การปรับปรุงการวิเคราะห์ขนาดความสามารถของเซอร์วิซด้วยการทำเหมืองกฎความสัมพันธ์

นายบวร เรืองแรงสกุล

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิทยาศาสตร์คอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2555

Improving an Analysis of Service Capability Granularity Using Association Rules Mining

Mr.Borvorn Ruengrangskul

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 2012

หัวข้อวิทยานิพนธ์	การปรับปรุงการวิเคราะห์ขนาดความสามารถของเซอร์วิซด้วยการทำเหมืองกฎความสัมพันธ์
โดย	นายบวร เรืองแรงสกุล
สาขาวิชา	วิทยาศาสตร์คอมพิวเตอร์
อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก	รองศาสตราจารย์ ดร.ทวิตีย์ เสนีวงศ์ ณ อยุธยา

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

.....คณบดีคณะวิศวกรรมศาสตร์
(รองศาสตราจารย์ ดร.บุญสม เลิศหิรัญวงศ์)

คณะกรรมการสอบวิทยานิพนธ์

.....ประธานกรรมการ
(รองศาสตราจารย์ ดร.วิวัฒน์ วัฒนาวุฒิ)

.....อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก
(รองศาสตราจารย์ ดร.ทวิตีย์ เสนีวงศ์ ณ อยุธยา)

.....กรรมการภายนอกมหาวิทยาลัย
(ผู้ช่วยศาสตราจารย์ ดร.เบญจพร ลิ้มธรรมาภรณ์)

บวร เรืองแรงสกุล : การปรับปรุงการวิเคราะห์ขนาดความสามารถของเซอร์วิซด้วยการทำเหมืองกฎความสัมพันธ์. (Improving an Analysis of Service Capability Granularity Using Association Rules Mining) อ. ที่ปรึกษาวิทยานิพนธ์หลัก : รศ. ดร. ทวิตีย์ เสนีวงศ์ ณ อยุธยา, 54 หน้า.

ปัจจัยหนึ่งที่ต้องคำนึงถึงในการออกแบบเว็บเซอร์วิซคือ หากโอเปอเรชันถูกออกแบบให้ทำงานได้ปริมาณน้อยในการเรียกใช้แต่ละครั้ง หรือเรียกว่ามีขนาดความสามารถเล็ก อาจทำให้ผู้ใช้บริการต้องเรียกใช้โอเปอเรชันต่าง ๆ ของเว็บเซอร์วิซนี้จากระยะไกลหลายครั้งกว่าจะบรรลุตามขอบเขตของงานที่ต้องการ และทำให้เกิดปริมาณการส่งข้อความในเครือข่ายเป็นจำนวนมาก งานวิจัยนี้นำเสนอการปรับปรุงการวิเคราะห์ขนาดความสามารถของเซอร์วิซด้วยการทำเหมืองข้อมูล จากเดิมซึ่งใช้อัลกอริทึมเอไพรออรีมาเป็นการใช้อัลกอริทึมเอไพรออรีเชิงทรัยโดยใช้ดับเบิลอาร์เรย์ทรัย ซึ่งทำให้สามารถหาลำดับของโอเปอเรชันที่ถูกเรียกต่อเนื่องกันบ่อยครั้ง และอาจเป็นสัญญาณว่า โอเปอเรชันเหล่านี้มีขนาดความสามารถเล็กเกินไป หากผู้ออกแบบเซอร์วิซสามารถปรับปรุงโดยรวมกลุ่มเป็นโอเปอเรชันเดียวกันได้ จะช่วยลดปริมาณการสื่อสารที่จะเกิดขึ้นต่อไป งานวิจัยนี้ทำการทดลองวิเคราะห์การเรียกใช้งานเว็บเซอร์วิซระบบซื้อสินค้าออนไลน์ ผลการวิเคราะห์จะได้เป็นกฎความสัมพันธ์ของลำดับการเรียกใช้โอเปอเรชันที่เกิดบ่อย รวมทั้งการแนะนำลำดับการเรียกใช้งานย่อยที่ปรากฏอยู่ในกฎความสัมพันธ์ส่วนใหญ่ ซึ่งสามารถรวมกลุ่มเข้าด้วยกันได้ จากการทดลองเปรียบเทียบกับวิธีเอไพรออรี พบว่าวิธีเอไพรออรีเชิงทรัยสามารถแนะนำลำดับโอเปอเรชันที่ควรจะรวมกันได้ดีกว่าวิธีการที่ผ่านมา อีกทั้งยังลดปริมาณลำดับของโอเปอเรชันที่แนะนำให้รวมกัน ซึ่งจะช่วยให้ผู้ออกแบบเซอร์วิซทำการตัดสินใจเลือกรวมกลุ่มโอเปอเรชันได้ง่ายขึ้น

ภาควิชา วิศวกรรมคอมพิวเตอร์ ลายมือชื่อนิสิต .....

สาขาวิชา วิทยาศาสตร์คอมพิวเตอร์ ลายมือชื่อ อ.ที่ปรึกษาวิทยานิพนธ์หลัก.....

ปีการศึกษา .....2555.....

5371465521 : MAJOR COMPUTER SCIENCE

KEYWORDS : Capability Granularity / Web Service / Association Rules Mining / Apriori / Trie

BORVORN RUENGRANGSKUL : IMPROVING AN ANALYSIS OF SERVICE CAPABILITY GRANULARITY USING ASSOCIATION RULES MINING. ADVISOR: ASSOC. PROF. TWITTIE SENIVONGSE, Ph.D., 54 pp.

An important issue in Web service design is service capability granularity. If a service operation is designed to have fine-grained capability with small function being achieved at each invocation, service consumers will have to remotely invoke several operations of the service for many times to achieve the work required. This can generate lots of message transmission in the network. This research presents an improvement of service capability granularity analysis by using trie-based Apriori association rules mining based on double-array trie, instead of the Apriori algorithm. Trie-based Apriori can discover frequent operation sequences which can be a sign that the operations in these sequences are too fine-grained. If the service designer can combine operations in a sequence into a single operation, it can help reduce invocation traffic. As an experiment, we analyze invocations to an online shopping Web service. The results are the association rules defining frequent operation sequences and a recommendation of the common sub-operation-sequences which can be combined. A comparison with the Apriori algorithm shows that trie-based Apriori can better suggest operation sequences and can reduce the number of recommended groups of operations to be combined. This will make it easier for the service designer when deciding to combine service operations.

Department : Computer Engineering Student's Signature .....

Field of Study : Computer Science Advisor's Signature .....

Academic Year : 2012

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จเรียบร้อยได้ด้วยดีเพราะได้รับคำแนะนำและคำปรึกษาจาก รองศาสตราจารย์ ดร.ทวิตีย์ เสนีวงศ์ ณ อยุธยา อาจารย์ที่ปรึกษาซึ่งเป็นผู้ให้ข้อเสนอแนะ และแนะนำแนวคิดที่เป็นประโยชน์ต่อการวิจัย จนเกิดเป็นวิทยานิพนธ์ฉบับนี้ขึ้น นอกจากนี้ขอขอบพระคุณประธานกรรมการสอบวิทยานิพนธ์ ได้แก่ รองศาสตราจารย์ ดร.วิวัฒน์ วัฒนาวุฒิ และกรรมการสอบวิทยานิพนธ์ ได้แก่ ผู้ช่วยศาสตราจารย์ ดร.เบญจพร ลิ้มธรรมาภรณ์ ที่ได้ชี้แนะถึงข้อบกพร่องและแนวทางแก้ไขต่างๆ

ขอขอบคุณ พ่อ แม่ ที่ให้การสนับสนุนการศึกษาในระดับอุดมศึกษา และขอขอบคุณกองทุนพัฒนาบุคลากรของกระทรวงการคลัง ที่ได้มอบทุนการศึกษาให้กับข้าพเจ้า ซึ่งสามารถช่วยลดภาระค่าใช้จ่ายในการศึกษาลงได้เป็นอย่างมาก

สุดท้ายนี้ ขอขอบคุณเพื่อน ๆ พี่ ๆ น้อง ๆ ทุกคนที่คอยให้กำลังใจ และให้คำปรึกษา ทำให้ประสบความสำเร็จในจุฬาลงกรณ์มหาวิทยาลัยแห่งนี้

สารบัญ

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

หน้า

สารบัญภาพ

หน้า

บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ปัจจุบันเว็บเซอร์วิซได้ถูกนำมาใช้งานอย่างแพร่หลาย ปัจจัยหนึ่งที่ต้องคำนึงถึงในการออกแบบเว็บเซอร์วิซคือ ความเหมาะสมของขนาดความสามารถของเซอร์วิซ (Service Capability Granularity) [**1**] ซึ่งหมายถึงขอบเขตการทำงานภายในแต่ละโอเปอเรชันของเว็บเซอร์วิซว่าครอบคลุมปริมาณงานมากน้อยเพียงใด ตัวอย่างเช่น โอเปอเรชัน CheckOrderItem ซึ่งมีขอบเขตการทำงานคือตรวจสอบรายการในใบสั่งซื้อ มีขนาดความสามารถเล็ก (Fine-Grained) กว่าโอเปอเรชัน ProcessPurchaseOrder ซึ่งทำการประมวลผลใบสั่งซื้อ การออกแบบให้ความสามารถของเซอร์วิซมีขนาดเล็กจะช่วยให้โอเปอเรชันของเซอร์วิซสามารถถูกเรียกใช้ในการประกอบเป็นเซอร์วิซอื่นได้ง่าย แต่ในขณะเดียวกันขนาดความสามารถที่เล็กเกินไปนั้น อาจทำให้ผู้ใช้บริการต้องทำการเรียกใช้งานหลายโอเปอเรชันของเซอร์วิซต่อเนื่องกัน กว่าจะบรรลุตามขอบเขตของงานที่ต้องการ จึงทำให้เกิดปริมาณการส่งข้อความในเครือข่ายเป็นจำนวนมาก งานวิจัยนี้เสนอการนำวิธีการทำเหมืองกฎความสัมพันธ์ (Association Rules Mining) ด้วยอัลกอริทึมเอไพรออรีเชิงทรัย (Trie-Based Apriori) ซึ่งใช้ในการหาลำดับของไอเท็มข้อมูลที่พบบ่อย (Frequent Item Sequence) [**2**] มาประยุกต์เพื่อวิเคราะห์การเรียกใช้โอเปอเรชันต่าง ๆ ของเว็บเซอร์วิซเพื่อหาลำดับของโอเปอเรชันที่ถูกเรียกต่อเนื่องกันบ่อยครั้ง (Frequent Operation Sequence) ซึ่งอาจเป็นสัญญาณว่าโอเปอเรชันเหล่านี้มีขนาดความสามารถเล็กเกินไป จึงทำให้มักถูกเรียกใช้ร่วมกันเสมอ หากผู้ออกแบบเซอร์วิซสามารถปรับปรุงโดยรวมกลุ่มเป็นโอเปอเรชันเดียวกันได้ จะช่วยลดปริมาณการสื่อสารที่จะเกิดขึ้นต่อไป ในการหาลำดับของโอเปอเรชันที่ถูกเรียกใช้บ่อยมีการประยุกต์ใช้ดับเบิล-อาร์เรย์ทรัย (Double-Array Trie) [**3**] เพื่อประสิทธิภาพในการจัดเก็บและค้นหาลำดับ ผู้วิจัยได้ทำการทดสอบวิธีการโดยวิเคราะห์การเรียกใช้งานเว็บเซอร์วิซระบบซื้อสินค้าออนไลน์ ผลที่ได้จะเป็นกฎความสัมพันธ์ของลำดับการเรียกใช้โอเปอเรชันที่เกิดบ่อย และมีการพิจารณาลำดับย่อยของการเรียกใช้งานโอเปอเรชันซึ่งปรากฏอยู่ในกฎความสัมพันธ์ส่วนใหญ่ (Common Sub-operation-sequence) เพื่อที่จะแนะนำผู้ออกแบบเว็บเซอร์วิซให้พิจารณาความเป็นไปได้ในการรวมกลุ่มโอเปอเรชันเหล่านี้

1.2 วัตถุประสงค์ของการวิจัย

เพื่อปรับปรุงวิธีการวิเคราะห์ขนาดความสามารถที่มีขนาดเล็กของเซอร์วิซด้วยวิธีการทำเหมืองกฎความสัมพันธ์ และปรับปรุงวิธีการแนะนำการรวมความสามารถขนาดเล็กเข้าด้วยกัน

1.3 ขอบเขตของการวิจัย

1.3.1 ปรับปรุงวิธีการวิเคราะห์ขนาดความสามารถที่มีขนาดเล็กของเซอร์วิซด้วยวิธีการทำเหมืองกฎความสัมพันธ์ โดยใช้เอไพรออรีเชิงทรัย

1.3.2 ปรับปรุงวิธีการแนะนำการรวมความสามารถขนาดเล็กเข้าด้วยกัน โดยคำนึงถึงลำดับของการเรียกใช้โอเปอเรชันเมื่อรวมกลุ่มกัน และการรวมกลุ่มจะไม่ส่งผลกระทบต่อประสิทธิภาพการทำงานหลังจากรวมกันแล้ว

1.3.3 ปรับปรุงเครื่องมือสนับสนุนการแนะนำผู้ออกแบบเซอร์วิซ

1.3.4 ทำการทดสอบกับกรณีของเซอร์วิซ Shopping เพื่อเปรียบเทียบกับผลการแนะนำเมื่อใช้วิธีของงานวิจัย [**4**] โดยจะมีการศึกษารูปแบบการเรียกใช้โอเปอเรชันต่าง ๆ ของเซอร์วิซและช่วงเวลาระหว่างการเรียกโอเปอเรชันเพื่อการพิจารณาความต่อเนื่อง จากการเรียกใช้งานของผู้ใช้จำนวนหนึ่ง และจะมีการทดลองปรับค่าพารามิเตอร์ต่าง ๆ ของการวิเคราะห์ เช่น ช่วงเวลาระหว่างการเรียกโอเปอเรชันเพื่อการพิจารณาความต่อเนื่อง ค่าสนับสนุนขั้นต่ำ และค่าความเชื่อมั่นขั้นต่ำเพื่อดูผลกระทบต่อผลการแนะนำผู้ออกแบบเซอร์วิซ

1.3.5 วิธีการที่เสนอสามารถใช้ได้กับเซอร์วิซทั่วไป แต่จะทดลองกับเว็บเซอร์วิซเพียงอย่างเดียว

1.3.6 การประมวลผลทรานแซกชัน จะพิจารณาเฉพาะการเรียกใช้งานในลักษณะลำดับเท่านั้น ไม่ได้พิจารณาการเรียกใช้งานแบบขนาน

1.3.7 การวิเคราะห์ขนาดความสามารถ จะทำการวิเคราะห์จากข้อมูลการเรียกใช้งานเท่านั้นโดยไม่คำนึงถึงผังการทำงานจริงของเซอร์วิซ

1.3.8 ผลจากการวิเคราะห์จะเป็นเพียงข้อแนะนำในการรวมกลุ่มโอเปอเรชัน การตัดสินว่าจะรวมโอเปอเรชันเข้าด้วยกันหรือไม่นั้น ให้เป็นไปตามการพิจารณาของผู้ออกแบบเซอร์วิซ

1.4 ขั้นตอนการทำวิจัย

1.4.1 ศึกษาการหากฎความสัมพันธ์ด้วยวิธีเอไพรออรีและเอไพรออรีเชิงทรัย

1.4.2 ศึกษาการวิเคราะห์ขนาดความสามารถเซอร์วิซ และวิธีการรวมกลุ่มเซอร์วิซ จากบันทึกการเรียกใช้งานเซอร์วิซ

1.4.3 กำหนดวิธีการหากฎความสัมพันธ์ด้วยวิธีเอไพรออรีเชิงทรัย และการรวมกลุ่มเซอร์วิซ

1.4.4 พัฒนาเครื่องมือสนับสนุนการแนะนำผู้ออกแบบเซอร์วิซ

1.4.5 ทดลองและประเมินผล

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1.5.1 ได้วิธีการในการวิเคราะห์เซอร์วิซที่มีความสามารถขนาดเล็ก และการรวมความสามารถเข้าด้วยกัน

1.5.2 ได้เครื่องมือแนะนำผู้ออกแบบเซอร์วิซในการพิจารณารวมโอเปอเรชัน เพื่อเพิ่มขนาดความสามารถของเซอร์วิซ

1.6 โครงสร้างของวิทยานิพนธ์

โครงสร้างของวิทยานิพนธ์ฉบับนี้ประกอบไปด้วย 5 บทหลักคือ บทนำ ทฤษฎีและงานวิจัยที่เกี่ยวข้อง แนวคิดการทำวิจัย ผลการทดลอง สรุปผลและแนวทางในการพัฒนาต่อ

ในบทแรกจะกล่าวถึงบทนำ วัตถุประสงค์ ขอบเขตในการวิจัย ขั้นตอนการวิจัย ประโยชน์ที่คาดว่าจะได้รับ โครงสร้างของวิทยานิพนธ์ และผลงานที่ตีพิมพ์จากวิทยานิพนธ์ ต่อมาในบทที่ 2 ได้กล่าวถึงทฤษฎี และงานวิจัยที่เกี่ยวข้อง บทที่ 3 ได้กล่าวถึงแนวคิดในการวิจัย ต่อมาในบทที่ 4 ได้อธิบายถึงการทดลองและผลการทดลอง และสุดท้ายในบทที่ 5 ได้อธิบายถึงบทสรุปของงานวิจัยและแนวทางในการพัฒนาต่อเพื่อปรับปรุงให้งานวิจัยมีประสิทธิภาพมากยิ่งขึ้นต่อไป

1.7 ผลงานที่ตีพิมพ์จากวิทยานิพนธ์

ส่วนหนึ่งของวิทยานิพนธ์นี้ได้รับการตอบรับให้ตีพิมพ์เป็นบทความทางวิชาการในหัวข้อเรื่อง “การวิเคราะห์ขนาดความสามารถของเซอร์วิซด้วยการทำเหมืองกฎความสัมพันธ์” โดย บวร เรืองแรงสกุล และ ทวิตีย์ เสนีวงศ์ ณ อยุธยา ในงานประชุมวิชาการ “The 9th National Conference on Computer and Information Technology: NCCIT2013” ณ มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ ระหว่างวันที่ 5 – 6 พฤษภาคม 2556

บทที่ 2

ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 ทฤษฎีที่เกี่ยวข้อง

2.1.1 ขนาดของเซอร์วิซในการออกแบบ

ขนาดของเซอร์วิซในการออกแบบ (Service Design Granularity) หมายถึง ขอบเขตหรือปริมาณงานที่เซอร์วิซครอบคลุมในการให้บริการ และเป็นประเด็นสำคัญหนึ่งที่ควรทำการพิจารณาตั้งแต่ขั้นตอนการออกแบบเซอร์วิซ ขนาดของเซอร์วิซในการออกแบบแบ่งออกได้หลายประเภท โดย T. Erl [**1**] ได้แบ่งไว้เป็น 4 ประเภทด้วยกัน

- ขนาดของเซอร์วิซ (Service Granularity) พิจารณาจากบริบทของการทำงานโดยรวมของเซอร์วิซ กล่าวคือ เซอร์วิซที่มีขนาดเล็กนั้นหมายถึงเซอร์วิซนั้นมีขอบเขตงานโดยรวมที่มีขนาดเล็ก

- ขนาดของความสามารถ (Capability Granularity) พิจารณาจากขอบเขตการทำงานของแต่ละโอเปอเรชันของเซอร์วิซ เช่น โอเปอเรชัน CheckOrderItem จะมีขนาดความสามารถเล็กกว่าโอเปอเรชัน ProcessPurchaseOrder เนื่องจากโอเปอเรชัน CheckOrderItem มีขอบเขตการทำงานเพียงแค่ตรวจสอบรายการในใบสั่งซื้อ ในขณะที่โอเปอเรชัน ProcessPurchaseOrder ทำการประมวลผลใบสั่งซื้อซึ่งมีขอบเขตการทำงานที่มากกว่า

- ขนาดของข้อบังคับ (Constraint Granularity) พิจารณาจากปริมาณข้อบังคับที่เกี่ยวข้องกับข้อมูลของโอเปอเรชันซึ่งต้องทำการตรวจสอบเมื่อข้อมูลเข้าออก ขนาดข้อบังคับที่หยาบหมายถึงมีจำนวนข้อบังคับที่น้อย หรือมีระดับการตรวจสอบที่น้อยในการเรียกใช้โอเปอเรชันของเซอร์วิซ

- ขนาดของข้อมูล (Data Granularity) พิจารณาจากปริมาณของข้อมูลที่ใช้ในการทำงานหรือการประมวลผล ในแง่ของเว็บเซอร์วิซนั้นหมายถึง ข้อมูลรับเข้า ข้อมูลผลลัพธ์ และข้อความแสดงความผิดพลาด ขนาดของข้อมูลที่เล็กนั้นหมายถึงมีปริมาณข้อมูลดังกล่าวน้อย

แม้ว่ายังไม่มีการกำหนดขนาดของเซอร์วิซที่เหมาะสมว่าควรมีขนาดเท่าใด แต่ T.Erl [**5**] ยังได้กล่าวถึงผลกระทบของขนาดต่อคุณภาพของเซอร์วิซ โดยเฉพาะในเรื่องความสามารถในการนำกลับมาใช้ซ้ำ (Reusability) และความสามารถในการประกอบ (Composability) (ซึ่งถือเป็นลักษณะหนึ่งของการนำกลับมาใช้ซ้ำ) กล่าวคือ ถ้าพิจารณาขนาดของเซอร์วิซจะพบว่า ขนาดของเซอร์วิซที่ใหญ่จะมีโอกาสนำเซอร์วิซกลับมาใช้ซ้ำได้ในหลาย ๆ งาน เพราะครอบคลุมความสามารถหลายอย่าง แต่ถ้ามีขนาดความสามารถใหญ่เกินไป ก็จะทำให้

โอกาสในการนำความสามารถนั้นไปใช้ซ้ำโดยประกอบเป็นเซอร์วิซอื่นลดลง เนื่องจากโอเปอเรชันมีความสามารถที่กว้างเกินไปกว่าความต้องการในบริบทอื่น ๆ

