if (name == "order_no")
0132                    index = 1;
0133            else if (name == "order_date")
0134                    index = 2;
0135            else if (name == "customer")
0136                    index = 3;
0137
0138            return GetString(index);
0139    }
0140
```

รูปที่ 6.52 ตัวอย่างภาษา C++ ของเมท็อด GetString รับชื่อ คลาสการสั่งซื้อ

4.3. เมท็อด GetTableName หน้าที่คืนค่า ชื่อตารางการสั่งซื้อ เมื่อโครงร่างฯ ต้องการทราบ

```
0141    std::string COrders::GetTableName()
0142    {
0143            return std::string("Orders");
0144    }
0145
```

รูปที่ 6.53 ตัวอย่างภาษา C++ ของเมท็อด GetTableName คลาสการสั่งซื้อ

4.4. เมท็อด GetFieldInfo หน้าที่คืนค่าวัตถุ CObjectField ซึ่งเป็นวัตถุรายละเอียดของคุณลักษณะในตาราง ซึ่งรับค่าดัชนี เมื่อโครงร่างฯ ต้องการทราบ

```
0146    CObjectField* COrders::GetFieldInfo(int index)
0147    {
0148            CObjectField* pField = NULL;
0149            if (index >= 1 && index <= GetFieldCount()) {
0150                    pField = new CObjectField();
0151                    switch (index) {
0152                    case 1:
0153                            pField->SetName("order_no");
0154                            pField->SetType(framework::character);
0155                            break;
0156                    case 2:
0157                            pField->SetName("order_date");
0158                            pField->SetType(framework::numeric);
0159                            break;
0160                    case 3:
0161                            pField->SetName("customer");
0162                            pField->SetType(framework::numeric);
0163                            break;
0164                    }
0165            }
0166
0167            return pField;
0168    }
0169
```

รูปที่ 6.54 ตัวอย่างภาษา C++ ของเมท็อด GetFieldInfo รับดัชนี คลาสการสั่งซื้อ

4.5. เมท็อด GetFieldInfo หน้าที่คืนค่าวัตถุ CObjectField ซึ่งเป็นวัตถุรายละเอียดของคุณลักษณะในตาราง ซึ่งรับค่าชื่อคุณลักษณะ เมื่อโครงร่างฯ ต้องการทราบ

```
0170    CObjectField* COrders::GetFieldInfo(std::string name)
0171    {
0172            int fldIndex = -1;
0173
0174            if (name == std::string("order_no"))
0175                    fldIndex = 1;
0176            else if (name == std::string("order_date"))
0177                    fldIndex = 2;
0178            else if (name == std::string("customer"))
0179                    fldIndex = 3;
0180
0181            return GetFieldInfo(fldIndex);
0182    }
0183
```

รูปที่ 6.55 ตัวอย่างภาษา C++ ของเมท็อด GetFieldInfo รับชื่อ คลาสการสั่งซื้อ

4.6. เมท็อด GetFieldCount หน้าที่คืนค่า จำนวนคุณลักษณะที่ต้องการจัดเก็บ เมื่อโครงร่างฯ ต้องการทราบ

```
0184    int COrders::GetFieldCount()
0185    {
0186            return 3;
0187    }
0188
```

รูปที่ 6.56 ตัวอย่างภาษา C++ ของเมท็อด GetFieldCount คลาสการสั่งซื้อ

4.7. เมท็อด SetObjectValue หน้าที่กำหนดค่าให้คุณลักษณะ โดยรับดัชนีที่ต้องการกำหนดค่าและค่าที่ต้องการกำหนดให้คุณลักษณะ เมื่อโครงร่างฯ ต้องการกำหนด

```
0189    void COrders::SetObjectValue(int index, std::string v)
0190    {
0191            if (index >= 0 && index <= GetFieldCount()) {
0192                    switch (index) {
0193                    case 0:
0194                            SetOID(atol(v.data()));
0195                            break;
0196                    case 1:
0197                            SetOrderNo(v);
0198                            break;
0199                    case 2:
0200                            SetDate(atol(v.data()));
0201                            break;
0202                    case 3:
0203                            SetCustomer(GetCustomerOID(atol(v.data())));
0204                            break;
0205                    }
0206            }
0207    }
0208
```

รูปที่ 6.57 ตัวอย่างภาษา C++ ของเมท็อด SetObjectValue คลาสการสั่งซื้อ

4.8. เมท็อด SaveParent

เนื่องจากคลาสการสั่งซื้อไม่มีคลาสแม่ จึงส่งค่า False บอกโครงร่างฯ ว่าไม่มีคลาสแม่ ดังตัวอย่าง

```
0209    bool COrders::SaveParent()
0210    {
0211            bool r = false;
0212
0213            return r;
0214    }
0215
```

รูปที่ 6.58 ตัวอย่างภาษา C++ ของเมท็อด SaveParent คลาสการสั่งซื้อ

4.9. เมท็อด SaveComponent เนื่องจากคลาสสั่งซื้อมีคลาสรายการสั่งซื้อเป็นองค์ประกอบดังนั้นจึงต้องทำการบันทึกรายการสั่งซื้อด้วย ดังบรรทัดที่ 221-228

```
0216    bool COrders::SaveComponent()
0217    {
0218            bool r = true;
0219
0220            std::vector <DbObject*>::iterator iter;
0221            COrderItem *item;
0222
0223            for (iter = m_pOrderItems.begin(); iter < m_pOrderItems.end();
iter++)  {
0224                    item = static_cast<COrderItem*>(*iter);
0225                    //item->SetLifeStatus(GetLifeStatus());
0226                    item->Save();
0227            }
0228
0229            return r;
0230    }
0231
```

รูปที่ 6.59 ตัวอย่างภาษา C++ ของเมท็อด SaveComponent คลาสการสั่งซื้อ

4.10. เมท็อด CloneMemberValue หน้าที่สำเนาเฉพาะค่าในคุณลักษณะของวัตถุหนึ่งให้อีกวัตถุหนึ่ง สังเกตว่าต้องทำการกำหนดให้ครบทุกคุณลักษณะ โดยโครงร่างฯกำหนดให้เรียกเมท็อด CloneMemberValue ของคลาสแม่ด้วยเสมอ ดังบรรทัด 257

```
0250    void COrders::CloneMemberValue(COrders *other)
0251    {
0252            SetOrderNo(other->GetOrderNo());
0253            SetDate(other->GetDate());
0254            SetCustomer(other->GetCustomer());
0255            SetOrderItems(other->GetOrderItems());
0256
0257            __super::CloneMemberValue(other);
0258    }
0259
```

รูปที่ 6.60 ตัวอย่างภาษา C++ ของเมท็อด CloneMemberValue คลาสการสั่งซื้อ

4.11. เมท็อด NewObject ทำหน้าที่สร้างวัตถุลูกค้าใหม่เมื่อโครงร่างฯ ต้องการโดยต้องการกำหนดวัตถุสำหรับการเชื่อมต่อขณะนั้นให้วัตถุใหม่เสมอ ดัง

บรรทัดที่ 176 และต้องแปลงเป็นวัตถุของคลาส DbObject เสมอ ดังบรรทัด 178

```
0266    DbObject* COrders::NewObject()
0267    {
0268            COrders *pObj = new COrders();
0269            pObj->SetConnect(GetConnect());
0270            return static_cast<DbObject*>(pObj);
0271    }
0272
```

รูปที่ 6.61 ตัวอย่างภาษา C++ ของเมท็อด NewObject คลาสการสั่งซื้อ

4.12. เมท็อด SelectParent ทำหน้าที่อ่านค่าคุณลักษณะของวัตถุของคลาสแม่ เนื่องจากคลาสลูกค้าการรับทอดมาจากคลาสบุคคล จึงทำการสร้างวัตถุของคลาสแม่ทำการสำเนาค่าคุณลักษณะจากวัตถุไป จากนั้นให้วัตถุคลาสแม่ทำการดึงค่าคุณลักษณะของวัตถุของคลาสแม่โดยเมท็อด GetMemberValue จากนั้นเปลี่ยนวัตถุให้เป็นวัตถุบุคคลและทำการสำเนาค่าคุณลักษณะกลับไปยังวัตถุ แล้วทำลายวัตถุของคลาสแม่ชั่วคราว เมื่อโครงร่างฯ ต้องการ โดยโครงร่างกำหนดให้เรียกเมท็อดของคลาสแม่เสมอ ดังบรรทัด 168

```
0273    bool COrders::SelectParent()
0274    {
0275            bool r = false;
0276
0277            r = __super::SelectParent();
0278
0279            return r;
0280    }
0281
```

รูปที่ 6.62 ตัวอย่างภาษา C++ ของเมท็อด SelectParent คลาสการสั่งซื้อ

4.13. เมท็อด GetNickName ทำหน้าที่คือค่าชื่อของวัตถุ สำหรับคลาสการสั่งซื้อใช้เลขที่การสั่งซื้อ

```
0260    std::string COrders::GetNickName()
0261    {
0262            return GetOrderNo();
0263
0264    }
0265
```

รูปที่ 6.63 ตัวอย่างภาษา C++ ของเมท็อด GetNickName คลาสการสั่งซื้อ

4.14. เมท็อด SelectComponent ทำหน้าที่อ่านค่าคุณลักษณะของวัตถุประกอบเมื่อโครงร่างฯ ต้องการ โดยโครงร่างกำหนดให้เรียกเมท็อดของคลาสแม่เสมอ ดังบรรทัด 245 เนื่องจากคลาสการสั่งซื้อมีคลาสรายการสั่งซื้อเป็นองค์ประกอบ จึงเพิ่มชุดคำสั่งสำหรับการเลือกรายการสั่งซื้อของการสั่งซื้อตามเลขที่การสั่งซื้อ ดังบรรทัด 241-244

```
0232    bool COrders::SelectComponent()
0233    {
0234            bool r = false;
0235
0236            COrderItem obj;
0237            char c[256];
0238
0239            DeleteAllOrderItems();
0240
0241            sprintf(c, "order_no = %s", GetOrderNo().data());
0242            obj.SetConnect(GetConnect());
0243            obj.Select(c, &m_pOrderItems);
0244
0245            r = __super::SelectComponent();
0246
0247            return r;
0248    }
0249
```

รูปที่ 6.64 ตัวอย่างภาษา C++ ของเมท็อด SelectComponent คลาสการสั่งซื้อ

4.15. เมท็อด GetCustomerOID เป็นเมท็อดเพิ่มสำหรับคลาสที่มีการอ้างถึงคลาสอื่นเพื่อใช้ในการสร้างวัตถุที่อ้างถึงโดยใช้หมายเลขวัตถุ และโครงร่างฯ กำหนดให้เรียก PushAssociate เพื่อโครงร่างฯ ว่ามีการสร้างวัตถุขึ้นมา

```
0282    CCustomer* COrders::GetCustomerOID(long oid)
0283    {
0284            CCustomer *customer = new CCustomer();
0285
0286            customer->SetConnect(GetConnect());
0287            customer->SetOID(oid);
0288            if (customer->GetMemberValue())
0289                    PushAssociate(customer); // framework
0290
0291            return customer;
0292    }
0293
```

รูปที่ 6.65 ตัวอย่างภาษา C++ ของเมท็อด GetCustomerOID คลาสการสั่งซื้อ

4.16. เมท็อด DeleteAllOrderItem เป็นเมท็อดเพิ่มสำหรับการลบรายการสั่งซื้อทั้งหมด

```
0060    void COrders::DeleteAllOrderItems()
0061    {
0062            std::vector <DbObject*>::iterator iter;
0063
0064            for (iter = m_pOrderItems.begin(); iter < m_pOrderItems.end();
iter++)
0065                    delete *iter;
0066
0067            m_pOrderItems.clear();
0068    }
0069
0070    int COrders::AddOrderItem(COrderItem *pItem)
0071    {
0072            COrderItem *newItem = new COrderItem();
0073
0074            newItem->SetConnect(GetConnect());
0075
0076            newItem->CloneMemberValue(pItem);
0077            m_pOrderItems.push_back(newItem);
0078
0079            return (int)m_pOrderItems.size();
0080    }
0081
```

รูปที่ 6.66 ตัวอย่างภาษา C++ ของเมท็อด DeleteAllOrderItems คลาสการสั่งซื้อ

4.17. เมท็อด RemoveOrderItem เป็นเมท็อดเพิ่มสำหรับการลบรายการสั่งซื้อแบบเจาะจงดัชนี

```
0082    void COrders::RemoveOrderItem(int n)
0083    {
0084            DbObject* item;
0085
0086            if (n > 0 && n < (int)m_pOrderItems.size()) {
0087                    item = m_pOrderItems.at(n);
0088                    delete item;
0089                    m_pOrderItems.erase(m_pOrderItems.begin() + n);
0090            }
0091    }
0092
```

รูปที่ 6.67 ตัวอย่างภาษา C++ ของเมท็อด RemoveOrderItem คลาสการสั่งซื้อ

4.18. เมท็อด UpdateOrderItem เป็นเมท็อดเพิ่มสำหรับการปรับปรุงรายการสั่งซื้อแบบเจาะจงดัชนี

```
0093    void COrders::UpdateOrderItem(int n, COrderItem *pItem)
0094    {
0095            if (n > 0 && n < (int)m_pOrderItems.size()) {
0096                    COrderItem *newItem = new COrderItem();
0097
0098                    newItem->SetConnect(GetConnect());
0099                    newItem->CloneMemberValue(pItem);
0100
0101                    m_pOrderItems[n] = newItem;
0102            }
0103    }
0104
```

รูปที่ 6.67 ตัวอย่างภาษา C++ ของเมท็อด UpdateOrderItem คลาสการสั่งซื้อ

6.6. คลาสโปรแกรมประยุกต์และคลาสเชื่อมต่อสำหรับระบบการสั่งซื้อ

6.5.1. คลาสโปรแกรมประยุกต์ระบบการสั่งซื้อ (SalesOrderODBCDbApp)

```
0001 // Static Model
0002
0003
0004 #ifndef __SALESORDERODBCDBAPP__
0005 #define __SALESORDERODBCDBAPP__
0006
0007
0008 // Include files
0009 #include "..\..\oofardb\thesis\odbc\ODBCDbApp.h"
0010 #include "SalesOrderODBCDbConnect.h"
0011
0012 class CSalesOrderODBCDbApp : public odbc::CODBCDbApp
0013 {
0014
0015 private:
0016
0017     CSalesOrderODBCDbConnect* m_pConnect;
0018
0019 protected:
0020
0021     CSalesOrderODBCDbConnect* NewConnect();
0022
0023     void DestroyConnect();
0024
0025 public:
0026
0027     CSalesOrderODBCDbConnect* GetConnect();
0028
0029 };// END CLASS DEFINITION CSalesOrderODBCDbApp
0030
0031 #endif // __SALESORDERODBCDBAPP__
```

รูปที่ 6.68 ตัวอย่างภาษา C++ คลาสโปรแกรมประยุกต์ระบบการสั่งซื้อสินค้า

1. สร้างคลาสการสั่งซื้อ CSalesOrderODBCDbApp โดยการรับทอดจาก CODBCDbApp
2. ออกแบบคุณลักษณะและเมท็อดตามความต้องการทางธุรกิจ
3. เพิ่มตัวแปรเพื่อเก็บวัตถุเชื่อมต่อสำหรับโปรแกรมประยุกต์นี้ ดังบรรทัด 17
4. การรับทอดเมท็อดของ CODBCDbApp สำหรับโครงร่างฯ เรียกใช้

4.1. เมท็อด NewConnect เป็นเมท็อดซึ่งสร้างวัตถุเชื่อมต่อ CSaleOrderODBCDbConnect ซึ่งเป็นวัตถุเพื่อการเชื่อมต่อไปยังฐานข้อมูลเชิงสัมพันธ์ของโปรแกรมประยุกต์ระบบการสั่งซื้อสินค้าแบบง่ายโดยเฉพาะ และส่งวัตถุเชื่อมต่อกลับไป ซึ่งเมท็อดนี้โครงร่างฯ จะเป็นผู้เรียกใช้

```
0005 CSalesOrderODBCDbConnect* CSalesOrderODBCDbApp::NewConnect()
0006 {
0007     if (m_pConnect == NULL)
0008             m_pConnect = new CSalesOrderODBCDbConnect();
0009     return GetConnect();
0010 }
0011
```

รูปที่ 6.69 ตัวอย่างภาษา C++ ของเมท็อด NewConnect คลาสโปรแกรมประยุกต์การสั่งซื้อสินค้า

4.2. เมท็อด GetConnect เป็นเมท็อดซึ่งส่งค่าวัตถุเชื่อมต่อในขณะนั้นกลับไป

