วิธีการหากฎความสัมพันธ์แบบใหม่โดยต้นไม้แสดงรายการความถี่

นายโกเมศ อัมพวัน

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิทยาศาสตร์คอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2548

ISBN 974-17-3712-2

A NOVEL APPROACH OF MINING ASSOCIATION RULES WITH FREQUENT ITEM TREE

Mr. Komate Amphawan

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Science Program in Computer Science
Department of Computer Engineering
Faculty of Engineering
Chulalongkorn University
Academic Year 2005

ISBN 974-17-3712-2

หัวข้อวิทยานิพนธ์	วิธีการหากฎความสัมพันธ์แบบใหม่โดยต้นไม้แสดงรายการความถี่
โดย	นายโกเมศ อัมพวัน
สาขาวิชา	วิทยาศาสตร์คอมพิวเตอร์
อาจารย์ที่ปรึกษา	อาจารย์ ดร. อรรถสิทธิ์ สุรฤกษ์

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

..... คณบดีคณะวิศวกรรมศาสตร์

(ศาสตราจารย์ ดร. ดิเรก ลาวัณย์ศิริ)

คณะกรรมการสอบวิทยานิพนธ์

..... ประธานกรรมการ

(รองศาสตราจารย์ ดร. วันชัย ริ้วไพบูลย์)

..... อาจารย์ที่ปรึกษา

(อาจารย์ ดร. อรรถสิทธิ์ สุรฤกษ์)

..... กรรมการ

(ผู้ช่วยศาสตราจารย์ ดร. ภัทรสินี ภัทรโกศล)

..... กรรมการ

(ผู้ช่วยศาสตราจารย์ ดร. อานนท์ รุ่งสว่าง)

โกเมศ อัมพวัน : วิธีการหากฎความสัมพันธ์แบบใหม่โดยต้นไม้แสดงรายการความถี่.
(A NOVEL APPROACH OF MINING ASSOCIATION RULES WITH FREQUENT ITEM TREE) อ. ที่ปรึกษา : อ.ดร. อรรถสิทธิ์ สุรฤกษ์, 54 หน้า. ISBN 974-17-3712-2.

ในปัจจุบันงานวิจัยเกี่ยวกับการวิเคราะห์หารูปแบบความสัมพันธ์ของข้อมูลจากฐานข้อมูลขนาดใหญ่มีบทบาทและความสำคัญในปัญหาของการทำเหมืองข้อมูลหรือการขุดค้นข้อมูล นอกจากนี้มีนักวิจัยจำนวนมากให้ความสนใจและทำการศึกษาเพื่อการพัฒนากระบวนการหรือคิดค้นวิธีการใหม่ในการหาความสัมพันธ์ให้มีประสิทธิภาพมากยิ่งขึ้น การสร้างกฎความสัมพันธ์เป็นวิธีหนึ่งในการสืบหากฎความสัมพันธ์ร่วมของกลุ่มข้อมูลในเชิงปริมาณ โดยที่แต่ละกฎถูกระบุด้วยค่าสนับสนุนและค่าความเชื่อมั่น โดยทั่วไปกฎความสัมพันธ์ถูกนำไปใช้ในการวิเคราะห์หาพฤติกรรมการซื้อของลูกค้า

การหากฎความสัมพันธ์ของข้อมูลประกอบด้วย 2 ขั้นตอนใหญ่ๆ ได้แก่ การหาเซตรายการความถี่ซึ่งก็คือ เซตของรายการที่มีค่าสนับสนุนเกินค่าสนับสนุนขั้นต่ำที่กำหนดให้ และการนำเอาเซตรายการความถี่ที่สามารถหาได้สร้างเป็นกฎความสัมพันธ์ โดยในขั้นตอนแรกจะเป็นขั้นตอนที่ใช้เวลาและหน่วยความจำมากเนื่องจากต้องทำการอ่านข้อมูลจากฐานข้อมูลเพื่อหาการเกิดร่วมกันของข้อมูลจำนวนมาก จึงเป็นเหตุให้มีนักวิจัยจำนวนมากให้ความสนใจที่จะปรับปรุงการหาเซตรายการความถี่จากฐานข้อมูล ในงานวิจัยนี้ได้นำเสนออัลกอริทึมเพื่อลดเวลาในการคำนวณซึ่งเป็นอัลกอริทึมที่พัฒนาจากเอฟพี-กโรธอัลกอริทึม โดยปรับปรุงขั้นตอนการสร้างต้นไม้แสดงรายการความถี่และการหาเซตรายการความถี่จากต้นไม้แสดงรายการความถี่ การปรับปรุงการสร้างต้นไม้แสดงรายการความถี่จะลดขั้นตอนการเรียงลำดับรายการในรายการเปลี่ยนแปลงทุกรายการเปลี่ยนแปลง และ การปรับปรุงการหาเซตรายการความถี่จะทำการรวมค่าสนับสนุน การหาสับเซตที่จำเป็น และการตัดเล็มต้นไม้แทนการหาคอนดิชันนอลแพทเทินเบซ และการสร้างคอนดิชันนอลเอฟพี-ทรี จากการทดลองและเปรียบเทียบเวลาการหาเซตรายการความถี่ปรากฏว่าการหาเซตรายการความถี่จากต้นไม้แสดงรายการความถี่ใช้เวลาในการคำนวณน้อยกว่าเอฟพี-กโรธอัลกอริทึม และ ความซับซ้อนเชิงเวลาของทั้งสองอัลกอริทึมมีค่าเท่ากับ $\Theta(n)$ เมื่อ n คือจำนวนรายการเปลี่ยนแปลงในฐานข้อมูล

ภาควิชา วิศวกรรมคอมพิวเตอร์ ลายมือชื่อนิสิต โกเมศ อัมพวัน
สาขาวิชา วิทยาศาสตร์คอมพิวเตอร์ ลายมือชื่ออาจารย์ที่ปรึกษา
ปีการศึกษา 2548

46702324 : MAJOR COMPUTER SCIENCE

KEY WORD: DATA MINING / FP-GROWTH ALGORITHM / FREQUENT ITEMSETS / ASSOCIATION RULES / SUPPORT / MINIMUM SUPPORT / CONFIDENCE / FREQUENT ITEM TREE / CONDITIONAL PATTERN BASE / CONDITIONAL FP-TREE

KOMATE AMPHAWAN : A NOVEL APPORACH OF MINING ASSOCIATION RULES WITH FREQUENT ITEM TREE. THESIS ADVISOR: ATHASIT SURARERKS, Ph.D., 54 pp. ISBN 974-17-3712-2.

One of the most well-studied problem in data mining is to discover association rules in market basket datasets. Association rules, whose significance is measured by support and confidence, are intended to identify relationships among sets of items. The task of mining association rules consists of two main steps. The first step is to find all itemsets whose frequencies are above minimum support. These itemsets are called frequent itemsets. The second step involves generating high confidence rules among frequent itemsets. According to the size of datasets, finding frequent itemsets is computationally the most expensive step in association rule discovery. Therefore, it is necessary to develop appropriated structure capable of high compression ratios and supporting of fast finding frequent itemsets. In this thesis, we proposes a new algorithm for frequent itemsets mining called Frequent Item Tree. It is improved from FP-growth algorithm in order to reduce computational time. The main idea of Frequent Item Tree is separate into 2 sections. First is frequent item tree building improvement which reduces transaction sorting procedure. Second is frequent itemsets mining improvement which replaces conditional pattern base and conditional FP-tree procedure with Item frequency combination, necessary subsets finding and Frequent Item Tree prunning. The experimental result shows advantages of our algorithm over FP-growth, in terms of runtime, although time complexity of them are $\Theta(n)$ whereas n is number of transactions.

Department	Computer Engineering	Student's	โอมพร อำพัน
Field of study	Computer Science	Advisor's	
Academic year	2005		

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จได้ด้วยความอนุเคราะห์ และความช่วยเหลืออย่างยิ่งจาก อาจารย์ ดร. อรรถสิทธิ์ สุรฤกษ์ อาจารย์ที่ปรึกษา ซึ่งเป็นผู้ให้ข้อคิด แนวทาง และคำปรึกษา ตลอดจนเป็นผู้ตรวจทานแก้ไข จนทำให้วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วง ขอขอบพระคุณ อาจารย์ ดร. อรรถสิทธิ์ สุรฤกษ์ เป็นอย่างสูงที่ให้ความเมตตา ช่วยเหลือ รวมทั้งโอกาส และสิ่งที่ดีแก่ผู้วิจัย เสมอมา

ขอขอบพระคุณ รองศาสตราจารย์ ดร. วันชัย ริ้วไพบูลย์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ผู้ช่วยศาสตราจารย์ ดร. ภัทรสินี ภัทรโกศล คณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย และผู้ช่วยศาสตราจารย์ ดร. อานนท์ รุ่งสว่าง คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเกษตรศาสตร์ ที่ได้กรุณาให้คำแนะนำในการแก้ไขวิทยานิพนธ์ให้มีคุณภาพยิ่งขึ้น วิทยานิพนธ์ฉบับนี้ไม่อาจจะสำเร็จได้หากไม่ได้รับความร่วมมือจากทุกท่าน และขอขอบคุณ อาจารย์ในภาควิชาวิศวกรรมคอมพิวเตอร์ จุฬาลงกรณ์มหาวิทยาลัยทุกท่าน และเพื่อนๆ ทุกๆคน ผู้ที่ให้คำแนะนำเพิ่มเติมกับผู้วิจัยเสมอมา

ท้ายนี้ขอขอบพระคุณ บิดา มารดา ที่เป็นกำลังใจสำคัญ และช่วยเหลือในทุกๆ ด้าน จน ผู้วิจัยสามารถทำวิทยานิพนธ์ฉบับนี้สำเร็จลุล่วง

สารบัญ

หน้า

บทที่ หน้า

สารบัญตาราง

ตารางที่ หน้า

สารบัญภาพ

ภาพที่ หน้า

บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

การทำเหมืองข้อมูล (data mining) หรือเรียกอีกอย่างหนึ่งว่า การค้นหาความรู้จากฐานข้อมูลขนาดใหญ่ (knowledge discovery in very large databases, KDD) เป็นงานวิจัยที่เพิ่งเกิดขึ้นประมาณสิบปีที่ผ่านมา โดยนักวิจัยกลุ่มหนึ่งซึ่งมีความสนใจในเรื่องการวิเคราะห์ข้อมูลอัตโนมัติ (automating data analysis) หลักการของการทำเหมืองข้อมูลคือ การค้นหาสารสนเทศที่สำคัญโดยไม่ทราบมาก่อนและมีแนวโน้มว่าสารสนเทศนั้นจะมีประโยชน์ โดยทำการค้นหาสารสนเทศจากข้อมูลในฐานข้อมูล (non-trivial extraction of implicit, previously unknown, and potentially useful information from data in databases) เนื่องจากในปัจจุบันข้อมูลในฐานข้อมูลมีจำนวนมากและระบบจัดการฐานข้อมูลทั่วไปไม่สามารถจัดการข้อมูลเหล่านั้นได้อย่างมีประสิทธิภาพ จึงเป็นเหตุให้การทำเหมืองข้อมูลได้รับความนิยม เพราะการทำเหมืองข้อมูลสามารถช่วยในการวิเคราะห์ข้อมูลที่เป็นประโยชน์ซึ่งมีผลต่อการตัดสินใจ หรือที่เรียกว่า ระบบการสนับสนุนการตัดสินใจ (decision support system) ประกอบกับเครื่องคอมพิวเตอร์สมรรถนะสูงมีราคาต่ำลงจึงเป็นการสนับสนุนให้มีการทำเหมืองข้อมูลมากขึ้น เพราะเทคนิคของการทำเหมืองข้อมูลต้องใช้อัลกอริทึมที่มีความซับซ้อนและมีความต้องการในการคำนวณมาก จึงจำเป็นต้องใช้เครื่องคอมพิวเตอร์สมรรถนะสูง โดยข้อมูลที่จะนำมาใช้ในการทำเหมืองข้อมูลจะต้องมีขนาดใหญ่เกินกว่าจะพิจารณาหาความสัมพันธ์ของข้อมูลได้จากการสังเกต หรือโดยการใช้ระบบการจัดการฐานข้อมูล (database management system, DBMS) และต้องเป็นข้อมูลที่ไม่มีการเปลี่ยนแปลงตลอดช่วงที่ทำการทำเหมืองข้อมูลหากข้อมูลเป็นข้อมูลที่เปลี่ยนแปลงตลอดเวลาจะทำให้ผลลัพธ์ที่ได้จากการทำเหมืองมีความถูกต้องในช่วงเวลาหนึ่งเท่านั้น ดังนั้น เพื่อให้ได้ผลลัพธ์ที่ถูกต้องและเหมาะสมอยู่ตลอดเวลา จะต้องทำการทำเหมืองข้อมูลใหม่ทุกครั้งในช่วงเวลาที่เหมาะสม หลายๆ เทคนิคของการทำเหมืองข้อมูลได้ถูกพัฒนาขึ้นเพื่อค้นหาความสัมพันธ์ของข้อมูลและรูปแบบที่น่าสนใจ ซึ่งถูกเก็บไว้ในฐานข้อมูลขนาดใหญ่ งานวิจัยทางด้านการทำเหมืองข้อมูลได้ถูกนำไปประยุกต์ใช้กับงานวิจัยในหลายสาขาวิชา เช่น ทางสถิติ การเรียนรู้ของเครื่อง ปัญญาประดิษฐ์ และข่ายงานประสาทเทียม ด้วยเทคนิคต่างๆ เช่น การสร้างกฎความสัมพันธ์ (association rules) การจำแนก/แบ่ง (classification/clustering) ข้อมูลโดยใช้ความรู้ทางด้านการเรียนรู้ของเครื่อง มาช่วยในการตัดสินใจและหาความสัมพันธ์ของข้อมูลและการค้นหารูปแบบที่คล้ายกัน (similar patterns) จากฐานข้อมูล นอกจากนั้น ยังสามารถนำไปประยุกต์ใช้กับงาน

ต่างๆ เช่น ธุรกิจการค้า กิจการโทรคมนาคม การวิเคราะห์บัตรเครดิต ธนาคาร กฎหมาย การแพทย์ และอื่นๆ อีกมากมาย โดยงานวิจัยนี้จะพิจารณาเฉพาะเทคนิคของการหากฎความสัมพันธ์

การสร้างกฎความสัมพันธ์ถูกคิดค้นเพื่อใช้ในการวิเคราะห์ข้อมูลจากฐานข้อมูลขนาดใหญ่ของห้างสรรพสินค้า เช่น การวิเคราะห์หาพฤติกรรมการซื้อของลูกค้า นั่นคือ การหากฎการซื้อสินค้าเมื่อซื้อสินค้าชนิดหนึ่งแล้วจะซื้อสินค้าอีกชนิดหนึ่งด้วย ซึ่งข้อมูลเหล่านี้สามารถช่วยในกิจกรรมต่างๆ ได้มากมาย เช่น การจัดชั้นของสินค้า การเพิ่มหรือลดราคาของสินค้าแต่ละประเภท และช่วยในการวิเคราะห์ผลกำไรจากการขายสินค้าได้อีกด้วย โดยทั่วไปขั้นตอนการสร้างกฎความสัมพันธ์ของข้อมูลมีขั้นตอนหลักอยู่ 2 ขั้นตอน คือ

1. การหาเซตรายการความถี่ (frequent itemsets) หรือ เซตของรายการขนาดใหญ่ (large itemsets) เป็นการหาเซตของรายการ (itemset) ที่มีค่าความถี่หรือค่าสนับสนุน (support) มากกว่าหรือเท่ากับค่าสนับสนุนขั้นต่ำ (minimum support) ที่กำหนดไว้ และเรียกเซตของรายการที่มีค่าสนับสนุนน้อยกว่าค่าสนับสนุนขั้นต่ำว่า เซตของรายการขนาดเล็ก (small itemsets)

2. การนำเอาเซตรายการความถี่มาสร้างเป็นกฎความสัมพันธ์ของข้อมูล ซึ่งกฎแต่ละกฎจะถูกยอมรับก็ต่อเมื่อกฎนั้นต้องมีค่าความเชื่อมั่น (confidence) มากกว่าหรือเท่ากับค่าความเชื่อมั่นขั้นต่ำ (minimum confidence) ที่กำหนดไว้

งานวิจัยในด้านนี้ส่วนใหญ่จะสนใจเฉพาะขั้นตอนการหาเซตรายการความถี่ ทั้งนี้เพราะการค้นหารูปแบบ (patterns) หรือเซตรายการความถี่จากฐานข้อมูลขนาดใหญ่จำเป็นต้องใช้การคำนวณ หน่วยความจำ รวมถึงเวลาสำหรับการทำไอ/โอ (I/O) มาก และขั้นตอนการหาเซตรายการความถี่ยังมีความซับซ้อนมากกว่าขั้นตอนการหากฎความสัมพันธ์ของข้อมูล ซึ่งสามารถสร้างกฎความสัมพันธ์ได้โดยตรงจากเซตรายการความถี่ที่หาไว้ก่อนหน้า ปัญหาหลักของงานวิจัยในด้านนี้คือ การค้นหาเซตรายการความถี่ที่ถูกต้อง รวดเร็ว โดยไม่ต้องเสียเวลากับการทำไอ/โอ และใช้เวลาในการคำนวณน้อย ดังนั้น งานวิจัยนี้จึงมีแนวความคิดในการปรับปรุงและนำเสนออัลกอริทึมเพื่อลดเวลาในการหาเซตรายการความถี่ โดยทำการพัฒนาจากเอฟพี-กโรธอัลกอริทึม (FP-growth algorithm) ซึ่งจะลดขั้นตอนการเรียงลำดับข้อมูลในตารางรายการ (header table) และในรายการเปลี่ยนแปลง (transaction) นอกจากนี้ยังสนใจการปรับปรุงขั้นตอนการหาคอนดิชันนอลแพทเทินเบซ (conditional pattern base) จากเอฟพี-ทรี และการสร้างคอนดิชันนอลเอฟพี-ทรี (conditional FP-tree) ด้วย

1.2 วัตถุประสงค์ของการวิจัย

นำเสนออัลกอริทึมการหาความสัมพันธ์ของข้อมูลจากฐานข้อมูลได้อย่างถูกต้องครบถ้วน โดยใช้เวลาในการคำนวณน้อย และสามารถสร้างกฎความสัมพันธ์จากผลลัพธ์ที่ได้

1.3 ขอบเขตของการวิจัย

1. เสนออัลกอริทึมในการหาเซตรายการความถี่ที่ลดขั้นตอนในการเรียงลำดับข้อมูล
2. สามารถหาเซตรายการความถี่ทุกระดับได้อย่างถูกต้อง
3. สามารถกำหนดค่าสนับสนุนขั้นต่ำเพื่อหาเซตรายการความถี่ได้ตามที่ต้องการ
4. สามารถพัฒนาโครงสร้างข้อมูลเอฟพี-ทรีเพื่อลดเวลาในการคำนวณได้
5. ฐานข้อมูลที่ทำการทดสอบจะต้องเป็นแบบรายการเปลี่ยนแปลง (transaction)

1.4 ขั้นตอนและวิธีดำเนินการวิจัย

1. ศึกษางานวิจัยการหาความสัมพันธ์ของข้อมูลที่มีในปัจจุบัน
2. วิเคราะห์ปัญหาจากงานวิจัยเหล่านั้น
3. ออกแบบโครงสร้างข้อมูล
4. ออกแบบแนวคิดและรูปแบบของอัลกอริทึมเพื่อใช้แก้ปัญหา
5. ปรับปรุงอัลกอริทึมให้ทำงานได้ตามเป้าหมาย
6. จัดทำวิทยานิพนธ์

1.5 ประโยชน์ที่คาดว่าจะได้รับจากการวิจัย

1. ได้อัลกอริทึมการหาความสัมพันธ์ของข้อมูลที่ใช้เวลาในการคำนวณน้อย
2. ได้อัลกอริทึมการหาความสัมพันธ์ของข้อมูลที่ให้ผลลัพธ์ถูกต้อง ครบถ้วน
3. สามารถนำเทคนิคการหาเซตรายการความถี่ไปประยุกต์ใช้กับปัญหาจริงในธุรกิจได้

1.6 ผลงานที่ตีพิมพ์จากวิทยานิพนธ์

ส่วนหนึ่งของวิทยานิพนธ์นี้ได้รับการตีพิมพ์เป็นผลงานวิชาการในหัวข้อเรื่องดังต่อไปนี้

1. “Association Rules Mining with Frequent Item Tree” โดย โกเมศ อัมพวัน ศิริชัย จันทร์สมัคร และอรรถสิทธิ์ สุรฤกษ์ ในงานประชุมวิชาการ National Computer Science and Engineering Conference (NCSEC2004)

2. “An Approach of Adaptive Frequent Item Tree for Association Rules Mining” โดย โกเมศ อัมพวัน และอรรถสิทธิ์ สุรฤกษ์ ในงานการประชุมวิชาการทางวิศวกรรมศาสตร์ มหาวิทยาลัยสงขลานครินทร์ ครั้งที่ 3 (PSU-Conference)

3. “An Approach of Frequent Item Tree for Association Generation” โดย โกเมศ อัมพวัน และอรรถสิทธิ์ สุรฤกษ์ ในงานการประชุมวิชาการ “Artificial Intelligent and Soft Computing (ASC2005)” Benidorm, Spain ในระหว่างวันที่ 12-14 กันยายน 2548

บทที่ 2

ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

ในบทนี้จะนำเสนอทฤษฎีและงานวิจัยที่เกี่ยวข้องในวิทยานิพนธ์ฉบับนี้ โดยเริ่มต้นจากนิยามและความรู้พื้นฐานที่จำเป็นต้องทราบ รวมถึงทฤษฎีการหากฎความสัมพันธ์ของข้อมูล รวมไปถึงอัลกอริทึมที่ใช้ในการหาเซตรายการความถี่จากฐานข้อมูลขนาดใหญ่ด้วย

2.1 นิยามพื้นฐานที่เกี่ยวข้อง

ในการหาเซตรายการความถี่และกฎความสัมพันธ์ของข้อมูล จะมีคำจำกัดความเบื้องต้นและงานวิจัยที่เกี่ยวข้องดังต่อไปนี้

นิยามที่ 2.1 *กำหนดให้* $I = \{i_1, i_2, \ldots, i_k\}$ *เป็นเซตของตัวอักษรหรือตัวเลข (literals) และเรียกแต่ละสมาชิกของ I ว่า รายการ (item)*

นิยามที่ 2.2 *กำหนดให้* $D = \{t_1, t_2, \ldots, t_n\}$ *เป็นฐานข้อมูลซึ่งประกอบด้วยเซตของรายการเปลี่ยนแปลง (transaction) และแต่ละรายการเปลี่ยนแปลง* t_i *ประกอบด้วยตัวระบุรายการเปลี่ยนแปลง (Tid) โดยที่เซตของรายการ* $t_i \subseteq I$

นิยามที่ 2.3 *เซตของรายการ (itemset) หมายถึง เซตที่มีสมาชิกเป็น รายการที่ปรากฏในเซตของตัวอักษรหรือตัวเลข (literals) และเซตของรายการที่มีจำนวนสมาชิก K จำนวนจะถูกเรียกว่าเป็น K-itemset*

