การพัฒนาเครื่องมือสำหรับวิเคราะห์หารูปแบบตามลำดับในทรานแซ็กชั่น โดยใช้ดาต้าไมนิ่ง

Development of analysis tool for sequential pattern in transaction by using data mining

โดย

นางสาวนิรมล กลั่นเรืองแสง

รหัส 42067005

อาจารย์ที่ปรึกษา

รศ.ดร วิเชียร เปรมชัยสวัสดิ์

รายงานนี้เป็นส่วนหนึ่งของวิชาโครงการพัฒนาระบบงาน

หลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ

ภาคเรียนที่ 2 ปีการศึกษา 2543

คณะเทคโนโลยีสารสนเทศ

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ชื่อหัวข้อ	การพัฒนาเครื่องมือสำหรับวิเคราะห์หารูปแบบตามลำดับในทรานแซ็กชัน โดยใช้ดาต้าไมนิ่ง
นักศึกษา	นางสาวนิรมล กลั่นเรืองแสง
อาจารย์ที่ปรึกษา	รศ.ดร วิเชียร เปรมชัยสวัสดิ์
ระดับการศึกษา	วิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ
แขนงวิชา	วิทยาการสารสนเทศ
ปีการศึกษา	2543

บทคัดย่อ

การทำไมนิ่งกับฐานข้อมูลมักถูกนำมาใช้เพื่อประกอบในการแก้ปัญหาในการตัดสินใจ หรือนำมาใช้เพื่อหาความรู้ใหม่ ๆ ที่ซ่อนอยู่ภายในฐานข้อมูล ปัจจุบันข้อมูลมีจำนวนมากขึ้น ทำให้เราสามารถนำข้อมูลเหล่านี้เพื่อนำมาหาความรู้ที่น่าสนใจและสามารถนำไปใช้ประโยชน์จากมันได้ สำหรับข้อมูลของการซื้อขาย สามารถนำการวิธีการค้นหาความสัมพันธ์ตามลำดับ (Sequential Relationship) หรือรูปแบบตามลำดับ (Sequential Pattern) ที่อยู่ในข้อมูลเหล่านี้มาใช้ประโยชน์ในหลาย ๆ แบบ เช่น การทำนายเหตุการณ์โดยใช้กฎการเกิดแบบเป็นลำดับ ทางด้านการตลาด การวางแผนการขายและกลยุทธ์เป็นต้น

ในโครงงานฉบับนี้ได้ศึกษาถึงวิธีการในการหารูปแบบตามลำดับที่เกิดในทรานแซ็กชันของการซื้อขายสินค้าของผู้บริโภค และนำมาปรับใช้ในการสร้างโปรแกรมวิเคราะห์หารูปแบบตามลำดับ

Tile	Development of analysis tool for sequential pattern in transaction by using data mining
Student	Miss. Niramon Glunruangsang
Advisor	Assoc. Prof. Dr. Wichian Premchaiswadi
Level of Study	Master of Science of Information Technology
Major	Information Science
Academic Year	2000

ABSTRACT

Data mining is motivated by the decision support problem or discovering knowledge that inherent among the data. From transaction have inherent sequential nature to them. The discovery of sequential relationship or pattern present in such data is useful for various purposes such as event or identification of sequential rule, enable marketers to develop and implement customize marketing programs and strategies

In this project we present step to solve the problem of finding all sequential pattern and present algorithm to solve and adjust it for Developing analysis tool for sequential pattern

กิตติกรรมประกาศ

โครงการพัฒนาระบบงานเรื่องการพัฒนาเครื่องมือสำหรับวิเคราะห์หารูปแบบตามลำดับในทรานแซ็กชันโดยใช้ดาต้าไมนิ่ง ผู้เขียนขอขอบพระคุณท่าน รศ.ดร วิเชียร เปรมชัยสวัสดิ์ อาจารย์ที่ปรึกษาได้กรุณาและแนะนำในการแก้ไขปัญหา ทำให้สามารถแก้ไขปัญหาที่เกิดขึ้นให้ผ่านพ้นไปได้ และขอขอบเพื่อน ๆ สำหรับคำแนะนำและแก้ไขปัญหา

นิรมล กลั่นเรืองแสง

กุมภาพันธ์ 2544

สารบัญ

หน้า

หน้า

สารบัญรูป

หน้า

รูปที่

รูปที่ หน้า

บทที่ 1

บทนำ

โดยทั่วไปในงานทางวิทยาศาสตร์และทางการค้า พบว่าข้อมูลจะมีปริมาณเพิ่มขึ้นเมื่อเวลาผ่านไป โดยข้อมูลเหล่านี้มีประโยชน์และสำคัญในการค้นหาความรู้หรือข้อมูลบางอย่างที่ซ่อนอยู่ข้อมูลเหล่านี้ การเก็บข้อมูลการทดลองทางวิทยาศาตร์หรือความสนใจข้อมูลของระบบทางฟิสิก เช่น telecommunication network หรือเป็นทรานแซ็กชัน(transaction)ของการซื้อขายใน supermarket โดยธรรมชาติแล้วมักพบว่าจะมีลำดับของข้อมูลซ่อนอยู่ภายในข้อมูลเหล่านั้น การค้นหาความสัมพันธ์ที่เป็นลำดับหรือรูปแบบที่อยู่ในข้อมูล สามารถนำไปให้เกิดเป็นประโยชน์ได้หลายทาง เช่นใช้ในการทำนายเหตุการณ์ที่จะเกิดขึ้นต่อไป หรือกำหนดกฎที่เป็นลำดับ

การหารูปแบบตามลำดับสามารถทำได้กับงานหลายแบบ ตัวอย่างเช่น

- การวิเคราะห์ข้อมูลจากกการทดลองทางวิทยาศาสตร์ ที่สัมพันธ์กับช่วงเวลา
- การค้นหาความสัมพันธ์ของเหตุการณ์ที่เกิดในตลาดหุ้น
- การวิเคราะห์ข้อมูลการรักษาทางการแพทย์ของผู้ป่วย เพื่อหารูปแบบที่เป็นข้อมูลระหว่างวิธีรักษาอาการของโรค, อาการ และผลที่ได้

การทำไมนิ่งกับฐานข้อมูลมักทำเพื่อการสนับสนุนการตัดสินใจ การหารูปแบบตามลำดับเป็นการค้นหาความรู้ในฐานข้อมูลที่มี ในกรณีของฐานข้อมูลของการซื้อขาย เราสามารถนำการหารูปแบบตามลำดับเพื่อหาข้อมูลที่มีประโยชน์เกี่ยวกับพฤติกรรมของผู้บริโภค โดยที่ข้อมูลเหล่านี่สามารถนำมาใช้ประโยชน์ทางด้านการตลาด การวางแผนการขายเป็นต้น

ฐานข้อมูลของการซื้อขายของลูกค้า (Customer Transaction) แต่ละทรานแซ็กชันประกอบด้วยฟิลด์คือ customer-id , transaction-time และ item โดยเราจะทำการพัฒนาเครื่องมือเพื่อทำการวิเคราะห์หารูปแบบตามลำดับ ผลที่ได้คือรูปแบบตามลำดับที่เกิดขึ้นในทรานแซ็กชันโดยมีความถี่ตามแต่ผู้วิเคราะห์กำหนด(minimum support) โดยรูปแบบตามลำดับที่ได้เหล่านี้สามารถนำไปใช้ประกอบการตัดสินใจเพื่อการวางแผนการจัดการคลังสินค้า เช่นเมื่อรู้ลำดับในการซื้อสินค้าของลูกค้า จะสามารถกำหนดได้ว่าควรจะเตรียมสินค้าในลำดับที่ลูกค้าน่าจะซื้อให้เพียง

พอ หรือนำไปใช้สำหรับการจัดการสนับสนุนการขาย เช่นการขายสินค้าเป็นคู่ ภายในสินค้าที่อยู่ในลำดับที่หาได้

1.1 จุดประสงค์

1. เพื่อพัฒนาเครื่องมือสำหรับวิเคราะห์หารูปแบบตามลำดับในทรานแอ็กชันของการขายสินค้า
2. สามารถนำรูปแบบตามลำดับที่ได้จากเครื่องมือที่พัฒนา นำไปใช้เพื่อประกอบการตัดสินใจ (Decision making)
3. นำความรู้ทางดาต้าไมนิ่งและอัลกอริทึมที่ศึกษา มาประยุกต์และ เพื่อให้ใช้งานจริงได้

1.2 แนวทางการศึกษา

1. ศึกษาแนวทางการนำดาต้าไมนิ่ง เพื่อนำมาใช้ในการวิเคราะห์หารูปแบบตามลำดับในทรานแซ็กชัน
2. กำหนดขอบเขตของการทำงาน
3. ศึกษาอัลกอริทึมในการหารูปแบบตามลำดับ วิเคราะห์ความเป็นไปได้และพิจารณาตัดสินใจเลือก อัลกอริทึมที่เหมาะสมในการทำงาน และเลือกเครื่องมือที่ใช้ในการพัฒนา
4. พัฒนาโปรแกรมและทดสอบ

1.3 ขอบเขตการทำงาน

ในโครงการพัฒนาระบบงานฉบับนี้มีจุดประสงค์เพื่อและพัฒนาโปรแกรมสำหรับวิเคราะห์หารูปแบบตามลำดับในทรานแซ็กชันโดยใช้ดาต้าไมนิ่ง การทำงานจะเป็นพัฒนาโปรแกรมบน Windows

แต่เนื่องจากข้อมูลเกี่ยวกับทรานแซ็กชันของการขายสินค้าในสถานะการจริงนั้นไม่สามารถหาได้ เนื่องจากเป็นข้อมูลที่ร้านค้าไม่สามารถเปิดเผยแก่บุคคลภายนอก ข้อมูล input สำหรับโปรแกรมที่ได้พัฒนานี้ จึงได้ใช้ข้อมูลที่ได้จากการใช้โปรแกรมสร้างจำลองขึ้น โดยโปรแกรมที่ใช้นี้เป็นโปรแกรมจาก web site ของ IBM (**http : / / www.almaden.ibm.com / cs / quest / syndata . html #instructions)** โดยที่โปรแกรมนี้มีการใช้ในการจำลองข้อมูลสำหรับการ

ทดลองเกี่ยวกับการหารูปแบบตามลำดับในหลาย paper ที่ได้ศึกษามา ผลที่ได้จากโปรแกรมที่เขียนจะได้รูปแบบตามลำดับที่มีในทรานแซ็กชันที่เป็นข้อมูลเข้า

1.4 ผลที่คาดว่าจะได้รับ

1. เครื่องมือสำหรับวิเคราะห์หารูปแบบตามลำดับในทรานแซ็กชันที่สามารถทำงานได้
2. เพิ่มทักษะในการเขียนและพัฒนาโปรแกรม
3. รู้จักการวางแผนการทำงานและแก้ไขปัญหาที่เกิดขึ้น

บทที่ 2

การหากฎความสัมพันธ์ (Association Rule)

2.1 Link Analysis

Link Analysis เป็นการทำไมนิ่งกับข้อมูลที่ที่แตกต่างออกไปจากการทำไมนิ่งเพื่อทำนาย (prediction) และ การจัดกลุ่มข้อมูลในฐานข้อมูล (segmentation) เป้าหมายของการทำ Link Analysis จะทำเพื่อวิเคราะห์หาสิ่งที่เชื่อมถึงความสัมพันธ์ที่ร่วมกันที่มีภายในแต่ละ record หรือภายในกลุ่มของ record โดยมีเป้าหมายเพื่อหาความสัมพันธ์ร่วมกัน โดยเป็นความสัมพันธ์ระหว่างผลิตภัณฑ์ (product) หรือบริการ (service) ซึ่งเป็นแนวโน้มของผลิตภัณฑ์ที่ผู้บริโภคต้องซื้อร่วมกัน หรือซื้อเป็นลำดับในเวลาที่ต่างกัน ที่มีความถี่การเกิดความสัมพันธ์ถึงระดับที่กำหนด

ตัวอย่างของธุรกิจที่สามารถนำ Link Analysis ไปสนับสนุนได้คือ

- การทำ cross-selling เป็นการหาว่าเมื่อลูกค้าซื้อของหรือบริการอย่างใดอย่างหนึ่งแล้วจะมีการซื้อสินค้าหรือบริการอย่างอื่นร่วมด้วยหรือไม่ เช่นในธุรกิจธนาคาร ตัวอย่างธนาคาร Mellon ใน USA มีการนำดาต้าไม่นิ่งมาใช้เพื่อหารูปแบบการใช้บริการของลูกค้า ได้ว่าลูกค้าที่มีการฝากเงินมักเป็นผู้ที่สนใจที่จะหาเงินกู้เพื่อซื้อบ้าน และนำรูปแบบนี้ไปใช้เพื่อเสนอเงื่อนไขการกู้เงินให้แก่ลูกค้าที่ตรงตามรูปแบบข้างต้น
- หารูปแบบการเคลื่อนไหวของราคาหุ้น

Link Analysis ในที่นี้แบ่งเป็น 2 แบบคือ

1. การหากฎความสัมพันธ์ (Association Rule)
2. การหารูปแบบตามลำดับ (Sequential Pattern)

ข้อแตกต่างระหว่างทั้ง 2 คือ ในกรณีที่กำหนด 1 ทรานแซ็กชัน เป็นกลุ่มของ Item (โดยที่ item ในที่นี้หมายถึงสินค้า)ที่ลูกค้าซื้อในการไปร้านค้า 1 ครั้ง (1 ใบเสร็จ) การค้นหาความสัมพันธ์ร่วมกันจะวิเคราะห์สินค้าภายในแต่ละทรานแซ็กชัน โดยหาแนวโน้มของสินค้าที่ซื้อร่วมกัน โดยความถี่ของรูปแบบความสัมพันธ์ที่ได้จะต้องถึงระดับที่สนใจ โดยการนับความถี่จะนับต่อหนึ่งทรานแซ็กชัน ที่เกิดโดยไม่สนใจว่า ทรานแซ็กชันเกิดจากลูกค้าคนเดียวกันกระทำหรือเปล่า

การวิเคราะห์แบบนี้เรียกว่า market basket analysis และนำรูปแบบที่ได้ สร้างกฎที่มีความสัมพันธ์ นั้นคือ การหากฎความสัมพันธ์

การค้นหารูปแบบตามลำดับ (Sequential Pattern) จะใช้ในการหาความสัมพันธ์ข้ามทรานแซ็กชันที่เกิดคนละช่วงเวลาในช่วงเวลาที่กำหนด โดยข้อมูลที่ได้จะเป็นลำดับที่ลูกค้าซื้อสินค้าหรือบริการ (ไม่ได้จำกัดว่าต้องเกิดในทรานแซ็กชันเดียวกัน) เป้าหมายเพื่อให้เข้าใจพฤติกรรม การบริโภคของลูกค้าในระยะยาว ความถี่ของรูปแบบความสัมพันธ์ที่ได้จะต้องถึงระดับที่สนใจ โดยการนับความถี่จะนับต่อลูกค้าหนึ่งคน ไม่ใช่หนึ่ง ทรานแซ็กชันเหมือนการหากฎความสัมพันธ์ (association rule) ถ้ารูปแบบมีความถี่เกิดหลาย ๆ ครั้งในลูกค้าคนเดียวกัน ความถี่ที่นับได้ยังคงเป็นหนึ่งความถี่ การหารูปแบบตามลำดับจึงต้องพิจารณาด้วยว่า ทรานแซ็กชันที่เกิดนั้นเป็นของลูกค้าคนใด ทฤษฎีของการการูปแบบตามลำดับจะอาศัยทฤษฎีพื้นฐานที่สืบเนื่องกับการหากฎความสัมพันธ์ เช่นทฤษฎีสร้างรูปแบบตามลำดับของแต่ละความยาว และทฤษฎีของการตัดรูปแบบที่ไม่จำเป็นทิ้ง

