การปรับรูปแบบบรรทัดฐานในฐานข้อมูลเชิงสัมพันธ์ด้วยเทคนิคการวิเคราะห์ความสัมพันธ์

นายณัฐพล พันนุรัตน์

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์
มหาวิทยาลัยเทคโนโลยีสุรนารี
ปีการศึกษา 2551

NORMALIZATION IN RELATIONAL DATABASE WITH ASSOCIATION ANALYSIS TECHNIQUE

Natthapon Pannurat

A Thesis Submitted in Partial Fulfillment of the Requirements for the

Degree of Master of Engineering in Computer Engineering

Suranaree University of Technology

Academic Year 2008

การปรับรูปแบบบรรทัดฐานในฐานข้อมูลเชิงสัมพันธ์
ด้วยเทคนิคการวิเคราะห์ความสัมพันธ์

มหาวิทยาลัยเทคโนโลยีสุรนารี อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

คณะกรรมการสอบวิทยานิพนธ์

_____

(รศ. ดร.กิตติศักดิ์ เกิดประสพ)
ประธานกรรมการ

_____

(รศ. ดร.นิตยา เกิดประสพ)
กรรมการ (อาจารย์ที่ปรึกษาวิทยานิพนธ์)

_____

(ผศ. ดร.พิชโยทัย มหัทธนาภิวัฒน์)
กรรมการ

_____

(อ. ดร.ปรเมศวร์ ห่อแก้ว)
กรรมการ

_____

(ศ. ดร.ไพโรจน์ สัตยธรรม)
รองอธิการบดีฝ่ายวิชาการ

_____

(รศ. น.อ. ดร.วรพจน์ ขำพิศ)
คณบดีสำนักวิชาวิศวกรรมศาสตร์

ณัฐพล พันนุรัตน์ : การปรับรูปแบบบรรทัดฐานในฐานข้อมูลเชิงสัมพันธ์ด้วยเทคนิคการวิเคราะห์ความสัมพันธ์ (NORMALIZATION IN RELATIONAL DATABASE WITH ASSOCIATION ANALYSIS TECHNIQUE) อาจารย์ที่ปรึกษา : รศ. ดร.นิตยา เกิดประสพ, 131 หน้า.

ระบบการจัดการฐานข้อมูลเชิงสัมพันธ์ (Relational Database Management System) ทุกระบบจำเป็นต้องมีรีเลชัน (Relation) หรือตาราง (Table) ไว้สำหรับเก็บข้อมูลเสมอ ยิ่งข้อมูลมีรายละเอียดมากก็ต้องมีแอททริบิวท์ (Attribute) สำหรับเก็บข้อมูลมากเท่านั้น การออกแบบตารางเก็บข้อมูลที่ไม่ดีอาจส่งผลให้มีการเก็บข้อมูลที่ซ้ำซ้อนกันได้ ซึ่งจะส่งผลให้เกิดข้อผิดพลาดในการเรียกใช้ข้อมูล เช่น การเพิ่มข้อมูล การลบหรือแก้ไขข้อมูลที่มีอยู่ในรีเลชัน หรืออาจเกิดความไม่คงที่ไม่แน่นอนหรือขัดแย้ง (Inconsistency) ของข้อมูลซึ่งเรียกว่า ความผิดปกติ (Anomaly) จึงเป็นที่มาของการคิดค้นกระบวนการปรับรูปแบบบรรทัดฐาน (Normalization) ตารางข้อมูลเพื่อแก้ไขปัญหาดังกล่าว

ในงานวิจัยชิ้นนี้มุ่งเน้นที่จะพัฒนาแนวทางในการปรับรูปแบบบรรทัดฐานถึงขั้นที่สาม ซึ่งเป็นขั้นที่สามารถนำไปสร้างฐานข้อมูลได้อย่างมีประสิทธิภาพ โดยจะนำเทคนิคการค้นหาความสัมพันธ์ของข้อมูล ซึ่งเป็นเทคนิคของงานทางด้านการทำเหมืองข้อมูลเข้ามาช่วยในขั้นตอนการปรับรูปแบบบรรทัดฐานของฐานข้อมูลเชิงสัมพันธ์ โดยจะปรับตารางให้อยู่ในรูปแบบบรรทัดฐานขั้นที่สาม ซึ่งเพียงพอสำหรับงานด้านฐานข้อมูลจึงไม่จำเป็นต้องปรับไปถึงบรรทัดฐานขั้นที่ห้า

สาขาวิชาวิศวกรรมคอมพิวเตอร์ ลายมือชื่อนักศึกษา_____

ปีการศึกษา 2551 ลายมือชื่ออาจารย์ที่ปรึกษา _____

NATTHAPON PANNURAT : NORMALIZATION IN RELATION DATABASE WITH ASSOCIATION ANALYSIS TECHNIQUE.
THESIS ADVISOR : ASSOC. PROF. NITTAYA KERDPRASOP, Ph.D.,
131 PP.

NORMALIZATION/RELATIONAL DATABASE/ASSOCIATION ANALYSIS

Relational database management systems store data in relations or tables. The design of tables in relational databases is important. The bad design of data tables brings some problems to the database systems such as insert anomaly, delete anomaly and update anomaly. Therefore, the normalization process has been introduced to resolve these problems. In this paper we apply the association analysis technique from the field of data mining to help normalizing tables that are badly designed. In the real world databases, normalization up to fifth normal form is hard to happen. So we would normalize to the third normal form which is adequate for most database applications.

School of Computer Engineering Student's Signature _____
Academic Year 2008 Advisor's Signature _____

กิตติกรรมประกาศ

วิทยานิพนธ์นี้สำเร็จลุล่วงด้วยดี ผู้วิจัยขอกราบขอบพระคุณ บุคคล และกลุ่มบุคคลต่าง ๆ ที่ได้กรุณาให้คำปรึกษา แนะนำ ช่วยเหลืออย่างดียิ่ง ทั้งในด้านวิชาการ และด้านการดำเนินงานวิจัย ดังต่อไปนี้

- รศ. ดร. นิตยา เกิดประสพ อาจารย์ที่ปรึกษาวิทยานิพนธ์
- รศ. ดร. กิตติศักดิ์ เกิดประสพ หัวหน้าสาขาวิชาวิศวกรรมคอมพิวเตอร์ ผศ. ดร. พิชโยทัย มหัทธนาภิวัฒน์ ผศ. ดร.คะชา ชาญศิลป์ ผศ. สมพันธุ์ ชาญศิลป์ และ ดร. ปรเมศวร์ ห่อแก้ว อาจารย์ประจำสาขาวิชาวิศวกรรมคอมพิวเตอร์ สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี
- คุณจริยาพร ศรีวิไลลักษณ์ ที่ให้คำปรึกษาการจัดรูปแบบ และช่วยตรวจทานความถูกต้องของวิทยานิพนธ์
- คุณกัลญา พับโพธิ์ เลขานุการสาขาวิชาวิศวกรรมคอมพิวเตอร์ ที่ให้ความช่วยเหลือในการประสานงานด้านเอกสารต่าง ๆ ระหว่างศึกษา
- คุณลักษมี โขมโนทัย ที่เปิดเผยซอสโค้ด และคอยช่วยเหลือ ให้คำแนะนำ
- คุณนริศ มิ่งโมรา คุณปฐมพงศ์ พันนุรัตน์ คุณอภิชัย ฤทธิ์ธงชัยเลิศ และบัณฑิตศึกษาสาขาวิชาวิศวกรรมคอมพิวเตอร์ ทุกท่านที่ให้คำปรึกษาและช่วยเหลือด้วยดีมาโดยตลอด
- คุณวิภาศิริ พุ่มโพธิ์ ที่คอยให้กำลังใจ และคอยกระตุ้นให้ผู้วิจัยทำงานจนสำเร็จลุล่วงทันเวลาที่กำหนด

นอกจากนี้ ขอขอบคุณครู อาจารย์ทั้งในอดีตและปัจจุบันที่ให้ความรู้แก่ผู้วิจัยจนประสบความสำเร็จในชีวิต

ท้ายที่สุดที่จะลืมไม่ได้ ขอกราบขอบพระคุณ บิดา มารดา ที่ให้กำเนิด อบรม เลี้ยงดูด้วยความรัก และส่งเสริมการศึกษาเป็นอย่างดีมาโดยตลอด ทำให้ผู้วิจัยมีความรู้ ความสามารถ มีจิตใจที่เข้มแข็ง รวมทั้งเป็นกำลังใจที่ยิ่งใหญ่ให้แก่ผู้วิจัย จนทำให้ผู้วิจัยประสบความสำเร็จในชีวิตเรื่อยมา

ณัฐพล พันนุรัตน์

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

ตารางที่ **หน้า**

สารบัญตาราง (ต่อ)

ตารางที่ **หน้า**

สารบัญตาราง (ต่อ)

ตารางที่ **หน้า**

สารบัญรูป

รูปที่ **หน้า**

สารบัญรูป (ต่อ)

รูปที่ **หน้า**

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มาของปัญหา

ระบบการจัดการฐานข้อมูลเชิงสัมพันธ์ (Relational Database Management System: RDBMS) ทุกระบบจะต้องมีรีเลชันหรือตาราง (Table) ไว้สำหรับเก็บข้อมูลเสมอ ยิ่งข้อมูลมีรายละเอียดมากก็ต้องมีแอทริบิวท์ (Attribute) สำหรับเก็บข้อมูลมากเท่านั้น การออกแบบตารางเก็บข้อมูลที่ไม่ดีอาจส่งผลให้มีการเก็บข้อมูลที่ซ้ำซ้อนกันได้ หรือเกิดข้อผิดพลาดในการเรียกใช้ข้อมูล เช่น การเพิ่มข้อมูล การลบหรือแก้ไขข้อมูลที่มีอยู่ในรีเลชัน หรืออาจเกิดความไม่คงที่ ไม่แน่นอน หรือขัดแย้ง (Inconsistency) ของข้อมูลซึ่งเรียกว่า ความผิดปกติ (Anomaly) จึงเป็นที่มาของการคิดค้นกระบวนการปรับบรรทัดฐาน (Normalization process) ของตารางข้อมูลเพื่อแก้ไขปัญหาดังกล่าว การเปลี่ยนแปลงข้อมูลที่อยู่ในโครงสร้างของตารางที่ไม่อยู่ในรูปแบบบรรทัดฐาน (Unnormalized form) จากรูปแบบที่มีความซ้ำซ้อน (Redundancy) มาอยู่ในรูปแบบบรรทัดฐาน โดยรูปแบบบรรทัดฐานมีทั้งหมดด้วยกัน 6 รูปแบบ ดังนี้

1. รูปแบบบรรทัดฐานระดับที่หนึ่ง (First Normal Form: 1NF)
2. รูปแบบบรรทัดฐานระดับที่สอง (Second Normal Form: 2NF)
3. รูปแบบบรรทัดฐานระดับที่สาม (Third Normal Form: 3NF)
4. รูปแบบบรรทัดฐานคอดด์ (Boyce-Codd Normal Form: BCNF)
5. รูปแบบบรรทัดฐานระดับที่สี่ (Fourth Normal Form: 4NF)
6. รูปแบบบรรทัดฐานระดับที่ห้า (Fifth Normal Form: 5NF)

การค้นหาความรู้จากฐานข้อมูล (Knowledge Discovery in Databases: KDD) เป็นการประยุกต์ใช้ข้อมูลที่เก็บอยู่ในฐานข้อมูลให้เกิดประโยชน์สูงสุด แก่องค์กรหรือหน่วยงานที่เป็นเจ้าของข้อมูลนั้น ๆ การประยุกต์ใช้ข้อมูลที่กล่าวถึงนี้ มักจะเป็นการสรุปภาพรวมของข้อมูล การวิเคราะห์แนวโน้มการเปลี่ยนแปลงของข้อมูล และการค้นหาความสัมพันธ์ที่ซ่อนอยู่ภายในกลุ่มของข้อมูล เป็นต้น สิ่งที่ค้นพบจาการค้นหาความรู้จากฐานข้อมูลจะเรียกว่า ความรู้ (Knowledge) และความรู้ที่ได้นั้นสามารถนำไปใช้ประกอบการวางแผนการดำเนินธุรกิจ หรือนำไปใช้เพื่อให้ก่อประโยชน์แก่หน่วยงานนั้น ๆ เช่น องค์การนาซ่าใช้เทคโนโลยีการค้นหาความรู้จากฐานข้อมูล ใน

การวิเคราะห์ข้อมูลที่ส่งมาจากดาวเทียมที่โคจรรอบโลก เพื่อค้นหารูปแบบการเปลี่ยนแปลงของสภาพภูมิอากาศบนพื้นผิวโลก เป็นต้น

การทำเหมืองข้อมูล (Data mining) คือ กระบวนการในการสกัดหรือค้นหาความรู้จากฐานข้อมูล เหมือนกับการค้นหาความรู้จากฐานข้อมูล แต่คำว่า Data mining จะนิยมใช้ในหมู่นักสถิติ นักวิเคราะห์ข้อมูล และนักสารสนเทศ ในขณะที่คำว่า KDD มักจะใช้ในกลุ่มของนักวิจัย สาขาปัญญาประดิษฐ์ (Artificial intelligence) และสาขา Machine learning

เทคในการทำเหมืองข้อมูลที่นิยมนำมาใช้กับฐานข้อมูลขนาดใหญ่ทางด้านธุรกิจ คือ เทคนิควิธีในการวิเคราะห์ความสัมพันธ์ของข้อมูล (Association analysis technique) การค้นหาความสัมพันธ์ของข้อมูลในฐานข้อมูลทางด้านธุรกิจ จะทำให้ทราบความต้องการของผู้บริโภค โดยความรู้ที่ได้จากการทำเหมืองข้อมูลในลักษณะนี้ จะอยู่ในรูปของกฎความสัมพันธ์ในลักษณะ ถ้า ... แล้ว ... หรือจากส่วนที่เป็นสาเหตุไปสู่ผลลัพธ์ ซึ่งเราจะนำกฎความสัมพันธ์ที่ได้นี้ไปใช้ประกอบการวางแผนทางธุรกิจ

ในงานวิจัยนี้มุ่งเน้นที่จะพัฒนาแนวทางในการปรับรูปแบบบรรทัดฐานถึงขั้นที่สาม ซึ่งเป็นขั้นที่สามารถนำไปสร้างฐานข้อมูลได้อย่างมีประสิทธิภาพ โดยจะนำเทคนิคการค้นหาความสัมพันธ์ของข้อมูล ซึ่งเป็นเทคนิคของงานทางด้านการทำเหมืองข้อมูลเข้ามาช่วย ในขั้นตอนการปรับรูปแบบบรรทัดฐานของฐานข้อมูลเชิงสัมพันธ์

1.2 วัตถุประสงค์ของการวิจัย

1. เพื่อออกแบบตารางในการเก็บข้อมูลได้อย่างมีประสิทธิภาพ
2. เพื่อช่วยลดระยะเวลาในการออกแบบตารางในการเก็บข้อมูล
3. เพื่อทำการออกแบบรูปแบบโมเดลข้อมูลจากการทำเหมืองข้อมูลที่เหมาะสมกับการปรับรูปแบบบรรทัดฐาน
4. เพื่อพัฒนาวิธีการแปลงโมเดลข้อมูลให้เป็นตารางข้อมูลที่อยู่ในรูปแบบบรรทัดฐานระดับที่ 3

1.3 ขอบเขตของงานวิจัย

1. ศึกษาวิธีการค้นหากฎความสัมพันธ์ด้วยอัลกอริทึมเอไพรออริ
2. โครงการวิจัยนี้เป็นการนำหลักการของกระบวนการการปรับรูปแบบบรรทัดฐาน และเทคนิคการทำเหมืองข้อมูลด้วยการหาความสัมพันธ์ของข้อมูลมาทำงานรวมกัน เพื่อใช้ในการออกแบบตารางในฐานข้อมูลให้มีประสิทธิภาพ

3. อัลกอริทึมที่นำมาใช้ คือ อัลกอริทึมเอไพรออริ เพียงอัลกอริทึมเดียวเท่านั้น
4. งานวิจัยนี้มุ่งเน้นที่จะพัฒนาอัลกอริทึมเอไพรออริ ด้วยภาษา SQL เพื่อใช้งานในระบบฐานข้อมูลเชิงสัมพันธ์เท่านั้น
5. การปรับบรรทัดฐาน จะทำถึงบรรทัดฐานรูปแบบที่ 3

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1. สามารถปรับรูปแบบบรรทัดฐานจากรีเลชันที่ไม่เป็นบรรทัดฐาน ให้อยู่ในรูปแบบบรรทัดฐานระดับที่ 3 ได้อย่างรวดเร็ว
2. ได้โปรแกรมสำหรับช่วยงานในการปรับรูปแบบบรรทัดฐาน
3. ความรู้ที่ได้จากการทำงานวิจัยสามารถนำไปใช้ในการพัฒนา การออกแบบระบบฐานข้อมูลให้มีประสิทธิภาพยิ่งขึ้นต่อไปในอนาคต

บทที่ 2
ปริทัศน์วรรณกรรมและงานวิจัยที่เกี่ยวข้อง

ทฤษฎีหนึ่งที่ผู้ออกแบบฐานข้อมูลจะต้องนำมาใช้ในการแปลง หรือแก้ไขปัญหาเกี่ยวกับข้อมูลที่อยู่ในรูปแบบที่ซับซ้อน ให้อยู่ในรูปแบบที่ง่ายต่อการนำไปใช้และก่อให้เกิดปัญหากับระบบฐานข้อมูลน้อยที่สุด ได้แก่ ทฤษฎีที่เรียกว่า กระบวนการปรับบรรทัดฐาน (Normalization process) ซึ่งเป็นกระบวนการที่ใช้ในการกระจายรีเลชัน (Relation) ที่มีโครงสร้างที่ซับซ้อนออกเป็นรีเลชันย่อย ๆ เป็นโครงสร้างที่ง่าย และอยู่ในรูปแบบบรรทัดฐาน ที่สามารถนำไปใช้งานได้อย่างมีประสิทธิภาพ เนื้อหาในบทนี้จะนำเสนอปริทัศน์วรรณกรรมและงานวิจัยที่เกี่ยวข้องกับงานวิจัยชิ้นนี้ โดยในหัวข้อที่ 2.1 จะกล่าวถึงสาเหตุที่ต้องมีการปรับบรรทัดฐาน และความสำคัญของการปรับบรรทัดฐาน หัวข้อที่ 2.2 จะกล่าวถึงความหมายของกระบวนการปรับบรรทัดฐาน และจุดประสงค์ของการปรับบรรทัดฐาน หัวข้อที่ 2.3 จะกล่าวถึงฟังก์ชันการขึ้นต่อกันว่ามีกี่ประเภท และแต่ละประเภทมีรูปแบบเป็นอย่างไร หัวข้อที่ 2.4-2.9 จะกล่าวถึงนิยามของรูปแบบบรรทัดฐานแต่ละระดับ และรายละเอียดขั้นตอนวิธีการปรับรูปแบบบรรทัดฐานในแต่ละระดับ ตั้งแต่ระดับที่ 1 จนกระทั่งถึงระดับที่ 5 โดย หัวข้อที่ 2.10 จะกล่าวถึง การแตกรีเลชันมากเกินไป หัวข้อที่ 2.11 จะกล่าวถึง การค้นหากฎความสัมพันธ์ หัวข้อที่ 2.12 จะกล่าวถึงวิธีการค้นหากฎความสัมพันธ์ หัวข้อที่ 2.13 จะกล่าวถึง ประเภทของกฎความสัมพันธ์ว่ามีทั้งหมดกี่ประเภทอะไรบ้าง หัวข้อที่ 2.14 จะกล่าวถึง การค้นหากฎความสัมพันธ์ของข้อมูลในฐานข้อมูลขนาดใหญ่ และในหัวข้อที่ 2.15 จะกล่าวถึงอัลกอริทึมเอไพรออริ

2.1 ความสำคัญของการปรับบรรทัดฐาน

ในขั้นตอนการออกแบบโมเดลฐานข้อมูลเชิงสัมพันธ์ (Codd, 1982) โดยไม่ได้ผ่านโมเดลแบบ E-R นั้น ขั้นตอนแรกที่สุดที่ผู้ออกแบบระบบจะต้องทำคือ การเก็บรวบรวมข้อมูลที่เกี่ยวข้องกับการทำงานของระบบเดิม โดยข้อมูลที่เก็บได้นี้อาจอยู่ในรูปแบบของเอกสารต่าง ๆ ดังตัวอย่างที่แสดงในรูปที่ 2.1 ที่แสดงตัวอย่างของรายงานการสรุปการคำนวณค่าแรงงานของคนงานแต่ละคนที่ทำงานกับบริษัท (สมจิตร อาจอินทร์ และ งามนิจ อาจอินทร์, 2549)

รายงานค่าแรงคนงานก่อสร้าง ประจำวันที่ 15 มกราคม พ.ศ. 2551 บริษัท ขยันก่อสร้าง จำกัด									
รหัสคนงาน	ชื่อคนงาน	ประเภทความชำนาญ	อัตราค่าแรง/ช.ม.	สถานที่ก่อสร้าง	วันเริ่มทำงาน	จำนวนช.ม. ที่ทำงาน	ค่าแรงปกติ	ค่าแรงนอกเวลา	รวมค่าแรง
1245	สุดใจ	ไฟฟ้า	16.0	516 311	01/01/49 08/01/49	50 40	640.0 640.0	560.0 -	1200.0 640.0
								รวมค่าแรงสุทธิ	1840.0
2521	ดีใจ	ปูน	16.0	516 450 431	01/01/49 15/01/49 23/01/49	56 45 36	800.0 720.0 576.0	240.0 - -	1040.0 720.0 576.0
								รวมค่าแรงสุทธิ	2336.0

รูปที่ 2.1 แสดงตัวอย่างรายงานค่าแรงงานของคนงานประจำเดือน มกราคม ปี 2551

จากตัวอย่างรายงานค่าแรงงานของบริษัทฯในรูปที่ 2.1 เป็นรูปแบบรายงานที่ผู้ปฏิบัติงาน เช่น เจ้าหน้าที่การเงินของบริษัทต้องการ โดยปกติถ้าไม่มีคอมพิวเตอร์มาช่วยในการทำงาน การสร้างรายงานดังกล่าวจะมีความยุ่งยากและใช้เวลานานมาก อีกทั้งยังมีข้อผิดผลาดค่อนข้างมาก แต่เมื่อมีการนำคอมพิวเตอร์เข้ามาช่วยในการทำงานดังกล่าว ผู้ออกแบบระบบจะต้องเก็บรวบรวมข้อมูลที่ได้นี้มาทำการวิเคราะห์ว่าควรจะออกแบบระบบฐานข้อมูลอย่างไร เพื่อให้ได้ฐานข้อมูลที่ดี มีประสิทธิภาพ สามารถเรียกใช้ข้อมูลได้โดยง่าย และสะดวกต่อการควบคุมจัดการฐานข้อมูลนั้น

จากข้อมูลที่ได้จากรายงานดังกล่าวจะมีรูปแบบที่ซับซ้อน เนื่องจากมีการเก็บรายละเอียดของข้อมูลทุกอย่างไว้ด้วยกันหมด ซึ่งถ้าผู้ออกแบบระบบมีการนำข้อมูลจากรายงานนี้บางส่วนมาประกอบกันเป็นข้อมูลอื่น ๆ ที่เกี่ยวกับคนงานแต่ละคน แล้วสร้างรีเลชันที่มีรูปแบบดังตารางที่ 2.1 ซึ่งเป็นรีเลชันเพียงรีเลชันเดียวและตั้งชื่อรีเลชันนี้ว่า “รายงานการทำงาน” จะเห็นว่ารีเลชันดังกล่าวมีรูปแบบที่ผิดคุณสมบัติข้อที่ 1 ของรีเลชันที่กล่าวไว้ว่า ช่องแต่ละช่องของรีเลชันจะต้องบรรจุข้อมูลที่มีเพียงค่าเดียวเท่านั้น แต่ช่องข้อมูลของคอลัมน์ที่เป็น รหัสสถานที่ก่อสร้าง ประเภทสถานที่ก่อสร้าง วันที่เริ่มงาน และ จำนวนช.ม.ทั้งหมดที่ทำ จะมีข้อมูลมากกว่าหนึ่งค่าเก็บอยู่ (Multi-valued attribute) ซึ่งหมายความว่าคนงานแต่ละคนอาจถูกกำหนดให้ไปทำงานยังสถานที่

ก่อสร้างได้มากกว่าหนึ่งแห่ง จึงเรียกลักษณะของข้อมูลที่อยู่ในคอลัมน์เหล่านี้ว่า กลุ่มข้อมูลซ้ำ (Repeating groups) และจะเรียกรีเลชันที่มีกลุ่มข้อมูลซ้ำนี่ว่า เป็นรีเลชันที่ยังไม่ผ่านการปรับบรรทัดฐาน (Unnormalized relation)

ตารางที่ 2.1 ตารางรายงานการทำงาน

รหัสคนงาน	ชื่อคนงาน	ประเภทความชำนาญ	รหัสผู้ควบคุม	อัตราค่าแรง/ช.ม.	รหัสสถานที่ก่อสร้าง	ประเภทสถานที่ก่อสร้าง	วันเริ่มทำงาน	จำนวนช.ม. ที่ทำงาน
1245	สุดใจ	ไฟฟ้า	1441	16.0	516	บ้านพัก	01/01/49	50
					311	สำนักงาน	08/01/49	40
2521	ดีใจ	ปูน	2533	16.0	516	บ้านพัก	01/01/49	56
					450	ร้านค้า	15/01/49	45
					431	บ้านพัก	23/01/49	36

2.2 กระบวนการปรับบรรทัดฐาน (Normalization process)

กระบวนการปรับบรรทัดฐาน เป็นกระบวนการที่ใช้ในการกระจายรีเลชันที่มีโครงสร้างที่ซับซ้อนออกเป็นรีเลชันย่อย ๆ เป็นโครงสร้างที่ง่าย ซึ่งจะช่วยทำให้ไม่มีข้อมูลที่ซ้ำซ้อน และอยู่ในรูปแบบบรรทัดฐาน (Normal form) ที่สามารถนำไปใช้งานและไม่ก่อให้เกิดปัญหาใด ๆ ได้ (Codd, 1970)

กระบวนการปรับบรรทัดฐาน เป็นการดำเนินการอย่างเป็นลำดับที่กำหนดไว้ด้วยกันเป็นขั้นตอน แสดงดังรูปที่ 2.2 ตามปัญหาที่เกิดขึ้นในขั้นตอนนั้น ๆ ซึ่งเป็นกระบวนการที่ถูกนำเสนอโดยบอยส์ (Boyce) และ คอดด์ (Codd) (Date and Fagin, 1992) ซึ่งแต่ละขั้นตอนสามารถแบ่งออกเป็นระดับตามคุณสมบัติได้ดังนี้

1. รูปแบบบรรทัดฐานระดับที่หนึ่ง (First Normal Form: 1NF)
2. รูปแบบบรรทัดฐานระดับที่สอง (Second Normal Form: 2NF)
3. รูปแบบบรรทัดฐานระดับที่สาม (Third Normal Form: 3NF)
4. รูปแบบบรรทัดฐานบอยส์-คอดด์ (Boyce-Codd Normal Form: BCNF)
5. รูปแบบบรรทัดฐานระดับที่สี่ (Fourth Normal Form: 4NF)
6. รูปแบบบรรทัดฐานระดับที่ห้า (Fifth Normal Form: 5NF)

ในแต่ละขั้นตอนของกระบวนการปรับบรรทัดฐานจะมีการระบุรูปแบบโครงสร้างของข้อมูลที่เป็นคุณสมบัติของรูปแบบบรรทัดฐานของขั้นตอนนั้น ๆ ไว้ ซึ่งโครงสร้างที่ระบุนี้จะสามารถแก้ไขปัญหาที่เกิดขึ้นในโครงสร้างข้อมูลของขั้นตอนก่อนหน้าได้ หรืออาจกล่าวได้ว่ากระบวนการปรับบรรทัดฐานแต่ละขั้นตอนจะต้องอาศัยผลที่ได้จากกระบวนการปรับบรรทัดฐานในขั้นตอนก่อนหน้ามาปรับปรุงเพื่อให้มีโครงสร้างตามที่กำหนดไว้ในขั้นตอนนั้น ๆ แต่อย่างไรก็ตามกระบวนการปรับบรรทัดฐานไม่จำเป็นต้องเริ่มจากการทำให้อยู่ในรูปแบบบรรทัดฐานระดับที่หนึ่ง และสิ้นสุดที่ขั้นตอนการปรับให้อยู่ในรูปแบบบรรทัดฐานระดับที่ห้าเสมอไป กล่าวคือกระบวนการปรับบรรทัดฐานจะพิจารณาโครงสร้างของข้อมูลที่นำมาทำการปรับให้อยู่ในรูปแบบบรรทัดฐานนั้น ว่าจัดอยู่ในรูปแบบบรรทัดฐานขั้นตอนใด แล้วจึงเริ่มกระบวนการปรับบรรทัดฐานจากขั้นตอนนั้นเป็นต้นไป และเช่นเดียวกันในการพิจารณาว่าจะสิ้นสุดที่ขั้นตอนใด จะขึ้นอยู่กับโครงสร้างของข้อมูลที่ได้นั้นมีความถูกต้องตามความหมายของข้อมูลที่กำหนดไว้แล้วหรือไม่ ถ้าผลที่ได้จากกระบวนการปรับบรรทัดฐานในขั้นตอนใดส่งผลให้โครงสร้างของข้อมูลมีความหมายตามที่กำหมดไว้ กระบวนการปรับบรรทัดฐานก็จะสิ้นสุดที่ขั้นตอนนั้น แต่โดยปกติแล้วระบบฐานข้อมูลจะจัดเป็นระบบที่มีประสิทธิภาพ ถ้าทุกรีเลชันในระบบฐานข้อมูลอยู่ในรูปแบบบรรทัดฐานระดับที่สาม

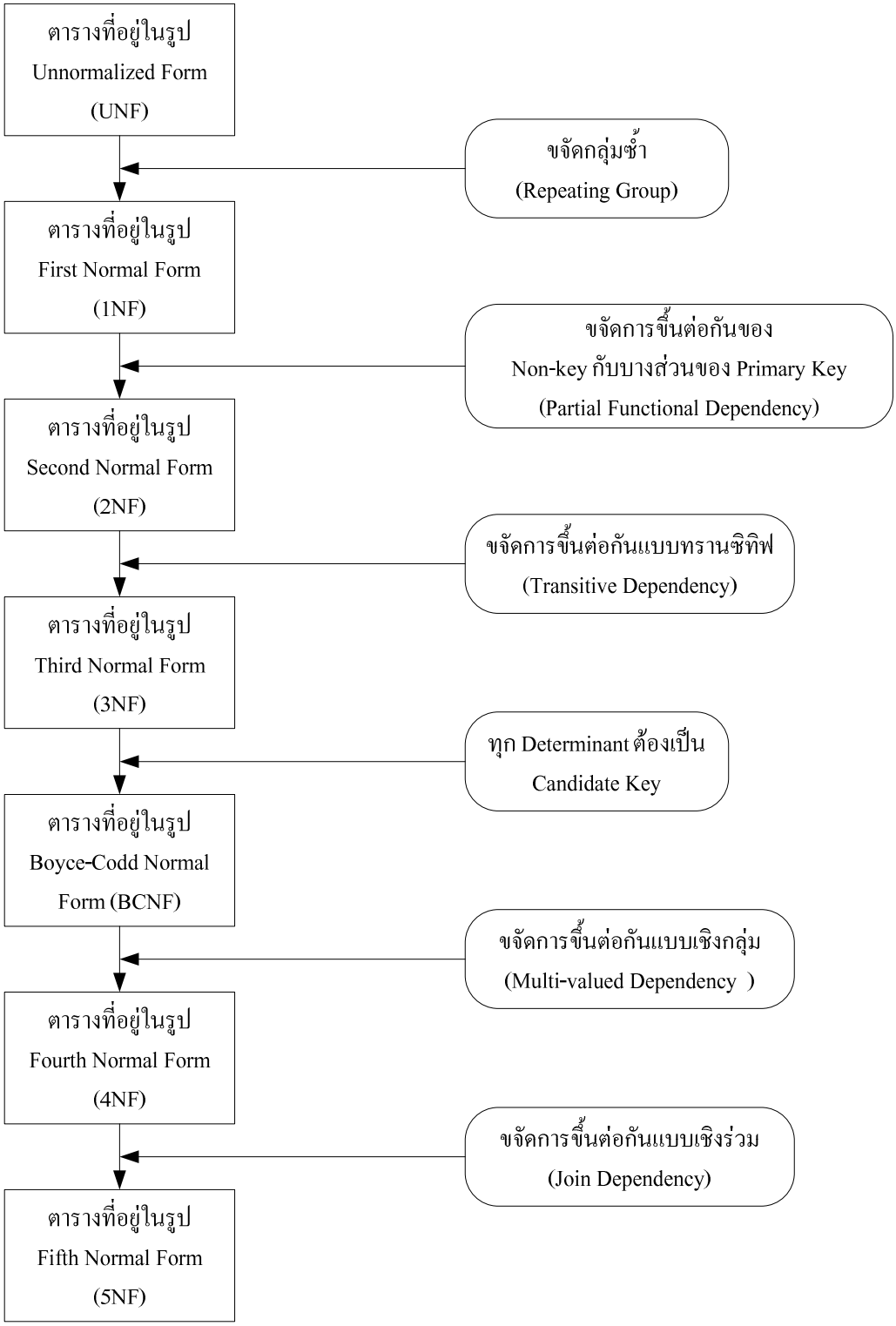
จุดประสงค์ของการปรับบรรทัดฐาน

1. ลดเนื้อที่ในการจัดเก็บข้อมูล

กระบวนการปรับบรรทัดฐานเป็นการออกแบบเพื่อลดความซ้ำซ้อนของข้อมูล ดังนั้นการลดความซ้ำซ้อนของข้อมูลย่อมลดเนื้อที่ในการจัดเก็บข้อมูลตามมาด้วย

2. ลดปัญหาข้อมูลที่ไม่ถูกต้อง

เมื่อข้อมูลไม่มีความซ้ำซ้อน การปรับเปลี่ยนแปลงแก้ไขข้อมูลก็สามารถทำได้จากแหล่งข้อมูลเพียงแหล่งเดียว จึงช่วยลดปัญหาการเปลี่ยนแปลงข้อมูลไม่ถูกต้องได้ (Inconsistency)



รูปที่ 2.2 ขั้นตอนการปรับบรรทัดฐาน

2.3 ฟังก์ชันการขึ้นต่อกัน (Functional Dependency: FD)

การพิจารณาโครงสร้างของแต่ละรีเลชันว่ามีโครงสร้างอยู่ในรูปแบบบรรทัดฐานระดับใด จะพิจารณาจากฟังก์ชันการขึ้นต่อกัน ซึ่งเป็นความสัมพันธ์ระหว่างแอททริบิวท์ต่าง ๆ ภายในรีเลชันกับแอททริบิวท์หรือกลุ่มของแอททริบิวท์ที่ทำหน้าที่เป็นคีย์ของรีเลชัน ซึ่งความสัมพันธ์นี้จะถูกนิยามโดยรูปแบบทางคณิตศาสตร์ที่เรียกว่า ฟังก์ชันการขึ้นต่อกัน (Armstrong, 1974) ซึ่งมีรูปแบบดังนี้

FD: Determinant-attribute $\longrightarrow$ Dependency-attribute

โดยที่ Determinant-attribute หมายถึง แอททริบิวท์ที่ระบุค่าใดค่าหนึ่งแล้ว จะสามารถแสดงค่าของ Dependency-attribute ซึ่งเป็นแอททริบิวท์ที่มีความสัมพันธ์กับ Determinant-attribute นั้นออกมา ดังตัวอย่างในตารางที่ 2.2

ตารางที่ 2.2 ตารางแสดงความสัมพันธ์ระหว่างหมายเลขบัตรประชาชน และชื่อเจ้าของบัตร

Personal_ID	**Personal_Name**
320500116321	ณัฐพล
1320100453011	วิภาศิริ
3120512450182	จรูญ
3120356781201	ทัศนีย์
1208130303111	อนุพงศ์
1105102630213	ทัศนีย์

จากตารางจะสังเกตเห็นว่า เมื่อมีการระบุค่าหมายเลขบัตรประชาชน (Personal_ID) ด้วยหมายเลขใดหมายเลขหนึ่ง จะสามารถทราบถึงชื่อของเจ้าของบัตรประชาชน (Personal_Name) ตามหมายเลขที่ระบุนั้นได้ เช่น เมื่อระบุหมายเลขบัตรประชาชนเป็น “1320500116321” จะได้ชื่อเจ้าของบัตร คือ “ณัฐพล” หรือเมื่อระบุหมายเลขบัตรประชาชนเป็น “1320100453011” จะได้ชื่อเจ้าของบัตร คือ “วิภาศิริ” เป็นต้น แต่ในทางกลับกัน เมื่อมีการระบุชื่อเจ้าของบัตรประชนชน กลับไม่สามารถทราบถึงหมายเลขบัตรประชาชนตามชื่อที่ระบุนั้นได้ในทุกกรณี เนื่องจากมีชื่อเจ้าของบัตรประชาชนที่ซ้ำกัน เช่น เมื่อระบุชื่อเจ้าของบัตรเป็น “ทัศนีย์” จะไม่สามารถทราบถึงหมายเลขบัตรประชาชนที่แน่ชัดได้ เนื่องจากบุคคลที่ชื่อ “ทัศนีย์” ในตารางมีอยู่ 2 คน ได้แก่ บัตรประชาชนหมายเลข “3120356781201” และ “1105102630213” เป็นต้น ดังนั้นจากตารางดังกล่าวสามารถ

กล่าวได้ว่า หมายเลขบัตรประชาชน เป็น Determinant-attribute และชื่อเจ้าของบัตรประชาชน เป็น Dependency-attribute ซึ่งสามารถเขียนในรูปฟังก์ชันการขึ้นต่อกันได้ดังนี้

FD: Personal_ID $\longrightarrow$ Personal_Name

โดยทั่วไปแล้วจะสามารถแบ่งประเภทการขึ้นต่อกันได้เป็น 3 ประเภท ดังต่อไปนี้

2.3.1 การขึ้นต่อกันอย่างสมบูรณ์ (Complete dependency)

การขึ้นต่อกันอย่างสมบูรณ์เป็นรูปแบบการขึ้นต่อกัน ที่แอททริบิวท์ที่มิใช่คีย์หลักของรีเลชัน (Nonkey attribute) มีการขึ้นต่อแอททริบิวท์หรือกลุ่มของแอททริบิวท์ที่ทำหน้าที่เป็นคีย์หลัก (Primary key) ของรีเลชัน

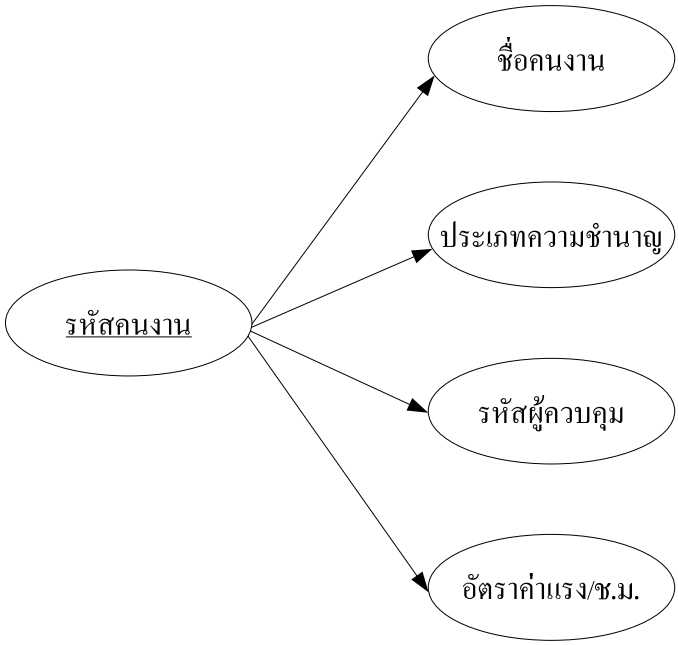
ตัวอย่างที่ 1 กำหนดให้รีเลชันคนงาน ประกอบด้วยแอททริบิวท์ต่าง ๆ ดังนี้

คนงาน (รหัสคนงาน, ชื่อคนงาน, ประเภทความชำนาญ, รหัสผู้ควบคุม, อัตราค่าแรง/ช.ม.)

รีเลชันคนงาน จะมีแอททริบิวท์รหัสคนงาน ที่มีข้อมูลไม่ซ้ำกันและใช้เป็นคีย์หลักของรีเลชัน ดังนั้น ถ้ามีการอ้างอิงถึงรหัสคนงานเพียงหนึ่งรหัส ก็จะทำให้ทราบถึงข้อมูลที่มีอยู่ภายในแอททริบิวท์อื่น ๆ ของคนงานที่มีรหัสนั้นได้ ตัวอย่างเช่น ถ้าอ้างถึงคนงานที่มีรหัสเป็น “1245” จะทำให้ทราบว่าเป็นคนงานที่ชื่อ “สุดใจ” ที่มีประเภทความชำนาญด้าน “ไฟฟ้า” มีรหัสผู้ควบคุมหมายเลข “1441” และมีอัตราค่าแรงต่อชั่วโมงเป็น “16 บาท/ช.ม.”

จึงกล่าวได้ว่าแอททริบิวท์รหัสคนงาน จะถูกใช้ในการเลือก หรือเป็นตัวเลือกข้อมูลที่อยู่ในแอททริบิวท์ชื่อคนงาน ประเภทความชำนาญ รหัสผู้ควบคุม และอัตราค่าแรง/ช.ม. จากรีเลชันคนงาน หรือกล่าวอีกนัยหนึ่งคือ แอททริบิวท์ชื่อคนงาน ประเภทความชำนาญ รหัสผู้ควบคุม และอัตราค่าแรง/ช.ม. จะขึ้นกับแอททริบิวท์รหัสคนงาน อย่างสมบูรณ์นั่นเอง ซึ่งสามารถแสดงด้วยแผนภาพได้ดังรูปที่ 2.3 หรือเขียนแทนด้วยข้อความสัญลักษณ์ได้ดังนี้

รหัสคนงาน $\longrightarrow$ ชื่อคนงาน, ประเภทความชำนาญ, รหัสผู้ควบคุม, อัตราค่าแรง/ช.ม.



