รายงานการวิจัย

การเพิ่มขยายการค้นหากฎความสัมพันธ์แบบหลายมิติ

Multidimension Incremental Association Rule Discovery

ผู้วิจัย รองศาสตราจารย์ ดร.วรพจน์ กรีสุระเดช

ได้รับทุนสนับสนุนงานวิจัยจากเงินรายได้ ประจำปีงบประมาณ 2553

คณะเทคโนโลยีสารสนเทศ

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

กิตติกรรมประกาศ

งานวิจัยเรื่องการเพิ่มขยายการค้นหากฎความสัมพันธ์แบบหลายมิติ ผู้วิจัยได้ทำการศึกษาและพัฒนาอัลกอริทึมเพื่อช่วยในการเพิ่มขยายการค้นหากฎความสัมพันธ์แบบหลายมิติเพื่อช่วยเพิ่มศักยภาพการหากฎความสัมพันธ์แบบเพิ่มขยายในกรณีที่มีการเพิ่มแถวข้อมูลใหม่และแอททริบิวต์ใหม่เข้าไปในฐานข้อมูลเดิมพร้อมๆ กัน งานวิจัยนี้ได้รับทุนจากคณะเทคโนโลยีสารสนเทศ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง จึงขอกราบขอบพระคุณเป็นอย่างสูงมา ณ ที่นี้

วรพจน์ กรีสุระเดช

บทคัดย่อ

ชื่อโครงการ (ภาษาไทย) การเพิ่มขยายการค้นหากฎความสัมพันธ์แบบหลายมิติ
(ภาษาอังกฤษ) Multidimension Incremental Association Rule Discovery

ได้รับทุนอุดหนุนการวิจัยจาก คณะเทคโนโลยีสารสนเทศ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ประจำปี 2553 จำนวนเงิน 100,000 บาท
ระยะเวลาทำการวิจัย 2 ปี ตั้งแต่ ตุลาคม พ.ศ. 2552 ถึง กันยายน พ.ศ. 2554

ผู้ดำเนินการวิจัย รองศาสตราจารย์ ดร.วรพจน์ กรีสุระเดช คณะเทคโนโลยีสารสนเทศ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง เบอร์โทรศัพท์ 02-723-4957

บทคัดย่อ (ภาษาไทย)

งานวิจัยนี้นำเสนอ อัลกอริทึมการค้นหากฎความสัมพันธ์แบบเพิ่มขยายสำหรับการเพิ่มข้อมูลที่มีแอททริบิวต์ใหม่ โดยอัลกอริทึมการค้นหากฎความสัมพันธ์แบบเพิ่มขยายสำหรับการเพิ่มข้อมูลที่มีแอททริบิวต์ใหม่ ออกแบบมาเพื่อใช้สำหรับการค้นหากฎความสัมพันธ์เมื่อมีการเพิ่มขึ้นของข้อมูลทั้งส่วนของเพิ่มแถวข้อมูลใหม่ และแอททริบิวต์ใหม่ในฐานข้อมูลเดิมพร้อมกัน

หลักการทำงานของอัลกอริทึม เริ่มจากการหา Large itemset ของแอททริบิวต์ใหม่ที่ถูกเพิ่มเข้ามา จากนั้นจึงนำ Large itemset ที่ได้ไปทำการหาความสัมพันธ์ที่เป็นไปได้ระหว่างข้อมูลเดิมกับแอททริบิวต์ใหม่ที่ถูกเพิ่ม โดยใช้ Large itemset ของข้อมูลเดิมและการประมาณค่าสนับสนุนมาช่วยเพื่อลดการค้นหาค่าสนับสนุนในการสร้าง Candidate itemset ของไอเท็มเซตที่เกิดขึ้นใหม่ หลังจากนั้นนำ Large itemset ที่ได้มาช่วยหา Large iteset ที่เกิดขึ้นจากการเพิ่มแถวข้อมูลใหม่ เพื่อให้ได้ Large itemset ทั้งหมดของข้อมูลที่ปรับปรุง

ในการวัดประสิทธิภาพทางด้านเวลาการทำงานและความถูกต้องของอัลกอริทึมนี้ ได้ทำการทดลองกับชุดข้อมูลสังเคราะห์ และทดสอบกับค่าสนับสนุนขั้นต่ำที่แตกต่างกัน จากผลการทดลองที่ได้พบว่าการทางานของอัลกอริทึมนี้ มีความถูกต้องและใช้เวลาการทำงานน้อยกว่าเมื่อเปรียบเทียบกับ อัลกอริทึมอะพิโอริและอัลกอริทึมการค้นหากฎความสัมพันธ์แบบมิติผสม

บทคัดย่อ (ภาษาอังกฤษ)

This research presents an incremental association rule discovery algorithm for appending data with new attributes. The proposed algorithm is designed to find association rules when new rows and new attributes are appended into an original dataset simultaneously.

Firstly, the algorithm finds the large itemsets of new attributes. Then, the large itemsets of new attributes are joined with the large itemsets of an original data in order to obtain the candidate itemsets of attribute updated data (AUD). To reduce time to find the supports of the candidate itemsets of attribute updated data (AUD), this work is also proposed to estimate the support of the candidate itemsets of attribute updated data (AUD). Then, the candidate itemsets of updated data (UD) are found by joining the large itemsets of attribute updated data (AUD) with the large itemsets of new rows. Finally, the large itemsets of updated data (UD) are obtained by scanning the original data.

To evaluate the correctness and performance of the proposed algorithm, several experiments are conducted with different synthesis data and different minimum support. The results show that the proposed algorithm is correctness and provides better performance than that of Apriori algorithm and Hybrid Apriori algorithm.

สารบัญ

หน้า

สารบัญ(ต่อ)

หน้า

สารบัญตาราง

สารบัญรูป

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

รูปที่ **หน้า**

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

การค้นหากฎความสัมพันธ์ (Association Rule Discovery) เป็นเทคนิคหนึ่งในการทำดาต้าไมนิ่งซึ่งนิยมใช้กับฐานข้อมูลขนาดใหญ่ โดยเป็นกระบวนการค้นหารูปแบบความสัมพันธ์ของข้อมูลระหว่างรายการต่างๆ ที่จัดเก็บในฐานข้อมูลทั้งหมด โดยความสัมพันธ์ของข้อมูลดังกล่าวจะอยู่ในรูปของกฎความสัมพันธ์ ซึ่งสามารถนำไปใช้สำหรับช่วยตัดสินใจและวางแผนด้านการบริหารเพื่อสร้างความได้เปรียบให้กับองค์กร เช่น การค้นหากฎความสัมพันธ์จากการซื้อสินค้าของลูกค้าเพื่อนำกฎความสัมพันธ์ที่ได้นั้นไปช่วยในการสร้างกลยุทธ์ส่งเสริมการขายสินค้า

โดยงานวิจัยเกี่ยวกับการค้นหากฎความสัมพันธ์ส่วนใหญ่ เป็นการค้นหากฎความสัมพันธ์กับข้อมูลทรานแซคชั่นการซื้อขาย แต่ในฐานข้อมูลยังมีแอททริบิวต์หรือมิติของข้อมูลอื่นๆ ที่สามารถนำมาใช้สำหรับการค้นหาความสัมพันธ์เพื่อค้นพบความรู้ใหม่ เช่น อายุ เพศ เงินเดือน ซึ่งเรียกว่าการค้นหากฎความสัมพันธ์แบบหลายมิติ (Multidimension Association Rules) สำหรับการค้นหากฎความสัมพันธ์แบบหลายมิติที่ลักษณะกฎความสัมพันธ์ไม่มีการซื้อกันของแอททริบิวต์ เรียกว่าการค้นหากฎความสัมพันธ์แบบระหว่างมิติ (Inter-Dimension Association Rules) และกฎความสัมพันธ์แบบหลายมิติที่ลักษณะของกฎความสัมพันธ์สามารถเกิดซ้ำของแอททริบิวต์ เรียกว่าการค้นหากฎความสัมพันธ์แบบมิติผสม (Hybrid-Dimension Association Rules) ซึ่งการค้นหากฎความสัมพันธ์ระหว่างมิติไม่สามารถแสดงให้เห็นถึงความสัมพันธ์ภายในแอททริบิวต์ได้เหมือนกับการค้นหากฎความสัมพันธ์แบบมิติผสม และการค้นหากฎความสัมพันธ์ส่วนใหญ่มักจะดำเนินการโดยตั้งสมมติฐานให้ฐานข้อมูลไม่มีการเปลี่ยนแปลงเกิดขึ้น (Static database) แต่ในความเป็นจริงข้อมูลในฐานข้อมูลมีการเปลี่ยนแปลงอยู่ตลอดเวลา (Dynamic database) ซึ่งอาจมีผลทำให้ Large itemsets และกฎความสัมพันธ์ที่มีอยู่เดิมเกิดการเปลี่ยนแปลงตามไปด้วย เมื่อมีการเปลี่ยนแปลงของฐานข้อมูล ลักษณะการค้นหากฎความสัมพันธ์ใหม่เมื่อมีการเปลี่ยนแปลงข้อมูลส่วนใหญ่เกิดได้จากการค้นหาข้อมูลเดิมซ้ำในฐานข้อมูลเก่า (Original Database) และค้นหาในฐานข้อมูลใหม่ (Increment Database) เพื่อจะปรับปรุงค่า Large itemsets ใหม่ทั้งหมด ทำให้ใช้เวลามากในการค้นหา เพราะไม่มีการนำค่า Large itemsets ที่ได้จากการค้นหาในฐานข้อมูลเดิมมาใช้ให้เกิดประโยชน์เพื่อลดการค้นหา Large itemsets ในฐานข้อมูลเดิม ซึ่งในอัลกอริทึม HDFUP ผู้วิจัยได้นำเสนอการค้นหากฎความสัมพันธ์เมื่อมีการเพิ่มของข้อมูลทรานแซคชั่นข้อมูลหลายมิติ

แต่เมื่อพิจารณากระบวนการของการทำไมนิ่งการค้นหากฎความสัมพันธ์เพื่อให้ได้กฎความสัมพันธ์ แล้วนำกฎความสัมพันธ์ที่ได้มาทำการกลั่นกรองกับสมมติฐานเพื่อทำการวิเคราะห์ข้อมูล ในบางครั้งผลลัพธ์ที่ได้จากการวิเคราะห์ความสัมพันธ์อาจไม่เพียงพอต่อความต้องการจึงอาจจะต้องมีการเพิ่มข้อมูลใหม่เข้าไปในฐานข้อมูลเดิมเพื่อทำให้ได้ความสัมพันธ์ที่เพียงพอต่อการวิเคราะห์ข้อมูล ซึ่งโดยความเป็นจริงแล้วการเพิ่มข้อมูลบางครั้งไม่จำเป็นต้องทำการเพิ่มเฉพาะข้อมูลที่เป็นข้อมูลทรานแซคชั่น เพื่อให้ได้กฎความสัมพันธ์ที่เพียงพอสำหรับการวิเคราะห์ข้อมูล แต่ยังสามารถเพิ่มข้อมูลแอททริบิวต์หรือมิติที่เกี่ยวข้อง เพื่อทำการค้นหาความสัมพันธ์ใหม่ ทั้งนี้อัลกอริทึม HDFUP ไม่สามารถแก้ไขปัญหาในลักษณะของการค้นหาความสัมพันธ์เมื่อมีการเพิ่มขึ้นของทั้งข้อมูลทรานแซคชั่นและข้อมูลแอททริบิวต์ใหม่ในข้อมูลเดิมได้ การที่จะให้ได้มาถึงความสัมพันธ์ที่เพียงพอต่อการวิเคราะห์ข้อมูล จึงต้องทำการค้นหากฎความสัมพันธ์ข้อมูลที่ปรับปรุงนั้นใหม่ทั้งหมดโดยไม่สามารถนำเอา Large itemsets เดิมที่เป็นความรู้จากข้อมูลในการค้นหากฎความสัมพันธ์เดิมมาใช้ประโยชน์ เพื่อลดการค้นหากฎความสัมพันธ์ เพื่อให้ได้ถึงความสัมพันธ์ที่เพียงพอสำหรับการวิเคราะห์ข้อมูลได้

เพื่อเป็นการเพิ่มประสิทธิภาพและพัฒนาการค้นหากฎความสัมพันธ์ให้มีความหลากหลาย งานวิจัยนี้ได้เสนอแนวคิดการค้นหากฎความสัมพันธ์แบบมิติผสมและการค้นหากฎความสัมพันธ์สำหรับการเพิ่มขึ้นของข้อมูลในฐานข้อมูลทรานแซคชั่นแบบหลายมิติ โดยนำเสนออัลกอริทึมการค้นหากฎความสัมพันธ์แบบเพิ่มขยายสำหรับการเพิ่มข้อมูลที่มีแอททริบิวต์ใหม่ ซึ่งการเพิ่มขึ้นของข้อมูลในฐานข้อมูลของงานวิจัยนี้เป็นการเพิ่มขึ้นของทรานแซคชั่นและการเพิ่มขึ้นของแอททริบิวต์ใหม่ในฐานข้อมูลเดิมพร้อมกัน เพื่อเป็นการเพิ่มและขยายประสิทธิภาพของการค้นหากฎความสัมพันธ์ของข้อมูลที่หลากหลายมากยิ่งขึ้น และให้ได้ทราบถึงองค์ความรู้บางอย่างที่ไม่เคยปรากฏมาก่อน

1.2 วัตถุประสงค์ของงานวิจัย

งานวิจัยฉบับนี้ มีวัตถุประสงค์เพื่อพัฒนาอัลกอริทึมการค้นหากฎความสัมพันธ์ให้มีประสิทธิภาพสำหรับกรณีที่มีการเพิ่มข้อมูลใหม่เข้าไปในฐานข้อมูล โดยลักษณะการเพิ่มข้อมูลนั้นเป็นการเพิ่มทั้งข้อมูลทรานแซคชั่นและข้อมูลแอททริบิวต์รองใหม่พร้อมกัน เพื่อให้มีความสามารถค้นหากฎความสัมพันธ์ของข้อมูลด้วยวิธีการค้นหากฎความสัมพันธ์หลายมิติแบบมิติผสมได้

1.3 สมมติฐานงานวิจัย

การพัฒนาอัลกอริทึมเพื่อสำหรับการเพิ่มขยายการค้นหากฎความสัมพันธ์แบบหลายมิติของรายงานวิจัยนี้ เพื่อทดสอบสมมติฐานที่ว่า Large itemsets ที่ได้จากอัลกอริทึมที่นำเสนอมีความถูกต้องครบถ้วน และมีประสิทธิภาพด้านเวลาในการทำงาน ดีกว่าเมื่อเปรียบเทียบกับอัลกอริทึมอะพริโอริ (Apriori) และอัลกอริทึมการค้นหากฎความสัมพันธ์หลายมิติแบบมิติผสม

1.4 ทฤษฎีหรือแนวคิดที่ใช้ในงานวิจัย

งานวิจัยนี้ได้นำทฤษฎีและเทคนิคต่างๆ ที่เกี่ยวข้องดังนี้

1. การค้นหากฎความสัมพันธ์ เป็นทฤษฎีและแนวคิดเกี่ยวกับการค้นหากฎความสัมพันธ์จากฐานข้อมูลขนาดใหญ่ โดยข้อมูลจะถูกนำมาสร้างกฎความสัมพันธ์จะต้องผ่านเกณฑ์ค่าชี้วัด 2 ค่า ได้แก่ ค่า minimum support ซึ่งใช้วัดค่าความถี่ของไอเท็มที่สามารถเป็น Large itemsets และค่า minimum confidence ซึ่งใช้สำหรับวัดค่าของกฎที่น่าสนใจ

2. การค้นหากฎความสัมพันธ์แบบมิติผสมเป็นทฤษฎีและแนวคิดเกี่ยวกับการค้นหากฎความสัมพันธ์แบบหลายมิติโดยกฎความสัมพันธ์ที่ได้จะประกอบด้วยข้อมูลหลายมิติ

3. การเพิ่มขยายการค้นหากฎความสัมพันธ์ เป็นทฤษฎีและแนวคิดเกี่ยวกับการค้นหากฎความสัมพันธ์เมื่อมีการเพิ่มข้อมูลเข้าสู่ฐานข้อมูล ซึ่งมีผลต่อความสัมพันธ์ และ Large itemsets เดิมที่ได้ทำการไมนิ่งในฐานข้อมูลเดิม

4. การเพิ่มขยายการค้นหากฎความสัมพันธ์แบบมิติผสม เป็นทฤษฎีและแนวคิดเกี่ยวกับการค้นหากฎความสัมพันธ์เมื่อมีการเพิ่มข้อมูลเข้าสู่ฐานข้อมูลแบบหลายมิติซึ่งมีผลต่อความสัมพันธ์และ Large itemsets เดิมที่ได้ทำการไมนิ่งในฐานข้อมูลเดิม

1.5 ขอบเขตการวิจัย

ผู้วิจัยได้กำหนดขอบเขตการวิจัย ดังนี้

1. งานวิจัยนี้จะทำการพัฒนาอัลกอริทึมการเพิ่มขยายการค้นหากฎความสัมพันธ์บนพื้นฐานของอัลกอริทึมอะพริโอริ

2. การเพิ่มข้อมูลแอททริบิวต์ของงานวิจัยนี้ เป็นการเพิ่มเฉพาะข้อมูลแอททริบิวต์รองเท่านั้น

3. การพัฒนาอัลกอริทึมของงานวิจัยนี้เป็นการพัฒนาขั้นตอนการทำงานในส่วนของการค้นหากฎความสัมพันธ์เฉพาะในส่วนของการค้นหา Large itemsets เท่านั้น

4. ข้อมูลที่ใช้ในการทดสอบประสิทธิภาพของอัลกอริทึม เป็นข้อมูลที่ได้จากการสังเคราะห์ด้วยวิธีการที่นิยมใช้ในงานวิจัยด้านการค้นหากฎความสัมพันธ์

1.6 ขั้นตอนของการศึกษา

ขั้นตอนในการศึกษางานวิจัย มีดังนี้

1. ศึกษาทฤษฎี แนวคิด และงานวิจัย จากเอกสาร บทความต่างๆ ในส่วนที่เกี่ยวข้องกับงานวิจัย
2. กำหนดหัวข้อ เป้าหมาย วัตถุประสงค์ และขอบเขตของงานวิจัย
3. วิเคราะห์และออกแบบอัลกอริทึม
4. เตรียมข้อมูลเพื่อใช้ทดลอง
5. พัฒนาโปรแกรมของงานวิจัย ทดสอบ และแก้ไขข้อผิดพลาด
6. รวบรวมผลการทดลองจากการทำงานของอัลกอริทึม
7. วิเคราะห์และสรุปผลการทดลอง
8. ดำเนินการจัดทำเอกสารงานวิจัย

1.7 นิยามศัพท์

ในงานวิจัยนี้มีการใช้ศัพท์เฉพาะในการศึกษาอัลกอริทึม เพื่อให้เข้าใจตรงกันผู้วิจัยได้นิยามศัพท์ที่สำคัญและนิยมใช้ในงานวิจัย ดังนี้

Association Rule หรือ กฎความสัมพันธ์ เป็นกระบวนการการหารูปแบบของข้อมูลที่น่าสนใจในฐานข้อมูลแล้วแสดงออกมาในรูปกฎความสัมพันธ์

Original database หมายถึง ฐานข้อมูลดั้งเดิมที่ยังไม่มีการเพิ่มข้อมูลใหม่เข้าไปในฐานข้อมูล

Increment database หมายถึง ฐานข้อมูลใหม่ที่มีการเพิ่มเข้าไปใน Original Database

Update database หมายถึง ฐานข้อมูลที่ได้รับการปรับปรุงแล้ว นั่นคือมีการเพิ่มข้อมูลใหม่เข้าไปจากฐานข้อมูลเรียบร้อยแล้ว

Frequent itemset หรือ **Large itemsets** หมายถึง เซตของไอเท็มที่ผ่านการพิจารณาแล้วจัดว่าเป็นไอเท็มที่มีความน่าสนใจ เพื่อที่จะนำไปสร้างกฎความสัมพันธ์ต่อไป

Infrequent itemset หรือ **Small itemset** หมายถึง เซตของไอเท็มที่ผ่านการพิจารณาแล้วไม่จัดว่าเป็นไอเท็มที่มีความน่าสนใจ

Candidate itemset หมายถึง เซตของไอเท็มที่ใช้พิจารณาว่าเป็น frequent itemset โดยทำการเปรียบเทียบกับค่าทดสอบ หาก candidate itemset ใดที่มีค่ามากกว่าหรือเท่ากับค่าที่ใช้ทดสอบ candidate itemset นั้นจะเรียกว่า frequent itemset ในทางกลับกันหากเปรียบเทียบแล้วพบว่ามีค่าน้อยกว่าค่าทดสอบ candidate itemset นั้นจะเรียกว่า infrequent itemset

Minimum support หรือ **min_sup** หมายถึง ค่าที่ใช้ทดสอบความสามารถในการเป็น frequent itemset

Minimum confidence หรือ **min_conf** หมายถึง ค่าที่ใช้ทดสอบว่า กฎความสัมพันธ์ใดเป็นกฎที่มีความน่าสนใจหรือเป็นกฎที่เข้มแข็ง (strong rule)

k-itemset หมายถึง เซตของไอเท็มที่ประกอบด้วยไอเท็มจำนวน k ตัว โดย k มากกว่าหรือเท่ากับ 1

บทที่ 2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

ในบทนี้จะกล่าวถึงทฤษฎีพื้นฐานต่างๆ และงานวิจัยที่เกี่ยวข้องในการทำงานวิจัย โดยในเนื้อหาบทนี้จะกล่าวถึงการค้นหากฎความสัมพันธ์ (Association rule discovery) การค้นหากฎความสัมพันธ์แบบหลายมิติ (Multi-Dimension Association rule discovery) และการเพิ่มขยายการค้นหากฎความสัมพันธ์ (Incremental Association Rule Discovery) โดยมีรายละเอียดดังนี้

2.1 การค้นหากฎความสัมพันธ์ของข้อมูล (Association Rule Discovery)

การค้นหากฎความสัมพันธ์ของข้อมูลเป็นวิธีการในการทำดาต้าไมนิ่งอย่างหนึ่ง ซึ่งนิยมใช้กันอย่างแพร่หลายในหลายสาขา ทั้งในการวิจัยเชิงการศึกษาและประยุกต์ใช้กับองค์กรธุรกิจเพื่อค้นหารูปแบบความสัมพันธ์ของข้อมูลในฐานข้อมูล ความสัมพันธ์ที่ได้มาสามารถนำไปใช้ในกระบวนการตัดสินใจ ซึ่งการประยุกต์การใช้งานส่วนใหญ่จะเป็นการวิเคราะห์การขายสินค้า (Market basket analysis) เพื่อสรุปเป็นความสัมพันธ์ของสินค้า เป็นการค้นหากฎความสัมพันธ์ของข้อมูลการซื้อสินค้าของลูกค้าโดยศึกษาวิเคราะห์พฤติกรรมของลูกค้าว่าเมื่อลูกค้าทำการซื้อสินค้าชนิดหนึ่งแล้วจะซื้อสินค้าใดควบคู่กันไปด้วย และนำกฎความสัมพันธ์ที่ค้นพบนั้นมาใช้ในการปรับปรุงกลยุทธ์การขายสินค้า หรือใช้ประกอบการพิจารณาในการจัดวางสินค้า ทำให้เกิดความสะดวกในการเลือกซื้อสินค้า และเป็นการเพิ่มยอดจำหน่ายให้กับร้านค้า

การไมนิ่งกฎความสัมพันธ์ (Association rule mining) ได้ถูกนำเสนอครั้งแรก โดย R. Agrawal, T. Imielinski และ A. Swami ในปีค.ศ. 1993 [1] เพื่อใช้ในการค้นหาความสัมพันธ์ที่น่าสนใจระหว่างไอเท็มในชุดข้อมูลของทรานแซคชั่น (transaction dataset) ในฐานข้อมูลขนาดใหญ่ การไมนิ่งกฎความสัมพันธ์สามารถนำเสนอโดยรูปแบบทางคณิตศาสตร์ได้ดังนี้

กำหนดให้

I เป็นเซตของไอเท็ม โดย $I = \{i_1, i_2, \ldots, i_n\}$ ที่แตกต่างกัน n ตัว

T เป็นทรานแซคชั่นซึ่งแต่ละทรานเซคชั่นเป็นเซตของไอเท็ม โดยที่ $T \subseteq I$ และT แต่ละตัวจะสัมพันธ์กับตัวระบุ ทรานแซคชั่นที่เรียกว่า TID (Transaction Identifier) ซึ่งจะมีค่าเป็นหนึ่งเดียว (unique) ในฐานข้อมูล

D เป็นเซตของ ทรานแซคชั่นในฐานข้อมูล $\{T_1, T_2, \ldots, T_m\}$

X เป็น ไอเท็มในทรานแซคชั่น นั้นคือ $X \subseteq T$

กฎความสัมพันธ์อยู่ในรูปของกฎ IF...THEN rule แสดงโดยรูปแบบ $X \rightarrow Y$ คือ "ถ้า X แล้ว Y" หรือ "IF X THEN Y" โดยค่า $X \subset I$ กับ $Y \subset I$ และ $X \cap Y = \phi$ จากรูปแบบกฎความสัมพันธ์ของข้อมูลประกอบด้วย 2 ส่วนคือส่วนที่เป็นด้านซ้ายของกฎเป็นสิ่งที่เกิดขึ้นก่อน (Antecedent หรือ Rule body หรือ Leaf-hand side) และส่วนที่เป็นด้านขวาของกฎคือสิ่งที่เป็นผลตามมา (Consequent หรือ Rule head หรือ Right-hand side)

โดยไอเท็มเซตที่สามารถนำไปสร้างกฎความสัมพันธ์ของข้อมูลจะต้องผ่านค่าที่นำมาพิจารณาอยู่ 2 ค่าที่สำคัญ คือ ค่าสนับสนุน (s) และ ค่าความเชื่อมั่น (c) กฎความสัมพันธ์ $X \rightarrow Y$ จะถูกจัดให้มีในเซตของทรานแซคชั่น D ด้วยค่า สนับสนุน (s) แสดงดังสมการที่ 2.1ซึ่งเป็นจำนวนเปอร์เซ็นต์ของข้อมูลทรานแซคชั่นใน D ที่มีทั้ง item X และ Y นั้นคือ $(X \cup Y)$ กับจำนวนทรานแซคชันทั้งหมดที่อยู่ใน D โดยจะเป็นค่าความน่าจะเป็นที่จะเกิด X และ Y พร้อมกัน $(P(X \cup Y))$

$$\text{support}(X \rightarrow Y) = P(X \cup Y) = \frac{\text{support\_count}(X \cup Y)}{\text{amount\_transaction}} \qquad (2.1)$$

ส่วนกฎความสัมพันธ์ $X \rightarrow Y$ ใดที่จะเป็นกฎที่น่าสนใจ จะต้องมีค่าความเชื่อมั่น (c) แสดงดังสมการที่ 2.2 ในเซตของข้อมูลทรานแซคชั่นใน D เมื่อ c เป็นเปอร์เซ็นต์ของข้อมูลทรานแซกชั่นใน D ที่มี X แล้วจะต้องมี Y ด้วยซึ่งเป็นเรื่องความน่าจะเป็นแบบมีเงื่อนไข P(X|Y)

$$\text{confidence}(X \rightarrow Y) = P(Y|X) = \frac{\text{support\_count}(X \cup Y)}{\text{support\_count}(X)} \qquad (2.2)$$

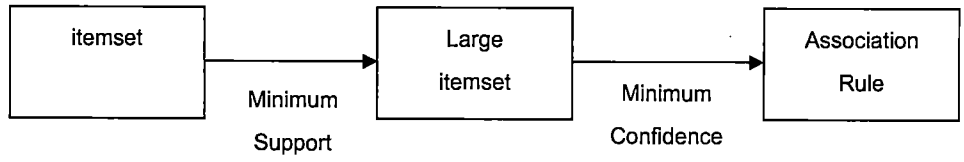
การค้นหากฎความสัมพันธ์จะประกอบด้วยขั้นตอนหลัก 2 ขั้นตอนที่สำคัญ ดังรูปที่ 2.1 ได้แก่

1. การหา Large itemsets หรือ frequent itemsets ทั้งหมด

ไอเท็มเซต X ใดๆ สามารถเป็น Large itemsets ได้จะต้องมีค่าสนับสนุนมากกว่าหรือเท่ากับค่าสนับสนุนขั้นต่ำที่กำหนดไว้ นั้นคือ $\{X \mid X.\text{support} \geq s_{min}\}$ ในขั้นตอนนี้จะเป็นการหา Large k-itemsets โดยที่ k เป็นจำนวนไอเท็มที่อยู่ใน Large itmeset หรือแทนด้วยสัญลักษณ์ $\{L_1, L_2, ..., L_k\}$

2. การสร้างกฎความสัมพันธ์ที่แข็งแกร่งจาก Large itemsets

โดยการสร้างกฎความสัมพันธ์จะสร้างจาก Large itemsets จากขั้นตอนที่ 1 โดยกฎจะถูกต้องหรือน่าสนใจก็ต่อเมื่อมีค่ามากกว่าหรือเท่ากับค่าสนับสนุนและความเชื่อมั่นขั้นต่ำจึงเป็นกฎที่น่าเชื่อถือ



รูปที่ 2.1 แสดงขั้นตอนการค้นหากฎความสัมพันธ์

Transaction ID (TID)	Item
1	Computer, Software
2	Computer, Printer
3	CD, Scanner, Printer
4	Computer, Software, CD
5	USB, CD, Computer

รูปที่ 2.2 ตัวอย่างข้อมูลการซื้อสินค้าของลูกค้า

จากตัวอย่างดังรูปที่ 2.2 แสดงข้อมูลการซื้อสินค้าของลูกค้า เซตของไอเท็ม I = {Computer, CD, Printer, Software, Scanner, USB} การที่ลูกค้าถ้าซื้อ Computer แล้วจะซื้อ Software ด้วย แสดงในรูปแบบของกฎความสัมพันธ์ Computer → Software การที่ซื้อ Computer แล้วจะซื้อ Software ด้วยอยู่ใน ทรานแซคชั่นที่ 1 และ 4 ดังนั้นค่าสนับสนุนสำหรับเซตไอเท็ม {Computer, Software} คือ 2/5*100 = 40% หมายความว่า จากรายการขายทั้งหมดที่นำมาวิเคราะห์ ลูกค้าที่ซื้อ Computer และ Software ไปด้วยกัน คิดเป็นร้อยละ 40 ของทั้งหมด ในที่นี้คือจำนวน 2 ทรานแซคชั่น จากทั้งหมด 5 ทรานเซคชั่น และมีค่าความเชื่อมั่นสำหรับกฎความสัมพันธ์ คือ 2/4*100 = 50% หมายความว่า ในจำนวนผู้ซื้อที่ซื้อ Computer ทั้งหมดพบว่ามีจำนวนร้อยละ 50 ที่ซื้อ Software ไปด้วย กล่าวคือมีจำนวน 4 ทรานแซคชั่นที่ซื้อคอมพิวเตอร์โดยใน 4 รายการนั้นมีจำนวน 2 ทรานแซคชั่นที่ซื้อ Software ด้วย

ดังนั้นกฎความสัมพันธ์ที่มีความน่าสนใจ คือ กฎที่มีค่าสนับสนุนและค่าความเชื่อมั่นสูงกว่าหรือเท่ากับค่าสนับสนุนขั้นต่ำ(Minimum support) คือ ค่าสนับสนุนน้อยสุดที่ทำให้ความสัมพันธ์ที่ได้นั้นยังมีความน่าสนใจ และความมั่นใจขั้นต่ำ (Minimum confidence) คือ ค่าความเชื่อมั่นน้อยสุดที่ทำให้กฎความสัมพันธ์ที่ได้นั้นยังมีความน่าสนใจที่กำหนดไว้ โดยค่าสนับสนุนและค่าความเชื่อมั่นจะอยู่ในช่วง 0% ถึง 100% และจะมีค่าที่ได้อยู่ในช่วง 0 ถึง 1.0

ประเภทของกฎความสัมพันธ์

การวิเคราะห์พฤติกรรมการซื้อสินค้าของผู้บริโภคเป็นอีกรูปแบบหนึ่งของการค้นหากฎความสัมพันธ์ ในความเป็นจริงกฎความสัมพันธ์มีอยู่หลายประเภท โดยกฎความสัมพันธ์สามารถจำแนกได้หลายแนวทางขึ้นอยู่กับเกณฑ์ที่ใช้ในการจำแนก [6]

1. กฎความสัมพันธ์ที่เป็นข้อเท็จจริง (Boolean association rule)

กฎความสัมพันธ์ที่เกี่ยวกับความสัมพันธ์ระหว่างการมีอยู่หรือไม่มีอยู่ของไอเท็ม เช่น ขนมปัง → นม เป็นกฎความสัมพันธ์ที่เป็นข้อเท็จจริงที่ได้มาจากการวิเคราะห์พฤติกรรมการซื้อสินค้าของผู้บริโภค

2. กฎความสัมพันธ์เกี่ยวกับปริมาณ (Quantitative association rule)

เป็นกฎความสัมพันธ์ที่อธิบายความสัมพันธ์ระหว่างปริมาณของไอเท็มหรือแอททริบิวต์ โดยในกฎจะมีค่าปริมาณของไอเท็มหรือแอททริบิวต์ โดยจะถูกพิจารณาเป็นช่วงของข้อมูล เช่น อายุ (X, "17 - 22") ∧ สีผม(X, "ดำ") → รายได้(X, "1000 - 2000") แอททริบิวต์ที่เป็นการบอกปริมาณคือ อายุ และรายได้

3. กฎความสัมพันธ์หนึ่งมิติ (Single-dimensional association rule)

เป็นกฎความสัมพันธ์ที่มีไอเท็มหรือแอททริบิวต์ในกฎความสัมพันธ์อ้างถึงข้อมูลเพียงหนึ่งมิติ เช่น ซื้อ (X, "ขนมปัง") → ซื้อ (X, "นม") จะเห็นได้ว่าอ้างถึงข้อมูลเพียงหนึ่งมิติเท่านั้น คือ มิติ "ซื้อ"

4. กฎความสัมพันธ์หลายมิติ (Multi-dimensional association rule)

เป็นกฎความสัมพันธ์ที่มีไอเท็มหรือแอททริบิวต์ภายในกฎความสัมพันธ์ มีการอ้างอิงมิติของข้อมูลมากกว่าหนึ่งมิติขึ้นไป เช่น อาชีพ (X, "นักเรียน") ∧ ซื้อ (X, "คอมพิวเตอร์") → ซื้อ (X, "เครื่องพิมพ์") สามารถพิจารณาได้ว่าเป็นกฎความสัมพันธ์หลายมิติเนื่องจากมีการอ้างถึงข้อมูล 2 มิติ คือ มิติอายุ และ มิติซื้อ

5. กฎความสัมพันธ์หลายระดับ (Multi-level association rule)

เนื่องจากการค้นหากฎความสัมพันธ์บางวิธีสามารถค้นหากฎความสัมพันธ์ที่มีระดับของนามธรรมที่แตกต่างกัน นั้นคือชุดของกฎความสัมพันธ์ที่มีกฎความสัมพันธ์อื่นตามมาด้วย เช่น อายุ (X, "30 .. 39") → ซื้อ (X, "เบียร์") และ อายุ (X, "30 .. 39") → ซื้อ (X, "เครื่องดื่มแอลกอฮอล์") กฎความสัมพันธ์ทั้งสอง ไอเท็มที่ถูกอ้างอิงอยู่ในระดับที่แตกต่างกัน คือ เครื่องดื่มแอลกอฮอล์ อยู่ในระดับที่สูงกว่า เบียร์

6. กฎความสัมพันธ์ระดับเดียว (Single-level association rule)

จะคล้ายกับกฎความสัมพันธ์หลายระดับแตกต่างกันเพียงกฎความสัมพันธ์ระดับเดียวจะมีการอ้างอิงข้อมูลที่อยู่ในระดับเดียวกัน เช่น อายุ (X, "30 .. 39") □ ซื้อ(X, "เบียร์") และ อายุ (X, "30 .. 39") □ ซื้อ (X, "เหล้า")

ในงานวิจัยทางด้านการค้นหากฎความสัมพันธ์อัลกอริทึมที่ได้รับความนิยมสำหรับใช้ในการค้นหากฎความสัมพันธ์คือ อัลกอริทึมอะพริโอริ ซึ่งมีลักษณะดังนี้

2.1.1 อัลกอริทึมอะพริโอริ (Apriori Algorithm) [2]

อัลกอริทึมอะพริโอริได้ถูกเสนอโดย Agrawal and Srikant เป็นอัลกอริทึมหนึ่งที่ได้รับการยอมรับว่าเป็นอัลกอริทึมสำหรับการค้นหากฎความสัมพันธ์ที่มีประสิทธิภาพและนิยมนำมาประยุกต์ใช้ในงานวิจัยทางด้านการค้นหากฎความสัมพันธ์ หลักการทำงานของอัลกอริทึมเป็นการทำงานแบบลำดับชั้นหรือที่เรียกว่า Level-wise-step โดยอัลกอริทึมจะมีการคำนวณหา Large itemsets ซึ่งการทำงานของอัลกอริทึมจะค้นหาแต่ละทรานแซคชั่นในฐานข้อมูล ซึ่งจะทำการวนรอบค้นหาซ้ำหลายครั้ง (Iteration) โดย large k- itemsets ที่ค้นหาได้ในแต่ละรอบจาก large k-1 itemsets ในรอบก่อนหน้า กล่าวคือ L_1 จะถูกใช้ในการค้นหา L_2 และ L_2จะถูกใช้ในการค้นหา L_3 เช่นนี้ไปเรื่อยๆ จนกว่าไม่สามารถหา Large itemsets ได้อีก โดยอัลกอริทึมอะพริโอริแสดงดังรูป 2.3 มีขั้นตอนการทำงานที่สำคัญ 2 ขั้นตอน