นิยามที่ 2.4 ค่าสนับสนุน (support) *ของเซตของรายการ (S) หมายถึง อัตราการปรากฏของรายการเปลี่ยนแปลง* (t_i) *ที่รายการทั้งหมดในเซตของรายการ S ปรากฏอยู่ในรายการเปลี่ยนแปลงเดียวกัน เทียบกับจำนวนรายการเปลี่ยนแปลงทั้งหมดในฐานข้อมูล (D ค่าสนับสนุนนี้สามารถคำนวณได้จาก*

$$\text{ค่าสนับสนุน} = \frac{\text{จำนวนรายการเปลี่ยนแปลงที่ประกอบด้วยเซตของรายการ } S}{\text{จำนวนรายการเปลี่ยนแปลงทั้งหมดในฐานข้อมูล}}$$

เพื่อความเข้าใจในการกำหนดค่าพารามิเตอร์ต่างๆ เหล่านี้ ตัวอย่างที่ 2.1 ต่อไปนี้จะแสดงให้เห็นถึงความสัมพันธ์ของ ฐานข้อมูล รายการ เซตของรายการ และการคำนวณค่าสนับสนุนของเซตของรายการใดๆ

ตัวอย่างที่ 2.1 กำหนดให้ฐานข้อมูลหนึ่งภายใต้เซตของตัวอักษร I = { 1, 2, 3, 4 } ในฐานข้อมูลนี้มีรายการเปลี่ยนแปลงหกรายการ ดังต่อไปนี้

รายการเปลี่ยนแปลง	รายการ
t_1	1, 2, 3, 4
t_2	1, 2, 3
t_3	2, 3, 4
t_4	2, 4
t_5	1, 3
t_6	1, 2, 3

ถ้ากำหนดให้ เซตของรายการ S = { 1, 3 } ค่าสนับสนุนของเซตของรายการ S จะมีค่าเท่ากับ 2/3 ทั้งนี้เนื่องจาก มีจำนวนเซตของรายการเปลี่ยนแปลงที่มีรายการ 1 และ 3 ปรากฏร่วมกันคือ 4 เมื่อเทียบกับจำนวนเซตของรายการเปลี่ยนแปลงทั้งหมด 6 เซต จะได้เป็น 4/6 นั่นเอง

นิยามที่ 2.5 ค่าสนับสนุนขั้นต่ำ (minimum support) *เป็นค่าขีดแบ่ง (threshold) ที่ใช้วัดค่าสนับสนุนของเซตของรายการใดๆ ซึ่งเป็นค่าที่ผู้ใช้ (users) กำหนด (นิยมระบุเป็นร้อยละ)*

การกำหนดค่าสนับสนุนขั้นต่ำจะถูกกำหนดขึ้น เพื่อใช้เป็นข้อกำหนดในการพิจารณาเซตของรายการว่ามีความสำคัญต่อการหาความสัมพันธ์ของรายการต่างๆ หรือไม่ เช่น กำหนดค่าสนับสนุนขั้นต่ำ 80% ของสินค้าที่สนใจจากรายการเปลี่ยนแปลงการซื้อสินค้าของลูกค้า 100 คน นั่นคือ ชนิดของสินค้าที่พิจารณาจะต้องปรากฏอยู่ในรายการสินค้าที่ถูกซื้อจากลูกค้าอย่างน้อย 80 คน เป็นต้น โดยเซตของสินค้าที่สนใจจะถูกนิยามดังต่อไปนี้

นิยามที่ 2.6 เซตรายการความถี่ (frequent itemset) *หมายถึง เซตของเซตของรายการที่สมาชิก (เซตของรายการแต่ละตัว) มีค่าสนับสนุนมากกว่าหรือเท่ากับค่าสนับสนุนขั้นต่ำ ในกรณีที่ เซตรายการความถี่มีจำนวนสมาชิกเท่ากับ K จำนวน เซตนี้จะถูกเรียกว่าเป็น* K-frequent itemset

นิยามที่ 2.7 Candidate K-itemset (C_k) *หมายถึงเซตของ K-itemset ที่มีโอกาสจะเป็นเซตรายการความถี่ ซึ่งเป็นเซตของรายการที่ยังไม่ได้ทำการนับค่าสนับสนุน และยังไม่ได้ตรวจสอบว่าเป็นเซตของรายการความถี่หรือไม่*

ในการสร้างกฎความสัมพันธ์ของรายการต่างๆ ที่เกิดขึ้นในฐานข้อมูลทั้งหมดนั้น จะมีวิธีการพิจาณาและสร้างมาจาก เซตรายการความถี่ที่หาได้จากฐานข้อมูลนั้นๆ โดยการกำหนดค่าสนับสนุนขั้นต่ำขึ้นเพื่อตัดข้อมูลที่ไม่มีนัยสำคัญออกไปจากการคำนวณ ซึ่งผู้ใช้เป็นผู้กำหนดว่าต้องการค่าสนับสนุนขั้นต่ำเท่าใด เมื่อสามารถคำนวณหาเซตของความถี่แล้ว ประเด็นสำคัญอีก

ประการหนึ่งก็คือ การนำเซตของความถี่นี้ไปคำนวณหา กฎความสัมพันธ์ของข้อมูล ดังจะได้กล่าวในลำดับต่อไป

2.2. ทฤษฎีการหากฎความสัมพันธ์ของข้อมูล

ในการวิเคราะห์การหาความสัมพันธ์ของข้อมูลนั้น สามารถหาความสัมพันธ์ของข้อมูลได้ด้วยกฎความสัมพันธ์ของข้อมูล ซึ่งจะเขียนอยู่ในรูปความสัมพันธ์จากเซตของรายการหนึ่งไปยังเซตของรายการอีกเซตหนึ่งดังบทนิยามต่อไปนี้

นิยามที่ 2.8 *กำหนดให้ I เป็นเซตของตัวอักษรหรือตัวเลขที่แสดงรายการทั้งหมดในฐานข้อมูล D และกำหนดให้ X และ Y เป็นเซตของรายการ นั้นคือ* $X \subseteq I$ *และ* $Y \subseteq I$ *และ* $X \cap Y = \varnothing$ กฎความสัมพันธ์ของข้อมูลจาก *X* ไปยัง *Y* จะหมายถึง *ความสัมพันธ์ของการปรากฏของรายการทั้งหมดใน X ที่มีผลต่อการปรากฏของรายการทั้งหมดใน Y ในรายการเปลี่ยนแปลงเดียวกันในฐานข้อมูล D*

ในการสร้างกฎความสัมพันธ์ จะต้องทำการหาเซตรายการความถี่ที่มีค่าสนับสนุนมากกว่าหรือเท่ากับค่าสนับสนุนขั้นต่ำที่ผู้ใช้กำหนดก่อน จากนั้น นำแต่ละเซตรายการความถี่ที่หาได้ที่มีขนาดตั้งแต่ 2 รายการขึ้นไป มาทำการสร้างกฎความสัมพันธ์ของข้อมูล แต่ในการสร้างกฎแต่ละกฎจะมีความน่าเชื่อถือมากหรือน้อยเพียงใดนั้นจะต้องมีค่าขีดแบ่งที่ใช้วัดค่าความเชื่อมั่นของกฎ โดยค่าความเชื่อมั่นของกฎความสัมพันธ์นั้นสามารถนิยามได้ดังต่อไปนี้

นิยามที่ 2.9 ค่าความเชื่อมั่น (confidence) *ของกฎความสัมพันธ์จาก X ไปยัง Y หมายถึง อัตราค่าสนับสนุนของการเกิดรายการใน X และ Y พร้อมกัน เทียบกับค่าสนับสนุนของ X ดังคำนวณได้จาก*

$$\text{ค่าความเชื่อมั่น } (X \rightarrow Y) = \frac{\text{ค่าสนับสนุนของ } X \cup Y}{\text{ค่าสนับสนุนของ } X}$$

นิยามที่ 2.10 ค่าความเชื่อมั่นขั้นต่ำ (minimum confidence) *เป็นค่าขีดแบ่งที่ใช้ในการวัดความเชื่อมั่นของกฎ ที่ผู้ใช้เป็นผู้กำหนด (นิยมระบุเป็นร้อยละ)*

เพื่อให้กฎที่หาได้มีความถูกต้องสมบูรณ์พร้อมนำไปใช้ในงานต่างๆ ได้ ค่าความเชื่อมั่นของกฎนั้นๆ จะต้องมีค่ามากกว่าหรือเท่ากับค่าความเชื่อมั่นขั้นต่ำซึ่งทั้งค่าสนับสนุนและค่าความเชื่อมั่นของกฎสามารถเขียนอยู่ในรูป

$$X \rightarrow Y\,[support = S,\ confidence = C]$$

นั่นหมายถึง กฎความสัมพันธ์ระหว่างเซตของรายการ X กับ เซตของรายการ Y มีค่าสนับสนุนเท่ากับ S (เขียนแทนด้วยค่าสนับสนุน $support(X \cup Y)$) และมีค่าความเชื่อมั่นเท่ากับ C (เขียนแทนด้วย $confidence(X \rightarrow Y)$) โดยค่าสนับสนุนและค่าความเชื่อมั่นของกฎสามารถหาได้จาก

$$support\ (X \cup Y) = \frac{\text{จำนวนรายการเปลี่ยนแปลงที่ประกอบด้วยเซตของรายการ } X \text{ และ } Y}{\text{จำนวนรายการเปลี่ยนแปลงทั้งหมดในฐานข้อมูล}}$$

$$confidence\ (X \rightarrow Y) = \frac{support\ (X \cup Y)}{support\ (X)}$$

การสร้างกฎความสัมพันธ์จะพิจารณาจากเซตรายการความถี่ทั้งหมดที่สามารถหาได้ โดยจะนำแต่ละเซตรายการความถี่มาแยกสับเซต แล้วจึงแยกออกเป็นกฎต่างๆ ดังตัวอย่างต่อไปนี้

ตัวอย่างที่ 2.2 กำหนดให้เซตของรายการความถี่ {1, 2, 3} การพิจารณากฎความสัมพันธ์จะสามารถแยกกฎออกได้ทั้งหมดดังต่อไปนี้

Antecedent		Consequence
{1, 2}	→	{3}
{1, 3}	→	{2}
{2, 3}	→	{1}
{1}	→	{2, 3}
{2}	→	{1, 3}
{3}	→	{1, 2}

จากกฎข้างต้นสามารถสังเกตุได้ว่า รายการทางด้านซ้าย (antecedent) เมื่อนำมารวมกับรายการทางด้านขวา (consequent) ต้องเท่ากับเซตรายการความถี่ที่พิจารณา โดยขนาดของรายการทางด้านซ้ายและด้านขวาสามารถเพิ่มหรือลดได้ แต่เมื่อจำนวนข้อมูลของด้านใดด้านหนึ่งเพิ่ม จำนวนข้อมูลอีกด้านหนึ่งจะต้องลดลง และจำนวนรายการทางด้านซ้ายและด้านขวาจะต้องไม่เท่ากับ 0 จากนั้น นำกฎที่แยกรายการออกเป็นรายการทางด้านซ้ายและด้านขวาแล้ว มาหาค่าความเชื่อมั่นของกฎ แล้วจึงนำมาเปรียบเทียบกับค่าความเชื่อมั่นขั้นต่ำ หากค่าความเชื่อมั่นของกฎมีค่ามากกว่าหรือเท่ากับค่าความเชื่อมั่นขั้นต่ำจะสามารถสรุปได้ว่า กฎนั้นสามารถยอมรับได้

2.3 การหาเซตรายการความถี่จากฐานข้อมูลขนาดใหญ่

การสร้างกฎความสัมพันธ์ถูกคิดค้นครั้งแรกโดยนักวิจัยจากศูนย์วิจัย IBM Almaden ประเทศสหรัฐอเมริกาในปี 1993 [1] โดยวิเคราะห์หากฎความสัมพันธ์ของข้อมูลจากฐานข้อมูลขนาดใหญ่ ต่อมา Agrawal R. และ Srikant R. ได้เสนออัลกอริทึมในการหาเซตรายการความถี่ได้อย่างมีประสิทธิภาพ เรียกว่า อะพริออริอัลกอริทึม (apriori algorithm) [2] อะพริออริเป็นอัลกอริทึมพื้นฐานสำหรับผู้ที่จะศึกษา เรียนรู้ทางด้านเทคนิคการสร้างกฎความสัมพันธ์ของข้อมูล และเพื่อศึกษาให้เข้าใจในส่วนของการหาเซตรายการความถี่ เพราะอัลกอริทึมนี้สามารถเรียนรู้เข้าใจได้ง่าย และไม่ซับซ้อน

ขั้นตอนการหาเซตรายการความถี่ของอะพริออริอัลกอริทึม เริ่มจากการหาเซตรายการความถี่ระดับที่ 1 (L_1) ที่มีค่าสนับสนุนมากกว่าหรือเท่ากับค่าสนับสนุนขั้นต่ำ โดยทำการอ่านข้อมูลจากฐานข้อมูลเพื่อนับค่าสนับสนุนของเซตของรายการ นำเซตของรายการกับค่าสนับสนุนที่นับไว้มาเปรียบเทียบกับค่าสนับสนุนขั้นต่ำ หากค่าสนับสนุนของเซตของรายการใดมีค่ามากกว่าหรือเท่ากับค่าสนับสนุนขั้นต่ำ ให้เก็บเซตของรายการนั้นไว้ในเซตรายการความถี่ระดับที่ 1 จากนั้น ทำการรวมเซตของรายการจากเซตของรายการที่เป็นสมาชิกของเซตรายการความถี่ระดับที่ 1(หาแคนดิเดทไอเท็มเซตระดับที่ 2) โดยการนำเซตของรายการ 2 เซต มารวมกันเป็นคู่ๆ โดยยึดถือลำดับของเซตของรายการเป็นสำคัญ เช่น เซตรายการความถี่ในระดับที่ 1 = {A, B, C} ทำการรวมเซตของรายการแล้วจะได้แคนดิเดทไอเท็มเซตระดับที่ 2 เป็น {AB, AC, BC} (ไม่ได้ BA เพราะการนำเซตของรายการมารวมกันจะคำนึงถึงลำดับเป็นสิ่งสำคัญ เพราะฉะนั้น BA จะมีความหมายเหมือนกับ AB อะพิออริอัลกอริทึมจึงเลือกใช้เฉพาะ AB เท่านั้น) เมื่อทำการหาแคนดิเดทไอเท็มเซตระดับที่ 2 (C_2) แล้วทำการนับค่าสนับสนุนให้กับแคนดิเดทไอเท็มเซตระดับที่ 2 โดยอ่านข้อมูลจากฐานข้อมูล ขั้นตอนต่อไปทำการรวมเซตของรายการแต่ละเซตรายการความถี่ที่อยู่ในเซตรายการความถี่ระดับที่ 2 เพื่อหาแคนดิเดทไอเท็มเซตระดับที่ 3 (C_3) แต่ในการหาแคนดิเดทไอเท็มเซตระดับที่ 3 มีอยู่ 2 สิ่งที่เพิ่มมาจากการหาแคนดิเดทไอเท็มเซตระดับที่ 2 คือ การตรวจสอบความเหมือนกันของเซตของรายการ กล่าวคือ เซตของรายการที่นำมารวมกันนั้นจะต้องมีความเหมือนกันตั้งแต่รายการลำดับแรกจนถึงรายการลำดับรองสุดท้ายถึงจะรวมกันได้ เช่น L_2 = {(A,B),(A,C),(B,C),(C,D)} หากต้องการนำเซตของรายการ (A,B) มารวมกับ (A,C) ต้องตรวจสอบว่า รายการแรกของเซตของรายการมีความเหมือนกันหรือไม่ ถ้าเหมือนกันจะสามารถรวมกันได้ ในกรณีนี้สามารถรวมกันได้เพราะรายการลำดับแรกของเซตของรายการ (A,B) และ (A,C) เหมือนกันคือรายการ A ดังนั้น (A, B) และ (A, C) สามารถรวมกันได้เป็น (A,B,C) และ สิ่งที่สองที่เพิ่มเข้ามา คือ การตัดเล็ม (prune) แคนดิเดทไอเท็มเซตเพื่อหาแคนดิเดทไอเท็มเซตที่แท้จริง

(จากนิยาม downward closure property) หมายถึง การหาสับเซตจากแคนดิเดทไอเท็มเซต(c) เป็นสมาชิกของเซตของแคนดิเดทไอเท็มเซต (C) (สับเซตที่ทำการหาจะต้องมีขนาดเท่ากับขนาดของแคนดิเดทไอเท็มเซตลบหนึ่ง) จากนั้นนำทุกสับเซตที่ได้มาทำการตรวจสอบกับเซตรายการความถี่ที่มีสมาชิกเท่ากัน(จากตัวอย่างคือเซตรายการความถี่ระดับที่ 2) ถ้าสับเซตใดไม่เป็นสมาชิกของเซตรายการความถี่ที่มีสมาชิกเท่ากัน ให้ลบเซตของรายการนั้นๆ ออกจากแคนดิเดทไอเท็มเซต จากนั้นทำการอ่านข้อมูลจากฐานข้อมูลเพื่อนับค่าสนับสนุนของเซตของรายการที่เป็นสมาชิกของแคนดิเดทไอเท็มเซต แล้วนำมาเปรียบเทียบกับค่าสนับสนุนขั้นต่ำเพื่อทำการหาเซตรายการความถี่ ในการหาแคนดิเดทไอเท็มเซตและเซตรายการความถี่ระดับที่ 4 และระดับถัดไปจะมีขั้นตอนการทำงานเหมือนกับระดับที่ 3 โดยอะพิออริอัลกอริทึมจะทำการหาแคนดิเดทไอเท็มเซตและเซตรายการความถี่ จนกระทั่งไม่สามารถหาเซตรายการความถี่หรือแคนดิเดทไอเท็มเซตได้ ซึ่งขั้นตอนการทำงานโดยละเอียดของอะพริออริอัลกอริทึมแสดงได้ ดังนี้

อัลกอริทึมการคำนวณการหาเซตรายการความถี่จากอะพริออริอัลกอริทึม

Input : Transaction database *DB*, minimum support threshold ξ.
Output : Frequent itemsets
Method : Call Apriori (*DB*,ξ)

Procedure Apriori (*DB*, ξ)
begin
(1) **for** each transaction t_i in *DB* **do**
(2) **for** each item a_{ij} in t_i **do**
(3) support (a_{ij}) $\leftarrow$ support (a_{ij}) + 1
(4) **end**
(5) **end**
(6) $L_1 \leftarrow$ for all item $a_j \in I$ such that support (a_j) $\geq \xi$
(7) **for** k $\leftarrow$ 2 to $L_{k-1} \neq \varnothing$ **do**
(8) $C_k \leftarrow$Candidate-gen (L_{k-1})
(9) **for** each transaction t_i in *DB* **do**
(10) C_t = subset (C_k, t_i)
(11) **for** all c $\in C_t$ **do**
(12) c.count++
(13) **end**
(14) **end**
(15) $L_k \leftarrow \{c \in C_k \mid c.count \geq \xi\}$
(16) **end**
end

แคนดิเดท-เจน (candidate-gen) เป็นฟังก์ชันที่ใช้ในการหาแคนดิเดทไอเท็มเซตที่มีสมาชิก K ตัว (C_k) จากเซตรายการความถี่ระดับที่ (K-1) โดยที่ขั้นตอนการทำงานของแคนดิเดท-เจน เริ่มจากการรวมเซตของรายการ โดยนำเซตของรายการที่เป็นสมาชิกของ L_{k-1} 2 เซตมาเปรียบเทียบกันว่ามีสมาชิกเหมือนกันตั้งแต่ลำดับที่ 1 จนถึงลำดับที่ K-2 หรือไม่ หากเหมือนกันให้นำเซตของรายการแรกมารวมกับรายการลำดับสุดท้ายในเซตของรายการที่สอง จะได้แคนดิเดทไอเท็มเซตที่มีสมาชิก K ตัว ต่อมานำทุกๆเซตของรายการที่เป็นสมาชิกของ L_{k-1} มารวมกัน จะได้เซตของแคนดิเดทไอเท็มเซต C_k จากนั้นทำการตัดเล็มแคนดิเดทไอเท็มเซต โดยการหาสับเซตขนาด K-1 ของแต่ละ $c \in C_k$ แล้วนำแต่ละสับเซตที่หาได้มาเปรียบเทียบกับเซตของรายการที่เป็นสมาชิกของ L_{k-1} ถ้าสับเซตตัวใดของ c ไม่เป็นสมาชิกของ L_{k-1} จะทำการตัด c ออกจากC_k รายละเอียดการทำงานของแคนดิเดท-เจนมีดังนี้

อัลกอริทึมการหาแคนดิเดทไอเท็มเซต

Input : L_{k-1}
Output : C_k set of candidate k-itemsets
Method : Call Candidate-gen (L_{k-1})

Procedure Candidate-gen (L_{k-1})
begin
(1) **insert** into C_k
(2) **select** $p.item_1$, $p.item_2$, …, $p.item_{k-2}$, $q.item_{k-2}$
(3) **from** L_{k-1} p, L_{k-1} q
(4) **where** $p.item_1 = q.item_1$, $p.item_2 = q.item_2$, …, $p.item_{k-2} = q.item_{k-2}$, $p.item_{k-1} < q.item_{k-1}$
(5) **for** each $c \in C_k$ **do**
(6) **for** all (k-1)-subset s of c **do**
(7) **if** $s \notin L_{k-1}$ **then**
(8) **delete** c from C_k
(9) **end**
(10) **end**
end

ตัวอย่างที่ 2.3 กำหนดให้ฐานข้อมูลประกอบด้วย 10 รายการเปลี่ยนแปลง ดังตารางที่ 2.1 และกำหนดค่าสนับสนุนขั้นต่ำ (minimum support) เท่ากับ 2 (20 %) สามารถหาเซตรายการความถี่จากอะพริออริอัลกอริทึมได้ดังนี้

ตารางที่ 2.1 ฐานข้อมูลแบบรายการเปลี่ยนแปลง

TID	Items
1	1, 2, 3
2	2, 3, 5
3	1, 2, 3, 5, 6
4	1, 2, 3, 4
5	1, 2, 3, 5
6	1, 2, 3, 5, 6
7	2, 3, 4, 5, 6
8	1, 2, 3
9	1, 3, 4, 5
10	2, 5, 6

