รายงานการวิจัย

ส่วนประกอบในการค้นหากฎความเกี่ยวข้องสำหรับข้อมูลในรูปแบบ XML

Association rule extraction component for XML data

คณะผู้วิจัย

หัวหน้าโครงการ

นางสาวอารยา ไพธิสรณ์

คณะวิศวกรรมศาสตร์

มหาวิทยาลัยอุบลราชธานี

ผู้ร่วมวิจัย

นายวิชิต สมบัติ

นายธนกร ลิ้มสุวรรณ

โครงการวิจัยนี้ได้รับทุนสนับสนุนการวิจัยจากมหาวิทยาลัยอุบลราชธานี

หมวดเงินอุดหนุนทั่วไป ประจำปีงบประมาณ พ.ศ. 2546

A Research Report

Association rule extraction component for XML data

Researchers

Head of Project

Araya Pothisorn

Faculty of Engineering

Ubon Rajathanee University

Co-researchers

Wichit Sombat

Thanakorn Limsuwan

This Research was Financially Supported from Ubon Rajathanee University

In Finance Year, 2002

รายงานการวิจัยเรื่อง	ส่วนประกอบในการค้นหากฎความเกี่ยวข้องสำหรับข้อมูลในรูปแบบ XML
หัวหน้าโครงการวิจัย	นางสาวอารยา โพธิสรณ์
ผู้ร่วมโครงการวิจัย	นายวิชิต สมบัติ
	นายธนกร ลิ้มสุวรรณ
คณะวิศวกรรมศาสตร์	มหาวิทยาลัยอุบลราชธานี
ปีงบประมาณ	2546
งบประมาณที่ได้รับ	58,200.- บาท
คำสำคัญ	กฎความเกี่ยวข้อง, เหมืองข้อมูล, จาวา

บทคัดย่อ

ข้อมูลเป็นสิ่งสำคัญในการตัดสินใจ อย่างไรก็ดีถ้าข้อมูลมีมากแต่ไม่มีเครื่องมือช่วยวิเคราะห์ ข้อมูลนั้นก็ใช้ประโยชน์ได้น้อยหรือไม่มีประโยชน์เลย กฎความสัมพันธ์เป็นเครื่องมือที่ใช้ในการค้นหาความสัมพันธ์ของข้อมูลเพื่อช่วยให้เราสามารถตัดสินใจได้อย่างมีเหตุผล

กฎความสัมพันธ์เป็นกระบวนการที่ใช้กันอย่างแพร่หลายในวงการธุรกิจ การเงิน การค้า การจัดการข้อมูลของแม่ข่าย การวิจัยทางสังคมและชีววิทยา และยังนำไปใช้ทางด้านต่างๆอีกมาก เช่น ช่วยวิเคราะห์หาความสัมพันธ์ของสินค้าที่ลูกค้าซื้อจากร้านขายของชำ เพื่อช่วยในการสั่งสินค้าเพิ่ม หรือกลยุทธ์ในการลดราคา

XML เป็นรูปแบบเอกสารมาตรฐานที่มีหน่วยงานสนับสนุนมากมาย ระบบฐานข้อมูลในปัจจุบันก็มีการพัฒนาให้สามารถรองรับรูปแบบเอกสาร XML ดังนั้นระบบที่สามารถดึงข้อมูลในรูปแบบดังกล่าวจึงจะสามารถนำมาใช้ประโยชน์ได้อย่างสูง

ระบบที่พัฒนาขึ้นในการวิจัยครั้งนี้ใช้เทคโนโลยีของจาวาที่เป็นที่ยอมรับกันอย่างแพร่หลาย โดยระบบจะใช้กระบวนวิธี Apriori ที่พัฒนาขึ้นโดยกลุ่มผู้พัฒนาของบริษัท IBMในการค้นหากฎความสัมพันธ์จากข้อมูลนำเข้าในรูปแบบ XML แล้วเขียนเป็นข้อมูลในรูปแบบของ PMML ซึ่งเป็นรูปแบบที่เหมาะสำหรับการสืบค้นกฎความสัมพันธ์ของข้อมูล เนื่องจาก PMML เป็นรูปแบบที่พัฒนาขึ้นโดยกลุ่ม Data Mining Group มีมีบริษัทใหญ่อย่าง Microsoft, IBM, SPSS, Oracle Corporations เป็นสมาชิกหลัก

ระบบที่พัฒนามีความสมบูรณ์ กล่าวคือสามารถเขียนผลลัพธ์ที่ได้จากการค้นหาความสัมพันธ์ของแฟ้มข้อมูลในรูปแบบ XML ให้เป็นข้อมูลในรูปแบบ PMML ทั้งนี้ระบบที่พัฒนายังสามารถพัฒนาต่อให้สามารถใช้งานผ่านเครือข่ายได้ ผู้ใช้สามารถกำหนดจำนวนข้อมูลสนับสนุนและระดับความน่าเชื่อถือให้กับระบบได้ ในกรณีที่ผู้ใช้ไม่ระบุระบบจะใช้ระดับความน่าเชื่อถือเป็นร้อยละ 30 และจำนวนข้อมูลสนับสนุนเป็นร้อยละ 20

Association rule extraction component for XML data

Head of Project	Ms. Araya Pothisorn
Co-researchers	Mr. Wichit Sombat
	Mr. Thanakorn Limsuwan
Faculty of Engineering,	Ubon Rajathanee University

In Finance Year	2002 for 58,200.- Bath
Keyword	Association rule, PMML, Java-XML, Data Mining, Apriori

Abstract

Data is important assisting in making decision as well as a proper operation. However, having too much data without tools for refining the knowledge from the data is less useful or even useless. As one of such tools, association rules are among the most popular representations for local patterns in data mining. It can summarize the association of the information in order to clarify the characteristics of the data which we then can use to assist in making decision.

Association rules are used widespread in many aspects such as business, finance, marketing, server administration, social and biological research. Also, the applications are still extending to other fields. For example, the grocery, the result may be something similar to that most of the customers who buy beer often buy snacks. From this knowledge, the grocery can decide to stock more snacks when beer price is reduced.

Nowadays, because of its flexibility, XML is the standard for exchange the data and it is well supported by many large organizations. Also, many commercial database products are heading to support XML, so the tool for extracting the knowledge from data in form of XML will be substantially useful.

This project uses the widely accepted Java technology to develop a system for extracting association rules using A Priori algorithm developed by the IBM research team from XML-format input to product output file of PMML, a more suitable format for data mining process. PMML is specifications of the Data Mining Group:DMG which has IBM Corp. Microsoft, SPSS Inc., Oracle Corporations and fifteen software company as full members.

The project successfully implements the system. Output files in PMML format are produced corresponding to XML input format. The system runs in command-line mode, but could easily be ported to standalone GUI application or even an on-line version. User can specify minimum support and confidential level through configuration files. The system uses the default values for minimum support and confidential level of 20% and 30% respectively.

กิตติกรรมประกาศ

โครงงานวิจัยครั้งนี้ลุล่วงไปได้ด้วยดีเพราะบุคคลากรที่เกี่ยวข้องในองค์กรหลายฝ่าย ซึ่งโครงการนี้จะเกิดขึ้นไม่ได้เลยถ้าไม่มีผู้เสนอโครงการอาจารย์ธัชพงศ์ กตัญญุกุล และหน่วยงานที่สนุนงานวิจัยของมหาวิทยาลัยที่คอยให้คำแนะนำเรื่องการจำทำโครงการ และอธิบายกฎระเบียบต่างๆ โดยสุดท้ายนี้ขอขอบคุณผู้ร่วมวิจัยทุกท่านที่มอบเวลาอันมีค่าในการประชุมและปฏิบัติงานวิจัยจนลุล่วงได้ด้วยดี และการวิจัยครั้งนี้ได้รับทุนอุดหนุนการวิจัยจากมหาวิทยาลัยอุบลราชธานี ประจำปีงบประมาณ พ.ศ. 2546

คณะผู้วิจัย

มิถุนายน 2548

สารบัญ

สารบัญตาราง

สารบัญรูป

สารบัญขั้นตอนวิธี

สารบัญรายการ

บทที่ ๑
บทนำ

๑.๑ ความสำคัญ และที่มาของงานวิจัย

ข้อมูลเป็นส่วนสำคัญที่ช่วยในการตัดสินใจในการวางแผนรวมถือการปฏิบัติการที่เหมาะสม แต่การมีข้อมูลจำนวนมากโดยปราศจากเครื่องมือช่วยในการกลั่นกรองส่วนที่เป็นความรู้ออกมานั้น ข้อมูลนั้นก็ไม่สามารถนำมาใช้ประโยชน์ได้อย่างเต็มที่ หรือ อาจไม่สามารถนำมาใช้ประโยชน์ได้เลย กฎความเกี่ยวข้องของข้อมูล (Association rules) เป็นเครื่องมือตัวหนึ่งซึ่งการช่วยในการสรุปความสัมพันธ์ของข้อมูลออกมาเพื่อให้เกิดความรู้ความเข้าใจในข้อมูลนั้น ๆมากขึ้นซึ่งจะนำไปสู่การใช้ประโยชน์ต่อไป

กฎความเกี่ยวข้องนั้นปัจจุบันมีการนำไปใช้อย่างกว้างขวางทั้งทางธุรกิจ การเงิน การตลาด การบริหาร คอมพิวเตอร์เซิร์ฟเวอร์ การวิจัยทางด้านสังคม และ ชีวะวิทยา และยังมีความพยายามจะขยายการใช้งานออกไปอีกอย่างต่อเนื่อง ตัวอย่างผลจากการหากฎความเกี่ยวข้องเช่น ในร้านขายของ อาจให้ผลว่า ลูกค้าส่วนใหญ่ที่ซื้อเบียร์แล้วมักจะซื้อขนมขบเคี้ยวด้วย จากข้อมูลนี้ทำให้บริษัทรู้ว่าถ้าเบียร์ลดราคาก็ควรจะเตรียมขนมขบเคี้ยวไว้เพิ่มขึ้น หรือ อาจเป็นข้อมูลว่าหน้าหนาวคนจะซื้ออะไรมากขึ้น หรือลูกค้าผู้หญิงที่ซื้อโทรศัพท์มือถือภายในสองเดือนมีแนวโน้มว่าจะซื้ออะไร หรือ ร้านวีดีโออาจต้องการรู้ว่าลูกค้าคนไหนจะเปลี่ยนไปใช้บริการร้านอื่น และลูกค้ากลุ่มนี้มีลักษณะอย่างไร ร้านวีดีโอสามารถใช้ข้อมูลที่ได้ในการออกโปรโมชั่นเพื่อตึงลูกค้ากลุ่มนี้ไว้

เนื่องจาก ปัจจุบัน XML ก็เป็นภาษามาตราฐานที่ได้รับความนิยมในการเก็บและแลกเปลี่ยนข้อมูล เนื่องจากมีความยืดหยุ่นสูง และได้รับการสนับสนุนเป็นอย่างดีจากทั้งองค์กร และบริษัททางด้านเทคโนโลยีขนาดใหญ่ และการใช้ XML เป็นข้อมูลกลางกำลังขยายตัวออกไปอย่างรวดเร็ว มีกลุ่มทำงานที่นำ XML ไปใช้เป็นมาตราฐานมากมายไม่ว่าจะเป็น WML ของกลุ่มอุปกรณ์สื่อสาร, MathML และ CML ของกลุ่มทางด้านวิทยาศาสตร์ FpML และ FIXML ของกลุ่มทางการเงิน และยังมีอีกหลาย ๆ มาตรฐานที่พัฒนาจาก XML รวมถึงอีกหลาย ๆกลุ่มที่กำลังพยายามสร้างภาษากลางขึ้นมาจาก XML นอกจากนั้น ผลิตภัณฑ์ทางด้านระบบฐานข้อมูลของหลาย ๆบริษัทก็สนับสนุนการทำงานกับ XML ดังนั้นการสร้างเครื่องมือเพื่อค้นหากฎความเกี่ยวข้องจากข้อมูลในรูปแบบภาษา XML สามารถช่วยในสามารถเข้าใจจนถึงใช้ประโยชน์จากข้อมูลที่อยู่ในรูปแบบมาตราฐานนี้ได้มากขึ้น

นอกจากนี้ แนวทางที่ได้จากการศึกษาครั้งนี้ จะสามารถนำไปช่วยในการพัฒนาการทำเหมืองข้อมูลกับข้อมูล XML ในแง่มุมอื่น ๆได้อีกด้วย

๑.๒ วัตถุประสงค์ของการวิจัย

๑.๒.๑ เพื่อศึกษาแนวทางในการประยุกต์ทฤษฎีของการหากฎความเกี่ยวข้อง เพื่อใช้กับ ข้อมูล XML

๑.๒.๒ เพื่อสร้างส่วนประกอบต้นแบบที่ใช้ในการหากฎความเกี่ยวข้อง เพื่อใช้กับข้อมูล XML

๑.๓ ประโยชน์ที่คาดว่าจะได้รับ

๑.๓.๑ แนวทางในการประยุกต์ทฤษฎีของการหากฎความเกี่ยวข้อง เพื่อใช้กับ ข้อมูล XML

๑.๓.๒ ส่วนประกอบต้นแบบที่ใช้ในการหากฎความเกี่ยวข้อง เพื่อใช้กับข้อมูล XML

๑.๓.๓ สามารถใช้ประโยชน์จากข้อมูลในรูปแบบ XML ซึ่งกำลังเพิ่มการใช้งานอย่างรวดเร็ว ได้อย่างมีประสิทธิภาพมากขึ้น

๑.๓.๔ เป็นเครื่องมือให้นักศึกษาได้ทดลองฝึกการทำเหมืองข้อมูล

๑.๔ ขอบเขตของการวิจัยและแผนการดำเนินการ

โครงการนี้แบ่งออกเป็น 4 ระยะ โดยระยะแรกจะเป็นการศึกษาข้อมูลที่เกี่ยวข้องอัลกอริทึมในการหากฎความเกี่ยวข้องต่าง ๆ เทคโนโลยี XML และ API รวมถึง ผลของ XML ที่ได้จากระบบฐานข้อมูล และภาษา PMML ระยะที่สองจะเป็นการวิเคราะห์และออกแบบในการทำงานหลัก ๆรวมถึงการโปรแกรมและทดสอบ จากนั้นในระยะที่สามวิเคราะห์การทำงานจากระยะที่สอง แก้ไขในส่วนที่จำเป็นและเพิ่มหน้าที่รอง ๆของส่วนประกอบและทำการทดสอบ ระยะที่ 4 เป็นการแก้ไขทำความสะอาดโปรแกรมและจัดทำเอกสารฉบับสมบูรณ์ โดยมีแผนงานวิจัยย่อยดังนี้

๑. การศึกษาการใช้งานข้อมูลกลาง(Meta data) ของ XML เพื่อให้เครื่องสามารถเข้าใจข้อมูลของ XML ได้ รวมถึงภาษา PMML ที่เป็นมาตราฐานในการแลกเปลี่ยนข้อมูลจากการทำเหมืองข้อมูล เพื่อให้ส่วนประกอบซอฟแวร์ที่ได้มีความสามารถทำงานร่วมกับซอฟแวร์อื่น ๆได้

๒. การศึกษาถึงความเหมาะสมของอัลกอริทึ่มต่าง ๆของการหากฎความเกี่ยวข้อง ที่เหมาะสมกับลักษณะข้อมูลที่ยืดหยุ่นของ XML

๑.๕ วิธีดำเนินการวิจัย

วิธีดำเนินการวิจัยแบบย่อยเป็นขั้นตอนดังต่อไปนี้

๑. การศึกษารูปแบบ การใช้งาน และการทำงานของข้อมูลแบบ XML (Extended Mark Up language)

๒. การศึกษาอัลกอริทึ่มที่เกี่ยวข้องกับการทำเหมืองข้อมูล ได้แก่ Apriori Algorithm

๓. ทำการเขียนโปรแกรมเพื่อทดสอบการอ่านข้อมูล XML และจัดรูปแบบให้อยู่ในโครงสร้างข้อมูลที่ต้องการ

๔. ศึกษาข้อมูลเกี่ยวกับ PMML

๕. ทำการเขียนโปรแกรมส่วนของ Apriori Algorithm

๖. เก็บรายละเอียดของโปรแกรมให้ได้ผลลัพท์เป็นเอกสาร PMML

๗. ทดสอบการทำงานของโปรแกรมและแก้ไขข้อบกพร่อง

บทที่ ๒
วรรณกรรมหรืองานวิจัยที่เกี่ยวข้อง

การศึกษาเกี่ยวกับการรวมกันของการทำเหมืองข้อมูลและข้อมูล XML มีอยู่อย่างกว้างขวางถึงแม้ว่าทั้งสองจะเป็นสาขาที่ค่อนข้างใหม่ เนื่องจากแนวโน้มของข้อมูลบนเครื่องข่ายอินเตอร์เนตรวมถึงข้อมูลที่จะถูกแลกเปลี่ยนกันระหว่างองค์กร หรือระหว่างระบบฐานข้อมูลที่ต่างกัน หรือ ระหว่างแอปพลิเคชั่นต่าง ๆ ว่าจะเป็นเอกสาร XML และ ขนาดของข้อมูลก็คาดว่าจะเพิ่มขึ้นอย่างมากมมาย ดังนั้นการทำเหมืองข้อมูลบนเอกสาร XML จึงได้รับความสนใจ มิเช่นนั้นข้อมูลจำนวนมากมายที่ได้มานั้นอาจใช้ประโยชน์ได้น้อยมากหรืออาจไม่สามารถใช้ประโยชน์ได้เลย