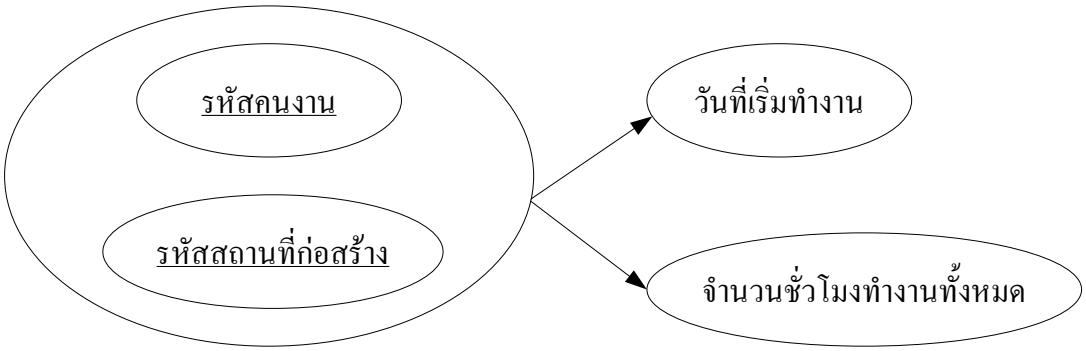
รูปที่ 2.3 การขึ้นต่อกันอย่างสมบูรณ์ของแอททริบิวท์ในรีเลชันคนงาน

ตัวอย่างที่ 2 กำหนดให้รีเลชันการทำงาน ประกอบด้วยแอททริบิวท์ต่าง ๆ ดังนี้

การทำงาน (รหัสคนงาน, รหัสสถานที่ก่อสร้าง, วันที่เริ่มทำงาน,
จำนวนชั่วโมงที่ทำงานทั้งหมด)

รีเลชันนี้จะมีแอททริบิวท์รหัสคนงาน และรหัสสถานที่ก่อสร้าง ร่วมกันทำหน้าที่เป็นคีย์หลักของรีเลชัน ดังนั้น การอ้างถึง รหัสคนงาน หนึ่ง ๆ และรหัสสถานที่ก่อสร้าง ที่คนงานคนนั้นทำงานอยู่ จะทำให้ทราบถึงข้อมูลในแอททริบิวท์วันที่เริ่มทำงาน และจำนวนชั่วโมงที่ทำงานทั้งหมด ของคนงานรหัสนั้น จึงกล่าวได้ว่า รหัสคนงาน และรหัสสถานที่ก่อสร้าง จะถูกใช้ในการเลือก หรือเป็นตัวเลือก ข้อมูลที่อยู่ในแอททริบิวท์วันที่เริ่มทำงาน และจำนวนชั่วโมงที่ทำงานทั้งหมด อย่างสมบูรณ์ หรือกล่าวอีกนัยหนึ่งคือ แอททริบิวท์วันที่เริ่มทำงาน และจำนวนชั่วโมงที่ทำงานทั้งหมด จะขึ้นกับแอททริบิวท์รหัสคนงาน และรหัสสถานที่ก่อสร้าง อย่างสมบูรณ์ ซึ่งสามารถแสดงด้วยแผนภาพได้ดังรูปที่ 2.4 หรือเขียนแทนด้วยข้อความสัญลักษณ์ดังนี้

รหัสคนงาน, รหัสสถานที่ก่อสร้าง $\rightarrow$ วันที่เริ่มทำงาน,
จำนวนชั่วโมงที่ทำงาน ทั้งหมด



รูปที่ 2.4 การขึ้นต่อกันอย่างสมบูร์ของแอททริบิวท์ในรีเลชันการทำงาน

2.3.2 การขึ้นต่อกันเพียงบางส่วน (Partial dependency)

การขึ้นต่อกันเพียงบางส่วนเป็นการขึ้นต่อกัน ที่แอททริบิวท์ที่ไม่ใช่คีย์หลักของรีเลชัน มีการขึ้นต่อแอททริบิวท์บางตัวที่ร่วมเป็นคีย์หลักของรีเลชัน

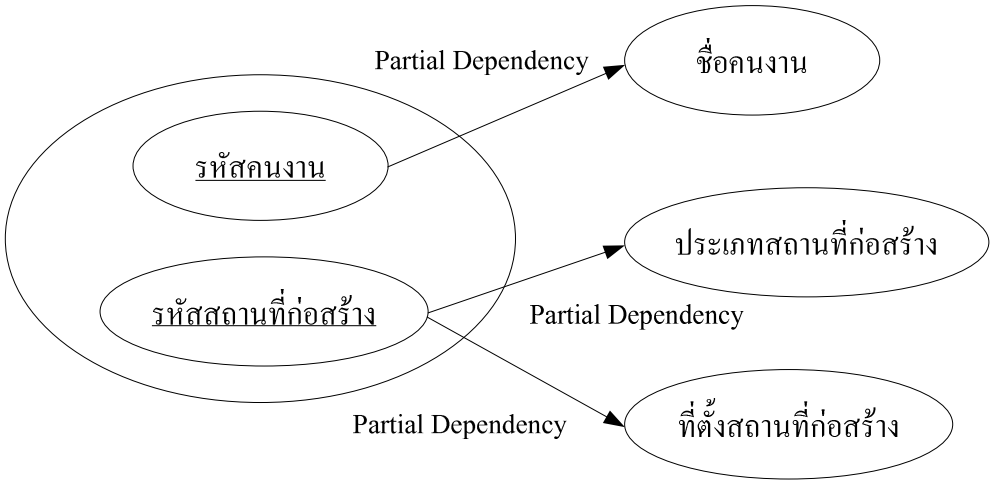
ตัวอย่าง กำหนดให้รีเลชันการทำงานตามสถานที่ ของคนงานมีสคีมา ดังนี้

การทำงานตามสถานที่ (รหัสคนงาน, ชื่อคนงาน, รหัสสถานที่ก่อสร้าง,
ประเภทสถานที่ก่อสร้าง, ที่ตั้งสถานที่ก่อสร้าง)

รีเลชันการทำงานตามสถานที่ จะมีแอททริบิวท์รหัสคนงาน และรหัสสถานที่ก่อสร้าง ร่วมกันเป็นคีย์หลักของรีเลชัน โดยมี ชื่อคนงาน เป็นแอททริบิวท์ที่ขึ้นกับแอททริบิวท์รหัสคนงาน ซึ่งเป็นเพียงบางส่วนของคีย์หลัก นอกจากนี้แอททริบิวท์ประเภทสถานที่ก่อสร้าง และที่ตั้งสถานที่ก่อสร้าง ก็เป็นแอททริบิวท์ที่ขึ้นกับแอททริบิวท์รหัสสถานที่ก่อสร้าง ซึ่งเป็นเพียงบางส่วนของคีย์หลักเช่นกัน ซึ่งสามารถแสดงด้วยแผนภาพได้ดังรูปที่ 2.5 หรือเขียนแทนด้วยข้อความสัญลักษณ์ดังนี้

รหัสคนงาน $\longrightarrow$ ชื่อคนงาน

รหัสสถานที่ก่อสร้าง $\longrightarrow$ ประเภทสถานที่ก่อสร้าง, ที่ตั้งสถานที่ก่อสร้าง



รูปที่ 2.5 การขึ้นต่อกันเพียงบางส่วนของแอททริบิวท์ในรีเลชันการทำงานตามสถานที่

2.3.3 การขึ้นต่อกันแบบทรานซิทีฟ (Transitive dependency)

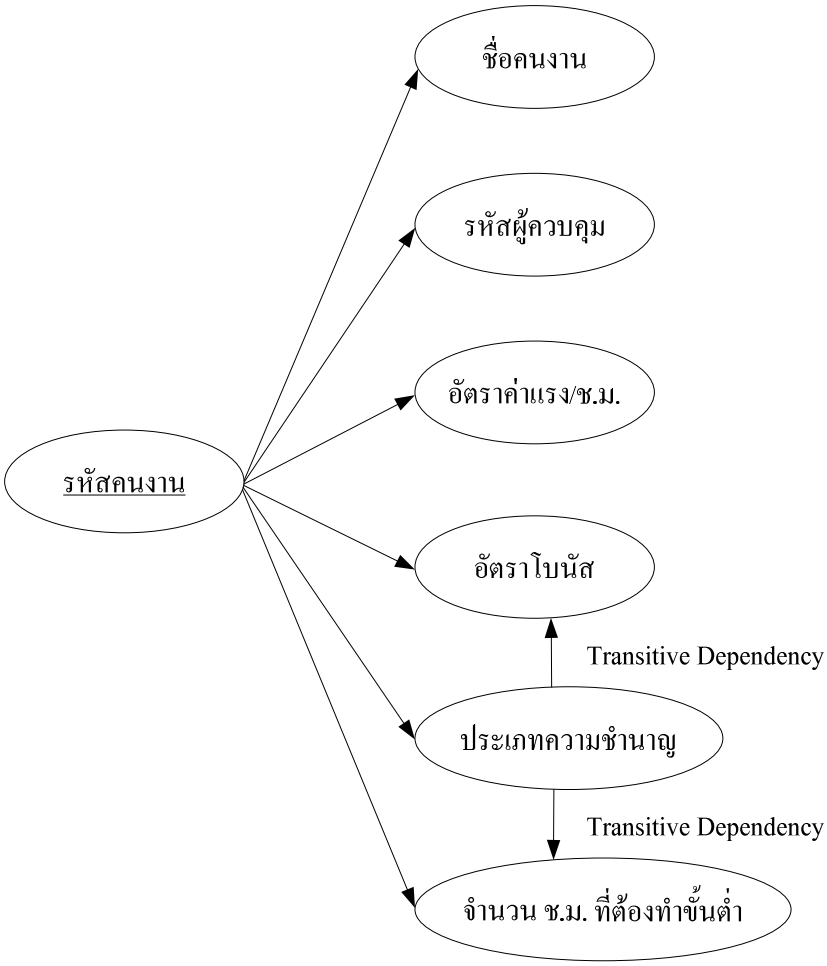
การขึ้นต่อกันแบบทรานซิทีฟเป็นรูปแบบการขึ้นต่อกัน ที่แอททริบิวท์ที่ไม่ใช่คีย์หลักของรีเลชัน มีการขึ้นกับแอททริบิวท์อื่นที่ไม่ได้เป็นคีย์หลักของรีเลชัน

ตัวอย่าง กำหนดให้รีเลชันคนงานตามความชำนาญ มีสคีมาดังต่อไปนี้

คนงานตามความชำนาญ (รหัสคนงาน, ชื่อคนงาน, ประเภทความชำนาญ, อัตราโบนัส, จำนวนช.ม.ที่ต้องทำขั้นต่ำ, รหัสผู้ควบคุม, อัตราค่าแรง/ช.ม.)

รีเลชันคนงานตามความชำนาญ นี้จะมีแอททริบิวท์รหัสคนงาน ทำหน้าที่เป็นคีย์หลักของรีเลชัน โดยมีแอททริบิวท์ชื่อคนงาน ประเภทความชำนาญ จำนวน ช.ม. ที่ต้องทำขั้นต่ำ อัตราโบนัส รหัสผู้ควบคุม และอัตราค่าแรง/ช.ม. ที่ขึ้นกับแอททริบิวท์รหัสคนงาน นี้อย่างสมบูรณ์ แต่ก็ยังคงมีบางแอททริบิวท์ เช่น แอททริบิวท์อัตราโบนัส และจำนวน ช.ม. ที่ต้องทำขั้นต่ำ ที่ขึ้นกับแอททริบิวท์ประเภทความชำนาญ ซึ่งทั้งแอททริบิวท์อัตราโบนัส จำนวน ช.ม. ที่ต้องทำขั้นต่ำ และประเภทความชำนาญ ต่างก็ไม่ใช่คีย์หลักหรือเป็นส่วนหนึ่งของคีย์หลัก จึงเรียกการขึ้นต่อกันระหว่างแอททริบิวท์ที่ไม่ใช่คีย์นี้ว่า การขึ้นต่อกันแบบทรานซิทีฟ ซึ่งสามารถแสดงด้วยแผนภาพได้ดังรูปที่ 2.6 หรือเขียนแทนด้วยข้อความสัญลักษณ์ดังนี้

ประเภทความชำนาญ $\rightarrow$ อัตราโบนัส, จำนวน ช.ม. ขั้นต่ำที่ต้องทำ



รูปที่ 2.6 การขึ้นต่อกันแบบทรานซิทีฟของแอททริบิวท์ในรีเลชันคนงานตามความชำนาญ

2.4 รูปแบบบรรทัดฐานระดับที่ 1

กระบวนการปรับบรรทัดฐานระดับแรกสุด จะเป็นกระบวนการในการปรับตารางข้อมูลของผู้ใช้ให้อยู่ในรูปแบบของรีเลชัน กล่าวคือ ค่าของแอททริบิวท์ของแต่ละแถวสามารถมีได้ค่าเดียวเท่านั้น เพื่อให้อยู่ในรูปแบบบรรทัดฐานระดับที่ 1

นิยาม: รีเลชันใดจะอยู่ในรูปแบบบรรทัดฐานระดับที่ 1 ได้ก็ต่อเมื่อ รีเลชันนั้นจะต้องมีคุณสมบัติต่อไปนี้

1. เป็นรีเลชันที่มีคีย์หลักของรีเลชัน
2. ไม่มีกลุ่มซ้ำอยู่ในรีเลชัน หรืออาจกล่าวได้ว่าค่าของแต่ละแอททริบิวท์ของแต่ละแถวสามารถมีได้ค่าเดียวเท่านั้น

3. แอททริบิวท์ทุกแอททริบิวท์ที่ไม่ใช่คีย์ จะต้องขึ้นกับแอททริบิวท์ที่เป็นคีย์หลักอย่างสมบูรณ์

2.4.1 การปรับรีเลชันที่ไม่เป็นบรรทัดฐานให้เป็นรีเลชันที่มีรูปแบบบรรทัดฐานระดับที่ 1

ในขั้นตอนการปรับรีเลชันที่ไม่เป็นบรรทัดฐานให้เป็นรีเลชันที่มีรูปแบบบรรทัดฐานระดับที่ 1 นี้ จะต้องทำการกำจัดกลุ่มซ้ำออกไป โดยตัวอย่างการปรับรีเลชันให้อยู่ในรูปแบบบรรทัดฐานระดับที่ 1 สามารถอธิบายได้ดังตัวอย่างต่อไปนี้

ตารางที่ 2.3 ตารางการสั่งสินค้าที่เป็น Unnormalized form

เลขที่ใบสั่ง	วันที่สั่ง	รหัสลูกค้า	ชื่อลูกค้า	รหัสสินค้า	ชื่อสินค้า	จำนวน
INV001	20-08-2008	C001	สาทร	CA101	CPU	20
				MB205	Main broad	40
INV002	20-08-2008	C002	บุญมี	RA300	Ram	10
INV003	21-08-2008	C003	บุญมา	MB205	Main broad	30
				HD101	Hard disk	25
				SG201	Sound Card	15

พิจารณาตารางที่ 2.3 จะเห็นว่าเลขที่ใบสั่ง 1 รหัส ประกอบไปด้วยสินค้ามากกว่า 1 รายการ จึงกล่าวได้ว่า แอททริบิวท์รหัสการสินค้า ชื่อสินค้า และจำนวน เป็นกลุ่มซ้ำของรีเลชัน ซึ่งไม่ตรงกับคุณสมบัติของรูปแบบบรรทัดฐานระดับที่ 1 ดังนั้นการปรับในระดับนี้ก็ได้แก่การขจัดกลุ่มซ้ำออกไป ทำให้ได้ผลลัพธ์ดังตารางที่ 2.4

ตารางที่ 2.4 ตารางการสั่งสินค้าที่เป็น 1NF

เลขที่ใบสั่ง	วันที่สั่ง	รหัสลูกค้า	ชื่อลูกค้า	รหัสสินค้า	ชื่อสินค้า	จำนวน
INV001	20-08-2008	C001	สาทร	CA101	CPU	20
INV001	20-08-2008	C001	สาทร	MB205	Main broad	40
INV002	20-08-2008	C002	บุญมี	RA300	Ram	10
INV003	21-08-2008	C003	บุญมา	MB205	Main broad	30
INV003	21-08-2008	C003	บุญมา	HD101	Hard disk	25
INV003	21-08-2008	C003	บุญมา	SG201	Sound Card	15

วิธีการขจัดกลุ่มซ้ำสามารถทำได้โดยแยกข้อมูลของเลขที่ใบสั่งซื้อ “INV001” ออกเป็น 2 แถว ดังแสดงในตารางที่ 2.4 ซึ่งผลที่ได้นี้จะเห็นว่าเลขที่ใบสั่งไม่ใช่คีย์หลักของรีเลชันอีกต่อไป แต่คีย์หลักจะประกอบด้วยเลขที่ใบสั่ง และรหัสสินค้า ซึ่งเราอาจกล่าวได้ว่าการปรับบรรทัดฐานให้อยู่ในรูปแบบบรรทัดฐานระดับที่ 1 นั้นจะต้องมีการเพิ่มแอททริบิวท์ที่มาเป็นคีย์หลักเสมอ โดยคีย์หลักตัวใหม่จะประกอบไปด้วยคีย์ตัวเดิมบวกกับแอททริบิวท์ที่เป็นคีย์หลักของกลุ่มซ้ำ

2.4.2 ปัญหาที่อาจเกิดขึ้นกับรีเลชันที่มีรูปแบบบรรทัดฐานระดับที่ 1

ถึงแม้ว่ารีเลชันจะถูกออกแบบหรือถูกปรับให้อยู่ในรูปแบบบรรทัดฐานแล้ว แต่จากลักษณะข้อมูลภายในรีเลชันบางรีเลชันที่อยู่ในรูปแบบบรรทัดฐานระดับที่ 1 ก็อาจก่อให้เกิดปัญหาขึ้นได้อีก เช่น รีเลชันการสั่งสินค้าที่เป็น 1NF แม้จะอยู่ในรูปแบบบรรทัดฐานระดับที่ 1 แต่จะเห็นว่ามีข้อมูลบางแถวที่มีค่าของข้อมูลซ้ำกัน เช่น ข้อมูลในแถวที่ 1 และ 2 มีแอททริบิวท์ที่เก็บข้อมูลซ้ำกัน คือ เลขที่ใบสั่ง วันที่สั่ง รหัสลูกค้า และชื่อลูกค้า เป็นต้น

ในการเก็บข้อมูลที่ซ้ำซ้อนกันนี้นอกจากจะทำให้ต้องสิ้นเปลืองเนื้อที่ในการจัดเก็บข้อมูลแล้ว ยังก่อให้เกิดปัญหาตามมาอีกหลายข้อดังต่อไปนี้

1. *ความผิดปกติต่อการแก้ไขข้อมูล (Update anomaly)*

ตัวอย่างเช่น ในรีเลชันการสั่งสินค้า ถ้าต้องการเปลี่ยนแปลงชื่อของลูกค้าที่มีรหัสลูกค้า “C001” จากเดิมชื่อ “สาทร” เป็น “สิทธิชัย” ก็ต้องทำการเปลี่ยนข้อมูลในหลายแถวที่มีรหัสลูกค้าเป็น “C001” ซึ่งนอกจากจะต้องใช้เวลาในการแก้ไขแล้ว ก็ยังไม่สามารถรับประกันได้ว่ามีการแก้ไขได้หมดครบทุกแถว และถ้ายิ่งมีการแก้ไขไม่ครบก็อาจทำให้ข้อมูลในรีเลชันเกิดความขัดแย้ง (Inconsistency) กันขึ้น เนื่องจากมีบางแถวที่มีรหัสลูกค้าเหมือนกัน แต่มีชื่อของลูกค้าแตกต่างกัน เป็นต้น

2. *ความผิดปกติต่อการลบข้อมูล (Delete anomaly)*

ถ้าหากต้องการลบลูกค้าที่ชื่อ “บุญมี” ออกไปจากรีเลชัน ก็จะทำให้ข้อมูลในแถวที่ 3 ทั้งหมดถูกลบทิ้ง แต่ก็พบว่าการลบข้อมูลในแถวที่ 3 ทิ้งไปจะทำให้ต้องสูญเสียข้อมูลเกี่ยวกับสินค้าที่มีรหัสเป็น “RA300” ไปด้วย เนื่องจากไม่ได้มีการอ้างอิงถึงรหัสนี้อีกในแถวอื่น ๆ

3. *ความผิดปกติต่อการเพิ่มข้อมูล (Insert anomaly)*

หากต้องการเพิ่มรายชื่อลูกค้าคนอื่น ๆ เข้าไปในรีเลชันการสั่งสินค้า โดยที่ลูกค้ายังไม่มีรายการสั่งสินค้าจะไม่สามารถทำได้ เนื่องจากคีย์หลักของรีเลชันการสั่งสินค้าประกอบไปด้วย แอททริบิวท์เลขที่ใบสั่ง และรหัสสินค้า ซึ่งจากคุณสมบัติของคีย์หลักจะไม่ยอมให้มีค่าว่างเกิดขึ้นในแอททริบิวท์ที่เป็นคีย์หลัก

ดังนั้นจึงเห็นได้ว่า แม้รีเลชันจะอยู่ในรูปแบบบรรทัดฐานระดับที่ 1 แล้ว แต่บางรีเลชันก็ยังมีความซ้ำซ้อนของข้อมูลและก่อให้เกิดปัญหาได้อีก จึงต้องมีกระบวนการปรับบรรทัดฐานในระดับต่อไป

2.5 รูปแบบบรรทัดฐานระดับที่ 2

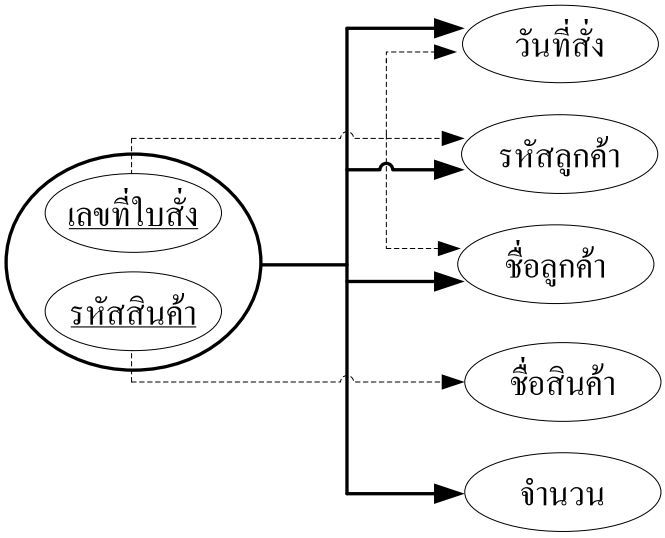
รูปแบบบรรทัดฐานระดับที่ 2 นี้จะเกี่ยวกับความสัมพันธ์ระหว่างคีย์หลักและแอททริบิวท์อื่นที่ไม่ได้เป็นส่วนใดส่วนหนึ่งของคีย์หลัก

Codd (1971) ได้ให้นิยามไว้ว่า รีเลชันใดจะอยู่ในรูปแบบบรรทัดฐานระดับที่ 2 ก็ต่อเมื่อรีเลชันนั้นจะต้องมีคุณสมบัติต่อไปนี้

1. รีเลชันนั้นอยู่ในรูปแบบบรรทัดฐานระดับที่ 1
2. ต้องไม่มีการขึ้นต่อกันเพียงบางส่วน กล่าวคือ ต้องไม่มีแอททริบิวท์ที่ไม่ใช่คีย์หลักตัวใดขึ้นกับส่วนใดส่วนหนึ่งของคีย์หลัก

2.5.1 การปรับรีเลชันรูปแบบบรรทัดฐานระดับที่ 1 เป็นระดับที่ 2

พิจารณารีเลชันการสั่งสินค้า ในตารางที่ 2.4 จะพบว่าคีย์หลักของรีเลชันประกอบไปด้วยแอททริบิวท์สองแอททริบิวท์ ได้แก่ เลขที่ใบสั่ง และรหัสสินค้า แต่จะมีแอททริบิวท์ชื่อสินค้า ซึ่งไม่ใช่คีย์ขึ้นอยู่กับแอททริบิวท์รหัสสินค้า ที่เป็นส่วนหนึ่งของคีย์หลักเพียงอย่างเดียว จากตารางที่ 2.4 สามารถเขียนแผนภาพของฟังก์ชันการขึ้นต่อกันได้ดังรูปที่ 2.7



รูปที่ 2.7 การขึ้นต่อกันของแอททริบิวท์ในรีเลชันการสั่งสินค้า

พิจารณารูปที่ 2.7 จะเห็นว่าแอททริบิวท์วันที่สั่ง รหัสลูกค้า และชื่อลูกค้า ขึ้นอยู่กับแอททริบิวท์เลขที่ใบสั่ง ซึ่งเป็นเพียงส่วนหนึ่งของคีย์หลัก และแอททริบิวท์ชื่อสินค้า ก็ขึ้นอยู่กับรหัสสินค้า ซึ่งเป็นเพียงส่วนหนึ่งของคีย์หลักเช่นกัน สองส่วนนี่เองที่เป็นการขึ้นต่อกันเพียงบางส่วนของรีเลชัน จึงทำให้รีเลชันขาดคุณสมบัติของรูปแบบบรรทัดฐานระดับที่ 2 ทำให้ต้องมีการปรับรีเลชันใหม่

โดยวิธีที่จะทำให้รีเลชันอยู่ในรูปแบบบรรทัดฐานระดับที่ 2 กระทำได้โดยการสร้างรีเลชันขึ้นมาใหม่สำหรับการขึ้นต่อกันที่เป็นปัญหา ซึ่งจะทำให้ได้รีเลชันย่อยเกิดขึ้น ดังแสดงในตารางที่ 2.5-2.7

ตารางที่ 2.5 ตารางการสั่งสินค้าที่ปรับปรุงจากตารางที่ 2.4

เลขที่ใบสั่ง	**วันที่สั่ง**	**รหัสลูกค้า**	**ชื่อลูกค้า**
INV001	20-08-2008	C001	สาทร
INV002	20-08-2008	C002	บุญมี
INV003	21-08-2008	C003	บุญมา

ตารางที่ 2.6 ตารางสินค้าที่ปรับปรุงจากตารางที่ 2.4

รหัสสินค้า	**ชื่อสินค้า**
CA101	CPU
MB205	Main broad
RA300	Ram
HD101	Hard disk
SG201	Sound Card

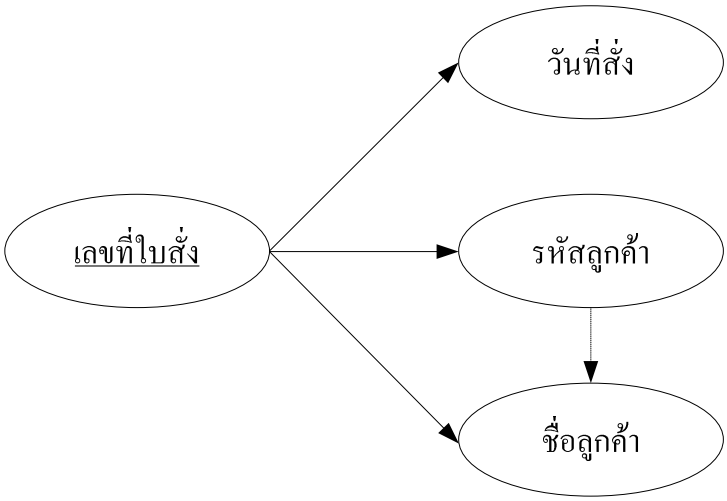
ตารางที่ 2.7 ตารางรายการสั่งสินค้าที่ปรับปรุงจากตารางที่ 2.4

เลขที่ใบสั่ง	**รหัสสินค้า**	**จำนวน**
INV001	CA101	20
INV001	MB205	40
INV002	RA300	10
INV003	MB205	30
INV003	HD101	25
INV003	SG201	15

จากตารางที่ 2.5-2.7 สามารถเขียนแทนด้วยข้อความสัญลักษณ์แทนรีเลชันได้ดังนี้

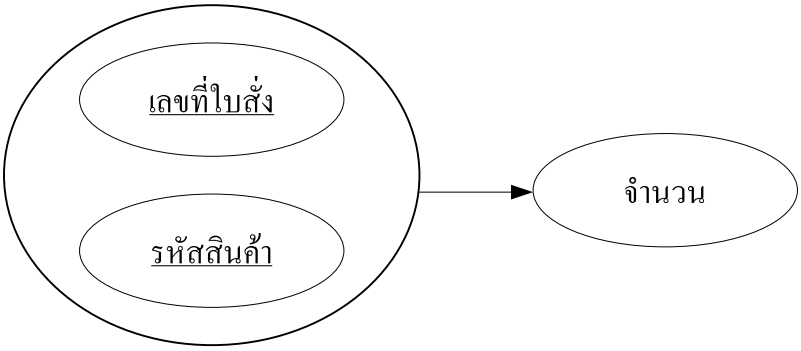
การสั่งสินค้า (เลขที่ใบสั่ง, วันที่สั่ง, รหัสลูกค้า, ชื่อลูกค้า)
สินค้า (รหัสสินค้า, ชื่อสินค้า)
รายการสั่งสินค้า (เลขที่ใบสั่ง, รหัสสินค้า, จำนวน)

หากนำแต่ละรีเลชันมาเขียนเป็นแผนภาพการขึ้นต่อกัน จะสามารถเขียนได้ดังแสดงในรูปที่ 2.8-2.10



รูปที่ 2.8 การขึ้นต่อกันของแอททริบิวท์ในรีเลชันการสั่งสินค้าหลังจากปรับรูปแบบบรรทัดฐาน

รูปที่ 2.9 การขึ้นต่อกันของแอททริบิวท์ในรีเลชันสินค้าหลังจากปรับรูปแบบบรรทัดฐาน



รูปที่ 2.10 การขึ้นต่อกันของแอททริบิวท์ในรีเลชันรายการสั่งสินค้า
หลังจากปรับรูปแบบบรรทัดฐาน

2.5.2 ปัญหาที่อาจเกิดขึ้นกับรีเลชันที่มีรูปแบบบรรทัดฐานระดับที่ 2

หากพิจารณารีเลชันในรูปที่ 2.8-2.10 จะเห็นว่าทุกรีเลชันเป็นรีเลชันที่อยู่ในรูปแบบบรรทัดฐานระดับที่ 1 และ 2 แล้ว เนื่องจากทุกแอททริบิวท์จะขึ้นอยู่กับคีย์หลักของรีเลชันเท่านั้น แต่ถ้าพิจารณาแต่ละรีเลชันอย่างละเอียดแล้ว จะพบว่า บางรีเลชันมีแอททริบิวท์ที่ขึ้นอยู่กับแอททริบิวท์อื่นที่ไม่ได้เป็นคีย์หลักของรีเลชันร่วมอยู่ด้วย เช่น รีเลชันการสั่งสินค้า ในรูปที่ 2.8 จะมีแอททริบิวท์ชื่อลูกค้า ซึ่งเป็นแอททริบิวท์ที่ไม่ใช่คีย์ที่ขึ้นอยู่กับ รหัสลูกค้า ที่เป็นแอทริบิวท์ที่ไม่ใช่คีย์เช่นกัน ซึ่งเป็นการขึ้นต่อกันแบบทรานซิทีฟ ส่งผลให้เกิดปัญหาความผิดปกติต่อการเพิ่มข้อมูล คือหากต้องการเพิ่มรายชื่อลูกค้าคนอื่น ๆ เข้าไปในรีเลชันการสั่งสินค้า โดยที่ลูกค้ายังไม่มีรายการสั่งสินค้าจะไม่สามารถทำได้ เนื่องจากคีย์หลักของรีเลชันการสั่งสินค้า คือ คีย์ผสมที่ประกอบไปด้วย แอททริบิวท์เลขที่ใบสั่ง และรหัสสินค้า ซึ่งจากคุณสมบัติของคีย์หลักจะไม่ยอมให้มีค่าว่างในแอททริบิวท์ที่เป็นคีย์หลัก

2.6 รูปแบบบรรทัดฐานระดับที่ 3

รีเลชันที่มีรูปแบบบรรทัดฐานระดับที่ 3 จะเป็นรีเลชันที่มีนิยามดังต่อไปนี้

นิยาม: รีเลชันใดจะอยู่ในรูปแบบบรรทัดฐานระดับที่ 3 ก็ต่อเมื่อ รีเลชันนั้นจะต้องมีคุณสมบัติต่อไปนี้

1. รีเลชันนั้นอยู่ในรูปแบบบรรทัดฐานระดับที่ 2
2. จะต้องไม่มีการขึ้นต่อกันแบบทรานซิทีฟกล่าวคือ ต้องไม่มีแอททริบิวท์ที่ไม่ใช่คีย์หลักตัวใดขึ้นกับแอททริบิวท์อื่น ซึ่งเป็นแอททริบิวท์ที่ไม่ใช่คีย์หลักเช่นกัน

2.6.1 การปรับรีเลชันรูปแบบบรรทัดฐานระดับที่ 2 เป็นระดับที่ 3

พิจารณารูปที่ 2.8 จะเห็นว่ายังคงมีปัญหาการขึ้นต่อกันอยู่ จึงจำเป็นต้องทำการสร้างรีเลชันขึ้นมาใหม่เพื่อแก้ไขปัญหาที่เกิดขึ้น ในการปรับรีเลชันที่อยู่ในรูปแบบบรรทัดฐานระดับที่ 2 ให้อยู่ในรูปแบบบรรทัดฐานระดับที่ 3 มีขั้นตอนดังต่อไปนี้

1. ทำการแยกแอททริบิวท์ที่มีการขึ้นต่อกันแบบทรานซิทีฟ ออกมาเป็นรีเลชันใหม่อีกหนึ่งรีเลชัน พร้อมทั้งกำหนดให้แอททริบิวท์ที่เป็นตัวเลือก (Determinant) ทำหน้าที่เป็นคีย์หลักของรีเลชันใหม่นั้นด้วย
2. แอททริบิวท์ที่เป็นตัวเลือกนี้จะยังคงถูกเก็บอยู่ในรีเลชันเดิมด้วย เพื่อทำหน้าที่เป็นคีย์นอก (Foreign key) เชื่อมโยงไปยังคีย์หลักของรีเลชันใหม่อีกที

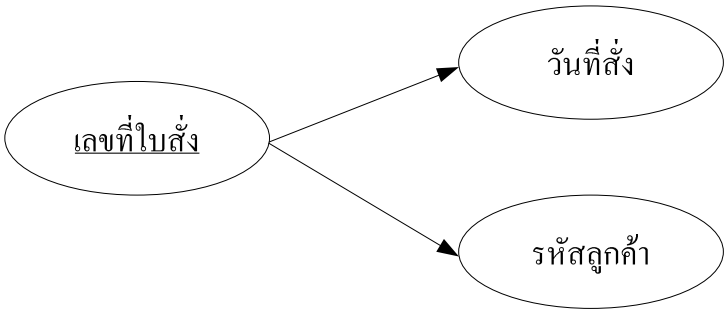
ดังนั้นรีเลชันการสั่งสินค้า ตามตารางที่ 2.5 จะต้องถูกแยกออกเป็น 2 รีเลชัน ได้แก่ รีเลชันการสั่ง และรีเลชันลูกค้า ซึ่งแสดงตัวอย่างข้อมูลที่ถูกเก็บอยู่ในแต่ละรีเลชันได้ดังตารางที่ 2.8 และ 2.9 หรือสามารถเขียนแทนด้วยแผนภาพการขึ้นต่อกันได้ดังรูปที่ 2.11 และ 2.12

ตารางที่ 2.8 ตารางการสั่งที่ปรับปรุงจากตารางที่ 2.5

เลขที่ใบสั่ง	วันที่สั่ง	รหัสลูกค้า
INV001	20-08-2008	C001
INV002	20-08-2008	C002
INV003	21-08-2008	C003

ตารางที่ 2.9 ตารางการลูกค้าที่ปรับปรุงจากตารางที่ 2.5

รหัสลูกค้า	ชื่อลูกค้า
C001	สาทร
C002	บุญมี
C003	บุญมา



รูปที่ 2.11 การขึ้นต่อกันของแอททริบิวท์ในรีเลชันการสั่ง

รูปที่ 2.12 การขึ้นต่อกันของแอททริบิวท์ในรีเลชันลูกค้า

2.7 รูปแบบบรรทัดฐานบอยส์-คอดด์

จากรีเลชันรูปแบบบรรทัดฐานทั้ง 3 แบบที่ได้กล่าวมาแล้ว รูปแบบบรรทัดฐานระดับที่ 3 จะเป็นรูปแบบที่นักออกแบบต้องการมากที่สุด เนื่องจากสามารถขจัดปัญหาความผิดปกติในการจัดการข้อมูลได้ อย่างไรก็ตาม แม้บางครั้งรีเลชันจะอยู่ในรูปแบบบรรทัดฐานระดับที่ 3 แล้วก็ตาม แต่ยังมีโอกาสที่จะเกิดความผิดปกติในการจัดการข้อมูลขึ้นได้อีก ถึงแม้ว่าจะพบค่อนข้างน้อยก็ตาม

Connolly and Begg (2002) ได้อ้างว่าบอยส์ และคอดด์ (Codd, 1974) ได้ให้นิยามรีเลชันที่จะอยู่ในรูปแบบบรรทัดฐานบอยส์-คอดด์ ว่าจะต้องมีคุณสมบัติต่อไปนี้

แอททริบิวท์ที่ทำหน้าที่เป็นตัวเลือก (Determinant) ทุกตัวในรีเลชันจะต้องทำหน้าที่เป็นคีย์คู่แข่ง (Candidate Key) ด้วย

ตัวอย่างที่แสดงในตารางที่ 2.10 เป็นตัวอย่างของรีเลชันคนงาน-ผู้ควบคุม ที่เก็บข้อมูลรหัสคนงาน ประเภทความชำนาญ และชื่อผู้ควบคุม

ตารางที่ 2.10 ตารางคนงาน-ผู้ควบคุม

รหัสคนงาน	ประเภทความชำนาญ	ผู้ควบคุม
1121	ไฟฟ้า	บรรเจิด
1254	ประปา	วันชัย
2543	ไฟฟ้า	สมัย
1511	ไฟฟ้า	บรรเจิด
4500	ประปา	สมหมาย
4500	ไฟฟ้า	สมัย

ถ้าหากมีข้อกำหนดสำหรับการเก็บข้อมูลในรีเลชัน คนงาน-ผู้ควบคุม ดังต่อไปนี้

1. คนงานแต่ละคนสามารถมีความชำนาญได้มากกว่าหนึ่งด้าน เช่น คนงานรหัส “4500” มีความชำนาญทั้งทางด้าน “ประปา” และ “ไฟฟ้า”
2. คนงานแต่ละคนสามารถมีผู้คุมได้มากกว่าหนึ่งคน เช่น คนงานรหัส “4500” จะมีผู้ควบคุม 2 คน คือ “สมหมาย” และ “สมัย”
3. ความชำนาญด้านหนึ่ง ๆ จะสามารถมีผู้ควบคุมได้มากกว่าหนึ่งคน เช่น คนงานรหัส “1121” ที่มีความชำนาญทางด้าน “ไฟฟ้า” จะมีผู้ควบคุมชื่อ “บรรเจิด” และคนงานรหัส “2543” ซึ่งมีความชำนาญทางด้าน “ไฟฟ้า” เช่นกัน แต่มีผู้ควบคุมชื่อ “สมัย”
4. ผู้ควบคุมแต่ละคนจะมีความชำนาญด้านใดด้านหนึ่งเท่านั้น (ไม่มีความชำนาญมากกว่าหนึ่งด้าน)
5. คนงานคนหนึ่ง ที่มีความชำนาญด้านใดด้านหนึ่ง จะมีผู้ควบคุมเพียงคนเดียวเท่านั้น
6. คนงานคนหนึ่ง ที่มีผู้ควบคุมคนหนึ่ง จะมีความชำนาญเพียงด้านเดียวเท่านั้น

จากข้อกำหนดของรีเลชันที่กล่าวในข้างต้น ถ้าหากพิจารณาด้านฟังก์ชันการขึ้นต่อกันของแอททริบิวท์ต่าง ๆ จะพบว่ามีรูปแบบดังต่อไปนี้

- ข้อกำหนดในข้อที่หนึ่งข้างต้น จะทำให้ได้ว่า รหัสคนงาน ไม่สามารถใช้ในการเลือก (Determine) ประเภทความชำนาญได้ หรือกล่าวอีกนัยหนึ่งคือ ประเภทความชำนาญจะไม่ขึ้นกับ รหัสคนงาน
- ข้อกำหนดในข้อที่สอง จะทำให้ได้ว่า รหัสคนงาน ก็ไม่สามารถใช้ในการเลือกผู้ควบคุมได้

- ข้อกำหนดข้อที่สาม จะทำให้ได้ว่า ประเภทความชำนาญ ไม่สามารถใช้ในการเลือกผู้ควบคุมได้
- ข้อกำหนดข้อที่สี่ จะทำให้ได้ว่า ผู้ควบคุม สามารถใช้ในการเลือก ประเภทความชำนาญได้ หรือ ผู้ควบคุม จะเป็นตัวเลือก (Determinant) ประเภทความชำนาญ ซึ่งสามารถเขียนเป็นข้อความสัญลักษณ์ได้ว่า

  ผู้ควบคุม $\rightarrow$ ประเภทความชำนาญ
- ข้อกำหนดข้อที่ห้า จะทำให้ได้ว่าการนำ รหัสคนงาน มารวมกับ ประเภทความชำนาญ จะสามารถใช้ในการเลือกผู้ควบคุมได้ ซึ่งสามารถเขียนเป็นข้อความสัญลักษณ์ได้ว่า

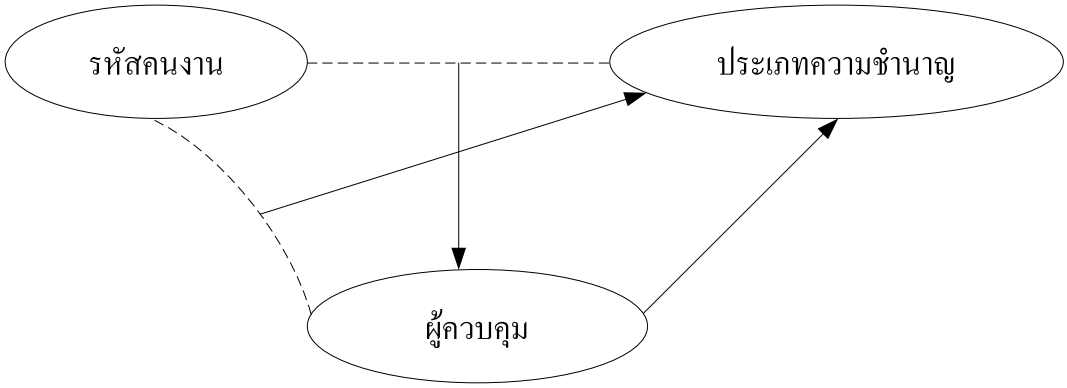
  รหัสคนงาน, ประเภทความชำนาญ $\rightarrow$ ผู้ควบคุม

  ดังนั้น ตัวเลือกคือการรวมกันของแอททริบิวท์รหัสคนงาน และประเภทความชำนาญ
- และข้อกำหนดข้อที่หก จะได้ว่า รหัสคนงาน เมื่อนำมารวมกับ ผู้ควบคุม ก็สามารถใช้ในการเลือกประเภทความชำนาญได้ ซึ่งสามารถเขียนเป็นข้อความสัญลักษณ์ได้ว่า

  รหัสคนงาน, ผู้ควบคุม $\rightarrow$ ประเภทความชำนาญ

  โดยตัวเลือกคือการรวมกันของแอททริบิวท์รหัสคนงาน และผู้ควบคุม

ดังนั้น จากฟังก์ชันการขึ้นต่อกันทั้งหมดที่กล่าวมา สามารถนำมาสรุปเป็นแผนภาพแสดงความสัมพันธ์ระหว่างแอททริบิวท์ได้ ดังรูปที่ 2.13



รูปที่ 2.13 การขึ้นต่อกันของแอททริบิวท์ในรีเลชันคนงาน-ผู้ควบคุม

ในการพิจารณาคีย์ของรีเลชันนี้ จะพบว่าสามารถเลือกใช้คีย์ร่วม (Composite key) อันได้แก่ (รหัสคนงาน, ประเภทความชำนาญ) หรือ (รหัสคนงาน, ผู้ควบคุม) คู่ใดคู่หนึ่งทำหน้าที่เป็นคีย์ของรีเลชันได้ ซึ่งคีย์ร่วมทั้งสองคู่นี้จะถูกเรียกว่าเป็นคีย์คู่แข่ง (Candidate key) ของรีเลชันนั้นเอง

ดังนั้น รีเลชันคนงาน-ผู้ควบคุม จะเป็นรีเลชันที่มีลักษณะดังต่อไปนี้

1. มีคีย์คู่แข่งมากกว่าหนึ่งขึ้นไป (Multiple candidate key) ได้แก่ คีย์คู่แข่ง (รหัสคนงาน, ประเภทความชำนาญ) และคีย์คู่แข่ง (รหัสคนงาน, ผู้ควบคุม)
2. คีย์คู่แข่งเหล่านี้แต่ละคีย์จะมีคุณสมบัติเป็นคีย์ร่วมนั่นคือ มีจำนวนแอททริบิวท์มากกว่าหนึ่งแอททริบิวท์ร่วมกันเป็นคีย์คู่แข่ง
3. คีย์คู่แข่งจะมีแอททริบิวท์ที่เหมือนกันเชื่อมกันอยู่ อันได้แก่ รหัสคนงาน
4. ถ้ามีการเลือกให้คีย์คู่แข่ง (รหัสคนงาน, ประเภทความชำนาญ) ทำหน้าที่เป็นคีย์หลักของรีเลชัน และ (รหัสคนงาน, ผู้ควบคุม) เป็นคีย์คู่แข่ง รีเลชันคนงาน-ผู้ควบคุม นี้จะอยู่ในรูปแบบบรรทัดฐานระดับที่ 1 และอยู่ในรูปแบบบรรทัดฐานระดับที่ 2 ด้วย เนื่องจากมีแอททริบิวท์ที่ไม่ใช่คีย์หลัก อันได้แก่ ผู้ควบคุม จะขึ้นอยู่กับคีย์หลักอย่างแท้จริง และรีเลชันนี้ยังอยู่ในรูปแบบบรรทัดฐานระดับที่ 3 ด้วย เนื่องจากเป็นรีเลชันที่ไม่มีการขึ้นต่อกันแบบทรานซิทีฟ ซึ่งเป็นการขึ้นต่อกันระหว่างแอททริบิวท์ที่ไม่ใช่คีย์หลักด้วยกันเอง

แม้ว่ารีเลชันนี้จะอยู่ในรูปแบบบรรทัดฐานระดับที่ 3 แล้ว แต่อย่างไรก็ตาม รีเลชันก็ยังคงมีปัญหาเกี่ยวกับความผิดปกติต่อการจัดารข้อมูลภายในรีเลชันอยู่ ดังต่อไปนี้

2.7.1 ปัญหาที่อาจเกิดขึ้นกับรีเลชันที่มีรูปแบบบรรทัดฐานระดับที่ 3

1. ความผิดปกติต่อการเปลี่ยนแปลงแก้ไขข้อมูล

ถ้าหากผู้ควบคุมชื่อ “บรรเจิด” มีการเปลี่ยนแปลงชื่อเป็น “บรรจบ” จะต้องมีการแก้ไขข้อมูลหลายแถวที่มีชื่อ “บรรเจิด” อยู่ ซึ่งถ้ามีแถวใดที่ไม่ได้แก้ไข จะทำให้เกิดปัญหาความขัดแย้งของข้อมูลขึ้นได้ เนื่องจากจะกลายเป็นว่ามีผู้คุมที่ชื่อทั้ง “บรรเจิด” และ “บรรจบ” ทั้ง ๆ ที่เป็นคนเดียวกัน

