การแปลงเปลี่ยนระบบฐานข้อมูลเชิงวัตถุ เป็นระบบฐานข้อมูลเชิงวัตถุสัมพันธ์

A TRANSFORMATION FROM AN OBJECT DATABASE TO AN OBJECT RELATIONAL DATABASE

กอบชัย นิยมธรรม

KOBCHAI NIYOMTHUM

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต

สาขาวิชาเทคโนโลยีสารสนเทศ

บัณฑิตวิทยาลัย

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ. 2546

ISBN 974-324-698-3

การแปลงเปลี่ยนระบบฐานข้อมูลเชิงวัตถุ เป็นระบบฐานข้อมูลเชิงวัตถุสัมพันธ์

A TRANSFORMATION FROM AN OBJECT DATABASE TO AN OBJECT RELATIONAL DATABASE

กอบชัย นิยมธรรม

KOBCHAI NIYOMTHUM

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต

สาขาวิชาเทคโนโลยีสารสนเทศ

บัณฑิตวิทยาลัย

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ.2546

ISBN 974-324-698-3

A TRANSFORMATION FROM AN OBJECT DATABASE TO AN OBJECT RELATIONAL DATABASE

KOBCHAI NIYOMTHUM

A THESIS SUBMITTED IN PATIAL FULFILMENT
OF TIIE REQUIREMENT FOR TIIE DEGREE OF
MASTER OF SCIENCE IN INFORMATION TECHNOLOGY
SCHOOL OF GRADUATE STUDIES
KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG
2003

ISBN 974-324-698-3

บัณฑิตวิทยาลัย

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ใบรับรองวิทยานิพนธ์

หัวข้อวิทยานิพนธ์ การปรับเปลี่ยนระบบฐานข้อมูลเชิงวัตถุเป็นระบบฐานข้อมูลเชิงวัตถุสัมพันธ์
A TRANSFORMATION FROM AN OBJECT DATABASE TO AN OBJECT RELATIONAL DATABASE

ชื่อนักศึกษา นายกอบชัย นิยมธรรม

รหัสประจำตัว 42067007

ปริญญา วิทยาศาสตรมหาบัณฑิต

สาขาวิชา เทคโนโลยีสารสนเทศ

อาจารย์ผู้ควบคุมวิทยานิพนธ์ รศ.ดร.ศุภมิตร จิตตะยโศธร

คณะกรรมการสอบวิทยานิพนธ์		ลายมือชื่อ
รศ.ดร.ศุภมิตร	จิตตะยโศธร	
รศ.ดร.บุญวัฒน์	อัตชู	
รศ.ดร.วิเชียร	เปรมชัยสวัสดิ์	
ดร.ภัทรชัย	ลลิตโรจน์วงศ์	
ดร.ธนารัตน์	ชลิดาพงศ์	

วัน/เดือนปี ที่สอบ 16 กรกฎาคม 2546 เวลา 10.00 น. เป็นต้นไป

สถานที่สอบ ณ ห้อง M22 (ชั้นลอย) อาคารเรียนรวมและปฏิบัติการคณะเทคโนโลยีสารสนเทศ

บัณฑิตวิทยาลัยรับรองแล้ว

(รศ.ดร.บุญวัฒน์ อัตชู)

คณบดีบัณฑิตวิทยาลัย

วันที่ 25 **เดือน** กรกฎาคม **พ.ศ.** 2546

หัวข้อวิทยานิพนธ์	การปรับเปลี่ยนระบบฐานข้อมูลเชิงวัตถุ เป็นระบบฐานข้อมูลเชิงวัตถุสัมพันธ์
นักศึกษา	นายกอบชัย นิยมธรรม
รหัสประจำตัว	42067007
ปริญญา	วิทยาศาสตรมหาบัณฑิต
สาขาวิชา	วิทยาการสารสนเทศ
พ.ศ.	2546
อาจารย์ผู้ควบคุมวิทยานิพนธ์	รศ.ดร.สุภมิตร จิตตะยโศธร

บทคัดย่อ

ระบบฐานข้อมูลเชิงวัตถุ (Object Database) ซึ่งหมายถึง ระบบฐานข้อมูลเชิงวัตถุ (Pure Object Oriented Database) และระบบฐานข้อมูลเชิงวัตถุสัมพันธ์ (Object Relational Database) สามารถสนับสนุนระบบสารสนเทศที่ต้องการโครงสร้างข้อมูลที่ซับซ้อนได้เป็นอย่างดี แนวคิดของทั้งสองโมเดล แตกต่างกันที่แนวคิดเรื่องการห่อหุ้มข้อมูล (Encapsulation) และความหลากหลายของโครงสร้างข้อมูลระดับแอททริบิวซ์ รวมถึงเครื่องมือที่ใช้สำหรับระบบฐานข้อมูลทั้งสองประเภท จะมีความโดดเด่นที่แตกต่างกัน อาทิ เช่น เครื่องมือที่ใช้สำหรับจัดการคลังข้อมูล (Information Warehouse) บนระบบฐานข้อมูลเชิงวัตถุสัมพันธ์จะเป็นที่แพร่หลายและง่ายต่อการประมวลผล ข้อมูลที่ใช้ในงานดังกล่าว อาจมาจากหลายแหล่งและหลายโมเดล ทำให้เกิดความจำเป็นในการแปลงข้อมูล ให้อยู่ในโมเดลเดียวกัน เพื่อนำมาใช้งานร่วมกับข้อมูลจากระบบอื่น

งานวิจัยนี้ได้เสนอกระบวนการปรับเปลี่ยนข้อมูลในระบบฐานข้อมูลเชิงวัตถุ มาสู่ระบบฐานข้อมูลเชิงวัตถุสัมพันธ์ โดยการปรับเปลี่ยนโครงสร้างเดิม ให้อยู่ในรูปที่ระบบฐานข้อมูลเชิงวัตถุสัมพันธ์เข้าใจและทำงานได้ถูกต้อง โดยกระบวนการปรับเปลี่ยนจะประกอบด้วย การอ่านและวิเคราะห์โครงสร้างข้อมูลจากระบบฐานข้อมูลเชิงวัตถุและแปลให้อยู่ในรูปโครงสร้างระบบฐานข้อมูลเชิงวัตถุสัมพันธ์ และในส่วนที่เป็นเมดธอด ระบบจะวิเคราะห์เมดธอดต่าง ๆ ให้และอนุญาตให้แก้ไขชุดคำสั่งในระหว่างการแปลง

Thesis Title	TRANSFORMATION FROM AN OBJECT DATABASE TO AN OBJECT RELATIONAL DATABASE
Student	Mr.Kobchai Niyomthum
Student ID.	42067007
Degree	Master of Science
Program	Information Science
Year	2003
Thesis Advisor	Assoc.Prof.Dr.Suphamit Chittayasothorn

ABSTRACT

This paper proposes a method for transforming database systems from an Object Database into an Object Relational Database. The result after the transformation, the system still has the same semantics and functions. Both Object database and Object-Relational Database are based on an object oriented technology but they are different in the encapsulation concept and also many features in an Object database are not supported in an Object-Relation database. Some software tools are designed to work on a relational or an object relational database rather than an object database. The transformation methodology will show techniques for analyze an object oriented model and transform it into an Object-Relational model. This proposed scheme allows user to edit these structures and methods before processing. Furthermore, commands and languages in any products are different. We will analyze a method's signature and allow users to correct them before the generation of database structures.

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้ สำเร็จลุล่วงได้ด้วยดี ด้วยความรู้ต่าง ๆ, คำแนะนำ, คำปรึกษา และแนวทางการทำวิจัย จาก รศ.ดร.ศุภมิตร จิตตะยโศธร ผู้อำนวยการสำนักวิจัยและบริการคอมพิวเตอร์ อาจารย์ผู้ควบคุมวิทยานิพนธ์ ผู้วิจัยรู้สึกซาบซึ้ง และกราบขอบพระคุณในความอนุเคราะห์และสนับสนุนเป็นอย่างสูง

กราบขอบพระคุณด้วยความเคารพและศรัทธาอย่างสูงสุด ด้วยพระคุณจากคุณพ่อประภัสร บิดาผู้ล่วงลับ, คุณแม่ศรียา มารดา ผู้ให้กำเนิด อบรมสั่งสอน และชี้นำให้ผู้วิจัยศึกษาด้านคอมพิวเตอร์, คุณย่าชลอ นิยมธรรม ผู้ล่วงลับ ด้วยความรักและผูกพันด้วยกำลังใจจวบจนวาระสุดท้าย, คุณอาขวัญชัย และญาติทุกท่านที่กรุณาสนับสนุน และแนะนำด้วยดีเสมอมา

ขอขอบพระคุณคณาจารย์ทุกท่าน ผู้ประสิทธิประสาทวิชาความรู้ ทำให้สามารถนำความรู้มาใช้ในงานวิจัย และขอบคุณบุคลากรคณะเทคโนโลยีสารสนเทศ, บัณฑิตวิทยาลัยทุกท่าน และคุณณัฐรียา พริ้งสกุลชัย เลขานุการผู้อำนวยการ สำนักวิจัยฯ สจล. สำหรับการอำนวยความสะดวกและประสานงานต่าง ๆ ด้วยดี

ขอขอบพระคุณ มหาวิทยาลัยแม่โจ้ หน่วยงานต้นสังกัด และทบวงมหาวิทยาลัย ที่สนับสนุนการศึกษาต่อระดับสูง และการสนับสนุนทุนพัฒนาอาจารย์ ระดับโท-เอก

ขอบคุณ IBM Toronto Lab, Canada และ IBM ประเทศไทย โดยคุณนิธิทัศน์ อุปถัมภ์วิภานนท์ และคุณอังคณา ล้อกิตติกูล ในการสนับสนุนซอฟแวร์ IBM DB2 UDB และการฝึกอบรม ขอบคุณ InterSystems โดย Mr. Phil Pybus และบริษัท วอร์เดน อินเตอร์เนชั่นแนล จำกัด โดยคุณวิชัย และคุณศุภศรี ตั้งสุขนิรันดร ในการสนับสนุนซอฟแวร์คาเช่ และการฝึกอบรม

ขอบคุณกำลังใจสำคัญจาก คุณธนวรรธน์, คุณน้ำฝน นิยมธรรม น้องชายและน้องสาว, คุณวิมล มาศ เผ่าจินดา, คุณผุสดี พรผล, คุณเธียรชัย พัฒนศิริเวทิน, คุณพิบูลย์ เทพบุตร, คุณมนต์ชัย , คุณโกศล, คุณบุญลือ และเพื่อนนักศึกษา รุ่น 7.1 สำหรับกำลังใจและความช่วยเหลือในการทำวิจัยอย่างสม่ำเสมอ

กอบชัย นิยมธรรม

สารบัญ

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญรูป

รูปที่ หน้า

สารบัญรูป (ต่อ)

รูปที่ หน้า

สารบัญตาราง

ตารางที่ หน้า

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ในปัจจุบัน องค์กรต่าง ๆ มักจะมีระบบฐานข้อมูลที่ต่างกันหลายโมเดล และหลากหลายแพลทฟอร์ม เครื่องมือที่ใช้ จึงมีความแตกต่างกันทั้งรูปแบบ และกระบวนการใช้งาน ระบบฐานข้อมูลเชิงวัตถุ (Object Database) ซึ่งหมายถึง ระบบฐานข้อมูลเชิงวัตถุ (Pure Object Oriented Database) และ ระบบฐานข้อมูลเชิงวัตถุสัมพันธ์ (Object Relational Database) ทั้งสองประเภท สามารถสนับสนุนระบบสารสนเทศที่ต้องการโครงสร้างข้อมูลที่ซับซ้อนได้เป็นอย่างดี อย่างไรก็ตาม เครื่องมือที่ใช้สำหรับระบบฐานข้อมูลทั้งสองประเภท มักจะมีความโดดเด่นแตกต่างกัน อาทิ เครื่องมือที่ใช้สำหรับจัดการคลังข้อมูล (Information Warehouse) บนระบบฐานข้อมูลเชิงวัตถุสัมพันธ์ (Object Relational Database) จะมีประสิทธิภาพ และง่ายต่อการประมวลผล ในขณะเดียวกัน ข้อมูลที่ใช้อาจมาจากหลายแหล่ง จากหลายโมเดล เช่น จากโมเดลเชิงวัตถุ ทำให้เกิดความจำเป็นในการแปลงข้อมูล ให้อยู่ในรูปโมเดลเชิงวัตถุสัมพันธ์ โดยยังคงความสมบูรณ์ของเดิมไว้ได้ และนำมาใช้งานร่วมกันกับข้อมูลจากระบบอื่น ๆ

1.2 ความมุ่งหมายและวัตถุประสงค์ของการศึกษา

เพื่อถ่ายโอนระบบฐานข้อมูลเชิงวัตถุ (Object Oriented Database) มาสู่ระบบฐานข้อมูลเชิงวัตถุสัมพันธ์ (Object-Relational Database) โดยคงความถูกต้องของข้อมูลเดิม เพื่อนำข้อมูลจากแหล่งข้อมูลต่างโมเดลมาใช้ร่วมกันในงานต่าง ๆ เช่น งานคลังข้อมูลได้

1.3 สมมติฐานของการศึกษา

ในงานวิจัยนี้ ระบบสารสนเทศที่พัฒนาบนระบบฐานข้อมูลเชิงวัตถุ จะเป็นระบบงานที่จะถูกปรับเปลี่ยนมาสู่ระบบสารสนเทศใหม่ที่ทำงานบนระบบฐานข้อมูลเชิงวัตถุสัมพันธ์ โดยกระบวนการปรับเปลี่ยน จะวิเคราะห์โครงสร้างข้อมูลของระบบสารสนเทศ และปรับเปลี่ยนให้อยู่ในรูปโมเดลเชิงวัตถุสัมพันธ์ และในส่วนของเมดธอด ผู้ใช้ สามารถปรับเปลี่ยนชุดคำสั่งต่าง ๆ ให้เหมาะสม โดยการทำงานทั้งหมด จะต้องง่ายและสะดวกต่อการทำงาน และได้ผลลัพธ์ที่ถูกต้องเหมือนระบบงานเดิม

1.4 ทฤษฎีและแนวคิดที่ใช้ในการวิจัย

ระบบฐานข้อมูลเชิงวัตถุ คือ ระบบฐานข้อมูลที่นำแนวคิดเชิงวัตถุเข้ามาประยุกต์ใช้ ระบบฐานข้อมูลดังกล่าว ในปัจจุบันมี 2 โมเดล ได้แก่ ระบบฐานข้อมูลเชิงวัตถุ (Object Oriented Database) และ ระบบฐานข้อมูลเชิงวัตถุสัมพันธ์ (Object Relational Database) โดยทั้งสองโมเดลต่างใช้คุณสมบัติและแนวคิดเชิงวัตถุ และสามารถสนับสนุนระบบสารสนเทศ ที่ต้องการโครงสร้างข้อมูลที่ซับซ้อนได้เป็นอย่างดีก็ตาม อย่างไรก็ดี ทั้งสองโมเดล มีความแตกต่างกันในเรื่องของแนวคิดการห่อหุ้มข้อมูล (Encapsulation) กล่าวคือ ระบบฐานข้อมูลเชิงวัตถุ จะห่อหุ้มข้อมูลทั้งหมดไว้ และจะต้องติดต่อผ่านทางเมดธอดที่กำหนดไว้เท่านั้น แต่ระบบฐานข้อมูลเชิงวัตถุสัมพันธ์ จะอนุญาตให้เข้าถึงแอตทริบิวต์ต่าง ๆ ได้ โดยผู้ใช้ สามารถเข้าถึงข้อมูลได้โดยตรงในแต่ละแอตทริบิวต์ นั่นคือการขาดคุณสมบัติ Encapsulation หรือจะเข้าถึงโดยผ่านทางเมดธอดที่กำหนดไว้ก็ได้ นอกจากนี้ วิธีการพัฒนาและการจัดการ มีความแตกต่างกัน ทั้งนี้เนื่องจากการมองโมเดลข้อมูลต่างกัน โดยระบบฐานข้อมูลเชิงวัตถุ มองข้อมูลเป็นวัตถุ (Real World Object) ในขณะที่ระบบฐานข้อมูลเชิงวัตถุสัมพันธ์ แม้จะอนุญาตให้มองข้อมูลเป็นวัตถุได้ แต่ในที่สุดวัตถุเหล่านั้น จะต้องแปลงให้อยู่ในรูปของตาราง (Relation) และใช้ภาษา SQL ในการจัดการข้อมูล

นอกจากนี้ เครื่องมือที่ใช้บนระบบฐานข้อมูลทั้งสองประเภทก็มีความแตกต่างกัน เช่น เครื่องมือสำหรับงานคลังข้อมูล บนระบบฐานข้อมูลเชิงวัตถุสัมพันธ์ ก็เป็นที่นิยมและใช้งานกันอย่างแพร่หลาย และง่ายต่อการประมวลผล ในส่วนของข้อมูลที่ใช้ในงานดังกล่าว อาจมาจากหลายแหล่งและหลายโมเดล ทำให้เกิดความจำเป็นในการแปลงข้อมูล ให้อยู่ในรูปแบบหรือโมเดลเดียวกัน เพื่อนำมาใช้งานร่วมกับข้อมูลจากระบบอื่นได้

1.5 งานวิจัยที่เกี่ยวข้อง

S.Y.Liao, และคณะ[21] กล่าวถึง แนวคิดในการแปลงระบบฐานข้อมูลเชิงสัมพันธ์ มาเป็นระบบฐานข้อมูลเชิงวัตถุ โดยวิธีการ Reengineering มีวิธีการหลัก 2-แนวทาง-คือ-การแปลงในส่วน Static และส่วนที่เป็น Dynamic โดยการแปลงแบบ Static เป็นการแปลงโครงสร้างข้อมูล (Data Structure) และ แบบ Dynamic เป็นการแปลงส่วนที่เป็นพฤติกรรมของระบบ (Behavior)
หลักการแปลงในส่วน Dynamic จะใช้วิธีการแปลงในส่วนต่าง ๆ ต่อไปนี้

(1) แปลง Procedures เป็น Methods

(2) แปลง Procedures Call เป็น Messages

(3) แปลง Constraints เป็น Constraints

จากนั้น จะต้องแปลงโมเดลเชิงสัมพันธ์กลับมาเป็น ER Model และแปลง ให้เป็นโมเดลเชิงวัตถุ ดังนี้

(1) แปลงโมเดลเชิงสัมพันธ์เป็น ER Model

(2) แต่ละ Entity Type แปลงเป็น Object Class

(3) Artificial Identity ที่อยู่ในบางแอตทริบิวต์ เช่น Primary key บางตัว สามารถละเว้นได้ ถ้าแอตทริบิวต์นั้น ไม่มีค่าอะไรเมื่อแปลงมาอยู่ในรูป Instance Variables แล้ว

(4) Link ยังคงมีอยู่ แต่ขึ้นอยู่กับความสัมพันธ์ระหว่าง Entities

(5) ความสัมพันธ์แบบ 1-m ระหว่าง Entity โดยปกติจะใช้ Primary Key ในฝั่งความสัมพันธ์ด้าน 1 เมื่อทำการแปลงมาเป็น Object แล้ว ตัว FK จะถูกแทนด้วย Direct Reference ที่จะชี้ไปที่คลาสที่เกี่ยวข้อง

(6) ความสัมพันธ์แบบ M-N โดยทั่วไปจะใช้ Primary Key ซึ่งมีลักษณะแบบ Composition ซึ่งมาจากแต่ละ Entity Type เป็นตัวเชื่อมความสัมพันธ์ แต่ในโมเดลเชิงวัตถุ จะใช้แอตทริบิวต์ชี้ตรงไปที่ออบเจคคลาสที่เกี่ยวข้อง ซึ่งถ้าในรีเลชั่นใหม่ที่เกิดขึ้น ไม่มีแอตทริบิวต์ที่เป็น Non-Key เราสามารถละไว้ได้ โดยความสัมพันธ์ระหว่างสอง Entity จะอยู่ในรูป Aggregate Object

(7) Association type ต่าง ๆ เช่น qualified association, ternary association สามารถเอาออกได้

(8) แนวคิดเกี่ยวกับ Generalization, Aggregation ไม่สนับสนุนจาก ER Model แต่สามารถกำหนดได้ตามความเหมาะสมถ้ามีความสัมพันธ์ปรากฏ

(9) Procedure สามารถแปลงเป็นเมดธอดลงในคลาสที่เกี่ยวข้องให้ถูกต้อง

(10) Procedure Call ที่ใช้ในการเรียก Procedure แปลงไปเป็น Message ซึ่งใช้ในการเรียกเมดธอด

(11) Constraints สำหรับการควบคุม Integrity Control สามารถละเว้นได้ระหว่างแปลง

Martin Gogolla, และคณะ [12] นำเสนอวิธีการทำ Reverse Engineering ดังนี้

(1) Basic Data Format ได้แก่ การพิจารณารายละเอียดของข้อมูลพื้นฐานที่เกี่ยวข้อง

(2) การสร้าง Semantic Database Schema คือ แปลง EER Model ให้อยู่ในรูปแบบของ ODMG

(3) แปลง Schema ที่ได้ให้อยู่ในรูป Object Model

(4) เริ่มพัฒนา และสร้างระบบฐานข้อมูลเชิงวัตถุจริง โดยการทำงานทั้งหมด จะอยู่บนพื้นฐานและหลักการของ Object Type ตามมาตรฐาน ODMG ในส่วนของ Object Definition Language (ODL) ซึ่งระบุไว้ว่า

1. Instance Properties ของออบเจค คือ แอตทริบิวต์และความสัมพันธ์ของตัวอินสแตนท์
2. ODMG Object Model จะสนับสนุนทั้ง Data Values และ Object-Valued attribute ที่เชื่อมโยงไปยังออบเจคอื่นที่เกี่ยวข้อง
3. Relationship หรือความสัมพันธ์ ถูกกำหนดให้เป็นตัวเชื่อมโยงระหว่างอินสแตนท์ ซึ่งในออบเจคโมเดลนี้ จะสนับสนุนเฉพาะ Binary-Relationships เท่านั้น ไม่สามารถกำหนดความสัมพันธ์แบบ N-ary ได้เหมือนใน ER Model
4. Cardinality ถ้ามีค่ามากกว่าหนึ่ง สามารถใช้ในรูป Collection ได้ เช่น Set, List, Bag เป็นต้น
5. พฤติกรรมของวัตถุ (Behavior) ถูกกำหนดโดยผ่าน Operations ซึ่งจะมี Signature ที่ประกอบด้วย ชื่อพารามิเตอร์, ประเภทข้อมูล ของแต่ละตัว และประเภทข้อมูลที่ส่งกลับ (Returned type)

Andrew McAllister. [2] เป็นงานวิจัยในการทำ Reverse Engineering ของระบบฐานข้อมูลการแพทย์ ซึ่งพบปัญหาสำคัญ คือ ระบบถูกพัฒนา โดยไม่มีการสร้าง Data Model มาก่อน และได้มีการพัฒนาต่อ โดยไม่มีเอกสารประกอบการอ้างอิง โดยงานวิจัยใช้ 3 แนวทางในการวิเคราะห์เพื่อให้ทราบลักษณะของระบบ และนำมาสร้างเป็น ER Model คือ

(1) ตั้งสมมติฐานบนการออกแบบฐานข้อมูล
(2) พิจารณาจากข้อมูลนำเข้า
(3) จากสิ่งที่เกี่ยวข้องกับระบบ รวมถึงการขอข้อมูล จากผู้เชี่ยวชาญในแต่ละส่วนประกอบกัน

งานวิจัยที่เกี่ยวข้องกับการแปลงระบบ โดย Michale Zimmer. [14] กล่าวถึงการแปลงระบบเดิม (Older Information) ไปสู่ระบบใหม่ แบ่งการแปลงออกเป็น 2 ระดับ คือ การแปลงในระดับลอจิคอล ได้แก่ การแปลง Entity set, Entity, Attribute ที่อยู่ในรูป File, Record, Field และการแปลงในระดับฟิสิคอล ได้แก่ Table, row, Column ในระบบฐานข้อมูล ปัญหาที่พบ คือ หลังการแปลงอาจได้ผลลัพธ์ที่ไม่ถูกต้องหรือเข้ากันไม่ได้ระหว่างโมเดล นั่นคือ Data Quality, Incomplete Data โดยมีกระบวนการวางแผนการแปลงดังนี้ คือ

(1) วิเคราะห์ Physical Data Model
(2) ตรวจสอบชนิดของข้อมูล
(3) วิเคราะห์ Logical Data Model ของระบบเดิม
(4) วิเคราะห์ Logical Data Model ของระบบใหม่
(5) วิเคราะห์ Physical Data Model ของระบบใหม่

(6) พิจารณาการแปลงในแต่ละส่วน (Mapping)

(7) พิจารณาวิธีการจัดการกับข้อมูลที่ไม่ถูกต้อง

นอกจากนี้ ในระหว่างการแปลงมีการใช้เทคนิคทางการเขียนโปรแกรม เพื่ออนุญาตให้แก้ไข Business rule ได้, เขียนรายการทำงานที่เกิดลง Log File เพื่อตรวจสอบผลการทำงานที่เกิดขึ้นได้ และจากงานวิจัย พบปัญหาความไม่ถูกต้องของข้อมูลภายหลังการแปลง (Type of Abnormality) ได้แก่

(1) ข้อมูลบางส่วนไม่ Unique

(2) ค่า Cardinality ไม่ถูกต้อง

(3) Wrong Optionally คือการที่ข้อมูลที่ควรมีอยู่ขาดหายไป

(4) Orphaned Records คือการที่ข้อมูลที่มีความสัมพันธ์กับตารางหลัก ไม่มี Patent

(5) Inconsistent Redundancy ได้แก่ ปัญหาที่เกิดขึ้นมาเนื่องจากการพัฒนาที่หลากหลาย เช่น หลายทีม, หลายครั้ง เป็นต้น โดยพิจารณาความเปลี่ยนแปลงต่าง ๆ ที่เกิดขึ้นมาก่อนทั้งในส่วนของ Business rules และค่าของข้อมูล

(6) Data Inconsistencies ได้แก่ ความไม่ถูกต้องของข้อมูล เช่น การแสดงช่วงของเวลา (Date/Time Dimension), ขนาดของประเภทข้อมูลเหมาะสมและรองรับได้หรือไม่, การเรียงลำดับค่าวันที่ เป็นต้น

Ringo Pang [18] กล่าวถึงหลักการแปลงจาก Object-Oriented Database เป็น Relational Database ดังต่อไปนี้

1. Schema Translation from Object Oriented to Relational โดย Schema จาก Object Database จะใช้สำหรับบอกว่ามีแอตทริบิวต์ใดที่ยังไม่ถูกเรียกใช้ และคลาส, แอตทริบิวต์, เมดธอด จะถูกเก็บไว้ด้วยกันในแต่ละออบเจค (Object) แต่สำหรับ Relational Database ใช้สำหรับกำหนดว่าข้อมูลจะถูกเก็บไว้ที่ไหนและอย่างไร ซึ่งจะจัดเก็บในรูปของเรคคอร์ด ในส่วนของ Triggers, Stored Procedures จะถูกแยกเก็บไว้เป็นออบเจคต่างหาก โดยไม่ได้นำมารวมกันเหมือน Object Database สำหรับวิธีการแปลงจาก Object Database มาเป็น Relational Database มีหลักการ คือ

   (1) แปลงคลาสให้เป็นตาราง

   (2) แปลงแอตทริบิวต์ในคลาสให้เป็นแอตทริบิวต์ในตาราง

   (3) Primary Key สามารถใช้ Oid เดิมซึ่งระบบเป็นผู้สร้าง หรืออาจเลือกจากแอตทริบิวต์ที่เหมาะสมก็ได้

   (4) Set-Values ถ้าแอตทริบิวต์ใด ที่มีลักษณะเป็นเซ็ต ให้แปลงให้เป็นอีกตาราง และกำหนด Primary Key ที่จะชี้กลับไปยังตารางเดิมด้วย

(5) ความสัมพันธ์แบบ One-to-One อาจนำคลาสทั้งสอง มารวมกันเป็นตารางเดียวกันได้ และใช้ Primary Key ตัวเดียวกัน โดยจะเพิ่มประสิทธิภาพในการทำงานเร็วขึ้น แต่จะเป็นการฝ่าฝืนกฎเกณฑ์ Normalization หรือ อาจแปลงคลาสทั้งสอง ให้อยู่ในรูปตารางสองตาราง และกำหนด Primary Key ไว้ในทั้งสองตารางเพื่อเชื่อมโยงกัน

(6) ความสัมพันธ์แบบ One-to-Many ให้แปลงเป็นสองตาราง และกำหนดให้ตารางในด้าน Many ให้นำ Primary Key จากฝั่ง One มาไว้เป็น Foreign Key