```
0012 CSalesOrderODBCDbConnect* CSalesOrderODBCDbApp::GetConnect()
0013 {
0014     return m_pConnect;
0015 }
0016
```

รูปที่ 6.70 ตัวอย่างภาษา C++ ของเมท็อด GetConnect คลาสโปรแกรมฯการสั่งซื้อสินค้า

4.3. เมท็อด DestroyConnect เป็นเมท็อดซึ่งทำลายวัตถุเชื่อมต่อ

```
0017 void CSalesOrderODBCDbApp::DestroyConnect()
0018 {
0019     if (m_pConnect != NULL)
0020             delete m_pConnect;
0021 }
```

รูปที่ 6.71 ตัวอย่างภาษา C++ ของเมท็อด DestroyConnect คลาสโปรแกรมฯการสั่งซื้อสินค้า

6.5.2. คลาสการเชื่อมต่อระบบการสั่งซื้อสินค้า (SalesOrderODBCDbConnect)

```
0001 // Static Model
0002
0003
0004 #ifndef __SALESORDERODBCDBCONNECT__
0005 #define __SALESORDERODBCDBCONNECT__
0006
0007
0008 // Include files
0009 #include "..\..\oofardb\thesis\odbc\ODBCDbConnect.h"
0010 class CSalesOrderODBCDbConnect : public odbc::CODBCDbConnect
0011 {
0012
0013 public:
0014
0015     std::string GetDefaultDSN();
0016
0017 };// END CLASS DEFINITION CSalesOrderODBCDbConnect
0018
0019 #endif // __SALESORDERODBCDBCONNECT__
```

รูปที่ 6.72 ตัวอย่างภาษา C++ ของคลาสการเชื่อมต่อระบบการสั่งซื้อสินค้า

1. สร้างคลาสการสั่งซื้อ CSalesOrderODBCDbConnect โดยการรับทอดจาก CODBCDbConnect
2. ออกแบบคุณลักษณะและเมท็อดตามความต้องการทางธุรกิจ
3. การรับทอดเมท็อดของ CODBCDbConnect สำหรับโครงร่างฯ เรียกใช้
4. เมท็อด GetDefaultDSN เป็นเมท็อดซึ่งคืนค่าชื่อของแหล่งข้อมูลซึ่งใช้ในการเชื่อมต่อผ่านมาตรฐานโอดีบีซี ซึ่งในโปรแกรมประยุกต์ระบบการสั่งซื้อสินค้าแบบง่ายใช้ชื่อแหล่งข้อมูลว่า Sales Order

```
0005 std::string CSalesOrderODBCDbConnect::GetDefaultDSN()
0006 {
0007     return "SalesOrder";
0008 }
```

รูปที่ 6.73 ตัวอย่างภาษา C++ ของเมท็อด GetDefaultDSN คลาสเชื่อมต่อระบบการสั่งซื้อสินค้า

6.7. ผลของการใช้งานโครงร่างฯ

จากการใช้งานโครงร่างฯ พบว่าการพัฒนาโปรแกรมประยุกต์จะดำเนินตามขั้นตอนการใช้งานโครงร่างฯ ซึ่งต่างจากการพัฒนาโปรแกรมประยุกต์โดยไม่ใช้โครงร่างฯ ซึ่งมักดำเนินตามวิธีการใช้งานของทางเทคนิคนั้น เช่น เทคนิคการใช้งานฐานข้อมูล เทคนิคการจัดเก็บวัตถุ เป็นต้น ซึ่งทำให้การพัฒนาโปรแกรมประยุกต์ใช้กระบวนการหรือขั้นตอนการพัฒนาที่กำหนดโดยโครงร่างฯ มากกว่าการใช้ประสบการณ์ในการพัฒนาโปรแกรมประยุกต์หรือความชำนาญในการใช้เทคโนโลยีต่างๆ ที่เกี่ยวข้อง ซึ่งโดยมากแล้วการสอนขั้นตอนการใช้งานโครงร่างฯ ย่อมง่ายกว่าการถ่ายทอดประสบการณ์ในการพัฒนาโปรแกรมประยุกต์หรือในการใช้เทคโนโลยีต่างๆ เมื่อพิจารณาส่วนของโปรแกรมจะไม่พบคำสั่งเอสคิวแอล ซึ่งแสดงให้เห็นในภาคผนวก ฉ

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

บทที่ 7

สรุปผลการวิจัย และข้อเสนอแนะ

7.1 สรุปผลการวิจัย

วิทยานิพนธ์นี้ทำการออกแบบพัฒนาโครงร่างประยุกต์เชิงวัตถุสำหรับพัฒนาโปรแกรมประยุกต์ฐานข้อมูลเชิงสัมพันธ์ เพื่อให้สามารถนำโครงร่างฯไปประยุกต์ใช้ในการพัฒนาโครงร่างโปรแกรมประยุกต์เชิงวัตถุเพื่อใช้งานกับฐานข้อมูลเชิงสัมพันธ์ที่เฉพาะเจาะจง เช่น MySQL, Oracle หรือฐานข้อมูลเชิงสัมพันธ์ใดๆ เพื่อให้โปรแกรมเมอร์ได้นำไปใช้งาน โดยทำการพัฒนาโครงร่างฯ ครอบคลุมการปฏิสัมพันธ์ระหว่างโปรแกรมประยุกต์กับฐานข้อมูลเชิงสัมพันธ์ดังนี้

1. การเชื่อมต่อฐานข้อมูลเชิงสัมพันธ์
2. การเพิ่มวัตถุในฐานข้อมูลเชิงสัมพันธ์สำหรับความสัมพันธ์คลาสแบบซิงเกิลคลาส
3. การเพิ่มวัตถุในฐานข้อมูลเชิงสัมพันธ์สำหรับความสัมพันธ์คลาสแบบการรับทอดคลาส
4. การเพิ่มวัตถุในฐานข้อมูลเชิงสัมพันธ์สำหรับความสัมพันธ์คลาสแบบภาพรวมกลุ่มคลาส
5. การเพิ่มวัตถุในฐานข้อมูลเชิงสัมพันธ์สำหรับความสัมพันธ์คลาสแบบคอมโพสิตชันคลาส
6. การปรับปรุงวัตถุในฐานข้อมูลเชิงสัมพันธ์สำหรับความสัมพันธ์คลาสแบบซิงเกิลคลาส
7. การปรับปรุงวัตถุในฐานข้อมูลเชิงสัมพันธ์สำหรับความสัมพันธ์คลาสแบบการรับทอดคลาส
8. การปรับปรุงวัตถุในฐานข้อมูลเชิงสัมพันธ์สำหรับความสัมพันธ์คลาสแบบภาพรวมกลุ่มคลาส
9. การปรับปรุงวัตถุในฐานข้อมูลเชิงสัมพันธ์สำหรับความสัมพันธ์คลาสแบบคอมโพสิตชันคลาส
10. การลบวัตถุในฐานข้อมูลเชิงสัมพันธ์สำหรับความสัมพันธ์คลาสแบบซิงเกิลคลาส
11. การลบวัตถุในฐานข้อมูลเชิงสัมพันธ์สำหรับความสัมพันธ์คลาสแบบการรับทอดคลาส

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

12. การลบวัตถุในฐานข้อมูลเชิงสัมพันธ์สำหรับความสัมพันธ์คลาสแบบภาพรวมกลุ่มคลาส
13. การลบวัตถุในฐานข้อมูลเชิงสัมพันธ์สำหรับความสัมพันธ์คลาสแบบคอมโพสิตชันคลาส
14. การยกเลิกการเชื่อมต่อไปยังฐานข้อมูลเชิงสัมพันธ์

เนื่องด้วยโครงร่างฯ ใช้วิธีการแยกคุณลักษณะของวัตถุเพื่อเก็บลงในตารางความสัมพันธ์ของวัตถุ และทำการรวมคุณลักษณะเมื่อต้องการใช้วัตถุนั้น ซึ่งวิธีการดังกล่าวมีข้อดีคือเป็นวิธีการที่สอดคล้องกับการแปลงแผนภาพคลาสเป็นแผนภาพตารางความสัมพันธ์แบบหนึ่งคลาสต่อหนึ่งตารางความสัมพันธ์ และยังสอดคล้องกับการออกแบบฐานข้อมูลเชิงสัมพันธ์ ซึ่งไม่ควรเก็บข้อมูลเหมือนกันความหมายเดียวในฐานข้อมูลเชิงสัมพันธ์มากกว่าหนึ่งแห่ง ควรใช้ความสัมพันธ์ในการอ้างถึงแทน แต่ด้วยวิธีการนี้มีความยุ่งยากในการออกแบบให้โครงร่างฯ สามารถทำงานตามวิธีการนี้

อีกประการหนึ่งโครงร่างฯ ใช้วิธีการอ้างถึงกันโดยใช้หมายเลขวัตถุเดียวกันในการแสดงถึงหมายเลขวัตถุและแสดงถึงความสัมพันธ์ระหว่างวัตถุ โดยความสัมพันธ์หมายถึงการรับทอดคลาส ภาพรวมกลุ่มคลาส คอมโพสิตชันคลาส ซึ่งมีข้อดีคือเมื่อโครงร่างฯทราบว่าคลาสที่ความสัมพันธ์ด้วยอยู่ในตารางใด โครงร่างฯ สามารถใช้หมายเลขวัตถุนั้นในการอ้างถึงวัตถุที่อยู่ในตารางความสัมพันธ์ของวัตถุที่สัมพันธ์ได้ทันที โดยไม่ต้องมีส่วนที่จัดการหรือจัดเก็บหมายเลขวัตถุสัมพันธ์เพิ่มเติม แต่ด้วยวิธีการนี้ทำให้มีความสับสนในการใช้งาน และทำความเข้าใจเช่นกัน และประเด็นสำคัญไม่สนับสนุนลักษณะการออกแบบแบบการการรับทอดคลาสจากหลายแม่

ผู้ใช้งานโครงร่างฯ มีอยู่ 2 กลุ่ม คือ ผู้ที่นำโครงร่างฯไปสร้างเพื่อใช้กับภาษาเชิงวัตถุและฐานข้อมูลเชิงสัมพันธ์ที่ต้องการ ในกลุ่มผู้ใช้โครงร่างฯ ควรมีความรู้เกี่ยวกับภาษาเชิงวัตถุและฐานข้อมูลเชิงสัมพันธ์นั้นเป็นอย่างดี และควรทำการศึกษาส่วนที่โครงร่างกำหนดให้เขียนโปรแกรมเพื่อให้โครงร่างฯทำงานได้ และอีกกลุ่มหนึ่งคือผู้นำโครงร่างฯไปใช้งาน กลุ่มนี้ควรมีความสามารถเกี่ยวกับภาษาโปรแกรมเชิงวัตถุระดับหนึ่ง และควรทำการศึกษาโครงร่างฯ ที่ถูกสร้างขึ้นแล้วว่าต้องใช้งานอย่างไร และมีข้อกำหนดเพิ่มเติมอย่างไร และควรทำการศึกษาโครงร่างฯว่ามีส่วนที่โครงร่างฯกำหนดให้เขียนโปรแกรม และวิธีการที่โครงร่างฯกำหนด

หลังจากได้นำโครงร่างฯ มาไปพัฒนาให้เป็นจริงโดยเลือกมาตรฐานการเชื่อมต่อไปยังฐานข้อมูลเชิงสัมพันธ์แบบโอดีบีซีกับภาษา C++ และได้นำโครงร่างฯที่สร้างขึ้น ไปใช้ใน

การพัฒนาระบบการสั่งซื้อสินค้าแบบง่ายโดยใช้ภาษา C++ ผ่านมาตรฐานโอดีบีซีซึ่งจัดเก็บวัตถุลงฐานข้อมูลเชิงสัมพันธ์ MySQL เพื่อแสดงว่าโครงร่างฯที่พัฒนาและนำสร้างเพื่อนำไปใช้ในการพัฒนาโปรแกรมยุกต์เชิงวัตถุและจัดเก็บวัตถุในฐานข้อมูลเชิงวัตถุ สามารถนำมาประยุกต์ใช้ได้จริง โดยทำการทดสอบทุกการปฏิสัมพันธ์ระหว่างโปรแกรมประยุกต์เชิงวัตถุกับฐานข้อมูลเชิงสัมพันธ์ พบว่าสามารถจัดเก็บวัตถุ ปรับปรุงวัตถุ ลบวัตถุ และนำวัตถุกลับมาใช้งานได้จริง ไม่ว่าวัตถุจะมีความสัมพันธ์คลาสเป็นแบบซิงเกิลคลาส การรับทอดคลาส ภาพรวมกลุ่มคลาส คอมโพสิตชันคลาส และสามารถเชื่อมต่อและยกเลิกการเชื่อมต่อไปยังฐานข้อมูลเชิงสัมพันธ์ได้จริง และจากการตรวจสอบไม่พบประโยคคำสั่งภาษาเอสคิวแอลในโปรแกรมประยุกต์เลย จากการพิจารณารหัสคำสั่งพบว่าโปรแกรมเมอร์ไม่มีความจำเป็นต้องมีความรู้ ความเข้าใจหรือประสบการณ์ในการออกแบบโปรแกรมและเขียนโปรแกรมเพื่อจัดเก็บวัตถุในฐานข้อมูลเชิงสัมพันธ์โดยภาษาเชิงวัตถุ และความรู้ในการใช้ภาษาเอสคิวแอล ซึ่งแสดงให้เห็นว่าสามารถนำการออกแบบและรหัสคำสั่งกลับมาใช้ใหม่ได้จริง

7.2 ข้อเสนอแนะ

เนื่องจากของจำกัดของโครงร่างฯ ผู้วิจัยเห็นว่าโครงร่างฯอาจได้รับการปรับปรุงเพื่อให้รองรับการออกแบบคลาสซึ่งมีความสัมพันธ์ในลักษณะการการรับทอดจากหลายคลาส (Multi Inheritance) เพื่อให้สอดคล้องกับความสามารถของภาษาเชิงวัตถุบางภาษา

ในการใช้งานโครงร่างฯ มีความจำเป็นต้องมีการเรียนรู้วิธีการใช้งานโครงร่างฯ ผู้วิจัยเห็นว่า อาจพัฒนาเครื่องมือเพื่อทำการสร้างรหัสคำสั่งให้อัตโนมัติ โดยโปรแกรมที่พัฒนาขึ้นอาจให้ผู้ใช้ป้อนข้อมูลเกี่ยวคลาสที่ต้องการให้สร้างรหัสคำสั่งเช่น ชื่อคลาส ชื่อตารางความสัมพันธ์ คุณลักษณะ ประเภทความสัมพันธ์ สัมพันธ์กับคลาสใด อย่างไร

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

รายการอ้างอิง

1. James R Rumbaugh, Michael R. Blaha, William Lorensen, Frederick Eddy, William Premerlani. Object-Oriented Modeling and Design. United States of America : Prentice-Hall, Inc, 1990.

2. Raghu Ramakrishnan. Database Management System. Singapore : McGraw-Hill, 1997.

3. Gary W. Hansen, James V. Hansen. Database Management and Design 2ed. United States of America : Prentice-Hall, Inc, 1996.

4. Scott W. Ambler. Mapping objects to relational databases. Available from : http://www-128.ibm.com/developerworks/webservices/library/ws-mapping-to-rdb/index.html

5. Douglas C. Schmidt. Using Design Patterns to Develop Reusable Object-Oriented Software. Available from : http://siesta.cs.wustl.edu/~schmidt/OOWG-statement.html

6. Glenn E. Krasner, Stephen E. Pope. Leveraging Object-Oriented Technology framework. Available from : http://ieeexplore.ieee.org/iel5/6972/18796/00868976.pdf?arnumber=868976

7. Mohamed E. Fayad, Douglas C. Schmidt, Ralph E. Johnson. Building Applicatoin Framework Object-Oriented Foundations of Framework Design. United States of America : Wiley Computer Publishing, 1999.

8. Sun Microsystems, Inc.. Java Data Objects (JDO). Available from : http://java.sun.com/products/jdo/

9. GemStone 1260 NW Waterhouse Ave. Suite 200 Beaverton, OR 97006 info@gemstone.com. GemStone/S Application Server. Available from : http://www.gemstone.com/products/smalltalk/

10. Objectmatter, 13764 NW 15 ST. Pembroke Pines, FL 33028, UNITED STATES. support@objectmater.com. ObjectMatter. Available from : http://www.objectmatter.com

11. Lewis, T., and others. Object-Oriented Application Framework. n.p. : Manning Publication, 1995.

ภาคผนวก

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

ภาคผนวก ก.

การต้นแบบโครงร่างฯ

ต้นแบบโครงร่างฯ โดยใช้ภาษา C++ แทนภาษาเชิงวัตถุใดๆ ดังนี้

คลาส DbApp

แฟ้มข้อมูล DbApp.h