การหาเซตรายการความถี่จากอะพริออริอัลกอริทึม เริ่มจากการอ่านข้อมูลจากฐานข้อมูลแล้วหาเซตรายการความถี่ระดับที่ 1 (L_1) ที่มีค่าสนับสนุนมากกว่าหรือเท่ากับค่าสนับสนุนขั้นต่ำที่กำหนดจะได้ {(1:7), (2:9), (3:9), (4:3), (5:7), (6:4)} (เซตของรายการ:ค่าสนับสนุน) นำสมาชิกในเซตรายการความถี่ระดับที่ 1 มาทำการหาแคนดิเดทไอเท็มเซตระดับที่ 2 ได้ {(1,2), (1,3), (1,4), (1,5), (1,6), (2,3), (2,4), (2,5), (2,6), (3,4), (3,5), (3,6), (4,5), (4,6), (5,6)} จากนั้นอ่านข้อมูลจากฐานข้อมูลอีกครั้ง เพื่อนับค่าสนับสนุนให้กับแคนดิเดทไอเท็มเซตระดับที่ 2 แล้วนำไปเปรียบเทียบกับค่าสนับสนุนขั้นต่ำ หากสมาชิกใดในแคนดิเดทไอเท็มเซตมีค่าสนับสนุนมากกว่าหรือเท่ากับค่าสนับสนุนขั้นต่ำจะเก็บไว้ใน L_2 ได้เป็น {(1,2 :6), (1,3 :7), (1,4 :2), (1,5 :4), (1,6 :2), (2,3 :8), (2,4 : 2), (2,5 :6), (2,6:4), (3,4 :3), (3,5 :6), (3,6 :3), (4,5 :2), (5,6 :4)} จากนั้นทำการหาแคนดิเดทไอเท็มเซตระดับที่ 3 โดยการนำแต่ละเซตของรายการที่เป็นสมาชิกของเซตรายการความถี่ระดับที่ 2 มารวมกัน หลังจากนั้นทำการตัดเล็มแคนดิเดทไอเท็มเซตระดับที่ 3 โดยนำแต่ละเซตของรายการที่เป็นสมาชิกของแคนดิเดทไอเท็มเซตระดับที่ 3 มาทำการหาสับเซตขนาด 2 แล้วนำแต่ละสับเซตมาตรวจสอบกับเซตรายการความถี่ระดับที่ 2 ถ้าสับเซตตัวใดไม่เป็นสมาชิกของเซตรายการความถี่ระดับที่ 2 ให้ตัดเซตของรายการนั้นออกจากแคนดิเดทไอเท็มเซตระดับที่ 3 เช่น นำเซตของรายการ (1,2) มารวมกับเซตของรายการ (1,3) ได้เป็นเซตของรายการ (1,2,3) จากนั้นหาสับเซตจากเซตของรายการ (1,2,3) ได้ {(1,2), (1,3), (2,3)} นำแต่ละสับเซตเปรียบเทียบกับเซตรายการความถี่ระดับที่ 2 ปรากฏว่าทุกสับเซตของเซตของรายการ (1,2,3) เป็น

สมาชิกของเซตรายการความถี่ระดับที่ 2 ดังนั้นเซตของรายการ (1,2,3) เป็นสมาชิกของแคนดิเดทไอเท็มเซตระดับที่ 3 เมื่อทำการรวมทุกเซตรายการความถี่จะได้ C_3 = {(1,2,3), (1,2,4), (1,2,5), (1,2,6), (1,3,4), (1,3,5), (1,3,6), (1,4,5), (1,5,6), (2,3,4), (2,3,5), (2,3,6), (2,4,5), (2,5,6), (3,4,5), (3,5,6)} ขั้นตอนต่อไปทำการอ่านข้อมูลจากฐานข้อมูลอีกครั้ง เพื่อหาเซตรายการความถี่ระดับที่ 3 จะได้ L_3 = {(1,2,3 :6), (1,2,5 :3), (1,2,6 :2), (1,3,4 :2), (1,3,5 :4), (1,3,6 :2), (1,5,6 :2), (2,3,4 :2), (2,3,5 :5), (2,3,6 :3), (2,5,6 :4), (3,4,5 :2), (3,5,6 :3)} จากนั้นทำการหาแคนดิเดทไอเท็มเซตระดับถัดไป คือ C_4 = {(1,2,3,5), (1,2,3,6), (1,2,5,6), (2,3,5,6)} และหา L_4 = {(1,2,3,5 :3), (1,2,3,6: 2), (1,2,5,6 :2), (2,3,5,6 :3)} ทำการหาแคนดิเดทไอเท็มเซตระดับที่ 5 คือ C_5 = {(1,2,3,4,5)} และอ่านข้อมูลจากฐานข้อมูลเพื่อหาเซตรายการความถี่ระดับที่ 5 = {(1,2,3,5,6 :2)} ทำการหาแคนดิเดทไอเท็มเซตระดับที่ 6 แต่ไม่สามารถทำการหาได้เนื่องจากเซตรายการความถี่ระดับที่ 5 มีสมาชิกเพียง 1 ตัวเท่านั้น จึงจบการทำงานของอะพริออริอัลกอริทึม ซึ่งผลลัพธ์ที่ได้มีดังนี้

L_1 = {(1:7), (2:9), (3:9), (4:3), (5:7), (6:4)}

L_2 = {(1, 2:6), (1, 3:7), (1, 4:2), (1, 5:4), (1, 6:2), (2, 3:8), (2, 4: 2), (2, 5:6), (2, 6:4), (3, 4:3), (3, 5:6), (3, 6:3), (4, 5:2), (5, 6:4)}

L_3 = {(1, 2, 3:6), (1, 2, 5:3), (1, 2, 6:2), (1, 3, 4:2), (1, 3, 5:4), (1, 3, 6:2), (1, 5, 6:2), (2, 3, 4:2), (2, 3, 5:5), (2, 3, 6:3), (2, 5, 6:4), (3, 4, 5:2), (3, 5, 6:3)}

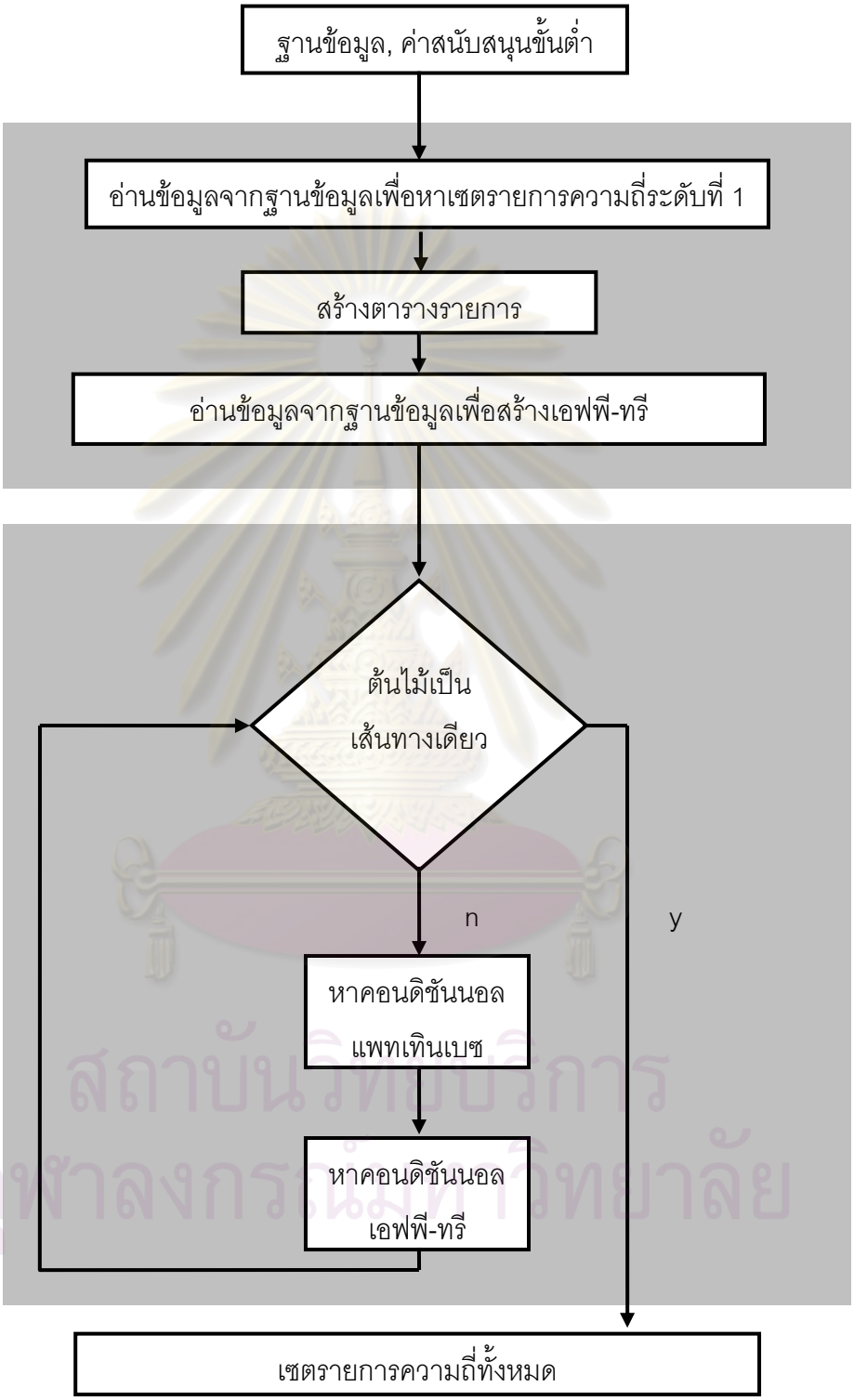
L_4 = {(1, 2, 3, 5:3), (1, 2, 3, 6:2), (1, 2, 5, 6:2), (2, 3, 5, 6:3)}

L_5 = {(1, 2, 3, 5, 6:2)}

ต่อมาในปี 1995 A. Savasere, E. Omiecinski และ S. Navathe ได้พัฒนาอัลกอริทึมเพื่อลดเวลาในการคำนวณและลดภาระในการเข้าถึงข้อมูล (I/O times) โดยพัฒนามาจากอะพริออริอัลกอริทึม โดยมีชื่อว่า พาทิฌันอัลกอริทึม (partition algorithm) [7] และนอกจากนี้ยังมีการพัฒนาอัลกอริทึมอื่นอีกมากมาย เช่น ไดนามิกไอเท็มเซตเคานทิง (dynamic itemset counting, DIC) [4] แพแร็ลแล็ลอัลกอริทึม (parallel algorithm) [9] การใช้โครงสร้างข้อมูลทไร (trie) [3] ไบนารีทไร (bnary tries) [8] เป็นต้น

จากนั้นในปี 2000 J. Han ได้พัฒนาอัลกอริทึมใหม่ในการหาเซตรายการความถี่ คือ เอฟพี-กโรธอัลกอริทึม (frequent pattern growth, FP-growth) [5] เพื่อลดเวลาในการคำนวณโดยไม่มีการหาแคนดิเดทไอเท็มเซตทุกระดับ และทำการอ่านข้อมูลจากฐานข้อมูลเพียง 2 ครั้ง โดยที่เอฟพี-กโรธใช้โครงสร้างข้อมูลที่มีชื่อว่า เอฟพี-ทรี (frequent pattern tree, FP-tree) ซึ่งเป็น

โครงสร้างต้นไม้ที่ใช้เก็บข้อมูลที่เป็นข้อมูลก่อนหน้าของรายการในรายการเปลี่ยนแปลง โดยเฉพาะรายการที่มีค่าสนับสนุนมากกว่าหรือเท่ากับค่าสนับสนุนขั้นต่ำ เท่านั้น



รูปที่ 2.1 ขั้นตอนการทำงานของเอฟพี-กโรธอัลกอริทึม

ขั้นตอนการทำงานของเอฟพี-กโรธอัลกอริทึมประกอบด้วย 2 ขั้นตอน คือ การสร้างเอฟพี-ทรีจากฐานข้อมูล และการหาเซตรายการความถี่จากเอฟพี-ทรีที่สร้างขึ้น ดังรูปที่ 2.1 โดยขั้นตอนการสร้างเอฟพี-ทรีเริ่มจากการอ่านข้อมูลจากฐานข้อมูลเพื่อหาเซตรายการความถี่ระดับที่ 1 ที่มีค่าสนับสนุนมากกว่าหรือเท่ากับค่าสนับสนุนขั้นต่ำ เรียงลำดับรายการเหล่านั้นตามค่าสนับสนุนจากมากไปน้อยแล้วเก็บไว้ในตารางรายการ (header table) ขั้นตอนต่อไปอ่านข้อมูลจากฐานข้อมูลอีกครั้งเพื่อตรวจสอบทุกรายการในรายการเปลี่ยนแปลง หากรายการใดในรายการเปลี่ยนแปลงไม่เป็นสมาชิกของรายการในตารางรายการให้ลบรายการนั้นออกจากรายการเปลี่ยนแปลง เมื่อทำการตรวจสอบทุกรายการแล้ว นำรายการในรายการเปลี่ยนแปลงมาเรียงลำดับตามลำดับของรายการในตารางรายการ แล้วจึงนำรายการเปลี่ยนแปลงนั้นไปสร้างเอฟพี-ทรี ขั้นตอนการสร้างเอฟพี-ทรีโดยละเอียดแสดงได้ ดังนี้

อัลกอริทึมการสร้างเอฟพี-ทรี

Input : Transaction database *DB*, minimum support threshold ξ.
Output : FP-Tree
Method : Call Create_FP-Tree (*DB*, ξ)

Procedure Create_FP-Tree (*DB*, ξ)
begin

(1) root of FP-Tree ← *null*
(2) **for** each transaction t_i in *DB* **do**
(3) **for** each item a_{ij} in t_i **do**
(4) support (a_{ij}) ← support (a_{ij}) + 1
(5) **end**
(6) **end**
(7) L_1 ← for all item $a_j \in I$ such that support (a_j) ≥ ξ
(8) Header Table ← Sort a_j in L_1 by support (a_j)
(9) **for** each t_i in *DB* **do**
(10) remove all a_{ij} such that support (a_{ij}) < ξ
(11) t_i←Sort support $a_{ij} \in t_i$ according to the order of Header table
(12) current node ← root of FP-Tree
(13) **for** j ← 1 to $|t_i|$ **do**
(14) **if** a_{ij} is a child of current node **then**
(15) add 1 to support of that node
(16) **else**
(17) child (current node) ← create node with a_{ij} and connect node with Header table
(18) support (child (current node)) ← 1
(19) **endif**
(20) current node ← child (current node)
(21) **end**
(22) **end**
end

ขั้นตอนการทำงานหลังจากสร้างเอฟพี-ทรีจากฐานข้อมูล คือ การหาเซตรายการความถี่จากเอฟพี-ทรี เริ่มจากการพิจารณารายการลำดับสุดท้ายเป็นลำดับแรก จากนั้นทำการหาคอนดิชันนอลแพทเทินเบซของรายการ (a_i) ในตารางรายการจากเอฟพี-ทรี ซึ่งผลลัพธ์ที่ได้จะเป็นเซตของเส้นทางในเอฟพี-ทรี โดยที่แต่ละเส้นทางจะประกอบด้วยเส้นทางก่อนหน้าของรายการ a_i และทุกรายการย่อยในเส้นทางก่อนหน้าจะมีค่าสนับสนุนเท่ากับค่าสนับสนุนของ a_i จากนั้นนำแต่ละคอนดิชันนอลแพทเทินเบซที่ได้มาสร้างเอฟพี-ทรีอีกหนึ่งต้น เรียกว่า คอนดิชันนอลเอฟพี-ทรี (conditional FP-Tree) หากต้นไม้ที่สร้างขึ้นมีเส้นทางไม่เกิน 1 เส้นทาง (single path) หรือไม่สามารถสร้างคอนดิชันนอลเอฟพี-ทรีได้ จะทำการนำรายการในต้นไม้รวมกับรายการที่พิจารณาแล้วสร้างเป็นเซตรายการความถี่ ขั้นตอนการหาเซตรายการความถี่แสดงได้ดังนี้

กำหนดให้ α คือ รายการในตารางรายการ และ B คือ คอนดิชันนอลแพทเทินเบซของ α โดยที่ β คือ รายการใน B

อัลกอริทึมการหาเซตรายการความถี่จากเอฟพี-ทรี

Input : FP-Tree, minimum support threshold ξ
Output : The complete set of frequent itemsets
Method : Call FP-frequent (*Tree, null*)

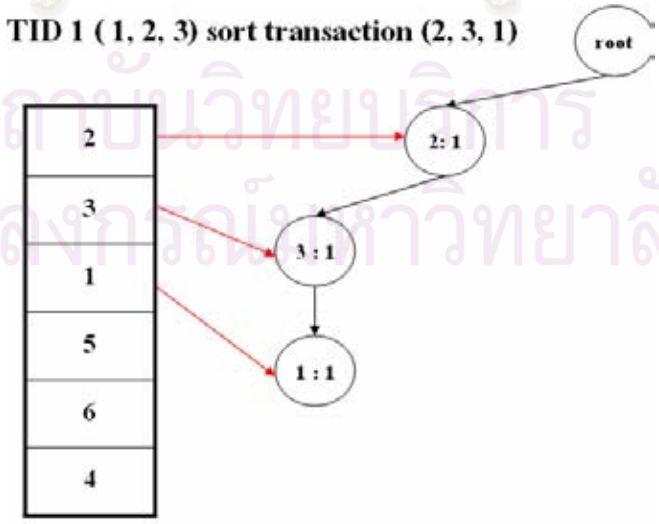
Procedure FP-frequent (*Tree*, α)
begin
(1) **if** *Tree* contain a single Path **then**
(2) **for** each combination β of the nodes in the Path **do**
(3) generate frequent itemset from $\beta \cup \alpha$ with support (frequent itemset) = support (node in β)
(4) **end**
(5) **else**
(6) **for** each α_i in the Header table **do**
(7) generate set of B by traverse Tree such that support (for all $a_i \in B$) = support ($\alpha \in B$)
(8) construct conditional FP-Tree $Tree_\beta$ from β's conditional pattern base
(9) **if** $Tree_\beta \neq \varnothing$ **then**
(10) **call** FP-frequent ($Tree_\beta$)
(11) **endif**
(12) **end**
(13) **endif**
end

ตัวอย่างที่ 2.4 กำหนดให้ฐานข้อมูลประกอบด้วยข้อมูล 10 รายการเปลี่ยนแปลง ดังตารางที่ 2.1 และกำหนดให้ค่าสนับสนุนขั้นต่ำเท่ากับ 2 (20 %) สามารถหาเซตรายการความถี่จากเอฟพี-กโรธอัลกอริทึมได้ดังนี้

การหาเซตรายการความถี่จากเอฟพี-กโรธอัลกอริทึม เริ่มจากอ่านข้อมูลจากฐานข้อมูลเพื่อหาเซตรายการความถี่ระดับที่ 1 ที่มีค่าสนับสนุนมากกว่าหรือเท่ากับค่าสนับสนุนขั้นต่ำที่กำหนดไว้ได้ {(1:7), (2:9), (3:9), (4:3), (5:7), (6:4)} นำสมาชิกในเซตรายการความถี่ระดับที่ 1 มาทำการเรียงลำดับตามค่าสนับสนุนได้ L_1 = {(2:9), (3:9), (1:7), (5:7), (6:4), (4:3)} จากนั้นนำทุกสมาชิกในเซตรายการความถี่ระดับที่ 1 มาสร้างตารางรายการดังรูปที่ 2.2

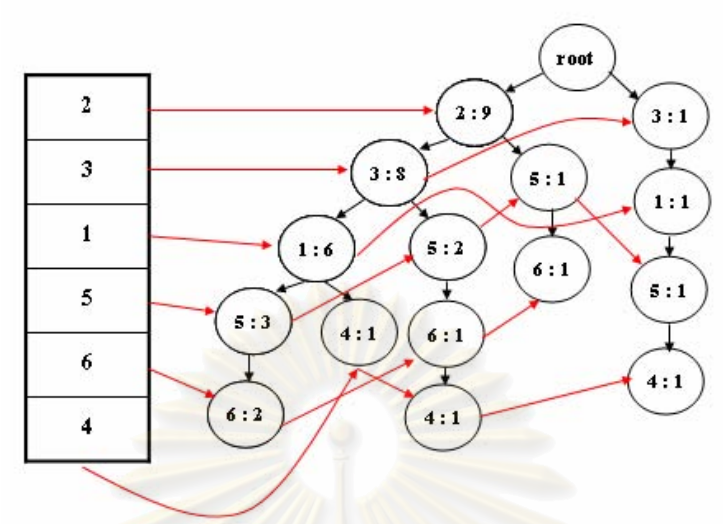
รูปที่ 2.2 ตารางรายการ

ขั้นตอนต่อไปทำการอ่านข้อมูลจากฐานข้อมูลทีละรายการเปลี่ยนแปลง เพื่อตรวจสอบทุกรายการในรายการเปลี่ยนแปลง หากรายการใดในรายการเปลี่ยนแปลงไม่เป็นสมาชิกของตารางรายการจะลบรายการนั้นๆออกจากรายการเปลี่ยนแปลง จากนั้นนำทุกรายการในรายการเปลี่ยนแปลงมาเรียงลำดับตามรายการในตารางรายการ แล้วจึงนำรายการที่เรียงลำดับแล้วไปสร้างเอฟพี-ทรี เช่น รายการเปลี่ยนแปลงที่ 1 = {1, 2, 3} เรียงลำดับแล้วได้เป็น {2, 3, 1} แสดงดังรูปที่ 2.3



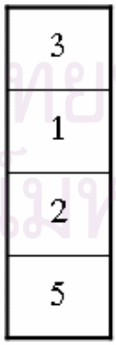
รูปที่ 2.3 การสร้างเอฟพี-ทรี

ทำการอ่านข้อมูลทุกรายการเปลี่ยนแปลงในฐานข้อมูลเพื่อนำมาสร้างเอฟพี-ทรีได้ ดังรูปที่ 2.4



รูปที่ 2.4 เอฟพี-ทรีที่สร้างจากฐานข้อมูล

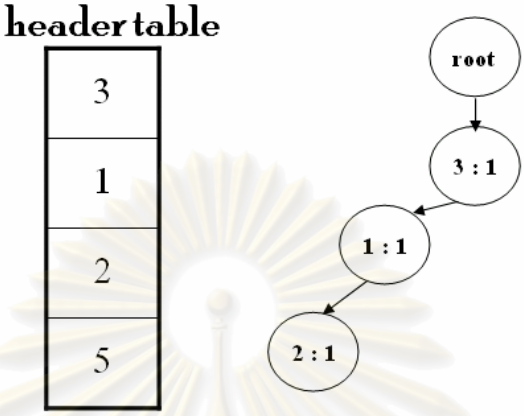
ขั้นตอนต่อไปทำการหาเซตรายการความถี่ เริ่มจากการพิจารณารายการลำดับสุดท้ายในตารางรายการ คือ รายการ 4 อ่านคอนดิชันนอลแพทเทินเบซจากเอฟพี-ทรีได้เป็น {(2, 3, 1:1), (2, 3, 5, 6:1), (3, 1, 5:1)} ทำการนับค่าสนับสนุนของทุกรายการในคอนดิชันนอลแพทเทินเบซเพื่อเปรียบเทียบกับค่าสนับสนุนขั้นต่ำ โดยที่รายการใดที่มีค่าสนับสนุนน้อยกว่าค่าสนับสนุนขั้นต่ำจะไม่นำมาพิจารณาและลบรายการนั้นออกจากคอนดิชันนอลแพทเทินเบซ ส่วนรายการที่มีค่าสนับสนุนมากกว่าหรือเท่ากับค่าสนับสนุนขั้นต่ำ จะนำมาเรียงลำดับรายการตามค่าสนับสนุนได้เป็น {(3:3), (1:2), (2:2), (5:2)} จากนั้นนำรายการที่ทำการเรียงลำดับแล้วมาสร้างเป็นตารางรายการของคอนดิชันนอลเอฟพี-ทรีดังรูปที่ 2.5