2.2 การหากฎความสัมพันธ์ (Association Rule)

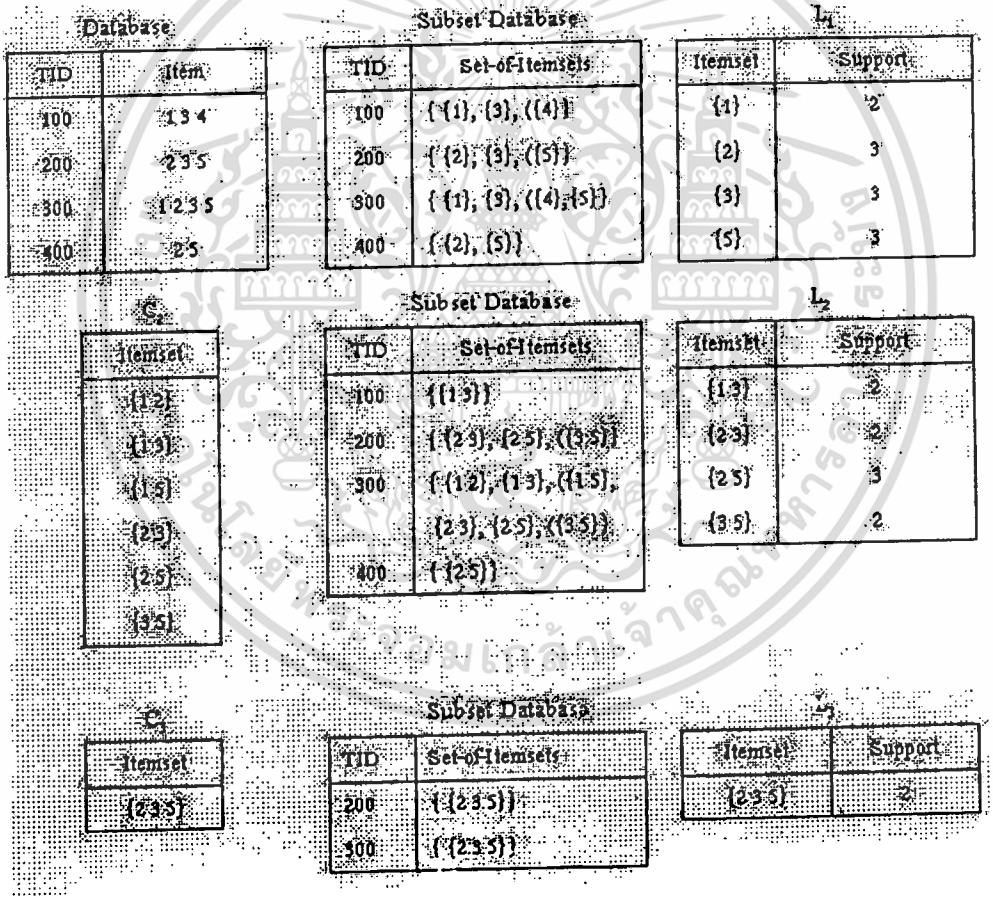
จะเป็นการหาทุกกลุ่มของสินค้า (set of items หรือ itemstes) ที่มีจำนวนของทรานแซ็กชันที่มีกลุ่มของสินค้าปรากฏอยู่มากกกว่าค่า minimum support โดยค่า support ของกลุ่มของสินค้าหมายถึงจำนวนของทรานแซ็กชัน ที่มีกลุ่มของสินค้าปรากฏอยู่ในทรานแซ็กชัน กลุ่มของสินค้าที่มีค่าถึง minimum support จะเรียกว่า large itemsets โดยที่กลุ่มของสินค้าที่เหลือเรียก small itemsets และนำ large itemsets และนำที่หาได้ไปสร้างเป็นกฎต่อไป

2.2.1 การหา large itemsets

อัลกอริทึมเพื่อหา large itemsets จะต้องทำงานหลายรอบกับข้อมูล โดยในรอบแรกเราจะทำการนับค่า support สำหรับแต่ละสินค้า เพื่อกำหนดว่าสินค้าใดเป็น large คือมีจำนวนทรานแซ็กชัน สินค้านั้นปรากฏอยู่มากกว่า minimum support และในรอบย่อยต่อ ๆ มาจะเริ่มทำการหากลุ่มของสินค้าที่จะ large itemsets โดยการใช้ข้อมูลจากรอบที่ผ่านมา เพื่อทำการสร้าง itemset ที่มีโอกาสจะเป็น large itemsets เรียกว่า candidate itemsets และทำการหา support สำหรับ candidate itemsets โดยการทำงานกับข้อมูล ที่จุดสุดท้ายของรอบนี้เราจะกำหนดได้ว่า candidate itemsets ใดที่ large ซึ่งจะกลายเป็น large itemsets ของรอบนั้น ๆ และจะนำ itemset เหล่านี้ไปใช้สำหรับการหา large itemsets ของรอบต่อไป กระบวนการเหล่านี้จะทำซ้ำไปเรื่อย ๆ จนกว่าจะไม่สามารถหา large itemsets ใหม่ได้

2.2.2 การสร้าง candidate itemsets

การสร้าง candidate itemsets (รอบ k)จะอาศัย large itemsets จากรอบก่อนหน้า (รอบ k-1) โดยอาศัยสมมุติฐานที่ว่า subset ของ large itemsets จะต้อง large ด้วย ดังนั้นเราจึงสามารถที่จะใช้ large itemsets ในรอบก่อนหน้า (L_{k-1}) นำสร้าง itemsets ที่มีความเป็นไปได้ที่จะ large ในรอบปัจจุบัน นั้นคือ candidate itemsets (C_k)รอบปัจจุบัน โดยการ join L_{k-1} ซึ่งเป็นของรอบก่อนหน้า และตัด itemset ที่ได้จากการ join ที่มี subset (ขนาด itemset= k-1) ที่ไม่ large (L_{k-1}) เราเรียกกระบวนการตัด subset ว่า prune และนำ itemset ที่ได้หลังการ join และ prune นำไปพิจารณาหาค่า support เพื่อกำหนดว่า candidate itemsets ใดที่จะเป็น la rge itensets สำหรับรอบนั้น ๆ



รูปที่ 2.1 แสดงตัวอย่างการหา Association Rule

ตัวอย่างของการทำงานแสดงโดยใช้ฐานข้อมูลในรูปที่ 2.1 สมมุติ minimum support ที่ 2 ทรานแซ็กชัน โดยในรอบแรกจะทำการหา L_1 (large item) คือหาว่าสินค้าใดบางที่มีค่าถึง minimum

support เราสามารถแจกแจงได้เป็น ฐานข้อมูลที่แสดงถึง itemset ที่มีในฐานข้อมูล โดยจำนวนของสินค้า (item)ที่อยู่ใน itemset จะสัมพันธ์กับรอบของการหา large itemset ในรอบแรกนี้คือการหาที่จำนวนสินค้าเป็น 1 โดยแสดงโดย subset database ในรอบแรกนี้เราสามารถหา L_1 ได้ดังที่แสดงในรูปที่ 2.1

จาก L_1 ที่หาได้ในรอบแรก นำ L_1 ที่ได้ไปสร้าง candidate itemsets สำหรับรอบต่อไป (รอบ 2) คือ C_2 ซึ่งเป็น itemsets ที่มีจำนวนสินค้าใน itemset = 2 ในรูปที่ 2.1 และได้ subset database ในรอบแรก เราสามารถหา L_2 (large itemsets ที่มีจำนวนสินค้าใน itemsets =2)ได้ และนำ L_2 ไปสร้าง candidate itemsets สำหรับรอบ 3 คือC_3 (มีจำนวนสินค้าใน itemset = 3) และพิจารณาฐานข้อมูลได้ L_3 (itemsets ที่มีจำนวนสินค้าใน itemsets =3)

จาก L_3 จะพบว่าเราไม่สามารถหา candidate itemsets สำหรับรอบต่อไปได้แล้ว กระบวนการสำหรับการหา large itemsets ลงที่รอบที่ 3 นี้

บทที่ 3

หลักการและทฤษฎีที่เกี่ยวข้องสำหรับ Sequence Pattern

การทำไมนิ่งกับฐานข้อมูลมักทำเพื่อการสนับสนุนการตัดสินใจ ด้วยเทคโนโลยี bar-code ทำให้การเก็บข้อมูลเกี่ยวกับการซื้อขายเป็นไปได้ เรียกข้อมูลพวกนี่ว่า basket data โดยแต่ละ record ในข้อมูลจะประกอบด้วยวันที่ทรานแซ็กชันเกิดขึ้น และ สินค้า(item) ที่ซื้อในทรานแซ็กชันนั้น ๆ เราจะทำการนำเสนอการใช้ไมนิ่งในการหารูปแบบตามลำดับกับข้อมูลเหล่านี้

การหารูปแบบตามลำดับเป็นการค้นหาความรู้ในฐานข้อมูลที่เรามี ในกรณีของฐานข้อมูลของการซื้อขาย เราสามารถนำการหารูปแบบตามลำดับเพื่อหาข้อมูลที่มีประโยชน์เกี่ยวกับพฤติกรรมของผู้บริโภค ผลที่ได้จะได้รูปแบบตามลำดับของสินค้าที่มีในทรานแซ็กชันที่เกิด โดยที่รูปแบบตามลำดับเหล่านี่สามารถนำมาใช้ประโยชน์ทางด้านการตลาด การวางแผนการขายและการวางแผนการจัดการสินค้าคงคลังเป็นต้น เช่นเมื่อพบว่ารูปแบบตามลำดับคือ A , K, C, D เมื่อ A K C และ D เป็นสินค้า (item) เราจะวางแผนการขายได้ว่า เมื่อมีการซื้อสินค้า A เกิดขึ้น เราจะนำเสนอสินค้า K, C หรือ D ให้ลูกค้าซื้อควบคู่กับไป เพราะเป็นสินค้าที่ลูกค้าโดยส่วนใหญ่มีแนวโน้มที่จะต้องซื้อ หรือวางแผนการจัดสินค้าคงคลังได้ว่าเมื่อมีการซื้อ A มาก ๆ ก็ควรจะตรวจสอบดูว่าสินค้า K , C และ D มีหรือไม่ เพียงพอหรือเปล่า เพราะเป็นสินค้าที่ลูกค้าน่าจะกลับมาซื้อ

3.1 ปัญหาของรูปแบบตามลำดับ

Problem Statement เมื่อมีฐานข้อมูลของการซื้อขายของลูกค้า (Customer Transaction) แต่ละ ทรานแอ็กชันประกอบด้วยฟิลด์คือ customer-id , transaction-time และ item ที่ทำการซื้อ โดยที่ไม่มี customer คนใดที่มี transaction มากกว่าหนึ่ง ณ transaction-time เดียวกับและไม่พิจารณาถึงปริมาณที่ซื้อในแต่ละ ทรานแอ็กชันโดยที่ $I = \{ i_1, i_2, \ldots I_m \}$ เป็นกลุ่มตามลำดับเรียก กลุ่มของ item (itemset) โดย item หมายถึงสินค้าแต่ละตัว และลำดับ (sequence) จะเป็นแถวของรายชื่อของ itemset

เมื่อมีฐานข้อมูล ปัญหาในการหารูปแบบตามลำดับคือการหาความสัมพันธ์ที่มีลำดับมากที่สุด (maximal Sequences) ที่เกิดขึ้นท่ามกลางลำดับที่มี ลำดับมากที่สุดพิจารณาจากลำดับที่มีค่า support ถึงค่าที่ผู้ใช้กำหนดเรียก minimum support (minsup) ค่า support ของลำดับเป็นอัตราส่วน

ของลูกค้าที่สนับสนุนลำดับนั้น แต่ละลำดับที่เป็นลำดับมากที่สุด(maximal sequence) จะแสดงถึงรูปแบบตามลำดับ (sequential pattern) และเรียกลำดับที่มีค่าถึง minsup ว่า large sequences

ลำดับ $(a_1, a2_2 .....a_n)$ จะอยู่ในลำดับอื่น $(b_1, b_2..... b_m)$ เมื่อ $i_1 <i_2< i_n$ ดังนั้น $a_1 \subseteq b_{i_1}$, $a_2 \subseteq b_{i_2}$,...,$a_n \subseteq b_{i_n}$ ตัวอย่าง ลำดับ <(3)(4 5) (8)> จะอยู่ในลำดับ <(7)(3 8)(9)(4 5 6)(8)> เมื่อ (3) $\subseteq$(3 8) , (4 5) $\subseteq$(4 5 6) และ (8) $\subseteq$(8) อย่างไรก็ตาม <(3)(5)> จะไม่อยู่ใน (3 5) เนื่องจาก <(3)(5)> แสดงให้เห็นว่า 3 ถูกซื้อก่อนและ 5 ซื้อถัดมาแต่ไม่ได้ซื้อในทรานแซ็กชันเดียวกัน ขณะที่ (3 5) ถูกซื้อต่อเนื่องในทรานแซ็กชันเดียวกัน จึงถือว่าไม่ใช้ลำดับแบบเดียวกัน ในกลุ่มของลำดับที่ได้ เราจะถือว่าลำดับ s นั้น *maximal* ถ้า s ไม่ได้อยู่ในลำดับอื่นเลย

ทุกทรานแซ็กชันของลูกค้าแต่ละคนเราสามารถมองเป็นลำดับเดียว โดยแต่ละทรานแซ็กชัน จะประกอบด้วยกลุ่มของสินค้า (item) โดยที่ลำดับที่ว่านี้เรียก customer-sequnce และลำดับของทรานแซ็กชันจะเพิ่มขึ้นตามเวลาที่เป็นลำดับ เมื่อ $T_1,T_2...T_n$ คือทรานแซ็กชันในแต่ละเวลา โดย item ที่อยู่ใน Ti แสดงโดย itemset(T_i) และ customer-sequnce สำหรับลูกค้าคือลำดับ <itemset(T_1) itemset(T_2) .... itemset(T_n)>

ลูกค้าจะสนับสนุน(support) ลำดับ s เมื่อ s อยู่ใน customer-sequence สำหรับลูกค้านั้น ๆ โดยที่ค่า support สำหรับลำดับกำหนดโดย อัตราส่วนของจำนวนลูกค้าที่สนับสนุนลำดับนั้น

ตัวอย่าง พิจารณารูปที่ 3.1 แสดงถึงฐานข้อมูลที่เก็บ customer-id, transaction-time และ item และในรูปที่ 3.2 แสดงฐานข้อมูลที่เป็นลักษณะของกลุ่มลำดับของ customer ถ้ากำหนดให้ minsup ที่ 25% นั่นคือต้องมีลูกค้าอย่างน้อย 2 คนที่สนับสนุน ทำให้ {(30)(90)} และ {(30)(40 70)}จะมากที่สุด ซึ่งจะเป็นรูปแบบตามลำดับ (Sequential Patterns) ลำดับ{(30)(90)} อยู่ใน customer 1และ 4 และ customer 4 จะเห็นว่า ซื้อ item (40 70) ในระหว่าง item 30 และ 90 แต่ก็ support รูปแบบ {(30)(90)} เนื่องจากไม่จำเป็นต้องต่อเนื่อง ในทำนองเดียวกันรูปแบบตามลำดับ {(30)(40 70)} พบใน customer 2 และ 4 โดยใน customer 2 จะมี item คั่นระหว่าง item (40) และ (70)

