การปรับปรุงประสิทธิภาพข้อคำถามเชิงความหมาย ด้วยการอุปนัยกฎความสัมพันธ์

นายอภิชัย ฤทธิ์ธงชัยเลิศ

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์
มหาวิทยาลัยเทคโนโลยีสุรนารี
ปีการศึกษา 2550

SEMANTIC QUERY OPTIMIZATION WITH ASSOCIATION RULE INDUCTION

Apichai Ritthongchailert

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 2007

การปรับปรุงประสิทธิภาพข้อคำถามเชิงความหมาย
ด้วยการอุปนัยกฎความสัมพันธ์

มหาวิทยาลัยเทคโนโลยีสุรนารี อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

คณะกรรมการสอบวิทยานิพนธ์

(ผศ. ดร.พิชโยทัย มหัทธนาภิวัฒน์)
ประธานกรรมการ

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

(ผศ. ดร.คะชา ชาญศิลป์)
กรรมการ

(ศ. ดร.ไพโรจน์ สัตยธรรม)
รองอธิการบดีฝ่ายวิชาการ

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

อภิชัย ฤทธิ์ธงชัยเลิศ : การปรับปรุงประสิทธิภาพข้อคำถามเชิงความหมายด้วยการอุปนัยกฎความสัมพันธ์ (SEMANTIC QUERY OPTIMIZATION WITH ASSOCIATION RULE INDUCTION) อาจารย์ที่ปรึกษา : รศ. ดร.นิตยา เกิดประสพ, 122 หน้า.

การปรับปรุงประสิทธิภาพข้อคำถามเชิงความหมาย หมายถึง การนำข้อคำถามเดิมมาจัดรูปแบบใหม่ให้มีรูปประโยคที่แตกต่างกับข้อคำถามเดิม แต่ยังคงให้ผลลัพธ์ที่เหมือนเดิมสิ่งที่แตกต่างกันของทั้งสองข้อคำถาม คือ เวลาที่ใช้ในการประมวลผลเพื่อตอบข้อคำถามนั้นจะใช้เวลาน้อยลงกว่าเดิม ความสมบูรณ์ของการปรับปรุงข้อคำถามเชิงความหมายนี้จะขึ้นอยู่กับเงื่อนไขหรือกฎข้อบังคับที่จะนำมาเพิ่มหรือลดตัวประโยคเงื่อนไขของข้อคำถาม โดยทั่วไปแล้วกฎข้อบังคับที่นำมาใช้ในการปรับปรุงข้อคำถามเชิงความหมายนี้จะได้มาจากผู้ดูแลระบบจัดการฐานข้อมูล ซึ่งอาจจะไม่ครอบคลุมกับข้อมูลทั้งหมดที่มีอยู่ในฐานข้อมูล ดังนั้น ในงานวิจัยนี้จึงนำเอาเทคโนโลยีการขุดค้นความรู้จากฐานข้อมูลหรือการทำเหมืองข้อมูล ซึ่งเป็นเทคโนโลยีที่เป็นที่รู้จักกันอย่างแพร่หลาย โดยนำมาเฉพาะส่วนของการค้นหากฎความสัมพันธ์ของข้อมูลมาประยุกต์ใช้เพื่อทำงานร่วมกับการปรับปรุงประสิทธิภาพข้อคำถามเชิงความหมายเพื่อใช้ลดเวลาในการประมวลผลข้อคำถาม

สาขาวิชาวิศวกรรมคอมพิวเตอร์ ลายมือชื่อนักศึกษา_____

ปีการศึกษา 2550 ลายมือชื่ออาจารย์ที่ปรึกษา_____

ลายมือชื่ออาจารย์ที่ปรึกษาร่วม_____

APICHAI RITTHONGCHAILERT : SEMANTIC QUERY OPTIMIZATION WITH ASSOCIATION RULE INDUCTION. THESIS ADVISOR : ASSOC. PROF. NITTAYA KERDPRASOP, Ph.D., 122 PP.

SEMANTIC QUERY OPTIMIZATION/ASSOCIATION RULE INDUCTION

Semantic query optimization is the process of transforming a given query into a semantically equivalent one that still returns the same answer for any database state satisfying query's constraints. The difference of both queries is lower execution cost of the transformed one. The efficiency of semantic query optimization depends on semantic constraints or integrity constraints which are used to remove a useless condition in a where clause of the given query. Basically, integrity constraints are defined by database developer. It may not cover all constraints in the database. Therefore, this paper aims at presenting the utilization of a well known data mining technique, association mining, to assist the semantic query optimization process.

School of Computer Engineering

Academic Year 2007

Student's Signature_____

Advisor's Signature_____

Co-advisor's Signature_____

กิตติกรรมประกาศ

วิทยานิพนธ์นี้สำเร็จลุล่วงด้วยดี ผู้วิจัยขอกราบขอบพระคุณ บุคคล และกลุ่มบุคคลต่าง ๆ ที่ได้กรุณาให้คำปรึกษา แนะนำ ช่วยเหลือ อย่างดียิ่ง ทั้งในด้านวิชาการ และด้านการดำเนินงานวิจัย ดังนี้

- รองศาสตราจารย์ ดร. นิตยา เกิดประสพ อาจารย์ที่ปรึกษาวิทยานิพนธ์
- รองศาสตราจารย์ ดร. กิตติศักดิ์ เกิดประสพ อาจารย์ที่ปรึกษาวิทยานิพนธ์ร่วม
- ผู้ช่วยศาสตราจารย์ ดร. พิชโยทัย มหัทธนาภิวัฒน์ ผู้ช่วยศาสตราจารย์ ดร.คะชา ชาญศิลป์ และผู้ช่วยศาสตราจารย์ สมพันธุ์ ชาญศิลป์ อาจารย์ประจำสาขาวิชาวิศวกรรมคอมพิวเตอร์ สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี
- คุณกัลญา พับโพธิ์ เลขานุการสาขาวิชาวิศวกรรมคอมพิวเตอร์ ที่ให้ความช่วยเหลือในการประสารงานด้านเอกสารต่าง ๆ ระหว่างศึกษา
- คุณจักรพันธ์ มหาวันตัง คุณวุฒิติพล หมัดเส็น คุณปฐมพงศ์ พันนุรัตน์ คุณชลดา พรหมสุข คุณณัฐพล พันนุรัตน์ และบัณฑิตศึกษาสาขาวิชาวิศวกรรมคอมพิวเตอร์ ทุกท่านที่ให้คำปรึกษาและช่วยเหลือด้วยดีมาโดยตลอด

นอกจากนี้ ขอขอบคุณครู อาจารย์ทั้งในอดีตและปัจจุบันที่ให้ความรู้แก่ผู้วิจัยจนประสบความสำเร็จในชีวิต

ท้ายที่สุด ขอกราบขอบพระคุณบิดา มารดา ที่ให้กำเนิด อุปการะเลี้ยงดูอบรม และส่งเสริมการศึกษาเป็นอย่างดีมาโดยตลอด ทำให้ผู้วิจัยมีความรู้ ความสามารถ มีจิตใจที่เข้มแข็งและช่วยเหลือตัวเองได้จนประสบความสำเร็จ

อภิชัย ฤทธิ์ธงชัยเลิศ

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

ตารางที่ **หน้า**

สารบัญตาราง (ต่อ)

ตารางที่ **หน้า**

สารบัญตาราง (ต่อ)

ตารางที่ **หน้า**

สารบัญตาราง (ต่อ)

ตารางที่ **หน้า**

สารบัญตาราง (ต่อ)

ตารางที่ **หน้า**

สารบัญตาราง (ต่อ)

ตารางที่ **หน้า**

สารบัญตาราง (ต่อ)

ตารางที่ **หน้า**

สารบัญตาราง (ต่อ)

ตารางที่ **หน้า**

สารบัญตาราง (ต่อ)

ตารางที่ **หน้า**

สารบัญรูป

รูปที่ **หน้า**

สารบัญรูป (ต่อ)

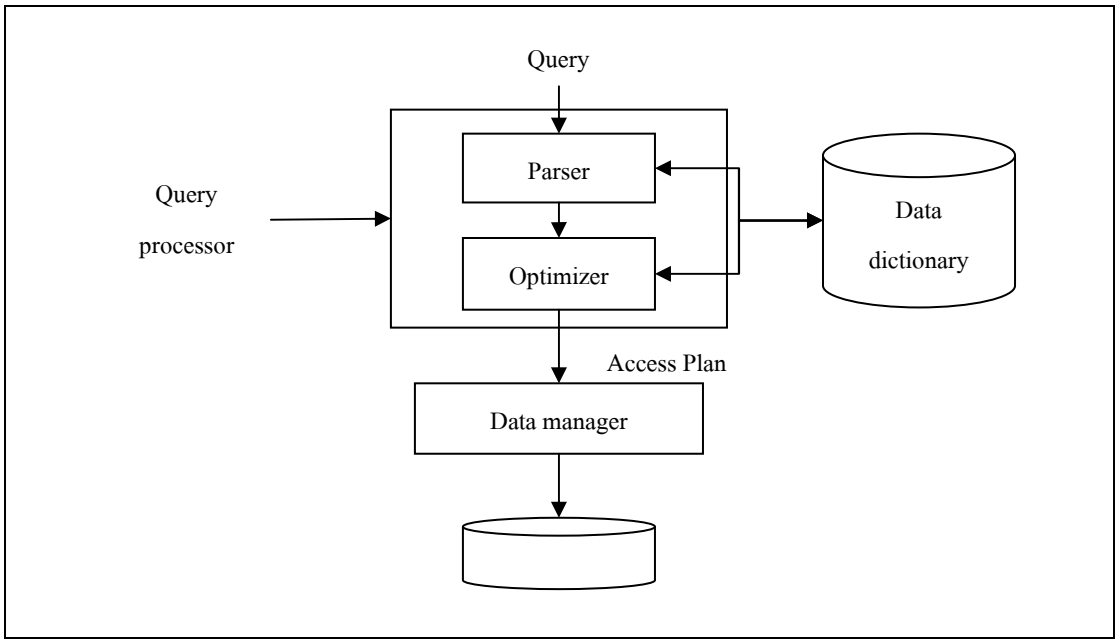
รูปที่ **หน้า**

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มาของปัญหา

ในปัจจุบัน องค์กรหรือสถาบันต่าง ๆ จำเป็นต้องมีการใช้ระบบจัดการฐานข้อมูล (Database Management System : DBMS) เพื่อจัดเก็บข้อมูลที่มีอยู่ภายในองค์กร ซึ่งบางองค์กรมีข้อมูลที่ถูกจัดเก็บอยู่ปริมาณมหาศาล และมีการเรียกใช้ข้อมูลภายในฐานข้อมูลจากผู้ใช้งาน (User) เป็นจำนวนมาก และอาจมีการเรียกใช้ข้อมูลนั้น ๆ ในเวลาเดียวกัน ในภาวะเช่นนี้จะทำให้เกิดความคับคั่งในการทำงานของระบบจัดการฐานข้อมูล อาจจะส่งผลให้ระบบฐานข้อมูลไม่สามารถตอบสนองการใช้งานจากผู้ใช้ได้ทันท่วงที ทำให้การทำงานในขั้นตอนอื่น ๆ เกิดความล่าช้าตามไปด้วย สาเหตุหนึ่งที่ทำให้ระบบจัดการฐานข้อมูลทำงานช้ากว่าปกติ คือ บางข้อคำถาม (Query) ที่ผู้ใช้ป้อนเข้ามาสู่ระบบจัดการฐานข้อมูล ไม่มีข้อมูลที่ผู้ใช้ต้องการปรากฏอยู่ หรือ เงื่อนไขภายในข้อคำถามอาจมีความซ้ำซ้อนกันเองภายในเงื่อนไขของข้อคำถามนั้น เมื่อระบบจัดการฐานข้อมูลรับข้อคำถามเหล่านี้เข้ามาประมวลผล จึงทำให้ใช้เวลาในการค้นหาข้อมูลที่ผู้ใช้ต้องการมากกว่าปกติ รวมทั้งมีการเรียกใช้ทรัพยากรของระบบอย่างไม่คุ้มค่า โดยเฉพาะอย่างยิ่งในกรณีที่ข้อคำถามนั้นไม่มีข้อมูลที่ต้องการอยู่ในฐานข้อมูล จึงทำให้ใช้เวลาในการประมวลผลข้อคำถามนั้นโดยสูญเปล่า และถ้าในฐานข้อมูลมีข้อมูลจำนวนมาก จะยิ่งส่งผลให้ใช้เวลาในการประมวลผลมากขึ้นไปด้วย แต่เราสามารถทำการตรวจสอบข้อคำถามเหล่านี้ได้ก่อนที่จะส่งข้อคำถามเข้าสู่ระบบจัดการฐานข้อมูล ซึ่งโดยทั่วไปแล้วระบบจัดการฐานข้อมูลจะมีส่วนประกอบที่สำคัญ คือ ส่วนประมวลผลข้อคำถาม (Query processor) เป็นส่วนที่ทำหน้าที่ประมวลผลข้อคำถามจากผู้ใช้ ตรวจสอบความถูกต้องของข้อคำถาม และแสดงข้อมูลที่เป็นคำตอบของข้อคำถามออกมาให้ผู้ใช้งานทราบ

ในระบบจัดการฐานข้อมูลขนาดใหญ่ เช่น Oracle, Microsoft SQL Server หรือ IBM DB2 ผู้พัฒนาระบบจัดการฐานข้อมูลเหล่านี้จะผนวกส่วนสำคัญอีกส่วนหนึ่งเข้าไป นั่นคือ ส่วนการเพิ่มประสิทธิภาพข้อคำถาม (Query optimizer) ซึ่งส่วนนี้มีหน้าที่ในการปรับปรุงรูปแบบข้อคำถาม (Query rewriting) ให้อยู่ในรูปแบบที่ดีขึ้น โดยให้คำตอบที่ดีที่สุด และถูกต้องตรงกับความต้องการให้แก่ผู้ใช้งานได้อย่างเหมาะสมในเวลาที่รวดเร็วกว่าเดิม รวมทั้งสามารถใช้ทรัพยากรของระบบให้มีประสิทธิภาพมากที่สุด ในส่วนนี้จะมีการเลือกแผน (Query plan) หรือกลยุทธ์ที่เหมาะสมในการสอบถามข้อมูลในฐานข้อมูล เพื่อให้ได้คำตอบที่ดีที่สุด ในเวลาที่รวดเร็วที่สุดให้แก่ผู้ใช้งาน โดยโครงสร้างของส่วนประมวลผลข้อคำถามมีรายละเอียดดังรูปที่ 1.1



รูปที่ 1.1 โครงสร้างส่วนประมวลผลข้อคำถาม

ในการประมวลผลข้อคำถามนี้ วิธีหนึ่งที่นิยมใช้ในการเพิ่มประสิทธิภาพ คือ การใช้ความรู้เกี่ยวกับสคีมาในลักษณะของ Semantic rule เพื่อนำกฎข้อบังคับเหล่านี้มาทำการแปลงรูปแบบข้อคำถามให้มีการประมวลผลที่รวดเร็วขึ้น หรือเรียกว่า การเพิ่มประสิทธิภาพข้อคำถามเชิงความหมาย (Semantic Query Optimization : SQO) ดังตัวอย่างจากสคีมาหรือโครงสร้างของข้อมูลเรือเดินสมุทร (ships) ประกอบกับความรู้เกี่ยวกับสคีมาที่กำหนดไว้ในลักษณะของ Semantic rules ช่วยให้สามารถแปลงข้อคำถาม Q_1 เป็นข้อคำถาม Q_1' ที่จะใช้เวลาประมวลผลหาคำตอบเร็วขึ้น

Schema : ships (Name, Type, Weight, Registry, Owner Name, Capacity)
index on Type

Semantic rule : IF ships.Weight > 200 THEN Ships.Type = 'Tanker'

Q_1 : SELECT *Name*
FROM *Ships*
WHERE *Weight > 250*

Q_1' : SELECT *Name*
FROM *Ships*
WHERE *Weight > 250*
AND *Type = 'Tanker'*

ซึ่งกฎข้อบังคับที่นำมาแปลงรูปแบบข้อคำถามจะได้มาจากผู้ดูแลระบบฐานข้อมูล ซึ่งเป็นผู้กำหนดข้อมูลกฎข้อบังคับแบบต่าง ๆ (Integrity constraints) โดยกฎเหล่านี้จะสอดคล้องกับข้อมูลที่มีอยู่ในฐานข้อมูล ในการเพิ่มประสิทธิภาพข้อคำถามเชิงความหมายนี้ จะยังคงให้ผลลัพธ์ที่เหมือนกันทั้งข้อคำถามเดิม และข้อคำถามที่มีการแปลงรูปแบบแล้ว แต่สิ่งที่แตกต่างกันของทั้งสองข้อคำถามคือเวลาที่ใช้ในการประมวลผล และค่าใช้จ่ายในการประมวลผลข้อคำถาม

ในการวิจัยนี้ มุ่งเน้นที่จะเสนอแนวความคิดในการค้นหากฎข้อบังคับที่จะนำมาสร้างเป็น Semantic rules ด้วยแนวทางใหม่ นั่นคือ การนำเทคโนโลยีทางด้านการทำเหมืองข้อมูลที่เริ่มเป็นที่รู้จักกันอย่างแพร่หลาย มาช่วยในการค้นหากฎข้อบังคับที่สามารถนำมาใช้ในกระบวนการเพิ่มประสิทธิภาพการประมวลผลข้อคำถาม การทำเหมืองข้อมูลเป็นการค้นหาความสัมพันธ์และรูปแบบทั้งหมด ที่มีอยู่จริงในฐานข้อมูล ความสัมพันธ์และรูปแบบเหล่านั้นได้ถูกซ่อนไว้ภายในข้อมูลจำนวนมากที่มีอยู่ การทำเหมืองข้อมูลจะทำการสำรวจและวิเคราะห์ข้อมูลให้อยู่ในรูปแบบที่เต็มไปด้วยความหมายและอยู่ในรูปของกฎ โดยความสัมพันธ์นี้จะแสดงให้เห็นถึงความรู้ต่าง ๆ ที่มีประโยชน์ในฐานข้อมูล ซึ่งเป็นความรู้ที่ถูกต้อง และสามารถนำไปใช้ได้ หรือเราจะเรียกการกระทำแบบนี้ว่า Knowledge Discovery in Databases หรือเรียกสั้น ๆ ว่า KDD (ในวิทยานิพนธ์เล่มนี้ใช้ชื่อ KDD ในความหมายเดียวกับคำว่า Data mining หรือการทำเหมืองข้อมูล) ในการวิจัยชิ้นนี้เราจะใช้เทคนิคการค้นหากฎความสัมพันธ์ (Association rules) เพื่อใช้ในการค้นหาความสัมพันธ์ของข้อมูลเพื่อนำมาสร้างเป็น Semantic rules โดยกฎที่ได้จากการทำเหมืองข้อมูลลักษณะนี้จะแสดงในรูปแบบที่สามารถเข้าใจได้ง่าย คือ สาเหตุไปสู่ผลลัพธ์ หรือ IF cause THEN effect

1.2 วัตถุประสงค์ของการวิจัย

1. เพื่อลดภาระการทำงานของระบบจัดการฐานข้อมูล โดยทำการตรวจสอบข้อคำถามของผู้ใช้งานที่ระดับ Front – End
2. เพื่อช่วยลดการจราจรที่คับคั่งของข้อคำถาม
3. เพื่อเพิ่มประสิทธิภาพการทำงานของแอพพลิเคชั่น โดยทำการลดเวลาการประมวลผลของข้อคำถาม
4. เพื่อทำการออกแบบรูปแบบโมเดลข้อมูล จากการทำเหมืองข้อมูลที่เหมาะสมกับงาน Query optimization
5. เพื่อพัฒนาวิธีการแปลงโมเดลข้อมูลให้เป็น Semantic rules

1.3 ขอบเขตของงานวิจัย

1. ศึกษาหลักการทำงานของการปรับปรุงประสิทธิภาพข้อคำถามเชิงความหมาย
2. ศึกษาวิธีการค้นหากฎความสัมพันธ์โดยใช้อัลกอริทึมเอไพรออรี
3. โปรแกรมที่ได้จากการพัฒนาขึ้นนี้ มุ่งเน้นเพื่อใช้ประโยชน์ในการทดสอบเวลาในการประมวลผลข้อถามเดิม กับข้อคำถามที่ถูกปรับปรุงแล้ว

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1. สามารถนำเทคโนโลยีที่มีประสิทธิภาพทั้งสองด้าน คือ การปรับปรุงประสิทธิภาพข้อคำถามเชิงความหมายและการทำเหมืองข้อมูล มาทำงานร่วมกันให้เกิดแนวทางใหม่เพื่อเป็นทางเลือกให้การพัฒนาระบบจัดการฐานข้อมูล
2. โปรแกรมที่สามารถรองรับการปรับปรุงประสิทธิภาพข้อคำถามเชิงความหมาย โดยใช้ความสัมพันธ์ของข้อมูลมาช่วยในการปรับปรุงข้อคำถาม
3. ความรู้ที่ได้จากงานวิจัยนี้ สามารถนำไปใช้เป็นแนวทางในด้านการพัฒนาระบบจัดการฐานข้อมูลให้มีประสิทธิภาพดีขึ้นต่อไปในอนาคต

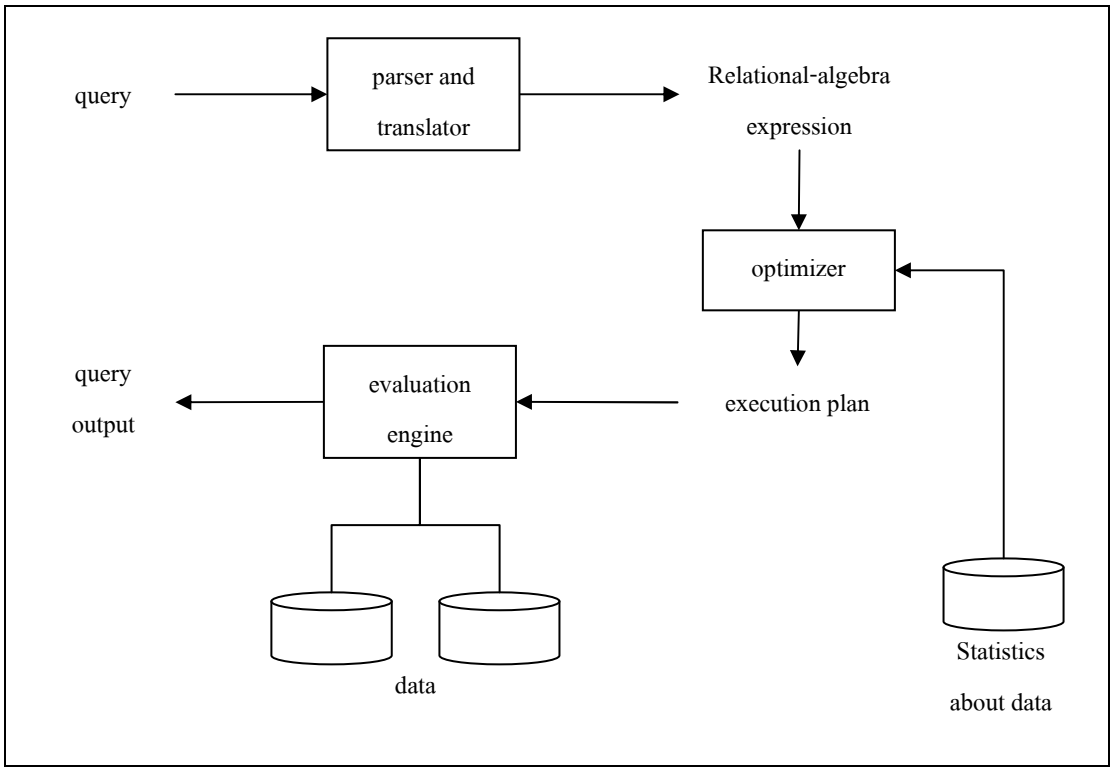
บทที่ 2
ปริทัศน์วรรณกรรมและงานวิจัยที่เกี่ยวข้อง

2.1 Query processor

Query processor เป็นส่วนสำคัญอีกส่วนหนึ่งที่อยู่ภายในระบบจัดการฐานข้อมูลในส่วน Query processor นี้จะเป็นส่วนช่วยให้ระบบจัดการฐานข้อมูลสามารถเข้าถึงข้อมูลที่อยู่ภายในฐานข้อมูลได้ง่าย และสะดวกมากขึ้นทำให้ผู้ใช้งานที่ต้องการทราบข้อมูลจากฐานข้อมูลได้รับข้อมูลตามเป้าหมายได้ง่าย และรวดเร็วขึ้นกว่าเดิมผู้ใช้งานระบบจัดการฐานข้อมูลไม่จำเป็นต้องทราบถึงกระบวนการที่ทำงานอยู่ภายในระบบจัดการฐานข้อมูล ดังนั้นเมื่อผู้ใช้งานต้องการสร้างโปรแกรมประยุกต์ขึ้นเพื่อใช้งานโดยมีการติดต่อเข้ากับฐานข้อมูล หรือต้องการสอบถามข้อมูลจากฐานข้อมูลโดยตรงเพื่อนำข้อมูลมาใช้งานภายในองค์กรหรือดำเนินธุรกิจต่าง ๆ จึงไม่จำเป็นต้องรับภาระในการจัดการข้อคำถามที่ต้องส่งเข้ามายังระบบจัดการฐานข้อมูล ดังนั้นจึงเป็นงานที่สำคัญของระบบจัดการฐานข้อมูลอีกส่วนหนึ่ง ซึ่งระบบจะทำการแปลงข้อคำถามที่รับเข้ามาจากผู้ใช้ให้อยู่ในรูปแบบที่ระบบจัดการฐานข้อมูลสามารถเข้าใจข้อคำถามได้ เพื่อนำข้อคำถามเหล่านั้นมาทำการประมวลผลข้อคำถามตามลำดับเครื่องหมายต่าง ๆ ที่อยู่ในข้อคำถามนั้น

ในส่วน Query processor มีส่วนประกอบพื้นฐานอยู่ด้วยกัน 3 ส่วนดังรูปที่ 2.1 คือ

- Parser ทำหน้าที่ในการตรวจสอบไวยากรณ์ข้อคำถามที่รับมา
- Translator ทำหน้าที่ในการแปลข้อคำถามที่รับเข้ามาให้อยู่ในแผนการประเมินค่า (Evaluation plan) ซึ่งจะอยู่ในภาษาระดับต่ำที่เครื่องสามารถเข้าใจได้ โดยปกติแล้วข้อคำถามหนึ่ง ๆ สามารถแปลให้อยู่ในรูปของแผนการประเมินค่าได้หลายทาง ซึ่งยังคงให้ผลลัพธ์ที่เหมือนกัน
- Optimizer ส่วนนี้จะทำหน้าที่ในการเลือกแผนการประเมินค่าที่มีค่าใช้จ่ายในการประมวลผลน้อยที่สุดจากทางเลือกทั้งหมดที่มีอยู่
- Query evaluation engine ทำหน้าที่ในการประมวลผลข้อคำถามที่ถูกส่งมาจากส่วน Optimizer



รูปที่ 2.1 ขั้นตอนในกระบวนการประมวลผลข้อคำถาม

ในการทำงานสำหรับลำดับแรกในส่วนของ Query processor คือ การแปลข้อคำถามที่รับมาจากผู้ใช้ระบบจำเป็นต้องแปลข้อคำถามเพื่อให้อยู่ในรูปแบบที่สามารถนำมาใช้งานได้ ภาษาที่ผู้ใช้เขียนข้อคำถามเข้ามานั้นส่วนใหญ่จะเขียนด้วนภาษา SQL (Structure Query Language) ซึ่งเป็นภาษาพื้นฐานที่ผู้ใช้สามารถเข้าใจได้ แต่ระบบไม่สามารถเข้าใจภาษาที่ผู้ใช้ป้อนเข้ามาได้ ซึ่งจะเป็นประโยชน์มากกว่า ถ้าข้อคำถามที่รับเข้ามาถูกแสดงด้วยรูปแบบนิพจน์พีชคณิตเชิงสัมพันธ์ หรือ Relational-algebra expression

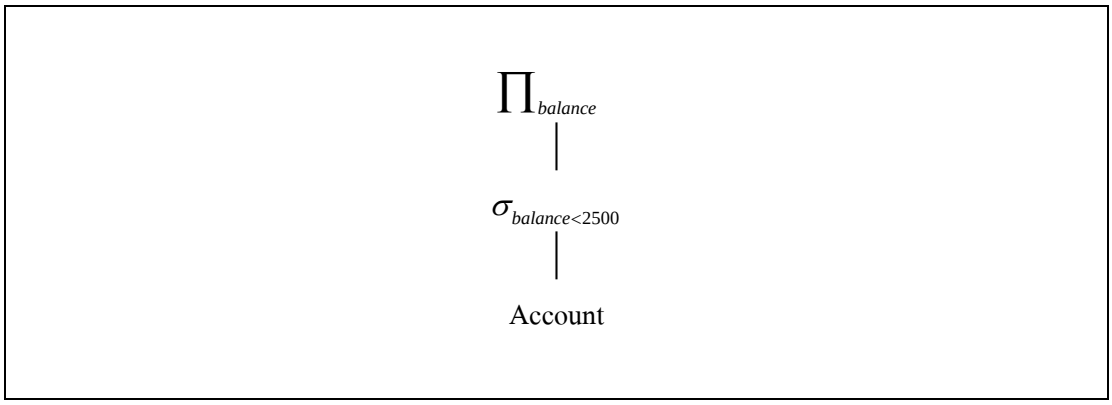
ข้อคำถามที่ส่งเข้ามานั้น โดยทั่วไปแล้วสามารถคำนวณหาคำตอบได้หลายวิธี ส่วนใหญ่เราสามารถเห็นข้อคำถามที่ป้อนเข้ามาอยู่ในรูปแบบของภาษา SQL ซึ่งในแต่ละข้อคำถามที่ถูกส่งเข้าสามารถแปลให้อยู่ในรูปแบบ Relational-algebra expression ได้หลายทาง ดังนั้น ระบบสามารถประเมินค่าใช้จ่ายของข้อคำถามได้จากรูปแบบที่เป็น Relational-algebra นี้ ถ้าเราพิจารณาข้อคำถามต่อไปนี้

```
SELECT    balance
FROM      account
WHERE     balance < 2500
```

สำหรับข้อคำถามข้างต้นนี้ เราสามารถทำการแปลให้อยู่ในรูปแบบของ Relational-algebra ได้ทั้งสองลักษณะคือ

$$\sigma_{balance<2500}(\prod_{balance}(account))$$
$$\prod_{balance}(\sigma_{balance<2500}(account))$$

มากกว่านั้นเราสามารถทำการประมวลผลแต่ละเครื่องหมายใน Relational-algebra ด้วยอัลกอริทึมที่แตกต่างกันได้ และลำดับค่าการประเมินค่าใช้จ่ายของเครื่องหมายภายในข้อคำถาม จะเรียกว่า Query-execution plan หรือ Query-evaluation plan ดังรูปต่อไปนี้



รูปที่ 2.2 Query-evaluation plan

จากนั้น Query-execution engine จะนำข้อคำถามที่ทำการแปลให้อยู่ในรูปแบบของ Query-evaluation plan มาทำการประมวลผลข้อคำถามนั้น และส่งคำตอบที่ถูกต้องกลับออกมาสู่ผู้ใช้งานระบบทราบ

ในข้อคำถามที่ส่งเข้ามา สามารถเขียนแทนด้วย Query-evaluation plan ได้หลายรูปแบบ ซึ่งในแต่ละรูปแบบที่ได้นั้นจะให้เวลาในการประมวลผลข้อคำถามที่แตกต่างด้วยเช่นกัน ดังนั้น ระบบจึงไม่คาดหวังว่าผู้ใช้งานจะเขียนข้อคำถาม โดยนึกถึงประสิทธิภาพในการประมวลผลข้อคำถามนั้น ๆ หรือเราสามารถพูดได้อีกแบบหนึ่งนั่นคือควรเป็นความรับผิดชอบของระบบจัดการ

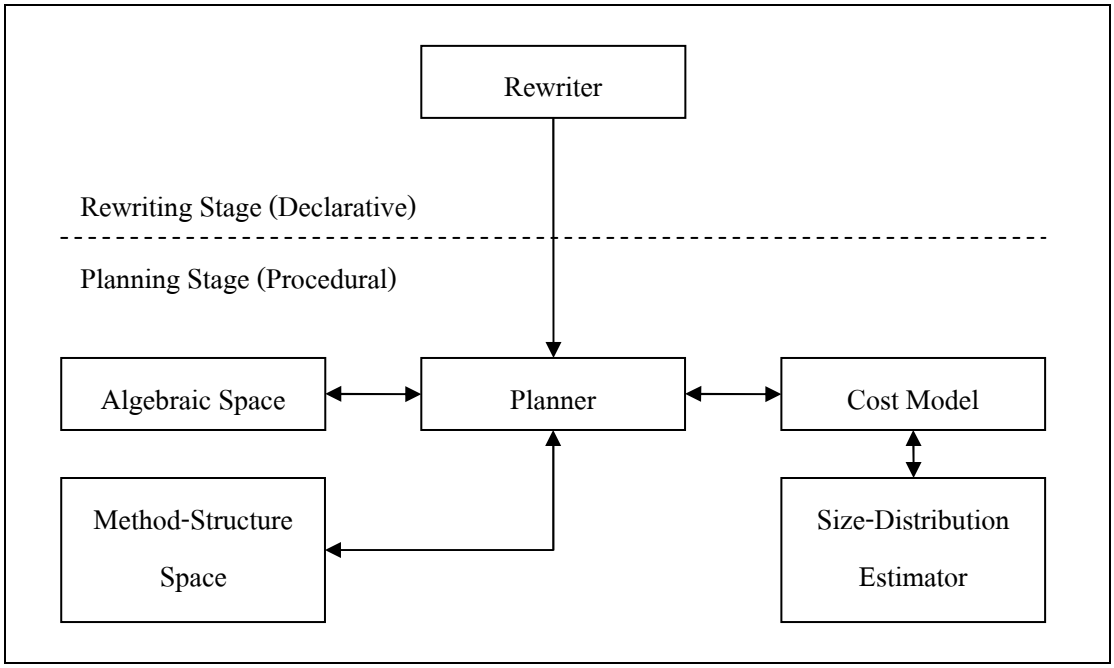
ฐานข้อมูลที่ควรสร้าง Query-evaluation plan ที่มีค่าใช้จ่ายในการประมวลผลข้อคำถามน้อยที่สุด ซึ่งในกระบวนการนี้เราจะเรียกอีกอย่างหนึ่งว่า Query optimization

จากลำดับขั้นตอนข้างต้นที่ได้อธิบายแล้วนั้น ไม่จำเป็นที่ระบบจัดการฐานข้อมูลทุกระบบจะต้องทำตามขั้นตอนที่กล่าวมาทั้งหมด แต่อย่างไรก็ตาม แนวความคิดที่ได้กล่าวมานี้ได้มาจากพื้นฐานของกระบวนการ Query processing ของระบบฐานข้อมูล

2.2 Query optimization

Jarke and Koch (1984) ได้อธิบายว่า Query optimization เป็นกระบวนการในการเลือก Query-evaluation plan ที่มีประสิทธิภาพมากที่สุดในการประมวลผล จากวิธีการ หรือกลยุทธ์ต่าง ๆ ทั้งหมดที่เป็นไปได้ในการประมวลผลข้อคำถามนั้น โดยเฉพาะอย่างยิ่งในกรณีที่ข้อคำถามที่รับเข้ามามีความซ้ำซ้อนมาก โดยทั่วไปแล้วผู้ใช้งานระบบจัดการฐานข้อมูลส่วนใหญ่จะไม่คำนึงถึงเรื่องของเวลาที่จะถูกใช้ในการประมวลผลข้อคำถามที่ผู้ใช้ส่งเข้ามา แต่ผู้ใช้จะคาดหวังว่าระบบจัดการฐานข้อมูลจะเป็นผู้สร้างแผนการหรือวิธีการในการประมวลผลข้อคำถามให้มีประสิทธิภาพมากที่สุด ดังนั้น Query optimization จึงมีบทบาทและหน้าที่ในการเลือกกลยุทธ์ หรือวิธีการต่าง ๆ ในการประมวลผลข้อคำถามที่ผู้ใช้ส่งเข้ามา

Ioannidis (1996) มีการรวบรวมข้อมูลเกี่ยวกับการทำงานในส่วนของ Query optimization โดยได้แสดงโครงสร้างสถาปัตยกรรมในส่วนของ Query optimization ไว้ดังรูปที่ 2.3



รูปที่ 2.3 โครงสร้างสถาปัตยกรรมในขั้นตอน Query optimization

ในกระบวนการ Query optimization ณ ปัจจุบันนี้มีการพัฒนาอย่างต่อเนื่องจนมีวิธีการ หรือเทคนิคใหม่ ๆ ที่เข้ามาช่วยในการเลือกแผนการที่ดีที่สุดในการประมวลผลข้อคำถาม โดยหนึ่งในเทคนิคที่มีประสิทธิภาพในการทำงาน คือ การเพิ่มประสิทธิภาพข้อคำถามเชิงความหมาย (Semantic Query Optimization : SQO) ในเทคนิคนี้เป็นการใช้ความรู้เกี่ยวกับสคีมาในลักษณะที่รูปแบบข้อคำถามเปลี่ยนไปจากเดิม แต่ยังคงให้ผลลัพธ์ที่เหมือนกัน

2.3 Semantic query optimization

Semantic query optimization เป็นกระบวนการที่ใช้หลักการแปลงรูปแบบข้อคำถามโดยใช้ความรู้ในเชิงความหมาย (Semantic transformation) เพื่อให้ได้ข้อคำถามที่ยังคงให้ความหมายเหมือนเดิม (Semantically equivalent) ซึ่งข้อคำถามที่มีการแปลงแล้วนั้น จะใช้เวลาในการประมวลผลที่น้อยลงกว่าเดิม และยังคงให้ผลลัพธ์เหมือนเดิม ในการแปลงข้อคำถามด้วยวิธีการนี้จะใช้เทคนิคการเพิ่ม หรือลดเงื่อนไขออกจากข้อคำถาม (Hsu and Knoblock, 1994)

กระบวนการแปลงรูปแบบข้อคำถามมีอยู่ 3 รูปแบบ หลัก ๆ คือ

- การเพิ่มกฎ (Constraint addition)

  เป็นการนำคอลัมน์ ที่เป็นอินเด็กซ์ เข้ามาเพิ่มในข้อคำถาม เช่น

  ข้อคำถามเดิม:
  ```
  select  *
  from    student
  where   entry = '94' ;
  ```

  Semantic constraint: entry = 94 → rengo >= 94000

  ** ซึ่งคอลัมน์ rengo เป็นคอลัมน์ที่เป็นอินเด็กซ์

  ข้อคำถามใหม่:
  ```
  select  *
  from    student
  where   entry = '94' and rengo >= 94000 ;
  ```

- การลบเงื่อนไข (Constraint deletion)

  เป็นการนำเงื่อนไขที่ไม่จำเป็นออกจากข้อคำถาม เช่น

  ข้อคำถามเดิม:
  ```
  select  *
  from    student
  where   advisor = '100' and status = 'A' ;
  ```

  Semantic constraint: status = A → advisor = 100

  ข้อคำถามใหม่:
  ```
  select  *
  from    student
  where   status = 'A' ;
  ```

- เงื่อนไขขัดแย้ง (Query contradiction)

  ในกรณีนี้ จะไม่มีการส่งข้อคำถามที่รับเข้ามาลงไปประมวลผล เนื่องจากเงื่อนไขภายในข้อคำถามที่รับมาเกิดความขัดแย้งกับ Semantic constraint ที่มีอยู่

ข้อคำถามเดิม: select *
from student
where advisor = ‘105’ and status = ‘A’ ;