รูปที่ 2.5 ตารางรายการของคอนดิชันนอลเอฟพี-ทรี

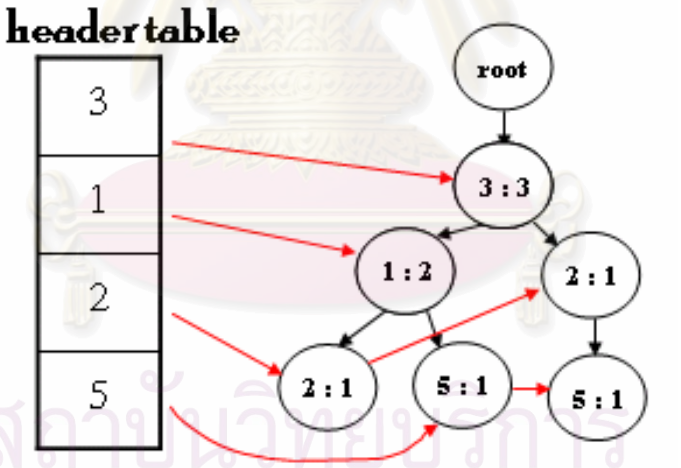
จากนั้นทำการอ่านข้อมูลทีละคอนดิชันนอลแพทเทินเบซ พิจารณาทุกๆรายการในคอนดิชันนอลแพทเทินเบซ ถ้ารายการใดไม่เป็นสมาชิกของตารางรายการจะลบรายการนั้นๆออกจาก

คอนดิชันนอลแพทเทินเบซ ขั้นตอนต่อไปเรียงลำดับรายการในคอนดิชันนอลแพทเทินเบซโดยเรียงลำดับตามรายการในตารางรายการ จากนั้นนำคอนดิชันนอลแพทเทินเบซที่เรียงลำดับแล้วสร้างเป็นคอนดิชันนอลเอฟพี-ทรีดังรูปที่ 2.6



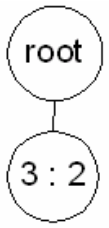
รูปที่ 2.6 คอนดิชันนอลเอฟพี-ทรีของรายการ 4

ทำการอ่านทุกคอนดิชันนอลแพทเทินเบซเพื่อนำไปสร้างคอนดิชันนอลเอฟพี-ทรี ดังรูปที่ 2.7



รูปที่ 2.7 คอนดิชันนอลเอฟพี-ทรีของรายการ 4

ขั้นตอนต่อไปทำการหาเซตรายการความถี่ โดยพิจารณารายการ 5 ในตารางรายการรวมกับรายการ 4 ที่พิจารณาก่อนหน้า แล้วทำการอ่านคอนดิชันนอลแพทเทินเบซของรายการ (5, 4) คือ {(3, 1:1), (3, 2:1)} จากนั้นนำคอนดิชันนอลแพทเทินเบซมานับค่าสนับสนุนเพื่อเปรียบเทียบกับค่าสนับสนุนขั้นต่ำ และนำคอนดิชันนอลแพทเทินเบซมาสร้างเป็นคอนดิชันนอลเอฟพี-ทรี ดังรูปที่ 2.8



รูปที่ 2.8 คอนดิชันนอลเอฟพี-ทรีของรายการ (5, 4)

เมื่อต้นไม้เป็นเส้นทางเดียว (single path) นำรายการ 3 มารวมกับเซตของรายการ (5,4) ที่พิจารณาก่อนหน้า จะได้เซตของรายการ (3,4,5) เป็นเซตรายการความถี่ซึ่งมีค่าสนับสนุนเท่ากับ 2 ต่อมาพิจารณารายการลำดับถัดไปในตารางรายการของคอนดิชันนอลเอฟพี-ทรี คือ รายการ 2 ทำวิธีการเดียวกันกับรายการ 5 และทำทุกรายการในตารางของคอนดิชันนอลเอฟพี-ทรี ได้เซตรายการความถี่ ดังนี้

{(4 :3), (5,4 :2), (3,5,4 :2), (2,4 :2), (3,2,4 :2), (1,4 :2), (3,1,4 :2), (3,4 :3)}

เมื่อพิจารณารายการลำดับสุดท้ายในตารางรายการของเอฟพี-ทรีแล้วจะพิจารณารายการลำดับถัดไปในตารางรายการของเอฟพี-ทรี คือ รายการ 6 หาคอนดิชันนอลแพทเทินเบซและสร้างคอนดิชันนอลเอฟพี-ทรีเช่นเดียวกับรายการ 4 เพื่อหาเซตรายการความถี่ เมื่อทำครบทุกรายการในตารางรายการแล้วเป็นอันจบการทำงานของเอฟพี-กโรธอัลกอริทึม ซึ่งผลลัพธ์แสดงดังตารางที่ 2.2

ตารางที่ 2.2 เซตรายการความถี่ที่ได้จากเอฟพี-กโรธอัลกอริทึม

Items	Frequent Itemsets (itemsets: support)
4	{ (4 :3), (4,5 :2), (1,4: 2), (2,4 :2), (3,4 :3), (3,4,5 :2), (1,3,4 :2), (2,3,4 :2) }
6	{ (6 :4), (1,6 :2), (3,6 :3), (5,6 :4), (2,6 :4), (1,3,6 :2), (1,2,3,6 :2), (1356:2), (1,2,3,5,6 :2), (1,5,6 :2), (1,2,5,6 :2), (1,2,6 :2), (3,5,6 :3), (2,3,5,6 :3), (2,3,6 :3), (2,5,6 :4) }
5	{ (5 :7), (1,5 :4), (2,5 :6), (3,5 :6), (1,2,5 :3), (1,3,5 :4), (2,3,5 :5), (1,2,3,5 :3) }
1	{ (1 :7), (1,2 :6), (1,3 :7), (1,2,3 :6) }
3	{ (3 :9), (2,3 :8) }
2	{ (2 :9) }

ต่อมาในปี 2001 J. Pei ได้พัฒนาโครงสร้างข้อมูลใหม่สำหรับการหาเซตรายการความถี่ คือ เอช-ซทรัค(H-struct) [6] และพัฒนาอัลกอริทึมสำหรับการหาเซตรายการความถี่จากเอช-ซทรัค ที่มีชื่อว่า เอช-ไมน (H-mine) ซึ่งข้อดีของ เอช-ไมน คือ การอ่านข้อมูลจากฐานข้อมูลเพียงครั้งเดียว ซึ่งการทำงานอื่นๆจะทำในหน่วยความจำ การทำงานเริ่มจากการอ่านข้อมูลจากฐานข้อมูลเพื่อหาเซตรายการความถี่ระดับที่ 1 และนำรายการเปลี่ยนแปลงที่มีเฉพาะรายการที่เป็นสมาชิกของเซตรายการความถี่ระดับที่ 1 มาเก็บไว้ในหน่วยความจำ จากนั้นทำการโพรเจคชัน (projection) แต่ละรายการเปลี่ยนแปลงเพื่อหาเซตรายการความถี่ที่เป็นเซตของคำตอบ จุดเด่นของอัลกอริทึม เอช-ไมน คือ สามารถคาดเดาการใช้หน่วยความจำได้ ไม่ต้องหาแคนดิเดทไอเท็มเซต และใช้เวลาในการคำนวณน้อย

บทที่ 3

การหาเซตรายการความถี่จากต้นไม้แสดงรายการความถี่

ในบทนี้จะกล่าวถึงการหาเซตรายการความถี่จากต้นไม้แสดงรายการความถี่ (frequent item tree, FI-tree) โดยประยุกต์การทำงานของเอฟพี-กโรธอัลกอริทึม (FP-growth) ในส่วนของการสร้างเอฟพี-ทรี (FP-tree) และ ประยุกต์การหาเซตรายการความถี่โดยใช้โครงสร้างข้อมูลทไร (trie) [3] ในส่วนของการหาสับเซต ซึ่งต้นไม้แสดงรายการความถี่จะมีโครงสร้างข้อมูลเหมือนกับเอฟพี-ทรีทุกประการ โดยที่การหาเซตรายการความถี่จากต้นไม้แสดงรายการความถี่นั้นแบ่งออกเป็น 2 ขั้นตอน คือ การสร้างต้นไม้แสดงรายการความถี่จากฐานข้อมูล และการหาเซตรายการความถี่จากต้นไม้แสดงรายการความถี่ที่สร้างขึ้น โดยมีรายละเอียดขั้นตอนการทำงาน ดังรูปที่ 3.1

รูปที่ 3.1 ขั้นตอนการทำงานของต้นไม้แสดงรายการความถี่

3.1 การสร้างต้นไม้แสดงรายการความถี่จากฐานข้อมูล

ในส่วนนี้จะกล่าวถึงการสร้างต้นไม้แสดงรายการความถี่จากฐานข้อมูล เริ่มจากการอ่านข้อมูลจากฐานข้อมูลทีละรายการเปลี่ยนแปลงเพื่อนับค่าสนับสนุนให้กับทุกรายการในรายการเปลี่ยนแปลง เมื่ออ่านข้อมูลครบทุกรายการเปลี่ยนแปลงแล้วจะได้เซตรายการความถี่ระดับที่ 1 ที่มีค่าสนับสนุนมากกว่าหรือเท่ากับค่าสนับสนุนขั้นต่ำ จากนั้นนำทุกรายการที่เป็นสมาชิกของเซตรายการความถี่ระดับที่ 1 มาสร้างเป็นตารางรายการ ขั้นตอนต่อไปทำการอ่านข้อมูลจากฐานข้อมูลอีกครั้งโดยอ่านข้อมูลทีละรายการเปลี่ยนแปลง ทำการตรวจสอบทุกรายการในรายการเปลี่ยนแปลง หากรายการใดในรายการเปลี่ยนแปลงเป็นรายการที่ไม่เป็นสมาชิกของตารางรายการจะทำการลบรายการนั้นออกจากรายการเปลี่ยนแปลง เมื่อทำการตรวจสอบครบทุกรายการแล้ว นำรายการในรายการเปลี่ยนแปลงนั้นไปสร้างต้นไม้แสดงรายการความถี่ ซึ่งขั้นตอนการสร้างต้นไม้แสดงรายการความถี่โดยละเอียดมีดังนี้

อัลกอริทึมการสร้างต้นไม้แสดงรายการความถี่

Input : Transaction database *DB*, minimum support thresholdξ.
Output : Frequent item tree
Method : Call Create Tree (*DB*,ξ)

Procedure Create Tree (*DB*,ξ)
begin
(1) root of Frequent item tree ← *null*
(2) **for** each transaction t_i in *DB* **do**
(3) **for** each item a_{ij} in t_i **do**
(4) support (a_{ij}) ←support (a_{ij}) +1
(5) **end**
(6) **end**
(7) Header Table←for all item $a_j \in$I such that support (a_j) $\geq \xi$
(8) **for** each t_i in *DB* **do**
(9) remove all a_{ij} in t_i such that support (a_{ij}) < ξ
(10) current node ← root of Frequent item tree
(11) **for** j ← 1 to $|t_i|$ **do**
(12) **if** a_{ij} is a child of current node **then**
(13) add 1 to support of that node
(14) **else**
(15) child (current node) ← create node with a_{ij} and connect node with Header table
(16) support (child (current node)) ← 1
(17) **endif**
(18) current node ← child (current node)
(19) **end**
(20) **end**
end

เมื่อจบขั้นตอนการสร้างต้นไม้แสดงรายการความถี่จากฐานข้อมูล เราจะได้ต้นไม้ที่ประกอบด้วยรายการในแต่ละรายการเปลี่ยนแปลง ซึ่งรายการเหล่านั้นเป็นรายการที่มีค่าสนับสนุนมากกว่าหรือเท่ากับค่าสนับสนุนขั้นต่ำเท่านั้น โดยที่ในการสร้างต้นไม้แสดงรายการความถี่จะมีขั้นตอนการสร้างต้นไม้เหมือนกับการสร้างเอฟพี-ทรีเกือบทุกประการ แต่จะมีข้อแตกต่าง คือ ในการสร้างต้นไม้แสดงรายการความถี่จะลดขั้นตอนการเรียงลำดับรายการในตารางรายการ และ ลดขั้นตอนการเรียงลำดับรายการในทุกรายการเปลี่ยนแปลงเพื่อนำรายการเปลี่ยนแปลงนั้นๆ ไปสร้างต้นไม้แสดงรายการความถี่ ซึ่งการลดขั้นตอนทั้งสองจะสามารถช่วยลดเวลาในการคำนวณได้

3.2 การหาเซตรายการความถี่จากต้นไม้แสดงรายการความถี่

เมื่อทำการสร้างต้นไม้แสดงรายการความถี่จากฐานข้อมูลแล้ว ขั้นตอนการทำงานต่อไป คือ การหาเซตรายการความถี่จากต้นไม้แสดงรายการความถี่ที่สร้างขึ้น ซึ่งขั้นตอนดังกล่าวสามารถแบ่งออกได้เป็น 3 ขั้นตอนย่อย คือ การรวมค่าสนับสนุนจากการเชื่อมโยงของรายการในตารางรายการไปยังบัพในต้นไม้แสดงรายการความถี่ การหาสับเซตที่จำเป็นจากทุกเส้นทางของต้นไม้แสดงรายการความถี่ และ การตัดเล็มต้นไม้แสดงรายการความถี่ให้มีความถูกต้องสมบูรณ์ โดยที่รายละเอียดขั้นตอนการหาเซตรายการความถี่มีดังนี้

3.2.1 การรวมค่าสนับสนุน

ขั้นตอนแรกของการหาเซตรายการความถี่จากต้นไม้แสดงรายการความถี่ คือ การรวมค่าสนับสนุนให้กับต้นไม้แสดงรายการความถี่ ซึ่งขั้นตอนการทำงานเริ่มจากพิจารณาการเชื่อมโยงของรายการในตารางรายการลำดับรองสุดท้ายไปยังบัพในต้นไม้ โดยแต่ละเส้นทางที่พิจารณานั้นจะต้องมีจำนวนบัพไม่น้อยกว่าสองบัพ ถ้าเส้นทางใดที่มีบัพน้อยกว่าสองบัพจะไม่นำมาพิจารณา (เนื่องจากอัลกอริทึมได้ทำการหาเซตรายการความถี่ระดับที่ 1 ไว้แล้ว ถ้าพิจารณาเส้นทางที่มีจำนวนบัพน้อยกว่าสองบัพจะทำให้ค่าสนับสนุนของรายการในเซตรายการความถี่ระดับที่ 1 มีความผิดพลาดและทำให้สิ้นเปลืองเวลาในการคำนวณ) เมื่อเราค้นพบเส้นทางที่มีบัพมากกว่าสองบัพแล้วทำการนำรายการและค่าสนับสนุนของแต่ละบัพในเส้นทางที่พิจารณามาเพิ่มให้กับบัพในต้นไม้ เมื่อทำการเพิ่มค่าสนับสนุนจากเส้นทางที่พิจารณาแล้วจะทำการพิจารณาเส้นทางต่อไปจนครบทุกเส้นทางของการเชื่อมโยง จากนั้นจะพิจารณาการเชื่อมโยงของรายการลำดับถัดไปในตารางรายการจนกระทั่งถึงรายการลำดับที่สองจึงหยุดการทำงาน เมื่อการรวมค่าสนับสนุนจบการทำงานจะทำให้แต่ละบัพในต้นไม้แสดงรายการความถี่มีค่าสนับสนุนใกล้เคียงความถูกต้องมากขึ้น ขั้นตอนการทำงานโดยละเอียดของการรวมค่าสนับสนุนมีดังนี้

อัลกอริทึมการรวมค่าสนับสนุน

Input : Frequent item tree, Header table
Output : Frequent item tree with increased support
Method : Increase Support

Procedure Increase Support (Tree, Header table)
begin
(1) **for** i←| Header table|-1 to 2 **do**
(2) Path(a_i)←read path of a_i from link of Header table to the tree
(3) **for** each $Path_{ij} \in Path(a_i)$ **do**
(4) **if** $|Path_{ij}| > 1$ **then**
(5) current node = root of Frequent item tree
(6) **for** k←1 to $|Path_{ij}|$ **do**
(7) **if** a_{ijk} is a child of current node **then**
(8) support(child(current node)) ← support(child (current node))+1
(9) **else**
(10) child (current node)←create node with a_{ijk} and connect node with Header table
(11) support (child (current node))← support of a_{ijk}
(12) **endif**
(13) current node←child (current node)
(14) **end**
(15) **endif**
(16) **end**
(17) **end**
end

3.2.2 การหาสับเซตที่จำเป็น

ขั้นตอนที่สองของการหาเซตรายการความถี่จากต้นไม้แสดงรายการความถี่ คือ การหาสับเซตที่จำเป็นเพื่อให้แต่ละบัพในต้นไม้แสดงรายการความถี่มีค่าสนับสนุนที่ถูกต้อง เหตุผลที่ทำการหาสับเซตเพียงบางสับเซต (สับเซตที่จำเป็น) เพื่อลดเวลาในการคำนวณ และเพื่อให้ได้ผลลัพธ์ที่ถูกต้อง เพราะการหาสับเซตทั้งหมดจะใช้เวลาในการคำนวณมาก และทำให้เกิดการทำงานซ้ำซ้อน ขั้นตอนการหาสับเซตเริ่มจากอ่านเส้นทางจากต้นไม้แสดงรายการความถี่ โดยที่เส้นทางที่จะนำมาพิจารณาจะต้องมีจำนวนบัพอย่างน้อย 3 บัพ เนื่องจากหากเส้นทางมีจำนวนน้อยกว่า 3 บัพจะทำให้ไม่สามารถหาสับเซตได้(กรณีที่มีเพียงบัพเดียว) เกิดการซ้ำซ้อนของการทำงาน และ เกิดความผิดพลาดของค่าสนับสนุนของเซตรายการความถี่ระดับที่ 1 เช่น เส้นทางที่มีจำนวนบัพ 2 บัพ ทำการหาสับเซตจะได้สับเซตขนาด 1 บัพ เมื่อนำค่าสนับสนุนของสับเซตไปเพิ่มให้กับต้นไม้แสดง

รายการความถี่จะทำให้รายการในเซตรายการความถี่ระดับที่ 1 มีค่าสนับสนุนเพิ่มขึ้น ซึ่งมีค่าเกินความเป็นจริง โดยในการอ่านจะเริ่มจากการอ่านทุกๆเส้นทางที่มีรายการแรกเหมือนกันเก็บรวมกันไว้ จากนั้นนำเส้นทางเหล่านั้นมาหาสับเซตที่จำเป็นเท่านั้น ซึ่งสับเซตที่ต้องการหาที่ต้องทำการหาจะต้องมีคุณสมบัติพิเศษ 2 ข้อ คือ สับเซตจะต้องมีรายการแรกเหมือนกับเส้นทางที่ทำการหาสับเซตเท่านั้น และ สับเซตจะต้องไม่เป็นเส้นทางย่อยของเส้นทางเดิม ตัวอย่างเช่น เซตของรายการ (1, 2, 3, 5) ทำการหาสับเซตที่ต้องการคือ {(1, 2, 5), (1, 3, 5), (1, 5)} เหตุที่ (1, 2, 3) ไม่เป็นสับเซตที่ต้องการเนื่องจาก (1, 2, 3) เป็นเส้นทางย่อยของเส้นทาง (1, 2, 3, 5) เป็นต้น เมื่อได้ทุกสับเซตที่ต้องการแล้วให้ทำการเพิ่มค่าสนับสนุนของแต่ละสับเซตให้กับต้นไม้แสดงรายการความถี่ จากนั้นทำการหาสับเซตของทุกเส้นทางในต้นไม้และเพิ่มค่าสนับสนุนของทุกสับเซตให้กับต้นไม้แสดงรายการความถี่ เมื่อทำการหาสับเซตครบทุกเส้นทางจะได้ต้นไม้แสดงรายการความถี่ที่มีเซตรายการความถี่ที่มีค่าสนับสนุนถูกต้อง ขั้นตอนการหาสับเซตโดยละเอียดมีดังนี้

อัลกอริทึมการหาสับเซตที่จำเป็น

Input : Frequent item tree
Output : Frequent item tree which is correct result
Method : Find Subset

Procedure FindSubset (Frequent item tree)
begin
(1) **for** all path from root to any leave in Frequent item tree **do**
(2) each $path_j$ such that have 3 node at least find necessary subset of the path
(3) **for** each $subset_i \in$ subset **do**
(4) current node = root of Frequent item tree
(5) **for** k←1 to $|subset_j|$ **do**
(6) **if** a_{jk} is a child of current node **then**
(7) add support(a_{jk}) to support of that node
(8) **else**
(9) child(current node)←create node with a_{jk} and connect node with Header table
(10) support (child (current node)) ← support(a_{jk})
(11) **endif**
(12) current node←child (current node)
(13) **end**
(14) **end**
(15) **end**
end

3.2.3. การตัดเล็มต้นไม้แสดงรายการความถี่

ขั้นตอนสุดท้ายของการหาเซตรายการความถี่จากต้นไม้แสดงรายการความถี่ คือ การตัดเล็มต้นไม้แสดงรายการความถี่เพื่อให้ได้ผลลัพธ์ที่ถูกต้องครบถ้วน ซึ่งการตัดเล็มต้นไม้จะทำการค้นหาบัพในต้นไม้แสดงรายการความถี่ ถ้าบัพใดมีค่าสนับสนุนน้อยกว่าค่าสนับสนุนขั้นต่ำจะทำการลบบัพนั้นออกจากต้นไม้แสดงรายการความถี่ ขั้นตอนการตัดเล็มต้นไม้แสดงรายการความถี่มีดังนี้

อัลกอริทึมการตัดเล็มต้นไม้แสดงรายการความถี่

Input : Frequent item tree, minimum support threshold ξ.
Output : Complete Frequent itemsets in the Frequent item tree
Method : Prune Tree

Procedure Prune Tree (Frequent item tree,ξ)
begin
(1) **for** each node in Frequent item tree **do**
(2) **if** node (support) < ξ **then**
(3) delete node and child node out of Frequent item tree
(4) **endif**
(5) **end**
end

เมื่อดำเนินการครบทุกขั้นตอนของการหาเซตรายการความถี่จากต้นไม้แสดงรายการความถี่ จะได้ต้นไม้แสดงรายการความถี่ที่มีเซตรายการความถี่ทั้งหมดที่มีค่าสนับสนุนถูกต้อง สามารถนำเซตรายการความถี่เหล่านั้นไปสร้างเป็นกฎความสัมพันธ์ของข้อมูลเพื่อนำไปประกอบการตัดสินใจต่อไป

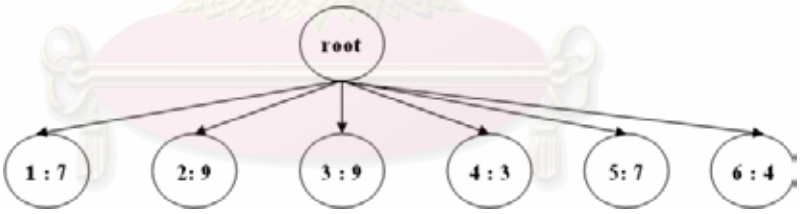
ในการหาเซตรายการความถี่จากต้นไม้แสดงรายการความถี่ ต้นไม้แสดงรายการความถี่ที่ทำการสร้างในขั้นตอน 3.1 จะมีขนาดใหญ่กว่าเอฟพี-ทรี เนื่องจากเอฟพี-ทรีต้องทำการเรียงลำดับข้อมูลในทุกรายการเปลี่ยนแปลง จึงเป็นเหตุให้ในแต่ละบัพของเอฟพี-ทรีจะมีการซ้ำซ้อนกันซึ่งสามารถใช้บัพเดียวกันได้ แต่เมื่อจำนวนรายการเปลี่ยนแปลงในฐานข้อมูลและจำนวนรายการเฉลี่ยในแต่ละรายการเปลี่ยนแปลงมีจำนวนมาก รวมถึงรายการเปลี่ยนแปลงในฐานข้อมูลมีการซ้ำกันมาก การสร้างเอฟพี-ทรีจะใช้เวลาในการคำนวณมาก เนื่องจากต้องทำการเรียงลำดับทุกรายการเปลี่ยนแปลง โดยที่ในการเรียงลำดับแต่ละรายการเปลี่ยนแปลงจะใช้เวลาในการคำนวณค่อนข้างมาก ซึ่งแตกต่างจากการสร้างต้นไม้แสดงรายการความถี่ที่ไม่ต้องการเรียงลำดับข้อมูล

