การพัฒนาภาษา SL/NIAM และเครื่องมือสำหรับออกแบบฐานข้อมูลเชิงวัตถุ และฐานข้อมูลเชิงสัมพันธ์

THE DEVELOPMENT OF SL/NIAM LANGUAGE AND DATABASE DESIGN TOOL FOR OBJECT-ORIENTED AND RELATIONAL DATABASE

เทวิน ธนะวงษ์

TAWIN TANAWONG

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต

สาขาวิชาวิศวกรรมไฟฟ้า

บัณฑิตวิทยาลัย

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ. 2544

ISBN 974-648-083-9

การพัฒนาภาษา SL/NIAM และเครื่องมือสำหรับออกแบบฐานข้อมูลเชิงวัตถุ และฐานข้อมูลเชิงสัมพันธ์

THE DEVELOPMENT OF SL/NIAM LANGUAGE AND DATABASE DESIGN TOOL FOR OBJECT-ORIENTED AND RELATIONAL DATABASE

เทวิน ธนะวงษ์

TAWIN TANAWONG

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต

สาขาวิชาวิศวกรรมไฟฟ้า

บัณฑิตวิทยาลัย

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ. 2544

ISBN 974-648-083-9

THE DEVELOPMENT OF SL/NIAM LANGUAGE AND DATABASE DESIGN TOOL FOR OBJECT-ORIENTED AND RELATIONAL DATABASE

TAWIN TANAWONG

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
OF THE REQUIREMENT FOR THE DEGREE OF
MASTER OF ENGINEERING IN ELECTRICAL ENGINEERING
SCHOOL OF GRADUATE STUDIES
KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG
2001

ISBN 974 – 648 – 083 – 9

บัณฑิตวิทยาลัย

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ใบรับรองวิทยานิพนธ์

หัวข้อวิทยานิพนธ์ การพัฒนาภาษา SL/NIAM และเครื่องมือสำหรับออกแบบฐานข้อมูลเชิงวัตถุและฐานข้อมูลเชิงสัมพันธ์

THE DEVELOPMENT OF SL/NIAM LANGUAGE AND DATABASE DESIGN TOOL FOR OBJECT-ORIENTED AND RELATIONAL DATABASE

ชื่อนักศึกษา นายเทวิน ธนะวงษ์

รหัสประจำตัว 41061149

ปริญญา วิศวกรรมศาสตรมหาบัณฑิต

สาขาวิชา วิศวกรรมไฟฟ้า

อาจารย์ผู้ควบคุมวิทยานิพนธ์ รศ.ดร.ศุภมิตร จิตตะยโศธร

คณะกรรมการสอบวิทยานิพนธ์		ลายมือชื่อ
ดร.วิศิษฏ์	หิรัญกิตติ	
ผศ.ดร.บุญธีร์	เครือตราชู	
ดร.วรวัฒน์	ลิ้มโภคา	
ผศ.ดร.เอื้อน	ปิ่นเงิน	
รศ.ดร.ศุภมิตร	จิตตะยโศธร	

วัน/เดือนปี ที่สอบ 5 กุมภาพันธ์ 2544 เวลา 12.00-13.00 น.

สถานที่สอบ ณ อาคาร 12 ชั้น ชั้น 4 (ห้อง E12-402)

บัณฑิตวิทยาลัยรับรองแล้ว

(รศ.ดร.บุญวัฒน์ อัตชู)

คณบดีบัณฑิตวิทยาลัย

วันที่ ๙ **เดือน** มีนาคม **พ.ศ.** ๒๕๔๔

หัวข้อวิทยานิพนธ์	การพัฒนาภาษา SL/NIAM และเครื่องมือสำหรับออกแบบฐานข้อมูลเชิงวัตถุและฐานข้อมูลเชิงสัมพันธ์
นักศึกษา	นายเทวิน ธนะวงษ์
รหัสประจำตัว	41061149
ปริญญา	วิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชา	วิศวกรรมไฟฟ้า
พ.ศ.	2544
อาจารย์ผู้ควบคุมวิทยานิพนธ์	รศ.ดร.ศุภมิตร จิตตะยโศธร

บทคัดย่อ

เครื่องมือสำหรับช่วยในการพัฒนาและออกแบบฐานข้อมูลในปัจจุบันได้มีการพัฒนาไปอย่างรวดเร็ว โดยเครื่องมือส่วนใหญ่ที่ใช้สำหรับออกแบบฐานข้อมูลเหล่านี้มีพื้นฐานทางด้าน ER โมเดล ซึ่งมีการออกแบบของสกีม่าอยู่ในลักษณะรูปภาพ และผลที่ได้เป็นเพียงภาษานิยามข้อมูลของฐานข้อมูลเชิงสัมพันธ์เท่านั้น

สำหรับวิทยานิพนธ์ฉบับนี้ได้นำเสนอการออกแบบและพัฒนาเครื่องมือสำหรับออกแบบฐานข้อมูลเชิงวัตถุ โดยมีรูปแบบการนำเข้าข้อมูลในลักษณะภาษากึ่งธรรมชาติ ผลที่ได้จะอยู่ในรูปลักษณะของภาษานิยามข้อมูลของฐานข้อมูลเชิงวัตถุและฐานข้อมูลเชิงสัมพันธ์ รวมทั้งกฎข้อบังคับความถูกต้องของข้อมูลในรูปแบบ Stored Procedures แล้วนำผลที่ได้เหล่านี้ไปพัฒนาระบบเป็นโครงสร้างฐานข้อมูลเชิงวัตถุ

Thesis Title	The Development of SL/NIAM Language and Database Design Tool for Object-Oriented and Relational Database
Student	Mr. Tawin Tanawong
Student ID.	41061149
Degree	Master of Engineering
Programme	Electrical Engineering
Year	2001
Thesis Advisor	Assoc. Prof.Dr. Suphamit Chittayasothorn

ABSTRACT

Nowadays, most database design tools are based on the Entity Relationship model (ERM). The design schema is in graphical ER format and the output schema is in relational database data definition language.

This thesis presents the design and development of an object-oriented database design tool. The input schema is in the form of fact-based natural language sentences. Output can be in relational database data definition language or object-oriented database data definition language. Integrity rules can also be included into relational stored procedures or object methods.

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงได้อย่างดี ด้วยคำแนะนำและคำปรึกษาจาก รองศาสตราจารย์ ดร. ศุภมิตร จิตตะยโศธร ซึ่งเป็นอาจารย์ผู้ควบคุมวิทยานิพนธ์ ผู้วิจัยรู้สึกซาบซึ้งในความอนุเคราะห์จากท่านและขอกราบขอบพระคุณเป็นอย่างสูง

และต้องขอขอบพระคุณบุคคลสำคัญที่สุดในชีวิตที่ทำให้ข้าพเจ้ามีวันนี้ ก็คือบิดามารดา อันเป็นที่เคารพรักยิ่ง และพี่น้องของผู้วิจัย ที่ได้ให้การสนับสนุนการทำวิจัยในครั้งนี้ อีกทั้งเป็นบุคคลที่คอยสร้างกำลังใจแก่ผู้วิจัยเสมอมา ข้าพเจ้าขอระลึกในพระคุณอันสูงสุด และขอกราบขอบพระคุณมา ณ ที่นี้

ขอขอบพระคุณ นายแพทย์กมล กลิ่นไทย และ พยาบาลนงนุช บุญอยู่ เพื่อนที่แสนดีที่คอยให้กำลังใจ และให้ความช่วยเหลือแก่ผู้วิจัยอย่างสม่ำเสมอ

ขอขอบพระคุณ คุณกานต์รวี โกมลดิษฐ์ หัวหน้าห้องสมุดคณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ที่ได้ให้ความช่วยเหลือผู้วิจัยในการทำวิทยานิพนธ์เป็นอย่างดี

ขอขอบพระคุณ คุณปรีชา สุทธิธารธวัช และคุณจุฑาภรณ์ เทียนทอง เจ้าหน้าที่แผนกไมโครคอมพิวเตอร์และสารสนเทศ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ที่ได้เอื้อเฟื้ออุปกรณ์ใช้งานให้แก่ผู้วิจัยในการทำวิทยานิพนธ์เป็นอย่างดี

ขอขอบพระคุณบริษัทออราเคิล (ประเทศไทย) จำกัด ที่เอื้อเฟื้อเรื่องเอกสารในการใช้งาน Oracle8.0.5 บนระบบปฏิบัติการลีนุกซ์

นายเทวิน ธนะวงษ์

สารบัญ

หน้า

สารบัญ(ต่อ)

หน้า

สารบัญ(ต่อ)

หน้า

สารบัญ(ต่อ)

หน้า

สารบัญตาราง

สารบัญรูป

สารบัญรูป(ต่อ)

รูปที่ หน้า

สารบัญรูป(ต่อ)

รูปที่ หน้า

สารบัญรูป(ต่อ)

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

เครื่องมือสำหรับช่วยในการพัฒนาและออกแบบฐานข้อมูลในปัจจุบันได้มีการพัฒนาไปอย่างรวดเร็วมาก โดยเครื่องมือส่วนใหญ่ที่ใช้สำหรับออกแบบฐานข้อมูลเหล่านี้มีพื้นฐานทางด้าน ER โมเดล ซึ่งมีการออกแบบสกีมม่าอยู่ในลักษณะรูปภาพและผลที่ได้เป็นเพียงภาษานิยามข้อมูลของฐานข้อมูลแบบเชิงสัมพันธ์เท่านั้น นอกจากนี้การออกแบบฐานข้อมูลในหลักการเดิม ๆ เหล่านี้นั้นทำให้ผู้ทำการออกแบบจำเป็นจะต้องได้รับการศึกษาวิชาการออกแบบฐานข้อมูลและมีทักษะในการปฏิบัติงานมากพอสมควรจึงจะสามารถทำการออกแบบฐานข้อมูลได้อย่างมีความถูกต้องและตรงวัตถุประสงค์ของการใช้งานตามที่ต้องการ สำหรับเครื่องมือออกแบบฐานข้อมูลส่วนใหญ่เน้นการออกแบบรีเลชั่นไม่เน้น Integrity Rules หากการแก้ปัญหาดังกล่าวนี้ได้ก็สามารถทำให้ผู้ใช้สามารถทำการออกแบบฐานข้อมูลเองได้ สำหรับงานวิจัยที่เกี่ยวข้องกับหน่วยรับข้อมูลในรูปแบบภาษาที่ผ่านมานั้นในกรณีของ James A. Sykes ในปี1994 จะมีการสร้างภาษาธรรมชาติโดยใช้แนวคิดจากไนแอม ซึ่งรูปแบบโครงสร้างของภาษาจะเป็นแบบโครงสร้างผิวมีรูปประโยคข้อเท็จจริงแบบง่าย ๆ ที่ประกอบด้วยกริยาเพียง 1 ตัว สำหรับงานวิจัยของ Clare Atkins และ Jon Patrick ในปี 1998 จากมหาวิทยาลัย Massey ได้ทำการสร้างผลงานที่ต้องการให้ผู้ออกแบบระบบงานมีความเข้าใจอย่างถ่องแท้ในรายละเอียดของER โมเดล จึงได้ทำการสร้างภาษา NaLER (Natural language for ER) ซึ่งผลงานชิ้นนี้สิ่งที่ได้จะเป็นเพียงแค่เฟิร์สนอร์มัลฟอร์ม ซึ่งจะอยู่ในระดับข้อเท็จจริงของการทำไนแอมนอกจากนี้รูปแบบภาษาที่ใช้อธิบายความสัมพันธ์ของข้อมูลในER โมเดลระดับง่าย ๆ นอกจากนี้มีผลงานวิจัยที่เกี่ยวข้องกับการเปลี่ยนแปลงไนแอมอยู่ในรูปรีเลชั่นสกีม่าซึ่งเป็นผลงานของ Yanchun Zhang และ Maria E. Orlowska ในปี 1990 โดยแสดงที่มาของงานวิจัยก็คือ เนื่องจากปัญหาที่เกิดขึ้นกับการออกแบบฐานข้อมูลเชิงความสัมพันธ์ เช่นความซับซ้อนของข้อมูล การวิเคราะห์ข้อมูลที่ไม่มีแบบแผนที่ชัดเจน ทำให้เกิดความไม่สมบูรณ์ของฐานข้อมูลที่ได้ การขาดความหลากหลายในกฎข้อบังคับความถูกต้องของข้อมูล นอกจากนี้ยังมีงานวิจัยที่เกี่ยวข้องกับการแปลงรูปแบบของออปเจกต์เป็นตารางฐานข้อมูลนั่นก็คืองานวิจัยของ Wenny J Rahayu และ Elizabeth Chang จากมหาวิทยาลัย Swinburnee โดยมีแนวคิดที่ทำการแบ่งประเภทของการแปลงรูปออกเป็น 4 ประเภทโดยอาศัยความหมายที่เกิดจากความสัมพันธ์ของข้อมูล แต่ละประเภทที่ทำการแปลงเป็นตารางนั้นยังไม่ได้รูปแบบของตารางที่อยู่ในระดับที่เหมาะสมที่สุดนั่นก็คือยังไม่ถึงในระดับ 5NF นอกจากนี้ปัญหารูปแบบของการลดหรือเพิ่มคีย์หลักในตารางยังไม่มีความแน่นอน เป็นต้น

นอกจากนี้การพัฒนาระบบงานฐานข้อมูลเชิงวัตถุกำลังเป็นที่นิยม งานวิจัยชิ้นนี้จึงได้นำเสนอเครื่องมือออกแบบฐานข้อมูลโดยสามารถทำการแปลง Relational Database เป็น Object Database ได้ โดยทำการพัฒนาภาษาสำหรับออกแบบฐานข้อมูล SLไนแอม ด้วย Compiler Generator แล้วทำการออกแบบตารางเมทต้าเพื่อจัดเก็บข้อมูลและกฎข้อบังคับความถูกต้องของข้อมูลสำหรับใช้ในการออกแบบฐานข้อมูล ซึ่งข้อมูลในตารางเมทต้านี้จะถูกนำเข้าโดยผู้ใช้ภาษา SL ไนแอมแล้วทำการพัฒนา Table Definition, Store Procedures และ Database Trigger โดยใช้หลักการของอัลกอริธึม ONF (Optimal Normal Form) ในการแมปข้อมูลในตารางเมทต้าไปอยู่ในรูปแบบของรีเลชั่นสกีม่า แล้วนำผลที่ได้เหล่านี้สร้างเป็น Object-Oriented Database

1.2 ความมุ่งหมาย และวัตถุประสงค์ของการศึกษา

1.2.1 ศึกษาการออกแบบภาษากึ่งธรรมชาติเพื่อใช้ออกแบบฐานข้อมูลแทนหลักการเดิม ๆ ที่ใช้รูปสัญลักษณ์ เพื่อช่วยให้ผู้ที่สนใจจะทำการออกแบบฐานข้อมูลมีความสะดวกและไม่จำเป็นต้องมีพื้นฐานทักษะของวิชาออกแบบฐานข้อมูลมากพอสมควรในการปฏิบัติงาน

1.2.2 ศึกษาการจัดเก็บข้อมูลจากกฎข้อบังคับความถูกต้องต่างๆ จากรูปประโยคภาษากึ่งธรรมชาติที่ได้ทำการพัฒนาขึ้นมาให้อยู่ในตารางเมทต้า

1.2.3 ศึกษาแนวทางการพัฒนาข้อมูลในฐานข้อมูลเมทต้าให้อยู่ในรูปของ Relational Schema ที่ใช้หลักการ ONF

1.2.4 ศึกษาเปรียบเทียบการทำงานในลักษณะฐานข้อมูลเชิงสัมพันธ์กับฐานข้อมูลเชิงวัตถุว่ามีข้อแตกต่างกันอย่างไรบ้าง และศึกษาแนวโน้มของการใช้ระบบฐานข้อมูลของทั้งสองแบบนี้ในอนาคต

1.2.5 ศึกษาการทำงานและการออกแบบระบบโครงสร้างการทำงานของระบบฐานข้อมูลเชิงวัตถุ

1.3 สมมติฐานของการศึกษา

งานวิจัยในการออกแบบภาษากึ่งธรรมชาตินี้จะอยู่อาศัยพื้นฐานกฎข้อบังคับความถูกต้องของข้อมูลเชิงไนแอม ซึ่งการออกแบบภาษาธรรมชาตินั้นถือว่ามีความยากมากเนื่องจากรูปแบบภาษาธรรมชาตมีรูปแบบที่ไม่แน่นอน การออกแบบภาษาของผู้วิจัยจะกระทำในลักษณะภาษาโครงสร้างลึกซึ่งรูปแบบของประโยคภาษานั้นบางครั้งจะไม่สื่อความหมายที่ชัดเจน การนำกฎข้อบังคับความถูกต้องของข้อมูลในรูปประโยคที่ได้เหล่านี้มาเก็บไว้ในตารางเมทต้านั้นจำเป็นจะต้องออกแบบตารางเมทต้าให้เหมาะสมพอที่จะครอบคลุมเนื้อทั้งหมดได้ ซึ่งบางครั้งอาจจะต้องมีการปรับปรุงและทำการเปลี่ยนแปลงบ่อยเพื่อให้ได้ความเหมาะสมมากที่สุดซึ่งในส่วนนี้ถือว่ามีสำคัญ

มากทีเดียวในการทำวิจัยครั้งนี้ การศึกษารูปแบบการพัฒนาระบบฐานข้อมูลเชิงสัมพันธ์โดยใช้อัลกอริธึม ONF นั้นอาจจะมีข้อมูลบางส่วนที่ไม่สามารถจะทำการพัฒนาขึ้นมาตามกฎเกณฑ์ที่วางไว้อาจจะจำเป็นต้องมีการเพิ่มรายละเอียดหรือกำหนดข้อบังคับขึ้นมาใหม่เพื่อให้การพัฒนาเป็นไปตามความเหมาะสมยิ่งขึ้น กรณีการทำงานในรูปแบบฐานข้อมูลเชิงวัตถุนั้นในกรณีนี้เป็นการแปลงรูปจากฐานข้อมูลเชิงสัมพันธ์ซึ่งจำเป็นจะต้องกำหนดรูปแบบระบบฐานข้อมูลทั้งสองให้สัมพันธ์กัน ซึ่งการวิจัยในครั้งนี้จะเพียงจะเน้นความสำคัญในลักษณะ Primary key และ Foreign key ในระบบฐานข้อมูลเชิงสัมพันธ์เพื่อสร้างรูปแบบการทำงานในลักษณะฐานข้อมูลเชิงวัตถุ จากกรณีนี้ถือว่าเป็นการใช้รายละเอียดข้อมูลพื้นฐานจริง ๆ ในการพัฒนา หากปัญหาที่เกิดขึ้นจากการแปลงรูปที่เกิดขึ้นไม่สามารถใช้ Primary key และ Foreign key ได้อาจจะจำเป็นจะต้องมีการกำหนดรูปแบบใหม่ในการพัฒนาต่อไปก็ได้ซึ่งอาจจะทำให้ขอบเขตการทำวิจัยของผู้ทำวิจัยกว้างมากเกินไป

1.4 ทฤษฎีหรือแนวคิดที่ใช้ในการวิจัย

การทำวิจัยในครั้งนี้เริ่มขึ้นจากการศึกษาการออกแบบฐานข้อมูลในรูปแบบวิธีของไนแอมและวิธีอื่น ๆ ซึ่งจากการศึกษาทั้งทฤษฎีและการได้ลองปฏิบัติงานจริงในการใช้โปรแกรมสำเร็จรูปในการออกแบบฐานข้อมูลจะเห็นได้ว่ายังมีข้อจำกัดของโปรแกรมอยู่ จึงมีแนวคิดที่จะเริ่มพัฒนารูปแบบการออกแบบฐานข้อมูลในลักษณะของภาษากึ่งธรรมชาติเพื่ออำนวยความสะดวกต่อผู้ปฏิบัติงาน โดยการออกแบบภาษาจะต้องศึกษาหลักไวยากรณ์ทางภาษาว่าจะออกแบบโดยใช้หลักไวยากรณ์ทางภาษาแบบใดที่จะเหมาะสมซึ่งความเหมาะสมนี้จะต้องเกี่ยวพันกับอุปกรณ์ที่ใช้ในการพัฒนาด้วย นอกจากนี้ได้มีการพัฒนาแนวคิดใหม่ ๆ ขึ้นมาใช้กับการออกแบบระบบฐานข้อมูลเชิงวัตถุจากรูปแบบฐานข้อมูลเชิงสัมพันธ์ โดยแนวคิดนี้ได้อาศัยหลักการพื้นฐานในเรื่องของไพรมารี่คีย์ และฟอเรนคีย์

1.5 ขอบเขตการวิจัย

การพัฒนาภาษา SL/NIAM และเครื่องมือสำหรับออกแบบฐานข้อมูลเชิงวัตถุและฐานข้อมูลเชิงสัมพันธ์ จะเริ่มต้นจากการพัฒนาภาษา SL/NIAM ด้วย Compiler Generator แล้วทำการออกแบบตารางเมทต้า เพื่อใช้จัดเก็บข้อมูลและกฎข้อบังคับความถูกต้องของข้อมูลสำหรับใช้ในการออกแบบระบบฐานข้อมูล ซึ่งข้อมูลในตารางเมทต้านี้จะถูกนำเข้าโดยผู้ใช้ภาษา SL/NIAM ทำการพัฒนา Table Definition, Store Procedures และ Database Trigger โดยใช้อัลกอริธึม ONF ปฏิบัติงานกับตารางฐานข้อมูลเมทต้า และทำการพัฒนา Data Structure ในรูปแบบของ Object

Database โดยใช้ข้อมูลจาก Relational Database ที่ได้ทำการออกแบบแล้วซึ่งในการพัฒนาในส่วนนี้ถือว่าเป็นส่วนสุดท้ายของผู้ทำวิจัยในครั้งนี้

1.6 ขั้นตอนของการศึกษา

งานวิจัยในครั้งนี้จะเริ่มด้วยการศึกษาทฤษฎีพื้นฐานต่าง ๆ ที่เกี่ยวข้องกับงานวิจัย ซึ่งมีเรื่องหลัก ๆ อยู่ 3 เรื่องด้วยกัน คือ ระบบฐานข้อมูลแบบเชิงสัมพันธ์และระบบฐานข้อมูลเชิงวัตถุ การออกแบบภาษากึ่งธรรมชาติ และการสร้างกฎข้อบังคับความถูกต้อง ซึ่งมีรายละเอียดดังในบทที่ 2, 3 และ 4 ตามลำดับ จากนั้นก็ศึกษาในเรื่องของการสร้าง Data Structure ของฐานข้อมูลเชิงวัตถุมีรายละเอียดในบทที่ 5 หลังจากนั้นได้นำมาออกแบบสถาปัตยกรรมระบบเพื่อให้ทราบถึงระบบงานทั้งหมดที่ทำงานวิจัยในครั้งนี้ มีรายละเอียดบทที่ 6 สำหรับกรณีขั้นตอนของการพัฒนาภาษา SL/NIAM และเครื่องมือสำหรับออกแบบฐานข้อมูลเชิงวัตถุและฐานข้อมูลเชิงสัมพันธ์จะกล่าวไว้ในบทที่ 7

จากนั้นก็จะเริ่มเข้าสู่ขั้นตอนของการทดสอบและการใช้งานระบบจะกล่าวในบทที่ 8 ซึ่งจะอธิบายถึงวิธีการใช้งานจากระบบที่ได้พัฒนาเสร็จเรียบร้อยแล้ว มีการทดลองใช้ตัวอย่างข้อมูลจริงในการทดสอบ และแสดงผลลัพธ์จากการทดสอบระบบงานจริง ๆ สุดท้ายจะเป็นการสรุปผลการวิจัยและข้อเสนอแนะจะกล่าวไว้ในบทที่ 9

บทที่ 2
ทฤษฎีและหลักการของระบบฐานข้อมูล

2.1 ฐานข้อมูลแบบรีเลชั่นแนล

แบบจำลองข้อมูลเชิงสัมพันธ์เป็นระบบฐานข้อมูลแรกที่มีพื้นฐานมาจากแบบจำลองข้อมูลแบบ hierarchical หรือแบบจำลองข้อมูลแบบเน็ตเวิร์ก แบบจำลองข้อมูลเชิงสัมพันธ์เป็นการนำเสนอฐานข้อมูลในลักษณะตาราง โดยจะมีลักษณะของตารางแบบง่าย ๆ ซึ่งจะมีแนวคิดระหว่างตารางและแนวคิดทางคณิตศาสตร์ในเรื่องของ รีเลชั่น (relation)

คุณสมบัติของรีเลชั่น จะเป็นกลุ่มทางคณิตศาสตร์ที่นำเสนออยู่ในรูปของตาราง แบบจำลองข้อมูลที่มีพื้นฐานทางด้านรีเลชั่น และมีการนำเสนออยู่ในรูปตารางถูกนำเสนอโดย Codd เป็นคนแรก โดยรูปแบบจำลองข้อมูลเชิงสัมพันธ์จะอาศัยทฤษฎีความสัมพันธ์ทางคณิตศาสตร์ เพื่อวัตถุประสงค์ก็คือใช้สำหรับจัดการทางด้านข้อมูล

2.1.1 โครงสร้างข้อมูล

data structure ถือเป็นเครื่องมือหนึ่งที่ใช้ในการทำเป็นแบบจำลองข้อมูลเชิงสัมพันธ์ในลักษณะรีเลชั่น โดยการกำหนดนิยามของรีเลชั่นในรูปแบบจำลองข้อมูลเชิงสัมพันธ์ได้กำหนดในรูปทางคณิตศาสตร์

รีเลชั่นสกีมม่า (relational schema) เป็นการกำหนดพื้นฐานของข้อมูลเชิงสัมพันธ์ที่ประกอบไปด้วย รีเลชั่น สกีมม่า หนึ่งหรือมากกว่านั้น ซึ่งรีเลชั่น สกีมม่า ก็คือเป็นการแสดงรายชื่อของรีเลชั่น โดยมีความสอดคล้องกับชื่อโดเมน

รีเลชั่นสกีมม่าสามารถที่จะนำมาใช้นำเสนอประเภทเอนติตี้ (entity type) ในแบบจำลองข้อมูลเชิงสัมพันธ์ตั้งแต่แบบจำลองข้อมูลเชิงสัมพันธ์ทำงานเป็นแบบจำลองข้อมูลในลักษณะตาราง จะเห็นได้ว่า รีเลชั่นสกีมม่า ไม่มีผลกระทบมีความชัดเจน ทุกประเภทของความสัมพันธ์ระหว่างรีเลชั่น ในพื้นฐานของข้อมูลมีบางประเภทของความสัมพันธ์ (relationship type) ที่ยังมีความไม่ชัดเจนในสกีมม่า

ประเภทของความสัมพันธ์ (relationship type) สามารถนำเสนอโดยแต่ละ key propagation หรือ รีเลชั่นสกีมม่า ในรูปของ รีเลชั่นสกีมม่า อย่างไรก็ตามเพราะว่าความหมายของประเภทของความสัมพันธ์ซึ่งปกติจะเป็น one-to-one ขณะที่ many-to-many ในประเภทของความสัมพันธ์ ถูกนำเสนอโดยการแยก “relationship” , “relations”

extension ของพื้นฐานข้อมูลเชิงสัมพันธ์จะถูกนำเสนอในรูปแบบตาราง โดยคอลัมน์ของตารางจะเรียกว่า แอทริบิ้ว (attribute)

2.1.2 คอนสเตรนท์ในรีเลชั่น

database relation เป็นพื้นฐานทางคณิตศาสตร์ซึ่งเป็นการถ่ายทอดทางคุณสมบัติของเซ็ต (set) สิ่งแรกก็คือ ตารางที่ได้นำเสนอของ database relation จะไม่มีการซ้ำของแถว ซึ่งในความเป็นจริงแล้วการซ้ำแถว จะไม่อนุญาตให้ซ้ำกัน นั่นหมายความว่าทุก ๆ รีเลชั่นอย่างน้อยจะมี 1 คีย์ อย่างไรก็ตามรีเลชั่น อาจจะมีคีย์ทั่ว ๆ ไป คุณสมบัติคีย์มีดังนี้คือ

p1. Unique Identification

ในแต่ละ tuple ของ รีเลชั่น จะมีการกำหนดค่าของ key unique นั่นก็คือ ไม่มี 2 แถวใด ๆ ที่จะมีค่าของแอทริบิ้วที่เป็นคีย์ซ้ำกัน

p2. Nonredundancy

ไม่มีแอทริบิ้วที่เป็นส่วนหนึ่งของคีย์โดยการเคลื่อนย้ายปราศจากการเสียหายทางด้านคุณสมบัติ p1 นั่นก็คือ เป็นคีย์ น้อยที่สุด

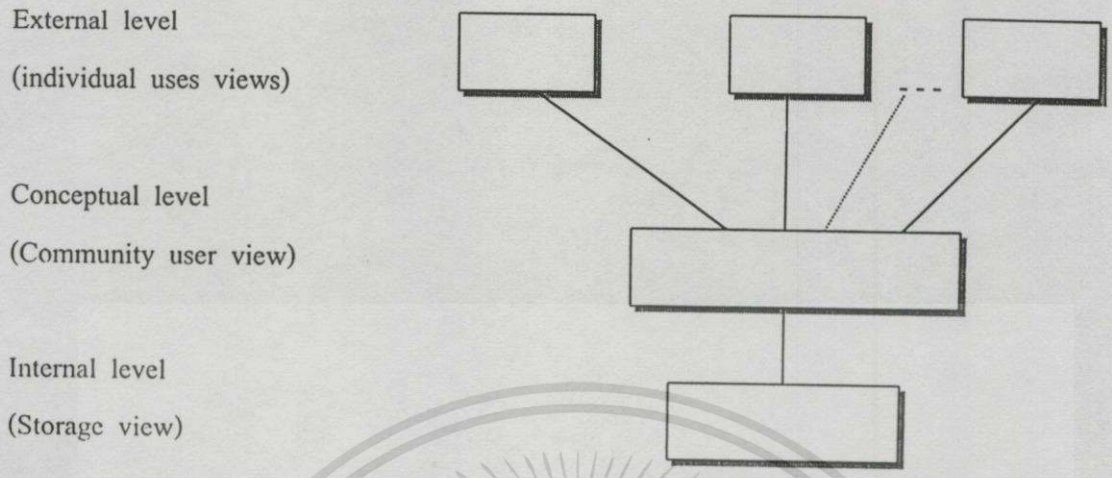
คุณสมบัติคีย์ของ database relations สามารถจะแสดงในรูปของฟังก์ชั่น นั่นก็คือ ถ้าเซ็ตของแอทริบิ้ว X เป็นคีย์ของรีเลชั่น R ค่าของ X จะเป็นของทุก ๆ แอทริบิ้วใน tuple ของ R

ในรีเลชั่นสกีมม่า หนึ่งใน candidate keys จะถูกคัดเลือกเป็น primary key โดย primary key จะแตกต่างจากคีย์อื่นๆ ในเรื่องของการยอมให้ทำงาน (ตัวอย่างเช่น ไม่สามารถทำการแก้ไขได้) และนอกจากนี้ไม่มีค่าของ primary key ในรีเลชั่นที่จะอนุญาตให้เกิดค่าว่าง (null) หรือมีส่วนประกอบเป็นค่าว่าง

2.1.3 สถาปัตยกรรมระบบฐานข้อมูล

สถาปัตยกรรมระบบฐานข้อมูลแสดงดังรูปที่ 2.1 สามารถทำการแบ่งได้ 3 แบบคือ

1. ระดับ internal เป็นระดับที่เก็บข้อมูลในระดับกายภาพ
2. ระดับ external เป็นระดับที่ผู้ใช้ทำงาน
3. ระดับ conceptual เป็นระดับที่อยู่ระหว่าง internal และ external ซึ่งในระดับนี้ไม่มีทิศทาง



รูปที่2.1 แสดงสถาปัตยกรรมในระบบฐานข้อมูล

2.1.4 ระบบเชิงความสัมพันธ์

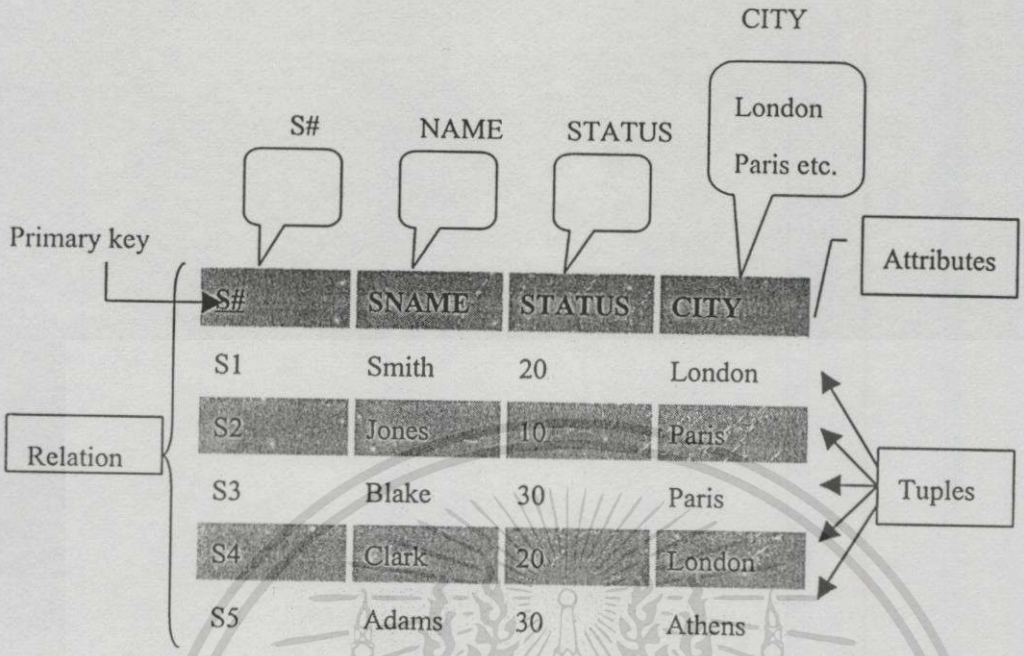
มีคำถามว่า relational systems คืออะไร นี่เป็นจุดเริ่มต้นที่เราจะต้องทำความรู้จักและหาคำตอบ ซึ่งเราสามารถอธิบายอย่างย่อ ๆ ได้ว่า relational system ก็คือ

1. ข้อมูลที่ถูกใช้แสดงด้วยตาราง
2. เป็นลักษณะที่ผู้ใช้จะกระทำ เช่น การเรียกค้นข้อมูล เป็นต้น

2.1.5 โครงสร้างข้อมูลเชิงสัมพันธ์ (Relational Data Structure)

2.1.5.1 แนะนำตัวอย่าง

ตัวอย่างรูปที่ 2.2 เป็นการแสดงความสัมพันธ์ในรูปแบบที่ใช้ความสัมพันธ์ S จาก suppliers โดยแยกเป็นส่วนฐานข้อมูล โดยแต่ละส่วนจะเป็น domain, attribute, tuple, primary key และ relation



รูปที่ 2.2 แสดงรีเลชั่น supplier ในตาราง S

2.1.5.2 โดเมน (DOMAINS)

เป็นหน่วยเล็กที่สุดของข้อมูลในโมเดลเชิงสัมพันธ์มีค่าของข้อมูลเฉพาะตัว ตัวอย่างเช่น จำนวน supplier หรือ ปริมาณชิป ซึ่งมีค่าในการพัฒนาเป็นอะตอมซึ่งไม่สามารถแยกได้อีกแล้ว โดเมนเป็นกลุ่มของค่าเหล่านี้ จากตัวอย่างโดเมนของจำนวน supplier ซึ่งมีความเป็นไปได้เกี่ยวกับ supplier โดเมนของชิปซึ่งเป็น integer มากกว่าศูนย์และน้อยกว่า 10,000 เป็นต้น

2.1.5.3 รีเลชั่น (RELATIONS)

Relation เกิดขึ้นในโดเมน D1, D2,...,Dn (ไม่จำเป็นต้องแตกต่างกันทั้งหมด) จะประกอบไปด้วย heading และ body

- heading ประกอบไปด้วยกลุ่มของแอทริบิ้ว A1, A2, ...,An ซึ่งในแต่ละแอทริบิ้ว Ai จะอยู่ภายใต้เพียงโดเมนเดียวคือ Di (i=1,2,...,n)
- body ประกอบไปด้วย time-varying ของ tuple ในขณะที่แต่ละ tuple ประกอบด้วยกลุ่มของแอทริบิ้วเป็นคู่ ๆ (Ai: vi) (i=1,2,...,n) ในคู่หนึ่ง ๆ ของแอทริบิ้ว Ai จะเป็นหัวข้อ (heading) สำหรับค่าที่ได้จาก attribute-value เป็นคู่ ๆ vi เป็นค่าที่ได้มาจากโดเมน Di ที่สัมพันธ์กับแอทริบิ้ว Ai (Ai : vi)

2.1.5.4 คุณสมบัติของรีเลชั่น (PROPERTIES OF RELATIONS)

คุณสมบัติที่สำคัญในการทำ relations จะเห็นได้ว่าความสำคัญในการกำหนด "relation" ซึ่งคุณสมบัติที่จะกำหนดนั้นมีอยู่ 4 ข้อคือ

1. ไม่มีความซ้ำซ้อนใน Tuples
2. Tuples จะไม่มีการเรียงลำดับ (จากบนไปล่าง)
3. แอทริบิ้วจะไม่มีการเรียงลำดับ (จากซ้ายไปขวา)
4. ค่าของแอทริบิ้วทั้งหมดเป็นหน่วยที่เล็กที่สุด

2.1.6 กฎข้อบังคับสอง 2 รูปแบบ (The Two Integrity Rules)

1. Entity integrity :

   ไม่มีแอทริบิ้วที่ทำหน้าที่เป็น primary key ในรีเลชั่นจะยอมให้มีค่าเป็นศูนย์

2. Referential integrity :

   ถ้ารีเลชั่น R2 ประกอบไปด้วย foreign key FK สอดคล้องกับ primary key PK ของบางรีเลชั่น R1 แล้วทุก ๆ ค่าของ FK ใน R2 จะต้องมีค่าเท่ากับ PK ในบาง tuple ของ R1 หรือเป็น null (ตัวอย่างเช่นในแต่ละแอทริบิ้วใน FK จะต้องเป็น null) R1 และ R2 ไม่จำเป็นต้องแตกต่างกัน

2.1.7 รีเลชั่นสกีมม่า (Relational Schema)

ฐานข้อมูลเชิงสัมพันธ์เป็นการรวบรวมรีเลชั่นเข้าไว้ด้วยกันในลักษณะ schema ลักษณะ schema ในรีเลชั่น R เขียนได้ดังนี้คือ

$$R : \{|A_1 : D_1, \ldots, A_n : D_n|\}$$

R เป็นชื่อของรีเลชั่น และมี $A_1, \ldots A_n$ เป็นชื่อของแอทริบิ้วในแต่ละคู่จะมีความแตกต่างกัน ในแต่ละแอทริบิ้ว A_i จะมีค่ามาจากโดเมนของ D_i ซึ่งโดเมน $D_1, \ldots, D_n$ จะไม่มีความแตกต่างกัน ตัวอย่างเช่น PhoneBook โดยในเครื่องหมาย [...] เป็นการแสดงถึง tuple เดี่ยว ๆ และใน {...} แสดงกลุ่มของ tuples ซึ่งกลุ่มของ relational schemas ทั้งหมดจะประกอบกันเป็น database schema สำหรับค่าที่ได้ทำการจัดเก็บใน relational database schema เรียกว่า relational database

2.1.8 การแมปคอบเซ็บชวลสกีมม่าให้อยู่ในรูปสกีมม่าของฐานข้อมูลเชิงสัมพันธ์ (Mapping a Conceptual Schema into a Relational Database Schema)

Database schema ได้ถูกพัฒนาในลักษณะแนวความคิดของแอปลิเคชั่นฐานข้อมูลที่ประกอบกันเป็นการออกแบบ relational schema เราสามารถทำการแปลง entity-relationship schema ไปเป็น relational schemaได้ แต่สำหรับการแปลงจาก relational schema ให้อยู่ในรูปของ ER- schema จะมีความยาก ซึ่งโดยทั่วไปโมเดล ER สามารถอธิบายความสัมพันธ์ของข้อมูลไปสู่รูปแบบ relational schema ได้ง่ายกว่าการอธิบายความสัมพันธ์จาก relational schemaไปสู่โมเดล ER

2.1.8.1 แสดงรูปแบบเอนติตี้ในรีเลชั่น (Representing Entity in Relations)

ในแต่ละ entity ได้กำหนดอยู่ในรูปแบบ ER schema เป็นโมเดลที่จัดการในด้าน relation โดยทั่ว ๆ ไป entity set E มีแอทริบิ้ว A_1 ,...,A_n ถูกแมปให้อยู่ในรูป relational schema

E: { [A_1 : D_{A1} ,...,A_n : D_{An}]}

D_{Ai} โดยที่ ($a \leq i \leq n$) เป็นการชี้ถึงโดเมนในแต่ละแอทริบิ้วของ A_i

ตัวอย่าง conceptual schema ของรูปที่ 2.3 สามารถแสดงรีเลชั่นของ relational database schema ได้ดังนี้คือ

Divisions : {[Name :string, Location :string, Profit :money]}

Products : {[PID :string, Description :string, Price :money]}

Tools : {[TID :string, Description :string, Precision :real]}

Robots : {[RID :string, LoadCapacity :real, ReachRadius :real]}

Engineers : {[SS# :string, LastName :string, FirstName :string]}

การนำเสนอเพิ่มเติมกับอีก 6 รีเลชั่น ในรูปของ relationship type ดังนี้คือ

WorksFor : {[DivisionName:string, EngineerSS#:string]}

กรณีการปฏิบัติตามกฎรีเลชั่น จะได้รูปแบบดังนี้คือ

WorksFor : {[Name:string, SS#:string]}

สำหรับ relational schema ที่แสดงได้ดังนี้คือ

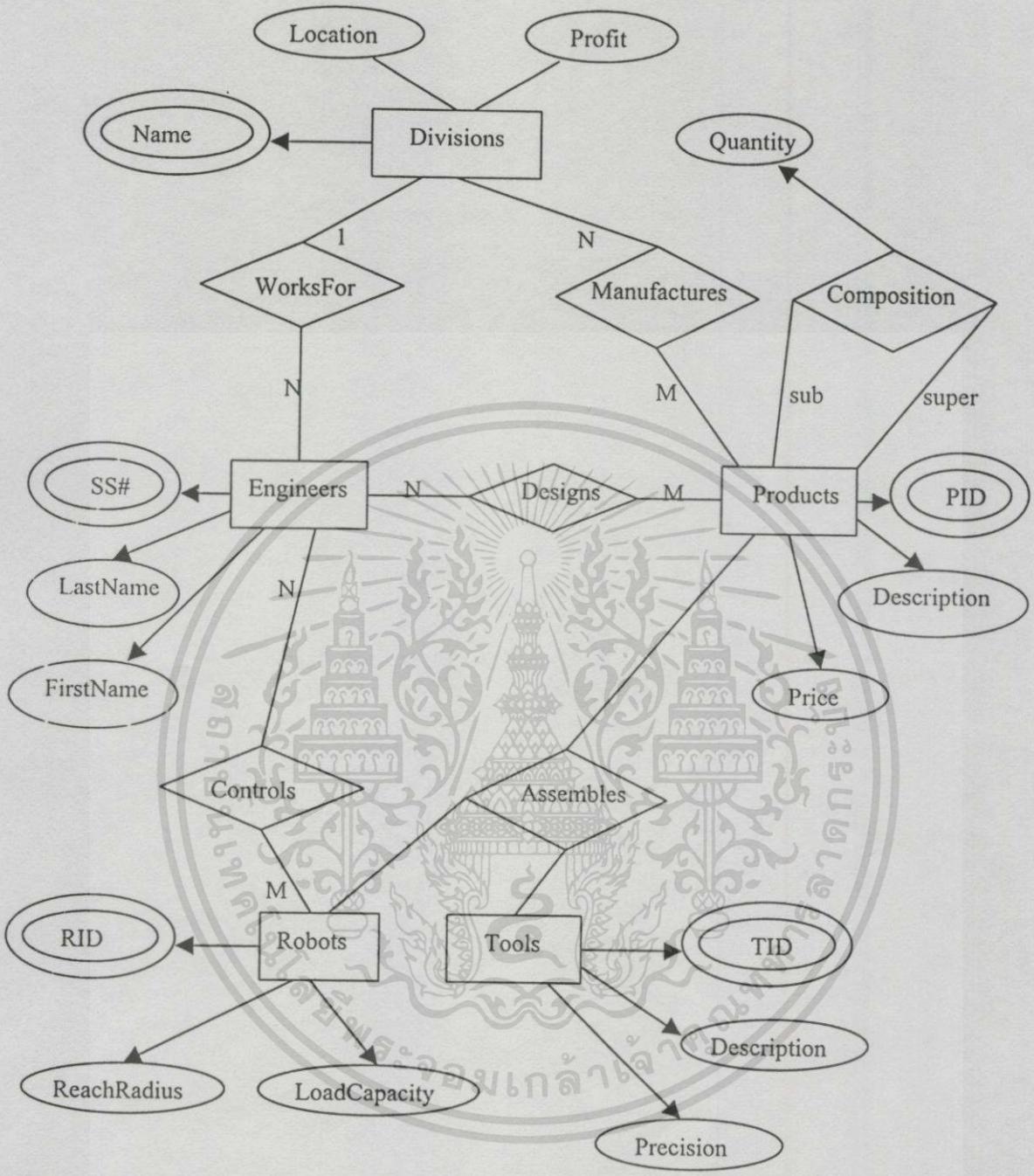
Designs :{[EngineerSS#:string, ProductID :string]}

Controls : {[EngineerSS#:string,RobotID:string]}

Manufactures: {[DivisionName:string, ProductID:string]}

Assembles : {[RobotID:string, ToolID : string, ProductID :string]}

Composition : {[SubPartID :string, superPartID :string, Quantity :integer]}



รูปที่ 2.3 แสดงคอนเซ็บชวลโมเดลของ Company

รีเลชั่นที่นำเสนอใน Designs, Controls และ Manufactures เป็นความสัมพันธ์แบบ N:M ตัวอย่างเช่น relation designs มีคีย์ {EngineerSS#, ProductID} ในขณะที่ EngineerSS# เป็น foreign key ที่อ้างอิงถึงรีเลชั่น Engineer และ ProductID อ้างถึงรีเลชั่น Products ในรูปความสัมพันธ์แบบเทอนารี่ (N:M:P) เช่น Assembles

สำหรับรีเลชั่นของ Composition ก็น่าสนใจเพราะเป็นโมเดลที่แสดงถึงเวลาความสัมพันธ์แบบวนซ้ำ (recursive) ซึ่งเป็นความสัมพันธ์เอนติตี้ของเอนติตี้เดียวกัน

2.2 ฐานข้อมูลเชิงวัตถุ

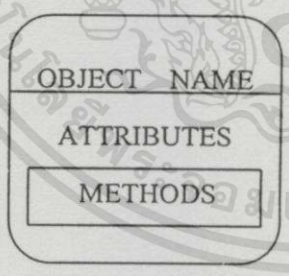
แนวคิดแบบเชิงออปเจกต์ เป็นแนวคิดที่รวมข้อมูลและวิธีการ โดยมีคลาสเป็นตัวกำหนดคุณสมบัติของออปเจกต์ ดังนั้นเราสามารถสรุปแนวคิดในลักษณะออปเจกต์ได้ดังต่อไปนี้

- กำหนดรายละเอียดในขั้นตอนการประมวลผลว่ามีอะไรบ้าง
- ออปเจกต์จะถูกกำหนดว่ามีสมาชิกหรือรายละเอียดของข้อมูลและฟังก์ชั่นที่เกี่ยวข้องต่าง ๆ นั้นมีอะไรบ้าง
- คลาสต่าง ๆ ที่นิยามสามารถนำมาใช้ใหม่โดยวิธีการถ่ายทอดคุณสมบัติของคลาส (Class Inheritance)
- โครงสร้างของโปรแกรมออปเจกต์ ขึ้นอยู่กับข่าวสาร (Message) ที่ได้ทำการส่งออกไปยังออปเจกต์เพื่อให้ทำงานตามที่ได้ระบุไว้

2.2.1 โครงสร้างของออปเจกต์ (Internal Structure of an Object)

โครงสร้างของออปเจกต์ ดังรูปที่ 2.4 ประกอบไปด้วยส่วนต่าง ๆ ดังนี้คือ

1. ชื่อออปเจกต์ (Object Name)
2. แอททริบิ้ว (Attributes)
3. เมทธอด (Methods)



รูปที่ 2.4 แสดงโครงสร้างของออปเจกต์

Object Name : เป็นการแสดงชื่อของออปเจกต์คลาส

Attributes : แสดงรายละเอียดต่าง ๆ ของออปเจกต์คลาสว่ามีแอทริบิ้วอะไรบ้าง

Methods : เป็นการเชื่อมโยงข่าวสารว่ามีการกระทำอะไรบ้าง

2.2.2 คำศัพท์ที่ใช้ในการสื่อสารข้อมูลเชิงวัตถุ

Object : ออปเจกต์เปรียบเหมือนรายละเอียดที่ใช้ในโปรแกรมทั่ว ๆ ไปที่มีการระบุถึงประเภทข้อมูลที่กำหนดให้ตัวแปรรวมถึงค่าของตัวแปรนั้น ๆ ด้วย หรือกล่าวอีกนัยหนึ่งออปเจกต์ก็คือ ตัวแปรของคลาส โดยที่มีรายละเอียดแอทริบิ้วของข้อมูล และขั้นตอนการทำงานหรือฟังก์ชั่นในออปเจกต์ที่เรียกว่า เมทธอด

Inheritance : เป็นการสืบทอดคุณสมบัติคลาสที่มีอยู่ในปัจจุบันมาเป็นคลาสใหม่ โดยที่คลาสใหม่นั้นจะสืบทอดคุณสมบัติของรายละเอียดและวิธีการต่าง ๆ จากคลาสเก่า

Encapsulation : เป็นคุณลักษณะของรายละเอียดของข้อมูล และวิธีการของออปเจกต์ที่รับให้ทราบถึงการที่จะได้มาซึ่งผลลัพธ์ โดยในระบบจะมีคัวประสานของออปเจกต์เป็นตัวจัดการให้ว่าต้องทำอะไรบ้าง

Association : เป็นการเชื่อมกันทางด้านกายภาพ หรือในด้านทางแนวคิดของแต่ละออปเจกต์เป็นกลุ่ม การจัดกลุ่มของออปเจกต์ที่มีความสัมพันธ์กันมากกว่าสองออปเจกต์ขึ้นไปก็จะเป็นกลุ่มที่มีโครงสร้างและความสัมพันธ์ร่วมกัน

Message passing : ออปเจกต์ต่าง ๆ ในแต่ละคลาสจะถูกสั่งให้ทำงานเมื่อมีการส่งข่าวจากออปเจกต์หนึ่งไปยังอีกออปเจกต์กนึ่งโดยตัวประสานหรือตัวเชื่อมข่าวสารจะแสดงเส้นทางการติดต่อสื่อสารระหว่างคลาส

Polymorphism : เป็นลักษณะเด่นของแนวคิดแบบออบเจ็กต์ก็คือ ความสามารถของฟังก์ชั่นหรือโอเปอเรชั่นต่าง ๆ กัน ใช้ชื่อเดียวกันได้โดยโปรแกรมออปเจกต์โอเรียนเตด จะแยกความแตกต่างของการใช้ชื่อที่เหมือนกันได้ด้วยจำนวนและชนิดของพารามิเตอร์

Object Identity : การระบุลักษณะเฉพาะของแต่ละออปเจกต์จะเป็นอิสระไม่ข้องเกี่ยวกันโดยวัตถุประสงค์ในการระบุนั้นสามารถอ้างอิงถึงออปเจกต์อื่นได้ เมื่อมีการรวบรวมหลาย ออปเจกต์ไอเด็นติตี้ แล้วหาความสัมพันธ์และสามารถจัดทำขึ้นเป็นโปรแกรมได้

การจัดเก็บข้อมูลของฐานข้อมูลเชิงวัตถุ สามารถอธิบายได้ดังนี้คือ

- Class families คือ คลาสที่จัดเก็บในสิ่งที่มีลักษณะเดียวกันซึ่งในคลาสใน class family จะอยู่ในรูปของ tree หรือเซ็ตของ tree

- Classes คือ ออปเจกต์ที่เก็บรวบรวมในฟีชเชอร์ (feature) ที่มีพื้นฐานเดียวกัน

- Objects/Instances คือ ส่วนของข้อมูลในคลาส ตัวอย่างเช่น แต่ละเดส (dress) ในเดสคลาส (dress class) เป็นออปเจกต์หรืออินสแตนท์ โดยในแต่ละออปเจกต์ประกอบด้วยรายละเอียดต่างๆ เช่น ราคา

- Method คือ ฟังก์ชั่นที่เราสามารถ จะทำการคำนวณค่าของข้อมูล สามารถทำการดูแลการเกิดความซ้ำซ้อนของข้อมูลได้ หรือทำการเข้าถึงรายละเอียดของข้อมูลที่อยู่นอกระบบฐานข้อมูลได้ เป็นต้น

- Subclass คือ คุณสมบัติที่ได้ทำการถ่ายทอด และเมทธอดของคลาส

เทคนิคในการจัดเก็บข้อมูลในฐานข้อมูลเชิงวัตถุจะต้องมีการบริหารหน่วยความจำใน OODBMS ให้เกิดประสิทธิภาพเพื่อรองรับลักษณะการทำงานดังต่อไปนี้คือ

- ออปเจกต์ ที่มีความซับซ้อนของแอทริบิ้ว และค่าในระดับอะตอม
- ออปเจกต์ ที่มีค่าในลักษณะแบบ หลายๆ ค่า (multi-value) ได้
- ออปเจกต์ ที่มีแอทริบิ้วแตกต่างกัน
- ออปเจกต์ ที่มีลักษณะลองฟิลด์แอทริบิ้ว (long field attribute) ตัวอย่างเช่น ข้อมูลในลักษณะ ภาพ เสียง เป็นต้น

การไอเดนติไฟล์ออปเจกต์ (Object Identifier, OID)

OID เป็นการระบุหรืออ้างอิงออปเจกต์เพื่อบอกถึงความสัมพันธ์ของออปเจกต์ซึ่งมักจะอยู่ในรูปของตัวเลขหรือตัวอักษรที่ไม่มีความซ้ำซ้อนกันเพื่อที่จะตั้งค่าให้กับออปเจกต์ซึ่งหลักการสร้าง OID ให้กับออปเจกต์มีผลอย่างมากกับประสิทธิภาพของ OODBMS

OID สามารถจะนำเสนอได้หลายรูปแบบ นั่นก็คือสามารถอธิบายได้ทั้งทางด้านฟีซิคอลและลอจิคอล โดยมีหลักการพื้นฐานดังนี้คือ

- ฟีซิคอลแอดเดรส (Physical address)

เป็นการให้ค่า OID โดยใช้เป็นฟีซิคอลแอดเดรสของออปเจกต์ซึ่งมักใช้กับภาษาที่ใช้ในการโปรแกรมมีข้อได้เปรียบที่มีประสิทธิภาพสูงมากในด้านความเร็วแต่ว่าหากมีการเคลื่อนย้ายทางด้านฟีซิคอลของการเปลี่ยนแปลงค่าของแอดเดรสหรือมีการลบค่าใด ๆ จะทำให้การแก้ไขทำได้ไม่สะดวกและทำให้การทำงานผิดพลาดได้