อย่างไรก็ตาม T. Erl [**1**] ได้กล่าวไว้เช่นกันว่า ความสามารถในการนำกลับมาใช้ซ้ำและความสามารถในการประกอบอาจไม่ได้เป็นความต้องการหลักในการออกแบบเซอร์วิซเสมอไป เช่น เป้าหมายของการออกแบบและพัฒนาเซอร์วิซอาจอยู่ที่การบูรณาการระบบ แม้ว่าเซอร์วิซจะมีขนาดประเภทต่าง ๆ ใหญ่หรือเล็กก็ตาม แต่หากสามารถช่วยในการบูรณาการระบบเข้าด้วยกันก็ถือว่าบรรลุเป้าหมายของการออกแบบแล้ว

2.1.2 การทำเหมืองกฎความสัมพันธ์

การทำเหมืองกฎความสัมพันธ์ (Association Mining) [**6**] เป็นกระบวนการหาความสัมพันธ์ระหว่างข้อมูลในเซตของข้อมูล นิยมใช้ในการวิเคราะห์ข้อมูลที่อยู่ในลักษณะของรายการทรานแซกชันในฐานข้อมูลขนาดใหญ่ เพื่อหาความสัมพันธ์ระหว่างข้อมูลหรือไอเท็มที่ปรากฏร่วมกันในทรานแซกชันเดียวกัน ความสัมพันธ์ระหว่างข้อมูลสามารถเขียนอยู่ในรูปของกฎดังนี้

$$X \rightarrow Y \,|X, Y \subseteq I \,, X \cap Y = \emptyset \tag{1}$$

โดยที่ I คือเซตของไอเท็มทั้งหมดในโดเมนที่สนใจ เช่น ข้อมูลชื่อสินค้าทั้งหมดที่วางขายในร้าน โดยเรียก X ว่าเซตที่เป็นเหตุ (Antecedent Set) และเรียก Y ว่าเซตที่เป็นผล (Consequent Set) จากกฎดังกล่าวสามารถแปลความหมายได้ว่า เมื่อพบข้อมูลในเซต X ในทรานแซกชันแล้ว จะพบข้อมูลในเซต Y ในทรานแซกชันนั้นด้วย เมื่อเซตข้อมูลมีขนาดใหญ่ จะทำให้ได้กฎความสัมพันธ์ออกมาเป็นจำนวนมาก จึงใช้เกณฑ์ในการคัดเลือกกฎโดยใช้ค่าความเชื่อมั่น (Confidence) และค่าสนับสนุน (Support)

- ค่าความเชื่อมั่น (Confidence) ของกฎความสัมพันธ์ $X \rightarrow Y$ นิยามได้โดย

$$conf(X \rightarrow Y) = \frac{sup\ (X \cup Y)}{sup\ (X)} \tag{2}$$

- ค่าสนับสนุน (Support) ของเซต X ใดๆ นิยามได้โดย

$$sup(X) = \frac{\text{จำนวนรายการทรานแซกชันที่มีข้อมูลในเซต } X}{\text{จำนวนรายการทรานแซกชันทั้งหมด}} \tag{3}$$

ตารางที่ 2.1 ตัวอย่างข้อมูลรายการทรานแซกชัน

Transaction ID	List of items
T1	Beef, Chicken, Milk
T2	Beef, Cheese
T3	Cheese, Boots
T4	Beef, Chicken, Cheese
T5	Beef, Chicken, Clothes, Cheese, Milk
T6	Chicken, Clothes, Milk
T7	Chicken, Milk, Clothes

จากตัวอย่างตามตารางที่ 2.1 เป็นบันทึกข้อมูลรายการทรานแซกชันการซื้อสินค้าของร้านค้าแห่งหนึ่ง มีทั้งหมด 7 ทรานแซกชันด้วยกัน หากเราต้องการทราบความสัมพันธ์ระหว่าง Beef และ Chicken ในลักษณะของ $\{Beef\} \rightarrow \{Chicken\}$ กล่าวคือ หากลูกค้าเลือกซื้อเนื้อวัวแล้ว ลูกค้าจะเลือกซื้อเนื้อด้วย จะพิจารณาได้ดังนี้

$$\sup(Beef \cup Chicken) = \frac{3}{7}$$

$$sup(Beef) = \frac{4}{7}$$

$$conf(Beef \rightarrow Chicken) = \frac{3}{4}$$

พบว่า ค่าความเชื่อมั่นของกฎความสัมพันธ์ $\{Beef\} \rightarrow \{Chicken\}$ เป็น 3 ทรานแซกชันใน 4 ทรานแซกชันที่พบ Beef จากทั้งหมด 7 ทรานแซกชัน

การค้นหากฎความสัมพันธ์เริ่มจากการที่ผู้ใช้งานกำหนดค่าสนับสนุนขั้นต่ำ (Minimum Support Threshold: minsup) เพื่อใช้หาเซตของข้อมูลที่พบบ่อย ซึ่งคือเซตของข้อมูลที่มีค่าสนับสนุนไม่น้อยกว่าค่าสนับสนุนขั้นต่ำ จากนั้นกำหนดค่าความเชื่อมั่นขั้นต่ำ (Minimum Confidence Threshold: minconf) และทำการค้นหากฎความสัมพันธ์จากเซตข้อมูลที่พบบ่อย ซึ่งเป็นกฎความสัมพันธ์ที่มีค่าความเชื่อมั่นไม่น้อยกว่าค่าความเชื่อมั่นขั้นต่ำ อัลกอริทึมที่นำมาใช้ในการค้นหากฎความสัมพันธ์ตามกระบวนการนี้คืออัลกอริทึมเอไพรออรี

เอไพรออรี (Apriori) [7] เป็นอัลกอริทึมที่ใช้ในการหาเซตของข้อมูลที่พบบ่อย (Frequent Itemsets) ที่นิยมใช้ในการหากฎความสัมพันธ์ โดยมีองค์ประกอบที่สำคัญได้แก่

- เซตของข้อมูลที่พบบ่อย (Frequent Itemsets): เซตของข้อมูลที่มีค่าสนับสนุนมากกว่าหรือเท่ากับค่าสนับสนุนขั้นต่ำ (แทนด้วยสัญลักษณ์ L_i สำหรับเซตข้อมูลขนาดเป็น i)
- คุณสมบัติเอไพรออรี (Apriori Property): ทุกเซตย่อย (Subset) ของเซตของข้อมูลที่พบบ่อย ต้องเป็นเซตข้อมูลที่พบบ่อยด้วย เช่น ถ้า {AB} เป็นเซตข้อมูลที่พบบ่อยแล้ว {A} และ {B} เป็นเซตข้อมูลที่พบบ่อยด้วย ดังนั้นสำหรับเซตข้อมูลที่ไม่พบบ่อยใด ๆ ซูเปอร์เซตทั้งหมดของเซตนี้จะเป็นเซตข้อมูลที่ไม่พบบ่อยด้วย

วิธีการค้นหากฎความสัมพันธ์คือ ทำการหาเซตของข้อมูลที่พบบ่อย ตั้งแต่ที่มีขนาดเป็น 1 จนถึงขนาดเป็น k ดังนี้

```
Ck: {candidate itemsets of size k}
Lk: {frequent itemsets of size k}

L1 = {frequent itemsets of size 1};
for (k = 2; Lk-1 != ∅; k++) do
  begin
    Ck = {candidate itemsets generated from Lk-1};
    for each transaction t in the database do
      increment the support count of all candidate
        itemsets in Ck whose items are found in t;
    Lk = {candidate itemsets in Ck whose support
      count ≥ minimum support threshold;}
  end
return L = ∪kLk
```

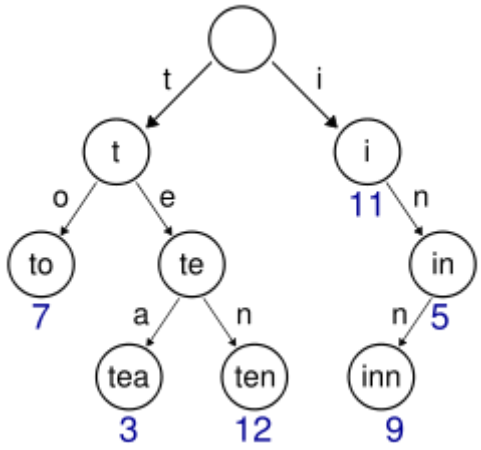
ภาพที่ 2.1 อัลกอริทึมเอไพรออรี

สำหรับอัลกอริทึมข้างต้น เซต C_k ที่มีขนาด k หาได้โดยการทำขั้นตอนการ join ตามด้วยการ prune ในขั้นตอนการ join ให้ทำการ join เซต L_{k-1} เข้ากับ L_{k-1} เพื่อให้ได้เซต C_k ก่อน จากนั้นตรวจสอบสมาชิกของ C_k (สมาชิกเป็นเซตขนาด k) ว่าทุกเซตย่อยขนาด k-1 ของสมาชิกนั้นอยู่ในเซต L_{k-1} หรือไม่ (คือเซตย่อยนั้นเป็นเซตข้อมูลที่พบบ่อยหรือไม่) หากมีบางเซตย่อยไม่อยู่ในเซต L_{k-1} แสดงว่าสมาชิกขนาด k ตัวนั้นจะไม่เป็นเซตข้อมูลที่พบบ่อยด้วย และให้ prune สมาชิกตัวนั้นออกจาก C_k

การสร้างกฎความสัมพันธ์สามารถสร้างได้จากทุกเซตย่อย (s) ของเซตข้อมูลที่พบบ่อย I โดยที่ I เป็นสมาชิกของ L และ s ไม่ใช่เซตว่างและไม่ใช่ I จะได้กฎความสัมพันธ์ s → I-s ถ้าค่าความเชื่อมั่นของ s → I-s หรือ $\frac{\sup(I)}{\sup(s)}$ ไม่น้อยกว่าค่าความเชื่อมั่นขั้นต่ำ

2.1.3 โครงสร้างข้อมูลแบบทรัย

ทรัย (Trie) [**8**] เป็นโครงสร้างข้อมูลต้นไม้ลำดับ นิยมใช้จัดเก็บข้อมูลสตริงที่ข้อมูลแต่ละตัวมีค่าไม่เหมือนกัน ซึ่งแตกต่างจากโครงสร้างต้นไม้ทวิภาคตรงที่โหนดไม่จำเป็นต้องเก็บค่าคีย์ไว้ แต่ตำแหน่งของโหนดในโครงสร้างต้นไม้สามารถแสดงคีย์ที่โหนดนั้นมีความสัมพันธ์ด้วย โหนดลูกทั้งหมดของแต่ละโหนดล้วนมีสตริงขึ้นต้น (Prefix) ที่เหมือนกับข้อมูลที่เก็บในโหนดนั้น ทรัยเป็นโครงสร้างข้อมูลที่รองรับการดำเนินการค้นหา แทรก และลบได้อย่างมีประสิทธิภาพ



ภาพที่ 2.2 ตัวอย่างทรัย

ภาพที่ 2.2 แสดงโครงสร้างทรัย ที่เก็บค่าคีย์ to, tea, ten, i, in และ inn โดยโหนดแต่ละโหนดแทนค่าท่องผ่าน (Traverse) จากโหนดรากมายังโหนดนั้น ๆ มีเฉพาะโหนดใบและโหนดภายในบางโหนดเท่านั้นที่สามารถอ้างถึงค่าคีย์ที่จัดเก็บอยู่ได้ ค่าที่แสดงภายในโหนดในรูปเพียงแค่ใช้อธิบายการทำงานของทรัยเท่านั้น

ลักษณะเด่นของโครงสร้างข้อมูลแบบทรัยพิจารณาได้ดังนี้

- สามารถรองรับข้อมูลที่มีลำดับได้หลายความยาว
- การเพิ่มหรือลบข้อมูลทำได้ง่าย
- มีความเร็วในการจัดเก็บและการเข้าถึงข้อมูล ในกรณีใช้เวลามากที่สุดจะเป็น $O(m)$ โดยที่ m คือความยาวของคีย์ในทรัย

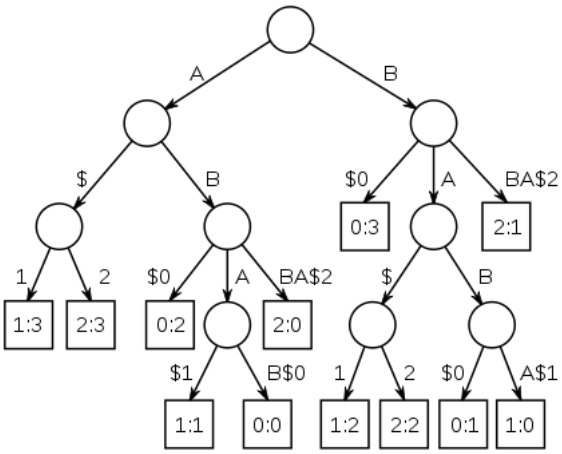
จุดเด่นของทรัย เมื่อเทียบกับต้นไม้ทวิภาค (Binary Search Tree: BST) คือ

- ค้นหาเร็วกว่า การค้นหาคีย์ที่มีความยาว m ใช้เวลามากที่สุดเป็น $O(m)$ ในขณะที่ BST ใช้เวลามากที่สุดเป็น $O(m \log n)$ ในการเปรียบเทียบคีย์ ซึ่งทรัยใช้วิธีการอย่างง่ายระหว่างการค้นหา เช่น การกำหนดดัชนีอาร์เรย์ด้วยตัวอักษร ซึ่งมีความเร็วบนระบบงานจริง
- ใช้พื้นที่น้อยกว่า เมื่อเก็บข้อมูลสตริงสั้น ๆ จำนวนมาก เนื่องจากไม่ได้จัดเก็บคีย์โดยชัดเจน และมีหลายโหนดใช้คีย์ที่มีคำขึ้นต้นเหมือนกันร่วมกัน
- ทำการเปรียบเทียบสตริงส่วนหน้าที่ยาวที่สุด (Longest-Prefix Matching) ง่ายขึ้น
- ทำการเพิ่มข้อมูลเร็วกว่าตารางแฮช (Hash Table) เพราะว่าแฮชจำเป็นต้องสร้างตารางแฮชใหม่ทุกครั้งที่ตารางแฮชเต็ม ซึ่งเป็นกระบวนการที่ใช้ทรัพยากรมาก

ลักษณะการจัดเก็บข้อมูลของทรัยมีอยู่หลายวิธีการ เช่น จัดเก็บข้อมูลในรูปแบบอาร์เรย์ ที่มีจุดเด่นที่มีความรวดเร็วในการสืบค้น การลบ และการเพิ่มข้อมูลเข้าไปในทรัย แต่ใช้ทรัพยากรในการจัดเก็บมาก ขนาดพื้นที่การจัดเก็บข้อมูลจะเท่ากับผลคูณจำนวนโหนดกับจำนวนตัวอักษรทั้งหมดที่เป็นไปได้ หรือการจัดเก็บทรัยในรูปแบบลิสต์ ที่ชี้ไปยังโหนดถัดไป ทำให้ใช้พื้นที่ในการจัดเก็บข้อมูลที่น้อยกว่า แต่จะทำให้การสืบค้นทำได้ช้าลง ถ้ามีการชี้จากโหนดหนึ่งไปยังหลาย ๆ โหนด

2.1.4 ลำดับเหมือนที่ยาวที่สุดในชุดอักขระ

การหาลำดับเหมือนที่ยาวที่สุดในอักขระ เป็นปัญหาหนึ่งที่ใช้ในการหาลำดับที่เหมือนกันที่ยาวที่สุดของชุดอักขระตั้งแต่ 2 ชุดขึ้นไป เพื่อตรวจสอบความเหมือนหรือความคล้ายของชุดอักขระ นิยมใช้ Suffix Tree ในการแก้ปัญหา โดยพบว่าใช้เวลาในการหาลำดับเหมือนที่ยาวที่สุดของชุดอักขระ 2 ชุดโดยใช้ Suffix Tree เป็นเวลาเชิงเส้น (Linear Time) เป็น $O(m)$ โดยที่ m คือความยาวของชุดอักขระ [**9**]



ภาพที่ 2.3 ตัวอย่าง Suffix Tree

ยกตัวอย่างเช่น การหาลำดับเหมือนที่ยาวที่สุดของชุดอักขระ ABAB BABA และ ABBA ชุดอักขระจะถูกเพิ่มอักขระพิเศษที่บ่งบอกว่าสิ้นสุดชุดอักขระ $0 $1 และ $2 ตามลำดับเป็น ABAB$0 BABA$1 และ ABBA$2 ด้วยการใช้ Suffix Tree ในการหาพบว่าลำดับเหมือนที่ยาวที่สุดคือ AB และ BA ดังแสดงในภาพที่ 2.3

2.2 งานวิจัยที่เกี่ยวข้อง

2.2.1 ขนาดของเซอร์วิซ

Haesen และคณะ [**10**] ได้นำเสนอการแบ่งประเภทขนาดเซอร์วิซ โดยที่เซอร์วิซหมายถึงโอเปอเรชัน ขนาดของเซอร์วิซแบ่งเป็น (1) ขนาดของการทำงาน (Functionality Granularity) ซึ่งบ่งบอกปริมาณการทำงานของเซอร์วิซนั้น และอาจเปลี่ยนแปลงได้ตามจำนวนหรือประเภทของพารามิเตอร์ที่รับเข้าเซอร์วิซ (2) ขนาดของข้อมูล (Data Granularity) คือปริมาณข้อมูลที่แลกเปลี่ยนกับเซอร์วิซทั้งรับเข้าและตอบกลับ และ (3) ขนาดคุณค่าทางธุรกิจ (Business Value Granularity) ซึ่งโดยทั่วไปจะวัดว่าเซอร์วิซนั้นตอบสนองต่อเป้าหมายขององค์กรมากน้อยเพียงใด นอกจากนี้ยังมีการนำเสนอผลกระทบของขนาดของเซอร์วิซประเภทต่าง ๆ ต่อสถาปัตยกรรมของระบบ ในด้านการนำกลับมาใช้ซ้ำ สมรรถนะ และความยืดหยุ่น ทั้งในมุมมองของผู้ให้บริการเซอร์วิซและผู้ใช้งาน โดยทั่วไปการออกแบบให้เซอร์วิซมีขนาดใหญ่จะส่งผลดีต่อทั้งผู้ให้บริการและผู้เรียกใช้งานมากกว่าผลเสีย เช่นเซอร์วิซที่มีขนาดการทำงานใหญ่ มีข้อมูลรับเข้ามาก จะส่งผลกระทบในเชิงบวกต่อการนำกลับมาใช้ซ้ำ

Jiang และคณะ [**11**] ได้เสนอการหาขนาดของเซอร์วิซที่เหมาะสมโดยการนำกระบวนการทางธุรกิจ (Business Process) ทั้งหลายที่มีในระบบมาพิจารณา การวิเคราะห์ทำใน

2 มุมมอง ได้แก่ มุมมองระดับกิจกรรม และมุมมองระดับโอเปอเรชัน โดยแตกกระบวนการธุรกิจออกเป็นกลุ่มของกิจกรรมและกลุ่มของโอเปอเรชัน แล้วใช้การทำเหมืองข้อมูลด้วยอัลกอริทึมเอไพรออรี ในการหากลุ่มของกิจกรรม หรือกลุ่มของโอเปอเรชันที่เกิดบ่อยในกระบวนการธุรกิจทั้งหลาย แล้วทำการพิจารณาการรวมกิจกรรม หรือโอเปอเรชัน ให้เป็นกิจกรรมเดียว หรือโอเปอเรชันเดียว โดยดูจากประเภทของกิจกรรม หรือประเภทของโอเปอเรชันประกอบด้วย ทั้งนี้อาจจะไม่ทำการรวมโอเปอเรชันทั้งหมดที่ได้จากทำเหมืองข้อมูล หากมีบางโอเปอเรชันที่มีประเภทหรือหน้าที่การทำงานแตกต่างจากโอเปอเรชันอื่น ๆ ภายในกลุ่มโอเปอเรชัน อีกทั้งงานวิจัยนี้ยังไม่สนใจถึงลำดับการเกิดเหตุการณ์ในการพิจารณาการรวมอีกด้วย

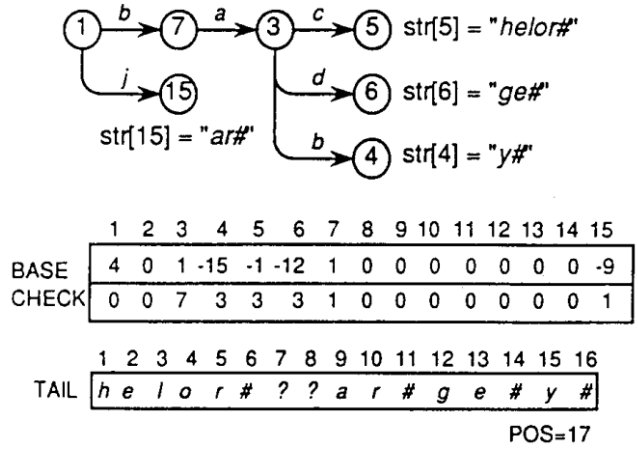
Senivongse และคณะ [**4**] ได้นำเสนอกรอบงานการวิเคราะห์ขนาดความสามารถของเว็บเซอร์วิซจากการวิเคราะห์การเรียกใช้งาน เพื่อหากลุ่มโอเปอเรชันที่ถูกเรียกใช้ร่วมกันบ่อย ๆ และแนะนำให้ผู้ออกแบบเซอร์วิซพิจารณารวมโอเปอเรชันเหล่านี้เข้าด้วยกันเพื่อลดต้นทุนของผู้ใช้งานในการเรียกใช้ กรอบงานมี 3 องค์ประกอบหลักคือ (1) การเฝ้าติดตามการเรียกใช้งานเซอร์วิซจากผู้ใช้งาน (2) การบันทึกการเรียกใช้งาน โดยจัดเก็บรายละเอียดต่าง ๆ เกี่ยวกับการเรียกใช้งานเซอร์วิซ ได้แก่ ชื่อเซอร์วิซ ชื่อโอเปอเรชัน หมายเลขเครื่องผู้ใช้งาน เวลาที่เรียก ลงในฐานข้อมูล (3)ทำการวิเคราะห์การเรียกใช้งาน โดยใช้การหากฎความสัมพันธ์โดยอัลกอริทึมเอไพรออรี แต่เนื่องด้วยการหากฎความสัมพันธ์โดยใช้อัลกอริทึมเอไพรออรีแบบทั่วไปนั้นสามารถระบุได้แค่เพียงเซตของโอเปอเรชันที่มีความสัมพันธ์กัน แต่ไม่สามารถระบุลำดับของโอเปอเรชันในเซตนั้นได้ การวิเคราะห์ดังกล่าวจึงยังไม่เพียงพอต่อการนำไปปรับปรุงเซอร์วิซตามที่แนะนำได้