ตัวอย่างของลำดับที่ไม่ถึง minsup ลำดับ {1,2,3} ที่มี Customer-ID 2 เท่านั้นที่สนับสนุน ลำดับ {3},{4},{7},{9},{3.4}{3,7} และ {4,7} ถึงแม้จะมีค่า support ถึง minsup แต่มันก็ไม่ใช่คำตอบเพราะมันไม่ใช่ลำดับที่มากที่สุด

Customer ID	Transaction Time	Item Bought
1	June 25 '93	30
1	June 30 '93	90
2	June 10 '93	10, 20
2	June 15 '93	30
2	June 20 '93	40, 60, 70
3	June 25 '93	30, 50, 70
4	June 25 '93	30
4	June 30 '93	40, 70
4	June 25 '93	90
5	June 12 '93	90

รูปที่ 3.1 แสดงฐานข้อมูลที่เรียงตาม Customer Id และ Transaction Time

3.2 การหารูปแบบตามลำดับ

ความยาว(length)ของลำดับจะเป็นจำนวนของ itemset ที่อยู่ในลำดับ โดยที่ความยาวของลำดับ k เรียก k-ลำดับ (k-sequence) ค่า support ของ itemset i กำหนดโดยอัตราส่วนของ ลูกค้าที่ซื้อ item ใน i และ itemset ที่มี minimum support เรียก large itemset หรือ litemsets สังเกตว่าแต่ละ itemset ใน large sequences จะต้องมีค่าถึง minsup ดังนั้น large sequences จึงเป็นเป็นรายชื่อ (list) ของ litemsets

อัลกอริทึมสำหรับการค้นหา large itemset จะต้องต้องทำงานหลาย ๆ รอบกับข้อมูล โดยหลักการพื้นฐานคือในรอบแรกของการทำงาน จะเป็นการหาค่า support ของแต่ละ items และกำหนดว่า item ใดเป็น large item ซึ่งดูจาก minsup และในรอบต่อ ๆ มาเราจะหา itemset ที่ large โดยใช้ข้อ มูลจาก itemset รอบก่อน เราใช้ itemset รอบก่อนเพื่อสร้าง itemset ที่มีโอกาสเป็น large itemset เรียก candidate sequence และนับค่า support สำหรับทุก ๆ candidate sequence ที่ได้ เพื่อกำหนดว่า candidate sequence ใดที่เป็น large sequence ในรอบนั้น และเป็น litensets สำหรับการหา large itemsets ของรอบต่อไปและเป็น large sequence ในรอบนั้น การทำงานจะวนทำจนกระทั่งไม่พบ large itemsets ใหม่

candidate sequence ใหม่จะถูกสร้างโดยการสืบทอดจาก large sequence รอบก่อน เนื่องจากอาศัยสมมุติฐานว่า ส่วนย่อย (subset) ของ large sequence จะต้องมาจากส่วนที่ large ด้วย ดังนั้น candidate sequence ที่มี k-items จะถูกสร้างจากการสัมพันธ์(join) ของ large sequence ที่มี k-1 item และทำการตัด (prune) itemset k-item ที่ได้จากการสัมพันธ์ ที่ ส่วนย่อยของมันไม่ได้อยู่ใน large sequence ของ รอบที่ k-1

Customer Id	Customer Sequence
1	{ (30)(90) }
2	{(10 20)(30)(40 60 70}
3	{(30 50 70)}
4	{(30)(40 70)(90)}
5	{(90)}

รูปที่ 3.2 แสดงฐานข้อมูลลำดับของ Customer

(Customer Sequence Database)

Sequential Pattern with support > 25 %
{ (30)(90) }
{(30)(40 70)}

รูปที่ 3.3 แสดงเซตคำตอบ

3.2.1 The Algorithm

จะแบ่งปัญหาของการนำไมนิ่งหารูปแบบตามลำดับ เป็นขั้นตอนดังนี้

1. Sort Phase เป็นการเรียงข้อมูลของทรานแซ็กชัน ตาม ลำดับ customer Id เป็น key หลักและในแต่ละ customer Id และ item ตามลำดับของ Transaction Time ซึ่งก็คือเปลี่ยนจากฐานข้อมูลของทรานแซ็กชันเดิมเป็นฐานข้อมูลของลำดับลูกค้า(customer sequence database)
2. Litemset Phase ในขั้นตอนนี้จะทำการหา litemsets ทุก ๆ เซตของ litemsets สำหรับทุก ๆ ความยาวของลำดับในทรานแซ็กชันของลูกค้า เหมือนกับการทำ association rule โดยที่การทำ association rule การหาค่า support จะเป็นอัตราส่วนของ ทรานแซ็กชันที่มีการซื้อ itemset ในทุกๆ ทรานแอ็กชันที่เกิดขึ้น โดยไม่คำนึงถึงว่าเป็นลูกค้าคนเดียวกันซื้อต่าง ทรานแซ็กชัน กัน

ข้อแตกต่างหลักที่แตกต่างกับ association rule คือค่า support ที่นับสำหรับแต่ละ itemset นั้นจะนับเพิ่มขึ้นเพียงหนึ่งต่อลูกค้าหนึ่งคนถึงแม้ว่าลูกค้านั้นจะซื้อ itemset เดิมแต่ต่างทรานแซ็กชั่นก็ตาม จากฐานข้อมูลรูปที่ 3.1 เราหา litemsets คือ (30),(40), (70), (40 70)และ (90) ซึ่งเราจะ map litemsets เหล่านี้เป็นเลข integer ที่ต่อเนื่องกัน ซึ่งแสดงในรูปที่ 3.4 เพื่อให้ง่ายต่อการเปรียบเทียบและลดเวลาในการตรวจสอบว่ามีลำดับอยู่ใน customer sequence หรือเปล่า

Large Itemsets	Mapped To
(30)	1
(40)	2
(70)	3
(40 70)	4
(90)	5

รูปที่ 3.4 แสดง Large Itemsets

3. Transaction Phase จะเห็นได้จากส่วนที่ 3 (ที่จะกล่าวต่อไป) มีการเปรียบเทียบเพื่อหา large sequence การทำงานจะต้องทำงานวนหลาย ๆ รอบกับข้อมูล ดังนั้นเพื่อการทำงานที่เร็วขึ้นกับข้อมูลจึงมีการลดทอนข้อมูลบางส่วนที่พิจารณาแล้วว่าไม่จำเป็นจากฐานข้อมูลเพื่อให้ฐานข้อมูลมีขนาดเล็กลง ทำให้ข้อมูลที่จะนำมาพิจารณาใน sequence phase มีจำนวนลดลงด้วย แต่ละทรานแซ็กชัน ที่มี large item (จาก Litemset Phase) item ที่เป็น large item จะแทนที่ด้วยค่าตัวเลขและตัด item ที่ไม่มีใน large item ออกไปจากทรานแซ็กชัน นั้น ๆ ในทำนองเดียวกัน ทรานแซ็กชันที่ไม่มี large item เลย ก็จะถูกตัดออกไปไม่นำมาเกี่ยวข้อง

ฐานข้อมูลที่ได้จากการทำ Transaction Phase ที่จะใช้ใน Sequence Phase ผลที่ได้คือฐานข้อมูลที่มีจำนวนทรานแซ็กชัน และ item ลดลง ฐานข้อมูลใหม่ที่ได้จะทีเพียง ทรานแซ็กชันที่มี large item ที่ใช้สำหรับหารูปแบบตามลำดับอยู่เท่านั้นและจะไม่มี item ที่ไม่ใช่ large item การตัด item ที่ไม่ใช่ large item เป็นการลด item ที่ไม่จำเป็นลง เพราะ item เหล่านี้มีค่า support ต่ำกว่า minsup ถึงแม้นำมาใช้ ในรอบต่อ ๆ มาก็ไม่สามารถทำให้เกิด large item ได้ จำไม่จำเป็นต้องนำมาหา sequential pattern

ฐานข้อมูลของการเปลี่ยนรูปแบบ (Transformation database) แสดงในรูปที่ 3.5 โดยเปลี่ยนรูปแบบมาจากรูปที่ 3.2 ขณะที่ทำการเปลี่ยนรูปแบบ จะเห็นว่า item{10}และ{20}ของ Customer Id 2 จะถูกตัดถอนออกไปเนื่องจากมันไม่อยู่ใน litemset (จาก Litemset Phase)

Customer Id	Original Customer sequence	Transformed Customer sequence	After Mapping
1	{ (30)(90) }	{(30)} {(90)}	{1} {5}
2	{(10 20)(30)(40 60 70}	{(30)} {(40)(70)(40 70)}	{1} {2 3 4 }
3	{(30 50 70)}	{(30) (70)}	{1 3}
4	{(30)(40 70)(90)}	{(30)} {(40)(70)(40 70)} {(90)}	{1} {2 3 4} {5}
5	{(90)}	{(90)}	{5}

รูปที่ 3.5 แสดงฐานข้อมูลที่ทำ Transaction Phase

4. Sequence Phase ใช้กลุ่มของ litemset เพื่อหาลำดับที่ต้องการโดยอัลกอริทึมที่จะใช้สำหรับขั้นตอนนี้จะกล่าวถึงในส่วนที่ 3 .3
5. Maximal Phase เป็นการหาลำดับที่ยาวที่สุดที่ซ่อนอยู่ในกลุ่มของ large sequences (maximal sequences) ลำดับที่ maximal จะต้องไม่อยู่ในลำดับอื่น โดยที่อัลกอริทึมในส่วนที่ 3.3 บางตัวจะรวมส่วนนี้ไว้ใน Sequence Phase

For (k=n;k>1;1- -)

For each k-sequence L_k do

Delete all sequences L_k contain in L_i , i>k

ตัวอย่าง n= 3 และ L_4 = (1 2 3 4) , L_3 = (1 2 3)(1 2 4)(1 3 4)(1 3 5)(2 3 4)(3 4 5) ในการหา maximal sequence ของ large 3-sequence จะเหลือแต่ (1 3 5)(3 4 5) ที่เป็น maximal large 3-sequence itemsets อื่นจะตัดทิ้งเพราะมันเป็น subsequence ของ L_4

3.3 The Sequence Phase

เราจะนำเสนออัลกอริทึมสำหรับการหารูปแบบตามลำดับ 2 อัลกอริทึมที่คล้ายกันคือ count-all และ count-some โดย count–all Algorithm จะนับทุก ๆ large sequences ที่รวมถึงลำดับที่

ไม่ยาวที่สุด (non-maximal sequences) โดยลำดับที่ไม่ยาวที่สุดนี้จะถูกตัดออกในขั้นตอน Maximal Phase โดย count –all Algorithm นี้ชื่อว่า *AprioriAll*

และอีกอัลกอริทึม คือ count-some Algorithm ที่สนใจเฉพาะ ลำดับที่มากที่สุด(maximal sequences) จริงเท่านั้น โดยจะละทิ้งการนับลำดับที่มันเป็น subsequence ของลำดับที่ยาวกว่า ถ้าเราได้นับลำดับที่ยาวกว่านี้ไปแล้ว อย่างไรก็ตามเราระวังที่จะไม่นับ long sequence จำนวนมากที่ไม่มี minimum support อัลกอริทึมนี้ชื่อ *AprioriSome*

หมายเหตุ ในอัลกอริทึม L_k หมายถึงกลุ่มทั้งหมดของ large k – sequences , และ C_k เป็นกลุ่มของ candidate k – sequences

3.3.1 Algorithm AprioriAll

อัลกอริทึมแสดงในรูปที่ 3.6 แต่ละรอบเราจะใช้ large sequences จากรอบก่อนเพื่อสร้าง candidate sequences และวัดค่า support ของมัน ในขณะที่อ่านข้อมูลในฐานข้อมูลท้ายที่สุดของแต่ละรอบ ค่า support ของ candidates จะใช้กำหนด large sequences

ในรอบแรก ผลที่ได้จาก litemset phase จะใช้เป็นตัวเริ่มต้น ของกลุ่มของ large 1 – sequences

3.3.1.1 Apriori Candidate Generation

กระบวนการทำงานของการทำ apriori – generation จะอ้างถึง L_{k-1} ที่เป็นกลุ่มของ large (k-1) – itemset ทั้งหมด การทำงานจะทำงานจะทำตามขั้นตอนด้านล่าง โดยเริ่มจากขั้นตอน การสัมพันธ์ (join) โดยสัมพันธ์ L_{k-1} กับ L_{k-1} และ prune ที่ไม่มีโอกาสเป็น large ออก :

If k=2 then (first find C_k)

for every pair of e_1 and of e_2 And Form $C = (e_1\ e_2)$

For c_1 in F_{k-1}

search c_2 in F_{k-1} such that

deleat first event of c_1 and last event in c_2 yield same subsequence

Then $c = (c_1\ e)$

Next , in the *prune* step, we delete all itemsets $c \in C_k$ such that some (k – 1) – sebset of c is not in L_{k-1} :

Forall itemsets $c \in C_k$ **do**

Forall (k-1) –subsets s of c **do**

If ($s \notin L_{k-1}$) **then**

Delete c from C_k;

Example L_2={(1 2), (1 3) , (2 3), (34)} จะได้ผลการ join คือ (1 2 3) และ (1 3 4)

c_1=(1 2) join c_2= (2 3) give 3 sequence (1 2 3)

c_1=(1 3) join c_2= (3 4) give 3 sequence (1 3 4)

ทำการ Prune (1 3 4) ออกเพราะว่า (1 4) ไม่ได้เป็นsubsequence ใน L_2

3.3.1.2 ตัวอย่าง

พิจารณาฐานข้อมูลตามลำดับของลูกค้าแสดงในรูปที่ 3.7 ในที่นี้จะไม่แสดงฐานข้อมูลเดิมของมันในตัวอย่างนี้ โดยฐานข้อมูลตามลำดับของลูกค้าแต่ละทรานแซ็กชัน จะแทนที่ด้วยเซตของ litemset ที่อยู่ใน transaction และ litemsets เป็นเลข integer ค่า minimum support กำหนดให้เท่ากัน 40 % (ประมาณ 2 customer sequence)

ขั้นแรกหา litemset ใน litemset Phase ได้ large 1-sequence แสดงในรูปที่ 3.8 และ large sequences ในการทำรอบต่าง ๆ ของ Sequence Phase จนจบการทำงานที่รอบที่ k=4 ก็ได้แสดงในรูปเดียวกันนี้ (รูปที่ 3.8) เนื่องจากในรอบที่ k=5 ไม่มี candidate เกิดขึ้น และแสดง Candidate k - sequence ในรูปที่ 3.9 เมื่อทำ maximal Phase ได้ maximal large sequence 3 ลำดับคือ (1 2 3 4), (1 3 5) และ (4 5)

3.3.2 Algorithm AprioriSome

อัลกอริทึมใน forward pass เราจะทำการนับเฉพาะลำดับที่กำหนดไว้เท่านั้น เช่นตัวอย่างเราจะนับลำดับที่ความยาว 1, 2, 4 และ 6 ใน forward phase และจะทำการนับลำดับ3 และ 5 ที่เหลือใน backward phase โดยที่ ฟังชั่น next จะเป็นตัวคำนวณ parameter เพื่อกำหนดว่าในความยาวลำดับนั้นจะทำการนับ candidate sequences ในรอบนั้นหรือไม่ และทำการคืนค่าเพื่อใช้ในการคำนวณในรอบต่อไป

