



ระบบช่วยออกแบบฐานข้อมูลแบบสัมพันธ์ ส่วนออกแบบ
A RELATIONAL DATABASE DESIGNER IN DESIGN PART



ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
ภาควิชาวิศวกรรมคอมพิวเตอร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2534

ปริญญานิพนธ์ปีการศึกษา 2534
ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหาร ลาดกระบัง

ผู้จัดทำ

1. นาย วรชัย เชาววิระประสิทธิ์ 31.1238

2. นาย สมพล สันลาวัฒน 31.1386

(ดร. ศุภมิตร จิตตะยโสธร)

..... อาจารย์ที่ปรึกษา

(อาจารย์ ธนา หงษ์สุวรรณ)

..... อาจารย์ที่ปรึกษา

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ระบบช่วยออกแบบฐานข้อมูลแบบสัมพันธ์ ส่วนออกแบบ

วรชัย เชาววิระประสิทธิ์
สมพล ลินลาวรรณ

ดร.ศุภมิตร จิตตะยโคตร อาจารย์ที่ปรึกษา
อ. ธนา หงษ์สุวรรณ อาจารย์ที่ปรึกษา

บทคัดย่อ

ปัญหานิตยภัตนั้น กล่าวถึงระบบคอมพิวเตอร์ระบบหนึ่ง ที่มีลักษณะเป็น โปรแกรมช่วยงานทางวิศวกรรมซอฟต์แวร์ พัฒนาขึ้นเพื่อช่วยงานนักออกแบบฐานข้อมูล ในการออกแบบฐานข้อมูลแบบสัมพันธ์ เพื่อให้ได้ฐานข้อมูลที่มีประสิทธิภาพ ช่วยประหยัดเวลา และช่วยตรวจสอบความถูกต้องในการออกแบบ ระบบช่วยออกแบบฐานข้อมูลแบบสัมพันธ์นี้ ขีดเอาแบบจำลองข้อมูลในเชิง เป็นหลักในการออกแบบนิยามข้อมูลระดับแนวคิด นักออกแบบจะทำการสร้างแบบจำลองในเชิงออกมาจาก โมเดลช่วยวาด ซึ่งทำงานในลักษณะการเชื่อมประสานกับผู้ใช้แบบกราฟิกผ่านสภาพแวดล้อมของไมโครซอฟต์วินโดว หลังจากนั้นจะทำการตรวจสอบแผนภาพที่ได้มา โดยอาศัยการตรวจสอบ หากการตรวจสอบไม่พบข้อผิดพลาดจึงเข้าสู่กระบวนการสร้างฐานข้อมูลบนระบบจัดการฐานข้อมูลแบบสัมพันธ์

สำหรับขอบเขตของโครงการนี้ จะครอบคลุมส่วนหนึ่งของระบบช่วยออกแบบฐานข้อมูลแบบสัมพันธ์ เรียบกล่าวกว่ประกอบด้วย ส่วนออกแบบ ประกอบขึ้นจาก โมดูลหลักสาม โมดูล คือ โมดูลช่วยวาด, โมดูลแปลงแบบจำลอง และ โมดูลตรวจสอบความถูกต้องของแบบจำลอง และเมื่อรวมเอาโมดูลสร้างฐานข้อมูลเข้าด้วย จะกลายเป็นระบบช่วยออกแบบฐานข้อมูลที่สมบูรณ์ตามหน้าที่ที่ได้ออกไว้.

A Relational Database Designer in design part

Worachai Chaoweeraprasit

Sompol Linlawan

Dr. Suphamit Chittayasothorn Advisor

Thana Hongsuwan Advisor

Abstract

This thesis , about one computer system that kind of Computer Aided Software Engineering (CASE) system. It has developed to support database designer. For design a relational database in order to be more efficiently database, reduced development time, and verify integrity of designing. This Relational Database Designer system regard to NIAM conceptual schema. NIAM model was designed by designer on Microsoft Windows Graphic User Interface module. After that, it was verify by validation rules. If verification completed, process of relational database generating will have be resumed.

Bound of this project covered on part of Relational Database Designer named as "design part" that consist of three main modules as Graphic User Interface module, Translator module and Analysis module. If database generator module was combined later, to be completed system.

กิตติกรรมประกาศ

ปัญญานิพนธ์ฉบับนี้รวมถึงโครงงานที่ยังสำเร็จลงไม่ได้ หากปราศจากคำชี้แนะอันทรงคุณค่าของพี่เต๋อ อาจารย์ธนา หงษ์สุวรรณ ความอุปการะที่ดียิ่งของดร.ศุภมิตร อาจารย์ที่ปรึกษา ที่ให้ความสนับสนุนในทุก ๆ ด้าน ไม่ว่าจะเป็นสถานที่ทำงานคือห้องโปรเจกต์พิเศษสุด หรือความเอาใจใส่ดูแลที่มีให้อย่างสม่ำเสมอ ขอขอบคุณชมรมคอมพิวเตอร์และน้อง ๆ ในชมรมที่เอื้อเฟื้อสถานที่ที่ใช้ผ่อนคลายความเครียด ขอขอบคุณพี่ไฉดที่ทำให้ร่างกายกระฉับกระเฉงอยู่เสมอ ขอขอบคุณน้องหนูที่ช่วยเพิ่มความตื่นตัวในเวลาใกล้ส่งปัญญานิพนธ์ ขอขอบคุณเพื่อน ๆ ที่ห้องอาจารย์พล ๆ ที่เอื้อเฟื้อกาแฟหลายต่อหลายครั้ง และสุดท้ายขอขอบคุณเพื่อน ๆ ทุกคนที่ช่วยกันสร้างบรรยากาศการทำงานที่สดชื่น.

สารบัญ

เรื่อง

หน้าที่

บทคัดย่อ	I
Abstract	II
บทที่ 1 บทนำ	1
1.1 ระบบฐานข้อมูลและการออกแบบฐานข้อมูล	1
1.2 ระบบช่วยออกแบบฐานข้อมูลแบบสัมพันธ์	1
1.2.1 วัตถุประสงค์ในการพัฒนาระบบ ฯ เวรด	3
1.2.2 องค์ประกอบของเวรด	3
1.2.3 เมตาสกีมม่า	4
1.2.4 เวรด-จีอาร์	4
1.2.5 เวรด-เอ็มดีบี	5
1.2.6 เวรด-เอ	5
1.3 รายละเอียดของเนื้อหาในแต่ละบท	6
บทที่ 2 หลักการเบื้องต้น	7
2.1 ระบบฐานข้อมูลแบบสัมพันธ์	7
2.2 โมเดลแบบสัมพันธ์	7
2.3 โครงสร้างข้อมูลแบบสัมพันธ์	8
2.3.1 ตัวอย่างตารางความสัมพันธ์	8
2.3.2 รีเลชัน (Relations)	8
2.4 ความถูกต้องของข้อมูล	11
2.4.1 ไทเมอร์รี่	12
2.4.2 กฎความถูกต้องของเฮนรี่	13
2.4.3 กฎความถูกต้องในการอ้างอิง	13
2.5 การจัดการข้อมูล	13
2.6 สถาปัตยกรรมของระบบฐานข้อมูล	15

เรื่อง	หน้า
2.7 การออกแบบฐานข้อมูลโดยใช้วิธีนอร์มอลไลซ์เซชัน	16
2.8 แบบจำลองข้อมูลในแอม	17
2.8.1 ส่วนประกอบต่างๆ ของในแอม	17
2.8.2 ขั้นตอนการออกแบบจำลองข้อมูลในแอม	19
2.8.3 การสร้างเค้าร่างฐานข้อมูลแบบสัมพันธ์ โดยวิธี แปลงแบบจำลองระดับแนวคิดของในแอม	46
บทที่ 3 เมตาสกินม่า	48
3.1 เอนคิโนในในแอมคอนเซ็ปชวลเมตาสกินม่า	49
3.2 ความสัมพันธ์ระหว่างเอนคิโนกับในในแอม	52
3.2.1 ความสัมพันธ์แบบสลับเซต	52
3.2.2 ความสัมพันธ์แบบพหุคูณ	52
3.3 เมตาสกินม่าในลักษณะการวิจัย	55
บนฐานข้อมูลแบบสัมพันธ์อวราเคิล	
3.4 แมปปิงอินฟอร์เมชัน (Mapping Information)	59
3.4.1 เอนคิโนที่เพิ่มเข้ามาในแมปปิงอินฟอร์เมชัน	59
3.5 แมปปิงอินฟอร์เมชันในลักษณะวิจัย	60
บนฐานข้อมูลแบบสัมพันธ์อวราเคิล	
3.6 ข้อมูลรายละเอียดของแอตทริบิวต์ต่าง ๆ ใน ตารางเมตาสกินม่าและแมปปิงอินฟอร์เมชัน	61
บทที่ 4 ไดอะแกรมของระบบช่วยออกแบบฐานข้อมูลแบบสัมพันธ์	67
4.1 สภาพแวดล้อมของระบบช่วยออกแบบ ฯ	68
บทที่ 5 โมดูลช่วยงาน REED-Gr	71
5.1 ความเข้ากันไม่ได้ระหว่างไมโครซอฟท์วินโดว และ ระบบจัดการฐานข้อมูลอวราเคิล	71
5.2 รูปแบบไฟล์เคลกริบเตอร์	73
5.3 รูปแบบตัวอย่าง	77
บทที่ 6 โมดูลแปลงแบบจำลองข้อมูล REED-MDB	82

เรื่อง	หน้า
6.1 การเข้ารหัสแบบไฮราเค่ไนรีเลข UNIQREF	84
6.2 การแมปอิงอัตโนมัติ (Automapping)	87
บทที่ 7 โมดูลวิเคราะห์แบบจำลองไมเนม REDD-A	
7.1 ตรวจสอบการเนมมิง (Naming analysis)	90
7.2 ตรวจสอบความบริบูรณ์ของการเชื่อมต่อ (Completeness analysis)	91
7.3 ตรวจสอบการใช้คอนสแตนต์	92
บทที่ 8 การใช้งานระบบช่วยออกแบบฐานข้อมูลแบบสัมพันธ์ ส่วนออกแบบ	97
8.1 เจาะใจในการทำงานของระบบช่วยออกแบบ ฯ	97
8.2 เริ่มต้นการใช้งาน	97
บทที่ 9 สรุปผลและวิจารณ์	106
บทที่ 10 แนวทางการพัฒนาต่อ	108
เอกสารอ้างอิง	111
ภาคผนวก	
ก. ไปรสดารชี	112
ข. ไมโครซอฟต์วินโดวส์	118
ค. ระบบจัดการฐานข้อมูลอวาคีล	120
ง. วิธีการคิดเงินเข้าลกับมา	124

สารบัญตาราง

ตารางที่	หน้า
2.1 แสดงตัวอย่างข้อมูลความสัมพันธ์ของรถและคนขับ	22
2.2 แสดงตัวอย่างข้อมูลของผลการเรียนนักศึกษา	24
2.3 แสดงตัวอย่างข้อมูลรายการของสินค้า	26
2.4 แสดงตัวอย่างข้อมูล	31
2.5 แสดงตัวอย่างข้อมูลของผลการเรียน	33
2.6 แสดงตัวอย่างข้อมูล	35
2.7 แสดง Projection ของข้อมูลในตารางที่ 2.6	35
2.8 แสดงผลการ join ข้อมูลในตารางที่ 2.6	35
2.9 แสดงข้อมูลของคนใช้	38
2.10 แสดงตัวอย่างข้อมูล	39
2.11 แสดงตัวอย่างข้อมูล	42
2.22 แสดงตัวอย่างข้อมูลของสโมสรกีฬาแห่งหนึ่ง	45

สารบัญรูป

รูปที่	เรื่อง	หน้าที่
2.1	แสดงตารางความสัมพันธ์ข้อมูล Supplier	8
2.2	แสดงตัวอย่างตารางที่มีกลุ่มซ้ำและไม่มี	11
2.3	แสดงแผนภูมิสถาปัตยกรรมของระบบข้อมูล 3 ระดับ	15
2.4	แสดงตัวอย่างระดับข้อมูลในระบบใช้งาน	16
2.5	แสดงตัวอย่างข้อมูลตารางเวียน	19
2.6	แสดงความสัมพันธ์ของข้อมูลในตารางด้วยรูปภาพ	22
2.7	แสดงแผนภาพจำลองข้อมูลระดับแนวคิด	23
2.8	แสดงการเขียนแบบจำลองข้อมูลที่มีเรฟเฟอร์เรนซ์โทโมแบบ 1:1	23
2.9	แสดง Schema-based diagram	24
2.10	แสดงแฟกต์ไทม์ของผลการเวียน	25
2.11	แสดงการเขียนแฟกต์ไทม์ของผลการเวียนแบบเนลลิ่ง	25
2.12	แสดงแผนภาพจำลองข้อมูลระดับแนวคิด	26
2.13	แสดงแผนภาพจำลองข้อมูลระดับแนวคิด	27
2.14	แสดงรูปแบบที่มีชื่อของแบบจำลองข้อมูล	28
2.15	แสดงผลการนำขึ้นตอนที่ 3 มาใช้แก้ไข	28
2.16	แสดงตัวอย่างการใช้งานคอมพิวเตอร์แบบชนิด	30
2.17	แสดงแบบร่าง ๆ ของคอนสเตรนส์แบบชนิด สำหรับแฟกต์ไทม์แบบไบนารี	30
2.18	แสดงคอนสเตรนส์แบบชนิดของเทอร์นารี	31
2.19	แสดงแบบจำลองข้อมูลที่ได้จากตาราง	31
2.20	แสดงคอนสเตรนส์แบบชนิดแบบ Inter	32
2.21	แสดงแบบจำลองข้อมูลที่ได้จากตารางที่ 2.5	34
2.22	แสดงตัวอย่างของแบบจำลองข้อมูลของผลการแข่งขันกีฬา	36
2.23	แสดงการใช้อุปกรณ์คอมพิวเตอร์แบบชนิดไทม์	37
2.24	แสดงความสัมพันธ์สิบไทย	39

รูปที่ เรื่อง	หน้าที่
2.25 แสดงการใช้สไลด์ทาบในแบบจำลองข้อมูล	40
2.26 แสดงตัวอย่างการใช้งานอีคเคอเรนท์ ฟรีควินซี	40
2.27 แสดงตัวอย่างการใช้งานอีคเคอเรนท์ ฟรีควินซี	41
2.28 แสดงตัวอย่างการใช้งานอีคเคอเรนท์ ฟรีควินซี	41
2.29 แสดงตัวอย่างแบบจำลอง	43
2.30 แสดงการอ้างอิงแบบไฮราคิ	44
2.31 แสดงแบบจำลองข้อมูลของตารางที่ 2.12	45
3.1 แผนภูมิแสดงระดับในการแทนข้อมูลระดับแนวคิด	48
5.1 แสดงรูปแบบไฟล์เดสคริปเตอร์	74
5.2 รูปในแผนที่ใช้เป็นตัวอย่างในการทดสอบ	77
5.3 ภาพหน้าจอรูปร่าง 5.2 ของโปรแกรมช่วยวาด	78
5.4 เดสคริปเตอร์ของรูปร่างที่ 5.2	79
5.5 รูปในแผนที่ตัวอย่างกรณีการอ้างอิงแบบไฮราคิ	80
5.6 ภาพแสดงหน้าจอรูปร่าง 5.5	80
5.7 เดสคริปเตอร์ของรูปร่างที่ 5.5	81
6.1 แสดงไดอะแกรมของโมดูลแปลงแบบจำลอง	82
6.2 แสดงอัลกอริทึมการแปลงค่าข้อมูลในรีเลย์ UNIQREF กรณีการอ้างอิงแบบไฮราคิ	86
6.3 แสดงค่าข้อมูลในรีเลย์ UNIQREF กรณีอ้างอิงแบบไฮราคิ	87
6.4 เชนส์ไทยมีการอ้างอิงแบบหนึ่งอันหนึ่งกับเลเบลไทย	88
6.5 แสดงรูปที่เชื่อมแล้วของรูปที่ 6.4	88
7.1 การอ้างอิงเลเบลไทยแบบปกติ	90
7.2 การอ้างอิงเลเบลไทยแบบแนว	91
7.3 องค์ประกอบของคอนเซ็ปต์วลลี	92
8.1 แสดงหน้าจอเมื่อทำการคลิ๊กที่ระบบจัดการอวาเคิล	98
8.2 แสดงหน้าจอเริ่มต้นของโปรแกรม NIAM.EXE ซึ่งเป็นส่วนโปรแกรมช่วยวาด	99
8.3 แสดงหน้าจอการสร้างเอนส์ไทย	101

รูปที่ เรื่อง	หน้าที่
8.4 แสดงหน้าจอการสร้างเลเบิลไทย	102
8.5 แสดงหน้าจอของแบบจำลองที่สำเร็จลงแล้ว	103
8.6 แสดงหน้าจอของโปรแกรมการแปลงข้อมูล และวิเคราะห์ DB.EXE	104
8.7 ตัวอย่างไฟล์แจ้งผลการวิเคราะห์ E001.REP	105
10.1 การอ้างอิงกรณีไฮวาคีแบบมีสับไทย	109
10.2 เรฟเฟอร์เรนซ์ไทยแบบหาได้	110
ง.1 แสดงไฟล์ติดตั้งเมตาสกิมม่าและแมปปิงอินฟอร์เมชัน MET.SQL	124-129



1.1 ระบบฐานข้อมูลและการออกแบบฐานข้อมูล

จากนิยามคำว่า "ฐานข้อมูล" หรือ Database ซึ่งหมายถึง ข้อมูลและความสัมพันธ์ของข้อมูล และคำว่า "ระบบฐานข้อมูล" ที่หมายถึง ระบบคอมพิวเตอร์ประเภทหนึ่งซึ่งมีหน้าที่ดูแลความถูกต้องของเรคคอร์ดข้อมูลหรือ record keeping systems [:An Introduction to Database Systems ; C.J.Date 1986] ทำให้สามารถสรุปได้ว่า ระบบฐานข้อมูลนั้นคือ ระบบดูแลข้อมูลระบบหนึ่ง ซึ่งจะทำการดูแลความสัมพันธ์ของข้อมูลภายใน ปัจจุบันการใช้งานข้อมูลไม่ว่าจะเป็น การเก็บข้อมูลทางสถิติ ทางวิชาการ การใช้ข้อมูลเพื่อการตัดสินใจ ล้วนแล้วแต่เป็นข้อมูลจำนวนมากที่มนุษย์ไม่สามารถดูแลได้ทั่วถึงหรือมีประสิทธิภาพเพียงพอ การใช้ระบบฐานข้อมูลเข้ามาทำการดูแลข้อมูลข่าวสารแทนมนุษย์จึงมีใช้อย่างแพร่หลาย ฐานข้อมูลถูกนำไปประยุกต์ใช้ในระบบต่าง ๆ เช่น ระบบช่วยการตัดสินใจ ระบบวิเคราะห์ข้อมูล ระบบฐานความรู้ และอื่น ๆ และเนื่องจากการใช้งานฐานข้อมูลเป็นไปอย่างแพร่หลาย ความถูกต้องและความกระชับรัดกุมของข้อมูลภายในฐานข้อมูลเป็นสิ่งสำคัญ จึงต้องมีการออกแบบวิธีจัดเก็บข้อมูลภายในให้มีระเบียบแบบแผนที่ดีสมควร มีความซ้ำซ้อนของข้อมูลน้อยที่สุด เพื่อให้ง่ายต่อการแก้ไขเปลี่ยนแปลงข้อมูลภายหลัง เรียกกระบวนการนี้ว่า การออกแบบฐานข้อมูล (Database design) หากฐานข้อมูลที่ใช้อยู่ในรูปของไฟล์ฐานข้อมูล การออกแบบฐานข้อมูล อาจหมายถึงการออกแบบระบบไฟล์ที่จะใช้ แอดมินิวิวกที่จะจัดเก็บในไฟล์เรคคอร์ดของแต่ละไฟล์ หากฐานข้อมูลที่ใช้อยู่ในรูปของโครงสร้างไฮเอราคี (Hierarchicy structure) จะหมายถึงการออกแบบเรคคอร์ดไทป์ (record type) ไทป์ (tree type) ที่กำหนด เป็นต้น

1.2 ระบบช่วยออกแบบฐานข้อมูลแบบสัมพันธ์ (เรดดี : REDD system)

เรดดี (REDD) ย่อมาจาก Relation Database Designer คือระบบช่วยออกแบบฐานข้อมูลแบบสัมพันธ์ จัดมาขึ้นเพื่อให้เป็นเครื่องมือช่วยออกแบบฐานข้อมูลหรืออานาลิสต์ จัดเป็นซอฟต์แวร์ประเภท ซอฟต์แวร์วิศวกรรมซอฟต์แวร์ หรือ เคส (CASE : computer- aided software engineering) นิยามแบบจำลองข้อมูลที่เรียกว่า ไนแอม (NIAM :

Nijssen 's Information Analysis Methodology) ซึ่งเป็นนิยามข้อมูลระดับแนวคิด หรือ คอนเซ็ปทวลสกีมา (Conceptual schema) ที่คิดค้นขึ้นโดย Prof.G.M.Nijssen แห่งมหาวิทยาลัยคิงส์แลนด์ ประเทศออสเตรเลีย เป็นแบบจำลองข้อมูลที่ใช้ในการออกแบบ

ในแอมเป็นแบบจำลองประเภท แบบจำลองแทนความหมาย หรือ ซีแมนติกโมเดลลิง (semantic modeling) เช่นเดียวกับ แบบจำลองอีอาร์ (ER : entity/relationship model) แต่มีความสามารถในการแทนข้อมูล และตั้งข้อจำกัดของข้อมูลได้ดีกว่า ผู้ใช้ซึ่งก็คือ นักออกแบบฐานข้อมูล จะทำการสร้างแบบจำลองในแอม ของแอพพลิเคชันที่จะอาศัยฐานข้อมูล บนโปรแกรมช่วยวาด ซึ่งเป็นโมดูลโปรแกรมที่ทำงานภายใต้เอนไวรอนเมนต์ของ ไมโครซอฟต์ วินโดวส์ (Microsoft Windows) ซึ่งเป็นซอฟต์แวร์ประเภทกึ่งระบบปฏิบัติการ ที่ทำงานอยู่บนเครื่องคอมพิวเตอร์ระดับคอมพิวเตอร์ส่วนบุคคลที่ใช้ดอส (DOS) เป็นระบบปฏิบัติการ มีลักษณะโดดเด่นคือ สามารถประมวลผลงานมากกว่าหนึ่งงานได้ในเวลาพร้อม ๆ กัน หรือที่เรียกว่า (Multitasking) และแสดงผลตอบสนองต่อผู้ใช้แบบกราฟิก ในโปรแกรมช่วยงานดังกล่าวนี้ จะควบคุมขั้นตอนการวาดในแอม มิให้ผิดเพี้ยนจากข้อกำหนดเบื้องต้นของในแอมเอง เพื่อช่วยเพิ่มความสะดวกแก่นักออกแบบ เช่นนี้แล้ว ในทางปฏิบัติผู้ใช้อาจไม่มีความจำเป็น ที่จะต้องรู้ข้อกำหนดของในแอมดีตา ก็ยังสามารถสร้างแบบจำลองในแอมขึ้นมาได้ โดยอาศัยเวลาไม่นานนัก

เมื่อได้แบบจำลองในแอมจากโปรแกรมช่วยวาดแล้ว จะนำแบบจำลองในแอมที่ได้มาแปลงให้อยู่ในรูปที่สามารถใช้งานได้ โดยมีลักษณะเป็นฐานข้อมูลชนิดหนึ่ง ที่ข้อมูลที่อยู่ภายในใช้อธิบายคุณสมบัติของแบบจำลองในแอม เรียกว่า ฐานข้อมูลนี้ว่า อภิธานข้อมูล หรือ เมต้าเดต้าเบส (Metadatabase) โปรแกรมที่ใช้แปลงรูปแบบของแบบจำลองในแอมนี้ เรียกว่า โปรแกรมแปลงแบบจำลอง เนื่องจากเมต้าเดต้าเบสนี้เป็นฐานข้อมูลชนิดหนึ่ง จึงมีคุณสมบัติเช่นเดียวกับฐานข้อมูลทั่วไป คือสามารถมีความซับซ้อนของข้อมูลเกิดขึ้นได้ ถ้าหากการออกแบบเค้าร่างฐานข้อมูล (database schema) ไม่ดีพอ การออกแบบเมต้าเดต้าเบสจะแยกกล่าวในลำดับต่อไป

เมื่อแบบจำลองในแอม ถูกแปลงให้อยู่ในรูปของเมต้าเดต้าเบสเรียบร้อยแล้ว จะนำเมต้าเดต้าเบสไปทำการวิเคราะห์ความถูกต้อง โดยอาศัยโปรแกรมวิเคราะห์แบบจำลองในลักษณะการวิเคราะห์ จะแบ่งลำดับการวิเคราะห์ออกเป็น 3 ระดับ คือ 1. การวิเคราะห์ความถูกต้องของการแมป 2. การวิเคราะห์ความถูกต้องในการเชื่อมโยง และ 3. การวิเคราะห์ความถูกต้องในการกำหนดเงื่อนไขของข้อมูล ทั้ง 3 ระดับจะแยกกล่าวโดยละเอียดในหัวข้อต่อไป หากผลการวิเคราะห์ส่งผลให้เกิดความผิดพลาดในการสร้างแบบจำลอง จะรายงาน

งานกลับไปไปยังนักออกแบบเพื่อแก้ไขแบบจำลอง ทำการแปลงแบบจำลอง และทำการวิเคราะห์ใหม่อีกครั้ง แต่ถ้าหากผลการวิเคราะห์ไม่พบข้อผิดพลาด การทำงานจะเข้าสู่ส่วนสุดท้าย ที่ จะทำการสร้างเค้าร่างหรือสกีมา (schema) ซึ่งหมายถึงตารางฐานข้อมูลแบบสัมพันธ์ที่ยังไม่ มีข้อมูลอยู่ภายใน โดยอาศัยแบบจำลองในแอมที่วิเคราะห์แล้ว สกีมาที่ได้จากแบบจำลองใน แอม จะเป็นตารางฐานข้อมูลที่ประกาศความซับซ้อนของข้อมูลเมื่อใส่ข้อมูลน้อยมาก ทำให้ฐาน ข้อมูลที่ได้กระทัดรัด และง่ายต่อการเปลี่ยนแปลงแก้ไขข้อมูลภายหลัง ข้อดีอีกอย่างหนึ่งในการ ใช้แบบจำลองในแอม คือ สามารถจะกำหนดข้อจำกัดของข้อมูลในฐานข้อมูลได้ ทำให้ข้อมูลที่อยู่ ภายในมีลักษณะที่ถูกต้อง แต่ทั้งนี้ทั้งนั้นต้องขึ้นอยู่กับโปรแกรมจัดการฐานข้อมูลที่ใช้ด้วย

1.2.1 วัตถุประสงค์ในการพัฒนาระบบช่วยออกแบบฐานข้อมูลแบบสัมพันธ์ : เรดด์

1. เพื่อใช้เป็นเครื่องมือสำหรับนักออกแบบฐานข้อมูล เพื่อใช้ในการออกแบบฐานข้อมูลแบบสัมพันธ์ของระบบงานที่ต้องการใช้ฐานข้อมูล
2. ช่วยประหยัดเวลาที่ใช้ในกระบวนการออกแบบฐานข้อมูล
3. เพิ่มความสะดวกในการวาดแบบจำลอง และเพิ่มความถูกต้องในแบบจำลองในแอม ซึ่งสามารถสรุปข้อดีได้สามประการคือ
 - ช่วยให้การวาดในแอมง่ายและสะดวกยิ่งขึ้น ง่ายต่อการแก้ไขเปลี่ยนแปลง และได้เอาท์พุทที่สวยงามกว่าการวาดด้วยมือ
 - ช่วยตรวจสอบความถูกต้องเบื้องต้นของแบบจำลองในแอม ความถูกต้องในการใส่ข้อจำกัด (constraint) ของข้อมูล
 - ช่วยให้การสร้างเค้าร่างฐานข้อมูลจากแบบจำลองในแอม มีความสมบูรณ์และถูกต้องมากยิ่งขึ้น
4. เพื่อเป็นส่วนหนึ่งของระบบการออกแบบฐานความรู้ ซึ่งเป็นที่เก็บข้อมูล กฎความรู้ และข้อกำหนดลักษณะของข้อมูล เพื่อใช้ในการสร้างระบบจัดการฐานข้อมูล (RDBMS) ที่มีประสิทธิภาพสูงกว่าระบบจัดการฐานข้อมูลทั่ว ๆ ไป

1.2.2 องค์ประกอบของเรดด์

เรดด์ ประกอบด้วยจากส่วนประกอบทั้งหมด 5 ส่วน อันได้แก่

2.1 เม้าสชีเมา (Mouseschema) คือ เค้าร่างฐานข้อมูลของเม้าเดต้าเบส

2.2 โปรแกรมช่วยวาด หรือ REDD-Gr (เรดด์-จีอาร์) คือ โมดูลรับแบบจำลองในแอมโดยผ่านระบบกราฟิกยูสเซอร์อินเทอร์เฟซ (GUI)

2.3 โปรแกรมแปลงแบบจำลอง หรือ REDD-MDB (เรดด์-เอ็มดีบี) คือ โมดูลแปลงแบบจำลองกราฟิกของในแอมให้อยู่ในรูปแบบเดต้าเบส และจัดเก็บข้อมูลลงในเมตาสคีมม่า

2.4 โปรแกรมวิเคราะห์แบบจำลอง หรือ REDD-A (เรดด์-เอ) คือ โมดูลวิเคราะห์และตรวจสอบความสมบูรณ์และถูกต้องของแบบจำลองในแอม ซึ่งอยู่ในรูปแบบเดต้าเบสเรียบร้อยแล้ว

2.5 โปรแกรมสร้างเค้าร่างฐานข้อมูล หรือ REDD-G (เรดด์-จี) คือ โมดูลสร้างตารางฐานข้อมูลแบบสัมพันธ์ จากเมตาสคีมม่าและที่ผ่านกระบวนการวิเคราะห์แล้วโดย REDD-A ให้เป็นตารางฐานข้อมูลระดับ ONF (รายละเอียดในส่วนที่อยู่ในโครงการระบบช่วยออกแบบฯ ส่วนโครงสร้างข้อมูล)

1.2.3 เมตาสคีมม่า

หรือ เมตาสคีมม่า คือ เค้าร่างฐานข้อมูลซึ่งใช้เก็บข้อมูลลักษณะของแบบจำลองในแอม โดยเรียกข้อมูลในเมตาสคีมม่าเหล่านี้ว่า เมตาดেটাเบส เมตาสคีมม่าจะอยู่ในรูปตารางความสัมพันธ์เช่นเดียวกับตารางฐานข้อมูลในระบบฐานข้อมูลแบบสัมพันธ์ทั่วไป และเนื่องจากตัวมันเองเป็นตารางฐานข้อมูลหรือวีเลชันด้วย จึงเหมาะที่จะถูกออกแบบด้วยแบบจำลองในแอมเช่นเดียวกัน เวทในแอมที่กล่าวได้ว่าใช้แทนไวยากรณ์ของในแอมด้วยกันเอง นี้ว่า ในแอม-คอนเซ็ปชวลเมตาสคีมม่า (NIAM Conceptual metaschema :NIAM-CM) จาก NIAM-CM ที่ออกแบบมานี้ จะสามารถสร้างตารางฐานข้อมูลทั้งหมด 21 ตารางกับอีก 1 วิว และเราเรียกตารางและวิวดังกล่าวนี้ว่า เมตาสคีมม่า (Metaschema)

1.2.4 เรดด์-จีอาร์ (REDD-Gr)

เป็นโปรแกรมโมดูลที่ใช้รับแบบจำลองในแอมผ่านทางระบบอินเทอร์เฟซผู้ใช้แบบ จียูไอ (GUI:Graphic User Interface) ซึ่งใน REDD นี้เราใช้เอาเอนไวรอนเมนต์ของไมโครซอฟท์วินโดวส์เป็นกลาง ไมโครซอฟท์วินโดวส์ หรือ Microsoft Windows 3.0 เป็นเอนไวรอนเมนต์แบบกราฟิกที่รองรับระบบจัดการเอ็มแอลเอส (MSDOS) ช่วยจัดการการทำงานของโปรแกรมต่างๆ ในระบบแบบกราฟิกทั้งนี้ โดยวินโดวส์จะเป็นผู้จัดสรรทรัพยากรของระบบให้แก่และ

โปรแกรมที่เหมือนกัน เหตุผลหนึ่งที่เลือกเอาไมโครซอฟต์วินโดว์เป็นระบบอินเทอร์เน็ตเฟส ก็เนื่องจากเป็นระบบกราฟิกอินเทอร์เน็ตเฟสที่ถือว่าสมบูรณ์ที่สุดที่วิ่งอยู่บนเครื่องระดับพีซีในเวลานี้ เครื่องมือที่ใช้ในการพัฒนาโปรแกรมก็มีอยู่อย่างเพียงพอ และมีเอกสารอ้างอิงมากมาย ทำให้ง่ายต่อการพัฒนาต่อไปในอนาคต ถึงแม้ว่าโดยทางปฏิบัติจะหาความยุ่งยากอยู่มากในการพัฒนาโปรแกรมก็ตาม

1.2.5 เวดค์-เอ็มดีบี (REDD-MDB)

เริ่มโปรแกรม โมดูลที่ใช้แปลงลักษณะของแบบจำลองไมเอมให้อยู่ในรูปเมต้าเดต้าเบส รับอินพุตในรูปของเดสคริปเตอร์ของรูปไมเอม เดสคริปเตอร์ดังกล่าวนี้เป็นเท็กซ์ไฟล์ที่เกิดขึ้นในกระบวนการวาดรูปไมเอมบนจอของวินโดว์ เก็บรายละเอียดของอ็อบเจ็กต์และการเชื่อมต่อทุกอย่างในรูปไมเอม อาจกล่าวได้ว่าเดสคริปเตอร์นี้เป็นรูปแบบชั่วคราว หรือ อินเตอร์มีเดียเตอร์ (intermediate form) ของรูปไมเอมที่วาดขึ้น และใช้ในกระบวนการของโปรแกรมเช่นๆ แทนการใช้ไฟล์รูปภาพซึ่งเป็นรูปแบบไฟล์กราฟิกเฉพาะของไมโครซอฟต์วินโดว์ (เรียกว่า เมต้าไฟล์) ซึ่งยุ่งยากและขาดความคล่องตัว

หลังจากทำการแปลเดสคริปเตอร์จนได้เมต้าเดต้าเบสแล้ว จึงนำเข้าเมต้าเดต้าเบสนี้ไปตรวจสอบความถูกต้องของไมเอมโดยโมดูลส่วน REDD-A ต่อไป

1.2.6 เวดค์-เอ (REDD-A)

เริ่มโปรแกรม โมดูลที่ใช้วิเคราะห์เพื่อตรวจสอบความถูกต้องของแบบจำลองไมเอมที่อยู่ในรูปเมต้าเดต้าเบส หากวิเคราะห์แล้วพบว่าไม่มีผิดพลาดเกิดขึ้น จะทำการทำลายเมต้าเดต้าเบสทันทีพร้อมกับรายงานความผิดพลาด เพื่อให้โปรแกรมสร้างเค้าร่างฐานข้อมูลนำไปสร้างเป็นตาราง การวิเคราะห์จะกระทำตามกฎแห่งความถูกต้องของไมเอม ที่เรียกว่า วาเลชันรูล (Validation rules) ซึ่งประกอบด้วย

1. กฎการเนมมิง (Naming analysis) ใช้ตรวจสอบการอ้างอิงเบิ้ลไทร์ของเอมโอบีไทร์กับตัวของไมเอม
2. กฎความสมบูรณ์ (Completeness analysis) ตรวจสอบความสมบูรณ์ของความเชื่อมโยงระหว่างอ็อบเจ็กต์ในแบบจำลองไมเอม
3. กฎตรวจสอบข้อจำกัด (Constraint-Checking analysis) ตรวจสอบ

ความถูกต้องของข้อจำกัดหรือคอนสเตรนต์ (constraint) และความขัดแย้งระหว่าง
คอนสเตรนต์ในแบบจำลองไนเอม

กฎต่าง ๆ และลักษณะแบบจำลองไนเอมจะกล่าวในหัวข้อถัดไป

1.3 รายละเอียดของเนื้อหาในแต่ละบท

ภายในปริทัศน์บทฉบับนี้ ประกอบด้วยเนื้อหาทั้งหมด 9 บท โดยในบทที่ 2 กล่าวถึง
หลักการเบื้องต้น หรือ ทฤษฎีที่เกี่ยวข้องในโครงการ ได้แก่ หลักการของระบบฐานข้อมูล หลัก
การของฐานข้อมูลแบบสัมพันธ์ นิยามข้อมูลระดับแนวคิดของวิธีวิเคราะห์ข้อมูลของไนเอน หรือ
ไนเอน (NIAM Conceptual schema) บทที่ 3 กล่าวถึงการออกแบบ และเมตาด้าสกี
มาของแบบจำลองข้อมูลไนเอน เมบิงอินฟอร์เมชัน ทั้งในลักษณะแบบจำลองของไนเอน และวิ
เลชันนระบบจัดการฐานข้อมูลที่ใช้เก็บเมตาด้าเบสและข้อมูลเมบิงทั้ง 21 ตาราง บทที่ 4
กล่าวถึง ไดอะแกรมรวมของระบบช่วยออกแบบฐานข้อมูลแบบสัมพันธ์ บทที่ 5 กล่าวถึงรายละเอียด
ของโปรแกรมโมดูลช่วยวาด ซึ่งพัฒนาอยู่บนระบบการเชื่อมประลาผู้ใช้แบบกราฟิก บทที่ 6
กล่าวถึงรายละเอียดของโปรแกรมโมดูลแปลงแบบจำลองไนเอนเป็นเมตาด้าเบส บทที่ 7
กล่าวถึงรายละเอียดของโปรแกรมโมดูลวิเคราะห์แบบจำลองไนเอน และกฎที่ใช้ในการตรวจ
สอบ บทที่ 8 กล่าวถึงขั้นตอนการติดตั้งระบบ ระบบคอมพิวเตอร์ที่ระบบต้องการ การใช้งาน
ระบบกับงานออกแบบแบบจำลองไนเอน บทที่ 9 กล่าวถึงเนื้อหาโดยสรุปของเนื้อหาทุกบทใน
ปริทัศน์บทนี้ และบทที่ 10 กล่าวถึงแนวทางการพัฒนาต่อของระบบช่วยออกแบบฐานข้อมูลแบบสัมพันธ์
กรณีพิเศษที่ระบบช่วยออกแบบยังไม่สามารถครอบคลุมได้ทั่วไป ภาคผนวก ประกอบด้วย
เครื่องมือที่ใช้ในการพัฒนาระบบ ได้แก่ ภาคผนวก ก. โปรแกรมรีซี ซึ่งเป็นพริคอมไพล์เลอร์ของ
ภาษาเอสควแอล (SQL) ที่อยู่ภายในภาษามะเจ้าหรือภาษาโฮส (host language) ที่เป็นภาษา
ซี ภาคผนวก ข. ไมโครซอฟท์วิโดว์ ซึ่งเป็นระบบเอมไวรอนเม้นต์การทำงานแบบกราฟิกที่
ใช้เป็นเครื่องมือในการพัฒนาโปรแกรมช่วยวาด ภาคผนวก ค. ระบบจัดการฐานข้อมูลอราเคิล
(ORACLE RDBMS) ซึ่งใช้เป็นระบบจัดการฐานข้อมูลที่ระบบช่วยออกแบบฐานข้อมูลนี้อ้างอิงถึง
และภาคผนวก ง. แสดงวิธีการติดตั้งเมตาด้าสกีและเมบิงอินฟอร์เมชัน

2.1 ระบบฐานข้อมูลแบบสัมพันธ์ (Relational database system)

หมายถึงระบบฐานข้อมูลซึ่งยึดเอาโมเดลแบบสัมพันธ์ (relational model) เป็นหลัก ในปัจจุบันฐานข้อมูลส่วนใหญ่เป็นระบบฐานข้อมูลแบบสัมพันธ์ทั้งสิ้น เนื่องจากเป็นโมเดลซึ่งง่ายต่อการทำความเข้าใจ และสามารถตอบสนองต่อคำถาม (query) ที่ซับซ้อนได้ดีกว่าโมเดลชนิดอื่น ตัวอย่างของระบบจัดการฐานข้อมูลแบบสัมพันธ์ ได้แก่

ผลิตภัณฑ์

DB2

SQL/DS

INGRES

ORACLE

Rdb/VMS

ผู้ขาย

IBM

IBM

Relational Technology Inc.

ORACLE Corp.