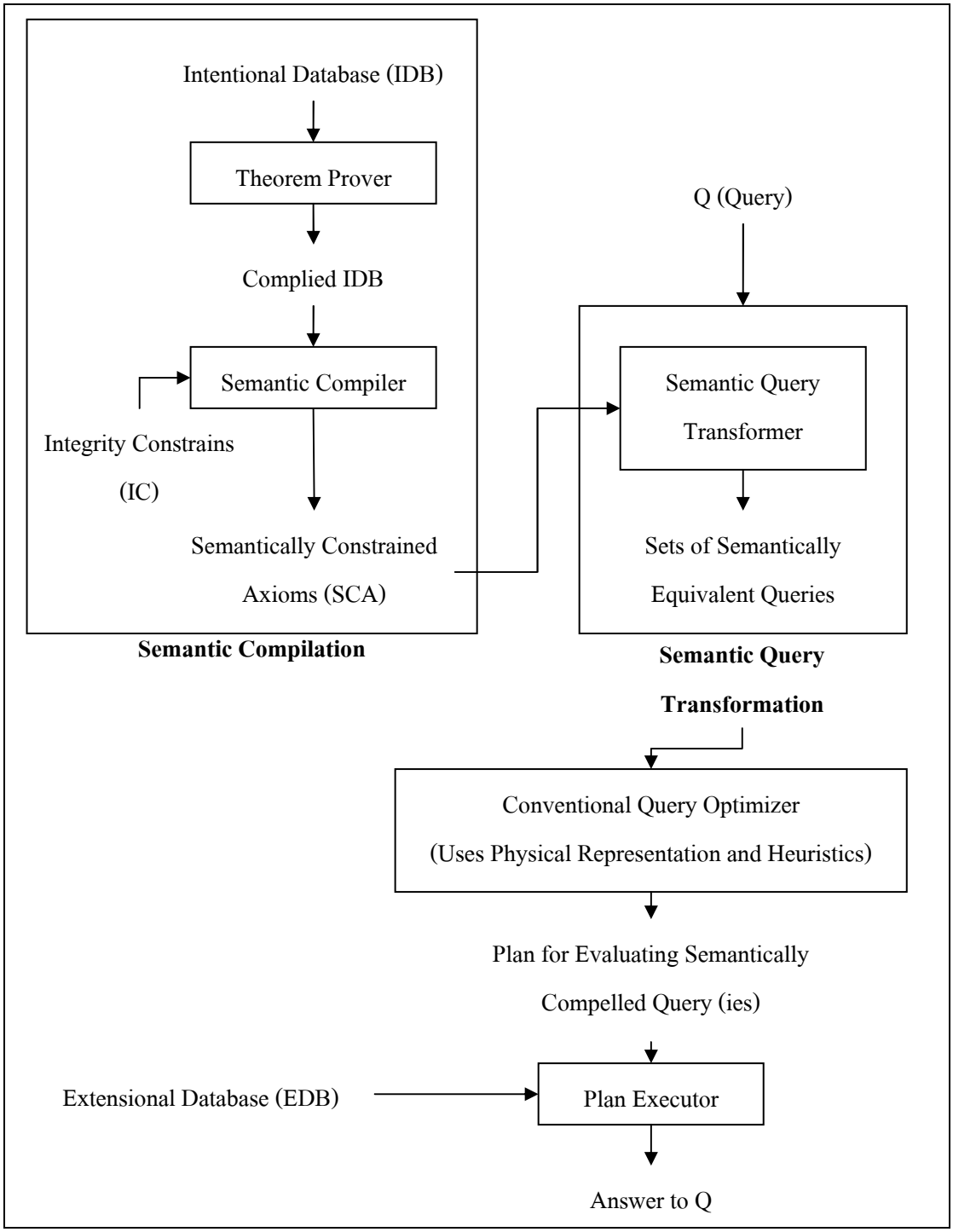
Semantic constraint: status = A → advisor = 100

ข้อคำถามใหม่: No execution, empty answer

ความรู้เชิงความหมาย หรือ Semantic constraint ที่นำมาใช้ในกระบวนการแปลงข้อคำถามนั้นอาจจะได้รับมาจากผู้ใช้งานหรือผู้ดูแลระบบเอง นั่นคือ Integrity constraint หรือได้รับมาจากระบบโดยตรงซึ่งมีการประมวลผลมาจากฐานข้อมูล ตัวอย่างของ Semantic constraint เช่น

dcode = ‘ACCT’ → dname = ‘Accounting’
total_children = 0 → marital = ‘S’
address = ‘bangkok’ → member_card = ‘gold’

Chakravarthy, Grant, and Minker (1990) ได้เสนอแนวคิดพื้นฐานเกี่ยวกับการทำงานของการปรับปรุงประสิทธิภาพข้อคำถามเชิงความหมาย โดยมีการเสนอขั้นตอนการทำงานในกระบวนการปรับปรุงประสิทธิภาพข้อคำถามเชิงความหมายไว้ดังรูปที่ 2.4



รูปที่ 2.4 กระบวนการทำงานของการปรับปรุงประสิทธิภาพข้อคำถามเชิงความหมาย

2.4 เทคนิค Semantic query optimization ใน DB2

Cheng, Grayz, Koo, Leung, Liu, Qian, and Schiefer (1999) มีการใช้เทคนิคของ Semantic query optimization สองเทคนิคใน DB2 Universal คือ Predicate introduction และ Join elimination โดยมีการแบ่งการแปลงรูปแบบข้อคำถามออกเป็น 5 รูปแบบด้วยกัน คือ

1) Join Elimination

- ใช้ข้อบังคับแบบฟอเรนคีย์ (Use foreign key)

เมื่อมีการกำหนดข้อบังคับแบบฟอเรนคีย์ ระหว่างการเชื่อมกันของสองตาราง ระบบจะรับประกันว่าสำหรับแต่ละแถว (Row) ในตารางลูก (Child table) ที่มีคอลัมน์ที่เป็นฟอเรนคีย์ทั้งหมดที่ไม่มีค่าเป็นค่าว่าง (Null) จะมีแถวในตารางหลัก (Parent table) มีค่าตรงกันในคอลัมน์ที่เป็นคีย์หลัก ดังนั้นถ้าจุดประสงค์ของการเชื่อมโยงตารางจึงเป็นเพียงเพื่อตรวจสอบว่า ค่าในคอลัมน์ฟอเรนคีย์ จะมีปรากฏอยู่ในคอลัมน์ที่เป็นคีย์หลัก ดังนั้นเราสามารถตัดการเชื่อมโยงนี้ทิ้งได้

- ตรวจสอบการเชื่อมตารางแบบว่างเปล่า

ในบางครั้งผลลัพธ์ของการเชื่อมตารางนั้นไม่มีค่าในฐานข้อมูล ถ้าเราพบผลลัพธ์ในลักษณะนี้ เราจะสามารถทำการปรับปรุงข้อคำถามเดิมได้โดยการกำจัดการเชื่อมโยงตารางแบบว่างเปล่า

2) Join Introduction

เป็นวิธีการที่ใช้ข้อได้เปรียบของการเชื่อมโยงตารางที่เพิ่มขึ้นมา ถ้าตารางนั้นมีความสัมพันธ์กับตารางเดิม (วิธีนี้จะน่าสนใจมากขึ้นถ้าคอลัมน์ที่นำมาเชื่อมโยงเป็นอินเด็กซ์)

3) Predicate Elimination

จากกฎข้อบังคับ เราสามารถสรุปได้ว่า บางเงื่อนไขที่อยู่ในข้อคำถามนั้นเป็นจริงเสมอ ดังนั้นเราสามารถตัดเงื่อนไขที่ซ้ำซ้อนนั้นออกได้เลย เพื่อเพิ่มประสิทธิภาพในการประมวลผลให้มากขึ้น ถ้าในเงื่อนไขนั้นไม่มีอินเด็กซ์อยู่

4) Predicate Introduction

- แบบใช้อินเด็กซ์ ถ้าเงื่อนไขที่ได้จากกฎข้อบังคับ (Semantic constraint) ซึ่งมีอินเด็กซ์อยู่ เราสามารถนำเงื่อนไขนั้นมาเพิ่มในเงื่อนไขของข้อคำถามนั้น เพื่อทำให้เพิ่มความเร็วในการประมวลผล

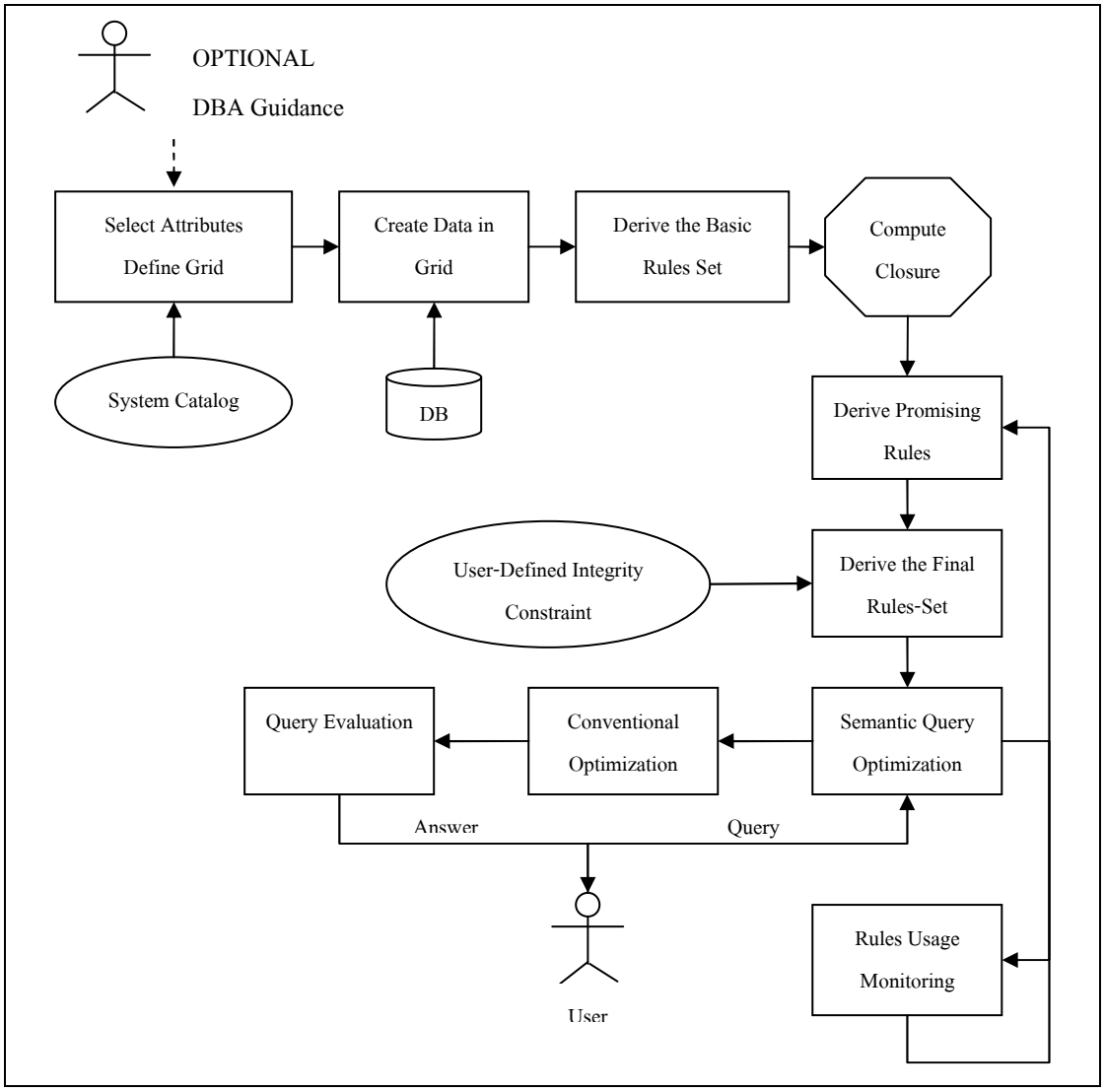
- แบบใช้ลดช่วงการค้นหา (Scan reduction) คล้ายกับแบบใช้อินเด็กซ์ ในบางครั้งเรานำเงื่อนไขใหม่ลงในข้อคำถาม ซึ่งเงื่อนไขนี้สามารถช่วยในการลดช่วงในการค้นหาสำหรับบางข้อมูล ดังนั้นจึงเป็นการเพิ่มประสิทธิภาพในการประมวลผล

5) Detecting the Empty answer set

ถ้าเงื่อนไขนั้นเกิดความขัดแย้งกับกฎข้อบังคับแล้วจะไม่พบคำตอบจากฐานข้อมูล ดังนั้น ถ้าเราทำการตรวจสอบเงื่อนไขนี้ก่อน เราจะสามารถหลีกเลี่ยงการประมวลผลข้อคำถามที่ไม่มีคำตอบนี้ได้

2.5 การเรียนรู้กฎจากการแปลงข้อคำถามสำหรับ Semantic query optimization

Shekhar, Hamidzadeh, Kohli, and Coyle (1993) ได้เสนอแนวความคิดเกี่ยวกับการสร้าง Semantic constraint จากกฎที่ใช้ในการแปลงข้อคำถามด้วยวิธี Data-driven โดยระบบจะสร้างตารางแสดงการกระจายตัวของข้อมูล ที่ง่ายต่อการเรียนรู้กฎที่ใช้ในการแปลงรูปแบบข้อคำถาม และมีการออกแบบกรอบความคิด (Framework) ของระบบไว้ดังรูปที่ 2.5



รูปที่ 2.5 กรอบแนวความคิดการเรียนรู้กฎจากการแปลงข้อคำถามสำหรับ Semantic query optimization

การทำงานเริ่มต้นจาก การเลือกคอลัมน์ที่จะนำไปใช้สำหรับการสร้างตารางการกระจายตัวของข้อมูลโดยคอลัมน์เหล่านี้ อาจถูกเลือกจากผู้ดูแลระบบ หรือถูกเลือกโดยอัตโนมัติ จากนั้นกลุ่มของกฎข้อบังคับ (Semantic constraint) จะถูกดึงมาจากตารางและส่งไปให้ Closure สำหรับสร้างกฎเพิ่มขึ้น ซึ่งผลลัพธ์ที่ได้ คือ กฎที่ถูกคัดเลือกโดย Rule selection เป็นกระบวนการในการคัดเลือกกฎ และกฎที่ถูกคัดเลือกมานี้จะนำไปใช้ในกระบวนการ Semantic query optimization ในการแปลงรูปแบบข้อคำถามต่อไป

การสร้างกฎใหม่โดย Closure มีเทคนิคอยู่ 4 วิธี คือ

สมมติว่าเรามีกฎเดิมอยู่ 2 กฎ ดังนี้

IC1 : A1 $\longrightarrow$ C1

IC2 : A2 $\longrightarrow$ C2

กำหนดให้ช่วงของค่าเป็นดังนี้

R100 = (0..100],

R150 = (100..150],

R200 = (150..200],

R250 = (200..250],

R300 = (250..300],

R350 = (300..350],

R∞ = (350..∞)

เทคนิคที่ 1 จาก IC1 และ IC2

ถ้าเรามี C1 $\longrightarrow$ A2 คือเงื่อนไขข้อบังคับที่มีอยู่แล้ว เราสามารถสรุปได้ว่า

A1 $\longrightarrow$ C2

ดังนั้น เราสามารถเพิ่มกฎข้อนี้เข้าไปยังกลุ่มของกฎข้อบังคับได้

<u>ตัวอย่าง</u> ถ้าเรามีกฎข้อบังคับดังนี้

(DeadWt $\in$ R100) $\longrightarrow$ (ShipType = Barge)

(ShipType = Barge) $\longrightarrow$ (CargoType $\in$ {Grain, Metal})

(CargoType $\in$ {Grain, Metal}) $\longrightarrow$ (DollarValue < 500000)

ดังนั้นเราสามารถเพิ่มกฎข้างล่างนี้ลงในกลุ่มของกฎข้อบังคับได้

(DeadWt $\in$ R100) $\longrightarrow$ (DollarValue < 500000)

เทคนิคที่ 2 จาก IC1 และ IC2

ถ้า A1 และ A2 อยู่ในคอลัมน์เดียวกันแล้ว C1 และ C2 ก็อยู่ในคอลัมน์เดียวกันแล้ว สามารถเพิ่ม

Union (A1, A2) $\longrightarrow$ Union (C1, C2)

ดังนั้น เราสามารถเพิ่มกฎข้อนี้เข้าไปยังกลุ่มของกฎข้อบังคับได้

ตัวอย่าง ถ้าเรามีกฎข้อบังคับ ดังนี้

(DeadWt $\in$ R200) $\longrightarrow$ (ShipType = SuperTanker)

(DeadWt $\in$ R250) $\longrightarrow$ (ShipType = SuperTanker)

(DeadWt $\in$ R300) $\longrightarrow$ (ShipType = SuperTanker)

(DeadWt $\in$ R∞) $\longrightarrow$ (ShipType = SuperTanker)

ดังนั้นเราสามารถเพิ่มกฎข้างล่างนี้ลงในกลุ่มของกฎข้อบังคับได้

(DeadWt > 150) $\longrightarrow$ (ShipType = SuperTanker)

เทคนิคที่ 3 จาก IC1 และ IC2

สามารถเพิ่ม

Intersection (A1, A2) $\longrightarrow$ Intersection (C1, C2)

ตัวอย่าง ถ้าเรามีกฎข้อบังคับ ดังนี้

(DollarValue > 3000) $\longrightarrow$ (Issuer $\in$ {Cloyds, Issuerx, Issuery})

(ShipType = SuperTanker) $\longrightarrow$ (Issuer $\in$ {Cloyds})

ดังนั้นเราสามารถเพิ่มกฎข้างล่างนี้ลงในกลุ่มของกฎข้อบังคับได้

(DollarValue > 3000) AND (ShipType = SuperTanker) $\longrightarrow$

(Issuer $\in$ {Cloyds})

เทคนิคที่ 4 จาก IC1 และ IC2

เราจะเพิ่ม B $\longrightarrow$ C ไปยังกลุ่มของกฎข้อบังคับเมื่อ

B = Not (C1) $\longrightarrow$ D = Not (A1)

ตัวอย่าง ถ้าเรามีกฎข้อบังคับ ดังนี้

(DeadWt > 150) $\longrightarrow$ (ShipType = SuperTanker)

ดังนั้นเราสามารถเพิ่มกฎข้างล่างนี้ลงในกลุ่มของกฎข้อบังคับได้

Not (ShipType = SuperTanker) $\longrightarrow$ Not(DeadWt > 150)

และถ้าค่าที่อยู่ในคอลัมน์ ShipType คือ {Barge, Tanker, SuperTanker}

ดังนั้น จะได้

Not (ShipType = SuperTanker) = (ShipType ∈ {Barge, Tanker})

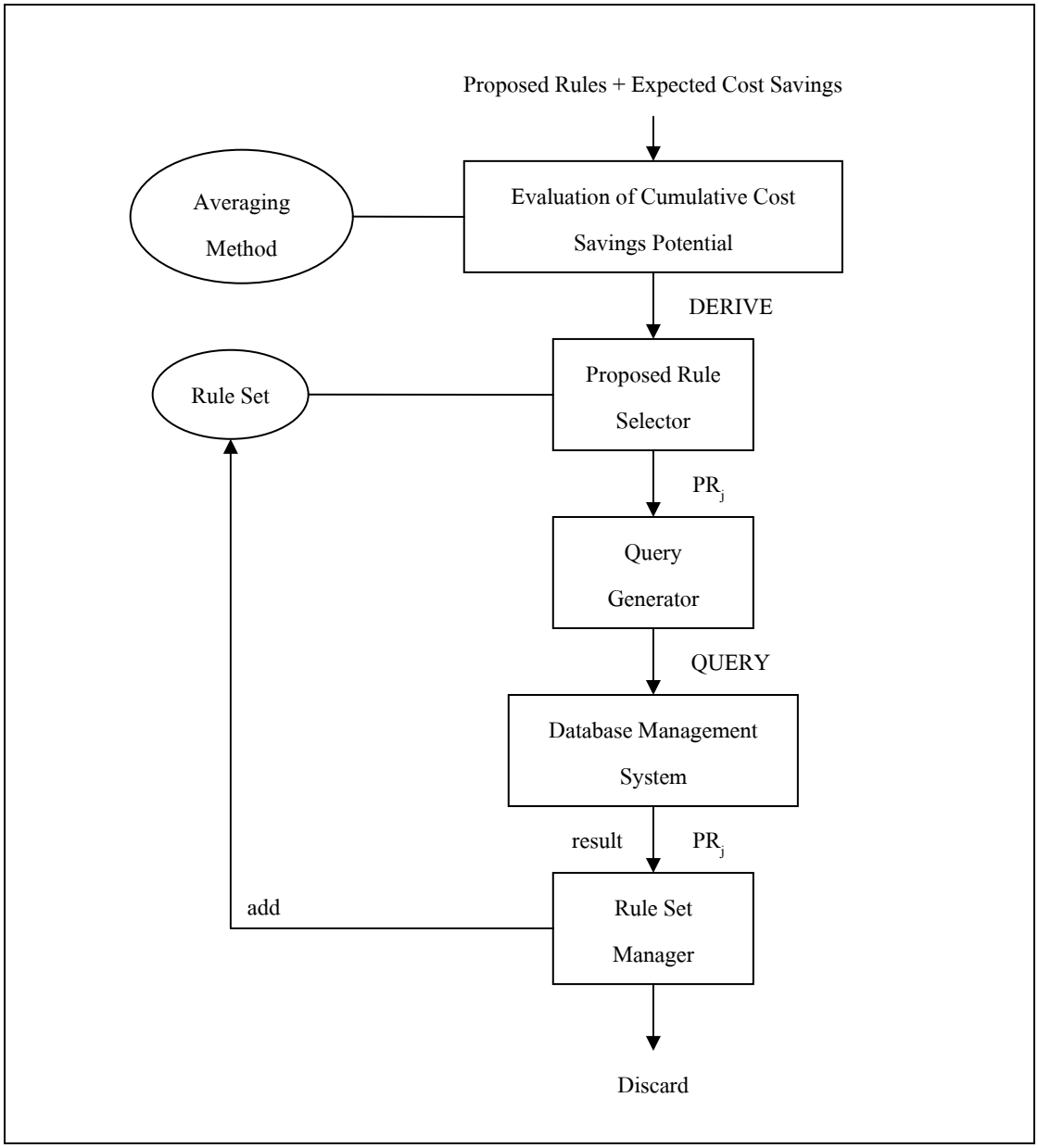
และ

Not (DeadWt > 150) = (DeadWt <= 150)

ดังนั้นเราจะได้กฎข้อบังคับใหม่ดังนี้

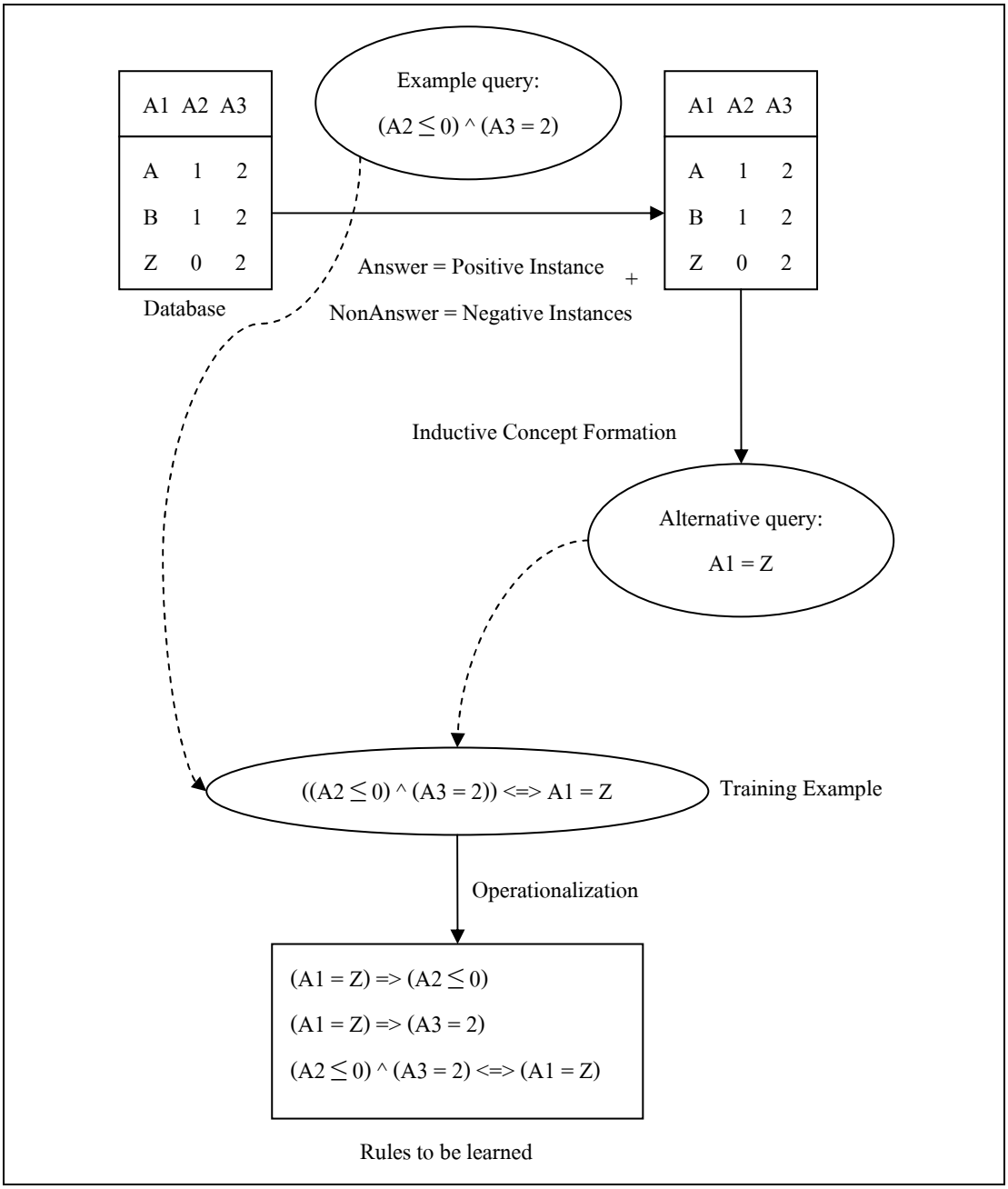
(ShipType ∈ {Barge, Tanler}) ⟶ (DeadWt <= 150)

ในการใช้เทคนิคการค้นหากฎข้อบังคับในลักษณะข้างต้นนี้ Siegel, Sciore, and Salveter (1992) ได้ออกแบบกระบวนการค้นหากฎข้อบังคับในลักษณะที่คล้ายกันดังรูปที่ 2.6



รูปที่ 2.6 กระบวนการสร้างกฎข้อบังคับแบบอัตโนมัติ

Hsu and Knoblock (1994) เสนอแนวคิดในการค้นหากฎข้อบังคับจากการเรียนรู้ลักษณะเงื่อนไขของข้อคำถามที่ส่งเข้ามาประมวลผลเปรียบเทียบกับข้อมูลในฐานข้อมูล ซึ่งมีลักษณะการทำงานดังรูปที่ 2.7



รูปที่ 2.7 ตัวอย่างการเรียนรู้ข้อคำถามเพื่อสร้างกฎข้อบังคับ

Chaudhuri, Narasayya, and Sarawagi (2002) ได้เสนอแนวคิดเกี่ยวกับเทคนิคที่นำมาใช้ในการทำ Semantic query optimization โดยเป็นการนำเทคโนโลยีทางด้านการทำเหมืองข้อมูล (Data mining) และการปรับปรุงประสิทธิภาพข้อคำถาม (Query optimization) มาประยุกต์เข้าด้วยกัน ในระบบจัดการฐานข้อมูลเชิงสัมพันธ์ยุคปัจจุบันนี้ มีความสามารถในการค้นหารูปแบบ

ของข้อมูล (Predicate) หรือความรู้ต่าง ๆ ที่ซ่อนอยู่ภายใน (Knowledge) ดังนั้นจึงมีการนำเอารูปแบบของข้อมูลที่ได้จากการทำเหมืองข้อมูล (Mining model) มาช่วยในการปรับปรุงประสิทธิภาพข้อคำถาม ซึ่งในการทดสอบครั้งนี้มีการใช้เทคนิคในการทำเหมืองข้อมูลด้วยกัน 3 เทคนิค คือ Decision tree, Naive bayes และ Clustering การหา Predicate model สามารถหาได้ 2 วิธีหลัก ๆ คือ

1) หาได้จากระบบฐานข้อมูลเอง

ในโปรแกรม Micorsoft Analysis Server สามารถหาโมเดลเหมืองข้อมูลจากระบบฐานข้อมูลได้เลย โดยโปรแกรมดังกล่าวเป็นส่วนหนึ่งของระบบฐานข้อมูล Micorsoft SQL Server 2000 ในที่นี้เราจะสร้างโมเดลเหมืองข้อมูลเพื่อหาระดับความเสี่ยงของลูกค้า โดยมีคอลัมน์ที่เกี่ยวข้อง คือ เพศของลูกค้า (Gender), การซื้อ (Purchases) และอายุของลูกค้า (Age) โดยใช้อัลกอริทึม Decision tree ได้ดังตัวอย่างต่อไปนี้

```
CREATE MINING MODEL Risk_Class (                    //Name of model
Customer_ID     LONG   KEY,                        //Soure column
Gender          TEXT   DISCRETE,                   //Source column
Risk            TEXT   DISCRETE PREDICT,           //Prediction column
Purchases       DOUBLE DISCRETIZED (),             //Source column
Age             DOUBLE DISCRETIZED ())             //Source column
USING [Decision_tree]                              //Mining algorithm
```

ในการสอบถามข้อมูลนั้นจะประกอบไปด้วย โมเดล M และกลุ่มข้อมูล (Data set) D โดยจะใช้การ Predicate join เข้ามาเชื่อมระหว่าง M และ D ซึ่ง Predicate join มีความแตกต่างจากการ Join กันของตารางทั่ว ๆ ไป เพราะ ภายในโมเดล M ไม่มีข้อมูลอยู่จริง ดังนั้นเราสามารถเขียนไวยากรณ์ในการสอบถามข้อมูลได้ดังนี้

```
SELECT      D.Customer_ID, M.Risk
FROM        [Risk_Class] M
PREDICTION JOIN (
    SELECT      Customer_ID, Gender, Age, sum (Purchases) as SP
    FROM        Customer D
    GROUP BY    Customer_ID, Gender, Age) as D
```

```
ON  M.Gender = D.Gender     and
        M.Age = D.Age       and
        M.Purchases = T.SP
WHERE M.Risk = "low"
```

2) นำเข้าโมเดลเหมืองข้อมูลจากภายนอก

ในระบบฐานข้อมูล DB2 Universal Database สามารถนำเข้าโมเดลเหมืองข้อมูลได้ โดยโมเดลเหมืองข้อมูลที่ได้นี้สร้างมาจาก IBM's Intelligent Miner (IM) ซึ่งจะออกมาอยู่ในรูปแบบของ Flat ไฟล์, XML ไฟล์ และ PMML ไฟล์ ในการนำไฟล์โมเดลเหมืองข้อมูลเข้าระบบฐานข้อมูลสามารถทำได้ดังนี้

```
INSERT INTO IDMMX.ClassifModels values ('Risk_Class',
IDMMX.DM impClasFile ('/tmp/myclassifier.x'))
```

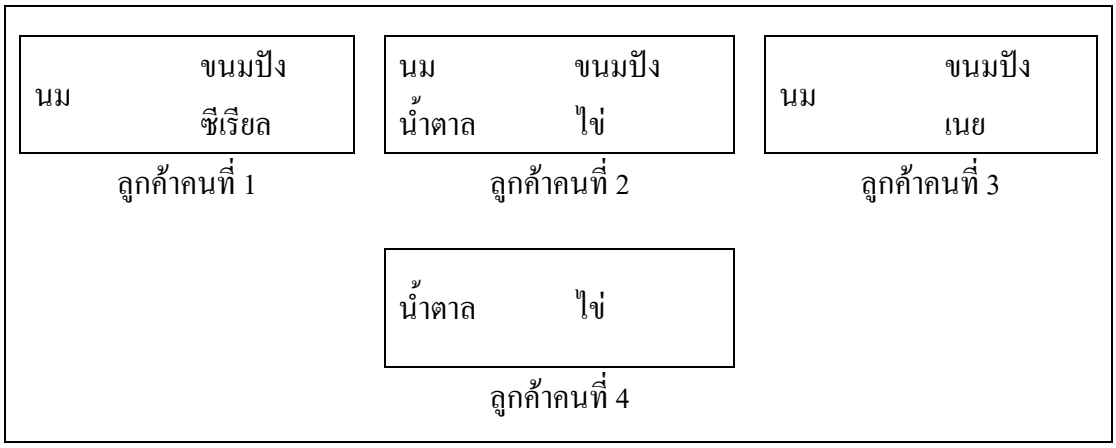
เมื่อผู้ใช้โหลดโมเดลเหมืองข้อมูลเข้าฐานข้อมูลเรียบร้อยแล้ว สามารถเรียกดูข้อมูลได้ดังนี้

```
SELECT Customer_ID, Risk
FROM (
        SELECT Customer_ID, IDMMX.DM_getPredClass (
                IDMMX.DM_applyClasModel (c.model,
                IDMMX.DM_applData (IDMMX.DM_applData ('AGE', s.age),
                'PURCHASE', s.purchase))) as Risk
        FROM ClassifModels c, Customer_list s
        WHERE c.modelname='Risk_Class' and s.salary < 40000
) WHERE Risk = 'low'
```

2.6 การค้นหากฎความสัมพันธ์

ข้อมูลจำนวนมหาศาลที่ถูกจัดเก็บไว้ในระบบจัดการฐานข้อมูลขนาดใหญ่ ของแต่ละองค์กรนั้น ณ ปัจจุบัน องค์กรต่าง ๆ จำนวนมากที่มีข้อมูลเหล่านี้อยู่เริ่มหันมาให้ความสนใจในการค้นหากฎความสัมพันธ์ของข้อมูล (Han and Kamber., 2001) ในระบบฐานข้อมูลของตนเองมากขึ้นเรื่อย ๆ ซึ่งในการค้นหากฎความสัมพันธ์จากข้อมูลที่มีอยู่จำนวนมหาศาลในฐานข้อมูลนี้ สามารถนำกฎความสัมพันธ์ที่ได้มาช่วยในการตัดสินใจเกี่ยวกับการวางแผนธุรกิจขององค์กรได้ เช่น การจัดวางสินค้า หรือ การวางแผนการตลาด

ข้อมูลที่ใช้ในการค้นหากฎความสัมพันธ์ เราจะเรียกข้อมูลนี้ว่า Market basket data ในกระบวนการนี้จะเป็นการวิเคราะห์พฤติกรรมการซื้อสินค้า โดยค้นหาจากความสัมพันธ์ของสินค้าแต่ละชิ้นในตะกร้าสินค้าของลูกค้าแต่ละคนดังรูปที่ 2.8 กฎความสัมพันธ์ที่ได้นี้สามารถนำไปช่วยในการพัฒนาแผนงาน หรือกลยุทธ์ทางการตลาด โดยเราสามารถทราบถึงความถี่ของลูกค้าในการซื้อสินค้าคู่กัน เช่น ถ้าลูกค้าซื้อนมแล้ว ลูกค้ามักซื้อขนมปังตามมาด้วยเช่นกัน จากข้อมูลเหล่านี้เราสามารถนำความรู้ที่ได้จากการวิเคราะห์เข้ามาช่วยในการพัฒนาแผนการตลาดของเราได้ เช่น การออกแบบรายการสินค้า หรือ การจัดวางสินค้าภายในร้านได้



รูปที่ 2.8 การวิเคราะห์พฤติกรรมการซื้อ

ตัวอย่างของความสัมพันธ์ เช่น ลูกค้า 98 เปอร์เซ็นต์เมื่อซื้อยางรถยนต์ และอุปกรณ์ประดับยนต์แล้ว จะใช้บริการเปลี่ยนที่นั้นเลย เป็นต้น ซึ่งถ้าเราหาความสัมพันธ์ออกมาได้ เราสามารถนำมาพัฒนาเกี่ยวกับการขายได้ รวมไปถึง การออกแบบนิตยสารการขายต่าง ๆ ของแถมที่จะแถมให้กับลูกค้า การจัดการวางชั้นสินค้าต่าง ๆ ภายในร้านค้า หรือรูปแบบการซื้อสินค้าต่าง ๆ ของลูกค้า ซึ่งในฐานข้อมูลจะมีข้อมูลมหาศาล ดั้งนั้นเราจึงต้องมีเครื่องมือเพื่อเข้ามาช่วยในการนำข้อมูลทั้งหมดมาหาความสัมพันธ์ต่าง ๆ (Association) ที่ซ่อนอยู่ภายใน หรือเราจะเรียกการกระทำแบบนี้ว่า Knowledge Discovery in Databases จากข้อมูลในฐานข้อมูลหรือคลังข้อมูล ทำให้ทราบถึง ภาวะความน่าจะเป็นของเหตุการณ์ต่าง ๆ ที่เกิดขึ้นพร้อมกันในขณะเดียวกัน หรือเกิดขึ้นตามกัน เมื่อเวลาผ่านไป เช่น ภาวะความน่าจะเป็นที่ลูกค้าซื้อสินค้า X และ Y ด้วยกันในการซื้อครั้งหนึ่ง ๆ หรือภาวะความน่าจะเป็นที่ลูกค้าซื้อสินค้า X และ Y แล้วจะเกิดการซื้อสินค้า Z ในการซื้อครั้งหนึ่ง ๆ

หรือภาวะความน่าจะเป็นที่ลูกค้าซื้อสินค้า X และ Y แล้วจะซื้อสินค้า Z ในครั้งต่อไปเป็นต้น การศึกษานี้มุ่งเฉพาะการเกิดขึ้นของเหตุการณ์มิได้สนใจเกี่ยวกับปริมาณของเหตุการณ์ เช่น มิได้สนใจว่าลูกค้าซื้อสินค้า X สินค้า Y หรือสินค้า Z จำนวนเท่าไร แต่สนใจความถี่ที่เกิดเหตุการณ์ซื้อสินค้า X สินค้า Y หรือสินค้า Z

ถ้าเรามองสินค้าที่มีอยู่เป็นไอเท็ม (Item) ดังนั้นเราสามารถเขียนกฎความสัมพันธ์ให้อยู่ในรูปของไอเท็มที่เป็นเหตุ ไปสู่ไอเท็มที่เป็นผลได้ ดังนี้

computer => software [support = 2% , confidence = 60%]

สำหรับค่าสนับสนุน หรือ Support และค่าความเชื่อมั่น หรือ Confidence เป็นค่าที่มีความสำคัญต่อการค้นหากฎความสัมพันธ์จากข้อมูล ทั้งสองค่านี้แสดงถึงความถูกต้องของกฎความสัมพันธ์ที่ได้ โดยค่าสนับสนุน หมายถึง ความน่าจะเป็นที่ลูกค้าจะซื้อสินค้าทั้งสองชนิดนั้น และค่าความเชื่อมั่น หมายถึง ความน่าจะเป็นที่ลูกค้าจะซื้อสินค้าทั้งสองชิ้นไปพร้อม ๆ กัน โดยค่าทั้งสองค่านี้จะถูกกำหนดโดยผู้ใช้ หรือผู้เชี่ยวชาญเฉพาะในแต่ละด้านนั้น ๆ แนวความคิดพื้นฐานสำหรับการค้นหากฎความสัมพันธ์ มีนิยาม (Han and Kamber., 2001) ดังต่อไปนี้