- แอดเดรสแบบโครงสร้าง (Structure address)

ค่าของ OID จะถูกแบ่งออกเป็น 2 ส่วนคือ ส่วนแรกจะเป็นเป็นส่วนที่เก็บหมายเลขเซ็กเม้นท์และหมายเลขเพจซึ่งจะมีประโยชน์ในด้านการดึงข้อมูลจากดิสก์ส่วนที่สองจะประกอบด้วยหมายเลขของลอจิคอลสลอท (Logical slot number) เพื่อใช้สำหรับอ้างอิงของออปเจกต์ในเพจซึ่งวิธีนี้สามารถเคลื่อนย้ายข้อมูลได้สะดวกกว่า

- เซอร์โรเกต (Surrogate)

วิธีนี้ค่าของ OID จะถูกสร้างขึ้นด้วยอัลกอริธึมที่จะรับประกันได้ว่าจะไม่มี

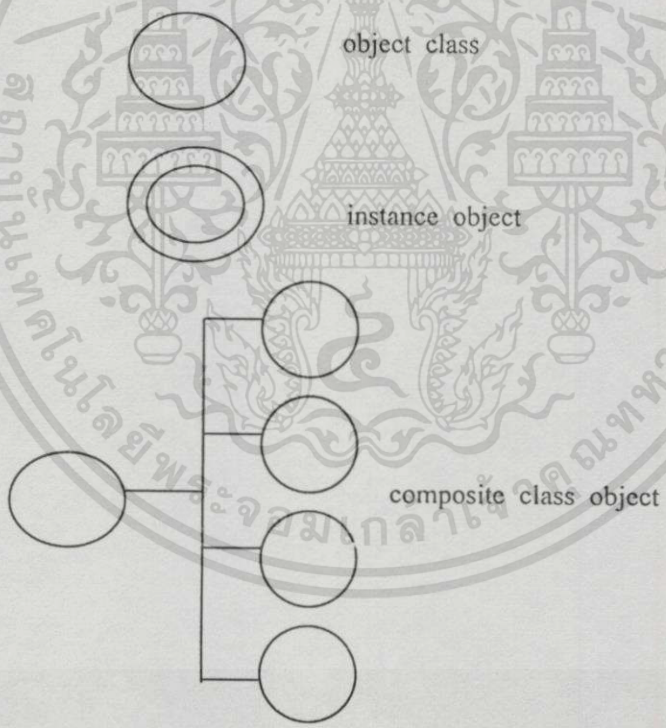
ความซ้ำซ้อนกันของ OID เช่นใช้ วัน เวลา หรือใช้การเพิ่มค่าหมายเลขขึ้นเรื่อย ๆ ค่าเซอร์โรเกต OID จะถูกแปลงกลับเป็นแอดเดรสทางฟิซิคอลและจะมีอินเด็กซ์ที่ใช้ในการค้นหา

- เซอร์โรเกตแบบอื่น ๆ (Typed surrogates)

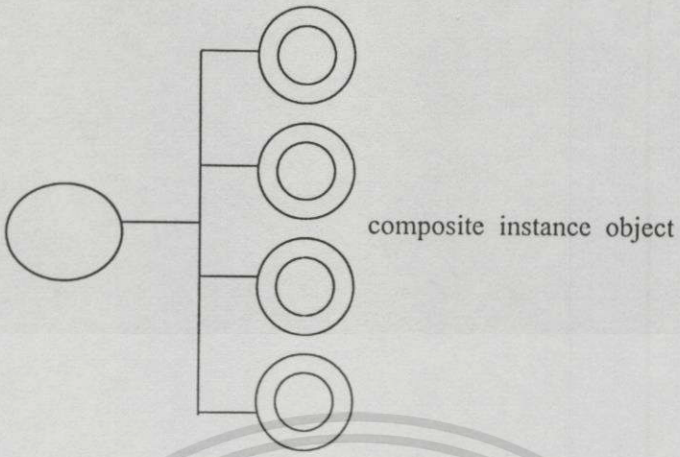
วิธีการเซอร์โรเกตจะสร้างให้ OID มีการเก็บค่าสองส่วนคือ ชนิดของ ID และออปเจกต์ไอเด็นติไฟล์จากนั้นจะมีตัวเพิ่มค่าให้กับแต่ละออปเจกต์ไปเรื่อย ๆ ซึ่งวิธีการนี้จะนิยมใช้มากทั้งในภาษาที่มีลักษณะเป็นออปเจกต์วิชวลทูลและฐานข้อมูลเชิงวัตถุใหม่ ๆ

2.2.3 ลักษณะไดอะแกรมของออปเจกต์ (Object Property diagram)

รูปแบบของออปเจกต์เราสามารถแสดงได้ดังในรูปที่ 2.5 และในรูปที่ 2.6 เป็นการแสดงลักษณะของภายในโครงสร้างของตัวออปเจกต์ว่าประกอบไปด้วยอะไรบ้าง และสามารถแสดงลักษณะตัวอย่างของคลาส A ในลักษณะสัญลักษณ์แบบกราฟฟิคได้ดังรูปที่ 2.7

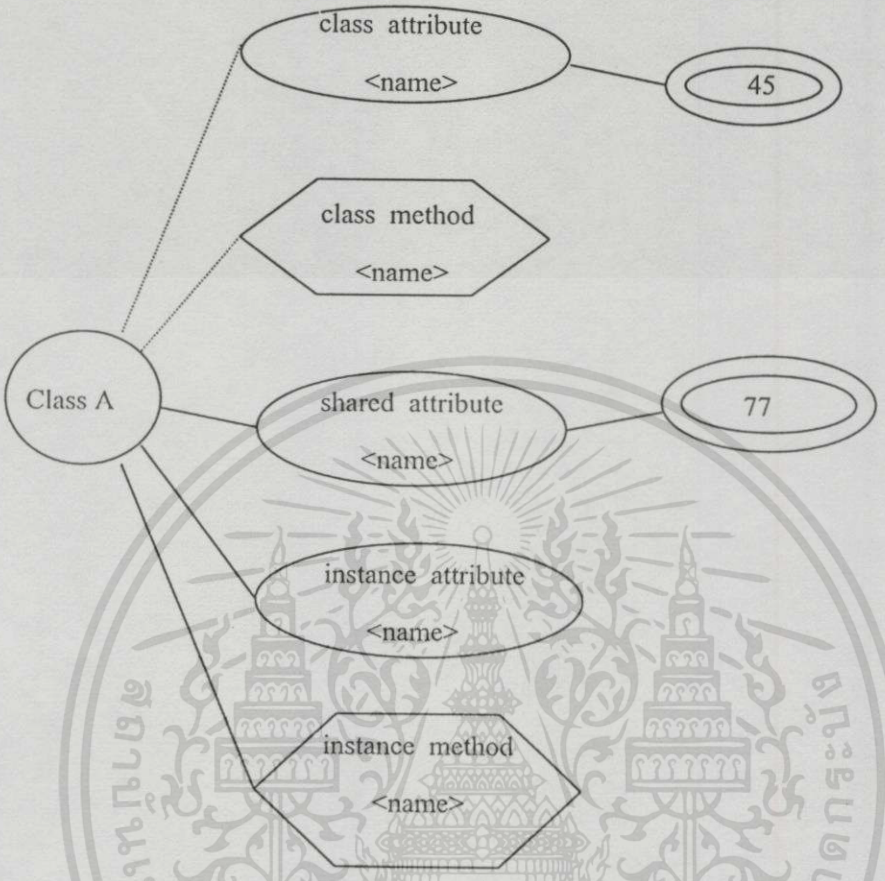


รูปที่ 2.5 แสดงสัญลักษณ์ของออปเจกต์



รูปที่ 2.5 (ต่อ)

รูปที่2.6 แสดงสัญลักษณ์ภายในโครงสร้างของออปเจกต์

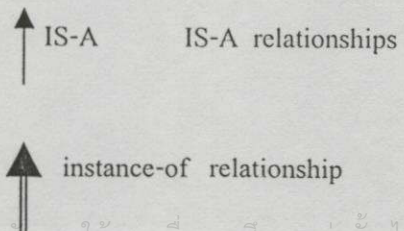


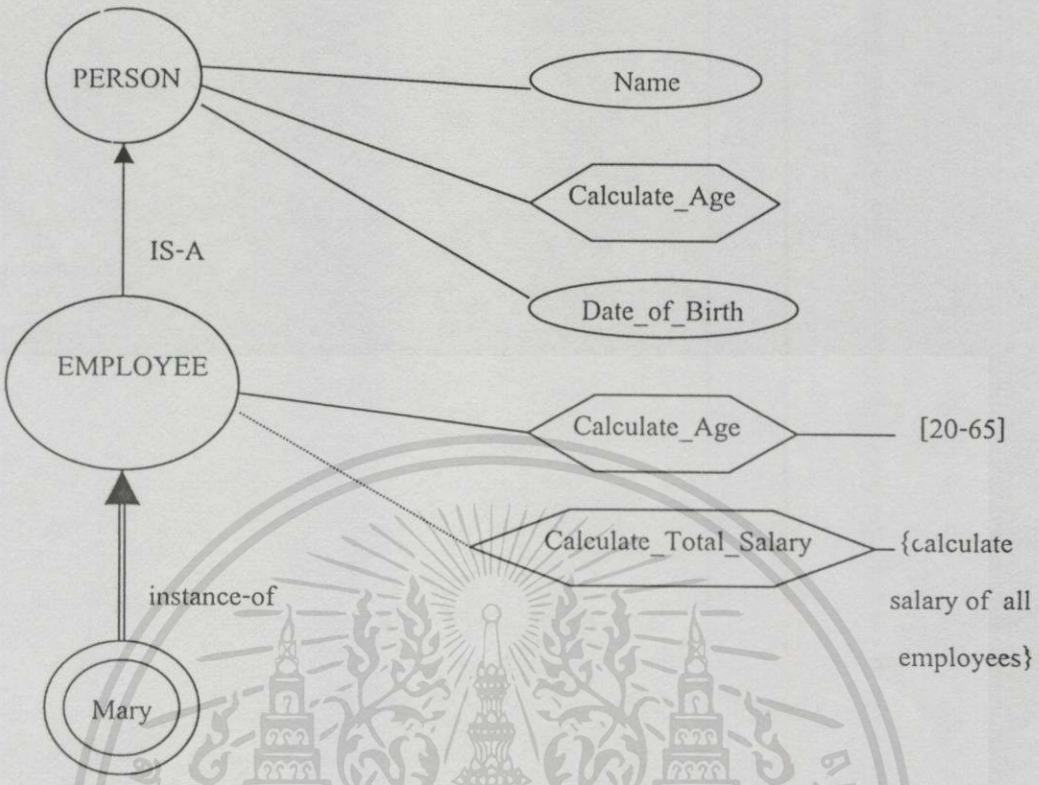
รูปที่2.7 แสดงตัวอย่างการใช้สัญลักษณ์ของคลาส A

2.2.4 การแสดงความสัมพันธ์ในลักษณะของไดอะแกรมออปเจกต์ (Expressing relationships and the Oobject-Property diagram)

ถ้าออปเจกต์ได้มีการนำเสนอแนวคิดและอินสแตนท์แล้วนั้น สามารถแสดงความสัมพันธ์ในลักษณะไฮราคี่ (Hicrarchical) ดังแสดงในรูปที่ 2.8 โดยสามารถทำได้โดยแสดงด้วย

- IS-A relationships
- Instance-of relationships





รูปที่2.8 แสดงตัวอย่างการใช้สัญลักษณ์

2.2.5 ข้อเปรียบเทียบการทำงานระหว่าง RDBMS กับ ODBMS

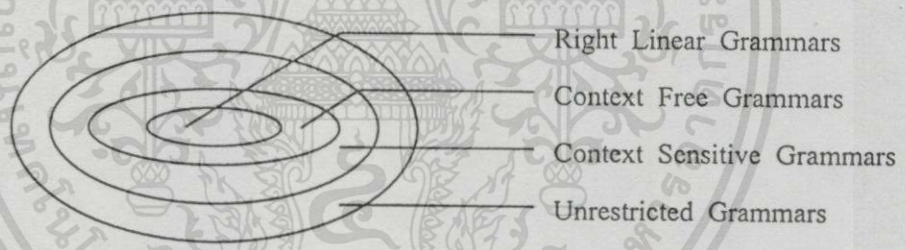
การทำงานของ RDBMS เป็นระบบจัดการฐานข้อมูลเชิงสัมพันธ์ที่เหมาะสำหรับเก็บข้อมูลที่ไม่ซับซ้อนเช่น ชื่อ สกุล ที่อยู่ ข้อดีของระบบนี้ก็คือได้รับการพัฒนามาเป็นเวลานานจนค่อนข้างเสถียรภาพมากแล้วและยังทำงานได้เร็วคุณสมบัติเหล่านี้ทำให้ RDBMS เหมาะกับงานทางด้านธุรกิจและงานด้านแอปพลิเคชั่น รูปแบบการทำงานของระบบนี้จะเป็นในลักษณะสองมิตินั่นก็คือจะเป็นแถวกับคอลัมน์ ซึ่งเป็นลักษณะตาราง ในขณะที่ระบบจัดการฐานข้อมูลเชิงวัตถุจะเก็บโมเดลที่มีความซับซ้อนโดยไม่ต้องพยายามบีบข้อมูลให้เหลือเพียงแค่สองมิติ ทำให้ODBMS เหมาะกับงานทางด้านมัลติมีเดียเป็นอย่างยิ่ง สำหรับการค้นหาใน RDBMS จะมีความซับซ้อนแต่ข้อมูลที่นำมาใช้ไม่ซับซ้อนแต่สำหรับใน ODBMS จะเน้นในจุดของการเก็บข้อมูลที่ซับซ้อนมาก ๆ แต่การค้นหาไม่ซับซ้อน สำหรับการทำงานของ ODBMS ได้รับความนิยมเริ่มตั้งแต่ปี 1980 ซึ่งในขณะนั้นออปเจกต์เป็นถือว่าเป็นคลื่นลูกใหม่ แต่ปัญหาที่พบในยุคแรกๆ ของ ODBMS ที่พบนั่นก็คือยังไม่เป็นระบบฐานข้อมูลที่สมบูรณ์ขาดความสามารถในการแบ็คอัพแอนด์รีโคร่ี คาต้าโมเดลก็ยังมีความสับสนอยู่ นอกจากนี้ภาษาก็ยังไม่มีมาตรฐาน แต่ในปัจจุบันนี้ ODBMS ได้มีการพัฒนาไปมากทำให้ระบบมีความเสถียรภาพมากยิ่งขึ้น

บทที่ 3
การพาร์สซิ่ง

3.1 รูปแบบและหลักไวยากรณ์ทางภาษาธรรมชาติ

การจำแนกรูปแบบและหลักไวยากรณ์และภาษาทางโปรแกรม เราสามารถที่จะทำการจัดจำแนกไวยากรณ์และภาษาทางโปรแกรมโดยอาศัยกฎข้อบังคับที่อยู่ในรูปของ productions ดังแสดงในรูปที่ 3.1 โดยสามารถทำการแบ่งออกเป็น 4 ประเภทได้ดังนี้คือ

1. Right Linear Grammars
2. Context Free Grammars
3. Context Sensitive Grammars
4. Unrestricted Grammars



รูปที่3.1 แสดงประเภทของไวยากรณ์

3.1.1 Right Linear Grammars

รูปแบบไวยากรณ์นี้จะเป็นส่วนหนึ่งของไวยากรณ์ในรูปแบบของ Context Free Grammars ลักษณะไวยากรณ์ในรูปแบบนี้จะมีการกำหนดให้สัญลักษณ์ที่เป็น nonterminal symbols อยู่ทางด้านขวามือของไวยากรณ์ โดยมีสัญลักษณ์ terminal symbols อยู่ทางด้านซ้ายมือเสมอซึ่งสามารถที่จะแสดงรูปแบบได้ดังนี้คือ

<A>::= t<B>

<A>::= t

กำหนดให้ A, B เป็น nonterminal symbols

t เป็น terminal symbols

ตัวอย่างรูปแบบที่ได้นำลักษณะไวยากรณ์ประเภทนี้มาใช้เช่นคำสั่งของประโยคในภาษาทางโปรแกรมดังนี้คือ

REPEAT...UNTIL

DO...WHILE

IF...THEN...ELSE

กรณีใช้รูปแบบของ WHILE...DO แสดงได้ดังนี้คือ

WHILE x=2

DO set y to y-x

3.1.2 Context Free Grammars

เป็นไวยากรณ์ที่สามารถใช้อธิบายโครงสร้างในภาษาธรรมชาติ และยังมีประสิทธิภาพในการวิเคราะห์ คำทางไวยากรณ์ นอกจากนี้ยังเป็นที่นิยมสำหรับนำมาใช้อธิบายภาษาทางโปรแกรมที่อยู่ในรูปแบบที่เรียกว่า BNF (Backus-Naur Form) รูปแบบของไวยากรณ์นี้สามารถที่จะแสดงได้ดังนี้คือ

<S>::= a<S> b

หรือ

S ⟶ a S b

ลักษณะของไวยากรณ์ประเภทนี้จะมีสัญลักษณ์ที่เป็น nonterminal symbols อยู่ทางด้านซ้ายของกฎเพียงตัวเดียว นอกจากนี้รูปแบบการวิเคราะห์ประโยค (parsing algorithm) สามารถทำได้ในลักษณะที่เรียกว่า บนลงล่าง (top-down parsing algorithm) และ ล่างขึ้นบน (bottom-up parsing algorithm) ตัวอย่างรูปแบบไวยากรณ์ และการสร้าง parse tree ดังในรูปที่ 3.2 โดยใช้รูปแบบไวยากรณ์ลักษณะดังนี้คือ

กำหนดให้เป็นไวยากรณ์ที่ 1

1. S ⟶ NP VP
2. NP ⟶ Art Noun
3. NP ⟶ Name
4. PP ⟶ Prep NP
5. VP ⟶ Verb
6. VP ⟶ Verb NP
7. VP ⟶ Verb NP PP
8. VP ⟶ Verb PP

หมายเหตุ: NP คือ Noun Phrase

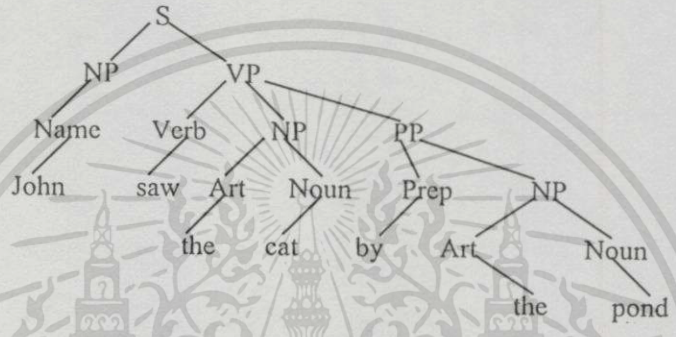
Art '' Article

PP '' Preposition Phrase

VP '' Verb Phrase

Prep '' Preposition

ตัวอย่างรูปประโยค John saw the cat by the pond



รูปที่3.2 แสดง parse tree โดยใช้ไวยากรณ์ที่ 1 ในรูปแบบ context free grammars

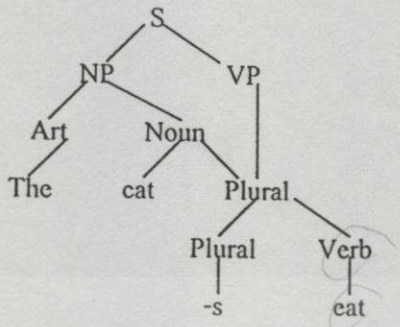
3.1.3 Context Sensitive Grammars

เป็นไวยากรณ์ที่สามารถนำมาใช้อธิบายถึงความสัมพันธ์ของรูปประโยคโดยความหมายของประโยคที่มีลักษณะไวยากรณ์แบบนี้สามารถที่จะอาศัยความสัมพันธ์ของเนื้อหาข้อความแต่ละส่วนที่รวมเป็นประโยคทำให้เกิดมีความหมายตามที่ต้องการ ดังในรูปที่ 3.3 ซึ่งตัวอย่างการนำรูปแบบไวยากรณ์ชนิดนี้มาใช้งานมีลักษณะที่สามารถแสดงได้ดังนี้คือ

กำหนดให้ไวยากรณ์ที่ 2

1. S → NP VP
2. NP → Art Noun Number
3. Number → Singular
4. Number → Plural
5. Singular VP → Verb -s
6. Plural VP → Plural Verb
7. Plural → -s

ตัวอย่างรูปประโยค The cats cat



รูปที่ 3.3 แสดง parse tree **โดยใช้ไวยากรณ์ที่เ ในรูปแบบ** context sensitive grammars

3.1.4 Unrestricted Grammars

เป็นไวยากรณ์ที่ไม่มีรูปแบบที่แน่นอน ซึ่งรูปแบบนี้จะถูกกำหนดตามความต้องการของผู้ใช้แต่ละบุคคล เป็นการยากที่จะนำเสนอได้ว่าไวยากรณ์นี้จะมีรูปแบบอย่างไร แต่ข้อบังคับของรูปแบบไวยากรณ์ชนิดนี้จะเน้นที่กฎการได้มาของไวยากรณ์ทั้งทางด้านซ้ายและขวาของกฎจะต้องประกอบไปด้วยข้อความที่เป็นทั้ง terminal symbols และ nonterminal symbols

3.2 การทำพาร์สซิ่ง (Parsing Technique)

คือการวิเคราะห์ประโยคโดยการตรวจสอบโครงสร้างของประโยคว่ามีความถูกต้องตามหลักไวยากรณ์หรือไม่ ซึ่งจะทำการตรวจสอบวิเคราะห์ไวยากรณ์ในลักษณะของต้นไม้กระจายคำ (Parse Tree) โดยให้ใบ (leaves) เป็นสัญลักษณ์ท้ายประกอบเป็นข้อความในโปรแกรม และมีรากเป็นสัญลักษณ์ nonterminal สำหรับการทำพาร์สซิ่งนี้สามารถกระทำได้ 2 วิธีดังนี้คือ

i) การทำพาร์สซิ่งแบบที่เรียกว่า บนลงล่าง (Top-down parsing)

ii) การทำพาร์สซิ่งแบบที่เรียกว่า ล่างขึ้นบน (Bottom-up parsing)

3.2.1 การทำพาร์สซิ่งแบบบนลงล่าง (Top-down parsing)

วิธีการแบบนี้เริ่มจากที่รากของต้นไม้ (Top-down parsing) สำหรับการวิเคราะห์แบบนี้น่าสนใจมาก เพราะอัลกอริธึมไม่ซับซ้อนและสามารถสร้างได้โดยตรงจาก production ของภาษา นอกจากนี้การตรวจวิเคราะห์ว่า วิธีการตรวจสอบหลักไวยากรณ์ที่ทำงานในลักษณะของบนลงล่างนี้ได้หรือไม่ ซึ่งนั่นก็คือจะต้องอาศัยที่เรียกว่า LL grammar เป็นตัวตรวจวิเคราะห์ซึ่งถ้าหากไวยากรณ์นั้นมีลักษณะที่เป็น Left Recursion จะไม่เป็น LL grammar ซึ่งทำให้ไวยากรณ์นี้ไม่เหมาะที่จะทำ Top-down parsing

ลักษณะไวยากรณ์ที่มีรูปแบบ Left Recursion เช่น

1. Expr ⟶ Expr + Term
2. Expr ⟶ Term
3. Term ⟶ Term * Factor
4. Term ⟶ Factor
5. Factor ⟶ (Expr)
6. Factor ⟶ var

จากหลักไวยากรณ์อธิบายได้ว่าทั้ง Expr และ Term เป็น Left Recursion นอกจากนี้ วิธีการแก้ไม่ให้เกิด Left Recursion สามารถทำได้โดยการเพิ่มสัญลักษณ์ nonterminal symbols ตัวใหม่ลงไปในไวยากรณ์ จะได้ไวยากรณ์ที่สามารถทำ Top-down parsing ได้ใหม่คือ

1. Expr ⟶ Expr Elist
2. Elist ⟶ +Term Elist
3. Elist ⟶ ε
4. Term ⟶ Factor Tlist
5. Tlist ⟶ * Factor Tlist
6. Tlist ⟶ ε
7. Factor ⟶ (Expr)
8. Factor ⟶ var

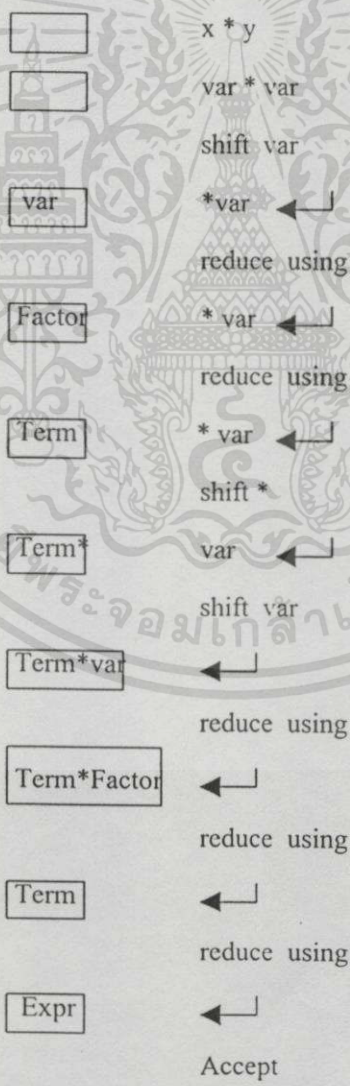
3.2.2 การทำพาร์ซิ่งแบบล่างขึ้นบน (Bottom-up parsing)

เป็นวิธีการกระจายคำจากใบไปสู่รากของต้นไม้กระจายคำ การทำงานแบบ Bottom-up parsing นี้จะประกอบด้วยการทำงาน 2 ส่วนคือ ส่วนที่ทำการเก็บสัญลักษณ์ของข้อมูลนำเข้าเรียกว่า Shift และส่วนที่ทำหน้าที่แทนสัญลักษณ์ข้อมูลเข้าด้วยสัญลักษณ์ nonterminal เรียกว่า reduce ดังนั้นการทำงานของ Bottom up parser นิยมเรียกว่า Shift reduce parser นอกจากนี้การทำงานของไวยากรณ์ในลักษระที่เรียกว่า Shift reduce นี้จะถือว่าเป็น LR grammar (L เป็นตัวชี้ถึงการอ่านข้อมูลนำเข้าจากด้านซ้ายมือ , R เป็นตัวชี้ถึงการค้นหากฎในด้านขวามือ) แต่หากเกิดการ ติดขัดจากการ Shift reduce แล้วแสดงว่าไวยากรณ์นั้นมีลักษณะกำกวมของไวยากรณ์ ซึ่งจะถือว่าไม่เป็น LR grammars

ตัวอย่างการทำงานแบบล่างขึ้นบน (Bottom-up parsing) โดยใช้หลักการ LR grammar ตรวจวิเคราะห์ ดังแสดงในรูปที่ 3.4

กำหนดให้เป็นไวยากรณ์ 3

1. Expr → Expr + Term
2. Expr → Term
3. Term → Term * Factor
4. Term → Factor
5. Factor → (Factor)
6. Factor → var



รูปที่ 3.4 แสดง Bottom-up parsing สำหรับ x * y ในลักษณะขั้นตอนของ LR grammar

บทที่ 4
กฎข้อบังคับความถูกต้อง

4.1 การสร้างกฎข้อบังคับความถูกต้องของข้อมูล

conceptual schema design procedure (CSDP) เป็นกระบวนการที่สร้างขึ้นมาเพื่อแก้ปัญหาในเรื่องลดค่าใช้จ่ายของการออกแบบระบบสารสนเทศที่มีขนาดใหญ่ ๆ ซึ่งแนวความคิดเริ่มต้นนี้ เกิดจากความคิดของศาสตราจารย์ G. M. Nijssen จากมหาวิทยาลัยควีนแลนด์ (University of Queenland) และ E. D. Falkenberg (Katholieke Universiteit, Nijmegen) โดย Falkenberg ได้ทำการออกแบบ conceptual schema โดยอาศัยพื้นฐานของประโยคภาษาธรรมชาติ

หลาย ๆ ปี ถัดมา NIAM เป็นงานวิจัยที่เกิดขึ้นของ Nijssen ร่วมกับ Falkenberg และได้ถูกทำเป็นหนังสือเพื่อเผยแพร่โดย T. A. Halpin (University of Queensland) เป็นการวิเคราะห์เกี่ยวกับข้อเท็จจริง (Fact Type) ซึ่ง NIAM จะถูกเรียกว่า fact-oriented modeling สำหรับ NIAM จะมีลักษณะคล้าย ๆ กับ Entity-Relationship model ที่นำเสนอโดย Chen (1976) แต่ในแอมนี้จะมีความง่าย ๆ กว่า และสื่อความหมายของโมเดลในลักษณะธรรมชาติมากกว่า

ขั้นตอนการทำ CSDP มีอยู่ 9 ขั้นตอนซึ่งกระบวนการเริ่มต้นด้วยการกำหนด fact type และขั้นตอนต่อ ๆ ไปจะเป็นการทำการตรวจสอบข้อผิดพลาด สำหรับขั้นตอนการทำ CSDP มีดังนี้คือ

1.การเปลี่ยนแปลงรูปของข้อมูลให้อยู่ในรูปของข้อเท็จจริงพื้นฐาน และกำหนดคุณภาพในการตรวจสอบ

2. วาดรูปของ conceptual schema diagram

3. ขจัด entity type ส่วนเกินที่จะได้ข้อเท็จจริง

4. เพิ่ม unique constraint ในแต่ละ fact type

5. ตรวจสอบข้อเท็จจริงของ right arity

6. เพิ่ม entity type , mandatory role, subtype และ occurrence frequency constraints

7. ตรวจสอบในแต่ละ entity ที่ได้ระบุไว้

8. เพิ่ม equality , exclusion, subset และ constraints อื่น ๆ

9. ตรวจสอบความสมบูรณ์ของ conceptual schema ไม่ให้มีความซ้ำซ้อน

4.1.1 ขั้นตอนการออกแบบ CSDP

ขั้นตอนที่ 1. เปลี่ยนแปลงรูปของข้อมูลให้อยู่ในรูปของข้อเท็จจริงพื้นฐาน

ถ้าเราทำการออกแบบ conceptual schema ในลักษณะของแอปพลิเคชั่นแล้วนั้น ซึ่งแต่ก่อนหน้านี้นั้นจะทำในลักษณะของ manual หรือใช้คอมพิวเตอร์ตัวของข้อมูลได้แก่จำพวกรายงานหรือไม่ก็เป็นเอกสารรายงานต่าง ๆ จากตัวอย่างทั้งสองนี้ถือว่ามีความสำคัญ

ตารางที่ 4.1 แสดงตัวอย่างรายงานของ Tutorial group

Tutorial group	Time	Room	Student#	Student name
A	Mon 3 p.m.	CS-718	302156	Bloggs FB
			180064	Fletcher JB
			278155	Jackson M
			334067	Jones EP
			200140	Kawamoto T
..	..	..	..	..
B1	Tue 2 p.m.	E-B18	266010	Anderson AB
..	..	..	..	..

จะเห็นได้ว่าเราพยายามที่จะอ่านข้อมูลที่แถวบนสุดก่อนซึ่งอยู่ในรูปของ elementary facts จากข้อมูลในตารางที่ 4.1 สามารถนำเสนออยู่ในรูปประโยค facts ได้ดังนี้

Tue group A meets at Time Mon 3 p.m.

Tue group A is held in Room CS-718.

Student 302156 belongs to Tue group A.

Student 302156 has Name 'Bloggs FB'.

ตัวอย่างรูปประโยคในลักษณะ elementary facts แบ่งออกเป็น 2 ลักษณะดังนี้คือ

1. one object เป็นลักษณะของ elementary facts ที่ง่าย ๆ มีเพียงออปเจกต์เดียวและ role เพียงเท่านี้

ตัวอย่างเช่น Ann smokes

2. two objects เป็นลักษณะของ elementary facts ที่เกิดจาก 2 ออปเจกต์มีความสัมพันธ์กันโดยมี role คั่นระหว่างออปเจกต์ทั้งสอง

ตัวอย่างเช่น Ann employs Bob

สำหรับรูปประโยคที่ไม่เป็น elementary facts มีลักษณะดังตัวอย่างต่อไปนี้คือ

Ann smokes **and** Bob smokes

If Bob smokes **then** Bob is cancer prone

All people who smoke are cancer prone

ขั้นตอนที่ 2. วาดรูปของ conceptual schema diagram

ในขั้นตอนนี้จะแสดงการวาดไดอะแกรมของ fact type ซึ่งมีทั้ง entity type, label type และ role โดยใช้ข้อมูลจากตารางที่ 4.2

ตารางที่ 4.2 แสดงข้อเท็จจริงทั่ว ๆ ไปเกี่ยวกับการขับรถยนต์

Drives:	Person	Car
	Adams B	235PZN
	Jones B	235PZN
	Jones E	108AAQ

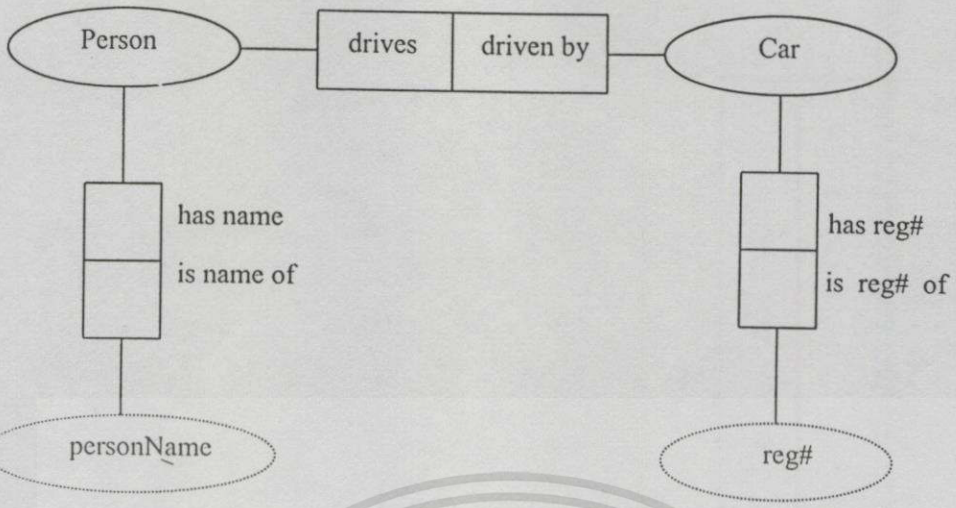
จากรายละเอียดข้อมูลจากตารางสามารถแสดงอยู่ในรูปของ elementary facts ได้โดยให้ "reg#" เป็นชื่อย่อของ "registration number"

The Person with name 'Adams B' drives the Car with reg# '235PZN'.

The Person with name 'Jones E' drives the Car with reg# '235PZN'.

The Person with name 'Jones E' drives the Car with reg# '108AAQ'.

ก่อนที่จะดูไดอะแกรมของ conceptual schema จะเห็นว่ามี entity type ซึ่งบรรยายด้วยรูปวงรีหรืออาจจะเป็นรูปวงกลมก็ได้โดยชื่อของ entity type ถูกเขียนไว้ด้านในของรูปวงรีหรืออาจจะเขียนไว้ข้าง ๆ ก็ได้ สำหรับ label type จะเป็นรุปวงรีเส้นประ โดยชื่อของ label type อาจจะเป็นชื่อเดียวกันกับ reference mode (เช่น "reg#") หรืออาจจะแตกต่าง (เช่น "personName") แสดงในรูปที่ 4.1 ทั้งเอนติตี้และเลเบิ้ลเป็นตัวอย่างของออปเจกต์โดยที่ในไดอะแกรมของ conceptual schema มีโรลแสดงอยู่ในรูปลักษณะกลุ่มโดยชื่อของโรลจะถูกแทนที่ภายในหรือภายนอกกลุ่มก็ได้



รูปที่4.1 แสดงไดอะแกรม conceptual schema

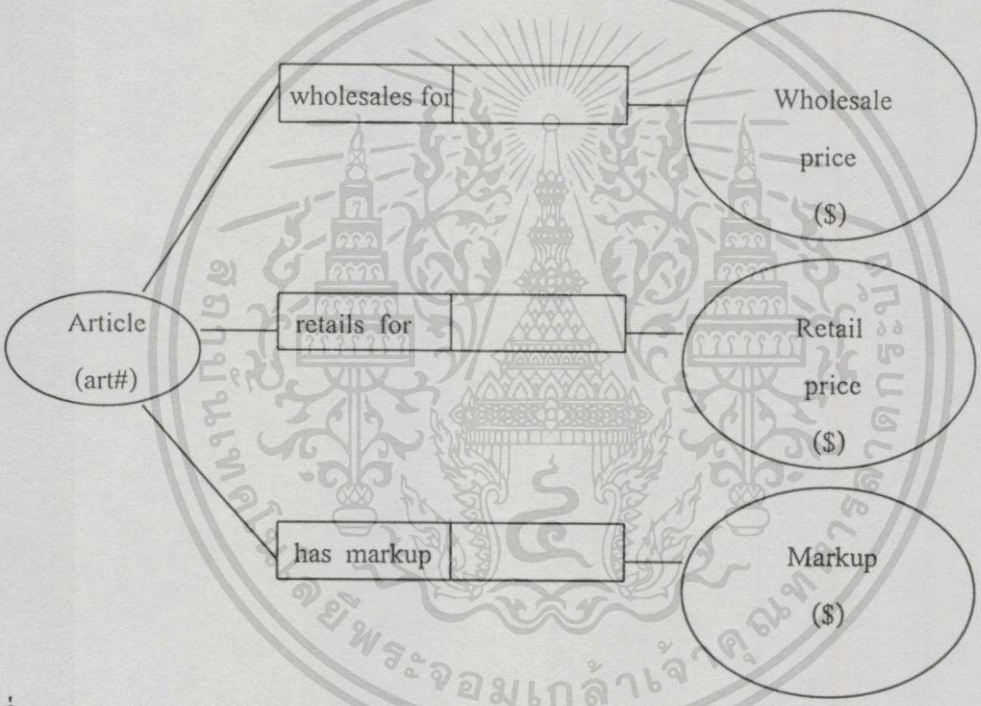
ขั้นตอนที่ 3. ขจัด entity types ส่วนเกินที่จะได้ข้อเท็จจริง

ในขั้นตอนนี้อันดับแรกให้มองดูรอบ ๆ โดยถ้าเราจะแนะนำถึงเอนติตี้ที่ไม่จำเป็นในไดอะแกรม ในวิธีพื้นฐานที่เป็นส่วนของ UoD ในแต่ละเอนติตี้จะถูกรวมเป็นกลุ่มของเอนติตี้เดียว ดังนั้นการเลือก entity types เราจะต้องมั่นใจได้ว่าเป็น mutually exclusive มีข้อสังสัยที่มีเหตุผลว่า entity type สองเอนติตี้นั้นจะรวมกันได้ถ้าเอนติตี้ทั้งสองมีพื้นฐานของ reference mode เดียวกัน ดังตัวอย่างในตาราง 4.3 เราจะพิจารณาไดอะแกรมที่ได้จากรูปที่ 4.2 จะเห็นได้ว่า Wholesale price, Retail price และ Markup ทั้งหมดนี้มีพื้นฐานของ reference mode เดียวกันคือเป็น ($) และนอกจากนี้ให้สังเกตดูจะเห็นได้ว่าข้อมูลจากตัวอย่างในตารางที่ $50 มีปรากฏอยู่ทั้ง Wholesale price และ Markup ในกรณีนี้จะมี entity type ของทั้งสองมีข้อมูลเดียวกันของเอนติตี้เราก็จะทำการรวมกันของ entity type ถ้าข้อมูลในตาราง retail price ไม่มีความสอดคล้องกับ markup แล้วนั้นก็จะเป็นเพียงการเปรียบเทียบกันระหว่างเอนติตี้ทั้งสอง กรณีตัวอย่างของ A1 มี retail price ซึ่งปรากฏอยู่สามครั้งจะพิจารณาได้ว่ามีความสอดคล้องกับ markup ดังนั้นทั้งสามเอนติตี้สามารถจะทำการรวมเป็นเอนติตี้เดียวกันได้ดังแสดงในรูปที่ 4.3 ซึ่งเอนติตี้ทั้งสามสามารถแสดงอยู่ในรูปสมการคณิตศาสตร์ได้ดังนี้คือ

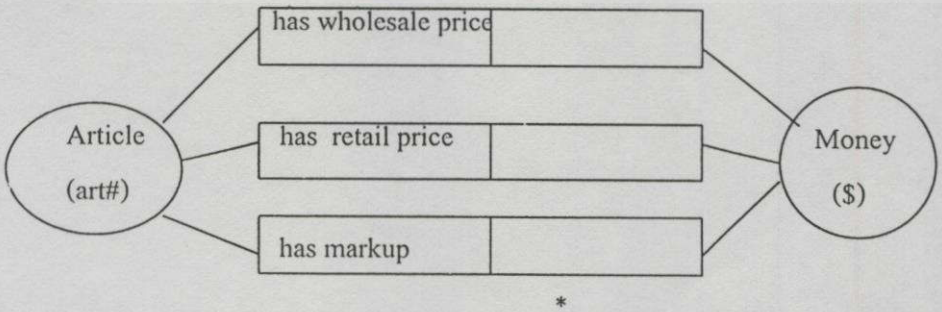
Markup = Retail price – Wholesale price

ตารางที่ 4.3 แสดงข้อเท็จจริงของข้อมูล Wholesale, Retail และ Markup

Article	Wholesale price($)	Retail price($)	Markup($)
A1	50	75	25
A2	80	130	50
A3	50	70	20
A4	100	130	30



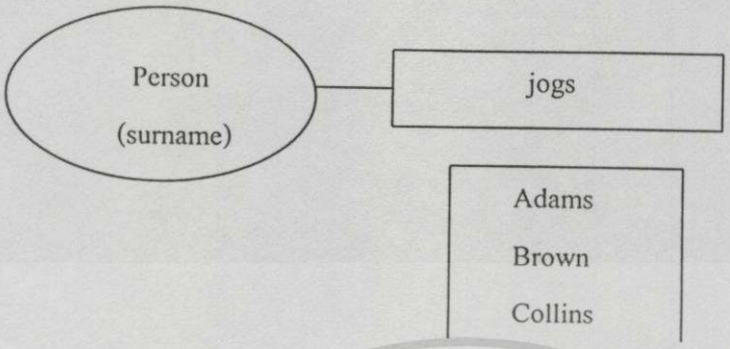
รูปที่4.2 แสดงข้อบกพร่องใน conceptual schema



รูปที่4.3 แสดงผลของการรวบรวมเป็นเอนติตี้เดียวกัน

ขั้นตอนที่ 4. เพิ่ม uniqueness constraints ในแต่ละ fact type

กรณี unary fact type



รูปที่4.4 แสดงไดอะแกรมของ schema-base ในลักษณะ unary fact type

จากรูปที่ 4.4 จะเห็นได้ว่าข้อมูลที่มีอยู่สามารถเขียนให้อยู่ในรูปของ fact type ได้ดังนี้คือ

The Person with surname 'Adams' jogs.

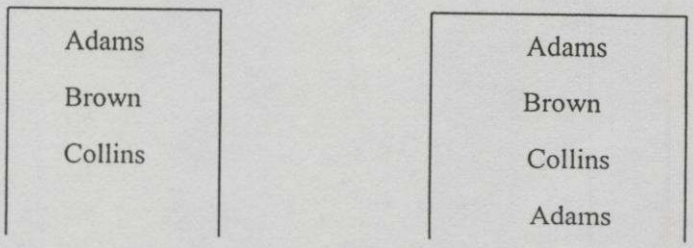
The Person with surname 'Brown' jogs.

The Person with surname 'Collins' jogs.

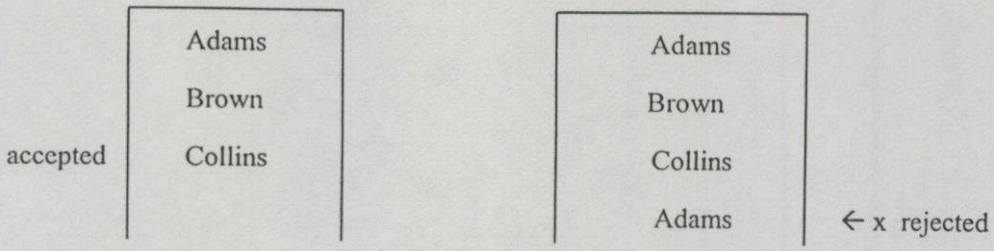
จากข้อเท็จจริงของเรคคอร์ดในฐานข้อมูลนี้จะเกิดอะไรขึ้นถ้ามีการเปลี่ยนแปลงข้อมูลขึ้นมา เช่น หากทำการเพิ่มจำนวนเรคคอร์ด

add: The Person with surname ''Adams' jogs.

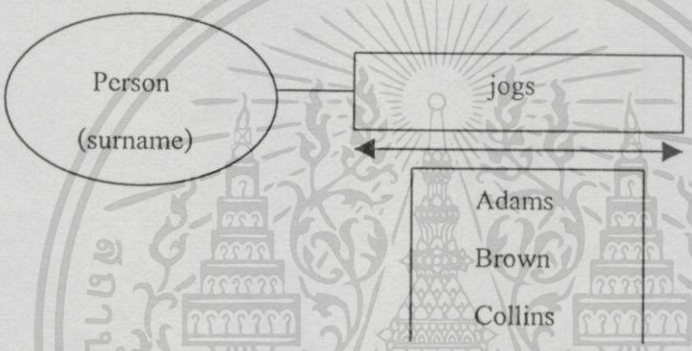
จะเห็นว่า Adams jogs เป็นข้อมูลที่มีอยู่แล้วในฐานข้อมูล และขณะนี้เราได้ทำการเพิ่มเข้าไปอีกแสดงดังในรูปที่ 4.5 ทำให้เกิดความซ้ำซ้อนใน elementary fact ที่เกิดขึ้นในแต่ละแถวซ้ำ ๆ กันเราจะไม่ยอมรับข้อมูลของแถวที่ซ้ำซ้อนกันดังรูปที่ 4.6 ดังนั้นเราจะแสดง unique constraint โดยแสดงในลักษณะเครื่องหมายลูกศรดังรูปที่ 4.7 ซึ่งเป็นการแสดงถึงข้อมูลในแต่ละคอลัมน์จะมีเพียงข้อมูลชื่อเดียวที่ไม่ซ้ำกัน



รูปที่4.5 แสดงการเพิ่มข้อมูลที่มีอยู่แล้วในฐานข้อมูล



รูปที่4.6 แสดงการไม่อนุญาตให้เกิดความซ้ำซ้อนของแถว



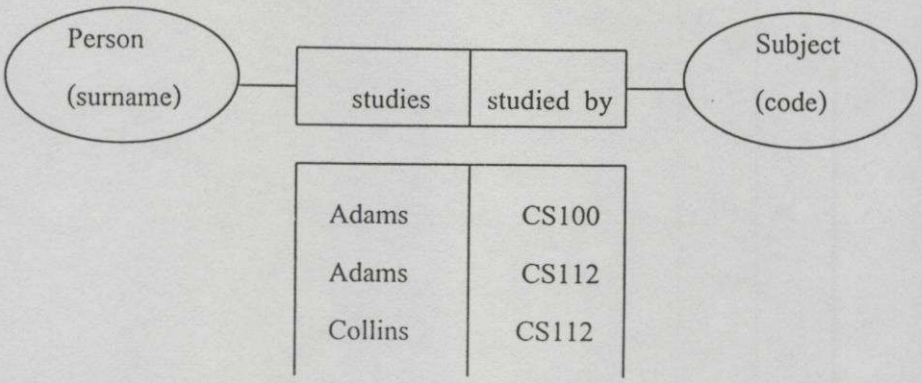
รูปที่4.7 แสดงการเป็น uniqueness constraint

4.1.2 ข้อเท็จจริงแบบไบนารี่ (binary fact type)

- ความสัมพันธ์แบบ many to many
- ความสัมพันธ์แบบ many to one
- ความสัมพันธ์แบบ one to one

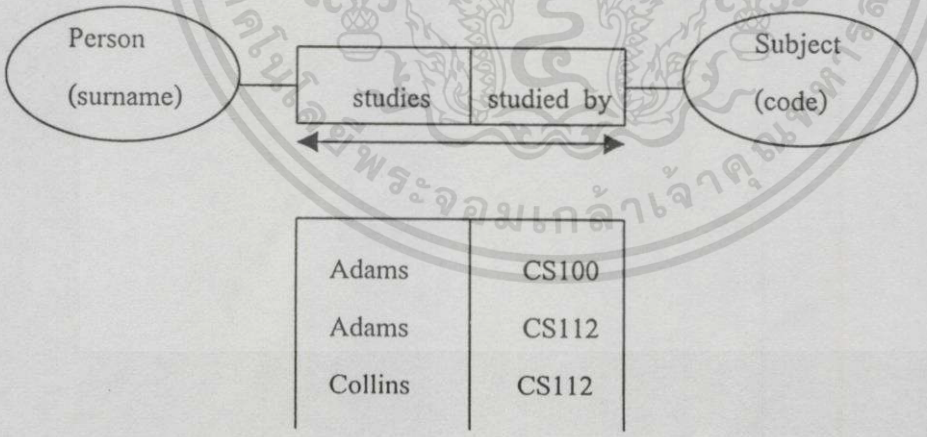
ความสัมพันธ์แบบ many to many

เริ่มจากไดอะแกรมของสกีมม่าในรูป 4.8 จะเห็นได้ได้ว่าในแต่ละแถวไม่มีการซ้ำกันในตารางฐานข้อมูล แต่เราก้ไม่สามารถบอกได้ว่าเป็น student เดียวกันจากตัวอย่าง student ที่ชื่อ Adams ปรากฏอยู่ถึงสองครั้งด้วยกันในคอลัมน์แรก และ CS112 ก็ปรากฏถึงสองครั้งเช่นกันในคอลัมน์ที่สอง จากตัวอย่างนี้แสดงถึงความสัมพันธ์ในลักษณะแบบ many to many (หรือ m : ๆ) ดังรูปที่ 4.9 ในกรณีนี้เราจะทำการรวม student และ subject เป็น unique เดียวกัน โดยแสดงลูกศรถึงทั้งสองคอลัมน์ของตารางความจริงดังรูปที่ 4.10



รูปที่4.8 แสดงไดอะแกรมของ schema-base ในลักษณะไบนารี fact type

รูปที่4.9 แสดงความสัมพันธ์แบบ many to many

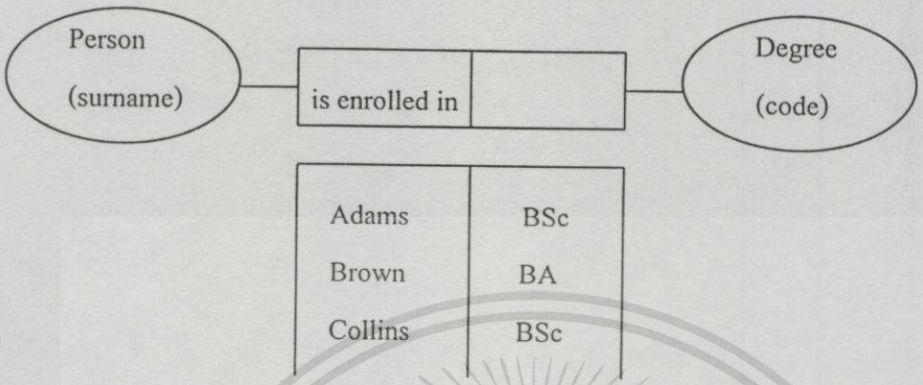


รูปที่4.10 แสดง uniqueness constraint

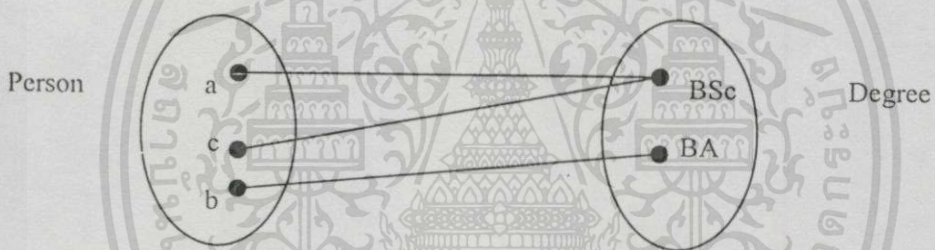
ความสัมพันธ์แบบ many to one

ให้พิจารณารูปที่ 4.11 จะเห็นได้ว่าอาจจะมี Person หลาย ๆคน ที่จะมีการลงทะเบียนที่มี degree เดียวกัน แต่ไม่มี Person คนใด ๆ มีการลงทะเบียนได้ในหลาย ๆ degree การบรรยายในรูปที่ 4.12 แสดงถึงการมีหลาย ๆ Person ที่จะมีการลงทะเบียน degree เดียวกัน ดังนั้นกล่าวได้

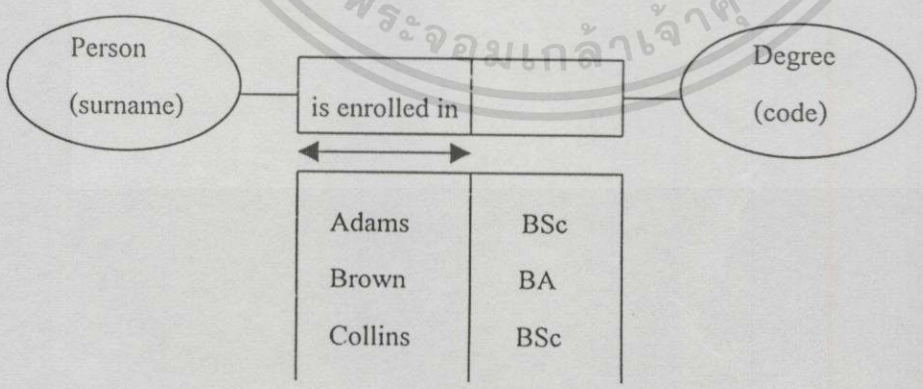
ว่า "...is enrolled in..." เป็น many to one (หรือ n : 1) สำหรับรูปที่ 4.13 แสดงการใช้ลูกศรชี้ในคอลัมน์ Person เพื่อไม่ให้ surname มีชื่อซ้ำซ้อนกัน



รูปที่4.11 แสดงอาจจะมีหลาย ๆ person ที่สามารถจะลงทะเบียนได้ degree เดียวกัน



รูปที่4.12 แสดงความสัมพันธ์แบบ many to one

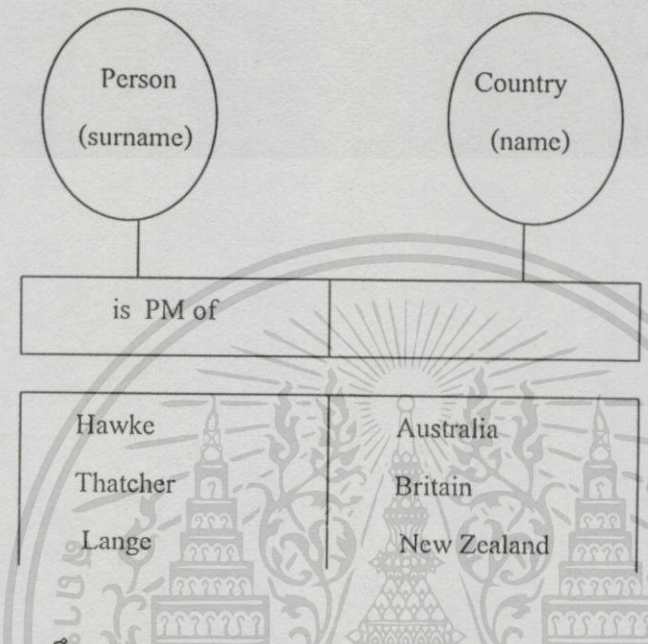


รูปที่4.13 แสดงการทำคอลัมน์ให้เป็น unique

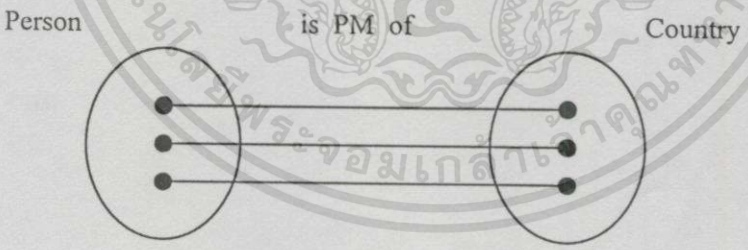
ความสัมพันธ์แบบ one to one

ให้พิจารณารูปที่ 4.14 เราใช้ "PM" เป็นชื่อย่อของ "Prime Minister" จะเห็นได้ว่า Person สามารถเป็น Prime Minister ได้เพียงประเทศเดียว และในแต่ละประเทศสามารถมีเพียง

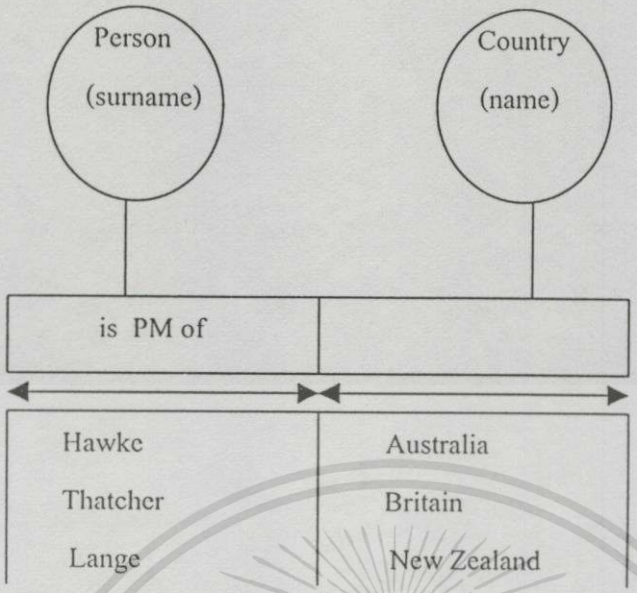
Prime Minister เดียวเช่นกัน ดังนั้นรูปแบบของ fact type นี้จะเป็นความสัมพันธ์แบบ one to one (หรือ 1 : 1) แสดงดังรูปที่ 4.15 และรูปที่ 4.16 แสดงลูกศรที่กำหนดให้ในแต่ละคอลัมน์ไม่ซ้ำซ้อนกัน



รูปที่4.14 แสดงถึง uniqueness constraints แบบใด?