2.2.2 การจัดเก็บข้อมูลในทรัย

การใช้ทรัยในการจัดเก็บข้อมูลเดิมนิยมใช้อาร์เรย์ในการจัดเก็บข้อมูล โดยแต่ละโหนดจะถูกเก็บอยู่ในอาร์เรย์ และมีการชี้ไปยังโหนดถัดไป ซึ่งมีจุดเด่นในการทำงานที่รวดเร็วทั้งการเพิ่ม ลบ หรือค้นคืน แต่จำเป็นต้องใช้พื้นที่ในการจัดเก็บมากขึ้นกับความยาวและปริมาณของข้อมูลที่จัดเก็บ วิธีหนึ่งที่ช่วยแก้ปัญหานี้คือการจัดเก็บทรัยในรูปแบบลิสต์ ซึ่งสามารถช่วยลดพื้นที่ในการจัดเก็บ แต่ก็ทำให้การค้นคืนช้าลงถ้าแต่ละโหนดมีเส้นขอบ (Arc) ที่ชี้ไปยังโหนดอื่นเป็นจำนวนมาก Aoe และคณะ [**3**] จึงได้เสนอวิธีการจัดเก็บข้อมูลทรัยในรูปแบบอาร์เรย์ 1 มิติ 2 ชุด ซึ่งเรียกว่าดับเบิลอาร์เรย์ (Double-Array) ประกอบด้วยอาร์เรย์ BASE และอาร์เรย์ CHECK โหนดแต่ละโหนดจะถูกจัดเก็บโดยสร้างความสัมพันธ์จาก BASE ไปยัง CHECK โดยที่จะไม่มีการอ้างอิงไปยังตำแหน่งของ CHECK ที่ซ้ำกัน เส้นขอบที่แทนค่าตัวอักษร a จากโหนด n ไปยังโหนด m จะ

ถูกแสดงอยู่ในรูปของ *g(n, a) = m* แต่ละเส้นขอบ สามารถสืบค้นได้ในเวลา $O(1)$ และไม่เกิน $O(k)$ สำหรับคีย์ที่มีความยาวเป็น k

ข้อมูลที่ถูกจัดเก็บในดับเบิลอาร์เรย์จะจัดเก็บเฉพาะที่เพียงพอต่อการแยกความแตกต่างของค่าคีย์ต่าง ๆ ที่จัดเก็บอยู่ในทรัยเดียวกัน ยกตัวอย่างการจัดเก็บข้อมูล K = {baby#, bachelor, badge#, jar#} โครงสร้างทรัยที่ทำการจัดเก็บข้อมูลจะเป็นดังรูปที่ 4 จะเห็นว่าข้อมูลในโหนดต่าง ๆ ที่จัดเก็บอยู่บนทรัย จะจัดเก็บเพียงเพื่อจำแนกความแตกต่างของคีย์ทั้งสี่เท่านั้น ไม่ได้จัดเก็บทุกตัวอักษรของค่าคีย์แต่ละตัว รูปแบบเช่นนี้เพื่อให้สามารถค้นคืนค่าคีย์ที่ค้นหาได้อย่างรวดเร็ว และเป็นการลดพื้นที่ในการจัดเก็บข้อมูลลงในอีกทางหนึ่ง



ภาพที่ 2.4 ตัวอย่างทรัยและดับเบิลอาร์เรย์สำหรับคีย์ K [**3**]

จากการทดลองเปรียบเทียบการใช้ดับเบิลอาร์เรย์กับการใช้ลิสต์พบว่า ดับเบิลอาร์เรย์ใช้เวลาเฉลี่ยในการเพิ่มข้อมูลมากกว่าการใช้ลิสต์ เมื่อคีย์มีจำนวนมากขึ้น ในขณะที่ลิสต์ใช้เวลาค่อนข้างคงที่ ในทางกลับกัน การลบ และการค้นคืนในดับเบิลอาร์เรย์กลับใช้เวลาน้อยกว่าลิสต์ ซึ่งลักษณะการนำทรัยไปใช้งานจะใช้การค้นคืนเป็นส่วนใหญ่ เนื่องจากนำไปประยุกต์ใช้ในการจัดเก็บพจนานุกรม ซึ่งมีการเรียกค้นคืนบ่อยครั้ง ทำให้ผลการค้นคืนมีความรวดเร็วในขณะที่มีปริมาณคำอยู่เป็นจำนวนมาก

สิ่งสำคัญของการใช้ดับเบิลอาร์เรย์คือ การกำหนดค่าตัวเลขให้กับตัวอักษรต่าง ๆ ไว้ล่วงหน้าแล้ว เนื่องจากจำเป็นต้องทำการแปลงตัวอักษรให้อยู่ในรูปตัวเลขที่กำหนด เพื่อใช้ในการคำนวณหาตำแหน่งที่จะใช้ในการจัดเก็บข้อมูล สร้างความเชื่อมโยงของข้อมูลไปยังตัวอักษรตัวถัดไป และใช้ในการระบุตำแหน่งของส่วนที่เหลือของค่าที่จัดเก็บอยู่ภายใน งานวิจัยนี้ได้ประยุกต์การเก็บข้อมูลในดับเบิลอาร์เรย์จากที่เดิมเก็บตัวอักษรเป็นการเก็บชื่อโอเปอเรชันของเซอร์วิซ

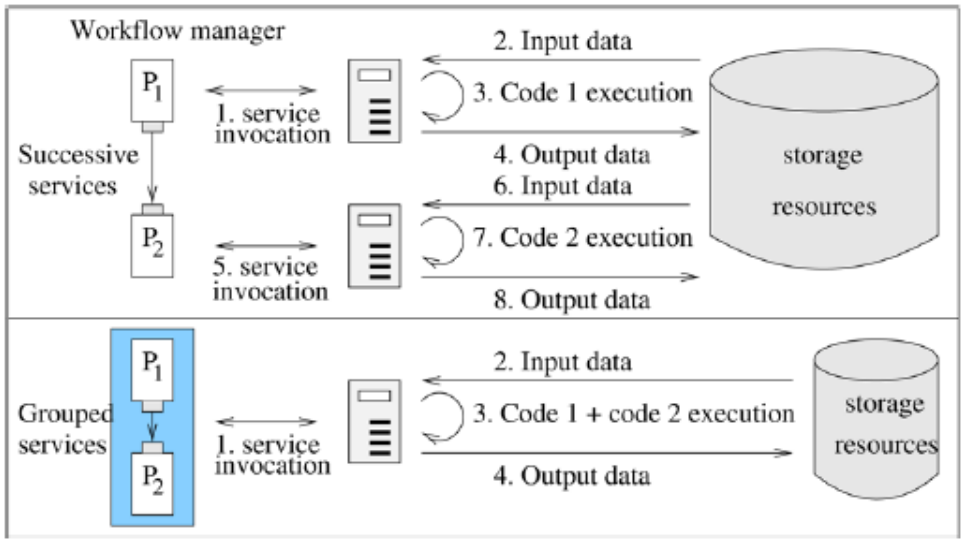
2.2.3 อัลกอริทึมเอไพรออรีเชิงทรัย

โครงสร้างข้อมูลที่ใช้จัดเก็บแคนดิเดต (Candidate) ของเซตข้อมูลที่พบบ่อย ที่ทำให้อัลกอริทึมเอไพรออรีมีประสิทธิภาพ คือโครงสร้างแบบทรัย Bodon [**2**] จึงได้เสนอการนำทรัยมาใช้กับเอไพรออรีเพื่อใช้ในการหาลำดับของไอเท็มที่เกิดบ่อย (Mining Frequent Item Sequence) โดยมีขั้นตอนการทำงานคร่าว ๆ คือ ในขั้นตอนการสร้างแคนดิเดต จะทำการอ่านข้อมูลมาทีละทรานแซกชัน แล้วจึงอ่านข้อมูลในแต่ละลำดับ เพื่อท่อง (Traverse) ไปยังโหนดต่าง ๆ ในทรัย หากเป็นข้อมูลที่ยังไม่มีอยู่ในทรัยจะทำการสร้างโหนดใหม่ขึ้นมา เมื่อถึงข้อมูลลำดับสุดท้ายที่โหนดใด โหนดนั้นจะถือเป็นตัวแทนของข้อมูลทรานแซกชันดังกล่าว และทำการนับค่าสนับสนุน (Support) เพิ่มขึ้น หลังจากอ่านข้อมูลครบทุกทรานแซกชันแล้วจะเข้าสู่ขั้นตอนการลบโหนดที่ไม่เกิดบ่อย นั่นคือโหนดที่มีค่าสนับสนุนต่ำกว่าที่กำหนดออกจากทรัย ซึ่งการลบโหนดจำเป็นต้องให้ความสำคัญกับการลบโหนดที่ไม่ใช่โหนดใบ (Leaf Node) ซึ่งอาจทำให้โหนดลูกหลานของโหนดดังกล่าวถูกลบออกจากทรัยไปด้วย ซึ่งจะส่งผลต่อการสร้างกฎความสัมพันธ์ได้

2.2.4 การรวมกลุ่มเซอร์วิซเพื่อการประกอบเซอร์วิซ

Glatard และคณะ [**12**] ได้กล่าวถึงจุดมุ่งหมายของการรวมกลุ่มเซอร์วิซ (Service Grouping) เพื่อปรับเหมาะ (Optimize) ระยะเวลาดำเนินการของผังงาน (Workflow) การรวมกลุ่มเซอร์วิซอาจช่วยลดภาระโดยรวมที่เกิดจากการส่งงาน (Submission) การจัดตารางเวลา (Scheduling) การจัดคิว (Queuing) และการส่งข้อมูล เนื่องจากเป็นการลดจำนวนครั้งในการเรียกใช้งาน แต่การรวมกลุ่มก็อาจลดลักษณะแบบขนาน (Parallelism) ในกระบวนการทำงานของเซอร์วิซลง ซึ่งจำเป็นต้องให้ความเอาใจใส่ในกลยุทธการรวมกลุ่มเพื่อหลีกเลี่ยงการสูญเสียประสิทธิภาพ โดยทั่วไปแล้วนิยมรวมกลุ่มเซอร์วิซที่ทำงานเชื่อมต่อเป็นลำดับ (Sequentially linked) เข้าด้วยกัน เนื่องจากจะไม่กระทบต่อลักษณะแบบขนานใด ๆ

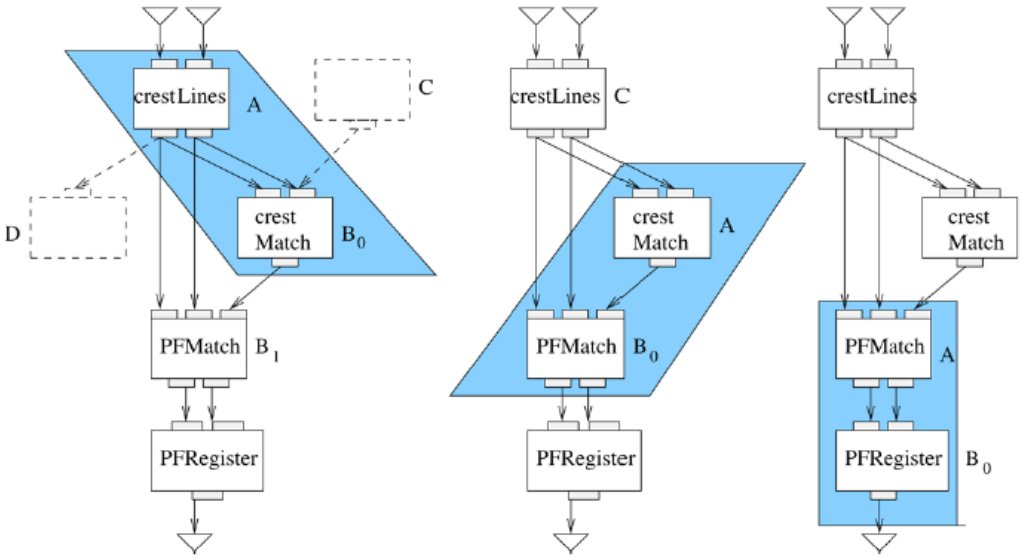
ยกตัวอย่างการเรียกใช้งานอย่างง่าย ๆ ในภาพที่ 2.5 โดยพิจารณาการเรียกใช้งานโอเปอเรชัน P1 และ P2 แบบปกติ ที่ทำการเรียกใช้งานแยกจากกันอย่างอิสระ การส่งข้อมูลจะถูกดำเนินการโดยการเรียกโอเปอเรชันแต่ละตัว ผลลัพธ์จาก P1 และข้อมูลป้อนเข้า P2 ถูกจัดการในระดับ Workflow Engine เปรียบเทียบกับการรวม P1 และ P2 ให้เป็นโอเปอเรชันเสมือนตัวเดียว โอเปอเรชันนี้สามารถเรียกใช้โอเปอเรชันทั้งสองที่ถูกรวมอยู่ภายในตามลำดับ ทำให้แก้ปัญหาการส่งข้อมูลและการขึ้นต่อกันของการเรียกใช้งานได้ อีกทั้งยังลดจำนวนการส่งข้อมูลของเซอร์วิซอีกด้วย



ภาพที่ 2.5 การเรียกใช้เซอร์วิซแบบคลาสสิก (บน) และการรวมเซอร์วิซ (ล่าง) [**12**]

แต่อย่างไรก็ตามการรวมโอเปอเรชันเข้าด้วยกันอาจทำให้สูญเสีย Parallelism ซึ่งอาจส่งผลต่อระยะเวลาเรียกใช้งานในมุมมองของผู้เรียกใช้งาน เพื่อให้มั่นใจว่าจะไม่สูญเสีย Parallelism จากการรวม จึงได้กำหนดกฎในการรวมไว้ดังนี้

ให้ A แทนโอเปอเรชันที่ทำงานอยู่ และ B คือโอเปอเรชันถัดไปที่จะถูกเรียกใช้ต่อจาก A การรวมโอเปอเรชัน B_i กับ A ทำได้ก็ต่อเมื่อเข้าเงื่อนไข 2 ประการ (1) B_i คือโอเปอเรชันที่ถูกเรียกใช้งานก่อน B_j สำหรับทุก $i \neq j$ และ (2) ทุกโอเปอเรชัน C ที่ต้องเรียกใช้งานก่อน B_i เป็นโอเปอเรชันที่เรียกใช้งานก่อน A ด้วยหรือคือ A เอง

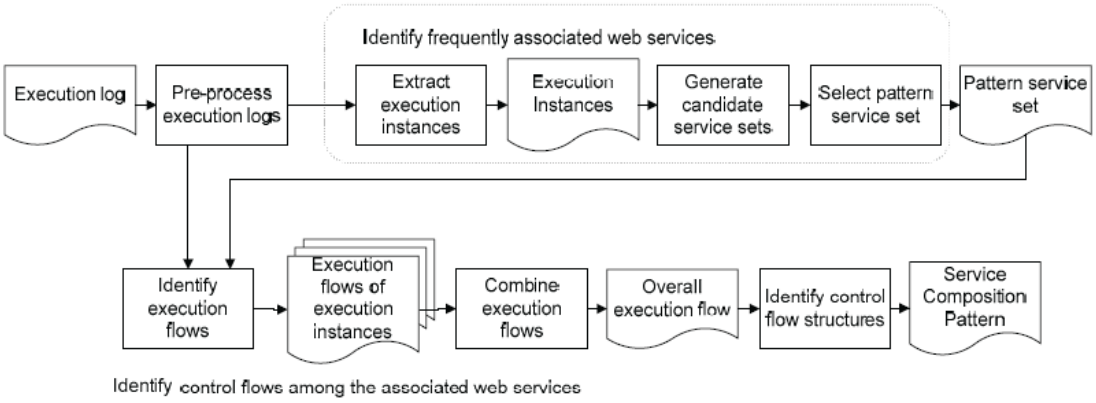


ภาพที่ 2.6 ตัวอย่างการรวมกลุ่มเซอร์วิซ [**12**]

จากภาพที่ 2.6 พิจารณาแผนผังการทำงานด้านซ้ายของรูป พิจารณา A คือ crestLines และมีโอเปอเรชันที่ถูกเรียกใช้งานต่อ 2 โอเปอเรชันคือ B_0 (crestMatch) และ B_1 (PFMatch) โดยที่ crestMatch ถูกเรียกใช้งานก่อน PFMatch และ crestMatch มีโอเปอเรชันที่เรียกใช้งานก่อนหน้าคือ crestLimes ซึ่งถูกต้องตามกฎการรวม จึงสามารถทำการรวมการทำงานของ crestLines และ crestMatch เข้าด้วยกันได้ แต่หากมีโอเปอเรชัน C เพิ่มเข้ามาในแผนผังการทำงาน ก็จะทำให้ไม่สามารถรวมได้เนื่องจาก C ไม่ได้ถูกเรียกใช้ก่อน A จึงไม่เข้ากับกฎการรวมกลุ่ม ตรงกลางของรูปพิจารณา A คือ crestMatch ซึ่งมีโอเปอเรชันที่เรียกใช้งานต่อคือ PFMatch (B_0) เพียงโอเปอเรชันเดียว และมีการเรียกใช้ crestLines (C) ก่อนที่จะมีการเรียกใช้ crestMatch (ก) และ PFMatch (B_0) ซึ่งตรงตามกฎการรวมกลุ่ม จึงสามารถรวม crestMatch และ PFMatch เข้าด้วยกันได้ ทางขวาของรูปพิจารณา A คือ PFMatch ซึ่งมีโอเปอเรชันที่เรียกใช้งานต่อคือ PFRegister(B_0) เพียงโอเปอเรชันเดียว และ PFRegister ก็มีเพียง PFMatch ที่ถูกเรียกใช้งานก่อนหน้าเพียงโอเปอเรชันเดียวเช่นกัน ซึ่งตรงตามกฎการรวมกลุ่ม จึงสามารถรวม PFMatch และ PFRegister เข้าด้วยกันได้ จากการพิจารณาทั้ง 3 รูปจะพบว่า เราสามารถทำการรวมกลุ่มโอเปอเรชันทั้งสี่เข้าเป็นโอเปอเรชันเดียวกันได้

นอกจากนี้ Tang และ Zou [**13**] ยังทำการพิจารณาการรวมกลุ่มโอเปอเรชันของหลาย ๆ เซอร์วิซจากบันทึกการเรียกใช้งาน (Execution Log) ซึ่งเครื่องเซิร์ฟเวอร์ได้บันทึกเหตุการณ์ทั้งหมดที่เกิดขึ้นบนเครื่องไว้ ในบันทึกการใช้งานจะประกอบด้วยเหตุการณ์หลายประเภทด้วยกัน แต่เลือกพิจารณาเฉพาะเหตุการณ์ที่เป็นการเรียกใช้งานเซอร์วิซ (Service Invocation Event) เพื่อให้ได้ทราบถึงเซตของแบบรูปของการเรียกใช้งานเซอร์วิซ และนำไปสู่การหาแบบรูปการรวมเซอร์วิซที่เป็นไปได้สำหรับการประกอบเซอร์วิซ ขั้นตอนเป็นดังรูปที่ 7 โดยมีขั้นตอนหลัก ๆ 3 ขั้นตอน ดังนี้ (1) ทำการบันทึกการทำงานของเครื่องเซิร์ฟเวอร์ และเลือกพิจารณาเฉพาะเหตุการณ์การเรียกใช้งานเซอร์วิซ ประกอบด้วยเหตุการณ์ ENTRY คือเหตุการณ์ถูกบันทึกเมื่อเซอร์วิซถูกเรียกใช้งาน และเหตุการณ์ EXIT คือเหตุการณ์ที่เซอร์วิซสิ้นสุดการทำงานและคืนผลลัพธ์ไปยังผู้เรียกใช้งาน เหตุการณ์ทั้งสองนี้จะเก็บข้อมูลเวลาที่เกิดเหตุการณ์ ชื่อเซอร์วิซ รหัสของการเรียกใช้งาน และแอพพลิเคชันที่เรียกใช้งาน (2) ทำการกำหนดเซอร์วิซที่มีการเรียกใช้งานบ่อย โดยการหากลุ่มของเซอร์วิซที่มีการเรียกใช้งานร่วมกันบ่อย โดยสนใจเฉพาะกลุ่มของเซอร์วิซที่มีขนาดอย่างน้อย 4 เซอร์วิซ และเลือกกลุ่มที่มีความความถี่สูงสุดเป็นตัวแทนแบบรูปของกลุ่มเซอร์วิซ และ (3) ทำการหาลำดับการทำงานของตัวแทนแบบรูปของกลุ่มเซอร์วิซ โดยทำการรวบรวมลำดับการทำงานจากการเรียกใช้งานเซอร์วิซทั้งหมด เพื่อหาโครงสร้างการทำงาน (แบบ

ลำดับหรือแบบขนาน เป็นต้น) จากลำดับการเรียกใช้งานทั้งหมดต่อไป ซึ่งทำให้เราสามารถสร้างแผนผังการทำงานได้

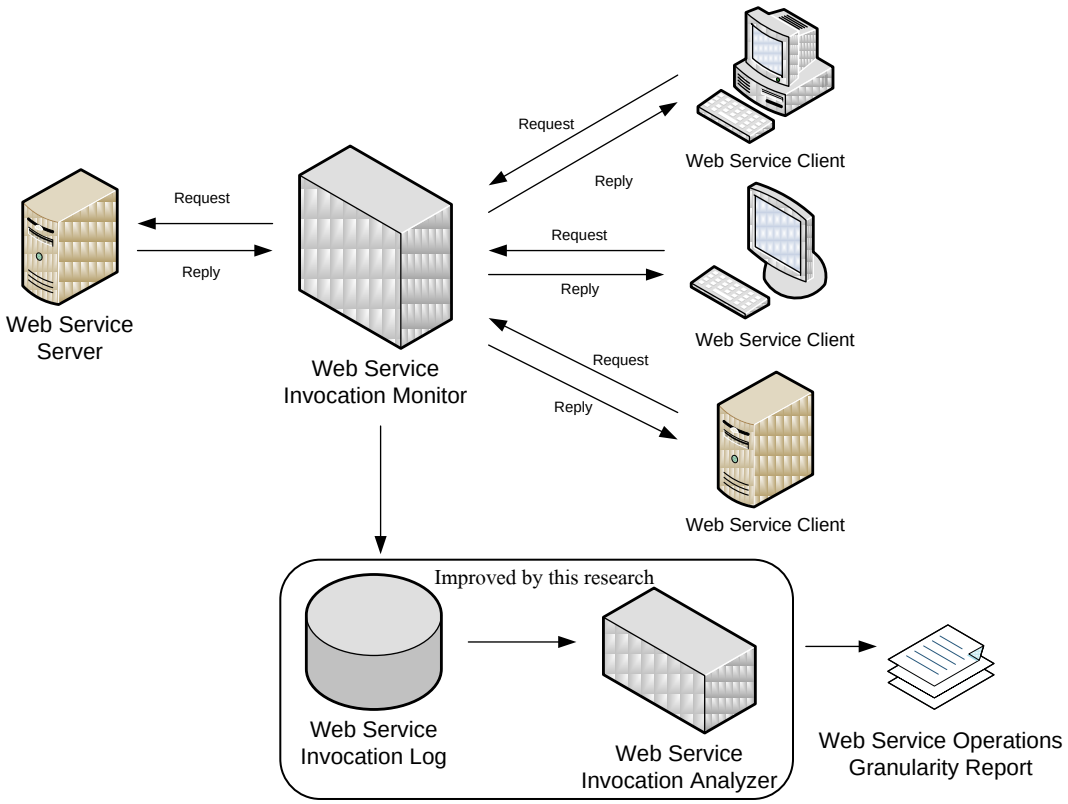


ภาพที่ 2.7 ขั้นตอนการวิเคราะห์การรวมกลุ่มเซอร์วิซจากบันทึกการเรียกใช้งาน [**13**]