ดังนั้นฟังชั่นนี้จะเป็นตัวกำหนดว่าลำดับใดบางที่ควรจะนับ ซึ่งต้องให้เกิดความสมดุลระหว่าง เวลาที่เสียไปสำหรับการนับ non-maximal sequence กับการนับ small candidate sequence ที่เพิ่มขึ้นมา(เป็น C_kที่เกิดจาก C_{k-1} ไม่ได้สร้างจาก L_{k-1} ทำให้ในรอบที่ k มี candidate มากกว่าปรกติ)

```
        last =k;
        end
    end
// Backward Phase
for ( k- - ; k >=1; k - -) do
    if (L_k not found in forward phase ) then begin
        Delete all sequence in C_k contain in some L_i, i>k.
        foreach customer sequence in database after
            transformation phase do
                Increment the count of all candidates
                    in C_k that are contained in customer
                    sequence
            L_k = Candidates in C_k with minimum support.
        end
    else // L_k already know
        Delete all sequence in L_k contain in some L_i, i>k
Answer = U_k L_k;
```

3.3.2.1 ตัวอย่าง

จากฐานข้อมูลที่ใช้เป็นตัวอย่างสำหรับอัลกอริทึม AprioiAll เราได้ large 1-sequence (L_1) ในรูปที่ 8

จะทำ forward phase โดย next(last)= 2×last สร้าง C_2 จาก litem phase ในรอบแรก ค่า last ที่จะใส่ในฟังชั่น next คือ 1 ได้ f(1)=2 ดังนั้น k= last นับ C_2 และหา L_2 และ last =2 รอบที่สองสร้าง C_3 จาก L_2 และ f(2)=4 ดังนั้นไม่นับ C_2 (เว้นการหา L_3) และ last=2 รอบที่สามสร้าง C_4 จาก C_3 และ f(2)=4 ดังนั้นนับ C_4 และ last =3 รอบที่สี่ สร้าง C_5 แต่ไม่มี

หลังทำ forward phase แล้วจะเริ่มทำ backward phase ในรอบแรกนี้จะไม่ตัดอะไรออกจาก L_4 เลยเพราะไม่มีลำดับที่ยาวกว่า รอบต่อมา เราได้เว้นการนับ support ของ C_3 ใน forward phase หลังจากที่ทำการตัดลำดับใน C_3 (แสดงในรูปที่ 8) ที่เป็น subsequences ของ L_4 แล้ว เช่น เมื่อตัด subsequences ของ L_4 ออกไปจะเหลือลำดับ (1 3 5) และ (3 4 5) แล้วจึงหา L_3 รอบต่อมาลำดับใน L_2 จะตัดออกไปยกเว้น (4 5) เนื่องจากมันอยู่ในบางลำดับที่ยาวกว่า ด้วยเหตุผลนี้ทุกลำดับใน

L_1 จะถูกตัดออกทั้งหมด ทำให้ Large sequences ที่เหลือเป็น maximal sequences นั้นคือสำหรับอัลกอริทึมนี้จะทำ Maximal Phase ในตอน Sequence Phase เลย

บทที่ 4
อัลกอริทึม SPADE

จากบทที่ 3 อัลกอริทึมที่ได้นำเสนอไปแล้วนั้น พบว่ามีปัญหาในการทำงานบางประการ ในบทนี้จึงได้นำเสนออัลกอริทึมใหม่ที่มีประสิทธิภาพมากขึ้นคืออัลกอริทึม SPADE (Sequential Pattern Discovery using Equivalence classes)

ปัญหาที่พบจาก อัลกอริทึม ในบทที่ 3 คือ

- ในการทำงานมีการ scan ฐานข้อมูลหลายรอบ นั้นคือถ้าลำดับที่หาได้มีความยาวมากที่สุด k ดังนั้นจะต้อง scan ฐานข้อมูล k รอบ ซึ่งเสียเวลา และยิ่งจำนวนรอบมากขึ้น สินค้าที่พิจารณาก็จะลดลง จำนวนทรานแซ็กชัน และ ลูกค้าที่พิจารณาก็ควรลดลงด้วยในแต่ละรอบที่ทำงาน แต่ฐานข้อมูลที่ใช้ในแต่ละรอบยังใช้ตัวเดิมที่เป็นข้อมูลทั้งหมด
- โครงสร้าง hash tree ที่ใช้การเก็บ candidate sequences มีความซับซ้อนและการคำนวณตำแหน่งจากการ hash ที่ไม่ดีนัก

ข้อดีของอัลกอริทึม SPADE

- ใช้เพียงการ Join และการเปรียบเทียบข้อมูลเท่านั้น
- จำนวนรอบในการใช้งานฐานข้อมูลน้อยกว่า

SPADE จะแตกปัญหาออกเป็นปัญหาย่อย ๆ และแก้ปัญหาย่อยเหล่านั้นโดยอิสระจากกัน แต่ละส่วนของปัญหาย่อยจะกระทำในหน่วยความจำหลัก (main-memory)

เราได้นำเสนอการทำงานของ SPADE โดยนำมาใช้ในการค้นหารูปแบบตามลำดับโดยสนใจการขายปลีกหรือการวิเคราะห์ market-basket ซึ่งเป็นกลุ่มของสินค้า, กลุ่มของเรคคอร์ดที่ประกอบด้วยสินค้า และกลุ่มเรคคอร์ดของลูกค้าแต่ละคน เพื่อหาหรือกำหนดให้ได้ถึงลำดับที่เกิดขึ้นของสินค้าที่ซื้อโดยลูกค้าแต่ละคน

4.1 ตัวอย่างฐานข้อมูล

พิจารณาฐานข้อมูลดังรูปที่ 4.1 จะนำเสนอรูปแบบตามลำดับ ที่ได้จากการใช้ SPADE

ฐานข้อมูลประกอบด้วยสินค้า 8 สินค้า, ลูกค้า 4 คน และ ทรานแซ็กชัน ทั้งหมด 10 ทรานแซ็กชัน กำหนด minimum support ที่ 50% คือลูกค้า 2 คน โดยลำดับที่ได้มีดังนี้

F_1 = {(A)[4], (B)[4], (D)[2], (F)[4]}

F_2 = {(AB)[3], (AF)[3], (B->A)[2], (BF)[4], (D->A)[2],(D->B)[2], (D->F)[2], (F->A)[2]}

F_3 = {(ABF)[3], (BF->A)[2], (D->BF)[2], (D->B->A)[2], (D->F->A)[2]}

F_4 = {(D->BF->a)[4]}

เมื่อ " -> " หมายถึงมีการซื้อสินค้าคนละทรานแซ็กชัน เช่น D->B หมายถึง ซื้อ D แล้วจึงซื้อ B ในอีกทรานแซ็กชัน แต่ถ้าเป็น DF หมายถึง สินค้า D และ F ถูกซื้อภายใน เดียวกัน

Customer-Id	Transaction-Time	Items
1	10	C D
1	15	A B C
1	20	A B C
1	25	A C D F
2	15	A B F
2	20	E
3	10	A B F
4	10	D G H
4	20	B F
4	25	A G H

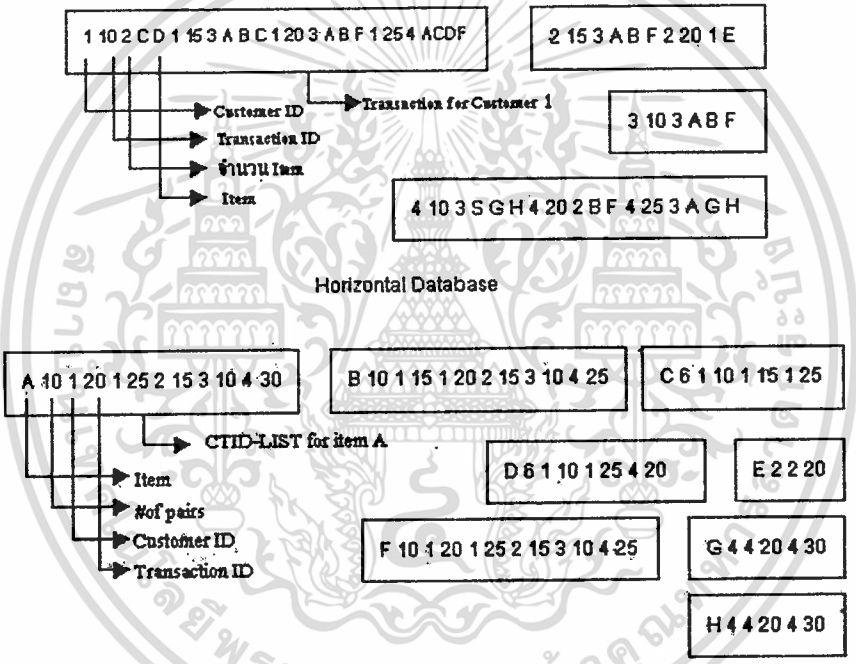
รูปที่ 4.1 แสดงฐานข้อมูลต้นแบบ

4.2 รูปแบบฐานข้อมูลและโครงสร้างลำดับ(Database Format and Sequence Structure)

อัลกอริทึม SPADE จะใช้รูปแบบของฐานข้อมูลที่ต่างจากอัลกอริทึมอื่น ๆ ที่กล่าวมา อัลกอริทึมที่กล่าวมานั้นจะใช้ฐานข้อมูลที่เป็นแบบ horizontal นั้นคือแต่ละทรานแซ็กชัน จะประกอบด้วย ตัวชี้เฉพาะลูกค้า (cid) , ตัวชี้เฉพาะทรานแอ็กชัน (tid) และจำนวนของสินค้า(item) ในทรานแซ็กชันและสินค้าต่างในทรานแซ็กชันนั้นๆ แต่SPADE จะใช้ฐานข้อมูลที่เป็นแบบvertical

โดยแต่ละสินค้าจะประกอบด้วยบัญชี (list) ที่เป็นคู่ของลูกค้าและตัวชี้เฉพาะทรานแซ็กชัน ที่สินค้านั้นเกิด นั้นคือบัญชีคู่ของ (cid, tid) โดยบัญชีนี้เรียก ctid-list ในรูปที่ 4.2 แสดงฐานข้อมูลที่เป็น horizontal และ vertical และแสดงถึง ctid-list สำหรับความถี่ของลำดับ 1 (frequent 1 sequences)

แต่ละลำดับใน SPADE ที่มี ctid-list จะเป็นอะเรย์ของสินค้าในลำดับ, ตัวนับค่า support และ sequence-template ที่ใช้สำหรับแยกความแตกต่างของแต่ละลำดับ ตัวอย่างเช่น ลำดับ (ABCF) , (A->BD->F) และ (A->B->D->F) ทั้งหมดมีสินค้าเดียวกัน {ABDF} แต่จะมี sequence-template ที่ต่างกัน คือ 000 , 101 และ 111 โดย bit ที่มีค่า " 1 " แสดงถึงความสัมพันธ์ที่ตามกันมาเกิดต่างทรานแซ็กชัน ("-> ") ในขณะที่ "0" แสดงถึงสินค้าที่ตามกันมาอยู่ในทรานแซ็กชันเดียวกัน



A	
CID	TID
1	
1	25
2	15
3	10
4	30

B	
CID	TID
1	15
1	20
2	15
3	10
4	25

D	
CID	TID
1	10
1	25
4	20

F	
CID	TID
1	20
1	25
2	15
3	10
4	25

CTID-List for Frequent Item

รูปที่ 4.2 แสดงรูปแบบฐานข้อมูลแบบ Horizontal และ Vertical

Horizontal และ Vertical รูปแบบ Horizontal ในขั้นตอนการนับ support ของ candidate sequence จะมี overhead เกิดขึ้น โดยเฉพาะแต่ละลำดับของลูกค้าในแต่ละรอบ k เราจะต้องตรวจสอบทุก ๆ k-subsequence ที่เกิดในกลุ่มของ candidate ที่เก็บในโครงสร้าง hash โดยรูปแบบแบบนี้จะสนใจการ scan ฐานข้อมูลที่เข้ามาในแต่ละระดับ

รูปแบบ Vertical ไม่จำเป็นต้อง scan ฐานข้อมูลทั้งหมดในการนับค่า support มันจะใช้การ intersect ของ ctid-lists ซึ่งมีข้อมูลเฉพาะส่วนที่สนใจจริง ๆ เท่านั้น เราไม่จำเป็ฯต้องใช้โครงสร้าง hash tree แต่รูปแบบ vertical ยังมีข้อจำกัดในการใช้รูปแบบนี้ในกรณีที่คำนวณ F_2 เพราะมันจะครอบครองพื้นที่ memory จำนวนมาก

4.3 Computing Frequent 1-Sequences (F_1)

จะใช้รูปแบบ vertical ที่แสดงในรูปที่ 4.2 โดยความถี่ของลำดับ 1 (frequent 1 sequences) จะคำนวณได้ในการ scan ฐานข้อมูลเพียงครั้งเดียว ได้ ctid-lists แต่ละสินค่า (item) เราจะทำการอ่าน ctid-list ของมันเพื่อนับค่า support สำหรับแต่ละสินค้า โดยอัลกอริทึม สำหรับการคำนวณ F_1 แสดงดังรูปที่ 4.3 Generate F_1 โดยที่ sequence-template ของ F_1 ยังไม่กำหนดค่า

```
Gen F1 (min_sup):
    For all database item i ∈ I do
        L = read_ctid_list(i);
        For all distinct cid ∈ L do i.sup=i.sup+1
        If (i.sup≥ min_sup) then F1 = F1 U {i};
```

รูปที่ 4.3 Generate F_1

4.4 Computing Frequent 2-Sequences (F_2)

ในส่วนนี้ประกอบด้วย 2 ขั้นตอนคือ

1. การเปลี่ยนจากรูปแบบที่เป็น vertical ให้กลับเป็น horizontal
2. ใช้รูปแบบใหม่ที่ได้ในการหา F_2

Database Inversion: วิธีการทำงานแสดงในรูปที่ 4.4 เมื่อรูปแบบ vertical ที่เป็น input แทนด้วย D, ฐานข้อมูลที่กลับรูปแบบแล้วแทนด้วย H_D และ H_D[k] หมายถึงกลุ่มของทรานแซ็กชัน ที่เป็นของลูกค้าคนที่ k แต่ละส่วนจะเป็นกลุ่มของรูปแบบ (item, tid) คือเป็น สินค้าและตัวชี้เฉพาะ

ทรานแซ็กชัน ที่สัมพันธ์กับสินค้านั้น แต่ละคู่ของ ctid-list (cid-tid) จะใช้ cid ในการคำนวณหา offset k เพื่อใช่ในการเพิ่มเข้าไปใน H_D[k] ในแบบคู่ของ (i, tid) รูปที่ 4.5 แสดงถึงฐานข้อมูลที่ทำการกลับรูปแบบแล้ว(H_D) ในฐานข้อมูลนี้จะต่างจากฐานข้อมูล horizontal ที่แสดงในรูป 4.2 สิ่งที่เป็นข้อแตกต่างที่น่าสนใจก็คือ สินค้าในฐานข้อมูลใหม่จะมีเฉพาะสินค้าเป็น F_1 เท่านั้น(1-sequence ที่ $\geq$ minimal support) ฐานข้อมูลจะมีขนาดเล็กลงและนำฐานข้อมูลนี้ไปใช้ในการหา F_2 ต่อไป

Compute F2: เราจะใช้ฐานข้อมูล horizontal ใหม่นี้ในการนับ support ของทุก ๆ 2-sequence เมื่อ $|F_1| = n$ และ $X, Y \in F_1$ เราจะทำการสร้างอะเรย์ S_1 ที่มีขนาด (n×n) สำหรับนับลำดับในรูปแบบลำดับ (X->Y) และอะเรย์อีกตัวสำหรับรูปแบบลำดับ (XY) ที่มีขนาด (n×(n-1))/2

```
Invert (D):
    For all frequent items i ∈F1 do
        L= read_ctid_list(i);
        For all (cid, itd) pair in L do
            k= cid-mincid;
            HD[k]= HD[k] U (i,tid)
```