1) ไอเท็มเซต (Itemsets : $I = \{i_1, i_2, i_3 \ldots\}$) คือกลุ่มของไอเท็มทั้งหมด ซึ่งไอเท็มในที่นี้อาจเป็นชื่อของสินค้า, วัตถุ หรือข้อมูลใด ๆ ก็ได้ที่ต้องการนำมาหาความสัมพันธ์
2) ทรานแซคชัน (Transaction : T) เป็นเซตย่อยของไอเท็ม โดยที่ $T \subseteq I$
3) เซตของข้อมูล (Data : D) คือเซตที่มีทรานแซคชันทุกตัวเป็นสมาชิก โดยที่ $D = \{t_1, t_2, \ldots, t_3\}$
4) การกำหนดค่าสนับสนุน (Support) หรือ s_{min} , โดยที่ $0 \leq s_{min} \leq 1$
5) การกำหนดค่าความเชื่อมั่น (Confidence) หรือ c_{min} , โดยที่ $0 \leq c_{min} \leq 1$
6) ทรานแซคชัน T บรรจุเซตย่อยของไอเท็ม X ก็ต่อเมื่อ $X \subseteq t$
7) ค่าสนับสนุนของไอเท็มเซต X คือ $s(X) = | \{t \in D | X \subseteq t\} | / | D |$
8) กฎที่ได้อยู่ในรูปแบบ $L \rightarrow R$ เมื่อ $L \subset I$, $R \subset I$ และ $L \cap R = \varnothing$
9) ค่าความเชื่อมั่นของกฎ $L \rightarrow R$ คือ $c(L, R) = s(L \cup R) / s(L)$
10) ค่าสนับสนุนของกฎ $L \rightarrow R$ คือ $s(L, R) = s(L \cup R) / | D |$
11) แต่ละกฎที่ถูกค้นพบเรียกว่ากฎความสัมพันธ์ (Association rule) โดยการค้นหากฎความสัมพันธ์ คือการค้นหาความสัมพันธ์ทั้งหมดในทรานแซคชันทุกตัว โดยกฎความสัมพันธ์ที่หามาได้ทั้งหมดจะต้องมีค่าสนับสนุนมากกว่าหรือเท่ากับค่าสนับสนุน

ขั้นต่ำที่ได้กำหนดเอาไว้ ($s (L \cup R) \geq s_{min}$) และมีค่าความเชื่อมั่นมากกว่าหรือเท่ากับค่าความเชื่อมั่นขั้นต่ำที่กำหนดไว้ ($c(L, R) \geq c_{min}$)

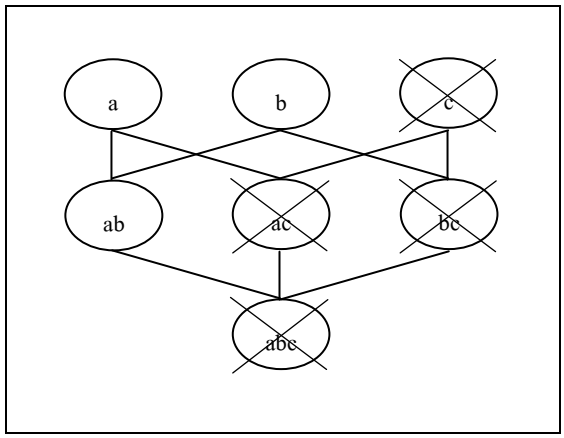
12) กลุ่มของไอเท็มที่มีค่าสนับสนุนสูงกว่าค่าขั้นต่ำเรียกว่า “ไอเท็มเซตที่ปรากฏบ่อย” (Frequent itemsets or large itemsets)

2.7 วิธีการค้นหากฎความสัมพันธ์

การค้นหากฎความสัมพันธ์แบ่งออกเป็น 2 ขั้นตอน คือ

ขั้นตอนที่ 1 การค้นหาไอเท็มเซตที่ปรากฏบ่อย (Find all frequent itemsets)

การค้นหาไอเท็มเซตที่ปรากฏบ่อย เป็นการนำไอเท็มทั้งหมดที่มีอยู่มาจัดให้อยู่ในหมู่ของไอเท็ม ซึ่งในแต่ละหมู่ของไอเท็มจะประกอบไปด้วยไอเท็มหลายไอเท็มหรือไอเท็มเดียวก็ตามเข้ามาจับกลุ่มด้วยกัน โดยจำนวนของหมู่ไอเท็ม จะมีขนาดเพิ่มขึ้นตามจำนวนของไอเท็มเซต โดยเฉพาะถ้าไอเท็มเซตมีจำนวนมากแล้วนั้น การค้นหาไอเท็มเซตที่ปรากฏบ่อยก็จะมีเซตที่ต้องค้นหามากขึ้นตามไปด้วย แต่ในการค้นหาไอเท็มเซตที่ปรากฏบ่อยนี้ ไม่จำเป็นต้องมีการแจกแจงเซตของไอเท็มทุกการจัดหมู่ เพราะถ้าทำการแจกแจงแล้วพบไอเท็มเซตใดที่ไม่ปรากฏอยู่ในไอเท็มเซตที่ปรากฏบ่อย เราไม่จำเป็นต้องทำการแจกแจงไอเท็มเซตอื่น ๆ ที่ไอเท็มเซตนี้เป็นเซตย่อยอีก เช่น จากไอเท็ม a, b และ c สามารถสร้างไอเท็มเซตที่ปรากฏบ่อยได้ดังรูป 2.9 แต่ถ้าเรารู้ว่าไอเท็ม {c} ไม่ใช่ไอเท็มเซตที่ปรากฏบ่อย ก็ไม่จำเป็นต้องจัดหมู่ที่มีไอเท็มนี้เป็นเซตย่อยของหมู่นั้น คือ {ac}, {bc} และ {abc}



รูปที่ 2.9 การจัดหมู่ของสมาชิกในไอเท็มเซต {a, b, c}

ในการคัดเลือกไอเท็มเพื่อนำมาใช้ในการสร้างไอเท็มเซตที่ปรากฏบ่อย จะนำมาเฉพาะไอเท็มที่มีจำนวนปรากฏที่มีค่ามากกว่า หรือเท่ากับค่าสนับสนุนที่กำหนดไว้

ขั้นตอนที่ 2 การสร้างกฎความสัมพันธ์ด้วยไอเท็มเซตที่ปรากฏบ่อย

การสร้างกฎความสัมพันธ์จะสร้างมาจากไอเท็มเซตที่ปรากฏบ่อย ที่ได้จากการค้นหาในขั้นตอนที่แล้ว โดยกฎที่สร้างขึ้นนั้นจะต้องมีค่าความเชื่อมั่นขั้นต่ำที่ได้จากกฎที่สร้างนี้มากกว่าหรือเท่ากับค่าความเชื่อมั่นที่กำหนดไว้โดยผู้ใช้

2.8 ประเภทของกฎความสัมพันธ์

ในการวิเคราะห์พฤติกรรมการซื้อสินค้าของลูกค้า (Market basket analysis) เป็นอีกรูปแบบหนึ่งในการค้นหากฎความสัมพันธ์ ในความเป็นจริงแล้วกฎความสัมพันธ์มีอยู่หลายประเภท เราสามารถจำแนกกฎความสัมพันธ์ได้หลายแนวทางขึ้นอยู่กับกฎเกณฑ์ที่ใช้ในการจำแนก

1) กฎความสัมพันธ์แบบข้อเท็จจริง (Boolean association rule) เป็นความสัมพันธ์ที่บ่งบอกถึงการมีหรือไม่มีไอเท็มนั้นอยู่ ซึ่งเป็นข้อเท็จจริงจากการวิเคราะห์พฤติกรรมของลูกค้า เช่น

$$\text{computer} \rightarrow \text{software} \tag{2.1}$$

2) กฎความสัมพันธ์เกี่ยวกับปริมาณ (Quantitative association rule) คือ กฎความสัมพันธ์ที่อธิบายเกี่ยวกับความสัมพันธ์ของปริมาณไอเท็ม หรือแอททริบิวต์ซึ่งในกฎความสัมพันธ์นี้ปริมาณของไอเท็ม หรือแอททริบิวต์จะถูกแสดงให้อยู่ในรูปของช่วงข้อมูล ดังตัวอย่างกฎความสัมพันธ์เกี่ยวกับปริมาณข้างล่างนี้ โดยที่ X คือ ตัวแปรของลูกค้า

$$\text{age}(X, \text{“30 … 39”}) \wedge \text{income}(X, \text{“42K … 48K”}) \rightarrow \text{buys}(X, \text{high resolution TV}) \tag{2.2}$$

3) กฎความสัมพันธ์แบบหนึ่งมิติ (Single-dimensional association rule) คือ กฎความสัมพันธ์ที่มีไอเท็ม หรือแอททริบิวต์อ้างอิงกับข้อมูลเพียงมิติเดียว ดังเช่นกฎความสัมพันธ์ที่ (2.1) ซึ่งสามารถเขียนให้อยู่ในรูปแบบของกฎความสัมพันธ์แบบหนึ่งมิติได้ดังนี้

$$\text{buys}(X, \text{“computer”}) \rightarrow \text{buys}(X, \text{“software”}) \tag{2.3}$$

4) กฎความสัมพันธ์หลายมิติ (Multi-dimensional association rule) คือ กฎความสัมพันธ์ที่มีการอ้างถึงไอเท็มหรือแอททริบิวต์มากกว่าสองมิติขึ้นไปอย่างเช่นกฎความสัมพันธ์ที่ (2.2) เราสามารถพิจารณาได้ว่าเป็นกฎความสัมพันหลายมิติเนื่องจากมีการอ้างถึงไอเท็มหรือแอททริบิวต์ด้วยกันสามมิติ คือ age (อายุ), income (รายได้) และ buys (การซื้อ)

5) กฎความสัมพันธ์หลายระดับ (Multilevel association rule) คือ การค้นหากฎความสัมพันธ์บางวิธีที่สามารถค้นหากฎที่มีระดับของนามธรรมที่แตกต่างกัน ดังตัวอย่าง ถ้ากฎของความสัมพันธ์มีดังนี้

age(X, “30 … 39”) ⟶ buys(X, “laptop computer”) (2.4)

age(X, “30 … 39”) ⟶ buys(X, “computer”) (2.5)

จากกฎความสัมพันธ์ที่ (2.4) และ (2.5) มีการอ้างถึงระดับของนามธรรมที่แตกต่างกัน นั้นคือ computer มีระดับที่สูงกว่า laptop computer

6) กฎความสัมพันธ์ระดับเดียว (Single-level association rule) คือ กฎความสัมพันธ์ที่มีลักษณะคล้ายกับกฎความสัมพันธ์หลายระดับ แตกต่างกันเพียงกฎความสัมพันธ์ระดับเดียวจะมีการอ้างอิงข้อมูลที่อยู่ในระดับเดียวกันเท่านั้น เช่น

age(X, “30 … 39”) ⟶ buys(X, “เบียร์”) (2.6)

age(X, “30 … 39”) ⟶ buys(X, “สุรา”) (2.7)

จากกฎความสัมพันธ์ที่ (2.6) และ (2.7) มีการอ้างถึงระดับของข้อมูลอยู่ในระดับเดียวกัน นั้นคือ เบียร์มีระดับเดียวกันกับสุรา

2.9 การค้นหากฎความสัมพันธ์จากฐานข้อมูลขนาดใหญ่

Agrawal, Imielinski, and Swami (1993) เสนอแนวคิดเกี่ยวกับการค้นหากฎความสัมพันธ์ของข้อมูลจากฐานข้อมูลที่มีขนาดใหญ่ โดยมีการนำข้อมูลมาจากรายการทรานแซคชัน (Transaction) ของลูกค้าซึ่งรายการทรานแซคชันเป็นข้อมูลการสั่งซื้อสินค้าของลูกค้า อัลกอริทึมที่ใช้ในการค้นหากฎความสัมพันธ์นี้ เรียกว่า อัลกอริทึมเอไอเอส (AIS algorithm) โดยกฎความสัมพันธ์ที่ได้จะอยู่ในรูปแบบของ $I_k \rightarrow I_l$ โดยที่ I_k เป็นข้อมูลชุดที่ k, I_l เป็นข้อมูลชุดที่ l และ $I_k \cap I_l = \varnothing$,k = 1, 2, 3, … อัลกอริทึมนี้มีการทำงานดังรูปที่ 2.10

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

while F ≠ ∅ do begin
        let Candidate set C = ∅;
        forall database tuples t do
                forall itemsets f in F do
                        if t contains f then begin
                                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 begin
                                                c_f.count = 0;
                                                C = C + c_f;
                                end
                end
        end
end
```

รูปที่ 2.10 อัลกอริทึมเอไอเอส (AIS algorithm)

```
let F = ∅;
        forall itemsets c in C do begin
                if count/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 2.10 อัลกอริทึมเอไอเอส (AIS algorithm) (ต่อ)

ในอัลกอริทึมนี้แคนดิเดทไอเท็มเซต จะถูกสร้างและนับความถี่ที่ปรากฏบ่อยในฉะเดียวกันที่มีการอ่านข้อมูลจากฐานข้อมูล (On-the-fly) โดยเมื่อมีการกำหนดค่าไอเท็มเซตที่ปรากฏบ่อยแล้ว จะทำการตรวจสอบว่าไอเท็มเซตที่ได้ปรากฏอยู่ในทรานแซคชันหรือไม่ และจะสร้างแคนดิเดทไอเท็มเซตได้จากไอเท็มเซตที่ปรากฏบ่อยนี้ จากนั้นจะเป็นการนับความถี่ของแคนดิเดทแต่ละตัว ถ้าแคนดิเดทที่ได้ปรากฏอยู่ในแคนดิเดทไอเท็มเซตเดิมอยู่แล้วจะทำการเพิ่มค่าความถี่ขึ้นอีกหนึ่ง แต่ถ้าไม่ปรากฏอยู่ในแคนดิเดทไอเท็มเซตเดิม จะทำการเพิ่มแคนดิเดทนั้นเข้าไปยังแคนดิเดทไอเท็มเซตและกำหนดค่าเริ่มต้นเป็นหนึ่ง

2.10 อัลกอริทึมเอไพรออรี

Agrawal and Srikant (1994) ได้เสนออัลกอริทึมในการค้นหากฎความสัมพันธ์จากข้อมูลด้วยกันสองอัลกอริทึม คือ อัลกอริทึมเอไพรออรี (Apriori algorithm) และ อัลกอริทึมเอไพรออรีทีไอดี (AprioriTid algorithm) เป็นอัลกอริทึมที่มีประสิทธิภาพในการค้นหากฎความสัมพันธ์ที่มีความสามารถเหนือกว่าอัลกอริทึมอื่น ๆ ก่อนหน้านี้ ชื่อของอัลกอริทึมถูกตั้งขึ้นมาตามหลักการทำงานของอัลกอริทึม นั่นคือ อัลกอริทึมเอไพรออรีจะทำงานแบบวนรอบไปเรื่อย ๆ เป็นลำดับชั้นหรือที่เรียกว่า Level–wise โดย k–itemset หรือไอเท็มเซตที่ปรากฏบ่อยในรอบปัจจุบันจะนำไปสร้างไอเท็มเซตที่ปรากฏบ่อยในรอบต่อไปคือ (k + 1)–itemset ยกตัวอย่างเช่น ถ้าในรอบแรกเมื่อเราพบ 1–itemset เราจะเรียกไอเท็มเซตนี้ว่า L_1 และต่อมาจะนำ L_1 มาใช้ในการสร้าง L_2 และเมื่อเราสร้าง 2–itemset เสร็จเรียบร้อย จากนั้นจะนำ L_2 มาใช้ในการสร้าง L_3 ต่อไป และทำการค้นหาแบบนี้

ไปเรื่อย ๆจนกระทั่งไม่สามารถสร้าง k–itemset ได้อีกซึ่งในขั้นตอนการค้นหากฎความสัมพันธ์มีการแบ่งออกเป็นสองส่วนหลัก ๆ คือ

1) ทำการค้นหาไอเท็มเซตจากทรานแซคชันที่ได้ซึ่งไอเท็มเซตที่ได้นั้นจะต้องมีค่าความถี่มากกว่าหรือเท่ากับค่าสนับสนุนที่ผู้ใช้ระบุไว้ โดยไอเท็มเซตที่มีค่าความถี่มากกว่าหรือเท่ากับค่าสนับสนุนนี้ เราจะเรียกว่า ไอเท็มเซตที่ปรากฏบ่อย (Large itemset) ส่วนไอเท็มเซตนอกจากนี้เรียกว่า Small itemset

2) นำไอเท็มเซตที่ปรากฏบ่อยที่ได้มาสร้างเป็นกฎความสัมพันธ์ กฎที่ได้จะอยู่ในรูปแบบ item x $\rightarrow$ item y โดยกฎที่ได้จะต้องมีค่าความเชื่อมั่น (Confidence) มากกว่าหรือเท่ากับที่ผู้ใช้ระบุไว้ ค่าความเชื่อมั่นนี้สามารถคำนวนได้จาก

$$Confidence = \frac{Support(x \cup y)}{Support(x)}$$

อัลกอริทึมทั้งสองอัลกอริทึมนี้ จะสร้างแคนดิเดทไอเท็มเซตจากไอเท็มเซตที่ปรากฏบ่อยในลำดับที่แล้วโดยไม่มีการอ่านข้อมูลจากทรานแซคชัน ดังนั้นจึงเป็นข้อแตกต่างจากอัลกอริทึมที่ผ่านมาแนวคิดพื้นฐานของการสร้างแคนดิเดทไอเท็มเซตนี้คือ ทุก ๆ สับเซตของไอเท็มเซตจะต้องเป็นสมาชิกของไอเท็มเซตก่อนหน้านี้ ดังนั้นแคนดิเดทไอเท็มเซตที่ k จะถูกสร้างขึ้นจากไอเท็มเซตที่ปรากฏบ่อยที่ k – 1 มารวมเข้ากับตัวมันเอง (Join) และทำการลบไอเท็มเซตย่อยที่เกิดจากการรวมกันของไอเท็มเซตที่ปรากฏบ่อยที่ไม่ได้ปรากฏอยู่ในไอเท็มเซตที่ปรากฏบ่อย ในรอบที่ผ่านมา ผลจากกระบวนการนี้จะทำให้แคนดิเดทไอเท็มเซตที่ได้มีจำนวนน้อยลงไปเรื่อย ๆ

อัลกอริทึมเอไพรออรีทีไอดีมีส่วนเพิ่มขึ้นมาจากอัลกอริทึมเอไพรออรีคือ อัลกอริทึมนี้จะมีการอ่านข้อมูลจากทรานแซคชันเพียงครั้งแรกของการสร้างแคนดิเดทไอเท็มเซตเท่านั้น หลังจากนั้นอัลกอริทึมจะทำการสร้างแคนดิเดทไอเท็มเซตอีกแบบหนึ่ง (C_k) เพื่อเก็บข้อมูลจากทรานแซคชันแทน และขนาดของ C_k นี้จะมีขนาดเล็กลงเรื่อย ๆ ตาม C_k จึงทำให้ใช้เวลาในการอ่านข้อมูลน้อยลงกว่าการอ่านข้อมูลจากทรานแซคชัน

การทำงานของอัลกอริทึมเอไพรออรีแบ่งออกเป็น 3 ช่วง คือ การสร้างไอเท็มเซตที่ปรากฏบ่อย (รูปที่ 2.11) โดยมีการเรียกใช้การสร้างแคนดิเดทไอเท็มเซต (รูปที่ 2.12) และสุดท้ายเป็นการสร้างกฎความสัมพันธ์ (รูปที่ 2.13)

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

Output: L, frequent itemsets in D.

Method:

L_1 = find_frquent_1-itemsets(D);

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

    C_k = apriori-gen(L_{k-1}, min_sup);

    for each transactions t $\in$ D do begin

        C_t = subset (C_k, t); t

        for each candidates c $\in$ C_t do

            c.count++;

        end

    L_k = {c $\in$ C_t | c.count $\geq$ minsup}

    end

end

return $\bigcup_k L_k$;

รูปที่ 2.11 การสร้างไอเท็มเซตที่ปรากฏบ่อย

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

for each itemset l_1 ∈ L_{k-1}
        for each itemset l_2 ∈ L_{k-1}
                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 {
                        c = l_1 x l_2 ;
                        if has_infrequent_subset(c, L_{k-1}) then
                           delete c;
                        else add c to C_k;
                }
        end
end
return C_{k;}

Procedure has_infrequent_subset(c : candidate k-itemset; L_{k-1}: frequent (k-1)-itemsets);
for each (k-1)-subset s of c
        if s ∉ L_{k-1} then
                return TRUE;
        return FALSE;
end
```

รูปที่ 2.12 การสร้างแคนดิเดทไอเท็มเซต

ในส่วนการสร้างแคนดิเดทไอเท็มเซตแบ่งการทำงานออกเป็น 2 ช่วงย่อย ๆ คือ

1) ขั้นตอนการรวม (The join step) ในการสร้างแคนดิเดทไอเท็มเซตที่ k จะนำไอเท็มเซตที่ปรากฏบ่อยที่ k – 1 มารวมเข้ากับตัวมันเอง

2) ขั้นตอนการตัด (The prune step) C_k จะเป็นซุปเปอร์เซตของ L_k โดยที่สมาชิกของ C_k อาจจะปรากฏขึ้นบ่อยหรือไม่ก็ได้ แต่สมาชิกใน L_k จะต้องอยู่ใน C_k ในการอ่านข้อมูลจากฐานข้อมูลแต่ละครั้งจะถูกกำหนดให้มีการนับแต่ละแคนดิเดทใน C_k แคนดิเดทจะเป็น L_k ได้ก็ต่อเมื่อมีการปรากฏบ่อยมากกว่าหรือเท่ากับค่าสนับสนุนขั้นต่ำ ส่วนแคนดิเดทที่มีการปรากฏบ่อยน้อยกว่าค่าสนับสนุนขั้นต่ำจะถูกตัดทิ้งไป อย่างไรก็ตามวิธีดังกล่าวอาจทำให้ C_k มีขนาดใหญ่ขึ้นและอาจเป็นงานหนักในการคำนวน ดังนั้นอัลกอริทึมเอไพรออรีจึงมีคุณสมบัติอีกประการหนึ่ง คือ ถ้าทุก ๆ สับเซตของ C_k ไม่ปรากฏอยู่ใน L_k-1 แล้วสามารถตัดแคนดิเดทตัวนั้นทิ้งออกไปได้ ตัวอย่างเช่น ถ้าให้ L_3 มีค่าเท่ากับ {{1, 2, 3}, {1, 2, 4}, {1, 3, 4}, {1, 3, 5}, {2, 3, 4}} เมื่อนำ L_3 ผ่านขั้นตอนการรวมจะได้ {{1, 2 , 3, 4}, {1, 3, 4, 5}} เมื่อเข้าสู่ขั้นตอนการตัดสามารถตัด {1, 3, 4, 5} ทิ้งได้เพราะไอเท็มเซต {1, 4, 5} ไม่ปรากฏอยู่ใน L_3 ดังนั้นเราจะได้ C_4 เพียงไอเท็มเซตเดียวคือ {1, 2 , 3, 4}

```
forall large k – itemsets lk, k ≥ 2 do begin
        H_1 = {consequents of rules derived from l_k with one item in the consequent};
        call ap-genrules(l_k, H_1);
end