1. ขั้นตอนการ Join

เป็นขั้นตอนเพื่อหา Large itemsets โดยการนำ L_{k-1} ที่ ($k \geq 2$) มาทำการ join เพื่อสร้าง Candidate itemset โดยอัลกอริทึมอะพริโอริจะต้องมีการเรียงลำดับข้อมูลไอเท็มที่อยู่ในทรานแซคชั่น (Lexicographic order) กรณีที่เป็นการสร้าง C_1 (candidate 1-itemset) จะนำแต่ละไอเท็มที่มีอยู่ในแต่ละทรานแซคชั่นในฐานข้อมูลมาสร้างเป็น C_1 โดยไม่ต้องทำการ join ซึ่งแต่ละ candidate itemset แต่ละตัวต้องมีค่าสนับสนุนมากกว่าศูนย์ และในกรณีที่ C_k ที่ ($k \geq 2$) สามารถหาได้จากการนำ L_{k-1} มา join กัน เช่น C_3 ได้จากการ join ระหว่าง L_2*L_2 เพื่อให้ได้ C_3 สำหรับใช้คำนวณหา L_3 เป็นลำดับถัดไป และทำแบบนี้ไปเรื่อยๆจน $C_k = \phi$ กระบวนการการทำการ join แสดงดังรูปที่ 2.4

```
1)  L1 = {large 1-itemsets};
2)  for ( k = 2; Lk-1 ≠ ∅; k++ ) do begin
3)     Ck = apriori-gen(Lk-1); // New candidates
4)     forall transactions t ∈ D do begin
5)        Ct = subset(Ck, t); // Candidates contained in t
6)        forall candidates c ∈ Ct do
7)           c.count++;
8)     end
9)     Lk = {c ∈ Ck | c.count ≥ minsup}
10) end
11) Answer = ⋃k Lk;
```

รูปที่ 2.3 แสดงการหา Large itemsets ของ Apriori Algorithm

2. ขั้นตอนการ Prune (Prune step)

2.1 เป็นขั้นตอนในการคัดสมาชิกเพื่อตัดไอเท็มเซตด้วยคุณสมบัติของอะพริโอริออกจาก C_k โดยการตัดไอเท็มเซตนั้นๆ ก็ต่อเมื่อซับเซต ของ k-1 itemset ใดๆ ใน C_k ที่ไม่ได้เป็นสมาชิกของ L_{k-1} แล้วไอเท็มเซตนั้นจะไม่สามารถเป็น L_k ได้ ซึ่งสามารถตัดไอเท็มเซตนั้นออกจาก C_k ได้ เช่น C_3 คือ {abc} ซับเซตของ {abc} จะต้องมีไอเท็มเซตอยู่ใน L_2 คือ {ab}, {ac} และ {bc} ทุกตัว กระบวนการการทำงานแสดงดังรูปที่ 2.5

The **apriori-gen** function takes as argument L_{k-1}, the set of all large $(k-1)$ itemsets. It returns a superset of the set of all large k-itemsets. The function works as follows. [1] First, in the *join* step, we join L_{k-1} with L_{k-1}:

```
insert into Ck
select p.item1, p.item2, ..., p.itemk-1, q.itemk-1
from Lk-1 p, Lk-1 q
where p.item1 = q.item1, ..., p.itemk-2 = q.itemk-2, p.itemk-1 < q.itemk-1;
```

รูปที่ 2.4 แสดง การ join ของprocedure apriori_gen

2.2 การคัดสมาชิกออกด้วยค่าสนับสนุนน้อยที่สุด หลังจากคัดสมาชิกออกด้วยคุณสมบัติของอะพริโอริจากข้อ 2.1 โดยคัดสมาชิกใน C_k ที่มีค่าความถี่น้อยกว่าค่าสนับสนุนน้อยที่สุดออก เพื่อสร้าง Large itemsets

การหา Large itemsets จะทำวนซ้ำตามขั้นตอนที่ 1 และ ขั้นตอนที่ 2 ไปเรื่อยๆ จนกว่าจะไม่สามารถหา L_k ได้อีก จึงหยุดการหา Large itemsets

Next, in the *prune* step, we delete all itemsets $c \in C_k$ such that some $(k-1)$-subset of c is not in L_{k-1}:
forall itemsets $c \in C_k$ **do**
  forall $(k-1)$-subsets s of c **do**
    if $(s \notin L_{k-1})$ **then**
      delete c from C_k;

รูปที่ 2.5 แสดง Prune step

เมื่อเสร็จสิ้นขั้นตอนของการหา Large itemsets ขั้นตอนถัดไปคือการสร้างกฎความสัมพันธ์ที่เป็นไปตามค่าสนับสนุนขั้นต่ำ และค่าความเชื่อมั่นขั้นต่ำที่กำหนด โดยค่าดังกล่าวจะมีช่วงระหว่าง 0-100% กฎความสัมพันธ์ใดที่เป็นไปตามค่าที่กำหนดจะถูกเรียกว่า กฎที่น่าสนใจ หรือกฎที่เข้มแข็ง (strong rule) ในทางกลับกันหากกฎความสัมพันธ์ใดที่ไม่เป็นไปตามค่าที่กำหนดจะจัดเป็นกฎที่ไม่น่าสนในหรือกฎที่อ่อนแอ (weak rule)

อัลกอริทึมอะพริโอรินั้นเป็นอัลกอริทึมของการไมนิ่งในรูปแบบกฎความสัมพันธ์แบบหนึ่งมิติ หมายถึง สามารถค้นหากฎความสัมพันธ์ที่อ้างถึงแอททริบิวต์เดียวหรืออยู่ภายในแอททริบิวต์เดียวเท่านั้น เช่น ความสัมพันธ์เฉพาะข้อมูลสินค้าในการซื้อของลูกค้า

ข้อด้อยของอัลกอริทึมอะพริโอริ

1. อัลกอริทึมอะพริโอริจำเป็นที่จะต้องเข้าไปอ่านข้อมูลหลายรอบเพื่อทำการนับค่าสนับสนุนของรูปแบบความสัมพันธ์ของไอเท็มต่างๆ ดังนั้นจะเห็นได้ว่าต้องเสียเวลามากในการทำการอ่านข้อมูลทรานแซคชั่น

2. ในกรณีเมื่อฐานข้อมูลเกิดการเปลี่ยนแปลง อัลกอริทึมอะพริโอริต้องทำการค้นหากฎความสัมพันธ์ใหม่ทั้งหมด จึงต้องทำการสแกนข้อมูลทั้งฐานข้อมูลใหม่ ทำให้ต้องสูญเสียเวลามากในการค้นหากฎความสัมพันธ์ใหม่ทั้งหมด

3. ในฐานข้อมูลจริงอาจมีการเก็บข้อมูลอื่นที่เกี่ยวข้องกับทรานแซคชั่น เช่น อายุ ที่อยู่ และเงินเดือน แต่อัลกอริทึมอะพริโอริเป็นอัลกอริทึมสำหรับการค้นหากฎความสัมพันธ์ของข้อมูลที่อยู่ในแอททริบิวต์เดียวกันเท่านั้น

2.1.2 อัลกอริทึมอะพริโอริสำหรับการหากฎความสัมพันธ์แบบมิติผสม [8]

การทำงานของอัลกอริทึมอะพริโอริดังที่ได้กล่าวมาแล้ว จะเห็นได้ว่าการค้นหากฎความสัมพันธ์ของข้อมูลใช้สำหรับการเก็บข้อมูลทรานแซคชั่นที่เป็นการเก็บข้อมูลแบบมิติเดียว ทำให้อัลกอริทึมอะพริโอริสามารถค้นหากฎความสัมพันธ์ของข้อมูลได้เพียงมิติเดียว ซึ่งในความเป็นจริงการจัดเก็บข้อมูลทรานแซคชั่นในฐานข้อมูล มีการเก็บรายละเอียดของข้อมูลหลายมิติ ดังนั้นการหากฎความสัมพันธ์แบบมิติผสมจึงมีการปรับเปลี่ยนบางขั้นตอนของอัลกอริทึมอะพริโอริ ให้สามารถนำไปใช้กับฐานข้อมูลที่มีการจัดเก็บทรานแซคชั่นข้อมูลแบบหลายมิติได้ การค้นหากฎความสัมพันธ์เมื่อแบ่งตามมิติที่ปรากฏในกฎความสัมพันธ์สามารถแบ่งได้เป็น 2 ประเภท คือ

1. กฎความสัมพันธ์มิติเดียว (Single Dimensional Association Rules)

กฎความสัมพันธ์มิติเดียว หรือกฎความสัมพันธ์มิติภายใน (Intra dimensional association rule) เป็นการค้นหากฎความสัมพันธ์ ที่แสดงความสัมพันธ์ของข้อมูลมิติเดียวหรือแอททริบิวต์เดียว ตัวอย่างแสดงดังกฎ

buys (X, “notebook computer”) → buys (X, “antivirus software”)

จากกฎความสัมพันธ์หมายถึง ถ้าลูกค้าซื้อเครื่องคอมพิวเตอร์โน้ตบุ๊กแล้ว จะซื้อซอฟท์แวร์ป้องกันไวรัสด้วย จะเห็นได้ว่ากฎความสัมพันธ์จะแสดงข้อมูลเพียงมิติเดียวคือ “มิติซื้อ”

2. กฎความสัมพันธ์หลายมิติ (Multi-dimensional Association Rules)

เป็นการค้นหากฎความสัมพันธ์ของข้อมูลหลายมิติ ที่ภายในกฎความสัมพันธ์แสดงมิติมากกว่ามิติเดียวหรือมากกว่าแอททริบิวต์เดียว โดยกฎความสัมพันธ์หลายมิติ สามารถแบ่งออกเป็น 2 ประเภทคือ

- กฎความสัมพันธ์หลายมิติแบบระหว่างมิติ (Inter Dimensional Association Rules)

เป็นการค้นหากฎความสัมพันธ์หลายมิติ โดยภายในกฎความสัมพันธ์ที่ได้จากการค้นหาจะไม่สามารถเกิดการซ้ำกัน หรือเกิดได้เพียงหนึ่งครั้งของแต่ละแอททริบิวต์หรือมิติ ตัวอย่างของกฎความสัมพันธ์หลายมิติแบบระหว่างมิติแสดงดังนี้

age(X, “20…29”) ∧ occupation (X, “student”) → buys (X, “laptop”)

จากกฎความสัมพันธ์ หมายถึง ถ้าลูกค้าที่มีอายุระหว่าง 20-29 ปี และมีอาชีพเป็นนักเรียนจะซื้อเครื่องคอมพิวเตอร์โน้ตบุ๊ก จะเห็นได้ว่ากฎความสัมพันธ์ดังกล่าวประกอบด้วย 3 มิติ คือ อายุ อาชีพ และการซื้อ ซึ่งแต่ละมิติไม่ซ้ำกันหรือแต่ละมิติเกิดขึ้นเพียงหนึ่งครั้ง

- กฎความสัมพันธ์หลายมิติแบบมิติผสม (Hybrid Dimensional Association Rules)

เป็นการค้นหากฎความสัมพันธ์แบบหลายมิติ โดยเป็นการหากฎความสัมพันธ์ทั้งการค้นหากฎความสัมพันธ์แบบมิติเดียวและการค้นหากฎความสัมพันธ์หลายมิติแบบระหว่างมิติ กฎความสัมพันธ์ที่ได้จากการค้นหาสามารถมีการเกิดซ้ำกันของแอททริบิวต์ ตัวอย่างของกฎความสัมพันธ์หลายมิติแบบมิติผสมแสดงดังตัวอย่าง

$$age(X,"20\ldots29") \wedge buys(X,"notebook\ computer") \rightarrow buys(X,"antivirus\ software")$$

จากกฎความสัมพันธ์ดังกล่าวมีความหมายคือ ถ้าลูกค้าที่มีอายุระหว่าง 20-29 ปี และซื้อเครื่องคอมพิวเตอร์โน้ตบุ๊ก จะซื้อซอฟท์แวร์แอนตี้ไวรัสไปด้วยกัน เห็นได้ว่ากฎความสัมพันธ์มี 2มิติ ที่ปรากฏ คือ มิติอายุ และมิติการซื้อ แสดงให้เห็นว่ามีการทำนายมิติการซื้อซึ่งเป็นแอททริบิวต์ที่เกิดซ้ำกัน

เงื่อนไขและข้อจำกัดของการค้นหากฎความสัมพันธ์หลายมิติแบบมิติผสม

ในกฎความสัมพันธ์หลายมิติแบบมิติผสม มิติหรือแอททริบิวต์ในฐานข้อมูลสามารถแบ่งออกได้เป็น 2 ประเภท ดังนี้

1. แอททริบิวต์หลัก (Main Attribute)

แอททริบิวต์หลัก เป็นแอททริบิวต์ที่ประกอบด้วยข้อมูลภายในทรานแซคชั่นของแอททริบิวต์นั้นซึ่งสามารถมีค่าหนึ่งค่าหรือมากกว่าหนึ่งค่าในแต่ละทรานแซคชั่น

2. แอททริบิวต์รอง (Subordinate Attribute)

แอททริบิวต์รอง เป็นแอททริบิวต์ที่ประกอบด้วย ข้อมูลภายในทรานแซคชั่นของแอททริบิวต์นั้นมีค่าเพียงหนึ่งค่าเท่านั้นในแต่ละทรานแซคชั่น

การสร้าง Large itemsets เป็นขั้นตอนหลักสำคัญสำหรับการค้นหากฎความสัมพันธ์ ดังที่ได้กล่าวมา การค้นหากฎความสัมพันธ์แบบมิติผสม เป็นการเป็นการค้นหากฎความสัมพันธ์ ที่รวมระหว่างการค้นหากฎความสัมพันธ์แบบมิติเดียว และการค้นหากฎความ สัมพันธ์หลายมิติแบบระหว่างมิติ

เงื่อนไขในกระบวนการสร้างความสัมพันธ์ของการค้นหากฎความสัมพันธ์แบบมิติผสมแบ่งออกเป็น 2 ขั้นตอน

1. ขั้นตอนการ join เพื่อสร้าง candidate 2- itemset (C_2)

จากการค้าหา Large 1 itemset ทั้งหมดของแต่ละแอททริบิวต์ในฐานข้อมูล และทำการระบุว่า ไอเท็มมาจาก แอททริบิวต์หลัก หรือ แอททริบิวต์รองเพื่อใช้สำหรับการพิจารณาการ join โดยการ join เพื่อทำการสร้าง C_2 จะตรวจสอบว่า ถ้าไอเท็มทั้งสองไอเท็มที่นำมา join เป็น ไอเท็มจากแอททริบิวต์หลัก จะทำการ join ระหว่าง L_1 แบบ intra-dimensional join และในกรณีอื่นๆ จะทำการ join แบบ inter-dimensional join เช่น L1 = {A,I_1,I_2} โดย A มาจากแอททริบิวต์รอง และ I_1,I_2 มาจาก แอททริบิวต์หลัก การ join L_1เพื่อสร้าง C_2 จะได้ผลลัพธ์ดังนี้ {A,I_1} {A,I_2} {I_1,I_2} ซึ่งจะเห็นได้ว่า {A,I_1},{A,I_2} เป็นไอเท็มเซตที่เกิดจากการ join แบบ inter-dimensional join และ {I_1,I_2} เป็น ไอเท็มเซตที่เกิดจากการ join แบบ intra-dimensional join

2. ขั้นตอนการ join เพื่อสร้าง candidate k- itemset (C_k) ที่ $k > 2$

โดยกำหนดให้

l_1 และ l_2 หมายถึง ไอเท็มเซต ที่เป็นสมาชิกของ L_{k-1}

$l_i[j]$ หมายถึง ลำดับไอเท็มที่ j ใน l_i

กระบวนการในการค้นหากฎความสัมพันธ์แบบมิติผสม แบ่งการ join ในกรณีการค้าหา candidate k- itemset (C_k) ที่ $k > 2$ ที่สำคัญออกเป็น 2 แบบ

■ การ join แบบภายในมิติ (intra-dimensional join)

เป็นการ join ในกรณีที่ทุกๆไอเท็มของไอเท็มเซต l_1 และ l_2 เป็นแอททริบิวต์หลัก โดยที่ไอเท็มลำดับที่ 1 ถึง ลำดับที่ [k-2] ใน l_1 และ l_2 เป็นไอเท็มที่เหมือนกัน และ ไอเท็มลำดับที่ [k-1] ของ l_1 น้อยกว่า ไอเท็มลำดับที่ [k-1] ของ l_2 ซึ่งเป็นเงื่อนไขตรวจสอบการซ้ำกัน โดยสามารถเขียนเป็นรูปแบบของการ join ได้ดังนี้

$$(l_1[1]=l_2[1])\cap(l_1[2]=l_2[2])\cap\ldots\cap(l_1[k-2]=l_2[k-2])\cap(l_1[k-1]<l_2[k-1])$$

ผลลัพธ์คือ $l_1[1]\,l_2[1]\ldots l_2[k-2]\,l_2[k-1]$

ตัวอย่าง กำหนดให้ $L_2=\{I_1, I_2\}, \{I_1, I_3\}$ โดย $l_1=\{I_1, I_2\}$, และ $l_2=\{I_1, I_3\}$

ผลลัพธ์ของการ join l_1 และ l_2 คือ $\{I_1, I_2, I_3\}$

■ การ join แบบระหว่างมิติ (inter-dimensional join)

เป็นการ join ในกรณีที่ทุกๆ ไอเท็มของไอเท็มเซต l_1 และ l_2 ที่มา join กัน เป็นแอททริบิวต์ที่มีทั้งแอททริบิวต์หลักและแอททริบิวต์รอง โดยที่ ไอเท็มลำดับที่ 2 ถึง ลำดับที่ [k-1] ใน l_1 เป็นไอเท็มเดียวกันกับ ไอเท็มลำดับที่ 1 ถึง [k-2] ใน l_2 และ ไอเท็มลำดับที่ 1 ของ l_1 น้อยกว่า ไอเท็มลำดับที่ [k-1] ของ l_2 โดยสามารถเขียนรูปแบบของการ join ได้ดังนี้

$$(l_1[2]=l_2[1])\cap(l_1[3]=l_2[2])\cap\ldots\cap(l_1[k-1]=l_2[k-2])\cap(l_1[1]<l_2[k-1])$$

ผลลัพธ์คือ $l_1[1]\,l_1[2]\ldots l_1[k-1]\,l_2[k-1]$

ตัวอย่าง กำหนดให้ $L_3=\{A, B, C\}, \{B, C, I_2\}$ โดย $l_1=\{A, B, C\}$ และ $l_2=\{B, C, I_2\}$

ผลลัพธ์ของการ join l_1 และ l_2 คือ $\{A, B, C, I_2\}$

การทำงานอัลกอริทึมอะพริโอริสำหรับการหากฎความสัมพันธ์แบบมิติผสมจะมีการทำงานที่คล้ายกับอัลกอริทึมอะพริโอริแต่จะแตกต่างกันตรงที่การ join กันระหว่าไอเท็มเซต ซึ่งมีขั้นตอนในการทำงานแสดงดังรูปที่ 2.6 ดังนี้

1. ทำการหาไอเท็มเซตที่เป็น L_1 จากฐานข้อมูล

2. ทำการค้นหา L_2 โดยเป็นการ join ระหว่าง L_1 กับ L_1 ด้วย procedure apriori_gen1 แสดงดังรูปที่ 2.7 เพื่อสร้าง C_2 โดยที่ l_1 กับ l_2 เป็นไอเท็มเซตที่เป็นสมาชิกใน L_1 โดยข้อกำหนดในการ join เพื่อหา C_2 ดังที่กล่าว คือ ถ้า l_1 และ l_2 เป็นแอททริบิวต์รองที่เป็นไอเท็มเซตมาจากแอททริบิวต์เดียวกัน ไม่สามารถทำการ join ระหว่างภายในของแอททริบิวต์รองที่เป็นแอททริบิวต์เดียวกันได้ เช่น แอททริบิวต์เพศ ซึ่งเป็นแอททริบิวต์รองไม่สามารถทำการ join ภายในของข้อมูลแอททริบิวต์เพศได้

นำ C_2 ที่ได้จากการ join ตรวจสอบว่าแต่ละไอเท็มเซตใน C_2 แต่ละตัวมีซับเซตทั้งหมดเป็นสมาชิกใน L_1 หรือไม่ ถ้าไอเท็มเซตที่พิจารณาไม่มีซับเซตทั้งหมดเป็นสมาชิกอยู่ใน L_1 จะทำการตัดไอเท็มเซตดังกล่าวออกจาก C_2 ทำการพิจารณา C_2 ที่ไอเท็มเซตใดใน C_2 มีค่าสนับสนุนมากกว่าหรือเท่ากับค่าสนับสนุนขั้นต่ำที่กำหนด ไอเท็มเซตที่ผ่านเกณฑ์ค่าสนับสนับสนุนจะกลายเป็น L_2

3. สร้าง L_k ตั้งแต่ k ที่มีค่าตั้งแต่ 3 เป็นต้นไป โดยเป็นการ join ระหว่าง L_{k-1} กับ L_{k-1}ด้วย procedure apriori_gen แสดงดังรูปที่ 2.8 เพื่อสร้างเป็น C_k จนกระทั่งไม่สามารถสร้าง C_k ได้ ซึ่งการ join จะมีการพิจารณารูปแบบของไอเท็มเซต 2 แบบ ดังนี้คือ กรณีเป็นการ join ด้วยแอททริบิวต์หลัก ทั้งหมดจะเป็นการ join แบบ intra-dimension หากเป็นรูปแบบอื่นๆให้ทำการ join แบบ inter-dimension

นำ C_k ที่ได้จากการ join มาทำการตรวจสอบว่าแต่ละไอเท็มเซตใน C_k มีซับเซตทั้งหมดอยู่ใน L_{k-1} หรือไม่ถ้าไอเท็มเซตที่พิจารณาไม่มีซับเซตทั้งหมดอยู่ใน L_{k-1} จะทำการตัดไอเท็มเซตนั้นออกจาก C_k จากนั้นพิจารณาค่าสนับสนุนของไอเท็มเซตแต่ละตัวภายใน C_k ว่ามีค่ามากกว่าหรือเท่ากับสนับสนุนขั้นต่ำหรือไม่ ถ้าผ่านเกณฑ์ที่กำหนดจะกลายเป็น L_k

```
1)  L1 = find_frequent_1_itemsets(D);
2)  //compress the transaction database, according
    // to the generated frequent 1-itemsets
    D' = trans_compression(D);
3)  //generate candidate 2-itemsets
    C2=apriori_gen1(L1);
4)  //generate frequent 2-itemsets
    L2=find_frequent_2_itemsets(D');
5)  //generate candidate k-itemsetsCk from frequent
    //(k-1)-itemsets Lk-1
    for ( k =3 ; Lk-1 ≠ Φ ; k++) do
    Begin
6)  //generate all the candidate k-itemsetsCk by joining
      Ck = apriori-gen(Lk-1);
7)    //use the Apriori property to eliminate
      //candidates having a subset that is not frequent
      for each transaction t ∈ D' do
8)      begin (the t equal to each Record)
9)        Ct =subset(Ck , t);
10)       for each candidates c∈Ct do
11)         c.count ++;
12)     end
13)   //all those candidate k-itemsets
      Lk ={ c∈Ck | c.count >= minsup}
      // Ck satisfying minimum support form the
      // set of frequent k-itemsets Lk
    End
14) Answer = Uk Lk ;
15) // generate rules from all frequent itemsets
    For each large itemsets Lk ∈ Answer, (k≥2) do
16)   genrules(Lk, Lk)
```

รูปที่ 2.6 อัลกอริทึมอะพริโอริสำหรับการหากฎความสัมพันธ์หลายมิติแบบมิติผสม

```
procedure  apriori_gen1(Lk-1:frequent-itemsets)
{  C[k] = null;
   for each l1 ∈ Lk-1
     for each l2 ∈ Lk-1
        if  isInnerJoin(l1) or isInnerJoin(l2)
        // if l1 or l2 can make intradimension join
       // isInnerJoin(l1) is a bool function, l1 is parameter,
       //its' function is to judge whether
       // an item l1can make intradimension join,
       //if the return value is 'true', it's allowed,
         then {
            c = l1 ⊳⊲ l2 ;
            InsertintoC[k];  }
       for each c  ∈  C[k]
        for each (k-1)-subset s of c
            if    s ∉ Lk-1
           then delete c from C[k];
}
```

รูปที่ 2.7 แสดง procedure apriori_gen1

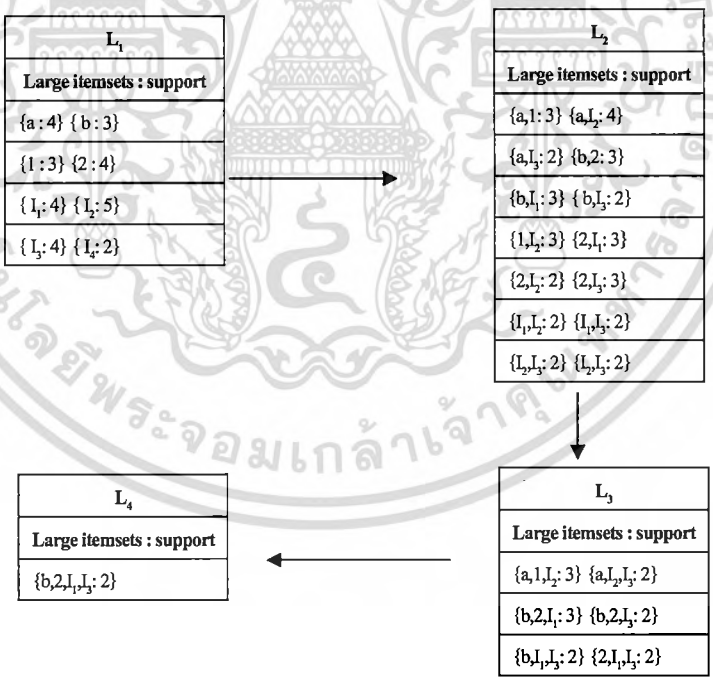
```
procedure   apriori-gen(Lk-1:frequent-itemsets)
{
   C[k] = null;
   for each l1 ∈ Lk-1
    for each l2 ∈ Lk-1
     if   not (isInnerJoin(l1)) and not (isInnerJoin(l2))
     then //if make intradimension join
     {if (l1[1]=l2[1] ∧ (l1[2]=l2[2]) ∧ ...
     ∧  (l1[k-2]=l2[k-2]) ∧ ( l1[k-1]< l2[k-1])
      then {
        c = l1 ⊳⊲ l2 ;
        InsertintoC[k];}
     }
     else //if make interdimension join
      {if(l1[2]=l2[1]) ∧ (l1[3]=l2[2]) ∧ ...
      ∧  (l1[k-1]= l2[k-2]) ∧ ( l1[1]< l2[k-1])
        then {
          c = l1 ⊳⊲ l2 ;
          InsertintoC[k];}
      }
   for each c  ∈  C[k]
     for each (k-1)-subset s of c
         if s  ∉  Lk-1
         then delete c from C[k];
}
```

รูปที่ 2.8 แสดง procedure apriori_gen

จากตัวอย่าง รูปที่ 2.9 เป็นฐานข้อมูลทรานแซคชั่นแบบหลายมิติ ที่เก็บข้อมูล อายุ พื้นที่ และ ข้อมูลการซื้อขาย ซึ่งมีแอททริบิวต์ อายุ และ พื้นที่ เป็น แอททริบิวต์รอง และ แอททริบิวต์การซื้อขายเป็น แอททริบิวต์หลัก กำหนดให้ค่า Minimum Support เท่ากับ 0.4 โดยกระบวนการในการหา Large itemsets ในฐานข้อมูลทรานแซคชั่นแบบหลายมิติดังกล่าว จะแสดงดังรูปที่ 2.10 โดยแต่ละส่วนจะทำการสร้าง Candidate itemset เพื่อค้นหา Large itemsets

TID	Age	Area	Item
1	a	1	$I_1 I_2 I_5$
2	a	1	$I_2 I_4$
3	a	2	$I_2 I_3$
4	b	2	$I_1 I_2 I_4$
5	b	2	$I_1 I_3$
6	a	1	$I_2 I_3$
7	b	2	$I_1 I_3$

รูปที่ 2.9 ตัวอย่างฐานข้อมูลทรานแซคชั่นแบบหลายมิติ



รูปที่ 2.10 กระบวนการค้นหา Large itemsets ของอัลกอริทึมอะพริโอริ
สำหรับการหากฎความสัมพันธ์แบบมิติผสม

จะสังเกตได้ว่าการค้นหากฎความสัมพันธ์แบบระหว่างมิติอย่างเดียว ไม่สามารถทราบถึงความสัมพันธ์ที่มีอยู่ภายในแอททริบิวต์หลักได้ ดังนั้น เพื่อให้สามารถค้นหากฎความสัมพันธ์ได้อย่างครบถ้วนในฐานข้อมูลทรานแซคชั่นหลายมิติ การค้นหากฎความสัมพันธ์แบบมิติผสมทำให้สามารถทราบความสัมพันธ์ที่มีระหว่างภายในแอททริบิวต์หลักที่สนใจ และยังทำให้ทราบความสัมพันธ์ของข้อมูลแบบระหว่างมิติได้

อัลกอริทึมนี้นำเสนอการค้นหากฎความสัมพันธ์แบบมิติผสมมาประยุกต์ใช้ในอัลกอริทึมอะพริโอริ โดยผู้วิจัยได้ปรับปรุงอัลกอริทึมอะพริโอริในขั้นตอนการ join ให้สามารถใช้ได้กับทรานแซคชั่นข้อมูลหลายมิติได้ อัลกอริทึมอะพริโอริสำหรับการค้นหากฎความสัมพันธ์แบบมิติผสมนี้ได้ปรับปรุงขั้นตอน การ join โดยเอาวิธีการสร้างกฎความสัมพันธ์ทั้งสองแบบคือ การ join แบบ intra-dimensional และ การ join แบบ inter-dimensional นำมาใช้ร่วมกันเพื่อให้สามารถค้นหากฎความสัมพันธ์แบบมิติผสม ทำให้ได้ค่า Large itemsets ในฐานข้อมูลที่มีความหลากหลาย ซึ่งผลลัพธ์ของกฎความสัมพันธ์ที่ได้จะหลากหลายมากกว่าการค้นหากฎความสัมพันธ์แบบหนึ่งมิติและการค้นหากฎความสัมพันธ์แบบระหว่างมิติ

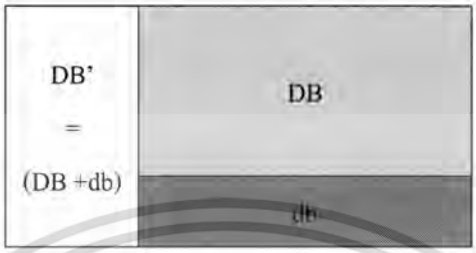
เมื่อในกรณีที่ฐานข้อมูลมีการเปลี่ยนแปลง อัลกอริทึมอะพริโอริสำหรับการหากฎความสัมพันธ์หลายมิติแบบมิติผสม ต้องทำการค้นหากฎความสัมพันธ์ของฐานข้อมูลใหม่ทั้งหมด ซึ่งในการค้นหากฎความสัมพันธ์ใหม่แต่ละครั้งจะต้องทำการสแกนข้อมูลในฐานข้อมูลใหม่ทั้งหมดเพื่อค้นหา Large itemsets ทำให้ใช้เวลานาน แทนที่จะนำความรู้เดิมที่เคยได้จากการค้นหาความสัมพันธ์ในฐานข้อมูลเดิมที่มีอยู่มาใช้ให้เกิดประโยชน์ เพื่อลดการค้นหา Large itemsets ในฐานข้อมูลใหม่ทั้งหมด

2.2 การเพิ่มขยายการค้นหากฎความสัมพันธ์ (Incremental Association Rule Discovery)

กฎความสัมพันธ์จะถูกต้องในฐานข้อมูลเดิม เมื่อมีการเปลี่ยนแปลงของฐานข้อมูลเดิม กฎความสัมพันธ์ที่ได้จากการค้นหาอาจมีการเปลี่ยนแปลงไป ดังนั้นในการค้นหากฎความสัมพันธ์ของข้อมูลเมื่อมีการเพิ่มข้อมูลทรานแซคชั่นเข้าสู่ฐานข้อมูล ทำให้เกิดกฎความสัมพันธ์ใหม่ หรือในขณะเดียวกันมีผลต่อกฎความสัมพันธ์เดิมที่ถูกสร้างไว้ เพื่อรักษาความถูกต้องของกฎความสัมพันธ์ของข้อมูลให้ถูกต้องอยู่เสมอ ทำให้ต้องมีการค้นหากฎความสัมพันธ์ใหม่เมื่อฐานข้อมูลเกิดการเปลี่ยนแปลง

โดยงานวิจัยส่วนใหญ่ในการค้นหากฎความสัมพันธ์เมื่อมีการเปลี่ยนแปลงของฐานข้อมูลจะเป็นการหาวิธีการเพื่อลดการค้นหาในฐานข้อมูลเดิม เนื่องจากฐานข้อมูลเดิมมักมีขนาดใหญ่กว่าฐานข้อมูลในส่วนที่เพิ่มข้อมูลใหม่ แนวคิดของการค้นหากฎความสัมพันธ์เมื่อมีการเพิ่มข้อมูลใหม่

แสดงดังรูป 2.11 โดยที่ฐานข้อมูลที่เพิ่มใหม่ (Increment Database: db) เมื่อรวมกับฐานข้อมูลเดิม (Original Database: DB) จะเรียกว่าฐานข้อมูลปรับปรุง (Updated Database: DB') ซึ่งภายหลังการเพิ่มข้อมูลเข้าสู่ฐานข้อมูลอาจทำให้ Large itemsets เกิดการเปลี่ยนแปลงไปจากเดิมที่เคยค้นหากฎความสัมพันธ์ไว้



รูปที่ 2.11 แสดง ฐานข้อมูล transaction สำหรับ incremental association rule mining

เหตุการณ์ที่สามารถเป็นไปได้ในของกฎความสัมพันธ์ที่มีอยู่เดิมเมื่อเกิดการเพิ่มข้อมูลใหม่เข้ามาในฐานข้อมูลประกอบด้วย 4 เหตุการณ์ [7]

1. itemsets ที่เป็น Large itemsets ในฐานข้อมูลเดิม ยังคงเป็น Large itemsets ในฐานข้อมูลที่มีการเพิ่มขึ้น
2. itemsets ที่เป็น Large itemsets ในฐานข้อมูลเดิม เปลี่ยนเป็น Small itemset ในฐานข้อมูลที่มีการเพิ่มขึ้น
3. itemsets ที่เป็น Small itemsets ในฐานข้อมูลเดิม เปลี่ยนเป็น Large itemsets ในฐานข้อมูลที่มีการเพิ่มขึ้น
4. itemsets ที่เป็น Small itemsets ในฐานข้อมูลเดิม ยังคงเป็น Small itemset ในฐานข้อมูลที่มีการเพิ่มขึ้น

ทั้งนี้งานวิจัยทางด้านการเพิ่มขยายกฎความสัมพันธ์ได้จำแนกออกเป็น 2 ประเด็นดังนี้

1. กฎความสัมพันธ์ที่หาได้จะไม่มีการเปลี่ยนแปลงเมื่อเวลาเปลี่ยนไป (association rule stable over time)

ในกฎความสัมพันธ์ที่หาได้จะไม่มีการเปลี่ยนแปลงเมื่อเวลาเปลี่ยนไป หมายถึง ข้อมูลเก่า (old dataset) และข้อมูลใหม่ (new dataset) มีค่าความสำคัญของข้อมูลเท่ากัน ซึ่งผลลัพธ์ของการค้นหากฎความสัมพันธ์จะเหมือนกับหลักการของอัลกอริทึมอะพริโอริ โดยทำการประมวลผลทั้งข้อมูลใหม่และข้อมูลเดิมทั้งหมดรวมกัน ซึ่งสามารถแบ่งออกเป็น 3 กลุ่ม [3]

1.1 Apiori base เป็นการนำหลักการของอัลกอริทึมอะพริโอริมาใช้ในการเพิ่มขยายกฎความสัมพันธ์ อัลกอริทึมที่ได้รับความนิยมได้แก่ FUP

1.2 Partition-base เป็นการนำเทคนิคการแบ่งข้อมูลที่อยู่ในฐานข้อมูลออกเป็นส่วนๆ ที่เรียกว่า partition มาใช้ในการเพิ่มขยายกฎความสัมพันธ์ อัลกอริทึมที่ได้รับความนิยมได้แก่ Sliding-Window Filtering : SWF

1.3 Pattern-growth base เป็นการนำเทคนิคโดยใช้หลักการของ FP-tree มาใช้ในการเพิ่มขยายกฎความสัมพันธ์ เช่น DB-tree และ PotFP-tree

2. กฎความสัมพันธ์ที่หาได้จะมีการเปลี่ยนแปลงเมื่อเวลาเปลี่ยน (association rules are not stable overtime)

ในประเด็นนี้มีสมมติฐานที่ว่า กำหนดให้ข้อมูลเก่า และข้อมูลใหม่มีความสำคัญไม่เท่ากันตัวอย่างของงานวิจัยในกลุ่มนี้ได้แก่ Weighting technique และ Time influence Function

2.2.1 การค้นหากฎความสัมพันธ์ด้วย FUP Algorithm [5]

อัลกอริทึม FUP (Fast UPdate Algorithm) เป็นงานวิจัยแรกที่นำเสนอเทคนิคสำหรับการแก้ปัญหาการค้นหากฎความสัมพันธ์เมื่อมีการเพิ่มข้อมูลเข้าสู่ฐานข้อมูล เพื่อที่จะรักษากฎความสัมพันธ์ให้ถูกต้องอยู่เสมอเมื่อมีการเพิ่มข้อมูลเข้ามาในฐานข้อมูล โดยการทำงานของ FUP อาศัยหลักการเดียวกันกับอัลกอริทึมอะพริโอริ ซึ่งมีการวนรอบการทำงานซ้ำเพื่อค้นหาความสัมพันธ์ของข้อมูล โดยจะเริ่มตั้งแต่ 1-itemset ไปจนถึง k-itemset ซึ่ง Candidate itemset แต่ละรอบจะได้มาจาก Large itemsets ที่พบในรอบก่อนหน้า โดยทำงานภายใต้การใช้ค่าสนับสนุนขั้นต่ำ และ ค่าความมั่นใจขั้นต่ำคงที่ ซึ่งเป้าหมายของอัลกอริทึม FUP มีการนำความรู้ที่เคยได้จากการทำไมนิ่งฐานข้อมูล นั้นคือ Large itemsets ในฐานข้อมูลเดิมก่อนหน้าที่จะมีการเพิ่มข้อมูลเข้าสู่ฐานข้อมูลมาใช้ประโยชน์ เพื่อลดการค้นหาไอเท็มเซตในทุกทรานแซคชั่นที่อยู่ในฐานข้อมูลทั้งหมดสำหรับการนับค่าสนับสนุนของแต่ละไอเท็มเซต ซึ่งจะเห็นได้ว่าแตกต่างจากอัลกอริทึมอะพริโอริ ที่ต้องทำการค้นหาค่าสนับสนุนของแต่ละไอเท็มเซต โดยการเข้าไปค้นหาในฐานข้อมูลใหม่ทั้งหมด โดยไม่นำ Large itemsets ที่เป็นความรู้ก่อนหน้าจากการค้นหากฎความสัมพันธ์ในฐานข้อมูลเดิม ที่สามารถนำมาใช้ให้เกิดประโยชน์ได้