รูปที่4.15 แสดงความสัมพันธ์แบบ one to one

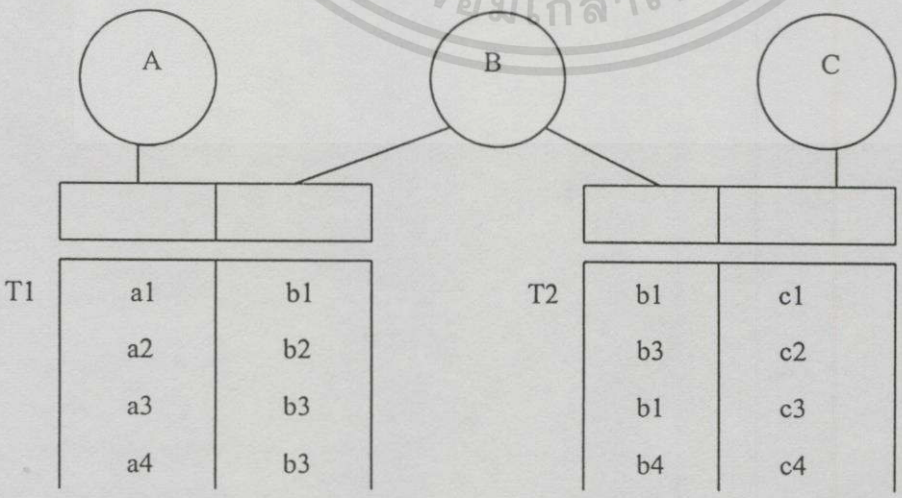


รูปที่4.16 แสดงในแต่ละคอลัมน์จะเป็น unique

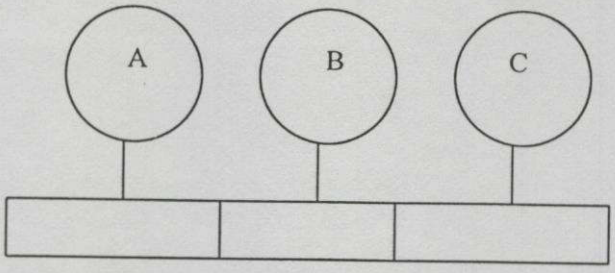
4.1.3 กฎข้อบังคับในลักษณะเชื่อมต่อกันหรือในลักษณะเป็นโครงร่างตาข่าย (Constraints involving joins or nesting)

Inter fact type

โดยธรรมชาติของการเชื่อมต่อกันของตาราง T1 และ T2 โดยแถวของ T1 กับแถวของ T2 โดยที่ในคอลัมน์ B เป็นค่าของการเชื่อมต่อกัน และสุดท้ายแล้วในคอลัมน์ B จะปรากฏอีกครั้งในผลลัพธ์ที่ได้จากตัวอย่างในรูปที่ 4.17 จะเห็นว่า row(a1,b1) คู่ของ (b1,c1) และ (b1,c3) จะรวมเป็น rows(a1,b1,c1) และ rows(a1,b1,c3) ดังในรูปที่ 4.18



รูปที่4.17 แสดงตาราง T1 และ T2 จะใช้ entity type ร่วมกัน



T3			
	a1	b1	c1
	a1	b1	c3
	a3	b3	c2
	a4	b3	c2

รูปที่4.18 แสดงตาราง T3 เป็นตารางรวมของตาราง T1 และ T2

เราจะพิจารณา constraint ที่มีความสัมพันธ์มากกว่าหนึ่งความสัมพันธ์จากตารางที่ 4.4 จะเขียนอยู่ในรูปของ elementary fact จากแถวที่หนึ่งได้ดังนี้คือ

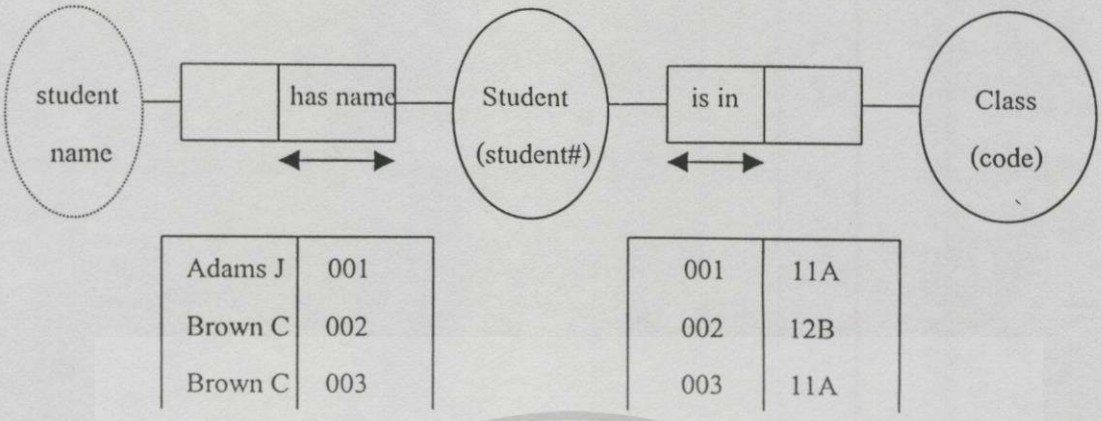
The Student with student# '001' has name 'Adams J'.

The Student with student# '001' is in the Class with code '11A'.

จากข้อเท็จจริงแรกเป็นความสัมพันธ์ระหว่าง entity และ label โดย entity ถูกนำเสนอโดยปกติ แต่ label "Adams J" เป็นการนำเสนอตัวเองสำหรับข้อเท็จจริงที่สองเป็นรูปแบบปกติ student ถูกกำหนดโดยจำนวนของ student ในแถวที่สอง และสาม เป็นไปได้ที่เกิดความแตกต่างของ student ที่มีชื่อเดียวกัน ("Brown C") ดังรูปที่ 4.19

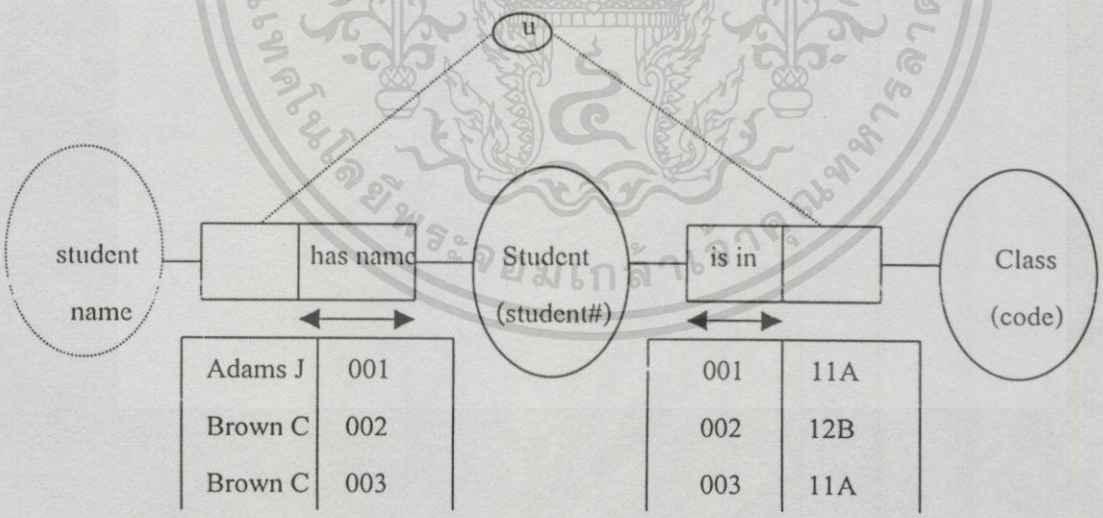
ตารางที่ 4.4 แสดงรายงานเกี่ยวกับนักเรียนในไฮสคูล

Student :	student#	name	class
	001	Adams J	11A
	002	Brown C	12B
	003	Brown C	11A



รูปที่4.19 แสดงไดอะแกรม conceptual schema โดยใช้ข้อมูลจากตาราง 4.4

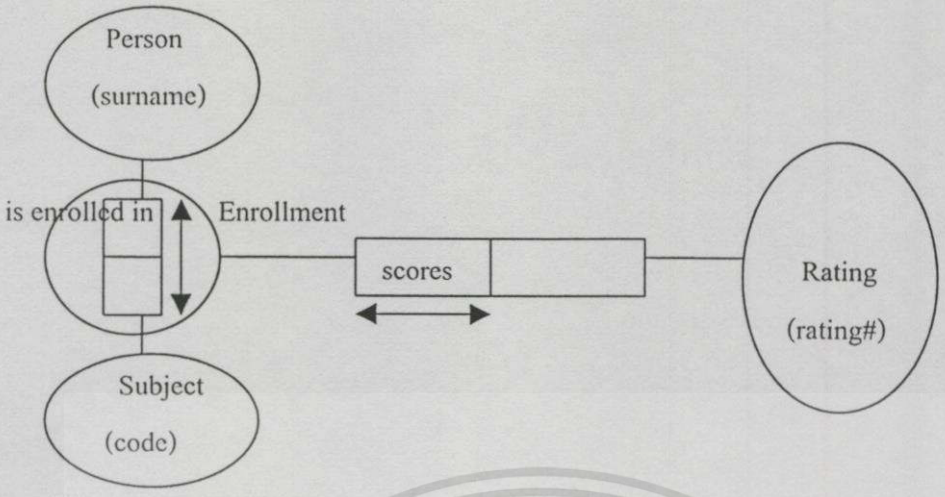
ดังนั้นในแต่ละแถวของเอาท์พุทสามารถทำการแยกออกเป็นสอง elementary fact ได้โดยการระบุชื่อของคลาส unique constraint ในไดอะแกรมของสกีมม่าซึ่งเราจำเป็นต้องรวมทั้งสอง fact type เข้าด้วยกันโดยการเชื่อม role เข้าด้วยกันโดยใช้เส้นประที่มีอักษรตัว u โดย u นี้แสดงถึงการเป็น unique กันดังรูปที่ 4.20 ซึ่งเราจะเรียกว่าเป็นคอนสเตรนท์ระหว่า fact type นี้ว่า " inter-fact-type constraint"



รูปที่4.20 แสดงการใช้สัญลักษณ์ u สำหรับ inter-uniqueness constraint

Nested fact types

ในกรณีที่การทำ unique constraint จำเป็นที่จะต้องทำสัญลักษณ์ลงบนความสัมพันธ์ซึ่งเป็นความสัมพันธ์ภายนอก จากตัวอย่างเกี่ยวกับ people ที่ทำการลงทะเบียนวิชาเรียน ซึ่งมีอันดับของคะแนนด้วย ซึ่งรูปแบบข่ายใยแมงมุมที่เรียกว่า nest fact type จะแสดงดังรูปที่ 4.21



รูปที่4.21 แสดง uniqueness constraints ในลักษณะ nested fact type

ขั้นตอนที่ 5 : ตรวจสอบข้อเท็จจริงของ right arity

ในขั้นตอนนี้ไม่จำเป็นต้องทำก็ได้ถ้าหากขั้นตอนที่ได้ทำที่ผ่าน ๆ มา ถ้ามีความรอบคอบในการวิเคราะห์ในการออกแบบได้ดี สำหรับการทำการตรวจสอบ arity ของ fact type ในขั้นตอนนี้มี 3 วิธีคือ

1. การแยกแยะข้อมูลจะมีรายละเอียดของข้อมูลบางส่วนที่จะต้องสูญหายไป ซึ่งการทำในกรณีนี้จำเป็นต้องอาศัยความรู้พื้นหลังที่มีอยู่พอสมควร

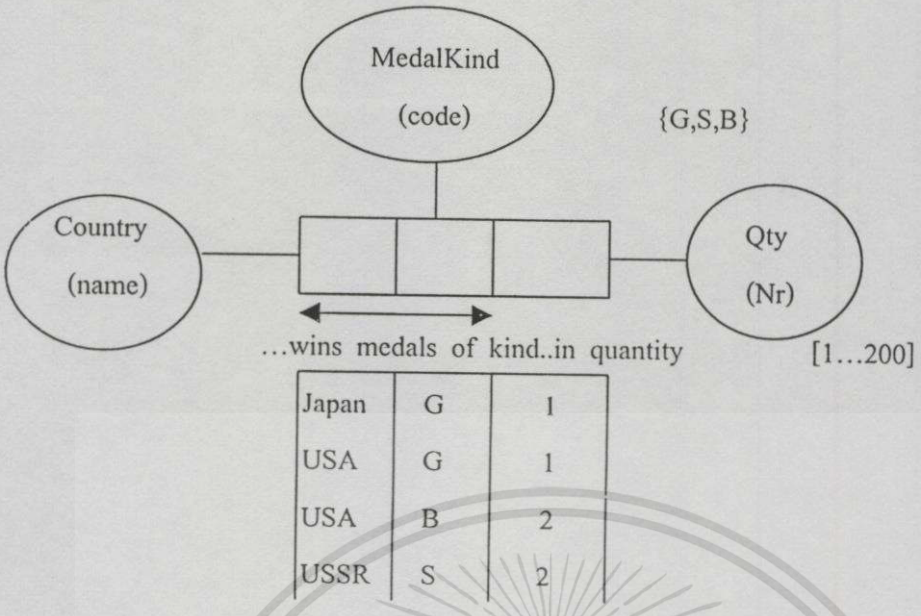
2. กฎการแยกแยะข้อมูลจะใช้การทำงานในลักษณะ shortest key

3. กำหนดสัญลักษณ์ของตารางข้อเท็จจริงให้กับ fact type ซึ่งการทำการแยกแยะจะทำโดยโปรเจกชั่น (projection) แล้วทำการรวมใหม่อีกครั้ง ถ้าข้อมูลใหม่นี้ปรากฏขึ้นแล้วไม่ต้องทำการแยกแยะ fact type อีก ซึ่งจะต้องทำการทดสอบในการแยกแยะไปเรื่อย ๆ จนไม่สามารถแยกแยะได้อีก

ขั้นตอนที่ 6: เพิ่ม entity type, mandatory role, subtype และ occurrence frequency constraints

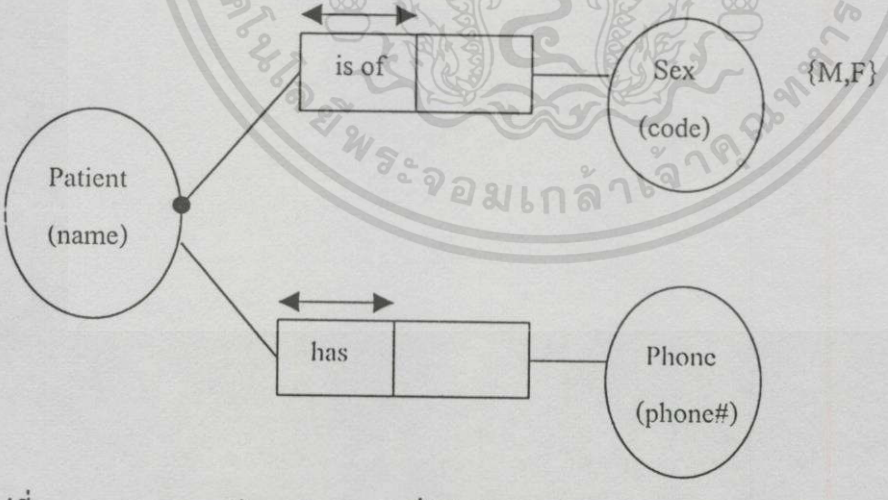
entity type constraints

พิจารณาในรูปที่ 4.22 จะเห็นได้ว่า entity type ของ Country เป็นกลุ่มประเทศที่อาจจะเป็นการรวมเข้าไว้ใน UoD อย่างไรก็ตามเป็นไปได้ที่จะมีบาง entity types ที่เป็นสมาชิกที่เราสามารถระบุได้จริง ๆ (เช่น เหรียญรางวัลในโอลิมปิกส์เกมมีเพียง 3 ชนิด คือ เหรียญทอง เหรียญเงิน และเหรียญทองแดง) จากข้อกำหนดเช่นนี้ MedalKind เป็นหนึ่งในตัวอย่างของ entity type constraint



รูปที่4.22 แสดง constraints ในการระบุ MedalKind และ Qty Mandatory และออปชั่นโรล

ในรูปที่ 4.23 แสดงถึงการระบุลักษณะของ role ของentity ที่ชื่อ Sex เป็น mandatory ซึ่งแสดงว่า Patient ทุกคนจะต้องมีเพศเสมอคือจะแสดง role ของ phone เป็นเพียง optional ซึ่งแสดงว่า patient ไม่จำเป็นต้องมี phone ทุกคน



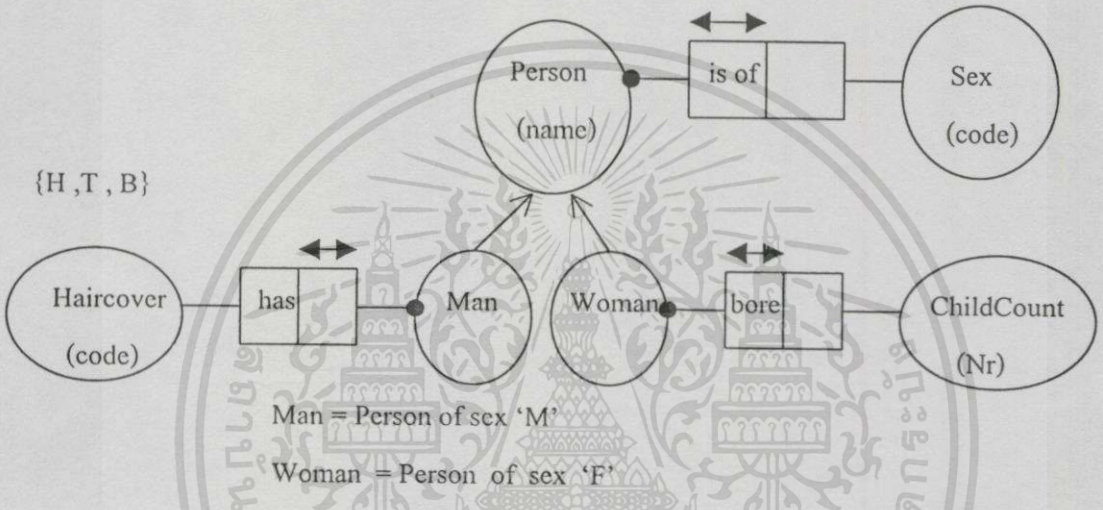
รูปที่4.23 แสดงความเป็น mandatory ที่ได้จากโรล

Subtypes

พิจารณา UoD ที่เป็น people ดังนั้นอาจจะพิจารณาได้ว่าเป็นผู้ชายหรือผู้หญิงได้เพราะผู้ชายส่วนมากหัวจะล้านซึ่งกำหนดให้ H (hairy); T (thin) และ B(bald) สำหรับผู้หญิงก็จะเป็นบุคคลที่

มีสามารถคลอดบุตร สำหรับเครื่องหมายที่แสดงลักษณะ "_" เป็นการไม่ได้บันทึกไว้ ซึ่งจะแตกต่างจาก "0"

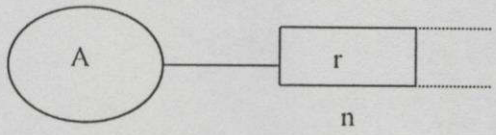
ต่อไปเราจะใช้ความรู้เกี่ยวกับ UoD ในการกำหนด uniqueness constraints (เรายอมรับกับคอลัมน์ของ Haircover และ NrChildren ที่มีการซ้ำซ้อน) จากตัวอย่างจะเห็นได้ว่า person มีเพียงหนึ่ง mandatory role และ สอง optional role โดยที่ Sex ของแต่ละบุคคลจะต้องมีการบันทึก แต่ใน haircover และจำนวนของ children เป็นการบันทึกแบบ optional แสดงSubtype ในรูปที่ 4.24 โดยมี Man และ Woman เป็น subtype มี person เป็น super type {M, F}



รูปที่4.24 แสดง conceptual schema ในลักษณะ subtypes

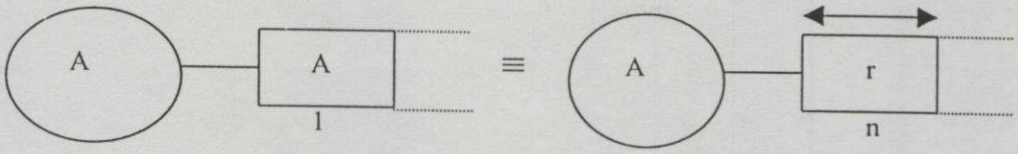
Occurrence frequencies

Occurrence frequency constraint เป็นความถี่ของข้อบังคับหรือเป็นจำนวนของ label ที่อาจจะเกิดขึ้นในคอลัมน์ของข้อเท็จจริง ดังตัวอย่างในรูปที่ 4.25

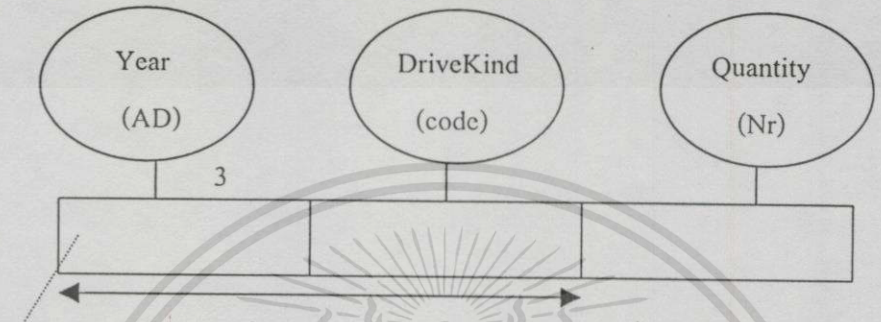


r เป็นคอลัมน์ที่จะเกิดจำนวน n ครั้ง

ถ้า n =1 แล้วความถี่ที่เกิดขึ้นในข้อบังคับจะ uniqueness constraint เช่น



{ HD, FD, T }

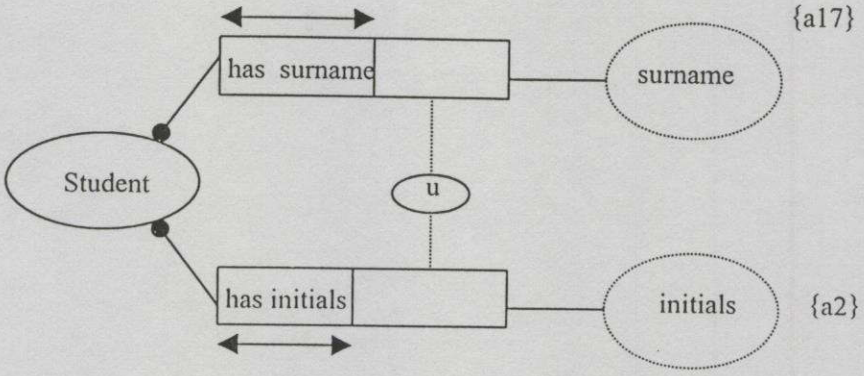


1986	HD	1000
1986	FD	7000
1986	T	1000
1987	HD	3000
1987	FD	4000
1987	T	500

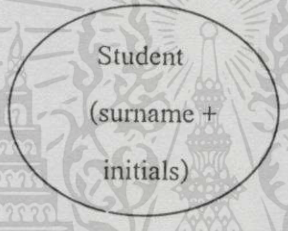
รูปที่4.25 แสดงการบันทึกการขายในแต่ละปี

ขั้นตอนที่ 7: ตรวจสอบในแต่ละ entity ที่ได้ระบุไว้

จากแบบจำลองข้อมูลไนแอมที่ได้ใช้เป็นตัวอย่างที่ผ่าน ๆ มานั้น จะเห็นได้ว่าเป็นจำลองที่แสดงถึงความสัมพันธ์ระหว่างชนิดเอนติตี้และชนิดเลเบิ้ลในลักษณะความสัมพันธ์แบบ 1 : 1 เพราะต้องการให้ง่ายต่อการอธิบายนั่นเอง ในโลกความเป็นจริงแล้วการอ้างอิงถึงสิ่งใด ๆ ในเอนติตี้หนึ่ง ๆ บางครั้งไม่สามารถอ้างได้โดยใช้เลเบิ้ลเดียวได้ ในรูปที่ 4.26 แสดงถึงการใช้ชนิดเลเบิ้ลอ้างอิงเอนติตี้ถึง 2 ชนิด และรูปที่ 4.27 แสดงรูปแบบอย่างย่อระหว่างชนิดเอนติตี้และชนิดเลเบิ้ล



รูปที่4.26 แสดงการอ้างอิงถึง schema ที่ได้จากสองเลเบิ้ล



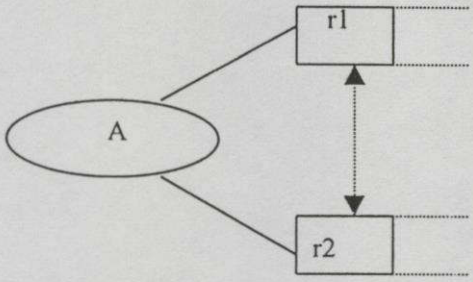
รูปที่4.27 แสดงรูปแบบอย่างง่ายของการทำงานในรูปที่ 4.26

ขั้นตอนที่ 8 : เพิ่ม equality, exclusion, subset และ constraints อื่น ๆ

ในขั้นตอนนี้เป็นการกล่าวถึงคอนสเตรนท์ที่ยังไม่ได้ยังไม่ได้กล่าวถึงที่ผ่านมา ซึ่งคอนสเตรนท์ที่จะกล่าวต่อไปนี้อย่างเช่น equality constraint, exclusion constraint และ subset constraint

Equality Constraint

จากรูป 4.28 จะเห็นได้ว่ามีหัวลูกศรแสดงระหว่างโรล ซึ่งเราจะเรียกคอนสเตรนท์นี้ว่าเป็น equality constraint สามารถอธิบายจากรูปนี้ได้ว่า ณ ภาวะใด ๆของฐานข้อมูลใน r1 ซึ่งได้ทำการบันทึกแล้ว ซึ่งจะ equal กับ กลุ่มของ r2 ที่ได้บันทึกแล้ว ดังสรุปได้ว่าจะมีการบันทึกใน r1 ก็ต่อเมื่อได้มีการบันทึก r2 แล้ว สำหรับการใช้หัวลูกศรในลักษณะดังรูปที่แสดงนั้นก็เพื่อใช้แทน "if and only if" นอกจากนี้เราสามารถแสดงรูปแบบของ equality



รูปที่4.28 แสดง equality constraint

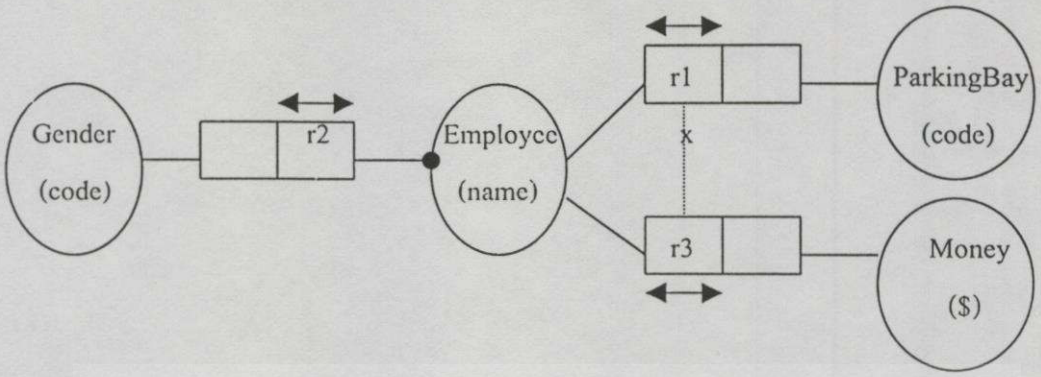
Exclusion Constraint

ตัวอย่างในตารางที่ 4.5 เป็นข้อมูลการจอดรถของบริษัทแห่งหนึ่ง ซึ่งการจอดรถนี้มีอยู่สองรูปแบบคือ อาจจะจอดในที่จอดรถของบริษัท หรืออาจจะไปจอดในที่จอดรถข้างนอก ซึ่งจะต้องเสียค่าใช้จ่ายเพิ่มเติม โดยทั่วไปแล้วพนักงานแต่ละคนจะเลือกกระทำได้แบบใดแบบหนึ่งเท่านั้น เพราะ แต่ละคนสามารถขับรถมาทำงานได้เพียงครั้งละคันเท่านั้น

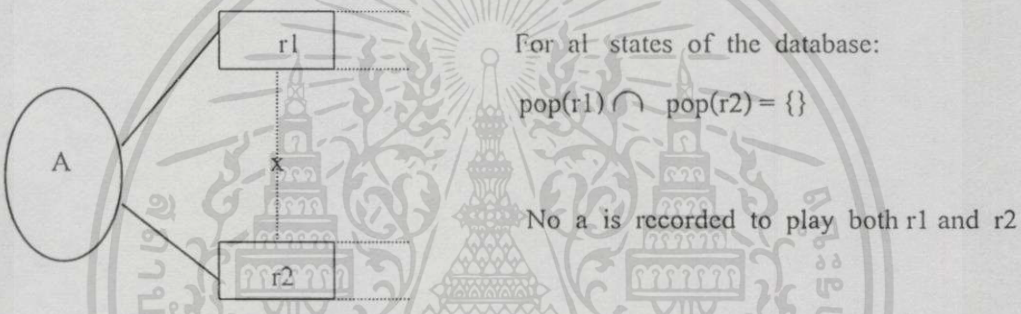
จากข้อมูลในตารางที่ปรากฏนี้จะเห็นได้ว่ามีชนิดข้อมูลว่างเปล่าอยู่ 2 รูปแบบนั่นก็คือ แบบ "-" และอีกแบบเป็น "?" โดยความหมายแล้ว "-" จะแสดงถึงไม่มีการบันทึกเพราะเนื่องมาจากจำเป็นต้องเลือกวิธีการจอดแบบใดแบบหนึ่งเท่านั้น แต่สำหรับกรณี "?" แสดงถึงยังไม่มีการบันทึกซึ่งอาจจะมีการบันทึกในอนาคตก็ได้ซึ่งในที่นี้แสดงว่าจะใช้สำหรับคนที่ยังไม่มีรถ จากข้อมูลเหล่านี้จะมีการบันทึกกฎข้อบังคับไว้โดยเรียกว่า Exclusion Constraint ดังรูปที่ 4.29 ซึ่งมี เครื่องหมาย "x" หรือกากบาทตรงกลางเส้นประของระหว่างโรล ซึ่งจะมีความหมายก็คือหากมีการบันทึกข้อมูลของพนักงานใน r1 แล้วจะต้องไม่มีการบันทึกข้อมูลของพนักงานใน r3 อีกครั้ง สำหรับรูปที่ 4.30 แสดงรูปแบบโดยทั่ว ๆ ไปในลักษณะ exclusion constraint

ตารางที่ 4.5 แสดงข้อมูลการจอดรถของบริษัทแห่งหนึ่ง

Employee	Gender	Parking bay	Parking claim($)
Adams B	F	C01	-
Bloggs F	M	-	200
Collins T	M	B05	-
Dancer S	F	-	250
Egghead E	M	?	?



รูปที่4.29 แสดงไดอะแกรม conceptual schema diagram ของตาราง 4.5



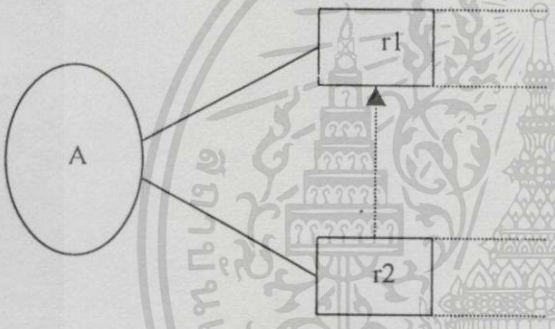
รูปที่4.30 แสดง exclusion constraint

Subset Constraint

จากตัวอย่างข้อมูลในตารางที่ 4.6 ซึ่งเป็นข้อมูลของการเป็นเจ้าของรถและการเป็นคนขับรถ โดยที่รถแต่ละคันจะใช้เลขทะเบียนเป็นชื่อเรียก จากข้อมูลจะสังเกตได้ว่ารถแต่ละคันมีเจ้าของได้หลายคน และในแต่ละคนสามารถเป็นเจ้าของรถได้หลายคัน จากข้อมูลในตารางนี้สามารถอธิบายได้ว่าผู้ขับรถนั้นจำเป็นต้องเป็นเจ้าของรถ แต่เจ้าของรถไม่จำเป็นต้องเป็นคนขับรถ ซึ่งจะเห็นได้ว่าข้อมูลชนิดความจริงที่ทำการบันทึกของข้อมูลในการขับรถจะเป็นซับเซ็ตของชนิดความของการเป็นเจ้าของรถ ซึ่งจะแสดงกฎข้อบังคับนี้ในลักษณะใช้เส้นประที่มีลูกศรข้างเดียวโดยชี้ไปที่ด้านเป็นซุปเปอร์เซ็ต เราจะเรียกคอนสเตรนท์นี้ว่า Subset Constraint สำหรับรูปแบบทั่ว ๆ ไปในลักษณะ subset constraint แสดงดังในรูป 4.31

ตารางที่ 4.6 แสดงข้อมูลของการเป็นเจ้าของรถและการเป็นคนขับรถ

Person	Sex	CarsOwned	CarsDriven
Fred	M	272NCP, 123BOM	272NCP, 123BOM
Sue	F	272NCP, 123BOM	272NCP
Tina	F	105ABC	?
Tom	M	?	?



รูปที่4.31 แสดง subset constraint

ขั้นตอนที่ 9 : ตรวจสอบความสมบูรณ์ของ conceptual schema ไม่ให้มีความซ้ำซ้อน

ขั้นตอนนี้เป็นขั้นตอนสุดท้ายในการออกแบบจำลองข้อมูล สำหรับการทำงานในขั้นตอนเป็นเพียงการทำงานเสริมที่เกี่ยวข้องกับการตรวจสอบความผิดพลาดที่อาจจะเกิดขึ้นในช่วงของการออกแบบและทำการแก้ไขข้อมูลเหล่านั้นให้มีความถูกต้อง สำหรับการทำงานขั้นตอนนี้ประกอบไปด้วยการทำงาน 3 รูปแบบก็คือ

1. การตรวจสอบกับตัวอย่างข้อมูล
2. การตรวจความซ้ำซ้อนของข้อมูล
3. การตรวจสอบขั้นสุดท้าย

บทที่ 5

การสร้าง Data Structure ของฐานข้อมูลเชิงวัตถุ

5.1 โมเดลเชิงวัตถุ (Object Modeling)

ออปเจกต์โมเดลเป็นลักษณะแสดงโครงสร้างถึงความสัมพันธ์ระหว่าง ออปเจกต์และแอทริบิ้ว และการกระทำที่เกิดขึ้นในแต่ละคลาสของออปเจกต์

5.1.1 ออปเจกต์และคลาส (Object and Classes)

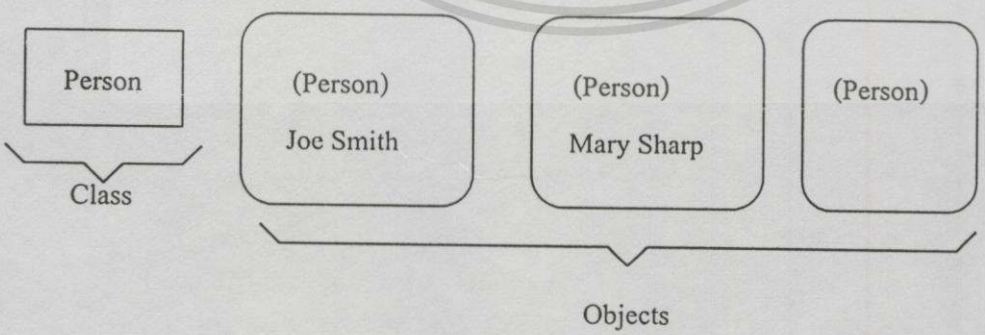
Objects

มีการใช้ในเรื่องออปเจกต์กับสิ่งพิมพ์ต่าง ๆ โดยไม่มีความชัดเจนซึ่งบางครั้งออปเจกต์หมายถึงสิ่งบางอย่างและบางครั้งก็จะอ้างอิงถึงสิ่งของอีกกลุ่มหนึ่งที่มีลักษระคล้าย ๆ กันโดยเราจะใช้ออปเจกต์อินแทนต์ (Object Instance) กับการอ้างอิงถึงกับสิ่ง ๆ เดี่ยว และอ้างอิงถึงกลุ่มสิ่งของที่คล้าย ๆ กันเป็นออปเจกต์คลาส โดยในรูปที่ 5.1 จะเป็นการแสดงให้ทราบลักษณะทั้งคลาสและออปเจกต์

Classes

ออปเจกต์คลาสจะอธิบายถึงกลุ่มของออปเจกต์ที่มีคุณสมบัติคล้าย ๆ กัน ตัวอย่างเช่น person, company, animal, process และ window ในแต่ละ person มีอายุ IQ และมีการทำงาน

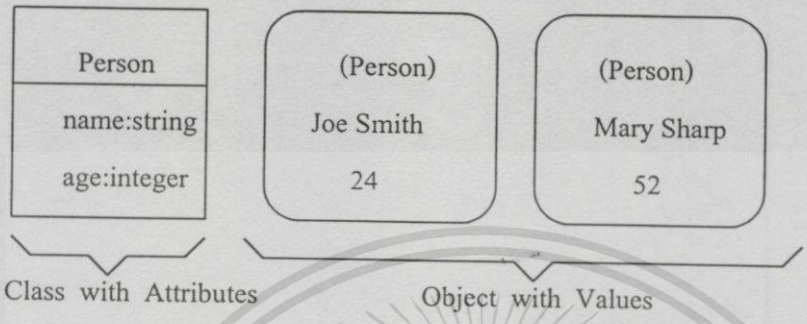
Object Diagrams



รูปที่ 5.1 แสดงคลาสและออปเจกต์

5.1.2 แอทริบิ้ว (Attributes)

แอทริบิ้วเป็นค่าของข้อมูลเป็นออปเจกต์ในคลาส เช่น name, age และ weight เป็นแอทริบิ้วของออปเจกต์ Person แสดงดังรูป 5.2

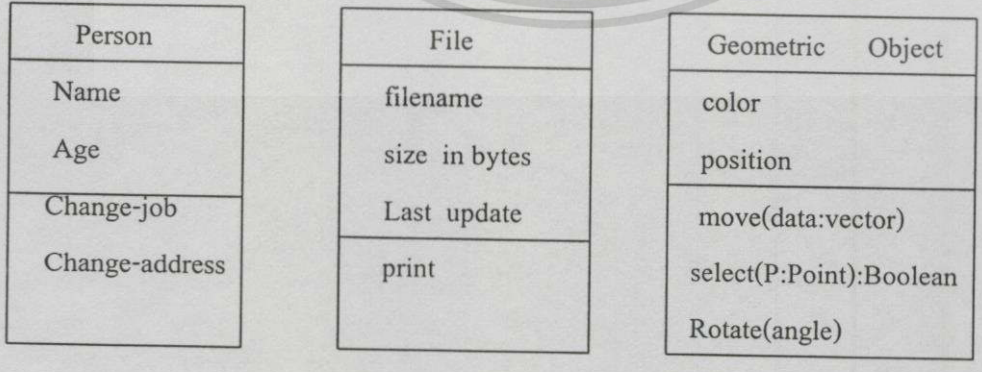


รูปที่ 5.2 แสดงแอทริบิ้วและค่าข้อมูล

5.1.3 โอเปอร์เรชั่นและเมทธอด (Operations and Methods)

Operation เป็นฟังก์ชั่นหรือการเปลี่ยนรูปโดยออปเจกต์ในคลาส สำหรับในคลาส company จะมี operation นั่นก็คือ Hire, fire และ pay-dividend สำหรับในคลาส window จะมี operation ก็คือ open, close, hide และ redisplay

กรณีจากรูปที่ 5.3 ข้างล่างนี้คลาส Person มีแอทริบิ้ว name และ age และมี operations เป็น changes-job และ change-address ใน File มี print เป็น operation สำหรับออปเจกต์ Geometric มี move, select และ rotate เป็นโอเปอร์เรชั่น



รูปที่ 5.3 แสดงโอเปอร์เรชั่น

5.1.4 วิธีการแปลงรูปจากออปเจกต์เป็นตาราง (Methodology of Transformation)

ในส่วนนี้จะเป็นงานวิจัยในระดับปริญญาเอกของนักศึกษาในมหาวิทยาลัย Swinburne ประเทศออสเตรเลีย ซึ่งจะเป็นวิธีพัฒนาการแมปโครงสร้างออปเจกต์ซึ่งจะเป็นการสร้าง object-oriented ให้อยู่ในรูปโครงสร้างตารางของ RDBMS

แบ่งแนวความคิดในการแมป object-oriented มีอยู่ 4 รูปแบบดังนี้คือ

1. แบบ Object เดี่ยว ๆ
2. แบบ inheritance
3. แบบ nonhierarchical relationship
4. แบบ aggregation

5.1.5 แนวคิดการทำแมปปิ้งออปเจกต์โอเรียนเตดเป็นตาราง (Mapping Object-Oriented Concepts to Tables)

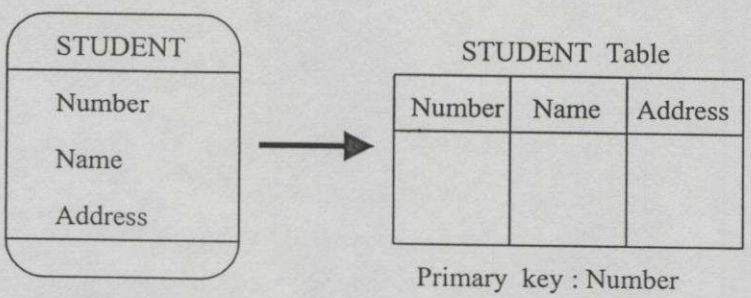
Mapping the Objects

ในแต่ละคลาสของออปเจกต์โอเรียลเตดโมเดลสามารถทำการแปลงรูปโดยตรงได้หนึ่งตารางของฐานข้อมูลเชิงสัมพันธ์ ความแตกต่างหลัก ๆ ของออปเจกต์นั่นก็คือ สามารถกำหนดค่าได้โดยเฉพาะในลักษณะ unique identifies ในขณะที่โมเดลเชิงสัมพันธ์จะมีค่าหลาย ๆ ค่าเพราะว่ากระทำได้จากทาง primary key

i) Object Identifier

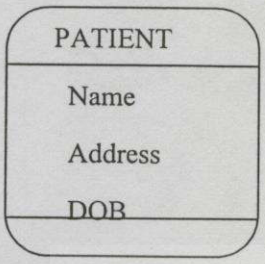
ในระบบ object-Oriented โดยออปเจกต์จะอนุญาตอ้างอิงโดยทาง unique ภายในโดย Object identifier ไม่ขึ้นอยู่กับค่าของ primary key ในกรณีนี้ภายในคลาสได้ทำการกำหนดแอทริบิ้วแล้ว โดยการใช้ unique identifier ซึ่งการกระทำแบบนี้ไม่มีปัญหาในการแมปไปอยู่ในรูปของตารางความสัมพันธ์ดังแสดงในรูปที่ 5.4 ซึ่งเป็นตัวอย่างในรูปคลาส STUDENT

Object-Oriented Model ----------→ Relational Table



รูปที่ 5.4 แสดงการแมปปิ้งออปเจกต์เป็นตาราง

อย่างไรก็ตามในความเป็นจริงแล้วออปเจกต์ไม่มีแอทริบิ้วที่ทำหน้าที่เป็น unique ในกรณีนี้จะต้องมีการกำหนด unique ID ให้กับออปเจกต์ดังรูปที่ 5.5 โดยแสดงตัวอย่างของคลาส PATIENT



PATIENT Table

ID	Name	Address	DOB

Primary key : ID

รูปที่ 5.5 แสดงการแมปปิ้งออปเจกต์โมเดลเป็นตารางเชิงความสัมพันธ์โดยมีการเพิ่ม ID

ii) Class Partition

อย่างไรก็ตามการพิสูจน์ประสิทธิภาพของการเข้าถึงตารางออปเจกต์ในคลาสอาจจะเป็นในลักษณะ partitional horizontal หรือ vertical ดังรูปที่ 5.6 และรูปที่ 5.7 ข้างล่างนี้

Frequently Accessed Object

ID	Name	Course
9051233	Grace	Computer
8976222	Linda	Business
9122333	Janet	Art

ID	Name	Course
9233444	Johnny	Engineering
9233555	Vision	Art

infrequently accessed objects

รูปที่ 5.6 แสดง Horizontal partition

ID	Name
9233444	Johnny
9233555	Vivien

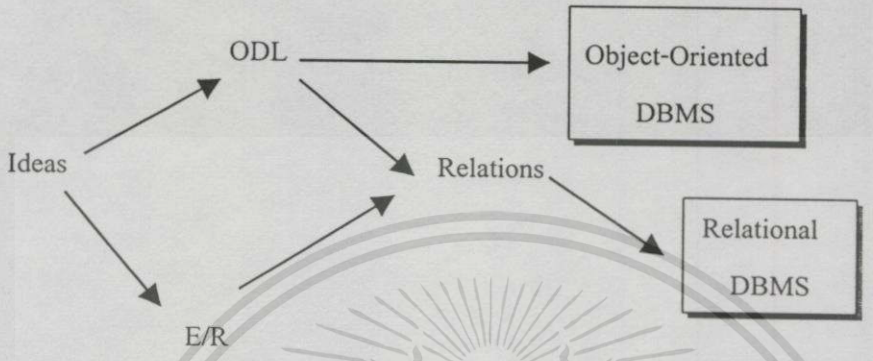
ID	Course
9233444	Engineering
9233555	Art

รูปที่ 5.7 แสดง Vertical Partition

5.2 ออกแบบ Data Structure ของฐานข้อมูลเชิงวัตถุ

ในกระบวนการออกแบบฐานข้อมูลจะเริ่มด้วยการวิเคราะห์ว่าฐานข้อมูลนั้นมีรายละเอียดอย่างไร และมีความสัมพันธ์ของข้อมูลเป็นอย่างไรบ้าง ซึ่งบ่อยครั้งโครงสร้างของฐานข้อมูลที่เรียกว่า

database schema นี้จะถูกระบุไว้ในภาษาทั่ว ๆ ไป นอกจากนี้ผู้ที่ทำการออกแบบฐานข้อมูลจะมีความนิยมที่จะเริ่มทำการพัฒนาสกีมม่าโดยใช้ E/R หรือ Object-based model แล้วทำการแปลงสกีมม่าให้อยู่ในรูปของ relational model เพื่อที่จะทำการพัฒนาเป็นเครื่องมือในขั้นต่อไปดังแสดงในรูปที่ 5.8

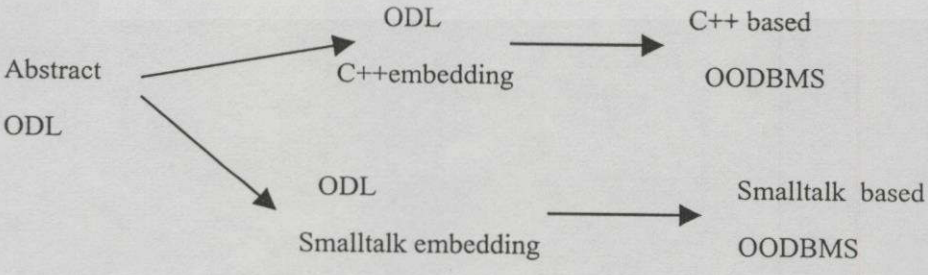


รูปที่ 5.8 แสดงโมเดลของฐานข้อมูลและกระบวนการทำงาน

5.2.1 ความรู้เดี่ยวกับ ODL เบื้องต้น (Introduction to ODL)

ODL (Object Definition Language) ถูกเสนอให้เป็นภาษามาตรฐานเพื่อใช้ระบุโครงสร้างของฐานข้อมูลในลักษณะ Object-Oriented การค้นหาจะกระทำโดยภาษาเช่น C++ หรือ Smalltalk

จุดมุ่งหมายเบื้องต้นของ ODL คือการออกแบบ Object-Oriented โดยการเขียนเป็นฐานข้อมูลแล้วทำการแปลงโดยตรงให้อยู่ในรูปของระบบการจัดการฐานข้อมูลแทนออปเจกต์โอเรียนเตด (OODBMS) โดยจะมี C++ หรือ Smalltalk เป็นภาษาพื้นฐาน (Primary Language) ทำให้ ODL จะต้องทำการแปลงไปเป็นภาษาใดภาษาหนึ่งดังที่ได้กล่าวไว้ข้างต้นดังรูปที่ 5.9



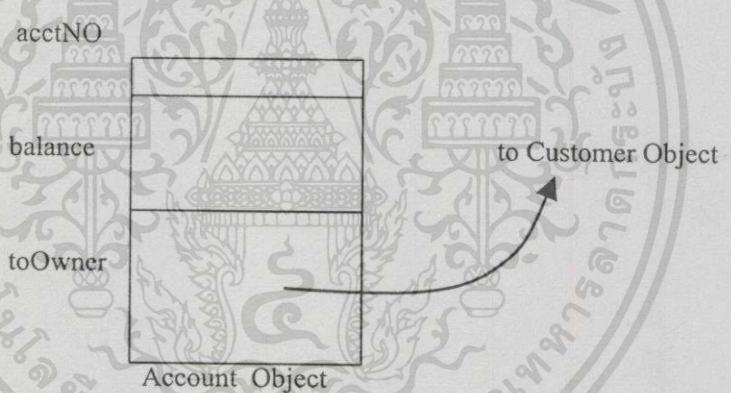
รูปที่ 5.9 แสดงการออกแบบการเปลี่ยนแปลงรูปแบบของ ODL

5.2.1.1 การออกแบบออปเจกต์โอเรียนเตด (Object-Oriented Design)

การออกแบบ Object-Oriented ในโลกของความเป็นจริงจะเป็นโมเดลที่กล่าวถึงในเรื่องออปเจต์ โดยสังเกตที่เอนติตี้ ตัวอย่างเช่นประชาชนทั่ว ๆ ไป อาจจะคิดสิ่งต่าง ๆ เป็นออปเจกต์ เช่นแผนกบัญชี เที่ยวบินของสายการบิน หลักสูตรในวิทยาลัย อาคาร เป็นต้น

โดยปกติเรากำหนดกลุ่มของออปเจต์ในรูปคลาส ที่มีคุณสมบัติคล้าย ๆ กันดังรูปที่ 5.10 ซึ่งแนวคิดเกี่ยวกับออปเจกต์และคลาสนั้น ในฐานข้อมูลจะถือว่ามีความสำคัญ อย่างไรก็ตามเราจะพูดถึง ODL object-oriented design เราก็จะคิดถึงคุณสมบัติที่คล้าย ๆ กันของออปเจกต์ที่เกิดขึ้นในคลาสที่มีความแตกต่างกัน 2 รูปแบบคือ

1. แนวคิดโลกแห่งความเป็นจริงจะนำเสนอ object ของ class ที่มีความคล้าย ๆ กัน สำหรับ instance จะเป็นกลุ่มของลูกค้าของธนาคารในหนึ่ง class และบัญชีอยู่อีก class หนึ่ง
2. คุณสมบัติของ object ใน class จะต้องมีลักษณะเหมือน ๆ กัน บ่อยครั้งเราจะคิดว่าออปเจกต์คือเรคคอร์ด



รูปที่ 5.10 แสดงออปเจกต์เกี่ยวกับบัญชี

การออกแบบคลาส ODL เราสามารถอธิบายคุณสมบัติได้ 3 ประเภทดังนี้คือ

1. Attributes เป็นคุณสมบัติที่สร้างขึ้นจาก integer หรือ string โดยที่แอทริบิ้วที่มี type นั้นจะไม่รวมอยู่ในคลาสใด ๆ
2. Relationships เป็นคุณสมบัติที่อ้างอิงถึงออปเจกต์ในบางคลาส จะอยู่ในรูป set
3. Methods เป็นฟังก์ชั่นที่อาจจะมีการประยุกต์การใช้งานของออปเจกต์ในคลาสต่าง ๆ