เครื่องยนต์ค้นหา (Search Engine) เป็นเครื่องมือที่ใช้ค้นหาข้อมูลจากเอกสารในอินเตอร์เนต แต่ผลจากการทำงานจะเป็นแค่เอกสารที่เกี่ยวข้องกับคำหลักของการค้นหา (Key words) เท่านั้นไม่ได้มีการสรุปความเกี่ยวข้องสัมพันธ์กันของข้อมูล ไม่เหมือนกับ การทำเหมืองข้อมูลที่นำเสนอ

Cooley และคณะ(1999) ได้ศึกษาถึงวิธีการเตรียมข้อมูลสำหรับข้อมูลในเครือข่าย WWW ซึ่งเป็นขั้นตอนที่สำคัญก่อนที่จะทำเหมืองข้อมูล Buechner และคณะ(2000) ได้ศึกษาถึงการรวมกันของสองเทคโนโลยีนี้ในแง่มุมต่าง ๆเอาไว้แต่ก็ยังไม่ได้มีการสร้างส่วนประกอบที่ทำงานนี้ขึ้น กลุ่มการทำเหมืองข้อมูล (Data Mining Group) ได้เสนอ ภาษา PMML (Predictive Model Markup Language) ซึ่งอยู่บนพื้นฐานของ XML เพื่อเป็นภาษาที่ใช้อธิบายข้อมูล วิธีการทำเหมืองข้อมูล และ ผลจากเหมืองข้อมูล ซึ่งสามารถทำให้ระบบเหมืองข้อมูลที่ต่างกันสามารถใช้ผลจากระบบอื่น ๆได้ ซึ่งส่วนประกอบที่เสนอในโครงการนี้ก็จะถูกออกแบบให้สามารถทำงานกับภาษานี้ได้เป็นอย่างดี และงานของ Ferguson (1999) ใกล้เคียงกับโครงงานนี้มากเพียงแต่โครงงานนี้เน้นที่การสร้างส่วนประกอบของซอปแวร์ในการหากฎความเกี่ยวข้อง ในขณะที่งานของ Ferguson เน้นไปที่การเก็บข้อมูลออกมาจากเอกสาร XML

บทที่ ๓
วิธีการดำเนินการวิจัย

๓.๑ วิธีที่ใช้ในการศึกษาค้นคว้า

ดำเนินการค้นข้อมูลผ่านทางเครือข่ายอินเตอร์เน็ต เข้าร่วมอบรมเกี่ยวกับจาวาและ XML รวมถึงใช้งบประมาณในการซื้อหนังสือประกอบการวิจัย

๓.๒ เครื่องมือที่ใช้ในการวิจัย

๓.๒.๑ XML (Extensible Markup Language)

เมื่อปี พ.ศ. ๒๕๔๑ การแลกเปลี่ยนข้อมูลระหว่างระบบแบบกระจาย(Distributed System) ยังมีข้อจำกัดอยู่หลายอย่าง รูปแบบของข้อมูลที่แลกเปลี่ยนกันในยุคนั้นเป็นลักษณะของ Hypertext Markup Language (HTML) ซึ่งสามารถแสดงผลได้บนโปรแกรมประยุกต์ที่เรียกว่า Web Browser ซึ่งเป็นการเปิดรูปแบบของการแลกเปลี่ยนข้อมูลบนระบบเครือข่ายคอมพิวเตอร์ขนาดใหญ่หรือที่รู้จักกันดีในชื่ออินเตอร์เน็ต HTML เป็นรูปแบบของการแลกเปลี่ยนข้อมูลบนอินเตอร์เน็ตแบบใหม่ที่สามารถเปิดโอกาสให้ผู้ใช้งานสามารถตอบโต้กับเนื้อหาที่ถูกแสดงผลบนเบราส์เซอร์ แต่ HTML มีข้อจำกัดเกี่ยวกับความไม่ยืดหยุ่นในการแสดงผลข้อมูลที่แลกเปลี่ยนกันในองค์กรแบบอุตสาหกิจ หรือที่เรียกว่า Enterprise ในช่วงเวลาต่อมาจึงมีการนำเสนอ Extensible Markup Language เพื่อรองรับการใช้งานเกี่ยวกับการแลกเปลี่ยนข้อมูลในองค์กรขนาดใหญ่ ทั้งในลักษณะของการค้าแบบอิเล็กทรอนิกส์ ที่ต้องรองรับการจัดการกับฐานข้อมูลขนาดใหญ่ และรองรับการใช้งานพร้อมกันจากผู้ใช้หลายการเชื่อมต่อในขณะเวลาหนึ่งๆ XML อนุญาตให้องค์กรสามารถกำหนดรูปแบบของลักษณะข้อมูลที่ต้องการใช้ในการแลกเปลี่ยนระหว่างองค์กร หรือใช้เฉพาะทางในองค์กรที่มีลักษณะใกล้เคียงหรือคล้ายกัน อันได้แก่ การค้าอิเล็กทรอนิกส์ การรวมตัวของห่วงโซ่อุปทาน การจัดการข้อมูล และการทำสื่อโฆษณา

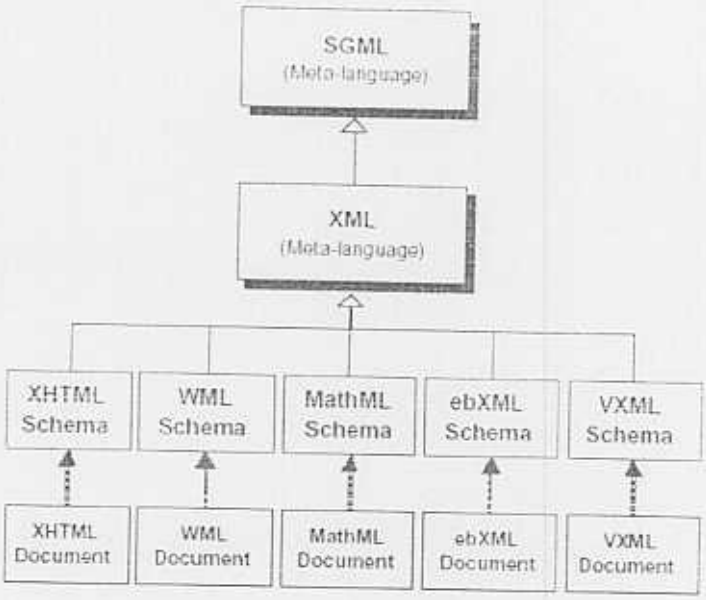
อันเนื่องมาจากสาเหตุดังกล่าว XML กลายเป็นเครื่องมือทางยุทธศาสตร์ในการกำหนดรูปแบบของข้อมูลที่ใช้งานระหว่างขอบเขตงาน คุณสมบัติของ XML เป็นสิ่งที่เหมาะสมที่ XML จะถูกนำมาใช้งานในการแสดงข้อมูล รูปแบบเชิงความคิดและคำอธิบายข้อมูลในระบบเปิดหรือระบบที่ไม่ขึ้นกับแพลตฟอร์ม (platform) XML ใช้ tag เป็นตัวกำหนดจุดเริ่มต้นและจำกัดจุดสิ้นสุดของท่อนข้อมูล เพื่อสร้างลำดับชั้นขององค์ประกอบข้อมูลที่เกี่ยวข้องกันที่เรียกว่า element ในทางกลับกันโครงสร้างลำดับชั้นของ element ก็แสดงถึงคำอธิบาย ที่มีความหมายทางนัยยะที่บอกตำแหน่งของ element โดยวิธีการนี้เรียกว่าการห่อหุ้มหรือ Encapsulation อันมีผลต่อเนื่องให้งานประยุกต์อื่นใดสามารถนำข้อมูลกลับมาใช้ได้ใหม่ในลักษณะที่ยังคงอยู่เดิมโดยไม่ต้องทำการแก้ไขรูปแบบเพื่อให้ขึ้นกับแพลตฟอร์ม

เทคโนโลยี XML ประสบความสำเร็จในการเป็นสิ่งสำคัญที่ถูกใช้งานในเชิงการแก้ปัญหาสำหรับการแลกเปลี่ยนข้อมูลเชิงวิกฤต การทำพลับลิชชิ่ง และการพัฒนาโปรแกรมประยุกต์ต่าง ๆ ยิ่งไปกว่านั้น XML กลายเป็นเครื่องกระตุ้นสำหรับกลุ่มธุรกิจอุตสาหกรรมที่มีลักษณะประกอบการคล้ายกัน หรือกลุ่มอุตสาหกรรมคู่ค้าในการพัฒนาและกำหนดรูปแบบของข้อมูลที่จะทำการแลกเปลี่ยนกันในเชิงอิเล็กทรอนิกส์ ซึ่งเป็นผลให้เกิดภาษาสำหรับใช้ในการแลกเปลี่ยนข้อมูลและแสดงผลระหว่างองค์กรลักษณะเดียวกันได้แก่ ebXML เป็นภาษาที่ใช้สำหรับธุรกิจอิเล็กทรอนิกส์พาณิชย์ หรือ PMML เป็นภาษาที่ใช้สำหรับงานทางเหมืองข้อมูล หรือ WML (Wireless Markup Language) เป็นภาษาที่ใช้สำหรับงานที่ประยุกต์ใช้กับข้อมูลที่ใช้อุปกรณ์ไร้สาย ซึ่งเป็นลักษณะของการกำหนดรูปแบบสารสนเทศที่ไม่ใช่เฉพาะเกณฑ์เดียวลักษณะหนึ่งต่อหนึ่ง แต่เป็นการกำหนดรูปแบบสารสนเทศที่มีไว้ใช้สำหรับการแบ่งปันสารสนเทศดังกล่าวข้ามองค์กรที่เป็นอาณาจักรหรือวงขนาดกว้าง

บริษัท Sun Microsystems พร้อมด้วยบริษัทคู่ค้าผู้นำทางเทคโนโลยีหลัก ๆได้แก่ IBM, Novell, Oracle และแม้แต่ Microsoft ก็ได้ร่วมกันพัฒนาเทคโนโลยีที่สอดคล้องรองรับการทำงานของ XML ทั้งสิ้น บริษัท Sun Microsystems ร่วมกับกลุ่มทำงานองค์กร W3C (World Wide Web Consortium) ได้ร่วมกันทำการกำหนดคุณลักษณะเฉพาะของ XML อีกด้วย ทั้งนี้ Sun Microsystems ยังมีเทคโนโลยี Java ที่มีคุณสมบัติสอดคล้องกับ XML ทั้งในแง่ของการแลกเปลี่ยนข้อมูลต่างแพลตฟอร์ม และการทำงานในเชิงกระจาย จึงทำให้ เทคโนโลยี Java เป็นเครื่องมือสำคัญสำหรับการพัฒนางานที่เกี่ยวข้องกับ XML และเป็นผลให้การพัฒนางานประยุกต์ในองค์กรส่วนใหญ่ที่ใช้ เทคโนโลยี XML มีการใช้งานเทคโนโลยี Java เป็นภาษาที่ใช้ในการเขียนงานประยุกต์เหล่านั้น

๓.๒.๑.๑ เทคโนโลยี XML และการใช้งาน

XML มีรากฐานมาจากภาษาอุปมาหรือ Meta-language ที่เรียกว่า SGML (Standard Generalized Markup Language) ตัว XML เองไม่ใช่ภาษาโดยตัวเอง แต่เป็น Meta-language ที่ใช้ในการสร้างภาษาอื่น ๆ XML ถูกใช้ในการสร้างโครงสร้างของภาษา เอกสารที่อธิบายตัวของมันเองอันประกอบด้วยกฎที่เกี่ยวเนื่องโดยเฉพาะกับภาษานั้น ๆ เช่น Mathematical Markup Language เป็นภาษาที่ใช้อธิบายงานทางคณิตศาสตร์ หรือ Voice Markup Language ที่เป็นภาษาที่ใช้กับงานประยุกต์ที่เกี่ยวกับการแลกเปลี่ยนข้อมูลที่เป็นเสียงโดยเฉพาะ ลำดับชั้นของเอกสาร XML แสดงในรูปที่ ๓.๒.๑



รูป ๓.๒.๑ ลำดับชั้นของ XML

XML เองประกอบด้วยส่วนที่เป็นสัญลักษณ์หรือ Markup และส่วนที่เป็นข้อมูลหรือ Content ทั้งนี้ Markup บางครั้งถูกเรียกว่า tag ด้วยคุณสมบัติอันยืดหยุ่นนี้ทำให้ข้อมูลสามารถถูกทำการรับ ส่ง หรือแปลงข้อมูลจากรูปแบบหนึ่งไปเป็นรูปแบบอื่นได้โดยง่าย เช่นการใช้งาน XML ในทางการค้าอิเล็กทรอนิกส์นั้นมีทั้งการใช้งานเกี่ยวกับเรื่องการกำหนดราคา การทำงานเรื่องสินค้าคงคลัง และรวมถึงการจัดการข้อมูลทางการค้าข้อมูลต่าง ๆเหล่านี้ถูกทดแทนในรูปแบบของ XML โดยทั้งสิ้น หลังจากนั้นก็สามารถถูกนำไปแลกเปลี่ยนหรือถ่ายโอนผ่านทางระบบเครือข่ายอินเตอร์เน็ต โดยใช้มาตรฐานระบบเปิดและโปรโตคอลต่าง ๆ XML แต่ละอันจะมีไวยากรณ์เป็นของตัวเอง และมีกฎเฉพาะเจาะจงในเรื่องส่วนข้อมูลและโครงสร้างของเอกสารที่ถูกเขียนโดยภาษานั้น ๆ เช่นส่วนของ pricing ใน ebXML จะไม่มีความหมายใน MathML แต่อย่างใด ดังนั้นแต่ละภาษาสามารถทำการแต่งเติมหรือกำหนดกฎไวยากรณ์สำหรับภาษาของตนเองได้โดยที่ XML ได้จัดหาสิ่งอำนวยความสะดวกสำหรับการสร้างเอกสารที่ถูกหลักไวยากรณ์สำหรับภาษาใด ๆก็ตามที่มีรากฐานมาจาก XML ทั้งนี้ตัวกระจายวลี (XML parser) สามารถตรวจสอบโครงสร้างใด ๆของเอกสาร XML จากกฎที่ได้มีการกำหนดไว้สำหรับภาษานั้น ๆ

XML ยังสามารถถูกใช้งานในแง่ของการเป็นพื้นฐานสามัญสำหรับภาษาชั้นสูงที่สนับสนุนการแลกเปลี่ยนข้อมูลระหว่างองค์ประกอบในงานประยุกต์ หรือ ในระบบ หรือแม้แต่ระหว่างองค์กรก็ดี เครื่องมือสำหรับการกระจายวลีและการแปลวลีถูกเขียนขึ้นมาเพื่อให้สามารถจัดการได้กับทุกรูปแบบของข้อมูลแบบ XML ที่ถูกสร้างขึ้นโดยกฎของแต่ละภาษาที่มีกำหนดไว้ เช่น XML parser สามารถใช้อ่านเอกสาร MathML และก็สามารถใช้กระจายวลีเอกสาร ebXML ได้เช่นกัน และตัวแปลง XMLตัวเดียวกันก็สามารถใช้แปลงเอกสารการสั่งซื้อที่เป็น ebXML ให้เป็นเอกสารที่อยู่ในรูปแบบของ RosettaNet PIP ได้เช่นกัน

XML เป็นเทคโนโลยีในอุดมคติสำหรับอินเตอร์เน็ต เนื่องจากข้อมูล XML ซึ่งมีลักษณะเดียวกันกับ HTML นั้นถูกเข้ารหัสด้วย tag ต่าง ๆที่ถูกสร้างขึ้นมาจากอักขระข้อความธรรมดาหรือที่เรียกว่า plain text ทั้งนี้ระบบปฏิบัติการส่วนใหญ่สามารถจัดการกับ plian text ได้เป็นอย่างดีอยู่แล้ว จึงทำให้เป็นการสะดวกที่จะสร้างตัวจัดการล่วงหน้าหรือ preprocessor และ browsers ที่สามารถรองรับข้อมูล XML ได้เป็นอย่างดี และเนื่องจาก XML ประกอบไปด้วย element และ attribute ที่ถูกกำหนดไว้ใน DTD ที่มีอยู่ในเครื่องนั้น ๆ เมื่อใดก็ตามที่เอกสาร XML ถูกอ้างถึง ข้อมูลเหล่านั้นก็จะเป็นที่รู้จักว่าหมายถึงอะไรตราบเท่าที่เอกสารนั้นอยู่ในรูปแบบมาตรฐานของ XML