Procedure ap-genrules(l_k: large k-itemsets, Hm: set of m-item consequents)
if(k > m+1) then begin
        H_m+1 = apriori-gen(Hm);
        forall h_m+1 ∈ H_m+1 do begin
                conf = support(l_k)/support(l_k – h_m+1);
                if(conf ≥ minconf) then
                        output the rule(l_k – h_m+1) ⇒ h_m+1 with confidence = conf and support
                        = support(l_k);
                else
                        delete h_m+1 from H_m+1;
        end
        call ap-genrules(l_k, H_m+1);
end
```

รูปที่ 2.13 การสร้างกฎความสัมพันธ์

บทที่ 3
ระเบียบวิธีวิจัย

งานวิจัยนี้ มีจุดมุ่งหมายที่จะพัฒนาวิธีการค้นหากฎข้อบังคับเพื่อนำมาใช้ในการปรับปรุงประสิทธิภาพข้อคำถามในรูปแบบเชิงความหมาย (Semantic query optimization) โดยโปรแกรมที่พัฒนาขึ้นจะใช้ภาษา JAVA ในการพัฒนา และนำเอาเทคโนโลยีทางด้านเหมืองข้อมูลที่เริ่มเป็นที่รู้จักกันแพร่หลายนี้มาประยุกต์ใช้ในการค้นหารูปแบบของข้อมูล เพื่อนำมาสร้างเป็นกฎข้อบังคับ ซึ่งการทำเหมืองข้อมูลนี้มีอัลกอริทึมมากมายที่ใช้ในการค้นหารูปแบบของข้อมูลโดยอัลกอริทึมที่ใช้ในการค้นหารูปแบบของข้อมูลเพื่อนำมาใช้ในการสร้างเป็นกฎข้อบังคับนี้คืออัลกอริทึมเอไพรออรี (Apriori) (Agrawal and Srikant., 1994) ซึ่งเป็นอัลกอริทึมที่ใช้ในการทำเหมืองข้อมูลประเภทการค้นหากฎความสัมพันธ์ โดยมุ่งเน้นให้อัลกอริทึมสามารถสร้างกฎความสัมพันธ์ที่ได้ให้อยู่ในรูปอย่างง่ายต่อการนำไปใช้ในการสร้างกฎข้อบังคับ กฎความสัมพันธ์ที่ได้จะต้องมีค่าความเชื่อมั่น 100 เปอร์เซ็นต์เท่านั้น รายละเอียดในเนื้อหาบทนี้ประกอบด้วยระเบียบวิธีวิจัย ปรากฏอยู่ในหัวข้อ 3.1 โปรแกรม SQOARI พัฒนาขึ้นโดยมีจุดมุ่งหมายที่จะใช้ในการค้นหากฎข้อบังคับจากความสัมพันธ์ของข้อมูล และทำการปรับปรุงข้อคำถามด้วยกฎข้อบังคับที่ได้ เพื่อนำเวลาที่ได้มาเปรียบเทียบระหว่างข้อคำถามเดิม และข้อคำถามที่มีการปรับปรุงแล้ว ปรากฏอยู่ในหัวข้อ 3.2 รายละเอียดข้อมูลที่ใช้ในการทดสอบโปรแกรม ปรากฏอยู่ในหัวข้อที่ 3.3 และหัวข้อ 3.4 เป็นรายละเอียดวิธีการเปรียบเทียบผลลัพธ์ที่ได้จากการทดสอบ

3.1 ขั้นตอนการวิจัย

การค้นคว้าวิจัยแบ่งออกเป็น ขั้นตอนดังนี้

1. การศึกษา และรวบรวมงานวิจัยที่เกี่ยวข้อง
2. ศึกษาการทำงานของอัลกอริทึมเอไพรออรี ศึกษาลักษณะของข้อมูลที่ใช้ในการค้นหาความสัมพันธ์ของข้อมูล และรูปแบบของกฎความสัมพันธ์ที่ได้จากการค้นหา จากนั้นศึกษาโปรแกรม WEKA ซึ่งเป็นโปรแกรมสำเร็จรูปที่เปิดเผยซอร์สโค้ด (Open-source environment) ซึ่งเป็นโปรแกรมสำหรับการทำเหมืองข้อมูลพัฒนาโดย Frank, Hall, Holmes, Martin, Mayo, Pfahringer, Smith and Witten (2005) (http://www.cs.waikato.ac.nz/ml/weka/) ซึ่งใช้เป็น

โปรแกรมสำหรับค้นหากฎความสัมพันธ์จากข้อมูลทดสอบในงานวิจัยครั้งนี้ และนำผลที่ได้ไปเปรียบเทียบความถูกต้องกับกฎความสัมพันธ์ที่ได้จากโปรแกรม SQOARI ที่พัฒนาขึ้น

3. ศึกษาเกี่ยวกับลักษณะการทำงานของกระบวนการปรับปรุงประสิทธิภาพข้อคำถามเชิงความหมาย รวมทั้งออกแบบกฎข้อบังคับที่จะนำมาใช้ในการปรับปรุงข้อคำถาม ซึ่งกฎข้อบังคับเหล่านี้จะได้มาจากการค้นหาความสัมพันธ์ของข้อมูลที่มีอยู่ ดังนั้นรูปแบบของกฎข้อบังคับจะต้องอยู่ในลักษณะที่เหมาะสมกับงานปรับปรุงประสิทธิภาพข้อคำถามเชิงความหมายรูปแบบของการจัดเก็บ และการเรียกกฎข้อบังคับที่ได้มาใช้

4. ออกแบบอัลกอริทึมสำหรับเรียกใช้กฎข้อบังคับที่ได้มาเปรียบเทียบกับข้อคำถามที่รับเข้ามาทำการประมวลผลเพื่อทำการจัดรูปแบบของเงื่อนไขข้อคำถาม และตัดเงื่อนไขที่ไม่จำเป็นออกไป

5. ออกแบบอัลกอริทึมเอไพรออรี ในรูปแบบของ JAVA และปรับปรุงกฎความสัมพันธ์ที่ได้มาเพื่อนำมาประยุกต์ใช้กับการปรับปรุงประสิทธิภาพข้อคำถามเชิงความหมาย โดยจำกัดการหาไอเท็มเซตที่ปรากฏบ่อย (Large itemset หรือ Frequent itemset) เพียงแค่สองไอเท็มเซต (2-Large itemset) เท่านั้น และกำหนดค่าความเชื่อมั่น (Confidence) ไว้เท่ากับ 100 เปอร์เซ็นต์เท่านั้น เพราะกฎความสัมพันธ์ที่ได้จากการหาความสัมพันธ์จะต้องมีความถูกต้องที่สุด

6. รวบรวมข้อมูลที่จะใช้ในการทดสอบจากแหล่งข้อมูลของมหาวิทยาลัยแห่งรัฐแคลิฟอร์เนีย เมืองเออร์ไวน์ (http://archive.ics.uci.edu/ml/index.html)

7. นำข้อมูลที่ได้มาบันทึกลงยังฐานข้อมูลเพื่อเตรียมใช้ในการทดสอบ

8. พัฒนาอัลกอริทึมที่ได้ออกแบบไว้ เพื่อใช้ในการทดสอบสมมติฐานที่ตั้งไว้

9. ทำการทดสอบอัลกอริทึมกับชุดข้อมูลที่รวบรวมไว้ โดยทดสอบกับระบบฐานข้อมูลทั้งหมดสองระบบด้วยกัน คือ Microsoft SQL Server 2000 และ Oracle 10g Express โดยทำการเปรียบเทียบเวลาที่ใช้ในการประมวลผลข้อคำถามเดิม กับข้อคำถามที่ได้ปรับปรุงแล้วกับระบบฐานข้อมูลทั้สองระบบ คอมพิวเตอร์ที่ใช้ในการทดสอบเป็นคอมพิวเตอร์ Desktop CPU Pentium IV ความเร็ว 3.2 GHz หน่วยความจำหลัก 512 MB ฮาร์ดดิสก์ความจุ 80 GB

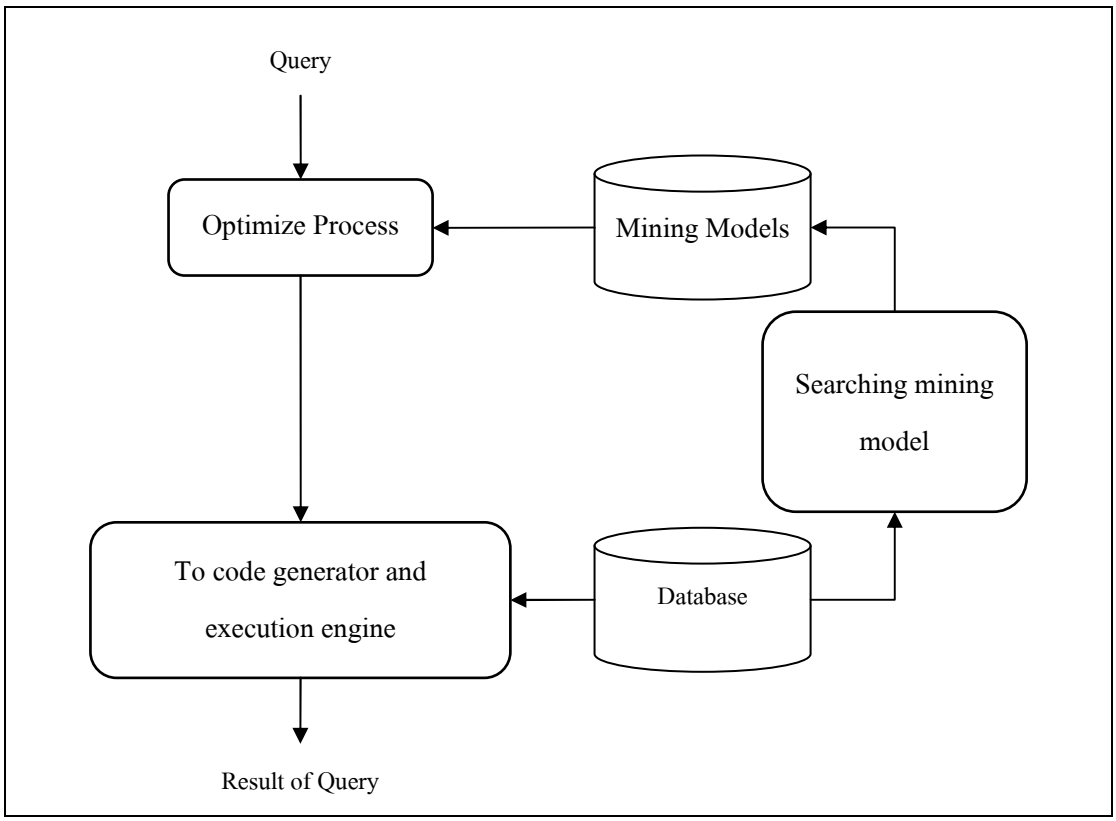
10.บันทึกผล และเสนอแนะรูปแบบของข้อมูลที่สามารถใช้ในการปรับปรุงประสิทธิภาพข้อคำถามเชิงความหมายด้วยกฎข้อบังคับที่ได้นี้ รวมทั้งข้อมูลที่จะนำมาใช้ในการค้นหากฎความสัมพันธ์เพื่อนำมาสร้างเป็นกฎข้อบังคับ และลักษณะของข้อคำถามที่เหมาะสมในการปรับปรุงประสิทธิภาพ เพื่อให้ได้ประสิทธิภาพมากที่สุด

3.2 โปรแกรม SQOARI เพื่อการค้นหากฎความสัมพันธ์ และปรับปรุงข้อคำถาม

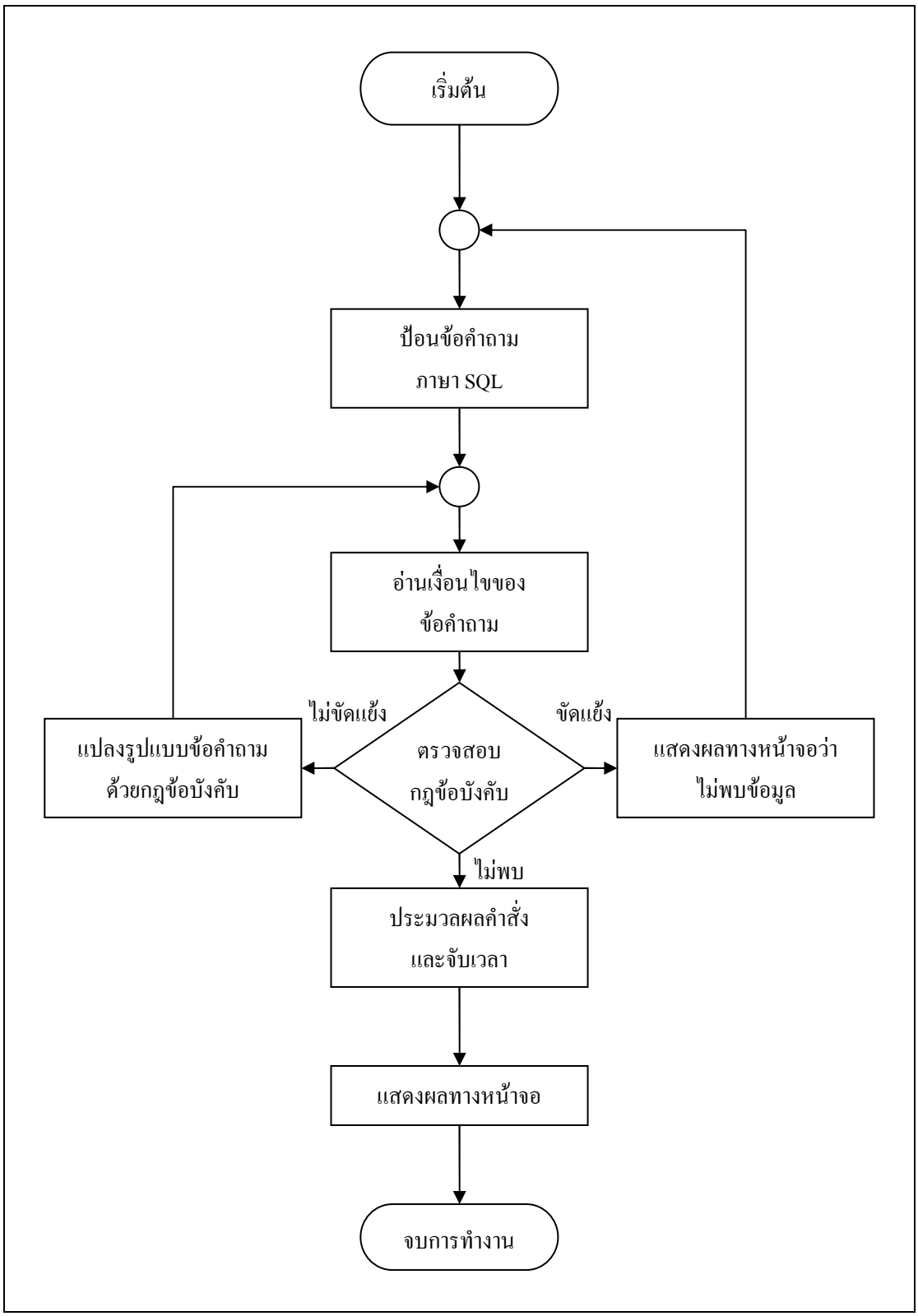
เนื้อหาในส่วนนี้กล่าวถึงการนำแนวคิดต่าง ๆ และอัลกอริทึมที่ได้ออกแบบไว้ มาพัฒนาเป็นโปรแกรม SQOARI ในรูปแบบของ JAVA โดยเป็นการนำเอาความรู้เดิมที่มีอยู่แล้วมาประยุกต์ใช้เข้าด้วยกัน นั่นคือ อัลกอริทึมเอไพรออรีซึ่งเป็นอัลกอริทึมหนึ่งในการทำเหมืองข้อมูลมาประยุกต์ใช้ให้ทำงานร่วมกันกับการปรับปรุงประสิทธิภาพข้อคำถามแบบเชิงความหมายเพื่อให้เกิดความรู้ใหม่ โดยโปรแกรมที่พัฒนาขึ้นมานี้มีจุดประสงค์เพียงเพื่อใช้ในการทดสอบสมมติฐานที่ตั้งไว้จึงเป็นเพียงโปรแกรมต้นแบบ

3.2.1 กระบวนการทำงานของโปรแกรม SQOARI

ในกระบวนการทำงานของโปรแกรม SQOARI แบ่งการทำงานออกเป็น 2 ส่วนหลักด้วยกันคือ การปรับปรุงประสิทธิภาพข้อคำถามเชิงความหมายจากกฎข้อบังคับที่ได้จากการหาความสัมพันธ์ของข้อมูล และกระบวนการค้นหากฎข้อบังคับจากความสัมพันธ์ของข้อมูล โดยในการปรับปรุงประสิทธิภาพข้อคำถามเชิงความหมายนี้จะทำการตรวจสอบเงื่อนไขของข้อคำถามที่รับเข้ามาจากผู้ใช้เปรียบเทียบกับกฎข้อบังคับที่เก็บไว้ซึ่งโปรแกรมจะทำการตรวจสอบจากชื่อของตารางที่เกี่ยวข้องกับข้อมูลที่ต้องการค้นหาเป็นสิ่งแรกเพื่อจะทำการเลือกกฎข้อบังคับที่ได้จากตารางนั้นขึ้นมา จากนั้นโปรแกรมจึงจะทำการตรวจสอบส่วนของชื่อของคอลัมน์ว่ามีกฎข้อบังคับใดที่มีค่าตรงกับคอลัมน์ที่ต้องการตรวจสอบหรือไม่ ถ้ามีโปรแกรมจะทำการตรวจสอบกับค่าที่อยู่ในคอลัมน์นั้นต่อไป และโปรแกรมจะทำการตรวจสอบแบบนี้ไปเรื่อย ๆ จนกระทั่งครบทุกเงื่อนไขที่ระบุมาซึ่งรูปแบบของเงื่อนไข และลักษณะของกฎข้อบังคับจะกล่าวถึงในหัวข้อถัดไป ในส่วนของการค้นหากฎข้อบังคับ จะได้มาจากการค้นหาความสัมพันธ์ของข้อมูลในฐานข้อมูล สำหรับการค้นหาความสัมพันธ์ในงานวิจัยครั้งนี้ มีการเลือกใช้อัลกอริทึมเอไพรออรีในการค้นหากฎข้อบังคับ ดังนั้นกฎข้อบังคับที่ได้จะมีลักษณะคอลัมน์เหตุไปสู่คอลัมน์ผล (IF – THEN) ซึ่งกฎข้อบังคับดังกล่าวจะถูกเก็บอยู่ในระบบฐานข้อมูลเช่นกัน ภาพรวมการทำงานของโปรแกรมสามารถแสดงได้ดังรูปที่ 3.1 และมีขั้นตอนการทำงานของโปรแกรม แสดงได้ดังรูปที่ 3.2



รูปที่ 3.1 ภาพรวมการทำงานของโปรแกรม SQOARI



รูปที่ 3.2 ขั้นตอนการทำงานของโปรแกรม SQOARI

3.2.2 ส่วนการคัดเลือกกฎข้อบังคับจากการค้นหาความสัมพันธ์ของข้อมูล

ในการนำกฎความสัมพันธ์ที่ได้จากการทำเหมืองข้อมูล มาใช้ร่วมกับการปรับปรุงประสิทธิภาพข้อคำถามเชิงความหมายนี้จำเป็นต้องมีการจัดรูปแบบของกฎความสัมพันธ์ที่ได้ ให้อยู่ในรูปแบบที่เหมาะสมกับการนำมาใช้งาน โดยส่วนใหญ่การสร้างกฎข้อบังคับเพื่อนำมาใช้ในการปรับปรุงประสิทธิภาพข้อคำถาม จะถูกกำหนดโดยผู้ดูแลระบบฐานข้อมูลเอง ดังนั้นกฎข้อบังคับที่ได้อาจไม่ครอบคลุมกับข้อมูลทั้งหมดที่มีอยู่ในระบบฐานข้อมูล ในงานวิจัยนี้จึงมีการนำเทคโนโลยีที่มีอยู่แล้วมาประยุกต์ใช้รวมกันเพื่อให้เกิดประสิทธิภาพสูงสุด เพื่อเป็นแนวทางในการพัฒนาระบบฐานข้อมูลต่อไป ซึ่งการค้นหาความสัมพันธ์จากข้อมูลนี้ ผลลัพธ์ที่ได้ออกมาจะอยู่ในรูปแบบของคอลัมน์เหตุไปสู่คอลัมน์ผล ตามรูปแบบของการค้นหาด้วยอัลกอริทึมเอไพรออรี ดังนี้

If column_M = ‘value_M’ Then column_N = ‘value_N’

จากกฎความสัมพันธ์ที่ได้จากการค้นหา เราจะนำกฎเหล่านี้มาปรับให้อยู่ในรูปแบบที่ไม่ซ้ำซ้อน โดยมีการกำหนดคอลัมน์ที่เป็นเหตุให้เหลือเพียงหนึ่งคอลัมน์เท่านั้น และคอลัมน์ที่เป็นผลให้เหลือเพียงหนึ่งคอลัมน์เช่นกัน เนื่องจากกฎความสัมพันธ์จะมีการเชื่อมค่าของแต่ละคอลัมน์ด้วย (AND) แต่ภายในเงื่อนไขของข้อคำถามนั้นจะประกอบด้วย (OR) และ (AND) ดังนั้นการนำกฎความสัมพันธ์ที่มีลักษณะของคอลัมน์ที่เป็นเหตุ และคอลัมน์ที่เป็นผล จะประกอบกันด้วยจำนวนคอลัมน์หลายคอลัมน์นั้นอาจจะไม่เกิดประโยชน์มากนัก ซึ่งทำให้เสียเวลาในการนำกฎข้อบังคับที่มีจำนวนมากเกินความจำเป็นมาเปรียบเทียบกับเงื่อนไขของข้อคำถาม และกฎความสัมพันธ์ที่ได้จากอัลกอริทึมเอไพรออรีนี้ จะประกอบไปด้วยกฎความสัมพันธ์ที่มีจำนวนของคอลัมน์ที่เป็นเหตุตั้งแต่หนึ่งคอลัมน์ จนถึงการประกอบกันจากคอลัมน์หลายคอลัมน์ และคอลัมน์ที่เป็นผลตั้งแต่หนึ่งคอลัมน์ไปจนถึงการประกอบกันจากหลายคอลัมน์เช่นกัน จึงทำให้มีกฎความสัมพันธ์ที่ได้มีความซ้ำซ้อนกันเอง นั่นคือการนำกฎความสัมพันธ์ที่ได้จากไอเท็มเซตที่ปรากฏบ่อยในชั้นที่ต่ำกว่ามาประกอบกันเป็นกฎความสัมพันธ์ในชั้นถัดไป เช่นถ้ามีข้อมูลดังแสดงในตารางที่ 3.1 ซึ่งเป็นข้อมูลตัวอย่างประกอบไปด้วยจำนวนคอลัมน์ทั้งหมด 3 คอลัมน์ด้วยกัน คือ ข้อมูลที่อยู่ (Address), ข้อมูลระดับบัตรเครดิต (Credit) และข้อมูลสถานะการสมรส (Marital) โดยความหมายของข้อมูลในแต่ละคอลัมน์มีดังนี้ คอลัมน์ข้อมูลที่อยู่ Nakhonratcahsima คือ จังหวัดนครราชสีมา, Bangkok คือ จังหวัดกรุงเทพมหานคร คอลัมน์ระดับบัตรเครดิต Silver คือ ระดับเงิน, Gold คือ ระดับทอง คอลัมน์ข้อมูลสถานะการสมรส S คือ โสด, M คือ สมรสแล้ว

ตารางที่ 3.1 ตัวอย่างข้อมูลใช้ในการทดสอบกฎความสัมพันธ์ที่เกิดความซ้ำซ้อน

ADDRESS	CREDIT	MARITAL
Nakhonratchasima	Silver	S
Bangkok	Gold	M
Bangkok	Gold	M
Bangkok	Gold	M
Nakhonratchasima	Silver	S

จากข้อมูลข้างต้นเมื่อนำมาค้นหาความสัมพันธ์ โดยกำหนดค่าความเชื่อมั่นให้มีค่าเท่ากับ 100 เปอร์เซ็นต์ และค่าสนับสนุนเท่ากับ 50 เปอร์เซ็นต์ จะได้กฎความสัมพันธ์ออกมาทั้งหมด 12 ข้อด้วยกันดังรูปที่ 3.3

1. credit=Gold 3 => city=Bangkok 3 conf(1)
2. city=Bangkok 3 => credit=Gold 3 conf(1)
3. marital=M 3 => city=Bangkok 3 conf(1)
4. city=Bangkok 3 => marital=M 3 conf(1)
5. marital=M 3 => credit=Gold 3 conf(1)
6. credit=Gold 3 => marital=M 3 conf(1)
7. credit=Gold marital=M 3 => city=Bangkok 3 conf(1)
8. city=Bangkok marital=M 3 => credit=Gold 3 conf(1)
9. city=Bangkok credit=Gold 3 => marital=M 3 conf(1)
10. marital=M 3 => city=Bangkok credit=Gold 3 conf(1)
11. credit=Gold 3 => city=Bangkok marital=M 3 conf(1)
12. city=Bangkok 3 => credit=Gold marital=M 3 conf(1)

รูปที่ 3.3 แสดงผลการค้นหากฎความสัมพันธ์ในกรณีเกิดความซ้ำซ้อนกันเอง

เมื่อวิเคราะห์กฎที่ได้จากการค้นหาความสัมพันธ์แล้วพบว่ากฎความสัมพันธ์ในข้อที่ 7 นั้นเกิดจากการคอลัมน์ที่เป็นเหตุจากกฎความสัมพันธ์ในข้อที่ 1 และคอลัมน์ที่เป็นเหุตจากกฎความสัมพันธ์ในข้อที่ 3 มาเชื่อมเข้าด้วยกันด้วย AND (ในรูปที่ 3.4 ข้อความในบรรทัดสุดท้ายหมายถึง credit = gold AND marital = M แต่ตัวกระทำ AND ถูกละไว้และตัวเลข 3 หมายถึงมีข้อมูล 3 รายการที่ตรงกับเงื่อนไขนี้) ดังนี้

credit = Gold 3 => city = Bangkok 3 conf(1)

marital = M 3 => city = Bangkok 3 conf(1)

credit = Gold marital = M 3 => city = Bangkok 3 conf(1)

รูปที่ 3.4 แสดงกฎความสัมพันธ์ที่มีความซ้ำซ้อนกันเอง

ดังนั้น เพื่อความเหมาะสมกับการนำกฎความสัมพันธ์ไปใช้ในการปรับปรุงประสิทธิภาพข้อคำถาม จึงมีการจัดรูปแบบให้อยู่ในลักษณะดังนี้

If column_M = ‘value_M’ Then column_N = ‘value_N’

เมื่อได้กฎความสัมพันธ์ตามที่ต้องการแล้วโปรแกรมจะจัดเก็บกฎความสัมพันธ์ดังกล่าวลงในระบบฐานข้อมูล ซึ่งเมื่อมีการจัดเก็บกฎความสัมพันธ์นี้จะถูกเรียกว่ากฎข้อบังคับ (Semantic constraint) โดยในระบบฐานข้อมูลจะมีตารางในการจัดเก็บกฎข้อบังคับจำนวนหนึ่งตาราง โดยมีโครงสร้างของตาราง ดังตารางที่ 3.2

ตารางที่ 3.2 รายละเอียดตารางเก็บกฎข้อบังคับ

ชื่อคอลัมน์	**คำอธิบาย**
Tname_Cause	ชื่อตารางของคอลัมน์ที่เป็นเหตุ
Column_Cause	ชื่อคอลัมน์ที่เป็นเหตุ
Attribute_Cause	ค่าของคอลัมน์ที่เป็นเหตุ
Tname_Result	ชื่อตารางของคอลัมน์ที่เป็นผล
Column_Result	ชื่อคอลัมน์ที่เป็นผล
Attribute_Result	ค่าของคอลัมน์ที่เป็นผล

ดังนั้น เมื่อทำการค้นหาความสัมพันธ์ของข้อมูลจากตารางของลูกค้า ปรากฏว่าได้กฎความสัมพันธ์ทั้งหมด ดังต่อไปนี้

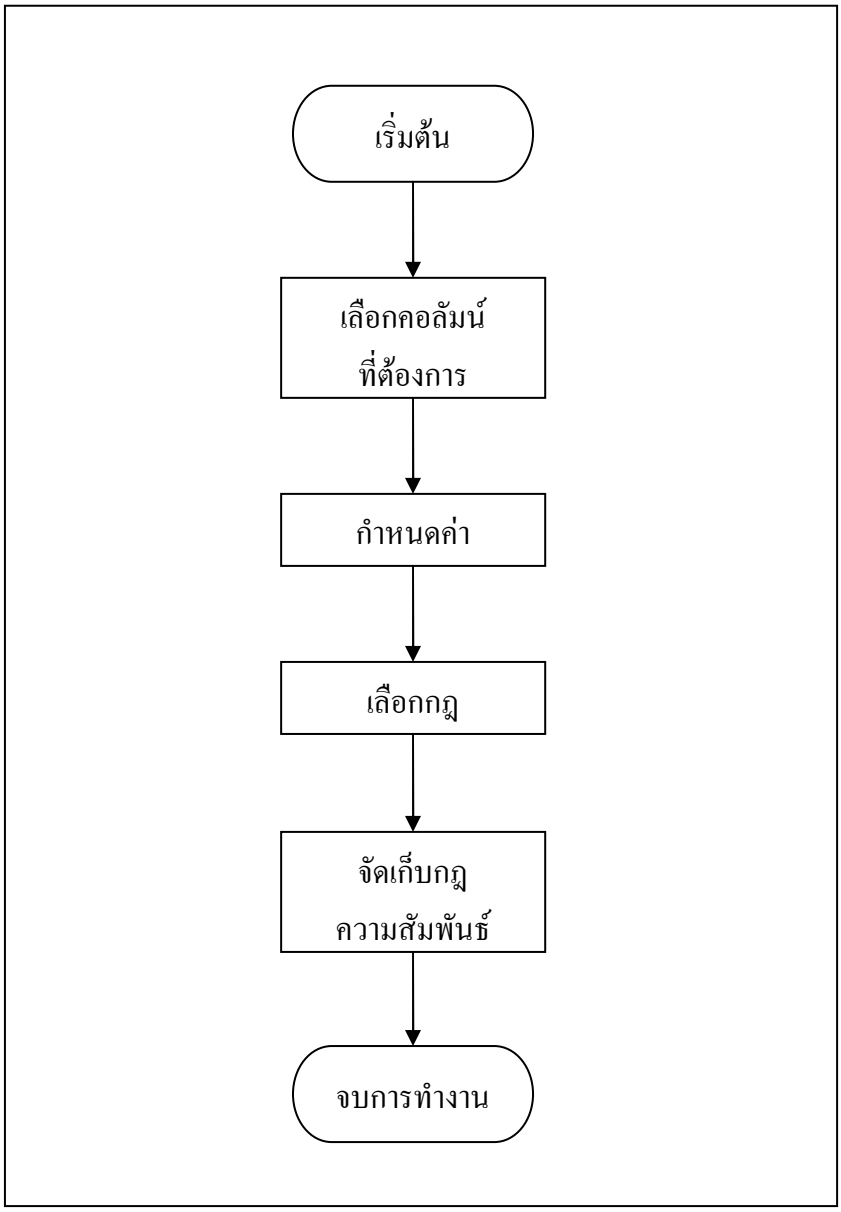
1. credit=Gold 3 => city=Bangkok 3 conf(1)
2. city=Bangkok 3 => credit=Gold 3 conf(1)
3. marital=M 3 => city=Bangkok 3 conf(1)
4. city=Bangkok 3 => marital=M 3 conf(1)
5. marital=M 3 => credit=Gold 3 conf(1)
6. credit=Gold 3 => marital=M 3 conf(1)

หลังจากได้กฎความสัมพันธ์ดังกล่าวแล้วนั้น โปรแกรมจะทำการจัดเก็บข้อมูลกฎความสัมพันธ์ที่ได้ลงยังตารางที่ใช้ในการจัดเก็บกฎข้อบังคับในระบบฐานข้อมูล โดยมีลักษณะดังตารางที่ 3.3

ตารางที่ 3.3 ลักษณะการจัดเก็บกฎความสัมพันธ์ลงยังตารางจัดเก็บกฎข้อบังคับ

Tname_ Cause	**Column_ Cause**	**Attribute_ Cause**	**Tname_ Result**	**Column_ Result**	**Attribute_ Result**
customer	credit	Gold	customer	city	Bangkok
customer	city	Bangkok	customer	credit	Gold
customer	marital	M	customer	city	Bangkok
customer	city	Bangkok	customer	marital	M
customer	marital	M	customer	credit	Gold
customer	credit	Gold	customer	marital	M

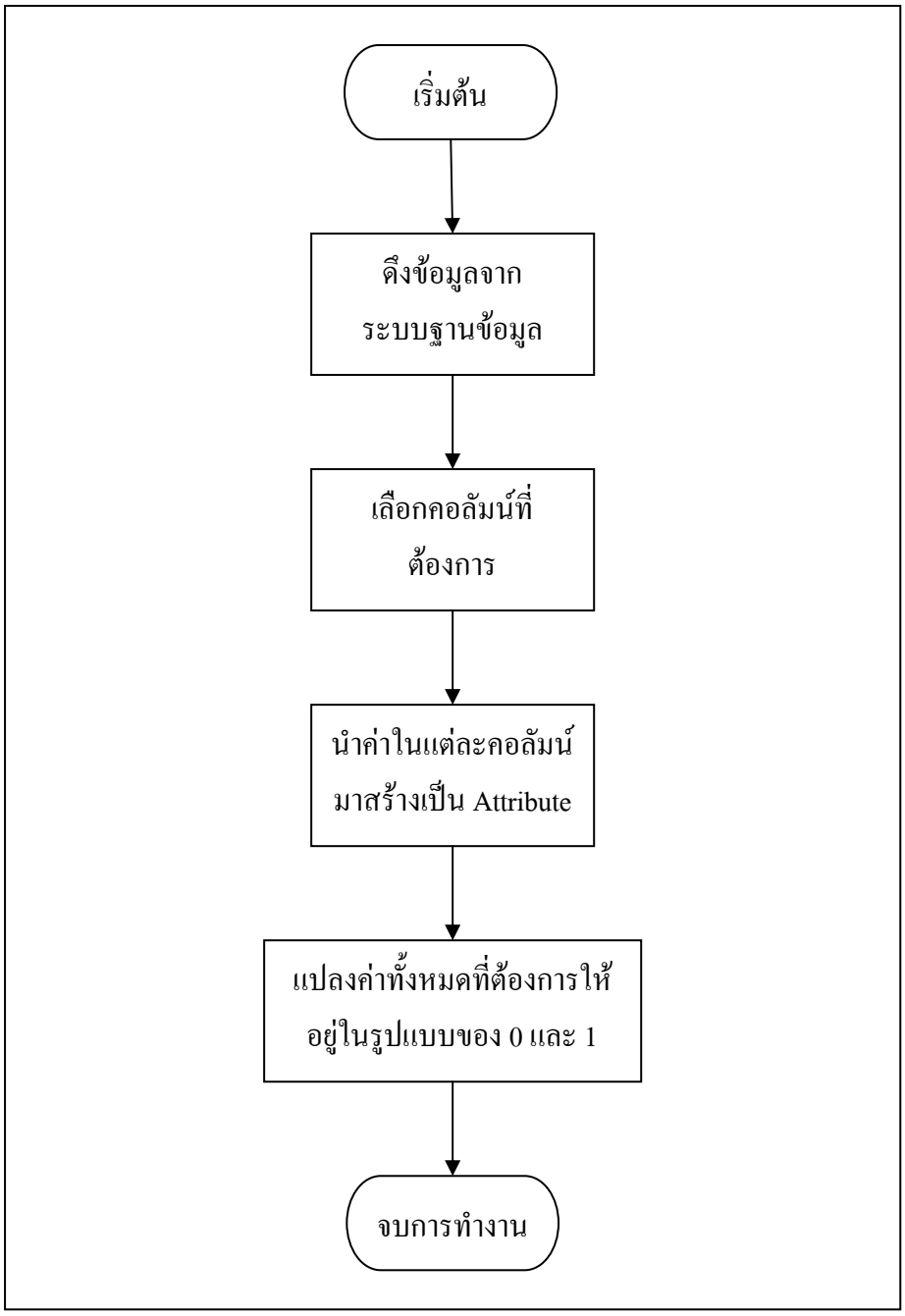
ในขั้นตอนการค้นหากฎความสัมพันธ์ เพื่อนำไปใช้เป็นกฎข้อบังคับในการปรับปรุงข้อคำถามเชิงความหมายมีขั้นตอนการทำงาน ดังรูปที่ 3.5



รูปที่ 3.5 ขั้นตอนการค้นหากฎความสัมพันธ์

3.2.3 การแปลงข้อมูลจากฐานข้อมูลให้เป็นข้อมูลสำหรับการค้นหากฎความสัมพันธ์

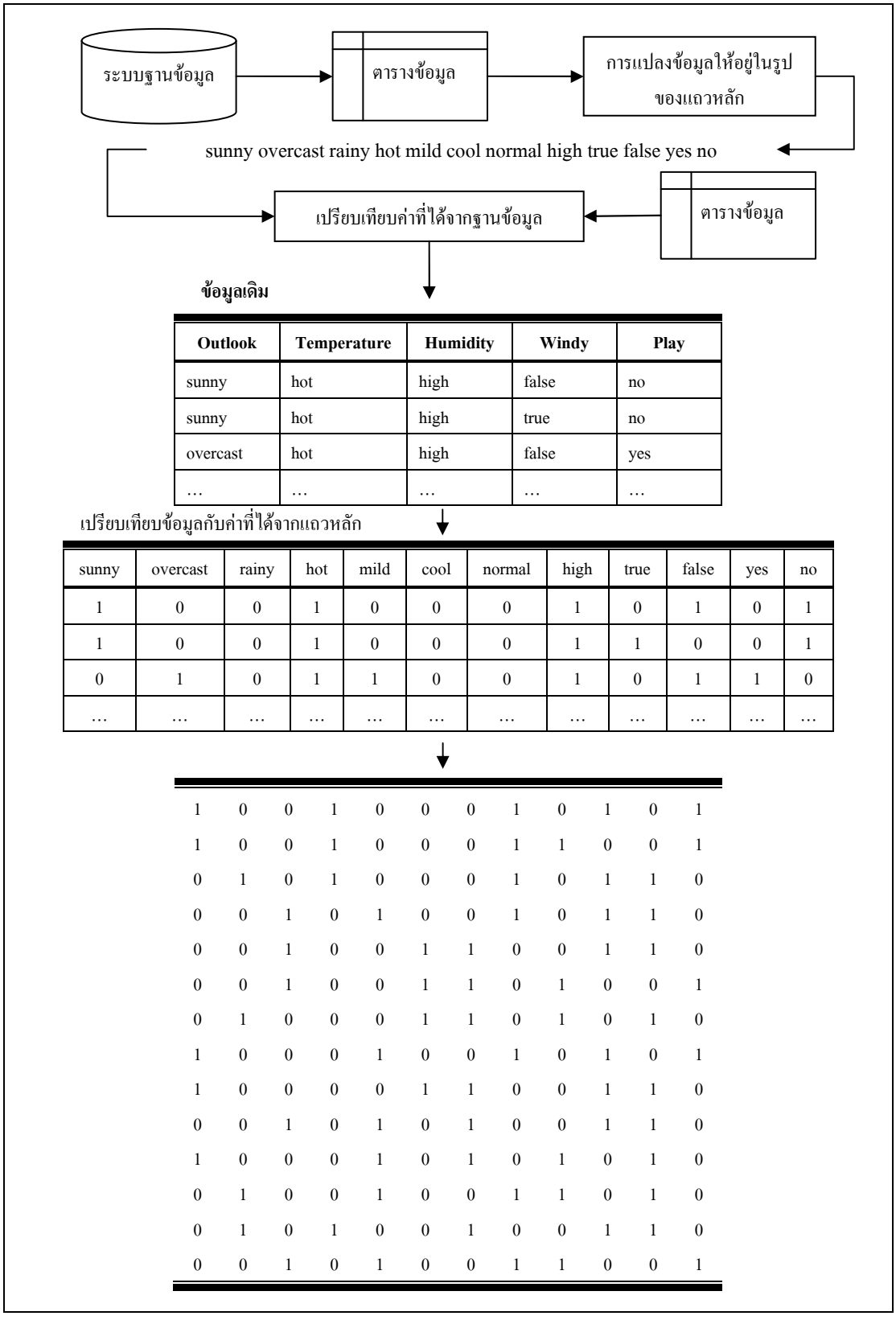
ในส่วนนี้เป็นการแปลงรูปแบบของข้อมูลที่อยู่ในระบบฐานข้อมูล เพื่อให้อยู่ในรูปแบบที่เหมาะสมกับอัลกอริทึมที่ได้ออกแบบไว้ ซึ่งข้อมูลที่อยู่ในระบบฐานข้อมูลจะเก็บอยู่ในรูปแบบของสัญลักษณ์ หรือตัวเลข และในแต่ละคอลัมน์จะประกอบไปด้วยค่าที่แตกต่างกันออกไป ในการแปลงรูปแบบของข้อมูล ในลำดับแรกจะหาค่าที่เป็นไปได้ทั้งหมดที่มีอยู่ในแต่ละคอลัมน์เพื่อนำมาเรียงใหม่ให้อยู่ในรูปแบบของแถวเดียวเพื่อใช้เป็นแถวหลักในการเปรียบเทียบ และการเปลี่ยนแปลงค่า จากนั้นทำการดึงข้อมูลแต่ละแถวขึ้นมาจากฐานข้อมูลเพื่อเปรียบเทียบกับค่าที่ได้เรียงไว้ในแถวหลัก โดยทำการเปรียบเทียบกับข้อมูลแต่ละคอลัมน์ที่ดึงขึ้นมา และทำการเปลี่ยนรูปแบบการจัดเก็บค่าใหม่ โดยถ้าค่าในคอลัมน์ในแถวที่ดึงขึ้นมามีค่าตรงกับค่าที่ปรากฏอยู่ในแถวหลัก จะกำหนดค่าให้มีค่าเท่ากับหนึ่ง และถ้าค่าในแถวที่ดึงขึ้นมาไม่พบค่าที่ปรากฏอยู่ในแถวหลักที่เรียงขึ้นใหม่จะกำหนดค่าให้เท่ากับศูนย์ ดังนั้นข้อมูลที่จะนำไปใช้ในการค้นหาความสัมพันธ์จะปรากฏเพียงค่าหนึ่งและศูนย์เท่านั้น และจำนวนของคอลัมน์ใหม่จะมีจำนวนเท่ากับจำนวนของแถวหลักที่ได้ รูปที่ 3.6 แสดงขั้นตอนของการแปลงรูปแบบข้อมูล และตารางที่ 3.4 ข้อมูลสภาพอากาศ ซึ่งจะใช้เพื่อแสดงตัวอย่างการทำงานของการแปลงรูปแบบของข้อมูล



รูปที่ 3.6 ขั้นตอนของการแปลงรูปแบบข้อมูล

ตารางที่ 3.4 ข้อมูลสภาพอากาศสำหรับใช้ในการตัดสินใจในการเล่นกอล์ฟ (ข้อมูลใช้ในการทดสอบโปรแกรม WEKA)

Outlook	**Temperature**	**Humidity**	**Windy**	**Play**
sunny	hot	high	false	no
sunny	hot	high	true	no
overcast	hot	high	false	yes
rainy	mild	high	false	yes
rainy	cool	normal	false	yes
rainy	cool	normal	true	no
overcast	cool	normal	true	yes
sunny	mild	high	false	no
sunny	cool	normal	false	yes
rainy	mild	normal	false	yes
sunny	mild	normal	true	yes
overcast	mild	high	true	yes
overcast	hot	normal	false	yes
rainy	mild	high	true	no



Outlook	Temperature	Humidity	Windy	Play
sunny	hot	high	false	no
sunny	hot	high	true	no
overcast	hot	high	false	yes
…	…	…	…	…

sunny	overcast	rainy	hot	mild	cool	normal	high	true	false	yes	no
1	0	0	1	0	0	0	1	0	1	0	1
1	0	0	1	0	0	0	1	1	0	0	1
0	1	0	1	1	0	0	1	0	1	1	0
…	…	…	…	…	…	…	…	…	…	…	…

1	0	0	1	0	0	0	1	0	1	0	1
1	0	0	1	0	0	0	1	1	0	0	1
0	1	0	1	0	0	0	1	0	1	1	0
0	0	1	0	1	0	0	1	0	1	1	0
0	0	1	0	0	1	1	0	0	1	1	0
0	0	1	0	0	1	1	0	1	0	0	1
0	1	0	0	0	1	1	0	1	0	1	0
1	0	0	0	1	0	0	1	0	1	0	1
1	0	0	0	0	1	1	0	0	1	1	0
0	0	1	0	1	0	1	0	0	1	1	0
1	0	0	0	1	0	1	0	1	0	1	0
0	1	0	0	1	0	0	1	1	0	1	0
0	1	0	1	0	0	1	0	0	1	1	0
0	0	1	0	1	0	0	1	1	0	0	1

รูปที่ 3.7 แสดงตัวอย่างการแปลงข้อมูลเพื่อค้นหากฎความสัมพันธ์

3.2.4 การสร้างแคนดิเดทไอเท็มเซต (Candidate Itemset) และไอเท็มเซตที่ปรากฏบ่อย (Large Itemset)

โดยปกติแล้ว เราจะทำการสร้างแคนดิเดทไอเท็มเซตก่อน (Candidate itemset) จากนั้นไอเท็มเซตใดมีค่าสนับสนุนน้อยกว่าค่าสนับสนุนที่กำหนดก็จะตัดไอเท็มเซตนั้นออก ไม่นำไปสร้างไอเท็มเซตในขั้นถัดไป ซึ่งไอเท็มเซตที่มีค่าสนับสนุนมากกว่าค่าสนับสนุนที่กำหนดนี้ เราจะเรียกว่า ไอเท็มเซตที่ปรากฏบ่อย (Large itemset) และการทำงานของอัลกอริทึมจะทำการวนในลักษณะนี้ไปเรื่อย ๆ จนกระทั่งไม่เหลือเซตของไอเท็มที่จะใช้สร้างในลำดับถัดไป สำหรับการค้นหาความสัมพันธ์ของข้อมูลเพื่อนำกฎความสัมพันธ์ที่ได้มาใช้ในการสร้างเป็นกฎข้อบังคับนี้ จะสิ้นสุดการค้นหาไอเท็มเซตที่ปรากฏบ่อยเพียงสองลำดับด้วยกัน (2-Large itemset : L_2) เนื่องจากในงานวิจัยชิ้นนี้ได้มีการกำหนดลักษณะของกฎข้อบังคับจากการค้นหาความสัมพันธ์ที่ได้ ดังกล่าวมาแล้วในหัวข้อ 3.2.2 โดยลักษณะของกฎข้อบังคับจะปรากฏเพียงคอลัมน์ที่เป็นเหตุหนึ่งคอลัมน์ และคอลัมน์ที่เป็นผลเพียงหนึ่งคอลัมน์เท่านั้น ดังนั้นจึงไม่จำเป็นต้องทำการค้นหาไอเท็มเซตที่ปรากฏบ่อยในขั้นอื่น ๆ ถัดไป การทำงานในขั้นตอนการสร้างแคนดิเดทไอเท็มเซต และการสร้างไอเท็มเซตที่ปรากฏบ่อยจึงมีขั้นตอนในการสร้างที่สั้นลง โดยการสร้างแคนดิเดทไอเท็มเซตในลำดับชั้นแรก (1-Candidate itemset : C_1) จะนำไอเท็มเซตที่ต้องการค้นหาความสัมพันธ์ทั้งหมดมาใช้ในการสร้างแคนดิเดทไอเท็มเซต จากนั้นจึงนับค่าสนับสนุน หากไอเท็มเซตใดที่มีค่าสนับสนุนมากกว่าหรือเท่ากับค่าสนับสนุนที่กำหนดไว้ จะนำไอเท็มเซตดังกล่าวไปสร้างเป็นไอเท็มเซตที่ปรากฏบ่อย และในการสร้างแคนดิเดทไอเท็มเซตในลำดับชั้นถัดไปจะใช้ไอเท็มเซตที่ปรากฏบ่อยในรอบก่อนหน้าเพื่อสร้างแคนดิเดทไอเท็มเซตในลำดับชั้นถัดไป

3.2.5 การสร้างกฎความสัมพันธ์

การสร้างกฎความสัมพันธ์ จะนำข้อมูลไอเท็มเซตที่ปรากฏขึ้นบ่อยจากขั้นตอนข้างต้นมาสร้างให้เป็นกฎความสัมพันธ์ โดยจะนำข้อมูลไอเท็มเซตที่ปรากฏบ่อยในแต่ละชุดมาสร้างกฎความสัมพันธ์ที่เป็นไปได้ โดยตรวจสอบจากค่าความเชื่อมั่น ซึ่งกฎความสัมพันธ์ที่สร้างขึ้นมาจะต้องมีค่าความเชื่อมั่นมากกว่าหรือเท่ากับค่าความเชื่อมั่นที่กำหนดไว้ ดังนี้

$$\text{Confidence } (x \Rightarrow y) = P(x|y) = \frac{Support(x \cup y)}{Support(x)}$$

หมายความว่า ค่าความเชื่อมั่นจะหาได้จากสัดส่วนของค่าสนับสนุนของกฎความสัมพันธ์ที่ต้องการและค่าความสนับสนุนของคอลัมน์ที่เป็นเหตุ เช่นถ้าได้ข้อมูลไอเท็มเซตที่ปรากฏบ่อยดังนี้

1 – Large itemset

1. outlook=sunny support (5)
2. outlook=overcast support (4)
3. outlook=rainy support (5)
4. temperature=hot support (4)
5. temperature=mild support (6)
6. temperature=cool support (4)
7. humidity=high support (7)
8. humidity=normal support (7)
9. windy=TRUE support (6)
10. windy=FALSE support (8)
11. play=yes support (9)
12. play=no support (5)

2 – Large itemset

1.outlook=overcast play=yes support (4)
2. temperature=mild play=yes support (4)

การสร้างกฎความสัมพันธ์ในงานวิจัยชิ้นนี้ใช้ไอเท็มเซตที่ปรากฏบ่อยในลำดับชั้นที่ 2 สร้างกฎความสัมพันธ์ จากข้อมูลข้างต้นเราสามารถสร้างกฎความสัมพันธ์ได้ 4 กฎ ด้วยกันคือ

1. outlook = overcast => play = yes conf(1.0)
2. temperature = mild => play = yes conf(0.67)
3. play = yes => outlook = overcast conf(0.44)
4. play = yes => temperature = mild conf(0.44)

ค่าความเชื่อมั่นสำหรับกฎความสัมพันธ์ข้อที่ 1 คำนวนได้จาก

Confidence = support (outlook=overcast play=yes) / support (outlook=overcast)

= 4 / 4

= 1

ค่าความเชื่อมั่นสำหรับกฎความสัมพันธ์ข้อที่ 2 คำนวนได้จาก

Confidence = support (temperature=mild play=yes) / support (temperature=mild)

= 4 / 6

= 0.67

ในงานวิจัยชิ้นนี้ จำเป็นต้องกำหนดค่าความเชื่อมั่นให้มีค่าเท่ากับหนึ่ง หรือหนึ่งร้อยเปอร์เซ็นต์ เนื่องจากต้องนำกฎความสัมพันธ์ที่ได้มาใช้ในการเปรียบเทียบกับเงื่อนไขของข้อคำถามที่รับเข้ามา เพื่อทำการตัดเงื่อนไขที่ไม่จำเป็นออก ดังนั้นกฎความสัมพันธ์ที่ได้จะต้องมีความถูกต้องมากที่สุดเพื่อให้ข้อคำถามที่ถูกปรับปรุงแล้วสามารถให้คำตอบที่ถูกต้องไม่บิดเบือนไปจากเดิม เช่น ถ้ามีข้อมูลดังตารางที่ 3.4 และกำหนดค่าความเชื่อมั่นมีค่าเพียง 60 เปอร์เซ็นต์ หรือเท่ากับ 0.6 และกำหนดค่าสนันสนุนเท่ากับ 0.15 หรือ 2 Instances ผลจากการค้นหากฎความสัมพันธ์ และคัดเลือกกฎความสัมพันธ์ที่มีคอลัมน์ที่เป็นเหตุเพียงหนึ่งคอลัมน์ และคอลัมน์ที่เป็นผลเพียงหนึ่งคอลัมน์ จะได้กฎความสัมพันธ์ทั้งหมดดังนี้

1. outlook = overcast => play = yes	conf(1)
2. temperature = cool => humidity = normal	conf(1)
3. humidity = normal => play = yes	conf(0.86)
4. play = no => humidity = high	conf(0.8)
5. windy = FALSE => play = yes	conf(0.75)
6. temperature = hot => humidity = high	conf(0.75)
7. temperature = hot => windy = FALSE	conf(0.75)
8. temperature = cool => play = yes	conf(0.75)
9. play = yes => humidity = normal	conf(0.67)
10. play = yes => windy = FALSE	conf(0.67)
11. temperature = mild => humidity = high	conf(0.67)
12. temperature = mild => play = yes	conf(0.67)

จากนั้นนำกฎความสัมพันธ์ที่ได้มาใส่ในเงื่อนไขของข้อคำถาม เพื่อใช้ทดสอบกฎความสัมพันธ์ที่ได้ว่ามีความถูกต้องเพียงใด ซึ่งจากความหมายของกฎความสัมพันธ์บ่งบอกว่าอะไรก็ตามที่ปรากฏอยู่ในส่วนของเหตุแล้ว ผลที่ตามมาจะตรงตามกฎที่ระบุไว้เสมอ นั่นคือ ถ้ากฎความสัมพันธ์บอกว่า outlook = overcast => play = yes หมายความว่า ถ้าทัศนวิสัย (outlook) นั้น

มีเฆฆมาก (overcast) แล้วการตัดสินใจในการเล่นกอล์ฟ (play) นั้นจะตกลงเสมอ (yes) ดังนั้นในการสอบถามข้อมูลจากฐานข้อมูลถ้ากำหนดทั้งเงื่อนไข outlook = overcast และเงื่อนไข play = yes เราสามารถตัดเงื่อนไขที่ไม่จำเป็นออกได้ คือ play = yes เพราะคำตอบในการตัดสินใจในกรณีนี้จะตกลงเสมอ เมื่อนำกฎความสัมพันธ์ที่ตัดคอลัมน์ที่เป็นผลออก กำหนดเป็นเงื่อนไขของข้อคำถามเพื่อทำการค้นหาข้อมูลจากฐานข้อมูลจะต้องได้ผลลัพธ์เท่ากับกฎความสัมพันธ์รูปแบบเต็ม ดังนั้นถ้านำกฎความสัมพันธ์ในข้อที่ 1 ที่มีความเชื่อมั่น 100 เปอร์เซ็นต์กำหนดเป็นเงื่อนไขของข้อคำถามทั้งสองกรณี คือ กรณีที่มีเงื่อนไขซ้ำซ้อน และกรณีตัดเงื่อนไขซ้ำออกจะได้ผลลัพธ์ที่เหมือนกัน ดังนี้

SELECT *

FROM weather

WHERE outlook = ‘overcast’ and play = ‘yes’

overcast	hot	high	false	yes
overcast	cool	normal	true	yes
overcast	mild	high	true	yes
overcast	hot	normal	false	yes

SELECT *

FROM weather

WHERE outlook = ‘overcast’

overcast	hot	high	false	yes
overcast	cool	normal	true	yes
overcast	mild	high	true	yes
overcast	hot	normal	false	yes

ผลลัพธ์จากการสอบถามข้อมูลของทั้งสองข้อคำถามให้ผลลัพธ์ออกมาทั้งหมด 4 แถว และมีค่าเหมือนกันทุกค่า แต่ถ้านำกฎความสัมพันธ์ในข้อที่ 12 (temperature = mild => play = yes) ที่มีค่าความเชื่อมั่นเพียง 67 เปอร์เซ็นต์กำหนดเป็นเงื่อนไขของข้อคำถามในการสอบถามข้อมูลแล้วจะได้ผลลัพธ์ดังต่อไปนี้

SELECT *

FROM weather

WHERE temperature = ‘mild’ and play = ‘yes’

rainy	mild	high	false	yes
rainy	mild	normal	false	yes
sunny	mild	normal	true	yes
overcast	mild	high	true	yes

SELECT *

FROM weather

WHERE temperature = ‘mild’

rainy	mild	high	false	yes
sunny	mild	high	false	no
rainy	mild	normal	false	yes
sunny	mild	normal	true	yes
overcast	mild	high	true	yes
rainy	mild	high	true	no

จากผลการทดสอบ จะเห็นได้ว่า เมื่อลองทำการตัดคอลัมน์เงื่อนไขที่ไม่จำเป็นออกจากเงื่อนไขทั้งหมดแล้ว ปรากฏว่าผลลัพธ์ที่ได้มีค่าไม่ตรงกัน ดังนั้นถ้านำกฎความสัมพันธ์ที่มีค่าความเชื่อมั่นไม่ถึง 100 เปอร์เซ็นต์ มาใช้ในการปรับปรุงข้อคำถามจะทำให้ผลลัพธ์ที่ได้ออกมามีค่าผิดผลาดไปจากเดิมที่ผู้ใช้ต้องการทราบผลจริง ๆ

3.2.6 การตรวจสอบเงื่อนไขกับกฎข้อบังคับ

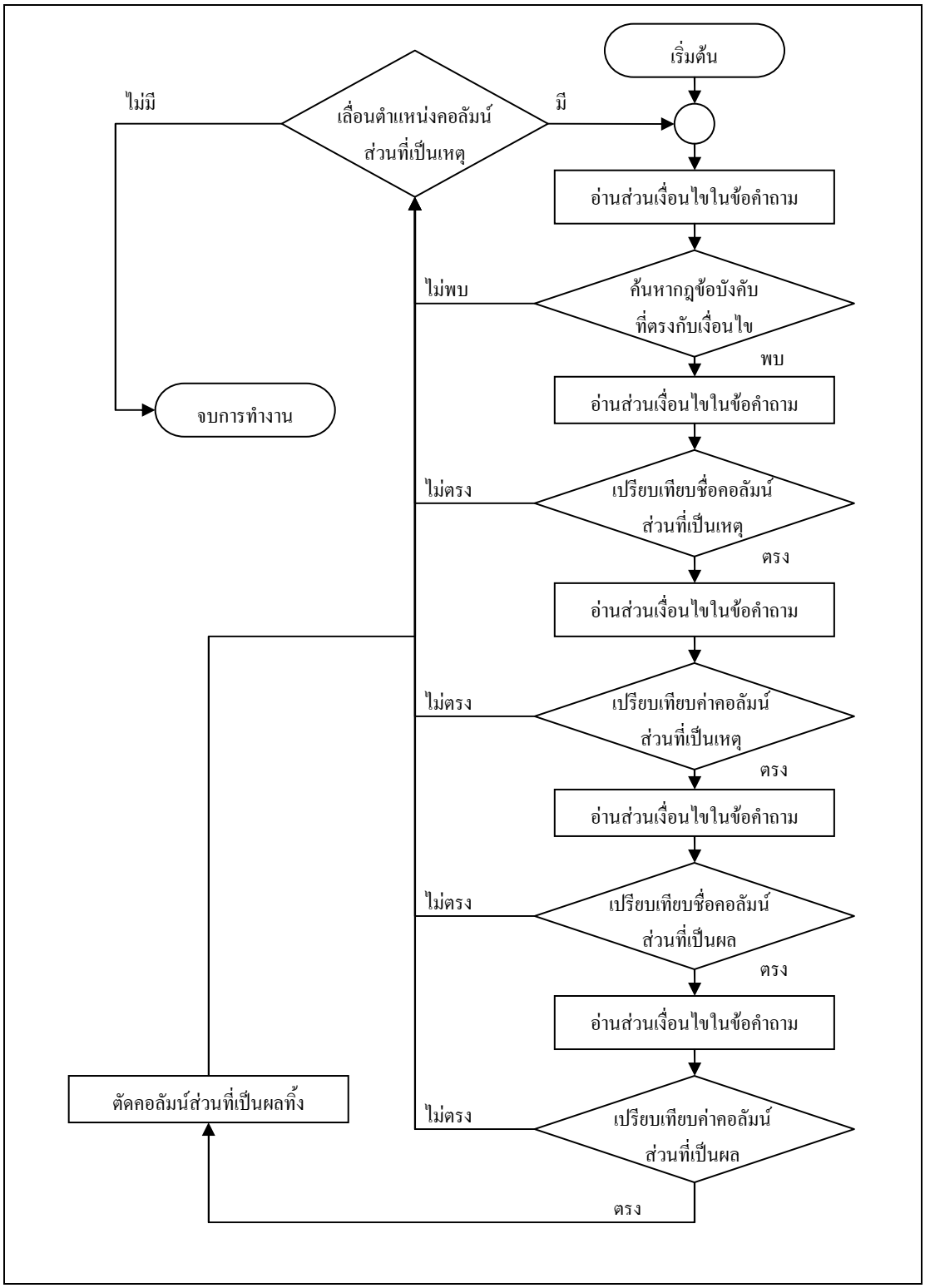
ในการตรวจสอบเงื่อนไขของข้อคำถาม จะทำการนำกฎข้อบังคับที่ได้จากการค้นหาความสัมพันธ์จากข้อมูลมาใช้ในการตัดสินใจ เพื่อทำการตัดเงื่อนไขที่ไม่จำเป็นออกจากข้อคำถาม ซึ่งกฎข้อบังคับที่จะนำมาใช้ในการตรวจสอบนี้ จะต้องเป็นกฎข้อบังคับที่ได้มาจากกฎความสัมพันธ์ที่สร้างมาจากข้อมูลเดียวกันกับข้อมูลที่ต้องการสอบถามข้อมูล ในการตรวจสอบจะมีขั้นตอนดังนี้

- อ่านเงื่อนไขของข้อคำถามในส่วนของ Where clause
- ทำการเปรียบเทียบชื่อคอลัมน์ในเงื่อนไข กับชื่อคอลัมน์ในส่วนเหตุของกฎข้อบังคับ (IF clause)
  - ถ้าชื่อของคอลัมน์ตรงกัน จะนำค่าแอททริบิวต์ในคอลัมน์นั้นมาเปรียบเทียบกับค่าแอททริบิวต์ในคอลัมน์ส่วนเหตุของกฎข้อบังคับ
  - ถ้าชื่อของคอลัมน์ไม่ตรงกัน จะนำชื่อของคอลัมน์ถัดไปมาทำการเปรียบเทียบแทนจนกระทั่งไม่สามารถตัดเงื่อนไขออกจากข้อคำถามได้แล้ว
  - ถ้าค่าแอททริบิวต์ตรงกัน
    - กรณีที่เป็นค่าแอททริบิวต์ในส่วนที่เป็นเหตุของกฎข้อบังคับ จะนำชื่อของคอลัมน์ถัดไปมาเปรียบเทียบกับชื่อคอลัมน์ในส่วนผลของกฎข้อบังคับ (THEN clause)
    - กรณีที่เป็นค่าแอททริบิวต์ในส่วนผลของกฎข้อบังคับ จะทำการตัดคอลัมน์ที่เป็นผลของเงื่อนไขนั้นทิ้ง เนื่องจากเกิดความซ้ำซ้อน เมื่อเทียบกับกฎข้อบังคับที่มี
  - ถ้าค่าของแอททริบิวต์ไม่ตรงกัน จะนำชื่อคอลัมน์ถัดไปมาเปรียบเทียบแทนจนกระทั่งไม่สามารถตัดเงื่อนไขออกจากข้อคำถามได้แล้ว

การปรับปรุงข้อคำถามโดยการตัดเงื่อนไขที่ไม่จำเป็นออกนี้ โปรแกรม SQOARI จะต้องจัดการรูปแบบเงื่อนไขของข้อคำถามให้อยู่ในรูปแบบที่เหมาะสมก่อนจะทำการนำค่าที่อยู่ภายในเงื่อนไขของข้อคำถามนั้น ๆ มาเปรียบเทียบกับกฎข้อบังคับที่มี ในการนำข้อคำถามปรับปรุงประสิทธิภาพนั้น โปรแกรมจะตัดมาเฉพาะในส่วนของเงื่อนไขของข้อคำถามเท่านั้น และนำเงื่อนไขดังกล่าวแยกออกมาเฉพาะคอลัมน์ในส่วนที่มีการเชื่อมกันด้วยเครื่องหมายและ (AND) เพื่อนำมาเปรียบเทียบกับกฎข้อบังคับที่มี ดังเช่น ถ้ามีข้อคำถามที่รับเข้ามาอยู่ในลักษณะดังนี้

