โครงการวิจัยประจำปีงบประมาณเงินรายได้ 2550

เรื่อง

# การศึกษาการเพิ่มขยายการค้นหากฎความสัมพันธ์

หัวหน้าโครงการวิจัย

รศ. ดร.วรพจน์ กรีสุระเดช



คณะเทคโนโลยีสารสนเทศ

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

# สารบัญ

หน้า

# บทคัดย่อ

ในค้นหากฎความสัมพันธ์ของระบบฐานข้อมูลที่มีการเปลี่ยนแปลงข้อมูล กฎความสัมพันธ์ที่ค้นหาได้อาจเกิดการเปลี่ยนแปลงได้ ดังนั้นการบำรุงรักษากฎความสัมพันธ์ให้สอดคล้องกับการเปลี่ยนแปลงของข้อมูลมีความสำคัญอย่างยิ่งในงานวิจัยนี้เสนอแนวทางการค้นหากฎความสัมพันธ์ในกรณีที่ฐานข้อมูลที่มีการเปลี่ยนแปลงข้อมูล โดยในการวิจัยนี้เสนอการใช้หลักการของเบอร์นูลี่เข้ามาใช้ในการประมาณข้อมูลที่คาดว่าจะสามารถนำมาใช้เป็นกฎความสัมพันธ์ และเสนออัลกอริทึมในการปรับเปลี่ยนกฎความสัมพันธ์ ซึ่งผลการทดลองสามารถแสดงถึงว่าการค้นหากฎความสัมพันธ์ของงานวิจัยนี้ สามารถค้นหากฎได้มีประสิทธิภาพ

# Abstract

In dynamic databases, new transactions are appended as time advances. This may introduce new association rules and some existing association rules would become invalid. Thus, the maintenance of association rules for dynamic databases is an important problem. In this paper, probability-based incremental association rule discovery algorithm is proposed to deal with this problem. The proposed algorithm uses the principle of Bernoulli trials to find expected frequent itemsets. This can reduce a number of times to scan an original database. This paper also proposes a new updating and pruning algorithm that guarantee to find all frequent itemsets of an updated database efficiently. The simulation results show that the proposed algorithm has a good performance

# บทที่ 1
# บทนำ

## 1.1 ความเป็นมาและความสำคัญของปัญหา

ในปัจจุบันเทคโนโลยีสารสนเทศเข้ามามีบทบาทกับหน่วยงานและองค์กรต่างๆ ทำให้มีการจัดเก็บข้อมูลจำนวนมากไว้ในฐานข้อมูล และมีแนวโน้มที่จะมีจำนวนเพิ่มขึ้นเรื่อยๆ โดยไม่ได้นำข้อมูลต่างๆ มาใช้ให้เกิดประโยชน์สูงสุด ซึ่งหากนำข้อมูลที่จัดเก็บไว้แล้วมาศึกษาสิ่งที่อาจจะพบคือความสำคัญของข้อมูลที่มีความสัมพันธ์อย่างคาดไม่ถึงซ่อนอยู่ก็คือความรู้ (knowledge) นั่นเอง

การจัดการความรู้ได้ถูกพัฒนาอย่างกว้างขวางทั้งทางเทคโนโลยีและโปรแกรมประยุกต์ สำหรับงานวิจัยนี้ให้ความสนใจในเรื่องการค้นหาความรู้ในฐานข้อมูล (knowledge discovery in database: KDD) ที่จัดอยู่ในกลุ่มของงานวิจัยด้านเทคโนโลยีฐานข้อมูล (database technology)[1] การค้นหาความรู้ในฐานข้อมูลเป็นการอ้างถึงกระบวนการทั้งหมดของการค้นหาความรู้ที่เป็นประโยชน์จากข้อมูล ในหลายๆ งานวิจัยอาจจะพบคำที่ใช้ในความหมายเดียวกันคือ ดาต้าไมน์นิ่ง (data mining) แต่ดาต้าไมน์นิ่งเป็นเพียงขั้นตอนหนึ่งในกระบวนการค้นหาข้อมูลจากฐานข้อมูลที่นำมาหารูปแบบ (pattern) เพื่อใช้ในการสกัดข้อมูล (extract data) ที่สามารถนำมาใช้ประโยชน์ได้

การไมน์นิ่งกฎความสัมพันธ์ (Association rule mining) เป็นกระบวนการสำคัญในการทำดาต้าไมน์นิ่ง โดยจะหาดึงรูปแบบของข้อมูลระหว่างรายการต่างๆ ทั้งหมดที่ได้จัดเก็บไว้ในฐานข้อมูลมาใช้ในการทำนายความสัมพันธ์ระหว่างข้อมูลด้วยการสร้างให้อยู่ในรูปของกฎความสัมพันธ์ เพื่อช่วยในการตัดสินใจและวางแผนด้านบริหารได้

ปัญหาที่พบในการหากฎความสัมพันธ์ของฐานข้อมูลคือ การหา large k-itemset หรือ frequent k-itemset ซึ่งหมายถึง itemset ที่ประกอบด้วย k item (k=1,2,…,n) ที่เกิดขึ้นร่วมกันและได้รับการพิจารณาว่ามีจำนวนมากกว่าหรือเท่ากับค่าที่ใช้ในการวัดความสัมพันธ์ที่ได้กำหนดไว้ (minimum support) และการนำ large k-itemset ที่ได้มาสร้างเป็นกฎความสัมพันธ์ ให้อยู่ในรูปของ IF…THEN rules เช่น IF milk THEN bread หมายถึง ถ้าซื้อนมแล้วจะซื้อขนมปังด้วย เป็นต้น ดังนั้นเมื่อ transaction ต่างๆ ที่ถูกจัดเก็บในฐานข้อมูลนั้นมีการปรับเปลี่ยนให้มีความทันสมัยหรือทันต่อเวลาจะมีผลต่อการเปลี่ยนแปลงค่าความสัมพันธ์ระหว่าง large k-itemset ทำให้กฎความสัมพันธ์ที่มีอยู่เดิมอาจจะไม่มีความถูกต้องอีกต่อไป

การเพิ่มขยายการค้นหากฎความสัมพันธ์เป็นการทำดาต้าไมน์นิ่งเพื่อค้นหากฎความสัมพันธ์ที่ได้จากเปลี่ยนแปลง transaction ในช่วงเวลาหนึ่งๆ เช่นการเพิ่ม, ลบ หรือแก้ไข เป็นต้น ซึ่งมีผลต่อ large k-itemset ที่มีอยู่เดิม ทำให้ข้อมูลที่เคยเป็น small itemset อาจกลายมา

เป็น large itemset หรือในทางกลับกันข้อมูลที่เคยเป็น large itemset อาจจะเป็นหรือไม่เป็น large itemset ในฐานข้อมูลที่ได้รับการปรับปรุงแล้วอีกต่อไป ซึ่งมีผลต่อการเปลี่ยนแปลงกฎความสัมพันธ์ทำให้ต้องหา large k-itemset ใหม่ ลักษณะปัญหาของการหากฎความสัมพันธ์ใหม่สามารถสรุปได้ดังนี้คือ

- จำนวนครั้งที่ใช้ในการสแกนฐานข้อมูลเดิมและข้อมูลใหม่ที่ต้องการปรับปรุงใหม่ ซึ่งใช้เวลามากในการค้นหาความสัมพันธ์ด้วยขั้นตอนการหากฎความสัมพันธ์ใหม่ทั้งหมด โดยไม่นำความรู้ที่ได้จากการทำดาต้าไมน์นิ่งก่อนที่จะทำการปรับปรุงมาใช้

- ปัญหาในการหา large k-itemset ที่เกิดในฐานข้อมูลที่ปรับปรุง ($L_{DB+db}$) ซึ่งจะต้องมีค่า support ที่เกิดขึ้นมากกว่าค่า minimum support ที่กำหนด เพื่อนำมาสร้างกฎความสัมพันธ์ใหม่ที่เกิดขึ้น

## 1.2 ความมุ่งหมายและวัตถุประสงค์ของการศึกษา

งานวิจัยนี้มีวัตถุประสงค์เพื่อค้นคว้าและพัฒนาอัลกอริทึมที่ใช้ในการเพิ่มขยายการค้นหากฎความสัมพันธ์เมื่อมีการปรับเปลี่ยน transaction ในฐานข้อมูลหรือการปรับเปลี่ยนค่า minimum support ซึ่งค่า minimum support คือ ค่าความน่าจะเป็นทางสถิติที่ใช้ในการวัดความสัมพันธ์ระหว่างข้อมูลที่มีโอกาสจะเกิดขึ้นด้วยกัน โดยมีผลต่อการเปลี่ยนแปลงกฎความสัมพันธ์ที่ได้ทำการค้นหาไว้แล้วจากการทำดาต้าไมน์นิ่งในฐานข้อมูลเดิม (original database) และนำแนวความคิดที่จะหลีกเลี่ยงหรือใช้จำนวนครั้งน้อยที่สุดในการสแกนฐานข้อมูลเดิมที่มีจำนวนของข้อมูลที่จัดเก็บไว้จำนวนมาก รวมถึงการจัดการฐานข้อมูลใหม่ (incremental database) ที่มีผลต่อการเปลี่ยนแปลงกฎความสัมพันธ์ใหม่ที่เกิดขึ้นภายหลังจากการปรับปรุงฐานข้อมูล (updated database)

เพื่อให้การค้นหากฎความสัมพันธ์ระหว่างข้อมูลครอบคลุมข้อมูลทั้งหมด ด้วยอัลกอริทึมที่มีประสิทธิภาพในการสร้าง candidate itemset ที่สามารถนำมาใช้ในการหา large k-itemset ได้อย่างถูกต้องและรวดเร็ว โดยใช้ทรัพยากรที่มีอยู่ให้เกิดประโยชน์มากที่สุด

## 1.3 ทฤษฎีหรือแนวคิดที่ใช้ในการวิจัย

งานวิจัยนี้ได้นำทฤษฎีและแนวความคิดต่างๆมาใช้ คือการประยุกต์ใช้ในการหากฎความสัมพันธ์ โดยมีทฤษฎีและแนวคิดที่สามารถนำมาประยุกต์ใช้ได้ดังนี้คือ

1. Knowledge discovery in database
2. Association rule
3. Incremental association rule

## 1.4 ขอบเขตของการวิจัย

งานวิจัยที่นำเสนอนี้เป็นโมเดลที่ประยุกต์ใช้กับการเพิ่มขยายการค้นหากฎความสัมพันธ์เมื่อฐานข้อมูลถูกปรับปรุงให้สอดคล้องกับข้อมูลที่ต้องการหากฎความสัมพันธ์ได้

## 1.5 ขั้นตอนของการศึกษา

ในขั้นตอนของการศึกษานี้ ได้แสดงลำดับการทำงานตั้งแต่เริ่มต้นจนถึงสิ้นสุดการทำงานดังรายละเอียดต่อไปนี้

1.5.1 ศึกษาทฤษฎีและงานวิจัยจากเอกสารบทความต่าง ๆ ที่เกี่ยวข้องกับการทำงานวิจัย

1.5.2 กำหนดหัวข้อ วัตถุประสงค์ และขอบเขตการทำงานวิจัย

1.5.3 วิเคราะห์อัลกอริทึมและออกแบบโมเดลใหม่

1.5.4 พัฒนาโปรแกรม โดยใช้ซอฟท์แวร์ MATLAB พร้อมทั้งแก้ไขข้อผิดพลาดและทดสอบการทำงานของโมเดลกับข้อมูลที่กำหนดขึ้นเอง

1.5.5 เตรียมข้อมูลที่ใช้งานจริง เพื่อนำมาทดสอบการทำงานโมเดล โดยในที่นี้จะใช้ synthetic dataset ที่ได้มีการคิดค้นโดย Agrawal et al [2] ด้วยหลักการสร้าง dataset จากหลักการทางสถิติที่เลียนแบบมาจากลักษณะการซื้อขายสินค้าจริง และได้มีการนำมาใช้ในการทดสอบอย่างแพร่หลายในงานวิจัยต่างๆ

1.5.6 รวบรวมผลการทดลองจากการทำงานของโมเดล

1.5.7 วิเคราะห์และสรุปผลการทดลอง

การหากฎความสัมพันธ์เป็นงานวิจัยหนึ่งที่ได้รับความนิยมเพื่อค้นหาความรู้หรือ information ที่ซ่อนอยู่ในฐานข้อมูล และนำความรู้ที่ได้มาแสดงอยู่ในรูปของกฎความสัมพันธ์เพื่อใช้ในงานด้านต่างๆ เช่น การบริหาร, การตัดสินใจ และการวางแผน เป็นต้น ซึ่งการหากฎความสัมพันธ์มีการทำวิจัยในหลายรูปแบบแตกต่างกันตามลักษณะของข้อมูล

ในบทที่ 2 จะกล่าวถึง ทฤษฎีและงานวิจัยต่างๆ ที่เกี่ยวข้องกับการทำงานวิจัยที่นำเสนอนี้

# บทที่ 2
# ทฤษฎีพื้นฐานที่ใช้ในการวิจัย และงานวิจัยที่เกี่ยวข้อง

การหากฎความสัมพันธ์เป็นการทำไมน์นิ่งอย่างหนึ่งที่ได้รับความสนใจในการทำงานวิจัย เพื่อค้นหาสารสนเทศที่ซ่อนอยู่ในฐานข้อมูล โดยเฉพาะในเรื่องที่เกี่ยวกับ market basket ทั้งนี้เพื่อให้สามารถนำข้อมูลหรือความรู้ที่ได้จากการกฎความสัมพันธ์มาใช้ในด้านต่างๆ เช่น การวางแผนกลยุทธ์ทางการตลาด, ช่วยในการตัดสินใจหรือวางแผนเกี่ยวกับผลิตภัณฑ์ เป็นต้น

ในบทนี้จะกล่าวถึงทฤษฎีพื้นฐานต่างๆ ของการหากฎความสัมพันธ์ (Association rules) , การเพิ่มขยายกฎความสัมพันธ์ (Incremental Association rule) และงานวิจัยด้านต่างๆ ที่เกี่ยวข้องกับการหากฎความสัมพันธ์ของข้อมูลที่ปรากฏในฐานข้อมูลซึ่งมีรายละเอียดดังนี้

## 2.1 การไมน์นิ่งกฎความสัมพันธ์

กระบวนการไมน์นิ่งกฎความสัมพันธ์ (Association rules mining) ได้ถูกนำเสนอในปีค.ศ. 1993 [1] เพื่อใช้ในการค้นหาความสัมพันธ์ระหว่างรายการในแต่ละรายการหรือกลุ่มรายการที่ปรากฏขึ้นในฐานข้อมูลขนาดใหญ่ ความสัมพันธ์ที่ได้สามารถใช้ในการบอกลักษณะของข้อมูลหรือใช้ในการทำนายลักษณะของข้อมูลที่อาจเกิดขึ้นต่อไป โดยจะแสดงอยู่ในรูปของกฎความสัมพันธ์ (Association Rules)

กฎความสัมพันธ์ เป็นเทคนิคที่ใช้หากฎเกณฑ์ความสัมพันธที่ปรากฏขึ้นระหว่างข้อมูลทั้งหมดในฐานข้อมูล โดยกำหนดให้ I เป็นเซตของ item I โดย $I = I_1, I_2, \ldots, I_m$ , D เป็นเซตของ transaction ในฐานข้อมูล ซึ่งแต่ละ transaction T เป็นเซตของ items โดย $T \subseteq I$ T แต่ละตัวจะสัมพันธ์กับตัวระบุ (identifier) ที่เรียกว่า TID (Transaction Identifier) ,A เป็นเซตของ items ใน transaction หนึ่ง ซึ่ง $A \subseteq T$ item ต่างๆ ที่ปรากฏในฐานข้อมูลจะถูกนำมาหาความสัมพันธ์ โดย item ที่จะถูกนำมาสร้างเป็นกฎความสัมพันธ์ได้จะต้องมีจำนวนของข้อมูลที่เกิดขึ้นมากกว่าหรือเท่ากับตัววัด 2 ตัวคือ minimum support และ minimum confidence

กฎความสัมพันธ์ที่ได้จะแสดงอยู่ในรูปของกฎ "ถ้า ... แล้ว..." (IF...THEN ...) ซึ่งในกฎหนึ่งๆ ประกอบด้วย 2 ส่วนคือ ส่วนที่ด้านซ้ายกฎหรือส่วนที่อยู่หลัง IF ที่ประกอบด้วยเงื่อนไขที่เป็นจริงหนึ่งหรือมากกว่าหนึ่งเงื่อนไข เรียกส่วนนี้ว่า Rule Body หรือ Antecedent หรือ Left-hand side และส่วนด้านขวาของกฎหรือส่วนที่อยู่หลัง THEN เป็นส่วนที่แสดงผลเมื่อเงื่อนไขที่ระบุในส่วนของ IF เป็นจริง เรียกส่วนนี้ว่า Rule head หรือ Consequent หรือ Right-hand side ตัวอย่างของกฎความสัมพันธ์เช่น IF A THEN B หรือ เขียนแทนด้วยสัญลักษณ์ $A \Rightarrow B$ โดยค่า $A \subset I$,

$B \subset I$ และ $A \cap B = \phi$ กฎ $A \Rightarrow B$ จะถูกจัดให้มีในเซต transaction D ด้วยค่า support s ซึ่งเป็นเปอร์เซ็นต์ของข้อมูล transaction ใน D ที่มีทั้ง A และ B ($A \cup B$) โดยจะเป็นค่าความน่าจะเป็นที่จะเกิด A และ B พร้อมกัน ($P(A \cup B)$) กฎ $A \Rightarrow B$ จะมีค่า confidence c ในเซตของข้อมูล transaction ใน D ถ้า c เป็นเปอร์เซ็นต์ของข้อมูล transaction ใน D ที่มี A แล้วจะมี B ด้วย ซึ่งเป็นเรื่องของความน่าจะเป็นแบบมีเงื่อนไข ($P(B \mid A)$) ดังสามารถสรุปได้ดังนี้

$$\text{Support}(A \Rightarrow B) = P(A \cup B)$$

$$\text{Confidence}(A \Rightarrow B) = P(B \mid A)$$

**นิยามและความหมายของคำต่าง ๆ ที่ใช้ในการค้นหากฎความสัมพันธ์ได้แก่**

1. ไอเทม (Item) คือข้อมูลแต่ละตัวที่ใช้ในการหากฎความสัมพันธ์ เช่น bread, cheese , shampoo, milk เป็นต้น
2. ไอเทมเซต (Itemset) คือ ความสัมพันธ์ของข้อมูลที่หาได้ Itemset ประกอบด้วย Item ที่มีความยาวแตกต่างกัน แทนขนาดความยาวของ itemset ได้ด้วย k (k= 1, 2 ,3,...,n) เรียกว่า k-itemsets เช่น 2-itemsets หมายถึง itemset ที่ประกอบด้วยสมาชิกของ item 2 ตัว เช่น {bread, cheese}, {milk, bread} , { milk , shampoo} เป็นต้น และ 3-itemsets หมายถึง itemset ที่ประกอบด้วยสมาชิกของ item 3 ตัว เช่น {milk, bread , shampoo} , { bread, cheese, milk} เป็นต้น
3. ไอเทมเซตตัวเลือก (Candidate Itemset) คือ Itemset ที่ได้จากการเชื่อมความสัมพันธ์ของ Itemset ก่อนหน้านี้ ซึ่งเป็นไอเทมเซตตัวเลือกสำหรับฟรีเควนท์ไอเทมเซต
4. ฟรีเควนท์ไอเทมเซต (Frequent Itemset) หรือ ลาร์จไอเทมเซต (Large Itemset) คือชุดของ Itemset ที่มีค่าสนับสนุนมากกว่าหรือเท่ากับค่าสนับสนุนน้อยที่สุด
5. ค่าสนับสนุน (Support value) เป็นค่าแสดงความสัมพันธ์ระหว่างจำนวนของไอเท็มที่ปรากฏรายการข้อมูลต่างๆ ทั้งหมด ในฐานข้อมูลสามารถแสดงเป็นสมการได้ดังสมการที่ 2.1

$$Support = \frac{n}{N} \qquad (2.1)$$

โดยที่ n คือจำนวนไอเทมที่ปรากฏในรายการข้อมูลต่างๆ ของฐานข้อมูล

N คือจำนวนรายการข้อมูลทั้งหมดในฐานข้อมูล

6. ค่าสนับสนุนน้อยที่สุด (Minimum support : min_sup) คือค่าสนับสนุนที่น้อยที่สุดที่ทำให้ความสัมพันธ์ที่ได้นั้นยังมีความน่าสนใจ

$$\text{Support}(A \Rightarrow B) = P(A \cup B) \qquad (2.2)$$

โดยที่ $P(A \cup B)$ คือ เป็นค่าความน่าจะเป็นที่จะปรากฏไอเท็ม A และ B พร้อมกันในรายการข้อมูลต่างๆ ของฐานข้อมูล