```
// Static Model


#ifndef __DBAPP__
#define __DBAPP__

#ifdef OOFARDB_EXPORTS
#define OOFARDB_API __declspec(dllexport)
#else
#define OOFARDB_API __declspec(dllimport)
#endif

// Include files
#include "DbConnect.h"
namespace framework
{
        class OOFARDB_API CDbApp
        {

        private:

        public:

                bool LoginDB(std::string user,
                             std::string password);

                void LogoutDB();

                virtual CDbConnect* GetConnect();

        protected:

                void InitialConnector(std::string user,
                                                std::string password);

                bool Connect();

                virtual CDbConnect* NewConnect();

                virtual void DestroyConnect();

        };// END CLASS DEFINITION CDbApp
} // framework


#endif // __DBAPP__
```

แฟ้มข้อมูล DbApp.cpp

```
// Static Model

#include "DbApp.h"
namespace framework
{

        bool CDbApp::LoginDB(std::string user,
                                        std::string password)
        {
                InitialConnector(user, password);
                return Connect();
        }

        void CDbApp::LogoutDB()
        {
                if (GetConnect()) {
                        GetConnect()->Disconnect();
                        //delete GetConnect();
                        DestroyConnect();
                }
        }

        CDbConnect* CDbApp::GetConnect()
        {
                return NULL;
        }

        CDbConnect* CDbApp::NewConnect()
```

```
        {
                return NULL;
        }

        void CDbApp::DestroyConnect()
        {

        }

        void CDbApp::InitialConnector(std::string user,
                                                        std::string password)
        {
                NewConnect();
                GetConnect()->SetUser(user);
                GetConnect()->SetPassword(password);
        }

        bool CDbApp::Connect()
        {
                if (!GetConnect())
                        return false;
                return GetConnect()->Connect();

        }
} // framework
```

คลาส DbConnect

แฟ้มข้อมูล DbConnect.h

```
// Static Model


#ifndef __DBCONNECT__
#define __DBCONNECT__

#ifdef OOFARDB_EXPORTS
#define OOFARDB_API __declspec(dllexport)
#else
#define OOFARDB_API __declspec(dllimport)
#endif

// Include files
#include "Record.h"
#include <string>

namespace framework
{
        class OOFARDB_API CDbConnect
        {

        protected:

                bool m_connect;

        private:

                std::string m_user;

                std::string m_password;

        public:

                void SetUser(std::string user);

                std::string GetUser();

                void SetPassword(std::string password);

                std::string GetPassword();

                bool IsConnect();

                virtual bool ExecuteSQL(std::string cmd);

                virtual bool QuerySQL(std::string cmd,
                                                        std::vector<CRecord*>* pRecSet);

                virtual bool Connect();

                virtual void Disconnect();

        };// END CLASS DEFINITION CDbConnect
} // framework

#endif // __DBCONNECT__
```

แฟ้มข้อมูล DbConnect.cpp

```
// Static Model
```

```
#include "DbConnect.h"
namespace framework
{
        void CDbConnect::SetUser(std::string user)
        {
                m_user = user;
        }

        std::string CDbConnect::GetUser()
        {
                return m_user;
        }

        void CDbConnect::SetPassword(std::string password)
        {
                m_password = password;
        }

        std::string CDbConnect::GetPassword()
        {
                return m_password;
        }

        bool CDbConnect::IsConnect()
        {
                return m_connect;
        }

        bool CDbConnect::ExecuteSQL(std::string cmd)
        {
                return false;
        }

        bool CDbConnect::QuerySQL(std::string cmd,
                                                        std::vector<CRecord*>* pRecSet)
        {
                return false;
        }

        bool CDbConnect::Connect()
        {
                m_connect = false;
                return m_connect;
        }

        void CDbConnect::Disconnect()
        {
                m_connect = false;
        }
} // framework
```

คลาส DbObject

แฟ้มข้อมูล DbObject.h

```
// Static Model


#ifndef __OBJECTFIELD__
#define __OBJECTFIELD__

#ifdef OOFARDB_EXPORTS
#define OOFARDB_API __declspec(dllexport)
#else
#define OOFARDB_API __declspec(dllimport)
#endif

// Include files
#include "DataType.h"
#include <string>

namespace framework
{
        class OOFARDB_API CObjectField
        {

        private:

                std::string m_name;

                framework::DataType m_type;

        public:

                void SetName(std::string name);

                std::string GetName();

                void SetType(DataType type);

                DataType GetType();
```

```
        };// END CLASS DEFINITION CObjectField
} // framework

#endif // __OBJECTFIELD__
```

แฟ้มข้อมูล DbObject.cpp

```
// Static Model

#include "DbObject.h"
#include <time.h>
#include <iostream>

namespace framework
{

        void CDbObject::SetNewOID()
        {
                long oid = 0;
                // calculate new oid
                time_t timer = time(NULL);
                oid = (long) timer;
                m_oid = oid;
        }

        bool CDbObject::SaveRDB()
        {
                std::string sql;
                CObjectField *pField;
                bool bResult = false;
                int f;
                char oid[30];
                sprintf(oid, "%d", m_oid);

                if (m_life == born) {
                        sql = "INSERT INTO " + GetTableName() + " (oid";
                        if (GetFieldCount() > 0)
                                sql += ",";
                        for (f = 1; f <= GetFieldCount(); f++) {
                                pField = GetFieldInfo(f);
                                if (pField != NULL)
                                        sql += pField->GetName();
                                if (f < GetFieldCount())
                                        sql += ",";
                        }
                        sql += ") VALUES (";
                        sql += oid;
                        if (GetFieldCount() > 0)
                                sql += ",";
                        for (f = 1; f <= GetFieldCount(); f++) {
                                pField = GetFieldInfo(f);
                                if (pField != NULL) {
                                        if (pField->GetType() == numeric) // 0 - numeric
                                                sql += GetString(f);
                                        else sql += "'" + GetString(f) + "'";
                                }
                                if (f < GetFieldCount())
                                        sql += ",";
                        }
                        sql += ")";
                        bResult = m_pConnect->ExecuteSQL(sql);
                        if (bResult)
                                m_life = live;
                }
                else if (m_life == live) {
                        // updated
                        sql = "UPDATE " + GetTableName() + " SET ";
                        for (f = 1; f <= GetFieldCount(); f++) {
                                pField = GetFieldInfo(f);
                                if (pField != NULL) {
                                        if (pField->GetType() == numeric)
                                                sql += pField->GetName()+"="+GetString(f);
                                        else sql += pField->GetName()+"='"+GetString(f)+"'";
                                }
                                if (f < GetFieldCount())
                                        sql += ",";
                        }
                        sql += " WHERE OID = ";
                        sql += oid;
                        bResult = m_pConnect->ExecuteSQL(sql);
                }
                else if (m_life == dead) {
                        // delete
                        sql = "DELETE FROM ";
                        sql += GetTableName();
                        sql += " WHERE OID=";
                        sql += oid;
                        bResult = m_pConnect->ExecuteSQL(sql);
                }

                return bResult;
        }

        std::string CDbObject::GenSQL(std::string criteria)
        {
                CObjectField *pField;
                int f;
```

```
	std::string sql;

	sql = "SELECT oid";
	if (GetFieldCount() > 0)
		sql += ", ";
	for (f = 1; f <= GetFieldCount(); f++) {
		pField = GetFieldInfo(f);
		if (pField != NULL)
			sql += pField->GetName();
		if (f < GetFieldCount())
			sql += ",";
	}
	sql += " FROM ";
	sql += GetTableName();
	if (criteria.length() > 0) {
		sql += " WHERE ";
		sql += criteria;
	}
	return sql;
}

int CDbObject::PushAssociate(CDbObject* pObject)
{
	m_pAssociate.push_back(pObject);
	return (int)m_pAssociate.size();
}

CDbObject* CDbObject::PopAssociate()
{
	std::vector <CDbObject*>::iterator iter;

	iter = m_pAssociate.end();
	if (iter != NULL)
		m_pAssociate.erase(iter);

	return *iter;
}

void CDbObject::DestroyAllAssociate()
{
	std::vector <CDbObject*>::iterator iter;

	for (iter = m_pAssociate.begin(); iter < m_pAssociate.end(); iter++)
		delete *iter;

	if (m_pAssociate.size() > 0)
		m_pAssociate.clear();
}

bool CDbObject::SelectRecordToObject(std::string criteria)
{
	bool r = false;
	std::string sql = GenSQL(criteria);
	std::vector<CRecord *> pRecord;

	r = GetConnect()->QuerySQL(sql, &pRecord);

	// transfer record ==> object

	std::vector<CRecord*>::iterator pRecord_iter;

	pRecord_iter = pRecord.begin();
	if (pRecord_iter != NULL)
		RecordToObject(*pRecord_iter, this);

	for (pRecord_iter = pRecord.begin(); pRecord_iter < pRecord.end(); pRecord_iter++)
		delete *pRecord_iter;
	pRecord.clear();

	if (r) {
		r = SelectComponent();
		SelectParent();
	}

	return r;
}

long double CDbObject::SelectRDB(std::string criteria, std::vector<CDbObject*>* pObjectSet)
{
	std::string sql = GenSQL(criteria);
	std::vector<CRecord *> pRecord;

	GetConnect()->QuerySQL(sql, &pRecord);

	// transfer record ==> object

	std::vector<CRecord*>::iterator pRecord_iter;

	for (pRecord_iter = pRecord.begin(); pRecord_iter < pRecord.end(); pRecord_iter++)
		RecordToObject(pObjectSet, *pRecord_iter, this->NewObject());


	// delete record
	for (pRecord_iter = pRecord.begin(); pRecord_iter < pRecord.end(); pRecord_iter++)
		delete *pRecord_iter;
	pRecord.clear();

	return pObjectSet->size();
}
```

```
CDbObject::CDbObject()
{
        SetLifeStatus(framework::born);
}

CDbObject::~CDbObject()
{
        DestroyAllAssociate();
}

void CDbObject::SetConnect(CDbConnect *pConnect)
{
        m_pConnect = pConnect;
}

CDbConnect* CDbObject::GetConnect()
{
        return m_pConnect;
}

void CDbObject::SetOID(long oid)
{
        m_oid = oid;;
}

long CDbObject::GetOID()
{
        return m_oid;
}

void CDbObject::SetLifeStatus(ObjectLife life)
{
        m_life = life;
        if (m_life == born)
                SetNewOID();
}

ObjectLife CDbObject::GetLifeStatus()
{
        return m_life;
}

bool CDbObject::Delete()
{
        bool result = false;
        if (m_life == live) {
                m_life = dead;
                if (Save() == false)
                        m_life = live;
                else result = true;
        }
        return result;

}

bool CDbObject::Save()
{
        bool r = false;

        if (m_life == framework::dead) {
                if (SaveComponent()) {
                        if (SaveRDB())
                                r = SaveParent();
                }
        }
        else {
                if (SaveParent()) {
                        if (SaveRDB())
                                r = SaveComponent();
                }
        }

        return r;
}

bool CDbObject::GetMemberValue()
{
        return GetRecordToObject(this->GetOID());
}

bool CDbObject::SelectParent()
{
        return true;
}

bool CDbObject::SelectComponent()
{
        return true;
}

bool CDbObject::SelectComponentRDB(std::vector<CDbObject*> *pObjectSet)
{
        CDbObject* pObj;
        std::vector<CDbObject*>::const_iterator pObj_iter;

        for (pObj_iter = pObjectSet->begin(); pObj_iter != pObjectSet->end(); pObj_iter++) {
                pObj = *pObj_iter;
                pObj->SelectComponent();
```

```
                }
                return true;
        }

        bool CDbObject::SelectParentRDB(std::vector<CDbObject*> *pObjectSet)
        {
                CDbObject* pObj;
                std::vector<CDbObject*>::const_iterator pObj_iter;

                for (pObj_iter = pObjectSet->begin(); pObj_iter != pObjectSet->end(); pObj_iter++) {
                        pObj = *pObj_iter;
                        pObj->SelectParent();
                }
                return true;
        }

        bool CDbObject::GetRecordToObject(long oid)
        {
                std::string criteria;
                char id[20];

                sprintf(id, "%d", oid);

                criteria = "oid = ";
                criteria += id;

                return SelectRecordToObject(criteria);
        }

        long double CDbObject::Select(std::string criteria, std::vector<CDbObject*>* pObjectSet)
        {
                SelectRDB(criteria, pObjectSet);
                SelectComponentRDB(pObjectSet);
                return SelectParentRDB(pObjectSet);
        }

        void CDbObject::RecordToObject(CRecord *pRecord, CDbObject *pObject)
        {
                CColumn *pCol;
                std::string v;

                pObject->SetLifeStatus(live);
                int c;
                for     (c = 0; c < pRecord->GetCount(); c++) {
                        pCol = pRecord->GetColumn(c);
                        v = pCol->GetValue();
                        pObject->SetObjectValue(c, v);
                }
        }

        void CDbObject::RecordToObject(std::vector<CDbObject*>* pObjectSet, CRecord *pRecord, CDbObject
*pObject)
        {
                CColumn *pCol;
                std::string v;

                pObject->SetLifeStatus(live);
                int c;
                for     (c = 0; c < pRecord->GetCount(); c++) {
                        pCol = pRecord->GetColumn(c);
                        v = pCol->GetValue();
                        pObject->SetObjectValue(c, v);
                }
                pObjectSet->push_back(pObject);
        }

        void CDbObject::CloneMemberValue(CDbObject *other)
        {
                SetConnect(other->GetConnect());
                SetLifeStatus(other->GetLifeStatus());
                SetOID(other->GetOID());
        }

} // framework
```

คลาส Record

แฟ้มข้อมูล Record.h

```
// Static Model


#ifndef __RECORD__
#define __RECORD__

#ifdef OOFARDB_EXPORTS
#define OOFARDB_API __declspec(dllexport)
#else
#define OOFARDB_API __declspec(dllimport)
#endif

// Include files
#include "Column.h"
#include <vector>
```

```
namespace framework
{
	class OOFARDB_API CRecord
	{

	private:

		std::vector<framework::CColumn*> m_pColumn;

	public:

		~CRecord();

		CColumn* GetColumn(int index);

		void AddColumn(CColumn *pColumn);

		long double GetCount();

		void DeleteAllColumn();

	};// END CLASS DEFINITION CRecord
} // framework

#endif // __RECORD__
```

แฟ้มข้อมูล Record.cpp

```
// Static Model

#include "Record.h"
namespace framework
{

	CRecord::~CRecord()
	{
		DeleteAllColumn();
	}

	CColumn* CRecord::GetColumn(int index)
	{
		return m_pColumn[index];
	}

	void CRecord::AddColumn(CColumn *pColumn)
	{
		m_pColumn.push_back(pColumn);
	}

	long double CRecord::GetCount()
	{
		return m_pColumn.size();
	}

	void CRecord::DeleteAllColumn()
	{
		std::vector<framework::CColumn*>::iterator pCol_iter;

		for (pCol_iter = m_pColumn.begin(); pCol_iter < m_pColumn.end();
pCol_iter++)
			delete *pCol_iter;

		m_pColumn.clear();
	}

} // framework
```

แฟ้มข้อมูล Column.h

```
// Static Model


#ifndef __COLUMN__
#define __COLUMN__

#ifdef OOFARDB_EXPORTS
#define OOFARDB_API __declspec(dllexport)
#else
#define OOFARDB_API __declspec(dllimport)
#endif

#include <string>
namespace framework
{
	class OOFARDB_API CColumn
	{
```

```
        private:
                std::string m_value;
        public:
                void SetValue(std::string value);
                std::string GetValue();
        };// END CLASS DEFINITION CColumn
} // framework

#endif // __COLUMN__
```

แฟ้มข้อมูล Column.cpp

```
// Static Model

#include "Column.h"
namespace framework
{
        void CColumn::SetValue(std::string value)
        {
                m_value = value;
        }
        std::string CColumn::GetValue()
        {
                return m_value;
        }
} // framework
```

คลาส ObjectField

แฟ้มข้อมูล ObjectField.h