DEC

2.2 โมเดลแบบสัมพันธ์ (Relational model)

การที่จะกล่าวว่าระบบฐานข้อมูลใด เป็นโมเดลแบบสัมพันธ์นั้น ต้องครบถ้วนในองค์ประกอบทั้งสามส่วน อันได้แก่

1. โครงสร้างข้อมูล (Data structure)

เป็นไปตามคุณสมบัติของรีเลชัน (relations) (กล่าวโดยละเอียดในหัวข้อ "โครงสร้างข้อมูลแบบสัมพันธ์")

2. ความถูกต้องของข้อมูล (Data integrity)

เป็นไปตามกฎความถูกต้องทั้งสี่ : กล่าวโดยละเอียดในหัวข้อ "กฎของความถูกต้องของข้อมูล")

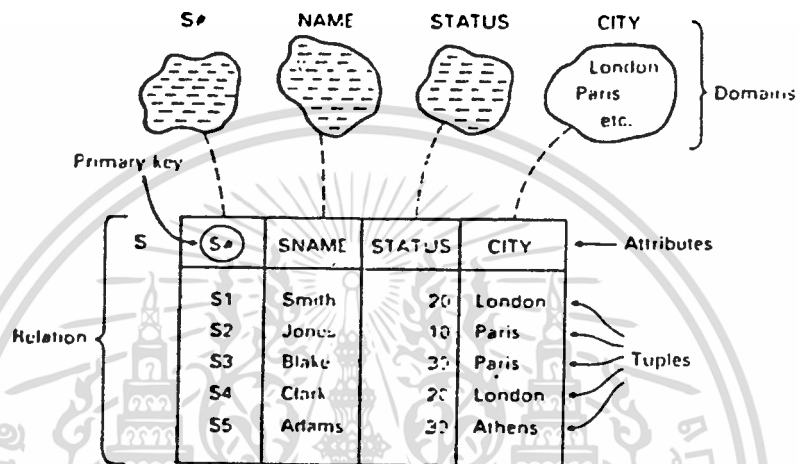
3. การจัดการข้อมูล (Data manipulation)

เป็นไปตามนิยามการจัดการข้อมูลในรีเลชัน (กล่าวโดยละเอียดในหัวข้อ "การจัดการ

การข้อมูล")

2.3 โครงสร้างข้อมูลแบบสัมพันธ์ (Relational data structure)

2.3.1 ตัวอย่างตารางความสัมพันธ์



รูปที่ 2.1 แสดงตารางความสัมพันธ์ข้อมูล supplier

จากรูปที่ 2.1 มีค่าเฉพาะที่จะใช้ในการเรียกขานส่วนต่าง ๆ ของตารางข้อมูล ดังนี้

- โดเมน (domain) หมายถึง หน่วยข้อมูลที่เก็บอยู่ในตารางความสัมพันธ์
- แอตทริบิวต์ (attribute) หมายถึง ชื่อคอลัมน์ของตารางความสัมพันธ์
- ทิวเพิล (tuple) หมายถึง แถวข้อมูลใน 1 เรคคอร์ดของตารางความสัมพันธ์
- ไพรมารีคีย์ (primary key) หมายถึง แอตทริบิวต์หรือกลุ่มแอตทริบิวต์ซึ่งมีข้อมูลที่มีลักษณะเฉพาะ กล่าวคือ ในเวลาใด ๆ ก็ตาม จะไม่ปรากฏว่ามีแถวข้อมูลสองแถวที่อยู่ในแอตทริบิวต์หรือกลุ่มแอตทริบิวต์ดังกล่าว มีข้อมูลเหมือนกันทุกประการ (รายละเอียดเรื่องไพรมารีคีย์อยู่ในหัวข้อ "ความถูกต้องของข้อมูล")

2.3.2 รีเลชัน (Relations)

หมายถึง ผลของการกระโดดของโดเมนที่เรียงในเซตเล็กอื่น ประกอบด้วยตัวของคี่ประกอบ 2 ส่วน คือ เฮดดิ้ง (heading) และ บอดี้ (body) โดยเรากำหนดให้ $D_1, D_2, \dots, D_n$



แทนโดเมนทั้งหลายในระบบงาน

- เฮดดิ้ง (heading) ประกอบขึ้นมาจากเซตที่มีขนาดคงที่ (fixed set) ของแอตทริบิวต์ $A_1, A_2, \dots, A_n$ โดย A_i คือ แอตทริบิวต์ที่แทนข้อมูลในโดเมนที่ D_i
- บอดี้ (body) ประกอบขึ้นจากเซตของทUPLE ที่มีขนาดแปรผันตามเวลาของรีเลชัน โดยแต่ละทUPLE ประกอบด้วยค่าข้อมูลของแอตทริบิวต์ต่าง ๆ เช่น $A_i: v_i$ แทนค่าที่ i ของแอตทริบิวต์ที่ i

หนึ่งทUPLE ประกอบขึ้นด้วยค่าข้อมูลของแอตทริบิวต์ที่อ้างอิงโดเมนต่างๆ ในตาราง จากรูปที่ 2.1 เขียนได้เป็น

(S# : 'S1')
(SNAME : 'Smith')
(STATUS : 20)
(CITY : 'London')

ตารางความสัมพันธ์ที่มีจำนวนแอตทริบิวต์ n แอตทริบิวต์ จะเรียกว่ามีดีกรีขนาด n (n degree) เรียกรีเลชันที่มีเพียงแอตทริบิวต์เดียวว่า อUnary (unary) รีเลชันที่มีสองแอตทริบิวต์ว่า ไบนารี (binary) และ ขนาดแอตทริบิวต์สามว่า เทอร์นารี (ternary) และหากมีดีกรีขนาด n ก็เรียกว่า เอ็นนารี (n-ary)

คุณสมบัติของรีเลชัน ที่พิจารณาจากข้อกำหนด 4 ประการ คือ

1. ต้องไม่มีทUPLE ที่ซ้ำกันภายในรีเลชัน
2. ลำดับของทUPLE ที่เก็บไม่มีความสำคัญในการเก็บ
3. ลำดับของแอตทริบิวต์ไม่มีความสำคัญในการเก็บ
4. ค่าข้อมูลของแต่ละแอตทริบิวต์ต้องเป็นค่าเดียว หรือ อะตอมมิกแวลู (atomic value)

2.3.2.1 ต้องไม่มีทUPLE ที่ซ้ำกันภายในรีเลชัน

เนื่องจากข้อนี้ของรีเลชันจึงประกอบขึ้นจากทUPLE หลายทUPLE มีลักษณะเป็นเซตทางคณิตศาสตร์ รีเลชันเป็นลูกข่ายที่เขียน การซ้ำกันของทUPLE จึงเหมือนกับการซ้ำกันของสมาชิกภายในเซต ซึ่งนิยามทางคณิตศาสตร์ ถัดนั้นจึงกำหนดคุณสมบัติขึ้น เพื่อให้โครงสร้างข้อมูลแบบสัมพันธ์

ทัศน์เป็นไปตามกฎทางคณิตศาสตร์ และก็เนื่องจากความเป็นหนึ่ง (ไม่มีซ้ำ) ของแต่ละทUPLE ในรีเลชันนี้เอง ทำให้เราสามารถกล่าวได้ว่า "ทุก ๆ รีเลชันจะต้องมีไพรมารีคีย์เกิดขึ้นเสมอ" เพราะอย่างน้อยที่สุดกลุ่มแอตทริบิวต์ที่ค่าข้อมูลมีความเป็นหนึ่ง ตามนิยามในหัวข้อ "ตัวอย่างตารางความสัมพันธ์" นั้น ก็คือเซตของแอตทริบิวต์ทั้งหมดในรีเลชันนั่นเอง

2.3.2.2 ลำดับของทUPLE ไม่มีความสำคัญในการเก็บ

จากนิยามทางคณิตศาสตร์ที่ว่าสมาชิกภายในเซต จะอยู่กระจัดกระจายไม่มีลำดับ บอด้ของรีเลชันซึ่งเป็นเซตที่มีสมาชิกคือ ทUPLE จึงไม่ให้ความสำคัญกับลำดับของทUPLE กล่าวคือ เมื่อมีการลบหรือเพิ่มทUPLE ก็ไม่มีผลต่อความถูกต้องของข้อมูลแต่อย่างใด แต่ในทางปฏิบัติ เวลาในการค้นหาข้อมูลก็ยังขึ้นอยู่กับตำแหน่งของทUPLE เป็นสำคัญ

2.3.2.3 ลำดับของแอตทริบิวต์ไม่มีความสำคัญในการเก็บ

ในนิยามเช่นเดียวกับคุณสมบัติในข้อที่สอง ลำดับของแอตทริบิวต์จึงไม่มีความสำคัญต่อการเก็บข้อมูลเช่นเดียวกัน สังเกตจากตัวอย่างรูปที่ 2.1 ข้อมูล " นายสมิทซึ่งมีเลขประจำตัว S1 มีสถานะที่ 20 และมีที่อยู่ที่ลอนดอน " สามารถบันทึกในรีเลชันว่า

S#	SNAME	STATUS	CITY
S1	Smith	20	London

หรือ

S#	CITY	SNAME	STATUS
S1	London	Smith	20

ก็ใช้ข้อมูลความจริงที่ไม่แตกต่างกัน

2.3.2.4 ค่าข้อมูลของแต่ละแอตทริบิวต์จะเป็นค่าเดียว หรือ อะตอมมิกแวลู (atomic value)

ข้อกำหนดสุดท้ายหมายถึง ค่าข้อมูลของแอตทริบิวต์ใด ๆ ต้องเป็นค่าเดียว หรือ อะตอมมิก แวลู หมายความว่าเมื่อมีการบ่งชี้รีเลชัน ชื่อแอตทริบิวต์ และทUPLEที่ต้องการแล้ว จะต้องได้ค่าข้อมูลแอตทริบิวต์ออกมาเพียงหนึ่งค่าเท่านั้น ข้อกำหนดอื่นสามารถกล่าวได้อีกอย่างหนึ่งว่า " รีเลชันใด ๆ จะต้องไม่มีกลุ่มซ้ำ (repeating group) อยู่ภายใน " ตัวอย่างที่แสดงตารางความสัมพันธ์ที่มีกลุ่มซ้ำ และรีเลชันที่มีค่าข้อมูลเป็นอะตอมมิกแวลู เป็นดังนี้

BEFORE	S#	PO		AFTER	S#	P#	QTY
		P#	QTY				
S1		P1	300	S1	P1	300	
		P2	200	S1	P2	200	
		P3	400	S1	P3	400	
		P4	200	S1	P4	200	
		P5	100	S1	P5	100	
		P6	100	S1	P6	100	
S2		P1	300	S2	P1	300	
		P2	400	S2	P2	400	
S3		P2	200	S3	P2	200	
S4		P2	200	S4	P2	200	
		P4	300	S4	P4	300	
		P5	400	S4	P5	400	

รูปที่ 2.2 แสดงตัวอย่างตารางที่มีกลุ่มซ้ำและ ไม่มี

จากตาราง BEFORE เป็นตารางที่มีกลุ่มซ้ำ เรียงตารางเช่นนี้ว่าตารางที่ไม่ نرمอล ไลซ์ (unnormalized) และตาราง AFTER ว่าตาราง نرمอล ไลซ์ (normalized) ตาราง نرمอล ไลซ์นี้เรียกได้อีกอย่างว่า ตาราง نرمอลฟอร์มที่ 1 (First normal form)

2.4 ความถูกต้องของข้อมูล (Data integrity)

ความถูกต้องของข้อมูลในรีเลชัน ขึ้นอยู่กับพฤติกรรมของข้อมูลที่มีต่อกฎความถูกต้อง (Integrity rules) ซึ่งมีอยู่ 2 กฎอันได้แก่

1. กฎความถูกต้องของเอนทิตี (entity integrity) ,
2. กฎความถูกต้องในการอ้างอิง (referencial integrity)

ก่อนที่จะกล่าวถึงรายละเอียดของกฎทั้งสอง จะได้กล่าวถึงความหมายของคำเฉพาะใน

การอ้างถึงเสียก่อน ดังต่อไปนี้

2.4.1 ไพรมารีคีย์ (Primary key)

ไพรมารีคีย์คือการมีพิเศษของแคนดิเดตคีย์ (candidate key) ความหมายของแคนดิเดตคีย์ที่อธิบายได้ตามนิยามสองประการ เมื่อกำหนดให้ R เป็นรีเลชันใด ๆ $A_1, A_2, \dots, A_n$ เป็นแอตทริบิวต์ของรีเลชันนั้น เซตของแอตทริบิวต์ K เท่ากับ $(A_i, A_j, \dots, A_k)$ จะเป็นแคนดิเดตคีย์ของ R ได้ก็ต่อเมื่อ

1. นิยามความเป็นหนึ่งเดียว (Uniqueness)

ในช่วงเวลาขณะใดขณะหนึ่ง จะไม่ปรากฏว่ามีทUPLEสองทUPLEในรีเลชัน R ที่มีค่าแอตทริบิวต์เท่ากันในแอตทริบิวต์ที่ A_i เท่ากันในแอตทริบิวต์ที่ $A_j \dots$ และเท่ากันในแอตทริบิวต์ที่ A_k

2. นิยามขนาดเล็กที่สุด (Minimally)

จะไม่สามารถตัดแอตทริบิวต์ใด ๆ ออกจากเซต K ได้ โดยไม่ขัดแย้งกับนิยามความเป็นหนึ่งเดียว

ไพรมารีคีย์ของรีเลชันใด ๆ ก็คือแคนดิเดตคีย์ตัวหนึ่งของรีเลชันนั้นที่ได้จากการกำหนดของผู้ออกแบบฐานข้อมูล เนื่องจากเป็นไปได้ว่ารีเลชันหนึ่งจะมีมากกว่าหนึ่งแคนดิเดตคีย์ จากรูปที่ 2.1 จะเห็นได้ว่าแอตทริบิวต์ SNAME ก็เป็นแคนดิเดตคีย์ เช่นกันแต่ในกรณีนี้ได้กำหนดให้เอาแอตทริบิวต์ S# เป็นไพรมารีคีย์ เรียก SNAME ว่าเป็น อัลเทอร์เนตคีย์ (alternate key) จากรีเลชัน 3 รีเลชันดังต่อไปนี้

S		P	
S#	SNAME	P#	PNAME
S1	Smith	P1	Nut
S2	John	P2	Screw

SP	S#	P#	QTY
	S1	P1	10
	S1	P2	20
	S2	P1	10

ไพรมารีคีย์ในรีเลชัน S คือ S# ไพรมารีคีย์ในรีเลชัน P คือ P# และไพรมารีคีย์ในรีเลชัน SP คือ S# และ P# ในลักษณะเช่นนี้เราเรียกแอตทริบิวต์ S# และ P# ในรีเลชัน SP ว่า ฟอว์เรนคีย์ (foreign key) ฟอว์เรนคีย์ หมายถึงแอตทริบิวต์ในรีเลชันหนึ่ง ที่มีสมาชิกเทียบเท่ากับแอตทริบิวต์เดียวกันในอีกรีเลชันหนึ่ง เรียกความสัมพันธ์ระหว่างแอตทริบิวต์ที่เป็นฟอว์เรนคีย์ กับไพรมารีคีย์ของรีเลชันอื่นว่า การอ้างอิง หรือเรฟเฟอว์เรนซ์ (reference)

2.4.2 กฎความถูกต้องของเอนิตี (entity integrity)

กล่าวว่า เอนิตีข้อมูลซึ่งหมายถึงค่าข้อมูลในแอตทริบิวต์ ซึ่งเป็นแอตทริบิวต์ไพรมารีคีย์ของรีเลชันใด ๆ ต้องไม่เป็น null (นัล หมายถึง ไม่นับถือค่า)

2.4.3 กฎความถูกต้องในการอ้างอิง (referencial integrity)

กล่าวว่า หาก FK คือแอตทริบิวต์ฟอว์เรนคีย์ของรีเลชัน R2 และ PK คือแอตทริบิวต์ไพรมารีคีย์ของรีเลชัน R1 แล้ว ค่าข้อมูลแต่ละค่าในแอตทริบิวต์ FK ในรีเลชัน R2 จะต้อง (ก) มีค่าเท่ากับค่าข้อมูลที่อยู่ในแอตทริบิวต์ PK ในรีเลชัน R1 ในทึพเพิลใดทึพเพิลหนึ่ง หรือ (ข) เป็น null ซึ่งหมายถึง ทุกค่าใน FK เป็น null ทั้งหมด

2.5 การจัดการข้อมูล (Data manipulation)

ข้อมูลภายในฐานข้อมูลจะถูกจัดการด้วยภาษาจัดการข้อมูลภายในฐานข้อมูล ภาษาจัดการฐานข้อมูลจะต้องเป็นไปตามนิยามของ รีเลชันแนล คอมพลีท (relational complete) กล่าวคือต้องสนับสนุนปฏิบัติการพื้นฐานทั้งสามในภาษา รีเลชันแนล (Relational

Complete) หรือ แคลคูลัสสัมพันธ์ (Relational Calculus) ซึ่งประกอบไปด้วยปฏิบัติการ 8 อย่างโดยแบ่งออกเป็น 2 กลุ่มได้ดังนี้

1. กลุ่มปฏิบัติการพื้นฐาน (Traditional Set Operations) ได้แก่ ผลคูณคาร์ทีเซียน หรือ ไบรด์ (CARTESIAN PRODUCT), ยูเนียน (UNION), อินเตอร์เซกชัน (INTERSECTION) และ ดิฟเฟอเรนซ์ (DIFFERENCE)

2. กลุ่มปฏิบัติการพิเศษ (Special Relational Operations) ได้แก่ ซีเล็ค (SELECT), โปรเจ็ค (PROJECT), จอยน์ (JOIN) และ ดีไวด์ (DIVIDE)

ลักษณะการทำงานของตัวปฏิบัติการต่าง ๆ

SELECT คือ การสร้างความสัมพันธ์จากการเลือกเอาข้อมูลในทัวเปิล ที่มีคุณสมบัติตามที่ระบุออกมาจากความสัมพันธ์ที่มีอยู่

PROJECT คือ การสร้างความสัมพันธ์โดยการเลือกเอาข้อมูลจากแอททริบิวต์ บางตัวตามที่ได้กำหนด ออกมาจากความสัมพันธ์ที่มีอยู่

PRODUCT คือ การสร้างความสัมพันธ์ขึ้นใหม่จากความสัมพันธ์ 2 ความสัมพันธ์ ซึ่งความสัมพันธ์ที่สร้างขึ้น จะประกอบด้วยข้อมูลที่เป็นไปได้ทั้งหมด ที่เกิดจากการนำข้อมูลในทัวเปิล ของความสัมพันธ์ทั้งสองมาจับคู่กัน

UNION คือ การสร้างความสัมพันธ์ขึ้นใหม่จากความสัมพันธ์ 2 ความสัมพันธ์ โดยความสัมพันธ์ที่สร้างขึ้น จะมีข้อมูลที่ประกอบไปด้วยข้อมูลของทั้งสองความสัมพันธ์

INTERSECT คือ การสร้างความสัมพันธ์จากความสัมพันธ์ 2 ความสัมพันธ์ที่กำหนดให้ โดยความสัมพันธ์ที่ได้ จะประกอบด้วยข้อมูลที่อยู่ในทัวเปิล ที่เหมือนกันของทั้งสองความสัมพันธ์ที่กำหนด

DIFFERENCE คือ การสร้างความสัมพันธ์ที่เกิดจากความสัมพันธ์ 2 ความสัมพันธ์ที่กำหนดให้ ซึ่งความสัมพันธ์ที่เกิดขึ้นใหม่จะประกอบด้วย ข้อมูล ในทัวเปิลทั้งหมดที่มีอยู่ในความสัมพันธ์อันแรกที่กำหนดให้ แต่ไม่มีอยู่ในความสัมพันธ์อันที่สอง

JOIN คือ การสร้างความสัมพันธ์ใหม่จากความสัมพันธ์ 2 ความสัมพันธ์ โดยความสัมพันธ์ที่สร้างขึ้นใหม่ จะประกอบด้วย ทัวเปิลที่เกิดจากการจับคู่ระหว่างทัวเปิลของทั้งสองความสัมพันธ์ ซึ่งการจับคู่ของทัวเปิลนี้ จะต้องถูกกำหนดเงื่อนไขอย่างหนึ่ง

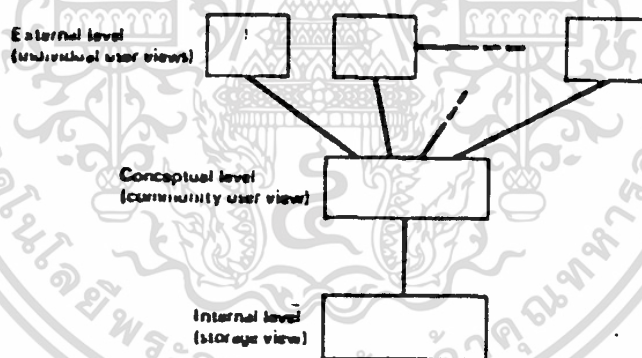
DIVIDE คือ การสร้างความสัมพันธ์จากความสัมพันธ์ 2 แบบ คือ ความสัมพันธ์แบบไบนารี (Binary) และความสัมพันธ์แบบยูนารี (Unary) ซึ่งความสัมพันธ์ที่สร้างขึ้นจะประกอบด้วยข้อมูล

ทั้งหมดของแอททริบิวต์แรกของความสัมพันธ์แบบไบนารี ที่มีข้อมูลในแอททริบิวต์ที่สองเหมือนกับข้อมูลทั้งหมดในความสัมพันธ์แบบยูนารี

2.6 สถาปัตยกรรมของระบบฐานข้อมูล (Architecture for a Database system)

สถาปัตยกรรมของระบบฐานข้อมูลประกอบด้วยส่วนประกอบสำคัญ 3 ระดับ ได้แก่ เรียกว่า 3 ระดับของข้อมูลสามระดับ (Three schemas) ได้แก่

1. ระดับภายใน (Internal level) หมายถึง ระดับชั้นข้อมูลที่ใช้แทนหน่วยเก็บระดับล่างสุดของฐานข้อมูล เช่น ไฟล์ , ลิงก์ลิสต์ เป็นต้น
2. ระดับภายนอก (External level) หมายถึง ระดับชั้นข้อมูลสำหรับผู้ใช้ ผู้ใช้จะทำงานอยู่บนระดับข้อมูลภายนอก โดยไม่สามารถรู้ถึงระดับชั้นข้อมูลระดับต่ำกว่า เช่น วิว
3. ระดับแนวคิด (Conceptual level) หมายถึง ระดับชั้นข้อมูลที่อยู่ระหว่างระดับภายนอก และระดับภายใน ใช้แทนโมเดลของข้อมูลในแง่ต่าง ๆ เช่น โมเดลแบบสัมพันธ์ โมเดลแบบไฮราคี หรือ โมเดลแบบเน็ตเวิร์ค



รูปที่ 2.3 แสดงแผนภูมิสถาปัตยกรรมของระบบฐานข้อมูล 3 ระดับ

<i>External (PL/I)</i>	<i>External (COBOL)</i>
DCL 1 EMP,	01 EMPC.
2 EMP# CHAR(6),	02 EMPNO PIC X(6).
2 SAL FIXED BIN(31);	02 DEPTNO PIC X(4).

Conceptual

EMPLOYEE	
EMPLOYEE_NUMBER	CHARACTER (6)
DEPARTMENT_NUMBER	CHARACTER (4)
SALARY	NUMERIC (5)

Internal

STORED_EMP	LENGTH=18
PREFIX	TYPE=BYTE(6),OFFSET=0
EMP#	TYPE=BYTE(6),OFFSET=6,INDEX=EMPX
DEPT#	TYPE=BYTE(4),OFFSET=12
PAY	TYPE=FULLWORD,OFFSET=16

รูปที่ 2.4 แสดงตัวอย่างระดับข้อมูลในระบบใช้งาน

2.7 การออกแบบฐานข้อมูลโดยใช้วิธีนอร์มอลไลซ์เซชัน (Normalization)

นอร์มอลไลซ์เซชันคือวิธีการแยกตารางฐานข้อมูลจากตารางรวม ออกเป็นตารางที่เรียกว่า Normal form (NF) ต่างๆ โดยเริ่มที่ตารางที่มีลักษณะของค่าเดี่ยว (Atomic value) หรือ ตารางนอร์มอลไลซ์ เรียกว่านอร์มอลฟอร์มที่ 1 (1NF) ใช้พยายามในการค้นหาแอตทริบิวท์ที่เกิดความสัมพันธ์ (dependent) กับแอตทริบิวท์อื่น แล้วทำการแยกแอตทริบิวท์ดังกล่าวนั้น ออกเป็นตารางในนอร์มอลฟอร์มใหม่ ไม่ว่าจะเป็น การขึ้นต่อกันของฟังก์ชัน (Functional dependency) การขึ้นต่อกันค่าข้อมูลหลายระดับ (Multivalued dependency) หรือ การขึ้นต่อกันในลักษณะร่วม (Join dependency) นอร์มอลฟอร์มต่างๆ ได้แก่ 2NF , 3NF , BCNF , 4NF , 5NF และ OAF การแปลงจากนอร์มอลฟอร์มหนึ่งไปเป็นอีกนอร์มอลฟอร์มหนึ่ง เป็นกระบวนการที่ค่อนข้างยุ่งยาก ใช้เวลานานในเวลาที่เป็นการวางแผนขนาดใหญ่ และขาดการนำไปพัฒนาเป็นโปรแกรม การนอร์มอลไลซ์ฐานข้อมูล 1 แอปพลิเคชัน อาจต้องใช้เวลานานนับเป็นปี ทั้งยังไม่อาจรับรองคุณภาพของฐานข้อมูลว่าดีพอ ซึ่งปกติแล้วจะทำการนอร์มอลไลซ์ก็ถึงเพียงแค่ BCNF เท่านั้น เพราะการทำมากกว่านี้ทำได้ยากมาก จึงสามารถเกิดความกำกวมของข้อมูลได้

2.8 แบบจำลองข้อมูลไนเยม (The NIAM Conceptual Schema)

ไนเยม (NIAM : Nijssen's Information Analysis Methodology) เป็นวิธีการในวิธีข้อมูลที่ให้แบบจำลองข้อมูลที่มีลักษณะต่างๆและมีความหมาย เพื่อจุดประสงค์ในการตรวจสอบความถูกต้องของระบบสารสนเทศ คิดค้นขึ้นโดย Prof.G.M.Nijssen ในขณะที่ทำงานอยู่ที่ Control Data ไนเยมได้รับการพัฒนาเรื่อยมา โดย Nijssen, Falkenberg และบุคคลอื่นๆ ไนเยมมีลักษณะเป็นแฟกต์โอเรียนท์โมเดลลิง (fact oriented modeling) ถึงแม้ว่าจะมีความคล้ายคลึงกับ เอนทิตีรีเลชันชิพโมเดล (Entity-relation model) ที่คิดค้นโดย Chen ในปี 1976 แต่มีความเชื่อว่า ไนเยมเข้าใจง่ายและเป็นธรรมชาติกว่า

2.8.1 ส่วนประกอบต่างๆของไนเยม

- 1.เอนทิตีไทป์ (Entity type) คือเซตของวัตถุที่สนใจ (object) หรือ สิ่งของ (thing) พื้นฐาน ที่เราต้องการกล่าวถึง เอนทิตีอาจจะเป็นวัตถุที่มีตัวตนเช่น คนชื่อ นาย ก. หรือ อาจจะเป็นวัตถุนามธรรม เช่น วิชา CS112 เป็นต้น ตัวอย่างของเอนทิตีไทป์ได้แก่ เอนทิตีไทป์ที่เป็นคนคนหนึ่ง จะเป็นเซตของคนทั้งหมดที่เราต้องการกล่าวถึง
- 2.เลเบลไทป์ (Label type) คือเซตของสิ่งที่อ้างอิงวัตถุที่เราเฉพาะเจาะจงได้ เช่น ชื่อคน ไนเก๋ นามวาร์กาเรต แดลเซอร์ เป็นชื่ออ้างอิงของเอนทิตีนายกรัฐมนตรี , รหัสนักศึกษา เป็นต้น
- 3.แฟกต์ไทป์ (Fact type) เป็นสิ่งที่อ้างอิงความสัมพันธ์ระหว่างสิ่งที่เราสนใจ นั่นคือ เป็นสิ่งที่ใช้สำหรับอ้างอิงความสัมพันธ์กันระหว่างเอนทิตี ซึ่งจะมีความสัมพันธ์กันแบบใดก็ได้
- 4.เรฟเฟอเรนซ์ไทป์ (Reference type) เป็นสิ่งที่ใช้อ้างอิงความสัมพันธ์ระหว่างเอนทิตีไทป์กับเลเบลไทป์
- 5.คอนสเทรนต์ (Constraint) หรือที่เรียกกันว่า ข้อจำกัดซึ่งเป็นสิ่งที่ใช้ในการกำหนดเงื่อนไขการยอมรับข้อมูลของประชากรที่ถูกต้องตามข้อกำหนดให้สามารถเก็บลงในฐานข้อมูล
 - 5.1 คอนสเทรนต์แบบยูนิคเนส (Uniqueness constraint)

สำหรับคอนสเทรนต์แบบยูนิคเนสแบ่งออกเป็น 2 อย่างคือ

 - 5.1.1 คอนสเทรนต์แบบยูนิคเนสภายใน (Intra-Fact Category Uniqueness Constraint) ใช้กำหนดจำนวนโวล (role) ที่น้อยกว่าหนึ่งสำหรับแฟกต์หนึ่งๆที่จะทำให้ข้อมูลแต่ละแถวของแฟกต์นั้นไม่ซ้ำกัน

5.1.2 คอนสเทรนต์แบบข้อยกเว้น (Extra-Fact Category Uniqueness Constraint) ใช้อ้างอิงสิ่งที่เราสนใจมาประกอบกัน เช่น การบุคคลอ้างอิงโดยการรวมชื่อกับนามสกุลด้วยกัน

5.2 คอนสเทรนต์แบบบังคับ (Mandatory constraint) อธิบายได้ดังนี้ เมื่อไรก็ได้มีผู้ให้ข้อมูลแบบบังคับปรากฏอยู่ โดยถ้าเรารู้ค่าข้อมูลของเอนทิตีแล้ว ค่าของข้อมูลของสิ่งอื่นที่มาเกี่ยวข้องกับเอนทิตีที่มีสัญลักษณ์แบบบังคับนั้น ต้องมีค่าของข้อมูล ตัวอย่างเช่น เมื่อปรากฏชื่อของบุคคลหนึ่ง บุคคลนั้นต้องมีนามสกุลปรากฏอยู่ด้วย

5.3 คอนสเทรนต์แบบกีดกัน (Exclusion constraint) เป็นข้อกำหนดว่า สมาชิกทุกตัวภายในโรลของเอนทิตีหนึ่งจะไม่ใช่สมาชิกภายในอีกโรลหนึ่งของเอนทิตีเดียวกัน

5.4 คอนสเทรนต์แบบอควิลิตี้ (Equality constraint) เป็นข้อกำหนดว่า สมาชิกทุกตัวภายในโรลของเอนทิตีหนึ่งต้องเป็นสมาชิกภายในอีกเอนทิตีในอีกโรลหนึ่งของเอนทิตีเดียวกัน

5.5 คอนสเทรนต์แบบสับเซต (Subset constraint) เป็นข้อกำหนดว่า เซตของสมาชิกทุกตัวภายในโรลของเอนทิตีหนึ่งต้องเป็นสับเซตกับเซตของสมาชิกภายในอีกโรลหนึ่งของเอนทิตีเดียวกัน

5.6 คอนสเทรนต์แบบก่อกวน (Occurrence frequency constraint) เป็นข้อกำหนดความถี่หรือจำนวนครั้งที่เอนทิตีใดๆ ที่ต่ออยู่กับโรลที่กำกับด้วยคอนสเตรนต์แบบนี้จะถูกเก็บข้อมูล

5.7 คอนสเทรนต์แบบเรนจ์ (Range constraint) เป็นคอนสเตรนต์แบบเอนทิตีไทม์ประเภทหนึ่ง เป็นข้อกำหนดขอบเขตของข้อมูลในแต่ละเอนทิตี ในแบบช่วงของจำนวนข้อมูล

5.8 คอนสเทรนต์แบบสมาชิก (Membership constraint) เป็นคอนสเตรนต์แบบเอนทิตีไทม์ประเภทหนึ่ง เป็นการกำหนดลักษณะของข้อมูลหรือประชากรในเอนทิตี

การสร้างแบบจำลองข้อมูลจะมีพื้นฐานจากการกำหนดตัวอย่างข้อมูลขึ้นมา และนำมาผ่านกระบวนการ (Procedure) ที่ได้กำหนดไว้เป็นขั้นตอนที่แน่นอนแล้ว เราจะได้แผนภาพที่มีความหมายในการแทนข้อมูลตัวอย่างข้อมูลได้ และเนื่องจากแบบจำลองข้อมูลนี้คิดค้นครั้งแรกโดย Nijssen ดังนั้นจึงเรียกแบบจำลองข้อมูลนี้ว่า NIAM (Nijssen's Information Analysis Methodology)

ทั้งงานนี้และงานนี้ได้ถูกนำไปพัฒนาต่อโดย Nijssen และ T.A. Halpin ที่มหาวิทยาลัย Ulster และเนื่องจากวิธีการที่ใช้นี้จะเน้นที่แฟกต์ไทม์ (Fact Type) เป็นหลัก

ดังนั้นอาจจะเรียกแบบจำลองข้อมูลในแอมว่า fact-oriented modeling และแม้ว่าในแอมมีความคล้ายคลึงกับโมเดลอีอาร์ (Entity-Relationship Model) และอาจจะมีความสัมพันธ์กันบางส่วนก็ตาม แต่ในความเป็นจริงแล้วแบบจำลองข้อมูลในแอมนั้นจะสามารถที่จะแสดงความหมายได้ดีกว่า ทำให้สามารถทำความเข้าใจได้ง่ายกว่าและเป็นธรรมชาติกว่า นอกจากนี้แบบจำลองข้อมูลในแอมยังสามารถที่จะแปลงรูปไปเป็น 5NF (Fifth Normal Form) ได้ทันที ในขณะที่โมเดลอีอาร์ ทำได้เพียง 3NF เท่านั้น ดังนั้นเราสามารถจะใช้ในแอมเพื่อช่วยในการออกแบบฐานข้อมูลแบบรีเลชันแนล โดยใช้ทฤษฎี "นอร์มอลไลซ์เชชัน" ได้อีกด้วย

2.8.2 ขั้นตอนการออกแบบแบบจำลองข้อมูลในแอม

จากที่ได้กล่าวมาข้างต้นเราอาจจะมองว่าสามารถออกแบบได้ง่าย อย่างไรก็ตามหลักการของการออกแบบ แบบจำลองข้อมูล หากจะทำให้มีความถูกต้องก็จะเป็นงานที่ซับซ้อนไม่น้อย ดังนั้นเราจะแบ่งออกเป็นงาน 9 ขั้นตอนด้วยกัน ดังนี้

ขั้นตอนที่ 1 จากตัวอย่างที่ "ความจริงพื้นฐาน" (Elementary Facts)

งานในขั้นตอนแรกของการออกแบบ ก็คือเราจะต้องหาตัวอย่างข้อมูลที่ใช้งานอยู่ในระบบ ตัวอย่างข้อมูลที่เข้าหรือออก รายงาน (Output Report) และแบบฟอร์มที่ใช้ในการรับข้อมูลเข้าสู่ระบบ (Input Form) ซึ่งอาจจะอยู่ในรูปของตารางภาพ หรือข้อความใดๆ ก็ตามในตารางที่ 2.5 นั้นเป็นตัวอย่างของข้อมูลของตารางเรียน

<i>Tutorial group</i>	<i>Time</i>	<i>Room</i>	<i>Student#</i>	<i>Student name</i>
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
			348112	Bloggs FB
..	..	..	..	..

รูปที่ 2.5 แสดงตัวอย่างข้อมูลของตารางเรียน

จากนั้นเราก็นำข้อมูลทั้งหมดมาได้ ซึ่งในที่นี้ได้แก่ข้อมูลในตารางดังกล่าวมาเขียนให้อยู่ในรูปของ "ความจริงพื้นฐาน" (Elementary Facts) เช่นจากข้อมูลในบรรทัดแรกเราสามารถเขียนได้เป็น

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

Tute group A is held in Room CS-718.

Student 302156 belongs to Tute group A.

Student 302156 has Name 'Bloggs FB'.

จะเห็นได้ว่าการเขียนในรูปแบบนี้จะสามารถแสดงความสัมพันธ์ของข้อมูลได้มากกว่าแบบตาราง เช่น จะเห็นได้ว่านักศึกษาที่มีชื่อและเลขประจำตัวที่อยู่ในแถวเดียวกันหมายถึงคนเดียวกัน อย่างไรก็ตาม แม้ว่าข้อมูลจะเป็นแบบที่ขึ้นอาจจะเป็นแผนภูมิต่างๆ แต่งานในที่นี้เราก็จะต้องนำมาเขียนเป็น "ความจริงพื้นฐาน" ทั้งหมด

คำว่า "ความจริงพื้นฐาน" นั้นอาจจะถูกมองว่าเป็น "วัตถุ กระทำ สิ่งต่างๆ" (Particular Object Play Particular Role) แต่ประโยคที่ว่า Ann smoke. นั้นเราสามารถมองได้ว่า Ann (Object) กระทำ smoke (Role) นอกจากนั้น "ความจริงพื้นฐาน" ยังจะต้องเป็นข้อมูลพื้นฐานที่ไม่สามารถแบ่งแยกได้

แต่ก่อนที่เราจะกล่าวถึง "ความจริงพื้นฐาน" ในด้านถัดไป เราจะกล่าวถึงนิยามของคำต่างๆ ที่เราจะใช้กันต่อไป โดยเริ่มจากคำว่าเอนทิตี (Entity) เอนทิตีเป็นหน่วยพื้นฐานที่สุดของระบบข้อมูลที่เรากำลังสนใจ เช่น ในระบบข้อมูลของมหาวิทยาลัย นักศึกษา ภาค-วิชา อาจารย์ ล้วนจัดเป็นเอนทิตี และเรียกชื่อของเอนทิตีว่าเอนทิตีไทป์ (Entity Type) ซึ่งมีสมาชิกเป็นตัวอย่างเอนทิตี (Entity Instance) แต่ละตัวอย่างเอนทิตีจำเป็นต้องมีชื่อเรียกซึ่งอาจเป็นชื่อ รหัส อย่างใดอย่างหนึ่ง เช่น นักศึกษาจะเรียกโดยใช้รหัสนักศึกษาเป็นชื่อเรียกการกำหนดว่าแต่ละชนิดเอนทิตีควรใช้ชื่ออะไรเป็นชื่อเรียก ทำได้โดยระบุแลเบลไทป์ (LabelType) สำหรับเอนทิตีนั้น เช่นชนิดเอนทิตี Person อาจจะใช้แลเบลไทป์ surname เป็นชื่อเรียก

ดังนั้นสำหรับเอนทิตีที่ว่า "นักศึกษาเลขประจำตัว 311308 ลงเรียนในวิชา CS112" หากจะเขียนให้อยู่ในรูปของ "ความจริงพื้นฐาน" ให้เราสรุป เราสามารถเขียนให้อยู่ในรูปของประโยคภาษาธรรมชาติได้ดังนี้

The STUDENT with student# '311308'

studies

the SUBJECT with code 'CS112'.

โดยที่ทั้ง STUDENT และ SUBJECT เป็น เอนติตี้ไทป์

student# และ code เป็น เลเบลไทป์

'311308' และ 'CS112' เป็น Instance

และ studies เป็น Role

และจากประโยคข้างต้นจะสังเกตว่าเราจะเขียนเอนติตี้ไทป์คู่กับเลเบลไทป์เสมอ อย่างไรก็ตาม เราอาจเขียนความสัมพันธ์ข้างต้นเสียใหม่ ดังนี้

The SUBJECT with code 'CS112'

was studied by

the STUDENT with student# '311308'

ซึ่งจะเห็นได้ว่าการเขียนในทั้งสองรูปแบบสามารถแทนความจริงเดียวกันได้ โดยในประโยคหลังนี้ was studied by เป็นไวยากรณ์ โดยที่เราจะเรียกการเขียนประโยคที่ต่างกันนี้ว่ามี "Surface Structure" ที่ต่างกัน หากแต่มี "Deep Structure" อันเดียวกัน ดังนั้นหากจะเขียนความสัมพันธ์ดังกล่าวในรูปแบบใดรูปแบบหนึ่ง ก็คงจะเป็นการไม่ถูกต้องนัก และเพื่อให้สื่อถึงความสัมพันธ์เดียวกันทุกครั้ง เราจะเขียนความจริงนี้ใหม่ในรูปแบบต่อไปนี้

Study ([The STUDENT with student# '311308', studies],

[The SUBJECT with code 'CS112', was studied by])

โดย Study เป็นเฟกไทป์

ที่เราได้กล่าวมาทั้งหมดนั้น เป็นการกล่าวถึงการใช้การเปลี่ยนจากตัวข้อมูลที่อยู่ในรูปแบบต่างๆ ให้อยู่ในรูปแบบของ "ความจริงพื้นฐาน" ซึ่งเป็นรูปแบบที่สามารถแสดงความสัมพันธ์ของข้อมูลแต่ละตัวได้

ขั้นตอนที่ 2 แบบนิยามของแบบจำลองข้อมูลระดับแนวคิด

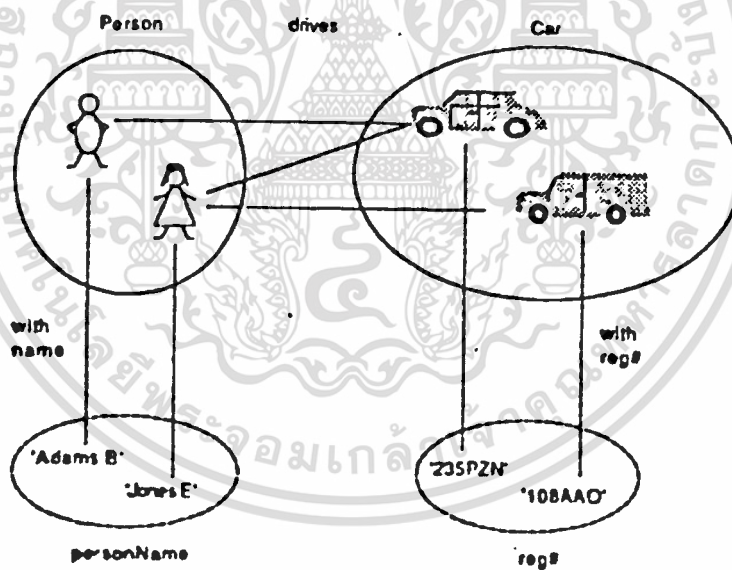
จากบทนำในขั้นตอนที่ 1 การนำ "ความจริงพื้นฐาน" ที่เราได้มาจากขั้นตอนที่ 1 มาวาดเป็นแผนภาพ ทั้งนี้เพราะได้กล่าวมาแล้วว่าโมเดลเป็นแบบจำลองแทนความหมาย (Semantic

Modelling) ดังนั้นการที่อธิบายความสัมพันธ์ของข้อมูลด้วยแผนภาพจะให้ความชัดเจนมากกว่า และเพื่อที่จะสามารถอธิบายได้ชัดเจนยิ่งขึ้น เราจะสมมติข้อมูลขึ้นมาชุดหนึ่ง ซึ่งเป็นข้อมูลว่า ใครขับรถเลขทะเบียนอะไรดังตารางที่ 2.1

DRIVES

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

ตารางที่ 2.1 แสดงตัวอย่างข้อมูลความสัมพันธ์ของรถและคนขับ

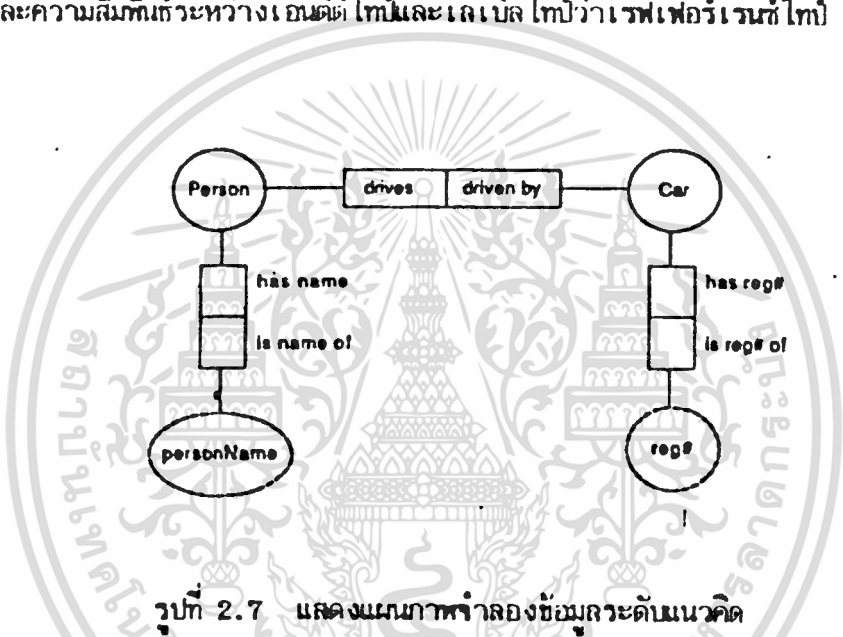


รูปที่ 2.6 แสดงความสัมพันธ์ของข้อมูลในตารางด้วยรูปภาพ

จากข้อมูลในตารางเราสามารถนำมาเขียนเป็น "ความจริงพื้นฐาน" ได้ดังนี้

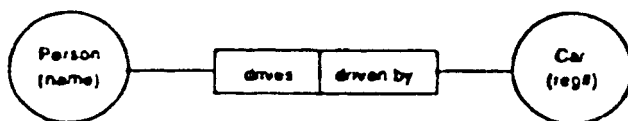
The Person with Name 'Adams B' drives the Car with reg# '235PZN'.
 The Person with Name 'Jones E' drives the Car with reg# '235PZN'.
 The Person with Name 'Jones E' drives the Car with reg# '108AAQ'.

ซึ่งแสดงให้เห็นว่ารูปที่ 2.6 และจาก "ความจริงพื้นฐาน" เราก็นำมาเขียนเป็นแผนภาพที่แสดงแนวคิด (Conceptual Schema Diagram) ได้ดังรูปที่ 2.7 และจากแผนภาพดังกล่าวจะเห็นได้ว่าเราได้แทนเอนทิตีใหม่โดยใช้วงรีที่เส้นทึบ และแทนเลเบลที่โยงด้วยวงรีที่เส้นประ และเขียนโวลต์ด้วยวงรีที่เส้นทึบ โดยมีเส้นลากจากเอนทิตีไปยังโวลต์เพื่อแสดงความสัมพันธ์ และเราจะเรียกความสัมพันธ์ระหว่างเอนทิตีใหม่ทั้งสองโดยผ่านโวลต์ทั้งสองว่า แฟกต์ไทป์ (Fact Type) และความสัมพันธ์ระหว่างเอนทิตีใหม่และเลเบลใหม่ว่า เรฟเฟอเรนซ์ไทป์ (Reference Type)



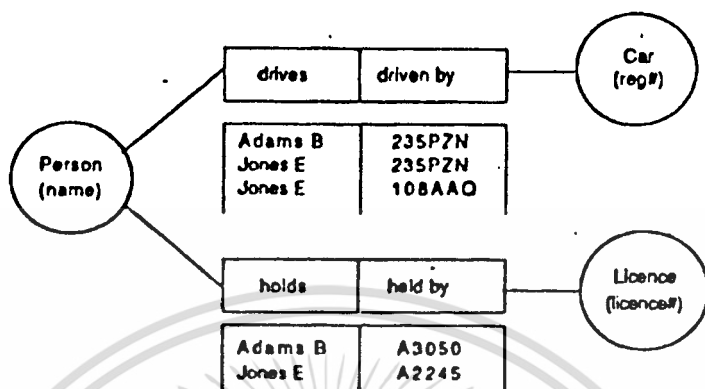
รูปที่ 2.7 แสดงแผนภาพจำลองข้อมูลระดับแนวคิด

อย่างไรก็ตามจะสังเกตได้ว่า สมาชิกแต่ละตัวในเอนทิตีใหม่จะมีความสัมพันธ์กับสมาชิกในเรฟเฟอเรนซ์ไทป์แบบ 1:1 เช่นในเอนทิตีใหม่แต่ละ Person จะมีเพียง 1 ชื่อและแต่ละชื่อก็จะหมายถึงคนคนหนึ่ง ดังนั้นเราสามารถเขียนแบบภูมิในรูปที่ 2.7 ได้ใหม่ดังในรูปที่ 2.8 โดยใส่เรฟเฟอเรนซ์ไทป์ไว้ในวงเล็บเพื่อที่จะเขียนได้ง่ายขึ้น โดยสามารถสื่อความหมายได้คงเดิม



รูปที่ 2.8 แสดงการเขียนแบบจำลองข้อมูลที่มีเรฟเฟอเรนซ์ไทป์เป็นแบบ 1:1

สำหรับวิธีการตรวจสอบความถูกต้องของแผนภูมิที่เราวาดนั้น สามารถทำได้โดยการใส่ตัวอย่างลงในแบบจำลองข้อมูล โดยจะเรียกแผนภูมิที่มีตัวอย่างนี้ว่า Schema-Base Diagram ดังรูปที่ 2.9 จะเห็นได้ว่าการเขียนความสัมพันธ์ในรูปแผนภูมิต้นแนวคิดนั้นสามารถมองเห็นความสัมพันธ์ได้ดีกว่า



รูปที่ 2.9 แสดง Schema-Based Diagram

Student	Subject	Rating
Adams	CS100	4
Brown	CS100	4
Brown	CS112	5

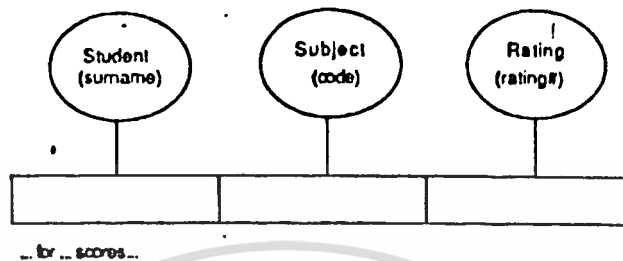
ตารางที่ 2.2 แสดงตัวอย่างข้อมูลของผลการเรียนนักศึกษา

ลองมาดูกันทีตัวกว้างโดยสมมติข้อมูลดังในตารางที่ 2.2 ซึ่งสามารถเขียนให้อยู่ในรูปของ "ความจริงพื้นฐาน" ได้ดังนี้

Student (surname) 'Adams' for Subject (code) 'CS100'
score: Rating (rating) 4.