Interface Declarations

การประกาศคลาสใน ODL ในรูปแบบง่าย ๆ ที่สุดจะประกอบด้วย

1. keyword interface
2. ชื่อของ interface เช่น คลาส

3. วงเล็บปีกกาสำหรับแสดงคุณสมบัติของคลาส ซึ่งคุณสมบัติที่ว่านี้ก็คือ attribute, relationships และ methods

ตัวอย่างรูปแบบในการประกาศ interface มีลักษณะดังนี้คือ

```
interface <name> {
    <list of properties>
}
```

5.2.1.2 แอทริบิ้วใน ODL (Attribute in ODL)

แอทริบิ้วเป็นคุณสมบัติที่จะอธิบายลักษณะของออปเจกต์ อย่างเช่น ออปเจกต์ Person ซึ่งจะมีแอทริบิ้วเป็นชื่อ โดยที่มี type เป็น string นอกจากนี้ออปเจกต์ Person ยังเป็นไปได้ที่จะมีแอทริบิ้วเป็น birthdate ซึ่งจะมี 3 integer นั่นก็คือ year, month และก็ day

ตัวอย่าง

```
1) interface Movie{
2) attribute string title;
3) attribute integer year;
4) attribute integer length;
5)attribute enumFilm{color, blackAndwhite}filmType;
};
```

รูปที่ 5.11 แสดง An ODL declaration of the class Movie

จากตัวอย่างในรูปที่ 5.11 จะเห็นได้ว่าจากบรรทัดที่ 1) เป็นการประกาศชื่อคลาส ของ Movie โดยมี keyword interface ซึ่งจะมีการใช้ใน ODL เพื่อชี้ถึงคลาส จากบรรทัดที่1 จะเห็นว่ามีแอทริบิ้ว ของ Movie ทั้งหมด 4 แอทริบิ้วด้วยกัน โดยแอทริบิ้วแรกในบรรทัดที่ 2) มีชื่อเป็น title ซึ่งมี type เป็น string และ แอทริบิ้วทั้งสองก็คือ year และ length ในบรรทัดที่ 3) และ 4) ตามลำดับ ซึ่งสำหรับบรรทัดที่ 5) เป็นแอทริบิ้ว fileType มี type เป็น enumeration และชื่อของ enumeration คือ Film ลักษณะออปเจกต์ในคลาสของ Movie เราสามารถแสดงได้ดังนี้คือ

("Gone With the Wind",1939,231,color)

เราสามารถระบุคลาสของ Star ได้โดย

```
1) interface Star {
2) attribute string name;
3) attribute Struct Addr
        {string street, string city}address;
};
```

5.2.1.3 ความสัมพันธ์ใน ODL (Relationships in ODL)

เราสามารถแสดงความสัมพันธ์ระหว่างคลาส Movie และ Star เราสามารถแสดงได้ดังนี้คือ

```
relationship Set<Star> stars;
```

อธิบายได้ว่าในแต่ละออปเจกต์ของคลาส Movie จะประกอบไปด้วย Set ที่อ้างอิงถึงออปเจกต์ Star เซ็ตของการอ้างอิงเรียกว่า stars คีย์เวิร์ด relationship จะระบุ stars และคีย์เวิร์ด <Stars> จะชี้ถึง stars โดยอ้างอิงเซ็ตของออปเจกต์ Star ซึ่งมีมากกว่า 1 ออปเจกต์

ในรูปแบบทางกายภาพเราสามารถจินตนาการได้ว่ากลุ่ม stars สามารถนำเสนอโดยแสดงรายชื่อที่ชี้ไปถึงออปเจกต์ Star ซึ่งออปเจกต์ Star อาจจะไม่ปรากฏทางกายภาพภายในออปเจกต์ Movie

5.2.1.4 การทำอินเวิร์สความสัมพันธ์ (Inverse Relationships)

เราสามารถทำการชี้ถึงการเชื่อมต่อระหว่าง relationship stars และ starredIn โดยแทนที่ในแต่ละกลุ่มโดยการประกาศด้วยคีย์เวิร์ด inverse และชื่อของ realtionship อื่น ๆ ถ้าrelationship อื่น ๆ อยู่ในบางส่วนของคลาสอื่นแล้วเราจะอ้างอิงถึง relationship โดยใช้ชื่อของคลาส แล้วตามด้วยเครื่องหมายโคลอน :: และชื่อของ relationship

ตัวอย่าง

```
1) interface Star {
2) attribute string name;
3) attribute Struct Addr
        {string street, string city } address;
4) relationship Set<Movie> starredIn
     inverse Movie :: stars;
   };
```

รูปที่ 5.12 แสดงความสัมพันธ์ของคลาส และอินเวิร์ส ของ Star

จากตัวอย่างในรูปที่ 5.12 ให้ดูที่บรรทัดที่ 4 จะเห็นได้ว่าไม่เพียงจะประกาศ relationship starredIn แต่ยังมี inverse Movie :: stars

5.2.1.5 ความหลากหลายของความสัมพันธ์ (Multiplicity of Relationships)

บ่อยครั้งที่ความสัมพันธ์ที่เกิดขึ้นกับออปเจกต์จะเป็นในลักษณะเดี่ยว ๆ หรืออาจจะเป็นลักษณะความสัมพันะกับออปเจกต์อื่นๆ มากกว่า 1 ออปเจกต์ สำหรับ ODL นี้จะระบุการทำงานแบบนี้โดยการใช้หรือไม่ใช้ Set ในการประกาศถึงความสัมพันธ์ที่เกิดขึ้นได้ เมื่อเรามีการ inverse คู่ของความสัมพันธ์ ซึ่งมี 4 รูปแบบที่เป็นไปได้ นั่นก็คือ ความสัมพันธ์ที่เป็นแบบมีทิศทางเดียวกันมีทั้งสองทิศทาง และสุดท้ายก็คือมีทิศทางที่ไม่เหมือนกัน

ตัวอย่าง

```
1)  interface Movie{
2)  attribute string title;
3)  attribute integer year;
4)  attribute integer length;
5)  attribute enum Film {color, blackAndwhite} filmType;
6)  relationship Set<Star> stars
            inverse Star :: starredIn;
7)  relationship Studio ownedBy
            inverse Studio :: owns;
    };
8)  interface Star {
9)  attribute string name;
10) attribute Struct Addr;
            {string street, string city} address;
11) relationship Set<Movie>starredIn
            inverse Movie ::stars;
    };
12) interface Studio {
13) attribute string name;
14) attribute string address;
```

```
    15) relationship Set<Movie> owns
            inverse Movie::ownedBy;
    };
```

รูปที่ 5.13 แสดงคลาสใน ODL และความสัมพันธ์

จากตัวอย่างในรูปที่ 5.13 ข้างต้นจะเห็นได้ว่ามีการประกาศคลาสจำนวน 3 คลาส นั่นก็คือ Movie, Star, และ Studio ให้สังเกตว่าแอทริบิ้วที่อยู่ในคลาสของ Studio ถึงแม้จะมีชื่อเหมือนกัน แต่ type แตกต่างกันกรณีนี้ไม่ถือว่าผิด

สำหรับในบรรทัดที่ 7 เราจะเห็นว่า relationship เป็น ownedBy จาก movie ใน studios สำหรับ type ของ relationship Studio ไม่เป็น Set<Studio> เราจะประกาศได้ว่าในแต่ละ movie จะมีขึ้นใน studio เดียว และ inverse ของความสัมพันะจะอยู่ในบรรทัดที่ 15 ซึ่ง owns เป็นความสัมพันธ์ที่ได้จาก studios ใน movies สำหรับ type ความสัมพันธ์ Set<Movie> ที่ชี้โดยแต่ละ studio ที่มี owns ในกลุ่มของ movis ซึ่งบางทีอาจจะเป็น 0 อาจจะเป็น 1 หรืออาจจะมี movies จำนวนมาก ๆ

รูปแบบของการเกิด inverse ที่เราเรียกว่า multiplicity ของความสัมพันธ์ จะมีอยู่ 3 รูปแบบ คือ

1. A many-many เป็นความสัมพันธ์จากคลาส C ถึงคลาส D ที่มีกลุ่มของ D เกี่ยวเนื่องกันในแต่ละ C และมี inverse ความสัมพันธ์ที่เป็นกลุ่มของ C เกี่ยวเนื่องกันในแต่ละ D ตัวอย่างเช่น จากตัวอย่างข้างบนจะเห็นได้ว่า stars มีลักษณะ many-many จากคลาส Movie ไปยังคลาส Star และ starredIn จะเป็น many-many จาก Star ไปยัง Movie

2. A many-one เป็นความสัมพันธ์จากคลาส C ไปยังคลาส D ในแต่ละ C จะมี D เพียงตัวเดียว แต่ในความสัมพันธ์ของกลุ่ม C จะเกี่ยวเนื่องกันในแต่ละ D จากตัวอย่างข้างบนนี้กล่าวได้ว่า ownedBy เป็นความสัมพันธ์จาก Movie ไปยัง Studio นั่นก็คือ inverse ของความสัมพันธ์จะเป็น one-many จากคลาส D ไปยังคลาส C และความสัมพันธ์ owns จะเป็นแบบ one-many จาก Studio ไปยัง movie

3. A one-one เป็นความสัมพันธ์จากคลาส C ไปยังคลาส D โดยที่แต่ละ C จะมีความสัมพันธ์กับ D เพียงหนึ่งเดียวเท่านั้น และในการ inverse ความสัมพันธ์นั้นในแต่ละ D จะมีความสัมพันธ์เพียงหนึ่งเดียวใน C

หมายเหตุ : การใช้คำว่า “unique” จะให้ความหมายเช่นเดียวกับ “one” ทั้งใน many-one หรือ one-one

5.2.1.6 ประเภทของ ODL (Types in ODL)

ผู้ที่ทำการออกแบบฐานข้อมูลมีการเสนอรูปแบบของ ODL ว่ามีลักษณะคล้ายกับภาษา C หรือไม่ก็เหมือนกับรูปแบบภาษาทางโปรแกรมอื่น ๆ โดยรูปแบบที่สร้างขึ้นนั้นสร้างมาจากพื้นฐานที่ได้กำหนดขึ้นมาด้วยตัวมันเองและมีความแน่นอนในเรื่องกฎของการวนซ้ำ องค์ประกอบพื้นฐานของ ODL มีดังนี้คือ

1. Atomic types: ได้แก่ integer, float, character string, boolean, และ enumerations

2. Interface types เช่น Movie หรือ Star และจะมีการนำเสนอ types ซึ่งถือว่าเป็นโครงสร้างที่แท้จริงจะประกอบไปด้วยแอทริบิ้วและ relationship ของ interface

องค์ประกอบของโครงสร้าง type จะมีดังนี้คือ

1. Set ถ้า T เป็น type ใด ๆ แล้ว Set<T> จะแสดงถึง type ที่มีค่าจำกัดซึ่งเป็นสมาชิกอยู่ใน type ของ T

2. Bag ถ้า T เป็น type ใด ๆ แล้ว Bag<T> จะแสดงถึง type ที่มีค่า bags หรือ multisets ของสมาชิกใน type T ตัวอย่างการเกิด bag ที่มากกว่า หนึ่ง เช่น {1, 2, 1} เป็น bag แต่ไม่ใช่ set เพราะว่า 1 ปรากฏขึ้นมากกว่าหนึ่งครั้ง

3. List ถ้า T เป็น type ใด ๆ แล้ว List<T>จะแสดงถึง type ที่มีค่าจำนวนจำกัดคือมีค่าเป็นศูนย์หรือมากกว่านี้แต่ยังอยู่เป็นสมาชิกของ type T

4. Array ถ้า T เป็น type ใด ๆ และ i มีค่าเป็น integer แล้ว Array<T,i> จะชี้ถึง type ของสมาชิกอะเรย์ของสมาชิก i ใน type T ตัวอย่างเช่น Array<char,10> จะแสดงถึงว่าตัวอักษรมีความยาวได้เพียง 10 ตัวอักษรเท่านั้น

5. Structure ถ้า $T_1,\ldots,T_n$ เป็น type ใด ๆ และ $F_1, F_2, \ldots, F_n$ เป็นชื่อของฟิลด์ แล้ว

Struct N $\{T_1F_1, T_2F_2, \ldots, F_nT_n\}$

จาก 1 ถึง 4 นั่นก็คือ set, bag, list, และ array จะเรียกว่า collection types ซึ่งความแตกต่างระหว่างกฎเกี่ยวกับ type อาจจะเกี่ยวเนื่องกันกับแอทริบิ้ว และ relationships

- type ของแอทริบิ้วสร้างขึ้นจาก atomic type หรือไม่ก็ โครงสร้างที่เป็นฟิลด์ atomic type แล้วเราอาจจะประยุกต์ collection type ให้เกิดเป็นช่วงเริ่มต้น atomic type หรือไม่ก็เป็นโครงสร้าง

- type ของ relationship จะเป็น interface type หรือไม่ก็เป็น collection type ที่ได้ทำการประยุกต์ไปเป็น interface type

5.2.2 ไดอะแกรม Entity-Relationship (Entity-Relationship Diagrams)

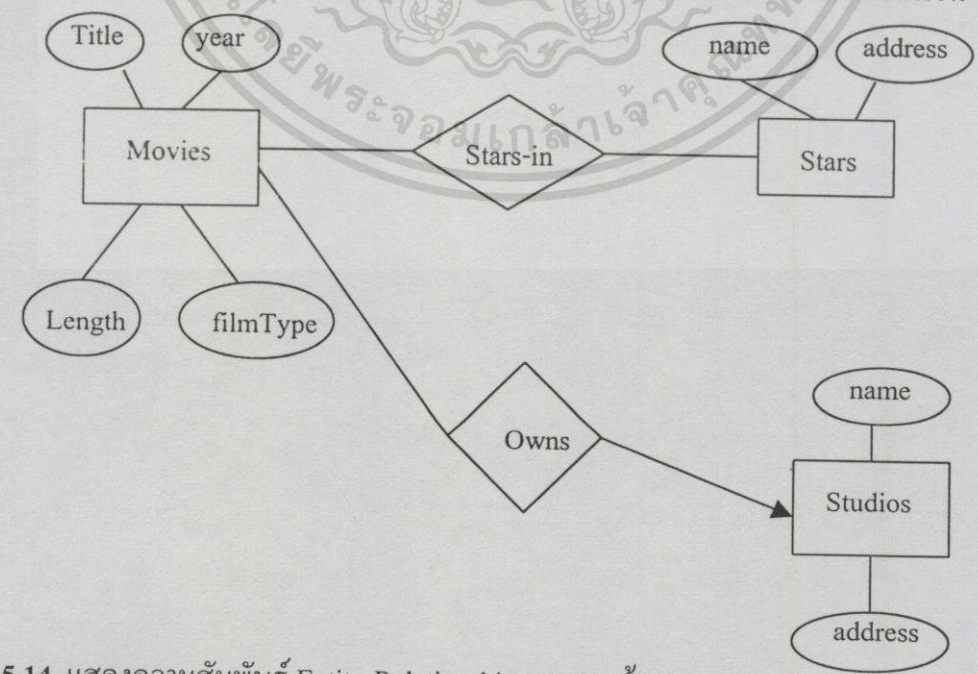
โมเดลฐานข้อมูลที่มีลักษณะเป็นกราฟิกซึ่งเรียกว่า entity-relationship diagrams มีสัญลักษณ์เหมือนกันกับ ODL ซึ่งเป็นวิธีทาง object-oreinted สำหรับไดอะแกรมแบบ Entity-Relationship (E/R) จะมีองค์ประกอบหลัก ๆ อยู่ 3 อย่างเหมือนกันกับที่มีใน ODL นั่นก็คือ

1. Entity sets ซึ่งเป็นเหมือนกับ คลาส โดยที่ชนิดเอนติตี้จะมีสมาชิกเป็นลักษณะกลุ่มชนิดเอนติตี้ เหมือนเช่นเดียวกันกับออปเจกต์ใน ODL
2. Attributes เป็นการอธิบายลักษณะของชนิดเอนติตี้ ซึ่งชนิดเอนติตี้ที่เกิดขึ้นทั้งใน E/R และ ODL นี้ จะมีแนวความคิดเหมือน ๆ กัน
3. Relationships เป็นการเชื่อมสัมพันธ์ระหว่างชนิดเอนติตี้ 2 ชนิดเอนติตี้ หรือมากกว่านั้น สำหรับ relationships ใน E/R นี้จะคล้ายๆ กันกับ relationship ใน ODL นั่นก็คือ

(a) ใน E/R โมเดล จะมีชื่อของ relationship เพียงหนึ่งชื่อที่เกิดขึ้นจากสองทิศทาง ในขอณะที่ ODL เราจะระบุทั้ง relationship และ inverse ตัวอย่างเช่น inverse relationship Movie :stars และ Star::starredIn

(b) Relationship ใน E/R โมเดลสามารถที่รวมเข้าด้วยกันมากกว่าสองชนิดเอนติตี้ ในขณะที่ ODL จะรวม relationship อย่างมากแค่สองคลาส

ตัวอย่างรูปแบบ E/R ที่มีการนำเสนอรายละเอียดเหมือนกันกับ ODL ที่ได้กล่าวมาก่อนหน้านี้ ซึ่งจะมีกลุ่มของชนิดเอนติตี้เป็น Movies, Stars และ Studios เราจะเห็นได้ว่าชื่อของกลุ่มชนิดเอนติตี้จะเป็นลักษณะพหูพจน์ ในขณะที่คลาสบ่อยครั้งจะมีชื่อเป็นเพียงเอกพจน์



รูปที่ 5.14 แสดงความสัมพันธ์ Entity-Relationship ของฐานข้อมูล movie

จากรูปที่ 5.14 จะเห็นได้ว่าชนิดเอนติตี้ Movie มี4 แอทริบิ้วเช่นเดียวกันกับคลาส Movie นั่นก็คือ title, year, length และ filmType นอกจากนี้จะเห็นได้ว่ามีหนึ่ง relationshipนั่นก็คือ Stars-inจะเป็นรายละเอียดที่เกี่ยวข้องกับ inverse ของ relationship stars และ starredIn จาก ODL ที่เป็นคลาส Movie และ Star ตามลำดับ สำหรับใน E/R จะเห็นว่าความสัมพันธ์ Owns ในรูปข้างบนนี้ จะนำเสนอ ODL ที่เป็น inverse relationship Movie::ownedBy และ Studio::owns ลูกศรที่ชี้ถึงชนิดเอนติตี้ Studios ในรูปข้างบนนี้จะชี้ให้เห็นว่าในแต่ละ movie จะถูกเป็นเจ้าของโดย studio เดียวเท่านั้น

5.2.2.1 ความหลากหลายของความสัมพันะแบบ E/R (Multiplicity of E/R Relationships)

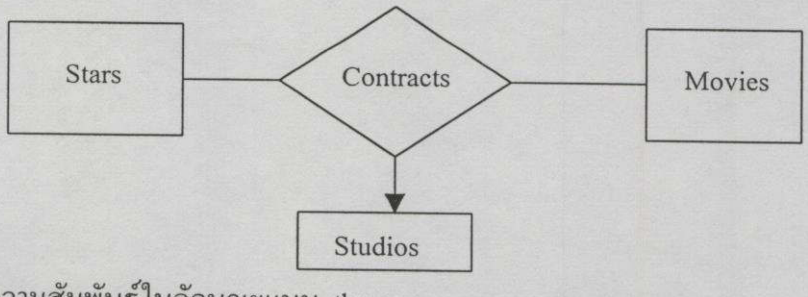
จากรูปที่มีลูกศรชี้จะเป็นการแสดงถึงความสัมพันธ์แบบมากกว่าหนึ่งในไดอะแกรม E/R ถ้าความสัมพันธ์เป็น many-one จากชนิดเอนติตี้ E ไปยังชนิดเอนติตี้ F แล้ว เราจะแสดงหัวลูกศรไปยัง F ลูกศรที่ชี้ในแต่ละชนิดเอนติตี้ของ E จะสัมพันธ์เพียงหนึ่งเดียวของชนิดเอนติตี้กับชนิดเอนติตี้ E ก็ได้

จากหลักการของความสัมพันธ์แบบ one-one ระหว่างชนิดเอนติตี้ E และ F นั้นจะถูกแทนที่ด้วยหัวลูกศรทั้ง E และ F ตัวอย่างเช่น ชนิดเอนติตี้ Studios และ Presidents แสดงได้ดังในรูปที่ 5.15 นี้คือ

รูปที่ 5.15 แสดงความสัมพันธ์แบบ one-to-one

5.2.2.2 ความสัมพันธ์แบบหลายเส้นทางเลือก (Multiway Relationships)

รูปแบบจะไม่มีความเหมือนกับ ODL ซึ่งใน E/R โมเดล สามารถทำให้สะดวกขึ้นได้โดยการกำหนดความสัมพันธ์รวมกันเข้ามากกว่าสองชนิดเอนติตี้ นั่นก็คือจะเป็นในลักษณะ ternary (three-way) หรือความสัมพันธ์แบบ higher-degree ดังตัวอย่างในรูปที่ 5.16 ข้างล่างนี้



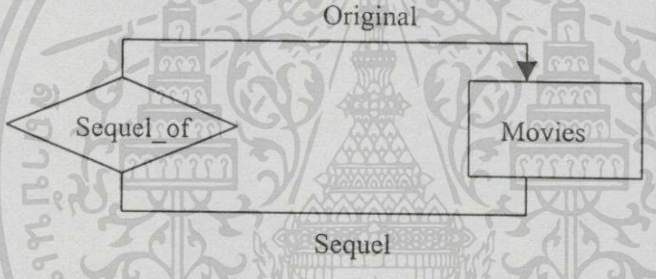
รูปที่ 5.16 แสดงความสัมพันธ์ในลักษณะแบบ three-way

จากรูปที่ 5.16 จะเห็นได้ว่าความสัมพันธ์ของ Constracts จะประกอบไปด้วย studio, star และ movie ซึ่งความสัมพันธ์ที่แสดงนั้นจะเห็นว่า studio ติดต่อเชื่อมกับ star ในลักษณะกระทำที่เกิดขึ้นใน movie โดยทั่ว ๆ ไปค่าของความสัมพันธ์ใน E/R สามารถที่จะคิดได้ว่าเป็นความสัมพันธ์ของทิวเพิล (tuple) ที่มีองค์ประกอบเป็นความสัมพันธ์ระหว่างชนิดเอนติตี้จากความสัมพันธ์ Contract สามารถอธิบายได้โดย 3 ทิวเพิล ที่มีรูปแบบคือ

(studio, star, movie)

5.2.2.3 บทบาทหน้าที่ในความสัมพันธ์ (Roles in Relationships)

เป็นไปได้ที่ชนิดเอนติตี้ใด ๆ สามารถที่จะปรากฏถึงสองครั้งหรือมากกว่านั้นในความสัมพันธ์แบบเดี่ยว ๆ ถ้าวาดรูปภาพก็จะมีหลาย ๆ เส้นจากความสัมพันธ์เดียวไปยังชนิดเอนติตี้อื่น ๆ ซึ่งแต่ละเส้นที่เชื่อมต่อกับชนิดเอนติตี้จะนำเสนอถึงความแตกต่างของ role ซึ่งสามารถแสดงความสัมพันธ์ที่เกิดขึ้นกับชนิดเอนติตี้ดังภาพที่ 5.17 ข้างล่างนี้

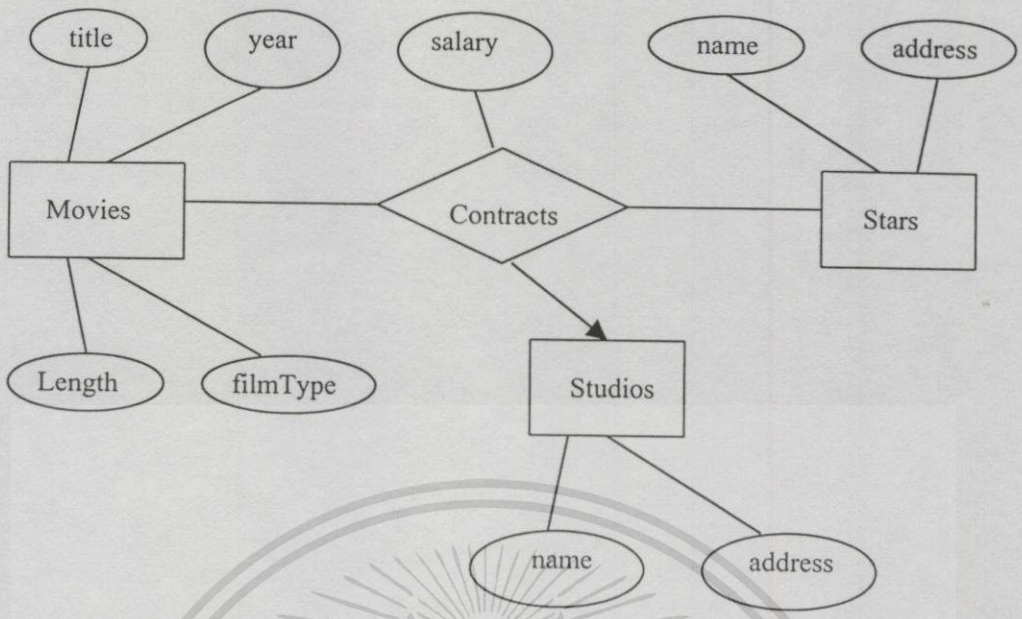


รูปที่ 5.17 แสดงโรลในความสัมพันธ์

จากรูปจะเห็นๆได้ว่าความสัมพันธ์ Sequel-of ระหว่างชนิดเอนติตี้ Movies และตัวมันเองโดยแต่ละความสัมพันธ์ระหว่างสอง movies มีอันหนึ่งที่เป็น sequel จากความแตกต่างของสอง movie ในความสัมพันธ์โดยมี role เป็น Original และอีก role หนึ่งเป็น Sequel จะเป็นได้ว่าคลาสสัมพันธ์นี้เป็น many-one จาก Sequel ที่เป็น movies ไปยัง Original ใน movies

5.2.2.4 แอทริบิ้วในความสัมพันธ์ (Attributes on Relationships)

จากรูปที่ 5.18 ไดอะแกรม E/R จะเห็นว่าในส่วนของแอทริบิ้ว salary ที่อยู่บนความสัมพันธ์ของ Contacts เราจะทำการสร้างเป็นชนิดเอนติตี้ว่า Salaries ซึ่งมี salary เป็นแอทริบิ้ว โดย Salaries นี้จะเป็นชนิดเอนติตี้ที่สี่ ที่มีความสัมพันธ์กับ Contracts



รูปที่ 5.18 แสดงแอทริบิ้วในความสัมพันธ์

5.2.2.5 สับคลาส (Subclasses)

บ่อยครั้งที่คลาสจะบรรจุออปเจกต์ นั่นก็คือจะมีคุณสมบัติพิเศษที่ไม่เป็นสมาชิกในคลาสนั้น ๆ เราพบว่าการทำงานในคลาสที่มีลักษณะเช่นนี้ว่า สับคลาส (subclass) ซึ่งจะมีลักษณะของแอทริบิ้ว และ relationship ที่พิเศษนั่นก็คือจะมีการเพิ่มจำนวนคลาสขึ้นมา สำหรับ ODL จะมีความง่ายที่จะทำการประกาศในลักษณะสับคลาส

Subclasses in ODL

กรณีประเภทของ movies เราอาจจะเก็บไว้ในฐานข้อมูลเป็นประเภทต่าง ๆ เช่น การ์ตูน ฆาตรกรรม การผจญภัย หรือแบบแนวชีวิต เป็นต้น ซึ่งแต่ละประเภทของ movie เราสามารถกำหนดเป็นสับคลาสของคลาส movie ได้ นั่นก็คือกำหนดคลาสขึ้นมาหนึ่งคลาสให้เป็น C แล้วมีวสับคลาสเป็น D โดยมีการประกาศชื่อ C ตามด้วยเครื่องหมายโคลอน (:) และชื่อ D

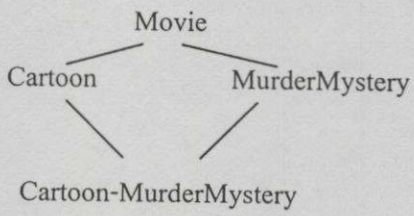
ตัวอย่างเช่น

ประกาศให้ Cartoon เป็นสับคลาสของ Movie และ Movie เป็น superclass ของ Cartoon ในรูปแบบ ODL เป็นดังนี้คือ

```
1) interface Cartoon : Movie{
2) relationship Set<Star> voices;
   };
```

5.2.2.6 คุณสมบัติการถ่ายทอดคุณลักษณะใน ODL (Multiple Inheritance in ODL)

ในคลาสหนึ่ง ๆ อาจจะมีอยู่หลาย ๆ สับคลาส ดังรูปที่ 5.19 โดยที่แต่ละสับคลาสจะมีคุณสมบัติถ่ายทอดมาจาก superclass

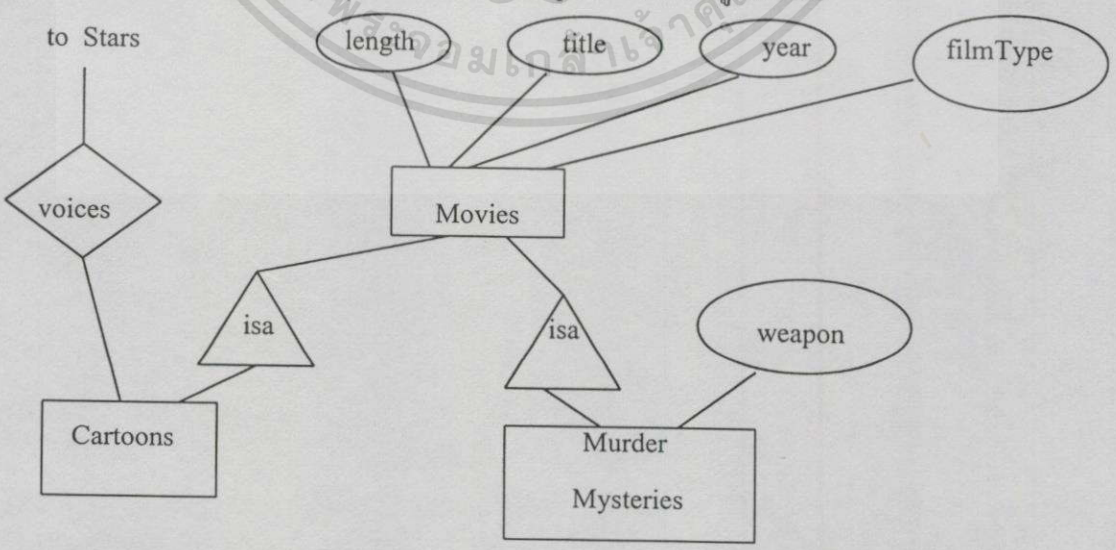


รูปที่ 5.19 แสดงไดอะแกรมแบบหลาย ๆ inheritance

5.2.2.7 สับคลาสในไดอะแกรมแบบ Entity-Relationship (Subclasses in Entity-Relationship Diagram)

คลาสใน ODL จะเป็นชนิดเอนติตี้ในรูปแบบของ E/R โมเดล สมมติว่าคลาส C เป็นสับคลาสของคลาส D แล้วจะแสดงใน E/R โมเดล ได้ว่า ความสัมพันธ์ที่เกิดขึ้นของชนิดเอนติตี้ระหว่างคลาส C และ D โดยจะเป็นความสัมพันธ์พิเศษที่เรียกว่า isa เราสามารถจะเขียนด้วยลักษณะเป็นกล่อง (boxes) สำหรับเป็นชนิดเอนติตี้ของ C และ D โดยที่แอทริบิ้วใด ๆ หรือความสัมพันธ์ใด ๆ ที่เป็นของชนิดเอนติตี้ C ที่ได้เชื่อมติดต่อกับกล่อง C แอทริบิ้วก็จะเป็นทั้ง C และ D โดยสภาพยังคงอยู่ที่ D

isa จะเขียนในลักษณะรูปสามเหลี่ยมมีคำว่า "isa" อยู่ตรงกลางของรูปสามเหลี่ยม ซึ่งที่จุดมุมยอดของสามเหลี่ยมจะเป็นจุดที่เชื่อมต่อกับ superclass ดังในรูปที่ 5.20



รูปที่ 5.20 แสดงความสัมพันธ์ลักษณะ Isa ในไดอะแกรม E/R

5.2.2.8 การถ่ายทอดคุณลักษณะในแบบ E/R โมเดล (Inheritance in the E/R Model)

มีความแตกต่างระหว่างแนวคิดของ inheritance ใน ODL หรือในออปเจกต์โอเรียนเตดโมเดลใด ๆ ก็ตาม รวมทั้งใน E/R โมเดล ซึ่ง ODL ออปเจกต์จะต้องเป็นสมาชิกของคลาสใดคลาสหนึ่งที่แน่นอน

ใน E/R โมเดล เราจะเห็นว่ามีชนิดเอนติตี้ที่อาจจะเป็นชนิดเอนติตี้ที่เป็นส่วนหนึ่งของ isa-hierarchy เดี่ยว ๆ ซึ่งจะมีองค์ประกอบเชื่อมต่อในรูปของชนิดเอนติตี้เดี่ยว ๆ โดยคอลัมน์ isa

5.2.2.9 โมเดลของกฎข้อบังคับ (The Modeling of Constaints)

คุณสมบัติที่เป็นแอทริบิ้วและ relationship นั้นทั้งใน ODL และที่มีการใช้ชนิดเอนติตี้และ relationship ใน E/R โมเดล จากสิ่งเหล่านี้เขามีการทำงานโดยการแบ่งเป็นย่อย ๆ เป็นโมเดลได้อย่างไร ซึ่งเราจะสนใจโครงสร้างกันเป็นจำนวนมากว่าโมเดลเหล่านี้มีการคิดค้นอย่างไร อย่างไรก็ตามจะมีคุณสมบัติที่สำคัญบางอย่างที่เราไม่สามารถแสดงให้เป็นไปได้ตามความเป็นจริง โดยทั่ว ๆ ไปรายละเอียดของข้อมูลที่ได้ใช้นั้นจะเป็นลักษณะของ constraints ที่ใช้กำหนดคลาส แอทริบิ้ว และ relationship

คุณลักษณะของ constraint ที่ใช้มีดังนี้คือ

1. keys ซึ่งเป็นแอทริบิ้วหรือเป็นกลุ่มของแอทริบิ้วที่จะมีความเป็น unique ของออปเจกต์ภายในคลาสหรือของชนิดเอนติตี้ภายในกลุ่มของชนิดเอนติตี้ ซึ่งจะไม่มีสองออปเจกต์ในคลาส ที่จะมีค่าที่เหมือนกันในแต่ละกลุ่มของแอทริบิ้ว
2. Single-value constraints จะมีค่าของ role ที่เป็น unique โดยทั่วไปคีย์จะเป็นแหล่งของ single-value constraints
3. Referencetial integrity constraints เป็นการอ้างอิงโดยมีบางออปเจกต์ที่มีอยู่จริงในฐานข้อมูล
4. Domain constraints เป็นค่าของแอทริบิ้วที่ต้องเขียนมาจากการระบุค่าหรือขอบเขตที่ระบุ
5. General constraints เป็นการยืนยันถึงความพอใจที่จพกำหนดกฎเกณฑ์ขึ้นมาใช้ตามความต้องการ

5.2.2.10 คีย์ (Keys)

ใน ODL ให้คีย์ในคลาสเป็นกลุ่มของ K ซึ่งจะมีหนึ่งแอทริบิ้วหรือมากกว่านี้ นั่นก็คือจะมีความแตกต่างกันของออปเจกต์ O1 และ O2 ในคลาส O1 และ O2 ไม่สามารถที่จะระบุค่าในแต่ละแอทริบิ้วในคีย์ K สำหรับใน E/R โมเดล คีย์จะเหมือนกันแต่ละคลาสจะถูกแทนที่ด้วยชนิดเอนติตี้ และออกเจกต์จะถูกแทนที่ด้วยชนิดเอนติตี้

Declaring keys in ODL

ใน ODL เราจะประกาศหนึ่งแอทริบิ้วหรือมากกว่านี้ให้เป็นคีย์ในคลาสโดยใช้คีย์เวิร์ดเป็น key ตัวอย่าง

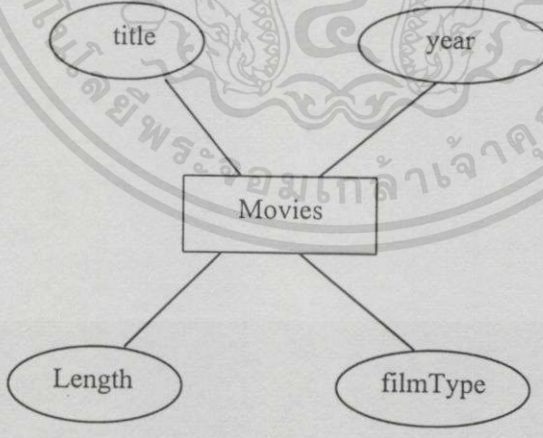
```
interface Movie
    (key(title,year))
{
```

การประกาศคีย์จะต้องทำหลังจากการประกาศ interface ก่อนที่จะเปิดด้วยเครื่องหมายวงเล็บปีกกาหรือก่อนจะประกาศแอทริบิ้วหรือ relationship ใด ๆ

5.2.2.11 การนำเสนอคีย์ในรูปแบบ E/R โมเดล (Representing Keys in the E/R Model)

ในกลุ่มของเอนติตี้จะเริ่มจากคลาส เราสามารถกำหนดคีย์เหมือนกันกับที่ทำในคลาสของ ODL ถ้ารูปแบบแอทริบิ้วที่เป็นคีย์ของเอนติตี้แล้วเราไม่สามารถที่จะมีสองชนิดเอนติตี้ที่มีค่าเหมือนกันในแต่ละแอทริบิ้วของคีย์ ในไดอะแกรม E/R เราจะขีดเส้นใต้ของแอทริบิ้วที่เป็นคีย์ในกลุ่มของเอนติตี้

ตัวอย่าง



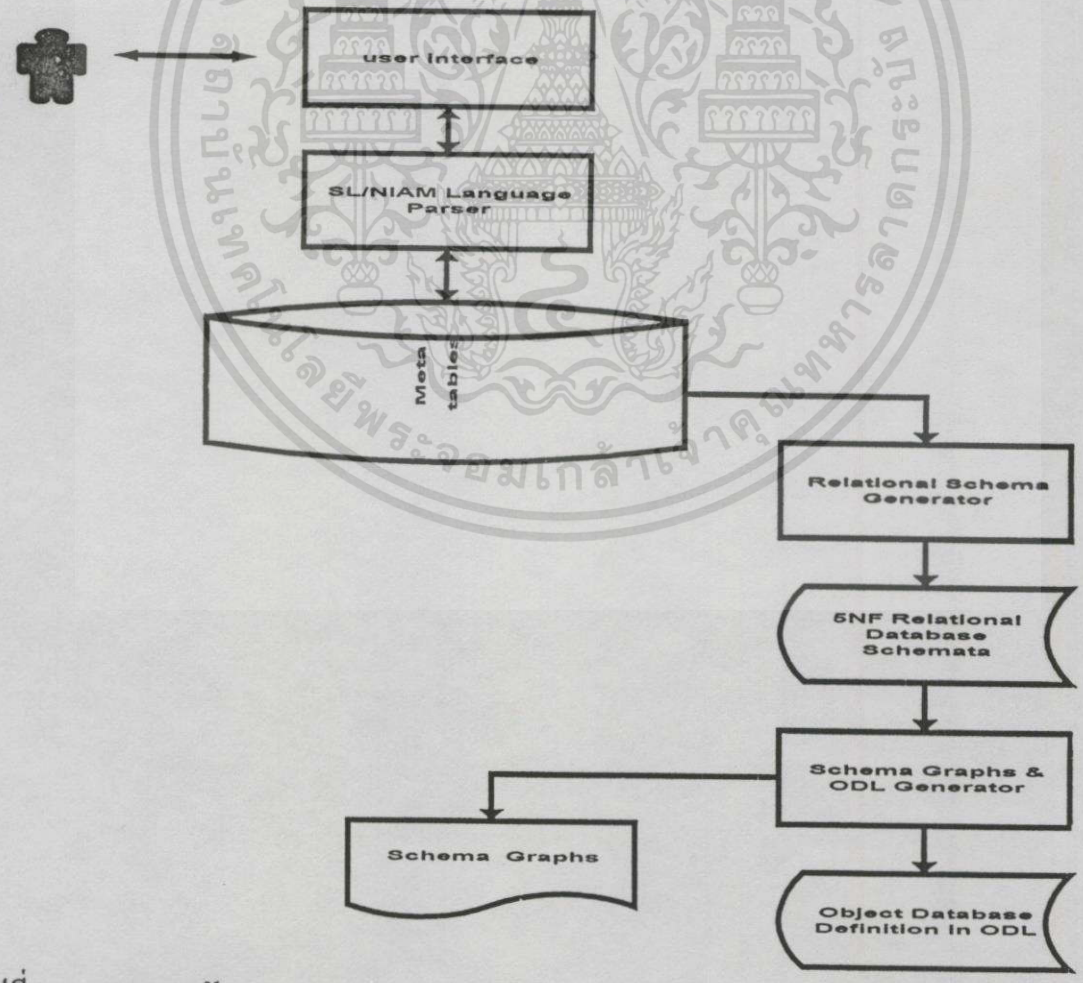
รูปที่ 5.21 แสดง Movies entity set with key indicated

จากรูปที่ 5.21 จะเห็นได้ว่าใน E/R โมเดล มีคีย์มากกว่าหนึ่งคีย์ซึ่งเราจะไม่ทำการชี้บอกถึงทุก ๆคีย์ จึงมีการออกแบบให้เป็นคีย์เดียวที่เรียกว่าเป็น primary key ในขณะที่ primary key เป็นคีย์ที่ได้ขีดเส้นใต้ใน E/R โมเดล สำหรับคีย์อื่น ๆ จะเรียกว่าเป็น secondary keys

บทที่ 6

สถาปัตยกรรมระบบ

การทำงานในส่วนนี้เป็นการประยุกต์แนวคิดและทฤษฎีต่าง ๆ ที่ผู้วิจัยได้ทำการศึกษาค้นคว้ามา เพื่อพัฒนาเป็นรูปแบบงานวิจัยของผู้ทำวิจัยในครั้งนี้ โดยทำการแสดงเป็นรูปแบบทางสถาปัตยกรรมระบบดังรูป 6.1 ซึ่งจะเห็นได้ว่าเราสามารถจะแบ่งเป็นหน่วย ๆ ของรูปแบบสถาปัตยกรรมระบบนี้ออกเป็น 5 หน่วยด้วยกันก็คือ หน่วยติดต่อผู้ใช้รับข้อมูลในรูปภาษา SL/NIAM (User Interface) หน่วยตรวจสอบหลักไวยากรณ์ทางภาษา SL/NIAM (SL/NIAM Language Parser) หน่วยของฐานข้อมูลเมทด้า (Meta Table) หน่วยการสร้างฐานข้อมูลเชิงสัมพันธ์โดยพัฒนาจากรีเลชั่นสกีม่า และสุดท้ายเป็นหน่วยของการสร้าง Data Structure ของฐานข้อมูลเชิงวัตถุอยู่ในรูป ODL โดยพัฒนาต่อจากสกีมม่ากราฟ



รูปที่ 6.1 แสดงสถาปัตยกรรมระบบ

6.1 หน่วยติดต่อผู้ใช้รับข้อมูลในรูปภาษา SL/NIAM

การทำงานในหน่วยนี้จะทำหน้าที่รับข้อมูลในรูปประโยคภาษา SL/NIAM จากผู้ใช้งานโดยตรง

6.2 หน่วยตรวจสอบหลักไวยากรณ์ทางภาษา SL/NIAM

การทำงานของหน่วยนี้เป็นการตรวจสอบความถูกต้องของหลักไวยากรณ์จากการใช้รูปประโยคภาษา SL/NIAM ของผู้ใช้งาน โดยหากรูปประโยคต่างๆ ที่ผ่านการตรวจความถูกต้องทางหลักไวยากรณ์ว่ามีความถูกต้องแล้วจะถูกนำไปเก็บไว้ที่ฐานข้อมูลเมทต้าเพื่อนำไปใช้ในกระบวนการต่อๆ ไป สำหรับการตรวจสอบหลักไวยากรณ์ทางภาษา SL/NIAM นี้เราได้ใช้ yacc/lex (อยู่ในภาคผนวก ข) ซึ่งเป็นโปรแกรมที่ใช้สำหรับสร้างพาร์สเซอร์เพื่อตรวจสอบหลักไวยากรณ์ทางภาษา

6.3 หน่วยของฐานข้อมูลเมทต้า

การทำงานของหน่วยนี้จะทำหน้าที่ในการจัดเก็บข้อมูลลงในฐานข้อมูล ซึ่งข้อมูลที่ได้จัดเก็บจะเป็นกฎข้อบังคับความถูกต้องของข้อมูลที่ผ่านการตรวจสอบความถูกต้องทางหลักไวยากรณ์ภาษา SL/NIAM แล้ว สำหรับฐานข้อมูลที่ใช้ในการจัดเก็บนี้ผู้วิจัยได้พัฒนาขึ้นมาจากการใช้แนวคิดการสร้างกฎข้อบังคับความถูกต้องของข้อมูลเชิงแนวคิดไนแอม ดังรูปที่ 6.2

จากรูปที่ 6.2 สามารถอธิบายรายละเอียดของเอนติตี้ต่าง ๆ ได้ว่าแต่ละเอนติตี้นั้นมีหน้าที่ในการเก็บข้อมูลอะไรได้บ้าง และในแต่ละเอนติตี้เหล่านี้จะมีความสัมพันธ์กันอย่างไรบ้าง เราจะอธิบายได้ดังตารางที่ 6.1 ต่อไปนี้คือ

ตารางที่ 6.1 แสดงรายละเอียดของเอนติตี้

ชนิดของเอนติตี้	หน้าที่ในการทำงาน
Constraint	เป็นข้อบังคับที่ใช้แอทริบิ้ว const# ในการอ้างอิงเพื่อใช้งาน
Derived Fact Type	เป็นสับเซ็ตของ Fact Type ใช้แทนความสัมพันธ์ที่ได้จากการคำนวณของขอ้มูลอื่น ใช้แอทริบิ้ว Rel_Name เป็นตัวอ้างอิงในการทำงาน
Derived Formula	ใช้เก็บสูตรของข้อมูลที่คำนวณโดยใช้ Form เป็นแอทริบิ้วในการอ้างอิงเพื่อใช้งาน
Entity_Type	เป็นชนิดเอนติตี้ที่มี Obj_Nameเป็นแอทริบิ้วในการอ้างอิงเพื่อใช้
Fact Type	เป็นชนิดความจริงมีรูปแบบการทำงาน 2 รูปแบบคือ รูปแบบปกติ

ตารางที่ 6.1 (ต่อ)

	(Normal) และรูปแบบหาได้ (Derived) มีการใช้ Rel_Name เป็นแอทริบิ้วสำหรับอ้างอิง
Frequency	เป็นค่าความถี่ที่ได้กำหนดไว้ใน Occurrence Frequency ใช้แอทริบิ้ว Time ในการอ้างอิงเพื่อใช้งาน
Kind_Of_Obj	ประเภทของออปเจกต์ที่ใช้อแทริบิ้ว Kind ในการอ้างอิงการทำงาน
Kind_Of_Rel	เป็นประเภทของรีเลชั่นที่ใช้แอทริบิ้ว Kind ในการอ้างอิงการทำงาน
Kind_Of_Const	เป็นประเภทของคอนสเตรนท์ที่ใช้ Kind ในการอ้างอิงการทำงาน
Label Type	ชนิดของเลเบิ้ลโดยเป็นสับเซ็ตของออปเจต์ไทป์ ใช้ Obj_Name ในการอ้างอิงเพื่อใช้งาน
Label_Size	เป็นขนาดของเลเบิ้ลที่ใช้ Size เป็นแอทริบิ้วในการอ้างอิงเพื่อใช้งาน
Limit	ขีดจำกัดที่กำหนดให้กับคอนสเตรนท์ต่าง ๆ ซึ่งมีทั้งขีดจำกัดบน และขีดจำกัดล่าง ใช้ Lim เป็นแอทริบิ้วในการอ้างอิงเพื่อใช้งาน
Membership Const	เป็นคอนสเตรนท์ ที่ใช้ Const# เป็นแอทริบิ้วในการอ้างอิงการทำงาน
Member	เป็นสมาชิกในคอนสเตรนท์ที่ใช้ MEMBER# เป็นแอทริบิ้วในการอ้างอิงเพื่อใช้งาน
Normal Role	เป็นโรลปกติที่ใช้ Role_Name เป็นแอทริบิ้วในการอ้างอิงเพื่อใช้งาน
Nesting Role	เป็นโรลแบบโครงร่างตาข่ายใช้ Role_Nname เป็นแอทริบิ้วในการอ้างอิงเพื่อใช้งาน
Normal Fact Type	เป็นแฟกท์ไทป์ปกติที่เป็นสับเซ็ตของแฟกต์ไทป์ใช้ Rel_Name เนแอทริบิ้วในการอ้างอิงเพื่อใช้งาน
Object_Type	เป็นประเภทของออปเจกต์ต่าง ๆ ใช้ Obj_Nnme เป็นแอทริบิ้วในการอ้างอิงเพื่อใช้งาน
Object Type Const	เป็นคอนสเตรนท์แบบออปเจกต์ไทป์ใช้ Const# เป็นแอทริบิ้วในการอ้างอิงเพื่อใช้งาน
Occurrence Freq	เป็นคอนสเตรนท์แบบ Occurrence Frequency เป็นสับเซ็ตของคอนสเตรนท์ใช้ Const# เป็นแอทริบิ้วในการอ้างอิงเพื่อใช้งาน
Role	โรล เป็นบทบาทหน้าที่ที่กระทำระหว่างเอนติตี้หรือ เลเบิ้ลต่าง ๆ ใช้ Role_Name เป็นแอทริบิ้วในการอ้างอิงเพื่อใช้งาน
Role_Type	เป็นชนิดของโรลต่าง ๆ ซึ่งมีทั้งแบบโรลปกติ และแบบโรลโครงร่างตาข่าย ใช้Type เป็นแอทริบิ้วในการอ้างอิงเพื่อใช้งาน

ตารางที่ 6.1 (ต่อ)

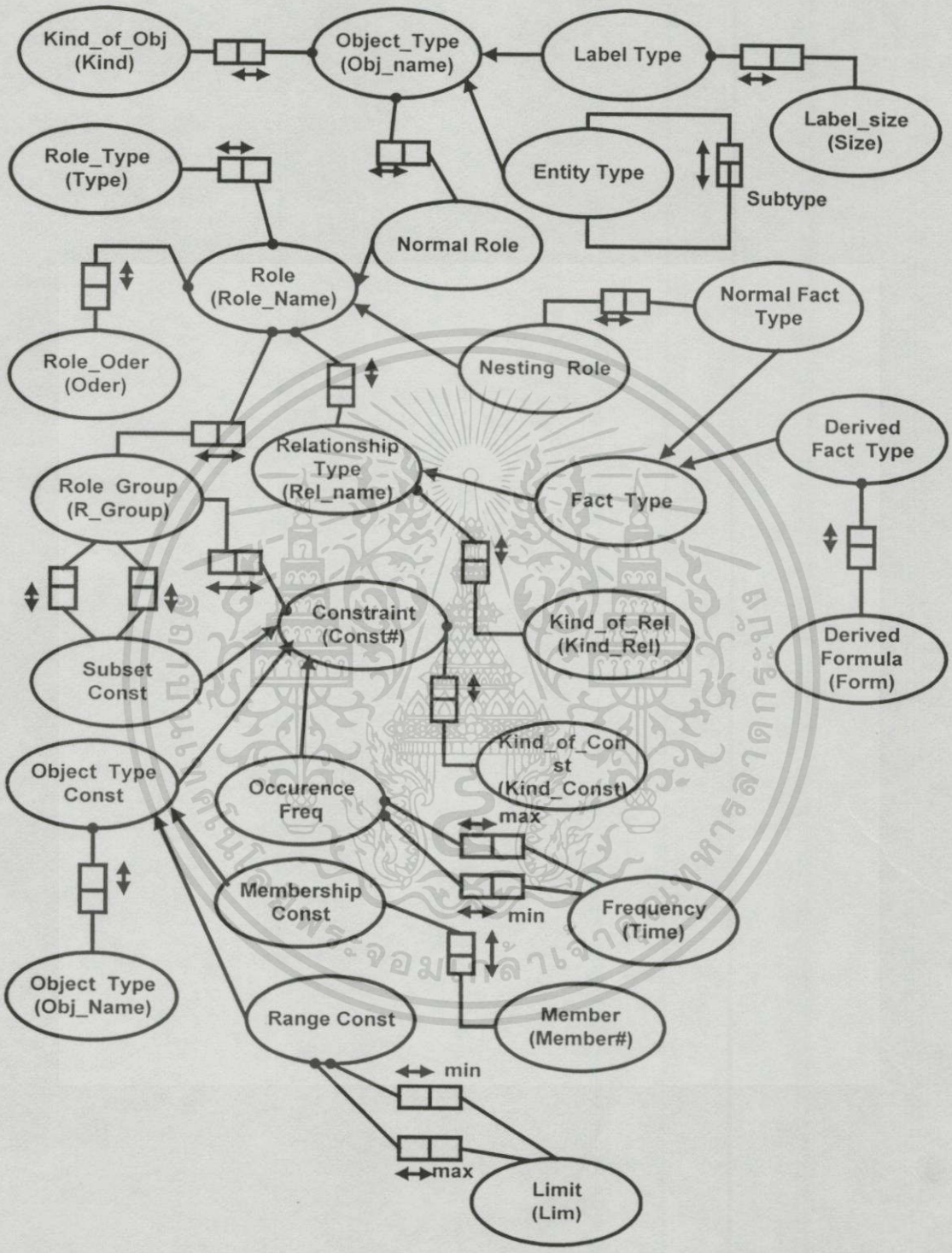
Role_ Order	เป็นลำดับของโรลในแต่ละชนิดความจริงและชนิงอ้างอิง ใช้ Order เป็นแอทริบิ้วในการอ้างอิงเพื่อใช้งาน
Role_Group	เป็นกลุ่มของโรล ใช้กับโรลที่มีความสัมพันธ์กันเป็นแบบกลุ่มใช้ R_Group เป็นแอทริบิ้วในการอ้างอิงเพื่อใช้งาน
Relationship Type	เป็นประเภทของความสัมพันธ์ซึ่งมีอยู่สองชนิดคือชนิดความจริง และชนิดอ้างอิง ใช้ Rel_Name เป็นแอทริบิ้วในการอ้างอิงเพื่อใช้งาน
Range Const	เป็นคอนสเตรนท์แบบเรนจ์ที่ใช้ Const# เป็นแอทริบิ้วในการอ้างอิงเพื่อใช้งาน
Subset Const	เป็นคอนสเตรนท์แบบสับเซ็ตที่ใช้ Const# เป็นแอทริบิ้วในการอ้างอิงเพื่อใช้งาน

6.4 หน่วยการสร้างรีเลชั่นสกีมม่า

เนื่องจากงานวิจัยในครั้งนี้ได้ทำการจัดเก็บกฎข้อบังของข้อมูลลงในตารางฐานข้อมูลเมทต้า ซึ่งฐานข้อมูลเมทต้าที่ได้นี้ก็ได้มาจากการทำการแมปจากแบบจำลองเชิงแนวคิดไนแอม โดยการแมปในแต่ละ conceptual schema แปลงไปเป็น relational schema นั้น อาจจะแปลงได้หลายรูปแบบ แต่งานวิจัยนี้ได้ยึดแนวทางของอัลกอริธึม ONF มาช่วยในการทำงานในครั้งนี้ เนื่องจากการทำงานของอัลกอริธึม ONF สร้างขึ้นมาเพื่อให้ใช้งานได้ง่าย ๆ สำหรับหลักการทำงานได้อธิบายไว้แล้วในบทที่ 4

6.5 หน่วยการสร้าง Data Structure ของฐานข้อมูลเชิงวัตถุ

การทำงานในส่วนนี้เป็นส่วนสุดท้ายของการทำงานวิจัยในครั้งนี้ ซึ่งเป็นการกำหนดภาษานิยามของฐานเชิงวัตถุ (Object Definition Language หรือเรียกย่อ ๆ ว่า ODL) หลักการทำงานโดยทั่ว ๆ ไปก็คือผู้วิจัยได้ทำการออกแบบอัลกอริธึมสำหรับทำการแมปจาก Relational Schema ให้อยู่ในรูปของ Data Structure ของฐานข้อมูลเชิงวัตถุ โดยอาศัยความสัมพันธ์ที่เกิดขึ้นของ Primary keys และ Foreign keys ใน Relational Schema



รูปที่ 6.2 แสดง A NIAM Meta Conceptual Schema

ผู้วิจัยได้ออกแบบแนวคิดในการทำแมปปิ้งจาก Relational Schema ให้อยู่ในรูปของ Data Structure ของฐานข้อมูลเชิงวัตถุ สามารถแบ่งออกเป็น 4 ขั้นตอนดังต่อไปนี้

ขั้นตอนที่ 1 : การสร้างโครงสร้างเชิงวัตถุแบบเดี่ยว ๆ

1.ข้อมูลของเอนติตี้ใด ๆ สามารถแปลงเป็นโครงสร้างเชิงวัตถุเดี่ยว ๆ ได้โดยการกำหนดให้ชื่อของรีเลชั่นกำหนดให้เป็นชื่อของออปเจกต์คลาส แล้วให้เอาแอทริบิ้วในรีเลชั่นสกีมม่ากำหนดให้เป็นแอทริบิ้วในโครงสร้างเชิงวัตถุโดยตรง

ตัวอย่าง.