```
// Static Model


#ifndef __OBJECTFIELD__
#define __OBJECTFIELD__

#ifdef OOFARDB_EXPORTS
#define OOFARDB_API __declspec(dllexport)
#else
#define OOFARDB_API __declspec(dllimport)
#endif

// Include files
#include "DataType.h"
#include <string>

namespace framework
{
        class OOFARDB_API CObjectField
        {
        private:
                std::string m_name;
                framework::DataType m_type;
        public:
                void SetName(std::string name);
                std::string GetName();
                void SetType(DataType type);
                DataType GetType();
        };// END CLASS DEFINITION CObjectField
} // framework


#endif // __OBJECTFIELD__
```

แฟ้มข้อมูล ObjectField.cpp

```
// Static Model

#include "ObjectField.h"
namespace framework
{

	void CObjectField::SetName(std::string name)
	{
		m_name = name;
	}

	std::string CObjectField::GetName()
	{
		return m_name;
	}

	void CObjectField::SetType(DataType type)
	{
		m_type = type;
	}

	DataType CObjectField::GetType()
	{
		return m_type;
	}
} // framework
```

ประเภท DataType

แฟ้มข้อมูล DataType.h

```
// Static Model


#ifndef __DATATYPE__
#define __DATATYPE__

#ifdef OOFARDB_EXPORTS
#define OOFARDB_API __declspec(dllexport)
#else
#define OOFARDB_API __declspec(dllimport)
#endif

namespace framework
{
	enum OOFARDB_API DataType
	{
		numeric = 0,
		character = 1
};
} // framework


#endif // __DATATYPE__
```

แฟ้มข้อมูล ObjectLife.h

```
// Static Model


#ifndef __OBJECTLIFE__
#define __OBJECTLIFE__

#ifdef OOFARDB_EXPORTS
#define OOFARDB_API __declspec(dllexport)
#else
#define OOFARDB_API __declspec(dllimport)
#endif

namespace framework
{
	enum OOFARDB_API ObjectLife
	{
		dead = 0,
		born = 1,
		live = 2
};
} // framework


#endif // __OBJECTLIFE__
```

ภาคผนวก ข.

ตัวอย่างการพัฒนาโครงร่างฯ ด้วยภาษา C++ ผ่านมาตรฐานโอดีบีซี

ตัวอย่างการนำโครงร่างฯไปพัฒนาด้วยภาษา C++ ผ่านมาตรฐานโอดีบีซี มีชุดคำสั่งภาษาดังนี้

คลาส ODBCDbApp

แฟ้มข้อมูล ODBCDbApp.h

```
// Static Model


#ifndef __ODBCDBAPP__
#define __ODBCDBAPP__

#ifdef OOFARDB_EXPORTS
#define OOFARDB_API __declspec(dllexport)
#else
#define OOFARDB_API __declspec(dllimport)
#endif

// Include files
#include "..\framework\DbApp.h"
#include "ODBCDbConnect.h"

namespace odbc
{
        class OOFARDB_API CODBCDbApp : public framework::CDbApp
        {

        private:

        protected:

                virtual CODBCDbConnect* NewConnect();

                virtual void DestroyConnect();

        public:

                virtual CODBCDbConnect* GetConnect();

        };// END CLASS DEFINITION CODBCDbApp
} // odbc


#endif // __ODBCDBAPP__
```

แฟ้มข้อมูล ODBCDbApp.cpp

```
// Static Model

#include "ODBCDbApp.h"
namespace odbc
{

        CODBCDbConnect* CODBCDbApp::NewConnect()
        {
                return NULL;
        }

        CODBCDbConnect* CODBCDbApp::GetConnect()
        {
                return NULL;
        }

        void CODBCDbApp::DestroyConnect()
        {

        }

} // odbc
```

คลาส ODBCDbConnect

แฟ้มข้อมูล ODBCDbConnect.h

```
// Static Model


#ifndef __ODBCDBCONNECT__
#define __ODBCDBCONNECT__

#ifdef OOFARDB_EXPORTS
#define OOFARDB_API __declspec(dllexport)
#else
#define OOFARDB_API __declspec(dllimport)
#endif

// Include files
#include <windows.h>
#include <sql.h>
#include <sqlext.h>

#include "..\framework\DbConnect.h"
#include <vector>
#include <string>

using namespace framework;
namespace odbc
{
        class OOFARDB_API CODBCDbConnect : public framework::CDbConnect
        {

        private:

                std::string m_dsn;

                SQLHENV m_hEnv;

                SQLHDBC m_hDbc;

        public:

                void SetDSN(std::string dsn);

                std::string GetDSN();

                virtual std::string GetDefaultDSN();

                virtual bool Connect(std::string dsn);

                virtual bool Connect();

                virtual void Disconnect();

                virtual bool ExecuteSQL(std::string cmd);

                virtual bool QuerySQL(std::string cmd,
                                                        std::vector<CRecord*>* pRecSet);

        };// END CLASS DEFINITION CODBCDbConnect
} // odbc


#endif // __ODBCDBCONNECT__
```

แฟ้มข้อมูล ODBCDbConnect.cpp

```
// Static Model

#include "ODBCDbConnect.h"
namespace odbc
{

        void CODBCDbConnect::SetDSN(std::string dsn)
        {
                m_dsn = dsn;
        }

        std::string CODBCDbConnect::GetDSN()
        {
                return m_dsn;
        }

        std::string CODBCDbConnect::GetDefaultDSN()
        {
                return ""; // dsn name
        }

        bool CODBCDbConnect::Connect(std::string dsn)
        {
                SQLRETURN retcode;

                SetDSN(dsn);

                m_connect = false;
                retcode = SQLAllocHandle(SQL_HANDLE_ENV, SQL_NULL_HANDLE, &m_hEnv);
                if (retcode == SQL_SUCCESS || retcode == SQL_SUCCESS_WITH_INFO) {
```

```
                        retcode = SQLSetEnvAttr(m_hEnv, SQL_ATTR_ODBC_VERSION, (void*)SQL_OV_ODBC3,
0);
                        if (retcode == SQL_SUCCESS || retcode == SQL_SUCCESS_WITH_INFO) {
                                retcode = SQLAllocHandle(SQL_HANDLE_DBC, m_hEnv, &m_hDbc);
                                if (retcode == SQL_SUCCESS || retcode == SQL_SUCCESS_WITH_INFO) {
                                        SQLSetConnectAttr(m_hDbc, SQL_LOGIN_TIMEOUT, (void*)5, 0);
                                        retcode = SQLConnect(m_hDbc, (SQLCHAR*)GetDSN().data(),
SQL_NTS, (SQLCHAR*)GetUser().data(), SQL_NTS, (SQLCHAR*)GetPassword().data(), SQL_NTS);
                                        if (retcode == SQL_SUCCESS || retcode ==
SQL_SUCCESS_WITH_INFO)
                                                m_connect = true;
                                }
                        }
                }

                return IsConnect();
        }

        bool CODBCDbConnect::Connect()
        {
                return Connect(GetDefaultDSN());
        }

        void CODBCDbConnect::Disconnect()
        {
                if (IsConnect()) {
                        if (m_hDbc) {
                                SQLDisconnect(m_hDbc);
                                SQLFreeHandle(SQL_HANDLE_DBC, m_hDbc);
                        }
                        if (m_hEnv)
                                SQLFreeHandle(SQL_HANDLE_ENV, m_hEnv);
                        m_connect = false;
                }
        }

        bool CODBCDbConnect::ExecuteSQL(std::string cmd)
        {
                SQLHSTMT hstmt;
                SQLRETURN retcode;
                bool bResult = false;

                if (IsConnect()) {
                        retcode = SQLAllocHandle(SQL_HANDLE_STMT, m_hDbc, &hstmt);
                        if (retcode == SQL_SUCCESS || retcode == SQL_SUCCESS_WITH_INFO) {
                                retcode = SQLExecDirect(hstmt, (SQLCHAR*)cmd.data(), SQL_NTS);
                                if (retcode == SQL_SUCCESS || retcode == SQL_NO_DATA)
                                        bResult = true;
                                SQLFreeHandle(SQL_HANDLE_STMT, hstmt);
                        }
                }
                return bResult;
        }

        bool CODBCDbConnect::QuerySQL(std::string cmd,
                                                        std::vector<CRecord*>* pRecSet)
        {
                SQLHSTMT hstmt;
                SQLRETURN retcode;
                bool bResult = false;
                bool bEnd = false;
                SQLCHAR         v[512]="";
                std::string s;
                SQLINTEGER      cb;
                SQLSMALLINT cbColAttr, nField;
                SQLINTEGER      nFieldCount;
                SQLINTEGER      nFieldLength;
                CRecord *pRec;
                CColumn *pCol;

                if (IsConnect()) {
                        retcode = SQLAllocHandle(SQL_HANDLE_STMT, m_hDbc, &hstmt);
                        if (retcode == SQL_SUCCESS || retcode == SQL_SUCCESS_WITH_INFO) {
                                SQLSetStmtAttr(hstmt, SQL_ATTR_CONCURRENCY,
(SQLPOINTER)SQL_CONCUR_READ_ONLY, 0);
                                SQLSetStmtAttr(hstmt, SQL_ATTR_CURSOR_TYPE,
(SQLPOINTER)SQL_CURSOR_STATIC, 0);
                                retcode = SQLExecDirect(hstmt, (SQLCHAR*)cmd.data(), SQL_NTS);
                                if (retcode == SQL_SUCCESS) {
                                        nFieldCount = 0;
                                        retcode = SQLColAttribute(hstmt, 0, SQL_DESC_COUNT, NULL,
sizeof(nFieldCount), &cbColAttr, &nFieldCount);
                                        if (retcode == SQL_SUCCESS || retcode ==
SQL_SUCCESS_WITH_INFO) {
                                                bResult = true;
                                                while (!bEnd) {
                                                        retcode = SQLFetch(hstmt);
                                                        if (retcode == SQL_SUCCESS || retcode ==
SQL_SUCCESS_WITH_INFO) {
                                                                pRec = new CRecord();

                                                                for (nField = 1; nField <=
nFieldCount; nField++) {
                                                                        pCol = new CColumn();
                                                                        retcode =
SQLColAttribute(hstmt, nField, SQL_DESC_OCTET_LENGTH, NULL, sizeof(nFieldLength), &cbColAttr,
&nFieldLength);
                                                                        SQLGetData(hstmt, nField,
SQL_C_CHAR, &v, 512, &cb);
```

```
                                                                        s = (LPCTSTR)v;
                                                                        pCol->SetValue(s);
                                                                        pRec->AddColumn(pCol);
                                                                }
                                                                pRecSet->push_back(pRec);
                                                        }
                                                        else bEnd = true;
                                                }
                                        }
                                }
                                SQLFreeHandle(SQL_HANDLE_STMT, hstmt);
                        }
                }
                return bResult;

        }
} // odbc
```

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

ภาคผนวก ค.

ตัวอย่างการพัฒนาโปรแกรมประยุกต์เชิงวัตถุด้วยโครงร่างฯ

ตัวอย่างการพัฒนาโปรแกรมประยุกต์ด้วยโครงร่างฯ โดยเลือกระบบการสั่งซื้อแบบง่ายซึ่งพิจารณาแล้วพบว่าครอบคลุมความสัมพันธ์ของคลาสและทุกลักษณะการปฏิสัมพันธ์ระหว่างโปรแกรมประยุกต์กับฐานข้อมูลเชิงสัมพันธ์ โดยการพัฒนาผู้วิจัยเลือกภาษา C++ กับฐานข้อมูลเชิงสัมพันธ์ MySQL มีชุดคำสั่งภาษาดังนี้

คลาส Person

แฟ้มข้อมูล Person.h

```
// Static Model

#ifndef __PERSON__
#define __PERSON__

// Include files
#include "..\..\..\oofardb\thesis\framework\ObjectField.h"
#include "..\..\..\oofardb\thesis\framework\DbObject.h"

using namespace framework;

namespace orders
{
        class CPerson : public framework::CDbObject
        {

        private:
                std::string m_name;
                std::string m_address;
                std::string m_phone1;

        public:
                void SetName(std::string name);
                std::string GetName();
                void SetAddress(std::string address);
                std::string GetAddress();
                void SetPhone1(std::string phone1);
                std::string GetPhone1();
                virtual std::string GetString(int index);
                virtual std::string GetString(std::string name);
                virtual void SetObjectValue(int index, std::string v);
                virtual std::string GetTableName();
                virtual CObjectField* GetFieldInfo(int index);
                virtual CObjectField* GetFieldInfo(std::string name);
                virtual int GetFieldCount();
                virtual bool SaveParent();
                virtual bool SaveComponent();
                virtual void CloneMemberValue(CPerson *other);
                virtual CDbObject* NewObject();
                virtual bool SelectParent();
                virtual std::string GetNickName();
```

```
                virtual bool SelectComponent();

        };// END CLASS DEFINITION CPerson
} // orders

#endif // __PERSON__
```

แฟ้มข้อมูล Person.cpp

```
// Static Model

#include "Person.h"
namespace orders
{

        void CPerson::SetName(std::string name)
        {
                m_name = name;
        }

        std::string CPerson::GetName()
        {
                return m_name;
        }

        void CPerson::SetAddress(std::string address)
        {
                m_address = address;
        }

        std::string CPerson::GetAddress()
        {
                return m_address;
        }

        void CPerson::SetPhone1(std::string phone1)
        {
                m_phone1 = phone1;
        }

        std::string CPerson::GetPhone1()
        {
                return m_phone1;
        }

        std::string CPerson::GetString(int index)
        {
                std::string value="";

                switch (index) {
                        case 1:  // name
                                 value = GetName();
                                 break;
                        case 2:  // address
                                 value = GetAddress();
                                 break;
                        case 3:  // phone1
                                 value = GetPhone1();
                                 break;
                }

                return value;
        }

        std::string CPerson::GetString(std::string name)
        {
                int index=-1;

                if (name == "name")
                        index = 1;
                else if (name == "address")
                        index = 2;
                else if (name == "phone1")
                        index = 3;

                return GetString(index);
        }

        std::string CPerson::GetTableName()
        {
                return std::string("Person");
        }

        CObjectField* CPerson::GetFieldInfo(int index)
        {
                CObjectField* pField = NULL;
                if (index >= 1 && index <= GetFieldCount()) {
                        pField = new CObjectField();
                        switch (index) {
                        case 1:
                                 pField->SetName("name");
                                 pField->SetType(framework::character);
                                 break;
                        case 2:
                                 pField->SetName("address");
                                 pField->SetType(framework::character);
```

```
                        break;
                case 3:
                        pField->SetName("phone1");
                        pField->SetType(framework::character);
                        break;
                }
        }

        return pField;
}

CObjectField* CPerson::GetFieldInfo(std::string name)
{
        int fldIndex = -1;

        if (name == std::string("name"))
                fldIndex = 1;
        else if (name == std::string("address"))
                fldIndex = 2;
        else if (name == std::string("phone1"))
                fldIndex = 3;

        return GetFieldInfo(fldIndex);
}

int CPerson::GetFieldCount()
{
        return 3;
}

void CPerson::SetObjectValue(int index, std::string v)
{
        if (index >= 0 && index <= GetFieldCount()) {
                switch (index) {
                        // framework
                case 0:
                        SetOID(atol(v.data()));
                        break;
                        // person
                case 1:
                        SetName(v.data());
                        break;
                case 2:
                        SetAddress(v.data());
                        break;
                case 3:
                        SetPhone1(v.data());
                        break;
                }
        }
}

bool CPerson::SaveParent()
{
        bool r = true;
        return r;
}

bool CPerson::SaveComponent()
{
        bool r = true;
        return r;
}

void CPerson::CloneMemberValue(CPerson *other)
{
        SetName(other->GetName());
        SetAddress(other->GetAddress());
        SetPhone1(other->GetPhone1());

        __super::CloneMemberValue(other);
}

bool CPerson::SelectParent()
{
        bool r = false;

        r = __super::SelectParent();

        return r;
}

bool CPerson::SelectComponent()
{
        bool r = false;

        r = __super::SelectComponent();

        return r;
}

CDbObject* CPerson::NewObject()
{
        CPerson *pObj = new CPerson();
        pObj->SetConnect(GetConnect());
        return static_cast<CDbObject*>(pObj);
}

std::string CPerson::GetNickName()
```

```
        {
                return GetName();
        }
} // orders
```

คลาส Customer

แฟ้มข้อมูล Customer.h