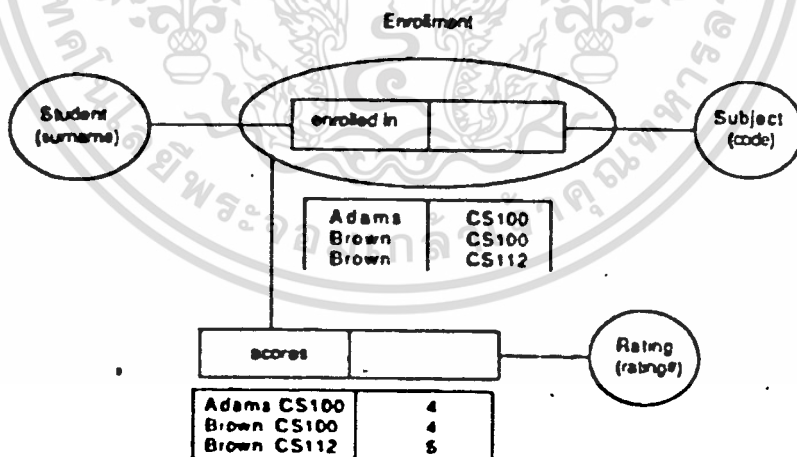
จึงจะ "ความจริงพื้นฐาน" นี้สามารถนำมาเขียนเป็นแผนภูมิต้นแนวคิดได้ดังรูปที่ 2.10

ซึ่งเป็นแฟกต์ไทม์ที่มี 3 โรล และแฟกต์ไทม์ที่มีจำนวนโรลต่างกันนี้ ก็จะมีชื่อเรียกต่างกัน เช่น จะเรียกแผนภูมิที่ประกอบด้วย 2 โรล ว่าแฟกต์ไทม์แบบไบนารี (Binary Fact Type) และ 3 โรล ว่า แฟกต์ไทม์แบบเทอร์นารี (Ternary Fact Type) เป็นต้น



รูปที่ 2.10 แสดงแฟกต์ไทม์ของผลการเรียน

อย่างไรก็ตาม การเขียนในลักษณะเช่นนี้นั้น อาจจะไม่เป็นการถูกต้องนักเพราะชนิดเอนทิตี Rating ถือได้ว่ามีใช้ความสัมพันธ์กับเพียงชื่อวิชาหรือวิชาที่นักศึกษาอย่างใดอย่างหนึ่งเท่านั้น แต่มีความสัมพันธ์กับทั้งเอนทิตีไทม์ Student และเอนทิตีไทม์ Subject เช่นนักศึกษาที่มีชื่อ Adams และลงเรียน CS100 เท่านั้น จึงจะมีคะแนนเป็น 4 จะเห็นได้ว่าเป็นความสัมพันธ์ร่วมกันซึ่งเราสามารถเขียนได้ใหม่ในรูปที่ 2.11 และเราเรียกความสัมพันธ์ในลักษณะเช่นนี้ว่าเนสติ้ง



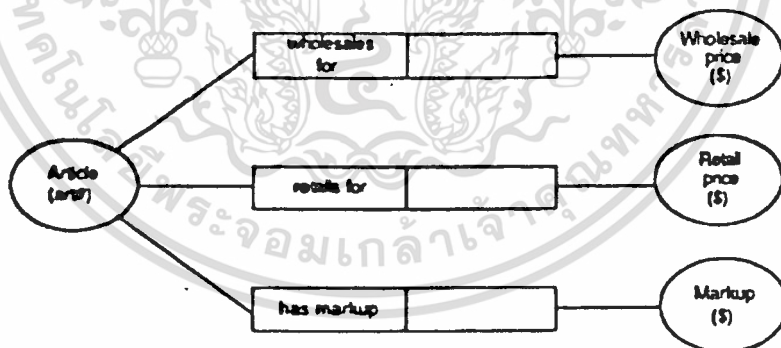
รูปที่ 2.11 แสดงการเขียนแฟกต์ไทม์ของผลการเรียนในรูปของ Nesting

ขั้นตอนที่ 3 ปรับปรุงแผนภาพและหาแฟกต์ไทป์ในบทหาได้ (derived fact type)

การทำงานในขั้นตอนนี้มีจุดประสงค์เพื่อที่จะหาว่ามีเอนติตี้ไทป์ใดบ้าง ที่เราสามารถที่จะลดลงได้บ้าง เพื่อให้เห็นได้ชัดเจนเราลองมาดูตัวอย่างในตารางที่ 3-4 ซึ่งเป็นรายการขายสินค้าและเงินทอน และจากตารางดังกล่าว เราสามารถนำมาเขียนเป็นแผนภูมิระดับแนวคิดได้ดังรูปที่ 3-8 ซึ่งจากรูป แผนภาพประกอบด้วยเอนติตี้ไทป์ 4 ชนิด หากแต่พิจารณาแต่ละเอนติตี้ไทป์ให้ดีแล้วจะเห็นว่า ที่จริงแล้วเอนติตี้ไทป์ Wholesale price, Retail price, Markup นั้นสามารถจะรวมเป็นเอนติตี้ไทป์ได้เช่นกันได้

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

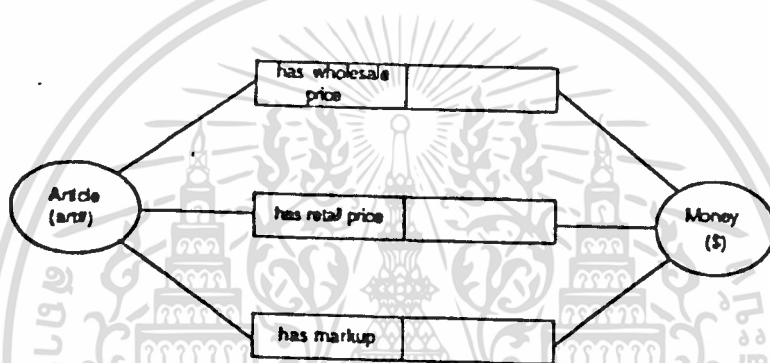
ตารางที่ 2.3 แสดงตัวอย่างข้อมูลราคาการขายสินค้า



รูปที่ 2.12 แสดงแผนภาพจำลองข้อมูลระดับแนวคิด

แผนภาพจากนั้นหาว่าเรายังคงเหลือข้อมูลในบรรทัดแรกของตาราง จะเห็นได้ว่าเงินทอนนั้นเป็น

ข้อมูลที่สามารถหาได้จากเงินที่จ่ายกับราคาขายตามสูตร $\text{Markup} = \text{Retail price} - \text{Wholesale price}$ ในลักษณะเช่นนี้หากเราเก็บข้อมูลทั้ง 3 ตัวแล้ว ก็จะเป็นการเก็บแบบซ้ำซ้อน และต้องคอยรักษาความถูกต้องของข้อมูลตามความสัมพันธ์อีกด้วย ดังนั้นเพื่อที่จะจัดปัญหาดังกล่าว เราจะเก็บข้อมูลเพียง Retail price และ Wholesale price เท่านั้น และเก็บ Markup อยู่ในรูปของกฎของความสัมพันธ์แทน การที่เก็บเช่นนี้จะมีข้อดีคือ จะสามารถป้องกันความผิดพลาดได้กล่าวคือ หากเราเก็บข้อมูลทั้งสามตัว หากค่าของเงินทอนของสินค้าตัวหนึ่งไม่เท่ากับค่าเงินที่จ่ายลบด้วยราคาขายก็จะเป็นการขัดแย้งกันของข้อมูล นอกจากนั้นยังประหยัดเนื้อที่ในการเก็บอีกด้วย และเราจะเรียกแฟกต์ไทม์แบบที่เก็บเป็นกฎนี้ว่าแฟกต์ไทม์แบบหาได้ (Derived Fact Type) และระบุในแผนภาพโดยเขียนเครื่องหมายดอกจันไว้ที่ด้านล่างของชนิดความจริงนั้น ดังนั้นจากแผนภาพในภาพที่ 2.12 เราสามารถนำมาเขียนใหม่ได้ดังภาพที่ 2.13



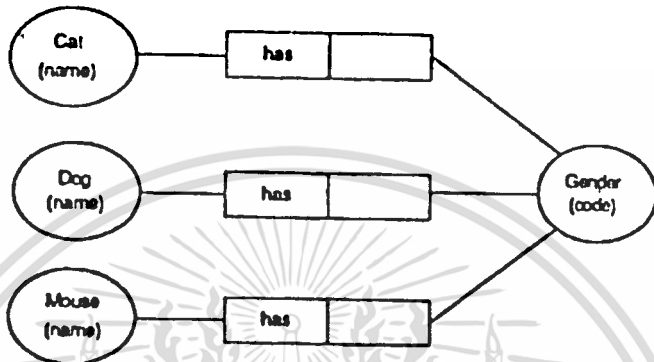
รูปที่ 2.13 แสดงแผนภาพจำลองข้อมูลระดับแนวคิด

ต่อไปเราลองมาดูแผนภาพของแบบจำลองข้อมูลระดับแนวคิดอีกแบบหนึ่งที่แสดงไว้ในรูปที่ 2.14 จากรูปจะเห็นว่าเรามีเอนทิตีไทม์ Cat, Dog และ Mouse ซึ่งเอนทิตีไทม์ทั้งสามนี้ต่างมีเลเบลไทม์ที่เหมือนกันคือ Name และยังมีแฟกต์ไทม์ที่เหมือนกันอีกด้วย ในภาพนี้เช่นนี้เราจะรวมเอนทิตีไทม์ Cat, Dog และ Mouse เข้าด้วยกัน เป็นเอนทิตีไทม์เดียวคือ Animal ดังแสดงในภาพที่ 2.15

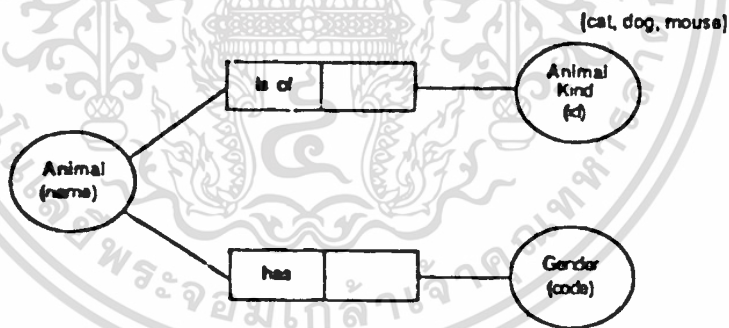
อย่างไรก็ตามจะสังเกตได้ว่าแบบจำลองข้อมูลในภาพที่ 2.14 นั้นอนุญาตให้สัตว์ประเภทต่างๆ มีชื่อที่ซ้ำกันได้ แต่แบบจำลองข้อมูลในภาพที่ 2.15 ไม่อนุญาตให้ทำเช่นนั้น ดังนั้นหากเราจะใช้แบบจำลองข้อมูลในรูปแบบนี้ ก็จะต้องทำการเปลี่ยนชื่อของสัตว์เล็กน้อยเพื่อให้ซ้ำกันและเพื่อที่จะแก้ปัญหาที่เช่นนี้ เราจะสร้างเอนทิตีไทม์ใหม่ชื่อ AnimalKind โดยกำหนดให้มีชนิดเลเบล

เป็น id เพื่อใช้ระบุชนิดของสัตว์ ในลักษณะเช่นนี้ก็จะยอมให้มีชื่อที่ซ้ำกันได้เช่นกัน

จากแบบจำลองข้อมูลในระดับแนวคิดที่สร้างขึ้นมานี้ จะเห็นได้ว่าง่ายกว่าแบบเก่ามา และยังมีพื้นที่ที่น้อยกว่าอีกด้วย และนอกจากนั้นหากเรามีชนิดของสัตว์ที่เพิ่มขึ้น เช่นอาจจะเพิ่ม Bird เข้าไป แบบจำลองข้อมูลใหม่นี้ก็ยังสามารถที่จะรองรับได้ ขณะที่แบบเก่าทำไม่ได้ และท้ายสุดเมื่อเราแปลงจากแบบจำลองข้อมูลไปเป็นตารางในฐานข้อมูลแบบสัมพันธ์ก็ยังสามารถได้จำนวนตารางที่น้อยกว่าอีกด้วย



รูปที่ 2.14 แสดงรูปแบบแนวคิดของแบบจำลองข้อมูล



รูปที่ 2.15 แสดงผลของการนำขั้นตอนที่ 3 มาใช้แก้ไข

จากที่กล่าวมาทั้งหมดในขั้นตอนที่ 3 นี้สามารถที่จะสรุปได้ดังต่อไปนี้

- หากมีเอนทิตีเดียวที่เป็นสมาชิกของ 2 เอนทิตีใดให้ทำการรวมทั้ง 2 เอนทิตีนั้น

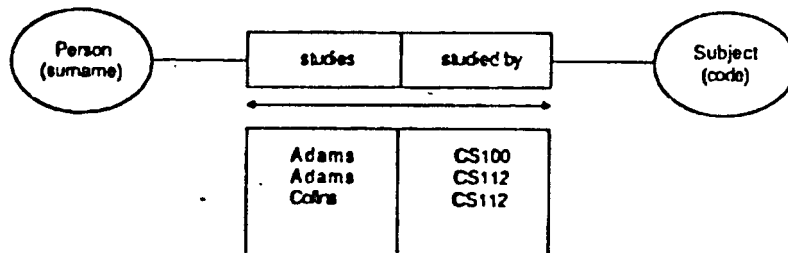
- หากมีเอนทิตีของเอนทิตีไทป์ที่ต่างกัน แต่มีความหมายเดียวกัน ให้ทำการรวมทั้ง 2 เอนทิตีนั้น
- หากตัวอย่างข้อมูลของเอนทิตีไทป์ที่ต่างกันมี role ที่เหมือนกัน ให้รวมชนิดเอนทิตีทั้งสอง และหากจำเป็นให้เพิ่มแฟกต์ไทป์อีก เพื่อรักษาความแตกต่างเดิมไว้
- หากแฟกต์ไทป์ใดที่สามารถที่จะหาได้จากแฟกต์ไทป์อื่น ให้ใส่เครื่องหมายดอกจันไว้ที่ชนิดความจริงนั้น

ขั้นที่ 4 คอนสเตรนทแบบยูนิคเนส (Uniqueness Constraint)

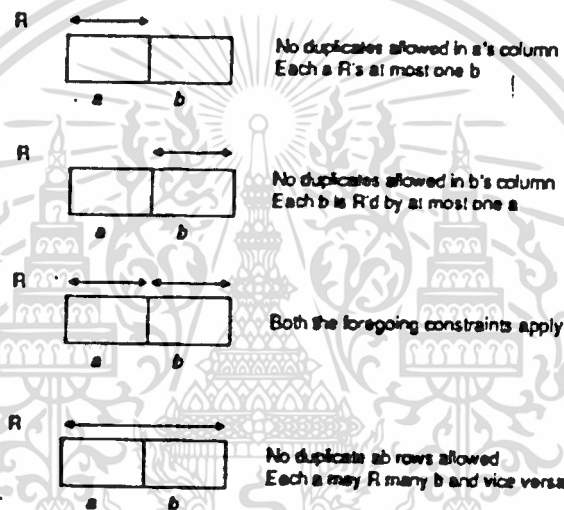
ที่ผ่านมาเราได้ทำการสร้างแผนภาพระดับแนวคิดขึ้น ทั้งในแบบที่เป็นเก็บจริง (store fact type) และแบบหาได้ (derive fact type) สำหรับในขั้นตอนที่เหลือนี้เราจะกล่าวถึงกฎข้อจำกัดหรือคอนสเตรนท (constraint) โดยในขั้นตอนนี้เราจะกล่าวถึงคอนสเตรนทแบบยูนิคเนส (Uniqueness Constraint) ซึ่งเป็นคอนสเตรนทที่สำคัญมาก ซึ่งจะมีบทบาทมากในภายหลังในขั้นตอนการแปลงจากฐานข้อมูลระดับแนวคิด (Conceptual Schema) เป็นฐานข้อมูลแบบสัมพันธ์ (Relational Schema)

สำหรับกฎข้อบังคับ ข้อที่บอกอยู่แล้วโดยมันจะทำหน้าที่ควบคุมความถูกต้องของฐานข้อมูล (static constraints apply to every possible state of database) เช่นอาจจะกำหนดว่า "นักศึกษาแต่ละคนจะสังกัดได้เพียงภาควิชาเดียวเท่านั้น" และโดยเฉพาะคอนสเตรนทแบบยูนิคเนสทุกๆ แฟกต์ไทป์ (store) จะต้องมียูนิคเนส บังคับอยู่อย่างน้อย 1 ตัว เพราะในแต่ละแฟกต์ไทป์นั้นจะต้องไม่มีข้อมูล 2 แถวที่ซ้ำกันเกิดขึ้น

สำหรับการเขียนคอนสเตรนทแบบยูนิคเนสนั้นจะเขียนโดยใช้ลูกศรที่มี 2 หัว เช่น สำหรับแฟกต์ไทป์ของข้อมูลที่ว่า นักศึกษาแต่ละคนจะลงเรียนได้หลายวิชา และแต่ละวิชาก็มีนักศึกษาได้หลายคนจะสามารถเขียน ได้ดังรูปที่ 2.16 ซึ่งจะหมายถึงว่านักศึกษาหนึ่งคนจะลงเรียนในรายวิชาหนึ่งได้เพียงครั้งเดียวเท่านั้น ดังนั้นเมื่อรวมข้อมูลของทั้ง 2 คอลัมน์แล้วจะไม่ซ้ำ (Unique) แต่ข้อมูลในแต่ละคอลัมน์จะมีการซ้ำได้ และเรียกความสัมพันธ์ของแฟกต์ไทป์แบบนี้เป็นแบบ many to many สำหรับความสัมพันธ์ในแบบต่างๆ ของแฟกต์ไทป์แสดงไว้ในรูป 2.17 ซึ่งได้แก่แบบ one to many, many to one, one to one และ many to many ซึ่งจากรูปก็ได้แสดงความหมายไว้ อย่างชัดเจนอยู่แล้ว จึงไม่กล่าวอะไรมาไกลกว่านี้

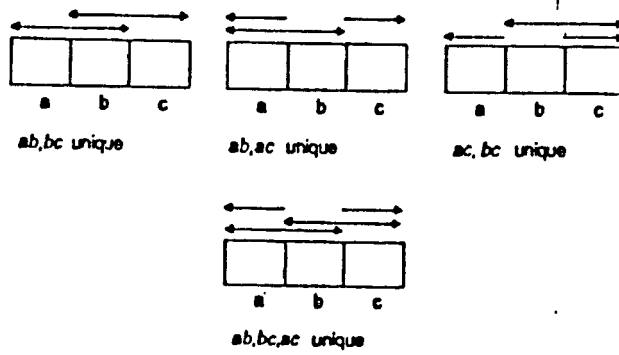


รูปที่ 2.16 แสดงตัวอย่างการใช้งานคอนสเตรนซ์แบบยูนิค



รูปที่ 2.17 แสดงแบบต่างๆของคอนสเตรนซ์แบบยูนิค สำหรับแฟกต์ไทม์แบบไบนารี

อย่างไรก็ตามสิ่งที่กล่าวมาทั้งหมด เราได้กล่าวถึงเฉพาะกับแบบที่เป็นไบนารีเท่านั้น หากแต่ในความเป็นจริงแล้วยังมีแฟกต์ไทม์ที่มี role มากกว่า 2 role และในทางดังกล่าว เราก็ใช้หลักที่ว่าสำหรับแฟกต์ไทม์ n role ใดๆ จะสามารถมี role ที่ไม่มี Uniqueness Constraint กำกับได้เพียง 1 role เท่านั้นเช่นแฟกต์ไทม์ 3 role จะสามารถมี คอนสเตรนซ์แบบยูนิค ที่เป็นไปได้อีกทั้งหมด 4 แบบดังรูปที่ 2.18

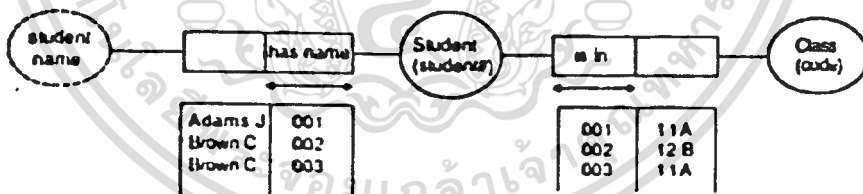


รูปที่ 2.18 แสดงคอนสเตรนซ์แบบยูนิคของเทอร์นารี

เท่าที่ผ่านมา เราได้กล่าวถึงคอนสเตรนซ์แบบยูนิค ในแบบที่ไรลทุกโรล ที่อยู่ภายใต้ คอนสเตรนซ์ หนึ่ง จะอยู่ในแพทช์ไทป์เดียวกันเท่านั้น ต่อไปเราลองมาดูข้อมูลในตารางที่ 2.4 จากตารางจะเห็นได้ว่านักศึกษาแต่ละคนจะมีชื่อเพียงชื่อเดียว และนักศึกษาแต่ละคนจะมีชั้นเรียนเพียงชั้นเรียนเดียว จากข้อมูลดังกล่าวจะสามารถเขียนแผนภาพระดับแนวคิดได้ดังรูปที่ 2.19

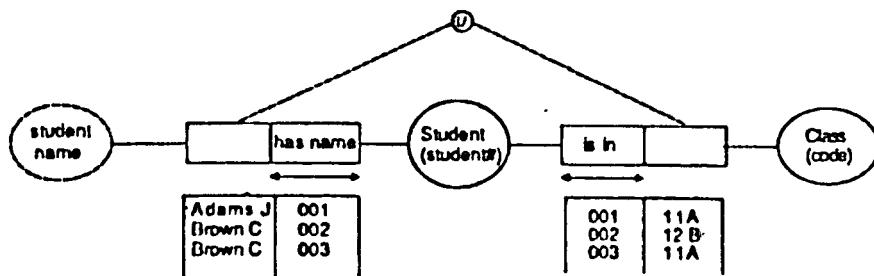
Student:	student#	name	class
	001	Adams J	11A
	002	Brown C	12B
	003	Brown C	11A

ตารางที่ 2.4 แสดงตัวอย่างข้อมูล



รูปที่ 2.19 แสดงแบบจำลองข้อมูลที่ได้จากตาราง

อย่างไรก็ตามหากถึงแก่ให้ได้จากตาราง จะเห็นได้ว่าผลรวมของชื่อและชั้นเรียนก็จะเป็นข้อมูลที่ซ้ำกัน และในภาพที่เห็น NIAM ก็อนุญาตให้สามารถสร้างคอนสเตรนซ์แบบยูนิค ระหว่างแพทช์ไทป์ 2 แพทช์ไทป์ได้ ดังรูปที่ 2.20



รูปที่ 2.20 แสดงคอนสเตรนซ์แบบยูนิคแบบ Inter

จากรูปจะเห็นได้ว่าเราจะใช้เครื่องหมาย "u" ล้อมด้วยวงกลมและมีเส้นโยงไปยังโวลที่อยู่ภายใต้คอนสเตรนซ์แบบยูนิค นั้น การสร้างคอนสเตรนซ์ ระหว่างชนิดความจริงนี้เราเรียกว่า Inter-fact-type และเรียกคอนสเตรนซ์ที่อยู่ในแฟกต์ไคป์เดียวกันนี้ว่า Intra-fact-type จากที่นำมามาก่อนหน้านี้ เราจะเห็นได้ว่าการตรวจสอบความถูกต้องโดยใช้ตัวอย่างในในแอม นั้นมีความสำคัญมาก ทั้งนี้เพราะจะทำให้ผู้ออกแบบสังเกต ข้อมูลบางอย่างเพิ่มเติมได้จากข้อมูลที่ได้มานั้น

ขั้นตอนที่ 5 ตรวจสอบอาร์ดิ (Arity Checks)

หากในขั้นตอนที่นำมามาก่อนหน้านี้ เราได้ทำการออกแบบอย่างระมัดระวังแล้ว การทำงานในขั้นตอนนี้ นับว่าเป็นสิ่งที่ไม่จำเป็นเลย อย่างไรก็ตาม จากความจริงที่ว่า นักปราชญ์ยังรู้ถึงนั้นเป็นข้อเตือนใจได้เป็นอย่างดี เพราะแม้ว่าเราจะเป็นผู้ออกแบบที่ชำนาญเพียงใดก็ตาม ก็ยังอาจเกิดการผิดพลาดกันได้

ความผิดพลาดที่เกิดขึ้นนี้มีอยู่ 2 ประการคือการออกแบบแฟกต์ไคป์ที่สั้นหรือยาวเกินไป ในกรณีของแฟกต์ไคป์ที่ยาวเกินไปนั้น ก็จะต้องทำการแยกแฟกต์ไคป์นั้นออกเป็น 2 แฟกต์ไคป์ และสำหรับแฟกต์ไคป์ที่สั้นเกินไปนั้น ไม่สามารถที่จะบรรจุข้อมูลได้ครบ (information Loss) ดังนั้นก็จะต้องทำการรวม 2 แฟกต์ไคป์เป็นแฟกต์ไคป์เดียว

สำหรับการตรวจสอบความถูกต้องนั้น จริงๆ แล้วอาจจะมีวิธีการมากมาย สุดแต่ใครจะใช้วิธีการอย่างไร หากแต่ในที่นี้เราจะกล่าวถึงเพียง 3 วิธี

1. ใช้สัญกรณ์ หรือความรู้พื้นฐานในการพิจารณาตัดสินว่ามีข้อมูลสูญหายเพราะการ

แยกแฟกต์ไทม์ออกมากหรือไม่

2. ให้อกฏของการแยกจากกัน (Splittability Rules) ซึ่งจะกล่าวถึงภายหลัง
3. ให้อกฏที่สร้างตัวอย่างที่เหมาะสมสำหรับแฟกต์ไทม์ และทำการแยกแฟกต์ไทม์นั้นออกโดยวิธี Projection (ซึ่งจะกล่าวถึงในภายหลัง) จากนั้นนำมารวมกันใหม่โดยให้อกฏ Natural Join และหากมีตัวอย่างเพิ่มขึ้นมาจากการ Join ก็แสดงว่าแฟกต์ไทม์นั้นเป็นชนิดความจริงที่ไม่สามารถทำการแยกได้

Person	Degree	Subject	Rating
Adams	BSc	CS112	7
Adams	BSc	CS110	6
Adams	BSc	PD102	7
Brown	BA	CS112	6
Brown	BA	PD102	7
Collins	BSc	CS112	7

ตารางที่ 2.5 แสดงตัวอย่างข้อมูลของผลการเรียน

สำหรับวิธีการแรกนั้น อาจจะเป็นวิธีที่ไม่ค่อยจะมีหลักการสักเท่าใดนัก แต่จริงๆ แล้วนับได้ว่าเป็นวิธีที่ใช้งานได้ เพราะเป็นวิธีที่ผสมผสานกันทั้งขั้นตอนการกำหนด Elementary Facts เพื่อให้การทำงานในวิธีนี้ชัดเจนยิ่งขึ้น เราลองมาดูตัวอย่างต่อไปนี้ เราจะเริ่มจากข้อมูลในตารางที่ 2.5 และจากวิธีการที่ผ่านมา เราสามารถสร้าง Elementary Facts ได้ดังนี้

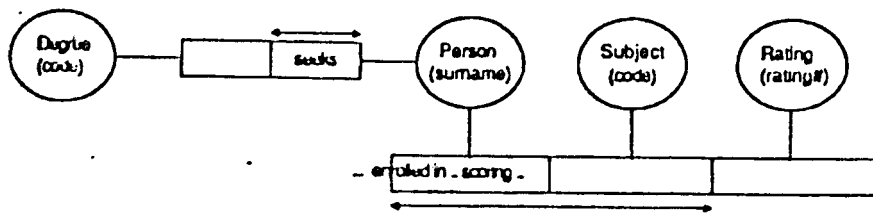
Person(surname) 'Adams' seeks Degree(code) 'BSc'.

Person(surname) 'Adams' enrolled in Subject(code) 'CS112'

scoring Rating(Nr) 7.

ในที่นี้คำว่า Seeks หมายถึงสำเร็จตามหลักสูตร และจาก Elementary Facts ดังกล่าว เราสามารถนำมาเขียนเป็นแผนภาพระดัมนามวลได้ดังรูปที่ 3-17 ที่นี้เราก็จะมาลองดูว่า หากเราต้องการจะแยกแฟกต์ไทม์แบบเทอร์นารี ในภาพออกเป็น 2 แฟกต์ไทม์จะได้หรือไม่ โดยพิจารณาว่าหากเราทำการแบ่งแล้ว เราก็จะได้ข้อมูลว่า Adams ได้คะแนนเท่ากับ 7 และ Adams ลงเรียนในวิชา CS112 ซึ่งจะเห็นได้ค่อนข้างชัดเจนว่ามีข้อมูลบางส่วนหายไป (Information Loss) นั่นคือข้อมูลที่ว่า Adams ได้คะแนนเท่ากับ 7 ซึ่งไม่ทราบว่าได้คะแนนในวิชาใด ดังนั้น

สรุปได้ว่าแพทช์ไทม์นี้ไม่สามารถจะแยกได้



รูปที่ 2.21 แสดงแบบจำลองข้อมูลที่ได้จากตารางที่ 2.5

1. การให้กฎการแยก (Splitability Rule)

จริงๆ แล้วเราได้กล่าวถึงกฎที่ว่านี้ไปครั้งหนึ่งแล้ว ในขั้นตอนที่ผ่านมา นั่นคือกฎที่ว่า จะต้องมียาวที่ไม่อยู่ภายใต้คอนสเตรนซ์แบบยูนิค ได้ไม่เกิน 1 เท่านั้น

2. การตรวจสอบโดยใช้ Projection-Join

ในตอนนั้นแล้ว เราได้กล่าวถึงเงื่อนไขของการแยกแพทช์ไทม์โดยใช้คอนสเตรนซ์แบบยูนิค อย่างไรก็ตามในบางครั้งเงื่อนไขดังกล่าวก็ไม่สามารถใช้งานได้และนอกจากนั้น ผู้ออกแบบบางคนก็อาจจะต้องการวิธีการที่แน่นอนในการตรวจสอบ ซึ่งเราจะได้อีกต่อไป

ต่อไปเราจะกล่าวถึงหลักการหรือวิธีการที่แน่นอนที่จะใช้ในการแก้ปัญหานี้ได้ ในวิธีนี้เราจะใช้การทำงาน 2 อย่างคือ Projection และ Joining โดยที่ Projection ก็คือการสร้างตารางใหม่ขึ้นมาจากรายเดิม 1 ตารางโดยเลือกคอลัมน์มาเพียงบางคอลัมน์ ซึ่งในตารางใหม่นี้ บางครั้งก็จะเกิดการซ้ำกันขึ้น ดังนั้นเราจะต้องคัดข้อมูลในบางแถวที่มีการซ้ำกันออกไป สำหรับ Join ก็คือการรวมตาราง 2 ตารางขึ้นมาเป็นตารางใหม่

ที่นี่เราลองมาดูที่ข้อมูลในตารางที่ 2.6 จากรูปจะเห็นได้ว่าเราทำการ Projection ตารางออกเป็น 2 ตาราง ดังแสดงในตารางที่ 2.7 และเมื่อเรานำ 2 ตารางนี้มาทำการ Join กันใหม่ก็จะได้ข้อมูลดังตารางที่ 2.8 ซึ่งจะเห็นได้ว่ามีแถวของข้อมูลเกิดขึ้นใหม่ (แถวที่มีเครื่องหมาย X) ซึ่งข้อมูลเหล่านี้เป็นข้อมูลที่ไม่ได้มีอยู่ในตารางที่เราทำการ Projection มา ดังนั้นจะสรุปได้ว่าเราไม่สามารถแยกแพทช์ไทม์นี้ได้

<i>Person</i>	<i>Degree</i>	<i>Subject</i>
Adams	BSc	CS112
Adams	BSc	CS110
Adams	BSc	PD102
Brown	BA	CS112
Brown	BA	PD102
Collins	BSc	CS112

ตารางที่ 2.6 แสดงตัวอย่างข้อมูล

<i>Person</i>	<i>Degree</i>	<i>Person</i>	<i>Subject</i>
Adams	BSc	Adams	CS112
Brown	BA	Adams	CS110
Collins	BSc	Adams	PD102
		Brown	CS112
		Brown	PD102
		Collins	CS112

ตารางที่ 2.7 แสดง Projection ของข้อมูลในตารางที่ 2.6

	<i>Person</i>	<i>Subject</i>	<i>Rating</i>
	Adams	CS112	7
X	Adams	CS112	6
X	Adams	CS110	7
	Adams	CS110	6
	Adams	PD102	7
X	Adams	PD102	6
	Brown	CS112	6
X	Brown	CS112	7
X	Brown	PD102	6
	Brown	PD102	7
	Collins	CS112	7

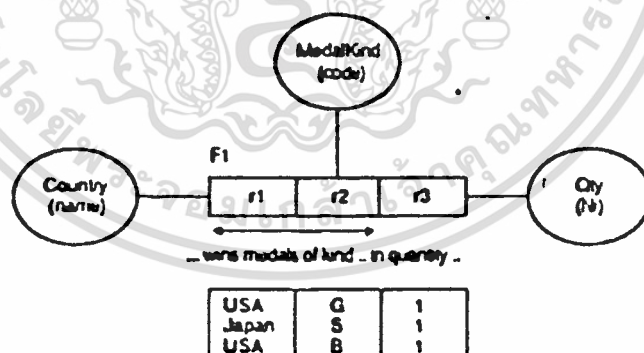
ตารางที่ 2.8 แสดงผลของการ Join ข้อมูลในตารางที่ 2.6

ขั้นตอนที่ 6 คอนสเตรนทอื่น ๆ

ในขั้นตอนที่ผ่านมา เราได้กล่าวถึงการสร้างแบบจำลองข้อมูลโดยเริ่มตั้งแต่การสร้าง "ความจริงพื้นฐาน" จนกระทั่งการตรวจสอบความถูกต้องขั้นต้น ซึ่งเมื่อถึงขั้นนี้เราจะเห็นได้ว่าเราได้กล่าวถึง คอนสเตรนท ไปเพียง 1 ชนิดเท่านั้น และจากที่กล่าวไปแล้วว่าในแอมเป็นแบบจำลองข้อมูลในระดับแนวคิดที่จะเน้นที่ความหมายเป็นสำคัญ และสิ่งที่สำคัญรองลงมาคือข้อจำกัดเพื่อความถูกต้อง ดังนั้นสิ่งที่เราจะกล่าวต่อไป ก็คือการสร้างข้อจำกัด (คอนสเตรนท) ต่างๆเพื่อให้ในแอมเป็นแบบจำลองข้อมูลที่มีความถูกต้องของข้อมูลมากที่สุด

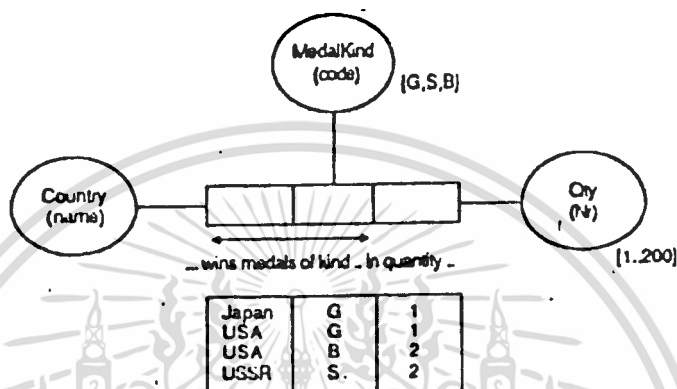
- คอนสเตรนทแบบเอนติตี้ไทป์ (entity type constraint)

จากรูปที่ 2.22 เป็นแบบจำลองข้อมูลของการได้เหรียญของประเทศต่างๆ จากรูปเอนติตี้ไทป์ Country เป็นชื่อของประเทศต่างๆ ทั้งหมดในโลกที่เป็นไปได้ ในกรณีนี้หากเราต้องการที่จะหาว่ามีสมาชิกใดบ้างที่สามารถอยู่ในเอนติตี้นี้ได้ อาจจะเป็นงานที่ยากเพราะมีการตั้งประเทศขึ้นมาใหม่ๆ อยู่เรื่อยๆ อย่างไรก็ตามสำหรับในบางเอนติตี้ไทป์นั้นง่ายมาที่จะระบุถึงสมาชิกของเอนติตี้นั้น เช่น เอนติตี้ของเหรียญที่ใช้ในกีฬาโอลิมปิกก็จะมีเหรียญ ทอง เงิน และบรอนซ์ และในบางครั้งหากเราต้องการให้ข้อมูลที่อยู่ในเอนติตี้ไทป์ MedalKind เป็นได้เพียง(G,S,B) เท่านั้น เราสามารถทำได้โดยการระบุ คอนสเตรนทแบบเอนติตี้ไทป์ บนเอนติตี้ไทป์นั้น



รูปที่ 2.22 แสดงตัวอย่างของแบบจำลองข้อมูลของผลการแข่งขันกีฬา

ซึ่งในการมีของเอนติตี้เหรียญนี้เราสามารถเขียนแบบจำลองข้อมูลขึ้นใหม่ โดยระบุคุณสมบัติแบบเอนติตี้ไทย ได้ดังในรูปที่ 2.23 ซึ่งจากรูปจะเห็นได้ว่า เราสามารถระบุชนิดของเอนติตี้ โดยเขียนที่หัวที่ด้านข้างของเอนติตี้ไทย โดยที่ในการมีสมาชิกจำกัดก็จะใช้เครื่องหมาย {} เช่น {G,S,B} หมายถึงสมาชิกของเอนติตี้ไทยนี้จะต้องเป็นสมาชิกของเซต {G,S,B} ด้วยและหากเป็นช่วงเราจะใช้เครื่องหมาย [] เช่น [1..200] ซึ่งหมายความว่าสมาชิกของเอนติตี้ไทยนี้จะต้องมีค่าอยู่ในช่วง 1 ถึง 200 เท่านั้น



รูปที่ 2.23 แสดงการใช้ คอนสเตรนแบบเอนติตี้ไทย

- คอนสเตรนแบบเมคาทอรี (Mandatory constraint)

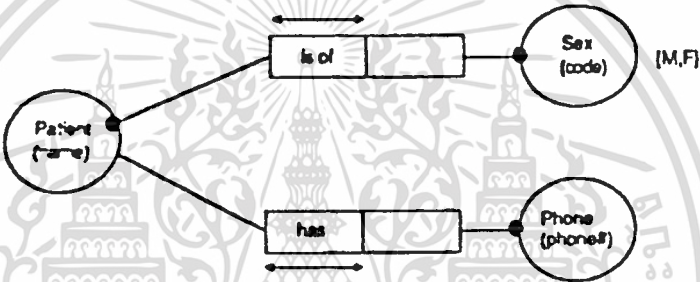
ก่อนที่จะอธิบายถึงรายละเอียดของ คอนสเตรนนี้ ในขั้นต้นเราจะอธิบายถึงสาเหตุที่เราจะต้องมี คอนสเตรนนี้ โดยเราจะพิจารณาจากตัวอย่างข้อมูลในตารางที่ 2.9 จะเห็นว่าข้อมูลอยู่ที่หนึ่งที่เราใส่เครื่องหมาย "7" ไว้เพื่อแสดงว่ายังไม่มีมีการบันทึกข้อมูลในตารางนี้ เช่น Row 5 ที่จริงแล้วอาจจะมีโทรศัพท์ก็เป็นได้ แต่ยังไม่ได้รับการบันทึก

อย่างไรก็ตามหากข้อมูลที่ยังไม่ได้รับการบันทึกนั้น เป็นข้อมูลที่สำคัญเช่นข้อมูลเหตุการณ์ที่เราจะไม่บันทึกข้อมูลนั้นลงไปที่การจะนับได้ว่าเป็นการไม่ถูกต้องนัก และอาจจะเป็นการผิดพลาดการนำไปใช้งาน เช่นข้อมูลชั้นปีของนักศึกษา หากไม่ได้รับการบันทึกก็อาจจะทำให้ความหมายอื่นๆ ที่สัมพันธ์กับค่าเหล่านี้ได้ ดังนั้นสำหรับในข้อมูลบางตัวที่ระบุให้ "ต้องบันทึก" นั้น เราสามารถทำได้โดยการระบุคอนสเตรนแบบเมคาทอรี โดยใช้เครื่องหมาย "จุด" ใส่ไว้ที่จุดเชื่อมระหว่างเส้นที่มาจากเอนติตี้ไทยและ ไลน์ตรงข้ามของ ไลน์ที่จะระบุ คอนสเตรนที่เช่นจากรูปที่ 2.23

ซึ่งมีความหมายว่าคนไข้ทุกคนจะต้องระบุเพศ จะเขียนโดยใส่จุดที่จุดเชื่อมระหว่างเอนดดี Patient และเส้นที่ลากมาจาก ไรล ซึ่งเป็นด้านตรงข้ามของ ไรล ที่ต้องการระบุ คอนสเตรนซ์

Patient	Sex	Phone
Adams C	F	2057642
Brown S	F	?
Collins T	M	8853020

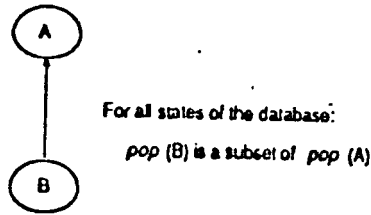
ตารางที่ 2.9 แสดงข้อมูลของคนไข้



รูปที่ 2.23 แสดงการใช้งานคอนสเตรนซ์แบบนาคาอวี่

- สับไทป์ (SubType)

สำหรับเรื่องของสับไทป์นี้เราจะใช้แนวคิดของออยเลอร์ (Euler’sDiagram) ในการแสดงความสัมพันธ์ระหว่างเอนดดี จากที่เราได้กล่าวมาแล้วว่าเอนดดีเป็นเซตที่ลงสมมติว่าเรามีเอนดดี Manager ซึ่งเป็นเซตของผู้จัดการทั้งหมด และเอนดดี Employeeซึ่งเป็นเซตของพนักงานที่ทำงานอยู่ และจากความจริงที่ว่าผู้จัดการทุกคนย่อมเป็นพนักงานด้วยเช่นกัน ดังนั้นหากจะเขียนแบบจำลองโดยให้เอนดดี Manager และ Employee ไม่เกี่ยวข้องซึ่งกันและกันนั้น ก็คงจะเป็นการผิดความหมายแน่นอน และลักษณะความสัมพันธ์ในลักษณะของสับเซตเช่นนี้เราจะเรียกว่าสับไทป์ ซึ่งจะเขียนได้ดังรูปที่ 2.24

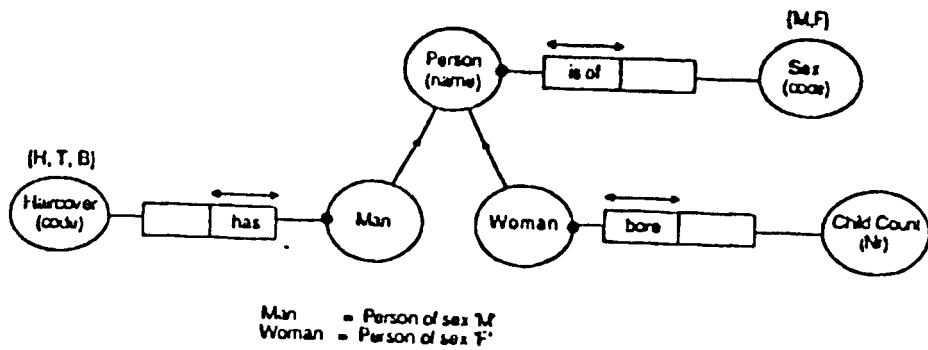


รูปที่ 2.24 แสดงความสัมพันธ์ระหว่างซูเปอร์ไค์และสับไค์

จากรูปหมายถึงว่าเอนทิตี B เป็นสับเซตของเอนทิตีไค์ A และจะเรียก A ว่าซูเปอร์ไค์ และเรียก B ว่าสับไค์ที่เน้นย่อหมายถึงว่าสมาชิกทุกตัวใน B ก็ย่อมจะมีคุณสมบัติของเอนทิตี A ด้วย เราลองมาดูตัวอย่างกันอีกตัวอย่างหนึ่งโดยอาศัยตารางที่ 2.10 จากตารางจะเห็นได้ว่า เฉพาะเพศชายเท่านั้นจึงจะมีการบันทึก Haircover และเฉพาะเพศหญิงเท่านั้นที่มีการบันทึกจำนวนบุตร (NrChildren) ในลักษณะดังกล่าวนี้หากเราใช้วิธีการเขียนในแบบที่ผ่านมาก็ย่อมได้ และไม่มีติดหากแต่ไม่สามารถแสดงความหมายได้ครบ เพราะจากที่กล่าวมาว่าไค์เป็นแบบจำลองที่มองในระดับแนวคิดดังนั้นจึงควรจะใช้วิธีการที่จะแสดงความหมายได้ดีกว่าและนั่นคือสับไค์นั่นเองซึ่งจะเขียนเป็นแผนภาพได้ดังรูปที่ 2.25

Person	Sex	Haircover	NrChildren
Jones E	F	-	2
Smith J	M	T	-
Blow J	M	B	-
Lane L	F	-	0
Blossom B	F	-	5
"-." = "not to be recorded"			

ตารางที่ 2.10 แสดงตัวอย่างข้อมูล

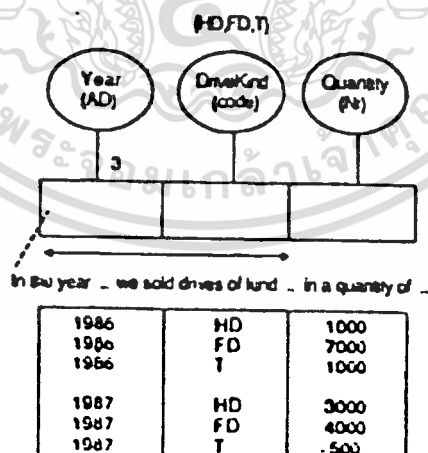


รูปที่ 2.25 แสดงการใช้สัมพันธ์ในแบบจำลองข้อมูล

- อ็อกเคอเรนท์ฟรควินซี (Occurrence frequencies)

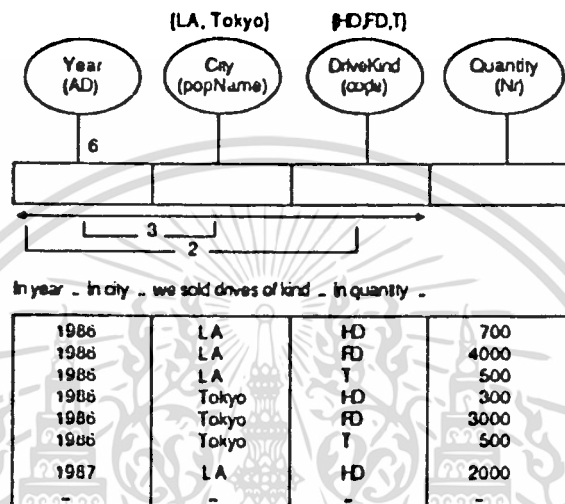
ในตอนที่เราจะกล่าวถึง คอนสเตรนท์ อีกตัวหนึ่งที่จะใช้ในการระบุจำนวนครั้งที่เอนทิตีใด ๆ จะถูกใช้ได้ใน ไรล หนึ่งๆ คอนสเตรนท์ นี้มีชื่อเรียกว่า อ็อกเคอเรนท์ ฟรควินซี

อ็อกเคอเรนท์ ฟรควินซี โดยทั่วไปจะใช้ในการระบุว่าจะมีเอนทิตีที่เกิดขึ้นในคอลัมน์ที่ระบุไว้เป็นจำนวนเท่าใด สำหรับวิธีระบุ คอนสเตรนท์ นี้เราจะใช้วิธีเขียนตัวเลขไว้เหนือหรือใต้ไรลที่ คอนสเตรนท์ นี้ระบุอยู่ ดังตัวอย่างในรูปที่ 2.26 จะเห็นว่าจะต้องมีการบันทึกการขายไดรฟ์ทั้ง 3 ชนิดในแต่ละปี ดังนั้นจึงต้องระบุอ็อกเคอเรนท์ ฟรควินซี 3 ไว้เหนือไรล year ซึ่งจะหมายถึงว่าหาถมีการบันทึกปีใดปีหนึ่งลงในไรลนี้ จะต้องบันทึกข้อมูลของไดรฟ์ทั้ง 3 ชนิดพร้อมกัน



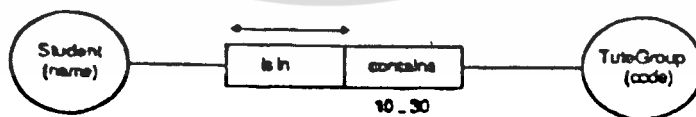
รูปที่ 2.26 แสดงตัวอย่างการใช้อ็อกเคอเรนท์ ฟรควินซี

ต่อไปเราลองมาดูอีกเคอเรนซ์ ฟังก์ชันที่มีความซับซ้อนมากขึ้น จากตัวอย่างในรูปที่ 2.27 เป็นแฟกต์ไพบ์ที่เก็บความสัมพันธ์ของไดรฟ์ชนิดต่างๆ ที่ขายได้ในแต่ละปี ในแต่ละเมือง ว่าเป็นจำนวนเท่าใด สำหรับอีกเคอเรนซ์ ฟังก์ชันนี้ 3 หมายถึงผลรวม(Combiantion) ของ role year และ city ที่เกิดขึ้นในตารางจะต้องมีได้ 3 เท่านั้น อีกเคอเรนซ์ ฟังก์ชันนี้ 2 หมายถึงผลรวมของ role year และ DriveKind เกิดขึ้นได้ 2 เท่านั้น และอีกเคอเรนซ์ ฟังก์ชันนี้ 6 หมายถึงจะมี role year เกิดขึ้นได้ครั้งละ 6 เท่านั้น



รูปที่ 2.27 แสดงตัวอย่างการใช้งานอีกเคอเรนซ์ ฟังก์ชัน

อย่างไรก็ตาม นอกจากจะระบุอีกเคอเรนซ์ ฟังก์ชันนี้ เป็นค่าคงที่แล้วยังสามารถที่จะระบุได้เป็นช่วงอีกด้วยเช่น 1..3 หมายถึงจะมี role นั้นเกิดขึ้นได้ตั้งแต่ 1 ถึง 3 เท่านั้น เช่นจากรูปที่ 2.28 จะเป็นแฟกต์ไพบ์ที่มีความหมายว่าในแต่ละ TutleGroup จะมีนักศึกษาได้ 10-30 คนเท่านั้น