รูปที่ 4.4 Invert Database

Cid	Size	(item,tid) pair
1	24	(A 15)(A 20)(A 25)(B 15)(B 20)(C 10)(C 15)(C 25)(D 10)(D 25)(F 20)(F 25)
2	8	(A 15)(B 15)(E 20)(F 15)
3	6	(A 10)(B 10)(F 10)
4	16	(A 25)(B 20)(D 10)(F 20)(G 10)(G25)(H 10)(H 25)

รูปที่ 4.5 แสดง Invert Database (H_D)

ด้วยลักษณะของฐานข้อมูลใหม่นี้ เราสามารถใช้เพียงการ join ตัวมันเองภายในแต่ละ cid เพื่อหา 2-sequence ก็สามารถทำได้ รูปที่ 4.9 แสดงขั้นตอนของการทำงานส่วนนี้ เมื่อเราเรียก function Contains() (กล่าวถึงต่อไปในส่วน 4.6) เพื่อใช้ในการนับค่า support ของลำดับ (XY), (X->Y) หรือ (Y->X) ถ้ามีลำดับเหล่านี้ปรากฏ โดยที่ลำดับเหล่านี้มี sequence-templates เป็น 0, 1 และ 1 ตามลำดับ

4.5 Computing frequent k-sequences, $k \geq 3$ (F_k)

4.5.1 Equivalence Class

เมื่อมีกลุ่มของความถี่ของลำดับ k (frequent k-sequence) F_k เราสามารถบอกได้ว่า 2 ลำดับเป็นส่วนหนึ่งของ equivalence class เดียวกันถ้ามันมีลำดับความยาว k-1 นับจากส่วนหน้า (prefix) ร่วมกัน นั้นคือเหมือนกันทั้ง items และ template

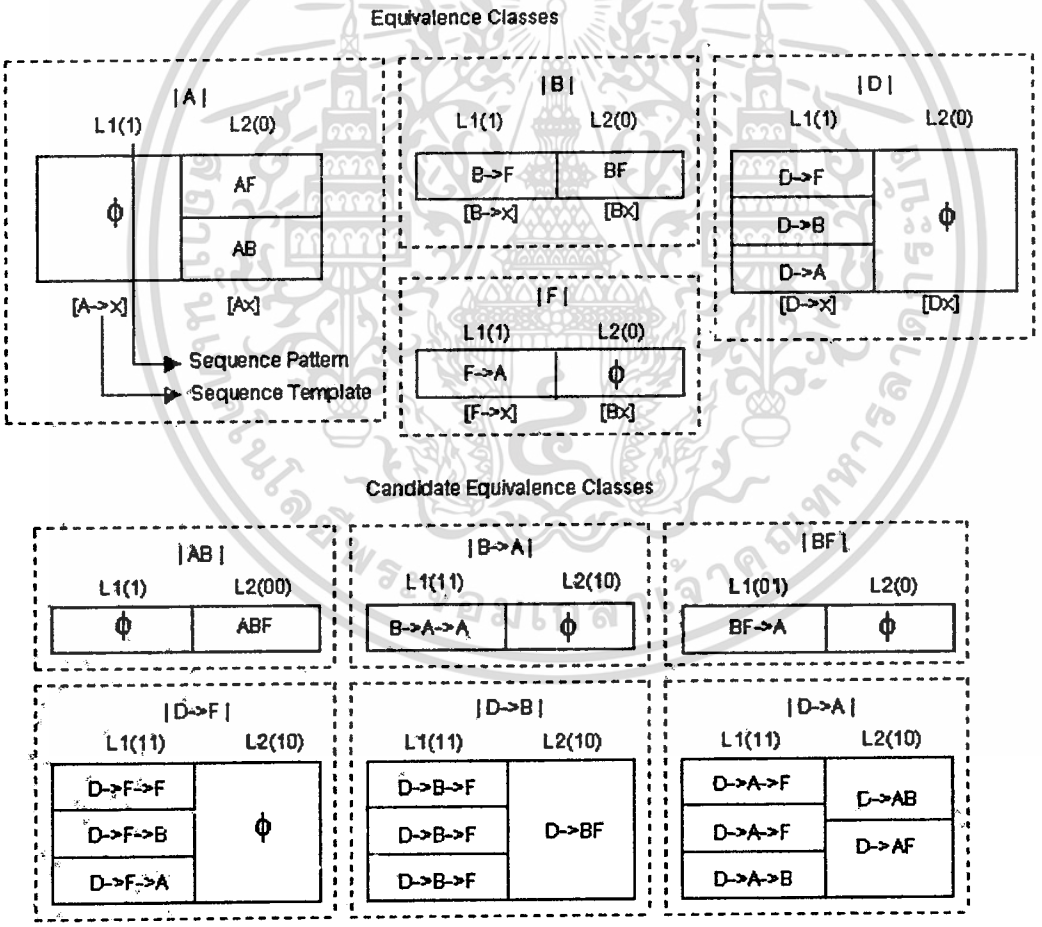
```
Comp F2 (min_sup, HD):
    For all C ∈ HD do
        For all distinct items X ∈ C do
            X.L = get_pair_with_item(X);
            For all distinct items Y ∈ C do
                Y.L = get_pair_with_item(Y);
                Contains (X, Y, S1[X][Y], S1[Y][X] S2[X][Y]);
    For all X, Y ∈ F1 do
        If (S1[X][Y]) ≥ mis_sup then F2 = F2 U {(X -> Y)};
        If (S1[Y][X]) ≥ mis_sup then F2 = F2 U {(X -> X)};
        If (Y > X and S2[X][Y]) ≥ mis_sup then F2 = F2 U {(XY)};
```

รูปที่ 4.6 Compute F_2

แต่ละ equivalence class ประกอบด้วยส่วนประกอบ 2 ตัวคือ $[\rho].l_1 = \{(\rho \rightarrow x)\}$ หรือ $[\rho].l_2 = \{(\rho \quad x)\}$ ซึ่งขึ้นกับรูปแบบของลำดับ พิจารณาจากตัวอย่างแสดงในรูป 3.7 แสดงถึง equivalence class ต่าง ๆ ที่สร้างจาก ([A], [B], [C], [D]) และ template ของมัน

การกำหนดแบบนี้เพื่อพยายามที่จะแบ่งแยกลำดับ k เป็นส่วน ๆ ที่อิสระต่อกันแยกจากกันอยู่คนละ equivalence classและลำดับที่มีความเหมือนกันอยู่ equivalence classเดียวกัน โดยที่ class [ρ] จะมีข้อมูลทุกอย่างเพื่อสร้างทุกลำดับที่มีส่านหน้า (prefix) ของลำดับ เป็น ρ การสร้าง candidate sequences ใหม่จะทำตามขั้นตอน 3 ขั้นดังนี้

1. Self-Join ($[\rho].l_1 \times [\rho].l_1$): เมื่อ $(\rho \rightarrow x) \in [\rho].l_1$ ในการสร้าง classes ที่ต่างกันเราต้องพิจารณาทุกคู่ของส่วนประกอบใน $[\rho].l_1$ นั้นคือ (ρ->x) และ (ρ->y) ด้วยการ intersection เพียงครั้งเดียวเราอาจจะได้ลำดับที่ต่างไปมากกว่าหนึ่งลำดับคือ (ρ->x->y), (ρ->y->x) หรือ (ρ->xy) และทำการเพิ่มลำดับเหล่านี้เข้าไปใน equivalence class ที่เหมาะสม โดยมี template เพิ่มจาก template เดิมของ ρ เพิ่มอีก 2 bit คือ “11”, “11” และ “10”



รูปที่ 4.7 แสดงถึง Equivalence Classes

2. Self-Join ($[\rho].l_2 \times [\rho].l_2$): เมื่อ $(\rho x) \in [\rho].l_2$ โดยต้องพิจารณาทุก ๆ คู่ของ (ρx) และ (ρy) การ join ของทั้งคู่จะสร้าง candidate ที่เป็นไปได้เพียงลำดับเดียวเท่านั้นคือ (ρxy) โดยมี template เพิ่มจาก template เดิมของ ρ เพิ่มอีก 2 bit คือ "00" และเพิ่มเข้าไปใน list ของ class $[\rho x].l_2$

3. Self-Join ($[\rho].l_2 \times [\rho].l_1$): เมื่อ $(\rho x) \in [\rho].l_2$ และ $(\rho->y) \in [\rho].l_1$ การ join ของทั้งคู่จะสร้าง candidate ที่เป็นไปได้เพียงลำดับเดียวเท่านั้นคือ $(\rho x->y)$ โดยมี template เพิ่มจาก template เดิมของ ρ เพิ่มอีก 2 bit คือ "00" และเพิ่มเข้าไปใน list ของ class $[\rho x].l_1$

4.5.2 Problem Decomposition using Equivalence Classes

อัลกอริทึม SPADE จะเริ่มด้วยการเรียกฟังชั่น $GenF_1$ และ $GenF_2$ เมื่อสิ้นสุดขั้นตอนนี้เราจะมีกลุ่มของลำดับ F_1 และ F_2 โดยที่ F_1 จะอยู่ใน equivalence class เดียว $[\phi]$ ไม่มีส่วนหน้า (prefix) และ F_2 แต่ละตัวจะอยู่ใน equivalence class $[C]$ เมื่อ $C \in F_1$ และ C เป็น ส่วนหน้าของลำดับ F_2 การแบ่งเป็น class แบบนี้จะเป็นการลด input ในการคำนวณหา candidate sequences ในรอบต่อไปให้น้อยลง เพราะเราสามารถคำนวณแยกกันในแต่ละ class

4.6 Ctid-List Intersection

```
Intersect (∝, β, φ_f , φ_r , φ_e)
    For all distinct cids C∝ ∈ ∝.ctid_list do
        For all distinct cids Cβ ∈ β.ctid_list do
            If (C∝ == Cβ ) then
                ∝.L = get_pairs_with_cid(C∝);
                β.L = get_pairs_with_cid(Cβ);
                Contains(∝, β, φ_f , φ_r , φ_e)
```

รูปที่ 4.8 Ctid-list Intersection

เราจะอธิบายการทำ ctid-list Intersection สำหรับ 2 ลำดับที่อยู่ใน equivalence class โดยการทำงานจะขึ้นกับรูปแบบของทั้ง 2 ลำดับที่พิจารณา ความเป็นไปได้ของ candidate ที่จะสร้างเป็น

ไปได้ 3 แบบด้วยกัน อย่างไรก็ตามการ intersection เพียงครั้งเดียวก็เพียงพอที่จะกำหนดให้ได้ถึงทั้ง 3 รูปแบบ

1. ถ้า 2 ลำดับที่พิจารณาคือ $\alpha=(\rho\text{->}x)$ และ $\beta=(\rho\text{->}y)$ โดยเราต้องทำการตรวจสอบ candidate ที่เป็นไปได้ทั้ง 3 แบบคือ $\phi_f=(\rho\text{->}x\text{->}y)$, $\phi_r=(\rho\text{->}y\text{->}x)$ และ $\phi_e=(\rho\text{->}x\ y)$
2. ถ้า 2 ลำดับที่พิจารณา คือ $\alpha=(\rho x)$ และ $\beta=(\rho y)$ โดยเราต้องทำการตรวจสอบ candidate $\phi_e=(\rho x\ y)$
3. สุดท้าย ถ้า 2 ลำดับที่พิจารณา คือ $\alpha=(\rho x)$ และ $\beta=(\rho\text{->}y)$ โดยเราต้องทำการตรวจสอบ candidate $\phi_f=(\rho x\text{->}y)$

โดยทั้ง 3 ข้อนี้จะสัมพันธ์กับกฎการสร้าง candidate ที่ได้เสนอไว้แล้ว วิธี intersection จะใช้ input 2 ลำดับและได้ candidate ที่เป็นไปได้ 3 แบบ คือ ϕ_f, ϕ_r และ ϕ_e ซึ่งบางแบบก็ไม่เกิดการเรียก ctid-list ซึ่งเป็นคู่ของ (cid, tid) ดูที่รูปที่ 4.2 การทำ intersection เราจะเริ่มด้วยการ scan ctid-list 2 ตัว และเรียก ฟังชั่น Contains มาทำงาน เพื่อหาว่า candidate ลำดับนั้นมีอยู่ในข้อมูลหรือเปล่า รูปที่ 4.9 แสดงอัลกอริทึมของการทำ intersection

ฟังชั่น Contain จะตรวจสอบและให้ลำดับที่มีอยู่ในทรานแอ็กชัน ของลูกค้า โดยเราให้α.L และ β.L ซึ่งเป็นคู่ลำดับของ (cid, tid) โดยที่มี cid จะเป็นค่าเดียวกัน โดยจะทำการตรวจสอบหาความสัมพันธ์ของ tid ระหว่างคู่ลำดับที่เข้ามา ความสัมพันธ์ระหว่าง tid คือ

1. equality (เกิดพร้อมกัน) แสดงโดย ϕ_e สำหรับการตรวจสอบเหตุการณ์ที่ (cid , tid) ตรงกันระหว่างคู่ลำดับ และจะทำการเพิ่มคู่ลำดับนี้ใน ϕ_e.ctid_list
2. follows (เกิดตามกัน) ประกอบด้วย 2 แบบคือ ϕ_fและ ϕ_r โดยที่ ϕ_r เราจะทำการเพิ่มรูปแบบที่เจอเข้าไปใน ϕ_f.ctid.L ซึ่งจะเป็นทุก ๆ tid ที่พบใน β.L ที่มากกว่า tid บางตัวใน α.L (cid ที่เหมือนกัน) และในกรณีของ ϕ_r กฎจะกลับกัน นั้นคือ เราจะใส่ทุก tid ใน α.L ที่มีค่ามากกว่า tid ใน β.L บางตัวเข้าไปใน ϕ. ctid.L

อัลกอริทึม ในส่วนนี้แสดงในรูปที่ 4.9 กระบวนการ intersection แสดงตัวอย่างให้เห็นในรูป 4.10 (การเปรียบเทียบ cid ที่เหมือนกันของ 2 ctid-list แสดงโดยกรอบเส้นประ)

Contains $(\propto, \beta, \phi_f, \phi_r, \phi_e)$

If $(\phi_f \neq \phi)$ **then**

For all tid $t_b \in \beta.L$ **do**

If $\exists$ tid $t_a \in \propto.L$ such that $t_b > t_a$ **then**

ϕ_f**.add_ctid_list** (cid, t_b) ;

If $(\phi_r \neq \phi)$ **then**

For all tid $t_a \in \propto.L$ **do**

If $\exists$ tid $t_b \in \beta.L$ such that $t_a > t_b$ **then**

ϕ_r**.add_ctid_list** (cid, t_a) ;