```
// Static Model

#ifndef __CUSTOMER__
#define __CUSTOMER__

// Include files
#include "Person.h"
#include "Sales.h"
namespace orders
{
        class CCustomer : public CPerson
        {
        private:
                double m_credit_limit;
                orders::CSales* m_pSales;
        public:
                CCustomer();
                ~CCustomer();
                void SetCreditLimit(double credit_limit);
                double GetCreditLimit();
                void SetSales(CSales* pSales);
                CSales* GetSales();
                virtual std::string GetString(int index);
                virtual std::string GetString(std::string name);
                virtual void SetObjectValue(int index, std::string v);
                virtual std::string GetTableName();
                virtual CObjectField* GetFieldInfo(int index);
                virtual CObjectField* GetFieldInfo(std::string name);
                virtual int GetFieldCount();
                virtual bool SaveParent();
                virtual bool SaveComponent();
                virtual void CloneMemberValue(CCustomer *other);
                virtual CDbObject* NewObject();
                virtual bool SelectParent();
                virtual std::string GetNickName();
                // get component
                virtual bool SelectComponent();
                CSales* GetSalesOID(long oid);
        };// END CLASS DEFINITION CCustomer
} // orders

#endif // __CUSTOMER__
```

แฟ้มข้อมูล Customer.cpp

```
// Static Model

#include "Customer.h"
```

```
namespace orders
{
	CCustomer::CCustomer()
	{
	}

	CCustomer::~CCustomer()
	{
	}

	void CCustomer::SetCreditLimit(double credit_limit)
	{
		m_credit_limit = credit_limit;
	}

	double CCustomer::GetCreditLimit()
	{
		return m_credit_limit;
	}

	void CCustomer::SetSales(CSales* pSales)
	{
		m_pSales = pSales;
	}

	CSales* CCustomer::GetSales()
	{
		return m_pSales;
	}

	std::string CCustomer::GetString(int index)
	{
		std::string value="";
		char c[20];

		switch (index) {
			case 1:  // credit limit
				sprintf(c, "%.2f", GetCreditLimit());
				value = c;
				break;
			case 2:  // sales
				sprintf(c, "%d", GetSales()->GetOID());
				value = c;
				break;
		}

		return value;
	}

	std::string CCustomer::GetString(std::string name)
	{
		int index=-1;

		if (name == "credit_limit")
			index = 1;
		else if (name == "sales")
			index = 2;

		return GetString(index);
	}

	std::string CCustomer::GetTableName()
	{
		return std::string("Customer");
	}

	CObjectField* CCustomer::GetFieldInfo(int index)
	{
		CObjectField* pField = NULL;
		if (index >= 1 && index <= GetFieldCount()) {
			pField = new CObjectField();
			switch (index) {
			case 1:
				pField->SetName("credit_limit");
				pField->SetType(framework::DataType::numeric);
				break;
			case 2:
				pField->SetName("sales");
				pField->SetType(framework::DataType::numeric);
				break;
			}
		}

		return pField;
	}

	CObjectField* CCustomer::GetFieldInfo(std::string name)
	{
		int fldIndex = -1;

		if (name == std::string("credit_limit"))
			fldIndex = 1;
		else if (name == std::string("sales"))
			fldIndex = 2;

		return GetFieldInfo(fldIndex);
	}

	int CCustomer::GetFieldCount()
```

```
{
        return 2;
}

void CCustomer::SetObjectValue(int index, std::string v)
{
        if (index >= 0 && index <= GetFieldCount()) {
                switch (index) {
                        // framework
                case 0:
                        SetOID(atol(v.data()));
                        break;
                case 1:
                        SetCreditLimit(atof(v.data()));
                        break;
                case 2:
                        SetSales(GetSalesOID(atol(v.data())));
                        break;
                }
        }
}

bool CCustomer::SaveParent()
{
        bool r = false;

        // parent
        CPerson *pObj = new CPerson();
        pObj->CloneMemberValue((CPerson*)this);
        r = pObj->Save();
        delete pObj;

        return r;
}

bool CCustomer::SaveComponent()
{
        bool r = true;
        return r;
}

void CCustomer::CloneMemberValue(CCustomer *other)
{
        SetCreditLimit(other->GetCreditLimit());
        SetSales(other->GetSales());

        __super::CloneMemberValue(other);
}

bool CCustomer::SelectParent()
{
        bool r = false;

        // parent
        CPerson *pObj = new CPerson();
        pObj->CloneMemberValue((CPerson*)this);
        pObj->GetMemberValue();

        // customer
        CPerson *pCusPerson = (CPerson*)this;
        pCusPerson->CloneMemberValue(pObj);

        delete pObj;

        __super::SelectParent();

        return r;
}

CDbObject* CCustomer::NewObject()
{
        CCustomer *pObj = new CCustomer();
        pObj->SetConnect(GetConnect());
        return static_cast<CDbObject*>(pObj);
}

CSales* CCustomer::GetSalesOID(long oid)
{
        CSales *sales = new CSales();

        sales->SetConnect(GetConnect());
        sales->SetOID(oid);
        if (sales->GetMemberValue())
                PushAssociate(sales); // framework

        return sales;
}

bool CCustomer::SelectComponent()
{
        bool r = false;

        r = __super::SelectComponent();

        return r;
}

std::string CCustomer::GetNickName()
{
```

```
                return GetName();
        }
} // orders
```

คลาส Employee

แฟ้มข้อมูล Employee.h

```
// Static Model

#ifndef __EMPLOYEE__
#define __EMPLOYEE__

// Include files
#include "Person.h"
namespace orders
{
        class CEmployee : public CPerson
        {
        private:
                double m_salary;
        public:
                void SetSalary(double salary);
                double GetSalary();
                // framework
                virtual std::string GetString(int index);
                virtual std::string GetString(std::string name);
                virtual std::string GetTableName();
                virtual CObjectField* GetFieldInfo(int index);
                virtual CObjectField* GetFieldInfo(std::string name);
                virtual int GetFieldCount();
                virtual void SetObjectValue(int index, std::string v);
                virtual bool SaveParent();
                virtual bool SaveComponent();
                virtual void CloneMemberValue(CEmployee *other);
                virtual CDbObject* NewObject();
                virtual bool SelectParent();
                virtual bool SelectComponent();
                virtual std::string GetNickName();
        };// END CLASS DEFINITION CEmployee
} // orders

#endif // __EMPLOYEE__
```

แฟ้มข้อมูล Employee.cpp

```
// Static Model

#include "Employee.h"
namespace orders
{
        void CEmployee::SetSalary(double salary)
        {
                m_salary = salary;
        }
        double CEmployee::GetSalary()
        {
                return m_salary;
        }
        std::string CEmployee::GetString(int index)
        {
                std::string value="";
```

```
        switch (index) {
                case 1:  // salary
                        char c[20];
                        sprintf(c, "%.2f", GetSalary());
                        value = c;
                        break;
        }

        return value;
}

std::string CEmployee::GetString(std::string name)
{
        int index=-1;

        if (name == "salary")
                index = 1;

        return GetString(index);
}

std::string CEmployee::GetTableName()
{
        return std::string("Employee");
}

CObjectField* CEmployee::GetFieldInfo(int index)
{
        CObjectField* pField = NULL;
        if (index >= 1 && index <= GetFieldCount()) {
                pField = new CObjectField();
                switch (index) {
                case 1:
                        pField->SetName("salary");
                        pField->SetType(framework::DataType::numeric);
                        break;
                }
        }

        return pField;
}

CObjectField* CEmployee::GetFieldInfo(std::string name)
{
        int fldIndex = -1;

        if (name == std::string("salary"))
                fldIndex = 1;

        return GetFieldInfo(fldIndex);
}

int CEmployee::GetFieldCount()
{
        return 1;
}

void CEmployee::SetObjectValue(int index, std::string v)
{
        if (index >= 0 && index <= GetFieldCount()) {
                switch (index) {
                        // framework
                case 0:
                        SetOID(atol(v.data()));
                        break;
                        // person
                case 1:
                        SetSalary(atof(v.data()));
                        break;
                }
        }
}

bool CEmployee::SaveParent()
{
        bool r = false;

        // parent
        CPerson *pObj = new CPerson();
        pObj->CloneMemberValue((CPerson*)this);
        r = pObj->Save();
        delete pObj;

        return r;
}

bool CEmployee::SaveComponent()
{
        bool r = true;

        return r;
}

void CEmployee::CloneMemberValue(CEmployee *other)
{
        SetSalary(other->GetSalary());

        __super::CloneMemberValue(other);
```

```
        }

        bool CEmployee::SelectParent()
        {
                bool r = false;

                // parent
                CPerson *pObj = new CPerson();
                pObj->CloneMemberValue((CPerson*)this);
                pObj->GetMemberValue();

                ((CPerson*)this)->CloneMemberValue(pObj);

                delete pObj;

                __super::SelectParent();

                return r;
        }

        bool CEmployee::SelectComponent()
        {
                bool r = false;

                r = __super::SelectComponent();

                return r;
        }

        CDbObject* CEmployee::NewObject()
        {
                CEmployee *pObj = new CEmployee();
                pObj->SetConnect(GetConnect());
                return static_cast<CDbObject*>(pObj);
        }

        std::string CEmployee::GetNickName()
        {
                return GetName();

        }

} // orders
```

คลาส Sales

แฟ้มข้อมูล Sales.h

```
// Static Model


#ifndef __SALES__
#define __SALES__


// Include files
#include "Employee.h"
namespace orders
{
        class CSales : public CEmployee
        {

        private:

                float m_commission_rate;

        public:

                void SetCommissionRate(float commission_rate);

                float GetCommissionRate();

                // framework

                virtual std::string GetString(int index);

                virtual std::string GetString(std::string name);

                virtual void SetObjectValue(int index, std::string v);

                virtual std::string GetTableName();

                virtual CObjectField* GetFieldInfo(int index);

                virtual CObjectField* GetFieldInfo(std::string name);

                virtual int GetFieldCount();

                virtual bool SaveParent();

                virtual bool SaveComponent();

                virtual void CloneMemberValue(CSales *other);
```

```
                virtual CDbObject* NewObject();

                virtual bool SelectParent();

                virtual std::string GetNickName();

                virtual bool SelectComponent();

        };// END CLASS DEFINITION CSales
} // orders

#endif // __SALES__
```

แฟ้มข้อมูล Sales.cpp

```
// Static Model

#include "Sales.h"
namespace orders
{

        void CSales::SetCommissionRate(float commission_rate)
        {
                m_commission_rate = commission_rate;
        }

        float CSales::GetCommissionRate()
        {
                return m_commission_rate;
        }

        std::string CSales::GetString(int index)
        {
                std::string value="";

                switch (index) {
                        case 1:  // commission rate
                                char c[20];
                                sprintf(c, "%.2f", GetCommissionRate());
                                value = c;
                                break;
                }

                return value;
        }

        std::string CSales::GetString(std::string name)
        {
                int index=-1;

                if (name == "commission_rate")
                        index = 1;

                return GetString(index);
        }

        std::string CSales::GetTableName()
        {
                return std::string("Sales");
        }

        CObjectField* CSales::GetFieldInfo(int index)
        {
                CObjectField* pField = NULL;
                if (index >= 1 && index <= GetFieldCount()) {
                        pField = new CObjectField();
                        switch (index) {
                        case 1:
                                pField->SetName("commission_rate");
                                pField->SetType(framework::DataType::numeric);
                                break;
                        }
                }

                return pField;
        }

        CObjectField* CSales::GetFieldInfo(std::string name)
        {
                int fldIndex = -1;

                if (name == std::string("commission_rate"))
                        fldIndex = 1;

                return GetFieldInfo(fldIndex);
        }

        int CSales::GetFieldCount()
        {
                return 1;
        }

        void CSales::SetObjectValue(int index, std::string v)
        {
                if (index >= 0 && index <= GetFieldCount()) {
                        switch (index) {
```

```
                        // framework
                case 0:
                        SetOID(atol(v.data()));
                        break;
                case 1:
                        SetCommissionRate((float)atof(v.data()));
                        break;
                }
        }
}

bool CSales::SaveParent()
{
        bool r = true;

        // parent
        CEmployee *pObj = new CEmployee();
        pObj->CloneMemberValue((CEmployee*)this);
        r = pObj->Save();
        delete pObj;

        return r;
}

bool CSales::SaveComponent()
{
        bool r = true;

        return r;
}

void CSales::CloneMemberValue(CSales *other)
{
        SetCommissionRate(other->GetCommissionRate());

        __super::CloneMemberValue(other);
}

bool CSales::SelectParent()
{
        bool r = false;

        // parent
        CEmployee *pObj = new CEmployee();
        pObj->CloneMemberValue((CEmployee*)this);
        pObj->GetMemberValue();

        ((CEmployee*)this)->CloneMemberValue(pObj);

        delete pObj;

        r = __super::SelectParent();

        return r;
}

bool CSales::SelectComponent()
{
        bool r = false;

        r = __super::SelectComponent();

        return r;
}

CDbObject* CSales::NewObject()
{
        CSales *pObj = new CSales();
        pObj->SetConnect(GetConnect());
        return static_cast<CDbObject*>(pObj);
}

std::string CSales::GetNickName()
{
        return GetName();

}
} // orders
```

คลาส Product

แฟ้มข้อมูล Product.h

```
// Static Model

#ifndef __PRODUCT__
#define __PRODUCT__

// Include files
```

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

```
#include "..\..\..\oofardb\thesis\framework\DbObject.h"
#include "..\..\..\oofardb\thesis\framework\ObjectField.h"

using namespace framework;

namespace orders
{
        class CProduct : public framework::CDbObject
        {

        private:

                std::string m_code_no;

                std::string m_name;

                float m_price;

        public:

                void SetCodeNo(std::string code_no);

                std::string GetCodeNo();

                void SetName(std::string name);

                std::string GetName();

                void SetPrice(float price);

                float GetPrice();

                virtual bool SaveParent();

                virtual bool SaveComponent();

                virtual std::string GetString(int index);

                virtual std::string GetString(std::string name);

                virtual std::string GetTableName();

                virtual CObjectField* GetFieldInfo(int index);

                virtual CObjectField* GetFieldInfo(std::string name);

                virtual int GetFieldCount();

                virtual void SetObjectValue(int index, std::string v);

                virtual void CloneMemberValue(CProduct *other);

                //virtual CProduct* NewObject();

                virtual CDbObject* NewObject();

                virtual bool SelectParent();

                virtual std::string GetNickName();

                virtual bool SelectComponent();


        };// END CLASS DEFINITION CProduct
} // orders


#endif // __PRODUCT__
```

แฟ้มข้อมูล Product.cpp

```
// Static Model

#include "Product.h"
namespace orders
{

        void CProduct::SetCodeNo(std::string code_no)
        {
                m_code_no = code_no;
        }

        std::string CProduct::GetCodeNo()
        {
                return m_code_no;
        }

        void CProduct::SetName(std::string name)
        {
                m_name = name;
        }

        std::string CProduct::GetName()
        {
                return m_name;
        }

        void CProduct::SetPrice(float price)
```

```
{
        m_price = price;
}

float CProduct::GetPrice()
{
        return m_price;
}

std::string CProduct::GetString(int index)
{
        std::string value="";

        switch (index) {
                case 1:  // code_no
                        value = GetCodeNo();
                        break;
                case 2: // name
                        value = GetName();
                        break;
                case 3: // price
                        char c[20];
                        sprintf(c, "%.2f", GetPrice());
                        value = c;
                        break;
        }

        return value;
}

std::string CProduct::GetString(std::string name)
{
        int index=-1;

        if (name == "code_no")
                index = 0;
        else if (name == "name")
                index = 1;
        else if (name == "price")
                index = 2;

        return GetString(index);
}

std::string CProduct::GetTableName()
{
        return std::string("Product");
}

CObjectField* CProduct::GetFieldInfo(int index)
{
        CObjectField* pField = NULL;
        if (index >= 1 && index <= GetFieldCount()) {
                pField = new CObjectField();
                switch (index) {
                case 1:
                        pField->SetName("code_no");
                        pField->SetType(framework::character);
                        break;
                case 2:
                        pField->SetName("name");
                        pField->SetType(framework::character);
                        break;
                case 3:
                        pField->SetName("price");
                        pField->SetType(framework::numeric);
                        break;
                }
        }

        return pField;
}

CObjectField* CProduct::GetFieldInfo(std::string name)
{
        int fldIndex = -1;

        if (name == std::string("code_no"))
                fldIndex = 1;
        else if (name == std::string("name"))
                fldIndex = 2;
        else if (name == std::string("price"))
                fldIndex = 3;

        return GetFieldInfo(fldIndex);
}

int CProduct::GetFieldCount()
{
        return 3;
}

void CProduct::SetObjectValue(int index, std::string v)
{
        if (index >= 0 && index <= GetFieldCount()) {
                switch (index) {
                case 0:
                        SetOID(atol(v.data()));
                        break;
```