2. ความผิดปกติต่อการลบข้อมูล

ถ้าคนงานที่มีรหัส “4500” ได้ลาออกไป ดังนั้นการลบแถวของคนงานที่มีรหัส “4500” ออกไปจากรีเลชัน จะทำให้ต้องสูญเสียข้อมูลเกี่ยวกับผู้ควบคุมที่ชื่อ “สมหมาย” ซึ่งมีความชำนาญทางด้าน “ประปา” ไปด้วย

3. ความผิดปกติต่อการเพิ่มข้อมูล

ถ้ามีผู้ควบคุมชื่อ “สมบัติ” ที่มีความชำนาญด้าน “ปูน” เข้ามาทำงาน แต่ยังไม่มีคนงานที่อยู่ภายใต้การควบคุมของ “สมบัติ” จะทำให้ไม่สามารถเพิ่มข้อมูลผู้ควบคุมที่ชื่อ “สมบัติ” นี้เข้าไปในรีเลชันได้ เนื่องจากแอททริบิวท์รหัสคนงาน จะค่าเป็นค่าว่าง (Null)

จากนิยามจะเห็นว่า รีเลชันคนงาน-ผู้ควบคุม ข้างต้นไม่อยู่ในรูปแบบบรรทัดฐานบอยส์-คอดด์ เนื่องจากข้อกำหนดในข้อที่สี่ข้างต้นจะพบว่ามีตัวเลือกหนึ่งตัวคือ ผู้ควบคุม ที่ไม่ได้เป็นคีย์คู่แข่ง ซึ่งการมีตัวเลือกที่ไม่ได้เป็นคีย์คู่แข่งนี้เอง ที่เป็นสาเหตุให้รีเลชันเกิดความผิดปกติในการจัดการข้อมูลดังที่กล่าวไว้ข้างต้น จึงต้องมีการปรับรีเลชันใหม่

2.7.2 การปรับรีเลชันรูปแบบบรรทัดฐานระดับที่ 3 เป็นรีเลชันรูปแบบบรรทัดฐานบอยส์-คอดด์

การจะทำให้รีเลชันคนงาน-ผู้ควบคุม อยู่ในรูปแบบบรรทัดฐานบอยส์-คอดด์ จะทำได้โดยการนำแอททริบิวท์ที่เป็นตัวเลือกที่ไม่ใช่คีย์คู่แข่ง และแอททริบิวท์ที่ถูกเลือกของตัวเลือกนั้น แยกออกไปสร้างรีเลชันใหม่อีกหนึ่งรีเลชัน และกำหนดให้แอททริบิวท์ที่เป็นตัวเลือกนั้น ทำหน้าที่เป็นคีย์หลักของรีเลชัน และสำหรับรีเลชันเดิมให้ตัดกลุ่มแอททริบิวท์ที่ถูกแยกไปยังรีเลชันใหม่นี้ออกไป ยกเว้นแอททริบิวท์ที่เป็นตัวเลือก (ที่ไม่ใช่คีย์คู่แข่ง) นั้นให้คงไว้ในรีเลชันเดิม แต่ให้ทำหน้าที่เป็นคีย์นอกเพื่อเชื่อมโยงไปยังรีเลชันใหม่แทน แต่ตัวเลือกที่เป็นคีย์คู่แข่งในรีเลชันเดิม ก็ให้ทำหน้าที่เป็นคีย์หลักเหมือนเดิม

ดังนั้นจากรีเลชันคนงาน-ผู้ควบคุม ในตารางที่ 2.10 แอททริบิวท์ผู้ควบคุม และประเภทความชำนาญ จะถูกแยกออกไปสร้างรีเลชันใหม่ชื่อ รีเลชันผู้ควบคุม เนื่องจากแอททริบิวท์ผู้ควบคุม ซึ่งไม่ใช่คีย์คู่แข่ง ทำหน้าที่เป็นตัวเลือก ประเภทความชำนาญ รีเลชันใหม่นี้จะมีคีย์หลักคือ ผู้ควบคุม และสำหรับรีเลชันเดิมจะต้องทำการตัดแอททริบิวท์ประเภทความชำนาญ ออกไปจากรีเลชัน แต่ยังคงเหลือแอททริบิวท์ผู้ควบคุมไว้ เพื่อให้ทำหน้าที่เป็นคีย์ร่วมกับแอททริบิวท์รหัสคนงาน เพื่อทำหน้าที่เป็นคีย์หลักของรีเลชันคนงาน ด้วย ดังนั้นจากรีเลชันคนงาน-ผู้ควบคุม ดังตารางที่ 2.10 เมื่อปรับรูปแบบบรรทัดฐาน จะได้เป็นรีเลชันใหม่สองรีเลชัน คือ รีเลชันคนงาน (ตารางที่ 2.11) และรีเลชันผู้ควบคุม (ตารางที่ 2.12) โดยรีเลชันคนงาน มีคีย์หลัก คือ (รหัสคนงาน, ผู้ควบคุม) และมีคีย์นอก คือ ผู้ควบคุม ส่วนรีเลชันผู้ควบคุม มีคีย์หลัก คือ ผู้ควบคุม

ตารางที่ 2.11 ตารางคนงาน

รหัสคนงาน	**ผู้ควบคุม**
1121	บรรเจิด
1254	วันชัย
2543	สมัย
1121	บรรเจิด
1254	วันชัย
2543	สมัย
1511	บรรเจิด
4500	สมหมาย
4500	สมัย

ตารางที่ 2.12 ผู้ควบคุม

ผู้ควบคุม	**ประเภทความชำนาญ**
บรรเจิด	ไฟฟ้า
วันชัย	ประปา
สมัย	ไฟฟ้า

รีเลชันที่อยู่ในรูปแบบบรรทัดฐานบอยส์-คอดด์นี้ สามารถรับประกันได้ว่าปัญหาที่เกิดจากฟังก์ชันการขึ้นต่อกัน อันได้แก่ ปัญหาในการจัดการข้อมูลแบบต่าง ๆ จะสามารถขจัดให้หมดไปได้

2.8 รูปแบบบรรทัดฐานระดับที่ 4

ถ้าสมมติว่า มีรีเลชันที่เก็บข้อมูลเกี่ยวกับนักศึกษาที่ลงเรียนวิชาต่าง ๆ ในเทอมหนึ่ง ๆ และความสนใจในกีฬาของนักศึกษาแต่ละคน โดยให้แต่ละแถวในรีเลชันเก็บข้อมูล รหัสนักศึกษา วิชาที่ลงเรียน และ กีฬาที่สนใจ ดังตารางที่ 2.13 และมีข้อกำหนดของแต่ละรีเลชัน ดังต่อไปนี้

1. นักศึกษาแต่ละคนสามารถลงทะเบียนได้มากกว่าหนึ่งวิชาขึ้นไป เช่น นักศึกษารหัส “B4602613” จะลงทะเบียนเรียน 2 วิชา ได้แก่ “คณิตศาสตร์” และ “ฟิสิกส์”

2. นักศึกษาแต่ละคนสามารถมีกีฬาที่ชอบได้มากกว่าหนึ่งอย่าง เช่น นักศึกษารหัส “B4602613” จะชอบกีฬา 2 ประเภท ได้แก่ “ว่ายน้ำ” และ “แบดมินตัน” เป็นต้น

จากรีเลชันรูปแบบข้างต้น คีย์หลักของรีเลชันจะมีลักษณะเป็นคีย์ร่วม คือ ประกอบไปด้วยแอททริบิวท์ทุกแอททิบิวท์ ได้แก่ รหัสนักศึกษา วิชาที่ลงทะเบียน และกีฬาที่สนใจ

ตารางที่ 2.13 ตารางนักศึกษา-วิชา-กีฬา-1

รหัสนักศึกษา	**วิชาที่ลงทะเบียน**	**กีฬาที่สนใจ**
B4602613	คณิตศาสตร์	ว่ายน้ำ
B4602613	ฟิสิกส์	ว่ายน้ำ
B4602613	คณิตศาสตร์	แบดมินตัน
B4602613	ฟิสิกส์	แบดมินตัน
B4603945	ดาต้าเบส	เทนนิส
B4603945	ดาต้าเบส	ว่ายน้ำ
B4609615	ภาษาซี	บาสเกตบอล

พิจารณาว่ารีเลชันที่อยู่ในรูปแบบนี้จะอยู่ในรูปแบบบรรทัดฐานระดับที่ 1 2 และ 3 หรือไม่

1. เนื่องจากรีเลชันนี้ไม่มีข้อมูลกลุ่มซ้ำ รีเลชันจึงอยู่ในรูปแบบบรรทัดฐานระดับที่ 1
2. เนื่องจากคีย์หลักของรีเลชันประกอบด้วยแอททริบิวท์ทุกตัวในรีเลชันรวมกัน ดังนั้นจะไม่มีแอททริบิวท์ที่ไม่ใช่คีย์แอททริบิวท์ใด ที่ขึ้นกับส่วนใดส่วนหนึ่งของคีย์หลัก รีเลชันนี้จึงอยู่ในรูปแบบบรรทัดฐานระดับที่ 2
3. เนื่องจากไม่มีแอททริบิวท์ที่ไม่ใช่คีย์ในรีเลชันนี้ ดังนั้น จึงย่อมไม่มีแอททริบิวท์ที่ไม่ใช่คีย์ใดที่ขึ้นต่อกัน จึงอยู่ในรูปแบบบรรทัดฐานระดับที่ 3 และอยู่ในรูปแบบบรรทัดฐานบอยส์-คอดด์ ด้วย เนื่องจากไม่มีตัวเลือกที่ไม่ใช่คีย์คู่แข่ง

แม้ว่ารีเลชันนี้จะอยู่ในรูปแบบบรรทัดฐานระดับที่ 3 แล้วก็ตาม แต่ถ้าลองพิจารณาดูดี ๆ จะพบว่า รีเลชันนี้ยังคงมีปัญหาเกี่ยวกับการจัดการข้อมูล อันได้แก่ การปรับปรุงเปลี่ยนแปลง การลบหรือเพิ่มข้อมูล ดังรายละเอียดต่อไปนี้

2.8.1 ปัญหาที่อาจเกิดกับบางรีเลชันที่มีรูปแบบบรรทัดฐานระดับที่ 3 และรูปแบบบรรทัดฐานบอยส์-คอดด์

1. ปัญหาความผิดปกติต่อการเปลี่ยนแปลงแก้ไขข้อมูล

ยกตัวอย่างเช่น ถ้านักศึกษารหัส “B4602613” ต้องการเปลี่ยแปลงวิชาที่ลงทะเบียนเรียนจากวิชา “ฟิสิกส์” เป็นวิชา “สถิติ” จะต้องมีการแก้ไขข้อมูลของนักศึกษาคนนั้นทุกแถวที่มีวิชาเป็น “ฟิสิกส์” ซึ่งถ้าแก้ไขไม่ครบก็จะทำให้เกิดปัญหาความขัดแย้งของข้อมูลขึ้น

2. ปัญหาความผิดปกติต่อการลบข้อมูล

ถ้านักศึกษารหัส “B4609615” ได้ทำการขอยกเลิกการลงทะเบียนวิชาเรียน “ภาษาซี” ดังนั้น การลบแถวข้อมูลของนักศึกษารหัส “B4609615” ออกไป จะทำให้ต้องสูญเสียข้อมูลของนักศึกษาคนอื่น ๆ ที่ชอบกีฬาประเภท “บาสเกตบอล” ไปด้วย

3. ปัญหาความผิดปกติต่อการเพิ่มข้อมูล

ถ้าหากต้องการเพิ่มข้อมูลของนักศึกษาที่มีรหัสเป็น “B4604595” ที่ลงทะเบียนวิชา “คอมพิวเตอร์เบื้องต้น” แต่นักศึกษาคนนี้เป็นคนที่ไม่ชอบกีฬาประเภทใดเลย ดังนั้น จะไม่สามารถเพิ่มข้อมูลของนักศึกษาคนนี้ลงในรีเลชันได้ เพราะไม่สามารถใส่ค่าว่างลงในแอททริบิวท์ กีฬาที่สนใจ ได้ เนื่องจากเป็นส่วนหนึ่งของคีย์หลัก

2.8.2 การปรับรีเลชันที่มีการขึ้นต่อกันเชิงกลุ่มไปเป็นรีเลชันที่มีรูปแบบบรรทัดฐานระดับที่ 4

ปัญหาที่เกิดขึ้นข้างต้น เป็นปัญหาที่เกิดจากลักษณะของข้อมูลที่มีการขึ้นต่อกันแบบเชิงกลุ่ม ซึ่งมีนิยามดังต่อไปนี้ (Fagin, 1977; Date, 2000)

นิยาม: การขึ้นต่อกันแบบเชิงกลุ่ม

ถ้ามีรีเลชัน R ใด ๆ ประกอบด้วยแอททริบิวท์อย่างน้อย 3 แอททริบิวท์ เช่น R (X, Y, Z) การขึ้นต่อกันแบบเชิงกลุ่มจะเกิดขึ้นถ้า

1. แอททริบิวท์ X ที่ใช้ในการเลือกกลุ่มข้อมูลในแอททริบิวท์ Y (กลุ่มข้อมูลใน Y ขึ้นอยู่กับ X) จะเขียนแทนด้วยสัญลักษณ์ $X \rightarrow\rightarrow Y$ และ
2. แอททริบิวท์ X จะใช้ในการเลือกกลุ่มข้อมูลใน Z (กลุ่มข้อมูล Z จะขึ้นกับ X) เขียนแทนด้วย $X \rightarrow\rightarrow Z$ และ
3. ข้อมูลใน Yและ Z จะมีความเป็นอิสระต่อกันคือไม่ขึ้นแก่กัน

จากลักษณะของข้อมูลในรีเลชันดังตารางที่ 2.13 จะพบว่านักศึกษารหัส “B4602613” จะมีข้อมูลอยู่ถึง 4 แถว โดยมีจุดมุ่งหมายคือ ต้องการเก็บข้อมูลว่านักศึกษาคนนี้ลงทะเบียนเรียนเพียงสองวิชา คือ “คณิตศาสตร์” และ “ฟิสิกส์” และกีฬาที่นักศึกษาคนนี้ชอบคือ “ว่ายน้ำ” และ

“แบดมินตัน” ซึ่งถ้าลองทำการเก็บข้อมูลนักศึกษาคนนี้ใหม่เพียง 2 แถว ดังตารางที่ 2.14 จะทำให้ตีความหมายได้ว่า นักศึกษาคนที่มีรหัสเป็น “B4602613” จะชอบกีฬาว่ายน้ำก็ต่อเมื่อเขาต้องลงเรียนวิชา “คณิตศาสตร์” และจะชอบกีฬา “แบดมินตัน” ก็ต่อเมื่อเขาต้องลงเรียนวิชา “ฟิสิกส์” ซึ่งเป็นรูปแบบการเก็บข้อมูลที่ไม่ถูกต้อง อันจะนำไปสู่การตีความหมายของข้อมูลที่ผิดได้ ทั้งนี้เนื่องจาก วิชาที่ลงทะเบียนกับกีฬาที่ชอบนั้นเป็นอิสระต่อกัน คือ ไม่ขึ้นต่อกัน

ตารางที่ 2.14 ตารางนักศึกษา-วิชา-กีฬา-2

รหัสนักศึกษา	**วิชาที่ลงทะเบียน**	**กีฬาที่สนใจ**
B4602613	คณิตศาสตร์	ว่ายน้ำ
B4602613	ฟิสิกส์	แบดมินตัน
B4603945	ดาต้าเบส	เทนนิส
B4603945	ดาต้าเบส	ว่ายน้ำ
B4609615	ภาษาซี	บาสเกตบอล

หากมีการเก็บข้อมูลในรีเลชันดังตารางที่ 2.14 และถ้านักศึกษารหัส “B4602613” มีกีฬาที่ชอบเพิ่มขึ้นมาอีกหนึ่งประเภท คือ “แฮนด์บอล” จึงจำเป็นต้องมีการเพิ่มแถวไปอีกสองแถว ตามจำนวนวิชาที่นักศึกษาคนนั้นลงเรียน เพื่อไม่ให้เกิดการขึ้นต่อกันระหว่างวิชาที่ลงทะเบียนและกีฬาที่สนใจ จะได้ข้อมูลดังตารางที่ 2.15

ตารางที่ 2.15 ตารางนักศึกษา-วิชา-กีฬา-3

รหัสนักศึกษา	**วิชาที่ลงทะเบียน**	**กีฬาที่สนใจ**
B4602613	คณิตศาสตร์	แฮนด์บอล
B4602613	ฟิสิกส์	แฮนด์บอล
B4602613	คณิตศาสตร์	ว่ายน้ำ
B4602613	ฟิสิกส์	ว่ายน้ำ
B4602613	คณิตศาสตร์	แบดมินตัน
B4602613	ฟิสิกส์	แบดมินตัน
B4603945	ดาต้าเบส	เทนนิส
B4603945	ดาต้าเบส	ว่ายน้ำ
B4609615	ภาษาซี	บาสเกตบอล

จากนิยามของการขึ้นต่อกันแบบเชิงกลุ่มจะเห็นว่า รีเลชันดังตารางที่ 2.15 เป็นรีเลชันที่มีการขึ้นต่อกันแบบเชิงกลุ่ม โดยมีแอททริบิวท์รหัสนักศึกษา ที่ใช้ในการเลือกกลุ่มข้อมูลของ วิชาที่ลงทะเบียน เขียนแทนด้วยสัญลักษณ์ได้ว่า

รหัสนักศึกษา $\rightarrow\rightarrow$ วิชาที่ลงทะเบียน

นอกจากนี้ แอททริบิวท์รหัสนักศึกษา ก็ยังใช้ในการเลือกกลุ่มข้อมูลของ กีฬาที่ชอบ ซึ่งเขียนแทนด้วยสัญลักษณ์ได้ว่า

รหัสนักศึกษา $\rightarrow\rightarrow$ กีฬาที่ชอบ

แต่ วิชาที่ลงทะเบียน และกีฬาที่ชอบ จะมีความเป็นอิสระต่อกัน คือ ไม่ขึ้นต่อกัน

Fagin (1977) ได้ให้นิยามไว้ว่า รีเลชันใดจะอยู่ในรูปแบบบรรทัดฐานระดับที่ 4 เมื่อรีเลชันนั้นอยู่ในรูปแบบบรรทัดฐานบอยส์-คอดด์ และต้องไม่มีการขึ้นต่อกันแบบเชิงกลุ่มเกิดขึ้นภายในรีเลชันนั้น

จากนิยามของรูปแบบบรรทัดฐานระดับที่ 4 จะพบว่า รีเลชันดังตารางที่ 2.15 ไม่อยู่ในรูปแบบบรรทัดฐานระดับที่ 4 เนื่องจากภายในรีเลชันมีการขึ้นต่อกันแบบเชิงกลุ่มอยู่ และยังคงก่อให้เกิดปัญหาเกี่ยวกับการจัดการข้อมูลภายในรีเลชันอยู่

ดังนั้น เพื่อขจัดปัญหาที่เกิดขึ้นดังกล่าว จึงจำเป็นต้องขจัดการขึ้นต่อกันแบบเชิงกลุ่มนี้ออกไปจากรีเลชัน เพื่อทำให้รีเลชันอยู่ในรูปแบบบรรทัดฐานระดับที่ 4 โดยทำการแยกแอททริบิวท์ที่มีการขึ้นต่อกันแบบเชิงกลุ่ม ออกเป็นรีเลชันใหม่ และกำหนดให้แอททริบิวท์ทุกตัวในรีเลชันทำหน้าที่เป็นคีย์หลักของรีเลชัน ดังนั้น รีเลชันนักศึกษา-วิชา-กีฬา-3 ดังตารางที่ 2.15 จะถูกแยกออกเป็นสองรีเลชันได้แก่ รีเลชันนักศึกษา-วิชา และรีเลชันนักศึกษา-กีฬา ดังแสดงในตารางที่ 2.16 และ 2.17 ตามลำดับ ซึ่งเป็นรีเลชันที่อยู่ในรูปแบบบรรทัดฐานระดับที่ 4 แล้ว

ตารางที่ 2.16 ตารางนักศึกษา-วิชา

รหัสนักศึกษา	**วิชาที่ลงทะเบียน**
B4602613	คณิตศาสตร์
B4602613	ฟิสิกส์
B4603945	ดาต้าเบส
B4609615	ภาษาซี

ตารางที่ 2.17 ตารางนักศึกษา-กีฬา

รหัสนักศึกษา	**กีฬาที่สนใจ**
B4602613	แฮนด์บอล
B4602613	ว่ายน้ำ
B4602613	แบดมินตัน
B4603945	เบนนิส
B4603945	ว่ายน้ำ
B4609615	บาสเกตบอล

2.9 รูปแบบบรรทัดฐานระดับที่ 5

Fagin (1979) ได้ให้นิยามไว้ว่า รีเลชันใดจะอยู่ในรูปแบบบรรทัดฐานระดับที่ 5 เมื่อรีเลชันนั้นอยู่ในรูปแบบบรรทัดฐานระดับที่ 4 และต้องไม่มีคุณสมบัติการขึ้นต่อกันแบบเชิงร่วม (Join dependency)

รูปแบบบรรทัดฐานระดับที่ 5 นี้บางครั้งอาจเรียกว่า Project-Join Normal Form (PJ/NF) เกิดขึ้นในกรณีที่ รีเลชันหนึ่งประกอบด้วยจำนวนแอททริบิวท์อย่างน้อยสามแอททริบิวท์ขึ้นไป และแอททริบิวท์เหล่านี้ต้องไม่มีความสัมพันธ์ระหว่างแอททริบิวท์แบบบางส่วน และต้องไม่มีการขึ้นต่อกันแบบเชิงกลุ่ม แต่ก็ยังมีปัญหาเกิดขึ้น (ศิริลักษณ์ โรจนกิจอำนวย, 2545)

ตัวอย่างเช่น รีเลชันพนักงาน-ความชำนาญ-โครงการ ประกอบด้วยแอททริบิวท์ทั้งหมด 3 แอททริบิวท์ ได้แก่ รหัสพนักงาน (EMP_NO) ความชำนาญ (SKILL) และรหัสโครงการ (PROJ_NO) รีเลชันนี้มีความสัมพันธ์ของแอททริบิวท์ทั้งสามในลักษณะที่เป็นวงจรกล่าวคือ

- พนักงานมีความชำนาญ
- โครงการต้องการความชำนาญ
- พนักงานที่มีความชำนาญถูกมอบหมายให้ทำงานในโครงการ

ตารางที่ 2.18 พนักงาน-ความชำนาญ-โครงการ

EMP_NO	**SKILL**	**PROJ_NO**
1001	Computer	Proj01
1001	Computer	Proj02
1001	English	Proj02
1001	Math	Proj03
1002	English	Proj02
1003	Computer	Proj02

พิจารณาตารางที่ 2.18 จะเห็นว่า พนักงานรหัส “1001” ที่มีความชำนาญ คือ “Computer” “English” และ “Math” โดยโครงการ “Proj02” มีความต้องการผู้ที่มีความชำนาทั้ง “Computer” และ “English” ซึ่งหากพิจารณาตารางตามคุณสมบัติของรูปแบบบรรทัดฐาน จะเห็นว่าตารางที่ 2.18 อยู่ในรูปแบบบรรทัดฐานระดับที่ 4 โดยมีทั้งสามแอททริบิวท์ประกอบกันเป็นคีย์หลัก แต่ก็ยังคงเป็นตารางที่ยังคงมีปัญหาการจัดการข้อมูลอยู่ เช่น ถ้าต้องการลบข้อมูลของ “Proj03” ทิ้ง จะมีผลทำให้ต้องสูญเสียข้อมูลความชำนาญด้าน “Math” ไป เนื่องจากไม่มีการอ้างอิงถึงข้อมูลนี้ในแถวอื่น ๆ จึงต้องทำการแตกตาราง เป็นสามตารางย่อย โดยในแต่ละตารางย่อยจะประกอบไปด้วยแอททริบิวท์แต่ละคู่เป็นคีย์หลักดังตารางที่ 2.19-2.21

ตารางที่ 2.19 พนักงาน-ความชำนาญ

EMP_NO	**SKILL**
1001	Computer
1001	English
1001	Math
1002	English
1003	Computer

ตารางที่ 2.20 ความชำนาญ-โครงการ

SKILL	**PROJ_NO**
Computer	Proj01
Computer	Proj02
English	Proj02
Math	Proj03

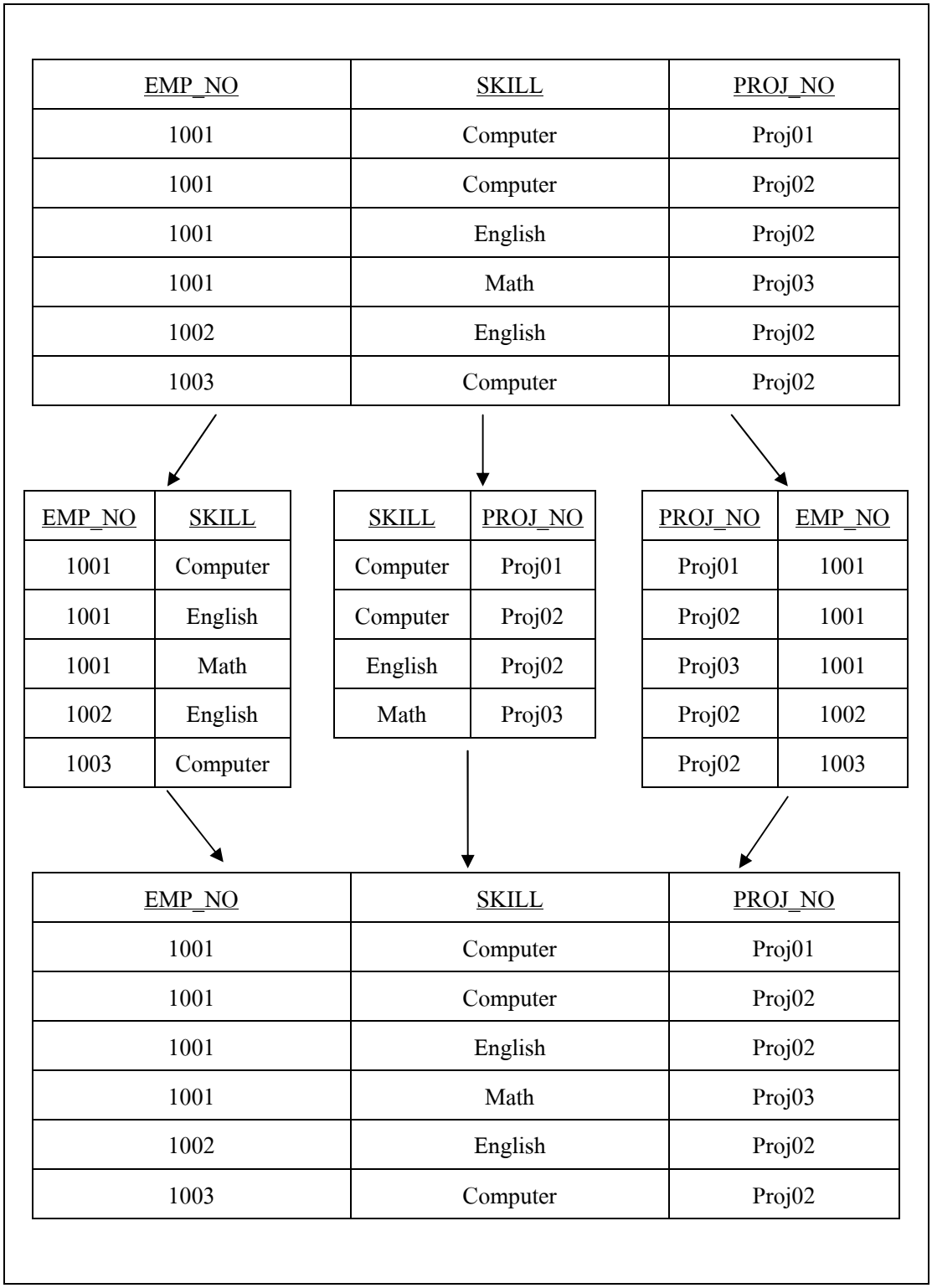
ตารางที่ 2.21 พนักงาน-โครงการ

PROJ_NO	**EMP_NO**
Proj01	1001
Proj02	1001
Proj03	1001
Proj02	1002
Proj02	1003

เมื่อมีการแตกรีเลชันจากรีเลชันหนึ่งเป็นหลายตารางในลักษณะนี้คือ R (a1,a2,a3) เป็น R1 (a1,a2), R2 (a2,a3) และ R3 (a3,a1) โดยรีเลชันที่แตกออกจะมีบางส่วนที่เหมือนกัน กล่าวคือ R1 มีแอททริบิวท์ a1 และ a2 R2 มีแอททริบิวท์ a2 และ a3 ซึ่งมีแอททริบิวท์เหมือนกับ R1 คือ a2 R3 มีแอททริบิวท์ a3 และ a1 ซึ่งมีแอททริบิวท์ที่เหมือนกับ R2 คือ a3 นั่นคือ แอททริบิวท์ a1, a2 และ a3 ของรีเลชัน R ถูกแตกออกเป็น (a1,a2), (a2,a3) และ (a3,a1) ตามลำดับ

Join dependency ของรีเลชันจะเกิดเมื่อมีการเชื่อมโยง (Join) รีเลชันที่แตกในลักษณะข้างต้น โดยการเชื่อมโยงจะใช้แอททริบิวท์ที่เหมือนกัน ระหว่างรีเลชันย่อยที่แตกออกมาเชื่อมโยงกัน ผลจากการเชื่อมโยงรีเลชันที่แตกออกมาทั้งหมด จะต้องได้ข้อมูลเหมือนรีเลชันเดิม

การแตกรีเลชันพนักงาน-ความชำนาญ-โครงการ เป็นสามรีเลชัน คือ รีเลชันพนักงาน-ความชำนาญ รีเลชันความชำนาญ-โครงการ และรีเลชัน พนักงาน-โครงการ เมื่อทำการรวมรีเลชันทั้งสามเข้าด้วยกันจะได้ผลดังรูปที่ 2.14



รูปที่ 2.14 การรวมกันของรีเลชันพนักงาน-ความชำนาญ รีเลชันความชำนาญ-โครงการ และรีเลชันพนักงาน-โครงการ

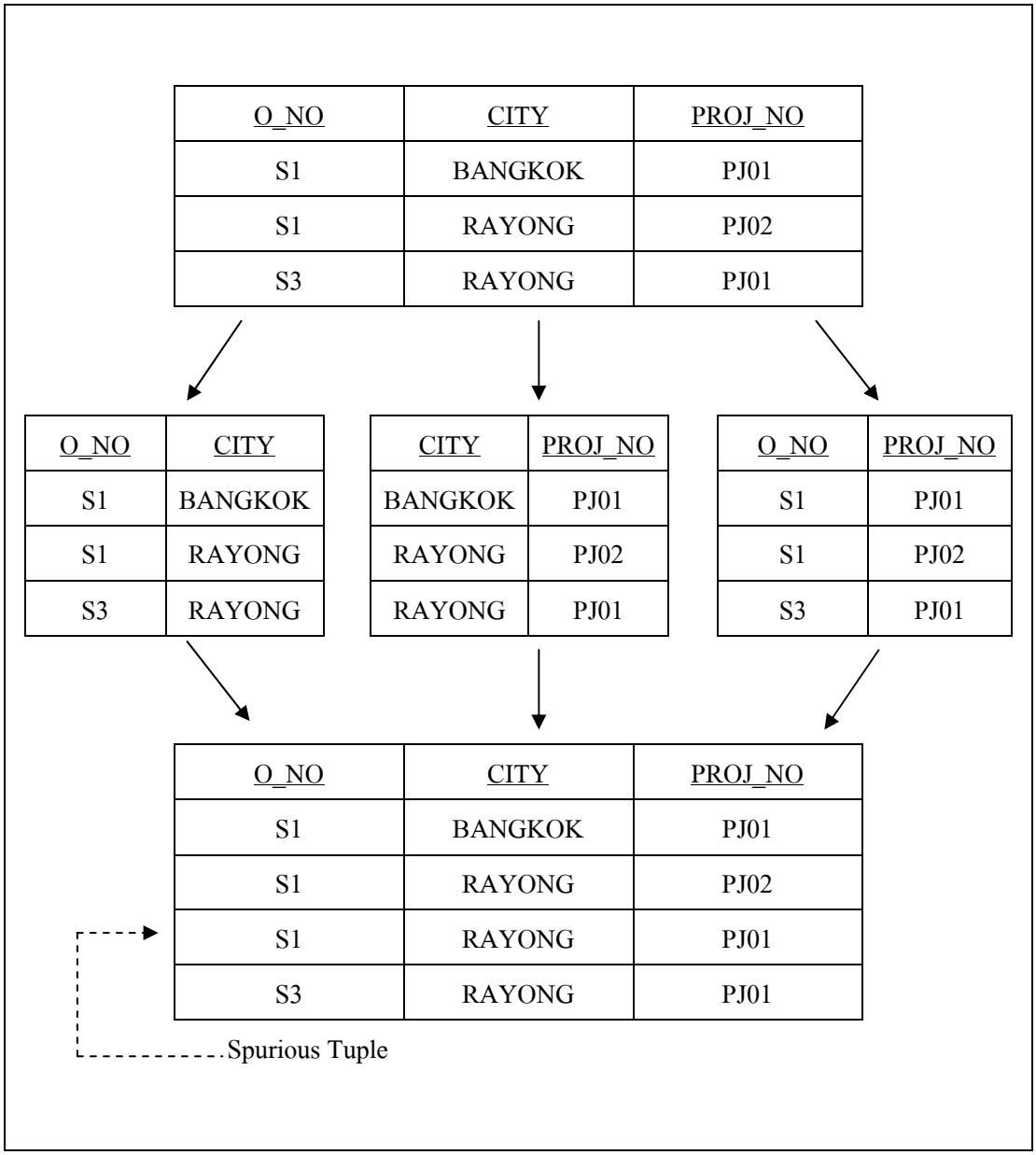
พิจารณารูปที่ 2.14 จะเห็นว่าผลลัพธ์ที่ได้จากการรวมทั้งสามรีเลชัน ให้ผลเหมือนเดิมคือเหมือนกับตารางที่ 2.18 นั่นคือ รีเลชันพนักงาน-ความชำนาญ-โครงการ ไม่ได้อยู่ในรูปแบบบรรทัดฐานระดับที่ 5 เนื่องจากมี Join Dependency และแอททริบิวท์ SKILL หรือ PROJ_NO หรือ EMP_NO ที่ใช้ในการเชื่อมโยงต่างก็เป็นเพียงส่วนหนึ่งของคีย์หลัก จึงต้องทำการแตกรีเลชันย่อยเป็นสามรีเลชัน

กล่าวอีกนัยหนึ่งคือ รีเลชันหนึ่งจะอยู่ในรูปแบบบรรทัดฐานระดับที่ 5 ก็ต่อเมื่อ Join Dependency ซึ่งเป็นผลมาจากการเชื่อมโยงแอททริบิวท์ที่เหมือนกันระหว่างรีเลชันที่แตกออก โดยแอททริบิวท์ที่ใช้ในการเชื่อมโยงเหล่านี้ ต้องใช้เป็นเชื่อมโยงของแอททริบิวท์ที่เป็นคีย์ในรีเลชันอื่น ในทางตรงกันข้ามถ้าไม่เกิด Join Dependency คือ หลังจากรวมรีเลชันที่แตกออกแล้วได้ข้อมูลไม่เหมือนเดิม ก็ให้ถือว่ารีเลชันนั้นอยู่ในรูปแบบบรรทัดฐานระดับที่ 5 แล้ว ตัวอย่างเช่น ตารางที่ 2.22

ตารางที่ 2.22 ผู้ผลิต-จังหวัด-โครงการ

O_NO	**CITY**	**PROJ_NO**
S1	BANGKOK	PJ01
S1	RAYONG	PJ02
S3	RAYONG	PJ01

พิจารณาตารางที่ 2.22 จะเห็นว่าเป็นรีเลชันที่อยู่ในรูปแบบบรรทัดฐานระดับที่ 4 โดยประกอบด้วย 3 แอททริบิวท์ ได้แก่ รหัสผู้ผลิต (O_NO) ชื่อจังหวัดของผู้ผลิต (CITY) และรหัสโครงการ (PROJ_NO) ประกอบกันเป็นคีย์หลัก รีเลชันนี้ยังคงมีปัญหาที่อาจเกิดความผิดพลาดในการเพิ่ม ลบหรือปรับปรุงข้อมูล จึงทำการแตกรีเลชันนี้ออกเป็น รีเลชันผู้ผลิต-จังหวัด รีเลชันจังหวัด-โครงการ และรีเลชันผู้ผลิต-โครงการ ซึ่งดูเหมือนจะเหมาะสม แต่เกิดปัญหาเมื่อนำรีเลชันทั้งสามมาเชื่อมโยงกันจะมีข้อมูลที่เกินมา (Spurious tuple) คือ ข้อมูลแถวที่ 3 ดังรูปที่ 2.15 ซึ่งเป็นข้อมูลที่ไม่มีอยู่ในรีเลชันเดิม หากเกิดปัญหาในลักษณะนี้ ให้ถือว่าตารางที่ 2.22 เป็นรีเลชันที่อยู่ในรูปแบบบรรทัดฐานระดับที่ 5 แล้ว ไม่ต้องทำการแตกรีเลชัน



รูปที่ 2.15 การรวมกันของรีเลชันผู้ผลิต-จังหวัด จังหวัด-โครงการ และผู้ผลิต-โครงการ

2.10 การแตกรีเลชันมากเกินไป

การทำให้รีเลชันอยู่ในรูปแบบบรรทัดฐาน มีวัตถุประสงค์เพื่อลดปัญหาความซ้ำซ้อนของข้อมูล และลดปัญหาในเรื่องการเพิ่ม ลบ หรือปรับปรุงข้อมูล โดยทั่วไปแล้วการออกแบบในระดับแนวคิด ผู้ออกแบบจะพยายามวิเคราะห์รีเลชันให้อยู่ในรูปแบบบรรทัดฐานระดับที่ 3 แต่ถ้ามีกรณีของปัญหาที่จำเป็นต้องทำต่อไปถึงรูปแบบบรรทัดฐานบอยส์-คอดด์ หรือระดับที่ 5 หรือระดับที่ 6

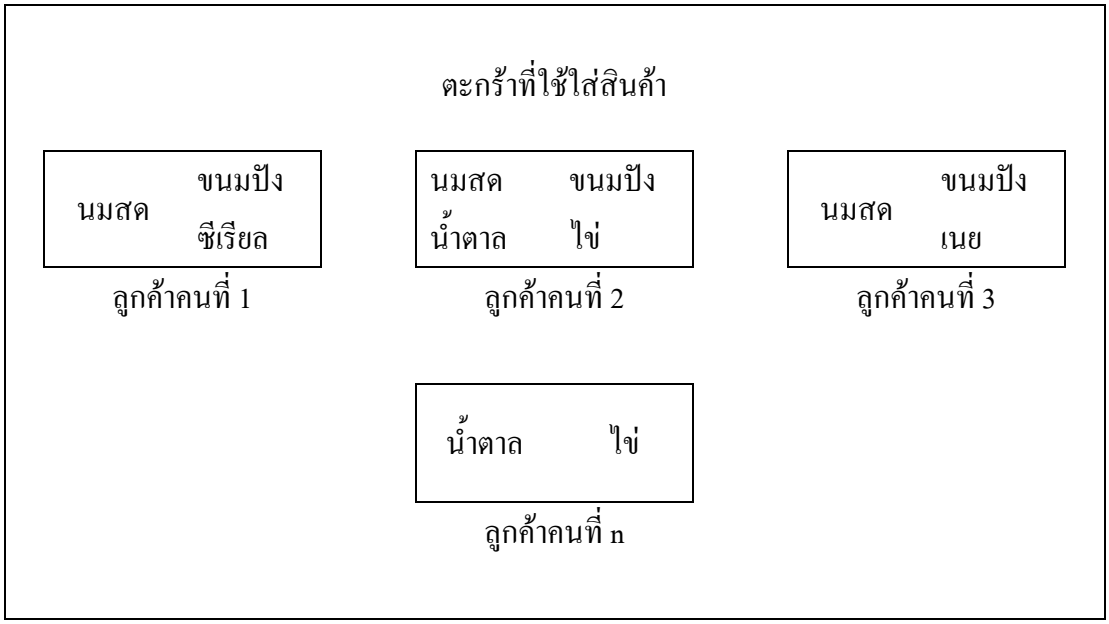
ซึ่งมีโอกาสเกิดขึ้นได้น้อยมากในทางปฏิบัติ ก็พยายามอย่าแตกรีเลชันมากเกินความจำเป็น (Overnormalization)

ข้อเสียของการแตกรีเลชันเกินความจำเป็น จะส่งผลให้ระบบทำงานมีประสิทธิภาพไม่ดีนัก หากมีการเรียกใช้ข้อมูลที่ต้องมีการเชื่อมโยงข้อมูลระหว่างรีเลชัน อาจจะทำให้สิ้นเปลืองทรัพยากร และอาจทำให้ไม่มีประสิทธิภาพเท่าที่ควร นอกจากนี้ยังทำให้สูญเสียความสัมพันธ์ของข้อมูลที่จะใช้ประโยชน์จากรีเลชันนั้นโดยตรง

2.11 กฎความสัมพันธ์

กฎความสัมพันธ์ เป็นกฎที่ได้จากการค้นหาความสัมพันธ์ระหว่างข้อมูลจากข้อมูลจำนวนมหาศาล ที่ถูกเก็บไว้ในระบบฐานข้อมูล ซึ่งส่วนใหญ่แล้วนิยมใช้กับข้อมูลทรานแซคชันทางด้านธุรกิจ (Business transaction) (Han and Kamber, 2001) โดยจะนำผลลัพธ์ที่ได้จากการค้นหาความสัมพันธ์มาใช้ประกอบการตัดสินใจในด้านธุรกิจ เช่น การวางแผนการจัดโปรโมชันการขายสินค้า ว่าควรขายสินค้าชนิดใดคู่กัน จึงจะทำให้มียอดจำหน่ายสูงสุด

ตัวอย่างของการค้นหากฎความสัมพันธ์อย่างง่าย เช่น การวิเคราะห์พฤติกรรมของผู้บริโภค โดยจะทำการค้นหาความสัมพันธ์ของสินค้าต่างชนิดกัน ที่ถูกซื้อพร้อมกันของลูกค้าแต่ละคน ดังแสดงในรูปที่ 2.16 เป็นพฤติกรรมการเลือกซื้อสินค้าของผู้บริโภค ซึ่งจะสังเกตเห็นว่าเมื่อลูกค้าซื้อนมสดแล้วก็จะซื้อขนมปังด้วย ดังพฤติกรรมการซื้อของลูกค้าคนที่ 1 2 และ 3 เป็นต้น



รูปที่ 2.16 การวิเคราะห์การซื้อสินค้าของผู้บริโภค

2.12 วิธีการค้นหากฎความสัมพันธ์

กระบวนการค้นหากฎความสัมพันธ์สามารถแบ่งออกเป็น 2 ขั้นตอน คือ

2.12.1 การหาไอเท็มเซตที่ปรากฏบ่อย (Frequent itemsets)

การค้นหาไอเท็มเซตที่ปรากฏบ่อย เป็นการนำไอเท็มทั้งหมดที่มีอยู่มาจัดหมวดหมู่ โดยไอเท็มหมายถึงสิ่งที่เราสนใจในขณะนั้น เช่น หากกล่าวถึงทรานแซคชันการซื้อสินค้า ไอเท็มก็จะหมายถึง สินค้าแต่ละตัวหรือแต่ละชนิด ซึ่งจำนวนของหมู่ไอเท็มเซตจะมีขนาดเพิ่มขึ้นตามจำนวนของไอเท็ม ยิ่งถ้าไอเท็มมีจำนวนมากขึ้น การค้นหาไอเท็มเซตที่ปรากฏบ่อยก็จะมีจำนวนเซตที่ต้องทำการค้นหามากขึ้นตามไปด้วย

การคัดเลือกไอเท็มเพื่อนำมาใช้ในการสร้างไอเท็มเซตที่ปรากฏบ่อย จะเลือกเอาเฉพาะไอเท็มที่มีค่าความถี่มากกว่าหรือเท่ากับค่าสนับสนุนที่ผู้ใช้กำหนดไว้มาใช้เท่านั้น โดยเรียกไอเท็มเซตที่มีค่าความถี่มากกว่าหรือเท่ากับค่าสนับสนุนนี้ว่า ไอเท็มเซตที่ปรากฏบ่อย (Large itemset) ส่วนไอเท็มเซตนอกจากนี้เรียกว่า Small itemset