If $(\phi_e \neq \phi)$ **then**

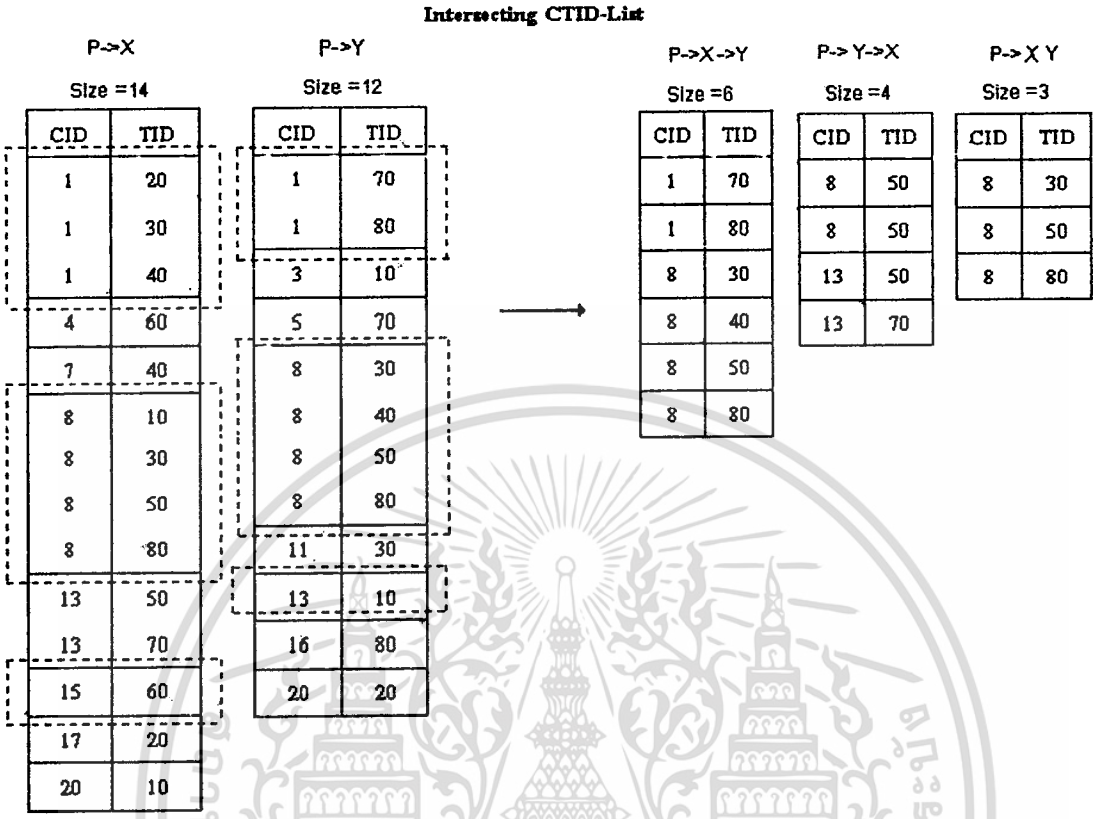
For all tid $t_b \in \beta.L$ **do**

If $\exists$ tid $t_a \in \propto.L$ such that $t_a = = t_b$ **then**

ϕ_e**.add_ctid_list** (cid, t_b);

รูปที่ 4.9 Containment Cheek

การ Join เพื่อให้การทำงานดีขึ้นและเร็วขึ้น เราจะนำค่า minimal support ที่ผู้ใช้เป็นคนกำหนด(min_sup) มาเกี่ยวข้องในขั้นตอนการ join ด้วย ตัวอย่าง ถ้าบอกว่า $\propto$ มีค่า support 120 (มันมีค่า cid ที่ต่างกัน 120 ค่า) และ β มีค่า support 100 โดยที่ min_sup =100 ในการเปรียบเทียบ ใน cit ที่เหมือนกัน เกราสามารถหยุดการ intersection ถ้าเราพบว่ามี 21 ค่าที่ไม่ตรงกันใน $\propto$ เนื่องจากผลของ support ของ support ที่เราจะได้มากที่สุดคือ 120-21 =99 จะเห็นว่าถึงแม้จะทำ intersection ไปจนจบก็ไม่มีทางถึง min_sup ที่ผู้ใช้กำหนด เราจึงไม่จำเป็นต้องทำจนจบ



รูปที่ 4.10 แสดง Ctid-list Intersection

4.7 Pruning Candidate

candidate sequence ความยาว k จะต้องมี subset ความยาว k-1 ของมันทุกตัวเป็น large k-1 sequence ถ้า subset ของลำดับนั้นทุกตัวเป็น เป็น large k-1 sequence ก็จะคง candidate นั้นไว้แล้วนำไปทำ ctid-list ภายหลัง ในทางกลับกันถ้าพบว่ามี subset ใดที่ไม่ได้เป็น large k-1 sequence ก็จะตัดลำดับนั้นออกจากการพิจารณา

4.8 สรุป

การใช้ SPADE จะทำให้การอ่านฐานข้อมูลมีจำนวนรอบที่ลดลง และทำให้การทำงานเร็วขึ้นสำหรับการหาลำดับเมื่อลำดับมีความยาวมากขึ้น

SPADE จะทำการแบ่งลำดับที่ได้ออกเป็นกลุ่ม ๆ คือการสร้าง Equivalence Class ทำให้ง่ายต่อการพิจารณาการสร้าง candidate และใช้ฐานข้อมูลแบบ vertical ทำให้ง่ายต่อการนับค่า support ของ candidate

บทที่ 5
การพัฒนาโปรแกรม

หลังจากได้ศึกษาถึงวิธีและขั้นตอนในการทำงานในดาต้าไมนิ่งมาใช้ในการหารูปแบบตามลำดับที่เกิดในฐานข้อมูล และอัลกอริทึมที่ใช้ในการทำงาน เราพบว่า SPADE เป็นอัลกอริทึมที่เหมาะสมในการที่จะนำมาใช้ในการเขียนโปรแกรมในการพัฒนาเครื่องมือของเรา ในส่วนบทนี้จะกล่าวถึงขั้นตอนการพัฒนาโปรแกรม

5.1 Visual Basic

การพัฒนาโปรแกรมจะใช้ภาษา Visual Basic ซึ่งเป็นภาษาคอมพิวเตอร์ที่นิยมนำมาใช้ในการพัฒนาโปรแกรมบน Windows เนื่องจากเป็นภาษาที่ใช้เทคโนโลยีในลักษณะ Visualize เพียงแต่เลือก Control ที่เหมาะสม แล้ววาดลงบน Form ก็สามารถสร้างที่ใช้ติดต่อกับผู้ใช้ได้ ทำให้งานที่เกี่ยวกับการทำ GUI (Graphic User Interface) เป็นไปได้ง่าย

ด้วยเวลาการทำงานจำกัด Visual Basic เป็นทางเลือกหนึ่งเนื่องจากระยะเวลาในการเรียนรู้การเขียนโปรแกรมน้อยกว่าภาษาคอมพิวเตอร์อื่นและมีความสามารถเพียงพอที่จะนำมาใช้ในการพัฒนาโปรแกรมในการพัฒนาระบบงาน

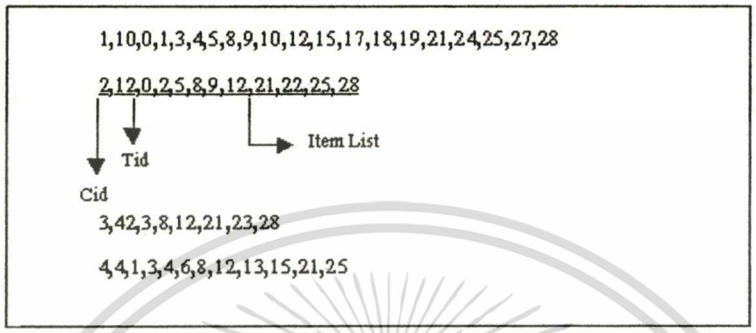
5.2 Input

ข้อมูลที่จะนำเข้าโปรแกรม สามารถนำเข้าได้ 2 แบบ

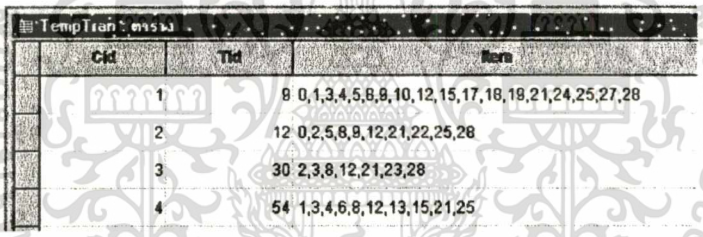
- ข้อมูลในรูปแบบ Text file
- ข้อมูลในรูปแบบฐานข้อมูล Access

1. ข้อมูลในรูปแบบ Text file ซึ่งประกอบด้วย Cid, Tid และ Item ในทรานแอ็กชันนั้น และต้องมีการกำหนดถึงตัวแบ่งสำหรับแต่ละค่าในทรานแอ็กชันนั้นๆ ลำดับของแต่ละทรานแอ็กชัน มีลำดับดังนี้คือ Cid, Tid และ Item-list ตามลำดับ แสดงตัวอย่างในรูปที่ 5.1 ใช้คอมม่าในการแบ่งข้อมูล

2. ข้อมูลในรูปแบบฐานข้อมูล Access ประกอบด้วย 3 field คือ Cid , Tid และ Item ในส่วน field จะประกอบด้วย item ทั้งหมดที่อยู่ในทรานแอ็กชันนั้น แต่ละ item จะใช้เครื่องหมายคอมม่า (“, ”) ในการแบ่ง แสดงตัวอย่างในรูปที่ 5.2



รูปที่ 5.1 ข้อมูลในรูปแบบ Text file



Cid	Tid	Item
1	8	0,1,3,4,5,8,9,10,12,15,17,18,19,21,24,25,27,28
2	12	0,2,5,8,9,12,21,22,25,28
3	30	2,3,8,12,21,23,28
4	54	1,3,4,6,8,12,13,15,21,25

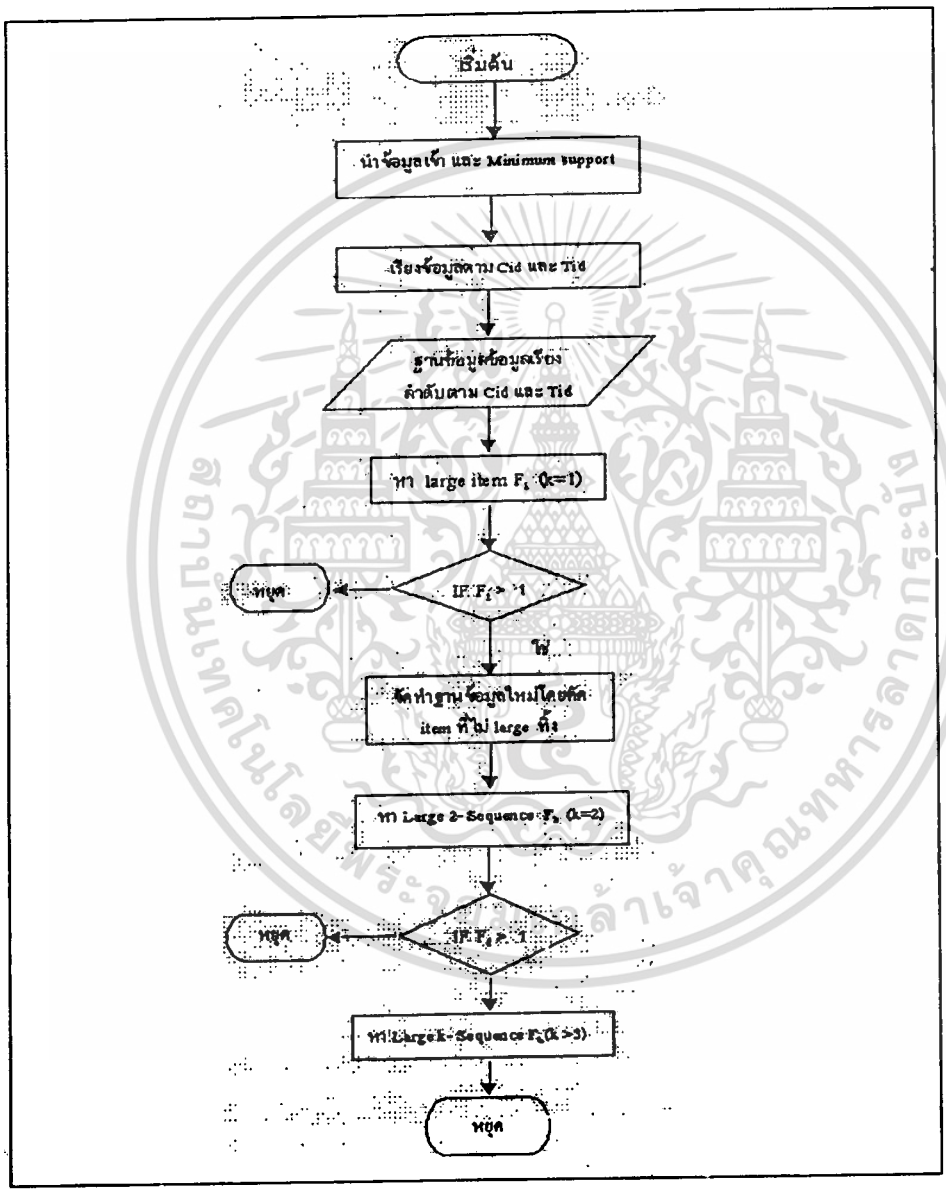
รูปที่ 5.2 ข้อมูลในรูปแบบฐานข้อมูล Access

5.3 โครงสร้างโปรแกรม

จากอัลกอริทึม SPADE ที่ได้กล่าวในบทข้างต้น ได้นำมา Implement เป็นโปรแกรม ที่ทำงานบนระบบปฏิบัติงาน Windows และต้องมี Ram อย่างน้อย 256 M เนื่องจากโดยโครงสร้างของอัลกอริทึม SPADE ต้องมีการสร้าง Equivalence Classและ ctid-list สำหรับแต่ละลำดับ (sequence) ที่หาได้ของรอบ k นั้น ๆ เพื่อความเร็วในการทำงานจึงต้องเก็บข้อมูลเหล่านี้ใน Memory การทำงานจะทำงานร่วมกับฐานข้อมูล Access เพื่อใช้ในการเรียงข้อมูลและเก็บรูปแบบตามลำดับ k- ลำดับที่หาได้ ไดอะแกรมการทำงานแสดงดังรูปที่ 5.3

การทำงานแบ่งเป็น 4 ส่วนใหญ่คือ

1. รับข้อมูลเข้าและค่า Minimum support จากผู้ใช้ โดยจะนำข้อมูลใส่ฐานข้อมูล Access
2. หา large item (F_1)
3. หา largc 2- Sequencc (F_2)
4. หา large k- Sequence (F_k)



รูปที่ 5.3 แสดงไดอะแกรมการหา Sequence Pattern

5.4 Output

โปรแกรมจะเก็บรูปแบบตามลำดับ (sequence pattern) ที่หาได้ไว้ในตารางของฐานข้อมูล Access ผู้ใช้งานโปรแกรมจะต้องใส่ชื่อฐานข้อมูลเอาต์พุต ผลของรูปแบบตามลำดับที่หาได้จะนำไปเก็บไว้ในตารางชื่อ "sq" ในฐานข้อมูลเอาต์พุต ตารางสำหรับเก็บรูปแบบตามลำดับประกอบด้วย 3 ฟิลว์ คือ