บทที่ 3

วิธีดำเนินการวิจัย

การวิเคราะห์ขนาดของความสามารถของเซอร์วิซในงานวิจัยของ Senivongse และคณะ [**4**] มีจุดมุ่งหมายเพื่อทำการตรวจหาขนาดความสามารถของโอเปอเรชันที่เล็กเกินไปของเซอร์วิซหนึ่ง ๆ และแนะนำผู้ออกแบบหรือผู้ให้บริการเซอร์วิซในการรวมความสามารถให้มีขนาดใหญ่ขึ้นสำหรับเซอร์วิซนั้น ซึ่งมีภาพรวมการวิเคราะห์ขนาดความสามารถเซอร์วิซ แสดงได้ดังภาพที่ 3.1



ภาพที่ 3.1 ภาพรวมการวิเคราะห์ขนาดความสามารถของเซอร์วิซ [**4**] และส่วนที่ปรับปรุง

เมื่อผู้ใช้บริการ (Web Service Client) ส่งการร้องขอมายังเซอร์วิซ (Web Service Server) จะถูกระบบเฝ้าติดตามการเรียกใช้งานเซอร์วิซ (Web Service Invocation Monitor) ตรวจจับการเรียกใช้งาน และเก็บข้อมูลการเรียกใช้งานบันทึกลงในฐานข้อมูลบันทึกการใช้งานเซอร์วิซ (Web Service Invocation Log) เพื่อนำไปใช้ในการทำเหมืองข้อมูลด้วยอัลกอริทึมเอไพรออรี โดยใช้ตัววิเคราะห์การเรียกใช้งานเว็บเซอร์วิซ (Web Service Invocation Analyzer) ผลลัพธ์ที่ได้จะอยู่ในรูปของกฎความสัมพันธ์ระหว่างกลุ่มของโอเปอเรชัน เช่น {A, B} → {C, D} หมายถึง เมื่อมีเหตุการณ์การเรียกใช้โอเปอเรชัน A และ B แล้ว จะพบเหตุการณ์การเรียกใช้งาน โอเปอเรชัน C และ D ด้วย

เป็นต้น จะเห็นได้ว่า กฎความสัมพันธ์มุ่งเน้นการหาความสัมพันธ์ของเหตุการณ์ที่เกิดร่วมกัน แต่ไม่สามารถบอกถึงลำดับการเกิดเหตุการณ์ว่าลำดับการเรียกใช้งาน A, B, C, D เป็นอย่างไร

ในการดำเนินงานวิจัยนี้ ยังคงใช้องค์ประกอบของกรอบงานที่เสนอใน [**4**] แต่ได้ทำการปรับปรุงในส่วนของตัววิเคราะห์การเรียกใช้งานเว็บเซอร์วิซ (Web Service Invocation Analyzer) โดยทำการปรับปรุงวิธีการหากฎความสัมพันธ์ของการเรียกใช้โอเปอเรชัน จากเดิมที่ใช้อัลกอริทึมเอไพรออรีในการหากฎความสัมพันธ์ ไปเป็นการหากฎความสัมพันธ์ด้วยเอไพรออรีเชิงทรัยตามวิธีการของ Bodon [**2**] ประกอบกับการใช้ดับเบิลอาร์เรย์ทรัย [**3**] ซึ่งสามารถระบุลำดับเหตุการณ์ของการเรียกใช้โอเปอเรชันภายในกฎได้อย่างชัดเจน โดยนำเสนอในรูปแบบของกฎความสัมพันธ์ของลำดับการเรียกใช้โอเปอเรชัน เช่น B → A → C → D หมายถึง เมื่อมีการเรียกใช้งานโอเปอเรชัน B แล้วจะมีการเรียกใช้งานโอเปอเรชัน A และ C และ D ตามลำดับต่อเนื่องกัน เป็นต้น นอกจากนี้ยังได้ทำการปรับปรุงการแนะนำการรวมกลุ่มของลำดับการเรียกใช้งานโอเปอเรชันที่ได้จากกฎความสัมพันธ์ ในส่วนของการรายงานขนาดความสามารถของโอเปอเรชัน (Web Service Operation Granularity Report) โดยทำการหาลำดับร่วมที่ยาวที่สุดของลำดับการเรียกใช้งานที่พบบ่อย รายละเอียดของงานวิจัยในแต่ละส่วนมีดังต่อไปนี้

3.1 เว็บเซอร์วิซระบบซื้อสินค้าออนไลน์

ในงานวิจัยนี้ได้ทำการสร้างเว็บเซอร์วิซระบบซื้อสินค้าออนไลน์ขึ้น เพื่อใช้เป็นเซอร์วิซตัวแทนในการวิเคราะห์ขนาดความสามารถของเซอร์วิซ โดยทำการสร้างให้แต่ละโอเปอเรชันมีขนาดความสามารถที่เล็ก กล่าวคือแต่ละโอเปอเรชันมีหน้าที่ครอบคลุมปริมาณงานที่น้อย หากผู้เรียกใช้ต้องการกระบวนงานที่ต้องการการทำงานหลายอย่างเพื่อให้ครอบคลุมกับความต้องการของผู้ใช้งานอาจจำเป็นต้องเรียกใช้งานหลายโอเปอเรชันต่อเนื่องกันจึงจะได้การทำงานที่เพียงพอกับความต้องการ เซอร์วิซที่สร้างขึ้นนี้ประกอบด้วยโอเปอเรชันต่างๆ ดังตารางที่ 3.1

ตารางที่ 3.1 โอเปอเรชันของเว็บเซอร์วิซซื้อสินค้าออนไลน์

ชื่อโอเปอเรชัน	คำอธิบาย
Login	เข้าใช้งานเซอร์วิซ
Logout	สิ้นสุดการใช้งานเซอร์วิซ
CategoryNameList	แสดงรายการหมวดหมู่สินค้า
browseCategory	แสดงรายการสินค้าในหมวดหมู่
cartCreate	สร้างตระกร้าสินค้า
cartAdd	เพิ่มสินค้าลงในตะกร้าสินค้า

ตารางที่ 3.1 โอเปอเรชันของเว็บเซอร์วิซซื้อสินค้าออนไลน์ (ต่อ)

ชื่อโอเปอเรชัน	คำอธิบาย
cartGet	แสดงรายการสินค้าในตะกร้าสินค้า
cartRemove	เอาสินค้าออกจากตะกร้าสินค้า
cartClear	เอาสินค้าทั้งหมดออกจากตะกร้าสินค้า
itemSearch	ค้นหาสินค้า
itemLookup	แสดงรายละเอียดสินค้า
Checkout	ทำรายการสั่งซื้อสินค้า

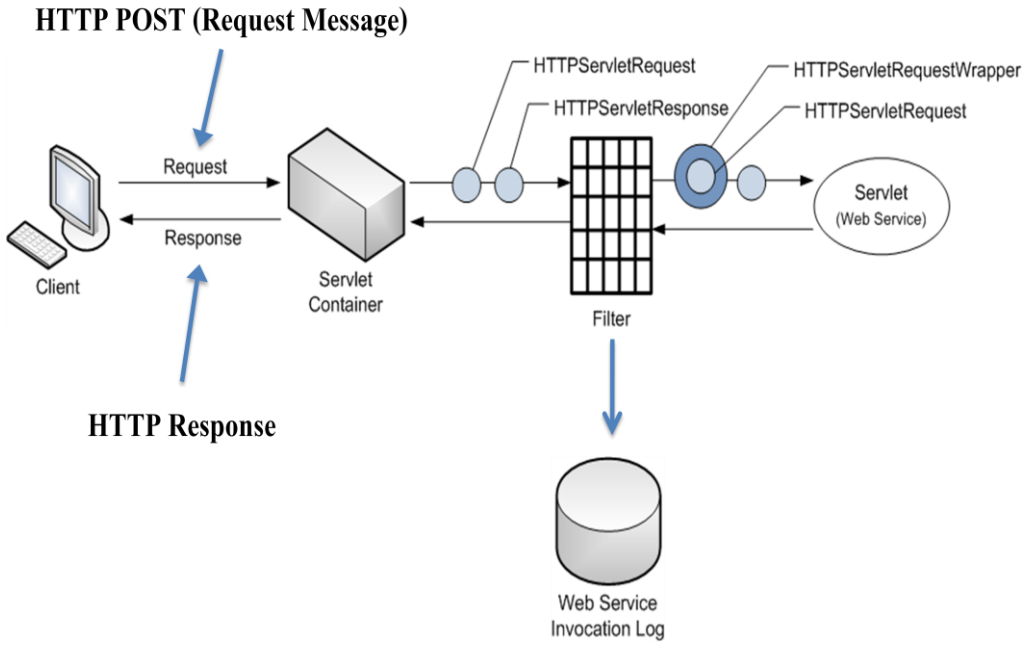
3.2 ระบบเฝ้าติดตามการเรียกใช้เซอร์วิซ

ในแต่ละครั้งที่ผู้ใช้งานส่งการร้องขอมายังเซอร์วิซ ระบบเฝ้าติดตามการเรียกใช้เซอร์วิซ (Web Service Invocation Monitor) จะทำการตรวจจับทุกการร้องขอจากผู้ใช้บริการที่ส่งมายังเซอร์วิซที่ต้องการวิเคราะห์ โดยการสร้างตัวกรอง (Filter) ทำการดักจับและกรองข้อความการร้องขอ เพื่อรวบรวมข้อมูลที่ต้องการเพื่อนำมาทำการวิเคราะห์ก่อนจะส่งต่อข้อความร้องขอต่อไปยังเซอร์วิซปลายทางด้วยวิธีการหุ้มห่อ (Request Wrapper) โดยยังคงสภาพคำร้องขอเสมือนว่าไม่เคยถูกอ่านมาก่อน ดังภาพที่ 3.2 ข้อมูลที่รวบรวมได้แก่

- หมายเลขไอพีเครื่องของผู้เรียกใช้งาน (Remote IP Address)
- ชื่อเซอร์วิซที่เรียกใช้งาน (Service Name)
- โอเปอเรชันที่เรียกใช้งาน (Operation Name)
- เวลาที่ส่งการร้องขอ (Request Time)
- เวลาที่ส่งการตอบกลับ (Response Time)

โดยผู้วิจัยได้ทำการเพิ่มการเก็บข้อมูลเวลาที่ส่งการตอบกลับขึ้นมา เพื่อให้สามารถทำการหาแผนผังการทำงานได้อย่างแม่นยำมากยิ่งขึ้น เนื่องจากอาจมีการเรียกใช้งานโอเปอเรชันอื่นในขณะที่กำลังเรียกใช้งานโอเปอเรชันหนึ่งอยู่

ข้อมูลทั้งหมดจะถูกบันทึกลงในฐานข้อมูลบันทึกการเรียกใช้งานเซอร์วิซ (Web Service Invocation Log) และส่งข้อความตอบกลับที่ได้รับจากเซอร์วิซให้กับผู้เรียกใช้งานต่อไป ข้อมูลการเรียกใช้งานที่รวบรวมได้เหล่านี้ จะถูกนำไปประมวลผลเพื่อสร้างเป็นทรานแซกชันสำหรับการหากฎความสัมพันธ์ต่อไป



ภาพที่ 3.2 ภาพรวมการทำงานของระบบเฝ้าติดตามการเรียกใช้งานเซอร์วิซ

3.3 ฐานข้อมูลบันทึกการใช้งานเซอร์วิซ

ฐานข้อมูลบันทึกการเรียกใช้งานเซอร์วิซ (Web Service Invocation Log) เป็นฐานข้อมูลที่ทำการจัดเก็บข้อมูลการเรียกใช้งานของทุกเซอร์วิซ และข้อมูลรายการทรานแซกชันที่ได้จากการประมวลผลการเรียกใช้งานเซอร์วิซ

3.4 การประมวลผลรายการทรานแซกชัน

เพื่อจัดกลุ่มการเรียกใช้งานให้อยู่ในทรานแซกชันเดียวกัน โดยที่การพิจารณาว่าการเรียกใช้งานโอเปอเรชันใดจะถูกรวมอยู่ในทรานแซกชันเดียวกันนั้น จะพิจารณาจากหมายเลขเครื่องของผู้เรียกใช้งาน ชื่อเซอร์วิซ และค่าขีดแบ่งทรานแซกชัน (Transaction Threshold) ซึ่งได้แก่ช่วงเวลาระหว่างเวลาสิ้นสุดการเรียกใช้งานโอเปอเรชันหนึ่งจนถึงเวลาที่เริ่มเรียกใช้งานโอเปอเรชันถัดไป ซึ่งช่วงเวลาดังกล่าวนี้ใช้เพื่อบ่งบอกว่าการเรียกสองโอเปอเรชันใดโดยผู้เรียกใช้งานรายหนึ่ง ๆ ถือเป็นการเรียกที่ต่อเนื่องกัน หากมีการเรียกใช้งานในลักษณะที่มีการเรียกใช้โอเปอเรชันอื่นในขณะที่โอเปอเรชันที่ทำการเรียกใช้ก่อนหน้ายังไม่สิ้นสุด จะทำการพิจารณาเวลาของโอเปอเรชันที่ถูกเรียกใช้ภายหลังว่ามีเวลาตอบกลับมากกว่าหรือน้อยกว่าโอเปอเรชันก่อนหน้า หากมีค่ามากกว่า ให้ถือว่าเป็นการใช้งานคนละทรานแซกชันกัน หากน้อยกว่าให้รวมอยู่ในทรานแซกชันเดียวกันโดยมีลำดับการเรียกใช้งานถัดจากโอเปอเรชันที่ถูกเรียกก่อนหน้านั้น จากข้อมูลตัวอย่างในตารางที่ 3.2

ตารางที่ 3.2 ตัวอย่างข้อมูลการเรียกใช้งาน

ลำดับ	เวลาที่ส่งการร้องขอ	ไอพีผู้เรียกใช้	ชื่อเซอร์วิซ	ชื่อโอเปอเรชัน	เวลาที่ส่งการตอบกลับ
1	2012-07-04 07:59:05	110.168.184.129	Shopping	browseCategory	2012-07-04 07:59:05
2	2012-07-04 07:59:26	110.168.184.129	Shopping	itemLookup	2012-07-04 07:59:26
3	2012-07-04 07:59:30	27.55.2.152	Shopping	cartCreate	2012-07-04 07:59:30
4	2012-07-04 07:59:30	27.55.2.152	Shopping	cartAdd	2012-07-04 07:59:30
5	2012-07-04 07:59:31	27.55.2.152	Shopping	calculatePrice	2012-07-04 07:59:31
6	2012-07-04 07:59:31	27.55.2.152	Shopping	cartGet	2012-07-04 07:59:31
7	2012-07-04 07:59:32	27.55.2.152	Shopping	itemLookup	2012-07-04 07:59:32
8	2012-07-04 07:59:35	110.168.184.129	Shopping	cartCreate	2012-07-04 07:59:35
9	2012-07-04 07:59:36	110.168.184.129	Shopping	cartAdd	2012-07-04 07:59:36
10	2012-07-04 07:59:36	110.168.184.129	Shopping	calculatePrice	2012-07-04 07:59:36
11	2012-07-04 07:59:36	110.168.184.129	Shopping	cartGet	2012-07-04 07:59:36
12	2012-07-04 07:59:37	110.168.184.100	Shopping	itemLookup	2012-07-04 07:59:37
13	2012-07-04 07:59:39	27.55.2.152	Shopping	browseCategory	2012-07-04 07:59:39
14	2012-07-04 07:59:46	110.168.184.129	Shopping	cartAdd	2012-07-04 07:59:46
15	2012-07-04 07:59:47	110.168.184.129	Shopping	calculatePrice	2012-07-04 07:59:47
16	2012-07-04 07:59:47	110.168.184.129	Shopping	cartGet	2012-07-04 07:59:47
17	2012-07-04 07:59:47	110.168.184.129	Shopping	itemLookup	2012-07-04 07:59:47
18	2012-07-04 07:59:52	27.55.2.152	Shopping	cartAdd	2012-07-04 07:59:52
19	2012-07-04 07:59:52	27.55.2.152	Shopping	calculatePrice	2012-07-04 07:59:52
20	2012-07-04 07:59:52	27.55.2.152	Shopping	cartGet	2012-07-04 07:59:52
21	2012-07-04 07:59:53	27.55.2.152	Shopping	itemLookup	2012-07-04 07:59:53
22	2012-07-04 07:59:56	110.168.184.129	Shopping	Checkout	2012-07-04 07:59:56
23	2012-07-04 07:59:57	27.55.2.152	Shopping	browseCategory	2012-07-04 07:59:57
24	2012-07-04 08:00:00	27.55.2.152	Shopping	itemLookup	2012-07-04 08:00:00
25	2012-07-04 08:00:03	27.55.2.152	Shopping	cartAdd	2012-07-04 08:00:03
26	2012-07-04 08:00:03	27.55.2.152	Shopping	calculatePrice	2012-07-04 08:00:03
27	2012-07-04 08:00:03	27.55.2.152	Shopping	cartGet	2012-07-04 08:00:03
28	2012-07-04 08:00:04	27.55.2.152	Shopping	itemLookup	2012-07-04 08:00:04
29	2012-07-04 08:00:06	27.55.2.152	Shopping	Checkout	2012-07-04 08:00:06

หากผู้ออกแบบเซอร์วิซกำหนดค่าขีดแบ่งทรานแซกชันสำหรับการพิจารณาความต่อเนื่องของการเรียกใช้โอเปอเรชันเป็น 2 วินาที จะได้จำนวนทรานแซกชันทั้งหมด 12 ทรานแซกชัน ดังแสดงในตารางที่ 3.3

ตารางที่ 3.3 ข้อมูลทรานแซกชัน

ลำดับ	ลำดับของโอเปอเรชันไอเท็ม
T1	browseCategory
T2	itemLookup
T3	cartCreate, cartAdd, calculatePrice, cartGet, itemLookup
T4	cartCreate, cartAdd, calculatePrice, cartGet
T5	itemLookup
T6	browseCategory
T7	cartAdd, calculatePrice, cartGet, itemLookup
T8	cartAdd, calculatePrice, cartGet, itemLookup
T9	Checkout
T10	browseCategory
T11	itemLookup
T12	cartAdd, calculatePrice, cartGet, itemLookup, Checkout

จากตัวอย่างจะเห็นได้ว่าการเรียกใช้งานในลำดับที่ 1 และ 2 เป็นการเรียกใช้งานจากผู้ใช้รายเดียวกัน แต่เนื่องจากช่วงเวลาจากการได้รับการตอบกลับ และการเรียกใช้งานครั้งถัดไปเกินระยะเวลา 2 วินาทีที่กำหนด จึงพิจารณาให้การเรียกใช้งานเป็นคนละทรานแซกชันกัน ในขณะที่การเรียกใช้งานลำดับที่ 11 และ 12 มีช่วงระยะห่างอยู่ภายในช่วงเวลาที่กำหนด แต่เกิดจากการเรียกใช้งานจากผู้ใช้งานต่างกัน จึงพิจารณาให้การเรียกใช้งานลำดับที่ 12 เป็นอีกทรานแซกชันหนึ่ง

3.5 การหาลำดับการเรียกใช้งานที่เกิดบ่อย

หลังจากที่ได้เตรียมข้อมูลการเรียกใช้งานให้อยู่ในรูปของทรานแซกชันแล้ว ขั้นตอนต่อไปเป็นการหาแบบรูปของลำดับการเรียกใช้งานที่เกิดบ่อยโดยประยุกต์ใช้อัลกอริทึมเอไพรออรีเชิงทรัย [**2**] โดยปกติแล้วอัลกอริทึมนี้ใช้เพื่อหาว่าในข้อมูลทรานแซกชันจำนวนมากที่วิเคราะห์อยู่นั้นมีลำดับของไอเท็มข้อมูลใดบ้างที่พบร่วมกันบ่อย ในที่นี้ไอเท็มจะหมายถึงหนึ่งโอเปอเรชันที่อยู่ในทรานแซกชัน โดยผู้วิเคราะห์ข้อมูลจะทำการกำหนดค่าสนับสนุนขั้นต่ำ (Minimum Support