ความหมายของสัญลักษณ์ต่างๆที่ใช้ในอัลกอริทึม FUP มีดังนี้

DB หมายถึง original database

db หมายถึง increment database

D หมายถึง จำนวน transaction ที่มีอยู่ในส่วน original database

d หมายถึง จำนวน transaction ที่มีอยู่ในส่วน increment database

s หมายถึง ค่า minimum support

C_k หมายถึง Candidate itemset เมื่อ k=1, 2 ..., k

L_k หมายถึง Large k-itemset ใน original database เมื่อ k=1, 2 ... k

L'_k หมายถึง Large k-itemset ใน updated database เมื่อ k=1, 2 ..., k

X หมายถึง ไอเท็มเซต X

$X.support_D$ หมายถึง ค่าความถี่ (support) ของไอเท็มเซต X ใน original database

$X.support_d$ หมายถึง ค่าความถี่ (support) ของไอเท็มเซต X ใน increment - database

$X.support_{UD}$ หมายถึง ค่าความถี่ (support) ของไอเท็มเซต X ใน updated - database

การทำงานของอัลกอริทึม FUP จะแบ่งการทำงานหลักออกเป็น 2 ส่วน คือ การค้นหา L'_1 และ การค้นหา L'_k โดยที่ $k \geq 2$ โดยขั้นตอนทั้งสองส่วนอธิบายดังนี้

1. การค้นหา L'_1

เป็นการค้นหา L_1' โดยทำการสร้าง C_1 (Candidate 1-itemset) โดยทำการค้นหาใน db และค่าสนับสนุนของแต่ละไอเท็มใน db ที่จะมาเป็น L_1' เพื่อใช้ในการปรับปรุงค่าความถี่ของไอเท็ม และ prune ไอเท็มที่มีค่าสนับสนุนน้อยกว่าค่าสนับสนุนขั้นต่ำที่กำหนด โดยมีหลักการพิจารณาดังนี้

1.1 กรณีถ้า $X \in L_1$ นำค่า support ของไอเท็ม X ใน DB และ db มารวมกัน นั่นคือ $X.support_{UD} = X.support_D + X.support_d$ แล้วทำการตรวจสอบค่า support ที่ได้ว่าผ่านค่าสนันสนุนขั้นต่ำของฐานข้อมูลปรับปรุง ($s \times (D+d)$) หรือไม่ โดย

1.1.1 ถ้า $X.support_{UD} \geq s \times (D+d)$ แสดงว่า ไอเท็ม X สามารถเป็น L_1' ในฐานข้อมูลที่ปรับปรุงได้ จะได้ว่า $X \in L_1'$ และเรียกไอเท็ม X นั้นว่า winner item

1.1.2 ถ้า $X.support_{UD} < s \times (D+d)$ แสดงว่า ไอเท็ม X ไม่สามารถเป็น L_1' ในฐานข้อมูลที่ปรับปรุงได้ เรียกไอเท็ม X นั้นว่า loser แล้วจะทำการลบ (prune) ไอเท็ม X ออกไป

1.2 กรณีถ้า $X \notin L_1$ จะมีการพิจารณา 2 ส่วน คือ

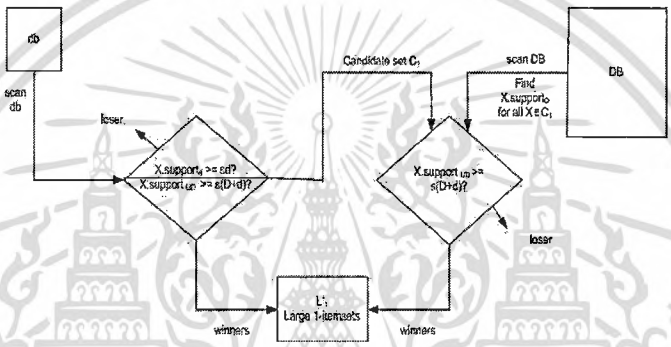
1.2.1 พิจารณาเพื่อทำการลบไอเท็มที่ไม่มีโอกาสเป็น L_1' โดยตรวจสอบว่า ถ้า $X \notin L_1$ และ $X.support_d < (s \times d)$ แสดงว่า ไอเท็มนั้นเป็น lose item แล้วทำการลบไอเท็ม X นั้นทิ้งไป ในส่วนนี้จะเป็นการช่วยในการลดจำนวนการค้นหาไอเท็มใน DB

1.2.2 พิจารณาไอเท็มที่มีโอกาสเป็น L_1' โดยตรวจสอบว่า ถ้า $X \notin L_1$ และ $X.support \geq (s \times d)$ นำ ไอเท็ม X ไปค้นหาความถี่ใน DB เพื่อหาค่า $X.support_{UD}$ แล้วนำค่าที่ได้มาตรวจสอบ โดย

■ ถ้า $X.support_{UD} \geq s \times (D+d)$ แสดงว่า ไอเท็ม X เป็น winner item โดยจะได้ว่า $X \in L_1'$

■ ถ้า $X.support_{UD} < s \times (D+d)$ แสดงว่า ไอเท็ม X เป็น lose item และทำการลบไอเท็ม X ทิ้ง

เมื่อเสร็จในขั้นตอนการทำงานจะได้ Large 1- itemset (L_1') ในฐานข้อมูลที่ถูกปรับปรุงแล้ว กระบวนการทำงานในรอบแรกเพื่อหา Large 1-itemset ของ FUP แสดงดังรูปที่ 2.12 และการทำงานในรอบนี้ของอัลกอริทึมแสดงดังรูปที่ 2.13



รูปที่ 2.12 แสดงขั้นตอนการทำงานสำหรับหา Large 1-itemset ของอัลกอริทึม FUP

2. การค้นหา L'_k โดยที่ $k \geq 2$

การทำงานในส่วนนี้จะเป็นการหา Large k-itemset เมื่อ $k \geq 2$ แสดงอัลกอริทึมดังรูปที่ 2.14 ขั้นตอนนี้จะทำการหา C_k เมื่อ k >= 2 โดยมีวิธีการเช่นเดียวกับอะพริโอริ คือ นำ L_1' มาทำการ join ($L_1' \bowtie L_1'$) และในส่วนของ FUP สำหรับก่อนการ join จะนำหลักการการทำงานที่เป็นการหา loser item ที่ไม่สามารถเป็น L_k เพื่อลดการค้นหา k-itemset ใน db โดยจะทำการพิจารณาว่า ในรอบที่ k เมื่อ $k \geq 2$ " *ถ้า ไอเท็ม X ใดๆ ที่เป็น loser item ในรอบ k-1 แล้ว ไอเท็มเซตใดๆ ของ L_k ในฐานข้อมูลเดิมที่มีไอเท็ม X เป็นซับเซตจะไม่สามารถเป็น winner item ในรอบที่ k ในฐานข้อมูลปรับปรุงได้*" จะมีการทำงานดังนี้

2.1 หาไอเท็มเซตที่เป็นสมาชิกของ L_2 ที่สามารถเป็น L_2' คือการหาไอเท็มเซต

ขั้นตอนนี้จะมีการลบ loser item ที่ไม่สามารถเป็น L_2 ได้เพื่อลดจำนวนไอเท็มเซตที่ใช้ในการค้นหาใน db โดยพิจารณา การลบ loser item จากไอเท็มที่เป็นสมาชิกอยู่ใน L_1 แต่ไม่ได้

เป็นสมาชิกอยู่ใน L_1' ($L_1 - L_1'$) นั้นคือ ไอเท็มเซตใน L_2 ใดๆ ที่มีไอเท็มที่เป็นสมากชิกใน $L_1 - L_1'$ จะไม่สามารถเป็น Large item ได้ เพราะฉนั้น ไอเท็มเซตนั้นใน L_2 จะถูกตัดทิ้ง

ตัวอย่าง

$L_1 = \{I_1\}, \{I_2\}, \{I_4\}$ $L_1' = \{I_1\}, \{I_2\}, \{I_3\}$ $L_2 = \{I_1,I_2\}, \{I_1, I_4\}$

ดังนั้น $L_1 - L_1' = \{I_4\}$

จะได้ว่า $L_2 = \{I_1 I_2\}$

จากตัวอย่างจะเห็นได้ว่า I_4 เป็นสมาชิกอยู่ใน L_1 แต่ไม่เป็นสมาชิกอยู่ใน L_1' เพราะฉะนั้น ไอเท็มเซตใดใน L_2 ที่มี I_4 เป็นซับเซตจะถูกเรียกว่า loser item และถูกลบออกจาก L_2 ก่อนทำการค้นหาใน db เพื่อหา L_2' นั้นคือ $\{I_1, I_4\}$ จะถูกลบออกจาก L_2 ดังนั้น L_2 จะประกอบด้วยสมาชิกที่เหลืออยู่ $L_2 = \{I_1, I_2\}$ เท่านั้นที่จะนำไปค้นหาใน db เพื่อปรับปรุงค่าสนับสนุน

เมื่อทำการค้นหาสมาชิก L_2 ใน db แล้วได้ค่าสนับสนุนใหม่ซึ่งเป็นค่าสนับสนุนของฐานข้อมูลที่ปรับปรุง จะมีการพิจารณาเพื่อหา L_2 ที่สามารถเป็น L_2' ดังนี้

■ ถ้า $X.support_{UD} \geq s \times (D+d)$ แสดงว่า ไอเท็ม X สามารถเป็น L_2' ในฐานข้อมูลที่ปรับปรุงได้ จะได้ว่า $X \in L_1'$ และเรียกไอเท็ม X นั้นว่า winner item

■ ถ้า $X.support_{UD} < s \times (D+d)$ แสดงว่า ไอเท็ม X ไม่สามารถเป็น L_2' ในฐานข้อมูลที่ปรับปรุงได้ เรียกไอเท็ม X นั้นว่า loser แล้วจะทำการลบ (prune) ไอเท็ม X ออกไป

2.2 หา Large 2-itemset

ในขั้นตอนนี้จะเป็นการ join ระหว่าง $L_1' * L_1'$ เพื่อสร้าง candidate 2-itemset (C_2) นำไปค้นหาใน db และหา L_2' โดยพิจารณาดังนี้

2.2.1 กรณี $X \in C_2$ และ $X \in L_2'$

ทำการตัดไอเท็ม X ออกจาก C_2 โดยไม่ต้องนำไปค้นหาใน db เพราะ ไอเท็ม X เป็นสมาชิกอยู่ที่ L_2' แล้ว ซึ่งได้จากการคำนวณใน ขั้นตอนที่ 2.1

2.2.2 กรณี $X \in C_2$ และ $X \notin L_2'$

นำไอเท็ม X ดังกล่าวไปทำการค้นหาใน db แล้วตรวจสอบว่า

2.2.1 ถ้า $X.support_d < (s \times d)$ ให้ลบไอเท็ม X ออกจาก C_2 และไม่ต้องนำไปค้นหาต่อใน DB

2.2.2 ถ้า $X.support_d \geq (s \times d)$ ให้นำเอาไอเท็ม X ไปค้นหาต่อใน DB เพื่อนทำการปรับปรุงค่าสนับสนุน แล้วทำการตรวจสอบต่อว่า

■ ถ้า $X.support_{UP} \geq s \times (D+d)$ ให้เพิ่มไอเท็ม X เป็นสมาชิกของ L_2'

■ ถ้า $X.support_{UP} < s \times (D+d)$ ให้ลบไอเท็ม X ออกไป

เมื่อเสร็จสิ้นในขั้นตอนนี้แล้วผลลัพธ์ที่ได้คือ Large 2-itemset (L_2') ในฐานข้อมูลที่ถูกปรับปรุง

2.3 ทำการวนรอบซ้ำเช่นเดียวกับขั้นตอนที่ 2.1 เพื่อหา k-itemset ($k \geq 3$) ต่อไป

จะเห็นได้ว่าอัลกอริทึม FUP มีการนำเอา Large itemsets ที่ได้จากค้นหาจากฐานข้อมูลเดิมเพื่อนำมาใช้ประโยชน์เมื่อมีการเพิ่มขึ้นของข้อมูลทรานแซคชั่นเข้าสู่ฐานข้อมูล จึงสามารถลดจำนวน Candidate Itemsets ที่เกิดจากฐานข้อมูลที่เพิ่มขึ้น เพื่อจะนำไปค้นหาในฐานข้อมูลเดิมทำให้การค้นหาไอเท็มในฐานข้อมูลเดิมน้อยลง แต่อัลกอริทึม FUP นั้นสามารถใช้ได้ในกรณีของการเพิ่มข้อมูลทรานแซคชั่นใหม่เข้าไปในฐานข้อมูลเดิมเท่านั้น และยังไม่สามารถทำการค้นหากฎความสัมพันธ์ที่เป็นฐานข้อมูลทรานแซคชั่นข้อมูลเป็นแบบหลายมิติได้

Algorithm 1 FUP: A fast update algorithm for maintenance of association rules on database updates.

Input: (1) DB: the original database (with its size, i.e., the total number of transactions, equal to D); (2) L_k: the set of all large k-itemsets in DB, where $k = 1, \ldots, r$; (3) db: an increment database (with its size equal to d); and (4) s: the minimum support threshold.

Output: L': The set of all large itemsets in $DB \cup db$.

Method:

The 1st iteration: /* find L'_1, the set of all large 1-itemsets in $DB \cup db$ */

```
W = L_1; C = ∅; L'_1 = ∅; P = ∅;
/* W: winners, C: candidate sets,
L'_1: initialized, P: for optimization */
for_all T ∈ db do /* scan db */
 for_all 1-itemset X ⊆ T do {
  if X ∈ W then X.support_d++;
  else {
   if X ∉ C
   then { C = C ∪ {X}; X.support_d = 0; }
   /*init the support count and add X into C */
   X.support_d++; }
 };
for_all X ∈ W do /*put winners into L'_1 */
 if X.support_UD ≥ s × (D + d)
  then L'_1 = L'_1 ∪ {X};
for_all X ∈ C do /*prune candidate sets in C */
 if X.support_d < s × d
  then { C = C − {X}; P = P ∪ {X}; }
  /* P will be used for optimization. */
for_all T ∈ DB do /* scan DB */
 for_all 1-itemset X ⊆ T do {
  if X ∈ C then X.support_D++;
  if X ∈ P then removes X from T;
  /* Transaction T is reduced */
 };
for_all X ∈ C do /*put winners into L'_1*/
 if X.support_UD ≥ s × (D + d)
  then L'_1 = L'_1 ∪ {X};
return L'_1. /* end of the 1st iteration */
```

รูปที่ 2.13 อัลกอริทึม FUP สำหรับหา Large 1-itemset

The k-th iteration: /* for $k = 2$ or larger, repeat this program fragment to find L'_k, the set of all large k-itemsets in the updated database, until either L'_k returned is empty or $db = \emptyset$ */

```
W = L_k; L'_k = ∅ ;
 /* W: winners; L'_k initialized */
C = apriori-gen(L'_k−1) −L_k;
 /* the size-k candidate sets */
for_all k-itemset X ∈ W do
 /* prune off losers in W */
 for_all (k-1)-itemset Y ∈ L_k−1 − L'_k−1 do
  if Y ⊆ X then { W = W − {X}; break; }
for_all T ∈ db do { /* scan db */
 for_all X ∈ Subset(W, T) do X.support_d++;
  /* Subset(W, T) returns all the sets in W
   contained in T [2] */
 for_all X ∈ Subset(C, T) do X.support_d++;
  /* find support of all X ∈ C */
 Reduce_db(T);
  /*Some items in transactions in db can
   be removed, discussed in next section*/
}
for_all X ∈ W do
 /*put the winners from W into L'_k */
 if X.support_UD ≥ s × (D + d)
  then L'_k = L'_k ∪ {X};
for_all X ∈ C do /* prune candidate sets in C */
 if X.support_d < s × d then C = C − {X};
for_all T ∈ DB do { /* scan DB */
 for_all X ∈ Subset(C, T) do X.support_D++;
 Reduce_Db(T); }
  /* Some items in transactions in DB can
   be removed, discussed in next section */
for_all X ∈ C do
 /* put the winners from C into L'_k */
 if X.support_UD ≥ s × (D + d)
  then L'_k = L'_k ∪ {X};
return L'_k. /* The end of the k-th iteration */
```

รูปที่ 2.14 อัลกอริทึม FUP สำหรับหา Large k-itemset ที่ k □ 2

2.2.2 การค้นหากฎความสัมพันธ์แบบมิติผสมสำหรับการเพิ่มข้อมูล (HDFUP Algorithm) [4]

อัลกอริทึม HDFUP เป็นการค้นหากฎความสัมพันธ์แบบมิติผสมสำหรับการเพิ่มข้อมูล โดยนำเอา อัลกอริทึม FUP มาทำการปรับปรุง เพื่อให้สามารถค้นหากฎความสัมพันธ์หลายมิติแบบมิติผสมในฐานข้อมูลที่มีทรานแซคชั่นข้อมูลแบบหลายมิติ โดยได้นำขั้นตอนการสร้างความสัมพันธ์ของข้อมูลแบบมิติผสมมาใช้ปรับปรุงในขั้นตอนการ join ของ อัลกอริทึม FUP เพื่อสร้างความสัมพันธ์ของข้อมูล ให้สามารถใช้ได้กับฐานข้อมูลทรานแซคชั่นข้อมูลแบบหลายมิติ

การทำงานของอัลกอริทึม HDFUP จะแบ่งการทำงานหลักออกเป็น 3 ส่วน คือ การค้นหา L'_1, การค้นหา L'_k โดยที่ $k = 2$ และ การค้นหา L'_k โดยที่ $k \geq 3$ โดยขั้นตอนทั้งสามส่วนอธิบายดังนี้

ความหมายของสัญลักษณ์ต่างๆที่ใช้ในอัลกอริทึม HDFUP

DB หมายถึง original database

db หมายถึง increment database

D หมายถึง จำนวน transaction ที่มีอยู่ใน original database

d หมายถึง จำนวน transaction ที่มีอยู่ใน increment database

s หมายถึง ค่า minimum support

C_k หมายถึง Candidate itemset เมื่อ k=1,2,…,k

L_k หมายถึง Large k-itemset ใน original database เมื่อ k=1,2,…,k

L'_k หมายถึง Large k-itemset ใน updated database เมื่อ k=1,2,…,k

X หมายถึง ไอเท็มเซต X

$X.support_D$ หมายถึง ค่าความถี่ (support) ของไอเท็มเซต X ใน original database

$X.support_d$ หมายถึง ค่าความถี่ (support) ของไอเท็มเซต X ใน increment database

$X.support_{UD}$ หมายถึง ค่าความถี่ (support) ของไอเท็มเซต X ใน updated database

1. รอบการทำงานที่ 1 เป็นการค้นหา L'_1

การทำงานในส่วนนี้จะมีลักษณะคล้ายกับการทำงานของ อัลกอริทึม FUP แสดงอัลกอริทึมดังรูป 2.15 ซึ่งเป็นการ prune ไอเท็มที่เป็น loser item และหาไอเท็มที่เป็น winner item ที่เป็น Large 1-itemset

1.1 ทำการค้นหาใน ฐานข้อมูลที่เพิ่มขึ้นมาใหม่ เพื่อทำการปรับปรุงค่า support ของไอเท็มเพื่อหาไอเท็มที่เป็น winner item และ prune ไอเท็มที่เป็น loser item โดยมีหลักการพิจารณาดังนี้

1.1.1 กรณีถ้า $X \in L_1$ นำค่า support ของไอเท็ม X ใน DB และ db มารวมกัน นั้นคือ $X.support_{UD} = X.support_D + X.support_d$ แล้วทำการตรวจสอบค่า support ที่ได้ว่าผ่านค่าสนันสนุนขั้นต่ำของฐานข้อมูลปรับปรุง ($s \times (D+d)$) หรือไม่ โดย

■ ถ้า $X.support_{UD} \geq s \times (D+d)$ แสดงว่า ไอเท็ม X สามารถเป็น L_1' ในฐานข้อมูลที่ปรับปรุงได้ จะได้ว่า $X \in L_1'$ และเรียกไอเท็ม X นั้นว่า winner item

■ ถ้า $X.support_{UD} < s \times (D+d)$ แสดงว่า ไอเท็ม X ไม่สามารถเป็น L_1' ในฐานข้อมูลที่ปรับปรุงได้ เรียกไอเท็ม X นั้นว่า loser แล้วจะทำการลบ (prune) ไอเท็ม X ออกไป

1.1.2 กรณีถ้า $X \notin L_1$ จะมีการพิจารณา 2 ส่วน คือ

■ พิจารณาเพื่อทำการลบไอเท็มที่ไม่มีโอกาสเป็น L_1' โดยตรวจสอบว่าถ้า $X \notin L_1$ และ $X.support_d < (s \times d)$ แสดงว่า ไอเท็มนั้นเป็น lose item หมายถึงไม่สามารถเป็นเกิดขึ้นใน DB ได้ แล้วทำการลบไอเท็ม X นั้นทิ้ง ในส่วนนี้จะเป็นการช่วยในการลดจำนวนการค้นหาไอเท็มใน DB

■ พิจารณาไอเท็มที่มีโอกาสเป็น L_1' โดยตรวจสอบจาก ถ้า $X \notin L_1$ และ $X.support_d \geq (s \times d)$ นำ ไอเท็ม X ไปค้นหาความถี่ใน DB เพื่อหาค่า $X.support_{UD}$ แล้วนำค่าที่ได้มาตรวจสอบ โดย

▶ ถ้า $X.support_{UD} \geq s \times (D+d)$ แสดงว่า ไอเท็ม X เป็น winner item โดยจะได้ว่า $X \in L_1'$

▶ ถ้า $X.support_{UD} < s \times (D+d)$ แสดงว่า ไอเท็ม X เป็น Lose item และทำการลบไอเท็ม X ทิ้ง

เมื่อเสร็จในขั้นตอนการทำงานจะได้ Large 1- itemset (L_1') ในฐานข้อมูลที่ถูกปรับปรุงแล้ว

2. รอบการทำงานที่ 2 เป็นการการค้นหา L'_2

การทำงานในส่วนนี้จะเป็นการหา Large 2-itemset แสดงดังรูป 2.16 โดยทำการหา loser itemset ที่ไม่สามารถเป็น L_2 ได้เพื่อลดการค้นหา 2- itemset ในฐานข้อมูลที่เพิ่มขึ้นมาใหม่โดยใช้แนวคิด ที่ว่า "*ถ้าไอเท็มใดๆ ที่เป็น loser item ในการทำงานรอบก่อนหน้าแล้ว itemset ใดๆ ของ Large itemsets ในฐานข้อมูลเดิมที่มีไอเท็มดังกล่าวเป็นซับเซตอยู่จะไม่สามารถเป็น winner item ในรอบนั้นได้*" จากแนวคิดดังกล่าวจะมีการทำงานดังนี้

2.1 หา itemset ที่เป็นสมาชิกของ L_2 และ L_2' คือเป็นการหา itemset ที่เป็น large itemsets ทั้งในฐานข้อมูลเดิม และฐานข้อมูลที่ปรับปรุงแล้ว

ขั้นตอนนี้จะ prune loser item ที่ไม่สามารถเป็น L_2 เพื่อลดการค้นหา L_2 ใน db โดยพิจารณาจาก $Y = L_1 - L_1'$ หมายถึง ไอเท็ม Y ใดๆที่เป็นสมาชิกของ L_1 แต่ไม่เป็นสมาชิกของ L_1' นั้น คือเมื่อ $X \in L_2$ ที่มีไอเท็ม Y เป็นซับเซต จะไม่สามารถเป็น Large itemsets ได้ ดังนั้นไอเท็ม X ดังกล่าวจะถูกตัดทิ้งไป แล้วนำสมาชิกที่เหลืออยู่ใน L_2 ไปค้นหาในฐานข้อมูลที่เพิ่มเข้ามาใหม่ เพื่อปรับปรุงค่า support

ตัวอย่าง

$L_1 = \{a\}, \{I_2\}, \{I_4\}$ $L_1' = \{a\}, \{I_2\}, \{I_5\}$ $L_2 = \{a, I_2\}, \{a,I_4\}\{I_2, I_4\},$

ดังนั้น $L_1 - L_1' = \{I_4\}$

จะได้ว่า $L_2 = \{a, I_2\}$

จากตัวอย่างจะเห็นได้ว่า I_4 เป็นสมาชิกอยู่ใน L_1 แต่ไม่เป็นสมาชิกอยู่ใน L_1' เพราะฉะนั้น ไอเท็มเซตใดใน L_2 ที่มี I_4 เป็นซับเซตจะถูกเรียกว่า loser item และถูกลบออกจาก L_2 ก่อนทำการค้นหาใน db เพื่อหา L_2' นั้นคือ $\{a, I_4\}$ และ $\{I_2, I_4\}$ จะถูกลบออกจาก L_2 ดังนั้น L_2 จะประกอบด้วยสมาชิกที่เหลืออยู่ $L_2 = \{a, I_2\}$ เท่านั้นที่จะนำไปค้นหาใน db เพื่อปรับปรุงค่าสนับสนุนแล้วนำมาพิจารณาว่า

- ถ้า $X.support_{UD} \geq s \times (D+d)$ itemset นั้นๆจะเป็น winner itemset แล้วจะถูกเก็บใน L_2'
- ถ้า $X.support_{UD} < s \times (D+d)$ itemset นั้นๆจะเป็น loser itemset แล้วจะถูกตัดทิ้ง

2.2 หา new Large 2-itemset(L_2') ใน db

ในขั้นตอนนี้จะเป็นการ join ระหว่าง $L_1' \bowtie L_1'$ตาม procedure apriori_gen1 เพื่อสร้าง candidate 2-itemset (C_2) แสดงดังรูปที่ 2.17 โดยมีข้อกำหนดในการ join คือไม่สามารถ join กันด้วยแอททริบิวต์รองที่มาจากแอททริบิวต์เดียวกันได้ เช่น แอททริบิวต์อายุ ซึ่งเป็นแอททริบิวต์รองไม่สามารถ join กับแอททริบิวต์อายุที่อยู่ภายในแอททริบิวต์เดียวกันได้ นำ candidate 2-itemset ที่ได้ ไปค้นหาใน db และหา L_2' โดยพิจารณาดังนี้

2.2.1 กรณี $X \in C_2$ และ $X \in L_2'$

ทำการตัดไอเท็ม X ออกจาก C_2 โดยไม่ต้องนำไปค้นหาใน db เพราะไอเท็ม X เป็นสมาชิกอยู่ที่ L_2' แล้ว ซึ่งได้จากการคำนวณใน ขั้นตอนที่ 2.1

2.2.2 กรณี $X \in C_2$ และ $X \notin L_2'$

นำเอาไอเท็ม X ดังกล่าวไปทำการค้นหาใน db แล้วทำการตรวจสอบว่า

■ ถ้า $X.support_d < (s \times d)$ ให้ลบไอเท็ม X ออกจาก C2 และไม่ต้องนำไปค้นหาต่อใน DB

■ ถ้า $X.support_d \geq (s \times d)$ ให้นำเอาไอเท็ม X ไปค้นหาต่อใน DB เพื่อนทำการปรับปรุงค่าสนับสนุน แล้วทำการตรวจสอบต่อว่า

▶ ถ้า $X.support_{UP} \geq s \times (D+d)$ ให้เพิ่มไอเท็ม X เป็นสมาชิกของ L_2'

▶ ถ้า $X.support_{UP} < s \times (D+d)$ ให้ลบไอเท็ม X

เมื่อเสร็จสิ้นในขั้นตอนนี้แล้วผลลัพธ์ที่ได้คือ Large 2-itemset (L_2') ในฐานข้อมูลที่ถูกปรับปรุง

```
Input: DB: the original database (D is equal to total number
       of transactions);
       L1: the set of all large k- itemsets in DB,
       where k= 1, ..., r;
       db: an increment database (with its size equal to d);
Output: L': The set of all large itemsets in DB∪ db.
Method:
       The 1st iteration:/* find L'1, the set of all
       large 1-itemsets in DB ∪ db */
       W = L1; C = φ ; L1'= φ ; P = φ ;
       /* W: winners, C: candidate sets,L'1: initialized,
       P: for optimization */
       for all T ∈ db do /* scan db */
        for all 1-itemset X ∈ T do {
          if X ∈ W then X.support_d++;
          else {
            if X ∉ C
             then { C = C ∪ {X}; X.support_d = 0;}
             /*init the support cont and add X into C */
             X.supprt_d++;}
        };
       for all X ∈ W do /* put winners into L'1 */
        if X.support_UD ≥ s ×(D + d)
         then L1' = L'1 ∪ {X};
       for all X ∈ C do /*prune candidate sets in C*/
        if X.support_UD < s ×(D + d)
         then { C = C − {X}; P = P ∪ {X};}
         /* P will be used for optimization */
       for all T ∈ DB do /* scan DB */
        for all 1-itemset X ⊆ T do {
         if X ∈ C then X.support_D++;
         if X ∈ P then removes X from T;
         /* Transaction T is reduced */
        };
       for all X ∈ C do /* put winners into L1' */
        if X.supportUD ≥ s × (D + d)
         then L1' = L1' ∪ {X};
       return L1'. /* end of the 1st iteration */
```

รูปที่ 2.15 อัลกอริทึม HDFUP สำหรับหา Large 1-itemset

3. รอบการทำงานที่ k โดย $k \geq 3$

ในขั้นตอนนี้จะมีลักษณะการทำงานเหมือนกับขั้นตอนในรอบที่ 2 แต่จะแตกต่างกันตรงขั้นตอนการ join จะใช้ procedure apriori_gen2 แสดงดังรูปที่ 2.18 ในการ join โดยข้อกำหนดในการพิจารณาการ join ของไอเท็มเซต มี 2 กรณี คือ

3.1 ถ้าเป็นการ join ด้วยภายในแอททริบิวต์หลักเดียวกันเองการ join จะเป็นแบบ intra-dimension join

3.2 แต่หากเป็นรูปแบบอื่นๆให้ทำการ join แบบ inter-dimension join

ทำการวนรอบซ้ำ เพื่อหา k-itemset ($k \geq 3$) ต่อไปจนไม่สามารถทำการ join เพื่อหา k-itemset ต่อได้

```
The k-th iteration: /*for k = 2 or larger, repeat this program
fragment to find L'k, the set of all large k-itemsets in the
updated database, until either L'k returned is empty or db = φ */
W = Lk; Lk' = φ;
/*W: winners; Lk': initialized */
if k = 2
  then {C = apriori_gen1(L'k-1) - Lk; }
  else {C = apriori_gen2(L'k-1) - Lk; };
  /* the size-k candidate sets */
for all k-itemset X∈ W do
/* prune off losers in W */
  for all (k-1)-itemset Y ∈ Lk-1 − L'k-1 do
    if Y ⊆ X then { W = W − {X}; break;}
for all T ∈ db do { /* scan db */
  for all X ∈ (W,T) do X.supportd++;
  /* Subset(W,T) returns all the sets in W contained in T*/
  for all X ∈ Subset(C,T) do X.supportd++;
  /* find support of all X ∈ C */
  Reduce db (T);
  /* Some items in transactions in db can be removed,
 discussed in next section */
}
for all X ∈ W do
/* put the winners from W into L'k */
if X.supportUD ≥ s × (D+d)
  then L'k = L'k ∪ {X};
for all X ∈ C do /* prune candidate sets in C*/
  if X.supportd < s × d then C = C − {X};
for all T ∈ DB do { /* scan DB */ }
  for all X ∈ Subset(C,T) do X.supportD++
  Reduce_DB(T);
  /* Some items in transactions in DB can be removed,
 discussed in next section */
for all X ∈ C do
  if X.supportUD ≥ s × (D + d)
    then L'k = L'k ∪ {X};
return Lk'. /* the end of the k-th iteration */
```

รูปที่ 2.16 อัลกอริทึม HDFUP สำหรับหา Large k-itemset ที่ $k \geq 2$

```
procedure apriori_gen1 (L'k-1: Large itemsets)
{  C = null;
   for each l1  ∈ L'k-1
    for each l2  ∈ L'k-1
     if isInnerJoin (l1) or isInnerJoin (l2)
      then {
        c = l1  ⊳⊲ l2 ;
        InsertInto C
     for each c  ∈ C
      for each (k-1)-subset s of c
        if  s  ∉ L'k-1
         then delete c from C
}
```

รูปที่ 2.17 apriori_gen1 procedure

```
procedure apriori_gen2 (L'k-1: Large itemsets)
{  C = null;
   for each l1 ∈ L'k-1
    for each l2 ∈ L'k-1
     if isInnerJoin (l1) and isInnerJoin (l2)
      then //make intradimension join
       {if (l1[1] = l2[1]) ∧ (l1[2] = l2[2]) ∧ ... ∧
        (l1[k-2] = l2[k-2]) ∧ (l1[k-1] < l2[k-1])
         then {
          c = l1  ⊳⊲ l2 ;
          InsertInto C;}
       }
       else //make interdimension join
        {if (l1[2]= l2[1]) ∧ (l1[3]= l2[2]) ∧ ... ∧
         (l1 [k-1 ] = l2[k-2]) ∧  l1[1 ] < l2[k-1])
    then {
     c = l1  ⊳⊲ l2 ;
     InsertInto C;
 }
for each c ∈ C
 for each (k-1)-subset s of c
  if  s  ∉ L'k-1
   then delete c from C
}
```

รูปที่ 2.18 apriori_gen2 procedure

จะเห็นได้ว่าอัลกอริทึม HDFUP ซึ่งเป็นการค้นหากฎความสัมพันธ์หลายมิติแบบมิติผสม สำหรับการเพิ่มขึ้นของข้อมูล มีการนำเอา Large itemsets ที่ได้จากค้นหาจากฐานข้อมูลเดิมมาใช้ประโยชน์เมื่อมีการเพิ่มขึ้นของข้อมูลเข้าสู่ฐานข้อมูล จึงสามารถลดจำนวน Candidate itemsets ที่เกิดจากฐานข้อมูลที่ถูกปรับปรุงเพื่อจะนำไปค้นหาในฐานข้อมูลเดิม ทำให้การค้นหาไอเท็มเซตใน

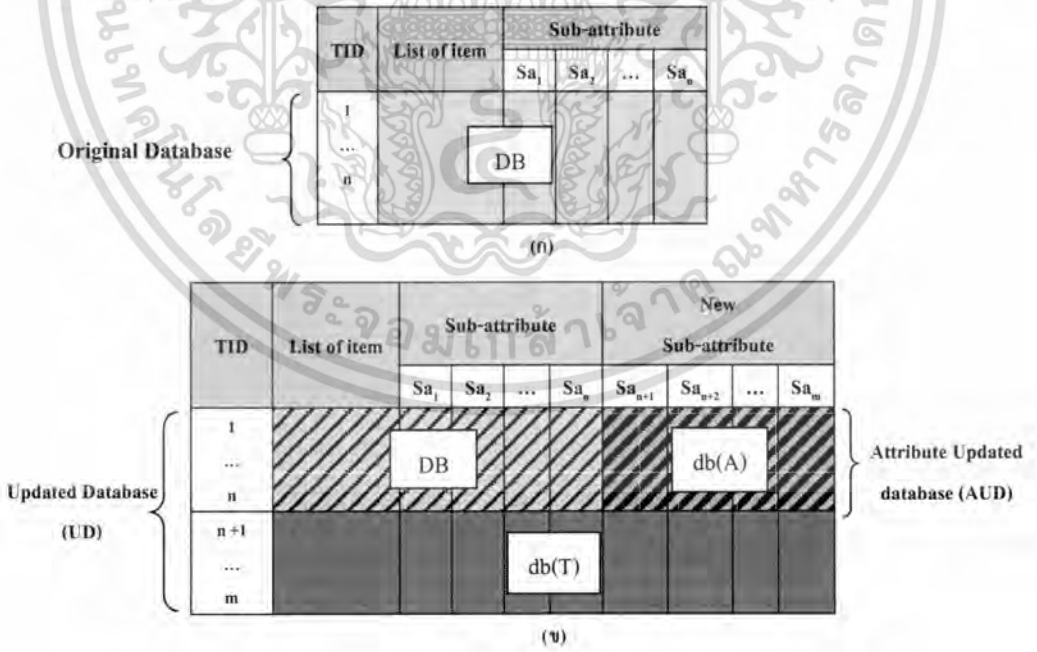
ฐานข้อมูลเดิมน้อยลง และอัลกอริทึม HDFUP นั้นยังสามารถสามารถทำการค้นหากฎความสัมพันธ์ที่เป็นแบบหลายมิติแบบมิติผสมได้ แต่เมื่อการเพิ่มขึ้นของข้อมูลเข้าสู่ฐานข้อมูลเป็นการเพิ่มขึ้นทั้งข้อมูลที่เป็นทรานแซคชั่นและข้อมูลแอททริบิวต์หรือข้อมูลมิติ อัลกอริทึม HDFUP ไม่สามารถที่จะค้นหากฎความสัมพันธ์ในกรณีนี้ได้

ดังนั้นงานวิจัยนี้เป็นการนำอัลกอริทึมที่ใช้ในการค้นหากฎความสัมพันธ์แบบมิติผสมและการค้นหากฎความสัมพันธ์สำหรับการเพิ่มขึ้นของข้อมูลในฐานข้อมูล ซึ่งการเพิ่มขึ้นของข้อมูลในฐานข้อมูลของงานวิจัยนี้เป็นการเพิ่มขึ้นของทรานแซคชั่นของฐานข้อมูลเดิม และการเพิ่มขึ้นของแอททริบิวต์ในฐานข้อมูลเดิมพร้อมกัน โดยนำเสนออัลกอริทึมสำหรับการเพิ่มขยายการค้นหากฎความสัมพันธ์แบบ 2 มิติ (An 2-Dimension Incremental Association rule discovery algorithm) เพื่อใช้ในการแก้ไขปัญหาในลักษณะดังที่กล่าวไป

บทที่ 3
อัลกอริทึมการค้นหากฎความสัมพันธ์แบบเพิ่มขยาย สำหรับการเพิ่มข้อมูลที่มีแอททริบิวต์ใหม่