```
SELECT *
FROM weather
WHERE (outlook = ‘overcast’ AND humidity = ‘normal’)
OR (outlook = ‘sunny’ AND windy = ‘true’)
```

เงื่อนไขที่สามารถนำมาปรับปรุงจากข้อคำถามข้างต้นนี้มีด้วยกัน 2 เงื่อนไข คือ outlook = ‘overcast’ AND humidity = ‘normal’ และ outlook = ‘sunny’ AND windy = ‘true’ โดยขั้นตอนการทำงานในส่วนของการปรับปรุงเงื่อนไขของข้อคำถามแสดงได้ดังรูปที่ 3.8



รูปที่ 3.8 ขั้นตอนการแปลงข้อคำถามด้วยกฎข้อบังคับ

การปรับปรุงประสิทธิภาพข้อคำถามแบบเชิงความหมายด้วยกฎความสัมพันธ์ ที่ได้จากข้อมูลสามารถแบ่งออกเป็นกรณีได้ทั้งหมด 2 กรณี คือ กรณีเงื่อนไขของข้อคำถามที่ความซ้ำซ้อนโดยไม่ขัดแย้งกับกฎข้อบังคับ และกรณีเงื่อนไขของข้อคำถามขัดแย้งกับกฎข้อบังคับ

กรณีที่ 1 เงื่อนไขของข้อคำถามมีความซ้ำซ้อน โดยไม่ขัดแย้งกับกฎข้อบังคับ

ในกรณีนี้ เงื่อนไขของข้อคำถามที่รับเข้ามาเกิดความซ้ำซ้อนกันเองภายในเงื่อนไข ซึ่งความซ้ำซ้อนที่เกิดขึ้นนี้สามารถตัดออกได้ โดยไม่มีผลกระทบกับข้อคำถามซึ่งผลลัพธ์ที่ได้ยังคงเหมือนกับข้อคำถามเดิม แต่ใช้เวลาในการประมวลผลที่น้อยลง

ตัวอย่างที่ 1 ให้ข้อคำถามที่รับเข้ามามีลักษณะดังนี้

```
SELECT *
FROM weather
WHERE outlook = ‘overcast’ AND play = ‘yes’
```

จากกฎข้อบังคับที่มี คือ

```
outlook = ‘overcast’ => play = ‘yes’
```

ความหมายของกฎข้อบังคับ คือ ถ้าค่าในคอลัมน์ที่ชื่อ outlook มีค่าเท่ากับ overcast แล้ว ค่าในคอลัมน์ที่ชื่อ play จะมีค่าเป็น yes เสมอ จากกฎข้อบังคับข้อนี้ทำให้เห็นว่า อย่างไรก็ตามผลที่ตามมาเมื่อ outlook = ‘overcast’ ค่าในคอลัมน์ play จะมีค่าเท่ากับ yes ทุกแถวข้อมูล ดังนั้น ถ้าพิจารณาจากเงื่อนไขของข้อคำถาม การระบุเงื่อนไข and play = ‘yes’ ไม่มีความจำเป็นที่ต้องค้นหาข้อมูลในเงื่อนไขนี้ ดังนั้นข้อคำถามนี้สามารถตัดเงื่อนไข and play = ‘yes’ นี้ออกได้ โดยไม่มีผลกระทบกับความหมายของข้อคำถาม ข้อคำถามสามารถเขียนใหม่ได้ดังนี้

```
SELECT *
FROM weather
WHERE outlook = ‘overcast’
```

ตัวอย่างที่ 2 ให้ข้อคำถามที่รับเข้ามามีลักษณะดังนี้

```
SELECT *
FROM weather
WHERE ( outlook = ‘overcast’ AND play = ‘yes’ )
OR (temperature = ‘cool’ AND humidity = ‘normal’)
```

จากกฎข้อบังคับที่มี คือ

```
outlook = ‘overcast’ => play = ‘yes’
temperature = ‘cool’ => humidity = ‘normal’
```

จากข้อคำถามเดิมสามารถเขียนให้อยู่ในรูปแบบใหม่ได้ ดังนี้

```
SELECT *
FROM weather
WHERE outlook = 'overcast' OR temperature = 'cool'
```

กรณีที่ 2 เงื่อนไขของข้อคำถามเกิดความขัดแย้งกับกฎข้อบังคับ

ในกรณีนี้ เงื่อนไขของข้อคำถามมีความขัดแย้งกับข้อมูลที่มีอยู่ภายในระบบฐานข้อมูล ทำให้ไม่พบข้อมูลตามที่เงื่อนไขระบุ ซึ่งถ้าข้อคำถามนี้ไม่ถูกกลั่นกรองก่อนแล้ว จะถูกส่งไปยังระบบฐานข้อมูลเพื่อนำไปประมวลผลข้อคำถาม ซึ่งทำให้เสียเวลาโดยไม่จำเป็น และเป็นการใช้ทรัพยากรในการประมวลผลอย่างเปล่าประโยชน์ โดยเฉพาะในกรณีที่ในระบบฐานข้อมูลที่ต้องการค้นหานั้นมีข้อมูลอยู่เป็นจำนวนมาก ดังนั้นถ้าระบบพบข้อคำถามลักษณะดังกล่าวจะทำการแจ้งให้ผู้ใช้ทราบทันทีว่าไม่พบข้อมูลที่ต้องการค้นหา โดยไม่จำเป็นต้องส่งข้อคำถามดังกล่าวไปยังระบบฐานข้อมูลเพื่อทำการประมวลผล

<u>ตัวอย่าง</u> ให้ข้อคำถามที่รับเข้ามามีลักษณะดังนี้

```
SELECT *
FROM weather
WHERE outlook = 'overcast' AND play = 'no'
```

จากกฎข้อบังคับที่มี คือ

```
outlook = 'overcast' => play = 'yes'
```

จากความหมายของกฎข้อบังคับที่มีระบุว่าถ้า outlook มีค่าเท่ากับ overcast แล้ว play จะมีค่าเท่ากับ yes เสมอ แต่จากเงื่อนไขของข้อคำถามระบุว่า ต้องการค้นหาข้อมูลที่มีค่า outlook เท่ากับ overcast และ play เท่ากับ no ซึ่งเงื่อนไขที่ระบุมานี้มีความขัดแย้งกันในคอลัมน์ play ดังนั้นจะไม่พบข้อมูลใด ๆ จากระบบฐานข้อมูล ระบบจะทำการแจ้งให้ผู้ใช้ทราบทันที โดยไม่ต้องมีการส่งข้อคำถามนี้ไปประมวลผลยังระบบฐานข้อมูล

3.3 การทดสอบกฎข้อบังคับ และข้อคำถามที่ได้จากการปรับปรุงแล้ว

การทดสอบงานวิจัยชิ้นนี้ จะประกอบไปด้วยการทดสอบทั้งกฎข้อบังคับที่ค้นพบจากการค้นหาความสัมพันธ์จากข้อมูล และการทดสอบผลลัพธ์ที่ได้จากการปรับปรุงประสิทธิภาพข้อคำถาม รวมทั้งเวลาที่ใช้ในการประมวลผลในแต่ละข้อคำถาม โดยจะนำเวลาที่ใช้ในการประมวลผลทั้งข้อคำถามเดิม และข้อคำถามที่มีการปรับปรุงแล้วมาเปรียบเทียบกันถึงความแตกต่างของเวลาที่ใช้ในการประมวลผล ซึ่งจะทำการเปรียบเทียบทั้งเวลาที่ได้จากโปรแกรม SQOARI และเวลาที่ได้

จากการนำข้อคำถามมาประมวลผลยังระบบฐานข้อมูลโดยตรง โดยใช้โปรแกรมสำเร็จรูปที่มีอยู่ในระบบฐานข้อมูลอยู่แล้ว

ในการทดสอบความถูกต้องของกฎข้อบังคับที่ได้จากการค้นหากฎความสัมพันธ์ จะนำกฎข้อบังคับที่ได้มาตรวจสอบความถูกต้องกับกฎความสัมพันธ์ที่หาได้จากโปรแกรมสำเร็จรูปภายนอก โดยโปรแกรมที่ใช้ในการค้นหากฎความสัมพันธ์เพื่อใช้ตรวจสอบความถูกต้องของกฎข้อบังคับ คือ โปรแกรม WEKA ซึ่งเป็นโปรแกรมสำเร็จรูปที่เปิดเผยซอร์สโค้ด (Open-source environment) ซึ่งเป็นโปรแกรมสำหรับการทำเหมืองข้อมูลทำการพัฒนาโดย Frank, Hall, Holmes, Martin, Mayo, Pfahringer, Smith and Witten (2005) (http://www.cs.waikato.ac.nz/ml/weka/) ในการนำข้อมูลมาค้นหากฎความสัมพันธ์จากโปรแกรม WEKA นี้ จะต้องมีการแปลงรูปแบบข้อมูลให้อยู่ในรูปแบบที่โปรแกรมกำหนดไว้ ซึ่งอยู่ในรูปแบบของ Flat file เป็นไฟล์ที่มีลักษณะเป็นข้อความ (Text file) เรียกว่า ARFF (Attribute-Relation File Format) ซึ่งประกอบด้วยข้อมูลสองส่วนคือส่วนอธิบายข้อมูลและส่วนที่เป็นข้อมูล ดังตัวอย่างในรูปที่ 3.9 (สำหรับการวิจัยนี้จะใช้ WEKA 3.5.4 ในการทดลอง)

```
@relation weather.symbolic

@attribute outlook {sunny, overcast, rainy}
@attribute temperature {hot, mild, cool}
@attribute humidity {high, normal}
@attribute windy {TRUE, FALSE}
@attribute play {yes, no}