2.12.2 การสร้างกฎความสัมพันธ์จากไอเท็มเซตที่ปรากฏบ่อย

การสร้างกฎความสัมพันธ์จะสร้างมาจากไอเท็มเซตที่ปรากฏบ่อย โดยกฎความสัมพันธ์ที่ได้จะต้องมีค่าความเชื่อมั่น (Confidence) มากกว่าหรือเท่ากับ ค่าความเชื่อมั่นขั้นต่ำที่ผู้ใช้ได้กำหนดไว้ ซึ่งค่าความเชื่อมั่นสามารถหาได้จาก

$$\text{confidence}(A \Rightarrow B) = \frac{\text{support\_count}(A \cup B)}{\text{support\_count}(A)}$$

โดย $\text{support\_count}(A \cup B)$ คือ จำนวนทรานแซคชันที่ประกอบด้วย A และ B

$\text{support\_count}(A)$ คือ จำนวนทรานแซคชันที่ประกอบด้วย A

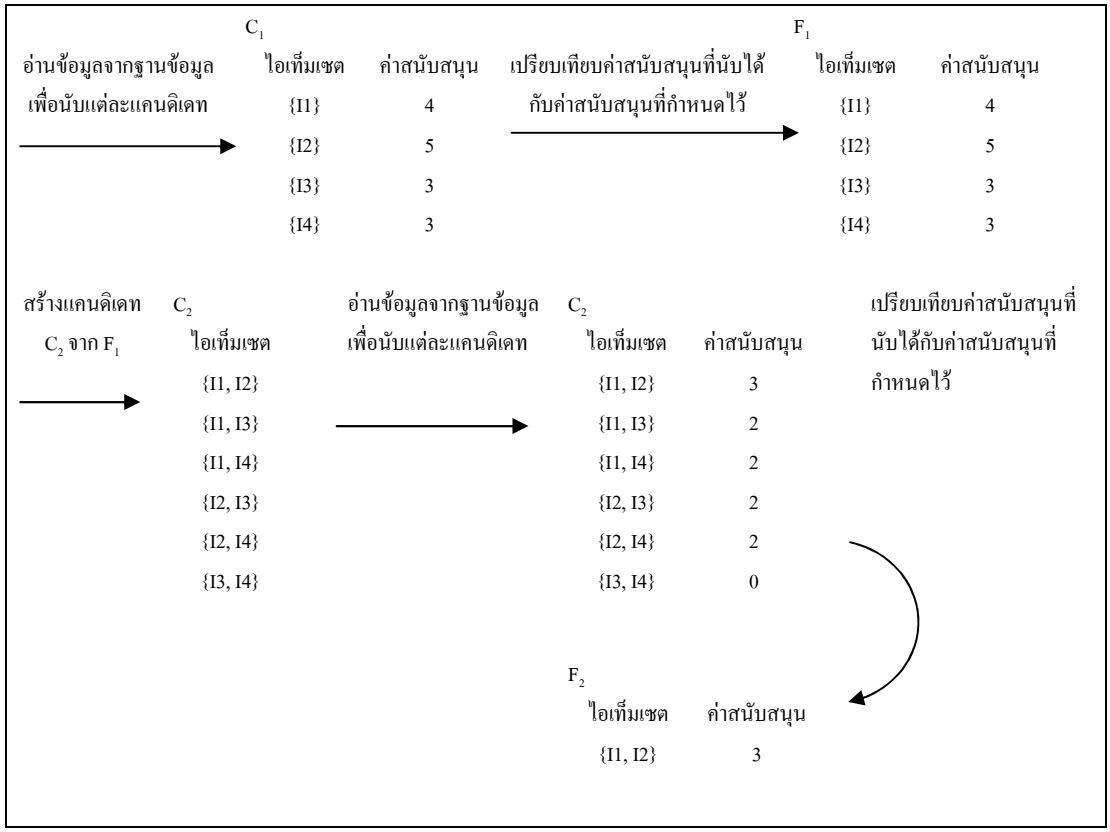
ตัวอย่างการค้นหากฎความสัมพันธ์

<u>ตัวอย่าง</u> พิจารณารายการทรานแซคชันการขายสินค้าของร้านค้าแห่งหนึ่ง ซึ่งมีสินค้าอยู่ 4 ชนิด คือ I1, I2, I3 และ I4 รายการซื้อสินค้าแสดงดังตารางที่ 2.23 โดยแต่ละแถว คือ รายการซื้อสินค้าของลูกค้าแต่ละคน

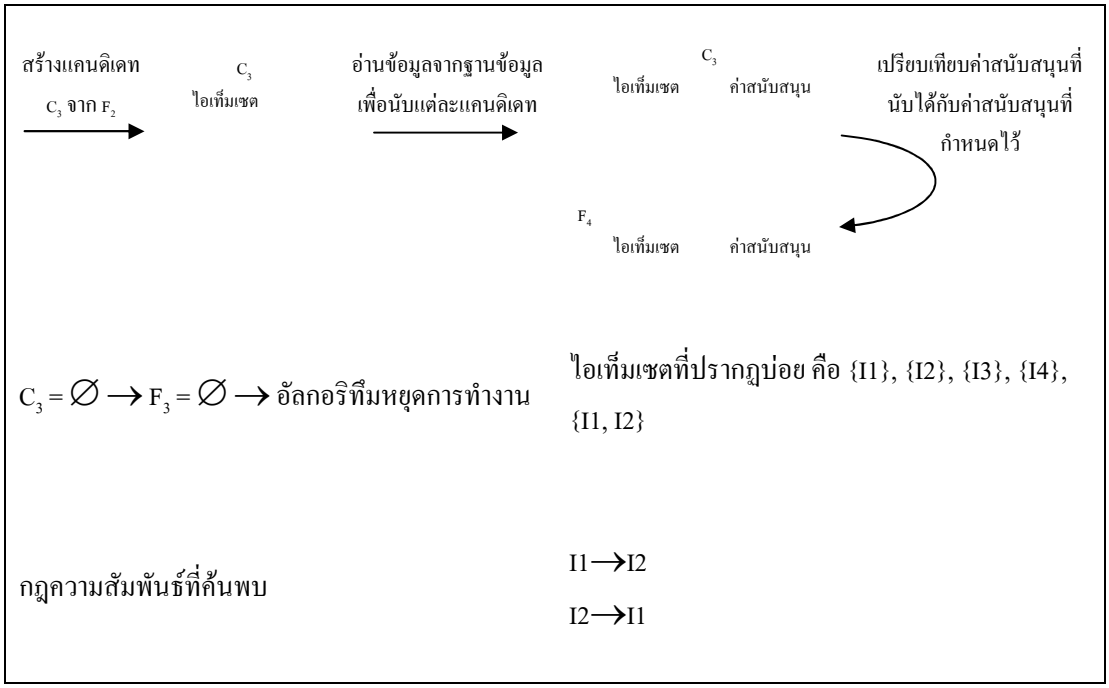
ตารางที่ 2.23 ข้อมูลทรานแซคชันการขายสินค้า

TID	ITEMSET
T001	I1, I2
T002	I2, I4
T003	I2, I3
T004	I1, I2, I4
T005	I1, I3
T006	I1, I2, I3

จากตารางที่ 2.23 สามารถจำลองภาพการค้นหากฎความสัมพันธ์ โดยกำหนดค่าสนับสนุนขั้นต่ำเท่ากับ 50 เปอร์เซ็นต์ หรือมีปรากฏขึ้น 3 ครั้ง จากจำนวน 6 ทรานแซคชัน ได้ดังรูปที่ 2.17



รูปที่ 2.17 ตัวอย่างการค้นหากฎความสัมพันธ์



รูปที่ 2.17 ตัวอย่างการค้นหากฎความสัมพันธ์ (ต่อ)

2.13 ประเภทของกฎความสัมพันธ์

ในการวิเคราะห์พฤติกรรมการซื้อสินค้าของลูกค้า (Market basket analysis) เป็นอีกรูปแบบหนึ่งในการค้นหากฎความสัมพันธ์ ในความเป็นจริงแล้วกฎความสัมพันธ์มีอยู่หลายประเภท เราสามารถจำแนกกฎความสัมพันธ์ได้หลายแนวทางขึ้นอยู่กับกฎเกณฑ์ที่ใช้ในการจำแนก

1) *กฎความสัมพันธ์แบบข้อเท็จจริง (Boolean association rule)*

เป็นความสัมพันธ์ที่บ่งบอกถึงการมีหรือไม่มีไอเท็มนั้นอยู่ ซึ่งกฎความสัมพันธ์ที่เป็นข้อเท็จจริงที่ได้มาจากการวิเคราะห์พฤติกรรมการซื้อสินค้าของผู้บริโภค เช่น

นม ⟶ ขนม (2.1)

2) *กฎความสัมพันธ์เกี่ยวกับปริมาณ (Quantitative association rule)*

เป็นกฎความสัมพันธ์ ที่อธิบายในเรื่องเกี่ยวกับความสัมพันธ์ระหว่างปริมาณไอเท็มหรือแอททริบิวท์ ซึ่งในกฎความสัมพันธ์นี้ปริมาณของไอเท็มหรือแอททริบิวท์จะถูกแสดงให้อยู่ในรูปของช่วงข้อมูล ดังตัวอย่างกฎความสัมพันธ์เกี่ยวกับปริมาณข้างล่างนี้ โดยที่ X คือ ตัวแปรที่หมายถึงลูกค้า และแอททริบิวท์ที่เป็นปริมาณคือ อายุ และ รายได้

อายุ(X, “30 … 39”) ∧ รายได้(X, “42000 … 50000”) ⟶ ซื้อ(X, “รถยนต์”) (2.2)

3) กฎความสัมพันธ์แบบหนึ่งมิติ (Single-dimensional association rule)

เป็นกฎความสัมพันธ์ที่มีไอเท็มหรือแอททริบิวท์ภายในกฎความสัมพันธ์ มีการอ้างอิงกับข้อมูลเพียงหนึ่งมิติ เช่น กฎความสัมพันธ์ที่ (2.1) สามารถเขียนให้อยู่ในรูปแบบของกฎความสัมพันธ์แบบหนึ่งมิติได้ดังนี้

ซื้อ (X, นม) ⟶ ซื้อ(X, ขนมปัง) (2.3)

กฎความสัมพันธ์ที่ 2.3 เป็นกฎความสัมพันธ์แบบหนึ่งมิติ เนื่องจากมีการอ้างอิงถึงข้อมูลเพียงแค่หนึ่งมิติ นั่นคือ มิติ ซื้อ

4) กฎความสัมพันธ์หลายมิติ (Multi-dimensional association rule)

เป็นกฎความสัมพันธ์ที่มีไอเท็มหรือแอททริบิวท์ภายในกฎความสัมพันธ์ มีการอ้างอิงของข้อมูลมากกว่าสองมิติขึ้นไป ตัวอย่างเช่น กฎความสัมพันธ์ที่ (2.2) สามารถพิจารณาได้ว่าเป็นกฎความสัมพันธ์หลายมิติ เนื่องจากมีการอ้างถึงข้อมูลสามมิติด้วยกัน คือ อายุ รายได้ และ ซื้อ

5) กฎความสัมพันธ์หลายระดับ (Multilevel association rule)

การค้นหากฎความสัมพันธ์บางวิธีที่สามารถค้นหากฎที่มีระดับของนามธรรมที่แตกต่างกันได้ คือ ชุดของกฎความสัมพันธ์มีกฎความสัมพันธ์อื่นตามมาด้วย ตัวอย่างเช่น

อายุ (X, “30 … 39”) ⟶ ซื้อ (X, “พาหนะส่วนตัว”) (2.4)

อายุ (X, “30 … 39”) ⟶ ซื้อ (X, “รถจักรยานยนต์”) (2.5)

จากกฎความสัมพันธ์ที่ (2.4) และ (2.5) จะเห็นว่า ไอเท็มที่ถูกอ้างอิงอยู่ในระดับที่แตกต่างกัน นั่นคือ พาหนะส่วนตัว มีระดับที่สูงกว่า รถจักรยานยนต์ เนื่องจากพาหนะส่วนตัวสามารถจำแนกย่อยได้เป็น รถยนต์ รถจักรยานยนต์ และรถจักรยาน

6) กฎความสัมพันธ์ระดับเดียว (single-level association rule)

เป็นกฎความสัมพันธ์ที่มีลักษณะคล้ายกับกฎความสัมพันธ์หลายระดับ แตกต่างกันเพียงกฎความสัมพันธ์ระดับเดียวจะมีการอ้างอิงข้อมูลที่อยู่ในระดับเดียวกันเท่านั้น เช่น

อายุ (X, “30 … 39”) $\rightarrow$ ซื้อ (X, “เบียร์”) (2.6)

อายุ (X, “30 … 39”) $\rightarrow$ ซื้อ (X, “สุรา”) (2.7)

จากกฎความสัมพันธ์ที่ (2.6) และ (2.7) จะเห็นว่าไอเท็มที่ถูกอ้างอิงอยู่ในระดับเดียวกัน นั่นคือ เบียร์ มีระดับเดียวกันกับ สุรา

2.14 การค้นหากฎความสัมพันธ์ในฐานข้อมูลขนาดใหญ่

Agrawal, Imielinski, and Swami (1993) ได้เสนอแนวคิดการค้นหากฎความสัมพันธ์ของข้อมูลจากฐานข้อมูลที่มีขนาดใหญ่ โดยใช้รายการทรานแซคชัน ซึ่งเป็นข้อมูลการสั่งซื้อสินค้าของลูกค้า โดยอัลกอริทึมที่นำมาใช้ในการค้นหากฎความสัมพันธ์นี้คือ อัลกอริทึมเอไอเอส (AIS) โดยกฎความสัมพันธ์ที่ได้จะอยู่ในรูปแบบของ $I_K \rightarrow I_l$ โดยที่ I_K คือ ข้อมูลชุดที่ k, I_l คือ ข้อมูลชุดที่ l และ $I_K \cap I_l = \varnothing$, k = 1, 2, 3, … การทำงานของอัลกอริทึมนี้ จะเป็นการวนอ่านข้อมูลหลายรอบ โดยข้อมูลที่อ่านมาแต่ละรอบ จะเป็นข้อมูลรายการทรานแซคชันทั้งหมดที่มีอยู่ในฐานข้อมูล การสร้างไอเท็มเซตจะสร้างจากข้อมูลแต่ละเรคคอร์ดที่ถูกอ่านขึ้นมา โดยอาจจะเป็นข้อมูลเพียงบางส่วนหรืออาจจะเป็นข้อมูลทั้งหมดของรายการสินค้า ส่วนแคนดิเดทไอเท็มเซตจะถูกสร้างเพื่อใช้ในการตรวจสอบค่าสนับสนุนของกฎความสัมพันธ์ที่ได้มา โดยแคนดิเดทไอเท็มเซตของรอบถัดไปจะถูกสร้างมาจาก ไอเท็มเซตที่ปรากฏบ่อยของรอบก่อนหน้า รายละเอียดการทำงานของอัลกอริทึม เอไอเอส แสดงดังรูปที่ 2.18

```
Procedure LargeItemsets
Begin
Let Large set L = ∅;
Let Frontier set F = {∅};

while F ≠ ∅ do
        Let Candidate set C = ∅;
        forall database tuples t do
                forall itemsets f in F do
                        if t contains f then
                        let C_f = candidate itemsets that are extensions of f and contained in t;
                                forall itemsets c_f in C_f do
                                        if c_f ∈ C then
                                                c_f.count = c_f.count + 1;
                                        else
                                                c_f.count = 0;
                                                C = C + c_f;
                                end
                end
        end
end
let F = ∅;
        forall itemsets c in C do
                if count(c)/dbsize > minsupport then
                        L = L + c;
                if c should be used as a frontier in the next pass then
                        F = F + c;
        end
end
```

รูปที่ 2.18 อัลกอริทึมเอไอเอส (AIS algorithm)

2.15 อัลกอริทึมเอไพรออริ

Agrawal and Srikant (1994) ได้เสนอ อัลกอริทึมเอไพรออริ (Apriori) ซึ่งเป็นอัลกอริทึมสำหรับการทำเหมืองข้อมูลประเภทการค้นหาความสัมพันธ์ โดยอัลกอริทึมนี้ได้รับการยอมรับว่าเป็นอัลกอริทึมสำหรับการค้นหาความสัมพันธ์ ที่มีประสิทธิภาพเหนือกว่าอัลกอริทึมอื่น ๆ ที่เคยมีมา โดยชื่อของอัลกอริทึมถูกตั้งขึ้นตามหลักการทำงานของอัลกอริทึม นั่นคือ อัลกอริทึมเอไพรออริจะมีการทำงานเป็นแบบลูป (Loop) หรือวนรอบไปเรื่อย ๆ เป็นลำดับชั้น หรือที่เรียกว่า Level-wise โดยไอเท็มเซตที่ปรากฏบ่อยในรอบปัจจุบันจะนำไปสร้างเป็นแคนดิเดทไอเท็มเซตในรอบถัดไป

การทำงานของอัลกอริทึมเอไพรออริแบ่งออกเป็น 3 ช่วง คือ การสร้างไอเท็มเซตที่ปรากฏบ่อย ดังรูปที่ 2.19 การสร้างแคนดิเดทไอเท็มเซต ดังรูปที่ 2.20 การสร้างกฎความสัมพันธ์ ดังรูปที่ 2.21

Input: Database, D, of transaction; minimum support threshold, min_sup.

Output: L, frequent itemsets in D.

Method:

(1) L_1 = find_frquent_1-itemsets(D);

(2) **for** (k=2; $L_{k-1} \neq \varnothing$; k++) {

(3) C_k = **apriori_gen**(L_{k-1}, min_sup);

(4) **for each** transactions t $\in$ D { // scan D for counts

(5) C_t = subset (C_k, t); // get the subsets of t that are candidates

(6) **for each** candidates c $\in$ C_t {

(7) c.count++;

(8) }

(9) $L_k = \{c \in C_t \mid c.count \geq min\_sup\}$

(10) }

(11) **return** $\cup_k L_k$;

รูปที่ 2.19 การสร้างไอเท็มเซตที่ปรากฏบ่อย

Procedure apriori_gen(L_{k-1}: frequent (k-1)-itemsets; min_sup: minimum support threshold)

(1) **for each** itemset $l_1 \in L_{k-1}$

(2) **for each** itemset $l_2 \in L_{k-1}$

(3) **if** ($l_1[1]= l_2[1]$)^ ($l_1[2]= l_2[2]$)^…^ ($l_1[k-2]= l_2[k-2]$)^($l_1[k-1]= l_2[k-1]$) **then** {

(4) $c = l_1 \bowtie l_2$; // join step: generate candidates

(5) **if has_infrequent_subset**(c, L_{k-1}) **then**

(6) delete c; // prune step: remove unfruitful candidates

(7) **else add** c **to** C_k;

(8) }

(9) **return** $C_{k;}$

Procedure has_infrequent_subset(c: candidate k-itemset; L_{k-1}: frequent (k-1)-itemsets);

// use prior knowledge

(1) **for each** (k-1)-subset s **of** c

(2) **if** s $\notin L_{k-1}$ **then**

(3) **return** TRUE;

(4) **return** FALSE;

รูปที่ 2.20 การสร้างแคนดิเดทไอเท็มเซต

```
Precondition: all frequent itemsets have been computed
each itemset has two attributes:
count   : "support" as an integer
size    : cardinality, i.e. number of items in itemset
(1)     for each frequent k-itemset f ∈ F where k ≥ 2 {
(2)         generateRules(f, f);
(3)     }

Procedure generateRules(Itemset f, Itemset left)
(1)         rules.add(a, f\a); // set of result association rules
(2)         if (left.size-1 > 1) then
(3)                 generateRules(f, a);
```

รูปที่ 2.21 การสร้างกฎความสัมพันธ์

Dehaspe and De Raedt (1997) ได้นำอัลกอริทึมเอไพรออริมาใช้เพื่อหาความสัมพันธ์ของข้อมูล จากตารางหลายตาราง โดยเป็นการนำอัลกอริทึมเอไพรออริมาพัฒนาด้วยภาษาโปรล็อก ใช้ชื่อว่า WARMR ซึ่งทำงานบนฐานข้อมูลเชิงนิรนัย

Han et al. (1997) ได้พัฒนาระบบ DBMiner ที่ทำงานร่วมกับระบบฐานข้อมูลเชิงสัมพันธ์ โดยเป็นระบบที่มีความสามารถทางด้านการทำเหมืองข้อมูลที่หลากหลาย เช่น การจำแนกประเภทของข้อมูล (Classification) การทำนายผล (Prediction) และ การค้นหาความสัมพันธ์ของข้อมูล (Association) เป็นต้น

Thomas, Bodagala, Alsabti, and Ranka (1997) ได้นำอัลกอริทึมเอไพรออริมาทำงานร่วมกับระบบฐานข้อมูลเชิงสัมพันธ์ และได้พัฒนาอัลกอริทึม FUP (Fast UPdate) ขึ้นมา เพื่อให้ระบบสามารถค้นหากฎความสัมพันธ์ได้ใหม่เมื่อมีการเพิ่ม หรือลบข้อมูลในทรานแซคชัน

Saar Tsechansky, Pliskin, Rabinowitz, and Porath (1999) ได้นำอัลกอริทึมเอไพรออริมาใช้เพื่อค้นหารูปแบบของความสัมพันธ์ของข้อมูลของโรงพยาบาลแห่งหนึ่ง โดยเป็นการทำงานบนระบบฐานข้อมูลเชิงสัมพันธ์ ซึ่งทำการค้นหาความสัมพันธ์จากหลาย ๆ รีเลชัน

Berzal, Cubero, Marín, and Serrano (2001) ได้พัฒนาอัลกอริทึม Tree-Based Association Rule mining (TBAR) สำหรับการค้นหาความสัมพันธ์ของข้อมูลบนระบบฐานข้อมูลเชิงสัมพันธ์ โดยเก็บไอเท็มเซตที่ปรากฏบ่อยไว้ในรูปแบบของโครงสร้างต้นไม้ เพื่อช่วยลดระยะเวลาในการค้นหาความสัมพันธ์ความสัมพันธ์ของข้อมูล

Hipp, Güntze,r and Grimmer (2001) ได้นำอัลกอริทึมเอไพรออริมาพัฒนาด้วยภาษา C++ และเป็นการทำงานบนระบบฐานข้อมูลเชิงสัมพันธ์ (DB2 database system) เพื่อทำการค้นหาความสัมพันธ์ของข้อมูลให้กับบริษัท DaimlerChrysler

จากอดีตที่ผ่านมาจนถึงปัจจุบัน ยังไม่มีการนำเทคนิคการวิเคราะห์ความสัมพันธ์ของข้อมูลมาใช้สำหรับกระบวนการปรับรูปแบบบรรทัดฐาน งานวิจัยนี้จึงได้นำเทคนิคดังกล่าวมาใช้ ซึ่งรายละเอียดของขั้นตอนต่าง ๆ และวิธีดำเนินการวิจัยทั้งหมดนั้น จะกล่าวถึงในบทถัดไป

บทที่ 3
ระเบียบวิธีวิจัย

งานวิจัยนี้มีจุดมุ่งหมายที่จะพัฒนากระบวนการปรับรูปแบบบรรทัดฐานของตารางข้อมูลในฐานข้อมูลเชิงสัมพันธ์ โดยโปรแกรมที่พัฒนาขึ้นจะใช้ภาษา SQL และนำเอาเทคนิคการวิเคราะห์ความสัมพันธ์ ซึ่งเป็นงานทางด้านการทำเหมืองข้อมูลเข้ามาช่วย โดยอัลกอริทึมที่นำมาใช้เพื่อค้นหารูปแบบของข้อมูลเพื่อนำไปช่วยในการปรับรูปแบบบรรทัดฐาน คือ อัลกอริทึมเอไพรออริ (Apriori) ซึ่งเป็นอัลกอริทึมที่ใช้ในการทำเหมืองข้อมูลประเภทการค้นหากฎความสัมพันธ์ กฎความสัมพันธ์ที่นำมาใช้สำหรับกระบวนการปรับรูปแบบบรรทัดฐานต้องเป็นกฎที่ค่าความเชื่อมั่น 100 เปอร์เซ็นต์เท่านั้น รายละเอียดเนื้อหาในบทนี้ประกอบไปด้วย ขั้นตอนการวิจัย ปรากฏอยู่ในหัวข้อที่ 3.1 โปรแกรม NoWARs เพื่อการค้นหากฎความสัมพันธ์และการปรับรูปแบบบรรทัดฐาน ปรากฏอยู่ในหัวข้อที่ 3.2 และการทำงานของโปรแกรม NoWARs ปรากฏอยู่ในหัวข้อที่ 3.3

3.1 ขั้นตอนการวิจัย

แนวทางการวิจัยของงานวิจัยนี้จะประกอบไปด้วยการออกแบบและพัฒนาขั้นตอนในการปรับรูปแบบบรรทัดฐาน ด้วยการใช้เทคนิคการวิเคราะห์ความสัมพันธ์ โดยมีรายละเอียดขั้นตอนวิธีดังต่อไปนี้

1) ศึกษาและรวบรวมงานวิจัยที่เกี่ยวข้อง
2) ศึกษาขั้นตอนวิธีและกระบวนการปรับรูปแบบบรรทัดฐาน
3) ศึกษาขั้นตอนการทำงานของอัลกอริทึมเอไพรออริ โดยได้ทำการศึกษาจากโปรแกรม WEKA พัฒนาโดย Frank, Hall, Holmes, Martin, Mayo, Pfahringer, Smith, and Witten (2008) (http://www.cs.waikato.ac.nz/ml/weka/) ซึ่งเป็นโปรแกรมสำเร็จรูปที่เปิดเผยซอร์สโค้ด (Open-source environment) ที่ใช้วิเคราะห์หารูปแบบความสัมพันธ์ของข้อมูลในงานทางด้านการทำเหมืองข้อมูล
4) ออกแบบโครงสร้างข้อมูลทดสอบที่เหมาะสมกับการปรับรูปแบบบรรทัดฐานด้วยเทคนิคการวิเคราะห์ความสัมพันธ์

5) ออกแบบอัลกอริทึมสำหรับการปรับรูปแบบบรรทัดฐานด้วยเทคนิคการวิเคราะห์กฎความสัมพันธ์
6) พัฒนาโปรแกรมตามที่ได้ออกแบบอัลกอริทึมไว้
7) ทดสอบโปรแกรมกับข้อมูลที่ได้ออกแบบไว้
8) วิเคราะห์และสรุปผลการวิจัย

3.2 โปรแกรม NoWARs เพื่อการวิเคราะห์กฎความสัมพันธ์และการปรับรูปแบบบรรทัดฐาน

เนื้อหาในส่วนนี้จะกล่าวถึงการออกแบบอัลกอริทึมเพื่อการวิเคราะห์กฎ และการนำแนวคิดและอัลกอริทึมที่ได้ออกแบบไว้มาพัฒนาเป็นโปรแกรม NoWARs โดยเป็นการนำเทคนิควิธีที่มีอยู่แล้ว คือ เทคนิคการวิเคราะห์ความสัมพันธ์ มาประยุกต์ใช้ร่วมกับเทคนิคการปรับรูปแบบบรรทัดฐาน เพื่อพัฒนาเป็นโปรแกรมสำหรับการปรับรูปแบบบรรทัดฐาน

3.2.1 อัลกอริทึม NoWARs: Normalization With Association Rules

อัลกอริทึม NoWARs เป็นการนำเทคนิคการปรับรูปแบบบรรทัดฐานมาประยุกต์ใช้ร่วมกับเทคนิคการวิเคราะห์ความสัมพันธ์ โดยอัลกอริทึมสำหรับการวิเคราะห์ความสัมพันธ์ของข้อมูลที่นำมาใช้ คือ อัลกอริทึมเอไพรออริ ภายในอัลกอริทึม NoWARs ประกอบไปด้วยส่วนสำคัญสามส่วน คือ ส่วนที่ทำหน้าที่ในการวิเคราะห์หาความสัมพันธ์ของข้อมูลซึ่งจะให้ผลลัพธ์ออกมาในรูปกฎความสัมพันธ์ ส่วนที่ใช้ในการคัดเลือกกฎความสัมพันธ์เพื่อนำไปใช้ในการปรับรูปแบบบรรทัดฐาน และส่วนที่ทำหน้าที่ในการแปลงกฎความสัมพันธ์ที่ได้ให้อยู่ในรูปของ Relational Schema ซึ่งเป็นสคีมาของตารางฐานข้อมูลที่ใช้ในการออกแบบฐานข้อมูล รายละเอียดของอัลกอริทึม NoWARs แสดงดังรูปที่ 3.1 และแสดงขั้นตอนการทำงานโดยรวมของโปรแกรม NoWARs ดังรูปที่ 3.2

Algorithm: NoWARs

//Input: Data in table format

//Output: Relational schema

(1) **Set** Minimum Confidence value = 1.0

(2) **Set** Minimum Support value = 0.0 and Maximum Support value = 1.0

(3) **Call** Apriori to obtain association rules

(4) **For** every rule from i to n **do**

(5) Select one cause rule with max support value

(6) Select primary key of relation (2NF)

(7) Find corresponding attribute

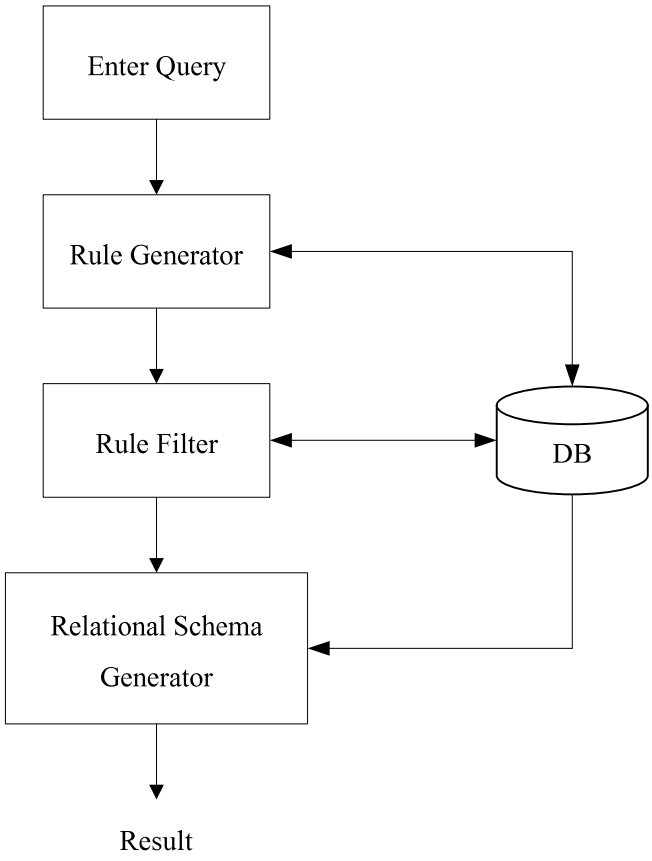
(8) Select primary key of relation (3NF)

(9) Compare non key attribute in (8) with all attributes in (5) then cut it off and we get one 3NF relation

(10) Compare all non key attributes with 1NF relation and cut it off

(11) Combine all primary keys of 2NF with the remaining attribute in (10) to become the last 3NF

รูปที่ 3.1 อัลกอริทึม NoWARs



รูปที่ 3.2 การทำงานโดยรวมของโปรแกรม NoWARs

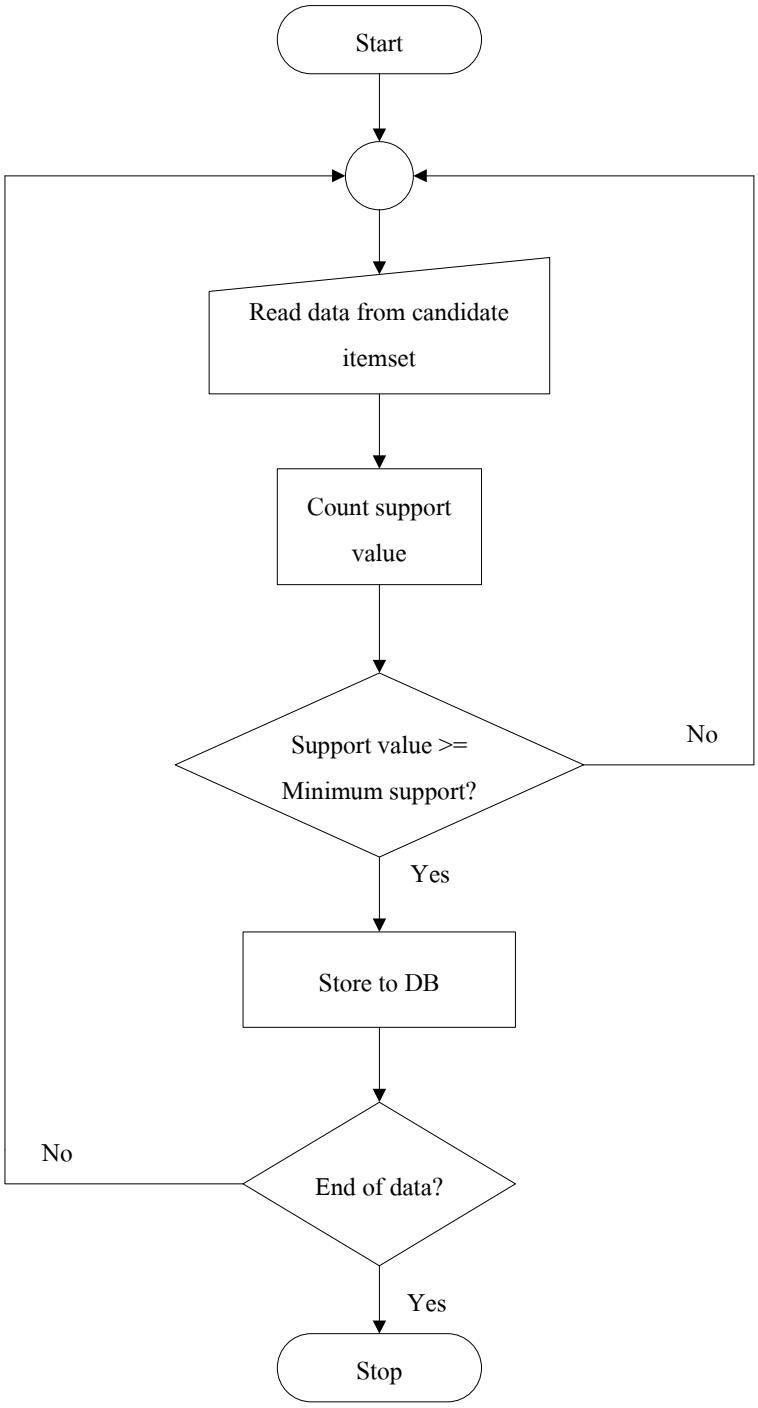
3.2.2 การสร้างแคนดิเดทไอเท็มเซต (Candidate itemset)

การสร้างแคนดิเดทไอเท็มเซตในรอบแรกจะสร้างมาจาก ไอเท็มที่ปรากฏอยู่ในรายการทรานแซคชันทั้งหมด โดยในรอบแรกจะเป็นเซตของไอเท็มที่มีสมาชิกเพียงหนึ่งตัวเท่านั้น จากนั้นจึงนับค่าสนับสนุนของแต่ละไอเท็มเซต เพื่อทำการเปรียบเทียบกับค่าสนับสนุนขั้นต่ำ ซึ่งเราจะเรียกไอเท็มเซตที่มีค่าสนับสนุนมากกว่าหรือเท่ากับค่าสนับสนุนขั้นต่ำว่า ไอเท็มเซตที่ปรากฏบ่อย (Large itemset) โดยไอเท็มเซตที่ปรากฏบ่อยในรอบแรกจะนำไปใช้สร้างแคนดิเดทไอเท็มเซตในรอบถัดไป และการทำงานของอัลกอริทึมก็จะทำการวนรอบลักษณะนี้ไปเรื่อย ๆ จนกระทั่งไม่สามารถสร้างแคนดิเดทไอเท็มเซตในระดับถัดไปได้

3.2.3 การสร้างไอเท็มเซตที่ปรากฏบ่อย (Large itemset)

การสร้างไอเท็มเซตที่ปรากฏบ่อยจะสร้างมาจากแคนดิเดทไอเท็มเซต โดยพิจารณาจากค่าสนันสนุนของแต่ละแคนดิเดทไอเท็มเซตว่า มีค่ามากกว่าหรือเท่ากับค่าสนันสนุนขั้นต่ำที่กำหนดไว้หรือไม่ ถ้าใช่ก็จะนำเซตนั้นไปสร้างเป็นไอเท็มเซตที่ปรากฏบ่อยและทำการ

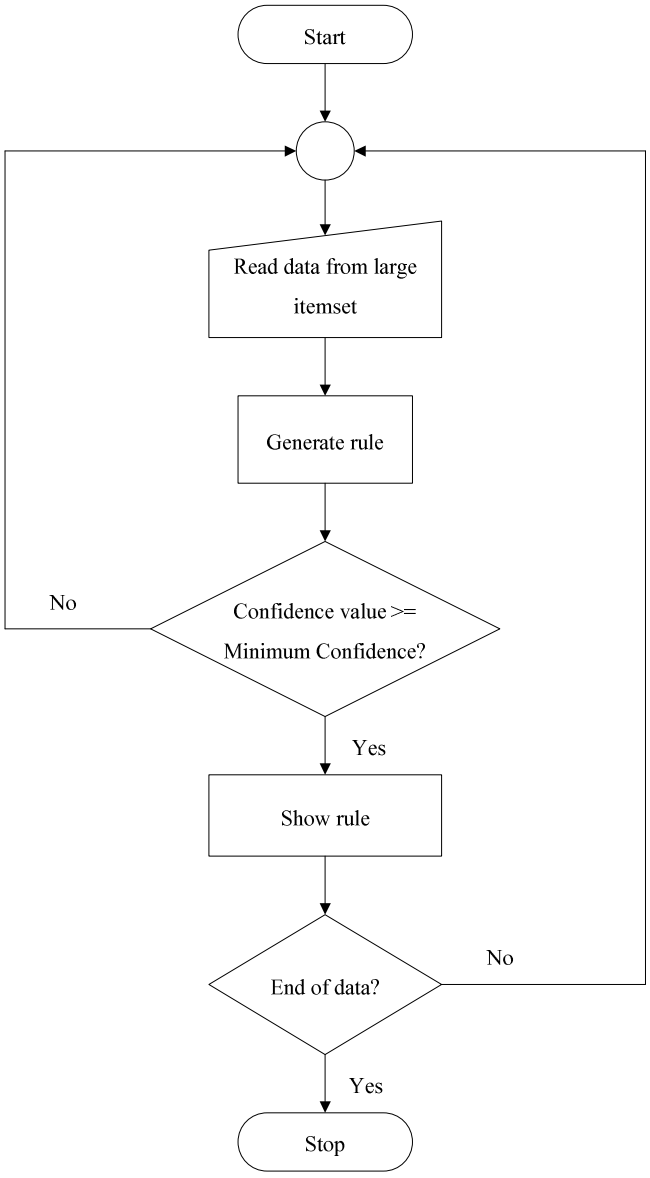
บันทึกลงในตาราง ดังรูปที่ 3.3 แต่ถ้าไม่ใช่ก็จะทำการตัดเซตนั้นทิ้ง ไม่นำมาเป็นไอเท็มเซตที่ปรากฏบ่อย รวมทั้งไม่นำมาพิจารณาสร้างเป็นแคนดิเดทไอเท็มเซตในระดับถัดไปด้วย เนื่องจากแคนดิเดทไอเท็มเซตระดับถัดไปจะสร้างมาจากไอเท็มเซตที่ปรากฏบ่อยในระดับก่อนหน้านี้



รูปที่ 3.3 ขั้นตอนการสร้างไอเท็มเซตที่ปรากฏบ่อย

3.2.4 การสร้างกฎความสัมพันธ์

การสร้างกฎความสัมพันธ์จะสร้างมาจากไอเท็มเซตที่ปรากฏบ่อย โดยนำไอเท็มเซตที่ปรากฏบ่อยแต่ละชุดมาสร้างเป็นกฎความสัมพันธ์ที่เป็นไปได้ทั้งหมด กฎความสัมพันธ์ที่ถูกสร้างขึ้นมาจะอยู่ในรูปแบบ ถ้า $\rightarrow$ แล้ว โดยกฎความสัมพันธ์ที่แสดงออกมาจะแสดงเฉพาะกฎความสัมพันธ์ที่มีค่าความเชื่อมั่นและค่าสนับสนุน มากกว่าหรือเท่ากับค่าความเชื่อมั่นและค่าสนับสนุนขั้นต่ำที่กำหนดไว้เท่านั้น และทุก ๆ กฎความสัมพันธ์ต้องเป็นกฎที่มีค่าความน่าเชื่อถือ 100 เปอร์เซ็นต์อีกด้วย ขั้นตอนการสร้างกฎความสัมพันธ์แสดงดังรูปที่ 3.4



รูปที่ 3.4 ขั้นตอนการสร้างกฎความสัมพันธ์

3.2.5 เทคนิคการคัดเลือกกฎความสัมพันธ์เพื่อการปรับรูปแบบบรรทัดฐาน

สำหรับเนื้อหาในส่วนนี้จะเป็นการนำเสนอเทคนิคการคัดเลือกกฎความสัมพันธ์ที่ได้จากการวิเคราะห์ความสัมพันธ์ของข้อมูล เพื่อนำมาใช้สำหรับการปรับรูปแบบบรรทัดฐาน การทำงานในส่วนจะเริ่มทำงานหลังจากที่โปรแกรมทำการค้นหากฎความสัมพันธ์ทั้งหมดจากข้อมูลได้แล้ว โดยกฎความสัมพันธ์ที่ได้จะอยู่ในรูปแบบของแอททริบิวท์เหตุไปสู่แอททริบิวท์ผล ตามรูปแบบของการค้นหาความสัมพันธ์ด้วยอัลกอริทึมเอไพรออริ ดังนี้

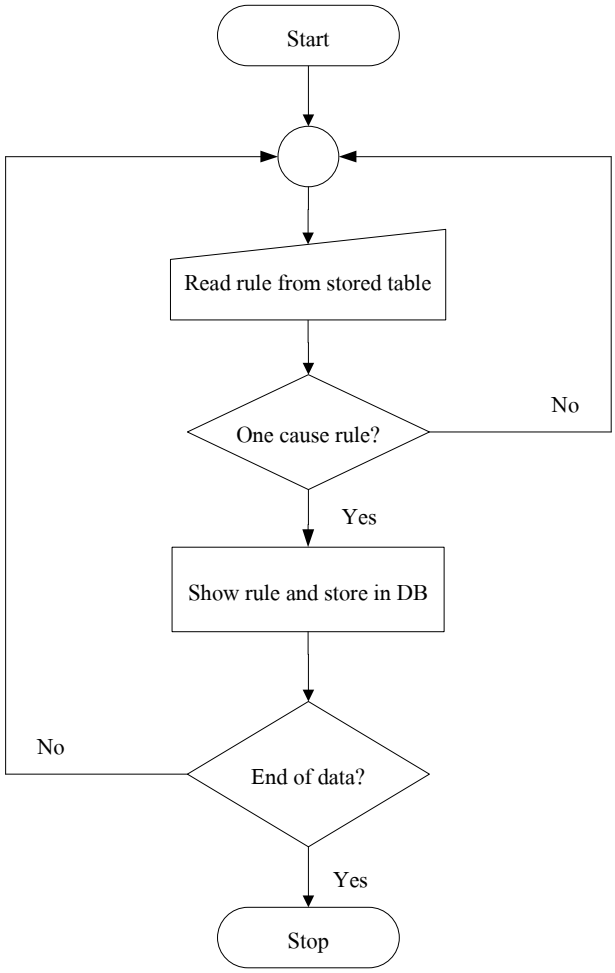
IF attribute1 = ‘value1’ THEN attribute2 = ‘value2’

โดยปกติแล้วกฎความสัมพันธ์ที่ได้อาจมีส่วนที่เป็นเหตุมากกว่าหนึ่งแอททริบิวท์และส่วนที่เป็นผลมากกว่าหนึ่งแอททริบิวท์เช่นกัน แต่สำหรับการปรับรูปแบบบรรทัดฐานในงานวิจัยนี้จะพิจารณาเฉพาะกฎที่มีส่วนที่เป็นเหตุที่มีเพียงหนึ่งแอททริบิวท์เท่านั้น โดยส่วนที่เป็นผลตามมาสามารถมีกี่แอททริบิวท์ก็ได้ ซึ่งเป็นผลมาจากการพิจารณาส่วนที่เป็นเหตุเป็นคีย์หลักของรีเลชันนั่นเอง และกฎที่จะนำมาใช้ในการปรับรูปแบบบรรทัดฐานต้องเป็นกฎที่มีรูปแบบซ้ำ ๆ กันอีกด้วย ยกตัวอย่างเช่น

1. A=a1 => B=b1
2. A=a2 => B=b2
3. A=a3 => C=c1
4. A=a4 => D=d1
5. A=a5 => B=b3

จากตัวอย่างข้างต้นจะเห็นว่า กฎความสัมพันธ์ที่สามารถบอกได้ว่าแอททริบิวท์ C ขึ้นอยู่กับแอททริบิวท์ A ปรากฏแค่ครั้งเดียว คือ กฎความสัมพันธ์ในข้อที่ 3 และกฎความสัมพันธ์ที่สามารถบอกได้ว่าแอททริบิวท์ D ขึ้นกับแอททริบิวท์ A ก็ปรากฏเพียงแค่ครั้งเดียวเช่นกัน คือ กฎความสัมพันธ์ในข้อที่ 4 ส่วนกฎความสัมพันธ์ที่บอกได้ว่าแอททริบิวท์ B ขึ้นกับแอททริบิวท์ A ปรากฏอยู่ถึงสามแห่ง คือ กฎความสัมพันธ์ในข้อที่ 1 2 และ 5 ซึ่งเราจะเรียกกฎความสัมพันธ์ลักษณ์นี้ว่า กฎความสัมพันธ์ที่มีรูปแบบที่ซ้ำกัน