```
                        case 1:
                                SetCodeNo(v);
                                break;
                        case 2:
                                SetName(v);
                                break;
                        case 3:
                                SetPrice((float)atof(v.data()));
                                break;
                        }
                }
        }

        bool CProduct::SaveParent()
        {
                bool r = true;

                return r;
        }

        bool CProduct::SaveComponent()
        {
                bool r = true;

                return r;
        }

        void CProduct::CloneMemberValue(CProduct* other)
        {
                SetCodeNo(other->GetCodeNo());
                SetName(other->GetName());
                SetPrice(other->GetPrice());

                __super::CloneMemberValue(other);
        }

        bool CProduct::SelectParent()
        {
                bool r = false;

                r = __super::SelectParent();

                return r;
        }

        bool CProduct::SelectComponent()
        {
                bool r = false;

                r = __super::SelectComponent();

                return r;
        }

        CDbObject* CProduct::NewObject()
        {
                CProduct *pObj = new CProduct();
                pObj->SetConnect(GetConnect());
                return static_cast<CDbObject*>(pObj);
        }

        std::string CProduct::GetNickName()
        {
                return GetName();
        }

} // orders
```

คลาส Orders

เพิ่มข้อมูล Orders.h

```
// Static Model


#ifndef __ORDERS__
#define __ORDERS__


// Include files
#include "..\..\..\oofardb\thesis\framework\DbObject.h"
#include "Customer.h"
#include "OrderItem.h"
#include <vector>
#include <string>

namespace orders
{
        class COrderItem;
        class COrders : public framework::CDbObject
        {

        private:
```

```
                std::string m_order_no;
                unsigned long m_date;
                std::vector<CDbObject*> m_pOrderItems;
                orders::CCustomer* m_pCustomer;
        public:
                COrders();
                ~COrders();
                void SetOrderNo(std::string order_no);
                std::string GetOrderNo();
                void SetDate(unsigned long date);
                unsigned long GetDate();
                void SetCustomer(CCustomer *pCustomer);
                CCustomer *GetCustomer();
                void SetOrderItems(std::vector<CDbObject*>* pOrderItem);
                std::vector<CDbObject*>* GetOrderItems();
                void DeleteAllOrderItems();
                int AddOrderItem(COrderItem *pItem);
                void RemoveOrderItem(int n);
                void UpdateOrderItem(int n, COrderItem *pItem);
                virtual std::string GetString(int index);
                virtual std::string GetString(std::string name);
                virtual std::string GetTableName();
                virtual CObjectField* GetFieldInfo(int index);
                virtual CObjectField* GetFieldInfo(std::string name);
                virtual int GetFieldCount();
                virtual CDbObject* NewObject();
                virtual void SetObjectValue(int index, std::string v);
                virtual bool SelectParent();
                virtual bool SaveParent();
                virtual bool SaveComponent();
                virtual void CloneMemberValue(COrders* other);
                virtual std::string GetNickName();
                // get component
                virtual bool SelectComponent();
                CCustomer* GetCustomerOID(long oid);
        };// END CLASS DEFINITION COrders
} // orders

#endif // __ORDERS__
```

แฟ้มข้อมูล Orders.cpp

```
// Static Model

#include "Orders.h"
namespace orders
{
        COrders::COrders()
        {
        }

        COrders::~COrders()
        {
                DeleteAllOrderItems();
        }

        void COrders::SetOrderNo(std::string order_no)
        {
                m_order_no = order_no;
        }

        std::string COrders::GetOrderNo()
```

```
{
        return m_order_no;
}

void COrders::SetDate(unsigned long date)
{
        m_date = date;
}

unsigned long COrders::GetDate()
{
        return m_date;
}

void COrders::SetCustomer(CCustomer *pCustomer)
{
        m_pCustomer = pCustomer;
}

CCustomer *COrders::GetCustomer()
{
        return m_pCustomer;
}

void COrders::SetOrderItems(std::vector<CDbObject*>* pOrderItem)
{
        DeleteAllOrderItems();

        std::vector <CDbObject*>::iterator iter;

        for (iter = pOrderItem->begin(); iter < pOrderItem->end(); iter++)
                m_pOrderItems.push_back(*iter);
}

std::vector<CDbObject*>* COrders::GetOrderItems()
{
        return &m_pOrderItems;
}

void COrders::DeleteAllOrderItems()
{
        std::vector <CDbObject*>::iterator iter;

        for (iter = m_pOrderItems.begin(); iter < m_pOrderItems.end(); iter++)
                delete *iter;

        m_pOrderItems.clear();
}

int COrders::AddOrderItem(COrderItem *pItem)
{
        COrderItem *newItem = new COrderItem();

        newItem->SetConnect(GetConnect());

        newItem->CloneMemberValue(pItem);
        m_pOrderItems.push_back(newItem);

        return (int)m_pOrderItems.size();
}

void COrders::RemoveOrderItem(int n)
{
        CDbObject* item;

        if (n > 0 && n < (int)m_pOrderItems.size()) {
                item = m_pOrderItems.at(n);
                delete item;
                m_pOrderItems.erase(m_pOrderItems.begin() + n);
        }
}

void COrders::UpdateOrderItem(int n, COrderItem *pItem)
{
        if (n > 0 && n < (int)m_pOrderItems.size()) {
                COrderItem *newItem = new COrderItem();

                newItem->SetConnect(GetConnect());
                newItem->CloneMemberValue(pItem);

                m_pOrderItems[n] = newItem;
        }
}

std::string COrders::GetString(int index)
{
        std::string value="";
        char c[20];

        switch (index) {
                case 1:  // order no
                         value = GetOrderNo();
                         break;
                case 2:  // order date
                         sprintf(c, "%d", GetDate());
                         value = c;
                         break;
                case 3:  // customer
                         sprintf(c, "%d", GetCustomer()->GetOID());
```

```
                        value = c;
                        break;
        }

        return value;
}

std::string COrders::GetString(std::string name)
{
        int index=-1;

        if (name == "order_no")
                index = 1;
        else if (name == "order_date")
                index = 2;
        else if (name == "customer")
                index = 3;

        return GetString(index);
}

std::string COrders::GetTableName()
{
        return std::string("Orders");
}

CObjectField* COrders::GetFieldInfo(int index)
{
        CObjectField* pField = NULL;
        if (index >= 1 && index <= GetFieldCount()) {
                pField = new CObjectField();
                switch (index) {
                case 1:
                        pField->SetName("order_no");
                        pField->SetType(framework::character);
                        break;
                case 2:
                        pField->SetName("order_date");
                        pField->SetType(framework::numeric);
                        break;
                case 3:
                        pField->SetName("customer");
                        pField->SetType(framework::numeric);
                        break;
                }
        }

        return pField;
}

CObjectField* COrders::GetFieldInfo(std::string name)
{
        int fldIndex = -1;

        if (name == std::string("order_no"))
                fldIndex = 1;
        else if (name == std::string("order_date"))
                fldIndex = 2;
        else if (name == std::string("customer"))
                fldIndex = 3;

        return GetFieldInfo(fldIndex);
}

int COrders::GetFieldCount()
{
        return 3;
}

void COrders::SetObjectValue(int index, std::string v)
{
        if (index >= 0 && index <= GetFieldCount()) {
                switch (index) {
                case 0:
                        SetOID(atol(v.data()));
                        break;
                case 1:
                        SetOrderNo(v);
                        break;
                case 2:
                        SetDate(atol(v.data()));
                        break;
                case 3:
                        SetCustomer(GetCustomerOID(atol(v.data())));
                        break;
                }
        }
}

bool COrders::SaveParent()
{
        bool r = true;

        return r;
}

bool COrders::SaveComponent()
{
        bool r = true;
```

```
                std::vector <CDbObject*>::iterator iter;
                COrderItem *item;

                for (iter = m_pOrderItems.begin(); iter < m_pOrderItems.end(); iter++)  {
                        item = static_cast<COrderItem*>(*iter);
                        //item->SetLifeStatus(GetLifeStatus());
                        item->Save();
                }

                return r;
        }

        bool COrders::SelectComponent()
        {
                bool r = false;

                COrderItem obj;
                char c[256];

                DeleteAllOrderItems();

                sprintf(c, "order_no = %s", GetOrderNo().data());
                obj.SetConnect(GetConnect());
                obj.Select(c, &m_pOrderItems);

                r = __super::SelectComponent();

                return r;
        }

        void COrders::CloneMemberValue(COrders *other)
        {
                SetOrderNo(other->GetOrderNo());
                SetDate(other->GetDate());
                SetCustomer(other->GetCustomer());
                SetOrderItems(other->GetOrderItems());

                __super::CloneMemberValue(other);
        }

        std::string COrders::GetNickName()
        {
                return GetOrderNo();

        }

        CDbObject* COrders::NewObject()
        {
                COrders *pObj = new COrders();
                pObj->SetConnect(GetConnect());
                return static_cast<CDbObject*>(pObj);
        }

        bool COrders::SelectParent()
        {
                bool r = false;

                r = __super::SelectParent();

                return r;
        }

        CCustomer* COrders::GetCustomerOID(long oid)
        {
                CCustomer *customer = new CCustomer();

                customer->SetConnect(GetConnect());
                customer->SetOID(oid);
                if (customer->GetMemberValue())
                        PushAssociate(customer); // framework

                return customer;
        }
} // orders
```

คลาส OrdersItem

แฟ้มข้อมูล OrdersItem.h

```
// Static Model


#ifndef __ORDERITEM__
#define __ORDERITEM__


// Include files
#include "..\..\..\oofardb\thesis\framework\DbObject.h"
#include "Product.h"
```

```
namespace orders
{
	class COrderItem : public framework::CDbObject
	{
	private:
		std::string m_order_no;
		float m_qty;
		double m_price;
		orders::CProduct *m_pProduct;
	public:
		void SetOrderNo(std::string orderno);
		std::string GetOrderNo();
		void SetProduct(CProduct* pProduct);
		CProduct* GetProduct();
		void SetQty(float qty);
		float GetQty();
		void SetPrice(double price);
		double GetPrice();
		virtual bool SaveParent();
		virtual bool SaveComponent();
		virtual std::string GetString(int index);
		virtual std::string GetString(std::string name);
		virtual std::string GetTableName();
		virtual CObjectField* GetFieldInfo(int index);
		virtual CObjectField* GetFieldInfo(std::string name);
		virtual int GetFieldCount();
		virtual void SetObjectValue(int index, std::string v);
		virtual void CloneMemberValue(COrderItem* other);
		virtual CDbObject* NewObject();
		virtual bool SelectParent();
		virtual std::string GetNickName();
		// get component
		virtual bool SelectComponent();
		CProduct* GetProductOID(long oid);
	};// END CLASS DEFINITION COrderItem
} // orders

#endif // __ORDERITEM__
```

แฟ้มข้อมูล OrdersItem.cpp

```
// Static Model

#include "OrderItem.h"
namespace orders
{
	void COrderItem::SetOrderNo(std::string orderno)
	{
		m_order_no = orderno;
	}
	std::string COrderItem::GetOrderNo()
	{
		return m_order_no;
	}
	void COrderItem::SetProduct(CProduct* pProduct)
	{
		if (pProduct != NULL)
			m_pProduct = this->GetProductOID(pProduct->GetOID());
		else m_pProduct = NULL;
	}
	CProduct* COrderItem::GetProduct()
```

```
{
        return m_pProduct;
}

void COrderItem::SetQty(float qty)
{
        m_qty = qty;
}

float COrderItem::GetQty()
{
        return m_qty;
}

void COrderItem::SetPrice(double price)
{
        m_price = price;
}

double COrderItem::GetPrice()
{
        return m_price;
}

std::string COrderItem::GetString(int index)
{
        std::string value="";
        char c[20];

        switch (index) {
                case 1:  // order
                        value = GetOrderNo();
                        break;
                case 2:  // product
                        sprintf(c, "%d", GetProduct()->GetOID());
                        value = c;
                        break;
                case 3:  // qty
                        sprintf(c, "%.2f", GetQty());
                        value = c;
                        break;
                case 4:  // price
                        sprintf(c, "%.2f", GetPrice());
                        value = c;
                        break;
        }

        return value;
}

std::string COrderItem::GetString(std::string name)
{
        int index=-1;

        if (name == "order_no")
                index = 1;
        else if (name == "product")
                index = 2;
        else if (name == "qty")
                index = 3;
        else if (name == "price")
                index = 4;

        return GetString(index);
}

std::string COrderItem::GetTableName()
{
        return std::string("OrderItem");
}

CObjectField* COrderItem::GetFieldInfo(int index)
{
        CObjectField* pField = NULL;
        if (index >= 1 && index <= GetFieldCount()) {
                pField = new CObjectField();
                switch (index) {
                case 1:
                        pField->SetName("order_no");
                        pField->SetType(framework::character);
                        break;
                case 2:
                        pField->SetName("product");
                        pField->SetType(framework::numeric);
                        break;
                case 3:
                        pField->SetName("qty");
                        pField->SetType(framework::numeric);
                        break;
                case 4:
                        pField->SetName("price");
                        pField->SetType(framework::numeric);
                        break;
                }
        }

        return pField;
}
```

```
CObjectField* COrderItem::GetFieldInfo(std::string name)
{
        int fldIndex = -1;

        if (name == std::string("order_no"))
                fldIndex = 1;
        else if (name == std::string("product"))
                fldIndex = 2;
        else if (name == std::string("qty"))
                fldIndex = 3;
        else if (name == std::string("price"))
                fldIndex = 4;

        return GetFieldInfo(fldIndex);
}

int COrderItem::GetFieldCount()
{
        return 4;
}

void COrderItem::SetObjectValue(int index, std::string v)
{
        if (index >= 0 && index <= GetFieldCount()) {
                switch (index) {
                case 0:
                        SetOID(atol(v.data()));
                        break;
                case 1:
                        SetOrderNo(v.data());
                        break;
                case 2:
                        SetProduct(GetProductOID(atol(v.data())));
                        break;
                case 3:
                        SetQty((float)atof(v.data()));
                        break;
                case 4:
                        SetPrice(atof(v.data()));
                        break;
                }
        }
}

bool COrderItem::SaveParent()
{
        bool r = true;

        return r;
}

bool COrderItem::SaveComponent()
{
        bool r = true;

        return r;
}

void COrderItem::CloneMemberValue(COrderItem *other)
{
        SetOrderNo(other->GetOrderNo());
        SetProduct(other->GetProduct());
        SetQty(other->GetQty());
        SetPrice(other->GetPrice());

        __super::CloneMemberValue(other);
}

bool COrderItem::SelectParent()
{
        bool r = false;

        r = __super::SelectParent();

        return r;
}


CDbObject* COrderItem::NewObject()
{
        COrderItem *pObj = new COrderItem();
        pObj->SetConnect(GetConnect());
        return static_cast<CDbObject*>(pObj);
}

bool COrderItem::SelectComponent()
{
        bool r = false;

        r = __super::SelectComponent();

        return r;
}

CProduct* COrderItem::GetProductOID(long oid)
{
        CProduct* product = new CProduct();

        product->SetConnect(GetConnect());
```

```
                product->SetOID(oid);
                if (product->GetMemberValue())
                        PushAssociate(product);

                return product;
        }

        std::string COrderItem::GetNickName()
        {
                return std::string("");