Person(NAME, CODE_GEN, PHNR)

ขั้นตอนที่ 2: การสร้างโครงสร้างเชิงวัตถุแบบ inheritance

1. ข้อมูลใด ๆ ที่มีชนิดเอนติตี้ตั้งแต่ 3 ชนิดเอนติตี้ขึ้นไปโดยที่มีคีย์หลัก (primary key) เป็นชนิดเดียวกัน และแอทริบิ้วที่มีอยู่ในแต่ละแอทริบิ้วไม่ซ้ำกัน ให้ถือว่ากลุ่มของชนิดเอนติตี้เหล่านี้ทำงานสัมพันธ์กันในลักษณะออปเจกต์แบบ inheritance

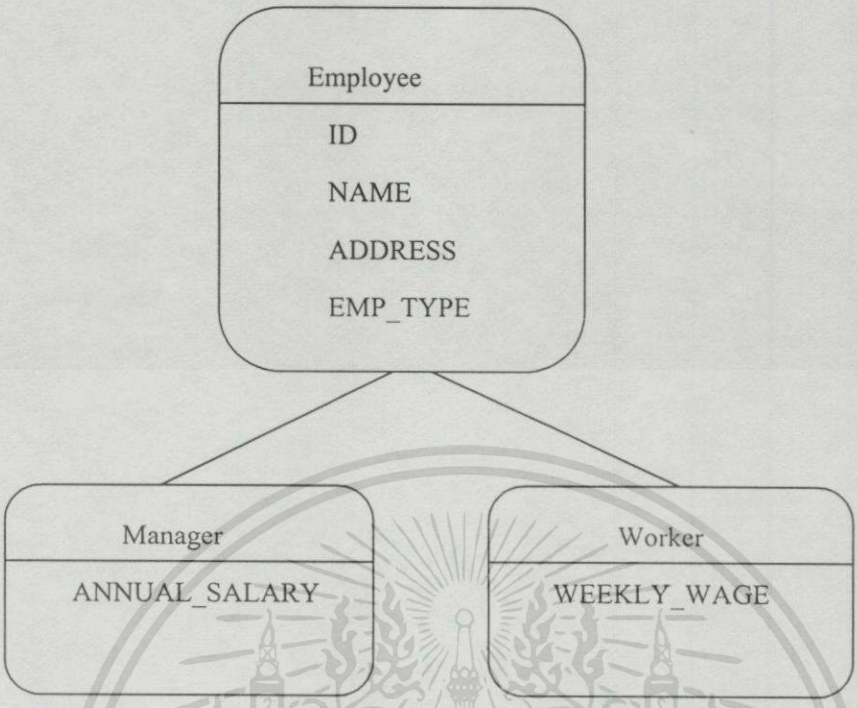
2. รูปแบบออปเจกต์ที่ได้ประกอบด้วย superclass ที่มีคีย์หลัก และแอทริบิ้วสำหรับใน subclass จะลดรูปของคีย์หลักออกไปเหลือเพียงแอทริบิ้วเท่านั้น

ตัวอย่าง.

Employee(ID, NAME, ADDRESS, EMP_TYPE)

Manager(ID, ANNUAL_SALARY)

Worker(ID, WEEKLY_WAGE)



ขั้นตอนที่ 3 : การสร้างโครงสร้างเชิงวัตถุแบบ non-hierarchical relationship

1. ข้อมูลชนิดเอนติตี้ใด ๆ ที่มีความสัมพันธ์กันในลักษณะ 3 ชนิดเอนติตี้ที่มีคีย์หลักแตกต่างกัน แต่มีคีย์แอทริบิ้วของเอนติตี้ที่เหลือเป็นตัวเดียวกันกับคีย์หลักของทั้ง 2 ชนิดเอนติตี้ที่ได้กล่าวไว้ข้างต้นจะถือว่ากลุ่มของชนิดเอนติตี้ทั้ง 3 นี้มีรูปแบบออปเจกต์ลักษณะ non-hierarchical relationship ซึ่งออปเจกต์ที่ได้จะลดรูปของชนิดเอนติตี้ที่มีคีย์แอทริบิ้วซ้ำกับเอนติตี้ทั้ง 2 ออกไป

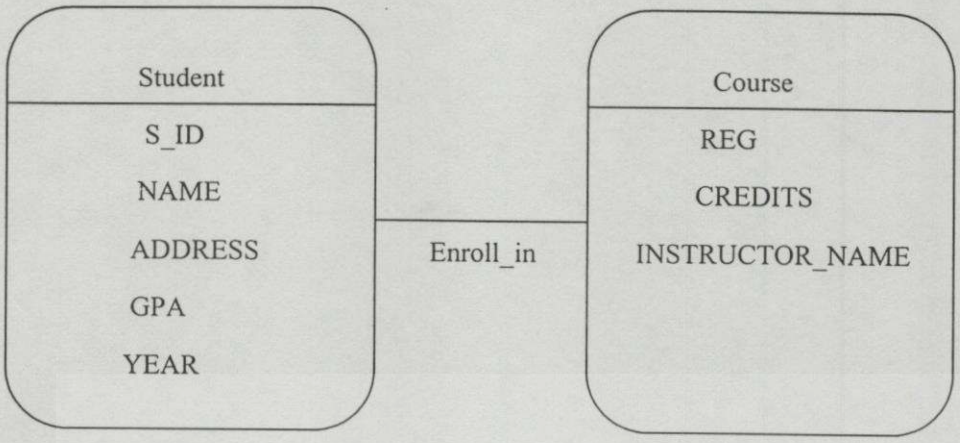
หมายเหตุ: หากคีย์แอทริบิ้วของชนิดเอนติตี้ที่เหลือเป็นคีย์หลักทั้งหมดจะถือว่าความสัมพันธ์ของทั้ง 3 ชนิดเอนติตี้ทำให้เกิดออปเจกต์แบบ many to many non-hierarchical relationship แต่หากมีเพียงแอทริบิ้วเดียวเป็นคีย์หลักก็จะถือว่าความสัมพันธ์ของทั้ง 3 ชนิดเอนติตี้ทำให้เกิดออปเจกต์แบบ one to many non-hierarchical relationship

ตัวอย่าง .

Student(S_ID, NAME, ADDRESS, GPA, YEAR)

Course(REG, CREDITS, INSTRUCTOR_NAME)

Enroll_in(S_ID, REG)



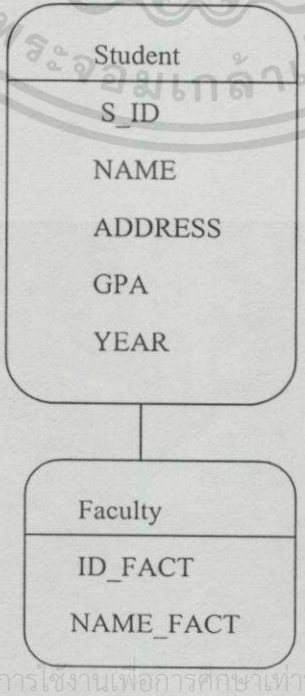
ขั้นตอนที่ 4 การสร้างโครงสร้างเชิงวัตถุแบบ aggregation

1. ข้อมูลที่มี 2 ชนิดเอนติตี้ใด ๆ โดยที่ในแต่ละชนิดเอนติตี้มีคีย์หลักแตกต่างกัน แต่มีบางแอทติบิ้วของชนิดเอนติตี้นั้น ๆ ประกอบรวมอยู่ในชนิดเอนติตี้ที่กล่าวข้างต้น จะทำให้เกิดความสัมพันธ์ของทั้ง 2 ชนิดเอนติตี้นี้จะทำให้เกิดความสัมพันธ์ในรูปแบบออปเจกต์ในลักษณะ aggregation โดยที่จะมีออปเจกต์หนึ่งออปเจกต์ใด ทำการลดรูปแอทริบิ้วที่ซ้ำซ้อนกันออกไปตามความเหมาะสมที่จะเกิดขึ้นในออปเจกต์นั้น ๆ

ตัวอย่าง.

Student(S_ID, NAME, ADDRESS, GPA, YEAR)

Faculty(ID_FACT, NAME_FACT, S_ID)



บทที่ 7

การพัฒนาภาษา SL/NIAM และเครื่องมือสำหรับออกแบบฐานข้อมูลเชิงวัตถุและฐานข้อมูลเชิงสัมพันธ์

งานวิจัยที่เกิดขึ้นจากการทำวิจัยในครั้งนี้สามารถที่จะทำการอธิบายตามขั้นตอนต่างๆ ได้ว่าแต่ละขั้นตอนที่ได้กระทำนั้นมาวิธีการในการกระทำอย่างไรบ้าง มีการนำหลักทางทฤษฎีอะไรบ้างมาเกี่ยวข้อง มีการพัฒนาหลักการใหม่ ๆ ไหนบ้างที่ได้วิจัยขึ้นมา สำหรับขั้นตอนทั้งหมดที่จะอธิบายจะเริ่มต้นจากการสร้างภาษา SL/NIAM จนกระทั่งสิ้นสุดของการทำวิจัยโดยได้ผลงานวิจัยบรรลุตามเป้าหมายนั่นก็คือ ระบบสามารถทำการพัฒนาจนได้ ODL ซึ่งเป็นการนิยามฐานข้อมูลเชิงวัตถุ ในงานวิจัยครั้งนี้ได้มีการพัฒนาขึ้นมาได้อาศัยการทำงานของอุกรณ์ต่าง ๆ ซึ่งมีทั้งอุปกรณ์ในส่วนของฮาร์ดแวร์ และในส่วนอุปกรณ์ทางซอฟต์แวร์ โดยผู้ทำการวิจัยพยายามที่จะเลือกนำเอาอุปกรณ์ที่เหมาะสมมาใช้ในการวิจัยในครั้งนี้ เพื่อให้การทำวิจัยบรรลุตามวัตถุประสงค์ตามเป้าหมายที่วางไว้ให้มากที่สุด

7.1 อุปกรณ์ทางฮาร์ดแวร์

- เครื่องไมโครคอมพิวเตอร์รุ่นเพนเทียม MMX 200
- ฮาร์ดิสก์ขนาด 3.2 GB. และ 2.1 GB.
- หน่วยความจำขนาด 32 MB.
- จอภาพแบบ Super VGA (1024 X 768) 256 สี แป้นพิมพ์ 101 คีย์
- ฟลอปปี้ดิสก์ไดร์ขนาด 1.44 1 ตัว
- เครื่องพิมพ์ Cannon รุ่น BJC-265SP Color Bubble Jet Printer

7.2 อุปกรณ์ทางซอฟต์แวร์

ในการทำวิจัยครั้งนี้ได้เกี่ยวข้องกับอุปกรณ์ทางซอฟต์แวร์ต่างๆ ซึ่งสามารถจะระบุชื่อและขออธิบายรูปแบบของการทำงานของซอฟต์แวร์ในบางชนิดพอสังเขปได้ดังต่อไปนี้

- โปรแกรมภาษา yacc/lex
- โปรแกรมภาษา C, C++
- โปรมแกรมภาษา Pro* C
- โปรแกรมจัดการฐานข้อมูล Oracle บนลีนุกซ์
- โปรแกรมระบบปฏิบัติการ Linux

7.3 ขั้นตอนและวิธีการพัฒนางานวิจัย

7.3.1 ขั้นตอนและวิธีการพัฒนารูปแบบภาษา

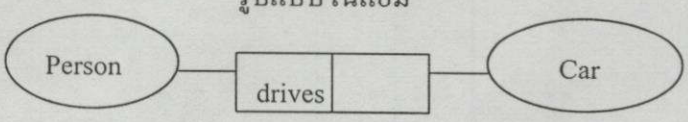
ขั้นตอนแรกในการพัฒนาภาษา SL/NIAM เริ่มจากการศึกษาทฤษฎีทางหลักไวยากรณ์ภาษา และการเลือกใช้โปรแกรมภาษาสำหรับพัฒนาตัวแปลภาษาขึ้นมา งานวิจัยในครั้งนี้ได้ทำการพิจาณาที่จะเลือกโปรแกรมภาษา yacc/lex เป็นตัวพัฒนาตัวแปลภาษา SL/NIAM สำหรับเหตุผลที่ได้ทำการพิจารณานั้นเนื่องจาก โปรแกรมภาษาดังกล่าวนี้เหมาะสมในการพัฒนาตัวแปลภาษาเนื่องจากมีความสะดวกและไม่ต้องเขียนโปรแกรมเป็นจำนวนมาก และนอกจากนี้ภาษา SL/NIAM ที่ได้ทำการพัฒนานี้เป็นเพียงภาษากึ่งธรรมชาติ ซึ่งเป็นไปตามหลักทางทฤษฎีที่หลักไวยากรณ์ภาษาในรูปแบบ context free grammars ซึ่งเป็นประเภทของหลักไวยากรณ์ที่โปรแกรม yacc/lex ทำงานอยู่เช่นกัน สำหรับรูปแบบของภาษาที่สร้างขึ้นนี้ได้ใช้แนวคิดมาจากพื้นฐานของกฎข้อบังคับเชิงไนแอมมาปฏิบัติโดยจะเป็นรูปแบบภาษาในลักษณะชนิดความจริงแบบไบนารี (binary fact type) ซึ่งเป็นการอธิบายของความสัมพันธ์ระหว่างชนิดเอนติตี้ที่มีโรลคั่นระหว่างชนิดเอนติตี้ทั้งสอง เราจะกำหนดให้ชนิดเอนติตี้ที่กล่าวขึ้นก่อนเป็น ประธานในประโยค โดยมีโรลเป็นกริยาของประโยค และมีกรรมในประโยค เทนชนิดเอนติตี้สุดท้ายที่สัมพันธ์กัน เนื่องจากภาษาที่พัฒนาขึ้นนี้จะแยกออกเป็นคำ ๆ ดังนั้นในกรณีของโรลผู้วิจัยได้อนุโลมให้ทำการเชื่อมต่อคำเพื่อสื่อความหมายโดยการนำเอาคำมาต่อกันโดยมีเครื่องหมาย "_" เป็นตัวเชื่อม

สำหรับรูปประโยคที่แสดงในลักษณะ fact type จะมีรูปแบบของหลักไวยากรณ์ที่ใช้ในภาษา SL/NIAM เป็นดังนี้คือ entity_type role entity_type|entity_type role object โดยการกำหนดให้รูปประโยคที่เราเขียนขึ้นนี้เป็นรูปแบบของประโยคในลักษณะชนิดความจริง โดยเราได้นำเอารูปแบบของไนแอมมาพัฒนา จากประโยคภาษา SL/NIAM เราเขียนรูปประโยคโดยให้เอนติตี้เขียนขึ้นต้นด้วยอักษรตัวใหญ่แล้วตามด้วยอักษรตัวเล็ก และโรลเราจะกำหนดเป็นตัวอักษรตัวเล็กทั้งหมด

รูปแบบของภาษา SL/NIAM

Person drives Car

รูปแบบไนแอม



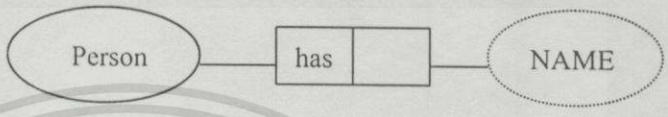
สำหรับรูปประโยคที่แสดงในลักษณะ reference type จะมีรูปแบบของหลักไวยากรณ์ที่ใช้ในภาษา SL/NIAM เป็นดังนี้คือ entity_type role label_type โดยการกำหนดให้รูปประโยคที่เราเขียนขึ้นนี้เป็นรูปแบบของประโยคในลักษณะชนิดอ้างอิง โดยเราได้นำเอารูปแบบของไนแอมมา

พัฒนา จากประโยคภาษา SL/NIAM เราเขียนรูปประโยคโดยให้เอนติตี้เขียนขึ้นต้นด้วยอักษรตัวใหญ่แล้วตามด้วยอักษรตัวเล็ก และโรลเราจะกำหนดเป็นตัวอักษรตัวเล็กทั้งหมด และกรรมของประโยคเราได้กำหนดให้เป็นตัวอักษรใหญ่ทั้งหมดเนื่องจากชนิดเอนติตี้สุดท้ายเป็นลักษณะอ้างอิงเราเรียกว่าเลเบิ้ลซึ่งแสดงเป็นเส้นประ

รูปแบบประโยคภาษา SL/NIAM

Person has NAME .

รูปแบบไนแอม

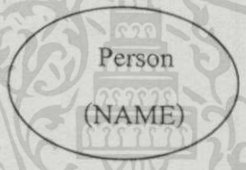


หรืออาจจะแสดงได้อีกรูปแบบหนึ่งของลักษณะชนิดอ้างอิงดังนี้คือ

รูปแบบประโยคภาษา SL/NIAM

Person(NAME).

รูปแบบไนแอม



วิธีการเขียนรูปประโยคในกรณีนี้เราจะให้เอนติตี้ขึ้นต้นด้วยอักษรตัวใหญ่แล้วตามด้วยวงเล็บโดยที่กำหนดให้ชนิดเอนติตี้ที่อ้างอิงที่เรียกว่าเลเบิ้ลให้เป็นอักษรตัวใหญ่ทั้งหมดที่อยู่ภายในวงเล็บ

สำหรับรูปประโยคที่แสดงในลักษณะ lexical จะมีรูปแบบของหลักไวยากรณ์ที่ใช้ในภาษา SL/NIAM เป็นดังนี้คือ label_type datatype size โดยการกำหนดให้รูปประโยคที่แสดงถึงการกำหนดชนิดของข้อมูลและขนาดของข้อมูล (Identifier)ให้กับชนิดเอนติตี้หรือชนิดเลเบิ้ลก็ตามในรูปประโยคนั้นมีวิธีการเขียนดังนี้คือ เริ่มจากให้เขียนอักษรตัวใหญ่นำหน้าคำถ้าเป็นชนิดเอนติตี้หรือถ้าเป็นชนิดเลเบิ้ลให้เป็นตัวใหญ่ทั้งหมด หลังจากนั้นก็เว้นวรรคแล้วตามด้วยชนิดข้องข้อมูลซึ่งอาจจะเป็นข้อมูลชนิดต่าง ๆ ไม่ว่าจะเป็น number หรือ character (เขียนสั้น ๆ ในประโยคเป็น char) หรืออื่น ๆ โดยตัวอักษรของชนิดข้อมูลจะกำหนดให้ใช้เป็นอักษรตัวเล็กทั้งหมด แล้วตามด้วยวงเล็บซึ่งภายในวงเล็บก็เป็นขนาดของข้อมูลที่เหมาะสม

รูปแบบประโยคภาษา SL/NIAM

NAME char(25).

สำหรับรูปประโยคที่แสดงในลักษณะ unique สามารถเขียนได้โดยในรูปของหลักไวยากรณ์จะแสดงเป็น one entity_type role quantity object โดยที่ quantity สามารถแสดงได้สองรูปแบบคือ เป็น one หรือ เป็น many และสำหรับ object สามารถแสดงได้ว่าเป็น entity_type หรือ label_type ตามความเหมาะสม กรณีความสัมพันธ์ของประโยคจะเป็นแบบ one to one ก็ต่อเมื่อประโยคมี quantity เป็น one และประโยคที่สองซึ่งเป็นรูปประโยคย้อนกลับมี quantity เป็น one เช่นเดียวกัน แต่ตัวโรลผู้วิจัยจะกำหนดไว้ว่าหากในรูปประโยคแรกมีโรลในเป็นคำต่อไปนี้คือ "has , have, in, is, with, for, take" จะมีผลต่อโรลในที่ประโยคย้อนกลับนั่นคือจะเป็น "of และนอกจากคำที่กล่าวมาจะให้โรลในประโยคย้อนกลับเพิ่มคำว่า by ลงไปด้วย" เป็นต้น ดังนั้นแสดงว่าความสัมพันธ์ของชนิดเอนติตี้กับชนิดเอนติตี้หรือกับชนิดเลเบิ้ลนั้นเป็นแบบ one to one แต่ถ้าประโยคแรกมี quantity เป็น one และในประโยคที่สองซึ่งเป็นรูปประโยคย้อนกลับมี quantity ในประโยคเป็น many นั่นแสดงว่ารูปประโยคแสดงความสัมพันธ์แบบ many to one สำหรับกรณีที่ในประโยคแรกมี quantity เป็น many และมี quantity ในประโยคที่สองที่เป็นประโยคย้อนกลับเป็น many นั่นแสดงว่ารูปประโยคแสดงความสัมพันธ์แบบ many to many

รูปประโยค SL/NIAM

one Person is_prime_minister one Country.

one Country is_prime_minister_by one Person.

รูปแบบไนแอม

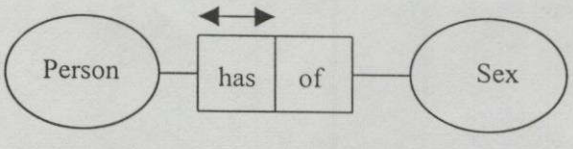
จากรูปประโยคที่แสดงข้างบนนี้ เป็นการแสดงความสัมพันธ์ของประโยคในรูปแบบ one to one ของ Person กับ Country นั่นก็คือในความหมายจากประโยคแสดงได้ว่า Person คนหนึ่ง ๆ จะเป็นนายกรัฐมนตรีได้เพียงประเทศเดียวเท่านั้น และในรูปประโยคย้อนกลับจะแสดงถึงว่าในในประเทศหนึ่ง ๆ จะมีนายกรัฐมนตรีได้เพียงคนเดียวเท่านั้น

รูปประโยค SL/NIAM

one Person has one Sex.

one Sex of many Person.

รูปแบบไนแอม



จากรูปประโยคที่แสดงข้างบนนี้ เป็นการแสดงความสัมพันธ์ของประโยคในรูปแบบ many to one ของ Person กับ Sex นั่นก็คือในความหมายจากประโยคแสดงได้ว่า Person คน

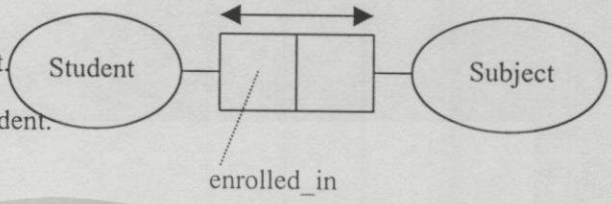
หนึ่ง ๆ จะมีเพศได้เพียงเพศเดียวเท่านั้น และในรูปประโยคย้อนกลับจะแสดงถึงว่าในเพศเดียวกันจะมีได้ในหลาย ๆ Person

รูปประโยค SL/NIAM

one Student enrolled_in many Subject.

one Subject enrolled_in_by many Student.

รูปแบบไนแอม



จากรูปประโยคที่แสดงข้างบนนี้ เป็นการแสดงความสัมพันธ์ของประโยคในรูปแบบ many to many ของ Student กับ Subject นั่นก็คือในความหมายจากประโยคแสดงได้ว่า Student คนหนึ่ง ๆ จะมีลงทะเบียนวิชาได้หลาย ๆ วิชา และในรูปประโยคย้อนกลับจะแสดงถึงว่าในวิชาหนึ่ง ๆ จะถูกลงทะเบียนจากนักศึกษาได้หลาย ๆ คน

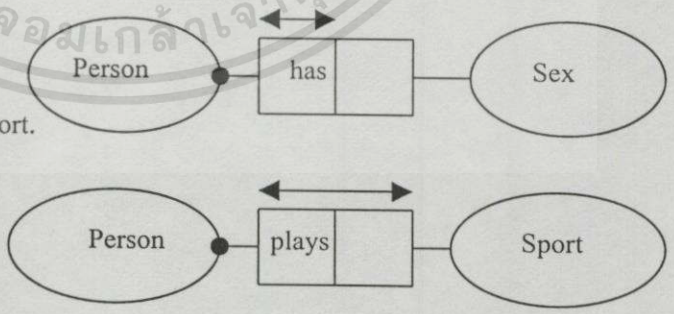
สำหรับรูปประโยคที่แสดงในลักษณะ mandatory จะมีรูปแบบของหลักไวยากรณ์ที่ใช้ในภาษา SL/NIAM เป็นดังนี้คือ every one entity_type role quantity object โดยการกำหนดให้ every และ one เป็นคีย์เวิร์ดในรูปประโยค ซึ่งจริง ๆ แล้วรูปแบบประโยคนี้เป็นการเสริมรายละเอียดที่เกิดขึ้นกับประโยคในลักษณะของ unique นั่นเอง โดยเพียงเพิ่มเติมคำว่า every ซึ่งแสดงถึงว่าเอนติตี้นั้นจะต้องมีความสัมพันธ์กับ object เสมอ ดังจะแสดงในรูปประโยคข้างล่างต่อไปนี้

รูปแบบประโยค

every one Person has one Sex.

every one Person plays many Sport.

รูปแบบไนแอม



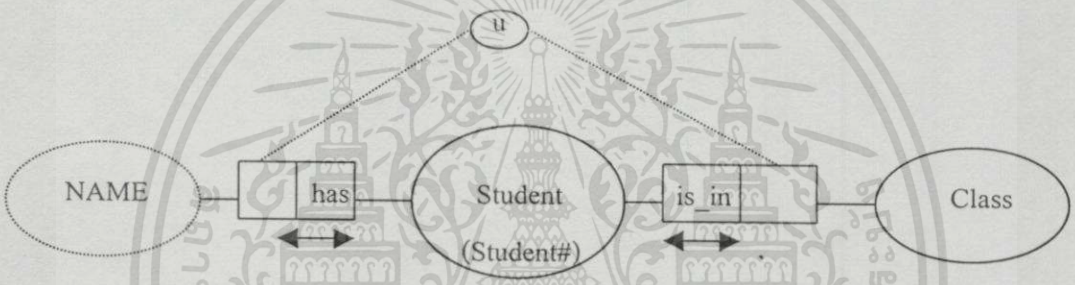
จากรูปประโยคแรกข้างบนนี้แสดงให้ทราบว่าทุก ๆ Person จะมีเพศเสมอและก็มีเพียงเพศเดียวเสมอ และในรูปประโยคที่สองจะแสดงให้ทราบว่าทุก ๆ Person จะมีการเล่นกีฬาเสมอและก็เล่นหลาย ๆ ชนิดด้วย

สำหรับรูปประโยคที่แสดงในลักษณะของ Inter fact type Unique เราสามารถแสดงหลักไวยากรณ์ของรูปประโยคได้ดังนี้คือ entity_type combination_of object and object is_unique จากหลักไวยากรณ์เราจะกำหนดให้ combination, and และ is_unique เป็นคีย์เวิร์ดในประโยค โดย combination และ and เป็นการรวมกันของ fact type ทั้งสองว่ามีความสัมพันธ์กันกับ entity_type ในชนิดเดียวกันไม่ซ้ำซ้อน เราสามารถอธิบายให้เห็นภาพชัดเจนได้ดังรูปประโยคที่สอดคล้องกับหลักไวยากรณ์ที่กำหนดดังข้างล่างต่อไปนี้

รูปแบบภาษา SL/NIAM

Student combination_of NAME and Class is_unique

รูปแบบไนแอม



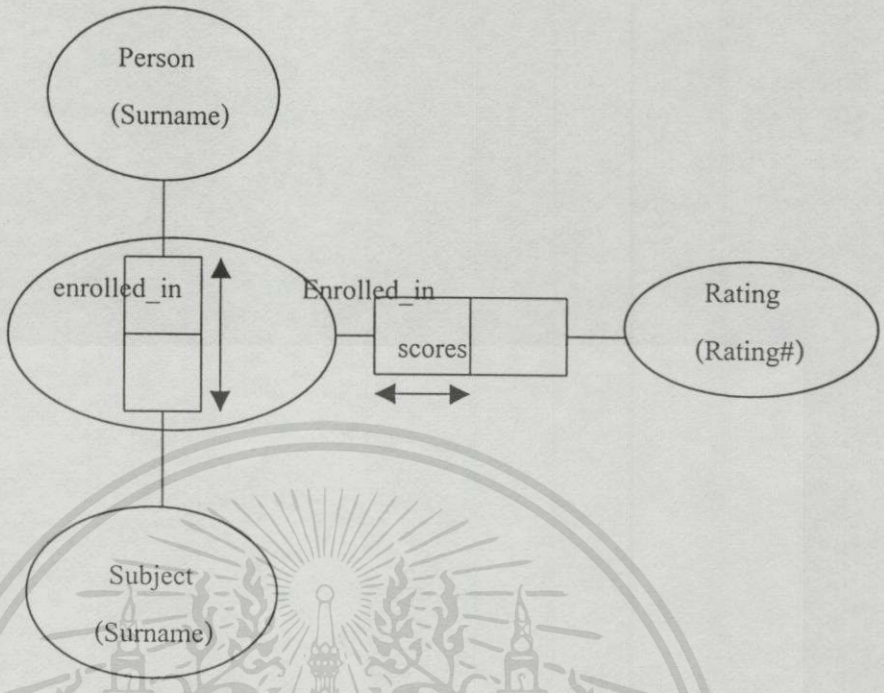
จากรูปประโยคข้างบนแสดงถึงการรวม fact type ที่เกิดขึ้นจากประโยค Student has NAME กับรูปประโยคของ Student is_in Class เป็นประโยคเดียวกันเนื่องจากเอาท์พุทของประโยคในแต่ละแถวจะมีลักษณะ unique กับ Student เดียวกัน

สำหรับรูปประโยคที่แสดงในลักษณะ Nest fact type สามารถแสดงด้วยหลักไวยากรณ์ทางภาษา SL/NIAM ได้ดังนี้คือ entity_type role object role quantity object จากรูปแบบของหลักไวยากรณ์นี้แสดงให้ทราบว่าเป็นการรวมชนิดเอนติตี้กับออปเจกต์แรกเกิดเป็นชนิดเอนติตี้ใหม่ขึ้นมาแล้วนำชนิดเอนติตี้ที่เกิดขึ้นใหม่นี้ไปมีความสัมพันธ์ในรูปแบบใด ๆ กับออปเจกต์ที่สองโดยอาจจะเป็นแบบ one to one หรือ แบบ many to many หรือ many to one ตามความต้องการ ซึ่งเราจะอธิบายให้เกิดความชัดเจนได้ดังรูปแบบที่แสดงดังข้างล่างต่อไปนี้

รูปแบบภาษา SL/NIAM

Person enrolled_in Subject scores one Rating

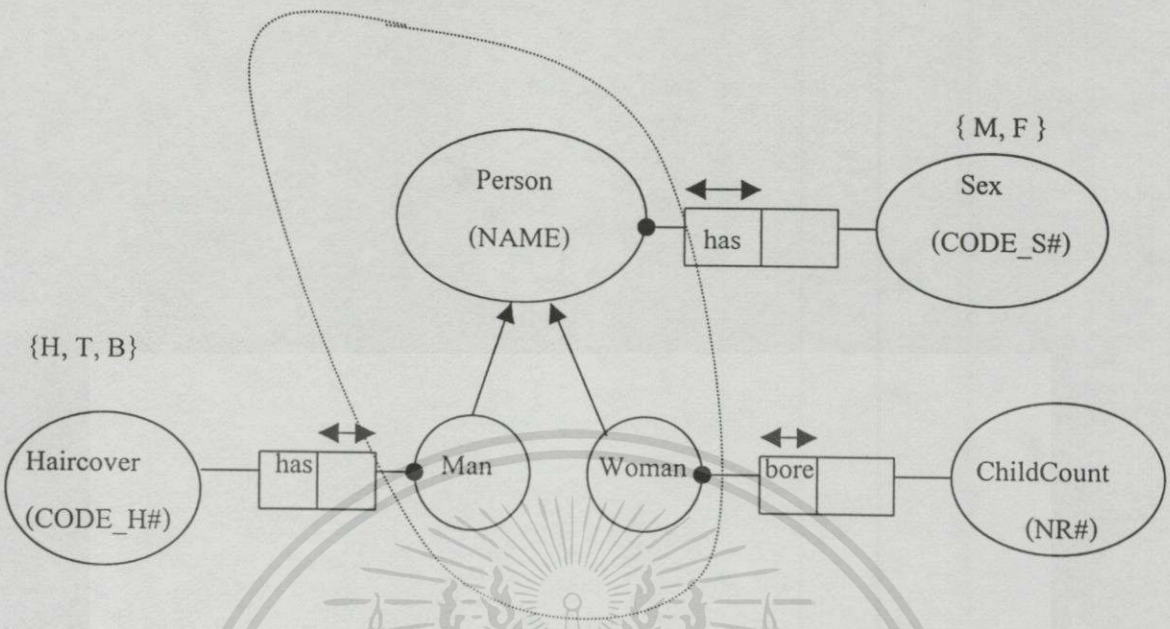
รูปแบบไนแอม



จากรูปประโยคที่แสดงดังข้างบนสามารถอธิบายได้ว่า fact type ของประโยค Person enrolled_in Subject ถูกกำหนดให้เป็นชนิดเอนติตี้ใหม่ขึ้นมาโดยผู้วิจัยจะกำหนดชื่อของชนิดเอนติตี้ใหม่ขึ้นมาโดยสร้างมาจากโรลของ fact type แรกที่เกิดขึ้นนั่นคือเป็นชนิดเอนติตี้ Enrolled_in เพื่อแสดงความสัมพันธ์กับในลักษณะ fact type ของรูปประโยคกับชนิดเอนติตี้ที่เหลือคือชนิดเอนติตี้ Rating

สำหรับรูปประโยคที่แสดงในลักษณะ subtype เราสามารถแสดงหลักไวยากรณ์ที่นำมาใช้กับรูปแบบนี้ได้คือ entity_type=entity_type role entity_type label_type โดยทั่วๆ ไปแล้วความสัมพันธ์ในแต่ละชนิดเอนติตี้มักจะมีความสัมพันธ์กันในลักษณะอื่น ๆ ด้วยไม่ว่าจะเป็นในลักษณะของการเป็น unique หรือการเป็น mandatory ก็ตาม เพราะการพัฒนาระบบหนึ่ง ๆ จะมีการเชื่อมโยงความสัมพันธ์กันขนาดใหญ่มาก ๆ แต่กรณีนี้จะแยกมาแสดงเฉพาะ ส่วน ๆ ของลักษณะประโยคให้เห็นชัดเจน ถึงแม้รูปแบบของไนแอมบางส่วนที่แสดงควบคู่กับรูปประโยคภาษา SL/NIAM นั้นจะมีความสัมพันธ์ตามที่ได้กล่าวมีก่อนหน้านี้แล้วก็จะไม่เขียนร่วมกับประโยค SL/NIAM ในส่วนที่แยกเฉพาะ ๆ ขึ้นมา ขอได้โปรดทำความเข้าใจตามนี้ด้วย สำหรับรูปแบบภาษา SL/NIAM ของความสัมพันธ์ในกรณี subtype นั้นสามารถแสดงได้ดังนี้คือ

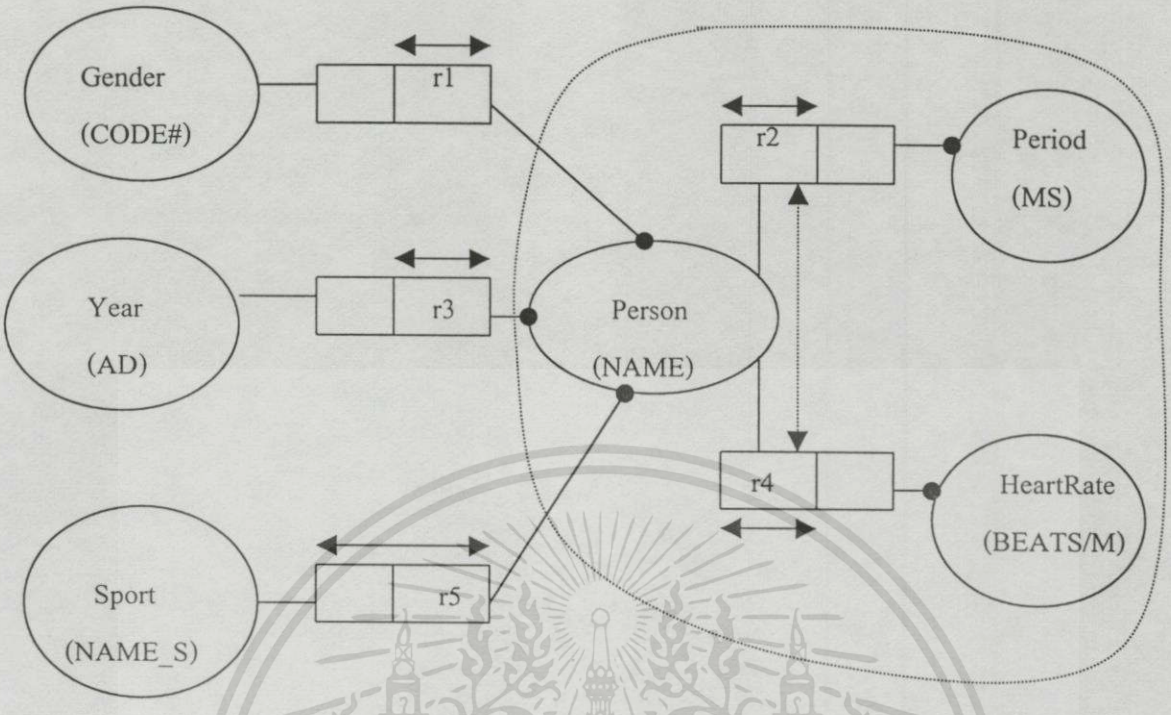
รูปแบบประโยคภาษา SL/NIAM	รูปแบบไนแอม
Man=Person of Sex 'M'	
Woman=Person of Sex 'F'	



จากรูปประโยคสามารถอธิบายได้ว่าผู้วิจัยได้กำหนดให้ชนิดเอนติตี้ตัวแรกในประโยคแสดงถึงการเป็น subtype ซึ่งในตัวอย่างนี้ก็คือ Man และ Woman และชนิดเอนติตี้ตัวที่สองในประโยคแสดงถึงการเป็น super type โดยในตัวอย่างของประโยคจะเป็น Person

รูปแบบของประโยคที่แสดงความสัมพันธ์ในลักษณะ equality สามารถแสดงหลักไวยากรณ์ในรูปแบบของภาษา SL/NIAM ได้ดังนี้คือ object role entity_type iff object role entity_type เราได้กำหนดคำว่า iff เป็นคีย์เวิร์ดก็เพื่อนำมาอธิบายแทนคำว่า "if and only if" ซึ่งความหมายก็คือ "ก็ต่อเมื่อ" ดังนั้นในรูปประโยคที่สอดคล้องกับหลักไวยากรณ์ที่เขียนขึ้นมานี้เพื่อสื่อว่า fact type แรกจะกระทำได้ก็ต่อเมื่อมีการกระทำใน fact type ที่สองเสร็จแล้ว เราสามารถอธิบายเพิ่มเติมให้ชัดเจนดังตัวอย่างของรูปประโยคข้างล่างต่อไปนี้คือ

รูปประโยคภาษา SL/NIAM	รูปแบบไนแอม
Person has_reaction_time Period iff Person has_resting_HeartRate HeartRate	



r1= has r2=has_reaction_time

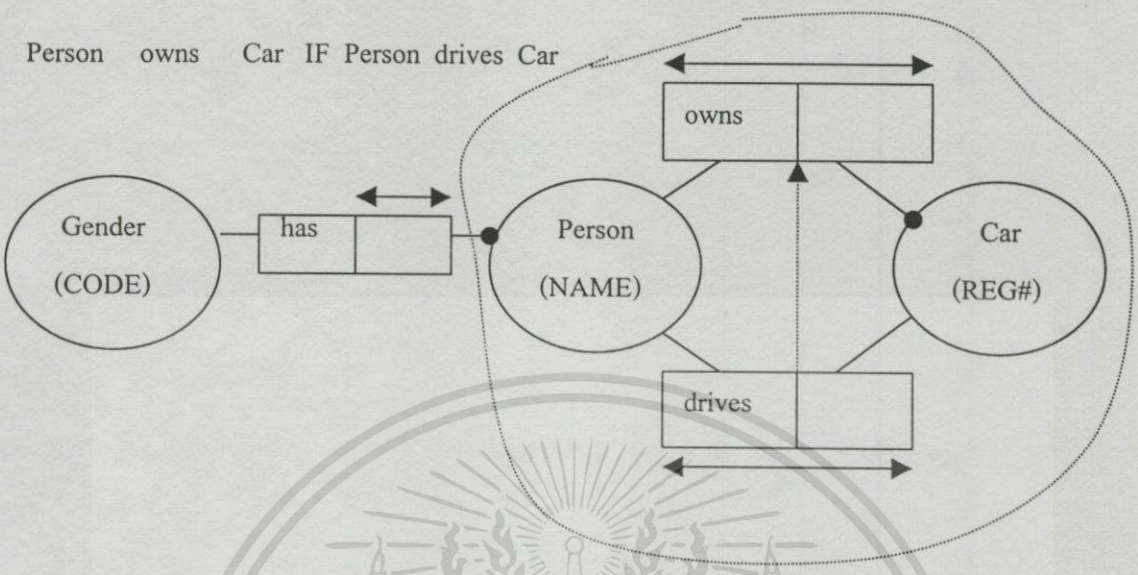
r3=was_born_in r4=has_resting_HeartRate

r5=plays

จากรูปประโยคเป็นการอธิบายเพื่อให้ทราบว่า การทำงานในส่วน fact type ของ Person กับ Period จะเกิดขึ้นได้ก็ต่อเมื่อมีการกระทำระหว่าง Person กับ HeartRate เสร็จสิ้นเรียบร้อยแล้วซึ่งการกระทำที่เกิดขึ้นทั้งหมดนี้เป็นการแสดงถึงความสัมพันธ์ของรูปประโยคในลักษณะ equality

สำหรับรูปแบบประโยคที่นำเสนอมานี้เป็นเพียงส่วนหนึ่งที่ผู้วิจัยได้ทำการพัฒนาขึ้นมา (ภาคผนวก ก) เพื่อสร้างเป็นหลักไวยากรณ์ของภาษา SL/NIAM โดยจะขอนำเสนอในหลักไวยากรณ์สุดท้ายในที่นี้ ซึ่งจะมีความสัมพันธ์กันในลักษณะของ subset รูปแบบของหลักไวยากรณ์นั่นก็คือ object role entity_type IF object role entity_type คล้าย ๆ กับการเชื่อมรูปประโยคจาก fact type ทั้งสองประโยคแต่มี IF เป็นตัวเชื่อมประโยคทั้งสองซึ่งเป็นการกำหนดให้สื่อถึงความหมายว่าผลการกระทำของ fact type ที่เกิดขึ้นทั้งหมดในประโยคที่สองจะอยู่ในผลของ fact type ของประโยคแรกทั้งหมดหรืออาจจะกล่าวอีกอย่างได้ว่าถ้าเหตุการณ์ที่เกิดขึ้นกับ fact type ที่สองแล้วจะต้องเกิดขึ้นกับ fact type ในประโยคแรกด้วยเช่นกัน กรณีย้อนกลับว่าเหตุการณ์ที่เกิดขึ้นกับ fact type แรกแล้วอาจจะไม่เกิดขึ้นกับเหตุการณ์ที่สองก็ได้เป็นต้น เราสามารถอธิบายให้ชัดเจนด้วยรูปประโยคที่ใช้ในภาษา SL/NIAM ดังที่จะได้กล่าวไว้ในข้างล่างต่อไปนี้คือ

รูปแบบประโยคภาษา SL/NIAM รูปแบบไนแอม



จากรูปประโยคแสดงให้ทราบว่าคนที่ขับรถทุกคนจะเป็นเจ้าของรถเสมอซึ่งแสดงให้เห็นว่าข้อมูลรายชื่อของผู้ที่ขับรถทั้งหมดนั้นจะมีรายชื่อเป็นเจ้าของรถทุกคน ซึ่งแสดงให้เห็นว่าข้อมูลของผู้ขับรถจะเป็นสับเซ็ตกับผู้ที่เป็นเจ้าของรถ ดังนั้นอาจจะกล่าวในลักษณะตรงข้ามได้อีกว่าผู้ที่เป็นเจ้าของรถไม่จำเป็นต้องเป็นคนขับเองเสมอก็ย่อมกล่าวได้เช่นกัน

7.3.2 ขั้นตอนและวิธีการตรวจสอบหลักไวยากรณ์ทางภาษา

จากที่ได้กล่าวมาทั้งหมดข้างต้นนั้นเป็นการออกแบบหลักไวยากรณ์รูปแบบภาษา SL/NIAM ที่จะนำมาใช้ในงานวิจัยเพื่อให้ได้ผลการวิจัยในขั้นสุดท้าย สำหรับในขั้นตอนต่อจากการออกแบบหลักไวยากรณ์ภาษา ผู้วิจัยได้ทำการเขียนโปรแกรมตัวแปลภาษาขึ้นโดยใช้หลักไวยากรณ์ภาษาที่ได้ทำการออกแบบไว้ทั้งหมด ซึ่งโปรแกรมภาษาที่ผู้วิจัยใช้นั้นได้นำเอาโปรแกรมภาษา yacc/lex ซึ่งทำงานทั้งในระบบปฏิบัติการยูนิกซ์และระบบปฏิบัติการลีนุกซ์ สำหรับผู้วิจัยได้เลือกใช้กับระบบปฏิบัติการลีนุกซ์เนื่องจากสามารถทำงานในเครื่องไมโครคอมพิวเตอร์แบบเครื่องพีซีที่ผู้วิจัยใช้อยู่

การพัฒนาตัวแปลภาษา SL/NIAM ในลักษณะโปรแกรมภาษา yacc/lex นำหลักไวยากรณ์ที่ได้ทำการออกแบบมาเขียนไว้ในส่วนถัดจากส่วนของโทเคน และทำการเขียนฟังก์ชั่นต่าง ๆ เพื่อติดต่อกับข้อมูลนำเข้านำมาเก็บไว้ในหน่วยความจำเพื่อจะนำมาตรวจสอบว่าจะเข้ากับหลักไวยากรณ์ตรงกับที่ได้สร้างหรือไม่ ตัวอย่างที่ได้เขียนจะยกมาบางส่วนในงานวิจัยก็คือ

กรณีหลักไวยากรณ์ที่เป็น |entity_type(label_type) ก็จะทำการกำหนดตัวแปรเป็นสองตัวเพื่อเก็บค่าที่เป็น Entity และค่าที่เป็น LABEL นั่นก็คือจะกำหนดตัวแปรเป็น u2_t_test_link และ u2_t_test_link1 ตามลำดับแล้วให้ $1และ $2 เป็นค่าที่ถูกฝังไว้ในหน่วยความจำส่งค่าให้กับตาม

แปรทั้งสองตามลำดับเช่นเดียวกัน ค่าที่ได้นี้จะถูกนำไปตรวจสอบว่าถูกหลักไวยากรณ์หรือไม่ หากถูกต้องก็จะนำข้อมูลจากตัวแปรที่รับค่าดังกล่าวไว้นี้ส่งต่อไปยังกระบวนการในส่วนของการจัดเก็บข้อมูลต่อไป แต่หากตรวจสอบแล้วไม่ถูกต้องก็จะยกเลิกการนำไปจัดเก็บในตัวแปรแล้วให้ทำการตรวจสอบข้อมูลนำเข้าต่อ ๆ ไป สำหรับการจองหน่วยความจำไว้ในกับข้อมูลในแต่ละตัวเพื่อจะนำมาตวรสอบนั้นก็เขียนฟังก์ชั่นการจองหน่วยความจำเช่นเดียวกัน

หลังจากที่ได้ทำการพัฒนาตัวแปลภาษา SL/NIAM ด้วยโปรแกรมภาษา yacc/lex เสร็จแล้วก็จะทำการคอมไพล์ตัวโปรแกรมโดยการใช้ชุดคำสั่งดังนี้คือ

$คำสั่ง lex ตามด้วยชื่อโปรแกรม และใส่นามสกุลด้วย

$lex tlexe.l

จะได้ lex.yy.c เป็นโปรแกรมภาษา C

$ คำสั่ง yacc ตามด้วยชื่อโปรแกรมและใส่นามสกุลด้วย

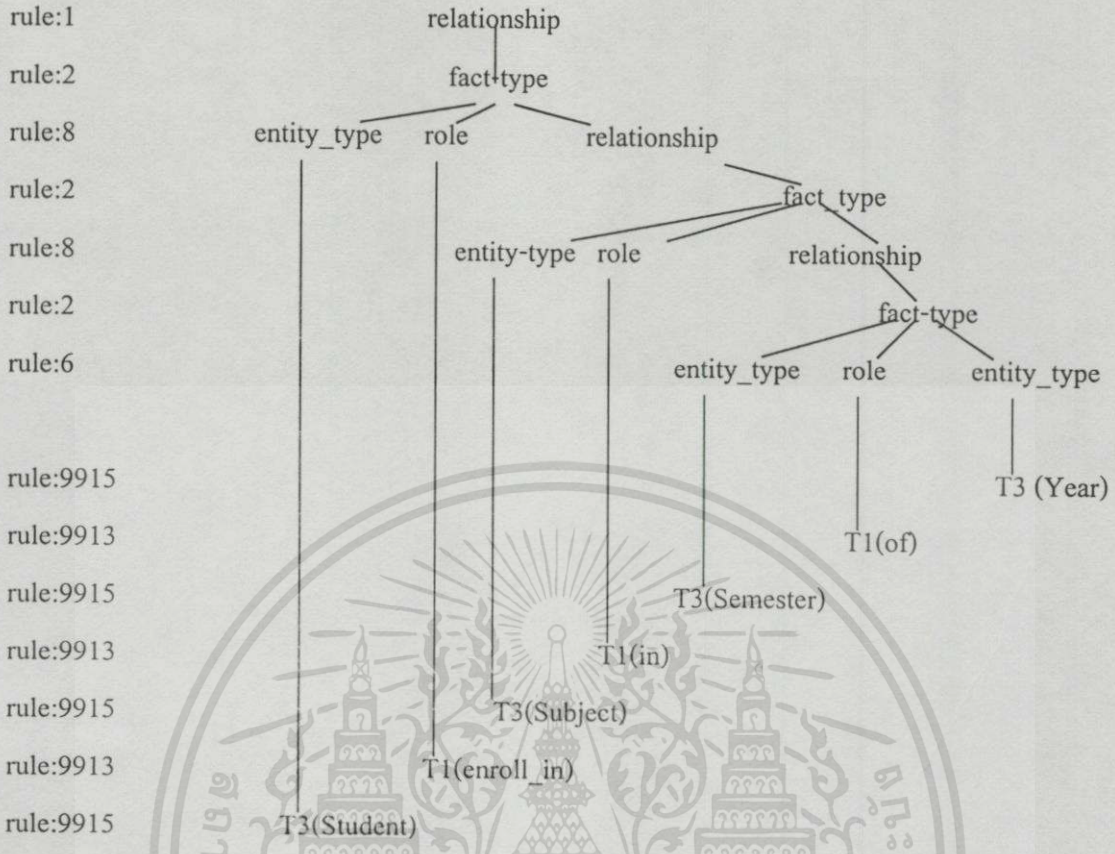
$yacc parser.y

จะได้ y.tab.c เป็นโปรแกรมภาษา C

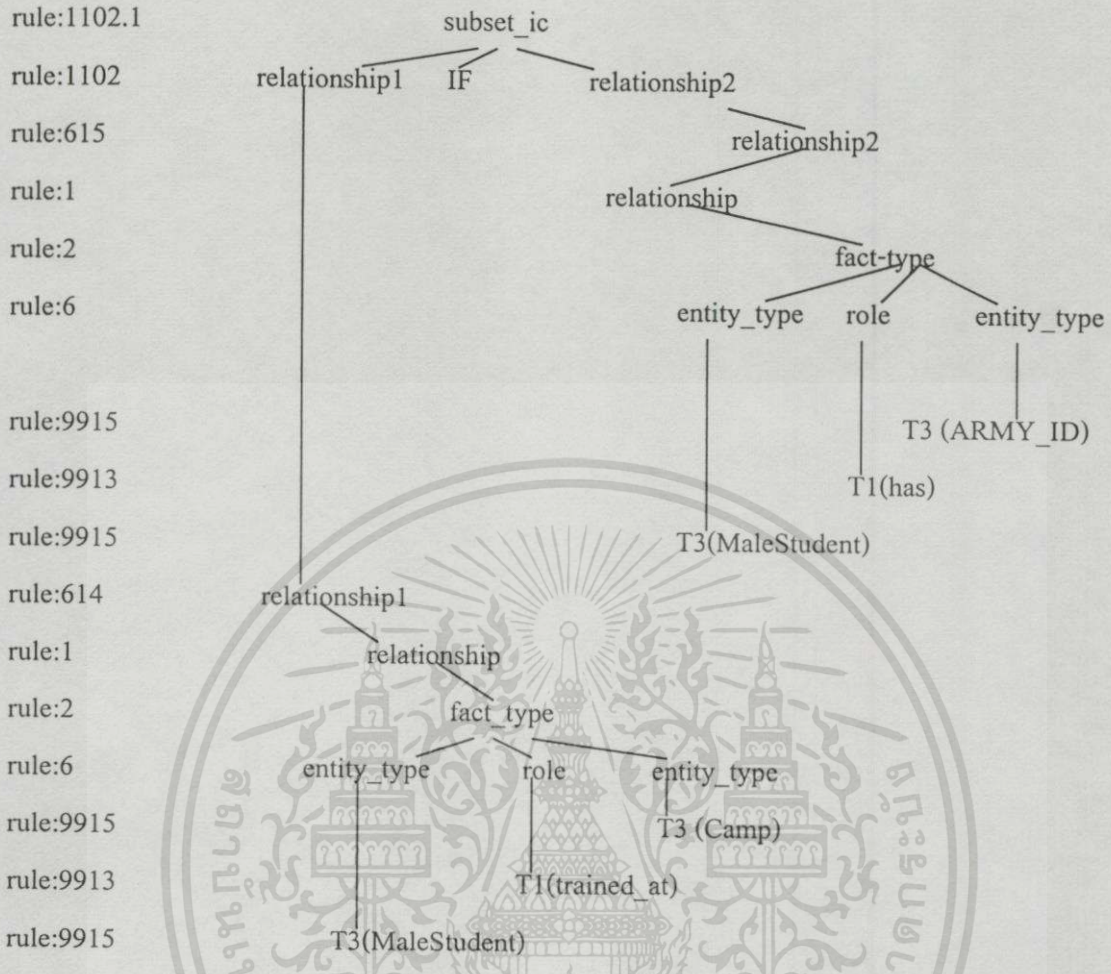
$cc y.tab.c จะได้เอ็กซิคิวไฟล์สำหรับเป็นตัวแปลภาษา SL/NIAM เพื่อนำไปใช้ในการพัฒนาในส่วนของงานวิจัยในลำดับที่จะกล่าวถัด ๆ ไป