รูปที่ 2.28 แสดงตัวอย่างของอีกเคอเรนซ์ ฟังก์ชัน

- ชนิดของข้อมูล (Data type)

ในขั้นตอนก่อนหน้านี้ เราได้กล่าวถึงเรื่องราวต่างๆ ระหว่างเอนิตี กับเลเบล ซึ่งโดยส่วนใหญ่ก็จะเป็นเรื่องที่เกี่ยวข้องกับเอนิตี แต่ต่อจากนี้เราจะเน้นแต่เรื่องราวเลเบลบ้าง ว่ามีรายละเอียดอะไรบ้างที่เราจะกล่าวเพิ่มเติม

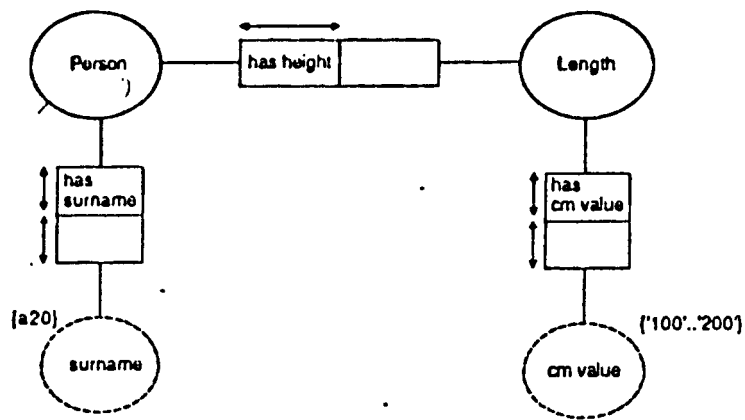
ที่ผ่านมาเราได้ใช้ คอนสเตรนส์แบบเอนิตีไพล์ เพื่อใช้บังคับความถูกต้องของข้อมูลในแต่ละเอนิตี แต่สำหรับการเลเบลแล้วเราก็มีข้อบังคับในการทำงานเช่นกัน โดยจะใช้ระบุนิตีที่เป็นไปได้ทั้งหมดของชื่อ เช่น ชื่อวิชาที่อาจจะระบุเป็น (abcd) หมายถึงชื่อของตัวอักษร 2 ตัวตามด้วยตัวเลขอีก 3 ตัว เป็นต้น

อย่างไรก็ตาม แม้ขณะนี้ยังเป็นข้อกักเกี่ยกันอยู่ว่าควรจะรวมเอา คอนสเตรนส์แบบเลเบลไพล์ ไว้ในส่วนข้อมูลระดับแนวคิด (conceptual level) หรือจะถือว่ายู่ในส่วนข้อมูลระดับภายนอก (external level) เช่น ในงานหนึ่งๆ เราอาจต้องการรูปแบบของการติดต่อกับผู้ใช้ (User Interface) ที่ต่างกันของวิชาต่างๆ ซึ่งอาจจะอ้างถึงโดยใช้ ตัวอักษร ตัวเลข อาจจะต้องกำกับด้วยภาษาอังกฤษ หรือภาษาไทย เป็นต้น

และเช่นเดียวกัน เราสามารถที่จะระบุเลเบลได้โดยอาศัยการให้ตัวอย่าง เช่น จากข้อมูลในตารางที่ 3-12 สำหรับเอนิตี Person เราจะใช้ Surname เป็นเลเบลซึ่งอาจจะระบุโดยเขียนว่า (c20) ซึ่งหมายถึงตัวอักษรจำนวน 20 ตัว และความสูงซึ่งอาจจะระบุโดย ('100'..'200') หมายถึงจะมีค่าได้ตั้งแต่ 100-200 หรือระบุโดยใช้ [100..200] เป็นต้นดังแสดงในรูปที่ 2.29

Person	City	Height (cm)	Chest (cm)	Mass (kg)	IQ
Adams	Brisbane	175	100	77	100
Brisbane	Sydney	182	91	80	120
Collins	Sydney	173	80	67	100
Darwin	Darwin	175	95	90	95

ตารางที่ 2.11 แสดงตัวอย่างข้อมูล



รูปที่ 2.29 แสดงตัวอย่างของแบบจำลอง

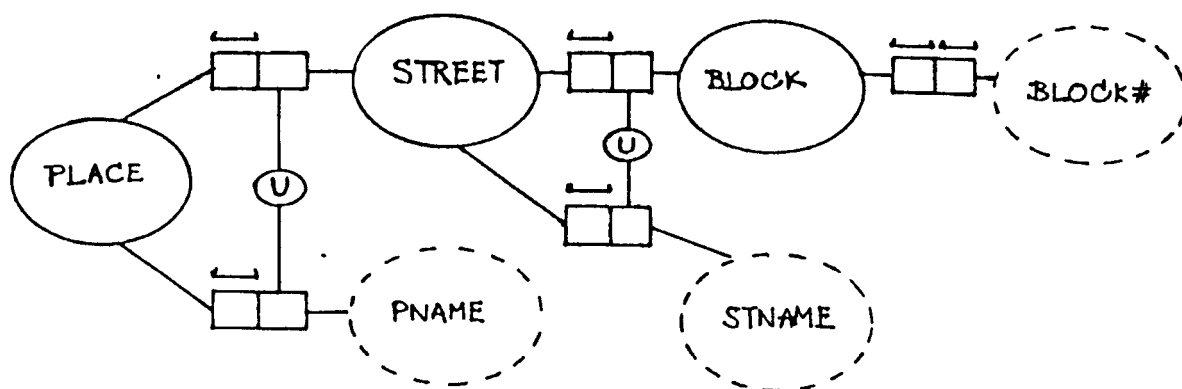
ขั้นตอนที่ 7 การอ้างอิงเลเบลไทป์ของเอนติตี้ไทป์ (Entity Identification schemes)

ที่ผ่านมาทั้งหมดเราได้สร้างแผนภาพ โดยสมมติว่าแต่ละเอนติตี้ไทป์มีความสัมพันธ์แบบเรเฟอเรนซ์ไทป์กับเลเบลไทป์เพียงเลเบลไทป์เดียว แต่ในบางครั้งก็มีเอนติตี้ไทป์ที่มีความสัมพันธ์กับหลายๆ เลเบลไทป์ได้เช่นกัน อย่างไรก็ตามจะต้องมีเรเฟอเรนซ์ไทป์อย่างน้อยหนึ่งเรเฟอเรนซ์ไทป์ที่สามารถใช้ในการระบุ (Identification) ถึงแต่ละสมาชิกในเอนติตี้ไทป์ได้ นั่นก็หมายความว่าต้องมีเรเฟอเรนซ์ไทป์อย่างน้อยหนึ่งเรเฟอเรนซ์ไทป์ที่มีความสัมพันธ์ในแบบ 1:1 กับเอนติตี้ไทป์นั้น เช่น เอนติตี้ไทป์ศึกษาอาจจะมีเลเบลหรือชื่อเรียก 2 แบบได้แก่ รหัสนักเรียนและชื่อศึกษา อย่างไรก็ตาม อย่างน้อยก็จะต้องมีเลเบลรหัสศึกษาที่แต่ละรหัสจะต้องไม่ซ้ำและสามารถใช้ในการระบุถึงแต่ละสมาชิกในเอนติตี้ศึกษาได้ และเราเรียกรหัสศึกษาว่าเป็นไอดีไฟเอนเดอร์ (Identifier)

ดังนั้นหากจะกล่าวโดยสรุปงานในขั้นตอนนี้ก็คือ การตรวจสอบว่าในแต่ละเอนติตี้จะต้องมี Identifier อย่างน้อยหนึ่ง เพื่อใช้ในการระบุถึงสมาชิกแต่ละตัวของเอนติตี้ไทป์ได้

- การอ้างถึงแบบไฮราซึ (Hierarchical reference)

อันที่จริงการอ้างถึงที่สัมพันธ์กับชั้นอนุภูมิภาคหนึ่ง ซึ่งต้องยกหลักการอ้างถึงหลายระดับขึ้น เรียกสั้นๆ เช่นนี้ว่า การอ้างถึงแบบไฮราซึ แสดงดังรูปที่ 2.30



รูปที่ 2.38 แสดงการอ้างถึงแบบไฮราคี

จากรูปจะเห็นว่าทั้งหมดนั้นเป็นการอ้างถึงของเอนิตีไทย PLACE เท่านั้น หากแต่มีการอ้างถึงที่ละลำดับชั้น เริ่มตั้งแต่ชื่อถนนซึ่งขึ้นกับเอนิตีไทย BLOCK เนื่องจากเป็นไปได้ว่า ในต่างบล็อกจากมีชื่อถนนที่ซ้ำกันได้ เมื่ออยู่ในระดับชั้นของเอนิตีไทย BLOCK ก็มีการอ้างถึงชื่อบล็อกจากขึ้นกับเอนิตีไทย DISTRICT อีก ในเงื่อนไขเดียวกัน คืออาจมีบล็อกจากเดียวกันในต่างอำเภอได้ก็อีก เช่นนี้เรียกไปจนมาถึงสตักชื่อของ STATE ซึ่งไม่เกิดการซ้ำซ้อนขึ้นอีก

ในลักษณะนี้เมื่อต้องการอ้างถึงที่อยู่ของใคร ๆ หนึ่ง จะต้องอ้างถึงเลเบลไทยต่าง ๆ มากมาย จึงนับเป็นการอ้างถึงในภาพพิเศษกรณีหนึ่ง กรณีของไฮราคีนี้อาจปรากฏสับไต้ขึ้น ณ ที่ใดที่หนึ่งในลำดับชั้นได้ เช่น กรณีที่อ้างท่าเรือที่เป็นนับเขตของอำเภออีกที เรียกกรณีเช่นนี้ว่า "การอ้างถึงแบบไฮราคีที่มีสับไต้ (Heirarchical with subtype)"

ขั้นตอนที่ 8 คณสเตรนต่อไป (Further constraint)

เราได้นำมาพิจารณาถึงขั้นตอนที่ 8 ซึ่งด้วยข้อบังคับต่างๆ ซึ่งจะว่าไปแล้วกฎข้อบังคับหรือคณสเตรนเหล่านี้มีเพื่อรักษาสถานะของฐานข้อมูลให้มีความถูกต้องอยู่เสมอ และแม้ว่าเราจะได้กล่าวถึงคณสเตรนที่มีความสำคัญไปหลายแบบแล้วก็ตาม แต่ก็ยังมีคณสเตรนอีกหลายแบบที่ทั้งต้องกล่าวถึงต่อไป

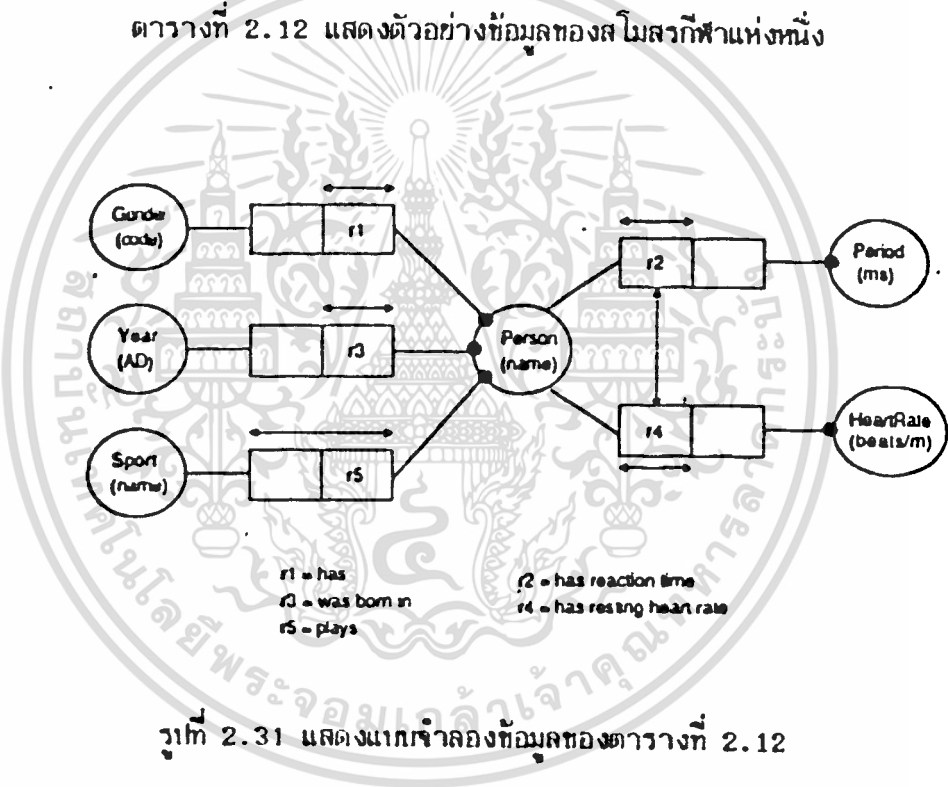
- คณสเตรนแบบมีความก้ำกึ่ง . แบบเอกซ์คลูชัน และ แบบสับเซต

(Equality , Exclusion and Sulset constraint)

เราจะเริ่มต้นด้วยวิธีการดังเช่นที่ผ่านมาคือ เริ่มด้วยการพิจารณาข้อมูลที่เราสถมมดัดขึ้น เราลองมาดูตารางที่ 2.12 ซึ่งเป็นตารางข้อมูลของสมาชิกในสโมสรแห่งหนึ่ง และจากข้อมูลดังกล่าวเราสามารถที่จะสร้างแบบจำลองข้อมูลได้ดังรูปที่ 2.31

Member	Sex	BirthYr	Sport	Reaction Time (ms)	Resting Heart Rate (beats/m)
Anderson PE	M	1940	golf	250	80
Bloggs F	M	1940	golf	?	?
Fitt LM	F	1960	acrobatics	250	80
Hume PE	F	1946	tennis	305	93
Jones T	M	1965	tennis	?	?

ตารางที่ 2.12 แสดงตัวอย่างข้อมูลของสโมสรกีฬาแห่งหนึ่ง



จากตารางจะเห็นได้ว่าข้อมูล Reaction Time และ Heart Rate นั้นเป็นข้อมูลที่อาจจะมีหรือไม่ก็ได้ (Optional) โดยเครื่องหมาย ? จะมีความหมายว่าข้อมูลนั้นไม่ได้รับการบันทึก นอกจากนี้เราจะสังเกตเพิ่มได้อีกว่าข้อมูลทั้งสองนี้ จะต้องได้รับการบันทึกไว้ด้วยกันเสมอ จะบันทึกเฉพาะข้อมูลในหัวใดหัวหนึ่งไม่ได้ ในภาพของข้อมูลลักษณะเช่นนี้ ในแอมสามารถรองรับข้อจำกัดนี้ได้ โดยระบุบนแผนผังแบบอ็อบเจกต์ทั้งสอง โดยใช้เครื่องหมายลูกศร 2 หัว

ดังในรูปที่ 2.31 ซึ่งจากรูปลูกี่จะมีความหมายว่า เช็ดของคนที่ได้รับการบันทึกใน r2 จะต้องเท่ากับเช็ดของคนที่ได้รับการบันทึกใน r4 หรือสามารถเขียนได้ว่า $pop(r2) = pop(r4)$

2.8.3 การสร้างเค้าร่างฐานข้อมูลแบบสัมพันธ์ โดยวิธีแปลงแบบจำลองระดับแนวคิดของโนแอม (Create relational database schema from NIAM conceptual schema)

ในขั้นตอนก่อนหน้านี้ เราได้ศึกษาถึงวิธีการออกแบบแบบจำลองข้อมูลระดับแนวคิดมาแล้ว หากแต่ในปัจจุบันนี้ยังไม่มีระบบฐานข้อมูลใดที่สามารถใช้งานกับแบบจำลองเช่นนี้ได้ หากแต่เป็นระบบฐานข้อมูลที่เป็นแบบสัมพันธ์

ระบบฐานข้อมูลแบบสัมพันธ์ ได้รับการพัฒนาจนมีประสิทธิภาพสูง และมีภาษา SQL (Structured Query Language) ซึ่งเป็นภาษาหลักของแบบจำลองข้อมูล (Data Model) ที่เป็นวีเลทแนล และได้นำไปใช้งานอย่างกว้างขวางในการต่างๆ ซึ่งเมื่อไม่นานมานี้ภาษา SQL ก็ได้รับการรับรองให้เป็นภาษามาตรฐาน ทำให้มีการใช้งานภาษานี้ทั้งบนเครื่องเมนเฟรมมินิคอมพิวเตอร์ ไมโครคอมพิวเตอร์ ในระบบปฏิบัติการ (Operating System) ต่างๆ และนอกจากนี้ได้กล่าวมาแล้วนั้น SQL ยังสามารถที่จะเชื่อมต่อกับภาษาต่างๆเช่นโคบอล หรือ ซี ได้อีกด้วย ทำให้การใช้งานคล่องตัวยิ่งขึ้น

ดังนั้นต่อไปเราก็นำเสนออัลกอริทึมที่ใช้ในการแปลงจาก แบบจำลองข้อมูลระดับแนวคิดของโนแอมให้เป็นเค้าร่างฐานข้อมูลแบบสัมพันธ์

นอร์มัลฟอร์มที่เหมาะสม (Optimal Normal Form Algorithm : ONF)

จากที่เราได้กล่าวมาแล้ว เราได้นำตัวอย่างที่เราจะแปลงแบบจำลองข้อมูลในแบบแนวคิดไปเป็นแบบสัมพันธ์ โดยกำหนดรูปแบบของตาราง และการเก็บกฎขึ้นเพื่อจัดการกับคอนสเตรนต์ต่างๆ ซึ่งต่อไปเราจะแนะนำวิธีการที่เป็นระบบในการแปลงแบบจำลองข้อมูลในแบบแนวคิดไปเป็นแบบสัมพันธ์

สำหรับแต่ละคอนเซ็ปชวลสกีมนั้น เมื่อทำการแปลงเป็น Relational Schema นั้นอาจจะแปลงได้หลายรูปแบบ แต่สำหรับอัลกอริทึม ONF นี้มีจุดประสงค์เพื่อสร้างรูปแบบที่ง่าย ประหยัด และมีประสิทธิภาพ โดยหลักการต่อไปนี้ ประการแรกคือเพื่อให้ได้โครงสร้างข้อมูลที่ย่าง ไม่มีทวิภาคซ้ำ ประการที่สองคือป้องกันการมีสหพาดกันเนื่องมาจากการเปลี่ยนแปลงแก้ไข (Update Anomalies) ข้อมูลไม่มีการซ้ำซ้อน ประการที่ 3 เพื่อให้ได้ประสิทธิภาพ ดัง

นั้นจะต้องได้จำนวนตารางที่น้อยด้วย เพราะการลดจำนวนตารางก็เท่ากับการลดเวลาที่ใช้ในการเข้าถึงข้อมูล และยังทำให้การทำควิรีสามารถทำได้ง่ายขึ้นอีกด้วย

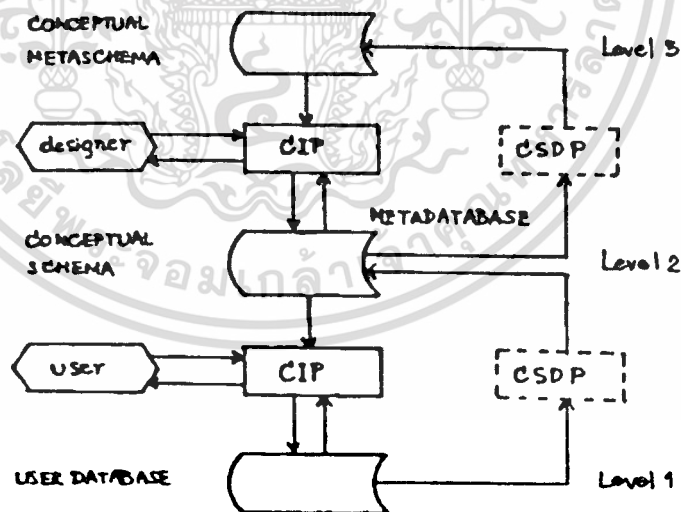
การแปลงเป็นตารางนี้ เมื่อเทียบกับวิธีนอร์มอลไลซ์เซชัน ซึ่งเป็นวิธีการสร้างตารางเช่นกัน โดยจะเริ่มจากนอร์มอลฟอร์มที่ 1 และสิ้นสุดที่นอร์มอลฟอร์มที่ 5 ในขณะที่ ONF สามารถที่จะสร้างนอร์มอลฟอร์มที่ 5 ได้เลย และสำหรับ ONF Algorithm ก็มีดังต่อไปนี้

1. สำหรับแต่ละแฟกต์ไพบ์ที่ไม่ได้มีคีย์มาตรฐาน (simple key) ให้สร้างเป็นตารางแยกออกมา โดยเลือกคีย์ที่เป็นที่สุดของแฟกต์ไพบ์นั้นเป็นคีย์หลัก
2. รวมแฟกต์ไพบ์ที่มีคีย์มาตรฐานที่เชื่อมกับอีกแฟกต์ไพบ์เดียวกันเป็นตารางเดียวกัน และให้อีกแฟกต์ไพบ์นั้นเป็นคีย์หลัก
3. สำหรับแฟกต์ไพบ์ที่เหลือให้สร้างเป็นตารางแยกต่างหาก



ในงานที่ต้องการใช้ระบบฐานข้อมูลในการเก็บข้อมูล ไม่ว่าจะเป็นฐานข้อมูลที่มีนิยามข้อมูลในระดับแนวคิดแบบใด เช่น แบบข่ายงาน (Network model) แบบลำดับขั้น (Hierarchical model) หรือ แบบสัมพันธ์ (Relational model) ต้องผ่านกระบวนการที่เรียกว่า " การออกแบบฐานข้อมูล (Database design) " ก่อนทั้งสิ้น ในงานที่อาศัยนิยามข้อมูลระดับแนวคิดแบบสัมพันธ์ สามารถออกแบบฐานข้อมูลได้โดยวิธีนอร์มอลไลซ์เช่น วิธีวาดแผนภูมิอีอาร์ หรือ แบบจำลองข้อมูลของไนแอม (NIAM)

จากเนื้อหาในบทนำ ที่กล่าวถึงการนำเอาแบบจำลองไนแอมที่รับจากโมดูลตัวवाद มาทำการวิเคราะห์ที่โมดูลวิเคราะห์นั้น โครงสร้างข้อมูลหนึ่งที่ได้ว่ามีบทบาทสูงสุดในกระบวนการดังกล่าวเรียกว่า เมตาดาต้าเบส ซึ่งหมายถึง ฐานข้อมูล ๆ หนึ่งที่เก็บข้อมูลแทนความสัมพันธ์ของนิยามข้อมูลระดับแนวคิดของไนแอม ในเค้าร่างฐานข้อมูลที่ได้จากเมตาสเก็มม่า รูปที่ 3.1 แสดงแผนภูมิความสัมพันธ์ระหว่าง เมตาสเก็มม่า เมตาดาต้าเบส ผู้ใช้และแบบจำลองนิยามข้อมูลระดับแนวคิดของไนแอม หรือ คอนเซ็ปชวลสเก็มม่า



รูปที่ 3.1 แผนภูมิแสดงระดับในการแทนข้อมูลในระดับแนวคิด

จากรูปที่ 3.1 มีเจ้าหน้าที่ที่เกี่ยวข้องในระบบ 2 คน คือ นักออกแบบฐานข้อมูล (designer) และ ผู้ใช้งานฐานข้อมูล (user) ในระดับที่ 2 , 3 ผู้ออกแบบจะทำการออกแบบคอนเซ็ปชวลสกินมาจากรหัสตัวประมวลผลที่ข้อมูลระดับแนวคิด (CIP : Conceptual Information Processor) ซึ่งทำหน้าที่จัดการและดูแลความเปลี่ยนแปลงของข้อมูลในฐานข้อมูล ในลักษณะตอบโต้กับผู้ใช้ ซีไอพีจะเป็นตัวกลางในการจัดการข้อมูลที่ผู้ใช้ (ในระดับนี้คือนักออกแบบ) ต้องการเปลี่ยนแปลง โดยตรวจสอบจากกฎความถูกต้องที่กำหนดไว้ในคอนเซ็ปชวลเมตาสกินมา (Conceptual metaschema) ซึ่งหมายถึง นิยามข้อมูลระดับแนวคิดที่เก็บข้อมูล กฎ และข้อกำหนดของข้อมูลของคอนเซ็ปชวลสกินมาเอง ข้อมูลในฐานข้อมูลที่ได้ในระดับนี้คือข้อมูลของคอนเซ็ปชวลสกินมาที่นักออกแบบทำการสร้างขึ้น โดยได้รับการตรวจสอบแล้วว่าถูกต้องตามข้อกำหนดของลักษณะข้อมูล ฐานข้อมูลดังกล่าวนี้เอง ที่เราเรียกว่า ฐานข้อมูล หรือ เมตาดาต้าเบส

เมื่อได้เมตาดาต้าเบสเรียบร้อยแล้ว ขั้นตอนจะผ่านมาถึงในระดับ 1 , 2 ซึ่งในระดับนี้ผู้ใช้คือผู้ใช้งานฐานข้อมูล เมตาดาต้าเบสที่ได้จากขั้นแรกนั้น แท้จริงแล้วก็คือคอนเซ็ปชวลสกินมาของงานแอปพลิเคชันที่ต้องการฐานข้อมูลใช้งานนั่นเอง (เพียงแต่อยู่ในรูปฐานข้อมูลมิใช่อยู่ในรูปแบบจำลองในแอม) เช่นเดียวกับในระดับ 2,3 มีตัวประมวลผลซีไอพี ทำหน้าที่ดูแลฐานข้อมูลโดยอ้างอิงตามข้อกำหนดลักษณะข้อมูลที่กำหนดในคอนเซ็ปชวลสกินมา ข้อมูลที่ผู้ใช้ทำงานอยู่คือฐานข้อมูลของแอปพลิเคชัน ซึ่งใช้เก็บข้อมูลในฐานข้อมูลของระบบงานต่างๆ ตามจุดประสงค์

หากสังเกตให้ดี จะพบว่าระหว่างระดับที่ 1 กับ 2 และ 2 กับ 3 จะมีกระบวนการออกแบบคอนเซ็ปชวลสกินมา (Conceptual Schema Design Procedure) ซึ่งได้กล่าวไว้ในหลักการเบื้องต้นที่ 2

3.1 เอนทิตีในโมเดลคอนเซ็ปชวลเมตาสกินมา (entity in NIAM - Conceptual metaschema)

เนื่องจากในโครงงานนี้ อ้างอิงแบบจำลองข้อมูลในแอม เป็นนิยามข้อมูลระดับแนวคิดในการออกแบบ เมตาสกินมาที่ใช้ในโครงงานจึงเป็นแบบจำลองระดับแนวคิดของในแอมด้วย เอนทิตีไพบ์ของเมตาสกินมานี้มีดังนี้

1. OBJECT TYPE เซตของอ็อบเจกต์ไพบ์ของในแอม ใช้ OBJNAME ในการอ้างอิง
2. LABEL TYPE เซตของเลเบลไพบ์ของในแอมซึ่งเป็นสับเซตของอ็อบเจกต์ไพบ์ ใช้

OBJNAME ในการอ้างอิง

3. ENTITY TYPE เขตของเอนทิตีไทป์ของไทม์แมชชีนซึ่งเป็นสับเซตของอ็อบเจกต์ไทป์ ใช้ OBJNAME ในการอ้างอิง

4. KINDOFOBJ เขตของประเภทของอ็อบเจกต์ของไทม์แมชชีน ใช้ชี้เฉพาะว่าอ็อบเจกต์ที่พิจารณาเป็นเลเบลหรือเอนทิตีไทป์ ใช้ KIND ในการอ้างอิง

5. LABELORDER เขตของลำดับของเลเบลไทป์ ในกรณีที่เอนทิตีไทป์หนึ่งมีความสัมพันธ์กับเลเบลไทป์มากกว่าหนึ่งตัว ใช้ ORDER ในการอ้างอิง

6. ROLE เขตของโรลของไทม์แมชชีน ใช้ ROLENAME ในการอ้างอิง

7. ROLE TYPE เขตของชนิดของโรลที่สนใจของไทม์แมชชีน! ใช้ชี้เฉพาะว่าเป็นโรลแบบใด ได้แก่ โรลแบบปกติ และ โรลที่สัมพันธ์กับเนตติ้ง ใช้ TYPE ในการอ้างอิง

8. ROLE ORDER เขตของลำดับของโรลของไทม์แมชชีน ใช้เพื่อบอกลำดับของโรลในแต่ละแฟกต์ไทป์ หรือ เรฟเฟอเรนซ์ไทป์ ใช้ ORDER ในการอ้างอิง

9. NORMAL ROLE เขตของโรลแบบปกติของไทม์แมชชีน ซึ่งหมายถึงโรลที่สัมพันธ์กับอ็อบเจกต์ของไทม์แมชชีน เป็นสับเซตของ ROLE ใช้ ROLENAME ในการอ้างอิง

10. ROLE GROUP เขตของกลุ่มโรลที่ใช้ในการแบ่งกลุ่มของโรลที่มีความสัมพันธ์กับคอนสแตนต์ของไทม์แมชชีน ใช้ RGROUP ในการอ้างอิง

11. NESTING ROLE เขตของโรลแบบเนตติ้งของไทม์แมชชีน ซึ่งหมายถึงโรลที่สัมพันธ์กับเนตติ้งของไทม์แมชชีน เป็นสับเซตของ ROLE ใช้ ROLENAME ในการอ้างอิง

12. RELATIONSHIP TYPE เขตของรีเลชันที่กำหนดในไทม์แมชชีน มีเลขชั้นอยู่ 2 ประเภทหลักคือ แฟกต์ไทป์ และ เรฟเฟอเรนซ์ไทป์ ใช้ RELNAME ในการอ้างอิง

13. FACT TYPE เขตของแฟกต์ไทป์ของไทม์แมชชีนซึ่งเป็นสับเซตของ RELATIONSHIP TYPE ประกอบด้วยแฟกต์ไทป์ 2 ประเภทคือ แฟกต์ไทป์แบบปกติหรือ นอร์มอลแฟกต์ไทป์ (Normal fact type) และแฟกต์ไทป์แบบหาได้ หรือ ดีไรฟด์แฟกต์ไทป์ (Derived fact type) ใช้ RELNAME ในการอ้างอิง

14. NORMAL FACT TYPE เขตของแฟกต์ไทป์แบบปกติของไทม์แมชชีน ซึ่งเป็นสับเซตของแฟกต์ไทป์ FACT TYPE หมายถึงแฟกต์ไทป์ซึ่งใช้แทนข้อมูลของความสัมพันธ์ที่เก็บในฐานข้อมูลจริงในรูปของตาราง (TABLE) ใช้ RELNAME ในการอ้างอิง

15. DERIVED FACT TYPE เขตของแฟกต์ไทป์แบบหาได้ หรือ ดีไรฟด์แฟกต์ไทป์ เป็นสับเซตของ FACT TYPE ชนิดที่ข้อมูลของความสัมพันธ์ในฐานข้อมูล ถูกเก็บให้อยู่ในรูปที่ได้มาจากการคำนวณค่าข้อมูลของความสัมพันธ์อื่น ๆ เช่น ดีไรฟด์แฟกต์ไทป์ AREA (พื้นที่) ได้มาจาก

การนำเอาפקก์ไทป์ WIDTH (กว้าง) และ LENGTH (ยาว) มาคูณกัน ก็ได้ไรฟว์แพกก์ไทป์ AREA นี้ในรูปของวิว (VIEW) ในฐานข้อมูล ใช้ RELJNAME ในการอ้างถึง

16. DERIVED FORMULA เขตของสูตรที่ใช้ในการคำนวณค่าข้อมูลในความสัมพันธ์ที่เป็นแพกก์ไทป์แบบหาได้ เก็บในลักษณะสตริง สามารถเก็บได้ทั้งสูตรการคำนวณและตัวโปรแกรมภาษา SQL ที่ใช้ในการสร้างฐานข้อมูล ใช้ FORM ในการอ้างถึง

17. KINDOFJREL เขตของประเภทของรีเลชัน ได้แก่ เรฟเฟอร์เรนซ์ไทป์ (REF) แพกก์ไทป์แบบหาได้ (FJDERIVED) แพกก์ไทป์แบบปกติ (FJNORMAL) ใช้ KIND ในการอ้างถึง

18. CONSTRAINT เขตหรือข้อจำกัดหรือคอนสเตรนจ์ของไบนแอม โดยใช้ CONST# ในการอ้างถึง

19. KINDOFJCONST เขตของประเภทคอนสเตรนจ์ของไบนแอม ได้แก่ คอนสเตรนจ์แบบยูนิคเนส (UNIQUE) แบบแมนดาทอรี (MAND) แบบเอ็กคลูซีฟ (EXCLUSIV) แบบสับเซต (SUBSET) แบบอีควอลิตี้ (EQUAL) แบบเมมเบอร์ชิพ (MEMBER) แบบเรนจ์ (RANGE) แบบอ็อกเคอเรนซ์ ฟรีกวินซ์ (FREQU) ใช้ KIND ในการอ้างถึง

20. SUBSET CONST เขตของคอนสเตรนจ์แบบสับเซตของไบนแอม เป็นสับเซตของ CONSTRAINT เนื่องจากเป็นประเภทหนึ่งของคอนสเตรนจ์ จึงใช้การอ้างถึงเช่นเดียวกันคือ CONST#

21. OBJECT TYPE CONST เขตของคอนสเตรนจ์แบบอ็อบเจ็คไทป์ของไบนแอม เป็นสับเซตของ CONSTRAINT เนื่องจากเป็นประเภทหนึ่งของคอนสเตรนจ์ที่เกี่ยวข้องอยู่กับอ็อบเจ็คในไบนแอม จึงใช้การอ้างถึงเช่นเดียวกันคือ CONST#

22. OCCURRENCE FREQU เขตของคอนสเตรนจ์แบบอ็อกเคอเรนซ์ ฟรีกวินซ์ของไบนแอม เป็นสับเซตของ CONSTRAINT เนื่องจากเป็นประเภทหนึ่งของคอนสเตรนจ์ที่เกี่ยวข้องอยู่กับอ็อบเจ็คในไบนแอม จึงใช้การอ้างถึงเช่นเดียวกันคือ CONST#

23. FREQUENCY เขตของค่าความถี่ที่กำหนดในคอนสเตรนจ์แบบอ็อกเคอเรนซ์ ฟรีกวินซ์ มีทั้งค่าความถี่เริ่มต้น และค่าความถี่สุดท้ายของข้อมูล ใช้ TIME ในการอ้างถึง

24. MEMBERSHIP CONST เขตของคอนสเตรนจ์แบบเมมเบอร์ชิพของไบนแอม ซึ่งเป็นสับเซตของคอนสเตรนจ์แบบอ็อบเจ็คไทป์ เนื่องจากเป็นประเภทหนึ่งของคอนสเตรนจ์ที่เกี่ยวข้องอยู่กับอ็อบเจ็คในไบนแอม จึงใช้การอ้างถึงเช่นเดียวกับคอนสเตรนจ์แบบอ็อบเจ็คไทป์คือ CONST#

25. MEMBER เขตของสมาชิกที่กำหนดอยู่ในคอนสเตรนจ์แบบเมมเบอร์ชิพใช้ MEMBER# ในการอ้างถึง

26. RANGE CONST เขตของคอนสเตรนจ์แบบเรนจ์ของไบนแอม ซึ่งเป็นสับเซตของ

คอนสเตรนส์แบบอ็อบเจ็คไทป์ เนื่องจากเป็นประเภทหนึ่งของคอนสเตรนส์ที่เกี่ยวข้องอยู่กับอ็อบเจ็คในโมเดล จึงใช้การอ้างถึงเช่นเดียวกับคอนสเตรนส์แบบอ็อบเจ็คไทป์และเช่นเดียวกับคอนสเตรนส์แบบเมมเบอร์ชิพ คือ CONST#

27. LIMIT เซตของค่าจำกัดที่กำหนดอยู่ในคอนสเตรนส์แบบเรนจ์ ค่าจำกัดนี้ใช้ได้ทั้งค่าจำกัดบน และค่าจำกัดล่าง ใช้ LIM ในการอ้างถึง

3.2 ความสัมพันธ์ระหว่างเอนทิตีไทป์ในโมเดลคอนเซ็ปชวลเมต้าสคีมม่า

(relationship between objects in NIAM – Conceptual metaclasschema)

3.2.1 ความสัมพันธ์แบบสับเซต

มีอยู่ 4 กลุ่ม ได้แก่

1. OBJECT TYPE ในโมเดลแบ่งเป็น 2 ประเภทกัน ได้แก่ LABEL TYPE และ ENTITY TYPE เอนทิตีทั้งสองจึงเป็นสับไทป์ของ OBJECT TYPE

2. ROLE ในโมเดลแบ่งเป็น 2 ประเภทกัน ได้แก่ NORMAL ROLE และ NESTING ROLE เอนทิตีทั้งสองจึงเป็นสับไทป์ของ ROLE

3. RELATIONSHIP TYPE ในโมเดลแบ่งเป็น 2 ประเภท ได้แก่ FACT TYPE และ REFERENCE TYPE ในส่วนของเอนทิตี FACT TYPE จะปรากฏในรูปแบบเมต้าสคีมม่า แต่สำหรับ REFERENCE TYPE จะอยู่ในส่วนแมปปิงอินฟอร์เมชัน (Mapping information) ซึ่งจะกล่าวต่อไป ในเอนทิตี FACT TYPE จะมีแบ่งประเภทเป็นสับไทป์ย่อยลงไปอีกคือ NORMAL FACT TYPE และ DERIVED FACT TYPE

4. CONSTRAINT ในโมเดลแบ่งเป็น 8 ประเภท หากแต่มีเพียง 4 ประเภทที่ปรากฏในรูปแบบโมเดลเมต้าสคีมม่า เนื่องจากคอนสเตรนส์ตัวอื่น ๆ ไม่มีข้อมูลบ่งพิเศษ จะมีความสัมพันธ์ก็แต่เพียงกับ ROLE GROUP เท่านั้น ตัวอย่างคอนสเตรนส์เหล่านี้ เช่น แบบยูนิค แบบเมมเบอร์ชิพ แบบเอ็กคลูซีฟ เป็นต้น จากรูปจะมีสับไทป์ 3 สับไทป์ อันได้แก่ SUBSET CONST , OBJECT TYPE CONST และ OCCURENCE FREQ โดยที่เอนทิตี OBJECT TYPE CONST ยังแบ่งประเภทย่อยออกเป็นอีก 2 สับไทป์ ซึ่งได้แก่ MEMBERSHIP CONST และ RANGE CONST

3.2.2 ความสัมพันธ์แบบพหุไทป์

แบ่งการอธิบายตามกลุ่มสับเซตทั้ง 4 กลุ่ม ดังนี้

1. กลุ่ม OBJECT TYPE

- มีความสัมพันธ์กับ NORMAL ROLE เนื่องจากโรลแบบปกติ (ไม่ใช่โรลแบบเนสตั้ง) ในไวยากรณ์ต้องอยู่คู่กับอีอบเจ็คไทป์ตัวใดตัวหนึ่งเสมอ จะอยู่ลอย ๆ ไม่ได้
- มีความสัมพันธ์กับ KINDOFOBJOB เพื่อชี้บ่งว่าอีอบเจ็คไทป์ในไวยากรณ์ตัวใด ๆ เป็นอีอบเจ็คไทป์ประเภทเอนติตี้ไทป์ หรือ เลเบลไทป์ เรียกเอนติตี้ไทป์อย่าง KINDOFOBJOB นี้ว่า "ตัวบ่งชนิด" หรือ category
- สับไทป์ ENTITY TYPE มีความสัมพันธ์กับตัวเอง กล่าวคือ ความสัมพันธ์ดังกล่าวนี้เกิดขึ้นในกรณีที่การวาดสับไทป์ในไวยากรณ์ แพกไทป์นี้จะเก็บข้อมูลของชื่อเอนติตี้ไทป์ที่เป็นซูเปอร์ไทป์ และชื่อของเอนติตี้ที่เป็นสับไทป์
- สับไทป์ LABEL TYPE มีความสัมพันธ์กับ LABEL ORDER เพื่อใช้บ่งว่าเลเบลไทป์ตัวใด ๆ ในไวยากรณ์มีลำดับเท่าไรในการอ้างอิง ของเอนติตี้ที่เกี่ยวข้องด้วย เกิดในกรณีที่การวาดเอนติตี้ไทป์ในไวยากรณ์ที่มีการอ้างอิง (reference) เลเบลไทป์มากกว่าหนึ่ง หรือกรณีที่มีการอ้างอิงแบบไฮราดี้ (ดังที่กล่าวไว้ในบทที่ 2 ขั้นตอนที่ 7 ของหัวข้อ " ขั้นตอนการออกแบบแบบจำลองข้อมูลไวยากรณ์ ")

2. กลุ่ม ROLE

- มีความสัมพันธ์กับเอนติตี้ไทป์ ROLE TYPE เพื่อเป็นตัวบ่งชี้ว่าโรลในไวยากรณ์ตัวใด ๆ เป็นโรลประเภท โรลปกติ หรือ โรลแบบเนสตั้ง (ซึ่งหมายถึงโรลที่อยู่คู่กับเนสตั้ง)
- มีความสัมพันธ์กับ ROLE ORDER เพื่อเก็บหมายเลขกำหนดลำดับของโรลภายในแพกไทป์ หรือ เรฟเฟอร์เรนซ์ไทป์ ของไวยากรณ์
- มีความสัมพันธ์กับ ROLE GROUP เนื่องจากจากการกำหนดคอนสเตรนท์ในไวยากรณ์จะกระทำบนโรลในรีเลชันชิฟไทป์ (แพกไทป์ และ เรฟเฟอร์ไทป์) ทั้งในลักษณะของการใช้โรลมากกว่า 1 โรล หรือการผนวกโรลเข้าไว้เป็นกลุ่ม เช่น การกำหนดคอนสเตรนท์แบบขั้วภายนอก ซึ่งต้องอาศัยกลุ่มโรล 2 กลุ่มในการกำหนด แต่ละกลุ่มอาจมีเพียง 1 หรือมากกว่า 1 โรลก็ได้ ความสัมพันธ์อันนี้จึงหมายถึง การกำหนดว่าโรลใดเป็นสมาชิกในกลุ่มโรลใดในไวยากรณ์ (หากไม่มีการกำหนดคอนสเตรนท์บนโรล โรลดังกล่าวก็ไม่จำเป็นต้องมีกลุ่ม)
- มีความสัมพันธ์กับ RELATIONSHIP TYPE เนื่องจากในแต่ละรีเลชันชิฟไทป์ประกอบด้วยโรลต่างๆ ดังนั้นทุกโรลที่ปรากฏในไวยากรณ์จะต้องมีการบันทึกที่อยู่ในรีเลชันชิฟใดรีเลชันหนึ่งเสมอ
- สับไทป์ NORMAL ROLE มีความสัมพันธ์กับ OBJECT TYPE เพื่อเป็นการบันทึก

ว่าโรคปกติใดต่ออยู่กับอ็อบเจ็คใดในรูปในแอม

- สับไทร์ NESTING ROLE มีความสัมพันธ์กับเอนติตี้ไทร์ NORMAL FACT TYPE เนื่องจากโรคนั้นเนสตั๋งต้องต่ออยู่กับเนสตั๋ง ซึ่งที่จริงแล้วโดยตัวเนสตั๋งเองก็มีฐานะเป็นแพทช์ไทร์ปกติตัวหนึ่งในรูปในแอม จึงต้องบันทึกโรคนั้นตั้งคู่ไปกับแพทช์ปกติที่เป็นเนสตั๋ง

3. กลุ่ม RELATIONSHIP TYPE

- มีความสัมพันธ์กับ ROLE ในลักษณะที่บอกว่าโรคชื่อใด อยู่ภายในรีเลชันใดในรูปในแอม

- มีตัวบ่งชนิดคือ KINDJOFJREL ซึ่งจะบ่งชนิดของรีเลชันชิฟไทร์ ว่าเป็นแพทช์หรือเรเฟอเร็นซ์ไทร์ หากเป็นแพทช์จะเป็นประเภทหาได้หรือประเภทปกติ

- สับไทร์ DERIVED FACT TYPE ซึ่งเป็นสับไทร์ของ FACT TYPE อีกทีหนึ่ง มีความสัมพันธ์กับ DERIVED FORMULA ซึ่งเป็นสูตรที่ใช้ในการหาค่าข้อมูลของแพทช์ไทร์แบบหาได้ โดยจะเก็บเป็นภาษาสร้างวิวของ SQL หรือ เก็บในลักษณะสูตรคำนวณก็ได้

4. กลุ่ม CONSTRAINT

- มีตัวบ่งชนิดคือ KINDJOFJCONST เพื่อบ่งว่าคอนสเตรนซ์ในรูปในแอมเป็นคอนสเตรนซ์ชนิดใด

- มีความสัมพันธ์กับ ROLE GROUP เนื่องจากคอนสเตรนซ์ที่เป็นข้อจำกัดที่กระทำอยู่บนโรค ซึ่งเป็นสมาชิกอยู่ในกลุ่มโรค คอนสเตรนซ์จึงต้องสัมพันธ์กับกลุ่มโรค แต่ก็ยังมีคอนสเตรนซ์บางประเภทที่ไม่ต้องยึดอยู่บนโรค ได้แก่คอนสเตรนซ์ในกลุ่มของอ็อบเจ็คไทร์คอนสเตรนซ์ที่ติดอยู่กับอ็อบเจ็คในในแอมแทน

- สับไทร์ OBJECT TYPE CONST เป็นคอนสเตรนซ์ประเภทที่ติดอยู่กับอ็อบเจ็คในในแอมเป็นหลัก จึงมีความสัมพันธ์กับ OBJECTJTYPE แทนที่จะสัมพันธ์กับ ROLE GROUP เช่นคอนสเตรนซ์ตัวอื่น ๆ

- สับไทร์ OCCURRENCE FREQ เป็นคอนสเตรนซ์ที่ต้องมีการกำหนดค่าความถี่และค่าความถี่ล่างคือ FREQUENCY

- สับไทร์ MEMBERSHIP CONST เป็นคอนสเตรนซ์ที่ต้องมีการประกาศสมาชิกที่จะใช้ จึงมีความสัมพันธ์กับ MEMBER ซึ่งเก็บสมาชิกที่ใช้ในคอนสเตรนซ์แบบเมมเบอร์ชิฟ

- สับไทร์ RANGE CONST เป็นคอนสเตรนซ์ที่ต้องมีการกำหนดค่าลิมิตสูงต่ำ จึงมีความสัมพันธ์กับ LIMIT ที่เก็บค่าลิมิต 2 ค่า คือค่าลิมิตบนและลิมิตล่าง