Threshold) สำหรับใช้หากลุ่มข้อมูลที่พบร่วมกันบ่อย ซึ่งการหาลำดับการเรียกใช้งานที่พบบ่อยจะมีขั้นตอนดังภาพที่ 3.3 โดยการอ่านข้อมูลลำดับของโอเปอเรชันขึ้นมาทีละทรานแซกชัน แล้วจึงอ่านข้อมูลโอเปอเรชันไอเท็มตามลำดับที่ปรากฏในทรานแซกชันนั้นเพื่อท่องไปยังโหนดต่างๆ ในทรัย หากเป็นโอเปอเรชันไอเท็มที่ยังไม่มีอยู่ในทรัยจะทำการสร้างโหนดใหม่ขึ้นมา เมื่อถึงโอเปอเรชันไอเท็มลำดับสุดท้ายที่โหนดใด โหนดนั้นจะถือเป็นตัวแทนของข้อมูลลำดับของโอเปอเรชันในทรานแซกชันนั้น และจะทำการเพิ่มค่าสนับสนุนให้กับโหนดนั้น ในการเพิ่มลำดับของโอเปอเรชันที่มีความยาว L โดยที่ L มากกว่าหรือเท่ากับ 2 จะทำการตรวจสอบว่ามีลำดับของโอเปอเรชันที่มีความยาว L-1 อยู่บนทรัยหรือไม่ หากไม่พบลำดับที่มีความยาว L-1 นั่นแสดงว่าลำดับ L นั้นไม่ใช่ลำดับที่พบบ่อย และไม่จำเป็นต้องเพิ่มลำดับนั้นเข้าไปในทรัย หลังจากอ่านข้อมูลครบทุกทรานแซกชันแล้ว จะเข้าสู่ขั้นตอนการลบ (Prune) โหนดที่ไม่เกิดบ่อยออกจากทรัย ซึ่งก็คือโหนดที่มีค่าสนับสนุนต่ำกว่าค่าสนับสนุนขั้นต่ำที่กำหนด เพื่อให้เหลือแต่ลำดับที่เกิดบ่อยเท่านั้นบนทรัย

```
Let T = Transactional data
Let maxlength = Maximum length of transactions
for L=1 to maxlength
   for all transaction t in T do
      for startpos=0 to size of t
         itemsequence_L = {t_startpos, ..., t_startpos+L-1}
         if L = 1
            Trie.InsertAndAddSuppCount (itemsequence_L);
         else
            itemsequence_L-1 = {t_startpos, ..., t_startpos+L-2}
            if itemsequence_L-1 is frequent
               Trie.InsertAndAddSuppCount (itemsequence_L);
         end
      end
   end
   Pruning infrequent itemsequence
end
```

ภาพที่ 3.3 ขั้นตอนวิธีของเอไพรออรีเชิงทรัย

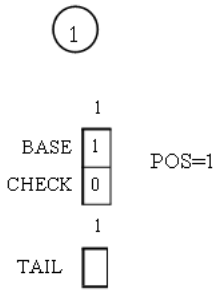
โครงสร้างข้อมูลแบบทรัยที่ใช้ในอัลกอริทึมเอไพรออรีเชิงทรัยนี้ ผู้วิจัยได้นำโครงสร้างแบบดับเบิลอาร์เรย์ทรัยมาประยุกต์ใช้ ซึ่งใช้พื้นที่ในการจัดเก็บข้อมูลน้อยกว่าทรัยแบบทั่วไป และมีการจัดการการชนกันของข้อมูล (Data Collision) โดยการคำนวณทางตัวเลขเพื่อหาตำแหน่งว่างที่สามารถวางข้อมูลเพื่อไม่ให้ข้อมูลชนกัน ซึ่งจำเป็นต้องทำการกำหนดค่าตัวเลขให้กับทุกอักขระก่อนที่จะเพิ่มข้อมูลลงในทรัย โดยที่ในงานวิจัยนี้ใช้ชื่อโอเปอเรชันแทนอักขระที่จะจัดเก็บลงในทรัย โดยการกำหนดค่าตัวเลขให้แต่ละโอเปอเรชันเป็นดังตารางที่ 3.4

ตารางที่ 3.4 ค่าตัวเลขของโอเปอเรชันที่ใช้ในทรัย

ชื่อโอเปอเรชัน	ค่าตัวเลข
#	1
CategoryNameList	2
browseCategory	3
cartCreate	4
cartAdd	5
cartGet	6
cartRemove	7
cartClear	8
calculatePrice	9
itemSearch	10
itemLookup	11
Checkout	12
Login	13
Logout	14

ในดับเบิลอาร์เรย์ทรัยจะเพิ่มอักขระพิเศษ # ใช้ในการบ่งบอกว่าเป็นอักขระสุดท้ายของชุดอักขระ ซึ่งในที่นี้ทุกทรานแซกชันจะเพิ่ม # เข้าไปต่อท้าย เพื่อแสดงว่าเป็นโอเปอเรชันลำดับสุดท้ายนั่นเอง ขั้นตอนการทำงานของอัลกอริทึมเอไพรออรีเชิงทรัย เมื่อมีการกำหนดค่าสนับสนุนขั้นต่ำเป็น 3 ครั้ง เป็นดังนี้

1. จากตารางที่ 3.3 หาค่าความยาวสูงสุดของทรานแซกชันทั้งหมด จะได้เท่ากับ 5
2. กำหนดค่าเริ่มต้นให้กับดับเบิลอาร์เรย์ทรัย ดังภาพที่ 3.4



ภาพที่ 3.4 ค่าเริ่มต้นของดับเบิลอาร์เรย์ทรัย

3. ให้ L = 1 อ่านข้อมูลชื่อโอเปอเรชันในทรานแซกชันครั้งละ 1 โอเปอเรชัน จากข้อมูลทรานแซกชันในตารางที่ 3.3 สามารถแสดงข้อมูลการเรียกใช้งานโอเปอเรชันที่ L=1 ที่จะจัดเก็บลงในทรัยได้ดังตารางที่ 3.5

ตารางที่ 3.5 การเรียกใช้งานโอเปอเรชันที่ L=1

ลำดับที่	ลำดับโอเปอเรชัน	ข้อมูลจากทรานแซกชัน
1	browseCategory, #	T1
2	itemLookup, #	T2
3	cartCreate, #	T3
4	cartAdd, #	T3
5	calculatePrice, #	T3
6	cartGet, #	T3
7	itemLookup, #	T3
8	cartCreate, #	T4
9	cartAdd, #	T4
10	calculatePrice, #	T4
11	cartGet, #	T4
12	itemLookup, #	T5
13	browseCategory, #	T6
14	cartAdd, #	T7

ตารางที่ 3.5 การเรียกใช้งานโอเปอเรชันที่ L=1 (ต่อ)

ลำดับที่	ลำดับโอเปอเรชัน	ข้อมูลจากทรานแซกชัน
15	calculatePrice, #	T7
16	cartGet, #	T7
17	itemLookup, #	T7
18	cartAdd, #	T8
19	calculatePrice, #	T8
20	cartGet, #	T8
21	itemLookup, #	T8
22	Checkout, #	T9
23	browseCategory, #	T10
24	itemLookup, #	T11
25	cartAdd, #	T12
26	calculatePrice, #	T12
27	cartGet, #	T12
28	itemLookup, #	T12
29	Checkout, #	T12

4. ทำการเพิ่มลำดับโอเปอเรชันลงในทรัยทีละลำดับ โดยการหาตำแหน่งที่จะทำการบันทึกข้อมูลทีละโอเปอเรชัน ซึ่งสามารถคำนวณได้จาก

basepos = 1

Index = BASE[basepos] + ‘c’

โดยที่ c คือค่าตัวเลขแทนโอเปอเรชันที่ทำการเพิ่ม (คือ 1-12 ตามตารางที่ 3.4)

Index คือตำแหน่งที่จะทำการบันทึกข้อมูล

ซึ่งมีลักษณะการตรวจสอบ 2 ลักษณะด้วยกัน

ก. หากค่า CHECK[Index] มีค่าเท่ากับ 0 หมายความว่ายังไม่มีชื่อโอเปอเรชันนั้นอยู่ในทรัย ให้ทำการเพิ่มชื่อโอเปอเรชันนั้นลงในทรัยโดยการกำหนดค่า BASE[Index] = –POS และกำหนดค่า CHECK[Index] = 1 และเพิ่มค่าสนับสนุนเป็น 1

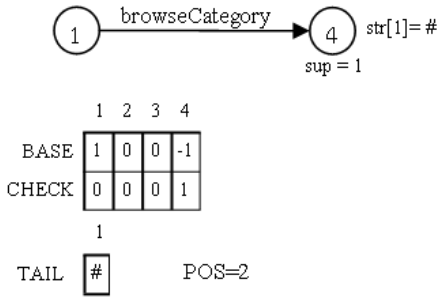
ข. หากค่า CHECK[Index] มีค่าเท่ากับ basepos นั่นคือ 1 และ ลำดับส่วนที่เหลือทั้งหมดของลำดับที่จะทำการเพิ่มมีค่าเท่ากับลำดับที่เก็บอยู่ใน TAIL[POS] หมายความว่ามีชื่อโอเปอเรชันนั้นอยู่ในทรัยแล้ว ให้ทำการเพิ่มค่าสนับสนุนสำหรับชื่อโอเปอเรชันนั้นอีก 1

จากตารางที่ 3.5 ทำการเพิ่มลำดับ browseCategory, # เป็นลำดับแรกเข้าไปในทรัย โดยการหาตำแหน่งที่จะจัดเก็บข้อมูล คำนวณจาก

Index = BASE[1] + ‘browseCategory’

Index = 1 + 3 = 4

เนื่องจาก CHECK[4] มีค่าเท่ากับ 0 นั่นหมายความว่ายังไม่มีลำดับ browseCategory อยู่บนทรัย จึงทำการกำหนดค่า CHECK[4] เท่ากับ 1 และ BASE[4] = -POS นั่นคือค่า -1 และจัดเก็บลำดับส่วนที่เหลือ นั่นคือ # ไว้ใน TAIL[POS] แล้วจึงเพิ่มค่า POS ขึ้นอีก 1 ดังแสดงในภาพที่ 3.5



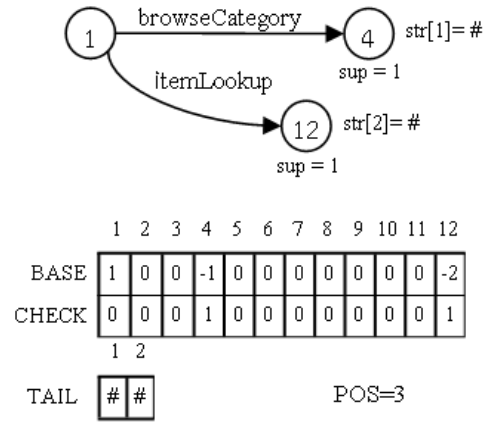
ภาพที่ 3.5 ดับเบิลอาร์เรย์ทรัยหลังการเพิ่มลำดับ browseCategory

จากนั้นทำการเพิ่มลำดับ itemLookup, # เข้าไปในทรัย ทำการคำนวณหาตำแหน่งที่จัดเก็บข้อมูล

Index = BASE[1] + ‘itemLookup’

Index = 1 + 11 = 12

เนื่องจาก CHECK[12] มีค่าเป็น 0 หมายความว่ายังไม่มีลำดับ itemLookup อยู่บนทรัย จึงทำการกำหนดค่า CHECK[12] เท่ากับ 1 และ BASE[12] = -POS นั่นคือค่า -2 และจัดเก็บลำดับส่วนที่เหลือ นั่นคือ # ไว้ใน TAIL[POS] แล้วจึงเพิ่มค่า POS ขึ้นอีก 1 ดังแสดงในภาพที่ 3.6



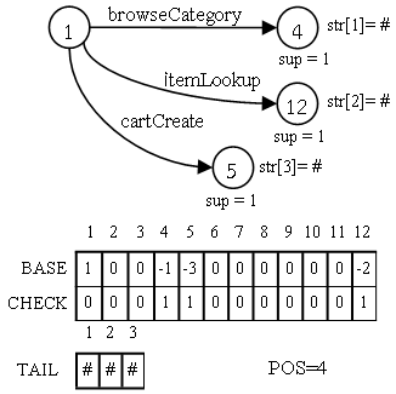
ภาพที่ 3.6 ดับเบิลอาร์เรย์ทรัยหลังการเพิ่มลำดับ itemLookup

ขั้นต่อไปทำการเพิ่มลำดับ cartCreate, # เข้าไปในทรัย ทำการคำนวณหาตำแหน่งที่จัดเก็บข้อมูล

Index = BASE[1] + ‘cartCreate’

Index = 1 + 4 = 5

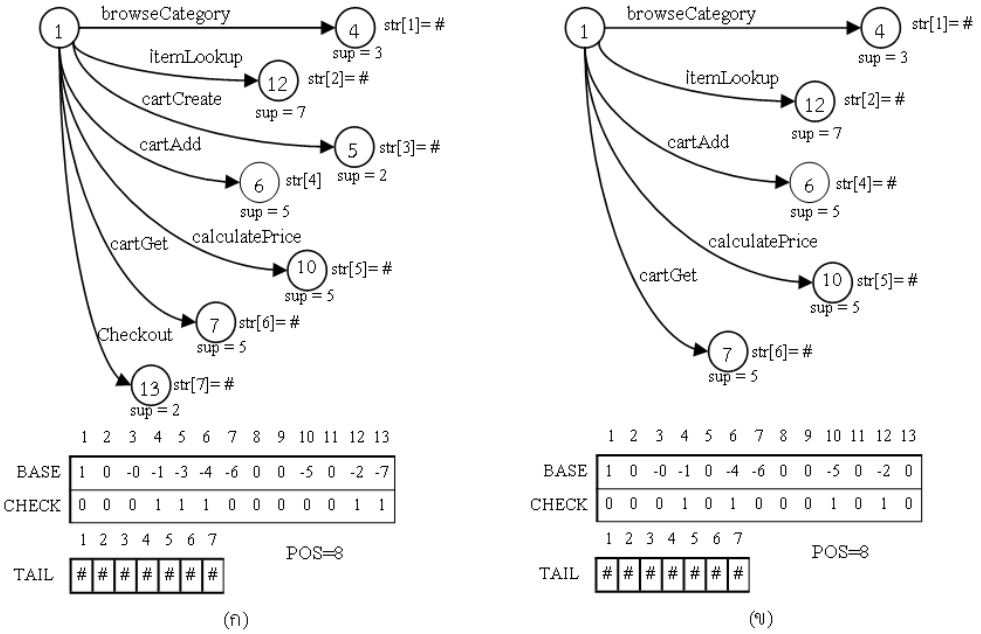
เนื่องจาก CHECK[5] มีค่าเป็น 0 หมายความว่ายังไม่มีลำดับ cartCreate อยู่บนทรัย จึงทำการกำหนดค่า CHECK[5] เท่ากับ 1 และ BASE[5] = -POS นั่นคือค่า -3 และจัดเก็บลำดับส่วนที่เหลือนั่นคือ # ไว้ใน TAIL[POS] แล้วจึงเพิ่มค่า POS ขึ้นอีก 1 ดังแสดงภาพที่ 3.7



ภาพที่ 3.7 ดับเบิลอาร์เรย์ทรัยหลังจากการเพิ่มลำดับ cartCreate

การเพิ่มลำดับอื่น ๆ ที่เหลือในตารางที่ 3.5 ดำเนินการได้ในลักษณะเดียวกัน เมื่อเพิ่มลำดับทุกลำดับแล้ว สามารถแสดงข้อมูลลำดับการเรียกใช้งานในรูปของทรัยได้ดังภาพที่ 3.8 (ก)

5. ตรวจสอบค่าสนับสนุนของโหนดใบทุกโหนดกับค่าสนับสนุนขั้นต่ำ เพื่อลบ (Prune) ลำดับที่ไม่เกิดบ่อยออกจากทรัย นั่นคือ หากโหนดใดมีค่าสนับสนุนต่ำกว่าค่าสนับสนุนขั้นต่ำที่กำหนดไว้คือ 3 จะถูกลบออกจากทรัย จากภาพที่ 3.8 (ก) พบว่าลำดับ cartCreate และ Checkout มีค่าสนับสนุนต่ำกว่าค่าสนับสนุนขั้นต่ำที่กำหนด จึงทำการลบออกจากทรัย ดังภาพที่ 3.8 (ข)



ภาพที่ 3.8 ดับเบิลอาร์เรย์ทรัยของลำดับโอเปอเรชันที่ L=1

6. เพิ่มค่า L ขึ้นอีก 1 แล้วเริ่มอ่านข้อมูลใหม่ตั้งแต่ทรานแซกชันแรก โดยอ่านข้อมูลชื่อโอเปอเรชันในทรานแซกชันครั้งละ L โอเปอเรชัน ที่ L = 2 จะได้ลำดับโอเปอเรชันดังแสดงในตารางที่ 3.6

ตารางที่ 3.6 การเรียกใช้งานโอเปอเรชันที่ L=2

ลำดับที่	ลำดับโอเปอเรชัน	ข้อมูลจากทรานแซกชัน
1	cartCreate, cartAdd, #	T3
2	cartAdd, calculatePrice, #	T3
3	calculatePrice, cartGet, #	T3
4	cartGet, itemLookup, #	T3
5	cartCreate, cartAdd, #	T4
6	cartAdd, calculatePrice, #	T4
7	calculatePrice, cartGet, #	T4
8	cartAdd, calculatePrice, #	T7
9	calculatePrice, cartGet, #	T7
10	cartGet, itemLookup, #	T7
11	cartAdd, calculatePrice , #	T8
12	calculatePrice, carrtGet, #	T8

ตารางที่ 3.6 การเรียกใช้งานโอเปอเรชันที่ L=2 (ต่อ)

ลำดับที่	ลำดับโอเปอเรชัน	ข้อมูลจากทรานแซกชัน
13	cartGet, itemLookup, #	T8
14	cartAdd, calculatePrice,	T12
15	calculatePrice, cartGet ;#	T12
16	cartGet,,itemLookup, #	T12
17	itemLookup, Checkout	T12

7. ทำการเพิ่มลำดับโอเปอเรชันลงในทรัยทีละลำดับ โดยทำการตรวจสอบลำดับ L-1 ว่ามีอยู่บนทรัยหรือไม่ โดยใช้วิธีคำนวณเช่นเดียวกับข้อ 4. ซึ่งมีลักษณะการตรวจสอบ 2 ลักษณะด้วยกัน

ก. หาก CHECK[Index] มีค่าเท่ากับ basepos และ BASE[Index] มีค่าน้อยกว่า 0 นั่นคือพบลำดับ L-1 อยู่บนทรัย ให้ดำเนินการต่อไปนี้

i. หากส่วนที่เหลือของลำดับที่กำลังเพิ่มกับลำดับที่อยู่ใน TAIL[BASE[Index]] มีค่าเท่ากันให้เพิ่มค่าสนับสนุนสำหรับลำดับนั้น

ii. หากส่วนที่เหลือของลำดับที่กำลังเพิ่มกับลำดับที่อยู่ใน TAIL[BASE[Index]] มีค่าไม่เท่ากัน จำเป็นต้องคำนวณหาตำแหน่งที่ว่างที่เหมาะสมเพื่อทำการเพิ่มลำดับและเพิ่มค่าสนับสนุนต่อไป โดยการหาค่า q ซึ่ง q>0 ที่ทำให้ CHECK[q+c] = 0 สำหรับทุกค่า c โดยที่ c คือส่วนที่เหลือของลำดับที่กำลังเพิ่ม และ ลำดับที่อยู่ใน TAIL

ข. หาก CHECK[Index] มีค่าเป็น 0 นั่นคือไม่พบว่ามีลำดับ L-1 อยู่บนทรัย แสดงว่าลำดับที่ทำการเพิ่มนี้ไม่ใช่ลำดับที่เกิดบ่อย จึงไม่ต้องทำการเพิ่มลงในทรัย

อนึ่ง การคำนวณหาตำแหน่งที่ว่างที่เหมาะสมและการจัดการการชนกันของข้อมูลระหว่างการเพิ่มข้อมูลลงในทรัยมีรายละเอียดค่อนข้างมาก ดังนั้นผู้วิจัยจึงไม่ได้อธิบายรวมอยู่ในงานวิจัยนี้ ผู้ที่สนใจสามารถศึกษารายละเอียดได้จาก [**3**]

จากตารางที่ 3.6 การเพิ่มลำดับ cartCreate, cartAdd, # เข้าในทรัยมีขั้นตอนดังต่อไปนี้

basepos = 1

Index = BASE[basepos] + ‘cartCreate’

Index = 1 + 4 = 5

จากภาพที่ 3.8(ข) พบว่าค่า CHECK[5] มีค่าเป็น 0 นั่นแสดงว่าลำดับที่เริ่มต้นด้วย cartCreate ไม่ใช่ลำดับที่พบบ่อย จึงไม่ต้องทำการเพิ่มลำดับ cartCreate, cartAdd, #เข้าไปในทรัย

ลำดับต่อไปที่ทำการเพิ่มคือ cartAdd, calculatePrice, # ซึ่งมีขั้นตอนดังต่อไปนี้

basepos = 1

Index = BASE[basepos] + ‘cartAdd’

Index = 1 + 5 = 6

จากภาพที่ 3.8(ข) พบว่าค่า CHECK[6] มีค่าเป็น 1 แสดงว่ามีลำดับ cartAdd อยู่บนทรัยแล้ว และ BASE[6] มีค่า -4 แสดงว่าส่วนที่เหลือของลำดับ cartAdd ที่อยู่บนทรัยถูกจัดเก็บไว้ใน TAIL และมีค่าเป็น # ซึ่งไม่เท่ากับ calculatePrice จึงจำเป็นต้องคำนวณหาที่ว่างเพื่อทำการจัดเก็บลำดับทั้งสอง โดยการหาค่า q ที่ทำให้ CHECK[q+ ‘#’] = 0 และ CHECK[q+ ‘calculatePrice’] = 0 ให้ q=1 จะได้

สำหรับ # : CHECK[q+ ‘#’] = CHECK[1+1] = CHECK[2] = 0

สำหรับ calculatePrice : CHECK[q+ ‘calculatePrice’]

= CHECK[1+9] =CHECK[10] =1

ให้ q=2 จะได้

สำหรับ # : CHECK[q+ ‘#’] = CHECK[2+1] = CHECK[3] = 0

สำหรับ calculatePrice : CHECK[q+ ‘calculatePrice’]= CHECK[2+9]