ในบทนี้จะเป็นการกล่าวถึงอัลกอริทึมการค้นหากฎความสัมพันธ์แบบเพิ่มขยายสำหรับการเพิ่มข้อมูลที่มีแอททริบิวต์ใหม่ สมมุติฐานของการเพิ่มขยายการเพิ่มขึ้นสำหรับข้อมูลของอัลกอริทึมนี้เป็นการเพิ่มข้อมูลแอททริบิวต์ใหม่ที่เป็นแอททริบิวต์รอง และทรานแซคชั่นลงในฐานข้อมูลเดิมที่เป็นฐานข้อมูลแบบหลายมิติพร้อมกัน งานวิจัยนี้ได้นำแนวคิดของอัลกอลิทึม HDFUP มาใช้ในการปรับปรุงเพื่อให้สามารถค้นหากฎความสัมพันธ์แบบมิติผสมเมื่อมีการเพิ่มขึ้นของข้อมูลทรานแซคชั่นและแอททริบิวต์รองใหม่ในฐานข้อมูลที่มีทรานแซคชั่นข้อมูลแบบหลายมิติ จึงได้นำเสนออัลกอริทึมการค้นหากฎความสัมพันธ์แบบเพิ่มขยายสำหรับการเพิ่มข้อมูลที่มีแอททริบิวต์ใหม่ โดยอัลกอริทึมจะแบ่งการทำงานออกเป็น 3 ส่วน คือ คือ การค้นหา Large itemset ของแอททริบิวต์รองที่เพิ่มใหม่ การค้นหา Large itemset ของฐานข้อมูลเดิมกับแอททริบิวต์รองที่ใหม่บางส่วนที่เพิ่มขึ้น และการค้นหา Large itemset ทั้งหมดของฐานข้อมูลปรับปรุงใหม่ (Update database)

3.1 อัลกอริทึมการค้นหากฎความสัมพันธ์แบบเพิ่มขยายสำหรับการเพิ่มข้อมูลที่มีแอททริบิวต์ใหม่



รูปที่ 3.1 แสดงลักษณะฐานข้อมูลเดิม (ก) และ ฐานข้อมูลที่ถูกปรับปรุงใหม่ (ข)

การค้นหากฎความสัมพันธ์ของการเพิ่มขึ้นของข้อมูลในฐานข้อมูลเดิมในงานวิจัยที่กล่าวมาแล้วในบทที่ 2 เป็นเพียงการเพิ่มขึ้นของทรานแซคชั่นของฐานข้อมูลเดิม แต่อย่างไรก็ตามเมื่อในกรณีที่ไม่ใช้เฉพาะการเพิ่มขึ้นในฐานข้อมูลเป็นการเพิ่มขึ้นของข้อมูลทรานแซคชั่น แต่มีการเพิ่มขึ้นของแอททริบิวต์ใหม่เข้าสู่ฐานข้อมูลเดิมพร้อมกัน งานวิจัยดังกล่าวมาไม่สามารถที่จะแก้ปัญหาของการเกิดขึ้นในกรณีนี้ได้ ซึ่งต้องทำการค้นหากฎความสัมพันธ์ใหม่ทั้งหมดในฐานข้อมูลที่ปรับปรุง

ดังนั้นงานวิจัยนี้ได้นำ อัลกอลิทึม HDFUP ซึ่งเป็นอัลกอริทึมสำหรับการค้นหากฎความสัมพันธ์แบบมิติผสมเมื่อมีการเพิ่มขึ้นของข้อมูล สำหรับฐานข้อมูลทรานแซคชั่นแบบหลายมิติมาทำการปรับปรุงเพื่อให้สามารถค้นหากฎความสัมพันธ์แบบมิติผสมเมื่อมีการเพิ่มขึ้นของข้อมูลที่เป็นทั้งข้อมูลทรานแซคชั่น และแอททริบิวต์รอง ลงในฐานข้อมูลที่ทรานแซคชั่นข้อมูลแบบหลายมิติ

จากรูปที่ 3.1 แสดงลักษณะของฐานข้อมูลเดิมก่อนการปรับปรุง และ แสดงฐานข้อมูลที่ปรับปรุงใหม่ที่มีการเพิ่มขึ้นในส่วนของข้อมูลทรานแซคชั่น และแอททริบิวต์รอง จากภาพที่ 3.1 (ข) ส่วนของฐานข้อมูลที่เพิ่มขึ้น (Increment database) แบ่งออกเป็น 2 ส่วน คือ

1. แอททริบิวต์ที่เพิ่มขึ้น นั้นคือ db(A)
2. ทรานแซคชั้นที่เพิ่มขึ้น นั้นคือ db(T)

ความหมายของสัญลักษณ์ต่างๆที่ใช้ในอัลกอริทึม

DB หมายถึง ฐานข้อมูลเดิม (Original database)

AUD หมายถึง ฐานข้อมูลเดิมที่มีการเพิ่มแอททริบิวต์รองโดยจำนวนทรานแซคชั่นของแอททริบิวต์รองเท่ากับจำนวนทรานแซคชั่นของฐานข้อมูลเดิม (Attribute updated database)

D หมายถึง จำนวน transaction ที่มีอยู่ใน original database

d หมายถึง จำนวน transaction ที่มีอยู่ใน Increment transaction

UD หมายถึง จำนวน transaction ที่มีอยู่ใน Updated database

s หมายถึง ค่า minimum support

X หมายถึง ไอเท็มเซต X

$C_k^{db(A)}$ หมายถึง Candidate itemset ใน db(A) เมื่อ k=1, 2, …, k

C_k^{AUD} หมายถึง Candidate itemset ใน AUD เมื่อ k=1, 2, …, k

$C_k^{db(T)}$ หมายถึง Candidate itemset ใน db(T) เมื่อ k=1, 2, …, k

L_k^{DB} หมายถึง Large k-itemset ใน original database เมื่อ k=1, 2, …, k

$L_k^{db(A)}$ หมายถึง Large k-itemset ใน increment attribute เมื่อ k=1, 2, …, k

L_k^{AUD} หมายถึง Large k-itemset ใน AUD เมื่อ k=1, 2, …, k

L_k^{UD} หมายถึง Large k-itemset ใน updated database เมื่อ k=1, 2, …, k

$X.support_D$ หมายถึง ค่าความถี่ (support) ของไอเท็มเซต X ใน original database

$X.support_{UD}$ หมายถึง ค่าความถี่ (support) ของไอเท็มเซต X ใน updated Database

$X.support_d$ หมายถึง ค่าความถี่ (support) ของไอเท็มเซต X ใน db(T)

X.sup_mark หมายถึง ค่าความถี่ (support) ของไอเท็มเซต X ที่เป็นการประมาณค่าความถี่

3.1.1 วิธีการทำงานของอัลกอริทึมการค้นหากฎความสัมพันธ์แบบเพิ่มขยายสำหรับการเพิ่มข้อมูลแอททริบิวต์ใหม่

จากปัญหาดังกล่าว อัลกอริทึมการค้นหากฎความสัมพันธ์แบบเพิ่มขยายสำหรับการเพิ่มข้อมูลที่มีแอททริบิวต์ใหม่ แบ่งการทำงานออกเป็น 3 ส่วน คือ ส่วนแรกเป็นการหา Large itemset ในแอททริบิวต์รองที่เพิ่มขึ้น (หา Large itemset ใน db(A)) ส่วนที่ 2 เป็นการค้นหา Large itemset ของฐานข้อมูลเดิมเมื่อมีการเพิ่มแอททริบิวต์รอง (หา Large itemset ใน AUD) และส่วนที่ 3 เป็นการค้นหา Large itemset ทั้งหมดของฐานข้อมูลปรับปรุงใหม่ (หา Large itemset ใน UD)

3.1.1.1 การค้นหา Large itemset ใน db(A) หรือ ($L_k^{db(A)}$ เมื่อ k = 1, 2 …, k)

ขั้นตอนนี้จะเป็นการหา Large itemset ในส่วนของ db(A) อัลกอริทึมแสดงดังรูปที่ 3.2 ซึ่งในการหา Large itemset จะเป็นการใช้การเชื่อมความสัมพันธ์แบบ inter-dimension join เท่านั้นเพราะเป็นการหาความสัมพันธ์ของแอททริบิวต์ที่เป็นแอททริบิวต์รองที่เพิ่มขึ้นเท่านั้น โดยการทำงานมีดังนี้

1. รอบการค้นหา Large 1-itemset ใน db(T)

1.1 ทำการค้นหา เฉพาะใน db(A) เพื่อหา $C_1^{db(A)}$ และทำการนับค่าสนับสนุน แล้วนำค่าสนับสนุนของไอเท็มเซตที่ได้มาตรวจสอบว่า

- ถ้า $X.support_{db(A)} \geq s \times D$ แสดงว่าไอเท็มเซต X ที่ได้จากแอททริบิวต์รองที่เพิ่มนั้นสามารถเป็น $L_1^{db(A)}$ ได้ และจะได้ว่า ไอเท็มเซต $X \in L_1^{db(A)}$
- ถ้า $X.support_{db(A)} < s \times D$ แสดงว่าไอเท็มเซต X ที่ได้จากแอททริบิวต์รองที่เพิ่มนั้นไม่สามารถเป็น $L_1^{db(A)}$ ได้ จะทำการตัดไอเท็มเซต X ทิ้ง

2. รอบการค้นหา Large k - itemset ที่ $k \geq 2$ ใน db(A)

2.1 เป็นการหา Large k-itemset ที่ k = 2 โดยจะได้การ join กันระหว่าง $L_1^{db(A)}$ กับ $L_1^{db(A)}$ โดยใช้ Frequent_Gen Procedure แสดงดังรูปที่ 3.3 ซึ่งเป็นการ join แบบ inter-dimension join เท่านั้นเพื่อสร้าง $C_2^{db(A)}$

2.2 นำ $C_2^{db(A)}$ ที่ได้จากการ join ตรวจสอบว่าแต่ละไอเท็มเซตใน $C_2^{db(A)}$

แต่ละตัวมาซับเซตทั้งหมดเป็นสมาชิกใน $L_1^{db(A)}$ หรือไม่ ถ้าไอเท็มเซตที่พิจารณาไม่มีซับเซตทั้งหมดเป็นสมาชิกอยู่ใน L_1 จะทำการตัดไอเท็มเซตดังกล่าวออกจาก C_2 นำค่าสนับสนุนของไอเท็มเซตที่ได้มาตรวจสอบ

- ถ้า $X.support_{db(A)} \geq s \times D$ แสดงว่าไอเท็มเซต X ที่ได้จากแอททริบิวต์รองที่เพิ่มนั้นสามารถเป็น $L_2^{db(A)}$ ได้ และจะได้ว่า ไอเท็มเซต $X \in L_1^{db(A)}$
- ถ้า $X.support_{db(A)} < s \times D$ แสดงว่าไอเท็มเซต X ที่ได้จากแอททริบิวต์รองที่เพิ่มนั้นไม่สามารถเป็น $L_2^{db(A)}$ ได้ จะทำการตัดไอเท็มเซต X ทิ้ง

2.3 ทำการวนซ้ำเพื่อหา $L_k^{db(A)}$ ตั้งแต่ k ที่มีค่าตั้งแต่ 3 เป็นต้นไป โดยทำในขั้นตอนที่ 1.2.1 และ 1.2.2 จนกระทั่งไม่สามารถสร้าง $L_k^{db(A)}$ ได้

```
Phase 1
 Input: db(A)
 Output: L_k^db(A)
 1.  k = 1
 2.  L_1^db(A) = Large 1-itemset;
 3.  k = k+1
 4.  for (k = 2 : L_1^db(A) ≠ Ø : k++ ) do
 5.      C_k^db(A) = Frequent_Gen( L_k-1^db(A))
 6.      for each increment attribute (Sa_n+1,Sa_n+2,..,Sa_n) in db(A)
 7.          for all c (candidate) ∈ C_k^db(A)
 8.              c.count ++
 9.      L_k^db(A) = {c ∈ C_k^db(A) | c.count ≥ s × D}
10.  end
```

รูปที่ 3.2 แสดงอัลกอริทึมส่วนที่ 1 การค้นหา Large itemset ใน db(A)

```
Procedure : Frequent_Gen
 1.  for each l_1 ∈ L_k-1^db(A) do
 2.   for each l_2 ∈ L_k-1^db(A) do
 3.     if(k=2)
 4.        if ( l_1 and l_2 are not same sub-attribute)
 5.          then C_k^db(A)[k] = join l_1 ⋈ l_2
 6.     else
 7.        if (l_1[2] = l_2[1]) ∧ (l_1[3] = l_2[2]) ∧ ... ∧ (l_1[k-1] = l_2[k-2]) ∧ (l_1[1] < l_2[k-1])
 8.          then C_k^db(A)[k] = join l_1 ⋈ l_2
 9.     for each c ∈ C_k^db(A)[k]
10.        for each (k-1)-subset s of c
11.          if s ∉ L_k^AUD then delete c from C_k^db(A)[k]
12. end
13. return C_k^AUD[k]
```

รูปที่ 3.3 แสดง Frequent_Gen Procedure

3.1.1.2 การค้นหา Large itemset ใน AUD (L_k^{AUD} เมื่อ k=1, 2 …, k)

ขั้นตอนนี้เป็นการหาความสัมพันธ์ระหว่างไอเท็มเซตที่ได้จากฐานข้อมูลเดิมกับแอททริบิวต์รองที่เพิ่มขึ้นมาใหม่ นั้นคือการหา Large itemset ใน AUD หรือ L^{AUD} โดยจะนำเอา Large itemset ในส่วนของ db(A) จากการทำงานส่วนที่ 1 และการนำ Large itemset ที่ได้จากฐานข้อมูลเดิม มาใช้ประโยชน์ เพื่อลดการค้นหา Large itemset ในฐานข้อมูลเดิม ประกอบกับใช้การประมาณค่าสนับสนุนที่เป็นไปได้ที่ดีที่สุด เพื่อลดการค้นหาค่าสนับสนุนของไอเท็มเซตในฐานข้อมูลเดิม เพื่อทำการหา Large itemset ที่เป็นไปได้ทั้งหมดในส่วนของ AUD จากรูปที่ 3.4 อัลกอริทึมในส่วนนี้แบ่งออกเป็น 3 ขั้นตอนดังนี้

1. รอบการค้นหา Large 1-itemset ใน AUD

1.1 นำ $L_1^{db(A)}$ ที่ได้จากส่วนที่ 1 มาทำการรวมกับ L_1^{DB} จากฐานข้อมูลเดิมจะได้ เป็น L_1^{AUD} ในส่วนของ AUD นั้นคือ $L_1^{AUD} = L_1^{db(A)} \cup L_1^{DB}$

2. รอบการค้นหา Large k - itemset ที่ k = 2 ใน AUD

2.1 การทำงานในขั้นตอนนี้มีลักษณะเหมือนกันกับขั้นตอนที่ 1 คือจะมีการนำเอา Large itemset ของฐานข้อมูลเดิมเข้ามาใช้ แต่จะไม่มีการนำไอเท็มเซตที่ได้จากการหาความสัมพันธ์ ไปค้นหาในฐานข้อมูลสำหรับนับค่าสนับสนุนของแต่ละไอเท็มเซต แต่ค่าสนับสนุนของไอเท็มจะได้จากการการประมาณค่าความถี่ที่เป็นไปได้ที่ดีที่สุด (Frequency of false positives) ให้กับแต่ละไอเท็มเซต โดยลักษณะการทำงานของรอบการทำงานนี้มีดังนี้

2.2 ทำการ join กันระหว่าง L_1^{AUD} กับ L_1^{AUD} ที่ได้จากขั้นตอนที่ 2.1 โดยใช้ procedure Frequent_AppendAttribute1 ด้วยการ join แบบ inter-dimension join เท่านั้น แสดงดังรูปที่ 3.5 ซึ่งการ join ระหว่าง ไอเท็มเซตของ L_1^{AUD} ที่ได้จากแอททริบิวต์รองที่เพิ่มขึ้น กับไอเท็มเซตของ L_1^{AUD} ที่เป็นไอเท็มเซตของฐานข้อมูลเดิม เพื่อทำการสร้างเป็น C_2^{AUD}

จากการ join จะเห็นได้ว่าไม่มีการ join กันระหว่างไอเท็มเซตที่มาจากฐานข้อมูลเดิม เพราะ ถ้า join เพื่อหาความสัมพันธ์ของ Large itemset ที่มาจากฐานข้อมูลเดิม ความสัมพันธ์ที่ได้จากการ join นั้นเหมือนกับความสัมพันธ์ของไอเท็มเซตที่ได้ในฐานข้อมูลเดิม จึงสามารถนำ Large itemset ของฐานข้อมูลเดิมมาเป็นสมาชิกใน Large itemset ใน AUD ได้โดยที่ไม่ต้องทำการ join เพื่อนำไปค้นหาค่า support ของไอเท็มเซตซ้ำ

2.3 กำหนดค่าสนับสนุนให้แต่ละไอเท็มเซตใน C_2^{AUD} โดยค่าสนับสนุนจะได้จากการประมาณค่าสนับสนุนของการเกิดร่วมกันที่เป็นไปได้ดีที่สุดให้กับไอเท็มเซตนั้นๆ โดยไม่ต้องทำการค้นหาในฐานข้อมูลเดิม เช่น L_1^{AUD} ประกอบด้วย {a} = 1 และ $\{I_1\} = 3$ เมื่อทำการ join จะได้ $\{a, I_1\}$ ซึ่งค่าสนับสนุนของไอเท็ม $\{a, I_1\} = 1$ หมายถึง ค่าประมาณค่าสนับสนุนดีที่สุดที่เกิดขึ้นระหว่าง a และ I_1 เท่ากับ 1 และทำการระบุว่าไอเท็มเซตนั้นเป็นการประมาณค่าสนับสนุนโดยให้ X.sup_mark = true

2.4 นำ C_2^{AUD} ที่ได้จากขั้นตอนที่ 2.2.2 มารวมกับ $L_2^{db(A)}$ และ L_2^{DB} จะได้ เป็น L_2^{AUD} นั้นคือ $L_2^{AUD} = L_2^{db(A)} \cup C_2^{AUD} \cup L_2^{DB}$ ตามลำดับ

3. รอบการค้นหา Large k-itemset ที่ $k \geq 3$ ใน AUD

ลักษณะการทำงานของขั้นตอนนี้คล้ายกับขั้นตอนที่ 2.2 แต่จะแตกต่างกันตรงเงื่อนไขการ join เพื่อหาความสัมพันธ์ระหว่างไอเท็มเซต โดยใช้ Frequent_AppendAttribute2 procedure เพื่อทำการหา C_k^{AUD} แสดงดังรูปที่ 3.6 โดยมีขั้นตอนการทำงานดังนี้

3.1 เป็นการหา Large k-itemset ที่ $k \geq 3$ โดยจะได้การ join กันระหว่าง L_{k-1}^{AUD} กับ L_{k-1}^{AUD} รอบก่อนหน้า ด้วยการ join เฉพาะแบบ inter-dimension join ซึ่งการพิจารณาการ join ดังนี้

- การ join ระหว่าง ไอเท็มเซตของ L_{k-1}^{AUD} ที่มีไอเท็มมาจากแอททริบิวต์รองที่เพิ่มขึ้น กับ ไอเท็มเซตของ L_{k-1}^{AUD} ที่มีไอเท็มมาจากแอททริบิวต์รองที่เพิ่มขึ้น
- การ join ระหว่าง ไอเท็มเซตของ L_{k-1}^{AUD} ที่มีไอเท็มมาจากแอททริบิวต์รองที่เพิ่มขึ้น กับ ไอเท็มเซตของ L_{k-1}^{AUD} ที่เป็นไอเท็มเซตของฐานข้อมูลเดิม

3.2 กำหนดค่าสนับสนุนขอแต่ละไอเท็มเซตในขั้นตอนนี้จะได้จากการประมาณค่าสนับสนุนของการเกิดร่วมกันดีที่สุดที่เป็นไปได้ ให้กับไอเท็มเซตนั้นๆ โดยไม่ต้องทำการค้นหาในฐานข้อมูลเดิม เช่น L_2^{AUD} ประกอบด้วย $\{a,I_1\} = 4$ และ $\{I_1,I_3\} = 2$ เมื่อทำการ join จะได้ $\{a, I_1,I_3\}$ ซึ่งค่าสนับสนุนของไอเท็ม $\{a, I_1,I_3\} = 2$ และทำการกำหนดค่า X.sup_mark = true ให้กับไอเท็มเซตเพื่อเป็นการบอกว่าไอเท็มเซตมีค่าสนับสนุนที่มาจากการประมาณค่า

3.3 นำ $L_k^{db(A)}$ ที่ได้จากขั้นตอนส่วนที่ 1 มารวมกับ C_k^{AUD} และ L_k^{DB} จากฐานข้อมูลเดิม เพื่อสร้างเป็น L_k^{AUD} นั้นคือ $L_k^{AUD} = L_k^{db(A)} \cup C_k^{AUD} \cup L_k^{DB}$ ตามลำดับ

3.4 ทำการวนซ้ำการทำงานจนกว่าไม่สามารถหา L_k^{AUD} ได้เมื่อเสร็จจากขั้นตอนนี้จะได้ Large itemset ของความสัมพันธ์ระหว่างฐานข้อมูลเดิมกับแอททริบิวต์รองที่เพิ่มขึ้นใน AUD นั้นคือ L_k^{AUD}

หลังจากนั้นจะนำ L_k^{AUD} ที่ได้ไปใช้ในการทำงานส่วนที่ 3 ต่อไปเพื่อใช้สำหรับการค้นหา Large itemset ทั้งหมดในฐานข้อมูลปรับปรุง

Phase 2

Input: $L_k^{db(A)}$
Output: L_k^{AUD}

11. k = 1
12. $L_1^{AUD} = L_1^{db(A)} \cup L_1^{DB}$
13. k = k+1
14. **for** (k = 2 : $L_1^{AUD} \neq \emptyset$: k++) do
15. **if** k = 2
16. then C_2^{AUD} = Frequent_AppendAttribute1(L_1^{AUD})
17. **else**
18. C_k^{AUD} = Frequent_AppendAttribute2(L_{k-1}^{AUD})
19. $L_k^{AUD} = L_k^{db(A)} \cup C_k^{AUD} \cup L_k^{DB}$
20. **end**

รูปที่ 3.4 แสดงอัลกอริทึมส่วนการค้นหา L_k^{AUD} ใน AUD

Procedure : Frequent_AppendAttribute1

1. i = 1
2. **for** each $l_1[i] \in L_1^{AUD}$ do
3. **for** each $l_2[i+1] \in L_1^{AUD}$ do
4. **if** itemset of l_1 from increment attribute and itemset of l_2 from DB
5. then $C_2^{AUD}[k]$ = join $l_1 \bowtie l_2$
6. **if** $l_1.support_{AUD} <= l_2.support_{AUD}$
7. then $C_2^{AUD}[k].support_{AUD} = l_1.support_{AUD}$
8. **else** $C_2^{AUD}[k].support_{AUD} = l_2.support_{AUD}$
9. **end**
10. $C_2^{AUD}[k].sup\_mark$ = true
11. **end**
12. **end**
13. **return** C_2^{AUD}

รูปที่ 3.5 แสดง Frequent_AppendAttribute1 Procedure

Procedure : Frequent_AppendAttribute2

```
1.  i = 1
2.  for each l1∈ Lk^AUD do
3.      for each l2 ∈ Lk^AUD do
4.          if ( l1 and l2 are itemset from increment attribute) or ( l1 is itemset from
               increment attribute and l2 is original itemset from DB)
5.          if (l1[2] = l2[1]) ∧ (l1[3] = l2[2]) ∧ … ∧ (l1[k-1] = l2[k-2]) ∧
               (l1[1] < l2[k-1])
6.              then Ck^AUD[k] = join l1 ⋈ l2
7.               if l1.supportAUD <= l2.supportAUD then
8.                 Ck^AUD[k].supportAUD = l1.supportAUD
9.              else
10.                Ck^AUD[k].supportAUD = l2.supportAUD
11.         end
12.         Ck^AUD[k].sup_mark = true
13.     end
14. end
15. return Ck^AUD[k]
```

รูปที่ 3.6 แสดง Frequent_AppendAttribute2 procedure

3.1.1.3 การค้นหา Large k - itemset ของ UD เมื่อ k=1, 2 …, k

ขั้นตอนการทำงานของอัลกอริทึมในส่วนนี้ จะแบ่งการทำงานหลักออกเป็น 3 ขั้นตอน คือ การค้นหา L_1^{UD}, การค้นหา L_k^{UD} โดยที่ k = 2 และ การค้นหา L_k^{UD} โดยที่ k ≥ 3 โดยขั้นตอนทั้งสามส่วนอธิบายดังนี้

1. รอบการค้นหา Large 1-itemset ใน UD

ดังรูปที่ 3.7 แสดงการทำงานโดยจะมีลักษณะการทำงานคล้ายกับอัลกอริทึม HDFUP ซึ่งเป็นการตัดทิ้ง ไอเท็มที่เป็น loser item และหาไอเท็มที่เป็น winner item ที่เป็น Large 1-itemset ในฐานข้อมูลที่ปรับปรุง

1.1 ทำการค้นหาใน db(T) เพื่อทำการปรับปรุงค่าสนับสนุนของไอเท็ม และเพื่อหาไอเท็มที่เป็น winner item และ ตัดทิ้งไอเท็มที่เป็น loser item โดยมีหลักการพิจารณาดังนี้

- กรณี ถ้า $X \in L_1^{AUD}$

นำค่าสนับสนุนของไอเท็มเซต X ใน AUD และ db(T) มารวมกันได้เป็นค่าสนับสนุนของไอเท็ม X นั้นใน UD นั้นคือ $X.support_{UD} = X.support_{AUD} +$

$X.support_d$ แล้วทำการตรวจสอบค่า สนับสนุนที่ได้ว่าผ่านค่าสนับสนุนขั้นต่ำของฐานข้อมูลปรับปรุง $s \times (AUD + d)$ หรือไม่ โดยตรวจสอบว่า

- ถ้า $X.support_{UD} \geq s \times (AUD + d)$ แสดงว่า ไอเท็มเซต X สามารถเป็น L_1^{UD} ในฐานข้อมูลที่ปรับปรุงได้ จะได้ว่า $X \in L_1^{UD}$ และเรียกไอเท็มเซต X นั้นว่า winner item

- ถ้า $X.support_{UD} < s \times (AUD + d)$ แสดงว่า ไอเท็มเซต X ไม่สามารถเป็น L_1^{UD} ในฐานข้อมูลที่ปรับปรุงได้ เรียกไอเท็มเซต X นั้นว่า loser item แล้วจะทำการตัด (prune) ทิ้ง

- กรณีถ้า $X \notin L_1^{AUD}$ จะมีการพิจารณา 2 ส่วน คือ
  - พิจารณาเพื่อทำการตัดทิ้งไอเท็มที่ไม่มีโอกาสเป็น L_1^{UD} โดยตรวจสอบจาก ถ้า $X \notin L_1^{AUD}$ และ $X.support_d < (s \times d)$ แสดงว่า ไอเท็มนั้นเป็น lose item หมายถึงไม่สามารถเป็นเกิดขึ้นใน AUD ได้ แล้วทำการตัดไอเท็ม X นั้นทิ้ง ในส่วนนี้จะเป็นการช่วยในการลดจำนวนการค้นหาไอเท็มใน AUD
  - พิจารณาไอเท็มที่มีโอกาสเป็น L_1^{UD} โดยตรวจสอบจาก ถ้า $X \notin L_1^{AUD}$ และ $X.support_d \geq (s \times d)$ นำ ไอเท็ม X ไปค้นหาค่าสนับสนุนในส่วน AUD เพื่อหาค่า $X.support_{UD}$ แล้วนำค่าที่ได้มาตรวจสอบ
    - ถ้า $X.support_{UD} \geq s \times (AUD+d)$ แสดงว่า ไอเท็มเซต X เป็น winner item โดยจะได้ว่า $X \in L_1^{UD}$
    - ถ้า $X.support_{UD} < s \times (AUD+d)$ แสดงว่า ไอเท็ม X เป็น Lose item และทำการลบไอเท็ม X ทิ้ง

เมื่อเสร็จการทำงานในขั้นตอนนี้จะได้ Large 1- itemset ในฐานข้อมูลที่ถูกปรับปรุงแล้ว (L_1^{UD})

2. รอบการค้นหา Large 2-itemset ใน UD

การทำงานในส่วนนี้จะเป็นการหา Large 2-itemset แสดงดังรูปที่ 3.8 โดยทำการหา itemset ที่ไม่สามารถเป็น L_2^{UD} ได้ก่อน เพื่อลดการค้นหาใน db(T) โดยใช้แนวคิดที่ว่า "*ถ้าไอเท็มใดๆ ที่เป็น loser item ในการทำงานรอบก่อนหน้าแล้ว itemset ใดๆ ของ Large itemset ในฐานข้อมูลเดิมที่มีไอเท็มดังกล่าวเป็นซับเซตอยู่จะไม่สามารถเป็น winner item ในรอบนั้นได้*" จากแนวคิดดังกล่าวจะมีการทำงานดังนี้

2.1 หา new candidate 2-itemset

ในขั้นตอนนี้จะเป็นการ join ระหว่าง L_1^{UD} กับ L_1^{UD} ตาม procedure hybrid_gen1 ดังรูปที่ 3.9 เพื่อสร้าง candidate 2-itemset โดยมีข้อกำหนดในการ join คือ ไม่สามารถ join กันด้วยแอททริบิวต์รองที่มาจากแอททริบิวต์เดียวกันได้ เช่น แอททริบิวต์อายุ ซึ่ง

เป็นแอททริบิวต์รองไม่สามารถ join กับไอเท็มของแอททริบิวต์อายุที่อยู่ภายในแอททริบิวต์เดียวกันได้ โดยพิจารณาดังนี้

○ **กรณี** $X \in C_2^{db(T)}$ **และ** $X \in L_2^{AUD}$ ทำการตัดไอเท็ม X ออกจาก $C_2^{db(T)}$ เพราะ ไอเท็ม X เป็นสมาชิกอยู่ใน L_2^{AUD} เดิมแล้ว

○ **กรณี** $X \in C_2^{db(T)}$ **และ** $X \notin L_2^{AUD}$ นำเอาไอเท็ม X ดังกล่าวไปทำการค้นหาใน db(T) แล้วทำการเพื่อปรับปรุงค่า สนับสนุนแล้วนำมาพิจารณาว่า

- ถ้า $X.support_d < (s \times d)$ ให้ลบไอเท็ม X ออกจาก $C_2^{db(T)}$ เพื่อไม่ต้องนำไปค้นหาต่อใน AUD
- ถ้า $X.support_d \geq (s \times d)$ ให้จะเก็บไอเท็ม X ใน $C_2^{db(T)}$ เพื่อไปค้นหาต่อใน AUD

2.3 หา itemset ที่เป็นทั้งสมาชิกของ L_2^{AUD} และ L_2^{UD} คือเป็นการหา itemset ที่เป็น large itemset ทั้งใน AUD และ UD

ขั้นตอนนี้จะตัดทิ้ง loser item ของ L_2^{AUD} ที่ไม่สามารถเป็น L_2^{UD} ก่อนเพื่อลดการค้นหาใน db(T) โดยพิจารณาจาก $Y = L_1^{AUD} - L_1^{UD}$ หมายถึง ไอเท็มเซต Y ใดๆที่เป็นสมาชิกของ L_1^{AUD} แต่ไม่เป็นสมาชิกของ L_1^{UD} นั้นคือเมื่อ $X \in L_2^{AUD}$ ที่มีไอเท็มเซต Y เป็นซับเซต จะไม่สามารถเป็น large itemset ได้ ดังนั้นไอเท็มเซต X ดังกล่าวจะถูกตัดทิ้งไป เช่น

$L_1^{AUD} = \{a\},\{I_1\},\{I_2\}$ $L_1^{UD} = \{a\},\{I_1\},\{I_3\}$ $L_2^{AUD} = \{a\ I_1\}, \{a\ I_2\}$

จะได้ว่า $Y = L_1^{AUD} - L_1^{UD} = \{I_2\}$

ดังนั้น จะเหลือ $L_2^{AUD} = \{a\ I_1\}$

นำไอเท็มเซต ใน L_2^{AUD} ที่ได้ทำการตัด loser itemsetมาทำการค้นหาใน db(T) เพื่อปรับปรุงค่าสนับสนุนแล้วนำมาตรวจสอบว่า

● ถ้า $X.support_{UD} \geq s \times (AUD + d)$ ไอเท็มเซตนั้นๆจะนำมาตรวจสอบต่อว่า

- ถ้า X.sup_mark = true คือ ไอเท็มที่ค่า support ได้จากการประมาณ แล้วทำการกำหนดค่าสนับสนุนของ item เท่ากับ $X.support_d$ (คือค่าสนับสนุนที่ได้จากการค้นหาใน db(T)) และนำเอาไอเท็มเซตนั้นไปเพิ่มใน $C_2^{db(T)}$ เพื่อทำการค้นหาต่อใน AUD

ถ้า X.sup_mark = false แสดงว่าไอเท็มเซตนั้นเป็น winner itemset แล้วจะถูกเก็บใน L_2^{UD}

- ถ้า $X.support_{UD} < s \times (AUD+d)$ itemset นั้นๆจะเป็น loser itemset แล้วจะถูกตัดทิ้ง

2.4 หา itemset ใน $C_2^{db(T)}$ ที่สามารถเป็น L_2^{UD} ได้

ขั้นตอนนี้เป็นการนำเอาไอเท็มเซต X แต่ละตัวใน $C_2^{db(T)}$ ซึ่งไอเท็มเซตในส่วนนี้จะประกอบด้วยไอเท็มเซตที่มาจากขั้นตอนที่ 3.2.1 และ 3.2.2 ไปค้นหาต่อใน AUD เพื่อทำการปรับปรุงค่าสนับสนุนโดยทำการตรวจสอบต่อว่า

- ถ้า $X.support_{UP} \geq s \times (AUD + d)$ ให้เพิ่มไอเท็มเซต X เป็นสมาชิกของ L_2^{UD}
- ถ้า $X.support_{UP} < s \times (AUD + d)$ ให้ตัดไอเท็มเซต X ทิ้ง

เมื่อเสร็จสิ้นในขั้นตอนนี้แล้วผลลัพธ์ที่ได้คือ Large 2-itemset (L_2^{UD}) ในฐานข้อมูลที่ถูกปรับปรุงแล้ว

3. การทำงานรอบที่ k ตั้งแต่ $k \geq 3$

ขั้นตอนนี้จะมีลักษณะการทำงานเหมือนกับขั้นตอนที่ 3.2 แต่จะแตกต่างกันตรงการ join โดยจะใช้ procedure hybrid_gen2 ดังรูปที่ 3.10 ในการ join ซึ่งข้อกำหนดในการพิจารณาการ join ของไอเท็มเซต มี 2 กรณี คือ

- ถ้าเป็นการ join กันระหว่างไอเท็มเซตที่มีไอเท็มทั้งหมดมาจากแอททริบิวต์หลักทั้งหมด การ join จะเป็นแบบ intra-dimension join
- แต่หากเป็นรูปแบบอื่นๆให้ทำการ join แบบ inter-dimension join โดยหลังการ join แต่ละรูปแบบใน procedure hybrid_gen2

จะนำ candidate k- itemset ($C_k^{db(T)}$) ที่ได้ มาทำการตรวจสอบซับเซตว่าแต่ละซับเซตของ candidate itemset นั้นๆทั้งหมด ปรากฏอยู่ใน L_{k-1}^{UD} หรือไม่ ถ้ามีบางซับเซตของ $C_k^{db(T)}$ ไม่ปรากฏอยู่ใน L_{k-1}^{UD} จะทำการตัดไอเท็มเซตนั้นออกจาก $C_k^{db(T)}$ เพราะว่าไอเท็มเซตดังกล่าวไม่สามารถเป็น Large k-itemsets ได้

ทำการวนรอบซ้ำ เพื่อหา k-itemset ($k > 3$) ต่อไปจนไม่สามารถทำการหา L_k^{UD} ต่อได้ เมื่อเสร็จจากขั้นตอนนี้เราจะได้ Large itemset ทั้งหมดในฐานข้อมูลปรับปรุงนั้นคือ L_k^{UD}

```
Phase 3
Input: db(T), AUD, s, L_k^AUD
Output: L_k^UD: the set of all Large itemset in Updated database
1.  k = 1 /* The first iteration */
2.  if k = 1
3.     for each transaction ∈ db(T)
4.        for all 1- itemset ∈ L_1^AUD
5.           if X ∈ L_1^AUD then X.support_d ++;
6.           else if X ∉ C_1^db(T)
7.                then C_1^db(T) = C_1^db(T) ∪ {X}  X.support_d = 0
8.                     /* intitial value*/
9.                     X.support_d ++
10.          end
11.    end
12.    for all X ∈ L_1^AUD do
13.       if X.support_UD ≥ s × (AUD + d) then insert X into L_1^UD
14.    end
15.    for all X ∈ C_1^db(T) do
16.       if X.support_d ≤ s × d  then delete X from C_1^db(T)
17.    end
18.    for each attribute in each transaction ∈ AUD do
19.       for all item X ∈ C_1^db(T)
20.          if X ∈ item in attribute
21.             then X.support_AUD ++
22.    end
23.    for all X ∈ C_1^db(T) do
24.       if X.support_UD ≥ s × (AUD + d)
25.          then insert X into L_1^UD
26.    end
27. end /*end of Large 1-itemset */
```

รูปที่ 3.7 แสดงการทำงานในส่วนที่ 3 ในรอบที่ 1

```
/* The k-th iteration : for k ≥ 2 , */
 k = k + 1
 for (k = 2 : L_k^UD ≠ ∅ : k++ ) do
      if k = 2
        then C_2^db(T) = hybrid_gen1 (L_1^UD) – L_2^AUD;
      else
        C_k^db(T) = hybrid_gen2 (L_k-1^UD) – L_k^AUD;
      for all k-itmeset X ∈ L_k^AUD do/*prune L_k^AUD have subset in L_k-1^AUD-L_k-1^UD*/
          for all Y | Y = (k -1) itemset ∈ (L_k-1^AUD - L_k-1^UD) do
              if Y ⊆ X in then L_k^AUD { prune X from L_k^AUD ; break; }
      end
      for each transaction in db(T) do /* finding support count in db(T)*/
          for all X ∈ L_k^AUD do
              X.support_d ++
          end
          for all X ∈ C_k^db(T) do
              X.support_d ++
          end
      end
      for all X ∈ C_k^db(T) do
          if X.support_d < (s × d) then prune X from C_k^db(T)
      end
      for all X ∈ L_k^AUD do
          if X.support_UD ≥ s × (AUD+d) and X is itemset from increment attribute
             then assign support of X equal X.support_d /* support in db(T)*/
                  insert X into C_k^db(T)
          else insert X into L_k^UD
      end
      for each transaction ∈ AUD do /* search itemset in AUD */
          for all X ∈ Subset(C_k^db(T),T) do
              X.support_AUD ++
      end
      for all X ∈ C_k^db(T) do
          if X.support_UD ≥ s × (AUD + d)then insert X into L_k^UD
       end
end /* end of the k-th iteration : for k ≥ 2 */
```

รูปที่ 3.8 แสดงการทำงานในส่วนที่ 3 ในรอบที่ $k \geq 2$

```
Procedure : hybrid_gen1
1.  C2^db(T)[k] = ∅
2.  for each l1 ∈ L1^UD do
3.   for each l2 ∈ L1^UD {
4.        if itemset of l1 and l2 are main attribute
5.          then C2^db(T)[k] = join  l1 ⋈ l2 ; /* Intra-join */
6.        else { if l1 and l2 are not main attribute and not from same attribute
7.          then C2^db(T)[k] =   join l1 ⋈ l2 /* Inter-join */
8.        end
9.  end
10. for each C ∈ C2^db(T)[k] do
11.     for each s | s = (k-1) -subset of C
12.         if s ∉ L1^UD
13.           then delete c from C2^db(T)[k]
14.         end
15. end
16. return C2^db(T)[k]
```

รูปที่ 3.9 แสดง hybrid_gen1 procedure

```
Procedure : hybrid_gen2
1.  Ck^db(T)[k] = ∅
2.  for each l1 ∈ Lk-1^UD do
3.      for each l2 ∈ Lk-1^UD {
4.          if all item in l1 and l2 are main attribute then
5.            if (l1[1]= l2[1]) ∧ (l1[2]=l2[2]) ∧ ... ∧(l1[k-1] = l2[k-2])∧(l1[k-1]<l2[k-1])
6.              then Ck^db(T)[k] = join  l1 ⋈ l2;  /* Intra-join */
7.            else  if (l1[2]= l2[1])∧(l1[3]=l2[2])∧ ...∧(l1[k-1] = l2[k-2])∧(l1[1]<l2[k-1])
8.                  then Ck^db(T)[k] = join l1 ⋈ l2 ;} /* Inter-join */
9.            end
10. end
11. for each c ∈ Ck^db(T)[k] do
12.     for each s | s = (k-1) -subset of c {
13.         if s ∉ Lk-1^UD
14.           then delete c from Ck^db(T)[k] }
15.         end
16. end
17. return Ck^db(T)[k];
```

รูปที่ 3.10 แสดง hybrid_gen2 procedure

TID	Age	Order ID
1	a	I_2, I_4
2	a	I_1, I_2, I_5
3	a	I_2, I_3
4	b	I_1, I_3
5	b	I_1, I_2, I_4
6	a	I_2, I_3
7	b	I_1, I_3

(ก)

ฐานข้อมูลเดิม

TID	Area	Age	Order ID
1	1	a	I_2, I_4
2	1	a	I_1, I_2, I_5
3	2	a	I_2, I_3
4	2	b	I_1, I_3
5	2	b	I_1, I_2, I_4
6	1	a	I_2, I_3
7	2	b	I_1, I_3
8	3	a	I_1, I_5, I_6
9	3	b	I_1, I_2, I_5, I_6, I_7
10	3	c	I_2, I_5
11	1	c	I_2, I_6
12	1	b	I_5, I_6
13	2	b	I_1, I_5, I_6

(ข)

ฐานข้อมูลปรับปรุง

รูปที่ 3. 11 แสดงตัวอย่างฐานข้อมูลเดิม และ ฐานข้อมูลปรับปรุง

3.1.2 ตัวอย่างอัลกอริทึมการค้นหากฎความสัมพันธ์แบบเพิ่มขยายสำหรับการเพิ่มข้อมูลที่มีแอททริบิวต์ใหม่

จากรูปที่ 3.11 กำหนดให้รายการซื้อขายสินค้าในทรานแซคชั่นฐานข้อมูลเดิมมีการเก็บรายละเอียดเกี่ยวกับอายุ และสินค้า ดังรูปที่ 3.11 (ก) ซึ่งเป็นการเก็บข้อมูลภายในทรานแซคชั่นในฐานข้อมูลแบบหลายมิติ เมื่อมีการเพิ่มข้อมูลพื้นที่และเพิ่มข้อมูลการทรานแซคชั่นในฐานข้อมูลเดิมดังรูปที่ 3.11 (ข) โดยกำหนดให้ ค่าสนับสนุนขั้นต่ำเท่ากับ 20 เปอร์เซ็นต์ และมีการเก็บ Large itemset (L_k^{DB}) ดังรูปที่ 3.10 ของฐานข้อมูลเดิม การทำงานของอัลกอริทึมสำหรับการเพิ่มขยายการค้นหากฎความสัมพันธ์แบบ 2 มิติเมื่อทำกับฐานข้อมูลข้อมูลตัวอย่างจะมีการทำงานดังนี้

L_1^{DB}	
Item	support
{a}	4
{b}	3
$\{I_1\}$	4
$\{I_2\}$	5
$\{I_3\}$	4
$\{I_4\}$	2

L_2^{DB}	
Item	support
$\{a, I_2\}$	4
$\{a, I_3\}$	2
$\{b, I_1\}$	3
$\{b, I_3\}$	2
$\{I_1, I_2\}$	2
$\{I_1, I_3\}$	2
$\{I_2, I_3\}$	2
$\{I_2, I_4\}$	2

L_3^{DB}	
Item	support
$\{a, I_2, I_3\}$	2
$\{b, I_1, I_3\}$	2

รูปที่ 3.12 แสดง Large itemsets ของฐานข้อมูลเดิม

$C_1^{db(A)}$	
item	Scan db(A) support
{1}	3
{2}	4

(ก)

$X.support_{AUD} \geq s \times D$

$L_1^{db(A)}$	
item	Scan db(A) support
{1}	3
{2}	4

(ข)

$L_1^{db(A)}$ join $L_1^{db(A)}$

$L_2^{db(A)}$
item
∅

(ค)

รูปที่ 3.13 แสดงขั้นตอน การหา Large itemset ใน AUD

1. ส่วนที่ 1 การค้นหา Larege itemset ใน db(A)

1.1 ขั้นตอนการหา $L_1^{db(A)}$

ทำการค้นหา Candidate 1-itemset โดยการค้นหาในส่วน db(A) นั้นคือแอททริบิวต์ Area ซึ่งเป็นแอททริบิวต์รองที่เพิ่มขึ้น ผลลัพธที่ได้แสดงดังรูป 3.13(ก) แล้วนำมาตรวจสอบว่า $X.support_{AUD} \geq s \times D$ แล้วนำไอเท็มที่ผ่านการพิจารณาไปเก็บใน $L_1^{db(A)}$

1.2 ขั้นตอนการหา $L_k^{db(A}$ ที่ $k \geq 3$

ทำการ join แบบ inter dimension join ระหว่าง $L_1^{db(A)}$ กับ $L_1^{db(A)}$ โดยผลที่ได้จากการ join คือเซตว่างแสดงดังรูปที่ 3.13(ค) เป็นเพราะว่าไอเท็มใน $L_1^{db(A)}$ มาจากแอททริบิวต์รองเดียวกัน จึงไม่สามารถ join กันได้ จึงทำให้เสร็จสิ้นในการทำงานส่วนที่ 1

2. ส่วนที่ 2 การค้นหา Larege itemset ใน AUD

2.1 ขั้นตอนการหา L_1^{AUD}

นำ $L_1^{db(A)}$ ที่ได้จากส่วนที่ 1 ดังรูปที่ 3.14(ก) และ L_1^{DB} ของฐานข้อมูลเดิม รูปที่ 3.14(ข) มารวมกันแล้วนำไอเท็มเซตที่ได้ไปเก็บใน L_1^{AUD} แสดงดังรูปที่ 3.14(ค)

$L_1^{db(A)}$	
item	support
{1}	3
{2}	4

(ก)

L_1^{DB}	
item	support
{a}	4
{b}	3
$\{I_1\}$	4
$\{I_2\}$	5
$\{I_3\}$	4
$\{I_4\}$	2

(ข)

L_1^{AUD}	
item	support
{1}	3
{2}	4
{a}	4
{b}	3
$\{I_1\}$	4
$\{I_2\}$	5
$\{I_3\}$	4
$\{I_4\}$	2

(ค)

$L_1^{db(A)} \cup L_1^{DB}$

รูปที่ 3.14 แสดงขั้นตอนการหา L_1^{AUD}

C_2^{AUD}		
item	Estimate support	mark
{1, a}	3	*
{1,b}	3	*
{1,I_1}	3	*
{1,I_2}	3	*
{1,I_3}	3	*
{1,I_4}	2	*
{2, a}	4	*
{2, b}	3	*
{2, I_1}	4	*
{2, I_2}	4	*
{2, I_3}	4	*
{2, I_4}	2	*

(ก)

L_2^{DB}	
item	support
{a, I_2}	4
{a, I_3}	2
{b, I_1}	3
{b, I_3}	2
{I_1, I_2}	2
{I_1, I_3}	2
{I_2, I_3}	2
{I_2, I_4}	2

(ข)

L_2^{AUD}		
item	support	mark
{1,a}	3	*
{1,b}	3	*
{1,I_1}	3	*
{1,I_2}	3	*
{1,I_3}	3	*
{1,I_4}	2	*
{2, a}	4	*
{2, b}	3	*
{2,I_1}	4	*
{2,I_2}	4	*
{2,I_3}	4	*
{2,I_4}	2	*
{a, I_2}	4	
{a, I_3}	2	
{b, I_1}	3	
{b, I_3}	2	
{I_1,I_2}	2	
{I_1,I_3}	2	
{I_2,I_3}	2	
{I_2,I_4}	2	

(ค)

$L_2^{db(A)} \cup C_2^{AUD} \cup L_2^{DB}$

รูปที่ 3.15 แสดงขั้นตอน การหา L_2^{AUD}

2.2 ขั้นตอนการหา L_2^{AUD}

จากรูปที่ 3.15 เป็นการหา L_2^{AUD} โดยทำการ join กันระหว่าง L_1^{AUD} กับ L_1^{AUD} ผลลัพธ์ได้เป็น C_2^{AUD} ซึ่งการ join จะทำเฉพาะระหว่างไอเท็มเซตที่มาจากแอททริบิวต์รองที่เพิ่มใหม่นั้นคือ แอททริบิวต์ Area กับ แอททริบิวต์ Age เช่น {1} กับ {a} และ ระหว่างแอททริบิวต์ Area กับ แอททริบิวต์ Order ID เช่น {1} กับ {I_1} เท่านั้น แต่ไม่สามารถ join กันระหว่างไอเท็มเซตที่มากจากฐานข้อมูลเดิม

ส่วนค่าสนับสนุนจะไม่มีการค้นหาในฐานข้อมูล แต่จะทำการประมาณค่าสนับสนุนให้กับไอเท็มเซต ดังรูปที่ 3.15 (ก) เช่น ไอเท็มเซต {1,a} มาจากการ join ระหว่าง {1} ที่มีค่าสนับสนุนเท่ากับ 3 และ {a} ที่มีค่าสนับสนุนเท่ากับ 4 ดังนั้น {1,a} ค่าสนับสนุนที่เป็นไปได้ที่ดีที่สุดจะเท่ากับ 3 แล้วทำการระบุ (mark) ให้กับไอเท็มเพื่อบอกว่าเป็น ไอเท็มเซตที่ได้จากการประมาณค่า จากนั้นนำไอเท็มที่ได้ รวมกับ Large itemset ของ $L_2^{db(A)}$ และL_2^{DB} ของฐานข้อมูลเดิม แล้วเก็บใน L_2^{AUD} แสดงดังรูปที่ 3.15(ค) แต่ในตัวอย่างนี้ $L_2^{db(A)}$ เป็นเซตว่างทำให้ L_2^{AUD} ประกอบด้วย C_2^{AUD} และ L_2^{DB}

2.3 ขั้นตอนการหา L_k^{AUD} ที่ $k \geq 3$

เช่นเดียวกัน จากรูปที่ 3.16 และ 3.17 มีการทำงานลักษณะคล้ายกับขั้นตอนก่อนหน้า เป็นการหา L_3^{AUD} และ L_4^{AUD} ตามลำดับ โดยทำการ join กันระหว่าง Large itemset ที่ได้จากขั้นตอนก่อนหน้านั้นคือ L_{k-1}^{AUD} กับ L_{k-1}^{AUD} ผลลัพธ์ได้เป็น C_k^{AUD} ในรอบนั้นๆ แต่ขั้นตอนนี้ทจะเป็นการ join ระหว่างไอเท็มเซต ที่ประกอบด้วยไอเท็มที่มาจากแอททริบิวต์รองใหม่ คือ แอททริบิวต์ Area กับ ไอเท็มเซตที่ประกอบด้วยไอเท็มจากแอททริบิวต์ในฐานข้อมูลเดิมทั้งหมด ซึ่งจะทำการพิจารณาการ join เฉพาะแบบ inter-dimension join เท่านั้น

จากนั้นทำการประมาณค่าสนับสนุนให้แต่ละไอเท็มเซตและทำการระบุว่าไอเท็มเซตนั้นเป็นการประมาณค่าสนันสนุน แล้วนำไอเท็มเซต $L_k^{db(A)}$, C_k^{AUD} และ L_k^{DB} มารวมกันตามลำดับ แล้วนำไปเก็บใน L_k^{AUD} ดังรูปที่รูปที่ 3.16(ค) และจะเห็นได้ว่าไม่มี $L_3^{db(A)}$ ของแอททริบิวต์รองที่เพิ่มมาใหม่ ทำให้ L_3^{AUD} ประกอบไปด้วย C_3^{AUD} และ L_3^{DB} เท่านั้น

และในรอบที่ k = 4 แสดงดังรูปที่ 3.17(ข) เช่นกันจะเห็นได้ว่าไม่มี $L_4^{db(A)}$ ของแอททริบิวต์รองที่เพิ่มมาใหม่ และ L_4^{DB} ของฐานข้อมูลเดิม ดังนั้น L_4^{AUD} จะประกอบด้วย C_4^{AUD} ที่ได้จากการ join ระหว่าง L_3^{AUD} เท่านั้น เมื่อทำการ join ต่อระหว่าง L_4^{AUD} กับ L_4^{AUD} จะเห็นได้ว่าไม่สามารถ join ได้ จึงทำให้เสร็จสิ้นของขั้นตอนในส่วนที่ 2

หลังจากเสร็จขั้นตอนในส่วนที่ 2 จะได้ L_k^{AUD} ทั้งหมดใน AUD แล้วจะนำเอา L_k^{AUD} ที่ได้ไปทำการใช้พิจารณาในการทำงานของ ส่วนที่ 3

C_3^{AUD}		
item	support	mark
{1, a, I_2}	3	*
{1, a, I_3}	2	*
{1, b, I_1}	3	*
{1, b, I_3}	2	*
{1, I_1, I_2}	2	*
{1, I_1, I_3}	2	*
{1, I_2, I_3}	2	*
{1, I_2, I_4}	2	*
{2, a, I_2}	4	*
{2, a, I_3}	2	*
{2, b, I_1}	3	*
{2, b, I_3}	2	*
{2, I_1, I_2}	2	*
{2, I_1, I_3}	2	*
{2, I_2, I_3}	2	*
{2, I_2, I_4}	2	*

(ก)

L_3^{DB}	
Item	support
{a, I_2, I_3}	2
{b, I_1, I_3}	2

(ข)

L_3^{AUD}		
item	support	mark
{1,a, I_2}	3	*
{1,a, I_3}	2	*
{1,b, I_1}	3	*
{1,b, I_3}	2	*
{1,I_1, I_2}	2	*
{1,I_1, I_3}	2	*
{1,I_2, I_3}	2	*
{1,I_2, I_4}	2	*
{2,a, I_2}	4	*
{2,a, I_3}	2	*
{2,b, I_1}	3	*
{2,b, I_3}	2	*
{2,I_1, I_2}	2	*
{2,I_1, I_3}	2	*
{2,I_2, I_3}	2	*
{2,I_2, I_4}	2	*
{a,I_2, I_3}	2	
{b,I_1, I_3}	2	

(ค)

$L_3^{db(A)} \cup C_3^{db(A)} \cup L_3^{DB}$

รูปที่ 3.16 แสดงขั้นตอน การหา L_3^{AUD}

$C_4^{db(A)}$		
item	support	mark
{1, a, I_2, I_4}	2	*
{1, b, I_1, I_3}	2	*
{2, a, I_2, I_4}	2	*
{2, a, I_2, I_4}	2	*

(ก)

L_4^{AUD}		
item	support	mark
{1, a, I_2, I_4}	2	*
{1, b, I_1, I_3}	2	*
{2, a, I_2, I_4}	2	*
{2, b, I_1, I_3}	2	*

(ข)

$L_4^{AUD} \cup C_4^{db(A)} \cup L_4^{DB}$

รูปที่ 3.17 แสดงขั้นตอนการหา L_4^{AUD}

3. ส่วนที่ 3 การค้นหา Larege itemset ใน UD

การทำงานส่วนที่ 3 จะเป็นการนำเอา Large itemset จากส่วนที่ 2 มาทำการหา Large iteset ทั้งหมดในฐานข้อมูลที่ปรับปรุงใหม่

3.1 ขั้นตอนการหา L_1^{UD}

ทำการค้นหาใน db(T) เพื่อทำการปรับปรุงค่าสนับสนุนของไอเท็ม

1. ถ้าไอเท็ม ใน db(T)ไหนปรากฏใน L_1^{AUD} ดังรูปที่ 3.18(ก) ที่ได้จากการทำงานส่วนที่ 2 ทำการปรับปรุงค่าสนับสนุนของไอเท็มเซตนั้นที่อยู่ใน L_1^{AUD} โดยค่าสนับสนุนที่ได้เป็นค่าสนับสนุนของฐานข้อมูลใหม่ที่ปรับปรุงดังรูปที่ 3.18(ข)

2. ถ้าไอเท็มใน db(T) ตัวไหนที่ไม่ปรากฏใน L_1^{AUD} จะถือว่า ไอเท็มนั้นเป็นไอเท็มเกิดขึ้นใหม่ใน db(T) นั้นคือเป็น candidate 1-itemset ใน db(T) หรือ $C_1^{db(T)}$ แสดงดังรูปที่ 3.19(ก) แล้วนำค่าสนับสนุนของไอเท็มเซตใน $C_1^{db(T)}$ มาตรวจสอบว่า ถ้า $X.support_d < s \times d$ นั้นคือพิจารณาว่าน้อยกว่าค่าสนับสนุนใน db(T) ถ้าน้อยกว่าจะทำการตัดทิ้งไอเท็มเซตที่ไม่ผ่านค่าสนับสนุนออกจาก $C_1^{db(T)}$ ดังรูปที่ 3.19 (ข) เห็นได้ว่า {I_7} ถูกตัดออกจาก $C_1^{db(T)}$ แล้วนำไอเท็มเซตที่เหลือ ดังรูปที่ 3.19 (ค) ไปค้นหาต่อในฐานข้อมูลส่วนAUD เพื่อทำการนับค่าสนับสนุนของไอเท็มในส่วน AUD ผลลัพธ์ที่ได้ของค่าสนับสนุนของฐานข้อมูลปรับปรุงใหม่ แสดงดังรูปที่ 3.19 (ง)

3. จากนั้น นำเอาไอเท็มเซตใน L_1^{AUD} และ $C_1^{db(T)}$ ที่ได้จากขั้นตอนที่ 1 และ 2 ดังรูปที่ 3.20 (ก) และรูปที่ 3.20(ข) มาทำการตรวจสอบว่า ถ้า $X.support_{UD} \geq s \times (AUD + d)$ แสดงว่าไอเท็มนั้น เป็น winner itemset จะนำเอาไอเท็มเซตนั้นไปเก็บใน L_1^{UD} ของฐานข้อมูลที่ปรับปรุง ดังรูปที่ 3.20 (ค)

L_1^{AUD}		Scan db(T)
itemset	support	support
{1}	3	+2
{2}	4	+1
{a}	4	+1
{b}	3	+3
$\{I_1\}$	4	+3
$\{I_2\}$	5	+3
$\{I_3\}$	4	+1
$\{I_4\}$	2	+0

(ก)

L_1^{AUD}	
itemset	support
{1}	5
{2}	5
{a}	5
{b}	6
$\{I_1\}$	7
$\{I_2\}$	8
$\{I_3\}$	5
$\{I_4\}$	2

(ข)

รูปที่ 3.18 แสดงการปรับปรุงค่าสนับสนุนไอเท็มเซตใน L_1^{AUD} ภายหลังการค้นหาส่วนของ db(T)

$C_1^{db(T)}$	
Itemset	**Support**
{3}	3
{c}	2
$\{I_5\}$	4
$\{I_6\}$	5
$\{I_7\}$	1

(ก)

$x.support_1 \geq (s-d)$

$C_1^{db(T)}$	
Itemset	**Support**
{3}	3
{c}	2
$\{I_5\}$	4
$\{I_6\}$	5

(ข)

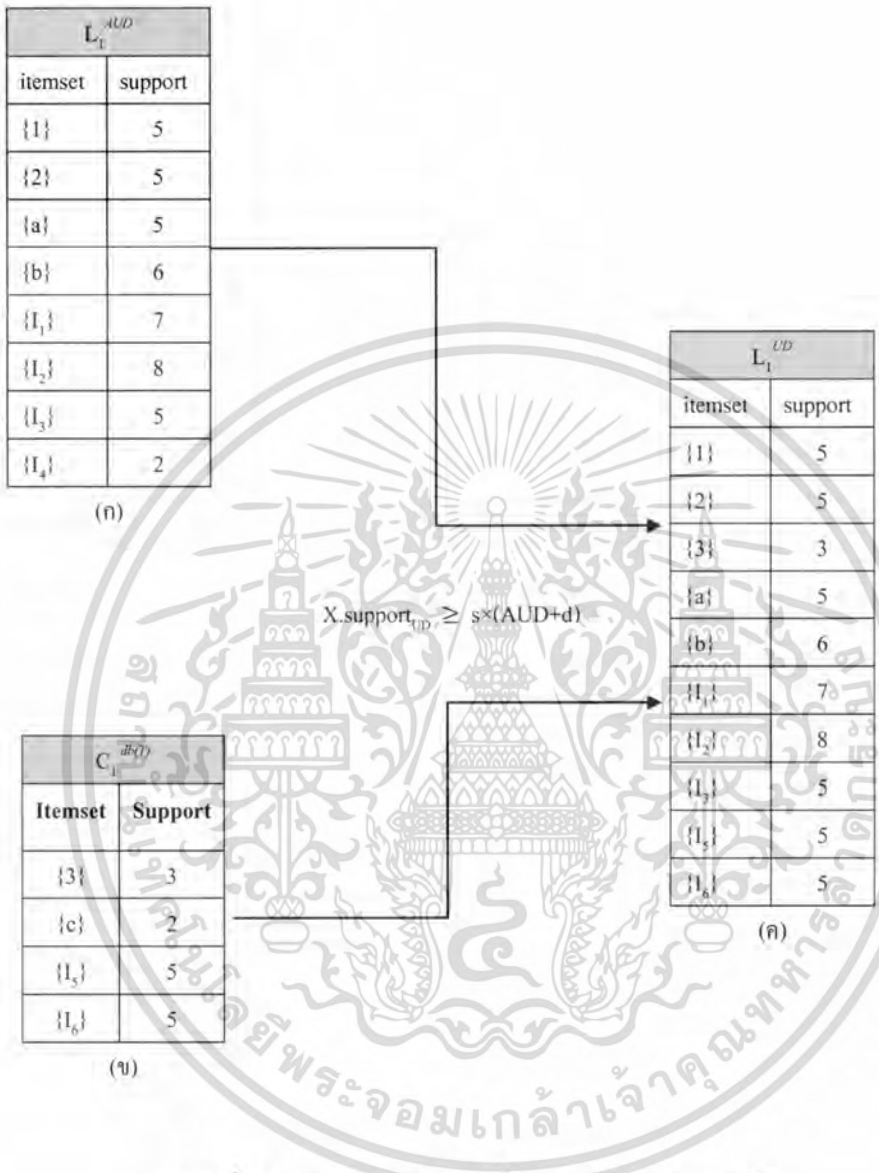
$C_1^{db(T)}$		Scan AUD
Itemset	**Support**	**Support**
{3}	3	+0
{c}	2	+0
$\{I_5\}$	4	+1
$\{I_6\}$	5	+0

(ค)

$C_1^{db(T)}$	
Itemset	**Support**
{3}	3
{c}	2
$\{I_5\}$	5
$\{I_6\}$	5

(ง)

รูปที่ 3.19 แสดงการหาค่า $C_1^{db(T)}$ ใน db(T) และปรับปรุงค่า support หลังค้นหาส่วน AUD



รูปที่ 3.20 แสดงการหา L_1^{UD} ทั้งหมดจาก L_1^{AUD} และ $C_1^{db(T)}$

3.2 ขั้นตอนการหา L_2^{UD}

1. จากรูปที่ 3.21 เป็นการหา $C_2^{db(T)}$ ด้วยการ join กันระหว่าง L_1^{UD} กับ L_1^{UD} ตาม procedure hybrid_gen1 ซึ่งการ join โดยผลที่ได้จากการ join ไม่เป็นสมาชิกอยู่ใน L_2^{AUD} ดังรูป 3.21 (ก)

2. จากนั้นทำการค้นหาใน db(T) เพื่อนับค่าสนับสนุนดังรูป 3.21 (ข) แล้วนำมาพิจารณาว่า ถ้า $X.support_d \geq s \times d$ จะถูกเก็บใน $C_2^{db(T)}$ ดังรูป 3.21 (ค) เพื่อไปค้นหาใน AUD ต่อ แต่ $X.support_d < s \times d$ จะทำการตัดไอเท็มนั้นออกจาก $C_2^{db(T)}$ เช่น $\{1, I_5\} = 1$ มีค่าสนับสนุนน้อยกว่า (0.2×7) จะทำการตัด$\{1, I_5\}$ ทิ้ง

3. จากนั้น นำ L_2^{AUD} จากส่วนที่ 2 มาทำการตัดไอเท็มเซตที่มีซับเซตอยู่ใน Y โดยที่ Y เป็นไอเท็มเซตได้มาจาก $L_1^{AUD} - L_1^{UD}$ แล้วมาทำการพิจารณาว่า ถ้าไอเท็มเซตใน L_2^{AUD} ที่มีซับเซตใน Y จะตัดไอเท็มเซตนั้นออกจาก L_2^{AUD} จากรูปที่ 3.22 จะเห็นได้ว่า $Y = \{ I_4\}$ ไอเท็มเซต $\{1,I_4\}$ ใน L_2^{AUD} มี I_4 เป็นซับเซตจะทำการตัด $\{1,I_4\}$ ออกจาก L_2^{AUD} แล้วนำเอา L_2^{AUD} ที่เหลือไปทำการค้นหาใน db(T) เพื่อปรับค่าสนับสนุน ดังรูปที่ 3.23 (ก) ทำการตรวจสอบ 2 กรณี

- ถ้า $X.support_{UD} \geq s \times (AUD + d)$ และ X.sup_mark = false นำไอเท็มเซตนั้นไปเพิ่มใน L_2^{UD} ดังรูป 3.23(ข) เช่น ไอเท็มเซต $\{a,I_2\}$ มีค่าสนับสนุนใน UD เท่ากับ 4 ซึ่งมากกว่าหรือเท่ากับ $0.2 \times (13 + 7)$ และ sup_mark ของ $\{a,I_2\}$ เท่ากับ false จึงนำไอเท็มเซต $\{a,I_2\}$ ไปเก็บใน L_2^{UD}
- ถ้า $X.support_{UD} \geq s \times (AUD + d)$ และ X.sup_mark = true ทำการกำหนดให้ค่าสนับสนุนของไอเท็มเซต เท่ากับค่าสนับสนุนที่ค้นหาได้ใน db(T) แล้วนำไอเท็มเซตนั้นไปเพิ่มใน $C_2^{db(T)}$ ดังรูป 3.23 (ค) เช่น ไอเท็มเซต {1,b} มีค่า support ใน UD เท่ากับ 4 ซึ่งมากกว่าหรือเท่ากับ $0.2 \times (13 + 7)$ แต่ sup_mark ของ {1,b} เท่ากับ ture ดังนั้น ค่าสนับสนุนของ {1,b} = 1 แล้วนำไปเพิ่มใน $C_2^{db(T)}$

4. นำ $C_2^{db(T)}$ ทำการค้นหาในฐานข้อมูลส่วนของ AUD เพื่อทำการปรับปรุงค่าสนับสนุน แล้วทำการพิจารณาว่า ถ้า $X.support_{UD} \geq s \times (AUD + d)$ จะทำการเพิ่มไอเท็มเซตนั้นใน L_2^{UD} ดังรูป 3.24

$(L_1^{UD} \bowtie L_1^{UD}) - L_2^{AUD}$
itemset
{1, I_5}
{1, I_6}
{2, I_5}
{2, I_6}
{3, a}
{3, b}
{3, I_1}
{3, I_2}
{3, I_3}
{3, I_5}
{3, I_6}
{a, I_1}
{a, I_5}
{a, I_6}
{b, I_2}
{b, I_5}
{b, I_6}
{I_1, I_5}
{I_1, I_6}
{I_2, I_5}
{I_2, I_6}
{I_3, I_5}
{I_3, I_6}
{I_5, I_6}

(ก)

$C_2^{db(T)}$	Scan db(T)
itemset	support
{1, I_5}	1
{1, I_6}	2
{2, I_5}	0
{2, I_6}	1
{3, a}	1
{3, b}	1
{3, I_1}	2
{3, I_2}	2
{3, I_3}	0
{3, I_5}	3
{3, I_6}	2
{a, I_1}	1
{a, I_5}	1
{a, I_6}	1
{b, I_2}	1
{b, I_5}	2
{b, I_6}	3
{I_1, I_5}	2
{I_1, I_6}	3
{I_2, I_5}	2
{I_2, I_6}	2
{I_3, I_5}	0
{I_3, I_6}	1
{I_5, I_6}	3

(ข)

$X.support_d \geq s \times d$

$C_2^{db(T)}$	
itemset	support
{1, I_6}	2
{3, I_1}	2
{3, I_2}	2
{3, I_5}	3
{3, I_6}	2
{b, I_5}	2
{b, I_6}	3
{I_1, I_5}	2
{I_1, I_6}	3
{I_2, I_5}	2
{I_2, I_6}	2
{I_5, I_6}	3

(ค)

รูปที่ 3.21 แสดงขั้นตอนการหา $C_2^{db(T)}$ ใน db(T)

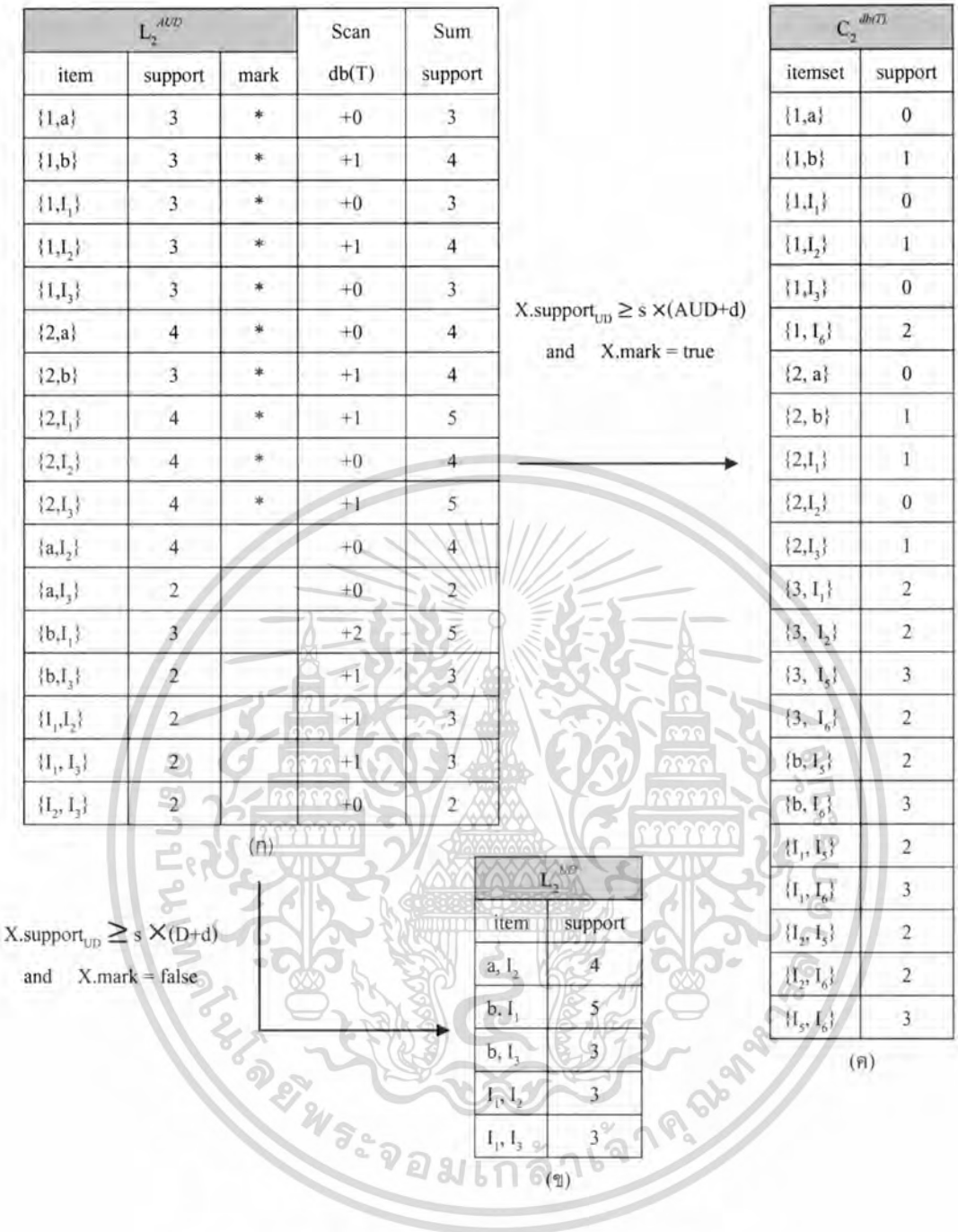
L_2^{AUD}		
item	support	mark
{1, a}	3	*
{1,b}	3	*
$\{1,I_1\}$	3	*
$\{1,I_2\}$	3	*
$\{1,I_3\}$	3	*
$\{1,I_4\}$	2	*
{2, a}	4	*
{2, b}	3	*
$\{2, I_1\}$	4	*
$\{2, I_2\}$	4	*
$\{2, I_3\}$	4	*
$\{2, I_4\}$	2	*
$\{a, I_2\}$	4	
$\{a, I_3\}$	2	
$\{b, I_1\}$	3	
$\{b, I_3\}$	2	
$\{I_1, I_2\}$	2	
$\{I_1, I_3\}$	2	
$\{I_2, I_3\}$	2	
$\{I_2, I_4\}$	2	

$Y = L_1^{AUD} - L_1^{UD}$
$\{I_4\}$

ตัดไอเท็ม L_2^{AUD} ที่มี subset ใน Y

L_2^{AUD}		
item	support	mark
{1, a}	3	*
{1,b}	3	*
$\{1,I_1\}$	3	*
$\{1,I_2\}$	3	*
$\{1,I_3\}$	3	*
{2, a}	4	*
{2, b}	3	*
$\{2,I_1\}$	4	*
$\{2,I_2\}$	4	*
$\{2,I_3\}$	4	*
$\{a, I_2\}$	4	
$\{a, I_3\}$	2	
$\{b,I_1\}$	3	
$\{b,I_3\}$	2	
$\{I_1,I_2\}$	2	
$\{I_1,I_3\}$	2	
$\{I_2, I_3\}$	2	

รูปที่ 3.22 แสดงขั้นตอนการตัดไอเท็มเซตใน L_2^{AUD} ที่มีซับเซตย่อยอยู่ใน $L_1^{AUD} - L_1^{UD}$



รูปที่ 3.23 แสดงขั้นตอนการพิจารณา L_2^{AUD} ที่สามารถเป็น L_2^{UD}

$C_2^{db(T)}$		Scan AUD	Sum support
itemset	support		
{1,a}	0	+3	3
{1,b}	1	+0	1
{1,I_1}	0	+1	1
{1,I_2}	1	+3	4
{1,I_3}	0	+1	1
{1, I_6}	2	+0	2
{2, a}	0	+1	1
{2, b}	1	+3	4
{2,I_1}	1	+3	4
{2,I_2}	0	+2	2
{2,I_3}	1	+3	4
{3, I_1}	2	+0	2
{3, I_2}	2	+0	2
{3, I_5}	3	+0	3
{3, I_6}	2	+0	2
{b, I_5}	2	+0	2
{b, I_6}	3	+0	3
{I_1, I_5}	2	+1	3
{I_1, I_6}	3	+0	3
{I_2, I_5}	2	+1	3
{I_2, I_6}	2	+0	2
{I_5, I_6}	3	+0	3

$X.support_{UD} \geq s \times (AUD+d)$ →

L_2^{UD}	
itemset	support
{1,a}	3
{1,I_2}	4
{2, b}	4
{2,I_1}	4
{2,I_3}	4
{3, I_5}	3
{a, I_2}	4
{b, I_1}	5
{b, I_3}	3
{b, I_6}	3
{I_1, I_2}	3
{I_1, I_3}	3
{I_1, I_5}	3
{I_1, I_6}	3
{I_2, I_5}	3
{I_5, I_6}	3

รูปที่ 3.24 แสดงขั้นตอนการพิจารณา $C_2^{db(T)}$ ที่สามารถเป็น L_2^{UD}

$(L_2^{UD} \bowtie L_2^{UD}) - L_3^{AUD}$
itemset
{b, I_1,I_6}
{I_1, I_2,I_5}
{I_1, I_5,I_6}

(ก)

$C_3^{db(T)}$	Scan
itemset	Db(T)
{b, I_1,I_6}	2
{I_1, I_2,I_5}	1
{I_1, I_5,I_6}	2

(ข)

$X.support_d >= s \times d$

$C_3^{db(T)}$	
itemset	support
{b, I_1,I_6}	2
{I_1, I_5,I_6}	2

(ค)

รูปที่ 3.25 แสดงขั้นตอนการหา $C_3^{db(T)}$

3.3 ขั้นตอนการหา L_3^{UD}

1. จากรูปที่ 3.25 เป็นการหา $C_3^{db(T)}$ ด้วยการ join กันระหว่าง L_2^{UD} กับ L_2^{UD} ตาม procedure hybrid_gen2 ซึ่งการ join โดยที่ผลจากการ join ไม่มีไอเท็มเซตใดเป็นสมาชิกอยู่ใน L_3^{AUD} ดังรูป3.25 (ก)