        }

} // orders
```

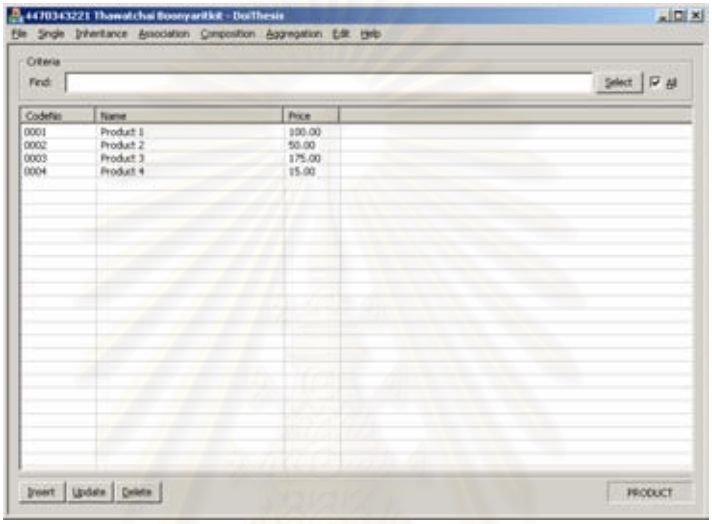

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

ภาคผนวก ง.

ตัวอย่างหน้าจอระบบการสั่งซื้อแบบง่าย

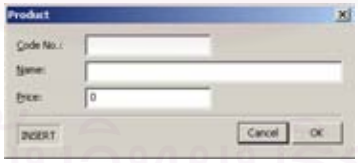
ซิงเกิลคลาส

รูปที่ ง-1 แสดงหน้าจอวัตถุสินค้าซึ่งเป็นตัวแทนของคลาสความสัมพันธ์แบบซิงเกิลคลาส จากรูปแสดงให้เห็นว่าผู้ใช้สามารถเลือกเพิ่ม แก้ไข ลบวัตถุสินค้าได้



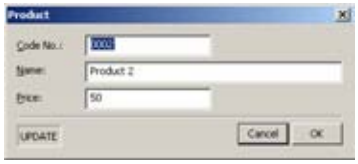
รูปที่ ง-1 หน้าจอแสดงวัตถุสินค้าทั้งหมด

รูปที่ ง-2 แสดงหน้าการเพิ่มวัตถุสินค้า ซึ่งเป็นหน้าจอจากการเลือกเพิ่มวัตถุสินค้าในหน้าจอแสดงวัตถุสินค้าทั้งหมด



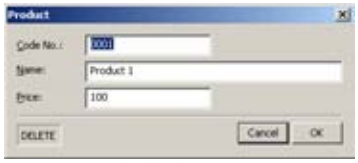
รูปที่ ง-2 หน้าจอแสดงการเพิ่มวัตถุสินค้า

รูปที่ ง-3 แสดงหน้าจอการปรับปรุงวัตถุสินค้ารหัสสินค้า 0002 หน้าจอดังกล่าวเป็นหน้าจอจากการเลือกปรับปรุงวัตถุสินค้าในหน้าจอแสดงวัตถุสินค้าทั้งหมด



รูปที่ ง-3 หน้าจอแสดงการปรับปรุงวัตถุสินค้า

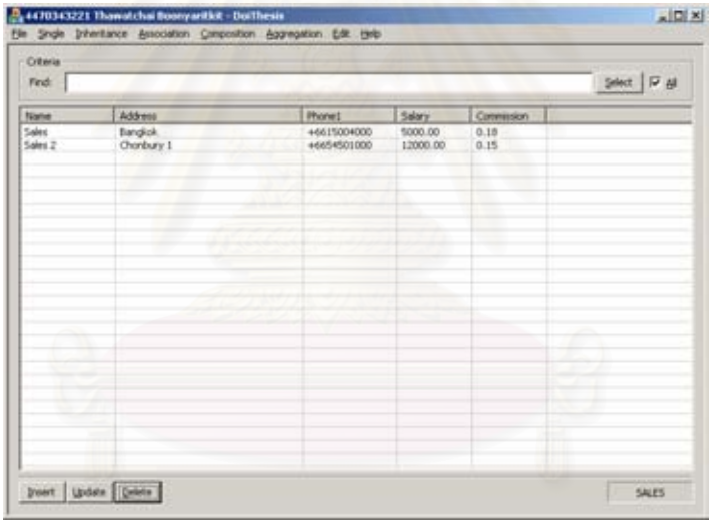
รูปที่ ง-4 แสดงหน้าจอการลบวัตถุสินค้ารหัสสินค้า 0001 ซึ่งหน้าจอจะเหมือนกับหน้าจอการปรับปรุงแต่ต่างกันที่การกระทำหลังเลือกคำสั่งตกลงจะทำการลบแทนการบันทึก หน้าจอดังกล่าวเป็นหน้าจอจากการเลือกลบวัตถุสินค้าในหน้าจอแสดงวัตถุสินค้าทั้งหมด



รูปที่ ง-4 หน้าจอแสดงการลบวัตถุสินค้า

การรับทอดคลาส

รูปที่ ง-5 แสดงหน้าจอวัตถุพนักงานขายทั้งหมดซึ่งเป็นตัวแทนของคลาสความสัมพันธ์แบบการรับทอดคลาส จากรูปแสดงให้เห็นว่าผู้ใช้สามารถเลือกเพิ่ม, แก้ไข, ลบวัตถุพนักงานขายได้

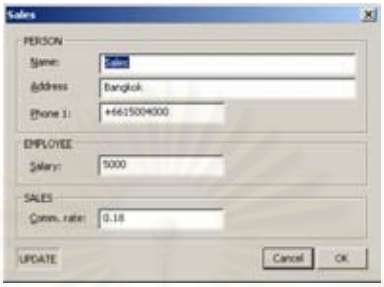


รูปที่ ง-5 หน้าจอแสดงวัตถุพนักงานขายทั้งหมด

รูปที่ ง-6 แสดงหน้าการเพิ่มวัตถุพนักงานขาย ซึ่งเป็นหน้าจอจากการเลือกเพิ่มวัตถุพนักงานขายในหน้าจอแสดงวัตถุพนักงานขายทั้งหมด

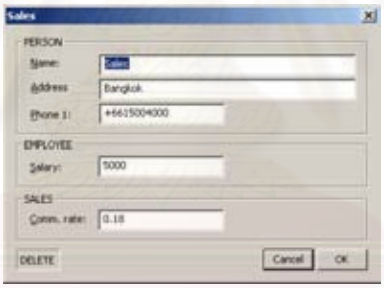
รูปที่ ง-6 หน้าจอแสดงการเพิ่มวัตถุพนักงานขาย

รูปที่ ง-7 แสดงหน้าการปรับปรุงวัตถุพนักงานขาย ซึ่งเป็นตัวแทนของคลาสความสัมพันธ์แบบการรับทอดคลาส จากรูปแสดงให้เห็นว่ามีคุณลักษณะของวัตถุ 3 ส่วนคือส่วนคลาสบุคคล, คลาสพนักงาน และคลาสพนักงานขาย



รูปที่ ง-7 หน้าจอแสดงการปรับปรุงวัตถุพนักงานขาย

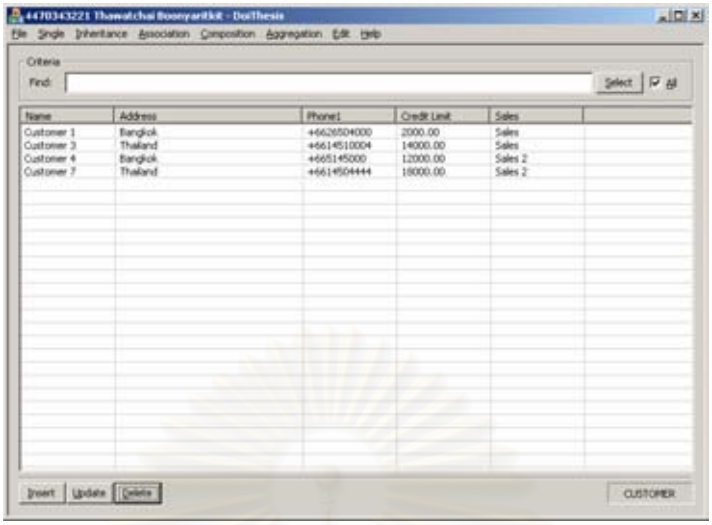
รูปที่ ง-8 แสดงหน้าจอการลบวัตถุพนักงานขายชื่อว่า Sales ซึ่งหน้าจอจะเหมือนกับหน้าจอการปรับปรุงแต่ต่างกันที่การกระทำหลังเลือกคำสั่งตกลงจะทำการลบแทนการบันทึก หน้าจอดังกล่าวเป็นหน้าจอจากการเลือกลบวัตถุสินค้าในหน้าจอแสดงวัตถุสินค้าทั้งหมด



รูปที่ ง-8 หน้าจอแสดงการลบวัตถุพนักงานขาย

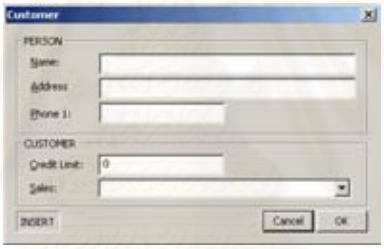
ภาพรวมกลุ่มคลาส

รูปที่ ง-9 แสดงหน้าจอวัตถุลูกค้าซึ่งเป็นตัวแทนของคลาสความสัมพันธ์แบบภาพรวมกลุ่มคลาส คือ คลาสลูกค้าอ้างถึงคลาสพนักงานขาย จากรูปแสดงให้เห็นว่าผู้ใช้สามารถเลือกเพิ่ม, แก้ไข, ลบวัตถุลูกค้าได้



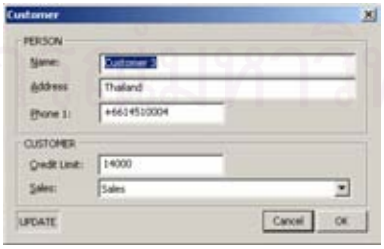
รูปที่ ง-9 หน้าจอแสดงวัตถุลูกค้าทั้งหมด

รูปที่ ง-10 แสดงหน้าการเพิ่มวัตถุลูกค้า จะเห็นว่าสามารถเลือกวัตถุพนักงานขาย ซึ่งเป็นหน้าจอจากการเลือกเพิ่มวัตถุลูกค้าในหน้าจอแสดงวัตถุลูกค้าทั้งหมด



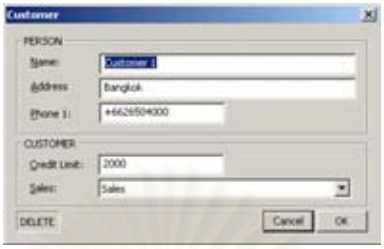
รูปที่ ง-10 หน้าจอแสดงการเพิ่มวัตถุลูกค้า

รูปที่ ง-11 แสดงหน้าการปรับปรุงวัตถุลูกค้า จะเห็นว่าสามารถเลือกวัตถุพนักงานขายใหม่ได้ ซึ่งเป็นหน้าจอจากการเลือกเพิ่มวัตถุลูกค้าในหน้าจอแสดงวัตถุลูกค้าทั้งหมด



รูปที่ ง-11 หน้าจอแสดงการปรับปรุงวัตถุลูกค้า

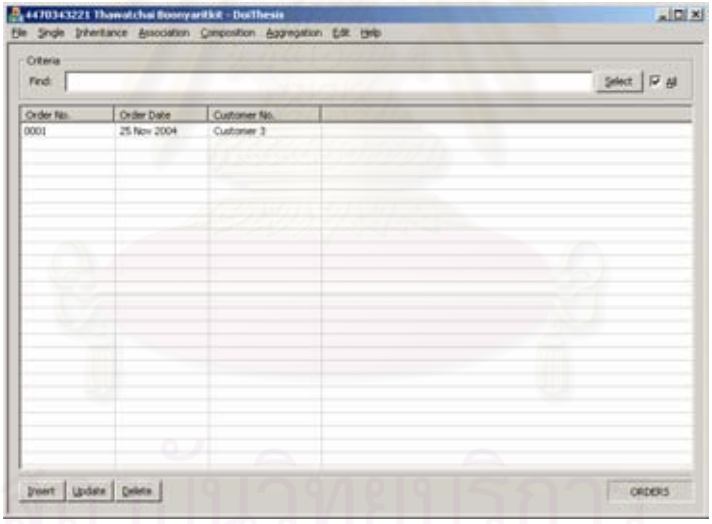
รูปที่ ง-12 แสดงหน้าจอการลบวัตถุลูกค้า จากรูปจะเห็นหน้าจอจะเหมือนกับหน้าจอการปรับปรุงวัตถุลูกค้าแต่ต่างกันที่การกระทำหลังเลือกคำสั่งตกลงจะทำการลบวัตถุแทนการบันทึก หน้าจอดังกล่าวเป็นหน้าจอจากการเลือกลบวัตถุลูกค้าในหน้าจอแสดงวัตถุลูกค้าทั้งหมด



รูปที่ ง-12 หน้าจอแสดงการลบวัตถุลูกค้า

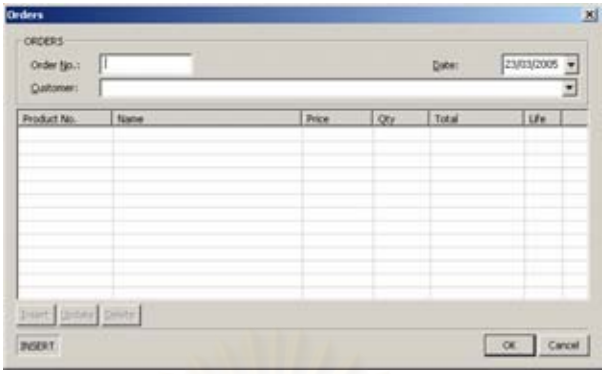
คอมโพสิตชันคลาส

รูปที่ ง-13 แสดงหน้าจอรายการของการสั่งซื้อซึ่งเป็นตัวแทนของคลาสความสัมพันธ์แบบคอมโพสิตชันคลาส คือ คลาสการสั่งซื้อประกอบด้วยคลาสรายการสั่งซื้อ จากรูปแสดงให้เห็นว่าผู้ใช้สามารถเลือกเพิ่ม, แก้ไข, ลบวัตถุการสั่งซื้อได้



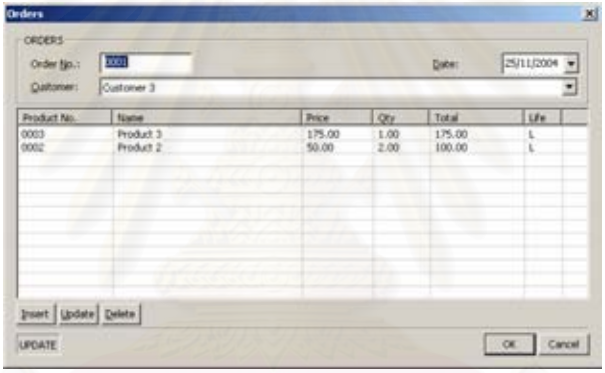
รูปที่ ง-13 หน้าจอแสดงวัตถุการสั่งซื้อทั้งหมด

รูปที่ ง-14 แสดงหน้าการเพิ่มวัตถุการสั่งซื้อ ซึ่งเป็นหน้าจอจากการเลือกเพิ่มวัตถุการสั่งซื้อในหน้าจอแสดงวัตถุสั่งซื้อทั้งหมด



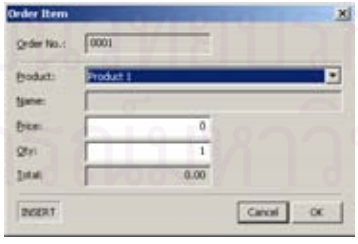
รูปที่ ง-14 หน้าจอแสดงการเพิ่มการสั่งซื้อใหม่

รูปที่ ง-15 แสดงหน้าการปรับปรุงวัตถุการสั่งซื้อหมายเลข 0001 จากรูปแสดงให้เห็นว่ามีการดึงวัตถุที่จัดเก็บไว้ของวัตถุการสั่งซื้อหมายเลข 0001 ขึ้นมา เพื่อรอการปรับปรุง หน้าจอดังกล่าวเป็นหน้าจอจากการเลือกปรับปรุงวัตถุรายการสั่งซื้อในหน้าจอแสดงวัตถุสั่งซื้อทั้งหมด



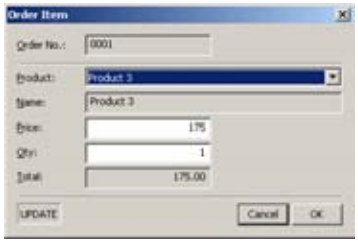
รูปที่ ง-15 หน้าจอแสดงการปรับปรุงวัตถุการสั่งซื้อ

รูปที่ ง-16 แสดงหน้าการเพิ่มวัตถุรายการสั่งซื้อของวัตถุการสั่งซื้อหมายเลข 0001 หน้าจอดังกล่าวเป็นหน้าจอจากการเลือกเพิ่มวัตถุรายการสั่งซื้อในหน้าจอปรับปรุงวัตถุสั่งซื้อ



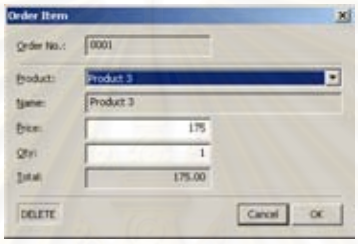
รูปที่ ง-16 หน้าจอแสดงเพิ่มวัตถุรายการสั่งซื้อใหม่

รูปที่ ง-17 แสดงหน้าการปรับปรุงวัตถุรายการสั่งซื้อของวัตถุการสั่งซื้อหมายเลข 0001 และวัตถุรายการสั่งซื้อสินค้าชนิดที่ 3 หน้าจอดังกล่าวเป็นหน้าจอจากการเลือกปรับปรุงวัตถุรายการสั่งซื้อในหน้าจอปรับปรุงวัตถุสั่งซื้อ



รูปที่ ง-17 หน้าจอแสดงการปรับปรุงวัตถุรายการสั่งซื้อ

รูปที่ ง-18 แสดงหน้าการลบวัตถุรายการสั่งซื้อของวัตถุการสั่งซื้อหมายเลข 0001 และวัตถุรายการสั่งซื้อสินค้าชนิดที่ 3 หน้าจอดังกล่าวเป็นหน้าจอจากการเลือกลบวัตถุรายการสั่งซื้อในหน้าจอปรับปรุงวัตถุสั่งซื้อ



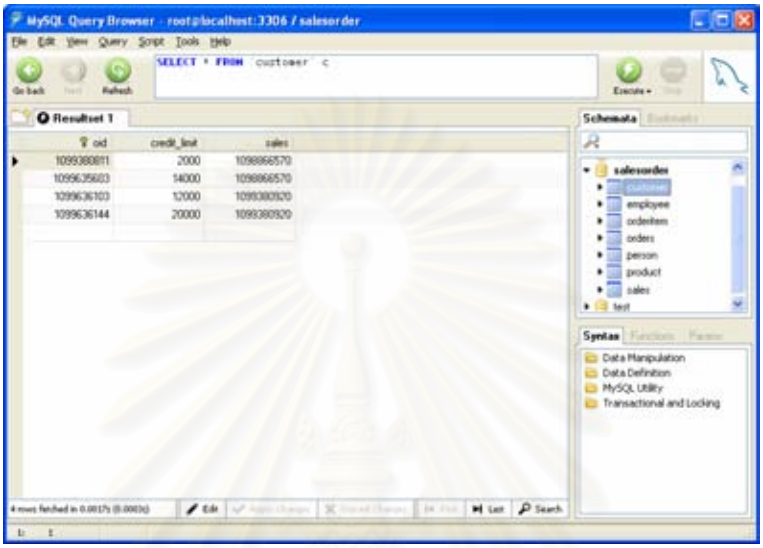
รูปที่ ง-18 หน้าจอแสดงการลบวัตถุรายการสั่งซื้อ

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

ภาคผนวก จ.

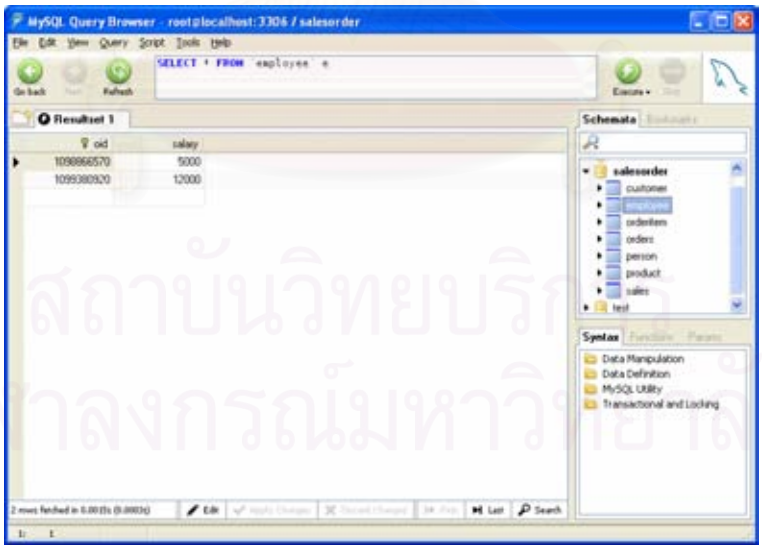
ตัวอย่างวัตถุในตารางความสัมพันธ์

ตารางลูกค้า



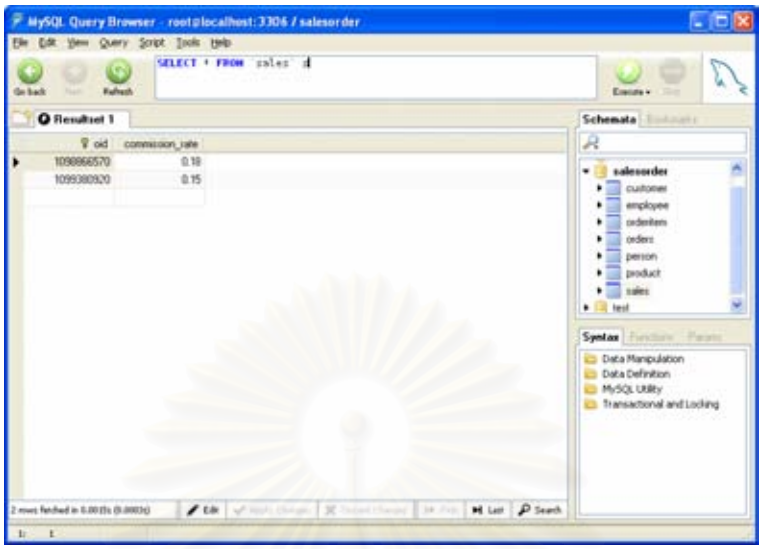
รูปที่ จ-1 ข้อมูลวัตถุลูกค้าในตารางลูกค้า

ตารางพนักงาน



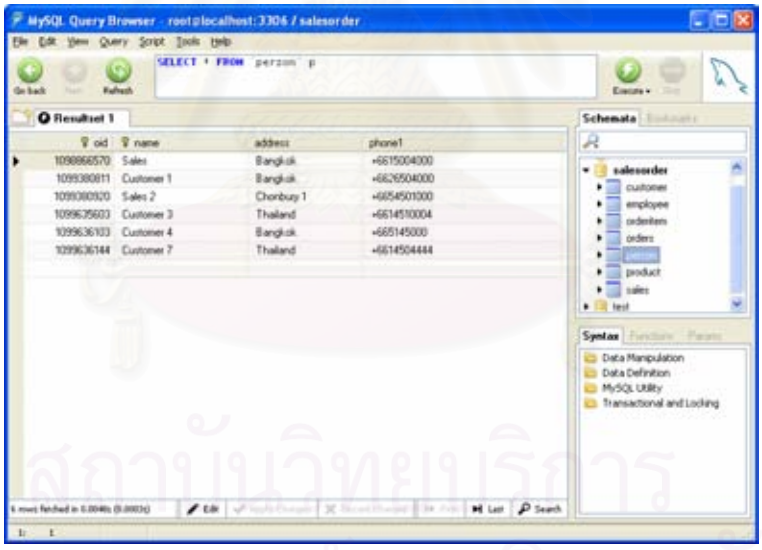
รูปที่ จ-2 ข้อมูลวัตถุพนักงานในตารางพนักงาน

ตารางพนักงานขาย



รูปที่ จ-3 ข้อมูลวัตถุพนักงานขายในตารางพนักงานขาย

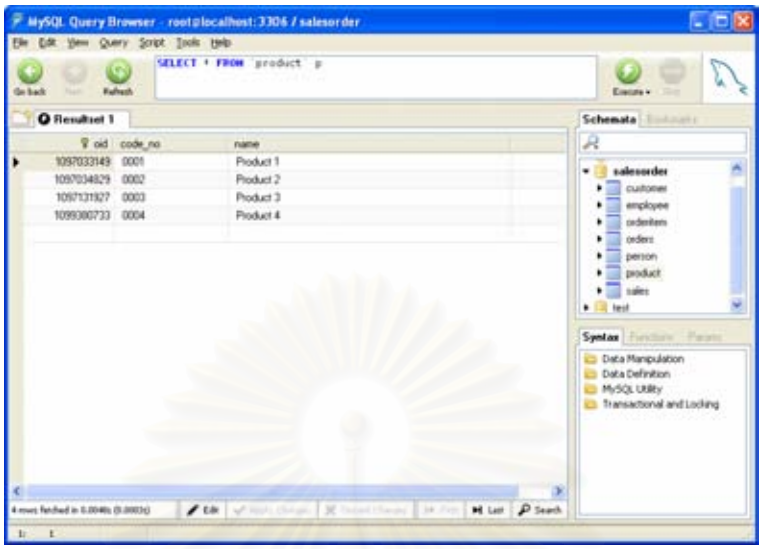
ตารางบุคคล



รูปที่ จ-4 ข้อมูลวัตถุบุคคลในตารางบุคคล

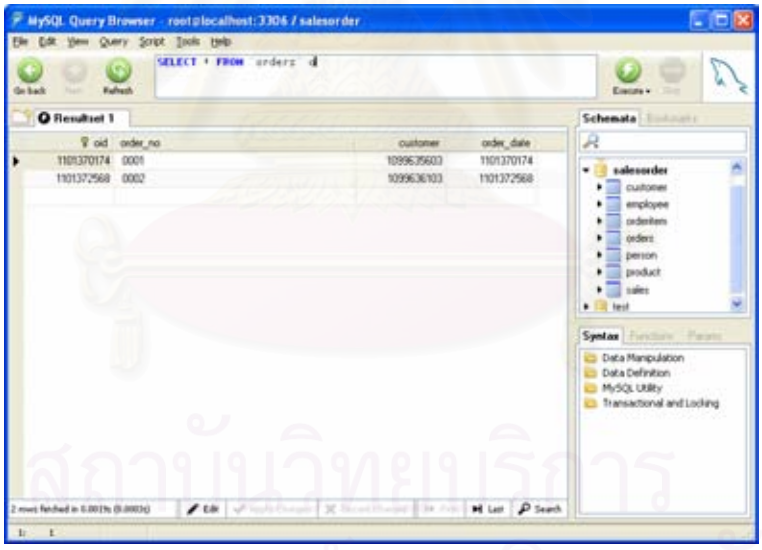
สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

ตารางสินค้า



รูปที่ จ-5 ข้อมูลวัตถุสินค้าในตารางสินค้า

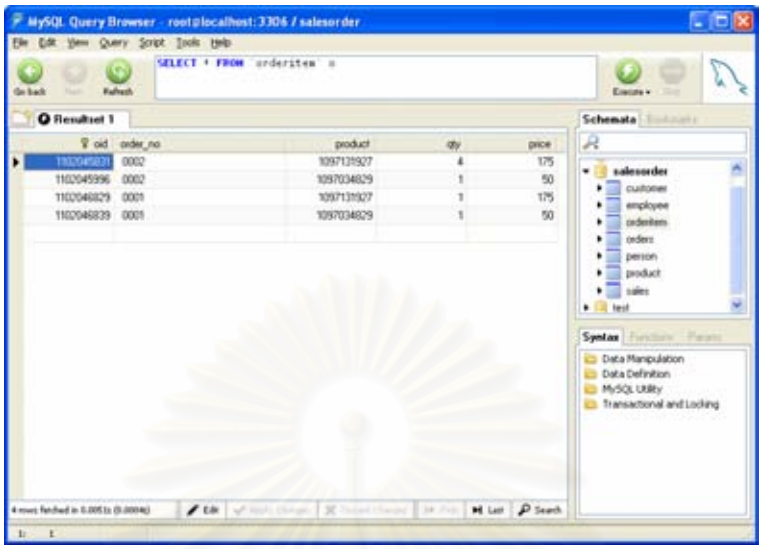
ตารางการสั่งซื้อ



รูปที่ จ-6 ข้อมูลวัตถุการสั่งซื้อในตารางการสั่งซื้อ

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

ตารางรายการสั่งซื้อ



รูปที่ จ-7 ข้อมูลวัตถุรายการสั่งซื้อในตารางรายการสั่งซื้อ

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

ภาคผนวก ฉ.

ตัวอย่างผลของชุดคำสั่งเมื่อใช้โครงร่างฯ

ซิงเกิลคลาส