=CHECK[11] =0

ที่ q = 2 สามารถทำการเพิ่ม # และ calculatePrice, # ได้พร้อมกัน ซึ่งสามารถคำนวณได้ดังนี้

basepos = 6

TEMP = BASE[basepos] = -4

BASE[basepos] = q = 2

สำหรับ #

Index = BASE[basepos] + ‘#’ = 2+1 = 3 ; CHECK[3] = 0

CHECK[3] = basepose =6

BASE[3] = TEMP = -4

สำหรับ calculatePrice, #

Index = BASE[basepos] + ‘calculatePrice’ = 2 + 9 = 11 ; CHECK[11] =0

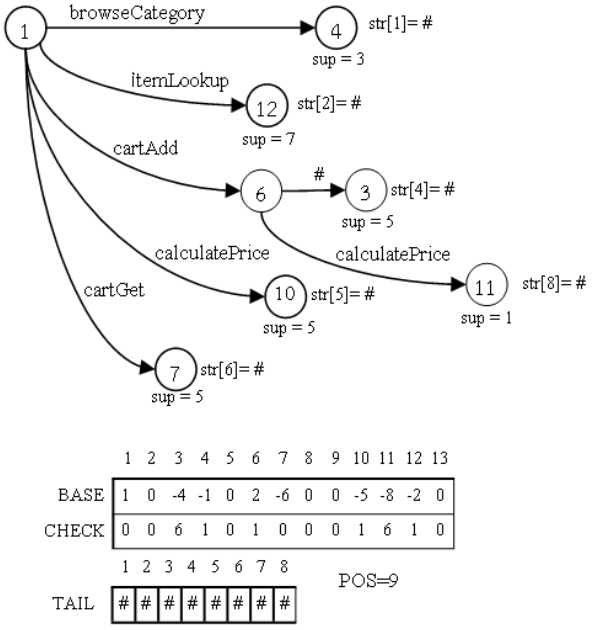
CHECK[11] = basepos = 6

BASE[11] = -POS = -8

TAIL[8] = #

POS = POS+1

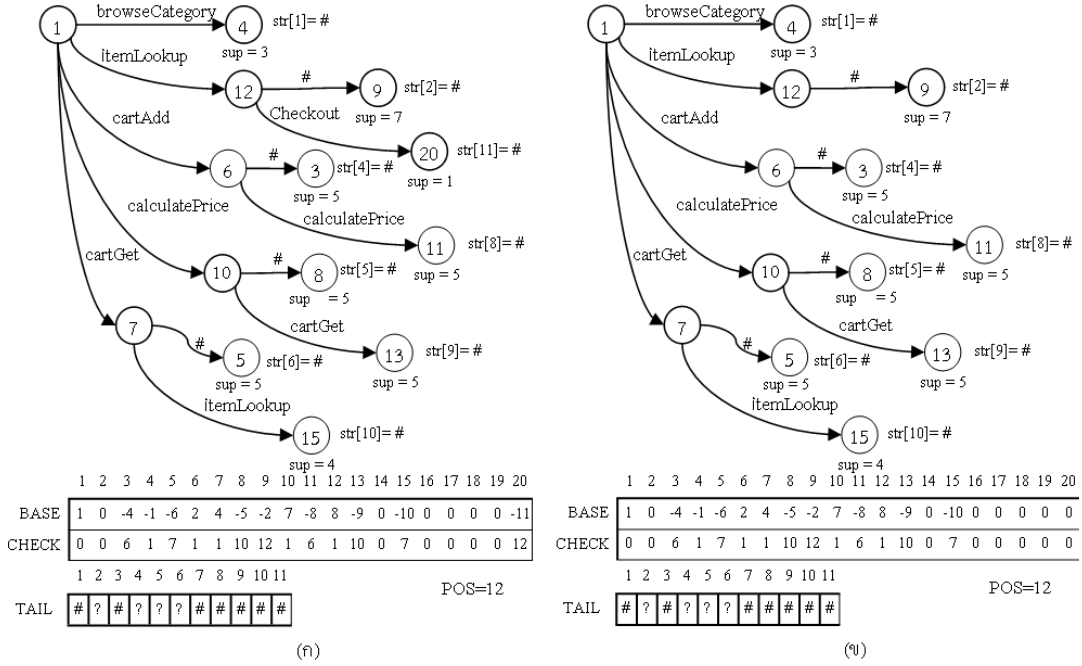
จากการคำนวณข้างต้น จะได้ดับเบิลอาร์เรย์ทรัย ดังภาพที่ 3.9



ภาพที่ 3.9 ดับเบิลอาร์เรย์ทรัยหลังจากการเพิ่มลำดับ cartAdd, calculatePrice

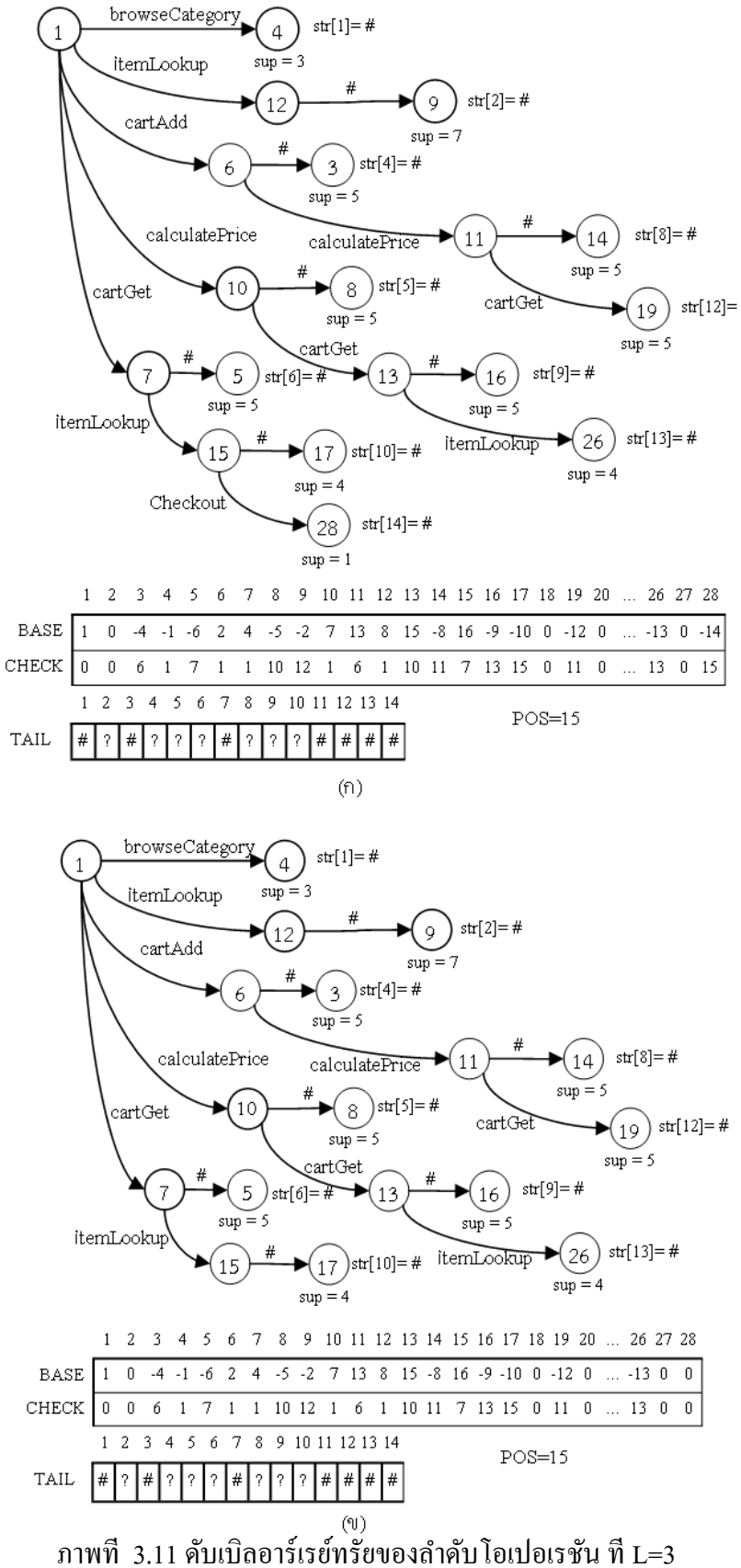
การเพิ่มลำดับอื่น ๆ ที่ L=2 ที่เหลืออยู่ในตารางที่ 3.6 จะดำเนินการด้วยวิธีเดียวกันนี้ ซึ่งหลังจากที่อ่านข้อมูลครบทุกทรานแซกชันแล้ว สามารถแสดงข้อมูลลำดับการเรียกใช้งานในรูปของทรัยได้ดังภาพที่ 3.10 (ก)

8. ตรวจสอบค่าสนับสนุนของโหนดใบทุกโหนดกับค่าสนับสนุนขั้นต่ำ จากภาพที่ 3.10 (ก) พบว่าโหนดที่ 20 ซึ่งเป็นตัวแทนของลำดับ itemLookup, Checkout มีค่าสนับสนุนน้อยกว่า 3 ซึ่งหมายความว่าลำดับ itemLookup, Checkout ไม่ใช่ลำดับที่เกิดบ่อย จึงทำการลบโหนดที่ 20 ออกจากทรัย จึงทำให้เหลือทรัย ดังภาพที่ 3.10 (ข)

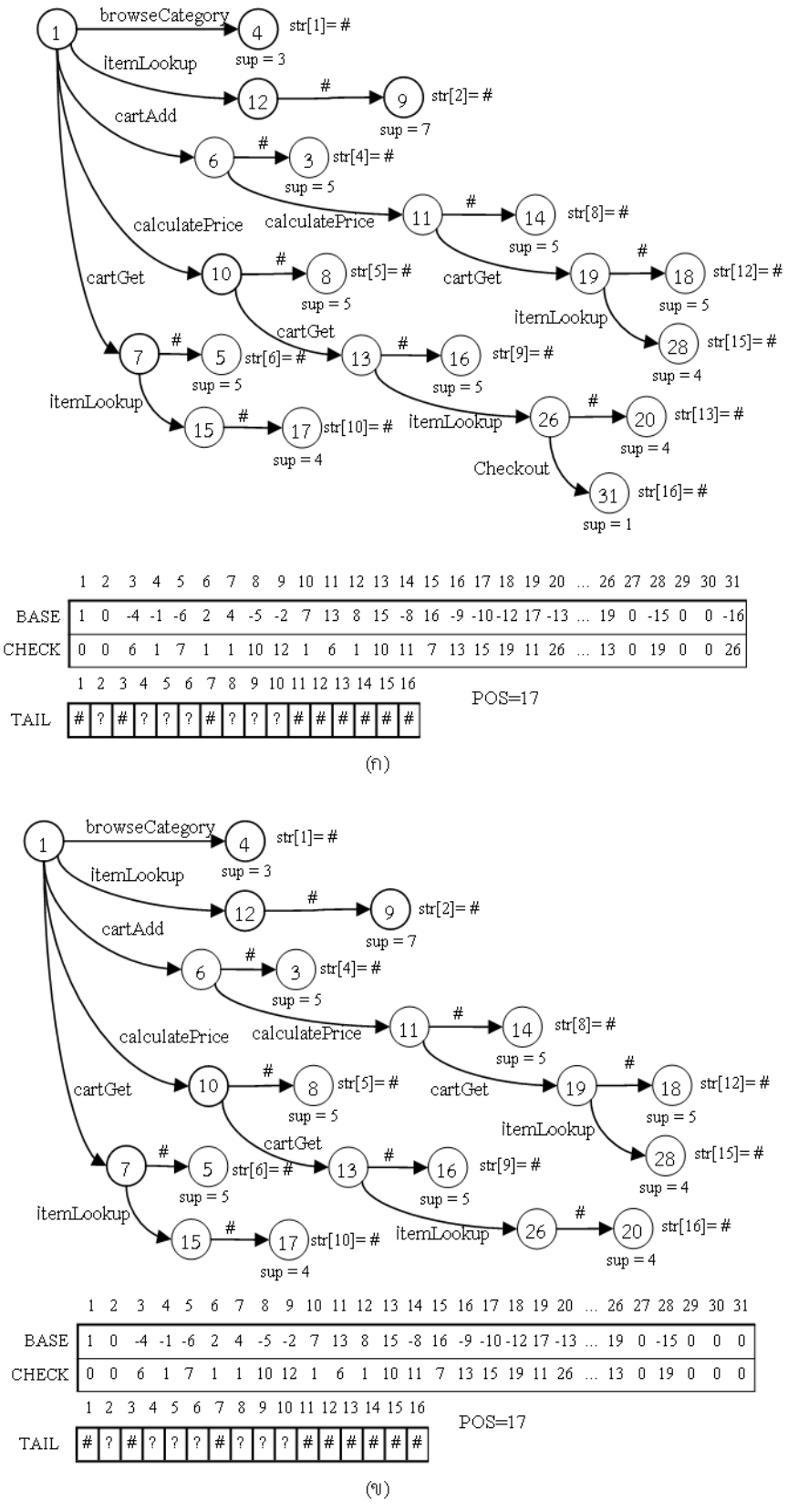


ภาพที่ 3.10 ดับเบิลอาร์เรย์ทรัยของลำดับโอเปอเรชันที่ L=2

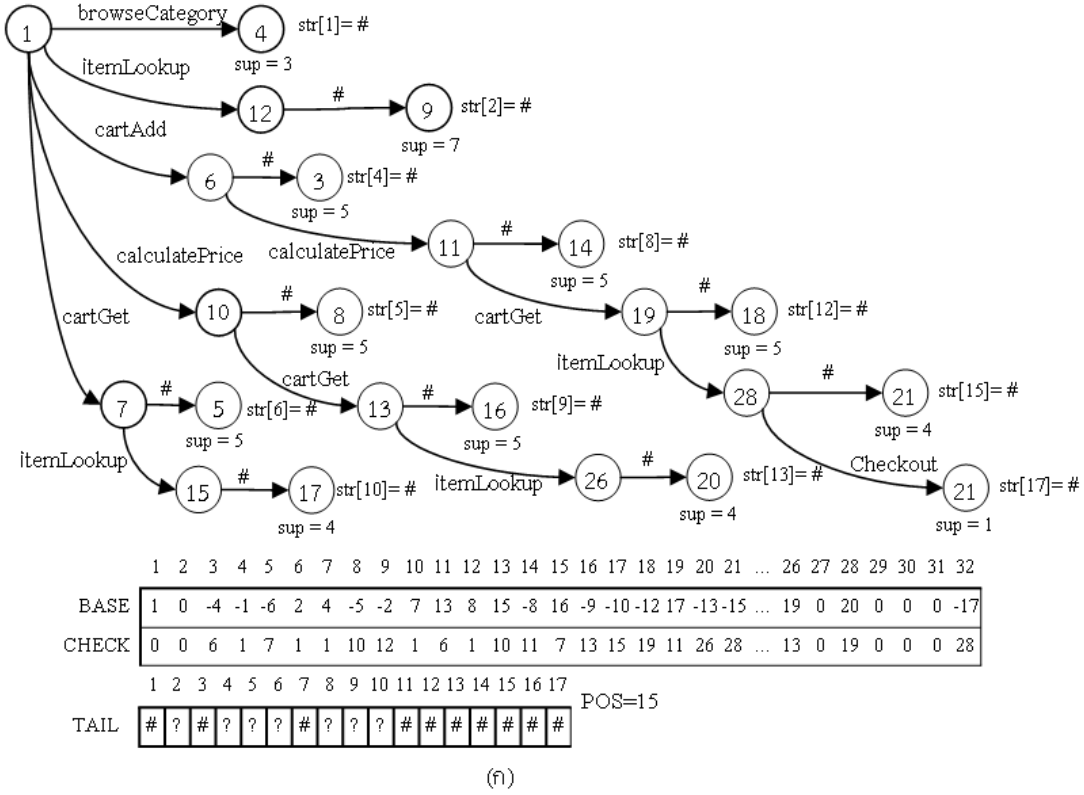
9. ดำเนินการตามข้อ 4 และ 5 ซ้ำจนกว่าค่า L จะเท่ากับค่าความยาวสูงสุดของทรานแซกชัน นั่นคือ L เท่ากับ 5 จึงหยุดดำเนินการ ดังแสดงในภาพที่ 3.11 – 3.13 โดย (ก) หมายถึงดับเบิลอาร์เรย์ทรัยที่ได้เมื่อ L = 3, 4, 5 ตามลำดับ และ (ข) หมายถึงดับเบิลอาร์เรย์ทรัยที่ได้จากการลบลำดับที่ไม่เกิดบ่อยออกไปแล้ว



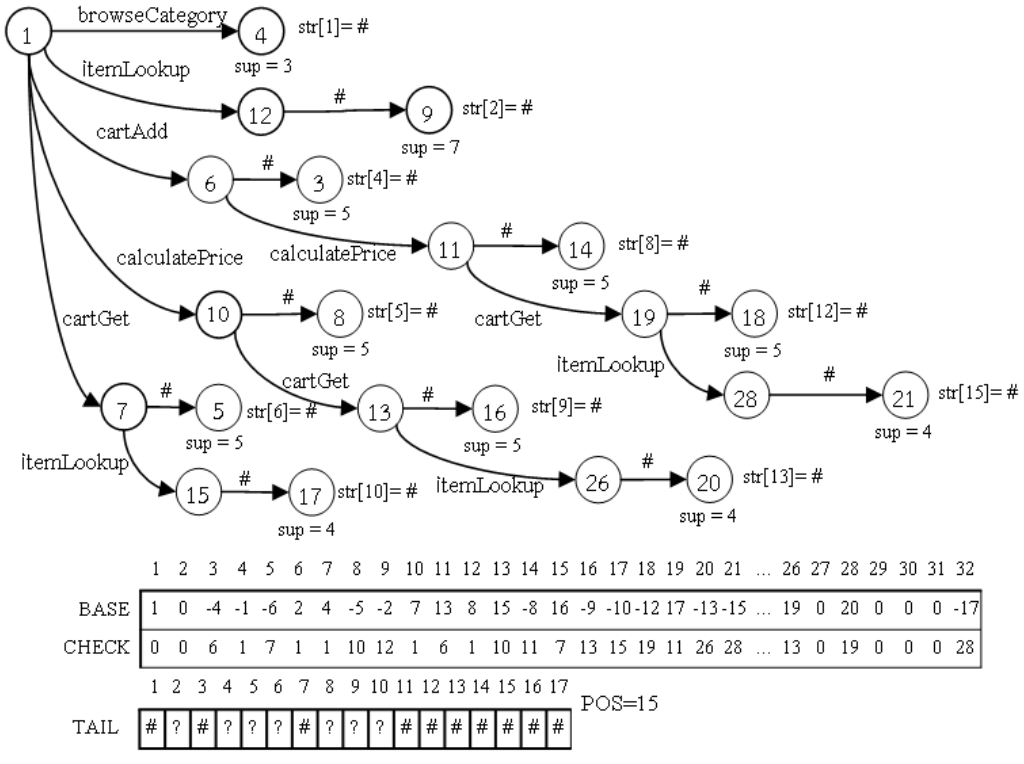
ภาพที่ 3.11 ดับเบิลอาร์เรย์ทรัยของลำดับโอเปอเรชัน ที่ L=3



ภาพที่ 3.12 ดับเบิลอาร์เรย์ทรัยของลำดับโอเปอเรชัน ที่ L=4



(ก)



(ข)

ภาพที่ 3.13 ดับเบิลอาร์เรย์ทรัยของลำดับโอเปอเรชัน ที่ L=5

ลำดับทั้งหมดที่เหลืออยู่บนดับเบิลอาร์เรย์ทรัยในภาพที่ 3.13 (ข) คือลำดับของโอเปอเรชันที่เกิดบ่อยของข้อมูลทรานแซกชันทั้งหมด สามารถสรุปได้ดังตารางที่ 3.7 ลำดับเหล่านี้จะถูกนำไปสร้างเป็นกฎความสัมพันธ์ในขั้นตอนต่อไป

ตารางที่ 3.7 ลำดับของโอเปอเรชันที่เกิดบ่อย

ลำดับที่เกิดบ่อย	ค่าสนับสนุน
browseCategory	3/12 = 25.00%
itemLookup	7/12 = 58.33%
cartAdd	5/12 = 41.66%
calculatePrice	5/12 = 41.66%
cartGet	5/12 = 41.66%
cartAdd → calculatePrice	5/12 = 41.66%
calculatePrice → cartGet	5/12 = 41.66%
cartGet → itemLookup	4/12 = 33.33%
cartAdd → calculatePrice → cartGet	5/12 = 41.66%
calculatePrice → cartGet → itemLookup	4/12 = 33.33%
cartAdd → calculatePrice → cartGet → itemLookup	4/12 = 33.33%

3.6 การสร้างกฎความสัมพันธ์จากลำดับที่พบบ่อย

จากลำดับของโอเปอเรชันที่พบบ่อยทั้งหมดจะนำมาใช้ในการสร้างกฎความสัมพันธ์ระหว่างโอเปอเรชันที่ถูกเรียกใช้ต่อเนื่องกันได้ ซึ่งอยู่ในรูป $s_i \rightarrow I\text{-}s_i$ โดยสร้างจากลำดับย่อยของโอเปอเรชัน (Sub-operation-sequence: s) ของลำดับที่พบบ่อย I และหาค่าความเชื่อมั่นของกฎความสัมพันธ์ โดยที่มีขั้นตอนดังต่อไปนี้

1. ตรวจสอบลำดับย่อย (s) ของลำดับที่พบบ่อย (I) โดยที่ s_i คือ ลำดับย่อยตั้งแต่ลำดับที่ 1 จนถึง i แล้วสร้างกฎความสัมพันธ์ในรูป $s_i \rightarrow I\text{-}s_i$ โดยที่ $I\text{-}s_i$ คือลำดับย่อยตั้งแต่ลำดับที่ i+1 จนถึงลำดับสุดท้ายของลำดับที่พบบ่อย I

2. คำนวณหาค่าความเชื่อมั่นของกฎความสัมพันธ์ โดยคิดจากค่าสนับสนุนของ I หารด้วยค่าสนับสนุนของ s_i

จากลำดับที่พบบ่อยที่ได้จากขั้นตอนที่ผ่านมาตามตารางที่ 3.7 จะพิจารณาสร้างกฎความสัมพันธ์จากลำดับที่ประกอบด้วยสองโอเปอเรชันขึ้นไปเท่านั้นตามเป้าหมายของการรวมโอเปอเรชันที่พบด้วยกันบ่อยเข้าด้วยกัน กฎความสัมพันธ์ที่ได้จะแสดงในตารางที่ 3.8

ตารางที่ 3.8 กฎความสัมพันธ์ที่ได้จากลำดับที่พบบ่อย

กฎความสัมพันธ์	ค่าความเชื่อมั่น
{cartAdd} → {calculatePrice}	5/5 = 100%
{calculatePrice} → {cartGet}	5/5 = 100%
{cartGet} → {itemLookup}	4/5 = 80%
{cartAdd} → {calculatePrice → cartGet}	5/5 = 100%
{cartAdd → calculatePrice} → {cartGet}	5/5 = 100%
{calculatePrice} → {cartGet → itemLookup}	4/5 = 80%
{calculatePrice → cartGet} → {itemLookup}	4/5 = 80%
{cartAdd} → {calculatePrice → cartGet → itemLookup}	4/5 = 80%
{cartAdd → calculatePrice} → {cartGet → itemLookup}	4/5 = 80%
{cartAdd → calculatePrice → cartGet} → {itemLookup}	4/5 =80%