ตัวอย่าง 3.1 กำหนดให้ฐานข้อมูลประกอบด้วยข้อมูล 10 รายการเปลี่ยนแปลง ดังตารางที่ 2.1 และ ค่าสนับสนุนขั้นต่ำมีค่าเท่ากับ 2 (20 %) สามารถหาเซตรายการความถี่โดยใช้ต้นไม้แสดงรายการความถี่ได้ดังนี้

ขั้นตอนการสร้างต้นไม้แสดงรายการความถี่ เริ่มจากอ่านข้อมูลจากฐานข้อมูลเพื่อหาเซตรายการความถี่ระดับที่ 1 ที่มีค่าสนับสนุนมากกว่าหรือเท่ากับค่าสนับสนุนขั้นต่ำที่กำหนดไว้ได้เป็น L_1 = {(1:7), (2:9), (3:9), (4:3), (5:7), (6:4)} จากนั้นนำสมาชิกในเซตรายการความถี่ระดับที่ 1 มาสร้างตารางรายการได้ดัง รูปที่ 3.2 และนำแต่ละรายการในเซตรายการความถี่ระดับที่ 1 มาสร้างต้นไม้แสดงรายการความถี่ดัง รูปที่ 3.3

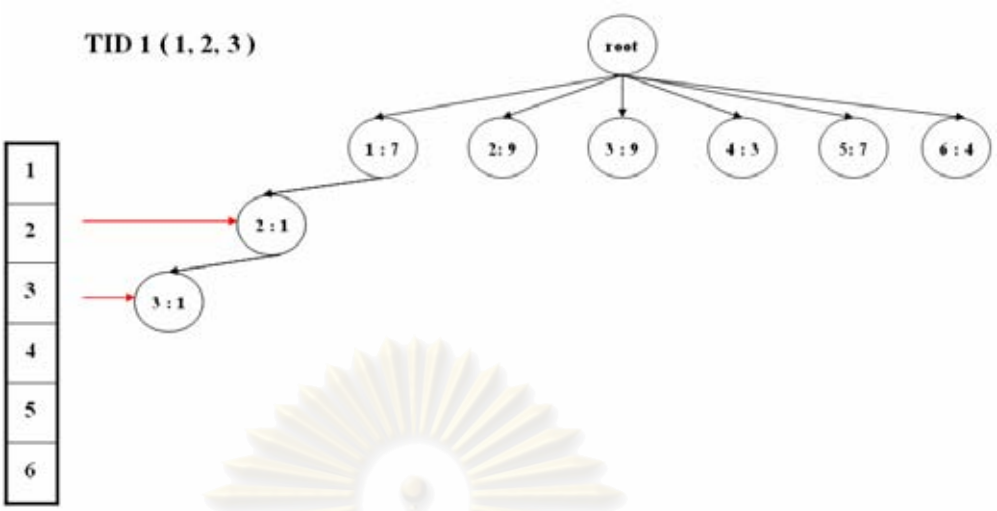
1
2
3
4
5
6

รูปที่ 3.2 ตารางรายการของต้นไม้แสดงรายการความถี่

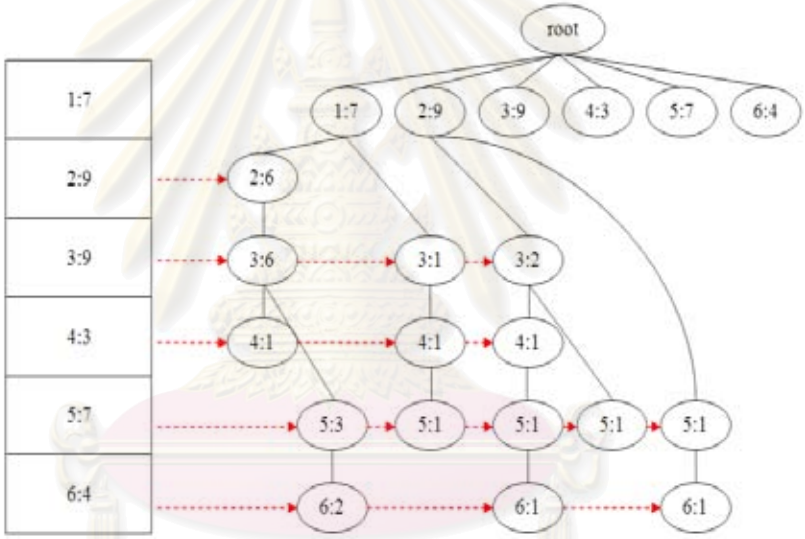


รูปที่ 3.3 การสร้างต้นไม้แสดงรายการความถี่จากเซตรายการความถี่ระดับที่ 1

ขั้นตอนต่อไปทำการอ่านข้อมูลจากฐานข้อมูลทีละรายการเปลี่ยนแปลงเพื่อตรวจสอบรายการในรายการเปลี่ยนแปลง หากรายการใดในรายการเปลี่ยนแปลงไม่เป็นสมาชิกของตารางรายการให้ทำการลบรายการนั้นออกจากรายการเปลี่ยนแปลง เมื่อทำการตรวจสอบทุกรายการในรายการเปลี่ยนแปลงแล้วจึงนำรายการในรายการเปลี่ยนแปลงไปสร้างต้นไม้แสดงรายการความถี่ เช่น รายการเปลี่ยนแปลงที่ 1 = {1, 2, 3} สร้างต้นไม้แสดงรายการความถี่ได้ดัง รูปที่ 3.4 และเมื่อทำการอ่านข้อมูลและตรวจสอบรายการครบทุกรายการเปลี่ยนแปลงจะได้ต้นไม้แสดงรายการความถี่ดังรูปที่ 3.5

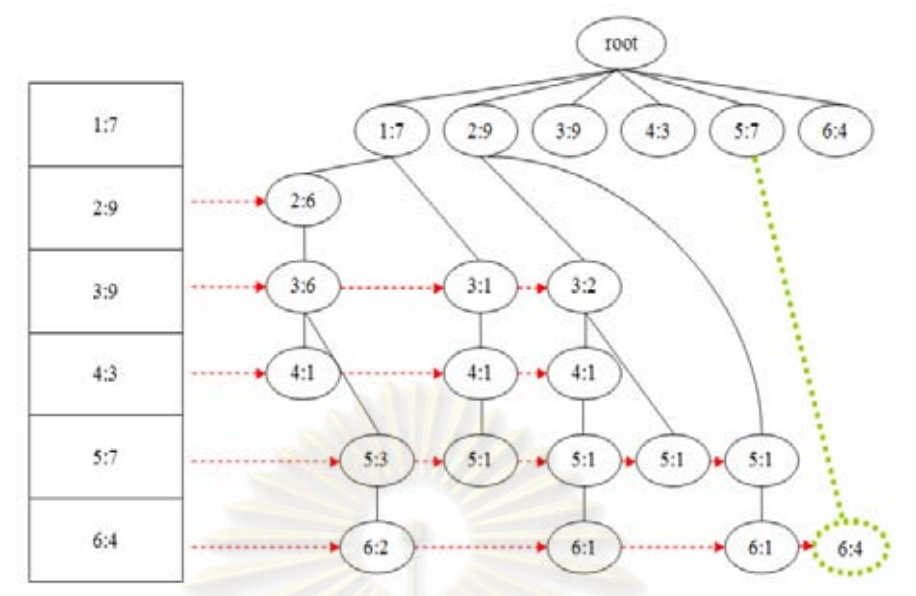


รูปที่ 3.4 ต้นไม้แสดงรายการความถี่จากรายการในรายการเปลี่ยนแปลงที่ 1



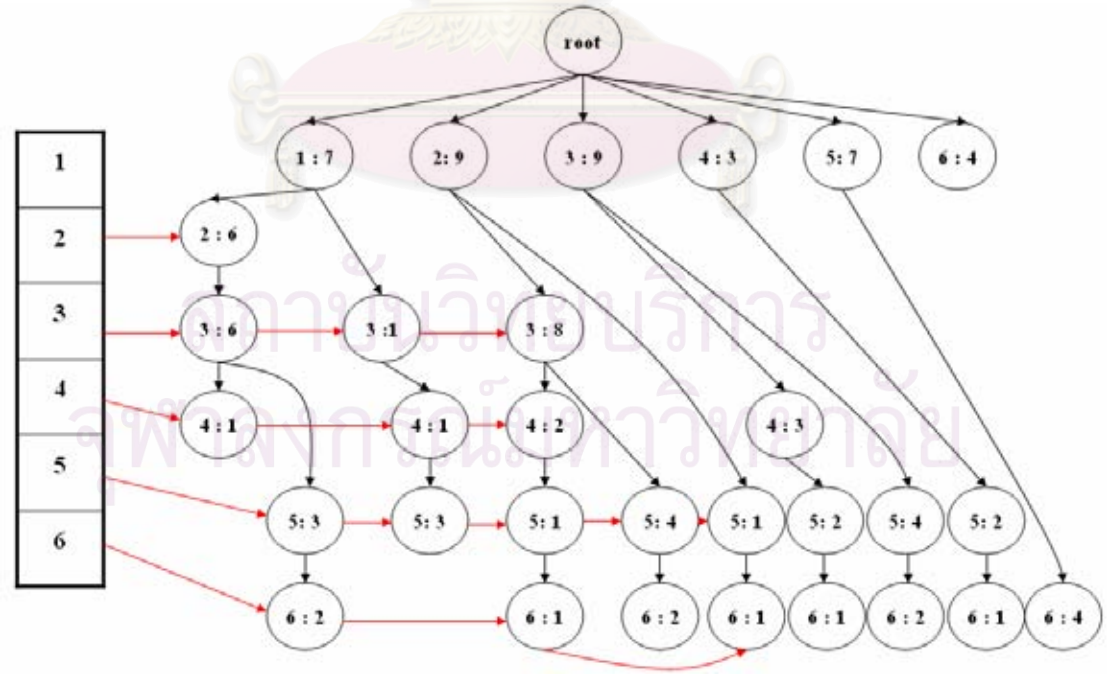
รูปที่ 3.5 ต้นไม้แสดงรายการความถี่ที่สร้างจากฐานข้อมูล

ขั้นตอนต่อไปทำการหาเซตรายการความถี่ เริ่มจากขั้นตอนการรวมค่าสนับสนุนซึ่งจะพิจารณาการเชื่อมโยงของรายการลำดับรองสุดท้ายในตารางรายการ (รายการ 5) ไปยังบัพในต้นไม้แสดงรายการความถี่ จากนั้นพิจารณาแต่ละเส้นทางของต้นไม้แสดงรายการความถี่ที่เชื่อมต่อกับตารางรายการ โดยเส้นทางที่จะนำมาพิจารณาจะต้องมีจำนวนบัพมากกว่าหรือเท่ากับ 2 บัพขึ้นไป เมื่อพิจารณาเส้นทางของรายการ 5 ได้เป็น {(5, 6:2), (5, 6:1), (5, 6:1)} นำค่าสนับสนุนของแต่ละบัพมาเพิ่มให้กับบัพในต้นไม้แสดงรายการความถี่ดังรูปที่ 3.6



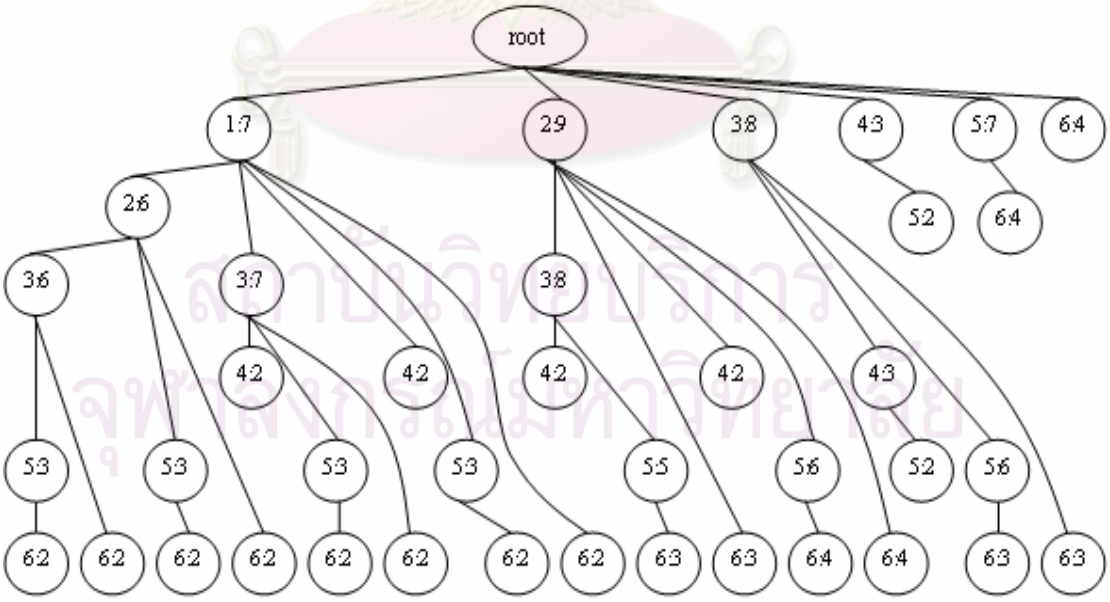
รูปที่ 3.6 การเพิ่มค่าสนับสนุนให้กับต้นไม้แสดงรายการความถี่

เมื่อพิจารณาการเชื่อมโยงของรายการ 5 แล้ว จะทำการพิจารณาการเชื่อมโยงของรายการลำดับถัดไปในตารางรายการ (รายการ 4) เพิ่มให้ค่าสนับสนุนให้กับต้นไม้แสดงรายการความถี่จนกระทั่งถึงรายการลำดับที่ 2 ในตารางรายการ (รายการ 2) จะได้ต้นไม้แสดงรายการความถี่ที่มีค่าสนับสนุนเพิ่มขึ้นดังรูปที่ 3.7



รูปที่ 3.7 ต้นไม้แสดงรายการเมื่อรวมค่าสนับสนุน

ขั้นตอนที่สองของการหาเซตรายการความถี่จากต้นไม้แสดงรายการความถี่ คือ การหาสับเซตที่จำเป็นของทุกเส้นทางในต้นไม้แสดงรายการความถี่ที่มีจำนวนบัพมากกว่าหรือเท่ากับ 3 บัพ การทำงานของการหาสับเซตที่จำเป็นเริ่มจากการพิจารณาเส้นทางของรายการลำดับที่ 3 จากข้างท้ายของตารางรายการ (รายการ 4) โดยจะเริ่มเก็บเส้นทางที่มีรายการแรกเหมือนกันรวมกันจนครบทุกเส้นทาง นำแต่ละเส้นทางไปทำการหาสับเซต เช่น รายการ 4 อ่านเส้นทางได้เป็น {4:3, 5:2, 6:1} ซึ่งมีเพียงเส้นทางเดียว เมื่ออ่านเส้นทางครบแล้วทำการหาสับเซต ซึ่งสับเซตที่เราต้องการหาจะต้องประกอบด้วยรายการแรกในเส้นทางแรกเสมอ และสับเซตที่ทำการหาจะต้องไม่เป็นเส้นทางย่อยของเส้นทางเดิม จากเส้นทางของรายการ 4 จะสามารถหาสับเซตได้เป็น {4:3,6:1} จากนั้นนำสับเซตที่หาได้มาเพิ่มค่าสนับสนุนให้กับต้นไม้แสดงรายการความถี่ แล้วจึงทำการพิจารณารายการลำดับถัดไปคือรายการ 3 อ่านเส้นทางของรายการ 3 ได้เป็น { (3:9, 4:3, 5:2, 6:1), (3:9, 5:4, 6:2)} จากนั้นทำการหาสับเซตของทุกเส้นทาง ซึ่งสับเซตของเส้นทาง (3:9, 4:3, 5:2, 6:1) คือ {(3:9, 4:3, 6:1), (3:9, 5:2, 6:1), (3:9, 6:1)} และสับเซตของเส้นทาง (3:9, 5:4, 6:2) คือ {(3:9, 6:2)} นำแต่ละสับเซตที่ได้มาเพิ่มค่าสนับสนุนให้กับต้นไม้แสดงรายการความถี่ จากนั้นพิจารณารายการลำดับถัดไปจนกระทั่งถึงรายการลำดับแรก เมื่อจบขั้นตอนการหาสับเซตแล้วจะเข้าสู่ขั้นตอนสุดท้าย คือ การตัดเล็มต้นไม้แสดงรายการความถี่ เมื่อทำการตัดเล็มต้นไม้เสร็จแล้วจะได้ต้นไม้แสดงรายการความถี่ที่บรรจุเซตรายการความถี่ทั้งหมดที่มีค่าสนับสนุนที่ถูกต้องครบถ้วนดังรูปที่ 3.8



รูปที่ 3.8 ต้นไม้แสดงรายการความถี่ที่ประกอบด้วยเซตรายการความถี่

จากรูปที่ 14 เซตรายการความถี่ทั้งหมดจะถูกบรรจุอยู่ในต้นไม้แสดงรายการความถี่ โดยที่การอ่านเซตรายการความถี่สามารถอ่านได้จากเส้นทางในต้นไม้แสดงรายการความถี่ เช่น เส้นทาง (1:7, 2:6, 3:6, 5:3, 6:2) มีความหมายว่า รายการ 1 เกิดขึ้นทั้งสิ้น 7 รายการเปลี่ยนแปลง รายการ 1 เกิดร่วมกับรายการ 2 ทั้งสิ้น 6 รายการเปลี่ยนแปลง รายการ 1 เกิดร่วมกับรายการ 2 และรายการ 3 ทั้งสิ้น 6 รายการเปลี่ยนแปลง รายการ 1 เกิดร่วมกับรายการ 2 รายการ 3 และรายการ 5 ทั้งสิ้น 3 รายการเปลี่ยนแปลง และ รายการ 1 เกิดร่วมกับรายการ 2 รายการ 3 รายการ 5 และรายการ 6 ทั้งสิ้น 2 รายการเปลี่ยนแปลง ซึ่งเซตรายการความถี่ที่ได้จากต้นไม้แสดงรายการความถี่จะมีความถูกต้องครบถ้วนเหมือนกับเซตรายการความถี่ที่ได้จากอัลกอริทึมอะพริออริ และเอฟพี-กโรธอัลกอริทึม โดยที่กฎความสัมพันธ์ของข้อมูลสามารถหาได้จากเซตรายการความถี่เหล่านั้น

บทที่ 4

การปรับปรุงการหาเซตรายการความถี่จากต้นไม้แสดงรายการความถี่

ในบทนี้จะกล่าวถึงการปรับปรุงการหาเซตรายการความถี่จากต้นไม้แสดงรายการความถี่ โดยขั้นตอนการทำงานจะเหมือนต้นไม้แสดงรายการความถี่ทุกประการ แต่จะมีส่วนที่ปรับปรุงจากเดิมเล็กน้อย เช่น การปรับปรุงขั้นตอนการสร้างต้นไม้แสดงรายการความถี่จะมีขั้นตอนซับซ้อนมากขึ้น การปรับปรุงการรวมค่าสนับสนุน และ การปรับปรุงขั้นตอนการหาสับเซตที่จำเป็น ซึ่งขั้นตอนดังกล่าวเป็นขั้นตอนการทำงานที่ใช้เวลาในการคำนวณมากที่สุด โดยในการปรับปรุงการหาสับเซตที่จำเป็นจะทำให้จำนวนสับเซตที่ต้องการหามีจำนวนน้อยลง แต่ยังคงให้เซตรายการความถี่ที่ถูกต้องครบถ้วน ซึ่งการปรับปรุงขั้นตอนดังกล่าวสามารถลดเวลาในการคำนวณได้ ขั้นตอนการทำงานโดยละเอียดมีดังนี้

4.1 การปรับปรุงการสร้างต้นไม้แสดงรายการความถี่จากฐานข้อมูล

ในส่วนนี้จะกล่าวถึงการปรับปรุงการสร้างต้นไม้แสดงรายการความถี่จากฐานข้อมูล เริ่มจากการอ่านข้อมูลจากฐานข้อมูลทีละรายการเปลี่ยนแปลงเพื่อนับค่าสนับสนุนให้กับทุกรายการในรายการเปลี่ยนแปลง เมื่ออ่านข้อมูลครบทุกรายการเปลี่ยนแปลงแล้วจะได้เซตรายการความถี่ระดับที่ 1 ที่มีค่าสนับสนุนมากกว่าหรือเท่ากับค่าสนับสนุนขั้นต่ำ จากนั้นนำทุกรายการที่เป็นสมาชิกของเซตรายการความถี่ระดับที่ 1 มาสร้างเป็นตารางรายการ ซึ่งในการปรับปรุงการสร้างต้นไม้แสดงรายการความถี่ ข้อมูลที่จัดเก็บในตารางรายการจะมีการเปลี่ยนแปลงเล็กน้อย ซึ่งจากเดิมตารางรายการจะเก็บข้อมูลเฉพาะชื่อรายการ และตัวชี้ (pointer) ที่ใช้สำหรับการเชื่อมโยงจากตารางรายการไปยังบัพในต้นไม้เท่านั้น แต่ในการปรับปรุงต้นไม้แสดงรายการความถี่ จะเพิ่มการจัดเก็บค่าสนับสนุนของรายการที่มีความสัมพันธ์กับรายการลำดับก่อนหน้าด้วย เหตุที่ต้องเก็บค่าสนับสนุนของรายการเพราะช่วยให้ไม่ต้องทำการหาสับเซตที่จำเป็นขนาด 2 และช่วยลดการใช้หน่วยความจำด้วย เมื่อสร้างตารางรายการแล้วทำการอ่านข้อมูลจากฐานข้อมูลอีกครั้งโดยอ่านข้อมูลทีละรายการเปลี่ยนแปลง ทำการตรวจสอบทุกรายการในรายการเปลี่ยนแปลง หากรายการใดในรายการเปลี่ยนแปลงเป็นรายการที่ไม่เป็นสมาชิกของตารางรายการจะทำการลบรายการนั้นออกจากรายการเปลี่ยนแปลง เมื่อทำการตรวจสอบครบทุกรายการแล้ว นำรายการในรายการเปลี่ยนแปลงนั้นไปสร้างต้นไม้แสดงรายการความถี่ และเพิ่มค่าสนับสนุนให้กับตารางรายการ ซึ่งขั้นตอนการปรับปรุงการสร้างต้นไม้แสดงรายการความถี่โดยละเอียดมีดังนี้

อัลกอริทึมการปรับปรุงการสร้างต้นไม้แสดงรายการความถี่

Input **:** Transaction database *DB*, minimum support thresholdξ.
Output **:** Frequent item tree
Method **:** Call Create Tree (*DB*,ξ)

Procedure Create Tree (*DB*,ξ)
begin
(1) root of Frequent item tree ← *null*
(2) **for** each transaction t_i in *DB* **do**
(3) **for** each item a_{ij} in t_i **do**
(4) support (a_{ij}) ←support (a_{ij}) +1
(5) **end**
(6) **end**
(7) Header Table←for all item $a_j \in$I such that support (a_j) ≥ ξ
(8) **for** each t_i in *DB* **do**
(9) remove all a_{ij} in t_i such that support (a_{ij}) < ξ
(10) current node ← root of Frequent item tree
(11) **for** j ← 1 to $|t_i|$ **do**
(12) **if** a_{ij} is a child of current node **then**
(13) add 1 to support of that node
(14) **else**
(15) child (current node) ← create node with a_{ij} and connect node with Header table
(16) support (child (current node)) ← 1
(17) **endif**
(18) increase support of a_{ij} in Header table
(19) current node ← child (current node)
(20) **end**
(21) **end**
end