- สับไทย์ SUBSET CONST เป็นคอนสเตรนทที่ต้องทำการเก็บกลุ่มโรล ที่เป็นกลุ่มโรลที่มีสมาชิกน้อยกว่าอีกกลุ่มโรลหนึ่ง เรียกกลุ่มโรลนี้ว่า กลุ่มโรลที่เป็นสับเซต จากรูปเมื่อดูกันแล้วจะสังเกตเห็นว่ามีความสัมพันธ์อยู่กับ ROLE GROUP ถึง 2 ความสัมพันธ์ โดยที่ความสัมพันธ์หนึ่งเก็บในลักษณะแพทท์ไทย์ปกติ ได้แก่ความสัมพันธ์ที่ใช้บอกกลุ่มโรลที่เป็นสับเซตกับอีกหนึ่งความสัมพันธ์ที่เป็นแพทท์ไทย์แบบหาได้ มีสมาชิกคือกลุ่มโรลที่เป็นซูเปอร์เซต เนื่องจากด้วยความสัมพันธ์ที่เป็นแพทท์ไทย์แบบหาได้นี้ จึงต้องมีการกำหนดสูตรที่ใช้ในการหาค่าข้อมูล โดยกรณีนี้จะเป็ภาษา SQL ที่ใช้สร้างวิวในฐานะข้อมูล มีความว่า

```
CREATE VIEW SUPERSETCONST ( CONST# , ROLEGROUP ) AS
SELECT      T1.CONST# , T1.ROLEGROUP
FROM        CONSTROLE T1 , SUBSETCONST T2
WHERE       T1.CONST# = T2.CONST#
AND         T1.ROLEGROUP <> T2.ROLEGROUP ;
```

ซึ่งหมายถึง ให้หาค่าข้อมูลจากการเชื่อม (join) ระหว่าง 2 ตาราง คือตาราง CONSTROLE ที่เก็บความสัมพันธ์ระหว่าง CONSTRAINT และ ROLE GROUP และตาราง SUBSETCONST ที่เก็บความสัมพันธ์ระหว่าง CONSTRAINT และ ROLE GROUP ที่เป็นสับเซต โดยมีเงื่อนไขว่า ค่าข้อมูลที่จะอยู่ในความสัมพันธ์นี้ จะต้องอยู่ในตาราง CONSTROLE แต่ไม่อยู่ใน SUBSETCONST

3.3 เมื่อดูกันแล้วในลักษณะตารางรีเลชันนฐานข้อมูลแบบสัมพันธ์อกราเซล (Relational schema on ORACLE RDBMS)

จากในแอมคอนเซปชวลเมื่อดูกันแล้ว ที่ได้อธิบายในหัวข้อที่ผ่านมา ได้นำมาทำการแปลงตามกลไกการแปลงจากคอนเซปชวลเมื่อดูกันแล้ว เป็นเค้าร่างตารางฐานข้อมูลแบบสัมพันธ์ ได้ทั้งสิ้น 15 ตาราง กับอีก 1 วิว ทำการสร้างเมื่อดูกันแล้วในรูปตารางฐานข้อมูลเหล่านั้นลงบนฐานข้อมูลที่ดูแลโดยระบบจัดการฐานข้อมูล ORACLE ซึ่งใช้เป็นระบบจัดการฐานข้อมูลที่โครงงานนี้อ้างอิง (ดูรายละเอียดในภาคผนวก) เมื่อดูกันแล้วเหล่านั้นแสดงดังรูปข้างล่าง

(1) TABLE OBJECTTYPE

OBJECTNAME CHAR(20)	KINDOFOBJECT CHAR(10)
------------------------	--------------------------

(2) TABLE LABELTYPE

OBJECTNAME CHAR(20)	TYPENAME CHAR(20)	LABELORDER CHAR(5)
------------------------	----------------------	-----------------------

(3) TABLE SUBTYPE

SUBNAME CHAR(20)	SUPERNAME CHAR(20)
---------------------	-----------------------

(4) TABLE ROLE

ROLENAME CHAR(20)	ROLETTYPE CHAR(10)	RELATIONNAME CHAR(10)	ROLEORDER CHAR(1)
----------------------	-----------------------	--------------------------	----------------------

(5) TABLE OBJECTROLE

ROLENAME CHAR(20)	OBJECTNAME CHAR(20)
----------------------	------------------------

(6) TABLE NESTING

ROLENAME CHAR(20)	RELATIONNAME CHAR(10)
----------------------	--------------------------

(7) TABLE RELATIONTYPE

RELATIONNAME CHAR(10)	KINDOFRRELATION CHAR(10)
--------------------------	-----------------------------

(8) TABLE ROLEGROUP

ROLENAME CHAR(20)	ROLEGROUP CHAR(10)
----------------------	-----------------------

(9) TABLE CONSTRAINT

CONST# CHAR(10)	KINDOFRCONST CHAR(10)	CONSTNOTE CHAR(1)
--------------------	--------------------------	----------------------

(10) TABLE CONSTROLE

CONST# CHAR(10)	ROLEGROUP CHAR(10)
--------------------	-----------------------

(11) TABLE OBJECTCONST

CONST#	OBJECTNAME
CHAR(10)	CHAR(20)

(12) TABLE RANGECONST

CONST#	MINNUMBER	MAXNUMBER
CHAR(10)	CHAR(5)	CHAR(5)

(13) TABLE MEMBERSHIPCONST

CONST#	MEMBER
CHAR(10)	CHAR(10)

(14) TABLE SUBSETCONST

CONST#	ROLEGROUP
CHAR(10)	CHAR(10)

(15) VIEW SUPERSETCONST

CONST#	ROLEGROUP
CHAR(10)	CHAR(10)

```
CREATE VIEW SUPERSETCONST AS
SELECT T1.CONST# , T1.ROLEGROUP
FROM CONSTROLE T1 , SUBSETCONST T2
WHERE T1.CONST# = T2.CONST#
AND T2.ROLEGROUP <> T1.ROLEGROUP ;
```

(16) TABLE FREQUENCY CONST

CONST# CHAR(10)	UPPERLIMIT CHAR(5)	LOWERLIMIT CHAR(5)
--------------------	-----------------------	-----------------------

3.4 แผนภูมิการแมป (Mapping Information)

ในกระบวนการแมปที่คอนเซ็ปต์ของเรามาให้เป็นตารางฐานข้อมูลรีเลชันแนล โดยอาศัยโมเดลการสร้างเค้าร่างฐานข้อมูล (REDD-G : ซึ่งเป็นโมเดลที่พัฒนาขึ้นโดยกลุ่มโครงการระบบช่วยออกแบบฐานข้อมูลแบบสัมพันธ์ ส่วนโครงสร้างข้อมูล) นั้น จำเป็นต้องมีข้อมูลส่วนหนึ่งไว้ใช้ในกระบวนการแมป และรีเลชันที่ใช้เก็บข้อมูลของตารางฐานข้อมูลที่ได้จากกระบวนการ เช่น ชื่อตารางรีเลชันที่ได้ ชื่อวิว เป็นต้น การออกแบบตารางรีเลชันดังกล่าวนี้ ใช้แบบจำลองในแอมเช่นเดียวกันเรียวคอนเซ็ปต์ของเรามาว่า แผนภูมิการแมป (Mapping information) ประกอบด้วย 2 ส่วน คือ 1. ส่วนเก็บการอ้างอิงของเอนิตีในแบบจำลองในแอม และ 2. ส่วนเก็บชื่อของตารางรีเลชันที่ได้จากกระบวนการแมป

3.4.1 เอนิตีที่เพิ่มเข้ามาในส่วนแผนภูมิการแมป

1. COLUMN NAME ใช้ในการเก็บชื่อคอลัมน์ของรีเลชันที่ได้จากกระบวนการแมป มีความสัมพันธ์กับ ENTITY TYPE RELATIONSHIP TYPE ROLE และ LABEL TYPE ในลักษณะของเนตติ้ง เป็นความสัมพันธ์ที่ต้องเกิดขึ้นทุกครั้งที่มีการสร้างในแอมเพื่อแมปเป็นรีเลชัน

2. DATA TYPE ใช้ในการเก็บชนิดข้อมูลของแต่ละแอตทริบิวต์ที่อยู่ในรีเลชันที่ได้จากกระบวนการ แมป ชนิดข้อมูลดังกล่าวนี้ผู้ใช้งานระบบ (นักออกแบบ) จะต้องกำหนดไว้ตั้งแต่เริ่มวาดแบบจำลองในแอม มิฉะนั้นจะไม่สามารถสร้างรีเลชันได้ ชนิดข้อมูลจะมีความสัมพันธ์กับเลเบลไทป์ LABEL TYPE เนื่องจากเลเบลไทป์เป็นการอ้างอิงของเอนิตีในระบบจึงถือเป็นคุณสมบัติที่เอนิตีต่าง ๆ พึงจะมี

3. TABLE NAME ใช้ในการเก็บชื่อของรีเลชันที่ได้จากกระบวนการแมป มีความสัมพันธ์กับ REFERENCE TYPE และ NORMAL FACT TYPE เนื่องจากในกระบวนการแมปจะอาศัยรีเลชันชื่อไทป์ทั้งสองเป็นหลักในการสร้างความสัมพันธ์ที่เป็นรีเลชัน นอกจากนี้ยังมีความสัมพันธ์

พันท์กับ PRIMARY KEY ซึ่งจะกล่าวต่อไป

4. PRIMARY KEY ใช้ในการเก็บที่แอดดริบิวท์ที่เป็นไพรมารีคีย์ของรีเลชันที่ได้ เพื่อ บ่งบอกคุณสมบัติของรีเลชันนั้น ๆ

5. VIEW NAME ใช้ในการเก็บชื่อวิวที่ได้จากกระบวนการสร้างวิวจากเฟกท์ไทป์แบบหา ได้ของโนแอม

3.5 แมปปิงอินฟอร์เมชันในลักษณะตารางรีเลชันบนฐานข้อมูลแบบสัมพันธ์ออราเคิล

(Mapping Information in relational form on ORACLE RDBMS)

(1) TABLE UNIQJREF

ENTITYNAME	RELATIONNAME	ROLENAME	LABELNAME	COLNAME
CHAR(20)	CHAR(20)	CHAR(20)	CHAR(20)	CHAR(20)

(2) TABLE NONUNIQJREF

ENTITYNAME	RELATIONNAME	LABELNAME	COLNAME
CHAR(20)	CHAR(20)	CHAR(20)	CHAR(20)

(3) TABLE NORMALJFACTTYPE

RELATIONNAME	TABLENAME
CHAR(20)	CHAR(20)

(4) TABLE REFERENCE

RELATIONNAME CHAR(20)	TABLENAME CHAR(20)
--------------------------	-----------------------

(5) TABLE PRIMARYKEY

TABLENAME CHAR(20)	COLNAME CHAR(20)
-----------------------	---------------------

(6) TABLE DERIVEDFACTTYPE

RELATIONNAME CHAR(10)	VIEWNAME CHAR(20)	DERIVEDDEF CHAR(100)
--------------------------	----------------------	-------------------------

3.6 ข้อมูลรายละเอียดของเอกวิภังค์ต่าง ๆ ในตารางเมตาสเกอมาและแมปปิงอินฟอร์เมชั่น

ชื่อเอกวิภังค์	จำนวนอักขระ	ความหมาย
OBJECTNAME	20	เป็นตัวเก็บชื่อของ ENTITY หรือ LABEL
KINDOFOBJECT	10	จะเก็บชนิดของ OBJECTNAME ("LABEL","ENTITY")
TYPENAME	20	ชี้ให้เห็นว่า OBJECTNAME นี้ เมื่อ Map เป็นตารางแล้ว จะเก็บลักษณะข้อมูลเป็นอย่างไร เช่น C10 จำนวนอักขระ 10 ตัวอักษร, N10 ค่าตัวเลข

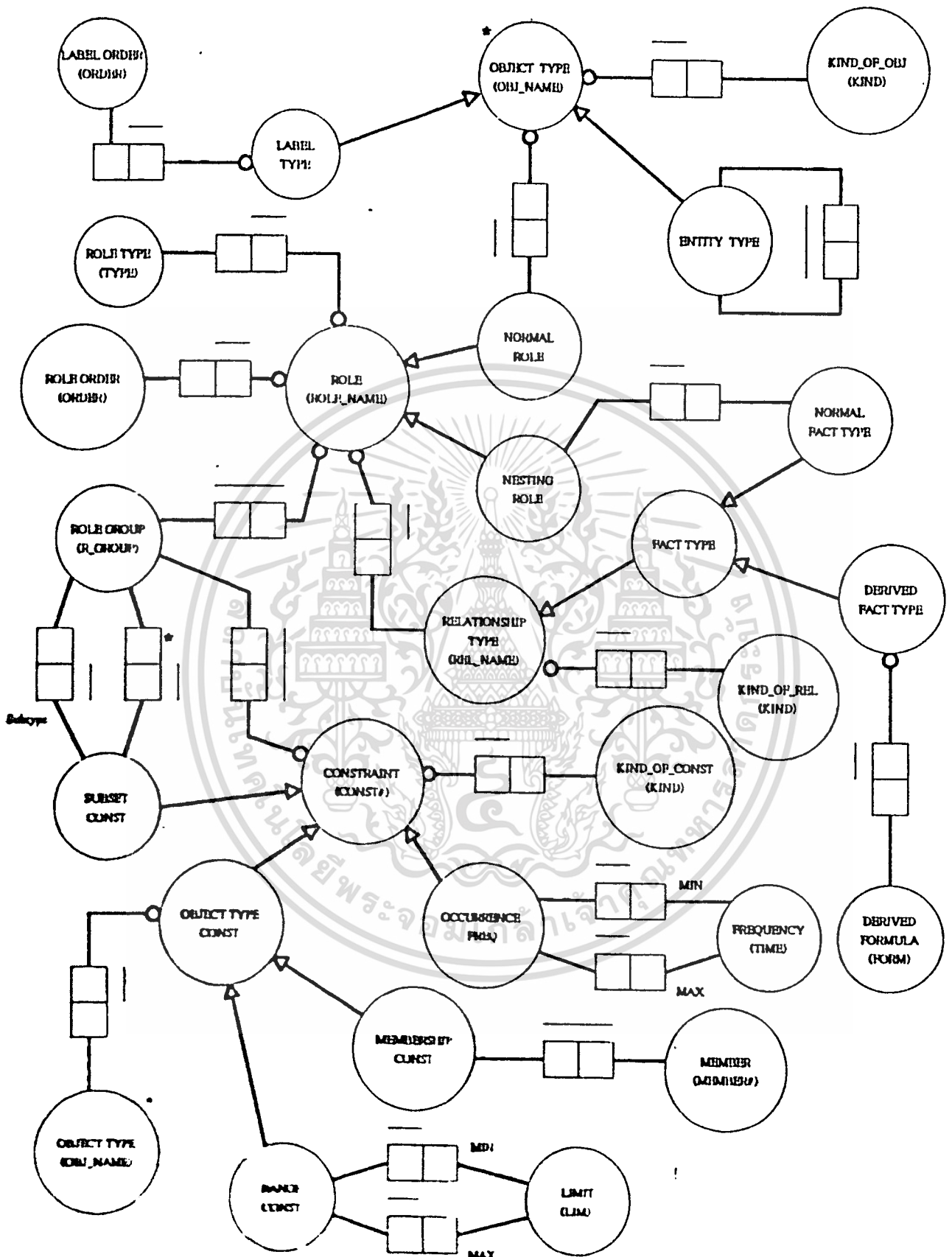
LABELJORDER	5	ขนาด 10 หลัก เป็นต้น เป็นตัวกำหนดลำดับค่าตัวเลขให้กับ LABEL ต่อ 1 ENTITY (1,2,3,4,5,...)
SUBJNAME	20	ชื่อ Subtype ของ Entity
SUPERJNAME	20	ชื่อ Supertype ของ Entity
ROLEJNAME	20	ชื่อของ ROLE
ROLEJTYPE	10	ชนิดของ ROLE ("NORMAL","NESTING")
ROLEJORDER	1	กำหนดลำดับของ ROLE (1,2,3,4,...,9)
RELATIONJNAME	10	ชื่อของความสัมพันธ์ (จะเก็บชื่อของ FACTJTYPE และ ชื่อของ REFERENCEJTYPE)
KINDJOFJRELATION	10	เป็นชนิดของความสัมพันธ์ ("FJNORMAL", "FJDERIVED","REF")
ROLEJGROUP	10	กลุ่มของ Constraint
CONST#	10	ID ของ Constraint (C1,C2,...,Cn)
KINDJOFJCONST#	10	ชนิดของ Constraint โดยจะมีอยู่ 7 ชนิดดังนี้ (UNIQUE,EXCLUSIV,EQUAL,SUBSET,RANGE, FREQ, MEMBER,MAND)
CONSTJNOTE	1	เป็น Flag แสดงว่าจะให้มีการเช็คConstraints หรือไม่ (N ไม่เช็ค , Y เช็ค)
MINJNUMBER	5	ความถี่ต่ำที่ใช้ในการกำหนด Occurence freq เก็บเป็นค่าจำนวนเต็ม
MAXJNUMBER	5	ความถี่บนที่ใช้ในการกำหนด Occurence freq. เก็บเป็นค่าจำนวนเต็ม
MEMBER	10	สมาชิกที่กำหนดใน Membership constraint เก็บเป็นสตริงขนาดไม่เกินที่กำหนด
UPPERJLIMIT	5	เป็นค่าตัวเลขที่มากที่สุดที่เก็บได้ในแต่ละ Object
LOWERJLIMIT	5	เป็นค่าตัวเลขที่น้อยที่สุดที่เก็บได้ในแต่ละ Object
ENTITYJNAME	20	ชื่อ Entity
LABELJNAME	20	ชื่อ Label

COLNAME	20	ชื่อ Column
TABLENAME	20	ชื่อตาราง
VIEWNAME	20	ชื่อวิว
DERIVEDDEF	100	สูตรที่ใช้ในการ Derived

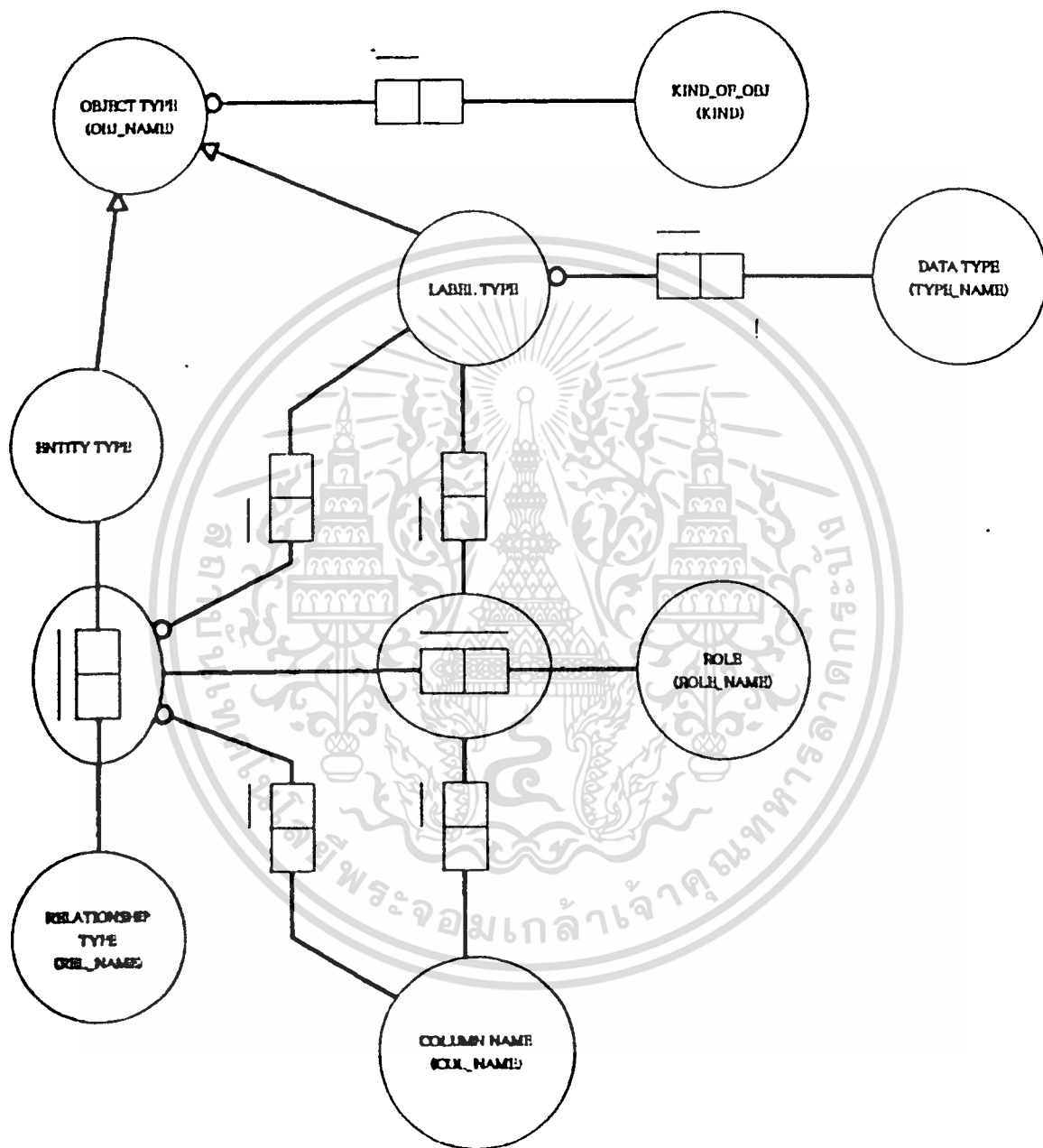


เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

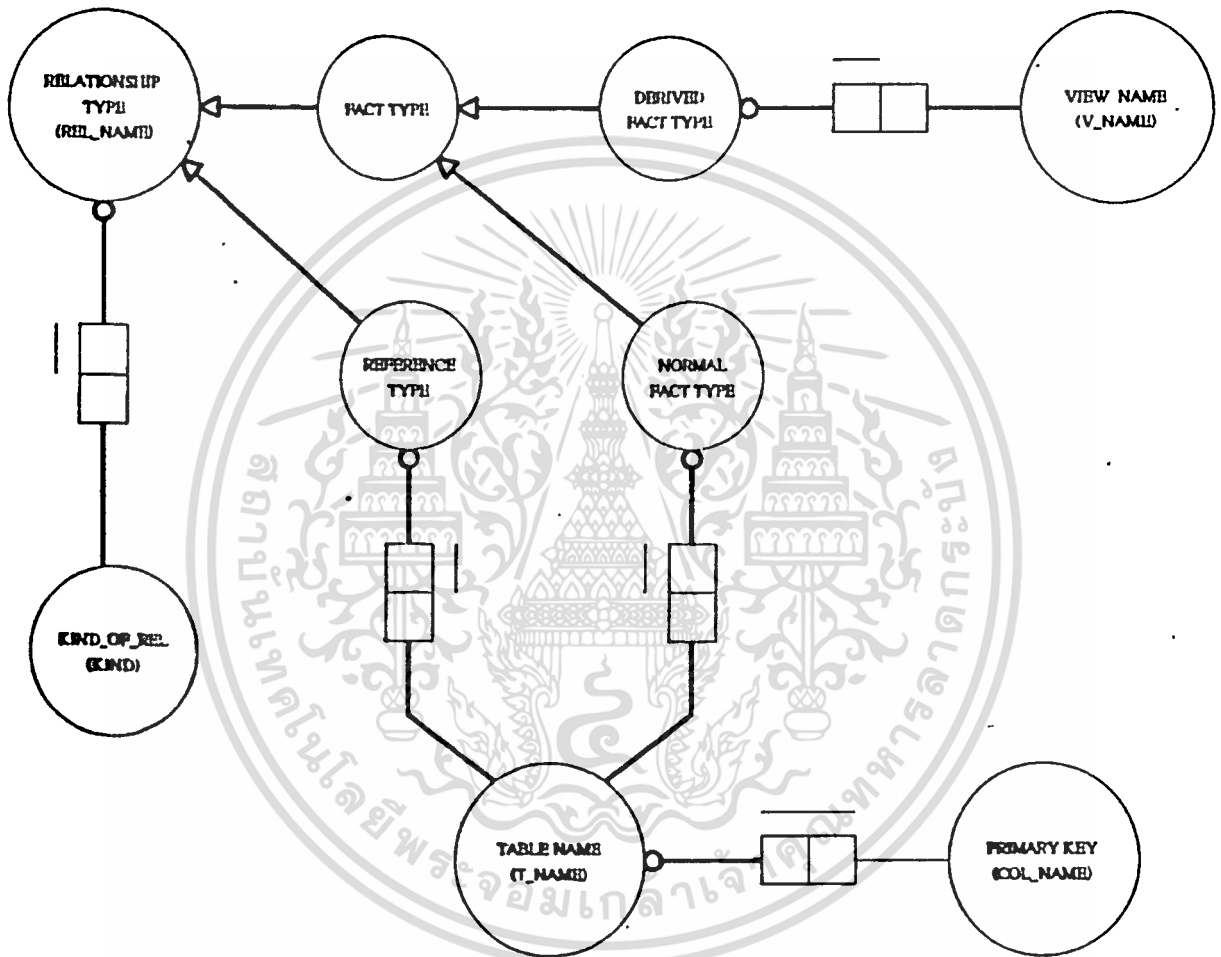
NIAM - CONCEPTUAL METASHEMA



MAPPING INFORMATION NO. 1



MAPPING INFORMATION NO.2



1

บทที่ 4 โดอะแกรมของระบบช่วยออกแบบฐานข้อมูลแบบสัมพันธ์

จากโดอะแกรมของระบบช่วยออกแบบฐานข้อมูลแบบสัมพันธ์ที่แสดง ประกอบด้วยกระบวนการจากโมดูลปฏิบัติการ 4 โมดูล อันได้แก่

1. REDD-Gr (นี้ออกหมายเลขที่ 1) เป็นกระบวนการรับอินพุตรูปแบบจำลองในแอมที่สร้างขึ้นภายใต้ระบบช่วยวาด ที่ทำงานภายใต้เอนไวรอนเมนต์ของระบบเชื่อมต่อผู้ใช้แบบกราฟิก หรือ จียูไอ (GUI:Graphic User Interface) แบบจำลองในแอมนี้ถูกออกแบบโดยผู้ใช้งานซึ่งก็คือนักออกแบบฐานข้อมูล โดยอาศัยขั้นตอนการออกแบบคอนเซ็ปชวลสกีมา (CSDP) ที่อธิบายไว้ในบทที่ 2 เรื่องหลักการเบื้องต้น เมื่อผู้ใช้ทำการเซฟแบบจำลองในแอม โปรแกรมช่วยวาดจะสร้างเอาต์พุตออกมา 2 ชนิด ได้แก่

1.1 ไฟล์เดสทริบเตอร์ คือเท็กซ์ไฟล์ซึ่งสร้างขึ้นโดยโปรแกรมช่วยวาด ใช้ในการบอกลักษณะของอ็อบเจ็คต่าง ๆ ของในแอมที่สร้างขึ้น ไม่ว่าจะเป็น เอนตี้ไทป์ เลเบิลไทป์ แพทไทป์ หรือ คอนสเตรนส์ต่าง ๆ รวมทั้งความสัมพันธ์ระหว่างอ็อบเจ็ค เช่น ความสัมพันธ์ในลักษณะการอ้างอิง (reference) ในลักษณะการถ่ายทอด (subtype) หรือ ลักษณะข้อมูลความจริง (fact type) โดยเก็บลักษณะเหล่านี้ในรูปแบบที่กำหนดขึ้น รายละเอียดของรูปแบบไฟล์เดสทริบเตอร์จะอยู่ในบทที่ 5 ต่อไป

1.2 เมต้าไฟล์ คือกราฟิกไฟล์ที่เป็นมาตรฐานในการเก็บภาพกราฟิกอีกแบบหนึ่งที่ระบบไมโครซอฟต์วินโดวส์เตรียมไว้สำหรับพัฒนาโปรแกรม มีลักษณะเป็นไฟล์ที่เก็บคำสั่งกราฟิกของระบบ จีดีไอ (GDI:Graphic Device Interface) ของไมโครซอฟต์วินโดวส์ เอนไวรอนเมนต์บนไมโครซอฟต์วินโดวส์ จะถือเสมือนหนึ่งว่าไฟล์ก็เป็นอุปกรณ์ภายนอกตัวหนึ่ง การใช้คำสั่งในการแสดงผลบนจอภาพ ด้วยฟังก์ชันในชุดโปรแกรมพัฒนาระบบของไมโครซอฟต์วินโดวส์ สามารถทำให้เกิดรูปภาพบนจอภาพ เช่นเดียวกับที่เกิดรูปภาพลงบนไฟล์ เมื่อมีการเปิดไฟล์ก็เหมือนกับการเปิดการทำงานของจอ การโหลดเมต้าไฟล์จากไฟล์ลงบนจอภาพ ก็เหมือนกับการกำหนดการแสดงผลจากอุปกรณ์หนึ่งไปยังอีกอุปกรณ์หนึ่ง (รายละเอียดของเมต้าไฟล์อยู่ในภาคผนวก)

2. REDD-MDB (นี้ออกหมายเลขที่ 6) กระบวนการสร้างเมต้าเดต้าเบส รับอินพุตในรูปแบบเดสทริบเตอร์ของแบบจำลองข้อมูลในแอม ที่ได้จากกระบวนการของโมดูล REDD-Gr

กระบวนการนี้จะนำเอาเดสคริปเตอร์ผ่านเข้าหักไว้ในโครงสร้างข้อมูลภายใน (in-core structure) แล้วทำการแปลงข้อมูลให้อยู่ในรูปของข้อมูลในทึพเพิลของรีเลชันฐานข้อมูลของเมตาสกีมมา เรียกเหล่านี้ว่าเมตาสกีมเบส จากไดอะแกรมจะพบจุดคลาการเพิ่มขึ้นในกลุ่มหนึ่งในบล็อกที่ 8 เรียกว่า ผู้พัฒนาเมตาสกีมมา ผู้พัฒนาเหล่านี้จะทำการค้นคว้าหาแบบจำลองระดับแนวคิดของเมตาสกีมมาเพื่อใช้แทนข้อมูล เอาท์พุทที่ได้จากกระบวนการแปลงนิยามข้อมูล รวมทั้งข้อกำหนดเพื่อความถูกต้องของแบบจำลองในแอม โดยมีรีเลชันฐานข้อมูลเมตาสกีมมารองรับอยู่ ก็คือเมตาสกีมเบสในบล็อกที่ 9 นั่นเอง

3. REED-A (บล็อกหมายเลขที่ 10) กระบวนการวิเคราะห์เมตาสกีมเบส เพื่อความถูกต้องของแบบจำลองข้อมูลในแอม กระบวนการนี้ประกอบด้วยกฎความถูกต้อง (NIAM Validation rules) เป็นหลัก การวิเคราะห์จะยึดถือกฎนี้เป็นบรรทัดฐาน หากการวิเคราะห์สัมฤทธิ์ผล คือเมตาสกีมเบสไม่มีข้อผิดพลาดร้ายแรง จะส่งต่อการทำงานไปยังกระบวนการสร้างรีเลชันฐานข้อมูลด้วยโมดูล REED-G ต่อไป แต่หากพบข้อผิดพลาดจะทำการทำลายเมตาสกีมเบสเสีย เพื่อไม่ให้นำไปทำการแมป

4. REED-G (บล็อกหมายเลขที่ 12) กระบวนการแมปเพื่อสร้างรีเลชันฐานข้อมูลของระบบงานที่ใช้คือผู้ใช้งานระบบ เอาท์พุทที่ได้อยู่ในลักษณะของรีเลชันและวิวภายในระบบจัดการฐานข้อมูลอราเคิล ภาษารูปแบบ SQL ที่ใช้ในการสร้างรีเลชัน และวิวในระบบจัดการฐานข้อมูลใด ๆ ที่เป็นแบบสัมพันธ์ และรูปรายงานแสดงภาพรีเลชันที่ได้รับจากกระบวนการ

4.1 สภาพแวดล้อมของระบบข้อมูลแบบ ๓

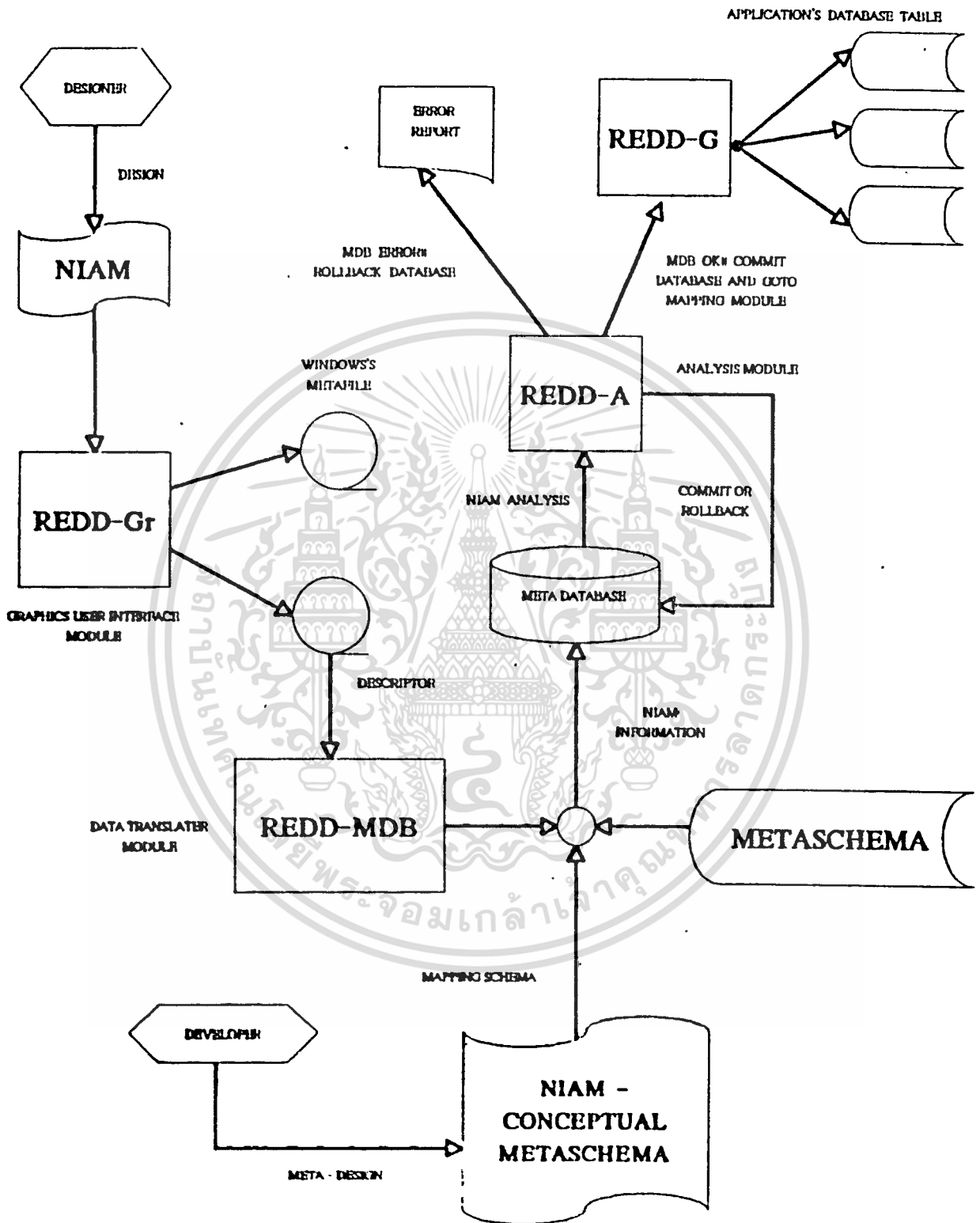
ระบบข้อมูลแบบฐานข้อมูลแบบสัมพันธ์ ทำงานอยู่ภายใต้ระบบปฏิบัติการ DOS ซึ่งทำงานบนเครื่องระดับไมโครคอมพิวเตอร์ โดยรีซอร์สแวร์กราฟิกเพนไวรอนเมนต์ ไมโครซอฟต์วินโดว์ 3.0 เป็นระบบเชื่อมประสานผู้ใช้ และใช้ระบบปฏิบัติการอราเคิล (ORACLE RDBMS) ในการดูแลฐานข้อมูลเมตาสกีมเบส และฐานข้อมูลของผู้ใช้งานระบบ ระบบคอมพิวเตอร์ที่จัดการในการทำงานของระบบข้อมูลแบบ ๓ มีดังนี้

1. เครื่องไมโครคอมพิวเตอร์ไอบีเอ็มพีซี หรือพีซีคอมแพคทีบิล ระดับเครื่อง 80286 ขึ้นไป 1 เครื่อง

2. หน่วยความจำภายในเครื่องไมโครคอมพิวเตอร์ขนาด 21 เมก้าไบต์
3. เมมส์ 1 ตัว
4. ฮาร์ดดิสก์ความจุอย่างน้อย 40 เมก้าไบต์
5. ซอฟต์แวร์ไมโครซอฟต์วินโดว เวอร์ชัน 3.0 ขึ้นไป
6. ซอฟต์แวร์ระบบปฏิบัติการอราเคิล (ORACLE RDBMS)



REDD - SYSTEM DIAGRAM



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โมดูลช่วยवादถือเป็นโมดูลแรกๆที่ผู้ใช้งานระบบออกแบบ ๙ จะพบเมื่อเข้ามาสู่สภาพแวดล้อมของระบบช่วยออกแบบ ๙ อยู่ในไฟล์โปรแกรมชื่อ NIAM.C มีลักษณะเป็นโปรแกรมที่ทำงานได้เฉพาะภายในเอนไวรอนเมนต์ของไมโครซอฟต์วินโดว์เท่านั้น ไม่สามารถทำงานภายใต้ระบบปฏิบัติการปกติ และต้องทำงานในโหมดการทำงานแบบเรียลไทม์ (real mode) ของไมโครซอฟต์วินโดว์เท่านั้น มิฉะนั้นจะไม่สามารถติดต่อกับระบบจัดการฐานข้อมูลออราเคิลได้ เนื่องจากข้อจำกัดบางประการก็จะกล่าวต่อไป

5.1 ความไม่เข้ากันระหว่างไมโครซอฟต์วินโดว์ และ ระบบจัดการฐานข้อมูลออราเคิล

ไมโครซอฟต์วินโดว์ เป็นระบบเอนไวรอนเมนต์แบบกราฟิกที่พัฒนาโดย Microsoft Corp. สามารถทำงานบนเครื่องไมโครคอมพิวเตอร์พีซีทุกรุ่น มีโหมดการทำงาน 3 โหมด คือ 1. เรียลไทม์ (real mode) 2. สแตนดาร์ดไทม์ (standard mode) และ 3. เอนฮานซ์ไทม์ (enhanced mode) โดยในเรียลไทม์ใช้ในเครื่องระดับพีซีเอ็กซ์ที หรือเครื่องที่ไม่มีหน่วยความจำแบบเอ็กซ์เทนส์ (extended memory) สแตนดาร์ดไทม์ใช้ในเครื่องระดับ 80286 ที่มีหน่วยความจำแบบเอ็กซ์เทนส์ และเอนฮานซ์ไทม์ใช้กับเครื่องระดับ 80386 DX หรือ 80386 SX หน่วยความจำแบบเอ็กซ์เทนส์ การทำงานในเรียลไทม์ระบบจะมีทรัพยากรจัดสรรให้กับโปรแกรมน้อยกว่าอีก 2 โหมด เนื่องจากทำงานในขอบเขตความสามารถเท่าที่ระบบปฏิบัติการดอสจะสนับสนุนให้ทำได้ (เช่น ใช้หน่วยความจำไม่เกิน 640 กิโลไบต์) ทำให้การทำงานของโปรแกรมต่าง ๆ ค่อนข้างช้า โดยเฉพาะในโปรแกรมที่ต้องการทรัพยากรระบบมากเป็นพิเศษ

ระบบจัดการฐานข้อมูลออราเคิล พัฒนาโดย Oracle Corp. เป็นระบบจัดการฐานข้อมูลแบบมีพันธะกับโหมดการทำงาน 2 โหมด คือ 1. โหมดติดต่อกับผู้ใช้โดยตรง (interactive mode) และ 2. โหมดพัฒนาโปรแกรม (embedded mode) โดยในโหมดติดต่อกับผู้ใช้โดยตรงจะกระทำผ่านโปรแกรม SQL-plus ซึ่งเป็นโปรแกรมจัดการภาษาฐานข้อมูล SQL และในโหมดพัฒนาโปรแกรมจะมีลักษณะเป็นภาษา SQL ที่ฝังตัวอยู่ในโปรแกรมภาษาโฮสต์ (host language) ที่พัฒนาโปรแกรมใช้ในการพัฒนาโปรแกรม ลักษณะของโหมดพัฒนาโปรแกรมจะ

เป็นลักษณะดังนี้

```
.....  
main () (  
    /* ภาษาโฮสต์ที่เป็นภาษาที่ */  
    .....  
    EXEC SQL ... /* ภาษา SQL */  
    .....  
)
```

จากรูปโปรแกรมเป็นโปรแกรมภาษาซีปกติ เพียงแต่ได้มีการฝังโค้ดของภาษา SQL ที่ใช้จัดการกับข้อมูลในฐานข้อมูล ไว้ในโปรแกรมภาษาซี ดังนั้นแล้วจะเรียกภาษาซีนี้ว่า ภาษาโฮสต์ (Host language) ภาษาโฮสต์สำหรับโหมดการทำงานแบบพัฒนาโปรแกรมนี้มีได้หลายภาษา ไม่ว่าจะเป็น โคบอล หรือ ปาสคาลก็ได้ขึ้นอยู่กับความสนับสนุนของโปรแกรมจัดการฐานข้อมูล (โดยปกติระบบจัดการฐานข้อมูลแบบสัมพันธ์จะสนับสนุนการทำงานในโหมดนี้อยู่แล้ว)

โปรแกรมที่มีภาษา SQL ฝังตัวอยู่ จะนำไปผ่านตัวแปลภาษาของภาษาโฮสต์ทันทีเลยไม่ได้ จะต้องนำไปผ่านตัวแปลภาษาพรีคอมไพเลอร์ (precompiler) เพื่อแปลภาษา SQL ในโปรแกรมให้เป็นภาษาโฮสต์เสียก่อน สำหรับในโครงการที่มีพัฒนาระบบงานด้วยการทำงานในโหมดนี้อยู่เช่นกัน โดยเลือกเอาภาษาซีเป็นภาษาโฮสต์ และใช้ซอฟต์แวร์โปรสตราซี (PRO* C) เป็นตัวแปลภาษาพรีคอมไพเลอร์ โปรสตราซีจะทำการแปลภาษา SQL ให้เป็นภาษาซี ที่ยึดตามไวยากรณ์ของไมโครซอฟท์เท่านั้น หมายความว่าตัวแปลภาษาซีที่จะต้องใช้ในโครงการ ต้องเป็นตัวแปลภาษาไมโครซอฟท์

ก่อนที่จะมีการรันโปรแกรมที่มีการจัดการฐานข้อมูลในโหมดการพัฒนาโปรแกรมอยู่ภายใน จะต้องทำการวิเคราะห์ระบบจัดการฐานข้อมูลออกมาเป็นलगงหน่วยความจำของเครื่องคอมพิวเตอร์เสียก่อน જેอนโตทงระบบจัดการฐานข้อมูลจะออกมาเป็นलगงหน่วยความจำเอ็ชเทนด์ที่ว่างอยู่ในระบบขณะนั้นไม่นักไปว่า 896 กิโลไบต์ เพราะฉะนั้นเครื่องคอมพิวเตอร์ที่มีหน่วยความจำ 2 เมกาไบต์ จะต้องแบ่งหน่วยความจำส่วนเอ็ชเทนด์ให้กับออราเคิลก่อนประมาณ 1 เมกาไบต์ที่เหลือจึงจะสวให้กับโปรแกรมอื่น ๆ ที่จะทำงานต่อไป ในขณะที่ตัวกับระบบไมโครซอฟท์วินโดวส์จะทำการตรวจสอบทรัพยากรระบบก่อนเข้าสู่การทำงาน หากขณะนั้นมีหน่วยความจำเอ็ชเทนด์ที่จะเข้าสู่โหมดการทำงานแบบสแตนด์ออลและ เทนทานท์โดยอัตโนมัติ ไมโครซอฟท์วินโดวส์

จะติดต่อกับทรัพยากรที่เป็นหน่วยความจำ อีกรุ่นหนึ่งโดยผ่านไดร์เวอร์ที่มีชื่อว่า HIMEM.SYS ในขณะที่การเกิดติดต่อกับไดร์เวอร์ จะเกิดปัญหาความเข้ากันไม่ได้ กล่าวคือทั้งสองระบบต่างก็พยายามจะจัดสรรทรัพยากรในระบบให้สนับสนุนการทำงานสูงสุดของระบบตนเอง โดยใช้วิธีการที่ต่างกัน ดังนั้นแล้วถึงแม้ว่าในขณะนั้นจะมีหน่วยความจำมากกว่า 2 เมกาไบต์อยู่ก็ตาม หากทำการเรียกอรรถาธิบายก่อน อรรถาธิบายซึ่งมีกระบวนการติดต่อกับหน่วยความจำแบบอีกรุ่นหนึ่งโดยตรงก็จะทำงานโดยอาศัยหน่วยความจำเพียง 896 กิโลไบต์ และเมื่อเรียกระบบไมโครซอฟต์วินโดวส์ตามเข้าไป วินโดวส์จะไม่สามารถค้นหาทรัพยากรที่เหลือได้ ก็ยังคงทำงานได้เพียงในโหมดการทำงานเรียลโหมดเท่านั้น ในทางกลับกันหากเรียกวินโดวส์ก่อน ไดร์เวอร์ HIMEM.SYS จะทำให้อรรถาธิบายไม่สามารถค้นหาทรัพยากรหน่วยความจำส่วนที่เหลือได้เช่นกัน

ดังนั้นแล้ว ระบบช่วยออกแบบฐานข้อมูลแบบสัมพันธ์กันจึงสามารถทำงานอยู่แต่เพียงโหมดการทำงานเรียลโหมดภายในไมโครซอฟต์วินโดวส์เท่านั้น

5.2 รูปแบบไฟล์เดสคริปเตอร์

จากไดอะแกรมของระบบงานในบทที่ 4 ที่กล่าวถึงไฟล์เดสคริปเตอร์ ซึ่งเป็นไฟล์เอาท์พุทที่เก็บลักษณะและความสัมพันธ์ของอ็อบเจกต์ในแบบจำลองในแอมเอชไอ ในลักษณะไฟล์ตัวอักษรนี้ รูปแบบของไฟล์ดังกล่าวเป็นดังรูปที่ 5.1

```

E/<name>      : <reference>
L/<name>      : <data type>,<type of objlconst>,<objlconst>
F/<name>      : <type>/<derived formula>,
                                     ( type - 1 Norm.
                                     - 2 Nest.
                                     - 3 Derived.
                                     - 4 Derived Nest. )
<role name No.1>/<role order>,<
<role name No.2>/<role order>,<
...
R/<name>      : <role name No.1>/<role order>-<flag>,<
<role name No.2>/<role order>-<flag>

```

(flag - 1 Unique.

- 0 Nun.)