7. ค่าความเชื่อมั่นน้อยที่สุด (Minimum confidence) คือค่าความเชื่อมั่นที่น้อยที่สุดที่ทำให้กฎความสัมพันธ์ที่ได้นั้นยังมีความน่าสนใจ โดยจะบอกถึงความเข้มแข็งของกฎความสัมพันธ์ที่เกิดขึ้น โดยพิจารณาความน่าจะเป็นของความสัมพันธ์แบบมีเงื่อนไข ซึ่งสามารถคำนวณได้ แสดงดังสมการที่ 2.3

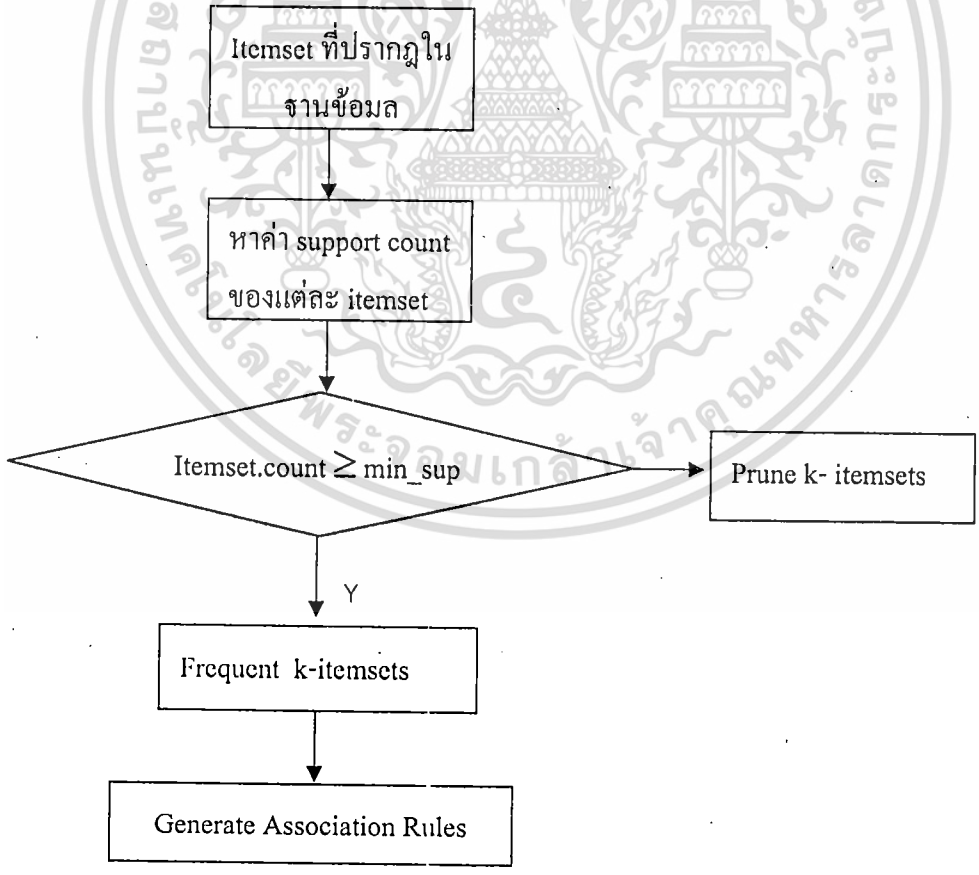
$$\text{Confidence}(A \Rightarrow B) = P(B \mid A) \qquad (2.3)$$

$$P(B|A) = \frac{P(A \cap B)}{P(A)}$$

โดยที่ P(B|A) หมายถึง ความน่าจะเป็นที่ B จะเกิดขึ้นเมื่อ A เกิดขึ้นแล้ว

P(A) คือความน่าจะเป็นของไอเท็ม A

การค้นหากฎความสัมพันธ์มีกระบวนการทำงาน 2 ขั้นตอนคือ หา Frequent Itemset ทั้งหมด และสร้างกฎความสัมพันธ์จาก Frequent Itemset แสดงดังรูปที่ 2.1




**รูปที่ 2.1** ขั้นตอนการค้นหากฎความสัมพันธ์

อัลกอริทึมที่ได้รับความนิยมในการหากฎความสัมพันธ์ก็คือ Apriori [2] ซึ่งมีลักษณะดังนี้คือ

### 2.1.1 Apriori Algorithm

เป็นอัลกอริทึมที่เป็นที่รู้จักและมีการนำมาประยุกต์ใช้กับงานวิจัยต่างๆ อย่างแพร่หลายโดยอัลกอริทึมของ Apriori จะทำการคำนวณหาข้อมูลที่เป็น frequent itemsetsในฐานข้อมูล โดยทำการสแกนข้อมูลแต่ละ transaction ในฐานข้อมูลผ่านการวนซ้ำหลายครั้ง (iterations) และในการวนซ้ำจะคำนวณจาก frequent k-itemsets ( itemsets ด้วยสมาชิก ตัว) โดยอาศัยความรู้ที่ได้จากขั้นตอนก่อนหน้าเช่น k-itemset จะได้จากการสร้าง candidate k-1 itemset ($C_{k-1}$) ในรูปแบบที่เรียกว่า level-wise search จากรูปที่ 2.2 เป็นการแสดงขั้นตอนการหา Frequent item ซึ่งจะประกอบด้วย 2 ขั้นตอนหลักๆ ดังนี้คือ

1. ขั้นตอนการ join (Join step)

เพื่อหา large itemset ($L_k$)การสร้างและนับ Candidate itemsets โดยอัลกอริทึมของ Apriori จะต้องมีการเรียงลำดับข้อมูลใน transaction (Lexicographic order) ที่ใช้ในการสร้างเซตของ candidate k-itemsets ($C_k$) ซึ่งมีค่า support > 0 และ candidate itemsets $C_k$ จะสามารถหาได้จาก $L_{k-1}$ ในกรณีที่ไม่ใช่ 1-itemset จะสามารถหา $C_k$ ได้จากการนำ $L_{k-1}$ มา join กัน ดังแสดงในรูปที่ 2.3 เนื่องจากในการหากฎความสัมพันธ์จะมีการนำข้อมูลมาเรียงลำดับ

การสร้าง Frequent itemset จาก Candidate itemset ที่ $L_{k-1}$ โดยจะตรวจสอบว่าค่าsupport ของ Candidate itemset ใดๆ ว่ามีค่ามากกว่าหรือเท่ากับค่า minimum support ที่กำหนดไว้ item นั้นๆ จะกลายเป็น Large k-itemset ($L_k$)

2. ขั้นตอนการ Prune (Prune step)

เมื่อ $C_k$ ถูกนำมาพิจารณาว่ามีค่าน้อยกว่า minimum support ก็จะเข้าสู่ขั้นตอนดังแสดงในรูปที่ 2.4 คือการ prune item นั้นๆ ออกไปจาก candidate itemsets เพื่อให้การคำนวณน้อยลงในส่วนของ Apriori ได้มีการกำหนดคุณสมบัติของ item ไว้ดังนี้ “ถ้าซับเซต (k-1) ใด ๆ ของ Candidate k-itemset ไม่ได้เป็นสมาชิก $L_{k-1}$ ดังนั้น candidate ไม่สามารถเป็น frequent ได้ และสามารถลบออกจาก $C_k$ ได้”

Algorithm Apriori. Find frequent itemsets using an iterative level-wise approaches based on candidate generation

Input: Database, D, of transactions; minimum support threshold, min_sup

Output: L,frequent itemsets in D.

Method:

1) $L_1$ = find_frequent_1-itemsets(D)
2) for (k=2; $L_{k-1} \neq \phi; k$ ==){
3) $C_k = apriori\_gen(L_k, \text{min\_sup})$;
4) for each transaction t $\in$ D {// scan D for counts
5) $C_t$ = subset($C_k, t$); // get the supsets of t that are candidates
6) for each candidate c $\in$ $C_t$
7) c.count++;
8) }
9) $L_k = \{c \in C_k | c.count \geq \text{min\_sup}\}$
10) }
11) return L=$\bigcup_k L_k$;

รูปที่ 2.2 Apriori algorithm

function apriori_gen

Generate the candidate itemsets in $C_k$ from the frequent itemsets in $L_{k-1}$

Join $L_{k-1}$ $p$ with $L_{k-1}$q, as follows:

**insert into** $C_k$

**select** $p$.item$_1$, $q$.item$_1$, . . . , $p$.item$_{k-1}$, $q$.item$_{k-1}$

**from** $L_{k-1}$ $p$, $L_{k-1}$q

**where** $p$.item$_1$ = $q$.item$_1$, . . . $p$.item$_{k-2}$ = $q$.item$_{k-2}$, $p$.item$_{k-1}$ < $q$.item$_{k-1}$

**รูปที่ 2.3** join step ของ procedure Apriori_gen

```
Prune Step :

for all itemsets c ∈ C_k do
   for all (k-1)-subsets s of c do
      if (s ∉ L_k-1) then
         delete c from C_k;
```

**รูปที่ 2.4** Prune step ของ Apriori algorithm

กฎความสัมพันธ์ที่เป็นไปตามค่าที่กำหนดไว้ทั้ง minimum support threshold (min_sup) และ minimum confidence threshold (min_conf) จะถูกเรียกว่า กฎที่เข้มแข็ง (Strong) โดยจะเขียนค่า support และ confidenceที่มีในช่วงระหว่าง 0% ถึง 100%

กฎความสัมพันธ์จะหาค่าความสัมพันธ์ระหว่างแอททริบิวต์ที่แตกต่างกัน โดยจะมีการอ้างถึงเซตของ item ที่เรียกว่า itemset โดย itemset ที่ประกอบด้วย k-items (k = 1,2,3,…,n) จะเรียกว่า k-itemsets เช่น {milk, bread} เป็นตัวอย่างของ 2-itemsets และจากข้อมูล transaction จะสามารถทราบถึงความสัมพันธ์ที่เกิดขึ้นระหว่างรายการข้อมูลได้ด้วยการนับจำนวนของ transaction ที่มี itemset ที่เรียกว่า Frequency ของ itemset โดยค่าของ Frequency ของ itemset จะต้องมีค่ามากกว่าหรือเท่ากับค่า minimum support ที่กำหนดไว้ set ของ Frequent k-itemsets จะเขียนแทนด้วยสัญลักษณ์ $L_k$

การหากฎความสัมพันธ์สามารถทำได้ 2 ขั้นตอนคือ

1. หา frequent itemsets ทั้งหมด โดยแต่ละ itemsets จะต้องเกิดขึ้นอย่างน้อยเท่ากับค่าของ minimum support ที่กำหนดไว้

2. การสร้างกฎความสัมพันธ์จาก frequent itemsets ที่ได้ โดยกฎจะต้องเป็นไปตามค่า minimum support และ minimum confidence ที่กำหนดไว้

**ปัญหาของ Apriori**

1. เมื่อฐานข้อมูล หรือข้อมูลมีการเปลี่ยนแปลงต้องทำการหากฎความสัมพันธ์ใหม่ทั้งหมด โดยจะต้องสแกนข้อมูลใหม่ทั้งฐานข้อมูล

2. การหากฎความสัมพันธ์จะมีความซับซ้อนในการคำนวณสูงทำให้มีการคิดค้นหาอัลกอริทึมมาช่วยลดการคำนวณและสแกนข้อมูล

3. การหากฎความสัมพันธ์ในรูปแบบ Numeric หรือ Boolean ไม่สามารถทำได้

การหากฎความสัมพันธ์ของข้อมูลที่เป็น crisp set ทำให้ต้องมีการระบุว่าข้อมูลที่นำมาหากฎความสัมพันธ์นั้นอยู่ในกลุ่มใดนั้นอาจทำให้สูญเสียสาระสนเทศที่ซ่อนอยู่ในความสัมพันธ์ได้

เนื่องจากข้อมูลที่นำมาหากฎความสัมพันธ์อาจมีความคลุมเครือในการที่จะเป็นสมาชิกในกลุ่มใดกลุ่มหนึ่ง

## 2.2 แนวคิดของการเพิ่มขยายการค้นหากฎความสัมพันธ์

ฐานข้อมูลเป็นแหล่งที่ใช้ในการจัดเก็บข้อมูลที่ประกอบด้วยข้อมูลต่างๆ มากมาย โดยทั่วไปจะมีการค้นหาความสัมพันธ์ระหว่างข้อมูลในฐานข้อมูล ซึ่งอาจพบความรู้ที่ซ่อนอยู่ภายในฐานข้อมูลจากนั้นสามารถนำความรู้ที่ได้มาสร้างเป็นกฎความสัมพันธ์

ข้อมูลที่ถูกจัดเก็บในฐานข้อมูลสามารถเปลี่ยนแปลงหรืออาจกล่าวได้ว่าข้อมูลในฐานข้อมูลนั้นไม่คงที่ เรียกฐานข้อมูลในลักษณะนี้ว่า dynamic database การเปลี่ยนแปลงฐานข้อมูลอาจเกิดจากการเพิ่มรายการข้อมูลเข้าไปในฐานข้อมูล (insert) , ลบรายการข้อมูลที่มีอยู่ในฐานข้อมูล (delete)หรืออาจมีทั้งการเพิ่มและลบรายการข้อมูลในฐานข้อมูล (modify)โดยจะเรียกส่วนของฐานข้อมูลก่อนทำการเปลี่ยนแปลงว่าฐานข้อมูลเดิม (original database : DB) และเรียกส่วนที่ทำการเปลี่ยนแปลงว่าฐานข้อมูลใหม่ (incremental database : db) และเรียกส่วนของฐานข้อมูลที่ผ่านการปรับปรุงหลังจากมีการเปลี่ยนแปลงข้างต้นแล้วว่า ฐานข้อมูลปรับปรุง (updated database : UP)

เมื่อฐานข้อมูลมีการเปลี่ยนแปลงจะมีผลต่อกฎความสัมพันธ์ที่ได้ไม่นิ่งไว้ใน original database เนื่องจาก frequent itemset ที่ได้จาก original database อาจไม่สามารถเป็น frequent itemset ในฐานข้อมูลที่ปรับปรุงหรือ itemset ที่ไม่ผ่านค่า min_sup ของฐานข้อมูลเดิม หรือเรียกว่า small itemset อาจกลายมาเป็น frequent itemset ใน updated database ทำให้กฎความสัมพันธ์ที่ได้จาก original database บางกฎไม่สามารถเป็นกฎความสัมพันธ์ใน updated database อีกต่อไป ในขณะเดียวกันอาจพบกฎความสัมพันธ์ใหม่ที่เกิดขึ้นได้

## 2.3 งานวิจัยเกี่ยวกับ Incremental Association Rule

งานวิจัยด้านนี้เป็นการนำเสนออัลกอริทึมในหาความสัมพันธ์ระหว่างข้อมูลในฐานข้อมูลขนาดใหญ่ เมื่อมีการเพิ่มข้อมูลจาก transaction ใหม่ที่เข้ามา ซึ่งจะมีผลต่อการปรับเปลี่ยนกฎความสัมพันธ์ที่ได้เคยถูกสร้างไว้ โดยจะต้องทำการสแกน original database ที่มีอยู่และใน incremental database ที่มีการเพิ่ม transaction ใหม่เข้าไปเพื่อทำการสร้างกฎความสัมพันธ์ขึ้นมาใหม่ให้สอดคล้องกับข้อมูลใน transaction ที่ถูกเพิ่มเข้ามา [7] ซึ่งการเพิ่มข้อมูลที่เข้ามาจะทำให้เสียเวลาในการสแกนและหากฎความสัมพันธ์ใหม่ๆ งานวิจัย[8] จะวิธีการแบ่งข้อมูลเป็นส่วนๆ (partition) เพื่อลดจำนวนการสแกนฐานข้อมูลจากนั้นจะใช้ threshold ทำการกรองในแต่ละส่วนที่

เกี่ยวข้องกับการสร้าง candidate itemset การทำในรูปแบบนี้จะช่วยลดจำนวนของ candidate itemset และมีผลทำให้ลดซีพียูและหน่วยความจำ

รูปแบบการเพิ่มขยายกฎความสัมพันธ์ (Incremental association rule) จะนำเสนออัลกอริทึมในการหา large itemset เพื่อหากฎความสัมพันธ์ใหม่ที่เกิดขึ้นภายหลังจากมีการเพิ่มหรือปรับปรุงฐานข้อมูลเก่า โดยงานวิจัยที่ส่วนใหญ่จะเป็นการหาวิธีเพื่อลดการสแกน [3][4][6]-[15] เนื่องจากฐานข้อมูลที่เป็น original database มักจะมีขนาดใหญ่กว่า incremental database ดังนั้น ในงานวิจัยจึงมักหลีกเลี่ยงที่จะทำการสแกน original database

จากงานวิจัยที่มีการนำเสนออัลกอริทึมต่างๆ ที่นำมาใช้ในการเพิ่มขยายการค้นหากฎความสัมพันธ์ที่น่าสนใจสามารถสรุปได้ดังนี้คือ

**2.3.1 FUP based algorithm**

FUP เป็นงานวิจัยแรกที่นำเสนอเกี่ยวกับการ incremental updating technique ที่พัฒนาขึ้นมาเพื่อ maintenance association rules เมื่อมีการเพิ่ม transaction เข้าไปในฐานข้อมูลเดิม โดยลักษณะเด่นของ FUP Algorithm มีดังนี้คือ

1. นำความรู้ที่ได้จากค้นหากฎความสัมพันธ์โดยการไมน์นิ่งในฐานข้อมูลก่อนที่จะมีการเปลี่ยนแปลงมาใช้ เพื่อลดจำนวน transaction ที่จะต้องสแกนในฐานข้อมูลทั้งหมด จากนั้นจึงหาค่า support count ของข้อมูลต่างๆ ที่เกิดขึ้น ซึ่งแตกต่างจากอัลกอริทึม Apriori ที่จะต้องสแกนฐานข้อมูลทั้งหมด (ทั้งข้อมูลเก่าและข้อมูลใหม่ที่เปลี่ยนแปลง) เพื่อหากฎความสัมพันธ์ใหม่โดยไม่นำความรู้ที่ได้จากการหากฎความสัมพันธ์ในฐานข้อมูลเดิมมาใช้ให้เกิดประโยชน์ ดังนั้น FUP จึงเป็นแนวคิดแรกที่มีการนำความรู้ที่ได้จากการค้นหากฎความสัมพันธ์ก่อนหน้ามาใช้เพื่อลดการสแกนข้อมูลทั้งหมด

2. การหากฎความสัมพันธ์จะอยู่บนฐานของ Apriori algorithm คือมีการวนหาข้อมูลในลักษณะของ level-wise algorithm และมีการหา large itemset จาก candidate itemset

3. ใช้ minimum support เดียวกันในการหากฎความสัมพันธ์ FUP จะใช้ค่า minimum support เดียวกันในการค้นหา large k-itemset ของฐานข้อมูลเก่าและใหม่

อัลกอริทึมของ FUP (stands for Fast Update) จะอยู่บนฐานของ Apriori โดยเป็นอัลกอริทึมที่กล่าวถึงเทคนิคการทำ incremental updating ในกรณีการเพิ่มข้อมูลรายการเปลี่ยนแปลงเพียงกรณีเดียว ลักษณะการทำงานจะคล้ายกับ Apriori คือจะมีการวนรอบซ้ำ (iteration) เพื่อหาความสัมพันธ์ของข้อมูลเริ่มจาก 1-itemset ไปจนถึง k-itemset (k = 2, 3, …, n) ซึ่ง candidate itemsets ในแต่ละ iteration จะ based on large itemsets ที่พบใน iteration ก่อนหน้า FUP จะใช้ในการแก้ปัญหาด้านประสิทธิภาพในการปรับกฎความสัมพันธ์ (updated association rules) หลังจากที่มีการเพิ่ม transaction ใหม่เข้ามาในฐานข้อมูล จะนำกฎความสัมพันธ์ที่ได้ถูกค้นหาไว้ก่อนในฐานข้อมูลเดิมมาใช้ประโยชน์ เพื่อลดจำนวนข้อมูลที่จะต้องนำมาหากฎ