@data
sunny,hot,high,FALSE,no
sunny,hot,high,TRUE,no
overcast,hot,high,FALSE,yes
rainy,mild,high,FALSE,yes
```

รูปที่ 3.9 แสดงรูปแบบไฟล์ข้อมูล ARFF (Attribute-Relation File Format)

ในการสร้างไฟล์ข้อมูลสำหรับใช้งานร่วมกับโปรแกรม WEKA นี้ จะต้องประกอบไปด้วยส่วนประกอบทั้งหมด 3 ส่วน ด้วยกันดังนี้

ส่วนที่ 1 ส่วนของหัวการประมวลผล

@relation weather.symbolic

ส่วนที่ 2 เป็นการประกาศ Attribute ของข้อมูล และชนิดข้อมูลในแต่ละ Attribute นั้น ๆ ซึ่งข้อมูลใน Attribute นี้สามารถแบ่งออกเป็น 2 ชนิดด้วยกัน คือ ข้อมูลชนิดตัวเลข (Numeric) และข้อมูลแบบแจกแจง (Categorical) ซึ่งข้อมูลแบบแจกแจงจะต้องมีการบอกค่าที่เป็นไปได้ทั้งหมดใน Attribute นั้น ๆ

@attribute outlook {sunny, overcast, rainy}

@attribute temperature {hot, mild, cool}

@attribute humidity {high, normal}

@attribute windy {TRUE, FALSE}

@attribute play {yes, no}

ส่วนที่ 3 เป็นส่วนของข้อมูล ซึ่งแต่ละแอททริบิวต์ในข้อมูลจะคั่นด้วยเครื่องหมายจุลภาค ‘,’ และข้อมูลหนึ่งแถวจะหมายถึง หนึ่งทูพเพิล หรือ หนึ่งเรคคอร์ด

@data

sunny,hot,high,FALSE,no

sunny,hot,high,TRUE,no

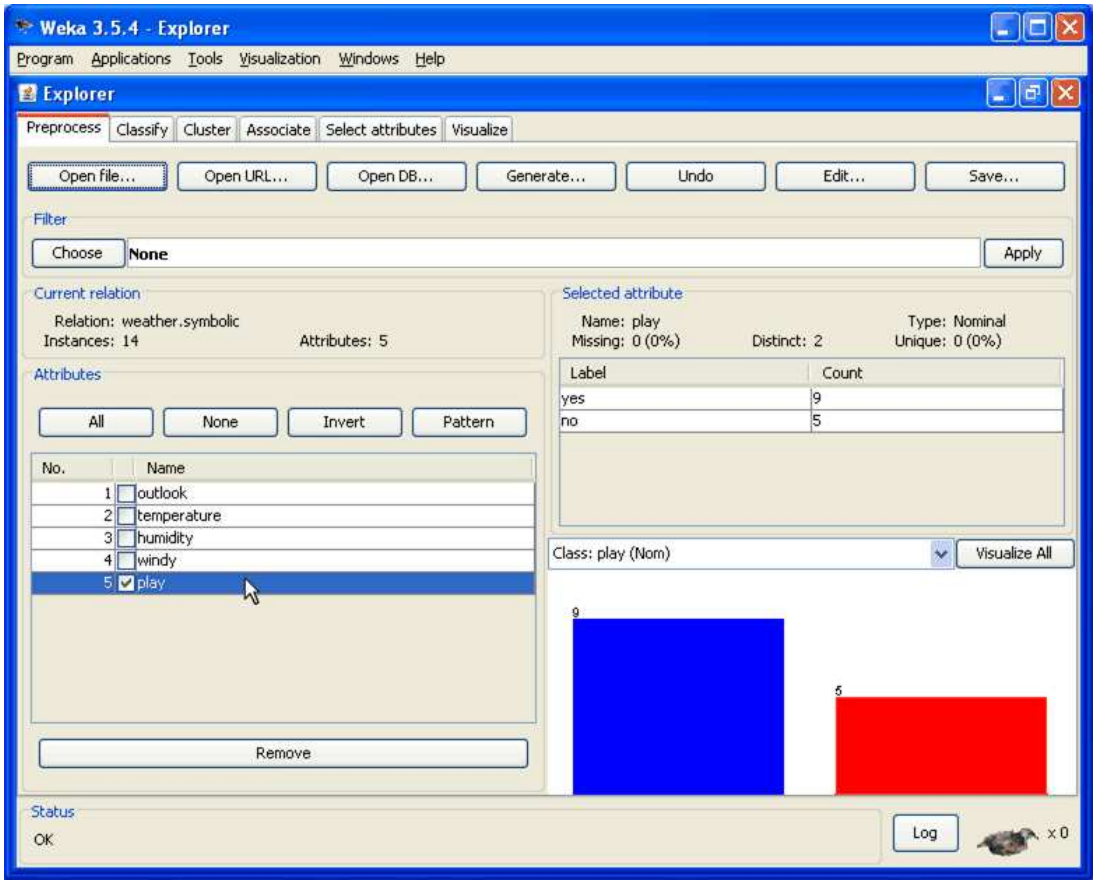
overcast,hot,high,FALSE,yes

rainy,mild,high,FALSE,yes

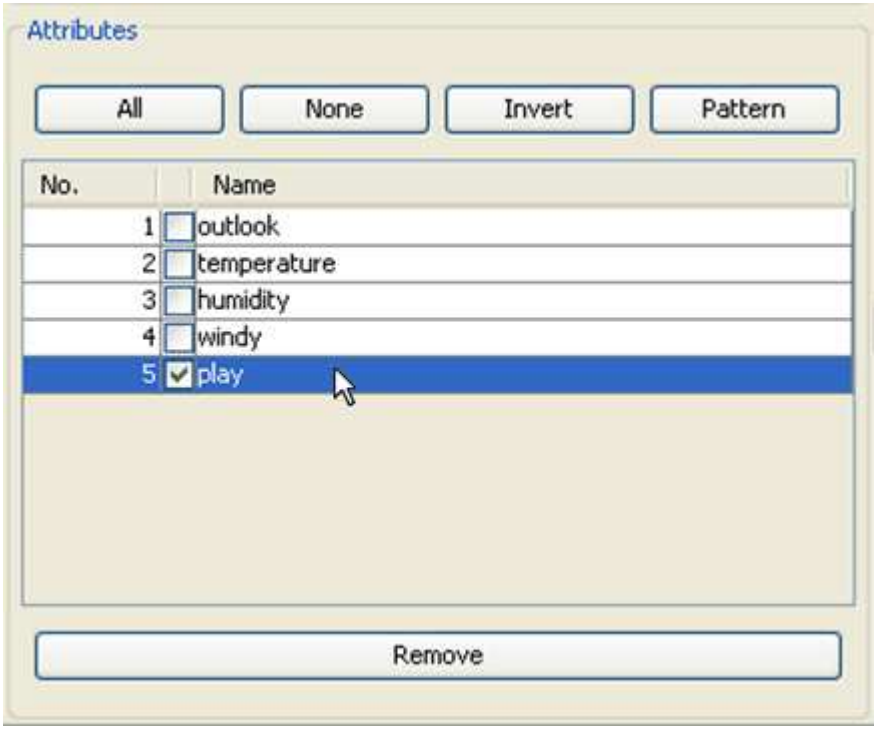
ขั้นตอนในการค้นหากฎความสัมพันธ์จากข้อมูลด้วยโปรแกรม WEKA ประกอบด้วยขั้นตอนทั้งหมด 2 ขั้นตอน ดังนี้

1. การเตรียมข้อมูลก่อนทำการค้นหากฎความสัมพันธ์

ทำการเปิดไฟล์ข้อมูลที่อยู่ในรูปแบบของ ARFF เพื่อนำข้อมูลที่ต้องการมาทำการค้นหากฎความสัมพันธ์ (รูปที่ 3.10) ในส่วนนี้ เราสามารถทำการตัดคอลัมน์ที่ไม่ต้องการออกจากการค้นหากฎความสัมพันธ์ โดยทำการเช็คเครื่องหมายถูกยังบริเวณ Check box ของคอลัมน์นั้น ๆ จากนั้นทำการเลือกที่ปุ่ม Remove (รูปที่ 3.11) เพื่อทำการตัดคอลัมน์นั้นออกจากการค้นหากฎความสัมพันธ์



รูปที่ 3.10 แสดงหน้าต่างการเตรียมข้อมูลสำหรับการค้นหากฎความสัมพันธ์



รูปที่ 3.11 แสดงการเลือกคอลัมน์ในกรณีไม่ต้องการคอลัมน์นั้นในการค้นหากฎความสัมพันธ์

2. การค้นหากฎความสัมพันธ์จากข้อมูลที่ต้องการ

ในการค้นหากฎความสัมพันธ์ด้วยโปรแกรม WEKA นี้ จะต้องทำการเลือกอัลกอริทึมที่ใช้ในการค้นหากฎความสัมพันธ์ โดยการเลือกที่ปุ่ม Choose (รูปที่ 3.12) ซึ่งในการค้นหากฎความสัมพันธ์ในงานวิจัยนี้จะใช้อัลกอริทึมเอไพรออรีในการค้นหาความสัมพันธ์ของข้อมูล และเมื่อเลือกอัลกอริทึมที่จะใช้ในการค้นหากฎความสัมพันธ์เรียบร้อยแล้ว จะต้องทำการกำหนดค่าต่าง ๆ ที่เกี่ยวข้องกับอัลกอริทึมเอไพรออรี ซึ่งค่าที่จะต้องกำหนดในโปรแกรม WEKA นี้มีด้วยกันทั้งหมด 3 ค่าคือ

ค่าสนับสนุนขั้นต่ำ (Minimum support)

ในโปรแกรม WEKA นี้จะเรียกค่าสนับสนุนขั้นต่ำนี้ว่า lowerBoundMinSupport และ upperBoundMinSupport ซึ่งในโปรแกรม WEKA นี้ผู้ใช้งานจะต้องทำการกำหนดค่าสนับสนุนทั้งสองค่าให้กับโปรแกรม โดยค่า lowerBoundMinSupport คือค่าสนับสนุนที่ต่ำที่สุดที่ต้องการค้นหากฎความสัมพันธ์ และค่า upperBoundMinSupport คือค่าสนับสนุนที่มากที่สุดที่ต้องการค้นหากฎความสัมพันธ์ ดังนั้นในการโปรแกรม WEKA ค่าสนันสนุนจะอยู่ในรูปแบบของช่วงตัวเลข ซึ่งเราจะต้องทำการกำหนดค่าสนับสนุนทั้งสองค่านี้ให้กับโปรแกรม โดยปกติแล้วโปรแกรมจะทำการกำหนดค่า upperBoundMinSupport มาให้เท่ากับ 1 อยู่แล้ว ดังนั้นในการค้นหากฎความสัมพันธ์นี้จึง

มีการกำหนดค่า lowerBoundMinSupport เพียงค่าเดียวเท่านั้น ซึ่งค่าที่กำหนดจะอยู่ในช่วง [0, 1] เท่านั้น

ค่าความเชื่อมั่นขั้นต่ำ (Minimum confidence)

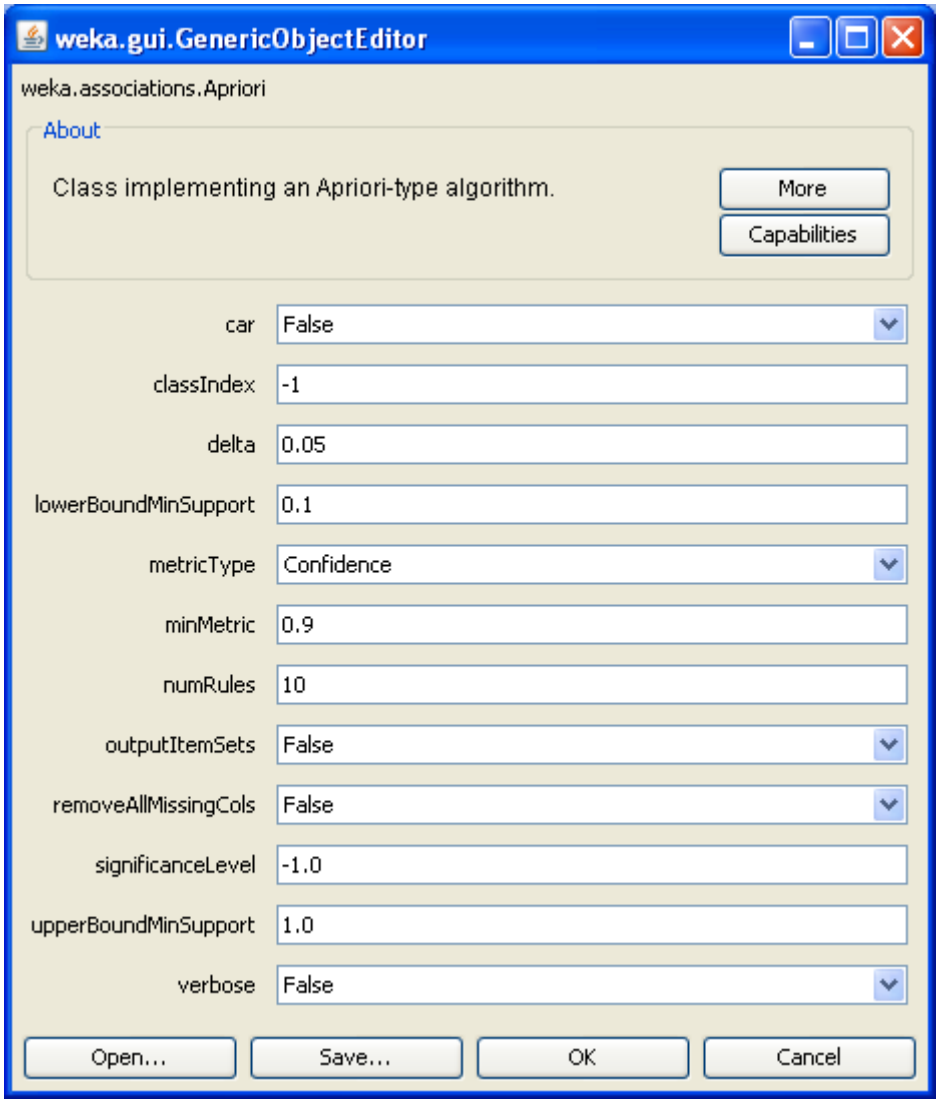
ในโปรแกรม WEKA นี้จะเรียกว่า minMetric ซึ่งในการกำหนดค่าความเชื่อมั่นนี้ลงในโปรแกรม จะต้องทำการเลือกชนิดของค่าให้เป็นค่าความเชื่อมั่นก่อน โดยทำการเลือกที่ metricType และกำหนดค่าให้เป็น Confidence โดยปกติแล้วโปรแกรมจะกำหนดค่า metricType มาให้เท่ากับ Confidence อยู่แล้ว และจากนั้นจึงจะทำการกำหนดค่าให้กับ minMetric ได้ ซึ่งค่าที่สามารถกำหนดได้นั้นจะอยู่ในช่วง [0, 1] เท่านั้น สำหรับในการทดสอบผลการทำงานนี้จะกำหนดค่าความเชื่อมั่นให้มีค่าเท่ากับ 1 เท่านั้น

จำนวนของกฎความสัมพันธ์ (Number of rules)

ใช้ในการกำหนดจำนวนการแสดงผลของกฎความสัมพันธ์ที่ค้นพบได้จากข้อมูล ซึ่งโปรแกรมจะกำหนดการแสดงผลของกฎความสัมพันธ์ให้เท่ากับ 10 กฎความสัมพันธ์ ดังนั้นถ้าต้องการเพิ่มจำนวนกฎความสัมพันธ์สามารถทำได้โดยเปลี่ยนตัวเลขที่ numRules

เมื่อทำการกำหนดค่าต่าง ๆ เรียบร้อยแล้วจึงคลิกที่ปุ่ม Start เพื่อทำการประมวลผลข้อมูลเพื่อค้นหากฎความสัมพันธ์ของข้อมูลนั้นออกมา

รูปที่ 3.12 แสดงหน้าต่างการค้นหากฎความสัมพันธ์



รูปที่ 3.13 แสดงหน้าต่างในการกำหนดค่าในการใช้ค้นหากฎความสัมพันธ์

ในการทดสอบข้อคำถามเดิมและข้อคำถามที่มีการปรับปรุงแล้ว จะทำการทดสอบกับระบบฐานข้อมูลด้วยกัน 2 ระบบ คือ Oracle 10g Express Edition, MS SQL Server 2000 ซึ่งในการทดสอบข้อคำถามนี้จะทำการทดสอบเวลาที่ใช้ในการประมวลผลข้อคำถาม และผลลัพธ์ที่ได้จากการประมวลผลข้อคำถาม โดยจะนำเอาผลของทั้งข้อคำถามเดิม และข้อคำถามที่มีการปรับปรุงแล้วมาเปรียบเทียบกัน และผลที่จะนำมาเปรียบเทียบกันนี้มีการแบ่งออกเป็น 2 ลักษณะด้วยกัน นั่นคือ ผลที่ได้จากการประมวลผลผ่านทางโปรแกรม SQOARI และผลที่ได้จากการประมวลผลผ่านจากระบบฐานข้อมูลโดยตรง

3.4 แหล่งที่มาของข้อมูลในการใช้ทดสอบ

ในการทดสอบงานวิจัยชิ้นนี้ ใชข้อมูลในการทดสอบผลทั้งหมดมาจากแหล่งข้อมูลของมหาวิทยาลัยแห่งรัฐแคลิฟอร์เนีย เมืองเออร์ไวน์ (University of California at Irvine) (Http: //www. Ics.uci.edu/~mlearn/MLRepository.html) จำนวนทั้งสิ้น 7 ชุด ซึ่งรายละเอียดของข้อมูลในแต่ละชุดมีดังนี้

ตารางที่ 3.5 รายละเอียดข้อมูลที่ใช้ในการทดสอบ

ชื่อข้อมูล	จำนวนแถวข้อมูล	จำนวนคอลัมน์
Automobile	205	26
Car Evaluation	1,728	6
Credit Approve	690	15
Mushroom	8,124	22
Nursery	12,960	8
Solar Flare	1,389	13
Weather	14	5

บทที่ 4
การทดสอบและอภิปรายผล

ในการทดสอบผลจะแบ่งการทดสอบออกเป็นสองส่วนหลักด้วยกัน คือส่วนของการตรวจสอบความถูกต้องของกฎความสัมพันธ์ที่ได้ และส่วนของการทดสอบประสิทธิภาพของข้อคำถามที่มีการปรับปรุงประสิทธิภาพแล้ว โดยในส่วนแรกจะเป็นการค้นหากฎความสัมพันธ์ที่ได้จากข้อมูล เพื่อนำกฎความสัมพันธ์ที่ได้มาสร้างเป็นกฎข้อบังคับ เพื่อนำไปใช้ในการปรับปรุงประสิทธิภาพของข้อคำถามในขั้นตอนถัดไป ซึ่งในการค้นหากฎความสัมพันธ์นี้ จะใช้โปรแกรม SQOARI ในการค้นหากฎความสัมพันธ์ และใช้ในการทดสอบผลการประมวลผลของข้อคำถามเดิมเปรียบเทียบกับข้อคำถามที่มีการปรับปรุงแล้ว โปรแกรม SQOARI นี้จะนำเอาอัลกอริทึมเอไพรออรีมาเขียนใหม่ให้อยู่ในรูปแบบของภาษาจาวา (Java) โดยมีส่วนของการติดต่อกับผู้ใช้งาน (Graphic user interface : GUI) ให้สามารถใช้ในการทดสอบได้สะดวกมากยิ่งขึ้น กฎความสัมพันธ์ที่ได้จากการค้นหาด้วยโปรแกรม SQOARI นี้จะนำผลมาเปรียบเทียบกับกับผลที่ได้จากการค้นหาด้วยโปรแกรม WEKA เพื่อตรวจสอบความถูกต้องของกฎความสัมพันธ์ที่จะนำไปสร้างเป็นกฎข้อบังคับต่อไป ในส่วนที่สองคือการเปรียบเทียบประสิทธิภาพของข้อคำถามเดิม และข้อคำถามที่มีการปรับปรุงประสิทธิภาพแล้ว โดยจะใช้เวลาในการประมวลผลของข้อคำถามในการเปรียบเทียบผลการทดสอบ รวมถึงผลลัพธ์ที่ได้จากการประมวลผลของข้อคำถามทั้งสองข้อคำถามว่ามีค่าที่ได้ตรงกันหรือไม่ ในการทดสอบการประมวลผลข้อคำถามนี้จะทดสอบกับระบบจัดการฐานข้อมูลด้วยกันสองระบบด้วยกัน คือ Oracle 10g Express Edition และ Microsoft SQL Server 2000 โดยข้อมูลที่ใช้ทดสอบมีด้วยกันทั้งหมด 7 ชุด ทำการทดสอบการประมวลผลข้อคำถามผ่านทางโปรแกรม SQOARI และผ่านทางระบบจัดการฐานข้อมูลโดยตรง โดยทำการทดสอบบนเครื่องคอมพิวเตอร์ Pentium IV ความเร็ว 3.2 GHz หน่วยความจำหลัก 512 MB ฮาร์ดดิสก์ความจุ 80 GB โดยการทดสอบการค้นหากฎความสัมพันธ์ด้วยโปรแกรม SQOARI กับข้อมูลทั้ง 7 ชุด ปรากฏรายละเอียดในหัวข้อ 4.1 ผลการทดสอบข้อคำถามที่มีการปรับปรุงประสิทธิภาพด้วยกฎข้อบังคับที่ได้ กับข้อคำถามเดิมบนระบบจัดการฐานข้อมูล Oracle 10g Express Edition ปรากฏรายละเอียดในหัวข้อ 4.2 ผลการทดสอบข้อคำถามที่มีการปรับปรุงประสิทธิภาพด้วยกฎข้อบังคับที่มีกับข้อคำถามเดิมบนระบบจัดการฐานข้อมูล Microsoft SQL Server 2000 ปรากฏรายละเอียดในหัวข้อ 4.3 และหัวข้อ 4.4 เป็นการอภิปรายสรุป

4.1 ผลการทดสอบการค้นหากฎความสัมพันธ์

ตารางที่ 4.1, 4.3, 4.5, 4.7 และ 4.9 แสดงผลการค้นหากฎความสัมพันธ์โดยการเรียกใช้โปรแกรม SQOARI ที่ทำการพัฒนาขึ้น โดยกำหนดค่าสนับสนุนขั้นต่ำที่ 10%, 20%, 30%, 40% และ 50% ตามลำดับ และตารางที่ 4.2, 4.4, 4.6, 4.8 และ 4.10 แสดงผลการค้นหากฎความสัมพันธ์โดยการเรียกใช้โปรแกรม WEKA โดยกำหนดค่าสนับสนุนขั้นต่ำที่ 10%, 20%, 30%, 40% และ 50% ตามลำดับ และในการทดสอบการค้นหากฎความสัมพันธ์ มีการกำหนดค่าความเชื่อมั่นขั้นต่ำที่ 100 เปอร์เซ็นต์

ตารางที่ 4.1 แสดงจำนวนของ 1-large itemset, 2-large itemset และจำนวนกฎความสัมพันธ์ที่ค้นพบจากโปรแกรม SQOARI ด้วยค่าสนับสนุนเท่ากับ 10 เปอร์เซ็นต์

ชื่อชุดข้อมูล	1-large itemset	2-large itemset	จำนวนกฎความสัมพันธ์
Automobile	22	122	21
Car Evaluation	23	52	2
Credit Approve	19	121	4
Mushroom	56	763	155
Nursery	30	137	2
Solar flare	23	138	19
Weather	12	47	2

ตารางที่ 4.2 แสดงจำนวนของ 1-large itemset, 2-large itemset และจำนวนกฎความสัมพันธ์ที่ค้นพบจากโปรแกรม WEKA ด้วยค่าสนับสนุนเท่ากับ 10 เปอร์เซ็นต์

ชื่อชุดข้อมูล	1-large itemset	2-large itemset	จำนวนกฎความสัมพันธ์
Automobile	22	122	21
Car Evaluation	23	52	2
Credit Approve	19	121	4
Mushroom	56	763	155
Nursery	30	137	2
Solar flare	23	138	19
Weather	12	47	2

ตารางที่ 4.3 แสดงจำนวนของ 1-large itemset, 2-large itemset และจำนวนกฎความสัมพันธ์ที่ค้นพบจากโปรแกรม SQOARI ด้วยค่าสนับสนุนเท่ากับ 20 เปอร์เซ็นต์

ชื่อชุดข้อมูล	1-large itemset	2-large itemset	จำนวนกฎความสัมพันธ์
Automobile	15	71	12
Car Evaluation	23	8	2
Credit Approve	17	73	4
Mushroom	Error	Error	Error
Nursery	30	1	2
Solar flare	16	63	10
Weather	12	26	2

ตารางที่ 4.4 แสดงจำนวนของ 1-large itemset, 2-large itemset และจำนวนกฎความสัมพันธ์ที่ค้นพบจากโปรแกรม WEKA ด้วยค่าสนับสนุนเท่ากับ 20 เปอร์เซ็นต์

ชื่อชุดข้อมูล	1-large itemset	2-large itemset	จำนวนกฎความสัมพันธ์
Automobile	15	71	12
Car Evaluation	23	8	2
Credit Approve	17	73	4
Mushroom	Error	Error	Error
Nursery	30	1	2
Solar flare	16	63	10
Weather	12	26	2

ตารางที่ 4.5 แสดงจำนวนของ 1-large itemset, 2-large itemset และจำนวนกฎความสัมพันธ์ที่ค้นพบจากโปรแกรม SQOARI ด้วยค่าสนับสนุนเท่ากับ 30 เปอร์เซ็นต์

ชื่อชุดข้อมูล	1-large itemset	2-large itemset	จำนวนกฎความสัมพันธ์
Automobile	14	44	11
Car Evaluation	10	2	2
Credit Approve	14	51	2
Mushroom	27	162	42
Nursery	17	1	2
Solar flare	11	40	2
Weather	12	9	2

ตารางที่ 4.6 แสดงจำนวนของ 1-large itemset, 2-large itemset และจำนวนกฎความสัมพันธ์ที่ค้นพบจากโปรแกรม WEKA ด้วยค่าสนับสนุนเท่ากับ 30 เปอร์เซ็นต์

ชื่อชุดข้อมูล	1-large itemset	2-large itemset	จำนวนกฎความสัมพันธ์
Automobile	14	44	11
Car Evaluation	10	2	2
Credit Approve	14	51	2
Mushroom	27	162	42
Nursery	17	1	2
Solar flare	11	40	2
Weather	12	9	2

ตารางที่ 4.7 แสดงจำนวนของ 1-large itemset, 2-large itemset และจำนวนกฎความสัมพันธ์ที่ค้นพบจากโปรแกรม SQOARI ด้วยค่าสนับสนุนเท่ากับ 40 เปอร์เซ็นต์

ชื่อชุดข้อมูล	1-large itemset	2-large itemset	จำนวนกฎความสัมพันธ์
Automobile	10	25	6
Car Evaluation	0	0	0
Credit Approve	13	26	2
Mushroom	21	97	31
Nursery	2	0	0
Solar flare	9	25	1
Weather	6	2	0

ตารางที่ 4.8 แสดงจำนวนของ 1-large itemset, 2-large itemset และจำนวนกฎความสัมพันธ์ที่ค้นพบจากโปรแกรม WEKA ด้วยค่าสนับสนุนเท่ากับ 40 เปอร์เซ็นต์

ชื่อชุดข้อมูล	1-large itemset	2-large itemset	จำนวนกฎความสัมพันธ์
Automobile	10	25	6
Car Evaluation	0	0	0
Credit Approve	13	26	2
Mushroom	21	97	31
Nursery	0	0	0
Solar flare	9	25	1
Weather	6	2	0

ตารางที่ 4.9 แสดงจำนวนของ 1-large itemset, 2-large itemset และจำนวนกฎความสัมพันธ์ที่ค้นพบจากโปรแกรม SQOARI ด้วยค่าสนับสนุนเท่ากับ 50 เปอร์เซ็นต์

ชื่อชุดข้อมูล	1-large itemset	2-large itemset	จำนวนกฎความสัมพันธ์
Automobile	7	16	4
Car Evaluation	0	0	0
Credit Approve	9	5	2
Mushroom	13	41	19
Nursery	2	0	0
Solar flare	6	15	0
Weather	0	0	0

ตารางที่ 4.10 แสดงจำนวนของ 1-large itemset, 2-large itemset และจำนวนกฎความสัมพันธ์ที่ค้นพบจากโปรแกรม WEKA ด้วยค่าสนับสนุนเท่ากับ 50 เปอร์เซ็นต์

ชื่อชุดข้อมูล	1-large itemset	2-large itemset	จำนวนกฎความสัมพันธ์
Automobile	7	16	4
Car Evaluation	0	0	0
Credit Approve	9	5	2
Mushroom	13	41	19
Nursery	0	0	0
Solar flare	6	15	0
Weather	0	0	0

เมื่อโปรแกรม SQOARI ทำการค้นหากฎความสัมพันธ์จากข้อมูลเสร็จเรียบร้อยแล้ว จะได้กฎความสัมพันธ์ที่อยู่ในรูปของหนึ่งคอลัมน์ที่เป็นเหตุและหนึ่งคอลัมน์ที่เป็นผล มีค่าความเชื่อมั่นเท่ากับ 100 เปอร์เซ็นต์ จากนั้นนำกฎความสัมพันธ์ที่ได้จากการค้นหาจากโปรแกรม WEKA มาเปรียบเทียบผลกับกฎความสัมพันธ์ที่ได้จากโปรแกรม SQOARI แต่เนื่องจากกฎความสัมพันธ์ที่ได้จากโปรแกรม WEKA นั้น จะเป็นผลลัพธ์ทั้งหมดที่อัลกอริทึมเอไพรออรีค้นหาพบ จึงประกอบไปด้วยกฎที่มีจำนวนคอลัมน์ที่เป็นเหตุตั้งแต่หนึ่งคอลัมน์ไปจนถึงหลายคอลัมน์และคอลัมน์ที่เป็นผลตั้งแต่หนึ่งคอลัมน์ไปจนถึงหลายคอลัมน์เช่นกัน ดังนั้นก่อนนำกฎความสัมพันธ์ที่ค้นพบได้มาทำการเปรียบเทียบกันนั้น เราจึงต้องทำการตัดกฎความสัมพันธ์ที่ไม่จำเป็นออกจากผลลัพธ์ที่ทำการค้นหาด้วยโปรแกรม WEKA ก่อน และจำเป็นต้องทำการเรียงลำดับค่าสนับสนุนของกฎความสัมพันธ์ที่ค้นพบเสียก่อน (การเรียงลำดับจะเรียงจากกฎความสัมพันธ์ที่มีค่าสนับสนุนสูงสุดไปหาต่ำสุด) ดังแสดงในรูปที่ 4.1 และ 4.2

1 : num-of-cylinders = four (159) -> engine-location = front (159) conf:(1)

2 : engine-type = ohc (148) -> engine-location = front (148) conf:(1)

3 : drive-wheels = fwd (120) -> engine-location = front (120) conf:(1)

4 : num-of-doors = four (114) -> engine-location = front (114) conf:(1)

5 : body-style = sedan (96) -> engine-location = front (96) conf:(1)

6 : fuel-system = mpfi (94) -> fuel-type = gas 94) conf:(1)

7 : body-style = hatchback (70) -> engine-location = front (70) conf:(1)

8 : symboling = 0 (67) -> engine-location = front (67) conf:(1)

9 : fuel-system = 2bbl (66) -> fuel-type = gas (66) conf:(1)

10 : fuel-system = 2bbl (66) -> aspiration = std (66) conf:(1)

11 : fuel-system = 2bbl (66) -> engine-location = front (66) conf:(1)

รูปที่ 4.1 กฎความสัมพันธ์ที่ค้นพบได้จากโปรแกรม SQOARI
ด้วยค่าสนันสนุนเท่ากับ 50 เปอร์เซ็นต์

1 : num-of-cylinders = four (159) -> engine-location = front (159) conf:(1)

2 : engine-type = ohc (148) -> engine-location = front (148) conf:(1)

3 : drive-wheels = fwd (120) -> engine-location = front (120) conf:(1)

4 : num-of-doors = four (114) -> engine-location = front (114) conf:(1)

5 : body-style = sedan (96) -> engine-location = front (96) conf:(1)

6 : fuel-system = mpfi (94) -> fuel-type = gas 94) conf:(1)

7 : body-style = hatchback (70) -> engine-location = front (70) conf:(1)

8 : symboling = 0 (67) -> engine-location = front (67) conf:(1)

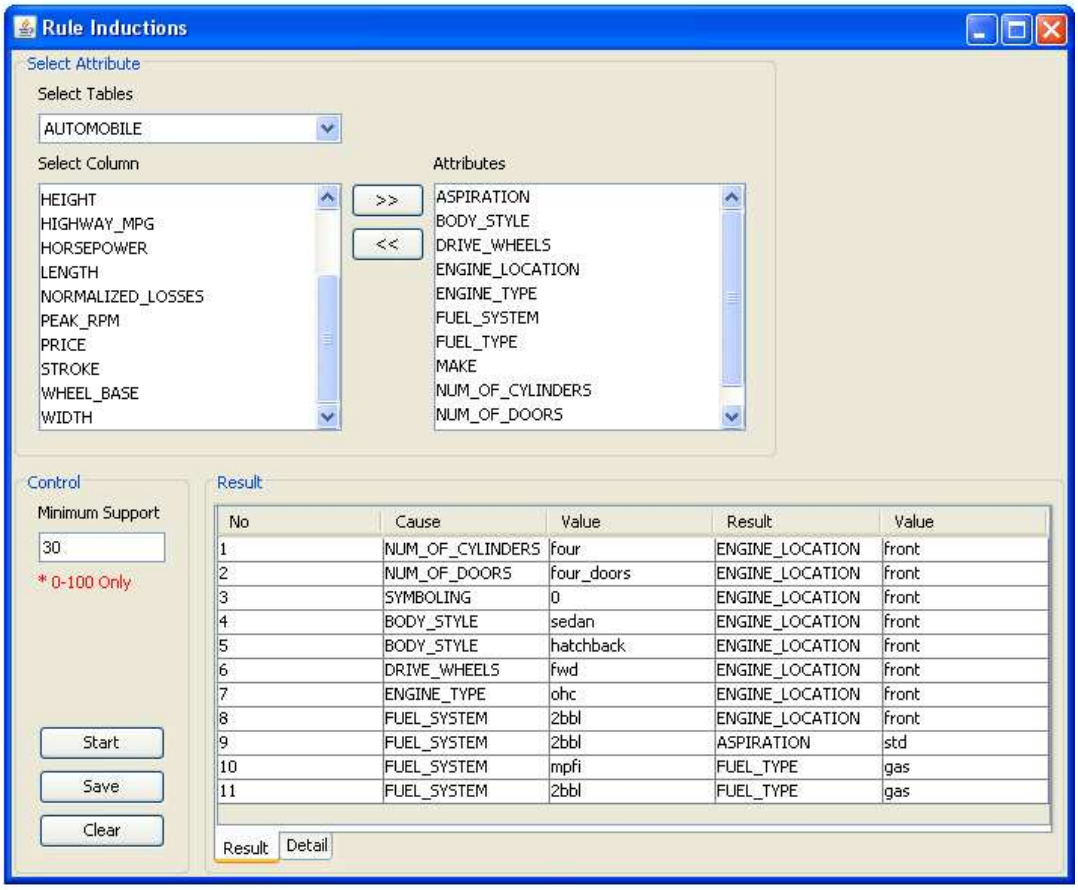
9 : fuel-system = 2bbl (66) -> fuel-type = gas (66) conf:(1)

10 : fuel-system = 2bbl (66) -> aspiration = std (66) conf:(1)

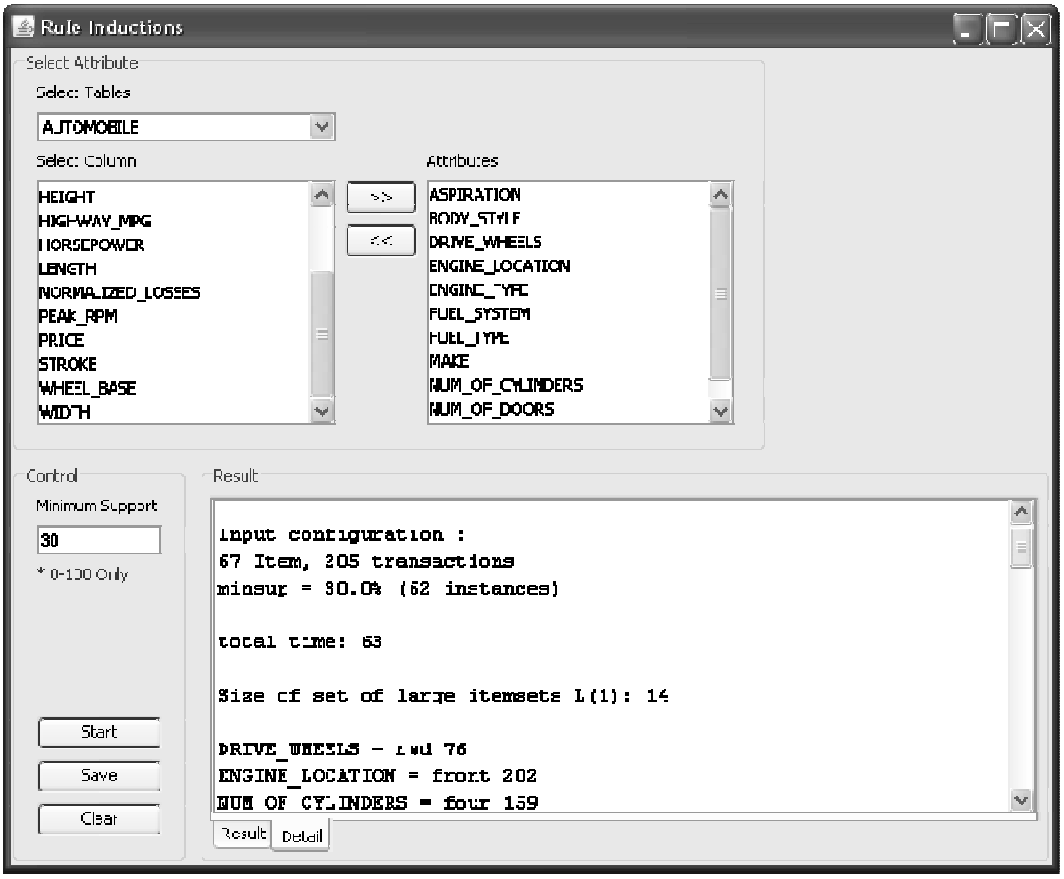
11 : fuel-system = 2bbl (66) -> engine-location = front (66) conf:(1)

รูปที่ 4.2 กฎความสัมพันธ์ที่ค้นพบได้จากโปรแกรม WEKA
ด้วยค่าสนันสนุนเท่ากับ 50 เปอร์เซ็นต์

เมื่อทำการจัดเรียงลำดับค่าสนับสนุนเรียบร้อยแล้ว จะเห็นว่ากฎความสัมพันธ์ที่ได้จากการค้นหาด้วยโปรแกรม SQOARI มีกฎความสัมพันธ์ด้วยกันทั้งหมด 11 กฎ และกฎความสัมพันธ์ที่ได้จากค้นหาด้วยโปรแกรม WEKA เมื่อทำการตัดกฎความสัมพันธ์ให้เหลือเพียงกฎที่อยู่ในรูปแบบที่ต้องการ หลังจากนั้นพบว่ามีกฎความสัมพันธ์ทั้งหมด 11 กฎเช่นกัน และกฎความสัมพันธ์ที่ได้มาจากทั้งสองโปรแกรมนี้ เมื่อนำมาเปรียบเทียบกันปรากฏว่ากฎความสัมพันธ์ที่ได้มีค่าตรงกันทั้งหมด แสดงว่ากฎความสัมพันธ์ที่ได้จากโปรแกรม SQOARI มีความถูกต้องสามารถนำมาใช้สร้างเป็นกฎข้อบังคับเพื่อใช้ในการปรับปรุงประสิทธิภาพข้อคำถามต่อไป ในการนำข้อมูลมาค้นหากฎความสัมพันธ์ด้วยโปรแกรม SQOARI เมื่อเสร็จสิ้นการค้นหาแล้ว โปรแกรม SQOARI จะแสดงกฎความสัมพันธ์ที่ค้นพบออกมาอยู่ในรูปแบบของตารางดังรูปที่ 4.3 และรายละเอียดเกี่ยวกับข้อมูลที่ใช้ในการค้นหา รวมทั้งจำนวนของไอเท็มเซตที่ค้นพบดังรูปที่ 4.4



รูปที่ 4.3 หน้าต่างแสดงกฎความสัมพันธ์ที่ค้นพบจากโปรแกรม SQOARI



รูปที่ 4.4 หน้าต่างแสดงรายละเอียดของข้อมูล และlarge itemset
ที่ค้นพบจากโปรแกรม SQOARI

4.2 ผลการทดสอบข้อคำถามบนระบบจัดการฐานข้อมูล Oracle 10g Express Edition

ในการทดสอบผลของการประมวลผลข้อคำถาม จะทำการทดสอบทั้งการประมวลผลข้อคำถามด้วยโปรแกรม SQOARI และประมวลผลข้อคำถามด้วยตัวจัดการระบบฐานข้อมูลเอง ในการปรับปรุงประสิทธิภาพข้อคำถามนี้ จะนำกฎข้อบังคับที่ได้จากการค้นหากฎความสัมพันธ์ในแต่ละข้อมูลมาใช้ในการปรับปรุงประสิทธิภาพข้อคำถาม ซึ่งในบางชุดข้อมูลจะมีกฎข้อบังคับที่ได้อยู่เป็นจำนวนมาก ดังนั้นจึงจะทำการสุ่มเลือกกฎข้อบังคับที่ได้มาเพียง 5 กฎเท่านั้นในการนำกฎข้อบังคับดังกล่าวมาใช้ในการปรับปรุงประสิทธิภาพข้อคำถามเพื่อใช้ในการทดสอบการประมวลผล โดยกฎข้อบังคับที่จะในไปใช้ในการทดสอบกับชุดข้อมูลทั้งหมด 7 ชุด มีรายละเอียดดังตารางที่ 4.11

ตารางที่ 4.11 รายละเอียดกฎข้อบังคับที่ใช้ในการทดสอบข้อคำถาม

ชื่อชุดข้อมูล	กฎข้อบังคับ
Automobile	FUEL_SYSTEM = 2bbl -> ASPIRATION = std
	NUM_OF_CYLINDERS = four -> ENGINE_LOCATION = front
	ENGINE_TYPE = ohc -> ENGINE_LOCATION = front
	FUEL_TYPE = diesel -> FUEL_SYSTEM = idi
	SYMBOLING = 3 -> FUEL_TYPE = gas
Car evaluation	PERSONS = 2 -> GOAL = unacc
	SAFETY = low -> GOAL = unacc
Credit approve	A4 = u -> A5 = g
	A4 = y -> A5 = p
Mushroom	GILL_SPACING = c -> VEIL_TYPE = p
	GILL_ATTACHMENT = f -> VEIL_TYPE = p
	POPULATION = v -> VEIL_TYPE = p
	SPORE_PRINT_COLOR = n -> RING_NUMBER = o
	CAP_SHAPE = k -> VEIL_TYPE = p
Nursery	GOAL = not_recom -> HEALTH = not_recom
Solar flare	SPOT_DISTRIBUTION = O -> REGION_BECOME_HISTORICALLY_COM = 2
	SPOT_DISTRIBUTION = I -> REGION_BECOME_HISTORICALLY_COM = 2
	SPOT_DISTRIBUTION = X -> GOAL = H
	REGION_BECOME_HISTORICALLY_COM = 1 -> GOAL = H
	LARGEST_SPOT_SIZE = X -> GOAL = B
Weather	OUTLOOK = overcast and PLAY = yes
	TEMPERATURE = cool and HUMIDITY = normal

.

ในการทดสอบผลการประมวลผลข้อคำถามจะทำการประมวลผลด้วยโปรแกรม SQOARI และประมวลผลยังระบบจัดการฐานข้อมูลโดยตรง และจะเก็บเวลาที่ใช้ในการประมวลผลในแต่ละข้อคำถามทั้งหมด 5 ครั้งด้วยกัน อยู่ในหน่วยของมิลลิวินาที (ms) ในการทดสอบการประมวลผลข้อคำถามมีการแสดงผลการทดสอบแบ่งออกเป็นสองส่วนด้วยกัน คือ เวลาที่ใช้ในการประมวลผลข้อคำถามเดิม และเวลาที่ใช้ในการประมวลผลข้อคำถามที่มีการปรับปรุงประสิทธิภาพแล้ว (ในกรณีที่เงื่อนไขข้อคำถามมีความซ้ำซ้อนกัน) ซึ่งทำการทดสอบกับข้อมูลทั้ง 7 ชุด คือ Automobile, Car evaluation, Credit approve, Mushroom, Nursery, Solar flare และ Weather ตามลำดับ ในแต่ละชุดข้อมูลจะมีข้อคำถามที่ใช้ในการทดสอบครั้งนี้ดังนี้

- ชุดข้อมูล Automobile

  กฎข้อบังคับ คือ

  FUEL_SYSTEM = 2bbl -> ASPIRATION = std

  NUM_OF_CYLINDERS = four -> ENGINE_LOCATION = front

  ENGINE_TYPE = ohc -> ENGINE_LOCATION = front

  FUEL_TYPE = diesel -> FUEL_SYSTEM = idi

  SYMBOLING = 3 -> FUEL_TYPE = gas

  ข้อคำถามเดิม

  1. SELECT * FROM automobile
     WHERE FUEL_SYSTEM='2bbl' and ASPIRATION='std'
  2. SELECT * FROM automobile
     WHERE NUM_OF_CYLINDERS = 'four'
     and ENGINE_LOCATION = 'front'
  3. SELECT * FROM automobile
     WHERE ENGINE_TYPE = 'ohc' and ENGINE_LOCATION = 'front'
  4. SELECT * FROM automobile
     WHERE FUEL_TYPE = 'diesel' and FUEL_SYSTEM = 'idi'
  5. SELECT * FROM automobile
     WHERE SYMBOLING = '3' and FUEL_TYPE = 'gas'

  ข้อคำถามที่มีการปรับปรุงแล้ว

  1. SELECT * FROM automobile
     WHERE FUEL_SYSTEM='2bbl'

2. SELECT * FROM automobile
   WHERE NUM_OF_CYLINDERS = 'four'
3. SELECT * FROM automobile
   WHERE ENGINE_TYPE = 'ohc'
4. SELECT * FROM automobile
   WHERE FUEL_TYPE = 'diesel'
5. SELECT * FROM automobile
   WHERE SYMBOLING = '3'

- ชุดข้อมูล Car evaluation

  กฎข้อบังคับ คือ

  PERSONS = 2 -> GOAL = unacc

  SAFETY = low -> GOAL = unacc

  ข้อคำถามเดิม

  1. SELECT * FROM car_evaluation
     WHERE PERSONS='2' and GOAL='unacc'
  2. SELECT * FROM car_evaluation
     WHERE SAFETY='low' and GOAL='unacc'

  ข้อคำถามที่มีการปรับปรุงแล้ว

  1. SELECT * FROM car_evaluation
     WHERE PERSONS='2'
  2. SELECT * FROM car_evaluation
     WHERE SAFETY='low'

- ชุดข้อมูล Credit approve

  กฎข้อบังคับ คือ

  A4 = u -> A5 = g

  A4 = y -> A5 = p

ข้อคำถามเดิม

1. SELECT * FROM credit

   WHERE A4='u' and A5='g'

2. SELECT * FROM credit

   WHERE A4='y' and A5='p'

ข้อคำถามที่มีการปรับปรุงแล้ว

1. SELECT * FROM credit

   WHERE A4='u'

2. SELECT * FROM credit

   WHERE A4='y'

- ชุดข้อมูล Mushroom

  กฎข้อบังคับ คือ

  GILL_SPACING = c -> VEIL_TYPE = p

  GILL_ATTACHMENT = f -> VEIL_TYPE = p

  POPULATION = v -> VEIL_TYPE = p

  SPORE_PRINT_COLOR = n -> RING_NUMBER = o

  CAP_SHAPE = k -> VEIL_TYPE = p

  ข้อคำถามเดิม

  1. SELECT * FROM mushroom

     WHERE GILL_SPACING = 'c' and VEIL_TYPE = 'p'

  2. SELECT * FROM mushroom

     WHERE GILL_ATTACHMENT = 'f' and VEIL_TYPE = 'p'

  3. SELECT * FROM mushroom

     WHERE POPULATION = 'v' and VEIL_TYPE = 'p'

  4. SELECT * FROM mushroom

     WHERE SPORE_PRINT_COLOR = 'n' and RING_NUMBER = 'o'

  5. SELECT * FROM mushroom

     WHERE CAP_SHAPE = 'k' and VEIL_TYPE = 'p'

ข้อคำถามที่มีการปรับปรุงแล้ว

1. SELECT * FROM mushroom
   WHERE GILL_SPACING = 'c'
2. SELECT * FROM mushroom
   WHERE GILL_ATTACHMENT = 'f'
3. SELECT * FROM mushroom
   WHERE POPULATION = 'v'
4. SELECT * FROM mushroom
   WHERE SPORE_PRINT_COLOR = 'n'
5. SELECT * FROM mushroom
   WHERE CAP_SHAPE = 'k'

- ชุดข้อมูล Nursery

กฎข้อบังคับ คือ

GOAL = not_recom -> HEALTH = not_recom

ข้อคำถามเดิม

1. SELECT * FROM nursery
   WHERE GOAL='not_recom' and HEALTH='not_recom'

ข้อคำถามที่มีการปรับปรุงแล้ว

1. SELECT * FROM nursery
   WHERE GOAL='not_recom'

- ชุดข้อมูล Solar flare

กฎข้อบังคับ คือ

SPOT_DISTRIBUTION = O ->
REGION_BECOME_HISTORICALLY_COM = 2
SPOT_DISTRIBUTION = I ->
REGION_BECOME_HISTORICALLY_COM = 2
SPOT_DISTRIBUTION = X -> GOAL = H
REGION_BECOME_HISTORICALLY_COM = 1 -> GOAL = H
LARGEST_SPOT_SIZE = X -> GOAL = B

ข้อคำถามเดิม

1. SELECT * FROM solar_flare
   WHERE SPOT_DISTRIBUTION = 'O'
   and REGION_BECOME_HISTORICALLY_COM = '2'
2. SELECT * FROM solar_flare
   WHERE SPOT_DISTRIBUTION = 'I'
   and REGION_BECOME_HISTORICALLY_COM = '2'
3. SELECT * FROM solar_flare
   WHERE SPOT_DISTRIBUTION = 'X' and GOAL = 'H'