(7) ความสัมพันธ์แบบ Many-to-Many จะต้องสร้างขึ้นเป็นตารางใหม่ และนำ Primary Key จากทั้งสองตารางมารวมเป็น Combined Primary Key ในตารางใหม่

(8) Inheritance ให้แปลง Super Class เป็นตาราง และแต่ละ Sub-Class แปลงเป็นตารางภายใต้ Super Class และกำหนด Primary Key เหมือนกัน โดยในแต่ละตารางอาจมีแอตทริบิวต์ที่แตกต่างกันได้

สิ่งที่ต้องพิจารณาระหว่างการแปลง คือ จะต้องกำหนดว่า (1) จะใช้ Oid ที่สร้างโดยระบบ หรือจะเลือกใช้แอตทริบิวต์จริงในแต่ละตาราง ซึ่งจะต้องแน่ใจว่าข้อมูลจะไม่ซ้ำซ้อนกัน (2) จะต้องกำหนดว่า แอตทริบิวต์ใดบ้างที่จะให้ปรากฏในแต่ละตาราง

2. Uploading Process from Object Oriented Database to ASCII File เป็นการอ่านข้อมูลจาก Object Database Schema และแปลงข้อมูลให้อยู่ในรูปของ Sequential File ซึ่งจะมีชุดคำสั่ง DDL, DML ที่จะใช้ในการนำข้อมูลสู่ระบบ และในระหว่างการทำงาน อาจเกิดกรณีที่ต้องตัดสินใจในการเลือก ซึ่งแบ่งเป็น 4 กรณี ได้แก่

   (1) ถ้าทุกออบเจคมี Attribute Key และ Oid สามารถเลือกตัวใดตัวหนึ่งเป็น Primary Key ได้

   (2) ถ้ามีเพียงบางออบเจคที่มี Attribute Key แต่มี Oid ให้เลือก Oid เป็น Primary Key

   (3) ถ้าทุกออบเจคมี Attribute Key แต่มีเพียงบาง Oid ให้ใช้ Attribute Key เป็น Primary Key

   (4) ถ้ามีเพียงบางออบเจคที่มี Attribute Key และมี OID เพียงบางตัว ให้สร้าง Primary Key ขึ้นใหม่ หรือให้รวม Attribute Key เข้าไว้ในทุกคลาส

3. Data Uploading Process from ASCII file to Relational Database เป็นขั้นตอนในการอ่าน Sequential File เพื่อนำข้อมูลเข้าสู่ตาราง

Jens Jahnke และคณะ [9] กล่าวถึง การแปลงข้อมูลและ Corresponding Schema จาก Relational Database ไปสู่ Object Oriented Database โดยประกอบด้วยกระบวนการหลัก 3 ส่วน ได้แก่

(1) Schema Migration Process เป็นการแปลง Schema ให้เหมือนกัน

(2) Data Migration Process เป็นการแปลงข้อมูล

(3) Application Migration Process เป็นการสร้างโปรแกรมใหม่ สำหรับแต่ละส่วนที่เคยใช้ในระบบเดิม

โดยพบว่างานวิจัยต่าง ๆ ที่ในอดีต มีการสร้างเครื่องมือหลายประเภท แต่เมื่อทำการแปลงระบบเสร็จสมบูรณ์แล้ว มักจะพบความผิดพลาดในการแปลงและถ่ายโอนระบบ ดังนั้น เพื่อให้คงไว้ซึ่งความถูกต้องของระบบให้มากที่สุด จึงได้ทดลองสร้างเครื่องมือสำหรับแปลงแบบ Interactive ขึ้น โดยจะอนุญาตให้มีการแก้ไข Schemas ได้ เพื่อป้องกันการสูญหายของข้อมูล จากนั้นเครื่องมือจะทำการรวมโครงสร้างให้เรียบร้อย

Meier, A. Dippold และคณะ [16] ทำการวิจัยที่เกี่ยวกับการพัฒนาซอฟต์แวร์ที่ทำหน้าที่แปลงและถ่ายโอนข้อมูลจาก Hierarchical IMS ไปสู่ระบบ Relational IBM DB2 ที่ Swiss Bank โดยคำนึงถึงปัจจัยสำคัญ ได้แก่ (1) Language Interfaces ซึ่งสนับสนุนทั้ง ภาษาชนิด Procedural และ Record-by-Record เพื่อติดต่อกับ Relational Database หรือ ภาษา SQL เพื่อติดต่อกับ Hierarchical Database (2) Source-code conversion ได้แก่ การแปลงชุดคำสั่งใน Database Application ให้เป็นชุดภาษา SQL แทน Hierarchical แบบเดิม (3) Data propagation ได้แก่ ชุดคำสั่งที่ช่วยส่งข้อมูลที่มีการเปลี่ยนแปลงไปยังระบบฐานข้อมูล หากระบบไม่ทำงาน ตัว Propagator จะเปลี่ยนข้อมูลกลับ จึงช่วยให้ข้อมูลมีความถูกต้องขณะทำงาน

Hsieh, S-Y และคณะ [6] แสดงถึงเป้าหมายหลักในการแปลง OODB Schema ให้เป็น Relational Schema จะพิจารณาถึงความแตกต่างระดับพื้นฐาน ที่อยู่บน Database Model ได้แก่ (1) Schema Transformation (2) Query Transformation

1. Schema Transformation from OODM to RDM

การแปลง Schema จาก OODM เป็น RDM จะพบลักษณะของแอตทริบิวต์ชนิดต่าง ๆ ดังนี้

1. Non-Reference Attribute ได้แก่ แอตทริบิวต์ที่มีโดเมนเป็น Primitive Type เช่น Integer, Character
2. Reference Attribute ได้แก่ แอตทริบิวต์ที่มีโดเมนเป็น User-Defined-Class และใช้ Oid ในการอ้างถึงออบเจคที่เกี่ยวข้อง
3. Non-Set Attribute ได้แก่ แอตทริบิวต์ที่มีโดเมนได้ทั้ง Primitive และ User-Defined-Class แต่จะมีค่าเพียงค่าเดียว (Single Instance)
4. Set Attribute ได้แก่ แอตทริบิวต์ที่มีโดเมนได้ทั้ง Primitive และ User-Defined-Class และอนุญาตให้จัดเก็บข้อมูลได้หลายค่าในแอตทริบิวต์ (Set of Instances)

กระบวนการแปลงสกีมา มีขั้นตอนดังต่อไปนี้

- กรณี CLASS

  แปลงคลาสให้เป็นรีเลชั่น โดย

  1. แต่ละคลาส แปลงให้เป็นรีเลชั่น
  2. ใช้ค่า Oid ของแต่ละออบเจคในคลาส จะใช้เป็น Primary Key ในรีเลชั่น
  3. แต่ละอินสแตนท์ในคลาส มีสถานะเป็นทูเปิลในรีเลชั่น

- กรณี Non-Reference Attributes

  แปลง Non-Reference Attributes มี 2 ลักษณะ ได้แก่

  1. กรณี Non-Reference Non-Set Attribute ให้แปลงให้อยู่ในรูปแอตทริบิวต์ในรีเลชั่น
  2. กรณี Non-Reference Set Attribute จะต้องสร้างเป็นรีเลชั่นใหม่ เรียกว่า Virtual-Relation เพื่อใช้เก็บค่า Non-Reference-Set Attribute โดยในรีเลชั่นที่เกิดขึ้นนี้ จะมีสองแอตทริบิวต์ คือ Oid และ Non-Reference-Set Attribute โดยทั้งสองแอตทริบิวต์นี้จะใช้เป็น Primary Key ในรีเลชั่น

- กรณี Class Hierarchy ซึ่งเป็นลักษณะ Inheritance โดยให้แปลง Class Hierarchy โดยให้สร้างรีเลชั่นขึ้นใหม่สำหรับแต่ละ Sub-Class และในแต่ละรีเลชั่นที่สร้างขึ้นนี้Primary Key จะได้จาก การนำแอตทริบิวต์ที่เป็น Primary Key ของตารางที่เป็น Super-Class ทั้งหมดมา (ถ้าเป็นกรณี Multiple Inheritance เมื่อแปลงมาแล้ว จะได้ Primary Key ที่เป็น Combine Key ซึ่ง Primary Key ที่ดึงมาจะใช้เป็น Foreign Key ในการอ้าง)
- กรณี Class Composition Hierarchy ซึ่งเป็นลักษณะความสัมพันธ์แบบ Association โดยแปลง Class Composition Hierarchy (Associative Relationship) จะมีสองลักษณะ ได้แก่

  1. Reference-Non-Set Attribute
     - จะต้องแปลงคลาสที่เกี่ยวข้องให้เป็นรีเลชั่นก่อน คือ ถ้าคลาส C1 มีการอ้างถึงคลาส C2 เมื่อแปลงเป็นรีเลชั่นแล้ว จะได้ R1 และ R2 ตามลำดับ จากนั้น ให้เพิ่มแอตทริบิวต์บนรีเลชั่น R1ที่เป็นตัวเก็บค่า OID ของอีกรีเลชั่น R2 เข้าไป โดยแอตทริบิวต์นี้ จะเป็น Foreign Key
  2. Reference-Set Attribute
     - ถ้าในคลาส C1 มีการจัดเก็บค่าที่เป็น Reference ที่จะอ้างไปยังชุดของออบเจคในคลาส C2 จะต้องสร้างตารางใหม่ขึ้นมา ที่เรียกว่า Virtual-Relation ซึ่งในรีเลชั่นใหม่ที่สร้างนี้ จะต้องกำหนด Primary Key ด้วย ซึ่งจะได้จากการนำ Primary Key ของทั้งสองรีเลชั่นมารวมกัน (Combine-Key) และ Primary Key แต่ละตัวที่ดึงมาจากแต่ละรีเลชั่น จะเป็น Foreign Key

- การตั้งชื่อ Relations และ Attributes ใช้หลักเกณฑ์ ดังต่อไปนี้

1. กรณี Non-Virtual Relation
   - ชื่อรีเลชั่น ให้ใช้ชื่อเดิมของคลาสเดิม
   - กำหนด Primary Key ขึ้นโดยใช้ ชื่อคลาส + "Oid"
2. กรณี Non-Reference, Non-Set Attribute ในแต่ละรีเลชั่น
   - แอตทริบิวต์ ให้ใช้ชื่อเดียวกับชื่อแอตทริบิวต์ในคลาสเดิม
3. กรณี Reference Non-Set Attribute ในแต่ละรีเลชั่น
   - ให้ใช้ Primary Key เป็นตัวชี้ เพื่ออ้างถึงรีเลชั่นที่เกี่ยวข้อง
4. กรณี Sub-Class Relation ที่สร้างขึ้นตามลักษณะ Class Hierarchy
   - ให้ตั้งชื่อแอตทริบิวต์ โดยใช้ชื่อ Super-Class + "Oid"
5. กรณีที่เป็น Virtual Relations ที่สร้างขึ้นโดย Non-Reference-Set Attributes
   - ให้ตั้งชื่อรีเลชั่นโดย ชื่อรีเลชั่น + ชื่อของ Non-Reference-Set Attribute
   - กำหนด Primary Key ใน Virtual Relation โดยได้จาก Primary Key ของรีเลชั่นที่เกี่ยวข้อง รวมกับแอตทริบิวต์ จาก Non-Reference-Set Attribute
6. กรณีที่เป็น Virtual Relations ที่สร้างขึ้นโดย Reference-Set Attributes
   - กำหนด Primary Key โดยนำมาจาก Primary Key ของรีเลชั่นที่เกี่ยวข้อง
   - ตั้งชื่อรีเลชั่นโดย ชื่อคลาสเดิม + ชื่อของ Reference-Set Attribute

2. Query Translation เป็นการแปล Relational SQL Query ให้อยู่ในรูปแบบที่เหมาะสำหรับ OODB โดยจากงานวิจัย มีการเลือกภาษาที่จะใช้ 2 ภาษา คือ Relational SQL, Object SQL ซึ่งมีความคล้ายคลึงกันมาก

   1. General Object SQL Select command

      โดย รูปแบบคำสั่ง จะใช้

      Select AS กลุ่มของแอตทริบิวต์ From C คือ คลาสที่ต้องการดึงข้อมูล

      Where PS คือ เงื่อนไข

      โดยคำสั่งจะทำงานโดยเมดธอดบนพื้นฐานของ OODB แต่จะระบุคลาสได้เพียงคลาสเดียว และคำสั่งจะพิจารณาตามเงื่อนไขที่ระบุไว้ในแต่ละออบเจค จากนั้นจึงส่งชุดของออบเจคที่ต้องการกลับมา ภายใต้ Where สามารถระบุ Reference Paths ของ Referenced Attributes ได้ ในรูป C.r1.r2...rm.Attr ได้

   2. Translation Issues

      ก่อนการแปลคิวรี จะต้องพิจารณาถึง

      1. Target Class(es) เนื่องจากรูปแบบการใช้ในลักษณะ "C.r1.r2...rm.Attr" ทำให้ต้องชี้ให้ถูกต้องให้ได้ว่า Root ของ Reference path อยู่ที่ใด และเพื่อให้คงไว้ซึ่ง Referential

Integrity ในงานวิจัยจึงสร้าง Information Base ขึ้นเพื่อเก็บ Reference Table, Reference Closures, Integration Information (Information Base = Class(es) of Reference Attribute(s) + Reference Attribute(s) + Referred Class) ซึ่งค่าเหล่านี้ ได้มาจาก Object-Oriented Schema ของระบบเดิม

2. ต้องชี้ให้ได้ถึง Reference path ของแต่ละแอตทริบิวต์

3. Translation Procedure
   1. หาค่า Minimum Cover Set (MCS) ที่ได้จากการคำนวณจากข้อมูลในข้อ 2
   2. พิจารณาแต่ละคลาสใน MCS และสร้างเป็น Object SQL Template ถ้ามีคลาสมากกว่าหนึ่งคลาส ให้แปลงให้เป็นอีก Object SQL Select Query
   3. ตรวจดูว่าแต่ละแอตทริบิวต์ใน Relational SQL Select clause และจัดให้อยู่ในรูป Object SQL Select clause ซึ่งจะต้องมีการเพิ่ม Reference paths รวมไว้ด้วย
   4. พิจารณาส่วนของ Where clause
   5. ในกรณีที่มี Object SQL Queries หลายตัว จะต้องนำมาประมวลผลร่วมกันเพื่อให้ได้ผลลัพธ์สุดท้ายที่ถูกต้อง

สำหรับงานวิจัยเกี่ยวกับความเหมาะสมในการเปลี่ยนระบบฐานข้อมูล ได้แก่

Patrick O' Neil และ Elizabeth O'Neil. [17] อธิบายถึง ระบบฐานข้อมูล ควรมีลักษณะที่เรียกว่า Object SQL Capabilities ซึ่งเป็นคุณสมบัติสำคัญที่เหมาะ ที่จะนำระบบฐานข้อมูลเชิงวัตถุสัมพันธ์มาช่วยงาน คือ การสนับสนุนข้อมูลที่มีความซับซ้อน (A Composite Structured Type) รวมถึงการสนับสนุนข้อมูลแบบคอลเลคชั่น (Collection Type) ได้แก่ การที่อนุญาตให้เก็บข้อมูลลงในแอตทริบิวต์ในแต่ละแถว (Row) ได้หลายค่า หรือเป็นเซต หรือสามารถแฮนเดิลข้อมูลเป็นตารางได้ (Table Nesting) นอกจากนี้ ในการออกแบบและพัฒนา ระบบฐานข้อมูลเชิงวัตถุสัมพันธ์ อนุญาตให้ฝืนกฎการออกแบบ โดยยอมให้มีข้อมูลที่เป็นโครงสร้างซ้อนกันภายในได้ (Breaking the First Normalization Rule) และในส่วนของเมดธอด (Method/UDF) ถ้าเป็นระบบฐานข้อมูลเชิงวัตถุ ข้อมูลจะมีลักษณะเป็น Private การเรียกใช้งานจะต้องผ่านเมดธอดหรือฟังก์ชั่นเท่านั้น แต่ถ้าเป็นระบบฐานข้อมูลเชิงวัตถุสัมพันธ์ ข้อมูลจะมีลักษณะเป็น Public คือ จะอนุญาตให้เข้าถึงข้อมูลได้โดยตรงผ่าน SQL Statements หรือ จะใช้ผ่านเมดธอดหรือฟังก์ชั่นก็ได้ ซึ่ง UDFs จะไม่ผูกติดในตารางหรือคลาส แต่จะอนุญาตให้นำ UDFs ที่สร้างขึ้น ไปผูกติดหรือเรียกใช้ได้กับคำสั่งต่าง ๆ เพื่อเข้าถึงข้อมูลแทน ซึ่งระบบฐานข้อมูลเชิงวัตถุ จะแตกต่างจากระบบฐานข้อมูลเชิงวัตถุสัมพันธ์ คือ ระบบฐานข้อมูลเชิงวัตถุสัมพันธ์ จะมีลักษณะ Encapsulation ไม่สมบูรณ์ ในส่วนของคุณสมบัติต่างอยู่ภายใต้มาตรฐาน SQL-99

V.Srinivasan และ D. T. Chang. [22] กล่าวถึง ข้อพิจารณาในเรื่องความเหมาะสมในการย้ายแอพพลิเคชั่นจาก Object Database มาสู่ Object Relational Database มีหลายประเด็น เช่น ในประเด็นของรูปแบบข้อมูล (Data Model) ORDBMS จะสนับสนุน Complex objects (objects containing non-first-normal form data) ผ่าน Relational Data Model และในส่วนที่เป็น Composite Objects จะสนับสนุนผ่าน ADTs และคอลเลคชั่น ซึ่งจะอยู่ในรูปของตาราง

ในด้านของ Data Sharing ทั้ง ORDBMS และ OODBMS ต่างสนับสนุนคุณสมบัติ ACID Transactions แต่ ORDBMS จะสนับสนุนการทำ Crash recovery รวมถึงคุณสมบัติด้าน Security, views, Integrity Constraints, Triggers ได้ดีกว่า ODBMS ซึ่งแต่ละผลิตภัณฑ์ต่างมีความแตกต่างกัน ในด้านการเข้าถึงข้อมูล ORDBMS จะมีภาษาที่มีความสามารถและมีประสิทธิภาพในการทำ Ad hoc Query ซึ่งผู้ใช้คุ้นเคยหรือทำความเข้าใจได้ง่าย

จากกรณีดังกล่าว จะเห็นได้ว่า ORDBMS จะมีคุณสมบัติที่จำเป็นหลายด้าน ซึ่ง OODBMS ไม่มีหรือค่อนข้างจำกัดกว่า นอกจากนี้ ORDBMS ผู้ใช้สามารถพัฒนาและเรียนรู้บนพื้นฐานของระบบฐานข้อมูลเชิงสัมพันธ์ได้ง่ายกว่า

รศ.ดร.ศุภมิตร จิตตะยโศธร [19] กล่าวถึงปัญหาของ ER Model คือ (1) ขาดการนำเสนอความสัมพันธ์ระดับแอตทริบิวต์ (2) สามารถมองได้เพียงระดับ Entity Type เท่านั้น (3) นำเสนอได้เพียงระดับ 1NF (4) ไม่สามารถแสดง Function Dependency (FD) และ Multi Value Dependency (MVD) ได้ ปัญหาต่าง ๆ เหล่านี้ แสดงให้เห็นถึงความไม่สมบูรณ์ของโมเดล นอกจากนี้ เมื่อมีการแปลง ER ไปเป็น Logical Model อาจทำให้ความสมบูรณ์ของข้อมูลบางส่วนขาดหายไป เช่น Total Participation แสดงบน ER แต่ไม่ได้ระบุว่าเป็นของแอททริบิวส์ใด เมื่อแปลงไปเป็น Logical Model, Physical Model ลักษณะ Total Participation ได้สูญหายไป เป็นต้น

หลักการแปลง ER Model เป็น Relational Model มีขั้นตอนดังต่อไปนี้

1. แปลง Regular Entity Type ให้เป็นตาราง 1 ตาราง โดยให้นำแอตทริบิวต์ที่ติดอยู่ทั้งหมดไปด้วย ยกเว้นแอตทริบิวต์ที่เป็น Multi Values จากนั้นให้กำหนด Identifier ที่เหมาะสมเป็น Primary Key
2. กรณี Weak Entity Type ให้แปลงเป็น 1 ตาราง และกำหนด Identifier ให้เหมาะสม จากนั้นให้นำ Primary Key ที่อยู่ในด้าน Parent มารวมกับ Identifier ที่กำหนดไว้ก่อน รวมเป็น Combined Key ในตารางที่เกิดขึ้น
3. พิจารณาความสัมพันธ์ 1:1 ไม่จำเป็นต้องสร้างเป็นตารางใหม่ สามารถใช้ Primary Key/ Foreign Key ในการอ้างถึงกันได้ และหากมีด้านใดเป็น Total Participation ให้ยึดตารางในด้าน Total Participation เป็นหลักและนำ Primary Key ของอีกตารางมาไว้ในฝั่ง Total ด้วย

4. พิจารณาความสัมพันธ์ 1:m ไม่จำเป็นต้องสร้างตารางใหม่อีก แต่ให้ยก Primary Key ในฝั่ง One มาไว้เป็น Foreign Key ในฝั่ง Many และในกรณีที่เป็นความสัมพันธ์แบบวนกลับ (Recursive) ให้สร้างแอตทริบิวต์ใหม่ขึ้นอีกหนึ่งตัว เพื่อแยกความแตกต่างให้ชัดเจน
5. พิจารณาความสัมพันธ์แบบ M:N จะต้องสร้างเป็นตารางใหม่ โดยให้นำ Primary Key ของทั้งสองตารางมารวมกัน เป็น Primary Key ในตารางใหม่ และถ้ามีแอตทริบิวต์อื่น ๆ ก็ให้นำมารวมไว้ในตารางใหม่ด้วย
6. พิจารณากรณี Multi Values Attributes ให้ยกแอตทริบิวต์ดังกล่าว ไปตั้งเป็นตารางใหม่ โดยตารางใหม่ที่เกิดขึ้น จะประกอบด้วย Multi Value Attribute และแอตทริบิวต์ที่เป็น Identifier ของ Entity Type นั้น
7. กรณีที่เป็นความสัมพันธ์แบบ N-ary จะต้องสร้างเป็นตารางใหม่ และนำ Identifier ของแต่ละ Entity Type มารวมกันในตารางใหม่

จากกระบวนการแปลงทั้ง 7-ขั้นตอน จะได้ตารางที่อยู่ในรูป First Normal Form เท่านั้น จำเป็นต้องทำให้ครบ 5NF จึงจะเสร็จครบกระบวนการ

1.6 ขอบเขตการวิจัย

ในงานวิจัยทดลองใช้ซอฟแวร์ InterSystems Caché ซึ่งเป็นระบบฐานข้อมูลเชิงวัตถุและ IBM DB2 UDB ซึ่งเป็นระบบฐานข้อมูลเชิงวัตถุสัมพันธ์ เป็นเครื่องมือทดสอบการแปลงระบบโดยให้คงความถูกต้องของข้อมูลในโมเดลต้นแบบ เพื่อเป็นฐานสู่การโอนย้ายข้อมูลไปสู่โมเดลใหม่ได้ ทั้งนี้ ในส่วนที่เป็นเมดธอด จะเป็นการเก็บชุดคำสั่งและคำอธิบาย (Method Signature) เพื่อให้ผู้ใช้สามารถนำไปแก้ไขให้เหมาะสมกับงานได้ต่อไป

1.7 ขั้นตอนการศึกษา

ในงานวิจัยนี้ ใช้ Inter System Caché เป็นระบบฐานข้อมูลเชิงวัตถุ และใช้ IBM DB2 UDB ซึ่งเป็นระบบฐานข้อมูลเชิงวัตถุสัมพันธ์ ของบริษัท IBM เป็นเครื่องมือทดสอบการแปลงโดยการศึกษา ประกอบด้วยขั้นตอนหลัก ได้แก่

1. การวิเคราะห์ลักษณะโครงสร้างข้อมูลและรูปแบบการจัดเก็บของระบบฐานข้อมูลเชิงวัตถุ
2. การวิเคราะห์ลักษณะโครงสร้างข้อมูลและรูปแบบการจัดเก็บของระบบฐานข้อมูลเชิงวัตถุสัมพันธ์

3. ทดลองอ่านโครงสร้างข้อมูล (Database Schema and Definitions) และแปลงให้อยู่ในรูปคำสั่งที่ทำงานบนระบบฐานข้อมูลเชิงวัตถุสัมพันธ์
4. พัฒนาเครื่องมือ สำหรับการช่วยปรับเปลี่ยนระบบงาน

1.8 สรุปแนวคิดการวิจัย

แนวคิดจากงานวิจัยนี้ กล่าวถึงเทคนิคการปรับเปลี่ยนโมเดลเชิงวัตถุ มาสู่โมเดลเชิงวัตถุสัมพันธ์ โดยให้คงความสมบูรณ์ของข้อมูลเดิมไว้ และถ่ายโอนข้อมูลมาสู่ระบบฐานข้อมูลโมเดลใหม่ โดยแนวคิดนี้ จะสามารถปรับปรุงงานวิจัยให้สมบูรณ์ขึ้นในลำดับต่อไป คือ เทคนิควิธีการโอนข้อมูลจากระบบฐานข้อมูลเชิงวัตถุอื่น ๆ และระบบฐานข้อมูลเชิงสัมพันธ์อื่น มาสู่ระบบฐานข้อมูลโมเดลเชิงวัตถุสัมพันธ์ (Object-Relational Database Model) เพื่อเป็นการนำไปสู่การพัฒนาระบบคลังข้อมูล (Information Warehouse) ต่อไป

บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

2.1 โมเดลข้อมูลและการใช้เทคโนโลยีเชิงวัตถุในระบบฐานข้อมูล

โมเดลข้อมูลและเทคโนโลยีเชิงวัตถุในระบบฐานข้อมูล โดย รศ.ดร.ศุภมิตร จิตตะยโศธร [20] อธิบายได้ว่า ถ้าลองพิจารณาดู อาจมีข้อสงสัยว่า ทำไมคนจึงนิยมใช้ Relational และทำไม Hierarchical, Network Database ถึงได้รับความเสื่อมถอยไป คำตอบ คือ Relational มีลักษณะโครงสร้างของข้อมูลที่ดูและเข้าใจง่ายกว่าโมเดลอื่น ๆ ลักษณะ คือ เป็น Simple Data Structure นอกจากนี้ คือ ภาษาที่ใช้มีความยืดหยุ่น คือ ภาษา SQL นั่นคือ หากต้องการค้นหาอะไรในตาราง ก็สามารถค้นหาได้ง่าย แต่อย่างไรก็ดี ก็ขึ้นอยู่กับฝีมือและความชำนาญของผู้ใช้ด้วย แต่ปัจจุบัน ลักษณะความเป็น Simple ใน Relational จะมีลักษณะเป็น too simple มากเกินไป หากมองภาพรวมในทางวิชาการ จะพบว่า นักวิทยาศาสตร์เมื่อจะคิดอะไร จะมีแนวคิด 2 แนวทาง ได้แก่ แนวคิดตามธรรมชาติ ที่บอกว่าทำอย่างไร จะพยายามลอกเลียนแบบ และแนวคิดตามหลักคณิตศาสตร์ คือการนำคณิตศาสตร์มาใช้

สำหรับธรรมชาติของ โมเดล Hierarchical และ Network คือ จะใช้ Pointer มาชี้ที่เป็น Tree แต่ Codd ซึ่งเป็นนักคณิตศาสตร์ ผู้คิดค้น Relational Model ใช้หลักคณิตศาสตร์ ต้องอธิบายได้ด้วยคณิตศาสตร์ รับ ซึ่งแนวคิดดังกล่าว จะเหมือนนักฟิสิกส์ หรือวิศวกรไฟฟ้า ที่พยายามเอาคณิตศาสตร์มาอธิบายไฟฟ้า หรืออธิบายอุบัติการณ์ทางฟิสิกส์ต่าง ๆ Codd ได้นิยาม Database ให้มีลักษณะเป็นตารางแบน ๆ ไม่มีอาเรย์ ทั้งนี้ เพราะตารางเป็นตัวแทนของ Subset of Cartesian products of domain และจากความเป็นตารางแบน ๆ แบบนี้ อาจไม่คล่องตัวกับบางแอพพลิเคชั่น นอกจากนี้ data type ของ DBMSs จะเป็น Simple Data Type คือ Character, Integer, Real เป็นต้น ซึ่งในปัจจุบัน ลักษณะข้อมูลมีการเปลี่ยนแปลงไป Data Type จะไม่ Simple เหมือนก่อน แต่มีลักษณะ Complex data เช่น Multimedia, Array ทำให้นักวิทยาศาสตร์รุ่นหลัง เกิดแนวคิดที่จะมองข้อมูลเป็นธรรมชาติก่อน แล้วจึงนำแนวคิดทางคณิตศาสตร์ ไปจับอีกครั้ง