ความสัมพันธ์ โดยจะทำการสแกนฐานข้อมูลใหม่ซึ่งมีจำนวน transaction ที่เกิดขึ้นน้อยกว่าฐานข้อมูลเดิม แล้วจึงนำค่า support ที่ได้จากการสแกนฐานข้อมูลใหม่มารวมกับฐานข้อมูลเดิม โดยจะทำการพิจารณาหาค่าของlarge itemset ใหม่ (L′) ที่ได้จากการรวมกันของฐานข้อมูลเดิมและฐานข้อมูลที่เพิ่มเข้ามา (DB∪db) ด้วยค่า support ของ item ต้องไม่น้อยกว่า $s \times (D+d)$ (โดย s หมายถึง support, D หมายถึง จำนวนของ transaction ใน original database, d หมายถึง จำนวนของ transaction ใน incremental)

**ข้อดีของ FUP คือ**

1. นำความรู้ที่ได้จากการค้นหาในฐานข้อมูลเดิมมาใช้ร่วมกับรายการข้อมูลที่เพิ่มเข้าไปใหม่ เนื่องจากมีการใช้ minimum support ที่คงที่
2. ลดจำนวนของ candidate โดยการหาค่า candidate ที่ได้จากการสแกนข้อมูลรายการที่เพิ่มเข้าไปในฐานข้อมูลใหม่ (db) ไปหา ค่า support ในฐานข้อมูลเดิม โดยไม่จำเป็นต้องสแกนข้อมูลทั้งหมดในฐานข้อมูลเดิม
3. สามารถลดจำนวน itemset ที่ไม่พบว่าเป็น Frequent k-itemset ในฐานข้อมูลที่ปรับปรุง

**ข้อด้อยของ FUP คือ**

1. การสแกนฐานข้อมูลเดิมทุก k-itemset ที่พบ frequent itemset ในฐานข้อมูลใหม่ เนื่องจากในการไมน์นิ่งจากฐานข้อมูลเดิมจะเก็บเฉพาะ frequent k-itemset เท่านั้น ดังนั้นการหา new large itemset จะต้องทำการสแกนฐานข้อมูลเดิมทุกรอบของการหา frequent k-itemset ถึงแม้จะเป็นการสแกนเฉพาะค่าของ item ที่เป็นสมาชิกของ candidate ที่เป็น frequent k-itemset ที่ปรากฏในฐานข้อมูลใหม่

### 2.3.2 Negative border based algorithm

งานวิจัยนี้เป็นการศึกษาการ maintenance association rules เมื่อมีการเพิ่ม transaction ใหม่หรือลบ transaction เก่าออกจากฐานข้อมูล การหากฎความสัมพันธ์ในงานวิจัยนี้จะอยู่ในรูปแบบของ level-wise algorithm เช่นเดียวกับ Apriori แต่จะอาศัยหลักการของ negative border ที่ได้นำเสนอโดย Mannila and Toivonen [5]

สำหรับแนวคิดของ negative border จะเป็นการนำเสนอส่วนของซับเซตของ itemset ที่เป็นสมาชิกของ candidate k-itemset ($C_k$) โดยส่วนของสมาชิกของ Candidate k-itemset ที่มีค่ามากกว่าหรือเท่ากับค่า minimum support threshold จะจัดเก็บไว้ในส่วนของ Frequent k-itemset แต่ส่วนที่เหลือจะเรียกว่า Border set ทั้งนี้ Candidate k-itemset จะถูกสร้างขึ้นมาด้วยหลักการเดียวกับ APRIORI ที่มีการนำ Frequent k-1 itemset มาสร้าง Candidate k-itemset ดังนั้น Border set ที่ได้จะมีซับเซตที่เป็น frequent k-1 itemset โดยที่สมาชิกของ Border set ตัวนั้นไม่ได้เป็น frequent k-

itemset ความสัมพันธ์ระหว่าง candidate k-itemset($C_k$) ,frequent k-itemset ($L_k$) และ negative border ($NBd(L_k)$ สามารถแสดงได้ดังนี้

$$NBd(L_k) = C_k - L_k \quad \text{หรือ} \quad C_k = L_k \cup NBd(L_k)$$

ตัวอย่างเช่น

$C_1$ = { A, B, C, D, E, F}

$L_1$ = {A,B, C, F}

$NBd(L_1)$ = {D, E}

$C_2$ = { {A, B}, {A, C},{A,F},{B,C},{B,F},{C,F}}

$L_2$ = { {A, B}, {A,C},{A,F},{C,F}}

$NBd(L_2)$ = { {B,C},{B,F}}

แนวคิดนี้ได้มีผู้นำเสนอในส่วนของ incremental association rule 2 งานวิจัยคือ Thomas et al [6] และFeldman et al [12] โดยทั้ง 2 งานวิจัยมีแนวคิดที่จะปรับปรุงการทำงานของอัลกอริทึม FUP ที่มีการนำความรู้ที่ได้จากการไมน์นิ่ง frequent k-itemset ที่ได้จาก original database มาใช้เพื่อลดการไมน์นิ่งฐานข้อมูลทั้งหมด ด้วยการลดจำนวน itemset ที่ต้องทำการไมน์นิ่ง แต่ยังคงต้องสแกนฐานข้อมูลทั้ง original database และ incremental database ในทุกรอบของการค้นหา ด frequent k-itemset ดังนั้นเพื่อเป็นการลดจำนวนการค้นหา frequent k-itemset จึงได้มีการนำส่วนของ border set เข้ามาช่วย ในงานวิจัยทั้ง2 จึงมีการเก็บทั้ง frequent k-itemset และส่วนที่เป็น border k-itemset

หลักการทำงานของ Border set ในส่วนของ incremental association rule นั้นจะประกอบด้วย 2 คุณลักษณะหลัก ๆ คือ

1. ถ้าไม่มี Frequent sets ใหม่ที่เกิดขึ้นจะไม่มีการ access original database ดังนั้นจึงสามารถรับประกันได้ว่าการสแกนฐานข้อมูลทั้งหมดจะเกิดกรณีเดียวเท่านั้นคือเมื่อมี frequent sets ใหม่ โดยอัลกอริทึมที่ได้มีการนำเสนอก่อนหน้านี้จะไม่สามารถรับประกันได้
2. ในกรณีที่ต้องสแกนฐานข้อมูลทั้งหมดใหม่นั้นจำนวนรอบของการสแกนฐานข้อมูลจะมีจำนวนไม่มากเนื่องจากจะทำการสร้าง candidate (k+1)-itemset ให้ครอบคลุมเฉพาะ frequent k-itemset ใหม่เท่านั้น ดังนั้นจึงมี Candidate (k+1)- itemset เท่านั้นที่จะถูกสแกนเพื่อหาค่า support

**ข้อดี**

1. ลดจำนวนครั้งในการสแกนฐานข้อมูลเดิม

2. สามารถปรับปรุงค่า support ของ frequent k-itemset และ border k-itemset ได้รวดเร็วในกรณีที่ itemset ต่างๆ มีการเปลี่ยนแปลงน้อย หรือไม่มีการเปลี่ยนแปลงจาก border k-itemset มาเป็น frequent k-itemset

**ข้อด้อย**

1. เมื่อมีการเปลี่ยนแปลงในส่วนของ border itemset มาเป็น Frequent itemset จะต้องทำการ rescan ฐานข้อมูลทั้งหมดใหม่ ทำให้ใช้เวลานานกว่าการทำไมน์นิ่งข้อมูลทั้งหมดอย่าง APRIORI ดังที่พบจากการทดลองในงานวิจัย [13] เนื่องจากต้องทำการปรับปรุงข้อมูลที่เป็น frequent และ border set และหา candidate k-itemset สำหรับ frequent itemset ใหม่ ทำให้ใช้เวลานาน

2. ใช้พื้นที่จำนวนมากในการจัดเก็บ itemset ทั้งหมด เนื่องจากเป็นแนวคิดที่มีการจัดเก็บข้อมูลที่ได้จากการไมน์นิ่งทั้งที่เป็นสมาชิกของ Frequent itemsets และ negative border

3. เมื่อมี item ใหม่เพิ่มขึ้นจากการไมน์นิ่ง ซึ่งอยู่นอกเหนือจาก border และ frequent itemset เดิมที่มีจะไม่สามารถทำการไมน์นิ่ง item ใหม่นี้ได้เนื่องจากอัลกอริทึมไม่มีการรองรับในส่วนนี้

### 2.3.3 Expected frequent itemset algorithm

Expected frequent itemset เป็นงานวิจัยที่มีการนำเสนอแนวคิดในการลดจำนวนครั้งในการสแกน original database โดยจะมีการใช้ข้อมูลที่ได้จากการไมน์นิ่งในฐานข้อมูลเดิมทั้งส่วนที่เป็น Frequent itemset และในส่วนที่คาดว่าจะกลายมาเป็น Frequent itemset ได้เมื่อมีการเพิ่มฐานข้อมูลใหม่เข้ามาด้วยขนาดของข้อมูลในช่วงที่กำหนดเข้ามา (incremental database) ในฐานข้อมูลเดิม เรียก itemset ที่คาดว่าจะกลายมาเป็น Frequent itemset นี้ว่า Expected frequent itemset

แนวคิดของ Expected frequent itemset ที่ Hong et al [14] และ Amornchewin และ Kreesuradej [15] ได้นำเสนอแนวคิดที่แตกต่างจากการใช้ border set ซึ่งมีการเก็บข้อมูลจำนวนมากมาใช้เกณฑ์ในการพิจารณาเลือกเก็บเฉพาะ itemset ที่เป็น small itemset หรือ itemset ที่มีค่า support น้อยกว่า minimum support ที่กำหนดไว้ แต่มีโอกาสกลายมาเป็น frequent itemset ได้เมื่อมีการเพิ่มฐานข้อมูลใหม่เข้ามา โดยเรียกส่วนของ itemset นี้ว่า expected frequent itemset หรือ promising frequent itemset

การค้นหา promising frequent itemset นี้จะมีการเกณฑ์ที่ใช้ในการพิจารณา itemset ที่คาดว่าจะกลายเป็นfrequent itemset โดยอาศัยหลักทางสถิติของ itemset ที่เกิดขึ้นในฐานข้อมูลเดิมจะมีความแตกต่างน้อยหรือไม่เปลี่ยนแปลงกับฐานข้อมูลใหม่ที่เกิดขึ้น ด้วยการนำค่า support ที่เกิดขึ้น

สูงสุดจากการทำไมน์นิ่งฐานข้อมูลเดิมมาใช้ในการคำนวณหาเกณฑ์ที่เรียกว่า min_PL ด้วยสมการดังต่อไปนี้

$$min\_sup_{DB} - \left( \frac{maxsupp}{total\ size} \times inc\_size \right) \leq min\_PL < min\_sup_{DB} \quad (1)$$

โดย maxsupp หมายถึง ค่า support สูงสุดที่ได้จากการไมน์นิ่งในฐานข้อมูลเดิม

total size หมายถึง ขนาดของฐานข้อมูลเดิม

inc_size หมายถึง ขนาดของฐานข้อมูลใหม่ที่จะเพิ่ม

จากสมการที่ (1) เมื่อทำการไมน์นิ่ง 1-itemset จะทำให้เราทราบค่า support สูงสุดของ 1-itemset ที่เกิดขึ้น และสามารถนำมาใช้ในการหาขนาดของฐานข้อมูลใหม่ที่เพิ่มเข้ามา ได้ดังนี้

$$0 < min\_PL < min\_sup$$

$$0 < min\_PL = \frac{max\_sup\ p}{|DB| + inc\_size} * inc\_size < min\_sup$$

$$0 < inc\_size < \frac{min\_sup * |DB|}{max\_sup\ p - min\_sup} \quad (2)$$

ขนาดของข้อมูลที่คำนวณได้จะช่วยในการประมาณค่าของฐานข้อมูลใหม่ที่เพิ่มเข้ามา ทำให้สามารถ guarantee ได้ว่าด้วยขนาดของ incremental database ที่เพิ่มเข้ามาและจากสมมติฐานของสถิติการเกิดของ items ไม่แตกต่างหรือแตกต่างกันน้อย ทำให้สามารถนำค่า min_PL ที่ได้มาใช้ในการพิจารณาหา itemset ที่คาดว่าจะกลายมาเป็น Frequent k-itemset ได้ และจะช่วยลดหรือหลีกเลี่ยงการสแกน original database ลงไปได้ เนื่องจากนำทั้ง Frequent k-itemset และ Promising Frequent k-itemset ที่ได้มาสร้าง Candidate k+1 itemset เพื่อให้การค้นหา Expected Frequent itemset ครอบคลุมทุก itemset ที่คาดว่าจะเกิดเมื่อมีการเพิ่มฐานข้อมูลใหม่เข้ามา

แนวคิดนี้จะใช้ได้ดีเมื่อ itemset ที่พบใน incremental database เป็นสมาชิกของ frequent k-itemset หรือ expected k-itemset ทั้งนี้ในกรณีที่บาง expected frequent k-itemset กลายมาเป็น frequent k-itemset ใน Updated database ก็ไม่มีความจำเป็นต้องสแกน original database ใหม่

เมื่อมี itemsetใหม่เกิดขึ้น อัลกอริทึมจะทำการตรวจสอบ เพื่อลดการสแกนฐานข้อมูลได้เป็น 2 กรณี

1. กรณีที่ itemset ใหม่ที่เกิดขึ้นนั้น เกิดเฉพาะใน incremental database เนื่องจากในการ update 1-itemset จะสามารถตรวจสอบได้ว่า itemset ต่างๆ ที่เกิดขึ้นในฐานข้อมูลใด ซึ่งถ้าพบว่า itemset ใหม่นั้นปรากฏเพียงใน incremental database อัลกอริทึมนี้จะทำการสแกน candidate k+1 itemset สำหรับ itemset ใหม่ที่เกิดขึ้น ใน incremental database เท่านั้น ทำให้ไม่ต้องเสียเวลาในการสแกน Original database

2. กรณีที่ itemset ใหม่ที่เกิดขึ้นนั้นเป็นสมาชิกของ Candidate 1-itemset แต่ไม่เป็นสมาชิกของ frequent 1-itemset และ expected frequent 1-itemset ซึ่งจาก 1-itemset นี้จะทำให้ทราบถึง itemset ใหม่ที่เกิดขึ้นแตกต่างจาก original database ดังนั้นในกรณีนี้จะมีการสร้าง candidate k+1 itemset โดยทำการ join ระหว่าง itemset ใหม่ และfrequent k-itemset และ expected frequent k-itemset ใน original database เพื่อให้สามารถค้นหา frequent k+1 itemset ที่ครอบคลุม และถ้า candidate k-itemset ใดมีค่ามากกว่าหรือเท่ากับ minimum support *|db| หรือมากกว่าหรือเท่ากับ min_PL ของ updated database หรือ มากกว่าหรือเท่ากับ min_PL ของ original database itemset เหล่านี้จะถูกนำไปสแกนเพื่อหาค่า support count ใน original database

ดังนั้น การสแกน original database จะทำเพียงกรณีเดียวเท่านั้นคือ กรณีที่ 2 ซึ่งจำนวนของ itemset จะมีจำนวนน้อยเนื่องจาก itemset ใหม่ที่จะถูกนำมาสแกนใน original database จะต้องเป็น frequent itemset ที่ปรากฏใน incremental database หรือ expected frequent itemset โดย itemset ที่มีค่า support น้อยกว่า min_PL ของ updated database จะถูก prune ออกไป

ตัวอย่าง original database ซึ่งประกอบด้วย 10 transaction โดยฐานข้อมูลจะประกอบด้วย item 5 ตัว คือ {A, B, C, D, E} ด้วย minimum support = 40% ดังแสดงในรูปที่ 2.5

| TID | List of item |
|---|---|
| 1 | A, B, E |
| 2 | B, D |
| 3 | B,C |
| 4 | A, B,D |
| 5 | A, C |
| 6 | B, C |
| 7 | A, C |
| 8 | A, B, C, E |
| 9 | A, B, E |
| 10 | A,C |

| Itemset | support |
|---|---|
| A | 7 |
| B | 7 |
| C | 6 |
| D | 2 |
| E | 3 |

**รูปที่ 2.5** Transaction ใน original database และ Candidate 1-itemset

จากรูปที่ 2.5 พบว่า max_supp มีค่าเท่ากับ 7 จากสมการ (1) และ (2) สามารถคำนวณ incremental size ได้คือ

$$min\_sup = 0.4 * 10 = 4$$

$$inc\_siize < \frac{4*10}{7-4} \approx 13$$

ถ้าขนาดของ incremental database คือ 3 transactions ดังนั้นค่า min_PL สามารถคำนวณได้ดังนี้

$$min\_PL = 4 - \frac{7}{10} * 3 \approx 2$$

จากรูปที่ 2.5 จะได้ frequent 1-itemset ($L_1$) คือ {A, B, C} และ Expected Frequent 1-itemset ในที่นี้แทนด้วย $PL_1$ คือ {D,E}

| C2 | support |
|---|---|
| AB | 4 |
| AC | 4 |
| AD | 1 |
| AE | 3 |
| BC | 3 |
| BD | 2 |
| BE | 3 |
| CD | 0 |
| CE | 1 |
| DE | 0 |

| L2 | Support |
|---|---|
| AB | 4 |
| AC | 4 |
| PL2 | support |
| AE | 3 |
| BC | 3 |
| BD | 2 |
| BE | 3 |

**รูปที่ 2.6** แสดง Candidate 2-itemset (C2) ,Frequent 2-itemset และ Expected Frequent 2-itemset

| C3 | Support |
|---|---|
| ABC | 1 |
| ABE | 3 |
| ACE | 1 |
| BCD | 0 |
| BCE | 0 |
| BDE | 0 |

| PL3 | support |
|---|---|
| ABE | 3 |

**รูปที่ 2.7** แสดง Candidate 3-itemset ($C_3$) ,Frequent 3-itemset ($L_3$)และ Expected Frequent 3-itemset ($PL_3$)

เมื่อมีการเพิ่ม incremental database จะทำการ update ค่า support count ของ frequent k-itemset และ Expected Frequent k-itemset โดยจะทำการคำนวณค่า min_PL ของ updated database ใหม่ ดังสมการที่ 3

$$min\_PL_{DB\cup db} = min\_sup_{DB\cup db} - \left(\frac{max\,supp}{total\_size} \times inc\_size\right) \quad (3)$$

**ข้อดี**

1. ลดจำนวนการสแกนฐานข้อมูลเดิม เนื่องจากข้อมูลบางตัวที่เป็นสมาชิกของ Expected Frequent itemset อาจกลายเป็น Frequent itemset ได้ ด้วยข้อมูลที่ได้จากฐานข้อมูลใหม่เพียงเล็กน้อย

2. ลดเวลาในการหา itemset ใหม่โดยใช้ expected value ในที่นี้คือ min_PL มาช่วยในการพิจารณา itemset ที่คาดว่าจะกลายเป็น Frequent itemset

3. เมื่อมีการเปลี่ยนแปลงจาก expected frequent itemset มาเป็น frequent itemset ไม่จำเป็นต้องเข้าไปทำการไมน์นิ่งใหม่เนื่องจากหา combination ของ frequent itemset และ expected frequent itemset รองรับไว้แล้ว

**ข้อเสีย**

1. การเพิ่มขนาดของรายการข้อมูลเข้าไปในฐานข้อมูลเดิมทำได้จำกัด เนื่องจากขนาดของข้อมูลต้องอยู่ในช่วงที่ได้จากการคำนวณซึ่งมีช่วงขนาดของข้อมูลที่ไม่มาก

2. การปรับปรุงข้อมูลที่ได้จากการไมน์นิ่งมีจำนวนมาก เมื่อมี incremental database เข้ามาจะต้องทำการปรับปรุง itemset ที่เป็น Frequent k-itemset และ Expected Frequent k-itemset และในกรณีที่ค่าของ expected value ที่คำนวณได้มีค่า = 1 จะเป็นกรณีที่ต้องเก็บข้อมูลทุก itemset ที่เกิดขึ้นในการไมน์นิ่ง ซึ่งอาจใช้เวลานานในการปรับปรุงทุก itemset

## 2.4 ทฤษฎีเบอร์นูลลี (Bernoulli Theorem)

ทฤษฎีทางสถิติที่ใช้ในการพิจารณาการทดลอง ซึ่งผลการทดลอง 1 ครั้งมีผลอย่างใดอย่างหนึ่งใน 2 แบบเท่านั้น คือ ความสำเร็จ (success) หรือ ความไม่สำเร็จ (failure) เช่น โยนเหรียญ 1 อัน จะปรากฏผลเป็นหัวหรือก้อย ซึ่งอาจถือว่าการขึ้นหัวเป็นผลสำเร็จ ส่วนการขึ้นก้อยเป็นความไม่สำเร็จ สำหรับตัวอย่างของการโยนเหรียญนี้ ความน่าจะเป็นที่จะขึ้นหัวเท่ากับ $\frac{1}{2}$ และความน่าจะเป็นที่จะขึ้นก้อยเท่ากับ $\frac{1}{2}$ โดยความน่าจะเป็นที่จะเกิดความสำเร็จเท่ากับ p และ ความน่าจะเป็นที่จะเกิดผลความไม่สำเร็จเท่ากับ q ซึ่ง p+q = 1