สำหรับกรณีที่ต้องเชื่อมการทำงานกับระบบอื่น ๆ ในงานวิจัยครั้งนี้จำเป็นจะต้องสร้างให้อยู่ในรูปแบบ y.tab.o โดยมีรูปแบบคำสั่งคือ $cc y.tab.c –c จะได้ y.tab.o สำหรับ y.tab.o นี้จะนำไปรวมกับการคอมไพล์ด้วยตัวโปรแกรมภาษาโปรสตาร์ซี

ตัวอย่างรูปแบบการทำงานตัวแปลภาษาที่ได้พัฒนา เมื่อนำมาใช้งานจะทำการตรวจสอบหลักไวยากรณ์ให้เป็นไปตามกฎต่าง ๆ ที่พัฒนาขึ้นมาโดยรูปแบบการตรวจความถูกต้องทางหลักไวยากรณ์นั้นสอบนั้นเราจะแสดงให้เห็นเป็นลักษณะของต้นไม้ได้ดังต่อไปนี้คือ กรณีรูปประโยคเป็น Student enrooled_in Subject in Semester of Year ดังรูปที่ 7.1 และกรณีรูปประโยคเป็น MaleStudent trained_at Camp IF MaleStudent has ARMY_ID ดังรูปที่ 7.2



รูปที่ 7.1 แสดง Parse Tree ของประโยค Student enroll_in Subject in Semester of Year



รูปที่ 7.2 แสดง Parse Tree ของประโยค MaleStudent trained_at Camp IF MaleStudent has ARMY_ID

7.3.3 ขั้นตอนการออกแบบตารางเมทต้าและวิธีการจัดเก็บข้อมูลลงในตารางเมทต้า

หลังจากที่เราได้ทำการพัฒนาตัวแปลภาษา SL/NIAM เสร็จเรียบร้อยแล้วขั้นตอนต่อไปก็คือการพัฒนาระบบฐานข้อมูลเมทต้าขึ้นมาเพื่อไว้สำหรับเก็บข้อมูลในลักษณะของกฎข้อบังคับความถูกต้องของข้อมูลโดยที่นำเข้าในรูปภาษา SL/NIAM ขั้นตอนที่ผู้วิจัยได้ทำการพัฒนาระบบฐานข้อมูลเมทต้านี้เริ่มจาก

- ศึกษารูปแบบกฎข้อบังคับความถูกต้องของข้อมูลเชิงไนแอมเพื่อนำมาสร้างเป็น NIAM Meta Conceptual Schema
- ทำการแมปกฎบังคับความถูกต้องของข้อมูลในข้อแรกโดยใช้หลักของอัลกอริธึม ONF เพื่อสร้างรีเลชั่นสกีมม่า ผลที่ได้ในที่นี้เราจะได้ตารางฐานข้อมูลที่เรียกว่าเป็นตารางเมทต้า เพื่อใช้สำหรับเก็บกฎข้อบังคับของข้อมูลจากรูปประโยคภาษา SL/NIAM

หลังจากแมปด้วยหลักการของอัลกอริธึม ONF ในกรณีนี้เราจะทำการสร้างในลักษณะของ

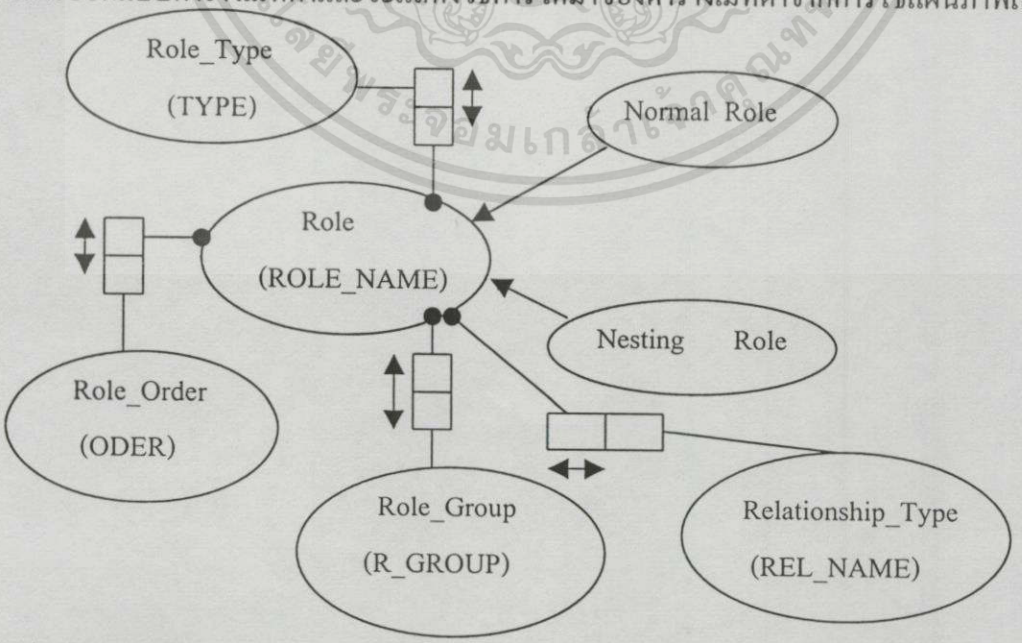
ตามหลักของอัลกอริธึมเพื่อออกแบบเป็นตารางฐานข้อมูลโดยการออกแบบนั้นจะต้องมีการปรับปรุงจนกว่าจะเหมาะสมในการใช้จัดเก็บข้อมูลให้ครอบคลุมมากที่สุด ซึ่งหากใช้หลักของอัลกอริธึม ONF โดยตรงจริง ๆ แล้วยังไม่เอื้ออำนวยในการออกแบบเป็นฐานข้อมูลในการจัดเก็บข้อมูลให้ได้ตามต้องการเพื่อจะนำข้อมูลที่จัดเก็บนี้สำหรับใช้ในการทำวิจัยในขั้นตอนต่อ ๆ ไป ซึ่งจะได้กล่าวถัดมา ขออธิบายการนำมาสร้างเป็นตารางฐานข้อมูลเมทต้าเพื่อใช้ในการเก็บกฎข้อบังคับความถูกต้องของข้อมูลได้ดังรายละเอียดต่อดังนี้คือ

จากหลักการของอัลกอริธึม ONF ที่มีการทำงานคือ

- สำหรับแต่ละชนิดความจริงที่ไม่ได้มี simple key ให้สร้างเป็นตารางแยกออกมาโดยเลือกคีย์ที่สั้นที่สุดของชนิดความจริงนั้นเป็นคีย์หลัก
- ให้ทำการรวมชนิดความจริงที่มี simple key ที่เชื่อมกับ object type เดียวกันเป็นตารางเดียวกัน และให้ object type นั้นเป็นคีย์หลัก
- สำหรับชนิดความจริงที่เหลือให้สร้างเป็นตารางแยกออกขึ้นมาใหม่

จากการออกแบบตารางเมทต้าเราจะเริ่มกำหนดให้ทุก ๆ ชนิดเอนติตี้มีความสัมพันธ์กันในลักษณะแบบ 1 : 1 และไม่มีการใช้การเป็นสับไทป์ ดังนั้นเราสามารถสร้างตารางเมทต้าดังนี้คือ

ลักษณะของตารางที่สอดคล้องกับกฎของการไม่มี simple key ก็คือเป็นการแมปข้อมูลชนิดความจริงที่เป็นแบบไบนารีที่มีความสัมพันธ์กันแบบ many to many หรือชนิดที่มีโรลมากกว่าสองขึ้นไป สำหรับข้อมูลในแผนภาพข้างล่างนี้รูปที่ 7.3 เป็นส่วนหนึ่งของข้อมูลที่ผู้วิจัยได้นำมาใช้ในการออกแบบตารางเมทต้าและจะแสดงวิธีการได้มาของตารางเมทต้าจากการใช้แผนภาพเหล่านี้



รูปที่ 7.3 แสดงตัวอย่างความสัมพันธ์ของข้อมูลในลักษณะไนแอม

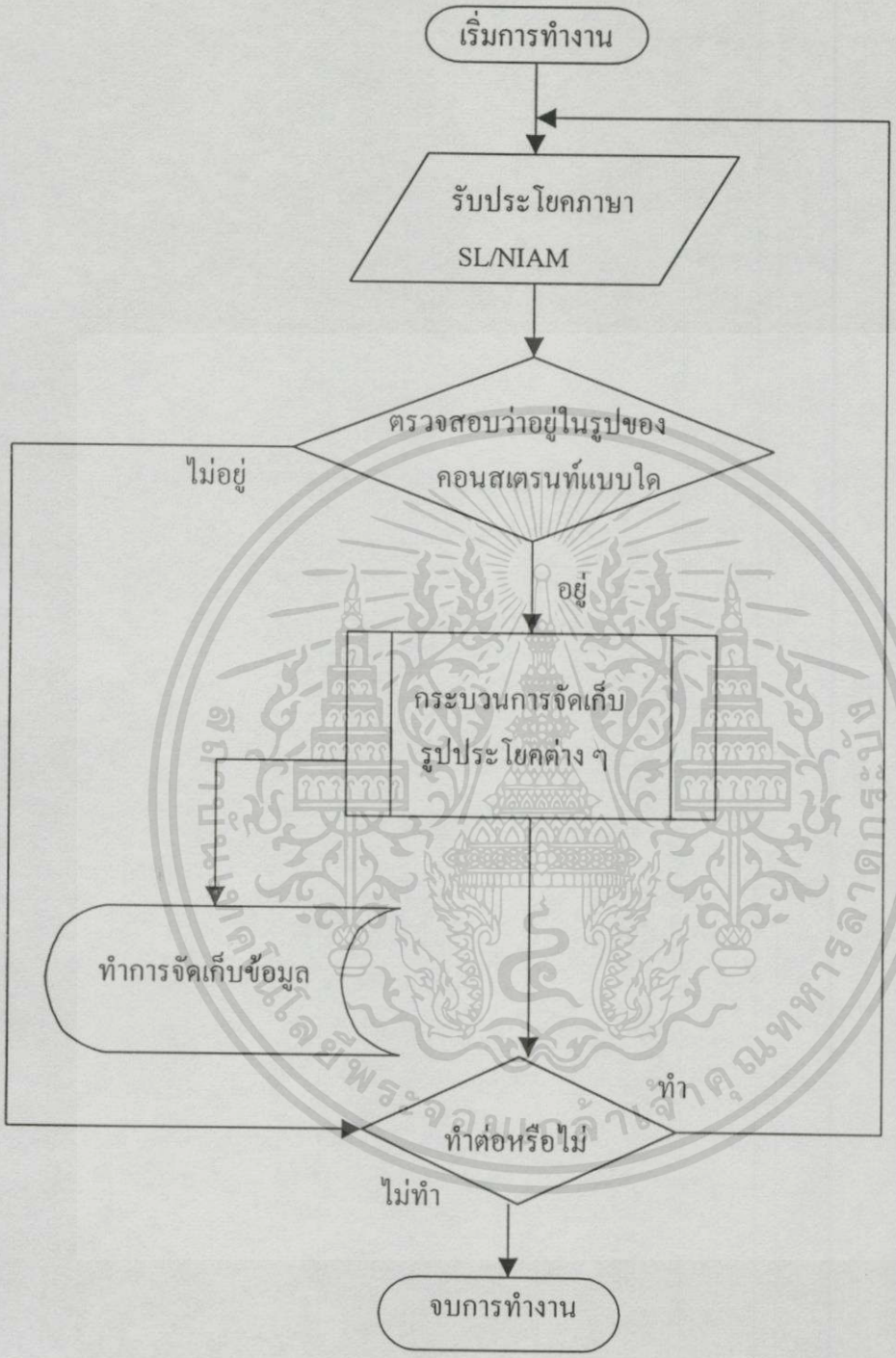
จากรูปภาพที่ 7.3 เราจะอธิบายการได้มาของตารางเมทต้าตามหลักของอัลกอริธึมของการไม่มี simple key นั่นคือให้คัดเลือกหาความสัมพันธ์ของชนิดเอนติตี้ที่เป็นแบบ many to many หรือมีโรลมากกว่าสองขึ้นไป ซึ่งเราจะพบว่ามีชนิดเอนติตี้ของ Role กับชนิดเอนติตี้ของ Role_Group ที่แสดงสัมพันธ์กันในลักษณะแบบ many to many โดยเรากำหนดให้ใช้แอทริบิ้วในชนิดเอนติตี้ของ Role เป็นคีย์หลัก ดังนั้นเราจะนำมาสร้างเป็นตารางได้ดังตารางข้างล่างนี้ แต่ในการทำงานของผู้วิจัยได้เพิ่มจำนวนคอลัมน์ขึ้นตามความเหมาะที่จะจัดเก็บข้อมูลเพื่อใช้ในการวิจัยในขั้นต่อ ๆ ไป

ROLE_NAME	R_GROUP

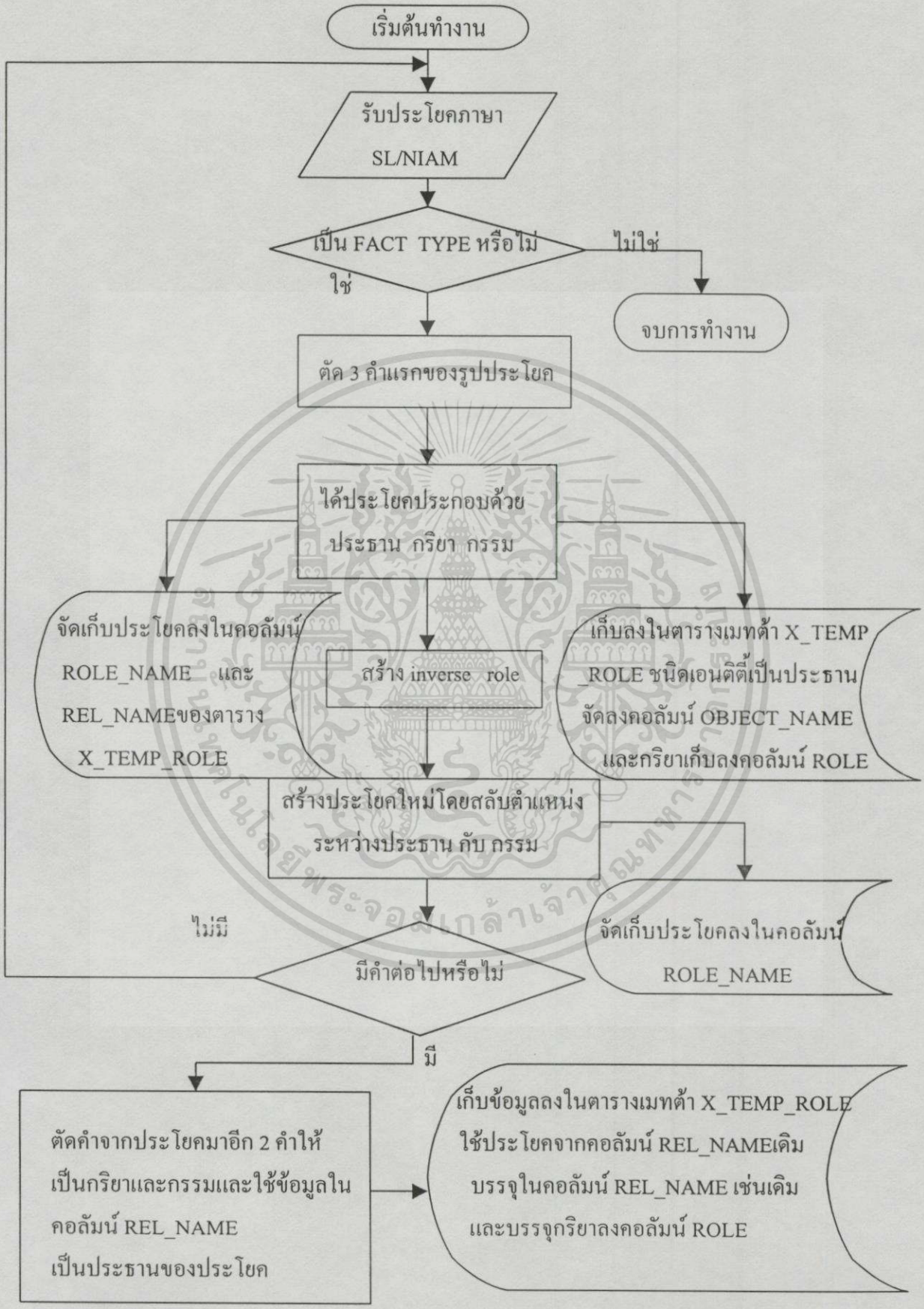
ลักษณะของตารางที่สอดคล้องกับกฎของการที่มี simple key ก็คือเป็นการแมปข้อมูลของชนิดความจริงที่เป็นแบบไบนารีที่มีชนิดเอนติตี้ใดมี simple key เชื่อมอยู่เราจะรวมเป็นชนิดความจริงเป็นหนึ่งตาราง และได้กำหนดให้แอทริบิ้วในชนิดเอนติตี้นั้นเป็นคีย์หลัก ใช้ข้อมูลในแผนภาพที่ 7.3 จากแผนภาพจะทราบได้ว่ามีชนิดเอนติตี้ที่มี simple key นั่นก็คือ Role, Role_Type, Role_Order และ Relationship_Type โดยเราจะกำหนดให้มีการใช้แอทริบิ้วของ Role เป็นคีย์หลัก ดังนั้นสามารถสร้างเป็นตารางเมทต้าได้ดังนี้คือ

ROLE_NAME	TYPE	ODER	REL_NAME

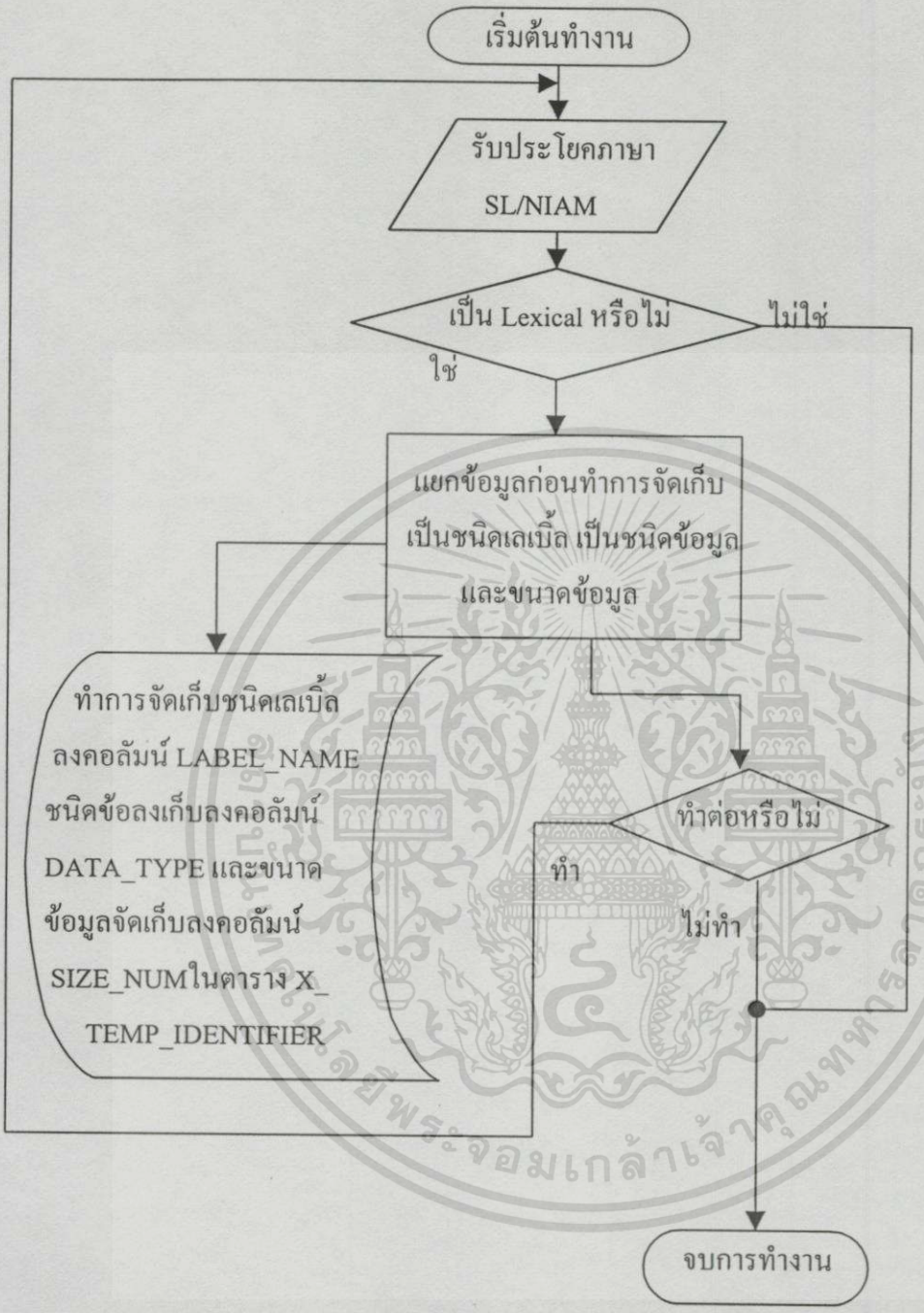
หลังจากที่ได้ทำการพัฒนาออกแบบตารางเมทต้าเพื่อใช้สำหรับทำการจัดเก็บข้อมูลนำเข้าเรียบร้อยแล้วขั้นตอนต่อไปผู้วิจัยได้ทำการพัฒนาอัลกอริธึมสำหรับใช้ในการแยกแยะข้อมูลที่ได้จากรูปแบบของภาษา SL/NIAM ในการจัดเก็บลงตารางเมทต้าเพื่อที่จะนำข้อมูลเหล่านี้ซึ่งเป็นกฎข้อบังความถูกต้องของข้อมูลไปใช้ในการพัฒนาวิจัยในขั้นตอนต่อๆ ไปซึ่งจะกล่าวต่อไปในลำดับถัดมา สำหรับขั้นตอนการแยกแยะวิเคราะห์ข้อมูลเพื่อทำการจัดเก็บลงในตารางเมทต้าสามารถแสดงในรูปงภาพรวม ๆ ได้ดังรูปที่ 7.4 และแสดงภาพย่อย ๆ ของการแยกแยะในแต่ละรูปประโยคตามแต่ละหลักไวยากรณ์ในบางส่วนของการทำวิจัยในครั้งนี้ดังแสดงในรูปที่ 7.5 รูปที่ 7.6 และรูปที่ 7.7 ตามลำดับ ดังรูปต่อไปนี้



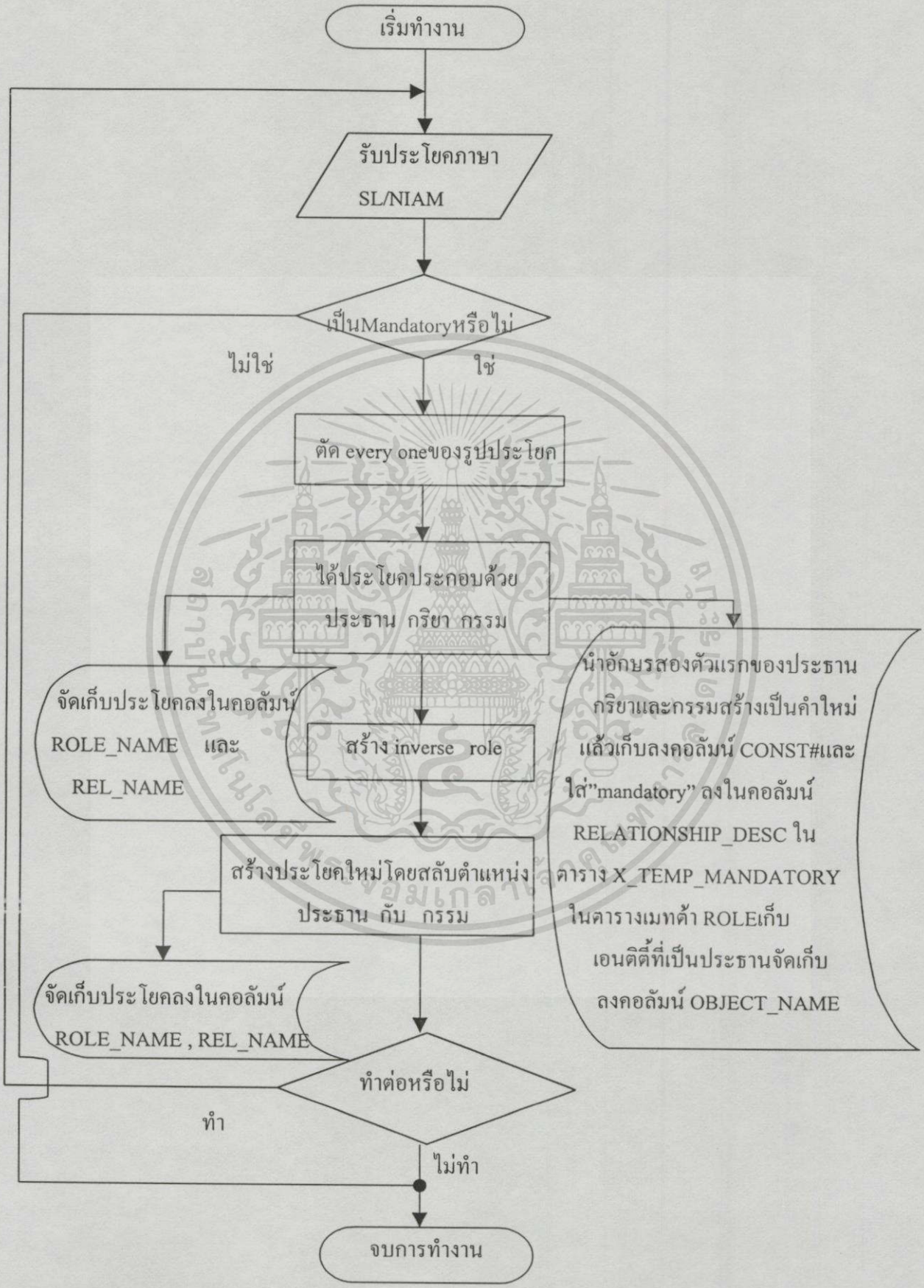
รูปที่ 7.4 แสดงภาพรวมการวิเคราะห์การจัดเก็บข้อมูลลงในตารางเมทต้า



รูปที่ 7.5 แสดงการวิเคราะห์และจัดเก็บรูปประโยคลักษณะ Fact Typeลงในตารางเมทต้า



รูปที่ 7.6 แสดงการวิเคราะห์และจัดเก็บรูปประโยคลักษณะ Lexical ลงในตารางเมทต้า



รูปที่ 7.7 แสดงการวิเคราะห์และจัดเก็บรูปประโยคลักษณะ Mandatory ลงในตารางเมทด้า

ตารางต่อไปนี้เป็นการแสดงลักษณะบางส่วนของข้อมูลที่ได้ทำการจัดเก็บลงในตารางเมทต้าซึ่งสามารถแสดงได้ดังลักษณะข้างล่างต่อไปนี้ในตารางต่าง ๆ โดยเราจะทำการจัดเก็บข้อมูลในรูปประโยคภาษา SL/NIAM ต่อไปนี้ลงในตารางเมทต้า

รูปประโยคแบบ Fact Type ข้อมูลที่ทำการจัดเก็บจะมีตารางที่เกี่ยวข้องในการจัดเก็บข้อมูลลงนั่นก็คือในตารางเมทต้า X_TEMP_ROLE

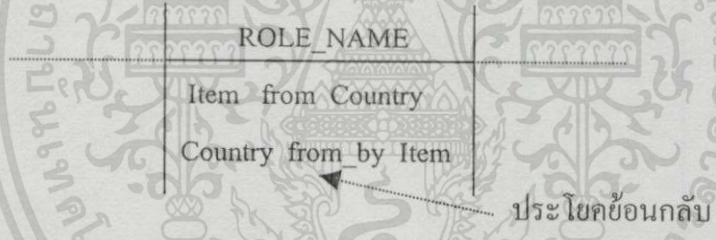
รูปประโยค

Item from Country

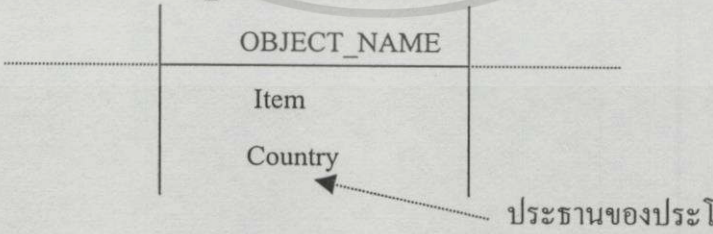
ตาราง X_TEMP_ROLE

ขั้นตอนการจัดเก็บ

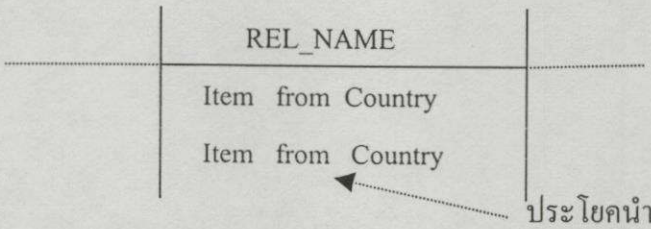
- นำประโยคใส่ลงในคอลัมน์ ROLE_NAME จะได้ Item from Country และประโยคย้อนกลับจะเป็น Country from_by Item ซึ่งได้ทำการเปลี่ยนกรรมของประโยคแรกให้เป็นประธานของประโยคย้อนกลับ และทำการเปลี่ยนแปลงโรลจาก from ไปเป็น from_by ของประโยคย้อนกลับ



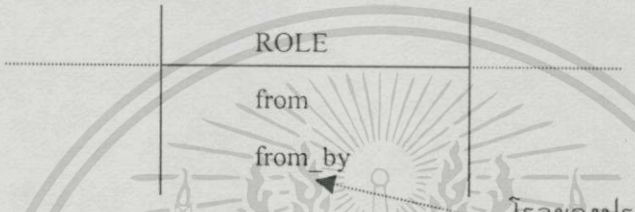
- นำประธานของประโยค Item from Country นั่นก็คือ Item จัดเก็บลงในคอลัมน์ OBJECT_NAME และนำประธานของประโยคย้อนกลับนั่นคือ Country จัดเก็บลงในคอลัมน์ OBJECT_NAME ด้วยเช่นกัน



- นำประโยคใส่ลงในคอลัมน์ REL_NAME จะได้ Item from Countryและประโยคย้อนกลับจะใช้เป็น Item from Country เหมือนกันไม่เปลี่ยนแปลงรูปประโยคแรก



- นำโรลของประโยค Item from Country นั่นก็คือ from จัดเก็บลงในคอลัมน์ ROLE และนำโรลของประโยคย้อนกลับนั่นคือ from_by จัดเก็บลงในคอลัมน์ ROLE ด้วยเช่นกัน



รูปประโยคแบบ Unique ข้อมูลที่ทำการจัดเก็บจะมีตารางที่เกี่ยวข้องในการจัดเก็บข้อมูลลงนั่นก็คือในตารางเมทต้า X_TEMP_ROLE และ X_TEMP_CONSTRAINT

รูปประโยค

one Item from one Country

ตาราง X_TEMP_ROLE

ขั้นตอนการจัดเก็บ

- นำประโยคมาตัดคำให้เหลือเป็นรูปประโยคแบบ Fact Type โดยทำการตัด one และ many ออกจากประโยค นั่นก็คือจะได้ประโยคจากที่ยกตัวอย่างเป็น Item from Country ใส่ลงในคอลัมน์ ROLE_NAME จะได้ Item from Country และประโยคย้อนกลับจะเป็น Country from_by Item ซึ่งได้ทำการเปลี่ยนกรรมของประโยคแรกให้เป็นประธานของประโยคย้อนกลับ และทำการเปลี่ยนแปลงโรลจาก from ไปเป็น from_by ของประโยคย้อนกลับ และทำขั้นตอนต่อ ๆ ไป เช่นเดียวกับที่ได้กล่าวถึงวิธีการจัดเก็บข้อมูลลงในตารางเมทต้า X_TEMP_ROLE ที่ได้กล่าวไว้ก่อนหน้านี้แล้ว

ตาราง X_TEMP_CONSTRAINT

ขั้นตอนการจัดเก็บ

- จากรูปประโยค one Item from one Country ทำการตัดคำเหลือรูปประโยคเป็น

Item from Country แล้วให้นำตัวอักษร 2 ตัวแรกของแต่ละคำมารวมกันแล้วทำการจัดเก็บลงในคอลัมน์ CONST#

CONST#
ItfrCo

- ในคอลัมน์ CONST_TYPE จะเป็นการใส่ชื่อแสดงถึงรูปแบบของคอนสเตรนท์ ซึ่งจากรูปประโยคนี้ก็จะเป็น unique

CONST_TYPE
unique

- ในคอลัมน์ของ OBJECT_NAME จะทำการจัดเก็บประธานของประโยคในที่นี้กรณีประโยคเป็น one Item from one Country เมื่อตัดคำแล้วเปลี่ยนรูปประโยคให้เป็น Fact Type แล้วจะมีประโยคเป็น Item from Country นั่นก็คือจะมี Item เป็นประธานของโยค

OBJECT_NAME
Item

- ในคอลัมน์ของ RELATIONSHIP_NAME จะทำการจัดเก็บรูปประโยคเมื่อตัดคำแล้วทำเป็นรูปประโยคแบบ Fact Type แล้วนั่นคือเป็น Item from Country

RELATIONSHIP_NAME
Item from Country

- ในคอลัมน์ของ RELATE_ENTITY จากรูปประโยคความสัมพันธ์จะเป็นแบบ many to one ดังนั้นจะกำหนดให้เป็น 'mo'

RELATE_ENTITY
mo

รูปประโยคแบบ LEXICAL ข้อมูลที่ทำการจัดเก็บจะมีตารางที่เกี่ยวข้องในการจัดเก็บข้อมูลลงนั่นก็คือในตารางเมทต้า X_TEMP_REFERENCE_TYPE

รูปประโยค

Item(ITEM_NR)

ตาราง X_TEMP_REFERENCE_TYPE

ขั้นตอนการจัดเก็บ

- ในคอลัมน์ LABEL_NAME จะทำการจัดเก็บชนิดเลเบิ้ลลงไปซึ่งในข้อมูลของประโยคจากตัวอย่างที่แสดงนี้จะทำการจัดเก็บ ITEM_NR ลงไป

LABEL_NAME
ITEM_NR

- ในคอลัมน์ของ ENTITY_NAME จะทำการจัดเก็บข้อมูลเกี่ยวกับชนิดเอนติตี้จากประโยคในที่นี้ก็คือ Item ลงไป

ENTITY_NAME
Item

รูปประโยคแบบ IDENTIFIER ข้อมูลที่ทำการจัดเก็บจะมีตารางที่เกี่ยวข้องในการจัดเก็บข้อมูลลงนั่นก็คือในตารางเมทต้า X_TEMP_IDENTIFIER

รูปประโยค

ITEM_NR char(15)

ตาราง X_TEMP_IDENTIFIER

ขั้นตอนการจัดเก็บ

- ในคอลัมน์ LABEL_NAME จะทำการจัดเก็บชนิดเลเบิ้ลลงไปซึ่งในข้อมูลของประโยคจากตัวอย่างที่แสดงนี้จะทำการจัดเก็บ ITEM_NR ลงไป

LABEL_NAME
ITEM_NR

- ในคอลัมน์ DATA_TYPE จะทำการจัดเก็บชนิดเลเบิ้ลลงไปซึ่งในข้อมูลของประโยค จากตัวอย่างที่แสดงนี้จะทำการจัดเก็บ char ลงไป

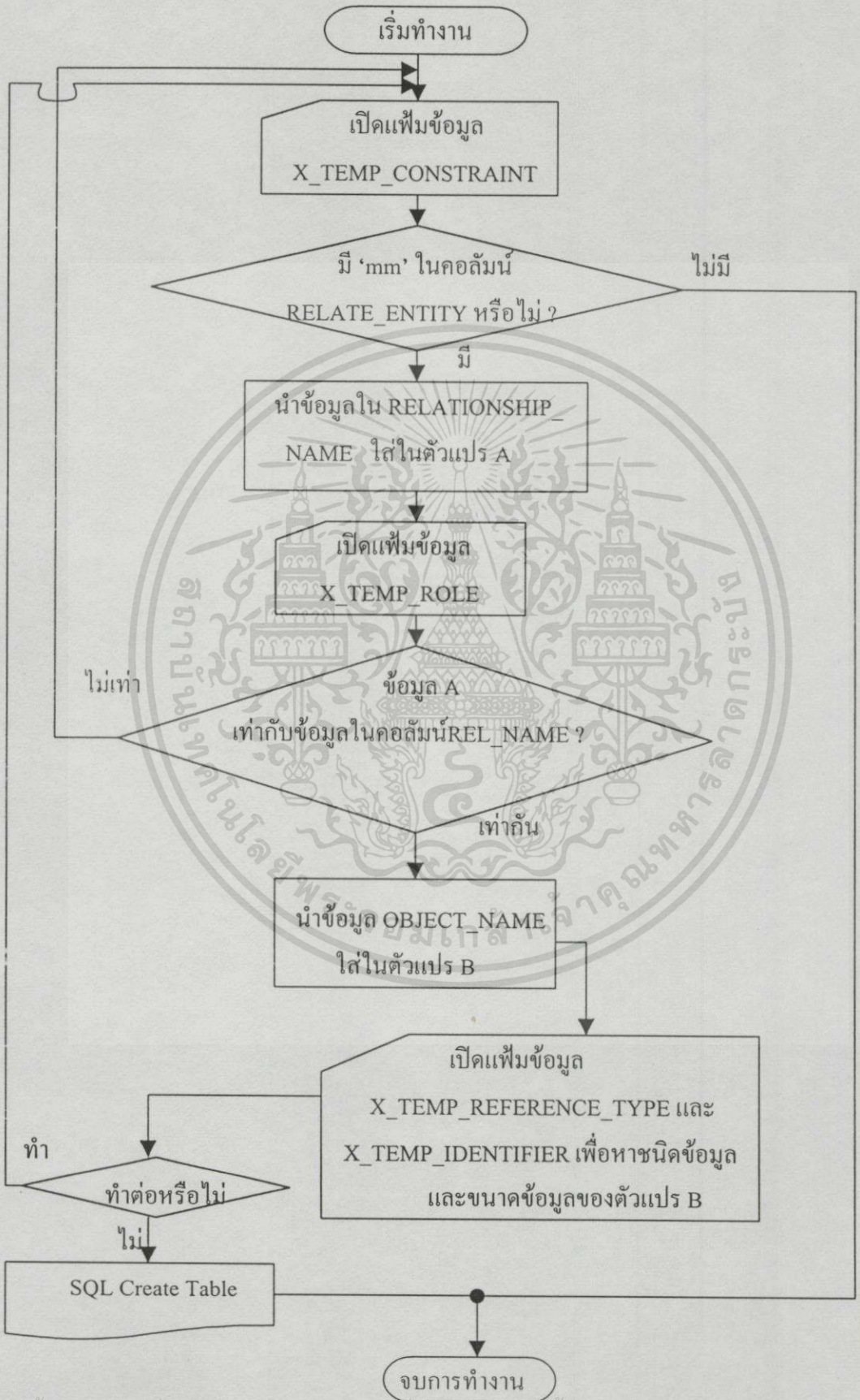
DATA_TYPE
char

- ในคอลัมน์ SIZE_NUM จะทำการจัดเก็บชนิดเลเบิ้ลลงไปซึ่งในข้อมูลของประโยค จากตัวอย่างที่แสดงนี้จะทำการจัดเก็บ 15 ลงไป

SIZE_NUM
15

ลักษณะของการจัดเก็บข้อมูลลงในตารางเมทต้าสามารถแสดงข้อมูลที่ได้ทำการจัดเก็บลงได้จริง ๆ โดยจะใช้ข้อมูลในบางคอลัมน์มาอธิบายที่ใช้สำหรับสร้างรีเลชั่นสกีมม่า ซึ่งจะแสดงดังลักษณะไดอะแกรมได้ดังต่อไปนี้คือ

กรณีไม่มี Simple key



รูปที่ 7.8 แสดงวิธีการทำรีเลชั่นสกีมม่ากรณีไม่มี Simple key

กรณีไม่มี Simple key

ตาราง X-TEMP_ROLE

ROLE_NAME	OBJECT_NAME	REL_NAME
Item from Country	Item	Item from Country
Country from_by Item	Country	Item from Country
Item own Dept	Item	Item own Dept
Dept own_by Item	Dept	Item own Dept
Item has ItemName	Item	Item has ItemName
ItemName of Item	ItemName	Item has ItemName

ตาราง X_TEMP_CONSTRAINT

RELATIONSHIP_NAME	RELATE_ENTITY
Item from Country	mo
Item own Dept	mm
Item has ItemName	mo

ตาราง X_TEMP_REFERENCE_TYPE

LABEL_NAME	ENTITY_NAME
ITEM_NR	Item
NAME	Country
CODE	Dept
ITEMNAME	ItemName

ตาราง X_TEMP_IDENTIFIER

LABEL_NAME	DATA_TYPE	SIZE_NUM
ITEM_NR	char	15
NAME	char	25
CODE	char	7
ITEMNAME	char	15

รูปที่ 7.9 แสดงการใช้ข้อมูลในตารางเมทต้ากรณีไม่มี Simple key

7.3.4 ขั้นตอนและการทำรีเลชั่นสกีม่า

ข้อมูลที่ได้จัดเก็บจากตารางเมทต้าเราสารมารถจะนำมาสร้างรีเลชั่นสกีมม่า โดยใช้หลักการอัลกอริธึม ONF ซึ่งผลที่ได้จากการทำรีเลชั่นสกีม่านี้จะได้ตารางในขั้นสุดท้ายที่อยู่ในรูปของ 5NF และในบางข้อมูลสามารถพัฒนาอยู่ในรูปของ Database Trigger และ Store Procedures สำหรับข้อมูลที่เราได้แสดงไว้ในข้างต้นที่ผ่านมาเราสามารถนำมาอธิบายขั้นตอนอย่างละเอียดได้ดังต่อไปนี้คือ

เริ่มทำการตรวจสอบข้อมูลจากตาราง X_TEMP_CONSTRAINT ในคอลัมน์ของ RELATE_ENTITY สามารถนำมาสร้างการทำงานในลักษณะแบบไม่มี Simple key ดังรูปที่ 7.8 และแบบมี Simple key จากข้อมูลจะขอเริ่มจากการทำงานแบบไม่มี Simple key ดังรูปที่ 7.9 ซึ่งข้อมูลในคอลัมน์ RELATE_ENTITY ที่มีการทำงานแบบนี้จะปรากฏข้อมูลเป็น 'mm' ซึ่งแสดงถึงการมีความสัมพันธ์แบบ many to many เมื่อเราพบว่ามี 'mm' แล้วให้เราทำการจัดเก็บข้อมูลที่อยู่ในคอลัมน์ RELATIONSHIP_NAME ที่อยู่ในแถวเดียวกันกับ 'mm' ใส่ในตัวแปรตัวหนึ่งโดยมีทำงานในลักษณะของอะเรย์ในที่นี้เราจะให้ตัวแปรเป็น A ในทางโปรแกรม นั่นคือ A[1]= 'Item own Dept' หลังจากนั้นก็ให้นำข้อมูลในตัวแปร A[1] นี้ไปทำการเปรียบเทียบว่ามีอยู่ในตารางของ X_TEMP_ ROLE หรือไม่ ในกรณีนี้จะพบว่ามีข้อมูล 'Item own Dept' อยู่ในคอลัมน์ของ REL_NAME ในที่นี้มีอยู่ถึง 2 แถว ดังนั้นให้เราทำการจัดเก็บข้อมูลในคอลัมน์ OBJECT_NAME แต่ละแถวไว้ในตัวแปร B ที่ทำงานแบบอะเรย์ ดังนั้นจะได้ B[1]= 'Item' และ B[2]= 'Dept' จากความสัมพันธ์ของเอนติตี้ทั้งสองนั่นก็คือ 'Item' และ 'Dept' ตามหลักการของการไม่มี Simple key แล้วเราจะถือว่าเป็นเอนติตี้ที่สามารถนำมาสร้างเป็นตารางฐานข้อมูลเดียวกัน หลังจากนั้นให้นำข้อมูลในตัวแปร B ไปทำการหาชนิดข้อมูลและขนาดของข้อมูล แล้วสร้างนิยามของตารางฐานข้อมูลด้วยภาษา SQL โดยมีคีย์หลักได้จากข้อมูลในตัวแปรของ B[1] และ B[2]

จะได้ตารางกรณีไม่มี Simple key

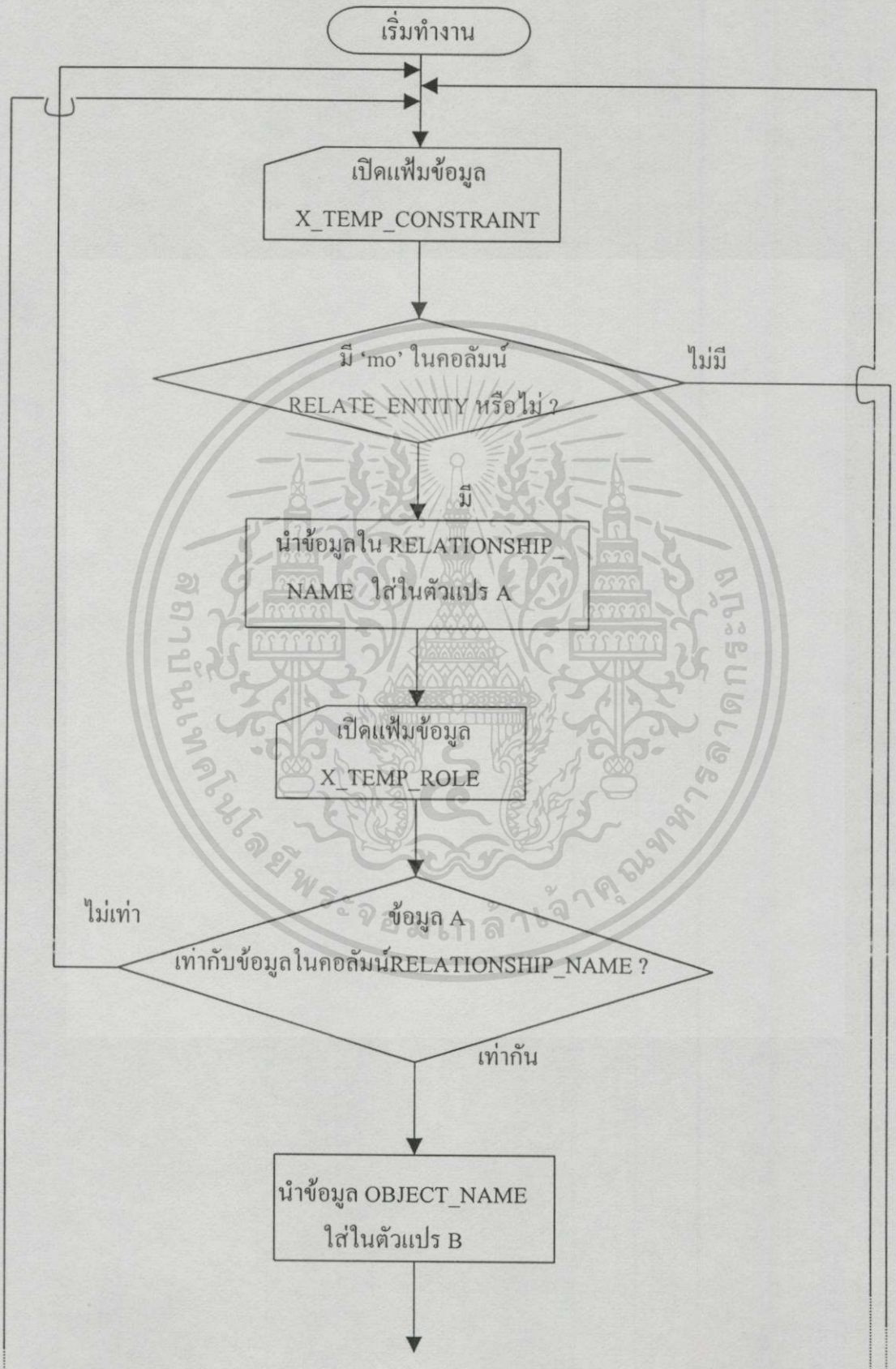
TABLE NAME IS : many_to_many_1

ITEM_NR char 15

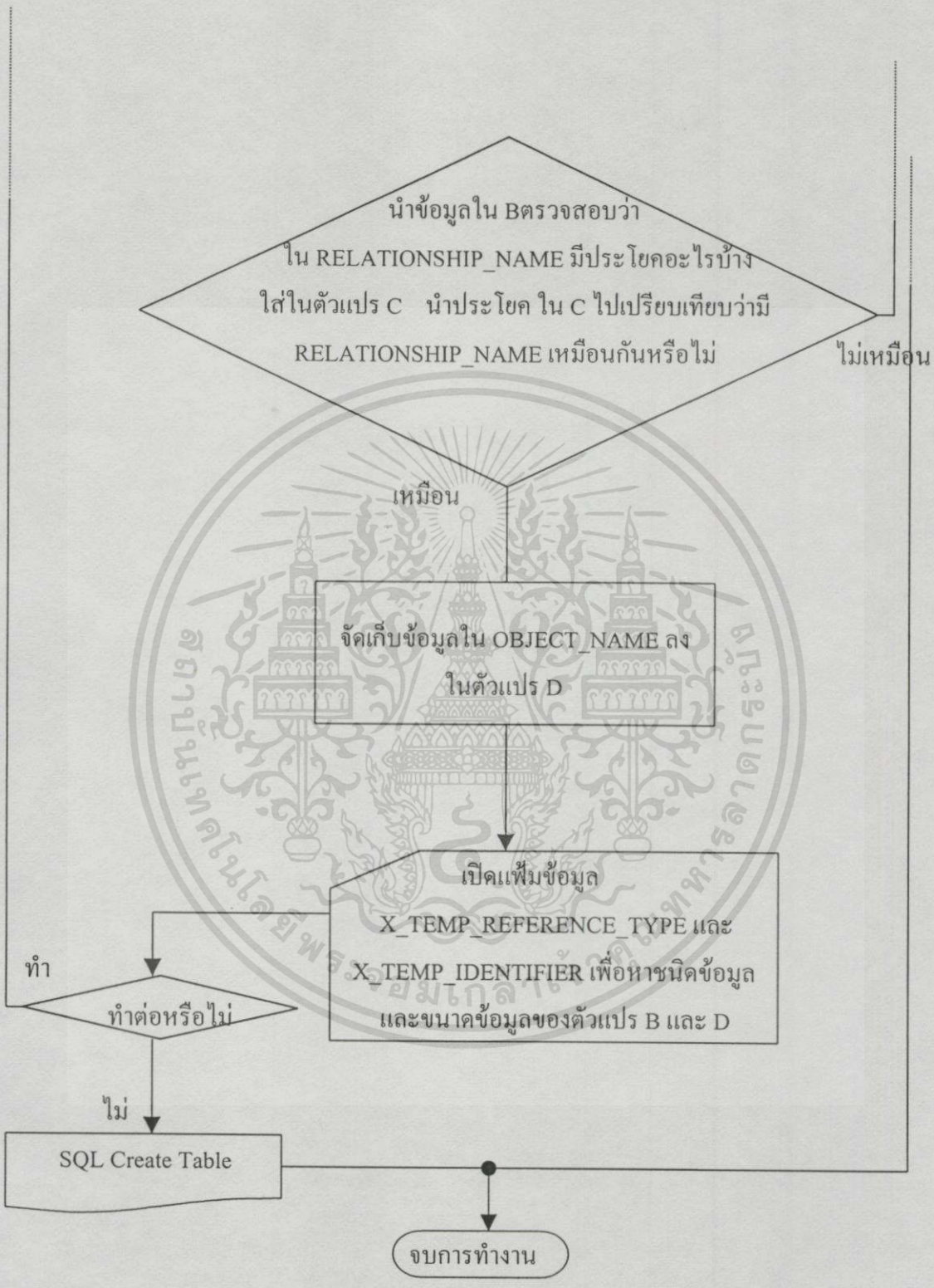
CODE char 7

PRIMARY KEY IS: ITEM_NR, CODE

กรณีมี Simple key



รูปที่ 7.10 แสดงวิธีการทำรีเลชั่นสกีมม่ากรณีมี Simple key



รูปที่ 7.10 (ต่อ)

กรณีมี Simple key

ตาราง X-TEMP_ROLE

ROLE_NAME	OBJECT_NAME	REL_NAME
Item from Country	Item	Item from Country
Country from_by Item	Country	Item from Country
Item own Dept	Item	Item own Dept
Dept own_by Item	Dept	Item own Dept
Item has ItemName	Item	Item has ItemName
ItemName of Item	ItemName	Item has ItemName

ตาราง X_TEMP_CONSTRAINT

RELATIONSHIP_NAME	RELATE_ENTITY
Item from Country	mo
Item own Dept	mm
Item has ItemName	mo

ตาราง X_TEMP_REFERENCE_TYPE

LABEL_NAME	ENTITY_NAME
ITEM_NR	Item
NAME	Country
CODE	Dept
ITEMNAME	ItemName

ตาราง X_TEMP_IDENTIFIER

LABEL_NAME	DATA_TYPE	SIZE_NUM
ITEM_NR	char	15
NAME	char	25
CODE	char	7
ITEMNAME	char	15