XML แก้ปัญหาการทำความเข้าใจกับข้อมูลให้กับเครื่อง ซึ่ง HTML ทำได้เพียงให้ผู้ใช้เข้าใจข้อมูล เช่น ใน HTML เพื่อแสดงข้อมูลว่าลูกค้าชื่อ John Smith อายุ 35 ปีสามารถเขียนได้ดังนี้

```
<P>Customer = <B>John Smith</B></P>
<P>Age = <B>35</B></P>
```

ซึ่งจะแสดงข้อความ

Customer=John Smith
Age=35

บนเบราส์เซอร์ อันเป็นที่เข้าใจได้เป็นอย่างดีสำหรับผู้ใช้ แต่สำหรับเครื่องแล้วเป็นการปฏิบัติต่ออักขระเพื่อการแสดงผลเท่านั้น หากเป็น XML อาจสามารถเขียนเพื่อให้เครื่องเข้าใจว่า ข้อมูลนั้นหมายถึงอะไรเมื่อทำการอ่าน ก็อาจจะเอยู่ในลักษณะดังนี้

```
<customer>
<name>John Smith</name>
<age>35</age>
</customer>
```

<age> และ </age> เป็น element ลูกที่ซ่อนอยู่ใน <customer> ดังนั้นเมื่อเครื่องทำการอ่านเอกสาร XML ก็จะสามารถแยกแยะได้ว่า 35 นั้นหมายถึงอายุ(ซึ่งถูกกำหนดไว้ใน <age>) และ John Smith คือชื่อของลูกค้า (ซึ่งถูกกำหนดไว้ใน <name>)

๓.๒.๑.๒ องค์ประกอบของเอกสาร XML

๑. XML Entity และ XML Data

เอกสาร XML ประกอบด้วย Entity และ Data โดยที่ตัวเอกสาร XML นั้นถูกประกอบขึ้นมาจาก Entity หลาย ๆอัน Entity นั้นสามารถถูกแยกแยะได้โดยชื่อเรียกหรือ Identifier ซึ่ง Entity นั้นอาจประกอบไปด้วยข้อมูลของมันเอง หรือการประกาศเฉพาะตัวของ Entity นั้น ๆ หรืออาจเป็นการอ้างถึงข้อมูลผ่านทาง URL ก็เป็นได้ ข้อมูลหรือ Data เป็นส่วนหนึ่งของเอกสาร XML ซึ่งประกอบด้วยวลีที่ถูกกระจายแล้ว (parsed entity) และวลีที่ยังไม่ถูกกระจาย (unparsed entity) Unparsed Entity ประกอบไปด้วยข้อมูลที่อาจจะเป็น text หรือไม่ใช่ text ก็เป็นได้ ถ้าเป็น text อาจจะเป็น text ที่อนุญาตไว้ใน parsed entity หรือไม่ก็ได้ โดยส่วนมากแล้ว unparsed Entity จะถูกใช้งานกับพวกข้อมูลที่เป็น binary ได้แก่พวกรูปภาพ ดังนั้น unparsed Entity จะต้องถูกกำหนดไว้ด้วยเครื่องหมายหรือ Notations เสมอ โดยทำการกำหนดไว้อย่างชัดเจนใน DTD เช่นบอกชนิดของภาพเช่น image/jpg เป็นต้น

Parsed Entity คือองค์ประกอบที่เป็นเนื้อหาหรือเป็น tag ที่ถูกจัดให้อยู่ในรูปแบบที่ถูกต้องตามกฎของ XML โดย parse Entities เหล่านั้นจะถูกอ้างอิงถึงในเอกสาร DTD ด้วยชื่อในลักษณะของ XML ที่เหมาะสม

๒. XML Vocabularies

XML Vocabulary คือชุดของ Elements และ Attributes ที่เกี่ยวกับเฉพาะเรื่อง อาจะหมายถึงเฉพาะอุตสาหกรรม หรือเฉพาะกลุ่มก็ได้ หรืออาจหมายถึงเฉพาะแนวทางเฉพาะเรื่อง

๓.๒.๑.๓ ส่วนต่าง ๆของเอกสาร XML

เอกสาร XML เป็นเอกสารที่มีการจัดรูปแบบและสัดส่วนเป็นอย่างดี สามารถแบ่งเป็น 3 ส่วนคือ ส่วนprolog ส่วน body และ ส่วนepilog ซึ่ง prologและ epilog นั้นจะมีหรือไม่มีก็ได้ แต่ส่วน body เป็นสิ่งที่จะต้องมี

เพื่อสามารถอธิบายถึงความสามารถและความยืดหยุ่นของ XML และเทคโนโลยีอื่น ๆที่เกี่ยวข้อง จึงแสดงตัวอย่างของเอกสาร XML ไว้ดังรายการที่ ๓.๒.๑

```
<?xml version="1.0"?>
<!DOCTYPE product-catalog SYSTEM "file:/product-catalog.dtd">
<product-catalog>
        <product sku="123456" name="The Product">
          <description locale="en_US">
               An excellent product.
          </description>
          <description locale="es_MX">
               Un producto excellente.
          </description>
          <price locale="en_US" unit="USD">
               99.95
          </price>
          <price locale="es_MX" unit="MXP">
               9999.95
          </price>
        </product>
</product-catalog>
```

รายการ ๓.๒.๑ ตัวอย่างเอกสาร XML

๑. Prolog

ส่วนของ prolog นั้นแม้กล่าวว่าจะมีหรือไม่มีก็ได้ แต่ในการใช้งานจริงควรจะมีการกำหนดไว้ เนื่องจากส่วนของ prolog เป็นส่วนที่บอกข้อมูลที่สำคัญหลายอย่างที่เกี่ยวกับเอกสาร XML นั้น ๆอยู่ เช่น อธิบายถึงการประกาศเอกสาร XML ได้แก่

```
<?xml version="1.0"?>
```

การประกาศเอกสาร XML จะต้องอยู่บรรทัดแรกสุดของส่วน prolog ของเอกสารเสมอไม่มีข้อยกเว้นแม้แต่พวกช่องว่าง ย่อหน้า หรือหมายเหตุก็ตาม ส่วนของการประกาศนี้อาจประกอบด้วยข้อมูลการเข้ารหัส (encoding) และชนิดของการใช้งาน (standalone) ด้วยก็ได้ encoding เป็นการบอกลักษณะการเข้ารหัสของอักขระที่ใช้งานในเอกสาร ส่วน standalone เป็นการกำหนดว่าเอกสารนี้ต้องมีการประกาศ entity ทุก ๆ entity ไว้ด้วยหรือไม่ ถ้าไม่จำเป็นแล้วต้องใช้เอกสาร DTD จากภายนอก จากตัวอย่างข้างต้น prolog แรกใช้อธิบายเวอร์ชั่นของ XML ที่ใช้งานอยู่ รูปแบบคือ

```
<?xml version="x.x" ?>
```

โดยมี `<?xml` เป็นtagเปิด และมี `?>` เป็น tag ปิด ทั้งนี้ tag เปิดและปิดต้องสัมพันธ์กัน ถ้ามีส่วนของ encoding และ standalone ก็จะมีลักษณะดังนี้

```
<?xml version="1.0" encoding="UTF-16" standalone="yes"?>
```

prolog ส่วนต่อมาในเอกสารตัวอย่างคือการประกาศ DOCTYPE เป็นการประกาศว่าเอกสาร DTD ที่ต้องนำมาใช้งานในการแปลและตรวจสอบเอกสาร XML นี้นั้นอยู่ที่ไหน ซึ่งสามารถทำได้ 2 ลักษณะ คือเป็นแบบภายใน (Internal)และภายนอก(External) ถ้าเป็นแบบภายในก็จะเป็นการนำ DTD มาเขียนรวมไว้ในเอกสาร XML นั้น ๆด้วย แต่ถ้าเป็นแบบภายนอกก็สามารถทำได้ดังเอกสาร XML ตัวอย่าง เมื่อ SYSTEM หมายถึง DTD นั้นอยู่ในเครื่องนั้น ๆ และ PUBLIC หมายถึง DTD อยู่ที่ใดที่หนึ่งในอินเตอร์เน็ต เป็นต้น จากเอกสารตัวอย่าง มีการประ DOCTYPE ดังนี้

```
<!DOCTYPE product-catalog SYSTEM "file:/product-catalog.dtd">
```

เป็นการอธิบายว่าไฟล์ DTD (Document Type Definition) ได้ถูกกำหนดไว้ที่ใด ในที่นี้กำหนดไว้ในไดเรคทอรี่เดียวกับที่ product-catalog.xml อาศัยอยู่ การกำหนดดังกล่าวมีรูปแบบทั่วไปคือ

```
<!DOCTYPE xml_root_document_name SYSTEM
"file:/where_the_dtd_file_resided">
```

เมื่อ *xml_root_document_name* หมายถึงชื่อของรากของเอกสารXML และ *where_the_dtd_file_resided* หมายถึงตำแหน่งของไฟล์ DTD ในระบบ

สำหรับภาษา HTML มีการกำหนด DOCTYPE แบบ Frameset DTD ไว้ดังนี้

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
"http://www.w3.org/TR/xhtml1/DTD/frameset.dtd">
```

ถ้าเป็นแบบ Strict หรือแบบ Transition ก็ทำการเปลี่ยนแปลงดังนี้

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/strict.dtd">
```

สำหรับ DOCTYPE แบบ Strict และ

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transition//EN"
"http://www.w3.org/TR/xhtml1/DTD/transition.dtd">
```

สำหรับ DOCTYPE แบบ Transition

การกำหนดดังกล่าวเป็นการบอกว่าไฟล์ DTD นั้นให้ทำการอ้างอิงไปที่ URL (Universal Resource Locator) ดังที่กำหนดไว้ และใช้คำสำคัญ PUBLIC เนื่องจาก DTD นี้เป็นมาตรฐานที่ใช้กันอยู่ซึ่งถูกกำหนดโดยองค์กร W3C

ไฟล์ DTD นั้นมีสาวนสำคัญในการที่ตัวกระจายวลีหรือ parser สามารถตรวจสอบไวยากรณ์และความหมายของวลีและคำ ในเอกสาร XML

๒. **Body**

ส่วนต่อมาของเอกสาร XML คือส่วนของ Body อันได้แก่ XML elements, attributes, entities และการประกาศ DTD ส่วนอื่น ๆอีก จากเอกสารตัวอย่าง<product-catalog> ... </product-catalog> เรียกว่า Root Element ของเอกสาร XML และ <product> ...</product> เป็น child element ของ <product-catalog> ซึ่งใน <product> ก็มี child element คือ <price> และ <description> ส่วน sku และ name ใน <product> นั้นเป็น attributes ของ product ทั้งนี้ attribute เองไม่สามารถมี attribute ของมันเองได้อีก ส่วน locale ก็เป็น attribute ของ description ซึ่งมีเนื้อหาของ description แรกคือ
An excellent product.

๓. **Epilog**

คือส่วนสุดท้ายของเอกสาร XML ในส่วนนี้จะประกอบไปด้วยหมายเหตุของเอกสาร XML คำสั่งในการปฏิบัติการ (Processing Instructions) หรือพวกช่องว่าง ทั้งนี้ส่วน Epilog นั้นรวมไว้หรือไม่ก็ได้ เนื่องจาก สิ่งเหล่านี้จะต้องถูกประมวลผลในส่วนอื่น ๆอยู่แล้ว อย่างไรก็ตาม ตราบใดที่มาตรฐานหรือเวอร์ชั่นใหม่ยังไม่มีการกำหนดข้อกำหนดอื่นใดสำหรับ Epilog ก็ยังไม่มีความจำเป็นที่ต้องเพิ่มไว้ในเอกสาร XML ที่พัฒนาอยู่

๓.๒.๑.๔ Document Type Definitions (DTD)

DTDs เป็นส่วนสำคัญของ XML เนื่องจาก DTD ประกอบไปด้วยกฎและระเบียบของแต่ละ element ที่จะเกิดขึ้นในเอกสาร XML และรวมไปถึง attribute ที่สามารถมีได้ใน element ซึ่งเป็นหนึ่งทางเลือกสำหรับการใช้งาน ที่ไม่ต้องใช้ XML schema เนื่องจาก DTD ได้เกิดขึ้นก่อนและมีการใช้งานมาอย่างต่อเนื่อง แต่เนื่องจาก DTD นั้นซับซ้อนและยากแก่การทำความเข้าใจ ทำให้ปัจจุบันมีการคิดค้น XML schema ขึ้นมาและเป็นที่นิยมกว่า DTD อย่างไรก็ตาม DTD ยังเป็นสิ่งที่เหมาะสมกับการสร้างเอกสาร XML อยู่ ตัวอย่างของ DTD ที่ใช้งานกับเอกสาร XML: product-catalog มีดังแสดงในรายการที่ ๓.๒.๒

```
<!ELEMENT product-catalog (product+)>

<!ELEMENT product (description+, price+)>
<!ATTLIST product
        sku ID #REQUIRED
        name CDATA #REQUIRED
>

<!ELEMENT description (#PCDATA)>
<!ATTLIST description
        locale CDATA #REQUIRED
>

<!ELEMENT price (#PCDATA)>
```

```
<!ATTLIST price
   locale CDATA #REQUIRED
   unit CDATA #REQUIRED
>
```

รายการ ๓.๒.๒ ตัวอย่างเอกสาร DTD

องค์ประกอบของการเขียน DTD นั้นมีหลักการต่าง ๆดังนี้

`<!ENTITY % Text "CDATA">` เรียกว่า parameter entity definition คือเป็นการกำหนดว่า entity นี้สามารถมี parameter ใดได้บ้าง และสามารถมีได้แค่ 1 element หรือสามารถมีได้มากกว่า 1 หรือไม่ เช่น `<!ELEMENT product-catalog (product+)>`

ส่วนต่อมาคือ `<!ATTLIST` ในที่นี้เป็นการกำหนด attribute ของ entity นั้น ๆว่าเป็นอย่างไร มี attribute อะไรบ้าง และแต่ละ attribute จะรับค่าชนิดใด และเป็นสิ่งที่ต้องใส่ค่า หรือสามารถละเว้นการกำหนดค่าได้ เช่น

```
<!ATTLIST product
        sku ID #REQUIRED
        name CDATA #REQUIRED
>
```

หากทำการพัฒนางานที่ใช้ XML ที่เป็นมาตรฐานกันดีอยู่แล้วโดยปกติ DTD จะมีกำหนดไว้ที่องค์กร W3C ดังนั้นจึงเป็นสาเหตุให้ไม่มีความจำเป็นที่จะต้องเขียน DTD สำหรับภาษา HTML เมื่อพัฒนา HTML เนื่องจาก HTML ได้ถูกกำหนด DTD ที่ใช้กันอย่างแพร่หลายทั่วไป ส่วน DTD ของอุตสาหกรรมอื่น ๆ หรือเฉพาะทางอื่น ๆ ถ้าเป็นสายเฉพาะที่มีการใช้กันอย่างแพร่หลายและรับทราบโดย W3C แล้วก็จะมี DTD ไว้ให้ที่เวบไซต์ของ W3C (http://www.w3c.org)

๓.๒.๑.๕ XML schema

XML schema เป็นอีกวิธีการหนึ่งในการกำหนดรูปแบบของเอกสาร XML ที่จะถูกพัฒนาขึ้นมา XML schema สามารถถูกสร้างได้ง่ายกว่า DTD และตัว XML schema เองก็เป็นเอกสารแบบ XML ด้วยอีกทั้ง XML schema ยังมีความสามารถในการตรวจสอบชนิดข้อมูลได้ดีกว่า DTD อีกด้วย ตัวอย่างการเขียน XML schema มีดังแสดงในรายการที่ ๓.๒.๓

```
<xsd:schema
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<xsd:element type="product-catalog"/>
<xsd:complexType name="productCatalog">
<xsd:element type="productType"
minOccurs="1"/>
</xsd:complexType>
<xsd:complexType name="productType">
<xsd:element name="description"
type="xsd:string" minOccurs="1">
<xsd:attribute name="locale"
type="xsd:string"/>
</xsd:element>
```