การค้นหากฎความสัมพันธ์ได้มีการนำเบอร์นูลลีมาประยุกต์ใช้ในการพิจารณา itemset ที่มีโอกาสจะกลายมาเป็น Frequent itemset เมื่อทำการเพิ่มรายการข้อมูลใหม่เข้ามาในฐานข้อมูลเดิม ด้วยสูตรการคำนวณดังต่อไปนี้

$$P(x) = \binom{n+m}{x} \cdot p^{x} \cdot (1-p)^{n+m-x}$$

โดย P หมายถึงค่าความน่าจะเป็น

(n+m) หมายถึง ขนาดของข้อมูลที่จะทำการทดลอง

p หมายถึง ค่าความน่าจะเป็นที่เกิดความสำเร็จจากการทดลอง

(1-p) หมายถึง ค่าความน่าจะเป็นที่การทดลองไม่มีความสำเร็จ

x หมายถึง จำนวนครั้งที่การทดลองจะเกิดความสำเร็จ

ในส่วนของ association rule mining ได้นำ Bernoulli distribution มาใช้ในการประมาณค่า itemset ที่จะเป็น Frequent itemset หรือ small itemset โดย itemset ที่เป็น Frequent จะเป็นเหตุการณ์ที่สำเร็จ (success) ส่วนitemset ใดที่ไม่เป็น Frequent itemset จะเรียกว่า ความไม่สำเร็จ (Failure) โดย Zhang et al [16] ได้นำ Bernoulli distribution มาใช้ในการประมาณค่าการเกิด frequent itemset จากข้อมูลสุ่มของ original database ด้วย method ของ Central limit theorem มาช่วยเพิ่มประสิทธิภาพ แนวคิดนี้พัฒนาขึ้นมาภายใต้ข้อจำกัดของเวลา ซึ่งเวลาเป็นสิ่งที่สำคัญ

มากกว่าความถูกต้องที่สูง เช่น การลงทุนในตลาดหุ้น ถ้านักลงทุนสามารถ ที่จะประมาณค่าของ frequent itemset จากข้อมูลใน stock database ได้อย่างรวดเร็ว จะช่วยให้การตัดสินใจที่จะลงทุนได้รวดเร็วและถูกต้อง แทนที่จะต้องทำการค้นหา frequent itemset จากฐานข้อมูลทั้งหมด

การเพิ่มขยายการค้นหากฎความสัมพันธ์ของงานวิจัยนี้ สามารถค้นหากฎความสัมพันธ์ของฐานข้อมูลได้อย่างมีประสิทธิภาพในกรณีของการเพิ่ม incremental database เข้าไป โดยจะสามารถลดจำนวน itemset ที่จะต้องนำไปสแกนใน original database และสามารถหา frequent itemsetใหม่ได้อย่างมีประสิทธิภาพโดยนำ Bernoulli Theorem มาประยุกต์ใช้ดังจะกล่าวรายละเอียดต่างๆ ในบทต่อไป

# บทที่ 3
# อัลกอริทึมสำหรับการเพิ่มขยายการค้นหากฎความสัมพันธ์โดยอาศัยหลักของความน่าจะเป็น

ปัจจุบันพบว่าข้อมูลต่างๆ มีการปรับเปลี่ยนตลอดเวลาทั้งในด้านการเพิ่ม, การลบหรือแก้ไขข้อมูล การเปลี่ยนแปลงข้อมูลให้มีความทันสมัยเป็นปัจจุบันจะมีผลต่อการเปลี่ยนแปลงกฎความสัมพันธ์ที่ได้ค้นหาไว้แล้ว ดังนั้นจึงต้องทำการค้นหากฎความสัมพันธ์ใหม่เมื่อข้อมูลเกิดการเปลี่ยนแปลงกับฐานข้อมูล

การเพิ่มขยายการค้นหากฎความสัมพันธ์เป็นอัลกอริทึมที่ใช้ในปรับปรุงกฎความสัมพันธ์ของข้อมูลในฐานข้อมูลขนาดใหญ่ เมื่อมีการเพิ่ม transaction ใหม่เข้าไปใน original database ในที่นี้จะเรียกส่วนของรายการข้อมูลใหม่ที่เพิ่มเข้าไปนี้ว่า incremental database (db) ซึ่ง transaction ที่เพิ่มเข้ามานี้จะมีผลต่อกฎความสัมพันธ์ที่ได้จากการทำดาต้าไมน์นิ่งใน original database (DB) โดยอาจทำให้กฎที่มีอยู่ไม่มีความถูกต้องเมื่อทำการค้นหากฎความสัมพันธ์ที่ปรากฏในฐานข้อมูลที่ได้รับการปรับปรุงใหม่ทั้งหมด (updated database : UP)

การปรับปรุงฐานข้อมูลใหม่ อาจหมายถึง การเพิ่มฐานข้อมูลใหม่เข้าไปในฐานข้อมูลเดิม, การลดรายการในฐานข้อมูลเดิม และกรณีที่มีทั้งการลดและเพิ่มรายการในฐานข้อมูลเดิม สำหรับในงานวิจัยนี้จะเป็นการค้นหากฎความสัมพันธ์ในกรณีของเพิ่มรายการใหม่เข้าไปในฐานข้อมูลเดิมเท่านั้น

เมื่อฐานข้อมูลที่ได้จากการปรับปรุงได้ผ่านกระบวนการค้นหากฎความสัมพันธ์แล้ว อาจเกิดการเปลี่ยนแปลง frequent itemset เดิมทำให้เกิดการสแกนฐานข้อมูลเดิมใหม่ โดยทั่วไปขนาดของฐานข้อมูลเดิมมักจะมีขนาดใหญ่กว่าฐานข้อมูลใหม่ที่เพิ่มเข้ามา ดังนั้นการสแกนฐานข้อมูลเดิมจะมีผลต่อความถูกต้องของกฎความสัมพันธ์, เวลาที่ใช้ในการประมวลผล รวมถึงทำให้เกิดการสิ้นเปลืองทรัพยากรในการค้นหา frequent itemset ที่พบในฐานข้อมูลใหม่แต่เป็น small itemset ใน original database

ซึ่งกรณีที่ต้องทำการสแกน original database สำหรับค้นหา frequent itemset ที่พบจาก incremental database นี้ จะพบว่าสมาชิกของ frequent itemset ใหม่ไม่ทุกตัวที่จะกลายมาเป็น frequent itemset ใน updated database ดังนั้นการที่ต้องทำการสแกน original database ใหม่นั้นจะเป็นการเสียเวลาที่ใช้ในการค้นหาเนื่องจาก itemset ที่เกิดใน incremental database จะมีจำนวนมากกว่าที่พบใน original database ด้วยการพิจารณาค่าของ frequent itemset จาก $min\_sup_{db} * |db|$

อัลกอริทึม Apriori เป็นอัลกอริทึมที่ได้รับความนิยมในการค้นหากฎความสัมพันธ์ของข้อมูลในฐานข้อมูลขนาดใหญ่ การไมน์นิ่งเป็นในลักษณะที่เรียกว่า level-wise search ดังนั้นในแต่ละ k-itemset จะเก็บ frequent itemset ที่เกิดจากการไมน์นิ่ง เพื่อใช้ในการสร้างกฎความสัมพันธ์ที่ได้จากการดึงความรู้ที่ซ่อนอยู่ในฐานข้อมูลไว้

การที่ข้อมูลต่างๆ ในฐานข้อมูล อาจเกิดการเปลี่ยนแปลงขึ้น ในลักษณะของฐานข้อมูลที่เรียกว่า dynamic database การไมน์นิ่ง dynamic database จากหลักการทำงานของ Apriori ถึงแม้ผลการค้นหากฎความสัมพันธ์ที่ได้จะมีความถูกต้องด้วยกระบวนการไมน์นิ่งฐานข้อมูลที่ปรับปรุงใหม่ทั้งหมด โดยไม่ได้นำความรู้เดิมที่ได้จากการไมน์นิ่ง original databaseมาใช้ ทำให้เสียเวลาในการค้นหา frequent k-itemset ใหม่ทั้งหมด

เพื่อลดจำนวนการค้นหา frequent k-itemset ใหม่ทั้งหมด ด้วยหลักการของ Apriori อัลกอริทึม FUP ได้นำเสนอวิธีการแก้ปัญหาด้วยการนำความรู้ที่ได้จากการไมน์นิ่ง original database มาใช้ให้เกิดประโยชน์ คือนำ frequent k-itemset เดิมที่ได้จากการไมน์นิ่งมาทำการปรับปรุงค่า support count ที่ปรากฏในฐานข้อมูลใหม่ ซึ่งเป็นการลดการค้นหา frequent itemset ใน original database ซ้ำ และเมื่อเกิด frequent k-itemset ใน incremental database จะมีการนำ frequent k-itemset ใหม่นี้มาทำการสแกนเพื่อหาค่า support count ใน original database เพื่อให้ได้frequent k-itemset ที่ถูกต้องสำหรับสร้างกฎความสัมพันธ์

ความรู้ที่ได้จากการไมน์นิ่งในฐานข้อมูลทั้งหมดจะเป็น frequent itemset เท่านั้น ดังนั้น Thomas et al [6] และFeldman et al [12] ได้นำแนวคิดของ negative border มาพัฒนาอัลกอริทึม โดยจะทำการเก็บข้อมูลของ frequent k-itemset และส่วนที่เรียกว่า negative border k-itemset ไว้ทั้งหมดเพื่อลดการสแกน original database อัลกอริทึมนี้จะทำงานได้ดีในกรณีที่ไม่พบว่ามี frequent k-itemset ใหม่เกิดขึ้น แต่ในทางกลับกัน ถ้าพบว่ามี frequent k-itemset ใหม่เกิดขึ้นจะต้องมีการสแกนฐานข้อมูลปรับปรุงทั้งหมด ดังที่งานวิจัย [13] ได้ทำการเปรียบเทียบประสิทธิภาพการทำงานระหว่างอัลกอริทึม negative border [6] เทียบกับอัลกอริทึมApriori และพบว่าการทำงานของอัลกอริทึม negative border เหมาะกับ item ที่ไม่มากและ execution time ของ negative border ในกรณีที่พบ frequent k-itemset ใหม่และต้องสแกนฐานข้อมูลทั้งหมดนั้นใช้เวลาไม่แตกต่างหรืออาจจะนานกว่าการค้นหากฎความสัมพันธ์ด้วย Apriori นอกจากนี้ทั้งงานวิจัยของ Thomas et al และFeldman et al ไม่มีการนำเสนอในส่วนของการ recover frequent k-itemset ใหม่ซึ่งอาจเกิดขึ้นได้ใน incremental database เมื่อค่า minimum support threshold มีค่าน้อย

เพื่อลดปัญหาการจัดเก็บข้อมูลและการสแกนฐานข้อมูลใหม่ทั้งหมดในกรณีที่พบ frequent itemset ใหม่ Hong et al [14] และ Amornchewin และ Kreesuradej [15] ได้นำเสนอแนวคิด expected frequent itemset เพื่อลดปัญหาการจัดเก็บข้อมูลทุกตัวที่เป็นสมาชิกของ candidate k-itemset ที่ไม่เป็น frequent k-itemset แนวคิดนี้จะมีการนำเสนอการหา expected frequent itemset ที่

มีโอกาสจะกลายมาเป็น frequent itemset เมื่อมี incremental database เพิ่มเข้ามา ทั้งนี้เพื่อลดหรือหลีกเลี่ยงการสแกน original database โดยจะมีการคำนวณหาขนาดของ incremental database ที่ใช้ในการประมาณค่าของ expected frequent itemset เพื่อให้สามารถทำงานได้อย่างมีประสิทธิภาพ แนวคิดนี้จะให้ความสำคัญกับ expected frequent k-itemset เช่นเดียวกับ frequent k-itemset ยกเว้นในส่วนของการสร้างกฎความสัมพันธ์ที่จะใช้ frequent k-itemset ($k \geq 2$) มาสร้างกฎความสัมพันธ์ ดังนั้น expected frequent k-itemsetจะถูกนำมาใช้ในการสร้าง candidate k-itemset ในขั้นตอนของการ join ด้วยหลักการเดียวกับ apriori ซึ่งทำให้จำนวน itemset ที่ต้องจัดเก็บมีจำนวนมาก

สำหรับงานวิจัยนี้จะเป็นการนำเสนอวิธีการแก้ปัญหาการทำดาต้าไมน์นิ่งในการหากฎความสัมพันธ์ที่ได้จากฐานข้อมูลที่ปรับปรุงใหม่จากการเพิ่มรายการข้อมูลเข้าไปในฐานข้อมูลเดิม เพื่อลดจำนวนครั้งและจำนวน itemset ที่จะต้องสแกนในฐานข้อมูลเดิม ด้วยแนวคิดดังนี้

1. การใช้ค่าความน่าจะเป็นที่ได้จากการคำนวณ โดยอาศัยหลักการของ Bernoulli trial มาช่วยในการค้นหา itemset ที่มีโอกาสจะกลายเป็น frequent itemset ได้เมื่อมีรายการจากฐานข้อมูลใหม่เพิ่มเข้ามา เรียก itemset นี้ว่า expected frequent itemset โดยค่าของ expected frequent itemset ได้จะต้องผ่านค่ากำหนดของค่าความน่าจะเป็นอย่างน้อยที่ผู้ใช้เป็นผู้กำหนด (user-defined) คือ $prob_{pl}$ ด้วยค่าความน่าจะเป็นของ itemset ที่มีโอกาสเป็น frequent itemset แน่นอนจะมีค่าค่อนข้างสูงหรือใกล้เคียง 1 ในขณะที่บาง itemset ที่มีค่าความน่าจะเป็นที่ผ่าน $prob_{pl}$ อาจจะกลายเป็น frequent itemset ได้เมื่อมีค่า support count จำนวนหนึ่งปรากฏใน incremental database ทำให้สามารถลดจำนวน expected frequent itemset ที่จะเก็บไว้สำหรับไมน์นิ่งในครั้งต่อไป

2. ในกรณีที่การไมน์นิ่งค้นพบ frequent itemset ใหม่ ซึ่งในกรณีนี้จะหมายถึง itemset ที่เป็น small itemset ใน original database ถ้าค่า support count ของ frequent itemset นี้มีค่าจำนวนหนึ่งที่มากกว่าค่า minimum support threshold ของ incremental database การจะทราบว่า frequent itemset ใหม่นี้จะเป็น frequent itemset ใหม่ใน updated database ได้จะสามารถทำได้ด้วยการสแกน original database ดังที่พบในงานวิจัยของ FUP

งานวิจัยนี้ได้นำเสนอแนวคิดที่จะลดจำนวน frequent itemset ใหม่ที่จะนำไปสแกนใน original database โดยนำค่า minimum expected frequent ที่ได้จากการไมน์นิ่ง original database มาใช้ในการประมาณค่า support count สูงสุดให้กับfrequent itemset ใหม่ หรือเป็น small itemset ใน original database

จากที่พบว่า frequent itemset ใหม่เป็น small itemset ใน original database ซึ่งค่า support count สูงสุดของ small itemset นี้ที่เป็นไปได้จะต้องน้อยกว่าค่า minimum expected frequent แน่นอน ดังนั้นค่าประมาณสูงสุดที่เป็นไปได้ของ small itemset นี้คือ minimum expected frequent ลบ 1 เมื่อทราบค่าประมาณสูงสุดแล้วเราจะนำมาพิจารณาความเป็นไปได้หรือโอกาสที่ frequent itemset ใหม่นี้จะกลายมาเป็น frequent itemset ใน updated database ได้โดยนำค่า คือ minimum expected

frequent -1 มาบวกกับค่า support count ของ frequent itemset ใหม่ ถ้าผลบวกที่ได้มากกว่าหรือเท่ากับ $min\_sup_{UP}$ ซึ่งจัดว่าเป็น itemset ที่มีโอกาสจะเป็น frequent itemset หรือ expected frequent itemset ใน updated database ดังนั้นจะเก็บไปสแกนใน original database ในขั้นตอนสุดท้ายสำหรับทุก k-itemset

3. การค้นหา frequent i temset สำหรับ itemset ใหม่ที่เกิดขึ้น ในที่นี้หมายถึง itemset นั้นๆ ปรากฏแต่เพียงใน incremental database เนื่องจากถ้าการไมน์นิ่งด้วยค่า minimum support threshold ที่น้อย อาจทำให้ itemset ใหม่ที่เพิ่งปรากฏใน incremental database กลายเป็น frequent itemset ได้ในที่นี้เปรียบเสมือนสินค้าใหม่ที่เพิ่งผลิตออกมาและได้รับความนิยมมาก ดังนั้นนอกจากการปรับปรุงค่า support count ของ frequent itemset และ expected frequent itemset แล้วงานวิจัยนี้จะครอบคลุมการค้นหา frequent itemset ใหม่ที่เกิดขึ้นจาก incremental database ที่เพิ่มเข้ามาได้อย่างถูกต้องและมีประสิทธิภาพ โดยอัลกอริทึมที่นำเสนอนี้จะสามารถตรวจสอบได้ว่า item ใดเป็น item ใหม่ที่เกิดใน incremental database จากการ update 1-itemset ทำให้ทราบว่าถ้าเป็น item ใหม่ที่เกิดใน incremental databaseและไม่มีความจำเป็นต้องไปสแกนใน original database จึงแตกต่างจากอัลกอริทึมของ FUP นอกจากนี้การค้นหา frequent itemset ใหม่นี้จะแตกต่างจาก negative border ที่ไม่มีในส่วนของการค้นหา frequen t itemset ใหม่ที่นอกเหนือจาก frequent k - itemset และ Boder k-itemset ที่มี

ดังจะมีการนำเสนอขั้นตอนการทำงานในส่วนต่อไปดังนี้

### 3.1 Probability-based Incremental Association Rule Discovery Algorithm

งานวิจัยนี้เป็นการนำเสนอแนวคิดของการใช้ค่าความน่าจะเป็นมาช่วยในการประมาณค่า itemset ที่คาดว่าจะกลายเป็น frequent itemset โดยนำทฤษฎีของเบอร์นูลลีมาประยุกต์ใช้ในการพิจารณาการเกิดของ frequent itemset ซึ่งประกอบด้วย 2 เหตุการณ์คือสำเร็จและไม่สำเร็จ ถ้าพบว่า itemset ใดที่เป็น frequent itemset และเหตุการณ์ที่ itemset ใดเป็น small itemset หรือไม่ผ่านค่า minimum support threshold เป็นความไม่สำเร็จ