จากเหตุผลดังกล่าว ทำให้ Relational ทำไม่ได้ ต้องแยกเป็นอีก 2 หรือ 3 ตารางแล้วแต่กรณี ซึ่งเป็นจุดที่นักวิทยาศาสตร์หลัง Codd อยากกลับมาใช้วิธีการตามธรรมชาติ คือ เอาธรรมชาติมาก่อน มาทำเป็น Object เสร็จแล้วมาหาคณิตศาสตร์ มาจับ Object Model เพื่อประโยชน์ในการทำ Application ในบางลักษณะ

ในปัจจุบัน ระบบฐานข้อมูลที่ใช้เทคโนโลยีเชิงวัตถุเข้ามาช่วย มี 2-ชนิด ได้แก่ (1) Object Oriented Database พัฒนาขึ้นใหม่ โดยใช้คุณสมบัติและเทคโนโลยีเชิงวัตถุทั้งระบบ และพัฒนาตามมาตรฐาน ODMG (2) Object Relational Database พัฒนาจาก RDBMS เดิม โดยการเพิ่มคุณสมบัติเชิงวัตถุรวมเข้าไว้ และใช้มาตรฐาน SQL3 โดยคุณสมบัติเทคโนโลยีเชิงวัตถุ ที่มีอยู่ในระบบฐานข้อมูลทั้งสองแบบ มีความเหมือนกันในการนำทฤษฎีเชิงวัตถุมาช่วย หากแต่จะมีความแตกต่างกัน ในกระบวนการทำงาน และวิธีการเข้าถึงข้อมูล รวมถึงข้อกำหนดบางประการ

ใน Object Model ซึ่งมีความสัมพันธ์แบบ 1:m คือ มีออบเจคคลาส Department, Employee ความจริงรูปนี้ จะต้องมี Attribute หนึ่งของ Department ที่มี Content Type เป็น Pointer ที่ชี้ไปมาระหว่างคลาส เรียกว่า Inverse Relationship โดยไม่ต้องมี PK, FK ชี้ไปมาระหว่างคลาส

ในส่วนที่เป็นลักษณะ M:N ซึ่งมีลักษณะคล้ายกับแบบแรก คือมี Reference Pointer ชี้ไปที่ Object Class Hour ที่อยู่ตรงกลาง ข้อดี จะเห็นว่า ในโมเดลนี้ จะไม่ต้องใช้ PK,FK ซึ่งดูแล้วจะเป็นธรรมชาติ

หากพิจารณาถึง PK, FK ที่นำมาใช้ใน Relational จะเป็น External Identifier ซึ่งมนุษย์สร้างขึ้นมา และตามทฤษฎีค่าของ PK จะเปลี่ยนไม่ได้ เพราะถ้าเปลี่ยนจะไม่ทราบว่าเป็นตัวเก่า แต่ในทางปฏิบัติก็มีการเปลี่ยนแปลงบ่อย เช่น นักศึกษาคนหนึ่ง เข้าศึกษาในสถานศึกษา จะได้ Student NO สมมติถ้ามหาวิทยาลัยนำรหัสคณะ และรหัสภาคไปใส่ในรหัสนักศึกษาด้วย เมื่อนักศึกษาย้ายคณะ รหัสก็จะเปลี่ยนแปลง ซึ่งจะต่างทฤษฎีที่บอกไว้ นั่นคือ ธรรมชาติของแอพพลิเคชั่น ทำให้ต้องเปลี่ยน

ถ้านักศึกษาคนเดียวกัน ขึ้นระดับปริญญาโท แล้วเรียนต่อ รหัสก็จะต้องเปลี่ยนอีก และถ้าเรียนต่อระดับปริญญาเอก ที่สถานศึกษาเดิมอีก รหัสก็จะต้องเปลี่ยนแปลงอีก เป็นต้น ดังนั้น นักศึกษา 1 คน อาจมีหลายรหัสในมหาวิทยาลัย ทั้ง ๆ ที่เป็นเรคคอร์ดของคน ๆ เดียวกัน

จุดแข็งของการใช้คุณสมบัติ Object Oriented ใน Database คือ จะไม่ขึ้นกับ External Identifier แต่จะใช้ Pointer คือ Object ID (OIDs) ในการอ้างแทน ซึ่ง OID จะสร้างโดย DBMS โดยมีลักษณะ unique และไม่มีการนำค่ากลับมาใช้ใหม่ เมื่อมีการลบออก ซึ่งตรงนี้ จะแตกต่างจากโมเดลอื่น ๆ เพราะ OID จะได้จาก DBMS โดยเฉพาะ ดังนั้น จากตัวอย่างนักศึกษา ถ้ามีการเปลี่ยนแปลง ID อย่างไร DBMS ก็จะทราบว่าเป็นคนเก่า รหัสนักศึกษา จะเป็นเพียง Description ซึ่งจะใช้ก็ได้ ไม่ใช้ก็ได้

ใน Object Model จะมีการนำเมดธอด ไปรวมไว้ในแต่ละคลาส จะเห็นว่า มีการนำเมดธอดเข้ามาเกี่ยวข้อง โดยทฤษฎีเมดธอด จะเป็น interface เดียวที่ใช้ในการติดต่อกับออบเจคได้ ซึ่งภาษาที่ใช้เขียนเมดธอด ก็จะแตกต่างกันออกไป คือเป็นการเขียนเมดธอด หรือฟังก์ชั่น เพื่อให้ทำงานตามต้องการ

จากนั้น โปรแกรมเมอร์ จะเรียกใช้ข้อมูลโดยผ่านฟังก์ชั่นหรือเมดธอดนี้เท่านั้น โดยไม่จำเป็นต้องทราบว่า โครงสร้างข้อมูลมีอะไรหรือเป็นอย่างไร

จากตัวอย่าง จะเห็นว่า มี Employee 1 แอททริบิวซ์ ที่เก็บ Pointer ที่ชี้ไปยัง Employee ถ้ามองเป็น Relational จะเป็น Emp# ซึ่งเป็นการใช้ External Identifies แทน

การมองแบบธรรมชาติ การมองแบบธรรมชาติ เป็นลักษณะการมองอย่างที่มนุษย์คิด นั่นคือ คิดอย่างไร ก็เห็นอย่างนั้น สิ่งที่มนุษย์มีในธรรมชาติ คือ Sub Type โดยจะเห็นได้ว่า Object มี Super Class, Sub Class ซึ่ง คือ Graduate, Undergrads ซึ่งในความเป็นจริง ต่างก็จะมีแอททริบิวซ์ภายในที่เพิ่มขึ้นเฉพาะพอสมควร เช่น ทาง Graduate จะต้องมี Topic Research , Publication ซึ่งไม่ใช่สิ่งที่ Undergrads จะต้องมี แต่ Undergrads อาจจะมี Activity, Sport เพิ่มเข้ามา ถ้าพิจารณาต่อไป สามารถแบ่งเป็น Sub Type ลงไปได้อีก เช่น Graduate จะแยกเป็น Master, Doctoral ส่วน Undergrad จะแยกเป็นภาคปกติและภาคค่ำ ซึ่งเป็นส่วนพิเศษที่เพิ่มขึ้นมา ดูแล้วเป็นธรรมชาติ นอกจากนี้ ความสัมพันธ์ของ Relationship อาจมีในระดับ Super Class เช่น นักศึกษาตรี โท เอก ต้องสังกัดภาควิชา ซึ่งจะติดต่อกับ Student ก็เป็นได้ ถ้าแปลงโมเดลนี้ ให้อยู่ในรูป Relational จะแปลงได้เป็น 3 ตาราง คือ Student, Graduate, Undergrad หรือ อาจนำทั้งสามตารางมาสร้างรวมกันเป็นตารางเดียวก็ได้ แต่จะเสียความเป็น Semantic ไป

ถ้าพิจารณาต่อ จะเห็นได้ว่าเมดธอดที่ใช้ชื่อ finish() เหมือนกัน แต่วิธีการภายในจะแตกต่างกัน เพราะ Undergrad เงื่อนไขจะจบการศึกษาได้ จะต้องได้ผลการเรียนไม่น้อยกว่า 2.00 ส่วน Graduate ต้องได้ผลการเรียน 3.00 ขึ้นไป แต่เวลาที่เรียกใช้เมดธอด จะเรียก finish() เหมือนกัน แต่ถ้าเป็น Undergrad จะชี้ไปใช้ finish() ของ Undergrad ซึ่งเรียกคุณสมบัติดังกล่าวนี้ว่า Polymorphism

ข้อดีอีกประการ คือ Encapsulation กล่าวคือ เป็นการซ่อนรายละเอียดของแอททริบิวซ์ โดยที่โปรแกรมเมอร์ หรือผู้ใช้ ไม่จำเป็นต้องทราบ รายละเอียดภายใน คือถือว่าเป็น Private คือเรียกใช้ผ่าน finish() ได้เลย กล่าวคือ เวลาจะใช้งาน เพียงเรียกใช้ผ่านเมดธอดเท่านั้น หรือ ถ้าจะเปรียบเทียบกับการขับรถยนต์ รถก็จะมีอินเตอร์เฟสต่าง ๆ เช่น พวงมาลัย, เกียร์, เบรก ให้ใช้ โดยไม่จำเป็นต้องทราบกลไกการทำงานภายในรถว่าเป็นอย่างไร เพียงเรียกใช้ผ่านอินเตอร์เฟสในรถ ที่เตรียมไว้ให้เท่านั้น

กรณีคลาส Patient โดยมีแอททริบิวซ์ Name จะสังเกตได้ว่า จะไม่มี PatientNo ซึ่งปกติใน Relational จะใช้เป็น PK แต่ใน Object Model ไม่จำเป็นต้องมี หรืออาจจะมีก็ได้ ซึ่งถ้ามี ก็จะใช้เป็นเพียง Description เท่านั้น แต่การอ้างอิงจริง ๆ จะอ้างผ่าน OIDs จากรูปแสดง Simple Attribute

การใช้ Pointer แทน Primary Key จะพบว่าคลาส Patient มีแอททริบิวซ์ ชื่อ PrimaryDoc ที่เก็บ Pointer ที่อ้างอิงไปที่คลาส Doctor ซึ่งบอกว่า คนไข้รายนี้ มีคุณหมอท่านใด เป็นผู้ดูแล ซึ่งจะเห็นได้ว่า ไม่ต้องใช้ PK, FK ซึ่งเป็น External Identifiers

การใช้ Embedded Object ถ้าดูจาก ER Model จะเรียกลักษณะนี้ว่าเป็น Composit Attribute แต่ใน Object Model จะเรียกว่า Embedded Object ซึ่งสามารถทำเป็น แอททริบิวซ์หนึ่ง หรืออาจทำเป็น Collection ก็ได้ ซึ่ง Collection หรืออีกนัยหนึ่ง คือ Array นั่นเอง เช่น คนไข้คนหนึ่ง อาจมีได้หลายบ้าน สามารถทำได้โดยกำหนดให้แอททริบิวซ์ Address เป็น Collection ก็ได้

กรณีของ Relational โดยทฤษฎีทางคณิตศาสตร์ จะบังคับให้อยู่ในรูปตาราง 2 มิติ จึงไม่สามารถมีอาเรย์ได้ ดังนั้น ถ้ามีหลายบ้าน จะต้องแยกออกเป็นตาราง และจะไม่สามารถอ้าง Address ทั้งชุดได้ จะต้องอ้างเป็น Street หรือ City เพราะ Relational จะทำงานกับข้อมูลในระดับย่อย คือเป็น Atomic

การใช้ Collection โดยหลักการ ไม่ได้จำกัดแค่เป็นอาเรย์ แต่สามารถกำหนดให้เป็นอื่น ๆ ได้ กล่าวคือ (1) Set: สมาชิกไม่ซ้ำ ลำดับไม่สนใจ (2) List: สมาชิกซ้ำได้ ลำดับมีนัยสำคัญ (3) Bag: สมาชิกซ้ำได้ ลำดับไม่สนใจ โดยข้อดีของ Collection คือ ทำให้เกิดความคล่องตัวในการทำงาน โดยไม่จำเป็นต้องแยกออกเป็นตาราง แต่ต้องระวังว่า มีความซ้ำซ้อนกันอยู่ หรือไม่

ประโยชน์ของเทคโนโลยี Object Oriented ที่นำมาใช้ในDatabase

1) สนับสนุนการใช้ Internal Identifiers ซึ่งกำหนดโดย DBMS และไม่ผูกติดกับ OS แทนการใช้ PK, FK ซึ่งเป็น External Identifiers
2) สนับสนุน Complex Structure คือ ข้อมูลอาจมีลักษณะเป็น Collection, Array, Group of Records อยู่ด้วยกันได้ เช่น Address เป็น Composite Attribute ซึ่งรวม City, State ไว้ด้วยกัน
3) สนับสนุน Complex Data Type ซึ่งเป็นจุดอ่อนของ Relational Model ที่สนับสนุนเพียง Simple Data Type

สรุปลักษณะแอพพลิเคชั่นชนิดต่าง ๆ ในทาง Database แบ่ง เป็น 4 ส่วน คือ

1) Simple Data/Without Queries เป็นลักษณะการเก็บข้อมูลแบบ Simple เช่น Character, Integer, Number เป็นต้น โดยไม่มีการถามหาข้อมูลอะไรในเนื้อหานั้น เช่น Word Processing ทำการบันทึกลงแฟ้ม ซึ่งไม่จำเป็นต้องค้นหาข้อมูล การบันทึกเป็นแฟ้มธรรมดา ก็สามารถรองรับการทำงานได้

2) Simple Data with Queries ลักษณะข้อมูล เช่น Character, Integer, Real สามารถใช้ RDBMS รองรับได้ และสามารถค้นหาข้อมูล โดยใช้ภาษา SQL ได้ เช่น สามารถค้นหาข้อมูลของ คุณสมชาย และมีบ้านอยู่ที่เชียงใหม่ จากฐานข้อมูลได้
3) Complex Data Without Queries ลักษณะข้อมูลเริ่มมีความซับซ้อน เช่น มัลติมีเดีย ข้อมูลต้องการจัดเก็บ ไม่ต้องการคิวรี่ แต่ต้องการเพียงแค่แสดงให้เห็นเท่านั้น ซึ่ง RDBMS ก็สามารถรองรับได้ เช่น การเก็บรูปเข้าไป และสามารถดึงออกมาแสดงผลได้
4) Complex Data with Queries ลักษณะข้อมูลซับซ้อน เป็นได้หลายรูปแบบ และต้องการค้นหาได้ เช่น ต้องการค้นหาโดยให้ Match หน้าคนกับรูปที่มีใน Database หรือ Match ลายนิ้วมือให้เข้ากับลายนิ้วมือที่มีใน Database ซึ่งใน RDBMS จะทำได้ลำบาก หรือ GIS ต้องการหาพนักงานที่มีบ้านอยู่ไม่เกิน 5 กิโลเมตร จะต้องมีการเก็บพิกัด และมีเมดธอดรองรับเพื่อค้นหา เป็นต้น ระบบฐานข้อมูลที่เหมาะสม ได้แก่ Object DBMS และ Object Relational DBMS ต่างกันที่ความยากง่ายและความคุ้นเคยของผู้ใช้ กล่าวคือ
    1. Object DBMS เป็นระบบฐานข้อมูลที่มีประสิทธิภาพมาก และรองรับ Complex Object ได้ดี รวมถึงสามารถสืบค้นข้อมูลได้ แต่ข้อเสียประการสำคัญของ ODBMS คือ ยึดติดกับ Host Language มากเกินไป เช่น C++, Java เป็นต้น ทำให้ยากต่อการเรียนรู้และใช้เวลานาน ในด้านภาษาที่ใช้ในการสืบค้น คือ OQL (Object Query Language) ซึ่งเป็นภาษาตามมาตรฐาน ODMG
    2. Object Relational DBMS พัฒนาจาก RDBMS ซึ่งผู้ใช้คุ้นเคย สามารถทำความเข้าใจและเรียนรู้ได้ง่าย นอกจากนี้ ยังสามารถรองรับความซับซ้อนของข้อมูลและระบบงาน รวมถึงมีภาษาที่มีประสิทธิภาพในการสืบค้นข้อมูล คือ ภาษา SQL ตามมาตรฐาน SQL3

แอพพลิเคชั่นในภายหน้า คาดหมายว่า จะมีลักษณะเป็น Complex Data with Queries ซึ่งจำเป็นต้องใช้ Object Oriented Technology เข้ามาช่วยใน Database ซึ่งจะช่วยให้มีศักยภาพมากขึ้น เช่น อนุญาตให้สร้าง Data Type ใหม่ ๆ หรือสร้าง Operation Type ใหม่ ๆ ให้ใช้งาน ซึ่งโดยปกติผู้ผลิต DBMS จะไม่ทำคลาสพิเศษเหล่านั้น แต่จะส่งให้ Third Party เป็นผู้ออกแบบ และพัฒนาขึ้นมาแทน โดยอาจมีลักษณะเป็น Library Class เฉพาะด้านและมีเมดธอดมาเสริม เช่น Library Class เกี่ยวกับการค้นหา และเปรียบเทียบลายนิ้วมือ เป็นต้น

2.2 ระบบฐานข้อมูลเชิงวัตถุสัมพันธ์ (Object Relational DBMS)

เป็นระบบฐานข้อมูลที่ออกแบบมาเพื่อรองรับความต้องการงานที่มีความซับซ้อน และต้องการความง่ายในการพัฒนาและดูแลรักษา โดยการเพิ่มเติมส่วนที่เป็นเทคโนโลยีเชิงวัตถุเข้าไปไว้รวมกับ Relational แบบเดิม ดังนั้น ORDBMS ที่ดี จะมีคุณสมบัติที่สามารถ EXTEND ระบบได้ คือ สามารถสร้าง UDTs/UDFs ได้ และ สามารถสร้าง Complex Objects/Functions ได้ โดยการใช้ Type Constructor ประเภทต่าง ๆ คือ row, collection, reference

2.2.1 กลไกสนับสนุน 4 ประการการสนับสนุนเชิงวัตถุ ORDBMS

1) Naturalness ได้แก่ การช่วยให้มองข้อมูลได้ตามลักษณะที่เป็นจริง (Semantic) ง่ายต่อการทำความเข้าใจ นอกจากนี้ สามารถสร้างหรือกำหนด Data Type, Function ใหม่ได้ตามความเหมาะสม
2) Encapsulation ใน ORDBMS จะให้ USER มองข้อมูลต่าง ๆ เป็น Row Objects แต่จะอนุญาตให้ผู้ใช้มองเห็น Attribute(s) หรือ Field(s) ต่าง ๆ ได้ และใช้งานผ่าน Query Language เท่านั้น
3) OIDs ใช้สำหรับ Complex Objects ซึ่งจะช่วยในการอ้างถึงข้อมูลอื่น (References) ที่เกี่ยวข้อง (OIDs จะไม่ใช้กับ Base Types) OIDs สามารถกำหนดโดยผู้ใช้ หรือให้ระบบสร้างให้ก็ได้ โดยหากให้ระบบสร้างให้ ค่า OID จะมีความยากต่อการอ่านและแปลความหมาย นอกจากนี้ ค่า OID จะมีลักษณะ Unique เสมอ
4) Data Conversion & Ordering ช่วยให้ผู้ใช้ สามารถกำหนด Input หรือ Output ของ Function/Operator สำหรับ Complex Objects ได้ตามความเหมาะสม

2.2.2 การถ่ายทอดคุณสมบัติ (Inheritance)

Inheritance เป็นคุณสมบัติที่สำคัญ ที่ใช้ในเทคโนโลยีเชิงวัตถุ ซึ่งช่วยให้สามารถเรียกใช้วัตถุชนิดต่าง ๆ ที่มีอยู่แล้วได้ (Reusability) และผู้ใช้สามารถเพิ่มเติม หรือเปลี่ยนแปลงแก้ไขคุณสมบัติที่ได้รับถ่ายทอดมาด้วย ลักษณะของ Inheritance ข้อมูลจะเป็น Composite Types โดยลักษณะของ Inheritance ใน ORDBMS

1. สร้าง Type ที่ต้องการด้วยคำสั่ง Create Type โดยจะเป็นการสร้างข้อมูลชนิดใหม่ขึ้น

Create Type PERSON_T(Name varchar(30));

Create Type EMPLOYEE_T(Salary integer, StartDate Date, Address varchar(30))

UNDER PERSON_T;

Create Type STUDENT_T(Gpa float) UNDER PERSON_T;

Create Type STUDENT_EMP_T(Percent float) UNDER EMPLOYEE_T, STUDENT_T

2. สร้างตารางเพื่อรองรับข้อมูลที่ต้องการจัดเก็บ โดยใช้คำสั่ง Create Table

Create Table PERSON of type PERSON_T;

Create Table EMPLOYEE of type EMPLOYEE_T under PERSON_T;

Create Table STUDENT of type STUDENT_T under PERSON_T;

Create Table STUDENT_EMP of type STUDENT_EMP_T

under STUDENT_T, EMPLOYEE_T;

2.2.3 ขอบเขตของการใช้ SQL Command Inheritance

ใน ORDBMS จะมองข้อมูลได้ 2-ลักษณะ กล่าวคือ

1. มองข้อมูลตามแนว Hierarchy ลงไป เช่น

Select name from emp where salary=10000;

ผลลัพธ์ที่ได้ จะหมายถึง name ทั้งจาก emp และ student_emp หรือ

Select * from PERSON;

ผลลัพธ์ที่ได้ คือ ข้อมูลทั้งหมด คือ PERSON, EMP และ STUDENT และ STUDENT_EMP

จากตัวอย่างนี้ จะเป็นการดึงข้อมูลทั้งหมด โดยการเรียกจากส่วนที่เป็น Root เรียกลักษณะนี้เรียกว่า "Jagged Return"

2. มองข้อมูลเฉพาะตำแหน่งที่อยู่

Select name from ONLY(emp) where salary=10000;

ผลลัพธ์ที่ได้ จะหมายถึง name เฉพาะ emp เท่านั้น

2.2.4 การถ่ายทอดคุณสมบัติของฟังก์ชั่น Inheritance of Functions

1. สร้าง Function ที่ต้องการใช้งาน ด้วยคำสั่ง Create Function

Create Function OVERPAID(Employee_T, Arg1)

Returning Boolean

As Select Arg1.salary > (select salary from EMP where name='Joe');

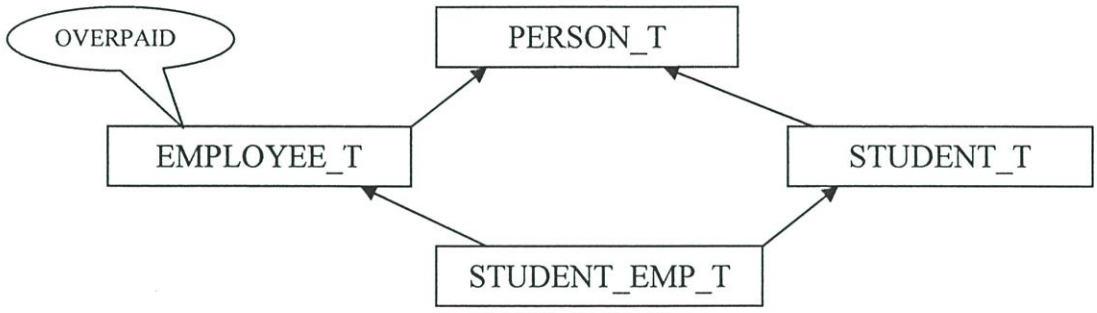
End Function

2. การเรียกใช้งาน

Select e.name from ONLY(emp) e where OVERPAID(e); จะได้เฉพาะข้อมูลใน EMP

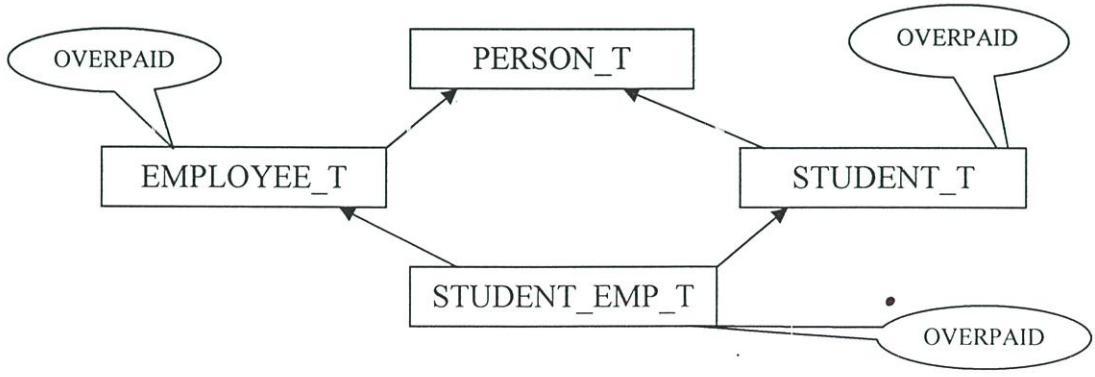
Select e.name from emp e where OVERPAID(e); จะได้ข้อมูลใน EMP +STUDENT_EMP

สิ่งที่เกิดขึ้นจากการเรียกใช้ คือ ORDBMS จะ Inherit UDF ลงมาให้ด้วย



รูปที่ 2. 1 ตัวอย่างคลาสและเมดธอด

สิ่งที่ต้องระวัง คือ จากตัวอย่าง Select e.name from emp e where OVERPAID(e); สิ่งที่ได้คือข้อมูลจะมาจากทั้ง EMP และ STUDENT_EMP นั้นหมายความว่า OVERPAID จะทำงานทั้งใน EMP และใน STUDENT_EMP หากทั้งสองตารางมีการคำนวณหรือประมวลผลใน OVERPAID ที่แตกต่างกัน จะทำให้เกิดความผิดพลาด ดังนั้น หาก OVERPAID ทำงานแตกต่างกัน จะต้องทำการ OVERIDE เพื่อป้องกันความผิดพลาดที่อาจเกิดขึ้นได้



รูปที่ 2. 2 ตัวอย่างคลาสและเมดธอดที่ได้รับการถ่ายทอดจากคลาสแม่ 2 คลาส

ในกรณีที่ Sub Type ซึ่งจากตัวอย่าง คือ STUDENT_EMP_T ได้รับการถ่ายทอด จากทั้ง EMPLOYEE_T และ STUDENT_T ดังนั้น "OVERPAID" จาก EMPLOYEE_T จะถูกถ่ายทอดมาที่ STUDENT_EMP_T และหาก STUDENT_T มี "OVERPIAD" ด้วย หมายความว่า STUDENT_EMP_T จะได้รับการถ่ายทอด "OVERPIAD" จากทั้ง 2 ทาง ทำให้ STUDENT_EMP_T เกิดความสับสนในการเลือกใช้งาน "OVERPIAD"

การแก้ไขกรณีนี้ ผู้ที่ดูแล ซึ่งโดยปกติ คือ DBA อาจแก้ไขโดยทำการ Override Function "OVERPAID" ใน STUDENT_T ไว้ใหม่ให้เหมาะสมและถูกต้อง

2.2.5 สรุป Inheritance ใน ORDBMS

1. DATA และ METHOD สามารถถ่ายทอดได้
2. สนับสนุน Polymorphism คือ สามารถ Override และ Overloading
3. Types / Tables มีความแตกต่างกัน โดย Types เป็นเสมือนแม่แบบ หรือชนิดของข้อมูล แต่ Tables เป็นวัตถุที่จะเรียกใช้ Types และเป็นตัวเก็บข้อมูล
4. สนับสนุนการทำ Multiple Inheritance ทั้งนี้ขึ้นอยู่กับ DBMS ว่าสนับสนุนหรือไม่

2.2.6 คุณสมบัติ Extensible User Define Type

1. User Define Data Type (UDTs) เป็นการสร้างข้อมูลประเภทใหม่ เพิ่มจากข้อมูลพื้นฐาน
2. User Define Function (UDFs) เป็นการสร้างฟังก์ชั่น เพื่อรองรับการทำงานที่ซับซ้อน รองรับการทำงานกับข้อมูลพื้นฐาน และข้อมูลประเภทใหม่ที่สร้างขึ้นมา

   ตัวอย่างการสร้าง

   Create Function Hour_Pay(Integer Arg1)

   Returning Float

   As Select Arg1/2000;

   การเรียกใช้งานฟังก์ชั่น เรียกผ่าน Where Clause Condition เช่น select name from emp where HourPay(salary)>15.50;

   เรียกผ่าน Select Clause เช่น select salary_dif_joe(e.salary) from emp e where e.name="Fred";
3. User Define Operator (UDOs) เป็นการสร้าง Operator ชนิดใหม่ ให้เหมาะกับงานและข้อมูล