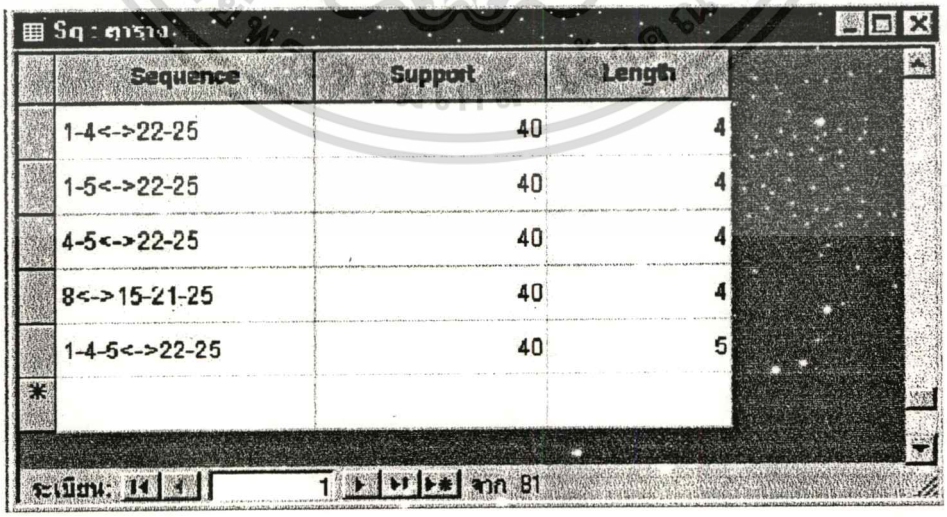
1. รูปแบบตามลำดับ โดยจะแสดงในรูปแบบที่ผู้ใช้เข้าใจง่ายขึ้น โดยจะแปรความหมายของค่า template ให้อยู่ในรูปแบบที่เข้าใจกว่าแบบตัวเลข ดังแสดงฐานข้อมูลเอาต์พุตดังรูป 5.4 โดยที่ " - " หมายถึงคู่สินค้าที่คั่นด้วย "-" เกิดในทรานแซ็กชั่นเดียวกัน และ " <-> " เป็นสัญลักษณ์แสดงถึงการเกิดข้ามทรานแซ็กชั่น

จากตัวอย่างสามารถแปรความหมายได้ว่า 1 – 4 <-> 22 - 25 นั้นคือการซื้อสินค้ามีการซื้อตามลำดับดังนี้คือซื้อสินค้า "1" และ สินค้า "4" ในทรานแซ็กชั่นและในทรานแซ็กชั่นอื่นถัด ๆ มาจะมีการซื้อสินค้า "22" และสินค้า "25" โดยที่ สินค้า "22" และสินค้า "25" ต้องซื้อในทรานแซ็กชั่นเดียวกัน

เราอาจจะพิจารณารูปแบบดังกล่าวไปปรับใช้ในการโปรโมตการขายได้ เช่น อาจนำเสนอการขายเป็นคู่ระหว่าง "1" กับ "4" หรือ "22" กับ "25" เพราะกลุ่มคนที่ซื้อ "1" และ "4" มีแนวโน้มที่จะต้องซื้อ "22" และ "25"ในคราวต่อไป เราจึงเหมือนนำเสนอให้ซื้อไปพร้อมกันเลย

2. Support บอกค่า support สำหรับรูปแบบตามลำดับ แสดงเป็นเปอร์เซ็นของจำนวนลูกค้า

3. ความยาวของรูปแบบ (จำนวนสินค้าในรูปแบบตามลำดับ)


Sq : ตาราง

Sequence	Support	Length
1-4<->22-25	40	4
1-5<->22-25	40	4
4-5<->22-25	40	4
8<->15-21-25	40	4
1-4-5<->22-25	40	5

ระเบียน: 1 จาก 81

รูปที่ 5.4 แสดงฐานข้อมูลเอาต์พุต

5.5 ขั้นตอนการหารูปแบบตามลำดับของโปรแกรม

ในการทำงานของโปรแกรมการหารูปแบบตามลำดับ โปรแกรมจะทำการสร้างฐานข้อมูลชื่อ "T ransaction" ในพาธเดียวกับที่โปรแกรมอยู่ ฐานข้อมูลนี้จะใช้ในการคำนวณของโปรแกรม เมื่อโปรแกรมทำงานเสร็จจะทำการสร้างฐานข้อมูลนี้ทิ้ง

ขั้นตอนการทำงานประกอบด้วย

(1) หา large item (F_1)

1.1 ทำการรับข้อมูลทรานแอ็กชันในการซื้อขายสินค้า ในขั้นตอนนี้จะใช้ฐานข้อมูลที่เป็นแบบ horizontal (หนึ่งทรานแอ็กชันหมายถึงหนึ่งใบเสร็จการซื้อ)และค่า minimum support โดยทรานแซ็กชันที่จะนำมาใช้งาน ต้องเป็นข้อมูลที่ได้ผ่านการเตรียมข้อมูลให้อยู่ในรูปแบบเหมาะสมสำหรับการทำงานของโปรแกรม ในกรณีถ้าข้อมูลของผู้ใช้เป็น Text File จะต้องทำการคอนเวิร์ตให้อยู่ในรูปแบบของฐานข้อมูล Access ก่อน

ตารางอินพุตประกอบด้วย 3 ฟิลด์

field	Type	Discription
Cid	Long	เก็บ Id ของ Customer
Tid	Long	เก็บ Id ของ Transaction
Item	memo	เก็บสินค้าที่ซื้อใน Transaction นั้น แต่ละเลขสินค้าแบ่งด้วย ","

1.2 หลังจากทำการอ่านข้อมูลเข้าเรียบร้อยแล้ว จะใช้คำสั่ง SQL เพื่อให้อ่านข้อมูลจากตารางในฐานข้อมูลที่เป็นอินพุตตามลำดับของ Cid และ Tid พิจารณาทีละ 1 ทรานแซ็กชัน พิจารณาสินค่าที่มีอยู่ในทรานแซ็กชันก็ทำการเพิ่มค่า support สำหรับสินค่านั้น 1 โดยจะต้องมีการตรวจสอบก่อนว่าสินค้านั้นเคยปรากฏ อยู่ในทรานแซ็กชันอื่นซึ่งเป็นของลูกค้าเดียวกัน (Cid เดียวกัน) มาก่อนหรือเปล่า ถ้าเคยเราก็ไม่ นับค่า Suppor เพิ่ม เพราะ support จะนับต่อลูกค้าหนึ่งค่าต่อลูกค่าหนึ่งคนที่ซื้อสินค้านั้น ไม่ได้นับตามจำนวนทรานแซ็กชันที่สินค้านั้นปรากฏ

1.3 เมื่อทำจนครบทุกทรานแซ็กชันแล้ว เราจะสามารถกำหนดได้ว่าสินค้าใดที่มีค่ามากกว่า minimum support ที่กำหนด สินค้านั้นจะ เป็น large item (F_1) หรือรูปแบบตามลำดับที่มีความยาว = 1 และเราจะนำสินค้าเหล่านี้ไป ใช้ในการหารูปแบบตามลำดับต่อไป

1.4 จาก F_1 ที่หาได้ เราจะทำการสร้างฐานข้อมูลใหม่ เพื่อใช้สำหรับการคำนวณหา F_2 โดยฐานข้อมูลใหม่นี้จะมีเฉพาะสินค้าที่มีสามารถทำไปสร้างรูปแบบตามลำดับได้ เพื่อเป็นการลดการอ่าน

ฐานข้อมูลลง ถ้าทรานแซ็กชันใดที่ไม่มีสินค้าที่มีรูปแบบตามลำดับความยาวหนึ่งเลย จะตัดทรานแซ็กชันทิ้ง จะเห็นได้ว่าเราสามารถลดขนาดฐานข้อมูลให้น้อยลงได้

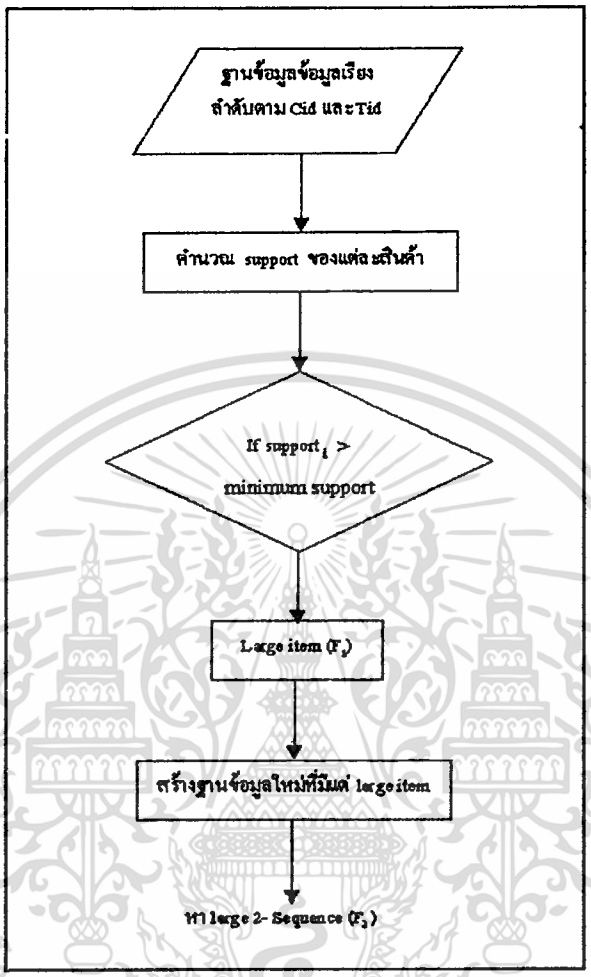
เราจะทำการอ่านฐานข้อมูลแต่ละทรานแซ็กชันตาม Cid และ Tid เหมือนในการหา F_1 ถ้าสินค้าใดไม่ large ก็จะทำตัดสินค้านั้นออกไป ถ้าสินค้าใดที่ large เราก็จะเก็บเข้าฐานข้อมูลใหม่ แต่รูปแบบการเก็บจะต่างออกไป คือจากฐานข้อมูลในข้อ 1 นั้น 1 ทรานแซ็กชันจะประกอบด้วยสินค้าที่เกิดขึ้นใน Tid เดียวกัน แต่ฐานข้อมูลใหม่นี้เราจะนำ ทรานแซ็กชันหลาย ๆ Tid ที่มี Cid เดียวกันมารวมกันให้เป็น 1 ทรานแซ็กชันเดียว และเพื่อให้สามารถพิจารณาได้ว่าสินค้าใน Cid นั้นเกิดจาก Tid ใด เกิดจากภายใน Tid เดียวหรือต่างกันเพื่อใช้ในการพิจารณาชนิดของรูปแบบตามลำดับที่จะเกิด จึงต้องทำการเพิ่มข้อมูลเพื่อจะสามารถระบุได้ว่าสินค้านั้นเกิดจาก Tid ใด โดยเราเพิ่ม Tid เข้าไปในแต่ละสินค้า คล้ายคู่ลำดับของสินค้ากับ Tid ตามที่ได้อธิบายในบทที่ 4 และแสดงในรูปที่ 4.6

จะเห็นได้ว่ามาถึงจุดนี้เราได้อ่านฐานข้อมูล 2 รอบ และได้ฐานข้อมูลใหม่ที่ได้ทำการต่อทรานแซ็กชันที่เป็นของ Cid เดียวกันให้เป็นทรานแซ็กชันเดียวกัน

TempTran : ตาราง

	Cid	Tid	Item
	1	1	48,55,71,72,85,104,115,131,138,145,151,154,183,198,236,275,279
	2	12	21,54,55,57,105,115,140,187,236,242,243,274,278
	3	26	21,25,68,90,93,96,100,116,131,154,168,249
	4	19	43,85,100,119,176,234,257,258,275
	5	100	15,52,69,96,136,142,180,196,201,287
	6	752	1,88,93,97,117,155,217,281
	7	81	41,45,71,72,214,227,239,296

รูปที่ 5.5 แสดงตัวอย่างตารางอินพุต



รูปที่ 5.6 แสดงไดอะแกรมหา large item (F_1)

(2) หา large 2- Sequence (F_2)

จากฐานข้อมูลที่ได้จากข้อ 1 นำมาหาลำดับ 2-sequence ซึ่งจะแบ่งเป็น 2 ขั้นตอนคือ

- **หา support ของลำดับ 2-sequence** โดยจะอ่านฐานข้อมูล เพื่อหาให้ได้ว่ามีลำดับ 2-sequence ใดบ้างที่เกิดในฐานข้อมูลและแต่ละลำดับที่ได้มี support เท่าไร เพื่อจะนำลำดับที่ถึง minimum support มาสร้าง ctid-list ของลำดับนั้น ๆ
- **สร้าง ctid-list สำหรับ large 2-sequence**

2.1 หา support ของลำดับ 2-sequence

การหาลำดับ 2-sequence จะพิจารณาที่ละ Cid จะทำการอ่านสินค้ามาทีละคู่ของ Cid เดียวกัน เพื่อนำมาพิจารณาตามกฎการสร้างลำดับที่ได้อธิบายในบทที่ 4 โดยพิจารณาจาก Tid ของทั้งคู่สินค้า คู่ของสินค้าที่อ่านมาจะเหมือนการจับ subset มาพิจารณาที่ละคู่ และทำการเพิ่มค่า support สำหรับลำดับที่พบ

ตัวอย่าง 1, 0.1, 1.1, 3.1, 4.1, 5.1, 18.1, 22.1001, 25.1001, 27.1001,

ในทรานแซ็กชันเป็นของ Cid = 1 ประกอบด้วยสินค้า 0, 1 , 3, 4, 5, 18, 22, 25 และ 27 โดยมี Tid คือ 1 และ 1001

คู่แรกที่อ่านมาได้คือ 0.1 และ 1.1 โดยจะใช้ "." ในการแบ่งสินค้ากับ Tid คู่แรกนี้ลำดับที่สร้างได้คือ 0,1 คือมีการซื้อสินค้า "0" พร้อมกับสินค้า "1" เพราะ Tid ของทั้ง 2 สินค้าเท่ากัน

คู่ 0.1 และ 22.1001 ลำดับที่สร้างได้คือ 0->22 คือ ซื้อสินค้า "0" และมีการซื้อสินค้า "22" ต่อมาแต่คนละทรานแซ็กชัน เพราะ Tid ของ "0" น้อยกว่า Tid ของ "22"

การทำงานในส่วนนี้จะเสียเวลามาก เพราะต้องพิจารณาที่ละคู่ของสินค้า โดยถ้ามีสินค้า n สินค้าสำหรับ Cid 1 คน ต้องมีการเปรียบเทียบ $C_{n,2}$ ครั้ง

ผลจากการเปรียบเทียบแต่ละคู่จะได้ลำดับ 2-sequence เราจะนำไปตรวจสอบกับตารางสำหรับเก็บ candidate ของ k=2 เพื่อตรวจดูว่าเคยมีลำดับนี้เกิดขึ้นหรือยัง ถ้าไม่ก็ทำการเพิ่มลำดับนี้และนับ support เป็น 1 และใส่ Cid ของทรานแซ็กชันนั้นลงในตารางนี้ที่ฟิลด์ Cid ด้วย เพื่อใช้ในการตรวจสอบการเพิ่มค่าค่า support ในทางตรงกันข้ามถ้าลำดับนั้นมีในตารางดังกล่าวแล้ว แล้วเราจะตรวจดูว่าลำดับนั้นเคยมีแล้วสำหรับ Cid เดียวกันหรือไม่โดยตรวจจากฟิลด์ Cid ในตารางเดียวกัน ถ้ายังก็ทำการเพิ่มค่า support แต่ถ้าเคยแล้วก็ไม่ต้องเพิ่ม support

เมื่อทำจนครบทุกทรานแซ็กชันแล้ว เราจะได้ F_2 ก็คือลำดับที่อยู่มีค่าในฟิลด์ support มากกว่าหรือเท่ากับค่า minimum support ที่ผู้ใช้กำหนด

2.2 สร้าง Ctid-list สำหรับ large 2-sequence

ในขั้นตอนนี้เราสามารถทำพร้อมกับขั้นตอนแรก จากการอ่านข้อมูลและจับคู่ subset เพื่อสร้างรูปแบบของลำดับ 2 –sequence ตามกฎนั้นในขั้นตอน 2.1 นั้น จะมีการจัดเก็บผลของการจับ subset และรูปแบบของลำดับ ที่ได้จากกฎ เพื่อนำมาใช้สร้าง ctid- list ของ F_2