4. SELECT * FROM solar_flare
   WHERE REGION_BECOME_HISTORICALLY_COM = '1'
   and GOAL = 'H'
5. SELECT * FROM solar_flare
   WHERE LARGEST_SPOT_SIZE = 'X' and GOAL = 'B'

ข้อคำถามที่มีการปรับปรุงแล้ว

1. SELECT * FROM solar_flare
   WHERE SPOT_DISTRIBUTION = 'O'
2. SELECT * FROM solar_flare
   WHERE SPOT_DISTRIBUTION = 'I'
3. SELECT * FROM solar_flare
   WHERE SPOT_DISTRIBUTION = 'X'
4. SELECT * FROM solar_flare
   WHERE REGION_BECOME_HISTORICALLY_COM = '1'
5. SELECT * FROM solar_flare
   WHERE LARGEST_SPOT_SIZE = 'X'

- ชุดข้อมูล Weather

  กฎข้อบังคับ คือ

  OUTLOOK = overcast and PLAY = yes

  TEMPERATURE = cool and HUMIDITY = normal

ข้อคำถามเดิม

1. SELECT * FROM weather
   WHERE OUTLOOK = 'overcast' and PLAY = 'yes'
2. SELECT * FROM weather
   WHERE TEMPERATURE='cool' and HUMIDITY='normal'

ข้อคำถามที่มีการปรับปรุงแล้ว

1. SELECT * FROM weather
   WHERE OUTLOOK = 'overcast'
2. SELECT * FROM weather
   WHERE TEMPERATURE='cool'

4.2.1 การทดสอบการประมวลผลข้อคำถามกับระบบจัดการฐานข้อมูลโดยตรง

ในระบบจัดการฐานข้อมูลของ Oracle 10g Express Edition จะมีเมนูย่อยในส่วนของ SQL เรียกว่า SQL Commands เพื่อใช้ในการประมวลผลคำสั่งต่าง ๆ ไปยังระบบจัดการฐานข้อมูล และในเมนูส่วนนี้ สามารถแสดงเวลาที่ใช้ในการประมวลผลคำสั่งที่ส่งไปยังหน่วยประมวลผลได้ โดยสามารถแสดงผลการประมวลผลข้อคำถามเดิมในแต่ละชุดข้อมูลได้ดังตารางที่ 4.12, 4.14, 4.16, 4.18, 4.20, 4.22 และ 4.24 และผลการประมวลผลข้อคำถามที่มีการปรับปรุงประสิทธิภาพข้อคำถามแล้วแสดงได้ดังตารางที่ 4.13, 4.15, 4.17, 4.19, 4.21, 4.23 และ 4.25 ตามลำดับ

ตารางที่ 4.12 แสดงเวลาที่ใช้ในการประมวลผลข้อคำถามเดิมด้วยโปรแกรม SQL Commands บนระบบจัดการฐานข้อมูล Oracle 10g Express Edition ในชุดข้อมูล Automobile

ข้อคำถาม	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ครั้งที่ 4	ครั้งที่ 5	เวลาเฉลี่ย (ms)
ข้อคำถามที่ 1	316	286	294	307	301	300.8
ข้อคำถามที่ 2	586	662	653	587	686	634.8
ข้อคำถามที่ 3	530	597	575	511	570	556.6
ข้อคำถามที่ 4	92	92	70	70	83	81.4
ข้อคำถามที่ 5	113	118	110	120	108	113.8

ตารางที่ 4.13 แสดงเวลาที่ใช้ในการประมวลผลข้อคำถามที่มีการปรับปรุงประสิทธิภาพแล้วด้วยโปรแกรม SQL Commands บนระบบจัดการฐานข้อมูล Oracle 10g Express Edition ในชุดข้อมูล Automobile

ข้อคำถาม	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ครั้งที่ 4	ครั้งที่ 5	เวลาเฉลี่ย (ms)
ข้อคำถามที่ 1	297	227	225	225	226	240
ข้อคำถามที่ 2	556	627	560	626	633	600.4
ข้อคำถามที่ 3	539	597	508	574	570	557.6
ข้อคำถามที่ 4	81	087	072	81	81	80.4
ข้อคำถามที่ 5	110	113	107	110	110	110

ตารางที่ 4.14 แสดงเวลาที่ใช้ในการประมวลผลข้อคำถามเดิมด้วยโปรแกรม SQL Commands บนระบบจัดการฐานข้อมูล Oracle 10g Express Edition ในชุดข้อมูล Car evaluation

ข้อคำถาม	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ครั้งที่ 4	ครั้งที่ 5	เวลาเฉลี่ย (ms)
ข้อคำถามที่ 1	222	225	227	192	214	216
ข้อคำถามที่ 2	233	214	192	192	213	208.8

ตารางที่ 4.15 แสดงเวลาที่ใช้ในการประมวลผลข้อคำถามที่มีการปรับปรุงประสิทธิภาพแล้วด้วยโปรแกรม SQL Commands บนระบบจัดการฐานข้อมูล Oracle 10g Express Edition ในชุดข้อมูล Car evaluation

ข้อคำถาม	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ครั้งที่ 4	ครั้งที่ 5	เวลาเฉลี่ย (ms)
ข้อคำถามที่ 1	200	217	214	214	216	212.2
ข้อคำถามที่ 2	200	217	214	214	216	212.2

ตารางที่ 4.16 แสดงเวลาที่ใช้ในการประมวลผลข้อคำถามเดิมด้วยโปรแกรม SQL Commands บนระบบจัดการฐานข้อมูล Oracle 10g Express Edition ในชุดข้อมูล Credit approve

ข้อคำถาม	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ครั้งที่ 4	ครั้งที่ 5	เวลาเฉลี่ย (ms)
ข้อคำถามที่ 1	857	832	845	859	850	848.6
ข้อคำถามที่ 2	526	525	528	520	519	523.6

ตารางที่ 4.17 แสดงเวลาที่ใช้ในการประมวลผลข้อคำถามที่มีการปรับปรุงประสิทธิภาพแล้วด้วยโปรแกรม SQL Commands บนระบบจัดการฐานข้อมูล Oracle 10g Express Edition ในชุดข้อมูล Credit approve

ข้อคำถาม	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ครั้งที่ 4	ครั้งที่ 5	เวลาเฉลี่ย (ms)
ข้อคำถามที่ 1	690	695	689	690	689	690.6
ข้อคำถามที่ 2	235	234	229	237	230	233

ตารางที่ 4.18 แสดงเวลาที่ใช้ในการประมวลผลข้อคำถามเดิมด้วยโปรแกรม SQL Commands บนระบบจัดการฐานข้อมูล Oracle 10g Express Edition ในชุดข้อมูล Mushroom

ข้อคำถาม	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ครั้งที่ 4	ครั้งที่ 5	เวลาเฉลี่ย (ms)
ข้อคำถามที่ 1	761	768	765	772	768	766.8
ข้อคำถามที่ 2	719	722	720	718	715	718.8
ข้อคำถามที่ 3	345	346	340	347	341	343.8
ข้อคำถามที่ 4	210	209	202	215	211	209.4
ข้อคำถามที่ 5	89	87	85	93	90	88.8

ตารางที่ 4.19 แสดงเวลาที่ใช้ในการประมวลผลข้อคำถามที่มีการปรับปรุงประสิทธิภาพแล้วด้วยโปรแกรม SQL Commands บนระบบจัดการฐานข้อมูล Oracle 10g Express Edition ในชุดข้อมูล Mushroom

ข้อคำถาม	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ครั้งที่ 4	ครั้งที่ 5	เวลาเฉลี่ย (ms)
ข้อคำถามที่ 1	735	730	731	730	728	730.8
ข้อคำถามที่ 2	667	665	663	668	670	666.6
ข้อคำถามที่ 3	363	345	351	352	347	351.6
ข้อคำถามที่ 4	185	180	181	179	178	180.6
ข้อคำถามที่ 5	85	80	81	83	80	81.8

ตารางที่ 4.20 แสดงเวลาที่ใช้ในการประมวลผลข้อคำถามเดิมด้วยโปรแกรม SQL Commands บนระบบจัดการฐานข้อมูล Oracle 10g Express Edition ในชุดข้อมูล Nursery

ข้อคำถาม	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ครั้งที่ 4	ครั้งที่ 5	เวลาเฉลี่ย (ms)
ข้อคำถามที่ 1	574	574	573	572	575	573.6

ตารางที่ 4.21 แสดงเวลาที่ใช้ในการประมวลผลข้อคำถามที่มีการปรับปรุงประสิทธิภาพแล้วด้วยโปรแกรม SQL Commands บนระบบจัดการฐานข้อมูล Oracle 10g Express Edition ในชุดข้อมูล Nursery

ข้อคำถาม	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ครั้งที่ 4	ครั้งที่ 5	เวลาเฉลี่ย (ms)
ข้อคำถามที่ 1	339	339	339	338	337	338.4

ตารางที่ 4.22 แสดงเวลาที่ใช้ในการประมวลผลข้อคำถามเดิมด้วยโปรแกรม SQL Commands บนระบบจัดการฐานข้อมูล Oracle 10g Express Edition ในชุดข้อมูล Solar flare

ข้อคำถาม	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ครั้งที่ 4	ครั้งที่ 5	เวลาเฉลี่ย (ms)
ข้อคำถามที่ 1	289	300	301	287	295	294.4
ข้อคำถามที่ 2	140	140	142	145	140	141.4
ข้อคำถามที่ 3	161	165	162	161	161	162
ข้อคำถามที่ 4	67	65	67	65	67	66.2
ข้อคำถามที่ 5	89	89	87	90	87	88.4

ตารางที่ 4.23 แสดงเวลาที่ใช้ในการประมวลผลข้อคำถามที่มีการปรับปรุงประสิทธิภาพแล้วด้วยโปรแกรม SQL Commands บนระบบจัดการฐานข้อมูล Oracle 10g Express Edition ในชุดข้อมูล Solar flare

ข้อคำถาม	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ครั้งที่ 4	ครั้งที่ 5	เวลาเฉลี่ย (ms)
ข้อคำถามที่ 1	232	231	230	232	232	231.4
ข้อคำถามที่ 2	120	120	121	120	120	120.2
ข้อคำถามที่ 3	154	155	154	153	155	154.2
ข้อคำถามที่ 4	58	60	58	58	59	58.6
ข้อคำถามที่ 5	81	80	81	80	80	80.4

ตารางที่ 4.24 แสดงเวลาที่ใช้ในการประมวลผลข้อคำถามเดิมด้วยโปรแกรม SQL Commands บนระบบจัดการฐานข้อมูล Oracle 10g Express Edition ในชุดข้อมูล Weather

ข้อคำถาม	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ครั้งที่ 4	ครั้งที่ 5	เวลาเฉลี่ย (ms)
ข้อคำถามที่ 1	136	122	122	122	124	125.2
ข้อคำถามที่ 2	140	136	134	120	122	130.4

ตารางที่ 4.25 แสดงเวลาที่ใช้ในการประมวลผลข้อคำถามที่มีการปรับปรุงประสิทธิภาพแล้วด้วยโปรแกรม SQL Commands บนระบบจัดการฐานข้อมูล Oracle 10g Express Edition ในชุดข้อมูล Weather

ข้อคำถาม	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ครั้งที่ 4	ครั้งที่ 5	เวลาเฉลี่ย (ms)
ข้อคำถามที่ 1	136	135	134	122	134	132.2
ข้อคำถามที่ 2	136	134	135	122	134	132.2

4.2.2 การทดสอบการประมวลผลข้อคำถามด้วยโปรแกรม SQOARI

ในส่วนนี้จะทำการประมวลผลข้อคำถามผ่านทางโปรแกรม SQOARI ซึ่งเป็นโปรแกรมต้นแบบที่ผู้วิจัยทำการพัฒนาขึ้น เพื่อใช้ในการทดสอบการปรับปรุงประสิทธิภาพข้อคำถาม และทำการค้นหากฎความสัมพันธ์จากข้อมูลเพื่อนำกฎความสัมพันธ์ที่ได้มาสร้างเป็นกฎข้อบังคับ โดยผลการประมวลผลข้อคำถามเดิมในแต่ละชุดข้อมูลแสดงได้ดังตารางที่ 4.26, 4.28, 4.30, 4.32, 4.34, 4.36 และ 4.38 และผลการประมวลผลข้อคำถามที่มีการปรับปรุงประสิทธิภาพ (ในกรณีที่เงื่อนไขข้อคำถามมีความซ้ำซ้อนกัน) แสดงได้ดังตารางที่ 4.27, 4.29, 4.31, 4.33, 4.35, 4.37 และ 4.39 ตามลำดับ

ตารางที่ 4.26 แสดงเวลาที่ใช้ในการประมวลผลข้อคำถามเดิมด้วยโปรแกรม SQOARI บนระบบจัดการฐานข้อมูล Oracle 10g Express Edition ในชุดข้อมูล Automobile

ข้อคำถาม	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ครั้งที่ 4	ครั้งที่ 5	เวลาเฉลี่ย (ms)
ข้อคำถามที่ 1	250	281	281	234	234	256
ข้อคำถามที่ 2	210	203	219	156	156	188.8
ข้อคำถามที่ 3	201	203	203	187	171	193
ข้อคำถามที่ 4	182	171	172	187	156	173.6
ข้อคำถามที่ 5	235	219	188	234	250	225.2

ตารางที่ 4.27 แสดงเวลาที่ใช้ในการประมวลผลข้อคำถามที่มีการปรับปรุงประสิทธิภาพแล้วด้วยโปรแกรม SQOARI บนระบบจัดการฐานข้อมูล Oracle 10g Express Edition ในชุดข้อมูล Automobile

ข้อคำถาม	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ครั้งที่ 4	ครั้งที่ 5	เวลาเฉลี่ย (ms)
ข้อคำถามที่ 1	218	218	203	219	187	209
ข้อคำถามที่ 2	156	141	203	140	125	153
ข้อคำถามที่ 3	203	180	200	203	178	192.8
ข้อคำถามที่ 4	170	187	156	165	155	166.6
ข้อคำถามที่ 5	188	188	172	204	141	178.6

ตารางที่ 4.28 แสดงเวลาที่ใช้ในการประมวลผลข้อคำถามเดิมด้วยโปรแกรม SQOARI บนระบบจัดการฐานข้อมูล Oracle 10g Express Edition ในชุดข้อมูล Car evaluation

ข้อคำถาม	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ครั้งที่ 4	ครั้งที่ 5	เวลาเฉลี่ย (ms)
ข้อคำถามที่ 1	218	157	156	282	172	197
ข้อคำถามที่ 2	125	140	125	188	141	143.8

ตารางที่ 4.29 แสดงเวลาที่ใช้ในการประมวลผลข้อคำถามที่มีการปรับปรุงประสิทธิภาพแล้วด้วยโปรแกรม SQOARI บนระบบจัดการฐานข้อมูล Oracle 10g Express Edition ในชุดข้อมูล Car evaluation

ข้อคำถาม	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ครั้งที่ 4	ครั้งที่ 5	เวลาเฉลี่ย (ms)
ข้อคำถามที่ 1	157	188	156	187	156	168.8
ข้อคำถามที่ 2	94	94	94	78	109	93.8

ตารางที่ 4.30 แสดงเวลาที่ใช้ในการประมวลผลข้อคำถามเดิมด้วยโปรแกรม SQOARI บนระบบจัดการฐานข้อมูล Oracle 10g Express Edition ในชุดข้อมูล Credit approve

ข้อคำถาม	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ครั้งที่ 4	ครั้งที่ 5	เวลาเฉลี่ย (ms)
ข้อคำถามที่ 1	172	172	141	156	171	162.4
ข้อคำถามที่ 2	109	125	110	125	110	115.8

ตารางที่ 4.31 แสดงเวลาที่ใช้ในการประมวลผลข้อคำถามที่มีการปรับปรุงประสิทธิภาพแล้วด้วยโปรแกรม SQOARI บนระบบจัดการฐานข้อมูล Oracle 10g Express Edition ในชุดข้อมูล Credit approve

ข้อคำถาม	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ครั้งที่ 4	ครั้งที่ 5	เวลาเฉลี่ย (ms)
ข้อคำถามที่ 1	171	157	156	153	109	149.2
ข้อคำถามที่ 2	109	125	109	97	98	107.6

ตารางที่ 4.32 แสดงเวลาที่ใช้ในการประมวลผลข้อคำถามเดิมด้วยโปรแกรม SQOARI บนระบบจัดการฐานข้อมูล Oracle 10g Express Edition ในชุดข้อมูล Mushroom

ข้อคำถาม	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ครั้งที่ 4	ครั้งที่ 5	เวลาเฉลี่ย (ms)
ข้อคำถามที่ 1	406	438	391	360	391	397.2
ข้อคำถามที่ 2	391	375	344	329	312	350.2
ข้อคำถามที่ 3	391	375	375	360	328	365.8
ข้อคำถามที่ 4	349	329	375	375	375	360.6
ข้อคำถามที่ 5	328	344	312	375	406	353

ตารางที่ 4.33 แสดงเวลาที่ใช้ในการประมวลผลข้อคำถามที่มีการปรับปรุงประสิทธิภาพแล้วด้วยโปรแกรม SQOARI บนระบบจัดการฐานข้อมูล Oracle 10g Express Edition ในชุดข้อมูล Mushroom

ข้อคำถาม	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ครั้งที่ 4	ครั้งที่ 5	เวลาเฉลี่ย (ms)
ข้อคำถามที่ 1	319	360	359	360	375	354.6
ข้อคำถามที่ 2	344	359	313	359	329	340.8
ข้อคำถามที่ 3	390	297	359	359	328	346.6
ข้อคำถามที่ 4	390	328	324	344	328	342.8
ข้อคำถามที่ 5	344	391	344	312	343	346.8

ตารางที่ 4.34 แสดงเวลาที่ใช้ในการประมวลผลข้อคำถามเดิมด้วยโปรแกรม SQOARI บนระบบจัดการฐานข้อมูล Oracle 10g Express Edition ในชุดข้อมูล Nursery

ข้อคำถาม	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ครั้งที่ 4	ครั้งที่ 5	เวลาเฉลี่ย (ms)
ข้อคำถามที่ 1	266	219	219	234	234	234.4

ตารางที่ 4.35 แสดงเวลาที่ใช้ในการประมวลผลข้อคำถามที่มีการปรับปรุงประสิทธิภาพแล้วด้วยโปรแกรม SQOARI บนระบบจัดการฐานข้อมูล Oracle 10g Express Edition ในชุดข้อมูล Nursery

ข้อคำถาม	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ครั้งที่ 4	ครั้งที่ 5	เวลาเฉลี่ย (ms)
ข้อคำถามที่ 1	250	219	203	250	197	223.8

ตารางที่ 4.36 แสดงเวลาที่ใช้ในการประมวลผลข้อคำถามเดิมด้วยโปรแกรม SQOARI บนระบบจัดการฐานข้อมูล Oracle 10g Express Edition ในชุดข้อมูล Solar flare

ข้อคำถาม	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ครั้งที่ 4	ครั้งที่ 5	เวลาเฉลี่ย (ms)
ข้อคำถามที่ 1	318	332	309	318	318	319
ข้อคำถามที่ 2	188	188	182	189	188	187
ข้อคำถามที่ 3	188	190	189	191	191	189.8
ข้อคำถามที่ 4	120	125	120	127	128	124
ข้อคำถามที่ 5	175	179	178	179	179	178

ตารางที่ 4.37 แสดงเวลาที่ใช้ในการประมวลผลข้อคำถามที่มีการปรับปรุงประสิทธิภาพแล้วด้วยโปรแกรม SQOARI บนระบบจัดการฐานข้อมูล Oracle 10g Express Edition ในชุดข้อมูล Soloar flare

ข้อคำถาม	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ครั้งที่ 4	ครั้งที่ 5	เวลาเฉลี่ย (ms)
ข้อคำถามที่ 1	188	198	195	190	190	192.2
ข้อคำถามที่ 2	133	130	123	130	130	129.2
ข้อคำถามที่ 3	131	130	129	130	130	130
ข้อคำถามที่ 4	70	75	77	77	75	74.8
ข้อคำถามที่ 5	98	99	98	95	97	97.4

ตารางที่ 4.38 แสดงเวลาที่ใช้ในการประมวลผลข้อคำถามเดิมด้วยโปรแกรม SQOARI บนระบบจัดการฐานข้อมูล Oracle 10g Express Edition ในชุดข้อมูล Weather

ข้อคำถาม	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ครั้งที่ 4	ครั้งที่ 5	เวลาเฉลี่ย (ms)
ข้อคำถามที่ 1	185	140	188	141	125	155.8
ข้อคำถามที่ 2	62	78	93	78	62	74.6

ตารางที่ 4.39 แสดงเวลาที่ใช้ในการประมวลผลข้อคำถามที่มีการปรับปรุงประสิทธิภาพแล้วด้วยโปรแกรม SQOARI บนระบบจัดการฐานข้อมูล Oracle 10g Express Edition ในชุดข้อมูล Weather

ข้อคำถาม	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ครั้งที่ 4	ครั้งที่ 5	เวลาเฉลี่ย (ms)
ข้อคำถามที่ 1	157	141	141	156	156	150.2
ข้อคำถามที่ 2	78	78	63	78	47	68.8

ในการทดสอบกับโปรแกรม SQOARI นี้สามารถตรวจสอบข้อคำถามที่มีเงื่อนไขขัดแย้งกับกฎข้อบังคับที่มีได้ นั่นคือถ้าข้อคำถามใดมีเงื่อนไขที่ขัดแย้งกับกฎข้อบังคับแสดงว่าข้อมูลที่ต้องการค้นหานั้นไม่ปรากฏอยู่ในระบบจัดการฐานข้อมูล ดังนั้นโปรแกรมจะทำการแสดงผลลัพธ์ออกมาแจ้งผู้ใช้ได้ทันทีว่าไม่พบข้อมูลที่ต้องการค้นหา โดยยังไม่มีการประมวลผลข้อคำถามนั้นที่ระบบจัดการฐานข้อมูล จึงไม่เสียเวลาในการประมวลผลข้อคำถาม และไม่ใช้ทรัพยากรของระบบอย่างเปล่าประโยชน์ เวลาที่ได้จากการประมวลผลข้อคำถามเหล่านี้ผ่านทางโปรแกรม SQOARI จึงมีค่าเป็นศูนย์เสมอ โดยข้อคำถามที่ใช้ในการทดสอบในกรณีนี้มีทั้งหมดดังนี้

- ชุดข้อมูล Automobile

  กฎข้อบังคับ คือ

  FUEL_SYSTEM = 2bbl -> ASPIRATION = std

  NUM_OF_CYLINDERS = four -> ENGINE_LOCATION = front

  ENGINE_TYPE = ohc -> ENGINE_LOCATION = front

  FUEL_TYPE = diesel -> FUEL_SYSTEM = idi

  SYMBOLING = 3 -> FUEL_TYPE = gas

  ข้อคำถามในการทดสอบ

  1. SELECT * FROM automobile
     WHERE FUEL_SYSTEM='2bbl' and ASPIRATION='turbo'
  2. SELECT * FROM automobile
     WHERE NUM_OF_CYLINDERS = 'four'
     and ENGINE_LOCATION = 'rear'
  3. SELECT * FROM automobile
     WHERE ENGINE_TYPE = 'ohc'
     and ENGINE_LOCATION = 'rear'
  4. SELECT * FROM automobile
     WHERE FUEL_TYPE = 'diesel' and FUEL_SYSTEM = 'mfi'
  5. SELECT * FROM automobile
     WHERE SYMBOLING = '3' and FUEL_TYPE = 'diesel'

- ชุดข้อมูล Car evaluation

  กฎข้อบังคับ คือ

  PERSONS = 2 -> GOAL = unacc

  SAFETY = low -> GOAL = unacc

  ข้อคำถามเดิม

  1. SELECT * FROM car_evaluation
     WHERE PERSONS='2' and GOAL='acc'
  2. SELECT * FROM car_evaluation
     WHERE SAFETY='low' and GOAL='acc'

- ชุดข้อมูล Credit approve

  กฎข้อบังคับ คือ

  A4 = u -> A5 = g

  A4 = y -> A5 = p

  ข้อคำถามเดิม

  1. SELECT * FROM credit
     WHERE A4='u' and A5='gg'
  2. SELECT * FROM credit
     WHERE A4='y' and A5='gg'

- ชุดข้อมูล Mushroom

  กฎข้อบังคับ คือ

  GILL_SPACING = c -> VEIL_TYPE = p

  GILL_ATTACHMENT = f -> VEIL_TYPE = p

  POPULATION = v -> VEIL_TYPE = p

  SPORE_PRINT_COLOR = n -> RING_NUMBER = o

  CAP_SHAPE = k -> VEIL_TYPE = p

  ข้อคำถามเดิม

  1. SELECT * FROM mushroom
     WHERE GILL_SPACING = 'c' and VEIL_TYPE = 'u'
  2. SELECT * FROM mushroom
     WHERE GILL_ATTACHMENT = 'f' and VEIL_TYPE = 'u'
  3. SELECT * FROM mushroom
     WHERE POPULATION = 'v' and VEIL_TYPE = 'u'
  4. SELECT * FROM mushroom
     WHERE SPORE_PRINT_COLOR = 'n' and RING_NUMBER = 'n'
  5. SELECT * FROM mushroom
     WHERE CAP_SHAPE = 'k' and VEIL_TYPE = 'u'

- ชุดข้อมูล Nursery

กฎข้อบังคับ คือ

GOAL = not_recom -> HEALTH = not_recom

ข้อคำถามเดิม

1. SELECT * FROM nursery
   WHERE GOAL='not_recom' and HEALTH='priority'

- ชุดข้อมูล Solar flare

กฎข้อบังคับ คือ

SPOT_DISTRIBUTION = O ->
REGION_BECOME_HISTORICALLY_COM = 2
SPOT_DISTRIBUTION = I ->
REGION_BECOME_HISTORICALLY_COM = 2
SPOT_DISTRIBUTION = X -> GOAL = H
REGION_BECOME_HISTORICALLY_COM = 1 -> GOAL = H
LARGEST_SPOT_SIZE = X -> GOAL = B

ข้อคำถามเดิม

1. SELECT * FROM solar_flare
   WHERE SPOT_DISTRIBUTION = 'O'
   and REGION_BECOME_HISTORICALLY_COM = '1'
2. SELECT * FROM solar_flare
   WHERE SPOT_DISTRIBUTION = 'I'
   and REGION_BECOME_HISTORICALLY_COM = '1'
3. SELECT * FROM solar_flare
   WHERE SPOT_DISTRIBUTION = 'X' and GOAL = 'A'
4. SELECT * FROM solar_flare
   WHERE REGION_BECOME_HISTORICALLY_COM = '1'
   and GOAL = 'A'
5. SELECT * FROM solar_flare
   WHERE LARGEST_SPOT_SIZE = 'X' and GOAL = 'A'

- ชุดข้อมูล Weather

  กฎข้อบังคับ คือ

  OUTLOOK = overcast and PLAY = yes

  TEMPERATURE = cool and HUMIDITY = normal

  ข้อคำถามเดิม

  1. SELECT * FROM weather

     WHERE OUTLOOK = 'overcast' and PLAY = 'no'

  2. SELECT * FROM weather

     WHERE TEMPERATURE='cool' and HUMIDITY='high'

ผลการประมวลผลข้อคำถามข้างต้นแสดงได้ดังตารางที่ 4.40 ถึง 4.46

ตารางที่ 4.40 แสดงเวลาที่ใช้ในการประมวลผลข้อคำถามที่เงื่อนไขมีความขัดแย้งกับกฎข้อบังคับด้วยโปรแกรม SQOARI บนระบบจัดการฐานข้อมูล Oracle 10g Express Edition ในชุดข้อมูล Automobile

ข้อคำถาม	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ครั้งที่ 4	ครั้งที่ 5	เวลาเฉลี่ย (ms)
ข้อคำถามที่ 1	1000	975	1215	996	1054	1048
ข้อคำถามที่ 2	813	845	820	794	777	809.8
ข้อคำถามที่ 3	687	685	649	699	623	668.6
ข้อคำถามที่ 4	687	689	685	679	691	686.2
ข้อคำถามที่ 5	656	645	675	650	625	650.2

ตารางที่ 4.41 แสดงเวลาที่ใช้ในการประมวลผลข้อคำถามที่เงื่อนไขมีความขัดแย้งกับกฎข้อบังคับด้วยโปรแกรม SQOARI บนระบบจัดการฐานข้อมูล Oracle 10g Express Edition ในชุดข้อมูล Car evaluation

ข้อคำถาม	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ครั้งที่ 4	ครั้งที่ 5	เวลาเฉลี่ย (ms)
ข้อคำถามที่ 1	297	250	246	289	295	275.4
ข้อคำถามที่ 2	266	267	275	250	251	261.8

ตารางที่ 4.42 แสดงเวลาที่ใช้ในการประมวลผลข้อคำถามที่เงื่อนไขมีความขัดแย้งกับกฎข้อบังคับด้วยโปรแกรม SQOARI บนระบบจัดการฐานข้อมูล Oracle 10g Express Edition ในชุดข้อมูล Credit approve

ข้อคำถาม	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ครั้งที่ 4	ครั้งที่ 5	เวลาเฉลี่ย (ms)
ข้อคำถามที่ 1	391	380	389	397	395	390.4
ข้อคำถามที่ 2	250	246	248	255	253	250.4

ตารางที่ 4.43 แสดงเวลาที่ใช้ในการประมวลผลข้อคำถามที่เงื่อนไขมีความขัดแย้งกับกฎข้อบังคับด้วยโปรแกรม SQOARI บนระบบจัดการฐานข้อมูล Oracle 10g Express Edition ในชุดข้อมูล Mushroom

ข้อคำถาม	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ครั้งที่ 4	ครั้งที่ 5	เวลาเฉลี่ย (ms)
ข้อคำถามที่ 1	2203	2205	2210	2200	2211	2205.8
ข้อคำถามที่ 2	2156	2158	2160	2151	2156	2156.2
ข้อคำถามที่ 3	2265	2263	2264	2260	2269	2264.2
ข้อคำถามที่ 4	2343	2345	2349	2341	2341	2343.8
ข้อคำถามที่ 5	2266	2267	2269	2261	2263	2265.2

ตารางที่ 4.44 แสดงเวลาที่ใช้ในการประมวลผลข้อคำถามที่เงื่อนไขมีความขัดแย้งกับกฎข้อบังคับด้วยโปรแกรม SQOARI บนระบบจัดการฐานข้อมูล Oracle 10g Express Edition ในชุดข้อมูล Nursery

ข้อคำถาม	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ครั้งที่ 4	ครั้งที่ 5	เวลาเฉลี่ย (ms)
ข้อคำถามที่ 1	953	950	964	948	945	952

ตารางที่ 4.45 แสดงเวลาที่ใช้ในการประมวลผลข้อคำถามที่เงื่อนไขมีความขัดแย้งกับกฎข้อบังคับด้วยโปรแกรม SQOARI บนระบบจัดการฐานข้อมูล Oracle 10g Express Edition ในชุดข้อมูล Solar flare

ข้อคำถาม	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ครั้งที่ 4	ครั้งที่ 5	เวลาเฉลี่ย (ms)
ข้อคำถามที่ 1	1172	1170	1175	1172	1172	1172.2
ข้อคำถามที่ 2	1266	1265	1266	1264	1266	1265.4
ข้อคำถามที่ 3	1172	175	1178	1175	1172	974.4
ข้อคำถามที่ 4	1171	1171	1174	1174	1176	1173.2
ข้อคำถามที่ 5	1063	1065	1063	1064	1063	1063.6

ตารางที่ 4.46 แสดงเวลาที่ใช้ในการประมวลผลข้อคำถามที่เงื่อนไขมีความขัดแย้งกับกฎข้อบังคับด้วยโปรแกรม SQOARI บนระบบจัดการฐานข้อมูล Oracle 10g Express Edition ในชุดข้อมูล Weather

ข้อคำถาม	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ครั้งที่ 4	ครั้งที่ 5	เวลาเฉลี่ย (ms)
ข้อคำถามที่ 1	172	170	175	174	172	172.6
ข้อคำถามที่ 2	188	189	187	188	188	188

4.3 ผลการทดสอบข้อคำถามบนระบบจัดการฐานข้อมูล Microsoft SQL Server 2000

การทดสอบการประมวลผลข้อคำถามในด้วยระบบจัดการฐานข้อมูลของ Microsoft SQL Sever 2000 จะใช้ชุดข้อมูล กฎข้อบังคับ และข้อคำถามเดียวกันกับการทดสอบด้วยระบบจัดการฐานข้อมูลของ Oracle 10g Express Edition ดังกล่าวไว้แล้วในหัวข้อที่ 4.2 ซึ่งการทดสอบในหัวข้อนี้แบ่งการทดสอบออกเป็นสองส่วนคือ การทดสอบการประมวลผลข้อคำถามด้วยตัวจัดการระบบฐานข้อมูลโดยตรง และการทดสอบการประมวลผลข้อคำถามด้วยโปรแกรม SQOARI

4.3.1 การทดสอบการประมวลผลข้อคำถามกับระบบจัดการฐานข้อมูลโดยตรง

การทดสอบผลการประมวลผลข้อคำถามในส่วนของหัวข้อนี้ จะใช้โปรแกรม Query Analyzer ซึ่งเป็นโปรแกรมที่ติดตั้งมาพร้อมกับระบบจัดการฐานข้อมูล Microsoft SQL Server 2000 การทดสอบกับโปรแกรม Query Analyzer ต้องมีการกำหนดค่า Set Statistics time ก่อนและหลังข้อคำถามที่ต้องการประมวลผลดังรูปที่ 4.5 เพื่อให้โปรแกรมแสดงเวลาที่ใช้ในการประมวลผลข้อคำถามนั้นออกมา โดยผลการทดสอบการประมวลผลข้อคำถามเดิมของข้อมูลแต่ละชุดแสดงได้ดัง

ตารางที่ 4.47, 4.49, 4.51, 4.53, 4.55, 4.57 และ 4.59 และผลการทดสอบการประมวลผลข้อคำถามที่มีการปรับปรุงประสิทธิภาพแล้วแสดงได้ดังตารางที่ 4.48, 4.50, 4.52, 4.54, 4.56, 4.58 และ 4.60