4. User Define Aggregation (UDAs) เป็นการสร้างฟังก์ชั่นใหม่ นอกเหนือจาก SQL Aggregate Operators 5 ตัว คือ Count, Sum, Min, Max, Avg โดยสามารถใช้ใน Function, Operator ได้ เช่น ต้องการหาค่า Median ของ Employee เป็นต้น โดยไม่จำเป็นต้องสร้างเป็น Procedure ซึ่งจะพบว่า การประมวลผลข้อมูลในลักษณะที่เป็น Aggregate Calculation จะต้องมีการรวบรวมค่าที่เกี่ยวข้องทั้งหมดก่อน และคำนวณ (โดยหลักการ คือ Initialization → Iteration → Finalization) ซึ่งจะทำให้ประสิทธิภาพของระบบลดลง

2.2.7 การสนับสนุน Complex Objects

ระบบฐานข้อมูลเชิงวัตถุสัมพันธ์ สนับสนุนการทำงานกับข้อมูลที่มีความซับซ้อน ผ่าน Type Constructor ชนิดต่าง ๆ ได้แก่

1. Rows as a Type Constructor ได้แก่ การสร้างประเภทข้อมูลชนิดใหม่ขึ้น (Type) เช่น

   Create row type PHONE (area varchar(3), number varchar(10));

   Create Table TEST of type PHONE; โดย TABLE จะเป็นตัว Containers ที่เก็บข้อมูล
2. Sets as a Type Constructor หรือ Collections โดย Set Type จะมี 4 ลักษณะได้แก่ Collection of BASE-TYPES, ROW TYPES, COLLECTIONS, REFERENCES
3. References as a Type Constructor ได้แก่ การสนับสนุนการใช้ Reference หรือ Pointer เพื่ออ้างถึงข้อมูลอื่นที่เกี่ยวข้อง โดยมี 3 ลักษณะ ได้แก่ Reference to a ROW, to a Collection, to a Base Data Type ตัวอย่างการสร้าง Reference

   Create row type DEPARTMENT(

   Dname varchar(30),

   Floor integer,

   ManageRef ref(emp_t) → Reference ไปข้อมูลประเภท “emp_t”

   References(emp) → Reference โดยชี้ไปที่ Table “emp”

   Workers set(ref(emp_t) → Set ของ Reference ประเภท “emp_t”

   References(emp)); → Reference โดยชี้ไปที่ Table “emp”

   Create table TEST of type DEPARTMENT;

2.2.8 การจัดการข้อมูลใน Row Types (Manipulate Row Types)

1. ใช้ UDFs รับค่าผ่าน Argument หรือ Return ค่าจาก Row Type

ตัวอย่าง สร้างฟังก์ชั่นเพื่อคำนวณค่า bonus

```
Create Function bonus(employee_t arg1)
        Returning integer
        As Select arg1.salary * .01;
End Function;
```

โจทย์ เมื่อต้องการหาว่า "Mark" ได้รับ Bonus เท่าใด สามารถทำได้โดยการส่งค่า Argument คือ e.name="Mark" ไปที่ UDF ชื่อ bonus

Select bonus(e) from emp e where e.name="Mark"; ผลลัพธ์ จะได้ค่า Bonus ของ Mark

2. ใช้ UDFs ที่มีการ Return ค่าของ Row Type สามารถวางไว้ใน from clause ของ SQL ได้

เช่น select e.name from emp e where bonus(e) > 1000;

ข้อสังเกต การใช้โดยผ่าน from clause ใน SQL Statement จะต้องระบุเป็นชื่อ Table หรือ ใช้ UDFs ที่มีการ return ค่ากลับในรูป row type เช่น

สร้างฟังก์ชั่นเพื่อหาหมายเลขโทรศัพท์

```
Create Function shoe_phone()
Returning (phone_t)      → คืนค่ากลับมาเป็น row Type
As Select phone from dept
        Where dname="Shoe";
End Function;
```

ตัวอย่าง Select number from table(shoe_phone());

ลักษณะการใช้แบบนี้ เทียบได้กับการใช้ Nested Query

Select number from table(Select phone from dept where dept="Shoe");

3. ใช้ Cascade Dot Notation อ้างถึง Attributes ต่าง ๆ ใน Row Object นั้น ๆ (Path Expression)

Select phone.number from dept Where dname="Shoe";

โดยที่ phone เป็น Attribute ใน Table "Dept" โดย Type ของ Phone คือ Phone_T ซึ่ง phone เป็น Attribute ย่อยใน Phone_T ซึ่งการอ้างแบบนี้ จะมีลักษณะคล้ายกับการอ้างในรูปแบบ Embedded Class, References Class ที่ใช้ใน Object DBMS

2.2.9 การจัดการข้อมูลแบบ Collections ที่เป็น Rows Types

เป็นการจัดการ Rows ที่มีลักษณะเป็น collection ชนิดต่าง ๆ โดยลักษณะข้อมูลที่เหมาะสมที่จะจัดเก็บใน Collection มักจะเป็นข้อมูลที่ไม่ค่อยมีการเปลี่ยนแปลง หรือ ไม่เปลี่ยนแปลง

1. UDFs ผ่าน Where เช่น Select dname from dept where has85(autos);
   เป็นการหาชื่อ dept name ที่มี 1985's Automobile โดยผ่าน UDF ที่มีการ return ค่าเป็น Boolean
2. UDF ผ่าน From Clause เช่น
   Create Function Tran_Auto()
   Return set(Auto_T) → Return ค่ากลับในรูปของ Set ของ Type ชื่อ "Auto_T"
   As Select autos from dept
   Where dname="Transportation";
   End Function
   เมื่อต้องการค้นหาชื่อรถในปี 1989 จะใช้
   Select name from table (Tran_Auto) where year=1989;
3. UDF ผ่าน Cascade Dot Notation
   สามารถเรียกใช้ได้หลายลักษณะ เช่น
   Select dname from dept where 1989 in autos.year;
   โดย เงื่อนไข คือ
   Autos เป็นชื่อ Attribute ที่อยู่ใน dept และมี Type เป็น Auto_T
   Year เป็น Attribute ย่อย ที่อยู่ใน "Auto_T" Type
   Select dname from dept
   Where autos.(year=1985 and name='Ford');
   หรือ Select dname from dept
   Where exists (Select 1 from dept.autos where year=1985 and name='Ford');

2.2.10 การจัดการข้อมูลแบบ Collections ที่เป็น Reference Types of Reference Types

การใช้ Reference จะใช้แทน PK/FK ในระบบฐานข้อมูลเชิงสัมพันธ์ ซึ่งจะช่วยให้ระบบมีประสิทธิภาพมากยิ่งขึ้น

ตัวอย่าง

รูปที่ 2. 3 แสดงการอ้างถึงข้อมูลด้วย Reference

ถ้าต้องการหา StartDate ของ Manager ที่ Dept="Shoe" จะต้องใช้คำสั่ง

Select e.StartDate From emp e, dept d Where e.name=d.manager and d.dname="Shoe"; โดยปกติ RDBMS จะดูแล Referential Integrity ซึ่งจะช่วยรับรองได้ว่า FK Values ที่ปรากฏในฝั่ง Dept จะต้องปรากฏใน Employee ในฐานะ PK

ในระบบฐานข้อมูลวัตถุเชิงสัมพันธ์ (ORDBMS) จะอนุญาตให้เก็บค่า OID ของข้อมูลอื่นที่เกี่ยวข้องในตารางอื่น ๆ หรืออีกนัยหนึ่ง หมายความว่า ODBMS อนุญาตให้ใช้ OID หรือ Pointer ในการอ้างถึงข้อมูลอื่น ๆ ได้ แทน PK/FK ซึ่งข้อดีของการใช้ OID คือ ช่วยให้มั่นใจได้ว่า ค่า OID จะ unique และเปลี่ยนแปลงไม่ได้ ตัวอย่าง Dept Table ที่สร้างไว้ จะมี Attribute ชื่อ Workers ที่เก็บ set ของ references ที่ชี้ไปที่ Employee_T

```
Create row type DEPARTMENT(
        Dname varchar(30),
        Floor   integer,
        ManageRef       ref(emp_t)              -- Reference ไปข้อมูลประเภท "emp_t"
                        References(emp)         -- Reference โดยชี้ไปที่ Table "emp"
        Workers         set(ref(emp_t)          -- Set ของ Reference ประเภท "emp_t"
                        References(emp));       -- Reference โดยชี้ไปที่ Table "emp"
```

2.2.11 วิธีการอ้างถึงข้อมูลแบบ Collection of ReferencesType

1. โดยการใช้ Select Statement

   Select workers from dept where dname="Shoe";

   จะได้ ค่า reference หรือ OID ของ workers ในแผนก "Shoe"

   Select deref(*)

   From table(Select workers from dept where dname="Shoe");

จะได้ ข้อมูล workers ในแผนก “Shoe”

Select deref(*).StartDate

From table(Select workers from dept where dname=”Shoe”);

จะได้ ค่า StartDate ของ workers ในแผนก “Shoe”

2. โดยการใช้ UDFs

โดยการสร้าง UDF และรับค่าผ่าน Argument เพื่อใช้เป็นเงื่อนไขในการอ้างถึงข้อมูล

Create Function get_mgr(Varchar(30) Arg1)

Returning Employee_T → Return type จะเป็น TYPE

As select deref(manager_ref) from dept where dept=Arg1);

เมื่อต้องการเรียกใช้งาน สามารถใช้คำสั่ง Select และระบุค่า Argument ได้โดยตรง

Select get_mgr(“Shoe”);

3. โดยการใช้ผ่าน Dot Notation

Select deref(manager_ref).birthdate from dept

Where dname=”Shoe”;

หรือ Select deref(manager_ref).StartDate, deref(manager_ref).salary from dept

Where dname=”Shoe”;

2.2.12 การทำ Query ข้อมูล

Select d.dname, e.name, e.salary From emp e, dept d

Where e.dname=d.dname and d.name=”Shoe”;

ในกรณีที่ข้อมูล return กลับมาในรูปของ set of rows สามารถลดรูปได้ ดังนี้

Select d.dname, set(e.name,e.salary) → set ของ emp e

From emp e, dept d

Where e.dname=d.dname and .name=”Shoe”

Order by d.dname;

2.2.13 การ Update สามารถทำได้ทั้งในแบบเดิมและผ่าน OID

1. กรณีการ Update โดยตรง เช่น

Update dept Set manager=”Mark”

```
        Where dname="Shoe";
```

2. กรณีการ Update โดย OID

```
Updete dept Set manager_ref=
        (select ref(emp) from emp where name="Mark");
```

2.3 ระบบฐานข้อมูลเชิงวัตถุ (Object Oriented Database Management System)

ยุคของระบบฐานข้อมูลยุคใหม่ เกิดขึ้นมาเพื่อช่วยแก้ปัญหาหลายประการ ที่ระบบฐานข้อมูลแบบเดิม อาจไม่สนับสนุน หรือทำได้ไม่ดีนัก ระบบฐานข้อมูลเชิงวัตถุ ซึ่งความจริงสามารถแบ่งได้เป็น 2 แนวทางได้แก่ ระบบฐานข้อมูลเชิงวัตถุสัมพันธ์ (Object Relational) ซึ่งใช้เทคโนโลยีเชิงสัมพันธ์แบบเดิม แต่เพิ่มความสามารถทางด้านเทคโนโลยีเชิงวัตถุ หรือ Object Oriented ซึ่งช่วยสนับสนุนให้รองรับระบบงานยุคใหม่ที่ข้อมูลมีลักษณะซับซ้อนได้ดี และแนวทางที่สอง คือ ระบบฐานข้อมูลเชิงวัตถุ (Pure Object Database) ที่ออกแบบมาเพื่อรองรับการเก็บและการเรียกใช้ข้อมูลที่มีความซับซ้อนมากได้เป็นอย่างดี แต่อย่างไรก็ตาม แม้นเทคโนโลยีดังกล่าว จะถูกออกแบบมาเป็นอย่างดีเพื่อรองรับงานที่มีความซับซ้อน สิ่งที่ต้องพิจารณาร่วมกัน คือ การออกแบบระบบฐานข้อมูลเชิงวัตถุที่ถูกต้อง

โดยทั่วไประบบสารสนเทศต่าง ๆ ที่ทำงานบนโมเดลเชิงสัมพันธ์ มักจะใช้เกี่ยวกับงานทางด้านการบริหารจัดการ, Invoicing เป็นต้น แต่สำหรับระบบฐานข้อมูลที่ใช้เทคโนโลยีใหม่ ทั้งโมเดลเชิงวัตถุสัมพันธ์ และ โมเดลเชิงวัตถุ แอพพลิเคชั่น จะครอบคลุมถึงเรื่องข้อมูลประเภทใหม่ ๆ เช่น วิดิโอ, การจัดการด้านรูปภาพกราฟฟิกส์ หรืองานมัลติมีเดียต่าง ๆ, Time series analysis, financial analysis, scientific databases หรืองานบนเวปไซต์ ที่ต้องนำเสนอข้อมูลที่มีลักษณะซับซ้อน ซึ่งนิยมออกแบบให้ทำงานบนระบบฐานข้อมูลที่ใช้เทคโนโลยีเชิงวัตถุ

ระบบฐานข้อมูลเชิงวัตถุสัมพันธ์ (Object Relational Database) ถูกนำเสนอโดยใช้มาตรฐาน SQL:1999 ในขณะที่ระบบฐานข้อมูลเชิงวัตถุ (Object Oriented Database) จะใช้มาตรฐานที่นำเสนอโดย ODMG

SQL เป็นมาตรฐานที่นำเสนอโดย ISO/IEC และ ANSI เพื่อใช้สำหรับระบบฐานข้อมูลเชิงสัมพันธ์ ที่ใช้โมเดลเชิงสัมพันธ์ที่นำเสนอโดย Codd ซึ่งมาตรฐานปัจจุบันจะใช้ SQL:1999

มาตรฐาน SQL ครั้งแรก ตรวจสอบโดย ANSI ในปี ค.ศ.1986 และโดย ISO ในปี 1987 ต่อมาในปี 1989 ทั้ง ANSI และ ISO ได้ออกมาตรฐานที่รู้จักกันในนาม Addendum ซึ่งได้รวมชุดความสามารถในเรื่อง Data Integrity รวมเข้าไปด้วย จากนั้นในปีต่อมา ANSI ประกาศมาตรฐานเกี่ยวกับ Embedded SQL จากนั้นสามปีได้ออกมาตรฐานใหม่ ที่เรียกกันว่า SQL-92 โดย ISO และ ANSI

SQL-92 ได้พัฒนาความสามารถในด้าน Semantic ของ Relational schema, การแนะนำ operators ใหม่, การพัฒนา Error Handling และ embedded SQL, Intermediate SQL, Full SQL

ในปี 1999 SQL 3 หรือ SQL:1999 ได้เพิ่มความสามารถในด้าน triggers, roles, recusion, object-relational เช่น User Defines Types, Structured Types, Type Hierarchies, Table Hierarchies, Typed Tables, Collection Types เป็นต้น รวมเข้าไป รวมถึงการสนับสนุนงาน SQL/MM หรืองานด้านมัลติมีเดีย

2.3.1 วิธีการนำเสนอของโมเดลเชิงวัตถุสัมพันธ์ (Object Relational representation)

1. Multi-valued attribute ในโมเดลเชิงสัมพันธ์ จะถูกนำเสนอโดยการสร้างเป็นตารางใหม่ แต่ในข้อกำหนดของ SQL:1999 สามารถนำเสนอในรูปของอาเรย์ได้ เนื่องจาก ในโมเดลเชิงวัตถุสัมพันธ์ ไม่บังคับข้อกำหนด 1NF ดังนั้น ค่าในแอททริบิวส์ สามารถมีได้มากกว่าหนึ่งค่า เรียกว่า คอลเลคชั่น (Collection)
2. Relationship ถูกนำเสนอโดยการใช้ Ref Attribute โดยการเก็บค่า OID ของแอททริบิวส์ที่อ้างถึง ซึ่งจะแตกต่างจากการใช้ foreign key โดย Ref Type จะเป็นตัวชี้เพื่อเชื่อมความสัมพันธ์ ไปยังตารางที่เชื่อมอยู่ (Join Table)
3. Composite Attribute ถูกนำเสนอในรูปของ Row Type โดยจะมีลักษณะคล้าย C++ Structure type
4. Generalization นำเสนอโดยการใช้คุณสมบัติ Inheritance ทั้ง Types และ Tables

2.3.2 ส่วนขยายเพิ่มสำหรับคุณสมบัติเชิงวัตถุ (Object Oriented Extension)

ในส่วนของ SQL มีหลายส่วนที่ไม่เป็นวัตถุ แม้ว่าจะได้มีการเพิ่มส่วนที่เป็นวัตถุรวมเข้าไปใน SQL ตัวภาษา SQL ยังคงความสามารถในการทำงานกับโมเดลเชิงสัมพันธ์ได้ โดยใน SQL:1999 ข้อมูลทั้งหมด ยังคงถูกจัดเก็บในรูปตาราง แม้ว่าข้อมูลบางส่วนจะเป็นวัตถุก็ตาม ในส่วนที่มีการขยายความสามารถด้านวัตถุ มีการเพิ่ม User-defined types และ reference types โดย

1. User-defined types มีสองประเภทได้แก่ distinct type และ Structure type กล่าวคือ Distinct type จะอนุญาตให้มีการสร้างข้อมูลชนิดใหม่ โดยใช้ SQL built-in primitive ที่มีอยู่เดิม และ Structured type จะมีความใกล้ชิดกับการเข้าถึง programming language classes โดยอนุญาตให้สร้าง complex data types ที่สามารถใช้ได้ทั้ง column types และ row types โดยหากเป็น column types จะอนุญาตให้ใช้กับแอททริบิวซ์ที่มีลักษณะซับซ้อน เช่น time series, point เพื่อเพิ่มความสามารถสำหรับงานมัลติมีเดีย (SQL/MM) ถ้าเป็นกรณี row types จะอนุญาตให้ออกแบบสำหรับ entities ที่มีความสัมพันธ์และ behavior เพื่อรองรับความต้องการพื้นฐานสำหรับ business objects

   สำหรับ structure types อาจเป็น sub type ของ user-defined type อื่น ๆ ที่ inherit โครงสร้างจาก super-type อื่น ๆ ก็ได้ เช่น create type MANAGER_T under EMPLOYEE_T และ Create table MANAGER of MANAGER_T under EMPLOYEE

   สำหรับเมดธอด จัดเป็นฟังก์ชั่นชนิดหนึ่ง ซึ่งผูกติดกับ structure type ซึ่งจะมี signature แสดงไว้ใน structure type

2. Reference type โดยปกติ structure types ใช้สำหรับสร้าง table types แล้ว แอททริบิวซ์ ของไทป์ จะกลายเป็นคอลัมน์ในตาราง โดยคอลัมน์ จะกำหนดค่า REF สำหรับ row นั้น ๆ เรียกว่า Object Identifier และค่า REF สามารถเป็นได้ทั้ง user generate และ system generated ซึ่งจะแตกต่างจาก referential integrity constraints

2.3.3 ODMG

มาตรฐานสำหรับระบบฐานข้อมูลเชิงวัตถุ ที่นำเสนอโดย ODMG (Object Database Management Group) โดยมีวัตถุประสงค์หลัก คือ เพื่อให้ผู้ใช้ สามารถใช้แอพพลิเคชั่นร่วมกันได้ โดย data schema, programming language binding, DML, Query language จะต้องมีลักษณะ portable และเพื่อช่วยให้แต่ละ products ทำงานร่วมกันได้

มาตรฐานแรก ประกาศในปี ค.ศ.1993 คือ ODMG 1.0 และมีการพัฒนามาเรื่อยถึงรุ่น 1.2 ได้มีการเพิ่มเรื่อง OQL รวมไว้ด้วย ต่อมาในปี 1997 มีการออก ODMG 2.0 เพิ่มเรื่อง Collection types, Java binding และปี 2000 ออก ODMG 3.0 ซึ่งมีการปรับปรุงความสามารถเดิมให้ทำงานได้ดียิ่งขึ้น รวมถึง Java binding และสนับสนุน Object-to-database mapping ซึ่งอนุญาตให้ objects เก็บอยู่ใน relational database ได้

2.3.4 หลักของ ODMG 3.0

1. Object Model: พัฒนาบนรากฐานของโมเดลเชิงวัตถุของ OMG (Object Management Group) โดยการเพิ่มคุณสมบัติต่าง ๆ เพื่อสนับสนุนความต้องการของ DBMS
2. Object Specification Language: มีสองส่วน ได้แก่ ODL ใช้กำหนด Object Types โดยพัฒนาเพิ่มจาก IDL ของ OMG และส่วนที่สอง คือ OIF เป็นข้อกำหนดภาษาสำหรับการ dump, load ข้อมูล
3. Object Query Language (OQL): เป็นภาษาสำหรับการทำ Query และ Update วัตถุต่าง ๆ โดยยืนบนพื้นฐานของ SQL เช่นกัน แต่ semantics ที่ใช้จะมีความแตกต่างกัน โดย OQL จะสนับสนุนผลลัพธ์ในลักษณะที่เป็น Collection types ได้ แต่ SQL จะมีลักษณะเป็นตารางเสมอ
4. Language Binding: เป็นข้อกำหนดการสนับสนุนภาษา C++, Smalltalk, Java รวมถึงกลไกการเรียกใช้ OQL และ Procedures สำหรับทำงานบนระบบฐานข้อมูล และการจัดการทรานแซคชั่นที่เกิดขึ้น

2.4 XML

Extensive Markup Language (XML) เป็นภาษาที่ให้ความชัดเจนในการให้รายละเอียดเกี่ยวกับข้อมูล และการเปลี่ยนแปลงข้อมูล หรือกล่าวได้ว่า เป็นฟอร์แมตที่อธิบายถึงรายละเอียดของโครงสร้างและแบบของข้อมูล โดยจะทำงานกับข้อมูลโดยตรง จากลักษณะการนำเสนอรายละเอียด และการนำข้อมูลตลอดจนโครงสร้างข้อมูลมาแสดงได้ในรูปแบบ Text ผ่านทาง HTTP ที่เปิดให้ข้อมูลขึ้นใหม่และมีความสามารถในการจัดข้อมูลได้อีกด้วย XML เป็นส่วนหนึ่งของ Standard Generalized Language Markup Language (SGML) ที่เป็นข้อกำหนดในการสร้างหรือจัดทำเอกสารในรูปแบบอิเล็คทรอนิกส์ที่กำหนดโดย W3C หรือ World Wide Web Consortium

XML จะเป็นความสะดวกในการจัดการด้านระบบการติดต่อกับผู้ใช้จากโครงสร้างของข้อมูล สามารถนำข้อมูลจากหลายแหล่งมาแสดงผลและประมวลผลร่วมกันได้ ไม่ว่าจะเป็นข้อมูลลูกค้า รายการสั่งซื้อ ผลการวิจัย รายการรับชำระเงินข้อมูลเวชระเบียน รายการสินค้าหรือข้อมูลสารสนเทศอื่นๆ ก็สามารถแปลงให้เป็น XML ได้ และในส่วนของข้อมูลสามารถปรับให้เป็น HTML เพื่อใช้แสดงผลได้ตามความเหมาะสม โดยในกระบวนการนำข้อมูล XML มาใช้งานใน Application นั้นจะมี XML Parser เป็นตัวกลางในการดึงข้อมูลจากเอกสาร XML และ Application ซึ่งเป็น API ชนิดหนึ่ง โดย API ที่นิยมกันมากคือ DOM ย่อมาจาก Document Object Model และ SAX ย่อมาจาก Simple API for XML ซึ่งต่างก็มีวิธีในการดึงข้อมูลที่แตกต่างกันคือ DOM จะมองเอกสาร XML ในลักษณะของโครงสร้าง

ต้นไม้ (Tree) โดยการอ่านเอกสาร XML มาเป็น Tree ในหน่วยความจำของเครื่องที่กำลังทำงาน ประกอบด้วย Element หรือ Attribute ต่างๆ การเข้าถึงข้อมูลจึงเป็นการเดินไปตามกิ่งก้านต่างๆ ทั้งเป็นแบบต่อเนื่อง หรืออ้างอิง ก็ได้ อย่างไรก็ดี ข้อจำกัดของ DOM อยู่ที่ขนาดหน่วยความจำของเครื่องว่าจะสามารถรองรับข้อมูลได้เพียงใด ทั้งนี้เนื่องจาก DOM จะทำการอ่านข้อมูลทั้งหมดมาเก็บไว้ในหน่วยความจำเพียงครั้งเดียว ข้อดีคือ ง่ายต่อการเขียนชุดคำสั่ง และในส่วนของ SAX เป็นวิธีการจัดการเอกสาร XML ด้วยแนวทาง Event-Driven โดย SAX จะไม่อ่านข้อมูลทั้งหมดเข้ามาในหน่วยความจำ แต่จะอ่านเอกสารเริ่มต้น และจะมีการสร้าง Event ควบคุม โปรแกรมต้องทำหน้าที่ ดักจับ Event เหล่านี้ก่อน จากนั้นจึงจัดการกับข้อมูลที่ต้องการ

บทที่ 3
ระบบฐานข้อมูลที่ใช้ในการวิจัย

3.1 IBM DB2 Universal Database

IBM DB2 เป็นระบบปฏิบัติการฐานข้อมูลเชิงวัตถุสัมพันธ์ (Object Relational DBMS) พัฒนาโดยบริษัท IBM สามารถทำงานได้หลายแพลทฟอร์ม สนับสนุนการใช้คำสั่งและกลไกต่าง ๆ ตามมาตรฐาน SQL3

3.1.1 ประเภทของวัตถุที่ใช้ในระบบฐานข้อมูล DB2 ดังรายละเอียดต่อไปนี้

3.1.1.1 ประเภทข้อมูล หรือ Data Types แบ่งเป็น 2 ประเภทใหญ่ ได้แก่

IBM DB2 Built-in data type แบ่งเป็น 3 ประเภทหลัก ได้แก่

1) NUMERIC ได้แก่ Integer, Decimal, Floating Point
2) STRING ได้แก่ Character String, Binary String
3) DATETIME ได้แก่ Date, Time, Timestamp

User Define Data Types ซึ่งแบ่งย่อยได้ดังนี้

1) User Define Distinct Type เป็นการกำหนดบนข้อมูลพื้นฐาน
2) User Define Structure Type เป็นการกำหนดชนิดข้อมูลที่เป็นโครงสร้าง ซึ่งประกอบด้วยแอททริบิวซ์ชนิดต่าง ๆ
3) User Define Reference Type เป็นการอ้างถึงข้อมูลในตารางอื่น ที่ใช้ User Define Structure Type

3.1.1.2 Tables เป็นเซตของข้อมูลที่ไม่มีการเรียงลำดับ โดยประกอบด้วย row, column โดยตารางมี 3 ประเภท ได้แก่

1) Permanent base ได้จากการสร้างโดยใช้คำสั่ง Create Table
2) Temporary declared เกิดจากการที่ผู้ใช้หรือ แอพพลิเคชั่นกำลังทำงาน
3) Temporary derived

3.1.1.3 Schemas

เป็นการรวมกลุ่มของวัตถุต่าง ๆ ในฐานข้อมูลเข้าไว้ด้วยกัน โดยมีรูปแบบ Schema_name.object_name ซึ่งโดยปกติถ้าไม่มีการระบุ ชื่อ Schema จะได้จาก Authorization Id หากต้องการเปลี่ยน สามารถระบุได้โดยใช้คำสั่ง “set current schema” ได้

3.1.1.4 Table Spaces

เป็นระดับชั้น (Logically layer) ที่สร้างขึ้นระหว่าง database และ tables ซึ่งถูกจัดเก็บอยู่ใน database โดย IBM DB2 แบ่ง Table Spaces ได้ 2-ประเภท ได้แก่

- System Managed Space (SMS) คือ ไฟล์ที่ถูกควบคุมและจัดการ โดยระบบปฏิบัติการ
- Database Managed Space (DMS) คือ พื้นที่ในการจัดเก็บข้อมูล ควบคุมและจัดการโดย DBMS

3.1.1.5 Views

เป็นตารางที่ได้จากการดึงค่าจากตารางอื่น ๆ โดยสามารถกำหนดสิทธิในการเข้าถึงข้อมูลเฉพาะส่วน เช่น บางคอลัมน์ เป็นต้น และสามารถกำหนดสิทธิการใช้งานได้ เช่น Delete, Update, Insert, Read-Only เป็นต้น

3.1.1.6 Indexes

เป็นวัตถุชนิดหนึ่ง ซึ่งถูกสร้างขึ้นและจัดเก็บจริงใน 1 ตาราง โดยอาจมี Index ได้หลายตัวในแต่ละตาราง เหตุผลในการสร้าง Index คือ เพื่อช่วยให้มั่นใจได้ว่าข้อมูลที่จัดเก็บจะไม่ซ้ำซ้อนกัน (Uniqueness) และ ช่วยเพิ่มประสิทธิภาพในการค้นหาข้อมูล Index ใน DB2 จะสามารถกำหนดได้ทั้งแบบ Ascending, Descending รวมถึงกำหนดให้ Unique, Non-Unique ตลอดจนสามารถสร้างได้บนคอลัมน์เดียว หรือหลายคอลัมน์รวมกันก็ได้ และสามารถเข้าถึงข้อมูลได้ทั้ง forward และ backward นอกจากนี้ Indexes จะถูกดูแลความถูกต้องโดย DB2

3.1.1.7 Packages

เป็นการรวมกลุ่มของ SQL Statements ซึ่งจะถูกอ้างจาก DB2 Applications

3.1.1.8 Transactions

เป็นเซตของคำสั่ง SQL (อาจมีคำสั่งเดียว หรือหลายคำสั่งก็ได้) ซึ่งถูกประมวลผลภายในงานเดียวกัน เรียกว่า Unit of Work โดยกิจกรรมต่าง ๆ ที่ DBMS ทำ จะจัดเก็บไว้ใน Database Log Files ไม่ว่ากิจกรรมนั้น จะสำเร็จหรือไม่ก็ตาม โดยแต่ละ Transaction จะสิ้นสุดเมื่อมีการ Commit, Rollback

3.1.1.9 Locks

เป็นกระบวนการที่ช่วยลดความผิดพลาด หรือหลีกเลี่ยงปัญหาต่าง ๆ ในการใช้งานข้อมูลเดียวกันในเวลาเดียวกัน โดยที่สามารถทำงานได้ถูกต้อง โดย DBMS จะจัดเก็บการ locks ไว้ใน Database Server Lock List memory ซึ่ง DB2 สนับสนุนการ lock ระดับตาราง และระดับ row และใช้กลไก Isolation Level ช่วยให้การทำงานถูกต้องและมีประสิทธิภาพมากยิ่งขึ้น

3.1.1.10 Log Files

เป็นกลุ่มของแฟ้มข้อมูลที่เกิดขึ้นในขณะที่ DBMS ประมวลผลจาก SQL Statements ต่าง ๆ โดย DB2 จะใช้เทคนิคที่เรียกว่า Write-Ahead logging method ซึ่งจะช่วยให้มั่นใจได้ว่าข้อมูลจะมีการปรับเปลี่ยนให้จริง โดยการบันทึกการเปลี่ยนแปลงต่าง ๆ ที่เกิดขึ้นลง Log Files ก่อน จากนั้นจึงเขียนจริงลงใน Database โดยขนาดของไฟล์สามารถขยายได้ถึง 32 GB.