เมื่อจบขั้นตอนการสร้างต้นไม้แสดงรายการความถี่จากฐานข้อมูล เราจะได้ต้นไม้ที่ประกอบด้วยรายการในแต่ละรายการเปลี่ยนแปลง และตารางรายการที่ประกอบด้วยค่าสนับสนุนของรายการ โดยที่ในการสร้างต้นไม้แสดงรายการความถี่จะมีขั้นตอนการสร้างต้นไม้เหมือนกับการสร้างต้นไม้แสดงรายการความถี่แบบเดิมเกือบทุกประการ แต่จะมีข้อแตกต่าง คือ การเพิ่มขั้นตอนการจัดเก็บค่าสนับสนุนของรายการในตารางรายการ

4.2 การปรับปรุงการหาเซตรายการความถี่จากต้นไม้แสดงรายการความถี่

การปรับปรุงการหาเซตรายการความถี่จากต้นไม้แสดงรายการความถี่ที่สร้างขึ้นประกอบด้วย 3 ขั้นตอนย่อย คือ การรวมค่าสนับสนุนจากการเชื่อมโยงของรายการในตารางรายการไปยังบัพในต้นไม้แสดงรายการความถี่ การหาสับเซตที่จำเป็นจากทุกเส้นทางของต้นไม้

แสดงรายการความถี่ และการตัดเล็มต้นไม้แสดงรายการความถี่ให้มีความถูกต้องสมบูรณ์ โดยที่รายละเอียดขั้นตอนการหาเซตรายการความถี่มีดังนี้

4.2.1 การรวมค่าสนับสนุน

ขั้นตอนแรกของการหาเซตรายการความถี่จากต้นไม้แสดงรายการความถี่ คือ การรวมค่าสนับสนุนให้กับต้นไม้แสดงรายการความถี่ ขั้นตอนการทำงานเริ่มจากพิจารณาการเชื่อมโยงของรายการในตารางรายการลำดับรองสุดท้ายไปยังบัพในต้นไม้ โดยแต่ละเส้นทางที่พิจารณาจะต้องมีจำนวนบัพไม่น้อยกว่าสองบัพ ถ้าเส้นทางใดที่มีบัพน้อยกว่าสองบัพจะไม่นำมาพิจารณา เมื่อเราค้นพบเส้นทางที่มีบัพมากกว่าสองบัพแล้วทำการนำรายการและค่าสนับสนุนของแต่ละบัพในเส้นทางที่พิจารณามาเพิ่มให้กับบัพในต้นไม้ และเพิ่มค่าสนับสนุนให้กับตารางรายการในตำแหน่งของรายการที่พิจารณา เมื่อทำการเพิ่มค่าสนับสนุนแล้วจะทำการพิจารณาเส้นทางต่อไปจนครบทุกเส้นทางของการเชื่อมโยง จากนั้นจะพิจารณาการเชื่อมโยงของรายการลำดับถัดไปในตารางรายการจนกระทั่งถึงรายการลำดับที่สองจึงหยุดการทำงาน ขั้นตอนการทำงานโดยละเอียดของการรวมค่าสนับสนุนมีดังนี้

อัลกอริทึมการรวมค่าสนับสนุน

Input : Frequent item tree, Header table
Output : Frequent item tree with increased support
Method : Increase Support

Procedure Increase Support (Tree, Header table)
begin
(1) **for** i←| Header table|-1 to 2 **do**
(2) Path(a_i)←read path of a_i from link of Header table to the tree
(3) **for** each $Path_{ij} \in Path(a_i)$ **do**
(4) **if** $|Path_{ij}| > 1$ **then**
(5) current node = root of Frequent item tree
(6) **for** k←1 to $|Path_{ij}|$ **do**
(7) **if** a_{ijk} is a child of current node **then**
(8) support(child(current node)) ← support(child (current node))+1
(9) **else**
(10) child (current node)←create node with a_{ijk} and connect node with Header table
(11) support (child (current node))← support of a_{ijk}
(12) **endif**
(13) increase support of a_{ijk} in Header table
(13) current node←child (current node)
(14) **end**

(15) **endif**
(16) **end**
(17) **end**
end

หลังจากขั้นตอนการรวมค่าสนับสนุน เราจะได้เซตรายการความถี่ระดับที่ 1 และ 2 โดยที่เซตรายการความถี่ระดับที่ 1 สามารถอ่านได้จากบัพในต้นไม้แสดงรายการความถี่ แลเซตรายการความถี่ระดับที่ 2 สามารถอ่านได้จากค่าสนับสนุนของรายการในตารางรายการ ซึ่งสามารถนำมาใช้งานได้โดยตรง

4.2.2 การหาสับเซตที่จำเป็น

ขั้นตอนที่สองของการหาเซตรายการความถี่จากต้นไม้แสดงรายการความถี่ คือ การหาสับเซตที่จำเป็นเพื่อให้แต่ละบัพในต้นไม้แสดงรายการความถี่มีค่าสนับสนุนที่ถูกต้อง ขั้นตอนการหาสับเซตเริ่มจากอ่านเส้นทางจากต้นไม้แสดงรายการความถี่ โดยที่เส้นทางที่จะนำมาพิจารณาจะต้องมีจำนวนบัพอย่างน้อย 4 บัพ เนื่องจากหากบัพในเส้นทางมีจำนวนน้อยกว่า 4 บัพ สามารถหาสับเซตได้เฉพาะขนาด 2 และ ขนาด 1 ซึ่งไม่มีความจำเป็น เนื่องจากเซตรายการความถี่ระดับที่ 1 ได้มาจากการอ่านข้อมูลจากฐานข้อมูล และเซตรายการความถี่ระดับที่ 3 สามารถอ่านได้จากค่าสนับสนุนของรายการในตารางรายการ หากเส้นทางมีจำนวนบัพน้อยกว่า 4 บัพ จะทำให้เกิดการซ้ำซ้อนของการทำงาน และ เกิดความผิดพลาดของค่าสนับสนุนของเซตรายการความถี่ระดับที่ 1 โดยในการอ่านเส้นทางจะเริ่มจากการอ่านทุกๆเส้นทางที่มีรายการแรกเหมือนกันเก็บรวมกันไว้ จากนั้นนำเส้นทางเหล่านั้นมาหาสับเซตที่จำเป็นเท่านั้น ซึ่งสับเซตที่ต้องการหาจะต้องมีคุณสมบัติพิเศษ 2 ข้อ คือ สับเซตจะต้องมีรายการแรกเหมือนกับเส้นทางที่ทำการหาสับเซตเท่านั้น และสับเซตจะต้องไม่เป็นเส้นทางย่อยของเส้นทางเดิม ตัวอย่างเช่น เซตของรายการ (1, 2, 3, 5) ทำการหาสับเซตที่ต้องการคือ {(1, 2, 5), (1, 3, 5), (1, 5)} จากนั้นพิจารณาค่าสนับสนุนของสับเซต ถ้าสับเซตใดมีค่าสนับสนุนน้อยกว่าค่าสนับสนุนขั้นต่ำ จะต้องตรวจสอบว่าเส้นทางที่ยังไม่ได้ทำการหาสับเซตนั้นมีจำนวนรายการมากกว่าจำนวนรายการในสับเซตหรือไม่ หากไม่มีเส้นทางที่มีความยาวมากกว่าความยาวของสับเซต จะทำการลบสับเซตนั้นออกจากการพิจารณา เมื่อได้ทุกสับเซตที่ต้องการแล้วให้ทำการเพิ่มค่าสนับสนุนของแต่ละสับเซตให้กับต้นไม้แสดงรายการความถี่ จากนั้นทำการหาสับเซตของทุกเส้นทางในต้นไม้ และเพิ่มค่าสนับสนุนของทุกสับเซตให้กับต้นไม้แสดงรายการความถี่ เมื่อทำการหาสับเซตครบทุกเส้นทางจะได้ต้นไม้แสดงรายการความถี่ที่มีเซตรายการความถี่ที่มีค่าสนับสนุนถูกต้อง ขั้นตอนการหาสับเซตโดยละเอียดมีดังนี้

อัลกอริทึมการหาสับเซตที่จำเป็น

Input : Frequent item tree, minimum support
Output : Frequent item tree which is correct result
Method : Find Subset

Procedure FindSubset (Frequent item tree, minimum support)
begin
(1) **for** all path from root to any leave in Frequent item tree **do**
(2) each $path_j$ such that have 4 node at least find necessary subset of the Path
(3) S = subset of $path_j$ such that have the first node as $path_j$ and subset is not subpath of $path_j$
(4) **for** each $subset_i \in S$ **do**
(5) if support of $subset_i \geq$ minimum support
(6) current node = root of Frequent item tree
(7) **for** k←1 to | $subset_j$| **do**
(8) **if** a_{jk} is a child of current node **then**
(9) add support(a_{jk}) to support of that node
(10) **else**
(11) child(current node)←create node with a_{jk} and connect node with Header table
(12) support (child (current node)) ← support(a_{jk})
(13) **endif**
(14) current node←child (current node)
(15) **end**
(16) **end**
(17) **end**
end

หลังจากขั้นตอนการหาสับเซตที่จำเป็น จะได้เซตรายการความถี่ทุกระดับ ซึ่งถูกจัดเก็บอยู่ในตารางรายการและต้นไม้แสดงรายการความถี่ แต่ในต้นไม้แสดงรายการความถี่ยังมีบางบัพที่มีค่าสนับสนุนน้อยกว่าค่าสนับสนุนขั้นต่ำจึงต้องทำขั้นตอนต่อไปคือ การตัดเล็มต้นไม้แสดงรายการความถี่

4.2.3. การตัดเล็มต้นไม้แสดงรายการความถี่

ขั้นตอนสุดท้ายของการหาเซตรายการความถี่จากต้นไม้แสดงรายการความถี่ คือ การตัดเล็มต้นไม้แสดงรายการความถี่เพื่อให้ได้ผลลัพธ์ที่ถูกต้องครบถ้วน ซึ่งการตัดเล็มต้นไม้มีขั้นตอนการทำงานเหมือนกับต้นไม้แสดงรายการความถี่ทุกประการ

เมื่อดำเนินการครบทุกขั้นตอนของการหาเซตรายการความถี่จากต้นไม้แสดงรายการความถี่ จะได้ต้นไม้แสดงรายการความถี่ที่มีเซตรายการความถี่ที่มีค่าสนับสนุนถูกต้อง สามารถนำ

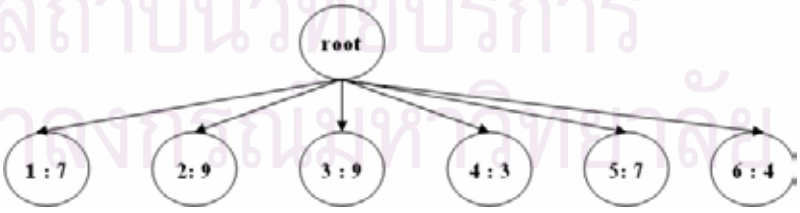
เซตรายการความถี่เหล่านั้นไปสร้างเป็นกฎความสัมพันธ์ของข้อมูลเพื่อนำไปประกอบการตัดสินใจต่อไป

ตัวอย่าง 4.1 กำหนดให้ฐานข้อมูลประกอบด้วยข้อมูล 10 รายการเปลี่ยนแปลง ดังตารางที่ 2.1 และ ค่าสนับสนุนขั้นต่ำมีค่าเท่ากับ 2 (20 %) สามารถหาเซตรายการความถี่โดยใช้การปรับปรุงต้นไม้แสดงรายการความถี่ได้ดังนี้

ขั้นตอนการสร้างต้นไม้แสดงรายการความถี่ เริ่มจากอ่านข้อมูลจากฐานข้อมูลเพื่อหาเซตรายการความถี่ระดับที่ 1 ที่มีค่าสนับสนุนมากกว่าหรือเท่ากับค่าสนับสนุนขั้นต่ำที่กำหนดไว้ได้เป็น L_1 = {(1:7), (2:9), (3:9), (4:3), (5:7), (6:4)} จากนั้นนำสมาชิกในเซตรายการความถี่ระดับที่ 1 มาสร้างตารางรายการได้ดัง รูปที่ 4.1 และนำแต่ละรายการในเซตรายการความถี่ระดับที่ 1 มาสร้างต้นไม้แสดงรายการความถี่ดัง รูปที่ 4.2

	1
{0}	2
{0,0}	3
{0,0,0}	4
{0,0,0,0}	5
{0,0,0,0,0}	6

รูปที่ 4.1 ตารางรายการของต้นไม้แสดงรายการความถี่



รูปที่ 4.2 การสร้างต้นไม้แสดงรายการความถี่จากเซตรายการความถี่ระดับที่ 1

ในการปรับปรุงการสร้างต้นไม้แสดงรายการความถี่ ข้อมูลที่จัดเก็บในตารางรายการจะมีการเปลี่ยนแปลงเล็กน้อย ซึ่งจากเดิมตารางรายการจะเก็บข้อมูลเฉพาะชื่อรายการ และตัวชี้ที่ใช้สำหรับการเชื่อมโยงจากตารางรายการไปยังบัพในต้นไม้เท่านั้น แต่ในการปรับปรุงต้นไม้แสดง

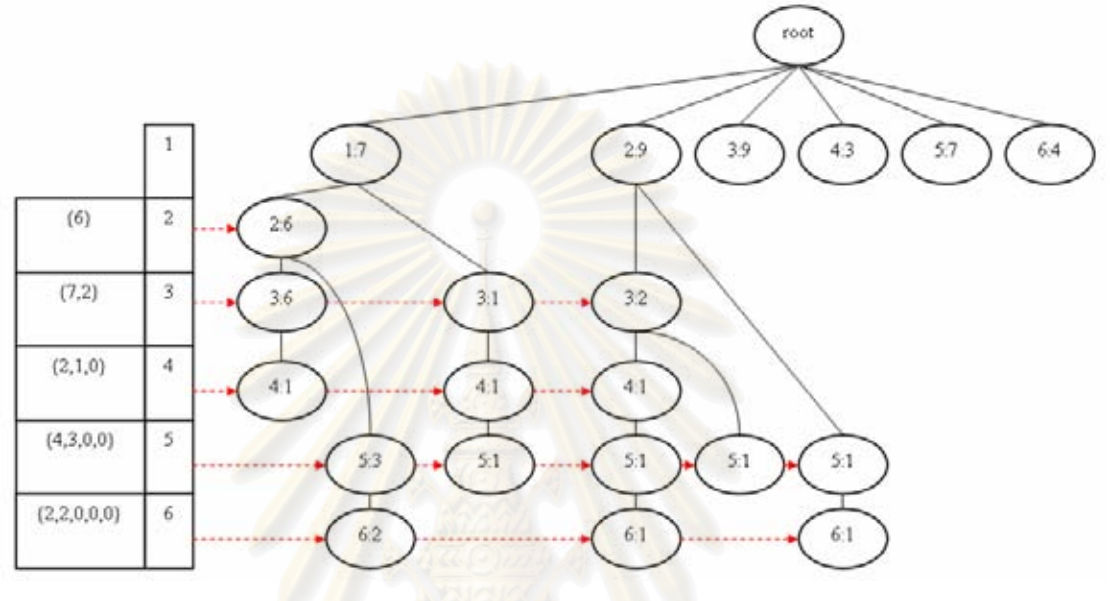
รายการความถี่ จะทำการเพิ่มค่าความถี่ของรายการที่มีความสัมพันธ์กับรายการแรกในรายการเปลี่ยนแปลงด้วยเพื่อลดการหาสับเซตขนาด 2 รายการ และลดการหาสับเซตของรายการที่มีความสัมพันธ์กับรายการแรกของตารางรายการที่มีค่าสนับสนุนน้อยกว่าค่าสนับสนุนขั้นต่ำ ซึ่งการลดขั้นตอนดังกล่าวสามารถลดเวลาในการคำนวณ และลดจำนวนหน่วยความจำที่ใช้ได้ จากรูปที่ 4.1 พิจารณารายการ 6 คอลัมน์แรกของตารางรายการประกอบด้วยเลขศูนย์ทั้งหมด 5 ตำแหน่ง หมายถึง ค่าสนับสนุนของรายการ 6 ที่มีความสัมพันธ์กับรายการอื่นๆที่เป็นรายการก่อนหน้า เลขศูนย์ตำแหน่งที่ 1 หมายถึง ค่าสนับสนุนของรายการ 6 ที่มีความสัมพันธ์กับรายการ 1 เลขศูนย์ตำแหน่งที่ 2 หมายถึง ค่าสนับสนุนของรายการ 6 ที่มีความสัมพันธ์กับรายการ 2 และเลขศูนย์ตำแหน่งอื่นๆ หมายถึง ค่าสนับสนุนของรายการ 6 ที่มีความสัมพันธ์กับรายการ 3 รายการ 4 และรายการ 5 ตามลำดับ

ขั้นตอนต่อไปทำการอ่านข้อมูลจากฐานข้อมูลทีละรายการเปลี่ยนแปลงเพื่อตรวจสอบรายการในรายการเปลี่ยนแปลง หากรายการใดในรายการเปลี่ยนแปลงไม่เป็นสมาชิกของตารางรายการให้ทำการลบรายการนั้นออกจากรายการเปลี่ยนแปลง เมื่อทำการตรวจสอบทุกรายการในรายการเปลี่ยนแปลงแล้วจึงนำรายการในรายการเปลี่ยนแปลงไปสร้างต้นไม้แสดงรายการความถี่ เช่น รายการเปลี่ยนแปลงที่ 1 = {1, 2, 3} สร้างต้นไม้แสดงรายการความถี่ได้ดัง รูปที่ 4.3

รูปที่ 4.3 ต้นไม้แสดงรายการความถี่จากรายการในรายการเปลี่ยนแปลงที่ 1

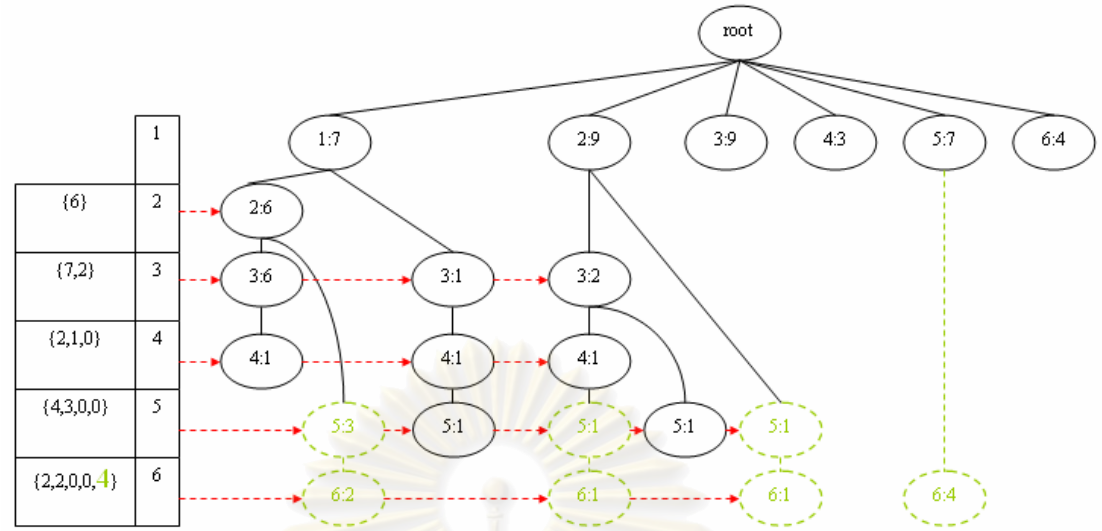
จากรูปที่ 4.3 เป็นการสร้างต้นไม้แสดงรายการความถี่ของรายการในรายการเปลี่ยนแปลงที่ 1 ซึ่งจะนำแต่ละรายการมาสร้างต้นไม้แสดงรายการความถี่ เมื่อทำการสร้างแต่ละบัพในต้นไม้แล้วจะต้องทำการเพิ่มค่าสนับสนุนให้กับตารางรายการด้วย ตัวอย่างเช่น สร้างบัพของรายการ 2

จะต้องเพิ่มค่าสนับสนุนให้กับตารางรายการด้วย จากคอลัมน์แรกของบรรทัดที่สองของตารางรายการ มีความหมายว่ารายการ 2 เกิดขึ้นพร้อมกับรายการ 1 ทั้งสิ้น 1 รายการเปลี่ยนแปลง จากนั้นสร้างบัพของรายการ 3 ต่อจากรายการ 2 และเพิ่มค่าสนับสนุนในตำแหน่งแรกของรายการ 3 นำรายการอื่นๆมาสร้างบัพและเพิ่มค่าสนับสนุนให้กับตารางรายการ เมื่อทำการสร้างบัพของทุกรายการในทุกรายการเปลี่ยนแปลงจะได้ต้นไม้แสดงรายการความถี่ ดังรูปที่ 4.4



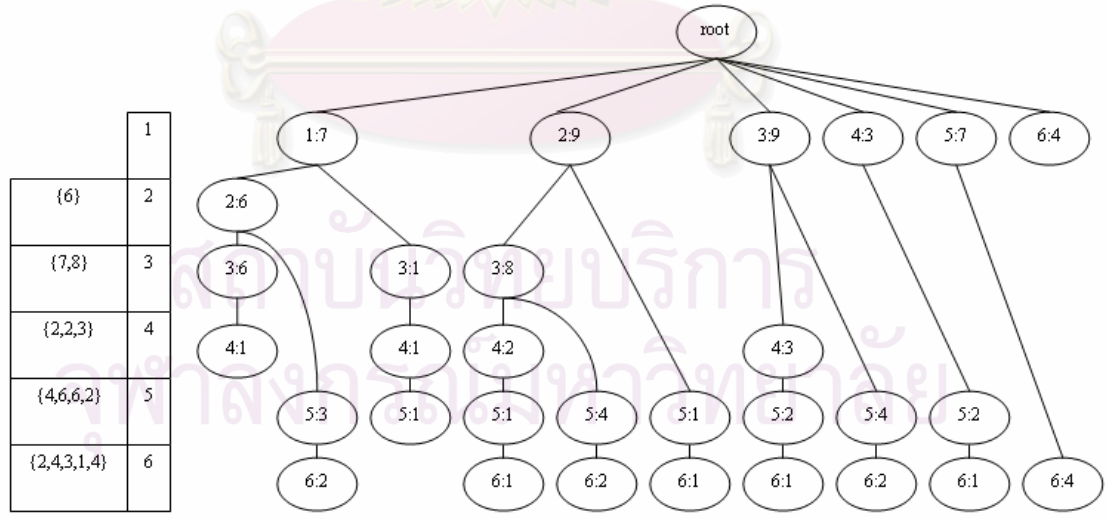
รูปที่ 4.4 ต้นไม้แสดงรายการความถี่ที่สร้างจากฐานข้อมูล