ด้วยแนวคิดของทฤษฎีเบอร์นูลลีในงานวิจัยนี้ได้นำมาประยุกต์ใช้ในการประมาณค่าของ itemset ที่คาดว่าจะกลายเป็น frequent itemset โดยนำค่าสถิติการเกิดของ itemset ต่างๆ ที่ปรากฏใน original database ซึ่งในที่นี้ให้มีขนาดเท่ากับ m transactions ด้วยค่าความน่าจะเป็นที่ itemset นั้นๆ ปรากฏใน original database ซึ่งหมายถึงค่าความสำเร็จที่ itemset นี้ปรากฏใน original database แทนด้วยค่า p คำนวณได้ดังนี้

$$p = \frac{sup\ port\ count}{|DB|}$$

โดย p หมายถึง ค่าความน่าจะเป็นที่เกิดความสำเร็จจากการทดลอง

|DB| หมายถึงขนาด original database ในที่นี้คือ m transactions

จากทฤษฎีของเบอร์นูลลีนี้สามารถนำมาใช้ในการประมาณค่าของการเกิดของ itemset ต่างๆ ที่มีโอกาสจะกลายเป็น frequent itemet ใน updated database เมื่อมี incremental ขนาด n transations เพิ่มเข้าไปใน original database ขนาด m transations ด้วยจำนวน support count x จำนวนที่จะทำให้ item นี้เป็น frequent itemset ใน updated database ในที่นี้ ค่า support count x นี้ คำนวณได้ดังนี้

x = (minimum support * (|DB|+|db|) - support count DB

จากค่าต่างๆ ข้างต้นสามารถนำมาคำนวณค่าความน่าจะเป็นที่ item A จะเป็น frequent itemset ใน updated database ขนาด n+m transaction ได้ดังสมการ(1)

$$P(A) = \binom{n+m}{x} \cdot p^{x} \cdot (1-p)^{n+m-x} \qquad (1)$$

โดยทั่วไป การพิจารณาว่า itemset ใดจะเป็น frequent itemset ได้นั้นจะดูได้จากค่า support count ของ itemset นั้นที่ปรากฏในฐานข้อมูลมีค่ามากกว่าหรือเท่ากับค่า minimum support threshold ที่ผู้ใช้กำหนด แต่ในบาง itemset ที่มีค่า support count น้อยกว่าค่า minimum support threshold นั้นอาจแต่มีค่าความน่าจะเป็นของการเกิดที่มากพอที่จะผ่านค่าความน่าจะเป็นที่ผู้ใช้กำหนดเมื่อมีการเพิ่ม incremental database ที่ขนาดหนึ่ง ในที่นี้ค่าความน่าจะเป็นที่ผู้ใช้กำหนดเรียกว่า $prob_{pl}$ ซึ่งเป็นค่าประมาณการที่ผู้ใช้ยอมรับได้ว่า itemset นี้มีโอกาสที่เกิดขึ้นอย่างน้อย k ค่า เมื่อมี incremental database ขนาด n transaction เพิ่มเข้ามาใน original database ขนาด m transaction ได้ด้วยค่าความน่าจะเป็นที่คำนวณได้ โดยอาศัยความน่าจะเป็น p ซึ่งเป็นค่าที่เกิดจริงใน original database โดยในแต่ละ itemset สามารถคำนวณหาค่าความน่าจะเป็นที่จะเกิดใน updated database อย่างน้อย k ค่าได้ดังสมการที่ (2)

$$P(x<k)_{item} = \sum_{x=0}^{k-1} \binom{n+m}{x} p^{x}(1-p)^{n+m-x} \qquad (2)$$

(n+m) หมายถึง ขนาดของข้อมูลของ updated database ด้วยขนาดของ original data (|DB|) ขนาด m transactions และ incremental database (|db|) ขนาด n transactions

p หมายถึง ค่าความน่าจะเป็นที่เกิดความสำเร็จจากการทดลอง ซึ่งคำนวณได้ดังนี้

$$p = \frac{sup\ port\ count}{|DB|}$$

(1-p) หมายถึง ค่าความน่าจะเป็นที่การทดลองไม่มีความสำเร็จ

k หมายถึง จำนวนครั้งที่การทดลองจะเกิดความสำเร็จ คำนวณได้ดังนี้

$$k = min\_sup_{UP} - sup\ port\ count_{DB}$$

สำหรับค่าความน่าจะเป็นที่คำนวณได้จากสมการ (2) จะเป็นค่าความน่าจะเป็นอย่างน้อยที่จะเกิดด้วยค่า support count อย่างน้อย k ค่า เมื่อมีการเพิ่ม incremental database ขนาด n transaction เข้าไป ดังนั้นค่าความน่าจะเป็นของ itemset นี้จะกลายเป็น frequent itemset ใน updated database ด้วยค่าความน่าจะเป็นเท่าใด สามารถคำนวณได้ดังสมการ (3)

$$P\left(x \geq k\right)_{item} = 1 - P\left(x < k\right)_{item} \qquad (3)$$

โดย $P\left(x \geq k\right)_{item}$ หมายถึง ความน่าจะเป็นที่ item จะมีโอกาสที่ค่า support count x มีค่ามากกว่า k

k หมายถึง ค่าความแตกต่างระหว่าง $\text{min\_sup}_{UP}$ และค่า support count ของ item ($\text{support count}_{item}$) ที่จะทำให้ item นั้นๆ กลายมาเป็น frequent itemset ใน updated database โดยค่า $k \geq 0$ แต่ถ้าค่าที่คำนวณได้มีค่าน้อยกว่า 0 จะถูกปรับให้เป็น 0

สำหรับงานวิจัยนี้จะเป็นการนำเสนออัลกอริทึมที่มีประสิทธิภาพในการไมน์นิ่งฐานข้อมูล โดยสามารถลดการสแกนฐานข้อมูลใหม่ทั้งหมด จากแนวคิดที่มีการนำความรู้จากการไมน์นิ่ง original database ด้วยการจัดเก็บทั้งส่วนที่เป็น frequent k-itemset และส่วนที่คาดว่าจะกลายเป็น frequent k-itemset โดยอาศัยหลักการของความน่าจะเป็นเข้ามาช่วยในการกรองข้อมูลที่คาดว่าจะกลายเป็น frequent k-itemset จริงเมื่อมีการเพิ่มฐานข้อมูลใหม่เข้ามา นอกจากนี้ยังสามารถค้นหากฎความสัมพันธ์ที่เกิดจาก items ใหม่ที่เกิดขึ้นในฐานข้อมูลใหม่ได้อย่างถูกต้องสมบูรณ์

โดยงานวิจัยนี้จะแบ่งการทำงานออกเป็น 2 ส่วนสำคัญดังนี้

1. การค้นหา frequent itemset และ expected frequent itemset ใน original database

ด้วยค่าทางสถิติของ items นั้นที่เกิดขึ้นจริงในฐานข้อมูลจะนำมาใช้ในการทำนายความน่าจะเป็นที่ items นั้นๆ จะเกิดขึ้น จากการคำนวณในสมการ (3) เมื่อมีการกำหนดขนาดของ incremental database ที่เพิ่มเข้ามาแล้วนั้น อัลกอริทึมนี้ได้มีการนำเสนอค่าที่กำหนดโดยผู้ใช้เพิ่มอีก 1 ค่าเพื่อใช้ในการพิจารณา itemset ที่คาดว่าจะเป็น frequent itemset ใน updated database เรียกค่านี้ว่า minimum expected frequent และเรียก itemset ที่มีค่าความน่าจะเป็นจากการคำนวณและมีค่าความน่าจะเป็นผ่าน minimum expected frequent itemsetนี้ว่า expected frequent itemset โดย expected frequent itemset นี้จะมีค่า support count น้อยกว่า minimum support threshold แต่มีค่า support count มากกว่าค่า minimum expected frequent itemset ในส่วนของ original database จะเก็บ itemset ที่เป็น frequent itemset และ Expected frequent itemset เพื่อลดการสแกน original database สำหรับ itemset ที่เป็น expected frequent itemset ทำให้การ update ข้อมูลต่างๆ ใช้เวลาน้อยลง

นอกจากนี้เมื่อเราทราบค่า incremental database ที่แน่นอน ทำให้เราสามารถคำนวณค่าความน่าจะเป็นที่ยอมรับได้ว่า itemset นี้มีโอกาสที่จะกลายมาเป็น frequent itemset ใน updated database นั้นควรจะต้องเกิดใน original database เท่าไร ซึ่งในที่นี้จะเรียก itemset ที่เรายอมรับนี้ว่า Promising Frequent Itemset ทั้งนี้เพื่อไม่ให้มีการเก็บจำนวน itemsetนี้มากเกินไปเราจะกำหนด user defined อีก 1 ตัวคือค่าความน่าจะเป็นที่ยอมรับได้ หรือ P เมื่อมีการเพิ่ม incremental database เข้าไปหรือใน updated database เช่น ถ้ากำหนดค่าความน่าจะเป็นที่ยอมรับให้เป็น Frequent itemset ได้ ที่ P = 0.06 นั่นหมายความว่าใน 100 ครั้งจะเกิด itemset นี้อย่างน้อย 6 ครั้ง ใน updated database จะเก็บ itemset ที่มีค่าความน่าจะเป็นมากกว่าหรือเท่ากับ P =0.06 เมื่อมีการเพิ่ม incremental database เข้าไป ซึ่งจะต้องเป็น itemset ที่ค่า support count น้อยกว่า min_sup เป็น Promising frequent itemset

ตัวอย่างการคิดค่าความน่าจะเป็น ด้วยทฤษฎีเบอร์นูลลี

การหาค่าความน่าจะเป็นของ item ใน original database โดยเมื่อทราบ candidate 1-itemset และค่า support count แล้วดังรูปที่ 3.1 จะนำค่า support ที่มีไปหา ค่าความน่าจะเป็นที่จะเกิดใน updated database โดยทำการเพิ่ม incremental database ด้วย size ที่เท่าๆ กัน เช่น ถ้า |DB| =10 ,|db| = 5 , % min_sup = 40% ดังนั้น $min\_sup^{DB}$ = 4 , $min\_sup^{db}$ =2 และ $min\_sup^{UP}$ = 6

| TID | List of item |
|---|---|
| 1 | A, B, E |
| 2 | B, D |
| 3 | B,C |
| 4 | A, B,D |
| 5 | A, C |
| 6 | B, C |
| 7 | A, C |
| 8 | A, B, C, E |
| 9 | A, B, E |
| 10 | A,C |

| Itemset | support |
|---|---|
| A | 7 |
| B | 7 |
| C | 6 |
| D | 2 |
| E | 3 |

**รูปที่ 3.1** ตัวอย่างของ original database และcandidate 1-itemset ของ original database

การคำนวณจะนำค่า $min\_sup^{UP}$ ลบด้วยค่า support เพื่อหาว่าค่า k ที่ต้องการใน incremental database เป็นเท่าไร ถ้าผลลบที่ได้มีค่าน้อยกว่าหรือเท่ากับ 0 (กรณีเป็น Frequent itemset หรือค่า support ที่มี มีค่ามากกว่า $min\_sup^{UP}$ ) จะคิดที่ k = 0 แต่ถ้าผลลบที่ได้ มากกว่า 0 จะนำมาเข้าสูตรการคำนวณหาค่าความน่าจะเป็นดังนี้

$$P(x \geq k) = 1 - P(x \leq k)$$

$$P(x < k) = \sum_{x=0}^{k-1} \binom{n}{x} p^x (1-p)^{n-x}$$

ตัวอย่างการคำนวณค่าความน่าจะเป็นของแต่ละ item แสดงในรูปที่ 3.2

$$P(x \geq 6)_A = 1 - \sum_{x=0}^{5} \binom{15}{x} \frac{7}{10}^x \frac{3}{10}^{15-x} = 1$$

$$P(x \geq 6)_B = 1 - \sum_{x=0}^{5} \binom{15}{x} \frac{7}{10}^x \frac{3}{10}^{15-x} = 1$$

$$P(x \geq 6)_C = 1 - \sum_{x=0}^{5} \binom{15}{x} \frac{6}{10}^x \frac{4}{10}^{15-x} = 1$$

$$P(x \geq 6)_D = 1 - \sum_{x=0}^{5} \binom{15}{x} \frac{2}{10}^x \frac{8}{10}^{15-x} = 0.06$$

$$P(x \geq 6)_E = 1 - \sum_{x=0}^{5} \binom{15}{x} \frac{3}{10}^x \frac{7}{10}^{15-x} = 0.28$$

**รูปที่ 3.2** ตัวอย่างการคำนวณหาค่าความน่าจะเป็นของ candidate 1-itemset

สำหรับงานวิจัยนี้เราจะกำหนดค่า user-defined อีก 1 ตัวคือ ค่า prob ที่สามารถยอมรับได้ เช่นกำหนดให้ $prob_{pi}$=0.03 หมายความว่า เรายอมรับค่าความน่าจะเป็นที่จะเกิดขึ้น ว่าใน 100 ครั้ง จะเกิด itemset นี้ 3 ครั้ง เก็บไว้เป็น expected frequent itemset

จากตัวอย่างคือ $L_1$ = { A,B,C} และ $PL_1$ = {E} เมื่อได้ค่า L และ PL 1-itemset ใน original database มาแล้วจะนำ L และ PL 1-itemset มา join กันจะได้ candidate 2-itemset เพื่อนำมาหาค่า frequent 2-itemset ($L_2$)และ expected frequent itemset ($PL_2$) ดังรูปที่ 3.3 และทำการ join L และ PL 2-itemset สำหรับ candidate 3-itemset ($C_3$) เพื่อนำมาหาค่า frequent 3-itemset ($L_3$)และ expected frequent itemset ($PL_3$) ดังรูปที่ 3.4

ได้ผลลัพธ์ดังนี้

| C2(DB) | support |
|---|---|
| AB | 4 |
| AC | 4 |
| AE | 3 |
| BC | 3 |
| BE | 3 |
| CE | 1 |

| L2 | support |
|---|---|
| AB | 4 |
| AC | 4 |
| PL2 | support |
| AE | 3 |
| BC | 3 |
| BE | 3 |

**รูปที่ 3.3** แสดงcandidate 2-itemset, frequent 2-itemset และ expected frequent 2-itemset

| C3(DB) | support |
|---|---|
| ABC | 1 |
| ABE | 3 |

| PL3 | support |
|---|---|
| ABE | 3 |

**รูปที่ 3.4** แสดงcandidate 3-itemset, frequent 3-itemset และ expected frequent 3-itemset

**2.** การupdate frequent itemset และ expected frequent itemset ใน Updated database

เมื่อมี incremental database เพิ่มเข้ามาใน original database ซึ่งมีผลต่อ frequent itemset และ expected frequent itemset และยังอาจปรากฏ itemset ใหม่ได้อีกด้วย ดังนั้นขั้นตอนการ update นี้จึงเป็นขั้นตอนที่สำคัญ

ในงานวิจัยส่วนใหญ่จะต้องมีการสแกนทั้ง original database และ incremental database ดังเช่น อัลกอริทึม FUP ที่จะมีการ update ค่า support count ใหม่ที่ปรากฏใน incremental database ให้กับ frequent itemset ในขณะเดียวกันเมื่อพบว่ามี frequent itemset ใหม่เกิดขึ้นใน incremental database จะต้องสแกน original database เพื่อหาค่า support count ให้กับ frequent itemetใหม่ที่พบ เนื่องจากส่วนใหญ่ incremental database จะมีขนาดเล็ก ดังนั้น อาจเกิดพบ itemset ที่มีค่า support count มากกว่าหรือเท่ากับค่า minimum support ของ incremental database จำนวนมาก ดังนั้น การสแกน original database จะต้องสแกน เพื่อหาค่า support count ให้กับ Frequent itemset ใหม่จำนวนมาก โดยที่ Frequent itemset ใหม่นี้อาจไม่สามารถจะกลายเป็น Frequent itemset ใน updated database ได้เลย

ในบางอัลกอริทึม เช่น negative border เมื่อพบว่ามีการเปลี่ยนแปลงจาก negative border itemset มาเป็น frequent itemset ต้องทำการสแกนฐานข้อมูลทั้ง 2 ส่วน ด้วยการไมน์นิ่งใหม่ทั้งหมดซึ่งทำให้ใช้เวลาในค้นหา frequent itemset และสร้างกฎความสัมพันธ์ใหม่ที่เกิดขึ้น

สำหรับ frequent k-itemset ที่พบใหม่นี้หมายถึง itemset ที่ปรากฏใน original database แต่มีค่า support count น้อยกว่าค่า minimum support ของ original database และมีค่า support count น้อยกว่า minimum expected frequent หรือเรียกว่าเป็น small itemset ดังนั้นจึงไม่ถูกจัดเก็บในส่วนของ frequent k-itemset และ expected frequent k-itemset

ในงานวิจัยนี้ได้นำเสนอวิธีการในการลดจำนวน frequent k-itemset ใหม่ที่พบใน incremental database ที่จะต้องนำไปสแกนใน original database เพื่อหาค่า support count ที่แท้จริง แนวคิดนี้จะทำการประมาณค่าให้กับ frequent k-itemset ใหม่ที่พบด้วยค่า minimum expected frequent ที่ได้กำหนดใน original database และนำมาใช้ในการประมาณค่าของ itemset ที่คาดว่าจะกลายเป็น frequent itemset ใน updated database

เนื่องจากค่าของ frequent k-itemset ที่พบใหม่เป็น small k-itemset ดังนั้นโอกาสที่ small itemset นี้จะเกิดใน original database นั้นสามารถประมาณได้ด้วยค่าสูงสุดคือ ค่าของ minimum expected frequent ลบ 1 เมื่อทราบค่าประมาณสูงสุดของ support count ของ small itemset สามารถนำมาใช้ในการกรอง itemset ที่มีโอกาสจะกลายเป็น frequent k-itemet ใหม่ใน updated database ได้โดยการหาผลรวมของ support count ของ frequent k-itemset ใหม่ ที่พบใน incremental database กับค่าประมาณการของ minimum expected frequent -1 ถ้าผลรวมที่ได้มีค่ามากกว่าหรือเท่ากับ minimum support ของ updated database จึงจะเก็บ frequent k-itemset ใหม่นั้นเพื่อไปสแกนใน original database ทั้งนี้จะทำการหา frequent itemset สำหรับทุก k-itemset แล้วจึงนำไปสแกนใน original database ในขั้นตอนสุดท้าย

ขั้นตอนการ update frequent itemset และ expected frequent Itemset หลักๆ ดังแสดงในอัลกอริทึมรูปที่ 5 ดังนี้คือ

เมื่อมี incremental database เพิ่มเข้ามาจะเริ่มจากการปรับปรุง support count ของ frequent itemset และ expected frequent itemset สำหรับ 1-itemset ดังแสดงในบรรทัดที่ 3 รูปที่ 3.5 และส่วนของการ updated 1-itemset เริ่มจากการสแกน incremental database เพื่อหา support count สำหรับ candidate 1-itemest ของ original database และ candidate 1-itemset ของ incremental database จากการทำงานในส่วนนี้จะทำให้ทราบค่าของ Frequent 1-itemset และ Expected Frequent itemset ใน updated database ดังแสดงในบรรทัดที่ 1- 6 ของรูปที่ 3.6

เมื่อค่า itemset ของ 1-itemset ถูกปรับปรุงแล้วจะทำการปรับปรุง Frequent itemset และ Expected Frequent itemset ของitemset ที่มากกว่า 1 itemset ถ้าพบว่ามี Frequent 1-itemset ของ updated database เกิดขึ้นใหม่จะทำการสร้าง Candidate 2-itemset โดยจะทำการ join Frequent k-itemset ของ updated database เข้าด้วยกันดังแสดงในรูปที่ 3.7 บรรทัดที่ 3 แต่ถ้า itemset มากกว่า 2 itemset ขึ้นไปจะทำการ join Frequent k-itemset (k>2) ด้วย Frequent k-itemset ใหม่ที่ปรากฏขึ้นใน incremental database ทั้ง k >=2 และ k>2 itemset จะถูกนำมาพิจารณาว่ามี frequent k-1 itemset ของ updated database เป็นสมาชิก ถ้าไม่ใช่ itemset นั้นจะถูก prune ออกไป

เมื่อได้ candidate k-itemset (k>=2) แล้วจะสแกน incremental database เพื่อหาค่า support count ของ Candidate itemset และ เพื่อ update support count สำหรับ itemset ที่เป็นสมาชิกของ Frequent itemset และ Expected Frequent itemset ดังรูปที่ 3.8

สำหรับ Candidate itemset เราจะทราบค่า support count ที่ itemset นั้นปรากฏใน incremental database ซึ่งถ้า Candidate itemset นี้มีค่า support ที่มากกว่าหรือเท่ากับค่า minimum support ของ incremental database แต่เนื่องจาก Candidate itemset นี้ไม่พบว่าเป็นสมาชิกของ Frequent itemset หรือ Expected Frequent itemset ในงานวิจัยนี้จะทำการตรวจสอบว่า candidate itemset นี้มีโอกาสที่จะกลายมาเป็น Frequent itemset ใน updated database หรือไม่โดยจะยังไม่ทำ

การสแกนเพื่อหาค่า support count ที่แท้จริงใน original database แต่จะใช้ค่า expected value ที่ได้จากการไมน์นิ่ง original database ในการประมาณค่า support count สูงสุดที่จะเกิดขึ้นในที่นี้คือ expected value -1 โดยถ้าผลรวมของ support count และ expected value-1 ของ candidate itemset มีค่ามากกว่าหรือเท่ากับค่า minimum support ของ updated database จะเก็บ candidate itemset นั้นๆ เพื่อนำไปหาค่า support ที่แท้จริงด้วยการสแกน original database ในขั้นตอนสุดท้ายดังรูปที่ 3.9

จากการทำงานของอัลกอริทึมจะพบว่า Frequent k-itemset ใหม่ (k>2) นั้น จะเกิดขึ้นได้ก็ต่อเมื่อ Frequent k-itemset นั้นอย่างน้อยจะต้องเป็น Frequent k-itemset ใน incremental database จึงจะมีโอกาสที่จะกลายเป็น Frequent k-itemset ของ updated database ได้