3.1.2 Constraints

ได้แก่ ข้อกำหนดที่ระบุไว้ในการสร้างหรือแก้ไขตาราง ได้แก่

3.1.2.1 Unique Constraint ช่วยให้ค่าของข้อมูลมีความเป็นหนึ่งเดียว ไม่มีการซ้ำซ้อน

3.1.2.2 Referential Integrity ช่วยดูแลความถูกต้องในขณะที่มีการ INSERT, UPDATE, DELETE โดยช่วยดูแลความถูกต้องของข้อมูลที่เป็น foreign key โดยใช้กฎเกณฑ์ 3 ประเภท คือ INSERT RULE, DELETE RULE เป็นกฎในการลบข้อมูล โดยมีเงื่อนไข ดังนี้ RESTRICT, NO ACTION, CASCADE, SET NULL และ UPDATE RULE เป็นกฎที่ใช้ในการแก้ไขข้อมูล โดยมีเงื่อนไข คือ RESTRICT, NO ACTION

3.1.2.3 Table Check Constraint ดูแลการเปลี่ยนแปลงของข้อมูล เพื่อไม่ฝ่าฝืนกฎเกณฑ์หรือเงื่อนไข

3.1.3 การจัดการข้อมูลเชิงวัตถุด้วย IBM DB2 ด้วย Structured type และ Typed Table

Table สามารถถูกกำหนดหรือสร้างขึ้นโดยใช้ Structured Type เรียกว่า Type Table โดย Structure ดังกล่าว สามารถมี Sub-Type ได้ หมายถึง สามารถสร้าง structure ที่เป็นลำดับชั้น (Hierarchy of Structured Types) ร่วมกับ Structured Type อื่น ๆ โดยใช้ความสัมพันธ์ในรูปของ Super-Type และ Subtype ซึ่งช่วยให้ง่ายต่อการออกแบบเพื่อรองรับลักษณะข้อมูลและความสัมพันธ์ต่าง ๆ ของระบบงานที่มีอยู่จริง Structured Type สามารถสร้างโดยใช้คำสั่ง Create Type และใช้นำเสนอในรูปของ Column Type หรือ Row Type ก็ได้ ตัวอย่างการสร้าง Structured Type

```
CREATE TYPE Person_t AS
    (name        VARCHAR(40),
    birthyear    INTEGER)
        REF USING INTEGER
        MODE DB2SQL;
```

ตัวอย่างการสร้าง Typed Table

```
CREATE TABLE Person OF Person_t
        (REF IS oid USER GENERATED);
INSERT INTO PERSON (OID, NAME, BIRTHYER) VALUES (PERSON_T(1),'JOHN',1771);
```

หรือ INSERT INTO PERSON VALUES (PERSON_T (1), 'JOHN', 1771);

ทุก Typed Table จะมี column พิเศษ ที่ถูกสร้างขึ้นมาเป็น column แรกเรียกว่า OID column โดยค่าใน column นี้จะมีลักษณะ Unique คือค่าไม่ซ้ำกัน ลักษณะของข้อมูล OID column เป็น REFERENCE

3.1.4 การสร้าง Structured Type

Structured Type สนับสนุนโครงสร้างแบบลำดับชั้น (Hierarchical Structured) ซึ่งสามารถสร้างในรูปของ Sub-Type ของ Structured Types อื่น ๆ ได้ ซึ่งจะได้รับการถ่ายทอดแอททริบิวซ์ต่าง ๆ ตามคุณสมบัติที่เรียกว่า Inheritance

"AS" ใช้กำหนด, attribute ต่างๆ ที่เกี่ยวข้องกับ Type

"UNDER" ใช้กำหนดในฐานะของ Subtype ของ Structured Type อื่น

"REF USING" ใช้กำหนดเมื่อ Structured Type นั้น เป็น Root Type และใช้นำเสนอแบบ Reference

Type Table จะต้องมี OID Column และเมื่อต้องการกำหนดค่าให้แก่ OID จะต้องทำให้อยู่ในรูป Casting Function ซึ่งกำหนดจาก Ref Using ใน Root Type สำหรับค่า Reference หรือค่า OID ดังกล่าว สามารถกำหนดโดยใช้ประเภทข้อมูลพื้นฐานได้แก่ INTEGER, SMALLINT, BIGINT, DECIMAL, CHAR, VARCHAR, GRAPHIC, หรือ VARGRAPHIC โดย Default จะเป็น VARCHAR (16) FOR BIT DATA

"MODE" ใช้กำหนดรูปของคำสั่ง Type โดย DB2 สนับสนุน MODE ที่เรียกว่า DB2SQL

```
CREATE TYPE Person_t AS
   (Name       VARCHAR(20),
    Birthyear  SMALLINT)
    REF USING INTEGER
    MODE DB2SQL;
CREATE TYPE Emp_t UNDER Person_t  AS
```

```
    (Salary INT)
    MODE DB2SQL;
CREATE TYPE Student_t UNDER Person_t as
    (Major VARCHAR(10),
    Archivm DECIMAL(5,2))
    MODE DB2SQL;
```

3.1.5 การเปลี่ยนแปลงโครงสร้าง (Altering Structured Type)

คำสั่ง ALTER TYPE ใช้สำหรับแก้ไข Structured type ที่สร้างมาก่อนแล้ว โดยใช้คำสั่ง Add Attribute ในการเพิ่ม Attribute ใหม่ และใช้คำสั่ง Drop Attribute ในการลบ Attribute ที่ไม่ต้องการออกจาก Structured Type การแก้ไขโครงสร้างโดยใช้คำสั่ง Alter Type จะไม่สามารถทำได้ถ้า Structured Type ดังกล่าวถูกนำมาใช้สร้างตาราง (Typed Table) เช่น

```
ALTER TYPE Person_t ADD ATTRIBUTE Tel CHAR(12);
ALTER TYPE Person_t DROP ATTRIBUTE Tel;
```

3.1.6 การสร้างตาราง Typed tables

Typed Table เป็นตารางที่ถูกสร้างขึ้นโดยใช้ Structured Type โดย Typed Table อาจอยู่ในรูปของโครงสร้างแบบลำดับชั้น (Hierarchical of Structured Type) ซึ่ง Sub-Table จะได้รับการถ่ายทอด Attribute ต่าง ๆ จาก Super-Table หาก Table ใด ๆ ไม่มี Super-Table เรียกว่า "Root-Table"

```
CREATE TABLE Person OF Person_t
        (REF IS OID USER GENERATED);
CREATE TABLE Emp OF Emp_t UNDER Person
        INHERIT SELECT PRIVILEGES;
CREATE TABLE Student OF Student_t UNDER Person
        INHERIT SELECT PRIVILEGES;
```

ทุก Typed Table จะมี Column OID เสมอ โดย OID จะถูกกำหนดโดย Root Table ส่วน Sub-Table จะได้รับการถ่ายทอด OID มาด้วย (ชนิดข้อมูลใน OID Column คือ Reference

"User Generated" ใช้กำหนดเมื่อต้องการกำหนดค่าให้กับ OID Column โดยจะกำหนดทุกครั้งเมื่อมีการ Insert ข้อมูลใหม่ โดยค่า OID ดังกล่าวไม่สามารถแก้ไขได้

"Inherit Select Privileges" หมายถึงการถ่ายทอดสิทธิในการใช้คำสั่ง Select ที่มีอยู่ใน Super-Table ให้กับ Sub-Table ที่ถูกสร้างใหม่

3.1.7 การลบ Typed Table (Dropping Typed Table)

การลบ Typed Table จะใช้คำสั่ง Drop Table เหมือนการลบตารางปกติ โดยจะต้องแน่ใจว่าไม่มี Sub-Table ใดๆ ผูกอยู่กับ Typed Table ที่ต้องการลบ เมื่อมีการลบ Sub-Table จะส่งผลให้มีการลบข้อมูลของ Sub-Table ที่อยู่ใน Super-Table ออกทั้งหมด ถ้าต้องการลบ ตารางในโครงสร้างระดับชั้นทั้งหมด (Hierarchical Table) จะใช้คำสั่ง DROP TABLE HIERARCHY เช่น Drop Table Hierarchy Person

3.1.8 การเพิ่มข้อมูล (Inserting Typed Table)

การเพิ่มข้อมูลใช้คำสั่ง Insert เมื่อจะกำหนดค่าลง OID Column หรือ Reference Type ใดๆ จะต้องใช้วิธีการแปลงค่าให้อยู่ในรูปของ Reference Type นั้น ๆ ก่อน เรียกว่า Casting Function โดย argument ของ Function จะเป็น Built-in Data Type ที่กำหนดจาก "Ref Using" (ชื่อ Casting Function จะเหมือนชื่อ Structure Type)

```
INSERT INTO Person(oid,name birthyear)
    VALUES (Person_t(10),'John',1968) โดย Person_t() คือ Casting Function
INSERT INTO Emp(oid,name  birthyear,salary)
        VALUES (Emp_t(30),'Pat',1970,60000)
INSERT INTO Student(oid,name birthyear,major,archivm)
        VALUES(Student_t(30),'Franz',1976,'pol',2.5)
```

3.1.9 การเลือก ROW จาก Typed Table (Selecting a row from a Typed Table)

การเลือกข้อมูลเพื่อแสดงผล จะใช้คำสั่ง Select เช่นเดียวกับตารางทั่วไป แต่ในกรณีที่ตารางเป็น Hierarchy จะมีข้อแตกต่าง กล่าวคือ

```
Select * from (Type-Table)
```

จะได้ข้อมูลในตารางโครงสร้างทั้งหมดลงไป (ถ้าเลือกจากตารางที่เป็น Root-Table จะได้ข้อมูลทั้งหมดที่เป็นโครงสร้างระดับชั้น)

```
Select * from ONLY(Type-Table)
```

จะได้เฉพาะข้อมูลที่อยู่ในตารางที่ระบุ

Select * from OUTER(Root-Table)

จะได้ข้อมูลทุกแอททริบิวซ์ที่อยู่ในแต่ละตารางในโครงสร้างระดับชั้น

ข้อสังเกต เมื่อมีการสร้างตารางที่มีลักษณะเป็น Hierarchy เมื่อใด DB2 จะสร้างตารางใหม่ที่จัดเก็บทุกแอททริบิวซ์ของแต่ละตารางในโครงสร้างระดับชั้นนั้น ๆ เรียกว่า H-Table หรือ Hierarchy Table โดยตารางนี้ จะไม่สามารถแก้ไขหรือปรับปรุงได้ด้วยคำสั่ง SQL Statements วัตถุประสงค์สำหรับตารางนี้ เพื่อเพิ่มประสิทธิภาพในการเข้าถึงข้อมูล โดยข้อมูลในตาราง H-Table จะมีโครงสร้างที่สำคัญกล่าวคือ Column แรก จะเป็น TYPE_ID เป็นส่วนที่เก็บค่าของ Structure-Type ของ Typed Table (ข้อมูลที่จัดเก็บนี้ จะได้จาก DB2 System Catalog tables ชื่อ SYSCAT.TABLES และ SYSCAT.DATATYPES

3.1.10 Reference Columns

Reference Typed Table หรือ Typed Table โดย Reference Column จะเก็บค่า OID (ชนิดข้อมูลของ Reference Column คือ REFERENCE) โดย Reference คล้ายกับ Foreign Key ต่างกันตรงที่ไม่สามารถ Insert Update Delete ได้ เมื่อมีการสร้างตารางและมีการกำหนด Reference Column จะต้องมีการกำหนดคำสั่ง Scope ไปด้วยทุกครั้ง

DEPT

OID	NAME	LOCATION
45	ADV	DALLAS
2	ADM	AUSTIN

EMP

OID	NAME	SALARY	DEPTREF
23	JOHN	45000	2
67	MEL	60000	88

รูปที่ 3. 1 ตารางที่ใช้ Reference Data Type

```
CREATE TYPE Dept_t AS
        (Name    CHAR (40),
        Location CHAR (20))
        REF USING INTEGER MODE DB2SQL;
CREATE TYPE Emp_t UNDER Person_t AS
```

```
        (Salary INTEGER,
        Deptref REF(Dept_t))
        MODE DB2SQL;
    CREATE TABLE Dept OF Dept_t
        (REF IS oid USER GENERATED);
    CREATE TABLE Emp OF Emp_t UNDER Person
        INHERIT SELECT PRIVILEGES
        (Deptref WITH OPTIONS SCOPE Dept);
```

จากตัวอย่างในตาราง Emp มี Reference Column ชื่อ Deptref อ้างไปที่ Dept เมื่อสร้างตาราง จะต้องมีการกำหนด with options scope เพื่อใช้อ้างถึงข้อมูลที่มีการ Reference

เมื่อต้องการอ้างถึงข้อมูลที่อยู่ใน Reference Table จะใช้เครื่องหมาย -> (Dereference Operator) เช่น

```
    SELECT E.name from emp E
        WHERE E.Deptref -> location = 'AUSTIN';
```

โดย Location เป็น Attribute ที่อยู่ในตาราง Dept

ตัวอย่างการสร้าง Structured Type

```
    CREATE TYPE Person_t As
        (Name  VARCHAR(20),
        Birthyear  SMALLINT)
        REF USING INTEGER MODE DB2SQL;
    CREATE TYPE Emp_t UNDER Person_t AS
       (Salary  INT)
        MODE DB2SQL;
CREATE TYPE Student_t UNDER Person_t AS
    (Major  CHAR(20),
    Archivm  DECIMAL(5,2))
       MODE DB2SQL;
    CREATE TYPE Dept_t  AS
        (Name   VARCHAR(20),
```

```
        Budget  INT,
        Mgr  REF(Emp_t))
        REF USING INTEGER MODE DB2SQL;
    ALTER TYPE Emp_t ADD ATTRIBUTE DEPT REF(Dept_t);
    CREATE TYPE Prof_t UNDER Emp_t AS
        (Speciality  VARCHAR(20))
        MODE DB2SQL;
```

ตัวอย่างการสร้าง Typed Table

```
    CREATE TABLE Person OF Person_t
        (REF IS oid USER GENERATED);
    CREATE TABLE  Emp OF Emp_t UNDER Person
        INHERIT SELECT PRIVILEGES;
    CREATE TABLE Student OF Student_t UNDER Person
        INHERIT SELECT PRIVILEGES;
    CREATE TABLE Dept OF Dept_t
        (REF IS oid USER GENERATED, Mgr with options scope Emp);
    CREATE TABLE Emp ALTER COLUMN Dept ADD SCOPE Dept;
    CREATE TABLE Prof OF Prof_t UNDER Emp
        INHERIT SELECT PRIVILEGES;
```

3.1.11 กฎเกณฑ์การสร้าง USER DEFINED STRUCTURE TYPE

3.1.11.1 คำสั่ง Create Type ชื่อไทป์ AS (แอททริบิวซ์ ชนิดข้อมูลของแอททริบิวซ์, ...รายชื่อแอททริบิวซ์และชนิดข้อมูล) โดยจะอยู่ภายในเครื่องหมายวงเล็บ ใช้เครื่องหมายคอมมา เพื่อแยกแอททริบิวซ์แต่ละชนิด

3.1.11.2 UNDER ชื่อ UDT ใช้กรณีที่ต้องการระบุ Super-Type โดย IBM DB2 UDB สนับสนุน Single Inheritance

3.1.11.3 REF USING ใช้เมื่อ TYPE นั้น เป็น ROOT-TYPE เท่านั้น

3.1.11.4 MODE DB2SQL ต้องระบุเสมอที่ท้ายประโยคคำสั่ง

3.1.11.5 เมื่อต้องการอ้างอิงกับ TYPE อื่น จะใช้ REF (ชื่อ UDT) เมื่อต้องการแก้ไข TYPE ที่สร้างก่อนแล้ว ใช้คำสั่ง Alter Type โดยใช้ ADD ATTRIBUTE ชื่อแอททริบิวซ์ ชนิดข้อมูลของแอททริบิวซ์ เมื่อต้องการเพิ่มแอททริบิวซ์ และใช้ DROP ATTRIBUTE ชื่อแอททริบิวซ์สำหรับการลบ

3.1.12 กฎเกณฑ์การสร้าง TYPED TABLE

3.1.12.1 คำสั่ง Create Table ชื่อ Typed-Table OF ชื่อ UDT

3.1.12.2 ระบุ OF ชื่อ UDT เพื่อกำหนด UDT ที่ต้องการนำมาใช้สร้างตาราง

3.1.12.3 UNDER Typed-Table จะใช้เมื่อตารางที่กำลังสร้างเป็น Sub-Typed-Table

3.1.12.4 ระบุ with options scope เพื่อให้สามารถอ้างถึงข้อมูลในตารางที่ Reference ไปถึง (หรืออาจระบุได้ภายหลัง โดยการใช้คำสั่ง ALTER ได้)

3.1.12.5

ส่วนข้อกำหนด เขียนภายใต้เครื่องหมายวงเล็บ ประกอบด้วย

3.1.12.6 REF IS ชื่อแอททริบิวซ์ที่เป็น OID ใช้เมื่อเป็น ROOT-TYPED-TABLE เสมอ และตามด้วย USER GENERATED ซึ่งเป็นโหมดการสร้าง OID

3.1.12.7 Constraints ต่าง ๆ ระบุได้ภายใต้เครื่องหมายวงเล็บ โดยเขียนต่อจาก REF IS ... โดยใช้เครื่องหมายคอมมาคั่นระหว่างคอนเสตรนท์แต่ละตัว

3.1.12.8 INHERIT SELECT PRIVILEGES เมื่อตารางที่สร้างเป็น Sub-Typed-Table อยู่นอกวงเล็บ โดยปกติจะกำกับไว้ท้ายคำสั่ง

3.1.12.9 การแก้ไขข้อกำหนดต่าง ๆ ของ TABLE ที่สร้างแล้ว จะใช้ ALTER TABLE ชื่อ TABLE เป็นคำสั่งที่ใช้แก้ไข

3.1.12.10 ALTER COLUMN ชื่อ Column และตามด้วย “ADD” เมื่อต้องการเพิ่ม Constraints ต่าง ๆ ที่ต้องการ เช่น ALTER TABLE Emp ALTER COLUMN Dept ADD SCOPE Dept เป็นต้น

3.1.12.11 ALTER COLUMN ชื่อ Column และตามด้วย “DROP” ชื่อคอลัมน์เมื่อต้องการลบ Constraints ต่าง ๆ ที่ต้องการ

3.1.13 การกำหนดแอททริบิวซ์ที่เป็น Embedded

3.1.13.1 เมื่อกำหนดแอททริบิวซ์ ไม่ต้องระบุ Ref กำกับ

3.1.13.2 การ Insert จะต้องใช้ Typed Constructor ในการเพิ่มข้อมูล เช่น INSERT INTO Person values ('John',Address_T()..city('Las Vegas')..zip(81200));

3.1.13.3 การ Update, Select ใช้ชื่อแอททริบิวซ์ในการอ้างถึงข้อมูล เช่น SELECT Name, Home..city, Home..zip from Person;

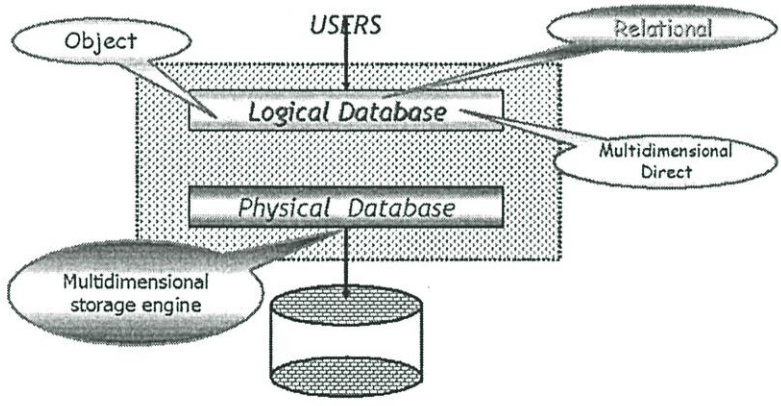
Update Person set Home=Home..City('Santa') where Home..zip=90102;

3.1.14 การกำหนดแอททริบิวซ์ที่เป็น Reference

ในตารางที่มีการใช้ Reference แอททริบิวซ์ที่อ้างและใช้ Reference จะต้องระบุ "REF" Keyword และระบุ Constraint "with options scope" กำกับตาราง เพื่ออ้างถึงข้อมูลในแต่ละแอททริบิวซ์ที่อยู่ในตารางที่อ้างถึงปลายทางด้วยเสมอ

3.2 Caché Object Database

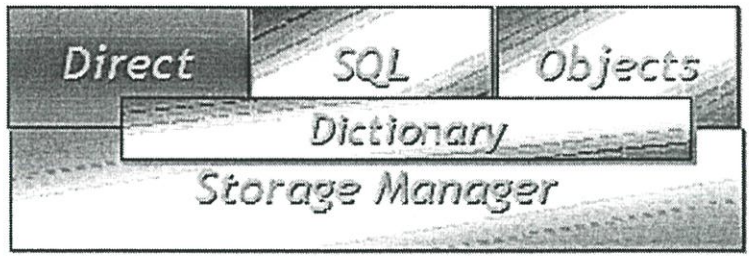
Caché ,"คาเช่" ระบบปฏิบัติการฐานข้อมูลเชิงวัตถุ (Object Database) พัฒนาโดย InterSystems สหรัฐอเมริกา เป็นระบบฐานข้อมูลที่คุณสมบัติมากกว่าระบบรีเลชั่นแนลแบบเดิม คาเช่จัดเป็นระบบฐานข้อมูลเชิงวัตถุ ที่มีความสามารถตัวหนึ่ง โดยปกติ เวลาจะติดต่อกับฐานข้อมูลโมเดลใด ก็จะติดต่อทาง Logical View ของโมเดลนั้น ๆ เช่น ถ้าเป็น Relational โมเดล ส่วนที่เป็น Logical View ก็จะเป็นตาราง หรือกล่าวได้ว่า สิ่งที่ผู้ใช้เห็นคือ Logical View ของฐานข้อมูลโมเดลนั้น ในทางปฏิบัติสามารถเข้าถึงข้อมูลในคาเช่ได้ถึงสามทางด้วยกัน (ดังรูป) เนื่องจากคาเช่ออกแบบมารองรับผู้ใช้ที่มีความหลากหลาย หมายความว่า คาเช่ สามารถมองข้อมูลเป็นออบเจคก็ได้ รีเลชั่นแนลก็ได้ และมองเป็นมัลติไดเมนชั่นก็ได้เช่นกัน หรืออีกนัยหนึ่งกล่าวได้ว่า คาเช่ มีลอจิคัลวิว 3 ทาง คือ Object, Relational, Multidimensional เราสามารถมองข้อมูลด้วยวิธีใดก็ได้ โดยได้ผลลัพธ์ที่ถูกต้องเหมือนกัน ในส่วนที่เป็น Physical คาเช่พัฒนารูปแบบการเก็บข้อมูลรูป Multidimensional Array



รูปที่ 3. 2 โมเดลข้อมูลของคาเช่

อย่างไรก็ดี ลักษณะนี้ ไม่จัดเป็น Object Relational DBMS เนื่องจาก ถ้าเป็น Object Relational DBMS จะต้องสร้าง Typed ขึ้นมาก่อน จากนั้นจึงจะสร้าง Table ที่เรียกว่า Typed Table ซึ่งจะใช้เก็บข้อมูล อาจอยู่ในรูปแบบ Column Type หรือ Row Type ก็แล้วแต่จะออกแบบ จากนั้นจะต้องมีการเชื่อมโยงเข้าหากัน (Synchronize)

3.2.1สถาปัตยกรรมข้อมูลของคาเช่



รูปที่ 3. 3 เทคโนโลยี Unified Data Architecture

คาเช่ อนุญาตให้เราออกแบบและจัดการข้อมูลได้ทั้งแบบออบเจค และแบบรีเลชั่นแนล โดยมีเครื่องมือสำหรับออกแบบให้ในแบบออบเจค ชื่อ Caché Studio ซึ่งเป็นเครื่องมือสำหรับจัดการโครงสร้างข้อมูลและเขียนชุดคำสั่ง โดยเราสามารถสร้างโครงสร้างข้อมูลโดยผ่านเครื่องมือนี้ จากนั้นคาเช่จะทำการคอมไพล์และสร้าง Data Dictionary ตัวเดียว ที่อนุญาตให้เข้าถึงข้อมูลได้ 3 ทาง คือ Direct Multidimensional, SQL, และ Object โดยเทคโนโลยีนี้เรียกว่า Unified Data Architecture หรือ UDA ที่ทำให้ 1 Database จะมีเพียง 1 Data Dictionary แต่สามารถเข้าถึงข้อมูลได้ 3 ทางนั่นเอง ส่วนการจัดเก็บข้อมูลทางกายภาพ จะเป็นการจัดเก็บแบบ Multidimensional Array

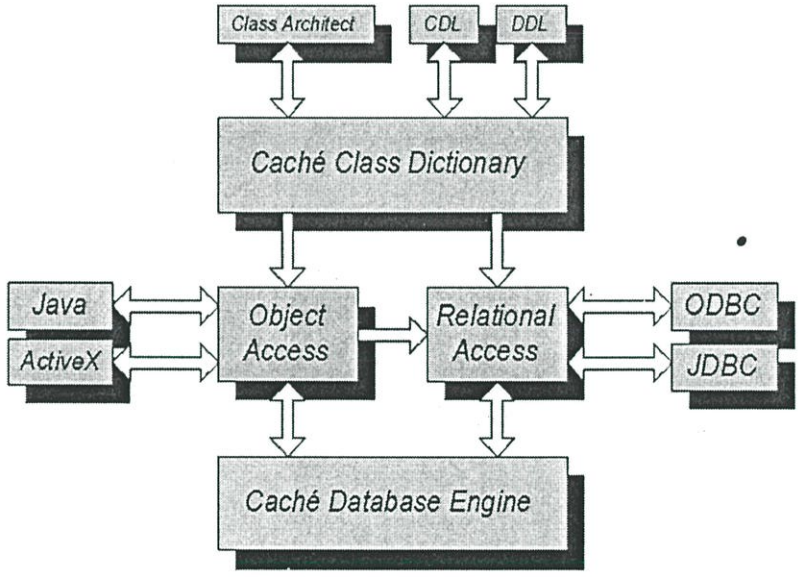
แม้ว่า คาเช่ จะให้ทางเลือกในการออกแบบทั้งสองวิธี แต่ผลลัพธ์ที่ได้ย่อมมีข้อแตกต่างกัน กล่าวคือ ถ้าออกแบบในแบบโมเดลออบเจค คาเช่จะทำการสร้างตารางให้โดยอัตโนมัติตามหลักการทำนอมัลไลเซชั่น เช่น กรณี Multi Values ถ้าเป็นรีเลชั่นแนล จะต้องสร้างเป็นอีกตาราง คาเช่ก็จะสร้างให้โดยอัตโนมัติ