C/<name> : <kind>,
(Unique,Mand,Exclusiv,Equal,Subset,Occurrence)
<role group No.1>
/ <role name No.1>-.../,
<role group No.2>
/ ... / , ...

c/r-<name> : <min>,<max>

c/m-<name> : <member1>+<member2>+...

c/o-<name> : <min>,<max>

c/s-<name> : <subset rolegroup>

*

R/<role name> + F/<entity name> (reference role + entity name)

R/<role name> + L/<label name> (reference role + label name)

F/<role name> + E/<entity name> (fact role + entity name)

F/<role name> + F/<nesting> (fact role + nesting)

<parent type> < <child type>

E/<entity name> : R/<reference type> (Enforced unique reference)

รูปที่ 5.1 แสดงรูปแบบไฟล์เดสคริปเตอร์

ไฟล์เดสคริปเตอร์ประกอบด้วยส่วน (session) 2 ส่วนขึ้นด้วยเครื่องหมายดอกจัน (*) ในส่วนที่หนึ่ง เก็บรายละเอียดของอ็อบเจกต์แต่ละตัวในโมเดล ไม่ว่าจะเป็นเอนทิตีไทป์ แฟกต์ ไทป์ หรือ ไรล และในส่วนที่สองเก็บความสัมพันธ์ของอ็อบเจกต์ในโมเดล รูปแบบในแต่ละส่วน มีดังนี้

1. ส่วนที่หนึ่ง

E/<name> : <reference>

เอนติตี้ไทย เก็บชื่อเอนติตี้ไทย และ ชื่อของเลเบล ไทยที่อ้างอิงภายในเอนติตี้ไทย

L/<name> : <data type>,<type of objlconst>,<objlconst>

เลเบลไทย เก็บชื่อเลเบลไทย ชนิดข้อมูล ประเภทของคอนสเตรนซ์แบบอ็อบเจ็คไทย และ ชื่อคอนสเตรนซ์นั้น (ถ้ามี)

F/<name> : <type>/<derived formula>.

<role name No.1>/<role order>.

<role name No.2>/<role order>.

...

แฟกต์ไทย เก็บชื่อแฟกต์ไทย ประเภทของแฟกต์ไทยว่าเป็นแบบหาได้หรือปกติ หากเป็นแฟกต์ไทยแบบหาได้จะตามหลังด้วยสูตรที่ใช้ในการหา ชื่อโรลภายในแฟกต์ไทยนั้นตามด้วยลำดับเลขที่ของโรลในแฟกต์ไทยนั้น ไม่เรียงไปจนครบตามจำนวนโรลที่มีอยู่

R/<name> : <role name No.1>/<role order>-<flag>.

<role name No.2>/<role order>-<flag>

(flag - 1 Unique.

- 0 Nun.)

เรฟเฟอร์เรนซ์ไทย เก็บชื่อเรฟเฟอร์เรนซ์ไทย ชื่อโรล 2 โรลโดยตามหลังด้วยลำดับที่ และแฟกต์ที่มีอยู่ 2 ค่า คือ 1 และ 0 หากเป็น 1 หมายถึงมีคอนสเตรนซ์แบบทุกคู่กำกับอยู่ในโรลนั้น และหากเป็น 0 หมายถึงไม่มี

C/<name> : <kind>.

(Unique,Mand,Exclusiv,Equal,Subset,Occurrence)

<role group No.1>

/<role name No.1>-.../.

<role group No.2>

/ ... / , ...

คอนสเตรนท์ เก็บชื่อคอนสเตรนท์ ชนิดของคอนสเตรนท์ซึ่งกำกับไว้ 6 ชนิด เฉพาะชนิดที่ต้องกำกับอยู่กับกลุ่มโรล ชื่อกลุ่มโรลตามด้วยลิสต์ของโรลสมาชิกภายในกลุ่มโรลนั้นที่ครอบด้วยเครื่องหมาย / และขึ้นด้วยเครื่องหมาย - ไล่ตามกลุ่มโรลที่สัมพันธ์กัน

c/r-⟨name⟩ : ⟨min⟩,⟨max⟩

คอนสเตรนท์แบบเรนจ์ เก็บชื่อคอนสเตรนท์ ค่าลิมิตบนและลิมิตล่างของคอนสเตรนท์แบบเรนจ์นั้น

c/m-⟨name⟩ : ⟨member1⟩+⟨member2⟩+...

คอนสเตรนท์แบบเมมเบอร์ชิพ เก็บชื่อคอนสเตรนท์ และสมาชิกทั้งหมดในคอนสเตรนท์นั้น ขึ้นด้วยเครื่องหมาย +

c/o-⟨name⟩ : ⟨min⟩,⟨max⟩

คอนสเตรนท์แบบอ็อกเคอเรนท์ ฟรีควินซ์ เก็บชื่อคอนสเตรนท์ และค่าความถี่บนความถี่ล่าง

c/s-⟨name⟩ : ⟨subset rolegroup⟩

คอนสเตรนท์แบบสับเซต เก็บชื่อคอนสเตรนท์ และชื่อกลุ่มโรลที่เป็นกลุ่มโรลสับเซต

2. ส่วนที่สอง

R/⟨role name⟩ + E/⟨entity name⟩ (reference role + entity name)

แสดงการนำโรลของเรเฟอเรนซ์โรลไทยเชื่อมกับเอนทิตีไทย โดยแทนด้วยชื่อโรลและชื่อเอนทิตีไทย

R/⟨role name⟩ + L/⟨label name⟩ (reference role + label name)

แสดงการนำโรลของเรเฟอเรนซ์โรลไทยเชื่อมกับเลเบลไทย โดยแทนด้วยชื่อโรลและชื่อเลเบลไทย

F/⟨role name⟩ + E/⟨entity name⟩ (fact role + entity name)

แสดงการนำโรลของแพทช์ไทป์เชื่อมกับเอนติตี้ไทป์ โดยแทนด้วยชื่อโรลและชื่อเอนติตี้ไทป์

$F/\langle \text{role name} \rangle + F/\langle \text{nesting} \rangle$ (fact. role + nesting)

แสดงการนำโรลของแพทช์ไทป์เชื่อมกับเอนติตี้ โดยแทนด้วยชื่อโรลและชื่อเอนติตี้

$\langle \text{parent type} \rangle < \langle \text{child type} \rangle$

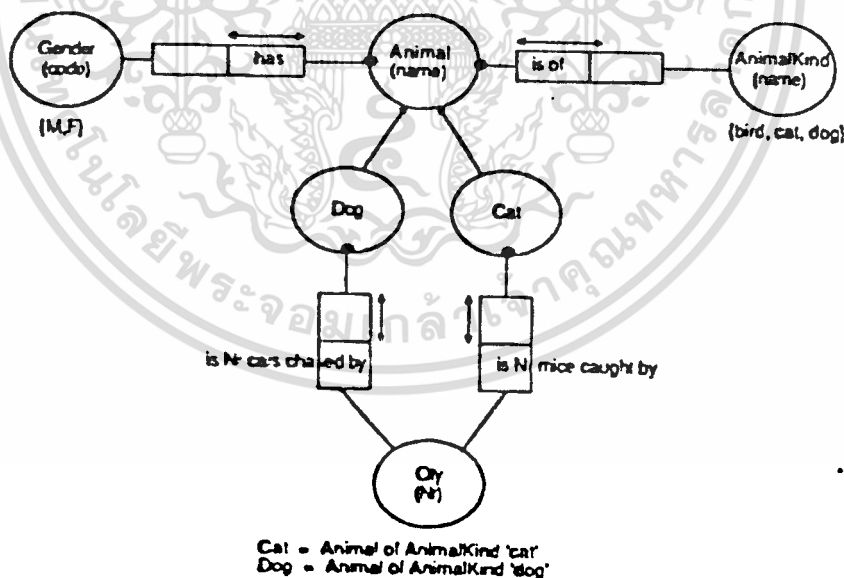
แสดงความสัมพันธ์ในลักษณะถ่ายทอด โดยใช้ที่กึ่งเปอร์ไทป์ และชื่อสับไทป์ ซึ่งก็คือชื่อเอนติตี้ไทป์ 2 เอนติตี้ไทป์ในการแทน

$E/\langle \text{entity name} \rangle : R/\langle \text{reference type} \rangle$ (Enforced unique reference)

แสดงความสัมพันธ์แบบบังคับให้เลือกการอ้างถึงในเรเฟเฟอร์เรนซ์ไทป์ที่กำหนด ในกรณีที่มีการอ้างถึงมากกว่า 1 เอนติตี้ไทป์ ใช้ชื่อเอนติตี้ไทป์และชื่อเรเฟเฟอร์เรนซ์ไทป์ในการแทน

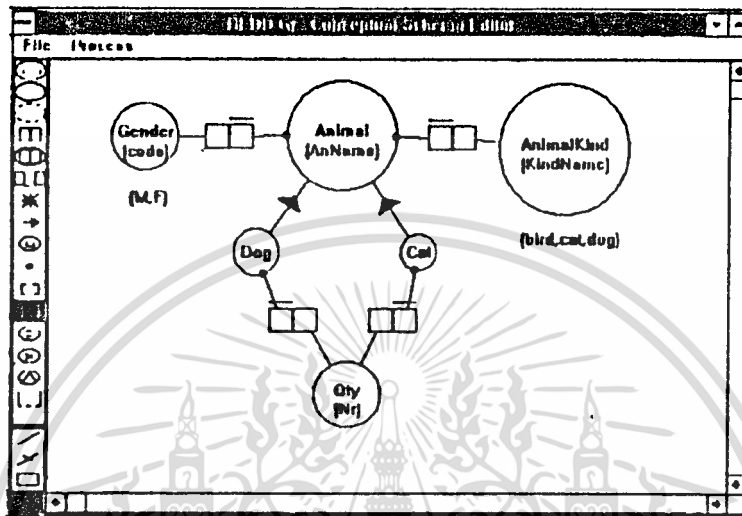
5.3 รูปโมเดลตัวกว้าง

แสดงรูปโมเดลที่จะใช้เป็นตัวอย่างในการทดสอบ ดังรูปที่ 5.2



รูปที่ 5.2 รูปโมเดลที่ใช้เป็นตัวอย่างในการทดสอบ

จากรูปตัวอย่างที่ 5.2 แสดงแบบจำลองในแอมของยูนิเวอร์สออฟดิสคอร์ดหรือยูโอดี (UoD: Universe of Discourse) ซึ่งหมายถึงเซตเหตุการณ์ที่น่าสนใจหนึ่ง ประกอบด้วยลักษณะความสัมพันธ์แบบสับไทย์ และความสัมพันธ์ปกติ ซึ่งจะนำไปวาดลงในโปรแกรมช่วยวาดได้รูปจากหน้าจอ ดังรูปที่ 5.3 และจากโปรแกรมช่วยวาด จะสร้างเดสคริปเตอร์ดังรูปที่ 5.4



รูปที่ 5.3 ภาพแสดงหน้าจอรูปตัวอย่างที่ 5.2 ของโปรแกรมช่วยวาด

/* This is file descriptor of example Figure 5.2

Generate by program module REDD-Gr */

E/GENDER : CODE

E/ANIMAL : ANIMALNAME

E/ANTMALKIND : KINDNAME

E/DOG : -

E/CAT : -

E/QTY : NR

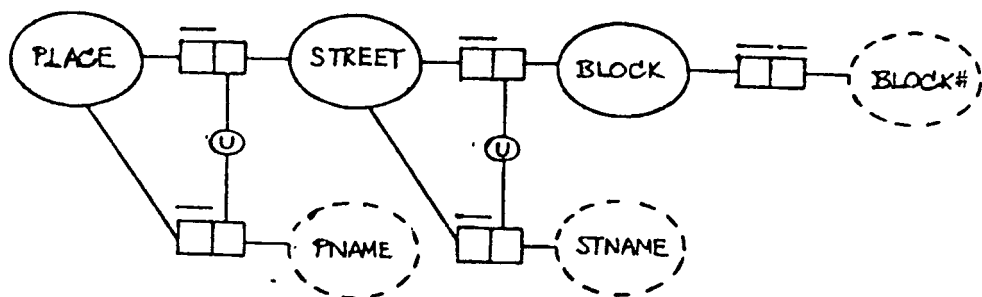
L/CODE : C10,1,C1

L/ANIMALNAME : C10,-,-

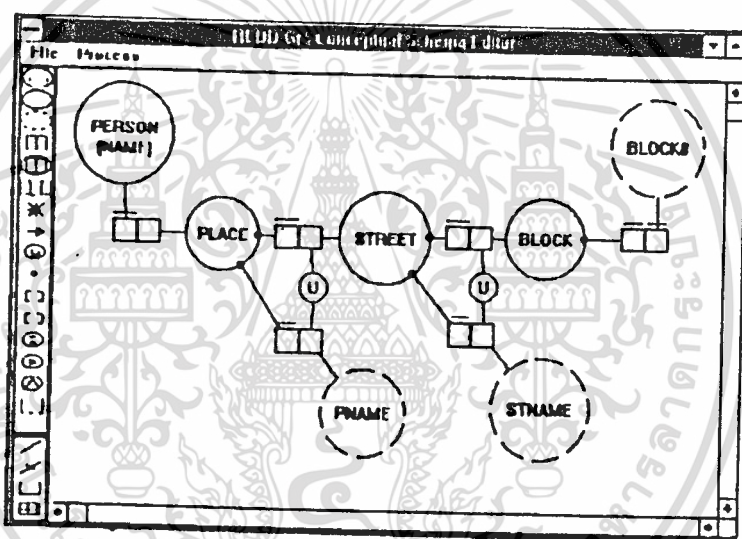
L/KINDNAME : C10,1,C4

L/NR : C10,-,-
 F/F1 : 1,R1/1,R2/2
 F/F2 : 1,R3/1,R4/2
 F/F3 : 1,R5/1,R6/2
 F/F4 : 1,R7/1,R8/2
 C/C2 : MAND,RG1/R2/
 C/C3 : MAND,RG2/R3/
 C/C5 : UNIQUE,RG1/R2/
 C/C6 : UNIQUE,RG2/R3/
 C/C7 : MAND,RG3/R5/
 C/C8 : MAND,RG4/R7/
 C/C9 : UNIQUE,RG3/R5/
 C/C10 : UNIQUE,RG4/R7/
 c/M-C1 : M+F
 c/M-C4 : BIRD+CAT+DOG
 *
 F/R1+E/GENDER , F/R2+E/ANIMAL , F/R3+E/ANIMAL , \
 F/R4+E/ANIMALKIND , \
 F/R5+E/DOG , F/R6+E/QTY , F/R7+E/CAT , F/R8+E/QTY , \
 ANIMAL<DOG , ANIMAL<CAT

รูปที่ 5.4 เดสคริปเตอร์ของรูปตัวอย่างที่ 5.2



รูปที่ 5.5 รูปไบนารีแอมพลีฟายด์ที่เป็นการอ้างถึงแบบไฮราคิ



รูปที่ 5.6 ภาพหน้าจอของไบนารีแอมพลีฟายด์ที่ 5.5

E/PLACE : -
 E/STREET : -
 E/BLOCK : -
 L/BLOCK# : C10, -, -
 L/PNAME : C10, -, -
 L/STNAME : C10, -, -

R/Ref1 : Ref1role1/1-1,Ref1role2/2-0
 R/Ref2 : Ref2role1/1-1,Ref2role2/2-0
 R/Ref3 : Ref3role1/1-1,Ref3role2/2-1
 F/Fact1 : 1,Fact1role1/1,Fact1role2/2
 F/Fact2 : 1,Fact2role1/1,Fact2role2/2
 C/Uni1 : UNIQUE,Rq1/Fact1role1/
 C/Uni2 : UNIQUE,Rq2/Fact2role1/
 C/Uni3 : UNIQUE,Rq3/Ref1role1/
 C/Uni4 : UNIQUE,Rq4/Ref2role1/
 C/Uni5 : UNIQUE,Rq5/Ref3role1/
 C/Uni6 : UNIQUE,Rq6/Ref3role2/
 C/Uni7 : UNIQUE,Rq12/Fact1role2-Ref1role2/
 C/Uni8 : UNIQUE,Rq13/Fact2role2-Ref2role2/
 C/Mda1 : MAND,Rq7/Fact1role1/
 C/Mda2 : MAND,Rq8/Fact2role1/
 C/Mda3 : MAND,Rq9/Ref3role1/
 C/Mda4 : MAND,Rq10/Ref1role1/
 C/Mda5 : MAND,Rq11/Ref2role1/
 *
 F/Fact1role1+E/PLACE ,\
 F/Fact1role2+E/STREET ,\
 F/Fact2role1+E/STREET ,\
 F/Fact2role2+E/BLOCK ,\
 R/Ref1role1+E/PLACE ,\
 R/Ref1role2+L/PNAME ,\
 R/Ref2role1+E/STREET ,\
 R/Ref2role2+L/STNAME ,\
 R/Ref3role1+E/BLOCK ,\
 R/Ref3role2+L/BLOCK# ,\

รูปที่ 5.7 เพลบริเบเตอร์ของตัวถ่วงรูปที่ 5.5

โมดูลแปลงแบบจำลองมีหน้าที่รับช่วงต่อจากโปรแกรมช่วยवाद โดยโปรแกรมช่วยवादจะเรียกใช้โปรแกรมชื่อ MAIN.EXE (ซึ่งเป็นโปรแกรมที่รวมเอา 2 โมดูลเข้าด้วยกัน คือ โมดูลแปลงข้อมูล และ โมดูลวิเคราะห์แบบจำลอง) ผ่านเมนู การทำงานจะสวิตซ์จากเอนไวรอนเมนต์ของไมโครซอฟท์วินโดวส์ ออกไปสู่ระบบจัดการฐานข้อมูลออรากเคิล ที่อยู่ในระบบเว็บบร็อกซ์แล้วในขณะนั้น โมดูลนี้จะนำเอาเดสคริปเตอร์ที่มาจากโมดูลช่วยवादของแบบจำลองในแอมมาทำการแปลงให้อยู่ในรูปของเมตาดาต้าเบส ภายในเมตาดาต้าเบสที่ถูกรวบรวมไว้ในฐานข้อมูลอยู่ก่อนแล้ว ไต่อะแกรมการทำงานของโมดูลนี้แสดงดังรูปที่ 6.1



รูปที่ 6.1 แสดงไต่อะแกรมของโมดูลแปลงแบบจำลอง

จากรูปหลังจากรับเดสคริปเตอร์มาแล้ว โมดูลแปลงจะทำการสร้างข้อมูลเก็บไว้ในโครงสร้างข้อมูลภายใน หรือ in-core structure โครงสร้างข้อมูลภายในนี้ออกแบบมาเพื่อรองรับกับเจ้าจากเดสคริปเตอร์เพื่อใช้ในการประมวลผลไปสู่เมตาดาต้าเบส มีลักษณะเป็นลิงก์ลิสต์ทางเดียว (singly link list) ของแต่ละก๊อบเจกต์ของในแอม รวมถึงโครงสร้างที่จะรองรับข้อมูลการเชื่อมระหว่างก๊อบเจกต์ โครงสร้างข้อมูลภายในนี้กำหนดไว้ในไฟล์เฮดเดอร์ชื่อ OBJECT.H ซึ่งจะแยกกล่าวในราชงานส่วนที่เป็นตัวโปรแกรม (Programming manual)

การทำงานของโมดูลแปลงแบบจำลองแบ่งออกเป็น 2 เฟส คือ

1. ขั้นตอนการแปลงเป็นโครงสร้างข้อมูลภายใน อยู่ในไฟล์โปรแกรมภาษาซีชื่อ X.C ซึ่งมีการอ้างถึงเฮดเดอร์ OBJECT.H

2. ขั้นตอนการแปลงจากโครงสร้างข้อมูลภายในเป็นเมตาดาต้าเบส อยู่ในไฟล์โปรแกรมภาษาซีที่มี SQL ฝังอยู่ทั้งหมด 6 ไฟล์ ซึ่งเป็นโปรแกรมย่อยที่ทำงานกับเมตาดาต้าทั้งหมด 6 ส่วน อันได้แก่

2.1 ไฟล์ 1.PC ทำการสร้างเมตาดาต้าเบสบนเมตาดาต้าในรูปรีเลชัน 5 รีเลชัน อันได้แก่รีเลชัน

- LABELTYPE
- ROLEGROUP
- CONSTROLE
- SUBSETCONST
- FREQUENCYCONST

2.2 ไฟล์ 2.PC ทำการสร้างเมตาดาต้าเบสบนเมตาดาต้าในรูปรีเลชัน 3 รีเลชัน อันได้แก่รีเลชัน

- OBJECTTYPE
- RELATIONTYPE
- OBJECTCONST

2.3 ไฟล์ 3.PC ทำการสร้างเมตาดาต้าเบสบนเมตาดาต้าในรูปรีเลชัน 3 รีเลชัน อันได้แก่รีเลชัน

- SUBTYPE
- OBJECTROLE
- NESTING

2.4 ไฟล์ 4.PC ทำการสร้างเมตาดาต้าเบสบนเมตาดาต้าในรูปรีเลชัน 3 รีเลชัน อันได้แก่รีเลชัน

- CONSTRAINT
- RANGECONST
- MEMBERSHIPCONST

2.5 ไฟล์ 5.PC ทำการสร้างเมตาดาต้าเบสบนเมตาดาต้าในรูปรีเลชัน 1 รีเลชันคือรีเลชัน ROLE และทำการสร้างข้อมูลอัตโนมัติที่เกิดขึ้นจากการอ้างถึงแบบภาคในตัวเอง เรียกว่าแบบทวนทวนว่า แบบอิงอัตโนมัติ (Auto

mapping)

2.6 ไฟล์ G.PC ทำการสร้างข้อมูลเมตาดาต้าแบบขนานตารางแมปिंगอินฟอร์เมชัน ที่ชื่อ UNIQUEREF และ NONUNIQUEREF ซึ่งใช้เก็บความสัมพันธ์ในการอ้างอิงของ เอนิตตี้ไทยแต่ละตัวในรูปในแอมพร้อมกันชื่อคอลัมน์ของแอตตริบิวท์ในรีเลชันที่ได้จาก กระบวนการแมปिंग เพื่อเป็นข้อมูลให้กับ โมดูลสร้างรีเลชันฐานข้อมูล REDD-G ใน กระบวนการแมปिंग

6.1 การอ้างถึงแบบไฮราคีในรีเลชัน UNIQUEREF

จากตัวอย่างรูปที่ 5.5 ที่แสดงถึงการอ้างอิงของเอนิตตี้ไทยในการที่ที่เป็นไฮราคี ในการนี้ ดังกล่าวนี้นำการแปลงข้อมูลเดสคริเบเตอร์ให้เป็นค่าข้อมูลในรีเลชัน UNIQUEREF มีความซับซ้อนสูง ในการพัฒนาโปรแกรมเพื่อรองรับส่วนนี้ โมดูลแปลงค่าข้อมูลใช้การมวนวิธการเรียกตัวเอง (Recursion) ตามอัลกอริทึมดังต่อไปนี้

```
WHILE FACTLIST ( 'NO FACT IN FACTLIST' ) {  
  GET FACT FROM FACTLIST;  
  WHILE FACT ( 'NO ROLE IN FACT' ) {  
    GET ROLE FROM FACT;          < ROLEJNAME >  
    ROLE -> ENTITY1;  
    GET ENTITY1;                  < ENTITYJNAME1 >  
    ENTITYJNAME ENTITY1->NAME;  
  
    ----- MODULE RECURSION -----  
    : INT RECURS ( ENTITY1 , ENTITYJNAME , ROLEJNAME )  
  
    IF ( ENTITY1.REF ) {  
      ENTITY1 + LABEL -> REL FROM METASCIHEMA;  
      INSERT ( ROLEJNAME , ENTITYJNAME1 , RELJNAME ,  
              LABELJNAME , - );  
      RETURN RECURS;          /* BACKING */  
    }
```

```

)
IF ( 'NO REF' ) {
    SEARCH PARENT TYPE OF ENTITY1;
    IF ( "FOUND RECURSIVE PARENT TYPE" ) {
        CALL RECURS ( PARENT TYPE , ENTITY1NAME ,
                      ROLE1NAME );
        RETURN ( WITH SUCCESS or ERROR );
    }
    ELSE
        RETURN RECURS ( WITH ERROR );
    /* RETURN ERROR */
}
ENTITY1 -> REL;
WHILE CONST1ROLE ( 'NO CONST1ROLE ANY MORE' ) {
    GET REL;
    IF ( REL = 'FACT' ) {
        REL -> ENTITY2;
        GET ENTITY2;
        CALL RECURS ( ENTITY2 , ENTITY2->NAME ,
                      ROLE1NAME );
        IF ( "RECURS ERROR" )
            RETURN RECURS ( WITH ERROR );
        SEARCH UNIQ1CONST;
        IF ( "FOUND UNIQ1CONST" )
            INSERT ( ROLE1NAME , ENTITY1NAME1 ,
                    REL1NAME , - , - );
        ELSE
            RETURN RECURS ( WITH ERROR );
    }
    ELSE {

```

```

        IF ( REL = 'FALSE REF' )
            !
            RETURN RECURS ( WITH ERROR );
            /* RETURN ERROR */
        IF ( REL = 'REF 1 TO 1'
            AND 'REF IS UNIQUE IDENT.' ) {
            GET REF;                < RELNAME >
            REF -> REFROLE;
            GET LABEL;              < LABELNAME >
            INSERT ( ROLENAME , ENTITYNAME1 ,
                RELNAME , LABELNAME , - );
            RETURN RECURS;
        }
        SEARCH UNIQCONST;
        IF ( 'NOT FOUND UNIQCONST' )
            RETURN RECURS ( WITH ERROR );
        GET REF;                < RELNAME >
        REF -> REFROLE;
        GET LABEL;              < LABELNAME >
        INSERT ( ROLENAME , ENTITYNAME1 ,
            RELNAME , LABELNAME , - );
    )
    FETCH NEXT CONSTROLE;
    CONSTROLE -> REL;
)
IF ( "MODULE RECURS ABNORMAL MAPPING" )
    ROI.BACK ABNORMAL. CASE;
NEXT ROLE;
)
NEXT FACT;
)

```

รูปที่ 6.2 แสดงอัลกอริทึมการแปลงค่าข้อมูลในรีเลชัน UNIQREF กวณาการอ้างอิงแบบไฮวาที

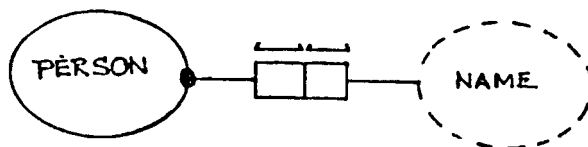
ENTITYNAME	RELATIONNAME	ROLENAME	LABELNAME	COLNAME
PLACE	R1	F1r1	PNAME	-
STREET	R2	F1r1	STNAME	-
BLOCK	R3	F1r1	BLOCK#	-
STREET	F2	F1r1	-	-
PLACE	F1	F1r1	-	-
STREET	R2	F1r2	STNAME	-
BLOCK	R3	F1r2	BLOCK#	-
STREET	F2	F1r2	-	-
STREET	R2	F2r1	STNAME	-
BLOCK	R3	F2r1	BLOCK#	-
STREET	F2	F2r1	-	-
BLOCK	R3	F2r2	BLOCK#	-
PERSON	RL*1	F3r1	NAME	-
PLACE	R1	F3r2	PNAME	-
STREET	R2	F3r2	STNAME	-
BLOCK	R3	F3r2	BLOCK#	-
STREET	F2	F3r2	-	-
PLACE	F1	F3r2	-	-

รูปที่ 6.3 แสดงค่าข้อมูลในรีเลชัน UNIQUEREF การอ้างถึงแบบไฮราดี้

6.2 การแมททิงอัตโนมัติ (Automapping)

แบบจำลองระดับแนวคิดของโปรแกรมจัดการการอ้างถึง หรือที่เรียกว่า เรฟเฟอเรนซ์ไทป์ ซึ่งใช้ในการอ้างถึงของเอนทิตีไทป์ของแบบจำลอง เอนทิตีไทป์หนึ่งต้องมีการอ้างถึงเลเบลไทป์ โดยใช้เรฟเฟอเรนซ์ไทป์ ที่มีความสัมพันธ์หนึ่งต่อหนึ่งอย่างน้อย 1 ความสัมพันธ์เสมอ ด้วยลักษณะดังกล่าวนี้ ไม่มีการระบุในแบบที่เอนทิตีไทป์มีความสัมพันธ์หนึ่งต่อหนึ่งกับเลเบลไทป์

หนึ่ง ให้เหลือเพียงเฉพาะเอนทิตีไทย ที่ม้วนเส้นแสดงการอ้างอิง ภายในวงเส้นนำเอาชื่อเอนทิตีไทยมาใส่แทน ดังรูป 6.4 และ 6.5



รูปที่ 6.4 เอนทิตีไทยมีการอ้างอิงแบบหนึ่งต่อหนึ่งต่อเอนทิตีไทย

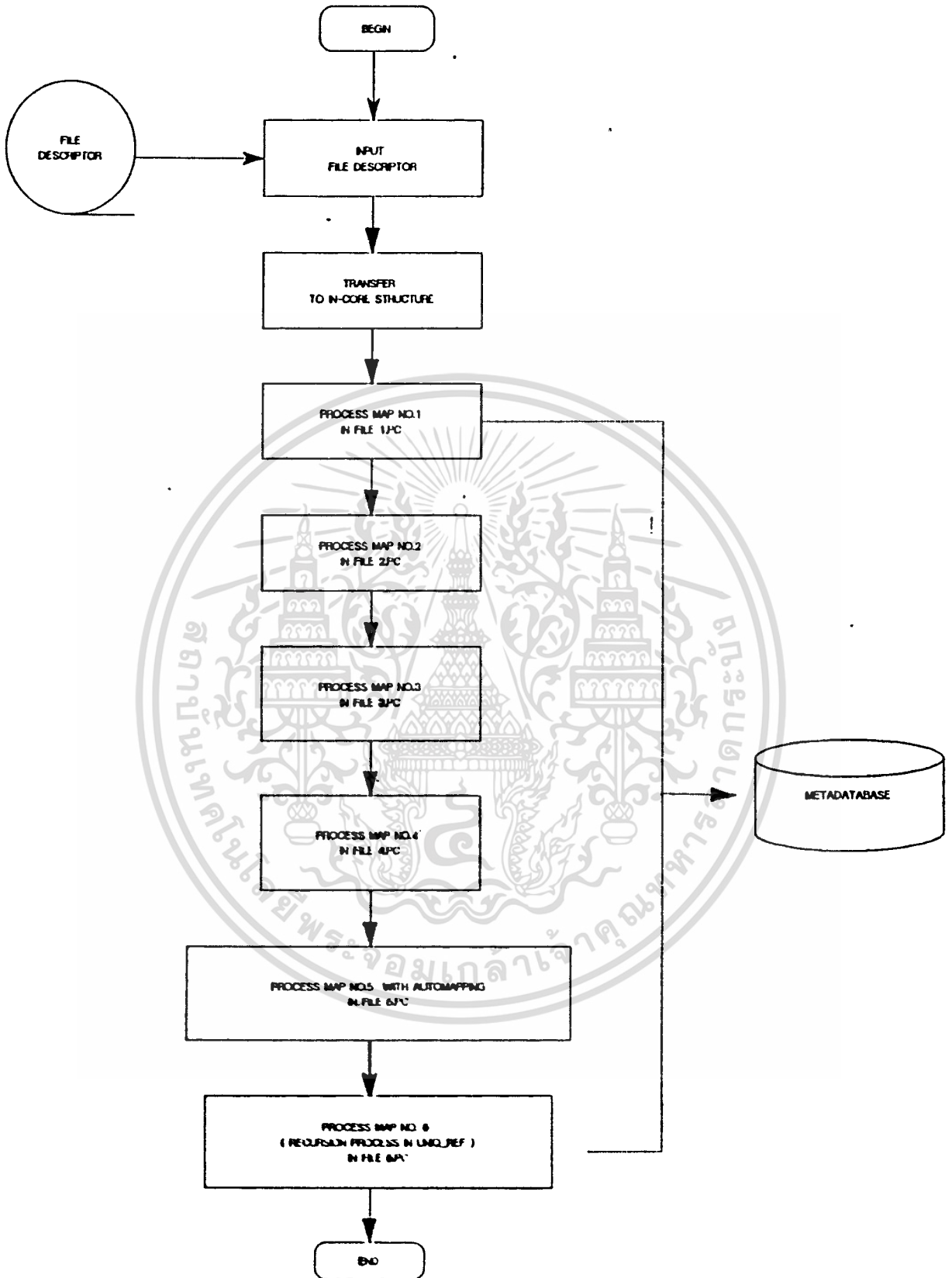


รูปที่ 6.5 แสดงรูปแบบที่ซ่อนแล้วของรูปที่ 6.4

ในรูปแบบจำลองที่วาดขึ้นโดยโปรแกรมช่วยวาด ผู้ใช้ไม่ได้กำหนดเรฟเฟอร์เรนซ์ไทยแต่ มักใช้รูปแบบชื่อย่ออยู่เสมอ แต่ในความเป็นจริงแล้ว ในการแปลงข้อมูลจะต้องคำนึงถึงอ็อบเจ็ค แอนเฟดต่าง ๆ ที่ไม่ปรากฏในรูปย่อกับผู้ใช้วาดขึ้นมา อันได้แก่ คอนสเตรนส์แบบชนิดที่ประจำ อยู่บนโวลของเรฟเฟอร์เรนซ์ไทย คอนสเตรนส์แบบแมนดาทอรี หรือ เรฟเฟอร์เรนซ์ไทย เป็น ต้น อ็อบเจ็คแอนเฟดในรูปเหล่านี้ โปรแกรมแปลงจะต้องสร้างขึ้นมาเอง โดยสมมติชื่อที่ใช้ขึ้น มาเอง และเก็บในรีเลชั่นที่เกี่ยวกับอ็อบเจ็คดังกล่าว เรีกการะบวนการนี้ว่า แมปปีงอัตโนมัติ การทำแมปปีงอัตโนมัติจะทำการบรีเลชั่นดังต่อไปนี้

- | | |
|-----------------|---|
| 1. OBJECTROLE | แมปปีงโวลของเรฟเฟอร์เรนซ์ไทยอัตโนมัติ |
| 2. ROLE | แมปปีงโวลและกลุ่มโวลของเรฟเฟอร์เรนซ์ไทยอัตโนมัติ |
| 3. ROLEGROUP | แมปปีงกลุ่มโวลของเรฟเฟอร์เรนซ์ไทยอัตโนมัติ |
| 4. CONSTRAINT | แมปปีงคอนสเตรนส์แบบชนิดและแบบแมนดาทอรีอัตโนมัติ |
| 5. CONSTROLE | แมปปีงคอนสเตรนส์แบบชนิดและแบบแมนดาทอรีกับโวลอัตโนมัติ |
| 6. RELATIONTYPE | แมปปีงเรฟเฟอร์เรนซ์ไทยอัตโนมัติ |

PROCESS IN REDD-MDB



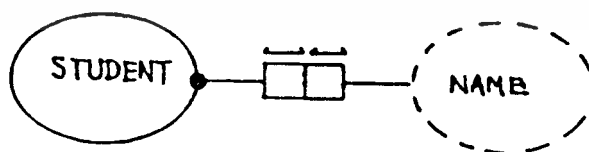
หลังจากได้เมตาดาต้าเบสของแบบจำลองในแอมของแอปพลิเคชันที่ได้จากการออกแบบของนักออกแบบฐานข้อมูล จากการแปลงข้อมูลในรูปเคสคริเบเตอร์ของโมเดลแปลงข้อมูล REDD-MDB แล้ว ขั้นตอนต่อไป คือการวิเคราะห์เมตาดาต้าเบสว่ามีความถูกต้องตามนิยามของแบบจำลองข้อมูลระดับแนวคิดในแอมหรือไม่ การวิเคราะห์ (analysis) แบบจำลองในแอม ในขอบเขตของโครงการนี้ ทำได้แต่เพียงการวิเคราะห์เชิงความถูกต้องของแบบจำลองเท่านั้น ยังไม่สามารถวิเคราะห์ถึงระดับความหมายของความสัมพันธ์ของข้อมูล การวิเคราะห์ในกรณีหลังนี้เรียกว่า " วิเคราะห์เชิงความหมาย (semantic analysis) " ทำได้โดยวิธีที่เรียกว่าตรวจสอบตัวประชากร (Population check) รายละเอียดอยู่ภายในหนังสือเรื่อง Conceptual Schema and Relational Database Design A Fact Oriented Approach ซึ่งเป็นหนังสืออ้างอิงเล่มหนึ่งของปริญาทิทัศน์ฉบับนี้

ในโมเดล REDD-A นี้ครอบคลุมการตรวจสอบใน 3 ระดับอันได้แก่

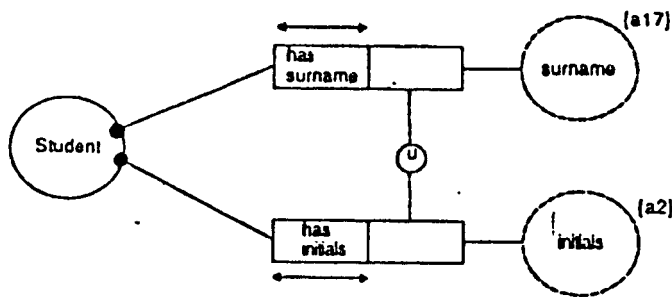
1. ตรวจสอบการเนมมิง (Naming analysis)
2. ตรวจสอบความบริบูรณ์ของการเชื่อมต่อ (Completeness analysis)
3. ตรวจสอบการใช้คอนสเตรนต์ (Constraint-checking analysis)

1. ตรวจสอบการเนมมิง (Naming analysis)

การเนมมิงในที่นี้คือการอ้างอิงเลเบล ไทป์ของเอนทิตี ไทป์ในแบบจำลองในแอมนั่นเอง การเนมมิงนี้ เป็นไปได้ตั้งแต่การเนมมิงแบบปกติคือเอนทิตีที่หนึ่งมีความสัมพันธ์หนึ่งต่อหนึ่งกับเลเบล ไทป์เพียงตัวเดียว ในรูปที่ 7.1 การอ้างอิงแบบเลเบล ไทป์มากกว่า (combine-label type) ดังรูปที่ 7.2 หรือการอ้างอิงในแบบไฮราร์คียังตัวอย่างรูปที่ 5.5



รูปที่ 7.1 การอ้างอิงเลเบล ไทป์แบบปกติ



รูปที่ 7.2 การอ้างอิงเลเบลไทม์นาม

สำหรับการที่เอนิตีไทม์หนึ่ง มีการเนมมิงถึงเลเบลไทม์มากกว่าหนึ่งตัวแบบหนึ่งต่อหนึ่ง ต้องเป็นหน้าที่ของนักออกแบบในแอม ที่จะกำหนดว่าจะเลือกเอาเลเบลไทม์ใดเป็นเนมมิงหลัก ไปรุกรมการทำงานในส่วนตรวจสอบเนมมิงนี้อยู่ในไฟล์ ANA.PC ซึ่งเป็นภาษาซีร่วมกับภาษา SQL โดยในส่วนตรวจสอบเนมมิงยึดเอากฎการเนมมิงเป็นบรรทัดฐาน ใช้การตรวจสอบด้วยภาษา SQL เนื่องจากมีความสามารถในการค้นหาข้อมูลในฐานข้อมูล (เมต้าเดต้าเบส) ได้ดีและรวดเร็ว อีกทั้งยังแก้ไขง่ายพัฒนาต่อได้ง่าย

กฎการเนมมิง (Naming Analysis rules)

- 1 ทุก ๆ เอนิตีไทม์ที่ปรากฏในในแอม จะต้องมีการกำหนดเลเบลไทม์ที่ใช้อ้างอิงในรูปของ เรฟเฟอร์เรนซ์ (reference) ภายในเอนิตีไทม์นั้น หากเอนิตีไทม์ไม่เป็นไปตามกฎให้พิจารณากฎข้อต่อไป.
- 2 เอนิตีไทม์ที่ไม่เป็นไปตามกฎที่ 1 นั้น จะต้องมีความสัมพันธ์กับเลเบลไทม์หรือเลเบลไทม์แบบคอมไบน์ (หมายถึงการประสานเลเบลไทม์มากกว่าหนึ่งตัวด้วยคอนสเตรนทชนิด Uniqueness ระหว่างเรฟเฟอร์เรนซ์ไทม์ที่เกี่ยวข้องในภาษิงกล่าว) ในลักษณะความสัมพันธ์แบบหนึ่งต่อหนึ่งอย่างน้อย 1 ความสัมพันธ์ หรือหากมีมากกว่า 1 ความสัมพันธ์ต้องประกาศเลเบลไทม์ใดตัวหนึ่ง เป็นตัวใช้งาน.

2. ตรวจสอบความบริบูรณ์ของการเนมมิง (Completeness analysis)

เมื่อเอนิตีไทม์ในแอมจำลองในแอมมีการเนมมิงที่ถูกต้องแล้ว ขั้นตอนต่อไปคือการตรวจสอบ

ว่าแบบจำลองในแอม มีความบริบูรณ์ในการเชื่อมต่อหรือไม่ ซึ่งหมายถึง การตรวจหาข้อบกพร่องได้ ในแบบจำลองที่มีความสัมพันธ์พื้นฐาน ไม่ครบถ้วนตามกฎของในแอม ความสัมพันธ์พื้นฐานดังกล่าวก็เช่น ทุก ๆ เอนติตี้ไทป์ต้องมีการเชื่อมต่อกับโรลใด ๆ อย่างน้อยหนึ่งโรลอยู่เสมอ เป็นต้น ไปรแกรมส่วนตรวจสอบความบริบูรณ์อยู่ในไฟล์ชื่อ ANA.PC

กฎของความบริบูรณ์ (Completeness analysis rules)

- 1 เอนติตี้ไทป์แต่ละตัว ต้องมีความสัมพันธ์กับโรลของแฟกต์ไทป์ (fact type role) อย่างน้อย 1 โรล นอกเสียจากว่าเอนติตี้ไทป์ดังกล่าว จะเป็นสับไทป์ของเอนติตี้ไทป์อื่น และในขณะเดียวกันก็เป็นเบสไทป์ของอีกเอนติตี้ไทป์หนึ่ง.
- 2 เลเบลไทป์ตัวหนึ่งต้องเชื่อมกับเรเฟอเร็นซ์ไทป์เพียงหนึ่งตัวเท่านั้น.
- 3 เอนติตี้ตัวหนึ่งต้องมีความสัมพันธ์กับโรลของแฟกต์ไทป์เพียง 1 โรล.
- 4 โรลใด ๆ ไม่ว่าจะ เป็นของแฟกต์ไทป์หรือของเรเฟอเร็นซ์ไทป์ จะต้องมีความสัมพันธ์กับเอนติตี้ไทป์ เลเบลไทป์ หรือ เอนตติ้ง อย่างใดอย่างหนึ่งเสมอ.
- * 5 ต้องไม่มีการประกาศสับไทป์ในลักษณะลูป (อยู่ในที่ตอนของโมดูล REDD-MDB)

3. การตรวจสอบการใช้คอนสเตรนต์ (Constraint-checking analysis)

แบบจำลองของในแอม มีลักษณะเด่นกว่าแบบจำลองอื่น ๆ ตรงที่สามารถกำหนดคอนสเตรนต์ ซึ่งเบี่ยงเบนจากข้อมูล ซึ่งจะนิยามบังคับความถูกต้องของข้อมูลได้อย่างสมบูรณ์ คือมีการใช้คอนสเตรนต์ทั้ง 8 ชนิดด้วยกัน ทำให้ผู้ออกแบบสามารถสร้างคอนเซ็ปชวลสเก็มมาที่แท้จริงได้ กล่าวคือสามารถมีทั้ง ข้อมูลความจริง (Stored fact type) กฎการถ่ายทอด (Derivation rules) และ ข้อจำกัดข้อมูลหรือคอนสเตรนต์ (constraint) ดังแสดงในรูปที่ 7.3 เพื่อใช้เป็นบรรทัดฐานในการสร้างระบบฐานความรู้ (Knowledge base) หรือระบบจัดการฐานข้อมูลที่มีประสิทธิภาพได้

Stored fact types	Constraints	Derivation rules
-------------------	-------------	------------------

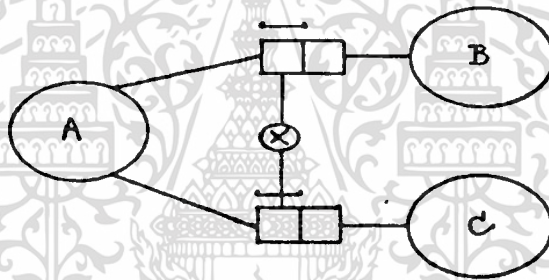
รูปที่ 7.3 องค์ประกอบของคอนเซ็ปชวลสเก็มมา

ขั้นตอนการตรวจสอบแบบจำลองในแอมจึงต้องรวมเอา ส่วนตรวจสอบความถูกต้องในการใช้งานคอนสเตรนต์ในรูปแบบในแอม ไม่ว่าจะเป็นความถูกต้องในด้านลักษณะการกำหนดคอนสเตรนต์ที่ชนิดต่าง ๆ การเกิดความซ้ำซ้อนของข้อบังคับที่มากเกินไปกว่าความจำเป็น หรือการเกิดความขัดแย้งกันเองระหว่างคอนสเตรนต์แต่ละตัว เป็นต้น โปรแกรมส่วนตรวจสอบคอนสเตรนต์อยู่ในไฟล์ชื่อ C.PC และอาศัยโปรแกรมในไฟล์ D.PC ในการลบรีเลชั่นชั่วคราว ที่โปรแกรมสร้างขึ้นในกระบวนการตรวจสอบ พร้อมกับสร้างรายงานการวิเคราะห์ด้วยโปรแกรมในไฟล์ชื่อ E.C โดยแสดงผ่านโปรแกรมส่วนจัดการหน้าจอซึ่งอยู่ในไฟล์ชื่อ SCR.C

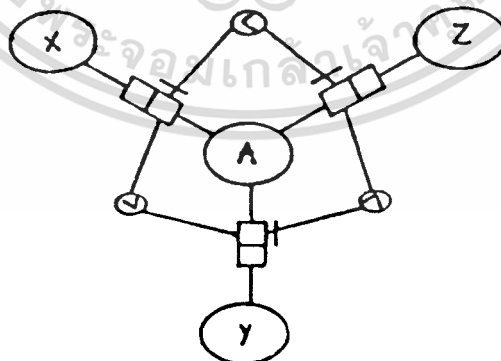
กฎของคอนสเตรนต์ (Constraint-Checking analysis rules)

- กฎไวยากรณ์ (Syntax rules)

1. คอนสเตรนต์แบบเซต กันประกอบไปด้วยคอนสเตรนต์ชนิดอ็ควาลิตี้ , เอ็กชคลูชันและชนิดสับเซต ต้องเชื่อมระหว่างโหนดของเอนิตีที่ใกล้เคียงกัน.