เมื่อได้กฎความสัมพันธ์ที่ต้องการแล้ว โปรแกรมจะทำการเก็บกฎความสัมพันธ์เหล่านั้นลงฐานข้อมูลไว้ใช้ในขั้นตอนต่อไป ในขั้นตอนการค้นหากฎความสัมพันธ์เพื่อนำไปใช้ในการปรับรูปแบบบรรทัดฐานสามารถแสดงได้ดังรูปที่ 3.5



รูปที่ 3.5 ขั้นตอนการคัดเลือกกฎความสัมพันธ์

3.3 การทำงานของโปรแกรม NoWARs

สำหรับเนื้อหาในส่วนนี้ จะเป็นการอธิบายการทำงานของโปรแกรม NoWARs ที่ได้พัฒนาขึ้น เพื่อสร้างความเข้าใจในการทำงานของอัลกอริทึมด้วยตัวอย่าง ดังต่อไปนี้

ตัวอย่าง พิจารณารายการขายสินค้าของร้านขายสินค้าอิเล็กทรอนิกส์แห่งหนึ่ง ซึ่งมีสคีมาของตารางสินค้า ดังนี้

Order (Inv, C_ID, C_Name, P_ID, P_TYPE, QTY)

รายการสั่งซื้อของลูกค้าแสดงดังตารางที่ 3.1 แต่ละแถวแสดงถึงรายการสั่งสินค้าของลูกค้าแต่ละคน ซึ่งหนึ่งแถวแทนสินค้าหนึ่งชนิดเท่านั้น โดยลูกค้าแต่ละคนสามารถที่จะซื้อสินค้ากี่ชนิดก็ได้ ในการค้นหากฎความสัมพันธ์จะค้นหาเฉพาะกฎความสัมพันธ์ที่มีค่าสนับสนุนตั้งแต่ 0 เป็นต้นไป ส่วนค่าความเชื่อมั่นต้องเป็น 100 เปอร์เซ็นต์เท่านั้น

ตารางที่ 3.1 รายการสั่งซื้อ

INV	DATE	C_ID	C_NAME	P_ID	P_TYPE	QTY
INV001	20-08-2008	C001	Somjit	P001	Refrigerator	10
INV001	20-08-2008	C001	Somjit	P002	TV	15
INV002	20-08-2008	C002	Deejai	P004	Phone	20
INV003	21-10-2008	C002	Deejai	P001	Refrigerator	5
INV004	21-10-2008	C003	Somjit	P003	Fan	11
INV004	21-10-2008	C003	Somjit	P004	Phone	30

จากตัวอย่างข้อมูลรายการสั่งซื้อสินค้าดังตารางที่ 3.1 เมื่อส่งข้อมูลเข้าไปยังโปรแกรม NoWARs ส่วนของโปรแกรมที่ทำหน้าที่ในการวิเคราะห์หาความสัมพันธ์ของข้อมูล จะแสดงความสัมพันธ์ที่ได้ในรูปของกฎความสัมพันธ์ที่เรียกว่า One cause rule ดังนี้

1. INV=INV001 2 ==> DATE=20-08-2008 2
2. C_ID=C001 2 ==> INV=INV001 2
3. INV=INV001 2 ==> C_ID=C001 2
4. INV=INV001 2 ==> C_NAME=Somjit 2
5. INV=INV004 2 ==> DATE=21-10-2008 2
6. C_ID=C003 2 ==> INV=INV004 2
7. INV=INV004 2 ==> C_ID=C003 2
8. INV=INV004 2 ==> C_NAME=Somjit 2

รูปที่ 3.6 กฎความสัมพันธ์จากการวิเคราะห์ความสัมพันธ์

9. C_ID=C001 2 ==> DATE=20-08-2008 2

10. C_ID=C003 2 ==> DATE=21-10-2008 2

11. C_ID=C001 2 ==> C_NAME=Somjit 2

12. C_NAME=Deejai 2 ==> C_ID=C002 2

13. C_ID=C002 2 ==> C_NAME=Deejai 2

14. C_ID=C003 2 ==> C_NAME=Somjit 2

15. P_TYPE=Refrigerator 2 ==> P_ID=P001 2

16. P_ID=P001 2 ==> P_TYPE=Refrigerator 2

17. P_TYPE=Phone 2 ==> P_ID=P004 2

18. P_ID=P004 2 ==> P_TYPE=Phone 2

19. C_ID=C001 2 ==> INV=INV001 DATE=20-08-2008 2

20. INV=INV001 2 ==> DATE=20-08-2008 C_ID=C001 2

21. INV=INV001 2 ==> DATE=20-08-2008 C_NAME=Somjit 2

22. C_ID=C001 2 ==> INV=INV001 C_NAME=Somjit 2

23. INV=INV001 2 ==> C_ID=C001 C_NAME=Somjit 2

24. C_ID=C003 2 ==> INV=INV004 DATE=21-10-2008 2

25. INV=INV004 2 ==> DATE=21-10-2008 C_ID=C003 2

26. INV=INV004 2 ==> DATE=21-10-2008 C_NAME=Somjit 2

27. C_ID=C003 2 ==> INV=INV004 C_NAME=Somjit 2

28. INV=INV004 2 ==> C_ID=C003 C_NAME=Somjit 2

29. C_ID=C001 2 ==> DATE=20-08-2008 C_NAME=Somjit 2

30. C_ID=C003 2 ==> DATE=21-10-2008 C_NAME=Somjit 2

31. C_ID=C001 2 ==> INV=INV001 DATE=20-08-2008 C_NAME=Somjit 2

32. INV=INV001 2 ==> DATE=20-08-2008 C_ID=C001 C_NAME=Somjit 2

33. C_ID=C003 2 ==> INV=INV004 DATE=21-10-2008 C_NAME=Somjit 2

34. INV=INV004 2 ==> DATE=21-10-2008 C_ID=C003 C_NAME=Somjit 2

รูปที่ 3.6 กฎความสัมพันธ์จากการวิเคราะห์ความสัมพันธ์ (ต่อ)

กฎความสัมพันธ์ที่แสดงในรูปที่ 3.6 ทุกกฎเป็นกฎความสัมพันธ์ที่มีค่าความเชื่อมั่น 100 เปอร์เซ็นต์ และมีค่าสนับสนุนตั้งแต่ 0 เป็นต้นไป โดยทุกกฎมีส่วนที่เป็นเหตุเพียงหนึ่งแอททริบิวท์เท่านั้น ส่วนที่เป็นผลตามมาจะมีกี่แอททริบิวท์ก็ได้ขึ้นอยู่กับความสัมพันธ์ระหว่างแอททริบิวท์ที่เป็นเหตุและแอททริบิวท์อื่นในตารางข้อมูล การอธิบายกฎความสัมพันธ์จะยกตัวอย่างกฎความสัมพันธ์ลำดับที่ 11 มาอธิบาย ซึ่งสามารถอธิบายได้ดังนี้

11. C_ID=C001 2 ==> C_NAME=Somjit 2

เมื่อวิเคราะห์กฎความสัมพันธ์ลำดับที่ 11 แล้ว จะเห็นว่าเป็นกฎความสัมพันธ์ที่มีส่วนที่เป็นเหตุเพียงหนึ่งแอททริบิวท์ (C_ID=C001) และส่วนที่เป็นผลตามมาก็มีหนึ่งแอททริบิวท์เช่นกัน (C_NAME=Somjit) ตัวเลข 2 ที่ปรากฏอยู่หลัง C_ID=C001 และ C_NAME=Deejai หมายความว่าในตารางมีข้อมูลที่ตรงกับเงื่อนไขนี้อยู่ 2 รายการ หรืออาจตีความหมายจากกฎความสัมพันธ์นี้ได้เป็น แอททริบิวท์ C_ID จะเป็นตัวระบุแอททริบิวท์ C_NAME (สามารถระบุได้ว่า ลูกค้าที่มีหมายเลขประจำตัว (C_ID) เป็น C001 คือ ลูกค้าที่ชื่อ (C_NAME) Somjit)

หลังจากผ่านกระบวนการสร้างกฎความสัมพันธ์แล้ว โปรแกรมจะทำการค้นหารูปแบบของกฎความสัมพันธ์ที่มีลักษณะเหมือน ๆ กัน เช่น กฎความสัมพันธ์ลำดับที่ 11 เป็นกฎความสัมพันธ์ที่บอกว่าแอททริบิวท์ C_ID สามารถระบุแอททริบิวท์ C_NAME ได้ โปรแกรมก็จะพิจารณาว่ามีกฎไหนอีกบ้างที่สามารถอธิบายข้อมูลลักษณะนี้ได้ ซึ่งจากรูปที่ 3.6 จะเห็นว่ากฎความสัมพันธ์ลำดับที่ 13 และกฎความสัมพันธ์ลำดับที่ 14 ก็สามารถบอกความสัมพันธ์ลักษณะนี้ได้ โดยโปรแกรมจะทำการจัดเก็บกฎความสัมธ์ที่มีลักษณะเหมือน ๆ กันลงในตาราง ดังแสดงในตารางที่ 3.2

ในขั้นตอนต่อจากนี้จะเป็นการพิจารณาหากฎความสัมพันธ์ในลักษณะที่ว่า ส่วนที่เป็นเหตุสามารถระบุส่วนที่เป็นผลตามมาได้มากที่สุด จากตารางที่ 3.2 จะเห็นว่ากฎความสัมพันธ์ลำดับที่ 7 และ 13 เป็นกฎความสัมพันธ์ที่มีส่วนที่เป็นเหตุเพียง 1 แอททริบิวท์ แต่สามารถระบุส่วนที่เป็นผลตามมาได้ถึง 3 แอททริบิวท์

ตารางที่ 3.2 รูปแบบกฎความสัมพันธ์ที่พบบ่อย

ลำดับที่	กฎความสัมพันธ์	จำนวนกฎที่มีรูปแบบเหมือนกัน
1	INV ==> DATE	4
2	INV ==> C_ID	4
3	INV ==> C_NAME	4
4	INV ==> DATE, C_ID	4
5	INV ==> DATE, C_NAME	4
6	INV ==> C_ID, C_NAME	4
7	INV ==> DATE, C_ID, C_NAME	4
8	C_ID ==> INV	4
9	C_ID ==> DATE	4
10	C_ID ==> C_NAME	6
11	C_ID ==>INV, DATE	4
12	C_ID ==>INV, C_NAME	4
13	C_ID ==> DATE, INV, C_NAME	4
14	C_NAME ==> C_ID	2
15	P_ID ==> P_TYPE	4
16	P_TYPE ==> P_ID	4

จากตารางที่ 3.2 จะเห็นว่ากฎความสัมพันธ์ลำดับที่ 7 และ 13 ทั้งสองกฎต่างก็มีจำนวนกฎที่มีรูปแบบเหมือนกันเท่ากัน คือ 4 เท่ากัน แต่หากพิจารณาแอททริบิวท์ INV และแอททริบวท์ C_ID ในตารางที่ 3.1 จะเห็นว่า แอททริบิวท์ที่สมควรที่จะเป็นคีย์หลักของรีเลชัน คือ แอททริบิวท์ INV ที่มีค่าในแอททริบิวท์ซ้ำกันน้อยที่สุด ดังนั้นกฎที่ควรนำมาใช้สำหรับการปรับรูปแบบบรรทัดฐาน คือ กฎความสัมพันธ์ลำดับที่ 7

7. INV ==> DATE, C_ID, C_NAME

จากกฎความสัมพันธ์ลำดับที่ 7 สามารถแปลงให้อยู่ในรูปแบบของรีเลชันนอลสคีมา ซึ่งเป็นรีเลชันที่อยู่ในรูปแบบบรรทัดฐานระดับที่ 2 ได้ดังนี้

Order (<u>INV</u>, DATE, C_ID, C_NAME)

รีเลชัน Order และข้อมูลในรีเลชันหลังการปรับรูปแบบบรรทัดฐานในขั้นนี้แล้ว สามารถแสดงได้ดังตารางที่ 3.3

ตารางที่ 3.3 Order

INV	**DATE**	**C_ID**	**C_NAME**
INV001	20-08-2008	C001	Somjit
INV002	20-08-2008	C002	Deejai
INV003	21-10-2008	C002	Deejai
INV004	21-10-2008	C003	Somjit

เมื่อพิจารณาแอททริบิวท์ส่วนที่เป็นผลตามมาของกฎความสัมพันธ์ลำดับที่ 7 จะเห็นว่าประกอบไปด้วยแอททริบิวท์ทั้งหมดจำนวน 3 แอทรริบิวท์ด้วยกัน คือ DATE C_ID C_NAME และหากพิจารณาตารางที่ 3.2 ประกอบด้วย จะพบว่ามีกฎความสัมพันธ์ลำดับที่ 10 ที่สามารถบ่งบอกได้ว่าแอททริบิวท์ C_ID เป็นตัวระบุแอททริบิวท์ C_NAME ในขณะเดียวกันกฎความสัมพันธ์ลำดับที่ 14 ก็สามารถบ่งบอกได้ว่า C_NAME ก็สามารถเป็นตัวระบุ C_ID ได้เช่นกัน ในกรณีนี้ต้องทำการเลือกว่า จะใช้แอททริบิวท์ไหนเป็นคีย์หลักของรีเลชัน และถ้าหากพิจารณาจำนวนกฎที่มีรูปแบบเหมือนกัน ของกฎความสัมพันธ์ลำดับที่ 10 และ 14 จะพบว่า กฎความสัมพันธ์ลำดับที่ 10 มีจำนวนกฎที่มีรูปแบบเหมือนกันมากกว่ากฎความสัมพันธ์ลำดับที่ 14 ดังนั้นกฎความสัมพันธ์ที่สมควรนำมาใช้คือ กฎความสัมพันธ์ลำดับที่ 10 ซึ่งโปรแกรมจะกำหนดให้กฎความสัมพันธ์นี้เป็นตารางที่อยู่ในรูปแบบบรรทัดฐานระดับที่ 3 โดยแสดงให้อยู่ในรูปสคีมาได้ดังนี้

Customer (C_ID, C_NAME)

รีเลชัน Customer และข้อมูลในรีเลชันหลังการปรับรูปแบบบรรทัดฐานในขั้นนี้แล้วสามารถแสดงได้ดังตารางที่ 3.4

ตารางที่ 3.4 Customer

C_ID	**C_NAME**
C001	Somjit
C002	Deejai
C003	Somjit

พิจารณารีเลชัน Customer จะพบว่า เป็นรีเลชันที่ประกอบไปด้วยแอททริบิวท์ทั้งหมดจำนวน 2 แอททริบิวท์ ได้แก่ แอททริบิวท์ C_ID ทำหน้าที่เป็นคีย์หลักของรีเลชัน และแอททริบิวท์ C_NAME ซึ่งเป็นแอททริบิวท์ที่ไม่ใช่คีย์ที่ขึ้นกับแอททริบิวท์ C_ID และเมื่อนำแอททริบิวท์ C_NAME ไปเปรียบเทียบกับรีเลชันก่อนหน้านี้ ซึ่งก็คือ รีเลชัน Order จะเห็นว่า รีเลชัน Order เองก็มีแอททริบิวท์ C_NAME ด้วยเหมือนกัน โปรแกรมก็จะทำการตัดแอททริบิวท์ C_NAME ออกจากรีเลชัน Order และสร้างรีเลชันใหม่ ซึ่งเป็นรีเลชันที่อยู่ในรูปแบบบรรทัดฐานระดับที่ 3 ได้อีกหนึ่งรีเลชัน ดังนี้

Purchase (INV, DATE, C_ID)

รีเลชัน Purchase และข้อมูลในรีเลชันหลังการปรับรูปแบบบรรทัดฐานในขั้นนี้แล้วสามารถแสดงได้ดังตารางที่ 3.5

ตารางที่ 3.5 Purchase

INV	**DATE**	**C_ID**
INV001	20-08-2008	C001
INV002	20-08-2008	C002
INV003	21-10-2008	C002
INV004	21-10-2008	C003

เมื่อพิจารณาแอททริบิวท์ที่เป็นส่วนสาเหตุของกฎความสัมพันธ์ ในตารางที่ 3.2 จะเห็นว่าแอททริบิวท์ INV และ C_ID ล้วนถูกนำไปใช้ในการสร้างรีเลชัน และต่างก็ทำหน้าที่เป็นคีย์หลักของรีเลชันต่าง ๆ ดังที่กล่าวมาหมดแล้ว และหากพิจารณากฎความสัมพันธ์ลำดับที่ 14 เพื่อนำมาสร้างเป็นรีเลชันในรูปแบบบรรทัดฐานระดับที่ 2 อันต่อไป จะพบว่ากฎความสัมพันธ์นี้ไม่สมควรนำมาสร้างเป็นรีเลชันใหม่ เนื่องจากมีกฎความสัมพันธ์ลักษณะคล้ายกัน คือ กฎความสัมพันธ์ลำดับที่ 10 ซึ่งถูกนำไปใช้ในการสร้างเป็นรีเลชันในรูปแบบบรรทัดฐานระดับที่ 3 แล้ว หากนำกฎความสัมพันธ์ลำดับที่ 14 มาสร้างเป็นรีเลชันใหม่อีก ก็จะทำให้เกิดความขัดแย้งกันได้ ดังนั้นกฎความสัมพันธ์ที่ควรนำมาใช้ในการสร้างรีเลชันใหม่คือ กฎความสัมพันธ์ลำดับที่ 15 และ 16 โดยมีรูปแบบของกฎความสัมพันธ์ดังนี้

15. P_ID ==> P_TYPE

16. P_TYPE ==> P_ID

พิจารณากฎความสัมพันธ์ลำดับที่ 15 จะเห็นว่า P_ID เป็นตัวระบุ P_TYPE หรืออาจกล่าวได้ว่า P_TYPE ขึ้นอยู่กับ P_ID และกฎความสัมพันธ์ลำดับที่ 16 ก็สามารถบอกได้ว่า P_TYPE เป็นตัวระบุ P_ID หรืออาจกล่าวได้ว่า P_ID ขึ้นอยู่กับ P_TYPE ดังนั้น P_TYPE และ P_ID จึงมีคุณสมบัติเป็นคีย์คู่แข่งซึ่งกันและกัน ในกรณีนี้โปรแกรมจะเลือกกฎความสัมพันธ์ลำดับที่ 15 มาใช้ในการปรับรูปแบบบรรทัดฐาน ซึ่งสามารถเขียนให้อยู่ในรูปของรีเลชันนอลสคีมา ได้ดังนี้

Product (P_ID, P_TYPE)

รีเลชัน Product และข้อมูลในรีเลชันหลังการปรับรูปแบบบรรทัดฐานในขั้นนี้แล้ว สามารถแสดงได้ดังตารางที่ 3.6

ตารางที่ 3.6 Product

P_ID	**P_TYPE**
P001	Refrigerator
P002	TV
P003	Fan
P004	Phone

เมื่อพิจารณารีเลชัน Product จะพบว่าเป็นรีเลชันที่มีเพียง 2 แอททริบิวท์ โดยมีแอททริบิวท์ P_ID ทำหน้าที่เป็นคีย์หลักของรีเลชัน และมีแอททริบิวท์ P_TYPE ซึ่งเป็นแอททริบิวท์ที่ไม่ใช่คีย์ที่ขึ้นกับแอททริบิวท์ P_ID และหากพิจารณาคุณสมบัติของรีเลชันตามกระบวนการปรับรูปแบบบรรทัดฐาน จะเห็นว่ารีเลชัน Product เป็นรีเลชันที่อยู่ในรูปแบบบรรทัดฐานระดับที่ 3 แล้ว เนื่องจากไม่มีปัญหาเรื่องข้อมูลกลุ่มซ้ำ ไม่มีการขึ้นต่อกันเพียงบางส่วน รวมทั้งไม่มีการขึ้นต่อกันแบบทรานซิทีฟในรีเลชันนี้

ในขั้นตอนสุดท้ายของกระบวนการปรับรูปแบบบรรทัดฐานของอัลกอริทึม NoWARs คือการนำคีย์หลักของรีเลชันในรูปแบบบรรทัดฐานระดับที่ 2 ทั้งสองรีเลชันที่ได้ทำการค้นหาแล้วในขั้นตอนก่อนหน้านี้ มาประกอบกันเป็นคีย์หลักเพื่อสร้างรีเลชันใหม่ โดยรีเลชันที่สร้างใหม่นี้จะอยู่ในรูปแบบบรรทัดฐานระดับที่ 3 ที่มีคีย์หลักเป็นคีย์ผสม (Composite key) และมีแอททริบิวท์ที่ไม่ได้ใช้ในทุก ๆ ขั้นตอนที่ผ่านมีเป็นแอททริบิวท์ที่ไม่ใช่คีย์ ที่ขึ้นอยู่กับคีย์ผสมคู่นี้ ซึ่งหากย้อนกลับไปดูตารางที่ 3.1 จะเห็นว่ายังคงแอททริบิวท์ที่ยังไม่ถูกใช้หรือแอททริบิวท์ที่เหลืออยู่คือ แอททริบิวท์ QTY นั่นเอง และรีเลชันที่สร้างในขั้นตอนนี้จะถือรีเลชันสุดท้ายที่เราต้องการ โดยรีเลชันที่อยู่ในรูปแบบบรรทัดฐานระดับที่ 2 รีเลชันแรกที่โปรแกรมค้นหาได้คือ รีเลชัน Order ซึ่งมี

แอททริบิวท์ INV ทำหน้าที่เป็นคีย์หลักของรีเลชัน และรีเลชันที่อยู่ในรูปแบบบรรทัดฐานระดับที่ 2 รีเลชันถัดมาคือ รีเลชัน Product ซึ่งมีแอททริบิวท์ P_ID ที่ทำหน้าที่เป็นคีย์หลักของรีเลชัน ดังนั้น รีเลชันสุดท้ายที่ได้จะมีรายละเอียดของรีเลชัน ดังนี้

Detail (INV, P_ID, QTY)

รีเลชัน Detail พร้อมด้วยข้อมูลในรีเลชันหลังการปรับรูปแบบบรรทัดฐานในขั้นนี้แล้ว สามารถแสดงได้ดังตารางที่ 3.7

ตารางที่ 3.7 Detail

INV	**P_ID**	**QTY**
INV001	P001	10
INV001	P002	15
INV002	P004	20
INV003	P001	5
INV004	P003	11
INV004	P004	30

บทที่ 4
การทดสอบและอภิปรายผล

ในการทดสอบประสิทธิภาพของระบบ จะเป็นการทดสอบประสิทธิภาพในการปรับรูปแบบบรรทัดฐานด้วยโปรแกรม NoWARs ซึ่งการเปรียบเทียบประสิทธิภาพในการปรับรูปแบบบรรทัดฐานของโปรแกรม จะทำการเปรียบเทียบระหว่างรีเลชันที่ได้มีการปรับรูปแบบบรรทัดฐานแล้วโดยโปรแกรม NoWARs กับรีเลชันที่ยังไม่มีการปรับรูปแบบบรรทัดฐาน สำหรับการทดสอบระบบจะทำการทดสอบบนเครื่องคอมพิวเตอร์ Pentium IV ความเร็ว 3.2 GHz หน่วยความจำหลัก 512 MB ฮาร์ดดิสก์ความจุ 80 GB และทดสอบบนระบบจัดการฐานข้อมูล Oracle Database 10g Express Edition โดยการทดสอบประสิทธิภาพของการปรับรูปแบบบรรทัดฐาน จะทำการทดสอบกับข้อมูลทั้งหมด 5 ชุด ดังปรากฏรายละเอียดในหัวข้อที่ 4.1 ผลของการปรับรูปแบบบรรทัดฐานปรากฏในหัวข้อที่ 4.2 ผลการคัดเลือกกฎความสัมพันธ์ ปรากฏในหัวที่ 4.3 และหัวข้อที่ 4.4 เป็นการอภิปรายผล

4.1 ข้อมูลที่ใช้ในการทดสอบ

การทดสอบประสิทธิภาพการปรับรูปแบบบรรทัดฐานของโปรแกรม NoWARs ใช้ข้อมูลที่เป็นแอพพลิเคชันที่แตกต่างกันทั้งหมดจำนวน 5 ชุดข้อมูล ข้อมูลที่ใช้ทดสอบระบบชุดแรก เป็นรายละเอียดของข้อมูลการลงทะเบียนเรียน (Register) โดยมีจำนวนข้อมูลทั้งหมด 12 แถว ประกอบไปด้วยแอททริบิวท์ทั้งหมดจำนวน 7 แอททริบิวท์ ได้แก่ STUDENT_CODE, STUDENT_NAME, TEACHER_CODE, TEACHER_NAME, UNIT, SUBJECT_CODE และ SUBJECT_NAME โดยมีรายละเอียดของข้อมูลดังตารางที่ 4.1

ข้อมูลชุดที่สองเป็น ข้อมูลรายการเช่าวิดีโอ (Video_Rental) ของร้านเช่าวิดีโอแห่งหนึ่ง โดยมีจำนวนข้อมูลทั้งหมด 12 แถว ประกอบไปด้วยแอททริบิวท์จำนวน 10 แอททริบิวท์ ได้แก่ TRANS_ID, RENT_DATE, CUST_ID, CUST_NAME, PHONE, ADDRESS, VDO_ID, COPY#, TITLE และ RENT โดยมีรายละเอียดของข้อมูลดังตารางที่ 4.2

ข้อมูลทดสอบชุดที่สาม เป็นรายละเอียดของข้อมูลการพัฒนาซอฟต์แวร์ (Data_Org) ของบริษัทไอทีแห่งหนึ่ง โดยมีจำนวนข้อมูลทั้งหมด 21 แถว ประกอบไปด้วยแอททริบิวท์ทั้งหมดจำนวน 8 แอททริบิวท์ ได้แก่ PROJ_ID, PROJ_NAME, EMP_ID, EMP_NAME, JOB_ID, JOB_DESC, JOB_CHG_HOUR และ ASSIGN_HOUR โดยมีรายละเอียดของข้อมูลดังตารางที่ 4.3

ข้อมูลทดสอบชุดที่สี่ เป็นข้อมูลการซื้อ-ขาย (Invoice) สินค้าของร้านค้าแห่งหนึ่ง โดยมีข้อมูลทั้งหมด 8 แถว ประกอบไปด้วยแอททริบิวท์ทั้งหมด 9 แอททริบิวท์ ได้แก่ ORDER_NO, ORDER_DATE, CUST_ID, CUST_NAME, ADDRESS, PRODUCT_ID, PRODUCT_DESC, UNIT_PRICE และ QTY โดยมีรายละเอียดของข้อมูลดังแสดงในตารางที่ 4.4

ข้อมูลทดสอบชุดที่ห้า เป็นรายละเอียดของข้อมูลการทำสีรถยนต์ (Car_Color) โดยมีจำนวนข้อมูลทั้งหมด 16 แถว ประกอบไปด้วยแอททริบิวท์ทั้งหมดจำนวน 9 แอททริบิวท์ ได้แก่ PRD_SIZE_ID, PRD_ID, PRD_DESC, CUST_ID, CUST_NAME, UPRICE, QTY, ORDER_NO และ ORDER_DATE โดยมีรายละเอียดของข้อมูลดังตารางที่ 4.5

ตารางที่ 4.1 ข้อมูลการลงทะเบียนเรียน

STUDENT_CODE	STUDENT_NAME	TEACHER_CODE	TEACHER_NAME	SUBJECT_CODE	SUBJECT_NAME	UNIT
41010730	Somchai Bholajan	Q1059	Samphan Yensamrarn	729101	Economy	2
41010730	Somchai Bholajan	Q1059	Samphan Yensamrarn	729111	Statistic	3
41010730	Somchai Bholajan	Q1059	Samphan Yensamrarn	999211	Computer	3
41010943	Suthisa Pinijpaidhool	Q1011	Siripattra Muanmalai	729111	Statistic	3
41010943	Suthisa Pinijpaidhool	Q1011	Siripattra Muanmalai	999211	Computer	3
41010943	Suthisa Pinijpaidhool	Q1011	Siripattra Muanmalai	729104	Financial Management	2
41012147	Natthaporn Prakeb	Q1061	Metee Piyakhun	729111	Statistic	3
41012147	Natthaporn Prakeb	Q1061	Metee Piyakhun	999211	Computer	3
41012451	Nopadol Thabthong	Q1035	Sirichai Sriphrom	729111	Statistic	3
41012451	Nopadol Thabthong	Q1035	Sirichai Sriphrom	999211	Computer	3
41013327	Matana Pinipaidhool	Q1059	Samphan Yensamrarn	729103	Marketing	2
41013780	Somchai Bholajan	Q1011	Siripattra Muanmalai	999211	Computer	3

ตารางที่ 4.2 ข้อมูลเช่าวิดีโอ

TRANS_ID	RENT_DATE	CUST_ID	CUST_NAME	PHONE	ADDRESS	VDO_ID	COPY#	TITLE	RENT
1	4/18/1995	3	Washington	502-777-757	95 Easy Street	1	2	A Space Odyssey	1.50
1	4/18/1995	3	Washington	502-777-757	95 Easy Street	6	3	My Dog	1.50
2	4/18/1995	7	Lasater	615-888-447	67 S. Ray Drive	8	1	Hopscotch	1.50
2	4/18/1995	7	Lasater	615-888-447	67 S. Ray Drive	2	2	Apocalypse	2.00
2	4/18/1995	7	Lasater	615-888-447	67 S. Ray Drive	6	1	My Dog	1.50
3	4/29/1995	8	Jones	615-452-116	867 Lakeside	9	1	The Gods	2.50
3	4/29/1995	8	Jones	615-452-116	867 Lakeside	15	2	Baker Boys	2.00
3	4/29/1995	8	Jones	615-452-116	867 Lakeside	4	3	Boy And His Dog	1.50
4	4/30/1995	3	Washington	502-777-757	95 Easy Street	3	1	Blues Brothers	2.00
4	4/30/1995	3	Washington	502-777-757	95 Easy Street	8	4	Hopscotch	1.50
4	4/30/1995	3	Washington	502-777-757	95 Easy Street	9	5	The Gods	2.50
4	4/30/1995	3	Washington	502-777-757	95 Easy Street	17	3	The Witches	2.00

ตารางที่ 4.3 ข้อมูลการพัฒนาซอฟต์แวร์

PROJ_ID	PROJ_NAME	EMP_ID	EMP_NAME	JOB_ID	JOB_DESC	JOB_CHG_HOUR	ASSIGN_HOUR
15	Evergreen	103	June E. Arbough	503	Elect. Engineer	84.50	23.8
15	Evergreen	101	John G. News	502	Database Designer	105.00	19.4
15	Evergreen	105	Alice K. Johnson	502	Database Designer	105.00	35.7
15	Evergreen	106	William Smithfield	500	Programmer	35.75	12.6
15	Evergreen	102	David H. Senoir	501	System Analyst	96.75	23.8
18	Amber Wave	114	Annelise Jones	508	Applications Designer	48.10	24.6
18	Amber Wave	118	James J. Frommer	510	General Support	18.36	45.3
18	Amber Wave	104	Anne K. Ramoras	501	System Analyst	96.75	32.4
18	Amber Wave	112	Darlene M. Smithson	507	DSS Analyst	45.95	44.0
22	Rolling Tide	105	Alice K. Johnson	502	Database Designer	105.00	64.7
22	Rolling Tide	104	Anne K. Ramoras	501	System Analyst	96.75	48.4
22	Rolling Tide	113	Delbert K. Joenbrood	508	Applications Designer	48.10	23.6
22	Rolling Tide	111	Geoff B. Wabash	506	Clerical Support	26.87	22.0
22	Rolling Tide	106	William Smithfield	500	Programmer	35.75	12.8
25	Starflight	107	Mirai D. Alonso	500	Programmer	35.75	24.6
25	Starflight	115	Travis B. Bawangi	501	System Analyst	96.75	45.8

ตารางที่ 4.3 ข้อมูลการพัฒนาซอฟต์แวร์ (ต่อ)

PROJ_ID	PROJ_NAME	EMP_ID	EMP_NAME	JOB_ID	JOB_DESC	JOB_CHG_HOUR	ASSIGN_HOUR
25	Starflight	101	John G. News	502	Database Designer	105.00	56.3
25	Starflight	114	Annelise Jones	508	Applications Designer	48.10	33.1
25	Starflight	108	Ralph B. Washington	501	System Analyst	96.75	23.6
25	Starflight	118	James J. Frommer	510	General Support	18.36	30.5
25	Starflight	112	Darlene M. Smithson	507	DSS Analyst	45.95	41.4

ตารางที่ 4.4 ข้อมูลการซื้อ-ขาย สินค้า

ORDER_NO	ORDER_DATE	CUST_ID	CUST_NAME	ADDRESS	PRODUCT_ID	PRODUCT_DESC	UNIT_PRICE	QTY
1006	10/24/2006	2	Value Furniture	Plano-TX	7	Dining Table	800.00	2
1006	10/24/2006	2	Value Furniture	Plano-TX	5	Writer Desk	325.00	3
1006	10/24/2006	2	Value Furniture	Plano-TX	4	Entertainment Center	650.00	1
1005	10/24/2006	4	Ball Furniture	Dallas-TX	3	Computer Desk	750.00	4
1008	10/25/2006	5	Best Furniture	Fris-TX	3	Computer Desk	750.00	5
1007	10/25/2006	6	Furniture DD	Boulder-CO	11	4-Dr Dresser	500.00	4
1007	10/25/2006	6	Furniture DD	Boulder-CO	4	Entertainment Center	650.00	3
1009	10/25/2006	2	Value Furniture	Plano-TX	11	4-Dr Dresser	500.00	5

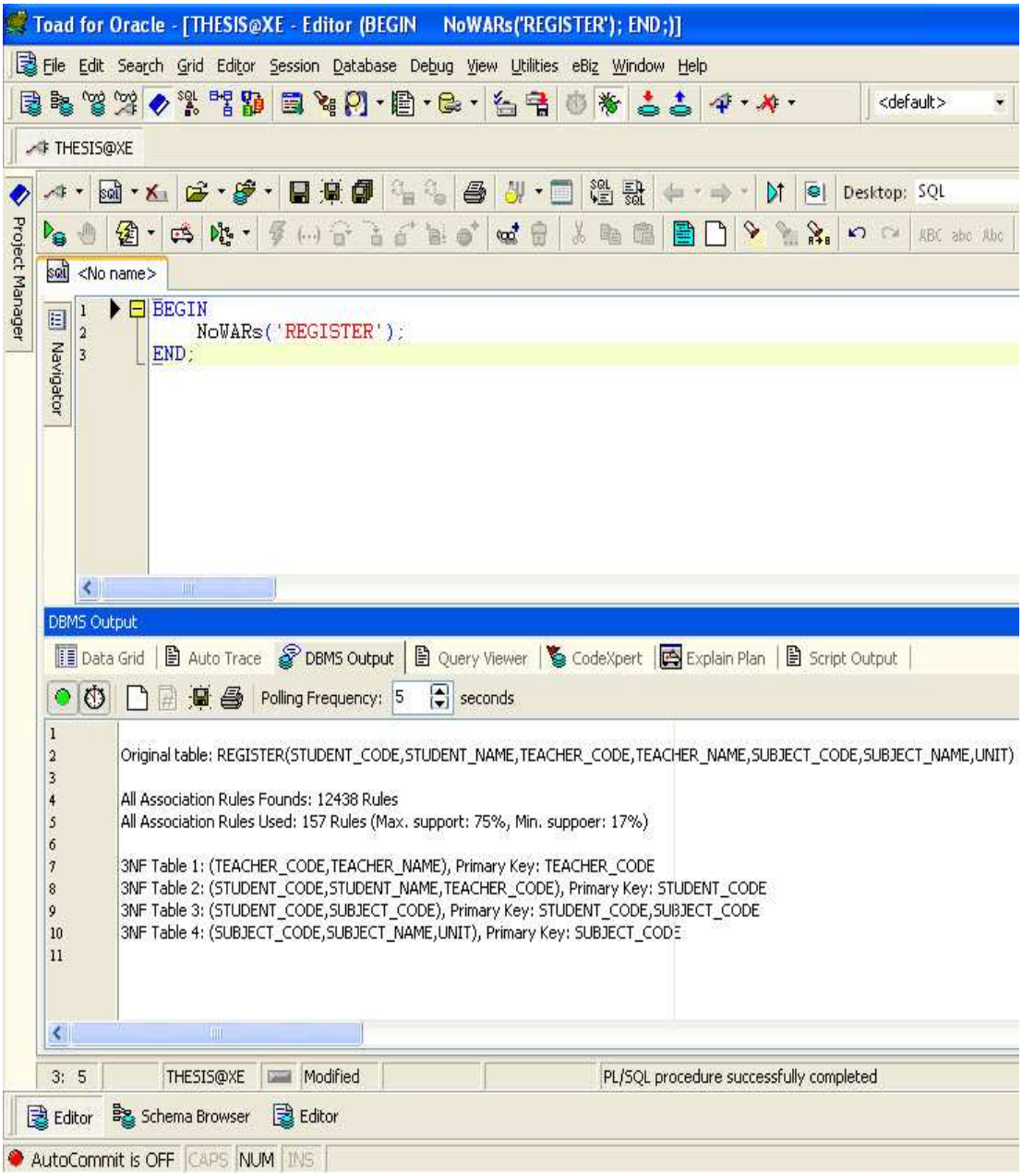
ตารางที่ 4.5 ข้อมูลการทำสีรถยนต์

PRD_SIZE_ID	PRD_ID	PRD_DESC	CUST_ID	CUST_NAME	UPRICE	QTY	ORDER_NO	ORDER_DATE
3mm	8	Blue Acrylic-3	8	Nimitr	2300	4	174	12/26/2008
5mm	7	Red Acrylic-5	3	Niyom	3800	2	171	12/24/2008
1mm	4	Green Acrylic-1	3	Niyom	800	1	171	12/24/2008
4mm	11	Blue Acrylic-4	6	Bancha	3000	1	172	12/26/2008
3mm	5	Green Acrylic-3	8	Nimitr	2300	2	174	12/26/2008
2mm	9	Green Acrylic-2	8	Nimitr	1500	5	174	12/26/2008
3mm	5	Green Acrylic-3	4	Sincharoen	2300	2	175	12/26/2008
4mm	11	Blue Acrylic-4	4	Sincharoen	3000	1	175	12/26/2008
2mm	6	Blue Acrylic-2	4	Sincharoen	1500	2	175	12/26/2008
1mm	4	Green Acrylic-1	4	Sincharoen	800	4	175	12/26/2008
5mm	7	Red Acrylic-5	2	Niyom	3800	7	176	12/27/2008
4mm	11	Blue Acrylic-4	2	Niyom	3000	2	176	12/27/2008
1mm	4	Green Acrylic-1	2	Niyom	800	4	176	12/27/2008
2mm	9	Green Acrylic-2	8	Nimitr	1500	10	177	12/27/2008
3mm	5	Green Acrylic-3	8	Nimitr	2300	2	177	12/27/2008
1mm	4	Green Acrylic-1	6	Bancha	800	3	172	12/26/2008

4.2 ผลของการปรับรูปแบบบรรทัดฐาน

4.2.1 ผลการทดสอบกับข้อมูลการลงทะเบียนเรียน

เมื่อนำข้อมูลการลงทะเบียนเรียน ไปทดสอบกับโปรแกรม NoWARs โปรแกรมจะให้ผลการทดสอบดังรูปที่ 4.1



รูปที่ 4.1 ผลลัพธ์การปรับรูปแบบบรรทัดฐานของข้อมูลการลงทะเบียนเรียน

จากรูปที่ 4.1 จะเห็นว่าโปรแกรมทำการแยกรีเลชันได้ทั้งหมด 4 รีเลชัน ซึ่งประกอบไปด้วย

Table1 (TEACHER_CODE, TEACHER_NAME),

Primary Key: TEACHER_CODE

Table2 (STUDENT_CODE, STUDENT_NAME, TEACHER_CODE),

Primary Key: STUDENT_CODE

Table3 (STUDENT_CODE, SUBJECT_CODE),

Primary Key: STUDENT_CODE, SUBJECT_CODE

Table4 (SUBJECT_CODE, SUBJECT_NAME, UNIT),

Primary Key: SUBJECT_CODE

หากนำรีเลชันทั้ง 4 ไปสร้างเป็นตารางจัดเก็บข้อมูล จะได้รายละเอียดการจัดเก็บข้อมูลดังตารางที่ 4.6-4.9 ตามลำดับ

ตารางที่ 4.6 Table1_Register

TEACHER_CODE	**TEACHER_NAME**
Q1059	Samphan Yensamrarn
Q1011	Siripattra Muanmalai
Q1061	Metee Piyakhun
Q1035	Sirichai Sriphrom

ตารางที่ 4.7 Table2_Register

STUDENT_CODE	**STUDENT_NAME**	**TEACHER_CODE**
41010730	Somchai Bholajan	Q1059
41010943	Suthisa Pinijpaidhool	Q1011
41012147	Natthaporn Prakeb	Q1061
41012451	Nopadol Thabthong	Q1035
41013327	Matana Pinipaidhool	Q1059
41013780	Somchai Bholajan	Q1011

ตารางที่ 4.8 Table3_Register

STUDENT_CODE	**SUBJECT_CODE**
41010730	729101
41010730	729111
41010730	999211
41010943	729111
41010943	999211
41010943	729104
41012147	729111
41012147	999211
41012451	729111
41012451	999211
41013327	729103
41013780	999211

ตารางที่ 4.9 Table4_Register

SUBJECT_CODE	**SUBJECT_NAME**	**UNIT**
729101	Economy	2
999211	Computer	3
729104	Financial Management	2
729111	Statistic	3

เมื่อพิจารณาตารางที่ 4.6-4.9 จะเห็นว่าทุกตารางไม่มีปัญหาเรื่อง ความซ้ำซ้อนของข้อมูล และหากต้องการที่จะทำการลบข้อมูล เพิ่มข้อมูล หรือแม้กระทั่งทำการแก้ไขข้อมูล ก็สามารถทำได้ โดยที่ไม่มีความผิดปกติใด ๆ เกิดขึ้นในฐานข้อมูล แต่หากพิจารณาตารางที่ 4.1 ข้อมูลการลงทะเบียนเรียน ซึ่งเป็นตารางที่ยังไม่ผ่านกระบวนการปรับรูปแบบบรรทัดฐานด้วยโปรแกรม NoWARs จะเห็นว่ายังคงมีปัญหาที่อาจเกิดขึ้นได้ดังต่อไปนี้