รูปที่ 7.11 แสดงการใช้ข้อมูลในตารางเมทต้ากรณีมี Simple key

จากรูปประโยคที่เป็นกฎข้อบังคับต่าง ๆ ในการออกแบบฐานข้อมูลที่ได้ทำการจัดเก็บลงในตารางเมทต้านั้น เราสามารถจะพิจารณาลักษณะของข้อมูลในแบบมี Simple key ดังรูปที่ 7.10 ได้ โดยจากการจัดเก็บข้อมูลโดยกำหนดให้ข้อมูลในคอลัมน์ของ RELATE_ENTITY ในตารางของ X_TEMP_CONSTRAINT เป็น 'mo' ซึ่งกำหนดให้เป็นความสัมพันธ์แบบ many to one และกำหนดให้ 'oo' เป็นความสัมพันธ์ในลักษณะแบบ one to one จากตัวอย่างเราจะอธิบายถึงการสร้างรีเลชั่นสกีมม่าในลักษณะแบบที่มี Simple key ดังรูปที่ 7.11 โดยขอเริ่มจากให้ทำการตรวจดูว่าในคอลัมน์ของ RELATE_ENTITY ของตาราง X_TEMP_CONSTRAINT เป็น 'mo' หรือ 'oo' หรือไม่ ในกรณีนี้เราจะพบว่ามี 'mo' และก็มีถึง 2 ครั้งเสียด้วย ขั้นต่อไปให้เราทำการจัดเก็บข้อมูลในคอลัมน์ของ RELATIONSHIP_NAME โดยให้แต่ละแถวตรงกับ 'mo' โดยทำการจัดเก็บลงในตัวแปร A โดยที่ตัวแปร A นี้จะทำงานในลักษณะของอะเรย์ ดังนั้นเราจะได้ว่า A[1]= 'Item from Country' และข้อมูลใน A[2] = 'Item has ItemName' หลังจากนั้นให้นำข้อมูลในตัวแปร A เหล่านี้ไปทำการเปรียบเทียบข้อมูลในคอลัมน์ REL_NAME จะพบว่าข้อมูลในตัวแปร A[1] มีอยู่ในคอลัมน์ REL_NAME ให้เราทำการจัดเก็บข้อมูลในคอลัมน์ OBJECT_NAME ไว้ในตัวแปร B ซึ่งตัวแปร B นี้จะทำงานในลักษณะอะเรย์ จากข้อมูลที่มีอยู่นี้เราจัดเก็บให้ B[1]= 'Item' และในตัวแปร B[2]= 'Country' และข้อมูลในตัวแปร A[2] มีอยู่ในคอลัมน์ REL_NAME เช่นกันดังนั้นเราจะจัดเก็บให้ B[3]= 'ItemName' โดยที่ Item มีซ้ำกันถ้าจัดเก็บแล้วจะไม่ทำการจัดเก็บอีก เนื่องจากความสัมพันธ์ของข้อมูลในตัวแปร B[1] ที่เป็น Item นี้ มีความสัมพันธ์ของข้อมูลกับ Country และ ItemName เนื่องจากเกิดมาจากรูปประโยคที่มี Item เดียวกัน และยังมี Simple key แบบเดียวกันอีกด้วย ดังนั้นเราจะรวมเอาข้อมูลที่เป็นเอนติตี้เหล่านี้เป็นข้อมูลเดียวกันและให้จัดเก็บลงในตัวแปร D โดยให้ตัวแปร D ทำงานในลักษณะอะเรย์เช่นกัน โดยให้ตัวแปร D[1]= 'Item' ตัวแปร D [2]= 'Country' และให้ตัวแปร D[3]= 'ItemName' หลังจากนั้นเราจะหาเอนติตี้อ้างอิงของข้อมูลที่เราได้ทำการจัดเก็บในแต่ละตัว โดยใช้ข้อมูลในตัวแปร D เปรียบเทียบกับข้อมูลในคอลัมน์ ENTITY_NAME ของตาราง X_TEMP_REFERENCE_TYPE ถ้าข้อมูลในตัวแปรไหนเหมือนกันให้ทำการนำข้อมูลในคอลัมน์ LABEL_NAME ไปหาขนาดและชนิดข้อมูลในตาราง X_TEMP_IDENTIFIER จนกระทั่งเราจะได้ข้อมูลทั้งหมดที่จะนำมาเขียนเป็นนิยามของตารางฐานข้อมูลด้วยภาษา SQL ซึ่งตารางฐานข้อมูลที่ได้นี้จะถือว่าอยู่ในขั้นของ 5NF แล้ว และเราจะกำหนดให้ ตัวแปรใน B[1] ที่เป็น Item เป็นคีย์หลักของตารางที่ได้ทำการสร้าง

จะได้ตารางกรณีมี Simple key

TABLE NAME IS : Item

ITEM_NR char 15

NAME char 25

ITEMNAME char 15

PRIMARY KEY IS: ITEM_NR

การพัฒนารีเลชั่นสกีม่า

ขั้นตอนที่1: สร้างรีเลชั่นสกีม่าในแต่ละความสัมพันธ์โดยพิจารณาที่ความเป็นยูนิค (unique) ที่ไม่เกิดจากโรลเดียว ซึ่งกรณีนี้จะถือว่าไม่มี Simple key จะพิจารณาได้จากความสัมพันธ์ของข้อมูลที่มีความสัมพันธ์กันแบบ many-to-many แล้วทำการกำหนดคีย์หลักจากความสัมพันธ์

ขั้นตอนที่2: ให้ทำการตรวจสอบความสัมพันธ์ที่เหลือโดยพิจารณาจากการแสดงความเป็นยูนิค (unique) จากโรลเดียว ซึ่งกรณีนี้จะถือว่ามี Simple key จะพิจารณาได้จากความสัมพันธ์ของข้อมูลที่ความสัมพันธ์กับแบบ many-to-one หรือ one-to-many หรือ one-to-one นำความสัมพันธ์ทั้งหมดมาระบุโรลที่แสดงการกระทำของชนิดเอนติตี้จากที่เดียวกัน ดังนั้นกลุ่มความสัมพันธ์เหล่านี้จะใช้มีคีย์หลักจากชนิดเอนนติตี้นี้

ขั้นตอนที่3: สร้างรีเลชั่นสกีม่าในแต่ละความสัมพันธ์ที่เหลือและกำหนดความสัมพันธ์มาเป็นคีย์หลัก

ขั้นตอนที่4: ทำการเคลื่อนย้ายความซ้ำซ้อนที่เกิดขึ้นจากขั้นตอนที่2 ซึ่งเป็นการเกี่ยวพันของความสัมพันธ์ที่เกิดมากกว่าหนึ่งยูนิคจากโรลเดียว

ขั้นตอนที่5: ตรวจสอบคีย์หลักในแต่ละรีเลชั่นสกีม่า หากมีคีย์หลักในรีเลชั่นสกีม่าใด ๆ ซ้ำกับคีย์หลักในรีเลชั่นสกีม่าอื่นและคีย์ที่เป็นพาร์เชียลคีย์ไม่ซ้ำกับคีย์ใดๆ ในรีเลชั่นสกีม่าอื่นแสดงว่ารีเลชั่นสกีม่านั้น ๆ แสดงถึงการเกิดชนิดเอนติตี้แบบ weak

ขั้นตอนที่6: หากฟอเรนท์คีย์ใด ๆ ได้ทำการอ้างอิงถึงคีย์หลักภายในรีเลชั่นสกีม่าเดียวกันแล้วแสดงว่าความสัมพันธ์ของรีเลชั่นสกีม่านั้น ๆ เกิดการรีเคอร์ซีฟ (recursive)

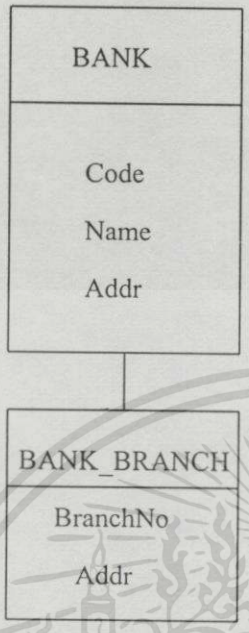
ตัวอย่าง การแมปปิ้ง กรณีเกิด weak เป็นรีเลชั่นสกีม่า

เริ่มจากได้ตารางข้อมูลที่ประกอบด้วยแอทริบิ้วต่างๆ ที่มีความสัมพันธ์ในแบบรีเลชั่นสกีม่าของ BANK และ BANK_BRANCH

รีเลชั่นสกีม่าของ BANK คือ BANK(Code , Name , Addr) มีคีย์หลักคือ Code

รีเลชั่นสกีม่าของ BANK_BRANCH คือ BANK_BRANCH(Code, BranchNo, Addr) มีคีย์หลักคือ Code และ BranchNo

ตัวอย่าง การแมปปิ้ง กรณีเกิดชนิดเอนติตี้แบบ weak เป็นสกีม่ากราฟ



ตัวอย่าง การแมปปิ้ง กรณีเกิดชนิดเอนติตี้แบบ weak เป็น ODL

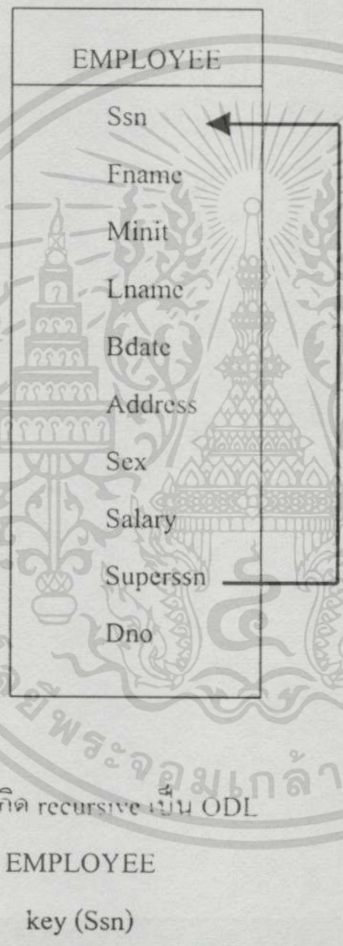
```
interface  BANK
           key (Code)
           {
              attribute  number   Code;
              attribute  char     Name;
              attribute  char     Addr;
             relationship Set<BANK_BRANCH> has
                                  inverse  BANK_BRANCH :: of;
           }

interface  BANK_BRANCH
           {
              attribute  number   BranchNo;
              attribute  char     Addr;
              relationship  Set<BANK>  of
                                  inverse  BANK :: has;
           }
```

ตัวอย่าง การแมปปิ้ง กรณีเกิด recursive เป็นรีเลชั่นสกีม่า

เริ่มจากได้ตารางข้อมูลที่ประกอบด้วยแอทริบิ้วต่างๆ ที่มีความสัมพันธ์ในแบบรีเลชั่นสกีม่าของ EMPLOYEE

รีเลชั่นสกีม่าของ EMPLOYEE คือ EMPLOYEE(<u>Ssn</u>, Fname, Minit, Lname, Bdate, Address, Sex, Salary, Superssn, Dno) มีคีย์หลักคือ Ssn และฟอร์เรนคีย์คือ Superssn

ตัวอย่าง การแมปปิ้ง กรณีเกิด recursive เป็นสกีม่ากราฟ



ตัวอย่าง การแมปปิ้ง กรณีเกิด recursive เป็น ODL

```
interface EMPLOYEE
        key (Ssn)
        { Attribute number Ssn;
          Attribute char Fname;
          Attribute char Minit;
          Attribute char Lname;
          Attribute char Bdate;
          Attribute char Address;
          Attribute char Sex;
          Attribute char Salary;
```

```
        Attribute char Superssn;
        Attribute char Dno;
        relationship Set<Superssn> has
                inverse Superssn :: of;
        }

interface Superssn
    {
        relationship EMPLOYEE supervisor;
        relationship EMPLOYEE supervisee;
    }
```

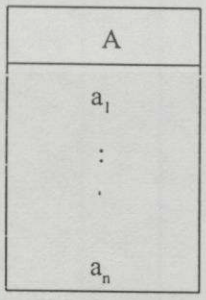
7.3.5 ขั้นตอนและวิธีการทำ Schema Graphs

ในงานวิจัยขั้นต่อไปหลังจากที่ได้พัฒนาอยู่ในรูปของ 5NF Relational Database Schema แล้ว เราจะนำเอาผลลัพธ์ที่ได้ไปพัฒนาเป็น Schema Ggraphs แล้วทำการเจนเนอเรทเป็น Object Database Definition ในรูปของ ODL ซึ่งเราจะเรียกว่าเป็นภาษานิยามฐานข้อมูลเชิงวัตถุ โดยที่ก่อนที่จะทำให้ได้เป็น ODL นี้ เราจะอธิบายแนวคิดที่ได้พัฒนาขึ้นมาโดยเริ่มจากการสร้างเป็น Schema Graphs ก่อน จะขอเริ่มอธิบายได้ว่า เราจะอาศัยความสัมพันธ์ที่เกิดจากการสร้างฐานข้อมูลเชิงสัมพันธ์ในเรื่องของ ไพรมารีคีย์ และ ฟอเรนคีย์ ตามที่ได้อธิบายเป็นอัลกอริธึมไว้ในบทที่ 6 ในเรื่องของสถาปัตยกรรมระบบ ในก่อนหน้านี้มาแล้ว แต่ก็จะขออธิบายขยายเพิ่มเติมถึงแนวคิดอย่างละเอียดโดยเริ่มจาก

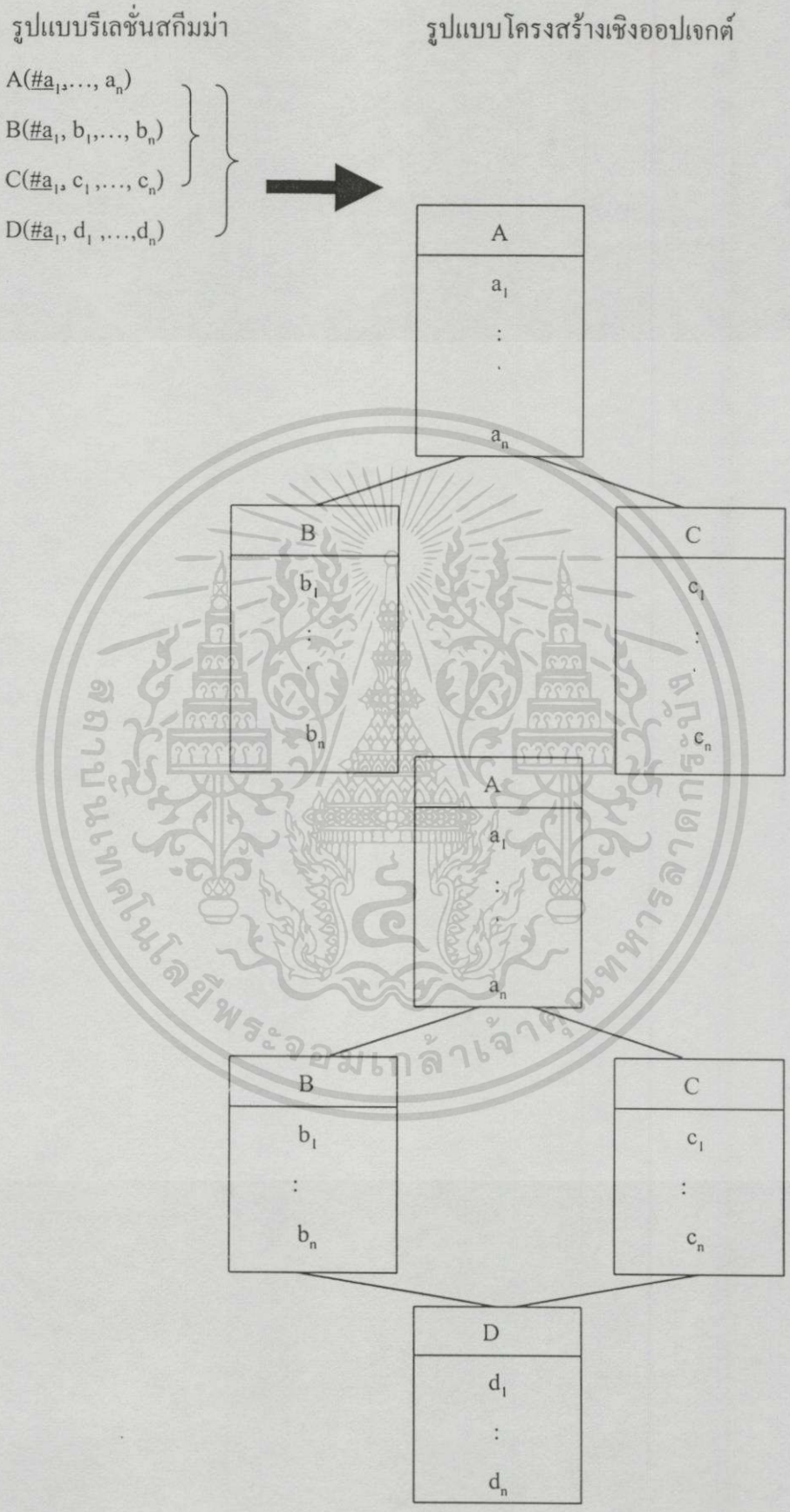
กลุ่มของรีเลชั่นสกีมม่ากับรูปแบบโครงสร้างเชิงออปเจกต์ในลักษณะเดี่ยว ๆ

รูปแบบรีเลชั่นสกีมม่า รูปแบบโครงสร้างเชิงออปเจกต์

$A(\underline{\#a_1}, \ldots, a_n)$ 



กลุ่มของรีเลชั่นสกีมม่ากับรูปแบบโครงสร้างเชิงออปเจกต์ในลักษณะ single inheritance และกลุ่มของ multiple inheritance



กลุ่มของรีเลชั่นสกีมม่ากับรูปแบบโครงสร้างเชิงออปเจกต์ในลักษณะ non-hierarchical relationship

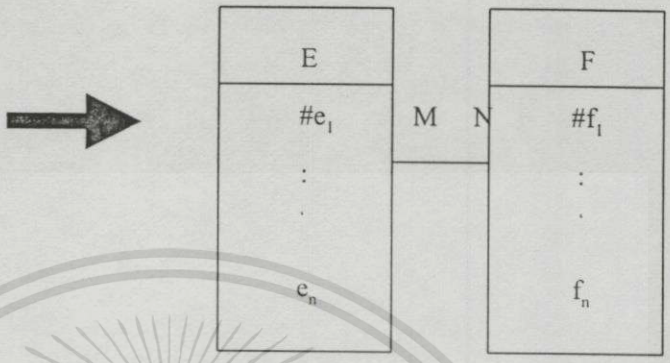
รูปแบบรีเลชั่นสกีมม่า

$E(\underline{\#e_1},\ldots, e_n)$

$F(\underline{\#f_1},\ldots,f_n)$

$G(\underline{\#e_1,\#f_1})$

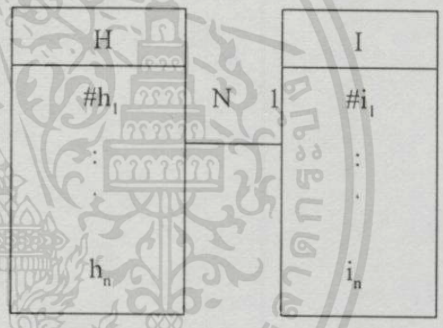
รูปแบบโครงสร้างเชิงออปเจกต์



$H(\underline{\#h_1},\ldots,h_n)$

$I(\underline{\#i_1},\ldots,i_n)$

$J(\underline{\#h_1,\#i_1})$



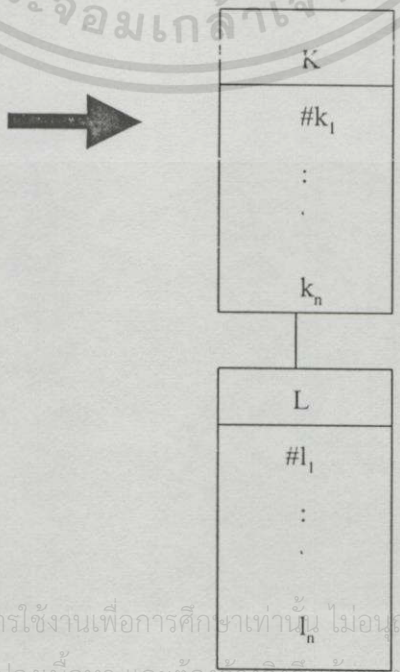
กลุ่มของรีเลชั่นสกีมม่ากับรูปแบบโครงสร้างเชิงออปเจกต์ในลักษณะ aggregation

รูปแบบรีเลชั่นสกีมม่า

$K(\underline{\#k_1},\ldots, k_n)$

$L(\underline{\#l_1},k_1,\ldots,l_n)$

รูปแบบโครงสร้างเชิงออปเจกต์



การพัฒนาสกีม่ากราฟเพื่อหาความสัมพันธ์ของกลุ่มข้อมูลเพื่อนำไปใช้ในการสร้าง ODL

ขั้นตอนที่1: รวบรวมความสัมพันธ์ของข้อมูลที่ได้จากการสร้างรีเลชั่นสกีม่า

ขั้นตอนที่2: แบ่งกลุ่มความสัมพันธ์ของข้อมูลออกเป็นกลุ่มๆ โดยอาศัยความสัมพันธ์ในระดับคีย์หลักและฟอร์เรนคีย์ โดยกลุ่มแรกนั้นจะประกอบไปด้วยกลุ่มละ 2 ชนิดเอนติตี้ที่มีความสัมพันธ์กันและในกลุ่มที่สองนั้นจะประกอบด้วยความสัมพันธ์ตั้งแต่ 2 ชนิดเอนติตี้ขึ้นไป

ขั้นตอนที่3: ออปเจกต์คลาสที่แสดงชนิดเอนติตี้แบบ weak จะไม่มีคีย์หลักภายในออปเจกต์

ขั้นตอนที่4: ออปเจกต์คลาสที่แสดงแบบ recursive จะแสดงโดยมีการชี้ระหว่างแอทริบิ้วที่เกิด recursive ภายในออปเจกต์เดียวกัน

ขั้นตอนต่อไปนี้เป็นกระบวนการคิดในการที่จะสร้างเป็นโครงสร้างเชิงออปเจกต์ โดยสมมติข้อมูลตัวอย่างขึ้นมาใช้ในขั้นตอน โดยเราจะเริ่มการทำงานดังต่อไปนี้คือ

กำหนดให้ A เป็นชนิดเอนติตี้ของ Company ที่ประกอบด้วยแอทริบิ้วดังนี้

รูปแบบ A=Company:{NAME, PHNR, PNAME}

กำหนดให้ B เป็นชนิดเอนติตี้ของ CpLanguage ที่ประกอบด้วยแอทริบิ้วดังนี้

รูปแบบ B=CpLanguage:{CLNAME, GENNR, AD}

กำหนดให้ C เป็นชนิดเอนติตี้ของ Programmer ที่ประกอบด้วยแอทริบิ้วดังนี้

รูปแบบ C=Programmer:{PNAME, CODE, ADDRESS, PHNR, NAME}

กำหนดให้ D เป็นชนิดเอนติตี้ของ SwPackage ที่ประกอบด้วยแอทริบิ้วดังนี้

รูปแบบ D=SwPackage:{SW, M_NUM, TITLE}

กำหนดให้ E เป็นชนิดเอนติตี้ของ many_ to_ many1 ที่ประกอบด้วยแอทริบิ้วดังนี้

รูปแบบ E=many_to_many1:{PNAME, CLNAME}

กำหนดให้ F เป็นชนิดเอนติตี้ของ owns ที่ประกอบด้วยแอทริบิ้วดังนี้

รูปแบบ F=owns:{COMKIND, NAME, NR}

กำหนดให้ G เป็นชนิดเอนติตี้ของ works ที่ประกอบด้วยแอทริบิ้วดังนี้

รูปแบบ G=works:{PNAME, SW, CLNAME}

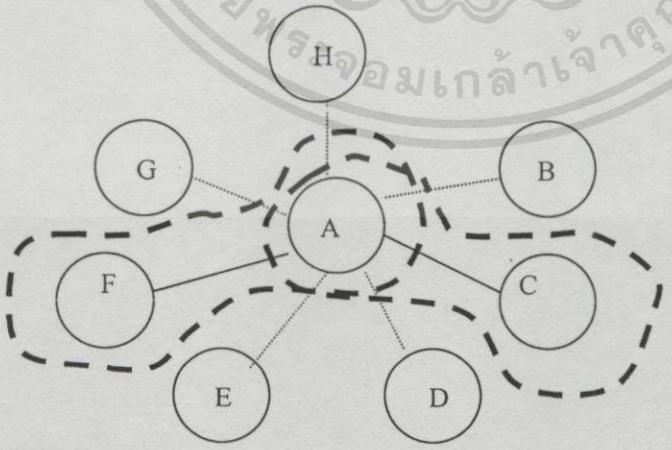
กำหนดให้ H เป็นชนิดเอนติตี้ของ ComputerKind ที่ประกอบด้วยแอทริบิ้วดังนี้

รูปแบบ H=ComputerKind:{ }

ความสัมพันธ์ของระหว่างชนิดเอนติตี้ที่เกิดขึ้นในกรณีตัวอย่างของข้อมูลข้างต้นพร้อมทั้งแสดงถึงความสัมพันธ์ของชนิดเอนติตี้ต่างๆ ที่อยู่ในแนวคิดตามหมายเลขขั้นตอนของอัลกอริธึมที่ได้พัฒนาในบทที่ 6 ตามที่ได้กล่าวไว้ เราจะกำหนดให้ชนิดเอนติตี้ที่มีเส้นประเชื่อมติดต่อแสดงว่าไม่มีความสัมพันธ์ระหว่างชนิดเอนติตี้นั้น ๆ แนวความคิดจะใช้ไพรมารีคีย์ และ ฟอเรนคีย์ โดยจะแบ่งเป็นสาม ส่วนคือ ส่วนที่มีไพรมารีคีย์เพียงหนึ่งตัว ส่วนที่สองจะมีมากกว่าหนึ่งตัว และส่วนที่สามจะผสมระหว่างมีไพรมารีคีย์เพียงหนึ่งตัวและมีมากกว่าหนึ่งตัว

กรณีมี primary key /foreign key มีเพียง 1

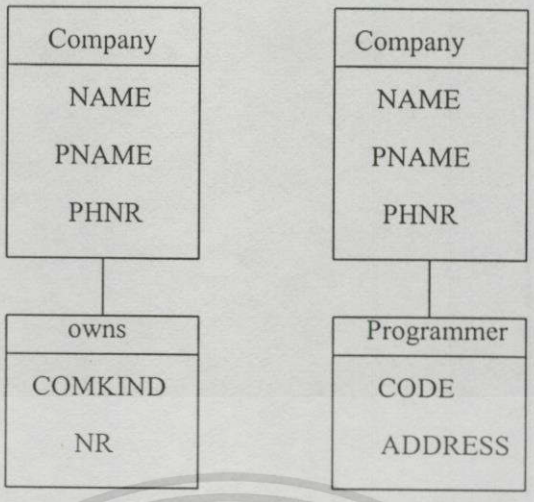
ในขั้นตอนนี้จะเริ่มจากให้หาความสัมพันธ์ของชนิดเอนติตี้ A ดังรูปที่ 7.12 ที่มีความสัมพันธ์กับชนิดเอนติตี้อื่น ๆ ไหนบ้าง (ในลักษณะเพียง 2 เอนติตี้ที่กระทำระหว่างกันเท่านั้น) ซึ่งก่อนที่จะได้ทราบว่ามีชนิดเอนติตี้ไหนบ้างมีความสัมพันธ์กันนั้น จะต้องทำการเปรียบเทียบดูที่ความสัมพันธ์ในระดับของแอทริบิ้วในแต่ละชนิดเอนติตี้ว่าเป็นไปกฎที่ได้นิยามขึ้นมาในข้อไหน จากกรณีนี้จะพบว่าชนิดเอนติตี้ A มีความสัมพันธ์กับชนิดเอนติตี้ C และชนิดเอนติตี้ F โดยเป็นไปตามกฎนิยามในข้อที่ 4



A=Company:{NAME, PHNR, PNAME}

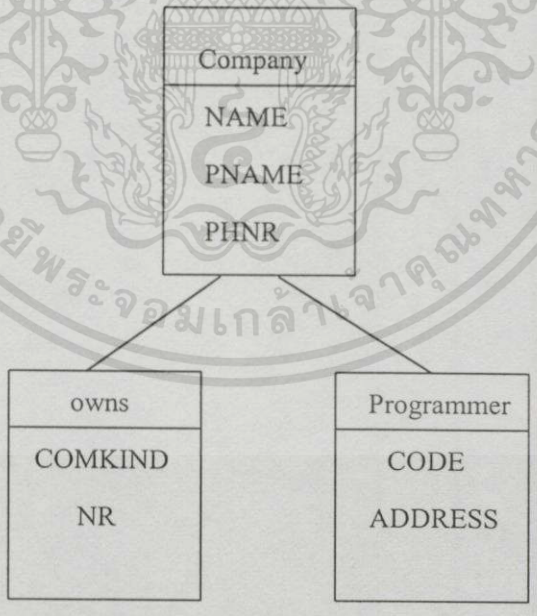
C=Programmer:{PNAME, CODE, ADDRESS, PHNR, NAME}

F=owns:{COMKIND, NAME, NR}



รูปที่ 7.12 แสดงความสัมพันธ์ของชนิดเอนติตี้ A

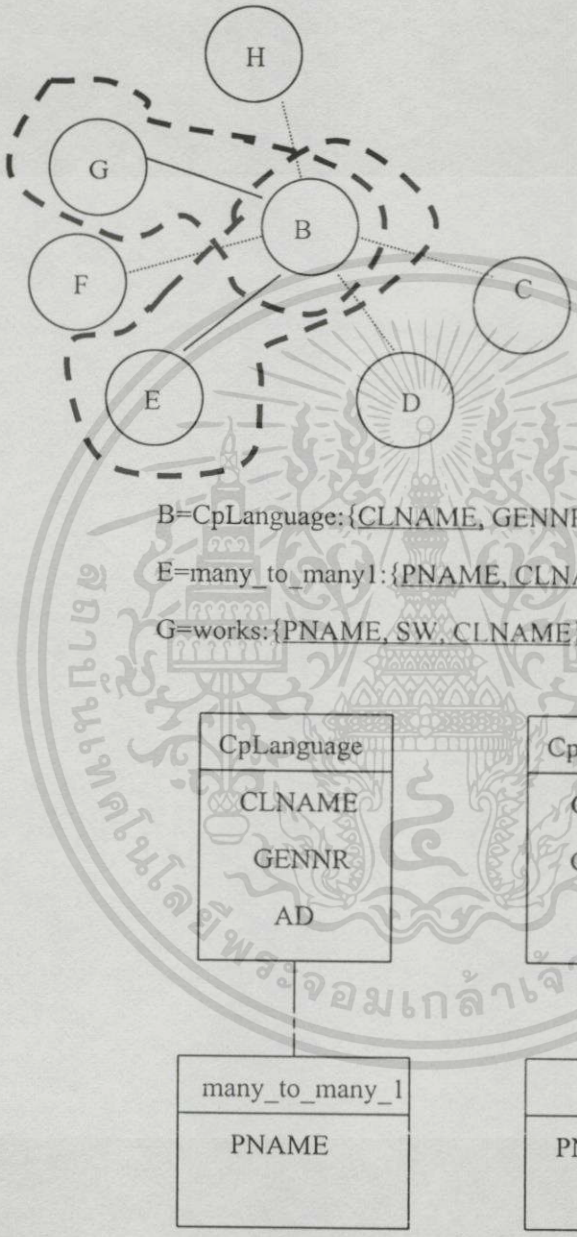
ดังนั้นเราสามารถยุบการทำงานจากกรณีที่เกิดขึ้นจากความสัมพันธ์ระหว่างชนิดเอนติตี้A กับชนิดเอนติตี้ C และชนิดเอนติตี้ A กับชนิดเอนติตี้ F ซึ่งเป็นไปตามกฎนิยามข้อที่ 4 นั้น ให้เป็นไปตามกฎข้อนิยามที่ 2 ได้โดยรวมความสัมพันธ์ระหว่างชนิดเอนติตี้ทั้งหมดเป็น A สัมพันธ์ทั้ง C และ F ดังจะแสดงรูป 7.13ของโครงสร้างเชิงออปเจกต์ใหม่ได้ดังนี้คือ



รูปที่ 7.13 แสดงรูปแบบใหม่ของความสัมพันธ์ชนิดเอนติตี้ A

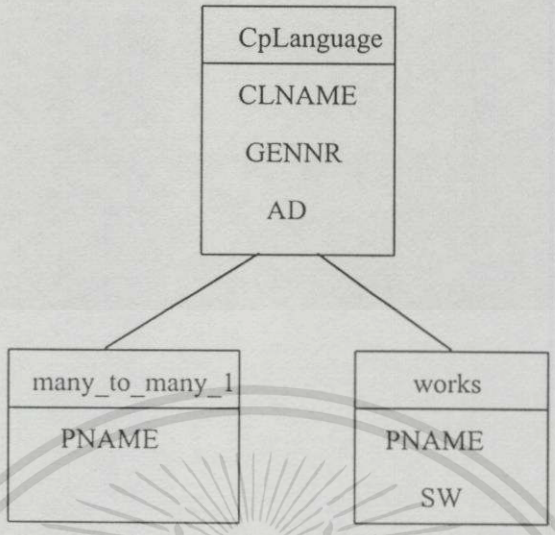
ในขั้นตอนต่อไปจะเริ่มจากให้หาความสัมพันธ์ของชนิดเอนติตี้ B ดังรูปที่ 7.14 ที่มีความสัมพันธ์กับชนิดเอนติตี้อื่น ๆ ไหนบ้าง (ในลักษณะเพียง 2 ชนิดเอนติตี้ที่กระทำระหว่างกันเท่านั้น) ซึ่งก่อนที่จะได้ทราบว่ามีชนิดเอนติตี้ไหนบ้างมีความสัมพันธ์กันนั้น จะต้องทำการเปรียบเทียบดูที่

ความสัมพันธ์ในระดับของแอทริบิ้วในแต่ละชนิดเอนติตี้ว่าเป็นไปกฎที่ได้นิยามขึ้นมาในข้อไหน จากกรณีนี้จะพบว่าชนิดเอนติตี้ B มีความสัมพันธ์กับชนิดเอนติตี้ E และชนิดเอนติตี้ G โดยเป็นไปตามกฎนิยามในข้อที่ 4



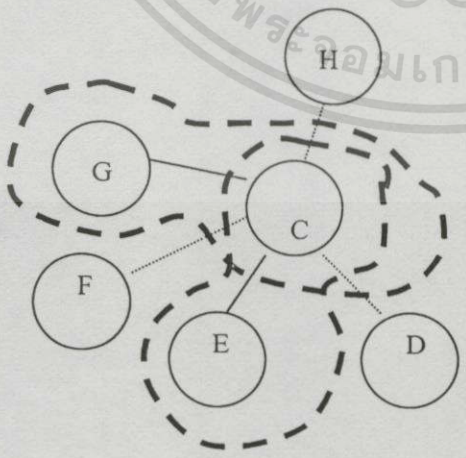
รูปที่ 7.14 แสดงความสัมพันธ์ของชนิดเอนติตี้ B

ดังนั้นเราสามารถยุบการทำงานจากกรณีที่เกิดขึ้นจากความสัมพันธ์ระหว่างชนิดเอนติตี้ B กับชนิดเอนติตี้ E และชนิดเอนติตี้ B กับชนิดเอนติตี้ G ซึ่งเป็นไปตามกฎนิยามข้อที่ 4 นั้น ให้เป็นไปตามกฎข้อนิยามที่ 2 ได้โดยรวมความสัมพันธ์ระหว่างชนิดเอนติตี้ทั้งหมดเป็น B สัมพันธ์ทั้ง E และ G ดังจะแสดงรูปที่ 7.15 ของโครงสร้างเชิงออปเจกต์ใหม่ได้ดังนี้คือ



รูปที่ 7.15 แสดงรูปแบบใหม่ของความสัมพันธ์ชนิดเอนติตี้ B

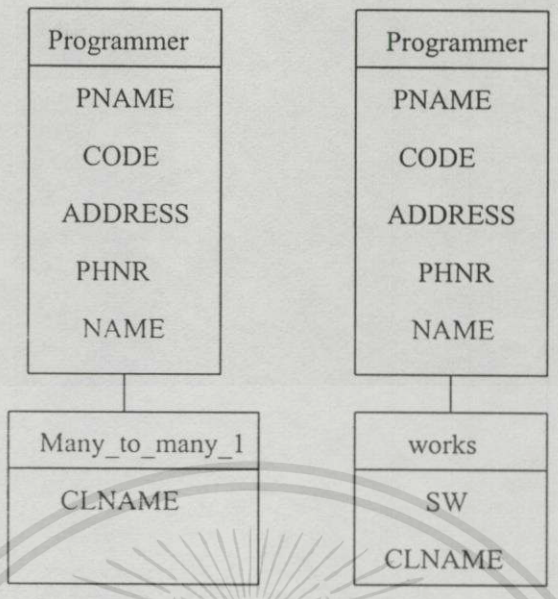
ในขั้นตอนต่อไปจะเริ่มจากให้หาความสัมพันธ์ของชนิดเอนติตี้ C ดังรูปที่ 7.16 ที่มีความสัมพันธ์กับชนิดเอนติตี้อื่น ๆ ไหนบ้าง (ในลักษณะเพียง 2 ชนิดเอนติตี้ที่กระทำระหว่างกันเท่านั้น) ซึ่งก่อนที่จะได้ทราบว่ามีชนิดเอนติตี้ไหนบ้างมีความสัมพันธ์กันนั้น จะต้องทำการเปรียบเทียบดูที่ความสัมพันธ์ในระดับของแอทริบิ้วในแต่ละชนิดเอนติตี้ว่าเป็นไปกฎที่ได้นิยามขึ้นมาในข้อไหน จากกรณีนี้จะพบว่าชนิดเอนติตี้ C มีความสัมพันธ์กับชนิดเอนติตี้ E และชนิดเอนติตี้ G โดยเป็นไปตามกฎนิยามในข้อที่ 4



C=Programmer:{PNAME, CODE, ADDRESS, PHNR, NAME}

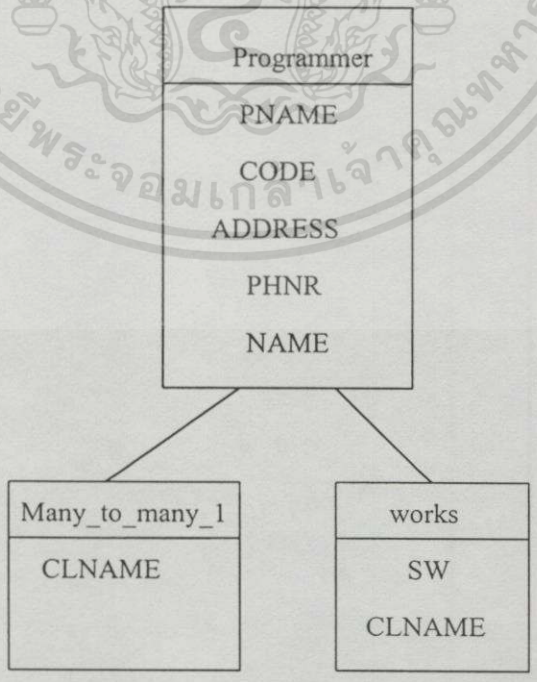
E=many_to_many1:{PNAME, CLNAME}

G=works:{PNAME, SW, CLNAME}



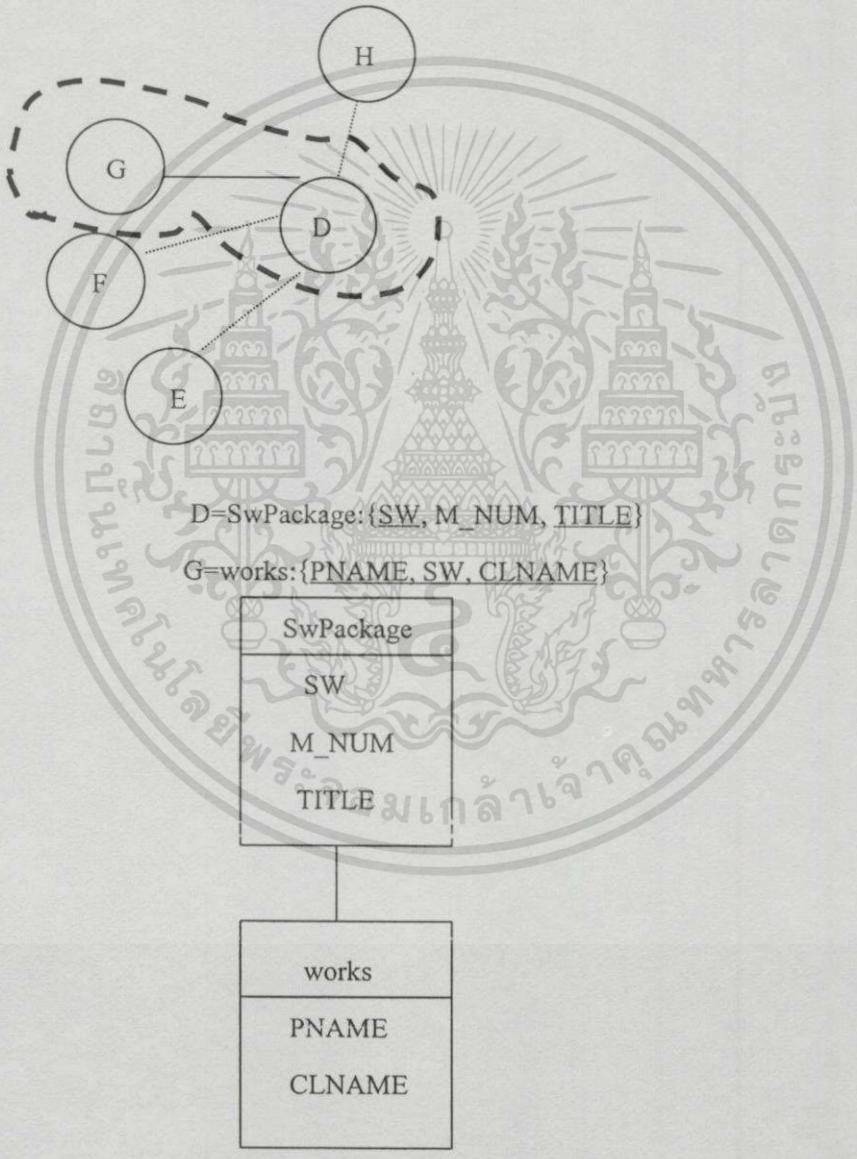
รูปที่ 7.16 แสดงความสัมพันธ์ของชนิดเอนติตี้ C

ดังนั้นเราสามารถยุบการทำงานจากกรณีที่เกิดขึ้นจากความสัมพันธ์ระหว่างชนิดเอนติตี้ C กับชนิดเอนติตี้ E และชนิดเอนติตี้ C กับชนิดเอนติตี้ G ซึ่งเป็นไปตามกฎนิยามข้อที่ 4 นั้น ให้เป็นไปตามกฎข้อนิยามที่ 2 ได้โดยรวมความสัมพันธ์ระหว่างชนิดเอนติตี้ทั้งหมดเป็น C สัมพันธ์ทั้ง E และ G ดังจะแสดงรูปที่ 7.17 ของโครงสร้างเชิงออปเจกต์ใหม่ได้ดังนี้คือ



รูปที่ 7.17 แสดงรูปแบบใหม่ของความสัมพันธ์เอนติตี้ C

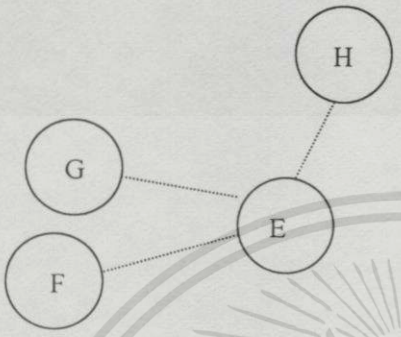
ในขั้นตอนต่อไปจะเริ่มจากให้หาความสัมพันธ์ของชนิดเอนติตี้ D ดังรูปที่ 7.18 ที่มีความสัมพันธ์กับชนิดเอนติตี้อื่น ๆ ไหนบ้าง (ในลักษณะเพียง 2 ชนิดเอนติตี้ที่กระทำระหว่างกันเท่านั้น) ซึ่งก่อนที่จะได้ทราบว่ามีชนิดเอนติตี้ไหนบ้างมีความสัมพันธ์กันนั้น จะต้องทำการเปรียบเทียบดูที่ความสัมพันธ์ในระดับของแอทริบิ้วในแต่ละชนิดเอนติตี้ว่าเป็นไปกฎที่ได้นิยามขึ้นมาในข้อไหน จากกรณีนี้จะพบว่าชนิดเอนติตี้ D มีความสัมพันธ์กับชนิดเอนติตี้ G โดยเป็นไปตามกฎนิยามในข้อที่ 4



รูปที่ 7.18 แสดงความสัมพันธ์ของชนิดเอนติตี้ D

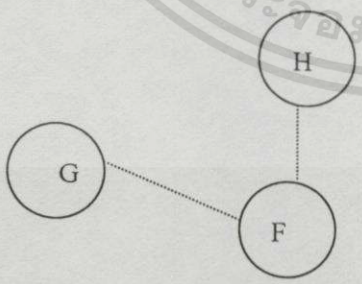
ในขั้นตอนต่อไปจะเริ่มจากให้หาความสัมพันธ์ของชนิดเอนติตี้ E ดังรูปที่ 7.19 ที่มีความสัมพันธ์กับชนิดเอนติตี้อื่น ๆ ไหนบ้าง(ในลักษณะเพียง 2 ชนิดเอนติตี้ที่กระทำระหว่างกันเท่านั้น) ซึ่งก่อนที่จะได้ทราบว่ามีชนิดเอนติตี้ไหนบ้างมีความสัมพันธ์กันนั้น จะต้องทำการเปรียบเทียบดูที่

ความสัมพันธ์ในระดับของแอทริบิ้วในแต่ละเอนติตี้ว่าเป็นไปกฎที่ได้นิยามขึ้นมาในข้อไหน จากกรณีนี้จะพบว่าชนิดเอนติตี้ E มีความสัมพันธ์กับชนิดเอนติตี้ B และชนิดเอนติตี้ C ที่ได้กล่าวมาแล้ว ซึ่งในขั้นตอนนี้จะไม่มีการทำกระทำซ้ำอีก ดังนั้นในขั้นนี้ จะถือได้ว่าชนิดเอนติตี้ E ไม่มีความสัมพันธ์กับชนิดเอนติตี้ใด ๆ เลย



รูปที่ 7.19 แสดงความสัมพันธ์ของชนิดเอนติตี้ E

ในขั้นตอนต่อไปจะเริ่มจากให้หาความสัมพันธ์ของชนิดเอนติตี้ F ดังรูปที่ 7.20 ที่มีความสัมพันธ์กับชนิดเอนติตี้อื่น ๆ ไหนบ้าง (ในลักษณะเพียง 2 ชนิดเอนติตี้ที่กระทำระหว่างกันเท่านั้น) ซึ่งก่อนที่จะได้ทราบว่ามีชนิดเอนติตี้ไหนบ้างมีความสัมพันธ์กันนั้น จะต้องทำการเปรียบเทียบดูที่ความสัมพันธ์ในระดับของแอทริบิ้วในแต่ละชนิดเอนติตี้ว่าเป็นไปกฎที่ได้นิยามขึ้นมาในข้อไหน จากกรณีนี้จะพบว่าชนิดเอนติตี้ F มีความสัมพันธ์กับชนิดเอนติตี้ A ที่ได้กล่าวมาแล้ว ซึ่งในขั้นตอนนี้จะไม่มีการทำกระทำซ้ำอีก ดังนั้นในขั้นนี้ จะถือได้ว่าชนิดเอนติตี้ F ไม่มีความสัมพันธ์กับชนิดเอนติตี้ใด ๆ เลย



รูปที่ 7.20 แสดงความสัมพันธ์ของชนิดเอนติตี้ F

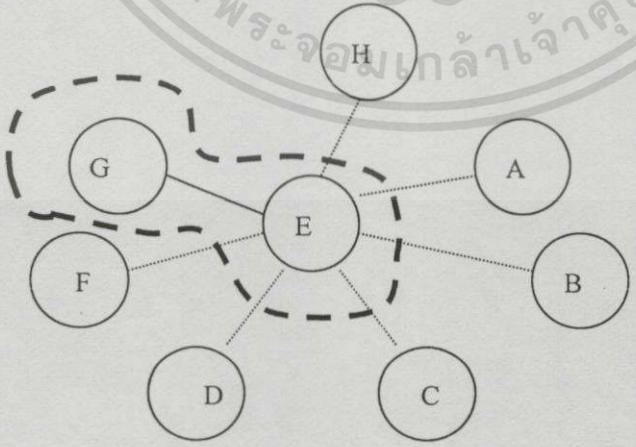
ในขั้นตอนต่อไปจะเริ่มจากให้หาความสัมพันธ์ของชนิดเอนติตี้ G ดังรูปที่ 7.21 ที่มีความสัมพันธ์กับชนิดเอนติตี้อื่น ๆ ไหนบ้าง (ในลักษณะเพียง 2 ชนิดเอนติตี้ที่กระทำระหว่างกันเท่านั้น) ซึ่งก่อนที่จะได้ทราบว่ามีชนิดเอนติตี้ไหนบ้างมีความสัมพันธ์กันนั้น จะต้องทำการเปรียบเทียบดูที่ความสัมพันธ์ในระดับของแอทริบิ้วในแต่ละเอนติตี้ว่าเป็นไปกฎที่ได้นิยามขึ้นมาในข้อไหน จาก

กรณีนี้จะพบว่าชนิดเอนติตี้ G มีความสัมพันธ์กับชนิดเอนติตี้ B ชนิดเอนติตี้ C และชนิดเอนติตี้ D ที่ได้กล่าวมาแล้ว ซึ่งในขั้นตอนนี้จะไม่มีการทำกระทำซ้ำอีก ดังนั้นในขั้นนี้ จะถือได้ว่าชนิดเอนติตี้ D ไม่มีความสัมพันธ์กับชนิดเอนติตี้ใด ๆ เลย และนอกจากนี้ชนิดเอนติตี้ของ H ก็ไม่มีความสัมพันธ์กับชนิดเอนติตี้ใด ๆ เช่นเดียวกัน

รูปที่ 7.21 แสดงความสัมพันธ์ของชนิดเอนติตี้ G

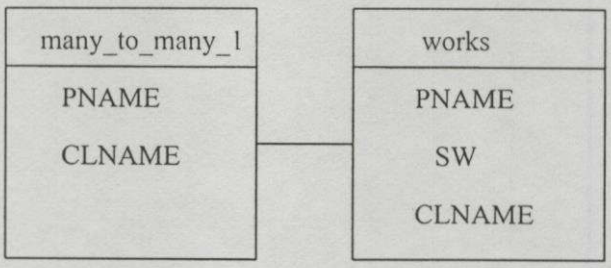
กรณี primary keys มากกว่า 1

ในขั้นตอนต่อไปจะเริ่มจากให้หาความสัมพันธ์ของชนิดเอนติตี้ E ดังรูปที่ 7.22 ที่มีความสัมพันธ์กับชนิดเอนติตี้อื่น ๆ ไหนบ้าง (ในลักษณะเพียง 2 ชนิดเอนติตี้ที่กระทำระหว่างกันเท่านั้น) ซึ่งก่อนที่จะได้ทราบว่ามีชนิดเอนติตี้ไหนบ้างมีความสัมพันธ์กันนั้น จะต้องทำการเปรียบเทียบดูที่ความสัมพันธ์ในระดับของแอทริบิวในแต่ละชนิดเอนติตี้ว่าเป็นไปกฎที่ได้นิยามขึ้นมาในข้อไหน จากกรณีนี้จะพบว่าชนิดเอนติตี้ E มีความสัมพันธ์กับชนิดเอนติตี้ A และชนิดเอนติตี้ G โดยเป็นไปตามกฎนิยามในข้อที่ 3 และข้อที่ 2 ตามลำดับ



E=many_to_many1:{PNAME, CLNAME}

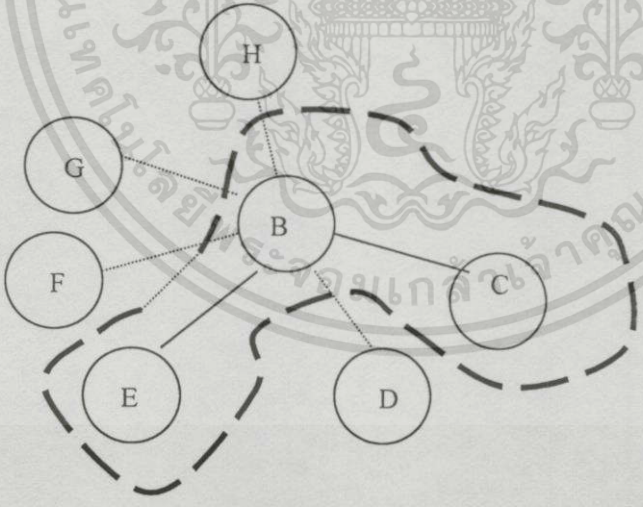
G=works:{PNAME, SW, CLNAME}



รูปที่ 7.22 แสดงความสัมพันธ์ของชนิดเอนติตี้ E

กรณี primary key/ foreign key จำนวน 1 หรือมากกว่า

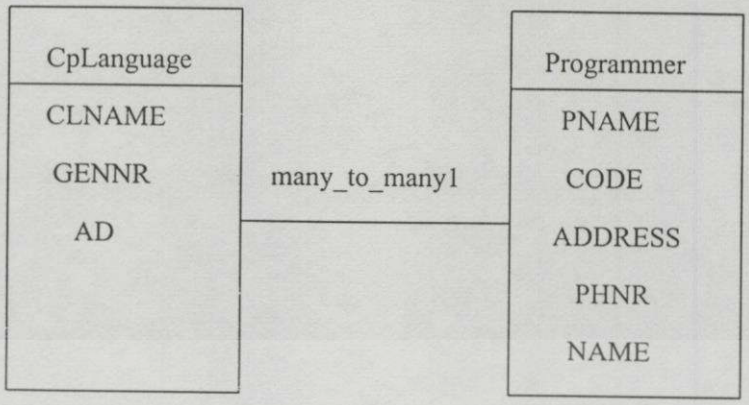
ในขั้นตอนต่อไปจะเริ่มจากให้หาความสัมพันธ์ของชนิดเอนติตี้ B ดังรูปที่ 7.23 ที่มีความสัมพันธ์กับชนิดเอนติตี้อื่น ๆ ไหนบ้าง (ในลักษณะเพียง 3 ชนิดเอนติตี้ที่กระทำระหว่างกันเท่านั้น) ซึ่งก่อนที่จะได้ทราบว่ามีชนิดเอนติตี้ไหนบ้างมีความสัมพันธ์กันนั้น จะต้องทำการเปรียบเทียบดูที่ความสัมพันธ์ในระดับของแอทริบิ้วในแต่ละชนิดเอนติตี้ว่าเป็นไปกฎที่ได้นิยามขึ้นมาในข้อไหน จากกรณีนี้จะพบว่าชนิดเอนติตี้ B มีความสัมพันธ์กับชนิดเอนติตี้ C และชนิดเอนติตี้ E โดยเป็นไปตามกฎนิยามในข้อที่ 3