```
void CDoiThesisView::ProductDlg(int action)
{
	CProductDlg dlg;
	CProduct pObject;

	pObject.SetConnect(theApp.GetConnect());

	if (action != _INSERT)

	pObject.CloneMemberValue((CProduct*)m_pObjectSet[m_wObjectList.GetSelectionMark()]);

	dlg.m_nAction = action;
	dlg.m_pObject = &pObject;
	if (dlg.DoModal() == IDOK) {
		pObject.SetCodeNo((LPCTSTR)dlg.m_code_no);
		pObject.SetName((LPCTSTR)dlg.m_name);
		pObject.SetPrice((float)dlg.m_price);

		if (action == _DELETE)
			pObject.Delete();

		pObject.Save();
	}
}
```

การรับทอดคลาส

```
void CDoiThesisView::SalesDlg(int action)
{
	CSalesDlg dlg;
	CSales pObject;

	pObject.SetConnect(theApp.GetConnect());

	if (action != _INSERT)

	pObject.CloneMemberValue((CSales*)m_pObjectSet[m_wObjectList.GetSelectionMark()]);

	dlg.m_nAction = action;
	dlg.m_pObject = &pObject;
	if (dlg.DoModal() == IDOK) {
		pObject.SetName((LPCTSTR)dlg.m_name);
		pObject.SetAddress((LPCTSTR)dlg.m_address);
		pObject.SetPhone1((LPCTSTR)dlg.m_phone1);

		pObject.SetSalary(dlg.m_salary);

		pObject.SetCommissionRate(dlg.m_commission_rate);

		if (action == _DELETE)
			pObject.Delete();

		pObject.Save();
	}
}
```

ภาพรวมกลุ่มคลาส

```
void CDoiThesisView::CustomerDlg(int action)
{
	CCustomerDlg dlg;
	CCustomer pObject;

	pObject.SetConnect(theApp.GetConnect());

	if (action != _INSERT)

	pObject.CloneMemberValue((CCustomer*)m_pObjectSet[m_wObjectList.GetSelectionMark()]);

	dlg.m_nAction = action;
	dlg.m_pObject = &pObject;
	if (dlg.DoModal() == IDOK) {
		pObject.SetName((LPCTSTR)dlg.m_name);
```

```
                pObject.SetAddress((LPCTSTR)dlg.m_address);
                pObject.SetPhone1((LPCTSTR)dlg.m_phone1);
                pObject.SetCreditLimit((float)dlg.m_credit_limit);
                pObject.SetSales((CSales*)dlg.m_pSales[dlg.m_nSales]);

                if (action == _DELETE)
                        pObject.Delete();

                pObject.Save();
        }
}
```

คอมโพสิตชันคลาส

```
void CDoiThesisView::OrdersDlg(int action)
{
        COrdersDlg dlg;
        COrders pObject;

        pObject.SetConnect(theApp.GetConnect());

        if (action != _INSERT)

        pObject.CloneMemberValue((COrders*)m_pObjectSet[m_wObjectList.GetSelectionMark()]);

        dlg.m_nAction = action;
        dlg.m_pObject = &pObject;
        if (dlg.DoModal() == IDOK) {
                pObject.SetOrderNo((LPCTSTR)dlg.m_order_no);
                pObject.SetDate(dlg.m_dtDate.GetTime());
                pObject.SetCustomer((CCustomer*)dlg.m_pCustomer[dlg.m_nCustomer]);

                // don't call seting value because it's set in orderitem dialog
                if (action == _DELETE)
                        pObject.Delete();

                pObject.Save();
        }
}
```

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

ประวัติผู้เขียนวิทยานิพนธ์

นายธวัชชัย บุญยฤทธิ์กิจ สำเร็จการศึกษาปริญญาวิทยาศาสตรบัณฑิตสาขาวิทยาการคอมพิวเตอร์จากมหาวิทยาลัยหอการค้าไทย ในปีการศึกษา 2544 จากนั้นเข้าศึกษาต่อในหลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิทยาศาสตร์คอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย เมื่อปีการศึกษา 2547

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย