การค้นหากฎความสัมพันธ์ที่แข็งแกร่ง

นางสาวลักษมี โขมโนทัย

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์
มหาวิทยาลัยเทคโนโลยีสุรนารี
ปีการศึกษา 2548

ISBN 974-533-518-5

MINING STRONG ASSOCIATION RULES

Laksamee Khomnotai

A Thesis Submitted in Partial Fulfillment of the Requirements for the

Degree of Master of Engineering in Computer Engineering

Suranaree University of Technology

Academic Year 2005

ISBN 974-533-518-5

การค้นหากฎความสัมพันธ์ที่แข็งแกร่ง

มหาวิทยาลัยเทคโนโลยีสุรนารี อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

คณะกรรมการสอบวิทยานิพนธ์

(ผศ. ดร.พิชโยทัย มหัทธนาภิวัฒน์)
ประธานกรรมการ

(ผศ. ดร.กิตติศักดิ์ เกิดประสพ)
กรรมการ (อาจารย์ที่ปรึกษาวิทยานิพนธ์)

(รศ. ดร.นิตยา เกิดประสพ)
กรรมการ

(ผศ. ดร.คะชา ชาญศิลป์)
กรรมการ

(รศ. ดร.เสาวณีย์ รัตนพานี)
รองอธิการบดีฝ่ายวิชาการ

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

ลักษมี โขมโนทัย : การค้นหากฎความสัมพันธ์ที่แข็งแกร่ง (MINING STRONG ASSOCIATION RULES) อาจารย์ที่ปรึกษา : ผศ. ดร.กิตติศักดิ์ เกิดประสพ, 133 หน้า. ISBN 974-533-518-5

กฎความสัมพันธ์ คือ กฎในลักษณะ ถ้า-แล้ว ใช้อธิบายความสัมพันธ์ในกลุ่มข้อมูลว่า ถ้าเกิดลักษณะหนึ่งแล้วจะเป็นผลให้เกิดอีกลักษณะหนึ่งตามมา กฎความสัมพันธ์ที่มีความแข็งแกร่งหมายถึง กฎความสัมพันธ์ที่มีจำนวนข้อมูลสนับสนุนมาก และมีค่าความถูกต้องสูง การทำเหมืองข้อมูลในลักษณะการค้นหาความสัมพันธ์ภายในกลุ่มข้อมูลมีประโยชน์ เพื่อให้ได้กฎที่มีความสัมพันธ์แข็งแกร่งจะต้องทำการค้นหาจากข้อมูลที่มีปริมาณมาก ต้องมีจำนวนข้อมูลสนับสนุนและมีความถูกต้องสูง เพื่อความน่าเชื่อถือของกฎที่ได้ และสามารถนำผลที่ได้ไปใช้ได้อย่างมีประสิทธิภาพ แต่อุปสรรคในการทำเหมืองข้อมูลกับข้อมูลทรานแซคชันคือปริมาณของข้อมูลที่มีมาก ทำให้การประมวลผลเพื่อค้นหาความสัมพันธ์ภายในข้อมูลต้องใช้เวลามาก งานวิจัยนี้จึงมีจุดมุ่งหมายเพื่อเพิ่มความเร็วในการค้นหากฎความสัมพันธ์ที่มีความแข็งแกร่งจากข้อมูลปริมาณมากด้วยเทคนิคการสุ่มข้อมูลโดยผลลัพธ์ที่ได้จะต้องไม่แตกต่างจากการค้นหากฎจากข้อมูลทั้งหมด

จากการศึกษาพบว่าข้อมูลสุ่มเพียง 1 เปอร์เซ็นต์ สามารถให้ผลการค้นพบกฎความสัมพันธ์ที่แข็งแกร่งได้เทียบเท่ากับการใช้ข้อมูลทั้งหมด และสามารถลดเวลาที่ใช้ในการค้นหากฎความสัมพันธ์ที่แข็งแกร่งได้มากกว่า 95 เปอร์เซ็นต์

จากผลการศึกษาเบื้องต้นนี้ผู้วิจัยได้พัฒนาโปรแกรม เพื่อค้นหากฎความสัมพันธ์ที่แข็งแกร่ง โดยเพิ่มส่วนการสุ่มข้อมูลในอัลกอริทึมเอไพรออริด้วยภาษา SQL และปรับปรุงการทำงานของโปรแกรมให้สามารถรองรับการทำงานกับข้อมูลสตรีมมิง

สาขาวิชา วิศวกรรมคอมพิวเตอร์ ลายมือชื่อนักศึกษา ลักษมี โขมโนทัย

ปีการศึกษา 2548 ลายมือชื่ออาจารย์ที่ปรึกษา กิตติศักดิ์ เกิดประสพ

ลายมือชื่ออาจารย์ที่ปรึกษาร่วม

LAKSAMEE KHOMNOTAI : MINING STRONG ASSOCIATION RULES.
THESIS ADVISOR : ASST. PROF. KITTISAK KERDPRASOP, Ph.D.,
133 PP. ISBN 974-533-518-5

APRIORI/APRIORI IN SQL/STRONG ASSOCIATION RULES/RANDOM SAMPLING/STREAMING

Association rules are if-then rules that are used to express the causes and effects among items in a dataset. Strong association rules are association rules with high support and high confidence. Association rule mining for discovering strong association rules requires a very large dataset to assure high data support and high confidence. Time usage is an important problem in association rule mining for strong rules, especially when mining from large transactional data. We are interested in studying the technique to decrease time usage in association rule mining from a very large dataset. We use random sampling technique to mine strong association rules from a sampled transactional data with the constraint that the discovered rules have to be identical to these discovered from the original large transactions.

From the experimental results, we have found that a 1% random sample of the data can yield exactly the same set of strong rules as those obtained from the large original transaction data set. By means of sampling, we can save as much as 95% of association mining time.

The results of the preliminary study lead to the development of strong association mining program. The program is an extension of APRIORI algorithm to include the sampling phase. We also design our program to handle streaming data.

School of Computer Engineering

Academic Year 2005

Student's Signature Laksamee Khomnotai

Advisor's Signature

Co-advisor's Signature

กิตติกรรมประกาศ

วิทยานิพนธ์นี้สำเร็จลุล่วงด้วยดี ผู้วิจัยขอกราบขอบพระคุณ บุคคลต่างๆ ที่ได้กรุณาให้คำปรึกษา แนะนำ ช่วยเหลือ ทั้งในด้านวิชาการ และการดำเนินงานวิจัยดังต่อไปนี้

ผู้ช่วยศาสตราจารย์ ดร.กิตติศักดิ์ เกิดประสพ อาจารย์ที่ปรึกษาวิทยานิพนธ์

รองศาสตราจารย์ ดร.นิตยา เกิดประสพ อาจารย์ที่ปรึกษาวิทยานิพนธ์ร่วม

ผู้ช่วยศาสตราจารย์ ดร.พิชโยทัย มหัทธนาภิวัฒน์ และผู้ช่วยศาสตราจารย์ ดร.คะชา ชาญศิลป์ ที่กรุณาให้คำแนะนำในการเขียนวิทยานิพนธ์ฉบับนี้

ขอขอบคุณ เพื่อนๆ บัณฑิตศึกษาทุกท่าน ที่ให้คำปรึกษา กำลังใจ และให้ความช่วยเหลืออย่างดีมาตลอด

ขอขอบคุณ คุณอาทุกท่านที่ได้สนับสนุนทุนการศึกษา และคอยให้กำลังใจแก่ผู้วิจัยมาโดยตลอด

ขอขอบคุณ คุณนริศ มิ่งโมรา ที่คอยช่วยเหลือในด้านต่างๆ และคอยให้กำลังใจในยามที่ผู้วิจัยเกิดความท้อแท้

ขอขอบคุณ คุณมนัส สธนเสาวภาคย์ ที่คอยให้กำลังใจในการทำวิจัยจนสำเร็จลุล่วง

สุดท้ายนี้ ผู้วิจัยขอขอบคุณคณาจารย์ทุกท่านที่ได้ประสิทธิ์ประสาทวิชาความรู้ต่างๆ ทั้งในอดีต และปัจจุบัน และขอกราบขอบพระคุณบิดา มารดา ที่ให้ความรัก กำลังใจ การอบรมเลี้ยงดูและส่งเสริมการศึกษาเป็นอย่างดีมาโดยตลอด รวมถึงเป็นกำลังใจอันยิ่งใหญ่แก่ผู้วิจัย จนทำให้ผู้วิจัยประสบความสำเร็จในชีวิตเรื่อยมา

ลักษมี โขมโนทัย

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

ตารางที่ **หน้า**

สารบัญตาราง (ต่อ)

ตารางที่ **หน้า**

สารบัญตาราง (ต่อ)

ตารางที่ **หน้า**

สารบัญรูป

รูปที่ **หน้า**

สารบัญรูป (ต่อ)

รูปที่ **หน้า**

สารบัญรูป (ต่อ)

รูปที่ **หน้า**

สารบัญรูป (ต่อ)

รูปที่ **หน้า**

สารบัญรูป (ต่อ)

รูปที่ **หน้า**

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มาของปัญหาการวิจัย

ในปัจจุบัน สภาวะการแข่งขันเพื่อให้ได้ชัยชนะทางธุรกิจจำเป็นต้องมีกลยุทธ์หรือยุทธวิธี (Business strategies) ที่เชื่อมั่นได้ว่าจะลดความเสี่ยงขององค์กรลงได้ กลยุทธ์วิธีการต่างๆจำเป็นต้องมีฐานความรู้ (Knowledge base) เพื่อใช้ในการสร้างกรอบการทำงาน (Framework) ที่สนองตอบกับกลยุทธ์ทางธุรกิจ การที่จะได้มาซึ่งฐานความรู้และกรอบการทำงานที่มีประโยชน์ จำเป็นต้องมีเทคโนโลยีสารสนเทศที่สามารถวิเคราะห์ข้อมูลทางธุรกิจ เทคโนโลยีสารสนเทศที่สามารถกลั่นกรองข้อมูลทางธุรกิจที่มีปริมาณมหาศาล เพื่อให้ได้ข้อมูลที่มีประโยชน์ในการคิดกลยุทธ์ (Useful information from very large data) ดังนั้นในขณะนี้การทำเหมืองข้อมูล (Data mining) จึงเป็นเทคโนโลยีสารสนเทศที่ได้รับการกล่าวถึงมากที่สุด คำตอบที่สำคัญสำหรับคำถามที่ว่าทำไมถึงมีการทำเหมืองข้อมูลและทำไมถึงต้องทำเหมืองข้อมูลนั้นก็เพราะว่า การทำเหมืองข้อมูลเป็นเทคโนโลยีสารสนเทศที่สามารถกลั่นกรอง วิเคราะห์ข้อมูลที่มีปริมาณมหาศาลเพื่อให้ได้ข้อมูลที่มีประโยชน์หรือได้ข้อมูลที่ซ่อนเร้นอยู่ในข้อมูลที่มีปริมาณมหาศาล และนำข้อมูลที่มีประโยชน์มีใช้เป็นฐานความรู้เพื่อช่วยในการบริหารงาน เช่น การบริหาร CRM (Customer Relationship Management)

การทำเหมืองข้อมูล คือกระบวนการทำงานที่เรียกว่า โปรเซส (Process) ที่สกัดข้อมูล (Extract data) จากฐานข้อมูลขนาดใหญ่ (Large database) เพื่อให้ได้สารสนเทศ (Useful information) ที่เรายังไม่รู้ (Unknown data) โดยเป็นสารสนเทศที่ถูกต้อง (Valid) และสามารถนำไปใช้ได้ (Actionable) ซึ่งเป็นสิ่งสำคัญในการที่จะช่วยการตัดสินใจในการทำธุรกิจ การทำเหมืองข้อมูลเป็นโปรเซสที่สำคัญในการทำ “การค้นหาความรู้จากฐานข้อมูล” (Knowledge Discovery in Databases) ที่เราเรียกสั้นๆว่า KDD

เทคนิคในการทำเหมืองข้อมูลที่นิยมใช้ในการทำเหมืองข้อมูลกับฐานข้อมูลขนาดใหญ่ทางด้านธุรกิจ คือ “การค้นหากฎความสัมพันธ์” (Association rule mining) เพื่อใช้ในการค้นหาความสัมพันธ์ของข้อมูล การค้นหากฎความสัมพันธ์ในเชิงธุรกิจทำให้เราสามารถทราบพฤติกรรมและความต้องการของผู้บริโภค โดยความรู้ที่ได้จากการทำเหมืองข้อมูลลักษณะนี้ คือ กฎความสัมพันธ์โดยจะแสดงในรูปแบบที่สามารถเข้าใจได้ง่าย คือ “สาเหตุไปสู่ผลลัพธ์” หลังจากนั้น

เราสามารถนำความรู้ที่ได้มาใช้ในกระบวนการตัดสินใจในการบริหารจัดการเกี่ยวกับสินค้าได้อย่างมีประสิทธิภาพ

การค้นหากฎความสัมพันธ์ (Association rule mining) เป็นเทคนิคหนึ่งของ การทำเหมืองข้อมูลที่สำคัญ และสามารถนำไปประยุกต์ใช้ได้จริงกับงานต่างๆ หลักการทำงานของวิธีนี้ คือ การค้นหาความสัมพันธ์ของข้อมูลจากฐานข้อมูลขนาดใหญ่ที่มีอยู่เพื่อนำไปใช้ในการวิเคราะห์ หรือทำนายปรากฏการณ์ต่าง ๆ การใช้งานส่วนใหญ่จะเป็นการวิเคราะห์การซื้อสินค้าของลูกค้าเรียกว่า “Market Basket Analysis” ซึ่งประเมินจากข้อมูลในฐานข้อมูลที่รวบรวมไว้ ผลการวิเคราะห์ที่ได้จะเป็นคำตอบของปัญหา ซึ่งการวิเคราะห์แบบนี้เป็นการใช้ “กฎความสัมพันธ์” เพื่อหาความสัมพันธ์ของข้อมูล ตัวอย่างการนำเทคนิคนี้ไปประยุกต์ใช้กับงานจริง ได้แก่ ระบบแนะนำหนังสือให้กับลูกค้าแบบอัตโนมัติ ของ Amazon ข้อมูลการสั่งซื้อทั้งหมดของ Amazon ซึ่งมีขนาดใหญ่มากจะถูกนำมาประมวลผลเพื่อหาความสัมพันธ์ของข้อมูล คือ ลูกค้าที่ซื้อหนังสือเล่มหนึ่ง ๆ มักจะซื้อหนังสือเล่มใดพร้อมกันด้วยเสมอ ความสัมพันธ์ที่ได้จากกระบวนการนี้จะสามารถนำไปใช้คาดเดาได้ว่าควรแนะนำหนังสือเล่มใดเพิ่มเติมให้กับลูกค้าที่เพิ่งซื้อหนังสือจากร้าน ตัวอย่างเช่น ซื้อ (x , database) $\rightarrow$ ซื้อ (x , data mining) [80% , 60%] หมายความว่า เมื่อลูกค้า x ซื้อหนังสือ database แล้วมีโอกาสที่ลูกค้า x จะซื้อหนังสือ data mining ด้วย 60 % จากสถิติการซื้อทั้งหนังสือ database และหนังสือ data mining พร้อม ๆ กัน 80 %

อีกตัวอย่างคือ ในการซื้อสินค้าของลูกค้า 1 ครั้ง โดยไม่ต้องจำกัดว่าจะซื้อสินค้าในห้างร้านหรือสั่งผ่านทางไปรษณีย์ หรือการซื้อสินค้าจากร้านค้าเสมือนจริง (Virtual store) บน web โดยปกติเราจะต้องการทราบว่าสินค้าใดบ้างที่ลูกค้ามักซื้อด้วยกัน เพื่อนำไปพิจารณาปรับปรุงการจัดวางสินค้าในร้าน หรือใช้เพื่อหาวิธีวางรูปคู่กันในใบโฆษณาสินค้า ก่อนอื่นขอกำหนดคำว่า กลุ่มรายการ (item set) หมายถึง กลุ่มสินค้าที่ปรากฏร่วมกัน เช่น {รองเท้า , ถุงเท้า}, {ปากกา, หมึก} หรือ {นม , น้ำผลไม้} โดยกลุ่มรายการดังกล่าวนี้ อาจจะจับคู่กลุ่มลูกค้ากับสินค้าก็ได้เช่น วิเคราะห์หา “ลูกค้าที่ซื้อสินค้าบางชนิดซ้ำ ๆ กัน อย่างน้อย 5 ครั้งแล้ว ” กรณีนี้ฐานข้อมูลเรามีการเก็บรายการซื้อขายเป็นจำนวนมาก และคำถามข้างต้น (query) นี้จำเป็นต้องค้นหาทุกๆ คู่ของลูกค้ากับสินค้า เช่น {คุณ ก , สินค้า A} , {คุณ ก , สินค้าB} , {คุณ ก , สินค้า C} , {คุณ ข , สินค้า B} เป็นต้น นับเป็นงานที่หนักพอสมควรสำหรับ DBMS เนื่องจากมีข้อมูลที่ต้องตรวจสอบมากมายหลายคู่และแต่ละคู่ต้องค้นหามาจากฐานข้อมูลโดยตรง แต่ผลลัพธ์ของ query แบบนี้ มักจะมีจำนวนน้อยมาก จึงเรียก query ชนิดนี้ว่าเป็น “iceberg query” ซึ่งเปรียบกับสำนวนไทย คือ “งมเข็มในมหาสมุทร”

สำหรับอัลกอริทึมที่นิยมนำมาใช้ในการค้นหากฎความสัมพันธ์ของข้อมูลคือ อัลกอริทึม เอไพรออริ (APRIORI algorithm) โดยอัลกอริทึมนี้จะทำการค้นหาความสัมพันธ์ที่เกิดขึ้นซ้ำๆ จากชุด

ข้อมูลที่ใส่เข้าไป ในกรณีที่ชุดข้อมูลมีขนาดเล็ก อัลกอริทึมจะสามารถทำการค้นหาความสัมพันธ์ได้อย่างรวดเร็ว ไม่ซับซ้อน แต่ถ้าชุดข้อมูลมีขนาดใหญ่ อัลกอริทึมก็จะใช้เวลาในการค้นหาความสัมพันธ์ของข้อมูลเพิ่มมากขึ้น และกฎความสัมพันธ์จะมีความซับซ้อนมากยิ่งขึ้น

สำหรับงานวิจัยนี้จะทำการพัฒนาวิธีการเพิ่มความเร็วในการค้นหากฎความสัมพันธ์ของข้อมูลที่มีปริมาณมาก โดยใช้เทคนิคการสุ่มข้อมูล เพื่อให้อัลกอริทึมในการขุดค้นความรู้สามารถทำการค้นหากฎความสัมพันธ์ที่มีความแข็งแกร่งจากชุดข้อมูลที่มีขนาดใหญ่ได้ในระยะเวลาอันรวดเร็วและมีความถูกต้อง กฎที่มีความแข็งแกร่ง หมายถึง กฎความสัมพันธ์ที่ปรากฏขึ้นในลำดับแรกๆ และปรากฏขึ้นซ้ำด้วยความถี่สูงถึง 80 เปอร์เซ็นต์ และมีค่าความเชื่อมั่นของกฎความสัมพันธ์สูงถึง 95 เปอร์เซ็นต์ โดยงานวิจัยฉบับนี้จะเน้นเกี่ยวกับการค้นหากฎความสัมพันธ์ที่มีความแข็งแกร่ง และลดระยะเวลาในการทำงานของอัลกอริทึมในการขุดค้นข้อมูล โดยกำหนดให้อัลกอริทึมทำการค้นหากฎความสัมพันธ์ที่มีข้อมูลสนับสนุนสูงถึง 80 เปอร์เซ็นต์ และมีความถูกต้องสูงถึง 95 เปอร์เซ็นต์ และใช้เทคนิคการสุ่มข้อมูลเพื่อช่วยลดระยะเวลาในการทำงานของอัลกอริทึม โดยกฎความสัมพันธ์ที่แข็งแกร่งที่ถูกค้นพบจากอัลกอริทึม เมื่อนำมาเปรียบเทียบกับกฎความสัมพันธ์ที่แข็งแกร่งที่ค้นพบในชุดข้อมูลทั้งหมดจะต้องมีความถูกต้องตรงกันสูงถึงเกณฑ์ที่กำหนด

1.2 วัตถุประสงค์ของการวิจัย

1.2.1 เพื่อศึกษาค้นคว้าและวิเคราะห์อัลกอริทึมเอไพรออริที่ใช้ในการค้นหากฎความสัมพันธ์ที่แข็งแกร่ง และวิธีการลดระยะเวลาในการทำงานของอัลกอริทึมในการขุดค้นความรู้โดยใช้เทคนิคการสุ่มข้อมูล

1.2.2 เพื่อศึกษาวิธีการสุ่มตัวอย่างที่น่าสนใจ และสามารถนำมาประยุกต์ใช้ในการพัฒนาอัลกอริทึมเอไพรออริให้สามารถลดระยะเวลาในการทำงานได้อย่างมีประสิทธิภาพและมีความถูกต้องสูงสุด

1.2.3 เพื่อค้นหาเทคนิคที่ช่วยปรับปรุงลักษณะการทำงานของอัลกอริทึมให้สามารถทำงานกับข้อมูลที่มีลักษณะเป็นแบบสตรีมมิง (Streaming data) ได้

1.2.4 สามารถสรุปลักษณะของเทคนิคที่ใช้ในการสุ่มข้อมูลและเทคนิคที่เหมาะสมที่ช่วยให้อัลกอริทึมในการค้นหากฎความสัมพันธ์ให้สามารถทำงานกับข้อมูลที่มีลักษณะเป็นแบบสตรีมมิง เพื่อเป็นแนวทางในการพัฒนาอัลกอริทึมในการค้นหากฎความสัมพันธ์ที่เหมาะสมกับข้อมูลแบบสตรีมมิงต่อไปในอนาคต

1.3 ขอบเขตของงานวิจัย

1.3.1 ศึกษาวิธีการค้นหากฎความสัมพันธ์โดยใช้อัลกอริทึมเอไพรออริ

1.3.2 งานวิจัยนี้เน้นการพัฒนาอัลกอริทึมเอไพรออริด้วยภาษา SQL และทำงานกับฐานข้อมูลเชิงสัมพันธ์

1.3.3 โปรแกรมที่ทำการพัฒนาขึ้นมุ่งเน้นในการทดสอบกับข้อมูลเชิงธุรกิจ

1.3.4 เกณฑ์ที่ใช้ศึกษาเพื่อเปรียบเทียบประสิทธิภาพการทำงานของโปรแกรม ประกอบด้วย ขนาดของตัวอย่างที่ถูกสุ่มมาใช้ในกระบวนการ, ระยะเวลาการทำงานของอัลกอริทึม และจำนวนกฎความสัมพันธ์ที่ถูกค้นพบจากอัลกอริทึม

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1.4.1 สามารถทำการค้นหากฎความสัมพันธ์ที่แข็งแกร่งจากฐานข้อมูลขนาดใหญ่ได้ในระยะ เวลาอันรวดเร็ว

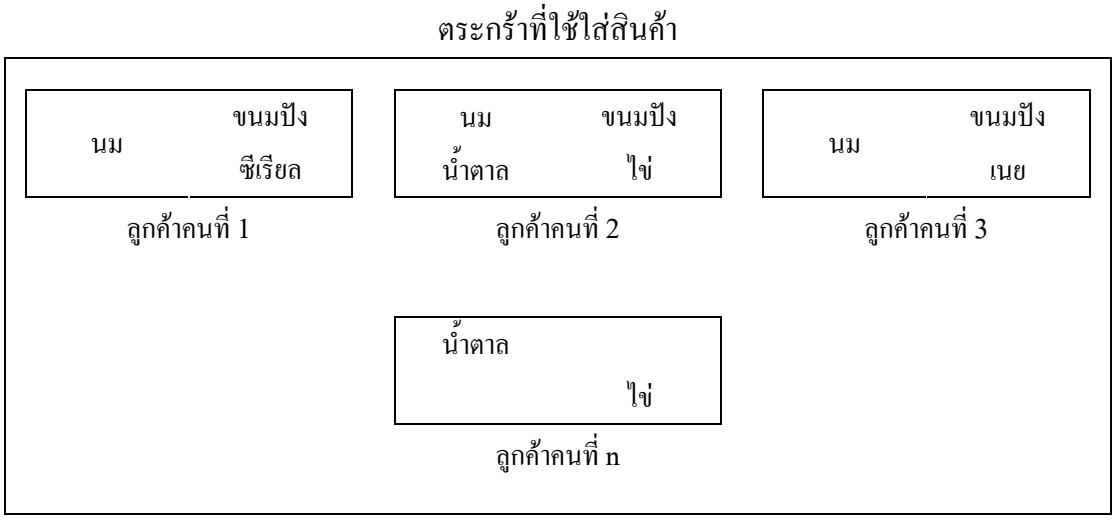
1.4.2 ได้โปรแกรมที่สามารถรองรับการค้นหากฎความสัมพันธ์กับข้อมูลที่เป็นแบบสตรีมมิง

1.4.3 ความรู้ที่ได้จากงานวิจัยนี้ สามารถใช้เป็นแนวทางในการพัฒนาการทำเหมืองข้อมูลในการค้นหากฎความสัมพันธ์ที่เหมาะสมกับข้อมูลแบบสตรีมมิงต่อไปในอนาคต

บทที่ 2
ปริทัศน์วรรณกรรมและงานวิจัยที่เกี่ยวข้อง

2.1 การค้นหากฎความสัมพันธ์

การค้นหากฎความสัมพันธ์ของข้อมูลในฐานข้อมูลขนาดใหญ่ เป็นงานสำคัญงานหนึ่งในการทำเหมืองข้อมูล โดยจะทำการค้นหาความสัมพันธ์ที่น่าสนใจระหว่างไอเท็มเซต การค้นหากฎความสัมพันธ์นั้นนิยมใช้งานกับข้อมูลทรานแซคชันทางด้านธุรกิจ (Business transaction) โดยกฎความสัมพันธ์ที่ได้มีส่วนช่วยในกระบวนการตัดสินใจทางด้านธุรกิจ ตัวอย่างของการค้นหากฎความสัมพันธ์คือการวิเคราะห์พฤติกรรมของผู้บริโภค โดยการหาความสัมพันธ์ระหว่างสินค้าที่ต่างชนิดกันแต่ลูกค้าจะเลือกซื้อไปพร้อมกัน ดังรูปที่ 2.1 โดยการค้นหากฎความสัมพันธ์นี้จะช่วยทำให้ทราบว่าสินค้าใดถูกลูกค้าเลือกซื้อไปคู่กันมากที่สุด และนำกฎความสัมพันธ์ที่ค้นพบนั้นมาใช้ในการปรับปรุงกลยุทธ์การขายสินค้าหรือใช้ประกอบการพิจารณาในการจัดวางสินค้า ทำให้เกิดความสะดวกในการเลือกซื้อสินค้า และเป็นการเพิ่มยอดจำหน่ายให้กับร้านค้า



รูปที่ 2.1 การวิเคราะห์พฤติกรรมการซื้อสินค้า

กฎความสัมพันธ์สามารถเขียนให้อยู่ในรูปไอเท็มเซตที่เป็นเหตุ ไปสู่ไอเท็มเซตที่เป็นผล เช่น ลูกค้าที่ซื้อนมส่วนใหญ่จะซื้อขนมปังด้วย ก็สามารถเขียนความสัมพันธ์ได้เป็น {นม} $\rightarrow$ {ขนมปัง} เป็นต้น การอธิบายวิธีการค้นหากฎความสัมพันธ์จะต้องใช้นิยามดังต่อไปนี้

1) ไอเท็มเซต (Itemsets : I) คือเซตที่มีไอเท็มทั้งหมดเป็นสมาชิก ซึ่งไอเท็มในที่นี้อาจเป็นชื่อของหน่วยใดๆ ที่นำมาใช้ในการเรียนรู้
2) ทรานแซคชัน (Transaction : T) เป็นเซตย่อยของไอเท็ม โดยที่ $T \subseteq I$
3) เซตของข้อมูล (Data : D) คือเซตที่มีทรานแซคชันทุกตัวเป็นสมาชิก โดยที่ $D = \{t_1, t_2,\ldots, t_3\}$
4) การกำหนดค่าสนับสนุน (Support) s_{min}, โดยที่ $0 \leq s_{min} \leq 1$
5) การกำหนดค่าความเชื่อมั่น (Confidence) c_{min}, โดยที่ $0 \leq c_{min} \leq 1$
6) ทรานแซคชัน T บรรจุเซตย่อยของไอเท็ม X ก็ต่อเมื่อ $X \subseteq t$
7) ค่าสนับสนุนของไอเท็มเซต X คือ $s(X) = \left| \{t \in D \mid X \subseteq t\} \right| / \left| D \right|$
8) กฎที่ได้อยู่ในรูปแบบ $L \rightarrow R$ เมื่อ $L \subset I$, $R \subset I$ และ $L \cap R = \varnothing$
9) ค่าความเชื่อมั่นของกฎ $L \rightarrow R$ คือ $c(L, R) = s(L \cup R) / s(L)$
10) ค่าสนับสนุนของกฎ $L \rightarrow R$ คือ $s(L, R) = s(L \cup R)$
11) ทุกกฎความสัมพันธ์ $L \rightarrow R$ เมื่อ $L \subset I$, $R \subset I$ และ $L \cap R = \varnothing$ แล้วจะต้องประกอบไปด้วยค่าสนับสนุนและค่าความเชื่อมั่น ซึ่งนิยามได้ดังนี้
    (1) กฎความสัมพันธ์ $L \rightarrow R$ มีค่าสนับสนุนเท่ากับ s ก็ต่อเมื่อ s% ของทรานแซคชันใน D บรรจุ $L \cup R$
    (2) กฎความสัมพันธ์ $L \rightarrow R$ มีค่าความเชื่อมั่นเท่ากับ c ก็ต่อเมื่อ c% ของทรานแซคชันใน D บรรจุ $L \cup R$
12) แต่ละกฎที่ถูกค้นพบเรียกว่ากฎความสัมพันธ์ โดยการค้นหากฎความสัมพันธ์ คือการค้นหาความสัมพันธ์ทั้งหมดในทรานแซคชันทุกตัว โดยกฎความสัมพันธ์ที่หาได้ทั้งหมดจะต้องมีค่าสนับสนุนมากกว่าหรือเท่ากับค่าสนับสนุนขั้นต่ำที่กำหนดไว้ ($s(L \cup R) \geq s_{min}$) และมีค่าความเชื่อมั่นมากกว่าหรือเท่ากับค่าความเชื่อมั่นขั้นต่ำที่กำหนดไว้ ($c(L, R) \geq c_{min}$) โดยกฎความสัมพันธ์ที่มีค่าทั้งสองสูงถึงเกณฑ์ที่กำหนดจะถูกเรียกว่า “แข็งแกร่ง” (Strong)
13) ค่าสนับสนุนเรียกว่า “การปรากฏบ่อย”

Database D

tid	Item
101	beer
101	cheese
102	beer
102	milk
102	tomatoes
103	cheese

$s_{min} = 0.2$
$c_{min} = 0.5$

จะได้กฎความสัมพันธ์ดังนี้

{beer} → {milk, tomatoes}
ค่าสนับสนุน = s({beer, milk, tomatoes})
= 1/3
= 33.33%
ค่าความเชื่อมั่น = 1/2
= 50%

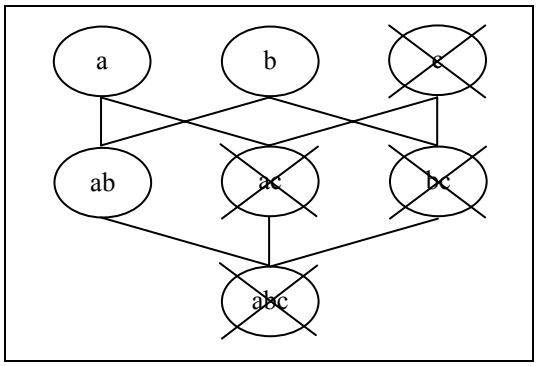
รูปที่ 2.2 แสดงตัวอย่างการค้นหากฎความสัมพันธ์

2.2 วิธีการค้นหากฎความสัมพันธ์

การค้นหากฎความสัมพันธ์สามารถแบ่งออกเป็น 2 ขั้นตอนคือ

2.2.1 การหาไอเท็มเซตที่ปรากฏบ่อย (Frequent itemsets)

การค้นหาไอเท็มเซตที่ปรากฏบ่อยๆ เป็นการค้นหาการจัดหมู่ของไอเท็มเซตทั้ง หมด ซึ่งจำนวนการจัดหมู่จะมีขนาดเพิ่มขึ้นตามจำนวนของไอเท็มเซต ยิ่งถ้าไอเท็มเซตมีขนาดใหญ่ขึ้นไอเท็มเซตที่ปรากฏบ่อยก็จะมีเซตที่ต้องค้นหามากขึ้น แต่การค้นหาไม่จำเป็นที่จะต้องแจกแจงค้นหาในทุกการจัดหมู่เพราะถ้าทำการแจกแจงแล้วพบไอเท็มเซตใดที่ไม่ใช่ไอเท็มเซตที่ปรากฏบ่อยก็ไม่ต้องแจกแจงไอเท็มเซตอื่นๆ ที่มีไอเท็มเซตนี้เป็นเซตย่อยอีก เช่น จากไอเท็ม a, b และ c สามารถสร้างไอเท็มเซตทั้งหมดได้ดังรูปที่ 2.3 ถ้ารู้ว่า {c} ไม่ได้เป็นไอเท็มเซตที่ปรากฏบ่อยแล้วก็ไม่ต้องสร้าง {ac}, {bc} และ {abc} ซึ่งมี {c} เป็นเซตย่อย



รูปที่ 2.3 การจัดหมู่ของสมาชิกในไอเท็มเซต {a, b, c}

ไอเท็มเซตที่ปรากฏขึ้นบ่อยที่ถูกค้นพบนั้น จะต้องมีจำนวนการปรากฏขึ้นมากกว่าหรือเท่ากับค่าสนับสนุนขั้นต่ำที่กำหนดไว้

2.2.2 การสร้างกฎความสัมพันธ์จากไอเท็มเซตที่ปรากฏบ่อย

เมื่อได้ไอเท็มเซตที่ปรากฏบ่อยมาแล้ว จะต้องสร้างกฎความสัมพันธ์จากไอเท็มเซตที่ปรากฏบ่อย โดยกฎความสัมพันธ์ที่ได้จะต้องมีค่าความเชื่อมั่นมากกว่าค่าความเชื่อมั่นขั้นต่ำที่กำหนดไว้ โดยกฎความสัมพันธ์ทั้งหมดที่เป็นไปได้คือการจัดหมู่ทุกแบบของสมาชิกในไอเท็มเซตที่ปรากฏบ่อยการจัดหมู่อาจเป็นการจัดหมู่จากกฎความสัมพันธ์ส่วนที่เป็นเหตุ หรือการจัดหมู่ของกฎความสัมพันธ์ในส่วนที่เป็นผลก็ได้เช่น $(F - S) \rightarrow S$ เมื่อ S คือไอเท็มเซตของกฎความสัมพันธ์ส่วนที่เป็นผล และ F เป็นไอเท็มเซตที่ปรากฏบ่อย โดยที่ $S \subset F$ และ $S \neq \varnothing$

2.3 ประเภทของกฎความสัมพันธ์

การวิเคราะห์พฤติกรรมการซื้อสินค้าของผู้บริโภค เป็นอีกรูปแบบหนึ่งของการค้นหากฎความสัมพันธ์ ในความเป็นจริงกฎความสัมพันธ์มีอยู่หลายประเภท โดยกฎความสัมพันธ์สามารถจำแนกได้หลายแนวทางขึ้นอยู่กับเกณฑ์ที่ใช้ในการจำแนก (Han and Kamber, 2001)

1) กฎความสัมพันธ์ที่เป็นข้อเท็จจริง (Boolean association rule) คือ กฎความสัมพันธ์เกี่ยวกับความสัมพันธ์ระหว่างการมีอยู่หรือไม่มีอยู่ของไอเท็ม เช่น
   ขนมปัง $\rightarrow$ นม (2.1)
   เป็นกฎความสัมพันธ์ที่เป็นข้อเท็จจริงที่ได้มาจากการวิเคราะห์พฤติกรรมการซื้อสินค้าของผู้บริโภค
2) กฎความสัมพันธ์เกี่ยวกับปริมาณ (Quantitative association rule) คือ กฎความสัมพันธ์ที่อธิบายความสัมพันธ์ระหว่างปริมาณของไอเท็มหรือแอททริบิวต์ โดยในกฎจะมีค่าปริมาณของไอเท็มหรือแอททริบิวต์โดยจะถูกพิจารณาเป็นช่วงของข้อมูล เช่น
   อายุ(X, “17 .. 22”) $\wedge$ สีผม(X, “ดำ”) $\rightarrow$ รายได้(X, “1000 .. 2000”) (2.2)
   <u>หมายเหตุ</u> แอททริบิวต์ที่เป็นปริมาณคือ อายุ และรายได้
3) กฎความสัมพันธ์หนึ่งมิติ (Single-dimensional association rule) คือ กฎความสัมพันธ์ที่มีไอเท็มหรือแอททริบิวต์ภายในกฎความสัมพันธ์อ้างอิงข้อมูลเพียงหนึ่งมิติ เช่น กฎความสัมพันธ์ที่ (2.1) สามารถเขียนเป็นกฎความสัมพันธ์หนึ่งมิติได้ดังนี้
   ซื้อ(X, “ขนมปัง”) $\rightarrow$ ซื้อ(X, “นม”) (2.3)
   กฎความสัมพันธ์ที่ (2.1) เป็นกฎความสัมพันธ์หนึ่งมิติเพราะอ้างอิงข้อมูลเพียงแค่หนึ่งมิติเท่านั้นคือมิติ “ซื้อ”

4) กฎความสัมพันธ์หลายมิติ (Multidimensional association rule) คือ กฎความ-สัมพันธ์ที่มีไอเท็มหรือแอททริบิวต์ภายในกฎความสัมพันธ์ มีการอ้างอิงมิติของข้อมูลมากกว่าสองมิติขึ้นไป กฎความสัมพันธ์ที่ (2.2) สามารถพิจารณาได้ว่าเป็นกฎความสัมพันธ์หลายมิติเนื่องจากมีการอ้างถึงข้อมูลสามมิติ คือ อายุ, สีผม และรายได้

5) กฎความสัมพันธ์หลายระดับ (Multilevel association rule) เนื่องจากการค้นหากฎความสัมพันธ์บางวิธีสามารถค้นหากฎความสัมพันธ์ที่มีระดับของนามธรรมที่แตกต่างกัน คือชุดของกฎความสัมพันธ์ที่มีกฎความสัมพันธ์อื่นตามมาด้วย เช่น

   อายุ(X, “30 .. 39”) $\longrightarrow$ ซื้อ(X, “เบียร์”) (2.4)

   อายุ(X, “30 .. 39”) $\longrightarrow$ ซื้อ(X, “เครื่องดื่มแอลกอฮอล์”) (2.5)

   กฎความสัมพันธ์ที่ 2.4 และ 2.5 ไอเท็มที่ถูกอ้างอิงอยู่ในระดับที่แตกต่างกัน (“เครื่องดื่มแอลกอฮอล์” อยู่ในระดับที่สูงกว่า “เบียร์”)

6) กฎความสัมพันธ์ระดับเดียว (Single-level association rule) จะคล้ายกับกฎความสัมพันธ์หลายระดับแตกต่างกันเพียงกฎความสัมพันธ์ระดับเดียวจะมีการอ้างอิงข้อมูลที่อยู่ในระดับเดียวกัน เช่น

   อายุ(X, “30 .. 39”) $\longrightarrow$ ซื้อ(X, “เบียร์”) (2.6)

   อายุ(X, “30 .. 39”) $\longrightarrow$ ซื้อ(X, “เหล้า”) (2.7)

   กฎความสัมพันธ์ที่ 2.6 และ 2.7 ไอเท็มที่ถูกอ้างอิงอยู่ในระดับเดียวกัน (“เหล้า” อยู่ในระดับเดียวกันกับ “เบียร์”)

2.4 การค้นหากฎความสัมพันธ์ของข้อมูลในฐานข้อมูลขนาดใหญ่

Agrawal, Imielinski และ Swami (1993) ได้เสนอแนวคิดการทำเหมืองข้อมูลโดยการค้นหากฎความสัมพันธ์ของข้อมูลจากฐานข้อมูลขนาดใหญ่โดยใช้รายการทรานแซคชัน (Transac-tion) ของลูกค้า โดยแต่ละรายการทรานแซคชันเป็นการสั่งสินค้าของลูกค้า อัลกอริทึมที่นำมาใช้ในการค้นหาความสัมพันธ์นี้คือ อัลกอริทึม เอไอเอส (AIS algorithm) โดยอัลกอริทึมถูกนำเสนอในส่วนการใช้งานในการสร้างกฎความสัมพันธ์ โดยกฎความสัมพันธ์จะอยู่ในรูปแบบ $I_k \longrightarrow I_l$ โดยที่ I_k เป็นชุดข้อมูลที่ k, I_l เป็นชุดข้อมูลที่ l และ $I_k \cap I_l = \varnothing$, k = 1, 2, 3, …. อัลกอริทึมดังกล่าวจะทำการอ่านข้อมูลหลายรอบ โดยแต่ละรอบจะอ่านข้อมูลทรานแซคชันทั้งหมดในฐานข้อมูล ไอเท็มเซตจะถูกสร้างขึ้นจากข้อมูลแต่ละเรคคอร์ดที่ถูกอ่านเข้ามา โดยอาจจะเป็นข้อมูลเพียงบางส่วนหรืออาจเป็นข้อมูลทั้งหมดของรายการสินค้า ส่วนแคนดิเดทไอเท็มเซต ถูกสร้างขึ้นเพื่อใช้ในการ

ตรวจสอบค่าสนับสนุนของกฎความสัมพันธ์ที่ได้มา เมื่อมีการสร้างรอบการอ่านข้อมูลขึ้นมาใหม่ชุดข้อมูลชุดใหม่จะถูกสร้างขึ้นมาจากไอเท็มเซตที่ปรากฏบ่อยจากรอบการอ่านที่แล้ว

2.5 อัลกอริทึมเอไพรออริ

Agrawal และ Srikant (1994) ได้เสนออัลกอริทึมใหม่ที่ใช้ในการทำเหมืองข้อมูลโดยการค้นหากฎความสัมพันธ์คือ อัลกอริทึม เอไพรออริ (APRIORI) โดยอัลกอริทึมที่นำเสนอนี้ได้รับการยอมรับทางด้านสมรรถนะเหนือกว่าอัลกอริทึมอื่นๆ ก่อนหน้านี้ โดยอัลกอริทึมตัวใหม่นี้จะมีความแตกต่างจากอัลกอริทึม AIS ทางด้านเทคนิคที่ใช้ในการสร้างแคนดิเดทไอเท็ม และการคัดเลือกแคนดิเดทไอเท็มเพื่อนำมาใช้ในการนับค่าสนับสนุน โดยแคนดิเดทไอเท็มจะถูกสร้างจากไอเท็มเซตที่ปรากฏบ่อยในรอบที่ผ่านมา จะสร้างขึ้นโดยการนำไอเท็มเซตที่ปรากฏบ่อยในรอบที่ผ่านมามาเชื่อมกัน และทำการลบไอเท็มเซตของเซตย่อยที่เกิดจากการเชื่อมกันของไอเท็มเซตที่ปรากฏบ่อยที่ไม่ได้ปรากฏอยู่ในไอเท็มเซตที่ปรากฏบ่อยในรอบที่ผ่านมา การทำงานในลักษณะเช่นนี้ทำให้อัลกอริทึมนี้จะต้องมีการอ่านข้อมูลจากฐานข้อมูลซ้ำกันหลายรอบ

อัลกอริทึมเอไพรออริเป็นอัลกอริทึมที่นิยมใช้กันอย่างแพร่หลาย โดยอัลกอริทึมจะทำการค้นหาแบบแนวกว้าง โดยจะทำการสร้างและตรวจสอบไอเท็มเซตที่ปรากฏขึ้นบ่อยทีละขั้น ชื่อของอัลกอริทึมถูกตั้งขึ้นโดยใช้พื้นฐานการทำงานของอัลกอริทึม เริ่มจากไอเท็มเซตที่มีจำนวนสมาชิกเท่ากับหนึ่ง ถ้าไอเท็มเซตใดมีค่าสนับสนุนน้อยกว่าค่าสนับสนุนที่กำหนดก็จะตัดไอเท็มเซตนั้นออกไม่นำไปสร้างไอเท็มเซตในขั้นถัดไป และการทำงานของอัลกอริทึมจะวนอย่างนี้ไปเรื่อยๆ จนกว่าจะไม่เหลือเซตของไอเท็มที่จะสร้างในขั้นถัดไป ในการนับจำนวนทรานแซคชัน อัลกอริทึมเอไพรออริจะอ่านข้อมูลทรานแซคชันเพียงครั้งเดียวในแต่ละระดับชั้นในการค้นหาไอเท็มเซตที่ปรากฏบ่อยอัลกอริทึมเอไพรออริจะมีกระบวนการที่สำคัญอยู่สองกระบวนการ (Two-step process) คือ

1) ขั้นตอนการรวม (The join step) ในการหาไอเท็มเซตที่ปรากฏบ่อย (F_k) เซตของแคนดิเดท ไอเท็มเซตที่ k (k-itemsets) จะถูกสร้างขึ้นโดยการเอา F_{k-1} รวมเข้ากับตัวมันเอง

2) ขั้นตอนการตัด (The prune step) C_k เป็นซุปเปอร์เซตของ F_k โดยที่สมาชิกใน C_k อาจจะปรากฏขึ้นบ่อยหรือไม่ก็ได้ แต่ไอเท็มเซตที่ k ที่ปรากฏขึ้นบ่อยจะต้องปรากฏอยู่ใน C_k การอ่านฐานข้อมูลแต่ละครั้งจะถูกกำหนดให้ทำการนับแต่ละแคนดิเดทใน C_k และจะได้เป็น F_k โดยแคนดิเดทที่จะเป็นผลลัพธ์ของ F_k จะต้องมีการปรากฏขึ้นบ่อยไม่น้อยกว่าค่าสนับสนุนขั้นต่ำ ส่วนแคนดิเดทที่มีค่าการปรากฏขึ้นบ่อยน้อยกว่าค่าสนับสนุนต่ำสุดจะถูกตัดทิ้งไปจาก C_k

การทำงานของอัลกอริทึมเอไพรออริแบ่งออกเป็น 3 ช่วงคือ การสร้างไอเท็มเซตที่ปรากฏบ่อย (รูปที่ 2.4) การสร้างแคนดิเดทไอเท็มเซต (รูปที่ 2.5) และการสร้างกฎความสัมพันธ์ (รูปที่ 2.6)

```
F      : result set of frequent itemsets (of different sizes)
F[k]   : set of frequent itemsets of size k (k-itemsets)
C[k]   : set of candidate itemsets of size k
SetOfItemsets generateFrequentItemsets(Integer minimumSupport) begin
    F[1] = {frequent items};
    for (k = 1; F[k] ≠ Ø; k++) do begin
      C[k+1] = generateCandidates(k, F[k]);
      for each transaction t ∈ database do
        for each candidate c ∈ C[k+1] do
          if c ⊆ t then
            c.count++;
      for each candidate c ∈ C[k+1] do
        if c.count ≥ minimumSupport then
            F[k+1] = F[k+1] ∪ {c};
    end;
    F = Ø;
    while k ≥ 1 do begin
      F = F ∪ F[k];
      k--;
    end;
    return F;
end;
```

รูปที่ 2.4 อัลกอริทึมเอไพรออริ

```
C   : result set of candidate itemsets of size k+1
c   : candidate itemset of size k+1
SetOfItemsets generateCandidates(Integer k; SetOfItemsets F) begin
    C = Ø;
    for each k-itemset f ∈ F do
      for each k-itemset g ∈ F do begin
        i = 1;
        while (i < k) and (f[i] = g[i]) do begin
          c[i] = f[i];
          i++;
        end;
        if (i = k) and (f[k] < g[k]) then begin
          c[k] = f[k];
          c[k+1] = g[k];
          if not hasInfrequentSubset(c, F) then
            // only those cand's are kept where all (k-1)-subsets are frequent
            C = C ∪ {c};
        end;
      end;
    return C;
end;
Boolean hasInfrequentSubset(Itemset c; SetOfItemsets F) begin
    for each (k-1)-subset s of c do
      if s ∉ F then
        return true;
    return false;
        end;
```

รูปที่ 2.5 อัลกอริทึมเอไพรออริ : การสร้างแคนดิเดทไอเท็มเซต (Candidate itemset)

```
Precondition: all frequent itemsets have been computed
each itemset has two attributes:
    count   : "support" as an integer
    size    : cardinality, i.e. number of items in itemset
for each frequent k-itemset f ∈ F where k ≥ 2 do
    generateRules(f, f);
// Computes all rules (g --> f\g) for all g ⊂ f.
generateRules(Itemset f, Itemset left) begin
            rules.add(a, f\a); // set of result association rules
            if (left.size-1 > 1) then
              generateRules(f, a);
          end;
      end;
end;
```

รูปที่ 2.6 อัลกอริทึมเอไพรออริ : การสร้างกฎความสัมพันธ์ (Compute association rules)

2.6 การค้นหากฎความสัมพันธ์จากข้อมูลในฐานข้อมูลเชิงสัมพันธ์

Sarawagi, Thomas และ Agrawal (1998) ได้เสนอแนวคิดการทำเหมืองข้อมูลโดยการค้นหาความสัมพันธ์ของข้อมูลรวมเข้าไว้กับฐานข้อมูลเชิงสัมพันธ์ โดยทำการพัฒนา อัลกอริทึม เอไพรออริด้วย SQL เพื่อใช้ในการทำเหมืองข้อมูลในฐานข้อมูลเชิงสัมพันธ์ โดยจะมีรูปแบบของข้อมูลที่จะนำเข้ามาทำเหมืองข้อมูลคือ จะมีตารางทรานแซคชัน T (Table T) ภายในตารางประกอบด้วย 2 คอลัมน์ คือ ตัวชี้ทรานแซคชัน (Transaction identifier : TID) และตัวชี้รายการข้อมูล (Item identifier : Item)

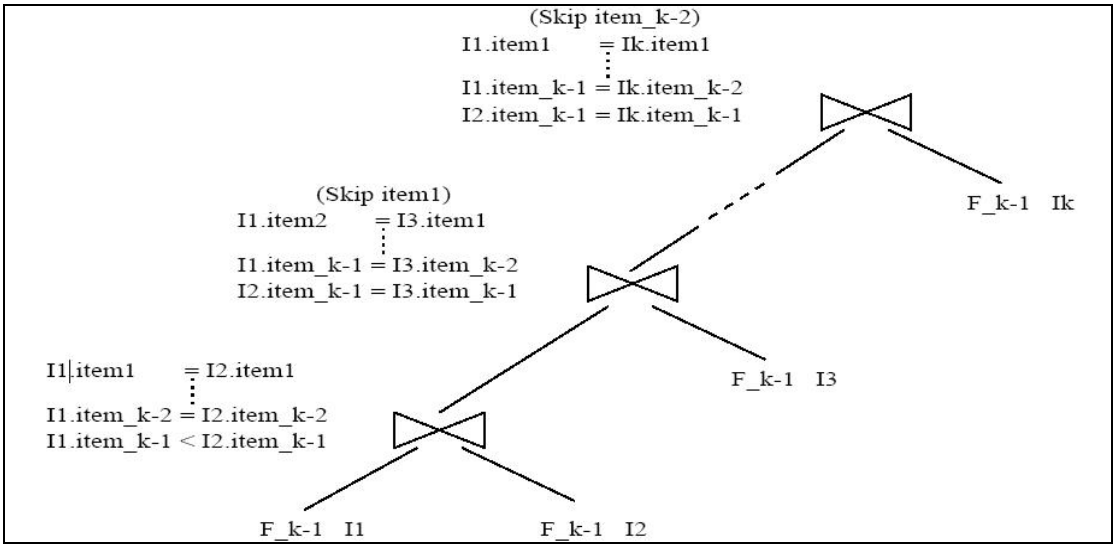
การสร้างแคนดิเดทไอเท็มเซต ใน SQL

ในแต่ละรอบของการอ่านข้อมูลที่ k ของอัลกอริทึมเอไพรออริจะมีการสร้างแคนดิเดทไอเท็มเซต C_k ขึ้นมาโดยสร้างจากไอเท็มเซตที่ปรากฏบ่อยตัวที่ F_k-1 ของรอบการอ่านข้อมูลที่ผ่านมาแล้วนำมารวมกันโดยนำซุปเปอร์เซต (Superset) ของแคนดิเดทไอเท็มเซตที่ถูกสร้างนำมารวมกับไอเท็มเซตที่ปรากฏบ่อยตัวที่ F_k-1 เช่น ให้ F_3 เป็น {{1 2 3}, {1 2 4}, {1 3 4}, {1 3 5}, {2 3 4}}

หลังจากที่นำมารวมกันแล้ว C_4 จะได้เป็น {{1 2 3 4}, {1 3 4 5}, {1 3 4}, {1 3 5}, {2 3 4}} ดังแสดงในรูปที่ 2.7 และรูปที่ 2.8

```
insert into  Ck select I1.item1, …, I1.itemk-1, I2.itemk-1
from         F k-1 I1, F k-1 I2
where        I1.item1   = I2.item1     and
                        :
             I1.itemk-2 = I2.itemk-2   and
             I1.itemk-1 < I2.itemk-1
```

รูปที่ 2.7 การสร้างแคนดิเดทไอเท็มเซตใน SQL



รูปที่ 2.8 แสดงการสร้างแคนดิเดทไอเท็มเซตสำหรับรอบการอ่านข้อมูลที่ k

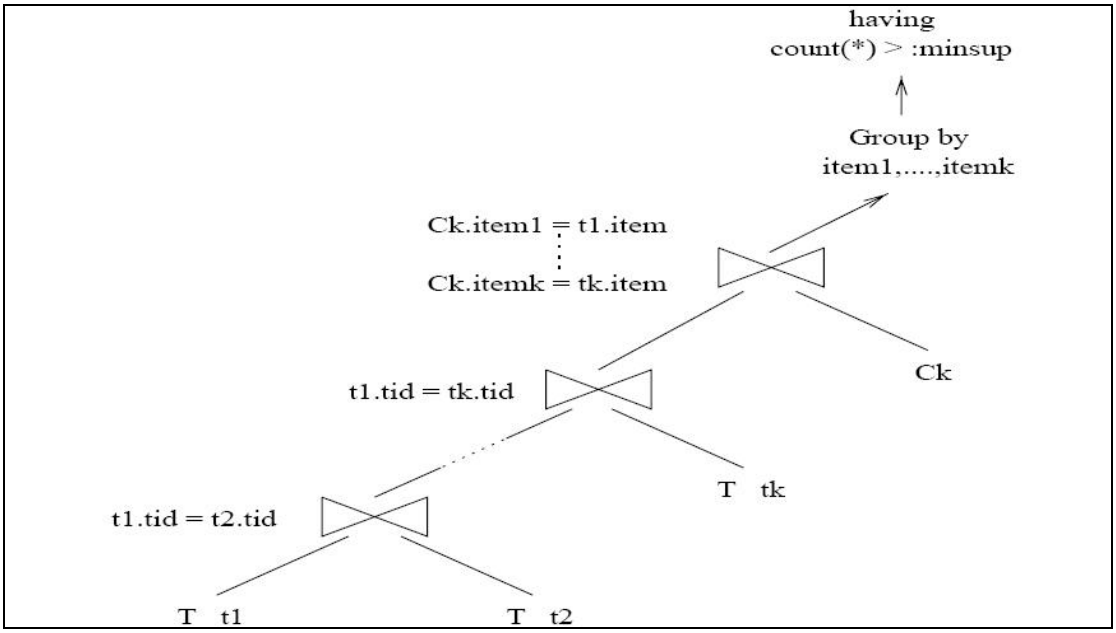
การนับค่าสนับสนุนเพื่อหาไอเท็มเซตที่ปรากฏบ่อย

ในการนับค่าสนับสนุนเพื่อหาไอเท็มเซตที่ปรากฏบ่อยจะใช้แคนดิเดทไอเท็มเซต C_k และข้อมูลในตารางทรานแซคชัน T ในการนับค่าสนับสนุน โดยเสนอวิธีการนับค่าสนับสนุนที่พัฒนาจาก SQL-92 คือในแต่ละรอบการอ่านอ่านข้อมูลที่ k จะทำการรวมแคนดิเดทไอเท็มเซต C_k กับชุด

ข้อมูลที่ k ในตารางทรานแซคชัน T แล้วจัดกลุ่มโดยใช้คำสั่ง group by ตามไอเท็มเซต ดังแสดงในรูปที่ 2.9 และรูปที่ 2.10

```
insert into  F_k select item_1, …, item_k, count(*)
from         C_k, T t_1, T t_k
where        t_1.item    = C_k.item_1    and
                         :
             t_k.item_k-2 = C_k.item_k    and
             t_1.tid     = t_2.tid       and
                         :
             t_k-1.tid     = t_k.tid
group by     item_1, …, item_k
having       count(*)    > :minsup
```

รูปที่ 2.9 การนับค่าสนับสนุน และการสร้างไอเท็มเซตที่ปรากฏบ่อยใน SQL



รูปที่ 2.10 แสดงการนับค่าสนับสนุน

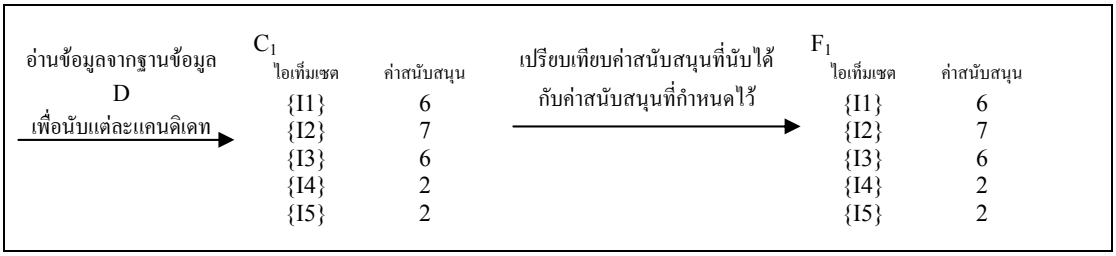
2.7 ตัวอย่างการค้นหากฎความสัมพันธ์

ในส่วนนี้จะแสดงกระบวนการค้นหากฎความสัมพันธ์จากตัวอย่างต่อไปนี้

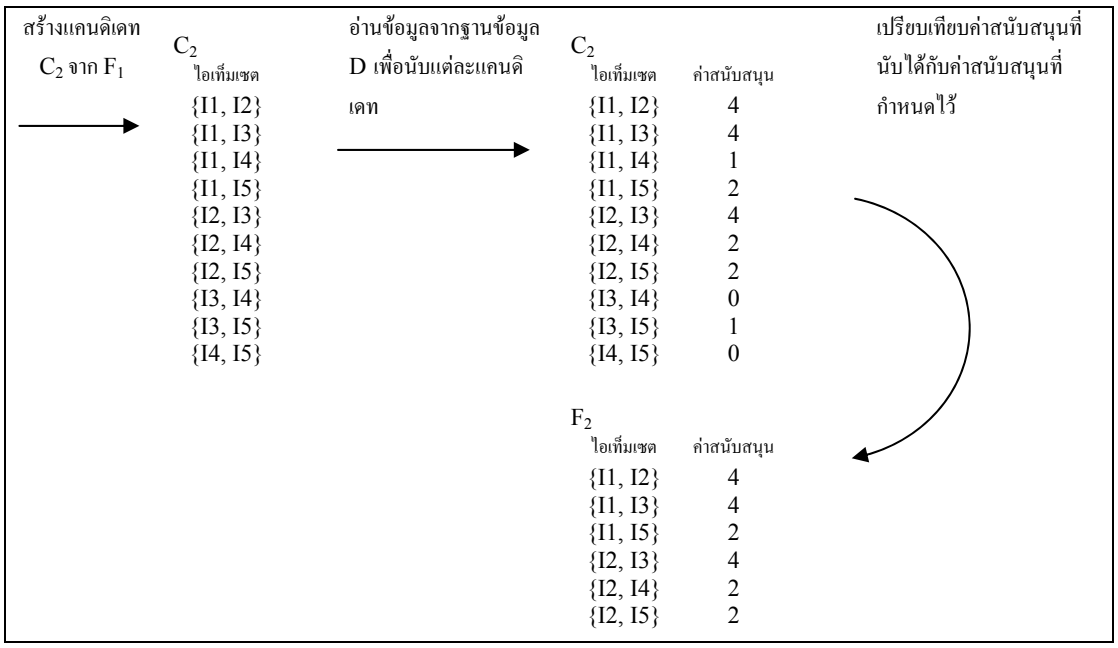
ตัวอย่าง พิจารณารายการทรานแซคชันการขายสินค้าของร้านสะดวกซื้อแห่งหนึ่ง (ฐานข้อมูล D) มีสินค้าจำนวน 5 ชนิด คือ I1, I2, I3, I4, I5 โดยรายการซื้อสินค้าของลูกค้าแสดงในตารางที่ 2.1 แต่ละแถวของตารางหมายถึงหนึ่งรายการทรานแซคชัน รูปที่ 2.11 เป็นภาพจำลองแสดงการทำงานของอัลกอริทึมในการค้นหาไอเท็มเซตที่ปรากฏบ่อยในฐานข้อมูล D โดยกำหนดค่าสนับสนุนขั้นต่ำที่ 22 เปอร์เซ็นต์ หรือมีปรากฏขึ้น 2 ครั้งจากจำนวนทรานแซคชัน 9 ทรานแซคชัน (2/9 = 22%)

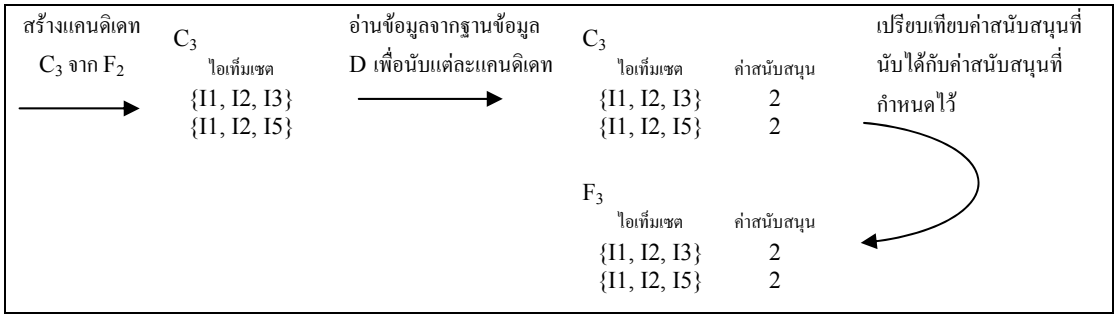
ตารางที่ 2.1 ข้อมูลทรานแซคชันในฐานข้อมูล D

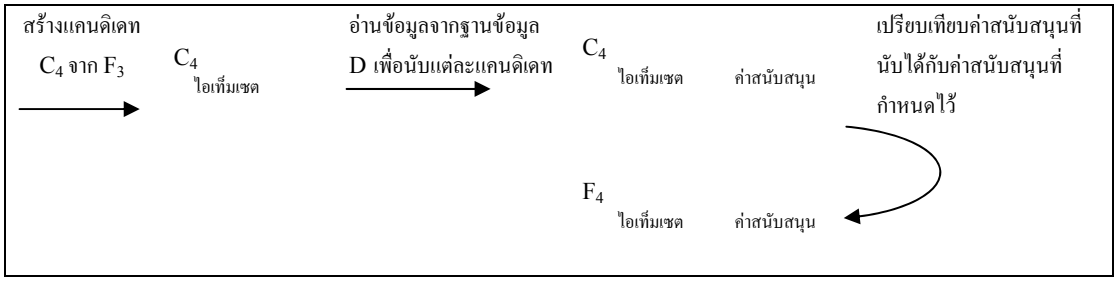
TID	Itemset
T100	I1, I2, I5
T200	I2, I4
T300	I2, I3
T400	I1, I2, I4
T500	I1, I3
T600	I2, I3
T700	I1, I3
T800	I1, I2, I3, I5
T900	I1, I2, I3



รูปที่ 2.11 ตัวอย่างการทำงานของอัลกอริทึมเอไพรออริ







$C_4 = \varnothing \rightarrow F_4 = \varnothing \rightarrow$ อัลกอริทึมหยุดการทำงาน	ไอเท็มเซตที่ปรากฏบ่อย คือ {I1}, {I2}, {I3}, {I4}, {I5}, {I1, I2}, {I1, I3}, {I1, I5}, {I2, I3}, {I2, I4}, {I2, I5}, {I1, I2, I3}, {I1, I2, I5}

รูปที่ 2.11 ตัวอย่างการทำงานของอัลกอริทึมเอไพรออริ (ต่อ)

กฎความสัมพันธ์ที่ค้นพบ	I1 → I2	I1 → I2, I3
	I1 → I3	I2 → I1, I3
	I1 → I5	I3 → I1, I2
	I2 → I1	I1, I2 → I3
	I3 → I1	I1, I3 → I2
	I5 → I1	I2, I3 → I1
	I2 → I3	I1 → I2, I5
	I2 → I4	I2 → I1, I5
	I2 → I5	I5 → I1, I2
	I3 → I2	I1, I2 → I5
	I4 → I2	I1, I5 → I2
	I5 → I2	I2, I5 → I1

รูปที่ 2.11 ตัวอย่างการทำงานของอัลกอริทึมเอไพรออริ (ต่อ)

2.8 การสุ่มตัวอย่าง (Sampling)

Mannila, Toivonen และ Verkamo (1994) ได้เสนอแนวความคิดในการนำเทคนิคการสุ่มข้อมูลมาประยุกต์ใช้ในกระบวนการค้นหากฎความสัมพันธ์เพื่อช่วยลดขนาดของข้อมูล และระยะเวลาที่ใช้ในการค้นหากฎความสัมพันธ์ โดยนำเทคนิคการสุ่มตัวอย่างมาประยุกต์ใช้ในส่วนของการค้นหาแคนดิเดทไอเท็มเซตทั้งหมดที่มีความครอบคลุม โดยใช้ Chernoff bounds (Alon and H. Spencer, 1992, Hagerup and Rub, 1989/90) ในการวิเคราะห์ว่าขนาดของชุดข้อมูลสุ่มที่เหมาะสมควรเป็นเท่าไร ซึ่งวิธีการนี้สามารถช่วยลดระยะเวลาในการค้นหาความสัมพันธ์ของข้อมูลได้อย่างมีประสิทธิภาพ วิธีการค้นหาแคนดิเดทไอเท็มเซตจะมีความแตกต่างไปจากอัลกอริทึม เอไอเอส คือ จะใช้ข้อมูลที่เหมาะสมจากรอบการอ่านฐานข้อมูลที่ผ่านมา ในการตัดแคนดิเดทไอเท็มเซตระหว่างรอบการอ่านฐานข้อมูล โดยให้ เซต L_s เป็นเซตของเซตทั้งหมดที่ค้นพบจากข้อมูลขนาด s เซต C_{s+1} จะมีแคนดิเดทไอเท็มเซตของ L_{s+1} อยู่ด้วย ดังนั้นเซตของข้อมูลขนาด s+1 มีความเป็นไปได้ที่จะมีปรากฏอยู่ในเซต L_{s+1} ซึ่งครอบคลุมเซตของ L_s ด้วย เช่น ถ้าทราบว่าในเซต L_2 = {AB, BC, AC, AE, BE, AF, CG} สามารถที่จะสรุปได้ว่า ABC และ ABE เท่านั้นที่มีความเป็นไปได้ที่จะเป็นสมาชิกของ L_3

Toivonen (1996) ได้เสนอแนวความคิดในการนำเทคนิคการลดการอ่านข้อมูลจากฐานข้อมูล โดยการนำเทคนิคการสุ่มตัวอย่างมาประยุกต์ใช้ในกระบวนการค้นหากฎความสัมพันธ์ โดยวิธีการที่นำเสนอการค้นหากฎความสัมพันธ์ และแสดงกฎความสัมพันธ์ออกมาเป็นบางส่วนด้วยการอ่านข้อมูลจากฐานข้อมูลเพียงรอบเดียว แต่กฎความสัมพันธ์ที่หายไปบางส่วนสามารถค้นพบได้ในรอบที่สอง เนื่องจากการค้นหาแคนดิเดทไอเท็มเซตและการค้นหากฎความสัมพันธ์

ทั้งหมดทำให้ฐานข้อมูลทำงานหนักเนื่องจากต้องมีการอ่านข้อมูลจากฐานข้อมูลหลายรอบ ในปัจจุบันงานวิจัยจึงมีการลดการทำงานกับดิสก์ (Disk) และการอ่านข้อมูลจากฐานข้อมูลหลายๆ รอบ (Savasere, Omiecinski and Navathe, 1995) จึงได้มีการนำเทคนิคการสุ่มตัวอย่างมาประยุกต์ใช้ในการค้นหาไอเท็มเซตที่ปรากฏบ่อยด้วยการอ่านข้อมูลจากฐานข้อมูลเพียงรอบเดียว

การสุ่มตัวอย่าง หมายถึง การเลือกตัวอย่างขึ้นมาเพื่อใช้เป็นตัวแทนในการศึกษาโดยสมาชิกของกลุ่มตัวอย่างที่จะถูกเลือกขึ้นมานั้นจะต้องมีโอกาสที่จะถูกเลือกเท่าๆ กันหรือปราศจากความลำเอียง (Unbias) กลุ่มตัวอย่างที่เป็นตัวแทนที่ดีจะต้องมีค่าสถิติ (Statistic) ที่ได้จากการคำนวณจากกลุ่มตัวอย่างใกล้เคียงหรือเกือบเท่ากับค่าพารามิเตอร์ (Parameter) ของประชากร การสุ่มตัวอย่างเป็นกระบวนการที่เป็นระบบในการสุ่มหน่วยตัวอย่างมาจากประชากรที่สนใจศึกษา หลักการสุ่มตัวอย่างมีดังนี้ (สุภาพร ศรีสัตตรัตน์)

1) การสุ่มตัวอย่างแบบใส่คืนที่ (Sampling with replacement) คือ เมื่อมีการสุ่มตัวอย่างขึ้นมาหนึ่งหน่วย แล้วทำการใส่หน่วยที่ถูกสุ่มขึ้นมากลับลงไปคืนก่อนที่จะทำการสุ่มเพื่อเลือกสมาชิกหน่วยต่อไป ดังนั้นจะทำให้ทุกหน่วยสมาชิกจะมีโอกาสในการถูกสุ่มขึ้นมาในแต่ละครั้งเท่ากัน แต่มีโอกาสที่สมาชิกแต่ละหน่วยอาจจะถูกสุ่มขึ้นมาซ้ำได้

2) การสุ่มตัวอย่างแบบไม่ใส่คืนที่ (Sampling without replacement) คือ เมื่อมีการสุ่มตัวอย่างขึ้นมาหนึ่งหน่วยแล้วจะไม่ใส่หน่วยที่ถูกสุ่มขึ้นมาได้กลับลงไปคืน โดยการสุ่มตัวอย่างลักษณะนี้จะทำให้แต่ละหน่วยสมาชิกจะมีโอกาสในการถูกสุ่มขึ้นมาในแต่ละครั้งไม่เท่ากัน แต่จะไม่เกิดการสุ่มหน่วยที่ซ้ำกัน

การสุ่มตัวอย่างจากประชากรสามารถแบ่งได้เป็น 2 ประเภทใหญ่ๆ ได้ดังนี้

2.8.1 การสุ่มตัวอย่างโดยไม่ใช้หลักทฤษฎีค่าความน่าจะเป็น (Non-probability sampling)

เป็นการสุ่มตัวอย่างแบบไม่คำนึงถึงค่าความน่าจะเป็นหรือโอกาสที่จะถูกเลือกของกลุ่มตัวอย่าง โดยการสุ่มตัวอย่างลักษณะนี้จะขึ้นอยู่กับความควบคุมของผู้วิจัย การสุ่มตัวอย่างแบบนี้สมาชิกทุกหน่วยจากกลุ่มประชากรนั้นอาจมีโอกาสในการได้รับการคัดเลือกมาเป็นสมาชิกในกลุ่มตัวอย่างไม่เท่ากัน อาจทำให้เกิดความลำเอียงในการสุ่มตัวอย่างได้ง่ายกว่าการสุ่มตัวอย่างที่อาศัยความน่าจะเป็นในการสุ่ม โดยการสุ่มตัวอย่างแบบนี้มีหลายวิธีดังต่อไปนี้

1) การสุ่มตัวอย่างแบบบังเอิญ (Accidental sampling) การสุ่มตัวอย่างโดยวิธีนี้จะทำการสุ่มตัวอย่างจากสมาชิกของกลุ่มประชากรเป้าหมายเท่าที่จะหาได้ เช่น ประชาชนในเขตกรุงเทพมหานครที่ดื่มเครื่องดื่มแอลกอฮอล์เป็นประจำ โดยการสุ่มตัวอย่างแบบบังเอิญนี้มีข้อบกพร่องคือ จะได้ตัวอย่างของประชากรเป้าหมายที่อาจจะไม่สามารถเป็นตัวแทนของประชากรทั้งหมดได้ เนื่องจากตัวอย่างที่สุ่มมาได้อาจมีคุณสมบัติต่างจากประชากรเป้าหมาย

2) การสุ่มตัวอย่างแบบโควต้า (Quota sampling) การสุ่มตัวอย่างโดยวิธีนี้จะทำการจำแนกประชากรออกเป็นส่วนๆ ตามระดับของตัวแปรที่สนใจ โดยสมาชิกที่อยู่แต่ละระดับจะมีลักษณะที่คล้ายคลึงกัน แล้วเลือกสมาชิกแต่ละระดับตามโควต้าที่กำหนดโดยไม่มีการสุ่ม เช่น ต้องการกลุ่มประชากรในเขตกรุงเทพมหานครที่ดื่มเครื่องดื่มแอลกอฮอล์เป็นประจำ และให้โควต้าโดยการกำหนดจำนวนกลุ่มตัวอย่างโดยกำหนดโควต้าเป็น อายุตั้งแต่ 25-40 ปี จำนวน 200 คน และอายุ 40 ปีขึ้นไปจำนวน 200 คน

3) การสุ่มตัวอย่างแบบเจาะจง (Purposive sampling) การสุ่มตัวอย่างโดยวิธีนี้จะทำการกำหนดสมาชิกของกลุ่มประชากรที่จะมาเป็นสมาชิกในกลุ่มตัวอย่าง โดยใช้ดุลยพินิจของผู้วิจัย เช่น ต้องการศึกษาแนวความคิดของประชาชนในเขตกรุงเทพมหานครที่ดื่มเครื่องดื่มแอลกอฮอล์เป็นประจำ การสุ่มตัวอย่างแบบเจาะจงมีหลักในการเลือกตัวอย่างคือ จะเลือกตัวอย่างในกรณีที่คิดว่าสามารถเป็นตัวแทนของประชากรเป้าหมายได้

4) การสุ่มตัวอย่างแบบลูกโซ่ (Snowball sampling) การสุ่มตัวอย่างวิธีนี้จะเลือกกลุ่มตัวอย่างโดยอาศัยการแนะนำของตัวอย่างที่ได้เก็บข้อมูลไปแล้ว เช่น นาย ก. มีคุณสมบัติตรงกับกลุ่มตัวอย่างที่นักวิจัยต้องการศึกษา จากนั้นให้ นาย ก. แนะนำคนอื่นๆ ที่มีคุณสมบัติและลักษณะตรงกับที่ต้องการและให้คนอื่นๆ แนะนำเช่นเดียวกับ นาย ก. จนกระทั่งได้กลุ่มตัวอย่างครบตามที่ต้องการ

2.8.2 การสุ่มตัวอย่างโดยใช้หลักทฤษฎีค่าความน่าจะเป็น (Probability sampling)

เป็นการสุ่มตัวอย่างจากประชากร โดยมีเงื่อนไขดังต่อไปนี้

1) รู้จำนวนประชากรทั้งหมด

2) ประชากรทั้งหมดมีโอกาสที่จะถูกสุ่มมาเป็นกลุ่มตัวอย่างเท่าเทียมกัน

3) ใช้วิธีการสุ่มที่เหมาะสม เพื่อให้หน่วยตัวอย่างมีโอกาสถูกสุ่มเท่าเทียมกัน

4) ใช้วิธีประมาณค่าพารามิเตอร์ที่เหมาะสม

การสุ่มแบบอาศัยความน่าจะเป็น เป็นวิธีที่นิยมใช้กันมาก เพราะมีความน่าเชื่อถือ และกลุ่มตัวอย่างที่ถูกเลือกขึ้นมานี้มีโอกาสที่สมาชิกที่ถูกเลือกการสุ่มแบบนี้มีหลายวิธี ดังต่อไปนี้

1) การสุ่มตัวอย่างแบบง่าย (Simple random sampling) การสุ่มตัวอย่าง โดยวิธีนี้เป็นวิธีการเลือกสมาชิก n หน่วยออกจากสมาชิกทั้งหมด N หน่วย โดยวิธีนี้สมาชิกของกลุ่มประชากรทุกๆ หน่วยมีโอกาสเท่าๆ กัน และเป็นอิสระต่อกันในการที่จะได้รับเลือกมาเป็นสมาชิกของกลุ่มตัวอย่าง การเป็นอิสระต่อกัน หมายความว่า การเลือกสมาชิกแต่ละหน่วยนั้นจะไม่มีผลกระทบต่อการเลือกสมาชิกหน่วยอื่นๆ แต่ละหน่วยของประชากรทั้งหมดจะเป็น 1, 2, …, N โดยชุดของข้อมูลสุ่มที่ถูกเลือกขึ้นมาจะอยู่ระหว่าง 1 ถึง N วิธีการสุ่มตัวอย่างอย่างง่ายอาจทำได้ดังนี้

(1) การจับสลาก วิธีนี้เหมาะสมและใช้ได้ดีสำหรับประชากรที่มีขนาดใหญ่ไม่มากนัก โดยการเขียนตัวเลขแทนหน่วยสมาชิกหรือเลขประจำตัวสมาชิกทั้งหมดของประชากรลงในแผ่นกระดาษ แล้วนำแผ่นกระดาษที่เขียนแล้วทั้งหมดใส่ลงในกล่องเขย่าให้ทั่วกัน แล้วใช้วิธีการจับฉลากตัวอย่างขึ้นมาจำนวนหนึ่งตามที่ต้องการ โดยการจับฉลากแต่ละครั้งให้หยิบขึ้นมาครั้งละหนึ่งอันเท่านั้น หลังจากนั้นจึงหยิบอันที่สอง และก่อนที่จะทำการจับฉลากแต่ละครั้งจะต้องมีการเขย่ากล่องให้ทั่วกันทุกครั้ง ทำเช่นนี้ไปเรื่อยๆ จนกว่าจะได้จำนวนสมาชิกของกลุ่มตัวอย่างตามที่ต้องการ เช่น การสำรวจความคิดเห็นในการเลือกซื้อน้ำมันพืชของแม่บ้านที่มาซื้อสินค้าที่ห้างสรรพสินค้าแห่งหนึ่งจำนวน 300 คน ก็ใช้การจับฉลากแม่บ้านขึ้นมาจำนวนหนึ่งอาจเป็น 100 คนเพื่อใช้สำหรับศึกษาเป็นต้น

(2) ตารางเลขสุ่ม (Table of random number) วิธีนี้เหมาะสำหรับประชากรที่มีขนาดใหญ่ โดยวิธีการสุ่มตัวอย่างแบบนี้มีวิธีการดังนี้

- ผู้วิจัยต้องกำหนดตัวเลขแทนหน่วยของประชากร หรือเลขประจำตัวสมาชิกของประชากรทั้งหมดเสียก่อน เช่น มีประชากรทั้งหมด 100 คน ก็จะกำหนดเป็น 01, 02, 03, …, 100 เป็นต้น
- กำหนดจุดเริ่มต้นในการอ่านตัวเลขในตารางเลขสุ่มว่าจะเริ่มจากที่ใด เพราะในตารางเลขสุ่มจะมีข้อมูลเลขประจำตัวของสมาชิกทั้งหมดที่ถูกจัดเรียงเป็นแถว (Row) และคอลัมน์ (Column) เมื่อลากเส้นจากแถวไปตัดกับเส้นที่ลากมาจากคอลัมน์ ณ จุดที่ตัดกันนั้นจะมีตัวเลขอยู่เสมอ เช่น แถวที่ 8 คอลัมน์ที่ 9 เมื่อลากเส้นมาตัดกันก็จะได้จุดเริ่มต้นในการอ่านตัวเลข สมมติว่าที่จุดที่เส้นทั้งสองตัดกันนั้นมีเลขประจำตัวสมาชิกคือ 72 ก็จะเป็นสมาชิกตัวแรกที่สุ่มได้
- การกำหนดทิศทางในการอ่านตาราง เช่น อ่านตัวเลขจากซ้ายไปขวา จากบนไปล่าง หรือตามแนวทแยง เป็นต้น หลังจากที่เราได้จุดเริ่มต้นแล้วเราก็จะทำการอ่านข้อมูลจากทิศทางที่เรากำหนดไว้ จะทำให้เราทราบได้ว่าตัวเลขต่อไปจะเป็นหมายเลขใดที่ปรากฏอยู่ในตารางนั้น ทำเช่นนี้ต่อไปเรื่อยๆ จนได้ครบจำนวนตามที่ต้องการ จุดเด่นของการสุ่มแบบนี้ก็คือมีความสะดวกและใช้ได้ง่าย แต่มีข้อเสียคือ ถ้ากลุ่มตัวอย่างที่ต้องการมีจำนวนมาก การใช้วิธีนี้ก็จะเสียเวลามาก

ตารางที่ 2.2 ตัวอย่างตารางเลขสุ่ม

01172	22345	22216	03276	06228	56545
91233	97915	23398	10923	93412	98767
08640	01626	41114	25128	60234	65908
05939	02233	08067	45455	01156	23787
66627	10659	87980	89903	90987	19890
05196	00457	03690	03770	50009	04666
06304	78632	09800	51037	02435	14567
01172	22345	22216	03276	06228	56545

2) การสุ่มตัวอย่างแบบมีระบบ (Systematic sampling) การสุ่มตัวอย่างโดยวิธีนี้เป็นการสุ่มกลุ่มตัวอย่างที่อาศัยช่วงระยะห่างเป็นหลักในการสุ่ม เหมาะสำหรับใช้ในกรณีที่หน่วยตัวอย่างของกลุ่มประชากรจัดเรียงไว้อย่างเป็นระบบอยู่แล้ว เช่น การสุ่มได้กลุ่มตัวอย่างที่ประกอบไปด้วยสมาชิกหน่วยที่ 3, 9, 15, 21, 27, 33, 39, 25, … หรือการสุ่มได้กลุ่มตัวอย่างที่ประกอบไปด้วยสมาชิกหน่วยที่ 10, 25, 40, 55, 70, 85, … ไปเรื่อยๆ จนครบตามจำนวนตัวอย่างที่ต้องการ จากตัวอย่างกลุ่มแรกจะมีช่วงระยะห่างเท่ากับ 6 คือ ตัวอย่างที่จะได้รับการคัดเลือกได้แก่สมาชิกที่อยู่ในตำแหน่งทุกช่วงระยะห่างเท่ากับ 6 ส่วนกลุ่มที่สองจะมีช่วงระยะห่างเท่ากับ 15 การสุ่มตัวอย่างแบบนี้จะต้องพิจารณาดังนี้

(1) สมาชิกของประชากรทั้งหมดจะต้องถูกเรียงไว้ตามลำดับอย่างถูกต้องตั้งแต่หน่วยแรกจนถึงหน่วยสุดท้ายก่อนที่จะดำเนินการสุ่มตัวอย่าง เช่น รายชื่อของนักเรียนถูกจัดเรียงตามเลขประจำตัว

(2) ช่วงระยะห่าง (Sampling interval, i) ที่จะเป็นตัวกำหนดสมาชิกหาได้จากการนำจำนวนสมาชิกทั้งหมดของประชากร (N) หารด้วยจำนวนสมาชิกของกลุ่มตัวอย่างที่ต้องการ (n)

$$\text{ระยะห่าง (i)} = \frac{n}{N}$$

(3) จุดเริ่มต้น คือ สมาชิกที่เป็นหน่วยเริ่มต้น เราจะต้องหาว่าสมาชิกที่เป็นหน่วยเริ่มต้นนั้นอยู่ ณ ตำแหน่งใด หลังจากที่หาได้แล้วจะทำให้เราทราบได้ว่าจำนวนสมาชิกหน่วยอื่นๆ ที่เหลือทั้งหมดว่าเป็นตัวใดโดยการนำช่วงระยะห่าง (i) ที่หาได้ในข้อ 2 มาบวกเข้าไป โดยการที่จะหาสมาชิกหน่วยเริ่มต้นสามารถทำได้ดังนี้

- การจับฉลากหรือ ใช้ตารางเลขสุ่ม (Table of random number) ทำการสุ่มสมาชิกที่อยู่ในช่วงระยะห่างแรกๆ โดยนำเอาสมาชิกตามจำนวนช่วงระยะห่างมาทำการจับฉลาก เช่น ช่วงระยะห่างเท่ากับ 12 ก็ให้นำเอาสมาชิก 12 หน่วยแรกมาทำการจับฉลาก เมื่อได้สมาชิกหน่วยใดก็จะนับสมาชิกหน่วยนั้นเป็นหน่วยเริ่มต้น สมาชิกหน่วยต่อๆ ไปก็จะอยู่ในตำแหน่งที่เป็นช่วงระยะห่างที่หาได้
- หาจุดเริ่มต้นหรือสมาชิกหน่วยเริ่มต้น โดยการนำสมาชิกทั้งหมดมาจับฉลาก หรืออาจจะกำหนดหมายเลขให้กับสมาชิกแต่ละหน่วย แล้วใช้ตารางเลขสุ่ม สุ่มได้สมาชิกหน่วยใดหน่วยนั้นก็เป็นจุดเริ่มต้น ในกรณีที่สุ่มตัวอย่างจากจุดเริ่มต้นไปจนครบสมาชิกตัวสุดท้ายแล้วแต่ยังได้ตัวอย่างไม่ครบตามจำนวนจะต้องย้อนกลับไปนับที่สมาชิกตัวแรกต่อไปจนครบตามจำนวนที่ต้องการ

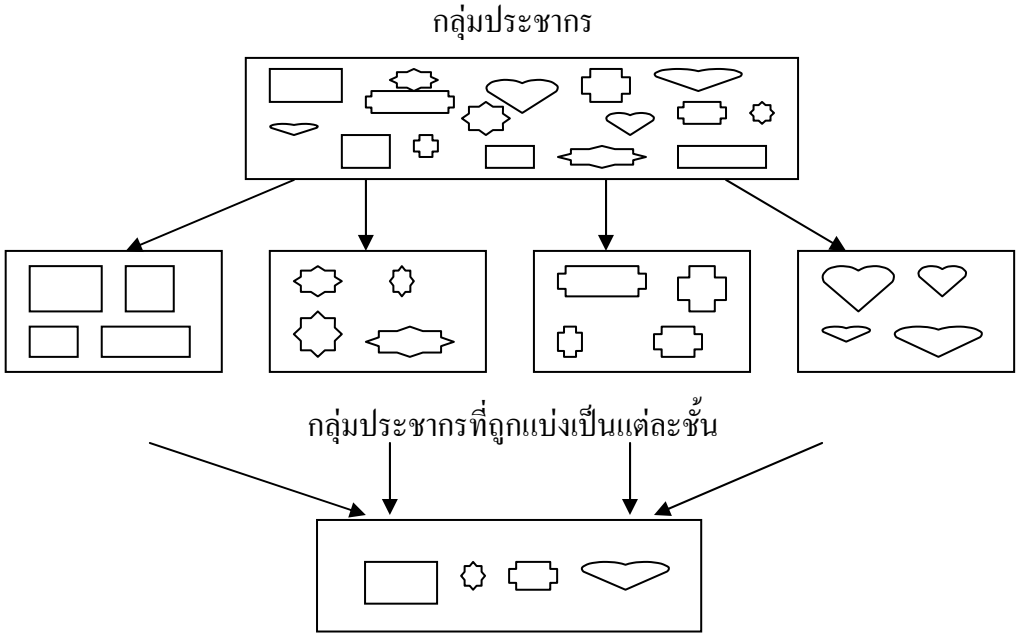
(4) การสุ่มตัวอย่างแบบมีระบบไม่เหมาะสมกับประชากรที่มีการจัดเรียงตามลำดับของค่าหรือปริมาณหรือคุณลักษณะ เนื่องจากอาจทำให้กลุ่มตัวอย่างที่ถูกสุ่มขึ้นมาไม่เป็นตัวแทนที่ดีของประชากร แต่ทั้งนี้จะขึ้นอยู่กับสมาชิกที่เป็นหน่วยเริ่มต้นของประชากร แต่ในกรณีที่ประชากรดังกล่าวมีการจัดเรียงกันอย่างสุ่ม (Random) การใช้วิธีการสุ่มตัวอย่างแบบมีระบบจะทำให้กลุ่มตัวอย่างที่ถูกสุ่มขึ้นมาเป็นตัวแทนที่ดีของประชากร

3) การสุ่มตัวอย่างแบบแบ่งชั้น (Stratified random sampling) การสุ่มตัวอย่างโดยวิธีนี้เหมาะสำหรับใช้ในการศึกษาประชากรที่มีความแตกต่างกันมาก ถ้าทำการศึกษาด้วยการสุ่มตัวอย่างแบบง่ายจะทำให้ได้ข้อมูลที่คลาดเคลื่อนมาก ดังนั้นถ้าประชากรที่มีความแตกต่างกันมากๆ สามารถแบ่งออกเป็นกลุ่มย่อยได้หลายกลุ่มจึงควรใช้วิธีการสุ่มตัวอย่างแบบแบ่งชั้นจะสามารถทำให้สุ่มตัวอย่างที่เป็นตัวแทนของประชากรที่ดีกว่า โดยการสุ่มตัวอย่างแบบแบ่งชั้นมีหลักการดังนี้

(1) แบ่งประชากรออกเป็นกลุ่มย่อยๆ หรือเป็นชั้น (Strata) โดยให้สมาชิกในแต่ละชั้นมีลักษณะคล้ายกันมากๆ และแต่ละชั้นจะต้องมีความแตกต่างกันมากๆ

(2) สมาชิกแต่ละหน่วยจะต้องอยู่ภายในชั้นใดชั้นหนึ่งเท่านั้น จะอยู่คาบเกี่ยวในแต่ละชั้นไม่ได้

(3) การสุ่มกลุ่มตัวอย่างแต่ละชั้นจะต้องเป็นอิสระจากกัน (Independent samples) คือ การสุ่มในชั้นหนึ่งจะไม่มีผลกระทบกับชั้นอื่นๆ ดังแสดงในรูปที่ 2.12



รูปที่ 2.12 แสดงลักษณะการจัดชั้น ในการสุ่มตัวอย่างแบบแบ่งชั้น

หลังจากที่ทำการแบ่งชั้นแล้ว ทำการสุ่มกลุ่มตัวอย่างจากทุกชั้นที่ทำการแบ่งด้วยวิธีการสุ่มตัวอย่างแบบง่าย (Simple random sampling) โดยอาจใช้วิธีการจับฉลากหรือใช้ตารางเลขสุ่ม กลุ่มตัวอย่างที่ได้มาจะประกอบด้วยสมาชิกจากทุกชั้นทำให้กลุ่มตัวอย่างที่ได้มาสามารถเป็นตัวแทนที่ดีกว่ากลุ่มตัวอย่างที่ได้จากการสุ่มตัวอย่างแบบง่าย

4) การสุ่มตัวอย่างแบบแบ่งกลุ่ม (Cluster sampling หรือ Area sampling) การสุ่มตัวอย่างโดยวิธีนี้เหมาะสำหรับใช้ในการศึกษาประชากรที่มีขนาดใหญ่มาก โดยการแบ่งกลุ่มตัวอย่างออกเป็นกลุ่มๆ โดยสมาชิกภายในกลุ่มแต่ละกลุ่มมีลักษณะหรือคุณสมบัติที่แตกต่างกันมากที่สุดโดยภายในแต่ละกลุ่มควรมีทุกๆ ลักษณะหรือคุณสมบัติที่ต้องการ แต่ระหว่างกลุ่มให้ลักษณะหรือคุณสมบัติที่เหมือนกันหรือคล้ายกันมากที่สุด หลังจากที่ทำการแบ่งกลุ่มแล้วจึงทำการสุ่มตัวอย่างโดยใช้วิธีการสุ่มตัวอย่างแบบง่าย (Simple random sampling) มีหน่วยสมาชิกเป็นกลุ่มของประชากร ในการแบ่งกลุ่มประชากรนั้นควรให้สมาชิกภายในกลุ่มมีจำนวนน้อย เพื่อที่จะทำให้จำนวนกลุ่มมีจำนวนมาก และเมื่อสุ่มขึ้นมาแล้วเพื่อจะได้เป็นตัวแทนที่ดีของประชากร หลังจากที่ทำการสุ่มได้กลุ่มใดๆ แล้วให้ทำการบันทึกข้อมูลของสมาชิกทุกหน่วยภายในกลุ่มที่สุ่มได้ วิธีการสุ่มตัวอย่างแบบนี้จะช่วยลดค่าใช้จ่ายได้มากกว่าการสุ่มตัวอย่างแบบง่าย แต่มีความคลาดเคลื่อนในการประเมินค่าพารามิเตอร์สูงกว่าการสุ่มตัวอย่างแบบง่าย

5) การสุ่มตัวอย่างแบบแบ่งกลุ่มชนิดสองขั้นตอน (Two-Stage Cluster sampling) การสุ่มตัวอย่างโดยวิธีนี้จะมีหลักการเดียวกันกับการสุ่มตัวอย่างแบบแบ่งกลุ่ม แต่มีความแตกต่างกันที่เมื่อทำการสุ่มตัวอย่างได้กลุ่มใดมาแล้ว จะทำการสุ่มหน่วยสมาชิกภายในแต่ละกลุ่มอีกครั้งหนึ่งโดยใช้วิธีการสุ่มตัวอย่างแบบง่าย (Simple random sampling) จึงมีสมาชิกบางส่วนของกลุ่มที่สุ่มได้มาเท่านั้นที่จะถูกเก็บข้อมูลไว้

6) การสุ่มตัวอย่างแบบหลายขั้น (Multi-Stage Cluster sampling) การสุ่มตัวอย่างโดยวิธีนี้จะประกอบด้วยหลายขั้นตอนโดยแต่ละขั้นตอนอาจจะใช้เทคนิคการสุ่มตัวอย่างที่เหมือนหรือแตกต่างกันก็ได้ขึ้นอยู่กับความเหมาะสม เช่น ต้องการสุ่มนักเรียนชั้นประถมศึกษาปีที่ 6 ที่ดื่มนมเป็นประจำจากทั่วประเทศ โดยใช้วิธีการสุ่มตัวอย่างแบบหลายขั้นตอนดังนี้

ขั้นตอนที่ 1 ใช้เทคนิคการสุ่มตัวอย่างแบบแบ่งชั้น (Stratified random sampling) สุ่มจังหวัดในประเทศไทยโดยแบ่งออกตามภาคภูมิศาสตร์ ได้แก่ ภาคกลาง ภาคเหนือ ภาคใต้ ภาคตะวันออกเฉียงเหนือ และภาคตะวันตก

ขั้นตอนที่ 2 ใช้เทคนิคการสุ่มตัวอย่างแบบง่าย (Simple random sampling) หลังจากที่ได้จังหวัดเป็นตัวแทนของทุกภาคในประเทศไทยแล้วทำการสุ่มอำเภอโดยให้ทุกอำเภอในจังหวัดตัวอย่างมีโอกาสถูกเลือกโดยเท่าเทียมกัน

ขั้นตอนที่ 3 ใช้เทคนิคในการแบ่งชั้น (Stratified random sampling) แบ่งโรงเรียนในแต่ละอำเภอเป็น 3 ชั้นตามขนาดของโรงเรียน คือ ใหญ่ กลาง เล็ก

ขั้นตอนที่ 4 ใช้เทคนิคการสุ่มตัวอย่างแบบง่าย (Simple random sampling) ในการสุ่มนักเรียนชั้นประถมศึกษาปีที่ 6

การทำเหมืองข้อมูลประเภท “การค้นหากฎความสัมพันธ์” (Association rule mining) เป็นเทคนิคการทำเหมืองข้อมูลที่นิยมใช้ในการทำเหมืองข้อมูลกับฐานข้อมูลที่มีขนาดใหญ่ โดย เฉพาะฐานข้อมูลทางด้านธุรกิจ เนื่องจากการค้นหากฎความสัมพันธ์จะทำให้เราสามารถทราบถึงพฤติกรรมของผู้บริโภคได้ โดยการนำเอาความรู้ที่แสดงอยู่ในรูปกฎความสัมพันธ์ที่ได้จากการทำเหมืองข้อมูลมาใช้ในกระบวนการตัดสินใจในการบริหารจัดการเกี่ยวกับสินค้าและการส่งเสริมการขายได้อย่างมีประสิทธิภาพ แต่เนื่องจากในปัจจุบันมีการเล็งเห็นว่าการเก็บประวัติรายการซื้อสินค้าของลูกค้ามีความสำคัญมากยิ่งขึ้น เพราะสามารถนำข้อมูลเหล่านี้กลับมาใช้ในการวิเคราะห์พฤติกรรมการซื้อของลูกค้า และช่วยให้สามารถจัดรายการส่งเสริมการขายได้อย่างมีประสิทธิภาพ จากสาเหตุนี้ทำให้ฐานข้อมูลมีขนาดใหญ่มากยิ่งขึ้น และส่งผลทำให้กระบวนการทำเหมืองข้อมูลใช้

ระยะเวลาในการทำงานนานมากยิ่งขึ้น งานวิจัยนี้มุ่งเน้นที่จะพัฒนาอัลกอริทึม เอไพรออริ (APRIORI) ด้วย SQL โดยนำเทคนิคการสุ่มข้อมูลมาประยุกต์ใช้เพื่อช่วยให้อัลกอริทึมในการขุดค้นความรู้สามารถทำการค้นหากฎความสัมพันธ์ที่แข็งแกร่งได้อย่างรวดเร็วและมีความถูกต้อง โดยกฎความสัมพันธ์ที่ค้นพบจะต้องมีค่าสนับสนุนไม่ต่ำกว่า 80 เปอร์เซ็นต์ และมีค่าความเชื่อมั่นไม่ต่ำกว่า 95 เปอร์เซ็นต์

บทที่ 3
ระเบียบวิธีวิจัย

งานวิจัยนี้มีจุดมุ่งหมายที่จะพัฒนาอัลกอริทึมเอไพรออริ (APRIORI) ที่ใช้ในการทำเหมืองข้อมูลประเภทการค้นหากฎความสัมพันธ์โดยโปรแกรมที่พัฒนาขึ้นจะใช้ภาษา SQL และมุ่งเน้นให้อัลกอริทึมสามารถทำการค้นหากฎความสัมพันธ์ที่แข็งแกร่งกับชุดข้อมูลเชิงธุรกิจ หรือข้อมูลที่มีการเกิดขึ้นซ้ำกันบ่อยๆ ที่มีขนาดใหญ่ได้ในระยะเวลาอันรวดเร็วด้วยการนำเทคนิคการสุ่มตัวอย่างมาประยุกต์ใช้งานร่วมกับอัลกอริทึมที่จะทำการพัฒนาขึ้น กฎความสัมพันธ์ที่ค้นพบจะต้องมีจำนวนข้อมูลสนับสนุนไม่ต่ำกว่า 80 เปอร์เซ็นต์ และมีค่าความเชื่อมั่นไม่ต่ำกว่า 95 เปอร์เซ็นต์ รายละเอียดในเนื้อหาบทนี้ประกอบด้วยระเบียบวิธีวิจัย ปรากฏอยู่ในหัวข้อ 3.1 โปรแกรม SASS ที่ทำการพัฒนาขึ้นมาเพื่อใช้ในกระบวนการค้นหากฎความสัมพันธ์ ปรากฏอยู่ในหัวข้อที่ 3.2 คำอธิบายชุดข้อมูลที่ใช้ในการทดสอบอัลกอริทึม ปรากฏอยู่ในหัวข้อ 3.3 และหัวข้อ 3.4 เป็นรายละเอียดวิธีที่ใช้ในการเปรียบเทียบผลลัพธ์ที่ได้จากการทดสอบกับอัลกอริทึม

3.1 ระเบียบวิธีวิจัย

การค้นคว้าวิจัยจะแบ่งออกเป็น ขั้นตอนดังนี้

3.1.1 ศึกษาและรวบรวมสรุปงานวิจัยที่เกี่ยวข้อง

3.1.2 ศึกษาการใช้งานอัลกอริทึม เอไพรออริ จาก WEKA (Frank, Hall, Holmes, Martin, Mayo, Pfahringer, Smith and Witten, 2005) ซึ่งเป็นโปรแกรมสำเร็จรูปแบบเปิดเผยซอร์สโค้ด (Open-source environment) ที่ใช้ในการทำเหมืองข้อมูล โดยทำการทดสอบการค้นหาความสัมพันธ์ของข้อมูลทั้งเจ็ดชุดที่จะนำมาทดสอบในงานวิจัยนี้ในรูปแบบเท็กซ์ไฟล์ โดยกำหนดค่าสนับสนุนขั้นต่ำที่ 80 เปอร์เซ็นต์ และค่าความเชื่อมั่นขั้นต่ำที่ 95 เปอร์เซ็นต์ แล้วบันทึกผลการค้นหากฎความสัมพันธ์ที่ค้นพบ เพื่อนำกฎความสัมพันธ์ที่ได้มาเปรียบเทียบความถูกต้องกับผลการทดสอบที่ได้จากโปรแกรม SASS ที่ทำการพัฒนาขึ้น

3.1.3 รวบรวมข้อมูลที่จะใช้ในการทดสอบจำนวน 7 ชุดข้อมูล โดยข้อมูล 6 ชุดสืบค้นสืบค้นจากแหล่งข้อมูลของมหาวิทยาลัยแห่งรัฐแคลิฟอร์เนีย เมืองเออร์ไวน์ (University of California at Irvine) (http://www.ics.uci.edu/~mlearn/MLReposi tory.html) และข้อมูลอีก 1 ชุด เป็นข้อมูลที่ทำการสังเคราะห์ขึ้น

3.1.4 ทำการบันทึกข้อมูลที่ได้มาลงในฐานข้อมูลให้อยู่ในรูปแบบรายการทรานแซคชัน

3.1.5 ออกแบบอัลกอริทึมเอไพรออริ ในรูปแบบ SQL และนำเทคนิคการสุ่มตัวอย่างมาประยุกต์ใช้ในการสุ่มข้อมูล โดยทำการทดสอบกับเทคนิคการสุ่มตัวอย่าง 2 แบบคือ

1) การสุ่มตัวอย่างแบบง่าย

2) การสุ่มตัวอย่างแบบมีระบบ

โดยการใช้เทคนิคการสุ่มตัวอย่างทั้งสองแบบนี้จะทดสอบโดยใช้หลักการสุ่มตัวอย่างทั้งสองแบบคือ

1) การสุ่มตัวอย่างแบบใส่คืนที่

2) การสุ่มตัวอย่างแบบไม่ใส่คืนที่

3.1.6 พัฒนาอัลกอริทึมตามที่ได้ออกแบบไว้

3.1.7 ทำการทดสอบอัลกอริทึมกับชุดข้อมูลที่รวบรวมไว้ โดยการทดสอบจะทดสอบกับข้อมูลทั้งหมดและทดสอบกับข้อมูลที่ถูกสุ่มขึ้นมาให้มีขนาดต่างๆ กันดังนี้ 1, 10, 20, 30, 40 และ 50 เปอร์เซ็นต์ โดยเครื่องคอมพิวเตอร์ที่ใช้ในการทดสอบเป็นคอมพิวเตอร์ Laptop CPU Pentium IV ความเร็ว 1.4 GHz หน่วยความจำหลัก 768 MB ฮาร์ดดิสก์ความจุ 40 GB

3.1.8 บันทึกผลและเสนอแนะขนาดของตัวอย่างที่เหมาะสมในการค้นหากฎความสัมพันธ์ที่แข็งแกร่งจากข้อมูลเชิงธุรกิจเพื่อให้ได้กฎความสัมพันธ์ที่มีค่าสนับสนุนและค่าความเชื่อมั่นสูงในระยะเวลาอันรวดเร็ว

3.2 โปรแกรม SASS ที่พัฒนาขึ้นเพื่อใช้ในกระบวนการค้นหากฎความสัมพันธ์

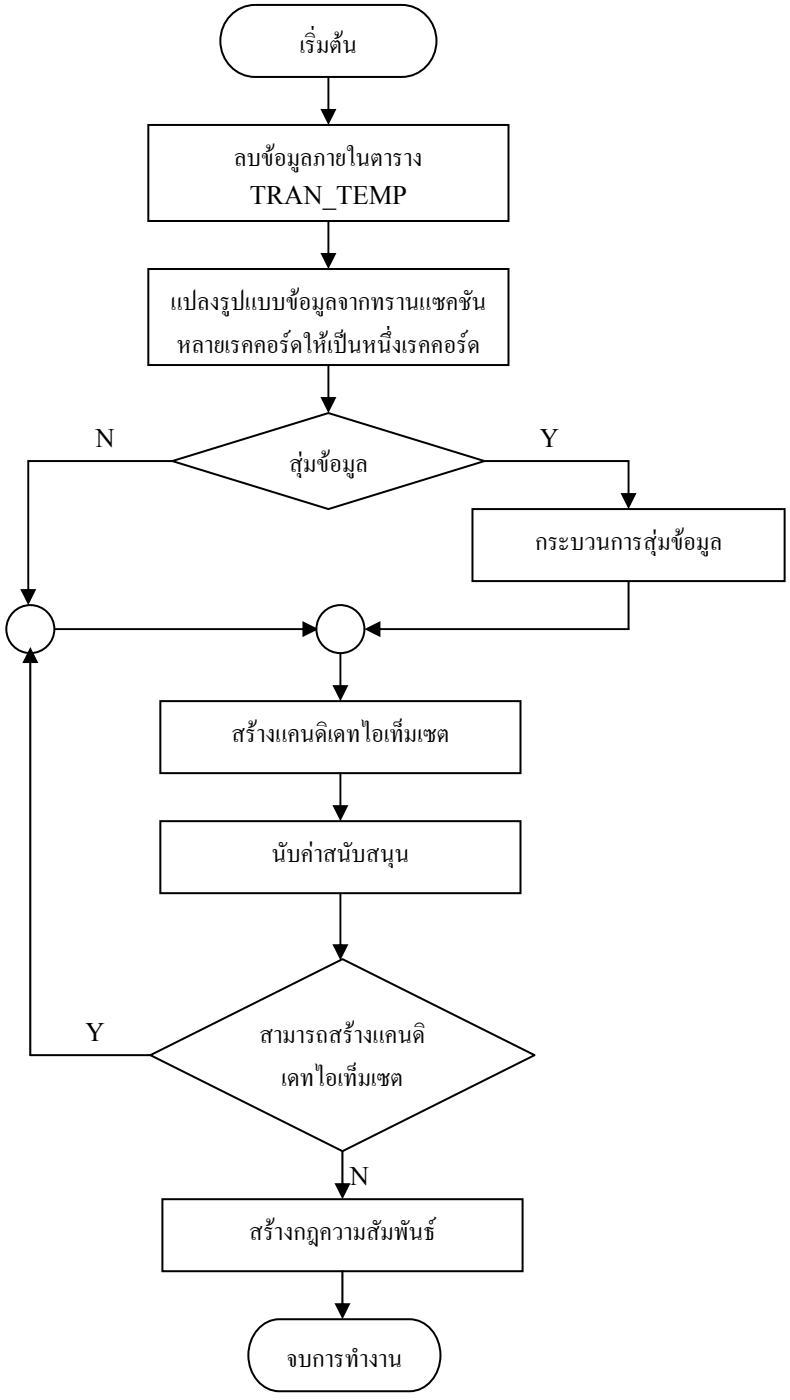
เนื้อหาในส่วนนี้กล่าวถึงการออกแบบโปรแกรม SASS ที่พัฒนามาจากอัลกอริทึมเอไพรออริในรูปแบบ SQL การนำเทคนิคการสุ่มตัวอย่างมาใช้ในโปรแกรม อัลกอริทึมการจัดการกับข้อมูลแบบสตรีมมิง รวมถึงการพัฒนาโปรแกรม SASS เพื่อให้สามารถทำงานได้ตามวัตถุประสงค์

3.2.1 อัลกอริทึมเอไพรออริ

อัลกอริทึมเอไพรออริในงานวิจัยนี้พัฒนาขึ้นในรูปแบบของภาษา SQL การเขียนโปรแกรมใช้รูปแบบของของสโตร์โพรซิเยอร์ (Store procedure) และโปรแกรมสำเร็จรูป โดยตัวอัลกอริทึมเองจะอยู่ในรูปของโพรซิเยอร์ (Procedure) ภายในอัลกอริทึมจะประกอบด้วยส่วนที่สำคัญคือ ส่วนที่ใช้ในการสร้างแคนดิเดทไอเท็มเซต ส่วนที่ใช้ในการนับค่าสนับสนุน และส่วนที่ใช้ในการสร้างกฎความสัมพันธ์ โดยอัลกอริทึมและส่วนต่างๆ ที่เกี่ยวข้องที่ถูกเรียกใช้งานโดย

อัลกอริทึมจะอยู่ในรูปแบบของ โพรซิเยอร์ ฟังก์ชัน (Function) และแพ็กเกจ (Package) โดยจะอธิบายรายละเอียดในแต่ละส่วนต่อไป

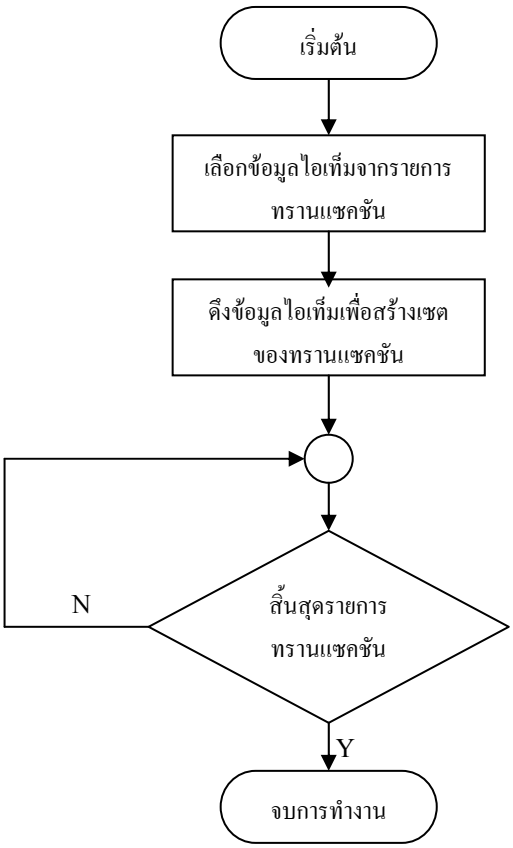
เมื่ออัลกอริทึมเอไพรออริถูกเรียกใช้งานจะทำการลบข้อมูลต่างๆ ที่อาจจะมีค้างอยู่ในตารางชั่วคราว (Temporary table) โดยตารางที่สร้างขึ้นมาเพื่อให้เป็นตารางชั่วคราวคือ TRAN_TEMP จะเป็นตารางที่ใช้ในการเก็บข้อมูลที่อ่านมาจากข้อมูลทรานแซคชัน จากนั้นอัลกอริทึมจะทำการอ่านข้อมูลจากรายการทรานแซคชันที่ระบุ เนื่องจากรายการทรานแซคชันแต่ละรายการนั้นอาจมีไอเท็มเกิดขึ้นได้หลายไอเท็มแต่ละไอเท็มจะถูกจัดเก็บแยกเป็นแต่ละเรคคอร์ด จึงจะต้องมีการแปลงข้อมูลของรายการทรานแซคชันเดียวกันให้เหลือเพียงเรคคอร์ดเดียว โดยจะต้องผ่านกระบวนการแปลงข้อมูลจากหลายเรคคอร์ดให้เหลือเพียงเรคคอร์ดเดียวเสียก่อน แล้วนำข้อมูลที่ถูกแปลงแล้วไปเก็บไว้ในตารางชั่วคราว จากนั้นในกรณีที่ต้องการสุ่มข้อมูลจะต้องผ่านอัลกอริทึมในการสุ่มข้อมูลเสียก่อน จึงจะเรียกใช้งานส่วนที่ใช้ในการสร้างแคนดิเดทไอเท็มเซต หลังจากนั้นทำการนับค่าสนับสนุน แล้วนำไอเท็มเซตที่มีค่าสนับสนุนไม่ต่ำกว่าค่าสนับสนุนขั้นต่ำไปทำการสร้างแคนดิเดทไอเท็มเซตอีกครั้ง อัลกอริทึมจะทำงานซ้ำเช่นนี้ไปเรื่อยๆ จนกว่าจะไม่สามารถสร้างแคนดิเดทไอเท็มเซตได้อีก จึงจะเรียกใช้งานในส่วนการสร้างกฎความสัมพันธ์เพื่อแสดงกฎความสัมพันธ์ที่เป็นไปได้ที่ค้นพบจากไอเท็มเซตที่ปรากฏขึ้นบ่อย การทำงานของอัลกอริทึมแสดงได้ดังรูปที่ 3.1



รูปที่ 3.1 แสดงผังการทำงานของอัลกอริทึมเอไพรออริในรูปแบบ SQL

3.2.2 ส่วนการแปลงรูปแบบข้อมูลจากทรานแซคชันหลายเรคคอร์ดเป็นหนึ่งเรคคอร์ด

ในส่วนนี้จะเป็นโปรแกรมย่อยที่ใช้ในการแปลงรูปแบบข้อมูลทรานแซคชันที่จัดเก็บข้อมูลแต่ละไอเท็มเป็นแต่ละเรคคอร์ดให้กลายเป็นหนึ่งเรคคอร์ด โปรแกรมย่อยนี้จะอยู่ในรูปแบบฟังก์ชันเริ่มทำงานโดยการดึงข้อมูลจากรายการทรานแซคชันที่เลือกแล้วนำข้อมูลของแต่ละไอเท็มมาต่อกันไปเรื่อยๆ โดยมีตัวคั่นคือเครื่องหมายจุลภาค (,) ผลที่ได้คือแต่ละเรคคอร์ดคือหนึ่งรายการทรานแซคชันที่ประกอบไปด้วยจำนวนไอเท็มทั้งหมดของรายการทรานแซคชันนั้น การทำงานของฟังก์ชันนี้แสดงในรูปที่ 3.2



รูปที่ 3.2 แสดงผังการทำงานส่วนการแปลงรูปแบบข้อมูลจากทรานแซคชันหลายเรคคอร์ดเป็นหนึ่งเรคคอร์ด

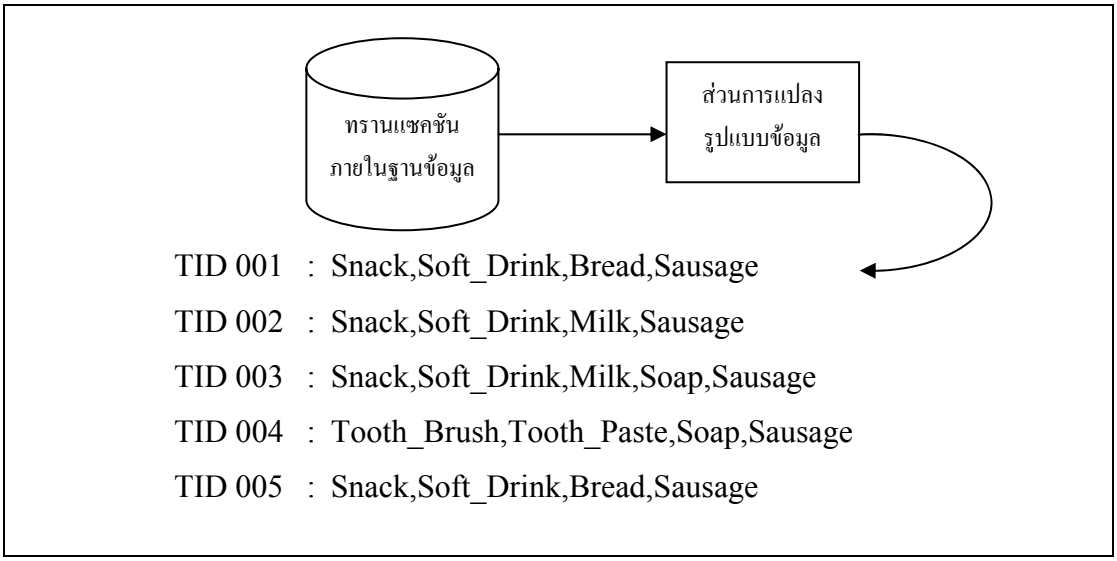
```
p_slct          : select statement
c_dummy         : REF CURSOR
lc_str          : variable
lc_colval       : set of transaction item
p_dlmtr         : delimit
Begin
   open c_dummy for p_slct;
   loop
      fetch c_dummy into lc_colval;
      exit when c_dummy%notfound;
      lc_str := lc_str || p_dlmtr || lc_colval;
   end loop;
End;
```

รูปที่ 3.3 อัลกอริทึมการแปลงรูปแบบข้อมูลจากทรานแซคชันหลายเรคคอร์ดเป็นหนึ่งเรคคอร์ด

TID	ITEM
001	Snack
001	Soft_Drink
001	Bread
001	Sausage
002	Snack
002	Soft_Drink
002	Milk
002	Sausage
003	Snack
003	Soft_Drink
003	Milk

TID	ITEM
003	Soap
003	Sausage
004	Tooth_Brush
004	Tooth_Paste
004	Soap
004	Sausage
005	Snack
005	Soft_Drink
005	Bread
005	Sausage

รูปที่ 3.4 แสดงข้อมูลทรานแซคชันที่จัดเก็บข้อมูลแต่ละไอเท็มเป็นแต่ละเรคคอร์ด



รูปที่ 3.5 แสดงข้อมูลทรานแซคชันเมื่อผ่านอัลกอริทึมการแปลงรูปแบบข้อมูลจากทรานแซคชันหลายเรคคอร์ดเป็นหนึ่งเรคคอร์ด

3.2.3 ส่วนที่ใช้ในการสุ่มข้อมูล

อัลกอริทึมในส่วนนี้จะเป็นส่วนที่ใช้ในการสุ่มข้อมูลโดยจะเป็นส่วนที่ถูกรวมอยู่ในอัลกอริทึมหลักเพียงแต่จะต้องมีการส่งค่าพารามิเตอร์เพื่อเป็นตัวบอกว่าการเรียกใช้อัลกอริทึมนี้ต้องการค้นหากฎความสัมพันธ์จากข้อมูลทั้งหมดหรือว่าต้องการค้นหาจากข้อมูลสุ่ม โดยอัลกอริทึมที่ใช้ในการสุ่มข้อมูลนี้จะเลือกใช้เทคนิคในการสุ่มตัวอย่างโดยใช้หลักทฤษฎีความน่าจะเป็นโดยคัดเลือกมาใช้สองเทคนิคคือ การสุ่มตัวอย่างแบบง่าย และการสุ่มตัวอย่างแบบมีระบบ โดยใช้หลักการสุ่มตัวอย่างแบบใส่คืนที่ และการสุ่มตัวอย่างแบบไม่ใส่คืนที่

1) การสุ่มตัวอย่างแบบง่ายโดยใช้หลักการสุ่มแบบใส่คืนที่ วิธีการสุ่มแบบนี้ก่อนที่จะทำการสุ่มตัวอย่างขึ้นมาจะต้องทำการนับข้อมูลทรานแซคชันทั้งหมดว่ามีข้อมูลทั้งหมดกี่รายการ จากนั้นทำการหาจำนวนตัวอย่างที่ต้องการสุ่มขึ้นมาโดยคิดจาก

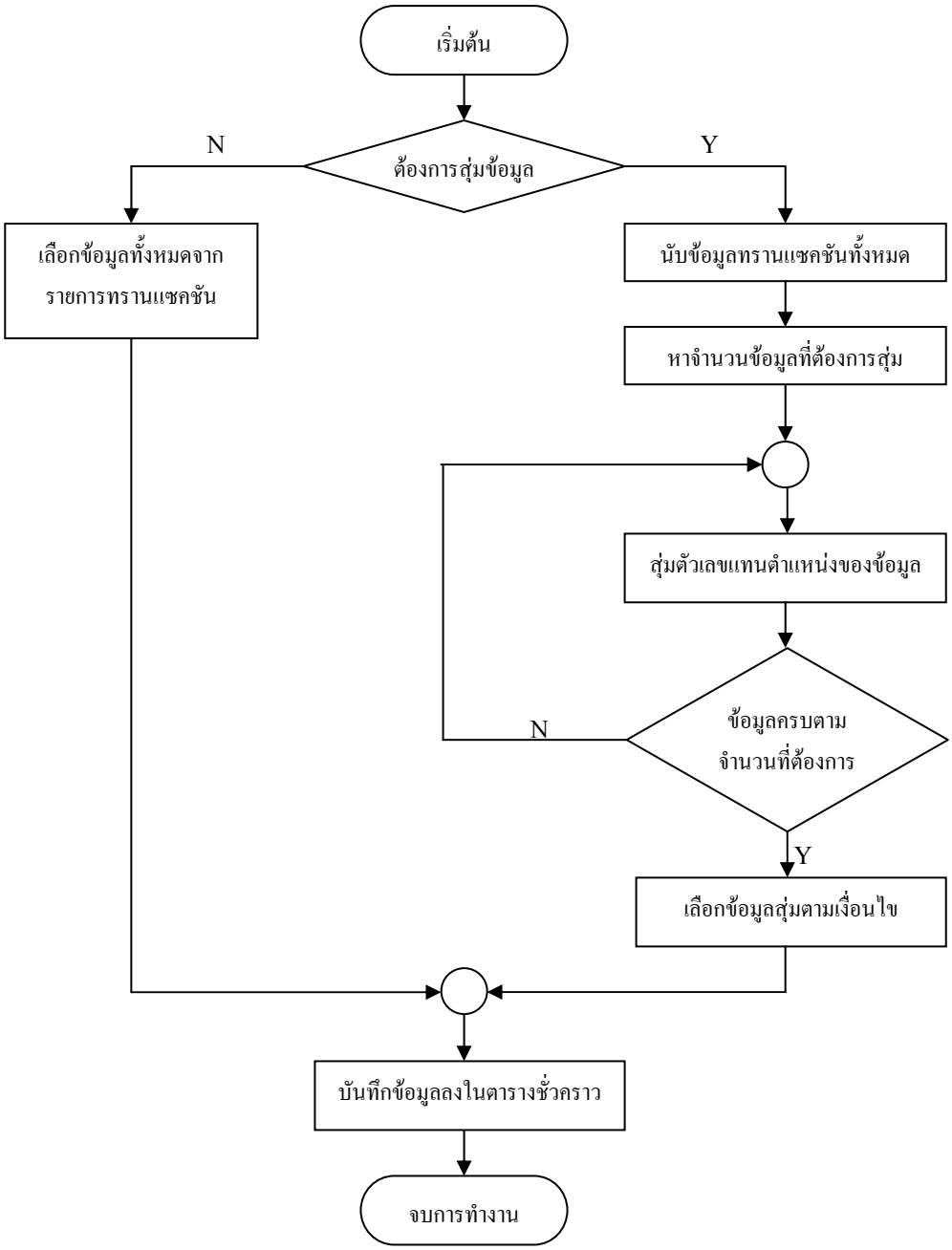
ขนาดของกลุ่มตัวอย่าง (n) = (N * p)/100

N คือ ขนาดของประชากร

p คือ ขนาดของกลุ่มตัวอย่างเป็นเปอร์เซ็นต์

หลังจากที่ได้ขนาดของประชากรตัวอย่างที่ต้องการแล้ว ทำการสุ่มข้อมูลโดยสุ่มตัวเลขขึ้นมาเพื่อนำไปใช้เป็นเงื่อนไขในการดึงข้อมูลสุ่ม โดยตัวเลขที่ถูกสุ่มขึ้นมาได้จะหมายถึง

ข้อมูลที่อยู่ ณ ตำแหน่งนั้นจะถูกดึงมาเป็นประชากรตัวอย่าง ทำเช่นนี้ไปเรื่อยๆ จนกว่าจะได้ข้อมูลครบตามที่ต้องการ โดยการสุ่มโดยวิธีนี้อาจทำให้สุ่มข้อมูลตัวอย่างขึ้นมาซ้ำกันได้ ดังแสดงในภาพ 3.6

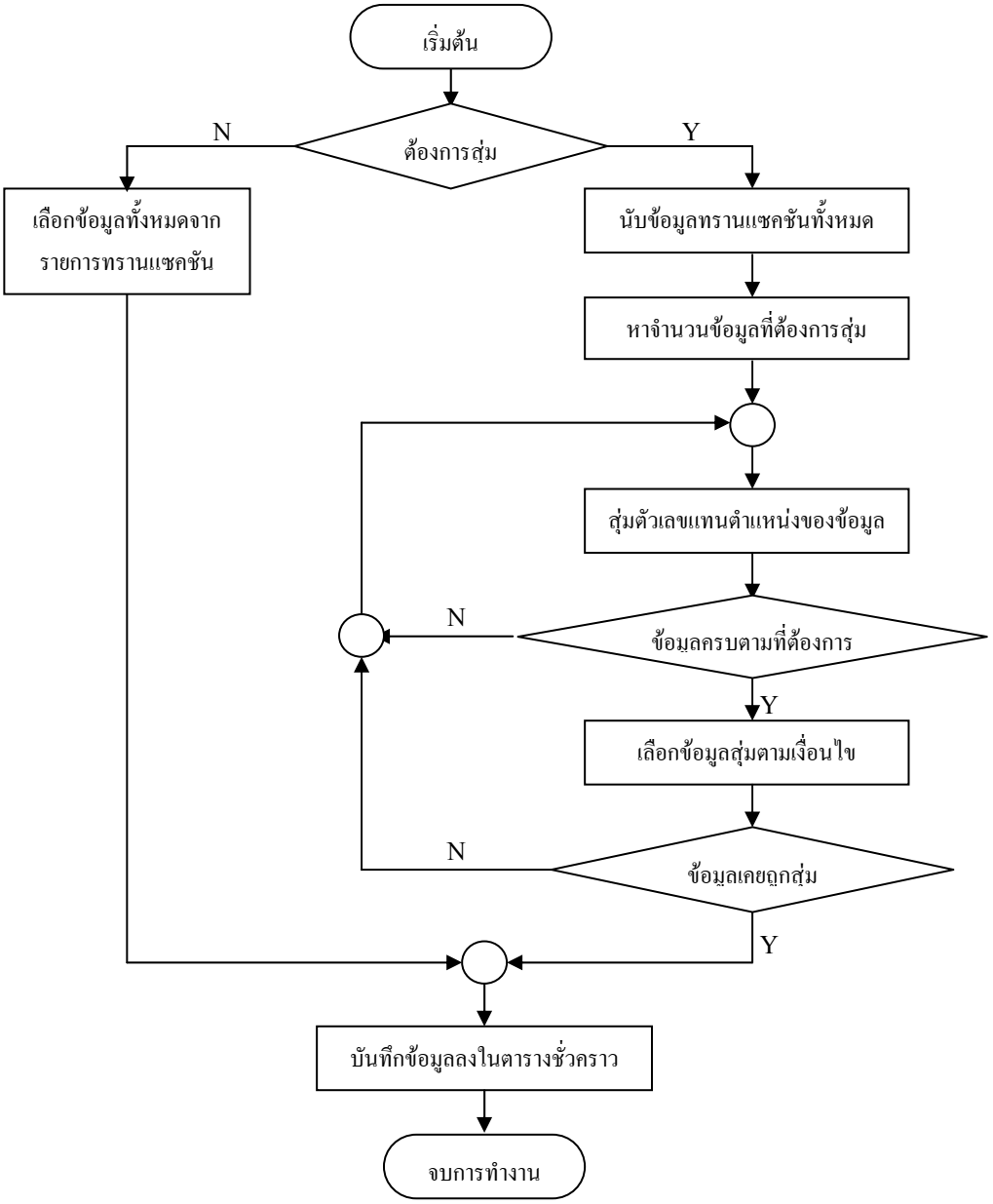


รูปที่ 3.6 แสดงผังการทำงานส่วนการสุ่มตัวอย่างแบบง่ายโดยใช้หลักการสุ่มแบบใส่คืนที่

```
p_slct          : select statement
c_dummy         : REF CURSOR
cnt_tran        : count all transaction
pc_rand         : percent of sample size
sample          : sample size
lc_colval       : store sampling column
begin
   count all transaction;
   if sampling then
      sample := trunc((cnt_tran*pc_rand)/100); -- sample size
      for i in 1..cnt_tran loop
          -- condition for sampling
          sample_rec := sample_rec||sampling_value ||',';
      end loop;
   end if;
   p_slct := p_slct|| ‘ where ’||sample_rec;
   open c_dummy for p_slct ;
   loop
       fetch c_dummy into lc_colval;
       exit when c_dummy%notfound;
       begin
           insert into temporary_table
           values (lc_colval);
       end;
   end loop;
end;
```

รูปที่ 3.7 อัลกอริทึมการสุ่มตัวอย่างแบบง่ายโดยใช้หลักการสุ่มแบบใส่คืนที่

2) การสุ่มตัวอย่างแบบง่ายโดยใช้หลักการสุ่มแบบไม่ใส่คืนที่ วิธีการสุ่มแบบนี้จะมีลักษณะการทำงานเหมือนกับวิธีการสุ่มตัวอย่างแบบง่ายโดยใช้หลักการสุ่มแบบใส่คืนที่ แต่จะแตกต่างกันตรงที่ การสุ่มแบบไม่ใส่คืนที่นี้ข้อมูลที่ถูกสุ่มไปแล้วจะไม่มีโอกาสถูกสุ่มขึ้นมาเป็นตัวอย่างได้อีก โดยจะทำการตรวจสอบว่าถ้าข้อมูลที่ถูกสุ่มขึ้นมาได้มีอยู่แล้วให้ทำการสุ่มข้อมูลตัวใหม่ขึ้นมาแล้วทำการตรวจสอบอีกครั้งจนกว่าจะได้ข้อมูลที่ยังไม่เคยถูกสุ่มขึ้นมา จึงทำการเก็บข้อมูลนั้นไว้เป็นตัวอย่าง ทำเช่นนี้ไปเรื่อยๆ จนกว่าจะได้ข้อมูลตัวอย่างครบตามจำนวนที่ต้องการ ดังแสดงในรูปที่ 3.8

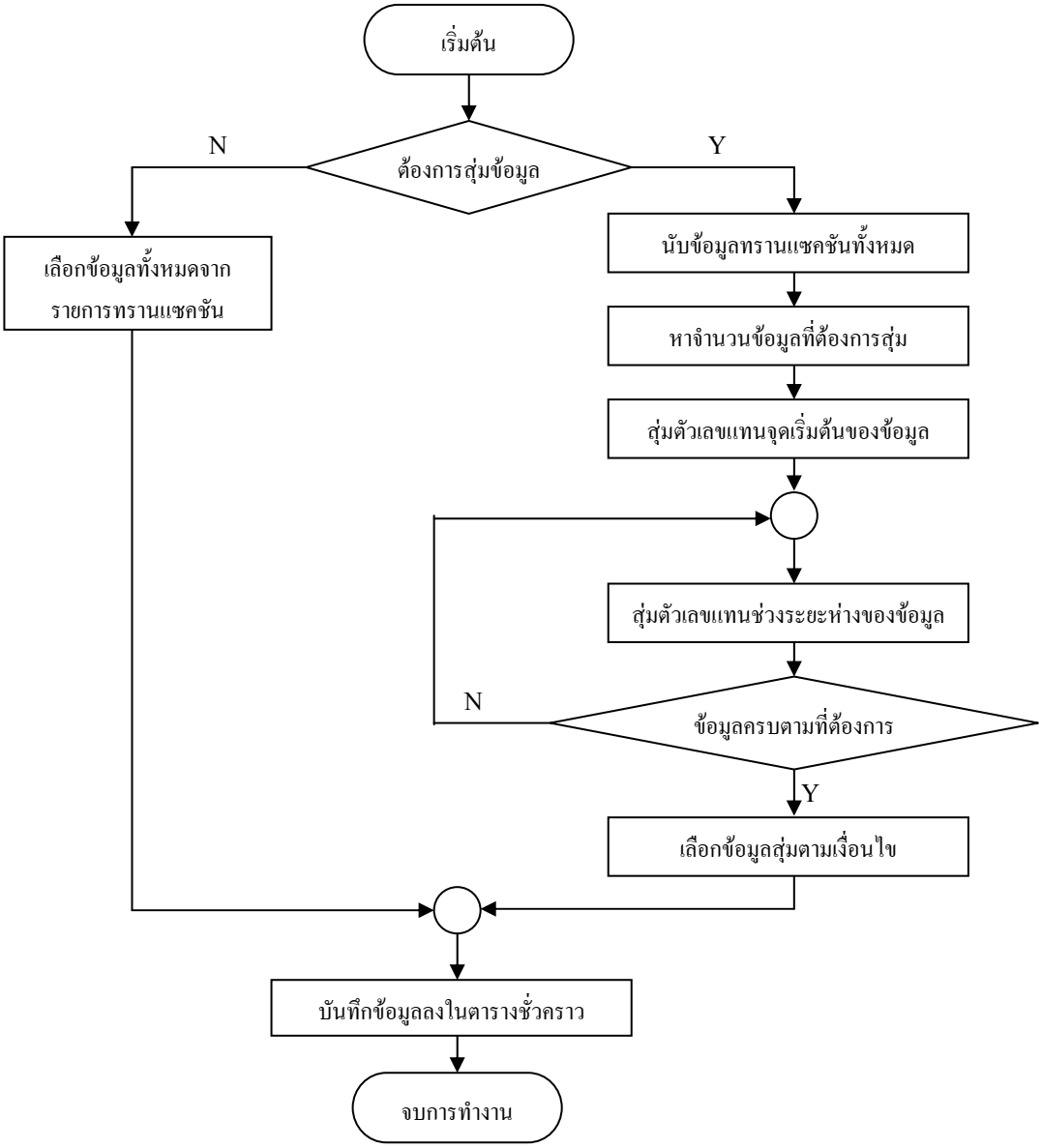


รูปที่ 3.8 แสดงผังการทำงานส่วนการสุ่มตัวอย่างแบบง่ายโดยใช้หลักการสุ่มแบบไม่ใส่คืนที่

```
p_slct          : select statement
c_dummy         : REF CURSOR
cnt_tran        : count all transaction
pc_rand         : percent of sample size
sample          : sample size
lc_colval       : store sampling column
begin
   count all transaction;
   if sampling then
      sample := trunc((cnt_tran*pc_rand)/100);
      for i in 1..cnt_tran loop
          -- condition for sampling
          sample_rec := sample_rec||sampling_value ||',';
      end loop;
   end if;
   p_slct := p_slct|| ‘ where ’||sample_rec;
   open c_dummy for p_slct ;
   loop
       fetch c_dummy into lc_colval;
       exit when c_dummy%notfound;
       begin
           insert into temporary_table
           values (lc_colval);
           exception when dup_val_on_idex then null; -- check duplicate
       end;
   end loop;
end;
```

รูปที่ 3.9 อัลกอริทึมการสุ่มตัวอย่างแบบง่ายโดยใช้หลักการสุ่มแบบไม่ใส่คืนที่

3) การสุ่มตัวอย่างแบบมีระบบโดยใช้หลักการสุ่มแบบใส่คืนที่ วิธีการสุ่มตัวอย่างแบบนี้พัฒนาขึ้นโดยนำวิธีการสุ่มตัวอย่างแบบมีระบบมาประยุกต์โดยจากหลักการสุ่มแบบมีระบบจะต้องมีการกำหนดจุดเริ่มต้นและคำนวณหาระยะห่างระหว่างข้อมูล เมื่อกำหนดจุดเริ่มต้นได้แล้วข้อมูลต่อไปจะมีระยะห่างที่เท่ากัน แต่วิธีที่ทำการพัฒนาขึ้นนั้นตัวเลขที่เป็นตำแหน่งข้อมูลที่ถูกสุ่มขึ้นมาเป็นตัวแรกจะถือว่าเป็นจุดเริ่มต้น จากนั้นในการหาตำแหน่งข้อมูลตัวต่อไปจะทำการสุ่มตัวเลขที่เป็นระยะห่างของข้อมูลขึ้นมาแล้วนำไปบวกเพิ่มกับตำแหน่งของข้อมูลที่เป็นจุดเริ่มต้นก็จะได้ข้อมูลของตัวอย่างตัวถัดไปโดยเมื่อมีการสุ่มตัวเลขขึ้นมาใหม่จะถูกบวกเพิ่มเข้าไปเรื่อยๆ ถ้ามีการอ่านข้อมูลไปจนหมดจำนวนข้อมูลแล้วแต่ยังได้ตัวอย่างไม่ครบตามที่ต้องการก็จะวนกลับมาอ่านที่ข้อมูลลำดับที่หนึ่งต่อไปจนได้ข้อมูลตัวอย่างครบตามที่ต้องการ การหาขนาดของกลุ่มตัวอย่างใช้วิธีเดียวกันกับวิธีการสุ่มตัวอย่างแบบง่ายโดยใช้หลักการสุ่มแบบใส่คืนที่ ดังแสดงในรูปที่ 3.10

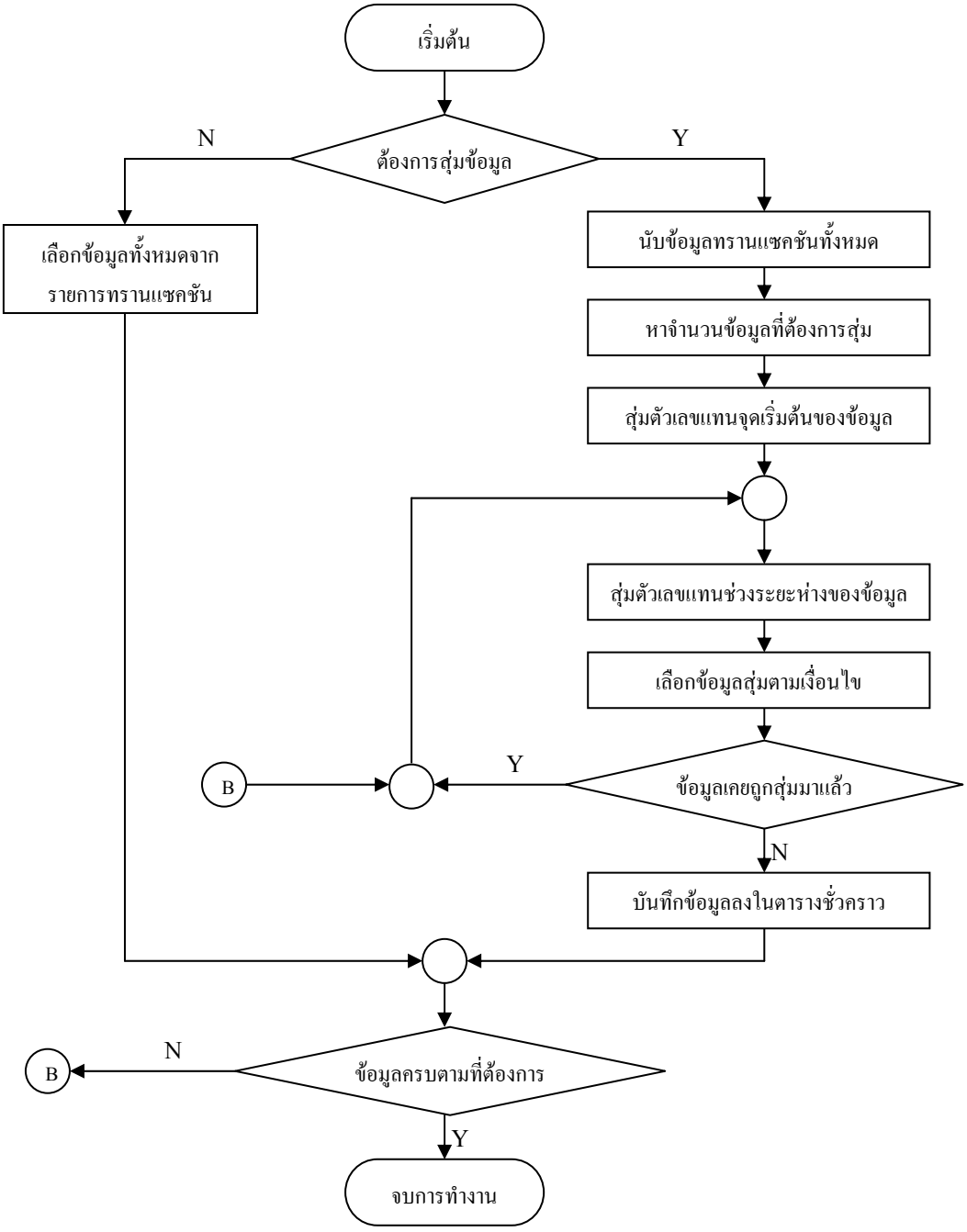


รูปที่ 3.10 แสดงผังการทำงานส่วนการสุ่มตัวอย่างแบบมีระบบโดยใช้หลักการสุ่มแบบใส่คืนที่

```
cnt_tran        : count all transaction
pc_rand         : percent of sample size
sample          : sample size
lc_colval       : store sampling column
begin
   count all transaction;
   if sampling then
      sample := trunc((cnt_tran*pc_rand)/100); -- sample size
      if start then
        rank := findFirstPosition;
      else
        rank := mod(rank + sampling_value, cnt_tran);
      end if;
      begin
         select transaction_id
         into lc_colval
         from transaction
         where row_number = rank;
      end;
      begin
         insert into temporary_table
         values (lc_colval);
      end;
   end if;
      end;
```

รูปที่ 3.11 อัลกอริทึมการสุ่มตัวอย่างแบบมีระบบโดยใช้หลักการสุ่มแบบใส่คืนที่

4) การสุ่มตัวอย่างแบบมีระบบโดยใช้หลักการสุ่มแบบไม่ใส่คืนที่ วิธีการสุ่มแบบนี้จะมีลักษณะการทำงานเหมือนกับวิธีการสุ่มตัวอย่างแบบมีระบบโดยใช้หลักการสุ่มแบบใส่คืนที่ แต่จะแตกต่างกันตรงที่การสุ่มแบบไม่ใส่คืนที่นี้ข้อมูลที่ถูกสุ่มไปแล้วจะไม่มีโอกาสถูกสุ่มขึ้นมาเป็นตัวอย่างได้อีก โดยจะทำการตรวจสอบว่าถ้าข้อมูลที่ถูกสุ่มขึ้นมาได้มีอยู่แล้วให้ทำการสุ่มข้อมูลตัวใหม่ขึ้นมา แล้วทำการตรวจ สอบอีกครั้งจนกว่าจะได้ข้อมูลที่ยังไม่เคยถูกสุ่มขึ้นมาจึงทำการเก็บข้อมูลนั้นไว้เป็นตัวอย่าง ทำเช่นนี้ไปเรื่อยๆ จนกว่าจะได้ข้อมูลตัวอย่างครบตามจำนวนที่ต้องการ ดังแสดงในรูปที่ 3.12

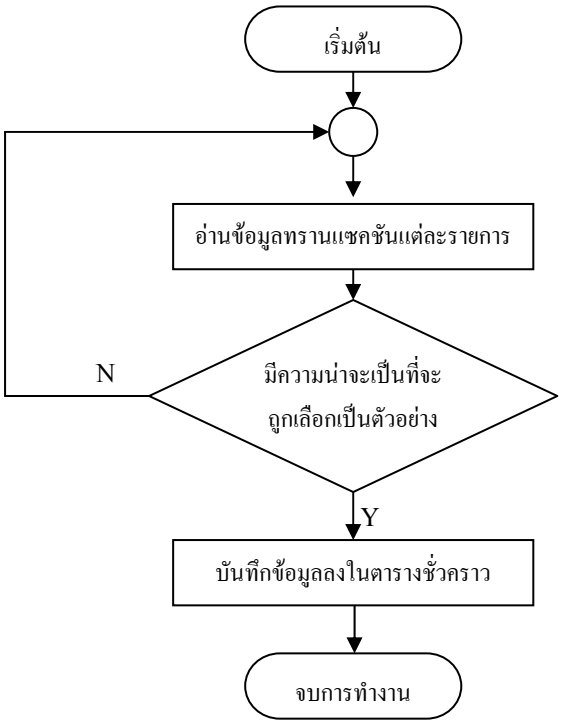


รูปที่ 3.12 แสดงผังการทำงานส่วนการสุ่มตัวอย่างแบบมีระบบโดยใช้หลักการสุ่มแบบไม่ใส่คืนที่

```
cnt_tran        : count all transaction
pc_rand         : percent of sample size
sample          : sample size
lc_colval       : store sampling column
begin
   count all transaction;
   if sampling then
      sample := trunc((cnt_tran*pc_rand)/100); -- sample size
      if start then
        rank := findFirstPosition;
      else
        rank := mod(rank + sampling_value, cnt_tran);
      end if;
      begin
         select transaction_id
         into lc_colval
         from transaction
         where row_number = rank;
      end;
      begin
         insert into temporary_table
         values (lc_colval);
         exception when dup_val_on_index then null; -- check duplicate
      end;
   end if;
         end;
```

รูปที่ 3.13 อัลกอริทึมการสุ่มตัวอย่างแบบมีระบบโดยใช้หลักการสุ่มแบบไม่ใส่คืนที่

5) การสุ่มตัวอย่างแบบง่ายโดยใช้หลักความน่าจะเป็นและใช้หลักการสุ่มแบบไม่ใส่คืนที่ วิธีการสุ่มตัวอย่างแบบนี้พัฒนาขึ้นโดยนำวิธีการสุ่มตัวอย่างแบบง่ายมาประยุกต์โดยจากหลักการสุ่มแบบง่ายจะต้องมีการนับข้อมูลทรานแซคชันทั้งหมดว่ามีจำนวนข้อมูลทั้งหมดกี่รายการ จากนั้นทำการหาจำนวนตัวอย่างที่ต้องการสุ่มขึ้นมา แต่วิธีที่ทำการพัฒนาขึ้นใช้หลักความน่าจะเป็นเข้ามาช่วยในการสุ่มข้อมูล โดยจะต้องกำหนดว่าตัวอย่างที่ต้องการนั้นมีขนาดเท่าไร โดยต้องระบุเป็นเปอร์เซ็นต์ จากนั้นเมื่ออ่านข้อมูลทรานแซคชันขึ้นมาจะมีการหาค่าความน่าจะเป็นว่ารายการทรานแซคชันนั้นมีความน่าจะเป็นที่จะถูกสุ่มไปเป็นตัวอย่างหรือไม่ ทำเช่นนี้ไปเรื่อยๆ จนจบข้อมูลรายการทรานแซคชันก็จะได้ตัวอย่างที่มีขนาดใกล้เคียงกับวิธีการสุ่มตัวอย่างแบบง่าย และตัวอย่างที่ได้มาจะไม่มีสมาชิกที่ซ้ำกัน เนื่องจากวิธีนี้ใช้วิธีการอ่านทรานแซคชันเพียงหนึ่งรอบเท่านั้น ดังนั้นข้อมูลทุกรายการจะถูกอ่านเพียงหนึ่งครั้ง ทำให้ข้อมูลที่ถูกสุ่มไปแล้วจะไม่มีโอกาสถูกสุ่มขึ้นมาได้อีก ดังนั้นวิธีการสุ่มตัวอย่างแบบนี้จะทำงานได้รวดเร็วกว่าวิธีการสุ่มตัวอย่างแบบง่ายทั้งสองวิธีแรก เนื่องจากไม่ต้องคำนวณหาขนาดของตัวอย่าง และไม่ต้องทำการตรวจสอบว่าตัวอย่างที่ถูกสุ่มได้นั้นเคยถูกสุ่มขึ้นมาแล้วหรือไม่ ดังแสดงในรูปที่ 3.14



รูปที่ 3.14 แสดงผังการทำงานส่วนการสุ่มตัวอย่างแบบง่ายโดยใช้หลักความน่าจะเป็นและใช้หลักการสุ่มแบบไม่ใส่คืนที่

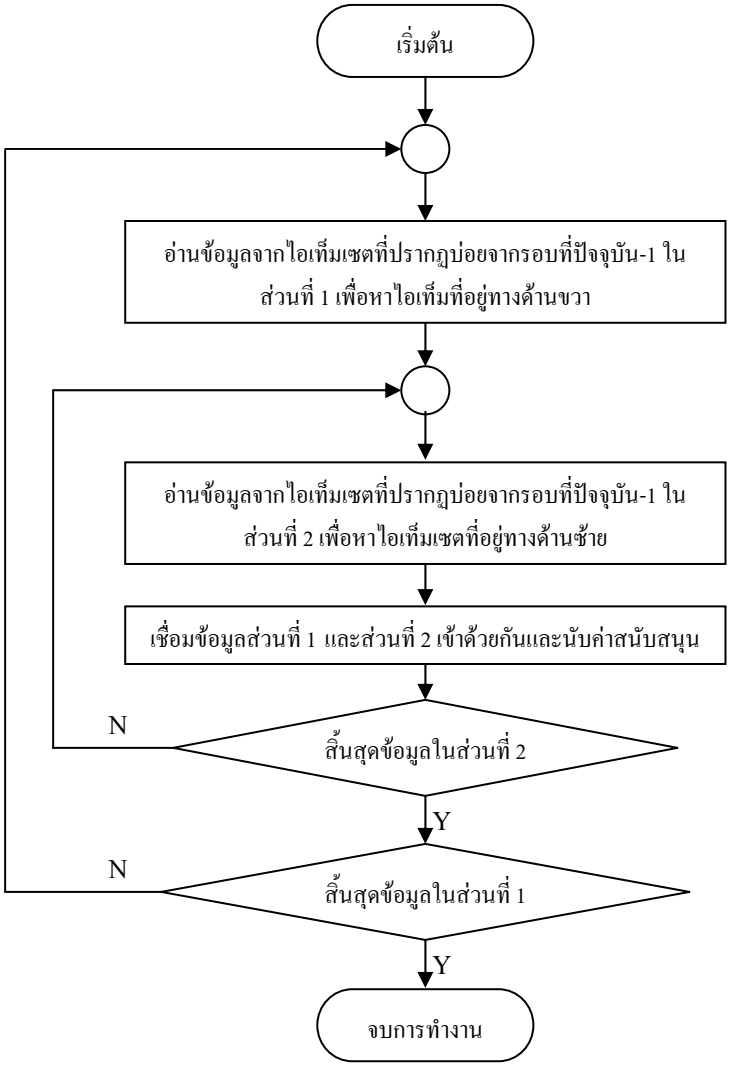
```
p_hslct        : select statement for head transaction
h_dummy        : REF CURSOR for head transaction
pc_rand        : percent of sample size
lc_hcol        : store head transaction column
begin
   open h_dummy for p_hslct ;
   loop
      fetch h_dummy into lc_hcol;
      exit when h_dummy%notfound;
      if  dbms_random.value (0, 100) <= pc_rand then
         begin
            insert into temporary_table
            values (lc_hcol);
         end;
      end if;
   end loop;
      end;
```

รูปที่ 3.15 อัลกอริทึมการสุ่มตัวอย่างแบบง่ายโดยใช้หลักความน่าจะเป็นและใช้หลักการสุ่มแบบไม่ใส่คืนที่

3.2.4 ส่วนการสร้างแคนดิเดทไอเท็มเซต

อัลกอริทึมในส่วนนี้จะเป็นส่วนที่ใช้ในสร้างแคนดิเดทไอเท็มเซตหรือไอเท็มเซตที่มีความเป็นไปได้ที่จะเกิดขึ้น การสร้างแคนดิเดทไอเท็มเซตขึ้นเป็นครั้งแรกไอเท็มที่มีปรากฏขึ้นในรายการทรานแซคชันทั้งหมดจะถูกนำมาสร้างเป็นแคนดิเดทไอเท็มเซต โดยเป็นเซตที่มีสมาชิกเพียงหนึ่งตัวคือไอเท็มแต่ละตัวจากนั้นจะทำการนับค่าสนับสนุนของแต่ละไอเท็ม ถ้าแคนดิเดทไอเท็มเซตที่มีค่าสนับสนุนมากกว่า หรือเท่ากับค่าสนับสนุนขั้นต่ำที่กำหนดไว้จะถูกเรียกว่าไอเท็มเซตที่ปรากฏบ่อย จากนั้นจะถูกนำมาทำการสร้างแคนดิเดทไอเท็มเซตในรอบต่อไป ในการสร้างแคนดิเดทไอเท็มเซตนั้นจะทำการอ่านข้อมูลไอเท็มเซตที่ปรากฏบ่อยโดยอ่านจากรอบที่ปัจจุบัน - 1 โดยการอ่านข้อมูลในส่วนที่ 1 จะอ่านข้อมูลโดยดึงข้อมูลไอเท็มเซตที่ปรากฏบ่อยและไอเท็มที่ปรากฏ

เป็นตัวสุดท้ายของแต่ละไอเท็มเซตที่ปรากฏบ่อยเพื่อนำมาใช้เป็นเงื่อนไขในการดึงข้อมูลในส่วนที่ 2 เมื่อได้ข้อมูลจากส่วนที่ 1 แล้วจะทำการอ่านข้อมูลในส่วนที่ 2 จะอ่านข้อมูลเฉพาะข้อมูลไอเท็มเซตที่ปรากฏบ่อยที่เกิดจากการนับค่าสนับสนุนในรอบการอ่านแรกโดยมีเงื่อนไขว่าไอเท็มที่ปรากฏเป็นตัวสุดท้ายของแต่ละไอเท็มเซตที่ปรากฏบ่อยที่ได้จากการอ่านข้อมูลในส่วนที่ 1 จะต้องมีค่าน้อยกว่าไอเท็มเซตที่ปรากฏบ่อยในรอบการอ่านแรกของข้อมูลในส่วนที่ 2 เพื่อเป็นการป้องกันการสร้างแคนดิเดทไอเท็มเซตที่ซ้ำกัน ดังแสดงในภาพ 3.16 หลังจากที่ได้ข้อมูลทั้งสองส่วนแล้ว นำข้อมูลทั้งสองส่วนมาเชื่อมกันเพื่อสร้างเป็นแคนดิเดทไอเท็มเซตโดยให้ข้อมูลไอเท็มเซตที่ได้จากส่วนที่ 2 เป็นไอเท็มแรกแล้วเชื่อมกับไอเท็มเซตในส่วนที่ 1 หลังจากที่ได้แคนดิเดทไอเท็มเซตชุดใหม่แล้วก็จะนำไปผ่านกระบวนการนับค่าสนับสนุน และสร้างไอเท็มเซตที่ปรากฏบ่อยและจะทำงานซ้ำไปเรื่อยๆ จนกว่าจะไม่สามารถสร้างแคนดิเดทไอเท็มเซตชุดใหม่ได้อีกอัลกอริทึมจึงจะหยุดการทำงานในส่วนนี้และเข้าสู่กระบวนการสร้างกฎความสัมพันธ์ ดังแสดงในรูปที่ 3.16



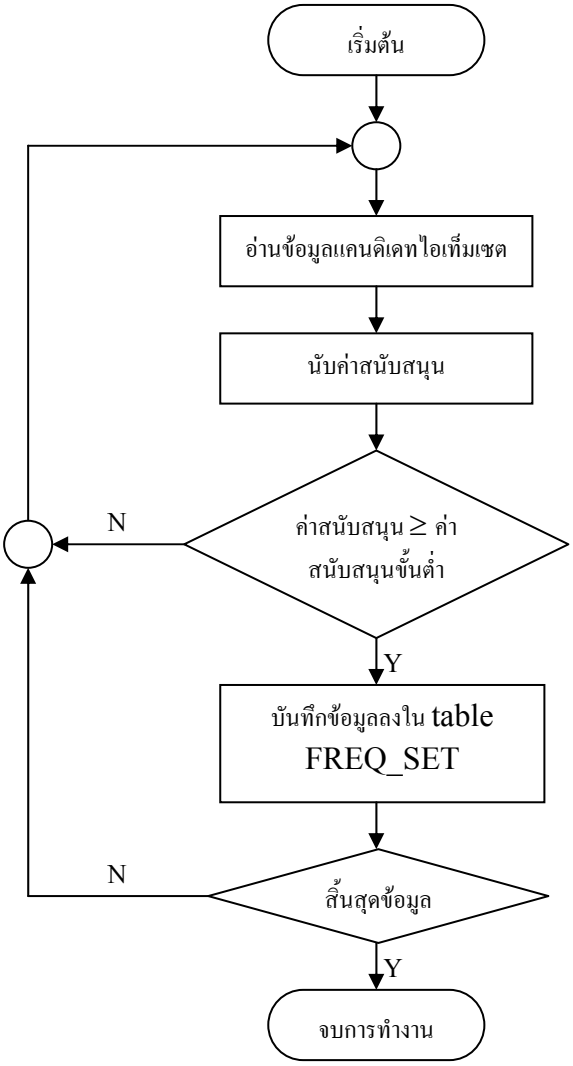
รูปที่ 3.16 แสดงผังการทำงานส่วนการสร้างแคนดิเดทไอเท็มเซต

```
Cursor Sec1 is          select   item,
                        substr(item, instr(item, ',', -1, 1)+1)) l_item
                        from     Fk-1;
Cursor Sec2 (l_item)    select   item r_item
                        from     F1
                        where    l_item < item;
begin
   for S1 in Sec1 loop
      for S2 (l_item) in Sec2 loop
         insert into candidate_table
         values (l_item|| ',' ||r_item);
         count support;
      end loop;
   end loop;
end;
```

รูปที่ 3.17 อัลกอริทึมการสร้างแคนดิเดทไอเท็มเซต

3.2.5 ส่วนการนับค่าสนับสนุน และสร้างไอเท็มเซตที่ปรากฏบ่อย

อัลกอริทึมในส่วนนี้จะเป็นส่วนที่ใช้ในการนับค่าสนับสนุนของแคนดิเดทไอเท็มเซตที่ถูกสร้างขึ้นมาจากส่วนการทำงานที่ 4 โดยการนับค่าสนับสนุนคือการนับค่าความบ่อยครั้งในการเกิดของไอเท็มเซตนั้น โดยเมื่อค่าสนับสนุนที่นับได้มีค่ามากกว่าหรือเท่ากับค่าสนับสนุนขั้นต่ำที่กำหนดไว้จะถูกเรียกว่า ไอเท็มเซตที่ปรากฏบ่อย และจะทำการบันทึกไอเท็มเซตที่ปรากฏบ่อยลงในตารางที่ใช้เก็บข้อมูลของไอเท็มเซตที่ปรากฏบ่อย (Table FREQ_SET) จากนั้นจึงนำไอเท็มเซตที่ปรากฏบ่อยไปสร้างแคนดิเดทไอเท็มเซตต่อไป ดังแสดงในรูปที่ 3.18



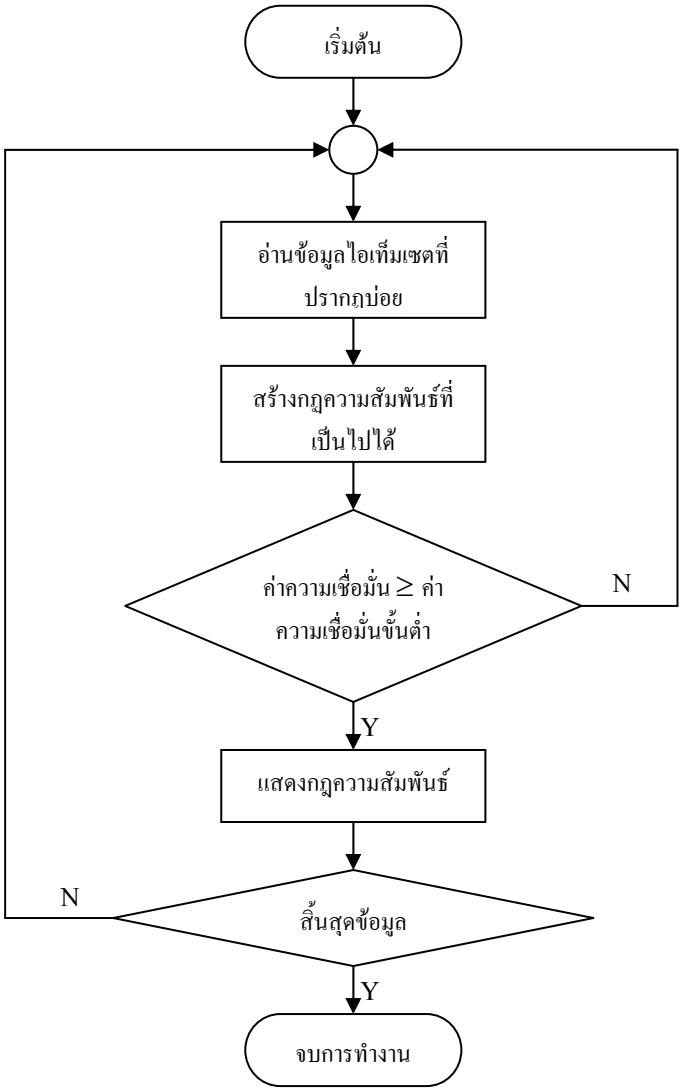
รูปที่ 3.18 แสดงผังการทำงานส่วนการนับค่าสนับสนุน และสร้างไอเท็มเซตที่ปรากฏบ่อย

```
begin
   insert into frequent
   select candidate_item
   from   candidate
   where  support >= minimum_support;
end;
```

รูปที่ 3.19 อัลกอริทึมการนับค่าสนับสนุน และสร้างไอเท็มเซตที่ปรากฏบ่อย

3.2.6 ส่วนการสร้างกฎความสัมพันธ์

อัลกอริทึมในส่วนนี้จะเป็นส่วนที่ใช้ในการสร้างกฎความสัมพันธ์โดยในส่วนนี้จะนำข้อมูลไอเท็มเซตที่ปรากฏขึ้นบ่อยมาสร้างเป็นกฎความสัมพันธ์ โดยกฎความสัมพันธ์ที่ถูกสร้างขึ้นมาจะอยู่ในรูปแบบ “ถ้า $\rightarrow$ แล้ว” (IF $\rightarrow$ THEN) โดยจะนำไอเท็มเซตที่ปรากฏบ่อยแต่ละชุดมาสร้างกฎความสัมพันธ์ที่มีความเป็นไปได้จากนั้นตรวจสอบดูว่าถ้ากฎความสัมพันธ์ที่สร้างขึ้นมานั้นมีค่าความเชื่อมั่นมากกว่า หรือเท่ากับค่าความเชื่อมั่นขั้นต่ำที่กำหนดไว้จะถูกนำออกมาแสดงเป็นกฎความสัมพันธ์ดังแสดงในรูปที่ 3.20



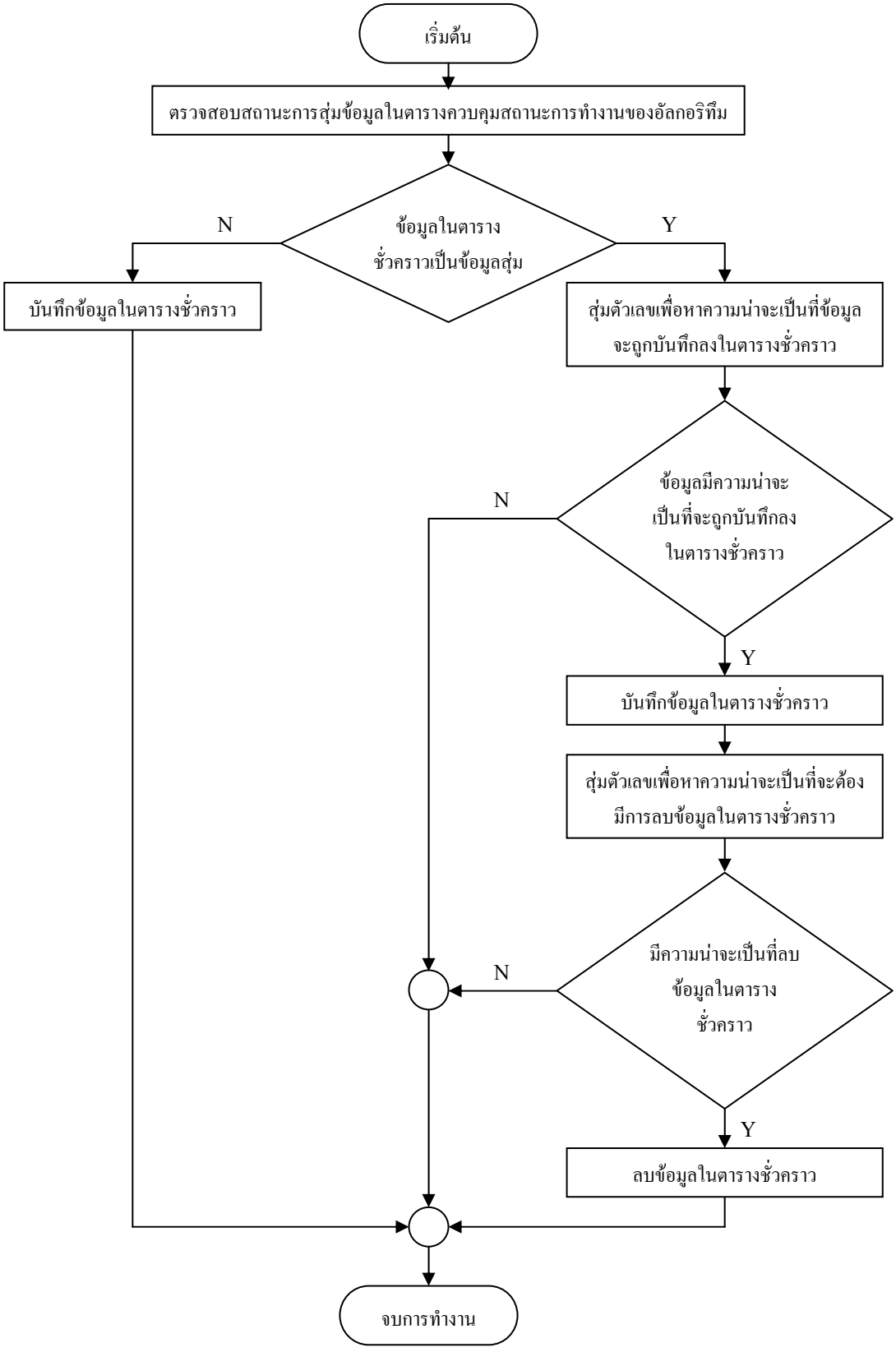
รูปที่ 3.20 แสดงผังการทำงานส่วนการสร้างกฎความสัมพันธ์

```
Cursor Freq is          select   frequent_item,
                        from     frequent;
begin
   for F in Freq loop
       cnt_left      := countSupport.leftItemSet;
       cnt_item      := countSupport.ItemSet;
       confidence  := (cnt_item/cnt_left);
       if confidence >= min_confidence then
          print_rules;
       end if;
   end loop;
end;
```

รูปที่ 3.21 อัลกอริทึมการสร้างกฎความสัมพันธ์

3.2.7 ส่วนการจัดการข้อมูลสตรีมมิง

อัลกอริทึมในส่วนนี้จะเป็นส่วนที่ใช้ในการจัดการกับข้อมูลสตรีมมิง หรือข้อมูลที่ถูกบันทึกเข้ามา ณ ขณะใดๆ และเป็นข้อมูลที่ไม่มีขอบเขตจำกัด โดยใช้ดาต้าเบสทริกเกอร์เป็นตัวจัดการ คือ เมื่อมีการบันทึกข้อมูลเข้ามาในฐานข้อมูล ดาต้าเบสทริกเกอร์จะทำการตรวจสอบสถานะการสุ่มข้อมูลในตารางควบคุมสถานะการทำงานของอัลกอริทึม ว่าข้อมูลที่ถูกประมวลผลแล้วบันทึกลงในตารางชั่วคราวได้มาจากการสุ่มข้อมูลจากฐานข้อมูลหรือไม่ ถ้าไม่ได้เป็นข้อมูลที่ถูกสุ่มมา ดาต้าเบสทริกเกอร์จะทำการบันทึกข้อมูลที่ถูกบันทึกเข้ามา ณ ขณะนั้นลงในตารางชั่วคราวทันที แต่ถ้าข้อมูลที่มีอยู่ในตารางชั่วคราวเป็นข้อมูลที่ถูกสุ่มขึ้นมา ดาต้าเบสทริกเกอร์จะทำการสุ่มตัวเลขขึ้นมาเพื่อตรวจสอบดูว่า ข้อมูลที่ถูกบันทึกเข้ามานั้นจะถูกบันทึกลงในตารางชั่วคราวเพื่อนำไปใช้ในกระบวนการค้นหากฎความ-สัมพันธ์หรือไม่ โดยความน่าจะเป็นที่ข้อมูลจะถูกบันทึกลงในตารางชั่วคราวนั้นมีอยู่ 50 เปอร์เซนต์ เมื่อตรวจสอบแล้วว่าข้อมูลมีความน่าจะเป็นที่จะถูกบันทึกลงในตารางชั่วคราวก็จะทำการสุ่มตัวเลขอีกครั้งหนึ่ง เพื่อตรวจสอบว่าเมื่อบันทึกข้อมูลลงในตารางชั่วคราวแล้วจะต้องมีการลบข้อมูลที่อยู่ในตารางชั่วคราวออกด้วยหรือไม่ และถ้าต้องลบข้อมูลออกจะต้องทำการสุ่มข้อมูลจากตารางชั่วคราวเพื่อหาว่าจะต้องลบข้อมูลตัวใดออก ดังแสดงในรูปที่ 3.22



รูปที่ 3.22 แสดงผังการทำงานส่วนการจัดการข้อมูลสตรีมมิง

```
v_rand          : random value
v_tcnt          : count all TRAN_TEMP
v_tid           : transaction id
v_newtid        : new transaction id
v_rand_status   : sampling status
begin
   check sampling status from MINING_CONTROL;
   if sampling then
      v_rand := random value (0, 100); -- insert probability
      if v_rand > 50 then
        begin
          insert into tran_temp (tid, random_status)
          values (v_newtid, v_rand_status);
        end;
        v_rand := random value (0, 100); -- delete TRAN_TEMP probability
        if v_rand > 50 then
          count TRAN_TEMP;
          find record for delete;
          delete from TRAN_TEMP;
        end if;
      end if;
   else
      begin
        insert into tran_temp (tid, random_status)
        values (v_newtid, v_rand_status);
      end;
   end if;
```

รูปที่ 3.23 อัลกอริทึมส่วนการจัดการข้อมูลสตรีมมิง

3.3 การทำงานของโปรแกรม SASS

3.3.1 การทำงานของโปรแกรม SASS กับข้อมูลทรานแซคชัน

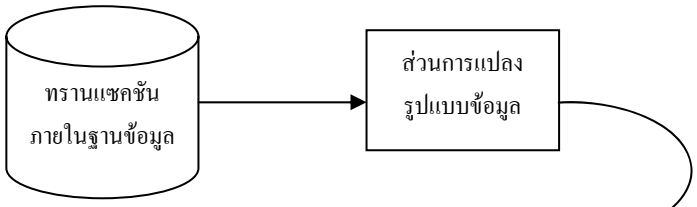
คำอธิบายในส่วนนี้เป็นการจำลองการทำงานของโปรแกรม SASS ที่ทำการพัฒนาขึ้น เพื่อสร้างความเข้าใจการทำงานของอัลกอริทึมด้วยตัวอย่างดังต่อไปนี้

ตัวอย่าง พิจารณารายการทรานแซคชันการขายสินค้าของร้านสะดวกซื้อแห่งหนึ่ง โดยพิจารณาสินค้า 8 ชนิด (Snack, Soft Drink, Milk, Soap, Sausage, Tooth Brush, Tooth Paste) โดยรายการซื้อสินค้าของลูกค้าแสดงในรูปที่ 3.24 แต่ละแถวของตารางในภาพหมายถึงรายการทรานแซคชัน โดยรายการทรานแซคชันที่มีตัวชี้ทรานแซคชัน หรือหมายเลขทรานแซคชันเดียวกันถือเป็นหนึ่งรายการทรานแซคชัน ในการค้นหากฎความสัมพันธ์กำหนดให้อัลกอริทึมทำการค้นหาความสัมพันธ์จากข้อมูลสุ่ม 50 เปอร์เซ็นต์ กฎความสัมพันธ์ต้องมีค่าสนับสนุนขั้นต่ำไม่ต่ำกว่า 80 เปอร์เซ็นต์ และมีค่าความเชื่อมั่นขั้นต่ำไม่ต่ำกว่า 95 เปอร์เซ็นต์ การทำงานในส่วนต่างๆ ของอัลกอริทึมสามารถได้ดังรูปที่ 3.25 ถึง 3.28 โดยรูปที่ 3.25 เป็นภาพแสดงการทำงานในส่วนการแปลงรูปแบบของข้อมูล รูปที่ 3.26 แสดงรายการทรานแซคชันภายหลังจากการสุ่มข้อมูล รูปที่ 3.27 แสดงการสร้างแคนดิเดทไอเท็มเซต การสร้างไอเท็มเซตที่ปรากฏบ่อย และการนับค่าสนับสนุน รูปที่ 3.28 แสดงกฎความสัมพันธ์ที่ค้นพบจากข้อมูล เมื่อมีการเรียกใช้งานโปรแกรม SASS ข้อมูลจะถูกอ่านขึ้นมาจากฐานข้อมูลและส่งเข้าไปแปลงรูปแบบของข้อมูล โดยจะแปลงจากข้อมูลจากทรานแซคชันหลายเรคคอร์ดที่มีตัวชี้ทรานแซคชัน (TID) หรือหมายเลขทรานแซคชันตัวเดียวกันให้เป็นหนึ่งเรคคอร์ดซึ่งจะได้ผลลัพธ์ดังรูปที่ 3.23 หลังจากที่ข้อมูลถูกแปลงให้อยู่ในรูปแบบหนึ่งเรคคอร์ดแล้วโปรแกรมจะทำการตรวจสอบค่าพารามิเตอร์ที่ถูกส่งเข้ามาว่าจะต้องทำการสุ่มข้อมูลหรือไม่ โดยในที่นี้ให้ทำการสุ่มข้อมูล 50% แสดงได้ดังรูปที่ 3.24

TID	ITEM
001	Snack
001	Soft_Drink
001	Bread
001	Sausage
002	Snack
002	Soft_Drink
002	Milk
002	Sausage
003	Snack
003	Soft_Drink
003	Milk
003	Soap
003	Sausage
004	Tooth_Brush
004	Tooth_Paste
004	Soap
004	Sausage

TID	ITEM
005	Snack
005	Soft_Drink
005	Bread
005	Sausage
006	Tooth_Brush
006	Tooth_Paste
006	Soap
006	Sausage
007	Snack
007	Soft_Drink
008	Snack
008	Soft_Drink
008	Sausage
009	Snack
009	Soft_Drink
009	Milk
009	Bread
010	Snack
010	Soft_Drink

รูปที่ 3.24 แสดงข้อมูลทรานแซคชันที่ถูกจัดเก็บไว้ภายในฐานข้อมูล



TID 001 : Snack,Soft_Drink,Bread,Sausage
TID 002 : Snack,Soft_Drink,Milk,Sausage
TID 003 : Snack,Soft_Drink,Milk,Soap,Sausage
TID 004 : Tooth_Brush,Tooth_Paste,Soap,Sausage
TID 005 : Snack,Soft_Drink,Bread,Sausage
TID 006 : Tooth_Brush,Tooth_Paste,Soap,Sausage
TID 007 : Snack,Soft_Drink
TID 008 : Snack,Soft_Drink,Sausage
TID 009 : Snack,Soft_Drink,Milk,Bread
TID 010 : Snack,Soft_Drink

รูปที่ 3.25 แสดงข้อมูลทรานแซคชันที่ถูกแปลงให้อยู่ในรูปแบบหนึ่งเรคคอร์ด

TID 002 : Snack,Soft_Drink,Milk,Sausage
TID 003 : Snack,Soft_Drink,Milk,Soap,Sausage
TID 005 : Snack,Soft_Drink,Bread,Sausage
TID 007 : Snack,Soft_Drink
TID 009 : Snack,Soft_Drink,Milk,Bread

รูปที่ 3.26 แสดงข้อมูลทรานแซคชันหลังจากที่ผ่านการสุ่มข้อมูล

หลังจากที่ผ่านการสุ่มข้อมูลแล้วจะทำการสร้างแคนดิเดทไอเท็มเซตจากข้อมูลที่เหลืออยู่ จากนั้นทำการนับค่าสนับสนุนเพื่อนำแคนดิเดทไอเท็มเซตที่มีค่าสนับสนุนมากกว่า หรือเท่ากับค่าสนับสนุนขั้นต่ำที่กำหนดไว้มาสร้างเป็นไอเท็มเซตที่ปรากฏบ่อย และนำไอเท็มเซตที่ปรากฏบ่อยมาสร้างเป็นแคนดิเดทไอเท็มเซตในระดับต่อไป โปรแกรมจะทำเช่นนี้ไปเรื่อยๆ จนกว่าจะไม่สามารถหาไอเท็มเซตที่ปรากฏบ่อยได้จึงจะสร้างกฎความสัมพันธ์ ในที่นี้กำหนดให้ต้องมีค่าสนับสนุนไม่น้อยกว่า 80 เปอร์เซ็นต์ ดังแสดงในรูปที่ 3.25

1

C1	Support
Snack	5
Soft_Drink	5
Milk	3
Sausage	3
Soap	1
Bread	2

แคนดิเดทไอเท็มเซตระดับที่ 1

2

F1	Support
Snack	5
Soft_Drink	5

ไอเท็มเซตที่ปรากฏบ่อยระดับที่ 1

3

C2	Support
Snack, Soft_Drink	5

แคนดิเดทไอเท็มเซตระดับที่ 2

4

F2	Support
Snack, Soft_Drink	5

ไอเท็มเซตที่ปรากฏบ่อยระดับที่ 2

5

C3	Support
no item	

แคนดิเดทไอเท็มเซตระดับที่ 3

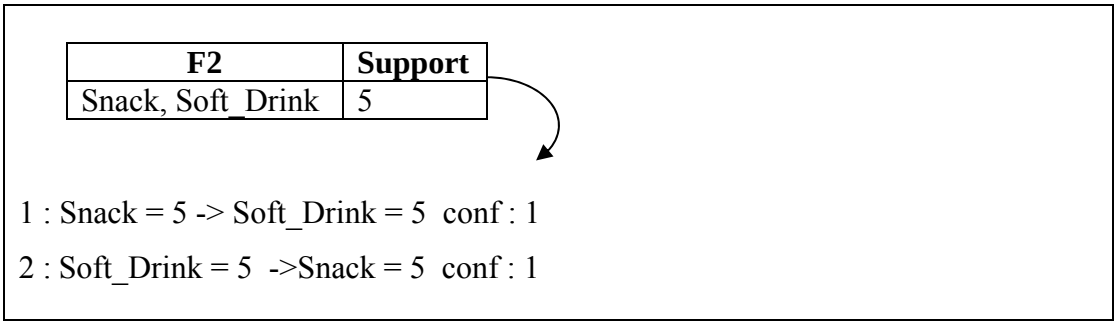
6

F3	Support
no item	

ไอเท็มเซตที่ปรากฏบ่อยระดับที่ 3

รูปที่ 3.27 แสดงการสร้างแคนดิเดทไอเท็มเซต และไอเท็มเซตที่ปรากฏบ่อย

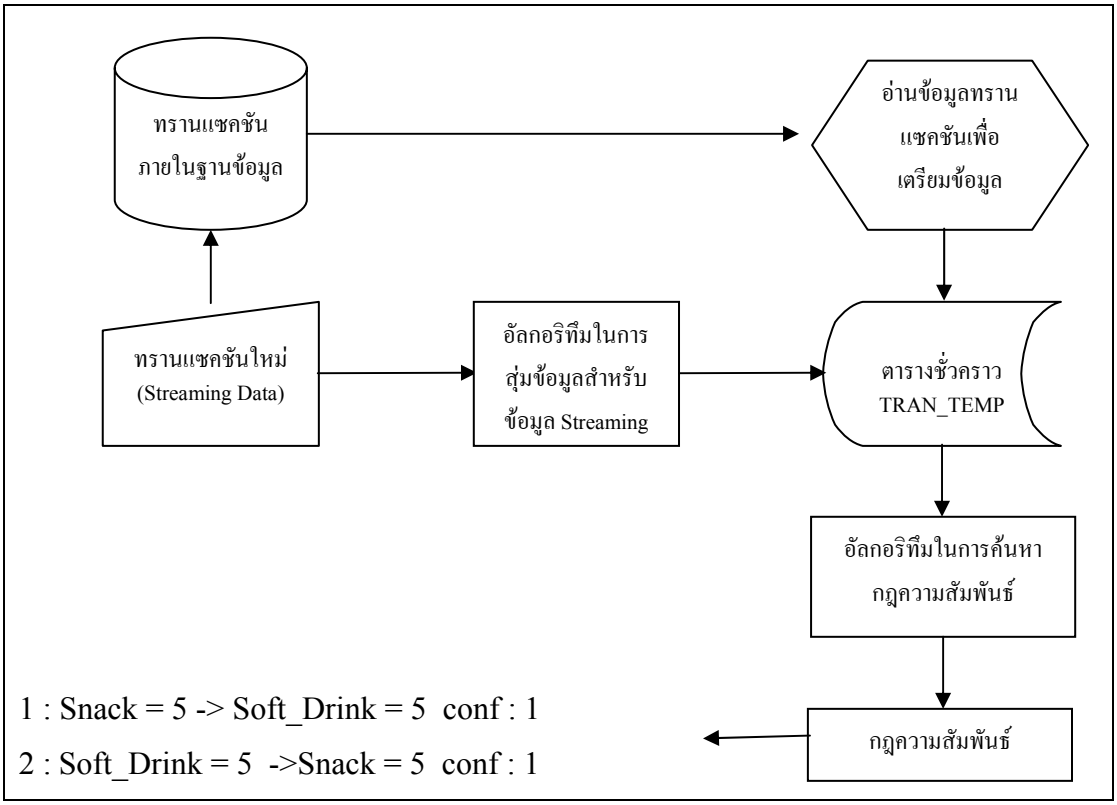
หลังจากที่โปรแกรมไม่สามารถสร้างแคนดิเดทไอเท็มเซต และไอเท็มเซตที่ปรากฏบ่อยได้แล้วก็จะทำการสร้างกฎความสัมพันธ์ โดยนำเอาไอเท็มเซตที่ปรากฏบ่อยมาสร้างเป็นกฎความสัมพันธ์ และกฎความสัมพันธ์ที่ถูกสร้างขึ้นนั้นจะต้องมีค่าความเชื่อมั่นมากกว่า หรือเท่ากับค่าความเชื่อมั่นขั้นต่ำที่กำหนดไว้ ในที่นี้กำหนดให้กฎความสัมพันธ์ต้องมีค่าความเชื่อมั่นไม่น้อยกว่า 95 เปอร์เซ็นต์ ดังแสดงในรูปที่ 3.28



รูปที่ 3.28 แสดงการสร้างกฎความสัมพันธ์

3.3.2 การทำงานของโปรแกรม SASS กับข้อมูลทรานแซคชันและข้อมูลสตรีมมิง

ในส่วนนี้เป็นการจำลองการทำงานของโปรแกรม SASS โดยเป็นการทำงานกับข้อมูลทรานแซคชันที่มีอยู่แล้วในฐานข้อมูลและข้อมูลสตรีมมิง ดังแสดงในรูปที่ 3.29



รูปที่ 3.29 แสดงการทำงานของโปรแกรม SASS ในการค้นหากฎความสัมพันธ์กับข้อมูลทรานแซคชัน และข้อมูลสตรีมมิง

3.4 แหล่งที่มาของข้อมูล

การทดสอบอัลกอริทึมที่พัฒนาขึ้นนี้ใช้ข้อมูลในการทดสอบทั้งสิ้นจำนวน 7 ชุด สืบค้นจากแหล่งข้อมูลของมหาวิทยาลัยแห่งรัฐแคลิฟอร์เนีย เมืองเออร์ไวน์ (University of California at Irvine) (http://www.ics.uci.edu/~mlearn/MLReposi tory.html) โดยทำการคัดเลือกข้อมูลมาจำนวน 6 ชุด และอีก 1 ชุดได้จากการสังเคราะห์ขึ้น ชุดข้อมูลทั้ง 7 ชุดนี้จำแนกได้เป็น 2 กลุ่มคือ

1) ข้อมูลที่มีลักษณะของข้อมูลที่มีความตรงกันมาก หรือมีข้อมูลเหมือนกันเกิดขึ้นซ้ำมากๆ ได้แก่ข้อมูล Census Income, Mushroom, Soybean และ Convenient store ดังแสดงในภาพ 3.30

```
TID1  : edu_inst Not_in_universe,birth United_States,citizenship
Native_Born_in_ the_United_States,salary -50000
TID2  : edu_inst Not_in_universe,birth United_States,citizenship
Native_Born_in_ the_United_States,salary -50000
TID3  : edu_inst High_school,birth Vietnam,citizenship
Foreign_born_Not_a_ citizen_of_US,salary -50000
TID4  : edu_inst Not_in_universe,birth United_States,citizenship
Native_Born_in_ the_United_States,salary -50000
TID5  : edu_inst Not_in_universe, birth United_States,citizenship
Native_Born_in_ the_United_States, salary -50000
TID6  : edu_inst Not_in_universe,birth United_States,citizenship
Native_Born_in_ the_United_States,salary -50000
TID7  : edu_inst Not_in_universe,birth United_States,citizenship
Native_Born_in_ the_United_States,salary -50000
TID8  : edu_inst Not_in_universe,birth United_States,citizenship
Native_Born_in_ the_United_States,salary -50000
TID9  : edu_inst Not_in_universe,birth United_States,citizenship
Native_Born_in_ the_United_States,salary -50000
TID10 : edu_inst Not_in_universe,birth United_States,citizenship
Native_Born_in_ the_United_States,salary -50000
```

รูปที่ 3.30 ตัวอย่างข้อมูลที่มีลักษณะของข้อมูลที่มีความตรงกันมาก หรือมีข้อมูลเหมือนกันเกิดขึ้นซ้ำมากๆ

2) ข้อมูลที่มีลักษณะของข้อมูลแตกต่างกันมาก หรือมีข้อมูลเหมือนกันเกิดขึ้นซ้ำกันไม่มาก หรืออาจไม่มีการซ้ำกันเลย ได้แก่ข้อมูล Adult, Thyroid และ Nursery ดังแสดงในภาพ 3.31

```
TID1  : parents usual,social nonproblem,health recommended,class
recommend
TID2  : parents usual,social nonproblem,health priority,class
priority
TID3  : parents usual,social nonproblem,health not_recom,class
not_recom
TID4  : parents usual,social slightly_problem,health
recommended,class recommend
TID5  : parents usual,social slightly_problem,health priority,class
priority
TID6  : parents usual,social slightly_problem,health
not_recom,class not_recom
TID7  : parents usual,social problematic,health recommended,class
priority
TID8  : parents usual,social problematic,health priority,class
priority
TID9  : parents usual,social problematic,health not_recom,class
not_recom
TID10 : parents usual,social nonproblem,health not_recom,class
not_recom
TID11 : parents usual,social nonproblem,health priority,class
priority
TID12 : parents usual,social nonproblem,health recommended,class
very_recom
```

รูปที่ 3.31 ข้อมูลที่มีลักษณะของข้อมูลแตกต่างกันมาก หรือมีข้อมูลเหมือนกันเกิดขึ้นซ้ำกันไม่มาก

รายละเอียดข้อมูลแต่ละชุด มีรายละเอียดดังต่อไปนี้

1) ข้อมูล Census Income เป็นข้อมูลของประชากรและรายได้ของประชากรในมลรัฐเคนซัส มีจำนวนข้อมูล 131,072 เรคคอร์ด

2) ข้อมูล Mushroom เป็นข้อมูลลักษณะของเห็ดมีพิษและเห็ดที่สามารถนำมาใช้ประกอบอาหารรับประทานได้ มีจำนวนข้อมูล 100,992 เรคคอร์ด
3) ข้อมูล Soybean เป็นข้อมูลของลักษณะต้นถั่ว มีจำนวนข้อมูล 105,865 เรคคอร์ด
4) ข้อมูล Convenient store เป็นข้อมูลเกี่ยวกับการซื้อสินค้าในร้านสะดวกซื้อ มีจำนวนข้อมูล 60,000 เรคคอร์ด
5) ข้อมูล Adult เป็นข้อมูลเกี่ยวกับประชากรที่ถือว่าเป็นผู้ใหญ่ในมลรัฐเคนซัส มีจำนวนข้อมูล 120,000 เรคคอร์ด
6) ข้อมูล Thyroid เป็นข้อมูลเกี่ยวกับอาการป่วยของคนไข้โรคไทรอยด์ มีจำนวนข้อมูล 73,344 เรคคอร์ด
7) ข้อมูล Nursery เป็นข้อมูลเกี่ยวกับโรงเรียนสำหรับเด็กเล็กก่อนขึ้นชั้นอนุบาล (Nursery school) มีจำนวนข้อมูล 90,721 เรคคอร์ด

3.5 วิธีการทดสอบและเปรียบเทียบผลลัพธ์ที่ได้

การทดสอบจะทำโดยการเรียกใช้โปรแกรม SASS ที่พัฒนาขึ้น และจะต้องมีการส่งค่าพารามิเตอร์เข้าไปให้ถูกต้อง โดยโปรแกรมสามารถทำงานได้กับข้อมูลที่ถูกเก็บอยู่ในฐานข้อมูลเท่านั้น ชุดข้อมูลทั้งหมดที่ใช้ในการทดสอบ และเปรียบเทียบผลลัพธ์จะต้องถูกเปลี่ยนจากรูปแบบเท็กซ์ไฟล์ (Text file) ให้จัดเก็บอยู่ภายในฐานข้อมูล จากนั้นทำการค้นหากฎความสัมพันธ์จากข้อมูลโดยการเรียกใช้โปรแกรมและส่งค่าพารามิเตอร์ให้ถูกต้องตามรูปแบบดังนี้

```
begin
   apriori (p_hcol, p_htab, p_dslcol,  p_dtab, p_dkcol, p_min_sup,
            p_min_conf, p_pc_rand);
end;
```

รูปที่ 3.32 รูปแบบการเรียกใช้อัลกอริทึม SASS

รายละเอียดของพารามิเตอร์ที่ต้องส่งเข้าไปเมื่อทำการเรียกใช้โพรซิเยอร์ apriori ในการค้นหากฎความสัมพันธ์ มีดังนี้

p_hcol เป็นชื่อคอลัมน์ที่เป็นคีย์ (key) ของตารางทรานแซคชัน ประเภทของข้อมูลเป็นตัวอักษร (character)

p_htab เป็นชื่อของตารางทรานแซคชัน ประเภทของข้อมูลเป็นตัวอักษร

p_dslcol เป็นชื่อคอลัมน์ที่จะนำมาสร้างเป็นไอเท็มเซต อยู่ภายในตารางทรานแซคชันที่เป็นรายละเอียด (detail) ประเภทของข้อมูลเป็นตัวอักษร

p_dtab เป็นชื่อของตารางทรานแซคชันที่เป็นรายละเอียด ประเภทของข้อมูลเป็นตัวอักษร

p_dkcol เป็นชื่อคอลัมน์ที่เป็นคีย์ของตารางทรานแซคชันที่เป็นรายละเอียด รายละเอียด ประเภทของข้อมูลเป็นตัวอักษร

p_min_sup ค่าสนับสนุนขั้นต่ำ ประเภทของข้อมูลเป็นตัวเลข ตั้งแต่ 1-100

p_min_conf ค่าความเชื่อมั่นขั้นต่ำ ประเภทของข้อมูลเป็นตัวเลข ตั้งแต่ 1-100

p_pc_rand ขนาดของข้อมูลที่ต้องการสุ่ม ประเภทของข้อมูลเป็นตัวเลข ตั้งแต่ 1-100

ในงานวิจัยนี้ต้องการทดสอบประสิทธิภาพการทำงานของโปรแกรม SASS ที่มีต่อข้อมูลและข้อมูลสุ่มที่มีขนาดต่างๆ กันโดยวิเคราะห์จากค่าต่างๆ ดังนี้

1) เวลาที่ใช้ในการค้นหากฎความสัมพันธ์จากข้อมูลขนาดต่างๆ วัดในหน่วยของวินาที
2) ปริมาณกฎความสัมพันธ์ที่ค้นพบจากข้อมูลขนาดต่างๆ กัน
3) ตำแหน่งของกฎความสัมพันธ์ที่ปรากฏจากการค้นหากฎความสัมพันธ์จากข้อมูลที่มีขนาดต่างๆ กัน

บทที่ 4
การทดสอบและอภิปรายผล

การค้นหากฎความสัมพันธ์ที่แข็งแกร่ง ใช้โปรแกรม SASS ที่ทำการพัฒนาโดยปรับปรุงจากอัลกอริทึม เอไพรออริ โดยพัฒนาขึ้นด้วย SQL โดยการทดสอบจะทดสอบด้วยโปรแกรมสำเร็จรูปซึ่งเป็นโปรแกรมที่ผู้วิจัยทำการพัฒนาขึ้นด้วย Oracle Developer 6i โดยมีส่วนติดต่อกับผู้ใช้งานในรูปแบบกราฟิก (Graphic User Interface : GUI) เพื่อให้ผู้ใช้งานสามารถใช้งานได้สะดวกขึ้น และทดสอบด้วยการเรียกใช้โปรแกรมโดยตรงโดยพิมพ์คำสั่งเรียกใช้โพรซิเยอร์ apriori และทำการระบุค่าพารามิเตอร์ตามที่ได้อธิบายไว้ในข้อ 3.5 หน้าที่ 62 การทดสอบการค้นหากฎความสัมพันธ์ทำการทดสอบกับข้อมูลสองแบบคือ ข้อมูลที่มีลักษณะของข้อมูลที่มีความตรงกันมาก หรือมีข้อมูลเหมือนกันเกิดขึ้นซ้ำมากๆ หรือมีลักษณะเป็นข้อมูลเชิงธุรกิจ และข้อมูลที่มีลักษณะของข้อมูลแตกต่างกันมาก หรือมีข้อมูลเหมือนกันเกิดขึ้นซ้ำกันไม่มาก หรืออาจไม่มีการซ้ำกันเลย การทดสอบทำการประมวลผลบนเครื่องคอมพิวเตอร์ Laptop Pentium IV ความเร็ว 1.4 GHz หน่วยความจำหลัก 768 MB ฮาร์ดดิสก์ความจุ 40 GB โดยผลการทดสอบการทำงานของโปรแกรม SASS กับข้อมูลที่มีลักษณะของข้อมูลที่มีความตรงกันมาก หรือมีข้อมูลเหมือนกันเกิดขึ้นซ้ำมากๆ ปรากฏรายละเอียดในหัวข้อ 4.1 ผลการทดสอบการทำงานของโปรแกรม SASS กับข้อมูลที่มีลักษณะของข้อมูลที่มีลักษณะของข้อมูลแตกต่างกันมาก หรือมีข้อมูลเหมือนกันเกิดขึ้นซ้ำกันไม่มาก ปรากฏรายละเอียดในหัวข้อ 4.2 ผลการทดสอบการทำงานของโปรแกรม SASS กับข้อมูลสตรีมมิงปรากฏรายละเอียดในหัวข้อ 4.3 และหัวข้อ 4.4 เป็นการอภิปรายสรุป

4.1 ผลการทดสอบกับข้อมูลที่มีลักษณะของข้อมูลเหมือนกันเกิดขึ้นซ้ำมากๆ

ตารางที่ 4.1, 4.4, 4.7, 4.10 และ 4.13 แสดงผลการค้นหากฎความสัมพันธ์โดยการเรียกใช้โปรแกรม SASS ที่ทำการพัฒนาขึ้นโดยการเรียกใช้โดยตรง การทดสอบทำโดยการทดสอบด้วยเทคนิคการสุ่มตัวอย่างแบบต่างๆ ทั้ง 5 แบบ จำนวนกฎความสัมพันธ์ที่ค้นพบที่มีความถูกต้องตรงกันปรากฏในตารางที่ 4.2, 4.5, 4.8, 4.11 และ 4.14 และเวลาที่ใช้ในการค้นหากฎความสัมพันธ์ของอัลกอริทึมปรากฏในตารางที่ 4.3, 4.6, 4.9, 4.12 และ 4.15 การทดสอบการค้นหากฎความสัมพันธ์โดยใช้เทคนิคการสุ่มตัวอย่างแบบต่างๆ มีการกำหนดค่าสนับสนุนขั้นต่ำที่ 80 เปอร์เซ็นต์ และกำหนดค่าความเชื่อมั่นขั้นต่ำที่ 95 เปอร์เซ็นต์ ทำการทดสอบกับข้อมูลที่มี

ลักษณะของข้อมูลที่มีความตรงกันมาก หรือมีข้อมูลเหมือนกันเกิดขึ้นซ้ำมากๆ หรือมีลักษณะเป็นข้อมูลเชิงธุรกิจ โดยทำการทดสอบกับข้อมูลทั้งหมด (100 เปอร์เซ็นต์) และข้อมูลสุ่มที่มีขนาดต่างๆ กันคือ 1, 10, 20, 30, 40 และ 50 เปอร์เซ็นต์

ตารางที่ 4.1 แสดงจำนวนกฎความสัมพันธ์ทั้งหมดที่ค้นพบจากข้อมูลทั้งหมดเปรียบเทียบกับชุดข้อมูลสุ่มที่มีขนาดต่างๆ กัน ทดสอบโดยใช้เทคนิคการสุ่มตัวอย่างแบบง่ายโดยใช้หลักการสุ่มแบบใส่คืนที่ โดยตัวเลขที่ปรากฏในตารางคือจำนวนกฎความสัมพันธ์ที่ค้นพบ (เฉลี่ยจากผลการทดสอบจำนวน 5 ครั้ง)

ชื่อชุดข้อมูล	ข้อมูลทั้งหมด (100%)	ข้อมูลสุ่ม					
		50%	40%	30%	20%	10%	1%
Convenient store	2	2	2	2	2	2	2
Census Income	6	6	6	6	6	6	7
Mushroom	32	32	32	32	32	32	32
Soybean	10	11	9	10	9	11	11

ตารางที่ 4.2 แสดงจำนวนกฎความสัมพันธ์ที่มีความถูกต้องตรงกัน ทดสอบโดยใช้เทคนิคการสุ่มตัวอย่างแบบง่ายโดยใช้หลักการสุ่มแบบใส่คืนที่ (เฉลี่ยจากผลการทดสอบจำนวน 5 ครั้ง)

ชื่อชุดข้อมูล	ข้อมูลทั้งหมด (100%)	ข้อมูลสุ่ม					
		50%	40%	30%	20%	10%	1%
Convenient store	2	2	2	2	2	2	2
Census Income	6	6	6	6	6	6	6
Mushroom	32	32	32	32	32	32	32
Soybean	10	10	9	10	9	10	10

ตารางที่ 4.3 แสดงเวลา (วินาที) ที่ใช้ในการค้นหากฎความสัมพันธ์ของข้อมูลแต่ละชุด ทดสอบโดยใช้เทคนิคการสุ่มตัวอย่างแบบง่ายโดยใช้หลักการสุ่มแบบใส่คืนที่ (เฉลี่ยจากผลการทดสอบจำนวน 5 ครั้ง)

ชื่อชุดข้อมูล	ข้อมูลทั้งหมด (100%)	ข้อมูลสุ่ม					
		50%	40%	30%	20%	10%	1%
Convenient store	1868	926	771	552	353	201	23
Census Income	8078	4587	3606	2785	1875	986	117
Mushroom	5891	3378	2687	1983	1270	654	51
Soybean	9799	5339	4399	3596	2359	1285	107

ตารางที่ 4.4 แสดงจำนวนกฎความสัมพันธ์ทั้งหมดที่ค้นพบจากข้อมูลทั้งหมดเปรียบเทียบกับชุดข้อมูลสุ่มที่มีขนาดต่างๆ กัน ทดสอบโดยใช้เทคนิคการสุ่มตัวอย่างแบบง่ายโดยใช้หลักการสุ่มแบบไม่ใส่คืนที่ โดยตัวเลขที่ปรากฏในตารางคือจำนวนกฎความสัมพันธ์ที่ค้นพบ (เฉลี่ยจากผลการทดสอบจำนวน 5 ครั้ง)

ชื่อชุดข้อมูล	ข้อมูลทั้งหมด (100%)	ข้อมูลสุ่ม					
		50%	40%	30%	20%	10%	1%
Convenient store	2	2	2	2	2	2	2
Census Income	6	6	6	6	6	6	6
Mushroom	32	32	32	32	32	32	32
Soybean	10	11	11	9	10	9	12

ตารางที่ 4.5 แสดงจำนวนกฎความสัมพันธ์ที่มีความถูกต้องตรงกัน ทดสอบโดยใช้เทคนิคการสุ่มตัวอย่างแบบง่ายโดยใช้หลักการสุ่มแบบไม่ใส่คืนที่ (เฉลี่ยจากผลการทดสอบจำนวน 5 ครั้ง)

ชื่อชุดข้อมูล	ข้อมูลทั้งหมด (100%)	ข้อมูลสุ่ม					
		50%	40%	30%	20%	10%	1%
Convenient store	2	2	2	2	2	2	2
Census Income	6	6	6	6	6	6	6
Mushroom	32	32	32	32	32	32	32
Soybean	10	10	10	9	10	9	10

ตารางที่ 4.6 แสดงเวลา (วินาที) ที่ใช้ในการค้นหากฎความสัมพันธ์ของข้อมูลแต่ละชุด ทดสอบโดยใช้เทคนิคการสุ่มตัวอย่างแบบง่ายโดยใช้หลักการสุ่มแบบไม่ใส่คืนที่ (เฉลี่ยจากผลการทดสอบจำนวน 5 ครั้ง)

ชื่อชุดข้อมูล	ข้อมูลทั้งหมด (100%)	ข้อมูลสุ่ม					
		50%	40%	30%	20%	10%	1%
Convenient store	1957	1075	803	598	392	237	22
Census Income	8127	4609	3808	2821	1923	1012	124
Mushroom	5927	3259	2589	2023	1299	697	57
Soybean	9863	5410	4429	3630	2405	1307	97

ตารางที่ 4.7 แสดงจำนวนกฎความสัมพันธ์ทั้งหมดที่ค้นพบจากข้อมูลทั้งหมดเปรียบเทียบกับชุดข้อมูลสุ่มที่มีขนาดต่างๆ กัน ทดสอบโดยใช้เทคนิคการสุ่มตัวอย่างแบบมีระบบโดยใช้หลักการสุ่มแบบใส่คืนที่ โดยตัวเลขที่ปรากฏในตารางคือจำนวนกฎความสัมพันธ์ที่ค้นพบ (เฉลี่ยจากผลการทดสอบจำนวน 5 ครั้ง)

ชื่อชุดข้อมูล	ข้อมูลทั้งหมด (100%)	ข้อมูลสุ่ม					
		50%	40%	30%	20%	10%	1%
Convenient store	2	2	2	2	2	2	2
Census Income	6	6	6	6	6	6	6
Mushroom	32	32	32	32	32	32	32
Soybean	10	10	10	10	10	8	10

ตารางที่ 4.8 แสดงจำนวนกฎความสัมพันธ์ที่มีความถูกต้องตรงกัน ทดสอบโดยใช้เทคนิคการสุ่มตัวอย่างแบบมีระบบโดยใช้หลักการสุ่มแบบใส่คืนที่ (เฉลี่ยจากผลการทดสอบจำนวน 5 ครั้ง)

ชื่อชุดข้อมูล	ข้อมูลทั้งหมด (100%)	ข้อมูลสุ่ม					
		50%	40%	30%	20%	10%	1%
Convenient store	2	2	2	2	2	2	2
Census Income	6	6	6	6	6	6	7
Mushroom	32	32	32	32	32	32	32
Soybean	10	10	10	10	10	8	10

ตารางที่ 4.9 แสดงเวลา (วินาที) ที่ใช้ในการค้นหากฎความสัมพันธ์ของข้อมูลแต่ละชุด ทดสอบโดยใช้เทคนิคการสุ่มตัวอย่างแบบมีระบบโดยใช้หลักการสุ่มแบบใส่คืนที่ (เฉลี่ยจากผลการทดสอบจำนวน 5 ครั้ง)

ชื่อชุดข้อมูล	ข้อมูลทั้งหมด (100%)	ข้อมูลสุ่ม					
		50%	40%	30%	20%	10%	1%
Convenient store	1897	987	817	571	427	271	21
Census Income	8249	4563	3674	2788	1945	995	127
Mushroom	6048	3321	2709	2058	1325	705	50
Soybean	9948	5398	4525	3499	2377	1252	93

ตารางที่ 4.10 แสดงจำนวนกฎความสัมพันธ์ทั้งหมดที่ค้นพบจากข้อมูลทั้งหมดเปรียบเทียบกับชุดข้อมูลสุ่มที่มีขนาดต่างๆ กัน ทดสอบโดยใช้เทคนิคการสุ่มตัวอย่างแบบมีระบบโดยใช้หลักการสุ่มแบบไม่ใส่คืนที่ โดยตัวเลขที่ปรากฏในตารางคือจำนวนกฎความสัมพันธ์ที่ค้นพบ (เฉลี่ยจากผลการทดสอบจำนวน 5 ครั้ง)

ชื่อชุดข้อมูล	ข้อมูลทั้งหมด (100%)	ข้อมูลสุ่ม					
		50%	40%	30%	20%	10%	1%
Convenient store	2	2	2	2	2	2	2
Census Income	6	6	6	6	6	6	6
Mushroom	32	32	32	32	32	32	32
Soybean	10	9	10	10	9	10	11

ตารางที่ 4.11 แสดงจำนวนกฎความสัมพันธ์ที่มีความถูกต้องตรงกัน ทดสอบโดยใช้เทคนิคการสุ่มตัวอย่างแบบมีระบบโดยใช้หลักการสุ่มแบบไม่ใส่คืนที่ (เฉลี่ยจากผลการทดสอบจำนวน 5 ครั้ง)

ชื่อชุดข้อมูล	ข้อมูลทั้งหมด (100%)	ข้อมูลสุ่ม					
		50%	40%	30%	20%	10%	1%
Convenient store	2	2	2	2	2	2	2
Census Income	6	6	6	6	6	6	6
Mushroom	32	32	32	32	32	32	32
Soybean	10	9	10	10	9	10	10

ตารางที่ 4.12 แสดงเวลา (วินาที) ที่ใช้ในการค้นหากฎความสัมพันธ์ของข้อมูลแต่ละชุด ทดสอบโดยใช้เทคนิคการสุ่มตัวอย่างแบบมีระบบโดยใช้หลักการสุ่มแบบไม่ใส่คืนที่ (เฉลี่ยจากผลการทดสอบจำนวน 5 ครั้ง)

ชื่อชุดข้อมูล	ข้อมูลทั้งหมด (100%)	ข้อมูลสุ่ม					
		50%	40%	30%	20%	10%	1%
Convenient store	2055	1005	877	607	458	296	24
Census Income	8357	4697	3708	2912	1908	1017	129
Mushroom	6177	3417	2807	2099	1401	720	52
Soybean	9997	5417	4498	3556	2433	1289	91

ตารางที่ 4.13 แสดงจำนวนกฎความสัมพันธ์ทั้งหมดที่ค้นพบจากข้อมูลทั้งหมด เปรียบเทียบกับชุดข้อมูลสุ่มที่มีขนาดต่างๆ กัน ทดสอบโดยใช้เทคนิคการสุ่มตัวอย่างแบบง่ายโดยใช้ความน่าจะเป็นและใช้หลักการสุ่มแบบไม่ใส่คืนที่ โดยตัวเลขที่ปรากฏในตารางคือจำนวนกฎความสัมพันธ์ที่ค้นพบ (เฉลี่ยจากผลการทดสอบจำนวน 5 ครั้ง)

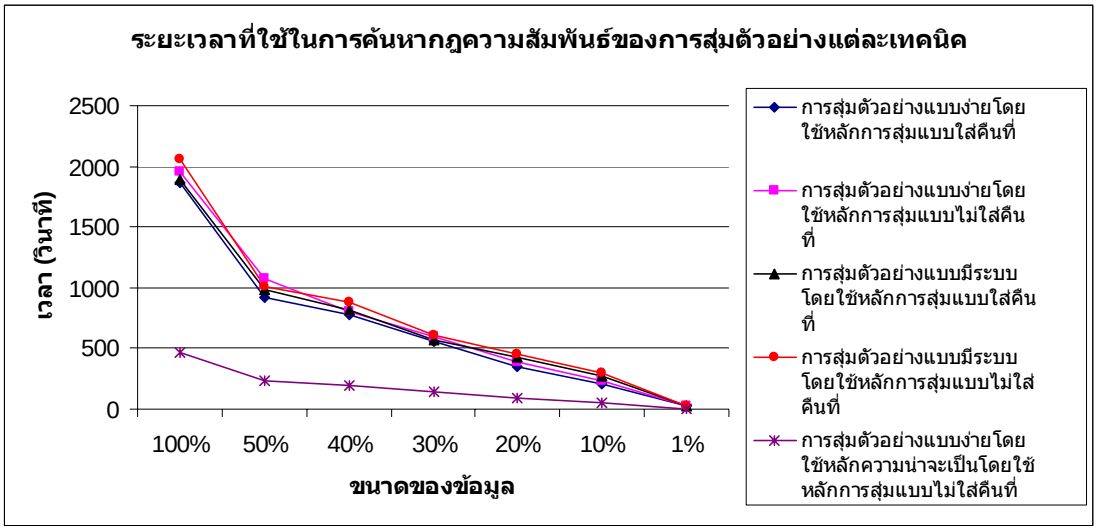
ชื่อชุดข้อมูล	ข้อมูลทั้งหมด (100%)	ข้อมูลสุ่ม					
		50%	40%	30%	20%	10%	1%
Convenient store	2	2	2	2	2	2	2
Census Income	6	6	6	6	6	6	6
Mushroom	32	32	32	32	32	32	32
Soybean	10	11	9	10	9	7	11

ตารางที่ 4.14 แสดงจำนวนกฎความสัมพันธ์ที่มีความถูกต้องตรงกัน ทดสอบโดยใช้เทคนิคการสุ่มตัวอย่างแบบง่ายโดยใช้ความน่าจะเป็นและใช้หลักการสุ่มแบบไม่ใส่คืนที่ (เฉลี่ยจากผลการทดสอบจำนวน 5 ครั้ง)

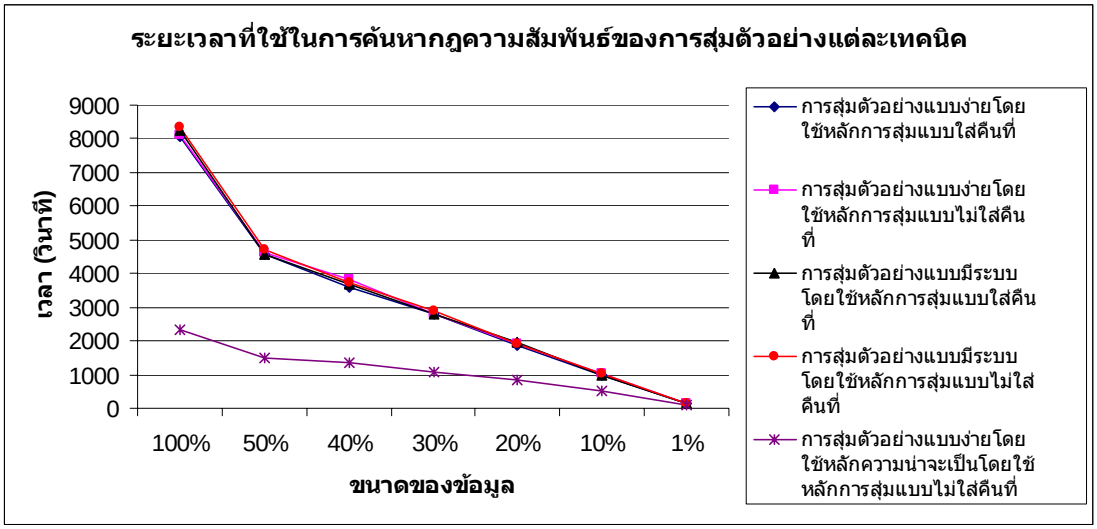
ชื่อชุดข้อมูล	ข้อมูลทั้งหมด (100%)	ข้อมูลสุ่ม					
		50%	40%	30%	20%	10%	1%
Convenient store	2	2	2	2	2	2	2
Census Income	6	6	6	6	6	6	6
Mushroom	32	32	32	32	32	32	32
Soybean	10	10	9	10	9	7	10

ตารางที่ 4.15 แสดงเวลา (วินาที) ที่ใช้ในการค้นหากฎความสัมพันธ์ของข้อมูลแต่ละชุด ทดสอบโดยใช้เทคนิคการสุ่มตัวอย่างแบบง่ายโดยใช้ความน่าจะเป็นและใช้หลักการสุ่มแบบไม่ใส่คืนที่ (เฉลี่ยจากผลการทดสอบจำนวน 5 ครั้ง)

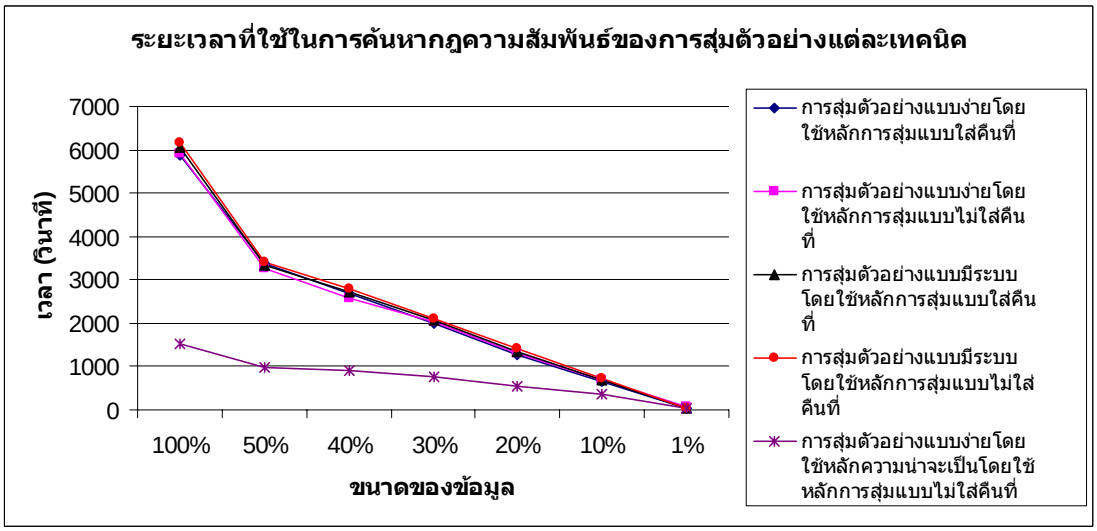
ชื่อชุดข้อมูล	ข้อมูลทั้งหมด (100%)	ข้อมูลสุ่ม					
		50%	40%	30%	20%	10%	1%
Convenient store	465	234	190	146	89	49	4
Census Income	2343	1492	1331	1050	838	503	76
Mushroom	1528	977	902	758	555	358	22
Soybean	3709	2194	1884	1542	1189	638	59



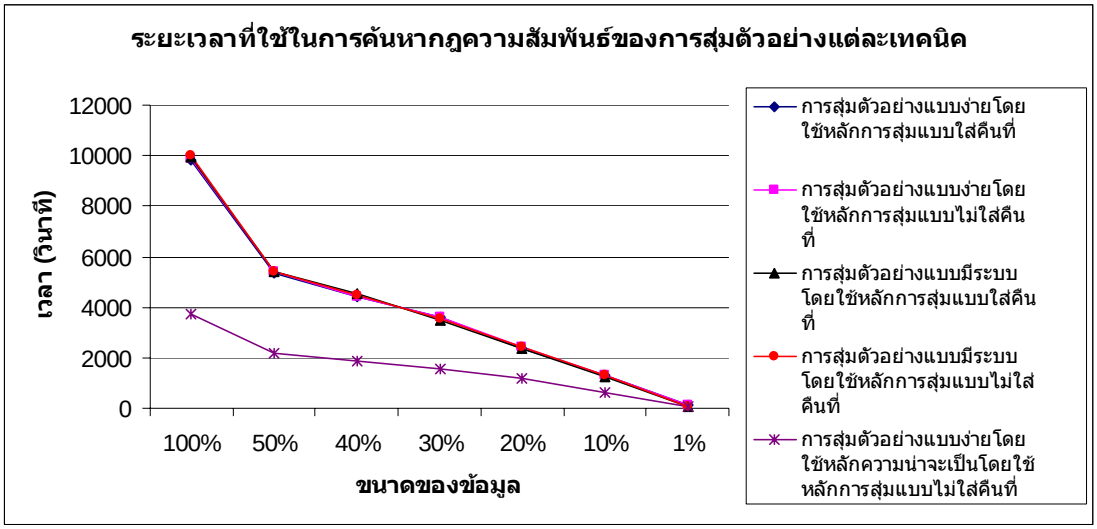
รูปที่ 4.1 กราฟเปรียบเทียบระยะเวลาในการค้นหาความสัมพันธ์จากชุดข้อมูล Convenient store ของเทคนิคการสุ่มตัวอย่างแต่ละเทคนิค



รูปที่ 4.2 กราฟเปรียบเทียบระยะเวลาในการค้นหาความสัมพันธ์จากชุดข้อมูล Census Income ของเทคนิคการสุ่มตัวอย่างแต่ละเทคนิค



รูปที่ 4.3 กราฟเปรียบเทียบระยะเวลาในการค้นหาความสัมพันธ์จากชุดข้อมูล Mushroom ของเทคนิคการสุ่มตัวอย่างแต่ละเทคนิค



รูปที่ 4.4 กราฟเปรียบเทียบระยะเวลาในการค้นหาความสัมพันธ์จากชุดข้อมูล Soybean ของเทคนิคการสุ่มตัวอย่างแต่ละเทคนิค

ในกรณีที่การค้นหากฎความสัมพันธ์โดยการเรียกใช้โปรแกรม SASS โดยตรง เมื่อโปรแกรมสิ้นสุดการทำงานจะแสดงกฎความสัมพันธ์ที่ค้นพบออกมา โดยไม่ได้ทำการเรียงลำดับของกฎความสัมพันธ์ที่ค้นพบตามค่าความเชื่อมั่น (การเรียงลำดับจะเรียงจากกฎความสัมพันธ์ที่มีค่าความเชื่อมั่นสูงสุดไปหาต่ำสุด และเรียงลำดับจากค่าสนับสนุนสูงสุดไปหาต่ำสุด) ดังนั้นเมื่อทำการเปรียบเทียบกฎความสัมพันธ์จะต้องจัดเรียงกฎความสัมพันธ์ที่ถูกค้นพบ ก่อนที่จะทำการเปรียบ-เทียบความถูกต้องของกฎความสัมพันธ์ดังแสดงในรูปที่ 4.5 และ 4.6 เมื่อจัดเรียงกฎความสัมพันธ์เรียบร้อยแล้วจึงทำการเปรียบเทียบกฎความสัมพันธ์ทั้งสองชุด โดยเอากฎความสัมพันธ์ที่ค้นพบจากข้อมูลทั้งหมดเป็นเกณฑ์ในการเปรียบเทียบความถูกต้องของกฎความสัมพันธ์ที่ถูกค้นพบจากข้อมูลสุ่ม

```
1 : birth United_States = 116306 -> citizenship
    Native_Born_in_the_United_States = 116306 conf : 1
2 : citizenship Native_Born_in_the_United_States = 116309 -> birth
    United_States = 116306 conf : 1
3 : birth United_States,edu_inst Not_in_universe = 108876 ->
    citizenship Native_Born_in_the_United_States = 108876 conf : 1
4 : citizenship Native_Born_in_the_United_States,edu_inst
    Not_in_universe = 108878 -> birth United_States = 108876 conf :
    1
5 : birth United_States,salary -50000 = 108997 -> citizenship
    Native_Born_in_the_United_States = 108997 conf : 1
6 : citizenship Native_Born_in_the_United_States,salary -50000 =
    109000 -> birth United_States = 108997 conf : 1
```

รูปที่ 4.5 แสดงกฎความสัมพันธ์ที่ค้นพบจากข้อมูลทั้งหมด (100 เปอร์เซ็นต์) ของชุดข้อมูล Census Income

```
1 : birth United_States = 1169 -> citizenship
    Native_Born_in_the_United_States = 1169 conf : 1
2 : citizenship Native_Born_in_the_United_States = 1169 -> birth
    United_States = 1169 conf : 1
3 : birth United_States = 1169 -> edu_inst Not_in_universe = 1105
    conf : .95
4 : citizenship Native_Born_in_the_United_States = 1169 -> edu_inst
    Not_in_universe = 1105 conf : .95
5 : birth United_States = 1169 -> citizenship Native_Born_in_
    the_United_States, edu_inst Not_in_universe = 1105 conf : .95
6 : birth United_States,citizenship
    Native_Born_in_the_United_States = 1169 -> edu_inst
    Not_in_universe = 1105 conf : .95
7 : birth United_States,edu_inst Not_in_universe = 1105 ->
    citizenship Native_Born_in_the_United_States = 1105 conf : 1
8 : citizenship Native_Born_in_the_United_States,edu_inst
    Not_in_universe = 1105 -> birth United_States = 1105 conf : 1
9 : birth United_States,salary -50000 = 1099 -> citizenship
    Native_Born_in_the_United_States = 1099 conf : 1
10 : citizenship Native_Born_in_the_United_States,salary -50000 =
    1099 -> birth United_States = 1099 conf : 1
```

รูปที่ 4.6 แสดงกฎความสัมพันธ์ที่ค้นพบจากข้อมูลสุ่มขนาด 1 เปอร์เซ็นต์ ของชุดข้อมูล Census Income

```
1 : birth United_States = 116306 -> citizenship
    Native_Born_in_the_United_States = 116306 conf : 1
2 : citizenship Native_Born_in_the_United_States = 116309 -> birth
    United_States = 116306 conf : 1
3 : citizenship Native_Born_in_the_United_States,salary -50000 =
    109000 -> birth United_States = 108997 conf : 1
4 : birth United_States,salary -50000 = 108997 -> citizenship
    Native_Born_in_the_United_States = 108997 conf : 1
5 : citizenship Native_Born_in_the_United_States,edu_inst
    Not_in_universe = 108878 -> birth United_States = 108876 conf :
    1
6 : birth United_States,edu_inst Not_in_universe = 108876 ->
    citizenship Native_Born_in_the_United_States = 108876 conf : 1
```

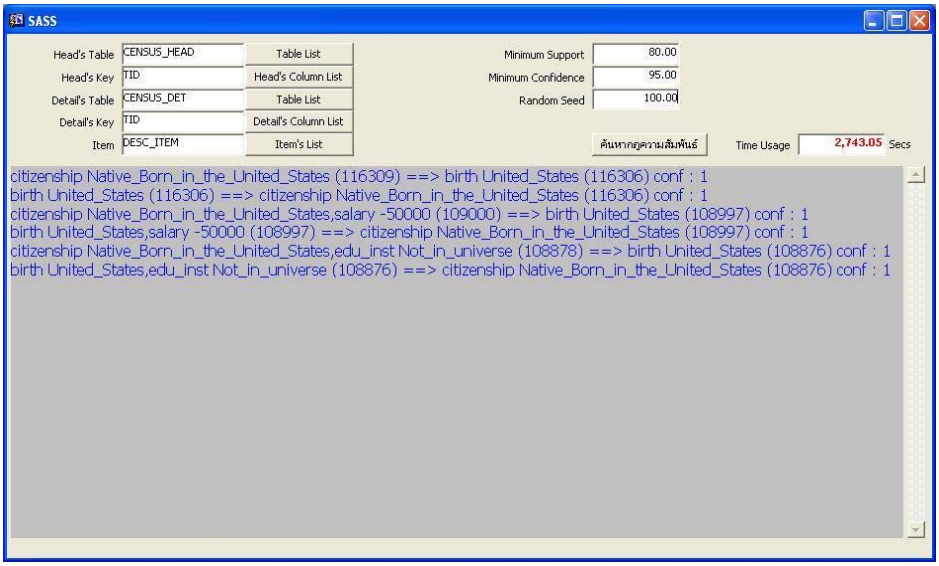
รูปที่ 4.7 แสดงการจัดเรียงกฎความสัมพันธ์ที่ถูกค้นพบจากข้อมูลทั้งหมด (100 เปอร์เซ็นต์) ของชุดข้อมูล Census Income เพื่อใช้เป็นเกณฑ์ในการเปรียบเทียบความถูกต้องของกฎความสัมพันธ์

```
1 :  birth United_States = 1169 -> citizenship
     Native_Born_in_the_United_States = 1169 conf : 1
2 :  citizenship Native_Born_in_the_United_States = 1169 -> birth
     United_States = 1169 conf : 1
3 :  birth United_States,edu_inst Not_in_universe = 1105 ->
     citizenship Native_Born_in_the_United_States = 1105 conf : 1
4 :  citizenship Native_Born_in_the_United_States,edu_inst
     Not_in_universe = 1105 -> birth United_States = 1105 conf : 1
5 :  birth United_States,salary -50000 = 1099 -> citizenship
     Native_Born_in_the_United_States = 1099 conf : 1
6 :  citizenship Native_Born_in_the_United_States,salary -50000 =
     1099 -> birth United_States = 1099 conf : 1
7 :  birth United_States = 1169 -> edu_inst Not_in_universe = 1105
     conf : .95
8 :  citizenship Native_Born_in_the_United_States = 1169 ->
     edu_inst Not_in_universe = 1105 conf : .95
9 :  birth United_States = 1169 -> citizenship
     Native_Born_in_the_United_States,edu_inst Not_in_universe =
     1105 conf : .95
10 : birth United_States,citizenship
     Native_Born_in_the_United_States = 1169 -> edu_inst
     Not_in_universe = 1105 conf : .95
```

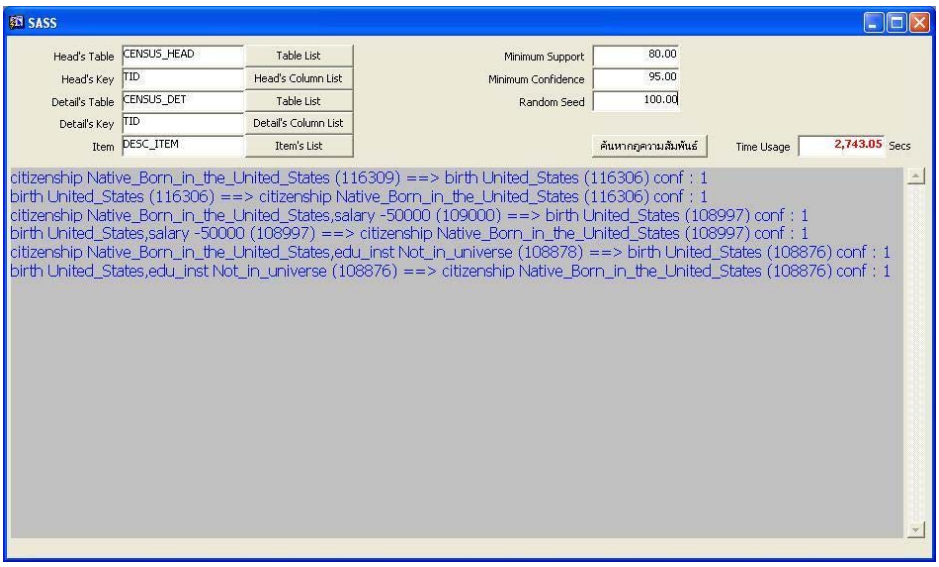
รูปที่ 4.8 แสดงการจัดเรียงกฎความสัมพันธ์ที่ถูกค้นพบจากข้อมูลสุ่มขนาด 1 เปอร์เซ็นต์ของชุดข้อมูล Census Income เพื่อใช้ในการเปรียบเทียบความถูกต้องของกฎความสัมพันธ์

จากรูปที่ 4.7 และ 4.8 จะแสดงกฎความสัมพันธ์ที่ถูกค้นพบจากชุดข้อมูลทั้งสองชุดหลังจากเรียงลำดับแล้ว จะเห็นว่ากฎความสัมพันธ์ที่ค้นพบจากชุดข้อมูลทั้งหมดมีจำนวน 6 กฎ และกฎความสัมพันธ์ที่ค้นพบจากข้อมูลสุ่มขนาด 1 เปอร์เซ็นต์มีจำนวน 11 กฎ เมื่อจัดเรียงลำดับและทำการเปรียบเทียบกับกฎความสัมพันธ์จากชุดข้อมูลทั้งหมดแล้วจะมีความถูกต้องตรงกันกับกฎความสัมพันธ์ที่ค้นพบจากชุดข้อมูลทั้งหมดเป็นจำนวน 6 กฎ แต่ถ้าใช้โปรแกรม SASS ในรูปแบบโปรแกรมสำเร็จรูปในการค้นหากฎความสัมพันธ์ เมื่อสิ้นสุดการทำงานของอัลกอริทึมโปรแกรมจะแสดงกฎความสัมพันธ์ที่ค้นพบ โดยจะจัดเรียงกฎความสัมพันธ์ตามค่าความเชื่อมั่น ดังรูปที่ 4.7 แสดงกฎ

ความสัมพันธ์ที่ค้นพบจากชุดข้อมูลทั้งหมดของข้อมูล Census Income และรูปที่ 4.8 แสดงกฎความสัมพันธ์ที่ค้นพบได้จากชุดข้อมูลสุ่มขนาด 1 เปอร์เซ็นต์ของข้อมูล Census Income



รูปที่ 4.9 แสดงการค้นหากฎความสัมพันธ์จากข้อมูลทั้งหมด (100 เปอร์เซ็นต์) ของข้อมูล Census Income โดยใช้โปรแกรมสำเร็จรูป



รูปที่ 4.10 แสดงกฎความสัมพันธ์จากข้อมูลสุ่มขนาด 1 เปอร์เซ็นต์ของข้อมูล Census Income โดยใช้โปรแกรมสำเร็จรูป

4.2 ผลการทดสอบกับข้อมูลที่มีลักษณะของข้อมูลเหมือนกันเกิดขึ้นซ้ำไม่มาก

การทดสอบการค้นหากฎความสัมพันธ์กับข้อมูลที่มีลักษณะของข้อมูลที่มีลักษณะของข้อมูลแตกต่างกันมาก หรือมีข้อมูลเหมือนกันเกิดขึ้นซ้ำกันไม่มาก โดยการทดสอบกำหนดค่าสนับสนุนขั้นต่ำเริ่มต้นที่ 80 เปอร์เซ็นต์ และกำหนดค่าความเชื่อมั่นขั้นต่ำเริ่มต้นที่ 95 เปอร์เซ็นต์ เนื่องจากข้อมูลมีข้อมูลที่เหมือนกันเกิดขึ้นซ้ำกันไม่มากจึงมีความน่าจะเป็นที่ทำการทดสอบ โดยกำหนดค่าสนับสนุนขั้นต่ำ และค่าความเชื่อมั่นขั้นต่ำดังกล่าวแล้วอาจทำให้อัลกอริทึมไม่สามารถค้นหากฎความสัมพันธ์ได้ ดังแสดงในรูปที่ 4.9 ดังนั้นเมื่อทำการทดสอบแล้วหากอัลกอริทึมไม่สามารถค้นหากฎความสัมพันธ์จากข้อมูลได้ จะทำการลดค่าสนับสนุนขั้นต่ำ หรือค่าความเชื่อมั่นขั้นต่ำลง จนกว่าจะถึงระดับที่โปรแกรมสามารถค้นหากฎความสัมพันธ์จากข้อมูลได้จึงจะทำการทดสอบกับข้อมูลทั้งหมดและข้อมูลสุ่มที่มีขนาดต่างๆ กันคือ 1, 10, 20, 30, 40 และ 50 เปอร์เซ็นต์

```
กำหนดค่าสนับสนุนขั้นต่ำ 80 เปอร์เซ็นต์ และกำหนดค่าความเชื่อมั่นขั้นต่ำ 95 เปอร์เซ็นต์

No Association
```

รูปที่ 4.11 แสดงการค้นหากฎความสัมพันธ์จากข้อมูล Nursery เป็นข้อมูลที่มีลักษณะของข้อมูลเหมือนกันเกิดขึ้นซ้ำกันไม่มาก

จากรูปที่ 4.11 จะแสดงผลที่ได้จากการทดสอบการค้นหากฎความสัมพันธ์จากข้อมูลที่มีรูปแบบของข้อมูลแตกต่างกันมาก เมื่อกำหนดค่าสนับสนุนขั้นต่ำ และค่าความเชื่อมั่นขั้นต่ำในเกณฑ์ที่สูงจะทำให้โปรแกรมไม่สามารถค้นหากฎความสัมพันธ์ได้ เนื่องจากข้อมูลมีค่าสนับสนุนไม่เพียงพอ หรือกฎความสัมพันธ์ที่ได้มีค่าความเชื่อมั่นไม่เพียงพอ เมื่อทำการลดค่าสนับสนุนขั้นต่ำหรือลดความค่าเชื่อมั่นน้อยสุดลงจนถึงระดับหนึ่งก็จะทำให้อัลกอริทึมสามารถค้นพบกฎความสัมพันธ์จากข้อมูลได้ ดังแสดงในรูปที่ 4.12

กำหนดค่าสนับสนุนขั้นต่ำ 30 เปอร์เซ็นต์ และกำหนดค่าความเชื่อมั่นขั้นต่ำ 95 เปอร์เซ็นต์

```
1 : class not_recom = 30240 -> health not_recom = 30240 conf : 1
2 : health not_recom = 30240 -> class not_recom = 30240 conf : 1
```

รูปที่ 4.12 แสดงการค้นหากฎความสัมพันธ์จากข้อมูล Nursery เมื่อมีการปรับลดค่าสนับสนุนขั้นต่ำและค่าความเชื่อมั่นขั้นต่ำ

จากผลการทดสอบการค้นหากฎความสัมพันธ์จากข้อมูลที่มีลักษณะของข้อมูลแตกต่างกันมาก เมื่อมีการปรับลดค่าสนับสนุนขั้นต่ำ หรือค่าความเชื่อมั่นขั้นต่ำลงจนถึงระดับที่โปรแกรมสามารถค้นพบกฎความสัมพันธ์จากชุดข้อมูลแล้ว ทำการทดสอบโดยการสุ่มข้อมูลก็จะให้ผลการค้นหากฎความสัมพันธ์ได้ดังแสดงในตารางที่ 4.16 ถึง 4.18

ตารางที่ 4.16 แสดงจำนวนกฎความสัมพันธ์ทั้งหมดที่ค้นพบจากข้อมูลทั้งหมดเปรียบเทียบกับชุดข้อมูลสุ่มที่มีขนาดต่างๆ กัน โดยทดสอบกับชุดข้อมูลที่มีลักษณะของข้อมูลเหมือนกันเกิดขึ้นซ้ำกันไม่มาก โดยตัวเลขที่ปรากฏในตารางคือจำนวนกฎความสัมพันธ์ที่ค้นพบ

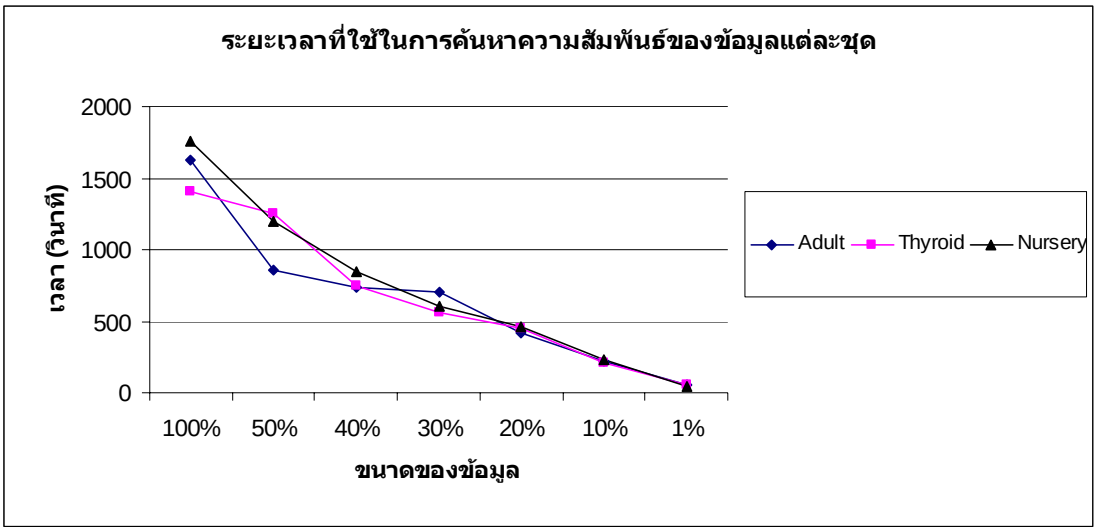
ชื่อชุดข้อมูล	ข้อมูลทั้งหมด (100%)	ข้อมูลสุ่ม					
		50%	40%	30%	20%	10%	1%
Adult	1	1	1	1	1	1	1
Thyroid	2	2	2	2	2	2	2
Nursery	2	2	2	2	2	2	2

ตารางที่ 4.17 แสดงจำนวนกฎความสัมพันธ์ที่มีความถูกต้องตรงกัน

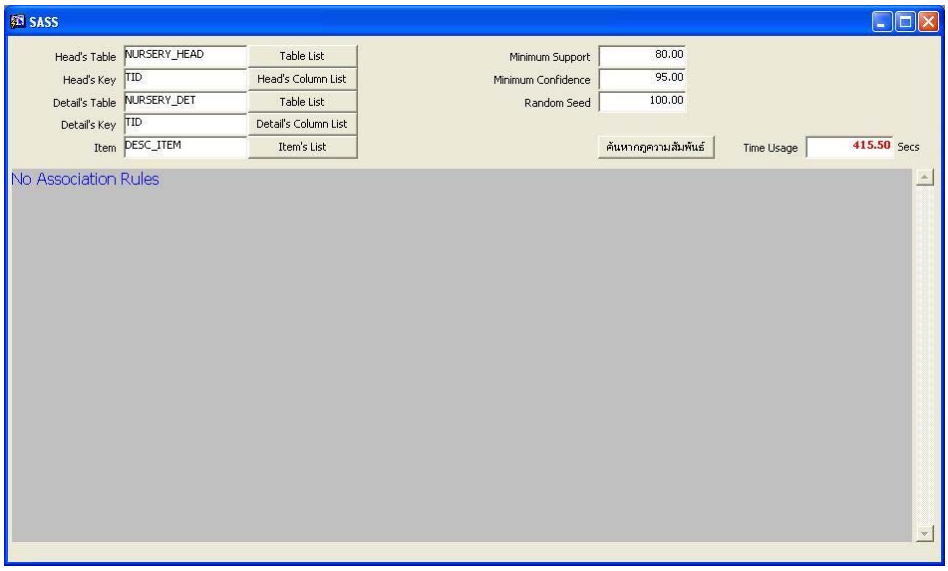
ชื่อชุดข้อมูล	ข้อมูลทั้งหมด (100%)	ข้อมูลสุ่ม					
		50%	40%	30%	20%	10%	1%
Adult	1	1	1	1	1	1	1
Thyroid	2	2	2	2	2	2	2
Nursery	2	2	2	2	2	2	2

ตารางที่ 4.18 แสดงเวลา (วินาที) ที่ใช้ในการค้นหากฎความสัมพันธ์ของข้อมูลแต่ละชุด

ชื่อชุดข้อมูล	ข้อมูลทั้งหมด (100%)	ข้อมูลสุ่ม					
		50%	40%	30%	20%	10%	1%
Adult	1625	861	733	700	422	215	51
Thyroid	1403	1253	750	562	456	212	53
Nursery	1761	1195	850	604	459	229	46



รูปที่ 4.13 กราฟเปรียบเทียบระยะเวลาในการค้นหาความสัมพันธ์ของข้อมูลที่มีลักษณะของข้อมูลเหมือนกันเกิดขึ้นซ้ำกันไม่มาก



รูปที่ 4.14 แสดงการค้นหากฎความสัมพันธ์จากข้อมูล Nursery โดยใช้โปรแกรมสำเร็จรูป โดยกำหนดค่าสนับสนุนขั้นต่ำที่ 80 เปอร์เซ็นต์ และกำหนดค่าความเชื่อมั่นขั้นต่ำที่ 95 เปอร์เซ็นต์

กฎความสัมพันธ์ที่ค้นพบจากชุดข้อมูลทั้งหมดของข้อมูล Nursery

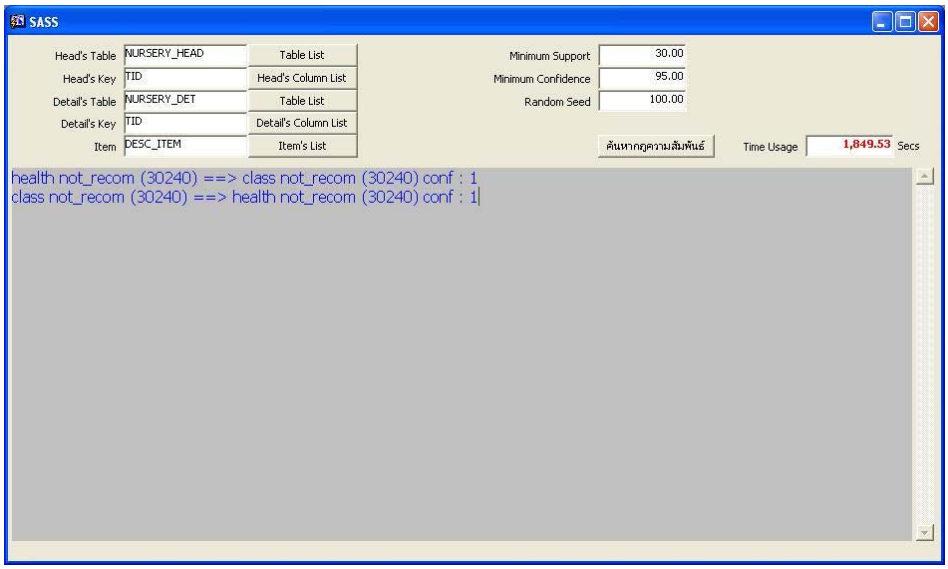
```
1 : class not_recom = 30240 -> health not_recom = 30240 conf : 1
2 : health not_recom = 30240 -> class not_recom = 30240 conf : 1
```

รูปที่ 4.15 แสดงการเปรียบเทียบกฎความสัมพันธ์ระหว่างกฎความสัมพันธ์ที่ค้นพบจากข้อมูลทั้งหมดกับกฎความสัมพันธ์ที่ค้นพบจากข้อมูลสุ่ม

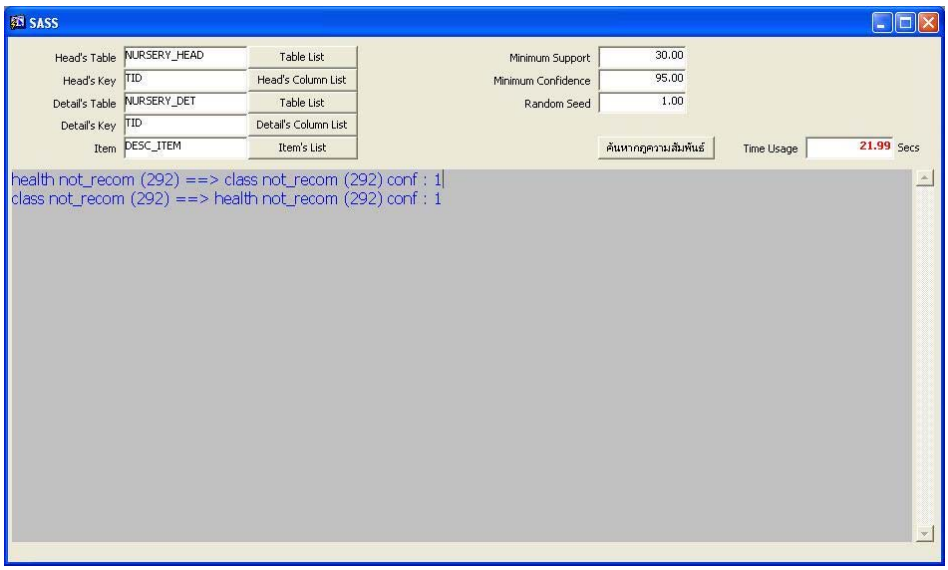
กฎความสัมพันธ์ที่ค้นพบจากชุดข้อมูลสุ่มขนาด 1 เปอร์เซ็นต์ของข้อมูล Nursery

```
1 : class not_recom = 298 -> health not_recom = 298 conf : 1
2 : health not_recom = 298 -> class not_recom = 298 conf : 1
```

รูปที่ 4.16 แสดงการเปรียบเทียบกฎความสัมพันธ์ระหว่างกฎความสัมพันธ์ที่ค้นพบจากข้อมูลทั้งหมดกับกฎความสัมพันธ์ที่ค้นพบจากข้อมูลสุ่ม



รูปที่ 4.17 แสดงการค้นหากฎความสัมพันธ์จากข้อมูลทั้งหมด (100 เปอร์เซ็นต์) ของข้อมูล Nursery โดยใช้โปรแกรมสำเร็จรูป โดยกำหนดค่าสนับสนุนขั้นต่ำอยู่ที่ 30 เปอร์เซ็นต์ และกำหนดค่าความเชื่อมั่นขั้นต่ำที่ 95 เปอร์เซ็นต์



รูปที่ 4.18 แสดงการค้นหากฎความสัมพันธ์จากข้อมูลสุ่มขนาด 1 เปอร์เซ็นต์ของข้อมูล Nursery โดยใช้โปรแกรมสำเร็จรูป โดยกำหนดค่าสนับสนุนขั้นต่ำอยู่ที่ 30 เปอร์เซ็นต์ และกำหนดค่าความเชื่อมั่นขั้นต่ำที่ 95 เปอร์เซ็นต์

4.3 ผลการทดสอบกับข้อมูลสตรีมมิง

การทดสอบการค้นหากฎความสัมพันธ์กับข้อมูลสตรีมมิง โดยการทดสอบกับข้อมูลทั้งสองกลุ่มคือ กลุ่มของข้อมูลที่มีลักษณะของข้อมูลเหมือนกันเกิดขึ้นซ้ำมากๆ และกลุ่มของข้อมูลที่มีลักษณะของข้อมูลที่เหมือนกันเกิดขึ้นซ้ำกันไม่มาก ทำการค้นหากฎความสัมพันธ์โดยการเรียกใช้โปรแกรม SASS โดยตรง โดยการทดสอบกับข้อมูลที่มีลักษณะของข้อมูลที่เหมือนกันเกิดขึ้นซ้ำกันมากๆ จะกำหนดค่าสนับสนุนขั้นต่ำที่ 80 เปอร์เซ็นต์ และกำหนดค่าความเชื่อมั่นขั้นต่ำ 95 เปอร์เซ็นต์ ส่วนการทดสอบกับข้อมูลทำการทดสอบกับข้อมูลที่มีลักษณะของข้อมูลที่เหมือนกันเกิดขึ้นซ้ำกันไม่มากจะกำหนดค่าสนับสนุนขั้นต่ำ และค่าความเชื่อมั่นขั้นต่ำในระดับที่สามารถค้นหากฎความสัมพันธ์ได้ การทดสอบกับข้อมูลทั้งสองกลุ่มจะทำการทดสอบกับข้อมูลทั้งหมด (100 เปอร์เซ็นต์) และข้อมูลสุ่มที่มีขนาดต่างๆ กันคือ 1, 10, 20, 30, 40 และ 50 เปอร์เซ็นต์ และทำการเพิ่มข้อมูลบางส่วนเข้าไปในขณะที่โปรแกรมกำลังทำการค้นหากฎความสัมพันธ์ของข้อมูลแต่ละชุด

```
1 : birth United_States = 116306 -> citizenship
    Native_Born_in_the_United_States = 116306 conf : 1
2 : citizenship Native_Born_in_the_United_States = 116309 -> birth
    United_States = 116306 conf : 1
3 : birth United_States,edu_inst Not_in_universe = 108876 ->
    citizenship Native_Born_in_the_United_States = 108876 conf : 1
4 : citizenship Native_Born_in_the_United_States,edu_inst
    Not_in_universe = 108878 -> birth United_States = 108876 conf :
    1
5 : birth United_States,salary -50000 = 108997 -> citizenship
    Native_Born_in_the_United_States = 108997 conf : 1
6 : Native_Born_in_the_United_States,salary -50000 = 109000 ->
    birth United_States = 108997 conf : 1
```

รูปที่ 4.19 แสดงกฎความสัมพันธ์ที่ค้นพบจากข้อมูลทั้งหมด (100 เปอร์เซ็นต์) ของชุดข้อมูล Census Income

```
1 : birth United_States = 117508 -> citizenship
    Native_Born_in_the_United_States = 117508 conf : 1
2 : citizenship Native_Born_in_the_United_States = 117511 -> birth
    United_States = 117508 conf : 1
3 : birth United_States,salary -50000 = 110134 -> citizenship
    Native_Born_in_the_United_States = 110134 conf : 1
4 : citizenship Native_Born_in_the_United_States,salary -50000 =
    110137 -> birth United_States = 110134 conf : 1
5 : birth United_States,edu_inst Not_in_universe = 110001 ->
    citizenship Native_Born_in_the_United_States = 110001 conf : 1
6 : citizenship Native_Born_in_the_United_States,edu_inst
    Not_in_universe = 110003 -> birth United_States = 110001 conf :
    1
```

รูปที่ 4.20 แสดงกฎความสัมพันธ์ที่ค้นพบจากข้อมูลทั้งหมด (100 เปอร์เซ็นต์) และมีการเพิ่มข้อมูลเข้าไปขณะที่อัลกอริทึมกำลังทำการประมวลผลของชุดข้อมูล Census Income

```
1 : birth United_States = 117508 -> citizenship
    Native_Born_in_the_United_States = 117508 conf : 1
2 : citizenship Native_Born_in_the_United_States = 117511 -> birth
    United_States = 117508 conf : 1
3 : birth United_States,salary -50000 = 110134 -> citizenship
    Native_Born_in_the_United_States = 110134 conf : 1
4 : citizenship Native_Born_in_the_United_States,salary -50000 =
    110137 -> birth United_States = 110134 conf : 1
5 : birth United_States,edu_inst Not_in_universe = 110001 ->
    citizenship Native_Born_in_the_United_States = 110001 conf : 1
6 : citizenship Native_Born_in_the_United_States,edu_inst
    Not_in_universe = 110003 -> birth United_States = 110001 conf :
    1
```

รูปที่ 4.21 แสดงกฎความสัมพันธ์ที่ค้นพบจากข้อมูลทั้งหมด (100 เปอร์เซ็นต์) รวมกับข้อมูลที่ทำการเพิ่มเข้าไปของชุดข้อมูล Census Income

```
1 : birth United_States = 58325 -> citizenship
    Native_Born_in_the_United_States = 58325 conf : 1
2 : citizenship Native_Born_in_the_United_States = 58326 -> birth
    United_States = 58325 conf : 1
3 : birth United_States,edu_inst Not_in_universe = 54500 ->
    citizenship Native_Born_in_the_United_States = 54500 conf : 1
4 : citizenship Native_Born_in_the_United_States,edu_inst
    Not_in_universe = 54500 -> birth United_States = 54500 conf : 1
5 : birth United_States,salary -50000 = 54645 -> citizenship
    Native_Born_in_the_United_States = 54645 conf : 1
6 : citizenship Native_Born_in_the_United_States,salary -50000 =
    54646 -> birth United_States = 54645 conf : 1
```

รูปที่ 4.22 แสดงกฎความสัมพันธ์ที่ค้นพบจากข้อมูลสุ่มขนาด 50 เปอร์เซ็นต์ ของชุดข้อมูล Census Income

```
1 : birth United_States = 58788 -> citizenship
    Native_Born_in_the_United_States = 58628 conf : 1
2 : citizenship Native_Born_in_the_United_States = 58790 -> birth
    United_States = 58628 conf : 1
3 : birth United_States,edu_inst Not_in_universe = 54853 ->
    citizenship Native_Born_in_the_United_States = 54836 conf : 1
4 : citizenship Native_Born_in_the_United_States,edu_inst
    Not_in_universe = 54854 -> birth United_States = 54836 conf : 1
5 : birth United_States,salary -50000 = 54929 -> citizenship
    Native_Born_in_the_United_States = 54911 conf : 1
6 : citizenship Native_Born_in_the_United_States,salary -50000 =
    54931 -> birth United_States = 54911 conf : 1
```

รูปที่ 4.23 แสดงกฎความสัมพันธ์ที่ค้นพบจากข้อมูลสุ่มขนาด 50 เปอร์เซ็นต์ และมีการเพิ่มข้อมูลเข้าไปขณะที่อัลกอริทึมกำลังทำการประมวลผลของชุดข้อมูล Census Income

```
1 : birth United_States = 58667 -> citizenship
    Native_Born_in_the_United_States = 58667 conf : 1
2 : citizenship Native_Born_in_the_United_States = 58667 -> birth
    United_States = 58667 conf : 1
3 : birth United_States,edu_inst Not_in_universe = 54908 ->
    citizenship Native_Born_in_the_United_States = 54908 conf : 1
4 : citizenship Native_Born_in_the_United_States,edu_inst
    Not_in_universe = 54908 -> birth United_States = 54908 conf : 1
5 : birth United_States,salary -50000 = 54976 -> citizenship
    Native_Born_in_the_United_States = 54976 conf : 1
6 : citizenship Native_Born_in_the_United_States,salary -50000 =
    54976 -> birth United_States = 54976 conf : 1
```

รูปที่ 4.24 แสดงกฎความสัมพันธ์ที่ค้นพบจากข้อมูลสุ่มขนาด 50 เปอร์เซ็นต์ รวมกับข้อมูลที่ทำการเพิ่มเข้าไปของชุดข้อมูล Census Income

```
1 : birth United_States = 1145 -> citizenship
    Native_Born_in_the_United_States = 1145 conf : 1
2 : citizenship Native_Born_in_the_United_States = 1145 -> birth
    United_States = 1145 conf : 1
3 : birth United_States,salary -50000 = 1070 -> citizenship
    Native_Born_in_the_United_States = 1070 conf : 1
4 : citizenship Native_Born_in_the_United_States,salary -50000 =
    1070 -> birth United_States = 1070 conf : 1
5 : birth United_States,edu_inst Not_in_universe = 1070 ->
    citizenship Native_Born_in_the_United_States = 1070 conf : 1
6 : citizenship Native_Born_in_the_United_States,edu_inst
    Not_in_universe = 1070 -> birth United_States = 1070 conf : 1
```

รูปที่ 4.25 แสดงกฎความสัมพันธ์ที่ค้นพบจากข้อมูลสุ่มขนาด 1 เปอร์เซ็นต์ ของชุดข้อมูล Census Income

```
1 : birth United_States = 1230 -> citizenship
    Native_Born_in_the_United_States = 1220 conf : .99
2 : citizenship Native_Born_in_the_United_States = 1230 -> birth
    United_States = 1220 conf : .99
3 : birth United_States,edu_inst Not_in_universe = 1138 ->
    citizenship Native_Born_in_the_United_States = 1130 conf : .99
4 : citizenship Native_Born_in_the_United_States,edu_inst
    Not_in_universe = 1138 -> birth United_States = 1130 conf : .99
5 : birth United_States,salary -50000 = 1134 -> citizenship
    Native_Born_in_the_United_States = 1127 conf : .99
6 : citizenship Native_Born_in_the_United_States,salary -50000 =
    1134 -> birth United_States = 1127 conf : .99
```

รูปที่ 4.26 แสดงกฎความสัมพันธ์ที่ค้นพบจากข้อมูลสุ่มขนาด 1 เปอร์เซ็นต์ และมีการเพิ่มข้อมูลเข้าไปขณะที่อัลกอริทึมกำลังทำการประมวลผลของชุดข้อมูล Census Income

```
1 : birth United_States = 1167 -> citizenship
    Native_Born_in_the_United_States = 1167 conf : 1
2 : citizenship Native_Born_in_the_United_States = 1168 -> birth
    United_States = 1167 conf : 1
3 : birth United_States,edu_inst Not_in_universe = 1082 ->
    citizenship Native_Born_in_the_United_States = 1082 conf : 1
4 : citizenship Native_Born_in_the_United_States,edu_inst
    Not_in_universe = 1083 -> birth United_States = 1082 conf : 1
5 : birth United_States,salary -50000 = 1095 -> citizenship
    Native_Born_in_the_United_States = 1095 conf : 1
6 : citizenship Native_Born_in_the_United_States,salary -50000 =
    1096 -> birth United_States = 1095 conf : 1
```

รูปที่ 4.27 แสดงกฎความสัมพันธ์ที่ค้นพบจากข้อมูลสุ่มขนาด 1 เปอร์เซ็นต์ รวมกับข้อมูลที่ทำการเพิ่มเข้าไปของชุดข้อมูล Census Income

พิจารณารูปที่ 4.20 และ 4.21 เมื่อนำกฎความสัมพันธ์ที่ปรากฏอยู่ในภาพทั้งสองนี้มาเปรียบเทียบกัน จะพบว่ากฎความสัมพันธ์ที่ถูกค้นพบค่าสนับสนุน และค่าความเชื่อมั่นมีความเหมือนกันทุกประการ เนื่องจากเป็นการค้นหากฎความสัมพันธ์จากข้อมูลทั้งหมด (100 เปอร์เซ็นต์) และกฎความสันพันธ์มีความถูกต้องตรงกันกับกฎความสัมพันธ์ที่ปรากฏอยู่ในรูปที่ 4.19 เมื่อพิจารณารูปที่ 4.23 และ 4.24 จะพบว่ากฎความสัมพันธ์ที่ถูกค้นพบมีค่าสนับสนุนแตกต่างกันเล็กน้อย เนื่องจากเป็นการค้นหากฎความสัมพันธ์จากข้อมูลสุ่มขนาด 50 เปอร์เซนต์ทำให้ข้อมูลที่ถูกเลือกมาเป็นตัวแทนในแต่ละครั้งอาจมีความแตกต่างกันเล็กน้อย แต่กฎความสัมพันธ์ที่ถูกค้นพบยังมีความถูกต้องตรงกันกับกฎความสัมพันธ์ที่ปรากฏอยู่ในรูปที่ 4.19 เมื่อพิจารณารูปที่ 4.26 และ 4.27 จะพบว่ามีค่าสนับสนุนและค่าความเชื่อมั่นแตกต่างกันเล็กน้อย เนื่องจากเป็นการค้นหากฎความสัมพันธ์จากข้อมูลสุ่มขนาด 1 เปอร์เซ็นต์ดังนั้นซึ่งขนาดของตัวแทนมีขนาดเล็กมากอาจทำให้มีการคลาดเคลื่อนได้ แต่เมื่อพิจารณากฎความสัมพันธ์ที่ถูกค้นพบจะพบว่ากฎความสัมพันธ์ยังมีความถูกต้องตรงกันกับกฎความสัมพันธ์ที่ปรากฏในรูปที่ 4.19

```
1 : class not_recom = 30240 -> health not_recom = 30240 conf : 1
2 : health not_recom = 30240 -> class not_recom = 30240 conf : 1
```

รูปที่ 4.28 แสดงกฎความสัมพันธ์ที่ค้นพบจากข้อมูลทั้งหมด (100 เปอร์เซ็นต์) ของชุดข้อมูล Nursery

```
1 : class not_recom = 30556 -> health not_recom = 30556 conf : 1
2 : health not_recom = 30556 -> class not_recom = 30556 conf : 1
```

รูปที่ 4.29 แสดงกฎความสัมพันธ์ที่ค้นพบจากข้อมูลทั้งหมด (100 เปอร์เซ็นต์) และมีการเพิ่มข้อมูลเข้าไปขณะที่อัลกอริทึมกำลังทำการประมวลผลของชุดข้อมูล Nursery

```
1 : class not_recom = 30556 -> health not_recom = 30556 conf : 1
2 : health not_recom = 30556 -> class not_recom = 30556 conf : 1
```

รูปที่ 4.30 แสดงกฎความสัมพันธ์ที่ค้นพบจากข้อมูลทั้งหมด (100 เปอร์เซ็นต์) รวมกับข้อมูลที่ทำการเพิ่มเข้าไปของชุดข้อมูล Nursery

```
1 : class not_recom = 15233 -> health not_recom = 15233 conf : 1
2 : health not_recom = 15233 -> class not_recom = 15233 conf : 1
```

รูปที่ 4.31 แสดงกฎความสัมพันธ์ที่ค้นพบจากข้อมูลสุ่มขนาด 50 เปอร์เซ็นต์ ของชุดข้อมูล Nursery

```
1 : class not_recom = 15214 -> health not_recom = 15192 conf : 1
2 : health not_recom = 15214 -> class not_recom = 15192 conf : 1
```

รูปที่ 4.32 แสดงกฎความสัมพันธ์ที่ค้นพบจากข้อมูลสุ่มขนาด 50 เปอร์เซ็นต์ และมีการเพิ่มข้อมูลเข้าไปขณะที่อัลกอริทึมกำลังทำการประมวลผลของชุดข้อมูล Nursery

```
1 : class not_recom = 15504 -> health not_recom = 15504 conf : 1
2 : health not_recom = 15504 -> class not_recom = 15504 conf : 1
```

รูปที่ 4.33 แสดงกฎความสัมพันธ์ที่ค้นพบจากข้อมูลสุ่มขนาด 50 เปอร์เซ็นต์ รวมกับข้อมูลที่ทำการเพิ่มเข้าไปของชุดข้อมูล Nursery

```
1 : class not_recom = 298 -> health not_recom = 298 conf : 1
2 : class not_recom = 298 -> health not_recom = 298 conf : 1
```

รูปที่ 4.34 แสดงกฎความสัมพันธ์ที่ค้นพบจากข้อมูลสุ่มขนาด 1 เปอร์เซ็นต์ ของชุดข้อมูล Nursery

```
1 : class not_recom = 325 -> health not_recom = 325 conf : 1
2 : health not_recom = 325 -> class not_recom = 325 conf : 1
```

รูปที่ 4.35 แสดงกฎความสัมพันธ์ที่ค้นพบจากข้อมูลสุ่มขนาด 1 เปอร์เซ็นต์ และมีการเพิ่มข้อมูลเข้าไปขณะที่อัลกอริทึมกำลังทำการประมวลผลของชุดข้อมูล Nursery

```
1 : class not_recom = 292 -> health not_recom = 292 conf : 1
2 : class not_recom = 292 -> health not_recom = 292 conf : 1
```

รูปที่ 4.36 แสดงกฎความสัมพันธ์ที่ค้นพบจากข้อมูลสุ่มขนาด 1 เปอร์เซ็นต์ รวมกับข้อมูลที่ทำการเพิ่มเข้าไปของชุดข้อมูล Nursery

จากรูปที่ 4.28 และ 4.36 เป็นการผลทดสอบการทำงานของโปรแกรม SASS กับข้อมูลสตรีมมิง โดยเป็นการทดสอบกลุ่มข้อมูลที่มีลักษณะของข้อมูลที่เหมือนกันเกิดขึ้นซ้ำกันไม่มาก ทำการทดสอบเช่นเดียวกันกับการทดสอบกับกลุ่มข้อมูลที่มีลักษณะของข้อมูลที่เหมือนกันเกิดขึ้นซ้ำกันมากๆ ผลการทดสอบที่ได้มีลักษณะเช่นเดียวกันกับผลการทดสอบที่ได้จากการทดสอบกับกลุ่มข้อมูลที่มีลักษณะของข้อมูลที่เหมือนกันเกิดขึ้นซ้ำกันมากๆ ดังนี้ รูปที่ 4.28, 4.29 และ 4.30 เป็นผลการทดสอบกับข้อมูลทั้งหมด (100 เปอร์เซ็นต์) รูปที่ 4.31, 4.32 และ 4.33 เป็นผลการทดสอบกับข้อมูลสุ่ม 50 เปอร์เซ็นต์ รูปที่ 4.34, 4.35 และ 4.3 เป็นผลการทดสอบกับข้อมูลสุ่ม 1 เปอร์เซ็นต์

4.4 อภิปรายผล

ผลการทดสอบเทคนิคการสุ่มตัวอย่างแต่ละแบบ โดยทดสอบกับชุดข้อมูลที่มีลักษณะของข้อมูลเหมือนกันเกิดขึ้นซ้ำมากๆ และผลการทดสอบอัลกอริทึมในการค้นหากฎความสัมพันธ์กับข้อมูลทั้งสองประเภท โดยทดสอบกับชุดข้อมูลทั้งหมด และชุดข้อมูลสุ่ม โดยผลที่ได้จากการทดสอบปรากฏอยู่ในตารางและรูปที่แสดงข้างต้น สามารถสรุปได้ดังนี้

4.4.1 ผลการทดสอบเทคนิคการสุ่มตัวอย่างแต่ละแบบ โดยทดสอบกับชุดข้อมูลที่มีลักษณะของข้อมูลเหมือนกันเกิดขึ้นซ้ำมากๆ

1) เมื่อใช้เทคนิคการสุ่มตัวอย่างทั้ง 5 เทคนิคทำการทดสอบกับชุดข้อมูลที่มีลักษณะของข้อมูลเหมือนกันเกิดขึ้นซ้ำมากๆ ทั้ง 4 ชุดข้อมูล โดยทดสอบกับข้อมูลทั้งหมด และข้อมูลสุ่มที่มีขนาดต่างๆ กัน คือ 1, 10, 20, 30, 40 และ 50 เปอร์เซ็นต์ ปรากฏว่าเทคนิคการสุ่มตัวอย่างแบบง่ายโดยใช้หลักความน่าจะเป็นและใช้หลักการสุ่มแบบไม่ใส่คืนที่ จะใช้ระยะเวลาในการค้นหากฎความสัมพันธ์น้อยที่สุด

2) เมื่อนำกฎความสัมพันธ์ที่ค้นพบจากชุดข้อมูลสุ่มที่ได้จากการค้นพบจากเทค-นิคการสุ่มตัวอย่างทั้ง 5 เทคนิค มาเปรียบเทียบกัน จะเห็นได้ว่ากฎความสัมพันธ์ที่ค้นพบจากชุดข้อมูลสุ่ม โดยการใช้เทคนิคการสุ่มข้อมูลแต่ละเทคนิคจะมีความตรงกัน ยกเว้นกฎความสัมพันธ์ที่ค้นพบจากชุดข้อมูลสุ่มบางชุด โดยเฉพาะชุดข้อมูลสุ่ม 1 เปอร์เซ็นต์ ที่อาจมีการค้นพบกฎความสัมพันธ์ที่มีความแตกต่างไปจากชุดข้อมูลสุ่มชุดอื่นๆ

4.4.2 ผลการทดสอบของโปรแกรม SASS เมื่อทดสอบกับข้อมูลที่มีลักษณะของข้อมูลเหมือนกันเกิดขึ้นซ้ำมากๆ

1) เมื่อทำทดสอบโปรแกรม SASS กับชุดข้อมูลที่มีลักษณะของข้อมูลเหมือนกันเกิดขึ้นซ้ำมากๆ โดยทดสอบกับข้อมูลทั้งหมดเปรียบเทียบกับข้อมูลสุ่ม จะเห็นได้ว่าเมื่อข้อมูลมีขนาดเล็กลงจะทำให้ระยะเวลาในการค้นหากฎความสัมพันธ์ของโปรแกรมจะลดลงไปด้วย

2) เมื่อเปรียบเทียบปริมาณของกฎความสัมพันธ์ที่ค้นพบจากข้อมูลสุ่มกับปริมาณกฎความสัมพันธ์ที่ค้นพบจากข้อมูลทั้งหมด จะเห็นได้ว่ากฎความสัมพันธ์ที่ค้นพบจากข้อมูลสุ่ม และกฎความสัมพันธ์ที่ค้นพบจากข้อมูลทั้งหมดมีปริมาณเท่ากัน ยกเว้นข้อมูลสุ่มบางชุด โดยเฉพาะข้อมูลสุ่ม 1 เปอร์เซ็นต์ ที่อาจมีการค้นพบกฎความสัมพันธ์ในปริมาณที่แตกต่างไปจากกฎความสัมพันธ์ที่ค้นพบจากข้อมูลทั้งหมด

3) เมื่อนำกฎความสัมพันธ์ที่ค้นพบจากข้อมูลสุ่ม มาเปรียบเทียบความถูกต้องตรงกันกับกฎความสัมพันธ์ที่ค้นพบจากข้อมูลทั้งหมด จะเห็นได้ว่ากฎความสัมพันธ์ที่ค้นพบจากข้อมูลสุ่มแต่ละชุดมีความถูกต้องตรงกันกับกฎความสัมพันธ์ที่ค้นพบจากข้อมูลทั้งหมด โดยมีอัตราความ

ถูกต้องของกฎความสัมพันธ์เป็น 100 เปอร์เซ็นต์ โดยการเปรียบเทียบความถูกต้องของกฎความสัมพันธ์จะดูจากกฎความสัมพันธ์ที่ค้นพบจากข้อมูลทั้งหมดเป็นหลัก แล้วนำกฎความสัมพันธ์ที่ค้นพบได้จากข้อมูลสุ่มมาเปรียบเทียบ เช่น ถ้ากฎความสัมพันธ์ที่ค้นพบได้จากชุดข้อมูลทั้งหมดมี 10 กฎ จะนำกฎความสัมพันธ์ที่ค้นพบได้จากชุดข้อมูลสุ่มมาเรียงลำดับตามค่าความเชื่อมั่น โดยเรียงจากมากสุดไปน้อยสุด จากนั้นนำกฎความสัมพันธ์ใน 10 ลำดับแรกไปเปรียบเทียบหากมีความถูกต้องตรงกันกับกฎความสัมพันธ์ที่ได้จากชุดข้อมูลทั้งหมดโดยลำดับอาจไม่ตรงกันก็ได้ ให้ถือว่ากฎความสัมพันธ์นั้นมีความถูกต้องตรงกัน 100 เปอร์เซ็นต์

4.4.3 ผลการทดสอบของโปรแกรม SASS เมื่อทดสอบกับข้อมูลที่มีลักษณะของข้อมูลเหมือนกันเกิดขึ้นซ้ำกันไม่มาก

1) เมื่อทำการทดสอบโปรแกรม SASS กับชุดข้อมูลที่มีลักษณะของข้อมูลเหมือนกันเกิดขึ้นซ้ำไม่มาก โดยกำหนดค่าสนับสนุนขั้นต่ำที่ 80 เปอร์เซ็นต์ และค่าความเชื่อมั่นขั้นต่ำที่ 95 เปอร์เซ็นต์จะพบว่าอัลกอริทึมไม่สามารถค้นหากฎความสัมพันธ์ได้จากข้อมูลประเภทนี้ และเมื่อทำการลดค่าสนับสนุนขั้นต่ำ หรือค่าความเชื่อมั่นขั้นต่ำลงจนถึงระดับหนึ่งโปรแกรมก็จะสามารถค้นหากฎความสัมพันธ์พบได้

2) เมื่อทำการทดสอบโปรแกรม SASS กับข้อมูลทั้งหมดเปรียบเทียบกับชุดข้อมูลสุ่มจะเห็นได้ว่าเมื่อข้อมูลมีขนาดเล็กลงจะทำให้ระยะเวลาในการค้นหากฎความสัมพันธ์ของโปรแกรมลดลงไปด้วยเช่นเดียวกับการทดสอบกับข้อมูลที่มีลักษณะของข้อมูลเหมือนกันเกิดขึ้นซ้ำมากๆ

3) เมื่อเปรียบเทียบปริมาณของกฎความสัมพันธ์ที่ค้นพบจากข้อมูลสุ่มกับปริมาณกฎความสัมพันธ์ที่ค้นพบจากข้อมูลทั้งหมด จะเห็นได้ว่ากฎความสัมพันธ์ที่ค้นพบจากข้อมูลสุ่ม และกฎความสัมพันธ์ที่ค้นพบจากข้อมูลทั้งหมดมีปริมาณเท่ากัน ยกเว้นข้อมูลสุ่มบางชุด โดยเฉพาะข้อมูลสุ่ม 1 เปอร์เซ็นต์ ที่อาจมีการค้นพบกฎความสัมพันธ์ในปริมาณที่แตกต่างไปจากกฎความสัมพันธ์ที่ค้นพบจากข้อมูลทั้งหมด

4) เมื่อนำกฎความสัมพันธ์ที่ค้นพบจากข้อมูลสุ่ม มาเปรียบเทียบความถูกต้องตรงกันกับกฎความสัมพันธ์ที่ค้นพบจากข้อมูลทั้งหมด จะเห็นได้ว่ากฎความสัมพันธ์ที่ค้นพบจากข้อมูลสุ่มแต่ละชุดมีความถูกต้องตรงกันกับกฎความสัมพันธ์ที่ค้นพบจากข้อมูลทั้งหมด โดยมีอัตราความถูกต้องของกฎความสัมพันธ์เป็น 100 เปอร์เซ็นต์

4.4.4 ผลการทดสอบของโปรแกรม SASS เมื่อทดสอบกับข้อมูลสตรีมมิง

1) เมื่อทำทดสอบโปรแกรม SASS กับชุดข้อมูลที่มีลักษณะของข้อมูลเหมือนกันเกิดขึ้นซ้ำมากๆ โดยทดสอบกับข้อมูลทั้งหมดและทำการเพิ่มข้อมูลเข้าไปจำนวนหนึ่ง ในขณะที่โปรแกรมกำลังทำการประมวลผล เมื่อโปรแกรมสิ้นสุดการประมวลผลจะพบว่าโปรแกรมสามารถ

นำข้อมูลที่ถูกเพิ่มเข้าไป ไปทำการค้นหาความสัมพันธ์ได้เช่นเดียวกันกับข้อมูลทรานแซคชันที่มีอยู่แล้วในฐานข้อมูล โดยการเปรียบเทียบจากการทำการทดสอบโปรแกรม SASS กับข้อมูลทั้งหมดอีกครั้งหนึ่งหลังจากที่เพิ่มข้อมูลเข้าไปแล้ว แล้วนำกฎความสัมพันธ์ที่ค้นพบไปเปรียบเทียบกฎความสัมพันธ์ที่ค้นพบขณะที่ทำการเพิ่มข้อมูล จะพบว่าโปรแกรมสามารถค้นหาความสัมพันธ์ของข้อมูลขณะที่ทำการเพิ่มข้อมูลเข้าไปได้ถูกต้องตรงกัน มีค่าสนับสนุน และค่าความเชื่อมั่นที่เท่ากันทุกประการ

2) เมื่อทำการทดสอบโปรแกรม SASS กับข้อมูลสุ่มขนาด 50 เปอร์เซ็นต์ และทำการเพิ่มข้อมูลเข้าไปจำนวนหนึ่งขณะที่โปรแกรมกำลังทำการประมวลผล เมื่อโปรแกรมสิ้นสุดการประมวลผลจะพบว่าโปรแกรมสามารถค้นหากฎความสัมพันธ์ได้ถูกต้อง และมีค่าความเชื่อมั่นของกฎความสัมพันธ์ถูกต้องตรงกันกับการค้นหาความสัมพันธ์จากข้อมูลทั้งหมด

3) เมื่อทำการทดสอบโปรแกรม SASS กับข้อมูลสุ่มขนาด 1 เปอร์เซ็นต์ และทำการเพิ่มข้อมูลเข้าไปจำนวนหนึ่งในขณะที่โปรแกรมกำลังทำการประมวลผล เมื่อโปรแกรมสิ้นสุดการประมวลผลจะพบว่าโปรแกรมสามารถค้นหาความสัมพันธ์ได้ถูกต้องตรงกันกับการค้นหากฎความ-สัมพันธ์จากข้อมูลทั้งหมด แต่ค่าความเชื่อมั่นของกฎความสัมพันธ์ที่ถูกค้นพบจากข้อมูลสุ่มอาจมีความคลาดเคลื่อนไปเล็กน้อย เนื่องจากกลุ่มตัวอย่างมีขนาดเล็กอาจทำให้ตัวแทนของข้อมูลมีข้อมูลที่แตกต่างกันมากขึ้น

4) เมื่อทำการทดสอบโปรแกรม SASS กับข้อมูลที่มีลักษณะของข้อมูลเหมือนกันเกิดขึ้นซ้ำกันไม่มาก โดยทำการทดสอบเช่นเดียวกับการทดสอบกับข้อมูลที่มีลักษณะของข้อมูลเหมือนกันเกิดขึ้นซ้ำมากๆ และกำหนดค่าสนับสนุนขั้นต่ำ และค่าความเชื่อมั่นขั้นต่ำในระดับที่เหมาะสม จากการทดสอบได้ผลการทดสอบในลักษณะเช่นเดียวกันกับผลการทดสอบกับข้อมูลที่มีลักษณะของข้อมูลเหมือนกันเกิดขึ้นซ้ำมากๆ

บทที่ 5
บทสรุป

การค้นหากฎความสัมพันธ์เป็นกระบวนการในการทำเหมืองข้อมูลรูปแบบหนึ่ง โดยเป็นการอธิบายความรู้ที่ได้มาให้อยู่ในรูปแบบของกฎความสัมพันธ์ โดยการวิเคราะห์ข้อมูลที่มีความสัมพันธ์กัน กระบวนการค้นหากฎความสัมพันธ์บนฐานข้อมูลขนาดใหญ่มากๆ เป็นกระบวนการที่ต้องใช้เวลาในการประมวลผลที่ยาวนานมาก อาจทำให้ไม่สามารถนำเอาความรู้ที่ได้จากกระบวนการวิเคราะห์ข้อมูลไปใช้ประโยชน์ทันในระยะเวลาที่ต้องการ การลดขนาดของข้อมูลเป็นวิธีการที่จะช่วยลดปัญหาเรื่องระยะเวลาในกระบวนการขุดค้นความรู้ได้

งานวิจัยนี้มีจุดมุ่งหมายที่จะพัฒนาโปรแกรม SASS ที่ประกอบด้วยอัลกอริทึม “เอไพรออริ” ที่ใช้ในการทำเหมืองข้อมูล ในการค้นหากฎความสัมพันธ์ที่แข็งแกร่งกับฐานข้อมูลขนาดใหญ่ โดยเป็นข้อมูลที่มีลักษณะของข้อมูลที่มีความตรงกันมาก หรือมีข้อมูลเหมือนกันเกิดขึ้นซ้ำมากๆ หรือมีลักษณะเป็นข้อมูลเชิงธุรกิจ โดยมุ่งเน้นให้โปรแกรม SASS สามารถค้นหากฎความสัมพันธ์ที่แข็งแกร่งได้อย่างรวดเร็ว โดยศึกษาและพัฒนาเทคนิคการสุ่มข้อมูลเพื่อพิจารณาเทคนิคการสุ่มข้อมูลที่มีประสิทธิภาพโดยสามารถสุ่มข้อมูลได้โดยที่ไม่สามารถจำนวนข้อมูลทั้งหมดก่อนการสุ่ม และจะต้องสามารถสร้างชุดข้อมูลสุ่มได้โดยการอ่านฐานข้อมูลเพียงหนึ่งรอบเท่านั้น

ขั้นตอนการดำเนินงานวิจัยแบ่งออกเป็น การศึกษาและทดสอบประสิทธิภาพของอัลกอริทึมในการสุ่มข้อมูลแบบต่างๆ เพื่อวิเคราะห์หาลักษณะเด่นของแต่ละอัลกอริทึมเพื่อนำไปใช้ประกอบการพัฒนาอัลกอริทึมเอไพรออริให้สามารถทำงานได้รวดเร็วขึ้นตามวัตถุประสงค์ของงานวิจัยนี้ การทดสอบประสิทธิภาพของอัลกอริทึมในการสุ่มข้อมูลแบบต่างๆ จะทำโดยการทดสอบอัลกอริทึมกับข้อมูลทั้งหมดสี่ชุด โดยแต่ละชุดจะเป็นการทดสอบกับชุดข้อมูลทั้งหมด และทดสอบกับชุดข้อมูลสุ่มที่มีขนาด 1, 10, 20, 30, 40 และ 50 เปอร์เซ็นต์ ทำการเปรียบเทียบผลที่ได้ โดยพิจารณาจากระยะเวลาที่ใช้ในการค้นหากฎความสัมพันธ์ของแต่ละอัลกอริทึม

ขั้นตอนการพัฒนาโปรแกรม SASS ผู้วิจัยได้ทำการพัฒนาโดยการปรับปรุงมาจากอัลกอริทึมเอไพรออริโดยเพิ่มส่วนที่ใช้ในการสุ่มข้อมูลเพื่อช่วยลดขนาดของข้อมูล และช่วยให้อัลกอริทึมสามารถทำงานได้รวดเร็วขึ้น และปรับปรุงการทำงานของอัลกอริทึมให้สามารถรองรับการทำงานกับข้อมูลสตรีมมิง

ขั้นตอนการทดสอบประสิทธิภาพของโปรแกรม SASS จะทำการทดสอบกับข้อมูลทั้งหมดเจ็ดชุดโดยแต่ละชุดจะเป็นการทดสอบกับชุดข้อมูลทั้งหมด และทดสอบกับชุดข้อมูลสุ่มที่มีขนาด 1, 10, 20, 30, 40 และ 50 เปอร์เซ็นต์ จากนั้นเปรียบเทียบผลที่ได้โดยพิจารณาจากระยะเวลาที่ใช้ในการค้นหากฎความสัมพันธ์ ปริมาณกฎความสัมพันธ์ที่ถูกค้นพบ และความถูกต้องตรงกันของกฎความสัมพันธ์ที่ถูกค้นพบจากข้อมูลทั้งหมด และข้อมูลสุ่ม และการทดสอบโปรแกรม SASS กับข้อมูลทรานแซคชันพร้อมกับทำการเพิ่มข้อมูลเข้าไปในขณะที่โปรแกรมกำลังทำการประมวลผลค้นหากฎความสัมพันธ์โดยทำการทดสอบกับข้อมูลทั้งหมด และข้อมูลสุ่ม

5.1 สรุปผลการวิจัย

ผลการทดสอบสรุปได้ดังนี้

5.1.1 เมื่อทำการทดสอบโปรแกรม SASS โดยทำการประมวลผลในการค้นหากฎความสัมพันธ์ของข้อมูลกับชุดข้อมูลที่มีลักษณะของข้อมูลที่มีความตรงกันมาก หรือมีข้อมูลเหมือนกันเกิดขึ้นซ้ำมากๆ หรือมีลักษณะเป็นข้อมูลเชิงธุรกิจ เมื่อกำหนดค่าสนับสนุนขั้นต่ำที่ 80 เปอร์เซ็นต์ และค่าความเชื่อมั่นขั้นต่ำที่ 95 เปอร์เซ็นต์ โปรแกรม SASS สามารถทำการค้นพบกฎความสัมพันธ์ได้ โดยกฎความสัมพันธ์ที่ถูกค้นพบนี้เรียกว่า “กฎความสัมพันธ์ที่แข็งแกร่ง” เมื่อทำการทดสอบกับชุดข้อมูลที่มีขนาดเล็กลงโดยการสุ่มข้อมูลจะพบว่าระยะเวลาที่โปรแกรม SASS ใช้ในการค้นหากฎความสัมพันธ์จะลดลงตามขนาดของข้อมูล แต่กฎความสัมพันธ์ที่ค้นพบได้จากชุดข้อมูลสุ่มยังคงมีความถูกต้องคงเดิมเมื่อนำไปเปรียบเทียบกับกฎความสัมพันธ์ที่ถูกค้นพบชุดข้อมูลทั้งหมด

5.1.2 เมื่อทำการทดสอบโปรแกรม SASS โดยทำการประมวลผลในการค้นหากฎความสัมพันธ์ของข้อมูลกับชุดข้อมูลที่มีลักษณะของข้อมูลแตกต่างกันมาก หรือมีข้อมูลเหมือนกันเกิดขึ้นซ้ำกันไม่มาก เมื่อกำหนดค่าสนับสนุนขั้นต่ำที่ 80 เปอร์เซ็นต์ และค่าความเชื่อมั่นขั้นต่ำที่ 95 เปอร์เซ็นต์ โปรแกรม SASS ไม่สามารถทำการค้นพบกฎความสัมพันธ์ที่แข็งแกร่งได้ แต่เมื่อมีการปรับลดค่าสนับสนุนขั้นต่ำ และค่าความเชื่อมั่นขั้นต่ำลงจนถึงระดับหนึ่ง โปรแกรม SASS ก็จะสามารถค้นพบกฎความสัมพันธ์ได้เช่นเดียวกันแต่ความสัมพันธ์ที่ถูกค้นพบนี้ไม่ถือว่าเป็นกฎความ-สัมพันธ์ที่แข็งแกร่ง เนื่องจากมีค่าสนับสนุนขั้นต่ำ และค่าความเชื่อมั่นขั้นต่ำสูงไม่ถึงเกณฑ์ที่กำหนด และเมื่อทำการทดสอบโปรแกรม SASS กับชุดข้อมูลที่มีขนาดเล็กลงโดยการสุ่มข้อมูลก็จะได้ผลลัพธ์เช่นเดียวกับการประมวลผลค้นหากฎความสัมพันธ์กับชุดข้อมูลที่มีลักษณะของข้อมูลที่มีความตรงกันมาก หรือมีข้อมูลเหมือนกันเกิดขึ้นซ้ำมากๆ คือระยะเวลาที่โปรแกรม SASS ใช้ในการค้นหากฎความสัมพันธ์จะลดลงตามขนาดของข้อมูล และกฎความสัมพันธ์ที่ค้นพบได้จากชุดข้อมูล

สุ่มยังคงมีความถูกต้องคงเดิม เมื่อนำไปเปรียบเทียบกับกฎความสัมพันธ์ที่ถูกค้นพบชุดข้อมูลทั้งหมด

5.1.3 กฎความสัมพันธ์ที่ค้นพบจากชุดข้อมูลสุ่มขนาด 1 เปอร์เซ็นต์เมื่อนำมาเปรียบเทียบความถูกต้องตรงกันกับกฎความสัมพันธ์ที่ถูกค้นพบจากชุดข้อมูลทั้งหมด จะพบว่ากฎความสัมพันธ์มีความถูกต้องตรงกันกับกฎความสัมพันธ์ที่ค้นพบจากชุดข้อมูลทั้งหมด โดยอัตราความถูกต้องของกฎความสัมพันธ์เป็น 100 เปอร์เซ็นต์ แต่การค้นหากฎความสัมพันธ์จากชุดข้อมูลสุ่ม 1 เปอร์เซ็นต์ช่วยให้โปรแกรม SASS สามารถลดระยะเวลาในการค้นหากฎความสัมพันธ์จากข้อมูลได้มากกว่า 95 เปอร์เซ็นต์เมื่อเปรียบเทียบกับระยะเวลาที่โปรแกรม SASS ใช้ในการค้นหากฎความสัมพันธ์จากชุดข้อมูลทั้งหมด

5.1.4 เมื่อทำการทดสอบโปรแกรม SASS โดยทำการค้นหากฎความสัมพันธ์จากข้อมูลทรานแซคชันพร้อมกันกับทำการเพิ่มข้อมูลเข้าไปในขณะที่โปรแกรมกำลังทำการประมวลผลค้นหากฎความสัมพันธ์ เมื่อโปรแกรมสิ้นสุดการประมวลผลค้นหากฎความสัมพันธ์ปรากฏว่ากฎความสัมพันธ์ที่ถูกค้นพบมีความถูกต้องตรงกัน และมีค่าความเชื่อมั่นของกฎเท่ากันกับกฎความสัมพันธ์และค่าความเชื่อมั่นที่ถูกค้นพบจากข้อมูลทรานแซคชัน โดยมีอัตราความถูกต้องตรงกันของกฎความสัมพันธ์เป็น 100 เปอร์เซ็นต์ และเมื่อทำการทดสอบโปรแกรม SASS โดยทำการค้นหากฎความสัมพันธ์จากข้อมูลสุ่มขนาด 50 เปอร์เซ็นต์ และทำการค้นหากฎความสัมพันธ์จากข้อมูลสุ่มขนาด 1 เปอร์เซ็นต์ และทำการเพิ่มข้อมูลเข้าไปในขณะที่โปรแกรมกำลังทำการประมวลผลค้นหากฎความสัมพันธ์ เมื่อโปรแกรมสิ้นสุดการประมวลผลค้นหากฎความสัมพันธ์ปรากฏว่ากฎความสัมพันธ์ที่ถูกค้นพบมีความถูกต้องตรงกัน โดยมีอัตราความถูกต้องตรงกันของกฎความสัมพันธ์เป็น 100 เปอร์เซ็นต์ เช่นเดียวกัน แต่ค่าความเชื่อมั่นของกฎความสัมพันธ์ที่ถูกค้นพบจากข้อมูลสุ่มขนาด 1 เปอร์เซ็นต์อาจมีความคลาดเคลื่อนไปเล็กน้อย

5.2 การประยุกต์งานวิจัย

โปรแกรม SASS ที่ทำการพัฒนาขึ้นสามารถนำมาใช้ในการทำเหมืองข้อมูลทางด้านการค้นหาความสัมพันธ์ของข้อมูล เพื่อใช้ในการเพิ่มประสิทธิภาพในกระบวนการค้นหากฎความสัมพันธ์ ในกรณีที่ทำงานกับฐานข้อมูลขนาดใหญ่และข้อมูลมีปริมาณมาก สามารถใช้โปรแกรม SASS ในการลดขนาดของข้อมูลได้ตามความต้องการของผู้ใช้งาน และโปรแกรม SASS สามารถทำงานได้ในขณะที่มีการเพิ่มข้อมูลทรานแซคชันเข้ามาในระหว่างที่โปรแกรมกำลังทำการประมวลผล โดยโปรแกรม SASS สามารถทำงานได้อย่างมีประสิทธิภาพและค้นหากฎ

ความสัมพันธ์ได้อย่างถูกต้อง โดยสามารถนำเอาโปรแกรม SASS มาประยุกต์ใช้งานได้ในกรณีต่างๆ เช่น

1) เมื่อต้องการค้นหากฎความสัมพันธ์ที่แข็งแกร่งจากฐานข้อมูลขนาดใหญ่ และมีข้อมูลปริมาณมาก เนื่องจากการทำงานกับฐานข้อมูลขนาดใหญ่ และมีข้อมูลปริมาณมากนี้จะต้องใช้ระยะเวลาในการประมวลผลเพื่อค้นหากฎความสัมพันธ์เป็นเวลานานมาก จึงอาจทำให้ไม่สามารถนำความรู้ที่ได้รับมาไปใช้ประโยชน์ได้ทันหากมีระยะเวลาที่จำกัด
2) ในกรณีที่ข้อมูลมีลักษณะของข้อมูลที่มีความตรงกันมาก หรือมีข้อมูลเหมือนกันเกิดขึ้นซ้ำมากๆ โปรแกรม SASS จะกำหนดค่าเริ่มต้นที่เหมาะสำหรับการประมวลผลค้นหากฎความสัมพันธ์ที่เหมาะสมกับข้อมูลประเภทนี้ไว้แล้ว ทำให้การประมวลผลค้นหากฎความสัมพันธ์กับข้อมูลประเภทนี้เป็นไปได้อย่างรวดเร็ว และกฎความสัมพันธ์ที่ถูกค้นพบก็มีความถูกต้องตรงกับกับกฎความสัมพันธ์ที่ถูกค้นพบจากข้อมูลทั้งหมด
3) ในกรณีที่ข้อมูลมีข้อมูลที่เหมือนกันเกิดขึ้นซ้ำกันไม่มาก หรือข้อมูลมีความแตกต่างกันเป็นปริมาณมากสามารถปรับลดค่าสนับสนุนขั้นต่ำ และค่าความเชื่อมั่นขั้นต่ำ เพื่อให้โปรแกรม SASS สามารถทำการค้นหากฎความสัมพันธ์ได้
4) ในกรณีที่ต้องการทราบความสัมพันธ์ของข้อมูลที่ถูกเพิ่มเข้าไปในขณะที่โปรแกรมกำลังทำการประมวลผลค้นหากฎความสัมพันธ์

5.3 ข้อเสนอแนะ

ในการทำเหมืองข้อมูลเพื่อค้นหาความสัมพันธ์ในฐานข้อมูลขนาดใหญ่ การลดขนาดข้อมูลด้วยการสุ่มข้อมูล เป็นเทคนิคสำคัญที่ช่วยลดระยะเวลาในการประมวลผลค้นหากฎความสัมพันธ์ทำให้การค้นหากฎความสัมพันธ์ทำได้รวดเร็วขึ้น โปรแกรม SASS ที่ทำการพัฒนาขึ้นนี้สามารถทำการประมวลผลค้นหาความสัมพันธ์ของข้อมูลบนฐานข้อมูลขนาดใหญ่ได้ โดยโปรแกรม SASS สามารถปรับลดหรือเพิ่มขนาดของข้อมูลที่จะนำมาใช้ในการประมวลผลค้นหากฎความสัมพันธ์ได้ตามความต้องการของผู้ใช้งาน และโปรแกรม SASS สามารถนำข้อมูลที่ถูกบันทึกเข้าไปในขณะที่โปรแกรมกำลังทำการประมวลผลค้นหากฎความสัมพันธ์ไปทำการวิเคราะห์ เพื่อค้นหากฎความสัมพันธ์ได้เช่นกัน ดังนั้นโปรแกรม SASS ที่พัฒนาขึ้นมานี้น่าจะเป็นประโยชน์แก่นักวิจัยท่านอื่นที่สนใจที่จะพัฒนางานทางด้านการค้นหากฎความสัมพันธ์ของข้อมูลต่อไป เช่น

1) ข้อมูลที่นำมาใช้ในการทดสอบการค้นหากฎความสัมพันธ์ด้วยโปรแกรม SASS ทุกชุดถูกนำมาทดสอบภายใต้สมมติฐานที่ว่าข้อมูลเป็นข้อมูลที่มีการกระจายแบบปกติ แต่จากผลการทดสอบการค้นหากฎความสัมพันธ์พบว่าข้อมูลชุด Census Income และ Soybean

มีการค้นพบปริมาณกฎความสัมพันธ์จากข้อมูลสุ่มแตกต่างกันไปในแต่ละรอบของการสุ่มข้อมูล ซึ่งอาจเกิดจากข้อมูลชุดดังกล่าวเป็นข้อมูลที่มีการกระจายที่ไม่ปกติ ดังนั้นการวิจัยโดยการปรับปรุงอัลกอริทึมในการสุ่มข้อมูลให้สามารถทำงานได้ดีทั้งกับข้อมูลที่มีการกระจายแบบปกติและข้อมูลที่มีการกระจายแบบไม่ปกติ จะทำให้การค้นหากฎความสัมพันธ์โดยใช้ข้อมูลสุ่มเป็นไปได้อย่างมีประสิทธิภาพมากยิ่งขึ้น

2) การวิจัยเพื่อเพิ่มประสิทธิภาพของโปรแกรม SASS โดยการปรับปรุงเทคนิคที่ใช้ในการจัดการกับข้อมูลสตรีมมิง เพื่อให้โปรแกรมสามารถจัดการและนำข้อมูลประเภทนี้ไปใช้ประมวลผลในการวิเคราะห์หาความสัมพันธ์ได้ถูกต้องและรวดเร็วขึ้น

3) โปรแกรม SASS ที่ทำการพัฒนาขึ้นนี้ยังมีข้อจำกัดในการใช้งานในการค้นหากฎความสัมพันธ์ คือ สามารถทำการประมวลผลในการค้นหากฎความสัมพันธ์ของข้อมูลกับข้อมูลทรานแซคชันที่จะต้องมีการเชื่อมต่อกันสองตาราง โดยมีส่วนที่เป็นมาสเตอร์ (Master) และส่วนที่เป็นรายระเอียด (Detail) และโปรแกรม SASS รองรับการทำงานกับทรานแซคชันที่มีคีย์ของตารางเพียงตัวเดียวเท่านั้น ดังนั้นการพัฒนาโปรแกรม SASS ให้สามารถทำการประมวลผลการค้นหากฎความสัมพันธ์ของข้อมูลกับทรานแซคชันที่หลากหลายยิ่งขึ้น และสามารถรองรับการประมวลผลค้นหากฎความสัมพันธ์ของข้อมูลกับทรานแซคชันที่มีคีย์ของตารางหลายตัวได้นั้น จะทำให้โปรแกรมทำงานได้อย่างมีประสิทธิภาพมากยิ่งขึ้น

4) การพัฒนาโปรแกรม SASS เพิ่มเติมเพื่อนำไปสู่โปรแกรมที่สามารถทำการประมวลผลในการค้นหาความกฎสัมพันธ์ของข้อมูลภายในระบบงานเดียวกัน ซึ่งข้อมูลภายในระบบงานเดียวกันอาจมีทรานแซคชันที่มีความเกี่ยวข้อง และมีความสัมพันธ์กันอยู่ เนื่องจากในระบบงานใดๆ อาจมีรายการทรานแซคชันที่มีความแตกต่างกันแต่มีความสัมพันธ์กันอยู่ การพัฒนาโปรแกรมเพื่อใช้ในการประมวลผลในการค้นหากฎความสัมพันธ์ของข้อมูลทรานแซคชันภายในระบบเดียวกัน จะทำให้เราสามารถทราบความสัมพันธ์ของข้อมูลในระดับที่มีความซับซ้อนได้ยิ่งขึ้น

5) การพัฒนาโปรแกรม SASS เพิ่มเติม โดยการพัฒนาให้โปรแกรม SASS สามารถแสดงผลของกฎความสัมพันธ์ได้ ในกรณีที่ต้องการทราบกฎความสัมพันธ์ของข้อมูลที่มีค่าสนับสนุนและค่าความเชื่อมั่นของกฎความสัมพันธ์คงเดิม โดยไม่จำเป็นต้องทำการประมวลผลในการนับค่าสนับสนุน และตรวจสอบค่าความเชื่อมั่นของกฎความสัมพันธ์ทุกครั้งเมื่อมีความต้องการทราบกฎความสัมพันธ์ของข้อมูล ซึ่งการลดกระบวนการ

ดังกล่าว จะช่วยทำให้โปรแกรมสามารถลดระยะเวลาในการประมวลผลเพื่อค้นหากฎความสัมพันธ์ได้มากยิ่งขึ้น

รายการอ้างอิง

ฉัตรศิริ ปิยะพิมลสิทธิ์. (2544). **การสุ่มตัวอย่าง (Sampling)** [ออนไลน์]. ได้จาก: http://www.watpon.com/Elearning/res22.htm.

ชัชวาลย์ วงษ์ประเสริฐ. (2549). **เอกสารประกอบคำบรรยาย.** [ออนไลน์]. ได้จาก: http://www.childdept.com.

บุญเสริม กิจศิริกุล. (2546) **รายงานการวิจัยฉบับสมบูรณ์: อัลกอริทึมการทำเหมืองข้อมูล.** กรุงเทพมหานคร: ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย

มยุรี ศรีชัย. (2538). **เทคนิคการสุ่มกลุ่มตัวอย่าง.** จำนวน 1,000 เล่ม. พิมพ์ครั้งที่ 1. กรุงเทพมหานคร: ห้างหุ้นส่วนจำกัด วี.เจ. พริ้นติ้ง 426/57-60 ถนนกำแพงเพชร จตุจักร บางเขน.

สุภาพร ศรีสัตตรัตน์. **เทคนิคการสุ่มตัวอย่างในงานวิจัยนิเทศศาสตร์.** กรุงเทพมหานคร: คณะนิเทศน์ศาสตร์ มหาวิทยาลัยสยาม.

(2549). **วิธีการเลือกกลุ่มตัวอย่าง.** [ออนไลน์]. ได้จาก: http://www.sut.ac.th/e-texts/Social/ProjectCAI/Less2-2.htm.

Agrawal, R., Imielinski, T. and Swami, A. (1993). Mining association rules between set of items in large databases. **Proceedings of ACM SIGMOD International Conference on Management of Data,** 22 (June): 207-216.

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

Agrawal, R., and Srikant, R. (1994). Fast algorithm for mining association rules. **Proceedings of the 20th International Conference on Very Large Data Bases Conference (VLDB'94),** (September): 487-499.

Alon, N., and Spencer, H., J. (1992). The Probabilistic Method. John Wiley Inc., New York.

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

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

Hergerup, T., and Rub., C. (1989/90). A guide tour of Cnernoff Bounds. **Information Processings Letters,** 33:305-308

Mannila, H., Toivonen, H., Verkamo, A., I. (1994). Association rules in sequential data, **Manuscirpt,** (July).

Mannila, H., Toivonen, H., Verkamo, A., I. (1994). Efficient Algorithms for Discovering Association Rules, **Knowledge Discovery in Databases,** (July): 181-192.

Olken, F. (1993). **Random sampling from databases**. Ph.D. Dissertation, University of California at Berkeley, California.

Patchalee Wongkasem, (2004). A two-step algorithm to determine a sample size using random sampling for discovering association rules**.** Ph.D. Dissertation, School of Applied Statistics National Institute of Development Administration.

Savasere, A., Omiecinski, E., and Navathe, S. (1995). An Efficient Algorithm for Mining Association Rules in Large Databases, **Proceedings of the 21st International Conference on Very Large Data Bases,** 432-444.

Shwarz, H. (2004). **Lecture's Document: Data-Warehouse, Data-Mining and OLAP-Technology.** University Stuttgart.

Toivonen, H. (1996). Sampling Large Databases for Association Rules**, The VLDB Jounal,** 134-145.

ภาคผนวก ก

บทความผลงานวิจัยที่นำเสนอในการประชุมวิชาการวิทยาศาสตร์และเทคโนโลยี แห่งประเทศไทย ครั้งที่ 31

การค้นหากฎความสัมพันธ์ที่มีความแข็งแกร่งด้วย SQL

MINING STRONG ASSOCIATION RULES WITH SQL

ลักษมี โขมโนทัย, นิตยา เกิดประสพ และ กิตติศักดิ์ เกิดประสพ

Laksamee Khomnotai, Nittaya Kerdprasop and Kittisak Kerdprasop

Data Engineering and Knowledge Discovery (DEKD) Research Unit, School of Computer Engineering, Suranaree University of Technology, Nakhon Ratchasima, Thailand,
E-mail address: kajchanz@hotmail.com, nittaya@sut.ac.th, kerdpras@sut.ac.th

บทคัดย่อ : กฎความสัมพันธ์ที่มีความแข็งแกร่ง หมายถึง กฎที่มีจำนวนข้อมูลสนับสนุนมาก และมีค่าความถูกต้องสูง การทำเหมืองข้อมูลในลักษณะการค้นหาความสัมพันธ์ภายในกลุ่มข้อมูลมีประโยชน์เพื่อให้ได้กฎที่มีความแข็งแกร่งจะต้องทำการค้นหาจากข้อมูลที่มีปริมาณมาก ต้องมีจำนวนข้อมูลสนับสนุนและมีความถูกต้องสูง เพื่อความน่าเชื่อถือของกฎที่ได้ และสามารถนำผลที่ได้ไปใช้ได้อย่างมีประสิทธิภาพ แต่อุปสรรคในการทำเหมืองข้อมูลกับข้อมูลทรานแซคชันคือปริมาณของข้อมูลที่มีมาก ทำให้การประมวลผลเพื่อค้นหาความสัมพันธ์ภายในข้อมูลต้องใช้เวลามาก งานวิจัยนี้จึงมีจุดมุ่งหมายเพื่อเพิ่มความเร็วในการค้นหากฎความสัมพันธ์ที่มีความแข็งแกร่งจากข้อมูลปริมาณมากด้วยเทคนิคการสุ่มข้อมูลโดยผลลัพธ์ที่ได้จะต้องไม่แตกต่างจากการค้นหากฎจากข้อมูลทั้งหมด

Abstract : Strong association rules are association rules with high support and high confidence. Association rule mining for discovering strong association rules requires a very large dataset to assure high data support and high confidence. Time usage is an important problem in association rule mining for strong rules, especially when mining from large transactional data. We are interested in studying the technique to decrease time usage in association rule from a very large dataset. We use random sampling technique to mine strong association rules from a sampled transactional data with the constraint that the discovered rules have to be identical to these discovered from the original large transactions.

Introduction : Strong association rule is a rule with high data support and high confidence (or accuracy). For example, discovering association rule from a conveniencet store's transactional data in a previous month. The discovered strong association rule is

"a customer who buys bread also buys jam"
(data support = 80 percents, confidence = 100 percents)

The rule is reliable and can be applieds to support decision making on product pricing, promotion, store layout and so on.

Methodology : We perform strong association rule mining from transactional data using the Apriori algorithm implemented in SQL statements [1, 2]. In our experiment, we use Oracle 10g database, tested on Pentium IV 1.4 GHz with RAM 768 MB machine. The data sets used in our experiment are synthetic data (Conveniencet store) and data taken from the UCI Repository (http://www.ics.uci.edu/~mlearn/MLRepository.html), (Census Income, Mushroom and Soybean). The number of instances in all four data sets are not less than 60,000 instances. We test with the Apriori algorithm by assigning minimum support as 80% and minimum confidence as 95%. The tests are performed on the whole data sets and on sample data with sampling proportions 1%, 10%, 20%, 30%, 40% and 50% (sampling algorithm is shown in Figure1). Mined association rules, as well as mining time, have been compared among different sample proportions of the data against the result obtained from the original data.

```
open h_dummy for v_sl_head;
loop
  fetch h_dummy into v_head_col;
  exit when h_dummy%notfound;
  if dbms_random.value (0, 100)<= p_pc_rand then
     begin
       insert into tran_temp (tid)
       values (v_head_col);
     end;
     v_tot_tran := v_tot_tran + 1;
  end if;
end loop;
```

Figure 1 Sampling from transaction

Results : In our experiment, we compared the association rules discovered from the original data with the rules discovered from the sampled data of various sizes. The rules compared have to be perfectly matched. The results in Table 1 confirm that all sample data sets can produce the same association rules as the original data set. Table 2 shows the running time for discovering the association rule. When we decrease the sample size, the discovery time will be significantly decreased.

Table 1 Association rule mining from the whole population compared against rule mining from sample data. Number appeared in each cell is the number of discovered rules.

Data Set	Original data	Sampling proportion					
		50%	40%	30%	20%	10%	1%
Conveniencet store	2	2	2	2	2	2	2
Census Income	6	6	6	6	6	6	6
Mushroom	32	32	32	32	32	32	32
Soybean	10	10	10	10	7	7	10

Table 2 Mining time (seconds) of each data set.

Data Set	Original data	Sampling proportion					
		50%	40%	30%	20%	10%	1%
Conveniencet store	337	165	135	101	66	37	4
Census Income	2343	1492	1331	1050	838	503	76
Mushroom	1528	977	902	758	555	358	22
Soybean	3932	2890	2507	2178	1623	978	128

Conclusion : The objective of our study is to investigate the property of strong association rules. We define strong association rules as a set of rules discovered from a large amount of transactional data with frequent items co-occuring not less than 80 percent of the time and the rule's accuracy has to be at least 95 percent correct. With such definition of association rules, we have found that a 1% random sample of the data can yield exactly the same set of strong rules as those obtained from the large original transaction data set. By means of sampling, we can save as much as 90% of association mining time.

Acknowledgement : This research is funded supported by DEKD Research Unit. DEKD Research Unit is fully supported by Suranaree Univesity of Technology.

References: [1]. S. Sarawagi, S. Thomas, R. Agrawal, Integrating association rule mining with relational database. *Proc. of the ACM SIGMOD Conference on Management of Data*, 1998, pp 343-354.

[2]. R Agrawal, T. Imielinski, A. Swami, Mining association rules between set of items in large databases. *Proc. of the ACM SIGMOD Conference on Management of Data*, 1993, pp.207-216.

Keywords : Association rule, Sampling, Apriori.

ภาคผนวก ข

รหัสต้นฉบับ ของโปรแกรม SASS

1. โพรซิเยอร์ APRIORI

```
CREATE OR REPLACE PROCEDURE apriori (p_hcol varchar2,
                                     p_htab varchar2,
                                     p_dslcol varchar2,
                                     p_dtab varchar2,
                                     p_dkcol varchar2,
                                     p_min_sup number,
                                     p_min_conf number,
                                     p_pc_rand number) IS

    type c_refcur is ref cursor;

    cursor tran (p_stream_status varchar2) is
                 select t.tid, rowtocol (p_dslcol, p_dtab, p_dkcol,
                        t.tid, p_stream_status) tran_item,
                        t.stream_status
                 from   tran_temp t
                 where  stream_status = p_stream_status
                 order by t.tid;

    v_tot_tran      number(10)     := 0; -- total of transaction
    v_tran_item     varchar2(4000);
    v_can_seq       number(10)     := 1;
    v_status        varchar2(10)   := 'TRUE';
    v_freq_seq      varchar2(50);
    v_rand_status   varchar2(1)    := 'Y';
    v_stream_status varchar2(1)    := 'N'; -- stream and cnt
    v_cnt_stream    number         ; --  store count in stream's
                                         detail
    v_stream_fitem  varchar2(4000) ; --  store first item in stream's
                                         detail
    v_sl_head       varchar2(4000) := 'select '||p_hcol||
                                      ' from '||p_htab;
    v_sl_det        varchar2(4000) := 'select d.'||p_dslcol||
                                      ' from tran_temp t, '||p_dtab||
                                      ' d where d.'||p_dkcol||
                                      ' = t.tid and stream_status =
                                      ''N''';
    v_head_col      varchar2(4000); --   for store head's key
    v_det_item      varchar2(4000); --   for store dets key use in
                                         first candidate
    h_dummy         c_refcur; -- for select head
    d_dummy         c_refcur; -- select detail use in first candidate
    v_start         date := sysdate;
    v_assoc         date;
    v_rand          date;
    v_gen           date;
    v_end           date;
  BEGIN
    delete from tran_temp;
    delete from can_set;
    delete from freq_set;
```

```
delete from assoc_rule;
commit;

/** random transaction **/
begin
  if p_pc_rand != 100 then
     v_rand_status := 'Y';
  else
     v_rand_status := 'N';
  end if;

  begin
    update mining_control
    set    mining_status = 'Y',
           random_status = v_rand_status;
    if sql%notfound then
       begin
         insert into mining_control (mining_status,
                                     random_status)
         values ('Y', v_rand_status);
       end;
    end if;
  end;
  commit;

  open h_dummy for v_sl_head;
  loop
    fetch h_dummy into v_head_col;
    exit when h_dummy%notfound;
    if dbms_random.value (0, 100) <= p_pc_rand then
       begin
         insert into tran_temp (tid, stream_status)
         values (v_head_col, 'N');
               exception when dup_val_on_index then null;
       end;
       v_tot_tran := v_tot_tran + 1; -- count all transaction
    end if;
  end loop;
  close h_dummy;
  commit;
  exception when others then dbms_output.put_line (sqlerrm);
  -- print err message
            if h_dummy%isopen then
               close h_dummy;
            end if;
end;

  v_rand := sysdate;

-- read from transaction

v_stream_status := 'N'; -- set for read from transaction
while v_status = 'TRUE' loop
  if v_can_seq != 1 then
     for t in tran (v_stream_status) loop
         item.find_candidate ( v_can_seq, v_status, t.tran_item,
                               p_min_sup);
     end loop; -- end tran loop
  else
     begin
       open d_dummy for v_sl_det;
       loop
         fetch d_dummy into v_det_item;
         exit when d_dummy%notfound;
         item.gen_candidate (v_can_seq, v_det_item);
        -- generate first candidate
```

```
        end loop; -- for generate first candidate
        close d_dummy;
        exception when others then dbms_output.put_line (sqlerrm);
        -- print err message
                  if d_dummy%isopen then
                     close d_dummy;
                  end if;
      end;
   end if; -- end chk candidate sequence
   -- generate frequence item set
   item.gen_frequence (v_can_seq, v_tot_tran, p_min_sup);
   v_can_seq := v_can_seq + 1; -- plus candidate sequence
   commit;
 end loop;  -- end v_status loop

   -- read from stream
 v_stream_status := 'Y';    -- set for read from stream
 for t in tran (v_stream_status) loop
     v_status     := 'TRUE'; -- reset status
     v_can_seq    := 1;      -- reset sequence
     v_cnt_stream := to_number(substr(t.tran_item,
                     instr(t.tran_item, '@')+1));
     while v_status = 'TRUE' loop
           if v_can_seq != 1 then
              item.stream_candidate (v_can_seq, v_status,
              t.tran_item, p_min_sup);
           else
              for i in 1..to_number(substr(t.tran_item,
                          instr(t.tran_item, '@')+1)) loop
                  v_stream_fitem := item.find_chk_pos(i,
                               t.tran_item); -- find single item
                  item.gen_candidate (v_can_seq, v_stream_fitem);
                  -- generate first stream candidate
                  item.stream_frequence (v_can_seq,
                                         v_stream_fitem);
              end loop;
           end if;
           v_can_seq := v_can_seq + 1; -- plus candidate sequence
           commit;
     end loop;
 end loop;

   v_gen := sysdate;

begin
  update mining_control
  set    mining_status = 'N',
         random_status = 'N';
end;
commit;
 /** generate association rules **/
 rule.gen_rules (p_min_conf);
 /** clear candidate-frequence **/
v_end := sysdate;


begin
  select freq_seq
  into   v_freq_seq
  from   freq_set
  where  rownum = 1;
  exception when no_data_found then
  dbms_output.put_line ('No Association rule');
end;

dbms_output.put_line ('start at : '||
```

```
                to_char(v_start, 'dd/mm/rrrr hh24:mi:ss')||
                ' end at : '||
                to_char(v_rand, 'dd/mm/rrrr hh24:mi:ss')||
                ' time usage for sampling : '||
                trunc(((v_rand-v_start)*100000), 2)||' secs');

   dbms_output.put_line ('start at : '||
                to_char(v_start, 'dd/mm/rrrr hh24:mi:ss')||
                ' end at : '||
                to_char(v_gen, 'dd/mm/rrrr hh24:mi:ss')||
                ' time usage for gen item : '||
                trunc(((v_gen-v_rand)*100000), 2)||' secs');

   dbms_output.put_line ('start at : '||
                to_char(v_start, 'dd/mm/rrrr hh24:mi:ss')||
                ' end at : '||
                to_char(v_end, 'dd/mm/rrrr hh24:mi:ss')||
                ' time usage for assoc : '||
                trunc(((v_end-v_rand)*100000), 2)||' secs');

   dbms_output.put_line ('start at : '||
                to_char(v_start, 'dd/mm/rrrr hh24:mi:ss')||
                ' end at : '||
                to_char(v_end, 'dd/mm/rrrr hh24:mi:ss')||
                ' time usage : '||
                trunc(((v_end-v_start)*100000), 2)||' secs');

  END apriori;
  --
/
```

2. ฟังก์ชันที่ใช้ในการแปลงรูปแบบข้อมูลจากทรานแซคชันหลายเรคคอร์ดเป็นหนึ่งเรคคอร์ด

```
CREATE OR REPLACE FUNCTION rowtocol (p_sel_col in varchar2,
                                     p_table in varchar2,
                                     p_key_col in varchar2,
                                     p_con_col in varchar2,
                                     p_cnt_status in varchar2
                                     default 'N',
                                     p_dlmtr in varchar2 default ',')
                                RETURN VARCHAR2 AUTHID CURRENT_USER IS

  TYPE c_refcur IS REF CURSOR;
  p_slct VARCHAR2(4000):= 'select '||p_sel_col||' item from
                          '||p_table||' where '||p_key_col||
                          ' = '''||p_con_col||''' order by item';
  lc_str VARCHAR2(4000);
  lc_colval VARCHAR2(4000);
  c_dummy c_refcur;
  v_cnt number := 0;
  l number;

BEGIN
  --  open c_dummy for
  OPEN c_dummy FOR p_slct;
  LOOP
     FETCH c_dummy INTO lc_colval;
     EXIT WHEN c_dummy%NOTFOUND;
     lc_str := lc_str || p_dlmtr || lc_colval;
     v_cnt  := v_cnt + 1;
  END LOOP;
  CLOSE c_dummy;
  if p_cnt_status = 'Y' then
     RETURN SUBSTR(lc_str,2)||'@'||to_char(v_cnt);
  else
     RETURN SUBSTR(lc_str,2);
  end if;
  EXCEPTION
     WHEN OTHERS THEN
     lc_str := SQLERRM;
     IF c_dummy%ISOPEN THEN
        CLOSE c_dummy;
     END IF;
  RETURN lc_str;
END;
/
```

3. แพ็กเกจที่ใช้ในการสร้างแคนดิเดทไอเท็มเซต การนับค่าสนับสนุน และสร้างไอเท็มเซตที่ปรากฏบ่อย

```
CREATE OR REPLACE PACKAGE item IS
  FUNCTION chk_match (p_status varchar2, p_chk_item varchar2,
                      p_tran_item varchar2) return varchar2 IS
  FUNCTION find_chk_pos (p_seq number, p_set varchar2) return
           varchar2;
  PROCEDURE find_candidate ( p_can_seq in out number,
                             p_status in out varchar2,
                             p_tran_item varchar2, p_min_sup number);
  PROCEDURE find_candidate (p_can_seq in out number,
                            p_status in out varchar2,
                            p_tran_item varchar2, p_min_sup number);
 PROCEDURE gen_candidate (p_can_seq number, p_item_set varchar2);
  PROCEDURE gen_frequence (p_freq_seq number, p_tot_rec number,
                           p_min_sup number);
  PROCEDURE stream_candidate (p_can_seq in out number,
                              p_status in out varchar2,
                              p_tran_item varchar2,
                              p_min_sup number);
  PROCEDURE stream_frequence ( p_can_seq in out number,
                               p_item_set varchar2);
END;
/


CREATE OR REPLACE PACKAGE BODY item IS

  FUNCTION chk_match (p_status varchar2, p_chk_item varchar2,
                      p_tran_item varchar2) return varchar2 IS
  /* for chk item_set are in transaction */
  BEGIN
    if instr(p_tran_item, p_chk_item) > 0 and p_status = 'TRUE' then
       return 'TRUE';
    else
       return 'FALSE';
    end if;
  END chk_match;

  FUNCTION find_chk_pos (p_seq number, p_set varchar2) return
           varchar2 IS

  /* find item for chk in transaction */
    v_last  number(10);
    v_first number(10);
  BEGIN
   if p_seq = 1 then
      v_first := 1;
      v_last := instr(p_set, ',', 1, p_seq)-1;
   else
      v_first   := instr(p_set, ',', 1, p_seq-1) + 1;
      if instr(p_set, ',', 1, p_seq) > 0 then
         v_last := (instr(p_set, ',', 1, p_seq) -
                    instr(p_set, ',', 1, p_seq - 1))-1;
      else
         v_last := length(p_set) - instr(p_set, ',', -1, 1);
      end if;
   end if;
   return substr(p_set, v_first, v_last);
 END find_chk_pos;
```

```
  PROCEDURE find_candidate (p_can_seq in out number,
                            p_status in out varchar2,
                            p_tran_item varchar2, p_min_sup number)IS

    cursor set_l is
               select item_set, substr(item_set, (instr(item_set,
                                        ',', -1, 1)+1)) l_set
               from   freq_set
               where  freq_seq = 'F'||(p_can_seq-1)
               order by l_set;

    cursor set_r (p_lset varchar2) is
               select item_set r_set
               from   freq_set
               where  freq_seq = 'F1'
               and    p_lset < item_set
               order by r_set;

    v_chk      varchar2(4000);
    v_status   varchar2(10);
    v_tmp      varchar2(32767);
  BEGIN
    p_status := 'FALSE';
    for l in set_l loop
        for r in set_r (l.l_set) loop
            v_status := 'TRUE'; -- initial v_status for chk member
              for i in 1.. p_can_seq loop -- for chk member of set
                v_chk    := item.find_chk_pos (i, l.item_set||','||
                                               r.r_set);
                v_status := item.chk_match (v_status, v_chk,
                                            p_tran_item);
                p_status := 'TRUE';
            end loop; -- end loop chk member
            if v_status = 'TRUE' then -- chk item_set match with
                                         transaction

               -- update support in candidate
               item.gen_candidate (p_can_seq, l.item_set||','||
                                   r.r_set);
            end if; -- end if chk status
        end loop;
    end loop;
--    commit;
  END find_candidate;

  PROCEDURE gen_candidate (p_can_seq number, p_item_set varchar2) IS
  BEGIN
    update can_set
    set    sup_cnt = nvl(sup_cnt, 0)+1
    where  item_set = p_item_set;
    if sql%notfound then
       begin
         insert into can_set
         values ('C'||p_can_seq, p_item_set, 1);
       end;
    end if;
  END gen_candidate;

  PROCEDURE gen_frequence (p_freq_seq number, p_tot_rec number,
                           p_min_sup number) IS

  BEGIN
    begin
      insert into freq_set
      select 'F'||to_char(p_freq_seq), item_set, sup_cnt
```

```
      from   can_set
      where  can_seq = 'C'||to_char(p_freq_seq)
      and    (sup_cnt*100)/p_tot_rec >= p_min_sup;
    end;
  END gen_frequence;

  PROCEDURE stream_candidate (p_can_seq in out number,
                              p_status in out varchar2,
                              p_tran_item varchar2,
                              p_min_sup number) IS

    cursor s_cand is
               select item_set
               from   freq_set
               where  freq_seq = 'F'||(p_can_seq);

    v_chk      varchar2(4000);
    v_status   varchar2(10);
    v_tmp      varchar2(32767);
  BEGIN
    p_status := 'FALSE';
    for s in s_cand loop
        v_status := 'TRUE'; -- initial v_status for chk member
          for i in 1.. p_can_seq loop -- for chk member of set
            v_chk    := item.find_chk_pos (i, s.item_set);
            v_status := item.chk_match (v_status, v_chk,
                                        p_tran_item);
            p_status := 'TRUE';
        end loop; -- end loop chk member
        if v_status = 'TRUE' then -- chk item_set match with
                                  transaction

               -- update support in candidate
           item.gen_candidate (p_can_seq, s.item_set);
           item.stream_frequence (p_can_seq, s.item_set);
        end if; -- end if chk status
    end loop;
--    commit;
  END stream_candidate;

  PROCEDURE stream_frequence (p_can_seq in out number,
                              p_item_set varchar2) IS

  BEGIN
    begin
      update freq_set
      set    sup_cnt = nvl(sup_cnt, 0)+1
      where  item_set = p_item_set;
      if sql%notfound then
         begin
           insert into freq_set
           values ('F'||p_can_seq, p_item_set, 1);
         end;
      end if;
    end;
    commit;
  END stream_frequence;
END item;
/
```

4. แพ็กเกจที่ใช้ในการสร้างกฎความสัมพันธ์

```
CREATE OR REPLACE PACKAGE rule AS
  FUNCTION find_chk_pos (p_seq number, p_set varchar2,
                         p_status in out varchar2) return varchar2;
  FUNCTION find_sup_cnt (p_item varchar2) return number;
  PROCEDURE gen_sub_rule (p_start number, p_freq_seq number,
                          p_set varchar2, p_front varchar2,
                          p_back varchar2, p_min_conf number,
                          p_status in out varchar2,
                          p_rule_seq in out number);
  PROCEDURE gen_rules (p_min_conf number);
  PROCEDURE ins_rules (p_iset varchar2, p_rule_seq in out number,
                       p_rule_if varchar2, p_rule_then varchar2,
                       p_min_conf number);
END;
/

CREATE OR REPLACE PACKAGE BODY rule AS
  FUNCTION find_chk_pos (p_seq number, p_set varchar2,
                         p_status in out varchar2) return varchar2 IS
    v_fpos number(10)   := 0;
    v_bpos number(10)   := 0;
  BEGIN
    if p_seq = 1 then
       v_fpos := 1;
    else
       v_fpos := instr(p_set, ',', 1, p_seq - 1) + 1;
    end if;

    if instr(p_set, ',', 1, p_seq) > 0 then
       v_bpos   := instr(p_set, ',', 1, p_seq) - v_fpos;
       p_status := 'TRUE';
    else
       v_bpos   := length (p_set) - instr(p_set, ',', 1, p_seq);
       p_status := 'FALSE';
    end if;
    return substr(p_set, v_fpos, v_bpos);
return null;

  END find_chk_pos;

  FUNCTION find_sup_cnt (p_item varchar2) return number IS
    v_cnt number(10);
  BEGIN
    select sup_cnt
    into   v_cnt
    from   can_set
    where  item_set = p_item;
    return v_cnt;
    exception when no_data_found then return 1;
  END find_sup_cnt;



  PROCEDURE gen_sub_rule (p_start number, p_freq_seq number,
                          p_set varchar2, p_front varchar2,
                          p_back varchar2, p_min_conf number,
                          p_status in out varchar2,
                          p_rule_seq in out number) IS
    v_status varchar2(5) := 'TRUE';
    v_front  varchar2(4000) ;
```

```
  v_back   varchar2(4000) := '^';
  v_fcnt   number(10); -- front set's sup_cnt
  v_icnt   number(10); -- item_set's sup_cnt
  v_conf   number(10,2);
BEGIN
  for j in (p_start + 1) .. (p_freq_seq) loop
      if p_start > p_freq_seq then
         exit;
      end if;
      if j > (p_start + 1) then
         rule.gen_sub_rule (j-1, p_freq_seq, p_set, v_front,
                            v_back, p_min_conf, p_status,
                            p_rule_seq);
      end if;

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

      if v_front != replace(p_back, ','||v_front) then
         if v_status = 'TRUE' then
            v_back  := replace(p_back, v_front||',');
         else
            v_back  := replace(p_back, ','||v_front);
         end if;
      else
         exit;
      end if;

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

PROCEDURE gen_rules (p_min_conf number) IS
  cursor freq is select freq_seq, item_set, sup_cnt
                 from   freq_set
                 where  freq_seq > 'F1'
                 order by freq_seq;
  v_front    varchar2(4000);
  v_back     varchar2(4000);
  v_status   varchar2(10) := 'TRUE';
  v_icnt     number(10); -- item_set's sup_cnt
  v_fcnt     number(10); -- front set's sup_cnt
  v_conf     number(10,2);
  v_freq_seq number(10) := 0;
  v_rule_seq number(10) := 0;
BEGIN
  for f in freq loop
      v_freq_seq := to_number(replace(f.freq_seq, 'F', ''));
      for i in 1..v_freq_seq loop
          v_front := rule.find_chk_pos (i, f.item_set, v_status);
          if v_status = 'TRUE' then
             v_back  := replace(f.item_set, v_front||',');
          else
             v_back  := replace(f.item_set, ','||v_front);
          end if;

          rule.ins_rules (f.item_set, v_rule_seq, v_front, v_back,
                          (p_min_conf/100));
```

```
            rule.gen_sub_rule (i, v_freq_seq, f.item_set, v_front,
                               v_back, p_min_conf, v_status,
                               v_rule_seq);
        end loop;
    end loop;
  END gen_rules;

  PROCEDURE ins_rules (p_iset varchar2, p_rule_seq in out number,
                       p_rule_if varchar2, p_rule_then varchar2,
                       p_min_conf number) IS
    v_if_cnt number := 0;
    v_cnt    number := 0;
    v_conf   number := 0;
  BEGIN
    v_if_cnt := rule.find_sup_cnt(p_rule_if);
    v_cnt    := rule.find_sup_cnt(p_iset);
    v_conf   := round(nvl(v_cnt, 0)/nvl(v_if_cnt, 0), 2);

    if v_conf >= nvl(p_min_conf, 0) then
       p_rule_seq := nvl(p_rule_seq, 0) + 1;
       dbms_output.put_line (p_rule_seq||' : '||p_rule_if||
                             ' = '||to_char(v_if_cnt)||' -> '||
                             p_rule_then||' = '||to_char(v_cnt)||
                             ' conf : '||to_char (v_conf));
       begin
         insert into assoc_rule (rule_conf, rule_if_cnt, rule_cnt,
                                 rule_if, rule_then)
         values (v_conf, v_if_cnt, v_cnt, p_rule_if, p_rule_then);
       end;
       commit;
    end if;
  END ins_rules;
END;
/
```

5. ส่วนของดาต้าเบสทริกเกอร์ที่ใช้ในการจัดการกับข้อมูลสตรีมมิง

```
CREATE OR REPLACE TRIGGER TABLE_AFT_INS_ROW
AFTER INSERT
ON USER.TABLE_NAME
REFERENCING NEW AS NEW OLD AS OLD
FOR EACH ROW
declare
  v_rand          number := 0;
  v_tcnt          number := 0;
  v_tid           varchar2(20) := null;
  v_newtid        varchar2(20) := :new.tid;
  v_rand_status   varchar2(1)  := 'Y';
  v_mining_status varchar2(1)  := 'Y';
BEGIN
  /** check random status **/
  begin
    select mining_status, random_status
    into   v_mining_status, v_rand_status
    from   mining_control;
    exception when no_data_found then v_rand_status   := null;
                                      v_mining_status := null;
  end;

  if v_mining_status = 'Y' then
     if v_rand_status = 'Y' then
        begin
          select dbms_random.value (0, 100)
          into   v_rand
          from   dual;
        end;

        if v_rand > 50 then
           begin
             insert into tran_temp (tid, stream_status)
             values (v_newtid, 'Y');
           end;

           begin
             select dbms_random.value (0, 100)
             into   v_rand
             from   dual;
           end;

           if v_rand > 50 then
              begin
                select count(tid)
                into   v_tcnt
                from   tran_temp;
                exception when no_data_found then v_tcnt := 0;
              end;

              begin
                select tid
                into   v_tid
                from   (select rownum rnum, tid
                         from   tran_temp)
                where  rnum = round(dbms_random.value(1, (v_tcnt-1)))
                         and rownum = 1;
                exception when no_data_found then v_tid := null;
              end;

              /** find tid from tran_temp for delete **/
              begin
```

```
                delete tran_temp
                where  tid = v_tid;
                exception when no_data_found then null;
              end;
            end if;
         end if;
      elsif v_rand_status = 'N' then
        begin
          insert into tran_temp (tid, stream_status)
          values (v_newtid, 'Y');
        end;
      end if;
  end if;
END;
/
```

ภาคผนวก ค

วิธีการใช้งานโปรแกรม SASS

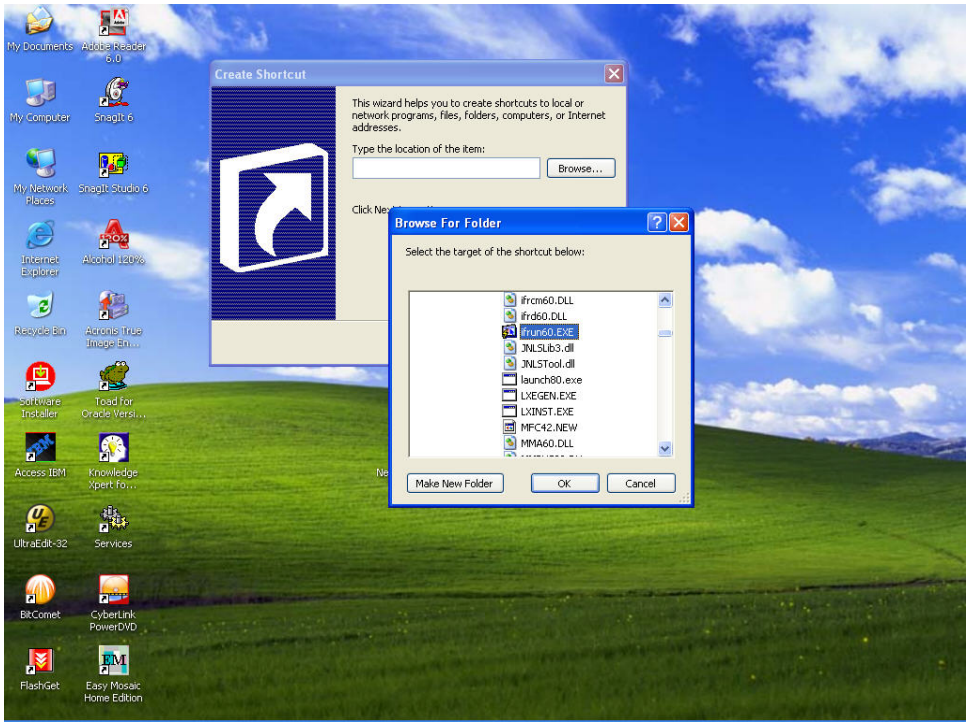
1. การใช้โปรแกรม SASS ในรูปแบบโปรแกรมสำเร็จรูปที่ทำการพัฒนาขึ้นในการประมวลผลค้นหากฎความสัมพันธ์ของข้อมูลภายในฐานข้อมูล

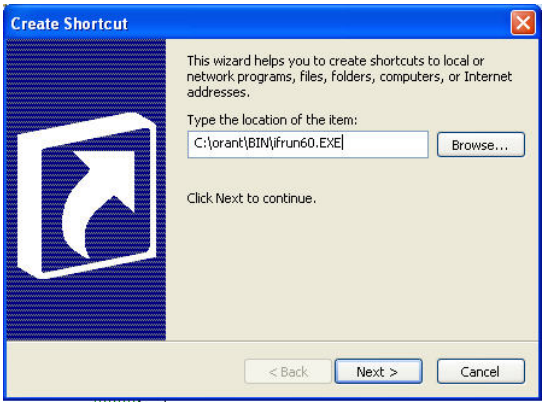
โปรแกรม SASS ในรูปแบบโปรแกรมสำเร็จรูปที่ทำการพัฒนาขึ้นนั้นทำการพัฒนาขึ้นด้วย Oracle Developer 6i โดยใช้ Form Builder ดังนั้นก่อนที่จะใช้งานโปรแกรมตัวนี้จะต้องทำการติดตั้ง Oracle Developer 6i ภายในเครื่องที่จะใช้งานเสียก่อน โดยถ้าไม่ต้องการพัฒนาเพิ่มเติมอาจจะทำการติดตั้งเฉพาะในส่วนของ Form Runtime เพื่อใช้งานเพียงอย่างเดียวก็ได้ หลังจากทำการติดตั้ง Oracle Developer 6i เสร็จแล้ว ให้ทำการจัดเตรียมเพื่อใช้งานโปรแกรมดังนี้

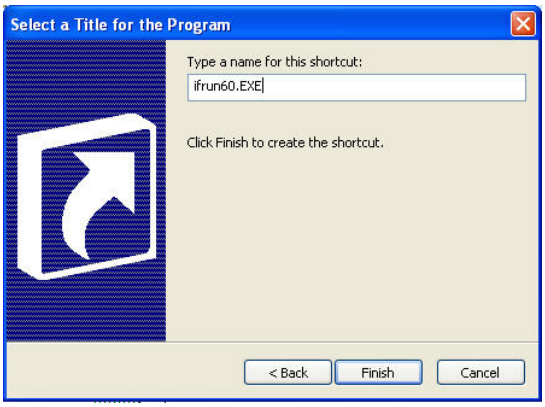
1.1 ทำการสร้างชอร์ทคัทเพื่อเรียกใช้โปรแกรมดังรูปที่ ค.1

รูปที่ ค.1 วิธีการสร้างชอร์ทคัท

1.2 เลือกสร้างชอร์ทคัทของ Form Runtime โดยเลือกไปที่โฟลเดอร์ที่ทำการติดตั้ง Oracle Developer 6i ไว้ ในที่นี้ติดตั้งไว้ที่ C:\orant จากนั้นเลือกโฟลเดอร์ BIN เลือก ifrun60.EXE ดังรูปที่ ค.2

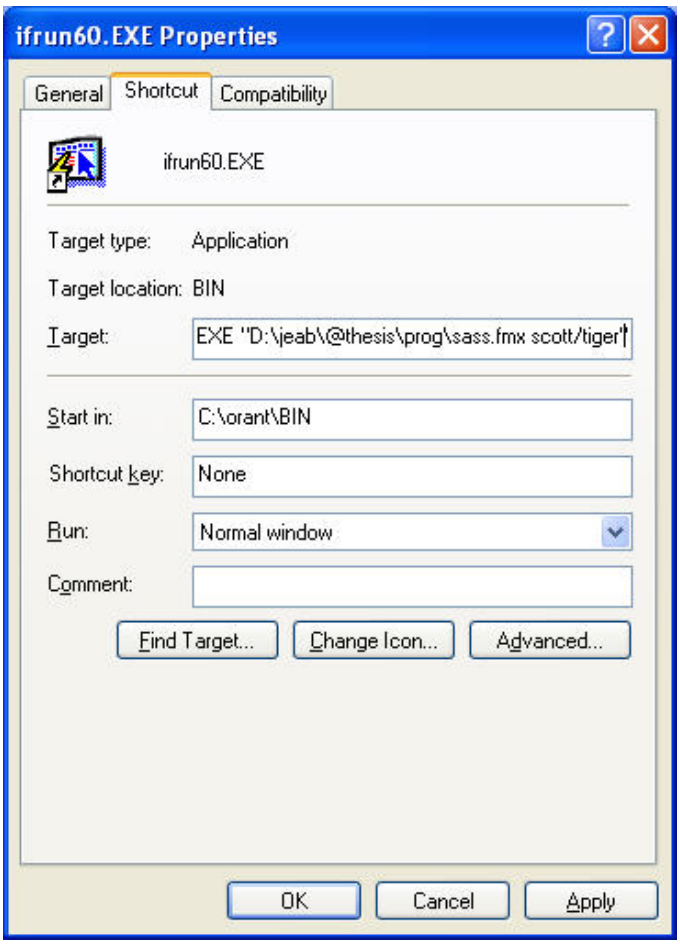






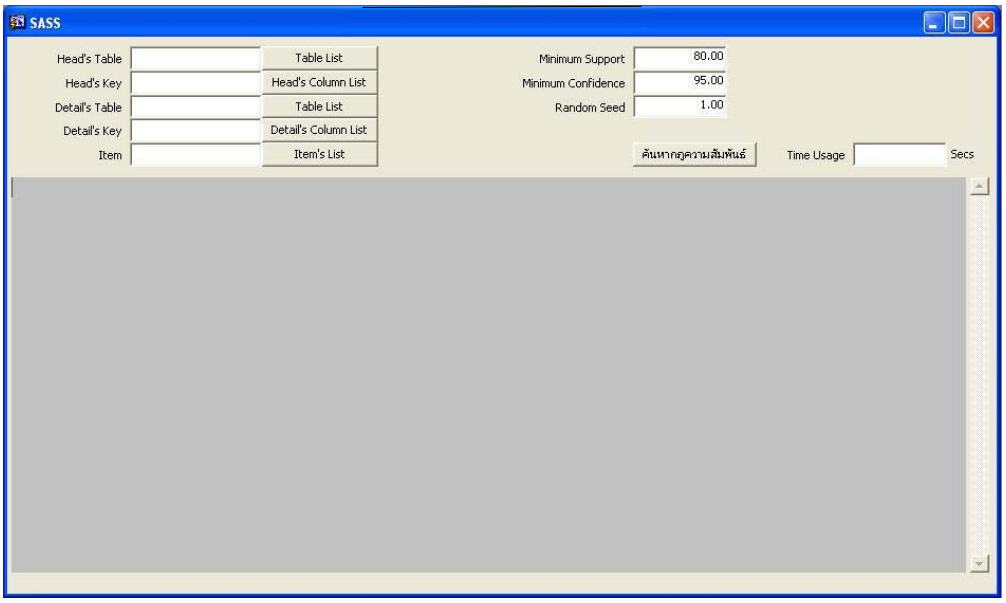
รูปที่ ค.2 วิธีการสร้างชอร์ทคัทของ Form Runtime

1.3 หลังจากที่ได้ชอร์ทคัท ifrun60.EXE แล้วให้ทำการระบุโฟลเดอร์ที่เก็บโปรแกรม SASS เอาไว้ ในที่นี้เก็บไว้ที่ “D:\jeab\@thesis\prog” ตามด้วยไฟล์ที่เป็นตัวรันโปรแกรมคือไฟล์นามสกุล fmx จากนั้นเว้นหนึ่งวรรค แล้วทำการระบุ ยูสเซอร์เนม (Username) และรหัสผ่าน (Password) ของฐานข้อมูลที่จะทำการเชื่อมต่อเพื่อใช้งานดังแสดงในรูปที่ ค.3



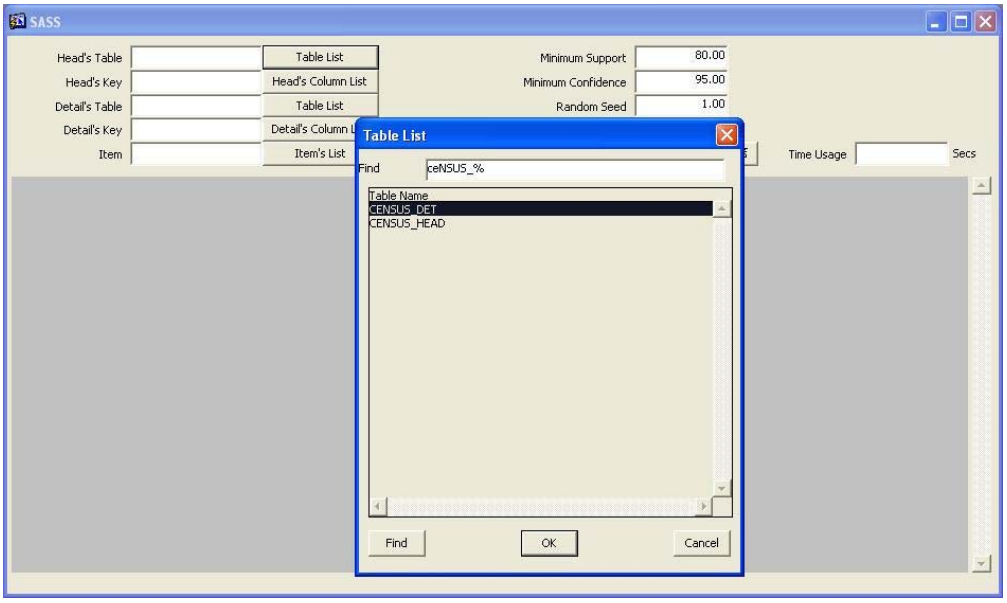
รูปที่ ค.3 วิธีการกำหนดตัวรันโปรแกรม SASS

1.4 จากนั้นเมื่อทำการดับเบิ้ลคลิกที่ชอร์ทคัทโปรแกรมจะเข้าสู่สภาวะพร้อมทำงาน โดยโปรแกรมจะทำการกำหนดค่าสนับสนุนขั้นต่ำ ค่าความเชื่อมั่นขั้นต่ำ และขนาดของข้อมูลสุ่มที่สามารถใช้เป็นตัวแทนที่ดีของข้อมูลทั้งหมดได้ โดยค่าทั้งหมดสามารถเปลี่ยนแปลงได้ ดังแสดงในรูปที่ ค.4



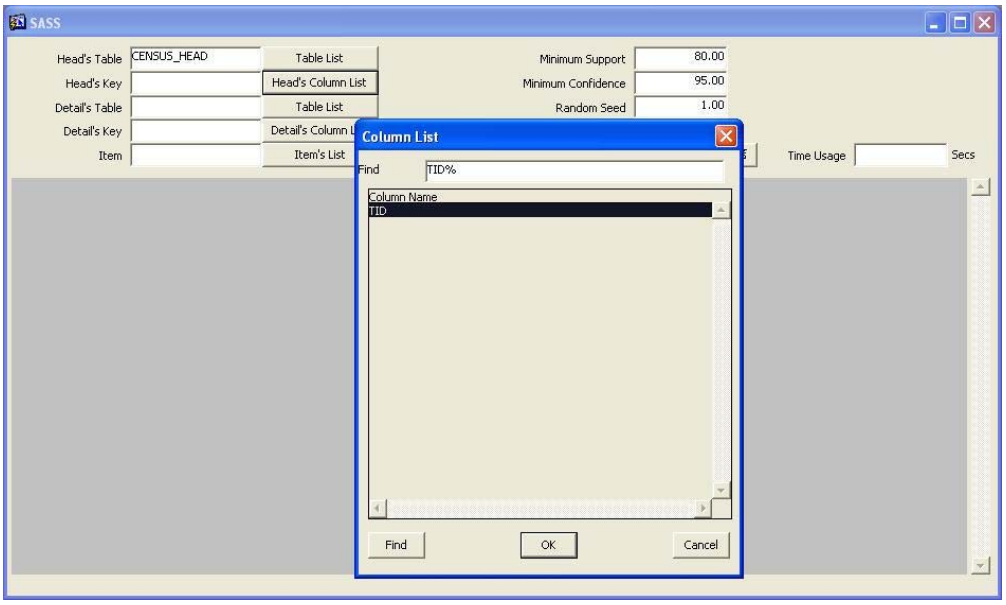
รูปที่ ค.4 โปรแกรม SASS เข้าสู่สภาวะพร้อมทำงาน

1.5 ทำการระบุตารางที่เป็นทรานแซคชันในส่วนของมาสเตอร์ หรือโดยการคลิกปุ่ม Table List ที่อยู่ด้านข้างเพื่อดูรายชื่อของตารางทั้งหมดที่มีอยู่ในฐานข้อมูล ดังแสดงในรูปที่ ค.5



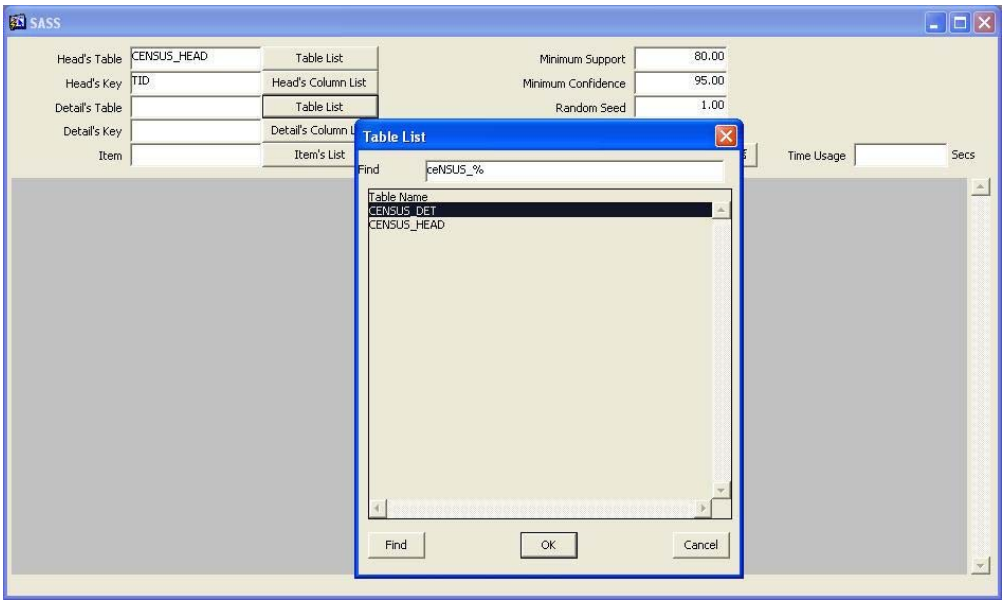
รูปที่ ค.5 วิธีการระบุตารางที่เป็นทรานแซคชันในส่วนของมาสเตอร์

1.6 ทำการระบุคีย์ของตารางทรานแซคชันในส่วนของมาสเตอร์ หรือโดยการคลิกปุ่ม Head's Column List ที่อยู่ด้านข้างเพื่อดูรายชื่อของคอลัมน์ที่มีอยู่ในตารางทรานแซคชันในส่วนของมาสเตอร์ทั้งหมดที่ระบุไว้ข้างต้น ดังแสดงในรูปที่ ค.6



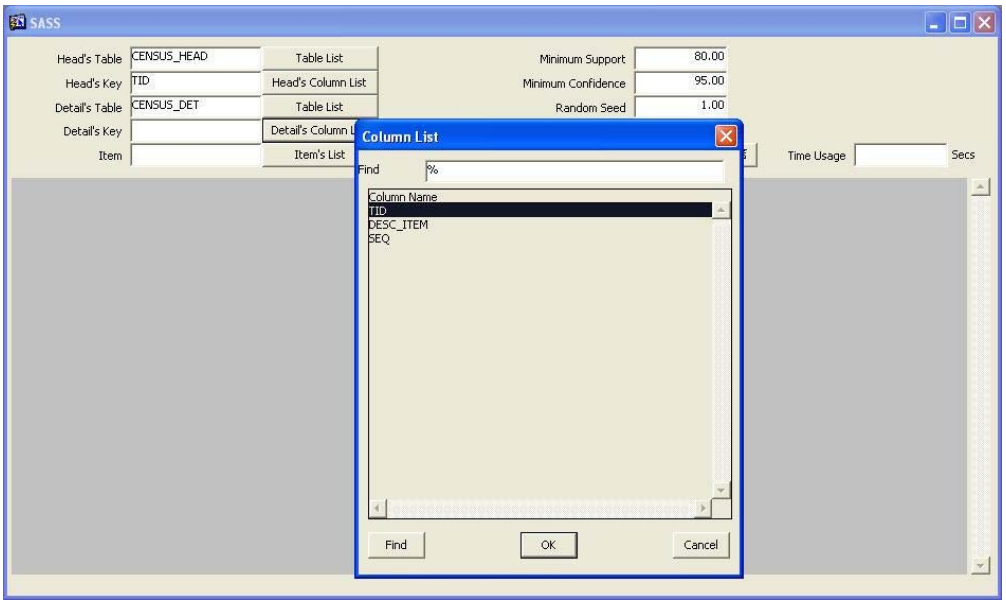
รูปที่ ค.6 วิธีการระบุคีย์ของตารางที่เป็นทรานแซคชันในส่วนของมาสเตอร์

1.7 ทำการระบุตารางที่เป็นทรานแซคชันในส่วนของรายละเอียด หรือโดยการคลิกปุ่ม Table List ที่อยู่ด้านข้างเพื่อดูรายชื่อของตารางทั้งหมดที่มีอยู่ในฐานข้อมูล ดังแสดงในรูปที่ ค.7



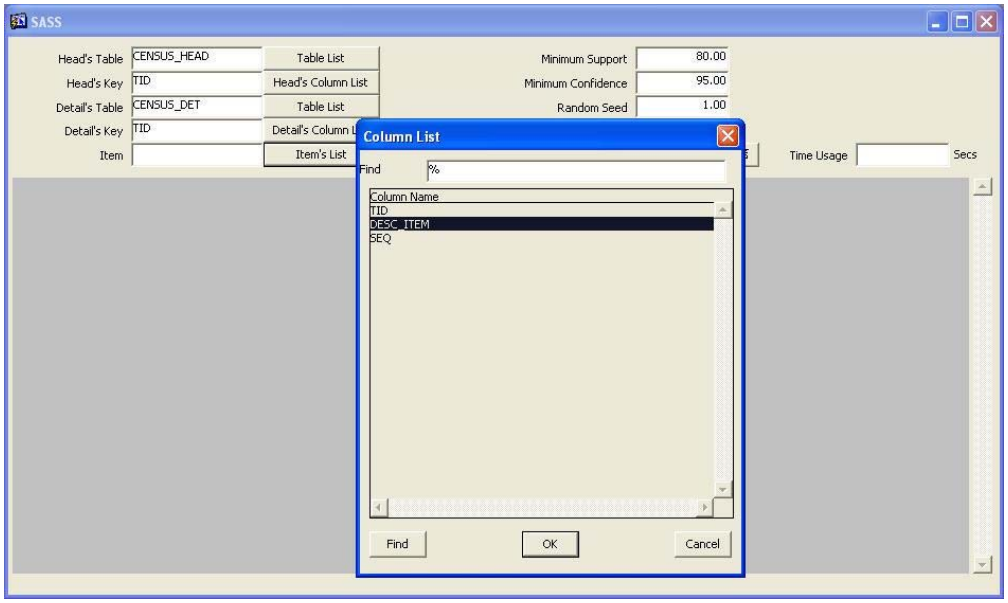
รูปที่ ค.7 วิธีการระบุตารางที่เป็นทรานแซคชันในส่วนของรายละเอียด

1.8 ทำการระบุคีย์ของตารางทรานแซคชันในส่วนของรายละเอียด หรือโดยการคลิกปุ่ม Detail's Column List ที่อยู่ด้านข้างเพื่อดูรายชื่อของคอลัมน์ที่มีอยู่ในตารางทรานแซคชันในส่วนของรายละเอียดทั้งหมดที่ระบุไว้ข้างต้น ดังแสดงในรูปที่ ค.8



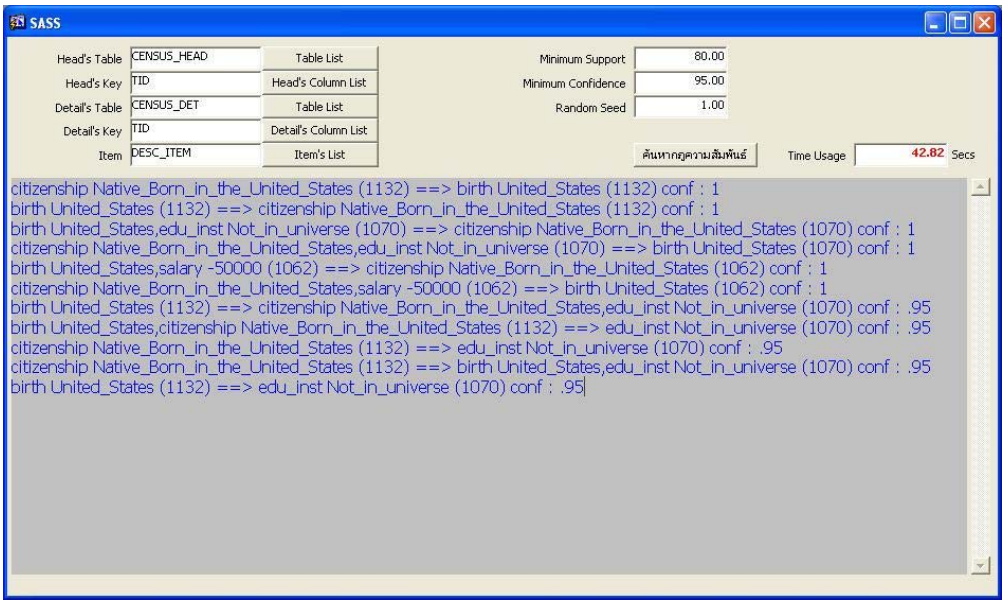
รูปที่ ค.8 วิธีการระบุคีย์ของตารางที่เป็นทรานแซคชันในส่วนของรายละเอียด

1.9 ทำการระบุคอลัมน์ที่จะนำไปใช้ในการค้นหาความสัมพันธ์ หรือโดยการคลิกปุ่ม Item's List ที่อยู่ด้านข้างเพื่อดูรายชื่อของคอลัมน์ที่มีอยู่ในตารางทรานแซคชันในส่วนของรายละเอียดทั้งหมดที่ระบุไว้ข้างต้น ดังแสดงในรูปที่ ค.9



รูปที่ ค.9 วิธีการระบุคอลัมน์ที่จะนำไปใช้ในการค้นหาความสัมพันธ์

1.10 หลังจากที่ระบุตารางและคอลัมน์ต่างๆ และกำหนดค่าสนับสนุนขั้นต่ำ ค่าความเชื่อมั่นขั้นต่ำ และขนาดของข้อมูลสุ่มตามที่ต้องการแล้ว คลิกที่ปุ่ม "ค้นหากฎความสัมพันธ์" โปรแกรมก็จะเริ่มทำการประมวลผลค้นหากฎความสัมพันธ์ เมื่อสิ้นสุดการทำงานแล้วจะแสดงกฎความสัมพันธ์ให้ทราบดังแสดงในรูปที่ ค.10

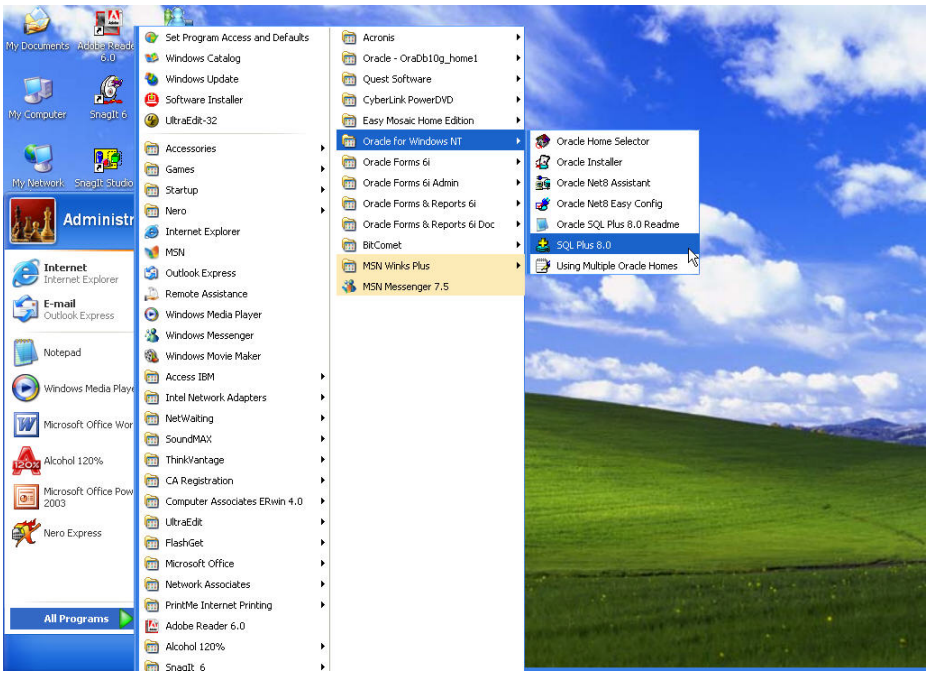


รูปที่ ค.10 ผลการค้นหาความสัมพันธ์ของโปรแกรม SASS

2. การใช้โปรแกรม SASS โดยการเรียกใช้โปรแกรมโดยตรง

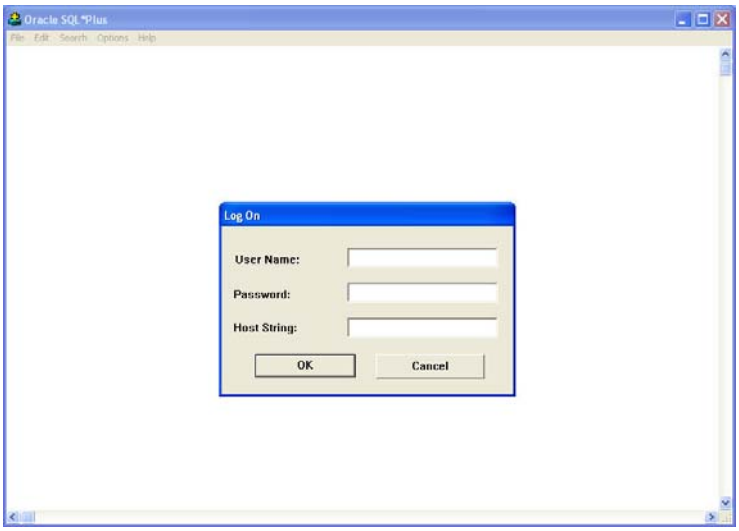
โปรแกรม SASS เป็นโปรแกรมที่พัฒนาขึ้นในรูปแบบของภาษา SQL การเขียนโปรแกรมใช้รูปแบบของของสโตร์โพรซิเยอร์ในการเรียกใช้โปรแกรม SASS โดยตรงสามารถทำได้ดังนี้

2.1 ให้ทำการเปิดโปรแกรม SQL Plus ที่ติดตั้งมาพร้อมกับ Oracle ดังแสดงในรูปที่ ค.11 ในที่นี้ใช้ SQL Plus เวอร์ชัน 8.0



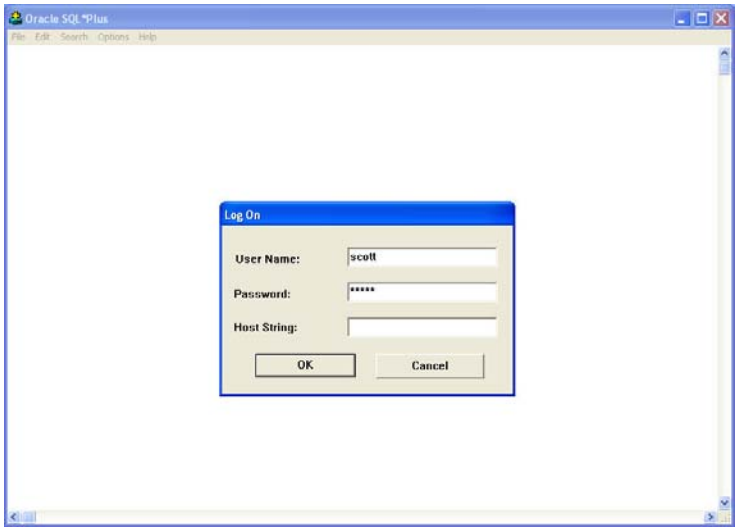
รูปที่ ค.11 การเปิดโปรแกรม SQL Plus 8.0

2.2 หลังจากที่โปรแกรม SQL Plus 8.0 ถูกเรียกใช้งานจะแสดงในรูปที่ ค.12



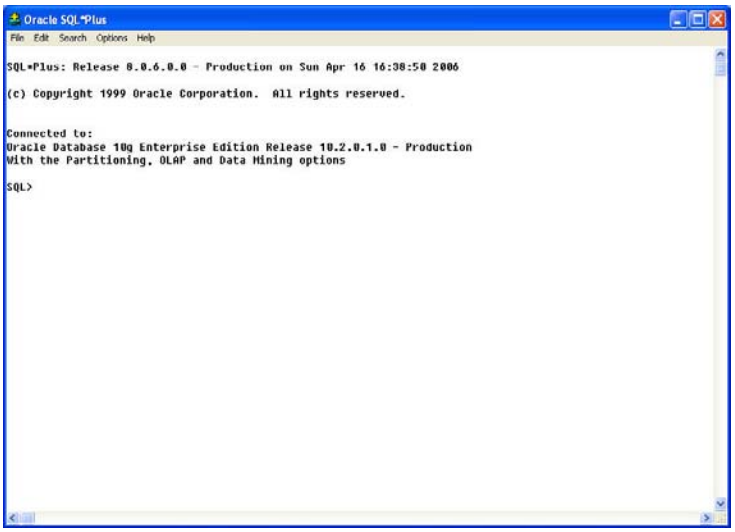
รูปที่ ค.12 เมื่อโปรแกรม SQL Plus 8.0 ถูกเรียกใช้งาน

2.3 ให้ทำการระบุยูสเซอร์เนม และรหัสผ่านของฐานข้อมูลที่จะทำการเชื่อมต่อเพื่อใช้งาน ดังแสดงในรูปที่ ค.13



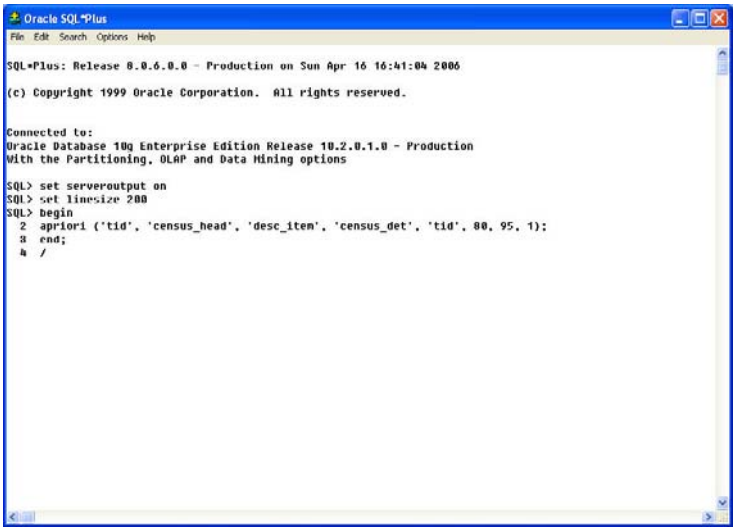
รูปที่ ค.13 ระบุยูสเซอร์เนม และรหัสผ่านของฐานข้อมูล

2.4 หลังจากที่ระบุยูสเซอร์เนม และรหัสผ่านของฐานข้อมูลแล้ว โปรแกรมจะทำการเชื่อมต่อกับฐานข้อมูลและพร้อมใช้งานดังแสดงในรูปที่ ค.14

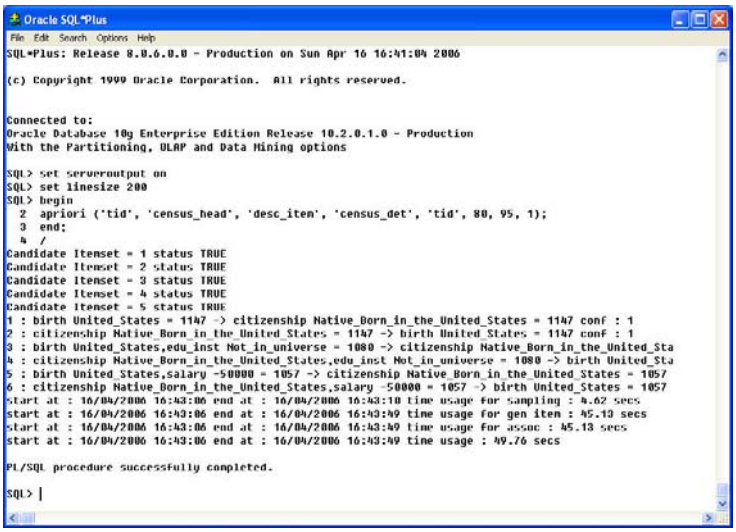


รูปที่ ค.14 โปรแกรมทำการเชื่อมต่อกับฐานข้อมูล และเข้าสู่สถานะพร้อมใช้งาน

2.5 การเรียกใช้งาน SASS จะต้องเรียกใช้ในรูปแบบดังที่แสดงในรูปที่ ค.15 และผลการค้นหากฎความสัมพันธ์แสดงในรูปที่ ค.16



รูปที่ ค.15 การเรียกใช้โปรแกรม SASS



รูปที่ ค.16 ผลการค้นหากฎความสัมพันธ์จากการเรียกใช้โปรแกรม SASS โดยตรง

ประวัติผู้เขียน

นางสาวลักษมี โขมโนทัย เกิดเมื่อวันที่ 3 พฤษภาคม พ.ศ. 2516 เกิดที่จังหวัดนครราชสีมา สำเร็จการศึกษาระดับปริญญาตรี บริหารธุรกิจบัณฑิต (คอมพิวเตอร์ธุรกิจ) จากมหาวิทยาลัยวงษ์ชวลิตกุล จังหวัดนครราชสีมา เมื่อ พ.ศ.2538 ภายหลังสำเร็จการศึกษาได้เข้าทำงานกับบริษัท ซอฟต์สแควร์ 1999 จำกัด เป็นเวลา 8 ปี จึงได้เข้าศึกษาต่อในระดับปริญญาโทในสาขาวิชาวิศวกรรมคอมพิวเตอร์ สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี ในปีการศึกษา 2547 ในระหว่างการศึกษาได้รับความไว้วางใจให้เป็นผู้ช่วยวิจัย และผู้สอนปฏิบัติการของสาขาวิศวกรรมคอมพิวเตอร์ในรายวิชา Database System