รูปที่ 3. 4 ผลลัพธ์ที่ได้จากการสร้างด้วยโมเดลเชิงวัตถุ

แต่ถ้าออกแบบในลักษณะรีเลชั่นแนล ผลลัพธ์ที่ได้จะเหมือนกับการสร้างตาราง คือ 1 ตาราง คือ 1 คลาส ซึ่งเหมาะกับการถ่ายโอนข้อมูลจากโมเดลรีเลชั่นแนลเดิมเข้ามาสู่คาเช่

รูปที่ 3. 5 ผลลัพธ์ที่ได้จากการสร้างด้วยโมเดลเชิงสัมพันธ์

ไม่ว่าจะสร้างด้วยวิธีใดก็ตาม การเข้าถึงข้อมูล ก็จะเป็นข้อมูลตัวเดียวกันเสมอ ในส่วนของการสร้างโครงสร้างได้จากเครื่องมือ ชื่อว่า Caché Studio หรืออาจสร้างโดยการ Import แฟ้ม CDL (Caché Definition Language) หรืออาจเป็นแฟ้ม DDL (Data Definition Language) ก็ได้ จากนั้นคาเช่จะคอมไพล์และสร้าง Data Dictionary ตัวเดียว จากนั้น ผู้ใช้สามารถเลือกได้ว่า จะเข้าถึงข้อมูลได้โดยวิธีใด สุดแล้วแต่ผู้ใช้จะพิจารณา ทำให้การพัฒนาแอพพลิเคชั่นสะดวกยิ่งขึ้น



รูปที่ 3. 6 แสดงการเข้าถึงข้อมูลด้วยวิธีการต่าง ๆ

3.2.2 มาตรฐานของคาเช่

คาเช่ สนับสนุนมาตรฐาน ODMG และ ANSI SQL นอกจากนี้การใช้ภาษา SQL คาเช่จะมี Optimizer เพื่อช่วยหากทางเลือกในการเข้าถึงข้อมูลได้เร็วและมีประสิทธิภาพ ในด้าน Recovery Control คาเช่สนับสนุนการทำ Before Image Journal (BIJ) ที่ใช้สำหรับการทำ Undo และ After Image Journal (AIJ) ซึ่งใช้สำหรับการทำ Redo นอกจากนี้ หากพิจารณาในด้านของ Concurrency Control หรือการใช้งานพร้อมกัน คาเช่สนับสนุน ISOLATION LEVEL ซึ่งเป็นมาตรฐานที่ใช้ควบคุมความ

ถูกต้องของการใช้ข้อมูลในระดับ Read Committed หรือ Cursor Stability ซึ่งช่วยแก้ปัญหาในเรื่อง Uncommitted Dependency Problem ได้

3.2.3ประเภทของแอททริบิวซ์

3.2.3.1 Data Type Attributes

ได้แก่ แอททริบิวซ์ประเภทข้อมูล โดยสามารถใช้เมดธอดในการตรวจสอบ (Validation), และสามารถกำหนด Parameter Constraints ต่าง ๆ ได้ เช่น MaxLen คือการกำหนดความยาวสูงสุดของข้อมูล

3.2.3.2 Reference Attributes

ได้แก่ แอททริบิวซ์ ที่บันทึกค่า ID หรือ OID ที่อ้างถึงวัตถุที่เกี่ยวข้อง

3.2.3.3 Embedded Object Attributes

ได้แก่ กลุ่มของแอททริบิวซ์ ที่ออกแบบให้มีลักษณะ Serialize Objects คือ เมื่อจะใช้งานจะต้องนำแอททริบิวซ์กลุ่มนี้ ไปรวม (Embedded) ไว้ในคลาสหลัก (Class) โดยการกำหนดคอลัมน์ประเภทข้อมูลในคลาสหลัก ให้เป็นคลาสแบบฝัง (Embedded Class) ซึ่งสนับสนุนข้อมูลที่มีลักษณะซับซ้อน (Complex Data)

3.2.3.4 Reference Attributes

ได้แก่ แอททริบิวซ์ที่จัดเก็บค่าที่อ้างถึงวัตถุอื่น ๆ ที่เกี่ยวข้อง โดยการจัดเก็บค่า ID หรือ OID ของวัตถุที่ถูกอ้างถึง (referenced object)

3.2.3.5 Array Attributes

ได้แก่ แอททริบิวซ์ที่มีลักษณะเป็น Collection คือ สามารถเก็บข้อมูลได้หลายค่าในแอททริบิวซ์ (Multi values) โดยลักษณะการเก็บจะอยู่ในรูปแบบ Child Table ประกอบด้วยค่า ID ของวัตถุในคลาสหลัก (Parent's Object), Element Key คือ ค่า Identifier ของค่าวัตถุ, Siblings ได้แก่ ค่าของข้อมูล (array member) และถ้าค่าของข้อมูลในคลาสหลัก เป็นค่าว่าง (one that contains no elements) ค่าของ ID จะไม่ปรากฏในคลาสลูก

3.2.3.6 Stream Attributes

ได้แก่ แอททริบิวซ์ ที่ใช้เก็บข้อมูลที่มีลักษณะต่อเนื่อง และมีความยาวที่ไม่แน่ชัด ลักษณะข้อมูลชนิดนี้ แบ่งออกเป็น 2 ประเภท ได้แก่ character stream และ binary stream โดยจะจัดเก็บในลักษณะ LOB (Large Objects)

3.2.3.7 Methods

ได้แก่ ชุดคำสั่งที่ถูกเขียนขึ้นและรวมไว้ในคลาส

3.2.3.8 Query

เป็นเมดธอดอีกชนิดหนึ่ง ที่อยู่ในรูปแบบ SQL STATEMENT

3.2.4 การพัฒนาแอพพลิเคชั่น

ภาษาที่ใช้ในคาเช่ สามารถใช้ได้หลายภาษา คือ Caché Object Script ที่มีต้นแบบมาจาก Mumps หรืออาจใช้ภาษา SQL ก็ได้ และล่าสุด คุณสามารถเขียนภาษาเบสิค (Basic Script) ได้ด้วยเช่นกัน ทั้งนี้ จากข้อมูลของ IDC พบว่า ภาษาที่ทั่วโลกนิยมใน 5 อันดับต้น ๆ ภาษาวิชวลเบสิค เป็นภาษาที่ได้รับความนิยมสูง ดังนั้น คาเช่จึงได้พัฒนาให้ใช้ได้ทั้งสามภาษา โดยสามารถเขียนด้วยภาษาใดก็ได้ หรือจะเขียนร่วมกันก็ได้ ไม่มีความแตกต่างด้านประสิทธิภาพ เนื่องจากคาเช่จะแปลให้อยู่ในรูป Native Code ทุกครั้ง

3.3 การเปรียบเทียบคุณสมบัติเชิงวัตถุระหว่าง DB2 และ Caché

ระบบปฏิบัติการฐานข้อมูล มีรายละเอียดเปรียบเทียบ ดังแสดงดังต่อไปนี้

ตารางที่ 3. 1 แสดงการเปรียบเทียบคุณสมบัติของระบบฐานข้อมูลที่ใช้ในงานวิจัย

รายละเอียด	DB2	Caché
ผู้ผลิต	IBM	InterSystems
ประเภทฐานข้อมูล	Object Relational	Object Oriented
Platform	Windows, Linux, Unix, IBM OS	Windows, Linux, Unix
มาตรฐานที่รับรอง	SQL3 (X3H2)	ODMG 2.0 (X3H7)
ภาษาที่ใช้สืบค้น	SQL	Caché Object Script, SQL, Basic
Logical View	Relational	Object, Relational, Multidimensional
Physical View	B-Tree	Multidimensional Array
Transaction Recovery	สนับสนุน AIJ, BIJ	สนับสนุน AIJ, BIJ

ตารางที่ 3. 2 (ต่อ)

รายละเอียด	DB2	Caché
SQL 92 ISOLATION LEVEL	Serializable, Repeatable Read, Read Committed, Read Uncommitted	Read Committed (Cursor Stability), Read Uncommitted
Backup Type	Volume Backup, Cumulative Backup	Volume Backup, Cumulative Backup
Large Object	BLOB, CLOB, DBLOB	Binary Stream, Character Stream
Query Optimization	Cost Based Optimizer	Cost Based Optimizer
Index	B-Tree	B-Tree
Index Type	Dense Index, Bitmap, Cluster Bi-directional	Sparse Index, Bitmap
Encapsulation	Break	Full
Inheritance	Single	Multiple Inheritance
User Defined Function	สนับสนุน โดยภาษา SQL, Java, C, C++	สนับสนุน โดยภาษา Caché Object Script, SQL, Basic
User Defined Method	สนับสนุน โดยภาษา SQL, Java, C, C++	สนับสนุน โดยภาษา Caché Object Script, SQL, Basic
User Defined Structure Type	สนับสนุน	สนับสนุน
User Defined Reference Type	สนับสนุน	สนับสนุน
Object Identifier (OID)	User Generated	System Generated
Object Identifier Data Type	String, Numeric	Numeric
Collection Type	ไม่สนับสนุน	Array, List, Relationship
Composite Attribute	สนับสนุน	สนับสนุน
Import/Export	Import, Export, Load	Import, Export
OID Imported	ไม่สนับสนุนโดยเครื่องมือ	สนับสนุน

บทที่ 4
หลักและวิธีการแปลงโครงสร้างข้อมูล

4.1 ลักษณะโครงสร้างข้อมูลที่จะใช้แปลง

ในกระบวนการแปลง จะใช้โครงสร้างข้อมูลจากระบบเดิม เป็นข้อมูลนำเข้าจากระบบฐานข้อมูลคาเช่ ซึ่งมี 2 รูปแบบ คือ แบบที่หนึ่ง เรียกว่า Caché Definition Language (CDL) และแบบที่สอง เป็นรูปแบบใหม่ จัดเก็บในรูป XML ซึ่งโครงสร้างข้อมูลทั้งสองรูปแบบนี้ จะแสดงรายละเอียดเกี่ยวกับ คลาส, แอททริบิวซ์, และเมดธอดต่าง ๆ ดังแสดงในรูปที่ 4.1 และ 4.2 ตามลำดับ

```
Property Home As User.Address;
Property Name As %Library.String [ Required ];
Property Phone As %Library.String;
Property Picture As %Library.Stream [ Collection = binarystream ];
Property SSN As %Library.String [ Required ];
Property Sex As %Library.String(DISPLAYLIST = ",Male,Female,ชาย,หญิง", MAXLEN = 1, VALUELIST = ",M,F,ช,ญ") [ Required ];
Property Spouse As User.Person;
Property TotalBill As User.DtCurrency;
Index NameDOBSSN On (Name, DOB, SSN);
Method AgeGet() As %Library.Integer
{
 if ..DOB="" quit ""
 quit $h-..DOB\365.25
}
```

รูปที่ 4. 1 ตัวอย่างโครงสร้างข้อมูลจากคาเช่ แบบ CDL

```
<?xml version="1.0" encoding="UTF-8" ?>
- <Export generator="Cache" version="9" zv="Cache for Windows NT (Intel) 5.0 (Build 462)" ts="2003-03-13
    15:02:59">
- <Class name="User.Dog">
  <ClassType>persistent</ClassType>
  <Super>%Library.Persistent</Super>
  <TimeChanged>59172,489.076</TimeChanged>
  <TimeCreated>58923,42926</TimeCreated>
+ <Storage name="Default">
+ <Property name="DogBreed">
+ <Property name="DogName">
+ <Property name="DogPicture">
+ <Index name="DogNameIndex">
+ <Method name="Populate">
+ <Query name="ByName">
+ <Query name="Dogs">
    </Class>
  <Checksum value="3176747157" />
    </Export>
```

รูปที่ 4. 2 ตัวอย่างโครงสร้างข้อมูลจากคาเช่ แบบ XML

4.2 หลักเกณฑ์การแปลง

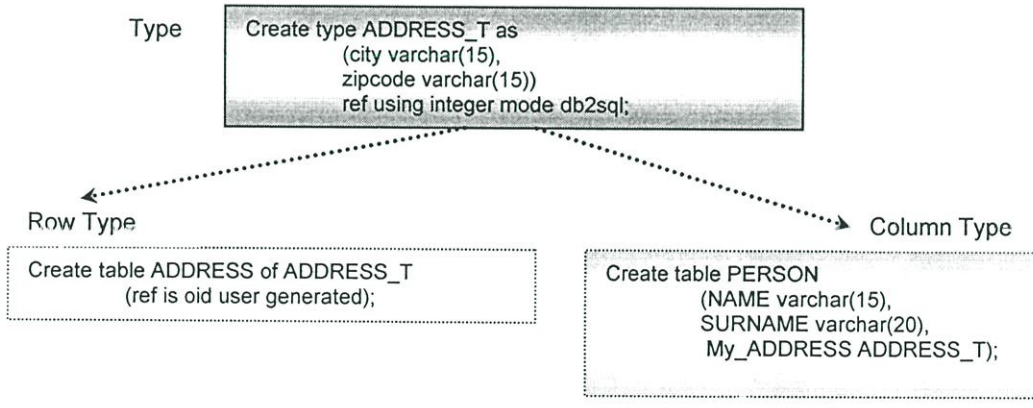
ในงานวิจัย ใช้โครงสร้างข้อมูลที่จัดเก็บในรูปแบบ XML ทั้งนี้ เนื่องจากเป็นมาตรฐานที่เป็นที่ยอมรับ และง่ายต่อการจัดการและตรวจสอบข้อมูล โดยโครงสร้างดังกล่าว จะประกอบด้วยโหนดและแอททริบิวซ์ต่าง ๆ มากมาย ซึ่งการแปลงจะวิเคราะห์โหนดและแอททริบิวซ์ที่เกี่ยวข้อง และแปลงให้เป็นคำสั่งในรูปแบบ DB2 โดยใช้เกณฑ์การพิจารณาต่อไปนี้

4.2.1 กรณี Class

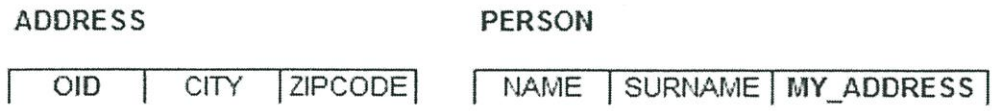
CLASS จะต้องสร้างเป็น Typed ใหม่ก่อนเสมอ จากนั้นจึงจะพิจารณาว่าจะต้องสร้างเป็นรีเลชั่น หรือ Extent หรือไม่ กล่าวคือ ถ้า <ClassType> เป็น serial แสดงว่าคลาสนี้ มีลักษณะเป็น Embedded Class ให้สร้าง Typed ใหม่ แต่ไม่ต้องสร้างรีเลชั่น เนื่องจาก Class นี้ จะมีลักษณะแบบ Column Typed และในทางกลับกัน ถ้า <ClassType> เป็น persistent แสดงว่า Class มีลักษณะเป็น Persistent Class ให้สร้าง Typed ใหม่ และสร้างรีเลชั่นใหม่ด้วย ลักษณะแบบนี้ เรียกว่า Row Typed

```
<Class name="User.Dog">
<ClassType>persistent</ClassType>
```

รูปที่ 4. 3 แสดงชื่อคลาสและประเภทของคลาส



รูปที่ 4. 4 แสดงลักษณะ Row Type และ Column Type



รูปที่ 4.5 ลักษณะตารางที่ได้จากการสร้างตารางประเภท Row Type และ Column Type

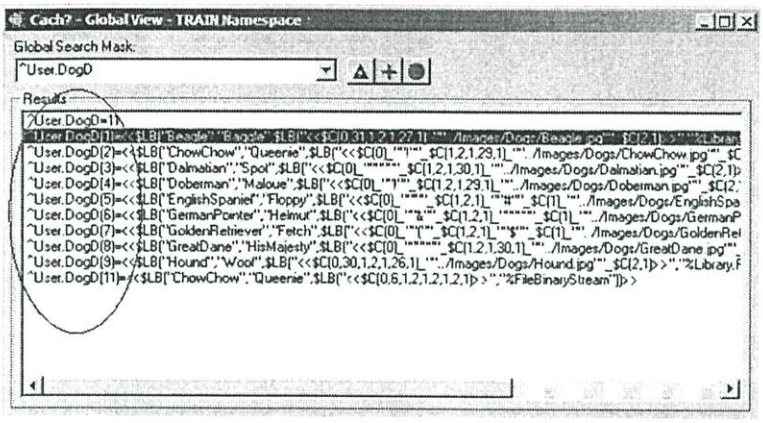
หมายเหตุ คำสั่งใน DB2 จะใช้แตกต่างกันระหว่างชนิดที่เป็น ROW TYPE และ COMPOSITE TYPE กล่าวคือ

- กรณีที่เป็น ROW TYPE จะต้องระบุ REF USING กำกับไว้ด้วย
- แต่ถ้าเป็น COMPOSITE TYPE ไม่ต้องระบุ REF USING กำกับไว้

4.2.2 กรณี Object Identified (OID)

ในระบบฐานข้อมูลคาเช่ ค่า OID จะกำหนดโดยระบบ และมีลักษณะเป็นเลขลำดับเรียงลำดับจากน้อยไปมาก และค่าจะถูกกำหนดโดยอัตโนมัติ ในขณะที่ DB2 ก็สนับสนุนการใช้ OID แต่จะเป็น User Generated นั่นคือ กำหนดโดยผู้ใช้งาน ดังนั้น เพื่อให้มีลักษณะการเพิ่มค่าแบบเดียวกัน เราสามารถใช้ฟังก์ชั่น Generate Unique() ช่วยในการกำหนดค่า OID ดังกล่าวได้ หรือ อาจกำหนดโดยระบุค่าโดยตรงก็ได้เช่นกัน

Object Identifier (OID) OID ของ ODBMS แต่ละค่าย มีรูปแบบการจัดเก็บ OID แตกต่างกัน รวมถึงความยืดหยุ่นในการอนุญาตให้ผู้กำหนดได้เองแทนการกำหนดโดย DBMS สำหรับ InterSystems Caché จัดการ OID ให้โดยอัตโนมัติ โดยเป็นค่าตัวเลขจำนวนเต็ม เริ่มจาก 1 และเพิ่มค่าขึ้นทีละ 1 และสำหรับ IBM DB2 อนุญาตให้ USER เป็นผู้กำหนดรูปแบบการจัดเก็บด้วยตนเอง โดยใช้คีย์เวิร์ด USER GENERATED โดยสามารถกำหนดโดยใช้ Data Types ชนิดต่าง ๆ ได้ เช่น Integer, Double, Varchar เป็นต้น



รูปที่ 4. 6 ตัวอย่างรูปแบบ Object Identifier ใน Cache'

การใช้คุณสมบัติ OID จะใช้เมื่อมีการสร้าง Type ใหม่ขึ้น โดยกำหนด 2 ขั้นตอน ได้แก่

1. การสร้าง TYPE จะต้องกำหนด Ref Using เพื่อระบุว่า ต้องการใช้ Basic Data Type แบบใดและมีขนาดเท่าใดโดยจะต้องไม่มากกว่า 255 เช่น REF USING INTEGER หมายความว่า ต้องการเก็บค่า OID โดยใช้ประเภทข้อมูลเป็น INTEGER หรือ REF USING VARCHAR (13) FOR BIT DATA หมายถึง ต้องการเก็บค่า OID โดยใช้ประเภทข้อมูลเป็น VARCHAR ขนาด 13 ตัว โดยเก็บเป็นค่าไบนารี เป็นต้น ข้อจำกัดคือ ไม่สามารถกำหนดประเภทเป็นLONG VARCHAR, LONG VARGRAPHIC, BLOB, CLOB, DBCLOB, DATALINK และ STRUCTURE-TYPED
2. การสร้างตาราง จะต้องกำหนด Ref is OID USER GENERATED โดย OID จะเป็นชื่อแอททริบิวซ์ที่เป็น Object Identifier ดังแสดงตัวอย่างในรูปที่ 4.7

```
db2 => insert into person(oid,name) values(person_t(generate_unique()),'John')
DB20000I  The SQL command completed successfully.
db2 => select oid,name from person

OID                              NAME
-------------------------------- --------------------------------
x'20030310167804677317000000' John

  1 record(s) selected.
```

รูปที่ 4. 7 ตัวอย่างค่า OID ที่ได้จาก generate_unique () ใน DB2

4.2.3 กรณี Inheritance

โดยทั่วไประบบฐานข้อมูลเชิงวัตถุ จะสนับสนุนการถ่ายทอดคุณสมบัติทั้งแบบ Single และ Multiple สำหรับกรณีคาเช่ สนับสนุนทั้งสองแบบ แต่ DB2 สนับสนุนเฉพาะ Single Inheritance เท่านั้น ดังนั้น การแปลงจำเป็นต้องนำแอททริบิวซ์และเมดธอดทั้งหมดของคลาสแม่แต่ละคลาสมารวมไว้ด้วยกัน โดยพิจารณาจากคีย์เวิร์ด "SUPER" เช่น super = %Library.Persistent; สำหรับคลาสที่มี Super Class เป็น System Library ชนิดต่าง ๆ (ดูจากคลาสที่ขึ้นต้นด้วย %Library) จะยกเว้นไม่นำมาแปลง เนื่องจากเป็นคลาสของระบบ

```
Class MyApp.Person Extends %Persistent [ClassType = persistent]
{ Property Name As %String(MAXLEN=100);
  Property Home As Address; }
```

```
Class MyApp.Employee Extends Person [ClassType = persistent]
{ Property Salary As %Integer(MINVAL=0,MAXVAL=100000); }
```

รูปที่ 4. 8 การถ่ายทอดคุณสมบัติในแบบ CDL

ID	Name	Home_City	Home_State	Salary
3	Divad, Gino	Irvine	CA	22000

รูปที่ 4. 9 ตัวอย่างผลลัพธ์จากการสร้างตารางจากรูปที่ 4.8

```
<Super>%Library.Persistent</Super>
```

รูปที่ 4. 10 แสดง SUPER TYPE ในรูป XML

4.2.4 กรณี Collections

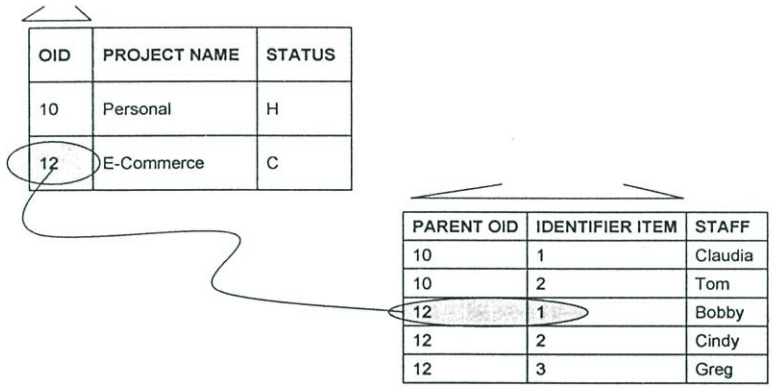
COLLECTIONS หรือค่าที่สามารถจัดเก็บได้มากกว่าหนึ่งค่าในหนึ่งแอททริบิวซ์ ได้แก่ Array, Bag, List, Set สำหรับคาเช่สนับสนุน Array, List โดย Array ข้อมูลจะถูกเรียงโดยอัตโนมัติ โดยใช้คีย์หรือ Subscript Item ในการอ้างถึงข้อมูล ในส่วนของ DB2 สนับสนุนข้อมูลประเภทนี้เฉพาะ SET ดังนั้น ในการแปลงโครงสร้าง จะต้องสร้างรีเลชั่นใหม่แยกออกมา เพื่อเก็บค่าต่าง ๆ ของแต่ละอินสแตนท์ ลักษณะข้อมูลประเภท Collection- Array ในคาเช่ มีลักษณะเป็นชุดของข้อมูล ที่ลำดับก่อนหลังไม่มีนัยสำคัญ

```
: <Property name="Dogs">
        <Type>User.Dog</Type>
        <Collection>array</Collection>
 </Property>
```

รูปที่ 4. 11 แสดง ATTRIBUTE ที่เป็น ARRAY

หลักและวิธีการแปลงข้อมูลประเภท Collection นี้ จะพิจารณาจากโหมด Collection ซึ่งจะปรากฏใน Property เป็นสำคัญ โดยให้สร้างเป็นรีเลชั่นใหม่ ใช้รูปแบบการตั้งชื่อรีเลชั่นใหม่ คือ ให้พิจารณาว่าแอททริบิวซ์เดิมที่เป็น Collection ชื่อแอททริบิวซ์อะไร นำมารวมกับ "ARRAY" โดยใช้ "_" คั่นกลาง เช่น แอททริบิวซ์ชื่อ STAFF ให้สร้างเป็นรีเลชั่นใหม่ ชื่อ STAFF_ARRAY เป็นต้น โดยในรีเลชั่นใหม่ที่สร้างนี้ จะประกอบด้วยแอททริบิวซ์ที่ถูกสร้าง 3 ตัว โดยมีรายละเอียดการเก็บ ดังต่อไปนี้

- ParentOID ได้แก่ ค่า OID จาก Row ในรีเลชั่นหลัก (Parent Table)
- Identifier ได้แก่ ค่าลำดับข้อมูลในชุดข้อมูล
- Value ได้แก่ ข้อมูลที่จัดเก็บจริง



OID	PROJECT NAME	STATUS
10	Personal	H
12	E-Commerce	C

PARENT OID	IDENTIFIER ITEM	STAFF
10	1	Claudia
10	2	Tom
12	1	Bobby
12	2	Cindy
12	3	Greg

รูปที่ 4. 12 แสดงเทคนิคการแปลงโครงสร้างที่เป็น Collection

จากการวิจัยพบว่า นอกจากการสร้างเป็นรีเลชั่นใหม่แล้ว สามารถเปลี่ยนวิธีโดยสร้างแอททริบิวซ์ใหม่หนึ่งแอททริบิวซ์รองรับ โดยแอททริบิวซ์นี้ มีชนิดข้อมูลเป็น LOB ที่เก็บ Identifier Item + Value และสร้าง UDF เพื่อจัดการข้อมูลดังกล่าวได้เช่นกัน ดังรูป 3.13 และสร้างเมดธอดรองรับเพื่อเข้าถึงและจัดการข้อมูลดังกล่าว

OID	PROJECT NAME	STAFF	STATUS
10	Personal	1:Claudia,2:Tom	H
12	E-Commerce	1:Bobby,2:Cindy,3:Greg	C

รูปที่ 4.13 แสดงเทคนิคการเก็บข้อมูลแบบ Collection โดยการรวมในหนึ่งแอททริบิวซ์

4.2.5 กรณีเมดธอด (Method)

ในคาเช่ มีเมดธอด ซึ่งเขียนด้วยภาษา COS, SQL, Basic Scripts สามารถทำได้ แปลงเป็นเมดธอดเช่นเดียวกัน โดยพิจารณาจากโหนดเมดธอด

สำหรับการแปลงเมดธอด ไม่ครอบคลุมในงานวิจัยนี้ แต่จะจัดเตรียมเมดธอดให้ โดยวิเคราะห์หาส่วนที่เป็นชื่อเมดธอด, พารามิเตอร์ และชุดคำสั่ง โดยการใส่หมายเหตุกำกับไว้ (--) เพื่อให้ ผู้ใช้ปรับแก้ไขก่อน เนื่องจากภาษาที่ใช้ใน Cache และ DB2 มีความแตกต่างกันในหลายประเด็น ตัวอย่างคำสั่งที่ใช้ในเมดธอด แสดงดังรูป 3.14

```
<Method name="Populate">
<ClassMethod>1</ClassMethod>
<FormalSpec>Count:%Library.Integer</FormalSpec>
<Implementation>
<![CDATA[
new n,b,ok,dir,i,d,file,f
kill ^User.DogD,^User.DogI,^User.DogS
set n="Fred,Baggie,Queenie,Spot,Maloue,Floppy,Helmut,Fetch"
set n=n_",HisMajesty,Woof,Kelly,Sweetie,Lobinho,Nosey,Puddle,Ralf"
set n=n_",Mitch,Schnautz,HushPuppy,Spitz,Tibo,Tobi,Vision,Why,Yorkie"
set b="BassetHound,Beagle,ChowChow,Dalmatian,Doberman"
set b=b_",EnglishSpaniel,GermanPointer,GoldenRetriever"
set b=b_",GreatDane,Hound,Kelpie,LabradorRetriever"
set b=b_",Lobinho,Pointer,Poodle,Ralf,Rhodesian,Schnauzer"
set b=b_",Spaniel,Spitz,Tibo,Tobi,Vizsla,Weimaraner,Yorkshire"
set dir="../Images/Dogs/"
for i=1:1:$get(Count,10) {
set d=##class(Dog).%New()
set d.DogName=$p(n,",",i#$l(n,",")+1)
set d.DogBreed=$p(b,",",i#$l(b,",")+1)
set file=dir_d.DogBreed_".jpg"
if ##class(%File).Exists(file) { do d.DogPicture.LinkToFile(file) }
else {w !,file," does not exist"}
do d.%Save()
do d.%Close()
}
quit

]]>
</Implementation>
<Method>
```