แต่เนื่องจากจะได้กฎความสัมพันธ์เป็นจำนวนมาก ผู้ออกแบบเซอร์วิซสามารถกำหนดความเชื่อมั่นขั้นต่ำ (Minimum Confidence Threshold) เพื่อเลือกเฉพาะกฎความสัมพันธ์ที่น่าสนใจ (Strong Rule) ซึ่งมีค่าความเชื่อมั่นไม่น้อยกว่าค่าขั้นต่ำมาใช้เท่านั้น จากตารางที่ 3.8 หากผู้ออกแบบเซอร์วิซกำหนดค่าความเชื่อมั่นขั้นต่ำไว้ที่ร้อยละ 90 จะเหลือกฎความสัมพันธ์ที่น่าสนใจดังแสดงในตารางที่ 3.9

ตารางที่ 3.9 กฎความสัมพันธ์ที่น่าสนใจ

กฎความสัมพันธ์	ค่าสนับสนุน	ค่าความเชื่อมั่น
{cartAdd} → {calculatePrice}	5/12 = 41.66%	5/5 = 100%
{calculatePrice} → {cartGet}	5/12 = 41.66%	5/5 = 100%
{cartAdd} → {calculatePrice → cartGet}	5/12 = 41.66%	5/5 = 100%
{cartAdd → calculatePrice} → {cartGet}	5/12 = 41.66%	5/5 = 100%

กฎความสัมพันธ์ที่น่าสนใจเหล่านี้ จะถูกนำไปใช้ในการแนะนำการรวมกลุ่มโอเปอเรชันให้ผู้ออกแบบเซอร์วิซในขั้นตอนถัดไป

3.7 การแนะนำการรวมกลุ่มโอเปอเรชัน

การพิจารณาการรวมกลุ่มโอเปอเรชัน จะต้องทำการรวมโดยไม่ส่งผลกระทบเชิงลบต่อประสิทธิภาพการทำงานของเซอร์วิซ ลำดับการเรียกใช้งานที่พบบ่อยนั้นอาจมีบางลำดับที่หากทำการรวมกลุ่มแล้ว อาจส่งผลกระทบต่อการทำงานของลำดับอื่น ๆ ดังนั้นเพื่อไม่ให้การรวมกลุ่มของลำดับการเรียกใช้งานใด ๆ ส่งผลกระทบต่อลำดับการเรียกใช้งานอื่น ๆ กฎความสัมพันธ์ที่น่าสนใจทั้งหมดจะถูกนำมาพิจารณาเพื่อหาความเหมาะสมในการรวมกลุ่มโอเปอเรชัน โดยทำการพิจารณาลำดับย่อยซึ่งปรากฏอยู่ในกฎความสัมพันธ์ส่วนใหญ่แล้วทำการจัดอันดับ (Rank) ตามจำนวนที่ปรากฏ (Count) เพื่อแนะนำให้ผู้ออกแบบเว็บเซอร์วิซพิจารณาความเป็นไปได้ในการรวมโอเปอเรชันในลำดับย่อยนี้เข้าด้วยกันเป็นโอเปอเรชันที่ใหญ่ขึ้น

จากกฎความสัมพันธ์ที่น่าสนใจในตารางที่ 3.9 ลำดับของโอเปอเรชันที่แนะนำให้รวมกันเป็นดังตารางที่ 3.10 จะเห็นได้ว่าลำดับ cartAdd → calculatePrice และลำดับ calculatePrice → cartGet ถูกจัดอันดับให้อยู่ใน 2 อันดับแรกของคำแนะนำ เนื่องจากเป็นลำดับย่อยที่ปรากฏในกฎความสัมพันธ์ที่น่าสนใจจำนวน 3 กฎความสัมพันธ์ด้วยกัน

ตารางที่ 3.10 ลำดับโอเปอเรชันที่แนะนำให้รวมกัน

อันดับ	ลำดับที่แนะนำให้รวมกัน	จำนวนครั้งที่พบ
1	cartAdd → calculatePrice	3
2	calculatePrice → cartGet	3
3	cartAdd → calculatePrice → cartGet	2

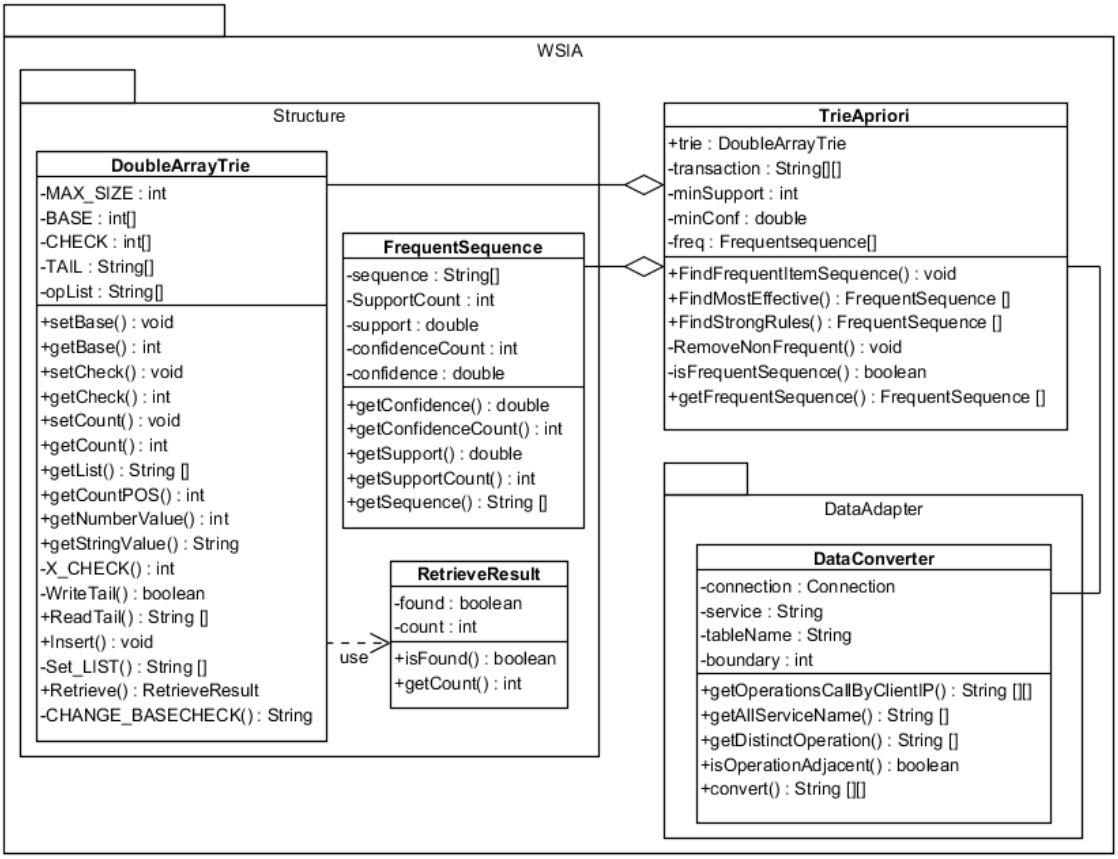
อย่างไรก็ตาม ลำดับของโอเปอเรชันที่แนะนำให้รวมกันนี้ ไม่ได้เป็นการแนะนำให้รวมโอเปอเรชันดังกล่าวแทนที่โอเปอเรชันเดิม แต่เป็นการแนะนำการรวมกันเพื่อสร้างเป็นโอเปอเรชันใหม่ที่มีขนาดความสามารถใหญ่ขึ้นสำหรับเว็บเซอร์วิซเดิม หรือสำหรับสร้างเว็บเซอร์วิซเวอร์ชันใหม่ เพื่อเป็นทางเลือกให้กับผู้ใช้งานที่ต้องการลดจำนวนครั้งในการเรียกใช้งานลง โดยที่ไม่กระทบต่อผู้ใช้งานที่เรียกใช้แบบเดิมอยู่

3.8 การพัฒนาเครื่องมือสนับสนุนการแนะนำผู้ออกแบบเซอร์วิซ

ผู้วิจัยทำการพัฒนาเครื่องมือสนับสนุนการแนะนำในรูปแบบเว็บแอพพลิเคชันโดยใช้ภาษาจาวาในการพัฒนา ซึ่งมีขั้นตอนการทำงานดังภาพที่ 3.14

ภาพที่ 3.14 ขั้นตอนการทำงานของเครื่องมือสนับสนุนการแนะนำการรวมกลุ่มเซอร์วิซ

ในส่วนของรายละเอียดการออกแบบเครื่องมือสนับสนุนการแนะนำการรวมกลุ่มเซอร์วิซสามารถแสดงได้ดังภาพที่ 3.15



ภาพที่ 3.15 คลาสไดอะแกรมของเครื่องมือสนับสนุนการแนะนำการรวมกลุ่มเซอร์วิซ

คลาส DataConverter ทำหน้าที่ในการติดต่อกับฐานข้อมูล เพื่อดึงข้อมูลการเรียกใช้งานเซอร์วิซ ตามชื่อเซอร์วิซและชื่อตารางบันทึกข้อมูลการเรียกใช้งานที่ได้รับค่าเข้ามา โดยจะทำการตรวจสอบโครงสร้างของตารางบันทึกข้อมูลการเรียกใช้งานว่ามีโครงสร้างถูกต้องหรือไม่ หากถูกต้องจะทำการแปลงข้อมูลให้อยู่ในรูปของข้อมูลทรานแซกชันตามค่าช่วงเวลา boundary ที่

ผู้ใช้งานกำหนดโดยเรียกใช้งานเมท็อด convert ในการพิจารณาว่าข้อมูลการเรียกใช้งานเป็นทรานแซกชันเดียวกันหรือไม่ จะทำการตรวจสอบโดยเมท็อด isOperationAdjacent หากสองโอเปอเรชันของผู้ใช้งานรายเดียวกันมีช่วงเวลาระหว่างการตอบกลับของโอเปอเรชันหนึ่งกับการเริ่มต้นทำงานของอีกโอเปอเรชันหนึ่งซึ่งมีค่าน้อยกว่าหรือเท่ากับค่า boundary จะถือว่าโอเปอเรชันทั้งสองถูกเรียกใช้ต่อเนื่องกันและจะจัดให้อยู่ในทรานแซกชันเดียวกัน หากมากกว่า จะจัดให้อยู่ต่างทรานแซกชันกัน

คลาส DoubleArrayTrie เป็นโครงสร้างหลักที่ใช้ในการเก็บข้อมูลทรัยแบบดับเบิลอาร์เรย์ทรัยที่นำมาใช้ในการเก็บข้อมูลลำดับการเรียกใช้งาน ประกอบด้วยอาร์เรย์ BASE และ CHECK สำหรับการเพิ่มลำดับลงในทรัยจะเรียกใช้ผ่านเมท็อด Insert สำหรับทุกการเพิ่มลำดับ L โดยที่ L มากกว่า 1 จะทำการตรวจสอบว่ามีลำดับ L-1 อยู่บนทรัยหรือไม่โดยการเรียกใช้เมท็อด Retrieve หากพบว่ามีลำดับขนาด L-1 อยู่ จึงจะทำการเพิ่มลำดับนั้นลงในทรัย สำหรับกรณีการเพิ่มลำดับลงในทรัยแล้วเกิดการชนกันของข้อมูล จะมีการเรียกใช้งานเมท็อด X_CHECK เพื่อตรวจสอบหาตำแหน่งที่ว่างที่ทำให้การชนกันของข้อมูลหมดไป แล้วใช้เมท็อด CHANGE_BASECHECK ในการย้ายข้อมูลลำดับที่ทำให้เกิดการชนกันของข้อมูลออกไปยังตำแหน่งใหม่

คลาส RetrieveResult เป็นโครงสร้างข้อมูลที่ใช้สำหรับแสดงสถานะการมีอยู่ของลำดับที่ทำการค้นหา และค่าสนับสนุนของลำดับนั้น ๆ

คลาส FrequentSequence ใช้ในการเก็บลำดับที่พบบ่อย โดยเก็บลำดับไว้ในแอตทริบิวต์ sequence โดยสามารถบอกถึงค่าสนับสนุน และค่าความเชื่อมั่นของลำดับนั้น ๆ ได้

คลาส TrieApriori เป็นคลาสหลักที่ใช้ในการหากฎความสัมพันธ์ด้วยวิธีเอไพรออรีเชิงทรัย ซึ่งมีส่วนประกอบที่สำคัญได้แก่ลำดับการเรียกใช้โอเปอเรชันที่จัดเก็บในแอตทริบิวต์ trie ซึ่งมีลักษณะข้อมูลตามคลาส DoubleArrayTrie เมท็อด FindFrequentItemSequence จะใช้ข้อมูลทรานแซกชันที่ได้จากเมท็อด convert ของคลาส DataConverter โดยการอ่านค่าทรานแซกชันทีละทรานแซกชัน แล้วเพิ่มข้อมูลลงในทรัยจนครบทุกทรานแซกชันแล้ว จะทำการตัดลำดับที่ไม่เกิดบ่อยออกจากทรัยโดยเมท็อด RemoveNonFrequent เพื่อให้เหลือแต่ลำดับที่เกิดบ่อยเท่านั้น จากนั้นจึงทำการหากฎความสัมพันธ์ที่น่าสนใจโดยเมท็อด FindStrongRules ซึ่งทำการเลือกเฉพาะกฎความสัมพันธ์ที่มีค่าความเชื่อมั่นสูงกว่าหรือเท่ากับค่าความเชื่อมั่นขั้นต่ำที่กำหนด จากนั้นจะทำการจัดอันดับโอเปอเรชันที่ควรจะรวมกันโดยเรียงลำดับตามจำนวนครั้งที่พบในลำดับที่แนะนำให้รวมกันทั้งหมดจากมากไปหาน้อย

บทที่ 4

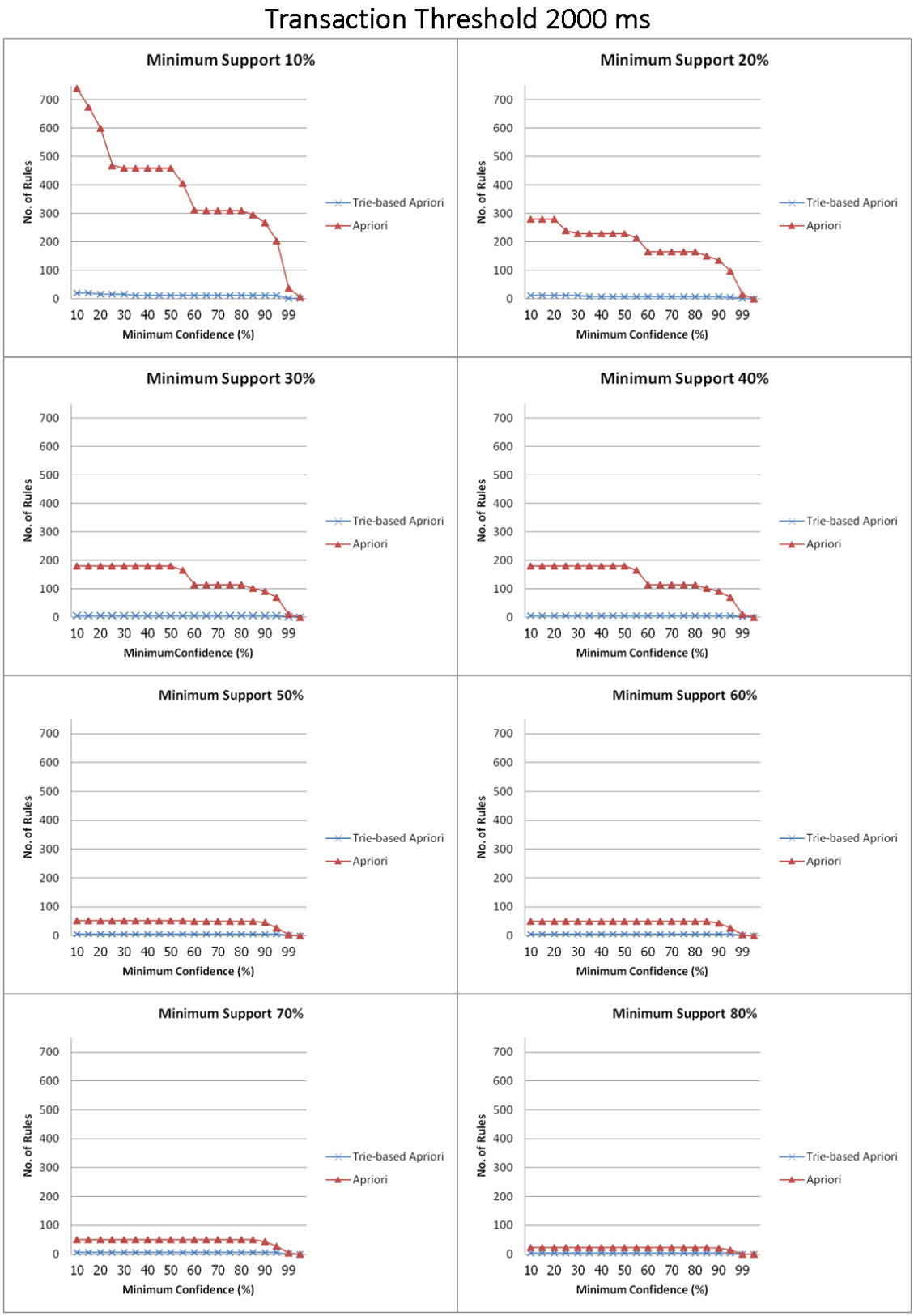
การทดลอง และผลการทดลอง

4.1 การทดลอง

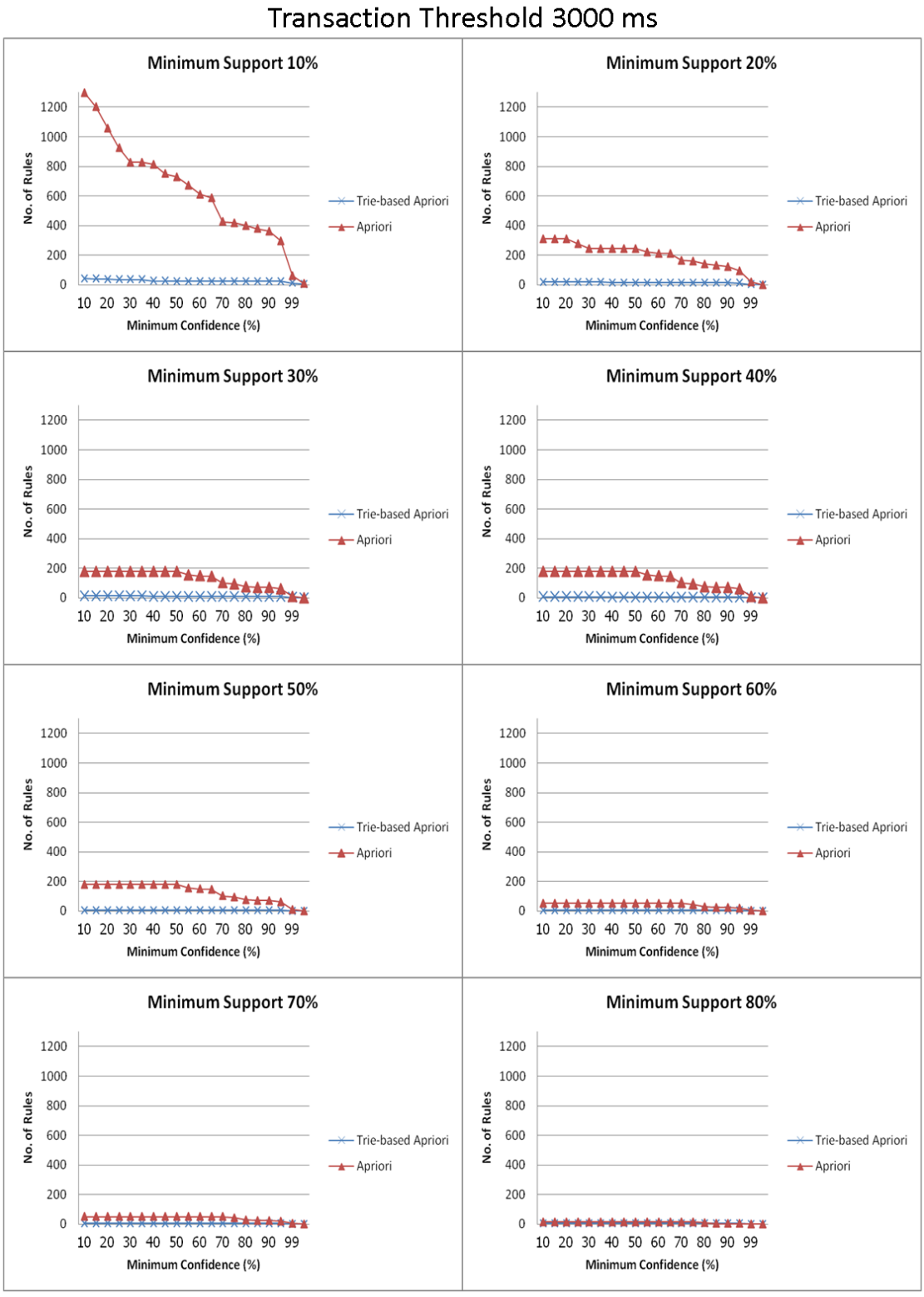
จากเว็บเซอร์วิซระบบซื้อสินค้าออนไลน์ที่สร้างขึ้นมา ผู้วิจัยได้ทำการติดตั้งเว็บเซอร์วิซไว้ที่ห้องปฏิบัติการ ISEL ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย และใช้กลุ่มผู้ทดลองใช้งานเป็นนักศึกษาปริญญาตรี ชั้นปีที่ 4 สาขาวิชาวิทยาการคอมพิวเตอร์ มหาวิทยาลัยราชภัฏสวนสุนันทา จำนวน 15 ราย ในช่วงระยะเวลา 1 เดือน โดยให้ผู้ใช้งานต่าง ๆ ทำการส่งคำร้องขอมายังเว็บเซอร์วิซ เพื่อทำการซื้อสินค้าออนไลน์ตามความต้องการ โดยไม่มีการกำหนดรูปแบบการเรียกใช้งานของแต่ละผู้ใช้งาน ทั้งนี้เพื่อให้ได้รูปแบบการเรียกใช้งานจริงบนเว็บเซอร์วิซ โดยทุกการเรียกใช้งานจะผ่านตัวกรองซึ่งดักจับการเรียกใช้งานที่ส่งไปยังเซอร์วิซ และทำการบันทึกรายละเอียดการเรียกใช้งานต่าง ๆ ลงในฐานข้อมูล จำนวนข้อมูลการเรียกใช้งานทั้งหมดที่นำมาทำการวิเคราะห์ในการทดลองมีจำนวน 18,018 ระเบียน