```
<xsd:element name="price"
type="xsd:decimal" minOccurs="1">
<xsd:attribute name="locale"
type="xsd:string"/>
<xsd:attribute name="unit"
type="xsd:string"/>
</xsd:element>
<xsd:attribute name="sku"
type="xsd:decimal"/>
<xsd:attribute name="name"
type="xsd:string"/>
</xsd:complexType>
</xsd:schema>
```

รายการ ๓.๒.๓ ตัวอย่างเอกสาร XML schema

๓.๒.๒ Java Technology

Java Technology ถูกพัฒนาโดยบริษัท Sun Microsystems ตัวเทคโนโลยีเองมีหลายส่วนและหลายองค์ประกอบเฉพาะด้วยกัน รวมไปถึงมีการพัฒนา Middleware สำหรับการพัฒนางานเฉพาะด้านในแต่ละส่วนของ Java เกิดขึ้นอย่างต่อเนื่อง เทคโนโลยี Java อาจแบ่งตามขนาดของการพัฒนางานได้ 3 ส่วนหลัก ดังนี้

๑. Java Micro Edition
๒. Java Community Edition/Standard Edition
๓. Java Enterprise Edition

Java Micro Edition เป็นเทคโนโลยีที่อธิบายและใช้งานกับงานประยุกต์ขนาดเล็ก โดยเฉพาะการพัฒนางานประยุกต์สำหรับอุปกรณ์ขนาดเล็ก อันได้แก่โทรศัพท์เคลื่อนที่ คอมพิวเตอร์ขนาดพกพา (PDA) สมาร์ทคาร์ด (Smart Card) และอื่น ๆอีกมากมาย

Java Community Edition/Standard Edition เป็นเทคโนโลยีหลักของ Java ใช้สำหรับการพัฒนางานสำหรับระบบขนาดเล็ก หรือพัฒนางานสำหรับเครื่องที่เป็นลูกข่าย โดยใช้ภาษาเขียนโปรแกรมจาวา โดยมีเทคโนโลยีย่อย ๆเพื่อสนับสนุนการทำงานของ Java อีกมากมาย

Java Enterprise Edition

Java Enterprise Edition กำหนดมาตรฐานสำหรับการพัฒนางานบนระบบหลายระดับชั้น (Multitier) โดยอาศัยเทคโนโลยีจาก Community Edition ทั้งในเรื่องของ Database Access และการแลกเปลี่ยนองค์ประกอบโดยใช้เทคโนโลยี CORBA ซึ่งครอบคลุมถึงระบบความปลอดภัยในการแลกเปลี่ยนข้อมูลข้ามระบบเครือข่ายอินเตอร์เน็ต จึงเป็นเทคโนโลยีที่เหมาะสมอย่างยิ่งการพัฒนางานระหว่างสำหรับผู้ประกอบการ

๓.๒.๒.๑ Java Language

Java เป็นภาษาคอมพิวเตอร์ระดับสูง คือมีไวยากรณ์ใกล้เคียงกับภาษาอังกฤษที่ใช้งานกันอยู่ทั่วไป Java ถูกพัฒนามาจากภาษา C/C++ จึงมีส่วนคล้ายคลึงอย่างมากกับภาษาดังกล่าว Java มีข้อดีอยู่หลาย

อย่าง และมีสโลแกนของตัวภาษาว่า "Write once , run anywhere" ซึ่งสโลแกนดังกล่าว บอกความสามารถที่เด่นและชัดเจนที่สุดของ Java แล้ว

๑. Java เป็นภาษาที่เป็น platform independent คือ Java สามารถ run ได้บนทุกรูปแบบของเครื่องคอมพิวเตอร์ หรือแม้แต่อุปกรณ์สื่อสารที่มี Java Virtual Machine(JVM) อยู่ ดังนั้น Java ที่จะถูกนำไปใช้งานบนคอมพิวเตอร์รูปแบบต่าง ๆกันนั้นเป็น class file ไม่ใช่ .exe เพื่อรันบนระบบปฏิบัติการ Windows หรือเป็น .hqx เพื่อรันบนระบบปฏิบัติการ Mac OS Classic เมื่อเป็น class file และบนเครื่องคอมพิวเตอร์นั้น ๆ มี Java Virtual Machine อยู่ก็สามารถรันโปรแกรมประยุกต์ Java นั้น ๆได้

๒. Java เป็นการเขียนโปรแกรมเชิงวัตถุ (Object Oriented Programming: OOP) ซึ่งเป็นการเขียนโปรแกรมแบบใหม่ที่เกิดขึ้นในยุคหลัง โดยการเขียนโปรแกรมเชิงวัตถุดังกล่าวมีข้อดีที่สนับสนุนอยู่หลายอย่างเช่น การนำ code มาใช้งานได้ใหม่ (Reuse), การป้องกันข้อมูลจากการเข้าถึง code บางส่วน และอื่น ๆที่เป็นคุณสมบัติของ OOP ซึ่งจะกล่าวถึงในภายหลัง

๓. Java เป็นภาษาที่ง่ายต่อการทำความเข้าใจ (Simple) เนื่องจาก Java มีรากฐานมาจากภาษา C /C++ และเป็นภาษาชั้นสูงซึ่งมีไวยากรณ์ใกล้เคียงกับภาษาอังกฤษที่ใช้งานอันอยู่ปัจจุบัน จึงทำให้ Java เป็นภาษาที่ง่ายต่อการทำความเข้าใจและไม่ซับซ้อน

๔. Java มีแบบจำลองของระบบความปลอดภัยในตัวเองทั้งในเวอร์ชั่น 1.1 และ 1.4.0 ขึ้นมา จึงทำให้ตัวภาษา Java เองรองรับการใช้งานที่ต้องการความปลอดภัยอยู่แล้ว

๕. Java code มีขนาดเล็กเมื่อเทียบกับโปรแกรมประยุกต์ภาษาอื่น เนื่องจาก Java มี Utility classes ที่อยู่บน JVM อยู่แล้วทำให้ class file ที่จะนำไปรันบน JVM นั้น ๆไม่จำเป็นต้องแบกรับ Utility classes ที่ใช้งานไปด้วย

๖. Java สนับสนุนการทำงานแบบ Multithread ซึ่งเป็นหัวใจสำคัญสำหรับการทำงานแบบกระจายทำให้ Java เหมาะสมที่สุดกับการเขียนโปรแกรมแบบกระจาย

๗. Java เป็นการทำงานแบบ Interpreted ดังนั้นจึงสะดวกแก่การแก้ไขหาจุดผิดพลาด

๓.๒.๒.๒ XML Supported

เทคโนโลยี Java เป็นเทคโนโลยีที่เหมาะสมและใช้งานกันอย่างแพร่หลายในการสร้างงานประยุกต์บนระบบเครือข่าย รวมไปถึงอินเตอร์เน็ต ตัว Java เองมีลักษณะเป็น Portable กล่าวคือสามารถนำไปใช้งานบนแพลตฟอร์มอื่นได้โดยไม่ต้องทำการคอมไพล์ใหม่ ซึ่งเป็นคุณสมบัติที่สนับสนุนการใช้งานบนอินเตอร์เน็ตเป็นอย่างดี ปัจจุบัน Java Development Kit เวอร์ชั่นใหม่ ๆได้ทำการบรรจุ XML parserไว้ให้โดยทีผู้ใช้งานไม่จำเป็นต้องทำการติดตั้งเพิ่มในภายหลัง

Java API สำหรับ XML สามารถแบ่งได้เป็น 2 ลักษณะคือ

๑. เชิงเอกสาร (Document- oriented) ประกอบด้วย

๑.๑ JavaTM API for XML Processing (JAXP) — ใช้สำหรับจัดการกับเอกสาร XML โดยมี parser หลาย ๆลักษณะให้เลือกใช้งาน

๑.๒ JavaTM Architecture for XML Binding (JAXB)— จัดโครงร่างเอกสาร XML โดยทำการจับคู่แต่ละ element ของ XML ให้อยู่ในลักษณะของ class ในการเขียนโปรแกรมแบบ Java

๒. เชิงการจัดการ (Procedure-oriented) ประกอบด้วย

๒.๑ JavaTM API for XML Messaging (JAXM) — ใช้การส่ง วัตถุชนิด SOAP ข้ามเครือข่ายอินเตอร์เน็ตตามแบบมาตรฐาน

๒.๒ JavaTM API for XML Registries (JAXR)—ให้บริการเกี่ยวกับมาตรฐานในการแบ่งปันข้อมูลและข้อมูลลงทะเบียนเชิงการค้าผ่านเครือข่ายอินเตอร์เน็ต

๒.๓ JavaTM API for XML-based RPC (JAX-RPC) — ใช้ SOAP method ในการทำการเรียกใช้งาน method ในระยะไกล ผ่านระบบเครือข่ายอินเตอร์เน็ต

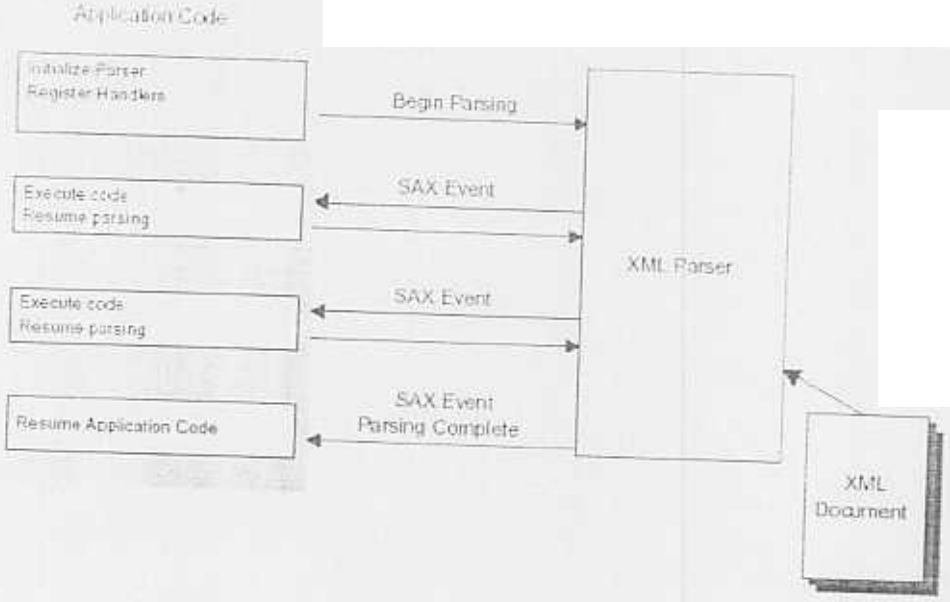
๓.๒.๒.๓ XML parsing Technologies

ก่อนที่เอกสาร XML จะถูกทำการตรวจสอบให้ถูกต้องและนำไปใช้งานได้นั้นจะต้องผ่านขั้นตอนของการกระจายวลีหรือการแยกองค์ประกอบข้อมูล (parsing) โดยใช้ parser ซึ่งเป็นโปรแกรมประยุกต์ที่ถูกพัฒนามาเพื่อใช้สำหรับ parse เอกสาร XML โดยเฉพาะ ปัจจุบันมีโปรแกรมประยุกต์เหล่านี้อยู่มากมาย รวมทั้ง Crimson และ Xerces ที่ถูกพัฒนาโดย The Apache Software Foundation (http://xml.apache.org) ซึ่งเป็น open source และถูกใช้งานอย่างแพร่หลายในอุตสาหกรรม parser มีหน้าที่ทำให้ข้อมูลในเอสาร XML ที่เป็นโครงสร้างอยู่ พร้อมสำหรับโปรแกรมประยุกต์อื่น ๆที่จะนำข้อมูลเหล่านั้นไปใช้งาน

XML parser ส่วนมากสามารถทำงานได้ทั้ง 2 ลักษณะ ทั้งนี้ขึ้นอยู่กับงานประยุกต์ที่ต้องการจะใช้ข้อมูลว่าต้องการข้อมูลในลักษณะใด การทำงานลักษณะแรกคือ SAX แบบที่สองคือ DOM

๑. SAX (Simple API for XML)

Parser ที่ทำงานในลักษณะ SAX นี้จะทำการอ่านเอกสาร XML ที่จุดเริ่มต้นและ parser จะทำการเรียกกลับไปยังโปรแกรมประยุกต์ที่ต้องการข้อมูล (Client application) ทุกครั้งที่ parser พบส่วนของเอกสารที่มีความแตกต่างเด่นชัด เช่นพบ tag เริ่มต้น พบ tag สิ้นสุด ต่อจากนั้นก็จะอ่านเอกสารต่อไปเรื่อย ๆ จนกระทั่งจบเอกสาร วิธีการนี้เป็นการทำ parsing ที่เร็วที่สุดเนื่องจากไม่สูญเสียหน่วยความจำในการเก็บข้อมูลทั้งหมดที่ได้จากการอ่านเอกสาร ซึ่งเหมาะแก่การใช้งานกับเอกสาร XML ที่มีขนาดใหญ่ที่ไม่สามารถอ่านทั้งหมดมาเก็บไว้ในหน่วยความจำในคราวเดียวได้ และเหมาะกับชุดข้อมูลที่ไม่มีความเกี่ยวข้องกันหรือมีความสัมพันธ์กัน ข้อเสียเพียงอย่างเดียวของ SAX ก็คือ parser ไม่สามารถบอกความสัมพันธ์ระหว่างวัตถุที่เจอได้ parser จะไม่สามารถบอกได้ว่า node ที่อ่านออกมานี้เป็น child node ของ node ก่อนหน้านี้หรือว่าเป็น node ที่อยู่ในชั้นเดียวกัน (sibling node) ขั้นตอนการทำงานของ SAX สามารถอธิบายได้ดังรูปที่ ๓.๒.๒

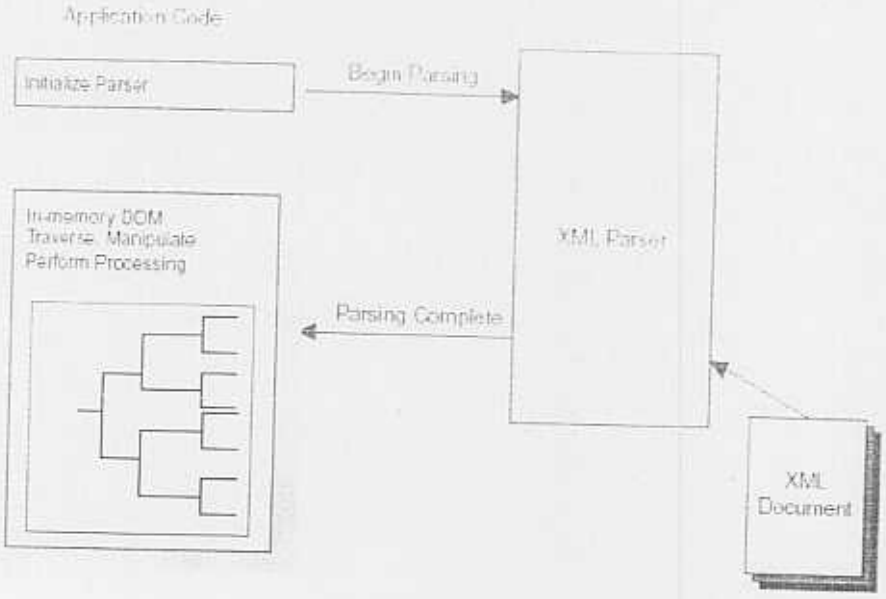


รูป ๓.๒.๒ การทำงานของ SAX

๒. **DOM (Document Object Model)**

DOM เป็นอีกวิธีการหนึ่งของ parser ในการอ่านข้อมูลออกมาจากเอกสาร XML ในการใช้งานลักษณะของ DOM นั้น parser จะทำการอ่านเอกสารทั้งหมดแล้วทำการสร้างโครงสร้างแบบต้นไม้ (tree) เก็บข้อมูลเหล่านั้นแล้วส่งค่าผลลัพธ์ของตำแหน่งของรากบนสุดของต้นไม้เพื่อให้ผู้ใช้งานสามารถท่องลงไปตามต้นไม้ได้อย่างสะดวก ดังนั้นโปรแกรมประยุกต์ผู้ใช้งานก็สามารถที่จะจัดการกับข้อมูลทั้งหมดในเอกสารได้ รวมถึงการปรับเปลี่ยน node การเพิ่มและลบ เนื้อหาได้ตามที่ต้องการ

ในขณะที่การใช้งาน DOM ทำให้ผู้ใช้งานสะดวกในการจัดการกับข้อมูลทั้งเอกสาร แต่ DOM ก็มีข้อเสียคือทำงานได้ช้าเนื่องจากต้องอ่านเอกสารทั้งหมดก่อนจึงสามารถทำการประมวลผลได้ ทั้งนี้ก็กินทรัพยากรเป็นอย่างมาก เนื่องจากต้องใช้หน่วยความจำที่มีขนาดใหญ่ในการเก็บข้อมูลที่เป็นลักษณะโครงสร้างแบบต้นไม้ ดังนั้นการใช้ DOM จึงเหมาะกับเอกสาร XML ที่มีขนาดไม่ใหญ่มากนัก การทำงานของ DOM สามารถอธิบายได้ดังรูปที่ ๓.๒.๓



รูป ๓.๒.๓ การทำงานของ DOM API

JAXP API สนับสนุนการทำงานทั้งสองลักษณะ ซึ่งก็ขึ้นอยู่กับงานประยุกต์ที่พัฒนาอยู่ว่ามีความต้องการอย่างไร มีข้อจำกัดในเรื่องความเร็ว หรือเรื่องของการจัดการข้อมูลทั้งเอกสาร หรือเรื่องของขนาดของเอกสาร

๓.๒.๓ เอกสาร PMML

PMML หรือ Predictive Model Markup Language คือเอกสารรูปแบบหนึ่งของ XML ที่ใช้อธิบายกฎความเกี่ยวเนื่องของข้อมูล การทำเหมืองข้อมูล และข้อมูลสถิติ โดยมีข้อดีคือ PMML เป็นเอกสารที่ใช้งานข้ามแพลตฟอร์ม และไม่ขึ้นกับชนิดหรือแบรนด์ของโปรแกรมประยุกต์ใด ๆ เนื่องจากเป็นเอกสาร XML ชนิดหนึ่งนั่นเอง โดยที่มีการพัฒนางานที่ใช้ PMML ในการสนับสนุนการทำงาน ได้แก่ JSR 73 และ SQL/MM Part 6: Data Mining เป็นต้น

PMML ปัจจุบันเป็เวอร์ชั่น 3.0 (http://www.dmg.org) โดยมีการใช้งานอย่างแพร่หลายทั้งในวงการธุรกิจไฟแนนซ์ การตลาด การผลิต และรวมถึงทางทหาร PMML ถูกพัฒนาโดย Data Mining Group (DMG) ซึ่งประกอบด้วยผู้ผลิตงานประยุกต์เกี่ยวกับการทำเหมืองข้อมูลหลากหลายบริษัท ได้แก่ Angoss Software Corp., IBM Corp., Insightful Corp., Magnify, Inc., Microsoft, และอื่น ๆอีกมากมาย

รูปแบบหรือไวยากรณ์ของ PMML เป็นลักษณะเดียวกับ XML ทั้งสิ้น เนื่องจาก PMML เป็นส่วนหนึ่งของ XML เช่นเดียวกับ MathML หรือ FIXML ที่มีเพื่อใช้งานเกี่ยวกับทางคณิตศาสตร์และทางการเงินตามลำดับ ตัวอย่างเอกสาร PMML ดังรายการที่ ๓.๒.๔