รูปที่ 4. 14 เมดธอดในแบบ XML

4.2.6 กรณีคิวรี่ (Query)

สำหรับคาเช่ คิวรี่ จัดเป็นเมดธอดประเภทหนึ่ง ที่อนุญาตให้เข้าถึงข้อมูลโดยใช้ภาษา SQL มีลักษณะเป็นการ Look up ข้อมูล สามารถแปลงให้อยู่ในรูปของเมดธอดได้

สำหรับการแปลงเมดธอดแบบคิวรี่ ไม่ครอบคลุมในงานวิจัยนี้ แต่จะจัดเตรียมเมดธอดให้ โดยวิเคราะห์หาเฉพาะส่วนที่เป็นชื่อเมดธอด, พารามิเตอร์ และชุดคำสั่ง โดยการใส่หมายเหตุกำกับไว้ (--) เพื่อให้ ผู้ใช้ปรับให้อยู่ในรูปแบบที่เหมาะสมกับ DB2

```
<Query name="ByName">
<Type>%Library.SQLQuery</Type>
<FormalSpec>Name:%Library.String</FormalSpec>
<SqlQuery>SELECT ID, DogName, DogBreed FROM Dog WHERE (DogName %STARTSWITH
    :Name)</SqlQuery>
<Parameter name="CONTAINID" value="1" />
<Parameter name="ROWSPEC" value="ID,DogName:%Library.String,DogBreed:%Library.String" />
  </Query>
```

รูปที่ 4. 15 ตัวอย่างเมดธอด ประเภทคิวรี เขียนด้วยภาษา SQL

4.2.7 กรณี Large Object Binary (LOB)

ในคาเช่ แบ่งประเภทของ LOB เป็น 2 ชนิด ได้แก่ Character Stream สำหรับเก็บข้อมูลประเภทเอกสาร หรือข้อมูลขนาดใหญ่ และ Binary Stream สำหรับเก็บข้อมูลประเภทไบนารี เช่น รูปภาพ, เสียง เป็นต้น ในการแปลงเป็น DB2 สามารถแปลงได้โดยตรง เนื่องจาก DB2 สนับสนุนข้อมูลทั้งสองประเภท กล่าวคือ แปลงเป็น CLOB และ BLOB ตามลำดับ

```
<Property name="DogPicture">
<Type>%Library.Stream</Type>
<Collection>binarystream</Collection>
<Parameter name="STORAGE" value="FILE" />
</Property>
```

รูปที่ 4. 16 ตัวอย่างรายละเอียดแอททริบิวซ์ที่เป็น LOB

4.2.8 กรณี Index

INDEX สามารถสร้าง Index ได้เหมือนกับ Index ที่มีอยู่เดิมในระบบฐานข้อมูลเชิงวัตถุ โดยการใช้คำสั่ง CREATE INDEX ชื่อINDEX ON ชื่อตาราง (แอททริบิวซ์);

```
<Index name="DogNameIndex">
  <Description>Index for property DogName</Description>
  <Properties>DogName</Properties>
</Index>
```

รูปที่ 4. 17 ข้อกำหนด INDEX

4.3 การเทียบเคียงประเภทข้อมูล

ในการแปลงโครงสร้างข้อมูลในงานวิจัยนี้ ใช้ตารางเทียบประเภทข้อมูลที่แตกต่างกันของ ODBMS ดังรูป 4.18

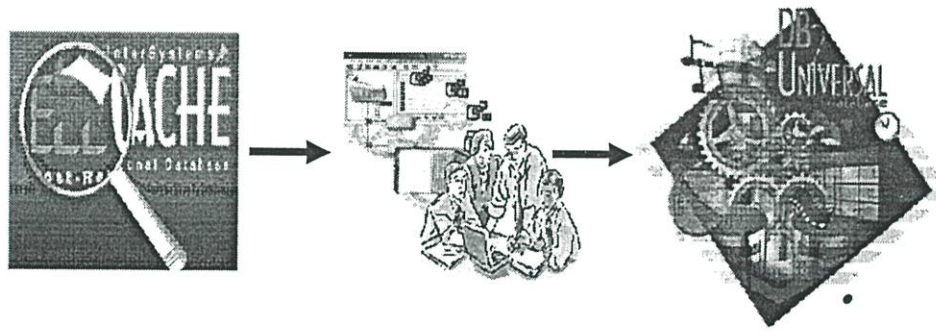
ประเภทข้อมูลคาเช่	ประเภทข้อมูล DB2
%Currency	Decimal
%Date	Date
%Float	Floating Point
%Integer	Integer
%Numeric	Numeric
%String	Varchar
%Time	Time
%BinaryStream	BLOB
%CharacterStream	CLOB

รูปที่ 4. 18 แสดงการเทียบประเภทข้อมูลระหว่าง Caché และ DB2

บทที่ 5
การพัฒนาเครื่องมือและการทำงาน

5.1 ภาพรวมการทำงาน

วัตถุประสงค์หลักของงานวิจัย ได้แก่ การย้ายระบบฐานข้อมูลเชิงวัตถุมาสู่ระบบฐานข้อมูลเชิงวัตถุสัมพันธ์ ซึ่งในงานวิจัยนี้ใช้ระบบฐานข้อมูล Caché และ DB2 เป็นเครื่องมือทดสอบ ดังรูปที่ 5.1 ลักษณะและกระบวนการโดยรวมคือ จะเริ่มจากนำข้อมูลที่เกี่ยวข้องทั้งหมดออกจาก Caché เข้าสู่กระบวนการแปลงระบบข้อมูล และท้ายสุดนำข้อมูลที่ได้จากการแปลง มาเข้าสู่ระบบฐานข้อมูลปลายทาง ในที่นี้ คือ ระบบฐานข้อมูลเชิงวัตถุสัมพันธ์ IBM DB2 โดยกระบวนการแปลง

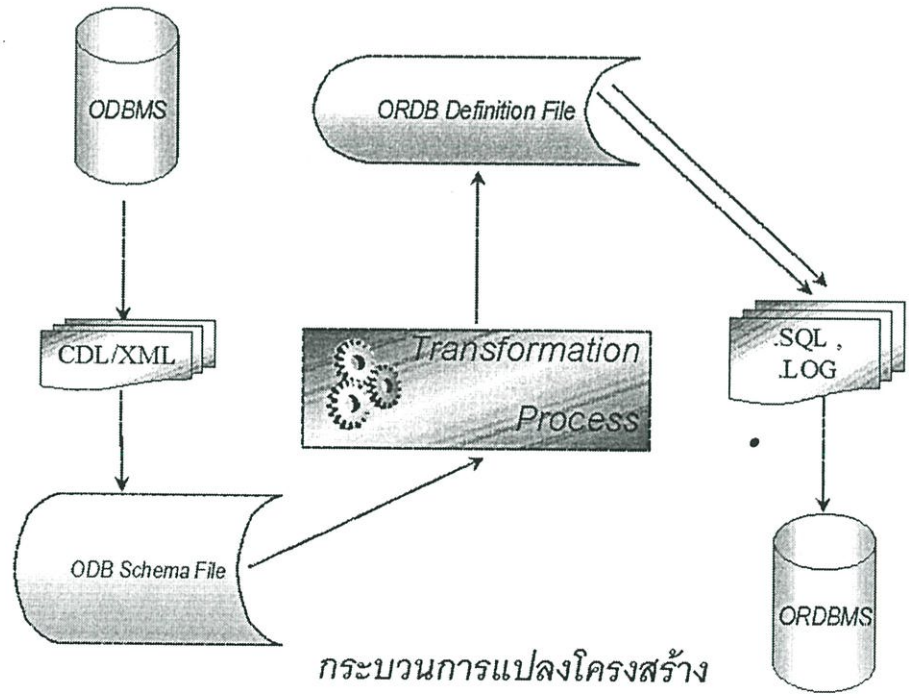


รูปที่ 5. 1 ภาพรวมการแปลงระบบฐานข้อมูล

5.2 กระบวนการแปลงระบบ

กระบวนการแปลงระบบฐานข้อมูลเชิงวัตถุ Caché มาสู่ระบบฐานข้อมูลเชิงวัตถุสัมพันธ์ DB2 ประกอบ ด้วยขั้นตอนหลัก 3 ส่วนหลัก (แสดงในรูปที่ 5.2) ได้แก่

1. ส่วนการเตรียมข้อมูลต้นทางจากระบบฐานข้อมูลเชิงวัตถุ (Caché) ได้แก่ โครงสร้างข้อมูล (Data Definition) และส่วนที่เป็นข้อมูล (Data) ในขั้นตอนนี้ จะได้ คลาสสกีมาที่ต้องการ
2. ส่วนการวิเคราะห์และจัดเตรียมโครงสร้างในรูป DB2 โดยการอ่านโครงสร้างข้อมูลที่ได้จากส่วนที่ 1 และแปลงให้เป็นคำสั่งสำหรับสร้างโครงสร้างข้อมูลบน DB2 ผลลัพธ์ที่ได้ในขั้นตอนนี้ คือ DB2 Script ที่จะนำไปใช้
3. ส่วนการจัดการนำเข้าข้อมูลปลายทางบนระบบฐานข้อมูล DB2 ได้แก่ การสร้างโครงสร้าง (Create Type, Table) และ นำเข้าข้อมูล (Import Data)



รูปที่ 5. 2 กระบวนการแปลงโครงสร้าง

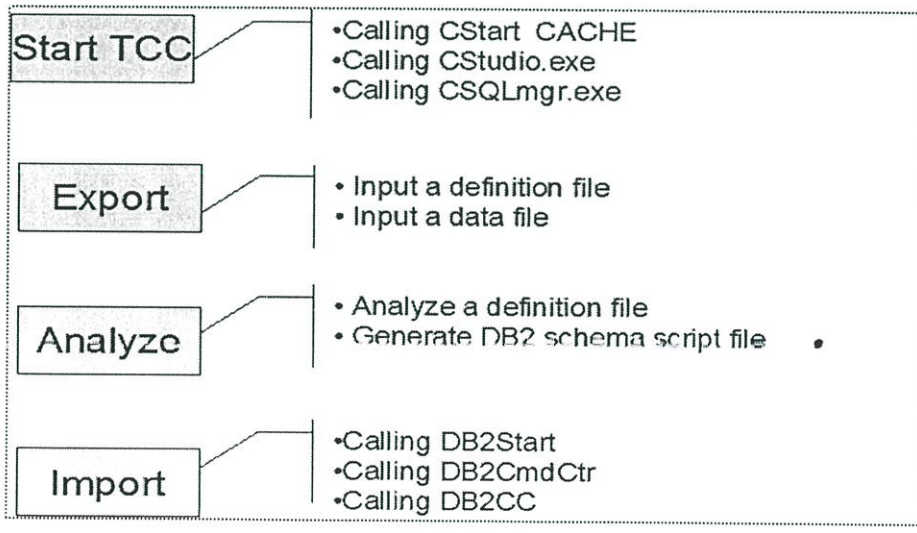
5.3 เกี่ยวกับเครื่องมือ

จากการวิจัยพบข้อจำกัดของการทำงานหลายประการ เช่น การสนับสนุนการเรียกใช้ API ของระบบฐานข้อมูลโดยตรงเพื่อทำงานบางอย่าง เป็นไปไม่ได้หรือยากต่อการทำ การติดต่อหรือจัดการข้อมูล จะต้องทำผ่านเครื่องมือที่ติดตั้งมาให้เท่านั้น ดังนั้น เพื่อให้เกิดความง่ายต่อการทำงาน จึงสร้างเครื่องมือช่วยการแปลงระบบ ชื่อ Transformation Control Center ในที่นี้จะเรียกว่า TCC พัฒนาด้วยภาษาจาวา โดย TCC จะทำหน้าที่เป็นศูนย์กลางในการเรียกใช้เครื่องมือต่าง ๆ ที่เกี่ยวข้อง รวมถึงการวิเคราะห์และสร้างชุดคำสั่งที่ถูกต้องให้กับผู้ใช้

การทำงานของ TCC จะมีการเรียกใช้เครื่องมือของระบบฐานข้อมูล Caché เพื่อการ Export โครงสร้างและข้อมูล และเรียกใช้เครื่องมือจากระบบฐานข้อมูล DB2 เพื่องาน Import โครงสร้างและข้อมูล

5.4 ลำดับการทำงานของเครื่องมือ TCC

การทำงานของ TCC หรือ Transformation Control Center ประกอบด้วยโมดูลหลักรวม 4 โมดูล เรียงลำดับการทำงาน ดังรูปที่ 5.3



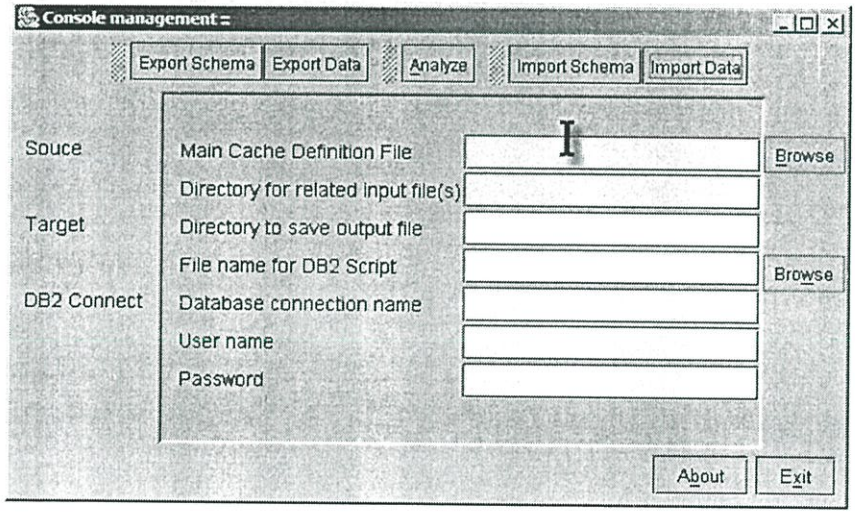
รูปที่ 5. 3 แสดงโมดูลและลำดับการทำงาน

5.5 โมดูลและโปรแกรมที่เกี่ยวข้อง

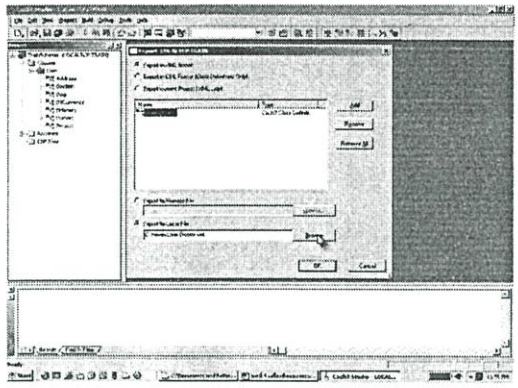
5.5.1 โมดูลที่ 1: Start TCC

TCC Console Manager จะทำหน้าที่หลักในการประสานการทำงานระหว่างผู้ใช้ และโปรแกรมหรือโมดูลต่าง ๆ ที่เกี่ยวข้อง โดยผู้ใช้จะต้องระบุข้อมูล ซึ่งจะใช้ในการปรับเปลี่ยนและจัดการโครงสร้างและข้อมูลตามที่ต้องการ ดังแสดงในรูปที่ 5.3

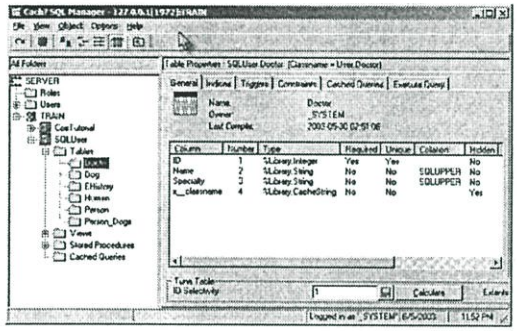
การทำงานเริ่มต้น โดย TCC จะเปิดการติดต่อกับระบบฐานข้อมูล Caché ก่อน โดยในส่วนนี้จะมีการเรียกใช้เครื่องมือจาก Caché 3 โปรแกรม ได้แก่ CStart ทำหน้าที่เปิดการติดต่อกับ Caché , CStudio ทำหน้าที่ช่วยสร้างข้อกำหนดคลาสใน Caché ในรูปแบบ XML (ดังแสดงในรูปที่ 5.4) และ CSQLMgr ทำหน้าที่ถ่ายโอนข้อมูลออกจาก Caché ในรูปแบบ Text File (ดังแสดงในรูปที่ 5.5, 5.6, และ 5.7 ตามลำดับ)



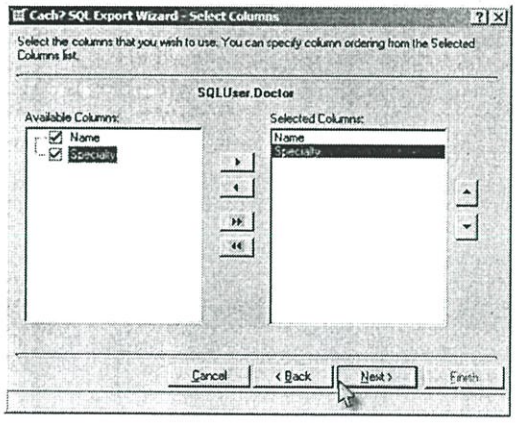
รูปที่ 5. 1 ส่วนการรับค่าพารามิเตอร์และควบคุมโปรแกรม



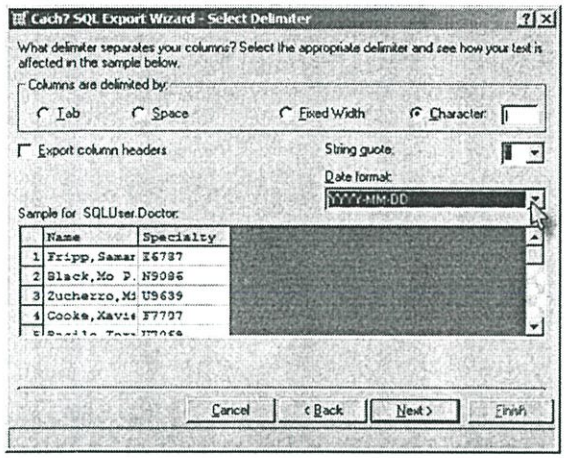
รูปที่ 5. 2 แสดงการ Export Class Definition



รูปที่ 5. 3 แสดงโครงสร้างของคลาสในเชิงรีเลชั่นแนล



รูปที่ 5. 4 เลือกแอททริบิวซ์ที่ต้องการส่งออก

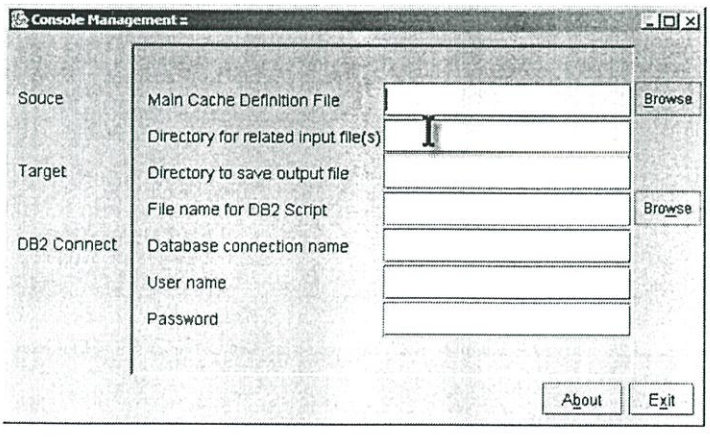


รูปที่ 5. 5 แสดงข้อมูลที่จะส่งออกและกำหนด Column Delimiter

ผลลัพธ์ที่ได้จากโมดูลนี้ ได้แก่ Caché คลาสสกีมาในรูป XML ทั้งคลาสหลัก และคลาสที่เกี่ยวข้อง, แฟ้มข้อมูลที่จะถ่ายโอน

5.5.2 โมดูลที่ 2: Export

โมดูลส่วน EXPORT จะทำหน้าที่รับข้อมูลต่าง ๆ (Parameters) เพื่อส่งให้กับโมดูลที่ 3 โดยข้อมูลที่รับเข้ามาแบ่งออกเป็น 3 ส่วนได้แก่ Source ได้แก่ ข้อมูลต้นทาง ผู้ใช้จะต้องระบุคลาสหลักที่ต้องการแปลง และไดเรคทอรี่ที่เก็บคลาสอื่น ๆ ที่เกี่ยวข้อง, ส่วนที่สอง Target ได้แก่ ไดเรคทอรี่ที่จะเก็บข้อมูลผลลัพธ์ และชื่อแฟ้มข้อมูลที่จะเก็บผลการแปลงโครงสร้าง, และส่วนที่สาม DB2 Connect ได้แก่ ชื่อฐานข้อมูล DB2 ที่ต้องการติดต่อ, ชื่อผู้ใช้ และรหัสผ่าน (ดังแสดงในรูปที่ 5.8)



รูปที่ 5. 6 ส่วนการรับข้อมูล

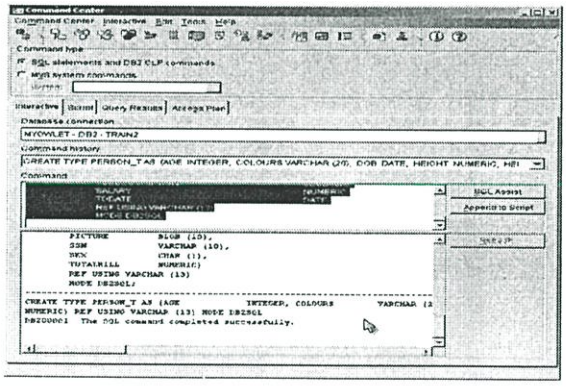
5.5.3 โมดูลที่ 3: Analyze

การ Analyze เป็นส่วนการอ่านข้อมูลที่ได้จากโมดูลที่ 2 โดยข้อมูลเหล่านั้น จัดเป็น Parameters ที่จะใช้ประกอบการวิเคราะห์และแปลงโครงสร้าง โดยหลักการแปลงจะใช้วิธีอ่านโหนดต่าง ๆ จากโครงสร้างที่เป็น XML และแปลงผล โดยผลลัพธ์ที่ได้ จะเก็บบันทึกไว้ใน DB2 Result File ซึ่งจะนำไปใช้ในโมดูลที่ 4 ต่อไป

นอกจากนี้ ในส่วนนี้ โมดูลจะอ่าน Input Data File ที่ได้จาก Caché ทั้งหมดว่ามีจำนวนเท่าใด และจะสร้างชุดคำสั่งเพื่อเตรียมกำหนดค่า OID ให้แก่แต่ละ ROW เนื่องจาก DB2 จะให้ผู้ใช้เป็นผู้กำหนดค่า OID เองทุกครั้ง ซึ่งแตกต่างจาก Caché ที่ Caché จะสร้างให้เองโดยอัตโนมัติ

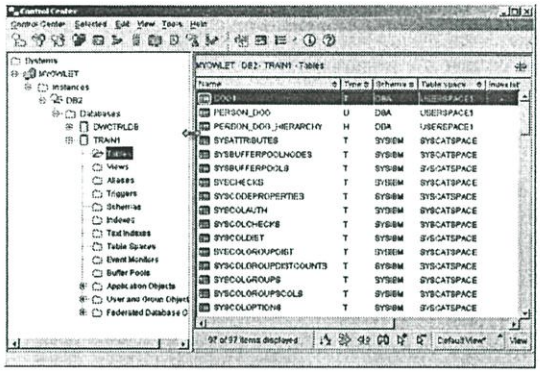
5.5.4 โมดูลที่ 4: Import

เป็นการสร้างโครงสร้างข้อมูลใหม่บนฐานข้อมูล DB2 โดยการเรียกเครื่องมือของ DB2 ชื่อ DB2CmdCtr ซึ่งเป็นเครื่องมือสำหรับการทำงานแบบ Interactive โดยนำ DB2 Script File ที่ได้มาสร้างทั้งในส่วน TYPE และ TABLE ให้เรียบร้อย และเรียก SCRIPT ที่เตรียมไว้สำหรับการสร้าง OID ให้ทำงานด้วย



รูปที่ 5. 7 การเรียกใช้ DB2CmdCtr เพื่ออ่าน Script

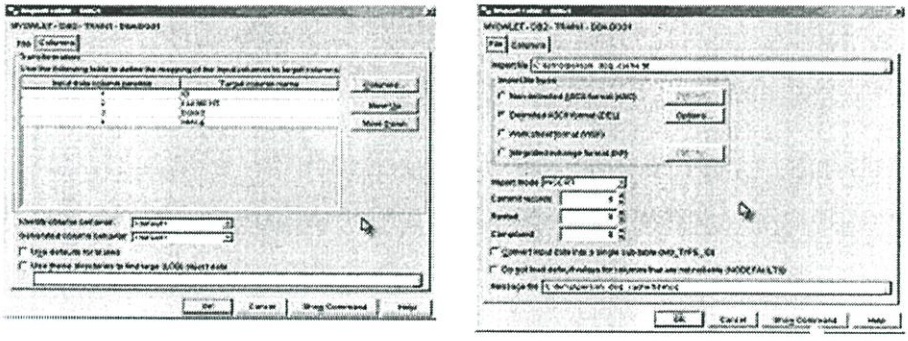
ส่วนต่อมาเป็นการนำข้อมูลเข้า TCC จะเรียกโปรแกรม DB2CC (ดังแสดงในรูปที่ 5.10) ซึ่งเป็น Control Center สำหรับ DBA ทำงาน โดยจะใช้ในการนำข้อมูลเข้าสู่ TABLE ที่สร้างเตรียมไว้ก่อนแล้ว ในส่วนของการใช้งานผู้ใช้จะต้องเลือกฐานข้อมูลและตารางที่ต้องการทำงานก่อน



รูปที่ 5. 8 DB2CC สำหรับการจัดการโครงสร้างและข้อมูล

เมื่อตรวจสอบพบว่า มีการสร้างตารางให้ถูกต้องแล้ว ลำดับต่อมาคือการนำข้อมูลเข้า ในที่นี้ จะใช้วิธีการ Import ซึ่งมีข้อดี คือ หากนำข้อมูลเข้าไม่ครบหรือติดปัญหาใด ๆ ที่ทำให้ฐานข้อมูลไม่ถูกต้องหรือไม่สมบูรณ์ DB2 จะทำการ Roll Back การทำงานทั้งหมด

ในส่วนนี้ จะต้องมีการกำหนดรายละเอียดการนำข้อมูลเข้า โดยการเลือกแอททริบิวซ์ที่ต้องการและลักษณะของแฟ้มข้อมูล (Text) ที่จะนำเข้าสู่ DB2



รูปที่ 5. 11 การกำหนดรายละเอียดสำหรับการ Import ข้อมูล

5.6 สรุปการทำงาน

การใช้เครื่องมือ TCC นี้ เครื่องมือจะทำหน้าที่เป็นศูนย์กลางการเรียกใช้เครื่องมือต่าง ๆ ที่เกี่ยวข้องในแต่ละส่วนให้ เพื่อลดความสับสนในการเรียกใช้เครื่องมือแต่ละส่วน รวมถึง TCC จะทำการวิเคราะห์และแปลงโครงสร้างจากฐานข้อมูลเชิงวัตถุ เป็นฐานข้อมูลเชิงวัตถุสัมพันธ์ให้ถูกต้อง เพื่อนำไปสู่การสร้างโครงสร้างและนำเข้าข้อมูล

โครงสร้างข้อมูลที่ได้จาก Caché จะอยู่ในรูป XML SCHEMA โดยในหนึ่งไฟล์ อาจประกอบด้วยข้อมูลที่มีหลายประเภท หลายคุณสมบัติร่วมกันก็ได้ เช่น PRIMITIVE DATA TYPES, ARRAY, INHERITANCE, LOB นั่นหมายความว่า โปรแกรม TCC โดยโมดูล Analyze จะทำหน้าที่ตรวจสอบข้อมูลที่เกี่ยวข้องและจำเป็นต้องใช้ สำหรับกรณีถ้าคลาสใดมีเมดธอดหรือคิวรี ก็สร้างด้วยคำสั่ง CREATE METHOD และเก็บรายละเอียดต่าง ๆ รอไว้ โดยใส่คอมเมนท์กำกับไว้ โดยเครื่องหมายคอมเมนท์ คือ “- -“