เมื่อจบขั้นตอนนี้เราจะได้ F_2 และ ctid- list ของลำดับเหล่านั้น ซึ่งก็คือได้ฐานข้อมูลแบบ Vertical นั้นเอง โดยที่หนึ่ง ctid-list จะเท่ากับหนึ่งหนึ่งเรคคอร์ด ของฐานข้อมูลแบบ Vertical

ตัวอย่างของการสร้างฐานข้อมูลแบบ Vertical แสดงในรูปที่ 5.7 จากรูปมีลูกค้า 2 คน คือ Cid "1" และ "8" โดยที่พิจารณาตัวอย่างการสร้าง ctid-list สำหรับการ join สินค้า X และ Y เท่านั้น เราจะได้ candidate sequence ที่ต้องพิจาณาสร้าง ctid-list 3 แบบด้วยกันคือ (XY), (X->Y), (Y->X) การสร้างจะอาศัยตามกฎที่เคยกล่าวถึงบทที่ 4

1 (X 20)(X 30) (X 40)(Y 70) (Y 80)

8 (X 10)(X 30)(Y30)(Y 40)(X 50)(Y 50)(X 80)(Y 80)

X->Y Size =6

CID	TID
1	70
1	80
8	30
8	40
8	50
8	80

Y->X Size =4

CID	TID
8	50
8	50

X Y Size =3

CID	TID
8	30
8	50
8	80

รูปที่ 5.7 แสดงตัวอย่างการสร้าง Ctid-list

(3) หา large k- Sequence (F_k)

จะนำ F_1 และ F_2 มาสร้าง Equivalence Class และหา candidate F_3 ตามขั้นตอนที่ได้กล่าวในบทที่ 4 โดยการ join ภายในแต่ละ Equivalence class เพื่อสร้าง candidate F_3 ของ class นั้น และ prune candidate นั้น ถ้า candidate นั้นยังคงอยู่ก็ทำการ intersection เพื่อสร้าง ctid-list สำหรับ candidate นั้น ๆ และหา support ต่อไป เพื่อกำหนดให้ได้ว่า candidate ใดที่เป็น F_3 และสร้าง Equivalence และ ctid-list สำหรับการหา F_4 ต่อไป

การทำงานจะทำซ้ำไปเรื่อย ๆ ในการหาลำดับที่มีความยาวของลำดับต่อ ๆ มา ต่อไปจนไม่สามารถสร้าง Equivalence class ต่อได้

5.6 การทดสอบโปรแกรม

จากที่ได้กล่าวว่าได้แบ่งการทำงานออกเป็นส่วน ๆ ในการเขียนโปรแกรมจึงได้แบ่งส่วน

การเขียนโปรแกรมแบ่งตามการทำงานที่แยกกัน แล้วทำการแยกการทดสอบออกเป็นส่วน ๆ เมื่อทดสอบแล้วว่าแต่ละส่วนสามารถทำงานได้ จึงนำแต่ละส่วนมารวมกัน และทำการทดสอบอีกครั้ง และได้มีการแก้ไขปรับปรุงโปรแกรมเพื่อให้เหมาะสมต่อการทำงาน

ในการวิเคราะห์หารูปแบบตามลำดับที่เกิดในทรานแอ็กชันการซื้อขายสินค้า ข้อมูลที่จะนำมาใช้คือ ข้อมูลในการซื้อสินค้าในแต่ละใบเสร็จ โดยที่ในใบเสร็จในแต่ละใบต้องเก็บข้อมูลเกี่ยวกับผู้ซื้อไว้ด้วย ซึ่งในที่นี้อาจหมายถึงผู้ซื้อที่ใช่เครดิตการ์ดในการซื้อของ แต่ในความจริงแล้วข้อมูลเหล่านี้ถือเป็นความลับภายในองค์กร ไม่สามารถนำเปิดเผยภายนอกได้ จึงเป็นการยากที่จะนำข้อมูลที่เกิดขึ้นจริงเหล่านี้มาใช้ในการทำงาน

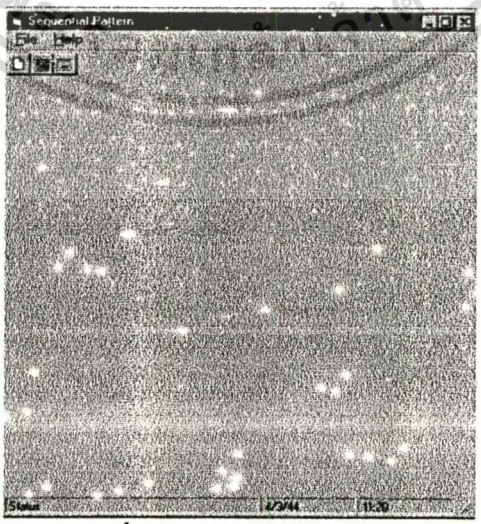
ในโครงงานพัฒนาระบบงานครั้งนี้ จึงได้ใช้โปรแกรมในการสร้าง dataset ของ IBM Quest data mining project ทำงานใน Unix เป็นโปรแกรมที่นำมาใช้อย่างกว้างขวางในกาวิจัยเกี่ยวกับการหารูปแบบตามลำดับ

ข้อมูลตัวอย่างของการทดลอง

- จำนวนของทรานแซ็กชัน 1,916350 ทรานแซ็กชัน
- จำนวนลูกค้า 400,00 คน
- จำนวนสินค้า 10,000 ชนิด
- Support 0.4 % หรือประมาณ 500 ลูกค้า

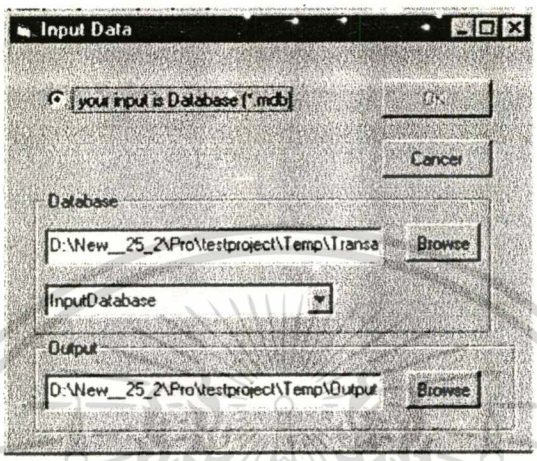
5.6.1 การทำงาน

- เมื่อเริ่มรันโปรแกรม แสดงหน้าจอการทำงานดังรูป 5.8



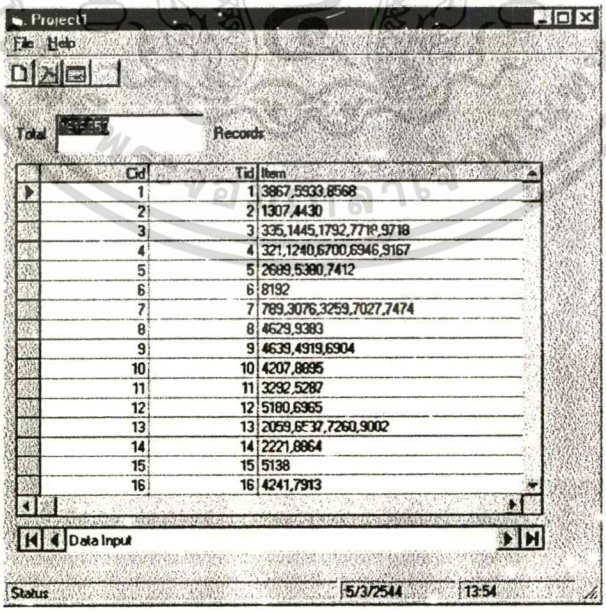
รูปที่ 5.8 แสดงหน้าจอหลัก

- เมื่อต้องการทำงาน จะทำการอินพุตข้อมูลเข้าและกำหนดฐานข้อมูลเอาร์พุต โดยการกดปุ่ม  หรือเลือกจากเมนู File > Open จะปรากฏหน้าจอการรับข้อมูลเข้าดังรูปที่ 5.9



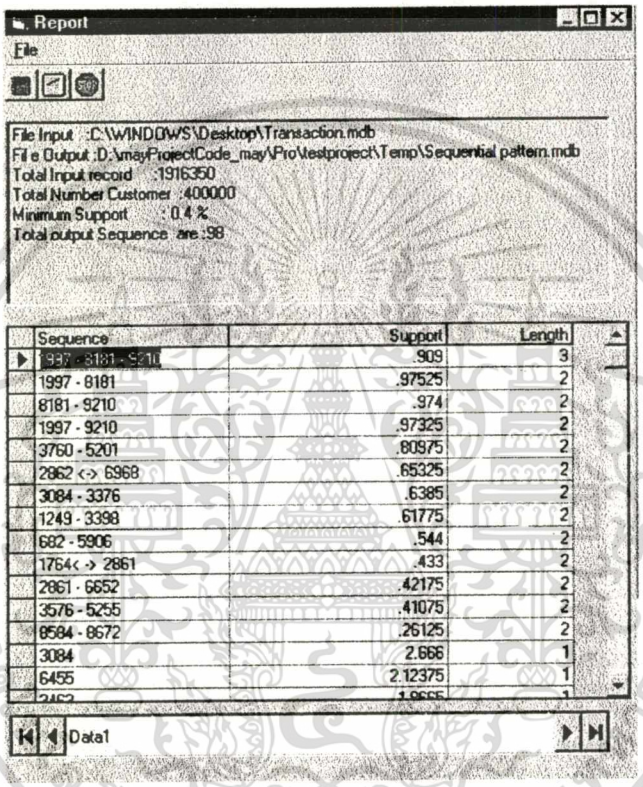
รูปที่ 5.9 แสดงหน้าจอการกำหนดฐานข้อมูลอินพุต และฐานข้อมูลเอาร์พุต

- หลังจากอินพุตข้อมูลเข้าและกำหนดฐานข้อมูลเอาร์พุตแล้ว จะกลับไปหน้าจอแรก ซึ่งจะแสดงฐานข้อมูลอินพุตและจำนวนเรคคอร์ดในฐานข้อมูลนั้น



รูปที่ 5.10 แสดงหน้าจอการกำหนดฐานข้อมูลอินพุต

- เมื่อต้องการจะทำการหารูปแบบตามลำดับ เราจะทำการกดปุ่ม  และจะปรากฏหน้าจอให้ผู้ใช้อินพุตค่า minimum support และเข้าสู่กระบานการหารูปแบบตามลำดับ
- การแสดงผลการทำงาน ผู้ใช้สามารถเลือกได้ว่าต้องการให้แสดงรูปแบบตามลำดับที่หาได้ทั้งหมด หรือแสดงเฉพาะความยาวที่เลือก



รูปที่ 5.11 แสดงหน้าจอการกำหนดฐานข้อมูลอินพุต

ผลการทดลอง

- Large item (F_1) หาได้จำนวน 84 สินค้า
- Large item (F_2) หาได้จำนวน 11 ลำดับ
- จาก F_2 ที่ได้นำไปสร้าง Candidate Sequence ความยาว = 3 และทำการนับค่า support ของ Candidate Sequence พบว่าไม่มี Candidate Sequence ใดที่มีค่าถึง minimun support การหารูปแบบตามลำดับจึงหยุดการทำงานส่วนนี้
- ดังนั้นรูปแบบตามลำดับที่หาได้ทั้งหมดจำนวน 95 ลำดับ

บทที่ 6
บทสรุป

6.1 สรุปผลการศึกษา

โปรแกรมสำหรับหารูปแบบตามลำดับของทรานแซ็กชันที่พัฒนาขึ้นมานั้น เป็นโปรแกรมสำหรับหาความสัมพันธ์ที่เกิดอย่างเป็นลำดับของแต่ละสินค้าที่เกิดจากการซื้อลูกค้าคนเดียวกัน โดยที่ข้อมูลที่จะนำมาใช้งานกับโปรแกรม จะต้องผ่านการเตรียมข้อมูลให้เป็นไปตามรูปแบบที่โปรแกรมกำหนดก่อน จึงจะสามารถนำมาใช้งานกับโปรแกรมได้

ในการศึกษานี้ การพัฒนาโปรแกรมสำหรับหารูปแบบตามลำดับของทรานแซ็กชัน ได้ประยุกต์นำอัลกอริทึม SPADE ซึ่งเป็นอัลกอริทึมที่ทำการแบ่งกลุ่มลำดับที่จะนำมาสร้างรูปแบบตามลำดับออกเป็นกลุ่ม ๆ เพื่อให้ง่ายต่อการพิจารณาการสร้าง candidate sequence และได้เสนอรูปแบบของฐานข้อมูลแบบ vertical ซึ่งทำให้ง่ายต่อการนับ support ของลำดับที่หาได้

ผลลัพธ์ที่ได้จากโปรแกรมจะได้รูปแบบตามลำดับ ซึ่งผู้ใช้งานจะต้องเป็นคนที่พิจารณาการที่จะนำผลของรูปแบบตามลำดับที่ได้นี้ไปใช้งานได้อย่างไรต่อไป โดยส่วนใหญ่แล้วการหารูปแบบตามลำดับมักนำไปใช้เพื่อประกอบการตัดสินใจ เช่น การตัดสินใจในการจัดโปรโมชั่นสินค้า เป็นต้น

6.2 ข้อเสนอแนะ

- ควรมีปรับปรุงการทำงาน หาวิธีในการคำนวณ F2 เพราะยังทำงานช้า ถ้าฐานข้อมูลมีขนาดใหญ่การคำนวณก็จะใช้เวลานานขึ้น
- ในการหารูปแบบตามลำดับ ลำดับที่เกิดจะเกิดมากหรือน้อย ขึ้นกับค่า support ที่ผู้ใช้กำหนด ถ้ากำหนดต่ำเกินไป F_1 ที่หาได้จะมีปริมาณมาก ทำให้ฐานข้อมูลที่ใช้เป็นฐานข้อมูลในหา F_2 มีจำนวนของสินค้าที่ต้องพิจารณา และ transaction ที่พิจารณามาก ทำให้การที่ต้องตัด subset มาพิจารณาสร้าง candidate ทีละคู่ของสินค้าของ F_2 ต้องใช้เวลามาก และได้ลำดับออกมามากมาย จนอาจไม่สามารถพิจารณานำไปใช้ประโยชน์ได้ การกำหนดค่า support ที่เหมาะสมจึงเป็นสิ่งที่ต้องพิจารณาด้วย

บรรณานุกรม

Mohammed J. Zaki.1997. **Fast Mining of Sequential Pattern in very large database.**The University of Rochester Computer Science Department, New York.

Mahesh Joshi, George Karypis and Vipin Kumer. 1999. **A Universal Formulation of Sequential Pattern** . www-users.cs.umn.edu/~mjoshi/hpdmtut/tsld138.htm. Expanded version available as Department of Computer Science University of Minnesota, Minneapolis.

R.Agrawal and R. Srikant . 1994. **Fast Algorithms for Mining Asociation Rule**. In Proc. Of the VLDB Conference, Santiago, Chile : Expanded version available as IBM Research Report RJ19839.

R.Agrawal and R. Srikant . 1995. **Mining Sequential Pattern**. IBM Almaden Research Center San Jose.

The Quest group. 2000. **Quest Synthetic Data Generation Code**. Available : http//www .almaden.ibm.com/cs/quest/syndata.html#instruction.

ประวัติผู้ชียน

ชื่อ นามสกุล	นางสาวนิรมล กลั่นเรืองแสง
สถานที่เกิด	นครสวรรค์
การศึกษา	วิศวกรรมศาสตร์บัณฑิต(คอมพิวเตอร์) มหาวิทยาลัยเชียงใหม่