2. ทำการค้นหา $C_3^{db(T)}$ ใน db(T) เพื่อนับค่าสนับสนุนแล้วนำมาพิจารณาว่า ถ้า $X.support_d \geq s \times d$ จะถูกเก็บใน $C_3^{db(T)}$ ดังรูป 3.25 (ค) เพื่อไปค้นหาในฐานข้อมูลส่วน AUD ต่อไป แต่ถ้า $X.support_d < s \times d$ จะทำการตัดไอเท็มนั้นออกจาก $C_3^{db(T)}$ เช่น $\{I_1, I_2, I_5\} = 1$ มีค่าสนับสนุนน้อยกว่า (0.2×7) จะทำการตัด{I_1, I_5} ทิ้ง

3. นำ L_3^{AUD} มาทำการตัดไอเท็มเซตที่มีซับเซตย่อยอยู่ใน Y โดยที่ Y เป็นไอเท็มเซตได้มาจาก $L_2^{AUD} - L_2^{UD}$ แล้วมาทำการพิจารณาว่า ถ้า ไอเท็มเซตใน L_3^{AUD} ที่มีซับเซตใน Y จะตัดไอเท็มเซตนั้นออกจาก L_3^{AUD}

จากรูปที่ 3.26 ตัวอย่างเช่น Y = {1,I_1} ไอเท็มเซต {1,b,I_1} ใน L_3^{AUD} มี {1,I_1} เป็นซับเซต จะทำการตัด {1,b,I_1} ออกจาก L_3^{AUD} แล้วนำเอา L_3^{AUD} ที่เหลือไปทำการค้นหาใน db(T) เพื่อปรับค่าสนับสนุน ดังรูปที่ 3.27 (ก) ทำการตรวจสอบ 2 กรณี

- ถ้า $X.support_{UD} \geq s \times (AUD + d)$ และ X.mark = false นำไอเท็มเซตนั้นไปเพิ่มใน L_2^{UD} ดังรูป 3.27 (ข) เช่น ไอเท็มเซต {b,I_1,I_3} มีค่า support ใน UD เท่ากับ 3 ซึ่งมากกว่าหรือเท่ากับ 0.2 ×(13 + 7) และ sup_mark ของ {b,I_1,I_3} เท่ากับ false แล้วนำ{b,I_1,I_3}ไปเก็บใน L_2^{UD}
- ถ้า $X.support_{UD} \geq s \times (AUD + d)$ และ X.mark = true ทำการกำหนดให้ค่าสนับสนุนของไอเท็มเซต เท่ากับค่าสนับสนุนที่ค้นหาได้ใน db(T) แล้วนำไอเท็มเซตนั้นไปเพิ่มใน $C_3^{db(T)}$ ดังรูป 3.27(ค) เช่น ไอเท็มเซต {1,a,I_2} มีค่า support ใน UD เท่ากับ 3 ซึ่งมากกว่าหรือเท่ากับ 0.2 ×(13 + 7) แต่ mark ของ {1,a,I_2}เท่ากับ ture ดังนั้น กำหนดค่าสนับสนุนของ {1,a,I_2}= 0 แล้วนำไปเพิ่มใน $C_3^{db(T)}$

4. จากรูปที่ 3.28 เป็นการนำ $C_3^{db(T)}$ ไปทำการค้นหาในฐานข้อมูลส่วน AUD ดังรูปที่ 3.28 (ข) เพื่อทำการปรับปรุงค่าสนับสนุน แล้วทำการพิจารณาว่า ถ้า $X.support_{UD} \geq s \times (AUD + d)$ จะทำการเพิ่มไอเท็มเซตนั้นใน L_3^{UD} ดังรูป 3.28(ค)

L_3^{UD}		
item	support	mark
{1,a, I_2}	3	*
{1,a, I_3}	2	*
{1,b, I_1}	3	*
{1,b, I_3}	2	*
{1,I_1, I_2}	2	*
{1,I_1, I_3}	2	*
{1,I_2, I_3}	2	*
{1,I_2, I_4}	2	*
{2,a, I_2}	4	*
{2,a, I_3}	2	*
{2,b, I_1}	3	*
{2,b, I_3}	2	*
{2,I_1, I_2}	2	*
{2,I_1, I_3}	2	*
{2,I_2, I_3}	2	*
{2,I_2, I_4}	2	*
{a,I_2, I_3}	2	
{b,I_1, I_3}	2	

$Y = L_2^{UD} - L_2^{UD}$
{1,b}
{1,I_1}
{1,I_3}
{1,I_4}
{2,a}
{2,I_2}
{a,I_3}
{I_2,I_3}
{I_2,I_4}

ตัดไอเท็มเซต L_3^{AUD} ที่มี subset ใน Y

L_3^{UD}		
item	support	mark
1, a, I_2	3	*
2, b, I_1	3	*
2, b, I_3	2	*
2, I_1, I_2	2	*
b, I_1, I_3	2	

รูปที่ 3.26 แสดงขั้นตอนการตัดไอเท็มเซต L_3^{UD} ที่มีซับเซตย่อยใน $L_2^{UD} - L_2^{UD}$

L_3^{AUD}			Support db(T)	Sum support
item	support	mark		
{1,a,I_2}	3	*	+0	3
{2,b,I_1}	3	*	+1	4
{2,b,I_3}	2	*	+1	3
{2,I_1,I_2}	2	*	+1	3
{b,I_1,I_3}	2		+1	3

(ก)

X.support ≥ s ×(D+d)
and X.mark = true

$C_3^{db(T)}$	
item	support
{1, a, I_2}	0
{2, b, I_1}	1
{2, b, I_3}	1
{2, I_1, I_2}	1
{b, I_1,I_6}	2
{I_1, I_5,I_6}	2

(ข)

X.support ≥ s × (D+d)
and X.mark = false

L_3^{UD}	
item	support
{b, I_1, I_3}	3

(ค)

รูปที่ 3.27 แสดงขั้นตอนการพิจารณา L_3^{AUD} ที่สามารถเป็น L_3^{UD}

$C_3^{db(T)}$	
item	support
{1,a,I_2}	0
{2,b,I_1}	1
{2,b,I_3}	1
{2,I_1,I_2}	1
{b,I_1,I_6}	2
{I_1,I_5,I_6}	2

(ก)

$C_3^{db(T)}$		Support	Sum
item	support	AUD	support
{1,a,I_2}	0	+3	3
{2,b,I_1}	1	+3	4
{2,b,I_3}	1	+2	3
{2,I_1,I_2}	1	+2	3
{b,I_1,I_6}	2	+0	2
{I_1,I_5,I_6}	2	+0	2

(ข)

X.support ≥ s ×(D+d)

L_3^{UD}	
item	support
1,a,I_2	3
2,b,I_1	4
2,b,I_3	3
2,I_1,I_2	3
b, I_1,I_3	3

(ค)

รูปที่ 3.28 แสดงขั้นตอนการพิจารณา $C_3^{db(T)}$ ที่สามารถเป็น L_3^{UD}

$(L_3^{UD} \bowtie L_1^{UD}) - L_4^{AUD}$
Ø

รูปที่ 3.29 แสดงขั้นตอนการหา $C_3^{db(T)}$

$Y = L_3^{UD} - L_3^{UD}$
{1,a,I_3}
{1,b,I_1}
{1,b,I_3}
{1, I_1,I_2}
{1, I_1,I_3}
{1, I_2,I_3}
{1, I_2,I_4}
{2,a,I_2}
{2,a,I_3}
{2,I_1,I_2}
{2,I_2,I_3}
{2,I_2,I_4}
{a, I_2,I_3}

L_4^{AUD}		
item	support	mark
{1, a, I_2, I_4}	2	*
{1, b, I_1, I_3}	2	*
{2, a, I_2, I_4}	2	*
{2, b, I_1, I_3}	2	*

ตัดไอเท็มเซต L_3^{AUD} ที่มี subset ใน

L_4^{AUD}		
item	support	mark
2, b, I_1, I_3	2	*

รูปที่ 3.30 แสดงขั้นตอนการตัดไอเท็ม L_4^{UD} ที่มีซับเซตใน $L_3^{UD} - L_3^{UD}$

3.4 ขั้นตอนการหา L_4^{UD}

1. จากรูปที่ 3.29 เป็นการหา $C_4^{db(T)}$ ด้วยการ join กันระหว่าง L_3^{UD} กับ L_3^{UD} ตาม Procedure hybrid_gen2 โดยที่ผลจากการ join ไม่เป็นสมาชิกอยู่ใน L_4^{AUD} ดังรูป 3.29 ปรากฏว่าเป็นเซตว่างจึงทำให้ไม่มี $C_4^{db(T)}$ ที่ต้องค้นหาใน db(T)

2. จากรูปที่ 3.30 นำ L_4^{AUD} มาทำการตัดไอเท็มเซตที่มีซับเซตย่อยอยู่ใน Y โดยที่ Y เป็นไอเท็มเซตได้มาจาก L_3^{AUD} - L_3^{UD} ตัวอย่างเช่น Y = {1, I_2,I_4} ไอเท็มเซต {1, a, I_2, I_4} ใน L_4^{AUD} มี {1, I_2,I_4} เป็นซับเซต จะทำการตัด {1, a, I_2, I_4} ออกจาก L_4^{AUD} แล้วนำเอา L_4^{AUD} ที่เหลือไปทำการค้นหาใน db(T) เพื่อปรับค่าสนับสนุน ดังรูปที่ 3.31 (ก) ทำการตรวจสอบ 2 กรณี

- ถ้า $X.support_{UD} \geq s \times (AUD + d)$ และ X.mark = false นำไอเท็มเซตนั้นไปเพิ่มใน L_4^{UD} ซึ่งไม่มีในเงื่อนไขนี้ปรากฏใน L_4^{AUD}
- ถ้า $X.support_{UD} \geq s \times (AUD + d)$ และ X.mark = true ทำการกำหนดให้ค่าสนับสนุนของไอเท็มชุดเท่ากับค่าสนับสนุนที่ค้นหาได้ใน db(T) แล้วนำไอเท็มเซตนั้นไปเพิ่มใน $C_4^{db(T)}$ ดังรูป 3.31 (ข) เช่น ไอเท็มเซต {2, b, I_1, I_3} มีค่า support ใน UD เท่ากับ 3 ซึ่งมากกว่าหรือเท่ากับ $0.2 \times (13 + 7)$ แต่ mark ของ {2, b, I_1, I_3} เท่ากับ ture ดังนั้น ค่าสนับสนุนของ {2, b, I_1, I_3} = 1 แล้วนำไปเพิ่มใน $C_4^{db(T)}$

3. จากรูปที่ 3.32(ก) เป็นการนำ $C_4^{db(T)}$ ไปทำการค้นหาในส่วนของ AUD เพื่อทำการปรับปรุงค่าสนับสนุน แล้วทำการพิจารณาว่า ถ้า $X.support_{UD} \geq s \times (AUD + d)$ จะทำการเพิ่มไอเท็มเซตนั้นใน L_4^{UD} ดังรูปที่รูปที่ 3.32(ก) เมื่อพิจารณาต่อในการหา L_5^{UD} ไม่สามารถ join L_4^{UD} ได้จึงหยุดการทำงานของอัลกอริทึม

L_4^{AUD}			Support	Sum
item	support	mark	db(T)	support
{2, b, I_1, I_3}	2	*	+1	3

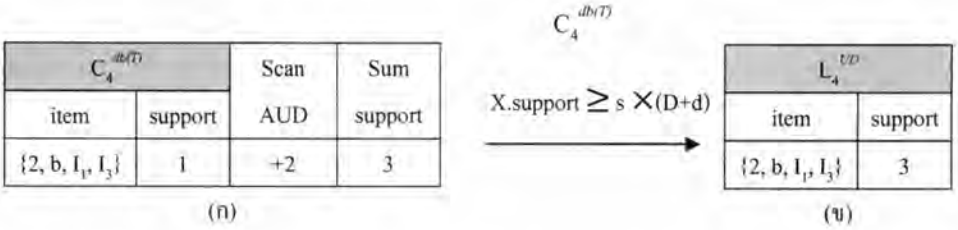
(ก)

$X.support \geq s \times (D+d)$ and X.mark = true

$C_4^{db(T)}$	
item	support
{2, b, I_1, I_3}	1

(ข)

รูปที่ 3.31 แสดงการพิจารณาหา L_4^{AUD} ที่สามารถเป็น



รูปที่ 3.32 แสดงการพิจารณาหา $C_4^{db(T)}$ ที่สามารถเป็น L_4^{UD}

บทที่ 4
ผลการทดลอง

ในบทนี้จะกล่าวถึงการทดลองเพื่อเปรียบเทียบวัดประสิทธิภาพในด้านความถูกต้อง และเวลาสำหรับใช้ในการทำงานของอัลกอริทึมการค้นหากฎความสัมพันธ์แบบเพิ่มขยายสำหรับการเพิ่มข้อมูลที่มีแอททริบิวต์ใหม่กับอัลกอริทึมอื่นๆ และนำผลการทดลองที่ได้มาทำการวิเคราะห์ผลสรุปของการทดลอง

4.1 วัตถุประสงค์การทดลอง

เพื่อแสดงให้เห็นถึงประสิทธิภาพการทำงานของอัลกอริทึมการค้นหากฎความสัมพันธ์แบบเพิ่มขยายสำหรับการเพิ่มข้อมูลที่มีแอททริบิวต์ใหม่ เมื่อมีการเพิ่มข้อมูลแอททริบิวต์รอง และข้อมูลทรานแซคชั่นใหม่เข้าสู่ฐานข้อมูลเดิมที่เป็นลักษณะฐานข้อมูลแบบหลายมิติ โดยมีวัตถุประสงค์ในการทดลองเพื่อเปรียบเทียบประสิทธิภาพการทำงานกับอัลกอริทึมที่ใช้ในการค้นหากฎความสัมพันธ์แบบต่างๆ ดังนี้ คือ

1. เพื่อทดสอบความถูกต้องของความสัมพันธ์ที่ได้จากการเพิ่มแอททริบิวต์รอง และทรานแซคชั่นใหม่เข้าไปในฐานข้อมูลเดิม

อัลกอริทึมการค้นหากฎความสัมพันธ์แบบเพิ่มขยายสำหรับการเพิ่มข้อมูลที่มีแอททริบิวต์ใหม่เป็นอัลกอริทึมที่มีการทำงานอยู่บนพื้นฐานของอัลกอริทึมอะพริโอริ ดังนั้นในการทดสอบความถูกต้องของอัลกอริทึม จะทำการทดสอบเปรียบเทียบผลลัพธ์ของจำนวน Large itemset ที่ได้จากการค้นหากฎความสัมพันธ์ของอัลกอริทึมในงานวิจัยนี้ กับอัลกอริทึมอะพริโอริ และอัลกอริทึมที่มีรูปแบบการทำงานอยู่บนพื้นฐานของอัลกอริทึมอะพริโอริ ซึ่งมีรูปแบบการทำงานเป็นลักษณะของการทำงานวนรอบซ้ำเพื่อค้นหา Large itemset ในฐานข้อมูล โดยมีการใช้ Large itemset รอบก่อนหน้ามาใช้ในการค้นหา Large itemset ในรอบถัดไป ที่เรียกว่า Level-wise

อัลกอริทึมที่นำมาเปรียบเทียบความถูกต้องของการค้นหากฎความสัมพันธ์ของการเพิ่มฐานข้อมูลใหม่เข้าไปในฐานข้อมูลเดิมที่ประกอบด้วย 2 อัลกอริทึมดังนี้คือ

1.1 อัลกอริทึมอะพริโอริ

การทำงานของอัลกอริทึมอะพริโอริ เมื่อกรณีของการเพิ่มข้อมูลแอททริบิวต์รองและทรานแซคชั่นใหม่เข้าสู่ฐานข้อมูลเดิม อัลกอริทึมอะพริโอริจะต้องทำการค้นหากฎความสัมพันธ์ของข้อมูลในฐานข้อมูลที่ถูกปรับปรุงใหม่ทั้งหมด โดยไม่มีการนำความรู้จากการค้นหาความสัมพันธ์ในฐานข้อมูลเดิมมาใช้ ดังนั้น อัลกอริทึมอะพริโอริจะทำการวนรอบซ้ำเพื่อค้นหากฎความสัมพันธ์ทั้งหมดใหม่ นั่นคือค้นหาในข้อมูลที่เป็นฐานข้อมูลเดิมรวมกับฐานข้อมูลที่เพิ่มเข้ามาใหม่

1.2 อัลกอริทึมอะพริโอริสำหรับการหากฎความสัมพันธ์แบบมิติผสม

การทำงานของอัลกอริทึมอะพริโอริสำหรับการหากฎความสัมพันธ์แบบมิติผสม เมื่อมีการเพิ่มแอททริบิวต์และทรานแซคชั่นเข้าไปในฐานข้อมูลเดิม จะมีการทำงานที่ลักษณะคล้ายกับการทำงานของอัลกอริทึมอะพริโอริ คือ ต้องทำการวนรอบซ้ำเพื่อค้นหากฎความสัมพันธ์ใหม่ทั้งหมด แต่อัลกอริทึมอะพริโอริสำหรับการหากฎความสัมพันธ์แบบมิติผสมจะต่างกับอัลกอริทึมอะพริโอริ ที่รูปแบบของการเชื่อมเพื่อหาความสัมพันธ์ (join) ระหว่างไอเท็มเซต เพื่อลดจำนวนของไอเท็มเซตที่ไม่สามารถเกิดความสัมพันธ์ขึ้นได้ในฐานข้อมูลนั้นคือ ความสัมพันธ์ระหว่างไอเท็มเซตที่มาจากแอททริบิวต์รองเดียวกัน ซึ่งค่าความสัมพันธ์ของแอททริบิวต์รองเดียวไม่สามารถเกิดขึ้นพร้อมกันได้ภายในไอเท็มเซต เช่น แอททริบิวต์เพศ ภายในไอเท็มเซตไม่สามารถมีเพศ ชาย และหญิงอยู่ในไอเท็มเซตเดียวกันได้

ในงานวิจัยนี้จะทำการเปรียบเทียบความถูกต้องที่ได้จากการค้นหากฎความสัมพันธ์โดยการเปรียบเทียบกับ Large itemset ที่ได้จากการปรับปรุงฐานข้อมูลกับอัลกอริทึมต่างๆ ดังกล่าวข้างต้น

2. เพื่อวัดประสิทธิภาพของเวลาการทำงานในการเพิ่มขยายการค้นหากฎความสัมพันธ์ของอัลกอริทึมในกรณีของการเพิ่มแอททริบิวต์รอง และทรานแซคชั่นใหม่เข้าไปในฐานข้อมูลเดิม

การทดสอบเพื่อวัดประสิทธิภาพเวลาการทำงานของอัลกอริทึมในวัตถุประสงค์นี้ เป็นการทดสอบเพื่อวัดประสิทธิภาพอัลกอริทึมการค้นหากฎความสัมพันธ์แบบเพิ่มขยายสำหรับการเพิ่มข้อมูลที่มีแอททริบิวต์ใหม่ โดยวัดจากเวลาการทำงาน (Execution time) ที่ได้จากการค้นหากฎความสัมพันธ์ในฐานข้อมูลที่ปรับปรุง เมื่อทำการเพิ่มแอททริบิวต์รองและทรานแซคชั่นใหม่ที่ขนาดต่างๆกัน เพื่อใช้เป็นตัววัดประสิทธิภาพ สำหรับงานวิจัยนี้ โดยจะทำการทดสอบเปรียบเทียบประสิทธิภาพการทำงานของอัลกอริทึมการค้นหากฎความสัมพันธ์แบบเพิ่มขยายสำหรับการเพิ่มข้อมูลที่มีแอททริบิวต์ใหม่ กับ 2 อัลกอริทึมที่ได้กล่าวมาข้างต้น

ทำการทดสอบเพื่อวัดประสิทธิภาพของการที่อัลกอริทึมในงานวิจัยนี้ที่มีการนำหลักการของการประมาณค่าสนับสนุนมาใช้ในส่วนของขั้นตอนการค้นหาค่าสนับสนุนของ

ความสัมพันธ์ระหว่าง Large itemset ของข้อมูลเดิมกับ Lage itemset จากแอททริบิวต์รองใหม่ เพื่อลดเวลาการค้นหาค่าสนับสนุนให้กับ Large itemset ในข้อมูลเดิมซ้ำซ้อน มาเปรียบเทียบกับการทำงานของอัลกอริทึมงานวิจัยนี้ในรูปแบบของวิธีการทำการค้นหาค่าสนับสนุนจริงในข้อมูลที่ใช้หาความสัมพันธ์ โดยวัดจากเวลาการทำงานของการประมาณค่าสนับสนุนกับเวลาการทำงานของการค้นหาค่าสนับสนุนจริง

4.2 อุปกรณ์และข้อมูลที่ใช้ในการทดลอง

ในการทดลองเพื่อวัดประสิทธิภาพของอัลกอริทึมการค้นหากฎความสัมพันธ์แบบเพิ่มขยายสำหรับการเพิ่มข้อมูลที่มีแอททริบิวต์ใหม่ตามวัตถุประสงค์ที่กำหนดได้ทำการทดสอบบนเครื่องคอมพิวเตอร์พีซี โดยคุณสมบัติของเครื่องคอมพิวเตอร์พีซีที่ใช้ทดสอบมีดังนี้

- RAM 2.96 GB
- Hard Disk 2.93 GHz
- CPU Intel(R) core(TM)i7

ซอฟแวร์ที่ใช้ในการพัฒนาโปรแกรมสำหรับการทดสอบประสิทธิภาพของอัลกอริทึม คือ โปรแกรม MATLAB R2008A

สำหรับข้อมูลที่ใช้ในการทดลอง ได้มาจากการสร้างชุดข้อมูลสังเคราะห์ [2] โดยวิธีการสร้างชุดข้อมูลสังเคราะห์แสดงดัง ภาคผนวก ก. เพื่อใช้ในการทดลองสำหรับวัดประสิทธิภาพของการทำงานของอัลกอริทึมของงานวิจัยนี้ และอัลกอริทึมต่างๆที่นำมาเพื่อใช้เปรียบเทียบในการวัดประสิทธิภาพ โดยการสร้างชุดข้อมูลสังเคราะห์จะมีการกำหนดค่าพารามิเตอร์สำหรับการสร้างชุดสังเคราะห์ต่างๆ แสดงดังตารางที่ 4.1

ตารางที่ 4.1 ค่าพารามิเตอร์สำหรับการสร้างชุดข้อมูลสังเคราะห์ สัญลักษณ์

สัญลักษณ์	ความหมาย
|D|	จำนวนของทรานแซคชั่นในฐานข้อมูล
|T|	ค่าเฉลี่ยจำนวนไอเท็มต่อทรานแซคชั่น
|I|	ค่าเฉลี่ยขนาดสูงสุดชุดรูปแบบความสัมพันธ์ของไอเท็มที่จะเป็น Large itemset
L	จำนวนรูปแบบความสัมพันธ์ที่จะเป็น Large itemset
N	จำนวนไอเท็ม
M	จำนวนแอททริบิวต์หลัก
O	ค่าความแตกต่างกันของแอททริบิวต์หลัก

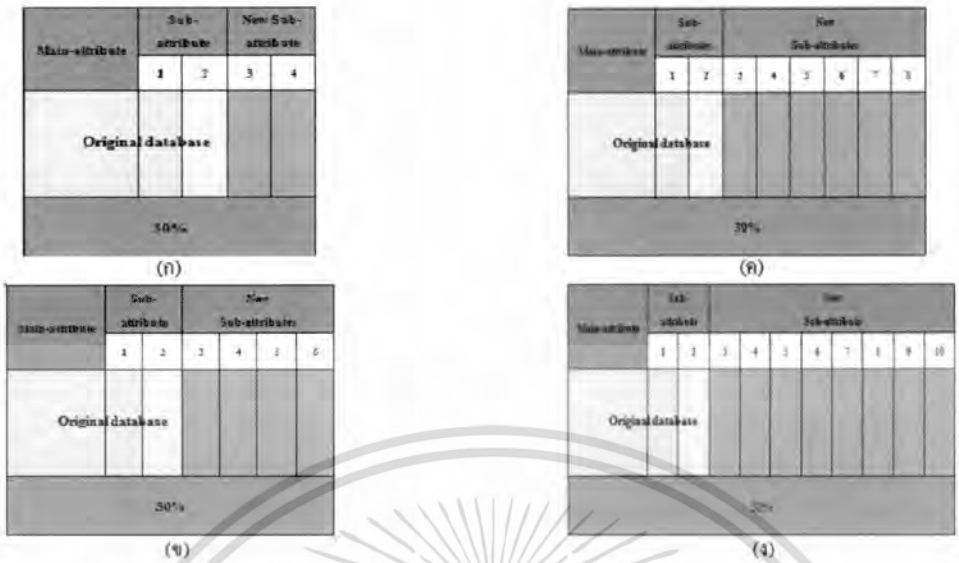
4.3 วิธีการทดลอง

ในการทดลองความถูกต้องและประสิทธิภาพของอัลกอริทึมการค้นหากฎความสัมพันธ์แบบเพิ่มขยายสำหรับการเพิ่มข้อมูลที่มีแอททริบิวต์ใหม่ ผู้วิจัยได้ทดลองโดยอาศัยชุดข้อมูลสังเคราะห์ 3 ชุด โดยค่าพารามิเตอร์ที่กำหนดให้ข้อมูลแต่ละชุดแสดงดังตารางที่ 4.2 เพื่อวัดประสิทธิภาพของอัลกอริทึมการค้นหากฎความสัมพันธ์แบบเพิ่มขยายสำหรับการเพิ่มข้อมูลที่มีแอททริบิวต์ใหม่ ในสมมติฐานคือเมื่อมีการเพิ่มข้อมูลของแอททริบิวต์รองขนาดต่างๆ กัน กับข้อมูลทรานแซคชั่นที่มีขนาดคงที่ เพื่อทดสอบว่าอัลกอริทึมการค้นหากฎความสัมพันธ์แบบเพิ่มขยายสำหรับการเพิ่มข้อมูลที่มีแอททริบิวต์ใหม่ สามารถทำงานได้อย่างถูกต้องและเร็วกว่าอัลกอริทึมที่นำมาเปรียบเทียบเมื่อมีการเพิ่มข้อมูลแอททริบิวต์รองขนาดต่างๆ กัน

ลักษณะของข้อมูลในการทดลองกำหนดให้ มีฐานข้อมูลเดิมจำนวน 10,000 ทรานแซคชั่น ซึ่งในฐานข้อมูลเดิม ประกอบด้วยข้อมูล แอททริบิวต์รอง 2 แอททริบิวต์ และ แอททริบิวต์หลัก จำนวน 1 แอททริบิวต์ ซึ่งการทดลองจะทำการเพิ่มทรานแซคชั่นใหม่ จำนวนคงที่ คือ 30% ของฐานข้อมูลเดิม และสำหรับการเพิ่มแอททริบิวต์รองใหม่จะเป็นการการเพิ่มขึ้นในลักษณะขนาดต่างๆกัน คือ 2, 4, 6 และ 8 ดังรูปที่ 4.1(ก), ดังรูปที่ 4.1(ข), ดังรูปที่ 4.1(ค) และ ดังรูปที่ 4.1(ค) ตามลำดับ ที่ค่าสนับสนุนขั้นต่ำเท่ากับ 4%, 8% และ 12%

ตารางที่ 4.2 ค่าพารามิเตอร์ที่กำหนดสำหรับชุดข้อมูลสังเคราะห์ที่ใช้ในการทดลอง

ชุดข้อมูลที่	|D|	|T|	|I|	L	N	M	O
2.1	15,000	4	10	100	50	10	10
2.2	15,000	10	4	100	50	10	10
2.3	15,000	10	10	100	50	10	10



รูปที่ 4.1 แสดงลักษณะของการเพิ่มข้อมูลการทดลองที่ 2

4.4 ผลการทดลอง

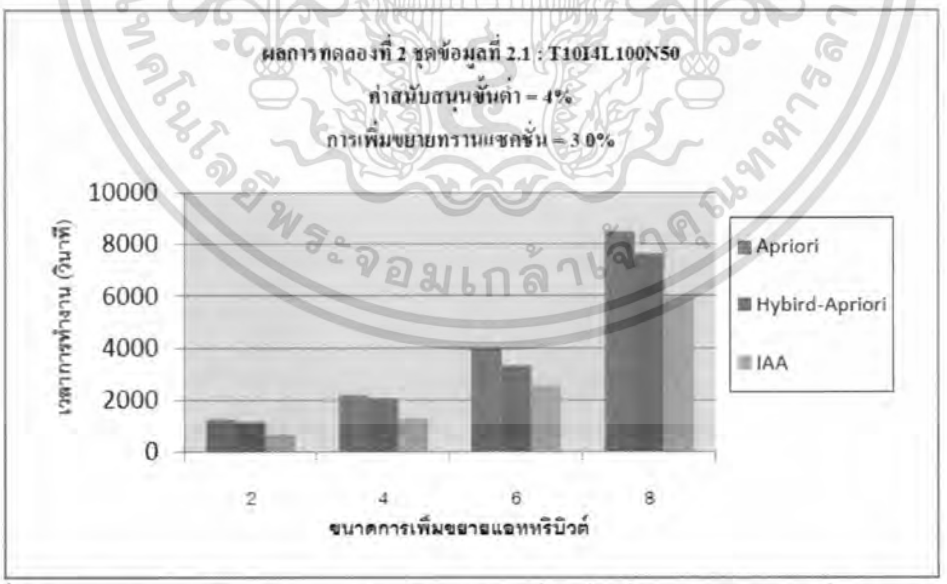
ผลการทดลองในงานวิจัยนี้ ประกอบด้วยผลการทดลองที่วัดจากประสิทธิภาพการทำงานของอัลกอริทึมทางด้านเวลา (Execution time) จำนวน Large itemsets ทั้งหมด และกราฟแสดงผลการเปรียบเทียบเวลาในการทำงานระหว่างอัลกอริทึมอะพริโอริ อัลกอริทึมการค้นหากฎความสัมพันธ์หลายมิติแบบมิติผสม และอัลกอริทึมการค้นหากฎความสัมพันธ์แบบเพิ่มขยายสำหรับการเพิ่มข้อมูลที่มีแอททริบิวต์ใหม่ (เขียนแทนด้วย increment attribute) ที่มีการเพิ่มทรานแซคชั่นใหม่คงที่ขนาดข้อมูล 30% และแอททริบิวต์รองใหม่จำนวน 2,4,6 และ 8 แอททริบิวต์ ด้วยค่าสนับสนุนขั้นต่ำ 4%, 8% และ 12% กับชุดข้อมูลที่ 1 , 2 และ 3

■ ชุดข้อมูลที่ 1 T4I10L100N50

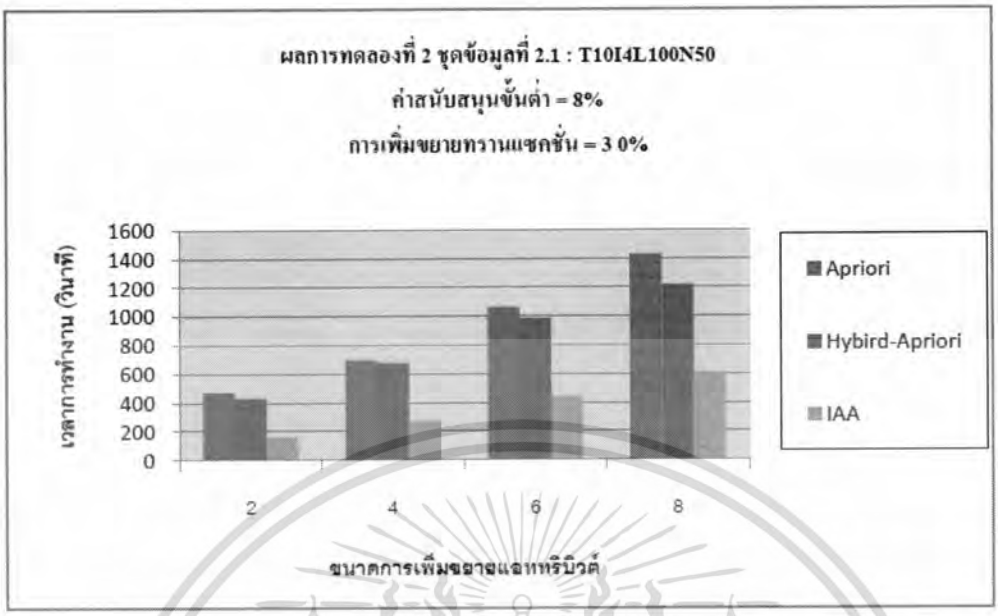
ผลการทดลองของชุดข้อมูลที่ 1 แสดงดังตารางที่ 4.3 ซึ่งแสดงการเปรียบเทียบเวลาการทำงานและจำนวนของ Large itemset ที่ค่าสนับสนุนขั้นต่ำ 4%, 8% และ 12% โดยรูปที่ 4.2, 4.3 และ 4.4 แสดงผลการเปรียบเทียบทางด้านเวลา เมื่อมีการเพิ่มขนาดทรานแซคชั่นใหม่คงที่ขนาดข้อมูล 30% กับการเพิ่มแอททริบิวต์รองใหม่ขนาด 2,4,6 และ 8 แอททริบิวต์ ด้วยค่าสนับสนุนขั้นต่ำที่ 4%, 8% และ 12% ตามลำดับ

ตารางที่ 4.3 แสดงผลการทดลองข้อมูลชุดที่ 1 T4I10L100N50 เพิ่มทรานแซคชั่นใหม่ 30% กับแอททริบิวต์รองใหม่ 2,4,6 และ 8 แอททริบิวต์ ที่ค่าสนับสนุนขั้นต่ำ 4%, 8% และ 12%

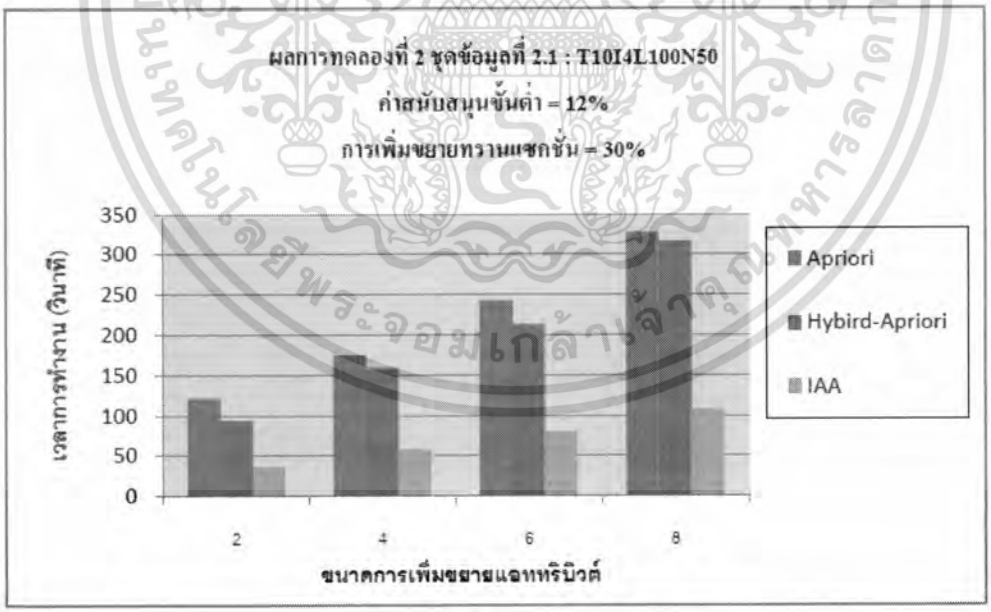
ค่าสนับสนุนขั้นต่ำ	ขนาดการเพิ่มขยาย		เวลาการทำงาน (sec)			Large itemset
	ทรานแซคชั่น	แอททริบิวต์	Apriori	Hybrid-Apriori	IAA	
4%	30%	2	1,243.990	1,158.904	671.494	193
		4	2,196.843	2,050.847	1,292.418	521
		6	3,927.187	3,297.589	2,552.542	1,545
		8	8,470.482	7,624.636	5,949.955	5,568
8%	30%	2	486.800	432.741	161.161	54
		4	693.038	677.675	271.675	73
		6	1,062.679	983.879	445.066	94
		8	1,426.056	1,117.023	605.262	119
12%	30%	2	121.185	93.453	36.138	21
		4	174.208	158.718	57.890	27
		6	242.723	212.830	79.475	33
		8	327.228	313.895	107.678	39



รูปที่ 4.2 แสดงผลการเปรียบเทียบเวลาสำหรับชุดข้อมูลที่ 1 เมื่อเพิ่มขนาดทรานแซคชั่น 30% กับแอททริบิวต์รองใหม่ขนาด 2,4,6 และ 8 แอททริบิวต์ ด้วยค่าสนับสนุนขั้นต่ำ 4%



รูปที่ 4.3 แสดงผลการเปรียบเทียบเวลาสำหรับชุดข้อมูลที่ 1 เมื่อเพิ่มขนาดทรานแซคชั่น 30% กับแอททริบิวต์รองใหม่ขนาด 2,4,6 และ 8 แอททริบิวต์ ด้วยค่าสนับสนุนขั้นต่ำ 8%



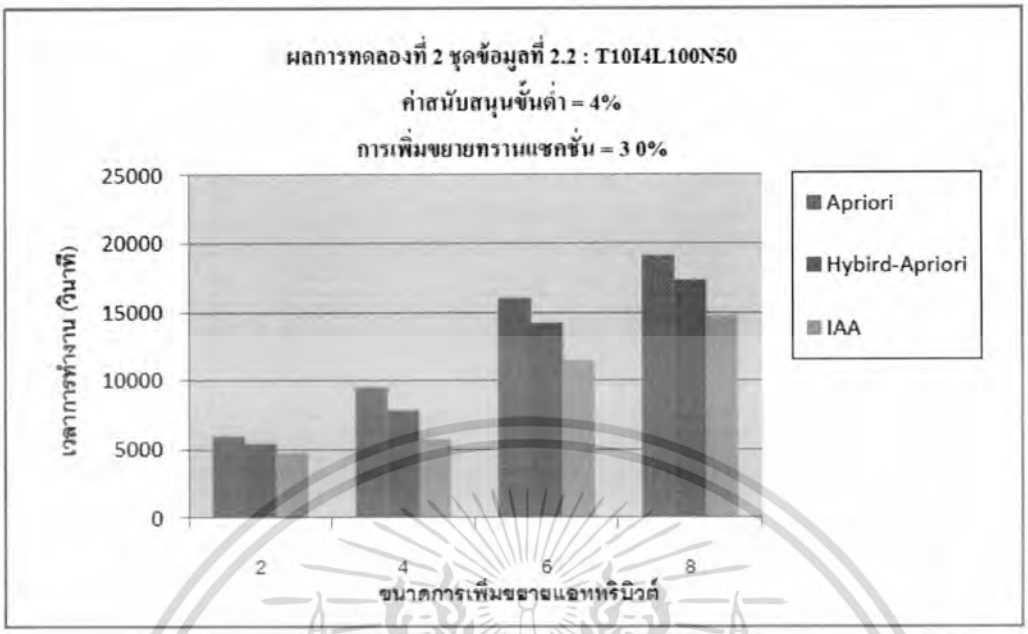
รูปที่ 4.4 แสดงผลการเปรียบเทียบเวลาสำหรับชุดข้อมูลที่ 1 เมื่อเพิ่มขนาดทรานแซคชั่น 30% กับแอททริบิวต์รองใหม่ขนาด 2,4,6 และ 8 แอททริบิวต์ ด้วยค่าสนับสนุนขั้นต่ำ 12%