B=CpLanguage:{CLNAME, GENNR, AD}

C=Programmer:{PNAME, CODE, ADDRESS, PHNR, NAME}

E=many_to_many1:{PNAME, CLNAME}



รูปที่ 7.23 แสดงความสัมพันธ์ของชนิดเอนติตี้ B

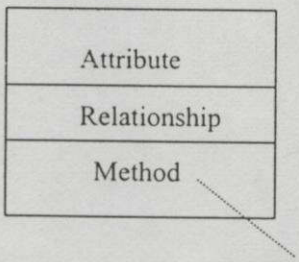
7.3.6 ขั้นตอนและวิธีการพัฒนา ODL

จากขั้นตอนทั้งหมดที่ได้กล่าวมานี้ซึ่งเป็นวิธีการหาความสัมพันธ์ของข้อมูลในลักษณะสกีมม่ากราฟ ก่อนที่จะนำไปทำการสร้างเป็น ODL ในขั้นตอนสุดท้าย สำหรับการสร้างเป็น ODL นั้นเราได้ทำการพัฒนารูปแบบแนวคิดจากเป็นสกีมม่ากราฟให้เป็นไปตามรูปแบบของ ODL ตามที่ได้กล่าวไว้ในบทที่ 5 นั้น เราจะอธิบายขั้นตอนและแนวคิดได้ดังต่อไปนี้คือ

ผู้วิจัยได้สร้างขั้นตอนการพัฒนาจาก Schema Graphs ให้อยู่ในรูปของ ODL โดยจะแบ่งรูปแบบการพัฒนาออกได้เป็น 2 รูปแบบดังนี้คือ

1. รูปแบบแนวคิดในการใช้ 5NF Relational Database , Database Trigger และ Store Procedures ในการกำหนด Attributes และ Methods ของแต่ละออปเจกต์คลาส
2. รูปแบบแนวคิดในการใช้ Schema Graphs ในการกำหนด Relationships ระหว่างออปเจกต์คลาส

ส่วนประกอบ ODL



จากรูปแบบที่ได้ใน 5NF , Database Trigger และ Store Procedures เราสามารถแสดงการสร้างเป็น Attributes ของรูปแบบ ODL ตามลำดับขั้นตอนง่าย ๆ ดังต่อไปนี้คือ

ขั้นตอนที่ 1. นำข้อมูลจากการพัฒนาตารางฐานข้อมูลเชิงสัมพันธ์ที่ได้มาใช้ดังตัวอย่างข้อมูล

B=CpLanguage:{CLNAME, GENNR, AD}

C=Programmer:{PNAME, CODE, ADDRESS, PHNR, NAME}

E=many_to_many1:{PNAME, CLNAME}

จะอธิบายโดยขอใช้เพียงตาราง 5NF ของ CpLanguage เพียงตารางเดียวเท่านั้น ณ ในที่นี้

TABLE NAME IS : CpLanguage

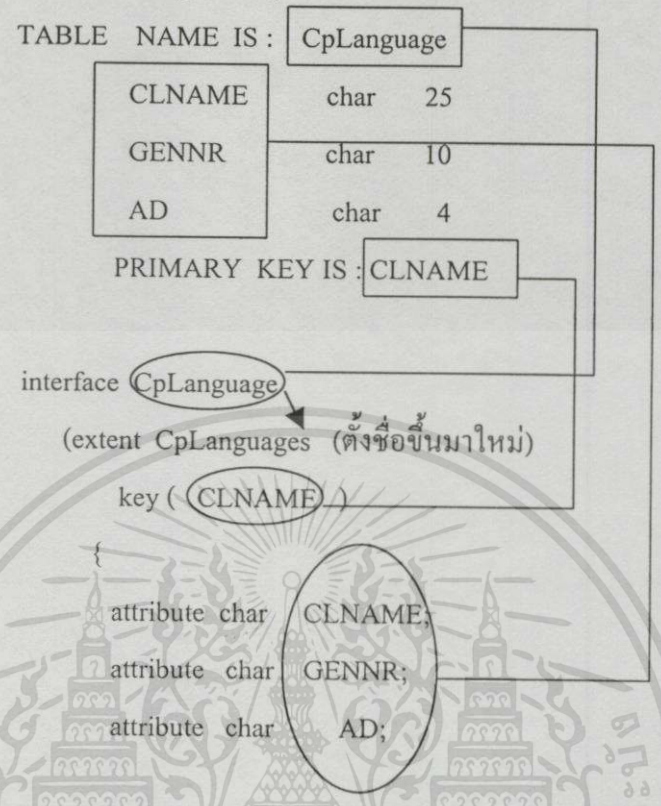
CLNAME	char	25
GENNR	char	10
AD	char	4

PRIMARY KEY IS : CLNAME

ขั้นตอนที่ 2. เขียนตามรูปแบบของ ODL จากข้อมูลที่ได้ในขั้นตอนที่ 1 จะได้ดังนี้คือ

```
interface CpLanguage
 (extent CpLanguages
   key (CLNAME)
  {
   attribute char CLNAME;
   attribute char GENNR;
   attribute char AD;
```

ในขั้นตอนที่2 นี้รูปแบบของการใช้ข้อมูลที่มีอยู่จะเป็นลักษณะดังนี้คือ



ขั้นตอนที่ 3. ใช้ข้อมูลที่ได้จากการหาความสัมพันธ์ในลักษณะ Schema Graphs เพื่อสร้างเป็น relationship ระหว่างออปเจกต์คลาสที่เกิดขึ้น

Set แสดงถึงความเป็น many fluent เป็นโรลของ CpLanguage

relationship Set<Programmer> fluent

inverse Programmer ::fluentin;

fluentin เป็นโรลของ Programmer

ขั้นตอนที่ 4. ใช้ข้อมูลจาก Store Procedures หรือ Database Trigger สร้างเป็นลักษณะของ Method ซึ่งในกรณีนี้อาจจะมีก็ได้ เพราะเป็นการแสดงถึงการทำงานของฟังก์ชั่นที่เกิดขึ้นในออปเจกต์คลาส

แสดงข้อผิดพลาด

checkCompany_Owns(in Set<owns>, out Set<Company>) raise(not company owns);

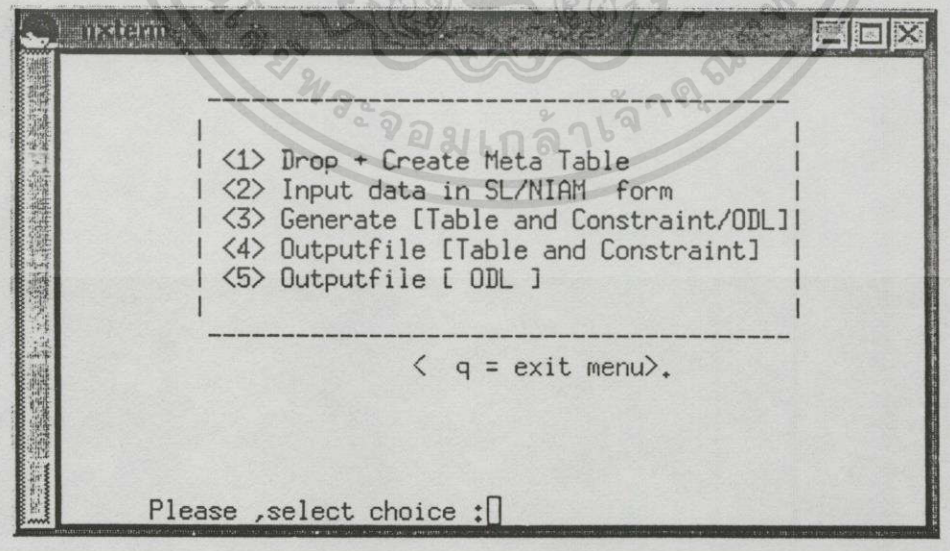
เป็นชื่อของฟังก์ชั่น ข้อมูลนำเข้าและนำออก

บทที่ 8
การทดสอบและการใช้งานระบบ

การทดสอบและใช้งานระบบนี้ เนื่องจากการทำงานของระบบทั้งหมดจะเกี่ยวข้องกับการใช้งานบนระบบปฏิบัติการลีนุกซ์ ซึ่งในบางครั้งจะพบว่ามีความแตกต่างไม่ว่าจะเป็นหน้าจอหรือแม้แต่ชุดคำสั่งต่างๆ ที่แตกต่างจากการใช้งานในระบบวินโดวส์อย่างเห็นได้ชัดเจน การทำงานวิจัยในครั้งนี้จะทำการติ้งตั้งตัวเซอร์เวอร์และตัวไคลเอนต์อยู่บนเครื่องในตัวเดียวกัน รูปแบบการและขั้นตอนการใช้งานระบบและผลการทดสอบสามารถอธิบายได้ตามขั้นตอนต่างๆ ดังข้างล่างต่อไปนี้

8.1 วิธีการเริ่มใช้งาน

- ให้ทำการบูตระบบปฏิบัติการลีนุกซ์
- ใส่ลอกอินและพาร์สเวิร์ดในการใช้งาน
- เปิดระบบจัดการฐานข้อมูลออราเคิล
- เปิดหน้าจอหลักการใช้งานโปรแกรมภาษา SL/NIAM (ดังรูปที่ 8.1)



รูปที่ 8.1 แสดงหน้าจอหลักของการใช้งาน

จากหน้าจอหลักของการทำงานสามารถอธิบายการทำงานได้ดังต่อไปนี้คือ

Drop + Create Meta Table : เป็นการสร้างตารางฐานข้อมูลขึ้นมาใหม่เพื่อรองรับการทำงาน ซึ่งจะทำทุกครั้งเมื่อมีการเริ่มต้นใช้งาน และการทำงานนี้เพื่อไม่ให้มีข้อมูลเก่า ๆ เหลืออยู่

Input data in SL/NIAM form : เป็นการรับข้อมูลในรูปประโยคภาษา SL/NIAM แล้วทำการจัดเก็บข้อมูลลงในตารางฐานข้อมูลโดยอัตโนมัติ

Generate [Table and Constraint/ODL] : เป็นกระบวนการพัฒนา Table Constraints และ ODL

Outputfile [Table and Constraint] : เป็นการเปิดดูไฟล์ผลลัพธ์ของระบบที่ได้ทำวิจัย ซึ่งไฟล์ที่ได้นี้ชื่อ outputfile.dat จะแสดงรายละเอียดของ Table และคอนสเตรนท์ที่ได้ทำการพัฒนาขึ้นมา

Outputfile [ODL] : เป็นการเปิดดูไฟล์ผลลัพธ์ของระบบที่ได้ทำวิจัย ซึ่งไฟล์ที่ได้นี้ชื่อ odlfile.dat จะแสดงรายละเอียดของ ODL ที่ได้ทำการพัฒนาขึ้นมา

8.2 แสดงตัวอย่างชุดข้อมูลในการทดสอบ

ชุดข้อมูลแบบ Fact Type

Company has Phone
Company workby Programmer
Programmer is_of Gender
Programmer lives_at Place
Programmer has Phone
CpLanguage is_of Gen
CpLanguage was Year
SwPackage costs Money
SwPackage has TITLE

ชุดข้อมูลแบบ Lexical

Phone(PHNR)
Company(NAME)
Programmer(PNAME)
ComputerKind(COMKIND)
Quantity(NR)

Gender(CODE)

Place(ADDRESS)

CpLanguage(CLNAME)

Gen(GENNR)

Year(AD)

SwPackage(SW)

Money(M_NUM)

ชุดข้อมูลแบบ identifier

PHNR char(7)

NAME char(10)

PNAME char(30)

COMKIND char(10)

NR char(15)

CODE char(1)

TITLE char(20)

ADDRESS char(20)

CLNAME char(25)

GENNR char(10)

AD char(4)

SW char(25)

M_NUM char(5)

BRANDNAME char(15)

MODELNAME char(15)

ชุดข้อมูลแบบ Unique

one Programmer has one Phone

one SwPackage costs one Money

one CpLanguage is_of one Gen

ชุดข้อมูลแบบ Mandatory

every one Company has one Phone

every one Company workby one Programmer

every one Programmer work one Company

every one Programmer is_of one Gender

every one Programmer lives_at one Place

every one Programmer fluent many CpLanguage

every one CpLanguage was one Year

every one SwPackage has one TITLE

ชุดข้อมูลแบบ Inter Unique

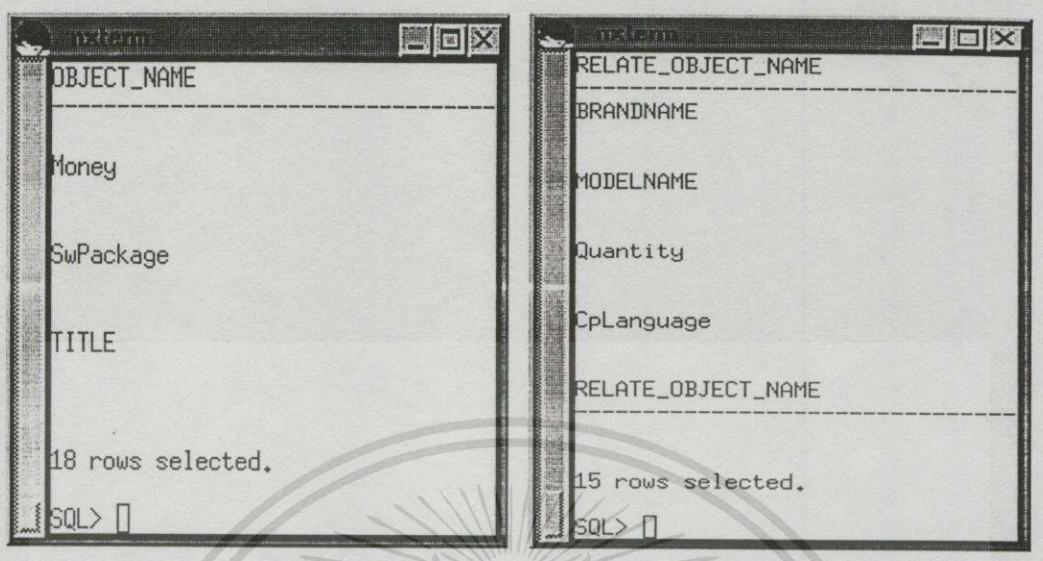
ComputerKind combination_of BRANDNAME and MODELNAME is_unique

ชุดข้อมูลแบบ Nest Fact Type

Company owns ComputerKind has one Quantity

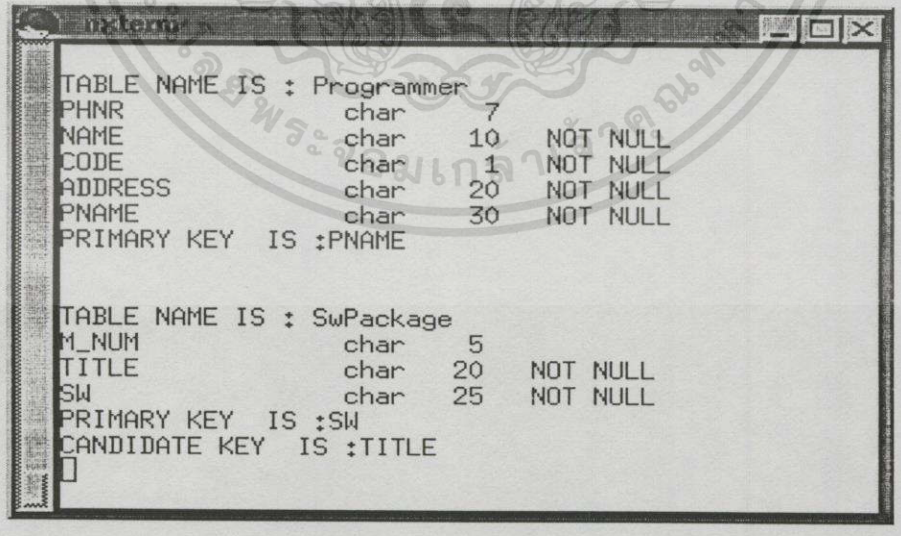
Programmer works SwPackage uses many CpLanguage

ข้อมูลที่นำเข้าเหล่านี้ได้ถูกจัดเก็บลงในตารางเมทด้า ซึ่งรูปที่ 8.2 แสดงถึงการจัดเก็บข้อมูลนำเข้าทั้งหมดเพื่อจะนำไปใช้ในกระบวนการพัฒนาระบบในขั้นตอนต่อไป หลังจากนั้นระบบจะทำการสร้าง Table Definition จะได้ในลักษณะ RDB ดังรูปที่ 8.3 ขึ้นมาเพื่อจะนำไปใช้ในการสร้าง ODL ดังในรูปที่ 8.4 ซึ่งจุดมุ่งหมายสุดท้ายของการทำวิจัยในครั้งนี้



รูปที่ 8.2 แสดงตัวของข้อมูลที่จัดเก็บในตารางเมทต้าในคอลัมน์ OBJECT_NAME ของตาราง X_TEMP_ROLE และในคอลัมน์ RELATE_OBJECT_NAME ของตาราง X_TEMP_CONSTRAINT

8.3 แสดงผลลัพธ์ของระบบในรูปของ RDB และ ODL



รูปที่ 8.3 แสดงผลลัพธ์ของระบบในรูปของ RDB

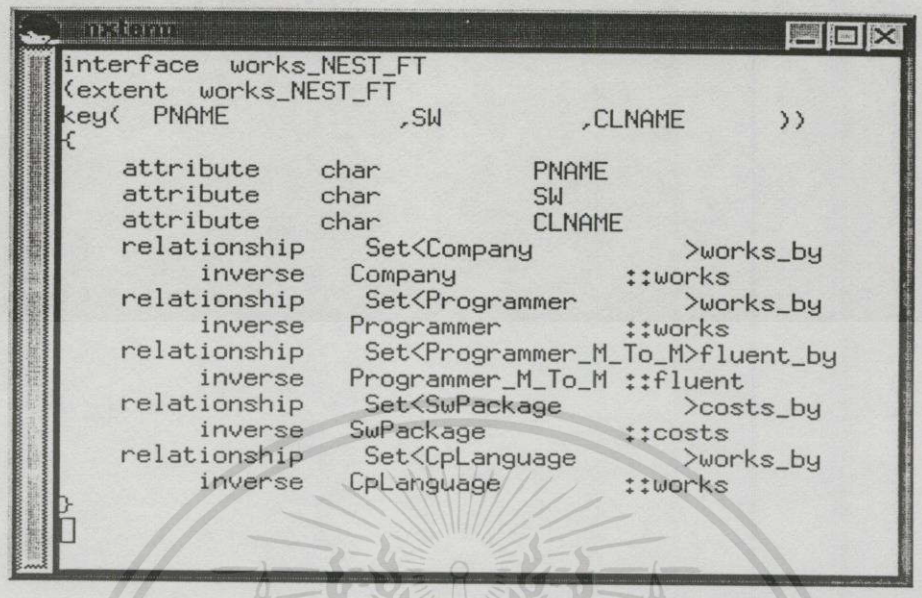
```
TABLE NAME nest fact type  IS : owns_NEST_FT
NAME                  char   10   NOT NULL
COMKIND               char   10   NOT NULL
NR                    char   15
PRIMARY KEY  IS :NAME ,COMKIND

TABLE NAME nest fact type  IS : works_NEST_FT
PNAME                 char   30    NOT NULL
SW                    char   25    NOT NULL
CLNAME                char   25    NOT NULL
PRIMARY KEY  IS :PNAME ,SW ,CLNAME

TABLE NAME INTER UNIQUE  IS : ComputerKind
BRANDNAME             char    15   NOT NULL
MODELNAME             char    15   NOT NULL
PRIMARY KEY  IS :BRANDNAME ,MODELNAME
```

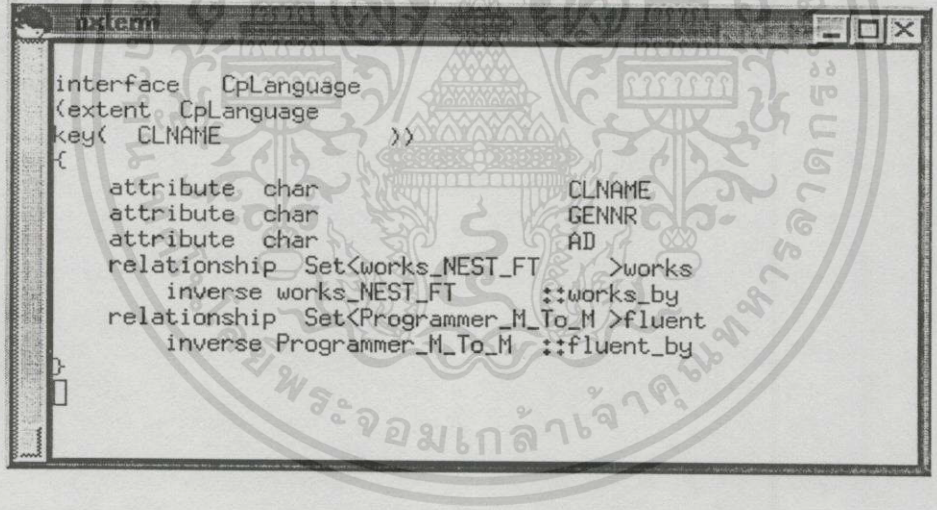
รูปที่ 8.3 (ต่อ)

```
interface   Company
(extent  Company
key(  NAME                ))
{
    attribute  char                    NAME
    attribute  char                    PHNR
    attribute  char                    PNAME
    relationship  Set<Programmer       > work
      inverse Programmer              ::work_by
    relationship  Set<works_NEST_FT    > works
      inverse works_NEST_FT           ::works_by
    relationship  Set<owns_NEST_FT     >owns
      inverse owns_NEST_FT            ::owns_by
    relationship  Set<Programmer_M_To_M >fluent
      inverse Programmer_M_To_M       ::fluent_by
}
```

```
interface   SwPackage
(extent  SwPackage
key(  SW                  ))
{
    attribute  char                    SW
    attribute  char                    M_NUM
    attribute  char                    CODE
    relationship  Set<works_NEST_FT    >works
      inverse works_NEST_FT           ::works_by
}
```

รูปที่ 8.4 แสดงผลลัพธ์ของระบบในรูป ODL ของออปเจกต์คลาส Company และ SwPackage





รูปที่ 8.4 (ต่อ)

8.4 สรุปผลการทดลอง

จากผลลัพธ์ของระบบจากการทดลองจะเห็นได้ว่าเราสามารถดูผลที่ได้ในสองรูปแบบนั่นก็คือในรูปของ RDB ซึ่งเป็นการสร้างตารางลักษณะ 5NF และคอนสเตรนท์ต่าง ๆ ซึ่งผลที่ได้ในกรณีนี้สามารถนำไปใช้งานในด้านฐานข้อมูลเชิงสัมพันธ์ได้โดยตรง สำหรับผลลัพธ์ของ ODL นั้นจะได้จากการพัฒนา RDB ต่อจนได้ ODL ที่จะนำไปใช้ในงานด้านฐานข้อมูลเชิงวัตถุ ซึ่งเป็นมุ่งหมายสุดท้ายของงานวิจัยในครั้งนี้ จากผลการทดลองแสดงให้เห็นได้ว่าได้ผลการทดลองตรงตามที่ได้

วางไว้เป็นอย่างดียิ่ง นอกจากนี้ผลการทดลองที่นำแสดงในตรงนี้เป็นเพียงส่วนหนึ่งของผลลัพธ์ทั้งหมด ซึ่งเรจได้นำผลการทดลองที่ได้ในส่วนที่เหลือไว้ในภาคผนวก ข

บทที่ 9
สรุปผลการวิจัยและข้อเสนอแนะ

9.1 สรุป

การพัฒนาภาษา SL/NIAM และเครื่องมือสำหรับออกแบบฐานข้อมูลเชิงวัตถุและฐานข้อมูลเชิงสัมพันธ์นี้ ช่วยเอื้ออำนวยความสะดวกแก่ผู้ปฏิบัติงานในหน่วยงานต่าง ๆ ที่ต้องการจะพัฒนาระบบฐานข้อมูลขึ้นเพื่อเพิ่มประสิทธิภาพในการทำงานและลดปัญหาเกี่ยวกับข้อมูลผิดพลาด เนื่องจากการใช้งานนั้นจะเป็นลักษณะของภาษากึ่งธรรมชาติที่เรียกว่า SL/NIAM ซึ่งมีรูปแบบการทำงานที่ง่าย ๆ ถึงแม้ผู้ใช้งานจะมีพื้นฐานความรู้ในด้านการออกแบบฐานข้อมูลน้อยก็ตาม ซึ่งจะแตกต่างจากการออกแบบฐานข้อมูลที่มีอยู่ในปัจจุบันโดยส่วนใหญ่จะมีลักษณะของการใช้งานทางด้านกราฟฟิคด้วยเหตุนี้ความซับซ้อนและรูปแบบของการแสดงความสัมพันธ์ของข้อมูลจะสร้างความสับสนและเป็นอุปสรรคต่อการใช้งานของผู้ที่มีพื้นฐานน้อยทางด้านการออกแบบระบบงานด้านฐานข้อมูลเป็นอย่างยิ่ง นอกจากนี้งานวิจัยได้แสดงให้เห็นถึงกระบวนการ ที่บริษัทต่างๆ นิยมนำมาใช้พัฒนาเกี่ยวกับระบบฐานข้อมูลที่ได้มีการใช้กันอยู่มากมายหลายยี่ห้อในปัจจุบัน

การพัฒนาภาษา SL/NIAM และเครื่องมือสำหรับออกแบบฐานข้อมูลเชิงวัตถุและฐานข้อมูลเชิงสัมพันธ์ ผลของงานวิจัยนี้จะสามารถที่พัฒนาออกเป็นสองส่วนนั่นก็คือในส่วนของระบบฐานข้อมูลเชิงสัมพันธ์และในรูปของภาษานิยามฐานข้อมูลเชิงวัตถุ จากในส่วนของระบบฐานข้อมูลเชิงสัมพันธ์ที่พัฒนานี้จะเป็นการสร้างรูปแบบนิยามตารางฐานข้อมูลรวมทั้งกฎข้อบังคับต่าง ๆ โดยตารางที่ได้นี้จะอยู่ในระดับฟิฟนอร์มัลฟอร์ม ซึ่งถือว่าได้รูปแบบของตารางที่ดีที่สุด ปัจจัยที่สามารถสรุปได้ว่าตารางฐานข้อมูลที่ได้นั้นดีที่สุด เนื่องจากการทำงานวิจัยในครั้งนี้ได้นำกฎข้อบังคับเชิงไนแอมต่างๆ มาสร้างเป็นรูปแบบภาษา SL/NIAM ซึ่งการทำงานของข้อมูลที่แสดงความสัมพันธ์กันของข้อมูลในลักษณะแนวคิดเชิงไนแอมจะเป็นลักษณะที่เป็นข้อมูลแบบอีลีเมนทารีแฟกไทค์ (elementary fact type) ถือได้ว่าข้อมูลเหล่านี้ไม่สามารถที่จะแบ่งแยกได้อีกแล้ว นอกจากนี้ผลงานวิจัยที่ได้สามารถที่จะขจัดปัญหากรณีที่รูปแบบของกฎที่ไม่สามารถแสดงได้ในรูปของตารางก็จะแสดงอยู่ในรูปของดาต้าเบสทริกเกอร์ (Database trigger) และอยู่ในรูปของสทอร์โปร์ซีเยอร์ (Stored procedures) ได้อีกด้วย สำหรับผลงานวิจัยในส่วนที่สองนั้นจะเป็นการพัฒนารูปแบบภาษานิยามโครงสร้างฐานข้อมูลเชิงวัตถุที่ใช้ข้อมูลจากการทำงานในลักษณะฐานข้อมูลเชิงสัมพันธ์ ซึ่งการพัฒนาที่ได้นี้หากต้องการนำมาใช้งานจริงในระบบการจัดการฐานข้อมูลเชิงวัตถุจะต้องมีการแปลงรูปแบบของภาษานิยามที่ได้ต่อไปอีก เนื่องจากงานวิจัยทั่ว ๆไปมีการแปลงรูปแบบของออปเจกต์เป็นตารางฐานข้อมูลมักจะเกิดปัญหาเกี่ยวกับการเพิ่มคีย์หลักในระบบงานที่มีกฎเกณฑ์ที่ไม่แน่นอน แต่การทำงานวิจัยในส่วนนี้จะไม่เกิดปัญหาเหล่านี้เนื่องจากรูปแบบของการทำงานจะเป็นไป

ในลักษณะตรงข้ามกับรูปแบบของงานวิจัยที่มีอยู่ทั่ว ๆ ไปเพราะผลที่ได้นั้นจะอยู่ในรูปของภาษานิยามโครงฐานข้อมูลเชิงออปเจกต์ซึ่งการทำงานในลักษณะออปเจกต์ไม่มีความสนใจเกี่ยวกับการทำงานของคีย์หลัก

9.2 ปัญหาที่พบในการวิจัยและแนวทางในการพัฒนาต่อ

งานวิจัยนี้ได้ทำการพัฒนาบนระบบปฏิบัติการลีนุกซ์ซึ่งในช่วงระยะแรก ๆ ข้อมูลข่าวสารเกี่ยวกับระบบปฏิบัติการนี้ยังไม่แพร่หลายเหมือนในปัจจุบัน รวมทั้งการทำงานของระบบจัดการฐานข้อมูลผู้วิจัยได้เลือกนำเอาโปรแกรมออราเคิลมาใช้เนื่องจากเป็นฟรีแวร์ อุปสรรคระหว่างตัวระบบปฏิบัติการกับตัวระบบจัดการฐานข้อมูลในช่วงแรกสร้างความสับสนให้กับผู้วิจัยพอสมควร เนื่องจากการทำงานของทั้งสองโปรแกรมนี้ยังใหม่ที่เพิ่งได้มีการนำมาให้ใช้ในแบบฟรีแวร์ รวมทั้งข้อจัดกัดของระบบโปรแกรมทั้งสองที่จะต้องมีการสนับสนุนการทำงานกันให้ถูกกับระดับเวอร์ชั่นการทำงานด้วย สำหรับการใช้รูปแบบของภาษากึ่งธรรมชาติที่ผู้วิจัยได้ตั้งชื่อว่า SL/NIAM นั้นมาใช้แทนการพัฒนาระบบฐานข้อมูลในแบบเก่า ๆ ที่ได้มีการใช้การแสดงแผนรูปภาพในลักษณะทางกราฟฟิคนั้น โดยได้พัฒนาจนให้สามารถแสดงผลได้ทั้งรูปแบบของ Relational Database กับในรูปแบบ Object Definition Language (ODL) รูปแบบของประโยคของภาษาเอสแอลไนแอม ได้ทำการพัฒนาอยู่ในระดับของโครงสร้างลึกนั่นก็คือรูปแบบของภาษาจะไม่เน้นการสื่อทางความหมายเทียบเท่ากับในระดับภาษาธรรมชาติจริงๆ เนื่องจากรูปแบบการที่จะพัฒนาให้อยู่ในรูปภาษาธรรมชาตินั้นยาก เพราะภาษาธรรมชาติมีรูปแบบที่ไม่แน่นอนทำให้เกิดความยากต่อการพัฒนา แต่จะให้ความสำคัญที่ความสามารถของภาษาที่สามารถสร้างกระบวนให้สามารถนำไปสู่ผลลัพธ์ที่ได้ตั้งไว้นั่นก็คือให้สามารถแสดงได้อยู่ในรูปของ Relational Database และในรูปของ ODL ซึ่งผลที่ได้ลัพธ์จากการวิจัยในครั้งนี้ถือว่าประสบความสำเร็จลุล่วงตามที่ต้องการ นอกจากนี้ปัญหาที่เกิดขึ้นกับหัวใจหลักของการเก็บข้อมูลนั่นก็คือในตารางเมทต้า เพราะรูปแบบของตารางเมทต้าที่ได้ทำการพัฒนาขึ้นมานั้นได้มีปรับปรุงเป็นจำนวนหลายครั้งมากเนื่องจากต้องการให้มีการจัดเก็บข้อมูลให้ครอบคลุมมากที่สุดที่จะนำมาใช้ในการพัฒนาระบบในครั้งนี้ สำหรับในขั้นตอนของการทำสกีม่ากราฟนั้นปัญหาที่เกิดขึ้นก็คือการเก็บชื่อของความสัมพันธ์ และอินเวอร์สความสัมพันธ์ที่เกิดขึ้นระหว่างออปเจกต์คลาส เนื่องจากในบางออปเจกต์คลาสเป็นชื่อที่ได้ทำการสร้างขึ้นมาใหม่ เพราะไม่สามารถระบุได้จากความสัมพันธ์ของชนิดเอนติตี้ที่เกิดขึ้นได้ ทำให้จำเป็นต้องกำหนดรูปแบบลักษณะของการใช้ชื่อความสัมพันธ์และอินเวอร์สความสัมพันธ์เป็น 4 รูปแบบเพื่อแก้ปัญหาที่เกิดขึ้น แต่การแก้ปัญหาที่เกิดขึ้นนี้จะใช้ได้กับความสัมพันธ์ของข้อมูลในลักษณะแบบไบนารี่ได้เท่านั้น

แนวทางในการพัฒนาต่อนั้นสามารถทำการพัฒนาเปลี่ยนแปลงจากการรับข้อมูลนำเข้าในรูปแบบของภาษา SL/NIAM โดยทำการเปลี่ยนไปเป็นการรับข้อมูลนำเข้าด้วยแผนภาพไนแอมแล้ว

ทำการแมปไปเป็นภาษานิยามโครงสร้างฐานข้อมูลในเชิงวัตถุได้ หรืออาจจะทำการปรับเปลี่ยนจากรูปแบบของภาษา SL/NIAM ที่มีลักษณะโครงสร้างทางภาษาแบบโครงสร้างลึกให้อยู่ให้รูปแบบโครงสร้างผิวได้โดยอาศัยผลงานวิจัยที่มีอยู่ทั่วไปในการแปลงรูปแบบโครงสร้างทางภาษา จะทำให้เราสามารถใช้งานจากภาษาที่เรียกว่าโครงสร้างผิวที่ใกล้เคียงกับภาษาธรรมชาติมากที่สุดมาทำการออกแบบระบบงานฐานข้อมูล สำหรับการออกแบบหลักไวยากรณ์ทางภาษา SL/NIAM นั้นจะเห็นได้ว่าอาศัยกฎข้อบังคับแนวคิดคิดเชิงไนแอม ดังนั้นหากต้องการที่จะให้รูปแบบของภาษามีความหลากหลายมากยิ่งขึ้นจำเป็นที่จะต้องศึกษากฎข้อบังคับเพิ่มเติมกว่าที่ใช้งานอยู่ในขณะนี้ นอกจากนี้งานวิจัยนี้ไม่ได้กล่าวถึงการทำงานในลักษณะเมทธอด เนื่องจากรูปแบบภาษานิยามโครงสร้างไม่เน้นการทำงานในส่วนนี้ แต่ถ้าหากจะทำการพัฒนาต่อไปสามารถกระทำได้ โดยใช้ข้อมูลจากรูปแบบของคาด้าเบสทริกเกอร์ หรือสตอร์โพรซีเยอร์มาใช้งานได้ นอกจากนี้หากต้องการที่จะให้รูปแบบการทำงานครบสมบูรณ์โดยระบบสามารถทำงานบนระบบจัดการฐานข้อมูลเชิงวัตถุได้นั้นจำเป็นจะต้องทำการแปลงรูปแบบของภาษานิยามโครงสร้างที่ได้พัฒนาขึ้นมานี้โดยใช้ ภาษา C++ ช่วยในการพัฒนาต่อ

เอกสารอ้างอิง

[1] Aho, A. V., r. Sethi , J.D. Ullman, " Compilers: Principles, Technique, and Tools Reading, MA", Addison Wesley 1986.

[2] Alfred V. Aho, Jeffrey D. Ullman, "The Theory of Parsing, Translation, and Compiling" , Volumn I: Pparsing, Prentice-Hall_Inc 1972.

[3] C.J. DATE, Relational Database : Selected writings Addisonwesley November, 1998.

[4] Clare Atkins, John Patrick, " NaLER: A Natural Language Method for Interpreting E-R Models", IEEE 1998.

[5] Dan A. Simovic, Richard L. Tenney "Relational Database Systems" , Academic Press, Inc. 1995.

[6] Dionysios C. Tsichritzis, Frederick H. Lochorsky, "Data Models", Prentice-Hall, Inc. 1982.

[7] G.M. Nijssen, T.A. Halpin, "Conceptual Schema and Relational Database Design: A fact oriented approach", Prentice Hall 1989.

[8] Henry F. Korth, Abraham Silberschatz, "Database System Concepts", McGraw-Hill, Inc,1991, 1986.

[9] James Allen, "Natural Language Understanding", The Benjamin/Cummings Publishing Company, Inc 1987.

[10] John Carter, "The Relational Database" Chapman& Hall, UK, 1995.

[11] Jeffrey D. Ullman, "A first course in database system", Prentice-Hall 1998.

[12] James A. Sykes, "English Grammar as a Sentence Model for Conceptual Modelling using NIAM" , IEEE 1994.

[13] Kemper, Alfons Heinrich, "Object-Oriented Database Management Applications in Engineering and Computer Science", Prentice Hall 1994.

[14] P.P.S Chen. The entity relationship model: Toward a unified view of data. ACM Trans. Database Syst., I(1):9-36, March 1976.

[15] Roland C. Back house, "Syntax of programming languages theory and pratices", Prentice-hall international, INC London 1979.

[16] Seth Bergman, "Compiler Design Theory, Tool, and Examples", Wn C. Brown 1994.

[17] T. Tanawong, S Chittayasothorn, "The Development of a Database Definition Language Based On The NIAM Conceptual Schema Model", The National Computer Science

and Engineering Conference NCSEC98 October 19-21,1998 Kasetsart University, Bankok, Thailand.

[18] Wenny J. Rahayu, Elizabeth Chang, " A Methodology for Transforming an Object-Oriented Data Model to a Relational Database", IEEE 1994.

[19] Yanchun Zhang , Maria E. Orlowska , " Transform a NIAM Conceptual Schema into an EKNF Relational Database Schema" , IEEE 1990.

ภาคผนวก ก

หลักไวยากรณ์ของโปรแกรมภาษา SL/NIAM

```
RELATIONSHIP:
        |RELATIONSHIP
        |RELATIONSHIP FACT_TYPE
        |RELATIONSHIP LEXICAL
        |RELATIONSHIP UNIQUE_IDENTIFIER
        |RELATIONSHIP UNIQUENESS_CONSTRAINT
        |RELATIONSHIP INTER_UNIQUENESS_IC
        |RELATIONSHIP MANDATORY_ROLE_IC
        |RELATIONSHIP NEST_FACT_TYPE
        |RELATIONSHIP MEMBERSHIP_IC
        |RELATIONSHIP RANGE_IC
        |RELATIONSHIP MANDATORY_ENTITY_IC
        |RELATIONSHIP CARDINALITY_IC
        |RELATIONSHIP CARDINALITY_OF_ROLE_IC
        |RELATIONSHIP EQUALITY_IC
        |RELATIONSHIP SUBSET_IC
        |RELATIONSHIP SUBTYPE_IC
        |RELATIONSHIP TRANSITION_OF_STR_IC
        |RELATIONSHIP TRANSITION_OF_VALUE_IC
        |RELATIONSHIP DYNAMIC_IC
        |RELATIONSHIP ASYMMETRIC
        |RELATIONSHIP IRREFLEXIVE
        |RELATIONSHIP INTRANSITIVE
        |RELATIONSHIP ACLYCLIC
        |RELATIONSHIP USER_DEFINED

FACT_TYPE:      OBJECT ROLE LABEL_TYPE
                |OBJECT  ROLE OBJECT

OBJECT:         ENTITY_TYPE
                |FACT_TYPE

LEXICAL:        LABEL_TYPE  DATATYPE '('SIZE')'
DATATYPE:       CHAR
                |VARCHAR
```

```
SIZE:                    NUMBER
UNIQUE_IDENTIFIER:       ENTITY_TYPE  IS_UNIQUELY_IDENTIFIED_BY LABEL_TYPE
                         |ENTITY_TYPE '('LABEL_TYPE')'
UNIQUENESS_CONSTRAINT:   ONE  ENTITY_TYPE ROLE QUANTITY LABEL_TYPE
                           |ONE  LABEL_TYPE ROLE QUANTITY ENTITY_TYPE
                           |ONE ENTITY_TYPE  ROLE  QUANTITY  ENTITY_TYPE

INTER_UNIQUENESS_IC:     ENTITY_TYPE  COMBINATION_OF LABEL_TYPE  AND  LABEL_TYPE  IS_UNIQUE
MANDATORY_ROLE_IC:       EVERY  QUANTITY  ENTITY_TYPE  ROLE QUANTITY LABEL_TYPE
                         |EVERY  QUANTITY ENTITY_TYPE  ROLE  QUANTITY  ENTITY_TYPE

NEST_FACT_TYPE:          OBJECT ROLE OBJECT ROLE  QUANTITY ENTITY_TYPE
QUANTITY:                ONE
                         |MANY

MEMBERSHIP_IC:           LABEL_TYPE  MUST_BE_IN '{'ENTITY_TYPE'}'
RANGE_IC:                OBJECT  ROLE  ENTITY_TYPE IN_RANGE_OF '['LOWER_LIMIT ','UPPER_LIMIT ']'
                         |OBJECT  ROLE ENTITY_TYPE COMPARATIVE  LIMIT

UPPER_LIMIT:             NUMBER
COMPARATIVE:             GREATER_THAN
                         |LESS_THAN
                         |NOT_LESS_THAN
                         |NOT_GREATER_THAN
                         |MORE_THAN

LIMIT:                   NUMBER
MANDATORY_ENTITY_IC:     SET_OF  ENTITY_TYPE MUST_BE_HAS_AT_LEAST  NUMBER  ENTITY_TYPE
                         ROLE ENTITY_TYPE  LABEL_TYPE

CARDINALITY_IC:          NUMBER_OF      ENTITY_TYPE  MUST_BE_EQUAL_TO NUMBER
                         |NUMBER_OF  ENTITY_TYPE  MUST_NOT_BE  COMPARATIVE LIMIT
                         |NUMBER_OF  ENTITY_TYPE  MUST_BE_IN_RANGE_OF
                           '['LOWER_LIMIT','UPPER_LIMIT']'

CARDINALITY_OF_ROLE_IC:  ENTITY_TYPE  PLAY ROLE_QTY NUMBER ROLES_OF '{'FACT_TYPE'}'
                         |EVERY ENTITY_TYPE  PLAY ROLE_QTY NUMBER ROLES_OF  '{'FACT_TYPE'}'

ROLE_QTY:                AT_MOST
                         |AT_LEAST
                         |EXACTLY
```

```
EQUALITY_IC:        OBJECT ROLE ENTITY_TYPE   IFF OBJECT ROLE ENTITY _TYPE
SUBSET_IC:          OBJECT ROLE ENTITY_TYPE IF OBJECT ROLE ENTITY_TYPE
SUBTYPE_IC:         ENTITY_TYPE '=' ENTITY_TYPE ROLE ENTITY_TYPE LABEL_TYPE
                    |ENTITY_TYPE '=' ENTITY_TYPE ROLE ENTITY_TYPE IN LABEL_TYPE
                    |ENTITY_TYPE '=' ENTITY_TYPE ROLE LABEL_TYPE
                    |ENTITY_TYPE '=' ENTITY_TYPE ROLE LABEL_TYPE IN LABEL_TYPE
                    |ENTITY_TYPE '=' ENTITY_TYPE ROLE ENTITY_TYPE
                    |ENTITY_TYPE '=' ENTITY_TYPE ROLE ENTITY_TYPE IN_RANGE_OF
                      LOWER_LIMIT','UPPER_LIMIT
                    |ENTITY_TYPE '=' ENTITY_TYPE ROLE ENTITY_TYPE IN NUMBER

TRANSITION_OF_STR_IC: SEQUENCE_ORDER_FOR_UPDATE_OF ENTITY_TYPE ROLE OBJECT IS LABEL_TYPE
TRANSITION_OF_VALUE_IC: FACT_TYPE IS CHANGE_DIRECTION
                        |FACT_TYPE IS CHANGE_DIRECTION COMPARATIVE NUMBER
                        |FACT_TYPE IS_NOT CHANGE_DIRECTION COMPARATIVE NUMBER
                        |FACT_TYPE IS CHANGE_DIRECTION COMPARATIVE NUMBER PERCENT
                        |FACT_TYPE IS_NOT CHANGE_DIRECTION COMPARATIVE NUMBER

CHANGE_DIRECTION:       DECREASINGLY
                        |INCREASINGLY

DYNAMIC_IC: FACT_TYPE IS CHANGE_DIRECTION PER PERIOD
            |FACT_TYPE IS CHANGE_DIRECTION COMPARATIVE NUMBER PER PERIOD
            |FACT_TYPE IS_NOT CHANGE_DIRECTION COMPARATIVE NUMBER PER PERIOD
            |FACT_TYPE IS CHANGE_DIRECTION COMPARATIVE NUMBER PERCENT PER ERIOD
            |FACT_TYPE IS_NOT CHANGE_DIRECTION COMPARATIVE NUMBER PERCENT PER PERIOD

PERIOD:         MINUTE
                |HOUR
                |DAY
                |WEEK
                |MONTH
                |YEAR

ASYMMETRIC:     FACT_TYPE    IS_ASYMMETRIC
IRREFLEXIVE:    FACT_TYPE    IS_IRREFLEXIVE
INTRANSITIVE:   FACT_TYPE    IS_INTRANSITIVE
ACLYCLIC:       FACT_TYPE    IS_ACLYCLIC
USER_DEFINED:   IF STATEMENT THEN STATEMENT EXCEPTION ACTION
STATEMENT:      CONDITION CONJUNCTION
```

```
CONDITION:      FACT_TYPE  RELATION NUMBER
                |FACT_TYPE  IN '('NUMBER')'
                |FACT_TYPE  NOT_IN '('NUMBER')'
                |FACT_TYPE  IN_RANGE_OF '['LOWER_LIMIT','UPPER_LIMIT']'
                |FACT_TYPE NOT_IN_RANGE_OF '['LOWER_LIMIT','UPPER_LIMIT']'
                |FACT_TYPE RELATION LABEL_TYPE
                |FACT_TYPE  IN LABEL_TYPE
                |FACT_TYPE  NOT_IN LABEL_TYPE
                |FACT_TYPE  LIKE STR_PATTERN
                |FACT_TYPE  NOT_LIKE STR_PATTERN

RELATION:       '='
                | '>'
                | '>='
                | '<'
                | '<='
                |'<>'

STR_PATTERN:      ALPHABET '<'COMBINE_WITH'>' WILDCARD_SYMBOL
ALPHABET:       ROLE
                |LABEL
                |'_'
                |'#'
                | '$'

WILDCARD_SYMBOL:      '%'
CONJUNCTION:        AND
                    |OR

ACTION:         PRINT LABEL_TYPE
```

ภาคผนวก ข

ผลการทดลอง RDB และ ODL

รูปของ RDB ที่เหลือ

```
TABLE NAME IS : many_to_many_1
CLNAME          char 25   NOT NULL
PNAME           char 30   NOT  NULL
PRIMARY KEY  IS :CLNAME ,PNAME

TABLE NAME IS : Company
PHNR            char   7
PNAME           char  30  NOT NULL
NAME            char  10  NOT NULL
PRIMARY KEY  IS :NAME

TABLE NAME IS : CpLanguage
GENNR           char  10
AD              char   4  NOT NULL
CLNAME          char  25  NOT NULL
PRIMARY KEY  IS :CLNAME
```

รูปของ ODL ที่เหลือ

```
interface  ComputerKind
(extent  ComputerKind
key( BRANDNAME       ,MODELNAME       ))
{
  attribute  char              BRANDNAME
  attribute  char              MODELNAME
  relationship  Set<owns_NEST_FT    >owns
    inverse owns_NEST_FT      ::owns_by
}
```

```
interface  Programmer
(extent  Programmer
key( PNAME          ))
{
   attribute  char            PNAME
   attribute  char            NAME
   attribute  char            CODE
   attribute  char            PHNR
   attribute  char            ADDRESS
   relationship  Set<Company        > work_by
      inverse Company          ::work
   relationship  Set<owns_NEST_FT     > owns
      inverse owns_NEST_FT     ::owns_by
   relationship  Set<works_NEST_FT    >works
      inverse works_NEST_FT     ::works_by
   relationship  Set<Programmer_M_To_M >fluent
      inverse Programmer_M_To_M  ::fluent_by
}

interface  Programmer_M_To_M
(extent  Programmer_M_To_M
key( PNAME          ,CLNAME          ))
{
   attribute  char            PNAME
   attribute  char            CLNAME
   relationship  Set<Company        >fluent_by
      inverse Company          ::fluent
   relationship  Set<works_NEST_FT        >fluent
      inverse works_NEST_FT  ::fluent_by
   relationship  Set<CpLanguage        >fluent_by
      inverse CpLanguage      ::fluent
   relationship  Set<Programmer        >fluent_by
      inverse Programmer       ::fluent

}
```

```
interface  owns_NEST_FT
(extent owns_NEST_FT
key( COMKIND        ,NAME          ))
{
   attribute char          COMKIND
   attribute  char         NAME
   attribute char          NR
   relationship  Set<Programmer      >owns_by
      inverse Programmer      ::owns
   relationship  Set<Company       > owns_by
      inverse Company         ::owns
    relationship  Set<ComputerKind       > owns_by
      inverse ComputerKind   ::owns
}
```

ภาคผนวก ค

โปรแกรม

การทำงานของโปรแกรมที่ได้พัฒนาจะเกิดจากการลิ้งค์โปรแกรมต่าง ๆ โดยจะประกอบไปด้วยดังนี้คือ

```
$ yacc t_parser.y –c
$ make –f /u01/app/oracle/product/8.0.5/precomp/demo/proc/demo_proc.mk
OBJS='y.atb.o xmenu.o t_parser.o t_dtb.o t_ctb.o t_input.o t_gen_odl.o t4-odl.o
x3_t_21_j43_proc.o x4_t_21_j43_proc.o x5_t_21_j43_proc.o x6_t_21_j43_proc.o
x7_t_21_j43_proc.o x8_t_21_j43_proc.o' EXE=xmenu build DBMS=v7

/* PROGRAM : t_ctb.pc */
/* PROGRAM : create table */
#include<stdio.h>
#include<string.h>
#include<sqlca.h>
EXEC SQL BEGIN DECLARE SECTION;
  VARCHAR uid[20];
#include<stdlib.h>
#include<sqlda.h>
#include<sqlcpr.h>
  VARCHAR pwd[20];
EXEC SQL END DECLARE SECTION;
EXEC SQL INCLUDE SQLCA;
char e_name_test;
char id_name;
char j;
char t_object_name[5];
char tt_object_name[8];
char ttt_object_name[5];
t1_ctb()
{
```

```
createtb();
}
createtb()
{
 extern  char  *t_test_link;
 extern  char  *t_test_link1;
 extern  char  *old_t_test_link1;
 extern  char  *t_test_link2;
 extern  char  *t_test_link3;
 extern  char  *tt_test_link;
 extern  char  *x_test_link;
 extern  char  *x_test_link1;
 extern   char  *e_name;
 extern   char  *ee_name;
 extern   char  *object_transfer_temp;
 extern   char  *x_t_test_link2_1;
 extern   char  *t_test_link2_1;
 extern   char  *object_transfer;
 extern  char  *l_t_test_link;
 extern  char  *l_t_test_link1;
 extern  char  *l_t_test_link2;
extern  char   *xy_relate_constraint;  /*inter unique  28 dec 1999 */
 extern   char  *xy_nest_fact_3;  /* 30 dec 1999 */
 extern   char  *xy_nest_fact_6;  /* 30 dec 1999 */
 extern   char  *xy_nest_fact_7;  /* 30 dec 1999 */
extern  char *M_ODL1;  /* 10 sep  2543 */
 extern  char *M_ODL2;  /* 10 sep  2543 */
 extern  char *TN_ODL_1; /* 10 sep  2543 */
 extern  char *N_ODL1;  /* 10 sep  2543 */
 extern  char *N_ODL2;  /* 10 sep  2543 */
 extern  char *N_ODL3;  /* 10 sep  2543 */
 extern  char *TN_ODL_2; /* 10 sep  2543 */
 extern  char *TN_ODL_3; /* 10 sep  2543 */
 extern  char *Nest_ODL_2; /* 10 sep  2543 */
 extern  char *x_odl_table_role;  /* 21 dec  2543 */
```

```
extern char *x_odl_inverse_role; /* 21 dec 2543 */
strcpy(uid.arr,"SCOTT");
  uid.len=strlen(uid.arr);
    strcpy(pwd.arr,"TIGER");
     pwd.len=strlen(pwd.arr);
EXEC SQL CONNECT :uid IDENTIFIED BY : pwd;
EXEC SQL CREATE TABLE X_TEMP_GROUP_SAME_TABLE
 (entity char(100),label char(100),uniq_const_type char(100),PRIMARY KEY (entity,label));

EXEC SQL CREATE TABLE X_TEMP_PK
 (table_name char(100),primarykey_name char(100),first_name char(100),second_name char
(100),third_name char(100),PRIMARY KEY (table_name));

EXEC SQL CREATE TABLE X_TEMP_ROLE
  (role_name char(100),object_name char(100),relationship_name char(100),name_exteand char
(100),table_name char(100),comment_name char(100),label_reference_name char(100),role char
(100),PRIMARY KEY (role_name));

EXEC SQL CREATE TABLE X_TEMP_IDENTIFIER
  (label_name char(100),entity_name char(100),data_type char(100),size_num
number,PRIMARY KEY (label_name));

EXEC SQL CREATE TABLE X_TEMP_REFERENCE_TYPE
  (label_name char(100),entity_name char(100),PRIMARY KEY (label_name));

EXEC SQL CREATE TABLE X_TEMP_CONSTRAINT
  (const# char(6),const_type char(100),object_name char(100),relate_entity char(2),label_type
char(100),relate_object_name char(100),PRIMARY KEY (const#));

EXEC SQL CREATE TABLE X_TEMP_IC_ROLE
  (const# char(6),role char(100),PRIMARY KEY (const#));

EXEC SQL CREATE TABLE X_TEMP_MANDATORY
  (const# char(6),relationship_desc char(100),PRIMARY KEY (const#));
EXEC SQL CREATE TABLE X_TEMP_NEST_TYPE
```

```
    (role char(100),object1  char(100),object2 char (100),PRIMARY KEY (role));

EXEC SQL CREATE TABLE  X_TEMP_EQUALITY
    (table_name char(100),object_name1  char(100),object_name2 char (100),object_name3 char
(100),PRIMARY KEY (table_name,object_name1,object_name2));

EXEC SQL CREATE TABLE  X_TEMP_ODL
    (entity_odl1 char(100),entity_odl2  char(100),entity_odl3 char (100),entity_odl4 char
(100),table_role  char(100),inverse_role char(100),PRIMARY KEY (entity_odl1,entity_odl2));

EXEC SQL CREATE TABLE  X_TEMP_TNAME_ODL
    (table_name_odl char(100),table_entity_odl1  char(100),PRIMARY KEY (table_name_odl));

EXEC  SQL  COMMIT WORK RELEASE;
}
```

ประวัติผู้เขียน

นายเทวิน ธนะวงษ์ เป็นชาวจังหวัดกำแพงเพชร จบการศึกษาระดับอุดมศึกษาจากมหาวิทยาลัยเชียงใหม่ มีประสบการณ์ทำงานในด้านเครือข่ายและระบบงานฐานข้อมูลจากบริษัทดาต้าแมทมหาชนจำกัด และแผนกไมโครคอมพิวเตอร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหาร ลาดกระบัง มีผลงานตีพิมพ์ในระดับภายในและภายนอกประเทศเช่น การประชุมวิชาการ TENCON2000 ที่ประเทศมาเลเซีย ในบทความที่เสนอเรื่อง "A FACT-BASED OBJECT-ORIENTED DATABASE DESIGN TOOL WITH A NATURAL LANGUAGE INTERFACE"