1) ปัญหาในการลบข้อมูล

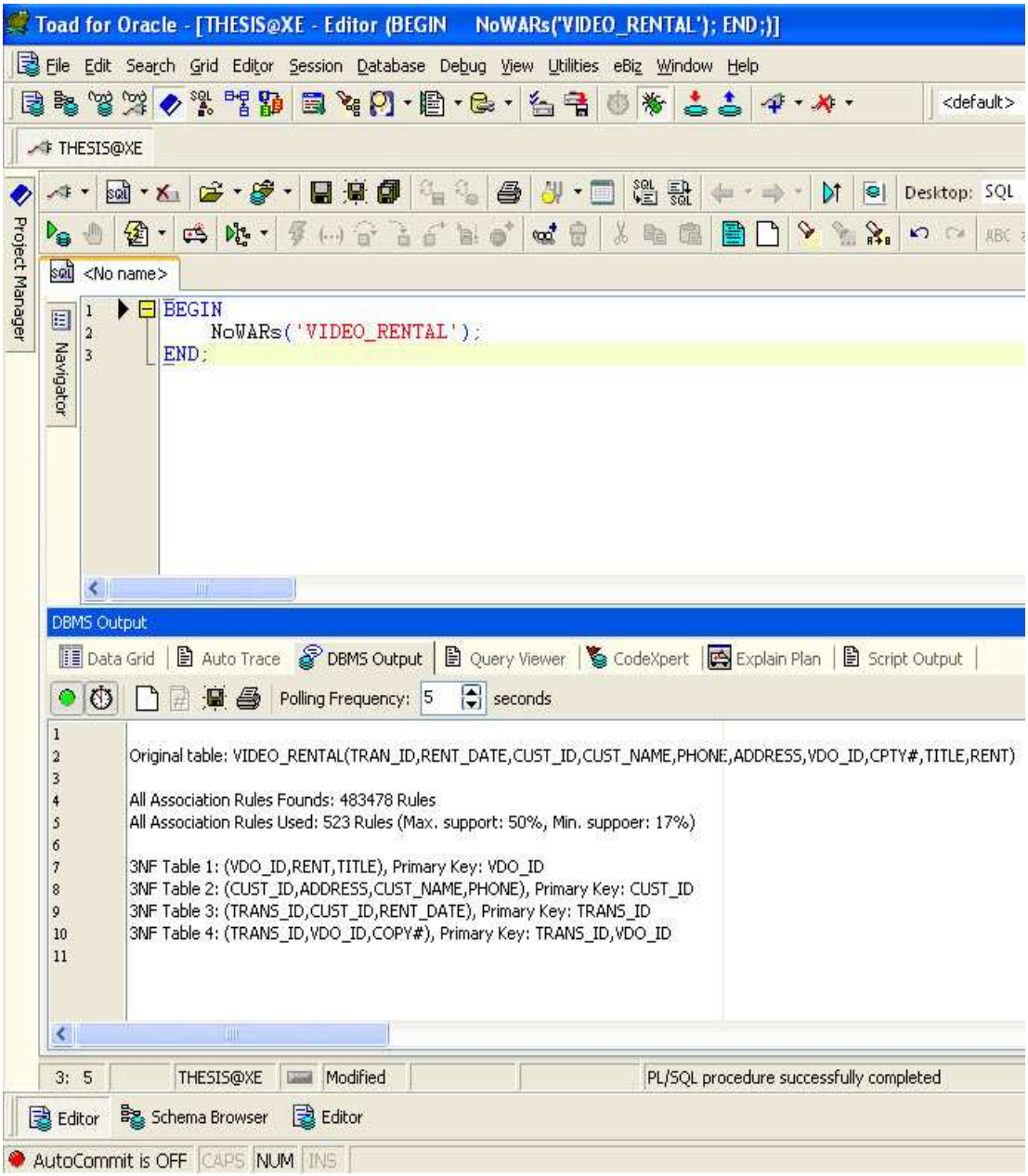
หากนักศึกษาที่ชื่อ “Natthaporn Prakongkeb” ได้ลาออกมหาวิทยาลัย ก็ต้องมีการลบข้อมูลของนักศึกษาคนนี้ทิ้ง ซึ่งนอกจากต้องทำการลบข้อมูลในหลายแถวแล้ว การลบข้อมูลของนักศึกษาคนนี้ทิ้ง จะมีผลทำให้สูญเสียข้อมูลของอาจารย์ที่มีรหัสอาจารย์ (TEACHER_CODE) เป็น “Q1061” คือ อาจารย์ “Metee Piyakhun” ไป เนื่องจากไม่มีการอ้างอิงถึงอาจารย์ท่านนี้ในข้อมูลแถวอื่น ๆ

2) ปัญหาในการปรับปรุงแก้ไขข้อมูล

หากต้องการลดจำนวนหน่วยกิต (UNIT) วิชา “Fundamental of Computer” จากเดิม “3” ให้เหลือ “2” ก็ต้องทำการแก้ไขข้อมูลในหลายแถวที่มีรหัสวิชา (SUBJECT_CODE) เป็น “999211” ซึ่งถ้าแก้ไขข้อมูลไม่ครบแล้ว จะทำให้ข้อมูลเกิดความขัดแย้งกันได้ เนื่องจากวิชาเดียวกันแต่มีจำนวนหน่วยกิตต่างกัน คือ “3” และ “2” หน่วยกิต

4.2.2 ผลการทดสอบกับข้อมูลการเช่าวิดีโอ

เมื่อนำข้อมูลการเช่าวิดีโอ ไปทดสอบกับโปรแกรม NoWARs โปรแกรมจะให้ผลการทดสอบดังรูปที่ 4.2



รูปที่ 4.2 ผลลัพธ์การปรับรูปแบบบรรทัดฐานของข้อมูลการเช่าวิดีโอ

จากรูปที่ 4.2 จะเห็นว่าโปรแกรมทำการแยกรีเลชันได้ทั้งหมด 4 รีเลชัน ซึ่งประกอบไปด้วย

Table1 (VDO_ID, RENT, TITLE), Primary Key: VDO_ID

Table2 (CUST_ID, ADDRESS, CUST_NAME, PHONE),

Primary Key: CUST_ID

Table3 (TRANS_ID, CUST_ID, RENT_DATE), Primary Key: TRANS_ID

Table4 (TRANS_ID, VDO_ID, COPY#),

Primary Key: TRANS_ID, VDO_ID

หากนำรีเลชันทั้ง 4 ไปสร้างเป็นตารางจัดเก็บข้อมูล จะได้รายละเอียดการจัดเก็บข้อมูลดังตารางที่ 4.10-4.13 ตามลำดับ

ตารางที่ 4.10 Table1_Video_Rental

VDO_ID	**RENT**	**TITLE**
1	1.50	A Space Odyssey
2	2.00	Apocalypse
3	2.00	Blues Brothers
4	1.50	Boy And His Dog
6	1.50	My Dog
8	1.50	Hopscotch
9	2.50	The Gods
15	2.00	Baker Boys
17	2.00	The Witches

ตารางที่ 4.11 Table2_Video_Rental

CUST_ID	**ADDRESS**	**CUST_NAME**	**PHONE**
3	95 Easy Street	Washington	502-777-7575
7	67 S. Ray Drive	Lasater	615-888-4474
8	867 Lakeside Drive	Jones	615-452-1162

ตารางที่ 4.12 Table3_Video_Rental

TRANS_ID	**CUST_ID**	**RENT_DATE**
1	3	4/18/1995
2	7	4/18/1995
3	8	4/29/1995
4	3	4/30/1995

ตารางที่ 4.13 Table4_Video_Rental

TRANS_ID	**VDO_ID**	**COPY#**
1	1	2
1	6	3
2	8	1
2	2	2
2	6	1
3	9	1
3	15	2
3	4	3
4	3	1
4	8	4
4	9	5
4	17	3

เมื่อพิจารณาผลที่ได้จากโปรแกรมดังตารางที่ 4.10-4.13 จะเห็นว่าทุกตารางเป็นตารางที่อยู่ในรูปแบบบรรทัดฐานระดับที่ 3 ซึ่งไม่มีปัญหาเรื่อง ความซ้ำซ้อนของข้อมูล และหากต้องการที่จะทำการลบข้อมูล เพิ่มข้อมูล หรือแม้กระทั่งทำการแก้ไขข้อมูล ก็สามารถทำได้โดยที่ไม่มีความผิดปกติใด ๆ เกิดขึ้นในฐานข้อมูล แต่หากพิจารณาตารางที่ 4.2 ข้อมูลการเช่าวิดีโอ ซึ่งเป็นตารางที่ยังไม่ผ่านกระบวนการปรับรูปแบบบรรทัดฐานด้วยโปรแกรม NoWARs จะเห็นว่ายังมีข้อมูลในบางแอททริบิวท์ที่มีการเก็บค่าของข้อมูลซ้ำ ๆ กัน เช่น แอททริบิวท์ TRANS_ID, RENT_DATE, CUST_ID, CUST_NAME, PHONE และ ADDRESS ซึ่งการเก็บข้อมูลซ้ำ ๆ กันเป็นจำนวนมากลักษณะนี้ จะทำให้ระบบฐานข้อมูลเสียพื้นที่จำนวนมากในการเก็บข้อมูลที่ซ้ำ ๆ กันโดยไม่จำเป็น และตารางที่ 4.2 ยังคงเป็นตารางที่อาจเกิดปัญหาขึ้นได้ ดังต่อไปนี้

1) ปัญหาในการลบข้อมูล

หากต้องการยกเลิกรายการเช่าวิดีโอที่มีรหัส TRANS_ID เป็น “1” ซึ่งเป็นรายการเช่าวิดีโอของลูกค้าชื่อ “Washington” ออกจากรีเลชัน ก็จะมีผลทำให้ต้องลบข้อมูลในแถวที่ และ 2 ทิ้ง ซึ่งการลบข้อมูลในแถวที่ 1 ทิ้ง จะมีผลทำให้ข้อมูลของวิดีโอ ที่มีรหัสวิดีโอ (VDO_ID) เป็น “1” ชื่อเรื่อง (TITLE) “A Space Odyssey” ถูกลบออกจากรีเลชันด้วย เนื่องจากไม่มีการอ้างถึงข้อมูลของวิดีโอนี้ในข้อมูลแถวอื่น ๆ หรือในกรณีคล้ายกัน ถ้าหากต้องการยกเลิกการเป็นสมาชิก

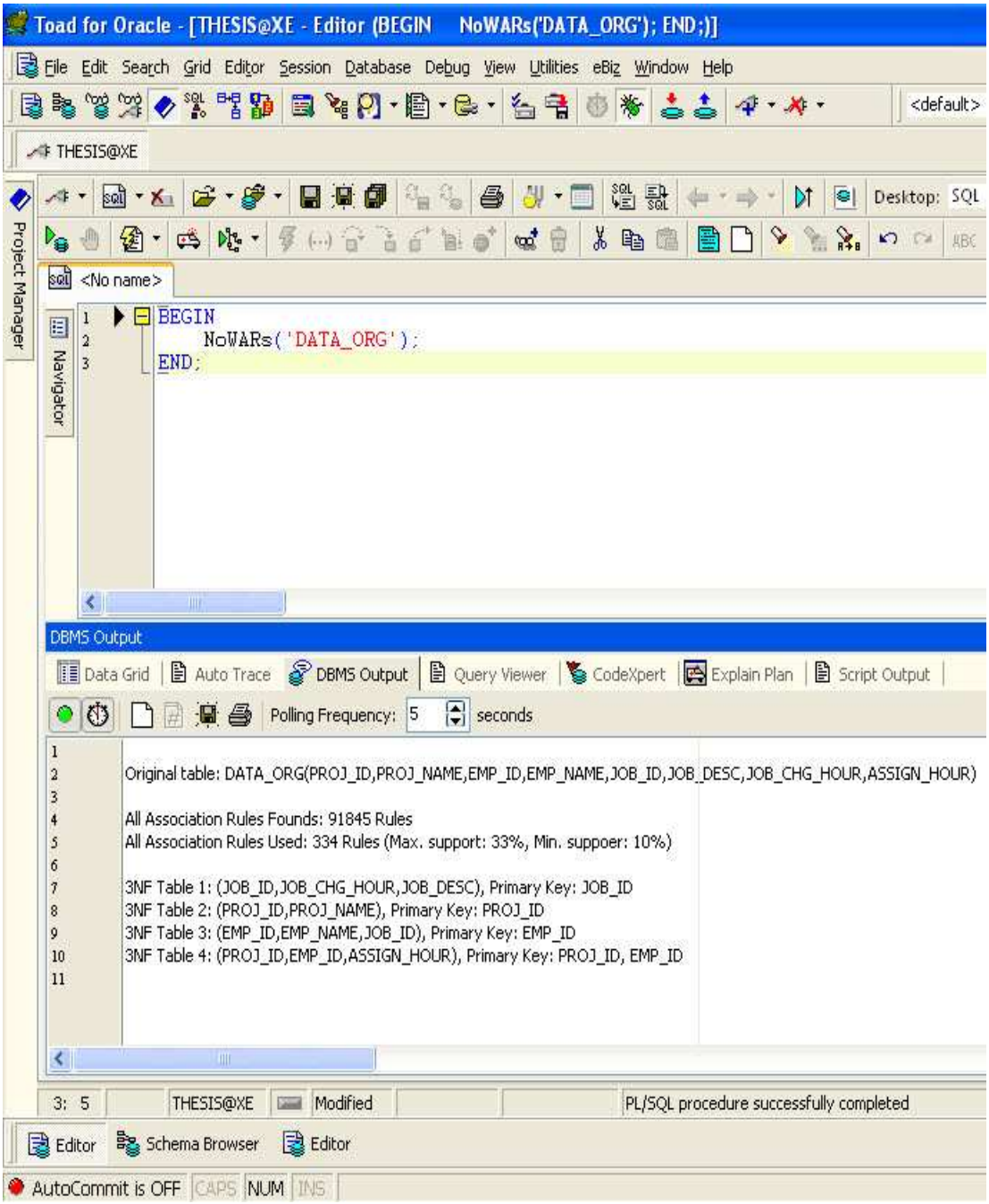
ของลูกค้าชื่อ (CUST_NAME) “Jones” ก็จะมีผลทำให้ข้อมูลในแถวที่ 6-8 ถูกลบทิ้งไปด้วย ซึ่งการลบข้อมูลในแถวที่ 7 และ 8 ทิ้ง จะมีผลทำให้ข้อมูลของวิดีโอ ที่มีรหัสวิดีโอ (VDO_ID) เป็น “9” ชื่อเรื่อง (TITLE) “The Gods” และ ข้อมูลของวิดีโอ ที่มีรหัสวิดีโอ (VDO_ID) เป็น “15” ชื่อเรื่อง (TITLE) “Baker Boys” ถูกลบออกจากรีเลชันด้วย เนื่องจากไม่มีการอ้างถึงข้อมูลของวิดีโอทั้งสองเรื่องในข้อมูลแถวอื่น ๆ

2) ปัญหาในการปรับปรุงแก้ไขข้อมูล

หากต้องมีการทำการเปลี่ยนแปลงราคาการเช่าวิดีโอของวิดีโอที่มีรหัสวิดีโอ (VDO_ID) เป็น “9” จากราคาเช่า (RENT) เดิม “2.50” เป็น “2.00” ก็ต้องทำการเปลี่ยนแปลงข้อมูลในหลายแถว ซึ่งนอกจากต้องเสียเวลาในการแก้ไขข้อมูลแล้ว ยังไม่สามารถรับประกันได้ว่า ได้มีการแก้ไขข้อมูลครบหมดแล้ว ซึ่งหากแก้ไขข้อมูลได้ไม่ครบก็จะทำให้ข้อมูลเกิดความขัดแย้งกันได้ เนื่องจากวิดีโอเรื่องเดียวกันแต่มีราคาเช่าสองราคา คือ “2.50” และ “2.00” หรือในกรณีที่ลูกค้ามีการเปลี่ยนชื่อ และต้องการปรับปรุงแก้ไข เช่น ลูกค้าที่มีรหัสลูกค้า (CUST_ID) เป็น “8” เปลี่ยนชื่อจาก “Jones” เป็น “John” ก็ต้องทำการปรับปรุงข้อมูลในอีกหลายแถว และถ้าแก้ไขข้อมูลไม่ครบก็จะทำให้ข้อมูลเกิดความขัดแย้งกันได้ เนื่องจากลูกค้ารหัสเดียวกันแต่กลับมีสองชื่อ เป็นต้น

4.2.3 ผลการทดสอบกับข้อมูลการพัฒนาซอฟต์แวร์

เมื่อนำข้อมูลการพัฒนาซอฟต์แวร์ ไปทดสอบกับโปรแกรม NoWARs โปรแกรมจะให้ผลการทดสอบดังรูปที่ 4.3



รูปที่ 4.3 ผลลัพธ์การปรับรูปแบบบรรทัดฐานของข้อมูลการพัฒนาซอฟต์แวร์

จากรูปที่ 4.3 จะเห็นว่าโปรแกรมทำการแยกรีเลชันออกได้ทั้งหมด 4 รีเลชัน ซึ่งประกอบไปด้วย

Table1 (JOB_ID, JOB_CHG_HOUR, JOB_DESC), Primary Key: JOB_ID

Table2 (PROJ_ID, PROJ_NAME), Primary Key: PROJ_ID

Table3 (EMP_ID, EMP_NAME, JOB_ID), Primary Key: EMP_ID

Table4 (PROJ_ID, EMP_ID, ASSIGN_HOUR),

Primary Key: PROJ_ID, EMP_ID

หากนำรีเลชันทั้ง 4 ไปสร้างเป็นตารางจัดเก็บข้อมูล จะได้รายละเอียดการจัดเก็บข้อมูลดังตารางที่ 4.14-4.17 ตามลำดับ

ตารางที่ 4.14 Table1_Data_Org

JOB_ID	**JOB_CHG_HOUR**	**JOB_DESC**
500	35.75	Programmer
501	96.75	System Analyst
502	105.00	Database Designer
503	84.50	Elect. Engineer
506	26.87	Clerical Support
507	45.95	DSS Analyst
508	48.10	Applications Designer
510	18.36	General Support

ตารางที่ 4.15 Table2_Data_Org

PROJ_ID	**PROJ_NAME**
15	Evergreen
18	Amber Wave
22	Rolling Tide
25	Starflight

ตารางที่ 4.16 Table3_Data_Org

EMP_ID	**EMP_NAME**	**JOB_ID**
101	John G. News	502
102	David H. Senoir	501
103	June E. Arbough	503
104	Anne K. Ramoras	501
105	Alice K. Johnson	502
106	William Smithfield	500
107	Mirai D. Alonso	500
108	Ralph B. Washington	501
111	Geoff B. Wabash	506
112	Darlene M. Smithson	507
113	Delbert K. Joenbrood	508
114	Annelise Jones	508
115	Travis B. Bawangi	501
118	James J. Frommer	510

ตารางที่ 4.17 Table4_Data_Org

PROJ_ID	**EMP_ID**	**ASSIGN_HOUR**
15	103	23.8
15	101	19.4
15	105	35.7
15	106	12.6
15	102	23.8
18	114	24.6
18	118	45.3
18	104	32.4
18	112	44.0
22	105	64.7
22	104	48.4

ตารางที่ 4.17 Table4_Data_Org (ต่อ)

PROJ_ID	EMP_ID	ASSIGN_HOUR
22	113	23.6
22	111	22.0
22	106	12.8
25	107	24.6
25	115	45.8
25	101	56.3
25	114	33.1
25	108	23.6
25	118	30.5
25	112	41.4

เมื่อพิจารณาตารางที่ 4.14-4.17 จะเห็นว่าทุกตารางไม่มีปัญหาเรื่อง ความซ้ำซ้อนของข้อมูล และหากต้องการที่จะทำการลบข้อมูล เพิ่มข้อมูล หรือแม้กระทั่งทำการแก้ไขข้อมูล ก็สามารถทำได้ โดยที่ไม่มีความผิดปกติใด ๆ เกิดขึ้นในฐานข้อมูล แต่หากพิจารณาตารางที่ 4.3 ข้อมูลการพัฒนาซอฟต์แวร์ ซึ่งเป็นตารางที่ยังไม่ผ่านกระบวนการปรับรูปแบบบรรทัดฐานด้วยโปรแกรม NoWARs จะเห็นว่ายังคงมีปัญหาที่อาจเกิดขึ้นได้ดังต่อไปนี้

1) ปัญหาในการลบข้อมูล

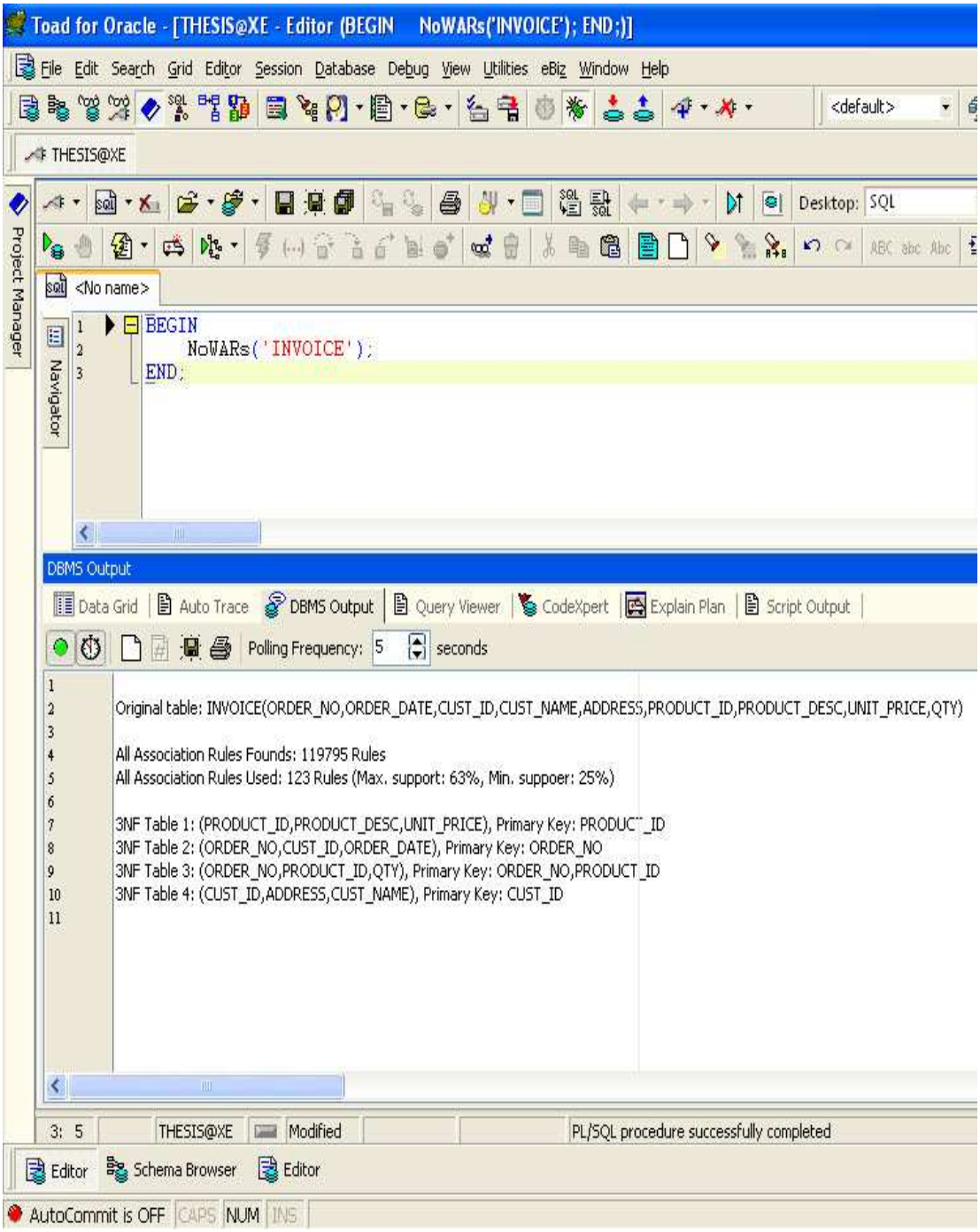
หากพนักงานที่ชื่อ “Darlene M. Smithson” ได้ลาออกจากบริษัท ก็ต้องมีการลบข้อมูลของพนักงานคนนี้ทิ้ง ซึ่งนอกจากต้องทำการลบข้อมูลในแถวแล้ว การลบข้อมูลของพนักงานคนนี้ทิ้ง มีผลทำให้ต้องสูญเสียข้อมูลของตำแหน่งหน้าที่งานในบริษัทที่มีรหัสงาน (JOB_ID) เป็น “507” คือ งานในตำแหน่ง “DSS Analyst” ไป เนื่องจากไม่มีการอ้างอิงถึงตำแหน่งงานนี้ในข้อมูลแถวอื่น ๆ

2) ปัญหาในการปรับปรุงแก้ไขข้อมูล

หากบริษัทต้องการปรับอัตราค่าแรงต่อชั่วโมง (JOB_CHG_HOUR) ของ “Database Designer” จากเดิม “105.00” เป็น “110.00” ก็ต้องทำการแก้ไขข้อมูลในหลายแถวที่มีรหัสงาน (JOB_ID) เป็น “502” ซึ่งถ้าแก้ไขข้อมูลไม่ครบแล้ว จะทำให้ข้อมูลเกิดความขัดแย้งกันได้ เนื่องจากงานตำแหน่งเดียวกันแต่มีอัตราค่าแรงต่อชั่วโมงสองค่า คือ “105.00” และ “110.00”

4.2.4 ผลการทดสอบกับข้อมูลการซื้อ-ขาย สินค้า

เมื่อนำข้อมูลการซื้อ-ขาย สินค้า ไปทดสอบกับโปรแกรม NoWARs โปรแกรมจะให้ผลการทดสอบดังรูปที่ 4.4



รูปที่ 4.4 ผลลัพธ์การปรับรูปแบบบรรทัดฐานของข้อมูลการซื้อ-ขายสินค้า

จากรูปที่ 4.4 จะเห็นว่าโปรแกรมทำการแยกรีเลชันได้ทั้งหมด 4 รีเลชัน ซึ่งประกอบไปด้วย

Table1 (PRODUCT_ID, PRODUCT_DESC, UNIT_PRICE),

Primary Key: PRODUCT_ID

Table2 (ORDER_NO, CUST_ID, ORDER_DATE), Primary Key: ORDER_NO

Table3 (ORDER_NO, PRODUCT_ID, QTY),

Primary Key: ORDER_NO, PRODUCT_ID

Table4 (CUST_ID, ADDRESS, CUST_NAME), Primary Key: CUST_ID

หากนำรีเลชันทั้ง 4 ไปสร้างเป็นตารางจัดเก็บข้อมูล จะได้รายละเอียดการจัดเก็บข้อมูลดังตารางที่ 4.18-4.21 ตามลำดับ

ตารางที่ 4.18 Table1_Invoice

PRODUCT_ID	**PRODUCT_DESC**	**UNIT_PRICE**
3	Computer Desk	750.00
4	Entertainment Center	650.00
5	Writer Desk	325.00
7	Dining Table	800.00
11	4-Dr Dresser	500.00

ตารางที่ 4.19 Table2_Invoice

ORDER_NO	**CUST_ID**	**ORDER_DATE**
1005	4	10/24/2006
1006	2	10/24/2006
1007	6	10/25/2006
1008	5	10/25/2006
1009	2	10/25/2006

ตารางที่ 4.20 Table3_Invoice

ORDER_NO	**PRODUCT_ID**	**QTY**
1006	7	2
1006	5	3
1006	4	1
1005	3	4
1008	3	5
1007	11	4
1007	4	3
1009	11	5

ตารางที่ 4.21 Table4_Invoice

CUST_ID	**ADDRESS**	**CUST_NAME**
2	Plano-TX	Value Furniture
4	Dallas-TX	Beauty Furniture
5	FRIS-TX	Best Furniture
6	Boulder-CO	Furniture Gallery

เมื่อพิจารณาตารางที่ 4.18-4.21 จะเห็นว่าทุกตารางไม่มีปัญหาเรื่อง ความซ้ำซ้อนของข้อมูล และหากต้องการที่จะทำการลบข้อมูล เพิ่มข้อมูล หรือแม้กระทั่งทำการแก้ไขข้อมูล ก็สามารถทำได้ โดยที่ไม่มีความผิดปกติใด ๆ เกิดขึ้นกับฐานข้อมูล แต่หากพิจารณาตารางที่ 4.4 ข้อมูลการซื้อ-ขายสินค้า ซึ่งเป็นตารางที่ยังไม่ผ่านกระบวนการปรับรูปแบบบรรทัดฐานด้วยโปรแกรม NoWARs จะเห็นว่าตารางดังกล่าวมีการจัดเก็บข้อมูลที่ยังคงมีข้อมูลที่เก็บค่าซ้ำ ๆ กัน เช่น แอททริบิวท์ ORDER_NO, ORDER_DATE, CUST_ID, CUST_NAME และ ADDRESS ซึ่งการเก็บข้อมูลที่มีค่าซ้ำ ๆ กัน ลักษณะนี้ จะทำให้เปลืองเนื้อที่ในการจัดเก็บข้อมูลเป็นอย่างมาก และการเก็บข้อมูลดังตารางที่ 4.1 ยังคงมีปัญหาที่อาจเกิดขึ้นได้ดังต่อไปนี้

1) ปัญหาในการลบข้อมูล

หากมีการยกเลิกรายการซื้อหมายเลข (ORDER_NO) “1006” ออกจากรีเลชัน จะมีผลทำให้ข้อมูลแถวที่ 1-3 ถูกลบทิ้งไปด้วย แต่ก็จะพบว่าการลบข้อมูลในแถวที่ 2 ทิ้งไป จะมีผล

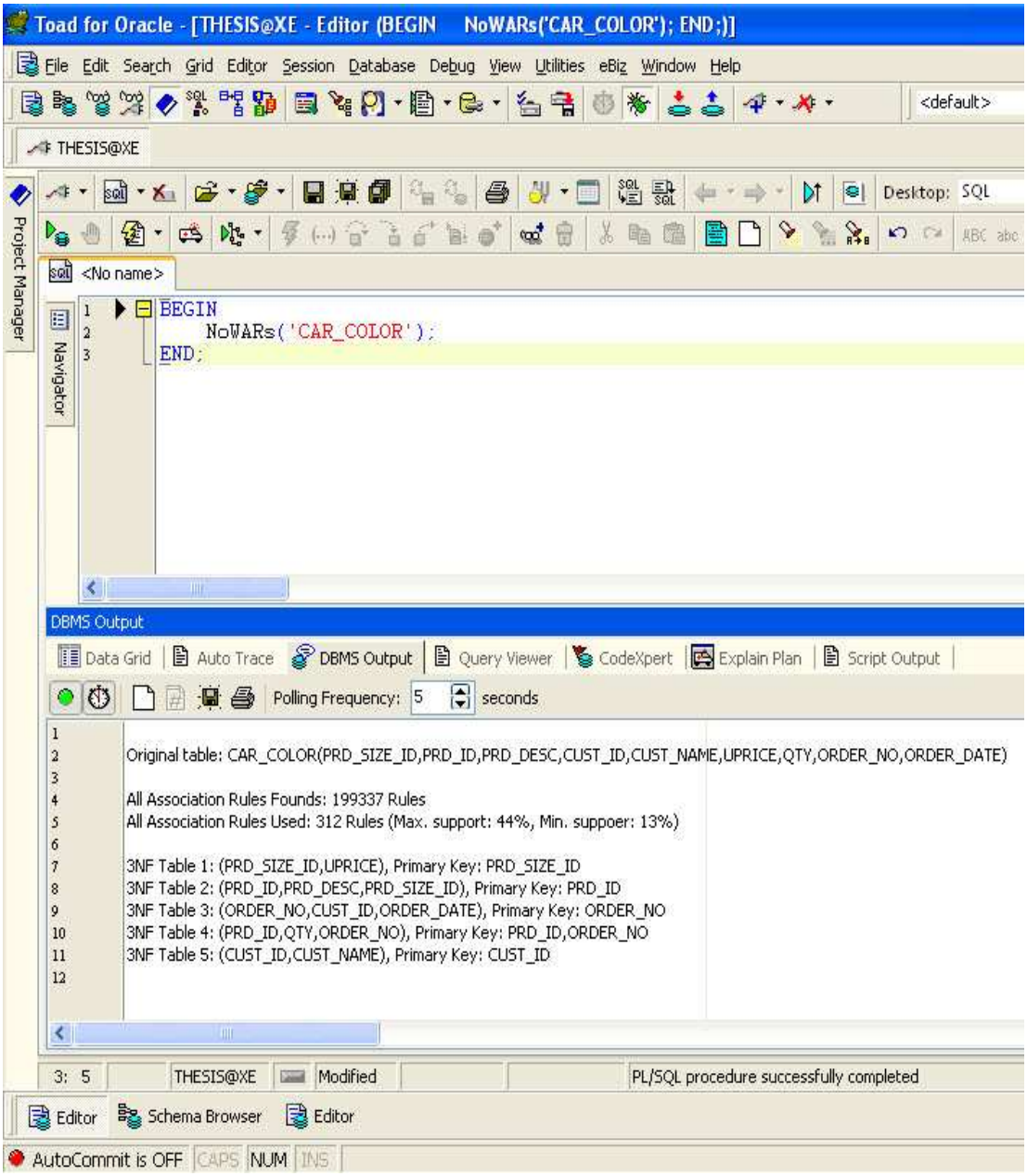
ทำให้ต้องสูญเสียข้อมูลสินค้า ที่มีรหัสสินค้า (PRODUCT_ID) เป็น “5” ซึ่งก็คือ “Writer Desk” ไปด้วย เนื่องจากไม่มีการอ้างอิงถึงสินค้าชิ้นนี้ในข้อมูลแถวอื่น ๆ

2) ปัญหาในการปรับปรุงแก้ไขข้อมูล

หากมีลูกค้าเปลี่ยนแปลงที่อยู่และต้องการแก้ไขข้อมูลของลูกค้ารายนั้น เช่น ร้าน “Value Furniture” ย้ายไปอยู่ที่ “Dallas-TX” ก็ต้องมีการเปลี่ยนแปลงข้อมูลในหลายแถวที่มีรหัสลูกค้า (CUST_ID) เป็น “1006” ซึ่งนอกจากจะต้องใช้เวลาในการแก้ไขแล้ว ก็ยังไม่สามารถรับประกันได้ว่า ได้มีการแก้ไขข้อมูลครบแล้ว ซึ่งหากแก้ไขได้ไม่ครบก็จะส่งผลให้ข้อมูลเกิดความขัดแย้งกันได้

4.2.5 ผลการทดสอบกับข้อมูลการทำสีรถยนต์

เมื่อนำข้อมูลการทำทำสีรถยนต์ ไปทดสอบกับโปรแกรม NoWARs โปรแกรมจะให้ผลการทดสอบดังรูปที่ 4.5



รูปที่ 4.5 ผลลัพธ์การปรับรูปแบบบรรทัดฐานของข้อมูลการทำสีรถยนต์

จากรูปที่ 4.5 จะเห็นว่าโปรแกรมทำการแยกรีเลชันได้ทั้งหมด 5 รีเลชัน ซึ่งประกอบไปด้วย

Table1 (PRD_SIZE_ID, UPRICE), Primary Key: PRD_SIZE_ID

Table2 (PRD_ID, PRD_DESC, PRD_SIZE_ID), Primary Key: PRD_ID

Table3 (ORDER_NO, CUST_ID, ORDER_DATE), Primary Key: ORDER_NO

Table4 (PRD_ID, QTY, ORDER_NO), Primary Key: PRD_ID, ORDER_NO

Table5 (CUST_ID, CUST_NAME), Primary Key: CUST_ID

หากนำรีเลชันทั้ง 5 ไปสร้างเป็นตารางจัดเก็บข้อมูล จะได้รายละเอียดการจัดเก็บข้อมูลดังตารางที่ 4.22-4.26 ตามลำดับ

ตารางที่ 4.22 Table1_Car_Color

PRD_SIZE_ID	**UPRICE**
1mm	800
2mm	1500
3mm	2300
4mm	3000
5mm	3800

ตารางที่ 4.23 Table2_Car_Color

PRD_ID	**PRD_SIZE_ID**	**PRD_DESC**
4	1mm	Green Acrylic-1
5	3mm	Green Acrylic-3
6	2mm	Blue Acrylic-2
7	5mm	Red Acrylic-5
8	3mm	Blue Acrylic-3
9	2mm	Green Acrylic-2
11	4mm	Blue Acrylic-4

ตารางที่ 4.24 Table3_Car_Color

ORDER_NO	**CUST_ID**	**ORDER_DATE**
171	3	12/24/2008
172	6	12/26/2008
174	8	12/26/2008
175	4	12/26/2008
176	2	12/27/2008

ตารางที่ 4.25 Table4_Car_Color

PRD_ID	**QTY**	**ORDER_NO**
8	4	174
7	2	171
4	1	171
11	1	172
5	2	174
9	5	174
5	2	175
11	1	175
6	2	175
4	4	175
7	7	176
11	2	176
4	4	176
9	10	177
5	2	177
4	3	172

ตารางที่ 4.26 Table5_Car_Color

CUST_ID	CUST_NAME
2	Niyom
3	Niyom
4	Sincharoen
6	Bancha
8	Nimitr
8	Nimitr

เมื่อพิจารณาตารางที่ 4.22-4.26 จะเห็นว่าทุกตารางไม่มีปัญหาเรื่อง ความซ้ำซ้อนของข้อมูล และหากต้องการที่จะทำการลบข้อมูล เพิ่มข้อมูล หรือแม้กระทั่งทำการแก้ไขข้อมูล ก็สามารถทำได้ โดยที่ไม่มีความผิดปกติใด ๆ เกิดขึ้นในฐานข้อมูล แต่หากพิจารณาตารางที่ 4.5 ซึ่งเป็นตารางที่ยังไม่ผ่านกระบวนการปรับรูปแบบบรรทัดฐานด้วยโปรแกรม NoWARs จะเห็นว่ายังคงมีปัญหาที่อาจเกิดขึ้นได้ดังต่อไปนี้

1) ปัญหาในการลบข้อมูล

หากต้องการยกเลิกรายการสั่งซื้อ (ORDER_NO) หมายเลข “175” จะมีผลทำให้ต้องลบข้อมูลในแถวที่ 7-10 ทิ้งไป ซึ่งการลบข้อมูลในแถวที่ 9 ทิ้ง จะทำให้ต้องสูญเสียข้อมูลของสินค้าที่มีรหัสสินค้า (PRD_ID) เป็น “6” ทิ้งไป เนื่องจากไม่มีการอ้างอิงถึงสินค้านี้ในข้อมูลแถวอื่น ๆ

2) ปัญหาในการแก้ไขข้อมูล

หากต้องการปรับราคาต่อหน่วย (UPRICE) ของ “Green Acrylic-3” จากเดิม “2300” เป็น “2000” ก็ต้องทำการแก้ไขข้อมูลในหลายแถวที่มีรหัสสินค้า (PRD_ID) เป็น “5” ซึ่งถ้าแก้ไขข้อมูลไม่ครบแล้ว จะทำให้ข้อมูลเกิดความขัดแย้งกันได้ เนื่องจากสินค้าตัวเดียวกันแต่มีราคาต่อหน่วยต่างกัน คือ “2300” และ “2000”

4.3 ผลการคัดเลือกกฎความสัมพันธ์

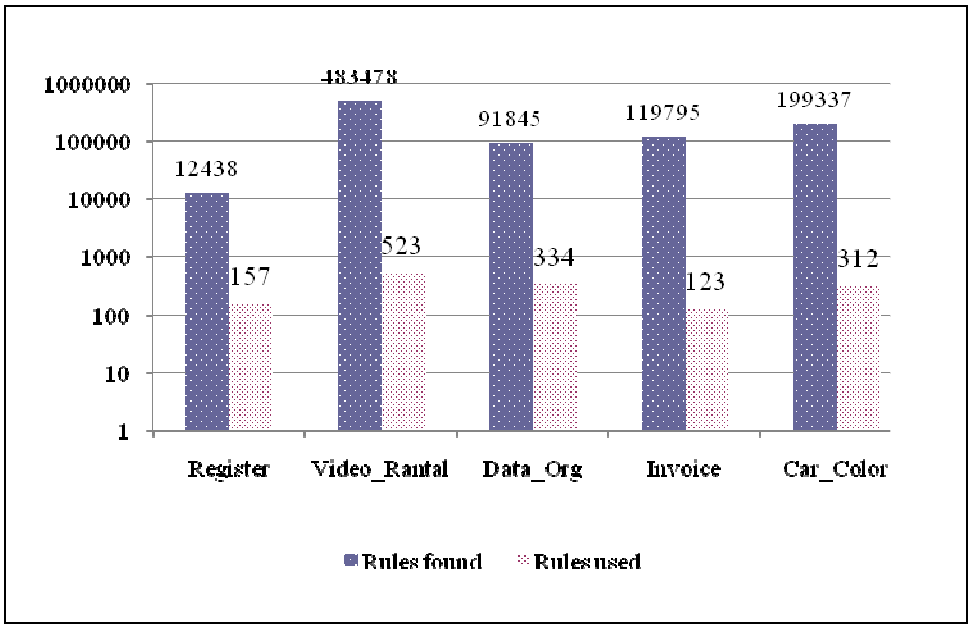
การค้นหาความสัมพันธ์ของข้อมูลใด ๆ ด้วยเทคนิคการวิเคราะห์ความสัมพันธ์ ผลลัพธ์ที่ออกมาจะเป็นกฎความสัมพันธ์จำนวนมากที่แสดงความสัมพันธ์ของข้อมูล ทั้งนี้จำนวนกฎความสัมพันธ์ที่ได้จะมากหรือน้อยขึ้นอยู่กับข้อมูลที่ใช้ทดสอบ ยิ่งข้อมูลมีจำนวนมากก็ยิ่งได้กฎความสัมพันธ์มากตามไปด้วย ซึ่งในงานวิจัยชิ้นนี้ได้ใช้ข้อมูลทดสอบจำนวน 5 ชุด และเมื่อนำ

ข้อมูลแต่ละชุดไปทดสอบเพื่อค้นหาความสัมพันธ์ด้วยโปรแกม NoWARs โปรแกรมจะให้ผลลัพธ์เป็นกฎความสัมพันธ์จำนวนมาก ซึ่งในกระบวนการ การปรับรูปแบบบรรทัดฐานจะไม่นำกฎความสัมพันธ์ทุกกฎที่ค้นพบโดยโปรแกรมมาใช้ แต่คัดเลือกกฎความสัมพันธ์เพียงบางส่วนมาใช้เท่านั้น โดยจำนวนกฎความสัมพันธ์ที่ค้นพบด้วยโปรแกรม NoWARs และจำนวนกฎความสัมพันธ์ที่นำไปใช้ในกระบวนการปรับรูปแบบบรรทัดฐานแสดงดังตารางที่ 4.27

ตารางที่ 4.27 แสดงจำนวนกฎความสัมพันธ์ที่ค้นพบ จำนวนกฎความสัมพันธ์ ค่า Minimum support ของกฎความสัมพันธ์ และ ค่า Maximum support ของกฎความสัมพันธ์ที่ใช้ในกระบวนการปรับรูปแบบบรรทัดฐาน

ชื่อชุดข้อมูล	**จำนวนกฎความสัมพันธ์ที่ค้นพบ**	**จำนวนกฎความสัมพันธ์ที่ใช้**	**ค่า Minimum support ของกฎความสัมพันธ์ที่ใช้**	**ค่า Maximum support ของกฎความสัมพันธ์ที่ใช้**
Register	12438	157	17%	75%
Video_Rental	483478	523	17%	50%
Data_Org	91845	334	10%	33%
Invoice	119795	123	25%	63%
Car_Color	199337	312	13%	44%

พิจารณาจากตารางที่ 4.27 จะเห็นว่า ข้อมูลแต่ละชุดสามารถค้นหากฎความสัมพันธ์ได้เป็นจำนวนมาก แต่กระบวนการปรับรูปแบบบรรทัดฐานด้วยโปรแกรม NoWARs กลับใช้จำนวนกฎความสัมพันธ์เพียงเล็กน้อยเท่านั้น ซึ่งแสดงให้เห็นว่าโปรแกรมสามารถคัดเลือกกฎความสัมพันธ์สำหรับกระบวนการปรับรูปแบบบรรทัดฐานได้อย่างชาญฉลาด และจากตารางที่ 4.27 สามารถแสดงความแตกต่างของจำนวนกฎความสัมพันธ์ที่ค้นพบ และจำนวนกฎความสัมพันธ์ที่ใช้ในกระบวนการปรับรูปแบบบรรทัดฐานด้วยโปรแกรม NoWARs ในรูปแบบของกราฟได้ดังรูปที่ 4.6



รูปที่ 4.6 แสดงจำนวนกฎความสัมพันธ์ที่ค้นพบและจำนวนกฎความสัมพันธ์ที่ใช้ในกระบวนการปรับรูปแบบบรรทัดฐาน

4.4 การอภิปรายผล

จากผลการทดสอบโปรแกรมด้วยข้อมูลทั้งห้าชุดพบว่า โปรแกรม NoWARs สามารถทำการปรับรูปแบบบรรทัดฐานได้ถูกต้องทั้งหมด โดยผลลัพธ์ที่ได้จะได้ตารางที่อยู่ในรูปแบบบรรทัดฐานระดับที่ 3 ซึ่งเป็นตารางที่สามารถแก้ปัญหาความซ้ำซ้อนของข้อมูล ปัญหาในการเพิ่มข้อมูล ปัญหาในการลบข้อมูล หรือแม้กระทั่งปัญหาในการแก้ไขข้อมูลได้ทั้งหมด ในขณะที่ตารางที่ยังไม่ได้มีการปรับรูปแบบบรรทัดฐานยังคงมีปัญหาเหล่านี้อยู่ และในส่วนการคัดเลือกกฎความสัมพันธ์มาใช้สำหรับกระบวนการปรับรูปแบบบรรทัดฐานพบว่า โปรแกรมสามารถคัดเลือกกฎความสัมพันธ์มาใช้ได้อย่างชาญฉลาด สังเกตได้จากความแตกต่างของจำนวนกฎความสัมพันธ์ที่ค้นพบโดยอัลกอริทึม Apriori และจำนวนกฎความสัมพันธ์ที่ใช้ในกระบวนการปรับรูปแบบบรรทัดฐานโดยโปรแกรม NoWARs

บทที่ 5
สรุปผลการวิจัยและข้อเสนอแนะ

ระบบการจัดการฐานข้อมูล (Database Management System: DBMS) กลายเป็นระบบที่จำเป็นที่ทุกบริษัทหรือทุกองค์กรต้องมี เพื่อจัดเก็บและจัดการกับข้อมูลที่สำคัญของบริษัท แล้วนำข้อมูลเหล่านั้นไปวิเคราะห์ หรือประมวลผลเพื่อให้เกิดประโยชน์สูงสุดแก่หน่อยงาน หรือองค์กรนั้น ๆ แต่ก่อนที่จะได้ระบบการจัดการฐานข้อมูลที่มีประสิทธิภาพออกมา ก็ต้องมาจากการออกแบบระบบจัดการฐานข้อมูลที่ดีด้วย และขั้นตอนหนึ่งที่สำคัญของการออกแบบระบบจัดการฐานข้อมูล คือ การออกแบบตารางหรือรีเลชันสำหรับจัดเก็บข้อมูล หากตารางหรือรีเลชันที่ออกแบบมามีประสิทธิภาพ ก็ย่อมทำให้ได้ระบบการจัดการฐานข้อมูลที่มีประสิทธิภาพออกมาด้วย ในทางกลับกัน หากตารางหรือรีเลชันที่ออกแบบมา ไม่มีประสิทธิภาพก็ย่อมทำให้ได้ระบบการจัดการฐานข้อมูลที่ไม่มีประสิทธิภาพออกมาเช่นกัน