4.2 ผลการทดลอง

ผู้วิจัยได้ทำการทดลองเปรียบเทียบการวิเคราะห์ขนาดความสามารถของเซอร์วิซด้วยการทำเหมืองข้อมูลโดยใช้วิธีเอไพรออรี กับวิธีเอไพรออรีเชิงทรัย โดยใช้ข้อมูลบันทึกการเรียกใช้งานเดียวกัน และได้ทำการทดลองเป็น 2 กลุ่มด้วยกันตามค่าขีดแบ่งทรานแซกชันเท่ากับ 2 วินาที และ 3 วินาที ด้วยค่าสนับสนุนและค่าความเชื่อมั่นที่แตกต่างกัน เพื่อเปรียบเทียบจำนวนกฎความสัมพันธ์ที่ได้จากวิธีการทั้งสองวิธี ซึ่งสามารถแสดงผลการทดลองได้ดังภาพที่ 4.1 และภาพที่ 4.2 ตามลำดับ



ภาพที่ 4.1 เปรียบเทียบจำนวนกฎความสัมพันธ์ โดยใช้วิธีเอไพรออรี กับวิธีเอไพรออรีเชิงทรัย เมื่อค่าขีดแบ่งทรานแซกชันเป็น 2 วินาที



ภาพที่ 4.2 เปรียบเทียบจำนวนกฎความสัมพันธ์ โดยใช้วิธีเอไพรออรี กับวิธีเอไพรออรีเชิงทรัย เมื่อค่าขีดแบ่งทรานแซกชันเป็น 3 วินาที

4.3 อภิปรายผลการทดลอง

จากผลการทดลองพบว่า การวิเคราะห์ข้อมูลด้วยวิธีเอไพรออรีเชิงทรัย ได้จำนวนกฎความสัมพันธ์น้อยกว่าวิธีเอไพรออรี โดยเฉพาะเมื่อกำหนดค่าสนับสนุนขั้นต่ำที่มีค่าน้อย ความต่างของจำนวนกฎความสัมพันธ์ที่ได้จะมาก และที่ระดับค่าสนับสนุนเดียวกัน ความต่างของจำนวนกฎความสัมพันธ์มีแนวโน้มลดลงเมื่อระดับความเชื่อมั่นเพิ่มสูงขึ้น นอกจากนี้ยังพบว่าค่าขีดแบ่งทรานแซกชันยังส่งผลกระทบต่อขนาดทรานแซกชัน และจำนวนกฎความสัมพันธ์ โดยที่ข้อมูลทรานแซกชันที่ได้จากค่าขีดแบ่งทรานแซกชันเป็น 3 วินาที จะมีขนาดทรานแซกชันยาวกว่า (คือ จำนวน โอเปอเรชันในทรานแซกชันมากกว่า) ทรานแซกชันที่ได้เมื่อค่าขีดแบ่งทรานแซกชันเป็น 2 วินาที และส่งผลให้กฎความสัมพันธ์ที่ได้เมื่อค่าขีดแบ่งทรานแซกชันเป็น 3 วินาที จะมีจำนวนมากกว่าเมื่อใช้ค่าขีดแบ่งทรานแซกชันเป็น 2 วินาที จะเห็นได้ว่าขนาดของทรานแซกชันและจำนวนกฎความสัมพันธ์แปรผันตามกับค่าขีดแบ่งทรานแซกชันที่ใช้ในการวิเคราะห์

ดังนั้นแล้ว ผู้ออกแบบเซอร์วิซควรเลือกพิจารณาค่าขีดแบ่งทรานแซกชันที่มีความเหมาะสม เพื่อให้ได้ทรานแซกชันที่มีความใกล้เคียงกับการใช้งานจริงมากที่สุด การเลือกใช้ค่าสนับสนุนขั้นต่ำในการพิจารณาควรใช้ค่าสนับสนุนที่ไม่ต่ำจนเกินไป เนื่องจากหากเลือกใช้ค่าสนับสนุนขั้นต่ำที่มีระดับต่ำมากเกินไป ลำดับแนะนำอาจมาจากกลุ่มผู้ใช้งานจำนวนไม่มากนัก ซึ่งอาจไม่คุ้มค่ากับการพิจารณารวมกลุ่มโอเปอเรชันเข้าด้วยกัน ในขณะที่การเลือกใช้ค่าสนับสนุนขั้นต่ำที่สูงเกินไปอาจจะทำให้ไม่พบลำดับที่แนะนำเลย เนื่องจากมีปริมาณการเรียกใช้งานน้อยกว่าค่าสนับสนุนขั้นต่ำที่กำหนด ซึ่งไม่เกิดประโยชน์ในการวิเคราะห์ นอกจากนี้การกำหนดค่าความเชื่อมั่นควรเลือกใช้ค่าความเชื่อมั่นที่มีค่าสูง เพราะจะช่วยยืนยันว่าลำดับที่ได้จากการวิเคราะห์นั้นเป็นลำดับที่เกิดขึ้นจริง และมีความถูกต้องอยู่ในระดับสูง

กฎความสัมพันธ์ที่น่าสนใจที่ได้จากวิธีเอไพรออรีเชิงทรัยในตารางที่ 4.1 และจากวิธีเอไพรออรีของ [**4**] ดังตารางที่ 4.2 ได้จากการกำหนดค่าสนับสนุนขั้นต่ำร้อยละ 40 และค่าความเชื่อมั่นขั้นต่ำร้อยละ 90 โดยการทดลองปรับค่าที่เหมาะสมในการทดลอง

ตารางที่ 4.1 กฎความสัมพันธ์ที่น่าสนใจจากวิธีเอไพรออรีเชิงทรัย

ลำดับ	กฎความสัมพันธ์	ค่าความเชื่อมั่น
1	{cartAdd → calculatePrice → cartGet} → {itemLookup}	0.992790195
2	{cartAdd} → {calculatePrice}	0.980913429
3	{calculatePrice → cartGet} → {itemLookup}	0.980392157
4	{cartGet} → {itemLookup}	0.966491459
5	{cartAdd → calculatePrice} → {cartGet}	0.963863794
6	{calculatePrice} → {cartGet}	0.959677419
7	{cartAdd → calculatePrice}→ {cartGet → itemLookup}	0.956914524
8	{cartAdd} → {calculatePrice → cartGet}	0.945466939
9	{calculatePrice} → {cartGet → itemLookup}	0.940860215

ตารางที่ 4.2 กฎความสัมพันธ์ที่น่าสนใจจากวิธีเอไพรออรี

ลำดับ	กฎความสัมพันธ์	ค่าความเชื่อมั่น
1	{calculatePrice} → {cartGet}	96.3038
2	{cartGet} → {calculatePrice}	94.1524
3	{itemLookup} → {cartGet}	92.6019
4	{cartGet} → {itemLookup}	97.0434
5	{cartAdd} → {cartGet}	95.6374
6	{cartGet} → {cartAdd}	92.1813
7	{browseCatalog} → {itemLookup}	90.6404
8	{cartAdd} → {itemLookup}	97.6142
9	{itemLookup} → {calculatePrice}	90.2821
10	{calculatePrice} → {itemLookup}	96.7742
11	{calculatePrice} → {cartAdd}	96.8414
12	{cartAdd} → {calculatePrice}	98.2277
13	{calculatePrice , browseCatalog} → {itemLookup}	98.3607
14	{itemLookup , browseCatalog} → {calculatePrice}	91.3043
15	{itemLookup , cartGet} → {cartAdd}	94.3128

ตารางที่ 4.3 กฎความสัมพันธ์ที่น่าสนใจจากวิธีเอไพรออรี(ต่อ)

ลำดับ	กฎความสัมพันธ์	ค่าความเชื่อมั่น
16	{cartAdd} → {itemLookup , cartGet}	94.9557
17	{cartAdd , cartGet} → {itemLookup}	99.2872
18	{itemLookup , cartAdd } → {cartGet}	97.2765
19	{cartGet } → { itemLookup , cartAdd}	91.5243
20	{cartAdd , browseCatalog } → {itemLookup}	98.5816
21	{itemLookup , browseCatalog } → {cartAdd}	90.6522
22	{cartGet , browseCatalog } → {cartAdd}	95.8968
23	{cartAdd , browseCatalog } → {cartGet}	96.6903
24	{calculatePrice , cartGet } → {cartAdd}	97.2087
25	{calculatePrice} → {cartAdd , cartGet}	93.6156
26	{cartAdd} → {calculatePrice , cartGet}	94.9557
27	{cartAdd , cartGet} → {calculatePrice}	99.2872
28	{cartGet} → {calculatePrice , cartAdd}	91.5243
29	{calculatePrice , cartAdd} → {cartGet}	96.669
⋮	⋮	⋮
68	{itemLookup , cartAdd , cartGet , browseCatalog} → {calculatePrice}	99.6324
69	{itemLookup , calculatePrice , browseCatalog}→{cartAdd , cartGet}	96.7857
70	{itemLookup , calculatePrice , cartAdd , browseCatalog}→{cartGet}	98.3071
71	{calculatePrice , cartGet , browseCatalog}→{itemLookup , cartAdd}	97.7163

จากผลการเปรียบเทียบผลลัพธ์ข้างต้น พบว่ากฎความสัมพันธ์ที่น่าสนใจและสามารถรวมกลุ่มโอเปอเรชันกันได้ ซึ่งได้จากวิธีเอไพรออรีเชิงทรัยมีทั้งหมด 9 กฎ โดยทุกกฎจะปรากฏอยู่ในกฎที่แนะนำให้รวมกันที่ได้จากวิธีเอไพรออรีทั้งหมด 71 กฎ และพบว่าลำดับของโอเปอเรชันในกฎที่แนะนำให้รวมกันโดยวิธีเอไพรออรีบางกฎจะเป็นลำดับที่ไม่มีการเรียกใช้งานจริง ทั้งนี้เนื่องมาจากวิธีเอไพรออรีเชิงทรัยทำการตรวจสอบลำดับภายในกฎความสัมพันธ์ตั้งแต่ขั้นตอนการสร้างลำดับที่พบบ่อย ในขณะที่วิธีเอไพรออรีใช้วิธีการเรียงสับเปลี่ยนของไอเท็มเซ็ตที่พบบ่อยทั้งหมด ทำให้ได้ลำดับที่ไม่สอดคล้องกับการเรียกใช้งานจริง ซึ่งอาจสร้างความสับสนให้กับผู้ออกแบบเซอร์วิซในการพิจารณาเลือกการรวมกลุ่มโอเปอเรชันได้

นอกจากนี้ ผู้วิจัยยังได้ปรับปรุงการแนะนำการรวมโอเปอเรชันโดยมีการจัดอันดับโอเปอเรชันที่แนะนำให้รวมกลุ่มกันดังแสดงในตารางที่ 4.4 เพื่อช่วยให้ผู้ออกแบบทำการรวมกลุ่มโอเปอเรชันที่มีขนาดความสามารถที่เหมาะสม รองรับการใช้งานของผู้ใช้งานส่วนใหญ่ โดยพิจารณาลำดับย่อยของโอเปอเรชันที่พบร่วมกันในกลุ่มของกฎความสัมพันธ์ที่น่าสนใจจากตารางที่ 4.1 เรียงจากมากไปน้อย

ตารางที่ 4.4 การจัดอันดับลำดับที่แนะนำให้รวมกัน

อันดับ	ลำดับที่แนะนำให้รวมกัน	จำนวนครั้งที่พบ
1	calculatePrice → cartGet	8
2	cartAdd → calculatePrice	6
3	cartGet → itemLookup	6
4	cartAdd → calculatePrice → cartGet	6
5	calculatePrice → cartGet → itemLookup	5
6	cartAdd → calculatePrice → cartGet → itemLookup	3

จากลำดับที่แนะนำให้รวมกันดังแสดงในตารางที่ 4.4 หากผู้ออกแบบเซอร์วิซทำการรวมโอเปอเรชัน calculatePrice และ cartGet เข้าเป็นโอเปอเรชันเดียวกัน เพียงลำดับเดียวแล้ว จะส่งผลต่อผู้ใช้งานที่ทำการเรียกใช้งานโอเปอเรชัน calculatePrice และ cartGet ต่อเนื่องกัน โดยที่ผู้ใช้งานเหล่านี้สามารถลดจำนวนครั้งในการส่งการร้องขอมายังเซอร์วิซ จากที่ต้องทำการส่งการร้องขอมายังเซอร์วิซ 2 ครั้ง จึงจะครอบคลุมการทำงานที่ต้องการ เหลือเพียงแค่ครั้งเดียวก็ครอบคลุมการทำงานที่ต้องการแล้ว นอกจากนี้ยังส่งผลดีต่อผู้ที่เรียกใช้งานลำดับ cartAdd → calculatePrice → cartGet และผู้เรียกใช้งานลำดับ calculatePrice → cartGet → itemLookup เช่นเดียวกัน ซึ่งสามารถลดจำนวนครั้งการเรียกใช้งานลงจาก 3 ครั้ง เหลือเพียง 2 ครั้ง สำหรับผู้ที่เรียกใช้งานลำดับ cartAdd → calculatePrice → cartGet → itemLookup ก็ได้รับประโยชน์ในทำนองเดียวกัน นั่นคือสามารถลดจำนวนครั้งการเรียกใช้งานลงได้ 1 ครั้ง เหลือเพียง 3 ครั้ง

ในขณะเดียวกันหากผู้ออกแบบเซอร์วิซเลือกที่จะทำการรวมลำดับ cartAdd → calculatePrice → cartGet เพียงลำดับเดียว จะส่งผลดีต่อผู้ที่เรียกใช้งานลำดับ cartAdd → calculatePrice → cartGet และ cartAdd → calculatePrice → cartGet → itemLookup เท่านั้น แต่จะไม่เป็นประโยชน์ต่อผู้ใช้งานที่เรียกใช้งานลำดับ cartAdd → calculatePrice หรือ calculatePrice →

cartGet เนื่องจากโอเปอเรชันที่ทำการรวมกันแล้วมีขนาดความสามารถที่เกินกว่าความต้องการของผู้เรียกใช้งาน

ดังนั้นแล้ว หากผู้ออกแบบเซอร์วิซทำการรวมกลุ่มโอเปอเรชันทุกลำดับที่ทำการแนะนำจะส่งผลดีต่อผู้เรียกใช้งานมากกว่าการเลือกรวมเฉพาะบางลำดับที่แนะนำให้รวมกัน แต่ทั้งนี้ต้องพิจารณาด้วยว่าการรวมกลุ่มโอเปอเรชันจะเป็นการสร้างเวอร์ชันใหม่ของการให้บริการ จึงย่อมมีประเด็นเรื่องการจัดการและการบำรุงรักษาบริการต่างเวอร์ชันพร้อมกันด้วย

บทที่ 5

บทสรุปงานวิจัย

5.1 สรุปผลการวิจัย

งานวิจัยนี้ได้ปรับปรุงการวิเคราะห์ขนาดความสามารถของเซอร์วิซด้วยการทำเหมืองข้อมูลโดยการประยุกต์ใช้เอไพรออรีเชิงทรัย และดับเบิลอาร์เรย์ทรัยในการวิเคราะห์ขนาดความสามารถของเซอร์วิซ เพื่อวิเคราะห์ว่าเซอร์วิซมีขนาดความสามารถเล็กเกินไปหรือไม่ และแนะนำกลุ่มของเซอร์วิซโอเปอเรชันที่มีการเรียกใช้งานต่อเนื่องกันบ่อย โดยทำการทดสอบกับเซอร์วิซซื้อสินค้าออนไลน์ จากการทดลองพบว่าสามารถแนะนำโอเปอเรชันที่มีขนาดความสามารถใหญ่ขึ้นจากการรวมกลุ่มโอเปอเรชันที่มีการเรียกใช้บ่อยได้ ซึ่งกลุ่มโอเปอเรชันที่แนะนำให้รวมกันที่ได้จากอัลกอริทึมเอไพรออรีเชิงทรัย และดับเบิลอาร์เรย์ทรัยสามารถแสดงลำดับของโอเปอเรชันที่แนะนำให้รวมกันได้ดีกว่าและมีจำนวนลำดับที่แนะนำน้อยกว่าวิธีเอไพรออรีทั่วไป และมีการจัดลำดับการแนะนำซึ่งช่วยให้ผู้ออกแบบเซอร์วิซทำการตัดสินใจรวมกลุ่มเซอร์วิซโอเปอเรชันที่มีการเรียกใช้บ่อยได้ดียิ่งขึ้น

ลำดับที่แนะนำให้รวมกันนี้ไม่ส่งผลกระทบต่อประสิทธิภาพการทำงาน เนื่องจากการพิจารณาการเรียกใช้งานเพื่อรวมเป็นทรานแซกชัน ไม่ได้ทำการพิจารณาการเรียกใช้งานในลักษณะขนาน แต่จะพิจารณาการเรียกใช้งานแบบลำดับเพียงอย่างเดียว นอกจากนี้ การรวมกลุ่มโอเปอเรชันนี้ยังช่วยลดปริมาณการส่งข้อมูลในระยะไกล จากการส่งการร้องขอและการตอบกลับการร้องขอที่น้อยลง ซึ่งสามารถส่งผลให้เซอร์วิซมีสภาพพร้อมใช้งานเพิ่มมากขึ้น

อย่างไรก็ตาม ลำดับของโอเปอเรชันนี้ที่แนะนำให้รวมกันนี้ไม่ได้เป็นการแนะนำให้รวมโอเปอเรชันดังกล่าวแทนที่โอเปอเรชันเดิม แต่เป็นการแนะนำให้รวมกันเพื่อสร้างเป็นโอเปอเรชันใหม่ที่มีขนาดความสามารถใหญ่ขึ้นสำหรับเซอร์วิซเดิม หรือสำหรับสร้างเว็บเซอร์วิซเวอร์ชันใหม่เพื่อเป็นทางเลือกให้กับผู้ใช้งานที่ต้องการลดจำนวนครั้งในการเรียกใช้งานลง โดยที่ไม่ส่งผลกระทบต่อผู้ใช้งานที่เรียกใช้แบบเดิมอยู่

5.2 อุปสรรคและข้อจำกัดในงานวิจัย

ในงานวิจัยนี้มีข้อจำกัดในด้านการประมวลผลทรานแซกชันที่พิจารณาการเรียกใช้งานในลักษณะลำดับเท่านั้น ไม่ครอบคลุมลักษณะการเรียกใช้งานแบบอื่น ๆ เช่น แบบขนาน หรือแบบวนซ้ำ เป็นต้น ซึ่งอาจมีการเรียกใช้งานบ่อย และทำให้ผลการแนะนำการรวมโอเปอเรชันผิดพลาดได้

5.3 แนวทางในการพัฒนาต่อ

5.3.1 ทำการทดลองรวมโอเปอเรชันตามคำแนะนำจากการวิเคราะห์ และวัดเวลาที่เรียกใช้งานว่ามีเวลาเฉลี่ยลดลงหรือไม่

5.3.2 ปรับปรุงการแนะนำการรวมโอเปอเรชันให้คำนึงถึงความเข้ากันได้ของพารามิเตอร์ของโอเปอเรชันที่จะรวมกัน

5.3.3 จัดการเวอร์ชันใหม่ของเซอร์วิซที่เกิดขึ้นเนื่องจากการรวมเป็นโอเปอเรชันใหม่

รายการอ้างอิง

[1] Thomas Erl. SOA:Principles of Service Design.: PRENTICE HALL, 2007.

[2] Ferenc Bodon. "A Trie-based APRIORI Implementation for Mining Frequent Item Sequences." in *OSDM'05*, pp. 56-65 Chicago,Illinois,USA 2005.

[3] Jun-Ichi Aoe, Katsushi Morimoto, and Takashi Sato. An Efficient Implementation of Trie Structures. Software-Practice and Experience 22(September 1992): 695-721.

[4] Twittie Senivongse, Nattawud Phacharintanakul, Choltida Ngamnitiporn, and Matee Tangtrongchit. "A Capability Granularity Analysis on Web Service Invocations." in *World Congress on Engineering and Computer Science 2010*, pp. 400-405 2010.

[5] Thomas Erl et al. Web Service Contract Design and Versioning for SOA.: PRENTICE HALL, 2007.

[6] Rakesh Agrawal, Tomasz Imielinski, and Arun Swami. Mining Association Rules between Sets of Items in Large Databases. ACM SIGMOD 22(June 1993).

[7] Rakesh Agrawal and Srikant Ramakrishnan. "Fast algorithms for mining association rules." in *The International Conference on Very Large Databases*, pp. 487-499 1994.

[8] Edward Fredkin. Trie Memory. Communication of the ACM 3(September 1960).

[9] Dan Gusfield. Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology.: Cambridge University Press, 1997.

[10] Raf Haesen, Monique Snoeck, Wilfried Lemahieu, and Stephan Poelmans. "On the Definition of Service Granularity and Its Architectural Impact." in *Proceedings of 20th International Conference on Advanced Information Systems Engineering*, pp. 375-389 Montpellier, France Springer-Verlag Berlin, 2008.

[11] Jinlei Jiang, Yongwei Wu, and Guangwen Yang. "Making Service Granularity Right: An Assistant Approach Based on Business Process Analysis." in *2011 Sixth Annual ChanaGrid Conference*, pp. 204-210 Liaoning 2011.

[12] Tristan Glatard, Johan Montagant, David Emsellem, and Diane Lingrand. A Service-Oriented Architecture enabling dynamic service grouping for optimizing distributed

workflow execution. Future Generation Computer Systems 24(July 2008): 720-730.

[13] Ran Tang and Ying Zou. "An Approach for Mining Web Service Composition Patterns from Execution Logs." in *12th IEEE Symposium on Web Systems Evolution (WSE)*, pp. 53-62 Timisoara, Romania 2010.

ประวัติผู้เขียนวิทยานิพนธ์

นายบวร เรืองแรงสกุล เกิดเมื่อวันที่ 28 มกราคม 2525 สำเร็จการศึกษาระดับประกาศนียบัตรวิชาชีพจากสถาบันเทคโนโลยีราชมงคล วิทยาเขตพระนครเหนือ และสำเร็จการศึกษาระดับปริญญาบัณฑิต สาขาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์และเทคโนโลยี มหาวิทยาลัยราชภัฏสวนสุนันทา เมื่อปีการศึกษา 2549 ปัจจุบันรับราชการตำแหน่งนักวิชาการคอมพิวเตอร์ปฏิบัติการ สังกัดกรมศุลกากร กระทรวงการคลัง