ขั้นตอนต่อไปทำการหาเซตรายการความถี่ เริ่มจากขั้นตอนการรวมค่าสนับสนุนซึ่งจะพิจารณาการเชื่อมโยงของรายการลำดับที่สองจากข้างท้ายในตารางรายการ (รายการ 5) ไปยังบัพในต้นไม้แสดงรายการความถี่ จากนั้นพิจารณาแต่ละเส้นทางของต้นไม้แสดงรายการความถี่ที่เชื่อมต่อกับตารางรายการ โดยเส้นทางที่จะนำมาพิจารณาจะต้องมีจำนวนบัพมากกว่าหรือเท่ากับ 2 บัพขึ้นไป เมื่อพิจารณาเส้นทางของรายการ 5 ได้เป็น {(5, 6:2), (5, 6:1), (5, 6:1)} นำค่าสนับสนุนของแต่ละบัพมาเพิ่มให้กับบัพในต้นไม้แสดงรายการความถี่และเพิ่มค่าสนับสนุนให้กับตารางรายการในตำแหน่งสุดท้ายของราย 6 ดังรูปที่ 4.5



รูปที่ 4.5 การเพิ่มค่าสนับสนุนให้กับต้นไม้แสดงรายการความถี่

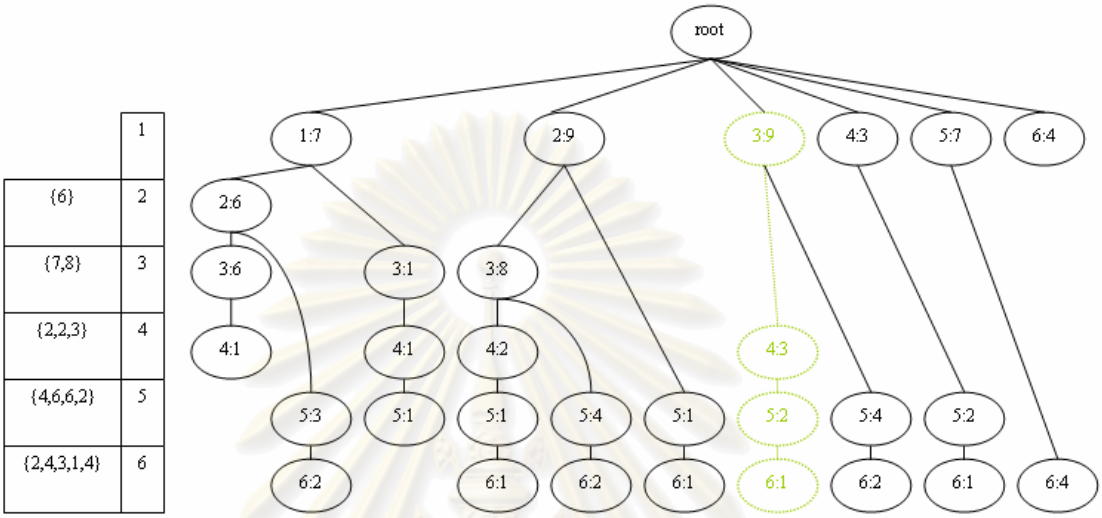
เมื่อพิจารณาการเชื่อมโยงของรายการ 5 แล้ว จะทำการพิจารณาการเชื่อมโยงของรายการลำดับถัดไปในตารางรายการ (รายการ 4) เพิ่มค่าสนับสนุนให้กับต้นไม้แสดงรายการความถี่และเพิ่มค่าสนับสนุนให้กับรายการในตารางรายการ ทำซ้ำจนกระทั่งถึงรายการลำดับที่ 2 ในตารางรายการ (รายการ 2) จะได้ต้นไม้แสดงรายการความถี่และตารางรายการที่มีค่าสนับสนุนเพิ่มขึ้นดังรูปที่ 4.6



รูปที่ 4.6 ต้นไม้แสดงรายการความถี่เมื่อรวมค่าสนับสนุน

ขั้นตอนการทำงานต่อไปของการหาเซตรายการความถี่จากต้นไม้แสดงรายการความถี่ คือ การหาสับเซตที่จำเป็นของทุกเส้นทางในต้นไม้แสดงรายการความถี่ที่มีจำนวนบัพมากกว่าหรือ

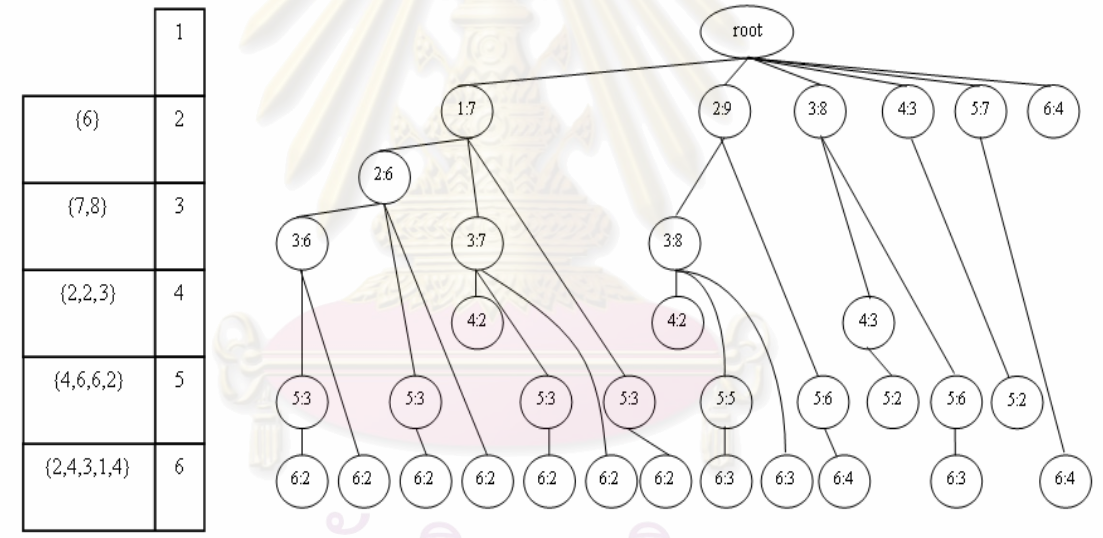
เท่ากับ 4 บัพ การทำงานของการหาสับเซตที่จำเป็นเริ่มจากการพิจารณาเส้นทางของรายการลำดับที่ 4 จากข้างท้ายของตารางรายการ (รายการ 3) โดยจะเริ่มเก็บเส้นทางที่มีรายการแรกเหมือนกันรวมกันจนครบทุกเส้นทาง นำแต่ละเส้นทางไปทำการหาสับเซต เช่น รายการ 3 อ่านเส้นทางได้เป็น {3:9, 4:3, 5:2, 6:1} ซึ่งมีเพียงเส้นทางเดียว ดังรูปที่ 4.7



รูปที่ 4.7 การอ่านเส้นทางเพื่อหาสับเซต

จากรูปที่ 4.7 เมื่ออ่านเส้นทางของรายการ 3 ที่มีจำนวนบัพมากกว่าหรือเท่ากับ 4 บัพ แล้วทำการตรวจสอบรายการในเส้นทางว่ามีค่าสนับสนุนที่เกิดร่วมกับรายการ 3 มากกว่าหรือเท่ากับค่าสนับสนุนขั้นต่ำหรือไม่ หากรายการใดมีค่าสนับสนุนที่เกิดร่วมกับรายการ 3 น้อยกว่าค่าสนับสนุนขั้นต่ำจะทำการลบรายการนั้นออกจากเส้นทาง ซึ่งวิธีการตรวจสอบค่าสนับสนุนของรายการในเส้นทางสามารถทำได้โดยการนำค่าสนับสนุนของรายการในเส้นทางไปตรวจสอบกับค่าสนับสนุนในตารางรายการที่ตำแหน่งของรายการแรกในเส้นทาง เช่น ตรวจสอบรายการ 4 และรายการ 5 ในเส้นทาง {3:9, 4:3, 5:2, 6:1} ซึ่งมีค่าสนับสนุนเท่ากับ 3 และ 2 ตามลำดับ ซึ่งมีค่ามากกว่าและเท่ากับค่าสนับสนุนขั้นต่ำไม่ต้องทำการตรวจสอบกับค่าสนับสนุนในตารางรายการ แต่การตรวจสอบรายการ 6 มีค่าสนับสนุนเท่ากับ 1 ซึ่งมีค่าสนับสนุนน้อยกว่าค่าสนับสนุนขั้นต่ำต้องทำการตรวจสอบกับค่าสนับสนุนของรายการ 6 ในตารางรายการในตำแหน่งที่ 3 (ตำแหน่งของรายการ 3) ปรากฏว่าค่าสนับสนุนของรายการของรายการ 6 ในตารางรายการมีค่าเท่ากับ 3 มากว่าค่าสนับสนุนขั้นต่ำ จึงไม่ต้องลบรายการ 6 ออกจากเส้นทาง เมื่อทำการอ่านเส้นทางและตรวจสอบค่าสนับสนุนของรายการในเส้นทางครบแล้วทำการหาสับเซต ซึ่งสับเซตที่เราต้องการหาจะต้องประกอบด้วยรายการแรกในเส้นทางแรกเสมอ และสับเซตที่ทำการหาจะต้องไม่เป็นเส้นทางย่อยของเส้นทางเดิม จากเส้นทางของรายการ 3 จะสามารถหาสับเซตได้เป็น {(3, 4, 6:1), (3, 5:2, 6:1)} จากนั้นทำการตรวจสอบค่าสนับสนุนของสับเซต หากค่าสนับสนุนของสับเซตมีค่าน้อยกว่า

ค่าสนับสนุนขั้นต่ำจะต้องตรวจสอบว่ามีเส้นทางที่ยังไม่ได้ทำการหาสับเซตและมีความยาวมากกว่าสับเซตหรือไม่ หากไม่มีเส้นทางที่มีความยาวมากกว่าสับเซตจะทำการลบสับเซตนั้นออกจากการพิจารณา (จากตัวอย่าง สับเซต (3, 4, 6:1) จะถูกลบจากการพิจารณาเนื่องจากมีค่าสนับสนุนน้อยกว่าค่าสนับสนุนขั้นต่ำและไม่มีเส้นทางที่มียังไม่ทำการหาสับเซตแล้ว) จากนั้นนำสับเซตที่หาได้มาเพิ่มค่าสนับสนุนให้กับต้นไม้แสดงรายการความถี่ แล้วจึงทำการพิจารณารายการลำดับถัดไปคือรายการ 2 อ่านเส้นทางของรายการ 2 ได้เป็น {(2:9, 3:8, 4:2, 5:1, 6:1), (2:9, 3:8, 5:4, 6:2)} จากนั้นทำการหาสับเซตของทุกเส้นทาง ซึ่งสับเซตของเส้นทาง (2:9, 3:8, 4:2, 5:1, 6:1) คือ {(2, 3, 6:1),(2, 4:2, 6:1)} และสับเซตของเส้นทาง (2:9, 3:8, 5:4, 6:2) คือ {(2, 3, 6:2)} นำแต่ละสับเซตที่ได้มาเพิ่มค่าสนับสนุนให้กับต้นไม้แสดงรายการความถี่ จากนั้นพิจารณารายการลำดับถัดไปจนกระทั่งถึงรายการลำดับแรก เมื่อจบขั้นตอนการหาสับเซตแล้วจะเข้าสู่ขั้นตอนสุดท้าย คือ การตัดเล็มต้นไม้แสดงรายการความถี่ เมื่อทำการตัดเล็มต้นไม้เสร็จแล้วจะได้ต้นไม้แสดงรายการความถี่ที่บรรจุเซตรายการความถี่ทั้งหมดที่มีค่าสนับสนุนที่ถูกต้องครบถ้วนดังรูปที่ 4.8



รูปที่ 4.8 ต้นไม้แสดงรายการความถี่ที่ประกอบด้วยเซตรายการความถี่

จากรูปที่ 4.8 เซตรายการความถี่ทั้งหมดจะถูกบรรจุอยู่ในต้นไม้แสดงรายการความถี่ และตารางรายการ โดยที่เซตรายการความถี่ระดับที่ 2 เท่านั้นที่อยู่ในตารางรายการส่วนเซตรายการความถี่ระดับอื่นๆทั้งหมดจะอยู่ในต้นไม้แสดงรายความถี่ทั้งหมด โดยที่การอ่านเซตรายการความถี่ระดับที่ 2 สามารถอ่านได้จากตารางรายการ เช่น {4, 6, 6, 2} ของรายการ 5 ในตารางรายการ มีความหมายว่า รายการ 1 เกิดร่วมกับรายการ 5 ทั้งสิ้น 4 รายการเปลี่ยนแปลง รายการ 2 เกิดร่วมกับรายการ 5 ทั้งสิ้น 6 รายการเปลี่ยนแปลง รายการ 3 เกิดร่วมกับรายการ 5 ทั้งสิ้น 6 รายการเปลี่ยนแปลง และ รายการ 4 เกิดร่วมกับรายการ 5 ทั้งสิ้น 2 รายการเปลี่ยนแปลง เป็นต้น และ

การอ่านเซตรายการความถี่ระดับอื่นๆ สามารถอ่านได้จากเส้นทางในต้นไม้แสดงรายการความถี่ เช่น เส้นทาง (1:7, 2:6, 3:6, 5:3, 6:2) มีความหมายว่ารายการ 1 เกิดขึ้นทั้งสิ้น 7 รายการ เปลี่ยนแปลง รายการ 1 เกิดร่วมกับรายการ 2 และ รายการ 3 ทั้งสิ้น 6 รายการเปลี่ยนแปลง รายการ 1 เกิดร่วมกับรายการ 2 รายการ 3 และรายการ 5 ทั้งสิ้น 3 รายการเปลี่ยนแปลง และ รายการ 1 เกิดร่วมกับรายการ 2 รายการ 3 รายการ 5 และรายการ 6 ทั้งสิ้น 2 รายการ เปลี่ยนแปลง ซึ่งเซตรายการความถี่ที่ได้จากต้นไม้แสดงรายการความถี่ จะมีความถูกต้องครบถ้วนเหมือนกับเซตรายการความถี่ที่ได้จากอัลกอริทึมอะพริออริ เอฟพี-กโรธอัลกอริทึม และต้นไม้แสดงรายการความถี่ โดยที่กฎความสัมพันธ์ของข้อมูลสามารถหาได้จากเซตรายการความถี่เหล่านั้น

บทที่ 5

การทดลอง

ในบทนี้จะกล่าวถึงการทดสอบการหาเซตรายการความถี่ โดยเปรียบเทียบเวลาในการหาเซตรายการความถี่จากต้นไม้แสดงรายการความถี่กับเอฟพี-กโรธอัลกอริทึม ในการทดสอบอัลกอริทึมทั้งสองจะใช้ข้อมูลจำลองในการทดสอบ โดยที่ข้อมูลจำลองได้มาจากโปรแกรมสร้างข้อมูลของไอบีเอ็มที่มีชื่อว่า ไอบีเอ็มเจน (IBM-gen) ข้อมูลที่ได้จะอยู่ในรูปของแฟ้มข้อมูลซึ่งถูกแบ่งออกเป็นแต่ละรายการเปลี่ยนแปลง ในการทดสอบการหาเซตรายการความถี่จะทดสอบกับข้อมูลหลายชุดข้อมูลและมีค่าพารามิเตอร์ต่างกัน ซึ่งพารามิเตอร์ต่างๆที่เกี่ยวข้องกับชุดข้อมูลมีดังนี้

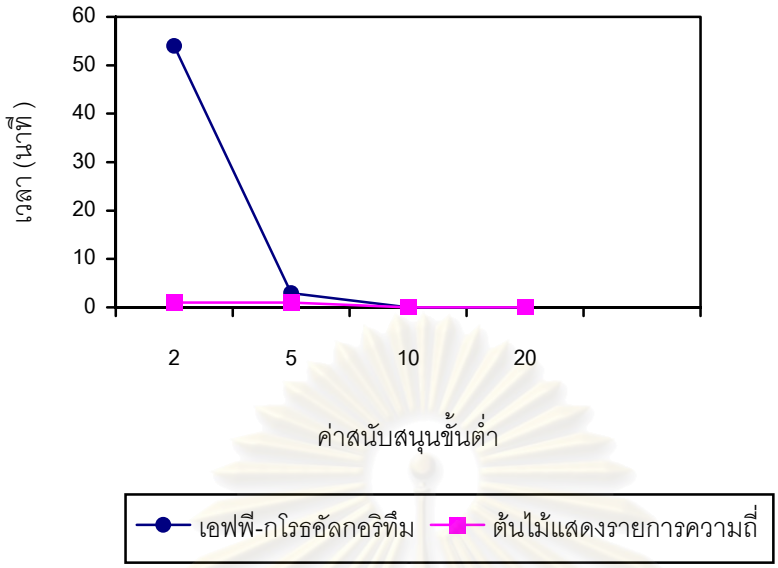
1. |D| คือ จำนวนรายการเปลี่ยนแปลงทั้งหมดของข้อข้อมูล
2. |T| คือ ความยาวโดยเฉลี่ยของแต่ละรายการเปลี่ยนแปลงในฐานข้อมูล
3. |I| คือ ค่าเฉลี่ยสูงสุดของเซตของรายการที่มีโอกาสเป็นเซตรายการความถี่
4. |L| คือ จำนวนสูงสุดของเซตของรายการที่มีโอกาสเป็นเซตรายการความถี่
5. |N| คือ จำนวนรายการทั้งหมดที่ถูกพิจารณาในชุดข้อมูล

การทดสอบอัลกอริทึมในงานวิจัยนี้ เครื่องที่ใช้มีหน่วยประมวลผลกลาง (CPU) เพนเทียม-โฟว์ (Pentium 4) 2.4 กิกะเฮิร์ทซ์ (GHZ) หน่วยความจำหลัก 1 กิกะไบต์ ซึ่งการทดสอบอัลกอริทึมถูกแบ่งออกเป็น 3 ส่วนย่อย คือ การทดสอบแบบแปรผันตามค่าสนับสนุนขั้นต่ำ การทดสอบแบบแปรผันตามจำนวนรายการเปลี่ยนแปลง และการทดสอบแปรผันตามจำนวนรายการโดยเฉลี่ยของแต่ละรายการเปลี่ยนแปลง ผลการทดสอบการหาเซตรายการความถี่แสดงได้ดังนี้

5.1 การทดสอบแบบแปรผันตามค่าสนับสนุนขั้นต่ำ

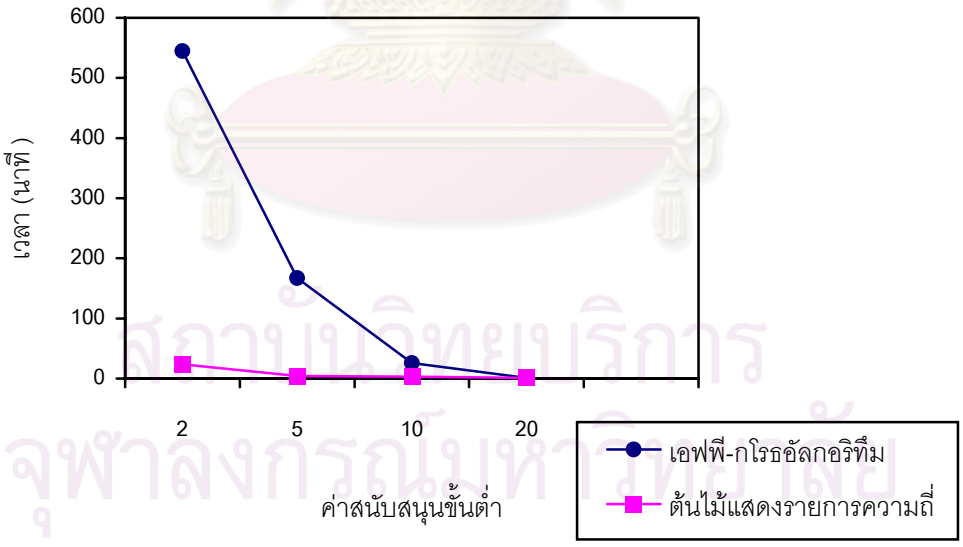
การทดสอบการหาเซตรายการความถี่แปรผันตามค่าสนับสนุนจะทำการทดสอบอัลกอริทึมกับค่าสนับสนุนขั้นต่ำหลายๆ ค่าต่อหนึ่งชุดข้อมูล ซึ่งค่าสนับสนุนขั้นต่ำที่ใช้ในการทดสอบอัลกอริทึมมีค่าตั้งแต่ ร้อยละ 20 จนกระทั่ง ร้อยละ 0.25 โดยที่ในการทดสอบข้อมูลแต่ละชุดจะต้องทดสอบกับค่าสนับสนุนขั้นต่ำดังกล่าว ผลการทดสอบอัลกอริทึมแปรผันตามค่าสนับสนุนขั้นต่ำมีดังนี้

การทดสอบการหาเซตรายการความถี่กับข้อมูลที่มี 100 รายการเปลี่ยนแปลง จำนวนรายการที่พิจารณา 1,000 รายการ และ ความยาวรายการเปลี่ยนแปลงโดยเฉลี่ย 10 รายการ



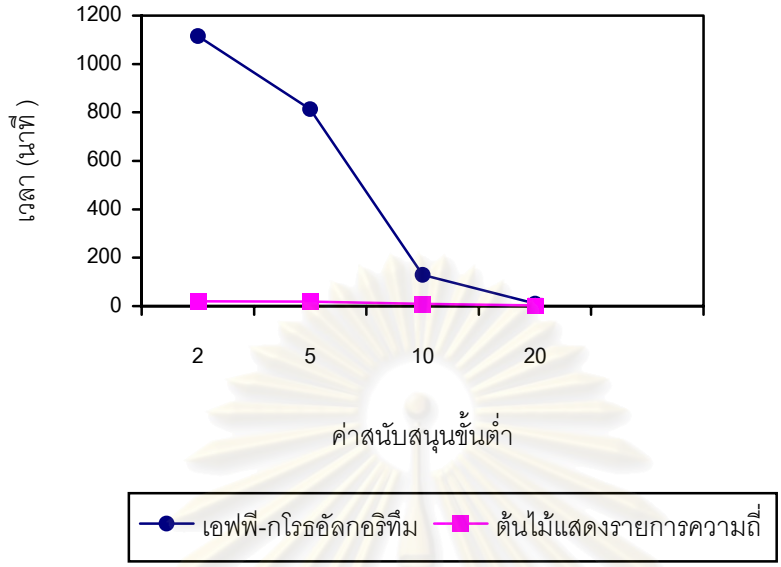
รูปที่ 5.1 การทดสอบการหาเซตรายความถี่กับข้อมูล 100 รายการเปลี่ยนแปลง ความยาวรายการเปลี่ยนแปลงโดยเฉลี่ย 10 รายการ

การทดสอบการหาเซตรายการความถี่กับข้อมูลที่มี 100 รายการเปลี่ยนแปลง จำนวนรายการที่พิจารณา 1,000 รายการ และ ความยาวรายการเปลี่ยนแปลงโดยเฉลี่ย 12 รายการ