**Algorithm1 : Main Algorithm**

*Input :* $DB, db, k, \sigma^{UP}, \rho^{UP}, \rho^{DB}, C_1^{DB}, F_1^{DB}, EF_1^{DB}$ *and their count*

*Output :* $F_k^{UP}, EF_k^{UP}$

1. $k = 1$
2. *if* $k = 1$
3. *Update* $1-$*itemset*
4. $k = k + 1$
5. *else*
6. *for* $\left(k = 2; F_k^{UP} \neq \phi; k++\right)$ *do*
7. *Generate Candidate Itemset*
8. *Update* $k-$*itemset (return m, Temp _ scanDB )*
9. *// m is the maximum itemset of Temp _ scanDB*
10. $k = k + 1$
11. *end do*
12. *end if*
13. $k = 2$
14. *while* $(Temp\_scanDBl_k \neq \phi$ *and* $(k \leq m)$ *do*
15. *Scan Original Database*$(Temp\_scanDB_k)$
16. $k = k + 1$
17. *end do*
18. *clear Temp _ scanDB*

**รูปที่ 3.5** Main Algorithm

**Algorithm 2 : Updating 1-itemsets**

*Input :* $DB, db, \sigma^{UP}, \rho^{UP}, C_1^{DB}, F_1^{DB}, EF_1^{DB}, C_1^{db}$ *and their count*

*Output :* $F_1^{UP}, EF_1^{UP}, C_1^{UP}$ *and their count*

1. *Scan db and find count* $c(X, db)$ *for all* $X \in C_1^{DB} \cup C_1^{db}$
2. *for all* $X \in C_1^{DB} \cup C_1^{db}$ *do*
3. $c(X,UP) = c(X,DB) + c(X,db)$
4. *end do*
5. $F_1^{UP} = \left\{X \in C_1^{UP} \middle| c(X,UP) \geq \sigma^{UP}\right\}$
6. $EF_1^{UP} = \left\{X \in C_1^{UP} \middle| \rho^{UP} \leq c(X,UP) < \sigma^{UP}\right\}$

**รูปที่ 3.6** Update 1-itemsest Algorithm

**Algorithm 3: Generating Candidate k- itemsets**

*Input :* $F_1^{UP}, F_{k-1}^{UP}, F_{k-1}^{db}, k$

*Output :* $C_k^{new}$

1. *if* $k = 2$ *then*
2. *if* $\left(length\left(F_1^{UP}\right) \geq 2\right)$ *then*
3. $C_2^{new} = F_1^{UP} * F_1^{UP}$
4. *end if*
5. *else if* $k > 2$ *then*
6. $C_k^{db} = F_{k-1}^{db} * F_{k-1}^{db}$
7. *fpr all* $X \in C_k^{db}$ *do*
8. $C_k^{new} = \left\{ X \in C_k^{db} \middle| X \in F_{k-1}^{UP} \right\}$
9. *end do*
10. *end if*

**รูปที่ 3.7** Generating Candidate k- itemsets Algorithm

**Algorithm 4 : Update (k ≥ 2) itemset**

*Input:* $DB, db, \sigma^{UP}, \rho^{UP}, \rho^{DB}, F_k^{DB}, EF_k^{DB}$ *and their count*

*Ouiput:* $F_k^{UP}$ *and* $EF_k^{UP}, F_k^{db}, Temp\_scanDB$ *and their count, m*

1. *Scan db and find count* $c(X, db)$ *and* $c(Y, db)$
2. *If* $X \in \left(F_k^{DB} \cup EF_k^{DB}\right)$ *and* $Y \in C_k^{new}$
3. *for all* $X \in \left(F_k^{DB} \cup EF_k^{DB} \cup C_k^{new}\right)$ *do*
4. *if* $X \in \left(F_k^{DB} \cup EF_k^{DB}\right)$ *and* $X \in C_k^{new}$ *then*
5. $c(X, UP) = c(X, DB) + c(X, db)$
6. *elseif* $X \in \left(F_k^{DB} \cup EF_k^{DB}\right)$ *and* $X \notin C_k^{new}$ *then*
7. $c(X, UP) = c(X, DB)$
8. *elseif* $X \notin \left(F_k^{DB} \cup EF_k^{DB}\right)$ *and* $X \in C_k^{new}$ *then*
9. $Temp\_scanDB_k = \{ X \mid \left(c(X, db) + \left(\rho^{DB} - 1\right)\right) \geq \sigma^{UP} \}$
10. *end if*
11. *end do*
12. $F_k^{UP} = \left\{ X \middle| c(X, UP) \geq \sigma^{UP} \right\}$
13. $EF_k^{UP} = \left\{ X \middle| \rho^{UP} \leq c(X, UP) < \sigma^{UP} \right\}$

**รูปที่ 3.8** Update (k≥ 2) itemset Algorithm

**Algorithm 5 : Scanning an original database**

*Input : Temp _ scanDB$_k$, $\sigma^{UP}$, $\rho^{UP}$, $F_k^{UP}$, $EF_k^{UP}$ and their count*

*Output : $F_k^{UP}$, $EF_k^{UP}$ and their count*

1. *Scan DB and obtain count $c(X, DB)$ for all Temp _ scanDB$_k$*
2. *for all $X \in$ Temp _ scanDB$_k$ do*
3. $c(X, UP) = c(X, DB) + c(X, db)$
4. *end do*
5. $F_k^{new} = \{X \mid X \in Temp\_scanDB_k \text{ and } c(X, UP) \geq \sigma^{UP}\}$
6. $EF_k^{new} = \{X \mid X \in Temp\_scanDB_k \text{ and } \rho^{UP} \leq c(X, UP) < \sigma^{UP}\}$
7. $F_k^{UP} = F_k^{UP} \cup F_k^{new}$
8. $EF_k^{UP} = EF_k^{UP} \cup EF_k^{new}$

**รูปที่ 3.9** Algorithm for Scanning an original database

จากขั้นตอนการค้นหา Frequent k-itemset ของงานวิจัยนี้เป็นการค้นหากฎความสัมพันธ์ที่มีประสิทธิภาพในกรณีที่มีการเพิ่ม incremental database เข้าไปใน original database โดยสามารถที่จะปรับปรุงค่า support count สำหรับ Frequent k-itemset และ Expected Frequent itemset และจะ prune itemset ที่ไม่อาจกลายเป็น Frequent k-itemset ออกไปทำให้ itemset ต่างๆ ได้มีการปรับปรุงให้มีความทันสมัยอยู่เสมอ และเมื่อมี Frequent k-itemset เกิดขึ้นสามารถที่จะทำการค้นหาได้อย่างถูกต้องและครบถ้วน และสามารถที่จะลดจำนวน itemset ที่จะถูกสแกนใน original database ได้ ทำให้ลดเวลาในการสแกน original database ที่มีขนาดของข้อมูลจำนวนมาก ดังแสดงการทดลองและผลการทดลองในบทที่ 4

# บทที่ 4

# การทดลองและผลการทดลอง

## 4.1 การวัดประสิทธิภาพการทำงานของโมเดล

การวัดประสิทธิภาพการทำงานของโมเดล ได้ใช้ตัววัดประสิทธิภาพที่ใช้ในงานวิจัยทั่วไป คือการวัดความเร็วในการค้นหากฎความสัมพันธ์ด้วย min_sup ที่แตกต่างกัน โดยจะทำการวัดประสิทธิภาพการค้นหากฎความสัมพันธ์ระหว่างข้อมูลเทียบระหว่างอัลกอริทึมที่เคยมีและอัลกอริทึม probability-based incremental association rule discovery algorithm

**การสร้างข้อมูลเพื่อใช้ในการทดลอง ด้วย Synthetic data generation**

การทดลองจะใช้ Synthetic data ซึ่งเป็นเทคนิคเดียวกับ Agrawal and Srikant 1994 ซึ่งใช้หลักทางสถิติมาประยุกต์ใช้เพื่อสร้างข้อมูลเลียนแบบข้อมูลจริงที่เกิดจากการเลือกซื้อสินค้าในลักษณะที่เรียกว่า market-basket

ชุดข้อมูลที่ใช้ในการทดลองจะได้จาก synthetic data ที่ใช้คือ T10.I4.D10K ( ขนาดเฉลี่ยของ transaction = 10, ขนาดเฉลี่ยของ maximal potentially large item sets = 4 และจำนวนของ transaction = 10,000 ) ประกอบด้วย 2 ส่วน ส่วนแรกเป็นข้อมูลที่ได้จากการสุ่มจำนวน 100,000 transaction เพื่อเป็นข้อมูลที่ในการค้นหากฎความสัมพันธ์ของฐานข้อมูลเดิม (original database) และส่วนที่ 2 เป็นส่วนของ incremental database จำนวน 100,000 transaction โดยจะทำการเพิ่มจำนวน incremental database ไปทีละ 10 % ของข้อมูลเดิมจนกว่าจะครบ 100% ขนาดของข้อมูลที่ใช้เพิ่มเข้าไปในฐานข้อมูลเดิมนี้จะนำมาใช้ในการพิจารณาความถูกต้องของกฎความสัมพันธ์ใหม่ที่ได้จากการปรับปรุงฐานข้อมูล และวัดประสิทธิภาพของอัลกอริทึมเมื่อนำไปใช้ในการค้นหากฎความสัมพันธ์เมื่อมีการเพิ่มข้อมูลใหม่เข้าไป

## วัตถุประสงค์ในการทดลอง

1. เพื่อทดสอบการไมน์นิ่ง Frequent itemset ที่ถูกต้องหลังจากเพิ่ม incremental database เข้าไปปรับปรุง original database โดยมีการนำความรู้ที่ได้จากการไมน์นิ่งใน original database มาใช้ให้เกิดประโยชน์เพื่อลดจำนวนครั้งและจำนวน itemset ในการสแกน original database
2. เพื่อวัดประสิทธิภาพในการเพิ่มขยายการค้นหากฎความสัมพันธ์ของอัลกอริทึม(ใหม่) ในกรณีของการเพิ่ม transaction เข้าไปในฐานข้อมูล

## วิธีการทดลอง

**1. ทดสอบความถูกต้องของอัลกอริทึม**

ในการทดสอบความถูกต้องของข้อมูลที่ได้จากการเพิ่ม transaction ใหม่เข้าไปในฐานข้อมูล โดยเปรียบเทียบกับกฎความสัมพันธ์ที่ได้จากการ rerun Apriori เมื่อมีการเพิ่ม transaction ใหม่เข้าไปในฐานข้อมูล

**2. ทดสอบประสิทธิภาพของอัลกอริทึม**

การทดสอบประสิทธิภาพของอัลกอริทึม ซึ่งทำการทดสอบเปรียบเทียบผลกับ 3 อัลกอริทึม FUP, Border,Probability-based โดยใช้ตัวชี้วัด 3 ตัว คือ เวลา, ขนาดของข้อมูลที่เพิ่ม และจำนวน candidate itemset

**minimum support threshold**

การค้นหากฎความสัมพันธ์ โดยกำหนดค่า Minimum support ในระดับต่างๆกัน ระหว่างค่า minimum support threshold 1.0 – 3.0%

**Effect of size of updates**

วัดความเร็วเมื่อมีการเพิ่มขนาดของ Incremental database เข้าไปในฐานข้อมูลจาก 10 - 100 % ด้วยค่า minimum support threshold ที่คงที่ โดยพิจารณาจาก execution time ที่ใช้ในการ run เพื่อหากฎความสัมพันธ์ที่ได้จากการปรับปรุง

**Varying the number of added transactions independently**

การทดลองนี้จะใช้สำหรับหาขนาดของ Transaction ที่เพิ่มเข้าไปใน original database ที่มีผลกับ performance ของอัลกอริทึม โดยใช้ T10.I4.D10 +10 ในการทดลองจะกำหนด initial database 10,000 transactions และทำการเพิ่มทีละ 1,000 transactions เข้าไปในฐานข้อมูล

## 4.2 ชุดข้อมูลที่ใช้ในการทดลองและผลการทดลอง

การประเมินประสิทธิภาพของ Probability-based incremental association rule discovery algorithm ได้ทำการทดลองด้วยเครื่องคอมพิวเตอร์ Pentium 4 หน่วยความจำหลัก 1 GB ชุดข้อมูลที่ใช้ในการทดลองคือ synthetic dataset จำนวน 110,000 transactions ที่ประกอบด้วย unique items 100 items ด้วยค่าเฉลี่ยของ items ที่ปรากฏในแต่ละ transaction คือ 10 items และ maximal size itemset คือ 4 itemsets นำสุ่มเลือกเพื่อสร้างเป็น original database 1 ชุด ขนาด 10000 transactions และ incremental database จำนวน 1000 ชุด ขนาด 1000 transactions

การทดลองเริ่มจากขั้นตอนการค้นหากฎความสัมพันธ์จาก original database จำนวน 10000 transactions ดังตารางที่ 4.1 แสดงจำนวน itemset ของ original database ที่จัดเก็บในแต่ละอัลกอริทึม

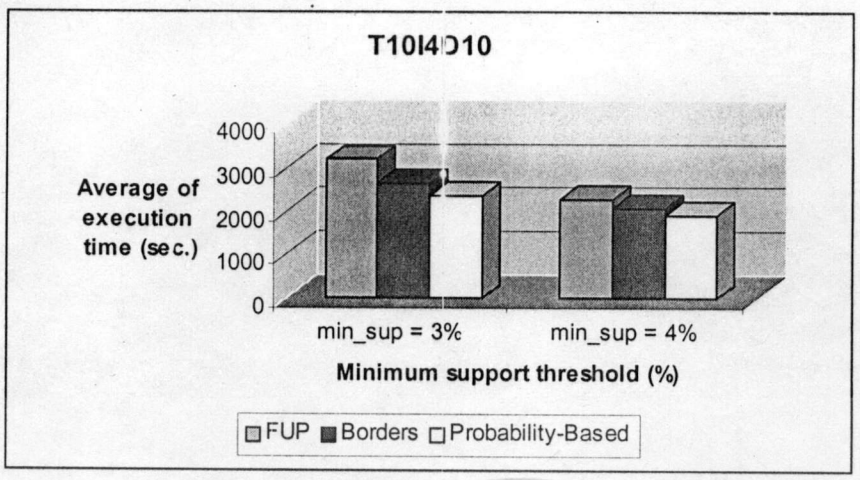
จากนั้นทำการเพิ่ม incremental database ในขนาด 10% ของ original database ในที่นี้คือ 1000 transactions เข้าไปใน original database และทำการทดลองจำนวน 100 ครั้ง โดยเปรียบเทียบการทดลองกับ 2 อัลกอริทึมคือ FUP และ Border ด้วยค่า minimum support 3% และ 4% ดังแสดงในตารางที่ 4.2 และรูปที่ 4.1

**ตารางที่ 4.1** จำนวน itemset ของ original database สำหรับแต่ละอัลกอริทึม

| Min_sup (%) | Algorithm | Number of Frequent k-itemset | | | Number of Infrequent k-itemset | | |
|---|---|---|---|---|---|---|---|
| | | k=1 | k=2 | k=3 | k=1 | k=2 | k=3 |
| 3% | FUP | 97 | 263 | 9 | 0 | 0 | 0 |
| | Borders | 97 | 263 | 9 | 3 | 4393 | 371 |
| | Probability-Based | 97 | 263 | 9 | 0 | 25 | 5 |
| 4% | FUP | 93 | 74 | 1 | 0 | 0 | 0 |
| | Borders | 93 | 74 | 1 | 7 | 4201 | 26 |
| | Probability-Based | 93 | 74 | 1 | 1 | 8 | 0 |

**ตารางที่ 4.2** ค่าเฉลี่ยของเวลาที่ใช้ในการประมวลผล

| | Average of Execution time for 100 trials | | |
|---|---|---|---|
| min_sup (%) | FUP | Borders | Probability- Based |
| 3% | 3195.6939 | 2660.9297 | 2354.24533 |
| 4% | 2274.4477 | 2087.8636 | 1931.19147 |




**รูปที่ 4.1** กราฟเปรียบเทียบเวลาที่ใช้ในการประมวลผลของอัลกอริทึม FUP, Borders และ Probability-based ด้วยค่า minimum support 3% และ 4%

จากผลการทดลองพบว่า Probability-base incremental association rule discovery algorithm มีผลการทำงานที่ดีกว่าอัลกอริทึม FUP และ Border

# เอกสารอ้างอิง


[1] R. Agrawal., T. Imielinski, and A. Swami, "Mining association rules between sets of items in large databases", In Proceeding of the ACM SIGMOD Int'l Conf. on Management of Data (A CM SIGMOD '93), Washington, USA,May 1993, pp. 207-216.

[2] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules", In Proceedings of 20 th Intl Conf. on Very Large Databases (VLDB'94), pages487-499, Santiago, Chile, September 1994, pp. 478 -499.

[3] D. Cheung, J. Han, V. Ng, and C. Y. Wong, "Maintenance of discovered association rules in large databases: An incremental updating technique", In 12th IEEE International Conference on Data Engineering, February 1996, pp. 106-114.

[4] D. W. Cheung, S.D. Lee, B. Kao, "A General incremental technique for maintaining discovered association rules" , In Proceedings of the 5 th Intl. Conf. on Database Systems for Advanced Applications (DASFAA'97), Melbourne, Australia, April 1997, pp. 185-194.

[5] H. Toivonen. "Sampling Large Databases for Association Rules", Proceeding of the 22th International conference on Very Large Data Bases, September 1996,pp. 134-145.

[6] S. Thomas, S. Bodagala, K. Alsabti, and S. Ranka, "An efficient algorithm for the incremental updation of association rules in large databases" , In Proceedings of the 3rd Intl. Conf. on Knowledge Discovery and Data Mining (KDD'97), New Port Beach, California, August 1997, pp. 263-266.

[7] N.F. Ayan, A.U. Tansel, and E. Arun, "An efficient algorithm to update large itemsets with early pruning", Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, August 1999, pp. 287-291.

[8] C. H. Lee, C. R. Lin, and M. S. Chen, "Sliding-Window Filtering: An Efficient Algorithm for Incremental Mining" , Proceeding of the ACM 10th International Conference on Information and Knowledge Management , November 2001.

[9] C.C. Chang, Y.C. Li and J.S. Lee, "An efficient algorithm for incremental mining of association rules", Proceedings of the 15th international workshop on research issues in data engineering: stream data mining and applications (RIDE-SDMA'05), IEEE, 2005.

[10] A. A. Veloso, W. Meira Jr, M.B. de Carvalho,B. Pôssas, S. Parthasarathy and M. Javeed Zaki, "Mining frequent itemsets in evolving databases" , In Proc. 2nd SIAM Intl. Conf. on Data Mining, Arlington, VA, April 2002.

[11] N. L. Sarda and N. V. Srinivas, "An adaptive algorithm for incremental. mining of association rules" , In Proc. 9th Intl.Workshop on Database and Expert System Applications,Vienna, Austria, Aug 1998, pp. 240-245.

[12] R. Feldman, Y. Aumann, and O. Lipshtat. "Borders: An efficient algorithm for association generation in dynamic databases",*Journal,*Intelligent Information System, 1990, pp. 61-73.

[13] M. Adnan, R. Alhajj and K. Barker, "Performance Analysis of Incremental Update of Association Rules Mining Approaches", In Proceeding of 9th IEEE International Conference on Intellgent Engineering System 2005, Sept. 16-19, 2005, pp.129-134.

[14] T.P. Hong C.Y. Wang and Y.H. Tao, "A new incremental data mining algorithm using pre-large itemsets", *Journal*, Intelligent Data Analysis, Vol. 5, No.2, pp. 111-129, 2001.

[15] R. Amornchewin and W. Kreesuradej, "Incremental association rule mining using promising frequent itemset algorithm", In Proceeding 6th International Conference on Information, Communications and Signal Processing, Dec. 10-13 2007, pp.1-5.

[16] C. Zhang, S. Zhang and G. I. Webb, "Identifying Approximate Itemsets of Interest in Large Databases", Applied Intelligence 18, 91–104, 2003.

# ภาคผนวก

1. R. Amornchewin and W. Kreesuradej, "Incremental association rule mining using promising frequent itemset algorithm", in Proceeding 6th International Conference on Information, Communications and Signal Processing, Dec. 10-13 2007, pp.1-5.
2. R. Amornchewin and W. Kreesuradej, "Probability-based Incremental association rule discovery algorithm", (will be published) in Proceeding of The 2008 International Symposium on Computer Science and its Applications, Oct. 13-15 2008.