```
<?xml version="1.0" encoding="UTF-8"?>
<PMML version="2.1" xmlns="http://www.dmg.org/PMML-2_1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<Header copyright="Copyright (c) Integral Solutions Ltd., 1994-
2004. All rights reserved.">
```

```
<Application name="Clementine" version="9.0"/>
<Annotation>Exported with PMML extensions for use with SPSS
SmartScore</Annotation>
</Header>
<DataDictionary numberOfFields="2">
<DataField name="cardid" optype="continuous" dataType="integer">
<Extension name="x-storageType" value="numeric"
extender="spss"/>
</DataField>
<DataField name="Product" optype="categorical"
dataType="string">
<Extension name="x-storageType" value="string" extender="spss"/>
<Value value="beer" property="valid"/>
<Value value="cannedmeat" property="valid"/>
<Value value="cannedveg" property="valid"/>
<Value value="confectionery" property="valid"/>
<Value value="dairy" property="valid"/>
<Value value="fish" property="valid"/>
<Value value="freshmeat" property="valid"/>
<Value value="frozenmeal" property="valid"/>
<Value value="fruitveg" property="valid"/>
<Value value="softdrink" property="valid"/>
<Value value="wine" property="valid"/>
</DataField>
</DataDictionary>
<AssociationModel modelName="cardid &amp;amp; Product"
algorithmName="Carma" functionName="associationRules"
numberOfTransactions="939" minimumSupport="0.177848775292865"
minimumConfidence="0.32013201320132" numberOfItems="7"
numberOfItemsets="10" numberOfRules="18">
<MiningSchema>
<MiningField name="cardid" usageType="group"/>
<MiningField name="Product" usageType="active"/>
</MiningSchema>
<Item id="5" value="wine"/>
<Item id="7" value="fruitveg"/>
<Item id="4" value="confectionery"/>
<Item id="3" value="frozenmeal"/>
<Item id="6" value="fish"/>
<Item id="2" value="cannedveg"/>
<Item id="1" value="beer"/>
<Itemset id="9" numberOfItems="1" support="0.31096912">
<ItemRef itemRef="6"/>
</Itemset>
<Itemset id="1" numberOfItems="2" support="0.17784878">
<ItemRef itemRef="1"/>
<ItemRef itemRef="2"/>
</Itemset>
<Itemset id="8" numberOfItems="1" support="0.30457934">
```

```
<ItemRef itemRef="5"/>
</Itemset>
<Itemset id="5" numberOfItems="2" support="0.18423855">
<ItemRef itemRef="2"/>
<ItemRef itemRef="3"/>
</Itemset>
<Itemset id="10" numberOfItems="1" support="0.31842386">
<ItemRef itemRef="7"/>
</Itemset>
<Itemset id="4" numberOfItems="1" support="0.32268371">
<ItemRef itemRef="2"/>
</Itemset>
<Itemset id="3" numberOfItems="2" support="0.18104366">
<ItemRef itemRef="1"/>
<ItemRef itemRef="3"/>
</Itemset>
<Itemset id="7" numberOfItems="1" support="0.29392971">
<ItemRef itemRef="4"/>
</Itemset>
<Itemset id="2" numberOfItems="1" support="0.32161874">
<ItemRef itemRef="3"/>
</Itemset>
<Itemset id="6" numberOfItems="1" support="0.31203408">
<ItemRef itemRef="1"/>
</Itemset>
<AssociationRule x-id="1" support="0.15548456" confidence="0.874251497005988"
antecedent="1" consequent="2" x-lift="2.718285283737161"/>
<AssociationRule x-id="2" support="0.15548456" confidence="0.8588235294117649"
antecedent="3" consequent="4" x-lift="2.6615026208503174"/>
<AssociationRule x-id="3" support="0.15548456" confidence="0.84393063583815"
antecedent="5" consequent="6" x-lift="2.704610467754342"/>
<AssociationRule x-id="4" support="0.18104366" confidence="0.5802047781569 97"
antecedent="6" consequent="2" x-lift="1.8040141943358283"/>
<AssociationRule x-id="5" support="0.18423855" confidence="0.5728476821192 05"
antecedent="2" consequent="4" x-lift="1.7752606386466425"/>
<AssociationRule x-id="6" support="0.18423855" confidence="0.5709570957095 71"
antecedent="4" consequent="2" x-lift="1.775260638646646"/>
<AssociationRule x-id="7" support="0.17784878" confidence="0.5699658703071 67"
antecedent="6" consequent="4" x-lift="1.7663298753083467"/>
<AssociationRule x-id="8" support="0.18104366" confidence="0.5629139072847 68"
antecedent="2" consequent="6" x-lift="1.804014194335825"/>
<AssociationRule x-id="9" support="0.17784878" confidence="0.5511551155115 51"
antecedent="4" consequent="6" x-lift="1.7663298753083483"/>
<AssociationRule x-id="10" support="0.15335463" confidence="0.5217391304347 83"
antecedent="7" consequent="8" x-lift="1.7129826695044081"/>
<AssociationRule x-id="11" support="0.15335463" confidence="0.5034965034965 04"
antecedent="8" consequent="7" x-lift="1.7129826695044097"/>
```

```
<AssociationRule x-id="12" support="0.15548456" confidence="0.498293515358362"
antecedent="6" consequent="5" x-lift="2.7046104677543497"/>
<AssociationRule x-id="13" support="0.1544196" confidence="0.496575342465753"
antecedent="9" consequent="10" x-lift="1.5594790855362624"/>
<AssociationRule x-id="14" support="0.1544196"
confidence="0.48494983277591996" antecedent="10" consequent="9" x-
lift="1.5594790855362641"/>
<AssociationRule x-id="15" support="0.15548456" confidence="0.483443708609272"
antecedent="2" consequent="1" x-lift="2.7182852837371603"/>
<AssociationRule x-id="16" support="0.15548456" confidence="0.481848184818482"
antecedent="4" consequent="3" x-lift="2.661502620850314"/>
<AssociationRule x-id="17" support="0.10330138" confidence="0.339160839160839"
antecedent="8" consequent="4" x-lift="1.051062798587549"/>
<AssociationRule x-id="18" support="0.10330138"
confidence="0.32013201320131995" antecedent="4" consequent="8" x-
lift="1.0510627985875496"/>
</AssociationModel>
</PMML>
```

รายการ ๓.๒.๔ ตัวอย่างเอกสาร PMML

๓.๒.๔ ความรู้เบื้องต้นเกี่ยวกับการเก็บเกี่ยวข้อมูล (Data mining)

กล่าวโดยย่อแล้วการเก็บเกี่ยวข้อมูลหมายถึงกระบวนการสรุปข้อมูลที่ต้องการจากกลุ่มข้อมูลใหญ่ที่รวบรวมมาได้ โดยกระบวนการดังกล่าวสามารถแบ่งออกเป็น 3 ประเภทได้แก่

๑. การจัดกลุ่มข้อมูล (Clustering /Classification) ซึ่งเป็นกระบวนการวิเคราะห์ข้อมูลและสร้างกฎแบ่งกลุ่มที่นำไปใช้ในการจัดกลุ่มสำหรับข้อมูลใหม่ ๆ ที่ได้รับ เช่นการจัดกลุ่มของโรคภัยไข้เจ็บโดยใช้การวิเคราะห์จากอาการที่ให้มาว่าควรจะเป็นไข้หวัด หรือ เป็นโรคซาร์ส เป็นต้น

๒. การหากฎความสัมพันธ์ (Association Rules) เป็นกระบวนการหาความสัมพันธ์ระหว่างชุดของข้อมูลในฐานข้อมูลในหลายระดับ เช่น ผู้ป่วยที่ตัวร้อนมากกว่า 38 องศาและเป็นไข้หวัด 90 เปอร์เซ็นต์ จะเป็นโรคซาร์สด้วย เป็นต้น

๓. การวิเคราะห์หาความสัมพันธ์ที่เกิดเป็นลำดับ (Sequential Analysis) เป็นกระบวนการค้นหารูปแบบในฐานข้อมูลที่เกิดขึ้นเป็นลำดับ เช่น ผู้ป่วยเป็นโรคหวัดมักจะเป็นไข้ในวันถัดมาเป็นต้น

ข้อมูลที่ใช้ในงานวิจัยนี้เป็นข้อมูลที่ได้แบ่งเป็นกลุ่มชัดเจนโดยจะอยู่ในรูปแบบของ XML การค้นหา association rule ของข้อมูลจึงเป็นงานหลักของการวิจัยนี้ โดย association rule มีนิยามว่า association rules สามารถเขียนให้อยู่ในรูป $X \Rightarrow Y$ เมื่อกำหนดเซตของ item $I = \{i_1, i_2, i_3, ..., i_m\}$ เมื่อ $X \subset I, Y \subset I$ และ $X \cap Y = \phi$ โดย

$X \Rightarrow Y$ จะเป็นจริงในเซตของ transaction D ด้วยความน่าเชื่อถือ c (***confidence***) ถ้ามี transaction $c\%$ ใน D ที่ประกอบไปด้วยทั้ง X และ Y

$X \Rightarrow Y$ จะถือว่ามี ***transaction support*** $= s$ ใน D ถ้ามี transaction $s\%$ ใน D มี $X \cup Y$

เมื่อกำหนด transaction set D ให้ ปัญหาของการค้นหา association rule คือการค้นหากฎความสัมพันธ์ทั้งหมดที่มี support มากกว่าค่า support ที่น้อยที่สุดที่กำหนดไว้เรียกว่า minimum support หรือ *minsup* และค่า confidence มีค่ามากกว่าค่า confidence ที่น้อยที่สุดที่กำหนดไว้เรียกว่า minimum confidence หรือ *minconf*

๓.๒.๔.๑ กระบวนวิธีที่ใช้ในงานวิจัย (Algorithm)

Algorithm ที่ใช้ในงานวิจัยนี้เรียกว่า Apriori algorithm ซึ่งนำเสนอโดยศูนย์วิจัย Almaden ของบริษัท IBM ในรายงานเรื่อง "Fast Algorithms for Mining Association Rules" ซึ่งได้ทำการทดสอบแล้วว่ามีประสิทธิภาพมากกว่า AIS algorithm และ SETM algorithm โดยมีรายละเอียดดังนี้

ขั้นตอนการย่อยปัญหาการค้นหา association rule

๑. ค้นหาเซตของ item เรียกว่า itemsets ที่มี transaction support มากกว่า ***minsup*** เรียก itemsets เหล่านี้ว่าว่า ***large itemset*** ส่วน itemset ที่มี transaction support น้อยกว่า ***minsup*** จะเรียกว่า small itemset

๒. นำ large itemset มาหา association rule โดยใช้แนวคิดที่ว่าถ้า $ABCD$ และ AB เป็น large itemset เราจะได้ association rule $AB \Rightarrow CD$ เมื่อ $conf = \text{support}(ABCD) / \text{support}(AB) \geq minconf$

๓.๒.๔.๒ Association Rule และ Apriori Algorithm

Association Rule หรือกฎความเกี่ยวเนื่องของข้อมูล คือ กฎที่ใช้สำหรับบอกความสำหรับชิ้นส่วนข้อมูล(data items) แต่ละชิ้นส่วน ว่าชิ้นส่วนใด ขึ้นอยู่กับชิ้นส่วนใด โดย X=>Y หมายถึงถ้า X แล้ว Y ใช้ในการอธิบายว่า หาก X, Y คือสินค้าในร้านค้าแล้ว เมื่อลูกค้าซื้อ X จะมีการซื้อ Y ด้วยเสมอ การค้นหากฎความเกี่ยวเนื่องของข้อมูลมีองค์ประกอบที่สำคัญดังแสดงในตารางที่ ๓.๒.๑

Term	Description
D	Database of Transactions
t_i	Transaction in D
s	Support
α	Confidence
X,Y	Itemsets
X=>Y	Association Rule
L	Set of large itemsets
l	Large itemset in L
C	Set of candidate itemsets
p	Number of partitions

ตาราง ๓.๒.๑ คำสำคัญที่ใช้ในกฎความเกี่ยวเนื่อง

ตัวอย่างข้อมูลที่ใช้หากฎความเกี่ยวเนื่องของข้อมูลในที่นี้ อาจเป็นข้อมูลการขายสินค้าของร้านค้าเบ็ดเตล็ด ที่มีการขายสินค้า ๕ รายการดังนี้ Bread, Beer, Jelly, PeanutButter และ Milk รายการขายสินค้าที่เกิดขึ้นเรียกว่า Transaction อาจเป็นดังตาราง ๓.๒.๒

Transaction	Items
t_1	Bread, Jelly, PeanutButter
t_2	Bread, PeanutButter
t_3	Bread, Milk, PeanutButter
t_4	Beer, Bread
t_5	Beer, Milk

ตาราง ๓.๒.๒ ตัวอย่างข้อมูลเพื่อใช้อธิบายการหากฎความเกี่ยวเนื่อง

Set	Support
Bread	80
Beer	40
Jelly	20
Milk	40
PeanutButter	60
Beer, Bread	20
Beer, Jelly	0
Beer, Milk	20
Beer, PeanutButter	0
Bread, Jelly	20
Bread, Milk	20
Bread, PeanutButter	60
Jelly, Milk	0
Jelly, PeanutButter	20
Milk, PeanutButter	20
Beer, Bread, Jelly	0
Beer, Bread, Milk	0
Beer, Bread, PeanutButter	0
Bread, Jelly, Milk	0
Bread, Jelly, PeanutButter	20
Bread, Milk, PeanutButter	20
Jelly, Milk, PeanutButter	0
Beer, Bread, Jelly, Milk	0
Beer, Bread, Jelly, PeanutButter	0
Beer, Bread, Milk, PeanutButter	0
Beer, Jelly, Milk, PeanutButter	0
Bread, Jelly, Milk, PeanutButter	0
Beer, Bread, Jelly, Milk, PeanutButter	0

ตาราง ๓.๒.๓ Support of All Sets of Items ของตารางที่ ๓.๒.๒

Support ที่ได้เกิดจากการนับว่า Item(s) นั้นมีการเกิดขึ้นใน Transactions เป็นกี่เปอร์เซ็นต์เมื่อเทียบกับจำนวน Transactions ทั้งหมด ซึ่ง Support นี้จะถูกใช้ในการหา Confidence ของกฎความเกี่ยวเนื่องว่าเป็นเท่าใด เช่น Bread => PeanutButter มี Support เป็น 60% และมี Confidence เป็น 75% ซึ่ง Confidence นี้ได้มาจาก Support (Bread,PeanutButter)/ Support (Bread) คือ 60/80 นั่นเอง ซึ่งกฎความเกี่ยวเนื่องสามารถถูกหามาได้จาก Large Itemsets ซึ่งหมายความว่าถ้า Bread => PeanutButter แล้ว {Bread, PeanutButter}, {Bread} และ {PeanutButter} ต่างก็เป็น Large Itemsets ทั้งสิ้น โดยที่ Large Itemset คือ ชุดข้อมูลหรือ

Itemset ที่มีการพบบ่อยใน Transactions ที่มีค่า support มากกว่าค่า support ที่กำหนดไว้เพื่อหากฎความเกี่ยวเนื่องของข้อมูล ค่า support ที่กำหนดไว้ถูกเรียกว่า Threshold ขั้นตอนวิธีในการหากฎความเกี่ยวเนื่องดังแสดงไว้ในขั้นตอนวิธีที่ ๓.๒.๑