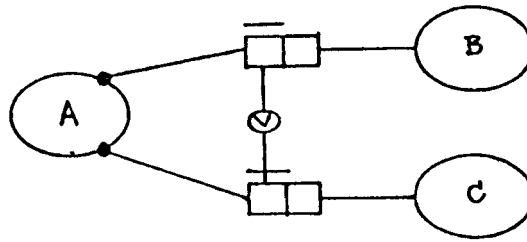


2. ไม่กำหนดคอนสเตรนต์ชนิด Subset ในลักษณะรูป

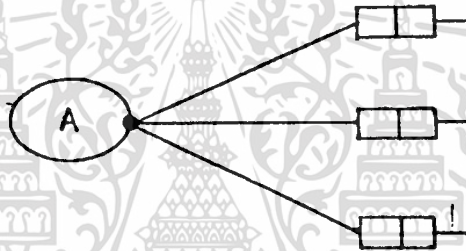


- กฎกล่าวเตือนคอนสเตรนต์เกิดความจำเป็น (Warning rules)

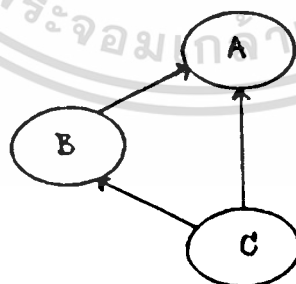
3 ไม่ควรกำกับคอนสเตรนซ์ชนิด Subset ไว้บนโรลของแพทช์ไทป์ ที่มีคอนสเตรนซ์ชนิด Mandatory กำกับอยู่แล้ว.



4 ไม่ควรกำกับคอนสเตรนซ์ชนิด Mandatory ที่ครอบคลุมทุกโรล ที่เชื่อมต่ออยู่กับเฮนดี้ไทป์เดียวกัน.



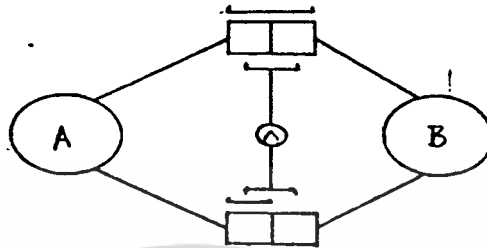
5 ไม่ควรประกาศสับไทป์ใดที่ไม่จำเป็น เนื่องจากมีลักษณะการถ่ายทอดอยู่แล้ว.



- กฎกล่าวเตือนเนื่องจากเงื่อนไขไม่สมบูรณ์

(Condition-incomplete warning rules)

6 ลักษณะของข้อมูลที่ถูกกำหนดโดยคอนสเตรนซ์ชนิด Uniqueness กับคอนสเตรนซ์ชนิด Subset ควรสอดคล้องกันทุกกรณี.



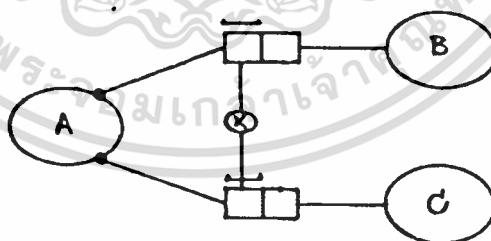
7 ไม่ควรกำหนดคอนสเตรนซ์ชนิด Uniqueness ในลักษณะซ้ำซ้อน.



- กฎความผิดพลาดอันเนื่องมาจากความขัดแย้งระหว่างคอนสเตรนซ์

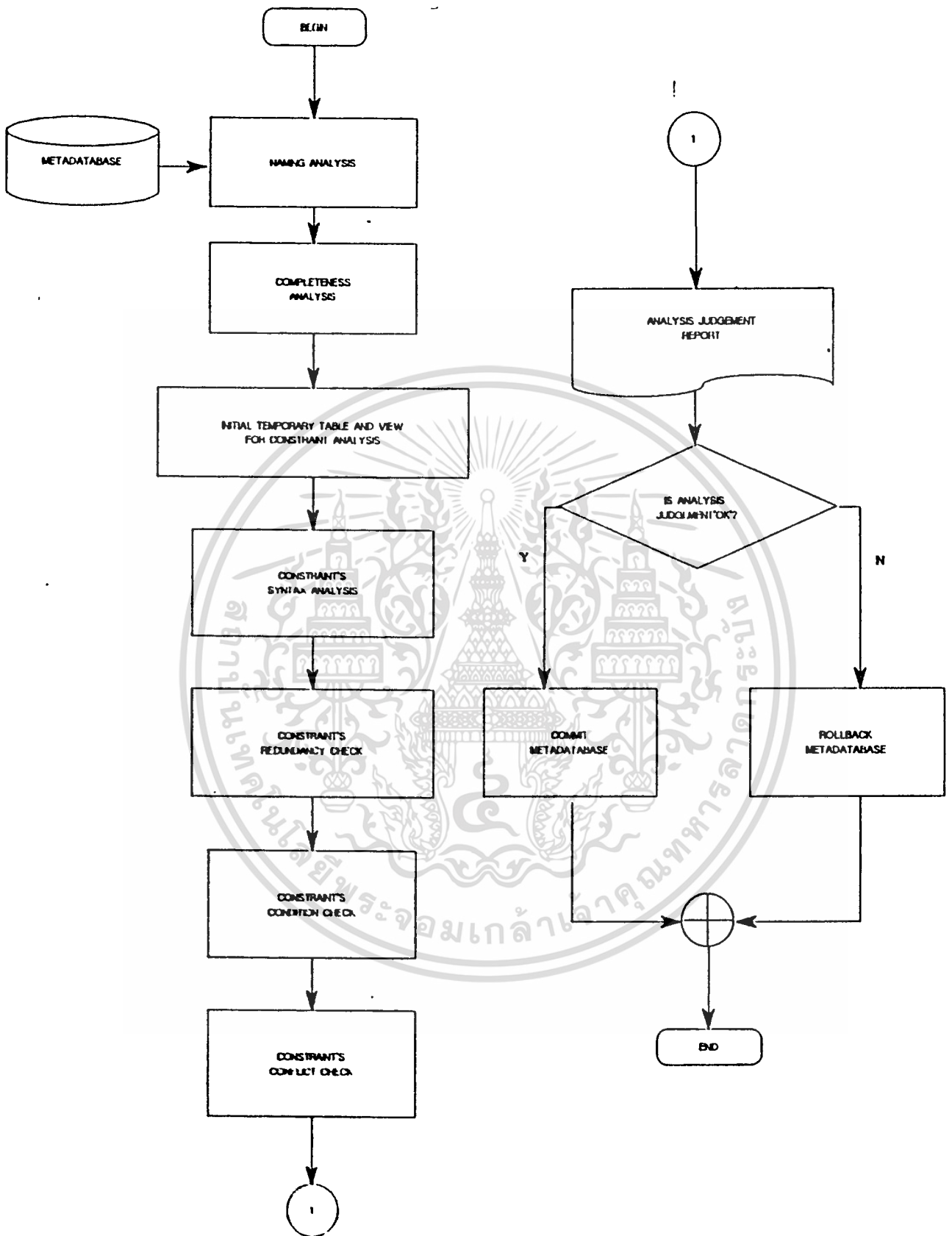
(constraint conflict error rules)

8 ต้องไม่กำกับคอนสเตรนซ์ชนิด Exclusion ระหว่างโวลที่ถูกกำกับโดยคอนสเตรนซ์ชนิด Mandatory อยู่ก่อนแล้ว.



9 สมาชิกที่อยู่ในเซตของคอนสเตรนซ์ชนิด Membership ที่กำกับบนเอนดตี้ไทป์หนึ่ง ต้องไม่ขัดแย้งกับลักษณะชนิดข้อมูล ที่กำหนดไว้ในเลเบลไทป์ของเอนดตี้ไทป์นั้น.

PROCESS IN REDD-A



8.1 เงื่อนไขในการทำงานของระบบช่วยออกแบบ ฯ

ระบบช่วยออกแบบฐานข้อมูลแบบสัมพันธ์ จะต้องทำงานอยู่บนระบบคอมพิวเตอร์ที่มีเงื่อนไขดังต่อไปนี้

1. หน่วยเก็บข้อมูลสำรองขนาดใหญ่ ในที่นี้คือฮาร์ดดิสก์ขนาดไม่ต่ำกว่า 40 เมกาไบต์ เพื่อใช้เป็นฐานข้อมูล
2. หน่วยความจำในระบบขนาด 2 เมกาไบต์
3. เมมส์ 1 ตัว
4. มีระบบจัดการฐานข้อมูลออร์ราเคิล (ORACLE RDBMS) ที่ปรากฏ USERID ชื่อ META ซึ่งมีรหัสผ่านว่า META เช่นกัน อยู่ในระบบเรียบร้อยแล้ว ซึ่งทำโดยการติดตั้งระบบจัดการผ่านทางโปรแกรมติดตั้งระบบ
5. มีซอฟต์แวร์เอนไวรอนเมนต์ไมโครซอฟต์วินโดวส์ อยู่ในระบบเรียบร้อยแล้ว
6. มีโปรแกรมระบบช่วยออกแบบฐานข้อมูล ซึ่งอยู่ในรูปเอกซ์คิวทีฟไฟล์ 2 ไฟล์ได้แก่
 1. โปรแกรม NIAM.EXE เป็นโปรแกรมที่ทำงานภายในระบบของไมโครซอฟต์วินโดวส์
 2. โปรแกรม DB.EXE เป็นโปรแกรมที่ทำงานกับระบบจัดการฐานข้อมูลออร์ราเคิลอยู่ในไดเรกทอรีเดียวกับไมโครซอฟต์วินโดวส์
7. มีรหัสพื้นฐานข้อมูลของเมตาดาทาอยู่ในระบบเรียบร้อยแล้ว (วิธีการติดตั้งอยู่ในภาคผนวก ง.)

8.2 เริ่มต้นการใช้งาน

หลังจากได้ทำการติดตั้งระบบจัดการฐานข้อมูลออร์ราเคิล และไมโครซอฟต์วินโดวส์ อยู่ในไดเรกทอรี C:\ORACLE5 และ C:\WINDOWS ตามลำดับแล้ว จะเริ่มใช้งานตามขั้นตอนดังต่อไปนี้

ขั้นตอนที่ 1 เริ่มต้นการทะเบียนจัดการออร์ราเคิล

พิมพ์คำว่า

C:\ORACLE5\> ORACLE <ENTER>

จะปรากฏข้อความลิขสิทธิ์ดังรูปที่ 8.1

Copyright (c) 1984, 1985, 1986. Oracle Corporation,
Belmont, California, USA. All rights reserved

This software has been provided under a license agreement
containing certain restrictions on use and disclosure.
Reverse engineering of object code is prohibited.

ORACLE V5.1.17.4DX - Development

Developer copy only. Production use prohibited by license agreement.

IOR: Version 5.1.17.4DX - Development on Mon Mar 16 23:50:41 1992

Copyright (c) 1986, Oracle Corporation, California, USA. All rights reserved.

IOR: connected to ORACLE V5.1.17.4DX - Development

IOR: System global area (SGA) is 70078 (69K) bytes composed of
1110 bytes of fixed data
25960 bytes of variable data
10240 bytes of before image buffers, and
32768 bytes of buffers.

IOR: instance 2 recovered

IOR: ORACLE warm started.

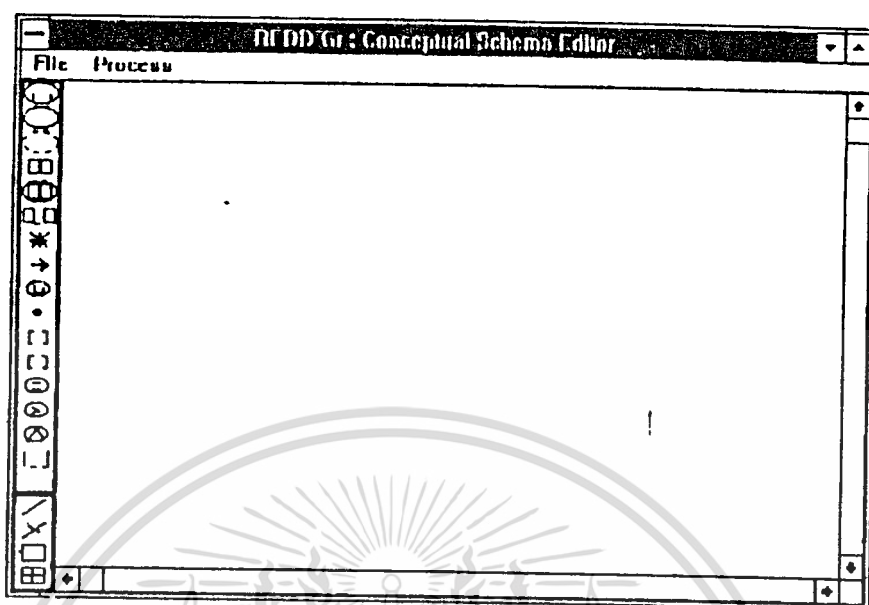
รูปที่ 8.1 แสดงหน้าจอเมื่อทำการสตาร์ทระบบจัดการออราเคิล

ขั้นตอนที่ 2 เรียกกระบวนการออกแบฐานข้อมูลจากไมโครซอฟต์วินโดวส์

พิมพ์คำว่า

C:\WINDOWS\> WIN/R NIAM <ENTER>

จึงจะแสดงหน้าจอดังรูปที่ 8.2



รูปที่ 8.2 แสดงหน้าจอเริ่มต้นของโปรแกรม NIAM.EXE ซึ่งเป็นส่วนโปรแกรมช่วยวาด

ขั้นตอนที่ 3 สร้างแบบจำลองข้อมูลในแอมจากโปรแกรมช่วยวาด

การวาด

3.1 วาดเอนติตี้ไทย - โดยการคลิกเมาส์ที่ตำแหน่งมุมซ้ายล่างสุดตาราง เพื่อใช้งานในโหมดการสร้างอีอบเจกต์ แล้วคลิกที่รูปเอนติตี้ไทยซึ่งมีอยู่ 2 แบบคือ แบบที่มีการอ้างอิงในตัว (มีวงเล็บ) และแบบไม่มีการอ้างอิง (ไม่มีวงเล็บ) เมื่อคลิกแล้วจะพบหน้าจอแสดงข้อความเพื่อให้ผู้ใช้กรอกชื่อและการอ้างอิงของเอนติตี้ไทย (ถ้าเป็นแบบมีวงเล็บ) หลังจากกรอกข้อความแล้ว จะปรากฏรูปร่างกลมขึ้นที่หน้าจอ ให้นำไปวาง ณ ตำแหน่งที่เหมาะสมโดยการลากเมาส์แล้วคลิก แสดงดังรูปที่ 8.3

3.2 วาดเลเิ้ลไทย - ใช้ขั้นตอนเกี่ยวกับการวาดเอนติตี้ไทย หน้าจอจะขึ้นไดอะล็อกมาถามข้อมูลเช่นเดียวกัน ดังรูปที่ 8.4

3.3 วาดพลาท์ไทย - ก่อนจะวาดพลาท์ไทย ต้องวาดพลาท์ไทยร่อยอยู่ 2 ตัวก่อนแล้ว เมื่อต้องการสร้างพลาท์ไทย ให้คลิกที่รูปพลาท์ไทยที่มีลักษณะเป็นสี่เหลี่ยม 2 รูปเชื่อมด้วยเส้นได้

หน้าจอก็จะถามจำนวนโรลภายในแพทช์ไทป์นั้น บอกจำนวนโรลพร้อมกับลากแพทช์ไทป์ที่ได้ไปยังตำแหน่งที่ต้องการ คลิกที่เอนดิ์ไทป์หนึ่งเพื่อเชื่อมโรล คลิกเอนดิ์ต่อไปเพื่อเชื่อมให้ครบทุกโรล โดยเรียงจากซ้ายไปขวาจนครบ หลังจากนั้นคลิกที่โรลที่ต้องการใส่คอนสเตรนซ์แบบยูนิค หากต้องการคอนสเตรนซ์แบบร่วม (combine-unique) ก็คลิกที่โรลถัดไปตามลำดับ หากต้องการสร้างยูนิคมากกว่าหนึ่งตัว กดตัวอักษร U เมื่อเคอร์เซอร์อยู่ในตำแหน่งที่ต้องการก่อนที่จะคลิกที่โรลนั้น

3.4 วาดเรฟเฟอร์เรนซ์ไทป์ - เช่นเดียวกับการวาดแพทช์ไทป์ ต่างกันเพียงจะทำการคลิกเชื่อมโรล โดยคลิกที่เอนดิ์ไทป์ตัวหนึ่งแล้วคลิกที่เลเบลไทป์ตัวหนึ่ง

3.5 วาดเนสตั้ง - ก่อนการวาดเนสตั้งต้องมีการสร้างแพทช์ไทป์ที่ต้องการทำเป็นเนสตั้งไว้ก่อนแล้ว ทำการคลิกที่เครื่องหมายเนสตั้ง แล้วคลิกที่แพทช์ไทป์ที่ต้องการทำเป็นเนสตั้ง

3.6 วาดแพทช์ไทป์แบบหาได้ - เช่นเดียวกับการวาดแพทช์ไทป์ปกติ สร้างได้โดยการคลิกที่เครื่องหมายดอกจัน หน้าจอก็จะปรากฏแพทช์ไทป์แบบหาได้ที่มีเครื่องหมายดอกจันกำกับอยู่

3.7 วาดสับไทป์ - ก่อนการสร้างสับไทป์ต้องมีเอนดิ์ไทป์อยู่ก่อนแล้ว 2 เอนดิ์ไทป์ ทำการคลิกที่เครื่องหมายลูกศร แล้วคลิกที่เอนดิ์ที่กำหนดให้เป็นซูเปอร์ไทป์ แล้วตามด้วยการคลิกที่สับไทป์ จะเกิดเครื่องหมายลูกศรคือสับไทป์ขึ้น โดยชี้จากเอนดิ์ไทป์ที่เป็นสับไทป์ ไปยังเอนดิ์ไทป์ที่เป็นซูเปอร์ไทป์

3.8 วาดคอนสเตรนซ์แบบยูนิคภายนอก - โดยการคลิกที่เครื่องหมายรูปตัว U ในวงกลม ลากวัดทางกลมไปวางทั้งตำแหน่งที่ต้องการ คลิกเชื่อมโรลกับเครื่องหมายยูนิคภายนอก 2 โรล หรือมากกว่า

3.9 วาดคอนสเตรนซ์แบบแมนดาทอรี - คลิกที่เครื่องหมายรูปจุดดำ นำเมาส์ไปคลิกที่โรลที่จะกำกับแมนดาทอรี จะเกิดแมนดาทอรีขึ้นที่ตัวเอนดิ์ไทป์ที่เชื่อมโรลนั้นอยู่ หากต้องการกำกับแมนดาทอรี เป็นกลุ่มโรลให้ทำการคลิกที่โรลต่อไปทันที แต่ถ้าต้องการเข้าสู่โหมดการทำงานปกติก็กดปุ่ม ESC

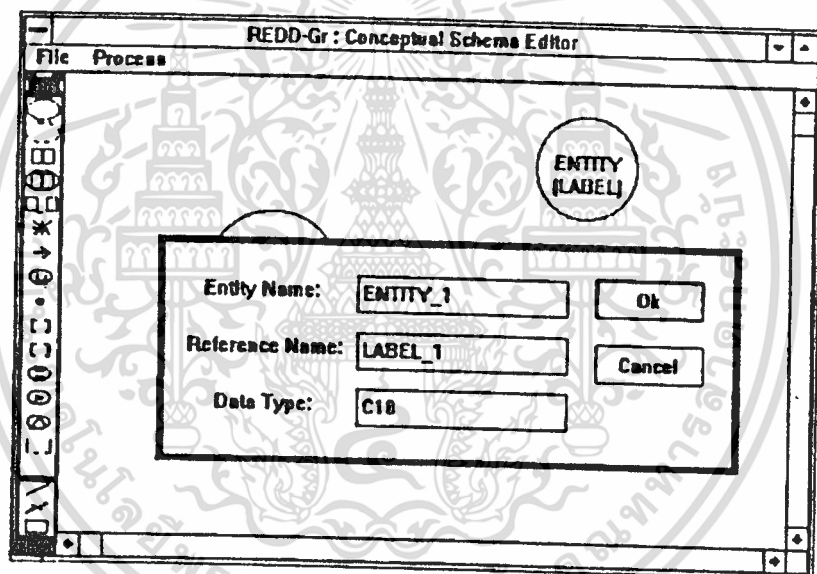
3.10 วาดคอนสเตรนซ์แบบเรนจ์ - คลิกที่เครื่องหมายรูปก้ามปู ([]) โดยก่อนหน้านั้นต้องมีเอนดิ์ไทป์ที่ต้องการกำกับเรนจ์อยู่แล้ว ลากเอาเรนจ์ไปวางในตำแหน่งใกล้ ๆ เอนดิ์ไทป์ที่ต้องการกำกับ คลิกเมาส์เพื่อวางตำแหน่งคอนสเตรนซ์ แล้วคลิกที่ตัวเอนดิ์ไทป์นั้นอีกครั้งเพื่อกำกับคอนสเตรนซ์

3.11 วาดคอนสเตรนซ์แบบเมมเบอร์ชิพ - มีลักษณะการสร้างเช่นเดียวกับคอนสเตรนซ์แบบเรนจ์ เนื่องจากเป็นคอนสเตรนซ์ที่ใช้กำกับเอนดิ์ไทป์เหมือนกัน คลิกเลือกวัดกึ่งที่เครื่อง

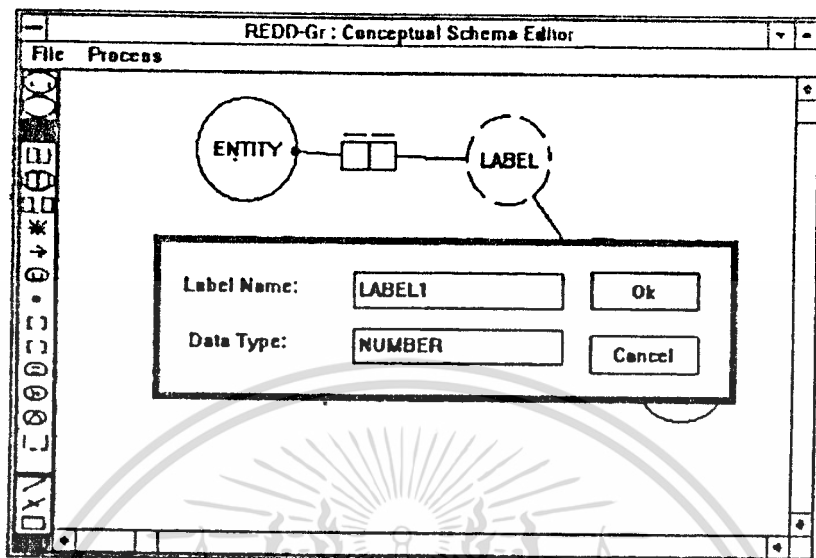
หมายเหตุ (1) เลื่อนเมมเบอร์ชิปไปวางยังตำแหน่งที่ต้องการ คลิกวางแล้วคลิกการกำกับ
ที่เอนติตี้ใหม่

3.12 วาดคอนสเตรนซ์แบบอ็ควอลิตี้ , เอ็กคลูชัน และ สับเซต - ใช้ร่วมกันทั้งสองแบบ โดยการคลิกที่เครื่องหมายคอนสเตรนซ์แต่ละชนิด วางรูปทรงกลม ณ ตำแหน่งที่ต้องการ เชื่อมโวลที่คอนสเตรนซ์กำกับ โดยการคลิกที่โวลที่ต้องการ 2 โวล

3.13 วาดคอนสเตรนซ์แบบอ็คเคอเรन्ซ์ หรือควินรี่ - โดยการคลิกที่เครื่องหมายสุดท้ายที่
ต่อจากเครื่องหมายคอนสเตรนซ์แบบสับเซต ใช้การกำกับในลักษณะเดียวกับการกำกับคอนส
เตรนซ์แบบยูนิคภายใน



รูปที่ 8.3 แสดงหน้าจอการสร้างเอนติตี้ใหม่



รูปที่ 8.4 แสดงหน้าจอการสร้างเลเบลไทย

การลบ

กระบวนการลบมีลักษณะ เช่นเดียวกับการวาด แตกต่างกันเพียง เริ่มต้นก่อนจะเริ่มลบอ็อบเจ็กต์ใดออก จะต้องคลิกไปที่เครื่องหมายสี่เหลี่ยมว่างที่แสดงโหมดการลบ คลิกอีกครั้งที่เครื่องหมายอ็อบเจ็กต์ที่เลือก แล้วนำเมาส์ไปคลิกที่อ็อบเจ็กต์ที่ต้องการ ขั้นตอนการลบอ็อบเจ็กต์จะต้องทำตามลำดับการลบ คือ หากเป็นคอนสเตรนทที่เกี่ยวข้องกับโรลกำกับอยู่บนโรล ต้องทำการลบคอนสเตรนทดังกล่าวออกก่อน หากเป็นคอนสเตรนทที่เกี่ยวข้องกับเอนิตีไทยกำกับอยู่บนเอนิตีไทย ต้องลบคอนสเตรนทดังกล่าวออกก่อน ถัดจากการลบคอนสเตรนทจะเป็นการลบแพทไทยหรือเรฟเฟอร์เรนซ์ไทยออก แล้วจึงตามด้วยการลบเอนิตีไทยหรือเลเบลไทยออก ตามลำดับ

ลบ => คอนสเตรนท -> แพทไทย (เรฟเฟอร์เรนซ์ไทย) -> เอนิตีไทย (เลเบลไทย)

: แสดงลำดับการลบอ็อบเจ็กต์ในโปรแกรมช่วยวาด

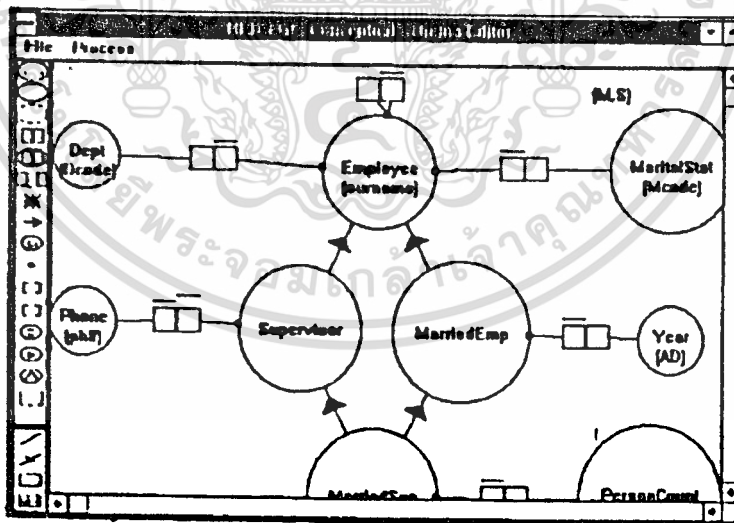
ขั้นตอนที่ 4 เก็บไฟล์ข้อมูล

เมื่อวาดรูปในแอมเสร็จแล้วและต้องการเก็บไฟล์ข้อมูลไว้ ให้เลื่อนเคอร์เซอร์ไปยังเมนู แล้วเลือกเมนู **Save as** ทำการใส่ชื่อไฟล์ข้อมูลแล้วคลิกที่เมนู ต่อไปเมื่อต้องการเรียกไฟล์ข้อมูลนั้นออกมา ก็ใช้เมนูการเรียกหรือโหลดไฟล์ ไฟล์ข้อมูลที่ทำาการเก็บประกอบด้วย 3 ไฟล์ คือ

1. ไฟล์เดสคริบเตอร์ เก็บลักษณะและความสัมพันธ์ของอ็อบเจ็คในรูปในแอม (รายละเอียดดูจากบทที่ 5 เรื่อง โมดูลช่วยวาด)
2. ไฟล์เมต้าไฟล์ เก็บคำสั่งการแสดงผลรูปภาพกราฟิกบนวินโดว์ เป็นไฟล์กราฟิกที่กรูปร่างแบบหนึ่งของไมโครซอฟต์วินโดว์
3. ไฟล์ลิงก์ เก็บค่าลิงก์หรือความสัมพันธ์ต่างๆ ของอ็อบเจ็คกราฟิก เป็นไฟล์ที่ใช้งานในกระบวนการภายในของโปรแกรมช่วยวาด ไม่มีความเกี่ยวข้องกับโครงสร้างของระบบงาน

ขั้นตอนที่ 5 แบบจำลองที่สำเร็จแล้ว

เมื่อทำการวาดแบบจำลองตามการวินิจฉัยดังกล่าวข้างต้นแล้ว จะ ได้รูปของแบบจำลองที่สำเร็จลง รูปที่ 8.5 แสดงรูปแบบจำลองที่สำเร็จแล้ว

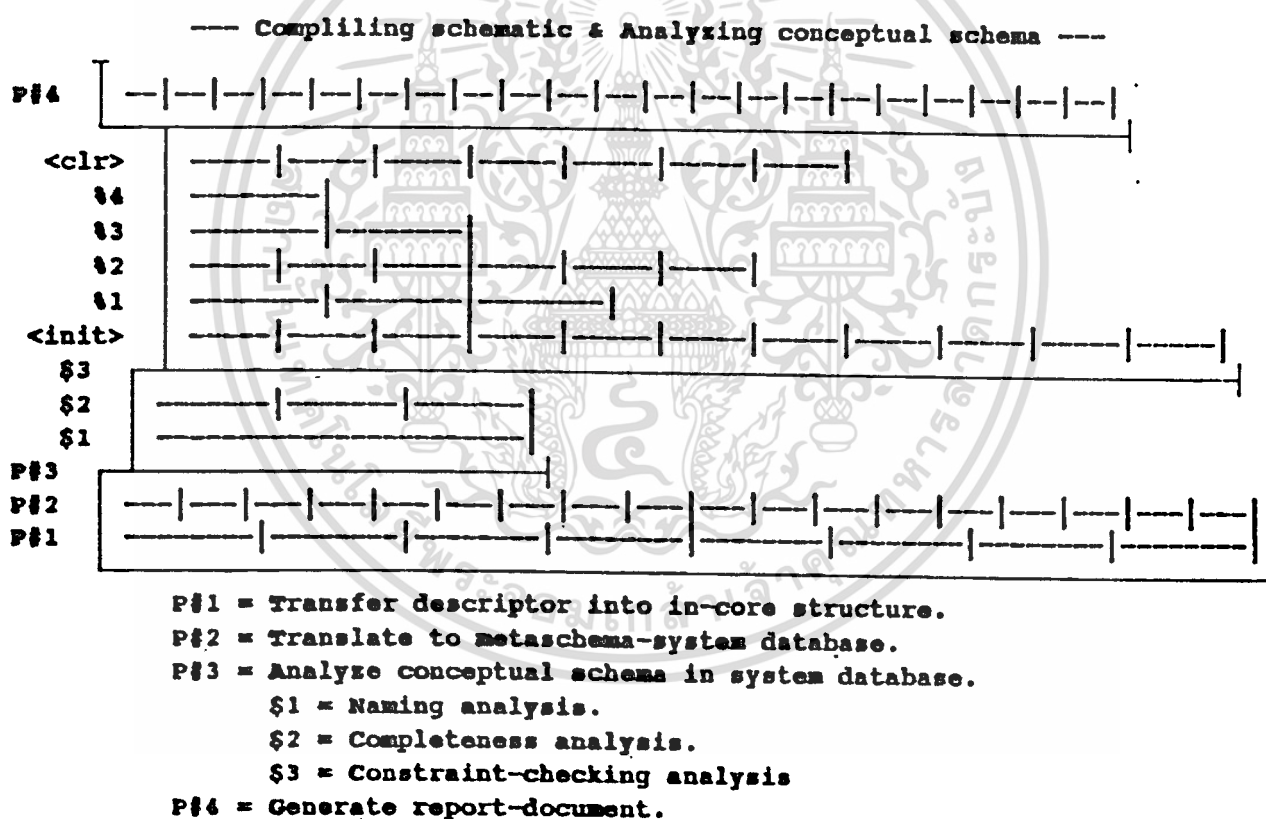


รูปที่ 8.5 แสดงหน้าจอของแบบจำลองที่สำเร็จลงแล้ว

ขั้นตอนที่ 6 เรียกโปรแกรมส่วนแปลงข้อมูลและส่วนวิเคราะห์

เมื่อทำการเก็บไฟล์เรียบร้อยแล้ว ขั้นตอนต่อไปคือการตรวจสอบแบบจำลองที่ได้ออกแบบเรียบร้อยแล้ว โดยการเรียกใช้โปรแกรมแปลงข้อมูลเป็นเมต้าเดต้าเบส แล้วตรวจสอบด้วยโปรแกรมวิเคราะห์ ผ่านทางเมนู Process เมนูย่อย Compile&Analyze โปรแกรมช่วยวาดจะทำการเรียกใช้โปรแกรมแปลงและวิเคราะห์ทันที โดยเรียกโปรแกรม DB.EXE ที่รวมเอา 2 โมดูล คือ โมดูลแปลงข้อมูล และ ตรวจสอบข้อมูล เข้าไว้ด้วยกัน

เมื่อเข้าสู่โปรแกรมแปลงและวิเคราะห์ หน้าจอจะแสดงผลระดับการวิเคราะห์ ทีละขั้นตอน ๆ จนกระทั่งจบกระบวนการ หน้าจอแสดงดังรูปที่ 8.6



รูปที่ 8.6 แสดงหน้าจอของโปรแกรมแปลงข้อมูลและวิเคราะห์ DB.EXE

เมื่อจบกระบวนการแล้ว โปรแกรมจะรายงานผลการวิเคราะห์ในไฟล์แจ้งผลการวิเคราะห์ที่ชื่อ E001.REP แล้ววีเทอร์นกลับมาถึงสภาพแวดล้อมไมโครซอฟต์วินโดวส์อีกครั้ง พร้อมกับแสดงไฟล์ผลการวิเคราะห์ทางวินโดวส์ลูก (child window) ของโปรแกรม ลักษณะไฟล์แจ้งผลการวิเคราะห์แสดงดังรูปที่ 8.7

***** PROCESS COMPILE & ANALYSIS "ERROR" *****

PASS 1 : Transfer conceptual schema into in-core structures
-> process complete...

PASS 2 : Translate conceptual schema into Metaschema
-> process complete...

PASS 3 : Analysis conceptual schema
-> process complete...

- (phase \$1) Naming analysis
-> No naming error objects.
- (phase \$2) Completeness analysis
-> No incomplete entity types!
-> No incomplete label types.
-> No incomplete roles.
- (phase \$3) Constraints analysis
 - > (sub-phase #1) Initial temp tables & views
-> Initial sub-phase complete...
 - > (sub-phase #2) Syntactic rules analysis
-> No abnormal syntax constraints
-> No loop deadlock subset-constraint...
 - > (sub-phase #3) Redundancy warning analysis
-> "MDA3" unnecessary !!
-> "MDA4" unnecessary !!
 - > (sub-phase #4) Condition warning analysis
-> Condition-complete between constraints...
 - > (sub-phase #5) Confliction analysis
-> "BIRD" conflict with own data type !!
-> "CAT" conflict with own data type !!
-> "DOG" conflict with own data type !!
-> "F" conflict with own data type !!
-> "M" conflict with own data type !!
 - > (sub-phase #6) Clear temp tables & views
-> Clearing complete...

PASS 4 : Flush schema in ROLLBACK PROCESS
-> Rollback complete...

รูปที่ 8.7 ตัวอย่างไฟล์แจ้งผลการวิเคราะห์ E001.REP

โครงการระบบช่วยออกแบบฐานข้อมูลแบบสัมพันธ์ หรือ เวดด์ เป็นโครงการที่จัดทำขึ้นในความพยายามที่จะสรุปเอาความสามารถและข้อดี ของนิยามข้อมูลระดับแนวความคิดหรือที่เรียกว่า คอนเซ็ปชวลสกีมา ไว้ในระบบคอมพิวเตอร์ เพื่อเหตุผลที่จะผลักดันแนวความคิดเกี่ยวกับการให้แบบจำลองระดับแนวความคิดของไมเอมให้เป็นต้นแบบหลาย ทั้งนี้เพื่อความก้าวหน้าของวิทยาการฐานข้อมูลในประเทศไทย การสร้างฐานข้อมูลแบบสัมพันธ์โดยอาศัยหลักการแปลงรูปแบบจำลองไมเอม ได้ช่วยให้ขั้นตอนเกี่ยวกับการออกแบบฐานข้อมูล ลดความยุ่งยากไปมากในขณะที่ฐานข้อมูลที่ได้มีประสิทธิภาพสูง คือมีความซับซ้อนของข้อมูลน้อยและง่ายต่อการเปลี่ยนแปลงแก้ไข

แต่ในการสร้างแบบจำลองข้อมูลแบบไมเอม หากกระทำโดยผู้ไม่มีความรู้ความชำนาญพอ อาจก่อให้เกิดความยุ่งยากในการออกแบบ ฐานข้อมูลที่ได้ไม่เหมาะสมกับระบบงาน และเกิดความผิดพลาดทางความหมาย วิชาหลักการ รวมถึงความถูกต้องตามลักษณะข้อมูล ปัญหาต่าง ๆ เหล่านี้ เป็นปัญหามิติพิเศษที่ค่อนข้างมากและบางปัญหาก็ซับซ้อนยากต่อการตัดสินใจ กระบวนการวิเคราะห์แบบจำลองข้อมูลตามกฎที่เสนอขึ้นมาั้น ไม่ครอบคลุมเท่าที่ควร จนอาจกล่าวได้ว่าเป็นการวิเคราะห์ที่ค่อนข้างหยาบ เมื่อเทียบกับความสลับซับซ้อนของลักษณะข้อมูลที่นักออกแบบต้องการ ทางผู้พัฒนาโครงการนี้ได้พยายามที่จะสรุปเอาการพิเศษที่ได้พบในขณะพัฒนาโครงการ และรวบรวมเอาไว้เพื่อเป็นแนวทางในการพัฒนาต่อ ของผู้พัฒนาระบบต่อไปในภายภาคหน้า

ในด้านการออกแบบอีก ส่วนที่มีความยุ่งยากมากที่สุดในการออกแบบคือ ส่วนของ ไมเอม-คอนเซ็ปชวลเมตาสกีมา ซึ่งกล่าวได้ว่าเป็นโครงสร้างข้อมูลร่วม ที่กลุ่มผู้พัฒนาโครงการระบบช่วยออกแบบฐานข้อมูลทั้งสองกลุ่มต้องใช้ร่วมกัน เมตาสกีมาที่แสดงไว้ในปริยายาณิธส์ฉบับนี้มีแนวทางมาจากหนังสือเรื่อง Conceptual Schema and Database Design A Fact Oriented Approach บทที่ 13 The conceptual metaschema ของ Prof.G.M. Nijssen และ T.A.Halpin ซึ่งเป็นหนังสืออ้างอิงหลักของโครงการ เมตาสกีมานี้ได้ถูกแก้ไขเปลี่ยนแปลงเพื่อความถูกต้องอย่างจริงจังไม่ถวน เพื่อให้ได้เมตาสกีมาที่สมบูรณ์จริง ๆ เท่านั้น การออกแบบเฉพาะในส่วนนี้จึงกินเวลาในภาคการศึกษาที่ 1 เกือบทั้งหมด เป็นผลให้ใช้เวลาในการพัฒนาโปรแกรมมีผลลงมา

นอกจากคอนกรีตมวลเบาแล้ว กลุ่มผู้พัฒนาโครงการยังต้องออกแบบภาษากลางที่จะใช้ในการสื่อสารในแต่ละส่วนของโมดูลอีก เช่น แม่บ็องอินฟอร์เมชั่นที่ใช้ในการเชื่อมต่อระหว่างส่วนออกแบบ และส่วนโครงสร้าง เดสคริปเตอร์ที่ใช้เชื่อมต่อระหว่างโปรแกรมช่วยวาดบนสภาพแวดล้อมแบบกราฟฟิก กับโปรแกรมแปลงรูปแบบข้อมูล รวมถึงการออกแบบโครงสร้างข้อมูลภายใน หรือ in-core structure ที่ใช้เป็นโครงสร้างชั่วคราวก่อนนำไปสร้างเมตาเดต้าเบส แต่ละส่วนล้วนแล้วแต่ต้องอาศัยเวลาในการออกแบบทั้งสิ้น

เมื่อมาถึงขั้นตอนในการพัฒนาระบบงาน ทางผู้พัฒนาโครงการประสบปัญหาต่าง ๆ มากมาย ดังจะสรุปเป็นข้อ ๆ ดังนี้

1. ความเข้ากันไม่ได้ของระบบที่เลือกใช้ในโครงการ เช่น ความเข้ากันไม่ได้ระหว่างระบบจัดการฐานข้อมูลอราเคิล กับ ระบบเชื่อมประสานผู้ใช้แบบกราฟิกไมโครซอฟต์วินโดวส์ ดังได้กล่าวไว้แล้วในบทที่ 5 เรื่องโมดูลช่วยวาด

2. ความไม่เหมาะสมของระบบจัดการฐานข้อมูลที่ใช้ เนื่องจากระบบจัดการฐานข้อมูลอราเคิลเป็นระบบจัดการที่มีความปลอดภัยสูง หากแต่กระบวนการติดต่อกับข้อมูลในฐานข้อมูลค่อนข้างล่าช้า จึงเกิดปัญหาประสิทธิภาพของระบบในด้านความเร็ว ซึ่งในอนาคตเมื่อความสามารถของฮาร์ดแวร์มีมากขึ้น ปัญหาดังกล่าวนี้จะลดลง

3. ความไม่สมบูรณ์ทางเอกสารของไมโครซอฟต์วินโดวส์ ถึงแม้ว่าไมโครซอฟต์วินโดวส์จะเป็นระบบที่แพร่หลาย และค่อนข้างยืดหยุ่นในการนำไปพัฒนาโปรแกรม แต่เอกสารเกี่ยวกับเครื่องมือที่ใช้ในการพัฒนาโปรแกรมไม่เพียงพอ ทำให้ยากต่อการพัฒนาโปรแกรม

4. ไม่สามารถนำโมดูลที่แยกพัฒนามารวมกันให้เกิดประสิทธิภาพตามที่ต้องการ เนื่องจากการพัฒนาระบบงาน ใช้วิธีการแบ่งส่วนการรับผิดชอบ โดยกำหนดภาษากลางที่จะใช้ในการสื่อสาร โดยตั้งความหวังว่าในขั้นตอนสุดท้ายจะนำมารวมกันเป็นระบบงานใหญ่ แต่ในทางปฏิบัติไม่สามารถกระทำได้นี้เนื่องจากข้อจำกัดทางด้านเวลา

ในบทนี้จะได้กล่าวถึง สิ่งที่ได้มีการกำหนดในเบื้องต้นว่าจะทำ แต่เนื่องจากข้อจำกัดทางเวลา ทำให้ไม่สามารถทำได้ตามความประสงค์ จึงบรรจุสิ่งเหล่านี้ไว้เป็นแนวทางในการพัฒนาต่อสำหรับนักพัฒนาโครงการต่อไป ได้แก่

1. รวมเอาโครงงานประเภททั่วกลไกแบบ ๆ ในส่วนกลไกแบบ และ ส่วนโครงสร้าง ให้กลุ่มในเกณฑ์รวมแบบการทบทวนเดียวกัน โดยเรียกให้ผ่านเมนูของระบบรวม

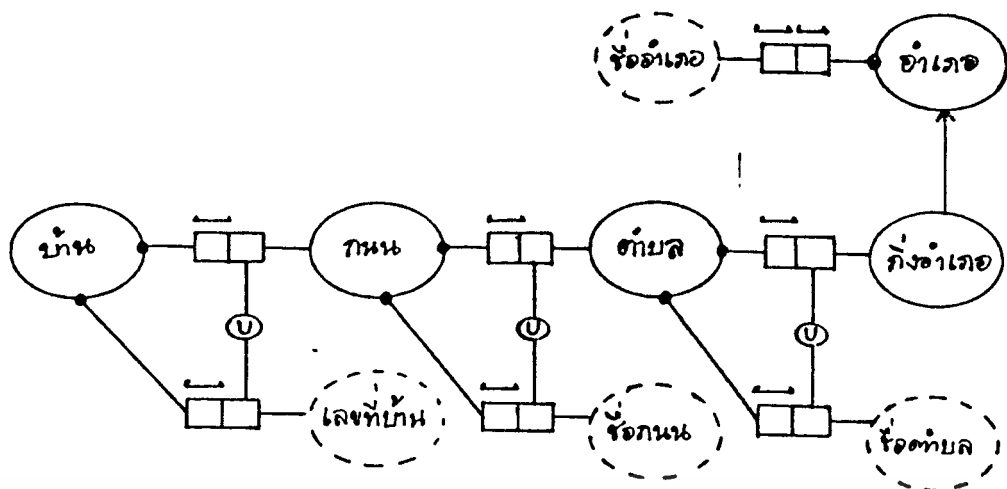
เนื่องจากประเภททั่วกลไกแบบฐานข้อมูลแบบสัมพันธ์ เป็นระบบงานรวมซึ่งต้องดำรงสภาพแวดล้อมแบบการทบทวน การเรียกใช้โมดูลการแปลงนิยามข้อมูล โมดูลวิเคราะห์แบบจำลอง และ โมดูลสร้างฐานข้อมูลของกลุ่มโครงสร้าง จะกระทำผ่านเมนูเรียกใช้ในสภาพแวดล้อมของระบบรวม และการแสดงผลของโมดูลเหล่านี้ ควรจะอยู่ในรูปแบบการทบทวนด้วยเพื่อความเป็นอันหนึ่งอันเดียวกัน

2. ให้สามารถรองรับความถูกต้อง (Integrity constraint) ของฐานข้อมูลกับได้แก่ กฎความถูกต้องของเอนทิตี Entity Integrity และกฎความถูกต้องในการอ้างอิง (Referencial integrity) ได้

กฎความถูกต้องถือเป็นเงื่อนไขหลักข้อหนึ่ง ที่ใช้ในการตัดสินใจว่าระบบจัดการฐานข้อมูลนั้นเป็นระบบแบบสัมพันธ์หรือไม่ การกำหนดไพรมารีคีย์และรีเลชันที่ได้จาก โมดูลการสร้างรีเลชันต้องเป็นไปตามกฎความถูกต้องของเอนทิตี และเมื่อมีการเปลี่ยนแปลงแก้ไขค่าข้อมูลที่เป็นฟอร์เรคีย์ ระบบจะต้องดูแลความถูกต้องให้กับไพรมารีคีย์ของรีเลชันที่เกี่ยวข้อง ตามกฎความถูกต้องในการอ้างอิงด้วย

3. ให้สามารถรองรับและสามารถแมทชิ่งในแอนทิตี เอนทิตี ที่ไม่มีการอ้างอิงหรือเชื่อมโยงแบบไฮราฟที่สัมพันธ์กันได้

ในการอ้างอิงของเอนทิตีไฮราฟที่นั้น ได้มีตัวอย่างแล้วในรูปที่ 6.5 แต่ในกรณีที่จะกล่าวถึงต่อไปนี้ เป็นกรณีพิเศษของการอ้างอิงแบบไฮราฟ ซึ่งเรียกว่าไฮราฟที่สัมพันธ์แบบแสดงในรูปที่ 10.1



รูปที่ 10.1 การอ้างอิงแบบไฮราคิแบบมีสับไพ์

4. สามารถตรวจสอบหรือวิเคราะห์ความถูกต้องเชิงความหมายหรือ semantic analysis ของแบบจำลองข้อมูลในแผนของระบบใช้งานงานข้อมูลได้ โดยวิธีตรวจสอบโดยทำ
ทดสอบประชากร (Population checking)