ในกรณีที่อ่าน SCHEMA แล้วพบว่า ไฟล์ที่เกี่ยวข้องไม่ครบ จะรายงานความผิดพลาด และหยุดการทำงาน ซึ่งสามารถตรวจสอบได้จาก LOG ที่ถูกสร้างขึ้น เช่น การไม่ระบุไฟล์ XML SCHEMA ที่เกี่ยวข้อง ในส่วนที่เป็น SUPER-CLASS, USER-DEFINED-CLASS เป็นต้น

ผลลัพธ์ทั้งหมดที่ได้ คือ ชุดคำสั่ง SQL ในรูปแบบของ DB2 โดยจัดเก็บบันทึกลงไฟล์ตามที่ระบุไว้ใน PARAMETERS โดยมีส่วนขยายแฟ้มเป็น .SQL และในส่วน LOG FILE จะมีนามสกุล .LOG โดยจัดเก็บบันทึกการทำงานที่เกิดขึ้น โดยจัดเก็บการเริ่มทำวันที่-เวลา สิ่งที่กำลังกระทำ เช่น เริ่มอ่านโหนดใด มีการสร้าง TYPE, TABLE, METHOD, INDEX อะไรบ้าง ถ้ามีข้อผิดพลาด ก็จะระบุไว้ใน LOG ด้วยเช่นกัน ทั้งนี้ เพื่อใช้ในการตรวจสอบการทำงานภายหลัง โดย API สำหรับจัดการ LOG ซึ่งใช้ในงานวิจัยนี้ ชื่อ Apache Log4j

บทที่ 6

สรุปผลการวิจัยและแนวทางการพัฒนาในอนาคต

6.1 สรุปผลการวิจัยและแนวทางการพัฒนาในอนาคต

แนวคิดจากงานวิจัยนี้ กล่าวถึงเทคนิคแนวทางการแปลงโมเดลเชิงวัตถุ มาสู่โมเดลเชิงวัตถุสัมพันธ์ โดยให้คงความสมบูรณ์ของข้อมูลเดิมไว้ โดยใช้วิธีการทำ Reverse Engineering และ Mapping Technique ซึ่งได้โครงสร้างในรูปโมเดลเชิงวัตถุสัมพันธ์ที่ถูกต้อง และสามารถนำไปใช้ในการทำงานอื่น ๆ เช่น การพัฒนาระบบคลังข้อมูล (Information Warehouse) ต่อไป

ในงานวิจัยได้พัฒนาเครื่องมือช่วยแปลงเปลี่ยนโมเดล โดยออกแบบให้ทำงานแบบ Interactive กล่าวคือ เครื่องมือจะอ่านโครงสร้างเดิม เพื่อนำมาวิเคราะห์แปลงรูปให้อยู่ในชุดคำสั่งของระบบฐานข้อมูลเชิงวัตถุสัมพันธ์ ในส่วนของเมดธอด จะวิเคราะห์ในส่วนที่เป็นลายเซ็นต์ (Method Signature) และให้ผู้ใช้สามารถปรับเปลี่ยนชุดคำสั่งภายใน เนื่องจากภาษาที่ใช้ในระบบฐานข้อมูลเชิงวัตถุเป็นภาษาเฉพาะ จากนั้นจึงทำการสร้างชุดคำสั่งที่สมบูรณ์เพื่อนำไปสู่การถ่ายโอนข้อมูลต่อไป

เนื่องจากในงานวิจัยนี้ เน้นการแปลงโครงสร้างข้อมูลเป็นหลัก โดยคำนึงถึงสิ่งที่ระบบฐานข้อมูลเชิงวัตถุมี แต่ไม่มีบนระบบฐานข้อมูลเชิงวัตถุสัมพันธ์ โดยการสร้างสิ่งทดแทน ให้ทำงานได้เหมือนเดิม หรือใกล้เคียงของเดิมให้ได้ อย่างไรก็ดี ในงานวิจัยนี้ ไม่ครอบคลุมการจัดการคุณสมบัติต่อไปนี้

1. Collection สนับสนุนเฉพาะการแปลง Array ยังไม่ครอบคลุมประเภท List
2. เมดธอด เครื่องมือจะช่วยวิเคราะห์ส่วนที่เป็นลายเซ็นต์ (Method Signature) และใส่เครื่องหมายคอมเมนท์ให้ โดยยังไม่แปลให้อยู่ในรูปเมดธอดที่ DB2 เข้าใจและทำงานได้ทันที
3. การนำเข้าข้อมูล สามารถนำข้อมูลเข้าได้เฉพาะประเภท String, Numeric และ Date ไม่ครอบคลุมประเภท LOB และ OID ซึ่งจำเป็นต้องพัฒนาโปรแกรมเฉพาะสำหรับนำเข้าข้อมูลโดยตรง

อย่างไรก็ดี ในส่วนของเมดธอด เนื่องจากผู้พัฒนาระบบฐานข้อมูลเชิงวัตถุ ต่างพัฒนาโดยใช้ภาษาเฉพาะของตนเองเป็นหลัก มีรูปแบบเฉพาะตัว ดังนั้น การแปลงเมดธอด จึงเป็นประเด็นที่สำคัญประเด็นหนึ่ง ที่จะช่วยให้ผู้ใช้ทำงานได้สะดวกยิ่งขึ้น

เอกสารอ้างอิง

[1] Alhajj, R. and Polat, F. 2001. "Reengineering relational databases to object-oriented: constructing the class hierarchy and migrating the data". **Reverse Engineering Proceedings Eighth Working Conference on 2001.** 335-343.

[2] Andrew J. McAllister. 1996. "Reverse Engineering a Medical Database". **IEEE Computer Society DL.** 121-130.

[3] Chamberlin, D.D. 1997. "Evolution of object-relational database technology in DB2". **Compcon '97 Proceedings, IEEE.** 131 -135.

[4] Cobb, M.A. Foley, et. al. "An OO database migrates to the Web" **IEEE Software.** 15(3) : 22 -30.

[5] Don Chamberlin. 1998. **A Complete Guide to DB2 Universal Database.** CA : Morgan Kaufmann Publishers.

[6] Hsieh, S.-Y, et. al. 1993. "Capturing the objected-oriented database model in relational form". **IEEE Transaction on Software.** 12 : 202 -208.

[7] IBM. **IBM DB2 UDB.** [Online]. Available: http://www.ibm.com/db2.

[8] InterSystems . **Cache Post-Relational Database Advanced System Management Guide.** [Online]. Available: http://www.e-dbms.com.

[9] Jens Janke, et. al. "A Design Environment for Migrating Relational to Object Oriented Database Systems". [Online]. Available: http://www.uni-paderborn.de/fachbereich/AG/schaefer/index_dt.html.

[10] Kitney, R.I. Forbes Dewey and C., Jr. 1998. "The electronic medical record (EMR) and object relational databases". **Proceedings of the 20th Annual International Conference of the IEEE.** 3 : 1181 -1184.

[11] Martin, R.D. and Chalana, V. 1997. "The S-PLUS DataBlade for INFORMIX-Universal Server.The natural wedding of an object relational database with an object-oriented data analysis engine Scientific and Statistical Database Management". **Proceedings of the Ninth International Conference on 1997.**

[12] Martin Gogolla, et. al. "Stepwise Re-Engineering and Development of Object-Oriented Database Schemata". [Online].

Available: http://www.informatik.uni-bremen.de/biss/1998/index_d.htm.

[13] Michael J. Carey. "A Decade of Turmoil Of Objects and Databases". [Online]. Available: http://db.cs.berkeley.edu/~jmh/cs262b/object-turmoil.

[14] Michael Zimmer. 2001. **Data Management Handbook.** 3rd ed. USA : CRC Press.

[15] Michael Stonebreaker and Paul Brown. 1999. **Object-Relational DBMSs Tracking the next great wave.** 2nd ed. CA : Morgan Kaufmann Publisher.

[16] Meier A. Dippold, et. al. 1994. "Hierarchical to relational database migration". **IEEE Software** , 11(3) : 21 -27.

[17] Patrick O' Neil and Elizabeth O'Neil. 2001. **"Database Principles Programming Performance".** 2nd ed. USA : Morgan Kaufmann Publishing.

[18] Ringo Pang. "Data Conversion from Object-Oriented to Relational database and its Verification by use of Information Capacity". [Online]. Available: http://www.cse.cuhk.edu/hk/~acm-hk/activity/pg/cityu-ptpang.pdf.

[19] Suphamit Chittayasothorn. 2001. "Database Design and SQL". **Seminar in Database Theory and Design.** Bangkok, July 2001.

[20] Suphamit Chittayasothorn. 2001. "Database Concepts and Models". **Cache Object Database Technology Conference 2001.** Bangkok. June 2001.

[21] S. Liao, Y.P. Shao and O.M. Wong. 1996. "Transformation from Relational Model to O-O Model: A Reengineering Approach". **International Journal of Computer and Engineering Management.** 4 : 30-39.

[22] V.Srinivasan and D.T. Chang. 1997. "Object persistence in object-oriented applications". **IBM Systems Journal.** 36(1).

ภาคผนวก ก

ผลงานวิจัยที่ได้รับการตีพิมพ์

A TRANSFORMATION FROM AN OBJECT DATABASE TO AN OBJECT RELATIONAL DATABASE

Kobchai Niyomthum[1]
Information Science, Faculty of Information Technology
King Mongkut's Institute of Technology Ladkrabang Bangkok Thailand 10520
E-mail: Kobchai@mju.ac.th
Suphamit Chittayasothorn[2]
Department of Computer Engineering, Faculty of Engineering
King Mongkut's Institute of Technology Ladkrabang Bangkok Thailand 10520
Phone (02)737-2667, Fax (02)326-4335
E-mail: Suphamit@kmitl.ac.th

Abstract

Object Databases and object relational databases are both widely used in industries. The major differences are mainly the level of encapsulation and the language supported by individual DBMSs. Object databases are fully encapsulated and allowed only the use of methods to communicate with the objects. Object relational database, on the other hand allows the use of object-oriented SQL on open attributes. Both databases support the use of reference pointers and subtype hierarchies.

Major software tools such as information warehouse software are now available commercially in the form of object relational representation. The purpose of this research project is to transform production-level systems on object database to object relational information warehouses.

Keyword: Object database, Object relational database, database transformation

1. Introduction.

At present, several models of databases are used in the industries. There could be several database management system products operate on different hardware and operating system platforms. During the last decade, most databases are relational and the transfer of data between relational products is a trivial activity. Recently, objects and object relational DBMSs become more common. Major DBMS vendors move towards the object relational model [1]. The ORDB seemed like a smooth transition towards pure OODB at the beginning but nowadays it looks like they are here to stay. Object relational databases have some features in common with object databases without the requirement to be encapsulated. On the other hand, new DBMS vendors jump direct to the OODB and new generation users greet them with joy. In an organization, there could be a mixture of relational, object relational and object databases. Since some DBMS products have available software tools different from others, database transformation is a good strategy to fully utilize these tools on different databases.

This research project presents a transformation from an object database to an object relational database. The source system is an object database based on InterSystems Caché [5] and the target system is an object relational database based on IBM DB2 [3].

2. Object and Object Relational Databases.

---

[1] Graduate Student, Information Science, Faculty of Information Technology, King Mongkut's Institute of Technology Ladkrabang.
[2] The Director of Computer Research and Service Center, King Mongkut's Institute of Technology Ladkrabang and Associate Professor, Department of Computer Engineering, Faculty of Engineering .

There are various database models which are supported by commercial DBMSs. The most popular one is the relational database model [11]. Object relational and object databases are increasingly popular. In fact the object relational database model is a superset of the relational database model. The object features are introduced to existing relational products. Some implementations are more relational than object but some are more on the object side.

Basically, each data model is distinguished by the logical data structure(s) that the model support and the language used on the data structures. In the relational database model, the only data structure is the relations. A relation is defined to be a subset of the Cartesian product of domains. This abstract notion of a relation is commonly represented by a flat table with atomic data items. Each row represents a tuple of a relation. Since tuples are members of a set, the rows that represent them are unique and unsorted. The language that works on such relations must be relational complete. A relational complete language must be at least as powerful as the relational algebra or the relational calculus. A popular relational complete language is the Structured Query Language (SQL).

The object database model follows the object paradigm. An object may have non-atomic attributes such as sets, lists, bags, arrays and structures. Each object is uniquely identified by an object identifier (OID) which is generated by the DBMS. Reference attributes which point to other objects are used instead of the traditional primary key foreign key relationships. The most significant property of an object is that all these attributes must be encapsulated. Users and programs that use the objects cannot see the attributes. The only way to which these objects are communicated is by calling the methods. Methods are functions or procedures defined by the object creators. Attributes and methods can be inherited down subtype hierarchies. Polymorphism and multiple inheritances are also supported. In summary, the data structure of an object database is the encapsulated object and the language is the method calling. The programming language that is used to implement the methods is product specific.

Information systems developed on object databases are easy to be maintained. Business rules can be coded as parts of methods. So, when such rules are changed, only the methods need modifications not the application programs. The price we paid for the ease of maintenance is the lack of ad hoc query facilities that relational products enjoy. In SQL, one may issue ad hoc queries on open attributes in the interactive SQL mode as well as routine queries in the embedded mode. Using object databases, since all attributes are encapsulated, the use of ad hoc queries is not possible and only the method-calling embedded mode is allowed.

Object relational databases try to fill the gap between relational and object databases. The logical data structure is objects without encapsulation. Attributes may not be atomic and can be referred to directly by application programs. Not all products support all collection types namely set, bag, list and array. Some ORDBMS only allows nested relations. The use of object identifiers and reference pointers are commonly supported but multiple inheritance is product specific. The highlight of object relational database could be the support of user defined data types which allows complex data types to be conveniently implemented.

The object relational language is an SQL-like language formerly known as Object SQL. There are two popular varieties: the Object Query Language (OQL) [7] and the SQL3 [7],[11]. The latter is more popular among commercial DBMS and is considered a superset of SQL.

3. The Transformation Process

Most literatures concentrate in the transformation from relational databases to object databases [6],[12],[14]. Interactive transformation tools that allowed users to dynamically modify database structures are suggested [2],[8]. Data structure and query transformations from object databases to relational databases are also discussed [4],[10]. It has been mentioned that the object relational databases are not only more suitable to handle complex objects than relational databases but an object relational DBMS also support better transaction processing facilities than a current object DBMS [9],[13]. So far, no literatures described the transformation from object to object relational databases are found. Our transformation technique describes a general transformation but the examples are the transformation from an InterSystems Caché database to an IBM DB2 database. The transformation is as follows:

1. An object class is transformed into a row type. Instances of the class are transformed into either a table or a collection depends on the type of the class.
   - If the class is a persistent class then a table is created for an extent (instances) of the class.
   - If the class is an embedded class then the corresponding type is a collection type of the row type and no tables are created.
2. Object identifiers generated by the ODBMS are transformed into user-generated object identifiers. In some systems the user-generated OIDs can be generated by the ORDBMS provided that the data definitions refer to the system generated option.
3. Multiple inheritance if not supported by the ORDBMS requires the creation of attributes that appear in all the parent classes which are not inherited.
4. Collection types (set, bag, list, and array) if not fully supported by the ORDBMS need to be simulated. The simulation of these types on an ORDBMS is product specific.
5. Methods in an object database are transformed into methods. The choice is product specific.

4. The Caché to DB2 Implementation.

As earlier mentioned, our source system is InterSystems Caché ODBMS and our target system is DB2 ORDBMS. We first obtained the source system details from the Caché system catalog described in Caché Definition Language. An example of the definition is shown in Figure 1.

```
Class User.Person Extends (%Library.Persistent,
%Library.Populate) [ ClassType = persistent ]
{
Property Dogs As User.Dog [ Collection = array ];
Property Home As User.Address;
Property Name As %Library.String [ Required ];
...
Method AgeGet() As %Library.Integer
        {
         if ..DOB="" quit ""
        quit $h-..DOB\365.25
        }
}
```

Figure 1 An Example of Caché Database Schema

```
Row Type
Create type ADDRESS_T as
  ( city    varchar(15),
    zipcode  varchar(5))
    ref using integer mode db2sql;
create table ADDRESS of ADDRESS_T
    (ref is oid user generated);
Column Type
Create type PERSON_T as
    ( name    varchar(15),
      surname varchar(15),
      home     ADDRESS_T)
      ref using integer mode db2sql;
create table PERSON of PERSON_T
      (ref is oid user generated);
```

Figure 2 shows the creation of a DB2 row type ADDRESS_T which is used as a type of the attribute My_ADDRESS of the row type PERSON TYPE. The type PERSON_TYPE is transformed from a class in Caché database and the corresponding extent of the class is transformed into the table PERSON

Figure 2 The implementation of Caché classes in DB2 row types and tables

An object identifier (OID) in Caché is system generated but the DB2 OID is user generated. The function GenerateUnique() in DB2 provides the same functionality within a table.

Caché supports multiple inheritance but DB2 only allows single ones. New attributes copied from the non-inherited parents have to be added to the subtypes.

DB2 only supports composite attributes and not collections. Caché collection types have to be simulated.

A table is created to handle each such type. The table comprises 3 columns. Each row is a triplet (OID, IDENTIFIER_ITEM, <ATTRIBUTE_NAME>). An example of this implementation technique is shown in Figure 3.

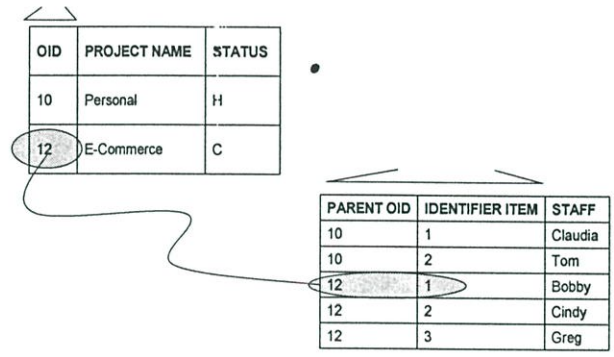


Figure 3 The simulation of a Caché Collection Type using a DB2 Table

There is another possible implementation of collection types in DB2. The collection attribute can be implemented as a DB2 Character Large Object Binary (CLOB) attribute. A user-defined function is implemented to extract each item from the attribute. Figure 4 shows this implementation approach for the same collection type of Figure 3.

OID	PROJECT NAME	STAFF	STATUS
10	Personal	1:Claudia,2:Tom	H
12	E-Commerce	1:Bobby,2:Cindy,3:Greg	C

Figure 4 The Simulation of a Caché Collection Type using DB2 CLOB data type

There are two types of methods in Caché namely the class methods and the instance methods. Both of them can be transformed into DB2 methods.

The Caché Query facility can be transformed into corresponding DB2 method.

There are two types of Large Object Binary (LOB) in Caché namely the Character Stream for mass data collections and the Binary Stream for binary data collections such as images and sound. The first one can be transformed into DB2 CLOB and the second one to BLOB.

Both systems support the use of indexes and the transformation is straight forwarded.

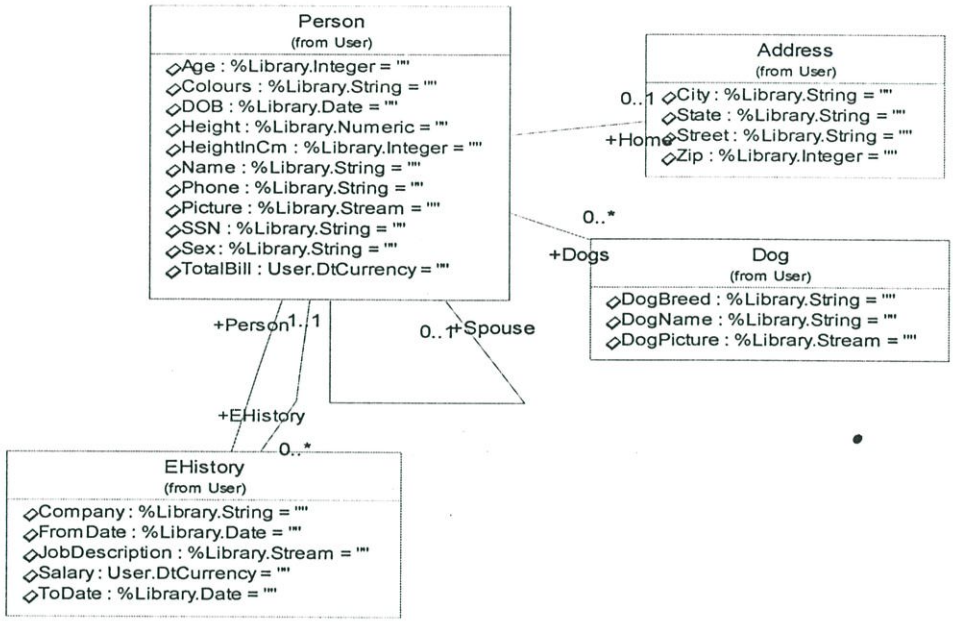


Figure 5 Personal Information System Class Diagram

```
create type DOG_T as (DogBreed  varchar(15), DogName  varchar(15),DogPicture  blob)
                      ref using integer mode db2sql;
create type ADDRESS_T as(City varchar(15), State  varchar(15), Street  varchar(15),Zip  varchar(5))
                     ref using integer mode db2sql;
create type EHISTORY_T as (Company  varchar(20),FromDate  date,JobDescription  clob,
                     Salary  double, ToDate date)
                     ref using integer mode db2sql;
create type PERSON_T as (Age integer, Colours varchar(10),DOB date,Name varchar(25)
                              Height double, HeightInCm double, ,Phone varchar(20),
                              Picture BLOB, SSN  varchar(10), Sex varchar(1),
                              Spouse  PERSON_T,
                              Home  ADDRESS_T,
                              Ehistory  EHISTORY_T,
                              Dogs  DOG_T)
                              ref using integer mode db2sql;
Create table dog of DOG_T (ref is oid user generated);
Create table ehistory of EHISTORY_T (ref is oid user generated);
Create table person of PERSON_T (ref is oid user generated);
```

Figure 6 DB2 row type and table definitions

Figure 5 shows a Caché class diagram which is our source schema. The corresponding DB2 row type and table definitions are shown in Figure 6.

5. Conclusions.

This research project presents the transformation from an object database to a corresponding object relational database. The prime motivation is to allow the data to be transfer among DBMSs of different data models in organizations where multiple data models are employed. A transformation procedure is proposed. An interactive software tool that transforms a Caché object database into a DB2 object relational database was developed. The transformation speed depends on the size of the schemas and not on the size of the database.

Biographies

Kobchai Niyomthum is a staff member of the Department of Computer Science, Maejo University, Chiangmai, Thailand. He is currently on a study leave under a Royal Thai Government Scholarship at the Faculty of Information Technology, King Mongkut's Institute of Technology, Ladkrabang, Bangkok, Thailand.

Suphamit Chittayasothorn is an associate professor at the Department of Computer Engineering, Faculty of Engineering and also the Director, Computer Research and Service Center, King Mongkut's Institute of Technology, Ladkrabang, Bangkok, Thailand. He obtained a Ph.D. in Computer Science from the University of Queensland, Australia. Dr. Chittayasothorn is an IEEE member and has served as a chairman of database sessions of IEEE conferences.

References

[1] Chamberlin, D.D. 1997. "Evolution of object-relational database technology in DB2". Compcon '97. Proceedings, IEEE : 131-135.

[2] Cobb, M.A.; Foley, et. al. 1998. "An OO database migrates to the Web" IEEE Software. 15(3): 22-30.

[3] Don Chamberlin. 1998. A Complete Guide to DB2 Universal Database. CA: Morgan Kaufmann Publishers.

[4] Hsieh, S.-Y, et. al. 1993. "Capturing the objected-oriented database model in relational form". Proceedings., Seventeenth Annual International on 1993: 202 -208.

[5] InterSystems . 2000. **Caché Post-Relational Database Advanced System Management Guide.** [Online]. Available: http://www.e-dbms.com.

[6] Jens Janke, et. al. 2001 . **"A Design Environment for Migrating Relational to Object Oriented Database Systems".** [Online]. http://www.uniaderborn.de/ fachbereich/AG/schaefer/index_dt.html.

[7] Michael Stonebreaker, Paul Brown. 1999. Object-Relational DBMSs Tracking the next great wave. 2nd ed. CA: Morgan Kaufmann Publisher.

[8] Meier, A.; Dippold, et. al. 1994. "Hierarchical to relational database migration". IEEE Software , 11(3): 21-27.

[9] Patrick O' Neil, Elizabeth O'Neil. 2001. "Database Principles Programming Performance". 2nd ed. USA: Morgan Kaufmann Publishing.

[10] Ringo Pang. "Data Conversion from Object-Oriented to Relational database and its Verification by use of Information Capacity". Hong Kong.

[11] Suphamit Chittayasothorn. 2001. "Database Design and SQL". NECTEC:Thailand.

[12] S.Y.Liao, et. al. "Transformation from Relational Model to O-O Model: A Reengineering Approach". Hong Kong.

[13] Theo Harder, et. al. 1997. "Supporting Adaptable Technical Information System in Heterogeneous Environment Using WWW and ORDBMS". Proceeding EIGHT International Workshops on Database and Expert System Application.

[14] V.Srinivasan, D.T. Chang. 1997. "Object persistence in object-oriented applications". IBM Systems Journal. 36(1).

IEEE SOUTHEASTCON 2003

TECHNICAL SESSIONS

Friday, April 4, 2003

(3:00pm - 4:30pm)

Session 1A.1 - Hanover (North)

REMOTE DATA ACQUISITION, CONTROL AND ANALYSIS USING LabVIEW FRONT PANEL AND REAL-TIME ENGINE

N. Swain, J. Anderson, A. Singh, M. Swain,
M. Fulton, J. Garrett, O. Tucker
School of Engineering Technology & Sciences
South Carolina State University

Session 1A.2 - Hanover (North)

OPEN-SOURCE SOFTWARE COMPONENTS FOR REAL -TIME LINUX TESTS AND EVALUATION WITH A MOBILE ROBOT APPLICATION

M. Silly, T. Garcia, C. Plot
University of Nantes, France

Session 1A.3 - Hanover (North)

K-MEANS AND ITERATIVE THRESHOLD SELECTION ALGORITHMS IN IMAGE SEGMENTATION

Chih-Cheng Hung
School of Computing and Software Engineering Southern Polytechnic State University, Marietta, GA

Glynn Germany
Center for Space and Plasma Research
University of Alabama

Session 1A.4 - Hanover (North)

A TRANSFORMATION FROM AN OBJECT DATABASE TO AN OBJECT RELATIONAL DATABASE

K. Niyomthum
Information Science, Faculty of Information Technology
King Mongkut' Institute of Technology
Bangkok

S. Chittayasothorn
Faculty of Engineering
King Mongkut' Institute of Technology
Bangkok, Thailand

The titles of technical papers accepted for presentation at SoutheastCon 2003.

[Conference Program]

[HOME] [CALL FOR PAPERS] [HOSTS] [REGISTRATION] [TRAVEL & ACCOMMODATION] [STUDENT CONFERENCE] [TECHNICAL CONFERENCE] [FINAL DATES] [SOCIAL EVENTS]

(m.moore@ieee.org)
Webmaster
(created:17-Mar-2003)

ประวัติผู้เขียน

นายกอบชัย นิยมธรรม เกิดเมื่อวันที่ 21 สิงหาคม 2512 ที่จังหวัดนครปฐม สำเร็จการศึกษาครุศาสตร์บัณฑิต (คอมพิวเตอร์ศึกษา) จากสถาบันราชภัฏเชียงใหม่ ปีการศึกษา 2534

ปี 2538 เข้ารับราชการในตำแหน่งนักวิชาการคอมพิวเตอร์ 4 ระดับ 4 สังกัดกองแผนงาน สำนักงานอธิการบดี มหาวิทยาลัยแม่โจ้ รับผิดชอบงานวางแผนการฝึกอบรมคอมพิวเตอร์และสารสนเทศ

ปี 2539 ช่วยราชการกองบริการการศึกษา เพื่อพัฒนาและปรับปรุงระบบสารสนเทศนิสิตนักศึกษา ปี 2542 ได้รับอนุมัติให้ลาศึกษาต่อในระดับปริญญาโท สาขาเทคโนโลยีสารสนเทศ ณ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง โดยทุนพัฒนาอาจารย์สาขาขาดแคลน สาขาคอมพิวเตอร์และสารสนเทศ ระดับปริญญาโท-เอก ปัจจุบันเป็นข้าราชการอาจารย์ สังกัดภาควิชาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ มหาวิทยาลัยแม่โจ้