```
Input:
        D  //Database of transactions
        I  //Items
        L  //Large itemsets
        s  //Support
        α  //Confidence
Output:
        R  //Association Rules satisfying s and α
ARGen algorithm:
        R=null;
        For each l in L do
           For each x subset of l such that x != null do
                If support(l)/support(x) >= α then
                     R=R union {x=>(l-x)};
```

ขั้นตอนวิธี ๓.๒.๑ ARGen Algorithm

Apriori Algorithm ถูกนำมาใช้เพื่อหา Large Itemsets เพื่อจะได้มาซึ่งกฎความเกี่ยวเนื่องของข้อมูล ซึ่งมีองค์ประกอบคือ Apriori-Gen Algorithm และ Apriori Algorithm ซึ่งขั้นตอนวิธี Apriori นี้มีแนวคิดพื้นฐานคือ Large Itemsets ของเซ็ตของจำนวนชิ้นส่วนข้อมูล i หาได้จาก Large Itemsets ของเซ็ตของจำนวนชิ้นส่วนข้อมูล i-1 เช่น จากข้อมูลตัวอย่างในตาราง ๓.๒.๒ และ ตาราง ๓.๒.๓ สามารถหา Large Itemsets โดยใช้ Apriori ได้ดังแสดงในตาราง ๓.๒.๔

Pass	Candidates	Large Itemsets
1	{Beer}, {Bread}, {Jelly}, {Milk}, {PeanutButter}	{Beer}, {Bread}, {Milk}, {PeanutButter}
2	{Beer, Bread}, {Beer, Milk}, {Beer, PeanutButter}, {Bread, Milk}, {Bread, PeanutButter}, {Milk, PeanutButter}	{Bread, PeanutButter}

ตาราง ๓.๒.๔ Large Itemsets ที่หาได้จากการใช้ Apriori Algorithm

ความสัมพันธ์ของ Large Itemsets ที่ได้สามารถแสดงได้ดังแผนภาพในรูปที่ ๓.๒.๔ จะเห็นได้ว่า Large Itemsets จะเกิดจากความสัมพันธ์ระหว่าง Large Itemset กับ Large Itemset เท่านั้น

```
Input:
      I       //Itemsets
      D       //Database of transactions
      S       //Support
Output:
      L       //Large itemsets
Apriori-gen algorithm:
      k=0;  //k is used as the scan number.
      L=null;
      C1=I1 //Initial candidates are set to be the items.
repeat
      k=k+1;
      Lk=null;
      for each I1∈Ck do
      Ci=0; // Initial counts for each itemset are 0.
      for each tj∈D do
        for each Ii∈Ck do
          if Ii∈tj then
              Ci=Ci+1;
      for each Ii∈Ck do
        If Ci≥(sx|D|)do
      Lk=Lk∪Ii;
      L=L∪Lk;
      Ck+1=Apriori-Gen(Lk)
until Ck+1=null;
```

ขั้นตอนวิธี ๓.๒.๓ Apriori Algorithm

บทที่ ๔
ผลการทดลอง

จากการทดลองสร้างเอกสาร XML เพื่อทำการทดสอบการแยกองค์ประกอบและข้อมูลของเอกสาร XML ออกมาโดยใช้ Java Technology โดยการสร้างเอกสาร XML และ เอกสาร DTD สำหรับเอกสาร XML ดังกล่าว ซึ่งได้แสดงไว้ในรายการที่ ๖.๒.๑ และ ๖.๒.๒ จากนั้นจึงทำการเขียนโปรแกรมจาวาเพื่ออ่านข้อมูลออกมาจากเอกสาร XML โดยทำการทำลองทั้งในลักษณะ SAX และ DOM

๔.๑ การทดลองใช้ SAX

ทำการเขียน Handler เพื่อใช้งานกับ SAX ดังรายการที่ ๔.๑.๑

```
import org.xml.sax.Attributes;
import org.xml.sax.SAXException;
import org.xml.sax.helpers.DefaultHandler;

public class ProductEventHandler extends DefaultHandler
{
        char[] info;

        public ProductEventHandler()
        {
           info = new char[100];
        }

        public void startDocument()
        {
           System.out.println("start document");
        }

        public void startElement(String namespaceURI, String
local_name, String q_name, Attributes atts) throws SAXException
        {
           System.out.println("start element:"+q_name);
        }

        public void characters(char[] info, int start, int end)
        {
           System.out.print("CONTENT:");
           System.out.println(new String(info, start, end));
        }

        public void endElement(String namespaceURI, String
local_name, String q_name) throws SAXException
        {
           System.out.println("end element:"+q_name);
        }
```

```
        public void endDocument()
        {
           System.out.println("end document");
        }
}
```

รายการที่ ๔.๑.๑ ProductEventHandler.java

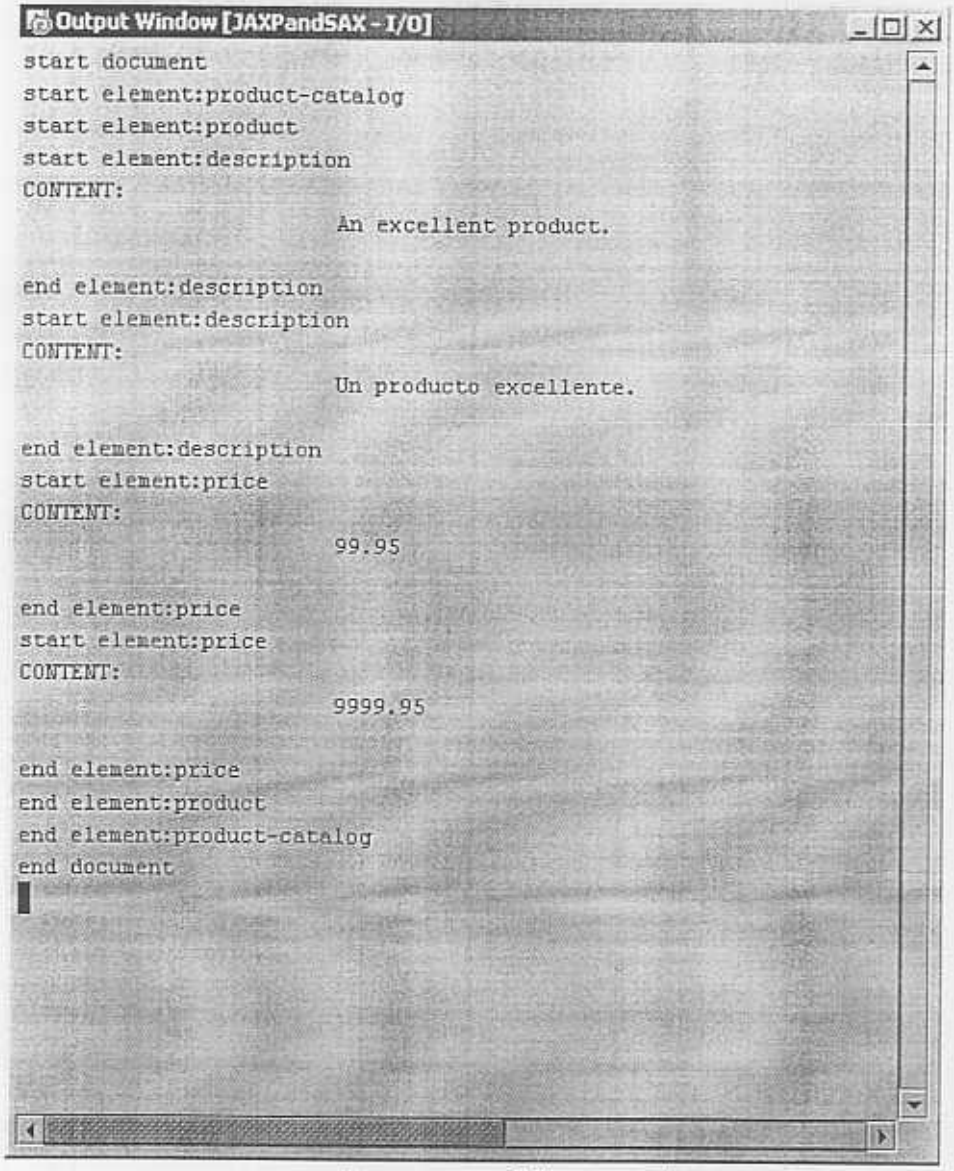
เมื่อทำการทดลองโดย implement หลาย ๆ method ได้แก่ `startDocument`, `startElement`, `characters`, `endElement` และ `endDocument` โดย `startDocument` method นั้นจะทำการแสดงข้อความว่า "start document" ออกที่หน้าจอเมื่อ parser ได้ทำการอ่านเอกสารไปถึงจุดเริ่มต้นของเอกสาร ส่วน characters method นั้นให้ทำการพิมพ์เนื้อหาของ element นั้น ๆออกมาที่หน้าจอ method อื่น ๆ นั้นก็จะทำในลักษณะที่คล้ายกันกับ `startDocument`

เมื่อทำการเขียนโปรแกรมโดยเลือกใช้งาน parser แบบ SAX โดยทำการเขียนโปรแกรมดังรายการที่ ๔.๑.๒

```
import javax.xml.parsers.*;
import java.io.File;

public class JAXPandSAX
{
        public static void main(String[] arg)
        {
           ProductEventHandler handler = new
ProductEventHandler();
           try
           {
                SAXParserFactory factory =
SAXParserFactory.newInstance();
                SAXParser parser = factory.newSAXParser();
                File product_file = new File("product-
catalog.xml");
                parser.parse(product_file, handler);
           }
           catch (Exception ex)
           {
                System.out.println(ex.getMessage());
                ex.printStackTrace();
           }
        }
}
```

รายการ ๔.๑.๒ JAXPandSAX.java

เมื่อทำการคอมไพล์ทั้ง Handler และ JAXandSAX.java หลังจากทำการประมวลผลแล้วก็จะได้ผลดังรูป ๔.๑.๑



รูป ๔.๑.๑ ผลการใช้งาน SAX อ่านเอกสาร XML

จากผลการทดลองจะเห็นว่า parser ทำการอ่านเอกสารเมื่อพบ tag หรือ content ก็จะติดต่อกลับมาที่ Handler แล้วทำการแสดงผลตามที่กำหนดไว้ใน Handler ออกที่หน้าจอ เมื่อตรวจสอบกับเอกสาร XML จะพบว่ามีลักษณะเดียวกันกับพฤติกรรมของมนุษย์เวลาอ่านเอกสารจากบรรทัดแรกจนบรรทัดสุดท้ายของเอกสารนั่นเอง

๔.๒ การทดลองใช้ DOM

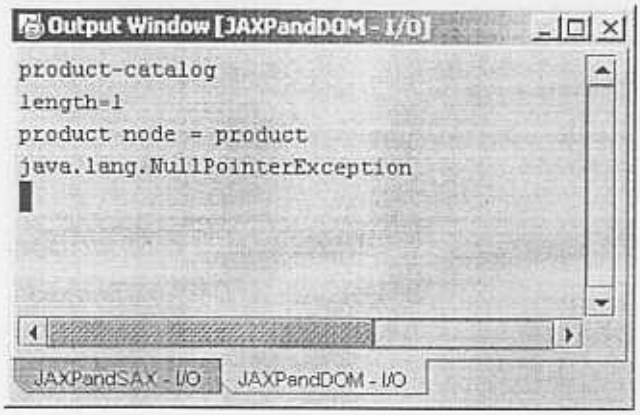
ทำการเขียนโปรแกรมโดยใช้งาน parser ในลักษณะของ DOM ดังรายการที่ ๔.๒.๑

```
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.*;
import org.w3c.dom.*;
```

```
import java.io.File;

public class JAXPandDOM
{
        public static void main(String[] arg)
        {
           try
           {
                DocumentBuilderFactory factory =
DocumentBuilderFactory.newInstance();
                DocumentBuilder builder =
factory.newDocumentBuilder();
                File xmlfile = new File("product-catalog.xml");
                Document document = builder.parse(xmlfile);

                Element root = document.getDocumentElement();
                System.out.println(root.getTagName());
                NodeList rootlist =
root.getElementsByTagName("product");

        System.out.println("length="+rootlist.getLength());
                Node productNode = rootlist.item(0);
                System.out.println("product node =
"+productNode.getNodeName());
                NodeList childlist = productNode.getChildNodes();

                Node node = null;
                for (int i=0; i<childlist.getLength(); i++)
                {
                     node = childlist.item(i);
                     //if (node.getNodeType() ==
Node.ELEMENT_NODE)
                     //{

        System.out.println(node.getFirstChild().getNodeValue());
                     //}
                          //
        System.out.println(node.getNodeName()+" =
"+node.getNodeValue());
                }
           }
           catch (Exception ex)
           {
                System.out.println(ex);
           }
        }

}
```

รายการ ค.ต.๑ JAXPandDOM.java

จะได้ผลการรันโปรแกรมดังรูป



รูป ๔.๒.๑ รูปแสดงผลการทดลองการใช้งาน DOM

การทำงานปัจจุบันกำลังทำการทดลองใช้งาน JDOM และ JAXB เนื่องจาก JAXB เป็นเทคโนโลยีที่ออกมาได้ไม่นานทำให้เอกสารและข้อมูลยังไม่มากนัก จึงทำให้เกิดการล่าช้าในงานวิจัยขึ้น อย่างไรก็ตามจากผลการทดลองสามารถสรุปได้ว่า Java สามารถกลั่นกรองข้อมูลออกมาจากเอกสาร XML ได้ด้วยวิธีการที่ได้ทำการเสนอไป เพื่อให้เหมาะสมกับงานทางเหมืองข้อมูล การใช้งาน DOM อาจเป็นการเหมาะสมกว่า SAX เนื่องจากมีการเก็บข้อมูลเป็นลักษณะของโครงสร้างแบบต้นไม้ ทำให้สามารถจัดการกับ node ต่าง ๆได้ง่ายและสะดวกกว่า ผลที่ได้จากการอ่านแบบ SAX

๔.๓ ผลการทดลองการทำเหมืองข้อมูล

เมื่อทำการทดลองเขียนโปรแกรมเพื่อหา Large Itemsets จากข้อมูลใน transactions ตัวอย่างที่มีอยู่ แล้วทำการหากฎความเกี่ยวข้องโดยข้อมูลที่ใช้ทดสอบเป็นข้อมูลที่เก็บไว้ในไฟล์ในรูปแบบที่นำไปใช้งานในลักษณะ Vector ได้ โดยทำการทดสอบด้วยเอกสาร XML ตัวอย่าง ดังแสดงในรายการที่ ๔.๓.๑

```
<?xml version="1.0"?>
<customer-transactions>
        <transaction tid="t001" cid="c002">
          <product pid="p004" items="1" category="hardware"
price="29,900">
               Computer
          </product>
          <product pid="p008" items="1" category="software"
price="24,000">
               Window XP
          </product>
        </transaction>
        <transaction tid="t001" cid="c002">
          <product pid="p004" items="1" category="hardware"
price="29,900">
               Computer
```

```
            </product>
            <product pid="p008" items="1" category="software"
price="24,000">
                Window XP
            </product>
            <product pid="p002" items="1" category="hadware"
price="4500">
                Printer
            </product>
        </transaction>
        <transaction tid="t002" cid="c030">
            <product pid="p004" items="1" category="hardware"
price="39,900">
                Computer
            </product>
            <product pid="p008" items="1" category="software"
price="24,000">
                Window XP
            </product>
            <product pid="p002" items="1" category="hadware"
price="4,500">
                Printer
            </product>
            <product pid="p011" items="1" category="software"
price="17,000">
                Microsoft Office XP
            </product>
        </transaction>
        <transaction tid="t002" cid="c030">
            <product pid="p004" items="1" category="hardware"
price="39,900">
                Computer
            </product>
            <product pid="p008" items="1" category="software"
price="24,000">
                Window XP
            </product>
            <product pid="p002" items="1" category="hadware"
price="4,500">
                Printer
            </product>
        </transaction>
        <transaction tid="t003" cid="c102">
            <product pid="p004" items="1" category="hardware"
price="29,900">
                Computer
            </product>
            <product pid="p002" items="1" category="hadware"
price="8,000">
                Printer
            </product>
```