■ **ชุดข้อมูลที่ 2 T10I4L100N50**

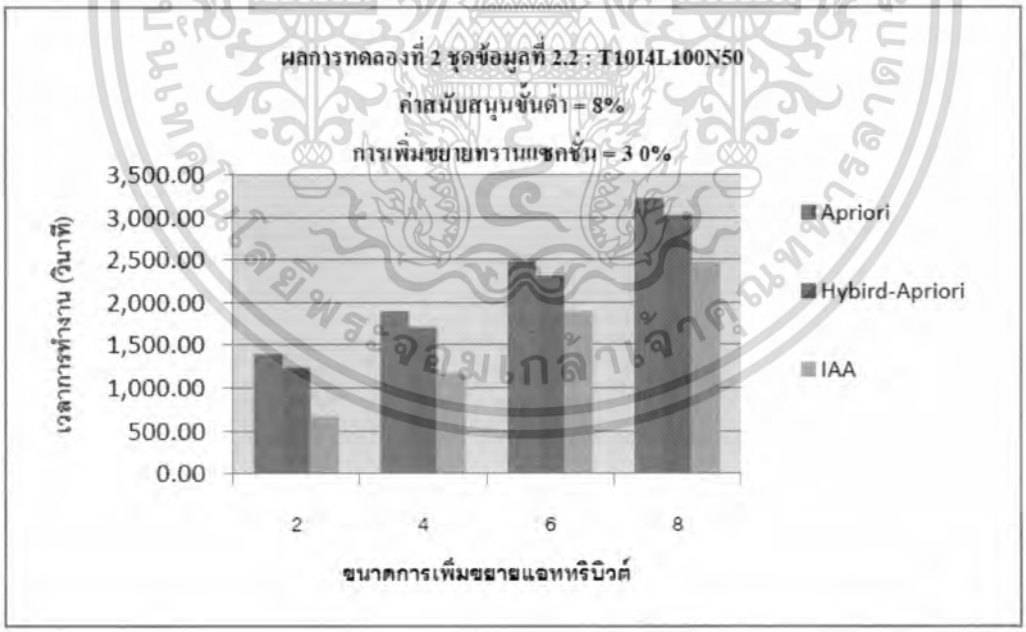
ผลการทดลองของชุดข้อมูลที่ 2 แสดงดังตารางที่ 4.4 ซึ่งแสดงการเปรียบเทียบเวลาการทำงานและจำนวนของ Large itemset ที่ค่าสนับสนุนขั้นต่ำ 4%, 8% และ 12% โดยรูปที่ 4.5, 4.6 และ 4.7 แสดงผลการเปรียบเทียบทางด้านเวลา เมื่อมีการเพิ่มขนาดทรานแซคชั่นใหม่คงที่ขนาดข้อมูล 30% กับการเพิ่มแอททริบิวต์รองใหม่ขนาด 2,4,6 และ 8 แอททริบิวต์ ด้วยค่าสนับสนุนขั้นต่ำที่ 4%, 8% และ 12% ตามลำดับ

ตารางที่ 4.4 แสดงผลการทดลองข้อมูลชุดที่ 1 T10I4L100N50 เพิ่มทรานแซคชั่นใหม่ 30% กับแอททริบิวต์รองใหม่ 2,4,6 และ 8 แอททริบิวต์ ที่ค่าสนับสนุนขั้นต่ำ 4%, 8% และ 12%

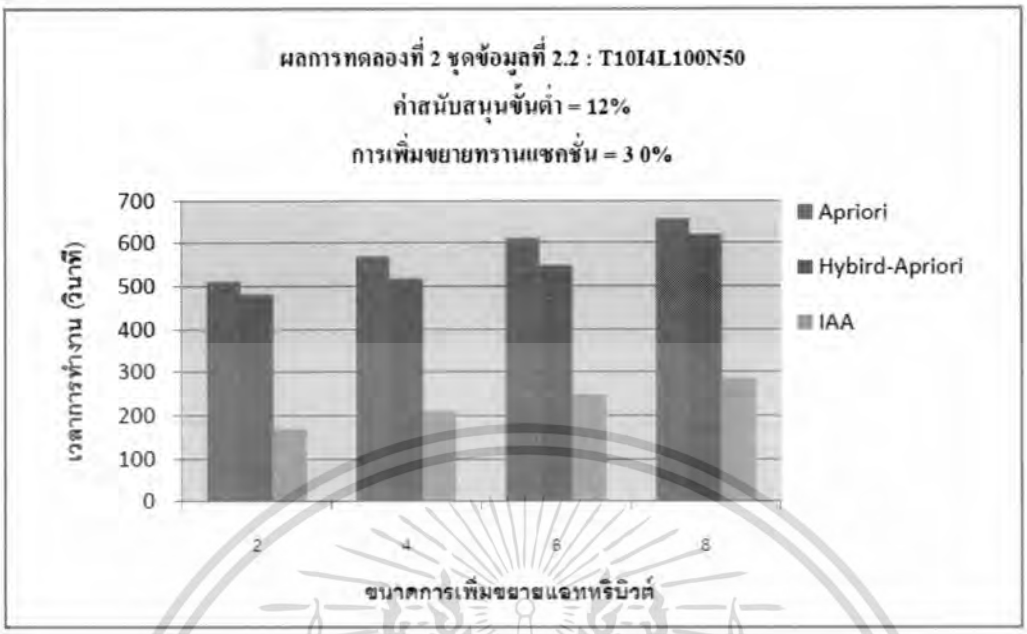
ค่าสนับสนุนขั้นต่ำ	ขนาดการเพิ่มขยาย		เวลาการทำงาน (sec)			Large itemset
	ทรานแซคชั่น	แอททริบิวต์	Apriori	Hybrid-Apriori	IAA	
4%	30%	2	5,970.167	5,083.457	4,740.233	1,677
		4	9,549.889	7,749.561	5,714.970	1,817
		6	16,004.363	9,204.385	6,976.237	1,909
		8	19,100.817	11,318.404	8,223.877	2,050
8%	30%	2	1,394.561	1,235.561	665.570	349
		4	1,892.535	1,711.322	1,262.747	368
		6	2,484.650	2,315.561	1,991.917	386
		8	3,221.656	3,023.279	2,493.633	407
12%	30%	2	511.315	501.168	169.530	113
		4	568.402	547.769	210.089	116
		6	612.118	608.156	247.986	118
		8	655.778	629.914	284.503	120



รูปที่ 4.5 แสดงผลการเปรียบเทียบเวลาสำหรับชุดข้อมูลที่ 2 เมื่อเพิ่มขนาดทรานแซคชั่น 30% กับแอททริบิวต์รองใหม่ขนาด 2,4,6 และ 8 แอททริบิวต์ ด้วยค่าสนับสนุนขั้นต่ำ 4%



รูปที่ 4.6 แสดงผลการเปรียบเทียบเวลาสำหรับชุดข้อมูลที่ 2 เมื่อเพิ่มขนาดทรานแซคชั่น 30% กับแอททริบิวต์รองใหม่ขนาด 2,4,6 และ 8 แอททริบิวต์ ด้วยค่าสนับสนุนขั้นต่ำ 8%



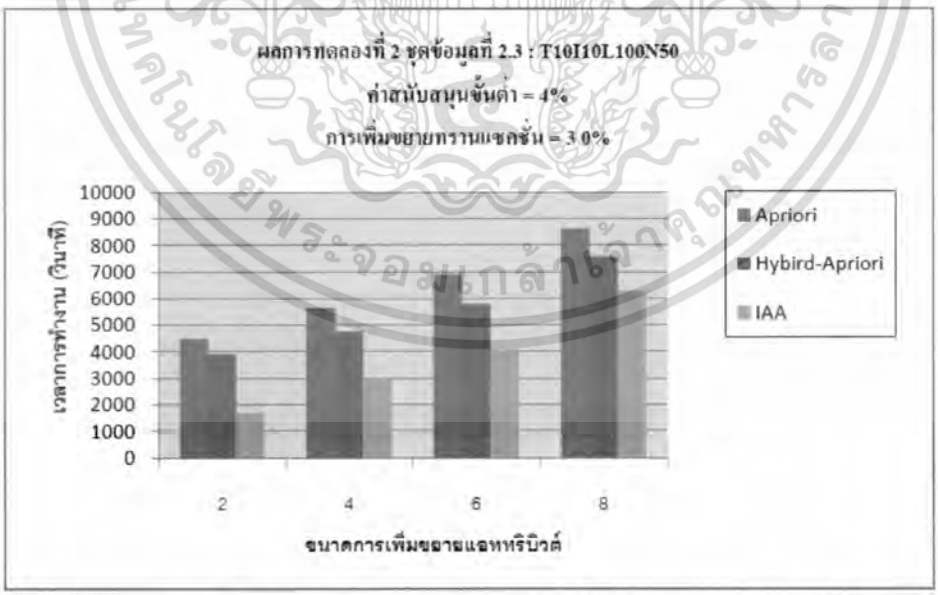
รูปที่ 4.7 แสดงผลการเปรียบเทียบเวลาสำหรับชุดข้อมูลที่ 2 เมื่อเพิ่มขนาดทรานแซคชั่น 30% กับแอททริบิวต์รองใหม่ขนาด 2,4,6 และ 8 แอททริบิวต์ ด้วยค่าสนับสนุนขั้นต่ำ 12%

■ **ชุดข้อมูลที่ 3 T10I10L100N50**

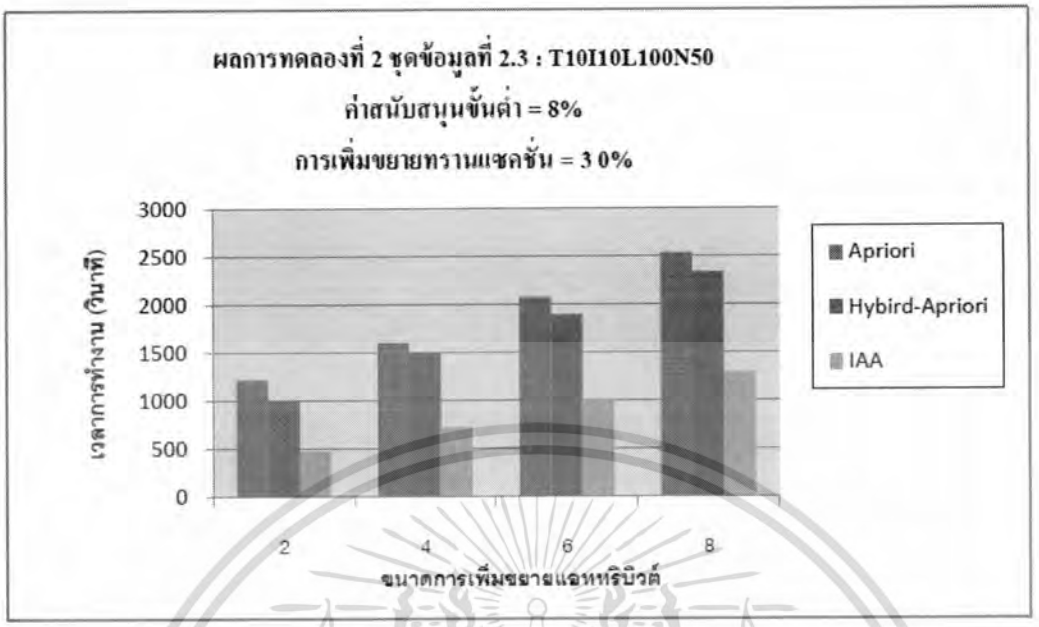
ผลการทดลองของชุดข้อมูลที่ 3 แสดงดังตารางที่ 4.5 ซึ่งแสดงการเปรียบเทียบเวลาการทำงานและจำนวนของ Large itemset ที่ค่าสนับสนุนขั้นต่ำ 4%, 8% และ 12% โดยรูปที่ 4.8, 4.9 และ 4.10 แสดงผลการเปรียบเทียบทางด้านเวลา เมื่อมีการเพิ่มขนาดทรานแซคชั่นใหม่คงที่ขนาดข้อมูล 30% กับการเพิ่มแอททริบิวต์รองใหม่ขนาด 2,4,6 และ 8 แอททริบิวต์ ด้วยค่าสนับสนุนขั้นต่ำที่ 4%, 8% และ 12% ตามลำดับ

ตารางที่ 4.5 แสดงผลการทดลองข้อมูลชุดที่ 3 T10I10L100N50 เพิ่มทรานแซคชั่นใหม่ 30% กับแอททริบิวต์รองใหม่ 2,4,6 และ 8 แอททริบิวต์ ที่ค่าสนับสนุนขั้นต่ำ 4%, 8% และ 12%

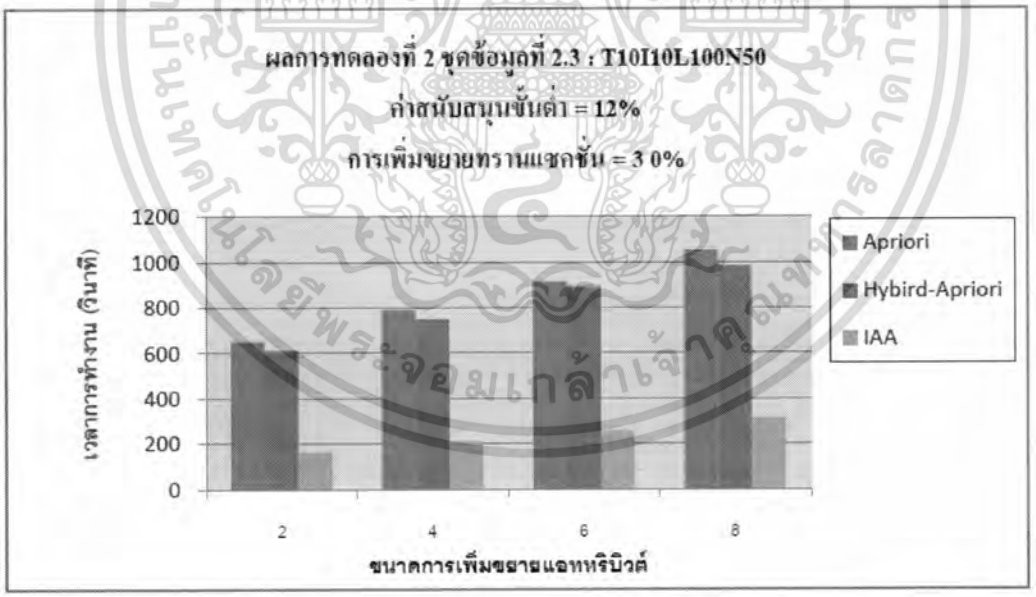
ค่าสนับสนุนขั้นต่ำ	ขนาดการเพิ่มขยาย		เวลาการทำงาน (sec)			Large itemset
	ทรานแซคชั่น	แอททริบิวต์	Apriori	Hybrid-Apriori	IAA	
4%	30%	2	4,472.267	3,919.449	1,678.635	758
		4	5,647.955	4,761.397	2,977.964	829
		6	6,928.837	5,804.125	4,089.794	911
		8	8,654.519	7,206.334	6,270.616	1,091
8%	30%	2	1,221.223	1,013.627	477.801	136
		4	1,602.602	1,493.860	724.349	148
		6	2,079.831	1,897.831	1,011.043	161
		8	2,540.383	2,338.190	1,294.239	172
12%	30%	2	650.756	610.276	164.413	58
		4	790.850	750.042	211.793	64
		6	913.407	893.570	254.617	69
		8	1,052.528	983.528	314.322	74



รูปที่ 4.8 แสดงผลการเปรียบเทียบเวลาสำหรับชุดข้อมูลที่ 3 เมื่อเพิ่มขนาดทรานแซคชั่น 30% กับแอททริบิวต์รองใหม่ขนาด 2,4,6 และ 8 แอททริบิวต์ ด้วยค่าสนับสนุนขั้นต่ำ 4%



รูปที่ 4.9 แสดงผลการเปรียบเทียบเวลาสำหรับชุดข้อมูลที่ 3 เมื่อเพิ่มขนาดทรานแซคชั่น 30% กับแอททริบิวต์รองใหม่ขนาด 2,4,6 และ 8 แอททริบิวต์ ด้วยค่าสนับสนุนขั้นต่ำ 8%



รูปที่ 4.10 แสดงผลการเปรียบเทียบเวลาสำหรับชุดข้อมูลที่ 3 เมื่อเพิ่มขนาดทรานแซคชั่น 30% กับแอททริบิวต์รองใหม่ขนาด 2,4,6 และ 8 แอททริบิวต์ ด้วยค่าสนับสนุนขั้นต่ำ 12%

4.5 วิเคราะห์ผลการทดลอง

สมมติฐานของการทดลองคือ อัลกอริทึมการค้นหากฎความสัมพันธ์แบบเพิ่มขยายสำหรับการเพิ่มข้อมูลที่มีแอททริบิวต์ใหม่มีประสิทธิภาพและความถูกต้องเมื่อมีการเพิ่มข้อมูลในกรณีที่มีการเพิ่มข้อมูลทรานแซคชั่นที่มีขนาดคงที่กับการเพิ่มแอททริบิวต์รองขนาดต่างๆ กัน

ในการทดลองครั้งนี้เป็นการทดลองกับข้อมูลที่มีการเพิ่มทรานแซคชั่นใหม่เข้ามาขนาดคงที่เท่ากับ 30% ของฐานข้อมูลเดิม และการเพิ่มขนาดของข้อมูลของแอททริบิวต์ใหม่เท่ากับ 2,4,6 และ 8 แอททริบิวต์ ด้วยค่าสนับสนุนขั้นต่ำ 4%, 8% และ 12% ตามลำดับ กับชุดข้อมูลสังเคราะห์ 3 ชุด ผลการทดลองของชุดข้อมูลที่ 1, 2 และ 3 จากตารางที่ 4.3, 4.4 และ 4.5 พบว่า เวลาการทำงานของอัลกอริทึมการค้นหากฎความสัมพันธ์แบบเพิ่มขยายสำหรับการเพิ่มข้อมูลที่มีแอททริบิวต์ใหม่สามารถทำงานได้เร็วกว่าอัลกอริทึมอะพริโอริ และอัลกอริทึมการค้นหากฎความสัมพันธ์หลายมิติแบบมิติผสม รวมถึงจำนวนของ Large itemsets ที่ได้จากทั้ง 3 อัลกอริทึมมีจำนวนที่เท่ากัน จึงสามารถสรุปได้ว่า อัลกอริทึมในงานวิจัยนี้สามารถทำงานได้อย่างมีประสิทธิภาพและมีความถูกต้องในการทำงาน เมื่อมีการเพิ่มข้อมูลในกรณีที่มีการเพิ่มข้อมูลทรานแซคชั่นที่มีขนาดคงที่กับการเพิ่มแอททริบิวต์รองขนาดต่างๆ กัน

บทที่ 5
สรุปผลการทดลองและข้อเสนอแนะ

ในบทนี้จะกล่าวถึงผลสรุปที่ได้จากการทำงานของอัลกอริทึมการค้นหากฎความสัมพันธ์แบบเพิ่มขยายสำหรับการเพิ่มข้อมูลที่มีแอททริบิวต์ใหม่และกล่าวถึงประสิทธิภาพของอัลกอริทึม รวมทั้งข้อเสนอแนะสำหรับผู้ที่สนใจที่จะนำอัลกอริทึมการค้นหากฎความสัมพันธ์แบบเพิ่มขยายสำหรับการเพิ่มข้อมูลที่มีแอททริบิวต์ใหม่เพื่อจะได้เป็นแนวทางในการศึกษาหรือพัฒนาต่อไป

5.1 สรุปผลการวิจัย

การค้นหากฎความสัมพันธ์ (Association rule discovery) เป็นการค้นหารูปแบบความสัมพันธ์ที่ซ่อนอยู่ในฐานข้อมูลขนาดใหญ่ โดยความสัมพันธ์ที่ได้ดังกล่าวจะอยู่ในรูปแบบของกฎความสัมพันธ์ ซึ่งสามารถนำกฎความสัมพันธ์ที่ได้มาช่วยตัดสินใจหรือวางแผนด้านการบริหารขององค์กร การค้นหากฎความสัมพันธ์โดยงานวิจัยส่วนใหญ่จะเป็นการค้นหาความสัมพันธ์ของข้อมูลทรานแซคชั่นของมิติการซื้อขาย แต่การเก็บข้อมูลในฐานข้อมูลมีการจัดเก็บข้อมูลที่เป็นแบบหลายมิติ ทำให้การค้นหากฎความสัมพันธ์มิติเดียวเป็นการค้นหากฎความสัมพันธ์ที่ไม่สอดคล้องกับการลักษณะของการจัดเก็บข้อมูลในฐานข้อมูล และประกอบกับลักษณะข้อมูลที่ใช้ในการค้นหาความสัมพันธ์นั้นไม่จำเป็นการเพิ่มขึ้นของข้อมูลจะเป็นการเพิ่มเฉพาะข้อมูลทรานแซคชั่นแต่อาจมีการเพิ่มขึ้นของข้อมูลแอททริบิวต์รองด้วย ดังนั้นเพื่อให้สอดคล้องกับการเพิ่มขึ้นของข้อมูลที่ใช้ในการค้นหาความสัมพันธ์ของข้อมูลที่เป็นทั้งข้อมูลทรานแซคชั่นและการเพิ่มขึ้นของข้อมูลแอททริบิวต์รอง และให้สามารถค้นหาความสัมพันธ์ได้อย่างครบถ้วนและถูกต้องของทุกความสัมพันธ์ที่สามารถเกิดขึ้นได้

งานวิจัยนี้จึงได้นำเสนออัลกอริทึมการค้นหากฎความสัมพันธ์แบบเพิ่มขยายสำหรับการเพิ่มข้อมูลที่มีแอททริบิวต์ใหม่ โดยมีการนำเอาแนวคิดของอัลกอริทึม HDFUP มาปรับปรุงใช้ให้มีความสามารถค้นหารูปแบบกฎความสัมพันธ์หลายมิติแบบมิติผสมเมื่อมีการเปลี่ยนแปลงของข้อมูลในการค้นหากฎความสัมพันธ์ในลักษณะของการเพิ่มขึ้นของข้อมูลที่เป็นทั้งข้อมูลทรานแซคชั่นและข้อมูลแอททริบิวต์รองลงในข้อมูลที่ใช้ในการหาความสัมพันธ์เดิมพร้อมกัน

หลักการทำงานของอัลกอริทึมที่นาเสนอจะแบ่งออกเป็น 3 ส่วนก็คือ การทำงานส่วนแรกเริ่มจากการทำงานในส่วนของแอททริบิวต์รองที่เพิ่มขึ้นโดยการหา Large itemset ของแอททริบิวต์รองที่เพิ่มขึ้นโดยใช้หลักการค้นหากฎความสัมพันธ์หลายมิติแบบมิติผสม สำหรับส่วนที่ 2 เป็นการทำงานของการหา Large itemset ที่น่าจะมีความสัมพันธ์ใหม่ที่เกิดขึ้นในส่วนของแอททริบิวต์รองที่เพิ่มขึ้นกับข้อมูลที่ใช้หาความสัมพันธ์เดิม ขั้นตอนการทำงานส่วนนี้เป็นการนำเอา Large itemset ของฐานข้อมูลเดิมมาใช้เพื่อลดการหาความสัมพันธ์ที่เกิดขึ้นแล้วภายในข้อมูลเดิม และใช้การประมาณค่าสนับสนุนที่เป็นไปได้ที่ดีที่สุดของ Large itemset ของความสัมพันธ์ที่เกิดใหม่เพื่อประมาณค่าสนับสนุนที่เป็นไปได้ของ Large itemset จากขั้นตอนการทำงานส่วนนี้เพื่อลดการค้นหาค่าสนับสนุนของ Large itemset และในการทำงานส่วนที่ 3 โดยขั้นตอนการทำงานนี้มีหลักการทำงานที่คล้ายกับการทำงานของอัลกอริทึม HDFUP ซึ่งเป็นขั้นตอนการหา Large itemset ของข้อมูลที่ถูกปรับปรุงทั้งหมดเมื่อมีข้อมูลทรานแซคชั่นใหม่เพิ่มขึ้น โดยนำเอา Large itemset จากการทำงานส่วนที่ 2 มาใช้เพื่อช่วยในการลดการค้นหาความสัมพันธ์ที่เคยเกิดขึ้นแล้วในส่วนของข้อมูลเดิม

จากผลการทดลองในบทที่ 4 เมื่อพิจารณาผลการทดลองจะเห็นได้ว่าอัลกอริทึมการค้นหากฎความสัมพันธ์แบบเพิ่มขยายสำหรับการเพิ่มข้อมูลที่มีแอททริบิวต์ใหม่ ที่ได้ทำการพัฒนาขึ้นมานี้สามารถทำการหารูปแบบของไอเท็มเซตที่น่าสนใจ (Large itemset) ได้รวดเร็วกว่า อัลกอริทึมอะพริโอริ และอัลกอริทึมสำหรับการค้นหากฎความสัมพันธ์หลายมิติแบบมิติผสม และยังสามารถค้นหากฎความสัมพันธ์ได้ถูกต้องครบถ้วนเท่ากับอัลกอริทึมเหล่านั้นอีกด้วย ซึ่งจากผลลัพธ์ที่ได้จึงทำให้สรุปได้ว่า การค้นหาไอเท็มเพื่อทำการนับค่าสนับสนุนในข้อมูล มีผลต่อความเร็วในการทำงานของอัลกอริทึม โดยในการทำงานอัลกอริทึมอะพริโอริ และ อัลกอริทึมการค้นหาความสัมพันธ์แบบมิติผสมจะไม่มีการนำ Large itemset เดิมมาใช้ในการช่วยในการหาความสัมพันธ์ในการเพิ่มขึ้นของข้อมูลจึงต้องทำการค้นหาทั้งข้อมูลที่ปรับปรุงใหม่ทั้งหมดจึงทำให้ใช้เวลามากในการค้นหาค่าสนับสนุนของไอเท็มเซตในแต่ละรอบการทำงาน แต่อัลกอริทึมการค้นหากฎความสัมพันธ์แบบเพิ่มขยายสำหรับการเพิ่มข้อมูลที่มีแอททริบิวต์ใหม่ ในการทำงานส่วนที่ 2 ซึ่งเป็นการทำงานในส่วนของ AUD มีการนำหลักการโดยทั่วไปของการเพิ่มขยายกฎความสัมพันธ์ ที่เป็นการนำเอา Large itemset ของเดิมมาใช้ประโยชน์เพื่อลดจำนวนการค้นหาของ Large itemset ที่จะต้องไปนับค่าสนับสนุนในข้อมูลเดิมที่ได้ค้นหาไว้ก่อนหน้าแล้ว และ ใช้การประมาณค่าสนับสนุนของไอเท็มเซตในส่วนของไอเท็มเซตที่ได้จากความสัมพันธ์ใหม่ที่ไม่เคยเกิดขึ้นมาก่อน เพื่อลดการค้นหาเพื่อทำการนับค่าสนับสนุนที่ต้องทำการค้นหาในส่วนของข้อมูลเดิม และการทำงานในส่วนที่ 3 เป็นการนำเอา Large itemset จากส่วนที่ 2 มาใช้ซึ่งประกอบด้วยไอเท็มเซตที่มาจากฐานข้อมูลเดิม ทำให้ลดการค้นหาในข้อมูลเดิม เพื่อให้ทำการค้นหาเพื่อนับค่าสนับสนุนในข้อมูลบางส่วนนั้นคือส่วนทรานแซคชั่นที่เพิ่มขึ้น ซึ่งบางครั้งค่าสนับสนุนที่เป็นค่าประมาณ ได้มาจากค่าที่เป็นไปได้ที่ดีที่สุดของการเกิด

ความสัมพันธ์ เมื่อมาค้นหาในส่วนของทรานแซคชั่นที่เพิ่มขึ้นแล้วค่าสนับสนุนที่ได้ ไม่ผ่านค่าสนับสนุนขั้นต่ำของฐานข้อมูลที่ปรับปรุง จึงสามารถตัดไอเท็มเซตนั้นทิ้งเพื่อลดเวลาในการค้นหาไอเท็มเซตเหล่านี้เพื่อหาค่าสนับสนุนจริงในฐานข้อมูลเดิมอีกครั้ง การเชื่อมเพื่อหาความสัมพันธ์มีผลต่อความเร็วในการหา Large itemset เนื่องจากความแตกต่างระหว่างวิธีการการเชื่อมเพื่อสร้างความสัมพันธ์ระหว่างไอเท็มเซต

5.2 ข้อเสนอแนะ

สำหรับการทำงานวิจัยเรื่องนี้ต่อไป ผู้วิจัยจึงเสนอข้อเสนอแนะต่างๆเพื่อที่จะได้เป็นประโยชน์ต่อการทำงานวิจัยนี้ต่อไปในอนาคต

1. การทดลองของอัลกอริทึมเป็นการทดลองกับชุดข้อมูลสังเคราะห์ ในความเป็นจริงควรจะมีการนำเอา ข้อมูลการซื้อ-ขายสินค้าและข้อมูลของลูกค้าที่ซื้อขายสินค้าจริงที่เกิดขึ้นเข้ามาทดสอบ เพื่อให้เห็นประสิทธิภาพการทำงานในสภาพแวดล้อมข้อมูลจริง

2. ในส่วนของการประมาณค่าสนับสนุนให้กับไอเท็มเซต ควรจะมีการนำหลักการทางสถิติเข้ามาใช้เพื่อประมาณค่าสนับสนุนให้ใกล้เคียงค่าสนับสนุนที่เป็นจริงของไอเท็มเซตนั้นๆ

บรรณานุกรม

[1] Agrawal, R., Imielinski, T., and Swami, A. "Mining association rules between sets of items in large database." **Proceeding of the 1993 ACM SIGMOD Conference on Washington DC**, USA, May 1993

[2] Agrawal, R., and Srikant, R., "Fast algorithm for mining association rules." Proceedings of 20th VLDB Conference Santiago. Chile, 1994. pp 487-499.

[3] Amornchewin, R., and Kreesuradej, W., "Mining Dynamic Database using Probability-Based Incremental Association Rule Discovery Algorithm." **Journal of Universal Computer Science**, pp. 2409-2428. Vol 15, no. 12, 2009.

[4] Chatsettakul, S., and Kreesuradaj, W., "Hybrid-dimension Fast Update Algorithm" Proceeding of 24 th International Technical Conference on Circuits Computer and Communication, Jeju, Korea, 2009.

[5] Cheung, D.W., Han, J., Ng, V.T. and Wong, C.Y., "Maintenance of Discovered Association Rule in Large Database: An incremental updating technique" **In 12 th IEEE International Conference on Data Engineering, 1996.**

[6] Han, J. and Kamber, M. 2006. **Data Mining: Concepts and Techniques.** 2nd ed. San Francisco : Morgan Kaufmann Publishers.

[7] Teng, W. –G. and Chen, M.-S. "Incremental Mining on Association Rule." **Foundations and Advances in Data Mining**, pp.125-162, edited by W. Chu and T.-Y. Lin, Springer, 2005.

[8] Yan, X., and Shi-G. Ju., " Mining Conditional Hybrid-dimension Association Rule on the basic of Multidimensional Transaction Database", Proceeding of 2 nd International Conference on Machine Learning and Cybernetics, November 2003. pp 216-221.

ภาคผนวก

ภาคผนวก ก.

การสร้างชุดข้อมูลในการทดลอง

การสร้างข้อมูลสังเคราะห์

(Synthetic dataset)

การสร้างชุดข้อมูลสังเคราะห์ (Synthetic dataset) นำเสนอแนวคิดโดย Agrawal et.al. [2] ซึ่งได้นำเสนอวิธีการสร้างชุดข้อมูลเพื่อใช้สำหรับการประเมินประสิทธิภาพของอัลกอริทึมสำหรับงานการค้นหากฎความสัมพันธ์ของข้อมูลในฐานข้อมูล ซึ่งลักษณะของข้อมูลสังเคราะห์เป็นข้อมูลขนาดใหญ่ที่เกิดขึ้นในทรานแซคชั่นที่เลียนแบบพฤติกรรมการซื้อขายสินค้า โดยใช้หลักการทางสถิติต่างๆ ได้แก่ การสุ่มค่าความน่าจะเป็นของขนาดของไอเท็มเซตและขนาดของทรานแซคชั่น รวมถึงการสุ่มค่าของไอเท็มที่จะนำไปใส่ในแต่ละทรานแซคชั่นด้วยการแจกแจงแบบต่างๆ เช่น การแจกแจงแบบปกติ การแจกแจงแบบยูนิฟอร์ม การแจกแจงแบบเอ็กซ์โพเนลเชียล เป็นต้น

โมเดลของชุดข้อมูลสังเคราะห์นี้จะแสดงแนวโน้มของเซตของสินค้าที่มักมีการซื้อไปด้วยกัน เซตของสินค้าที่ซื้อไปด้วยกันแสดงในรูปของจำนวนที่สามารถเป็นชุดความสัมพันธ์ของไอเท็มเซตสูงสุด(Potentially a maximal frequent itemset) ในการสร้างชุดข้อมูลสังเคราะห์จะประกอบด้วยพารามิเตอร์ที่สำคัญต่างๆ ที่แสดงในตาราง ก.1

ตารางที่ ก.1 พารามิเตอร์ในการสร้างข้อมูลสังเคราะห์

สัญลักษณ์	ความหมาย
|D|	จำนวนของทรานแซคชั่นในฐานข้อมูล
|T|	ค่าเฉลี่ยจำนวนไอเท็มต่อทรานแซคชั่น
|I|	ค่าเฉลี่ยขนาดสูงสุดชุดรูปแบบความสัมพันธ์ของไอเท็มที่จะเป็น Large itemset
L	จำนวนรูปแบบความสัมพันธ์ที่จะเป็น Large itemset
N	จำนวนไอเท็ม

ขั้นตอนการสร้างชุดข้อมูลสังเคราะห์

ประกอบด้วย 2 ขั้นตอน ดังนี้

1. **ขั้นตอนการสร้างชุดของจำนวนรูปแบบความสัมพันธ์ที่จะเป็น Large itemset**

ในขั้นตอนนี้การสร้างชุดข้อมูลสังเคราะห์ โดยขั้นตอนนี้จะเป็นการสุ่มเพื่อสร้างชุดรูปแบบความสัมพันธ์ที่จะเป็น Large itemset เท่ากับจำนวน |L| ที่ได้กำหนด โดยชุดรูปแบบความสัมพันธ์นี้จะมีอยู่ 2 ส่วนด้วยกัน คือ ส่วนที่เป็นชุดไอเท็มที่เป็นแอททริบิวต์หลัก และ ชุดไอเท็ม

ที่เป็นแอททริบิวต์รอง ดังตัวอย่างแสดงในตาราง ก.2 ซึ่งได้กำหนด | L | = 6 จะเห็นได้ว่าค่าที่ปรากฏจะมีการสุ่มจากการแจกแจงแบบปัวส์ซอง 2 ส่วน คือ สุ่มจำนวนไอเท็มในแอททริบิวต์หลัก และสุ่มจำนวนของแอทริบิวต์ที่มีโอกาสเป็น Large itemset โดยขนาดที่ได้จากการสุ่มด้วยค่าเฉลี่ยขนาดสูงสุดของชุดรูปแบบความสัมพันธ์ของไอเท็มที่จะเป็น Large itemset | I |

ตารางที่ ก.2 การสุ่มขนาดของ | L | ด้วยการแจกแจงแบบปัวส์ซอง

L	จำนวนไอเท็มในแอททริบิวต์หลักที่สุ่มได้จากการแจกแจงแบบปัวส์ซอง	จำนวนแอททริบิวต์รองที่สุ่มได้จากการแจกแจงแบบปัวส์ซอง
L1	5	3
L2	2	4
L3	6	2
L4	5	3
L5	5	2
L6	6	4

เมื่อได้ขนาดที่จะนำมาสร้างรูปแบบความสัมพันธ์ของไอเท็มที่จะเป็น Large itemset แล้วจะมีการสุ่มเลือกไอเท็มสำหรับ L ทั้งหมดโดยการสุ่มเลือกจะแบ่งออกเป็น 2 ส่วน คือ

■ สุ่มเลือกไอเท็มของแอททริบิวต์หลัก

การสุ่มเลือกไอเท็มของแอททริบิวต์หลักสำหรับ L จะเริ่มจาก L_i ด้วยการสุ่มเลือกจากชุดไอเท็มของแอททริบิวต์หลักเท่ากับขนาดของที่สุ่มได้จากการแจกแจงแบบปัวส์ซอง สำหรับ L ถัดไป จะประกอบด้วยไอเท็มบางส่วนจาก L ก่อนหน้าโดยขนาดของการสุ่มจะได้จากการแจกแจงแบบเอ็กซ์โพแนนเชียลด้วยค่าเฉลี่ยเท่ากับค่าระดับความสัมพันธ์ (correlation) ที่ 0.5 ส่วนไอเท็มเซตที่เหลือจะได้จากการสุ่มจากชุดไอเท็มจากแอททริบิวต์หลักที่ไม่ซ้ำกับไอเท็มที่สุ่มจาก L ก่อนหน้า

จากตารางที่ ก.2 L_i จำนวนไอเท็มในแอททริบิวต์หลักที่สุ่มได้จากการแจกแจงแบบปัวส์ซองเท่ากับ 5 จะทำการสุ่มแบบ random ขนาดเท่ากับ 5 ไอเท็ม จากชุดไอเท็มจากแอททริบิวต์หลัก

$$L_i = \{1\ 3\ 8\ 9\ 10\}$$

ตั้งแต่ L2 จะทำการสุ่ม ค่า e จากการสุ่มแบบเอ็กซ์โพแนนเชียลที่ correlation = 0.5 เพื่อสุ่มไอเท็มบางตัวจาก L ก่อนหน้า ถ้าใน L2 ค่า e จากการสุ่มแบบเอ็กซ์โพแนนเชียลเท่ากับ 1