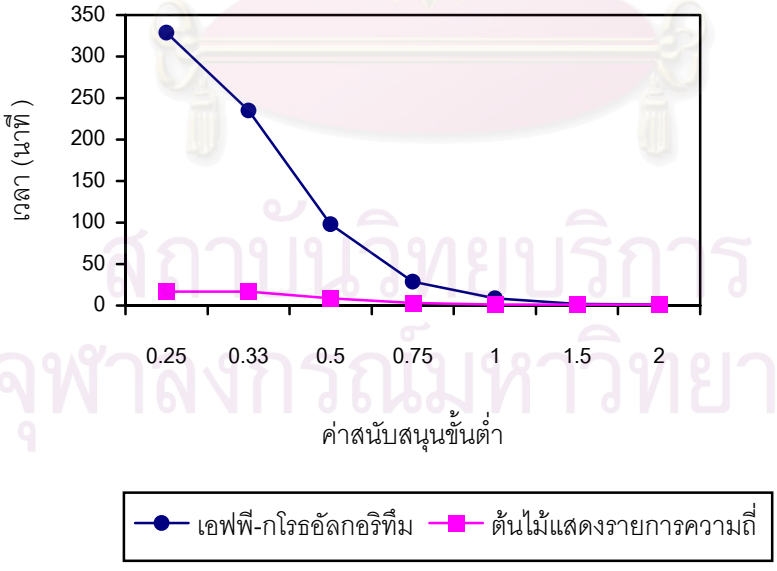
รูปที่ 5.2 การทดสอบการหาเซตรายความถี่กับข้อมูล 100 รายการเปลี่ยนแปลง ความยาวรายการเปลี่ยนแปลงโดยเฉลี่ย 12 รายการ

การทดสอบการหาเซตรายการความถี่กับข้อมูลที่มี 100 รายการเปลี่ยนแปลง จำนวนรายการที่พิจารณา 1,000 รายการ และ ความยาวรายการเปลี่ยนแปลงโดยเฉลี่ย 14 รายการ



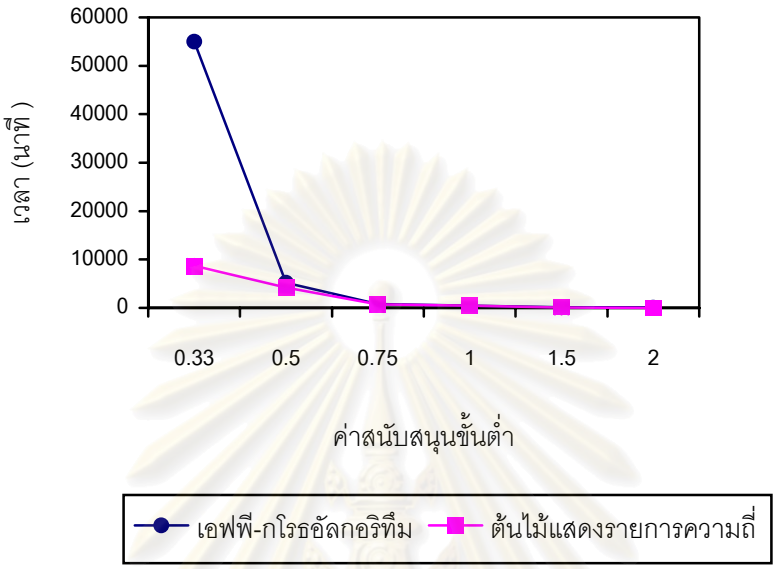
รูปที่ 5.3 การทดสอบการหาเซตรายความถี่กับข้อมูล 100 รายการเปลี่ยนแปลง ความยาวรายการเปลี่ยนแปลงโดยเฉลี่ย 14 รายการ

การทดสอบการหาเซตรายการความถี่กับข้อมูลที่มี 100,000 รายการเปลี่ยนแปลง จำนวนรายการที่พิจารณา 1,000 รายการ ความยาวรายการเปลี่ยนแปลงโดยเฉลี่ย 5 รายการ และจำนวนสูงสุดของเซตของรายการที่มีโอกาสเป็นเซตรายการความถี่เท่ากับ 2



รูปที่ 5.4 การทดสอบการหาเซตรายความถี่กับข้อมูล 100,000 รายการเปลี่ยนแปลง ความยาวรายการเปลี่ยนแปลงโดยเฉลี่ย 5 รายการ

การทดสอบการหาเซตรายการความถี่กับข้อมูลที่มี 100,000 รายการเปลี่ยนแปลง จำนวนรายการที่พิจารณา 1,000 รายการ ความยาวรายการเปลี่ยนแปลงโดยเฉลี่ย 10 รายการ และจำนวนสูงสุดของเซตของรายการที่มีโอกาสเป็นเซตรายการความถี่เท่ากับ 2



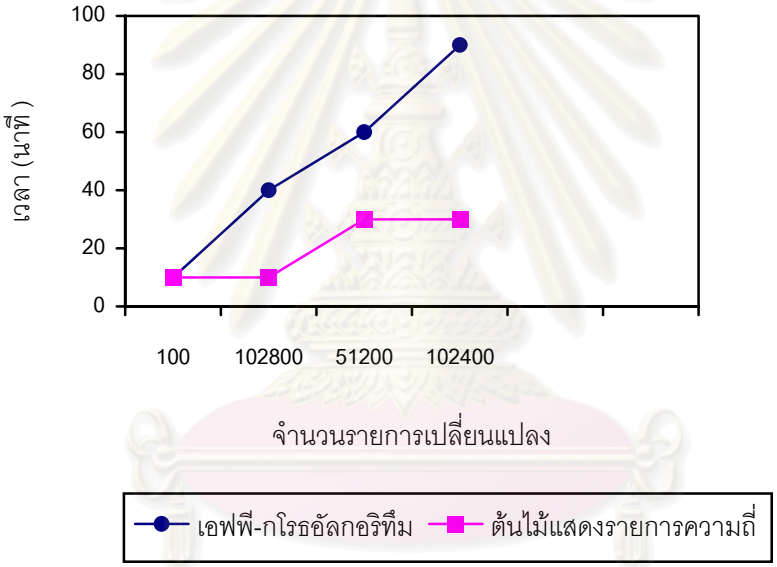
รูปที่ 5.5 การทดสอบการหาเซตรายความถี่กับข้อมูล 100,000 รายการเปลี่ยนแปลง ความยาวรายการเปลี่ยนแปลงโดยเฉลี่ย 10 รายการ

จากรูปที่ 5.1, 5.2, 5.3, 5.4 และ 5.5 สังเกตได้ว่าเมื่อค่าสนับสนุนขั้นต่ำมีค่ามากๆ เอฟพี-กโรธอัลกอริทึมและต้นไม้แสดงรายการใช้เวลาในการคำนวณใกล้เคียงกัน เนื่องจากเมื่อค่าสนับสนุนขั้นต่ำมีค่ามากจะทำให้จำนวนเซตรายการความถี่ที่มีค่าสนับสนุนมากกว่าหรือเท่ากับค่าสนับสนุนขั้นต่ำมีจำนวนน้อย แต่เมื่อค่าสนับสนุนขั้นต่ำมีค่าต่ำลงจนกระทั่งน้อยมากๆจะสามารถสังเกตความแตกต่างของเวลาที่ใช้ในการคำนวณระหว่างเอฟพี-กโรธอัลกอริทึม และต้นไม้แสดงรายการความถี่มากขึ้น เนื่องจากเมื่อค่าสนับสนุนขั้นต่ำมีค่าน้อยจะทำให้มีเซตรายการความถี่จำนวนมาก เอฟพี-กโรธอัลกอริทึมต้องสร้างต้นไม้จำนวนหลายต้นมากขึ้น จำนวนรอบในการเรียกซ้ำในการหาเซตรายการความถี่มีจำนวนมาก จำนวนการเรียงลำดับรายการในคอนดิชันนอลแพทเทินเบซหลายครั้ง และยังรวมถึงการนับค่าสนับสนุนให้กับรายการในคอนดิชันนอลแพทเทินเบซอีกด้วย ซึ่งขั้นตอนทั้งหมดจะใช้เวลาในการคำนวณสูง แต่ต้นไม้แสดงรายการความถี่ใช้การท่องบัพในต้นไม้เพื่อรวมค่าสนับสนุน และการหาสับเซตที่จำเป็น เพื่อนับค่าสนับสนุนของเซตรายการความถี่ ซึ่งเวลาส่วนใหญ่ที่ใช้ในการหาเซตรายการความถี่จะใช้ที่ขั้นตอนการหาสับเซตที่จำเป็น เนื่องจากเป็นขั้นตอนการนับค่าสนันสนุนให้กับทุกรายการความถี่ที่อยู่ในต้นไม้แสดงรายการความถี่

5.2 การทดสอบแบบแปรผันตามจำนวนรายการเปลี่ยนแปลง

การทดสอบการหาเซตรายการความถี่แปรผันตามจำนวนรายการเปลี่ยนแปลงจะทำการทดสอบอัลกอริทึมกับหลายชุดข้อมูลที่มีจำนวนรายการเปลี่ยนแปลงต่างกันต่อหนึ่งค่าสนับสนุน ซึ่งชุดข้อมูลที่ใช้ในการทดสอบอัลกอริทึมมีจำนวนรายการเปลี่ยนแปลงตั้งแต่ 100 รายการเปลี่ยนแปลง จนกระทั่ง 102400 รายการเปลี่ยนแปลง ผลการทดสอบอัลกอริทึมแปรผันตามจำนวนรายการเปลี่ยนแปลงมีดังนี้

การทดสอบการหาเซตรายการความถี่กับค่าสนับสนุนขั้นต่ำร้อยละ 5 โดยชุดข้อมูลที่ทำการทดสอบมีความยาวรายการเปลี่ยนแปลงโดยเฉลี่ยเท่ากับ 10 และจำนวนรายการที่พิจารณาเท่ากับ 1,000 รายการ



รูปที่ 5.6 การทดสอบการหาเซตรายการความถี่แปรผันตามจำนวนรายการเปลี่ยนแปลง

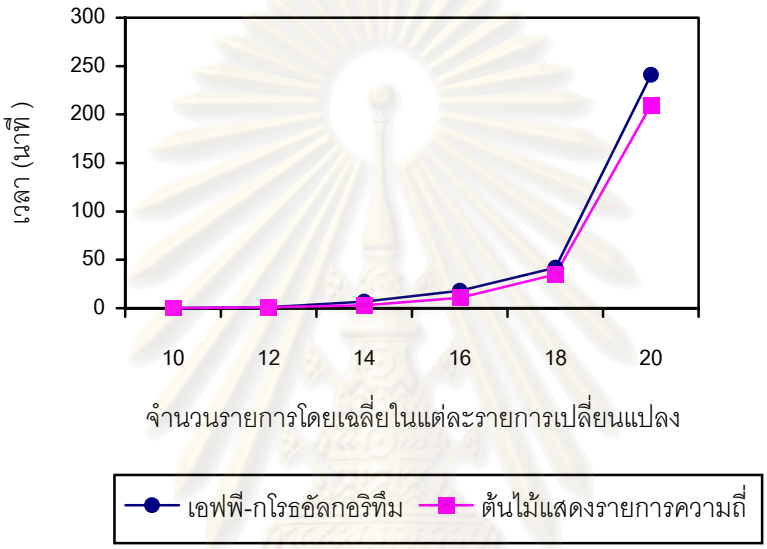
จากรูปที่ 5.6 สังเกตได้ว่าเมื่อจำนวนรายการเปลี่ยนแปลงของชุดข้อมูลมีจำนวนเพิ่มขึ้นจะทำให้อัลกอริทึมทั้งสองใช้เวลาในการคำนวณมากขึ้น โดยที่เอฟพี-กโรธอัลกอริทึมจะใช้เวลาในการคำนวณมากกว่าต้นไม้แสดงรายการความถี่ เนื่องจากต้องทำการเรียงลำดับรายการในทุกรายการเปลี่ยนแปลงที่อยู่ในฐานข้อมูล

5.3 การทดสอบแบบแปรผันตามจำนวนรายการโดยเฉลี่ยของแต่ละรายการเปลี่ยนแปลง

การทดสอบการหาเซตรายการความถี่แปรผันตามจำนวนรายการโดยเฉลี่ยของแต่ละรายการเปลี่ยนแปลงจะทำการทดสอบกับหลายชุดข้อมูลที่มีจำนวนรายการโดยเฉลี่ยของแต่ละ

รายการเปลี่ยนแปลงที่ต่างกันต่อหนึ่งค่าสนับสนุน ซึ่งชุดข้อมูลจะมีจำนวนรายการโดยเฉลี่ยของแต่ละรายการเปลี่ยนแปลงตั้งแต่ 10 รายการ จนกระทั่ง 14 รายการ ผลการทดสอบอัลกอริทึมแปรผันตามจำนวนรายการโดยเฉลี่ยของแต่ละรายการเปลี่ยนแปลงมีดังนี้

การทดสอบการหาเซตรายการความถี่กับค่าสนับสนุนร้อยละ 5 โดยชุดที่ทำการทดสอบมีจำนวนรายการเปลี่ยนแปลงเท่ากับ 100 รายการเปลี่ยนแปลง และจำนวนรายการที่พิจารณาเท่ากับ 1,000 รายการ



รูปที่ 5.7 การทดสอบการหาเซตรายการความถี่แปรผันตามจำนวนรายการเปลี่ยนแปลงโดยเฉลี่ยของแต่ละรายการเปลี่ยนแปลง

จากรูปที่ 5.7 สังเกตได้ว่าเมื่อจำนวนรายการโดยเฉลี่ยของแต่ละรายการเปลี่ยนแปลงในชุดข้อมูลมีจำนวนเพิ่มขึ้นจะทำให้อัลกอริทึมทั้งสองใช้เวลาในการคำนวณเพิ่มขึ้น โดยที่เอฟพี-กโรธอัลอริทึมจะใช้เวลาในการคำนวณมากกว่าค้นไม้แสดงรายการความถี่ เนื่องจากต้องทำการเรียงลำดับรายการในทุกรายการเปลี่ยนแปลง และเมื่อจำนวนรายการในรายการเปลี่ยนแปลงมากขึ้นจะต้องใช้เวลาในการเรียงลำดับมากขึ้นด้วย

บทที่ 6

สรุปผลการวิจัยและข้อเสนอแนะ

6.1 สรุปผลการวิจัย

งานวิจัยชิ้นนี้ได้นำเสนออัลกอริทึมการหาเซตรายการความถี่ทั้งหมดจากฐานข้อมูล เพื่อลดเวลาในการคำนวณ เนื่องจากเดิมเอฟพี-กโรธอัลกอริทึมจะใช้เวลาในการคำนวณมาก โดยเฉพาะในขั้นตอนการสร้างต้นไม้ที่ต้องทำการเรียงลำดับรายการในตารางรายการและรายการในรายการเปลี่ยนแปลง จากนั้นจึงนำรายการในรายการเปลี่ยนแปลงไปสร้างเป็นเอฟพี-ทรี ในขั้นตอนการหาเซตรายการความถี่จะต้องทำการหาคอนดิชันนอลแพทเทิน เบซจากเอฟพี-ทรี แล้วทำการหาเรียงลำดับรายการในคอนดิชันนอลแพทเทินเบซ จากนั้นรวมค่าสนับสนุนของแต่ละรายการในคอนดิชันนอลแพทเทินเบซ และทำการสร้างคอนดิชันนอลเอฟพี-ทรี ซึ่งการหาเซตรายการความถี่จากเอฟพี-กโรธต้องสร้างคอนดิชันนอลเอฟพี-ทรีหลายต้น โดยขั้นตอนการทำงานทั้งหมดของการหาเซตรายการความถี่โดยใช้เอฟพี-กโรธจะใช้เวลาในการคำนวณมาก

ดังนั้น งานวิจัยชิ้นนี้จึงได้นำเสนอวิธีการหาเซตรายการความถี่จากต้นไม้แสดงรายการความถี่โดยใช้ต้นไม้เพียงต้นเดียว ซึ่งใช้โครงสร้างข้อมูลเช่นเดียวกับเอฟพี-กโรธอัลกอริทึม คือ เอฟพี-ทรี ในขั้นตอนการสร้างต้นไม้จะทำการลดขั้นตอนการเรียงลำดับข้อมูลทั้งรายการในตารางรายการและรายการในรายการเปลี่ยนแปลง ในส่วนของขั้นตอนการหาเซตรายการความถี่จะทำการรวมค่าสนับสนุนให้กับต้นไม้ โดยพิจารณาการเชื่อมโยงของรายการในตารางรายการไปยังบัพในต้นไม้ จากนั้นหาสับเซตของทุกๆ เส้นทางในต้นไม้ที่มีจำนวนบัพมากกว่าหรือเท่ากับ 3 บัพ แล้วนำสับเซตที่หาได้มาเพิ่มค่าสนับสนุนให้กับต้นไม้ ขั้นตอนสุดท้ายของการหาเซตรายการความถี่จากต้นไม้แสดงรายการความถี่ คือ การตัดเล็มต้นไม้ เมื่อเสร็จสิ้นขั้นตอนสุดท้ายจะได้ต้นไม้ที่มีเซตรายการความถี่ทั้งหมด

จากการทดลองในบทที่ 4 จะสังเกตได้ว่า เมื่อข้อมูลมีจำนวนรายการเปลี่ยนแปลง และจำนวนรายการโดยเฉลี่ยของแต่ละรายการเปลี่ยนแปลงมีค่ามาก จะทำให้เอฟพี-กโรธอัลกอริทึมจะใช้เวลาในการคำนวณมากกว่าต้นไม้แสดงรายการความถี่ แต่เอฟพี-ทรีจะใช้หน่วยความจำน้อยกว่าต้นไม้แสดงรายการความถี่ เนื่องจากเอฟพี-กโรธอัลกอริทึมทำการเรียงลำดับข้อมูลทำให้มีการซ้ำกันของลำดับของข้อมูลมากจึงสามารถใช้บัพร่วมกันได้มากด้วย และเมื่อค่าสนับสนุนขั้นต่ำมีค่าน้อยทำให้มีเซตรายการความถี่จำนวนมาก ซึ่งเป็นเหตุให้เอฟพี-กโรธอัลกอริทึมใช้เวลาในการคำนวณสูง เนื่องจากต้องทำการหาคอนดิชันนอลแพทเทินเบซ และทำการสร้างคอนดิชันนอลเอฟ

พี-ทรีจำนวนมาก ซึ่งความซับซ้อนเชิงเวลาของทั้งสองอัลกอริทึมมีค่าเท่ากับ $\Theta(n)$ เมื่อ n คือ จำนวนรายการเปลี่ยนแปลงในฐานข้อมูล

นอกจากนี้ยังนำเสนอการปรับปรุงการหาเซตรายการความถี่จากต้นไม้แสดงรายการความถี่ โดยปรับปรุงการจัดเก็บข้อมูลในตารางรายการ ปรับปรุงขั้นตอนการสร้างต้นไม้แสดงรายการความถี่ ปรับปรุงการรวมค่าสนับสนุนให้กับต้นไม้แสดงรายการความถี่ และปรับปรุงการหาสับเซตที่จำเป็น จากการปรับปรุงดังกล่าวสามารถลดจำนวนการหาสับเซตได้ ซึ่งทำให้สามารถลดเวลาในการคำนวณได้ และสามารถลดจำนวนหน่วยความจำที่ใช้ได้ เนื่องจากขั้นตอนการหาสับเซตที่จำเป็น เป็นขั้นตอนที่ใช้เวลาในการคำนวณรวมถึงหน่วยความจำมากที่สุด

6.2 ปัญหาและข้อจำกัดที่พบจากการวิจัย

การทดสอบอัลกอริทึมเพื่อให้อัลกอริทึมมีความน่าเชื่อถือมากขึ้น ควรทำการทดสอบอัลกอริทึมกับข้อมูลที่มีจำนวนรายการเปลี่ยนแปลงมากๆ และ ทำการทดสอบอัลกอริทึมกับข้อมูลที่มีพารามิเตอร์แตกต่างกันหลายพารามิเตอร์ เช่น ทดสอบกับจำนวนรายการที่พิจารณามากน้อยต่างกัน ทดสอบกับจำนวนรายการโดยเฉลี่ยของแต่ละรายการเปลี่ยนแปลงที่ไม่เท่ากัน ทดสอบกับจำนวนรูปแบบหรือข้อมูลที่มีความสัมพันธ์มากน้อยต่างกัน เป็นต้น

จากการทดสอบการหาเซตรายการความถี่จากต้นไม้แสดงรายการความถี่ และการทำงานของเอฟพี-กโรธอัลกอริทึม พบว่าเมื่อข้อมูลมีขนาดใหญ่หรือค่าสนับสนุนขั้นต่ำที่กำหนดไว้มีค่าน้อยอัลกอริทึมทั้ง 2 จะใช้หน่วยความจำมาก เป็นเหตุให้ไม่สามารถทดสอบอัลกอริทึมได้ทุกชุดข้อมูล เนื่องจากเครื่องคอมพิวเตอร์ที่ใช้ในการทดสอบ มีหน่วยความจำไม่เพียงพอต่อความต้องการของอัลกอริทึม และไม่สามารถนำข้อมูลทางด้านธุรกิจมาทดสอบกับอัลกอริทึมได้

6.3 ข้อเสนอแนะ

งานวิจัยชิ้นนี้นำเสนอการหาเซตรายการความถี่ทั้งหมดจากฐานข้อมูล โดยใช้ต้นไม้แสดงรายการความถี่ ซึ่งสามารถนำกระบวนการหาเซตรายการความถี่จากต้นไม้แสดงรายการความถี่ไปปรับปรุงเพื่อหาเซตรายการความถี่เพียงบางตัว (Closed itemset) เพื่อลดเวลาในการคำนวณ นอกจากนี้ ยังสามารถนำวิธีการดังกล่าวไปสร้างเป็นโปรแกรมเพื่อในใช้การตัดสินใจในเชิงธุรกิจได้

รายการอ้างอิง

1. Agrawal, R. and Imielinski, T. 1993. Mining association rules between sets of items in large databases Proceeding of ACM SIGMOD.
2. Agrawal, R. and Srikant, R. 1994. Fast algorithm for mining association rules Proceeding of VLDB Conference.
3. Amir, A., Fedman, R. and Kashi, R. 1997. A new and versatile method for association generation In Infortmation Systems.
4. Brin, S., Motwani, R., Ullman, J.D. and Tsur, S. 1997. Dynamic itemset counting and implication rules for market basket data Proceeding of ACM SIGMOD.
5. Han, J., Pei, J. and Yin, Y. 2000. Mining frequent patterns without candidate generation Proceeding of ACM SIGMOD.
6. Pei, J., Han, J., Lu, H., Nishio, S., Tang, S. and Yang, D. 2001. H-mine: Hyper structure mining of frequent patterns in large databases Proceeding of ICDM.
7. Savasere, A., Omiecinski, E. and Navathe, S. 1995. An efficient algorithm for mining association rules in large databases Proceeding of VLDB conference.
8. Yang, D., Johar, A., Grama, A. and Szpankowski, W. 2000. Summary structures for frequency queries on large transaction sets Data compression conference.
9. Zaki, M.J., Parthasarathy, S., Orihara, M. and Li, W. 1997. Parallel algorithm for fast discovery of association rules Data mining and knowledge discovery: An international, special issue on scalable high-performance computing for KDD.

ประวัติผู้เขียนวิทยานิพนธ์

นายโกเมศ อัมพวัน เกิดเมื่อวันที่ 16 ตุลาคม พ.ศ. 2524 เรียนจบการศึกษาระดับมัธยมศึกษาตอนปลายจากโรงเรียนชลราษฎรอำรุง อ.เมือง จ.ชลบุรี เข้ารับการศึกษาต่อที่มหาวิทยาลัยบูพา ในคณะวิทยาศาสตร์ สาขาวิชาวิทยาการคอมพิวเตอร์ ภาควิชาวิทยาการคอมพิวเตอร์ และสำเร็จการศึกษาในระดับปริญญาบัณฑิตในปี พ.ศ. 2546