```
            <product pid="p010" items="1" category="software"
price="170">
                  Linux Red Hat
            </product>
         </transaction>
</customer-transactions>
```

รายการ ๔.๓.๑ เอกสาร XML ที่ใช้ในการทดลอง

สามารถแสดงในรูปแบบของตารางความสัมพันธ์ระหว่าง Transaction และสินค้าในแต่ละ Transaction ได้ดังตารางที่ ๔.๓.๑

Transaction	Items
t_1	Computer, Windows XP
t_2	Computer, Windows XP, Printer
t_3	Computer, Windows XP, Printer, MS-Office XP
t_4	Computer, Windows XP, Printer
t_5	Computer, Printer, Linux Red Hat

ตาราง ๔.๓.๑ ข้อมูลจากเอกสาร XML ที่จะใช้ทดสอบการหากฎความเกี่ยวเนื่อง

จากทฤษฎีการหากฎความเกี่ยวเนื่อง ทำการนับ Item เพื่อหา Support of All Set of Items ได้ดังตารางที่ ๔.๓.๒

Set	Support
Computer	100
Windows XP	80
Printer	80
MS-Office XP	20
Linux Red Hat	20
Computer, Windows XP	80
Computer, Printer	80
Computer, MS-Office XP	20
Computer, Linux Red Hat	20
Windows XP, Printer	60
Windows XP, MS-Office XP	20
Windows XP, Linux Red Hat	20
Printer, MS-Office XP	20
Printer, Linux Red Hat	20
MS-Office XP, Linux Red Hat	0
Computer, Windows XP, Printer	60
Computer, Windows XP, MS-Office XP	20
Computer, Windows XP, Linux Red Hat	0
Computer, Printer, MS-Office XP	0
Computer, Printer, Linux Red Hat	20
Computer, MS-Office XP, Linux Red Hat	0
Windows XP, Printer, MS-Office XP	20
Windows XP, Printer, Linux Red Hat	0
Printer, MS-Office XP, Linux Red Hat	0
Computer, Windows XP, Printer, MS-Office XP	20
Computer, Windows XP, Printer, Linux Red Hat	0
Computer, Printer, MS-Office XP, Linux Red Hat	0
Computer, Windows XP, Printer, MS-Office XP, Linux Red Hat	0

ตาราง ๔.๓.๒ Support of All Sets of Items ของข้อมูลจากตารางที่ ๔.๓.๑

ทำการหา Large Item Sets โดยใช้ Apriori Algorithm จะได้ผลดังตารางที่ ๔.๓.๓

Pass	Candidates	Large Itemsets
1	{Computer}, {Windows XP}, {Printer}, {MS-Office XP}, {Linux Red Hat}	{Computer}, {Windows XP}, {Printer}
2	{Computer, Windows XP}, {Computer, Printer}, {Windows XP, Printer}	{Computer, Windows XP}, {Computer, Printer}, {Windows XP, Printer}
3	{Computer, Windows XP, Printer}	{Computer, Windows XP, Printer}

ตาราง ๔.๓.๓ แสดง Large Itemsets ที่หาได้จากเอกสาร XML ทดสอบรายการที่ ๔.๓.๑

เพื่อลดการเปรียบเทียบข้อมูลใน transactions ในการทดลองได้ทำการจัดเก็บรายการสินค้าลงใน Vector ของ String และทำการเก็บข้อมูลของ transactions ลงในVector ของ Integer โดยข้อมูลที่อยู่ใน Vector ของ Integer จะใช้จำนวนเต็มแทนลำดับของสินค้าแทนการเก็บชื่อสินค้าซึ่งเป็นข้อความ

ผลจากการรันโปรแกรม จะได้ Large Itemsets คือ {Computer}, {Windows XP} และ {Printer} โดยสามารถตั้งค่า Threshold ของ support ที่ต้องการได้ในการรันโปรแกรมแต่ละครั้ง ซึ่งผลที่ได้ตรงตามทฤษฎีที่ได้แสดงไว้ในตารางที่ ๔.๓.๓ ผลของการรันโปรแกรมโดยกำหนด minimum support เป็น 20% และระดับของ confidence เป็น 30% จะได้กฎของความเกี่ยวเนื่องแสดงดังนี้ โดย { 0 } implies { 1 } หมายถึง ถ้าลูกค้าซื้อ Computer แล้วจะซื้อ Windows XP ด้วย ส่วน { 1 } implies { 0 }เป็นกฎของความเกี่ยวเนื่องในทางกลับกันของกฎข้างต้น ส่วนของโปรแกรมแสดงไว้ในภาคผนวก ก.

ผลที่ได้จากการหากฎความเกี่ยวเนื่องถูกนำไปเป็น Output ของโปรแกรมในรูปแบบของเอกสาร PMML โดยจะได้เอกสาร PMML ดังแสดงในรายการที่ ๔.๓.๒

```
<?xml version="1.0" encoding="UTF-8"?>
<PMML version="2.1" xmlns="http://www.dmg.org/PMML-2_1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
     <Header copyright="by Ubon Rajathanee University">
          <Application name="Jubu-Miner" version="1.0"/>
          <Annotation>This is an output of Jubu-
Miner</Annotation>
     </Header>

     <DataDictionary numberOfFields="2">
          <DataField name="Product" optype="categorical"
dataType="string">

               <Value value="Computer" property="valid"/>
               <Value value="Window XP" property="valid"/>
               <Value value="Printer" property="valid"/>
               <Value value="Microsoft Office XP"
property="valid"/>
               <Value value="Linux Red Hat" property="valid"/>
          </DataField>
     </DataDictionary>

     <AssociationModel modelName="Product"
algorithmName="Apriori" functionName="ARGen"
numberOfTransactions="5" minimumSupport="0.2"
minimumConfidence="0.3" numberOfItems="5" numberOfItemsets="6"
numberOfRules="18">
          <MiningSchema>
               <MiningField name="Product" usageType="active"/>
          </MiningSchema>

          <Item id="0" value="Computer"/>
          <Item id="1" value="Window XP"/>
          <Item id="2" value="Printer"/>
          <Item id="3" value="Microsoft Office XP"/>
          <Item id="4" value="Linux Red Hat"/>

          <Itemset id="0" numberOfItems="1" support="1.0">
```

```
            <ItemRef itemRef="0"/>
        </Itemset>
        <Itemset id="1" numberOfItems="1" support="0.8">
            <ItemRef itemRef="1"/>
        </Itemset>
        <Itemset id="2" numberOfItems="1" support="0.8">
            <ItemRef itemRef="2"/>
        </Itemset>
        <Itemset id="3" numberOfItems="2" support="0.6">
            <ItemRef itemRef="1"/>
            <ItemRef itemRef="2"/>
        </Itemset>
        <Itemset id="4" numberOfItems="2" support="0.8">
            <ItemRef itemRef="0"/>
            <ItemRef itemRef="2"/>
        </Itemset>
        <Itemset id="5" numberOfItems="2" support="0.8">
            <ItemRef itemRef="0"/>
            <ItemRef itemRef="1"/>
        </Itemset>

        <AssociationRule x-id="0" support="0.8"
confidence="0.8" antecedent="0" consequent="1"/>
        <AssociationRule x-id="1" support="0.8"
confidence="1.0" antecedent="1" consequent="0"/>
        <AssociationRule x-id="2" support="0.8"
confidence="0.8" antecedent="0" consequent="2"/>
        <AssociationRule x-id="3" support="0.8"
confidence="1.0" antecedent="2" consequent="0"/>
        <AssociationRule x-id="4" support="0.6"
confidence="0.75" antecedent="1" consequent="2"/>
        <AssociationRule x-id="5" support="0.6"
confidence="0.75" antecedent="2" consequent="1"/>
        <AssociationRule x-id="6" support="0.6"
confidence="0.6" antecedent="0" consequent="3"/>
        <AssociationRule x-id="7" support="0.6"
confidence="0.75" antecedent="1" consequent="4"/>
        <AssociationRule x-id="8" support="0.6"
confidence="0.75" antecedent="2" consequent="5"/>
    </AssociationModel>
</PMML>
```

รายการ ๔.๓.๒ เอกสารผลลัพท์ที่แสดงในรูปแบบ PMML

บทที่ ๕
การวิจารณ์ผล

๕.๑ ข้อสรุปของงานวิจัย

จากการวิจัยที่ได้ ทำให้สามารถอ่านเอกสาร XML ด้วย SAX แล้วทำการหากฎความเกี่ยวเนื่องของข้อมูลในเอกสาร XML แล้วทำการแสดงผลโดยเอกสาร PMML ได้ดังต้องการ ทำให้สามารถนำเอกสาร PMML ไปใช้งานต่อได้โดยสะดวก

๕.๒ คำชี้แจงเกี่ยวกับปัญหาและอุปสรรค

การดำเนินการวิจัยมีความล่าช้าในช่วงที่ผ่านมาเนื่องจากสาเหตุหลายประการเช่น

๑. ข้อมูลที่ได้จากการอ่านเอกสาร XML ยังไม่สามารถจัดให้อยู่ในรูปแบบที่นำไปใช้ใน Apriori Algorithm ได้ ซึ่งทำให้นำไปทดลองใช้งานจริงไม่ได้ โดยภายหลังทำการจัดเก็บใน Vector ของ String และ Vector ของ Integer ทำให้สะดวกต่อการจัดการยิ่งขึ้น

๒. ส่วนของ Apriori Algorithm สามารถค้นหา Association Rule ได้แต่ยังต้องการการปรับแต่งเพื่อประสิทธิภาพของโปรแกรมและรูปแบบให้สมบูรณ์ ภายหลังได้ทำการปรับปรุงให้สามารถแสดงผลให้เข้าใจได้ง่ายและ เขียนออกไปที่เอกสาร PMML ด้วย

บรรณานุกรม

1. Agrawal, R., Mannila, H., Srikant, R., Toivonen, H. and Verkamo, A. 1996. Fast Discovery of Association Rules, Advances in Knowledge Discovery and Data Mining, AAAI press, pp.307-328.
2. Buechner A., Baumgarten M., Mulvenna M., Boehm R., Anand, S. 2000. Data Mining and XML: Current and Future Issues, Computer society, IEEE.
3. Chawathe, S. 1999. Describing and Manipulating XML Data, IEEE Computer Society Technical Committee on Data Engineering
4. Cooley R., Mobasher B., Srivastava J. 1999, Data Preparation for Mining World Wide Web Browsing Patterns, Journal of Knowledge and Information Systems, 1(1)
5. Dunham, M. H. 2003. Data Mining Introductory and Advanced Topics, Pearson Education, New Jersey, USA
6. Fayyad U., Piatetsky-Shapiro G. and Smyth P. 1996. From Data Mining to Knowledge Discovery: An Overview, Advances in Knowledge Discovery and Data Mining, AAAI press, pp.1-36.
7. Ferguson N. 1999. A Data Mining Tool for XML Document, http://www.dcs.warwick.ac.uk/~csufh/project/
8. Gabrick, K. A., Weiss, D. B.,2002. J2EE and XML,Manning,USA
9. Mercer, D., 2001, XML A Beginner's Guide, Osbourne, USA
10. Specifications of Java APIs for XML from http://java.sun.com
11. Specifications and tutorials from http://www.oasis-open.org/cover/pmml.html
12. Specifications and tutorials from http://www.w3c.org
13. Tutorials from http://www.kdnuggets.com
14. David RR Webber, 1999, XML/EDI directions, presentation in San Jose'99

ภาคผนวก ก
รายการซอร์สโค้ด

//Apriori.java

```
import io.DBReader;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.Properties;
import java.util.Vector;

import test.Util;
import xml.SAXReader;
import xml.pmml.ARSet;
import xml.pmml.PMMLWriter;

public class Apriori
{
     static String DEFAULT_PRODUCT_FILE=
"res/db/products.GEN";
     static String DEFAULT_TRANSACTION_FILE=
"res/db/transactions.GEN";
     static float MIN_CONFIDENCE = 1, MAX_CONFIDENCE = 100;
     static String pfile, tfile, xml_file, output_file;
     static long productSize = 20; // read from
DEFAULT_PRODUCT_FILE
     static float minsupport = 20; // 20%
     static float confidence= 20; // 20%
     static boolean debug = false;


     public static void main(String[] arg) throws
IOException
     {
          System.out.println("Make sure you've adjusted
'conf/apriori.properties', before running.");
          readProperties("conf/apriori.properties");

          //debug =
System.getProperty("apriori.debug").equals("off");

          if ( debug )
          {
               //print info
               System.out.println("*************
Debugging Mode 'On' ***********");
               System.out.println("--- Using ---");
               System.out.println("product_file="+pfile);
```

```
        System.out.println("transaction_file="+tfile);

        System.out.println("minimum_support="+minsupport);

        System.out.println("confidence_level="+confidence);
             }

           if ( xml_file == null )
           {
               productSize =
DBReader.readProductSize(pfile);
               Vector<String> products =
DBReader.readProducts(pfile);
               Util.printProducts(products);
               Vector<Vector<Integer>> d =
DBReader.readTransactions(tfile);
               Vector<Vector<Integer>> l1=
genLargeItemSet(d);
               Vector<Vector<Integer>> largeItemset =
genLargeItemset(l1, d, minsupport);
               if ( debug )
               {
                   Util.printSets(largeItemset);
                   System.out.println("Calling ARGen(...)");
               }
               ARSet arset = ARGen(d, largeItemset,
minsupport, confidence);
               PMMLWriter.write(output_file, products,
arset, d.size(), minsupport, confidence);
               if ( debug ) System.out.println("************
Done ************");
           }
           else
           {

               SAXReader saxreader = new SAXReader();
               saxreader.read(xml_file);
               Util.printProducts(saxreader.getProducts());
               Vector<Vector<Integer>> l1=
genLargeItemSet(saxreader.getTransactions());
               Vector<Vector<Integer>> largeItemset =
genLargeItemset(l1, saxreader.getTransactions(),
minsupport);
               if ( debug )
               {
                   Util.printSets(largeItemset);
                   System.out.println("Calling ARGen(...)");
               }
```

```
            ARSet arset =
ARGen(saxreader.getTransactions(), largeItemset,
minsupport, confidence);
            PMMLWriter.write(output_file,
saxreader.getProducts(), arset,
                    saxreader.getTransactions().size(),
minsupport, confidence);
            if ( debug ) System.out.println("************
Done ************");
        }
    }


    public static void readProperties(String fileName)
throws IOException, FileNotFoundException
    {
        File file = new File(fileName);
        if ( file.exists() )
        {
            Properties prop = new Properties();
            prop.load(new FileInputStream(file));
          xml_file = prop.getProperty("xml_file");
          output_file =
prop.getProperty("output_file");
            if ( xml_file == null )
            {
                pfile = prop.getProperty("product_file");
                if ( pfile == null ) pfile =
DEFAULT_PRODUCT_FILE;
                tfile =
prop.getProperty("transaction_file");
                if ( tfile == null ) tfile =
DEFAULT_TRANSACTION_FILE;
            }
            try
            {
                minsupport =
Integer.parseInt(prop.getProperty("minimum_support"));
                confidence =
Float.parseFloat(prop.getProperty("confidence_level"));
            } catch (Exception ex) {}
        }
        minsupport =
(minsupport<0)?0:(minsupport>100)?100:minsupport;
        minsupport /= 100;
        confidence =
(confidence<0)?MIN_CONFIDENCE:(confidence>100)?MAX_CONFIDENC
E:confidence;
        confidence /= 100;
    }
```

```
    public static Vector<Vector<Integer>>
genLargeItemSet(Vector<Vector<Integer>> d)
    {
        //large 1-itemset
        Vector<Vector<Integer>> l1=new
Vector<Vector<Integer>>();

        int count = 0;
        for (int i=0; i<productSize; i++)
        {
            count = 0;
            for (Vector<Integer> transaction: d)
                if ( transaction.contains(i) )
                    count++;
            if ( count > minsupport*d.size() )
            {
                Vector<Integer> tmp = new
Vector<Integer>();
                tmp.add(i);
                l1.add(tmp);
            }
        }

        return l1;
    }

    /*
    *<PRE>
    *Intput:
    *    D(atabase)
    *    I(tems)
    *    L(arge Itemsets)
    *    s(support)
    *    alpha (confidence)
    *Output:
    *   (R) set of implications of sets.
    *ARGen:
    *    R = null;
    *    for each l in L do
    *        for each x subset of l such that x not
null do
    *            if ( support(l)/support(x) >= alpha
then
    *                R = R union { x implies (l-x) };
    *</PRE>
```

```
     */
     //why 's' is needed?
     public static ARSet ARGen(Vector<Vector<Integer>> D,
Vector<Vector<Integer>> L, float s, float alpha)
     {
          Vector<Vector<Integer>> subset;
        ARSet arset = new ARSet();

          if ( debug )
               System.out.println("Large Item Set: size()
= "+L.size());

          for (Vector<Integer> l : L)
          {
               //????? shouldn't large itemset contain
only elements(set) of size > 1 already?
               if ( l.size() > 1)
               {
                    long supportOfl = supportOf(l, D);
                float support =
(float)supportOfl/D.size();
                    if ( debug )
                    {
                         System.out.println();
                         Util.printSet(l);
                         System.out.println();
                         System.out.println("supportOfl =
"+supportOfl);
                    }
                    //for each subset of l: assuming |l|
is small.
    //if ( (float)supportOfl/D.size() >= s ) NOTE: 'l' is
already a large itemset
    //{
                    subset = genSubset(l);
                    for (Vector<Integer> x: subset)
                    {
                        long supportOfx = supportOf(x,
D);
                        float sx =
(float)supportOfx/D.size();
                        if ( debug )
                        {
                            System.out.println();
                            Util.printSet(x);
                            System.out.println();
                            System.out.print("supportOfx
count = "+supportOfx);
```