งานวิจัยนี้จึงมุ้งเน้นที่จะพัฒนากระบวนการในการออกแบบ หรือการปรับรีเลชัน เพื่อให้ได้รีเลชันที่อยู่ในรูปแบบที่เป็นบรรทัดฐาน เพื่อลดปัญหาต่าง ๆ ที่อาจเกิดขึ้นได้ เช่น ปัญหาในการลบข้อมูล ปัญหาในการเพิ่มข้อมูล ปัญหาในการปรับปรุงแก้ไขข้อมูล รวมถึงปัญหาความซ้ำซ้อนของข้อมูล เป็นต้น กระบวนการปรับรูปแบบบรรทัดฐานในงานวิจัยชินนี้ จะเป็นการนำทฤษฎีการปรับรูปแบบบรรทัดฐาน มาประยุกต์ใช้ร่วมกับเทคนิคการค้นหาความสัมพันธ์ของข้อมูล ซึ่งเป็นเทคนิคการทำเหมืองข้อมูลประเภทการค้นหาความสัมพันธ์ของข้อมูล โดยอัลกอริทึมที่ทำการค้นหาความสัมพันธ์ของข้อมูลที่นำมาใช้คือ อัลกอริทึมเอไพรออริ ซึ่งถือเป็นอัลกอริทึมที่ได้รับความนิยมสูงสุดในการทำเหมืองข้อมูลประเภทการค้นหาความสัมพันธ์ของข้อมูล

สำหรับขั้นตอนการวิจัยในงานวิจัยชิ้นนี้แบ่งออกเป็น การศึกษาขั้นตอนวิธีการทำงานของอัลกอริทึมเอไพรออริ โดยใช้โปรแกรม WEKA ซึ่งเป็นโปรแกรมสำเร็จรูปที่เปิดเผยซอร์ดโค้ดเป็นต้นแบบในการศึกษา เพื่อนำผลลัพธ์ที่ได้มาประยุกต์ใช้ร่วมกับเทคนิคการปรับรูปแบบบรรทัดฐานสำหรับการปรับรีเลชันให้อยู่ในรูปแบบบรรทัดฐานต่อไป และในงานวิจัยชิ้นนี้ก็ได้มีการพัฒนาโปรแกรม NoWARs ซึ่งเป็นโปรแกรมที่ช่วยในการค้นหาความสัมพันธ์ระหว่างข้อมูล โดยผลลัพธ์ที่ได้จะอยู่ในรูปของกฎความสัมพันธ์ นอกจากนี้โปรแกรมยังทำหน้าที่แปลงกฎความสัมพันธ์ที่ได้ให้เป็นรีเลชันที่อยู่ในรูปแบบบรรทัดฐาน โดยแสดงผลลัพธ์ออกมาในรูปของรีเลชันนอลสคีมา

การทดสอบประสิทธิภาพของโปรแกรม NoWARs จะทำการทดสอบกับข้อมูลทั้งหมด 5 ชุด โดยจะเปรียบเทียบระหว่างรีเลชันที่ถูกปรับให้อยู่ในรูปแบบบรรทัดฐานแล้วโดยโปรแกรม NoWARs กับรีเลชันที่ไม่ได้ถูกปรับให้อยู่ในรูปแบบบรรทัดฐานที่เหมาะสม ซึ่งประสิทธิภาพของรีเลชันสามารถวัดได้จาก ปัญหาความซ้ำซ้อนของข้อมูล ปัญหาในการเพิ่มข้อมูล ปัญหาในการลบข้อมูล และปัญหาในการปรับปรุงแก้ไขข้อมูล

5.1 สรุปผลการวิจัย

ในการทดสอบประสิทธิภาพของรีเลชันที่ได้จากการปรับรูปแบบโดยโปรแกรม NoWARs พบว่า รีเลชันที่ได้จากการปรับให้อยู่ในรูปแบบบรรทัดฐานโดยโปรแกรม NoWARs ทุกรีลชันเป็นรีเลชันที่อยู่ในรูปแบบบรรทัดฐานระดับที่ 3 ซึ่งสามารถแก้ไขปัญหาความซ้ำซ้อนของข้อมูล ปัญหาในการเพิ่มข้อมูล ปัญหาในการลบข้อมูล และปัญหาในการแก้ไขข้อมูล ได้ทั้งหมด ในขณะที่รีเลชันที่ยังไม่ได้มีการปรับให้อยู่ในรูปแบบบรรทัดฐานที่เหมาะสม ไม่สามารถที่จะแก้ไขปัญหาเหล่านี้ได้ ทั้งยังต้องสูญเสียพื้นที่จำนวนมากในการเก็บข้อมูลที่ซ้ำ ๆ กันโดยไม่จำเป็น ดังนั้นการปรับรีเลชันด้วยโปรแกรม NoWARs จะส่งผลให้ได้ระบบการจัดการฐานข้อมูลที่มีประสิทธิภาพกว่า รีเลชันที่ไม่ได้ปรับให้อยู่ในรูปแบบบรรทัดฐานแน่นอน

5.2 การประยุกต์งานวิจัย

งานวิจัยชิ้นนี้เป็นการนำเทคนิควิธีที่มีอยู่แล้วคือ กระบวนการปรับรูปแบบบรรทัดฐาน และเทคนิคการวิเคราะห์ความสัมพันธ์ของข้อมูล มาประยุกต์ใช้ร่วมกันเป็นเทคโนโลยีใหม่สำหรับการออกแบบตารางเก็บข้อมูลให้มีประสิทธิภาพ และส่งผลให้ระบบการจัดการฐานข้อมูลมีประสิทธิภาพมากยิ่งขึ้น งานวิจัยชิ้นนี้จึงเป็นเพียงแนวทางอีกแนวทางหนึ่งที่จะพัฒนาขั้นตอนวิธีในการปรับรีเลชันให้อยู่ในรูปแบบที่เป็นบรรทัดฐาน โดยการคัดเลือกกฎความสัมพันธ์ที่ได้จากการค้นหาความสัมพันธ์ของข้อมูล มาแปลงให้อยู่ให้อยู่ในรูปของรีเลชันนอลสคีมา ซึ่งงานวิจัยชิ้นนี้ยังคงเป็นแค่แนวทางต้นแบบขั้นต้นสำหรับงานด้านการปรับรูปแบบบรรทัดฐานเท่านั้น ดังนั้นจึงสามารถทำการวิจัยพัฒนาต่อ เพื่อให้มีประสิทธิภาพมากยิ่งขึ้นไปอีกได้ จึงน่าจะเป็นประโยชน์สำหรับนักวิจัยในอนาคต เพื่อเป็นจุดเริ่มต้นสำหรับการวิจัยงานในลักษณะนี้ต่อไป

5.3 ปัญหาและข้อเสนอแนะ

ในกระบวนการปรับรูปแบบบรรทัดฐาน ประกอบไปด้วยระดับขั้นของรูปแบบบรรทัดฐานจำนวนทั้งหมด 5 ระดับ ซึ่งในการปรับรูปแบบบรรทัดฐานในขั้นต่อ ๆ ไปต้องอาศัยรีเลชันที่อยู่

ในรูปแบบบรรทัดฐานในระดับก่อนหน้า เป็นเช่นนี้ไปเรื่อย ๆ ซึ่งโปรแกรม NoWARs สามารถทำการปรับรูปแบบบรรทัดฐานได้ในระดับหนึ่ง หากจะพัฒนาโปรแกรมให้มีประสิทธิมากยิ่งขึ้น สามารถทำได้หลายแนวทาง อาทิ เช่น

- กระบวนการปรับรีเลชันให้อยู่ในรูปแบบที่เป็นบรรทัดฐาน โปรแกรม NoWARs สามารถปรับรีเลชันได้ถึงรูปแบบบรรทัดฐานระดับที่ 3 (3NF) เท่านั้น ไม่สามารถที่จะทำการปรับรีเลชันให้อยู่ในรูปแบบบรรทัดฐานระดับที่สูงกว่านี้ได้ ซึ่งในบางครั้งการปรับรีเลชันถึงแค่รูปแบบบรรทัดฐานระดับที่ 3 อาจไม่เพียงพอสำหรับข้อมูลบางประเภท จึงมีความจำเป็นต้องปรับรีเลชันให้อยู่ในระดับที่สูงกว่าระดับที่ 3 เพื่อประสิทธิภาพที่ดีของระบบการจัดการฐานข้อมูล
- พัฒนาขั้นตอนการตัดสินใจเลือกแอททริบิวท์ที่จะทำหน้าที่คีย์หลักของรีเลชัน ซึ่งโปรแกรม NoWARs สามารถระบุแอททริบิวท์ที่จะมาทำหน้าที่เป็นคีย์หลักของรีเลชันได้เฉพาะแอททริบิวท์ที่มีคำเหล่านี้เป็นส่วนประกอบ คือ ID, CODE, NO, NUM และ NUMBER หากปราศจากคำเหล่านี้แล้ว โปรแกรมไม่สามารถที่หาแอททริวท์ที่จะมาทำหน้าที่เป็นคีย์หลักที่แน่นอนได้ ซึ่งจะเกิดขึ้นเมื่อกรณีที่มีแอทริบิวท์ที่มีคุณสมบัติเป็นคีย์คู่แข่งเกิดขึ้นในรีเลชันนั้น ๆ

รายการอ้างอิง

สมจิตร อาจอินทร์ และ งามนิจ อาจอินทร์. (2549). **หลักการวิเคราะห์และออกแบบระบบฐานข้อมูล**. ขอนแก่น: ขอนแก่นการพิมพ์.

ศิริลักษณ์ โรจนกิจอำนวย. (2545). **การออกแบบและบริหารฐานข้อมูล**. กรุงเทพมหานคร: มหาวิทยาลัยธรรมศาสตร์.

Agrawal, R., Imielinski, T. and Swami, A. (1993). Mining association rules between set of items in large databases. **Proceedings of ACM SIGMOD International Conference on Management of Data**, pp. 207-216.

Agrawal, R. and Srikant R. (1994). Fast algorithms for mining association rules in large database. **Proceedings of the 20th International Conference on Very Large Data Base**, pp. 487-499.

Armstrong, W. W. (1974). Dependency structures of database relationships. **Information Processing 74**, North-Holland, Amsterdam, pp. 580-583.

Berzal, F., Cubero, J., Marín, N. and Serrano, J. (2001). TBAR: An efficient method for association rule mining in relational databases. **Data & Knowledge Engineering**, 37(April 1), pp. 47-64.

Codd, E. F. (1970). A relational model of data for large shared data banks. **Communications of the ACM**, 13(June 6), pp. 377 – 387.

Codd, E. F. (1971). Further normalization of the database relational model. **Courant Computer Science Symposium 6**, Database Systems, Prentice-Hall, Englewood Cliffs, N.J., pp. 65-98.

Codd, E.F. (1974). Recent INVestigations in relational data base systems. **Information Processing 74**, North-Holland Pub. Co., Amsterdam, pp. 1017-1021.

Codd, E. F. (1982). Relational database: a practical foundation for productivity. **Communications of the ACM**, 25(February 2), pp. 109 – 117.

Connolly, T. and Begg, C. (2002). **Database System.** Harlow: Pearson Education Limited.

Date, C. J. (200). **An Introduction to Database Systems 7th edn**. MA: Adison-Wesley.

Date, C. J. and Fagin, R. (1992). Simple conditions for guaranteeing higher normal forms in relational databases. **ACM Transactions on Database Systems (TODS)**, 17 (September 3), pp. 465 – 476.

Dehaspe, L. and De Raedt, L. (1997). **Inductive logic programing**. London: Springer-Verlag.

Fagin, Ronald. (1977). Multivalued dependencies and a new normal form for relational databases. **ACM Transactions on Database Systems (TODS)**, 2(September 3), pp. 262-278.

Fagin, Ronald. (1979). Normal forms and relational database operators. **Proceedings of the 1979 ACM SIGMOD International Conference on Management of Data**, Boston, Massachusetts, pp. 153-160.

Frank, E., Hall, M., Holmes, G., Martin, B., Mayo, M., Pfahringer, B., Smith, T. and Witten, I. (2008). **Waikato Environment for Knowledge Analysis: Weka-3-5-8**. University of Waikato: New Zealand.

Han, J. et el. (1997). DBMiner: system for data mining in relational databases and data warehouses. **IBM Centre for Advance Studies Conference**, Toronto, Ontario, Canada, p. 8.

Han, J. and Kamber, M. (2001). **Data Mining: Concepts and Techniques**. San Diego: Academic Press.

Hipp, J., Güntzer, U., and Grimmer U. (2001). Integrating association rule mining algorithms with relational database systems. **In Proceedings of the 3rd International Conference on Enterprise Information Systems (ICEIS 2001)**, Setúbal, Portugal.

Thomas, S., Bodogala, S., Alsabti, K., and Ranka, S. (1997). An Efficient Algorithm for the Incremental Updation of Association Rules in Large Databases. **Proceedings of the 3rd International Conference On KDD And Data Mining, California**, USA.

Saar Tsechansky, M., Pliskin, N., Rabinowitz G., and Porath,A. (1999). Mining relational patterns from multiple relational tables. **Decision Support Systems**, 27(November 1-2), pp. 179-195.

ภาคผนวก ก

บทความผลงานวิจัยที่นำเสนอในการประชุมเสนอผลงานวิจัยระดับบัณฑิตศึกษาแห่งชาติ ครั้งที่ 11
วันที่ 17 – 18 ธันวาคม 2551
ณ มหาวิทยาลัยราชภัฏวไลยอลงกรณ์ ในพระบรมราชูปถัมภ์ จ. ปทุมธานี

การปรับรูปแบบบรรทัดฐานในฐานข้อมูลเชิงสัมพันธ์ ด้วยเทคนิคการวิเคราะห์ความสัมพันธ์

ณัฐพล พันนุรัตน์, นิตยา เกิดประสพ และ กิตติศักดิ์ เกิดประสพ

สาขาวิชาวิศวกรรมคอมพิวเตอร์ สำนักวิศวกรรมศาสตร์
มหาวิทยาลัยเทคโนโลยีสุรนารี อ. เมือง จ. นครราชสีมา 30000

บทคัดย่อ

การออกแบบตารางสำหรับเก็บข้อมูลนับเป็นจุดเริ่มต้นที่สำคัญของระบบฐานข้อมูล หากออกแบบตารางไม่ดี อาจก่อให้เกิดปัญหาต่าง ๆ ตามมา เช่น ปัญหาการเพิ่มข้อมูล ปัญหาการปรับปรุงหรือแก้ไขข้อมูล เป็นต้น จึงเป็นที่มาของการคิดค้นกระบวนการปรับรูปแบบบรรทัดฐานเพื่อแก้ไขปัญหาดังกล่าว งานวิจัยนี้จึงมุ่งเน้นที่จะพัฒนาระบบที่มีความสามารถในการปรับรูปแบบบรรทัดฐาน โดยจะนำเทคนิคการค้นหากฎความสัมพันธ์เข้ามาช่วยในการปรับรูปแบบบรรทัดฐานให้อยู่ในรูปแบบ 3NF เนื่องจากเป็นรูปแบบที่เหมาะสมในการใช้งานจริง

คำสำคัญ: ดาต้าไมนิ่ง, แอสโซซิเอชัน, เอไพรออริ, การปรับรูปแบบบรรทัดฐาน

บทนำ

ระบบการจัดการฐานข้อมูล (Database Management System : DBMS) ทุกระบบจะต้องมีตารางไว้สำหรับเก็บข้อมูลเสมอ ยิ่งข้อมูลมีรายละเอียดมากก็ต้องมีแอททริบิวท์ (Attribute) สำหรับเก็บข้อมูลมากเท่านั้น การออกแบบตารางเก็บข้อมูลที่ไม่ดีอาจส่งผลให้มีการเก็บข้อมูลที่ซ้ำซ้อนกันได้ หรือเกิดข้อผิดพลาดในการเรียกใช้ข้อมูล เช่น การเพิ่มข้อมูล การลบหรือแก้ไขข้อมูลที่มีอยู่ในรีเลชัน หรืออาจเกิดความไม่คงที่ ไม่แน่นอนหรือขัดแย้ง (Inconsistency) ของข้อมูลซึ่งเรียกว่าความผิดปกติ (Abnormal) จึงเป็นที่มาของการคิดค้นกระบวนการนอร์มัลไลซ์ (Normalization) หรือกระบวนการปรับรูปแบบบรรทัดฐาน เพื่อแก้ไขปัญหาดังกล่าวขึ้นมา ซึ่งรูปแบบบรรทัดฐานที่นำเสนอโดยบอยส์ (Boyce) และคอดด์ (Codd) (Codd, 1970; Date and Fagin, 1992) สามารถแบ่งออกเป็นระดับต่าง ๆ ได้ดังนี้

1. รูปแบบบรรทัดฐานระดับที่หนึ่ง (First Normal Form: 1NF)
2. รูปแบบบรรทัดฐานระดับที่สอง (Second Normal Form: 2NF)
3. รูปแบบบรรทัดฐานระดับที่สาม (Third Normal Form: 3NF)
4. รูปแบบบรรทัดฐานบอยส์คอดด์ (Boyce-Codd Normal Form: BCNF)
5. รูปแบบบรรทัดฐานระดับที่สี่ (Fourth Normal Form: 4NF)
6. รูปแบบบรรทัดฐานระดับที่ห้า (Fifth Normal Form: 5NF)

ในทางปฏิบัติ การปรับรูปแบบบรรทัดฐานจนกระทั่งถึงขั้นที่ 4 ที่ 5 หรือที่ 6 เกิดขึ้นได้ยากมาก ดังนั้นงานวิจัยชิ้นนี้จึงมุ่งเน้นที่จะพัฒนาแนวทางในการปรับรูปแบบบรรทัดฐานถึงขั้นที่ 3 เท่านั้น โดยจะนำเทคนิคการค้นหาความสัมพันธ์ของข้อมูล ซึ่งเป็นเทคนิคของงานทางด้านการทำเหมืองข้อมูลเข้ามาช่วย

วัตถุประสงค์

1. เพื่อออกแบบตารางในการเก็บข้อมูลได้อย่างมีประสิทธิภาพ
2. เพื่อช่วยลดระยะเวลาในการออกแบบตารางในการเก็บข้อมูล
3. เพื่อทำการออกแบบรูปแบบโมเดลข้อมูลจากการทำเหมืองข้อมูลที่เหมาะสมกับงาน Normalization
4. เพื่อพัฒนาวิธีการแปลงโมเดลข้อมูลให้เป็นตารางข้อมูลที่อยู่ในรูป 3NF

รูปแบบบรรทัดฐานและกฎความสัมพันธ์

การปรับรูปแบบบรรทัดฐานในงานวิจัยนี้จะใช้ความสามารถของการค้นหากฎความสัมพันธ์ของอัลกอริทึมเอไพรออริ (Apriori) (Argawal and Srikant, 1994; Argawal et al., 1993) ซึ่งเป็นงานแขนงหนึ่งของเทคนิคการทำเหมืองข้อมูล (Data mining) และนิยมนำมาใช้ร่วมกับฐานข้อมูลเชิงสัมพันธ์ (Sarawagi et al., 1998) โดยกฎที่ได้จากอัลกอริทึมเอไพรออริ จะอยู่ในรูปของกฎความสัมพันธ์ ถ้า-แล้ว เป็นกฎที่ใช้บอกความสัมพันธ์ในกลุ่มข้อมูลว่า ถ้าเกิดลักษณะหนึ่งแล้วจะเป็นผลให้เกิดอีกลักษณะหนึ่งตามมาเสมอ ยกตัวอย่างพฤติกรรมการเลือกซื้อสินค้าของลูกค้าของร้านค้าแห่งหนึ่ง เป็นดังนี้ ลูกค้าที่ซื้อนมสดจะซื้อขนมปังและน้ำเปล่าด้วยด้วย ซึ่งเขียนความสัมพันธ์ได้เป็น นมสด [support = 5] $\longrightarrow$ ขนมปัง, น้ำเปล่า [support = 5] [confidence = 100%] หมายความว่า ลูกค้าที่ซื้อนมสดมีโอกาสจะซื้อขนมปังและน้ำเปล่าด้วย ความสัมพันธ์นี้มีค่าความเชื่อมั่น 100% ซึ่งมีค่าสนับสนุนเท่ากับ 5 หรืออีกกรณีเช่น ลูกค้าที่ซื้อบะหมี่สำเร็จรูปกับไข่ไก่จะซื้อน้ำเปล่าด้วย ซึ่งเขียนความสัมพันธ์ได้เป็น บะหมี่สำเร็จรูป, ไข่ไก่ [support = 6] $\longrightarrow$ น้ำเปล่า [support = 6] [confidence = 100%] หมายความว่า ลูกค้าที่ซื้อบะหมี่สำเร็จรูปและไข่ไก่มีโอกาสจะ

ซื้อน้ำเปล่าด้วย ความสัมพันธ์นี้มีค่าความเชื่อมั่น 100% ซึ่งมีค่าสนับสนุนเท่ากับ 6 เป็นต้น ซึ่งงานวิจัยนี้จะนำเฉพาะกฎลักษณะแรกมาใช้ กล่าวคือ เป็นกฎที่มีส่วนหัวหรือส่วนที่เป็นสาเหตุเพียงแอททริบิวท์เดียวเท่านั้น อีกทั้งกฎความสัมพันธ์นั้นต้องมีค่าความเชื่อมั่นเป็น 100% และมีค่าสนับสนุนสูง ๆ อีกด้วย

ในแต่ละขั้นตอนของกระบวนการปรับบรรทัดฐาน จะมีการระบุรูปแบบโครงสร้างของข้อมูลที่ควรจะเป็นที่เรียกว่า รูปแบบบรรทัดฐาน (Normal form) ไว้ ซึ่งโครงสร้างที่ระบุนี้จะสามารถแก้ไขปัญหาที่เกิดขึ้นในโครงสร้างข้อมูลขั้นตอนก่อนหน้าได้ หรือกล่าวอีกนัยหนึ่งคือ การปรับรูปแบบบรรทัดฐานแต่ละขั้นตอนจะต้องอาศัยผลที่ได้จากการปรับรูปแบบบรรทัดฐานในขั้นตอนก่อนหน้า เพื่อนำมาปรับปรุงให้มีโครงสร้างตามที่กำหนดไว้ในขั้นตอนนั้น ๆ ซึ่งแต่ละขั้นมีรูปแบบขั้นตอนวิธีดังนี้

รูปแบบบรรทัดฐานระดับที่ 1

กระบวนการปรับบรรทัดฐานระดับแรกสุด จะเป็นกระบวนการในการปรับตารางข้อมูลของผู้ใช้ให้อยู่ในรูปแบบของรีเลชัน กล่าวคือ ค่าของแอททริบิวท์ของแต่ละแถวสามารถมีได้ค่าเดียวเท่านั้น เพื่อให้อยู่ในรูปแบบบรรทัดฐานระดับที่ 1

นิยาม: รีเลชันใดจะอยู่ในรูปแบบบรรทัดฐานระดับที่ 1 ได้ก็ต่อเมื่อ รีเลชันนั้นจะต้องมีคุณสมบัติต่อไปนี้

1. เป็นรีเลชันที่มีคีย์หลักของรีเลชัน
2. ไม่มีกลุ่มซ้ำอยู่ในรีเลชัน หรืออาจกล่าวได้ว่าค่าของแต่ละแอททริบิวท์ของแต่ละแถวสามารถมีได้ค่าเดียวเท่านั้น
3. แอททริบิวท์ทุกแอททริบิวท์ที่ไม่ใช่คีย์ จะต้องขึ้นกับแอททริบิวท์ที่เป็นคีย์หลักอย่างสมบูรณ์

ตัวอย่างการปรับรีเลชันให้อยู่ในรูปแบบบรรทัดฐานระดับที่ 1 สามารถอธิบายได้ดังตัวอย่างต่อไปนี้

ตารางที่ 1 การสั่งสินค้า Unnormal Form

เลขที่ใบสั่ง	วันที่สั่ง	รหัสลูกค้า	ชื่อลูกค้า	รหัสสินค้า	ชื่อสินค้า	จำนวน
INV001	20-08-2008	CU001	สาทร บุญมาก	CA101	CPU	20
				RA300	Ram	40
INV002	20-08-2008	CU002	บุญมี ใจดี	MB205	Broad	10
INV003	21-08-2008	CU003	บุญมา ดีใจ	RA300	Ram	50

จากตารางที่ 1 จะเห็นว่าเลขที่ใบสั่ง 1 รหัส (คีย์หลัก) ประกอบไปด้วยสินค้ามากกว่า 1 รายการ ดังนั้นจึงกล่าวได้ว่า แอททริบิวท์รหัสการสินค้า ชื่อสินค้า และจำนวนเป็นกลุ่มซ้ำของรีชัน ซึ่งไม่ตรงกับคุณสมบัติของรูปแบบบรรทัดฐานระดับที่ 1 ดังนั้นการปรับในระดับนี้ก็ได้แก่การขจัดกลุ่มซ้ำออกไป ดังที่ได้นิยามไว้ ดังนี้

ตารางที่ 2 การสั่งสินค้า 1NF

เลขที่ใบสั่ง	วันที่สั่ง	รหัสลูกค้า	ชื่อลูกค้า	รหัสสินค้า	ชื่อสินค้า	จำนวน
INV001	20-08-2008	CU001	สาทร บุญมาก	CA101	CPU	20
INV001	20-08-2008	CU001	สาทร บุญมาก	RA300	Ram	40
INV002	20-08-2008	CU002	บุญมี ใจดี	MB205	Broad	10
INV003	21-08-2008	CU003	บุญมา ดีใจ	RA300	Ram	50

วิธีการขจัดกลุ่มซ้ำสามารถทำได้โดยแยกข้อมูลของเลขที่ใบสั่งซื้อ INV001 ออกเป็น 2 แถว ดังแสดงในตารางที่ 2 ซึ่งผลที่ได้นี้จะเห็นว่าเลขที่ใบสั่งไม่ใช่คีย์หลักของรีเลชันอีกต่อไป แต่คีย์หลักจะประกอบด้วยเลขที่ใบสั่งและรหัสสินค้า ซึ่งเราอาจกล่าวได้ว่าการปรับรีเลชันให้อยู่ในรูปแบบบรรทัดฐานระดับที่ 1 นั้นจะต้องมีการเพิ่มแอททริบิวท์ที่มาเป็นคีย์หลักเสมอ โดยคีย์หลักตัวใหม่จะประกอบไปด้วยคีย์ตัวเดิมผนวกกับแอททริบิวท์ที่เป็นคีย์หลักของกลุ่มซ้ำ

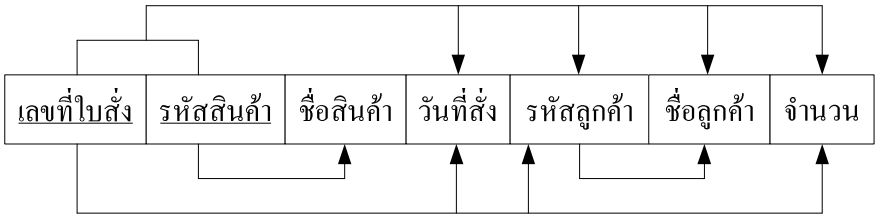
รูปแบบบรรทัดฐานระดับที่ 2

รูปแบบบรรทัดฐานระดับที่ 2 นี้จะเกี่ยวกับเรื่องความสัมพันธ์ระหว่างคีย์หลักและแอททริบิวท์อื่นที่ไม่ได้เป็นส่วนใดส่วนหนึ่งของคีย์หลัก

นิยาม: รีเลชันใดจะอยู่ในรูปแบบบรรทัดฐานระดับที่ 2 ก็ต่อเมื่อ รีเลชันนั้นจะต้องมีคุณสมบัติต่อไปนี้

1. รีเลชันนั้นอยู่ในรูปแบบบรรทัดฐานระดับที่ 1
2. ต้องไม่มีการขึ้นต่อกันเพียงบางส่วน กล่าวคือ ต้องไม่มีแอททริบิวท์ที่ไม่ใช่คีย์หลักตัวใดขึ้นกับคีย์หลักหรือส่วนใดส่วนหนึ่งของคีย์หลัก

ขั้นตอนการปรับรีเลชันให้อยู่ในรูปแบบบรรทัดฐานระดับที่ 2 สามารถแสดงได้ดังนี้ จากตารางที่ 2 สามารถเขียนไดอะแกรมของฟังก์ชันการขึ้นต่อกันได้ดังนี้



รูปที่ 1 ไดอะแกรมการขึ้นแก่กันของรีเลชันการสั่งสินค้า

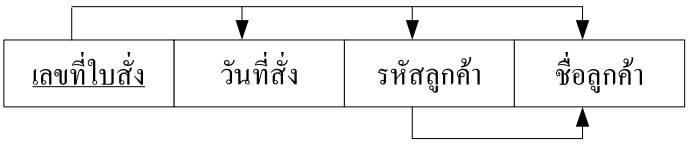
วิธีการที่จะทำให้รีเลชันอยู่ในรูปแบบบรรทัดฐานระดับที่ 2 กระทำได้โดยการสร้างรีเลชันขึ้นมาใหม่สำหรับการขึ้นแก่กันที่เป็นปัญหา ซึ่งจะได้รีเลชันใหม่ดังนี้

การสั่งสินค้า(เลขที่ใบสั่ง, วันที่สั่ง, รหัสลูกค้า, ชื่อลูกค้า)

สินค้า(รหัสสินค้า, ชื่อสินค้า)

รายการสั่งสินค้า(เลขที่ใบสั่ง, รหัสสินค้า, จำนวน)

โดยสามารถเขียนไดอะแกรมการขึ้นแก่กันของแต่ละรีเลชันได้ดังนี้



รูปที่ 2 ไดอะแกรมการขึ้นแก่กันของรีเลชันการสั่งสินค้า

รูปที่ 3 ไดอะแกรมการขึ้นแก่กันของรีเลชันสินค้า

รูปที่ 4 ไดอะแกรมการขึ้นแก่กันของรีเลชันรายการสั่งสินค้า

รูปแบบบรรทัดฐานระดับที่ 3

รีเลชันที่มีรูปแบบบรรทัดฐานระดับที่ 3 จะเป็นรีเลชันที่มีนิยามดังต่อไปนี้

นิยาม: รีเลชันใดจะอยู่ในรูปแบบบรรทัดฐานระดับที่ 3 ก็ต่อเมื่อ รีเลชันนั้นจะต้องมีคุณสมบัติต่อไปนี้

1. รีเลชันนั้นอยู่ในรูปแบบบรรทัดฐานระดับที่ 2
2. จะต้องไม่มีการขึ้นต่อกันแบบทรานซิทีฟกล่าวคือ ต้องไม่มีแอททริบิวท์ที่ไม่ใช่คีย์หลักตัวใดขึ้นกับแอททริบิวท์อื่น ซึ่งเป็นแอททริบิวท์ที่ไม่ใช่คีย์หลักเช่นกัน

ขั้นตอนการปรับรีเลชันให้อยู่ในรูปแบบบรรทัดฐานระดับที่ 3 สามารถแสดงได้ดังนี้ จากรูปที่ 2 จะเห็นว่ายังคงมีปัญหาการขึ้นแก่กัน จึงจำเป็นต้องสร้างรีเลชันขึ้นมาใหม่เพื่อแก้ไขปัญหาที่เกิดขึ้น ซึ่งได้ผลดังนี้

การสั่ง(<u>เลขที่ใบสั่ง</u>, วันที่สั่ง, รหัสลูกค้า)
ลูกค้า(<u>รหัสลูกค้า</u>, ชื่อลูกค้า)

เมื่อเสร็จสิ้นการกระบวนการปรับรูปแบบบรรทัดฐานในระดับที่ 3 แล้ว จะได้รีเลชันทั้งหมดดังนี้

การสั่ง(<u>เลขที่ใบสั่ง</u>, วันที่สั่ง, รหัสลูกค้า)
ลูกค้า(<u>รหัสลูกค้า</u>, ชื่อลูกค้า)
สินค้า(<u>รหัสสินค้า</u>, ชื่อสินค้า)
รายการสั่ง(<u>เลขที่ใบสั่ง</u>, <u>รหัสสินค้า</u>, จำนวน)

วิธีการวิจัย

ในงานวิจัยนี้จะนำหลักการของการปรับรูปแบบบรรทัดฐานและอัลกอริทึมเอไพรออริ มาพัฒนาต่อเป็นอัลกอริทึม NoWARs (<u>No</u>rmalization <u>W</u>ith <u>A</u>ssociation <u>Rule</u>s) โดยอัลกอริทึมจะรับ

ข้อมูลเข้าเป็นตารางข้อมูลและให้ผลลัพธ์เป็น Relation schema รายละเอียดของอัลกอริทึม NoWARs ดังแสดงในรูปที่ 5

Algorithm: NoWARs

//Input: Data

//Output: Relation schema

Call Apriori

for every rule from i to n **do**

1. Select one cause rule with max support value
    - 1.1 Select primary key of relation (2NF)
    - 1.2 Find corresponding attribute
        - 1.2.1 Select primary key of relation (3NF)
    - 1.3 Compare non key attribute in (1.2.1) with all attributoes in (1) then cut it off and we get one 3NF relation
2. Compare all non key attributes with 1NF relation and cut it off
3. Combine all primary keys of 2NF with the remaining attribute in step 3 to become the last 3NF

รูปที่ 5 แสดงอัลกอริทึม NoWARs

อัลกอริทึม NoWARs จะเริ่มทำงานจากรับข้อมูลเข้าเป็นตารางข้อมูลเข้ามา แล้วหาความสัมพันธ์ของข้อมูลจากตารางนั้นด้วยอัลกอริทึมเอไพรออริ ซึ่งกฎความสัมพันธ์จะถูกคัดมาเฉพาะกฎที่มีเพียงหนึ่งเหตุ (1 แอททริบิวท์) ส่วนผลที่ตามมานั้นจะมีกี่แอททริบิวท์ก็ได้ ในขั้นตอนแรกจะเป็นการหาตารางที่อยู่ในรูปแบบบรรทัดฐานระดับที่ 2 ให้ได้ก่อน ซึ่งเราสามารถหาตารางนี้ได้จากกฎความสัมพันธ์ที่มีค่าสนับสนุนสูงที่สุด และส่วนที่เป็นเหตุสามารถระบุส่วนที่เป็นผลที่ตามมาได้มากที่สุด ยกตัวอย่าง เช่น OrderNO=171 5 ==> custName=Niyom ID=2 Date=24-12-2008 5 conf:(1) ซึ่งกฎดังกล่าวสามารถแสดงให้อยู่ในรูป Relational Schema ได้ดังนี้ POrder (ID, custName, OderNO, Date) หลังจากนั้นจึงทำการหาตารางที่อยู่ในรูปแบบบรรทัดฐานระดับที่ 3

จากตารางที่อยู่รูปแบบบรรทัดฐานระดับที่ 2 อีกที ซึ่งสามารถหาตารางที่อยู่ในรูปแบบบรรทัดฐานระดับที่ 3 ได้จากกฎที่มีลักษณะดังนี้ ID=2 5 ==> custName=Niyom 5 conf:(1) และสามารถแสดงให้อยู่ในรูป Relational Schema ได้ดังนี้ Customer (ID, custName)

จากตาราง Customer จะเห็นว่าแอททริบิวท์ที่ไม่ใช่คีย์หลัก คือ แอททริบิวท์ custName เมื่อตัดแอททริบิวท์ custName ออกจากตาราง POrder จะได้ตารางที่อยู่ในรูปแบบบรรทัดฐานระดับที่ 3 คือ Order (ID, OrdrNO, Date) ลำดับต่อมาจะเป็นการหาตารางที่อยู่ในรูปแบบบรรทัดฐานระดับที่ 2 ตารางต่อไป ซึ่งสามารถหาได้จากกฎที่ว่า ProID=4 4 ==> productDesc=Green Acrylic Size=2 UPrice=3800 4 conf:(1) ซึ่งกฎดังกล่าวสามารถแสดงให้อยู่ในรูป Relational Schema ได้ดังนี้ PreProduct (ProID, productDesc, Size, UPrice) และจากกฎ Size=2 4 ==> UPrice=3800 4 conf:(1) จะได้ Relational Schema คือ Price (Size, UPrice) และเมื่อตัดแอททริบิวท์ UPrice ที่ไม่ได้เป็นคีย์ของตาราง Product ออกจากตาราง PreProduct จะได้ตารางที่อยู่ในรูปแบบบรรทัดฐานระดับที่ 3 คือ Product (ProID, producDesc, Size)

ขั้นตอนต่อไปเป็นการตัดแอททริบิวท์ที่ไม่ใช่คีย์ทั้งหมดออกจากตารางที่เป็นข้อมูลเข้า ซึ่งมี Schema ดังนี้ 1NFTB (Size, ProID, productDesc, ID, custName, UPrice, Qty, OrderNO, Date) จะเห็นว่ายังคงแอททริบิวท์ที่ยังเหลืออยู่ คือ แอททริบิวท์ Qty และเมื่อนำแอททริบิวท์ Qty ไปรวมกับแอททริบิวท์ที่เป็นคีย์ของตารางที่อยู่ในรูปแบบบรรทัดฐานระดับที่ 2 ทั้งสองตาราง จะได้ตารางที่อยู่ในรูปแบบบรรทัดฐานระดับที่ 3 ตารางสุดท้าย คือ Detail (ProID, OrderNO, Qty)

ผลการวิจัย

การทดสอบระบบจะทำการทดสอบบนเครื่องคอมพิวเตอร์ที่ใช้ซีพียู Pentium IV ความเร็ว 3.0 GHz แรมขนาด 512 MB โดยใช้ระบบฐานข้อมูล Oracle 10g ข้อมูลที่ใช้ในการทดสอบเป็นข้อมูลที่ได้จากการสังเคราะห์ขึ้นมาเอง และตัวอย่างของข้อมูลบางส่วนมีสคีมาดังนี้

Orders (Pid, Pname, Cust_ID, Cust_name, Brand, Warranty, Qty, INV, DATE)

ผลจากการทดสอบแสดงได้ดังนี้

จากกฎความสัมพันธ์ขั้นต้นที่ได้จากการรัน

Inv=A60 4 ==> Cust_ID=A4 Cust_name=Yinda Date=19-01-2008 4 conf:(1)

ซึ่งจากกฎความสัมพันธ์นี้เราจะได้รีเลชัน 2NF ออกมา 1 รีเลชัน คือ

Order (Inv, Cust_ID, Cust_name, Date)

และได้รีเลชัน 3NF ออกมา 1 รีเลชัน คือ

Customer (Cust_ID, Cust_name)

จากกฎความสัมพันธ์

Pid=503 3 ==> Pname=Cool-W Brand=Whirlpool Warranty=5 3 conf: (1)

ซึ่งจากกฎความสัมพันธ์นี้เราจะได้รีเลชัน 2NF ออกมา 1 รีเลชัน คือ

Product (Pid, Pname, Brand, Warranty)

และได้รีเลชัน 3NF ออกมา 1 รีเลชัน คือ

Brand (Brand, Warranty)

เมื่อรันโปรแกรมเสร็จสิ้นแล้วจะได้รีเลชัน 3NF ทั้งหมดดังนี้

Customer (Cust_ID, Cust_name)

Order (Cust_ID, Inv, Date)

Brand (Brand, Warranty)

Product (Pid, Pname, Brand)

Detail (Pid, Inv, Qty)

ผลการทดลองดังที่กล่าวมามีข้อมูลบางส่วน ที่ใช้สำหรับการทดสอบดังนี้

ตารางที่ 3 ตาราง Orders

Pid	**Pname**	**Cust_ID**	**Cust_name**	**Brand**	**Warranty**	**Qty**	**Inv**	**Date**
506	Wash-W	A4	Yinda	Whirlpool	5	1	A60	19-02-2008
503	Cool-W	A4	Yinda	Whirlpool	5	3	A60	19-02-2008
501	TV-Sh	A3	Saijai	Sharp	3	1	A59	20-01-2008
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

ผลลัพธ์ที่ได้จากการรัน แสดงตารางที่ 4-8 ดังนี้

ตารางที่ 4 ตาราง Customer

Cust_ID	**Cust_name**
A4	Yinda
A3	Saijai
⋮	⋮

ตารางที่ 5 ตาราง Order

Cust_ID	**Inv**	**Date**
A4	Yinda	19-02-2008
A3	Saijai	20-01-2008
⋮	⋮	

ตารางที่ 6 ตาราง Brand

Brand	Warranty
Whirlpool	5
Sharp	3
⋮	⋮

ตารางที่ 7 ตาราง Product

Pid	Pname
506	Wash-W
503	Cool-W
⋮	⋮

ตารางที่ 8 ตาราง Detail

Pid	Inv	Qty
506	A60	1
501	A59	1
⋮	⋮	

สรุปและอภิปรายผลการวิจัย

งานวิจัยนี้มีจุดมุ่งหมายที่จะพัฒนาแนวทางในการปรับรูปแบบบรรทัดฐานตารางข้อมูล เพื่อให้ได้ตารางสำหรับเก็บข้อมูลที่มีประสิทธิภาพ ลดปัญหาความซ้ำซ้อนของข้อมูล โดยใช้เทคนิคการหาความสัมพันธ์ของข้อมูลเข้ามาช่วย ซึ่งผลการทดสอบสรุปได้ดังนี้

ในการใช้โปรแกรม NoWARs เพื่อทำการปรับรูปแบบบรรทัดฐาน โดยการเลือกเฉพาะกฎที่มีค่าความเชื่อมั่น 100% และค่าสนับสนุนอยู่ในช่วง 0-40% ปรากฏว่าผลที่ได้จากการทดสอบด้วยโปรแกรม NoWARs กับผลที่ได้จากการที่ผู้ใช้กระทำการปรับรูปแบบบรรทัดฐานเองให้ผลที่เหมือนกัน นั่นคือ ผลที่ได้จากโปรแกรม NoWARs มีความถูกต้อง 100 เปอร์เซ็นต์ แต่การปรับรูปแบบบรรทัดฐานให้อยู่ในขั้นที่มากกว่านี้โปรแกรมไม่สามารถทำได้ ซึ่งถือว่าเป็นข้อจำกัดของโปรแกรมและเป็นแนวทางสำหรับการพัฒนาต่อไปในอนาคต

กิตติกรรมประกาศ

งานวิจัยนี้ได้รับทุนสนับสนุนจากมหาวิทยาลัยเทคโนโลยีสุรนารีผ่านหน่วยวิจัยด้านวิศวกรรมข้อมูลและการค้นหาความรู้

บรรณานุกรม

Argawal R. and Srikant R. (1994). Fast algorithm for mining association rules. ***Proceedings of the 20th International Conference on Very Large Data Bases Conference***.

Argawal R., Imielinski T., and Swami A. (1993). Mining association rules between set of items in large database. ***Proceedings of ACM SIGMOD International Conference on Management of Data***.

Codd E. F. A relational model of data for large shared data banks. (1970). ***Communications of the ACM****, 16(6)*, 377-387.

Date C. J. and Fagin R. (1992). Simple conditions for guaranteeing higher normal forms in relational databases. ***ACM Transactions on Database Systems (TODS)****, 17(3)*, 465-476.

Sarawagi S., Thomas S., and Agrawal R. (1998). Integrating association rule mining with relational database. ***Proc. of the ACM SIGMOD Conference on Management of Data***, 343-354.