# Probability-based incremental association rule discovery algorithm

Ratchadaporn Amornchewin
*Faculty of Information Technology*
*King Mongkut's Institute of Technology*
*Ladkrabang*
*Bangkok, 10520 Thailand*
ramornchewin@yahoo.com

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

## Abstract

*In dynamic databases, new transactions are appended as time advances. This may introduce new association rules and some existing association rules would become invalid. Thus, the maintenance of association rules for dynamic databases is an important problem. In this paper, probability-based incremental association rule discovery algorithm is proposed to deal with this problem. The proposed algorithm uses the principle of Bernoulli trials to find expected frequent itemsets. This can reduce a number of times to scan an original database. This paper also proposes a new updating and pruning algorithm that guarantee to find all frequent itemsets of an updated database efficiently. The simulation results show that the proposed algorithm has a good performance.*

## 1. Introduction

Data mining is one of the processes of Knowledge Discovery in Database (KDD) that is used for extracting information or pattern from large database. One major area of data mining is association rule mining [1] that discovers hidden knowledge in database. The association rule mining problem is to find out all the rules in the form of $X \Rightarrow Y$, where X and $Y \subset I$ are sets of items, called itemsets. The association rule discovery algorithm is usually decomposed into 2 major steps. The first step is find out all large itemsets that have support value exceed a minimum support threshold and the second steps is find out all the association rules that have value exceed a minimum confidence threshold.

However, a database is dynamic when new transactions are inserted into the database. This may introduce new association rules and some existing association rules would become invalid. As a brute force approach, apriori algorithm may be applied to mining a whole dynamic database when the database has been changed. However, this approach is very costly even if small amount of new transactions is inserted into a database. Thus, the association rule mining for a dynamic database is an important problem. Several research works [6, 7, 8, 9] have proposed several incremental algorithms to deal with this problem. Review of related works will be introduced in section 2.

In this paper, a new incremental algorithm, called probability-based incremental association rule discovery, is introduced. The goal of this work is to solve the updating problem of association rules after a number of new records have been added to a database. Based on probabilistic approach, our incremental algorithm predicts infrequent itemsets that have capable of being frequent itemsets after a number of new records have been added to a database. That infrequent itemsets is called expected frequent itemsets. Our algorithm can reduce a number of times to scan an original database. As a result, the algorithm has execution time faster than that of previous methods.

## 2. Previous work

An influential algorithm for association rule mining is Apriori [2]. Apriori computes frequent itemsets in a large database through several iterations based on a prior knowledge. Each iteration has 2 steps which are a joining step and a pruning step. For a frequent itemset, its support must be higher than a user-specified minimum support threshold. The association rule can be discoverd based on frequent itemsets.

For dynamic databases, several incremental updating techniques have been developed for mining association rules. One of the previous works for incremental association rule mining is FUP algorithm that was presented by Cheung et al [3]. The major idea of FUP is re-using frequent itemsets of previous

mining to update with frequent itemsets of an incremental database based on the concepts of Apriori.

Furthermore, negative border approach is presented by Toivonen [5], Thomas et al [6] and Feldman et al [8]. The negative border approach is an incremental mining algorithm based on FUP. The border itemset is not a frequent itemset but all its proper subsets are frequent itemsets. The approach need to keep a large number of border itemsets in order to reduce scanning times of an original database.

To reduce memory space, Hong et al [9] and Amornchewin et al[10] propose a new approach. The approach maintains both frequent itemsets and expected frequent itemsets. An expected frequent itemset is not a frequent itemset but is expected to become a frequent itemset when a new database is added to an original database. In order to guarantee that all frequent itemsets can be found when a new database is added to an original database, the approach can only allow very small size of an incremental database to insert into an original database.

Similarly, the proposed method in this paper also keeps not only frequent itemsets but also expected frequent itemsets from an original database. Unlike the previous works, this paper proposes a new technique for predicting expected frequent itemsets. Here, the principle of Bernoulli trials is used to predict the expected frequent itemsets. The expected frequent itemsets obtained from the proposed technique has lesser members than the border itemsets and the expected frequent itemsets obtained from the previous technique. This work also proposes a new updating algorithm that guarantee to find all frequent itemsets of a dynamic database efficiently.

## 3. Probability-based Incremental Association Rule Discovery Algorithm

When a dynamic database is inserted new transactions, not only some existing association rules may be invalidated but also some new association rules may be discovered. This is the case because frequent itemsets can be changed after inserting new transactions into a dynamic database. Therefore, an association rule discovery algorithm for a dynamic database has to maintain frequent itemsets when new transactions are inserted into the dynamic database.

The task of an association rule discovery algorithm for a dynamic database can be divided into three tasks. The first task is to update support count of existing frequent itemsets. The second task is to prune existing frequent itemsets that have support count below a minimum support threshold after updating the database. The third task is to discover new frequent itemsets that have support count equal or above a minimum support threshold after updating the database.

In this section, we describe our algorithm into 2 subsections. Firstly, probability-based expected frequent itemsets is presented. Secondly, updating frequent and expected frequent itemsets is introduced.

### 3.1 Probability-Based Expected Frequent Itemsets

For our algorithm, an original database, which is a database before being inserted new transactions, is firstly mined to find all frequent itemsets that satisfy a minimum support count, denoted $k_{original}$. Furthermore, the proposed algorithm also predicts and keeps expected frequent itemsets that may become frequent itemsets if new transactions are inserted into the original database.

Our assumption for the new algorithm is that the statistics of new transactions slightly change from original transactions and the maximum number of new transactions that be allowed to insert into an original database is available. According to the first assumption, the statistics of old transactions, obtained from previous mining, can be utilized for approximating that of new transactions. Therefore, the new algorithm uses support count of itemsets obtained from previous mining to approximate the probability of infrequent itemsets in an original database that may be capable of being frequent itemsets when new transactions are inserted into the original database.

Here, the process of inserting m transactions into an original database of n transaction can be considered as (m+n) Bernoulli trials, which are (m+n) sequence of identical trials. Each itemset has its probability of appearing in a transaction, denoted by p, i.e., the probability of success. According to the principle of Bernoulli trials, the probability of the number of an itemset to appearing in (n+m) transactions, denoted by P(x), can be found by the following equation:

$$P(x)=\binom{n+m}{x}\cdot p^{x}\cdot(1-p)^{n+m-x} \quad ,$$

where p is the probability of an itemset appearing in a transaction, m is a number of new transactions, and n is a number of transactions of an original database.

Thus, if k is a minimum support count after inserting new transactions into an original database, the probability of an itemset to be a frequent itemset in an updated database can be obtained as the following equations:

$$P\left(x \geq k\right)_{item} = 1 - P\left(x < k\right)_{item} \qquad (1)$$

Here, an expected frequent itemset is an itemset that is not a frequent itemset but has its probability to be a frequent itemset greater than $Prob_{pl}$. $Prob_{pl}$ is a threshold constant specified by users. $Prob_{pl}$ indicates the minimum confidence level that a promising

frequent itemset will be a frequent itemset after inserting new transaction into an original database.

### 3.2. Updating frequent and expected frequent itemsets

When new transactions are added to an original database, an old frequent k-itemset could become an infrequent k-itemset and an old expected frequent k-itemset could become a frequent k-itemset. This introduces new association rules and some existing association rules would become invalid. To deal with this problem, all k-itemsets must be updated when new transactions are added to an original database. The notation used in this section is given in Table1.

**Table 1.** The notation for Updating frequent and expected frequent itemsets algorithm

| | |
|---|---|
| DB | Original database |
| db | Incremental Database |
| UP | Updated database |
| k | Number of itemset |
| σ | Minimum support |
| ρ | Minimum Expected Frequent |
| $C_k$ | Candidate k-itemset |
| $F_k$ | Frequent k-itemset |
| $EF_k$ | Expected Frequent k-itemset |

Here, a new updating frequent and expected frequent itemsets algorithm shown in Figure 1 is proposed in this paper. The algorithm consists of three phases. The first phase is updating 1- frequent and expected frequent itemsets, i.e. line1-3. The second phase is repeatedly updating the other frequent and expected frequent itemsets by using only an incremental database, i.e. line 6-11. The third phase is scanning an original database, i.e. line 13-22.

The First phase is updating 1- frequent and expected frequent itemsets. According to our propose, the 1-candidate itemsets of an updated database, i.e. $C_1^{UP}$, can be found by combining the 1-candidate itemsets of an original database, i.e. $C_1^{DB}$, with the 1-candidate itemsets of an incremental database, i.e. $C_1^{db}$. Then, the support count of $C_1^{UP}$ can be updated by scanning only an incremental database. Then, the result of this phase is consist of 1-frequent and expected 1-itemsets of an updated database.

The second phase has 2 major steps which are a generating *k*- incremental candidate itemsets step and an updating support count of k- incremental frequent and k- incremental expected frequent itemsets step for *k* greater than or equal to 2. For *k*=2, the 2- incremental candidate itemsets are easily obtained by joining $F_1^{UP}$ with $F_1^{UP}$. For *k*>2, the algorithm is firstly find *k*-candidate itemsets of an incremental database, i.e. $C_k^{db}$, by joining $F_{k-1}^{db}$ with $F_{k-1}^{db}$. Similar to Apriori algorithm, the *k*- candidate itemsets of an incremental database can be the updated frequent itemsets, i.e. $F_k^{UP}$, only if the subsets of the *k*- candidate itemsets of an incremental database must be in the (*k*-1 )- updated frequent. Thus, the *k*- incremental candidate itemsets, will keep only the k- candidate itemsets of an incremental database whose subsets of the *k*- candidate itemsets are in the (*k*-1) - updated frequent intemsets. This can prune the *k*- candidate itemsets of an incremental database that can't be the *k*- updated frequent intemsets.

**Algorithm1 : Updating frequent and expected frequent itemsets Algorithm**

*Input : DB, db, k,* $\sigma^{UP}, \rho^{UP}, \rho^{DB}, C_1^{DB}, F_1^{DB}, EF_1^{DB}$ *and their count*
*Output :* $F_k^{UP}, EF_k^{UP}$

1. $k = 1$
2. *if* $k = 1$
3. *Update 1−itemset*
4. $k = k + 1$
5. *else*
6. *for* $(k = 2; F_1^{UP} \neq \phi; k++)$ *do*
7. *Generate Candidate Itemset*
8. *Update k − itemset(return m, Temp_scanDB)*
9. *// m is the maximum itemset of Temp_scanDB*
10. $k = k + 1$
11. *end do*
12. *end if*
13. $k = 2$
14. *while* $(Temp\_scanDB_k \neq \phi$ *and* $(k \leq m)$ *do*
15. *Scan Original Database for all* $X \in Temp\_scanDB_k$
16. *for all* $X \in Temp\_scanDB_k$ *do*
17. $c(X,UP) = c(X,DB) + c(X,db)$
18. *end do*
19. $F_k^{UP} = F_k^{UP} \cup \{X \mid X \in Temp\_scanDB_k \text{ and } c(X,UP) \geq \sigma^{UP}\}$
20. $EF_k^{UP} = EF_k^{UP} \cup \{X \mid X \in Temp\_scanDB_k \text{ and } \rho^{UP} \leq c(X,UP) < \sigma^{UP}\}$
21. $k = k + 1$
22. *end do*
23. *clear Temp_scanDB*

**Figure 1.** Updating frequent and expected frequent itmesets Algorithm

When any *k*-itemsets are not in the union set of the original *k*-frequent and the original *k*- expected frequent itemsets, but is in the k- incremental candidate itemsets, their support counts need to be specially updated. This is the case because their support counts obtained from an original database are not available. Here, their support counts in an original database are assumed to be equal to the sum of $\rho^{DB}$ - 1 and their support counts from an incremental database. If any k-itemsets have support counts below updated min support count, i.e. $\sigma^{UP}$, the *k*-itemsets can't be the *k*-updated frequent itemsets. On the other hand, if any *k*-itemsets have support counts above or equal to an

updated min support count, the *k*-itemsets are likely to be the *k*-updated frequent itemsets. Thus, the *k*-itemsets, which have support counts above or equal to an updated min support count, are set aside for finding their true support counts from an original database.

At the third phase, an original database is scanned to find true support counts for the *k*-itemsets that are likely to be the *k*-updated frequent itemsets. The support counts of the likely *k*-updated frequent itemsets are found and updated by scanning an original database. Then, all *k*-updated frequent itemsets and *k*-updated expected frequent itemsets are found.

## 4. Experiments

To evaluate the performance of probability-based incremental association rules discovery algorithm, the algorithm is implemented and tested on a PC with a 2.8 GHz Pentium 4 processor, and 1 GB main memory. The experiments are conducted on a synthetic dataset, called T10I4D10K. The technique for generating the dataset is proposed by Agrawal and etc. [1]. The synthetic dataset comprises 110,000 transactions over 100 unique items, each transaction has 10 items on average and the maximal size itemset is 4.

Firstly, the proposed algorithm with $Prob_{pi}$ = 0.06 used to find association rules from an original database of 10,000 transactions. Then, the same sizes of incremental databases, i.e. 10% of the original database, are added to the original database for 100 trials. For comparison purpose, FUP and Borders algorithm are also used to find association rules from the same original database and the same incremental databases. Figure 2 and Table 2 show the average of execution time for FUP, Borders and our approach. The results also show that the proposed algorithm has much better running time than that of FUP.

**Table 2.** Average of Execution time

| | Average of Execution time for 100 trials | | |
|---|---|---|---|
| Min_sup (%) | FUP | Borders | Probability-Based |
| 3% | 3195.6939 | 2660.9297 | 2354.24533 |
| 4% | 2274.4477 | 2087.8636 | 1931.19147 |

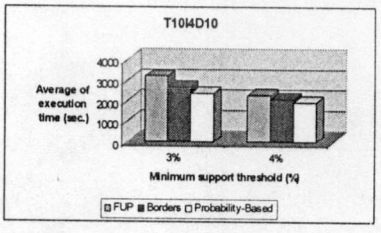



**Figure 2.** The execution time of FUP, Borders and the proposed algorithm

## 5. Conclusion

We have proposed probability-based incremental association rule discovery algorithm. Assuming that the two thresholds, minimum support and confidence, do not change, the algorithm can guarantee to discover all frequent itemsets. From the experiment, our algorithm has better running time than that of FUP and Borders algorithm. In the future, further researches and experiments on the proposed algorithm will be presented.

## 6. References


[1] R. Agrawal., T. Imielinski, and A. Swami, "Mining association rules between sets of items in large databases", In Proceeding of the ACM SIGMOD Int'l Conf. on Management of Data (A CM SIGMOD '93), Washington, USA,May 1993, pp. 207-216.

[2] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules", In Proceedings of 20 th Intl Conf. on Very Large Databases (VLDB'94), pages487-499, Santiago, Chile, September 1994, pp. 478 -499.

[3] D. Cheung, J. Han, V. Ng, and C. Y. Wong, "Maintenance of discovered association rules in large databases: An incremental updating technique", In 12th IEEE International Conference on Data Engineering, February 1996, pp. 106-114.

[4] D. W. Cheung, S.D. Lee, B. Kao, "A General incremental technique for maintaining discovered assoc ation rules" , In Proceedings of the 5 th Intl. Conf. on Database Systems for Advanced Applications (DASFAA'97), Melbourne, Australia, April 1997, pp. 185-194.

[5] H. Toivonen. "Sampling Large Databases for Association Rules", Proceeding of the 22th International conference on Very Large Data Bases, September 1996,pp. 134-145.

[6] S. Thomas, S. Bodagala, K. Alsabti, and S. Ranka, "An efficient algorithm for the incremental updation of association rules in large databases" , In Proceedings of the 3rd Intl. Conf. on Knowledge Discovery and Data Mining (KDD'97), New Port Beach, California, August 1997, pp. 263-266.

[7] C.C. Chang, Y.C. Li and J.S. Lee, "An efficient algorithm for incremental mining of association rules", Proceedings of the 15th international workshop on research issues in data engineering: stream data mining and applications (RIDE-SDMA'05), IEEE, 2005.

[8] R. Feldman, Y. Aumann, and O. Lipshtat. "Borders: An efficient algorithm for association generation in dynamic databases",Journal,Intelligent Information System, 1990, pp. 61-73.

[9] T.P. Hong C.Y. Wang and Y.H. Tao, "A new incremental data mining algorithm using pre-large itemsets", Journal, Intelligent Data Analysis, Vol. 5, No.2, pp. 111-129, 2001.

[10] R. Amornchewin and W. Kreesuradej, "Incremental association rule mining using promising frequent itemset algorithm", In Proceeding 6th International Conference on Information, Communications and Signal Processing, Dec. 10-13 2007, pp.1-5.

# Incremental Association Rule Mining Using Promising Frequent Itemset Algorithm

Ratchadaporn Amornchewin
Faculty of Information Technology
King Mongkut's Institute of Technology Ladkrabang
Bangkok, 10520 Thailand
ramornchewin@yahoo.com

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

*Abstract*— **Association rule discovery is an important area of data mining. In dynamic databases, new transactions are appended as time advances. This may introduce new association rules and some existing association rules would become invalid. Thus, the maintenance of association rules for dynamic databases is an important problem. In this paper, promising frequent itemset algorithm, which is an incremental algorithm, is proposed to deal with this problem. The proposed algorithm uses maximum support count of 1-itemsets obtained from previous mining to estimate infrequent itemsets, called promising itemsets, of an original database that will capable of being frequent itemsets when new transactions are inserted into the original database. Thus, the algorithm can reduce a number of times to scan the original database. As a result, the algorithm has execution time faster than that of previous methods. This paper also conducts simulation experiments to show the performance of the proposed algorithm. The simulation results show that the proposed algorithm has a good performance.**

*Keywords*—**association rule, maintain association rule, incremental associatin rule**

## I. INTRODUCTION

Data mining is one of the processes of Knowledge Discovery in Database (KDD) that is used for extracting information or pattern from large database. One major application area of data mining is association rule mining [1] that discovers hidden knowledge in database. The association rule mining problem is to find out all the rules in the form of X => Y, where X and Y $\subset$ I are sets of items, called itemsets. The association rule discovery algorithm is usually decomposed into 2 major steps. The first step is find out all large itemsets that have support value exceed a minimum support threshold and the second steps is find out all the association rules that have value exceed a minimum confidence threshold.

However, a database is dynamic when new transactions are inserted into the database. This may introduce new association rules and some existing association rules would become invalid. As a brute force approach, apriori may be reapplied to mining the whole dynamic database when the database has been changed. However, this approach is very costly even if small amount of new transactions is inserted into a database. Thus, the association rule mining for a dynamic database is an important problem. Several research works [7, 8, 9, 10, 11] have proposed several incremental algorithms to deal with this problem. Review of related works will be introduced in section 2.

In this paper, a new incremental algorithm, called promising frequent itemset algorithm, is introduced. The goal of this work is to solve the efficient updating problem of association rules after a nontrivial number of new records have been added to a database. Our approach introduces a promising frequent itemset for an infrequent itemset that has capable of being a frequent itemset after a number of new records have been added to a database. This can reduce a number of times to scan an original database. As a result, the algorithm has execution time faster than that of previous methods.

The remaining of this paper is organized as follows. We brief review of related works in Section 2. The Promising large itemset algorithm is described in Section 3. We evaluate the performance in Section 4. Finally, we conclude the work of this paper in section 5.

## II. RELATED WORK

An influential algorithm for association rule mining is Apriori [2]. Apriori computes frequent itemsets in the large database through several iterations based on a prior knowledge. Each iteration has 2 steps. For each iteration with 2 steps, processes are join and prune step. For an frequent itemset, its support must be higher than a user-specified minimum support threshold. The association rule can be discoverd based on frequent itemsets that must be higher than user-specified minimum confidence.

For dynamic databases, several incremental updating techniques have been developed for mining association rules. One of the previous work for incremental association rule mining is FUP algorithm that was presented by Cheung et al [3]. FUP algorithm is the first incremental updating technique

1-4244-0983-7/07/$25.00 ©2007 IEEE

for maintenance association rules when new data are inserted to database. Based on the framework of Apriori algorithm, FUP computes frequent itemsets using large itemsets found at the previous iteration. The major idea of FUP is re-use frequent itemsets of previous mining to update with incremental database. The key performance of FUP is pruning technique to reduce the number of candidate set in update process. The extension algorithm of FUP is FUP2 [4] that is proposed to handle all update cases when database are added to, deleted from a database.

Ayan et al [5] present an algorithm called UWEP (Update With Early Pruning). UWEP follows the approaches of FUP and partition algorithm. It employs a dynamic look-ahead strategy in updating existing large itemsets by detecting and pruning superset of large itemsets in an original database that will no longer remain large in updated database. UWEP scans at most once in both original database and incremental database. UWEP generates smaller candidate set from the set of itemsets that are large both an original and incremental database.