```
                                System.out.print("\t
support="+sx);
                                System.out.println("\talpha =
"+alpha);
                            }
                            float confidence =
(float)supportOfl/(float)supportOfx;
                            if ( confidence >= alpha )
                            {
                                Util.printSet(x);
                                System.out.print(" implies
");
                                Vector<Integer> tmp =
complement(l, x);
                                Util.printSet(tmp);
                                System.out.println();

                                arset.addRule(x, tmp, sx,
(float)supportOf(tmp, D)/D.size(), support, confidence);
                            }
                        }
    //}
                }
            }
        return arset;
    }

    /*
    *set of 'l-x', all element in 'l' not in 'x'.
    */
    public static Vector<Integer>
complement(Vector<Integer> l, Vector<Integer> x)
    {
        Vector<Integer> tmp = new Vector<Integer>();
        for (int i: l)
            if ( !x.contains(i) )
                tmp.add(i);

        return tmp;
    }

    /*
    *generate all subset of 'l'.
    */
    public static Vector<Vector<Integer>>
genSubset(Vector<Integer> l)
    {
        Vector<Vector<Integer>> subset = new
Vector<Vector<Integer>>();
```

```
            Vector<Vector<Integer>> sizeisubset = new
Vector<Vector<Integer>>();
            for (int i: l)
            {
                Vector<Integer> tmp = new
Vector<Integer>();
                tmp.add(i);
                sizeisubset.add(tmp);
                subset.add(tmp);
        }

            for (int i=2; i<l.size()-1; i++)
            {
                sizeisubset =
genGreaterSubset(sizeisubset, i);
                for (Vector<Integer> element: sizeisubset)
                    subset.add(element);
            }

            return subset;
        }

        /*
        *given all subset of size 'size-1' find all subset of
size 'size'
        */
        public static Vector<Vector<Integer>>
genGreaterSubset(Vector<Vector<Integer>> s, int size)
        {
            Vector<Vector<Integer>> subset = new
Vector<Vector<Integer>>();
            for (int i=0; i<s.size()-1; i++)
            {
                for (int j=i; j<s.size(); j++)
                {
                    //unionSet(s.elementAt(i),
s.elementAt(j));
                    Vector<Integer> tmp = new
Vector<Integer>(s.elementAt(i));
                    for (int k:s.elementAt(j))
                        if ( !tmp.contains(k) )
                            tmp.add(k);

                    if ( tmp.size() == size )
                        subset.add(tmp);
                }
            }

            return subset;
        }
```

```
     /*
     *count support of 'l' in database 'd'.
     */
     public static long supportOf(Vector<Integer> l,
Vector<Vector<Integer>> d)
     {
          long count = 0;
          for (Vector<Integer> t: d)
               if ( containsIn(l, t) )
                    count++;

          return count;
     }

     /*
     *<PRE>
     *Input:
     *    Ln (large itemset)
     *    n,
     *Output:
     *    Ci
     *Apriori-Gen:
     *    Ci = null;
     *    for each I in Li-1 do
     *         for each J != I in Li-1 do
     *              if ( i-2 of the elements in I and J
are equals then
     *                   Ck = Ck union { I union J };
     *    </PRE>
     */
     public static Vector<Vector<Integer>>
genCandidates(Vector<Vector<Integer>> Ln, int n)
     {
          Vector<Vector<Integer>> cn = new
Vector<Vector<Integer>>();
          for (int i=0; i<Ln.size(); i++)
          {
               for (int j=i+1; j<Ln.size(); j++)
               {
                    if ( Util.equals(Ln.elementAt(i),
Ln.elementAt(j), n-1))
                    {
                         Vector<Integer> iunionj = new
Vector<Integer>(Ln.elementAt(i));
                         for (int elementj :
Ln.elementAt(j))
```

```
                                if (
!(iunionj.contains(elementj)) )
                                     iunionj.add(elementj);

                           if ( !elementOf(iunionj, cn) )
                                cn.add(iunionj);
                      }
                 }
           }

           return cn;
      }

      /*
      *check if set 'e' is an element of 'set'
      */
      public static boolean elementOf(Vector<Integer> e,
Vector<Vector<Integer>> set)
      {
           for (Vector<Integer> s: set)
                if ( Util.equals(s, e, e.size()) )
                     return true;

           return false;
      }


      /**
      *<pre>
      *<b><u>Apriori-Gen Algo:</u></b>
      *<b>Input</b>:
      *    l(temsets)
      *    D
      *    s (support)
      *Output:
      *    L // large itemset
      *Apriori algo:
      *    k = 0;
      *    l = null;
      *    c=i;
      *    repeat
      *         k = k + 1;
      *         lk = null;
      *         for each ii in ck do
      *              ci = 0; // count
      *         for each tj in d do
```

```
     *              for each ii in ck do
     *                   if ( ii in tj then
     *                        ci++;
     *         for each ii in ck do
     *              if ci >= (s times d.size()) do
     *                   lk = lk union ii;
     *
     *         l = l union lk;
     *         ck+1=apriori-gen(lk);
     *
     *    util Ck+1=null;
     *</pre>
     */
     public static Vector<Vector<Integer>>
genLargeItemset(Vector<Vector<Integer>> l1,
               Vector<Vector<Integer>> d, float s)
     {
          Vector<Vector<Integer>> L = new
Vector<Vector<Integer>>();
          Vector<Vector<Integer>> lk = new
Vector<Vector<Integer>>();
          Vector<Vector<Integer>> ck = l1;
          int k = 0;
          while ( ck.size() > 0)
          {
               lk = new Vector<Vector<Integer>>();
               int[] counts = new int[ck.size()]; //
initialize to zero

               for (Vector<Integer> tj: d)
               {
                    for ( int i=0; i<ck.size(); i++) // ii
                         if ( containsIn(ck.elementAt(i),
tj) )
                              counts[i]++;
               }

               for (int i=0; i<ck.size(); i++)
                    if ( counts[i] >= s*d.size() )
                         lk.add(ck.elementAt(i));

               if ( debug )
               {
                    //Util.printSets(ck);

     System.out.println("ck.size()="+ck.size()+"\tlk.size
()="+lk.size());
```

```
                    Util.printSets(lk);
               }

               /*L = L union lk */
                for (Vector<Integer> l: lk) L.add(l);

               ck = genCandidates(lk, ++k);
          }
          return L;
     }

     /*
     *check if all elements of set 'a' are also elements
of set 'b'.
     */
     public static boolean containsIn(Vector<Integer> a,
Vector<Integer> b)
     {
          for (int elementA: a)
               if ( !(b.contains(elementA)) )
                    return false;

          return true;
     }

}
```

// io/DBReader.java

```
package io;

import java.io.File;
import java.io.IOException;
import java.util.Scanner;
import java.util.StringTokenizer;
import java.util.Vector;

public abstract class DBReader
{
     /*
     protected static Vector<String> products;
     protected static Vector<Vector<Integer>> trans;
     */

     public static long readProductSize(String file)
throws IOException
     {
          Scanner scanner = new Scanner(new File(file));
          long productSize = scanner.nextLong();
          scanner.close();
```

```
            return productSize;
      }

      public static Vector<String> readProducts(String
file) throws IOException
      {
            Scanner scanner = new Scanner(new File(file));
            int size = 0;
            try
            {
                  size = scanner.nextInt();
            } catch (Exception ex) {}
            //System.out.println("size = "+size);
            Vector<String> products = new
Vector<String>(size);
            while (scanner.hasNext())
            {
                  products.add(scanner.nextLine());
            }

            return products;
      }

      public static Vector<Vector<Integer>>
readTransactions(String file) throws IOException
      {
            Vector<Vector<Integer>> trans = new
Vector<Vector<Integer>>();
            Scanner mains = new Scanner(new File(file));
            //Scanner scanner;
            StringTokenizer st;
            while ( mains.hasNext() )
            {
                  //scanner = new Scanner(mains.nextLine());
                  //scanner.setPattern();
                  st = new StringTokenizer(mains.nextLine(),
",");
                  Vector<Integer> v = new Vector<Integer>();
                  while ( st.hasMoreTokens() )
                        v.add(new Integer(st.nextToken()));
                  trans.add(v);
            }

            return trans;
      }
}
```

#apriori.properties

```
# properties files for 'Apriori'

# specify minimum support for generating large item sets (%)
minimum_support=20

# specify confidence level
confidence_level=30

# xml file name
xml_file=res/db/test.xml

# output file name
output_file=output/result.xml

# ---- if xml_file isn't defined the following variables are used ---
#
# specify product file containing list of product names
product_file=res/db/products.GEN

# specify transaction file containing list of transactions
transaction_file=res/db/transactions.GEN
# ----------------------------------------------------------------
#
```

// test/Util.java

```
package test;

import java.util.Vector;

public abstract class Util
{
     public static void printProducts(Vector<String>
products)
     {
          int i=0;
          for (String product:products)
               System.out.printf("%3d=%s\n", i++,
product);
     }

     public static void
printTransactions(Vector<Vector<Integer>> t)
     {
          int i=0;
          for (Vector<Integer> v:t)
          {
               System.out.printf("%3d:",i++);
               for (int j:v)
                    System.out.printf("%3d,", j);
               System.out.printf("\n");
          }
     }
```

```
     /*
      * check if 'n' elements of set 'a' and set 'b' are
equal.
```

//ElementType.java

```
package xml;

public enum ElementType { NONE, TRANSACTION, PRODUCT };
```

//SAXReader.java

```
package xml;

import org.xml.sax.helpers.DefaultHandler;
import org.xml.sax.Attributes;
import javax.xml.parsers.SAXParserFactory;
import java.util.Vector;

import static xml.ElementType.*;

public class SAXReader extends DefaultHandler
{

    private  String product;
    private ElementType type = NONE;
    protected Vector<String> products = new
Vector<String>();
    protected Vector<Vector<Integer>> transactions = new
Vector<Vector<Integer>>();
    private Vector<Integer> trans;

    public Vector<String> getProducts() { return
products; }

    public Vector<Vector<Integer>> getTransactions() {
return transactions; }

    public void read(String file)
    {
        try
        {

SAXParserFactory.newInstance().newSAXParser().parse(new
java.io.File(file), this);
        }
```

```
        catch
(javax.xml.parsers.ParserConfigurationException pcex) {
pcex.printStackTrace(); }
        catch (org.xml.sax.SAXException saxex) {
saxex.printStackTrace(); }
        catch (java.io.IOException ioex) {
ioex.printStackTrace(); }
    }

    // Override methods

    public void startElement(String uri, String
localName, String qname, Attributes attributes)
    {
        //System.out.println("start
element"+localName+":"+qname);
        if ( qname.equals("transaction") )
        {
            type = TRANSACTION;
            trans = new Vector<Integer>();
        }
        else if ( qname.equals("product") )
            type = PRODUCT;
        else
            type = NONE;
    }

    public void characters(char[] carray, int start, int
length)
    {
        switch ( type )
        {
            case TRANSACTION:
                break;
            case PRODUCT:
                // trim() is used to get rid of the
leading 'n trailing new-lines
                product = (new String(carray, start,
length)).trim();
                int index = products.indexOf(product);
                if ( index == -1 )
                {
                    index = products.size();
                    products.add(product);
                }
                trans.add(index);
                type = NONE;
                break;
        }
    }
```

```
    public void endElement(String uri, String localName,
String qname)
    {
        if ( qname.equals("transaction") )
        {
            transactions.add(trans);
        }
        else if ( qname.equals("product") )
        {
            int index = products.indexOf(product);
            assert index==-1: "ERROR: products doesn't
contains '"+product+"'"; // this should NOT be happening
        }
    }
}
```

//ARSet.java

```
package xml.pmml;

import test.Util;
import java.util.Vector;

public class ARSet
{

    protected Vector<AssociationRule> associationRules;
    protected Vector<ItemSet> itemsets;

    public ARSet()
    {
        associationRules = new Vector<AssociationRule>();
        itemsets = new Vector<ItemSet>();
    }

    public void clear()
    {
        associationRules.clear();
        itemsets.clear();
    }

    /*
     * adding association rule x->y to the set.
     */
    public void addRule(Vector<Integer> x,
Vector<Integer> y, float sx, float sy, float s, float c)
    {
       associationRules.add(new AssociationRule(s, c,
addItemSet(x, sx), addItemSet(y, sy)));
    }
```

```
    public Vector<AssociationRule> getMembers() { return
associationRules; }


    protected int addItemSet(Vector<Integer> x, float s)
    {
        int i;
        for (i=0; i<itemsets.size(); i++)
            if (
Util.equals(itemsets.elementAt(i).itemref, x) )
                return i;

        itemsets.add(new ItemSet(x, s));
        return i;
    }

    public Vector<ItemSet> getItemSets() { return
itemsets; }

    public Vector<AssociationRule> getAssociationRules()
{ return associationRules; }
}
```

//AssociationRule.java

```
package xml.pmml;

public class AssociationRule
{
    public float support, confidence;
    public int antecedent, consequent;

    public AssociationRule() {}

    public AssociationRule(float support, float
confidence, int ante, int conseq)
    {
        this.support = support;
        this.confidence = confidence;
        this.antecedent = ante;
        this.consequent = conseq;
    }
}
```

//ItemSet.java

```
package xml.pmml;

import java.util.Vector;
```

```
public class ItemSet
{
    protected float support;
    protected Vector<Integer> itemref;

    public ItemSet()
    {
        itemref = new Vector<Integer>();
    }

    public ItemSet(float s)
    {
        this();
        this.support = s;
    }

    public ItemSet(Vector<Integer> set, float s)
    {
        this.support = s;
        itemref = set;
    }

    public float getSupport() { return support; }

    public Vector<Integer> getItemRef() { return itemref;
}
}
```

//PMMLWriter.java

```
package xml.pmml;

import java.io.PrintWriter;
import java.util.Vector;

public final class PMMLWriter
{
    public static void write(String fileName,
Vector<String> products, ARSet arset,
            int transactionSize, float minsupport, float
minalpha) throws
        java.io.IOException
    {
        PrintWriter pw = new PrintWriter(fileName);

        // name space
        pw.println(
            "<?xml version=\"1.0\" encoding=\"UTF-
8\"?>\n"+
```

```
            "<PMML version=\"2.1\"
xmlns=\"http://www.dmg.org/PMML-2_1\""+
            "
xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\">"
        );

        // header
        pw.println(
            "\t<Header copyright=\"by Ubon Rajathanee
University\">\n"+
            "\t\t<Application name=\"Jubu-Miner\"
version=\"1.0\"/>\n"+
            "\t\t<Annotation>This is an output of Jubu-
Miner</Annotation>\n"+
            "\t</Header>\n"
        );

        // data dictionary
        pw.println(
            "\t<DataDictionary numberOfFields=\"2\">\n"+
            "\t\t<DataField name=\"Product\"
optype=\"categorical\" dataType=\"string\">\n"
        );
        for (String product:products)
            pw.println("\t\t\t<Value
value=\""+product+"\" property=\"valid\"/>");
        pw.println(
            "\t\t</DataField>\n"+
            "\t</DataDictionary>\n"
        );

        //:=START=: association model
        pw.println(
            "\t<AssociationModel modelName=\"Product\""+
            "algorithmName=\"Apriori\"
functionName=\"ARGen\" "+
            "numberOfTransactions=\""+transactionSize+"\"
minimumSupport=\""+minsupport+"\" "+
            "minimumConfidence=\""+minalpha+"\"
numberOfItems=\""+products.size()+"\" "+

"numberOfItemsets=\""+arset.getItemSets().size()+"\"
numberOfRules=\"18\">"
        );
        pw.println(
            "\t\t<MiningSchema>\n"+
            "\t\t\t<MiningField name=\"Product\"
usageType=\"active\"/>\n"+
            "\t\t</MiningSchema>\n"
        );
        for (int i=0; i<products.size(); i++)
```

```
            pw.println("\t\t<Item id=\""+i+"\"
value=\""+products.elementAt(i)+"\"/>");

        pw.println("");
        // ItemSets
        int i=0;
        for (ItemSet itemset: arset.getItemSets())
        {
            pw.println("\t\t<Itemset id=\""+(i++)+"\"
numberOfItems=\""+
                    itemset.getItemRef().size()+"\"
support=\""+itemset.getSupport()+"\">");
            for (int j: itemset.getItemRef())
                pw.println("\t\t\t<ItemRef
itemRef=\""+j+"\"/>");
            pw.println("\t\t</Itemset>");
        }
        pw.println("");
        // AssociationRule
        i=0;
        for (AssociationRule ar:
arset.getAssociationRules())
        {
            pw.println("\t\t<AssociationRule x-
id=\""+(i++)+"\" support=\""+ar.support+
                    "\" confidence=\""+ar.confidence+"\"
antecedent=\""+ar.antecedent+
                    "\"
consequent=\""+ar.consequent+"\"/>");
        }
        pw.println("\t</AssociationModel>");
        //:=END=: association model

        pw.println("</PMML>");
        pw.close();
    }
}
```