```
Set Statistics time on
SELECT …
Set Statistics time off
```

รูปที่ 4.5 แสดงการเขียนคำสั่งแสดงเวลาที่ใช้ในการประมวลผลบน
ระบบจัดการฐานข้อมูล Microsoft SQL Server 2000

ตารางที่ 4.47 แสดงเวลาที่ใช้ในการประมวลผลข้อคำถามเดิมด้วยโปรแกรม Query Analyzer บน ระบบจัดการฐานข้อมูล Microsoft SQL Server 2000 ในชุดข้อมูล Automobile

ข้อคำถาม	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ครั้งที่ 4	ครั้งที่ 5	เวลาเฉลี่ย (ms)
ข้อคำถามที่ 1	1350	1480	1447	1489	1519	1457
ข้อคำถามที่ 2	1611	1563	1691	1624	1758	1649.4
ข้อคำถามที่ 3	1613	1738	1466	1631	1663	1622.2
ข้อคำถามที่ 4	1230	1350	1445	972	1364	1272.2
ข้อคำถามที่ 5	1426	1444	1344	1268	1477	1391.8

ตารางที่ 4.48 แสดงเวลาที่ใช้ในการประมวลผลข้อคำถามที่มีการปรับปรุงประสิทธิภาพแล้วด้วย โปรแกรม Query Analyzer บนระบบจัดการฐานข้อมูล Microsoft SQL Server 2000 ในชุดข้อมูล Automobile

ข้อคำถาม	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ครั้งที่ 4	ครั้งที่ 5	เวลาเฉลี่ย (ms)
ข้อคำถามที่ 1	1301	1348	1368	1351	1387	1351
ข้อคำถามที่ 2	1644	1615	1583	1640	1650	1626.4
ข้อคำถามที่ 3	1593	1423	1603	1578	1518	1543
ข้อคำถามที่ 4	1201	1253	1261	1254	1256	1245
ข้อคำถามที่ 5	1336	1270	1114	1254	1204	1235.6

ตารางที่ 4.49 แสดงเวลาที่ใช้ในการประมวลผลข้อคำถามเดิมด้วยโปรแกรม Query Analyzer บนระบบจัดการฐานข้อมูล Microsoft SQL Server 2000 ในชุดข้อมูล Car evaluation

ข้อคำถาม	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ครั้งที่ 4	ครั้งที่ 5	เวลาเฉลี่ย (ms)
ข้อคำถามที่ 1	2020	1010	951	1035	1001	1203.4
ข้อคำถามที่ 2	1154	1133	1164	1180	1047	1135.6

ตารางที่ 4.50 แสดงเวลาที่ใช้ในการประมวลผลข้อคำถามที่มีการปรับปรุงประสิทธิภาพแล้วด้วยโปรแกรม Query Analyzer บนระบบจัดการฐานข้อมูล Microsoft SQL Server 2000 ในชุดข้อมูล Car evaluation

ข้อคำถาม	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ครั้งที่ 4	ครั้งที่ 5	เวลาเฉลี่ย (ms)
ข้อคำถามที่ 1	965	1112	1052	1018	1074	1044.2
ข้อคำถามที่ 2	1112	998	956	1013	996	1015

ตารางที่ 4.51 แสดงเวลาที่ใช้ในการประมวลผลข้อคำถามเดิมด้วยโปรแกรม Query Analyzer บนระบบจัดการฐานข้อมูล Microsoft SQL Server 2000 ในชุดข้อมูล Credit approve

ข้อคำถาม	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ครั้งที่ 4	ครั้งที่ 5	เวลาเฉลี่ย (ms)
ข้อคำถามที่ 1	1066	1392	1207	1123	1070	1171.6
ข้อคำถามที่ 2	978	899	906	1108	1367	1051.6

ตารางที่ 4.52 แสดงเวลาที่ใช้ในการประมวลผลข้อคำถามที่มีการปรับปรุงประสิทธิภาพแล้วด้วยโปรแกรม Query Analyzer บนระบบจัดการฐานข้อมูล Microsoft SQL Server 2000 ในชุดข้อมูล Credit approve

ข้อคำถาม	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ครั้งที่ 4	ครั้งที่ 5	เวลาเฉลี่ย (ms)
ข้อคำถามที่ 1	1188	1017	1079	1178	1096	1111.6
ข้อคำถามที่ 2	956	845	952	946	1031	946

ตารางที่ 4.53 แสดงเวลาที่ใช้ในการประมวลผลข้อคำถามเดิมด้วยโปรแกรม Query Analyzer บนระบบจัดการฐานข้อมูล Microsoft SQL Server 2000 ในชุดข้อมูล Mushroom

ข้อคำถาม	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ครั้งที่ 4	ครั้งที่ 5	เวลาเฉลี่ย (ms)
ข้อคำถามที่ 1	7432	9158	8606	7942	7763	8180.2
ข้อคำถามที่ 2	9763	11378	11921	10270	11061	10878.6
ข้อคำถามที่ 3	10176	7741	10642	7261	7888	8741.6
ข้อคำถามที่ 4	6343	4937	6052	8974	8074	6876
ข้อคำถามที่ 5	5666	5151	6467	7376	5541	6040.2

ตารางที่ 4.54 แสดงเวลาที่ใช้ในการประมวลผลข้อคำถามที่มีการปรับปรุงประสิทธิภาพแล้วด้วยโปรแกรม Query Analyzer บนระบบจัดการฐานข้อมูล Microsoft SQL Server 2000 ในชุดข้อมูล Mushroom

ข้อคำถาม	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ครั้งที่ 4	ครั้งที่ 5	เวลาเฉลี่ย (ms)
ข้อคำถามที่ 1	7071	7389	7523	8071	7219	7454.6
ข้อคำถามที่ 2	8811	8586	8586	8467	8614	8612.8
ข้อคำถามที่ 3	8041	8606	6703	6426	6572	7269.6
ข้อคำถามที่ 4	6288	6055	5183	6236	5767	5905.8
ข้อคำถามที่ 5	5513	4766	4775	4493	5429	4995.2

ตารางที่ 4.55 แสดงเวลาที่ใช้ในการประมวลผลข้อคำถามเดิมด้วยโปรแกรม Query Analyzer บนระบบจัดการฐานข้อมูล Microsoft SQL Server 2000 ในชุดข้อมูล Nursery

ข้อคำถาม	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ครั้งที่ 4	ครั้งที่ 5	เวลาเฉลี่ย (ms)
ข้อคำถามที่ 1	2187	2311	1841	2174	2354	2173.4

ตารางที่ 4.56 แสดงเวลาที่ใช้ในการประมวลผลข้อคำถามที่มีการปรับปรุงประสิทธิภาพแล้วด้วยโปรแกรม Query Analyzer บนระบบจัดการฐานข้อมูล Microsoft SQL Server 2000 ในชุดข้อมูล Nursery

ข้อคำถาม	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ครั้งที่ 4	ครั้งที่ 5	เวลาเฉลี่ย (ms)
ข้อคำถามที่ 1	1585	1578	1511	1939	1581	1638.8

ตารางที่ 4.57 แสดงเวลาที่ใช้ในการประมวลผลข้อคำถามเดิมด้วยโปรแกรม Query Analyzer บนระบบจัดการฐานข้อมูล Microsoft SQL Server 2000 ในชุดข้อมูล Solar flare

ข้อคำถาม	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ครั้งที่ 4	ครั้งที่ 5	เวลาเฉลี่ย (ms)
ข้อคำถามที่ 1	5249	4773	5080	5491	4147	4948
ข้อคำถามที่ 2	4959	5763	5752	5754	5586	5562.8
ข้อคำถามที่ 3	4196	4775	5111	4214	3566	4372.4
ข้อคำถามที่ 4	6749	6220	6370	6713	7591	6728.6
ข้อคำถามที่ 5	5645	5927	5442	5457	5345	5563.2

ตารางที่ 4.58 แสดงเวลาที่ใช้ในการประมวลผลข้อคำถามที่มีการปรับปรุงประสิทธิภาพแล้วด้วยโปรแกรม Query Analyzer บนระบบจัดการฐานข้อมูล Microsoft SQL Server 2000 ในชุดข้อมูล Solar flare

ข้อคำถาม	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ครั้งที่ 4	ครั้งที่ 5	เวลาเฉลี่ย (ms)
ข้อคำถามที่ 1	4296	4389	4520	4224	4900	4465.8
ข้อคำถามที่ 2	4101	4033	4201	3749	3918	4000.4
ข้อคำถามที่ 3	3642	3841	3826	3703	3518	3706
ข้อคำถามที่ 4	6234	6289	6316	6727	6098	6332.8
ข้อคำถามที่ 5	5510	5439	5293	5124	5066	5286.4

ตารางที่ 4.59 แสดงเวลาที่ใช้ในการประมวลผลข้อคำถามเดิมด้วยโปรแกรม Query Analyzer บนระบบจัดการฐานข้อมูล Microsoft SQL Server 2000 ในชุดข้อมูล Weather

ข้อคำถาม	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ครั้งที่ 4	ครั้งที่ 5	เวลาเฉลี่ย (ms)
ข้อคำถามที่ 1	414	293	203	214	220	268.8
ข้อคำถามที่ 2	166	108	97	153	144	133.6

ตารางที่ 4.60 แสดงเวลาที่ใช้ในการประมวลผลข้อคำถามที่มีการปรับปรุงประสิทธิภาพแล้วด้วยโปรแกรม Query Analyzer บนระบบจัดการฐานข้อมูล Microsoft SQL Server 2000 ในชุดข้อมูล Weather

ข้อคำถาม	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ครั้งที่ 4	ครั้งที่ 5	เวลาเฉลี่ย (ms)
ข้อคำถามที่ 1	146	160	92	115	84	119.4
ข้อคำถามที่ 2	113	92	87	93	89	94.8

4.3.2 การทดสอบการประมวลผลข้อคำถามด้วยโปรแกรม SQOARI

การทดสอบนี้จะใส่ข้อคำถามผ่านทางโปรแกรม SQOARI จากนั้นโปรแกรมจึงจะทำการเชื่อมต่อและส่งข้อคำถามไปประมวลผลยังระบบจัดการฐานข้อมูล และจึงส่งผลของข้อคำถามมายังโปรแกรมเพื่อแสดงผลที่ได้ ซึ่งผลการทดสอบข้อคำถามเดิมแสดงดังตารางที่ 4.61, 4.63, 4.65, 4.67, 4.69, 4.71 และ 4.73 ผลการทดสอบข้อคำถามที่มีการปรับปรุงประสิทธิภาพแล้ว (ในกรณีที่เงื่อนไขข้อคำถามมีความซ้ำซ้อนกัน) แสดงดังตารางที่ 4.62, 4.64, 4.66, 4.68, 4.70, 4.72 และ 4.74 และผลการทดสอบข้อคำถามที่มีเงื่อนไขขัดแย้งกับกฎข้อบังคับแสดงดังตารางที่ 4.75, 4.76, 4.77, 4.78, 4.79, 4.80 และ 4.81

ตารางที่ 4.61 แสดงเวลาที่ใช้ในการประมวลผลข้อคำถามเดิมด้วยโปรแกรม SQOARI บนระบบจัดการฐานข้อมูล Microsoft SQL Server 2000 ในชุดข้อมูล Automobile

ข้อคำถาม	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ครั้งที่ 4	ครั้งที่ 5	เวลาเฉลี่ย (ms)
ข้อคำถามที่ 1	2343	3296	2234	2140	2328	2468.2
ข้อคำถามที่ 2	2390	2828	2188	2140	3187	2546.6
ข้อคำถามที่ 3	2406	2172	2312	2469	2625	2396.8
ข้อคำถามที่ 4	2188	1922	1828	1766	2016	1944
ข้อคำถามที่ 5	2219	1828	2266	1953	2312	2115.6

ตารางที่ 4.62 แสดงเวลาที่ใช้ในการประมวลผลข้อคำถามที่มีการปรับปรุงประสิทธิภาพแล้วด้วยโปรแกรม SQOARI บนระบบจัดการฐานข้อมูล Microsoft SQL Server 2000 ในชุดข้อมูล Automobile

ข้อคำถาม	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ครั้งที่ 4	ครั้งที่ 5	เวลาเฉลี่ย (ms)
ข้อคำถามที่ 1	1235	1016	984	1032	1140	1081.4
ข้อคำถามที่ 2	1500	1187	1656	1203	1797	1468.6
ข้อคำถามที่ 3	1125	1187	2734	1875	1188	1621.8
ข้อคำถามที่ 4	1204	953	1031	1703	1328	1243.8
ข้อคำถามที่ 5	969	1312	1125	1062	953	1084.2

ตารางที่ 4.63 แสดงเวลาที่ใช้ในการประมวลผลข้อคำถามเดิมด้วยโปรแกรม SQOARI บนระบบจัดการฐานข้อมูล Microsoft SQL Server 2000 ในชุดข้อมูล Car evaluation

ข้อคำถาม	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ครั้งที่ 4	ครั้งที่ 5	เวลาเฉลี่ย (ms)
ข้อคำถามที่ 1	1750	719	719	1422	765	1075
ข้อคำถามที่ 2	1937	1469	703	1078	703	1178

ตารางที่ 4.64 แสดงเวลาที่ใช้ในการประมวลผลข้อคำถามที่มีการปรับปรุงประสิทธิภาพแล้วด้วยโปรแกรม SQOARI บนระบบจัดการฐานข้อมูล Microsoft SQL Server 2000 ในชุดข้อมูล Car evaluation

ข้อคำถาม	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ครั้งที่ 4	ครั้งที่ 5	เวลาเฉลี่ย (ms)
ข้อคำถามที่ 1	579	1234	1188	1297	469	690.6
ข้อคำถามที่ 2	594	547	672	1203	1813	716.6

ตารางที่ 4.65 แสดงเวลาที่ใช้ในการประมวลผลข้อคำถามเดิมด้วยโปรแกรม SQOARI บนระบบจัดการฐานข้อมูล Microsoft SQL Server 2000 ในชุดข้อมูล Credit approve

ข้อคำถาม	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ครั้งที่ 4	ครั้งที่ 5	เวลาเฉลี่ย (ms)
ข้อคำถามที่ 1	1735	1578	1703	1672	1797	1697
ข้อคำถามที่ 2	1297	1360	1532	1484	1610	1456.6

ตารางที่ 4.66 แสดงเวลาที่ใช้ในการประมวลผลข้อคำถามที่มีการปรับปรุงประสิทธิภาพแล้วด้วยโปรแกรม SQOARI บนระบบจัดการฐานข้อมูล Microsoft SQL Server 2000 ในชุดข้อมูล Credit approve

ข้อคำถาม	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ครั้งที่ 4	ครั้งที่ 5	เวลาเฉลี่ย (ms)
ข้อคำถามที่ 1	1297	1282	1640	1297	1047	1312.6
ข้อคำถามที่ 2	1281	1016	1407	1172	1188	1212.8

ตารางที่ 4.67 แสดงเวลาที่ใช้ในการประมวลผลข้อคำถามเดิมด้วยโปรแกรม SQOARI บนระบบจัดการฐานข้อมูล Microsoft SQL Server 2000 ในชุดข้อมูล Mushroom

ข้อคำถาม	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ครั้งที่ 4	ครั้งที่ 5	เวลาเฉลี่ย (ms)
ข้อคำถามที่ 1	8266	12172	8031	5407	6234	8022
ข้อคำถามที่ 2	18625	8703	7875	5890	4407	9100
ข้อคำถามที่ 3	8344	6750	4922	7125	4406	6309.4
ข้อคำถามที่ 4	6297	7047	5328	5672	4015	5671.8
ข้อคำถามที่ 5	7828	6454	6562	6688	2578	6022

ตารางที่ 4.68 แสดงเวลาที่ใช้ในการประมวลผลข้อคำถามที่มีการปรับปรุงประสิทธิภาพแล้วด้วยโปรแกรม SQOARI บนระบบจัดการฐานข้อมูล Microsoft SQL Server 2000 ในชุดข้อมูล Mushroom

ข้อคำถาม	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ครั้งที่ 4	ครั้งที่ 5	เวลาเฉลี่ย (ms)
ข้อคำถามที่ 1	5719	4016	5656	5875	5689	5391
ข้อคำถามที่ 2	4344	5076	4500	5750	4634	4860.8
ข้อคำถามที่ 3	6485	4972	4250	5953	5100	5139.2
ข้อคำถามที่ 4	5343	4406	4078	4172	4221	4444
ข้อคำถามที่ 5	4719	6578	5312	4485	4714	4821

ตารางที่ 4.69 แสดงเวลาที่ใช้ในการประมวลผลข้อคำถามเดิมด้วยโปรแกรม SQOARI บนระบบจัดการฐานข้อมูล Microsoft SQL Server 2000 ในชุดข้อมูล Nursery

ข้อคำถาม	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ครั้งที่ 4	ครั้งที่ 5	เวลาเฉลี่ย (ms)
ข้อคำถามที่ 1	4250	5188	4610	2641	5797	4497.2

ตารางที่ 4.70 แสดงเวลาที่ใช้ในการประมวลผลข้อคำถามที่มีการปรับปรุงประสิทธิภาพแล้วด้วยโปรแกรม SQOARI บนระบบจัดการฐานข้อมูล Microsoft SQL Server 2000 ในชุดข้อมูล Nursery

ข้อคำถาม	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ครั้งที่ 4	ครั้งที่ 5	เวลาเฉลี่ย (ms)
ข้อคำถามที่ 1	2907	2125	2656	2514	2278	2496

ตารางที่ 4.71 แสดงเวลาที่ใช้ในการประมวลผลข้อคำถามเดิมด้วยโปรแกรม SQOARI บนระบบจัดการฐานข้อมูล Microsoft SQL Server 2000 ในชุดข้อมูล Solar flare

ข้อคำถาม	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ครั้งที่ 4	ครั้งที่ 5	เวลาเฉลี่ย (ms)
ข้อคำถามที่ 1	8515	7938	3766	8219	4250	6537.6
ข้อคำถามที่ 2	2390	9359	6531	4766	6594	5928
ข้อคำถามที่ 3	5609	9219	5672	6844	4750	6418.8
ข้อคำถามที่ 4	9063	7719	5781	4281	7985	6965.8
ข้อคำถามที่ 5	10484	6344	5406	5625	4781	6528

ตารางที่ 4.72 แสดงเวลาที่ใช้ในการประมวลผลข้อคำถามที่มีการปรับปรุงประสิทธิภาพแล้วด้วยโปรแกรม SQOARI บนระบบจัดการฐานข้อมูล Microsoft SQL Server 2000 ในชุดข้อมูล Solar flare

ข้อคำถาม	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ครั้งที่ 4	ครั้งที่ 5	เวลาเฉลี่ย (ms)
ข้อคำถามที่ 1	3829	2641	3250	1625	2750	2819
ข้อคำถามที่ 2	3609	2471	2297	1781	2484	2528.4
ข้อคำถามที่ 3	5204	2500	1734	2500	2421	2871.8
ข้อคำถามที่ 4	2265	2641	1891	4187	2375	2671.8
ข้อคำถามที่ 5	2266	2104	2422	2531	2156	2295.8

ตารางที่ 4.73 แสดงเวลาที่ใช้ในการประมวลผลข้อคำถามเดิมด้วยโปรแกรม SQOARI บนระบบจัดการฐานข้อมูล Microsoft SQL Server 2000 ในชุดข้อมูล Weather

ข้อคำถาม	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ครั้งที่ 4	ครั้งที่ 5	เวลาเฉลี่ย (ms)
ข้อคำถามที่ 1	1188	609	562	672	609	728
ข้อคำถามที่ 2	844	703	672	640	640	699.8

ตารางที่ 4.74 แสดงเวลาที่ใช้ในการประมวลผลข้อคำถามที่มีการปรับปรุงประสิทธิภาพแล้วด้วยโปรแกรม SQOARI บนระบบจัดการฐานข้อมูล Microsoft SQL Server 2000 ในชุดข้อมูล Weather

ข้อคำถาม	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ครั้งที่ 4	ครั้งที่ 5	เวลาเฉลี่ย (ms)
ข้อคำถามที่ 1	516	422	406	422	422	437.6
ข้อคำถามที่ 2	437	422	360	375	391	397

ตารางที่ 4.75 แสดงเวลาที่ใช้ในการประมวลผลข้อคำถามที่เงื่อนไขมีความขัดแย้งกับกฎข้อบังคับด้วยโปรแกรม SQOARI บนระบบจัดการฐานข้อมูล Microsoft SQL Server 2000 ในชุดข้อมูล Automobile

ข้อคำถาม	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ครั้งที่ 4	ครั้งที่ 5	เวลาเฉลี่ย (ms)
ข้อคำถามที่ 1	2938	2172	1640	2141	1704	2119
ข้อคำถามที่ 2	1375	1687	1703	1797	2906	1893.6
ข้อคำถามที่ 3	1688	1860	2047	2312	1890	1959.4
ข้อคำถามที่ 4	1672	1656	2062	1641	1860	1778.2
ข้อคำถามที่ 5	1438	1984	1968	2047	1843	1856

ตารางที่ 4.76 แสดงเวลาที่ใช้ในการประมวลผลข้อคำถามที่เงื่อนไขมีความขัดแย้งกับกฎข้อบังคับด้วยโปรแกรม SQOARI บนระบบจัดการฐานข้อมูล Microsoft SQL Server 2000 ในชุดข้อมูล Car evaluation

ข้อคำถาม	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ครั้งที่ 4	ครั้งที่ 5	เวลาเฉลี่ย (ms)
ข้อคำถามที่ 1	907	750	1266	641	1297	972.2
ข้อคำถามที่ 2	1532	1078	1187	1391	890	1215.6

ตารางที่ 4.77 แสดงเวลาที่ใช้ในการประมวลผลข้อคำถามที่เงื่อนไขมีความขัดแย้งกับกฎข้อบังคับด้วยโปรแกรม SQOARI บนระบบจัดการฐานข้อมูล Microsoft SQL Server 2000 ในชุดข้อมูล Credit approve

ข้อคำถาม	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ครั้งที่ 4	ครั้งที่ 5	เวลาเฉลี่ย (ms)
ข้อคำถามที่ 1	1579	1516	1531	1609	2469	1740.8
ข้อคำถามที่ 2	1563	1391	2031	2297	2985	2053.4

ตารางที่ 4.78 แสดงเวลาที่ใช้ในการประมวลผลข้อคำถามที่เงื่อนไขมีความขัดแย้งกับกฎข้อบังคับด้วยโปรแกรม SQOARI บนระบบจัดการฐานข้อมูล Microsoft SQL Server 2000 ในชุดข้อมูล Mushroom

ข้อคำถาม	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ครั้งที่ 4	ครั้งที่ 5	เวลาเฉลี่ย (ms)
ข้อคำถามที่ 1	8578	6297	7938	7750	6531	7418.8
ข้อคำถามที่ 2	9671	4344	4594	5297	7000	6181.2
ข้อคำถามที่ 3	4032	4094	4532	5375	7391	5084.8
ข้อคำถามที่ 4	6656	4015	3845	4953	5719	5037.6
ข้อคำถามที่ 5	15703	4696	4516	5844	6569	7465.6

ตารางที่ 4.79 แสดงเวลาที่ใช้ในการประมวลผลข้อคำถามที่เงื่อนไขมีความขัดแย้งกับกฎข้อบังคับด้วยโปรแกรม SQOARI บนระบบจัดการฐานข้อมูล Microsoft SQL Server 2000 ในชุดข้อมูล Nursery

ข้อคำถาม	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ครั้งที่ 4	ครั้งที่ 5	เวลาเฉลี่ย (ms)
ข้อคำถามที่ 1	5218	3265	3438	3734	2609	3652.8

ตารางที่ 4.80 แสดงเวลาที่ใช้ในการประมวลผลข้อคำถามที่เงื่อนไขมีความขัดแย้งกับกฎข้อบังคับด้วยโปรแกรม SQOARI บนระบบจัดการฐานข้อมูล Microsoft SQL Server 2000 ในชุดข้อมูล Solar flare

ข้อคำถาม	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ครั้งที่ 4	ครั้งที่ 5	เวลาเฉลี่ย (ms)
ข้อคำถามที่ 1	2985	6656	2812	3797	5203	4290.6
ข้อคำถามที่ 2	4703	2672	3734	4781	7375	4653
ข้อคำถามที่ 3	4719	3297	2875	5696	3796	4076.6
ข้อคำถามที่ 4	3766	3265	4565	5406	4438	4288
ข้อคำถามที่ 5	6219	5390	4015	5407	4296	5065.4

ตารางที่ 4.81 แสดงเวลาที่ใช้ในการประมวลผลข้อคำถามที่เงื่อนไขมีความขัดแย้งกับกฎข้อบังคับด้วยโปรแกรม SQOARI บนระบบจัดการฐานข้อมูล Microsoft SQL Server 2000 ในชุดข้อมูล Weather

ข้อคำถาม	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ครั้งที่ 4	ครั้งที่ 5	เวลาเฉลี่ย (ms)
ข้อคำถามที่ 1	1625	797	813	906	532	934.6
ข้อคำถามที่ 2	812	765	750	594	672	718.6

4.4 อภิปรายผล

ผลการทดสอบเวลาในการประมวลผลข้อคำถามในระบบจัดการฐานข้อมูล Oracle 10g Express Edition และ Microsoft SQL Server 2000 ทั้งในทดสอบกับระบบจัดการฐานข้อมูลโดยตรงหรือผ่านทางโปรแกรม SQOARI สามารถสรุปผลได้ดังนี้

4.4.1 ผลการทดสอบการประมวลผลข้อคำถามด้วยระบบจัดการฐานข้อมูลโดยตรง

1) ความแตกต่างของเวลาในการประมวลผลของข้อคำถามเดิม และข้อคำถามที่ปรับปรุงรูปแบบแล้วจะมีผลต่างของเวลามากหรือน้อยขึ้นอยู่กับจำนวนข้อมูลที่ต้องการค้นหาว่ามีปรากฏอยู่ในฐานข้อมูลจำนวนมากน้อยเพียงใด นั่นคือ ข้อคำถามในกรณีที่เงื่อนไขมีความซ้ำซ้อนกันเมื่อทำการปรับปรุงรูปแบบแล้วจะมีประสิทธิภาพในการประมวลผลมากที่สุด ก็ต่อเมื่อ กฎข้อบังคับที่ใช้ในการปรับปรุงนั้นสร้างมาจากกฎความสัมพันธ์ที่มีค่าสนับสนุนจำนวนมาก ซึ่งเราสามารถกำหนดได้ในขั้นตอนการค้นหากฎความสัมพันธ์เพื่อนำมาสร้างเป็นกฎข้อบังคับ

2) เวลาที่ได้จากการประมวลผลข้อคำถามจากระบบจัดการฐานข้อมูล Oracle มีความละเอียดของหน่วยเวลาไม่เพียงพอ และมีหน้าการจัดการระบบฐานข้อมูลที่ทำงานผ่านทางเว็บบราวเซอร์ (Web browser) จึงทำให้เกิดความล่าช้าในการประมวลผลไปบ้างเนื่องจากฐานข้อมูลตัวนี้เป็นฐานข้อมูลที่มีขนาดเล็ก รวมทั้งสภาวะของเครื่องคอมพิวเตอร์ที่ใช้ในการทดสอบนั้น อาจประมวลผลข้อคำถามในแต่ละครั้งที่แตกต่างกัน เนื่องจากอาจมีสิ่งแวดล้อมอื่น ๆ ที่มีผลกระทบกับระบบ ณ ขณะนั้น จึงทำให้ค่าเวลาที่ได้มีความผิดพลาดอยู่บ้าง

4.4.2 ผลการทดสอบการประมวลผลข้อคำถามด้วยโปรแกรม SQOARI

1) เวลาที่ได้จากการทดสอบการประมวลผลข้อคำถามผ่านทางโปรแกรม SQOARI จะเห็นความแตกต่างของเวลาที่ใช้ในการประมวลผลอย่างชัดเจน ในกรณีที่เงื่อนไขของข้อคำถามนั้นมีความขัดแย้งกับกฎข้อบังคับที่มี เพราะโปรแกรมไม่มีการส่งข้อคำถามนั้นไปประมวลผลยังระบบจัดการฐานข้อมูล แต่ในโปรแกรมจะแจ้งให้ผู้ใช้ทราบทันทีว่าไม่มีข้อมูลที่ต้องการค้นหาในฐานข้อมูล

2) สำหรับกรณีที่เงื่อนไขของข้อคำถามเกิดความซ้ำซ้อน ความแตกต่างของเวลาที่ใช้ในการประมวลผลข้อคำถามเดิม และข้อคำถามที่มีการเปลี่ยนรูปแบบแล้วจะมีความเปลี่ยนแปลงมากน้อยเพียงใดขึ้นอยู่กับจำนวนข้อมูลที่ต้องการค้นหาจากข้อคำถามนั้น ดังนั้น ถ้าข้อมูลที่ต้องการค้นหามีจำนวนแถวของข้อมูลมากจะทำให้เห็นความแตกต่างของเวลาชัดเจนยิ่งขึ้น

บทที่ 5
บทสรุป

ระบบจัดการฐานข้อมูลกลายเป็นสิ่งจำเป็นที่ทุกองค์กร หรือบริษัทต้องมีไว้เพื่อเก็บข้อมูลต่าง ๆ เพื่อนำไปวิเคราะห์ หรือประมวลผลให้เกิดประโยชน์ในด้านต่าง ๆ กับองค์กร หรือบริษัทนั้น ซึ่งข้อมูลที่จัดเก็บอยู่ในฐานข้อมูลอาจมีอยู่ในปริมาณมหาศาล และมีการเรียกใช้งานข้อมูลเหล่านั้นได้จากหลายแหล่งด้วยกัน ดังนั้นระบบจัดการฐานข้อมูลจะนำข้อคำถามที่ถูกส่งเข้ามาประมวลผลเพื่อค้นหาสิ่งที่ผู้ใช้ต้องการ และส่งค่าที่ค้นพบกลับออกมาแสดงให้ผู้ใช้ทราบ ซึ่งในการส่งข้อคำถามเข้ามาประมวลผลยังระบบจัดการฐานข้อมูล อาจมาจากเครื่องลูกข่าย (Client) หลายเครื่องและเกิดขึ้นในเวลาเดียวกัน ดังนั้นถ้ามีข้อคำถามที่ไม่สมบูรณ์ถูกส่งเข้ามาประมวลผลด้วยอาจทำให้ระบบจัดการฐานข้อมูลทำงานได้ไม่เต็มประสิทธิภาพ และใช้เวลาที่มากกว่าปกติทำให้เกิดการใช้ทรัพยากรของเครื่องคอมพิวเตอร์โดยเปล่าประโยชน์ได้ ดังนั้นจึงต้องมีส่วนในการนำข้อคำถามที่ผู้ใช้ป้อนเข้ามาตรวจสอบความถูกต้องให้สมบูรณ์ที่สุดเสียก่อน เพื่อให้ระบบจัดการฐานข้อมูลทำงานได้เต็มประสิทธิภาพมากที่สุด

โดยงานวิจัยชิ้นนี้ มีจุดมุ่งหมายที่จะพัฒนาเทคนิคการตรวจสอบข้อคำถามให้สมบรูณ์ที่สุดก่อนที่จะนำข้อคำถามไปประมวลผลยังระบบจัดการฐานข้อมูล การปรับปรุงประสิทธิภาพของข้อคำถามนี้สามารถทำได้หลายรูปแบบ แต่การปรับปรุงประสิทธิภาพข้อคำถามในงานวิจัยชิ้นนี้จะนำเทคนิคการปรับปรุงประสิทธิภาพข้อคำถามเชิงความหมาย (Semantic query optimization) มาใช้ในการตรวจสอบความสมบรูณ์ของข้อคำถามก่อน โดยเทคนิคการปรับปรุงประสิทธิภาพนี้จำเป็นต้องใช้กฎข้อบังคับเพื่อช่วยในการตรวจสอบความสมบรูณ์ของข้อคำถาม ซึ่งในงานวิจัยชิ้นนี้จะทำการค้นหากฎข้อบังคับจากความสัมพันธ์ของข้อมูลในฐานข้อมูล โดยการนำเทคโนโลยีทางด้านการทำเหมืองข้อมูลมาใช้ในการค้นหารูปแบบของข้อมูล หลังจากนั้นจึงนำความสัมพันธ์ที่ได้มาสร้างเป็นกฎข้อบังคับเพื่อใช้ในกระบวนการปรับปรุงประสิทธิภาพต่อไป เพื่อให้ข้อคำถามที่ผู้ใช้ต้องการสอบถามใช้เวลาในการประมวลผลที่น้อยลง และให้คำตอบที่ถูกต้องเหมือนเดิม

ขั้นตอนในการพัฒนางานวิจัยชิ้นนี้ มีการพัฒนาโปรแกรม SQOARI ขึ้นมาเพื่อช่วยในการทดสอบการประมวลผลของข้อคำถาม และใช้ในการค้นหากฎความสัมพันธ์ของข้อมูล ซึ่งเป็นเพียงโปรแกรมต้นแบบเพื่อใช้ในขั้นตอนการทดสอบผลเท่านั้น

ขั้นตอนการทดสอบประสิทธิภาพของข้อคำถามที่มีการปรับปรุงแล้ว จะทำการทดสอบกับข้อมูลทั้งหมดเจ็ดชุด โดยจะตรวจสอบความถูกต้องของกฎข้อบังคับที่ได้จากโปรแกรม SQOARI ที่พัฒนาขึ้นมาเปรียบเทียบกับกฎความสัมพันธ์ที่ได้จากโปรแกรม WEKA หลังจากนั้นจะทำการทดสอบประสิทธิภาพของข้อคำถาม ซึ่งจะทดสอบกับระบบจัดการฐานข้อมูลด้วยกันสองระบบ คือ Oracle 10g Express Edition และ Microsoft SQL Server 2000 การนำข้อคำถามมาประมวลผลจะใช้การประมวลผลที่ตัวระบบจัดการฐานข้อมูลโดยตรง และประมวลผลบนโปรแกรม SQOARI โดยจะทำการจับเวลาที่ใช้ในการประมวลผลในแต่ละข้อคำถามจำนวนห้าครั้งด้วยกัน และนำเวลาที่ได้จากการประมวลผลทั้งในนส่วนของข้อคำถามเดิม และข้อคำถามที่มีการปรับปรุงประสิทธิภาพแล้วมาเปรียบเทียบดูความแตกต่างของเวลาที่ใช้ไป

5.1 สรุปผลการวิจัย

ผลการทดสอบสรุปได้ดังนี้

1) ในการใช้โปรแกรม SQOARI ที่พัฒนาขึ้นค้นหากฎความสัมพันธ์จากข้อมูลเพื่อนำกฎความสัมพันธ์ที่ได้มาสร้างเป็นกฎข้อบังคับนั้น โดยการกำหนดค่าสนับสนุนขั้นต่ำตั้งแต่ 10%, 20%, 30%, 40% และ 50% ตามลำดับ เมื่อนำกฎความสัมพันธ์ที่ได้จากการค้นหาด้วยโปรแกรม SQOARI มาเปรียบเทียบกับกฎความสัมพันธ์ที่ได้จากการค้นหาด้วยโปรแกรม WEKA แล้ว ผลปรากฏว่ากฎความสัมพันธ์ที่ได้จากโปรแกรม SQOARI มีค่าตรงกันกับกฎความสัมพันธ์ที่ได้จากโปรแกรม WEKA นั่นคือ กฎความสัมพันธ์ที่ได้จากโปรแกรม SQOARI มีความถูกต้อง 100 เปอร์เซ็นต์

2) การทดสอบประสิทธิภาพในการประมวลผลข้อคำถาม ในกรณีที่เกิดความซ้ำซ้อนของเงื่อนไขภายในข้อคำถามนั้น เมื่อทำการปรับปรุงด้วยกฎข้อบังคับที่ตรงกับเงื่อนไขแล้ว ปรากฏว่าผลของเวลาที่ใช้ในการประมวลผลของข้อคำถามเดิม และข้อคำถามที่ปรับปรุงประสิทธิภาพแล้วจะเกิดความแตกต่างกันของเวลาที่ใช้ในการประมวลผลมากหรือน้อยเพียงใด จะขึ้นอยู่กับจำนวนผลลัพธ์ที่ได้จากการค้นหาของข้อคำถามนั้น นั่นคือ ถ้าข้อคำถามมีจำนวนแถวที่ค้นพบอยู่เป็นจำนวนมากแล้วผลของเวลาที่ใช้ในการประมวลผลของข้อคำถามเดิม และข้อคำถามที่ปรับปรุงแล้วจะมีความแตกต่างกันมากขึ้นด้วย ดังนั้น ในการเลือกกฎความสัมพันธ์เพื่อนำมาสร้างเป็นกฎข้อบังคับ ควรมีการกำหนดค่าสนับสนุนให้กับกฎความสัมพันธ์ที่ต้องการค้นหาให้มีค่าที่เหมาะสมและไม่ควรกำหนดค่าน้อยจนเกินไป

3) การทดสอบประสิทธิภาพในการประมวลผลข้อถาม ในกรณีที่เงื่อนไขของข้อคำถามมีความขัดแย้งกับกฎข้อบังคับจะเห็นความแตกต่างของเวลาที่ชัดเจน เนื่องจากข้อคำถามประเภทนี้ถ้ามีเงื่อนไขที่เกิดความขัดแย้งกับกฎข้อบังคับแล้ว แสดงว่าข้อมูลที่ต้องการค้นหาด้วยข้อคำถามนั้นไม่ปรากฏอยู่ในฐานข้อมูลอย่างแน่นอน ถ้าข้อคำถามประเภทนี้ถูกส่งเข้าไปประมวลผลยังระบบจัดการฐานข้อมูลจะทำให้เกิดการประมวลผลที่ศูนย์เปล่า ส่งผลให้ระบบจัดการฐานข้อมูลทำงานได้ไม่เต็มประสิทธิภาพ ดังนั้นโปรแกรม SQOARI จึงทำการตรวจสอบข้อคำถามประเภทนี้ก่อนที่จะถูกส่งไปประมวลผลยังระบบจัดการฐานข้อมูล เมื่อพบข้อคำถามประเภทนี้โปรแกรมจะแจ้งให้ผู้ใช้ทราบทันทีโดยไม่มีการส่งข้อคำถามไปประมวลผลยังระบบจัดการฐานข้อมูล

5.2 การประยุกต์งานวิจัย

งานวิจัยชิ้นนี้เป็นการนำเทคโนโลยีสองด้านที่มีอยู่แล้วในปัจจุบัน และเป็นเทคโนโลยีที่รู้จักกันอย่างแพร่หลายมาทำงานร่วมกันจนเกิดเป็นแนวทางใหม่ในการพัฒนาระบบจัดการฐานข้อมูล เพื่อให้มีประสิทธิภาพในการทำงานมากยิ่งขึ้น งานวิจัยชิ้นนี้จึงเป็นแนวทางอีกแนวทางหนึ่งในการคิดค้นวิธีที่จะสร้างกฎข้อบังคับเพื่อมาใช้งานในการปรับปรุงข้อคำถามเชิงความหมายโดยการนำกฎความสัมพันธ์ที่ได้จากการทำเหมืองข้อมูลมาใช้เป็นกฎข้อบังคับ เพื่อนำไปใช้ในการปรับปรุงประสิทธิภาพข้อคำถาม ซึ่งงานวิจัยชิ้นนี้ยังคงเป็นเพียงแนวทางเบื้องต้น ดังนั้นจึงสามารถทำการวิจัยเพื่อพัฒนาต่อไปให้มีประสิทธิภาพมากยิ่งขึ้น จึงน่าจะเป็นประโยชน์สำหรับนักวิจัยในอนาคตเพื่อเป็นจุดเริ่มต้นในการพัฒนางานวิจัยลักษณะนี้ต่อไป และเมื่อมีการพัฒนาต่อไปที่ดีขึ้นแล้วอาจนำเอาความรู้ใหม่นี้มาประยุกต์รวมเข้ากับระบบจัดการฐานข้อมูลต่อไปในอนาคตได้

5.3 ปัญหาและข้อเสนอแนะ

1) การทดสอบประสิทธิภาพในการประมวลผลของข้อคำถาม ที่ใช้สำหรับการทดสอบนั้น เมื่อส่งข้อคำถามไปประมวลยังระบบจัดการฐานข้อมูลแล้วระบบจัดการฐานข้อมูลจะมีระบบที่ใช้ในการจัดการเกี่ยวกับการปรับปรุงประสิทธิภาพ หรือจัดเก็บสถิติการใช้งานข้อคำถามต่าง ๆ ที่ส่งเข้ามายังระบบจัดการฐานข้อมูล ดังนั้นเวลาที่ได้จากการประมวลผลข้อคำถามที่ส่งไปทดสอบจึงมีค่าไม่คงที่เท่าที่ควร และเวลาที่ใช้ในการประมวลผลจะลดน้อยลงไปเรื่อย ๆ จนกระทั่งเป็นเวลาที่ใช้ในการประมวลผลที่น้อยที่สุด จึงเป็นการยากที่จะทำการจับเวลาเพื่อนำเวลาที่ได้มาเปรียบเทียบกัน ดังนั้น จึงจับเวลาที่ใช้ในการประมวลผลข้อคำถามครั้งแรกเสมอ เพื่อนำมาใช้ในการเปรียบเทียบ และจะทำการสร้างโครงสร้าง (Schema) ในการจัดเก็บข้อมูลขึ้นใหม่เพื่อใช้ในการจับเวลาครั้งต่อไป

2) การทดสอบประสิทธิภาพข้อคำถาม โดยการส่งข้อคำถามผ่านทางโปรแกรม SQOARI จะเกิดปัญหาในการจับเวลาขึ้นบางครั้ง เนื่องจากภาษาที่ใช้ในการพัฒนาโปรแกรมใช้เวลาในการประมวลผลคำสั่งสำหรับจำค่าของเวลา หรือคำสั่งที่ใช้ส่งข้อคำถามไปยังระบบฐานข้อมูล หรือคำสั่งที่รับค่าผลลัพธ์มาจากฐานข้อมูล ซึ่งในแต่ละครั้งที่ทำการประมวลผลนั้นอาจให้ค่าที่ไม่เท่ากัน จึงทำให้เวลาที่ได้จากการประมวลผลผ่านโปรแกรม SQOARI นี้มีความไม่ชัดเจนเกิดขึ้น ทำให้ต้องทำการทดสอบกรณีนี้หลายครั้งเพื่อเก็บข้อมูลที่มีความใกล้เคียงกันมากที่สุดในการทดสอบ

3) สภาวะของเครื่องคอมพิวเตอร์ที่ใช้ในการทดสอบการประมวลผลข้อคำถามในแต่ละครั้งอาจมีสภาวะแวดล้อมอื่น ๆ ส่งผลเกี่ยวข้องระหว่างการทำการประมวลผลข้อคำถามที่ระบบจัดการฐานข้อมูล จึงเป็นปัจจัยที่ทำให้เวลาที่ใช้ในการประมวลผลแต่ละครั้งไม่เท่ากัน และบางครั้งมีความผิดผลาดไปจากเดิมมาก

4) การนำกฎความสัมพันธ์ที่ได้จากการทำเหมืองข้อมูลมาใช้สร้างเป็นกฎข้อบังคับนี้ ยังคงเป็นเพียงกฎความสัมพันธ์แบบง่าย นั่นคือ มีเพียงหนึ่งคอลัมน์ที่เป็นเหตุ และหนึ่งคอลัมน์ที่เป็นผล ซึ่งกฎความสัมพันธ์ที่ได้จากการทำเหมืองข้อมูลสามารถอยู่ในรูปของจำนวนคอลัมน์หลายคอลัมน์ที่เป็นเหตุและจำนวนคอลัมน์หลายคอลัมน์ที่เป็นผล จึงมีกฎความสัมพันธ์ที่น่าสนใจอีกมากมายที่จะนำมาใช้เพื่อสร้างเป็นกฎข้อบังคับในการนำไปปรับปรุงประสิทธิภาพข้อคำถามเชิงความหมายเพื่อให้กฎข้อบังคับมีความครอบคลุมกับข้อมูลในฐานข้อมูลมากยิ่งขึ้น

5) การทดสอบนี้ ยังคงเป็นเพียงการค้นหากฎความสัมพันธ์ของข้อมูลที่มาจากข้อมูลเพียงหนึ่งตารางเท่านั้น ดังนั้นถ้านำข้อมูลมาจากหลายตารางที่มีความสัมพันธ์กันมาใช้ในการค้นหากฎความสัมพันธ์จะทำให้ได้กฎความสัมพันธ์ที่สามารถนำมาสร้างเป็นกฎข้อบังคับเพื่อใช้ในการปรับปรุงประสิทธิภาพของเงื่อนไขข้อคำถามที่มีการสอบถามข้อมูลจากหลายตาราง ให้มีเวลาที่ใช้ในการประมวลผลที่น้อยลง หรือในกรณีที่ไม่มีข้อมูลที่ต้องการสอบถามอยู่ในฐานข้อมูลก็จะทำให้ไม่ต้องเสียเวลาในการประมวลผลข้อคำถามนั้นโดยเปล่าประโยชน์

รายการอ้างอิง

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 algorithm for mining association rules. **Proceedings of the 20th International Conference on Very Large Data Bases (VLDB'94)**, pp. 487-499.

Chaudhuri, S., Narasayya, V., and Sarawagi, S. (2002). Efficient evaluation of queries with mining predicates. **Proceedings of International conference on Data Engineering**. pp. 529-540.

Cheng, Q., Grayz, J., Koo, F., Leung, C., Liu, L., Qian, X. and Schiefer, B. (1999). Implementation of two semantic query optimization techniques in DB2 Universal Database. **Proceedings of the 25th International Conference on Very Large Data Bases (VLDB'99)**, pp. 687-698.

Frank, E., Hall, M., Holmes, G., Martin, B., Mayo, M., Pfahringer, B., Smith, T. and Witten, I. (2005). **Waikato Environment for Knowledge Analysis: Weka-3-4-7**. University of Waikato: New Zealand.

Hsu, C-N. and Knoblock, C.A. (1994). Rule induction for semantic query optimization. **Proceedings of 11th International Conference on Machine Learning**, pp. 112-120.

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

Ioannidis, Y.E. (1996). Query optimization. **ACM Computing Surveys**, Vol.28, No.1, pp. 121-123.

Jarke, M. and Koch, J. (1984). Query optimization in database systems. **ACM Computing Surveys**, Vol.16, No. 2, pp. 111-152.

Shekhar, S., Hamidzadeh, B., Kohli, A. and Coyle, M. (1993). Learning transformation rules for semantic query optimization : A data-driven approach. **IEEE Transactions on Knowledge and Data Engineering**, Vol.5, No.6, pp. 950-964.

Siegel, M., Sciore, E. and Salveter, S. (1992). A method for automatic rule derivation to support semantic query optimization. **ACM Transactions on Database Systems (TODS)**, Vol.17, No.4, pp. 563-600.

Chakravarthy, U.S., Grant, J. and Minker J. (1990). Logic-based approach to semantic query optimization. **ACM Transactions on Database Systems (TODS)**, Vol.15, No.2, pp.162-207.

ภาคผนวก ก

บทความผลงานวิจัยที่นำเสนอในการประชุมวิชาการวิทยาศาสตร์และเทคโนโลยีแห่งประเทศไทย ครั้งที่ 33

การปรับปรุงประสิทธิภาพข้อคำถามเชิงความหมายด้วยการอุปนัยกฎความสัมพันธ์

SEMANTIC QUERY OPTIMIZATION WITH ASSOCIATION RULE INDUCTION

อภิชัย ฤทธิ์ธงชัยเลิศ, นิตยา เกิดประสพ และ กิตติศักด์ เกิดประสพ

Apichai Rintthongchailert, Nittaya Kerdprasop and Kittisak Kerdprasop

Data Engineering and Knowledge Discovery (DEKD) Research Unit, School of Computer Engiineering, Suranaree University of Technology, Muang, Nakhon Ratchasima, 30000.

บทคัดย่อ: การปรับปรุงประสิทธิภาพข้อคำถามเชิงความหมาย หมายถึง การนำข้อคำถามเดิมมาจัดรูปแบบใหม่ ให้มีรูปประโยคที่แตกต่างกับข้อคำถามเดิม แต่ยังคงให้ผลลัพธ์ที่เหมือนเดิม สิ่งที่แตกต่างกันของทั้งสองข้อคำถาม คือ เวลาที่ใช้ในการประมวลผล เพื่อตอบคำถามนั้นจะใช้เวลาน้อยลงกว่าเดิม ความสมบูรณ์ของการปรับปรุงข้อคำถามเชิงความหมายนี้ จะขึ้นอยู่กับเงื่อนไขหรือกฎข้อบังคับที่จะนำมาเพิ่มหรือลดตัวประโยคเงื่อนไขของข้อคำถามโดยทั่วไปแล้วกฎข้อบังคับที่นำมาใช้ในการปรับปรุงข้อคำถามเชิงความหมายนี้ จะได้มาจากผู้เขียนโปรแกรมฐานข้อมูล ซึ่งอาจจะไม่ครอบคลุมกับข้อมูลทั้งหมดที่มีอยู่ในฐานข้อมูล ดังนั้น ในงานวิจัยนี้จึงนำเอาเทคโนโลยีการขุดค้นความรู้จากฐานข้อมูล ซึ่งเป็นเทคโนโลยีที่เป็นที่รู้จักกันอย่างแพร่หลาย โดยนำมาเฉพาะส่วนของการค้นหากฎความสัมพันธ์ของข้อมูล มาประยุกต์ใช้เพื่อทำงานร่วมกับการปรับปรุงประสิทธิภาพข้อคำถามเชิงความหมาย เพื่อลดเวลาในการประมวลผลข้อคำถามนั้น

Abstract: Semantic query optimization is the process of transforming a given query into a semantically equivalent one that still returns the same answer for any database state satisfying query's constraints. The different of both queries is lower execution cost of the transformed one. The efficiently optimized query depends on semantic constraints or integrity constraints which are used to remove a useless condition in a query's where clause. Basically, integrity constraint is defined by user. It may not cover all data in the database. Therefore, this paper aims at presenting the utilization of a well known data mining technique, association mining, to assist the semantic query optimization process.

Introduction: The increase speed of query execution in the database management system is important for database development. One popular and well known method is query optimization.

Some queries may contain confusing condition in where clause or condition may conflict with data in the database such that no answer the exists for that query. Then it is wasteful to execute such query. Therefore, it is our aim to adjust pattern of query before sending it to database management system for execution. The adjustment or transformation is based on semantic constraint induced with a data mining technique.

Methodology: We can find semantic constraint by take a dataset on the database into data mining process. In data mining process, to find association of data. We use apriori algorithm [1, 2] in the data mining process. So the results of mining give association rule in the form of IF – THEN rule (cause column to result column). But we adjust pattern of rule to become simple rule that the IF part contains one cause and the THEN part contains one result as follow:

If column_1 = ‘value_1’ then column_2 = ‘value_2’

In applying algorithm, we must specify the minimum confidence value and the minimum support value. In this paper, we define minimum confidence to be 1 or 100% accurate for any induced rule to guarantee correctness in query answering and we define minimum support to be zero for finding all association among data in the database. In query optimization process, we get semantic constraint from database for comparing query condition in the where clause. In this process we divide query into two types [3]: query with redundant condition and query with conflict condition. For the first type of the given query, the condition that happens to be redundant will be removed before processing the query. Figure 1 shows an example of this case.

Original query:	select * from table_name
	where column_1 = ‘A’ and column_2 = ‘B’;
Association rule:	column_1 = ‘A’ => column_2 = ‘B’
Optimized query:	select * from table_name
	where column = ‘A’;

Figure 1 Query with redundant condition

For the second case, query condition may conflict to the semantic constraint induced from the database. Then, the optimizer can produce immediate answer and save processing time of the database management system. Figure 2 show the example of conflicting case.

Original query:	select * from table_name where column_1 = ‘S’ and column_2 = ‘T’;
Association rule:	column_1 = ‘S’ => column_2 = ‘Q’
Optimized query:	No execution.

Figure 2 Query with conflicting condition

Results, Discussion and Conclusion: In our experiment, we use Microsoft SQL Server 2000 database, tested on Pentium IV 3.0 GHz with RAM 512 MB machine. The data sets used in our experiment are synthetic data and data taken from the UCI Repository(http://www.ics.uci.edu/~mlearn/MLRepository.html). We compared execution time of original query with the optimize query. Table 1 shows the execution time for the second type of query: query with conflicting condition. Once the conflict was detected, the answer no is given immediately. Therefore, execution time of optimized query is zero. Experimental result for the first type of query (that is, query with redundant condition) are shown in Figure 4.

Table 1 The result of execution time for queries with conflicting condition

Query	**Query 1**	**Query 2**	**Query 3**	**Query 4**	**Query 5**
Original query	109 ms	103 ms	104 ms	101 ms	104 ms
Optimize query	0 ms	0 ms	0 ms	0 ms	0 ms

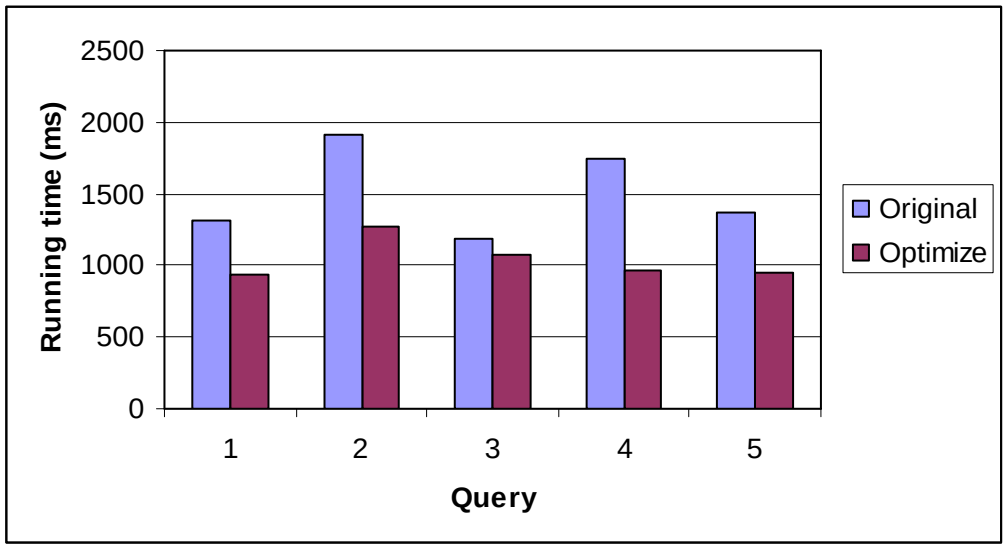


Figure 4 Response time of original queries compared to the optimized queries

This paper demonstrates that the efficiently optimized query depend on semantic constraints can be learned inductively under association data from a database and using data mining technique to fine data association. Experimental results show that there is significantly time different in query with conflict condition to the semantic constraint due to there is no execution query in database management system. Execution time in query with redundant condition depends on amount of data in the database.

References:

1. R. Argawal and R. Srikant, Fast algorithm for mining association rules. Proceedings of the 20th International Conference on Very Large Data Bases Conference, 1994.
2. R. Argawal, T. Imielinski and A. Swami, Mining association rules between set of items in large database. Proceedings of ACM SIGMOD International Conference on Management of Data. 1993.
3. Q. Cheng, J. Gryz, F. Koo, C. Lenung and L. Liu, Implementation of Two Semantic Query Optimization Techniques in DB2 Universal Database. Proceedings of the 25th International Conference on Very Large Data Bases VLDB '99, 1999

Keywords: Semantic query optimization, association rule induction.

Acknowledgements: This work was supported by grants from Nation Research Council of Thailand (NRCT). The authors are members of Data Engineering and Knowledge Discovery (DEKD) Research Unit, which is fully supported by Suranaree University of Technology.

ประวัติผู้เขียน

นายอภิชัย ฤทธิ์ธงชัยเลิศ เกิดเมื่อวันที่ 3 มิถุนายน พ.ศ. 2527 ที่ อ.สัตหีบ จ.ชลบุรี จบการศึกษาระดับประถมศึกษาที่โรงเรียนสัตหีบ จบการศึกษาระดับมัธยมศึกษาที่โรงเรียนสิงห์สมุทร และเข้าศึกษาในระดับปริญญาตรีในปีการศึกษา 2545 ที่สาขาวิชาวิศวกรรมคอมพิวเตอร์ สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี และสำเร็จการศึกษาเมื่อปี พ.ศ. 2548 ภายหลังจากจบปริญญาตรี ได้เข้าศึกษาต่อในระดับปริญญาโทสาขาวิชาวิศวกรรมคอมพิวเตอร์ สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี ในปีการศึกษา 2549

ในระหว่างการศึกษาได้รับความอนุเคราะห์อย่างดียิ่งจากคณาจารย์ในสาขาวิชา โดยได้รับความไว้วางใจให้เป็นผู้สอนปฏิบัติการรายวิชา Database System และ Computer Programming Laboratory