Furthermore, negative border algorithm [6], an incremental mining algorithm based on FUP, reduces to scan original database and keeps track of large itemsets and negative border when transaction is added to or delete from database. Negative border consists of all itemsets that are candidates of the level-wise method. An itemset is in the negative border did not have enough support but all its subsets are frequent. If the negative border of large itemsets expands, this algorithm is required to full scan a whole database. This is the case because negative border algorithm does not cover all large itemsets in updated database.

## III. PROMISING FREQUENT ITEMSET ALGORITHM

In an observation the itemset will be frequent itemset in updated database if it is member of large itemset in original database or incremental database. The main problem of incremental update is changing of frequent itemset that cause to re-execute from original database again.

In this paper we present the new idea to avoid scanning the original database. Then we compute not only frequent itemset but also compute itemset that may be potentially large in an incremental database called "Promising frequent Itemset".

An algorithm find all possible k-itemset of promising frequent itemset in original database. If member of frequent for each iteration is more than or equal to k-itemset. This idea is guarantee that promising frequent itemset algorithm are cover all frequent itemset that occur in updated database.Thus, updating the new transactions are quickly because it can use the information from the existing original database.

In this section we describe our algorithm into 2 subsection. In our approach, an original database is firstly mined and all frequent itemsets and promising frequent itemset. Secondly each incremental dataset in mined and updated to frequent and promising frequent itemset. The result of updating, some infrequent itemsets or new itemsets may be changed into frequent itemset.

### *A. Original database Discovery*

A dynamic database may allow insert new transactions. This may not only invalidate existing association rules but also activate new association rules. Maintaining association rules for a dynamic database is an important issue. Thus, this paper proposes a new algorithm to deal with such updating situation. Our assumption for the new algorithm is that the statistics of new transactions slowly change from original transactions. According to the assumption, the statistics of old transactions, obtained from previous mining, can be utilized for approximating that of new transactions. Therefore, Support count of itemsets obtained from previous mining may slightly different from support count of itemsets after inserting new transactions into an original database that contains old transactions. The new algorithm uses maximum support count of 1-itemsets obtained from previous mining to estimate infrequent itemsets of an original database that will capable of being frequent itemsets when new transactions are inserted into the original database. With maximum support count and maximum size of new transactions that allow insert into an original database, support count for infrequent itemsets that will be qualified for frequent itemsets, i.e. min_pl, is shown in equation 1:

$$\min\_\sup_{DB} - \left( \frac{maxsupp}{total\ size} \times inc\_size \right) \le \min\_PL < \min\_\sup_{DB} \quad (1)$$

where $\min\_\sup_{(DB)}$ is minimum support count for an original database, maxsupp is maximum support count of itemsets, current size is a number of transaction of an original database and inc_size is a maximum number of new transactions.

Here, a promising frequent itemsets is defined as following definition:

Definition A promising frequent itemset is an infrequent itemset that satisfies the equation 1.

As an example, an original database shown in figure 1. has 10 transactions, i.e. |DB|=10. Then, three new transactions is inserted into the original database, i.e. |db| =3. Here, minimum support count for mining association rules is set to 4 (40 percent). From figure 1, maximum support count of 1-itemsets of the original database is 7. min_PL is computed as the follows:

$$\min\_PL = 4 - \left( \frac{7}{10} \times 3 \right) = 2 \quad (2)$$

According to equation 2, if any itemset has support count at least 2 but less than 4, then it will be promising frequent itemsets. Thus, the frequent 1- itemset is {A, B, C} and the promising frequent 1- itemset is {D, E}.

In this paper, apriori algorithm is applied to find all possible frequent k- itemsets and promising frequent k-itemsets. Apriori scans all transactions of an original database for each iteration with 2 steps processes are join and prune step. Unlike typical apriori algorithm, items in both frequent k- itemsets and promising frequent k-itemsets can be joined together in the join step. For a frequent item, its support count must be higher than

a user-specified minimum support count threshold and for a promising frequent item, its support count must be higher than min_PL but less than the user-specified minimum support count. As examples, figure 2. and 3. show the promising frequent and frequent 2- itemsets and the promising frequent and frequent 3- itemsets respectively.

| TID | List of item |
|---|---|
| 1 | A, B, E |
| 2 | B, D |
| 3 | B,C |
| 4 | A, B,D |
| 5 | A, C |
| 6 | B, C |
| 7 | A, C |
| 8 | A, B, C, E |
| 9 | A, B, E |
| 10 | A,C |

| Itemset | support |
|---|---|
| A | 7 |
| B | 7 |
| C | 6 |
| D | 2 |
| E | 3 |

Figure 1. Transaction data and candidate 1-itemsets

| C2 | support |
|---|---|
| AB | 4 |
| AC | 4 |
| AD | 1 |
| AE | 3 |
| BC | 3 |
| BD | 2 |
| BE | 3 |
| CD | 0 |
| CE | 1 |
| DE | 0 |

| L2 | Support |
|---|---|
| AB | 4 |
| AC | 4 |
| PL2 | support |
| AE | 3 |
| BC | 3 |
| BD | 2 |
| BE | 3 |

Figure 2. candidate , frequent and promise frequent 2-itemset

| C3 | Support |
|---|---|
| ABC | 1 |
| ABE | 3 |
| ACE | 1 |
| BCD | 0 |
| BCE | 0 |
| BDE | 0 |

| PL3 | support |
|---|---|
| ABE | 3 |

Figure 3. candidate, frequent and promise frequent 3- itemset

### *B. Updating frequent and promising frequent itemsets*

When new transactions are added to an original database, an old frequent k-item could become an infrequent k-item and an old promising frequent k-item could become a frequent k-item. This introduces new association rules and some existing association rules would become invalid. To deal with this problem, all k-items must be updated when new transactions are added to an original database. In this section, we explain how to update all old items.

The size of an updated database increases when new transactions are inserted into an original database. Thus, min_PL must be recalculated in order to associate with the new size of an updated database. min_PL $_{(update)}$ is computed as the follows:

$$\min\_PL_{DB\cup db} = \min\_\sup_{DB\cup db} - \left( \frac{\max supp}{total\ size} \times inc\_size \right) \quad (3)$$

Then, If any k-item has support count greater than or equal to min_sup$_{(DB\cup db)}$, this itemset is moved to a frequent k-item of an updated database. In the other case, if any k-item has support count less than min_sup$_{(DB\cup db)}$ but it is greater or equal to min_PL$_{(update)}$ , this k-item is moved to a promise frequent itemset of an upated database. The following algorithms are developed to update frequent and promising frequent k-tems of an updated database.

The first algorithm, shown in figure 4, updates a frequent and a promising frequent 1-itemset. After obtaining support count of candidate 1-itemsets of an incremental database, the support count of the candidate 1-items is summed to that of the 1-items of an original database. Then, If any item has support count greater than or equal to min_sup$_{(DB\cup db)}$, this item is moved to a frequent 1-itemset of an updated database. In the other case, if any item has support count less than min_sup$_{(DB\cup db)}$ but it is greater or equal to min_PL$_{(update)}$ , this item is moved to a promise frequent 1-itemset of an updated database. In addition, if any item is a new frequent item or a new promising frequent item, this item will be added to Temp1 set. Then, Temp1 is joined and pruned with both a promise frequent k-itemset and a frequent k-itemset to obtain Temp_newCk. The algorithm for joining and pruning items is shown in figure 5.

The k-itemsets of the Temp_newCk are scanned in an incremental database. A k-itemset of Temp_newCk can become a frequent itemset in an updated database only if the k-itemset of the Temp_newCk is a frequent itemset in an incremental database. Thus, if a itemset of the Temp_newCk has support count greater than or equal to min_sup(db), the item is moved to an estimated frequent k-itemset. Similarly, a k-itemset of Temp_newCk can become a promising frequent itemset in an updated database only if the k-itemset of the Temp_newCk is a frequent itemset in an incremental database. Thus, if a k-itemset of Temp_newCk has support count less than min_sup(db) but greater or equal to min_PL(update) or min_PL(DB), the k-itemset is moved to an estimated promising frequent k-itemset. Then, both the estimated promise frequent k-itemsets and the estimated frequent k-itemsets are added to Temp_scanDB. Figure 6 shows updating k-itemset algorithm for k>=2.

As the last phase for the incremental algorithm, the k-items of Temp_scanDB is scanned in an original database to update their support count. Like previous cases, If any k-item has support count greater than or equal to min_sup$_{(DB\cup db)}$, this k-item is moved to a frequent k-itemset and if any k-item has support count less than min_sup$_{(DB\cup db)}$ but it is greater or equal to min_PL$_{(update)}$ , this k-item is moved to a promise frequent k-itemset. The algorithm for finding support count of k>=2 itemsets shows in figure 7.

**Algorithm 1 Updating frequent and promising frequent 1-itemset**

**Input :**
(1) $L^1_{DB}$: the set of all frequent 1-itemset in original database,
(2) $PL^1_{DB}$ : the set of all promising frequent 1-itemset original database,
(3) $C^1_{DB}$ : candidate 1-itemset of original database,
(4) $C^1_{db}$: candidate 1-itemset of incremental database.

**Output :**
(1) $L^1_{(DB \cup db)}$: frequent itemset in updated database,
(2) $PL^1_{(DB \cup db)}$ : promising frequent itemset in updated database,
(3) new frequent itemsets : new frequent itemset in updated database ,
(4) new promising frequent itemsets : new promising frequent itemset in updated database
(5) Temp_newCk : new candidate 2-itemset in updated database

```
1   C1db = all 1-itemsets in db with support >0
2   k=1
3   While C1db > 0 do
4   For each X ∈ C1DB do
5       X.support(DB∪db) = X.supportDB + X.supportdb
6       If (X ∉L1DB or X ∉ PL1DB ) and
7         (X.support(DB∪db) ) >= min_sup(DB∪db) Then
8          Add X to L1(DB∪db)
9          Add X to temp1
10      For each X ∈ L1DB do
11        If X.support(DB∪db)>= min_sup(DB∪db) Then
12          Add X to L1(DB∪db)
13        Else
14          If X.support(DB∪db)>= min_PL(DB∪db) Then
15            Add X to PL1(DB∪db)
16      For each X ∈ PL1DB do
17        If X.support(DB∪db)>= min_sup(DB∪db) Then
18          Add X to L1(DB∪db)
19          Add X to temp1
20        Else
21          If X.support(DB∪db)>= min_PL(DB∪db) Then
22            Add X to PL1(DB∪db)
23  For each X ∉ C1DB do
24     Add X to C1(DB∪db)
25     If X.supportdb >= minsup(DB∪db)  (new item in db) Then
26        Add X to L1(DB∪db)
27        Add X to temp1
28     Else
29       If X.support(db)>= min_PL(DB∪db) Then
30         Add X to PL1(DB∪db)
31         Add X to temp1
32  If temp1 ≠ {} Then
33     Y ∈ temp1
34     C2(DB∪db) (new) = gen_newcandidate (Y)
35     Clear temp1
36     Add C2(DB∪db) (new) to Temp_newCk
37  k=k+1
```

Figure 4. Updating frequent and promising frequent 1-itemset algorithm

**Algorithm 2 Gen_newcandidate**

**Input :**
(1 )$L^k_{(DB \cup db)}$: frequent k-itemset in updated database,
(2) $PL^k_{DB \cup db)}$ : promising k-itemset in updated database.
(3) Temp1 : new frequent k-itemset in updated database

**Output :**
(1) new $C^{k+1}$: new candidate k+1-itemset in updated database.

```
1   If k <= (length(L) + length(PL))
2     For each Y ∈ Temp1
3       Ck+1 (new) = Y *(Lk(DB∪db))∪PLk(DB∪db))
4     For c ∈ Ck+1 (new)
5       Delete c from Ck+1DB(new) if all subset of c is in Lk or PLk
```

Figure 5. Generating new candidate itemset algorithm

**Algorithm 3 Update frequent and promising frequent itemsets for k>= 2 itemset**

**Input :**
(1) $L^k_{DB}$ : frequent k-itemset in original database,
(2) $PL^k_{DB}$ : promising frequent k-itemset in original, database
(3) Temp_newCk : new candidate k-itemset in updated database.

**Output :**
(1) $L^k_{(DB \cup db)}$: frequent k-itemset in updated database,
(2) $PL^k_{DB \cup db)}$ : Promising frequent k-itemset in updated database,
(3) Temp_scanDB : estimated frequent k-itemset and estimated promising frequent k-itemset in updated database
(4) Temp1 : new estimated frequent k-itemset and new estimated promising frequent k-itemset in updated database
(5) Temp_newCk : new candidate k+1-itemset in updated database.

```
1   k=2
2   While k <= (length(Lk) +length(PLk)) do
3   Scan db for ∀(Lk) , ∀(PLk) and ∀(items) ∈ Temp_newCk
4     X.support (DB∪db) = X.supportDB + X.supportdb
5   For each X ∈ LkDB do
6     If X.support(DB∪db)>= min_sup(DB∪db) Then
7       Add X to Lk(DB∪db)
8     Else
9       If X.support(DB∪db)>= min_PL(DB∪db) Then
10        Add X to PLk(DB∪db)
11  For each X ∈ PLkDB do
12    If X.support(DB∪db)>= min_sup(DB∪db) Then
13      Add X to Lk(DB∪db)
14      Add X to temp1
15    Else
16      If X.support(DB∪db)>= min_PL(DB∪db) Then
17        Add X to PL1(DB∪db)
18  For each Y ∈ Temp_newCk do
19   If Y.support(db)>= min_sup(db) Then
20     Add Y to Temp_scanDB(Lk(DB∪db))
21     Add Y to Temp1(Lk(DB∪db))
22   Else
23     If Y.support(db)>= (min_PL (DB∪db))
24       or Y.support(db)>= min_PL (DB) )Then
25       Add Y to Temp_scanDB(PLk(DB∪db))
26       Add Y to Temp1(PLk(DB∪db))
27  Clear Temp_newCk
28  If Temp1 ≠ {} Then
29     Y ∈ Temp1
30     Ck+1(DB∪db) (new) = gen_newcandidate (Y)
31     Clear Temp1
32     Add Ck+1,(DB∪db) (new) to Temp_newCk
33  k=k+1
    If Temp_scanDB ≠ {} Then Find_SuppcountDB
```

Figure 6. Update k >= 2 itemset algorithm

**Algorithm 4 Find_SuppcountDB**

**Input :**

(1) $L^{k}_{(DB\cup db)} \in$ Temp_scanDB : Estimated frequent k-itemset,

(2) $PL^{k}_{(DB\cup db)} \in$ Temp_scanDB : Estimated promising frequent k-itemset

**Output :**

(1) $L_{(DB\cup db)}$ : frequent k-itemset in updated database,

(2) $PL_{(DB\cup db)}$ : promising frequent k-itemset in updated database

```
1     For each W ∈ Temp_scanDB
2       Scan DB for W
3       W.support(DB∪db) = W.supportDB + W.supportdb
4          If W.support(DB∪db) >= min_sup(DB∪db) Then
5            Add W to L^k(DB∪db)
6          Else
7            If W.support(DB∪db) >= min_PL(DB∪db) Then
8              Add W to PL^k(DB∪db)
9       Clear Temp_scamDB
```

Figure 7. Finding support count in an original database algorithm

## IV. EXPERIMENT

To evaluate the performance of promising frequent algorithm, the algorithm is implemented and tested on a PC with a 2.8 GHz Pentium 4 processor, and 1 GB main memory. The experiments are conducted on a synthetic dataset, called T10I4D10K. The technique for generating the dataset is proposed by Agrawal and etc. [1]. The synthetic dataset comprises 20,000 transactions over 100 unique items, each transaction has 10 items on average and the maximal size itemset is 4

Firstly, the proposed algorithm is used to find association rules from an original database of 10,000 transactions. Then, several sizes of incremental databases, i.e. 10%, 20%, 30%, 40% and 50% of the original database, are added to the original database. For comparison purpose, FUP algorithm is also used to find association rules from the same original database and the same incremental databases. The experimental results with various minimum support thresholds are shown in Table1 and Figure 8. From the results, the proposed algorithm has better running time than that of FUP algorithm.

TABLE I. EXECUTION TIME WITH VARYING SIZE OF INCREMENTAL DATABASE

| Min_sup | Algorithm | Execution time (sec.) Percent of Incremental database size | | | | |
|---|---|---|---|---|---|---|
| | | 10% | 20% | 30% | 40% | 50% |
| 4% | Promising Frequent Itemset | 185.2 | 546 | 1047 | 1563.4 | 1935.5 |
| | FUP | 2521.5 | 5012 | 7110.7 | 9996.3 | 11539 |
| 5% | Promising Frequent Itemset | 191.26 | 430.11 | 632.06 | 855.39 | 1048 |
| | FUP | 2023.6 | 4162.4 | 5970.4 | 8678.1 | 10681 |
| 6% | Promising Frequent Itemset | 93.562 | 180.41 | 304.56 | 366.78 | 452.7 |
| | FUP | 1680.5 | 3868 | 5446.8 | 6889.5 | 9516.3 |
| 7% | Promising Frequent Itemset | 52.985 | 98.296 | 463.22 | 1808.4 | 2147.1 |
| | FUP | 1350.6 | 2723.7 | 4873.3 | 5972.3 | 8856.3 |

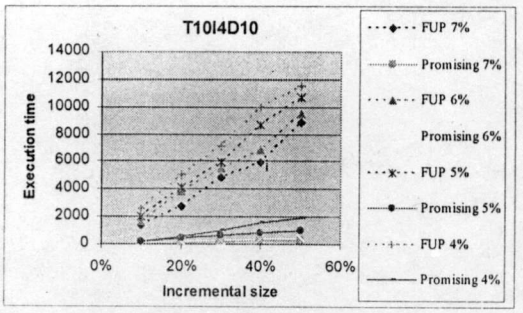



Figure 8. Execution Time comparison

## V. CONCLUSIONS

We have proposed promising frequent algorithm for incremental association rule mining. Assuming that the two thresholds, minimum support and confidence, do not change, the promising frequent algorithm can guarantee to discover frequent itemsets. From the experiment, our algorithm has better running time than that of FUP algorithm. In the future, further researches and experiments on the proposed algorithm will be presented.

## REFERENCES


[1] R. Agrawal., T. Imielinski, and A. Swami, "Mining association rules between sets of items in large databases," In Proc. of the ACM SIGMOD Int'l Conf. on Management of Data (A CM SIGMOD '93), Washington, USA,May 1993.

[2] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules." In Proceedings of 20 th Intl Conf. on Very Large Databases (VLDB'94), pages487-499,Santiago, Chile, 1994.

[3] D. W. Cheung, J. Han, V. T. Ng, and C. Y. Wong, "Maintenance of discovered association rules in large databases: An incremental updating technique,"In 12th IEEE International Conference on Data Engineering,1996.

[4] D. W. Cheung, S.D. Lee, B. Kao, "A General incremental technique for maintaining discovered association rules," In Proceedings of the 5 th Intl. Conf. on Database Systems for Advanced Applications (DASFAA'97), Melbourne, Australia, April 1997.

[5] N.F. Ayn, A.U. Tansel, and E. Arun, "An efficient algorithm to update large itemsets with early pruning." Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, August 1999.

[6] S. Thomas, S. Bodagala, K. Alsabti, and S. Ranka, "An efficient algorithm for the incremental updation of association rules in large databases," In Proceedings of the 3rd Intl. Conf. on Knowledge Discovery and Data Mining (KDD'97), New Port Beach, California,1997.

[7] C. H. Lee, C. R. Lin , and M. S. Chen, "Sliding-Window Filtering: An Efficient Algorithm for Incremental Mining ," ACM , 2000

[8] C.C. Chang, Y.C. Li and J.S. Lee, "An efficient algorithm for incremental mining of association rules," Proceedings of the 15th international workshop on research issues in data engineering: stream data mining and applications (RIDE-SDMA'05) ,IEEE, 2005

[9] A. A. Veloso et al., "Mining frequent itemsets in evolving databases," In Proc. 2nd SIAM Intl. Conf. on Data Mining, Arlington, VA, Apr. 2002

[10] K.L. Lee, G. Lee and A. L.P. Chen, "Efficient Graph-based algorithm for discovering and maintaining knowledge in large database," Proceedings of the third pacific-asia conference on methodologies for knowledge discovert and data mining, April 1999.

[11] N. L. Sarda and N. V. Srinivas, "An adaptive algorithm for incremental. mining of association rules," In Proc. 9th Intl.Workshop on Database and Expert System Applications,Vienna, Austria, pp. 240-245, Aug 1998.