ภาคผนวก ข

รหัสต้นฉบับของโปรแกรม NoWARs

```
CREATE OR REPLACE PROCEDURE THESIS.NoWARs (p_table varchar2, p_disp_rules
varchar2 DEFAULT 'N', p_dlmtr varchar2 DEFAULT',') IS
  v_tab_col        varchar2(4000); -- store column in table to normalize
  v_table          varchar2(30)   := upper(p_table);
  v_tot_tran       number(10)     := 0; -- total of transaction
BEGIN
  prepare_data (v_table, p_dlmtr, v_tab_col, p_disp_rules, v_tot_tran);
  find_column_nf;
  gen_all_nf (p_dlmtr);
  get_nf (p_dlmtr);
  find_3nf_from_result (p_dlmtr);
  find_3nf_from_2nf (v_tab_col, p_dlmtr);
  disp_all_complete (p_dlmtr, p_table, v_tab_col, v_tot_tran);
END NoWARs;
/

CREATE OR REPLACE PROCEDURE THESIS.find_column_nf IS
  TYPE c_refcur IS REF CURSOR;
  v_sl_det         varchar2(4000) ;
  v_hrule_if       varchar2(50);-- for store head's key
  v_drule_the      varchar2(30000);-- for store dets key use in first candidate
  d_dummy          c_refcur; -- select detail use in first candidate

  /** variable for find col **/
  v_seq            number(10)  := 1;
  v_status         varchar2(5)  := 'TRUE';
  v_last           number(10)  := 0;
  v_first          number(10)  := 0;
  v_round          number(10)  := 1;
  v_dlmtr          varchar2(1)  := '=';
  v_result         varchar2(5000);

BEGIN
  OPEN d_dummy FOR 'select substr(rule_if,1,instr(rule_if,"=")-1), rule_then '||
                   'from assoc_rule '||
                   'where instr(rule_if,",") = 0 '||
                   'and rule_if_cnt > 1';
  LOOP --** rule then
    FETCH d_dummy INTO v_hrule_if, v_drule_then;
    INTO WHEN d_dummy%NOTFOUND;
    BEGIN
      -- initial variable
      v_seq          := 1;
      v_status       := 'TRUE';
      v_last         := 0;
      v_first        := 0;
      -- get column for count
      WHILE v_status = 'TRUE' LOOP
          IF    v_seq = 1 THEN
                v_first := 1;
                v_last := instr(v_drule_then, v_dlmtr, 1, v_seq)-1;
          ELSE
            v_first   := instr(v_drule_then, ',', 1, v_seq-1) + 1;
            IF instr(v_drule_then, ',', 1, v_seq) > 0 THEN
              v_last := (instr(v_drule_then, v_dlmtr, 1, v_seq) - instr(v_drule_then, ',', 1, v_seq-1))-1;
```

```
            ELSE
              v_last     := instr(v_drule_then, v_dlmtr, -1, 1) - v_first;
              v_status := 'FALSE';
            END IF;
          END IF;
          v_seq     := v_seq + 1;
          v_result  := substr(v_drule_then, v_first, v_last);
          -- count column
          BEGIN
           UPDATE column_nf
           SET    cnt_if   = nvl(cnt_if, 0) + 1,
                  cnt_then = nvl(cnt_then, 0) + 1
           WHERE  column_if   = v_hrule_if
           AND    column_then = v_result;
           IF SQL%NOTFOUND THEN
             INSERT INTO column_nf(column_if, column_then, cnt_if, cnt_then)
             VALUES (v_hrule_if, v_result, 1, 1);
           END IF;
          END;
          COMMIT;
      END LOOP;
    END;
  END LOOP;
  CLOSE d_dummy;
  EXCEPTION WHENOTHER THEN DBMS_OUTPUT.PUT_LINE (SQLERRM);
          IF d_dummy%isopen THEN
            CLOSE d_dummy;
          END IF;
END find_column_nf;
/

CREATE OR REPLACE PROCEDURE THESIS.prepare_data (p_table varchar2, p_dlmtr
varchar2, p_tab_col IN OUT varchar2, p_disp_rules varchar2, p_tot_tran IN OUT number) IS
--  p_sel_col varchar2(100) := 'COLUMN_NAME';
--  p_table varchar2(30) := 'TEST1';
--  p_dlmtr varchar2(1) := ',';
  p_cnt_status      varchar2(1)       := 'N';
  v_stop_status     varchar2(5)       := 'FALSE';
  v_can_seq         number(10)        := 1;
  v_status          varchar2(10)      := 'TRUE';
  v_freq_seq        varchar2(50);
  v_tot_tran        number(10)        := 0; -- total of transaction
  p_min_sup         number(10)        := 1;
  p_min_conf        number(10)        := 100;
  v_first           number(3)         := 0;
  v_pos             number(3)         := 0;
  v_col             varchar2(4000);
  v_col_sl          varchar2(4000);
  v_chk             varchar2(4000);
  v_result          varchar2(4000);
  v_result2         varchar2(4000);
  TYPE c_refcur IS REF CURSOR;
  p_slct VARCHAR2(4000):= 'select column_name from user_tab_columns where table_name =
''||p_table||'';
  p_test            varchar2(4000);
```

```
  lc_str          VARCHAR2(4000);
  lc_colval       VARCHAR2(4000);
  c_dummy         c_refcur;
  cnt_dummy       c_refcur;
  c_dummy2        c_refcur;
  v_cnt           number := 0;
  l               number;

BEGIN
  DELETE FROM can_set;
  DELETE FROM freq_set;
  DELETE FROM assoc_rule;
  DELETE FROM column_nf;
  DELETE FROM result_nf;
  COMMIT;
  BEGIN
    OPEN cnt_dummy FOR 'select count(*) from '||p_table;
    FETCH cnt_dummy INTO v_tot_tran;
    CLOSE cnt_dummy;
    -- exception handler
    EXCEPTION WHENothers THEN DBMS_OUTPUT.PUT_LINE (SQLERRM);
           IF cnt_dummy%isopen THEN
              CLOSE cnt_dummy;
           END IF;
  END;

  p_tot_tran := v_tot_tran; -- add on 12-03-2009 use in display all for calculate % of max and min of
support rules to use in process
  -- open cursor for select column_name from user_tab_columns
  OPEN c_dummy FOR p_slct;
  BEGIN
    LOOP
       FETCH c_dummy INTO lc_colval;
       INTO WHEN c_dummy%NOTFOUND;
       lc_str    := lc_str || '||'||""||p_dlmtr|| ""|| '||'||""|| lc_colval||'='|| ""|| '||'|| lc_colval;
       p_tab_col := p_tab_col || p_dlmtr || lc_colval;
       v_cnt     := v_cnt + 1;
    END LOOP;
    CLOSE c_dummy;
    v_col := substr(lc_str,8);
    -- exception handler
    EXCEPTION WHENothers THEN DBMS_OUTPUT.PUT_LINE (SQLERRM);
           IF c_dummy%isopen THEN
              CLOSE c_dummy;
           END IF;
  END;

  WHILE v_status = 'TRUE' LOOP
    IF v_can_seq != 1 THEN
      BEGIN
        -- open cursor for select data from assign table
        OPEN c_dummy2 FOR 'select '||v_col||' from '||p_table;
        LOOP
           FETCH c_dummy2 INTO v_result2;
           INTO WHEN c_dummy2%NOTFOUND;
```

```
        item.find_candidate (v_can_seq, v_status, v_result2, p_min_sup);
      END LOOP;
      CLOSE c_dummy2;
      -- exception handler
      EXCEPTION WHENothers THEN DBMS_OUTPUT.PUT_LINE (SQLERRM);
            IF c_dummy%isopen THEN
              CLOSE c_dummy;
            END IF;
    END;
  ELSE
    BEGIN
      -- open cursor for select data from assign table
      OPEN c_dummy2 FOR 'select '||v_col||' from '||p_table;
      LOOP
        FETCH c_dummy2 INTO v_result;
        INTO WHEN c_dummy2%NOTFOUND;
          v_pos   := 0;
          v_first := 0;
          v_stop_status:= 'FALSE';
          WHILE v_stop_status = 'FALSE' LOOP
            IF instr(v_result, ',', v_first+1) <= 0 THEN
              v_col_sl := substr(v_result, v_first+1);
              v_stop_status := 'TRUE';
            ELSE
              v_pos := instr(v_result, ',', v_first+1);
              v_col_sl := substr(v_result, v_first+1, (v_pos-1)-v_first);
              v_first := v_pos;
            END IF;
            item.gen_candidate (v_can_seq, v_col_sl); -- generate first candidate
          END LOOP; -- for generate first candidate
      END LOOP;
      CLOSE c_dummy2;
      -- exception handler
      EXCEPTION WHENothers THEN DBMS_OUTPUT.PUT_LINE (SQLERRM);
            IF c_dummy2%isopen THEN
              CLOSE c_dummy2;
            END IF;
      DBMS_OUTPUT.PUT_LINE (v_result);
    END;
  END IF; -- end chk candidate sequence
  -- generate frequence item set
  item.gen_frequence (v_can_seq, v_tot_tran, p_min_sup);
  v_can_seq := v_can_seq + 1; -- plus candidate sequence
  COMMIT;
 END LOOP;  -- end v_status loop
 p_tab_col := substr(p_tab_col,2);
 /** generate association rules **/
 rule.gen_rules (p_min_conf, p_disp_rules);
END prepare_data;
/

CREATE OR REPLACE PROCEDURE THESIS.gen_all_nf (p_dlmtr varchar2) IS
 TYPE c_refcur IS REF CURSOR;
 v_hcol_if          varchar2(30);
 v_hcnt_if          number(5);
```

```
  v_dcol_then       varchar2(30);
  v_nf              varchar2(4000);
  h_dummy           c_refcur; -- for select head
  d_dummy           c_refcur; -- select detail use in first candidate
BEGIN
  OPEN h_dummy FOR 'select column_if, max(cnt_if) '||
                   'from   column_nf '||
                   'group by column_if '||
                   'order by column_if';
  LOOP --** column_if
    v_nf := NULL;
    FETCH h_dummy INTO v_hcol_if, v_hcnt_if;
    INTO WHEN h_dummy%NOTFOUND;
    BEGIN
      -- find column_then
      OPEN d_dummy FOR 'select column_then '||
                       'from   column_nf '||
                       'where  column_if = '''||v_hcol_if||''' '||
                       'and    cnt_if = '||v_hcnt_if||
                       ' order by column_then';
      LOOP
        FETCH d_dummy INTO v_dcol_then;
        INTO WHEN d_dummy%NOTFOUND;
        v_nf := v_nf||p_dlmtr||v_dcol_then;
      END LOOP;
      CLOSE d_dummy;
      EXCEPTION WHENothers THEN DBMS_OUTPUT.PUT_LINE (SQLERRM);
              IF d_dummy%isopen THEN
                CLOSE d_dummy;
              END IF;
    END;
    -- insert result_nf
    BEGIN
      INSERT INTO result_nf (tab_column, cnt_if)
      VALUES (v_hcol_if||v_nf, v_hcnt_if);
      EXCEPTION WHENdup_val_on_index THEN NULL;
    END;
    COMMIT;
    -- show nf
  END LOOP;
  CLOSE h_dummy;
  EXCEPTION WHENothers THEN DBMS_OUTPUT.PUT_LINE (SQLERRM);
          IF h_dummy%isopen THEN
            CLOSE h_dummy;
          END IF;
END gen_all_nf;
/

CREATE OR REPLACE PROCEDURE THESIS.get_real_nf (p_tab_col varchar2, p_cnt number,
p_dlmtr varchar2) IS  -- called by get_nf p_tab_col, p_cnt send from get_nf
  TYPE c_refcur IS REF CURSOR;
  h_dummy           c_refcur; -- for select head
  d_dummy           c_refcur; -- select detail use in first candidate
  v_ptab_col        varchar2(4000);
  v_htab_col        varchar2(4000);
```

```
  v_tab_col         varchar2(4000);
  v_hcnt_if         number(5);
  v_col             varchar2(30);
  v_status          varchar2(5)         := 'TRUE';
  v_tab_col         varchar2(4000);
  v_fpos            number              :=1; -- first position for substring
  v_bpos            number              :=0; -- last position for substrinb
  v_pos             number              :=1; -- position for calculate fpos and lpost
  v_seq             number              :=1; -- sequential for find next delimiter
BEGIN
  -- find column_then
  BEGIN
    OPEN d_dummy FOR 'select tab_column, cnt_if '||
                     'from   result_nf '||
                     'where  length(tab_column) = '||length(p_tab_col)||
                     'and    tab_column != '''||p_tab_col||''' '||
                     'and    use_to_nf = ''N''';
    LOOP -- fetch data
      FETCH d_dummy INTO v_htab_col, v_hcnt_if;
      INTO WHEN d_dummy%NOTFOUND;
      -- check all column in p_tab_col are match in v_htab_col
      LOOP -- for get column
        -- find position of delimiter
        v_pos := instr(p_tab_col, p_dlmtr, 1, v_seq);
        v_seq := v_seq + 1;
        -- set back position for substring
        IF v_pos = 0 THEN
          v_bpos := length(p_tab_col) - v_bpos + 2;
        ELSE
          v_bpos := v_pos - v_fpos;
        END IF;
        v_col := substr(p_tab_col, v_fpos, v_bpos);
        -- check column are match
        IF instr(v_htab_col, v_col) = 0 THEN
          v_status := 'FALSE';
          INTO;
        ELSE
          v_status := 'TRUE';
        END IF;
        -- check column are match
        -- exit when end of data
        INTO WHEN v_pos = 0;
        v_fpos := v_pos + 1;
      END LOOP;
      -- check support count
      IF v_hcnt_if > p_cnt AND v_status = 'TRUE' THEN
        v_ptab_col := v_htab_col;
      ELSE
        IF v_status = 'TRUE' THEN
          BEGIN
            UPDATE result_nf
            SET    use_to_nf  = 'D'
            WHERE  tab_column = v_htab_col;
            IF SQL%NOTFOUND THEN
              NULL;
```

```
            END IF;
            END;
            COMMIT;
        END IF;
        v_ptab_col := p_tab_col;
      END IF;
      -- check support count
    END LOOP;
    CLOSE d_dummy;
    EXCEPTION WHENothers THEN DBMS_OUTPUT.PUT_LINE (SQLERRM);
            IF d_dummy%isopen THEN
              CLOSE d_dummy;
            END IF;    -- update result
  END;
  BEGIN
    UPDATE result_nf
    SET    use_to_nf  = 'Y'
    WHERE  tab_column = v_ptab_col;
    IF SQL%NOTFOUND THEN
       NULL;
    END IF;
  END;
  COMMIT;
END get_real_nf;
/

CREATE OR REPLACE PROCEDURE THESIS.get_nf (p_dlmtr varchar2) IS
  TYPE c_refcur IS REF CURSOR;
  v_dlmtr          varchar2(3) := ',';
  v_cnt_dlmtr      number(5)   := 0;
  v_htab_col       varchar2(4000);
  v_hcnt_if        number(5);
  v_dcol_then      varchar2(30);
  v_nf             varchar2(4000);
  h_dummy          c_refcur; -- for select head
  d_dummy          c_refcur; -- select detail use in first candidate
BEGIN
  BEGIN
    OPEN h_dummy FOR 'select tab_column,cnt_if '||
                     'from   result_nf '||
                     'order by length(tab_column) asc, cnt_if desc, '||
                     'decode(instr(tab_column, "ID"), 0, decode(instr(tab_column, "NO"), 0,
instr(tab_column, "CODE"), instr(tab_column, "NO")), instr(tab_column, "ID"))';
    LOOP --** column_if
      v_nf := NULL;
      FETCH h_dummy INTO v_htab_col, v_hcnt_if;
      INTO WHEN h_dummy%NOTFOUND;
      -- count delimited in tab_column
      get_real_nf (v_htab_col, v_hcnt_if, p_dlmtr);
    END LOOP;
    CLOSE h_dummy;
    EXCEPTION WHENothers THEN DBMS_OUTPUT.PUT_LINE (SQLERRM);
            IF h_dummy%isopen THEN
              CLOSE h_dummy;
            END IF;
```

```
  END;
  -- set use_to_nf for rules are remain
END get_nf;
/

CREATE OR REPLACE PROCEDURE THESIS.find_3nf_from_result (p_dlmtr varchar2) IS
  TYPE c_refcur IS REF CURSOR;
  v_htab_col        varchar2(4000);
  v_dtab_col        varchar2(4000);
  v_ttab_col        varchar2(4000); -- temp v_htab_col for check is same record
  v_tab_col         varchar2(4000);
  v_cnt_fcol        number(7);
  v_dcnt_col        number(7);
  v_chk_3nf         varchar2(1)         := 'Y'; -- for check rule complete to 3nf
  v_tchk_3nf        varchar2(1)         := 'Y'; -- for temp check rule complete to 3nf
  h_dummy           c_refcur; -- for select head
  d_dummy           c_refcur; -- for select head
  v_col             varchar2(30);
  v_tcol            varchar2(30); -- temp for v_col
  v_fpos            number              :=1; -- first position for substring
  v_bpos            number              :=0; -- last position for substrinb
  v_pos             number              :=1; -- position for calculate fpos and lpost
  v_seq             number              :=1; -- sequential for find next delimiter
  v_pk              varchar2(3000); -- store primary key
  v_dpk             varchar2(3000); -- store primary key
BEGIN
  BEGIN
    OPEN h_dummy FOR 'select tab_column '||
                     'from   result_nf '||
                     'where  use_to_nf = "Y" '||
                     'order by cnt_if';
    LOOP --** column_if
      -- fine 3NF
      FETCH h_dummy INTO v_htab_col;
      INTO WHEN h_dummy%NOTFOUND;
      -- initial status to 'y'
      v_chk_3nf := 'Y';
      -- check qty that first_column from tab_column are in other tab_column
      BEGIN
        SELECT count(tab_column)
        INTO   v_cnt_fcol
        FROM   result_nf
        WHERE  use_to_nf  = 'Y'
        AND    tab_column <> v_htab_col
        AND    instr(tab_column, substr(v_htab_col, instr(v_htab_col, p_dlmtr, 1, 1))) > 0;
        EXCEPTION WHENno_data_found THEN v_cnt_fcol := 0;
      END;
      -- update nf3 to define that tab_column is 3NF
      -- find pk in 3NF
      v_pk   := NULL;
      v_fpos :=1; -- first position for substring
      v_bpos :=0; -- last position for substrinb
      v_pos  :=1; -- position for calculate fpos and lpost
      v_seq  :=1; -- sequential for find next delimiter
      BEGIN
```

```
      LOOP
        -- find position of delimiter
        v_pos := instr(v_htab_col, ',', 1, v_seq);
        v_seq := v_seq + 1;
        -- set back position for substring
        IF v_pos = 0 THEN
          v_bpos := length(v_htab_col) - v_bpos + 2;
        ELSE
          v_bpos := v_pos - v_fpos;
        END IF;
        v_col := substr(v_htab_col, v_fpos, v_bpos);
        -- check rule complete to 3nf
         IF nvl(v_cnt_fcol, 0) = 0 AND v_chk_3nf = 'Y' OR
          (instr(v_tcol, 'CODE') > 0 or instr(v_tcol, 'ID') > 0 or instr(v_tcol, 'NO') > 0) THEN
          BEGIN
            SELECT 'N'
            INTO   v_chk_3nf
            FROM   result_nf
            WHERE  instr(tab_column, v_col) > 0
            AND    use_to_nf = 'Y'
            AND    length(tab_column) < length(v_htab_col);
            EXCEPTION WHENno_data_found THEN v_chk_3nf := 'Y';
          END;
        END IF;
        -- check column are match to be primary key
        IF v_chk_3nf = 'Y' THEN
          IF v_tchk_3nf = 'N' AND (instr(v_tcol, 'CODE') > 0 OR instr(v_tcol, 'ID') > 0 OR
instr(v_tcol, 'NO') > 0) THEN
            v_pk := v_tcol;
            v_tchk_3nf := v_chk_3nf;
          ELSE
            IF instr(v_col, 'CODE') > 0 OR instr(v_col, 'ID') > 0 OR instr(v_col, 'NO') > 0  THEN
              v_pk := v_col;
            END IF;
          END IF;
        ELSE
          v_pk      := NULL;
          v_tchk_3nf := v_chk_3nf;
        END IF;
        v_tcol     := v_col;

        -- exit when end of data
        INTO WHEN v_pos = 0;
        v_fpos := v_pos + 1;
      END LOOP;
    END;

    -- update status to 3nf
    BEGIN
      UPDATE result_nf
      SET    nf3    = v_chk_3nf,
             tab_pk = v_pk
      WHERE  tab_column = v_htab_col;
      IF SQL%NOTFOUND THEN
        NULL;
```

```
    END IF;
   END;
   COMMIT;
  END LOOP;
  CLOSE h_dummy;
  EXCEPTION WHENothers THEN DBMS_OUTPUT.PUT_LINE (SQLERRM);
          IF h_dummy%isopen THEN
            CLOSE h_dummy;
          END IF;
END;

BEGIN
  -- fine 3NF
  -- to cut primary key from 3NF from 2NF
  OPEN h_dummy FOR 'select tab_column, tab_pk '||
                   'from   result_nf '||
                   'where  use_to_nf = "Y" '||
                   'and    NF3 = "Y"';
  LOOP --** column_if
   -- fine 3NF
   FETCH h_dummy INTO v_htab_col, v_pk;
   INTO WHEN h_dummy%NOTFOUND;
   -- check qty that first_column from tab_column are in other tab_column
   v_dpk      := NULL;
   v_dtab_col := NULL;
   v_dcnt_col := 0;
   BEGIN
    OPEN d_dummy FOR 'select tab_column, cnt_if '||
                     'from   result_nf '||
                     'where  use_to_nf = "Y" '||
                     'and    NF3 = "N" '||
                     'and    instr(tab_column, '"'||v_pk||'"') > 0';
    LOOP
     FETCH d_dummy INTO v_dtab_col, v_dcnt_col;
     -- if v_htab_col is 3nf and not found in detail then v_htab_col is 2nf
     IF v_dtab_col is NULL THEN
       BEGIN
        UPDATE result_nf
        SET    nf2 = 'Y'
        WHERE  tab_column = v_htab_col;
        IF SQL%NOTFOUND THEN
          NULL;
        END IF;
       END;
     END IF;
     INTO WHEN d_dummy%NOTFOUND;

     v_tab_col   := v_dtab_col;
     v_col       := NULL;
     v_fpos      :=1; -- first position for substring
     v_bpos      :=0; -- last position for substrinb
     v_pos       :=1; -- position for calculate fpos and lpost
     v_seq       :=1; -- sequential for find next delimiter
     BEGIN
      LOOP
```

```
          -- find position of delimiter
          v_pos := instr(v_tab_col, ',', 1, v_seq);
          v_seq := v_seq + 1;
          -- set back position for substring
          IF v_pos = 0 THEN
            v_bpos := length(v_tab_col) - v_bpos + 2;
          ELSE
            v_bpos := v_pos - v_fpos;
          END IF;
          v_col := substr(v_tab_col, v_fpos, v_bpos);
          -- cut col from 3nf
          IF instr(v_htab_col, v_col) > 0 AND v_col != v_pk THEN
            IF instr(v_dtab_col, p_dlmtr||v_col) > 0 THEN
              v_dtab_col := REPLACE(v_dtab_col, p_dlmtr||v_col);
            elsIF instr(v_dtab_col, v_col||p_dlmtr) > 0 THEN
              v_dtab_col := REPLACE(v_dtab_col, v_col||p_dlmtr);
            END IF;
          END IF;
          -- check column are match to be primary key
          IF v_col <> v_pk AND (instr(v_col, 'CODE') > 0 or instr(v_col, 'ID') > 0 or instr(v_col,
'NO') > 0)  THEN
            v_dpk := v_col;
          END IF;
          -- exit when end of data
          INTO WHEN v_pos = 0;
          v_fpos := v_pos + 1;
        END LOOP;
        v_tab_col := v_dtab_col;
      END;
      -- insert new 3NF
      BEGIN
        INSERT INTO result_nf(tab_column, cnt_if, use_to_nf, nf3, nf2, tab_pk)
        VALUES (v_tab_col, v_dcnt_col, 'Y', 'Y', 'Y', v_dpk);
        EXCEPTION WHENdup_val_on_index THEN NULL;
      END;
      COMMIT;
    END LOOP;
    CLOSE d_dummy;
    EXCEPTION WHENothers THEN DBMS_OUTPUT.PUT_LINE (SQLERRM);
          IF d_dummy%isopen THEN
            CLOSE d_dummy;
          END IF;
  END;
  -- update nf3 to define that tab_column is 3NF
  -- find pk in 3NF
 END LOOP;
 CLOSE h_dummy;
 EXCEPTION WHENothers THEN DBMS_OUTPUT.PUT_LINE (SQLERRM);
        IF h_dummy%isopen THEN
          CLOSE h_dummy;
        END IF;
 -- fine 3NF
 -- to cut primary key from 3NF from 2NF
END;
END find_3nf_from_result;  /
```

```
CREATE OR REPLACE PROCEDURE THESIS.find_3nf_from_2nf (p_tab_col varchar2, p_dlmtr
varchar2) IS
  TYPE c_refcur IS REF CURSOR;
  v_hcol          varchar2(30);
  v_dcol          varchar2(30);
  v_dnf2          varchar2(1);
  v_dtab_pk       varchar2(4000);
  v_htab_col      varchar2(4000)     := p_tab_col;
  v_dtab_col      varchar2(4000);
  v_tab_col       varchar2(4000);
  v_cnt_fcol      number(7);
  v_dcnt_col      number(7);
  h_dummy         c_refcur;          -- for select head
  d_dummy         c_refcur;          -- for select head
  v_col           varchar2(30);
  v_cfpos         number             :=1; -- first position dtab_col for name is dup
  v_cbpos         number             :=0; -- last position dtab_col for name is dup
  v_cpos          number              :=1; -- position for calculate fpos and lpost
  v_cseq          number             :=1; -- sequential for find next delimiter for dup
  v_chcol         varchar2(30);       -- check dup column
  v_fpos          number             :=1; -- first position for substring
  v_bpos          number             :=0; -- last position for substrinb
  v_pos           number             :=1; -- position for calculate fpos and lpost
  v_seq           number             :=1; -- sequential for find next delimiter
  v_pk            varchar2(3000);    -- store primary key
  v_dpk           varchar2(3000);    -- store primary key
BEGIN
  -- compare all non attribute key with 1NF and cut it off
  BEGIN
    -- open d_dummy for get all 3NF for fine
    OPEN d_dummy FOR 'select tab_column, tab_pk, nf2 '||
                     'from   result_nf '||
                     'where  nf3 = "Y" '||
                     'and    use_to_nf = "Y"';
    LOOP
      -- fine 3NF
      FETCH d_dummy INTO v_dtab_col, v_dtab_pk, v_dnf2;
      INTO WHEN d_dummy%NOTFOUND;
      -- update nf3 to define that tab_column is 3NF
      -- initial value
      v_pk   := NULL;
      v_fpos :=1; -- first position for substring
      v_bpos :=0; -- last position for substrinb
      v_pos  :=1; -- position for calculate fpos and lpost
      v_seq  :=1; -- sequential for find next delimiter
      BEGIN
        -- cut off primary key in 3NF
        IF instr(v_dtab_col, p_dlmtr||v_dtab_pk) > 0 THEN
          v_dtab_col := REPLACE(v_dtab_col, p_dlmtr||v_dtab_pk);
        elsIF instr(v_dtab_col, v_dtab_pk||p_dlmtr) > 0 THEN
          v_dtab_col := REPLACE(v_dtab_col, v_dtab_pk||p_dlmtr);
        END IF;
        -- cut off non primary key attribute from 1NF
        LOOP
          -- find position of delimiter
```

```
    v_pos := instr(v_dtab_col, p_dlmtr, 1, v_seq);
    v_seq := v_seq + 1;
    -- set back position for substring
    IF v_pos = 0 THEN
      v_bpos := length(v_dtab_col) - v_bpos + 2;
    ELSE
      v_bpos := v_pos - v_fpos;
    END IF;
    v_dcol := substr(v_dtab_col, v_fpos, v_bpos);

    -- initial value for check duplicate column name
    v_cfpos :=1; -- first position for substring
    v_cbpos :=0; -- last position for substrinb
    v_cpos  :=1; -- position for calculate fpos and lpost
    v_cseq  :=1; -- sequential for find next delimiter
    -- check duplicate column name
    LOOP
     -- find only found v_dcol in v_htab_col
     IF instr(v_htab_col, v_dcol) > 0 THEN
       v_cfpos := instr(v_htab_col, ',', instr(v_htab_col, v_dcol, 1, v_cseq)-1, 1)+1;
       v_cpos  := instr(v_htab_col, ',', instr(v_htab_col, v_dcol, 1, v_cseq), 1);
       v_cseq := v_cseq + 1;
       IF v_cpos = 0 THEN
         v_cbpos := length(v_htab_col) - v_cfpos + 1;
       ELSE
         v_cbpos := v_cpos - v_cfpos;
       END IF;
       -- cut column correct column from v_htab_col
       v_chcol := substr(v_htab_col, v_cfpos, v_cbpos);
       -- cut off non primary key attribute from in 1NF
       IF instr(v_htab_col, p_dlmtr||v_chcol) > 0 THEN
         v_htab_col := REPLACE(v_htab_col, p_dlmtr||v_chcol);
       elsIF instr(v_htab_col, v_chcol||p_dlmtr) > 0 THEN
         v_htab_col := REPLACE(v_htab_col, v_chcol||p_dlmtr);
       END IF;
     ELSE
       v_cpos := 0;
     END IF;
     INTO WHEN v_cpos = 0;
    END LOOP;
    -- check column are match to be primary key
    IF v_dnf2 = 'Y' AND nvl(instr(v_dpk, v_dtab_pk), 0) = 0  THEN
      v_dpk := v_dpk||p_dlmtr||v_dtab_pk;
    END IF;
    -- exit when end of data
    INTO WHEN v_pos = 0;
    v_fpos := v_pos + 1;
   END LOOP;
  end;
END LOOP;
CLOSE d_dummy;
-- trim delimited
v_dpk := substr(v_dpk, 2);
EXCEPTION WHENothers THEN DBMS_OUTPUT.PUT_LINE (SQLERRM);
        IF d_dummy%isopen THEN
```

```
            CLOSE d_dummy;
          END IF;
  END;
  -- insert new 3NF from 1NF
  BEGIN
    INSERT INTO result_nf(tab_column, cnt_if, use_to_nf, nf3, nf2, nf1, tab_pk)
    VALUES (v_htab_col, 0, 'Y', 'Y', 'Y', 'Y', v_dpk);
    EXCEPTION WHENdup_val_on_index THEN NULL;
  END;
  COMMIT;
END find_3nf_from_2nf;
/

CREATE OR REPLACE PROCEDURE THESIS.disp_all_complete (p_dlmtr varchar2, p_table
varchar2, p_tab_col varchar2, p_tot_tran number) IS
  TYPE c_refcur IS REF CURSOR;
  h_dummy         c_refcur;            -- for select head
  v_use_rule      number(10)       := 0;
  v_all_rule      number(10)       := 0;
  v_min_usup      number(10)       := 0;
  v_max_usup      number(10)       := 0;
  v_tab_pk        varchar2(4000);
  v_tab_col       varchar2(4000);
  v_seq           number           := 1; -- first position for substring
BEGIN
  -- get all rules and rules used
  BEGIN
    SELECT sum(decode(instr(rule_if,','), 0, decode(rule_cnt, 1, 0, 1), 0)) use_rule, count(rule_if)
all_rule
    INTO   v_use_rule, v_all_rule
    from   assoc_rule;
    EXCEPTION WHENno_data_found THEN
          v_use_rule := 0;
          v_all_rule := 0;
  END;
  -- get min and max support from rules used
  BEGIN
    SELECT round((max(sup_cnt)*100)/p_tot_tran,2) mxsup,
round((min(sup_cnt)*100)/p_tot_tran,2) mnsup
    INTO   v_max_usup, v_min_usup
    from   can_set
    WHERE  can_seq = 'C1'
    AND    sup_cnt > 1;
    EXCEPTION WHENno_data_found THEN
          v_max_usup := 0;
          v_min_usup := 0;
  END;

  DBMS_OUTPUT.PUT_LINE ('Original table  :  '||p_table||'  ('||p_tab_col||')');
  DBMS_OUTPUT.PUT_LINE ('');
  DBMS_OUTPUT.PUT_LINE ('All Association Rules Found :  '||v_all_rule||'  Rules');
  DBMS_OUTPUT.PUT_LINE ('All Association Rules Used :  '||v_use_rule||'  Rules, (Max.
Support :  '||v_max_usup||'% , Min. Support :  '||v_min_usup||'% )');
  DBMS_OUTPUT.PUT_LINE ('');
```

```
 BEGIN
   OPEN h_dummy FOR 'select tab_column, tab_pk from result_nf where nf3 = "Y" and
instr(tab_column, "'||p_dlmtr||'") > 0';
   -- for display all 3NF
   LOOP
    FETCH h_dummy INTO v_tab_col, v_tab_pk;
    INTO WHEN h_dummy%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE ('3NF Table '||v_seq||' ('||v_tab_col||') Primary Key : '||v_tab_pk);
    v_seq := v_seq+1;
   END LOOP;
   EXCEPTION WHENothers THEN DBMS_OUTPUT.PUT_LINE (SQLERRM);
          IF h_dummy%isopen THEN
            CLOSE h_dummy;
          END IF;
 END;
END disp_all_complete;
/

CREATE OR REPLACE PACKAGE THESIS.item IS
 FUNCTION chk_match (p_status varchar2, p_chk_item varchar2, p_tran_item varchar2)
RETURN varchar2;
 FUNCTION find_chk_pos(p_seq number, p_set varchar2) RETURN varchar2;
 PROCEDURE find_candidate (p_can_seq IN OUT number, p_status IN OUT varchar2,
                    p_tran_item varchar2, p_min_sup number);
 PROCEDURE gen_candidate (p_can_seq number, p_item_set varchar2);
 PROCEDURE gen_frequence (p_freq_seq number, p_tot_rec number, p_min_sup number);
END;
/

CREATE OR REPLACE PACKAGE BODY THESIS.item IS
 FUNCTION chk_match (p_status varchar2, p_chk_item varchar2, p_tran_item varchar2)
RETURN varchar2 is
 /* for chk item_set are in transaction */
 BEGIN
   IF instr(p_tran_item||',', p_chk_item) > 0 AND p_status = 'TRUE' THEN /*Modified on 23-11-
2008 add ||',' after p_tran_item*/
     RETURN 'TRUE';
   ELSE
     RETURN 'FALSE';
   END IF;
 END chk_match;
 --
 FUNCTION find_chk_pos(p_seq number, p_set varchar2) RETURN varchar2 is
 /* find item for chk in transaction */
   v_last  number(10);
   v_first number(10);
 BEGIN
  IF p_seq = 1 THEN
    v_first := 1;
    v_last := instr(p_set, ',', 1, p_seq)-1;
  ELSE
    v_first   := instr(p_set, ',', 1, p_seq-1) + 1;
    IF instr(p_set, ',', 1, p_seq) > 0 THEN
      v_last := (instr(p_set, ',', 1, p_seq) - instr(p_set, ',', 1, p_seq - 1))-1;
    ELSE
```

```
        v_last := length(p_set) - instr(p_set, ',', -1, 1);
      END IF;
    END IF;
    RETURN substr(p_set, v_first, v_last)||',';/*Modified on 23-11-2008 add ||',' after substr(p_set,
v_first, v_last)*/
  END find_chk_pos;
--
  PROCEDURE find_candidate (p_can_seq IN OUT number, p_status IN OUT varchar2,
                            p_tran_item varchar2, p_min_sup number)IS

    CURSOR set_l is
            SELECT item_set, substr(item_set, (instr(item_set, ',', -1, 1)+1)) l_set
            from   freq_set
            WHERE  freq_seq = 'F'||(p_can_seq-1)
            order by l_set;

    CURSOR set_r (p_lset varchar2) is
            SELECT item_set r_set
            from   freq_set
            WHERE  freq_seq = 'F1'
            AND    p_lset < item_set
            order by r_set;

    v_chk      varchar2(4000);
    v_status   varchar2(10);
    v_tmp      varchar2(32767);
  BEGIN
    p_status := 'FALSE';
    FOR l in set_l LOOP
      FOR r in set_r (l.l_set) LOOP
        v_status := 'TRUE'; -- initial v_status for chk member
        FOR i in 1.. p_can_seq LOOP -- for chk member of set
          v_chk    := item.find_chk_pos (i, l.item_set||','||r.r_set);
          v_status := item.chk_match (v_status, v_chk, p_tran_item);
          p_status := 'TRUE';
        END LOOP; -- end loop chk member
        IF v_status = 'TRUE' THEN -- chk item_set match with transaction
          -- update support in candidate
          item.gen_candidate (p_can_seq, l.item_set||','||r.r_set);
        END IF; -- end if chk status
      END LOOP;
    END LOOP;
--    commit;
  END find_candidate;
  --
  PROCEDURE gen_candidate (p_can_seq number, p_item_set varchar2) IS
  BEGIN
    UPDATE can_set
    SET    sup_cnt = nvl(sup_cnt, 0)+1
    WHERE  item_set = p_item_set;
    IF SQL%NOTFOUND THEN
      BEGIN
        INSERT INTO can_set
        VALUES ('C'||p_can_seq, p_item_set, 1);
      END;
```

```
  END IF;
 END gen_candidate;
 --
 PROCEDURE gen_frequence (p_freq_seq number, p_tot_rec number, p_min_sup number) is
 BEGIN
  BEGIN
   INSERT INTO freq_set
   SELECT 'F'||to_char(p_freq_seq), item_set, sup_cnt
   from   can_set
   WHERE  can_seq = 'C'||to_char(p_freq_seq)
   AND    (sup_cnt*100)/p_tot_rec >= p_min_sup;
  END;
 END gen_frequence;
 --
END item;
/

CREATE OR REPLACE PACKAGE THESIS.rule AS
 FUNCTION find_chk_pos (p_seq number, p_set varchar2, p_status IN OUT varchar2) RETURN
varchar2;
 FUNCTION find_sup_cnt (p_item varchar2) RETURN number;
 PROCEDURE gen_sub_rule (p_start number, p_freq_seq number, p_set varchar2, p_front
varchar2, p_back varchar2,
                  p_min_conf number, p_status IN OUT varchar2, p_rule_seq IN OUT number,
p_disp_rules varchar2);
 PROCEDURE gen_rules (p_min_conf number, p_disp_rules varchar2);
 PROCEDURE ins_rules (p_iset varchar2, p_rule_seq number, p_rule_if varchar2, p_rule_then
varchar2, p_min_conf number, p_disp_rules varchar2);
END;

CREATE OR REPLACE PACKAGE BODY THESIS.rule AS
 FUNCTION find_chk_pos (p_seq number, p_set varchar2, p_status IN OUT varchar2) RETURN
varchar2 IS
   v_fpos number(10)   := 0;
   v_bpos number(10)   := 0;
 BEGIN
   IF p_seq = 1 THEN
     v_fpos := 1;
   ELSE
     v_fpos := instr(p_set, ',', 1, p_seq - 1) + 1;
   END IF;

   IF instr(p_set, ',', 1, p_seq) > 0 THEN
     v_bpos   := instr(p_set, ',', 1, p_seq) - v_fpos;
     p_status := 'TRUE';
   ELSE
     v_bpos   := length (p_set) - instr(p_set, ',', 1, p_seq);
     p_status := 'FALSE';
   END IF;
   RETURN substr(p_set, v_fpos, v_bpos);
RETURN NULL;

 END find_chk_pos;
```

```
 FUNCTION find_sup_cnt (p_item varchar2) RETURN number is
  v_cnt number(10);
 BEGIN
  SELECT sup_cnt
  INTO   v_cnt
  from   can_set
  WHERE  item_set = p_item;
  RETURN v_cnt;
  EXCEPTION WHENno_data_found THEN RETURN 1;
 END find_sup_cnt;
 --
 PROCEDURE gen_sub_rule (p_start number, p_freq_seq number, p_set varchar2, p_front
varchar2, p_back varchar2, p_min_conf number, p_status IN OUT varchar2, p_rule_seq IN OUT
number, p_disp_rules varchar2) IS
  v_status varchar2(5) := 'TRUE';
  v_front  varchar2(4000) ;
  v_back   varchar2(4000) := '^';
  v_fcnt   number(10); -- front set's sup_cnt
  v_icnt   number(10); -- item_set's sup_cnt
  v_conf   number(10,2);
 BEGIN
  FOR j in (p_start + 1) .. (p_freq_seq) LOOP
    IF p_start > p_freq_seq THEN
      INTO;
    END IF;
    IF j > (p_start + 1) THEN
      rule.gen_sub_rule (j-1, p_freq_seq, p_set, v_front, v_back, p_min_conf, p_status, p_rule_seq,
p_disp_rules);
    END IF;

    v_front := rule.find_chk_pos (j, p_set, v_status);

    IF v_front != REPLACE(p_back, ','||v_front) THEN
      IF v_status = 'TRUE' THEN
        v_back  := REPLACE(p_back, v_front||',');
      ELSE
        v_back  := REPLACE(p_back, ','||v_front);
      END IF;
    ELSE
      INTO;
    END IF;

    v_front := p_front||','||v_front;
    v_fcnt  := rule.find_sup_cnt(v_front);
    v_icnt  := rule.find_sup_cnt(p_set);
    v_conf  := round(v_icnt/v_fcnt, 2);
    --
    -- display set and confidence
    --
      p_rule_seq := p_rule_seq + 1;
      rule.ins_rules (p_set, p_rule_seq, v_front, v_back, (p_min_conf/100), p_disp_rules);
  END LOOP;
 END gen_sub_rule;
 --
```

```
 PROCEDURE gen_rules (p_min_conf number, p_disp_rules varchar2) IS
  CURSOR freq is SELECT freq_seq, item_set, sup_cnt
            FROM   freq_set WHERE  freq_seq > 'F1' ORDER BY freq_seq;
  v_front          varchar2(4000);
  v_back           varchar2(4000);
  v_status         varchar2(10) := 'TRUE';
  v_icnt           number(10); -- item_set's sup_cnt
  v_fcnt           number(10); -- front set's sup_cnt
  v_conf           number(10,2);
  v_freq_seq       number(10) := 0;
  v_rule_seq       number(10) := 0;
 BEGIN
  FOR f IN freq LOOP
    v_freq_seq := to_number(REPLACE(f.freq_seq, 'F', ''));
    FOR i IN 1..v_freq_seq LOOP
      v_front := rule.find_chk_pos (i, f.item_set, v_status);
      IF v_status = 'TRUE' THEN
        v_back  := REPLACE(f.item_set, v_front||',');
      ELSE
        v_back  := REPLACE(f.item_set, ','||v_front);
      END IF;
      v_fcnt   := rule.find_sup_cnt(v_front);
      v_icnt   := rule.find_sup_cnt(f.item_set);
      v_conf   := round(v_icnt/v_fcnt, 2);
      v_rule_seq := v_rule_seq + 1;
      rule.ins_rules (f.item_set, v_rule_seq, v_front, v_back, (p_min_conf/100), p_disp_rules);
      rule.gen_sub_rule (i, v_freq_seq, f.item_set, v_front, v_back, p_min_conf, v_status,
v_rule_seq, p_disp_rules);
    END LOOP;
  END LOOP;
 END gen_rules; /
 PROCEDURE ins_rules (p_iset varchar2, p_rule_seq number, p_rule_if varchar2, p_rule_then
varchar2, p_min_conf number, p_disp_rules varchar2) IS
  v_if_cnt         number := 0;
  v_cnt            number := 0;
  v_conf           number := 0;
 BEGIN
  v_if_cnt         := rule.find_sup_cnt(p_rule_if);
  v_cnt            := rule.find_sup_cnt(p_iset);
  v_conf           := round(nvl(v_cnt, 0)/nvl(v_if_cnt, 0), 2);
  IF v_conf >= nvl(p_min_conf, 0) THEN
    -- check display rules
    IF p_disp_rules = 'Y' THEN
      DBMS_OUTPUT.PUT_LINE (p_rule_seq||' : '||p_rule_if||' = '||to_char(v_if_cnt)||
      ' ->   '||p_rule_then||' = '|| to_char(v_cnt)||' conf : '||to_char (v_conf));
    END IF;
    BEGIN
     INSERT INTO assoc_rule (rule_conf, rule_if_cnt, rule_cnt, rule_if, rule_then)
     VALUES (v_conf, v_if_cnt, v_cnt, p_rule_if, p_rule_then);
    END;
    COMMIT;
  END IF;
 END ins_rules;
END;  /
```

ประวัติผู้เขียน

นายณัฐพล พันนุรัตน์ เกิดเมื่อวันที่ 31 สิงหาคม พ.ศ. 2527 ที่ อำเภอสนม จังหวัดสุรินทร์ เริ่มเข้าศึกษาระดับชั้นอนุบาล 1 ถึงชั้นประถมศึกษาปีที่ 5 ที่โรงเรียนบ้านโนนเปือย ตำบลโพนโก อำเภอสนม จังหวัดสุรินทร์ หลังจากนั้นได้ย้ายไปศึกษาในระดับชั้นประถมศึกษาปีที่ 6 ที่โรงเรียนสังขะวิทยาคม อำเภอสังขะ จังหวัดสุรินทร์ จากนั้นได้เข้าศึกษาต่อในระดับมัธยมศึกษาตอนต้นและตอนปลาย ที่โรงเรียนสุรวิทยาคาร อำเภอเมือง จังหวัดสุรินทร์ ปีการศึกษา 2546 ได้เข้าศึกษาต่อระดับปริญญาตรีในสาขาวิชาวิศวกรรมคอมพิวเตอร์ สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี และสำเร็จการศึกษาเมื่อปี พ.ศ. 2549 ภายหลังสำเร็จการศึกษาในระดับปริญญาตรี ได้เข้าศึกษาต่อในระดับปริญญาโท สาขาวิชาวิศวกรรมคอมพิวเตอร์ สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี ในปีการศึกษา 2550

ในระหว่างการศึกษาได้รับความอนุเคราะห์อย่างดียิ่งจากคณาจารย์ในสาขาวิชา และได้รับความไว้วางใจให้เป็นผู้ช่วยสอนปฏิบัติการจำนวน 3 รายวิชา คือ (1) Computer Programming (2) Event-Driven Programming และ (3) Database System เป็นเวลา 2 ปี