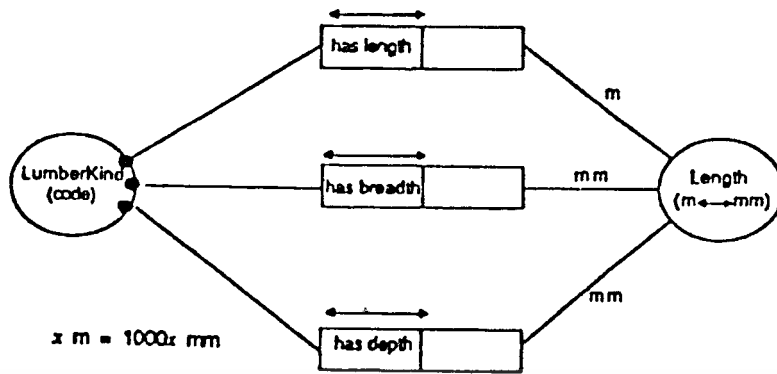
เป็นวิธีตรวจสอบความถูกต้องเชิงความหมายที่ค่อนข้างง่าย โดยจะทำการตรวจสอบความสัมพันธ์ที่เป็นแฟกต์ไทม์แบบเทอร์นารีขึ้นไป โดยการสุ่มประชากรตามคอนสเตรนที่ที่กำหนดอยู่ แล้วทำการแยกเทอร์นารีได้ออกมา เป็น 2 รีเลชัน หากผลลัพธ์ที่ได้จากการรวม (join) ระหว่าง 2 รีเลชันดังกล่าวเหมือนกับรีเลชันต้นแบบ ก็ถือว่าไม่สามารถแยก (split) รีเลชันที่เป็นเทอร์นารีได้ออกได้ แต่หากไม่ใช่ จะสามารถแยกออกได้เป็นไมนารี 2 รีเลชัน

5. ให้สามารถรองรับโมเดลที่มีแฟกต์ไทม์ชนิดหาได้ (Derived fact type)

ในขณะนี้ส่วนเมตาสกีมมาของโครงการได้สร้างชั้นรองรับแฟกต์ไทม์ชนิดหาได้อยู่แล้ว เหลือแต่เพียงการพัฒนากระบวนการ ที่ใช้ในการประมวลผลสูตรที่ใช้ในการหาหรือ Derived formula เพื่อให้ได้ค่าข้อมูลตามที่ต้องการ สูตรการหาเป็นได้ตั้งแต่ สูตรคำนวณธรรมดา จนถึงภาษา SQL ที่ใช้ในการสร้างวิวลักษณะต่าง ๆ ตามที่ตัวโปรแกรมจะกำหนด

6. ให้สามารถรองรับโมเดลที่มีเรเฟอเรนซ์ไทม์ชนิดหาได้ (Derived reference type)

ลักษณะของเรเฟอเรนซ์ไทม์ชนิดหาได้ แสดงดังรูปที่ 10.2



รูปที่ 10.2 เรฟเฟอเรนซ์ไทป์แบบหาได้

จากรูปแบบจำลอง LumberKind มีความสัมพันธ์แบบพหุต่อ 3 ความสัมพันธ์ โดยกระทำกับ Length ซึ่งมีการอ้างอิงด้วยหน่วยความยาว 2 หน่วย คือ เมตรกับมิลลิเมตร ทั้งนี้พหุต่อไทป์จะต้องกำหนดว่าจะให้การอ้างอิงในหน่วยใด กรณีเช่นนี้จะเห็นได้ว่าเรฟเฟอเรนซ์ไทป์ที่แฝงอยู่ในเอนทิตีไทป์ Length มีอยู่เพียงหนึ่งเดียวคือ มิลลิเมตร โดยที่จะสามารถหาการอ้างอิงในหน่วยเมตร ได้จากการคำนวณสูตร เมตร = 1000 * มิลลิเมตร เรียกเรฟเฟอเรนซ์ไทป์ลักษณะนี้ว่า เรฟเฟอเรนซ์ไทป์แบบหาได้ (Derived reference type)

7. ให้สามารถรองรับเลเบลไทป์ที่มีชนิดของข้อมูลได้มากกว่าที่เป็นอยู่

โครงสร้างนี้ครอบคลุมเพียงแค่นิดข้อมูล 2 ประเภทคือ สตริง (character) และ จำนวน (number) ซึ่งในทางปฏิบัติควรมีมากกว่าที่เป็นอยู่

8. แก้ไขส่วนโปรแกรมทั่วๆไปให้มีความสวยงามและสมบูรณ์มากขึ้น

โปรแกรมส่วนโมดูลทั่วๆไปหรือ REDD-Gr ที่ได้พัฒนาขึ้นนี้ ยังมีการติดต่อกับผู้ใช้อย่างไม่เป็นธรรมชาติอีกทั้งยังมีปัญหาะหว่างการทำงานบางประการ จึงควรจะมีการแก้ไขให้มีความสมบูรณ์มากยิ่งขึ้น

9. เพิ่มกฎในการวิเคราะห์แบบจำลองในคอมของโปรแกรมวิเคราะห์ให้มากขึ้น

กฎที่ใช้ในการวิเคราะห์ทั้งมีอยู่ไม่มากเพียงพอ จึงควรเพิ่มเติมแก้ไขกฎให้มีความสมบูรณ์มากขึ้น

เอกสารอ้างอิง

1. Prof.G.M.Nijssen , T.A.Halpin Conceptual Schema and Relational Database Design A Fact Oriented Approach Prentice Hall Pty Ltd.1989
2. C.J.Date An Introduction to Database Systems
Addison-Wesley Publishing Company. 1986



ภาคผนวก ก โปรสตาวิช (PRO*VC)

ระบบจัดการฐานข้อมูลแบบสัมพันธ์ของออราเคิล (ORACLE Relational Database Management System) ได้ทำการสร้างโปรแกรมที่ใช้ในการเปลี่ยนประโยคคำสั่งของเอสคิวแอลที่ประกอบอยู่ในโปรแกรมภาษาซี ซึ่งจะนำหน้าด้วย "EXEC SQL" ให้เป็นประโยคคำสั่งในภาษาซี ซึ่งสามารถจัดการฐานข้อมูลของออราเคิลได้ และเรียกวิธีนี้ว่า "การพรีคอมไพล์" (Precompiler) และจากนั้นจึงนำผลที่ได้ มาทำการคอมไพล์หรือลิงก์ โดยใช้คอมไพเลอร์ของภาษาสูงที่ให้ เพื่อให้ได้โปรแกรมที่สามารถปฏิบัติการได้ ซึ่งโปรแกรมที่ใช้ในการพรีคอมไพล์นี้คือ "โปรสตาวิชทูล" (Pro*c Tool)

ดังนั้น เราสามารถสรุปขั้นตอนในการเขียนโปรแกรมโปรสตาวิช ได้ดังนี้

1. เขียนโปรแกรมโปรสตาวิช
2. ทำการพรีคอมไพล์ โดยใช้โปรสตาวิชทูล
3. ทำการคอมไพล์แฟ้มข้อมูลที่ได้จากการคอมไพล์
4. ทำการลิงค์แฟ้มข้อมูลที่เกี่ยวข้องกัน
5. รันโปรแกรม

1. ประโยคคำสั่งของเอสคิวแอลในการปฏิบัติการและการกำหนด (Execute and Declarative SQL Statements)

ประโยคคำสั่งของเอสคิวแอลที่ใช้ในการโปรแกรมโปรสตาวิช สามารถแบ่งได้ 2 ประเภทคือ

1. ประโยคคำสั่งปฏิบัติการ คือ คำสั่งที่เกี่ยวข้องกับการจัดการข้อมูล (Data Manipulation Language) , การกำหนดข้อมูล (Data Definition Language) และการควบคุมข้อมูล (Data Control Language) ซึ่งการปฏิบัติการของประโยค คำสั่งเหล่านี้จะมีผลต่อส่วนการติดต่อข้อมูลของเอสคิวแอล (SQL Communication Area) หรือเรียกอีกอย่างว่าเอสคิวแอลซีเอ (SQLCA) โดยจะมีการแสดงรหัสสถานะที่ได้จากผลของประโยคคำสั่งการปฏิบัติการที่ส่วนนี้ นอกจากนี้ ยังมีผลต่อการเริ่มต้น หรือการสิ้นสุดของส่วนการเก็บการเรียงลำดับของประโยคคำสั่งเอสคิวแอล (Lexical Unit of Work)

2. ประโยคคำสั่งการกำหนด คือ ประโยคคำสั่งที่จะไม่มีการแสดงรหัสสถานะของการปฏิบัติที่ส่วนการติดต่อข้อมูลของเอสคิวแอล และส่วนการเก็บการเรียงลำดับของประโยคคำสั่งของเอ

2. ส่วนประกอบในโปรแกรมโปรสตาร์ซี (Parts of the Pro*C Program)

โปรแกรมโปรสตาร์ซี สามารถแบ่งได้เป็น 2 ส่วน คือ

1. ส่วนโครงสร้างการประยุกต์ (Application Prologue) คือส่วนกำหนดตัวแปรที่ใช้ในการติดต่อกับฐานข้อมูล , การเตรียมพร้อม และการเริ่มต้นเพื่อที่จะติดต่อกับฐานข้อมูล

2. ส่วนตัวโปรแกรมประยุกต์ (Application body) คือ ส่วนของโปรแกรมที่ประโยคคำสั่งต่างๆของเอสควแอล ในการจัดการฐานข้อมูลปรากฏอยู่

3 ส่วนโครงสร้างการประยุกต์

สามารถแบ่งออกได้เป็น 3 ส่วน คือ

1. ส่วนกำหนดตัวแปร (DECLARE section) คือ ส่วนที่ใช้กำหนดตัวแปรและชนิดในภาษาซีที่จะใช้ในการติดต่อกับฐานข้อมูล มีรูปแบบในการใช้โดยจะเริ่มต้นด้วยประโยค

EXEC SQL BEGIN DECLARE SECTION;

และจบด้วยประโยค

EXEC SQL END DECLARE SECTION;

ระหว่างประโยคทั้ง 2 จะเป็นการกำหนดตัวแปรและชนิดของตัวแปร ซึ่งตัวแปรที่กำหนดในส่วนนี้สามารถแบ่งได้เป็น ตัวแปรหลัก (Host Variable) และตัวแปรบ่งชี้ (Indicator Variable) ตัวแปรหลักเป็นตัวแปรที่เก็บค่าข้อมูลต่างๆที่ใช้ในการติดต่อกับฐานข้อมูล ส่วนตัวแปรบ่งชี้จะใช้ในการเพิ่มข้อมูลในคอลัมน์ที่ไม่ทราบค่า (NULL) และสามารถแสดงให้ทราบว่าข้อมูลที่ตัวแปรหลักได้รับจากฐานข้อมูลมีลักษณะอย่างไร เช่น ไม่ทราบค่า หรือ มีการตัดข้อมูลส่วนทั้ง เป็นต้น

สำหรับชนิดของตัวแปรหลักนั้น จะเหมือนกับชนิดของตัวแปรในภาษาซี แต่ที่เพิ่มเติมมาคือประเภท วาร์ชาร์ (VARCHAR) ซึ่งมีลักษณะเป็นแบบเรคอร์ด (Record)

สำหรับข้อผิดพลาด (error) ที่เกิดขึ้นในส่วนการกำหนดตัวแปรนี้ ในกรณีที่ผู้ใช้ไม่ได้กำหนดตัวแปรที่ใช้ในโปรแกรม แต่มีการเรียกใช้ จะมีการแสดงข้อความผิดพลาด คือ

Undeclared host variable <a> at line in file <c>

2. INCLUDE SQLCA จะมีรูปแบบการใช้คือ

EXEC SQL INCLUDE SQLCA;

ประโยคคำสั่งนี้ จะเป็นการกำหนดให้โปรสตาวิชี่ รวมส่วนของการติดต่อข้อมูลกับเอสคิวแอลเข้าไปในโปรแกรม โดยขณะที่ทำการคอมไพล์ ออราเคิลจะทำหน้าที่เปลี่ยนหรือแทนที่ตัวแปรหลักในโปรแกรมด้วยตัวแปรที่ได้จากแฟ้มข้อมูลที่น่ามารวม และหน้าที่สำคัญของการติดต่อข้อมูลกับเอสคิวแอล อีกอย่างหนึ่ง นอกเหนือจากการติดต่อกับออราเคิล ก็คือ การแสดงข้อผิดพลาดและข้อระวังต่างๆ (Warning) ที่เกิดขึ้นในการปฏิบัติคำสั่งของเอสคิวแอลโดยจะแสดงในรูปแบบ

- sqlca.sqlcode : ค่ามากกว่า 0 จะแสดงถึงการกระทำคำสั่ง

ค่าเท่ากับ 0 แสดงว่าทำคำสั่งได้สมบูรณ์

ค่าน้อยกว่า 0 เกิดการผิดพลาดขึ้น

- sqlca.sqlwarn : จะประกอบด้วยอาร์เรย์ (array) ของแฟล็กส์ (flags) 8 ตัว ซึ่งแต่ละตัวก็จะแสดงถึงลักษณะของข้อระวังที่แตกต่างกันออกไป

นอกจากจะสามารถเรียกใช้เอสคิวแอลได้แล้ว ยังสามารถใช้คำสั่งเฉพาะหรือติดต่อกับออราเคิลได้โดยตรง โดยใช้คำสั่ง

EXEC SQL INCLUDE ORACA;

3. การติดต่อกับออราเคิล (Connecting to ORACLE) มีรูปแบบการใช้คือ

EXEC SQL CONNECT (:oracleid) IDENTIFIED BY (:oraclepasswd)

หรือ

EXEC SQL CONNECT (:oracleid)

โดยที่ oracleid อยู่ใน (:oracleid)/(:oraclepasswd) จะเป็นส่วนที่ต้องใช้เพื่อให้โปรแกรมสามารถติดต่อกับระบบจัดการฐานข้อมูลแบบสัมพันธ์ของออราเคิลได้ ซึ่งจะทำให้สามารถเข้าถึงฐานข้อมูลของออราเคิลได้

4. ตัวโปรแกรม (Application Body)

เป็นส่วนที่ภาษาหลัก (Host) และภาษา Embedded รวมกันอยู่ ลักษณะโดยทั่วไปของโปรแกรม คือ

- ภาษาหลัก จะเป็นตัวจัดการเกี่ยวกับการแสดงผล (display) และรูปแบบการใช้งานต่างๆของโปรแกรม เช่น เมนู เป็นต้น

- ภาษา Embedded จะทำงานในด้านการจัดการเกี่ยวกับข้อมูล รวมทั้งการเรียกใช้คำสั่งของเอสคิวแอลและออราเคิลด้วย ซึ่งการเรียกใช้นั้น จะต้องใช้ "EXEC SQL" นำหน้าก่อนเสมอ

5. การถามตอบกับเควรี่ (Query)

เป็นส่วนหนึ่งในตัวโปรแกรม (Application Body) ซึ่งจะใช้ในการเรียกข้อมูลมาใช้ หรือ เก็บข้อมูลต่างๆ

สำหรับคำสั่งที่ใช้ในการสอบถาม (Query) จะประกอบด้วย

- | | |
|-------------|----------|
| - SELECT | - INTO |
| - FROM | - WHERE |
| - CONNECT | - UNION |
| - INTERSECT | - MINUS |
| - GROUP BY | - HAVING |
| - ORDER BY | ! |

สำหรับตัวแปรที่ใช้ในการสอบถามนั้น มาจาก 2 ที่ คือ จากตาราง (table) ในภาษาเควรี่ และจากตัวแปรในภาษาหลัก ซึ่งตัวแปรในภาษาหลักที่ใช้ในการสอบถามจะต้องที่เครื่องหมาย ":" (colon) นำหน้าชื่อตัวแปรเสมอ

ลักษณะการสอบถาม มี 2 แบบ คือ

1. การสอบถามที่ให้ผลลัพธ์ออกมาเพียง 1 แถว (Query which return SINGLE ROW only) เป็นการสอบถามที่จะต้องอ้างอิงกับค่าที่มีเพียง 1 แถวในตารางเท่านั้น (Unique index) ซึ่งถ้าให้ค่ามากกว่า 1 แถวจะแสดงข้อผิดพลาดออกมา

2. การสอบถามที่ให้ผลลัพธ์มากกว่า 1 แถว (Query which return MULTIPLE ROWS) การสอบถามลักษณะนี้มักจะใช้กับการเรียกข้อมูลที่มีเป็นกลุ่มในตาราง ซึ่งเมื่อกระทำการสอบถามแล้ว เอพริคแอลจะให้ผลลัพธ์ ทั้งหมดออกมาในครั้งเดียว ดังนั้น การใช้การสอบถามแบบนี้จึงจำเป็นต้องเตรียมเนื้อที่ส่วนหนึ่งในออร์ราเคิลหรือเอสคิวแอล เพื่อที่จะใช้ในการเก็บผลลัพธ์นั้นไว้ แล้วจึงเรียกออกมาใช้ตามที่ต้องการ ซึ่งขั้นตอนนี้เรียกว่า เคอร์เซอร์ (Cursor)

เคอร์เซอร์มีลักษณะการใช้ ดังนี้คือ

- การประกาศเคอร์เซอร์ (DECLARE CURSOR) เพื่อกำหนดพื้นที่ , ชื่อ และการสอบถามที่ต้องการ

รูปแบบ : EXEC SQL DECLARE <cursorname> CURSOR FOR [Query];

- การเปิดเคอร์เซอร์ (OPEN CURSOR) เพื่อเปิดให้สามารถเรียกใช้เคอร์เซอร์ได้

รูปแบบ : EXEC SQL OPEN <cursorname>;

- การเฟตช์ (FETCH) ให้เคอร์เซอร์แสดงผลลัพธ์ตัวต่อไป

รูปแบบ : EXEC SQL FETCH <cursorname> INTO <HostVar>;

- การปิดเคอร์เซอร์ (CLOSE CURSOR) เป็นการยกเลิกเคอร์เซอร์ที่ระบุออกไป

รูปแบบ : EXEC SQL CLOSE <cursorname>;

- ตำแหน่งของเคอร์เซอร์ปัจจุบัน (CURRENT OF CURSOR) ใช้ในการอ้างอิงตำแหน่งแถวปัจจุบันในเคอร์เซอร์ที่มีการเฟตช์ข้อมูล เพื่อนำมาใช้ในการแก้ไขหรือลบข้อมูล ตำแหน่งของเคอร์เซอร์ปัจจุบันสามารถใช้แทนด้วย ไรว์ไอดี (ROWID)

6. คอมมิต และ โรลแบค (Commit and Rollback)

ในการทำงานของโปรแกรม คำสั่งที่เป็นภาษาเอสคิวแอลแต่ละคำสั่ง จะถูกออราเคิลมองเป็นส่วนย่อย(logical unit of work) ซึ่งในแต่ละส่วนนี้ จะถูกประมวลผลเป็นลำดับกันไปจนจบ หรืออาจมีการถูกยกเลิกกลางคันก็ได้ สำหรับการยกเลิก unit of work นั้นเกิดได้จาก 2 กรณี คือ

1. ผู้ใช้ (user) ยกเลิกเอง
2. ระบบ (system) ไม่สามารถทำงานต่อไปได้ เช่น เกิดเดดล็อก (Deadlock) ขึ้น และการจบของ unit of work มี 2 แบบ คือ

- การคอมมิตเวิร์ก (Commit work) เป็นการจบ unit of work โดยให้ผ่านการเก็บการเปลี่ยนแปลงทั้งหมดที่เกิดขึ้นไว้ในฐานข้อมูล มีรูปแบบการใช้คือ

EXEC SQL COMMIT WORK [RELEASE];

โดยที่ทางเลือก RELEASE จะเป็นการคืนเนื้อที่ในหน่วยความจำทั้งหมดและออกจากระบบ (log off) ซึ่งจะเป็นการจบ unit of work สุดท้าย

- โรลแบคเวิร์ก (Rollback Work) เป็นการจบ unit of work เช่นกัน แต่จะทำการยกเลิกการแก้ไขข้อมูลทั้งหมด จะใช้ในกรณีที่เกิดการผิดพลาด ในการทำงานของโปรแกรม มีรูปแบบการใช้ลักษณะเดียวกับคอมมิต คือ

EXEC SQL ROLLBACK WORK [RELEASE]

7. การแสดงความผิดพลาด (Error and Warning)

หน้าที่สำคัญอีกประการหนึ่งของส่วนติดต่อข้อมูลของเอสคิวแอล คือ เป็นส่วนที่จะกระทำเกี่ยวกับการแสดงความผิดพลาดของโปรแกรม หรือในส่วนต่างๆของเอสคิวแอล โดยที่ลักษณะของ

ส่วนการติดต่อข้อมูลของเอสคิวแอล

8. การตรวจข้อผิดพลาด (Error Detection)

ในการตรวจสอบข้อผิดพลาด จะใช้คำสั่ง WHENEVER ซึ่งจะทำให้การตรวจสอบสถานะของส่วนการติดต่อข้อมูลของเอสคิวแอลทุกครั้งที่ทำคำสั่งเอสคิวแอล มีรูปแบบการใช้ ดังนี้

```
EXEC SQL WHENEVER [SQLERROR] [STOP]
[SQLWARNING] [CONTINUE]
[NOT FOUND] [GOTO label]
```

โดยที่

- SQLERROR : จะถูก set เมื่อ sqlca.sqlcode เป็นลบ
- SQLWARNING : จะถูก set เมื่อ sqlca.sqlwarn[0] = "พ"
- NOT FOUND : จะถูก set เมื่อ sqlca.sqlcode = +1403 (no row found)
- STOP : หยุดการทำงานของโปรแกรม และ ไรลนบก
- CONTINUE : ทำงานต่อไป ไม่ว่า sqlca จะเป็นอย่างไร
- GOTO label : ข้ามไปทำงานที่ label

ภาคผนวก ข. ไมโครซอฟท์วินโดวส์

ไมโครซอฟท์วินโดวส์ เป็นผลิตภัณฑ์ของบริษัทไมโครซอฟท์ซึ่งได้ออกมาถึง เวอร์ชัน 3.1 ซึ่งโดยทั่วไปเมื่อกล่าวถึงคำว่า วินโดวส์ ก็จะหมายถึงตัว ไมโครซอฟท์ วินโดวส์ ซึ่งเนื้อหาต่อไปในรายงานก็จะใช้คำว่า วินโดวส์ เพื่อสื่อความหมายถึงตัว ไมโครซอฟท์วินโดวส์ เวอร์ชัน 3.0 ซึ่งเป็นเวอร์ชันล่าสุดในขณะนี้ที่จะผู้จัดทำเริ่มโครงการนี้

วินโดวส์ 3.0 นั้นมีรูปร่างหน้าตาแทบจะเหมือนกับ PM (Presentation Manager) ของ OS/2 (ทั้ง วินโดวส์ และ PM ต่างก็เป็นระบบเชื่อมต่อผู้ใช้แบบกราฟิก - GUI แต่ทำงานบนระบบปฏิบัติการที่ต่างกันคือวินโดวส์รันบน ดอสแต่ PM รันบน OS/2) ซึ่งจากความสามารถที่ได้ถูกเพิ่มขึ้นมาจากเวอร์ชันเดิมทำให้สามารถใช้ได้กับฮาร์ดแวร์ได้กว้างขวางที่สุด ไม่ว่าจะเป็น จอภาพ, คีย์บอร์ด, เมาส์, พริ้นเตอร์, ฯลฯ นอกจากนั้นยังสนับสนุนระบบเน็ตเวิร์คได้มากมาย อาทิเช่น MS-Net, LAN Manager, NetWare และ Vines และในด้านการใช้งานนั้นผู้ใช้ (User) ก็สามารถใช้งานได้สะดวกกว่าการทำงานบนเท็กซ์โหมด ในด้านสื่อความหมายแม้ว่าจะมีการทำงานที่ช้าเมื่อเทียบกับการควบคุมโดยตรงที่ command line ของดอส แต่สิ่งที่สำคัญที่สุดของ วินโดวส์ ได้แก่ "การจัดการกับหน่วยความจำ" ซึ่งสามารถทำงานได้ทั้งโหมดการทำงานแบบเรียล แบบสแตนด์ออล และแบบเฮนเฮนซ์ 386 ซึ่งจะกล่าวต่อไปภายหลัง

ลักษณะเด่นของ วินโดวส์ 3.0

ลักษณะเด่น ๆ ของ ไมโครซอฟท์วินโดวส์ 3.0 พอจะแบ่งเป็นหัวข้อดังนี้

1. ง่ายต่อการเรียนรู้และใช้งาน เพราะลักษณะของโปรแกรมจะเป็นทูลดาวน์เมนู ผู้ใช้ไม่จำเป็นต้องจำคำสั่ง เพียงแต่ใช้แป้นพิมพ์หรือเมาส์ เรียกเมนูมาดู และลักษณะของเมนูแต่ละโปรแกรมจะมีลักษณะคล้ายคลึงกันเป็นมาตรฐานเดียวกัน ตามสถาปัตยกรรมที่ IBM กำหนดคือ SAA (Systems Application Architecture) ดังนั้น หากผู้ใช้คุ้นเคยกับโปรแกรมใดโปรแกรมหนึ่งแล้วก็สามารถไปใช้โปรแกรมอื่นได้อย่างง่ายดาย ซึ่งลักษณะการใช้งานเช่นนี้ก็สามารถไปใช้กับ OS/2 ของ IBM ได้เช่นกัน ดังนั้นผู้ที่จะใช้โปรแกรมภายใต้วินโดวส์จึงควรศึกษาโปรแกรมไมโครซอฟท์วินโดวส์เบื้องต้นก่อน เพราะโปรแกรมต่าง ๆ จะถูกควบคุมด้วยวินโดวส์ทั้งสิ้น
2. การติดตั้งวินโดวส์ 3.0 จะง่ายขึ้นเพราะโปรแกรมจะทราบว่าเครื่องที่ใช้เป็นประเภทไหนและจะทำการปรับและกำหนดค่าต่าง ๆ ให้การใช้งานรวดเร็วและมีประสิทธิภาพสูงที่สุดในขณะ

ที่จัดทำรายงานนี้ทางบริษัทไมโครซอฟท์ ได้พัฒนาไปถึง เวอร์ชัน 3.1 แล้ว แต่ยังไม่ออกวางตลาด ฉะนั้น เรื่องราวเกี่ยวกับ วินโดว์ ที่กล่าวถึงในรายงานจะหมายถึง เวอร์ชัน 3.0

3. สามารถใช้โปรแกรมได้ครั้งละหลาย ๆ โปรแกรมพร้อมกัน (Multitasking) รวมทั้งยังสามารถเลือกใช้งานจากโปรแกรมหนึ่งไปยังอีกโปรแกรมหนึ่งได้อย่างรวดเร็ว

4. สามารถใช้หน่วยความจำได้ตั้งแต่ 640 กิโลไบต์ไปจนถึง 16 เมกะไบต์ ของหน่วยความจำแบบคอนเวนชันนัล , เอ็กเซเทนดหรือกระทั่งเอ็กซเทนด (EMS-Expanded Memory Specification-4.0)

5. โอนย้ายข้อมูลทั้งที่เป็นตัวอักษร หรือรูปภาพจากโปรแกรมหนึ่งไปยังอีกโปรแกรมหนึ่งได้อย่างง่ายดาย

6. ตัวอักษรจะอยู่ในรูปกราฟิก เพราะฉะนั้นเมื่อกำหนดตัวอักษรเป็นตัวใหญ่, ตัวเข้ม, ตัวเอียง จะสามารถดูได้จากจอ และสามารถสั่งพิมพ์ได้ตามนั้น หรือที่เรียกว่า "WYSIWYG" (What you see is what you get)

7. การเชื่อมโยงข้อมูลระหว่างโปรแกรมที่เกี่ยวข้องกับข้อมูลนั้นจะเปลี่ยนแปลงแก้ไข ได้ทันที

8. ในวินโดว์ 3.0 ได้ออกแบบมาให้ใช้กับโปรแกรมข่ายงาน (Network) ได้อย่างง่ายดายโดยไม่ต้องซื้อชุดข่ายงานเพิ่มเติม

9. การใช้งานภายใต้ข่ายงานสามารถทำได้คือ

- File Manager สามารถควบคุมการต่อเชื่อมข่ายงานหรือไม่ก็ได้
- Control Panel สามารถกำหนดเครื่องพิมพ์ของข่ายงานหรือไม่กำหนดก็ได้
- Print Manager สามารถเปลี่ยนสถานะของงานพิมพ์ ทั้งที่เป็นงานพิมพ์ของข่ายงานและที่ไม่ใช่ข่ายงาน

ภาชนะนาท ค.

ระบบจัดการฐานข้อมูลออร์เรล

โดยทั่วไประบบจัดการฐานข้อมูลแบบสัมพันธ์ (RDBMS:Relational Database Management System) เป็นตัวเชื่อมต่อระหว่าง หน่วยเก็บข้อมูลแบบกายภาพ กับข้อมูลทางตรรกะ ในทางปฏิบัติ จะแบ่งออกเป็นกลุ่ม ๆ ตามลักษณะการกระทำกับข้อมูลบนเครื่องมือ (Tools) โดยสามารถใช้เครื่องมือเหล่านี้ในการทำงานดังต่อไปนี้

- ให้นิยามของฐานข้อมูล (Define a database)
- สอบถามข้อมูลจากฐานข้อมูล (Query the database)
- เพิ่มเติม แก้ไข และลบข้อมูล (Add, edit and delete data)
- เปลี่ยนแปลงโครงสร้างของฐานข้อมูล (Modify the structure of the database)
- รักษาความปลอดภัยแก่ข้อมูลจากการใช้งาน (Secure data from public access)
- ติดต่อสื่อสารระหว่างระบบเครือข่าย (Communicate within networks)
- ส่งข้อมูลออก และรับข้อมูลเข้า (Export and import data)

เนื่องจากระบบจัดการฐานข้อมูลแบบสัมพันธ์อนุญาตให้ผู้ใช้ ความคุมข้อมูลได้เต็มที่ ทำให้มีผลิตภัณฑ์รองรับการจัดการฐานข้อมูลแบบสัมพันธ์เกิดขึ้นมากมาย ซึ่งความสามารถเหล่านี้ ทำให้ผู้ใช้สามารถเลือกทำงานตามที่ต้องการได้

ในปัจจุบัน ระบบจัดการฐานข้อมูลแบบสัมพันธ์ ต้องการที่จะให้ผู้ใช้มากกว่า 1 คนใช้งานกับฐานข้อมูลเดียวกัน โดยมีความปลอดภัยเท่าที่ควร ดังนั้นโปรแกรมประยุกต์เกี่ยวกับฐานข้อมูลได้มีการพัฒนาเป็นระบบที่มีผู้ใช้หลายคน (Multiuser) ออร์เรลจะมีเครื่องมือช่วยหลาย ๆ แบบ เพื่อจัดการข้อมูลและการพัฒนาโปรแกรมประยุกต์ สำหรับเวอร์ชัน 5.1 ออร์เรลได้แบ่งกลุ่มของโปรแกรมกำหนดหน้าที่ ออกเป็นผลิตภัณฑ์ ซึ่งมีหลายทางที่จะเข้าไป และใช้ระบบตามที่ต้องการ

5.1 การจัดการข้อมูลของออร์เรล

ในออร์เรลข้อมูลทั้งหมดจะถูกเก็บและแสดงในรูปของตาราง (Table) ในแต่ละตารางจะประกอบไปด้วยคอลัมน์ (Columns) หรือบางทีเรียกว่าฟิลด์ (fields) และแถว (Rows)

!

แต่ละแถวข้อมูลจะเรียกว่า 1 เรกคอร์ด จากตารางผู้ใช้อาจจะเลือกสืบเขตของแถวหรือคอลัมน์ แต่ผลลัพธ์จะแสดงออกมาบนหน้าจอ หรือถูกพิมพ์ออกมาเป็นรูปแบบของตารางด้วย รูปที่ 5.1 แสดงโครงสร้างพื้นฐานของตาราง

Columns, or fields

Emp. #	Date	Time in	Timeout	Task
9041	9/22	8:05	11:59	ASSEMBLY

← One record

รูปที่ ค.1

วิว (view) ได้มาจากตารางซึ่งผู้ใช้สามารถสร้างเพื่อจุดประสงค์สำหรับการแสดงผล ถึงแม้ว่าวิวจะมองดูเหมือนกันว่าเป็นตารางจริง ๆ แต่ในฐานข้อมูล วิวจะเก็บในลักษณะนิยามเท่านั้น ด้วยเหตุผลนี้ วิวก็จะถูกอ้างว่าเป็นตารางเสมือน (Virtual Tables) โดยตารางที่ถูกวิวถ่ายทอดเรียกว่า ตารางพื้นฐาน (Base Tables) วิวอาจจะเกิดจากการรวมกันของ 2 ตารางพื้นฐาน หรือเป็นสืบเขตของตารางพื้นฐาน ผู้ใช้สามารถใช้วิวเพื่อที่จะเข้าถึงตารางได้ และเพื่อทำให้งานที่มีความยุ่งยากให้มีความง่ายขึ้น

เนื่องจากอราเคิลเป็นระบบจัดการฐานข้อมูลแบบสัมพันธ์ ผู้ใช้จึงสามารถที่จะเชื่อมโยงข้อมูลที่ถูกเก็บไว้ในหลาย ๆ ตารางเพื่อเพิ่มประสิทธิภาพในการทำงานและเพื่อการหลีกเลี่ยงการทำซ้ำ การเลือก (Selection) เป็นกระบวนการเพื่อสร้างตารางใหม่ซึ่งประกอบไปด้วยกลุ่มของแถวจากตารางใด ๆ ซึ่งตรงกับเกณฑ์ที่ต้องการ การออกแบบ (Projection) เป็นกระบวนการเพื่อสร้างตารางจากกลุ่มของคอลัมน์จากตารางใด ๆ ซึ่งตรงกับเกณฑ์ที่ต้องการ การร่วมกัน (Join) จะสร้างตารางใหม่ซึ่งเป็นการเชื่อมกันของแถวทั้งหมดใน 2 ตาราง โดยไม่รวมแถวที่มีข้อมูลเหมือนกัน

5.2 การเข้าถึงข้อมูลของออร์ราเคิล

ส่วนแกน (Core) ของออร์ราเคิลคือ SQL (Structured Query Language) เป็นภาษาซึ่งผู้ใช้สามารถติดต่อกับออร์ราเคิล SQL ประกอบไปด้วยกลุ่มคำพื้นฐานของภาษาอังกฤษ เช่น Select และ Create เป็นต้น ซึ่งผู้ใช้สามารถใช้คำสั่งโครงสร้างเพื่อที่จะเข้าถึงและจัดการข้อมูลที่ถูกเก็บไว้ในฐานข้อมูลแบบสัมพันธ์

ออร์ราเคิล ใช้ SQL เพื่อที่จะเข้าถึง (Access) เปลี่ยนแปลง (Modify) และแสดงผล (Display) ข้อมูลออร์ราเคิลยังสามารถใช้เครื่องมือ (Tools) อื่นในการกระทำกับข้อมูลได้เช่นกัน

5.3 เหตุผลของการเลือกออร์ราเคิล

ออร์ราเคิลมีมาหลายปีในวงการระบบฐานข้อมูลแบบสัมพันธ์ ถึงแม้จะไม่ใช่ความเป็นที่กังขัน และระบบที่ใช้กับไมโครคอมพิวเตอร์ เหมือนกับ dBASE IV แต่ออร์ราเคิล มีความสามารถในการใช้งานแบบผู้ใช้หลายคน การเข้าถึงการควบคุม และ SQL-Compatibility ก็ถูกเพิ่มเข้ามา นอกจากนี้ ออร์ราเคิลยังมีข้อดีสำหรับการใช้งาน คือ

5.3.1 ออร์ราเคิลให้ความปลอดภัยแก่ผู้ใช้

ออร์ราเคิลมีหลายลักษณะเพื่อที่จะให้ความแน่นอนในความถูกต้องของฐานข้อมูล เมื่อมีการขัดจังหวะ (Interruption) เกิดขึ้นในขณะที่ทำงานอยู่ การกลับคืนสู่ระดับเดิม (Rollback) สามารถที่จะเริ่มให้ฐานข้อมูลกลับไปยังตำแหน่งที่ก่อนที่จะเกิดความผิดพลาด

ออร์ราเคิลมีฟังก์ชันมากมายสำหรับการรักษาความปลอดภัยของข้อมูล คำสั่ง Grant และ Revoke จะจำกัดการเข้าถึงของข้อมูลลงไปถึงระดับแถวและคอลัมน์ วิว ก็เป็นรูปแบบหนึ่งสำหรับการจำกัดการเข้าถึงตารางพื้นฐานในระบบฐานข้อมูล ซึ่งผู้ใช้จะพบว่าหลายทางที่จะควบคุมการเข้าถึงฐานข้อมูลในออร์ราเคิล

5.3.2 ออร์ราเคิลสามารถดำเนินงานได้อย่างเหมาะสม

ออร์ราเคิล ได้ถูกปรับปรุงอย่างต่อเนื่อง เพื่อที่จะให้มีความสามารถดำเนินงานได้อย่างเหมาะสม ฐานข้อมูลที่มีขนาดใหญ่ เนื่องจากระบบฐานข้อมูลแบบสัมพันธ์มีข้อเสียในเรื่องของ

ความล่าช้าในการเข้าถึงข้อมูลเป็นอย่างมาก ออราเคิลได้พัฒนาตัวเองอย่างตัวเอง ซึ่งผลก็คือชุดคำสั่งประจำสามารถ ค้นหาทางที่ต่ำที่สุดที่จะไปยังข้อมูล อย่างรวดเร็ว โดยอัตโนมัติ เทคนิคในการจัดกลุ่มของ ออราเคิลเพื่อที่จะเก็บข้อมูลลงบนดิสก์ ก็เป็นอีกคุณสมบัติหนึ่ง และยังมีฟังก์ชันเพิ่มเติมช่วยควบคุมการติดตามข้อมูลที่มีความซับซ้อนด้วย

5.3.3 ออราเคิลสนับสนุนการพัฒนาโปรแกรมประยุกต์

SQL*Forms เป็นเครื่องมือที่ให้ผู้ใช้งานได้ง่าย (User Friendly) อย่างดียิ่ง ในการสร้างฟอร์มผู้ใช้งานสามารถเริ่มต้นด้วย การใช้ฟอร์มที่มีอยู่แล้วหรือฟังก์ชันในการจัดการหน้าจอเพื่อสร้างรายละเอียดบนหน้าจอ สำหรับการเข้าถึง และการแก้ไข ตารางหลาย ๆ ตาราง และเพื่อควบคุมและป้อนข้อมูลเข้าไป

ใน SQL*Forms ออราเคิลได้จัดให้หน่วยควบคุม ที่เรียกว่า Trigger โดยมีผลในการทำงานของผู้ใช้บนฟิลด์ก่อน ระหว่าง และหลัง การป้อนข้อมูล Trigger เหล่านี้สามารถปฏิบัติตามคำสั่ง SQL และคำสั่งใน SQL*Forms หรือโปรแกรมย่อยภายนอกของภาษาเชิงกระบวนการคำสั่ง (Procedural Language) SQL*Forms เป็นเครื่องมือของภาษายูคที 4 (4GL) ซึ่งจะปรับปรุงตามความต้องการของผู้ใช้ได้อย่างดี

5.3.4 ออราเคิลใช้ชุดคำสั่ง SQL

ออราเคิลใช้ชุดคำสั่ง SQL ซึ่งใกล้เคียงกับมาตรฐาน ANSI และมีความเข้ากันได้กับ DB2 ของ IBM และ DS/SQL

ผู้ใช้งานจะได้รับประโยชน์ของการปรับปรุงอย่างมากของออราเคิลเพื่อให้เข้าใกล้มาตรฐาน SQL ออราเคิลได้เพิ่มคำสั่งรูปแบบของการรายงานเพื่อที่จะขยายความสามารถในการรายงานผลของผลลัพธ์ และเพื่อที่จะให้มีการแสดงผลการรายงานอย่างต่อเนื่อง นอกจากนี้ ออราเคิล ได้เพิ่มฟังก์ชันทางสถิติ (Statistical) ทางคณิตศาสตร์ (Arithmetic) ชุดอักขระ (String) และวันที่เวลา (Date/Time) ด้วย

ภาคผนวก ง.
วิธีการติดตั้งเมตาสกีมม่า

ในการใช้งานระบบช่วยงานฐานข้อมูล สิ่งที่ต้องคำนึงถึงเป็นอันดับแรก คือการสร้างเมตาสกีมม่าขึ้นในระบบฐานข้อมูลเสียก่อน โดยอาศัยไฟล์ที่ชื่อ MET.SQL ในไดเรกทอรี C:\ORACLE5 ซึ่งเป็นไฟล์โปรแกรมภาษา SQL ที่มีคำสั่งการสร้างรีเลชันต่าง ๆ ของเมตาสกีมม่า รวมถึงรีเลชันของแมปปิงอินฟอร์เมชัน พิมพ์คำว่า

```
C:\ORACLE5\>SQLPLUS META META <ENTER>
```

เมื่อเข้าสู่การทำงานใน SQL-plus ซึ่งจะแสดงเครื่องหมายเตรียมพร้อมในลักษณะ

```
SQL > _
```

พิมพ์คำว่า

```
SQL > @MET <ENTER>
```

SQL-plus จะทำการสร้างรีเลชันของเมตาสกีมม่าและแมปปิงอินฟอร์เมชัน ลงในฐานข้อมูลที่อยู่โดยอาศัยไฟล์ MET.SQL แสดงดังรูปที่ ง.1

```
CREATE TABLE      OBJECTTYPE (
    OBJECTNAME      CHAR(20),
    KINDOFOBJECT    CHAR(10) );
```

```
CREATE TABLE      LABELTYPE (
    OBJECTNAME      CHAR(20),
    TYPENAME        CHAR(20),
    LABELORDER      CHAR(5) );
```

```
CREATE TABLE      SUBTYPE (
    SUBNAME         CHAR(20),
    SUPERNAME       CHAR(20) );
```

```
CREATE TABLE      ROLE (
    ROLENAME        CHAR(20),
    ROLETYPE        CHAR(10),
    RELATIONNAME     CHAR(20),
    ROLEORDER       CHAR(1) );
```

```
CREATE TABLE      OBJECTROLE (
    ROLENAME        CHAR(20),
    OBJECTNAME       CHAR(20) );
```

```
CREATE TABLE      NESTING (
    ROLENAME        CHAR(20),
    RELATIONNAME     CHAR(20) );
```

```
CREATE TABLE      RELATIONTYPE (
    RELATIONNAME     CHAR(20),
    KINDOFRELATION   CHAR(10) );
```

```
CREATE TABLE      ROLEJGROUP (
    ROLEJNAME      CHAR(20),
    ROLEJGROUP     CHAR(10) );
```

```
CREATE TABLE      CONSTRAINT (
    CONST#         CHAR(10),
    KINDJOFCONST   CHAR(10),
    CONSTJNOTE     CHAR(1));
```

```
CREATE TABLE      CONSTJROLE (
    CONST#         CHAR(10),
    ROLEJGROUP     CHAR(10) );
```

```
CREATE TABLE      OBJECTJCONST (
    CONST#         CHAR(10),
    OBJECTJNAME    CHAR(20) );
```

```
CREATE TABLE      RANGEJCONST (
    CONST#         CHAR(10),
    MINJNUMBER     CHAR(5),
    MAXJNUMBER     CHAR(5) );
```

```
CREATE TABLE      MEMBERSHIPJCONST (
    CONST#         CHAR(10),
    MEMBER         CHAR(10) );
```

```
CREATE TABLE      SUBSETJCONST (
    CONST#         CHAR(10),
    ROLEJGROUP     CHAR(10) );
```

```

CREATE VIEW      SUPERSETCONST AS
  SELECT  T1.CONST# , T2.ROLEGROUP
  FROM    CONSTROLE T1 , SUBSETCONST T2
  WHERE   T1.CONST# = T2.CONST#
  AND     T2.ROLEGROUP <> T1.ROLEGROUP ;

```

```

CREATE TABLE    FREQCONST (
  CONST#          CHAR(10),
  UPPERLIMIT      CHAR(5),
  LOWERLIMIT      CHAR(5) );

```

```

CREATE TABLE UNIQREF(
  ENTITYNAME      CHAR(20),
  RELATIONNAME     CHAR(20),
  ROLENAME        CHAR(20),
  LABELNAME       CHAR(20),
  COLNAME         CHAR(20) );

```

```

CREATE TABLE NONUNIQREF(
  ENTITYNAME      CHAR(20),
  RELATIONNAME     CHAR(20),
  LABELNAME       CHAR(20),
  COLNAME         CHAR(20) );

```

```

CREATE TABLE    NORMALFACTTYPE (
  RELATIONNAME     CHAR(20),
  TABLENAME       CHAR(20) );

```

```

CREATE TABLE    REFERENCE (
  RELATIONNAME     CHAR(20),
  TABLENAME       CHAR(20) );

```

```
CREATE TABLE      PRIMARYKEY (
    TABLENAME      CHAR(20),
    COLNAME          CHAR(20) );
```

```
CREATE TABLE      DERIVEDFACTTYPE (
    RELATIONNAME     CHAR(20),
    VIEWNAME         CHAR(20),
    DERIVEDIDIF      CHAR(100) );
```

```
CREATE UNIQUE INDEX OBJECTNAME1
    ON OBJECTTYPE(OBJECTNAME);
CREATE UNIQUE INDEX OBJECTNAME2
    ON LABELTYPE(OBJECTNAME);
CREATE UNIQUE INDEX SUBTYPE
    ON SUBTYPE(SUBNAME,SUPERNAME);
CREATE UNIQUE INDEX ROLENAME1
    ON ROLE(ROLENAME);
CREATE UNIQUE INDEX ROLENAME2
    ON OBJECTROLE(ROLENAME);
CREATE UNIQUE INDEX ROLENAME3
    ON NESTING(ROLENAME);
CREATE UNIQUE INDEX RELATIONNAME1
    ON NORMALFACTTYPE(RELATIONNAME);
CREATE UNIQUE INDEX RELATIONNAME2
    ON DERIVEDFACTTYPE(RELATIONNAME);
CREATE UNIQUE INDEX RELATIONNAME3
    ON RELATIONTYPE(RELATIONNAME);
CREATE UNIQUE INDEX RELATIONNAME4
    ON REFERENCE(RELATIONNAME);
CREATE UNIQUE INDEX PRIMARY
```



```

ON PRIMARY KEY (TABLENAME, COLNAME);
CREATE UNIQUE INDEX ROLEGR
ON ROLEGROUP (ROLENAME, ROLEGROUP);
CREATE UNIQUE INDEX CONST1
ON CONSTRAINT (CONST#);
CREATE UNIQUE INDEX CONST2
ON CONSTROLE (CONST#, ROLEGROUP);
CREATE UNIQUE INDEX CONST3
ON OBJECTCONST (CONST#);
CREATE UNIQUE INDEX CONST4
ON RANGECONST (CONST#);
CREATE UNIQUE INDEX CONST5
ON MEMBERSHIPCONST (CONST#);
CREATE UNIQUE INDEX CONST6
ON SUBSETCONST (CONST#);
CREATE UNIQUE INDEX CONST7
ON FREQCONST (CONST#);

```

รูปที่ ง.1 แสดงไฟล์ติดตั้งเมตาสกิมมาและเมปिंगอินฟอร์เมชัน MET.SQL