แสดงว่าต้องสุ่มจาก L_1 ขนาด 1 ไอเท็มส่วนสมาชิกที่เหลือจะได้จากการสุ่มจากชุดไอเท็มจากแอททริบิวต์หลัก

$$L2 = \{3\ 4\}$$

จาก L_2 ในตัวอย่างนี้จะพบว่าใน L_2 จะประกอบไปด้วยไอเท็ม 3 ที่ได้มาจาก L_1 โดยจะทำการสุ่มเลือกไอเท็มของแอททริบิวต์หลักสำหรับ L จนครบสำหรับ L ทั้งหมดดังแสดงดังตาราง ก.3

■ สุ่มเลือกไอเท็มของแอททริบิวต์รอง

การสุ่มไอเท็มของแอททริบิวต์รองจะมีการให้ค่าน้ำหนักให้กับแต่ละแอททริบิวต์ M โดยค่าน้ำหนักจะเป็นการบอกถึงว่าแอททริบิวต์ไหนจะถูกหยิบมาใส่ในแต่ละ | L | โดยน้ำหนักจะถูกกำหนดด้วยการแจกแจงเอ็กซ์โพแนนเชียลด้วยค่าเฉลี่ยเท่ากับ 1 และนำค่าน้ำหนักที่ได้มาทำให้เป็นค่าบรรทัดฐาน โดยนำค่าน้ำหนักของแอททริบิวต์รองแต่ละตัว หารด้วยจำนวนรวมของน้ำหนักของแอททริบิวต์รองทั้งหมด

ตารางที่ ก.3 แสดงค่าไอเท็ม ค่าน้ำหนัก และ ค่าบรรทัดฐานภายในแต่ละแอททริบิวต์รอง

แอททริบิวต์รอง	ค่าในแอททริบิวต์รอง	ค่าน้ำหนัก (weight)	ค่าบรรทัดฐาน (normalize)
A	A1, A2, A3	0.014	0.0112
B	B1, B2, B3	0.035	0.0390
C	C1, C2, C3	0.741	0.6897
D	D1, D2, D3	0.371	1

การสุ่มเลือกไอเท็มของแอททริบิวต์รองสำหรับ L จะเริ่มจากชุดแรก ทำการสุ่มหยิบค่าน้ำหนักด้วยการแจกแจงแบบยูนิฟอร์ม โดยน้ำหนักที่ได้ไปตรวจดูว่าอยู่ในช่วงแอททริบิวต์รองตัวใด จะนำไอเท็มภายในแอททริบิวต์รองนั้น มาทำการสุ่มเลือกมา 1 ไอเท็มจากแอททริบิวต์รอง จากตารางที่ ก.2 ในส่วนของจำนวนแอททริบิวต์รองที่สุ่มได้จากการแจกแจงแบบปัวส์ซองเท่ากับ 2 หมายถึง L1 จะประกอบด้วย 3 แอททริบิวต์ แล้วทำการสุ่มน้ำหนักว่าแต่ละแอททริบิวต์ได้มาจากแอททริบิวต์รองตัวไหน เช่น ค่าน้ำหนักที่สุ่มได้ เท่ากับ 0.21 ซึ่งอยู่ในช่วงของแอททริบิวต์ C ก็จะทำการสุ่มค่าภายในแอททริบิวต์ C มา 1 ค่า ให้กับ L1 ทำแบบนี้ไปเรื่อยๆจนครบจำนวนของแอททริบิวต์รองที่สุ่มได้จากการแจกแจงแบบปัวส์ซอง จะได้ L1 = {C1 D2}

ตั้งแต่ L2 จะทำการสุ่ม ค่า e จากการสุ่มแบบเอ็กซ์โพแนนเชียลที่ correlation = 0.5 เพื่อสุ่มไอเท็มบางตัวจาก L ก่อนหน้า ถ้าใน L2 ค่า e จากการสุ่มแบบเอ็กซ์โพแนนเชียลเท่ากับ 1 แสดงว่าต้องสุ่มจาก L1 ขนาด 1 ไอเท็มส่วนสมาชิกที่เหลือจะได้จากการสุ่มค่าของไอเท็มจากแอททริบิวต์หลักตัวอื่นที่ไม่ซ้ำกับตัวไอเท็มที่ได้จากการสุ่มก่อนหน้า เช่น L2 = {A1 B3 C1 D3} เห็นได้ว่าไอเท็ม C1 ได้จาก L1 ก่อนหน้า

แต่ละไอเท็มเซตที่มาจากแอททริบิวต์หลักและแอททริบิวต์รองใน | L | จะมีการให้น้ำหนัก (weight) ที่สัมพันธ์กับชุดข้อมูลที่สร้างโดยค่าน้ำหนักนี้จะเกี่ยวข้องกับความน่าจะเป็นที่ไอเท็มเซตเหล่านี้ถูกหยิบ ค่าน้ำหนักจะกำหนดได้ด้วยการแจกแจงแบบเอ็กซ์โพแนนเชียลด้วยค่าเฉลี่ยเท่ากับ 1 และนำค่าน้ำหนักที่ได้มาทำเป็นค่าบรรทัดฐาน (normalized) เพื่อใช้ค่าน้ำหนักสำหรับความน่าจะเป็นในการสุ่มหยิบแต่ละชุดความสัมพันธ์ไปใส่ในทรานแซคชั่นต่างๆ

สำหรับแอททริบิวต์หลักที่เกิดขึ้นโดยธรรมชาตินั้นไอเท็มของแอททริบิวต์หลักมักไม่ได้ถูกซื้อไปด้วยกันเสมอ ดังนั้นจะมีการกำหนดค่าให้แต่ละไอเท็มเซตของแอททริบิวต์หลักใน | L | ด้วยระดับของค่าคอรัปชั่น (Corruption level) เมื่อมีการเพิ่มไอเท็มเซตของแอททริบิวต์หลักลงในทรานแซคชั่น โดยระดับค่าคอรัปชั่นจะถูกกำหนดเป็นค่าคงที่ซึ่งได้จากการสุ่มด้วยการแจกแจงปกติด้วยค่าเฉลี่ยเท่ากับ 0.5 และความแปรปรวนเท่ากับ 0.1 แสดงดังตารางที่ ก.4

ตารางที่ ก.4 แสดงตัวอย่างการให้ค่าน้ำหนัก, ค่าบรรทัดฐาน ของแต่ละชุด | L |

L	ไอเท็มเซตของแอททริบิวต์หลัก	ไอเท็มเซตของแอททริบิวต์รอง	ค่าน้ำหนัก (Weight)	ค่าบรรทัดฐาน (Normalize)	ค่าระดับคอรัปชั่น (Corruption level)
L1	{1 3 8 9 10}	{C1 D2}	0.052	0.052	0.3632
L2	{3 4}	{A1 B3 C1 D3}	0.023	0.075	0.5396
L3	{1 4 5 7 8 9}	{A1 B2}	0.123	0.198	0.5910
L1	{1 3 8 9 10}	{C1 D2}	0.052	0.052	0.3632
L2	{3 4}	{A1 B3 C1 D3}	0.023	0.075	0.5396
L3	{1 4 5 7 8 9}	{A1 B2}	0.123	0.198	0.5910
L4	{1 3 4 5 7}	{A1 C1 D2}	0.218	0.416	0.1375
L5	{1 5 6 7 8}	{C1 D3}	0.263	0.679	0.8766
L6	{1 2 3 5 7 8}	{A1 B2 C1 D2}	0.321	1	0.8760

2. **ขั้นตอนการเลือกไอเท็มเข้าไปในทรานแซคชั่น**

เริ่มจากการพิจารณาขนาดของไอเท็มเซตจากแอททริบิวต์หลักของทรานแซคชั่นซึ่งจะทำการเลือกสุ่มขนาดด้วยการแจกแจงแบบปัวส์ซองด้วยการกำหนดค่าเฉลี่ยเท่ากับขนาดทรานแซคชั่น |T| แสดงดังตารางที่ ก.5 แต่สำหรับแอททริบิวต์รองขนาดที่เกิดในทรานแซคชั่นเท่ากับจำนวนแอททริบิวต์รองทั้งหมด

ตารางที่ ก.5 แสดงตัวอย่างของแอททริบิวต์หลักของแต่ละทรานแซคชั่น

ขนาดของไอเท็มเซตจากแอททริบิวต์หลักของทรานแซคชั่น									
ลำดับทรานแซคชั่น									
1-10	11-20	21-30	31-40	41-50	51-60	61-70	71-80	81-90	91-100
6	7	6	9	7	6	9	6	10	8
8	5	9	9	4	8	6	8	6	7
4	6	10	8	5	9	5	8	6	10
9	6	7	5	6	7	5	9	8	9
9	8	6	4	6	7	7	9	7	10
9	6	8	6	7	5	6	7	6	8
5	10	6	3	10	9	8	5	8	6

เมื่อได้ขนาดไอเท็มของแอททริบิวต์หลักในทรานแซคชั่นแล้ว ขั้นตอนต่อไปจะเป็นการนำเอาไอเท็มทั้งที่เป็นไอเท็มของแอททริบิวต์หลัก และไอเท็มของแอททริบิวต์รองในชุดของ | L | โดยจะทำการสุ่มชุดของข้อมูล | L | มาใส่ทรานแซคชั่นดังนี้

2.1 จะทำการค่าสุ่มค่าน้ำหนักด้วยการแจกแจงแบบยูนิฟอร์ม โดยจะนำค่าน้ำหนักที่ได้ไปตรวจสอบว่าอยู่ในช่วงข้อมูลของ | L | ชุดใด และจะนำสมาชิกของ | L |พิจารณาเพื่อทำการหยิบใส่ในทรานแซคชั่น เช่น ถ้าค่าน้ำหนักที่สุ่มได้คือ 0.123 ซึ่งค่านี้มีค่าค่าบรรทัดฐานมากกว่าค่าน้ำหนักของ | L | ชุดที่ 2 ที่มีค่า 0.075 และมีค่าน้อยกว่า | L | ชุดที่ 3 ที่มีค่า 0.198 จะนำสมาชิกของ | L | ชุดที่ 3 ที่ประกอบด้วย ไอเท็มของแอททริบิวต์หลัก {1 4 5 7 8 9} และไอเท็มของแอททริบิวต์รอง {A1 B2}

2.2 หยิบค่าของไอเท็มของแอททริบิวต์หลักและแอททริบิวต์รองที่ได้จากขั้นตอนที่ 2.1 ซึ่งแบ่งการหยิบออกเป็น 2 ส่วน

2.2.1 การหยิบส่วนของไอเท็มของแอททริบิวต์หลัก

ทำการสุ่มค่าความน่าจะเป็นด้วยการแจกแจงแบบยูนิฟอร์ม สำหรับไอเท็มของแอททริบิวต์หลักแต่ละตัวของชุด | L | ที่สุ่มได้จากขั้นตอนที่ 2.1 แล้วนำมาเปรียบเทียบกับค่าคอรัปชั่นของ ชุด | L | ที่สุ่มได้ ถ้าค่าความน่าจะเป็นของไอเท็มของแอททริบิวต์หลักแต่ละตัวมีค่าน้อยกว่าค่าคอรัปชั่น จะนำไอเท็มนั้นมาใส่ในทรานแซคชั่น โดยจะทำการเทียบกับไอเท็มทุกตัวจนกว่าจะครบเท่ากับขนาดของแอททริบิวต์หลักของทรานแซคชั่นนั้น

สำหรับในกรณีที่ชุดของ |L| ที่สุ่มได้เมื่อทำการเปรียบเทียบจนครบทุกตัวแล้วแต่ยังได้ไม่ครบตามจำนวนของขนาดของแอททริบิวต์หลักในทรานแซคชั่นนั้น จะทำการสุ่มค่าในขั้นตอนที่ 2.1 ใหม่ จากนั้นจึงมาพิจารณาการหยิบส่วนของไอเท็มของแอททริบิวต์หลักซ้ำ จนกว่าจะได้จำนวนของไอเท็มเท่ากับขนาดของแอททริบิวต์หลักที่เกิดภายในทรานแซคชั่นนั้น จึงจะทำการหยุดสุ่มและเปรียบเทียบกับค่าคอรัปชั่น

2.2.2 การหยิบส่วนของไอเท็มของแอททริบิวต์รอง

จะทำการหยิบไอเท็มของแอททริบิวต์รองจาก | L | ชุดแรกที่ทำการสุ่มได้จากขั้นตอนที่ 2.1 โดยทำการหยิบไอเท็มทั้งหมดภายใน | L | ชุดแรกที่ได้มาใส่ในทรานแซคชั่นตามแต่ละแอททริบิวต์ แล้วถ้าแอททริบิวต์รองใดยังไม่ค่าของแอททริบิวต์รองนั้นใน | L | จะทำการสุ่มหยิบไอเท็มจากชุดของแอททริบิวต์รองนั้นจนครบทุกแอททริบิวต์รองที่มีในทรานแซคชั่น เช่น จากการเลือก | L | ชุดที่ 3 ที่ไอเท็มของแอททริบิวต์รองที่มีภายใน | L | เท่ากับ {A1B2} ซึ่งยังขาดไอเท็มจากแอททริบิวต์รอง C และ D ดังนั้นจึงทำการสุ่มไอเท็มจากแอททริบิวรอง C และ D

จากนั้นจึงไปทำการสุ่มไอเท็มทั้งในแอททริบิวต์รองและแอททริบิวต์หลักสำหรับทรานแซคชั่นถัดไปโดยทำซ้ำขั้นตอนที่ 2.1 และ 2.2 จนกว่าจะครบจำนวนทรานแซคชั่นที่กำหนด โดยแสดงทรานแซคชั่นทั้งหมดที่ได้แสดงดังตารางที่ ก.6

ตารางที่ ก.6 แสดงตัวอย่างของทรานแซคชั่นที่ได้จากข้อมูลสังเคราะห์

TID	A	B	C	D	ITEM
1	A1	B2	C1	D2	1,2,4,5,7,8
2	A1	B3	C2	D1	1, 2, 3, 5, 7, 8, 9,10
3	A2	B2	C3	D1	1, 5, 6, 7
...	...	...	...	...	...
100	A3	B1	C1	D2	1, 2, 3, 5, 7, 8

ภาคผนวก ข.

ผลงานวิจัยที่ได้รับการตีพิมพ์เผยแพร่

2011 3rd International Conference on Machine Learning and Computing (ICMLC 2011)

An 2-Dimension Incremental Association Rule Discovery algorithm

Suriyes Suksean
Faculty of Information Technology
King Mongkut's Institute of technology Ladkrabang
Bangkok, 10520 Thailand
tumsuriyes@hotmail.com

Worapoj Kreesuradej
Faculty of Information Technology
King Mongkut's Institute of technology Ladkrabang
Bangkok, 10520 Thailand
worapoj@it.kmitl.co.th

Abstract— **In this paper, an 2-dimension Incremental Association Rule Discovery algorithm is proposed to discover hybrid-dimensional association rules. The proposed algorithm is designed to deal with the circumstance that not only new transactions but also new attributes are appended to an original database simultaneously. Basically, the proposed algorithm is modified from Fast Update algorithm (FUP) and Hybrid-Dimension Fast Update Algorithm (HDFUP). The experiments of the proposed algorithm are conducted to show the efficient of the algorithm.**

Keywords- Hybrid-dimension association rule; incremental association rule.

I. INTRODUCTION

Association rule mining, one of the most important and well researched techniques of data mining was first introduced in Agrawal et al. 1993 [1]. The association rule mining is discover interesting relationship among items in a given transaction database. Giving a set of transactions, where each transaction consists of items, an association rule is an expression of the form $X \rightarrow Y$, where X and $Y \subset I$ are set of items called itemset. X is called antecedent while Y is called consequent.

There are two important basic measures for association rules, support(s) is defined as the percentage of records that contain $X \cup Y$ to the total number of records in the database and confidence(c) is defined as the percentage of the number of transaction that contain $X \cup Y$ to the total number of records that contain X. The association rule discovery algorithm is decomposed into two-step process [2]. The first step finds all frequent itemsets that have support value greater than or equal to a minimum support threshold and the second step is generate association rules from the frequent itemset that have value greater than or equal a minimum confidence threshold.

Association rules can be classified as single-dimension association rules and multi-dimension association rules based on number of predicates or dimensions appearing in the rule. Single-dimension association rules describe relationship among items within an attribute or a dimension. Single-dimension association rules contain a single predicate [6]. As an example, buys (x, "computer") → buys (x, "printer") is a single-dimension association rule.

Multiple-dimension association rules describe relationship among items within an attribute and also relationship among item between attributes. Multi-dimension association rules contain multiple predicates. Furthermore, multi-dimension association rules can be divided into 2 types.

- Inter-dimension association rules are the multidimensional association rules which have no repeated dimension appeared in the same rule, an example of such a rule is the following: age (X, "20…29") occupation(X, "student") → buys(X, "laptop")
- Hybrid-dimension association rules are multidimensional association rules with repeated predicates, which contain multiple occurrences of some dimension, an example of such a rule is the following: age(X, "20…29") buys (X, "laptop") →buys (X, " printer")

Association rules discovered from a database are valid only the current state of database [5]. However, when new transactions are inserted into a database, association rules that are obtained before the new transactions are inserted possibly no valid. In addition, new association rules may be discovered in the updated database.

To deal with such problem efficiently, an incremental association rule algorithm is proposed. The well-known algorithm is Fast Update algorithm (FUP) presented by Cheung et al. [3]. However, the algorithm can only be applied for discovering single-dimension association rules.

To discover multi-dimension association rules incrementally, hybrid-dimension fast update algorithm (HDFUP), a FUP based algorithm, is proposed [4]. The algorithm can find multi-dimension association rules when new transactions or new records data are inserted into a database.

When both new transactions (or new records) and new attributes (or new dimensions) are appended to an original database, HDFUP fails to deal with this case. Thus, this paper proposed a new algorithm, called 2- Dimension Incremental Association rule Discovery algorithm to deal with this case. The objective of this algorithm is to solve the updating problem of hybrid-dimensional association rules when new transactions and new attributes are appended to a database. As a result, the proposed algorithm has execution time faster than that of previous algorithms.

978-1-4244-9253-4/11/$26.00 ©2011 IEEE

II. Related Work

A. Hybrid-Dimension Fast Update Algorithm (HDFUP)

Basically, Hybrid-Dimension Fast Update Algorithm (HDFUP) is modified from FUP. An original database, i.e. DB, has D transactions. Then, an increment database, i.e. db, contains the set of new coming transactions. An increment database has d transactions. The set of old and new coming transactions are called an updated database, i.e. UP.

The algorithm uses two new join operations in order to discover hybrid dimension association rules. Two new join definitions are defined as followings.

Definition 1 Intra-dimension join

The first to (k-2) item of l_1 and l_2 as same can be found by follows:

$l_1 \triangleright\triangleleft l_2 = (l_1[1] = l_2[1]) \cap (l_1[2] = l2[2]) \cap \ldots (l_1[k-2] = l_2[k-2]) \cap (l_1[k-1] = l_2[k-1])$.

Definition 2 Inter-dimension join

The items from the 2^{th} to the $(k-1)^{th}$ in l_1 are the same as the items from the 1^{th} to the $(k-2)^{th}$ in l_2 and $l_1[1] < l_2[k-1]$ can be found join by the follows:

$l_2 \triangleright\triangleleft l_2 = (l_1[2] = l_2[1]) \cap (l_1[3] = l_2[2]) \cap \ldots (l_1[k-1] = l_2[k-2]) \cap (l_1[1] < l_2[k-1])$.

```
Input: DB, the original database (D is equal to total number
       of transactions);
       L_k: the set of all large k- itemsets in DB,
       where k= 1, ..., r;
       db: an increment database (with its size equal to d);
Output: L': The set of all large itemsets in DB∪db.
Method:
    The 1st iteration/* find L'_1, the set of all
    large 1-itemsets in DB∪db */
    W = l_1; C = ∅; L'_1 = ∅; P = ∅;
    /* W: winners, C: candidate sets; L'_1: initialized,
    P: for optimization */
    for all T ∈ db do /* scan db */
     for all 1-itemset X ∈ T do {
      if X ∈ W then X.support_UP++;
      else {
       if X ∉ C
       then { C = C ∪ {X}; X.support_db = 0;}
       /* init the support cors and add X into C */
       X.support_UP++;}
      };
    for all X ∈ W do /* put winners into L'_1 */
     if X.support_UP ≥ s *(D + d)
      then L'_1 = L'_1 ∪ {X};
    for all X ∈ C do /*prune candidate sets in C*/
     if X.support_UP < s *(D + d)
      then { C = C − {X}; P = P ∪ {X};}
      /* P will be used for optimization */
    for all T ∈ DB do /* scan DB */
     for all 1-itemset X ⊆ T do {
      if X ∈ C then X.support_UP++;
      if X ∈ P then removes X from T;
      /* Transaction T is reduced */
     };
    for all X ∈ C do /* put winners into L'_1 */
     if X.supportUD ≥ s × (D+ d)
      then L'_1 = L'_1 ∪ {X};
    return L'_1; /* end of the 1st iteration */
The kth iteration: /*for k= 2 or larger, repeat this program
fragment to find L'_k the set of all large k-itemsets in the
updated database, until either L'_k returned is empty or do = ∅*/
W = L_k; L'_k = ∅;
/*W: winners; L'_k: initialized */
if k= 2
 then {C = apriori_gen(L'_k-1) - L_k }
 else {C = apriori_gen(L'_k-1) - L_k };
 /* the size-k candidate sets */
for all k-itemset X ∈ W do
/*prune off losers in W */
 for all (k-1)-itemset Y ∈ L_k-1 − L'_k-1 do
  if Y ⊆ X then {W = W − {X}; break;}
for all T ∈ db do { /* scan db */
 for all X ∈ (W ∩ T) do X.support_UP++;
 /* Subset(W,T) returns all the sets in W contained in T */
 for all X ∈ Subset(C,T) do X.support_db++;
 /* find support of all X ∈ C */
 Reduce db(T);
 /* Some items in transactions in db can be removed,
 discussed in next section */
}
for all X ∈ W do
/* put the winners from W into L'_k */
if X.support_UP ≥ s × (D+d)
 then L'_k = L'_k ∪ {X};
for all X ∈ C do /* prune candidate sets in C*/
 if X.support_db < s × d then C = C − {X};
for all T ∈ DB do { /* scan DB */}
 for all X ∈ Subset(C,T) do X.support_db++
 Reduce_DB(T);
 /* Some items in transactions in DB can be removed,
 discussed in next section */
for all X ∈ C do
 if X.support_UP ≥ s × (D + d)
  then L'_k = L'_k ∪ {X};
return L'_k; /* the end of the k-th iteration */
```

Figure 1. HDFUP algorithm

HDFUP algorithm, as shown in Figure1, 2 and 3, has three phases. The first phase is similar to FUP Algorithm. The 1^{st} large-itemset of updated database is discovered based on the following conditions:

- If $X \in C_1$ and X.supportdb < s × d, those items are removed from candidate itemset and insert them to P for optimization.
- A scan of original database is performed when $X \in C_1$. In case of X.supportUD ≥ s × (D+d), item X is 1^{st} large-item.

In the second phase, the 2^{nd} large-itemset of updated database is determined. The 2^{nd} large-itemset of updated database is discovered based on the following conditions:

- If $X \in C2$ and X.supportdb < s × d, these items are removed from candidate itemset.
- A scan of original database is performed when $X \in C2$. In case of X.supportUD ≥ s × (D+d), item X is 2^{nd} large-item.

In the third phase, the k large-itemsets of updated database, that k is greater than and equal to three, i.e. k≥3, are determined. The k large-itemsets of updated database are discovered based on the conditions that like the second phase.

```
procedure apriori_gen1 (L'k-1: Large itemsets)
{ C = null;
  for each l1 ∈ L'k-1
    for each l2 ∈ L'k-1
      if is1inset1oin (l1) or is1inset1oin (l2)
        then {
          c = l1 ⊳◁ l2;
          InsertInto C
      for each c ∈ C
        for each (k-1)-subset s of c
          if s ∉ L'k-1
            then delete c from C
}
```

Figure 2. Apriori_gen1 procedure

```
procedure apriori_gen2 (L'k-1: Large itemsets)
{ C = null;
  For each l1 ∈ L'k-1
    for each l2 ∈ L'k-1
      if is1inset1oin (l1) and is1inset1oin (l2)
        then //make intradimension join
          {if (l1[1] = l2[1]) ∧ (l1[2] = l2[2]) ∧
          (l1[k-2] = l2[k-2]) ∧ (l1[k-1] < l2[k-1])
            then {
              c = l1 ⊳◁ l2;
              InsertInto C;}
          }
          else //make interdimension join
            {if (l1[2]= l2[1]) ∧ (l1[3]= l2[2]) ∧ ... ∧
            (l1[k-1] = l2[k-2]) ∧ l1[1] < l2[k-1]}
        then {
          c = l1 ⊳◁ l2;
          InsertInto C;
        }
  for each c ∈ C
    for each (k-1)-subset s of c
      if s ∉ L'k-1
        then delete c from C
}
```

Figure 3. Apriori_gen2 procedure

TABLE I THE NOTATION FOR ALGORITHM

Notation	Meaning
DB	Original Database
DB(A)	DB ∪ db(A)$_D$
db(A)	Increment subattributes
db(T)	Increment transactions
db	db(T) ∪ db(A)$_D$
UP	Update database
D, d,U/D	length DB, db, UD
k	Number of itemset
s	Minimum support
C_k	Candidate k-itemset
F_k	Frequent k-itemset

III. An 2-Dimension Incremental Association Rule Discovery Algorithm

When not only new transactions but also new attributes are appended to an original database simultaneously, the previous algorithm cannot be applied under this circumstance. The updated database is shown in figure 4. From the figure 4, an increment database, i.e. db, consists of two parts: new increment sub attributes, i.e. db(A), and new increment transactions, i.e. db(T). The notation used in this section is shown in Table I.

TID	Sub Attribute	Main Attribute	New Sub attributes
1 2 3 M	DB		db(A)
N		db(T)	

Figure 4. An updated database when not only new transactions but also new attributes are appended to an original database

Unlike the previous work, the proposed algorithm is designed to deal with the circumstance that not only new transactions but also new attributes are appended to an original database simultaneously. The algorithm is divided into two parts. The first part, shown in figure 5 and 6, is deal with incremental updating when new increment sub attributes, i.e. db(A) append to an original database. The second part, shown in Figure 7, 8 and 9, is deal with incremental updating when new increment transactions, i.e. db(T) are inserted into an original database.

For the first part, the algorithm finds all updated large k-itemsets when new increment sub attributes, i.e. db(A) append to an original database. The first part consists of two major steps. The first step is finding candidate 1st-itemset i.e. $C_1^{DB(A)}$ by scanning only db(A). If $X \in C_1^{DB(A)}$ and $X.support_{DB} \geq s \times D$ then X will insert to $F_1^{DB'}$.

Phase 1
Input: db(A), s, F_k^{DB} **Output:** $F_k^{DB(A)}$
1 k = 1
2 if k = 1
3 for (i=1) D ∈ db(A) do
4 for all 1-itemset X ⊆ T ;
5 if X ∈ C
6 then {C_k ∪ {X}; X.support_k = 0 };
7 X.support_k++ ;
8 for all X ∈ C do
9 if X.support_k ≥ s × D
10 then {
11 $F_1^{DB(A)} = F_1^{DB} \cup \{X\}$;
12 $F_k^{DB(A)} \cup F_k^{DB}$
13 }
14 k = k+1
15 for (k = 2; $F_{k-1}^{DB(A)} \neq \emptyset$; k++) do
16 L = Frequent_AppendAttribute $F_{k-1}^{DB(A)}$;
17 if $F_k^{DB} \neq \emptyset$ then $F_k^{DB(A)} = F_k^{DB} \cup L$
18 else $F_k^{DB(A)} = L$

Figure 5. Main algorithm of first part

Then, the second step is finding frequent k-itemsets for k ≥ 2. The support counts of the k-itemsets that are obtained from inter-dimensional join need to be specially updated. This is the case because their support counts obtained from an original database are not available. Here, their support counts are assumed to be equal to the support counts of the (k-1)-itemsets. This estimation helps to reduce number of rescanning times of an original database.

```
Procedure : Frequent_AppendAttribute
1.  for each l1 ∈ F(k-1)^DB(A) do
2.    for each l2 ∈ F(k-1)^DB(A) {
3.      if (isinnerJoin(l1) = false and
4.          isinnerJoin (l2)= false)
5.        if (l1 and l2 not come from same attribute )
6.          then L^DB(A) [k] = join l1 ⋈ l2
7.              if l1.support <= l2.support
8.                then L^DB(A) [k].support = l1.support
9.              else
10.               L^DB(A) [k].support = l2.support
11.     else  if (isinnerJoin(l1) = true and
12.             isinnerJoin(l2)= false) {
13.           L^DB(A) [k] = join l1 ⋈ l2;
14.           if l1.support <= l2.support
15.             then L^DB(A) [k].support = l1.support
16.           else
17.             L^DB(A) [k].support = l2.support    }
18.     L^DB(A) [k].sup = true
19.   } return L^DB(A)
```

Figure 6. Frequent AppendAttribute procedure.

The second part of the algorithm can be divided 3 major steps. The second part of the algorithm is similar to HDFUP algorithm. As the first step, updated all frequent 1-itemset by scan db for generating candidate 1st itemset. if X ∈ F_1^{DB}, then support count in UD of the 1st itemset is sum of support count in db and support count in DB(A). If X.support$_{UD}$ ≥ s × (D + d), inserted the 1st itemset into F_1^{UD}. In case X ∈ C_1^{db} and X.support$_d$ ≥ s × d, the 1st itemset is scaned in DB(A). If X.support$_{UD}$ ≥ s × (D + d), the 1st itemset inserted to F_1^{UD}.

```
Phase 2
Input : db, DB(A), s, F_k^DB(A)
Output : F_k^UD : the st of all large itemset in DB U db
1.  k_inc = 1; W = F_1^DB
2.  if k = 1
3.    for all T ∈ db do
4.      for all 1-itemset ∈ W{
5.        if X ∈ W then X.support_d ++;
6.        else { if X ∉ C
7.             then {C = C {X}; X.support_d = 0 };
8.          X.support_d ++; }   };
9.    for all X ∈ W do
10.     if X.support_UD ≥ s × (D + d)
11.       then F_1^UD = F_1^UD ∪ {X} ;
12.   for all X ∈ C do
13.     if X.support_d ≤ s × d  then C = C− {X};
14.   for all T ∈ DB(A) do
15.     for all 1-itemset X ⊆ T {
16.       if X ∈ C then then X.support_D ++  };
17.   for all X ∈ C do
18.     if X.support_UD ≥ s × (D + d)
19.       then F_1^UD = F_1^UD ∪ {X} ;
20. /* The k-th iteration : for k ≥ 2 , */
21. k_inc = k_inc + 1
22. W = F_1^DB(A); F_k^UD = ∅; C_k^sup = ∅;
23. for {k = 2 ; F_k^UD ≠ ∅ ; k++ } do {
24.   if k = 2  then C_2 = hybrid_gen1 (F_1^UD) − F_2^DB(A);
25.   else C_k = hybrid_gen2 (F_k-1^UD) − F_k-1^DB(A);
26.   for all k-itemset X ∈ W do
27.     for all (k-1) itemset Y ∈ F_k-1^DB(A) − F_k-1^UD do
28.       if Y ⊆ X then { W = W − {X}; break; }
29.   for all T ∈ db do {
30.     for all X ∈ Subset (W,T) do X.support_d ++;
31.     for all X ∈ Subset (C_k,T) do X.support_d ++
32.   for all X ∈ W do
33.     if X.support_UD ≥ S × (D+d) then
34.       if X.sup_mark = true
35.         then X.support_DB = X.support_d + X.support_DB
36.           C_k^sup = C_k^sup ∪ {X}
37.         else { F_k^UD = F_k^UD ∪ {X} };
38.   for all X ∈ C do
39.     if X.support_d < S × d then C_k = C_k − {X};
40.   for all T ∈ DB(A) do {
41.     For all X ∈ Subset(C_k, T) do X.support_D ++
42.     For all X ∈ Subset(C_k^sup, T) do X.support_D ++ }
43.   for all X ∈ C ∪ C_k^sup do
44.     if X.support_UD ≥ S × (D + d) then
45.       F_k^UD = F_k^UD ∪ {X};
```

Figure 7. Main algorithm of second part.

As the second step, the algorithm finds large 2nd itemsets. The algorithm generates candidate 2nd itemsets using hybrid_gen1 procedure. Then, the algorithm removes joined items that are the same as $F_2^{DB(A)}$. Furthermore, If X ∈ $F_2^{DB(A)}$ which has subset $F_1^{DB(A)} - F_1^{UD}$, X is removed from $F_2^{DB(A)}$. Then, if X ∈ subset(W,T) and X ∈ subset(C_2^{sup},T), X is scaned in db for support count. Next, if support value of X ∈ subset(W,T) ≥ s × (D + d) and X.sup is false, X is inserted itemset into F_2^{UD}. However, if support value of X ∈ subset(W,T) ≥ s × (D + d) and X.sup is true, X is inserted item into C_2^{sup}. If X ∈ C_2^{db} and X.support$_d$ < s × d , X are removed from C_2^{db}. A scan DB(A) is performed when X ∈ C_2^{db} and X ∈ C_2^{sup}. In case of X.support$_{UD}$ ≥ s × (D + d), inserted item to F_2^{UD}.

Regarding the last step, the k large-itemsets of updated database when k ≥ 3 are discovered. This step is similar to that of the second step.

```
Procedure : hybrid_gen1 (F_{k-1}^{DB(k)})
1.  C[k] = ∅
2.  for each l_1 ∈ F_{k-1}^{DB(k)} do
3.    for each l_2 ∈ F_{k-1}^{DB(k)} {
4.    /* isInnerJoin() is function for check */
5.    /* all item in l_i are main attribute */
6.      if (isInnerJoin(l1) and isInnerJoin(l2)
7.        then C[k] = join l_1 ⋈ l_2;
8.      else { if (l1 and l2 not come from same attribute )
9.             then C[k] = join l_1 ⋈ l_2; }
10. for each c ∈ C[k] do
11.   for each (k-1)-subset s of c {
12.     if s ∉ F_{k-1}^{DB(k)} then delete c from C[k]; }
13. return C[k];
```

Figure 8. Hybrid_gen1 procedure

```
Procedure : hybrid_gen2 (F_{k-1}^{DB(k)})
1.  C[k] = ∅
2.  for each l_1 ∈ F_{k-1}^{DB(k)} do
3.    for each l_2 ∈ F_{k-1}^{DB(k)} {
4.    if isInnerJoin(l1) and isInnerJoin(l2) then
5.      if (l1[1]=l2[1]) ∧(l1[2]=l2[2]) ∧... ∧
6.        (l1[k-1] < l2[k-1]) ∧(l1[k-1]<l2[k-1])
7.        then C = join l_1 ⋈ l_2;
8.      else {
9.        if (l1[2]=l2[1]) ∧(l1[3]=l2[2]) ∧... ∧
10.         (l1[k-1] = l2[k-2]) ∧ (l1[1]<l2[k-1])
11.         then C = join l_1 ⋈ l_2; }
12.   for each c ∈ C do
13.     for each (k-1)-subset s of c {
14.       if s ∉ F_{k-1}^{DB(k)} then delete c from C[k];}
15.   return C;
```

Figure 9. Hybrid_gen2 procedure.

IV. EXPERIMENTS

To evaluate the efficiency of the proposed algorithm, the experiment is conducted on a sample database which consists of three dimensions (transaction dimension, age dimension and salary dimension). Here, the experiment uses an original database consisting of 600 transactions, an incremental transaction database consisting of 400 transactions and incremental attribute database consisting of 1,000 transactions. The efficiency of the proposed algorithm is compared with that of Apriori algorithm and Apriori algorithm using hybrid dimension join. The algorithm is implemented and tested on a PC with a 2.67 GHz Intel(R) Core(TM) i5 CPU and 2.0 GB main memory.

Figure 10 and TABLE II show the average of execution time for Apriori algorithm, Apriori algorithm using hybrid dimension join and our approach with various minimum support thresholds. The results also show that the proposed algorithm has much better running than that of Apriori algorithm, Apriori algorithm using hybrid dimension join.

TABLE II AVERAGE OF EXECUTION TIME

Min_sup	Average of Execution time		
	Apriori	*Multi-Apriori*	*2-Dimension*
5%	354.291	355.152	323.998
6%	229.519	230.744	206.982
7%	193.308	175.718	166.217
8%	173.371	156.761	140.695
9%	154.247	142.109	134.238
10%	148.219	177.997	122.549

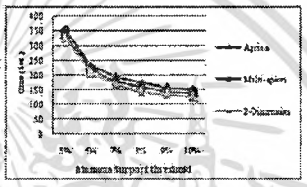


Figure 10. The execution time of Apriori, Multi-Apriori and the proposed algorithm

V. CONCLUSION

In this paper, we propose a new incremental algorithm when are increased attribute and row data for discovering hybrid-dimension association rules. In the future, further researches and experiments on the proposed algorithm will be presented.

VI. REFERENCES

[1] R. Agrawal, T. Imieliński, and A. Swami, " Mining association rule between sets of items in large database", In Proceeding of the ACM SIGMOD Int'l Conf. on Management of Data, Washington, USA, May 1993, pp. 207-216.

[2] R. Agrawal, and R. Srikant, "Fast Algorithm for Mining Association Rules." Proceedings of the International Conference on Very Large Database, Santiago, Chile, 1944, pp. 487-499.

[3] D. Cheung, J. Han, V. Ng, and C. Y. Wong, "Maintenance of Discovered Association Rules in Large Database: An Incremental Updating Technique," Proceedings of the 12th IEEE International Conference on Data Engineering, 1996, pp. 106-114.

[4] S. Chatsemakul, W. Kreesuradej, "Hybrid-dimension Fast Update Algorithm," Proceedings of 24th International Technical Conf. on Circuits, Computer and Communication, Jeju, Korea, July 2009.

[5] W.-G. Teng, M.-S. Chen, "Incremental Mining on Association Rule", Studies in Fuzziness and Soft computing, 2006, pp. 125-162.

[6] J. Han, and M. Kamber, "Data mining: Concepts and Techniques," Morgan Kaufmann Publishers, San Francisco, California, pp. 227-256, 2006.